BILBOKO INDUSTRIA INGENIARITZA TEKNIKOKO
UNIBERTSITATE ESKOLA
INDUSTRIA INGENIARITZA TEKNIKOA: INDUSTRIA ELEKTRONIKA

KARRERA AMAIERAKO PROIEKTUA

Kontrol esparruan ikasketa eta garapenerako oinarrizko ingurune praktiko, ireki eta askea

discrete_control, PID, real_time, matlab_simulink, xilinx_ise, fpga_spartan3

IKASLEAREN DATUAK	**ZUZENDARIAREN DATUAK**
UNAI MARTÍNEZ CORRAL	Koldo Basterretxea Oyarzabal koldo.basterretxea@ehu.es Teknologia Elektronikoa
SIN.:	SIN.:
DATA: 2013ko apirilaren 12a	DATA:

Gaien Aurkibidea

Irudien Zerrenda

Taulen Zerrenda

BILBOKO INDUSTRIA INGENIARITZA TEKNIKOKO UNIBERTSITATE ESKOLA

INDUSTRIA INGENIARITZA TEKNIKOA: INDUSTRIA ELEKTRONIKA

KARRERA AMAIERAKO PROIEKTUA

Anie - 1. Dokumentua: Memoria

IKASLEAREN DATUAK	**ZUZENDARIAREN DATUAK**
	Koldo Basterretxea Oyarzabal koldo.basterretxea@ehu.es Teknologia Elektronikoa
SIN.: DATA: 2013ko apirilaren 12a	SIN.: DATA:

Gurasoei, eta bihotzeko bigarren ama horri, Martari

"...hor egongo da begira.

Tú no nos diste el euskera,
pero nos diste la vida.
Nos llevaste a la ikastola,
aprendimos enseguida.
Ya es hora de que entiendas
el canto que nos motiva.
Esta txapela es tuya,
ay nire amatxu querida."

Xabier Paya
2006ko Bizkaiko Bertsolari Txapelketa

Gaien Aurkibidea

Irudien Zerrenda

Taulen Zerrenda

Ikurren Zerrenda

Letra larriak

$E(x)$	—	e seinalea x operadorearen menpe.
F_{cl}	—	Doitutako sistemaren banda-zabalera begizta itxian (jarraitua).
F_o	—	Plantaren elikadura seinalearen maiztasuna.
F_p	—	Plantaren banda-zabalera ($-3dB$).
F_{pwm}	—	PWM modulagailuaren erlojuaren maiztasuna.
F_s	—	Laginketa-maiztasuna, ${T_s}^{-1}$.
K	—	Hardwareak edo diseinuak baldintzatutako konstantea.
$K_{p,i,d}$	—	PID kontrolagailuaren irabazi finkoak.
P, I, D	—	*Matlab* lan-eremuan adierazitako PID jarraituaren irabaziak.
S	—	Sistema jarraituak lantzeko domeinu konplexua.
T_c	—	Sistemak kontrol begizta egikaritzeko behar duen denbora tartea.
T_f	—	PID-aren adierazpen jarraitu inpropio osoaren iragaziaren parametroa.
$T_{i,d}$	—	Hurrenez hurren, $\frac{K_p}{K_i}$ eta $K_p \cdot T_d$.
T_s	—	Laginketa-periodo finkoa.
$U(x)$	—	u seinalea x operadorearen menpe.
$U_{p,i,d}$	—	Gainezarmena aplikatuz kontrolagailuaren adar bakoitzeko U-ren osagaia.
$Y(x)$	—	y seinalea x operadorearen menpe.
Z	—	Sistema diskretuak lantzeko domeinu konplexua.

Letra xeheak

bp	—	*binary-point* (koma bitarra).
e	—	Kontrolerako errore-seinale jarraitua: $r - y$.
dc	—	*duty-cycle*, PWM seinalearen lan-erregimena.
k	—	Diskretizatutako seinalearen lagina.
r	—	Kontrol begiztaren erreferentziako seinalea (jarraitua).
s	—	S domeinuko operadorea.
u	—	Kontrolagailuaren irteerako seinalea (jarraitua).
$u_{p,i,d}$	—	Gainezarmena aplikatuz kontrolagailuaren adar bakoitzeko u-ren osagaia.
wl	—	*word-length* (hitzaren luzera).
y	—	Plantaren irteerako seinalea (jarraitua).
z	—	Z domeinuko operadorea.

Letra grekoak

ω	—	Abiadura angeluarra.
Θ	—	Fasea.

Akronimoak

ADC	—	*Analog-to-Digital Converter.*
ADQ	—	*Data AcQuisition.*
APES	—	*AdaPtive Electronic Systems* ikerketa-taldea.
ARM	—	*ARM Holdings* konpainia britaniarraren *RISC* egituran oinarritutako mikroprozesadore familia.
ASM	—	*Algorithmic State Machine.*
CPU	—	*Central Processing Unit.*
DAC	—	*Digital-to-Analog Converter.*
DCM	—	*Digital Clock Manager.*
FPGA	—	*Field Programmable Gate Array.*
FROH	—	*FRactional Order Hold.*
FSM	—	*Finite State Machine.*
HIL	—	*Hardware-In-the-Loop.*
HW	—	*Hardware.*
KAP	—	*Karrera Amaierako Proiektua.*
LCD	—	*Liquid Crystal Display.*
PID	—	*Proportional Integral Derivative.*
PWM	—	*Pulse Width Modulation.*
SW	—	*Software.*
SysGen	—	*Xilinx*en *System Generator for DSPs.*
VGA	—	*Video Graphics Array.*
VHDL	—	*VHSIC Hardware Description Language.*
VHSIC	—	*Very High Speed Integrated Circuit.*
x86	—	*Intel 8086* CPUan oinarritutako mikroprozesadore familia.
ZI	—	*Zirkuitu Integratua.*
ZOH	—	*Zero Order Hold.*

1.
Sarrera

Txosten honek, **U**niversidad del **P**aís **V**asco / **E**uskal **H**erriko **U**nibertsitateko[1] **Bi**lboko **I**ndustria **I**ngeniaritza **T**eknikoko **U**nibertsitate **E**skolan[2], Industria Elektronikan berezitutako Industria Ingeniaritza Teknikoa titulazioaren azken urratsa den *Unai Martinez Corral*ek garatutako Karrera Amaierako Proiektuaren memoria osatzen du.

Proiektuaren hazia, benetako denborak (*real-time*) ezartzen dituen mugak errespetatuz begizta itxian aritzen den kontrolagailuaren garapena helburu, *Koldo Basterretxea Oyarzabal* irakasleak jarri zuen, Teknologia Elektronikoa Departamentuko[3] eta **A**da**P**tive **E**lectronic **S**ystems[4] ikerketa taldeko kide denak; eta berak gainbegiratu ditu oso pixkanaka eman diren urratsak.

1.1. Helburuak

1.1. irudiak adierazten duenez, eta 4A. eranskinean zehaztasunez azaltzen denez, *Anie* proiektua ingurune handiagoa den *Pierre St. Martin*en atala da, eta ondorengoa da helburu nagusia:

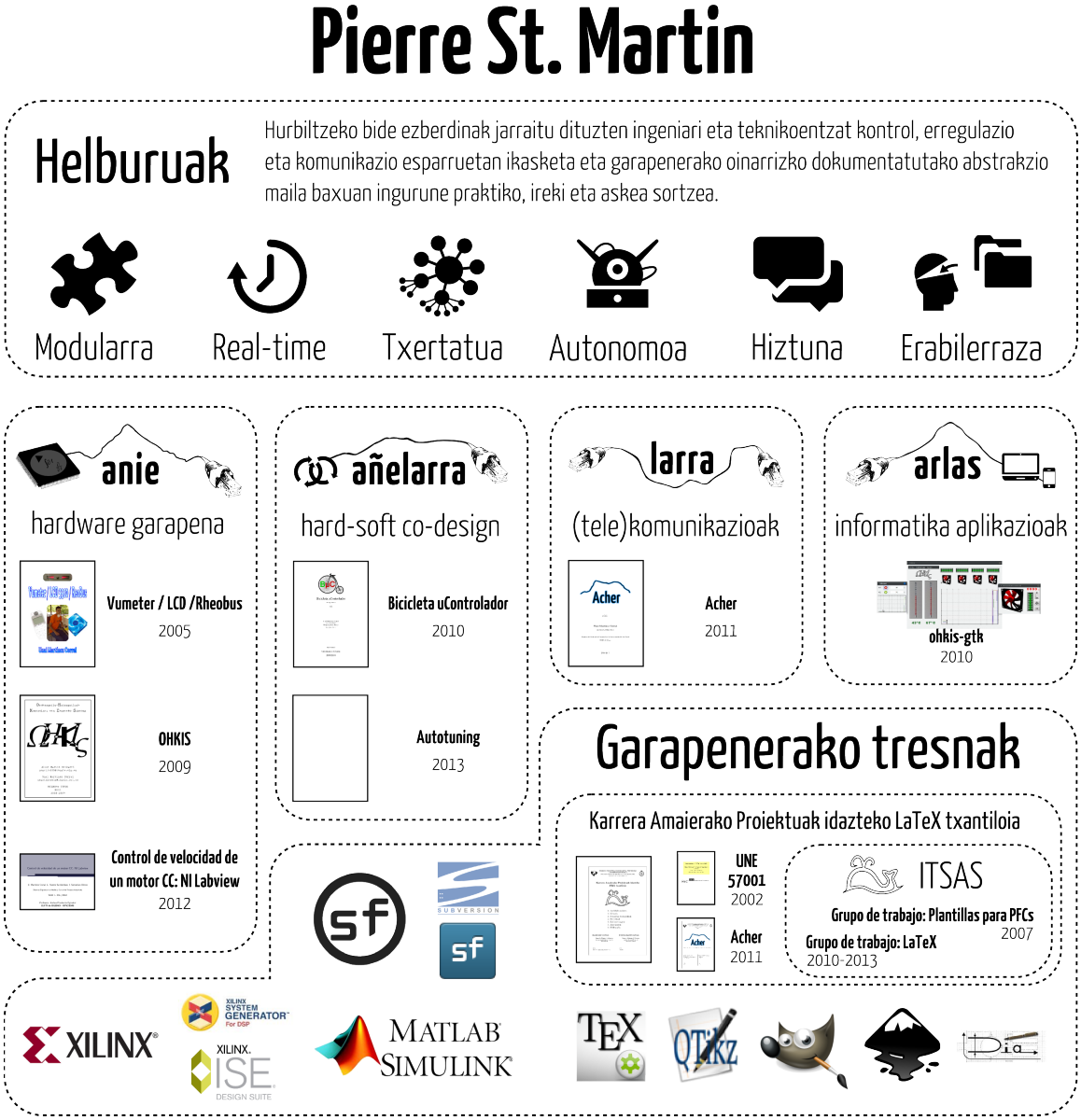
Benetako denborak aginduta, kontrolagailu jarraituaren erantzun baliokidea duen kontrol-sistema digital modularra, txertatua eta autonomoa FPGA batean inplementatzea eta kontrola gauzatzea.

Zehaztasunean behera, prozedura kronologiko teorikoak bigarren mailako hurrengo helburuak ezartzen ditu:

- Korronte zuzeneko motorra [1] azterketa kasu, plantaren modelo ahalik eta egiazkoa lortzea (sarreren/irteeren moduluak barne).
- Kontrolagailua doitzea eta modelo diskretuarekin ezberdintasunak aztertzea.
- Autonomia bermatzeko, monitorizazio eta kontrolerako modulu periferikoak deskribatzea (VGA, LCD, etab.).
- Sistema parametrizatu eta erabilerraza garatzea.
- Arkitektura ezberdinak konparatzea: konbinazional hutsa, erregistroetan oinarritutakoa, VHDL kodea, *System Generator* blokeak, mikrokontrolagailu/mikroprozesadoreak...

Ingurunearen diseinurako baldintzen arabera, erabilera errazteko, ahal den heinean, sistemaren osagai guztiak parametrizatuta daude. Trebakuntza-denbora aurrezteaz gain, hardwarea sakonean ezagutzen ez duenari diseinuan moldaketa txikiak egiteko aukera emateko.

[1] www.ehu.es
[2] www.industria-ingeniaritza-tekniko-bilbao.ehu.es
[3] det.bi.ehu.es
[4] www.ehu.es/apes



1.1. Irudia: Pierre St. Martin *ingurunearen helburu, atal eta tresnak.*

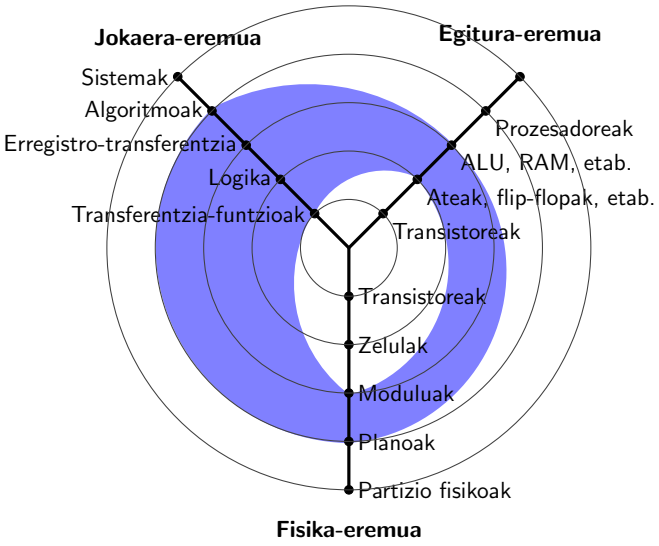
Azterketa eta ulertzea argitzeko, eta proiektuaren izaera praktikoak aginduta, abstrakzio-maila gorenetik hasita eta erregistroetara heldu arte urratsez-urrats azalduko dira aldaketak. 1.2. irudiak adierazten du proiektuaren mamiak zein esparru hartzen duen egituraren, jokaeraren eta inplementazio fisikoaren ikuspuntuetatik.

Badago *Pierre St. Martin* ingurunean zeharkako proiektu bat, informazioaren sostenguak baldintzatzen dituenak: inguruneari dagozkion dokumentazio eta iturri guztiak, ahal den heinean, estandarrak diren formatuetan gordeko dira, ez denboraren poderioz hedatuenetan.

Besteak beste, irakurleak eskuartean duen dokumentua egiteko LaTeX[5] baliatuz Bilboko IITUEen Karrera Amaierako Proiektuak aurkezteko jarraitu beharreko maketazio-argibideak betetzen dituen txantiloia egin da[6].

[5] `itsas.ehu.es/workgroups/latex`

[6] 4B. eranskinean ITSASeko web-gunean LaTeX lan-taldearen *Euskaraz* atalean txantiloiaren txostena dago.



1.2. Irudia: *Gajski-Kuhn Y-grafikoan proiektuaren abstrakzio-maila.*

1.2. Abiapuntua

2005ean, MODPC.com orrialde eta taldeko kideen argibideak jarraituz, Ikastolako Teknologiako irakaslearen babesarekin, bedelaren ezinbesteko laguntzarekin eta familiaren pazientzia tenkatuz, insoladora bat eskuz egin nuen [2]. Horrekin, ordenagailua ikuskatu eta kudeatzeko plaka diseinatu, egin eta muntatu nuen. Disko gogorraren jarduera adierazteko LM3914 ZIak agindutako 20 LED, LPT ataka medio informazioa aurkezteko Nokia 3310 sakelakoaren pantaila, eta haizagailu eta argiak kudeatzeko kommutadore eta erresistentzia aldakorretan oinarritutako zirkuitua biltzen zituen.

2009an, *Koldo Basterretxea Oyarzabal* irakasle nuela, *Eragingailu Logiko Programagarriak Dituzten Sistema Digitalak* ikasgaian *Aitor Martin*ekin batera hurrengo urratsa garatu nuen: Ordenagailu Haizagailuak Kontrolatu eta Ikusteko Sistema (OHKIS) [3]; Spartan3E Starter Kit txartela erabilita lau ordenagailu haizagailu LCD eta VGA bitartez ikuskatu eta begizta irekian PWM irteerak ezartzeko sistema.

2010-2011 bitartean mikrokontrolagailu ezberdinekin garapenak egin nituen Ferroleko Unibertsitate Eskola Politeknikoan[7] egindako *Industria Informatika* ikasgaian eta Bilboko IITUEn *Ordenagailu Pertsonalaren Erabilera Paneleko Instrumentazioan* ikasgaian.

2012an, *Sistemas Digitales en la Medida y Control de Procesos Industriales* ikasgaian *Luis Ranero Santisteban* eta *Iñigo Sarramian Olmos*ekin *National Instruments*en *Labview* softwarea baliatuz benetako denboran lan egiten duen PID kontrolagailuan oinarritutako begizta osatu eta 6 V-eko motorra PWM bitartez kudeatu genuen [4].

Ezinbestean, *Anie/Auñamendi* KAPak aurreko guztiak biltzen ditu. 2009ko OHKIS oinarri, hazia *Koldo Basterretxea*k jarri zuen, berak egin baitzidan garapena ildo honetatik jarraitzeko gonbitea. Era berean, bera izan da azken urte luze hauetan mantso-mantso eman ditudan

[7] `lucas.cdf.udc.es`

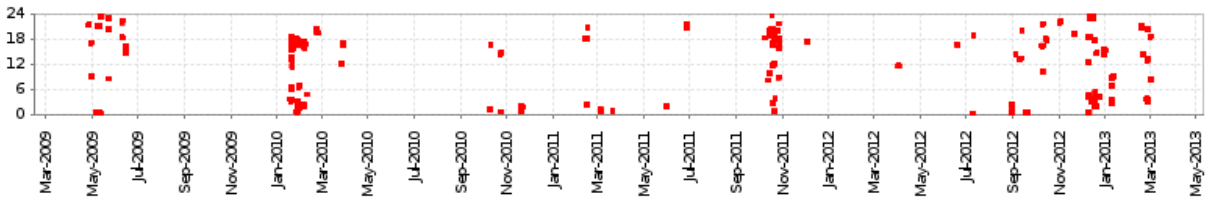
urratsen gainbegiralea, pazientziaren etengabeko apologian.

2008tik hona garatutako proiektu guztietan[8] kideekin lana koordinatzeko, eta nirea antolatu eta planifikatzeko hainbat plataforma erabili izan ditut, esaterako, *sourceforge.net* (ZTPK, OHKIS, ohkis-gtk), *laforja.rediris.es* (BμC), lokalean instalatutako *Subversion* (Acher) edo *Dropbox* (SDM).

*Anie*ren garapenerako *sourceforge.net* erabiltzen da, OHKIS eta ohkis-gtk gordetzen dituen proiektu berdinean (nahiz eta web-gune nagusian adierazgarri berezirik ez egon):

ohkis.sourceforge.net

Sostengua ematen duen *Allura*[9] plataformak esparru berdinean web-gunea, fitxategien bertsio-kontrola egiteko *Git* edo *Subversion*, eztabaida-guneak, tiket-kudeatzailea, wikia, etab. biltzen ditu.



1.3. Irudia: *Garapenean erabilitako* Subversion *zerbitzarira egindako bidalketak (*commit - ci*).*

Benetako denborak ezartzen dituen mugak errespetatuz begizta itxian aritzen den kontrolagailuaren garapena helburu zuen lana Galiziatik bueltan hasi nuen, 2010eko irailean. 1.3. irudian argi ikus daiteke beste proiektuetan emandako denborak garapenean izan duen eragina. Urte luze batez metodologia barneratzen eta ezagutza-gabeziak osatzen eman ostean, azken bederatzi hilabetetan pilatu dira kodearen garapena eta dokumentazioa egitea.

Bi urte baino gehiagoren buruan proiektuak hainbat moldaketa izan ditu, eta bitartean egin izandakoek izaera orraztu dute. Desagertzear dagoen nik egindako ikasketa-planean *Sistema Digitalak* lerroa hiru ikasgaiek osatzen dute:

1. Eragingailu Logiko Programagarriak Dituzten Sistema Digitalak
2. Ordenagailu Pertsonalaren Erabilera Paneleko Instrumentazioan
3. Sistemas Digitales en la Medida y Control de Procesos Industriales

Txostenetan idatzitakoa alde batera utzita, ikaslearentzat azkeneko bi ikasgaietan landutakoen arteko erlazioa zuzena eta argia da. Bietan erabiltzen da *Labview* eta erraz antzematen

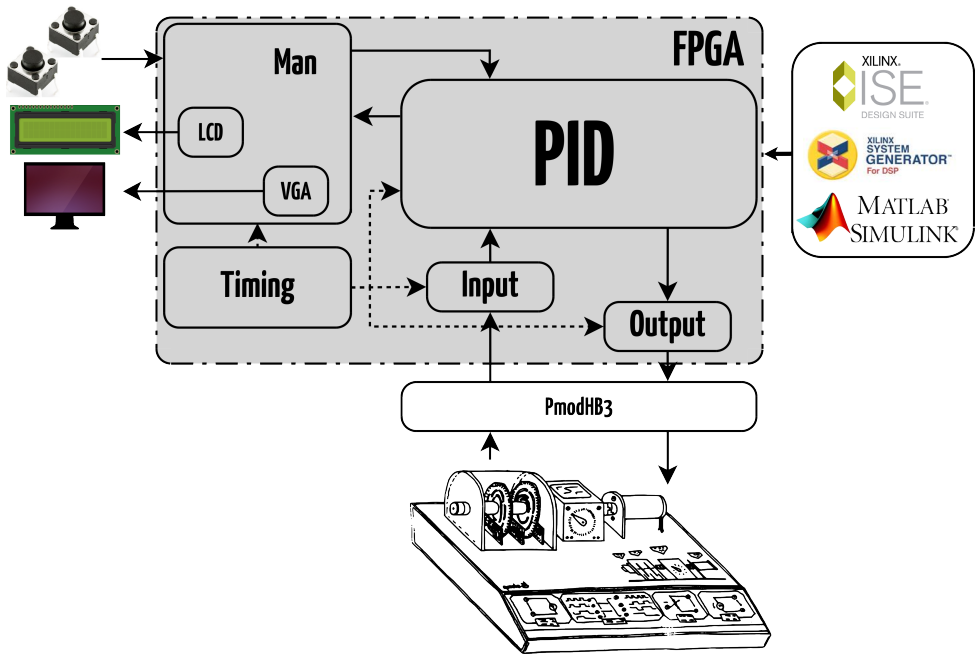
[8] Ikus 4A. eranskina.
[9] sourceforge.net/projects/allura

dira instrumentazio birtualaren eta kontrol-teoriaren arteko lotuneak. Oso abstrakzio-maila altuko ingurunea izanik, ordea, hardwarearekiko erreferentzia-zuzena galtzen da, fedea lantzera behartuz.

Proiektu honek lehenengoaren eta hirugarrenaren arteko zubia izan nahi du, benetako denboran aritzeko mugak maila baxuan aztertzeko aukera emanez, eta datuen kudeaketan egin beharreko aukeraketa kritikoak plazaratuz.

Aldi berean, *Pierre St. Martin*en baitan, gehiegizko espezializazioaren ondorioz hezkuntzan zentzugabeko oztopoa deritzodan gabezia bete nahi du: sistema elektroniko digitaletan ingeniaria izan nahi duen ikasleak aldi berean FPGAk, mikrokontrolagailuak, mikroprozesadoreak eta elkarren arteko komunikazioa maila baxuan ulertu eta kudeatzeko gai izatea.

1.3. Deskribapen orokorra



1.4. Irudia: *Sistemaren osagai nagusiak eta elkarren arteko funtsezko informazio-fluxuak.*

1.4. irudiak adierazi bezala, sistema bost oinarrizko osagaitan deskribatu da funtzionaltasunaren arabera:

- **Timing:** txartelaren 50 MHz-eko erloju seinalean oinarrituta, FPGAk dituen DCMak eta kontagailuak erabilita, beste osagaiek behar dituzten maiztasunetako seinale periodikoak sortzea.
- **Man:** erabiltzaileak sistema kudeatu (erreferentzia-seinalea sortzea) eta aldagaiak ikuskatzeko interfazeen kudeaketa (sakagailuak, kodetzaile birakaria, LCD, VGA, etab.).

- **Input:** plantaren kontrolatu nahi den aldagaia irakurtzea eta kontrolagailuaren berrelikadura seinalea sortzea.
- **PID:** kontrolagailua.
- **Output:** PIDaren irteeraren arabera eragingailuaren agindu seinalea sortzea (DAC, PWM, etab.)

Plantaren arabera sarrerako/irteerako osagai ezberdinak inplementatzea beharrezkoa bada ere, azterketa kasu guztietako sistemek hainbat osagai komun dute: aukeratutako txartelaren erlojuan oinarritutako maiztasun ezberdinak sortzea, esaterako. Hori dela eta, azterketa kasua edozein dela ere moldaketa errazteko, plantarekiko menpekotasunik ez dituztenak (*PID*, *Man -LCD* eta *VGA* barne- eta *Timing*) banaturik landu dira. Azterketa kasuak garatzean planta zehatzek behar dituzten gehikuntzak egin dira, izaera modularrari eskerrak.

1.4. Metodologia eta erabilitako tresnak

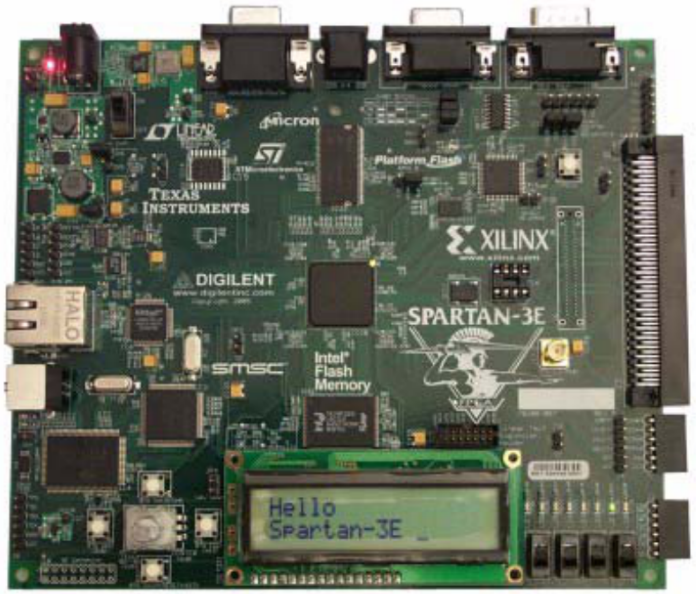
1. Begizta osatzen duten elementuen modelo matematikoak aztertzea, aukeratzea eta parametroak identifikatzea.
2. Konputagailuz burututako simulazio numerikoaz sistema osoaren modeloa balidatzea eta PID kontrolagailu jarraitua doitzea.
3. PID kontrolagailu diskretuaren egitura aukeratzea eta laginketa-maiztasunaren ikasketa burutzea.
4. Arkitektura ezberdinetan kalkuluak koma finkoan egiteko adierazpenak aukeratzea.
5. Bit-zehatza eta ziklo-zehatza den simulazio ingurunean modelo ezberdinen erantzunak konparatzea.
6. Sintesirako, mapa egiteko eta diseinua inplementatzeko software ingurune integratuan PID kontrolagailua VHDL lengoaian deskribatzea.
7. Sistema osatzeko, egoera finituko makinetan (*Máquina de Estados Finitos, MEF, edo Finite State Machine, FSM*) eta erregistroetan oinarritutako modulu periferikoak deskribatzea: kodetzaile inkrementalbirakariak, PWM modulagailua, H-zubia babesteko logika, VGA, LCD...
8. Sistema balidatzea
   (a) *HW/SW co-simulation* (*blackbox*).
   (b) *HW-in-the-loop* (*HIL*) (*blackbox*).
   (c) Sistema erreala balidatzea (parametroen identifikazioa, doiketa eta azkeneko frogak).

Sistema osoa *Digilent*en[10] **Spartan3E Starter Kit** txartel bakarrean inplementatu baino lehen, azterketa arindu eta errazteko, simulazioan oinarritu da garapena; eta begizta osatzen duten elementuak identifikatu eta modelatu dira horretarako.

Modeloak *Laplace*ren S domeinuan baliatuta, *Mathworks*en[11] **Matlab/Simulink** ingurunean PID kontrolagailua doitu da. Modelo jarraitutik abiatuta, eta beti erreferentzia- izanik, kontrolagailuaren egitura diskretua eskuratu da, Z planoan adierazita. Modelo digitala osatzeko laginketa-maiztasuna eta koma finkoko aritmetikak agindutako hitz-luzerak ezarri dira. *Xilinx*en[12] **System Generator** bitartez, *Matlab* eta **ISE/ISim** lotuz, liburutegiko blokeak zein idatzitako VHDL kodea erabilita deskribatutako kontrolagailuen erantzuna aztertu da. Erreferentziarekiko erantzunen konparaketa oniritzitakoan, tresna berdinak medio, hardware ko-simulazioa burutu da helburuko plataformarekin.

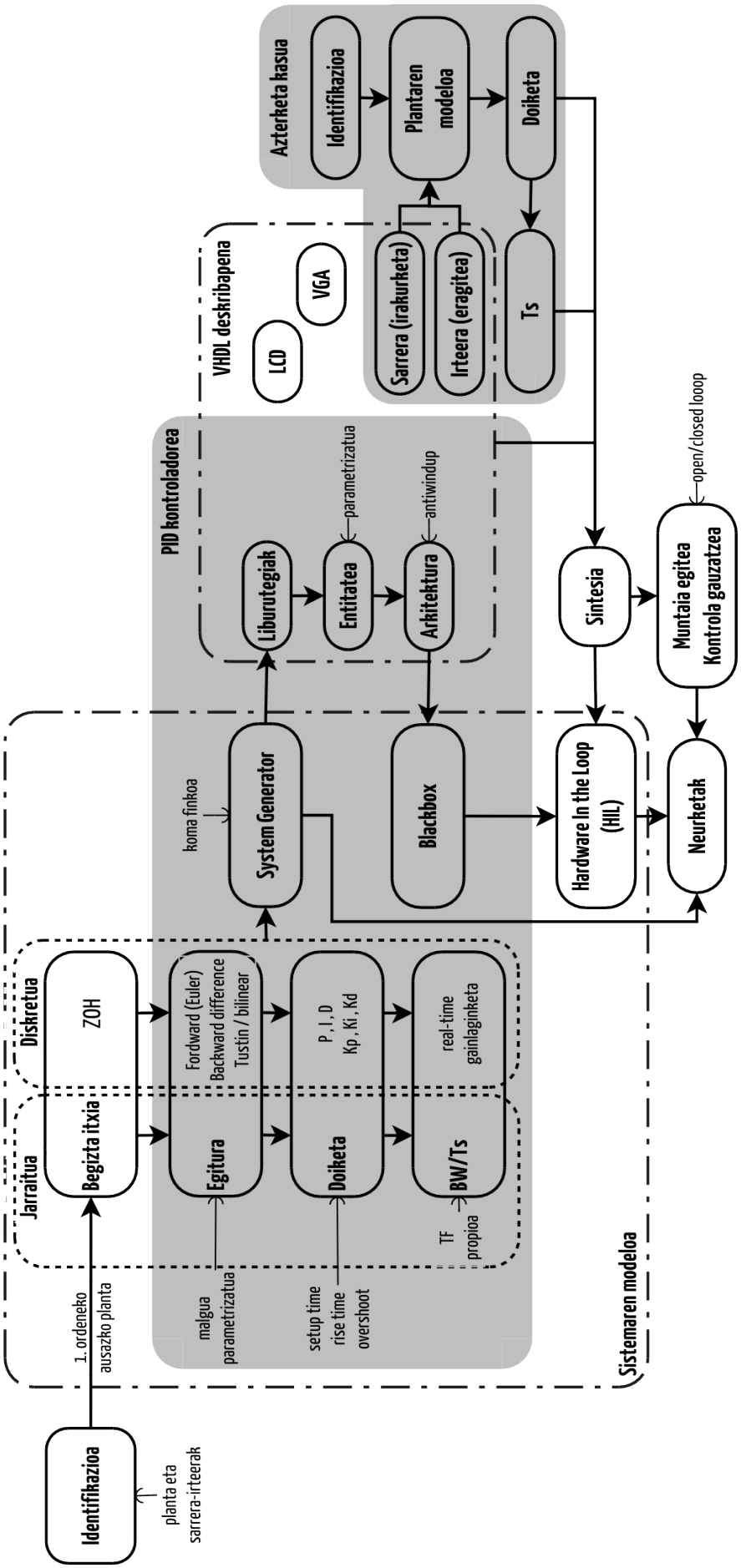
Sistemak helburuak bete ditzan plantarekiko menpekotasunik ez duten moduluak VHDL lengoaian deskribatu dira (LCD, VGA, sakagailuak, kommutadoreak, etab.). Azterketa kasuaren, *Alecop, S. Coop*ren **MV541** maketaren, araberako sarrerako/irteerako osagaiak egin dira azkenik, eta plantaren araberako doiketa eta parametrizazioa ezarri. Hauekin, alde batetik oinarrizko sistema (VGA gabekoa) eta bestetik aukera guztiak dituena sintetizatu dira.

Inplementazioek ezarritako helburuak eta azterketa teorikoan adierazitako mugak betetzen dituztela baieztatu ostean, eta praktikan kontrola zuzena izanik, emaitzak aztertu dira (hala nola, erabilitako azalera eta prozesatze abiadura mugak). Azkenik, *System Generator*en kodea zuzenean sortzeko aukeraren inplementazioaren emaitza zuzenean VHDL lengoaiaz garatutakoarekin konparatu da.



1.5. Irudia: *Garapenerako erabili den* Xilinx*en* Spartan3E Starter Kit *txartela.*

[10] www.digilentinc.com
[11] www.mathworks.com
[12] www.xilinx.com



1.6. Irudia: *Garapenean jarraitutako metodologiaren fluxu-diagrama.*

1.5. Dokumentazioaren antolaketa

Araua jarraituz dokumentu bakoitzak dagozkion aurkibide eta zerrenda osoak dituenez, 0. Dokumentuak lehen mailako atalak baino ez ditu adierazten, irudi eta taula guztien zerrendekin batera.

1. Dokumentuak azaldutako metodologiak agindutako ordenean lantzen ditu jarraitu diren urratsak:

Teknikaren egoera aurkeztearekin batera, azterketa eta garapena modeloen simulazioan eta hardware ko-simulazioan oinarrituko dela ondorioztatuz, helburuko plataforma gisa *Spartan3E* FPGA aukeratzearen arrazoiak agertu dira lehenengo. Abio puntua argi, plantak identifikatu eta hauen portaera modelatzen duten adierazpen matematikoak lortzeko prozedurak azaldu dira.

Zuzeneko motor baten erantzuna adierazten duen transferentzia-funtzioa adibide gisa erabiliz, eta *Matlab/Simulink* softwarea baliatuz, blokeez osatutako denbora jarraituan begizta itxiko sistema osoaren modeloaren garapena azaldu da. PID kontrolagailuak izan ditzakeen egituren arteko hautaketa burutu eta S planotik Z planora bihurketa egin da. Denbora diskreturako aldaketak atxikitzailea gehitzea behar du, eta daudenetan ZOH motakoaren hautaketa arrazoitzen da ondoren. Era berean, atxikitzailea aginduko duen laginketa-maiztasuna baldintzatzen dituztenak aztertu dira, benetako denboran arituko den sistema garatu dela kontuan izanda. Kontrolagailuaren ereduzko parametroak ezarri dira oinarrizko sistema erabilgarria osatzeko.

Kontrolagailuaren barne eragiketak koma finkoan burutzearen zergatia argudiatu eta *Xilinx/System Generator* baliatuz sistema barnean erabilitako hitz-luzerak aztertu dira ere, zehaztasunean eta portaeran berebiziko garrantzia baitute. *Xilinx/ISE* softwarea erabiliz eskuratutako modeloa VHDL lengoaian deskribatzeko jarraitutako pausuak erakutsi dira. Proiektu honen zutabe garrantzitsua Z planoan adierazitako transferentzia funtzioak hardwarera bihurtzea izanik, zehaztasun osoarekin azaldu da kontrolagailuaren deskribapena. *Hardware-in-the-loop* bitartez simulatu eta aurreko guztiekin eta oinarrizko denbora jarraituko ereduarekin konparatu da, bihurketa pausu ezberdinetan emandako aldaketek eragindako portaera ezberdintasunak aztertzeko.

Zehaztapenak betetzen dituen kontrolagailua egina, *Xilinx*en *Spartan3E Starter Kit* garapen txartelak eskainitako baliabideez, autonomia, modularitatea, hiztunkeria eta erabilerraztasun baldintzak betetzeko garatutako osagaien deskribapenak esplikatu dira: LCD, VGA, sakagailuak, etab. Edozein plantarekin erabili daitezkeenak, beharreko parametroak aldatuta.

Azterketa kasuetan, eskuragarri dauden sentsore eta eragingailuetan oinarrituta, planta ezberdinak kudeatzeko ebatzitako sarrerako/irteerako osagaiak eta VHDL lengoaian eraikitako logikak deskribatu dira. Planta aurreko ataletan adierazitakoen arabera modelatu eta identifikatu da, PID kontrolagailua doitu eta laginketa-maiztasuna ezarri, eta parametroak aldatu dira. Azterketa esperimentalen bidez garatutako sistemaren portaera zuzena dela egiaztatu da.

2. Dokumentuak azterketa kasu bakoitzean plantaren identifikatutako modeloa eskuratzeko jarraitutako pausuak (hartutako neurketak eta egindako kalkuluak) azaltzen dira. Inplementa-

zioen ezaugarri zehatzak ere biltzen ditu: egikaritze denbora, erabilitako azalera eta xahututako potentzia. Bertan azaltzen dira garapenerako zerbitzariaren erabilera-estatistikak, *Subversion* bertsio-kontrol-sistemarenak eta *Allura* tiket-kudeatzailearenak.

Irakurleak sistema osotasunean eta maila baxuenean aztertu ahal izateko 3. Dokumentuan erabilitako iturriak daude komentatuta, bai *Matlab script*ak, bai VHDL deskribapenak. 4. Dokumentuan *Pierre St. Martin* inguruneari dagozkion zehaztasunak eta lizentzien hitzez-hitzeko kopiak daude.

5. Dokumentuan edukien eta iturrien erabilerarako baldintzak azaltzen dira. Aldi berean, erabilitako iturri eta baliabideen jatorria aitortzen da bertan. Baldintza administratibo, tekniko eta ekonomikoek osatzen dute dokumentua.

Garapenean emandako denboraren eta erabilitako tresnen zein gailuen aurrekontua 6. Dokumentuan dago adierazita. Proiektuaren egitura dela eta, bi zerbitzu ezberdinen agiriak eskaintzen dira: garapen osoa eta azterketa kasu berria lantzea.

Iturri, maila eta izaera orotako biltegi bibliografikoa bildu da 7. Dokumentuan, landutako esparruak duen garrantziaren, historiaren eta gaurkotasunaren adierazgarri. Ezagutza ezberdineko irakurleek ere erregistro ezberdinak eskura izan ditzaten.

Azkenik, dokumentuokin banatutako CDak aurreko guztiak biltzen ditu: *Simulink* modeloak, VHDL eta M iturriak, inplementazioen txostenak, testu eta irudien iturriak, lizentzien kopia, txantiloiaren iturriak, etab. Era berean, aipatutako web-gunean proiektuaren fitxategi guztiez gain, aldiro emandako aldaketak daude eskuragarri.

2.
Teknikaren egoera eta ebatzien azterketa

*Åström eta Murray*k [5, ch. 1] bikain azaltzen dute berrelikatutako sistemen eta kontrolaren garrantzia, naturan aurkitu daitezkeen prozesu biologikoetatik hasita eta espazioan dabiltzan *Curiosity*[13] bezalako makinak barne.

*Laplace*ren S espazioan transformatuen bitartez definitutako *kontrol klasikoa*n PID kontrolagailuak berebiziko garrantzia izan du, egitura sinplea izanik baliabide gutxiko inplementazio erraza behar duelako. Honen adierazgarri da autore gehienek atal zehatzak garatzea, aipatutako *Åström eta Murray*k [5, ch. 10] esaterako, edo *Kuo*k [6, ch. 10].

Sistema digitalen agerrerarekin eta hauekin lotutako teknologiaren garapenarekin batera izaera diskretua duten egoera-espazioan (*state-space*) modelatzeko sistemak agertu dira, *kontrol moderno* deritzoten metodoak bilduz eta Z plano diskretuan. Hainbat urtetan kontrol digitalak PID jarraituen egitura zurrun eta sinpleak balio ez zuen sistema konplexuetan erabili izan dira soilik.

Egun, ordea, PID elektroniko gehienak teknologia digitalean eskaintzen dira. Honek komunikaziorako moduluak integratzea edo diseinuak parametrizatzea ahalbidetzen du. Aldi berean funtzio konplexuagoak ere garatzeko aukera eskaintzen dute, hala nola, auto-doiketa (*autotuning*) edo urrunetiko ikuskapena.

Espazio jarraituari buruzko eskuragarri dagoen bibliografia eta ikerketak baliatu ahal izateko, eta aldi berean integrazio eta parametrizazioa bermatzeko, kontrolagailu jarraituaren erantzun baliokidea duen sistema digitala garatu da. Baliokidetasuna eta *kausalitatea* ziurtatzeko, *benetako denbora* kontzeptuak ezarritako mugak bete dira, laginketa eta egikaritze denbora baldintzatzen dituztenak.

Mikroprozesadoreak (x86/PPC/ARM), automata programagarriak (PLC), mikrokontrolagailuak, FPGAk eta osagai diskretuetan oinarritutako diseinuak erabiltzen dira kontrol begiztak inplementatzeko. Sistema handiagoekin integrazioa denek bermatzen badute ere, baliabideen eta beharreko potentziaren ikuspuntutik ezberdintasun handia dago. Inplementatzeko FPGA plataforma aukeratu da, azkeneko hauek, lan-abiadura maximoa eta konputazio ahalera aintzat hartuta.

Autonomiaren ikuspuntutik, baliabideen beharra presente izanik, mikrokontrolagailuak edo ARMak ere maila berean daude. Baina, horiek ez bezala, FPGA sistemek kontrol begiztan abiadura handiagoak erdiestea ahalbidetzen dute, hardware konfiguragarritasuna baliatuz, egitura finkoa baino paralelismo fisikoa indartuz. *Arduino* proiektuko PID liburutegiak[14], adibidez, 200 ms inguruko laginketa-maiztasunak jartzen ditu eredutzat, eta proiektu honetan 1 ms-tik beherakoak erdiestea da xedea.

[13] en.wikipedia.org/wiki/Curiosity_(rover)
[14] playground.arduino.cc/Code/PIDLibrary

***Xilinx*en gailuak:**	**Konektore eta interfazeak:**
Spartan-3E FPGA (XC3S500E-4FG320C)	Ethernet 10/100 Phy
CoolRunner™-II CPLD (XC2C64A-5VQ44C)	JTAG USB download
Platform Flash (XCF04S-VO20C)	2x 9-pin RS-232 serial port
Erlojuak:	PS/2, rotary kodetzaile with push button
50 MHz crystal clock oscillator	4x Slide switches
Memoria:	8x Individual LED outputs
128 Mbit Parallel Flash	4x Momentary-contact push buttons
16 Mbit SPI Flash	100-Pin expansion connection ports
64 MByte DDR SDRAM	Three 6-pin expansion connectors
	16 character - 2 Line LCD

1.3. Taula: Spartan3E Starter Kit *txartelaren osagaiak.*

Spartan3E Starter Kit Board[15] aplikazio orokorrerako txartela da eta, 1.3. taulak adierazten duenez, sarrera/irteera ezberdinak inplementatzeko eta komunikazioa garatzeko baliabide ugari ditu. 2008-2009 ikasturtean egindako *Eragingailu Logiko Programagarriak Dituzten Sistema Digitalak* ikasgaian txartela erabili izanagatik eta eskura izateagatik, hau izan da inplementaziorako plataforma. Helburua kontrolagailuaren garapena izanik, hardwarearen ondorioz eman litezkeen arazoak murriztea hobetsi da, produktu komertzialak erabiliz.

Edonola ere, *Mathlab/Simulink* edo *NI/Labview* inguruneetako tresnek abstrakzio-maila altuko garapen-fluxua dute eta instrumentaziorako zein prototipoak azkar egiteko paregabeak dira. Diseinuak abstrakzio-maila baxuko sistemetara (C, VHDL, etab.) bihurtzeko algoritmoak dituzte, ADQ txartelak, mikrokontrolagailuak zein FPGAk erabiltzeko. Proiektuaren helburua kontrolagailuaren garapena izanik, teknika hauek ulermenean laguntzeko erabili dira.

Azken urteetan *MaKey MaKey*[16], *littleBits*[17], *Arduino*[18] edo *Raspberry Pi*[19] bezalako proiektuek berebiziko gorakada izan dute. Alderdi teknikoan aurrapen aipagarririk ekarri ez badute ere, teknologia ezagutza handia ez dutenei gerturatzeko grinak eta esparru horretan tresnen erabilera errazten eta dokumentazioa osatzen egindako lanak erantzuna aurkitu du mundu mailan.

Aurrekoek, *Arduino*k eta *Raspberry Pi*k bereziki, teknologia ezberdinetan aritzeko gertuko ekosistema[20] osatzen dute. Mikroprozesadoreek, mikrokontrolagailuek eta osagai diskretuek badute tokia ekosisteman, baina ez dago pareko FPGA proiekturik.

Badira *OpenCores*[21] bezalakoak, baina ezagutza maila aurreratua eskatzen dute. *Altera*k ere diseinu adibideak eskaintzen ditu VHDL[22] zein Verilog[23] lengoaietan. *Xilinx*ek, aldiz, txer-

[15] xilinx.com/products/boards-and-kits/HW-SPAR3E-SK-US-G.htm
[16] makeymakey.com
[17] littlebits.cc
[18] arduino.cc
[19] raspberrypi.org
[20] radikaldesig.com/2012/05/18/raspberry-pi-arduino-un-nuevo-ecosistema/
[21] opencores.org
[22] altera.com/support/examples/vhdl/vhdl.html
[23] altera.com/support/examples/verilog/verilog.html

tatutako mikrokontrolagailuen[24] sostengu gisa aurkezten ditu, *hard-soft co-design* deritzona [7].

Epe luzean, aurreko paragrafoan aipatutakoak erabileran erraztasuna bermatzeko itxuraldatuta aurkeztea da kriseilua. Adibide zehatz, azkar eta deigarriak garatzeko xedeaz, azterketa kasuen garapenean horiek izango dira buruan, praktika gisa burutu ahal direnak.

Fritzing[25] azkeneko aplikazio gisa aproposa bada ere, egun ez du FPGA txartelentzat sostengurik, ezta integratuentzat. Produktuara heldu arteko bideak azterketa sakonagoa behar du, plaka merke autonomoa edo *shield* bat garatu artekoak. *Fedora Electonic Lab*[26] *GNU/Linux* sistema eragileak garapenerako etapa guztiak burutzeko tresnak biltzen ditu, baina hauek ez dira erabat integratuta azaltzen eta erabilera ez da azkarra.

*Mathworks*ek *Matlab/Simulink*ekin eta *Xilinx*ek *ISE/System Generator*ekin ingurune paregabea eskaintzen dute. Abstrakzio-maila altuenetik (interfaze grafikoa eta blokeez adierazitako osagaiak) baxuenera (VHDL) lan egitea ahalbidetzen dute, eta bitarteko konbinazio anitz ere. Aukeratutako txartelarekin hardware ko-simulaziorako argibide zehatzak ere badituzte eta ezagunak dira aipatutako ikasgaian eta *Erregulazio Automatikoa*n erabili izanagatik.

Laburbilduz, PID kontrolagailudun begizta itxiko kontrol-sistema digitala *Spartan3E Starter Kit* txartelean inplementzea aukeratu da eta garapenerako *Matlab/Simulink* eta *ISE/System Generator* erabiltzea.

[24] xilinx.com/products/boards/s3estarter/reference_designs.htm
[25] fritzing.org
[26] spins.fedoraproject.org/fel/

3.
Plantaren modelatzea eta identifikazioa

Proiektuaren garapena modeloen simulazioan oinarritu den heinean, landutakoek errealitatearekin ahalik eta erlazio zuzenena izan dezaten berebiziko garrantzia du modeloak lortzeko prozedurak aztertzeak.

*Aguado et. al*ek [8] azaldu bezala, *prozesuen identifikazioa* deritzon terminoak, baldintza berdinetan (sarreran aldagai multzo berdina izanik) kontrol automatikorako zehaztasun nahikoarekin sistema erreal baten irteera aldagaien erantzuna errepikatzen duen modelo matematikoaren egitura eta parametroak lortzeko algoritmo, teoria eta ikerketak biltzen ditu.

Ikuspuntuaren arabera, prozesuen helburuak ezberdinak izan daitezke. Honela, diseinu eta garapenean zehar balioztatzeko prozesuaren simulazioa burutzeaz gain, *online* edo *airean* deritzen kontrolagailuen parametroak identifikazio jarraituaren emaitzen arabera moldatu daitezke (*autotuning*). Era berean, zenbait algoritmotan etor daitezkeen seinaleen balioak aurreikusteko erabiltzen dira, hala nola, *Smith*en *predictor*a.

Helburuen arabera identifikaziorako metodo anitz daude. Oinarrizkoa azterketa analitikoa da: prozesuaren erlazio fisiko-kimikoak oinarri ditu eta ekuazio zinematikoak, dinamikoak, masa balantzeak, energia balantzeak, etab. aztertzean datza. Honela, modelo konplexuak eta ez-linealak lortzen dira askotan, eta prozesuaren teknologiaren ezagutza handia eskatzen du. Identifikazio esperimentalak, ordea, sarreran seinale ezagunak (maila sarrerak, maldak, sinusoidalak, inpultsuak, sasiausazko sekuentzia bitarrak...) erabilita modeloa epe laburragoan lortzea ahalbidetzen du.

Identifikazio esperimentalak praktikan arazoak sortu ditzake informazioa eskuratzeko plantaren funtzionamendua asaldatu behar delako. Badaude eragina zeharo murrizten duten metodoak, baina zenbaitetan ezineskoa gertatzen da prozesua gelditu barik saiakuntzak burutzea. Era berean, modelo deterministikoak eskuratzen dira, eta ondorioz, ez dituzte kanpoko asaldadurak (zarata) kontuan hartzen.

Proiektuaren helburua kontrol-sistemaren hardware garapena izanik, plantaren identifikazioa zeharkakoa da; eta egin den lehenengo azterketa kasua korronte zuzeneko motorra duen maketa denez, saiakuntza esperimentalek zehaztasun nahikoa ematen dute. Sistema txertatuan baliabideen erabileraren optimizazioa gogoratuz, hauetan sinpleena maila sarrerari erantzuna da. *Laplace*ren S domeinuan adierazitako esponentzial eta senoidalen bidez osatutako lehen eta bigarren ordeneko modelo anitz lortu daitezke erantzuna aztertuz.

3.1. Lehen ordeneko transferentzia funtzioa

Korronte zuzeneko motorra egitura fisikoak baldintzatuta izaera integratzailea izateagatik[27], begizta irekian egonkorra da eta erantzunaren izaera esponentziala da. Lehenengo ordeneko

[27] Aldaketa bortitzei erantzun azkarra ematen ez diena.

transfrentzia funtzio bat erabili daiteke, beraz. Sarreran aldaketa gertatzen denetik plantan lehenengo erantzuna antzeman arte atzerapena gertatu daitekeenez (aginduek sisteman zehar hedatzeko behar duten tartea adibidez), bi modelo ezberdin aurkezten dira:

$$\textbf{F}\text{irst }\textbf{O}\text{rder }\textbf{T}\text{ransfer function:} \qquad \frac{K}{\tau s + 1}$$

$$\textbf{F}\text{irst }\textbf{O}\text{rder }\textbf{P}\text{lus }\textbf{D}\text{elay }\textbf{T}\text{ransfer function:} \qquad \frac{K}{\tau s + 1} e^{-\alpha s}$$

Azterketa kasuen 8.1. atalean adierazten denez, erabiliko diren sarrerako/irteerako interfazeek ere plantaren erantzunean dute eragina. Azterketa kasu bakoitzaren menpekoak direnez, horiek erabilita egingo da identifikazioa, plantaren transferentzia funtzioak dinamika guztien eragina bilduko duelarik.

Ageriko baldintza nagusia inplementatuko den kontrolagailua diskretua izatea da. *Aguado et. al*ek [8, pp. 15-16] azaldutako eta Z planoan adierazitako FODT funtzioak T_s laginketa-periodoaren eragina hartzen du kontuan:

$$\textbf{D}\text{iscrete }\textbf{F}\text{irst }\textbf{O}\text{rder }\textbf{P}\text{lus }\textbf{D}\text{elay }\textbf{T}\text{ransfer function:} \qquad z^{-\text{round}(\frac{\alpha}{T_s})} \frac{\frac{K \cdot T_s}{\tau}}{z - e^{\frac{-T_s}{\tau}}}$$

Matlab/Simulink inguruneak, aurreko adierazpen matematikoez gain, identifikazio analitikoan oinarritutako modeloak adierazteko hainbat bloke ditu, fabrikatzaileek emandako informazioa, ezaugarri-orrietan adierazitako parametro mekaniko zein elektrikoak, paramatroak ezartzeko baliatuz. Era berean, garapenean zehar deskribatutako bloke zehatzak (*Blackbox*) txertatzeko aukera dago, sakondu ahala modeloa berregin eta hobetzeko.

Azkenik, irakurketa-gailuek eta kontrolagailuaren egiturak ezarritako mugek bereizmenean duten eragina ere kontuan hartu beharra dago. *Simulink*ek *Data Casting* prozedura bitartez datuen formatua aldatzeko *Convert* blokeak ditu eskuragarri. Aldi berean, *System Generator*ek *Gateway In* eta *Gateway Out* blokeen bitartez adierazten ditu formatu aldaketak.

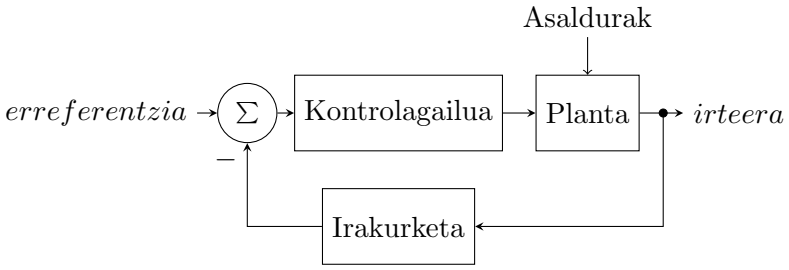
2. Dokumentuan bildutako neurketa eta kalkuluetan ikusi daitekeenez, azterketa kasu bakoitzaren identifikaziorako hurrengo prozedura jarraitu da:

1. Identifikazioa burutzea: laborategian maila sarrerei erantzunak irakurtzea.
2. Plantaren oinarrizko modeloa aukeratzea: FOT, FOPDT edo DFOPDT.
3. Sarrerako/irteerako osagaien modeloak aurkeratzea: transferentzia funtzioak, bloke zehatzak, *Blackbox*, etab.
4. Modeloen parametroak ezartzea: irakurritako erantzunak aztertuz modelo antzekoena eskuratzea.
5. Bereizmenaren eragina modelatzea: datuen formatua aldatzea.

4.
Kontrol-sistemaren modeloa egitea

Kontrol-sistema baten helburua planta ezagun batek nahi den erantzuna ematea da. Begizta irekiko sistema baten bitartez egin daiteke hori, zeinetan kontrolagailuaren irteera erreferentzia-seinalearen arabera zuzenean aldatzen den; edo begizta itxiko kontrol-sistema erabili daiteke, honetan irteera ezartzeko erreferentzia ez ezik plantaren irteera, berrelikadura seinalea, ere kontuan hartzen delarik.

Berrelikadura ezinbestekoa da plantaren irteera nahi den erantzunetik gertu mantentzeko, nahiz eta asaldurak edo plantaren dinamikan edozein aldaketa eman. Egun koste baxuko kontrolerako gailuak eskuragarri izatean erabilera zeharo hazi da, eta geroz eta aplikazio gehiagok (kontsumorako elektronika produktuek esaterako) erabiltzen dute berrelikadura.



1.7. Irudia: *Oinarrizko berrelikatutako kontrol-sistema.*

Ohiko berrelikatutako kontrol-sistema 1.7. irudian agertzen da adierazita. Sistema osoaren portaera elementu bakoitzaren aukeraketa zuzenaren menpekoa da. Kontrolagailuaren garapenerako, irakurketa zein eragingailuen dinamikak ez dira kontuan hartu, eta edozein kasutan plantaren barnean aztertu dira kasu praktikoetan.

Garapenean zehar arazoak murrizteko eredu planta gisa korronte zuzeneko motor baten transferentzia-funtzio arrunta erabiltzea aukeratu da. Hau, plantaren dinamikaren ikuspuntutik, zeharo sinplea denez, azterketa eta frogatzea erraztu ditu. Diseinatutako kontrolagailua egonkorra eta eraginkorra dela ziurtatzean, egin beharreko aldaketak egin dira planta zehatzak kontrolatzeko (azterketa kasuetan ikus daitekenez). Laplace-ren S domeinuan adierazitako lehenengo mailako transferentzia funtzio batek (1.1) beteko du eredu plantaren funtsa.

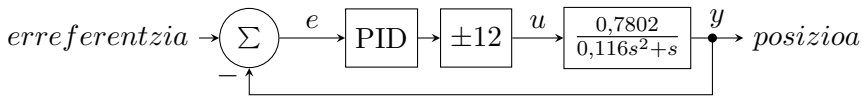
$$\frac{0,94}{0,116s+1} \tag{1.1}$$

Posizio kontrola egiteko xedez, plantaren transferentzia funtzioari irabazia ($K = 0.83$) eta integrazio funtzioa ($\frac{1}{s}$) biderkatu zaizkio eta emaitza (1.2) izan da planta gisa erabili dena.

$$\frac{0,94}{0,116s+1} \cdot 0,83 \cdot \frac{1}{s} = \frac{0,7802}{s(0,116s+1)} \tag{1.2}$$

Kontrolagailuari dagokionez Proportzional-Integratzaile-Deribatzaile (PID) deritzona aukeratu da, hiru parametro dituena. Hainbat prozesu ezberdinetan emaitza onak emateagatik eta egitura sinplea eta erabilera intuitiboa izateagatik, aurreko mendeko hasieratik kontrolagailu honen erabilera oso nabarmena izan da kontrol automatikoaren arloan. Industrian estandar bihurtu dela esan genezake. Teknologiaren garapenarekin batera bilakatzen joan da, eta egun sistema digitaletan inplementatu ohi da, sistema pneumatiko edo elektrikoekin baino. Askotan PID kontrolagailuak beste sistema sofistikatuagoetako oinarrizko atal gisa erabiltzen dira, nahi den portaera erdiesteko nahikoa ez denean, edo kontrolatu nahi den prozesua konplexutasun handiagokoa denean.

Praktikan erabiliko diren plantek elikadura tentsio maximoa ezarrita dute. Hori dela eta kontrolagailuaren irteeran saturazio blokea ezarri da, plantaren sarrera -12 eta 12 artean[28] mantendu dezan. Nahiz eta simulazioan balioek mugarik ez izan, praktikan planta erre daiteke behar baino tentsio handiagoz elikatuz gero, edo babes zirkuituak suntsitu litezke. Helburua *Matlab/-Simulink*en azterketak burutzeko sistemaren modelo ahalik eta fidagarriena egitea denez, jarri beharra da (1.8. irudia).



1.8. Irudia: *Azterketarako erabilitako sistemaren modelo jarraitua.*

4.1. Kontrolagailuaren modelo jarraitua

PID motako kontrol-araua aplikatzea paraleloan eragindako hiru kontrol funtzio ezberdinen emaitzen batura behar bezala lortzea da, funtzioak proportzionala, integratzailea eta deribatzailea izanik:

Proportzionala

Kontrolagailuaren sarrera den erreferentzia (r) eta plantaren irteeraren (y) arteko errore-seinalea (e) handitu egiten du irteeran (u).

$$u(t) = K_p \cdot e(t) \qquad \frac{U(s)}{E(s)} = K_d$$

Integratzailea

Errore-seinalea integratu eta konstante batez biderkatzen du. Plano konplexuaren jatorrian poloa izateak, egoera egonkorreko errorea zerora bultzatzen du erreferentzia-seinalean maila aldaketa ematen denean edo asaldurarik agertuz gero. Hori dela eta, *automatic reset* ere esaten zaio funtzio honi.

[28] Nahiz eta unitaterik ez izan, V gisa hartu dira.

$$u(t) = K_i \int_0^t e(\tau)d\tau \qquad \frac{U(s)}{E(s)} = \frac{K_i}{s}$$

Deribatzailea

Errore-seinalea deribatzen du eta konstante batez biderkatu. Proportzionalak momentuko sarreraren balioaren araberako balioa ematen du, integratzaileak aurreko balioen araberakoa eta deribatzaileak etorriko diren balioak hartzen ditu aintzat. Horregatik *anticipatory control*, *rate action* edo *pre-act* adierazpideak ere hartzen ditu.

$$u(t) = K_d \frac{de(t)}{dt} \qquad \frac{U(s)}{E(s)} = K_d \cdot s$$

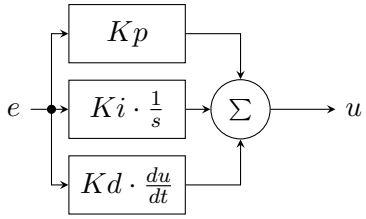
Hiru funtzioak adierazpen bakarrean batzea modu ezberdin askotan egin daiteke. Egitura ideala deritzonak (1.3)-k adierazitako transferentzia funtzioa du. Non K_p irabazi propotzionala, T_i integrazio denbora konstantea, eta K_d deribazio denbora konstantea diren.

$$\frac{U(s)}{E(s)} = K_p \left(1 + \frac{1}{T_i \cdot s} + Td \cdot s\right) \tag{1.3}$$

Inplementatzeko beste modu bat, orokorrean hedatuena, egitura paraleloa da, (1.4)-k adierazitakoa. Honetan hiru funtzioak erabat banaturik daude, nahi izanez gero adarretako edozeini dagokion konstanteari zero esleituz eragina deuseztatzeko aukera emanez[29]. Horregatik, beharren arabera eta parametroak soilik aldatuta P, PI, PD eta PID kontrol legeak erabiltzea ahalbidetzen duelako, egitura hau aukeratu da garapenerako. Paraleloa eta idealaren parametroen arteko bihurketa hurrengo formulen bitartez egin daiteke:

$$K_i = \frac{K_p}{T_i} \qquad K_d = K_p \cdot T_d$$

Garapenean zehar erreferentzia izango den modeloa *Laplace*ren domeinuan, hau da, denbora jarraituan, deskribaturiko transferentzia funtzioekin egin da. Nahiz eta helburuko sistema diskretua den, jarraituaren portaera berdina bilatu nahi denez, baliokideak direla onartu ahal izateko, doiketa eta banda-zabaleraren azterketarako modelo jarraitua erabili da; eta behin eta berriz aldaratuko dira erantzunak.



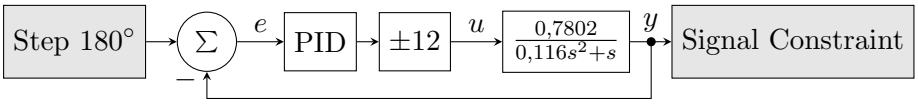
$$\frac{U(s)}{E(s)} = Kp + Ki \cdot \frac{1}{s} + Kd \cdot s \tag{1.4}$$

1.9. Irudia: *Azterketarako erabilitako PID kontrolagailuaren modelo jarraitua.*

[29] Besteetan integrazio denboraren balioa zero denean terminoak infiniturantz joko luke.

4.2. Doiketa (K_p, K_i, K_d)

Nahiz eta PID kontrolagailua eskuz doitu daitekeen, *Moreno et al.*ek [9] azaldu bezala metodo analitiko zein numerikoak erabiliz, proiektuaren helburua doiketa ez denez, baizik eta kontrolagailuaren barne egituraren diseinua, zuzenean *Matlab/Simulink* inguruneak eskaintzen dituen tresnak erabili dira. Modeloan oinarritutako doiketa burutu da, ondorioz, eta plantaren identifikazioan eragindako erroreek zeharo baldintzatzen dute simulazioen eta benetako plantaren erantzunen arteko diferentzia.



1.10. Irudia: *PIDaren doiketarako erabilitako sistema:* Step 180º *sarreran eta irteeran* Signal Constraint.

Kontsigna edo erreferentzia-sarrera gisa maila blokea erabili da, *Step 180º*. Irteera, aldiz, *Signal Constraint* blokera konektatu da (1.10. irudia). Sistemaren diagrama orokorrean agertzen den PID blokea berez eskuz egindako *Mask* bat da. Barneko egiturak, 1.9. irudian ikusi bezala, hiru konstante ditu: Kp, Ki eta Kd. Berauek ezarri eta balioak esleitzeko *Mask* barnean sartu behar ez izateko, parametro gisa definitu dira. Aldi berean, *Workspace*ean P, I eta D aldagaiak sortu dira eta aurrekoei hurrenez hurren esleitu.

P	6	Step time	$.01s$	Values	$(0:1)$
I	0.125	Rise time	$0.5s$	Rise	90%
D	0.25	Setting time	$1.0s$	Setting	1.5%
		Overshoot	2.5%	Undershoot	1.5%

1.4. Taula: Signal Constraint *eta* Step 180º *blokeen bidezko doiketaren emaitzak.*

Signal Constraint bitartez doiketa egiteko, *Workspace*eko sortu berri diren hiru aldagaiak jarri dira parametro gisa. Irteeran jarritako bloke horrek, maila sarrerari erantzunak nahi diren espezifikazioak bete ditzan iterazioak burutzen ditu, *Gradient descent* metodoa erabiliz eta *Active-Set* algoritmoa. *Step 180º* eta *Signal Constraint* blokeek 1.4. taulan adierazitako ezaugarriak izanik, bertan adierazitako emaitza eman du optimizazio prozesuak.

Horiek dira garapenean zehar erabili diren parametroen balioak, txostenean *doiketa*[30] hitzez adieraziko direnak.

4.3. Jarraitua diskretua bihurtzea ($S \to Z$)

Sistema jarraitua naturan denbora tarte infinitesimaletan ematen diren aldaketak antzemateko gai dena da. Sistema digitala, aldiz, erloju batek agintzen du beti, eta ondorioz aldaketak antzemateko gaitasuna mugatuta du, oszilazio iturriak baldintzatua. Ezberdintasun hau dela

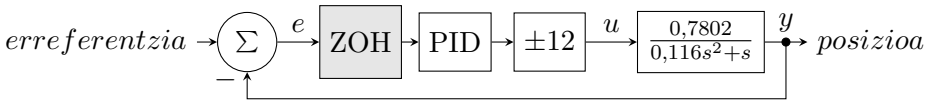
[30] 3A. eranskinean dagoen *tune.m script* hiru aldagaiak (P, I eta D) deklaratu eta balioak esleitzen ditu.

eta, sistema digitalak aztertzeko *Laplace*ren S planoan arituz gero, praktikan sistemak ez du analisi teorikoak emandako erantzuna beteko.

*Kuo*k [6] azaltzen duenez, Z transformatuak diferentzia linealetan ekuazioak eta datu diskretu edo digitalak dituzten sistema linealak ebazteko operatzeko metodoa osatzen du. Atal erreal zein irudikariak dituen z aldagai konplexua baliatuz, domeinu errealeko zenbaki sekuentzia Z domeinu konplexuko adierazpen gisa azaltzen du.

Transformatuaren hirugarren teoreman aurkeztu bezala, atzerapenak adierazteko operadorea z^{-n} da, non n laginketetan adierazitako atzerapena den. *Grout*ek [10, p. 501] gogoratzen duenez, balio bat laginketa-periodo batean gordetzen duen elementua erregistroa da, sistema digitalen oinarrizko memoria elementua. Hau dela eta, kontrol digitaleko sistemetan eta datuak lagintzekoetan, $y(t)$ seinalea F_s maiztasunarekin irakurtzen da, lagindutako aldiunetan seinalea adierazten duen denbora sekuentzia osatuz. Lagin arteko denbora tartea, T_s, konstantea izanik, eta benetako denboran jasoz gero, diferentzietan ekuazioak erabiliz landu daitezke datuak, aljebra matriziala medio.

Kontrolagailu ahalik eta fidagarriena erdiesteko, esparru honetan egindako garapenak baliatu dira *Laplace*ren S domeinuko adierazpenak hartuta deskribapenaren izaera diskretura moldatzen direnak lortzeko. Batetik besterako bihurketa zehatza ez denez, aldaketa burutzeko aplikazioaren eta beharrizanen arabera zenbait ekuazio ezberdin erabiltzen dira [10, p. 516]: *Euler* edo *Forward difference*, *Backward difference* eta *Tustin* edo bilinearra.



1.11. Irudia: *Azterketarako erabilitako sistemaren modelo diskretua.*

Denboraren azterketa ikuspuntu diskretutik burutzeko, *Simulink*eko sistema osoaren modelo orokorra aldatzea ezinbestekoa da. Kontrolagailua inplementatzeko gailua edozein dela ere, datu diskretuak prozesatzeko behintzat fabrikatzaileak adierazitako tartean balio finkoa izan beharko dute. Laginketaren ostean eta PID kontrolagailua baino lehen atxikitzailea jarri da (1.11. Irudia), lagindutako datuetan oinarrituta laginketa arteko seinalearen izaera baldintzatuko duena. Helburu honetarako erabili ohi direnak *Zero-Order Hold*, *First-Order Hold* eta *Fractional-Order Hold* dira [11]. Lehenengoak azken laginketaren balioa finko mantentzen du berritu artean, beste biek deribatuan oinarrituta hurrengoan irakurriko den balioa aurretiaz erdiesten saiatzen diren bitartean.

Nahiz eta *Ugalde et al.*ek [12] azaldu bezala zenbait helburutarako FROH atxikitzaileen erabilerak erantzuna hobetu dezakeen, proiektu honen izaera oinarrizkoa denez ZOH motakoa erabili da. Begizta sistemaren laginketa-maiztasunaren (F_s) ezarrera azaltzean adieraziko denez, planta banda-zabalera baino zeharo azkarrago ikuskatuko da. Ondorioz, Ts tarteetan seinaleak balio berdina mantenduko duela suposatu daiteke, elementu mekanikoen aldaketa abiadura ahaleraren azpitik egoteagatik.

Tarte infinitesimalak antzematen dituen sistemaren ikuspuntutik, naturan ematen diren aldaketa guztiak zeharo geldoak dira, momenturo baita erantzuna emateko gai. Are gehiago, momentuan momentuko erantzuna ematen du. Sistema digitalak, ordea, kontrol begizta egikaritzeko (sarreran (e) aldaketa antzematea, algoritmoa askatzea eta kontrol seinaleak (u) balio egokia hartzeko), oszilazio iturriaz gain kontrolagailuaren eta sistemaren konplexutasunaren (erabilitako algoritmoa, gailuen lan-abiadura maximoa, zehaztasuna, etab.) menpekoa den T_c denbora tartea behar du. Erreferentziako plataforma FPGA denez, hau da, hardware hutsa, T_c konstantea izango da.

Kausalitatearen araua deritzona bermatzeko, eta ondorioz garatutako kontrolagailuak jarraituaren portaera berdina duela ziurtatu ahal izateko, kontrolagailuaren ikuspuntutik plantaren dinamikak zeharo geldoa izaten jarraitu beharko du [13]. T_s-ren balio zehatza plantaren araberakoa denez, aurrerago azalduko diren irizpideen arabera, kontrolagailuaren garapenean (sarrerako/irteerako logikak barne) $T_c \ll T_s$ bermatu da:

$$\frac{T_c}{T_s} < 10^{-2} \tag{1.5}$$

(1.5) adierazpena betetzean benetako denborak (*Real time*) ezartzen dituen mugak errespetatzen dira. Horrek esan nahi du, sistema jarraituan bezala, momentuan momentuko erantzuna ematen dela, laginketa eta eragitea momentu berean ematen direla onartu baitaiteke.

Aurrekoan oinarrituta, kontrolagailuaren irteeran ez da atxikitzailerik gehitu. Nahiz eta *glitch*ak agertzerik badagoen, kontrolagailua eta plantaren dinamiken arteko erlazioa dela eta hauek ez dute eraginkortasunean eraginik.

4.3.1. Bihurketa adierazpenak

$$U(s) = U_p(s) + U_i(s) + U_d(s) \quad \begin{cases} \frac{U_p(s)}{E(s)} = Kp & U_p(s) = E(s) \cdot Kp \\ \frac{U_i(s)}{E(s)} = Ki \cdot \frac{1}{s} & U_i(s) = E(s) \cdot \frac{Ki}{s} \\ \frac{U_d(s)}{E(s)} = Kd \cdot s & U_d(s) = E(s) \cdot Kd \cdot s \end{cases} \tag{1.6}$$

(1.4) adierazpenean gainezartzea aplikatuz gero, atal proportzionala, deribatzailea eta integratzailea banaturik aztertu daitezke (1.6). Ikus daitekenez, irabazi proportzionalari dagokion adarrean ez dago s operadorerik, transferentzia funtzioa ez da denboraren menpekoa. Z planora bihurtzean, beraz, ez du inolako moldaketarik jasan beharko (1.7).

$$U_p(s) = E(s) \cdot Kp \quad \xrightarrow{s \rightarrow z} \quad U_p(z) = E(z) \cdot Kp \tag{1.7}$$

Adar deribatzaile eta integratzaileei dagokienez, bihurtzaile bakoitzari dagokion moldaketa eragin zaie.

$$\begin{array}{c}\text{Euler}\\ \text{Fordward difference}\end{array} \quad s = \frac{z-1}{Ts} \quad \begin{cases} \frac{U_i(s)}{E(s)} = Ki \cdot \frac{1}{s} & \xrightarrow{s \rightarrow \frac{z-1}{Ts}} & \frac{U_i(z)}{E(z)} = \frac{(Ki \cdot Ts)z^{-1}}{1-z^{-1}} \\ \frac{U_d(s)}{E(s)} = Kd \cdot s & \xrightarrow{s \rightarrow \frac{z-1}{Ts}} & \frac{U_d(z)}{E(z)} = \frac{Kd}{Ts} \cdot \frac{1-z^{-1}}{z^{-1}} \end{cases} \tag{1.8}$$

$$\text{Backward difference} \quad s = \frac{z-1}{z \cdot Ts} \quad \begin{cases} \frac{U_i(s)}{E(s)} = Ki \cdot \frac{1}{s} & \xrightarrow{s \rightarrow \frac{z-1}{z \cdot Ts}} & \frac{U_i(z)}{E(z)} = \frac{Ki \cdot Ts}{1-z^{-1}} \\ \frac{U_d(s)}{E(s)} = Kd \cdot s & \xrightarrow{s \rightarrow \frac{z-1}{z \cdot Ts}} & \frac{U_d(z)}{E(z)} = \frac{Kd}{Ts} \cdot (1-z^{-1}) \end{cases} \tag{1.9}$$

$$\begin{array}{c}\text{Tustin's}\\ \text{Bilinear}\end{array} \quad s = \frac{2}{Ts} \cdot \frac{z-1}{z+1} \quad \begin{cases} \frac{U_i(s)}{E(s)} = Ki \cdot \frac{1}{s} & \xrightarrow{s \rightarrow \frac{2}{Ts} \cdot \frac{z-1}{z+1}} & \frac{U_i(z)}{E(z)} = \frac{Ki\dot{T}s}{2} \cdot \frac{1+z^{-1}}{1-z^{-1}} \\ \frac{U_d(s)}{E(s)} = Kd \cdot s & \xrightarrow{s \rightarrow \frac{2}{Ts} \cdot \frac{z-1}{z+1}} & \frac{U_d(z)}{E(z)} = \frac{2 \cdot Kd}{Ts} \cdot \frac{1-z^{-1}}{1+z^{-1}} \end{cases} \tag{1.10}$$

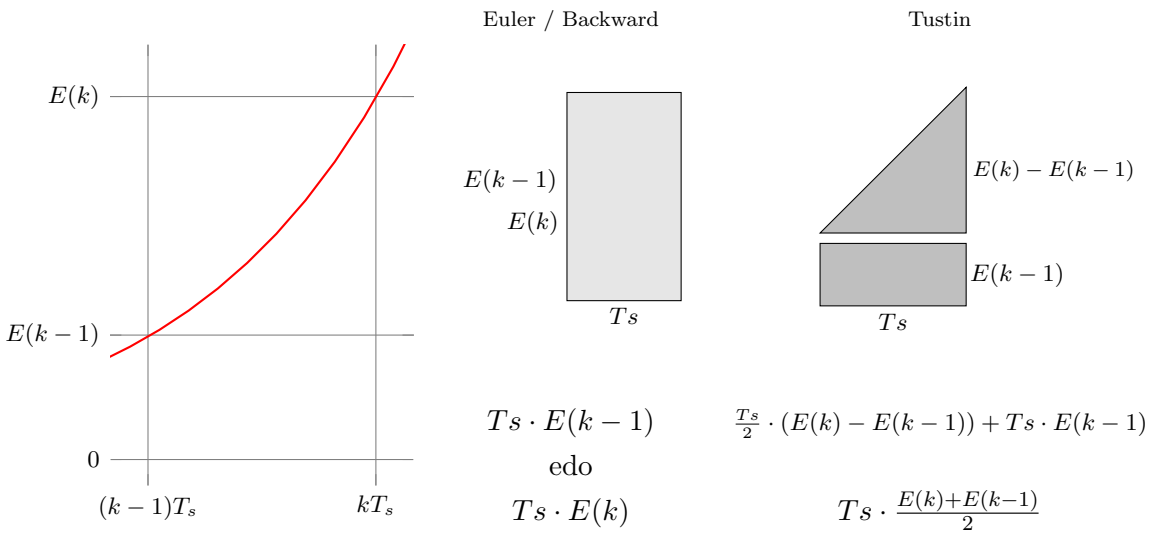
(1.8), (1.9) eta (1.10) adierazpenek azaldutakoa baliatuz, eta beharreko operazioen ostean 1.5. taulan adar integratzaileari dagozkien hiru ekuazioak parean jarri dira, konparatzeko xedez. Antzeman daitekenez, *Euler* eta *Backward* bihurketek egituraren aldetik emaitza berdina eman dute. Lehenengoak aurreko laginketako irakurketa erabiltzen du, eta bigarrenak azkenekoa. *Tustin*enak, aldiz, biak erabiltzen ditu.

Euler	$U_i(z) = Ki \cdot Ts$	$E(z)z^{-1}$	$+U_i(z)z^{-1}$
Backward	$U_i(z) = Ki \cdot Ts$	$E(z)$	$+U_i(z)z^{-1}$
Tustin	$U_i(z) = Ki \cdot Ts$	$\frac{E(z)+E(z)z^{-1}}{2}$	$+U_i(z)z^{-1}$

1.5. Taula: *Adar integratzailearen bihurketa adierazpenen konparaketa.*

Ezberdintasunen zergatia ulertzeko ezinbestekoa da integrazioaren kontzeptuaren oinarrira jotzea; honek funtzio baten azalera adierazten du limiteek definitutako tartean. Diskretizatzean, infinito balioen batura da, funtzioak laginketa bakoitzean izan duen anplitudeen batura hain zuzen ere. Funtzio jarraitu batek, lehenago adierazi bezala, laginketa momentuen artean balio aldakorrak izaten ditu.

(1.8) zein (1.9) adierazpenek azalera kalkulatzean aldaketa horiek arbuiatzen dituzte eta zuzenean irakurketa eta laginketa-periodoaz baliatuz laukizuzen baten azalera kalkulatzen dute. Funtzioa goranzkoa izanik, 1.12. irudian azaltzen den bezala, *Euler* bihurketak benetako funtzioak baino azalera gutxiago bereizten du; *Backward*ek aldiz, gehiago. Funtzioa beranzkoa



1.12. Irudia: *Adar integratzailearen bihurketa adierazpenen interpretazio geometrikoa.*

denean, errore berdina ematen da, baina trukatuta. Beraz, praktikan berdin dio bata zein bestea erabili.

*Tustin*en aukerak (1.10), berriz, bien tarteko emaitza ematen du, $E(k)-E(k-1)$ diferentziaren arteko erdia bereziz. *Euler*rek berezitutako laukizuzena eta triangelu baten batura/kenketa (goranzkoa edo beheranzkoa den arabera) bezala adierazi daitekeen trapezoide baten azalera kalkulatzen du. Hau dela eta, integrazio bihurketa honek *trapezoidala* izena ere hartzen du.

$$U_i(s) = E(s) \cdot Ki \cdot \frac{1}{s} \xrightarrow{s \to \frac{2}{Ts} \cdot \frac{z-1}{z+1}} U_i(z) = \frac{Ki \cdot Ts}{2}(E(z) + E(z)z^{-1}) + U_i(z)z^{-1} \qquad (1.11)$$

Inplementazioari dagokionez, *Tustin*en aukerak duen erdibitzailea konstantearekin batu daiteke. Honela aukera hau erabiltzeak batutzaile baten gehiketa baino ez du suposatzen[31]. Erabili den plataformak baliabide nahiko dituenez, eta elementu horren gehiketak eraginkortasunean eraginik ez duenez, adar integratzailea bihurtzeko *Tustin* aukeratu da (1.11).

$$\lim_{Ts \to 0} \frac{E(k) + E(k-1)}{2} \xrightarrow{(k-1) \to k} \frac{E(k) + E(k)}{2} = E(k) \qquad (1.12)$$

Laginketa-maiztasuna handitu ahala emaitzak berdintzen doaz (1.12), baina badaude ezberdintasunak *Santina et al.*ek [14, subsec. 6 Effect of Sampling Rate on Quatization Error] adierazi bezala. Bihurketa bilinearrak besteak baino hobeto egiten du S planoaren egonkortasun limiteen Z planorako aldaketa, eta frekuentzia erantzun hobea du, autore berdinen azterketan ikusi daitekeenez [15, subsec. 3 Digitilizing Analog Controllers]. *Franklin et al.*ek [16, ch. 6 Discrete Equivalents] sakonean jorratzen ditu bihurketak.

[31] Batutzailea gehitzean biderkatzailearen seinale batek bit bat gehiago beharko du, baina aldi berean konstantea erdibitzeak bit baten beharra kentzen du. Ondorioz, biderkatzailearen irteerak hitz-luzera berdina izango du.

Aukeratutako bihurketa edozein dela ere, integrazioak aurreko emaitza guztiak kontuan hartzen dituenez, berrelikadura positiboa agertzen adierazpen azken termino gisa. Kontrolagailua VHDL lengoaian deskribatzean sakonean aztertuko den *windup* ondorioa izan dezake: batutzaileak oso denbora gutxian balio oso altuak hartzea, beste adarren eragina deuseztuz eta sistema ezegonkortuz. Hau ekiditzeko modeloan saturazioa erabili da integratzailean, kontrolagailuaren irteeraren osteko blokearen muga berdinekin[32].

Euler	$U_d(z)z^{-1}$	$=$	$\frac{Kd}{Ts}(E(z) - E(z)z^{-1})$	
Backward	$U_d(z)$	$=$	$\frac{Kd}{Ts}(E(z) - E(z)z^{-1})$	
Tustin	$U_d(z)$	$= 2$	$\frac{Kd}{Ts}(E(z) - E(z)z^{-1})$	$-U_d(z)z^{-1}$

1.6. Taula: *Adar deribatzailearen bihurketa adierazpenen konparaketa.*

Adar deribatzailea bihurtzeko orduan *Tustin* bihurketak erantzun oszilakorra aurkezten du sistema egoera egonkorrean ari denean, *Schmidt*ek [17] ondorioztatzen duen bezala. 1.6. taulako konparaketak adierazten duenez, $E(k) = E(k-1)$ denean, hau da, sarrera konstantea denean aurretiaz batutakoen araberako ezegonkortasuna ezartzen du. *Euler*ren aukerak, berriz, azken laginketa eta aurreko laginketako balioak ezagututa, aurreko laginketan izan beharreko irteera kalkulatzea ahalbidetzen du. Bete nahi den aplikaziorako honek ez du zentzurik. Bihurketa hau laginketa bateko sarrera $E(k)$ eta irteera ezagututa $U(k)$ hurrengoan espero daitekeen sarrera $U(k+1)$ irudikatzeko erabili daiteke. Horregatik du *Fordward diference* izena, etorri behar diren balioak estimatzeko erabili daitekelako. Hala ere, hau ez da helbururako aproposa. Adar deribatzailearentzat baztertu gabekoa erabili da beraz: *Backward* bihurketa. Adierazpena (1.13) zuzen baten ekuazioa dela ikus daiteke, eta emaitza malda da.

$$U_d(s) = E(s) \cdot Kd \cdot s \quad \xrightarrow{s \rightarrow \frac{z-1}{z \cdot Ts}} \quad U_d(z) = \frac{Kd}{Ts}(E(z) - E(z)z^{-1}) \tag{1.13}$$

$$m = \frac{y_2 - y_1}{x_2 - x_1} \quad U_d(z) = \frac{E(k) - E(k-1)}{kT_s - (k-1)T_s}$$

Hiru adarren diskretizazioa banaturik aztertu ostean, berriz ere gainezartzea oinarri, adierazpen bakarrean batu dira (1.7), (1.11) eta (1.13).

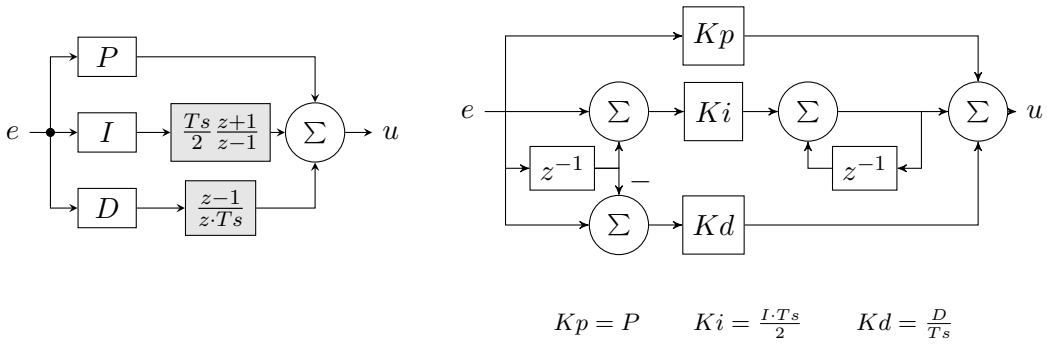
(1.14) adierazpena da txosten eta garapen osoan zehar erabiliko den modelo diskretua. *Simulink*eko modeloan 1.13. irudian agertzen den ezkerreko eskema erabili da, *Discrete Integrator* (Integrator method: Trapezoidal; Upper saturation limit: sat; Lower saturation limit: -sat) eta *Discrete Derivative* blokeak baliatuz.

[32] *tune.m* scriptak beste parametroekin batera ezartzen du.

$$U(z) = U_p(z)+U_i(z)+U_d(z) \begin{cases} \frac{U_p(z)}{E(z)} = Kp & U_p(z) = E(z) \cdot Kp \\ \frac{U_i(z)}{E(z)} = Ki \cdot \frac{Ts}{2} \cdot \frac{z+1}{z-1} & U_i(z) = \frac{Ki \cdot Ts}{2}(E(z) + E(z)z^{-1}) + U_i(z)z^{-1} \\ \frac{U_d(z)}{E(z)} = Kd \cdot \frac{z-1}{z \cdot Ts} & U_d(z) = \frac{Kd}{Ts}(E(z) - E(z)z^{-1}) \end{cases} \tag{1.14}$$

$$U(z) = Kp \cdot E(z) + \frac{Ki \cdot T_s}{2}(E(z) + E(z)z^{-1}) + U_i(z)z^{-1} + \frac{Kd}{T_s}(E(z) - E(z)z^{-1})$$

Inplementazioari begira, blokeak garatu egin dira ahalik eta oinarrizko gailu (batutzaileak, erregistroak eta biderkatzaileak) gutxien erabiltzeko, egitura konkurrentea mantenduz. Helburu horrekin, konstante mantentze diren aldagai guztiak batu dira, Kp, Ki eta Kd parametroen interpretazioa aldatuz. Garapenaren abstrakzio mailan jaitsi ahala, eskema hori erabili da, eskuman agertzen dena.



1.13. Irudia: *Garapenean erabilitako PID kontrolagailuaren modelo diskretua.*

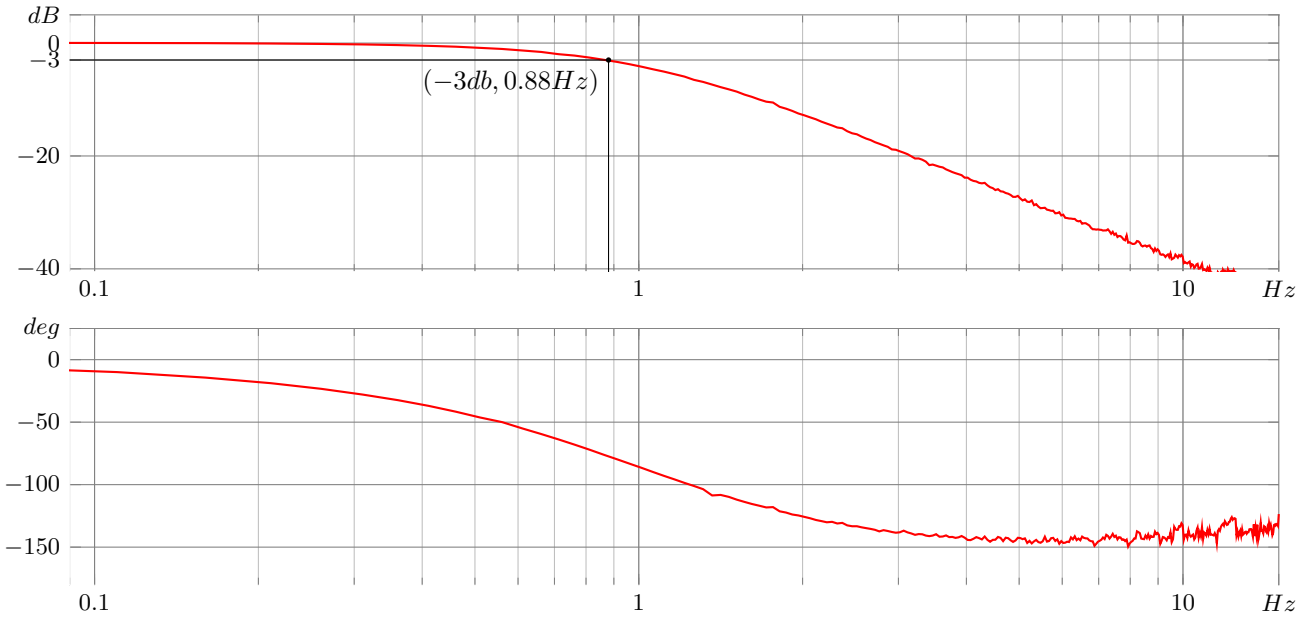
4.4. Laginketa-periodoaren azterketa (T_s)

Aurreko atalean aipatutakoa gogoratuz, sistema diskretua jarraituaren baliokidea dela ziurtatzeko kontrolagailuaren dinamika plantarena baino azkarragoa izan behar da. Horrek ziurtatzen du plantaren edozein aldaketaren aurrean kontrolagailuak erantzuna ematea. Hau dela eta, jarraian argituko denez, PID kontrolagailuen inplementazio digitalek sistemaren erantzuna moldatzen duten eta diseinatzaileak doitu beharreko lau parametro dituztela adierazi ohi da, hiru adarren konstanteei T_s gehitzen baitzaie.

K_p, K_i eta K_d parametroen ezarpenarekin gertatu bezala, laginketa-maiztasun egokia ezartzea ez da zuzeneko lana, ez argia ez azkarra. Badaude laurak aldi berean doitzeko proposatutako prozedurak, aski ezagunak diren *Zierger* eta *Nichols*en begizta irekian plantaren maila sarrera eta frekuentzia erantzunen azterketan oinarritutakok esaterako. *Åström*ek [18, sec. 6.6], *Zieger-Nichols*en taulek zarata ondo arbuiatzen dutela azaltzen du, baina maila aldaketetan izaera

oszilakorra dutela, eta hobetutako maila erantzunean oinarritutako prozedurarekin konparatzen ditu. *Seborg et al.*ek [19, p. 535] aurrekoak eta begizta irekian oinarritutako beste hainbat prozedura biltzen dituzte. Hala ere, *Zito et al.*en [20] adierazitakoarekin bat, ez dago edozein plantarako erantzun ona ziurtatzen duen hedatutako proposamenik.

Modelo jarraituaren irabaziak doitzean jarraitutako prozedura errepikatuz gero, iterazioetan oinarritutako optimizazio prozedurak lau aldagai aztertu beharko lituzke, eta baliteke irabaziei esleitutako balioak aldatzea. Garapenen osoan zehar sistemaren modelo jarraitua erreferentzia denez, 1.4. taulan adierazitakoak mantenduz, T_s bakarrik aztertu da. Erabat banatu dira, beraz, jarraituan burututako doiketa eta diskretuan ezarri beharreko laginketa-maiztasunaren aukeraketa.



1.14. Irudia: *Modelo jarraituaren banda-zabaleraren azterketa begizta itxian eta 1.4. taulan adierazitako doiketak erabilita.*

Shannon zein *Nyquist*en arabera [21] F_b banda-zabalera barnean mugatutako seinale baten laginak bakarrak dira baldin, soilik baldin, laginketa-maiztasuna $2 \cdot F_b$ baino handia bada. Hortaz, edozein seinale analogiko lagintzeko maiztasunak honen banda-zabaleraren bikoitza izan behar du, behintzat, aztertutakoa berreraiki ahal izateko. Honela ez bada, *Åström*ek [18, sec. 6.7] azaltzen duen *alias* efektua izan daiteke irakurketetan.

*Wescott*ek [22] egindako azalpenean ondorioztatzen denez, hainbat faktore izan behar dira kontuan eta ez da erraza askotan oinarri teorikotik abiatuta praktikan inplementatu beharreko maiztasuna ezartzea. *Designing for Absolute Time Response* atalean *Wescott*ek *real-time* sistemetan lagindutako seinalearen itxuraz gain honen bidean ezartzen diren atzerapenek ere berebiziko garrantzia dutela adierazten du. Ondorioz, laginketa-maiztasuna handitu ahala fase atzerapena (1.15) adierazpenaren arabera murrizten dela kontuan hartuta, banda-zabalera baino

$10-20$ aldiz maiztasun handiagoz gain-lagintzea gomendatzen da.

$$\Theta_{delay} = \frac{F_p}{F_s}\pi \quad [rad] \tag{1.15}$$

Era berean, *Santina et al.*en [14] gaiaren azterketa bikainean banda-zabalera baino $10-40$ aldiz maiztasun handiagoa proposatzen dute. Plantaren irteera modelatzeko lehen eta bigarren ordenako sistemak erabilita baldintza zehatzetan *Franklin et al.*ek [16] proposatutakoaren, $0,08 < T_s\omega_0 < 0,3$ ($\omega_0 \frac{rad}{s}$-tan adierazita), berdina dela ondorioztatzen dute.

Bertan adierazten denez, kontrol-sistema digital baten helburu nagusia irteerak sarrera jarraitu dezala izanik, diseinatzaileak berrelikadura kontuan izanik begizta itxian sistemaren banda-zabalera F_{cl} zein den aztertzea da zentzuzkoena. Senekoa da laginketa-maiztasunak duen F_{cl}ren menpekotasuna, sistema osoaren banda-zabalera erreferentziaren maiztasun handienak eta begiztan txertatutako elementu guztiek baitute eragina, eta ez plantak soilik. Nahiz eta, oraingoan ere, berebiziko garrantzia duten plantaren identifikazio zuzenak eta modelo egokia izateak.

Simulink inguruak linealizazio tresnak izanda, eta adierazitakoak aintzat hartuta, zuzenean sarrerako linealizazio puntua sortu da erreferentziaren puntuan eta irteerako linealizazio puntua plantaren irteeran. *Control Design* tresna sortako *Linear Analisys* erabilita[33] sistemaren frekuentzia erantzuna adierazten duen Bode diagrama lortu da (1.14. irudia).

Autore ezberdinek[34] egindako proposamenetan ordezkatu da 0,88 Hz edo $0,88 \cdot 2\pi = 5,529 \frac{rad}{s}$ mozketa-maiztasuna, tartea zeroan[35] hasten denez banda-zabalera dena (1.7. taula).

Wescott	$10 \cdot F_{cl} < F_s < 20 \cdot F_{cl}$	$(8,8 - 17.6)$ [Hz]
Santina et al.	$10 \cdot F_{cl} < F_s < 40 \cdot F_{cl}$	$(8,8 - 35,2)$ [Hz]
Franklin et al.	$\frac{\omega_0}{0,3} < F_s < \frac{\omega_0}{0,08}$	$(18,43 - 69,1)$ [Hz]

1.7. Taula: *Autore ezberdinek proposatutako laginketa-maiztasunak (F_{cl} =0,88 Hz).*

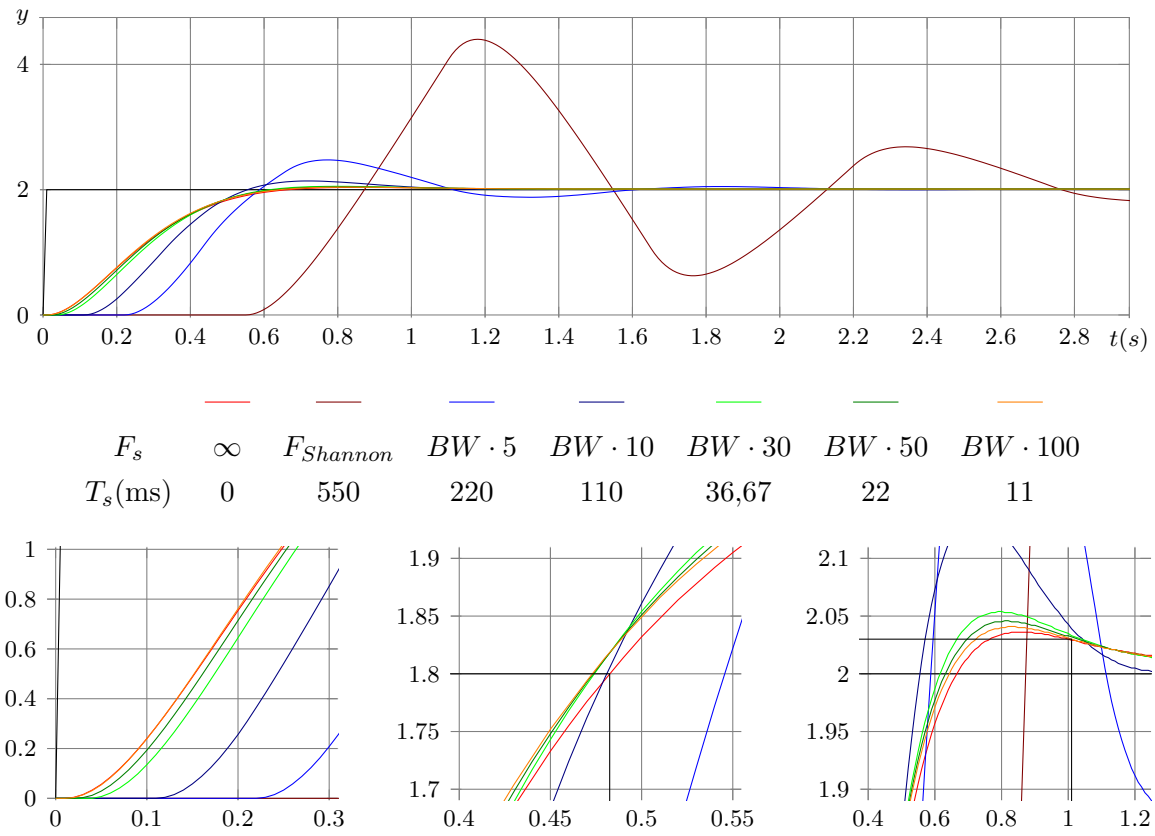
Nahiz eta taulan adierazitako emaitzei begira 20 Hz inguruko laginketa-maiztasuna zuzena den, autore guztiak bat datoz simulazioa eta froga ezberdinak egitea gomendatzean. $10X$ baino gutxiago ez erabiltzea ere guztiek adierazten dute, eta hortik aurrera behar izatekotan handitu daitekeela. Modelo jarraituarekin konparatuta maila sarrerari erantzuna aztertu eta gutxi gora behera T_s-ren zein balioetarako berdintzat jo daitezkeen ikusteko *Shannon*en frekuentziatik hasi eta $BW \cdot 100$ arteko zenbait laginketa-maiztasun ezarri dira.

1.15. irudian ikusi daitekeenez, *Shannon*en frekuentziarekin kontrola ez da onargarria. Banda-zabalera baino bost edo hamar aldiz handiagoko maiztasunekin ezberdintasun handia antzematen da, adibide plantarako kontrola onargarria izanik (gainditze nahikoarekin, baina onargarria).

[33] 3A. eranskinean bihurtu, simulazioak burutu eta emaitzak gordetzen dituen *save_bw.m script*a dago.

[34] *Franklin et al.*ek behintzat 20X proposatzen dute, eta ondorioz ez da *Wescott*en proposamenarekin bateragarria.

[35] Sistemak sarrera konstantea denean ere erantzungo du, errorea seinaleak maiztasunik ez duenean.



1.15. Irudia: *Laginketa-maiztasun ezberdinekin modelo diskretuaren eta modelo jarraituaren erantzunen konparaketa.*

Hogeita hamar aldiz handiagoa denean kontrola zeharo hobetzen da. Eta hortik ehun aldiz handiagorako ezberdintasuna oso txikia da (gehienez %0, 5-eko gainditze diferentziarekin).

1.7. taulan adierazitakoak gogoratuz, *Wescott*en proposamena ziurra dela ondorioztatu daiteke, baina ez da onena. Bat gatoz $10X$ minimoarekin, baina ahal den heinean *Santina*ren goi muga edo *Franklin*en erdibideko balioren bat erabiltzeak erantzun zeharo hobea ematen du, azken aukera hauen arteko diferentzia oso txikia delarik.

Adibideko planta nahiko motela denez, eta ondorioz sistemak *oversampling* handia erosotasunez kudeatu dezakeenez, modelo jarraituarekin konparaketetan egituraren moldaketaren emaitzak nabarmendu daitezen $100X$ gain-laginketa erabili da. Banda-zabalera 0,88 Hz-ekoa dela ikusita, $100 \cdot 0,88$ =88 Hz-eko laginketa-maiztasuna izango du sistemak, $T_s = \frac{1}{0,88\,\text{Hz}} = 0,011\,36\,\text{s} \approx 11\,\text{ms}$.

4.5. Modeloen erantzunen konparaketa

Aurreko atalean aztertu bezala, nahiz eta aukeratutako laginketa-maiztasuna modelo jarraituaren banda-zabalera baino ehun aldiz handiago izan, diskretuak ez du portaera berdin zehatza. Maiztasuna handitu ahala erantzunak berdintzen doazela ikusita, are handiagoa erabili genezake berdinketa osoa erdiesteko. Horrek, baina, konputazio arazoak sortu ditzake, kuantifikazio

arazoak areagotzen baitira.

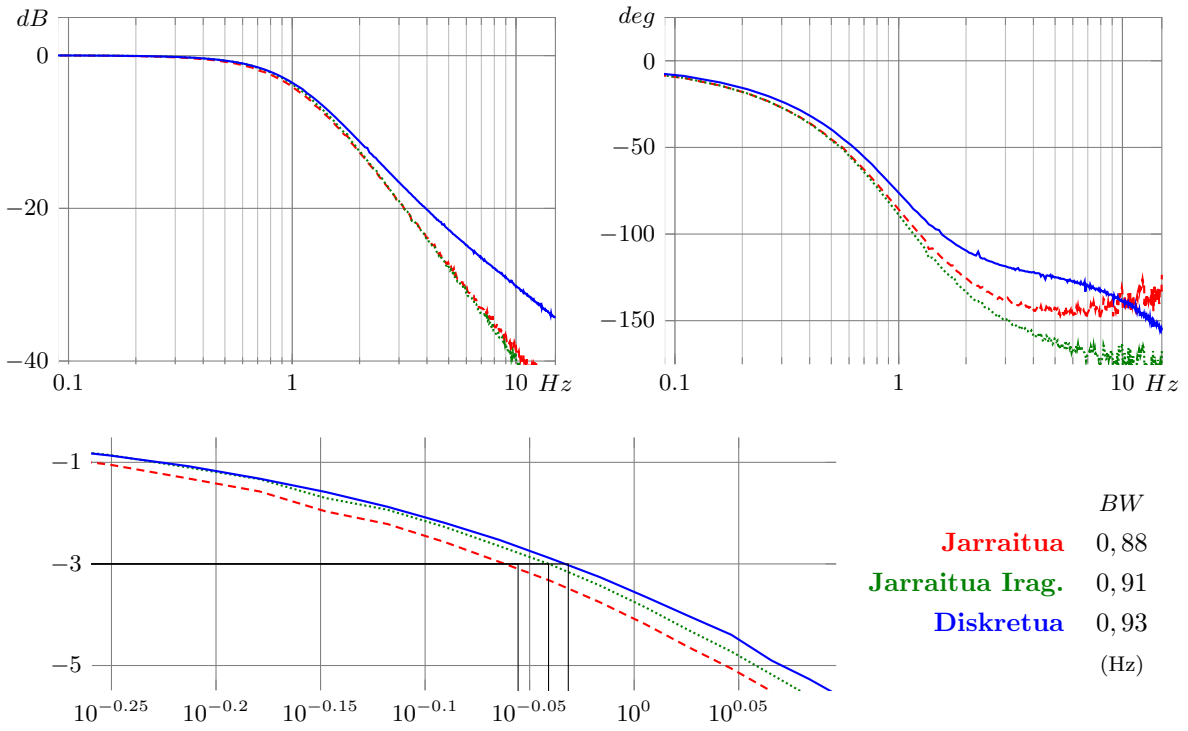
(1.4) transferentzia funtzioa inpropioa izanik praktikan ezin da inplementatu, eta beraz ezinezkoa izango da modelo diskretuak edo beste edozeinek praktikan honek aurkeztutako portaera izatea. Arazo hau adar deribatzaileak sortzen du, proportzionala propioa baita eta integratzailea hertsiki propioa. Zuzentzeko, adar deribatzailea behintzat lehen mailako behe-pasarekin iragazi daiteke [18, pp. 219-221]. Egitura idealean (1.16)ek adierazi bezala eraldatu ohi da.

$$\frac{U(s)}{E(s)} = K_p \left(1 + \frac{1}{T_i \cdot s} + \frac{T_d \cdot s}{\frac{T_d}{N} s + 1} \right) \tag{1.16}$$

Beste aukera bat, daudenetatik orokorrena eta besteek izan ditzaketen eragozpenak ekiditen dituena [23, ch. 2], kontrol seinale osoa iragaztea da. Aukera hau aukeratutako egitura paraleloarekin zuzenean bateragarria da.

$$\frac{U(s)}{E(s)} = K_p \left(1 + \frac{1}{T_i \cdot s} + T_d \cdot s \right) \frac{1}{T_f \cdot s + 1} = \left(K_p + \frac{K_i}{s} + K_d \cdot s \right) \frac{1}{T_f \cdot s + 1} \tag{1.17}$$

(1.17) propioa izanik, eta praktikan inplementagarria, (1.4) eta (1.14) adierazpenekin konparatu da (1.16. irudia).



1.16. Irudia: *Modelo jarraitu eta diskretuen maiztasun-erantzunen konparaketa.*

Diskretuak, bere egituragatik, termino deribatzaile iragazia du, algoritmoak sarreraren bi laginketa balio hartzen baititu. Honekin bat dator, beraz, $T_f = 0,01$ izanik, banda-zabaleraren

azterketan iragazitako modelo jarraituak beste bien arteko emaitza ematea.

Maila sarrerari hiru erantzunak aztertuz oso antzekoak direla ikusi daiteke (1.17. irudia). Modelo diskretuak (1.14) egonkortze denboraren %30-era arte jarraituaren (1.4) erantzun berdina ematen du, eta %35-etik aurrera iragazitako jarraituarena (1.17).

Proiektuaren helburua kontrolagailua inplementatzea denez eta agertutakoak aintzat hartuta, iragazitako modelo jarraitua erabili da azterketa zehatzagoetan erreferentzia gisa.

Adarrak banan-banan aztertuta, proportzionalak hiru modeloetan berdinak direla ikusi daiteke. Integratzailean, aldiz, bihurketa burututakoan antzemandako errorea agertzen da modelo jarraituen eta diskretuaren artean. Deribatzaileari dagokionez, diskretuaren iragazketa argi antzematen da.

4.6. Koma finkoan hitz-luzeraren azterketa

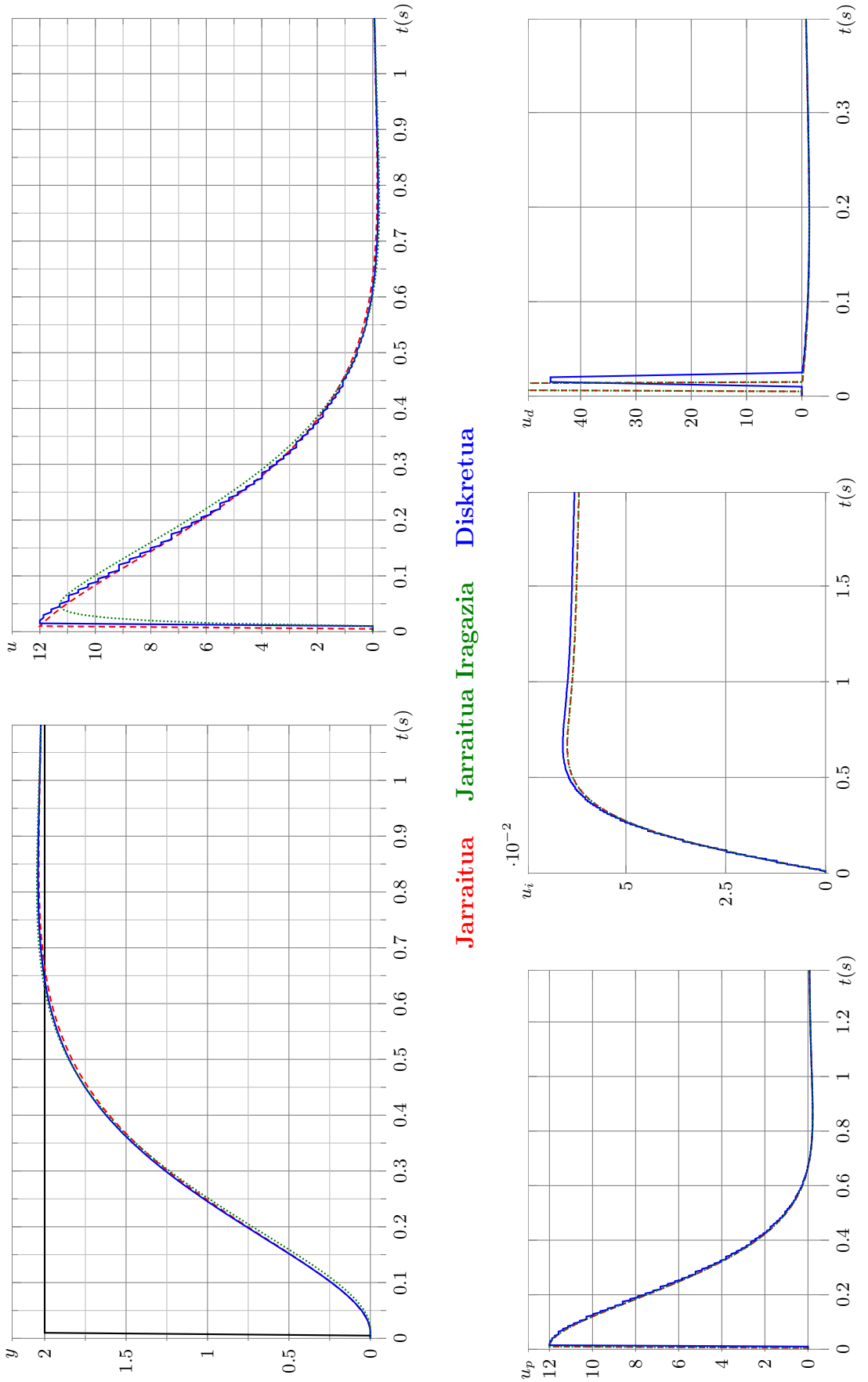
Orain arte egindako froga eta azterketa guztietan *Simulink*ek eskainitako blokeak erabili dira, *Laplace*ren S domeinuan -(1.4) eta (1.17)- zein Z planoan -(1.14)-. Kalkuluak, ondorioz, makina zehaztasunarekin[36] egin dira, programak eta ordenagailuak ahal duten zehaztasun handienarekin.

Helburua sistema deskribatzea denez, kalkuluak burutzeko erabiliko den aritmetikaren egitura definitu behar da. *Whitworth et al.*ek [24] *IEEE 754: Standard for Binary Floating-Point Arithmetic*[37] estandarrean bildutako koma finkoa eta koma higikorra izeneko adierazpen moduen arteko ezberdintasunak azaltzen ditu. *Frantz et al.*ek [25] egindako konparaketan ondorioztatu bezala, egun kostu eta erabilera aldetik ezberdintasun gutxi dago biak darabiltzaten txertatutako zirkuituetan. Koma higikorra aritmetika erreala, zehaztasun handia edo tarte dinamiko zabalagoa behar dituzten erabileretan hobesten da. Aurrekoak behar ez dituztenetan orokorrean koma finkoa erabili ahal delarik.

*Lerner*ek [26] proposatutako azterketa eta azaldutako adibideek argi uzten dute proiektu honen helburu den kontrolagailuak ez duela koma higikorreko aritmetika erabilerarik eskatzen. Are gehiago, FPGAk sintesiaren araberako egitura izanik, tarte dinamikorik gabeko eta hain eragiketa sinpleak burutu behar dituen kontrolagailua koma finkoan inplementatzeak azalerari begira silizio gutxiago behar du eta kontsumoa ere txikiagoa da. Paraleloan deskribatzeko aukera erabilita kalkuluak azkarrago burutzen ditu.

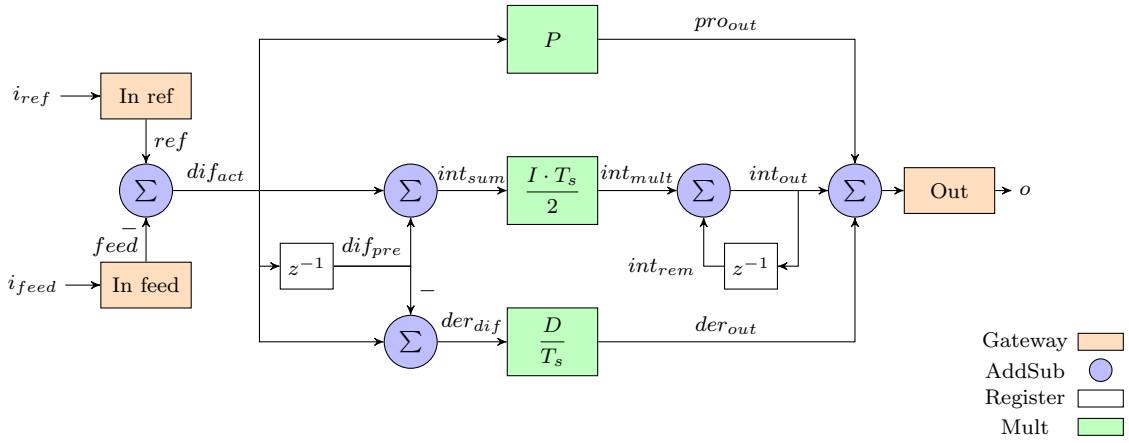
Koma finkoan arituta, seinaleen hitz-luzeraren azterketa ezinbestekoa da, interpretazioa zuzena izan dadin. Helburu hori lortzeko *Simulink*ek *Data Type Conversion* izeneko blokeak eskaintzen ditu. Seinale batek bloke hori zeharkatzean formatua aldatuko du. Programak koma finkoan lan egiteko formatua ezaguna du, eta hitz-luzera eta komaren kokapena ezartzea eskaintzen ditu. Zenbait ataletan, plantaren blokerako sarrera kasu, seinalea *double* motakoa izan behar da, eta aldaketa horiek ere kontuan hartu behar dira.

[36] Makina zehaztasuna: *floating point* eta *double precision*.
[37] http://grouper.ieee.org/groups/754/



1.17. Irudia: *Modelo jarraitu eta diskretuen denbora erantzunen konparaketa.*
u_p, u_i eta u_d seinaleak adierazten dituzten emaitzak eskuratzeko hiru egituretan e seinale berdina sartu da.
y eta u seinaleenak, aldiz, simulazio berezietan egin dira.

Aldi berean, *Xilinx ISE Design Suite*ak *System Generator* baliabidea du, *Matlab*ek eta aipatutako *Suite*ak simulazioak batera egin ditzaten. Aplikazioak instalatu eta konfiguratu ondoren, *Simulink*eko liburutegian *System Generator*ekin lan egiteko *Xilinx Blockset*, *Xilinx Reference Blockset* eta *Xilinx XtremeDSP Kit* bloke sorta berriak agertu dira. Horiek erabilitakoan, eta *System Generator*ekin lan egiteko argibideak jarraituta [27], simulazioak burutzean *Xilinx*en aplikazioak dagozkion blokeek egin beharreko kalkuluak burutzen ditu, eta *Matlab*ek *Simulink*eko blokeek egin beharrekoak.



1.18. Irudia: *PID diskretuaren (1.14)* Simulink *modeloa* Xilinx Blockset *erabilita.*

*System Generator*ek eskainitako baliabideak erabiltzean simulazioak bit-zehatzak eta ziklo-zehatzak dira [28, p. 17], hau da, simulazioak hardwarearen erantzunak[38] aurkezten ditu. Ondorioz, garapenari begira emaitza zehatzagoak eskaintzen ditu *Data Type Conversion* blokeek baino.

Simulink blokeek ez bezala, *System Generator*ek eskaintzen dituen blokeak erabiltzeak, zenbait bloke bereziren gehikuntza eskatzen du:

*System Generator*eko blokeak dauden modelo bakoitzean *token*a, bloke berezia, jarri beharra dago, *Xilinx*en aplikazioak burutu beharreko kalkuluak daudela adierazteko eta *Simulink*ekin komunikatzeko beharreko parametroak ezartzeko -nahiz eta lehenengo froga hauetan ezer konfiguratzeko beharrik ez izan-.

Bestetik, aplikazio biek kudeatu beharreko seinaleen arteko konexioa dagoen puntu guztietan datuak bihurtzeko blokeak gehitu behar dira. Kontrolagailuaren sarreretara heltzen diren *double* motako seinaleak *Xilinx*en aplikazioarentzat ezaguna den koma finkoko formatura bihurtzeko *Gateway In* blokeak erabili dira[39]. Irteeran, aldiz, *Gateway Out* blokea gehitu da, kontrolagailuaren koma finkoko irteera *double*era bihurtzeko.

[38] Blokeak nola sintetizatuko diren kontuan hartzen du, inplementazioaren osteko atzerapenak arbuiatuz.

[39] Bloke honetan laginketa-maiztasuna adieraztean, honek orain arte erabilitako sistemaren erreferentzia-modelo diskretuan (1.11. irudia) *Zero Order Hold* blokeak betetzen zuen funtsa ordezkatzen du. Beraz, ez da horrelako blokerik jarri behar.

Sarrerako eta irteerako blokeak behar bezala kokatuta, eta *System Generator*en *token*a jarrita, modeloa *Xilinx Blockset* sortako blokeekin osatu da. Oinarrizko blokeen artean beharreko bloke guztiak daude eskuragarri: *AddSub*, *Register* eta *Mult*. *Gateway Out* eta *Register* blokeek ez ezik, beste guztiek irteera mota definitzeko aukera ematen dute. *Gateway In*en kasuan, irteera nolakoa den definitzea beharrezkoa da, hori baita bloke horren funtsa (laginketa-maiztasuna zehazteaz gain). Besteetan, nahi den zehaztasunaren arabera *Full precision* (aplikazioak berak kalkulatzen du irteeran izan beharreko hitz-luzera) eta *User defined* aukerak daude. Bigarrena aukeratuz gero, hauek dira hautatu daitezkeenak:

- Output type
  - Boolean
  - Signed(2's complement)
  - Unsigned
- Number of bits
- Binary point
- Quantization
  - Truncate
  - Round
- Overflow
  - Wrap
  - Saturate
  - Flag as error

Aipatutakoez gain, bloke bakoitzak oinarrizko aukera bereziak ditu, blokearen nolakotasunaren araberakoak:

- Gateway In
  - Sample period
- AddSub
  - Operation
    - * Adition
    - * Substraction
    - * Adition or substraction
  - Optional Ports
    - * Provide carry-in port
    - * Provide carry-out port
    - * Provide enable port
- Register
  - Initial value
  - Optional Port
    - * Provide synchronous reset port
    - * Provide enable port
- Mult
  - Constant
    - * Value
    - * Number of bits
    - * Binary point
  - Optional Ports
    - * Provide enable port

Matlab eta *Xilinx ISE* aldi berean erabiltzeak, simulazioak burutu ostean eta emaitzak zuzenak direla frogatuta, *System Generator* blokeekin egindako modeloak VHDL kodera automatikoki bihurtzeko aukera ematen du. Hori dela eta, bloke guztiek *Implementation* aukerak dituzte. *Mult* blokearen kasuan RAM banatua edo bloke RAMa erabili nahi den edo/eta *Core*a erabili ordez HDL bidez deskribatzea nahi den hautatu daiteke, adibidez. Proiektuaren helburua kontrolagailua bera VHDL lengoaian deskribatzea denez, horrelako baliabideak erabiltzeak ez du zentzurik, eta, ondorioz, aukera horiek ez dira kontuan hartuko.

S planotik Z planora bihurketa egitean adar bakoitzeko biderketa koefizienteak aldatu behar

dira, laginketa-maiztasunaren eragina kontuan hartzeko. Atal proportzionalean, irabazia izanik, ez dago moldaketaren beharrik. Deribatzailea laginketa-periodoaz zatitu da, eta integratzailea, aldiz, biderkatu. Kalkuluak eskuz egin behar ez izateko, frogak egin bitartean modelo guztien parametroak aldi berean aldatu ahal izateko eta, aldi berean berdinak izango direla ziurtatzeko, balio zehatzen ordez koefizienteak kalkulatzeko adierazpenak jarri dira *Mult* blokeetan. 1.4. taulan agertzen diren parametroak erabili dira, *Matlab* lan-inguruneko aldagai gisa definituz. Honela, aipatutako aldagaiak (P, I, D edo/eta Ts) aldatuz gero zuzenean aldatuko dira balioak modelo guztietan eta dagozkion blokeetan. *System Generator*en blokeen zehaztasuna mugatuta dagoenez (hitz-luzera ezarria baitute) konstanteek izango duten balioa ez da zehatza izango, baizik eta hitz-luzera horrekin adierazi daitekeen gertukoena[40].

Hainbat froga burutu ondoren, 1.8. taulan adierazten diren konfigurazioak erabili dira. Sarrerako hitz-luzera 12 bitetan ezarri da, koma bitarra zortzigarrenean kokatuta. Hortik aurrerako blokeen irteeretan zehaztasunik galdu gabe eta baliabide ahalik eta murritzenak erabiltzea ahalbidetzen duten konfigurazioak aukeratu dira.

Ikus daitekeenez, batutzaile/kentzaile guztietan *Full* zehaztasuna erabili da. Horrek, sarrerek luzeera berdina izanik, irteerak bit bat gehiago izango duela kalkulatzen du. *int_out* batutzailea da irteera erabiltzaileak definituta duen bakarra. Berrelikadura positiboa duenez, *windup* efektua jasateko arriskua dauka, edo berdina dena, denbora oso txikian balio oso altuak hartzea. Horregatik, bit bat gehiago jarri da eta *overflow* egotekotan irteera saturatzeko adierazi. Batutzaile honek integrazio *memoria*ren funtsa betetzen du: irteeran hitz-luzera handiagoa jarriko bagenu, integrazio denbora luzeagoa izango litzateke; laburtzean, aldiz, integrazioa laburragoa litzateke.

Biderkatzaileetan, bestetik, zehaztasun osoa aukeratu da eta konstanteentzat emaitzak balio zehatzetik asko ez aldentzeko izan beharreko hitz-luzerak adierazi dira. 1.9.a taulan hitz-luzera mugatua izateagatik konstanteek izan beharreko balio zehatzarekiko zenbateko diferentzia duten adierazten da.

Ezarritako konfigurazio eta baldintzekin, programak 1.9.b taulan agertzen diren irteerako seinaleak definitu ditu. VHDL kodea idazterakoan bertan batutzaile bik osatzen duten funtzioa adierazpen bakar batean egin da, eta biderkatzaileen irteeren baldintzak aldatu dira. Baina, momentuz, taulan adierazitakoak dira eredu.

Maila sarrerari erantzunari dagokinez eta 1.19. irudiak adierazten duenez, modelo diskretuaren portaera berdina du. Biek iragazitako modelo jarraitua baina erantzun zertxobait hobea dute, gaindiketa txikiagoarekin eta erantzun iragankor azkarragoarekin.

Nahiz eta hitz-luzera mugatu, parametroekin adierazpenak askatu eta *Xilinx*en bloke zehatzak erabilita, azkeneko modelo honek modelo jarraituarekiko duen ezberdintasuna diskretu izateagatik dela iritzi daiteke, egindako pausu guztiek eragindako erroreak oso txikitzat hartuz.

[40] Kalkulua eta hurbilketa programak berak egiten du aldagaien balioak hartuta eta adierazpenak askatuz.

Blokea	Mota	Fitxa	Aukera	Balioa
Gateway In ref Gateway In feed	Gateway In	Basic	Output type	Signed (2's comp)
			Number of bits	12
			Binary point	8
			Quantization	Round
			Overflow	Saturate
			Sample period	Ts
dif_act	AddSub	Basic	Operation	Substraction
		Output Type	Precision	Full
P	Mult	Basic	Value	P
			Number of bits	3
			Binary point	0
		Output Type	Precision	Full
int_sum	AddSub	Basic	Operation	Addition
		Output Type	Precision	Full
$\frac{I \cdot Ts}{2}$	Mult	Basic	Value	I*Ts/2
			Number of bits	14
			Binary point	14
		Output Type	Precision	Full
int_out	AddSub	Basic	Operation	Addition
		Output Type	Precision	User defined
			Output Type	Signed (2's comp)
			Number of bits	29
			Binary point	22
			Quantization	Round
			Overflow	Saturate
der_dif	AddSub	Basic	Operation	Substraction
		Output Type	Precision	Full
$\frac{D}{Ts}$	Mult	Basic	Value	D/Ts
			Number of bits	11
			Binary point	6
		Output Type	Precision	Full
add_out1 add_out2	AddSub	Basic	Operation	Addition
		Output Type	Precision	Full

Adierazten ez diren aukeretan balio izendatua mantendu da

1.8. Taula: System Generator*ekin aritzeko* Xilinx Blockset *sortako blokeen konfigurazioa.*

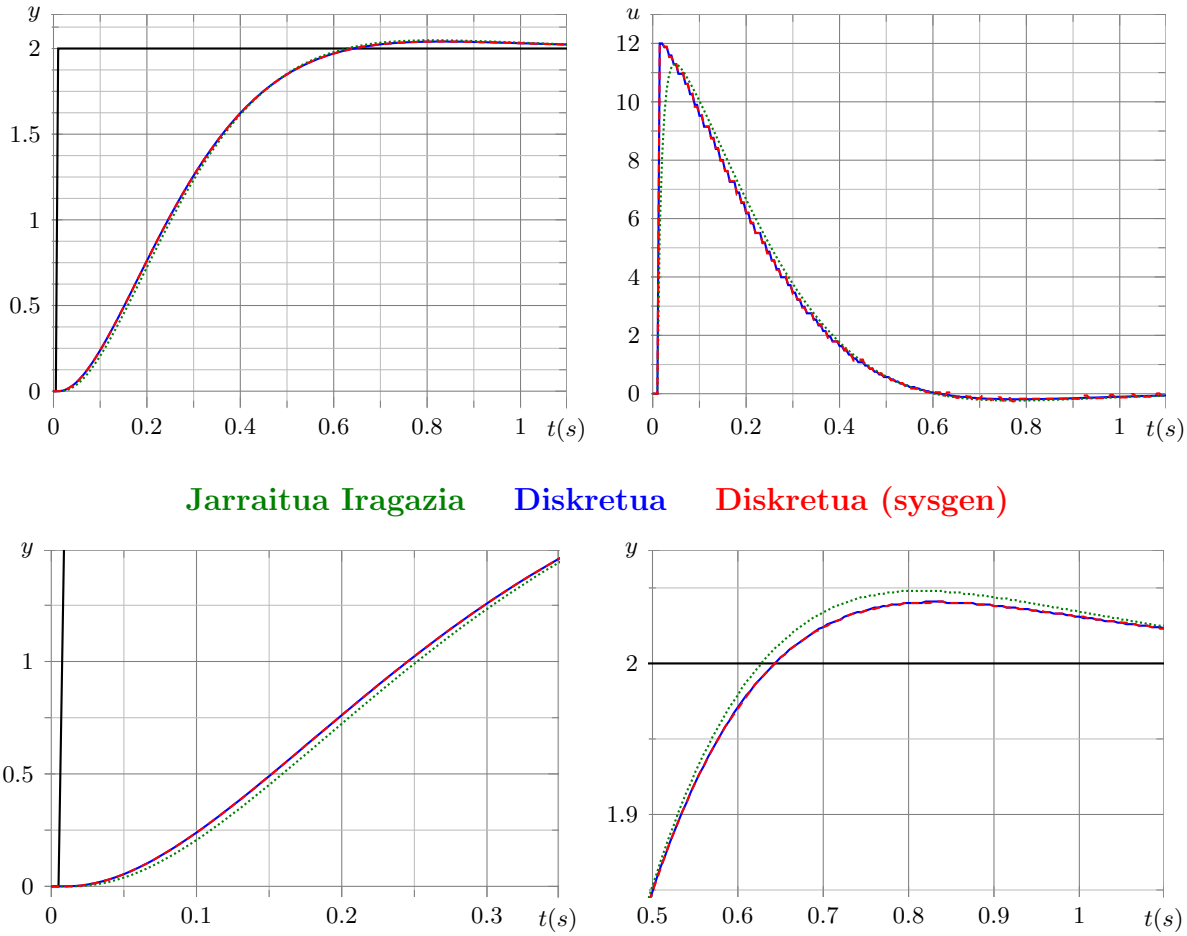
Zehatza	Konstantea
6 $K_p = P$	6 $\Delta 0$ $\%0$
$6,875 \cdot 10^{-4}$ $K_i = \frac{I \cdot Ts}{2}$	$6,714 \cdot 10^{-4}$ $\Delta 1,61 \cdot 10^{-5}$ $\%2,3418$
$22,7272$ $K_d = \frac{D}{Ts}$	$22,73$ $\Delta 2,73 \cdot 10^{-3}$ $\%0,012$

(a) Mult *blokeen konstanteen balioak*

Blokearen izena	Seinale mota
Gateway In ref	Fix_12_8
Gateway In feed	Fix_12_8
dif_act	Fix_13_8
dif_pre	Fix_13_8
Kp	Fix_16_8
int_sum	Fix_14_8
Ki	Fix_28_22
int_out	Fix_29_22
int_mem	Fix_29_22
der_dif	Fix_14_8
Kd	Fix_25_14
add_out1	Fix_26_14
add_out2	Fix_35_22

(b) *Irteeren seinale mota*

1.9. Taula: Xilinx Blockset *sortako blokeen konstanteak eta irteerak.*



1.19. Irudia: System Generator *blokeez egindako modeloaren eta aurrekoen arteko konparaketa.*

5.
Kontrolagailua deskribatzea (VHDL)

*System Generator*ek DSP blokeak erabilita, hau da, VHDL kodea sintetizatzeko erabiliko den softwareak eskainitako blokeak erabilita, VHDL lengoaia erabili gabe eskuratu daitekeen simulazio fidagarriena eskuratu da. Horrek kontrolagailuaren barne egituraren bloke funtzionalak eta horien arteko loturak eskaintzen ditu. VHDL deskribapena egiteko egitura berdina erabili da erreferentzia gisa, aurreko atalean egindako azterketaren ondoren eskuratutako hitz-luzeren datuak ere erabilita.

5.1. Liburutegiak

VHDL seinale mota eta izaerek bortizki mugatutako lengoaia da. *Murata et al.*ek [29] adierazi bezala, honen ondorioz lengoaiaren konpilazio zein sintesia seguruak dira, hau da, ez ditu emaitza ezegokiak eman ditzaketen operazio zein bihurketak egiten. Deskribatutako kodeak inplementatu nahi diren funtzioak bete ditzan, beraz, ezinbestekoa da seinale motak eta hauen operadoreak definitzen dituzten liburutegi eta paketeak ezagutzea.

VHDL lengoaiaren definizio estandarrak batzen dituen liburutegia *IEEE* da. Oinarrizko *std_ulogic* eta *std_ulogic_vector* motak eta hauen ebazpena diren *std_logic* eta *std_logic_vector* definizioak *std_logic_1164* paketeak biltzen ditu.

Zeinudun eta zeinurik gabeko seinaleak erabili behar direnez, aurrekoen asebetetakok diren *signed* eta *unsigned* definitzeko bi aukera daude. Nahiz eta industrian *std_logic_arith* berezko estandarra izan, *Lewis*ek [30] argitu bezala operadore aritmetikoak eta konparatzekoak baino ez ditu. *numeric_std* paketeak aurrekoez gain operadore logikoak definitzen dituenez, eta *IEEE Std 1076-2008* estandarraren parte denez, hau erabili da.

Proiektuaren garapenean *IEEE Std 1076-1993* estandarra jarraitu da. Honetan *numeric_std* ez da estandarraren parte, baina bai bateragarria. Aldi berean, kargatutako paketeak ez du *std_logic_vector* seinaleekin operazio aritmetikoak zein konparatzekoak egiteko aukerarik ematen. 2008ko estandarrean batutako baina 93koan banaturik dagoen *std_logic_unsigned* paketeak *unsigned* gisa hartzea ahalbidetzen du. Proiektuaren helburu nagusia ikasketa izanik, azkeneko pakete hau ez da kargatu. Honen ordez, diseinatzaileak bereziki adierazi beharko du sistemak noiz aztertu behar dituen *std_logic_vector* seinaleak *unsigned* bezala. Hau dela eta, erruz erabili dira *casting*ak interpretazioa adierazteko.

*Bishop*ek [31] aurkeztutako VHDL-93 arabera sintetizatu daitezkeen paketeek azkeneko berrikuspenetan gehitutako *fixed_float_types* eta *fixed_pkg* erabiltzea ahalbidetzen du. Erabiltzaile gidak [32] azaltzen denez, hauen bitartez esplizituki adierazi daiteke seinale batean koma bitarraren kokapena. Definizioak, baina, komak bektorearen tartearen barruan egon behar duela argitzen du. Honen ondorioz, erabiliko diren aldagaien eskalak geroz eta ezberdinagoak izan orduan eta informazio esanguratsurik gabeko bit gehiago erabiliko dira.

Aritmetikaren ikuspuntutik kontrolagailuaren egitura oso sinplea denez, bereizgarriak ez di-

ren datuak arbuiatuz baliabideen erabilera ahal beste optimizatu ahal izateko, ez da seinaleak koma finkoan adierazteko pakete berezirik erabili. Hauen ordez koma bitarraren kokapena jarraitu da blokeetan behar, desplazamendurako eta luzera ezartzeko operadoreak erabiliz.

VHDL-93 estandarreko *numeric_std* [33] paketearen *shift_right*, *shift_left* eta *resize* funtzioen argibideak jarraituz, hauek dira informazio esanguratsua babestuz koma bitarraren kokapenaren araberako moldaketak burutzeko erabilitako adierazpenak:

resize(shift_right(*signal*,n),$wl - n$)
: Pisu gutxieneko bitak galduz seinalea moztea

shift_right(*signal*,n)
: Luzera mantenduz koma bitarra eskumara mugitzea edo 2^n aldiz zatitzea

shift_left(resize(*signal*,$wl + n$),n)
: Koma hitza luzatuz ezkerrera mugitzea edo 2^n aldiz biderkatzea

Laburbilduz, *IEEE* liburutegiko *sdt_logic_1164* eta *numeric_std* paketeak baino ez dira kargatu. Burutuko diren eragiketak oso konplexuak ez direnez sintetizatzean programak berak aukeratuko ditu, *IP Core* bereziren instantzia barik. Hauen beharra kodearen bitartez adieraziko da, modu funtzionalean.

```
library ieee;
 use ieee.std_logic_1164.ALL;
 use ieee.numeric_std.all;
```

5.2. Entitatea

```
entity anie_pid is
```

5.2.1. *generic* parametroak

Kontrolagailuaren erabilera errazteko, eta konfigurazio denbora murrizteko, seinale guztiak parametrizatu egin dira. Honela, aldagaien (i_wl), irteeraren (o_wl), koefizienteen (k_wl) edota funtzio integratzailearen memoriaren ($irem\_wl$[41]) hitz-luzera aldatuz gero barne egitura moldatzen da.

Koefizienteei dagokienez, hirurentzat hitz-luzera berdina ezarri da (k_wl), diseinua behar baino korapilatsuagoa izan ez dadin. (1.14) adierazpeneko koefizienteen jatorria aztertuz, eta 1.9.a taulan agertutakoei begira, balioen magnitudeak zeharo ezberdinak direla ondorioztatu da. Zehaztasuna bermatu eta aldi berean behar baino baliabide gehiago ez erabiltzeko, koefiziente bakoitzaren koma bitarraren kokapena adierazteko parametro bana ezarri da: *pbp*, *dbp* eta *ibp*. Aldi berean, zehaztasun maximoa bermatzeko, dagokion atalean azalduko den *minbp* parametroa ezarri da.

[41] $irem\_wl$-ren funtsa *anti-windup* estrategiaren deskribapenean azaltzen da.

```
generic (
 i_wl: natural:=12; -- Kontrolagailuaren sarreren hitz-luzera
 o_wl: natural:=22; -- Kontrolagailuaren irteeraren hitz-luzera
 k_wl: natural:=8; -- Kontrolagailuen koefizienteen (Kp, Ki, Kd) hitz-luzeera
 irem_wl: natural:=1; -- Adar integrazailearen "memoria"ren luzera gehitua

 pbp: integer:=2; -- Kp koefizientearen koma bitarraren kokapena
 ibp: integer:=17; -- Ki koefizientearen koma bitarraren kokapena
 dbp: integer:=2; -- Kd koefizientearen koma bitarraren kokapena
 minbp: integer:=2 -- Adar baturan koma bitarrak lerrokatzeko parametroa
);
```

5.2.2. Atakak

```
port (
 clk,ce: in std_logic; -- Erloju eta erlojuaren gaitze seinaleak: T(m_ce)=Ts
 srst: in  std_logic; -- Reset sinkrono orokorra
 i_ref: in signed(i_wl-1 downto 0); -- Erreferentzia seinalea: (r)
 i_feed: in signed(i_wl-1 downto 0); -- Berrelikadura seinalea: (y)
 kp: in signed(k_wl-1 downto 0); -- Koefiziente proportzionala: Kp=P
 ki: in signed(k_wl-1 downto 0); -- Koefiziente integratzailea: Ki=I*Ts/2
 kd: in signed(k_wl-1 downto 0); -- Koefiziente deribatzailea: Kd=D/Ts
 o: out signed(o_wl-1 downto 0) -- Kontrol seinalea (u)
);
```

Nahiz eta orain arte erabilitako modeloetan erreferentzia sarrera eta berrelikaduraren arteko diferentzia, errorea, kontrolagailutik kanpo kalkulatu izan den, bukaerako sisteman FPGAk egiten du. Honela kanpo logikaren beharra murrizten da, eta planta ezberdinekin aritzea errazten du. Hori dela eta, kontrolagailuak bi aldagai izango ditu sarreran: erreferentzia (i_ref) eta berrelikadura (i_feed).

```
end anie_pid;
```

5.3. Arkitektura

```
architecture pid of anie_pid is
```

5.3.1. Osagaiak

Sarrerekin gertatu bezala, VHDL deskribapenean kontrolagailuak aurreko modeloetan irteeran agertu izan den saturazio blokea ere barnean izango du. Adar integratzailean ere, *windup* arazoa konpontzeko erabilitako estrategiaren arabera, saturatzeko beharra izan daiteke. Horregatik sarreran seinale bat jaso eta irteeran saturatutako seinale berdina ateratzen duen parametrizatutako osagaia deskribatu da.

```
component anie_sat
 generic (
  iref_wl: natural:=2; -- Saturatu nahi den seinalearen hitz-luzera
  osat_wl: natural:=2 ); -- Saturatutako seinalearen hitz-luzera
```

```
 port (
  iref: in signed(iref_wl-1 downto 0); -- Saturatu nahi den seinalea
  osat: out signed(osat_wl-1 downto 0) ); -- Saturatutako seinalea
end component;
```

Seinale bat saturatzea ezarritako tarte batetik kanpoko balioei minimo zein maximo zehatzak esleitzean datza. Hitz-luzerei begira, osagai honek minimo eta maximoari $\pm(2^{n-1}-1)$ balio finkoak esleitzen dizkie, hau da, biko osagarrian adierazi daitekeen modulu handiena. Hori dela eta, osagai honen funtzionamendua egokia izan dadin beharrezkoa da sarrerako seinalearen hitza irteerakoa baino luzeagoa izatea:

$$iref\_wl \geq osat\_wl + 1$$

Berdina balitz, saturazio osagaia erabiltzeak ez luke zentzurik izango, beti egongo baitzen sarrera mugen artean.

```
architecture sat of anie_sat is
```

Arkitektura, barne egitura, bina konparadorek eta bi sarrerako multiplexadorek osatzen dute. Konparadoreek sarrera zenbatesten dute zein tartetan[42] dagoen ondorioztatzeko. Informazio hori baliatuz, multiplexadoreek moztutako sarrerako seinalea, irteera minimoa edo maximo aukeratuko dute.

Hitz-luzeren arteko diferentzia *generic* parametroen menpekoa izanik, diseinuan zehar aldatu daiteke, baina inplementatzean finkoa da. Honela, konparadoreek sarreraren hitz-luzera dute, ref_wl. Multiplexadoreek, aldiz, irteerarena: sat_wl. Eginiko adierazpenak direla eta, nahiz eta hitz-luzera ezberdina izan, konparadore eta multiplexadore bikote bakoitzak sarrera batean modulu berdina izan behar du. Hori dela eta, mugentzat seinale bana definitu da, eta *resize* eragilearen bitartez luzerak moldatu dira osagaien sarrerek bat etor daitezen.

```
  signal max,min: signed(osat_wl-1 downto 0); -- Saturatutako seinalearen mugak

begin
```

Mugak konstanteak izanik, biko osagarrian dagozkien balioak parametroak kontuan izanik zuzenean esleitu zaizkie.

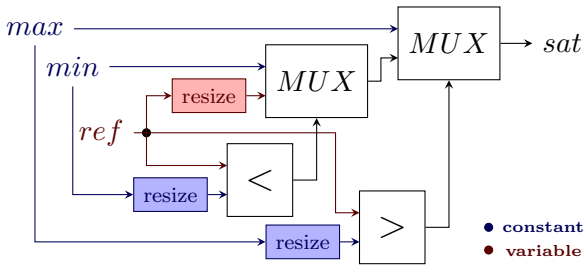
```
-- Saturazio maximoaren balioa ezartzea
max(osat_wl-1) <= '0';
max(osat_wl-2 downto 0) <= (others => '1');

-- Saturazio minimoaren balioa ezartzea
min(osat_wl-1) <= '1';
min(osat_wl-2 downto 0) <= (0 => '1', others => '0');
```

[42] $(-\infty, min)$, $[min, max]$ edo (max, ∞)

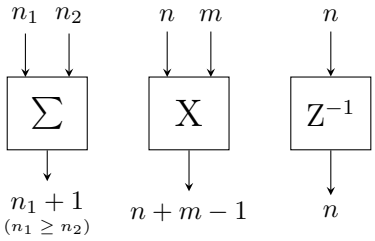
```
osat <=
  max when (iref>resize(max,iref_wl))
      else
  min when (iref<resize(min,iref_wl))
      else
  resize(iref,osat_wl);

end sat;
```



5.3.2. Seinaleak

*Cofer et al.*ek [34] adierazitako koma finkoko aritmetikan gertatu daitezkeen erroreak gogoratuz eta 1.8. taulan adierazitakoari jarraituz, seinaleak biko osagarrian adierazten direla aintzat hartuta zehaztasun osoa bermatzeko hurrengoak kontuan hartu behar dira:



- Biko osagarrian lan egiten duen batutzaile/kentzaile baten irteerak sarrera luzeenak baino bit bat gehiago izan beharko du.
- Biko osagarrian lan egiten duen biderkatzaile baten irteerak koefizienteen luzeren baturak baino bit bat gutxiago izan beharko du.
- Erregistro batek hitz-luzera berdina izango du sarreran eta irteeran.

Kodearen irakurketa eta ulermena errazteko, seinaleen hitz-luzerak entitatearen *generic* parametroek definitutako konstanteen bitartez deklaratu dira. Hurrenez hurren, sarrerako seinaleak kentzen dituen blokearen hitz-luzera (*aswl*), konstante proportzionalaren biderkatzailearen hitz-luzera (*mwl*) eta irteeraren saturatzailearen hitz-luzera maximoa (*oswl*) definitu dira.

```
constant aswl: natural:= i_wl+1;
constant mwl: natural:= i_wl+k_wl;
constant oswl: natural:= mwl+irem_wl+1;
```

Kontrolagailuaren arkitektura osatzen duten seinale guztien laburpena 1.10. taulan aurkitu daiteke. *overflow* arazoak ekiditeko, bertan adierazitako informazioaren arabera batuketa/kenketak burutu baino lehen[43] eragigaien luzera behartu da, *resize* funtzioa medio.

```
begin
```

[43] a eta b eragigaien luzera n izanik, *resize*$(a \pm b, n+1)$ eragileak luzatzea eragiketaren ostean burutzen du, *overflow* arazoak izateko arriskuaz.

id	msb	comment	id	msb	comment
ref	i_wl-1	$r(k)$	int_{mult}	mw	$K_i \cdot (e(k) + e(k-1))$
$feed$		$y(k)$	der_{out}		$K_d \cdot (e(k) - e(k-1))$
dif_{act}	aswl-1	$e(k)$	int_{out}	oswl-1	$u_{isat}(k)$
dif_{pre}		$e(k-1)$	int_{rem}		$u_{isat}(k-1)$
			out_{sat}		$u_{sat}(k)$
int_{sum}	aswl	$e(k) + e(k-1)$			
der_{dif}		$e(k) - e(k-1)$	int_{tosat}	oswl	$u_i(k)$
			out_{tosat}		$u(k)$
pro_{out}	mwl-1	$K_p \cdot e(k)$			

```
signal id: signed(msb downto 0); --comment
```

1.10. Taula: *Kontrolagailuaren arkitekturan deklaratutako seinaleak eta hitz-luzerak.*

Kontrolagailuaren deskribapena diseinuaren estrukturaren arabera egin da, datu fluxua jarraituz. Hau dela eta, 1.20. irudian aurkezten den kontrolagailuaren deskribapenaren adierazpen grafikoak, aurreko ataletan landutako modelo diskretuaren (1.11. irudia) eta *System Generator*en blokeez egindako modeloaren (1.18. irudia) antza handia du, ezberdintasun nagusia saturazio blokeak izanik.

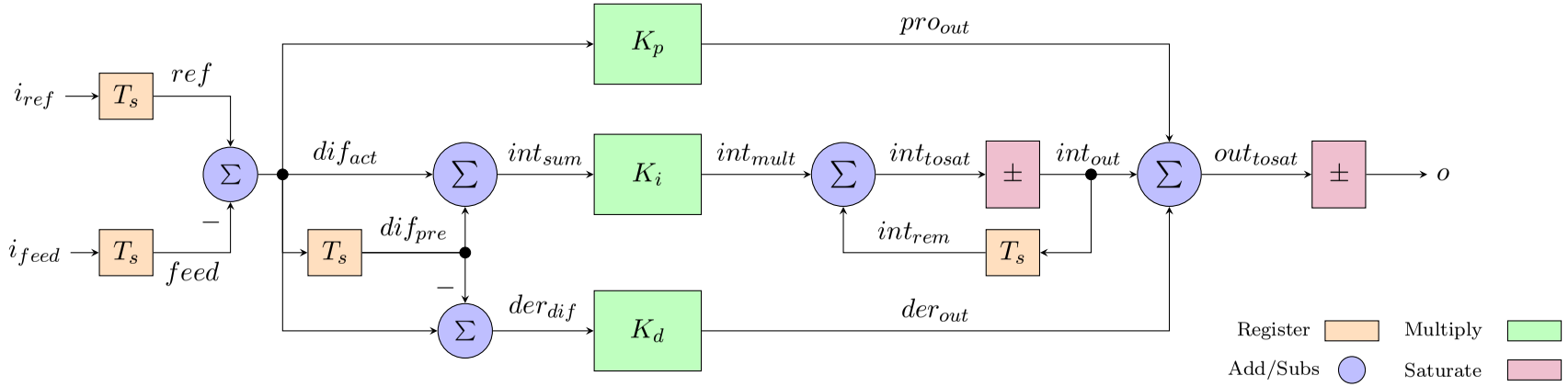
5.3.3. Anti-windup estrategia

*System Generator*en blokeekin egindako modeloan int_{out} da irteeraren zehaztasuna bereziki definituta duen bakarra (1.8. taula). Horren zergatia berrelikadura positiboa da:

Aurreko laginketetako irteera kontuan hartzeko, adar integratzaileak batugailua du irteeran, azken laginketa eta aurrekoak batzen dituena. Maila aldaketak gertatzen direnean, batugailu horren modulua oso azkar igotzen da, plantaren eragingailuak dituen mugak gaindituz eta *windup* deritzon efektua sortuz. Honek kontrolagailuak agindutako eta eragingailuak eragindako balioak ezberdinak izatera bultzatzen ditu. Errorea txikitzean berriz ere bat egiteko beharko duten denboran izango du honek eragina, luzatuz eta kontrola motelduz.

Adar deribatzailearen azterketan gertatu bezala, honetan ere arazoa ekiditeko oinarrizko egitura ideal jarraitua moldatzen duten proposamen ugari egin dira [23, ch. 3]. Ahal den heinean kontrolagailuaren linealtasuna bermatu eta bestalde egoera horretatik bueltatzeko beharreko denbora ahal beste murrizteko, berrelikadura gehigarriak eta parametro berrien doiketa eskatzen dituzte gehienek. Hala ere, ez dago orokorrean erantzun hoberik ematen duenik, kasuan kasuko azterketa gomendatzen delarik.

Oinarrizko transferentzia funtzioaren moldaketak kontrolagailuaren garapena behar baino gehiago nahastuko duelakoan, egitura sinpleagoa hobetsi da: arazoak sortu ditzaketen seinaleen




```
-- r(k) - y(k)
dif_act <= resize(ref,aswl) - resize(feed,aswl);

-- e(k)+e(k-1)
int_sum <= resize(dif_act,aswl+1) + resize(dif_pre,aswl+1);
-- e(k)-e(k-1)
der_dif <= resize(dif_act,aswl+1) - resize(dif_pre,aswl+1);

-- up(k)=Kp*e(k)
pro_out <= shift_right(resize(kp*dif_act,mwl),pbp-minbp);
-- Ki*(e(k)+e(k-1))
int_mult <= shift_right(resize(ki*int_sum,mwl+1),ibp-minbp);
-- ud(k)=Kd*(e(k)-e(k-1))
der_out <= shift_right(resize(kd*der_dif,mwl+1),dbp-minbp);

-- ui(k)=Ki*(e(k)+e(k-1))+ui(k-1)
int_tosat <= resize(int_mult,oswl+1) + resize(int_rem,oswl
    +1);

-- u(k)=up(k)+ud(k)+uisat(k)
out_tosat <= resize(pro_out,oswl+1) +
             resize(int_out,oswl+1) +
             resize(der_out,oswl+1);

-- Kontrol seinalea moztea
o <= out_sat(oswl-1 downto oswl-o_wl);
```

$$K_p = P \qquad K_i = \frac{I \cdot Ts}{2} \qquad K_d = \frac{D}{T_s}$$

$$pro_{out} \quad U_p(k) = Kp \cdot E(k)$$

$$int_{out} \quad U_i(k) = Ki \cdot (E(k) + E(k-1)) + U_i(k-1)$$

$$der_{out} \quad U_d(k) = Kd \cdot (E(k) - E(k-1))$$

```
-- ui(k) -> uisat(k)
INTSAT: anie_sat
  generic map (oswl+1,oswl)
  port map (int_tosat,int_out);

-- u(k) -> usat(k)
OUTSAT: anie_sat
  generic map (oswl+1,oswl)
  port map (out_tosat,out_sat);
```

1.20. Irudia: *Kontrolagailuaren algoritmoaren deskribapen estrukturala.*

balioak mugatu eta saturatzea (moztean informazio baliagarria galduko baikenuke). Nahiz eta aukera honek kontrolagailua lineala den tartea mugatzen duen, *oversampling* handia erabiltzen denez, horren ondorioz gerta litezkeen diferentziak behar baino gehiagotan lagindu eta, hortaz, zuzenduz konpentsatzen direla onartu da.

Osagai gisa deklaratutako saturazio blokea baliatuz, int_{out}ek adar integratzailearen biderkatzailearen irteera baino $irem_{rem}$ bit gehiagora saturatzen da. *Generic* motako parametro honek, beraz, kontrolagailuaren integrazioak aurreko balioak kontuan hartzeko zenbateko ahalmena duen adierazten du. Beste era batera adierazita: funtzio integratzaileak zein pisu izango duen sistemaren dinamikan. $irem_{rem}$ek natural eta positiboa izan behar du.

Kontrolagailuaren irteera eta eragingailuak eragindakoa berdinak izan behar duten neurri berean, kontrolagailuaren barneko adarrek irteerarekin bat egin behar dute. Hau dela eta, irteeraren hitz-luzera batugai luzeenaren arabera definitu da. Egoera honetan baliteke, adar integratzailea saturazioan egonik, proportzionala eta deribatzaileekin baturak ahal baino erantzun handiagoa eskatzea [35]. Nahiz eta oraingoan efektua zeharo txikiagoa izan, hau ere ekiditeko, berriz ere erabili da saturatzeko osagaia.

Honela, adar integratzaileak beti izango du irteera saturatzeko aukera, baina emaitza mugatuta egongo da, eragingailuaren ahaleraren arabera lan-tartean mantentzeko, eta praktikan eman daitezkeen erantzunak adierazteko.

5.3.4. Koma bitarrak lerrokatzea

Batuketa zein kenketa aritmetikoak burutzean, hitz-luzerei buruzko aurretiaz adierazitakoez gain, seinale guztien koma bitarrak toki berean egon beharko dute -eragiketa hamartarretan gertatzen den bezala-. Hiru adarrek sarrera berdinak izanik (r eta y), irteerako baturan termino bakoitzaren irteerak duen komaren kokapena konstanteak baldintzatzen du. Honela, erreferentziako seinalearekiko komaren desplazamendua *pbp*, *dbp* eta *ibp* parametroek adierazten dute, hurrenez hurren.

Informazio esanguratsuena babesteko, seinaleak eskumara baino ezingo dira mugitu, garrantzi gutxien duten bitak galduz. 1.20. irudian agertzen denez, *shift_right* eragilea erabili da parametroek eragindako desplazamendua atzera egiteko. Ahalik eta informazio gutxien galtzeko, desplazamendu txikiena jasan duen adarra hartu da erreferentzia gisa. VHDL lengoaiaren 2008 baino lehenagoko bertsioek minimoa kalkulatzeko funtziorik ez dutenez, erabiltzaileak adarren koefizienteen komaren kokapena adieraztean minimoa (*minbp*) zein den adieraziko du. Sarrerako seinaleen kokapen berdina nahi izatekotan, 0 adieraziko da.

Deskribapenean komaren zuzenketa biderketa burutu bezain laster eragiten da, irteerako batuketan egin ordez. Nahiz eta adar proportzional zein deribatzailean bi aukerek emaitza berdina eman, integratzailearen kasuan premiazkoa da seinalea saturatu baino lehen egitea. Honela ez balitz, saturazio maximo eta minimoa komaren zuzenketaren arabera aldatuko lirateke, modulua txikituz hain zuzen ere.

5.3.5. Irteera moztea

Adarrak batu ostean, irteerak *oswl* hitz-luzera izango du. Adierazitako irteera hitz-luzera txikiagoa izanez gero, pisu gutxieneko bitak moztuko dira. Hau da, *msb*-tik hasita *o_wl* bit hartuko dira. Deskribapenari begira, parametroak definitzean kontuan hartu beharreko baldintza ondorioztatu daiteke, zehaztasun osoa izateko adierazpenean berdinketa izan behar delarik:

$$\text{o\_wl} \leq \text{i\_wl} + \text{k\_wl} + \text{irem\_wl} + 1$$

5.3.6. Lagintzea eta erregistroak berritzea

Egituran adierazitako bloke sekuentzial guztiek maiztasun berdinarekin berritzen dituzte balioa, *clk* erloju seinalea eta *ce* gaitze seinalearen maiztasun eta egoeren arabera baldintzatuta dagoelarik. Aplikazioaren arabera, T_s periodoa duen *clk* seinalea izango da eta *ce* maila logiko altura finkatuta, edo $> T_s$ periododun erloju seinalea eta Ts periodoaz gaitzen den *ce*.

```
-- Fs
process(clk,ce)
begin
if rising_edge(clk) then
 if srst='1' then
  ref <= (others => '0');
  feed <= (others => '0');
  dif_pre <= (others => '0');
  int_rem <= (others => '0');
 elsif ce='1' then
  ref <= i_ref;
  feed <= i_feed;
  dif_pre <= dif_act;
  int_rem <= int_out;
 end if;
end if;
end process;
```

```
end pid;
```

6.
VHDL deskribapenaren ko-simulazioa

6.1. Blackbox

Kontrolagailua VHDL lengoian deskribatuta dagoela, erantzuna aurrekoekin aldaratuta nolakoa den aztertzeko, *System Generator*ek eskeinitako liburutegietako *Blackbox* blokea erabili da. Bloke horrek *Simulink* barneko bloke baten barnean VHDL kodea estekatzea ahalbidetzen du. Kodea *Xilinx*en *ISE*k aztertu eta sintetizatzen du, eta sarreren araberako irteerak bidaltzen dizkio *Simulink*eri.

Bloke hau erabili baino lehen, aurreko atalean deskribatutako kodean zenbait aldaketa burutu dira, aplikazioen arteko komunikazioa, eta ondorioz simulazioa, zuzena izan dadin:

- Erloju seinaleak eta horren gaitzeak $m\_$ aurrizkia izan behar dute. Identifikatzaileak aldatu dira.
- Sarrerak/irteerak *std_logic_vector* motakoak izan behar dira. Sarrerei/irteerei m aurrizkia gehitu zaie, *std_logic_vector* motara aldatuz, eta aldi berean aurreko izenak dituzten *signed* motako seinaleak definitu dira, hurrenez hurren lotuz.
- Simulazioan indeterminazioek eragindako erroreak ekiditzeko, seinale guztiak deklaratzean `:=(others=>'0')` abiarazte sententzia adierazi da.
- *Gateway In* blokeek sarrerak erregistratzen dituztenez, i_ref eta i_feed zuzenean esleitu zaizkie ref eta $feed$ seinaleei, atal sekuentzialetik kanpo. Barruan utziz gero, bi aldiz eragingo genuke z^{-1} eta ondorioz $k-1$ eta $k-2$ laginketekin burutuko genituzke kalkuluak. Horrek portaera okerragotuko luke.
- Seinaleak simulaziorako balio zehatz batekin abiarazi direnez, $srst$ sarrera eta honen eragina kendu dira.

Hitz-luzeraren lehenengo azterketan ez bezala, kontrolagailua VHDL lengoaian deskribatzean koefizienteak banan banan aztertu daitezke, aurrezarritako blokeen aukerek mugatu barik. Honela, zortzi biteko hitz-luzerarekin balio zehatzetik %1 baino gutxiago aldentzen diren emaitzak lortu dira (1.11. taula). Hau dela eta, aurreko modeloarekin aldaratuta biderkatzaileen osteko seinale guztien hitzak laburragoak izango dira.

Ko-simulazioa burutzeko *Blackbox* blokeak konfigurazio *.m* funtzioa duen fitxategia behar du. Blokea modeloan sartzean programak zein VHDL fitxategi esleitu nahi den galdetzen du, eta adierazitakoan konfigurazio fitxategi bat sortzen du. Arkitektura sinpleekin eta hitz-luzera zehatzeko deskribapenekin, zuzenean hartzen ditu parametro guztiak. Garatutako kontrolagailuaren kasuan, osagai bat izanik eta erabat parametrizatuta egonik, beharrezkoa da konfigurazio funtzioa osatzea. 3B. eranskinean moldatutako VHDL iturriak eta hurrengoetan oinarrituta osatutako konfigurazio funtzioa azaltzen dira:

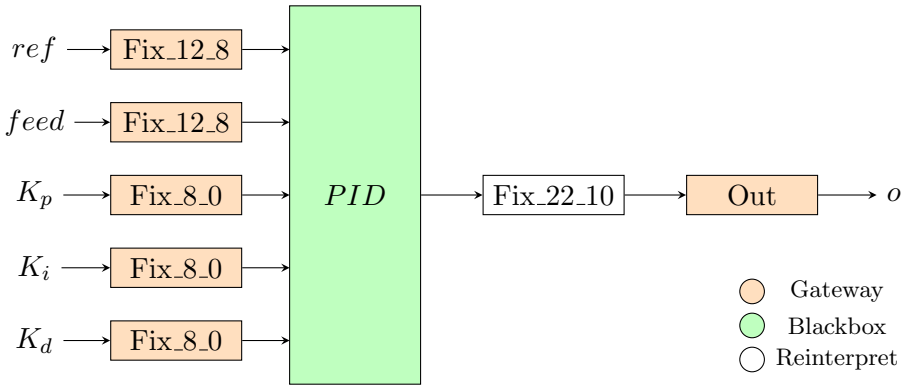
- Irteera den *mo* atakaren formatua adierazi behar da. $o\_wl = 22$ izanik, eta $minbp = 2$, Fix_22_2 litzateke sarrerak koma bitarretik eskumara zenbakirik ez dutela onartuz. Edozein kasutan, aurrerago adieraziko denez, komaren kokapena berriz interpretatuko da, eta ondorioz Fix_22 adieraztea izango da garrantzitsua, luzerak bat etor daitezen.
- *Generic* parametroen balioak aldatu dira, koefizienteak 8 biteko hitz-luzera dutela aintzat hartuta, 12 biteko sarrerak mantenduz, eta 1.11. taulan adierazitako koma bitarren kokapenak ikusita.
- Saturazio osagaia duen fitxategia kargatu da lehenengo eta kontrolagailuaren deskribapena duena ondoren.

	Kp	Ki	Kd
Hamartarra Zehatza	6	$6,875 \cdot 10^{-4}$	$22,7272$
7 bit	0000110.	[0.0000000000]1011011	10110.11
Zeinua 8 bit	00000110	01011011	01011011
Hamartarra 8 bitetik bihurtuta	6 $\Delta 0$ %0	$6,943 \cdot 10^{-4}$ $\Delta 6,8 \cdot 10^{-6}$ %0,9891	$22,75$ $\Delta 0,0228$ %0,1
Formatoa	Fix_8_0	Fix_8_17	Fix_8_2

1.11. Taula: *Balio zehatzetatik abiatuta, konstanteen seinaleei esleitu beharreko hitz bitarren kalkulua.*

Blokearen konfigurazio aukeretan *Simulation mode* aukeran *ISE Simulator* aukeratuko da. Honela egingo ez bazen, *Simulink*ek beste blokea guztien kalkuluak burutuko lituzke, baina *Blackbox*eko irteera guztien balioa zero izango litzateke.

Zuzenean simulazioa eginez gero, grafikoetan antzemango da irteerak balio oso altuak dituela, eta kontrola ezinezkoa dela. VHDL kontrolagailuaren irteeraren koma bitarra, aurreko atalean adierazi bezala, ez dago *lsb*an ezta *minbp* parametroak adierazitakoan. *Simulink*ek ez du sarreren koma bitarraren kokapena kontuan hartzen eta ondorioz interpretatzean 2^8 aldiz biderkatzen du irteeraren balioa.



1.21. Irudia: Blackbox *blokea erabiltzeko kontrolagailuaren modeloa ($K_p = 24$ eta $K_i = K_d = 91$).*

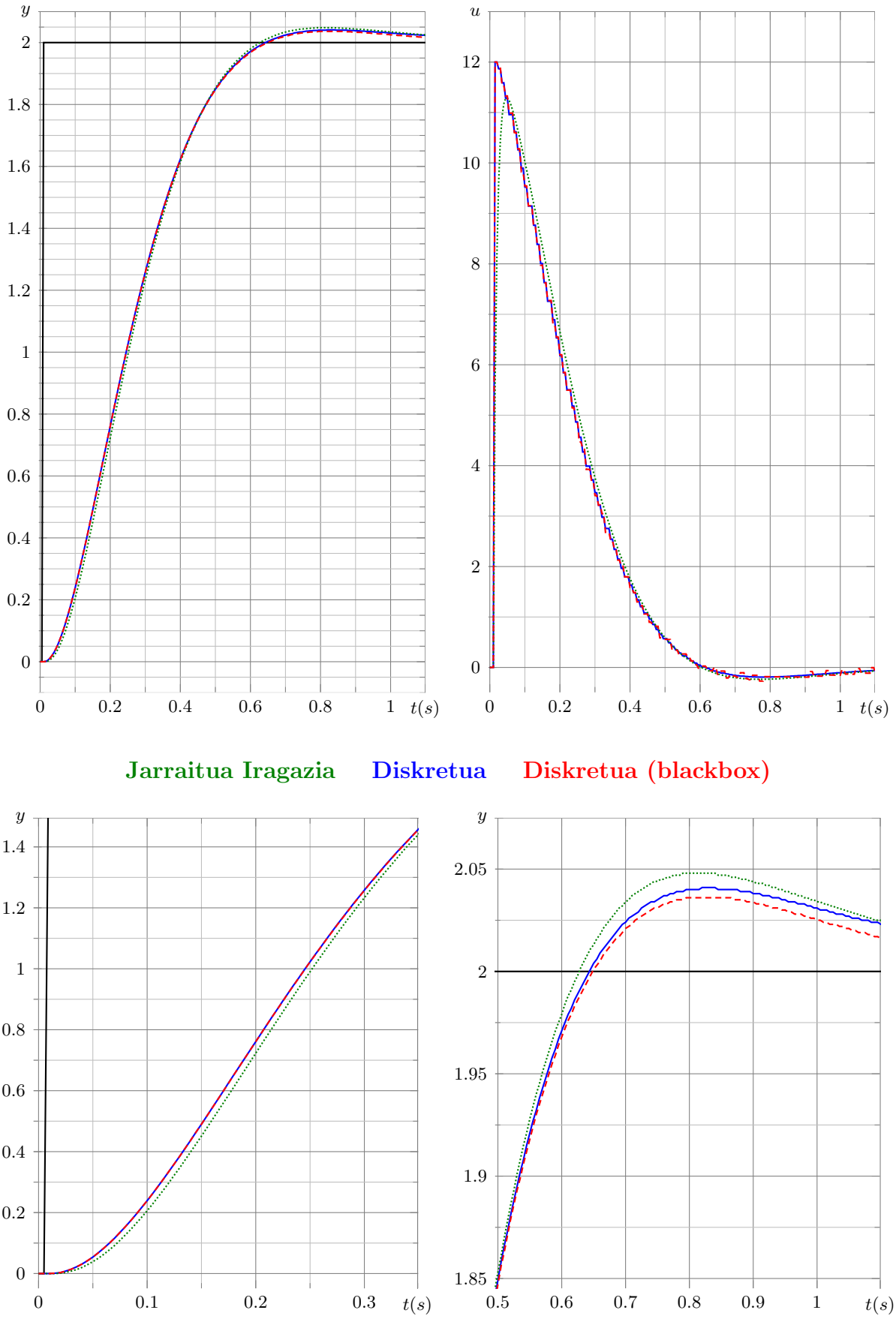
Arazo hori ekiditeko, *System Generator*ek ematen duenn *Reinterpret* blokea erabili da. Honek seinale mota eta formatua adieraztea ahalbidetzen du, *Simulink*ek biten pisua ezagutu dezan, beste blokeek balio zuzenak jaso ditzaten. *Output Arithmetic Type* aukeran *Signed (2's comp)* hartu da, eta koma bitarra non dagoen adierazi[44].

Seinaleak behar bezala interpretatzean programek VHDL kodea erabilita modeloaren funtzionamendua simulatzen dute. 1.22. irudiak adierazten duenez, honen emaitza, *System Generator* blokeekin egindako modeloak emandakoa eta modelo diskretuarenak oso antzekoak dira. Seinaleak begi hutsez aztertuta berdinak direla esan genezake, baina gaindiketa gertatzen den tartea begiratuz VHDL modeloak txikiagoa duela ikusten da, aurrekoen erantzuna zertxobait hobetuz.

Antzemandako diferentziak ulertzeko, parametroen adierazpenean zein kudeaketak egin diren aldaketak izan behar dira kontuan. 1.9.a eta 1.11. taulak konparatuz balio teorikoarekiko errore erlatiboa erdira txikitu da adar integratzailean, eta zortzi aldiz handitu deribazioaren kasuan, betiere guztiak azkenekoan $\%0,1$-etik behera mantenduz. Aldi berean, koma bitarra kokatzeko jarraitutako logikak eta parametroak adierazteko erabilitako hitz-luzeraren mugaketak, lerrokatzean mozketa bultzatzen du.

Nahiz eta hitz-luzera mugatu, parametroekin adierazpenak aldatu eta eskuz egindako VHDL deskribapena erabilita, azkeneko modelo honek modelo jarraituarekiko duen ezberdintasuna funtsean diskretu izateagatik dela iritzi daiteke, egindako pausu guztiek eragindako erroreak oso txikitzat hartuz ($\%0,5$-tik behera).

[44] Irteeren koma bitarra $(8) + minbp = 10$



1.22. Irudia: System Generator-*en* Blackbox *blokeaz egindako modeloaren eta aurrekoen arteko konparaketa.*

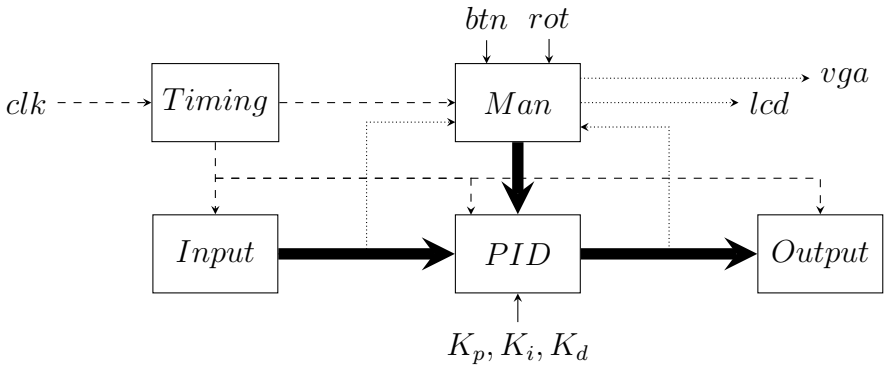
7.
Spartan3E Starter Kit

Plantaren arabera sarrerako/irteerako osagai ezberdinak inplementatzea beharrezkoa bada ere, azterketa kasu guztietako sistemek hainbat osagai komun dute: aukeratutako txartelaren erlojuan oinarritutako frekuentzia ezberdinak sortzea, esaterako. Hori dela eta, plantarekiko menpekotasunik ez dituztenak baina sistemaren autonomia bermatzeko ezinbestekoak direnak azalduko dira atal honetan; alde batetik baliabide minimoak eta bestetik aukeran dauden osagai guztiak.

Azterketa kasuak garatzean, planta zehatzek behar dituzten gehikuntzak (txartelaren sarreren/irteeren kudeaketari dagozkienak) landuko dira. Hurrengo atal eta azpiatalen azalpenean aurrekoekiko diferentzialak adieraziko dira, hau da, zer aldatu den. Sistema osotasunean ulertzeko, beraz, ezinbestekoa da aurrekoak aztertzea.

7.1. s3etiny: oinarrizko sistema

3C. eranskinean azaltzen diren VHDL iturrietan ikusi daitekeenez, sistema funtzionaltasunaren arabera bost oinarrizko osagaitan deskribatu da:



1.23. Irudia: *Sistemaren osagai nagusiak eta elkarren arteko funtsezko informazio fluxuak: erlojuak (marrak), ikuskatzekoak (puntuak) eta kontrolagailuaren sarrerak/irteerak (solidoa).*

- **Timing:** Txartelaren 50 MHz-eko erloju seinalean oinarrituta, FPGAk dituen DCMak eta kontagailuak erabilita, beste osagaiek behar dituzten frekuentzietako seinale periodikoak sortzea.
- **Man:** Erabiltzaileak sistema kudeatu (ref seinalea sortzea) eta aldagaiak ikuskatzeko interfazeen kudeaketa (sakagailuak, kodetzaile birakaria, LCD, VGA, etab.).
- **Input:** Plantaren kontrolatu nahi den aldagaia, y, irakurtzea eta kontrolagailuaren berrelikadura seinalea, $feed$, sortzea.
- **PID:** Aurreko ataletan deskribatu eta landutako kontrolagailua.

- **Output:** PIDaren irteeraren arabera, u, eragingailuaren agindu seinalea sortzea (DAC, PWM, etab.)

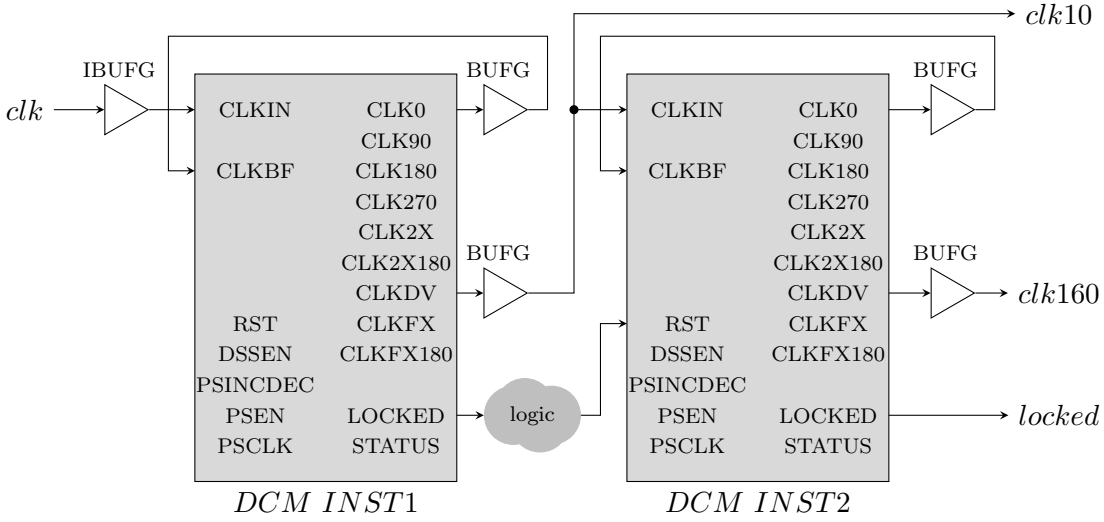
Input eta *Output* plantak bortizki baldintzatzen dituenez, eta PIDa sakonean aztertu denez, erloju seinaleen eta erabiltzailearen elkarrekintzaren nondik norakoak aztertuko dira jarraian.

7.1.1. Timing eta reset

Timing

Nahiz eta gain-laginketa handiarekin aritu, txartelak duen erlojua azkarregia da garatutako kontrol-sistemarako. 20 ns-ko periodoa izanik, 10^2 eta $10^{5,7}$ artean txikitu beharko da, plantaren arabera definitutako laginketa-maiztasuna erdiesteko.

Txartelak duen FPGAk erloju-seinalea kudeatzeko (skew-a kentzea, frekuentzia biderkatzea/zatitzea, fase aldaketa, etab.) *Digital Clock Manager* (DCM) izeneko lau gailu dituenez [36, p. 2], horietako bi erabili dira. Alde batetik, FPGAra heltzen den erloju seinale nagusia atontzeko (duty cycle-a zuzentzea, jatorrizkoa $\%40 - 60$ artekoa baita [37, p. 22]). Eta bestetik, berariazko baliabideak erabiltzeak, diseinua sinpletzeaz gain, fidagarritasun handiagoa eskaintzen duelako.



1.24. Irudia: Xilinx Clocking Wizard *bidez seriean konfiguratutako bi DCM.*

Architecture Wizard-eko *Clocking Wizard* laguntzailea erabiliz, *Cascading In Series With Two DCMs* konfigurazioa aukeratu da. Honek seriean konektatzen ditu bi DCM, lehenengoaren (INST1) irteera bigarrenaren sarrerara konektatzen du. 1.24. irudian antzeman daitekeenez, DCM bakoitzak irteera berrelikatzen du ezarritako baldintzak bermatzeko. Aldi berean, bitarteko eta irteera bakoitza *BUFG* motako erloju jarraitzaile orokor banak bideratzen du.

Spartan-3E familiako DCMen sarrerako seinalearen ($CLKIN$) frekuentzia minimoa 5 MHz-ekoa izanik, seriean konektatutako lehenengo gailuak gehienez hamar aldiz ($\frac{50\,\text{MHz}}{5\,\text{MHz}} = 10$) zatitu

ahal du txarteleko erloju seinalea. Bigarrenean, aldiz, adierazitako txikitze tarteak gogoratuz, maximoa ezarri da, hamasei. *PDCM* osagaiak, hortaz, 50 MHz, 5 MHz (*clk*10) eta 312,5 kHz (*clk*160) maiztasunetako seinaleak ematen ditu.

Maiztasun are txikiagoak erdiesteko kontagailuak erabili dira. Sistemaren atal ezberdinetan erruz erabili den elementua izanik, *wl* izeneko *generic*aren arabera parametrizatutako osagai gisa deskribatu da. Gaitze seinalea (*en*) eta reset sinkronoa (*srst*) dituen D motako flip-flop batean oinarritutako egitura du, irteera (*Q*) sarrerara (*D*) unitate batutzaile batez berrelikatuta duena, eta sarrera gisa definitutako mugarekin konparagailuak aktibatzen duen reset seinalea. *tc* irteera erregistroa irteerak zero balioa duen bitartean aurkezten du maila altua. Honen maiztasuna, beraz, osagaiaren erloju seinalearena baino $muga + 1$ aldiz txikiagoa da. *count* irteerak *D* seinalearen balioa ateratzen du, nahiz eta orain ez den erabiltzen.

```
constant dv25l: std_logic_vector(4 downto 0):=std_logic_vector(to_unsigned(24,5)
    );
constant dv125l:std_logic_vector(6 downto 0):=std_logic_vector(to_unsigned
    (124,7));
```

Instantzia	wl	clk	en	limit	tc	count
DV25	5	clk160	H	dv25l	clk12_5k	-
DV125	7	clk160	clk12_5k	dv125l	clk0_1k	-

1.12. Taula: *Oinarrizko kontagailuen instantzien sarrerak/irteerak.*

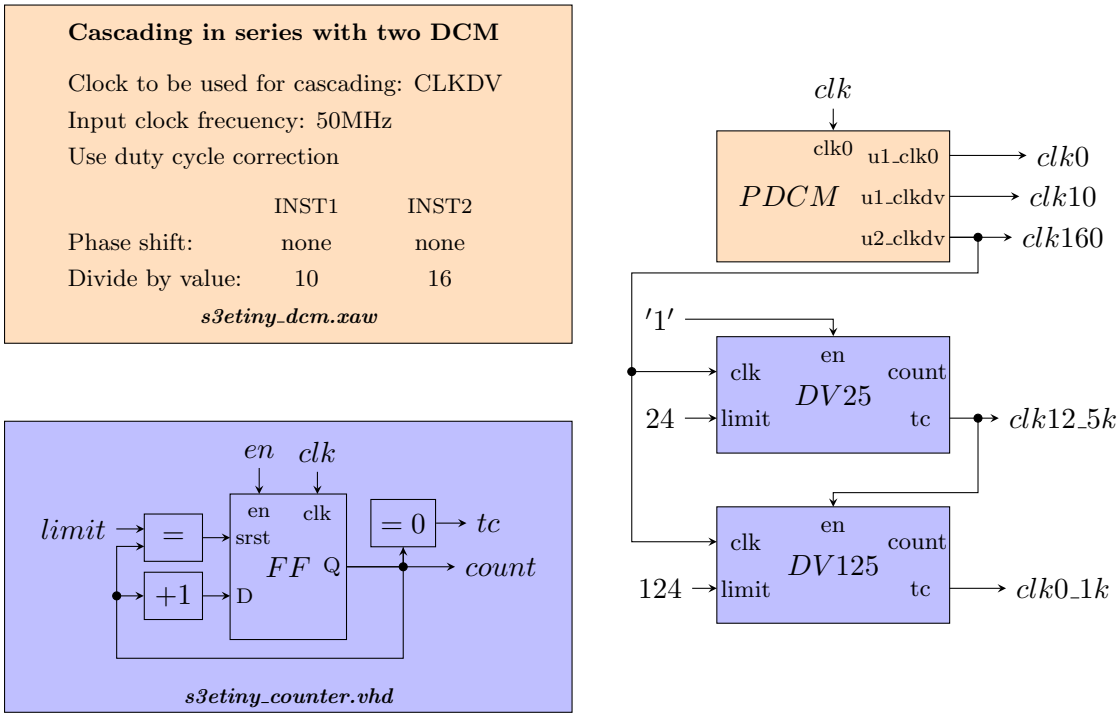
Mugak konstante gisa definitu dira instantzia deitu den arkitekturan. 1.12. taulak balio horiek erabilita, eta egindako konexioekin batera, erdietsitako maiztasunak adierazten ditu.

Aztertutako azkeneko irudi eta taula hauek bateratuz, 1.25. irudian adierazten den egitura orokorra osatu da. Erloju sarrera nagusitik abiatuta, eta FPGAk eskaintzen duen paraleloan deskribatzeko aukera baliatuz, 10^0 eta $10^{4,3}$ artean zatitutako maiztasuneko bost seinale daude eskuragarri. Hauekin, *man* ataleko baliabideak kudeatzeko nahikoa da, eta plantak baldintzatutako ataletarako moldatzeko *counter* osagaiaren instantzia berri bat seriean zein paraleloan ezarri baino ez da egin behar.

DCMek emandako seinaleak bezala, kontagailu bitartez sortutakoak ere BUFG motako erloju jarraitzaile orokor banak bideratzeko *clock_signal* eta *buffer_type* murrizketak erabili dira. Honela denboraren ikuspuntutik kritikoak direnen hedapena sistema osoan zehar ahal bezain homogeneoa izatea bermatzen da.

```
attribute clock_signal: string;
attribute clock_signal of i_clk12_5k: signal is "yes";
attribute clock_signal of i_clk0_1k: signal is "yes";

attribute buffer_type: string;
attribute buffer_type of i_clk12_5k: signal is "bufg";
```



1.25. Irudia: Timing *atalaren funtsezko egitura eta oinarrizko osagaiak.*

```
attribute buffer_type of i_clk0_1k: signal is "bufg";
```

Reset

Nahiz eta, irudiak eta azalpenak erraztearren, sistema deskribatzean ataletan berrezartze seinale orokorrik ez aipatu, egitura sekuentzialak dituzten osagai guztiek (*counter, udcounter, bintobcd, lcd, etab.*) berrezartze sinkronoa dute (*srst*); eranskinetan aurkeztutako kodeak adierazten duenez. Bestetik, seriean ezarritako bi DCMek asinkronoa (*rst*) dute.

Erabiltzaileak berrezartzeko sakagailua eragitean, FPGAren sarrera den *rst* seinaleak maila altua hartuko du, zuzenean DCMak berrabiaraziz. Hauek berriz ere behar bezala sinkronizatuta egon arte, *u2locked* seinalea maila baxuan egongo da. Sistema osoaren funtzionamendu zuzena DCMen menpekoa izanik, *mrst* seinalea maila altuan egongo da berrabiarazten dauden bitartean. Horrek erloju seinale ezberdinak berrabiatzen mantenduko ditu. *clk0_1k* erlojuak aginduta z^{-1} atzerapenaz sortuko da *lrst*. Azkeneko honek *TIMING* atalez kanpoko guztiak berrabiatzen mantenduko ditu DCMak eta erloju seinaleak egonkortu bitartean eta periodo bat iragan arte.

```
mrst <= rst or (not u2locked);
lrst <= rst or (lock nand u2locked);

process(iclk0_1k)
begin
```

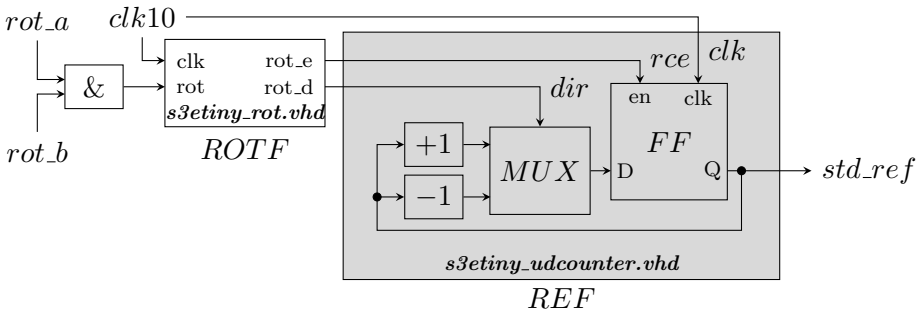
```
if rising_edge(iclk0_1k) then lock <= u2locked; end if;
end process;
```

7.1.2. Man

Atal honek elkarren arteko konexiorik ez duten azpiatal ezberdinak bateratzen ditu:

Erreferentzia-seinalea

Alde batetik kontrolagailuaren erreferentzia-sarrera den seinalea sortu edo jasotzeko osagaiak agertzen dira. Nahiz eta inplementatutako komunikazio osagai baten bitartez (UART, Ethernet, USB...) beste sistema batek bidalitako aginduek aldatu lezaketen, garatutako sistema autonomoa izanik, txartelak duen sakagailu birakaria, kodetzaile birakaria, erabili da.



1.26. Irudia: *Txarteleko sakagailu birakariaren bitartez erreferentzia-seinalea ezartzea.*

Zaratak eta errebote mekanikoek eragindako eraldaketak deuseztatzeko gailuaren bi seinaleak iragazi eta alde batetik pultsuak eta bestetik noranzkoa adierazten dituzten seinaleak sortu dira [38]. Iragazia darabilen erloju seinale berak agintzen duen erregistro batean oinarritutako gora eta beherako kontagailu baten gaitze seinaleari eta noranzkoa erabakitzen duen multiplexoreari aurrekoak esleitu zaizkie, hurrenez hurren (1.26. irudia).

Honela, $[0,2^n-1]$ tarteko balioak adierazi ditzakeen erreferentzia-seinalea da, zeinurik gabekoa. Zeinua ezartzeko *switch* baten bitartez eragindako seinalearen arabera biko osagarrira bihurtu da.

```
ref_man <= signed('0'&std_ref) when ref_dir='0' else signed(not ('0'&std_ref))
    +1;
```

Aldagaiak hartu ditzakeen balioak sortutako erreferentziarenekin bat badatoz baina maximoa $2^n-1 > var > 2^{n-1}-1$ artean dagoelarik, erreferentzia mugatzeko *rot_e* eta *en* ataken artean kontagailuak duen balioaren arabera aldaketa gaitzen dituen multiplexadorea deskribatu da.

```
rce <= '0' when ((unsigned(std_ref)>max and rot_d='0') or
                 (unsigned(std_ref)=0 and rot_d='1')) else rot_e;
```

LCD

Bestetik, sistemaren aldagaiak ikuskatzeko txartelak duen 16x2 LCDa kudeatzeko sistema garatu da. Alde batetik datuen aurkezpena kontrol begiztarekiko guztiz banatzeko, eta bestetik LCDaren kudeaketan aldagaien moldaketa/prestaketak eta gailuan idazketa ezberdintzeko, kudeatzailea osagai bereizietan deskribatu da.

Sarrerak/irteerak LCDaren komunikazio protokoloarekin [37, ch. 5] bat etor daitezen *LCD* osagaiak hiru egoera finituko makina (ingelesez *Finite State Machine*) eta hauen elkarlanerako beharreko logika bateratzen ditu. Honetan ere RAM memoria bat deskribatu da, *buffer* gisa jokatuko duena. LCDaren ageriko memoriak adina helbide ditu RAMak, nahiz eta 2D matrizea den, eta ez 3D-koa. *AUTO* abiarazte osagaiak aginduta, makina orokorrak gailua konfiguratzen du lehenik (*SETUP*) eta ondoren etengabe idazten (*WRITE*) ditu RAMeko hitzak (erdia lerro batean eta erdia bestean).

LCDMAN osagaiak, berriz, kontrolagailuak darabiltzan aldagaiak eta irteera zuzenean jasotzen ditu, eta beharreko moldaketak burutu ostean RAMan idazten ditu. Moldatzea BCDra bihurtzean eta ondoren LCDaren karaktere taularen [37, fig. 5-4] arabera osatzean datza.

Sarrera diren 14 biteko zeinudun (biko osagarrian) seinaleen pisu handieneko bitaren arabera hauen balio absolutua adierazten duten seinaleak sortu dira. Ondoren, BCD zifra bakoitzeko lau biteko moldatutako *shift register* (*bcdshift*) bana erabilita aurrekoa jaso eta 16 biteko BCD seinalea ematen duen osagaia deskribatu da (*bintobcd*) [39].

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	**E**	**F**	**:**	**-**	**0**	**0**	**0**	**0**		**P**	**I**	**D**	**O**	**U**	**T**
V	**A**	**R**	**:**	**-**	**0**	**0**	**0**	**0**			**-**	**0**	**0**	**0**	**0**
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

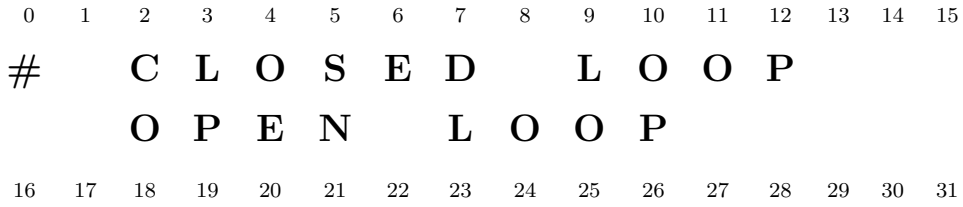
1.27. Irudia: *LCDMAN, 16x2 LCDaren aurkezpen nagusiaren antolaketa: ref (*r*), var (*y/feed*) eta out (*u*).*

Karaktere taulan adierazitakoen arabera, bihurketaren emaitza diren BCD zenbakiak erabilita, sarrerako seinaleen zeinua aintzat hartuta eta datuak izendatzeko testua gehituz, 1.27. irudiak adierazten duen antolaketa deskribatu da.

Bestetik, sistemaren autonomia bermatzeko, txertatutako sistema osoa izanik, erabiltzaileak (eta ez diseinatzaileak parametro eta *generic* seinaleen bitartez) zenbait aukera konfiguratzeko izan dezan, LCDa eta zenbait sakagailu baliatuz menua deskribatu da: *LCDMENU*. Honek *REF* eta *LCDMAN* osagaietan du eragina, hau da, menua aktibatzean, *inmenu* seinaleak balio logiko altua hartzean, erreferentzia-seinalea sortzen duen kontagailua ezgaitzen da *-en* ataka zerora finkatuz- eta *LCDMAN*en berrabiarazte seinalea aktibo mantentzen da. Aldi berean *LCD* osagaiak (*LCDRAM*ek ondorioz) ez ditu *LCDMAN*en *addr*, *data* eta *load* seinaleak jasotzen, baizik eta *LCDMENU*renak.

Osagai hau FSM batek osatzen du, sarrerak/irteerak aldatzeko dagokion logikarekin. Bi

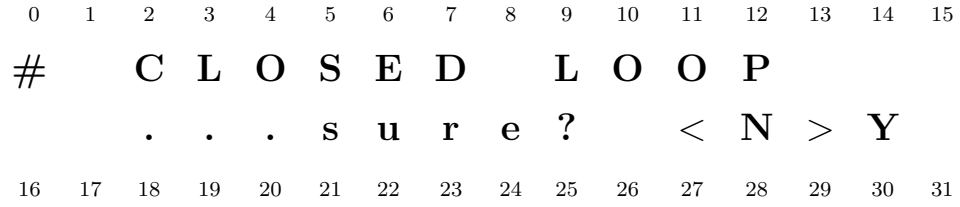
egoera finko ditu (*standby* eta *changed*) eta menuak duen leiho bakoitzeko beste hiru (batek elementu finkoak irudikatzen ditu, bigarrenak aktibo den edo aukeratutako balioaren araberakoak irudikatzen ditu, eta hirugarrenak erabiltzailearen erantzunak aztertzen ditu).



1.28. Irudia: *LCDMENU,* mode *seinalearen balioa aukeratzeko leihoa.*

1.29. Irudia: *LCDMENU,* encsel *seinalearen balioa aukeratzeko leihoa.*

Oinarrizko sistemak bi leiho ditu, hiru sakagailuz (*btnsel*, *btntg* eta *btnmode*) eta erreferentziarako erabilitako birakariarekin. *btnmode* sakatuz gero LCDak 1.28. irudiak aurkeztutakoa adierazten du, eta *btntg* sakatzean 1.29. irudikoa. Leihoetara sartu ostean, horietako edozein sakatzean menutik ateratzen da sistema, eta ez du aldaketarik gordetzen.

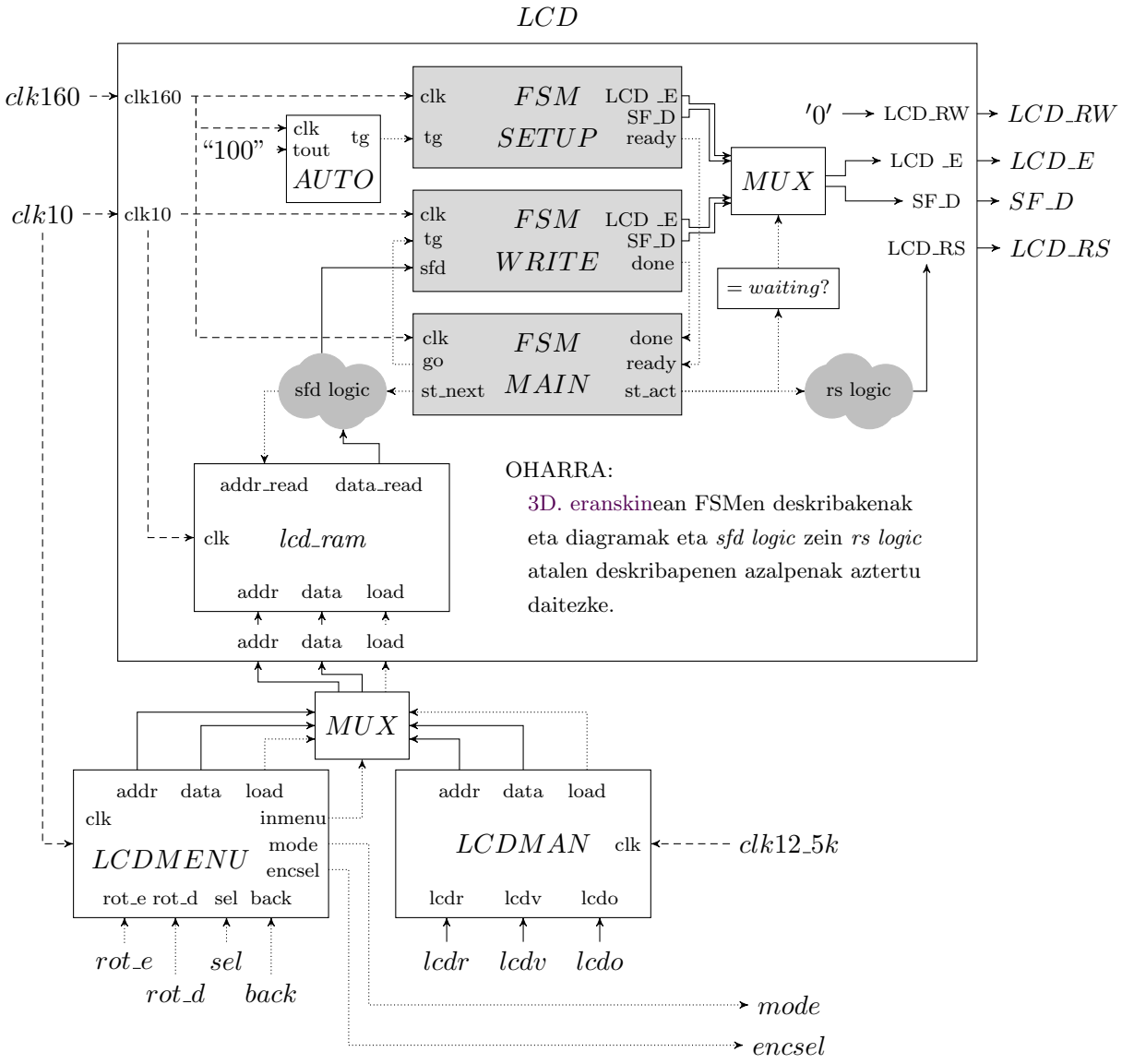


1.30. Irudia: *LCDMENU,* mode *seinalearen aukeratutako balioaren berrespena.*

1.31. Irudia: *LCDMENU,* encsel *seinalearen aukeratutako balioaren berrespena.*

Leihoetan egonda, sakagailu birakariak dauden aukeren artean aldatzen du. Lehenengoan #

sinboloak adierazten du aukeratutakoa. Bigarrenean, aldiz, < eta > artean agertzen da. *btnsel* sakatzean, aldaketak gorde baino lehen aurkezpena aldatzen da eta erabiltzaileari berrespena eskatzen dio (1.30. eta 1.31. irudiak). Aukera berresterakoan, aldagaiak aukeratutako leihoari dagokion barne seinalearen, eta ondorioz irteera atakaren, balioa aldatzen du eta menutik irteten da.



1.32. Irudia: *LCD-kudeatzailearen egitura orokorra: erlojuak (marrak), agindu seinaleak (puntuak) eta informazioa (solidoa).*

Debounce

Menuan ibiltzeko aipatutako hiru sakagailuek, egitura mekanikoa izanik, erreboteak dituzte. Hau da, sakatzean seinaleak ez du zuzenean maila aldatzen, baizik eta aurrekoa eta ondorengoaren artean hainbat aldaketa egiten ditu egonkortu arte.

Nahi ez diren aldaketa anitz horiek garbitu eta sistemara bakarra bidaltzeko iragazi gisa *shift-register* batean oinarritutako *s3etiny_debounce* osagaia deskribatu da. $clk0\_1k$ erloju seinaleak aginduta eta 4 biteko luzera izanda $10ms$-ko lau laginketa jarraituk balio altua izan behar dute, hau da $40ms$-tan gertatu daitezkeen erreboteak ez ditu irakurtzen.

Iragazi guztiak bateratzeko *s3etiny_btndeb* osagaian sakagailu bakoitzeko *s3etiny_debounce* instantzia bat deskribatu da.

7.1.3. Normalizatzea

Aipatutako bost atal nagusiez gain, hauen arteko datu fluxua kudeatzeko zeharkakoa den parametrizatutako normalizazio osagaia (*s3etiny_norm*) deskribatu da. Alde batetik erreferentzia ($nref$), aldagaia ($nvar$) eta irteera ($nout$) seinaleen hitz-luzera eta balio tarteak moldatzen ditu honek, atal ezberdinen baldintzak betetzeko. Bestetik, erregistroak ezartzen ditu eta laginketa-maiztasuneko erlojuan oinarrituta sekuentziatuta berritzen ditu.

Sekuentziari dagokionez, $clk10$ erlojuarekin $clk\_pid = clk\_adq \cdot z^{-1}$ definitu da, laginketa-erlojuak goranzko aldaketa duenean $INPUT$ atalean laginketa burutzeko $\frac{1}{clk10}$ms emanez, kontrolagailuak lagindu baino lehen. MAN ataletako osagaiek ez dutenez aldi berean lagintzen, normalizazio osagaiaren barnean kontrolagailuak egiten duen aldi berean erregistratzen dira sarrerak.

Moduluen berdinketarako irteera ataka bakoitzeko konstante bana definitu da, konstante guztiek luzera berdina izanik (*nwl generic* parametroak definituta):

Osagaiaren ataka guztien luzera beste osagaietan erabilitako *generic* berdinek baldintzatzen dituztenez, horiek baliatuz eta *resize* eta *shift_right* funtzioen bitartez konstantearekin biderketan *overflow*rik gertatzen ez dela ziurtatuz, irteeraren luzerara moldatu eta konstantearen koma bitarraren kokapena zuzentzen du.

Honela, kontrolagailuaren funtzionamendu zuzenerako, erreferentzia eta aldagai seinaleen eskalak berdintzen dira eta luzera berdineko seinaleak bidaltzen zaizkio, nahiz eta bereizmena ezberdina izan. Aldi berean, identifikazioa egitean erreferentzia ezberdinak sortzeko edo beste edozein egoeratan begizta irekian aritzeko, ref seinaleak zuzenean irteera agintzeko aukera egon dadin $open\_ref$ seinalea sortzen du osagai honek. LCDan aurkezteko seinaleei dagokionez, lehenago azaldu denez, moduluak 13 bit izan behar ditu gehienez.

7.1.4. UCF

```
NET "clk" LOC = "C9"  |IOSTANDARD=LVCMOS33;
NET "clk" TNM_NET=clk;
TIMESPEC TS_clk=PERIOD "clk" 20 ns HIGH 40%;

NET "rot(0)" LOC="K18" |IOSTANDARD=LVTTL |PULLUP;
NET "rot(1)" LOC="G18" |IOSTANDARD=LVTTL |PULLUP;
```

```
NET "LCD_E"  LOC="M18" |IOSTANDARD=LVCMOS33 |DRIVE=4 |SLEW=SLOW;
NET "LCD_RS" LOC="L18" |IOSTANDARD=LVCMOS33 |DRIVE=4 |SLEW=SLOW;
NET "LCD_RW" LOC="L17" |IOSTANDARD=LVCMOS33 |DRIVE=4 |SLEW=SLOW;

NET "SF_D(0)" LOC="R15" |IOSTANDARD=LVCMOS33 |DRIVE=4 |SLEW=SLOW;
NET "SF_D(1)" LOC="R16" |IOSTANDARD=LVCMOS33 |DRIVE=4 |SLEW=SLOW;
NET "SF_D(2)" LOC="P17" |IOSTANDARD=LVCMOS33 |DRIVE=4 |SLEW=SLOW;
NET "SF_D(3)" LOC="M15" |IOSTANDARD=LVCMOS33 |DRIVE=4 |SLEW=SLOW;

NET "btn(0)" LOC="H13"  |IOSTANDARD=LVTTL |PULLDOWN;
NET "btn(1)" LOC="D18" |IOSTANDARD=LVTTL |PULLDOWN;
NET "btn(2)" LOC="V16" |IOSTANDARD=LVTTL |PULLDOWN;

NET "dir" LOC="L13" |IOSTANDARD=LVTTL |PULLUP;
NET "en"  LOC="L14" |IOSTANDARD=LVTTL |PULLUP;
NET "rst" LOC="N17" |IOSTANDARD=LVTTL |PULLUP;
```

7.2. s3e: osagai guztiak

Sistema baliabide murritzeko gailuetan inplementatu ahal izateko, aurreko atalean erabilerarako ezinbestekoak direnak baino ez dira deskribatu eta azaldu. Jarraian sistema osatzen dutenak adieraziko dira, baina beharrezkoak ez direnak. Azalpenetan, beraz, lehenago aipatu bezala, gehitutakoak aztertuko dira soilik, irakurleak oinarrizko sistema ezaguna izanik. Hala ere, 3E. eranskinean azaltzen diren VHDL iturriak ez dira diferentzialak eta inplementatu nahi izatekotan izen bereko osagaiak berrizendatu eta gainidaztearekin nahikoa da.

7.2.1. Timing eta reset

Timing

MAN atalean azalduko denez, sistema osoak VGA seinalea sortuko du txartelera zuzenean monitore bat konektatu ahal izateko. Erabiltzaile gidak [37, p. 56] adierazten dituen sinkronismo seinaleak sortzeko baldintzak 25 MHz-eko erlojua dute oinarri.

Hori dela eta, DCMek eskaintzen duten eta irekita zegoen zuzendutako erloju iturria *TIMING*-en irteera ataka gisa gehitu da. Frekuentzia erdiko, *clk*2, gaitze seinalea sortzeko *s3etiny_counter* egituraren kasu sinpleena den *s3etiny_switch* erabili da, maila altu finkoak gaitzen duen bit bakarreko alderantziz berrelikatutako erregistroa.

```
DV2: anie_switch port map ( iclk0,'1',mrst,iclk2 );
```

Aldi berean, erabiltzaileak gertatzen diren aldaketak bereizi ahal izateko beste erloju guztiekin aldaratuz zeharo geldoa den 1 Hz frekuentziako *clk*1*s* deskribatu da, kontagailu baten bitartez. Honen sarrera-seinalea *clk*12_5*k* da, *clk*0_1*k*-k gaitzen du eta ehun unitateko modulua du, honetarako konstante berria definitu delarik:

```
constant dv100l:std_logic_vector(6 downto 0):=std_logic_vector(to_unsigned(99,7)
    );
```

```
DV100: anie_counter generic map ( 7 )
port map ( iclk12_5k,iclk0_1k,ilrst,dv100l,open,iclk1s );
```

*clk*2 seinaleak *clk*0 erlojua gaituko duenez, edo berebiziko garrantzirik ez duten osagaiak aginduko dituenez, ez zaio jarraitzaile zehatzik esleitu. *clk*1*s* seinaleari, aldiz, *clk*12_5*k* eta *clk*0_1*k*rekin egin bezala, BUFG motako erloju jarraitzaile orokorra erabiltzeko murrizketak gehitu zaizkio:

```
attribute clock_signal of iclk1s: signal is "yes";
attribute buffer_type of iclk1s: signal is "bufg";
```

Reset

Berrabiatzeari dagokionez, *clk*1*s* seinalea sortzen duen osagaiaren sarrerak *mrst*ren menpekoak izanik, honek ezin du berdina izan. Hau dela eta, *lrst*ren barneko ispilua deklaratu da, kontagailu hau logika orokorrarekin batera hasi dadin.

```
signal ilrst: std_logic;
```

```
ilrst <= rst or (lock nand u2locked);
lrst <= ilrst;
```

7.2.2. Man

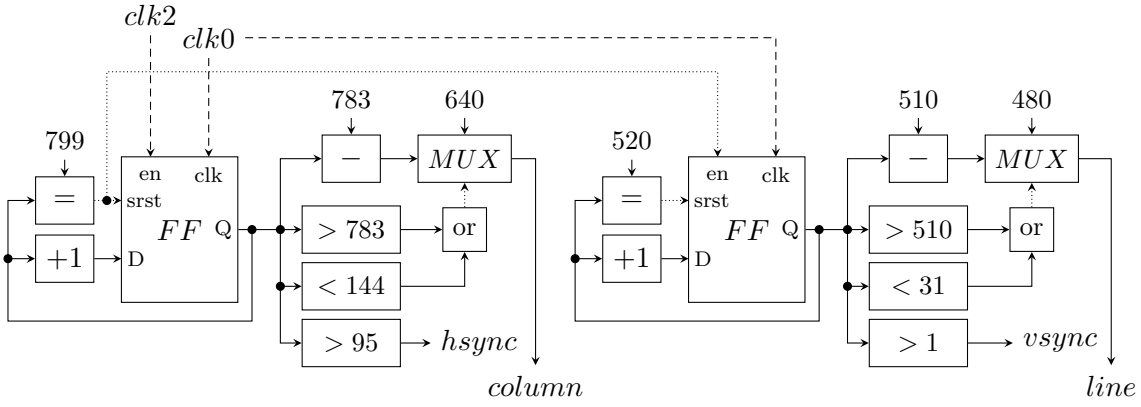
VGA

Erreferentziaren sorrerarekin gertatu bezala, edozein komunikazio osagairen bitartez txartelak seinaleen momentuko balioak beste sistema batera bidali genitzake, erabiltzaileak momentuko seinaleen balioak ez ezik aurrezarritako azkeneko iterazio kopuruan izandakoak ere ikuskatu ahal izateko. Nolanahi ere, garatutako proiektuaren autonomia bermatzeko, txartelak duen DB15 konektorearen bitartez CRT zein LCD pantaila erabiltzeko 640x480 pixeleko bereizmena eta 60 Hz-eko berritze-maiztasuna dituen VGA seinalea sortu da, *Video Electronics Standards Association* (VESA) erakundeak adierazitako zehaztapenak jarraituz [37, ch. 6].

Aukera honek grafikoen xehetasunak murrizten ditu. Alde batetik prozesatzea airean egiten delako, eta erabiltzen diren erloju maiztasunek aurkezpen konplexuak garatzeko logikaren berebiziko azkartasun eta optimizazioa eskatzen dutelako. Eta bestetik FPGAren baliabideak aurreztearren, geroz eta aurkezpen konplexuagoa egin kontrolerako logika deskribatzeko baliagarri izan litezkeen baliabide gehiago hartzen baitira. Eraginkortasuna bermatzeko, beraz, xehetasun edo bereizmen handiko grafikoak kontrol-sistematik at dagoen beste batek sortuko ditu.

LCDaren deskribapena baldintzatu duen logikari jarraiki, bi osagai nagusitan banaturik deskribatu dira aldagaien moldaketa/prestaketa eta VGA seinalearen zehaztapenak beteko dituen kontrol seinaleen sorrera.

1.33. irudian aztertu daitekeenez, *VGASYNC* osagaian sinkronismo seinaleen sorrerarako bateratutako bi kontagailu eta lau konparadore baino ez dira erabili: bata horizontalerako (*hsync*) eta bestea bertikalerako (*vsync*).



1.33. Irudia: *VGASYNC, 25 MHz-eko gaitze seinalearekin 640x480 60 Hz VGA seinalea kudeatzeko sinkronismo seinaleak (hsync eta vsync) eta dagozkien koordenatuak (column eta line) sortzea: erlojuak (marrak), agindu seinaleak (puntuak) eta informazioa (solidoa).*

Aurrekoez gain momentuan kolorea definituko duten osagaiek kurtsorearen kokapen zehatza zein den jakin dezaten, ageriko azalerari dagozkien koordenatuak sortzeko bina konparadore, kentzailea eta OR atea eta multiplexadorea erabili dira. Kentzaileak soilik kontagailuaren eta koordenatuaren balioen arteko offseta ezabatzen du, ezkutuko tarteek eragindako diferentzia deuseztatuz. Irteerak 640 zutabe ikusgai dituenez, nahiz eta sinkronismoa bermatzeko kontagailuak 799ra arteko balioak hartu, ez dira ezkutuko tarteko balioak adieraziko eta multiplexadoreak 640 balioa finkatuko du OR ateak aginduta. Zutabe kopurua baino aukera bat gehiago adieraziko du *column*ek, beraz, $(0, 640)$. Era berean, *line*k $(0, 480)$ tartea adieraziko du.

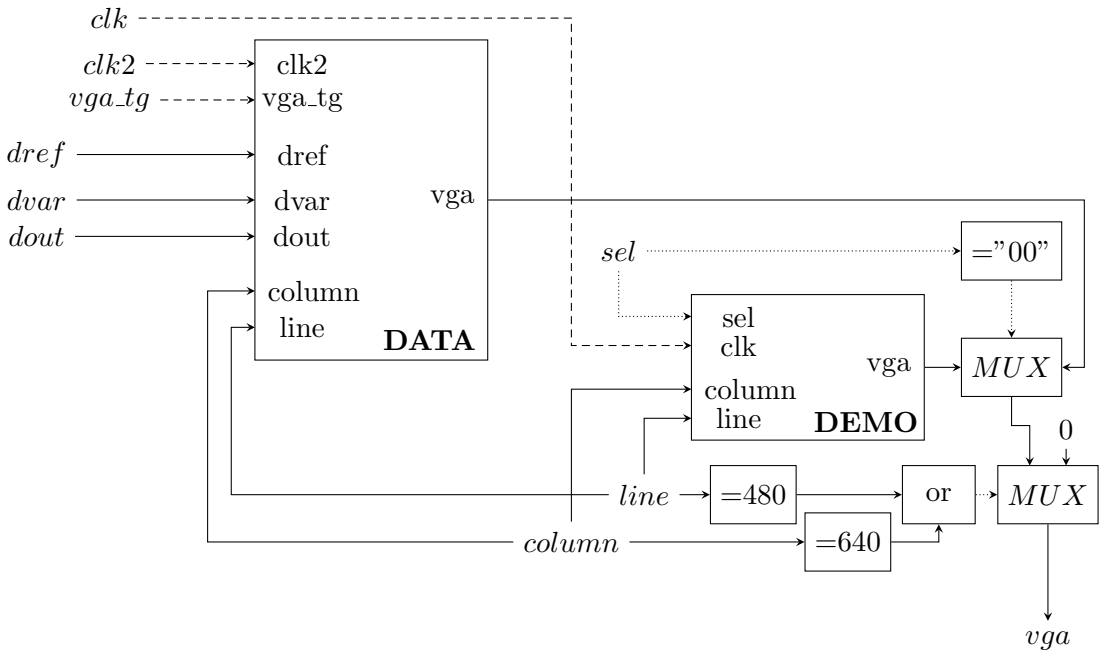
VGAMAN osagaia, sortu berri diren *column* eta *line* seinaleak baliatuz eta beharreko erloju zein aldagai seinaleek lagunduta, bi biteko seinalea den *sel*ek agindutako lau aurkezpen ezberdinen RGB seinaleak sortzeko atalek eta logikak osatzen dute (1.34. irudia).

Aurkezpen moduen artean aldatzeko, sakagailuen iragaziak darabilen erloju seinale berdina duen kontagailua erabili da, sakagailu batek gaitzen duena. Maximo finkoa "11" denez, sekuentzialki aldatzen da lau moduen artean. Osagai honen irteera, *vm*, da *VGAMAN*en *sel* sarrera.

```
VMODE: anie_counter generic map ( 2 )
port map ( clk0_1k,btndeb(0),srst,"11",vm,open);
```

Logikari dagokionez, 1.13. taulak adierazi bezala aurkezpena zein izango den aukeratzeaz gain, zutabea zein lerroa ageriko azaleratik kanpo daudenean (640 edo 480 adierazten dutenean, hurrenez hurren) irteera zerora finkatzen da. Tarte horietan izpia hasierara bueltan ari denez RGB seinaleak tentsiorik sortu ez dezan [40].

DEMO osagaiak konparadore anitzen bitartez ageriko azalera sei zutabe eta sei errenkadatan



1.34. Irudia: *VGAMAN, momentuan kolorea definitzeko logika eta osagaiak: erlojuak (marrak), agindu seinaleak (puntuak) eta informazioa (solidoa).*

sel	**modua**	
00	data	PIDaren erreferentzia, aldagai eta irteeren grafikoak
01	columns	Sei kolore ezberdineko zutabe zabalak
10	lines	Sei kolore ezberdineko errenkada zabalak
11	grid	*clk*1sek aginduta aldatzen diren zortzi kolore ezberdineko tableroa

1.13. Taula: sel *seinalearen araberako VGA osagaiaren RGB aurkezpen moduak.*

banatzen du. Aukeratutako moduaren arabera koloreak ezartzeko erdia erabiltzen ditu, beste erdia edo biak nahastuta. Koloreen ordena, ezkerretik eskumara zutabeen kasuan eta goitik behera errenkadetan, 1.14. taulak agertzen du:

Kolorea	yellow	cyan	green	magenta	red	blue
Kodea	110	011	010	101	100	001

1.14. Taula: *DEMO osagaiko* lines *eta* columns *moduen kolore ordena.*

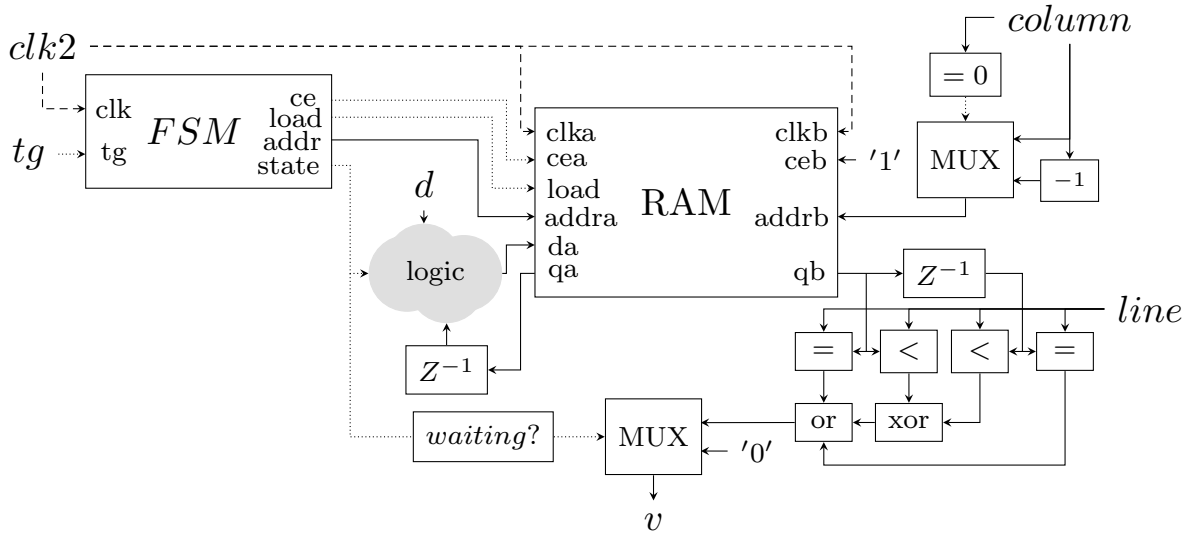
DATA osagaiari dagokionez, hiru geruza ezberdinek osatzen dute hau, behetik gora: koadrikula, grafikoak eta zeinuak.

Lehenak, *s3e_vgasimpgrid*ek, zutabe eta lerro seinaleak jasotzen ditu eta hauen arabera bit bakarreko irteera ematen du. Horrek bi koadrikula irudikatzen ditu, 235 pixeleko altuera dutenak, bata pantailaren behealdean lerrokatuta eta bestea goian. Biek dute zabalera osoa, 639 pixel. Koadrikula bakoitzeko ertzak irudikatzen dira; ordenatuan %25, %50 eta %75 adierazten duten lerroak; eta abzisan 100 pixeleko tarteak adierazten dituzten zutabeak (ertzetakoak salbu, zertxobait zabalagoak baitira).

Bigarren geruzan, irudikatu nahi den aldagai bakoitzeko (erreferentzia -*dref*-, *INPUT* atalean irakurritakoa -*dvar*- eta *OUPUT* atalera bidalitakoa -*dout*-) *s3e_vgashift* osagaiaren instantzia bat deklaratu da. Parametrizatutako osagai hau ataka multzo bikoitzeko RAM batean oinarritutako *shift-register*a da (1.35. irudia).

Multzo batek datuak idatzi zein irakurtzeko atakak ditu, eta FSM batek aginduta *tg* seinaleak maila altua hartzen duen bakoitzean eskumatik hasita eta era sekuentzialean datuak ezkerrera posizio bat mugitzen ditu, lehenengoan sarrerako datu berria idatziz. Egitura dela eta, berritze prozesuak behintzat $zutabe\_kopurua \cdot 2$ erloju ziklo behar ditu.

Beste multzoak, irakurketa atakak baino ez dituenak, zutabe seinalearen arabera dagokion helbidearen datua irakurtzen du eta aurreko helbidean irakurritakoarekin batera lerroak adierazitako balioarekin konparatzen du. Aktibo dagoen lerroa bi datuen artean badago edo bietako edozeinen berdina bada, puntua irudikatzen du. Zutabe seinalearen balioa 25 MHz-eko maiztasunarekin aldatzen denez, *VGASYNC* osagaian ikusi denez, multzo honek laginketa-maiztasun berdinarekin egin behar du lan, eta berez *hsync* seinalearen izaera sekuentzialaz baliatzen da irakurketa logika erregistro batez sinpletzeko.



1.35. Irudia: *VGASHIFT, ataka multzo bikoitzeko RAM batean oinarritutako* shift-register*a,* column *eta* line *seinaleen arabera edukia aztertzen duena.*

Idazketa FSMak lau egoera ditu: *init*, *clear*, *do_shift* eta *waiting*. Sistema abiarazi zein berrabiaraztean lehenengoan mantentzen da *tg* seinaleak maila altua izan arte. Maila aldaketa ikusitakoan, *clear*-era aldatzen da, eta bertan RAMaren helbide guztietan zeroak idazten dira jarraian *do_shift*en lehenengo desplazamendua burutuz. Egindakoan, *waiting* egoeran mantentzen da eta *tg*ek berriz maila altua duen bakoitzean *do_shift*era aldaketa errepikatzen da eta *waiting*era buelta. Osagaiak datu esanguratsuak dagokien kokapenean dituen egoera bakarra *waiting* da beraz. Horregatik multiplexadore baten bitartez irteera zerora finkatu da RAMeko datuak mugitzen diren egoera guztietan. Ataka multzo biek erloju seinale berdina dutenez, bi edo lau lerro oso irudikatzeko behar den denbora tartean maila baxuan mantentzen ditu irtee-

rak. Murrizketa honen eraginez erabiltzaileak monitorean berrikuntza antzeman dezake. Posizio bateko errorea onartuz gero, murrizketa kendu daiteke, efektu hau ezabatuz.

s3e_vgashift osagaiaren *generic* parametroek sarrerako ataka diren eta egitura osoa baldintzatzen duten zutabe eta lerro seinaleen luzera eta moduluaren maximoa adierazten dituzte. Grafikoak koadrikularen gainean irudikatzen direnez, hauen dimentsioak zaindu behar dira (*unsigned* motakoak direla aintzat hartuta) ez daitezen eremutik kanpo agertu. Hori dela eta, zutabeei $10bit$ esleitu zaizkie eta 639-eko maximoa; eta lerroei, $8bit$ eta 234. Beraz, sistemaren laginketa guztiak aurkezteko *clka* atakako erloju seinalearen maiztasuna tg baino $640 \cdot 2 \approx 1,3 \cdot 10^3$ aldiz handiagoa izan behar da.

Zein koadrikulan irudikatuko diren erabakitzeko, jatorrizko *line* seinaleari bi *offset* gehitzen zaizkio (bata koadrikula bakoitzerako):

```
tline <= resize(line-244,8);
bline <= resize(line-1,8);
```

Era berean, VGA seinalea sortzean *offset*en arabera aurkezten dira. Ereduzko deskribapenean $dref \rightarrow qr$ eta $dvar \rightarrow qv$ aldagaiak goian irudikatzen dira eta $dout \rightarrow qo$ behean:

```
dvga <= (qr & qv & '0') when line>240 else ("00"&qo);
```

Grafiko bakoitzak gutxi gora behera pantailaren erdia hartzen duenez eta aldagaien seinaleak zeinudunak direnez, irudikatzean bereizmen osoa bermatzeko bigarren geruzak modulua baino ez du aurkezten (horregatik dira seinaleak zeinubakoak *vgashift* osagaian). Erabiltzaileak LCDa begiratu gabe argumentua (zeinua) zein den ikusi ahal izateko, VGAk lagintzen duen bakoitzean aldagai bakoitzeko bit bakarra gorde behar du sistemak.

Aldagaien hitz-luzera, bereizmena koadrikulen tamainaren menpekoa izanik, *dwl generic* parametroak adierazten du. Hiru grafikoek zabalera berdina dutenez, zeinu bit kopuru berdina gorde behar du sistemak eta MSBak bektore bakarrean, *dsign* seinalean, batzen ditu. Aldi berean, *abs* eta *resize* funtzioak medio, balioa absolutuak dituzten bit bat gutxiagoko *unsigned* seinaleak sortzen ditu:

```
dsign <= dref(dwl-1)&dvar(dwl-1)&dout(dwl-1);

vref <= resize(unsigned(abs(dref)),dwl-1);
vvar <= resize(unsigned(abs(dvar)),dwl-1);
vout <= resize(unsigned(abs(dout)),dwl-1);
```

Azalera eta kalkulu denbora murrizteko inplementatu den koma finkoko aritmetika erabiltzeak ezartzen dituen kuantizazio erroreak direla eta, baliteke aldagaiek ikusgai den azaleratik kanpoko balioak hartzea, normalizazio atalean adieraziko denez. irudiak pantailatik at geratu ez daitezen, maximoa ezartzen duen konstantea definitu da, eta seinaleak saturatzen dira *vgsahift* osagaietara bidali baino lehen:

```
constant lmax: natural:=234;
```

```
hfref <= to_unsigned(lmax,8) when vref>lmax else vref;
hfvar <= to_unsigned(lmax,8) when vvar>lmax else vvar;
```

```
hfout <= to_unsigned(lmax,8) when vout>lmax else vout;
```

Azkeneko geruza, beraz, alderanztailea da, zeinuaren arabera aldatuko du dagokion aldagaiaren aurkezpen kolorea. *s3e_vgasignshift* osagaia moldatutako *s3e_vgashift*a da, irudikatuko diren aldagaiek beste zutabe (helbide) dituena [45] eta datuen hitz-luzera hiru bitekoa delarik. Idazketa eta desplazamendurako ataka multzoaren logika berdina du, era eta aldi berean mugitu behar baitira *vgashift* eta *vgasignshift* osagaietako datuak, irteerak bat etor daitezen. Honetan, aldiz, ez dago bigarren multzoari lotutako logikarik, RAMeko datuak *column*en arabera irakurtzen ditu soilik.

	layer	*wl*	**source**	**component**	
-	0	3	-	-	Atzeko planoaren kolorea
g	1	1	SGRID	*vgasimpgrid*	Koadrikulak, kolore bakarra
dvga	2	3	SREF/SVAR/SOUT	*vgashift*	Grafikoak, hiru kolore
qsign	3	3	SIGN	*vgasignshift*	Zeinuak, hiru kolore

1.15. Taula: *DATA osagaiko geruza seinaleen jatorria eta izaera.*

1.14. taulan adierazitako kolore kodeak aztertuz eta atzeko planoari kolore beltza (000) esleituz, red-magenta, green-yellow eta blue-cyan bikoteetako lehenengo elementuek maila altuan bit bakarra dutela ikusi daiteke, eta bigarrenek bi. Azaldutako geruzei dagozkien seinaleen jatorria eta izaera laburbiltzen dituen 1.15. taulak adierazitakoarekin bateratuz, eta *dvga* seinalea zuzenean hartuta, *ref*, *var* eta *out* seinaleen balio positiboei bikoteen lehenengo koloreak esleitu zaizkie, hurrenez hurren. Eta balio negatiboei bikotearen beste kolorea, *qsign* seinalea *dvga*rekin bateratuz eta kolore kodeekin bat egiteko biten kokapenak moldatuz.

Hirugarren geruza bigarrenarekin bateratuta izanik, koadrikula gehitzea baino ez da falta. Datuen azpian irudikatu dadin, eta ez hauen gainetik, grafikoen seinaleak atzeko planoaren kolorea adierazten duenean baino ez da irudikatuko, eta egitean kolore zuriz (111) izango da.

```
smask <= dvga and qsign;
svga <= dvga or smask(1 downto 0)&smask(2);
vga <= (others => '1') when dvga="000" and g='1' else svga;
```

Gertaera zehatzak momentuan behar beste denborarekin aztertu ahal izateko VGA kudeatze sistemaren azkeneko elementua *switch* osagaian oinarritutako *PAUSE* da. Honek sakagailu bati eragitean *VGAMAN*en *tg* seinalea gaitzen edo ezgaitzen du. Ezgaituta dagoenean *vgashift* eta *vgasignshift* osagaiak ez dira berritzen, eta ondorioz monitorean irudi estatikoa agertzen da. Gaitu bezain laster berriz ere mugitzen dira grafikoak, geldirik egondako denboran izandako aldaketak agertu barik.

```
PAUSE: anie_switch port map ( clk0_1k,btndeb(2),srst,vtge );
ivtg <= (not vtge) and vga_tg;
```

[45] Grafikoek zabalera ezberdinak izango balituzte, parametro ezberdineko instantziak deskribatu beharko lirateke.

Debounce

Oinarrizko sistemak hiru sakagailu dituenez (LCDaren menuan ibiltzeko), VGAren modua eta gelditzea kudeatzeko biak gehituta *s3e_btndeb* osagaiak bost *s3etiny_debounce* instantzia ditu.

7.2.3. Normalizatzea

VGA-kudeatzailearen *VGADATA* osagaira doazen aldagai seinaleak normalizatzeko LCDaren aldagaiekin jarraitutako prozedura berdina errepikatu da. Hauen hitz-luzera koadrikulen tamainaren menpekoa izanik, *VGAMAN* osagaiaren *dwl generic* parametroaren berdina gehitu da.

Aldi berean, *VGADATA* osagaien FSMak *clk*2 erlojuak agintzen dituenez, sarrerako aldagaien laginketan oinarrituta eta bihurketak burutzeko denbora eman ostean, *vga_tg* seinaleak *clk*2 erlojuaren periodo batez maila altua hartzen du. Honek laginketa bakoitzeko posizio bat baino gehiago mugitzea ekiditen du.

Kuantizazioari dagokionez, zeinudun seinaleak biko osagarraian kodetzen direnez, aldagai bipolar asimetrikoak dira, n seinalearen bit kopurua izanik $[-2^n, 2^n - 1]$ tarteko zenbaki hamartarrak adierazi daitezkeelarik. Honek emaitzan, zeinu aldaketak gertatzean, eskala aldatzean eta balio absolutuak kalkulatzean zenbaki positibo eta negatiboen bihurketan unitate bateko diferentzia sor dezake. Normalizatzean jatorrizko seinaleen eskala amaiera geroz eta irteerekiko txikiagoa izan, orduan eta nabarmenago da eragina.

7.2.4. UCF

```
NET "clk" LOC = "C9"  |IOSTANDARD=LVCMOS33;
NET "clk" TNM_NET=clk;
TIMESPEC TS_clk=PERIOD "clk" 20 ns HIGH 40%;

NET "rot(0)" LOC="K18" |IOSTANDARD=LVTTL |PULLUP;
NET "rot(1)" LOC="G18" |IOSTANDARD=LVTTL |PULLUP;

NET "LCD_E"  LOC="M18" |IOSTANDARD=LVCMOS33 |DRIVE=4 |SLEW=SLOW;
NET "LCD_RS" LOC="L18" |IOSTANDARD=LVCMOS33 |DRIVE=4 |SLEW=SLOW;
NET "LCD_RW" LOC="L17" |IOSTANDARD=LVCMOS33 |DRIVE=4 |SLEW=SLOW;

NET "SF_D(0)" LOC="R15" |IOSTANDARD=LVCMOS33 |DRIVE=4 |SLEW=SLOW;
NET "SF_D(1)" LOC="R16" |IOSTANDARD=LVCMOS33 |DRIVE=4 |SLEW=SLOW;
NET "SF_D(2)" LOC="P17" |IOSTANDARD=LVCMOS33 |DRIVE=4 |SLEW=SLOW;
NET "SF_D(3)" LOC="M15" |IOSTANDARD=LVCMOS33 |DRIVE=4 |SLEW=SLOW;

NET "hsync"  LOC="F15" |IOSTANDARD=LVTTL |DRIVE=8 |SLEW=FAST;
NET "vsync"  LOC="F14" |IOSTANDARD=LVTTL |DRIVE=8 |SLEW=FAST;
NET "vga(0)" LOC="G15" |IOSTANDARD=LVTTL |DRIVE=8 |SLEW=FAST;
NET "vga(1)" LOC="H15" |IOSTANDARD=LVTTL |DRIVE=8 |SLEW=FAST;
```

```
NET "vga(2)" LOC="H14" |IOSTANDARD=LVTTL |DRIVE=8 |SLEW=FAST;

NET "btn(0)" LOC="V4"  |IOSTANDARD=LVTTL |PULLDOWN;
NET "btn(1)" LOC="H13" |IOSTANDARD=LVTTL |PULLDOWN;
NET "btn(2)" LOC="K17" |IOSTANDARD=LVTTL |PULLDOWN;
NET "btn(3)" LOC="D18" |IOSTANDARD=LVTTL |PULLDOWN;
NET "btn(4)" LOC="V16" |IOSTANDARD=LVTTL |PULLDOWN;

NET "rst" LOC="N17" |IOSTANDARD=LVTTL |PULLUP;
```

8.
Azterketa kasuak

Spartan3E Starter Kit atalean adierazi denez, plantak bortizki baldintzatzen ditu sarrerako/irteerako osagaien deskribapena. Txostenean adierazitakoak oinarritzat harturik, jarraian planta aurkeztu ostean sistema osatu eta erabiltzeko garatu beharrekoak eta egin beharreko kalkulu zein doiketak azalduko dira. Saiakuntza praktikoen emaitzekin batera praktikan sistemaren portaera zein den adieraziko dute.

8.1. Korronte zuzeneko motorra

Alecop, S. Coop. erakundeak eginiko MV-541 maketa erabili da lehenengo azterketa kasu gisa. Korronte zuzeneko motor batek eragindako ardatzean industrian aurkitu daitezkeen kaptadore eta sentsore ezberdinak batzen ditu. Abiadura irakurtzeko takodinamoa eta deslazamendu angularrerako kodetzaile absolutu eta inkrementala ditu. Biraketa angelua adierazten duen murriztailea ere badu, zeinen posizio angeluarra kaptadore erresistibo baten jasotzen duen. Maketaren gidan [1] ikus daitezkeenez 1.16. taulak aipatutakoen ezaugarriak biltzen ditu.

K.Z. Motorra	V_{max}: $\pm 12\,\mathrm{V}$ I_n: 0,6 A Abiadura izendatua: $1200 rpm$
Takodinamoa	Bereizmena: $7\frac{\mathrm{mV}}{rpm}$
kodetzaile inkrementala	Bereizmena: $50\frac{impultsu}{bira}$ Sentsore kopurua: 2 (SA eta SB) Elikadura: 5 V
kodetzaile absolutua	Bereizmena: $4bit \rightarrow 22{,}5°$ edo $\frac{1}{8}\pi rad$ Elikadura: 5 V
Murriztailea	Murrizketa: 1 : 30
Kaptadore erresistiboa	R_n: $10 \pm 20\,\%\mathrm{k}\Omega$ P_{max}: 1 W Linealtasuna: $\pm 2\%(R_n)$ Errotazio erabilgarria: $340 \pm 4°$ Errotazio mekanikoa: 360° jarraitu

1.16. Taula: *MV-541 maketaren gidan adierazitako ezaugarriak.*

Sistema orokorrean eta kontrolagailua bereziki frogatzeko planta geldoa eta zeharo egonko-

rra egokia izateagatik aukeratu da. Honela, kontrol sinplea behar duen plantan, deskribapen ezegokiaren ondorioz egon litezkeen arazoak antzeman daitezke, eta aztertutakoak praktikan zuzenean jarri. Kaptadoreak maketan ikusgai egoteak eta hainbat izateak ere, helburu akademikoa duen proiektu honetan aukera anitz eskaintzen ditu. Konexioak aurreko aldean izanik, argi adierazita, eta fabrikatzailearen fidagarritasunarekin, muntaia zeharo errazten da, txertatutako sistemaren garapenerako denbora gehiago utziz.

8.1.1. Interfazeak aukeratzea eta muntaia egitea

Aldagaia irakurtzea

Maketak eskaintzen dituen kaptadoreetan sinpleena kodetzaile inkrementala da. Honek bi seinale baino ez ditu, sarrera digitalen bitartez zuzenean irakurri daitezkeenak, eta abiadura zein posizio kontrolerako inplementatu beharreko logikak zeharo sinpleak dira.

Kodetzaile absolutuak seinalea kopuru bikoitza du, sarrera digital kopurua bikoitza eskatuz. Abiadura zein posizio kontrolerako inplementatu beharreko logikak ere oso sinpleak dira. Baina inkrementalarekin konparatuz, honen bereizmena askoz txikiagoa da.

Kaptadore erresistiboak seinale analogiko bakarra ematen du. Inkrementala erabilita baino seinale gutxiago behar ditu, eta abiadura zein posizio kontrolerako logiken konplexutasuna absolutuaren parekoak dira, modulu aldetik antzeko izaera baitute. Baina, sistema digitala izanik, AD bihurgailuaren beharrak bitarteko beste gailu bat eskatzen du, antzeko bereizmena izateko. Horrek kontrol begizta korapilatzen du, eta neurketak zuzenak izan daitezen bitarteko etapa beharrezkoa izan daiteke. Bestetik, lineala ez denez, eman liratekeen kuantizazio erroreak aztertu behar dira.

Takogeneradoreak ere seinale analogiko bakarra ematen du, eta abiadura kontrolerako baino ez da baliagarria. Seinalea mugimenduaren menpekoa da, ez du posizioak agintzen aurrekoekin gertatu bezala. Halere, seinale jarraitua izanik, kodetzaileek baino bereizmen zeharo hobea du.

Txertatutako sistema azaleraren ikuspuntutik ahalik eta txikiena mantentzeko kodetzaile inkrementala aukeratu da, beraz, kontrol begiztarako. Aldi berean, plantaren identifikaziorako saiakuntza soila egin behar denez, takogeneradorea baliatuko dugu zuzenean osziloskopiora konektatuz.

Irteera atontzea

*Artaloytia*k [41] adierazten duenez, elektronikaren ikuspuntutik korronte zuzeneko motorrak eragiteko hamaika aukera daude, zeinetan eraginkorrenak potentziako kommutadore elektronikoz (transistore bipolarrak, IGBTak eta gehienetan MOSFETak) osatutako bihurgailuetan oinarritutakoak diren. Laneko modua kommutazioa izanik, bihurgailuan ematen diren galerak oso txikiak dira bihurgailu linealekin konparatuz, azkeneko hauek tarte aktiboan egiten baitute lan.

on egoeran kommutazio osagaiak oso tentsio jauzi txikia du, nahiz eta korrontea oso handia

izan. *offen*, aldiz, nahiz eta tentsio jauzia oso handia izan, zeharkako intentsitatea mikroampere ingurukoa da, edo txikiagoa. Edozein egoeratan osagaietan galerak oso txikiak dira, potentzia handiko aplikazioetarako egokiak izanik.

Kommutadore elektronikoz osatutako zubi osoko lau koadranteko CC/CC bihurgailuek (H-zubiek) Pulse Width Modulation (PWM) teknika bitartez kontrolatzeko aukera eskaintzen duenez, motorearen desplazamenduaz gain noranzkoa ere kontrolatu daitekeenez, eta sistema digital batez PWM motako seinalea sortzeko beharreko logika sinplea eta azkarra izanik, halakoa erabili da.

Muntaia

Spartan-3E Starter Kit txartelaren fabrikatzaile berdina den *Digilent*en *PmodHB3* moduluak, 2 A eta 12 V-eko H-zubia eta kodetzaile inkrementalaren sarreretarako bi *buffer* bateratzen ditu. Planta den motorraren ezaugarriak gogoratuz, tentsio maximo bera dute, eta H-zubiak korronte izendatua baino $\frac{2}{0{,}6} = 3,34$ aldiz handiagoa jasan dezake.

Moduluaren erreferentzia-gidak [42] adierazten duenez, kontrol txartelaren J1 eta J2 konektoreekin bateragarria den 6 pineko konektorea du. Kable egokia erabiliz gero, J4 konektorea ere erabili daiteke.

Motorra eta elikadura iturria moduluan adierazitako konektoreetara konektatu dira zuzenean. Kodetzaileko seinaleak *buffer*ak iragateko dagozkien pinetara konektatu badira ere, ez dira fotodiodo eta fototransistoreak moduluak eskainitako pinekin elikatu. Txartelak 3,3 V-eko tentsioarekin egiten du lan. Nahiz eta *PmodHB3* moduluak $3,3-5$ V tartean funtzionatu, maketako osagaiek 5 V behar dituzte. Horregatik motorrarentzat erabilitako elikadura iturri bereko zirkuitu digitalak elikatzeko irteera erabili da sentsoreetan.

8.1.2. Input

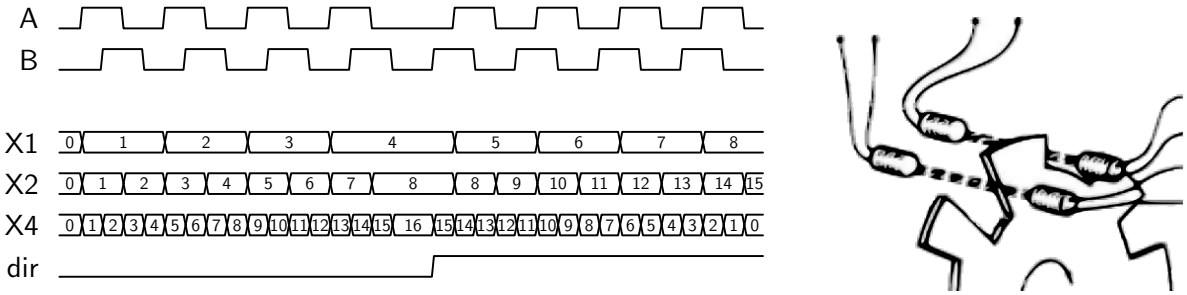
Maketaren gidako [1, pp. 24-34] azalpen zehatza jarraituz, kodetzaile inkrementalaren A eta B sentsoreen seinaleak erabilita eta inpultsu kopuru berdina izanik irakurketa logikaren arabera bereizmena bikoiztu edo lau aldiz handitu daiteke.

Irakurketa modua

Oinarria sentsore bakar bat erabiltzea eta goranzko aldaketak aztertzea da (X1), maketak ezaugarrietan adierazitako inpultsu kopurua irakurriz. Sentsore bakarraren goranzko eta beheranzko aldaketak aztertuz gero (X2), inpultsu bikoitza irakurtzen ditu, ondorioz gailu berdinak izanik logika soilik aldatuz bereizmena bikoizten da. Bi modu hauek, ordea, ezin dute norabidea zein den adierazi, erreferentzia bakarra izategatik.

Azkeneko aukera bi sentsoreen goranzko zein beheranzko aldaketak aztertzea da (X4). Bien arteko kokapena oinarrizko bereizmen tartearen erdia bada, 1.36. irudiko kronograman adierazi

denez, lau aldiz handitzen da eta bien arteko erlazioak noranzkoa (*dir*) asmatzea ere ahalbidetzen du.



1.36. Irudia: *sa eta sb sentsoreak dituen kodetzaile inkrementalaren irakurketa moduak.*

Garapenaren helburua txertatutako sistema modularra eta, ahal den heinean, parametrizatua denez, hiru irakurketa moduak erabiltzeko logika deskribatu da. Hala ere, bereizmen hobea izategatik eta sistemak noranzkoa ere kontuan hartzen duelako, praktikan X4 modua erabiliko da.

Gidan ez bezala, *anie_trigger* osagaiak logika digitala erabiliz deribatzen ditu sarrerak. Seinaleen aldaketak baino zeharo azkarragoa den erloju seinalea baliatuz, etengabe lagintzen dira *sa* eta *sb* seinaleak. *encsel* modu aukeratzaileak agindutako multiplexadore batek lagindutako eta momentuko balioak aztertzen dituzten ate logikoz osatutako hiru seinaleren artean aukeratzen du.

Modua	X1	X2	X4
encsel	1	2	3
Irteera	$(s \cdot z^{-1})$ *and not* s	$(s \cdot z^{-1})$ *xor* s	$((sa \cdot z^{-1})$ *xor* $sa)$ *or*
	non $s \rightarrow sa \vert sb$	non $s \rightarrow sa \vert sb$	$((sb \cdot z^{-1})$ *xor* $sb)$

1.17. Taula: *Irakurketa moduaren araberako sa eta sb seinaleen azterketa.*

dir seinalearen egoera, noranzkoa, *sa*ren goranzko aldaketetan *sb*ren egoerak adierazten du. Azaldu bezala, X1 zein X2 moduetan ezin da noranzkoa zein den asmatu, beraz, *dir*ek maila baxu finkoa izango du $encsel/=3$ denean.

Inpultsuak modulu bihurtzea

Adierazitako kronograman ikusi daitezkeen seinaleen balioak erdiesteko, *count_wl* luzerako *udcounter* osagaiaren instantzia gehitu da. *trigger*en erloju berdina agintzen du, eta honen irteerek *en* eta *dir* atakak kudeatzen dituzte. Honela, kodetzaile inkrementalak emandako inpultsuak $[0, 2^{count\_wl} - 1]$ tarteko modulua duen seinale bihurtu dira.

Laginketa-maiztasunak aginduta, eta burutu nahi den kontrolaren arabera, moduluaren alda-

ketak *anie_adq* osagaiak interpretatzen ditu. Abiadura kontrolerako, laginketa kT_s eta $(k-1)T_s$ artean moduluak duen diferentzia kalkulatzen du eta *var_wl generic*ak adierazitako luzerara mozten du. Diseinatzaileak, beraz, irakurketa modua eta laginketa-maiztasunaren arabera *var_wl* eta *count_wl* ezarri beharko ditu, (1.18) eta (1.19) adierazpenak betez.

$$2^{var\_wl} - 1 > \omega_i \cdot T_s \qquad \omega_i : \text{abiadura izendatua} \rightarrow \left[\frac{rpm}{60 \cdot s} \cdot \frac{\text{inpultsu}}{bira} \cdot modua\right] \tag{1.18}$$

$$count\_wl \geq var\_wl \tag{1.19}$$

overflow & underflow

Kontagailuaren irteera mugatua denez eta gorantz zein beherantz aldatu daitekeenez, noranzkoa berdina mantenduz gero $\frac{2^{count\_wl}}{\omega_i \cdot T_s}$ laginketa burutzean *overflow*a edo *underflow*a gertatuko da, hau da, maximo eta minimoa zeharkatuko ditu moduluak. Horrek sortzen dituen, eta kontrola ezegonkortu dezaketen, irakurketa arazoak ekiditeko *Martinez et al.*ek [4, pp. 17-26] proposatutako detekzio eta zuzenketa logika deskribatu da.

```
comp(0) <= '1' when diff>0 else '0';    comp(1) <= '1' when diff=0 else '0';
comp(2) <= comp(1) or (dir_int(1) xor dir_int(0)); --stop -> diff=0 or DIR
    change

gates(0) <= dir_int(0) and comp(0); --underflow -> diff>0 and DIR=1
gates(1) <= dir_int(0) nor comp(0); --overflow -> diff<0 and DIR=0

sel(1) <= comp(2) or gates(1);    sel(0) <= comp(2) or gates(0);

cor <= nlimit when gates(0)='1' else plimit; --correction limit
sum <= resize((diff+cor)+1,var_wl); --correction & 0-trough compensation

with sel select
tvar <= resize(diff,var_wl) when "00", (others => '0') when "11", sum when
    others;
```

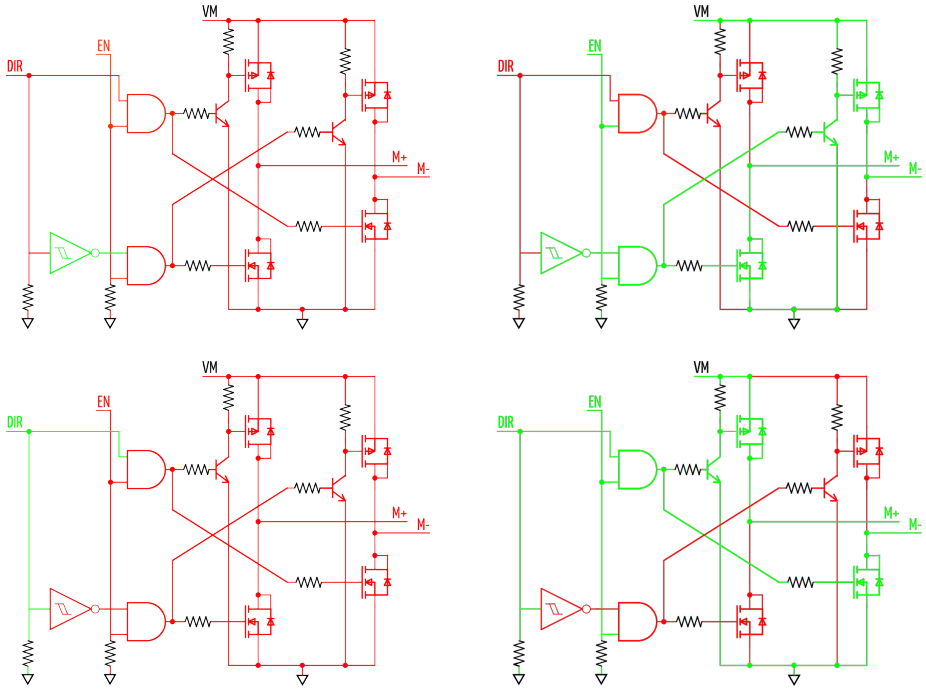
smooth

Kodetzailearen sentsoreen seinaleek zarata izatekotan garbitzeko eta inpultsu asalduren eragina murrizteko, txarteleko kommutadore batek agindutako *smooth* seinaleak *trigger* eta *adq* osagaietan iragaziak gehitzen ditu. Lehenengoan, *Chapman*ek [38] proposatutako eta *rot* osagaian deskribatutakoak *sa* eta *sb* seinaleen sekuentzia zuzena dela ziurtatzen du irakurketa modua X4 denean. Bigarrenak, azkeneko hiru laginketen balioetako edozein zero ez den bitartean, (1.20) adierazpenaren arabera ezartzen du sisteman plantaren aldagaia izango den seinalea.

$$\frac{2 \cdot var + var \cdot z^{-1} + var \cdot z^{-2}}{4} \tag{1.20}$$

Iragaziak aldaketa bortitzak leuntzen dituenez, begizta itxian erreferentzian maila aldaketei erantzunek gainditze handiagoa izan dezakete, kontrolagailuak benetako abiaduraren $\%50-100$ artean jasotzen baitu. Hala ere, laginketa-maiztasuna begiztaren banda-zabaleraren arabera ezarriz gero eta askoz azkarragoa izanik, kontrolak berak zuzendu dezake.

8.1.3. Output



1.37. Irudia: PmodHB3 *moduluko H-zubiaren* en *eta* dir *seinaleen araberako egoerak.*

CC-CC bihurgailu baten transistoreak kommutatzeko aukera errazenetarikoen artean frekuentzia finkoan *on* eta *off* egoeren iraupena, hau da, lan-erregimena aldatzea da. Pultsu zabaleraz modulatutako bihurgailuak dira hauek, ingelesez, *Pulse Width Modulation* (PWM).

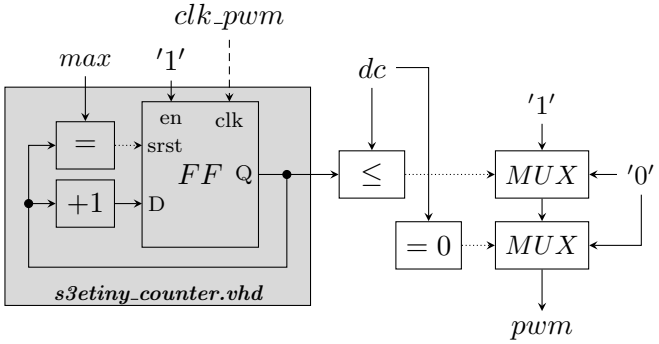
Sistema digital batez halako modulazioa kontagailu bakarra eta zenbait konparadore erabilita egin daiteke, eta horregatik hainbat abantaila aurkezten ditu *Artaloytia*k [41]. Hala nola, lan-erregimena %0 eta %100 artean aldatzeko aukera; maiztasuna, kommutazio denbora eta sekuentzia, etab. aldatzeko malgutasuna) edota plantak jasotzen duen bataz-besteko balioa lan-erregimenarekiko linealki proportzionala izatea[46].

PmodHB3 modulua modu bipolarrean lan egiteko dago prestatuta: lau transistoreak bi kommutazio pare gisa hartzen ditu, binaka kommutatzen direlarik. Testuan [41] azaldutako PWM sorrerarako prozedura orokorrek pare bakoitza gaitzeko seinale bana darabilte, alderantzizko balioko bi seinaleren iraupenaren bitartez aldi berean modulua eta noranzkoa kudeatuz. Honela, lan-erregimena %50era ezartzean plantak ez du korronterik jasotzen, %0 eta %100 balioek noranzkoa batean eta bestean maximoa esleitzen dutelarik.

[46] Transistoreetan tentsio jauziak mesprezatuz eta kommutazioa ideala dela onartuz.

Kontroleko seinaleak sortzea errazteko, eskeman ikusi daitekeenez eta 1.37. irudiak azaltzen duenez, *Digilent*en moduluan bi *and* eta *not* batez osatutako logikak kudeatzen ditu kommutazio pareak eta ataka banatan eskaintzen ditu lan-erregimena eta noranzkoaren kontrola. Honela *OUTPUT* atalak kontrolagailuaren irteeraren moduluarekiko, balio absolutuarekiko, proportzionala den PWM seinalea sortuko du, *antiphase* prozedurarekiko bereizmena bikoiztuz.

clk_pwm erlojuak agindutako, *dc_wl* luzerako eta muga maximoan ezarrita duen *counter* osagaia oinarri, modulua luzera bereko *dc* seinalearekin konparatzen da. $\leq$ motakoa irteerak $(0, 100]$ tarteko lan-erregimenak izango ditu. Agindua zero denean irteera ere halakoa izateko, $= 0$ motako bigarren konparadorea gehitu da, 1.38. irudian ikusi daitekeenez, irteera maila baxuan finkatzen duena.



1.38. Irudia: *dc lan-erregimeneko eta* $\frac{clk\_pwm}{2^{dc\_wl}}$ *maiztasuneko PWM seinalea sortzea.*

Planta eragingo duen H-zubiak anplifikatutako PWM seinalearen maiztasuna, beraz, *clk_pwm* erlojuaren eta irteeraren bereizmenaren menpekoa da, (1.21) adierazpenak baldintzatzen duenez. Ondorioz, bereizmena hobetzeko luzera handitzen goazen heinean, modulazio-maiztasun berdina mantentzeko erlojuaren maiztasuna handitu behar da. Egun txartelek dituzten erloju maiztasunekin bereizmen nahikoa lortu daiteke. Esaterako, garatutako sistemak 50 MHz-eko erlojua izanik, 20 kHz modulazio-maiztasuna izateko luzera maximoa $\frac{50\,\text{MHz}}{20\,\text{kHz}} = 2500 = log_2\, dc_wl \rightarrow 11\,bit$ekoa da. Hala ere, bereizmena hobetzeko H-zubiaren elikadura tentsioak ere kontrolatu daitezke, *Artaloytia*k proposatu bezala.

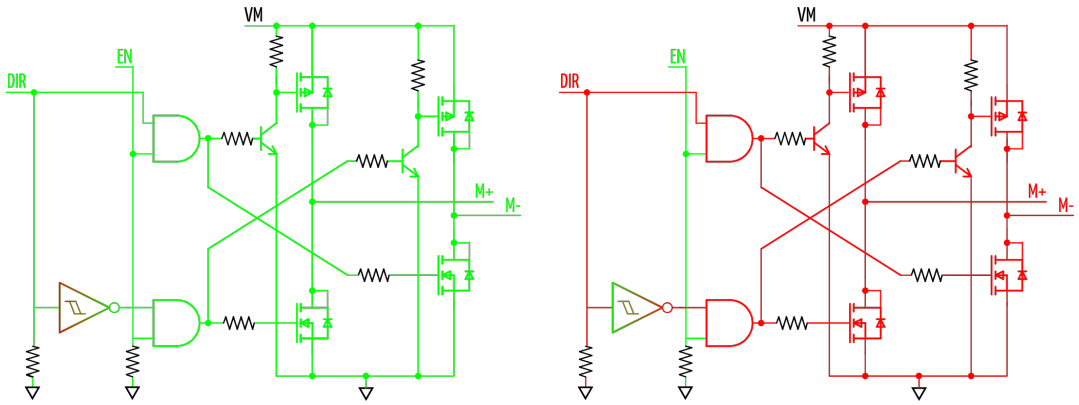
$$\left.\begin{aligned} F_o &= \frac{F_{pwm}}{2^{dc\_wl}} \\ \text{duty-cycle} &= \frac{\text{unsigned}(dc)}{2^{dc\_wl}-1}\cdot 100 \end{aligned}\right\} \quad 2^{dc\_wl} = \frac{F_o}{F_{pwm}} = \frac{\text{unsigned}(dc)\cdot 100}{duty-cycle} + 1 \tag{1.21}$$

Modulazio-maiztasunak motorrak periodoa osotasunean antzeman dezan nahikoa izan behar du, hau da, atal mekanikoen erantzun-ahalera baino azkarragoa izan beharko dute maila baxuko egoerak. Begizta irekian plantaren banda-zabalera aztertuz egin da kalkulua, eta kontrol begiztaren maiztasuna ezartzerakoan egin bezala modulazioa behintzat hamar aldiz handiagoa ezarri da. Honela lan-erregimenarekiko linealki proportzionala den bataz-besteko elikadura antzemango duela ziurtatzen da.

Aldi berean, bibrazioak eta matxurak ekiditeko modulazioak motorraren erresonantzia-maiztasuna baino handiagoa izan beharko du. Fabrikatzaileak ezaugarri-orrien bitartez adieraziko du maiztasunen arabera zein eragin eman daitekeen.

Aurrekoak eta modulazio-maiztasuna handitu ahala elikaduraren osagai alternoa murrizten dela aintzat hartuta, ahal bezain handi ezartzea litzateke ideala. Haatik, bereizmena bermatzeaz gain, maiztasunak handiak kommutazioan galerak ere handitzen ditu. Laburbilduz, bandazabalera eta erresonantziak ezarritako murrizketak betetzen dituen maiztasun txikiena ezarri da.

PmodHB3 moduluaren *dir* seinalea *and* ate batera bestera baino lehenago heltzen denez (*not* ateak eragindako atzerapenagatik), aldatzean *glitch*ak agertu daitezke. *en*ek maila baxua duen bitartean bi kommutazio pareak moztuta daude eta ez dago inolako arriskurik. Maila altua duenean, aldiz, eta pareetako bat behintzat *on* dagoenean, *dir* aldatuz gero bi pareek *not* atearen atzerapenak iraun bitartean egoera berdina hartzen dute (1.39. irudia) [4, pp. 34-35].

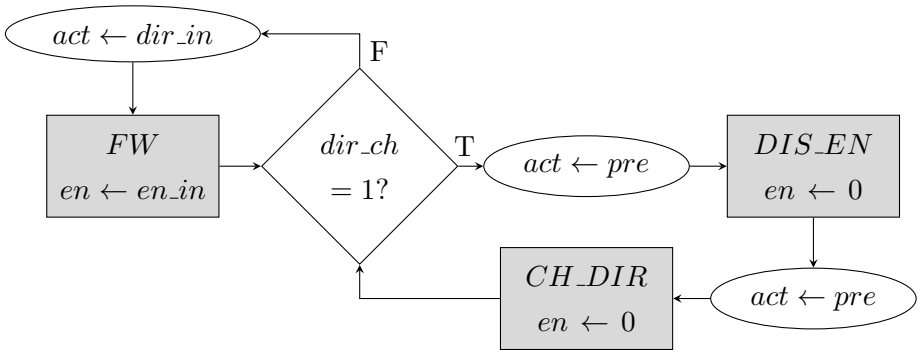


1.39. Irudia: PmodHB3 *moduluaren ate logiken* glitch*ak.*

*dir*en goranzko aldaketan bi kommutazio pareak *on* jartzen badira, VMek zirkuitulaburra egiten du H-zubiak zehar, transistoreak errez. Modulua babesteko, beraz, sistemak beharrez *dir* aldatu bitartean maila baxuan jarri beharko du *en*. Nahiz eta beheranzko aldaketetan erretzeko arriskurik ez egon (bi pareak *off* jartzen direlako), noranzko aldaketa edozein dela ere prozedura jarraituko du *anie_hbridge* osagaiak.

1.40. irudiak PWM osagaiaren irteera *en_in* gisa hartuta eta kontrolagailuaren noranzkoa *dir_in* bezala, txartelaren irteerak zuzenean kudeatzen dituen FSMaren fluxu diagrama aurkezten du. Sarrerak *clk_hb* erlojuaren arabera lagintzen ditu eta noranzkoan aldaketarik ez dagoen bitartean zuzenean esleitzen ditu seinaleak. Aldatzean, aurreko balioa mantentzen du *en* ezgaitu baino lehen, eta noranzkoa bi laginketetan behintzat egonkorra bada *act* seinaleak noranzko berria hartzen du. Aldatu eta egonkortu bitartean, ezgaituta mantentzen da H-zubia.

```
process(clk,srst)
begin
if rising_edge(clk) then
 dir_ipre <= dir_in;
 pre <= act;
```

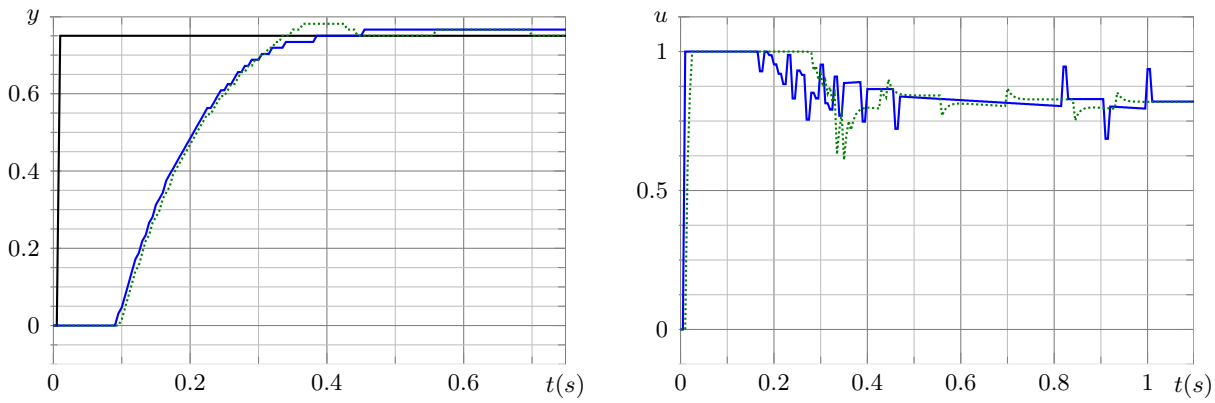


1.40. Irudia: *H-zubia ez erretzeko noranzko aldaketak kudeatzeko FSMa.*

```
 if dir_ch='1' then
      hbga <= dis_en; wa <= (others => '0');
 else hbga <= hbgn;   wa <= wn; end if;
end if;
end process;

dir_ch <= dir_ipre xor dir_in;
dir_out <= act;
en_out <= en_in when hbga=fw else '0';
```

8.1.4. Modeloa, PIDaren doiketa eta T_s



1.41. Irudia: *Doiketa: 1.18. taulan adierazitako doiketarekin maila sarrerari erantzuna (iragazitako jarraitua eta diskretua).*

2. Dokumentuan bildutako saiakuntzen emaitzak baliatuz eta hobestutako plantaren modeloa erabiliz, 1.18. taulan adierazitako doiketa ezarri da, bertan azaltzen diren baldintzekin.

Plantaren modeloa eta kontrolagailua doituta izanik, begizta itxiaren banda zabalera aztertu da, 4.4. atalean adierazitakoak jarraiki. 9,95 Hz-eko banda zabalera izanik, $\approx 10 \cdot 20 = 200$ Hz-eko laginketa-maiztasuna ezartzea litzateke seguruena. Behin-betikotzat jo baino lehen kodetzaile

P	1.625	Step time	.01	Values	(0 : .75)
I	6.25	Rise time	.275	Rise	90%
D	0.075	Setting time	.65	Setting	2.5%
		Overshoot	2.5%	Undershoot	2.5%

1.18. Taula: *Doiketa: korronte zuzeneko motorra.*

inkrementalaren bereizmena aztertu behar da (1.19. taula).

$$\frac{1200 \cdot 50}{60} \cdot T_s \cdot modua = 1000 \cdot modua \quad (\frac{inpultsu}{laginketa})$$

	$T_s = 5ms$	$T_s = 10ms$
X1	5	10
X2	10	20
X4	20	40

1.19. Taula: *Laginketa-maiztasuna eta irakurketa moduaren araberako bereizmena.*

Egoerarik txarrenean, *X1* moduan, F_s =200 Hz maiztasunarekin bereizmena murritzegia da kontrola praktikoa izan dadin. Sarreran etengabeko aldaketa bortitzak izateak kontrolagailuaren erantzuna ezegonkortu dezakenez, T_s =10 ms aukeratu da, bereizmen bikoitza ematen duena *X10* gain-laginketa faktorearekin.

8.1.5. Sistema osatzea

1.23. irudian adierazitako egitura jarraituz, *input*, *output*, *man*, *timing* eta *pid* moduluak fitxategi bakarrean bateratu dira, *norm* osagaiak elkarren arteko datu-fluxua kudeatzen duelarik.

Sarrerak

Parametroak ezartzerakoan, *var_wl* eta *count_wl* kalkulatzeko (1.18) eta (1.19) erabili dira, hurrenez hurren. Kalkulu erroreak murrizteko *ref_wl* eta *i_wl* parametroetan *var_wl*en luzera berdina jarri dira.

Irteera

Kontrolagailuaren irteeraren luzera ezartzeko (1.21) adierazpena erabili da eta bereizmena eta PWM seinalearen maiztasunaren arteko orekan *o_wl* parametroari 11 esleitu zaio. F_{pwm} = 5 MHz izanik, $\frac{5\,\text{MHz}}{2^{10}} = 4{,}89\,\text{kHz}$ maiztasuneko irteera du sistemak.

Konstanteak

6. atalean jarraitutako prozedura berdinak konstanteentzat 1.20. taulak biltzen dituen emaitzak eman ditu. *k_wl* parametroak, beraz, 5*bit*eko luzera adierazten du, eta *minbp* 0 da.

	Zehatza	Konstantea	Koma bitarra	Hitz-luzera
$K_p = P$	1.625	13	3 (pbp)	4+1
$K_i = I \cdot \frac{T_s}{2}$	0,03125	1	5 (ibp)	1+1
$K_d = D \cdot \frac{2}{T_s}$	15	15	0 (dbp)	4+1

1.20. Taula: *Kontrolagailuaren konstanteak koma finkoan adieraztea.*

5. atalean azaldu bezala, zehaztasun osoa bermatzeko irteerak $o_{wl} = i_{wl} + k_{wl} + irem_{wl} + 1$ baldintza bete behar du. 7+5+1+1 = 14 izanik, 3*bit* mozten dira. Honen eraginez irteera $2^3 = 8$ aldiz geldoagoa da, aldaketa txikienek ez baitute irteeran eraginik. Horrek zeharo moteltzen du kontrola eta praktikan eskasa da.

Aurrekoa ekiditeko 1.20. taula adierazitako koma bitarraren *generic*ei hiru unitate kendu zaizkie, hau da, 8 aldiz handiagoak dira konstanteak: $pbp = 0$, $ibp = 2$ eta $dbp = -3$.

Normalizazioa

Kontrol-sistema osatzen duten modulu ezberdinen arteko datu-fluxua kuantitatiboki zuzena izan dadin, 1.21. taulan bildutako konstanteak deklaratu dira, eta *norm_wl* parametroari 8 egotzi zaio.

Seinalea	Zehatza	Konstantea		Hitz-luzera	Maximo adierazgarria
npidr	40/40	= 1	1 (0)	1+1	40
npidv	40/40	= 1	1 (0)	1+1	40
nopenr	1023/40	≈ 25.5	102 (2)	7+1	1020
nlcdr	1000/40	= 25	25 (0)	5+1	1000
nlcdv	1000/40	= 25	25 (0)	5+1	1000
nlcdo	1000/1023	≈ 0.984375	63 (6)	6+1	1007,015625
nvgar	235/40	= 5.875	47 (3)	6+1	235
nvgav	235/40	= 5.875	47 (3)	6+1	235
nvgao	235/1023	≈ 0.23046875	59 (8)	6+1	235,76953125

1.21. Taula: *Normalizaziorako konstanteak koma finkoan adieraztea.*

8.1.6. UCF

```
NET "clk" LOC = "C9"  |IOSTANDARD=LVCMOS33;
```

```
NET "clk" TNM_NET=clk;
TIMESPEC TS_clk=PERIOD "clk" 20 ns HIGH 40%;

NET "rot(0)" LOC="K18" |IOSTANDARD=LVTTL |PULLUP;
NET "rot(1)" LOC="G18" |IOSTANDARD=LVTTL |PULLUP;

NET "LCD_E"  LOC="M18" |IOSTANDARD=LVCMOS33 |DRIVE=4 |SLEW=SLOW;
NET "LCD_RS" LOC="L18" |IOSTANDARD=LVCMOS33 |DRIVE=4 |SLEW=SLOW;
NET "LCD_RW" LOC="L17" |IOSTANDARD=LVCMOS33 |DRIVE=4 |SLEW=SLOW;

NET "SF_D(0)" LOC="R15" |IOSTANDARD=LVCMOS33 |DRIVE=4 |SLEW=SLOW;
NET "SF_D(1)" LOC="R16" |IOSTANDARD=LVCMOS33 |DRIVE=4 |SLEW=SLOW;
NET "SF_D(2)" LOC="P17" |IOSTANDARD=LVCMOS33 |DRIVE=4 |SLEW=SLOW;
NET "SF_D(3)" LOC="M15" |IOSTANDARD=LVCMOS33 |DRIVE=4 |SLEW=SLOW;

NET "hsync"  LOC="F15" |IOSTANDARD=LVTTL |DRIVE=8 |SLEW=FAST;
NET "vsync"  LOC="F14" |IOSTANDARD=LVTTL |DRIVE=8 |SLEW=FAST;
NET "vga(0)" LOC="G15" |IOSTANDARD=LVTTL |DRIVE=8 |SLEW=FAST;
NET "vga(1)" LOC="H15" |IOSTANDARD=LVTTL |DRIVE=8 |SLEW=FAST;
NET "vga(2)" LOC="H14" |IOSTANDARD=LVTTL |DRIVE=8 |SLEW=FAST;

NET "btn(0)" LOC="V4"  |IOSTANDARD=LVTTL |PULLDOWN;
NET "btn(1)" LOC="H13" |IOSTANDARD=LVTTL |PULLDOWN;
NET "btn(2)" LOC="K17" |IOSTANDARD=LVTTL |PULLDOWN;
NET "btn(3)" LOC="D18" |IOSTANDARD=LVTTL |PULLDOWN;
NET "btn(4)" LOC="V16" |IOSTANDARD=LVTTL |PULLDOWN;

NET "dir" LOC="L13" |IOSTANDARD=LVTTL |PULLUP;
NET "en"  LOC="L14" |IOSTANDARD=LVTTL |PULLUP;
NET "smooth" LOC="H18" |IOSTANDARD=LVTTL |PULLUP;
NET "rst" LOC="N17" |IOSTANDARD=LVTTL |PULLUP;

NET "j4o(0)" LOC="D7" |IOSTANDARD=LVTTL |SLEW=SLOW |DRIVE=6;
NET "j4o(1)" LOC="C7" |IOSTANDARD=LVTTL |SLEW=SLOW |DRIVE=6;
NET "j4i(0)" LOC="F8" |IOSTANDARD=LVTTL |SLEW=SLOW;
NET "j4i(1)" LOC="E8" |IOSTANDARD=LVTTL |SLEW=SLOW;

NET "led(7)" LOC="F9"  |IOSTANDARD=LVTTL |SLEW=SLOW |DRIVE=8;
NET "led(6)" LOC="E9"  |IOSTANDARD=LVTTL |SLEW=SLOW |DRIVE=8;
NET "led(5)" LOC="D11" |IOSTANDARD=LVTTL |SLEW=SLOW |DRIVE=8;
NET "led(4)" LOC="C11" |IOSTANDARD=LVTTL |SLEW=SLOW |DRIVE=8;
NET "led(3)" LOC="F11" |IOSTANDARD=LVTTL |SLEW=SLOW |DRIVE=8;
NET "led(2)" LOC="E11" |IOSTANDARD=LVTTL |SLEW=SLOW |DRIVE=8;
NET "led(1)" LOC="E12" |IOSTANDARD=LVTTL |SLEW=SLOW |DRIVE=8;
NET "led(0)" LOC="F12" |IOSTANDARD=LVTTL |SLEW=SLOW |DRIVE=8;
```

8.1.7. Anie eta Anie-tiny

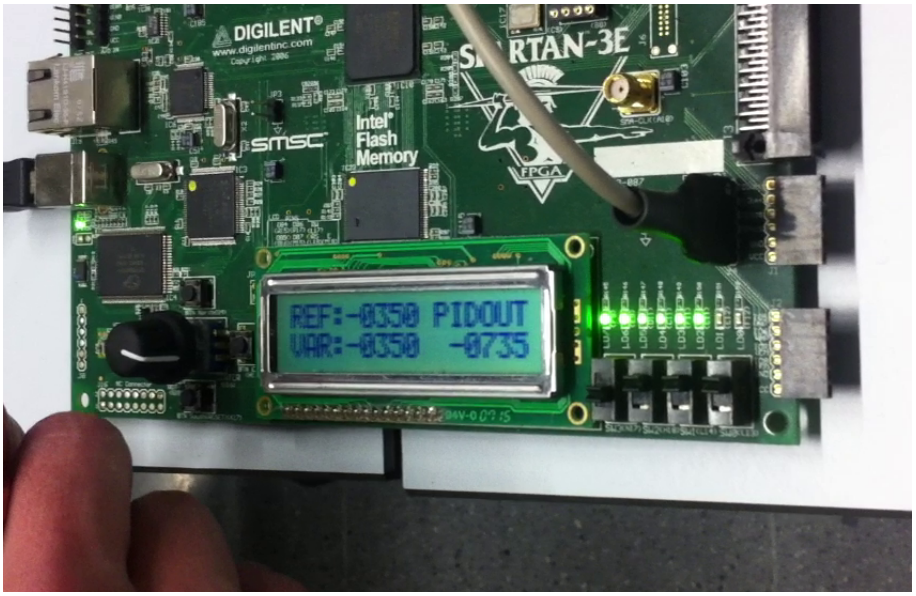
3C. edo 3E., 3D., eta 3F. eranskinetako iturriak bateratuz azaldutako bi sistemak inplementatu dira. 2-1.2. atalean bi sistemen ezaugarriak adierazten dira: exekuzio denbora, erabilitako

azalera eta baliabideak eta xahututako potentzia.

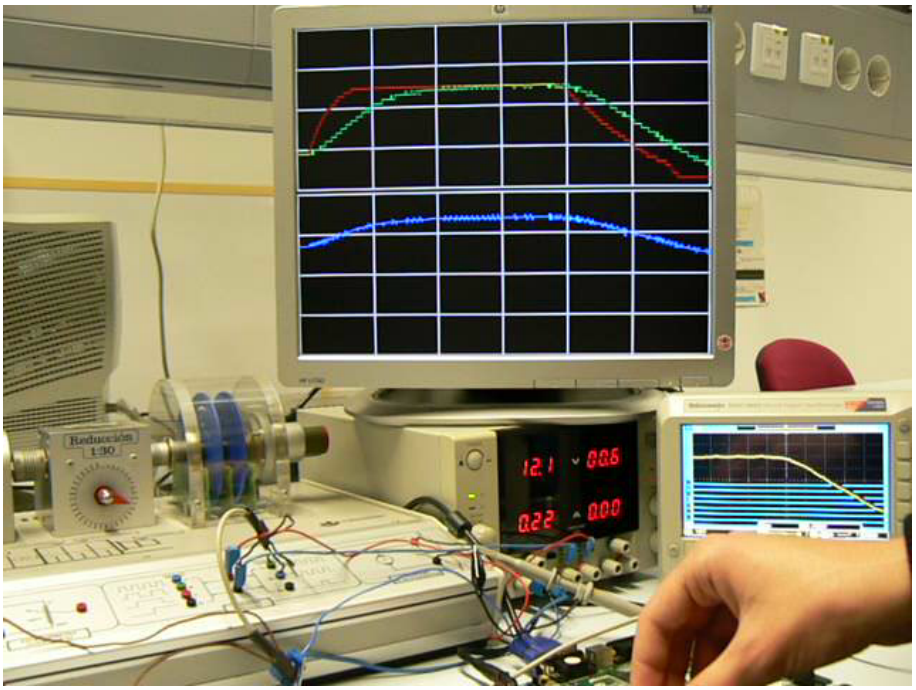
Nahiz eta *Spartan-3E* egun *Xilinx*ek eskaintzen duen FPGA familia zaharrena izan[47], garatutako diseinuek azalerari begira %12 – 21 bitartean behar dute soilik. Potentziari dagokionean 0,25 W baino gutxiago behar dute, tentsio maximoa 3,3 V-ekoa izanik.

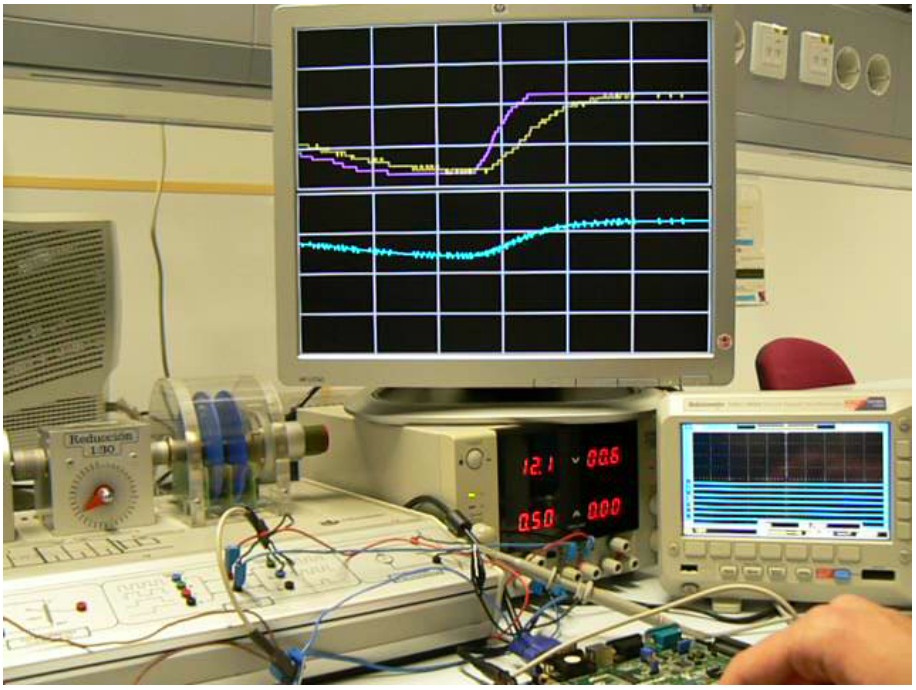
8.1.8. Abiadura-kontrola gauzatzea

Bilboko IITUEko APESek duen laborategian proiektuaren balidazioa gauzatu da. Sostengu digitalean, emandako CDan edo `vimeo.com/unaimartinezcorral` orrialdean, egindako frogen bideoa dago eskuragarri, funtzionalitate guztiak laburbilduz. Jarraian horren bideo-kapturak azaltzen dira:



[47] Logikaren garapenera zuzendutako kostu baxuko familia gisa azaltzen da.





VGAn aurkeztutako marra bertikalen arteko banaketa 100 pixelekoa dela jakinik, eta laginketa-maiztasuna 10 ms-koa denez, segundo bat adierazten dutela ondorioztatu daiteke. Grafikoetan praktikan motorrak 1,5 s inguruko egonkortze-denbora duela ikusi daiteke. 1.18. taulan adierazitako doiketaren helburuetatik gertu ez badago ere, erabilitako erreferentzia-seinalea mailakoa bainoa aldapa izan dela hartu behar da kontuan. Aldi berean, bideoak grabatzean erabilitako maketa ez da garapenean erabilitako bera izan, baizik eta berdina.

9.
Ondorioak

Proiektuaren garapenean emandako hilabetetan ikasitakoa balioestean, memorian agertzea zaila direnak dabiltza garunean bueltaka. Ezinezkoa izango litzateke hitz hauek idaztea zeharkako tresnak diren *sourceforge.net* orrialdeko *Subversion* eta *Allura* barik. Euskarriaren optimizazioa, aldiz, LaTeX eta *Texmaker*-ek ahalbidetu dute. *Dia*, *GIMP*, *Inkscape* eta *QtikZ* erabili dira irudi eta diagramak egiteko. Eta *BibTeX* izan da erreferentzia bibliografikoen kudeatzailea.

Matlab/Simulink eta *ISE/System Generator*-ekin batera, aipatutako tresna guztiak aldi berean erabili izana, eta horrek behartuta planifikazioa jarraitzeko eta baliabideak kudeatzeko prozedurak ikasi izana dira ondorio preziatuenak.

Sarreran adierazitako helburuei begira (1.1. atala), kontrolagailu jarraituaren erantzun baliokidea duen kontrol-sistema modularra, txertatuta eta autonomoa erdietsi da, *Matlab/Simulink*, *ISE/System Generator* eta VHDL lengoaia medio *Spartan3E Starter Kit* txartelean inplementatu da eta, *MV-541 maketa* azterketa kasu, kontrola gauzatu da.

Aurrezarritako metodologia jarraituz atal bakoitzaren aurretiko azterketa teorikoa burutu da, izaera anitzeko hamaika erreferentzia bibliografikotan aurkitutako laguntzari eskerrak. Bildutakoen sintesia egituratu da irakurketa zuzena errazteko eta argi banatuta ager daitezen azterketa teorikoa eta garapen praktikoa.

Kontrolagailu diskretuaren egituran eta VHDL deskribapenera zuzendutako prozeduretan arreta handiena jarri bada ere, plantaren modelo ahalik eta egiazkoena lortzeko argibideak garatu dira.

Ezaugarriei dagokionean, bi inplmentazio garatu dira: *anie-tiny* eta *anie*. Batak LCDa, lau interruptore eta lau sakagailu biltzen ditu, PWM/H-zubia eta kodetzaile inkrementala medio, korronte zuzeneko motorraren kontrola ahalbidetzeko. Bigarrenak, berriz, VGA irteerara konektatutako edozein pantailan 640x480 eta 60 Hz-eko benetako denboran berritutako grafikoa aurkezten du. Bi sakagailu gehiagok kudeatzen dute *anie*: grafikoa hobeto aztertu nahi izatekotan geldi mantentzeko eta pantailak irudiak ondo agertzen dituen ziurtatzeko hiru patroi ezberdin aurkezteko.

Normalizatuta eta norantza kontuan izanik agertzen dira bietan erreferentzia-seinalea, kontrolatu nahi den aldagaia eta kontrolagailuaren irteera. Bietan ere kontrola begizta irekian zein itxian egiteko eta kodetzailearen bereizmena (X1, X2 edo X4) aldatzeko aukera eskaintzen da.

Parametrizazioari esker doiketa eta sarrerako/irteerako osagaiak aldatzeagatik burutu beharreko moldaketak azkar egin daitezkeela ziurtatu da, eta garapenean erabilitako iturri guztiak daude eskuragarri ahal den heinean erabilera errazteko.

FPGAren $\%12-21$ azalera hartzen duten sistemak direnez, hardwarea aldatzeko beharrik gabe, sistema hazteko baliabide ugari daude erabilgarri. Hainbat PID darabiltzaten egitura konplexuagoek edo txertatutako mikrokontrolagailuek tokia dute kasu konplexuagoetara moldatzeko, adibidez. *anie-tiny* eta *anie* konparatuz, benetako denboran aldagaiak grafikatzen dituen

VGA sistemak baliabideen %10 inguru hartzen duela ondorioztatu daiteke, beste proiektuetan integratzeko aproposa izanik. Kontrolagailuak, berez, 36 flip-flop, 6 batutzaile/kentzaile, 4 konparadore eta 3 biderkatzaile baino ez ditu erabiltzen.

Potentziari dagokionean, 0,25 W baino gutxiago xahutzen dute egoerarik txarrenean, xedean autonomoak izanik baterien iraupena bermatuz. Nahiz eta kontuan hartu behar den, aldakorrak edo ezinbestekoak ez direnez, kalkulu horrek ez dituela kontuan hartzen plantari dagozkion kontsumoak, ezta monitore edo LCD pantailarenak.

Azterketa kasu izan den maketak eta aukeratutako sarrerek/irteerek erabat mugatzen dute sistemaren eraginkortasuna. Lehenengo hurrenerako egokia bada ere, ez du abiadura handiko kontrolik ahalbidetzen. Hau dela eta, ez da sistemaren ahalera osoa frogatzeko paradarik egon. Zuzena bada ere, horren planta motela kudeatzeko zeharo geldoagoak diren plataformak erabili daitezke.

9.1. Wishlist

Etengabean hobetzeko irrikak bultzatuta, aurrekoa eta beste hainbat osatzeko hau da aurkeztean autorearen etorkizunerako gurari zerrenda:

- Azterketa kasuak
  - Daudenekin egin daitezkeen ariketak osatzeko modulu berriak garatzea.
  - Kasu berriak lantzea.
- Mikrokontrolagailu eta mikroprozesadoreekin elkarlana
  - Komunikazioa
  - Hardwarea
- Autonomia
  - Erabiltzailearen kudeaketa hobetzeko moduluak hobetzea/garatzea.
  - Gailu periferiko berriak erabiltzeko moduluak garatzea.
- Dokumentazioa
  - Trebatzeko gidak, praktikak, laburpenak, etab. idaztea.
  - Formatu elektroniko dinamikoak baliatuz trebakuntzarako baliabideak sortzea.

*Pierre St. Martin*en baitan, elektronikan ingeniaritza ikasleentzat ikasgai ezberdinetan ikusitako ezagutza teoriko banatuak elkarrekin aztertu eta barneratzeko bidea nahi du izan. Behin eta berriz gurpila asmatu ordez, urte batetik bestera egindako garapenak integratuz. Bide batez, taldeen tamaina dela eta maiz behar bezala garatu gabe geratzen diren zeharkako konpetentzien jabetze-prozesua bultzatuz. Hala nola, proiektu zehatzak osotasunean egokitzea edo paraleloan

dabiltzan eta klase bereko kideak ez diren garatzaileekin, urruneko tresnak medio, talde-lana burutzea.

Azken lerroak profitatuz, nahiz eta proiektuarekin erlazio zuzena ez izan, beti laguntza adi eta zintzoa eskaini duela gogora, *Iñigo Oleagordia Aguirreri* esker onik beroenak. *Koldo*ri, bidai luze honetan karburoari eutsi dion gidariari, mirespen osoa.

BILBOKO INDUSTRIA INGENIARITZA TEKNIKOKO UNIBERTSITATE ESKOLA

INDUSTRIA INGENIARITZA TEKNIKOA: INDUSTRIA ELEKTRONIKA

KARRERA AMAIERAKO PROIEKTUA

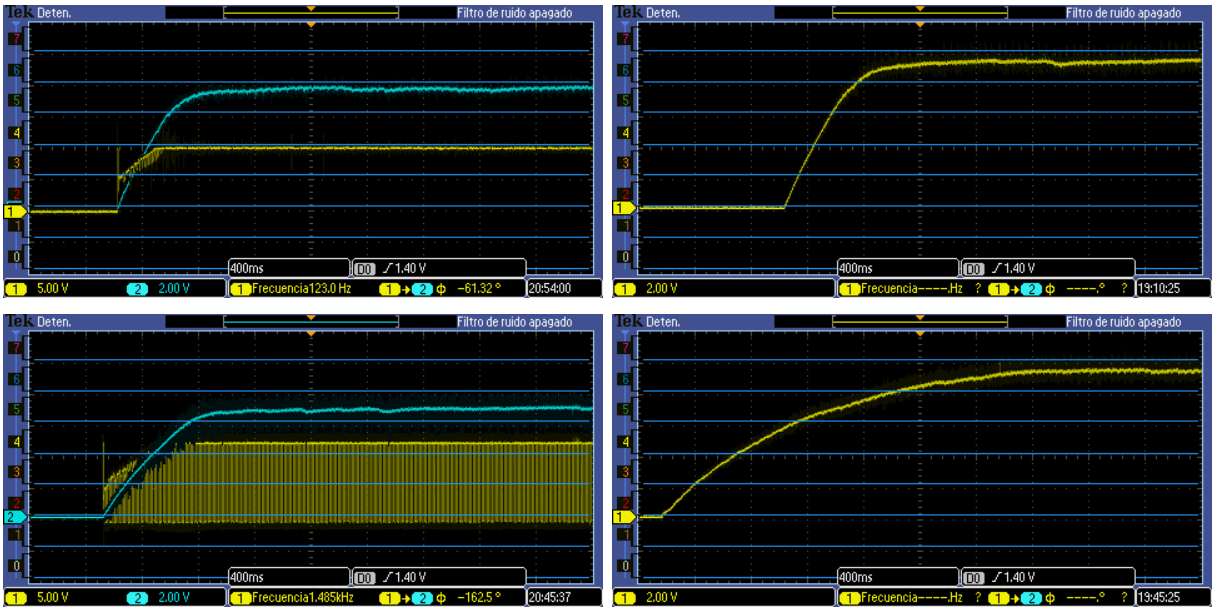
Anie - 2. Dokumentua: Neurketak eta kalkuluak

IKASLEAREN DATUAK	**ZUZENDARIAREN DATUAK**
	Koldo Basterretxea Oyarzabal koldo.basterretxea@ehu.es Teknologia Elektronikoa
SIN.: DATA: 2013ko apirilaren 12a	SIN.: DATA:

Gaien Aurkibidea

1.
Korronte zuzeneko motorra

1.1. Plantaren modelatzea eta identifikazioa



2.1. Irudia: *Identifikazioa: osziloskopio bitartez jasotako erantzunak (korronte zuzenarekin, goian, eta kontrolagailuarekin, behean).*

1.1.1. Identifikazioa burutzea

Nahiz eta kontrolerako nahikoa izan, kodetzaile inkrementalak eskainitako bereizmena ez da egokia plantaren identifikaziorako. Hau dela eta takogeneradorea erabili da plantaren irteera neurtzeko, osziloskopio batekin zuzenean irakurriz.

u_{dc}	$y_{dc\infty}$		K_{dc}	u_{pwm}	$y_{pwm\infty}$		K_{pwm}
12,0 V	9,6 V	%100	-	%100	9,4 V	%100	-
10,2 V	7,8 V	%81, 25	0,956	%85	7,6 V	%80, 85	0,951
9,6 V	7,4 V	%77, 08	0,964	%80	7,2 V	%76, 60	0,958
9,0 V	6,8 V	%70, 83	0,944	%75	6,6 V	%70, 21	0,936
8,4 V	6,2 V	%64, 58	0,923	%70	6,0 V	%63, 83	0,912
7,8 V	5,4 V	%56, 25	0,865	%65	5,2 V	%55, 32	0,851

2.1. Taula: *Identifikazioa: K (korronte zuzeneko iturria eta kontrolagailua begizta irekian).*

2.1. taulak adierazten dituen emaitzak aztertuta, H-zubia erabiltzeak 0,2 V-eko jauzia era-

giten duela ikus daiteke. Lehenengo kontrolerako berebiziko egiaztasuna behar ez denez, ordea, zuzeneko saiakuntzaren emaitza hautatu da.

Plantaren erantzuna, 1-3. atalean adierazitakoen arabera, lehenengo ordeneko esponentzial batez ordezkatzeko, funtzioaren τ parametroa askatu behar da, denbora konstantea hain zuzen ere. Azken balioaren teoremaren arabera, $\tau = \%63.2 \cdot y(\infty)$, $2\tau = \%86.4 \cdot y(\infty)$, eta $3\tau = \%95 \cdot y(\infty)$ berdinketak ezagututa 2.2. taulan bildutako saiakuntzak burutu dira.

u_{dc}	τ	2τ	3τ
10,2 V	4,93 V 300 ms	6,74 V 420 ms	7,41 V 540 ms
9,6 V	4,68 V 240 ms	6,39 V 400 ms	7,03 V 560 ms
9,0 V	4,30 V 230 ms	5,88 V 390 ms	6,46 V 560 ms
8,4 V	3,92 V 200 ms	5,36 V 360 ms	5,89 V 52 mss
7,8 V	3,41 V 160 ms	4,67 V 280 ms	5,13 V 400 ms
u_{pwm}	τ	2τ	3τ
%85	4,80 V 460 ms	6,57 V 700 ms	7,22 V 960 ms
%80	4,55 V 420 ms	6,22 V 540 ms	6,84 V 860 ms
%75	4,17 V 340 ms	5,70 V 480 ms	6,27 V 850 ms
%70	3,70 V 250 ms	5,18 V 410 ms	5,70 V 650 ms
%65	3,29 V 170 ms	4,49 V 290 ms	4,94 V 410 ms

2.2. Taula: *Identifikazioa: τ (korronte zuzeneko iturria eta kontrolagailua begizta irekian).*

1.1.2. Plantaren oinarrizko modeloa aukeratzea

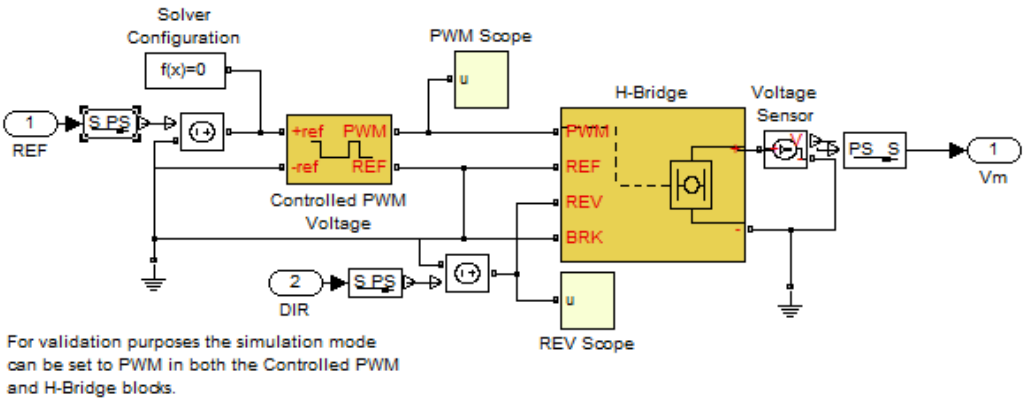
$$\alpha = \frac{\overline{3\tau - 2\tau} + \overline{2\tau - \tau} + \overline{\tau}}{3} \qquad \beta = \frac{\overline{3\tau - 2\tau} + \overline{2\tau - \tau}}{2} \qquad \gamma = \overline{\tau} - \beta \tag{2.1}$$

Aurrekoak bateratuz, bi identifikazio ezberdin burutu direnez, bost modelo aztertu dira (2.2): FOT_{dc}, FOT_{pwm}, $FOPDT_{dc}$, $FOPDT_{pwm}$ eta $DFOPDT_{pwm}$. (2.1) adierazpenak bildutako tarteko parametroek 2.1. eta 2.2. tauletako emaitzen interpretazioa azaltzen dute.

$$\frac{\overline{K}_{dc}}{\alpha_{dc}s+1} \qquad \frac{\overline{K}_{dc}}{\beta_{dc}s+1}e^{-\gamma_{dc}s} \qquad \frac{\overline{K}_{pwm}}{\alpha_{pwm}s+1} \qquad \frac{\overline{K}_{pwm}}{\beta_{pwm}s+1}e^{-\gamma_{pwm}s} \qquad z^{-\text{round}(\frac{\gamma_{pwm}}{T_s})}\frac{\frac{K_{pwm}\cdot T_s}{\beta_{pwm}}}{z - e^{\frac{-T_s}{\beta_{pwm}}}} \tag{2.2}$$

1.1.3. Sarrerako/irteerako osagaien modeloak aurkeratzea

H-zubia eta PWM modulazioaren eragina modelatzeko, *Mathworks*en adibideetan eskuragarri dauden *Controlled PWM Voltage* eta *H-Bridge* blokeak erabili dira. Darabiltzaten seinaleak *fisiko* motakoak izanik, beste elementuekin batera erabiltzeko *Simulink PS Converter* blokeak jarri dira, formatu aldaketa eginez.



2.2. Irudia: *Modeloak: PWM modulazioa eta H-zubia.*

1.1.4. Modeloen parametroak ezartzea

Hainbat saiakuntza ezberdin burutu direnez, modeloei esleitu beharreko parametroen balioak, (2.1), lortzeko batazbestekoak kalkulatu dira, 2.3. taulak biltzen duenez. Hauek (2.2)en ordezkatuz, 2.4. taulako transferentzia funtzioak lortu dira.

	$\overline{K}$	α	β	γ
dc	0,93	200 ms	145 ms	80 ms
pwm	0,922	280 ms	200 ms	110 ms

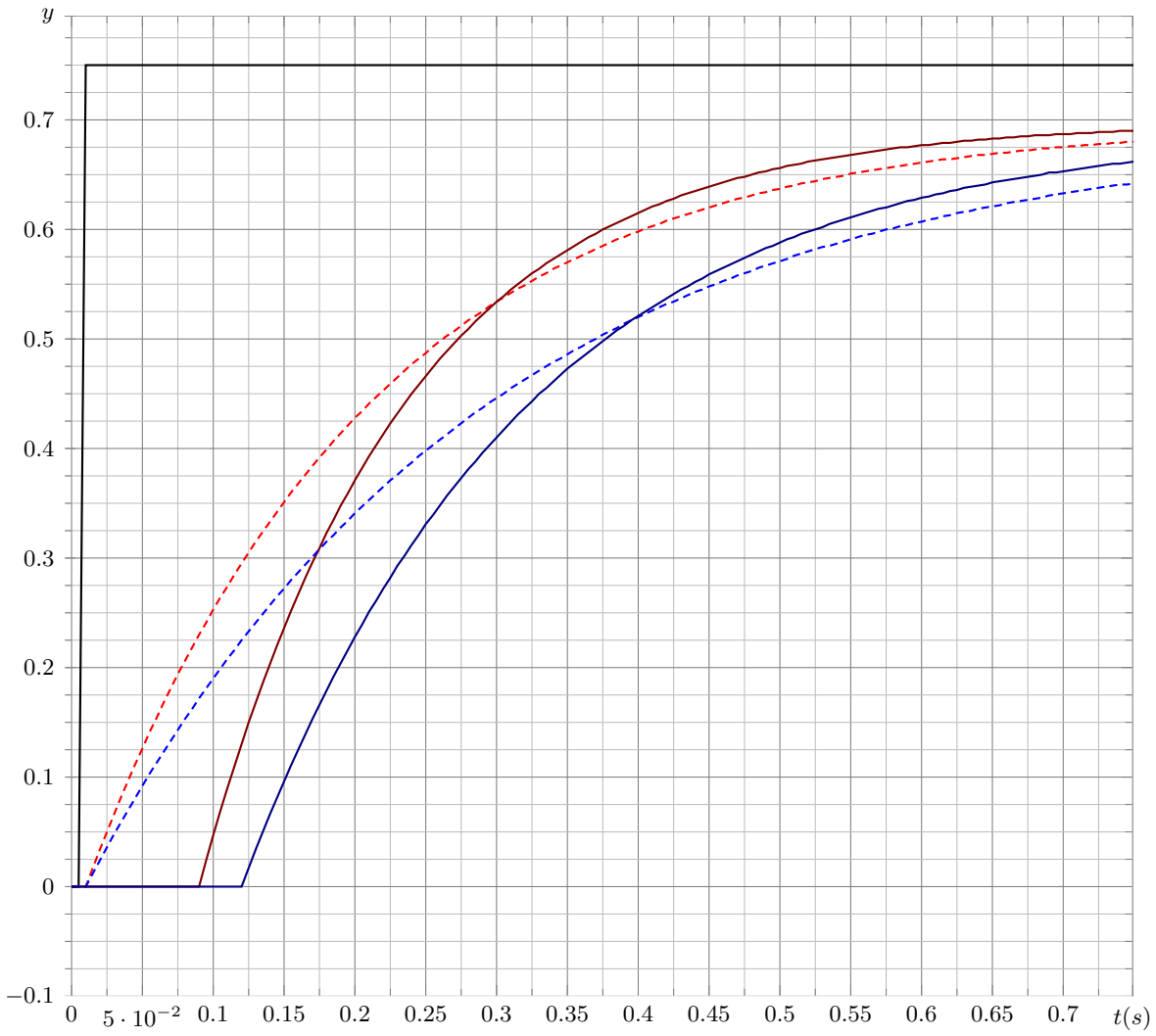
2.3. Taula: *α, β eta γ denbora konstateak: 2.1. eta 2.2. tauletako emaitzak oinarri.*

2.3. irudiak plano jarraituan adierazitako lau transferentzia funtzioen erantzunak aldaratzen ditu. Osziloskopiaren irudiei begira, FOPDT modeloak erabiltzea ondorioztatu da, hauek baitira gertukoenak.

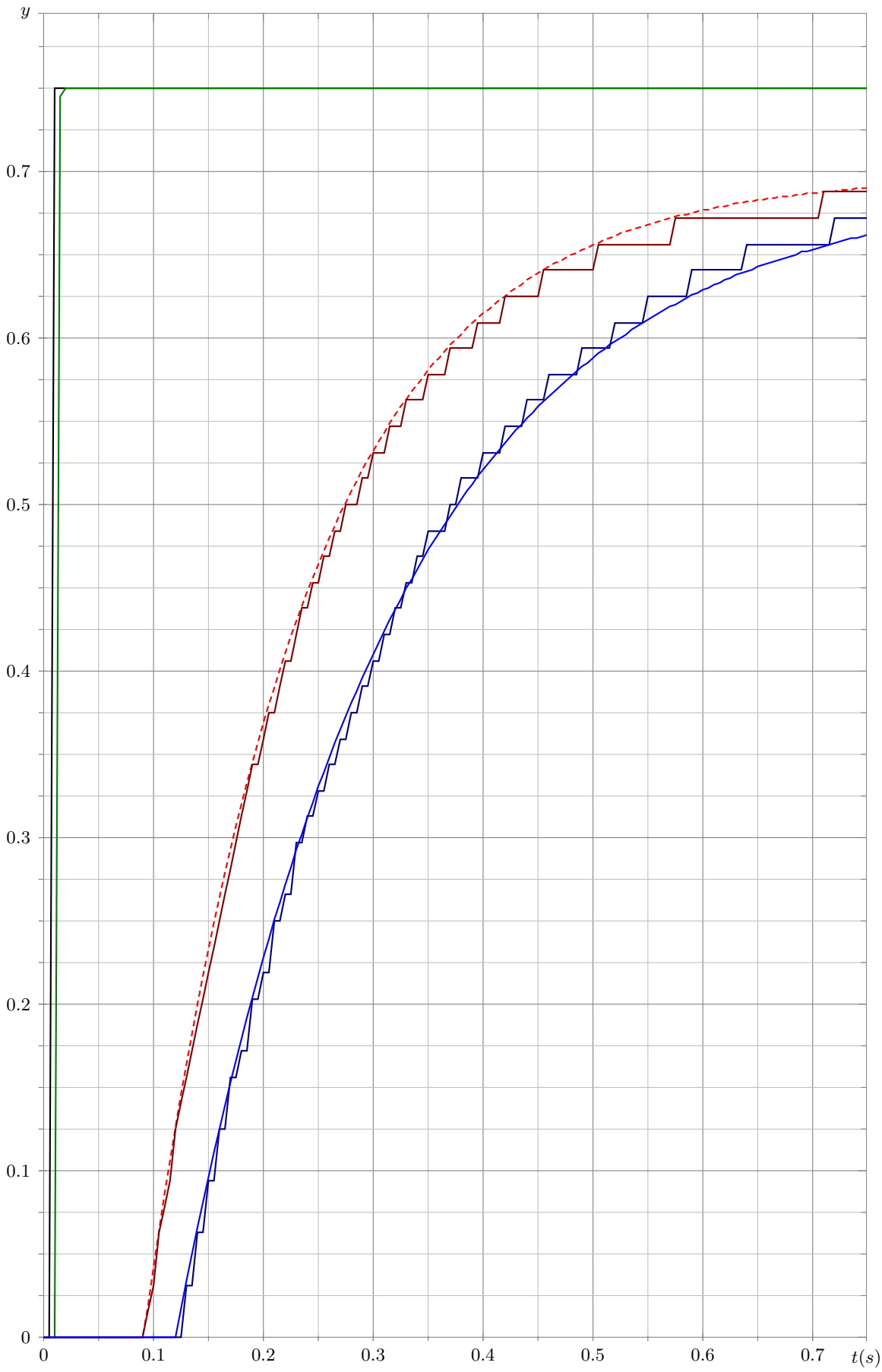
Bestelako blokerik gehitu gabe prozesu osoaren modelo gisa erabiltzeko $FOPDT_{pwm}$ edo $DFOPDT_{pwm}$ zuzenena litzateke. Aurreko azpiatalean adierazitako blokeekin, ordea $FOPDT_{dc}$ erabili da, H-zubiaren eragina bi aldiz kontuan ez hartzeko. 2.4. irudiak aukera biak konparatzen ditu. Bertan antzeman daiteke H-zubiak erreferentzia-seinalearen hedaduran eragindako atzerapena (berdez). Honek, planta osoaren banda zabalera murriztuko du, eta doiketak atal deribatiboaren ezarpena baldintzatuko du, hori baita atalik azkarrena.

FOT_{dc}	$\frac{0,93}{0,2s+1}$	
FOT_{pwm}	$\frac{0,922}{0,28s+1}$	
$FOPDT_{dc}$	$\frac{0,93}{0,145s+1}e^{-0,08s}$	
$FOPDT_{pwm}$	$\frac{0,922}{0,2s+1}e^{-0,11s}$	
$DFOPDT_{pwm}$	$\frac{\frac{0,922\cdot 0,01}{0,2}}{z-e^{\frac{-0,01}{0,2}}}z^{-\text{round}(\frac{0,11}{0,01})}$	$\frac{0,0461}{z-0,95123}z^{-11}$

2.4. Taula: *Modeloak: FOT_{dc}, FOT_{pwm}, $FOPDT_{dc}$, $FOPDT_{pwm}$ eta $DFOPDT_{pwm}$.*



2.3. Irudia: *Konparaketa: FOT_{dc}, FOT_{pwm}, $FOPDT_{dc}$, $FOPDT_{pwm}$ (begizta irekia).*



2.4. Irudia: *Konparaketa: H-zubia eta PWM modulazio bitarteko $FOPDT_{dc}$ (gorria) eta $DFOPDT_{pwm}$ (urdina).*

Hala ere, identifikazioan τ_{dc} eta τ_{pwm} denbora konstanteetan antzeman diren ezberdintasunak nabarmentzen dira: *Simulink* blokeek ez dute horrenbesteko eraginik motorraren denbora konstantean. Aldiz, doiketan egon daitezkeen arazoak antzemateko baliabide hobeak eskaintzen dituzte, eta egituraren konplexutasunaren ikuspuntutik bloke gehiago erabiltzeak parametrizazio aukera gehiago eskaintzen ditu. Hortaz, transferentzia funtzioa soilik erabiltzea alboratu da, sistema erreala erabilitako modeloa baino zertxobait geldoagoa dela jakinik[1].

1.1.5. Bereizmenaren eragina modelatzea

Maketaren kodetzaileak gehienez 40 inpultsu sortuko dituenez laginketa-periodo bakoitzeko, sei bit erabiliko dira gehienez. *Convert* blokeen bitartez *fixdt(1,7,6)* formatoa adierazi da kontrolean eragina modelatzeko. Sarrera eta irteeren balioak normalizatuta egoteagatik definitu da koma bitarra bit adierazgarrietatik ezkerra. Zazpigarren bita biko osagaian zeinua adieraztekoa da.

2.4. irudiak, aurretiaz aipatutakoen gainean, kontrolagailuak jasoko duen seinalea aurkezten du. Garapenean aurrera egin ahala bloke honen funtsa *System Generator*eko blokeek edo VHDL deskribapenean egindakoek ordezkatuko dute.

1.2. Kontrol sistemaren ezaugarriak

Target Device:	xc3s500e-4fg320
Design Goal:	Balanced
Design Strategy:	Xilinx Default (unlocked)

2.5. Taula: ISE*: sintesi eta inplementaziorako baldintzak.*

1-8.1. atalean adierazitako parametroak erabilita, eta 2.5. taulako diseinu baldintzekin, bi diseinu ezberdin inplementatu dira:

- **anie-tiny**: kontrol begiztarako ezinbestekoak direnak (VGArik ez).
- **anie**: osagai guztiak ditu.

2.6. taulan adierazitako *warning* gehienak diseinuaren modulartasunari atxikitu behar zaizkio. zenbait osagairen irteerak erabilgarriak ez izateagatik aske utzi baitira. Azkena, aldiz, PID kontrolagailuaren irteera PWM maiztasunera moldatzean egindako mozketa adierazten du.

Warnings:

- Unconnected output port 'U1_CLKIN_IBUFG_OUT' of component 'anie_dcm'.

[1] PWM seinalearen maiztasunak, laginketa-maiztasuna eta beste hainbat faktore balioetsi ahala diferentzia murriztuko da.

Parsers Errors:	No Errors
Warnings:	11 *anie-tiny* , 8 *anie*
Routing Results:	All Signals Completely Routed
Timing Constraints:	All Constraints Met

2.6. Taula: ISE*: sintesi eta inplementazioaren txostenen ondorioak.*

- Unconnected output port 'U2_CLK0_OUT' of component 'anie_dcm'.
- Unconnected output port 'count' of component 'anie_counter' (3).
- Unconnected output port 'tc' of component 'anie_counter' (2).
- Unconnected output port 'sout' of component 'anie_bcdshift'.
- Dangling pin *DOA3* on block: RAMB16_RAMB16A (2).
- Signal <out_sat<3:0>> is assigned but never used. This unconnected signal will be trimmed during the optimization process.

1.2.1. Kontrol begiztaren egikaritze denbora

Deskribatutako sistemaren lan-maiztasun maximoa 50 MHz ingurukoa da[2], 2.7. taulak adierazi bezala. *anie-tiny* zein *anie* sintesiek abiadura maximoa izanik, izaera modularra medio, VGA kontrolagailuak sistema geldotzen ez duela ondorioztatu deaiteke. Hala ere, txartelak duen erlojua 50 MHz-eko maiztasuna eskaintzen du, eta sistema ez da abiadura maximoan arituko.

Speed Grade:	−4	
Minimum period:	19,410 ns	
Maximum Frequency:	51,520 MHz	
Min. input arrival time before clock:	20,314 ns	19,267 ns
Max. output required time after clock:	40,600 ns	
Max. combinational path delay:	14,622 ns	14,632 ns
	anie-tiny	*anie*

2.7. Taula: ISE*: Timing Summary.*

Kontrol begiztari dagokionean, laginketaren *trigger* seinaleak maila altua hartzen duenetik normalizazio osagaiak baldintzatzen du. Barneko *FSM*ak $4-6$ ziklo behar ditu aldagaia eta erreferentzia irakurri, kontrolagailuaren sarrerak berritu eta konputatutako irteera berritzeko. 5 MHz-eko erloju seinaleak agintzen duenez, egoerarik txarrenean $\frac{1}{5\,\text{MHz}} \cdot 6 = 1{,}2\,\mu\text{s}$ behar ditu sistemak erantzuteko.

[2] *Design Goal* gisa *Timing* ezarriz gero 80 MHz arteko maiztasun maximoak erdietsi daitezke.

Irteera PWM bitartez eragiten denez, ziklo oso bat bukatzea ezinbestekoa da aldaketa benetakotzat jotzeko. Honetan ere erlojua 5 MHz-ekoa da eta 10*bit*eko bereizmena izanik $\frac{1}{5\,\mathrm{MHz}} \cdot 2^{10} = 204{,}8\,\mu$sko periodoa du.

Hortaz, bi faktoreak kontuan hartuz, 206 µs behar dira kontrol begizta egikaritzeko. Benetako denbora ziurtatzeko eta kontrolagailuaren erantzuna jarraituaren baliokidea dela onartzeko egikaritze denbora $\frac{T_s}{100}$ izan behar da. Aukeratutako balioek ez dute baldintza hori betetzen, $\frac{10\,\mathrm{ms}}{206\,\mu\mathrm{s}} \approx 50$. Berezimenari bit bat kentzearekin nahikoa litzateke, baina kontrolagailuaren aldaketak nabarmentzeko nahiago izan da bereizmena mantentzea.

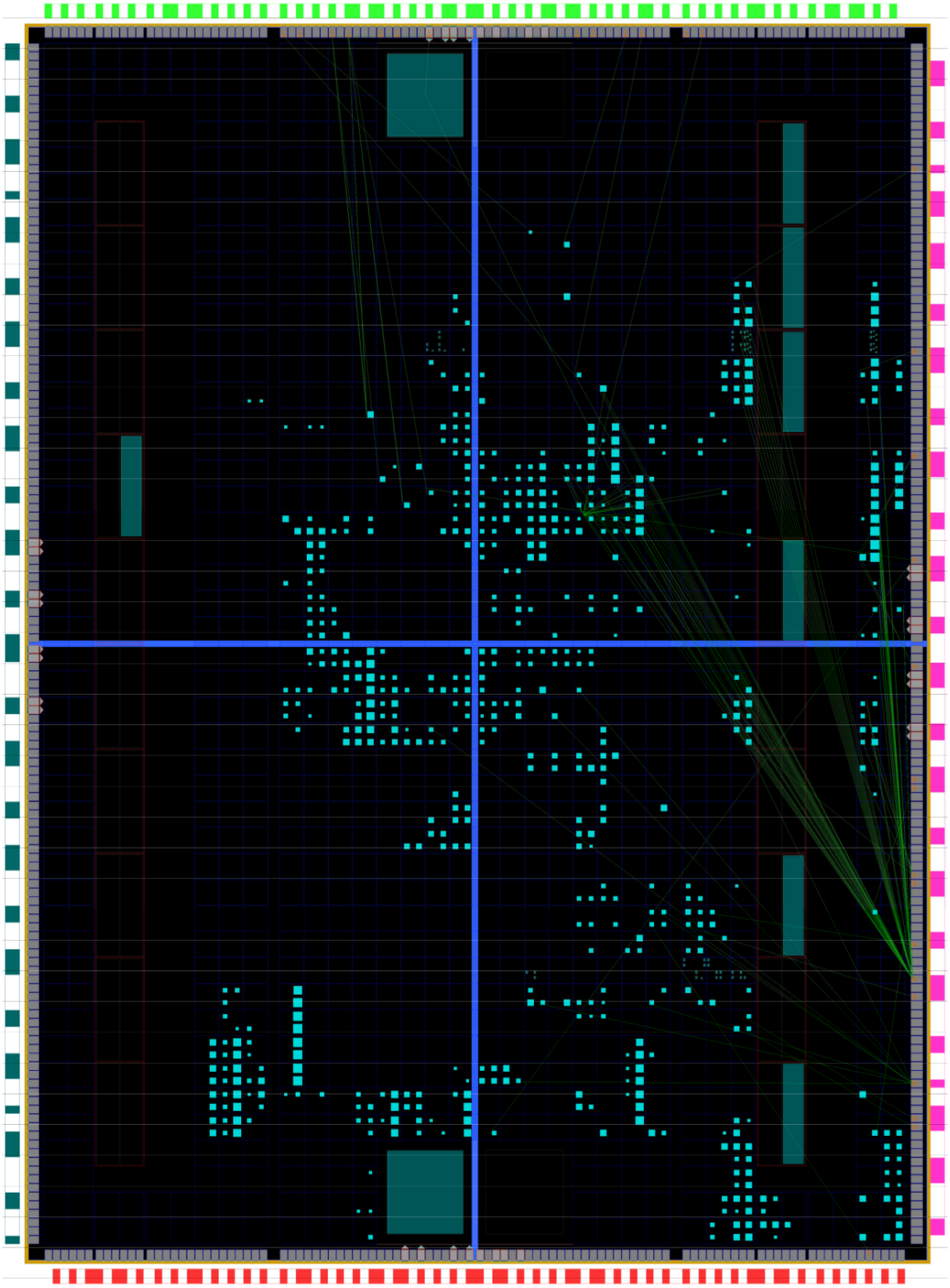
1.2.2. Erabilitako azalera

	anie-tiny		*anie*		
Logic Utilization	**Used**		**Used**		**Available**
Number of Slice Flip Flops	412	4%	583	6%	9,312
Number of 4 input LUTs	938	10%	1,529	16%	9,312
Number of occupied Slices	624	13%	1,009	21%	4,656
Total Number of 4 input LUTs	1,004	10%	1,665	17%	9,312
Number used as logic	903		1,492		
Number used as a route-thru	66		136		
Number used for Dual Port RAMs	32		32		
Number used as Shift registers	3		5		
Number of bonded IOBs	29	12%	36	15%	232
Number of RAMB16s			4	20%	20
Number of BUFGMUXs	7	29%	9	37%	24
Number of DCMs	2	50%	2	50%	4
Number of MULT18X18SIOs	7	35%	10	50%	20
Average Fanout of Non-Clock Nets	3.24		3.21		

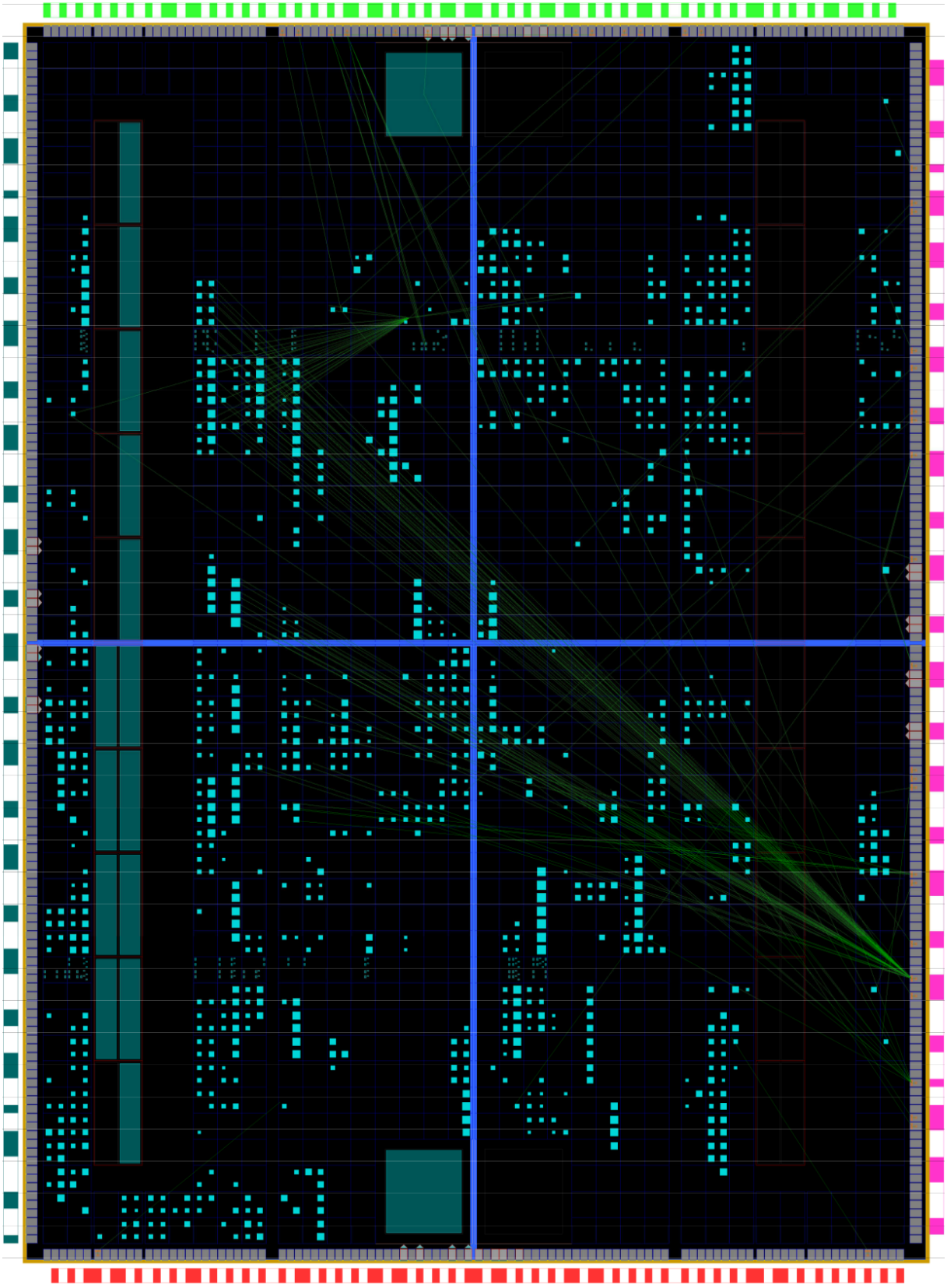
2.8. Taula: ISE*: Device Utilization Summary.*

Sintesiek %12 eta %21 azalera erabilera emaitzak eman dituzte *anie-tiny* eta *anie*rentzan, hurrenez hurren. 2.8. taulak erabilera logikoaren arabera azaltzen ditu baliabideen zenbakia eta ehunekoa.

2.5. eta 2.6. irudiek inplementazioen mapak agertzen dituzte. Bertan argi antzeman daiteke erabilitako azalera. Sintetizatutako gailu eta osagaien zerrenda zehatzagoa 3G. eranskineko txostenetan aurkitu daiteke.



2.5. Irudia: *Inplementazioa:* anie-tiny *diseinuaren FPGAren erabilera.*



2.6. Irudia: *Inplementazioa:* anie *diseinuaren FPGAren erabilera.*

1.2.3. Xahututako potentzia

*Xilinx*ek web-gunean[3] *Microsoft Excel 2003 (.xls)* formatuan banatutako *Xilinx Power Estimator (XPE)*[4] tresna eskaintzen du. Inplementazioek erabilitako baliabideen arabera gutxi gora beherako potentzia xahutzea kalkulatzea ahalbidetzen du, hardwarearen diseinuan zer nolako aireztapena behar den jakiteko xedez. Helburu berberarekin, *ISE* inguruneak *Place & Route* atalean *Xilinx XPower Analyzer (XPA)* tresna dauka, inplementatutako proiektutik datuak zuzenean hartzen dituena.

*Philofsky*k [43] azaldutako argibideak jarraituz, *ISE* ingurunean sintetizatu eta inplementatu ostean *.mrp* formatuan emandako *Map* txostenak erabili dira *XPE* tresnan eta *XPA* ere abiatu da. Biek oso antzeko emaitzak eman dituzte, *XPA*ek xehetasun gehiago izanik gutxi gora behera 10 mW gutxiagoko kontsumo osoa emanez.

Proiektu honetan produktu komertziala den txartela erabili denez, eta ez zaionez inolako aireztapen sistemarik ezta xahutzailerik gehitu, 0 LFM adierazi dira. Egoerarik txarrenean kontsumoa aztertzeko, 50 °C-eko giro tenperatura adierazi da eta bi tresnen emaitzetatik balio altuenak hartu dira. Honek jarraian adierazitako maximoak eman ditu *anie-tiny* eta *anie* inplementazioentzat:

Device

Part:	XC3S500E
Package:	FG320
Grade:	Commercial
Process:	Maximum
Speed Grade:	−4

Block Summary

Clock	0,009 W	0,011 W
Logic	0,003 W	0,005 W
IO	0,010 W	0,014 W
BRAM	-	0,004 W
DCM	0,034 W	
MULT	0,006 W	0,008 W

Thermal Information

Ambient Temp:	50 °C	
Airflow:	0 LFM	
ΘJA:	25,9 $\frac{°C}{W}$	
Max Ambient:	79,4 °C	79,0 °C
Junction Temp:	55,6 °C	56,0 °C
	anie-tiny	*anie*

Power Summary

Optimization:	None	
Data:	Production	
Quiescent:	0,155 W	
Dynamic:	0,063 W	0,075 W
Total	0,217 W	0,231 W
	anie-tiny	*anie*

Voltage Source Information

				ICC		ICCQ
VCCAUX	2,5 V	0,110 W		0,013 A		0,031 A
VCCINT	1,2 V	0,094 W	0,104 W	0,017 A	0,025 A	0,062 A
VCCO 3.3	3,3 V	0,013 W	0,016 W	0,003 A	0,004 A	0,001 A
		anie-tiny	*anie*	*anie-tiny*	*anie*	

[3] xilinx.com/products/design_tools/logic_design/xpe.htm
[4] Version: 11.1 ; Release Date: Apr 8, 2009.

FPGA familiaren ezaugarri orrietan begiratuz gero [36], erdietsitako balioak adierazitako mugen barnean daudela ikus daiteke, eta koherenteak direla.

Atal honetan azaldutako emaitzak Xilinx*en* ISE *inguruneko tresnen bitartez eskuratu dira: inplementazioan sortutako txostenak zein* PlanAhead Floorplanner *edo* Xilinx XPower Analyzer*. Formatuaren mugak direla eta, laburpena baino ez da azaltzen hemen. Sostengu digitalean txosten osoak aurkitu daitezke. Xehetasun txikiagoak aztertzeko eranskinetan azaldutako iturriak baliatuz* ISE *ingurunean proiektua eginez gero, tresna guztiek bereizmen osoko informazioa emango dute.*

2.
Garapenerako zerbitzariaren estatistikak

4A. eranskinean azaldutako proiektu osoaren aurrekarietan aipatutakoek adierazten dutenez, sei garapenetatik lautan software garapenerako bertsio-kontrol sistema erabili izan da:

sourceforge.net : ZTPK, OHKIS eta ohkis-gtk

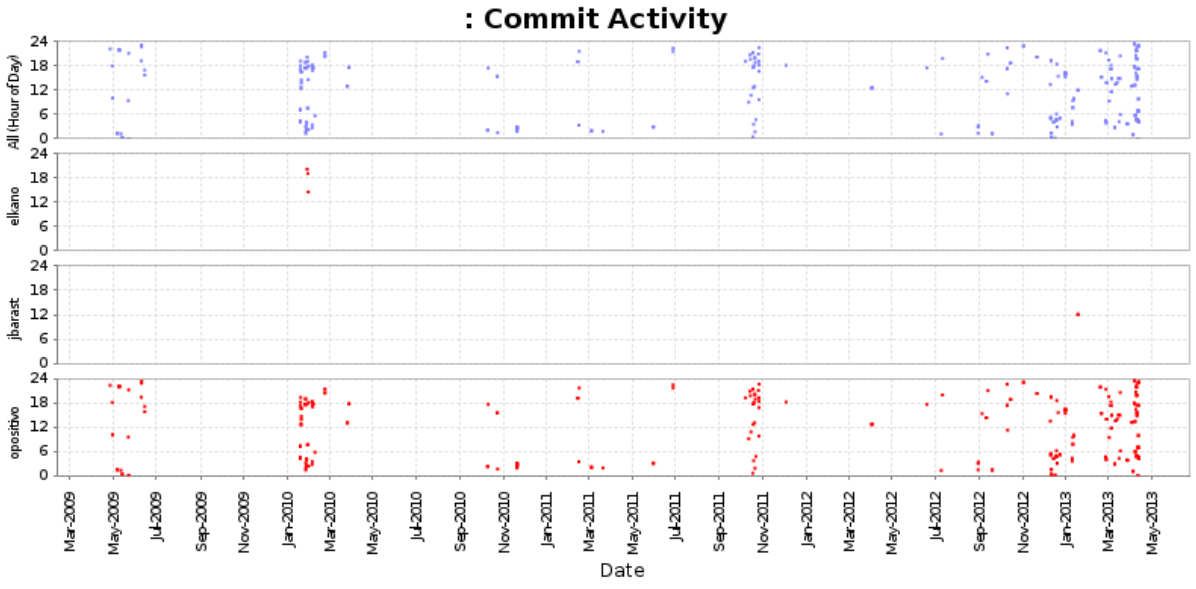
laforja.rediris.es : BμC

lokala : Acher

Lehenengo bietan, bertsio-kontrolaz gain, beste hainbat baliabide daude eskuragarri, hala nola, baimen ezberdineko hainbat erabiltzaileen kudeatzailea, web-gunea, banaketa zerrenda, erroreen konponketa eskaera eta jardueren jarraitzailea (*tracker*a), edota foroak. *laforja.rediris.es* plataforma *sourceforge.net*en oinarrituta egoteagatik bigarrena aukeratu da *Anie*ren garapenerako. Proiektu honek *OHKIS*en hainbat atal dituenez, eta *ohkis-gtk* ere zerbitzari eta proiektu berdinean gordeta dagoenez, eskaera berria egin ordez `ohkis.sourceforge.net` erabili da.

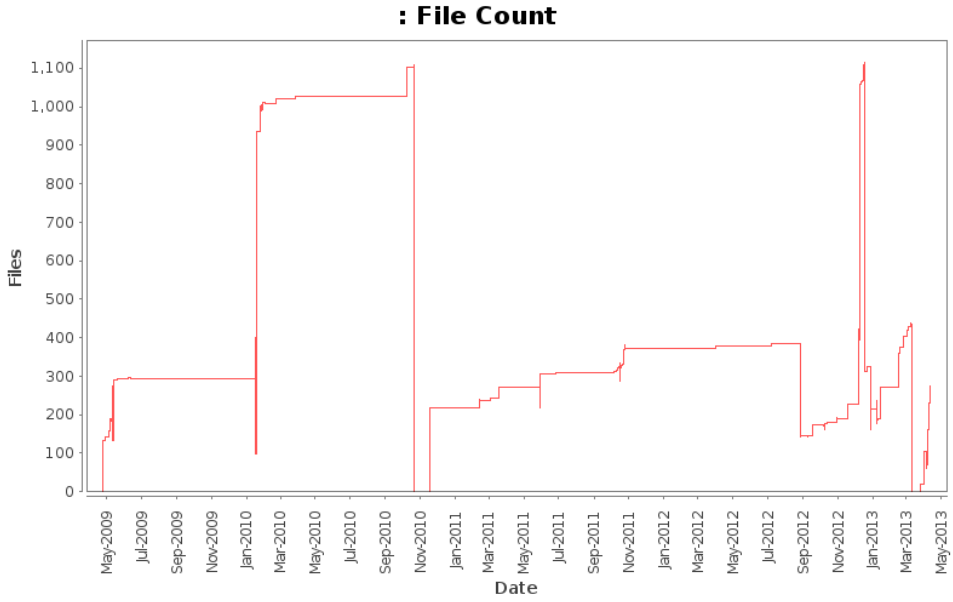
Honetan, hiru erabiltzailek hartu dute parte guztira. *elkano*k eta *jbarast*ek, soilik eta hurrenez hurren, *ohkis-gtk* eta *auto* proiektuak landu dituztenez, txosten honetan ez dira horiei eta soilik horiei dagozkienak adierazi[5]. Jarraian orokorrak eta autoreari, *opositivo* erabiltzaileari, dagozkionak aurkezten[6] dira:

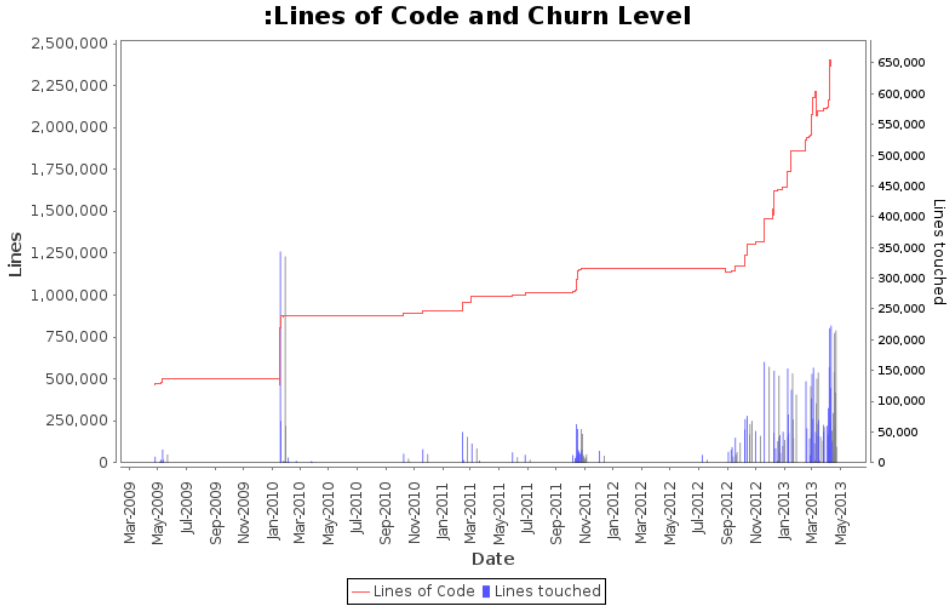
2.1. Subversion

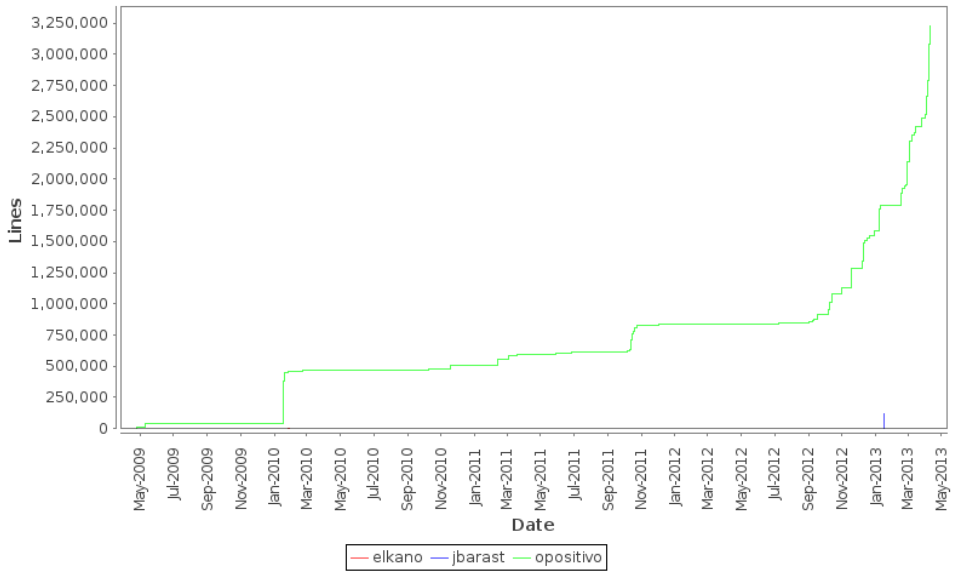


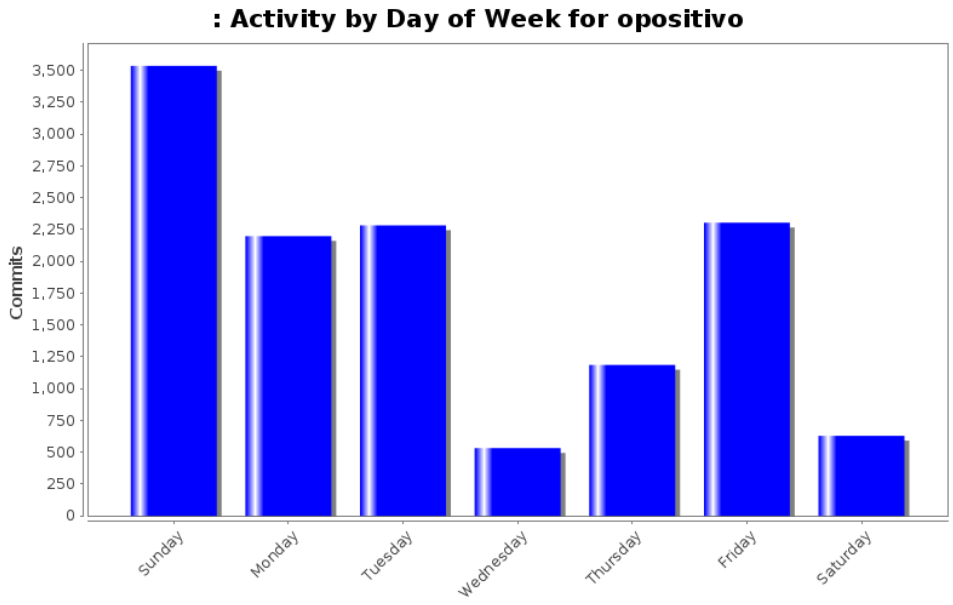
[5]Estatistika osoak ikusteko, jo proiektuaren zerbitzariko erregistro irekietara. Tresna zehatzak: *statsvn*, *gource*, *allura*...

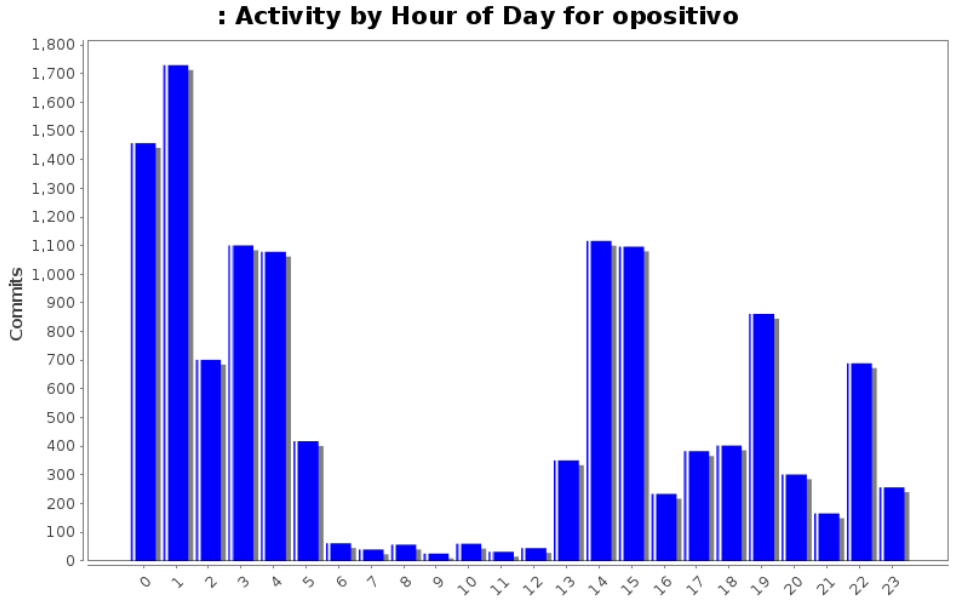
[6]2009 urteko jarduerak *OHKIS* proiektuari dagozkio, 2010eko irailera bitartekoak *ohkis-gtk* atalari, eta ordutik gaurrerakoak *Anie*ri.

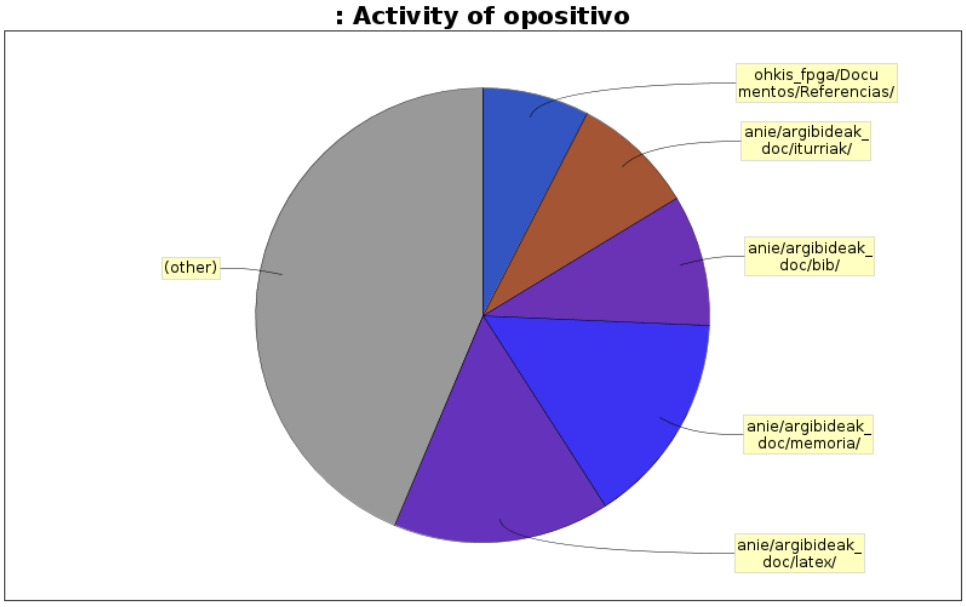

: File Count
Files
Date


:Lines of Code and Churn Level
Lines
Lines touched
Date
Lines of Code
Lines touched


: Contributed Lines of Code
Lines
Date
elkano
jbarast
opositivo


: Activity by Day of Week for opositivo
Commits
3,500
3,250
3,000
2,750
2,500
2,250
2,000
1,750
1,500
1,250
1,000
750
500
250
0
Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday


: Activity by Hour of Day for opositivo
Commits
1,800
1,700
1,600
1,500
1,400
1,300
1,200
1,100
1,000
900
800
700
600
500
400
300
200
100
0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23


: Activity of opositivo
ohkis_fpga/Documentos/Referencias/
anie/argibideak_doc/iturriak/
anie/argibideak_doc/bib/
anie/argibideak_doc/memoria/
anie/argibideak_doc/latex/
(other)

2.2. Allura tracker

 # tickets: 33

 # open tickets: 13

 # closed tickets: 20

# of comments on tickets: 62

	Summary	Due Date	Progress
1.0	IIT, Elektronika (KAP)	2013/04/05	6/6
doc	Dokumentazioa (KAP)	2013/04/07	11/11
pre	Aurkezpena	2013/04/20	2/5
r1.0	Lehenengo ateratzea	2013/04/09	1/1
1.5	Wishlist	2013/07/31	0/8
2.0	hard-soft co-design	2013/12/31	0/1
3.0	communication	2014/05/31	0/1

#	Summary	Status	Updated
	1.0		
5	VGA	closed	2012/12/16
14	Openloop	closed	2012/12/16
18	DIR eta REF	closed	2012/12/20
19	TRIGGER	closed	2012/12/21
21	Loop execution time	closed	2013/04/01
24	Identifikazioa	closed	2013/04/01
	doc		
1	Indice de simbolos	closed	2013/02/24
3	Revisar anexos	closed	2012/04/11
6	co-simulation results	closed	2012/12/24
10	Citas	closed	2013/01/06
13	Brushless	closed	2013/01/06
23	textbf, textit, emph	closed	2013/03/16
25	Ondorioak	closed	2013/04/05
28	Aurrekontua	closed	2013/04/10
30	Lizentzien irudiak	closed	2013/04/05
31	titlesubsection ordez, subsection soilik	closed	2013/04/06
33	Euskalterm Deuna	closed	2013/04/11
	pre		
2	Bosquejos	closed	2013/03/16
4	Aurkezpena	open	2012/12/12
8	Jueguetes	closed	2013/02/24
17	Beca	open	2012/12/17
20	Erabiltzaile gida	open	2012/12/25
	r1.0		
16	Lizentzia	closed	2013/04/01

	r1.5		
7	Implementation comparison	pending	2013/04/07
12	Hardware in the loop	pending	2013/04/07
15	Iturriak komentatzea	pending	2013/04/07
22	Slope	open	2013/02/07
26	Doiketak eta Ts LCD bitartez aldatzea	open	2013/04/04
27	int8, int16	open	2013/04/04
29	VGA/LCD normalizazioa X arabera	open	2013/04/05
32	Osagaiak paketatzea	pending	2013/04/07
	2.0		
11	Añelarra	open	2012/12/16
	3.0		
9	Larra	open	2012/12/12

BILBOKO INDUSTRIA INGENIARITZA TEKNIKOKO UNIBERTSITATE ESKOLA

INDUSTRIA INGENIARITZA TEKNIKOA: INDUSTRIA ELEKTRONIKA

KARRERA AMAIERAKO PROIEKTUA

Anie - 3. Dokumentua: Eranskinak [kodea]

IKASLEAREN DATUAK	**ZUZENDARIAREN DATUAK**
	Koldo Basterretxea Oyarzabal koldo.basterretxea@ehu.es Teknologia Elektronikoa
SIN.: DATA: 2013ko apirilaren 12a	SIN.: DATA:

Gaien Aurkibidea

3A.

Matlab: .m *script*ak

tune.m

```
P=6;
I=0.125;
D=0.25;
Tf=.01;
Ts=0.011;
sat=12;
sTs=Ts/3;
```

save_bw.m

```
% sTs -> Scope Sample Time
% Min Simulation Time sTs*samples
% bw.dat

clear;
tune;

sTs=Ts/3;
samples = [325,300];
freq_range = [0.01,15];
names={'y' 'u' 'mag' 'deg' '.dat'};
models={'cont','discrete','contdfilt'};

labels=strcat('freq');
in = frest.Sinestream('Frequency',linspace(freq_range(1),freq_range(2),samples
    (2)),'FreqUnits','Hz');

towrite=0;

for n=1:3
  mdl=char(strcat('PID_',models(n)));

  sysest=0; simout=0; mag=0; phase=0; freq=0;

  [sysest,simout] = frestimate(mdl,getlinio(mdl),in);
  [mag,phase,freq]=bode(sysest);

  for m=1:2
    labels=strcat(labels,'\t',names(2+m),'_',models(n));
  end

  towrite(1:samples(2),(2*n),1)=20*log10(mag(1,1,1:samples(2)));
  towrite(1:samples(2),(2*n)+1,1)=phase(1,1,1:samples(2));
end

```

```
towrite(1:samples(2),1)=freq(1:samples(2))/(2*pi);

filetowrite = fopen(char(strcat('bw',names(5))),'wt');
fprintf(filetowrite,char(labels));
fclose(filetowrite);
dlmwrite(char(strcat('bw',names(5))),towrite,'-append','delimiter','\t','
    precision','%.3f','roffset',1);
```

save_ts.m

```
% sTs -> Scope Sample Time
% Min Simulation Time sTs*samples
% ts.dat

clear;
tune;

sTs=.01;
mdl='PID_discrete';
names={'y' 'u' 'cont'};
samples = [300];
Tp=1.1;
mul=[2,5,10,30,50,100];

labels=strcat('t','\t','ref');
for m=1:2
  labels=strcat(labels,'\t',names(m),'_',names(3));
end

towrite=0;

for n=1:6
  Ts=Tp/mul(n);

  y=0; u=0; data=0;

  sim(mdl);
  data.y=y;
  data.u=u;

  for m=1:2
    labels=strcat(labels,'\t',names(m),'_',sprintf('%i',mul(n)));
  end

  towrite(1:samples,2*n+3)=data.y.signals.values(1:samples,2);
  towrite(1:samples,2*n+4)=data.u.signals.values(1:samples);
end

towrite(1:samples,1)=data.y.time(1:samples);
towrite(1:samples,2)=data.y.signals.values(1:samples);

```

```
mdl=char(strcat('PID_',names(3)));

y=0; u=0; data=0;

sim(mdl);
data.y=y;
data.u=u;
towrite(1:samples,3)=data.y.signals.values(1:samples,2);
towrite(1:samples,4)=data.u.signals.values(1:samples);

filetowrite = fopen(char('ts.dat'),'wt');
fprintf(filetowrite,char(labels));
fclose(filetowrite);
dlmwrite(char('ts.dat'),towrite,'-append','delimiter','\t','precision','%.3f','
    roffset',1);
```

save_step.m

```
% sTs -> Scope Sample Time
% Min Simulation Time sTs*samples
% step.dat

clear;
tune;

sTs=.005;
samples = [300];
names={'y' 'u' '.dat'};
models={'cont','discrete','contdfilt','fp'};

labels=strcat('t','\t','ref');
towrite=0;
for n=1:4
  mdl=char(strcat('PID_',models(n)));
  y=0; u=0; data=0;
  sim(mdl);
  data.y=y;
  data.u=u;

  for m=1:2
    labels=strcat(labels,'\t',names(m),'_',models(n));
  end

  towrite(1:samples(1),(2*(n+1))-1)=data.y.signals.values(1:samples(1),2);
  towrite(1:samples(1),2*(n+1))=data.u.signals.values(1:samples(1));
end

towrite(1:samples(1),1)=data.y.time(1:samples(1));
towrite(1:samples(1),2)=data.y.signals.values(1:samples(1));

filetowrite = fopen(char(strcat('step',names(3))),'wt');
```

```
fprintf(filetowrite,char(labels));
fclose(filetowrite);
dlmwrite(char(strcat('step',names(3))),towrite,'-append','delimiter','\t','
    precision','%.3f','roffset',1);
```

save_branch.m

```
% sTs -> Scope Sample Time
% Min Simulation Time sTs*samples
% adar.dat

clear;
tune;

mdl='PID_comp';
sTs=.005;
minmax=200;
samples=[400];
names={'pro','int','der','.dat'};
models={'cont','discrete','contdfilt'};

y=0; u=0; data=0;
sim(mdl);
data.pro=pro;
data.int=int;
data.der=der;

labels={};
towrite=0;
for n=1:3
  labels=strcat(labels,'t_',names(n),'\t');
  towrite(1:samples,((4*n)-3))=data.pro.time(1:samples);
  for m=1:3
    labels=strcat(labels,names(n),'_',models(m),'\t');
    towrite(1:samples,((4*n)-(3-m)))=data.(char(names(n))).signals.values(1:
        samples,m);
  end
end

towrite = max(towrite,-minmax);
towrite = min(towrite,minmax);

filetowrite = fopen(char(strcat('branch',names(4))),'wt');
fprintf(filetowrite,char(labels));
fclose(filetowrite);
dlmwrite(char(strcat('branch',names(4))),towrite,'-append','delimiter','\t','
    precision','%.4f','roffset',1);
```

save_cont.m

```
% sTs -> Scope Sample Time
```

```
% Min Simulation Time sTs*samples(3)
% ref.dat, step_cont.dat, bw_cont.dat

sTs=.01
samples = [3,256,300];
freq_range = [0.01,15];
names={'y' 'u' 'mag' '.dat'};
models={'cont','discrete','ii','backward','tustin','tustinbackward','contfilt'};

mdl = 'PID_cont';
sim(mdl);
data.y=y_cont;
data.u=u_cont;

%ref
towrite=0;
towrite(1:samples(1),1)=data.y.time(1:samples(1));
towrite(samples(1)+1,1)=data.y.time(samples(2));
towrite(1:samples(1),2)=data.y.signals.values(1:samples(1));
towrite(samples(1)+1,2)=data.y.signals.values(samples(2));
dlmwrite(char(strcat('ref',names(4))),towrite,'delimiter','\t','precision','%.3f
    ','roffset',1);

towrite=0;
%step_cont
towrite(1:samples(2),1)=data.y.signals.values(1:samples(2),2);
towrite(1:samples(2),2)=data.u.signals.values(1:samples(2));

labels=strcat(names(1),'_',models(1));
labels=strcat(labels,'\t',names(2),'_',models(1));
filetowrite = fopen(char(strcat('step_cont',names(4))),'wt');
fprintf(filetowrite,char(labels));
fclose(filetowrite);
dlmwrite(char(strcat('step_cont',names(4))),towrite,'-append','delimiter','\t','
    precision','%.3f','roffset',1);

in = frest.Sinestream('Frequency',linspace(freq_range(1),freq_range(2),samples
    (3)),'FreqUnits','Hz');
[sysest,simout] = frestimate(mdl,getlinio(mdl),in);

towrite=0;
%bw_cont
[mag,phase,freq]=bode(sysest);
towrite(1:samples(3),1)=freq(1:samples(3))/(2*pi);
towrite(1:samples(3),2,1)=20*log10(mag(1,1,1:samples(3)));
towrite(1:samples(3),3,1)=phase(1,1,1:samples(3));

labels=strcat('freq');
labels=strcat(labels,'\t',names(3),'_',models(1));
labels=strcat(labels,'\t','deg','_',models(1));
filetowrite = fopen(char(strcat('bw_cont',names(4))),'wt');
```

```
fprintf(filetowrite,char(labels));
fclose(filetowrite);
dlmwrite(char(strcat('bw_cont',names(4))),towrite,'-append','delimiter','\t','
    precision','%.3f','roffset',1);
```

save_box.m

```
% sTs -> Scope Sample Time
% Min Simulation Time sTs*samples
% box.dat

clear;
tune;

sTs=.005;
samples = 300;
names={'y' 'u'};
models={'box','discrete','contdfilt'};

labels=strcat('t','\t','ref');
towrite=0;

y=0; u=0; data=0;
sim('PID_box');
data.y=y;
data.u=u;

for n=1:3
  for m=1:2
    labels=strcat(labels,'\t',names(m),'_',models(n));
  end

  towrite(1:samples,(2*(n+1))-1)=data.y.signals.values(1:samples,n+1);
  towrite(1:samples,2*(n+1))=data.u.signals.values(1:samples,n);
end

towrite(1:samples,1)=data.y.time(1:samples);
towrite(1:samples,2)=data.y.signals.values(1:samples,1);

filetowrite = fopen('box.dat','wt');
fprintf(filetowrite,char(labels));
fclose(filetowrite);
dlmwrite('box.dat',towrite,'-append','delimiter','\t','precision','%.3f','
    roffset',1);
```

3B.

Ko-simulaziorako *Blackbox* blokea: deskribapena eta konfigurazio fitxategia

box_pid.vhd

```
1  library ieee;
2   use ieee.std_logic_1164.ALL;
3   use ieee.numeric_std.all;
4
5  entity anie_pid is
6   generic ( i_wl: natural:=12;   o_wl: natural:=22;
7             k_wl: natural:=8;    irem_wl: natural:=1;
8
9             pbp: integer:=2;     ibp: integer:=17;
10            dbp: integer:=2;     minbp: integer:=2 );
11  port ( m_clk,m_ce: in std_logic; --
12 --       srst: in  std_logic;
13         mi_ref: in std_logic_vector(i_wl-1 downto 0); --
14         mi_feed: in std_logic_vector(i_wl-1 downto 0); --
15         mkp: in std_logic_vector(k_wl-1 downto 0); --
16         mki: in std_logic_vector(k_wl-1 downto 0); --
17         mkd: in std_logic_vector(k_wl-1 downto 0); --
18         mo: out std_logic_vector(o_wl-1 downto 0) ); --
19 end anie_pid;
20
21 architecture pid of anie_pid is
22
23 component anie_sat
24  generic (iref_wl: natural:=2; osat_wl: natural:=2);
25  port (iref: in signed(iref_wl-1 downto 0); osat: out signed(osat_wl-1 downto 0)
        );
26 end component;
27
28  constant aswl: natural:= i_wl+1;
29  constant mwl: natural:= i_wl+k_wl;
30  constant oswl: natural:= mwl+irem_wl+1;
31
32  signal ref,feed: signed(i_wl-1 downto 0):=(others=>'0'); --
33  signal dif_act,dif_pre: signed(aswl-1 downto 0):=(others=>'0'); --
34  signal der_dif,int_sum: signed(aswl downto 0):=(others=>'0'); --
35  signal pro_out: signed(mwl-1 downto 0):=(others=>'0'); --
36  signal int_mult,der_out: signed(mwl downto 0):=(others=>'0'); --
37  signal int_out,int_rem,out_sat: signed(oswl-1 downto 0):=(others=>'0'); --
38  signal int_tosat,out_tosat: signed(oswl downto 0):=(others=>'0'); --
39
40  signal i_ref: signed(i_wl-1 downto 0):=(others=>'0'); --+
41  signal i_feed: signed(i_wl-1 downto 0):=(others=>'0'); --+
42  signal kp: signed(k_wl-1 downto 0):=(others=>'0'); --+
```

```
43  signal kd: signed(k_wl-1 downto 0):=(others=>'0'); --+
44  signal ki: signed(k_wl-1 downto 0):=(others=>'0'); --+
45  signal o: signed(o_wl-1 downto 0):=(others=>'0'); --+
46
47 begin
48
49 i_ref <= signed(mi_ref); --+
50 i_feed <= signed(mi_feed); --+
51 kp <= signed(mkp); --+
52 kd <= signed(mkd); --+
53 ki <= signed(mki); --+
54 mo <= std_logic_vector(o); --+
55
56 ref <= i_ref; --+
57 feed <= i_feed; --+
58
59 dif_act <= resize(ref,aswl) - resize(feed,aswl);
60
61 int_sum <= resize(dif_act,aswl+1) + resize(dif_pre,aswl+1);
62 der_dif <= resize(dif_act,aswl+1) - resize(dif_pre,aswl+1);
63
64 pro_out <= shift_right(resize(kp*dif_act,mwl),pbp-minbp);
65 int_mult <= shift_right(resize(ki*int_sum,mwl+1),ibp-minbp);
66 der_out <= shift_right(resize(kd*der_dif,mwl+1),dbp-minbp);
67
68 int_tosat <= resize(int_mult,oswl+1) + resize(int_rem,oswl+1);
69
70 out_tosat <= resize(pro_out,oswl+1) +
71              resize(int_out,oswl+1) +
72              resize(der_out,oswl+1);
73
74 o <= out_sat(oswl-1 downto oswl-o_wl);
75
76 INTSAT: anie_sat
77   generic map (oswl+1,oswl)
78   port map (int_tosat,int_out);
79
80 OUTSAT: anie_sat
81   generic map (oswl+1,oswl)
82   port map (out_tosat,out_sat);
83
84 process(m_clk,m_ce) --
85 begin
86 if rising_edge(m_clk) then --
87 -- if srst='1' then
88 --  ref <= (others => '0'); feed <= (others => '0');
89 --  dif_pre <= (others => '0'); int_rem <= (others => '0');
90 -- elsif m_ce='1' then
91  if m_ce='1' then --+
92 --  ref <= i_ref; feed <= i_feed;
93   dif_pre <= dif_act;
```

```
94    int_rem <= int_out;
95   end if;
96  end if;
97  end process;
98
99  end pid;
```

box_sat.vhd

```
1  library ieee;
2   use ieee.std_logic_1164.ALL;
3   use ieee.numeric_std.all;
4
5  entity anie_sat is
6   generic ( iref_wl: natural:=2; osat_wl: natural:=2 );
7   port ( iref: in signed(iref_wl-1 downto 0); osat: out signed(osat_wl-1 downto
        0) );
8  end anie_sat;
9
10 architecture sat of anie_sat is
11
12   signal max,min: signed(osat_wl-1 downto 0);
13
14 begin
15
16 max(osat_wl-1) <= '0';  max(osat_wl-2 downto 0) <= (others => '1');
17 min(osat_wl-1) <= '1';  min(osat_wl-2 downto 0) <= (0 => '1', others => '0');
18
19 osat <= max when (iref>resize(max,iref_wl)) else
20         min when (iref<resize(min,iref_wl)) else
21         resize(iref,osat_wl);
22
23 end sat;
```

box_pid_config.m

```
1  function anie_pid_config(this_block)
2
3    this_block.setTopLevelLanguage('VHDL');
4    this_block.setEntityName('anie_pid');
5
6    this_block.tagAsCombinational;
7
8    this_block.addSimulinkInport('mi_ref');
9    this_block.addSimulinkInport('mi_feed');
10   this_block.addSimulinkInport('mkp');
11   this_block.addSimulinkInport('mki');
12   this_block.addSimulinkInport('mkd');
13
14   this_block.addSimulinkOutport('mo');
15   mo_port = this_block.port('mo');
```

```
16   mo_port.setType('UFix_22_0');
17
18   if (this_block.inputRatesKnown)
19     setup_as_single_rate(this_block,'m_clk','m_ce')
20   end  % if(inputRatesKnown)
21
22   uniqueInputRates = unique(this_block.getInputRates);
23
24   this_block.addGeneric('i_wl','natural','12');
25   this_block.addGeneric('o_wl','natural','22');
26   this_block.addGeneric('k_wl','natural','8');
27   this_block.addGeneric('irem_wl','natural','1');
28   this_block.addGeneric('pbp','integer','2');
29   this_block.addGeneric('ibp','integer','17');
30   this_block.addGeneric('dbp','integer','2');
31   this_block.addGeneric('minbp','integer','2');
32
33   this_block.addFile('anie_boxsat.vhd');
34   this_block.addFile('anie_boxpid.vhd');
35
36 return;
37
38 % ------------------------------------------------------------
39
40 function setup_as_single_rate(block,clkname,cename)
41   inputRates = block.inputRates;
42   uniqueInputRates = unique(inputRates);
43   if (length(uniqueInputRates)==1 & uniqueInputRates(1)==Inf)
44     block.addError('The␣inputs␣to␣this␣block␣cannot␣all␣be␣constant.');
45     return;
46   end
47   if (uniqueInputRates(end) == Inf)
48      hasConstantInput = true;
49      uniqueInputRates = uniqueInputRates(1:end-1);
50   end
51   if (length(uniqueInputRates) ~= 1)
52     block.addError('The␣inputs␣to␣this␣block␣must␣run␣at␣a␣single␣rate.');
53     return;
54   end
55   theInputRate = uniqueInputRates(1);
56   for i = 1:block.numSimulinkOutports
57      block.outport(i).setRate(theInputRate);
58   end
59   block.addClkCEPair(clkname,cename,theInputRate);
60   return;
```

s3etiny_timing.vhd

```
1  entity anie_timing is
2   port ( clk,rst: in std_logic;
3          lrst,clk_adq,clk_pwm,clk_hb: out std_logic;
4          clk10,clk160,clk12_5k,clk0_1k: out std_logic );
5  end anie_timing;
6
7  architecture timing of anie_timing is
8
9  component anie_dcm
10  port( U1_CLKIN_IN: IN std_logic;
11        U1_RST_IN: IN std_logic;
12        U1_CLKDV_OUT: OUT std_logic;
13        U1_CLKIN_IBUFG_OUT: OUT std_logic;
14        U1_CLK0_OUT: OUT std_logic;
15        U2_CLKDV_OUT: OUT std_logic;
16        U2_CLK0_OUT: OUT std_logic;
17        U2_LOCKED_OUT: OUT std_logic );
18 end component;
19
20 component anie_counter
21  generic (wl: natural:=3);
22  port ( clk,ce,srst: in std_logic;
23         limit: in std_logic_vector(wl-1 downto 0);
24         count: out std_logic_vector(wl-1 downto 0);
25         tc: out std_logic);
26 end component;
27
28 constant dv25l:std_logic_vector(4 downto 0):=std_logic_vector(to_unsigned(24,5))
       ;
29 constant dv125l:std_logic_vector(6 downto 0):=std_logic_vector(to_unsigned
       (124,7));
30
31  signal u2locked,lock,mrst: std_logic;
32  signal iclk10,iclk160,iclk12_5k,iclk0_1k: std_logic;
33
34 attribute clock_signal : string;
35 attribute clock_signal of iclk12_5k: signal is "yes";
36 attribute clock_signal of iclk0_1k: signal is "yes";
37
38 attribute buffer_type: string;
39 attribute buffer_type of iclk12_5k: signal is "bufg";
40 attribute buffer_type of iclk0_1k: signal is "bufg";
41
42 begin
43
```

```
44 mrst <= rst or (not u2locked);
45
46 PDCM: anie_dcm
47 port map ( U1_CLKIN_IN => clk,
48            U1_RST_IN => rst,
49            U1_CLKDV_OUT => iclk10, --5 MHz
50            U1_CLKIN_IBUFG_OUT => open,
51            U1_CLK0_OUT => open,
52            U2_CLKDV_OUT => iclk160, --312,5 KHz
53            U2_CLK0_OUT => open,
54            U2_LOCKED_OUT => u2locked );
55
56 DV25: anie_counter generic map ( 5 )
57 port map ( iclk160,'1',mrst,dv25l,open,iclk12_5k );
58
59 DV125: anie_counter generic map ( 7 )
60 port map ( iclk160,iclk12_5k,mrst,dv125l,open,iclk0_1k );
61
62 clk10 <= iclk10;
63 clk160 <= iclk160;
64 clk12_5k <= iclk12_5k;
65 clk0_1k <= iclk0_1k;
66
67 clk_adq <= iclk0_1k;
68 clk_pwm <= iclk10;
69 clk_hb <= iclk0_1k;
70
71 process(iclk0_1k)
72 begin
73 if rising_edge(iclk0_1k) then lock <= u2locked; end if;
74 end process;
75 lrst <= rst or (lock nand u2locked);
76
77 end timing;
```

s3etiny_counter.vhd

```
1 entity anie_counter is
2  generic ( wl: natural:=3 );
3  port ( clk,ce,srst: in std_logic;
4         limit: in std_logic_vector(wl-1 downto 0);
5         count: out std_logic_vector(wl-1 downto 0);
6         tc: out std_logic );
7 end anie_counter;
8
9 architecture counter of anie_counter is
10
11  signal ca, cn: unsigned(wl-1 downto 0);
12
13 begin
14
```

```
15 process (clk,srst,ce)
16 begin
17 if rising_edge(clk) then
18  if srst='1' then ca <= (others => '0');
19  elsif ce='1' then ca <= cn; end if;
20  end if;
21 end if;
22 end process;
23
24 cn <= (others => '0') when ca=unsigned(limit) else ca +1;
25 count <= std_logic_vector(ca);
26 tc <= '1' when ca=0 else '0';
27
28 end counter;
```

s3etiny_man.vhd

```
1  entity anie_man is
2   generic ( ref_wl: natural:=3 );
3   port ( clk10,clk160,clk12_5k,clk0_1k: in std_logic;
4          srst,ref_dir: in std_logic;
5          rot: in std_logic_vector(1 downto 0);
6          btn: in std_logic_vector(2 downto 0);
7          lmode: out std_logic;
8          LCD_E,LCD_RS,LCD_RW: out std_logic;
9          SF_D: out std_logic_vector(3 downto 0);
10         encsel: out unsigned(1 downto 0);
11         ref_man: out signed(ref_wl-1 downto 0);
12         lcdr,lcdv,lcdo: in signed(13 downto 0) );
13 end anie_man;
14
15 architecture man of anie_man is
16
17 component anie_rot is
18  port ( clk: in std_logic;
19         rot: in std_logic_vector(1 downto 0);
20         rot_e,rot_d: out std_logic );
21 end component;
22
23 component anie_udcounter
24  generic (wl: natural:=1);
25  port ( clk,srst,ce,dir: in std_logic;
26         count: out std_logic_vector(wl-1 downto 0));
27 end component;
28
29 component anie_lcd
30  port ( clk10,clk160,srst,load: in std_logic;
31         addr: in std_logic_vector(4 downto 0);
32         data: in std_logic_vector(7 downto 0);
33         LCD_E, LCD_RS, LCD_RW: out std_logic;
34         SF_D: out std_logic_vector(3 downto 0) );
```

```
35 end component;
36
37 component anie_lcdman
38  generic ( ref_wl: natural:=3 );
39  port ( clk,srst: in std_logic;
40         load: out std_logic;
41         lcdr,lcdv,lcdo: in signed(13 downto 0);
42         addr: out std_logic_vector(4 downto 0);
43         data: out std_logic_vector(7 downto 0) );
44 end component;
45
46 component anie_lcdmenu
47  port ( clk,srst: in std_logic;
48         rot_e,rot_d,btnsel,btntg,btnmode: in std_logic;
49         inmenu,load,mode: out std_logic;
50         encsel: out unsigned(1 downto 0);
51         addr: out std_logic_vector(4 downto 0);
52         data: out std_logic_vector(7 downto 0) );
53 end component;
54
55 component anie_btndeb
56  port ( clk: in std_logic;
57         btn: in std_logic_vector(2 downto 0);
58         btndeb: out std_logic_vector(2 downto 0) );
59 end component;
60
61 component anie_switch
62  port ( clk,ce,srst: in std_logic;
63         sw: out std_logic);
64 end component;
65
66 --REF
67  signal rce,rot_e,rot_d: std_logic;
68  signal std_ref: std_logic_vector(ref_wl-2 downto 0);
69
70 --LCD
71  signal load,loadman,loadmenu,lcdman_rst,inmenu: std_logic;
72  signal addr,addrman,addrmenu: std_logic_vector(4 downto 0);
73  signal data,dataman,datamenu: std_logic_vector(7 downto 0);
74  signal enc_int: unsigned(1 downto 0);
75
76 --DEB
77  signal btndeb: std_logic_vector(2 downto 0);
78
79 begin
80
81 --REF
82 ROTF: anie_rot port map ( clk10,rot,rot_e,rot_d );
83 REF: anie_udcounter generic map ( ref_wl-1 )
84 port map ( clk10,srst,rce,rot_d,std_ref );
85 rce <= '0' when ((unsigned(std_ref)>39 and rot_d='0') or
```

```
                (unsigned(std_ref)=0 and rot_d='1') or inmenu='1') else rot_e;
ref_man <= signed('0'&std_ref) when ref_dir='0' else signed(not ('0'&std_ref))
    +1;

--LCD
LCD: anie_lcd
port map ( clk10,clk160,srst,load,addr,data,LCD_E,LCD_RS,LCD_RW,SF_D );

LCDMAN: anie_lcdman generic map( ref_wl )
port map ( clk12_5k,lcdman_rst,loadman,lcdr,lcdv,lcdo,addrman,dataman );
lcdman_rst <= srst or inmenu;

LCDMENU: anie_lcdmenu
port map ( clk10, srst, rot_e, rot_d, btndeb(2), btndeb(1), btndeb(0),
           inmenu, loadmenu, lmode, enc_int, addrmenu, datamenu );
encsel <= 3-enc_int;

load <= loadmenu when inmenu='1' else loadman;
addr <= addrmenu when inmenu='1' else addrman;
data <= datamenu when inmenu='1' else dataman;

--DEB
DEB: anie_btndeb port map ( clk0_1k, btn, btndeb );

end man;
```

s3etiny_rot.vhd

```
entity anie_rot is
 port ( clk: in std_logic;
        rot: in std_logic_vector(1 downto 0);
        rot_e,rot_d: out std_logic );
end anie_rot;

architecture rot of anie_rot is

 signal rot_int: std_logic_vector(1 downto 0);
 signal delay_rot,dir: std_logic;

begin

rot_filter: process(clk)
begin
if rising_edge(clk) then
 case rot is
  when "00" => rot_int(0) <= '0';          rot_int(1) <= rot_int(1);
  when "01" => rot_int(0) <= rot_int(0);   rot_int(1) <= '0';
  when "10" => rot_int(0) <= rot_int(0);   rot_int(1) <= '1';
  when "11" => rot_int(0) <= '1';          rot_int(1) <= rot_int(1);
  when others => rot_int(0) <= rot_int(0); rot_int(1) <= rot_int(1);
 end case;
```

```
end if;
end process;

rot_event: process(clk)
begin
if rising_edge(clk) then
 delay_rot <= rot_int(0);
 if rot_int(0)='1' and delay_rot='0' then
      rot_e <= '1'; dir <= rot_int(1);
 else rot_e <= '0'; dir <= dir; end if;
end if;
end process;

rot_d <= dir;

end rot;
```

s3etiny_udcounter.vhd

```
entity anie_udcounter is
 generic (wl: natural:=1);
 port ( clk,srst,ce,dir: in std_logic;
        count: out std_logic_vector(wl-1 downto 0));
end anie_udcounter;

architecture updown_counter of anie_udcounter is

 signal count_a,count_n: unsigned(wl-1 downto 0);

begin

process (clk)
begin
if rising_edge(clk) then
 if srst='1' then count_a <= (others => '0');
 elsif ce='1' then count_a <= count_n; end if;
end if;
end process;

count_n <= count_a +1 when dir='0' else count_a -1;
count <= std_logic_vector(count_a);

end updown_counter;
```

s3etiny_lcdman.vhd

```
entity anie_lcdman is
 generic ( ref_wl: natural:=3 );
 port ( clk,srst: in std_logic;
        load: out std_logic;
        lcdr,lcdv,lcdo: in signed(13 downto 0);
```

```
6         addr: out std_logic_vector(4 downto 0);
7         data: out std_logic_vector(7 downto 0) );
8 end anie_lcdman;
9
10 architecture lcdman of anie_lcdman is
11
12  constant sref: unsigned(2 downto 0):="001";
13  constant svar: unsigned(2 downto 0):="010";
14  constant sout: unsigned(2 downto 0):="011";
15  constant r: unsigned(4 downto 0):=(others=>'0');
16
17 component anie_bintobcd
18  port ( clk,srst: in  std_logic;
19         bin: in std_logic_vector(13 downto 0);
20         bcd: out std_logic_vector(15 downto 0) );
21 end component;
22
23  signal sel_sig: signed(13 downto 0);
24  signal sel_abs: std_logic_vector(13 downto 0);
25  signal sel_bcd: std_logic_vector(15 downto 0);
26  signal sign_bcd: std_logic_vector(7 downto 0);
27  signal iaddr,addrwh,addrbcd,addrbcd_r,addrfix,addrfixr: unsigned(4 downto 0);
28  signal idata,data_bcd,data_fix: std_logic_vector(7 downto 0);
29  signal lg: std_logic_vector(47 downto 0);
30
31  signal ca,cn: unsigned(4 downto 0);
32  signal st_rst: std_logic:='0';
33
34  signal sela,seln: unsigned(2 downto 0);
35  signal lgtlim: std_logic_vector(2 downto 0);
36
37  type st_type is (init,wr_lg,wr_wh,clear,pre_pid,do_load,convert,wr_sig);
38  signal sta, stn: st_type;
39
40 begin
41
42 with seln select
43 sel_sig <= lcdr when sref, lcdv when svar, lcdo when sout,
44            (others=>'0') when others;
45 sel_abs <= std_logic_vector(abs(sel_sig));
46
47 BINTOBCD: anie_bintobcd port map ( clk,st_rst,sel_abs,sel_bcd );
48 st_rst <= '1' when stn=pre_pid else '0';
49
50 with seln select
51 lgtlim <= "100" when sref|svar, "110" when sout, (others=>'0') when others;
52
53 process(clk,srst)
54 begin
55 if rising_edge(clk) then
56  if srst='1' then sta <= init; sela <= (others=>'0'); ca <= (others=>'0');
```

```
 else sta <= stn;  sela <= seln; ca <= cn;
 end if;
end if;
end process;

process(sta,sela,srst,lgtlim,ca)
begin
case sta is
 when init =>
  if srst='1' then stn <= init; seln <= sela;
  else seln <= sela+1;
   if sela=sout then stn <= wr_wh; else stn <= wr_lg; end if;
  end if;
  cn <= (others=>'0');

 when wr_lg =>
  if ca=unsigned(std_logic_vector(lgtlim)&'0') then
   stn <= init; cn <= (others=>'0');
  else stn <= wr_lg; cn <= ca+1; end if;
  seln <= sela;

 when wr_wh =>
  if ca="00101" then stn <= clear; cn <= (others =>'0');
  else stn <= wr_wh; cn <= ca+1; end if;
  seln <= sela;

 when clear => stn <= pre_pid; seln <= (others=>'0'); cn <= (others=>'0');

 when pre_pid =>
  if sela=sout then seln <= sref;
  else seln <= sela+1; end if;
  stn <= do_load; cn <= (others=>'0');

 when do_load =>
  if ca=2 then stn <= convert; cn <= (others=>'0');
  else stn <= do_load; cn <= ca+1; end if;
  seln <= sela;

 when convert =>
  if ca=30 then stn <= wr_sig; cn <= (others=>'0');
  else stn <= convert; cn <= ca+1; end if;
  seln <= sela;

 when wr_sig =>
  if ca="01001" then stn <= pre_pid; cn <= (others=>'0');
  else stn <= wr_sig; cn <= ca+1; end if;
  seln <= sela;

 when others => stn <= init; cn <= (others => '0'); seln <= (others=>'0');
end case;
end process;
```

```
108
109 with sta select load <= ca(0) when wr_lg|wr_wh|wr_sig, '0' when others;
110 sign_bcd <= "00101101" when sel_sig(13)='1' else "11111110";
111
112 with ca(3 downto 1) select
113 addrwh <= "01001" when "000", "11001" when "001",
114           "11010" when "010", (others=>'0') when others;
115
116 with seln select
117 addrbcd_r <= r+4 when sref, resize(r+20,5) when svar, resize(r+27,5) when sout,
118              (others=>'0') when others;
119 addrbcd <= addrbcd_r+unsigned(ca(3 downto 1));
120
121 with ca(3 downto 1) select
122 data_bcd <= sign_bcd when "000",
123             "0011"&sel_bcd(15 downto 12) when "001",
124             "0011"&sel_bcd(11 downto 8) when "010",
125             "0011"&sel_bcd(7 downto 4) when "011",
126             "0011"&sel_bcd(3 downto 0) when "100", (others=>'0') when others;
127
128 with seln select
129 addrfixr <= r when sref, resize(r+16,5) when svar, resize(r+10,5) when sout,
130             (others=>'0') when others;
131 addrfix <= addrfixr+unsigned(ca(3 downto 1));
132
133 with ca(3 downto 1) select
134 data_fix <= lg(7 downto 0) when "000",   lg(15 downto 8) when "001",
135             lg(23 downto 16) when "010", lg(31 downto 24) when "011",
136             lg(39 downto 32) when "100", lg(47 downto 40) when "101",
137             (others=>'0') when others;
138
139 with seln select
140 lg(23 downto 0) <= "01000110"&"01000101"&"01010010" when sref, --FER -> REF
141                    "01010010"&"01000001"&"01010110" when svar, --RAV -> VAR
142                    "01000100"&"01001001"&"01010000" when sout, --DIP -> PID
143                    (others=>'0') when others;
144
145 with seln select
146 lg(31 downto 24) <= "00111010" when sref|svar, "01001111" when sout,
147                      (others=>'0') when others; -- : O
148 lg(47 downto 32) <= "01010100"&"01010101"; --TU -> UT
149
150 with stn select
151 iaddr <= addrfix when init|wr_lg,
152           addrwh when wr_wh|clear,
153           addrbcd when others;
154 addr <= std_logic_vector(iaddr);
155
156 with stn select
157 idata <= data_fix when init|wr_lg,
158          "11111110" when wr_wh|clear,
```

```
159          data_bcd when others;
160 data <= std_logic_vector(idata);
161
162 end lcdman;
```

s3etiny_bintobcd.vhd

```
1  entity anie_bintobcd is
2   port ( clk,srst: in  std_logic;
3          bin: in std_logic_vector(13 downto 0);
4          bcd: out std_logic_vector(15 downto 0) );
5  end anie_bintobcd;
6
7  architecture bintobcd of anie_bintobcd is
8
9  component anie_bcdshift
10  port ( clk,ce,srst,sin: in std_logic;
11         sout: out std_logic;
12         q: out std_logic_vector(3 downto 0) );
13 end component;
14
15  signal stat,ce_shift,rst_shift: std_logic;
16  signal sin0,sin1,sin2,sin3,sout0,sout1,sout2: std_logic;
17  signal shift_act,shift_next: unsigned(3 downto 0);
18
19  signal q0,q1,q2,q3: std_logic_vector(3 downto 0);
20  signal conv: std_logic_vector(13 downto 0);
21
22 begin
23
24 process(clk,srst,shift_act)
25 begin
26 if rising_edge(clk) then
27  if    srst='1'    then bcd <= (others => '0'); conv <= (others => '0');
28  elsif shift_act=0 then bcd <= q3&q2&q1&q0;     conv <= bin; end if;
29 shift_act <= shift_next;
30 end if;
31 end process;
32
33 shift_next <= shift_act +1;
34 stat <= '0' when shift_act(3 downto 1)="000" else '1';
35
36 ce_shift <= stat or shift_act(0);
37 rst_shift <= ((not stat) and shift_act(0)) or srst;
38
39 with shift_act select
40 sin0 <= conv(13) when "0010", conv(12) when "0011",
41         conv(11) when "0100", conv(10) when "0101",
42         conv(9) when "0110",  conv(8) when "0111",
43         conv(7) when "1000",  conv(6) when "1001",
44         conv(5) when "1010",  conv(4) when "1011",
```

```
45          conv(3) when "1100",  conv(2) when "1101",
46          conv(1) when "1110",  conv(0) when "1111",
47          '0' when others;
48
49  sin1 <= sout0;  sin2 <= sout1;  sin3 <= sout2;
50
51  A0: anie_bcdshift port map ( clk,ce_shift,rst_shift,sin0,sout0,q0 );
52  A1: anie_bcdshift port map ( clk,ce_shift,rst_shift,sin1,sout1,q1 );
53  A2: anie_bcdshift port map ( clk,ce_shift,rst_shift,sin2,sout2,q2 );
54  A3: anie_bcdshift port map ( clk,ce_shift,rst_shift,sin3,open,q3 );
55
56  end bintobcd;
```

s3etiny_bcdshift.vhd

```
1  entity anie_bcdshift is
2   port ( clk,ce,srst,sin: in std_logic;
3          sout: out std_logic;
4          q: out std_logic_vector(3 downto 0) );
5  end anie_bcdshift;
6
7  architecture bcdshift of anie_bcdshift is
8
9   signal bcd_act: std_logic_vector(3 downto 0);
10  signal bcd_next: std_logic_vector(4 downto 0);
11
12 begin
13
14 process(clk,srst)
15 begin
16 if rising_edge(clk) then
17  if srst='1'  then bcd_act <= (others =>'0');
18  elsif ce='1' then bcd_act <= bcd_next(3 downto 0); end if;
19 end if;
20 end process;
21
22 with bcd_act select
23 bcd_next <= "1000"&sin when "0101", "1001"&sin when "0110",
24              "1010"&sin when "0111", "1011"&sin when "1000",
25              "1100"&sin when "1001", '0'&bcd_act(2 downto 0)&sin when others;
26
27 sout <= bcd_next(4);
28 q <= bcd_act;
29
30 end bcdshift;
```

s3etiny_lcdmenu.vhd

```
1  entity anie_lcdmenu is
2   port ( clk,srst: in std_logic;
3          rot_e,rot_d,btnsel,btntg, btnmode: in std_logic;
```

```
4         inmenu,load,mode: out std_logic;
5         encsel: out unsigned(1 downto 0);
6         addr: out std_logic_vector(4 downto 0);
7         data: out std_logic_vector(7 downto 0) );
8 end anie_lcdmenu;
9
10 architecture lcdmenu of anie_lcdmenu is
11
12 component anie_switch
13  port ( clk,ce,srst: in std_logic;
14         sw: out std_logic);
15 end component;
16
17 component anie_udcounter
18  generic (wl: natural:=1);
19  port ( clk,srst,ce,dir: in std_logic;
20         count: out std_logic_vector(wl-1 downto 0));
21 end component;
22
23  type modes_type is (standby,lmode,lmodet,lmodew, lsure,lsuret,lsurew, tgmode,
       tgmodet,tgmodew,changed);
24  signal ma, mn, lmoden, lsuren, tgmoden: modes_type;
25
26  signal ca,cn: unsigned(13 downto 0);
27
28  signal tenco,tencce,tencrst,tmode,imode,ans: std_logic;
29  signal swmodece,swmoderst,swansce,swansrst: std_logic;
30  signal s,g,m,stdenc: std_logic_vector(1 downto 0);
31  signal tenc,iencsel: unsigned(1 downto 0);
32
33  signal addr_ca: natural;
34  signal iaddr,addrmodet,addrsure,addrsuret: unsigned(4 downto 0);
35  signal addrtgtc,addrtgtb,addrtgt: unsigned(4 downto 0);
36  signal datamode,datamodet,datasure,datasuret: std_logic_vector(7 downto 0);
37  signal datatg,datatgt: std_logic_vector(7 downto 0);
38  signal suretw,suretc,surete,xw,xe: std_logic_vector(7 downto 0);
39
40 begin
41
42 s(0) <= btnsel; g(0) <= btntg; m(0) <= btnmode;
43 mode <= imode; encsel <= iencsel;
44
45 process(clk,srst)
46 begin
47 if rising_edge(clk) then
48  if srst='1' then
49   ma <= standby; ca <= (others => '0');
50   s(1) <= '0'; g(1) <= g(0); m(1) <= m(0);
51   imode <= '0'; iencsel <= (others => '0');
52  else
53   ma <= mn; ca <= cn;
```

```
54   s(1) <= s(0); g(1) <= g(0); m(1) <= m(0);
55   if mn=changed then case ma is
56     when tgmodew => iencsel <= tenc;
57     when lsurew => imode <= tmode;
58     when others =>
59    end case; end if;
60  end if;
61 end if;
62 end process;
63
64 SWMODE: anie_switch port map ( clk,swmodece,swmoderst,tmode);
65 swmodece <= rot_e when ma=lmodew else
66             '1' when (ma=lmode and tmode='0' and imode='1') else '0';
67 swmoderst <= '1' when (ma=lmode and tmode='1' and imode='0') else srst;
68
69 SWANS: anie_switch port map ( clk,swansce,swansrst,ans);
70 swansce <= rot_e when ma=lsurew else
71            '1' when (ma=tgmodew and s="01" and ans='0') else '0';
72 swansrst <= '1' when( ma=lsure or ma=tgmode or (ma=tgmodew and rot_e='1'))
73              else srst;
74
75 ENCCOUNT: anie_udcounter generic map (2)
76 port map ( clk,tencrst,tencce,rot_d,stdenc );
77 tenc <= unsigned(stdenc);
78 tenco <= '1' when tenc>1 and rot_d='0' else '0';
79 tencce <= rot_e when (ma=tgmodew and tenco='0') else
80           '1' when (ma=tgmode and iencsel/=tenc) else '0';
81 tencrst <= '1' when tenc>2 else srst;
82
83 lmoden <= lsure when s="01" else standby when (g="01" or m="01") else lmodew;
84 lsuren <= lmode when (s="01" and ans='0') else
85           changed when (s="01" and ans='1') else
86           standby when (g="01" or m="01") else lsurew;
87 tgmoden <= tgmodet when (s="01" and ans='0') else
88            changed when (s="01" and ans='1') else
89            standby when (g="01" or m="01") else tgmodew;
90
91 process(ma,ca,g,m,rot_e,lmoden,lsuren,tgmoden)
92 begin
93 case ma is
94  when standby =>
95   if g="01" then mn <= tgmode;
96   elsif m="01" then mn <= lmode;
97   else mn <= standby; end if;
98   cn <= (others=>'0');
99
100  when lmode =>
101   if ca="111111" then mn <= lmodet; cn <= (others=>'0');
102   else mn <= lmode; cn <= ca+1; end if;
103
104  when lmodet =>
```

```
105   if ca="000011" then mn <= lmodew; cn <= (others=>'0');
106   else mn <= lmodet; cn <= ca+1; end if;
107
108  when lmodew =>
109   if rot_e='1' then mn <= lmodet;
110   else mn <= lmoden; end if;
111   cn <= (others=>'0');
112
113  when lsure =>
114   if ca="011111" then mn <= lsuret; cn <= (others=>'0');
115   else mn <= lsure; cn <= ca+1; end if;
116
117  when lsuret =>
118   if ca="000101" then mn <= lsurew; cn <= (others=>'0');
119   else mn <= lsuret; cn <= ca+1; end if;
120
121  when lsurew =>
122   if rot_e='1' then mn <= lsuret;
123   else mn <= lsuren; end if;
124   cn <= (others=>'0');
125
126  when tgmode =>
127   if ca="111111" then mn <= tgmodet; cn <= (others=>'0');
128   else mn <= tgmode; cn <= ca+1; end if;
129
130  when tgmodet =>
131   if ca="001011" then mn <= tgmodew; cn <= (others=>'0');
132   else mn <= tgmodet; cn <= ca+1; end if;
133
134  when tgmodew =>
135   if rot_e='1' then mn <= tgmodet;
136   else
137    mn <= tgmoden;
138   end if;
139   cn <= (others=>'0');
140
141   when changed => mn <= standby; cn <= (others => '0');
142
143   when others => mn <= standby; cn <= (others => '0');
144
145 end case;
146 end process;
147
148 with ma select load <= ca(0) when lmode|lmodet|lsure|lsuret|tgmode|tgmodet,
149                        '0' when others;
150 inmenu <= '0' when ma=standby else '1';
151
152 with ma select
153 iaddr <= addrmodet when lmodet, addrsure when lsure, addrsuret when lsuret,
154          addrtgt when tgmodet, ca(5 downto 1) when others;
155 addr <= std_logic_vector(iaddr);
```

```
156 addr_ca <= to_integer(ca(5 downto 1));
157
158 with ma select
159 data <= datamode when lmode, datamodet when lmodet,
160         datasure when lsure, datasuret when lsuret,
161         datatg when tgmode, datatgt when tgmodet,
162         "11111110" when others;
163
164 --lmode
165 with addr_ca select
166 datamode <= "01000011" when 2, "01001100" when 3|9|23, -- C L
167             "01001111" when 4|10|11|18|24|25, "01010011" when 5, --O S
168             "01000101" when 6|20, "01000100" when 7, -- E D
169             "01010000" when 12|19|26, "01001110" when 21, -- P N
170             "11111110" when others;
171
172 --lmodet
173 addrmodet <= to_unsigned(0,5) when addr_ca=0 else to_unsigned(16,5);
174 datamodet <= "00100011" when (addr_ca=0 and tmode='0') or
175              (addr_ca/=0 and tmode='1') else "11111110";
176
177 --lsure
178 addrsure <= ca(5 downto 1) when tmode='1' else ca(5 downto 1)+16;
179 with addr_ca select
180 datasure <= "00101110" when 2|3|4, "01110011" when 5,-- . s
181             "01110101" when 6, "01110010" when 7,-- u r
182             "01100101" when 8, "00111111" when 9,-- e ?
183             "01001110" when 12, "01011001" when 14,-- N Y
184             "11111110" when others;
185
186 --lsuret
187 with addr_ca select
188 addrsuret <= addrsure+11 when 0, addrsure+12 when 1, addrsure+13 when others;
189 with addr_ca select datasuret <= suretw when 0, suretc when 1, surete when
        others;
190 suretw <= "00111100" when ans='0' else "11111110";
191 suretc <= "00111110" when ans='0' else "00111100";
192 surete <= "00111110" when ans='1' else "11111110";
193
194 --tgmode
195 with addr_ca select
196 datatg <= "01010100" when 2, "01010010" when 3|8, "01001001" when 4,-- T R I
197           "01000111" when 5|6, "01000101" when 7|13, "01001101" when 10,-- G E M
198           "01001111" when 11, "01000100" when 12, "00110001" when 29,--O D 1
199           "00110010" when 24, "00110100" when 19, "01011000" when 18|23|28,-- 2
                4 X
200           "11111110" when others;
201
202 --tgmodet
203 with tenc select
204 addrtgtc <= to_unsigned(17,5) when "00", to_unsigned(22,5) when "01",
```

```
205             to_unsigned(27,5) when "10", (others => '0') when others;
206
207 with addr_ca select
208 addrtgtb <= to_unsigned(17,5) when 0|1, to_unsigned(22,5) when 2|3,
209             to_unsigned(27,5) when 4|5, (others => '0') when others;
210 addrtgt <= addrtgtb+3 when ca(1)='1' else addrtgtb;
211
212 datatgt <= xw when addrtgtc=addrtgt else
213            xe when addrtgtc+3=addrtgt else "11111110";
214 xw <= "00111111" when ans='1' else "00111100";
215 xe <= "00111111" when ans='1' else "00111110";
216
217 end lcdmenu;
```

s3etiny_switch.vhd

```
1  entity anie_switch is
2   port ( clk,ce,srst: in std_logic;
3          sw: out std_logic);
4  end anie_switch;
5
6  architecture switch of anie_switch is
7
8   signal swa,swn: std_logic;
9
10 begin
11
12 process (clk,srst,ce)
13 begin
14 if rising_edge(clk) then
15  if srst='1' then swa <= '0';
16  elsif ce='1' then swa <= swn; end if;
17 end if;
18 end process;
19
20 swn <= not swa; sw <= swa;
21
22 end switch;
```

s3etiny_btndeb.vhd

```
1  entity anie_btndeb is
2   port ( clk: in std_logic;
3          btn: in std_logic_vector(2 downto 0);
4          btndeb: out std_logic_vector(2 downto 0) );
5  end anie_btndeb;
6
7  architecture btndeb of anie_btndeb is
8
9  component anie_debounce
10  port ( clk,p_in: in  std_logic;
```

```
11          p_out: out std_logic );
12 end component;
13
14  signal edeb,wdeb,xdeb: std_logic_vector(2 downto 0);
15
16 begin
17
18 DEBE: anie_debounce port map ( clk,btn(0),edeb(0) );
19 DEBW: anie_debounce port map ( clk,btn(1),wdeb(0) );
20 DEBX: anie_debounce port map ( clk,btn(2),xdeb(0) );
21
22 process(clk)
23 begin
24 if rising_edge(clk) then
25  edeb(1) <= edeb(0); wdeb(1) <= wdeb(0); xdeb(1) <= xdeb(0);
26 end if;
27 end process;
28
29 edeb(2) <= '1' when edeb(1 downto 0)="01" else '0'; btndeb(0) <= edeb(2);
30 wdeb(2) <= '1' when wdeb(1 downto 0)="01" else '0'; btndeb(1) <= wdeb(2);
31 xdeb(2) <= '1' when xdeb(1 downto 0)="01" else '0'; btndeb(2) <= xdeb(2);
32
33 end btndeb;
```

s3etiny_debounce.vhd

```
1 entity anie_debounce is
2  port ( clk,p_in: in  std_logic;
3          p_out: out std_logic );
4 end anie_debounce;
5
6 architecture debounce of anie_debounce is
7
8  signal shift: std_logic_vector(3 downto 0);
9
10 begin
11
12 process (clk)
13 begin
14 if rising_edge(clk) then
15  shift(3 downto 1) <= shift(2 downto 0);
16  shift(0) <= p_in;
17 end if;
18 end process;
19
20 p_out <= '1' when shift="1111" else '0';
21
22 end debounce;
```

s3etiny_norm.vhd

```
1  entity anie_norm is
2   generic ( rwl: natural:=3;
3             vwl: natural:=7;
4             iwl: natural:=7;
5             owl: natural:=13;
6             nwl: natural:=8 );
7   port ( clk,tg,srst: in std_logic;
8          nref: in signed(rwl-1 downto 0);
9          nvar: in signed(vwl-1 downto 0);
10         nout: in signed(owl-1 downto 0);
11         pid_ref,pid_var: out signed(iwl-1 downto 0);
12         open_ref: out signed(owl-1 downto 0);
13         lcdr,lcdv,lcdo: out signed(13 downto 0);
14         clk_pid: out std_logic );
15 end anie_norm;
16
17 architecture norm of anie_norm is
18
19  constant npidr: signed(nwl-1 downto 0):=to_signed(1,nwl);  --40/40 -> 1 * 40 =
       40
20  constant npidv: signed(nwl-1 downto 0):=to_signed(1,nwl); -- 40/40 -> 1 * 40 =
       40
21
22  constant nopenr: signed(nwl-1 downto 0):=to_signed(102,nwl); -- 1023/40 -> 25.5
        * 40 = 4080
23
24  constant nlcdr: signed(nwl-1 downto 0):=to_signed(25,nwl); -- 1000/40 -> 25 *
       40 = 1000
25  constant nlcdv: signed(nwl-1 downto 0):=to_signed(25,nwl); -- 1000/40 -> 25 *
       40 = 1000
26  constant nlcdo: signed(nwl-1 downto 0):=to_signed(63,nwl); -- 1000/1023 ->
       0.984375 * 1023 = 1007.015625
27
28  type norm_type is (standby,getrv,calcrv,putrv,dorv,geto,calco,puto,doo);
29  signal norm_act, norm_next: norm_type;
30
31  signal gr: signed(rwl-1 downto 0);
32  signal gv: signed(vwl-1 downto 0);
33  signal go: signed(owl-1 downto 0);
34
35  signal pr,pv: signed(iwl-1 downto 0);
36  signal lr,lv,lo: signed(13 downto 0);
37  signal wake: std_logic_vector(1 downto 0);
38
39 begin
40
41 pr <= resize(shift_right(resize(gr*npidr,rwl+nwl-1),0),iwl);
42 pv <= resize(shift_right(resize(gv*npidv,vwl+nwl-1),0),iwl);
43
44 open_ref <= resize(shift_right(resize(gr*nopenr,rwl+nwl-1),2),owl);
45
```

```
46 lr <= resize(shift_right(resize(gr*nlcdr,rwl+nwl-1),0),14);
47 lv <= resize(shift_right(resize(gv*nlcdv,vwl+nwl-1),0),14);
48 lo <= resize(shift_right(resize(go*nlcdo,owl+nwl-1),6),14);
49
50 process(clk,srst)
51 begin
52 if rising_edge(clk) then
53  if srst='1' then
54   wake <= (others => '0');
55   norm_act <= standby;
56
57   gr <= (others => '0');
58   gv <= (others => '0');
59   go <= (others => '0');
60
61   lcdr <= (others => '0');
62   lcdv <= (others => '0');
63   lcdo <= (others => '0');
64  else
65   wake(1) <= wake(0);
66   wake(0) <= tg;
67   norm_act <= norm_next;
68
69   if norm_next=getrv then gr <= nref; gv <= nvar; end if;
70   if norm_next=putrv then pid_ref <= pr; pid_var <= pv; end if;
71   if norm_next=geto then go <= nout; end if;
72   if norm_next=puto then
73    lcdr <= lr;
74    lcdv <= lv;
75    lcdo <= lo;
76   end if;
77  end if;
78 end if;
79 end process;
80
81 process(norm_act,wake)
82 begin
83 case norm_act is
84  when standby =>
85   if wake="01" then norm_next <= getrv;
86    else norm_next <= standby;
87   end if;
88
89  when getrv => norm_next <= calcrv;
90  when calcrv => norm_next <= putrv;
91  when putrv => norm_next <= dorv;
92  when dorv => norm_next <= geto;
93  when geto => norm_next <= calco;
94  when calco => norm_next <= puto;
95  when puto => norm_next <= doo;
96  when doo => norm_next <= standby;
```

```
97 end case;
98 end process;
99
100 clk_pid <= '1' when norm_act=dorv else '0';
101
102 end norm;
```

lcdfsm_lcd.vhd

```
entity anie_lcd is
 port ( clk10,clk160,srst,load: in std_logic;
        addr: in std_logic_vector(4 downto 0);
        data: in std_logic_vector(7 downto 0);
        LCD_E, LCD_RS, LCD_RW: out std_logic;
        SF_D: out std_logic_vector(3 downto 0) );
end anie_lcd;

architecture lcd of anie_lcd is

component anie_autoinit
 port ( clk,srst: in std_logic;
        tout: in std_logic_vector(2 downto 0);
        tg: out std_logic );
end component;

component anie_lcd_setup
 port ( clk,srst,tg: in std_logic;
        LCD_E,ready: out std_logic;
        SF_D: out std_logic_vector(3 downto 0) );
end component;

component anie_lcd_writing
 port ( clk,srst,tg: in std_logic;
        sfd: in std_logic_vector(7 downto 0);
        LCD_E,done: out std_logic;
        SF_D: out std_logic_vector(3 downto 0) );
end component;

 type ram_type is array (integer range 0 to 31) of std_logic_vector(7 downto 0);
 signal lcd_ram: ram_type;
 signal addr_read: std_logic_vector(4 downto 0);
 signal data_read,remaining,sfd_main: std_logic_vector(7 downto 0);
 signal sfd_setup,sfd_writing: std_logic_vector (3 downto 0);
 signal lcde_setup, lcde_writing: std_logic;
 signal count_act,count_next: unsigned(13 downto 0);
 constant tc80k: integer:=5000;
 signal tg_int, ready, done, paused: std_logic;
 signal go: std_logic_vector(2 downto 0);

 signal main_act, main_next: std_logic_vector(5 downto 0);
 constant waiting: std_logic_vector(5 downto 0):="100000";
 constant functionset: std_logic_vector(5 downto 0):="100001";
 constant entry: std_logic_vector(5 downto 0):="100010";
 constant display: std_logic_vector(5 downto 0):="100011";
```

```
46  constant clear: std_logic_vector(5 downto 0):="100100";
47  constant pause: std_logic_vector(5 downto 0):="100101";
48  constant addr_fline: std_logic_vector(5 downto 0):="100110";
49  constant addr_sline: std_logic_vector(5 downto 0):="100111";
50
51 begin
52
53 process (clk10)
54 begin
55 if rising_edge(clk10) then
56  data_read <= lcd_ram(to_integer(unsigned(addr_read)));
57  if (load = '1') then lcd_ram(to_integer(unsigned(addr))) <= data; end if;
58 end if;
59 end process;
60
61 LCD_RW <= '0';
62 with main_act select LCD_RS <=
63 '0' when waiting|functionset|entry|display|clear|addr_fline|addr_sline, '1' when
       others;
64 with main_act select LCD_E <= lcde_setup when waiting, lcde_writing when others;
65 with main_act select SF_D <= sfd_setup when waiting, sfd_writing when others;
66
67 AUTO: anie_autoinit port map ( clk160,srst,"100",tg_int );
68 FSM_SETUP: anie_lcd_setup
69 port map ( clk160,srst,tg_int,lcde_setup,ready,sfd_setup );
70 FSM_WRITE: anie_lcd_writing
71 port map ( clk10,srst,go(2),sfd_main,lcde_writing,done,sfd_writing );
72
73 -- FSM_MAIN
74 process(clk160,srst)
75 begin
76 if rising_edge(clk160) then
77  if srst='1' then main_act <= waiting;
78  else main_act <= main_next; go(2 downto 1) <= go(1 downto 0);
79   if paused='1' then count_act <= count_next;
80   else count_act <= (others=>'0'); end if;
81  end if;
82 end if;
83 end process;
84
85 count_next <= count_act +1;
86
87 with main_next select
88 sfd_main <= "00101000" when waiting,     "00101000" when functionset,
89             "00000110" when entry,       "00001100" when display,
90             "00000001" when clear,       "10000000" when addr_fline,
91             "11000000" when addr_sline, remaining when others;
92
93 remaining <= data_read when main_next(5)='0' else (others => '0');
94 addr_read <= main_next(4 downto 0) when main_next(5)='0' else (others => '0');
95
```

```
96  process(main_act,ready,done,count_act)
97  begin
98  case main_act is
99   when waiting =>
100   if ready='1' then main_next <= functionset; go(0) <= '1';
101   else main_next <= waiting; go(0) <= '0'; end if;
102   paused <= '0';
103
104  when functionset =>
105   if done='1' then main_next <= entry; go(0) <= '1';
106   else main_next <= functionset; go(0) <= '0'; end if;
107   paused <= '0';
108
109  when entry =>
110   if done='1' then main_next <= display; go(0) <= '1';
111   else main_next <= entry; go(0) <= '0'; end if;
112   paused <= '0';
113
114  when display =>
115   if done='1' then main_next <= clear; go(0) <= '1';
116   else main_next <= display; go(0) <= '0'; end if;
117   paused <= '0';
118
119  when clear =>
120   if done='1' then main_next <= pause; go(0) <= '1'; paused <= '1';
121   else main_next <= clear; go(0) <= '0'; paused <= '0'; end if;
122
123  when pause =>
124   if count_act=tc80k then main_next <= addr_fline; go(0) <= '1'; paused <= '0';
125   else main_next <= pause; go(0) <= '0'; paused <= '1'; end if;
126
127  when addr_fline =>
128   if done='1' then main_next <= (others=>'0'); go(0) <= '1';
129   else main_next <= addr_fline; go(0) <= '0'; end if;
130   paused <= '0';
131
132  when addr_sline =>
133   if done='1' then main_next <= (4 => '1', others => '0'); go(0) <= '1';
134   else main_next <= addr_sline; go(0) <= '0'; end if;
135   paused <= '0';
136
137  when others =>
138   paused <= '0';
139   if main_act(5)='1' then main_next <= clear; go(0) <= '0';
140   else
141    if done='1' then
142     if main_act(3 downto 0)="1111" then main_next <= "10011"&not main_act(4);
143     else main_next <= std_logic_vector(unsigned(main_act) +1); end if;
144     go(0) <= '1';
145    else main_next <= main_act; go(0) <= '0'; end if; end if;
146 end case;
```

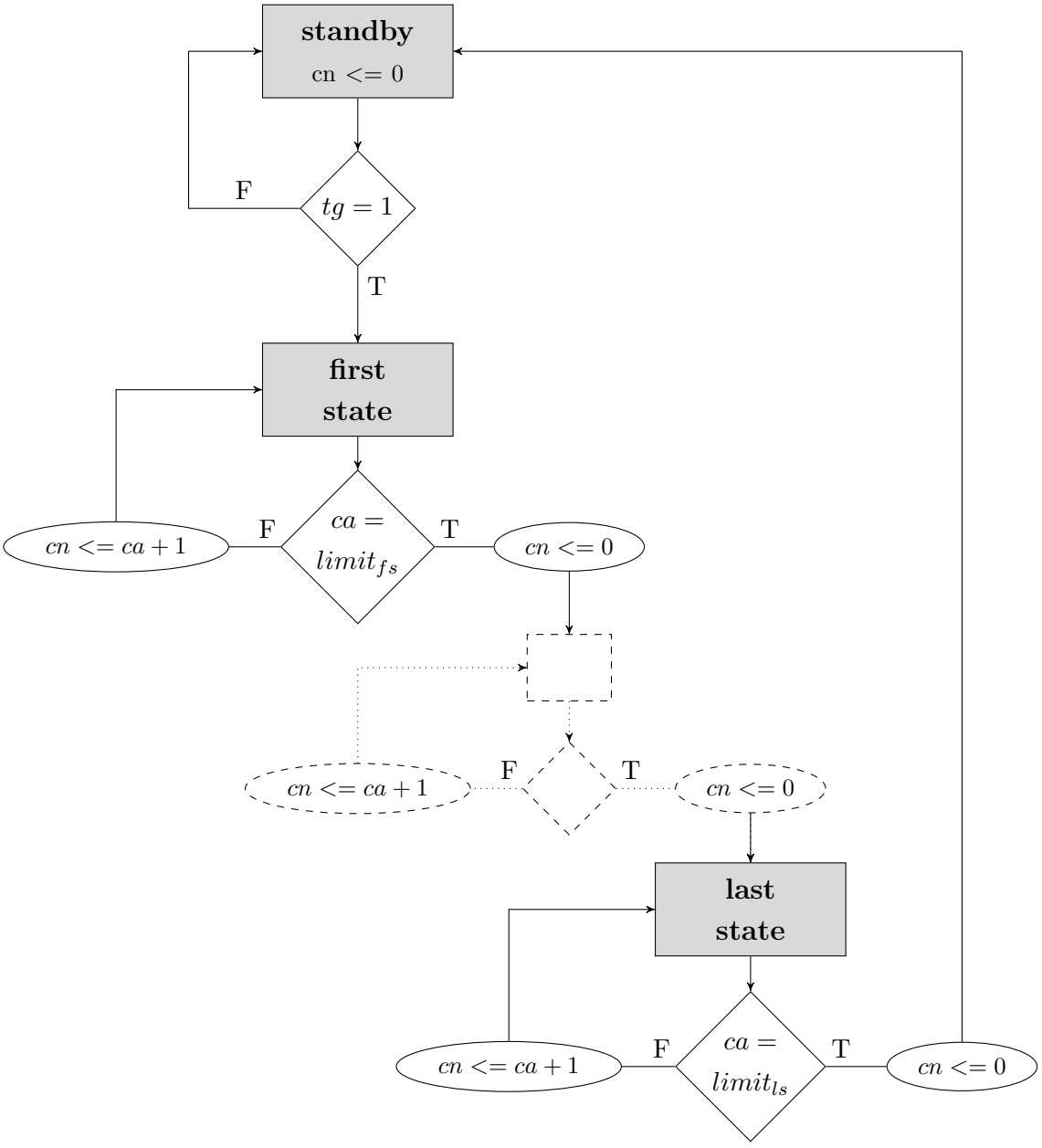
```
147 end process;
148
149 end lcd;
```

lcdfsm_setup.vhd

```
1  entity anie_lcd_setup is
2    port ( clk,srst,tg: in std_logic;
3           LCD_E,ready: out std_logic;
4           SF_D: out std_logic_vector(3 downto 0) );
5  end anie_lcd_setup;
6
7  architecture lcd_setup of anie_lcd_setup is
8
9   type setup_type is (standby,
10                      wait1,sf1,wait2,sf2,wait3,sf3,wait4,sf4,wait5,
11                      done);
12   signal sa, sa: setup_type;
13   signal ca,cn: unsigned(13 downto 0):=(others=>'0');
14
15   constant tc750k: integer:=4687;
16   constant tc205k: integer:=1281;
17   constant tc5k: integer:=31;
18   constant tc2k: integer:=12;
19   constant tc12: integer:=1;
20
21 begin
22
23 process(clk,srst)
24 begin
25 if rising_edge(clk) then
26    if srst='1' then sa <= standby; ca <= (others => '0');
27    else sa <= sa; ca <= cn; end if;
28 end if;
29 end process;
30
31 with sa select
32 SF_D <= "0000" when standby, "0000" when wait1, "0010" when sf4|wait5|done,
33         "0011" when sf1|wait2|sf2|wait3|sf3|wait4, "0000" when others;
34
35 with sa select LCD_E <= '1' when sf1|sf2|sf3|sf4, '0' when others;
36
37 with sa select ready <= '1' when done, '0' when others;
38
39 process(sa,tg,ca)
40 begin
41 case sa is
42   when standby =>
43    if tg='1' then sa <= wait1;
44    else sa <= standby; end if;
45    cn <= (others=>'0');
```

setup	limit	next state	
wait1	tc750k	sf1	first
sf1	tc12	wait2	
wait2	tc205k	sf2	
sf2	tc12	wait3	
wait3	tc5k	sf3	
sf3	tc12	wait4	
wait4	tc2k	sf4	
sf4	tc12	wait5	
wait5	tc2k	done	
done	tc12	standby	last

writing	limit	next state	
setup1	tc12	upper_en	first
upper_en	tc12	upper_dis	
upper_dis	tc48	setup2	
setup2	tc12	lower_en	
lower_en	tc12	lower_dis	
lower_dis	tc2k	written	
written	11	standby	last



3D.1. Irudia: *Inplementatutako* setup *eta* writing *makinen algoritmoaren diagrama.*

```
46
47  when wait1 =>
48   if ca=tc750k then sa <= sf1; cn <= (others=>'0');
49   else sa <= wait1; cn <= ca +1; end if;
50
51  when sf1 =>
52   if ca=tc12 then sa <= wait2; cn <= (others=>'0');
53   else sa <= sf1; cn <= ca +1; end if;
54
55  when wait2 =>
56   if ca=tc205k then sa <= sf2; cn <= (others=>'0');
57   else sa <= wait2; cn <= ca +1; end if;
58
59  when sf2 =>
60   if ca=tc12 then sa <= wait3; cn <= (others=>'0');
61   else sa <= sf2; cn <= ca +1; end if;
62
63  when wait3 =>
64   if ca=tc5k then sa <= sf3; cn <= (others=>'0');
65   else sa <= wait3; cn <= ca +1; end if;
66
67  when sf3 =>
68   if ca=tc12 then sa <= wait4; cn <= (others=>'0');
69   else sa <= sf3; cn <= ca +1; end if;
70
71  when wait4 =>
72   if ca=tc2k then sa <= sf4; cn <= (others=>'0');
73   else sa <= wait4; cn <= ca +1; end if;
74
75  when sf4 =>
76   if ca=tc12 then sa <= wait5; cn <= (others=>'0');
77   else sa <= sf4; cn <= ca +1; end if;
78
79  when wait5 =>
80   if ca=tc2k then sa <= done; cn <= (others=>'0');
81   else sa <= wait5; cn <= ca +1; end if;
82
83  when done =>
84   if ca=tc12 then sa <= standby; cn <= (others => '0');
85   else sa <= done; cn <= ca +1; end if;
86
87  when others => sa <= standby; cn <= (others => '0');
88
89 end case;
90 end process;
91
92 end lcd_setup;
```

lcdfsm_writing.vhd

```
1 entity anie_lcd_writing is
```

```
  port ( clk,srst,tg: in std_logic;
         sfd: in std_logic_vector(7 downto 0);
         LCD_E,done: out std_logic;
         SF_D: out std_logic_vector(3 downto 0) );
end anie_lcd_writing;

architecture lcd_writing of anie_lcd_writing is

 type writing_type is (standby,setup1,upper_en,upper_dis,setup2,lower_en,
    lower_dis,written);
 signal wa, wn: writing_type;
 signal ca,cn: unsigned(8 downto 0);

 constant tc2k: integer:=200;
 constant tc48: integer:=4;
 constant tc12: integer:=1;

begin

process(clk,srst)
begin
if rising_edge(clk) then
  if srst='1' then wa <= standby;
  else wa <= wn; ca <= cn; end if;
end if;
end process;

with wa select
SF_D <= sfd(7 downto 4) when standby|setup1|upper_en|upper_dis,
        sfd(3 downto 0) when setup2|lower_en|lower_dis|written,
        "0000" when others;

with wa select  LCD_E <= '1' when upper_en|lower_en, '0' when others;
with wa select done <= '1' when written, '0' when others;

process(wa,tg,ca)
begin
case wa is
 when standby =>
  if tg='1' then wn <= setup1;
  else wn <= standby; end if;
  cn <= (others=>'0');

 when setup1 =>
  if ca=tc12 then wn <= upper_en; cn <= (others=>'0');
  else wn <= setup1; cn <= ca +1; end if;

 when upper_en =>
  if ca=tc12 then wn <= upper_dis; cn <= (others=>'0');
  else wn <= upper_en; cn <= ca +1; end if;

```

```
 when upper_dis =>
  if ca=tc48 then wn <= setup2; cn <= (others=>'0');
  else wn <= upper_dis; cn <= ca +1; end if;

 when setup2 =>
  if ca=tc12 then wn <= lower_en; cn <= (others=>'0');
  else wn <= setup2; cn <= ca +1; end if;

 when lower_en =>
  if ca=tc12 then wn <= lower_dis; cn <= (others=>'0');
  else wn <= lower_en; cn <= ca +1; end if;

 when lower_dis =>
  if ca=tc2k then wn <= written; cn <= (others=>'0');
  else wn <= lower_dis; cn <= ca +1; end if;

 when written =>
  if ca=11 then wn <= standby; cn <= (others => '0');
  else wn <= written; cn <= ca +1; end if;

 when others => wn <= standby; cn <= (others => '0');
end case;
end process;

end lcd_writing;
```

lcdfsm_autoinit.vhd

```
entity anie_autoinit is
  port ( clk,srst: in std_logic;
         tout: in std_logic_vector(2 downto 0);
         tg: out std_logic );
end anie_autoinit;

architecture autoinit of anie_autoinit is

 signal auto: std_logic_vector(10 downto 0);

begin

process(clk)
begin
if rising_edge(clk) then
 auto(2 downto 0) <= auto(5 downto 3); tg <= auto(6);
end if;
end process;

auto(6) <= tout(2) and not auto(2);
auto(7) <= tout(1) and not auto(1);
auto(8) <= tout(0) and not auto(0);
auto(9) <= auto(6) or ((not auto(6)) and auto(7)) or
```

```
24            ((not auto(6)) and (not auto(7)) and auto(8));
25 auto(10) <= srst;
26
27 with auto(10 downto 9) select
28 auto(5 downto 3) <= tout when "00",
29                     std_logic_vector(unsigned(auto(2 downto 0))+1) when "01",
30                     (others => '0') when others;
31
32 end autoinit;
```

s3e_timing.vhd

```
entity anie_timing is
 port ( clk,rst: in std_logic;
        lrst,clk_adq,clk_pwm,clk_hb,clk1s: out std_logic;
        clk0,clk2,clk10,clk160,clk12_5k,clk0_1k: out std_logic );
end anie_timing;

architecture timing of anie_timing is

component anie_dcm
 port( U1_CLKIN_IN: IN std_logic;
       U1_RST_IN: IN std_logic;
       U1_CLKDV_OUT: OUT std_logic;
       U1_CLKIN_IBUFG_OUT: OUT std_logic;
       U1_CLK0_OUT: OUT std_logic;
       U2_CLKDV_OUT: OUT std_logic;
       U2_CLK0_OUT: OUT std_logic;
       U2_LOCKED_OUT: OUT std_logic );
end component;

component anie_counter
 generic (wl: natural:=3);
 port ( clk,ce,srst: in std_logic;
        limit: in std_logic_vector(wl-1 downto 0);
        count: out std_logic_vector(wl-1 downto 0);
        tc: out std_logic);
end component;

component anie_switch
 port ( clk,ce,srst: in std_logic;
        sw: out std_logic);
end component;

constant dv25l:std_logic_vector(4 downto 0):=std_logic_vector(to_unsigned(24,5))
    ;
constant dv125l:std_logic_vector(6 downto 0):=std_logic_vector(to_unsigned
    (124,7));
constant dv100l:std_logic_vector(6 downto 0):=std_logic_vector(to_unsigned(99,7)
    );

 signal u2locked,lock,mrst,ilrst: std_logic;
 signal iclk0,iclk2,iclk10,iclk160,iclk12_5k,iclk0_1k,iclk1s: std_logic;

attribute clock_signal : string;
attribute clock_signal of iclk12_5k: signal is "yes";
attribute clock_signal of iclk0_1k: signal is "yes";
```

```
43 attribute clock_signal of iclk1s: signal is "yes";
44
45 attribute buffer_type: string;
46 attribute buffer_type of iclk12_5k: signal is "bufg";
47 attribute buffer_type of iclk0_1k: signal is "bufg";
48 attribute buffer_type of iclk1s: signal is "bufg";
49
50 begin
51
52 mrst <= rst or (not u2locked);
53
54 PDCM: anie_dcm
55 port map ( U1_CLKIN_IN => clk,
56            U1_RST_IN => rst,
57            U1_CLKDV_OUT => iclk10, --5 MHz
58            U1_CLKIN_IBUFG_OUT => open,
59            U1_CLK0_OUT => iclk0,
60            U2_CLKDV_OUT => iclk160, --312,5 KHz
61            U2_CLK0_OUT => open,
62            U2_LOCKED_OUT => u2locked );
63
64 DV2: anie_switch port map ( iclk0,'1',mrst,iclk2 );
65 DV25: anie_counter generic map ( 5 )
66 port map ( iclk160,'1',mrst,dv25l,open,iclk12_5k );
67 DV125: anie_counter generic map ( 7 )
68 port map ( iclk160,iclk12_5k,mrst,dv125l,open,iclk0_1k );
69 DV100: anie_counter generic map ( 7 )
70 port map ( iclk12_5k,iclk0_1k,ilrst,dv100l,open,iclk1s );
71
72 clk0 <= iclk0;          clk2 <= iclk2;
73 clk10 <= iclk10;        clk160 <= iclk160;
74 clk12_5k <= iclk12_5k; clk0_1k <= iclk0_1k;
75
76 clk_adq <= iclk0_1k;    clk_pwm <= iclk10;
77 clk1s <= iclk1s;        clk_hb <= iclk0_1k;
78
79 process(iclk0_1k)
80 begin
81 if rising_edge(iclk0_1k) then lock <= u2locked; end if;
82 end process;
83 ilrst <= rst or (lock nand u2locked);
84 lrst <= ilrst;
85
86 end timing;
```

s3e_man.vhd

```
1 entity anie_man is
2  generic ( ref_wl: natural:=3;
3            vdata_wl: natural:=9 );
4  port ( clk0,clk2,clk10,clk160,clk12_5k,clk0_1k,clk1s,vga_tg: in std_logic;
```

```
5         srst,ref_dir: in std_logic;
6         rot: in std_logic_vector(1 downto 0);
7         btn: in std_logic_vector(4 downto 0);
8         lmode: out std_logic;
9         LCD_E,LCD_RS,LCD_RW: out std_logic;
10        SF_D: out std_logic_vector(3 downto 0);
11        hsync,vsync: out std_logic;
12        vga: out std_logic_vector(2 downto 0);
13        encsel: out unsigned(1 downto 0);
14        ref_man: out signed(ref_wl-1 downto 0);
15        lcdr,lcdv,lcdo: in signed(13 downto 0);
16        vgar,vgav,vgao: in signed(vdata_wl-1 downto 0) );
17 end anie_man;
18
19 architecture man of anie_man is
20
21 component anie_rot is
22  port ( clk: in std_logic;
23         rot: in std_logic_vector(1 downto 0);
24         rot_e,rot_d: out std_logic );
25 end component;
26
27 component anie_udcounter
28  generic (wl: natural:=1);
29  port ( clk,srst,ce,dir: in std_logic;
30         count: out std_logic_vector(wl-1 downto 0));
31 end component;
32
33 component anie_lcd
34  port ( clk10,clk160,srst,load: in std_logic;
35         addr: in std_logic_vector(4 downto 0);
36         data: in std_logic_vector(7 downto 0);
37         LCD_E, LCD_RS, LCD_RW: out std_logic;
38         SF_D: out std_logic_vector(3 downto 0) );
39 end component;
40
41 component anie_lcdman
42  generic ( ref_wl: natural:=3 );
43  port ( clk,srst: in std_logic;
44         load: out std_logic;
45         lcdr,lcdv,lcdo: in signed(13 downto 0);
46         addr: out std_logic_vector(4 downto 0);
47         data: out std_logic_vector(7 downto 0) );
48 end component;
49
50 component anie_lcdmenu
51  port ( clk,srst: in std_logic;
52         rot_e,rot_d,btnsel,btntg,btnmode: in std_logic;
53         inmenu,load,mode: out std_logic;
54         encsel: out unsigned(1 downto 0);
55         addr: out std_logic_vector(4 downto 0);
```

```
56          data: out std_logic_vector(7 downto 0) );
57  end component;
58
59  component anie_vgasync
60   port ( clk,clk2,rst: in  std_logic;
61          line: out unsigned(8 downto 0);
62          column: out unsigned(9 downto 0);
63          hsync,vsync:    out std_logic );
64  end component;
65
66  component anie_vgaman
67   generic ( dwl: natural:=9 );
68   port ( clk,clk1s,vga_tg,srst: in  std_logic;
69          line: in unsigned(8 downto 0);
70          dref,dvar,dout: in signed(dwl-1 downto 0);
71          column: in  unsigned(9 downto 0);
72          sel: in std_logic_vector(1 downto 0);
73          vga: out std_logic_vector(2 downto 0));
74  end component;
75
76  component anie_btndeb
77   port ( clk: in std_logic;
78          btn: in std_logic_vector(4 downto 0);
79          btndeb: out std_logic_vector(4 downto 0) );
80  end component;
81
82  component anie_counter
83   generic (wl: natural:=3);
84   port ( clk,ce,srst: in std_logic;
85          limit: in std_logic_vector(wl-1 downto 0);
86          count: out std_logic_vector(wl-1 downto 0);
87          tc: out std_logic);
88  end component;
89
90  component anie_switch
91   port ( clk,ce,srst: in std_logic;
92          sw: out std_logic);
93  end component;
94
95  --REF
96   signal rce,rot_e,rot_d: std_logic;
97   signal std_ref: std_logic_vector(ref_wl-2 downto 0);
98
99  --LCD
100  signal load,loadman,loadmenu,lcdman_rst,inmenu: std_logic;
101  signal addr,addrman,addrmenu: std_logic_vector(4 downto 0);
102  signal data,dataman,datamenu: std_logic_vector(7 downto 0);
103  signal enc_int: unsigned(1 downto 0);
104
105 --VGA
106  signal line: unsigned(8 downto 0);
```

```
107  signal column: unsigned(9 downto 0);
108  signal ivtg,vtge: std_logic;
109  signal vm: std_logic_vector(1 downto 0);
110
111 --DEB
112  signal btndeb: std_logic_vector(4 downto 0);
113
114 begin
115
116 --REF
117 ROTF: anie_rot port map ( clk10,rot,rot_e,rot_d );
118 REF: anie_udcounter generic map (ref_wl-1)
119 port map ( clk10,srst,rce,rot_d,std_ref );
120 rce <= '0' when ((unsigned(std_ref)>39 and rot_d='0') or
121        (unsigned(std_ref)=0 and rot_d='1')) or (inmenu='1' or vtge='1') else
            rot_e;
122 ref_man <= signed('0'&std_ref) when ref_dir='0' else signed(not ('0'&std_ref))
       +1;
123
124 --LCD
125 LCD: anie_lcd
126 port map ( clk10,clk160,srst,load,addr,data,LCD_E,LCD_RS,LCD_RW,SF_D );
127
128 LCDMAN: anie_lcdman generic map( ref_wl )
129 port map ( clk12_5k,lcdman_rst,loadman,lcdr,lcdv,lcdo,addrman,dataman );
130 lcdman_rst <= srst or inmenu;
131
132 LCDMENU: anie_lcdmenu
133 port map ( clk10, srst, rot_e, rot_d, btndeb(4), btndeb(3), btndeb(1),
134            inmenu, loadmenu, lmode, enc_int, addrmenu, datamenu );
135 encsel <= 3-enc_int;
136
137 load <= loadmenu when inmenu='1' else loadman;
138 addr <= addrmenu when inmenu='1' else addrman;
139 data <= datamenu when inmenu='1' else dataman;
140
141 --VGA
142 VGASYNC: anie_vgasync port map ( clk0, clk2, srst, line, column, hsync, vsync );
143 VMODE: anie_counter generic map ( 2 )
144 port map ( clk0_1k,btndeb(0),srst,"11",vm,open);
145 VGAMAN: anie_vgaman generic map ( vdata_wl )
146 port map ( clk2, clk1s, ivtg, srst, line, vgar, vgav, vgao, column, vm, vga );
147 ivtg <= (not vtge) and vga_tg;
148 PAUSE: anie_switch port map ( clk0_1k,btndeb(2),srst,vtge );
149
150 --DEB
151 DEB: anie_btndeb port map ( clk0_1k, btn, btndeb );
152
153 end man;
```

s3e_vgasync.vhd

```
entity anie_vgasync is
 port ( clk,clk2,rst: in  std_logic;
        line: out unsigned(8 downto 0);
        column: out unsigned(9 downto 0);
        hsync,vsync: out std_logic );
end anie_vgasync;

architecture vgasync of anie_vgasync is

 signal ha,hn: unsigned(9 downto 0);
 signal va,vn: unsigned(8 downto 0);
 signal hrst,vrst: std_logic:='0';

begin

process(clk,clk2,hrst,vrst)
begin
if rising_edge(clk) then
 if clk2='0' then
  if hrst='1' then
   ha <= (others => '0');
   if vrst='1' then va <= (others => '0');
   else va <= vn; end if;
  else ha <= hn; end if;
 end if;
end if;
end process;

hn <= ha+1; hrst <= '1' when (ha=799 or rst='1') else '0';
vn <= va+1; vrst <= '1' when (va=520 or rst='1') else '0';

hsync <= '0' when ha<96 else '1';
vsync <= '0' when va<2 else '1';

column <= to_unsigned(640,10) when (ha<144 or ha>783) else 783 - ha;
line <= to_unsigned(480,9) when (va<31 or va>510) else 510 - va;

end vgasync;
```

s3e_vgaman.vhd

```
entity anie_vgaman is
 generic ( dwl: natural:=9 );
 port ( clk,clk1s,vga_tg,srst: in std_logic;
        line: in unsigned(8 downto 0);
        dref,dvar,dout: in signed(dwl-1 downto 0);
        column: in unsigned(9 downto 0);
        sel: in std_logic_vector(1 downto 0);
        vga: out  std_logic_vector(2 downto 0) );
```

```
9  end anie_vgaman;
10
11 architecture vgaman of anie_vgaman is
12
13 component anie_vgademo
14  port ( clk: in std_logic;
15         sel: in std_logic_vector(1 downto 0);
16         line: in unsigned(8 downto 0);
17         column: in unsigned(9 downto 0);
18         vga: out std_logic_vector(2 downto 0) );
19 end component;
20
21 component anie_vgadata
22  generic ( dwl: natural:=9 );
23  port ( clk,vga_tg,srst: in std_logic;
24         line: in unsigned(8 downto 0);
25         dref,dvar,dout: in signed(dwl-1 downto 0);
26         column: in unsigned(9 downto 0);
27         vga: out std_logic_vector(2 downto 0) );
28 end component;
29
30  signal ivga,dem,yu: std_logic_vector(2 downto 0);
31
32 begin
33
34 DEMO: anie_vgademo port map ( clk1s, sel, line, column, dem );
35 DATA: anie_vgadata generic map ( dwl )
36 port map ( clk, vga_tg, srst, line, dref, dvar, dout, column, yu );
37
38 ivga <= yu when sel="00" else dem;
39 vga <= (others=>'0') when (column=640 or line=480) else ivga;
40
41 end vgaman;
```

s3e_vgademo.vhd

```
1  entity anie_vgademo is
2  port ( clk: in std_logic;
3         sel: in std_logic_vector(1 downto 0);
4         line: in unsigned(8 downto 0);
5         column: in unsigned(9 downto 0);
6         vga: out std_logic_vector(2 downto 0) );
7  end anie_vgademo;
8
9  architecture vgademo of anie_vgademo is
10
11  signal cv, ch: std_logic_vector(2 downto 0);
12  signal v, h: std_logic_vector(4 downto 0);
13  signal grid,aind,aindn: unsigned(2 downto 0);
14
15 begin
```

```
16
17 v(0) <= '1' when (column<110) else '0';
18 v(1) <= '1' when (column<215) else '0';
19 v(2) <= '1' when (column<320) else '0';
20 v(3) <= '1' when (column<425) else '0';
21 v(4) <= '1' when (column<530) else '0';
22
23 h(0) <= '1' when (line<80) else '0';
24 h(1) <= '1' when (line<160) else '0';
25 h(2) <= '1' when (line<240) else '0';
26 h(3) <= '1' when (line<320) else '0';
27 h(4) <= '1' when (line<400) else '0';
28
29
30 cv <= "001" when v(0)='1' else "100" when v(1)='1' else
31       "101" when v(2)='1' else "010" when v(3)='1' else
32       "011" when v(4)='1' else "110";
33
34 ch <= "001" when h(0)='1' else "100" when h(1)='1' else
35       "101" when h(2)='1' else "010" when h(3)='1' else
36       "011" when h(4)='1' else "110";
37
38 process(clk)
39 begin
40 if rising_edge(clk) then
41  if aind=6 then aind <= "001"; else aind <= aindn; end if;
42 end if;
43 end process;
44
45 aindn <= aind+1;
46 grid <= aind + unsigned(cv) + unsigned(ch);
47
48 with sel select
49 vga <= cv when "01", ch when "10", std_logic_vector(grid) when "11",
50        (others=>'0') when others;
51
52 end vgademo;
```

s3e_vgadata.vhd

```
1 entity anie_vgadata is
2  generic ( dwl: natural:=9 );
3  port ( clk,vga_tg,srst: in std_logic;
4         line: in unsigned(8 downto 0);
5         dref,dvar,dout: in signed(dwl-1 downto 0);
6         column: in unsigned(9 downto 0);
7         vga: out std_logic_vector(2 downto 0) );
8 end anie_vgadata;
9
10 architecture vgadata of anie_vgadata is
11
```

```
 constant lmax: natural:=234;

component anie_vgashift
 generic ( lwl: natural:=9;
           lm:  natural:=479;
           cwl: natural:=10;
           cm:  natural:=639 );
 port ( clk,tg,srst: in std_logic;
        line,d: in unsigned(lwl-1 downto 0);
        column: in unsigned(cwl-1 downto 0);
        v: out std_logic );
end component;

component anie_vgasimpgrid
 port ( line: in unsigned(8 downto 0);
        column: in unsigned(9 downto 0);
        v: out std_logic );
end component;

component anie_vgasignshift
 generic ( cwl: natural:=10;
           cm:  natural:=639 );
 port ( clk,tg,srst: in std_logic;
        d: in std_logic_vector(2 downto 0);
        column: in unsigned(cwl-1 downto 0);
        q: out std_logic_vector(2 downto 0) );
end component;

 signal g,qr,qv,qo: std_logic;
 signal dsign,qsign,dvga,smask,svga: std_logic_vector(2 downto 0);
 signal vref,vvar,vout,hfref,hfvar,hfout,tline,bline: unsigned(7 downto 0);

begin

dsign <= dref(dwl-1)&dvar(dwl-1)&dout(dwl-1);

vref <= resize(unsigned(abs(dref)),dwl-1);
vvar <= resize(unsigned(abs(dvar)),dwl-1);
vout <= resize(unsigned(abs(dout)),dwl-1);

hfref <= to_unsigned(lmax,8) when vref>lmax else vref;
hfvar <= to_unsigned(lmax,8) when vvar>lmax else vvar;
hfout <= to_unsigned(lmax,8) when vout>lmax else vout;

tline <= resize(line-244,8); bline <= resize(line-1,8);

SREF: anie_vgashift generic map ( dwl-1, lmax, 10, 639 )
port map ( clk, vga_tg, srst, tline, hfref, column, qr );
SVAR: anie_vgashift generic map ( dwl-1, lmax, 10, 639 )
port map ( clk, vga_tg, srst, tline, hfvar, column, qv );
SOUT: anie_vgashift generic map ( dwl-1, lmax, 10, 639 )
```

```
63 port map ( clk, vga_tg, srst, bline, hfout, column, qo );
64 SGRID: anie_vgasimpgrid port map ( line, column, g);
65 SIGN: anie_vgasignshift generic map ( 10, 639 )
66 port map ( clk,vga_tg,srst,dsign,column,qsign );
67
68 dvga <= (qr & qv & '0') when line>240 else ("00"&qo);
69 vga <= (others => '1') when dvga="000" and g='1' else svga;
70 smask <= dvga and qsign;
71 svga <= dvga or smask(1 downto 0)&smask(2);
72
73 end vgadata;
```

s3e_vgashift.vhd

```
1 entity anie_vgashift is
2  generic ( lwl: natural:=9;
3            lm:  natural:=479;
4            cwl: natural:=10;
5            cm:  natural:=639 );
6  port ( clk,tg,srst: in std_logic;
7         line,d: in unsigned(lwl-1 downto 0);
8         column: in unsigned(cwl-1 downto 0);
9         v: out std_logic );
10 end anie_vgashift;
11
12 architecture vgashift of anie_vgashift is
13
14  constant cmb: std_logic_vector(cwl-1 downto 0):=std_logic_vector(to_unsigned(cm
       ,cwl));
15
16  type shift_type is array (integer range cm downto 0) of std_logic_vector(lwl-1
       downto 0);
17  signal shift: shift_type;
18
19  type st_type is (init,clear,do_shift,waiting);
20  signal sta, stn: st_type;
21
22  signal ca,cn: unsigned(cwl+1 downto 0);
23  signal caddra: unsigned(cwl-1 downto 0);
24  signal addra,addrb: integer range 0 to cm;
25  signal da,qa,pa: std_logic_vector(lwl-1 downto 0);
26  signal qb,pb: unsigned(lwl-1 downto 0);
27  signal cea,load: std_logic;
28  signal comp: std_logic_vector(2 downto 0);
29
30 begin
31
32 process(sta,tg,ca)
33 begin
34 case sta is
35  when init =>
```

```
36   if tg='1' then stn <= clear; else stn <= init; end if;
37   cn <= (others=>'0');
38
39  when clear =>
40   if ca=unsigned('0'&cmb&'1') then
41        stn <= do_shift; cn <= (others=>'0');
42   else stn <= clear; cn <= ca+1; end if;
43
44  when do_shift =>
45   if ca=unsigned(cmb&'1'&'1') then
46        stn <= waiting; cn <= (others=>'0');
47   else stn <= do_shift; cn <= ca+1; end if;
48
49  when waiting =>
50   if tg='1' then stn <= do_shift; else stn <= waiting; end if;
51   cn <= (others=>'0');
52
53  when others => stn <= init; cn <= (others=>'0');
54 end case;
55 end process;
56
57 with sta select cea <= '1' when clear, ca(0) when do_shift, '0' when others;
58 with sta select load <= ca(0) when clear, ca(1) when do_shift, '0' when others;
59 caddra <= ca(cwl downto 1) when sta=clear else ca(cwl+1 downto 2);
60 addra <= to_integer(unsigned(caddra));
61 da <= (others=>'0') when sta=clear else std_logic_vector(d) when addra=0 else pa
       ;
62
63 process(clk)
64 begin
65 if rising_edge(clk) then
66  if srst='1' then
67   sta <= init; ca <= (others=>'0');
68   pb <= (others=>'0'); qb <= (others=>'0');
69  else
70   sta <= stn;  ca <= cn;
71   pb <= qb; qb <= unsigned(shift(addrb));
72  end if;
73  if cea='1' then
74   if load='1' then shift(addra) <= da; end if;
75   pa <= qa; qa <= shift(addra);
76  end if;
77 end if;
78 end process;
79
80 addrb <= to_integer(column) when column=0 else to_integer(column-1);
81 comp(2) <= '1' when line=qb or line=pb else '0';
82 comp(0) <= '1' when line<qb else '0';
83 comp(1) <= '1' when line<pb else '0';
84
85 with sta select
```

```
86 v <= (comp(0) xor comp(1)) or comp(2) when waiting, '0' when others;
87
88 end vgashift;
```

s3e_vgasimpgrid.vhd

```
1  entity anie_vgasimpgrid is
2   port ( line: in unsigned(8 downto 0);
3          column: in unsigned(9 downto 0);
4          v: out std_logic );
5  end anie_vgasimpgrid;
6
7  architecture vgasimpgrid of anie_vgasimpgrid is
8
9   signal l: std_logic_vector(10 downto 0);
10  signal c: std_logic_vector(7 downto 0);
11
12 begin
13
14 l(1) <= '1' when line=479 else '0';
15 l(2) <= '1' when line=420 else '0';
16 l(3) <= '1' when line=361 else '0';
17 l(4) <= '1' when line=302 else '0';
18 l(5) <= '1' when line=243 else '0';
19 l(6) <= '1' when line=236 else '0';
20 l(7) <= '1' when line=177 else '0';
21 l(8) <= '1' when line=118 else '0';
22 l(9) <= '1' when line=59 else '0';
23 l(10) <= '1' when line=0 else '0';
24
25 l(0) <= ((l(1) or l(2)) or (l(3) or l(4))) or
26         ((l(5) or l(6)) or (l(7) or l(8))) or (l(9) or l(10));
27
28 c(1) <= '1' when column=639 else '0';
29 c(2) <= '1' when column=519 else '0';
30 c(3) <= '1' when column=419 else '0';
31 c(4) <= '1' when column=319 else '0';
32 c(5) <= '1' when column=219 else '0';
33 c(6) <= '1' when column=119 else '0';
34 c(7) <= '1' when column=0 else '0';
35
36 c(0) <= ((c(1) or c(2)) or (c(3) or c(4))) or
37         ((c(5) or c(6)) or c(7));
38
39 v <= (l(0) or c(0)) when (line<237 or line>242) else '0';
40
41 end vgasimpgrid;
```

s3e_vgasignshift.vhd

```
1  entity anie_vgasignshift is
```

```
 generic ( cwl: natural:=10;
           cm:  natural:=639 );
 port ( clk,tg,srst: in std_logic;
        d: in std_logic_vector(2 downto 0);
        column: in unsigned(cwl-1 downto 0);
        q: out std_logic_vector(2 downto 0) );
end anie_vgasignshift;

architecture vgasignshift of anie_vgasignshift is

 constant cmb: std_logic_vector(cwl-1 downto 0):=std_logic_vector(to_unsigned(cm
     ,cwl));

 type shift_type is array (integer range cm downto 0) of std_logic_vector(2
     downto 0);
 signal shift: shift_type;

 type st_type is (init,clear,do_shift,waiting);
 signal sta, stn: st_type;

 signal ca,cn: unsigned(cwl+1 downto 0);
 signal caddra: unsigned(cwl-1 downto 0);
 signal addra,addrb: integer range 0 to cm;
 signal da,qa,pa: std_logic_vector(2 downto 0);
 signal qb: std_logic_vector(2 downto 0);
 signal cea,load: std_logic;

begin

process(sta,tg,ca)
begin
case sta is
 when init =>
  if tg='1' then stn <= clear; else stn <= init; end if;
  cn <= (others=>'0');

 when clear =>
  if ca=unsigned('0'&cmb&'1') then
       stn <= do_shift; cn <= (others=>'0');
  else stn <= clear; cn <= ca+1; end if;

 when do_shift =>
  if ca=unsigned(cmb&'1'&'1') then
       stn <= waiting; cn <= (others=>'0');
  else stn <= do_shift; cn <= ca+1; end if;

 when waiting =>
  if tg='1' then stn <= do_shift; else stn <= waiting; end if;
  cn <= (others=>'0');

 when others => stn <= init; cn <= (others=>'0');
```

```
51 end case;
52 end process;
53
54 with sta select cea <= '1' when clear, ca(0) when do_shift, '0' when others;
55 with sta select load <= ca(0) when clear, ca(1) when do_shift, '0' when others;
56 caddra <= ca(cwl downto 1) when sta=clear else ca(cwl+1 downto 2);
57 addra <= to_integer(unsigned(caddra));
58 da <= (others=>'0') when sta=clear else std_logic_vector(d) when addra=0 else pa
       ;
59
60 process(clk)
61 begin
62 if rising_edge(clk) then
63  if srst='1' then
64       sta <= init; ca <= (others=>'0'); qb <= (others=>'0');
65  else sta <= stn;  ca <= cn; qb <= shift(addrb); end if;
66  if cea='1' then
67   if load='1' then shift(addra) <= da; end if;
68   pa <= qa; qa <= shift(addra);
69  end if;
70 end if;
71 end process;
72
73 addrb <= to_integer(column) when column=0 else to_integer(column-1);
74
75 with sta select q <= qb when waiting, (others => '0') when others;
76
77 end vgasignshift;
```

s3e_btndeb.vhd

```
1 entity anie_btndeb is
2  port ( clk: in std_logic;
3         btn: in std_logic_vector(4 downto 0);
4         btndeb: out std_logic_vector(4 downto 0) );
5 end anie_btndeb;
6
7 architecture btndeb of anie_btndeb is
8
9 component anie_debounce
10  port ( clk,p_in: in  std_logic;
11         p_out: out std_logic );
12 end component;
13
14  signal ndeb,edeb,sdeb,wdeb,xdeb: std_logic_vector(2 downto 0);
15
16 begin
17
18 DEBN: anie_debounce port map ( clk,btn(0),ndeb(0) );
19 DEBE: anie_debounce port map ( clk,btn(1),edeb(0) );
20 DEBS: anie_debounce port map ( clk,btn(2),sdeb(0) );
```

```
21 DEBW: anie_debounce port map ( clk,btn(3),wdeb(0) );
22 DEBX: anie_debounce port map ( clk,btn(4),xdeb(0) );
23
24 process(clk)
25 begin
26 if rising_edge(clk) then
27  ndeb(1) <= ndeb(0); edeb(1) <= edeb(0);
28  sdeb(1) <= sdeb(0); wdeb(1) <= wdeb(0);
29  xdeb(1) <= xdeb(0); end if;
30 end process;
31
32 ndeb(2) <= '1' when ndeb(1 downto 0)="01" else '0'; btndeb(0) <= ndeb(2);
33 edeb(2) <= '1' when edeb(1 downto 0)="01" else '0'; btndeb(1) <= edeb(2);
34 sdeb(2) <= '1' when sdeb(1 downto 0)="01" else '0'; btndeb(2) <= sdeb(2);
35 wdeb(2) <= '1' when wdeb(1 downto 0)="01" else '0'; btndeb(3) <= wdeb(2);
36 xdeb(2) <= '1' when xdeb(1 downto 0)="01" else '0'; btndeb(4) <= xdeb(2);
37
38 end btndeb;
```

s3e_norm.vhd

```
1 entity anie_norm is
2  generic ( rwl: natural:=3;
3            vwl: natural:=7;
4            iwl: natural:=7;
5            owl: natural:=13;
6            dwl: natural:=9;
7            nwl: natural:=8 );
8  port ( clk,tg,srst: in std_logic;
9         nref: in signed(rwl-1 downto 0);
10        nvar: in signed(vwl-1 downto 0);
11        nout: in signed(owl-1 downto 0);
12        pid_ref,pid_var: out signed(iwl-1 downto 0);
13        open_ref: out signed(owl-1 downto 0);
14        lcdr,lcdv,lcdo: out signed(13 downto 0);
15        vgar,vgav,vgao: out signed(dwl-1 downto 0);
16        clk_pid,vga_tg: out std_logic );
17 end anie_norm;
18
19 architecture norm of anie_norm is
20
21  constant npidr: signed(nwl-1 downto 0):=to_signed(1,nwl);  --40/40 -> 1*40=40
22  constant npidv: signed(nwl-1 downto 0):=to_signed(1,nwl); -- 40/40 -> 1*40=40
23
24  constant nopenr: signed(nwl-1 downto 0):=to_signed(102,nwl); -- 1023/40 ->
        25.5*40=4080
25
26  constant nlcdr: signed(nwl-1 downto 0):=to_signed(25,nwl); -- 1000/40 ->
        25*40=1000
27  constant nlcdv: signed(nwl-1 downto 0):=to_signed(25,nwl); -- 1000/40 ->
        25*40=1000
```

```
28  constant nlcdo: signed(nwl-1 downto 0):=to_signed(63,nwl); -- 1000/1023 ->
        0.984375*1023=1007.015625
29
30  constant nvgar: signed(nwl-1 downto 0):=to_signed(47,nwl); -- 235/40 ->
        5.875*40=235
31  constant nvgav: signed(nwl-1 downto 0):=to_signed(47,nwl); -- 235/40 ->
        5.875*40=235
32  constant nvgao: signed(nwl-1 downto 0):=to_signed(59,nwl); -- 235/1023 ->
        0.23046875*1023=235.9423828125
33
34  type norm_type is (standby,getrv,calcrv,putrv,dorv,geto,calco,puto,doo);
35  signal norm_act, norm_next: norm_type;
36
37  signal gr: signed(rwl-1 downto 0);
38  signal gv: signed(vwl-1 downto 0);
39  signal go: signed(owl-1 downto 0);
40
41  signal pr,pv: signed(iwl-1 downto 0);
42  signal lr,lv,lo: signed(13 downto 0);
43  signal vr,vv,vo: signed(dwl-1 downto 0);
44  signal wake: std_logic_vector(1 downto 0);
45
46 begin
47
48 pr <= resize(shift_right(resize(gr*npidr,rwl+nwl-1),0),iwl);
49 pv <= resize(shift_right(resize(gv*npidv,vwl+nwl-1),0),iwl);
50
51 open_ref <= resize(shift_right(resize(gr*nopenr,rwl+nwl-1),2),owl);
52
53 lr <= resize(shift_right(resize(gr*nlcdr,rwl+nwl-1),0),14);
54 lv <= resize(shift_right(resize(gv*nlcdv,vwl+nwl-1),0),14);
55 lo <= resize(shift_right(resize(go*nlcdo,owl+nwl-1),6),14);
56
57 vr <= resize(shift_right(resize(gr*nvgar,rwl+nwl-1),3),dwl);
58 vv <= resize(shift_right(resize(gv*nvgav,vwl+nwl-1),3),dwl);
59 vo <= resize(shift_right(resize(go*nvgao,owl+nwl-1),8),dwl);
60
61 process(clk,srst)
62 begin
63 if rising_edge(clk) then
64  if srst='1' then
65   wake <= (others => '0');
66   norm_act <= standby;
67
68   gr <= (others => '0');
69   gv <= (others => '0');
70   go <= (others => '0');
71
72   lcdr <= (others => '0');
73   lcdv <= (others => '0');
74   lcdo <= (others => '0');
```

```
75
76    vgar <= (others => '0');
77    vgav <= (others => '0');
78    vgao <= (others => '0');
79   else
80    wake(1) <= wake(0);
81    wake(0) <= tg;
82    norm_act <= norm_next;
83
84    if norm_next=getrv then gr <= nref; gv <= nvar; end if;
85    if norm_next=putrv then pid_ref <= pr; pid_var <= pv; end if;
86    if norm_next=geto then go <= nout; end if;
87    if norm_next=puto then
88     lcdr <= lr;
89     lcdv <= lv;
90     lcdo <= lo;
91
92     vgar <= vr;
93     vgav <= vv;
94     vgao <= vo;
95    end if;
96   end if;
97  end if;
98  end process;
99
100 process(norm_act,wake)
101 begin
102 case norm_act is
103  when standby =>
104   if wake="01" then norm_next <= getrv;
105   else norm_next <= standby; end if;
106
107  when getrv => norm_next <= calcrv;
108  when calcrv => norm_next <= putrv;
109  when putrv => norm_next <= dorv;
110  when dorv => norm_next <= geto;
111  when geto => norm_next <= calco;
112  when calco => norm_next <= puto;
113  when puto => norm_next <= doo;
114  when doo => norm_next <= standby;
115 end case;
116 end process;
117
118 clk_pid <= '1' when norm_act=dorv else '0';
119 vga_tg <= '1' when norm_act=dorv or norm_act=doo else '0';
120
121 end norm;
```

anie_input.vhd

```
1  entity anie_input is
2   generic ( count_wl: natural:=16;
3             var_wl: natural:=5 );
4   port ( clk,clk_adq,rst,sa,sb,smooth: in std_logic;
5          encsel: in unsigned(1 downto 0);
6          var: out signed(var_wl-1 downto 0) );
7  end anie_input;
8
9  architecture input of anie_input is
10
11 component anie_trigger
12  port ( clk,sa,sb,smooth: in std_logic;
13         encsel: in unsigned(1 downto 0);
14         tg_e,tg_d: out std_logic);
15 end component;
16
17 component anie_udcounter
18  generic (wl: natural:=1);
19  port ( clk,srst,ce,dir: in std_logic;
20         count: out std_logic_vector(wl-1 downto 0));
21 end component;
22
23 component anie_adq
24  generic ( count_wl: natural:=16;
25            var_wl: natural:=5 );
26  port ( clk,clk_adq,rst,dir,smooth: in std_logic;
27         count: in std_logic_vector(count_wl-1 downto 0);
28         var: out signed(var_wl-1 downto 0) );
29 end component;
30
31  signal tg_dir: std_logic;
32  signal enc: std_logic_vector(2 downto 0);
33  signal enc_count: std_logic_vector(count_wl-1 downto 0);
34
35 begin
36
37 --VAR trigger coding selection and DIR detection
38 TRG: anie_trigger port map ( clk,sa,sb,smooth,encsel,enc(0),tg_dir );
39
40 --VAR counter enable
41 --rising_edge detection on selected encoder signal
42 process(clk) begin if rising_edge(clk) then enc(1) <= enc(0); end if; end
       process;
43 enc(2) <= '1' when enc(1 downto 0)="01" else '0';
44
```

```
45 VARC: anie_udcounter generic map ( count_wl )
46 port map ( clk,rst,enc(2),tg_dir,enc_count );
47
48 -- VAR sampling overflow, underflow and stop detection
49 -- correction & compensation
50 ADQ: anie_adq generic map ( count_wl,var_wl )
51 port map ( clk,clk_adq,rst,tg_dir,smooth,enc_count,var );
52
53 end input;
```

anie_trigger.vhd

```
1 entity anie_trigger is
2  port ( clk,sa,sb,smooth: in std_logic;
3         encsel: in unsigned(1 downto 0);
4         tg_e,tg_d: out std_logic);
5 end anie_trigger;
6
7 architecture trigger of anie_trigger is
8
9  signal sin,sx: std_logic_vector(1 downto 0);
10 signal s: std_logic_vector(3 downto 0);
11 signal dir_int: std_logic:='0';
12
13 begin
14
15 sin <= sb & sa;
16
17 sx_filter: process(clk)
18 begin
19 if rising_edge(clk) then
20  case sin is
21   when "00" => sx(0) <= '0';   sx(1) <= sx(1);
22   when "01" => sx(0) <= sx(0); sx(1) <= '0';
23   when "10" => sx(0) <= sx(0); sx(1) <= '1';
24   when "11" => sx(0) <= '1';   sx(1) <= sx(1);
25   when others => sx(0) <= sx(0); sx(1) <= sx(1);
26  end case;
27 end if;
28 end process;
29
30 s(1 downto 0) <= sx when (encsel=3 and smooth='1') else sb&sa;
31
32 process(clk)
33 begin
34 if rising_edge(clk) then
35  s(3 downto 2) <= s(1 downto 0);
36  if s(0)='1' and s(2)='0' then dir_int <= s(1); -- rising_edge(sa) -> sb sets
       DIR
37  else dir_int <= dir_int; end if;
38 end if;
```

```
39 end process;
40
41 -- X1: rising_edge(sa)
42 -- X2: rising_edge(sa),falling_edge(sa)
43 -- X4: rising_edge(sa or sb),falling_edge(sa or sb)
44 tg_e <= (s(0) and (not s(2))) when encsel=1 else
45         (s(0) xor s(2)) when encsel=2 else
46         (s(3) xor s(1)) or (s(2) xor s(0)) when encsel=3 else
47         '0';
48
49 tg_d <= dir_int when encsel=3 else '0';
50
51 end trigger;
```

anie_adq.vhd

```
1  entity anie_adq is
2   generic ( count_wl: natural:=16;
3             var_wl: natural:=5);
4   port ( clk,clk_adq,rst,dir,smooth: in std_logic;
5          count: in std_logic_vector(count_wl-1 downto 0);
6          var: out signed(var_wl-1 downto 0) );
7  end anie_adq;
8
9  architecture adq of anie_adq is
10
11  signal count_act,count_pre: unsigned(count_wl-1 downto 0);
12  signal diff,plimit,nlimit,cor: signed(count_wl downto 0);
13  signal dir_int,gates,sel,vf: std_logic_vector(1 downto 0);
14  signal comp: std_logic_vector(2 downto 0);
15  signal sum,tvar,z0var,z1var,z2var: signed(var_wl-1 downto 0);
16  signal zen: std_logic;
17
18 begin
19
20 --limits for overflow/underflow correction
21 plimit(count_wl) <= '0'; plimit(count_wl-1 downto 0) <= (others => '1');
22 nlimit(count_wl) <= '1'; nlimit(count_wl-1 downto 0) <= (0 => '1',others => '0')
       ;
23
24 process(clk_adq)
25 begin
26 if rising_edge(clk_adq) then
27  count_pre <= count_act;
28  count_act <= unsigned(count); --get sample
29  dir_int(1) <= dir_int(0);
30  dir_int(0) <= dir; --get DIR
31 end if;
32 end process;
33
34 diff <= signed('0'&count_act)-signed('0'&count_pre); --calculate diff
```

```
35
36 comp(0) <= '1' when diff>0 else '0';
37 comp(1) <= '1' when diff=0 else '0';
38 comp(2) <= comp(1) or (dir_int(1) xor dir_int(0)); --stop -> diff=0 or DIR
       change
39 gates(0) <= dir_int(0) and comp(0); --underflow -> diff>0 and DIR=1
40 gates(1) <= dir_int(0) nor comp(0); --overflow -> diff<0 and DIR=0
41 sel(1) <= comp(2) or gates(1);
42 sel(0) <= comp(2) or gates(0);
43
44 cor <= nlimit when gates(0)='1' else plimit; --limits for underflow&overflow
       correction
45 sum <= resize((diff+cor)+1,var_wl); --measurement correction & 0-trough
       compensation
46
47 with sel select
48 tvar <= resize(diff,var_wl) when "00", (others => '0') when "11", sum when
       others;
49
50 process(clk)
51 begin
52 if rising_edge(clk) then
53  if rst ='1' then
54   vf(1) <= '0';
55   z2var <= (others => '0');
56   z1var <= (others => '0');
57   z0var <= (others => '0');
58  else
59   vf(1) <= vf(0);
60   vf(0) <= clk_adq;
61   if vf="01" then
62    z2var <= z1var;
63    z1var <= z0var;
64    z0var <= tvar;
65   end if;
66  end if;
67 end if;
68 end process;
69
70 zen <= '0' when (z0var=0 or z1var=0 or z2var=0) else '1';
71
72 var <= resize(shift_right( shift_left(resize(z0var,var_wl+2),1) +
73        (resize(z1var,var_wl+1) + resize(z2var,var_wl+1)) ,2),var_wl)
74        when (smooth='1' and zen='1') else tvar;
75
76 end adq;
```

anie_output.vhd

```
1 entity anie_output is
2  generic ( dc_wl: natural:=7 );
```

```
 port ( clk_pwm,clk_hb,srst,dir: in std_logic;
        dc: in std_logic_vector(dc_wl-1 downto 0);
        dir_out,en_out: out std_logic );
end anie_output;

architecture output of anie_output is

component anie_pwm
 generic ( wl: natural:=3 );
 port ( clk,srst: in std_logic;
        dc: in std_logic_vector(wl-1 downto 0);
        pwm: out std_logic );
end component;

component anie_hbridge
 port ( clk,srst,dir_in,en_in: in  STD_LOGIC;
        dir_out,en_out: out  STD_LOGIC);
end component;

  signal pwm_en: std_logic;

begin

PWM: anie_pwm generic map (dc_wl) port map ( clk_pwm,srst,dc,pwm_en );
HBDG: anie_hbridge port map ( clk_hb,srst,dir,pwm_en,dir_out,en_out );

end output;
```

anie_pwm.vhd

```
entity anie_pwm is
 generic ( wl: natural:=3 );
 port ( clk,srst: in std_logic;
        dc: in std_logic_vector(wl-1 downto 0);
        pwm: out std_logic );
end anie_pwm;

architecture pwm of anie_pwm is

component anie_counter
 generic (wl: natural:=3);
 port ( clk,ce,srst: in std_logic;
        limit: in std_logic_vector(wl-1 downto 0);
        count: out std_logic_vector(wl-1 downto 0);
        tc: out std_logic);
end component;

 signal pwm_count,max: std_logic_vector(wl-1 downto 0);

begin

```

```
22 PWMC: anie_counter generic map (wl) port map ( clk,'1',srst,max,pwm_count,open);
23
24 max <= (others => '1');
25 pwm <= '0' when unsigned(dc)=0 else '1' when (unsigned(pwm_count)<=unsigned(dc))
       else '0';
26
27 end pwm;
28
29 --     0   50 75 100 %
30 --     00 01 10 11   dc
31 --00   0   1   1   1
32 --01   0   1   1   1
33 --10   0   0   1   1
34 --11   0   0   0   1
35 --count
```

anie_hbridge.vhd

```
1 entity anie_hbridge is
2  port ( clk,srst,dir_in,en_in: in std_logic;
3         dir_out,en_out: out std_logic);
4 end anie_hbridge;
5
6 architecture hbridge of anie_hbridge is
7
8  type hbridge_type is (fw,dis_en,ch_dir);
9  signal hbga,hbgn,hbgnc: hbridge_type;
10
11 signal dir_ch,dir_ipre,act,pre,actc: std_logic;
12
13 begin
14
15 process(clk,srst)
16 begin
17 if rising_edge(clk) then
18  dir_ipre <= dir_in; pre <= act;
19  if srst='1' then hbga <= dis_en; else hbga <= hbgn; end if;
20 end if;
21 end process;
22
23 dir_ch <= dir_ipre xor dir_in;
24 actc <= pre when dir_ch='1' else dir_in;
25 hbgnc <= dis_en when dir_ch='1' else fw;
26
27 process(hbga,dir_ch,pre,dir_in)
28 begin
29 case hbga is
30  when fw =>      act <= actc; hbgn <= hbgnc;
31  when dis_en => act <= pre;  hbgn <= ch_dir;
32  when ch_dir => act <= actc; hbgn <= hbgnc;
33  when others => act <= pre;  hbgn <= dis_en;
```

```
34 end case;
35 end process;
36
37 dir_out <= act;
38 en_out <= en_in when hbga=fw else '0';
39
40 end hbridge;
```

anietiny_bl.vhd

```
1  entity anietiny_bl is
2   generic ( var_wl: natural:=7; --signed -> i_wl-norm_wl+1 <= i_wl
3             ref_wl: natural:=7; --signed -> i_wl-norm_wl+1 <= i_wl
4             norm_wl: natural:=8; --signed
5
6             i_wl: natural:=7; --signed -> i_wl and o_wl max 14 for correct LCD
                  visualization
7             o_wl: natural:=11; --signed -> i_wl + k_wl + irem_wl +1 for full
                  resolution
8             k_wl: natural:=5; --signed
9             irem_wl: natural:=1; --antiwindup-sat
10
11            pbp: integer:=0; --kp binary point
12            ibp: natural:=2; --ki binary point
13            dbp: natural:=-3; --kd binary point
14            minbp: integer:=0 ); --  min(pbp,dbp,ibp)
15   port ( clk,rst,dir,en,smooth: in std_logic;
16          rot: in std_logic_vector (1 downto 0);
17              btn: in std_logic_vector (2 downto 0);
18          j4o: out std_logic_vector (1 downto 0);
19          j4i: in std_logic_vector (1 downto 0);
20          led : out  STD_LOGIC_VECTOR (7 downto 0);
21          LCD_E,LCD_RS,LCD_RW: out std_logic;
22          SF_D: out std_logic_vector(3 downto 0) );
23 end anietiny_bl;
24
25 architecture anietiny of anietiny_bl is
26
27  constant kp: signed(k_wl-1 downto 0):=to_signed(13,k_wl); --Kp
28  constant ki: signed(k_wl-1 downto 0):=to_signed(3,k_wl); --Ki * T / 2
29  constant kd: signed(k_wl-1 downto 0):=to_signed(15,k_wl); --2 * Kd / T
30
31 component anie_norm
32  generic ( rwl: natural:=3;
33            vwl: natural:=7;
34            iwl: natural:=7;
35            owl: natural:=13;
36            nwl: natural:=8 );
37  port ( clk,tg,srst: in std_logic;
38         nref: in signed(rwl-1 downto 0);
39         nvar: in signed(vwl-1 downto 0);
```

```
40          nout: in signed(owl-1 downto 0);
41          pid_ref,pid_var: out signed(iwl-1 downto 0);
42          open_ref: out signed(owl-1 downto 0);
43          lcdr,lcdv,lcdo: out signed(13 downto 0);
44          clk_pid: out std_logic );
45  end component;
46
47  component anie_timing
48   port ( clk,rst: in std_logic;
49          lrst,clk_adq,clk_pwm,clk_hb: out std_logic;
50          clk10,clk160,clk12_5k,clk0_1k: out std_logic );
51  end component;
52
53  component anie_man
54   generic ( ref_wl: natural:=3 );
55   port ( clk10,clk160,clk12_5k,clk0_1k: in std_logic;
56          srst,ref_dir: in std_logic;
57          rot: in std_logic_vector(1 downto 0);
58          btn: in std_logic_vector(2 downto 0);
59          lmode: out std_logic;
60          LCD_E,LCD_RS,LCD_RW: out std_logic;
61          SF_D: out std_logic_vector(3 downto 0);
62          encsel: out unsigned(1 downto 0);
63          ref_man: out signed(ref_wl-1 downto 0);
64          lcdr,lcdv,lcdo: in signed(13 downto 0) );
65  end component;
66
67  component anie_input
68   generic ( count_wl: natural:=16;
69             var_wl: natural:=5 );
70   port ( clk,clk_adq,rst,sa,sb,smooth: in std_logic;
71          encsel: in unsigned(1 downto 0);
72          var: out signed(var_wl-1 downto 0) );
73  end component;
74
75  component anie_pid
76   generic (
77    i_wl: natural:=12;
78    o_wl: natural:=22;
79    k_wl: natural:=8;
80    irem_wl: natural:=1;
81
82    pbp: integer:=2;
83    ibp: integer:=17;
84    dbp: integer:=2;
85    minbp: integer:=2
86   );
87   port (
88    clk,ce: in std_logic;
89    srst: in  std_logic;
90    i_ref,i_feed: in signed(i_wl-1 downto 0);
```

```
91    kp,ki,kd: in signed(k_wl-1 downto 0);
92    o: out signed(o_wl-1 downto 0)
93   );
94  end component;
95
96  component anie_output
97   generic ( dc_wl: natural:=7 );
98   port ( clk_pwm,clk_hb,srst,dir: in std_logic;
99          dc: in std_logic_vector(dc_wl-1 downto 0);
100         dir_out,en_out: out std_logic );
101 end component;
102
103  signal lmode,lrst,prst,orst,rzero,clk_adq,clk_pid,clk_pwm,clk_hb: std_logic;
104  signal clk10,clk160,clk12_5k,clk0_1k: std_logic;
105  signal en_out,dir_loop,dir_out: std_logic;
106  signal encsel: unsigned(1 downto 0);
107
108  signal ref: signed(ref_wl-1 downto 0);
109  signal var: signed(var_wl-1 downto 0);
110  signal pid_ref,pid_var: signed(i_wl-1 downto 0);
111  signal pid_out,open_ref,outmux: signed(o_wl-1 downto 0);
112  signal dc_loop: std_logic_vector(o_wl-2 downto 0);
113
114  signal lcdr,lcdv,lcdo: signed(13 downto 0);
115
116 begin
117
118 NORM: anie_norm generic map ( ref_wl, var_wl, i_wl, o_wl, norm_wl )
119 port map ( clk10,clk_adq,lrst,ref,var,outmux,
120            pid_ref,pid_var,open_ref,lcdr,lcdv,lcdo,clk_pid );
121
122 TIM: anie_timing
123 port map ( clk,rst,lrst,clk_adq,clk_pwm,clk_hb,
124            clk10,clk160,clk12_5k,clk0_1k );
125
126 MAN: anie_man generic map ( ref_wl )
127 port map ( clk10,clk160,clk12_5k,clk0_1k,
128            lrst,dir,rot,btn,lmode,LCD_E,LCD_RS,LCD_RW,SF_D,
129            encsel,ref,lcdr,lcdv,lcdo );
130 rzero <= '1' when pid_ref=0 else '0';
131
132 INPUT: anie_input generic map ( 16,var_wl )
133 port map ( clk160,clk_adq,lrst,j4i(0),j4i(1),smooth,encsel,var );
134
135 prst <= orst or lmode;
136 PID: anie_pid generic map ( i_wl,o_wl,k_wl,irem_wl,pbp,ibp,dbp,minbp )
137 port map ( clk_pid,en,prst,pid_ref,pid_var,kp,ki,kd,pid_out );
138
139 orst <= lrst or (not en) or rzero;
140 OUTPUT: anie_output generic map ( o_wl-1 )
141 port map ( clk_pwm,clk_hb,orst,dir_loop,dc_loop,dir_out,en_out );
```

```
142
143 j4o(0) <= not dir_out;
144 j4o(1) <= en_out when en='1' else '0';
145
146 --LOOP
147 outmux <= (others => '0') when en='0' else pid_out when lmode='0' else open_ref;
148 dc_loop <= std_logic_vector(resize(unsigned(abs(outmux)),o_wl-1));
149 dir_loop <= outmux(o_wl-1);
150
151 ---- INFO
152 led(7) <= encsel(1);
153 led(6) <= encsel(0);
154 led(5) <= j4i(1);
155 led(4) <= j4i(0);
156 led(3) <= en_out;
157 led(2) <= dir_out;
158 led(1) <= lmode;
159 led(0) <= '0';
160
161 end anietiny;
```

anie_bl.vhd

```
1  entity anie_bl is
2   generic ( var_wl: natural:=7; --signed -> i_wl-norm_wl+1 <= i_wl
3             ref_wl: natural:=7; --signed -> i_wl-norm_wl+1 <= i_wl
4             norm_wl: natural:=8; --signed
5             vdata_wl: natural:=9; --signed
6
7             i_wl: natural:=7; --signed -> i_wl and o_wl max 14 for correct LCD
                  visualization
8             o_wl: natural:=11; --signed -> i_wl + k_wl + irem_wl +1 for full
                  resolution
9             k_wl: natural:=5; --signed
10            irem_wl: natural:=1; --antiwindup-sat
11
12            pbp: integer:=0; --kp binary point
13            ibp: integer:=2; --ki binary point
14            dbp: integer:=-3; --kd binary point
15            minbp: integer:=0 ); --  min(pbp,dbp,ibp)
16   port ( clk,rst,dir,en,smooth: in std_logic;
17          rot: in std_logic_vector (1 downto 0);
18                  btn: in std_logic_vector (4 downto 0);
19          j4o: out std_logic_vector (1 downto 0);
20          j4i: in std_logic_vector (1 downto 0);
21          led : out  STD_LOGIC_VECTOR (7 downto 0);
22          LCD_E,LCD_RS,LCD_RW: out std_logic;
23          SF_D: out std_logic_vector(3 downto 0);
24        hsync,vsync: out std_logic;
25                  vga: out std_logic_vector(2 downto 0) );
26 end anie_bl;
```

```
27
28 architecture bl of anie_bl is
29
30  constant kp: signed(k_wl-1 downto 0):=to_signed(13,k_wl); --Kp
31  constant ki: signed(k_wl-1 downto 0):=to_signed(3,k_wl); --Ki * T / 2
32  constant kd: signed(k_wl-1 downto 0):=to_signed(15,k_wl); --2 * Kd / T
33
34 component anie_norm
35  generic ( rwl: natural:=3;
36            vwl: natural:=7;
37            iwl: natural:=7;
38            owl: natural:=13;
39                          dwl: natural:=9;
40                          nwl: natural:=8 );
41  port ( clk,tg,srst: in std_logic;
42         nref: in signed(rwl-1 downto 0);
43                   nvar: in signed(vwl-1 downto 0);
44                   nout: in signed(owl-1 downto 0);
45                   pid_ref,pid_var: out signed(iwl-1 downto 0);
46                   open_ref: out signed(owl-1 downto 0);
47         lcdr,lcdv,lcdo: out signed(13 downto 0);
48         vgar,vgav,vgao: out signed(dwl-1 downto 0);
49         clk_pid,vga_tg: out std_logic );
50 end component;
51
52 component anie_timing
53   port ( clk,rst: in std_logic;
54          lrst,clk_adq,clk_pwm,clk_hb,clk1s: out std_logic;
55          clk0,clk2,clk10,clk160,clk12_5k,clk0_1k: out std_logic );
56 end component;
57
58 component anie_man
59  generic ( ref_wl: natural:=3;
60            vdata_wl: natural:=9 );
61  port ( clk0,clk2,clk10,clk160,clk12_5k,clk0_1k,clk1s,vga_tg: in std_logic;
62         srst,ref_dir: in std_logic;
63         rot: in std_logic_vector(1 downto 0);
64                   btn: in std_logic_vector(4 downto 0);
65                   lmode: out std_logic;
66              LCD_E, LCD_RS, LCD_RW: out std_logic;
67           SF_D: out std_logic_vector(3 downto 0);
68                   hsync,vsync: out std_logic;
69                   vga: out std_logic_vector(2 downto 0);
70                   encsel: out unsigned(1 downto 0);
71         ref_man: out signed(ref_wl-1 downto 0);
72         lcdr,lcdv,lcdo: in signed(13 downto 0);
73         vgar,vgav,vgao: in signed(vdata_wl-1 downto 0) );
74 end component;
75
76 component anie_input
77   generic ( count_wl: natural:=16;
```

```
78              var_wl: natural:=5 );
79   port ( clk,clk_adq,rst,sa,sb,smooth: in std_logic;
80           encsel: in unsigned(1 downto 0);
81           var: out signed(var_wl-1 downto 0) );
82 end component;
83
84 component anie_pid
85  generic (
86   i_wl: natural:=12;
87   o_wl: natural:=22;
88   k_wl: natural:=8;
89   irem_wl: natural:=1;
90
91   pbp: integer:=2;
92   ibp: integer:=17;
93   dbp: integer:=2;
94   minbp: integer:=2
95  );
96  port (
97   clk,ce: in std_logic;
98   srst: in  std_logic;
99   i_ref,i_feed: in signed(i_wl-1 downto 0);
100  kp,ki,kd: in signed(k_wl-1 downto 0);
101  o: out signed(o_wl-1 downto 0)
102 );
103 end component;
104
105 component anie_output
106   generic ( dc_wl: natural:=7 );
107   port ( clk_pwm,clk_hb,srst,dir: in std_logic;
108           dc: in std_logic_vector(dc_wl-1 downto 0);
109                       dir_out,en_out: out std_logic );
110 end component;
111
112  signal lmode,lrst,prst,orst,rzero,clk_adq,clk_pid,clk_pwm,clk_hb,clk1s,vga_tg:
       std_logic:='0';
113  signal clk0,clk2,clk10,clk160,clk12_5k,clk0_1k: std_logic:='0';
114  signal en_out,dir_loop,dir_out: std_logic:='0';
115  signal encsel: unsigned(1 downto 0);
116
117  signal ref: signed(ref_wl-1 downto 0);
118  signal var: signed(var_wl-1 downto 0);
119  signal pid_ref,pid_var: signed(i_wl-1 downto 0);
120  signal pid_out,open_ref,outmux: signed(o_wl-1 downto 0);
121  signal dc_loop: std_logic_vector(o_wl-2 downto 0);
122
123  signal lcdr,lcdv,lcdo: signed(13 downto 0);
124  signal vgar,vgav,vgao: signed(vdata_wl-1 downto 0);
125
126 begin
127
```

```
128 NORM: anie_norm
129  generic map ( ref_wl, var_wl, i_wl, o_wl, vdata_wl, norm_wl )
130  port map ( clk10,clk_adq,lrst,ref,var,outmux,
131             pid_ref,pid_var,open_ref,lcdr,lcdv,lcdo,vgar,vgav,vgao,
132             clk_pid,vga_tg );
133
134 TIM: anie_timing
135  port map ( clk,rst,lrst,clk_adq,clk_pwm,clk_hb,clk1s,
136             clk0,clk2,clk10,clk160,clk12_5k,clk0_1k );
137
138 MAN: anie_man
139  generic map ( ref_wl, vdata_wl )
140  port map ( clk0,clk2,clk10,clk160,clk12_5k,clk0_1k,clk1s,vga_tg,
141             lrst,dir,rot,btn,lmode,LCD_E,LCD_RS,LCD_RW,SF_D,hsync,vsync,vga,
142             encsel,ref,lcdr,lcdv,lcdo,vgar,vgav,vgao );
143 rzero <= '1' when pid_ref=0 else '0';
144
145 INPUT: anie_input
146  generic map ( 16,var_wl )
147  port map ( clk160,clk_adq,lrst,j4i(0),j4i(1),smooth,encsel,var );
148
149 prst <= orst or lmode;
150 PID: anie_pid
151  generic map ( i_wl,o_wl,k_wl,irem_wl,pbp,ibp,dbp,minbp )
152  port map ( clk_pid,en,prst,pid_ref,pid_var,kp,ki,kd,pid_out );
153
154 orst <= lrst or (not en) or rzero;
155 OUTPUT: anie_output
156   generic map ( o_wl-1 )
157   port map ( clk_pwm,clk_hb,orst,dir_loop,dc_loop,dir_out,en_out );
158
159 j4o(0) <= not dir_out;
160 j4o(1) <= en_out when en='1' else '0';
161
162 --LOOP
163 outmux <= (others => '0') when en='0' else pid_out when lmode='0' else open_ref;
164 dc_loop <= std_logic_vector(resize(unsigned(abs(outmux)),o_wl-1));
165 dir_loop <= outmux(o_wl-1);
166
167 ---- INFO
168 led(7) <= encsel(1);
169 led(6) <= encsel(0);
170 led(5) <= j4i(1);
171 led(4) <= j4i(0);
172 led(3) <= en_out;
173 led(2) <= dir_out;
174 led(1) <= lmode;
175 led(0) <= '0';
176
177 end anie;
```

anie-tiny

```
=========================================================================
*                      Synthesis Options Summary                        *
=========================================================================

---- Source Options
Top Module Name                    : anietiny_bl
Automatic FSM Extraction           : YES
FSM Encoding Algorithm             : Auto
Safe Implementation                : No
FSM Style                          : LUT
RAM Extraction                     : Yes
RAM Style                          : Auto
ROM Extraction                     : Yes
Mux Style                          : Auto
Decoder Extraction                 : YES
Priority Encoder Extraction        : Yes
Shift Register Extraction          : YES
Logical Shifter Extraction         : YES
XOR Collapsing                     : YES
ROM Style                          : Auto
Mux Extraction                     : Yes
Resource Sharing                   : YES
Asynchronous To Synchronous        : NO
Multiplier Style                   : Auto
Automatic Register Balancing       : No

---- Target Options
Add IO Buffers                     : YES
Global Maximum Fanout              : 500
Add Generic Clock Buffer(BUFG)     : 24
Register Duplication               : YES
Slice Packing                      : YES
Optimize Instantiated Primitives   : NO
Use Clock Enable                   : Yes
Use Synchronous Set                : Yes
Use Synchronous Reset              : Yes
Pack IO Registers into IOBs        : Auto
Equivalent register Removal        : YES

---- General Options
```

```
Optimization Goal                  : Speed
Optimization Effort                : 1
Keep Hierarchy                     : No
Netlist Hierarchy                  : As_Optimized
RTL Output                         : Yes
Global Optimization                : AllClockNets
Read Cores                         : YES
Write Timing Constraints           : NO
Cross Clock Analysis               : NO
Hierarchy Separator                : /
Bus Delimiter                      : <>
Case Specifier                     : Maintain
Slice Utilization Ratio            : 100
BRAM Utilization Ratio             : 100
Verilog 2001                       : YES
Auto BRAM Packing                  : NO
Slice Utilization Ratio Delta      : 5

=========================================================================
*                        Advanced HDL Synthesis                         *
=========================================================================

Macro Statistics
# FSMs                                              : 6
# RAMs                                              : 1
 32x8-bit dual-port distributed RAM                  : 1
# ROMs                                              : 3
 16x8-bit ROM                                        : 1
 32x16-bit ROM                                       : 1
 4x5-bit ROM                                         : 1
# Multipliers                                       : 7
 11x8-bit registered multiplier                      : 1
 7x8-bit registered multiplier                       : 3
 8x5-bit multiplier                                  : 1
 9x5-bit multiplier                                  : 2
# Adders/Subtractors                                : 31
 10-bit adder                                        : 1
 14-bit adder                                        : 3
 15-bit adder                                        : 3
 16-bit addsub                                       : 1
 17-bit adder                                        : 1
 17-bit subtractor                                   : 1
 2-bit adder                                         : 1
 2-bit addsub                                        : 1
 3-bit adder                                         : 2
 5-bit adder                                         : 7
 6-bit adder                                         : 2
```

```
 6-bit addsub                                       : 1
 7-bit adder                                        : 1
 8-bit adder                                        : 1
 8-bit subtractor                                   : 1
 9-bit adder                                        : 3
 9-bit subtractor                                   : 1
# Counters                                          : 5
 10-bit up counter                                  : 1
 14-bit up counter                                  : 1
 4-bit up counter                                   : 1
 5-bit up counter                                   : 1
 7-bit up counter                                   : 1
# Registers                                         : 341
 Flip-Flops                                         : 341
# Comparators                                       : 13
 10-bit comparator lessequal                        : 1
 15-bit comparator greater                          : 2
 15-bit comparator less                             : 2
 17-bit comparator greater                          : 1
 2-bit comparator greater                           : 2
 2-bit comparator not equal                         : 1
 5-bit comparator equal                             : 3
 6-bit comparator greater                           : 1
# Multiplexers                                      : 8
 1-bit 16-to-1 multiplexer                          : 1
 1-bit 4-to-1 multiplexer                           : 5
 7-bit 4-to-1 multiplexer                           : 1
 8-bit 8-to-1 multiplexer                           : 1
# Xors                                              : 4
 1-bit xor2                                         : 4

=========================================================================
*                            Final Report                               *
=========================================================================

Design Statistics
# IOs                              : 29

Cell Usage :
# BELS                             : 1448
#      GND                         : 1
#      INV                         : 28
#      LUT1                        : 63
#      LUT2                        : 151
#      LUT2_D                      : 1
#      LUT3                        : 222
#      LUT3_D                      : 2
```

```
#      LUT4                        : 492
#      LUT4_D                      : 3
#      LUT4_L                      : 1
#      MUXCY                       : 216
#      MUXF5                       : 75
#      MUXF6                       : 1
#      OR2                         : 1
#      OR3                         : 1
#      VCC                         : 1
#      XORCY                       : 189
# FlipFlops/Latches                : 412
#      FD                          : 67
#      FDE                         : 42
#      FDR                         : 89
#      FDRE                        : 205
#      FDRS                        : 1
#      FDRSE                       : 1
#      FDS                         : 7
# RAMS                             : 16
#      RAM16X1D                    : 16
# Shift Registers                  : 3
#      SRL16                       : 3
# Clock Buffers                    : 7
#      BUFG                        : 7
# IO Buffers                       : 29
#      IBUF                        : 11
#      IBUFG                       : 1
#      OBUF                        : 17
# DCMs                             : 2
#      DCM_SP                      : 2
# MULTs                            : 7
#      MULT18X18SIO                : 7
```

anie

```
=========================================================================
*                      Synthesis Options Summary                        *
=========================================================================

---- Source Options
Top Module Name                   : anie_bl
Automatic FSM Extraction          : YES
FSM Encoding Algorithm            : Auto
Safe Implementation               : No
FSM Style                         : LUT
RAM Extraction                    : Yes
```

```
RAM Style                          : Auto
ROM Extraction                     : Yes
Mux Style                          : Auto
Decoder Extraction                 : YES
Priority Encoder Extraction        : Yes
Shift Register Extraction          : YES
Logical Shifter Extraction         : YES
XOR Collapsing                     : YES
ROM Style                          : Auto
Mux Extraction                     : Yes
Resource Sharing                   : YES
Asynchronous To Synchronous        : NO
Multiplier Style                   : Auto
Automatic Register Balancing       : No

---- Target Options
Add IO Buffers                     : YES
Global Maximum Fanout              : 500
Add Generic Clock Buffer(BUFG)     : 24
Register Duplication               : YES
Slice Packing                      : YES
Optimize Instantiated Primitives   : NO
Use Clock Enable                   : Yes
Use Synchronous Set                : Yes
Use Synchronous Reset              : Yes
Pack IO Registers into IOBs        : Auto
Equivalent register Removal        : YES

---- General Options
Optimization Goal                  : Speed
Optimization Effort                : 1
Keep Hierarchy                     : No
Netlist Hierarchy                  : As_Optimized
RTL Output                         : Yes
Global Optimization                : AllClockNets
Read Cores                         : YES
Write Timing Constraints           : NO
Cross Clock Analysis               : NO
Hierarchy Separator                : /
Bus Delimiter                      : <>
Case Specifier                     : Maintain
Slice Utilization Ratio            : 100
BRAM Utilization Ratio             : 100
Verilog 2001                       : YES
Auto BRAM Packing                  : NO
Slice Utilization Ratio Delta      : 5
```

```
=========================================================================
*                        Advanced HDL Synthesis                         *
=========================================================================

Macro Statistics
# FSMs                                                : 8
# RAMs                                                : 5
 32x8-bit dual-port distributed RAM                   : 1
 640x3-bit dual-port block RAM                        : 1
 640x8-bit dual-port block RAM                        : 3
# ROMs                                                : 3
 16x8-bit ROM                                         : 1
 32x16-bit ROM                                        : 1
 4x5-bit ROM                                          : 1
# Multipliers                                         : 10
 11x8-bit registered multiplier                       : 2
 7x8-bit registered multiplier                        : 5
 8x5-bit multiplier                                   : 1
 9x5-bit multiplier                                   : 2
# Adders/Subtractors                                  : 48
 10-bit adder                                         : 2
 10-bit subtractor                                    : 4
 12-bit adder                                         : 4
 14-bit adder                                         : 3
 15-bit adder                                         : 3
 16-bit addsub                                        : 1
 17-bit adder                                         : 1
 17-bit subtractor                                    : 1
 2-bit adder                                          : 1
 2-bit addsub                                         : 1
 3-bit adder                                          : 4
 5-bit adder                                          : 7
 6-bit adder                                          : 2
 6-bit addsub                                         : 1
 7-bit adder                                          : 1
 8-bit adder                                          : 4
 8-bit subtractor                                     : 3
 9-bit adder                                          : 4
 9-bit subtractor                                     : 1
# Counters                                            : 10
 10-bit up counter                                    : 2
 14-bit up counter                                    : 1
 2-bit up counter                                     : 1
 3-bit up counter                                     : 1
 4-bit up counter                                     : 1
 5-bit up counter                                     : 1
 7-bit up counter                                     : 2
```

```
 9-bit up counter                                  : 1
# Registers                                        : 479
 Flip-Flops                                        : 479
# Comparators                                      : 49
 10-bit comparator greater                         : 1
 10-bit comparator less                            : 7
 10-bit comparator lessequal                       : 1
 15-bit comparator greater                         : 2
 15-bit comparator less                            : 2
 17-bit comparator greater                         : 1
 2-bit comparator greater                          : 2
 2-bit comparator not equal                        : 1
 5-bit comparator equal                            : 3
 6-bit comparator greater                          : 1
 7-bit comparator equal                            : 2
 8-bit comparator equal                            : 6
 8-bit comparator greater                          : 3
 8-bit comparator less                             : 6
 9-bit comparator greater                          : 3
 9-bit comparator less                             : 8
# Multiplexers                                     : 9
 1-bit 16-to-1 multiplexer                         : 1
 1-bit 4-to-1 multiplexer                          : 5
 3-bit 4-to-1 multiplexer                          : 1
 7-bit 4-to-1 multiplexer                          : 1
 8-bit 8-to-1 multiplexer                          : 1
# Xors                                             : 7
 1-bit xor2                                        : 7

=========================================================================
*                            Final Report                               *
=========================================================================

Design Statistics
# IOs                              : 36

Cell Usage :
# BELS                             : 2405
#      GND                         : 1
#      INV                         : 40
#      LUT1                        : 133
#      LUT2                        : 278
#      LUT2_D                      : 1
#      LUT3                        : 346
#      LUT3_D                      : 2
#      LUT4                        : 811
#      LUT4_D                      : 3
```

```
#      LUT4_L                    : 1
#      MUXCY                     : 369
#      MUXF5                     : 111
#      MUXF6                     : 1
#      OR2                       : 1
#      OR3                       : 1
#      VCC                       : 1
#      XORCY                     : 305
# FlipFlops/Latches              : 583
#      FD                        : 71
#      FDE                       : 69
#      FDR                       : 167
#      FDRE                      : 262
#      FDRS                      : 5
#      FDRSE                     : 1
#      FDS                       : 8
# RAMS                           : 20
#      RAM16X1D                  : 16
#      RAMB16_S4_S4              : 1
#      RAMB16_S9_S9              : 3
# Shift Registers                : 5
#      SRL16                     : 5
# Clock Buffers                  : 9
#      BUFG                      : 9
# IO Buffers                     : 36
#      IBUF                      : 13
#      IBUFG                     : 1
#      OBUF                      : 22
# DCMs                           : 2
#      DCM_SP                    : 2
# MULTs                          : 10
#      MULT18X18SIO              : 10
```

BILBOKO INDUSTRIA INGENIARITZA TEKNIKOKO UNIBERTSITATE ESKOLA

INDUSTRIA INGENIARITZA TEKNIKOA: INDUSTRIA ELEKTRONIKA

KARRERA AMAIERAKO PROIEKTUA

Anie - 4. Dokumentua: Eranskinak [ingurunea]

IKASLEAREN DATUAK	**ZUZENDARIAREN DATUAK**
	Koldo Basterretxea Oyarzabal koldo.basterretxea@ehu.es Teknologia Elektronikoa
SIN.: DATA: 2013ko apirilaren 12a	SIN.: DATA:

Gaien Aurkibidea

4A.
Pierre St. Martin

Produktu industrial eta teknologikoekin sinbiosian bizi garen heinean, energia elektromagnetikoa eta mekanikoa bihurtzen, moldatzen eta darabilten makinak edonon aurkitu ditzakegu; eta fidagarritasuna bermatzeko ezinbestekoa dugu makinek momenturo espero diren irteerak eta lan erregimenak mantentzea.

Helburu hori erdiesteko beharrekoak aztertzen dituen arloan (kontrola eta erregulazioa deritzona) makinari, berarekin elkar eragiteko irakurketa eta elikadura gailuekin batera, planta deritzo. Honek duen erantzun dinamikoaren arabera, agindutako balioak emateko behar duen denbora, egonkortu arteko tarte iragankorrean izango duen erantzuna, eta egoera egonkorrean izango duen zehaztasuna aldatu egingo dira. Erantzun hori ez du makinaren egiturak baldintzatuko soilik; fabrikazio prozedurek, denbora eta erabileraren erruz agertutako deribek, funtzionamendu erregimena edo kargaren aldaketa eta beste hainbat faktorek bultzatutako asaldurek erantzuna balio izendatutik eta teorikotik aldenduko dute.

Badaude egitura dela eta irteeran zehaztasuna beti mantentzen duten plantak, urratsez urratseko motorra esaterako. Horrek begizta irekian kontrol fidagarria izatea ahalbidetzen du, baina aldi berean, beste makina batzuekin aldaratuz, egituraren konplexutasunak mantenua garestitzen du. Hainbat aplikaziotarako, aldiz, nahiz eta funtzionamendu egonkorra izan, ez dugu begizta irekian fidagarria den makinarik. Azkenik, baditugu begizta irekian ezegonkorrak diren eta ondorioz kontrola ezinezkoa duten plantak, hala nola lebitadore magnetikoak. Azkeneko hauek egonkortzeko eta egonkorrak direnen erantzuna hobetzeko begizta itxiko sistemak erabiltzen dira. Sistema hauetan plantaren irteerak kontrol logika berrelikatzen du eta berau baliatuz kontrolagailuak erabiltzaileak eskatutako erreferentziarekiko diferentzia kalkulatu eta zerora bultzatzen du irteera aldatuz. Kontrolagailuaren egitura eta parametroek erantzun iragankor zein egonkorra baldintzatzen dute, aplikazioaren beharretara moldatuz.

Erregulazioa eta kontrola edonon aurkitu ditzakegu egun, ordenagailuen oinarri diren disko gogorrak birarazten dituzten motorretatik hasita eta elektrizitatearen banaketa kudeatzen duten instalazioak barne; garraioen abiadura finkatzaileak, edozein prozesu industrialeko zinta zein biltegien likido maila neurtzaileak, mota guztietako robotak eta bukaezina litzatekeen zerrenda gehitu dezakegularik.

Era berean, kalitatearen kudeaketan prozesuak berrikusi eta optimizatzeko ezinbestekoa dugu plantaren lan erregimena eta baldintzak ezagutzea (kontsumoa, esaterako). Datuak biltzeko erabiltzailearekin interfazeak izan behar ditu kontrol sistemak.

Ingeniari batek, beraz, zentzu batean zein bestean, inoiz halako sistema erabili beharko du: kontrolagailuaren egitura diseinatzea ez bada, planta aztertuz diseinurako baldintzak ezartzea, bere parametroak ezartzea edo integratua duen makinaren bat erabiltzean. Hamaika esparrutan ezinbestekoa izanik, teknikoak gerturatzeko kritikoa dugu erakargarria den ikasketa eta garapen ingurunea.

4Aa. Helburua eta hedadura

Azaldutakoekin bat, hau dugu proiektu honen helburu nagusia:

Hurbiltzeko bide ezberdinak jarraitu dituzten ingeniari eta teknikoentzat kontrol, erregulazio eta komunikazio esparruetan ikasketa eta garapenerako oinarrizko eta abstrakzio maila baxuan dokumentatutako ingurune praktiko, ireki eta askea sortzea.

4Aaa. Diseinurako baldintzak

Bigarren mailako helburu gisa jokatzen duten ezarritako diseinurako baldintzek helburua nagusia mugatzen dute:

- **Modularra:** planta aniztasuna eta ondorioz inplementatu daitezkeen sistemen arteko ezberdintasunak direla eta, aldaketetan denborarik ez galtzeko argi banatuta agertu behar dira atalak eta ahal den heinean elkarren arteko data fluxua mugatzea. Honek ere, ikasketa eta trebakuntzarako izanik, banakako aztertzea ahalbidetzen du, sistema osotasunean ezagutzeko beharrik gabe.
- **Real-time:** ezegonkorrak diren plantetan bereziki, kontrol begiztak egoera aldaketa guztiei erantzuna behar denean emango duela ziurtatu behar da. Hortaz, determinismoa azaltzen duen "benetako denbora" kontzeptuak ezarritako mugak bete behar ditu.
- **Txertatua:** lekua txikia eta energia iturriak murritzak diren esparruetan kontrola gauzatu behar denean, erabilitako baliabideek berebiziko garrantzia dute eta kontsumoak zeharo baldintzatzen du produktuaren bideragarritasuna. Erabilera zehatz bakoitzerako sistemaren egitura bereziki egokitzean, beharrezkoak diren baliabideak erabiltzen ditu, eta soilik beharrezkoak direnak.
- **Autonomoa:** beste edozein sistemarekin komunikazioa ziurra edo finkoa ez bada, ez da onargarria kontrol begizta gaitasunik gabe uztea. Beraz, isolaturik ibiltzeko eta erabiltzeko gai izan behar da.
- **Hiztuna:** behar izatekotan maila altuagoan ezarritako beste sistema batek, edo urrunean dagoenak, kontrol begiztaren egoera ikusi eta moldatzeko aukera txertatua izateak ezartzen dituen mugen osagarria da.
- **Erabilerraza eta argia:** doiketen arabera finkatutako egituraren funtzioak osatzeko deskribapenak parametrizatua izan behar du. Honek, trebakuntza denbora aurrezteaz gain, hardwarea sakonean ezagutzen ez duenari ere diseinuan moldaketa txikiak egiteko aukera ematen dio.

4Ab. Berezitasunak agindutako banaketa

Nafarroako iparraldean, Zuberoa, Frantzia eta Huescarekin mugan, bada Belagua izeneko Erronkariko Ibarraren bazterra. Mendian gora, Laskunerako[1] bidean, leize, haitzulo eta lurrazpiko hamaika ganbarek osatzen dute Pierre St. Martin sistema miresgarria. Mila hirurehun metro baino gehiagoko sakonera du (munduko sakonenetakoa), ehun kilometrotik gora batzen dituzte galerien luzerek. Espeleologian nahitaezko erreferentzia diren Haroun Tazieff, Marcel Loubens eta Norbert Casteret frantsesek, beste zenbaitekin, aurreko mendean bertan aurkitu zuten Europako ganbara handiena, 200x120 metro inguruko azalera duena eta sabaia 100 metroko altuerara.

Azalaren gainean, era berean ikaragarria den Larrako *karst*a[2], txundigarri bezain beldurgarria. Erdian, lur azpia non hasten den itxuragabetzen duen kaosak inguraturik, Anie mendia (2507m); nahiz eta mugetatik at egon, euskaldunontzat, sakratutzat jo izan dugun heinean, badu izen berezia: Auñamendi, toponimiak aginduta *ahuntzen mendia*. Hego-mendebaldean, kilometro eskas batera, nahiko zabalagoa den eta garaiera galtzen duen Añelarra (2357m) dugu nafarren mugaren adierazgarri.

Pierre St. Martinetik abiatuta Aunñamendira zein Añelarrara norabidean, urrunetik *karst*ean zeharreko bidea igartzeko ikuspuntu bikaina ematen du Arlas[3] (2044m) tontorrak.

[1] Franstsesez eta ofizialki *Lescun*

[2] *Kareharrizko goi ordokia* esan nahi duen Alemanaren mailegua dugu, jatorria Italia-Eslovenia inguruko *Carso/Kras* eskualdean duena.

[3] Ipar euskal herrietako mendirik garaiena eta euskal herri guztietan hirugarrena.

Sinbiosia gogora, antagonismoaren aldarrikapenean banaketa geologikoarekin analogia eginez, garatu nahi den *Pierre St. Martin* ikasketa eta garapenerako inguruneak kontzeptualki lau atal nagusi ditu, bakoitzak proiektu bana osatzen duelarik, besteekin lotutakoa baina independentea. Zenbaki eta hitzak ulertzen dituzten ezizenen banaketa ezagutza esparruaren arabera egin da, nahiz eta ingeniari baten heziketan nahastea maiz ezinbestekoa izan.

- **Anie** → hardware garapena

  Kontrol begizta osatzen duten gailuak identifikatu, modelatu, garatu, deskribatu, doitu, batu eta praktikan kontrola gauzatzea.

- **Añelarra** → kontrol sistema aurreratuak

  Hardware eta softwarea aldi berean diseinatuz (*hard-soft co-design*) kontrol begiztaren baldintza teknikoak hobetzea.

- **Arlas** → informatika aplikazioak

  Urruneko gailu baten bitartez kontrol begiztaren informazioa jasotzeko xedez, edo honen funtzionamendua kudeatzeko, software aplikazioaren garapena.

- **Larra** → (tele)komunikazioak

  Anie, Añelarrak eta Arlasek elkarren artean komunikatzeko protokoloa eta bakoitzean integratzeko argibideak.

Badago lotura bistaratzen duen zeharkako bosgarren proiektu bat, informazioaren sostenguak baldintzatzen dituenak. Ezagutzak hedatzeko bidea behar duela sinetsiz, eta nahi baino gehiagotan bide horretan hainbat dirusari ordaintzeko beharra edo bereziki ezarritako oztopoak aurkitu izanagatik, *Pierre St. Martin* inguruneari dagozkion dokumentazio eta iturri guztiak, ahal den heinean, estandarrak diren

formatuetan gordeko dira, ez denboraren poderioz hedatuenetan. Besteak beste, eskuartean duzun dokumentua egiteko LaTeX[4] baliatuz Bilboko IITUEen Karrera Amaierako Proiektuak aurkezteko jarraitu beharreko maketazio argibideak betetzen dituen txantiloia egin da.[5].

4Ac. Aurrekariak

Karrera Amaierako Proiektua izanik, ezinbestean azken urteotan hainbat esparrutan jorratutakoak, eta guztietan ikasitakoa, biltzen ditu honek. Horrenbesteren adierazgarri sintetikoa denez, zenbaitekin zuzeneko lotura argiak ditu. Hona hemen *Pierre St. Martin* eta *Anie* proiektuen bidea eraiki dutenetan jatorri adierazgarrienak:

2005 - Vumeter / LCD / ReoBus

MODPC.com orrialde eta taldeko kideen argibideak jarraituz (*Nahiko* izeneko erabiltzailearenak bereziki), beste zenbait orrialdetan[6] aurkitutakoak oinarri, eta Ikastolako Teknologia irakasleak eta eskolazainak emandako material eta erraztasunei eskerrak, eskuz egindako insoladorarekin ordenagailua ikuskatu eta kudeatzeko plaka diseinatu, egin eta muntatzea. Disko gogorraren jarduera adierazteko *LM3914* ZIak agindutako 20 LED, LPT ataka medio informazioa aurkezteko *Nokia 3310* sakelakoaren pantaila, eta haizagailu eta argiak kudeatzeko kommutadore eta erresistentzia aldakorretan oinarritutako zirkuitua biltzen dituena [2].

Trebakuntzak:

- Araututako hezkuntzak ikusi ez baina ahalbidetzen dituen zeharkako aukerei onura ateratzea.
- Zirkuitu analogiko, digital eta mistoei lehen hurreratzea.
- Zirkuitu elektronikoak egiteko diseinu, fabrikazio eta muntaketa prozedura ezagutzea.

2008 - ZT Praktiketako Kalkuluak (ZTPK)

*Diego Cano Lagneaux*en laguntzarekin eta *Informatikaren Oinarriak* ikasgaian jorratutakoetan sakontzeko, *C* lengoaian eta interfaze grafikorako *curses* liburutegia baliatuz idatzitako programa[7], *Zirkuituen Teoria* ikasgaiko praktiketan ikusitako saiakuntzen emaitzekin burutu beharreko kalkuluak egiten dituena.

Trebakuntzak:

- Software garapenerako *sourceforge* plataforman gordetako bertsio kontrol sistema erabiltzea.
- Programazio modularra eta funtzioen banaketa.
- Funtzio zehatzak burutzeko liburutegiak bilatu eta erabiltzea.
- Interfaze grafikoa eta menuak garatzea.

[4] http://itsas.ehu.es/workgroups/latex

[5] 4B. eranskinean ITSASeko web gunean LaTeX lan taldearen *Euskaraz* atalean eskuragarri dagoen txantiloiaren txostena aurkitu daiteke.

[6] eurobotics.org, pcpaudio.com, hardcore-modding.com, shilmar.com eta maxoverclocking.com

[7] ztpk.sourceforge.net

2009 - Ordenagailu Haizagailuak Kontrolatu eta Ikusteko Sistema (OHKIS)

Eragingailu Logiko Programagarriak Dituzten Sistema Digitalak ikasgaian 2008-2009 ikasturtean *Aitor Martin*ekin batera garatutako proiektua: Spartan3E Starter Kit txartela erabilita lau ordenagailu haizagailu LCD eta VGA bitartez ikuskatu eta begizta irekian PWM irteerak ezartzea [3][8].

Trebakuntzak:

- Sistema digitalak VHDL bitartez deskribatzea.
- FPGAk VHDL bitartez konfiguratzea.
- *Spartan3E Starter Kit* txartela erabiltzea.

*Anie*ren hainbat osagai, hala nola, LCD, VGA, pultsadoreak, etab. kudeatzeko logiken VHDL deskribapenak bertatik hartu dira, berritu eta garbitua.

2010 - ohkis-gtk

*Diego Cano Lagneaux*en laguntzarekin, *OHKIS*en baldintza teknikoak oinarritzat hartuta *Ruby* lengoaian idatzitako programa[9], *gtk+* liburutegiarekin egindako eta bereizitako interfaze grafikoa duena eta datuak irudikatzeko *gnuplot* erabiltzen duena.

Trebakuntzak:

- Objektuetara zuzendutako programazioari lehen hurreratzea.
- Bereizitako fitxategietan deskribatutako interfazeak erabiltzea.
- Programa ezberdinen exekuzioa koordinatzea.

Hau dugu *Arlas*en kontzeptuzko oinarria.

2010 - Bicicleta μControlador (BμC)

2009-2010 ikasturtean *Ferrol*eko *EUP*en egindako *Informática Industrial* ikasgaian *Borja Iglesias Rozas*ekin batera garatutako proiektua: LCD eta teklatu matriziala baliatuz bizikleta baten funtzionamendua ikusteko *C517A* mikrokontrolagailuan oinarritutako eta *C* lengoaian idatzitako sistema [45].

Trebakuntzak:

- Mikrokontrolagailuen arkitektura eta *C* lengoaian programatzea.
- Gailu berdina kudeatzeko C eta VHDL lengoaien eta dagozkien sistemen konparaketa.
- Komentatutako kodea erabiliz *doxygen*ekin dokumentatzea.

2011 - Acher

Ordenadore Pertsonalaren Erabilera Paneleko Instrumentazioan ikasgaian 2010-2011 ikasturtean garatutako serie komunikazio bitartez LED matrizea kudeatzeko sistema [46]. *EHUko Software Libre Talde*ko[10] LaTeXeko lan taldean bildutakoei esker, *Bulego Teknikoa* ikasgaian izen bereko lana prestatu nuen,

[8] *Aitor Martin*ek ikasturte berean izen bereko Karrera Amaierako Proiektua [44] defendatu zuen.
[9] sourceforge.net/p/ohkis/code/HEAD/tree/ohkis/ohkis_gtk
[10] itsas.ehu.es

proiektuak aurkezteko argibideak jarraituz [47].

Trebakuntzak:

- *C* lengoaian arkitektura ezberdina eta dagozkion tresna berriak erabiltzea.
- Zirkuitu mistoa garatzea eta muntatzea.
- Ordenagailu eta sistema digital independentea komunikatzea.
- LaTeX bidez proiektuak aurkezteko txantiloia.

Hau dugu *Larra*ren kontzeptuzko oinarria. Aldi berean, txantiloi horren hobekuntza da dokumentu honetan erabilitakoa.

2012 - Control de velocidad de un motor CC: NI Labview

Sistemas Digitales en la Medida y Control de Procesos Industriales ikasgaian 2011-2012 ikasturtean *Luis Ranero Santisteban* eta *Iñigo Sarramian Olmos*ekin batera egindako lana: *National Instruments*en *Labview* softwarea baliatuz benetako denboran lan egiten duen PID kontrolagailuan oinarritutako begizta osatu eta 6*v*ko motorra PWM bitartez kudeatzea [4].

Trebakuntzak:

- Ordenagailuan *DAQ*an oinarritutako *real-time* kontrol begizta osatzea.
- *encoder* inkrementala eta *H-zubia* erabiltzea.

4B.

Karrera Amaierako Proiektuak idazteko txantiloia

2007 urtean zehar *Iñaki Silanes*ek, **U**niversidad del **P**aís **V**asco / **E**uskal **H**erriko **U**nibertsitateko[11] **ITSAS** Software Libre Taldeko kideak, LaTeX eta *OpenDocument* formatuetan Unibertsitatean gaztelaraz, euskaraz zein ingelesez Karrera Amaierako Proiektuak zein Doktorego Tesiak aurkezteko txantiloiak eskaintzeko helburuarekin *Plantillas para Proyecto de Fin de Carrera* lan taldea[12] osatu zuen.

2010 urtean *Digna González* eta *Unai Martinez*ek lan talde berrian[13] *Iñaki Silanes*en lana *LaTeX* erabiltzeko hainbat argibide, erreferentzia, aurkezpen eta abarrekin bateratu zuten eta material bera baliatuz zenbait ikastaro eman.

Idazleak, **Bi**lboko **I**ndustria **I**ngeniaritza **T**eknikoko **U**nibertsitate **E**skolan[14] Karrera Amaierako Proiektua euskaraz idazteko orduan eskuragarri zeuden txantiloiek premia[15] guztiak asetzen ez zituztenez, aipatutako lan taldeetan bildutakoak oinarri, eskuartean duzun txantiloi berria egin du. 4B.1. taulak *Iñaki Silanes*ek eskainitakoekiko ezberdintasun nagusiak biltzen ditu.

	Hizkuntza	**Formatua**	**Klasea**	
Iñaki Silanes	EU ES EN	LaTeX OpenDocument	*itsas_pfc.cls*	*book* oinarri
Unai Martinez	EU	LaTeX	*report*	*config* fitxategietan moldatuta

4B.1. Taula: Iñaki Silanes*en txantiloiekin konparaketa.*

Horietaz gain, hurrengo berrikuntzak ditu honek:

- Kapitulu, atal, azpiatal, azpiazpiatal, irudi eta taulak zenbakitu eta izendatzean zenbakia azaltzen da lehenengo, puntu ordinala ondoren eta hitza azkenik.
- *babel*ek *basque* aukeratzean ezatzen duen data komandoaren ordez *gaur* sortu da.
- *Kapitulu*en izen gisa *Dokumentu* ezarri da.
- BI-IITUEko web gunean soilik DOC formatuan eskuragarri dauden txantiloiak erabili dira *Kapitulu/Dokumentu*en portadak diseinatzeko.
- Atalen goiburuak aldatu dira.
- Ikurren Zerrenda gehitu da.
- BI-IITUEko arautegiak eskatu bezala, *UNE 157001-2002* araua erreferentzia izanik banatu da edukia. Hala ere, txantiloi honek ez du araua betetzen. Karrera Amaierako Proiektuen helburu nagusia hezkuntza eta ikastea izanik, edukia aurkitzea eta dokumentuen banakako azterketa errazteko diseinuan zenbait erabaki ezberdin hartu dira:

[11] www.ehu.es
[12] itsas.ehu.es/workgroups/plantillas_proyecto_fin_de_carrera
[13] itsas.ehu.es/workgroups/latex
[14] www.industria-ingeniaritza-tekniko-bilbao.ehu.es
[15] www.industria-ingeniaritza-tekniko-bilbao.ehu.es/p229-content/eu/contenidos/normativa/euiti_bi_pfc/eu_nor_gral/normativa_gral_fin_carrera.html

- Dokumentuen ordena aldatu da eta zenbait ezabatu.
- Portadak ez daude zenbakituta.
- Orrialde, irudi, taula eta ekuazioen zenbakitzea kapitulu bakoitzean berrabiatzen da.
- Zenbakitzea 0an hasten da.
- Aurkibideen orrialdeak zenbaki erromatarrez daude adierazita.
- Eranskinen dokumentuan atalak alfabetoz izendatzen dira.
- Goiburu eta orri oinen edukiak tokiz aldatuta daude eta dokumentu, atal zein azpiatalen arabera berritzen dira.

Hau dela eta, araua betetzeko *config* karpetako fitxategietan moldaketak egin behar ditu txantiloiaren erabiltzaileak.

Emaitza zuzena izan dadin hainbat aldiz konpilatu behar da lana, hurrengo ordena jarraituz:

PDFLaTeX + BibTeX + PDFLaTeX + PDFLaTeX

Atal eta azpiataletan aldaketa asko egitean, tarteko fitxategiak edo fitxategi laguntzaileak (*.aux*, *.mtc*, *.mlf*, *.mlt*, etab.) ezabatzea komeni da, aurreko katea exekutatu baino lehen.

4Ba. Txantiloia osatzen duten fitxategiak

Txantiloia osatzen duten fitxategien egitura azaltzen da atal honetan, kokatzeko ezinbestekoak direnak adieraziz. Azterketa zehatzagorako ikus 4Bb. atala edo jo aitorpenetan adierazitako iturrietara.

main.tex
Txantiloriaren fitxategi nagusia, *document* deklaratu eta beste guztiak kargatzen dituena.

dedicatory.tex
Memoriaren portadaren hurrengo orrialde hutsean adierazten den eskaintza.

intro.tex
Memoriaren lehenengo atalaren edukia, *Sarrera*.

license.tex
Memoriaren bigarren atalaren edukia, *Lizentzia eta aitorpenak*.

state.tex
Memoriaren hirugarren atalaren edukia, *Teknikaren egoera*.

sty_titlepg.tex
Portada nagusiaren edukia.

sty_head.tex
Portada guztien goiburuaren diseinua.

sty_who.tex
Portada guztien oinaren diseinua.

symbols.tex
*Ikurren Zerrenda*ren edukia.

bibliography.bib
Erreferentzia bibliografikoak *BibTeX*en arabera.

images/

logo.png
Portada nagusian erdian agertzen den logoa.

ehu.png
Portaden goiburuan ezkerrean agertzen den logoa.

euiti.png
Portaden goiburuan eskuinean agertzen den logoa.

ychart.tikz
Ereduzko Y-grafikoa TikZ bitartez deskribatua.

config/

config.tex
Konfigurazio fitxategi nagusia, kargatzen denean lehena eta pakete guztiak kargatzeaz gain hainbat komando (ber)ezartzen dituena.

config_basque.tex
Nahiz eta *babel* paketea erabili, euskaraz hainbat gauza formatu egokian adierazi daitezen moldaketak.

config_hdr.tex
Portaden atzeko planoko diseinua (laukizuzena) eta atalaren arabera goiburu eta oinen edukia moldatzea.

config_index.tex
minitoc paketeak eskainitako funtzioetan oinarrituta *DOtls* eta *DOmtls* komandoak sortzea eta aurkibideen marjinak doitzea.

config_titles.tex
Kapitulu, atal eta azpiatalak aldatzean izenburu berriak eskuratu eta aldagai ezagunetan gordetzea.

secta/

secta_main.tex
Ereduzko atal baten fitxategi nagusia.

images/

mod_closedloop.tikz
Ereduzko irudi bat.

mod_cont_lum.tikz
Ereduzko irudia ekuazio batekin batera.

s3etiny_lcd.tikz
Beste irudi bat.

sectb/

sectb_main.tex
Ereduzko beste atal baten fitxategi nagusia

sectb_first.tex
Atalaren lehenengo edukiak dituen fitxategia.

sectb_last.tex
Atalaren azkeneko edukiak dituen fitxategia.

anie_vhdl_sat.vhd
listings paketea baliatuz aurkeztutako ereduzko VHDL kodea.

anie_vhdl_pid.vhd
listings paketea baliatuz aurkeztutako ereduzko VHDL kodea.

images/

mod_box.tikz
Ereduzko irudia TikZ eta kolore ezberdinak erabilita.

vhdl_sat.tikz
Beste bat.

anie_pwm.tikz
Beste bat (PWM sortzailea).

anie_hbridge.tikz
Beste bat (FSM).

measures/

measures_main.tex
Neurketa eta kalkuluak dokumentuaren fitxategi nagusia.

att/

att_main.tex
Eranskinak dokumentuaren fitxategi nagusia.

atta.tex
Lehenengo eranskinaren edukia.

attb.tex
Bigarren eranskinaren edukia.

attc.vhd
Hirugarren eranskinaren edukia.

m/
Hirugarren eranskinaren iturriak, *Matlab script*ak.

tune.m

save_bw.m

save_step.m

save_ts.m

cond/

cond_main.tex
Baldintzen agiria dokumentuko fitxategi nagusia.

cond_adm.tex
Baldintza administratiboak atalaren edukia.

cond_tec.vhd
Baldintza teknikoak atalaren edukia.

cond_eco.vhd
Baldintza ekonomikoak atalaren edukia.

cond_comp.tex
Osagaiak eta ezaugarriak atalaren edukia.

4Bb. Txantiloiaren oinarrizko erabilera

Txantiloiaren erabilera zuzena da, hau da, dauden fitxategietan edukia beste edozein LaTeX dokumentutan egin bezala idaztearekin nahikoa dugu. Erabilitako paketeek ezarri litzaketen mugak izan behar ditugu kontuan, eta berriren bat kargatzekotan ordenari erreparatu behar diogu.

Bete beharreko baldintza bakarra dago: goiburuetan azpiatalak ondo adierazi daitezen *titlesubsection* erabili behar da *subsection* ordez[16].

Kodea garbi mantendu eta itxurari dagozkionak ahal den heinean banaturik mantentzeko hainbat fitxategi daude *config* karpetan eta *main.tex* fitxategian zenbait komando berri agertzen dira. Jarraian hauek azalduko dira, kapitulu zein atalak moldatu, gehitu zein kentzeko prozedura adierazteko.

main.tex

```
\documentclass[a4paper,titlepage,10pt,oneside]{report}
```

report klasea dugu oinarri, alde bakarrekoa, DIN A4 formatuarekin. *article* erabili nahi izatekotan, *minitoc* paketeari dagozkion *(do)minitoc*, *(do)minilof* eta *(do)minilot* aginduen kudeaketa aldatu beharko litzateke (ikus *config.tex* eta *config_index.tex*), paketearen dokumentazioan adierazitakoen arabera. Pakete hori erabiltzen ez bada, aldaketa zuzena da.

Orriaren tamainari dagokionez, aldatzekotan kapitulu eta atalen orrietan distantziak berrikusi beharko lirateke (ikus *config_titles.tex*). Letraren tamaina aldatzean ere baliteke aldaketa txikiak somatzea.

```
\usepackage{import}
\inputfrom{./config/}{config.tex}
```

Azpikarpetatan dauden fitxategiak kargatu eta hauetan kokapen erlatiboak erabili ahal izateko *import* paketea kargatu da lehenik, eta honekin *config* karpeta barruko konfigurazio fitxategi nagusia.

```
\begin{document}

\DOpresetDOtitlepg
```

Dokumentua hasi eta berehala *config.tex* fitxategian definituta dagoen *DOpresetDOtitlepg* komandoak euskaraz aurkezpena zuzenena izan dadin beharreko komandoak exekutatzen ditu, zenbakitzearen eta gaien aurkibidearen sakontasuna ezartzen ditu, kapituluen zenbakitzea zeroan abiarazten du, *minitoc*ek eskatutakoak exekutatzen ditu, portada aurkezten du eta orrialde berri batean hasteko prestatzen du dokumentua.

[16] *GNU/Linux*en *grep* erabilita zuzenean egin dezagu bihurketa.

```
\chapter{Aurkibide orokorra} \DOtls
```

Lehenengo kapitulua, zerogarrena, *Aurkibide orokorra* dugu. *config_index.tex* fitxategian definitutako *DOtls* komandoak orrien zenbakitzea erromatarrera aldatu eta *tableofcontents*, *listoffigures* eta *listoftables* exekutatzen ditu. Sekzio bezala gehitzen ditu aurkibidera, eta baten batek orri bat baino gehiago izatekotan goiburuak bat etor daitezen ezartzen ditu. Azkenik, berriz ere aldatzen du orrien zenbakitzea arabiarrera eta 1 balioa esleitzen dio.

```
\chapter{Memoria}
\pagestyle{empty}\input{dedicatory}\pagestyle{body}
\DOmtls{\DOmtoc\DOmlof\DOmlot\DOmlos}
```

Memoriaren hasiera adierazi eta berehala, goiburu eta oinik ez dituen estiloa ezartzen da eskaintza aurkezteko, *config_hdr.tex* fitxategian definitutako eta orokorrean erabiliko den *body* estilora bueltatu baino lehen.

DOmtls komandoak, *DOtls*ek egin antzera zenbakitzea eta goiburuak moldatuz, kapituluko gaien aurkibidea (*DOmtoc*), irudien zerrenda (*DOmlof*), taulen zerrenda (*DOmlot*) edota ikurren zerrenda (*DOmlos*) aurkezten ditu. Lehenengo hirurak sortzeko *minitoc* paketeak eskainitakoak erabiltzen diren bitartean, ikurren zerrendak zuzenean *symbols.tex* fitxategiko edukia kargatzen du.

```
\include{intro}
\include{license}\label{lic}
\include{state}
```

Memoriaren atalak dituzten fitxategiak zuzenean *include* edo *input* bitartez kargatzen dira. Lehenengoa *main.tex* fitxategian baino ezin daiteke erabili, beste fitxategi guztietan *input* erabiltzen da.

```
\subincludefrom{./secta/}{secta_main}
\subincludefrom{./sectb/}{sectb_main}
```

Hainbat fitxategi dituzten atalak karpeta banatuetan gordetzen dira eta *main.tex*en fitxategi bakarra kargatzen da. Honek atal aldaketa txikiak eginez horiek banatu eta bakarrik konpilatzea ahalbidetzen du, eta egitura aldatu barik hainbat atal gehitzea.

```
\chapter{Neurketak eta kalkuluak} \DOmtls{\DOmtoc}
\subincludefrom{./measures/}{measures_main}

\chapter{Eranskinak} \DOmtls{\DOmtoc}
\attref
\subincludefrom{./att/}{att_main}
\ordref

\chapter{Baldintzen agiria} \DOmtls{\DOmtoc}
\subincludefrom{./cond/}{cond_main}

\chapter{Aurrekontua}
```

Azaldutako tresna berdinak erabilita kargatzen dira hurrengo kapituluak. Ikus daitekeenez hauetan gaien aurkibidea aurkezten da soilik. *Eranskinak* dokumentuaren edukia kargatu baino lehen adierazitako *attref* komandoak atalen zenbakitzea alfabetora aldatzen du (ikus *config_index.tex*). Dokumentuaren bukaeran berriz ere bueltatzen da hasierako aurreko konfiguraziora (*ordref*, ikus *config_basque.tex*).

```
\nocite{*}
\chapter{Bibliografia}
\bibliographystyle{ieeetr}
\fancyhead[L]{\slshape \nouppercase{\bibname}}
\bibliography{bibliography}
```

Erreferentzia guztiak (estekatutakoak eta estekatu gabekoak) aurkezteko *nocite** deitu ostean, *Bibliografia* kapitulua hasi, estiloa aukeratu, orrialde bat baino gehiago izatekotan itxura egokia aurkezteko goiburua moldatu eta *bibligraphy.bib* fitxategia kargatzen du.

```
\end{document}
```

config/config.tex

```
\usepackage[utf8]{inputenc}
```

Karaktereen kodeketa adierazteko.

```
\usepackage[spanish,basque]{babel}
\selectlanguage{basque}
```

Tituluak batez ere, eta beste hainbat aukera, lokalizatzeko. Gaztelera kargatzen da izen propioak erabiltzean $\tilde{n}$ eta azentu-markekin arazorik ez izateko.

```
\usepackage[left=3.5cm, right=1.5cm, top=2.5cm, bottom=2.5cm]{geometry}
```

Normak adierazitako marjinak ezartzeko.

```
\usepackage{graphicx}
```

Irudiak txertatzeko.

```
\usepackage[numbers]{natbib}
```

Erreferentzia bibliografikoak testuan adieraztean *[X]* itxuraz adierazi daitezen.

```
\usepackage[font=scriptsize,format=plain,labelfont=bf,textfont=it,
            justification=centerlast]{caption}
```

Irudi eta taulen oinen itxura moldatzeko.

```
\usepackage{indentfirst}
```

babel paketeak gazteleraz paragrafo bakoitzaren lehenengo lerroari ezkerreko marjina handiagoa jartzen dio, baina euskaraz ez. Honek egitera bortxatzen du, baina kontuz ibili beharko dugu irudiak eta taulak erdiratzerakoan.

```
\usepackage{multirow}
```

Tauletan zutabe edo lerro anitz hartzen dituzten gelaxkak erabiltzeko.

```
\usepackage{eurofont}
```

€ sinboloa erabiltzeko.

```
\usepackage[usenames,dvipsnames]{xcolor}
\colorlet{urlcolor}{purple!65!black}
\colorlet{ilcolor}{violet!65!black}
```

Esteketan, irudietan eta grafikoetan koloreak definitzeko aukera ugari izateko.

```
\usepackage{listings}
\lstset{
language=VHDL,
basicstyle=\color{Blue}\footnotesize\ttfamily,
commentstyle=\color{CadetBlue},
stringstyle=,
identifierstyle=\color{Black},
backgroundcolor=\color{black!10!white},
columns=fixed,
extendedchars=true,
breaklines=true,
numbers=none
}
```

Lengoaia ezberdinetan idatzitako kodea dokumentuan txertatzeko. Eredu gisa erabili den VHDL kodea aurkezteko ausazko aurkezpenaren hautaketa.

```
\usepackage{tikz,pgfplots}
\usetikzlibrary{shapes,arrows}
\usepackage{tikz-timing}
```

Irudiak, grafikoak eta kronogramak egiteko.

```
\usepackage[hyphens]{url}
\usepackage[
```

```
 bookmarks=true,
 unicode=true,
 pdftitle={Karrera Amaierako Proiektuak idazteko LaTeX txantiloia},
 pdfsubject={},
 pdfauthor={Unai Martinez Corral},
 linktoc=all,
 colorlinks=true,
 linkcolor=ilcolor,
 urlcolor=urlcolor,
 citecolor=Blue,
 plainpages=false,
]{hyperref}
```

Dokumentuko erreferentziak estekatzeko eta irteerako PDF fitxategiaren propietateak ezartzeko.

```
\parskip=2mm
```

Paragrafoen arteko tartea ezartzea.

```
\usepackage{amsmath}
\numberwithin{figure}{chapter}
\numberwithin{table}{chapter}
\numberwithin{equation}{chapter}
```

Irudi, taula eta ekuazioen zenbakitzea kapitulu bakoitzean berrabiatzeko.

```
\usepackage{etoolbox}
```

Komandoei dei egitean exekutatu baino lehen bitarteko ekintzak burutzeko.

```
\input{config_titles}
\input{config_hdr}
\input{config_basque}
\input{config_index}
```

Funtzio zehatzen konfigurazioa: portadak eta atalek goiburuak, bestelako goiburu eta oinak, euskaraz erabiltzeko hobekuntzak eta aurkibideak aurkeztea.

```
\usepackage[basque,loose]{minitoc}
%\usepackage{mtcoff}
\setcounter{minitocdepth}{4}
\setlength{\mtcindent}{0pt}
\renewcommand{\mtcfont}{\small\rm}
\renewcommand{\mtcSfont}{\small\bf}
\nomtcrule \nomlfrule \nomltrule
```

Kapitulu bakoitzean aurkibideak eta zerrendak aurkeztea ahalbidetzen duen paketea kargatzea eta hainbat parametro ezartzea. Hauen artean garrantzitsuena *setcounterminitocdepth* dugu, aurkibideek aurkeztuko duten sakontasuna adierazten baitu: 0-kapitulua, 1-atala, 2-azpiatala, 3-azpiazpiatala edo 4-paragrafoa. Besteek letra mota ezartzen dute eta zerrenden inguruko lerroak ezabatzen dituzte.

```
\newcommand{\DOpresetDOtitlepg}{
\ordref
\setcounter{secnumdepth}{3}
\setcounter{tocdepth}{1}
\addtocounter{chapter}{-1}
\dominitoc[e]
\dominilof[e]
\dominilot[e]
\input{sty_titlepg}
\clearpage\pagestyle{body}
}
```

Dokumentua hasi eta berehala *config_basque.tex* fitxategian definitutako *ordref* komandoak euskaraz elementuen izenak ondo adierazi daitezen moldaketak burutzen ditu.

Atalak zenbakitzeko erabiliko den sakontasuna adierazten da ondoren (3-azpiazpiatala), eta aurkibide nagusiak aurkeztuko duena (1-atala). Kapituluen zenbakitzea zeroan hasteko izendatuari bat kentzen zaio.

*minitoc*ek eskatutako komandoak adierazten dira, izenbururik gabe aurkezteko parametroarekin (*[e]*). *config_index.tex* fitxategian ikus daitekeenez izenburuak banaturik sortzen dira, lan osoaren estiloa mantentzeko.

Portada aurkezten da azkenik, eta berehala orrialde huts berri batean estilo orokorra ezarri.

4Bc. Lizentzia eta aitorpenak

Txantiloi hau hurrengo lizentziaren arabera eskaintzen da:

Creative Commons Attribution-ShareAlike 3.0 (CC BY-SA 3.0)

- Egin ditzakezunak:

  Banatzea Kopiatu, banatu eta hedatzea

  Moldatzea Lana egokitzea eta eratorriak egitea

  Merkataritza helburuekin erabiltzea

- Hurrengoak bete bitartean:

  Aitortzea Lanaren iturria aitortu behar da, *Unai Martinez Corral* eta *ITSAS*i erreferentzia eginez, eta `itsas.ehu.es/workgroups/latex` orrialdea aipatuz (baina lan eratorriek edo lanaren erabilerek hauen babesa dutela adierazi barik).

Berdin partekatzea Lan hau moldatu edo egokituz gero, edo lan eratorririk sortzekotan, egindakoa banatzeko honetan erabilitako lizentzia berdina erabili behar da.

creativecommons.org/licenses/by-sa/3.0/es/legalcode.eu

Aipatutako ITSASen lan taldeetako baliabideez gain, jarraian zerrendatutakoak erabili dira:

- **TeXmaker** (xm1math.net/texmaker) *Pascal Brachet*
- **BibTeX** (bibtex.org) *Oren Patashnik, Leslie Lamport, Oren Patashnik*
- LaTeX paketeak (ctan.org/pkg/):

  import *Donald Arseneau*

  inputenc *Alan Jeffrey, Frank Mittelbach*

  babel *Javier Bezos, Johannes L. Braams*

  geometry *Hideo Umeki*

  graphicx *David Carlisle*

  natbib *Patrick W. Daly, Arthur Ogawa*

  caption *Axel Sommerfeldt*

  indentfirst *Davis Carlisle*

  multirow *Piet van Oostrum, Jerry Leichter*

  amsmath *The American Mathematical Society*

  eurofont *Rowland McDonnell*

  xcolor *Uwe Kern*

  listings *Brooks Moses, Carsten Heinz*

  tikz,pgfplots *Till Tantau, Christian Feuersänger*

  tikz-timing *Martin Scharrer*

  url *Donald Arseneau*

  hyperref *Heiko Oberdiek, Sebastian Rahtz*

  etoolbox *Philipp Lehman*

  minitoc *Jean-Pierre Drucbert*

  eso-pic *Rolf Niepraschk*

  fancyhdr *Piet van Oostrum*

4C.

GNU General Public License

GNU GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

Terms and Conditions

0. Definitions.

   "This License" refers to version 3 of the GNU General Public License.

   "Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

   "The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

   To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

   A "covered work" means either the unmodified Program or a work based on the Program.

   To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

   To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

   An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

   The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

   A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

   The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel,

window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

   All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

   You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

   Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

   No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

   When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

   You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in

accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

   You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

   (a) The work must carry prominent notices stating that you modified it, and giving a relevant date.

   (b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".

   (c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

   (d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

   A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

   You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

   (a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

   (b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

(c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

(d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

(e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the

public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

(a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

(b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

(c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or

(d) Limiting the use for publicity purposes of names of licensors or authors of the material; or

(e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

(f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

   You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

   However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

   Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

   Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

   You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

    Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

    An “entity transaction” is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party’s predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

    You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

    A “contributor” is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor’s “contributor version”.

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from

those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

    Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

    The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

    Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

    If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

    Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

    THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

    IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

    If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

End of Terms and Conditions

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
<one line to give the program's name and a brief idea of what it does.>

Copyright (C) <textyear>  <name of author>

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program.  If not, see <http://www.gnu.org/licenses/>.
```

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

```
<program>  Copyright (C) <year>  <name of author>

This program comes with ABSOLUTELY NO WARRANTY; for details type ‘show w’.
This is free software, and you are welcome to redistribute it
under certain conditions; type ‘show c’ for details.
```

The hypothetical commands `show w` and `show c` should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an "about box".

You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see http://www.gnu.org/licenses/.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read http://www.gnu.org/philosophy/why-not-lgpl.html.

4D.

GNU Lesser General Public License

GNU LESSER GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

This version of the GNU Lesser General Public License incorporates the terms and conditions of version 3 of the GNU General Public License, supplemented by the additional permissions listed below.

0. Additional Definitions.

   As used herein, "this License" refers to version 3 of the GNU Lesser General Public License, and the "GNU GPL" refers to version 3 of the GNU General Public License.

   "The Library" refers to a covered work governed by this License, other than an Application or a Combined Work as defined below.

   An "Application" is any work that makes use of an interface provided by the Library, but which is not otherwise based on the Library. Defining a subclass of a class defined by the Library is deemed a mode of using an interface provided by the Library.

   A "Combined Work" is a work produced by combining or linking an Application with the Library. The particular version of the Library with which the Combined Work was made is also called the "Linked Version".

   The "Minimal Corresponding Source" for a Combined Work means the Corresponding Source for the Combined Work, excluding any source code for portions of the Combined Work that, considered in isolation, are based on the Application, and not on the Linked Version.

   The "Corresponding Application Code" for a Combined Work means the object code and/or source code for the Application, including any data and utility programs needed for reproducing the Combined Work from the Application, but excluding the System Libraries of the Combined Work.

1. Exception to Section 3 of the GNU GPL.

   You may convey a covered work under sections 3 and 4 of this License without being bound by section 3 of the GNU GPL.

2. Conveying Modified Versions.

   If you modify a copy of the Library, and, in your modifications, a facility refers to a function or data to be supplied by an Application that uses the facility (other than as an argument passed when the facility is invoked), then you may convey a copy of the modified version:

   (a) under this License, provided that you make a good faith effort to ensure that, in the event an Application does not supply the function or data, the facility still operates, and performs whatever part of its purpose remains meaningful, or

(b) under the GNU GPL, with none of the additional permissions of this License applicable to that copy.

3. Object Code Incorporating Material from Library Header Files.

The object code form of an Application may incorporate material from a header file that is part of the Library. You may convey such object code under terms of your choice, provided that, if the incorporated material is not limited to numerical parameters, data structure layouts and accessors, or small macros, inline functions and templates (ten or fewer lines in length), you do both of the following:

(a) Give prominent notice with each copy of the object code that the Library is used in it and that the Library and its use are covered by this License.

(b) Accompany the object code with a copy of the GNU GPL and this license document.

4. Combined Works.

You may convey a Combined Work under terms of your choice that, taken together, effectively do not restrict modification of the portions of the Library contained in the Combined Work and reverse engineering for debugging such modifications, if you also do each of the following:

(a) Give prominent notice with each copy of the Combined Work that the Library is used in it and that the Library and its use are covered by this License.

(b) Accompany the Combined Work with a copy of the GNU GPL and this license document.

(c) For a Combined Work that displays copyright notices during execution, include the copyright notice for the Library among these notices, as well as a reference directing the user to the copies of the GNU GPL and this license document.

(d) Do one of the following:

0) Convey the Minimal Corresponding Source under the terms of this License, and the Corresponding Application Code in a form suitable for, and under terms that permit, the user to recombine or relink the Application with a modified version of the Linked Version to produce a modified Combined Work, in the manner specified by section 6 of the GNU GPL for conveying Corresponding Source.

1) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (a) uses at run time a copy of the Library already present on the user's computer system, and (b) will operate properly with a modified version of the Library that is interface-compatible with the Linked Version.

(e) Provide Installation Information, but only if you would otherwise be required to provide such information under section 6 of the GNU GPL, and only to the extent that such information is necessary to install and execute a modified version of the Combined Work produced by recombining or relinking the Application with a modified version of the Linked Version. (If you use option 4d0, the Installation Information must accompany the Minimal Corresponding Source and Corresponding Application Code. If you use option 4d1, you must provide the Installation Information in the manner specified by section 6 of the GNU GPL for conveying Corresponding Source.)

5. Combined Libraries.

You may place library facilities that are a work based on the Library side by side in a single library together with other library facilities that are not Applications and are not covered by this License, and convey such a combined library under terms of your choice, if you do both of the following:

(a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities, conveyed under the terms of this License.

(b) Give prominent notice with the combined library that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

6. Revised Versions of the GNU Lesser General Public License.

The Free Software Foundation may publish revised and/or new versions of the GNU Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library as you received it specifies that a certain numbered version of the GNU Lesser General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that published version or of any later version published by the Free Software Foundation. If the Library as you received it does not specify a version number of the GNU Lesser General Public License, you may choose any version of the GNU Lesser General Public License ever published by the Free Software Foundation.

If the Library as you received it specifies that a proxy can decide whether future versions of the GNU Lesser General Public License shall apply, that proxy's public statement of acceptance of any version is permanent authorization for you to choose that version for the Library.

4E.

Creative Commons BY-SA-3.0 Legal Code

Creative Commons Legal Code

Attribution-ShareAlike 3.0 Unported

CREATIVE COMMONS CORPORATION IS NOT A LAW FIRM AND DOES NOT PROVIDE LEGAL SERVICES. DISTRIBUTION OF THIS LICENSE DOES NOT CREATE AN ATTORNEY-CLIENT RELATIONSHIP. CREATIVE COMMONS PROVIDES THIS INFORMATION ON AN "AS-IS" BASIS. CREATIVE COMMONS MAKES NO WARRANTIES REGARDING THE INFORMATION PROVIDED, AND DISCLAIMS LIABILITY FOR DAMAGES RESULTING FROM ITS USE.

License

THE WORK (AS DEFINED BELOW) IS PROVIDED UNDER THE TERMS OF THIS CREATIVE COMMONS PUBLIC LICENSE ("CCPL" OR "LICENSE"). THE WORK IS PROTECTED BY COPYRIGHT AND/OR OTHER APPLICABLE LAW. ANY USE OF THE WORK OTHER THAN AS AUTHORIZED UNDER THIS LICENSE OR COPYRIGHT LAW IS PROHIBITED.

BY EXERCISING ANY RIGHTS TO THE WORK PROVIDED HERE, YOU ACCEPT AND AGREE TO BE BOUND BY THE TERMS OF THIS LICENSE. TO THE EXTENT THIS LICENSE MAY BE CONSIDERED TO BE A CONTRACT, THE LICENSOR GRANTS YOU THE RIGHTS CONTAINED HERE IN CONSIDERATION OF YOUR ACCEPTANCE OF SUCH TERMS AND CONDITIONS.

1. Definitions

   (a) "Adaptation" means a work based upon the Work, or upon the Work and other pre-existing works, such as a translation, adaptation, derivative work, arrangement of music or other alterations of a literary or artistic work, or phonogram or performance and includes cinematographic adaptations or any other form in which the Work may be recast, transformed, or adapted including in any form recognizably derived from the original, except that a work that constitutes a Collection will not be considered an Adaptation for the purpose of this License. For the avoidance of doubt, where the Work is a musical work, performance or phonogram, the synchronization of the Work in timed-relation with a moving image ("synching") will be considered an Adaptation for the purpose of this License.

   (b) "Collection" means a collection of literary or artistic works, such as encyclopedias and anthologies, or performances, phonograms or broadcasts, or other works or subject matter other than works listed in Section 1(f) below, which, by reason of the selection and arrangement of their contents, constitute intellectual creations, in which the Work is included in its entirety in unmodified form along with one or more other contributions, each constituting separate and independent works in themselves, which together are assembled into a collective whole. A work that constitutes a Collection will not be considered an Adaptation (as defined below) for the purposes of this License.

   (c) "Creative Commons Compatible License" means a license that is listed at http://creativecommons.org/compatiblelicenses

that has been approved by Creative Commons as being essentially equivalent to this License, including, at a minimum, because that license: (i) contains terms that have the same purpose, meaning and effect as the License Elements of this License; and, (ii) explicitly permits the relicensing of adaptations of works made available under that license under this License or a Creative Commons jurisdiction license with the same License Elements as this License.

(d) "Distribute" means to make available to the public the original and copies of the Work or Adaptation, as appropriate, through sale or other transfer of ownership.

(e) "License Elements" means the following high-level license attributes as selected by Licensor and indicated in the title of this License: Attribution, ShareAlike.

(f) "Licensor" means the individual, individuals, entity or entities that offer(s) the Work under the terms of this License.

(g) "Original Author" means, in the case of a literary or artistic work, the individual, individuals, entity or entities who created the Work or if no individual or entity can be identified, the publisher; and in addition (i) in the case of a performance the actors, singers, musicians, dancers, and other persons who act, sing, deliver, declaim, play in, interpret or otherwise perform literary or artistic works or expressions of folklore; (ii) in the case of a phonogram the producer being the person or legal entity who first fixes the sounds of a performance or other sounds; and, (iii) in the case of broadcasts, the organization that transmits the broadcast.

(h) "Work"" eans the literary and/or artistic work offered under the terms of this License including without limitation any production in the literary, scientific and artistic domain, whatever may be the mode or form of its expression including digital form, such as a book, pamphlet and other writing; a lecture, address, sermon or other work of the same nature; a dramatic or dramatico-musical work; a choreographic work or entertainment in dumb show; a musical composition with or without words; a cinematographic work to which are assimilated works expressed by a process analogous to cinematography; a work of drawing, painting, architecture, sculpture, engraving or lithography; a photographic work to which are assimilated works expressed by a process analogous to photography; a work of applied art; an illustration, map, plan, sketch or three-dimensional work relative to geography, topography, architecture or science; a performance; a broadcast; a phonogram; a compilation of data to the extent it is protected as a copyrightable work; or a work performed by a variety or circus performer to the extent it is not otherwise considered a literary or artistic work.

(i) "You" means an individual or entity exercising rights under this License who has not previously violated the terms of this License with respect to the Work, or who has received express permission from the Licensor to exercise rights under this License despite a previous violation.

(j) "Publicly Perform" means to perform public recitations of the Work and to communicate to the public those public recitations, by any means or process, including by wire or wireless means or public digital performances; to make available to the public Works in such a way that members of the public may access these Works from a place and at a place individually chosen by them; to perform the Work to the public by any means or process and the communication to the public of the performances of the Work, including by public digital performance; to broadcast and rebroadcast the Work by any means including signs, sounds or images.

(k) "Reproduce" means to make copies of the Work by any means including without limitation by sound or visual recordings and the right of fixation and reproducing fixations of the Work,

including storage of a protected performance or phonogram in digital form or other electronic medium.

2. Fair Dealing Rights.Nothing in this License is intended to reduce, limit, or restrict any uses free from copyright or rights arising from limitations or exceptions that are provided for in connection with the copyright protection under copyright law or other applicable laws.

3. License Grant.Subject to the terms and conditions of this License, Licensor hereby grants You a worldwide, royalty-free, non-exclusive, perpetual (for the duration of the applicable copyright) license to exercise the rights in the Work as stated below:

   (a) to Reproduce the Work, to incorporate the Work into one or more Collections, and to Reproduce the Work as incorporated in the Collections;

   (b) to create and Reproduce Adaptations provided that any such Adaptation, including any translation in any medium, takes reasonable steps to clearly label, demarcate or otherwise identify that changes were made to the original Work. For example, a translation could be marked "The original work was translated from English to Spanish," or a modification could indicate "The original work has been modified.";

   (c) to Distribute and Publicly Perform the Work including as incorporated in Collections; and,

   (d) to Distribute and Publicly Perform Adaptations.

   (e) For the avoidance of doubt:

      i. Non-waivable Compulsory License Schemes. In those jurisdictions in which the right to collect royalties through any statutory or compulsory licensing scheme cannot be waived, the Licensor reserves the exclusive right to collect such royalties for any exercise by You of the rights granted under this License;

      ii. Waivable Compulsory License Schemes. In those jurisdictions in which the right to collect royalties through any statutory or compulsory licensing scheme can be waived, the Licensor waives the exclusive right to collect such royalties for any exercise by You of the rights granted under this License; and,

      iii. Voluntary License Schemes. The Licensor waives the right to collect royalties, whether individually or, in the event that the Licensor is a member of a collecting society that administers voluntary licensing schemes, via that society, from any exercise by You of the rights granted under this License.

   The above rights may be exercised in all media and formats whether now known or hereafter devised.

   The above rights include the right to make such modifications as are technically necessary to exercise the rights in other media and formats. Subject to Section 8(f), all rights not expressly granted by Licensor are hereby reserved.

4. Restrictions.The license granted in Section 3 above is expressly made subject to and limited by the following restrictions:

   (a) You may Distribute or Publicly Perform the Work only under the terms of this License. You must include a copy of, or the Uniform Resource Identifier (URI) for, this License with every copy of the Work You Distribute or Publicly Perform. You may not offer or impose any terms on the Work that restrict the terms of this License or the ability of the recipient of the Work to exercise the rights granted to that recipient under the terms of the License. You may not

sublicense the Work. You must keep intact all notices that refer to this License and to the disclaimer of warranties with every copy of the Work You Distribute or Publicly Perform. When You Distribute or Publicly Perform the Work, You may not impose any effective technological measures on the Work that restrict the ability of a recipient of the Work from You to exercise the rights granted to that recipient under the terms of the License. This Section 4(a) applies to the Work as incorporated in a Collection, but this does not require the Collection apart from the Work itself to be made subject to the terms of this License. If You create a Collection, upon notice from any Licensor You must, to the extent practicable, remove from the Collection any credit as required by Section 4(c), as requested. If You create an Adaptation, upon notice from any Licensor You must, to the extent practicable, remove from the Adaptation any credit as required by Section 4(c), as requested.

(b) You may Distribute or Publicly Perform an Adaptation only under the terms of: (i) this License; (ii) a later version of this License with the same License Elements as this License; (iii) a Creative Commons jurisdiction license (either this or a later license version) that contains the same License Elements as this License (e.g., Attribution-ShareAlike 3.0 US)); (iv) a Creative Commons Compatible License. If you license the Adaptation under one of the licenses mentioned in (iv), you must comply with the terms of that license. If you license the Adaptation under the terms of any of the licenses mentioned in (i), (ii) or (iii) (the "Applicable License"), you must comply with the terms of the Applicable License generally and the following provisions: (I) You must include a copy of, or the URI for, the Applicable License with every copy of each Adaptation You Distribute or Publicly Perform; (II) You may not offer or impose any terms on the Adaptation that restrict the terms of the Applicable License or the ability of the recipient of the Adaptation to exercise the rights granted to that recipient under the terms of the Applicable License; (III) You must keep intact all notices that refer to the Applicable License and to the disclaimer of warranties with every copy of the Work as included in the Adaptation You Distribute or Publicly Perform; (IV) when You Distribute or Publicly Perform the Adaptation, You may not impose any effective technological measures on the Adaptation that restrict the ability of a recipient of the Adaptation from You to exercise the rights granted to that recipient under the terms of the Applicable License. This Section 4(b) applies to the Adaptation as incorporated in a Collection, but this does not require the Collection apart from the Adaptation itself to be made subject to the terms of the Applicable License.

(c) If You Distribute, or Publicly Perform the Work or any Adaptations or Collections, You must, unless a request has been made pursuant to Section 4(a), keep intact all copyright notices for the Work and provide, reasonable to the medium or means You are utilizing: (i) the name of the Original Author (or pseudonym, if applicable) if supplied, and/or if the Original Author and/or Licensor designate another party or parties (e.g., a sponsor institute, publishing entity, journal) for attribution ("Attribution Parties") in Licensor's copyright notice, terms of service or by other reasonable means, the name of such party or parties; (ii) the title of the Work if supplied; (iii) to the extent reasonably practicable, the URI, if any, that Licensor specifies to be associated with the Work, unless such URI does not refer to the copyright notice or licensing information for the Work; and (iv) , consistent with Ssection 3(b), in the case of an Adaptation, a credit identifying the use of the Work in the Adaptation (e.g., "French translation of the Work by Original Author," or "Screenplay based on original Work by Original Author"). The credit required by this Section 4(c) may be implemented in any reasonable manner; provided, however, that in the case of a Adaptation or Collection, at a

minimum such credit will appear, if a credit for all contributing authors of the Adaptation or Collection appears, then as part of these credits and in a manner at least as prominent as the credits for the other contributing authors. For the avoidance of doubt, You may only use the credit required by this Section for the purpose of attribution in the manner set out above and, by exercising Your rights under this License, You may not implicitly or explicitly assert or imply any connection with, sponsorship or endorsement by the Original Author, Licensor and/or Attribution Parties, as appropriate, of You or Your use of the Work, without the separate, express prior written permission of the Original Author, Licensor and/or Attribution Parties.

(d) Except as otherwise agreed in writing by the Licensor or as may be otherwise permitted by applicable law, if You Reproduce, Distribute or Publicly Perform the Work either by itself or as part of any Adaptations or Collections, You must not distort, mutilate, modify or take other derogatory action in relation to the Work which would be prejudicial to the Original Author's honor or reputation. Licensor agrees that in those jurisdictions (e.g. Japan), in which any exercise of the right granted in Section 3(b) of this License (the right to make Adaptations) would be deemed to be a distortion, mutilation, modification or other derogatory action prejudicial to the Original Author's honor and reputation, the Licensor will waive or not assert, as appropriate, this Section, to the fullest extent permitted by the applicable national law, to enable You to reasonably exercise Your right under Section 3(b) of this License (right to make Adaptations) but not otherwise.

5. Representations, Warranties and Disclaimer

   UNLESS OTHERWISE MUTUALLY AGREED TO BY THE PARTIES IN WRITING, LICENSOR OFFERS THE WORK AS-IS AND MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND CONCERNING THE WORK, EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, INCLUDING, WITHOUT LIMITATION, WARRANTIES OF TITLE, MERCHANTIBILITY, FITNESS FOR A PARTICULAR PURPOSE, NONINFRINGEMENT, OR THE ABSENCE OF LATENT OR OTHER DEFECTS, ACCURACY, OR THE PRESENCE OF ABSENCE OF ERRORS, WHETHER OR NOT DISCOVERABLE. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO SUCH EXCLUSION MAY NOT APPLY TO YOU.

6. Limitation on Liability

   EXCEPT TO THE EXTENT REQUIRED BY APPLICABLE LAW, IN NO EVENT WILL LICENSOR BE LIABLE TO YOU ON ANY LEGAL THEORY FOR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL, PUNITIVE OR EXEMPLARY DAMAGES ARISING OUT OF THIS LICENSE OR THE USE OF THE WORK, EVEN IF LICENSOR HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

7. Termination

   (a) This License and the rights granted hereunder will terminate automatically upon any breach by You of the terms of this License. Individuals or entities who have received Adaptations or Collections from You under this License, however, will not have their licenses terminated provided such individuals or entities remain in full compliance with those licenses. Sections 1, 2, 5, 6, 7, and 8 will survive any termination of this License.

   (b) Subject to the above terms and conditions, the license granted here is perpetual (for the duration of the applicable copyright in the Work). Notwithstanding the above, Licensor

reserves the right to release the Work under different license terms or to stop distributing the Work at any time; provided, however that any such election will not serve to withdraw this License (or any other license that has been, or is required to be, granted under the terms of this License), and this License will continue in full force and effect unless terminated as stated above.

8. Miscellaneous

   (a) Each time You Distribute or Publicly Perform the Work or a Collection, the Licensor offers to the recipient a license to the Work on the same terms and conditions as the license granted to You under this License.

   (b) Each time You Distribute or Publicly Perform an Adaptation, Licensor offers to the recipient a license to the original Work on the same terms and conditions as the license granted to You under this License.

   (c) If any provision of this License is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this License, and without further action by the parties to this agreement, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.

   (d) No term or provision of this License shall be deemed waived and no breach consented to unless such waiver or consent shall be in writing and signed by the party to be charged with such waiver or consent.

   (e) This License constitutes the entire agreement between the parties with respect to the Work licensed here. There are no understandings, agreements or representations with respect to the Work not specified here. Licensor shall not be bound by any additional provisions that may appear in any communication from You. This License may not be modified without the mutual written agreement of the Licensor and You.

   (f) The rights granted under, and the subject matter referenced, in this License were drafted utilizing the terminology of the Berne Convention for the Protection of Literary and Artistic Works (as amended on September 28, 1979), the Rome Convention of 1961, the WIPO Copyright Treaty of 1996, the WIPO Performances and Phonograms Treaty of 1996 and the Universal Copyright Convention (as revised on July 24, 1971). These rights and subject matter take effect in the relevant jurisdiction in which the License terms are sought to be enforced according to the corresponding provisions of the implementation of those treaty provisions in the applicable national law. If the standard suite of rights granted under applicable copyright law includes additional rights not granted under this License, such additional rights are deemed to be included in the License; this License is not intended to restrict the license of any rights under applicable law.

Creative Commons Notice

Creative Commons is not a party to this License, and makes no warranty whatsoever in connection with the Work. Creative Commons will not be liable to You or any party on any legal theory for any damages whatsoever, including without limitation any general, special, incidental or consequential damages arising in connection to this license. Notwithstanding the foregoing two (2) sentences, if Creative Commons has expressly identified itself as the Licensor hereunder, it shall have all rights and obligations of Licensor.

Except for the limited purpose of indicating to the public that the Work is licensed under the CCPL, Creative Commons does not authorize the use by either party of the trademark "Creative Commons" or any related trademark or logo of Creative Commons without the prior written consent of Creative Commons. Any permitted use will be in compliance with Creative Commons' then-current trademark usage guidelines, as may be published on its website or otherwise made available upon request from time to time. For the avoidance of doubt, this trademark restriction does not form part of the License.

Creative Commons may be contacted at `http://creativecommons.org/`.

BILBOKO INDUSTRIA INGENIARITZA TEKNIKOKO UNIBERTSITATE ESKOLA

INDUSTRIA INGENIARITZA TEKNIKOA: INDUSTRIA ELEKTRONIKA

KARRERA AMAIERAKO PROIEKTUA

Anie - 5. Dokumentua: Baldintzen agiria

IKASLEAREN DATUAK	**ZUZENDARIAREN DATUAK** Koldo Basterretxea Oyarzabal koldo.basterretxea@ehu.es Teknologia Elektronikoa
SIN.: DATA: 2013ko apirilaren 12a	SIN.: DATA:

Gaien Aurkibidea

1.
Baldintza orokorrak

Hardware deskribapenaren edota dokumentazioaren zatien banaketa, erabilera edota moldaketa agiriek *Free Software Foundation*[1] erakundeak definitutako lau oinarrizko askatasunak/eskubideak bermatzen dituzte:

0. edozein xederako erabiltzeko askatasuna,
1. lagun eta kideekin elkarbanatzeko askatasuna,
2. zure beharretara moldatzeko askatasuna, eta
3. egindako aldaketak banatzeko askatasuna.

Hala ere, materialek izaera ezberdina dutenez, hiru lizentzia ezberdinen arabera banatzen dira proiektua eta iturriak[2]:

1.1. GNU [Lesser] General Public License ([L]GPLv3)

VHDL iturriei eta hardware deskribapenei[3] dagokienean:

This description is free hard-software: you can redistribute it and/or modify it under the terms of the GNU [Lesser] General Public License (according to Table 5.1.) as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This description is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

[1] fsf.org/licensing/essays/free-sw

[2] 3A. eranskineko *script*ak *Matlab*en dokumentazioan eta *Mathworks*ek sarean eskainitako plataformetan aurkitutako edukiak baliatuz idatzi dira. Inolako baldintzarik gabe eskaintzan dira, beraz.

[3] Eranskinetan adierazitakoak, txosteneko beste ataletan adierazitako zatiak zein sostengu digitalean (zerbitzarian edo fisikoan -CDa kasu-) bildutakoak.

GNU GPLv3	GNU LGPLv3		
anietiny	sat	lcd	
anie	counter	setup	
pid	rot	writing	vgasync
norm	udcounter	autoinit	vgademo
timing	bintobcd		vgadata
man	bcdshift	trigger	vgashift
lcdman	lcdmenu	adq	vgasimpgrid
vgaman	switch		vgasignshift
input	btndeb	pwm	
output	debounce	hbridge	

5.1. Taula: *VHDL deskribapenak erabili, aldatu eta hedatzeko baldintzak osagaien arabera adierazita.*

You should have received a copy of the GNU GPLv3[4] *and GNU LGPLv3*[5] *along with this description. If not, see* `gnu. org/ licenses`.

1.2. Creative Commons Attribution-ShareAlike (CC-BY-SA)

1.1. eta 1.3. azpiataletan adierazitako edukiak izan ezik, txosten hau osatzen duten guztiak[6] CC-BY-SA-3.0 lizentziaren arabera ematen dira:

Attribution-ShareAlike 3.0 Unported[7]
(CC-BY-SA-3.0)

- Egin ditzakezunak:

  Banatzea Kopiatu, banatu eta hedatzea

  Moldatzea Lana egokitzea eta eratorriak egitea

  Lana merkataritza helburuekin erabiltzea

[4] Ikus 4C. eranskina edo `gnu.org/licenses/gpl-3.0.html`.
[5] Ikus 4D. eranskina edo `gnu.org/licenses/lgpl-3.0.html`.
[6] Azalpen, irudi, taula, diagrama, eta grafikoak.
[7] Ikus 4E. eranskina edo `creativecommons.org/licenses/by-sa/3.0/`.

- Hurrengoak bete bitartean:

  Aitortzea Lanaren iturria aitortu behar da, *Unai Martinez Corral* eta *Koldo Basterretxea Oyarzabal*i eta *APES* taldeari erreferentzia eginez, eta ehu.es/apes orrialdea aipatuz (baina lan eratorriek edo lanaren erabilerek hauen babesa dutela adierazi barik).

  Berdin partekatzea Lan hau moldatu edo egokituz gero, edo lan eratorririk sortzekotan, egindakoa banatzeko honetan erabilitako lizentzia berdina erabili behar da.

1.3. Aitorpen zuzenak

Hurrengoak eta aipatutako erakundeen zein hauek garatutako produktuen logotipo eta irudiak haiei dagozkie, eta bakoitzak adierazitako baldintzen arabera banatzen dira:

- 4C./4D. eta 4E. eranskinetako edukia *Free Software Foundation, Inc.* eta *Creative Commons Corporation* erakundei dagozkie, izenburuek adierazi bezala, eta zuzenean kopiatu dira iturrietatik (goiburu eta orri oinak moldatuz, soilik).
- 1.5. irudia [37] dokumentutik kopiatu da, eta *Xilinx, Inc.* erakundeari dagokio.
- 2-2. ataleko *Subversion* zerbitzariaren estatistiken grafikoak *StatSVN*[8] programarekin egin dira eta zuzenean itsatsi.
- 2.1. irudiko edukia *Tektronix MSO2024* osziloskopioarekin sortu da eta zuzenean itsatsi, *Tektronix, Inc.*[9] erakundeari dagozkio.
- 2.2. irudia *Simulink* inguruneko *Mathworks, Inc.* eradundeak eskeinitako osagaiak moldatuta egin da.
- 2-1.2.2. atalean adierazitako sintesi txostenek *Xilinx*en *ISE*k eskainitako *Synthesis Report*en informazio esanguratsuena biltzen dute.
- 3B. erankineko *box_pid_config.m* fitxategiak *Simulink/System Generator* ingurunean *Matlab*ek *Blackbox* blokea erabiltzean zuzenean sortutakoa du oinarri.
- 1.4. irudiko maketa eta 1.36. irudiko *encoder*a [1] dokumentutik hartu eta moldatu dira, eta *Alecop, S. Coop.* erakundeari dagozkio.
- 1.37. eta 1.39. irudietako eskemak *Digilent, Inc.* erakundeak [42] dokumentuarekin batera eskaintzen duen eskema moldatuta egin dira.
- The Noun Project[10]: Laptop and iPhone *iconoci*, Sync *P.J. Onori*, Robot *Luis Prado*, Network *Jerry Wang*, Time *Richard de Vos*, Puzzle *Dmitry Baranovskiy*, Conversation *Sébastien Desbenoit*, Memory *Andrew J. Young*, Data *United Nations OCHA*

[8] statsvn.org
[9] tek.com
[10] thenounproject.com

- TeXample[11]: Gajski-Kuhn Y-chart *Ivan Griffin*
- GNU - Christian Cadena[12]: GNU License Alternative Logos

1.4. Zeharkako aitorpenak

- 1.7. irudiak *Arturo Urquizo*ri dagokion *Wikipedia*ko[13] *PID.svg, Controlador PID en lazo cerrado* irudia du oinarri.
- 1.24. irudia *Xilinx*ek dokumentazioko *Clocking Wizard - Cascading In Series With Two DCMs* atalean aurkeztutako *Cascading in Series with Two DCMs* irudian oinarrituta dago.

Txostena egiteko jarraian zerrendatutako tresnak erabili dira:

- gedit (projects.gnome.org/gedit) / Notepadd++ (notepad-plus-plus.org)
- TeXmaker (xm1math.net/texmaker) / BibTeX (bibtex.org) / LaTeX paketeak (ctan.org/pkg/): import, inputenc, babel, geometry, graphicx, natbib, caption, indentfirst, multirow, amsmath, rotating, subfig, eurofont, siunitx, longtable, xcolor, listings, tikz, pgfplots, tikz-timing, url, hyperref, etoolbox, eso-pic, fancyhdr, minitoc
- QtikZ (hackenberger.at/blog/ktikz-editor-for-the-tikz-language)
- Dia (live.gnome.org/Dia) / Inkscape (inkscape.org) / GIMP (gimp.org)

Eta *Subversion* zerbitzaria kudeatzeko:

- RapidSVN (rapidsvn.tigris.org) / TortoiseSVN (tortoisesvn.net)
- Meld (meldmerge.org/) / WinMerge (winmerge.org)
- StatSVN (statsvn.org) / Gource (code.google.com/p/gource)

[11] texample.net/tikz/examples/
[12] http://www.gnu.org/graphics/license-logos.en.html
[13] wikipedia.org

2.
Baldintza administratiboak

Proiektu honetan, eskaera egiten den momentutik, salmentan ematen diren hurrengo baldintzak onartu beharko dira. Hauen edozein eraldaketak, legezko eragina izan dezan, idatziz egin beharko da eta onarpenena jaso beharko du.

Proiektua onartua izanez gero, saltzailea ez da derrigorturik egongo lana bere osotasunean egitera, baina bai zeharo bukaturik lagatzera. Honetarako subkontratatutako beste enpresa bati eman ahalko zaizkio proiektuaren zati batzuk. Edozein moldaketa nahi izanez gero, proiektua eta aurrekontua onartuta daudelarik, aurretik pentsatu gabe zegoen aparaturen bat edo hobekuntzaren bat egin behar bada, konstruktorearen jakinaren gainean jarri beharko da, baina argi geratu behar da ez dela saltzailearen konpetentzia hobekuntza edo material hauen horniketa egitea. Proiektua bukaturik dagoenean, bertaratutako Erakunde Ofizial batek ikusi eta konprobatu beharko du edozein erreklamazio eginez.

Proiektu eta suministroen prezioak eta jornalak ere moldaketak izan ditzakete eskaera egin eta entregatu bitartean, aurrekontuan azaltzen diren materialen balioetan eta jornaletan moldaketak egon daitezke. Eskaera egin zen eguneko prezioak izango dira eman beharrekoak. Zerga guztiak denborekin deribak izaten dituzte, hauek eskalearen edo konstruktorearen kargupean geratzen dira. Hauek euren artean duten kontratuaren araberakoa izango da.

2.1. Kontratistaren betebeharrak muntaketan

Proiektuaren dokumentuek eta planoek dioten bezala obra burutu dadin egin beharreko lanak egitera derrigortua dago kontratista, baita Ingeniari Teknikoaren indikazioak jarraitzera ere. Egindako errakuntzak moldatu beharko dira indemnizazio eskubiderik izan ez arren. Nahi izanez gero Ingeniariak onartu dezake hala ondo dagoela uste badu. Aldi berean, Ingeniariak askatasun osoa izango du berari iruditzen zaion edozein moldaketa txiki egitera.

Muntaketa egin bitartean elektrizitate instalazioan edozein arazo egongo balitz, konstruktorea derrigortua dago hauek konpontzera, honetarako pertsonal espezializatua izango duelarik. Segurtasuna eta garbitasunari eta langileen esperentzia faltari dagokionean ere, kontratista izango da arduradun bakarra. Neurri berdinak aplikatuko dira obran sartzen diren pertsonal baimenduarekin. Gaur egungo legearen arabera beharreko guztiak izango ditu bere langileen segurtasunari eta ezbeharrei dagokionean, honela lehen sorospenak erraztu beharko ditu.

2.2. Inspekzioak, baimenak eta erantzukizunak

Proiektu honen hedadura dela eta, Industriatik pasatu beharko da, ikasia eta onartua izan dadin. Hau instalazioa egin aurretik egin beharko da eta honekin ordaindu beharrekoak enpresa konstruktoreak ordaindu beharko ditu.

Enpresako konstruktoreko teknikoen lanean eta Ingeniari Teknikoaren direkzioan, ez du ino-

lako ardurarik izango enpresak ezbeharrei eta hirugarren batzuei eginiko beste kalte batzuei dagokionean. Erosleak beharreko babes guztiak izango ditu ezbeharrik gerta ez dadin. Gertatuz gero, enpresa edo pertsona bera izango dira arduradunak.

Enpresako teknikoek ezin izango dute konpromisorik izan enpresaren izenean. Honetarako kasu bakoitzerako desberdina den idatzizko dokumentua beharko du.

2.3. Konpetentzia jurisdikzionalak

Elkarrekin egindako akordio bategatik aurretik esandako baldintzetako bat aldatu edo kenduz gero beste guztiak ordu-arteko balioarekin geratuko dira.

3.
Baldintza teknikoak

3.1. Muntaiaren baldintzak, abioa eta funtzionamendua

Sistemak era egokian funtzionatzeko oinarrizko elementua FPGA da. Ondorioz, sistema era egokian eta arintasunez funtzionatu dezan honen ezaugarriak minimo bat gainditu behar dute. Memorian erabilitako *Spartan3E* baino baliabide gutxiago dituen gailu batekin ez da ziurtatzen sistemaren funtzionamendua, ez bakarrik egokiro funtzionatzea baizik eta sistemak ez funtzionatzea.

Honetaz gain instalatua izango den tokian beste sistema elektroniko gehiagorekin izatean ez ditu betebehar garrantzitsu gehiago armairuan leku bat izatea baino, eta honek txartelak behar duen elikadurarako sarrera bat izatea baino.

Proiektua bere osotasunean funtzionamenduan jarri aurretik, berrikuspena egin beharko da ezbeharrak gerta ez daitezen. Egin beharreko frogak egin beharko dira funtzionamendua egokia izan dadin. Era berean, erabiliko den material guztiak fabrikatzaileek egindako kalitate probak gaindituak izango ditu (baldintzarik txarrenetan). Proiektua martxan jarri aurretik, egoera normala baino altuagoekin egin beharko dira proba guztiak elkarkonektatuz eta sistemaren portaera egokia dela ikusiz. Hau denbora luzez utziko da eta haiek aukeratutako erabiltzailearen edo Ingeniariaren aurrean egingo da.

3.2. Erabilera eta mantentze baldintzak

Txartelari dagokion atalean beronen erabilera eta mantentzea pertsonal espezializatuaren esku geldituko da. Bai bere hardwareari dagokionez eta baita beronekin lotua dagoen konfiguragarritasunarekin. Honek zera esan nahi du: hardware funtzionamendu eskasaren aurrean zerbitzu teknikoari deitu beharko zaiola, eta berau izango dela FPGA birkonfiguratu dezakeen bakarra.

Hardware elektronikoari dagokion mantentzea ere pertsonal espezializatuaren esku geldituko da. Beronen funtzionamenduan arazoren baten aurrean zerbitzu teknikoari ditu beharko zaio, eta honek baino ez du gaitasuna izango osagai elektronikoak aldatzeko, aldatu beharreko kasuan, edo berauen konexioak konpontzeko, kasuan kasukoa.

4.
Baldintza ekonomikoak

4.1. Muntaia: epea eta garantia

Muntaketa osoa 10 eguneko luzapena izan dezake, hauek gainditua, instalazio osoa muntatuta eta probatuta egongo dela ziurtatzen da.

Proiektuaren jasotzaileak ez du eskubiderik izango eskaera deusezteko aurretik ezer esan gabe. Aurretik esatekoak idatziz egin beharko dira eta enpresara helduarazi. Eskaera deuseztatzea ez da gauzatuko hirugarren deuseztatze abisua bidali arte, hauen artean 15 eguneko epea utziko da atzerapena enpresaren errua denean.

Ezin izango zaio alde saltzaileari indemnizazioa eskatu atzerapenak direla eta, baldin eta eskaera gauzatzerakoan, hala estipulatua eta onartua izan bazen zigor moduan.

Enpresak urte beteko garantia eskaintzen du (instalazio osoaren bukaera eta berrikuspena egin ondorengo epea) proiektuan azaldutako edozein elementuren funtzionamendu ezegokia bada, proiektuan dagoen errakuntza betegatik. Hau gertatuz gero enpresa derrigortua dago elementua edo errakuntza zuzendu edo ordezkatzera ahalik eta denbora epe laburrean.

Deskribapenei dagokionean, garatzaileak material bidea erraztuko du. Jakin behar da erabilera normalerako materialak eta manufaktura erruz garbi daudela entregatze epetik 90 egunetara. Honen froga agiria izango da egin beharreko kopia bat.

4.2. Konpetentziak, tarifak eta ordainketarako baldintzak

Instalatzea eta muntaia egin ahal izateko Elektronikako 1.Ofiziala beharko da. Ez-ohiko orduei dagokionez, jornalak, likidazioa, etab., enpresa instalatzailea egingo da kargu ez baitira hasierako baldintzetan sartzen. Dietak ez daude kontabilizaturik, ezta ere egonaldiak, garraioak, etab. Honelako guztiak enpresa instalatzailearen esku geratuko dira. Lan ordutegia 8 ordukoa izango da.

Ordainketa bankuko transferentzia baten bidez egingo da. Kontratuan bestelakorik agertzen ez bada.

Ordainketa egiteko baldintzak hurrengoak dira:

1. Proiektua onartuz gero, eraikitzaileak %20ko fidantza ipini beharko du. Honela egingo da enpresak idatzizko dokumentu batekin kontrakoa esan ezean.

2. Bigarren ordainketa totalaren %30koa izango da, eta materialak entregatu eta instalatzea bukatzerakoan egin beharko da.

3. Hirugarren ordainketa totalaren %50-koa izango da eta Proiektu osoa bukatzetik aurrera 90 eguneko epean egin beharko da.

Ordainketa bakoitzeko agiri bat entregatuko da norbaitek kontratua apurtzekotan nolako klausulekin. Entregatzean atzerapenak egonez gero instalatze guztia gelditu beharko da atzerapena 15 egun baino gehiagokoa baldin bada. Hau gertatuz gero atzerapenen interesak ordaindu beharko dira.

Totala ordaintzen ez den bitartean instalatutako material guztia edo instalatzeke dagoena enpresarena dela argi utzi behar da.

4.3. Aseguruak eta sorospideak

Desplazatu beharreko langileak behar bezala aseguraturik egongo dira eta kontratua izango dute diru-laguntza, aseguru, oporrak, etab.-ekin. Bezeroa, bai jabegoa edo bai eraikitzailea, derrigortua dago ezbeharren bat gertatuz gero lehen sorosteak erraztera. Zerbait gertatuz gero, hauen garraioa, eta larria izanez gero, zauritua ospitale edo larrialdietara eramatea ere hauen erantzunpean geratzen dira.

BILBOKO INDUSTRIA INGENIARITZA TEKNIKOKO UNIBERTSITATE ESKOLA

INDUSTRIA INGENIARITZA TEKNIKOA: INDUSTRIA ELEKTRONIKA

KARRERA AMAIERAKO PROIEKTUA

Anie - 6. Dokumentua: Aurrekontua

IKASLEAREN DATUAK	**ZUZENDARIAREN DATUAK**
	Koldo Basterretxea Oyarzabal koldo.basterretxea@ehu.es Teknologia Elektronikoa
SIN.: DATA: 2013ko apirilaren 12a	SIN.: DATA:

1.
Garapen osoa

Hardwarea			**763,77€**
Oinarrizko sistema			
Spartan3E Starter Kit	168,00€	1	168,00€
Azterketa kasua			
MV-541	500€	1	500€
PmodHB3	16,28€	4	65,12€
Elikadura iturria ($6-15$V eta 5 A)	30,65€	1	30,65€
Softwarea			**437,5€**
Mathworks	1200€	0,3125	375€
Matlab	500€		
Simulink	500€		
Control System Toolbox	200€		
Xilinx	200€	0,3125	62,5€
ISE System Edition	200€		
Lan-orduak			**7500€**
Oinarrizko sistemaren garapena	12,5€	360	4500€
Erreferentzien bilaketa eta planifikazioa		40	
Sismulazioan oinarritutako garapena		160	
Oinarrizko sistema deskribatzea		80	
Periferikoen osagaiak deskribatzea		40	
Sistema balioztatzea		40	
Azterketa kasua	12,5€	80	1000€
Erreferentzien bilaketa eta planifikazioa		20	
Sarrerako/irteerako osagaiak deskribatzea		40	
Planta identifikatzea		8	
Kontrolagailua doitzea		6	
Sistema balioztatzea		6	
Dokumentazioa egitea	12,5€	160	2000€
			8701,27€

Zergak

Zerga-oinarria		8701,27€
BEZ		
Hardwarea	+%21	160,39€
Softwarea	+%21	91,88€
PFEZ		
Lan-orduak	-%5	-375€
		8578,54€

2.
Azterketa kasu berria lantzea

Softwarea			**87,5€**
Mathworks	1200€	0,0625	75€
Matlab	500€		
Simulink	500€		
Control System Toolbox	200€		
Xilinx	200€	0,0625	12,5€
ISE System Edition	200€		
Lan-orduak			**1500€**
Azterketa kasua	12,5€	80	1000€
Erreferentzien bilaketa eta planifikazioa		20	
Sarrerako/irteerako osagaiak deskribatzea		40	
Planta identifikatzea		8	
Kontrolagailua doitzea		6	
Sistema balioztatzea		6	
Dokumentazioa egitea	12,5€	40	500€
			1587,5€

Zergak

Zerga-oinarria		1587,5€
BEZ		
Softwarea	+%21	18,38€
PFEZ		
Lan-orduak	-%5	-75€
		1530,88€

BILBOKO INDUSTRIA INGENIARITZA TEKNIKOKO UNIBERTSITATE ESKOLA

INDUSTRIA INGENIARITZA TEKNIKOA: INDUSTRIA ELEKTRONIKA

KARRERA AMAIERAKO PROIEKTUA

Anie - 7. Dokumentua: Bibliografia

IKASLEAREN DATUAK	**ZUZENDARIAREN DATUAK**
	Koldo Basterretxea Oyarzabal koldo.basterretxea@ehu.es Teknologia Elektronikoa
SIN.: DATA: 2013ko apirilaren 12a	SIN.: DATA:

Bibliografia

[1] Alecop, S. Coop., *Descripción del equipo MV-541.*

[2] U. Martinez Corral, "Vumeter / LCD 3310 / ReoBus." Teknologia - Ander Deuna Ikastola, June 2005.

[3] U. Martinez Corral and A. Martin Uribarri, "Ordenagailu-Haizagailuak Kontrolatu eta Ikusteko Sistema (OHKIS)." Eragingailu Logiko Programagarriak Dituzten Sistema Digitalak - Bilboko IITUE - UPV/EHU, June 2009.

[4] U. Martinez Corral, L. Ranero Santisteban, and I. Sarramian Olmos, "Control de velocidad de un motor CC: NI Labview." Sistemas Digitales en la Medida y Control de Procesos Industriales - EUITI de Bilbao - UPV/EHU, June 2012.

[5] K. J. Åström and R. M. Murray, *Feedback Systems, An Introduction Scientists and Engineers.* Princeton University Press, 2009.

[6] B. C. Kuo, *Sistemas de Control Automático*, ch. 2-11 La transformada z. Prentice-Hall, Inc., 7 ed., 1995.

[7] G. De Micheli and R. K. Gupta, "Hardware/software co-design," in *Proceedings of the IEEE*, vol. 85, pp. 349–365, March 1997.

[8] A. Aguado Behar and M. Martínez Iranzo, *Identificación y Control Adaptativo.* Automática Robótica, Prentice-Hall, 2003.

[9] L. Moreno Fernández, S. Garrido Bullón, and C. Balaguer Bernaldo de Quirós, *Ingeniería de control: modelado, análisis y control de sistemas*, ch. 7.4.2 Métodos de Ziegler-Nichols and 7.5 Métodos analíticos de diseño de controladores PID. Editorial Ariel, S.A., 2003.

[10] I. Grout, *Digital Systems Design with FPGAs and CPLDs*, ch. 7. Introduction to Digital Signal Processing. Newnes, imprint of Elsevier B.V., 2008.

[11] A. Whitworth *et al.*, *The Wikibook of automatic Control Systems*, ch. Introduction to Digital Controls/Sampled Data Systems/Reconstruction. Wikimedia Foundation, Inc., 2012.

[12] U. Ugalde, R. Bárcena, and K. Basterretxea, "Generalized sampled-data hold functions with asymptotic zero-order hold behavior and polynomic reconstruction," *Automatica*, vol. 48, pp. 1171–1175, March 2012.

[13] P. Piqtek and W. Grega, "Speed analysis of a digital controller in time critical applications," *Journal of Automation, Mobile Robotics & Intelligent Systems*, vol. 3, pp. 57–61, 2009.

[14] M. Santina and A. R. Stubberud, *The Control Handbook, Control System Fundamentals*, ch. 15. Sample-Rate Selection. CRC Press, imprint of Taylor & Francis Group, LLC, 2 ed., 2011. Edited by W.S. Levine.

[15] M. Santina and A. R. Stubberud, *The Control Handbook, Control System Fundamentals*, ch. 12. Discrete-Time Equivalents of Continuous-Time Systems. CRC Press, imprint of Taylor & Francis Group, LLC, 2 ed., 2011. Edited by W.S. Levine.

[16] G. F. Franklin, J. D. Powell, and M. L. Workman, *Digital Control of Dynamic Systems*. Prentice-Hall, 3 ed., 2006.

[17] M. Schmidt, "Derivative action in discrete PID controllers." Faculty of Electrical Engineering and Communication - Brno University of Technology.

[18] K. J. Åström, *Control System Design, Lecture Notes for ME 155A*, ch. 6. PID Control. Department of Mechanical and Environmental Engineering - University of California, 2002.

[19] D. Seborg, T. Edgar, and D. Mellichamp, *Process Dynamics and Control*, ch. 22. Sampling and Filtering of Continuous Measurements. John Wiley & Sons, Inc., 1 ed., 1989.

[20] G. Zito, I. Dore Landau, F. Bouziani, and A. Voda-Besançon, "Digital PID tuning by controller complexity reduction," *Technical report*, vol. Laboratoire d'Automatique de Grenoble, March 2005.

[21] T. L. Lagö, "Digital Sampling According to Nyquist and Shannon," *Sound and vibration*, February 2002.

[22] T. Wescott, *Sampling: What Nyquist Didn't Say, and What to Do About It*. Wescott Design Services, Inc., December 2012.

[23] A. Visioli, *Practical PID Control*. Advances in industrial control (AIC), Springer-Verlag London Limited, 2006. Edited by M.J. Grimble and M.A. Johnson.

[24] A. Whitworth *et al.*, *Wikibook, Floating Point (IEEE 754 standard)*, ch. 1. Number Representation. Wikimedia Foundation, Inc., 2011.

[25] G. Frantz and R. Simar, *Comparing Fixed- and Floating-Point DSPs. Does your design need a fixed- or floating-point DSP?* Texas Instruments, Inc., 2004.

[26] B. Lerner, *Fixed vs. floating point: a surprisingly hard choice*. Analog Devices, February 2007. http://eetimes.com/design/.

[27] Xilinx, Inc., *System Generator for DSP, Getting Started Guide*, March 2008.

[28] Xilinx, Inc., *System Generator for DSP, User Guide*, April 2012.

[29] T. Murata *et al.*, *Wikipedia, Type system*, ch. 2.5 Strong and weak typing: Liskov Definition, 2.6 Strong and weak typing, 2.7 Safely and unsafely typed systems. Wikimedia Foundation, Inc., 2012.

[30] J. Lewis, *VHDL Math Tricks of the Trade*. SynthWorks Design Inc., 2003.

[31] D. W. Bishop, *VHDL-2008 Support Library*. EDA Industry Working Groups. http://vhdl.org/fphdl/.

[32] D. W. Bishop, *Fixed point package user's guide.* EDA Industry Working Groups.

[33] IEEE DASC Synthesis Working Group - PAR 1076.3, *Standard VHDL Synthesis Package (1076.3, NUMERIC_STD).* IEEE, 1995.

[34] R. Cofer and B. Harding, *Fixed-Point DSP and Algorithm Implementation.* Avnet, October 2006. http://eetimes.com/design/.

[35] B. Beauregard, *Improving the Beginner's PID: Reset Windup.* http://brettbeauregard.com.

[36] Xilinx, Inc., *Spartan-3E FPGA Family, Data Sheet*, October 2012.

[37] Xilinx, Inc., *Spartan-3E Starter Kit Board, User Guide*, March 2006.

[38] K. Chapman, *Rotary Encoder Interface for Spartan-3E Starter Kit.* Xilinx, Inc., February 2006.

[39] *Conversions, Binary to Bcd - Bcd to Binary.* http://jjmk.dk.

[40] J. 0. Hamblen and M. D. Furman, *Rapid prototyping of digital systems, a tutorial approach*, ch. 9. VGA Video Display Generation. Kluwer Academic, 2 ed., 2006.

[41] B. Artaloytia Encinas, *Laboratorio de Ingeniería Eléctrica, 4. Electrónica de excitación empleada en la actuación de motores de corriente contínua de baja potencia.* Dpto. TEAT-UPM. http://etsit.upm.es/departamentos/teat.

[42] Digilent, Inc., *Digilent PmodHB3, 2A H-Bridge Reference Manual*, February 2012.

[43] B. Philofsky, *Seven Steps to an Accurate Worst-Case Power Analysis Using Xilinx Power Estimator (XPE).* Xilinx, Inc., September 2008.

[44] A. Martin Uribarri, "Ordenagailu Haizagailuak Kontrolatu eta Ikusteko Sistema." Karrera Amaierako Proiektua - Bilboko IITUE - UPV/EHU, July 2009.

[45] B. Iglesias Rozas and U. Martinez Corral, "Bicicleta μControlador v1.1." Informática Industrial - EUP Ferrol - Universidade da Coruña, June 2010.

[46] U. Martinez Corral, "Acher v0.1." Empleo del Ordenador Personal en la Instrumentación de Panel - Bilboko IITUE - UPV/EHU, January 2011.

[47] U. Martinez Corral, "Acher, serie komunikazio bitartez LED matrizea kudeatzeko sistema." Bulego Teknikoa - Bilboko IITUE - UPV/EHU, June 2011.