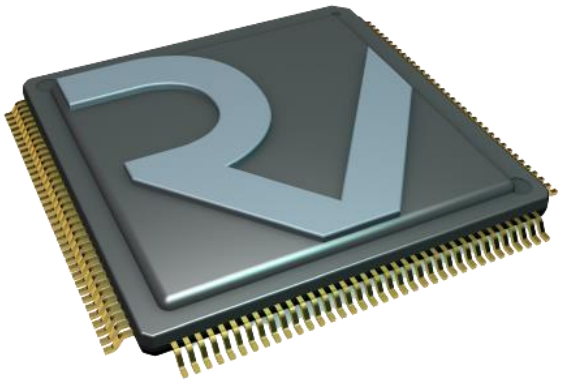
Universidad del País Vasco Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

Telekomunikazio Ingeniaritzako Unibertsitate Masterra

MASTER AMAIERAKO LANA

RISC-V-n OINARRITUTAKO SoC BATEN DISEINUA



Ikaslea: Galicia Zabala, Unai

Zuzendaria: Lázaro Arrotegui, Jesús

Ikasturtea: 2022-2023

Data: Bilbo, 2023ko irailak 18

Laburpena

Konputazio sistemetan, prozesagailuaren egokitasuna alderdi kritiko bat da sistemaren errendimendu eta honi lotutako kostuei dagokionez. Gailu honek sistemak exekutatu behar dituen atazak modu ahalik eta eraginkorrenean egitea ahalbidetzen duen hardwarea inplementatzen badu, efizientzia orokorra asko igo daiteke. Honela, haien helburuetarako ahaltsuak, energetikoki efizienteak eta fabrikazioari dagokionez merkeagoak diren sistemak sortu daitezke.

Hala ere, helburu zehatzetarako prozesagailuen diseinua ez zen ataza erraza gaur egunera arte. Tradizionalki, prozesagailuen diseinuaren industria itxia edo propietarioa izan da, gailu ezberdinen diseinu edota fabrikazioa enpresa gutxi batzuk gauzatzen zutela, hauek bakarrik ezagutzen zuten *Instruction Set Architecture* (ISA) jakin bat inplementatuz.

RISC-V ISA irekiaren agerraldiak prozesagailuen diseinua demokratizatu du, ingeniariei helburu zehatzetarako prozesagailuak eta hauetan oinarritutako System-on-Chip-ak (SoC) diseinatzeko tresna oso baliagarria eskura jarri baitie, alderdi bai tekniko bai ekonomikoetan hirugarrengoek ezarritako muga guztiak ezabatuz.

Proiektu honetan RISC-V prozesagailu batean oinarritutako SoC baten diseinua gauzatu da, horretarako eskuragarri dauden kode-irekiko RISC-V prozesagailuen inplementazio ezberdinak alderatu eta erabili direla. Diseinu hau, APERT taldeak garatu nahi duen ASICaren oinarria izango da, SoC diseinuak ikerkuntza taldeak gehitu nahi dituen IP (*Intellectual property*) bloke ezberdinen gehikuntza ahalbidetzen duela.

Resumen

En los sistemas de computación, el nivel de adecuación del procesador es un aspecto crítico respecto al rendimiento y los costes del sistema. Dicho dispositivo debe de estar compuesto por un hardware el cual permita ejecutar las tareas que se le presentan al sistema de la manera más eficiente, para así incrementar también la eficiencia general del sistema lo máximo posible. De esta manera, se pueden crear sistemas muy potentes para las tareas objetivo, energéticamente eficientes y más baratos en cuanto a su fabricación.

No obstante, el diseño de procesadores no ha sido, hasta hoy en día, una tarea fácil de realizar. El diseño de procesadores ha pertenecido tradicionalmente a una industria cerrada y propietaria, dado que el diseño y fabricación de estos dispositivos ha sido llevada a cabo por unas pocas empresas, las cuales implementaban una *Instruction Set Architecture* (ISA) conocida únicamente por ellos.

La aparición de la ISA abierta RISC-V ha democratizado el proceso del diseño de procesadores, ya que ha otorgado a los ingenieros una herramienta que permite el diseño de estos dispositivos y System-on-Chip (SoC) para tareas específicas, eliminando cualquier limitación en el diseño de estos, tanto en el aspecto técnico como el económico.

En este proyecto se ha diseñado un SoC basado en un procesador RISC-V, para ello realizando una comparativa entre los procesadores RISC-V de código abierto disponibles. Dicho diseño será la base para el ASIC que el equipo de investigación APERT quiere realizar. El diseño del SoC está pensado además para permitir al grupo de investigación añadir bloques IP (*Intellectual Property)* adicionales.

Abstract

The adequation of the processor in a Computing System is a critical aspect regarding performance and system costs. This device must be composed of hardware that enables the processor to execute the tasks that the system has to deal with most efficiently, in order to maximize the general efficiency of the system. By doing so, the creation of powerful systems for specific tasks can be achieved, such as obtaining energy-efficient systems that are cheaper to produce.

However, processor design has not been an easy task until now. The design process has always belonged to a closed and proprietary industry, mainly because a few companies have carried out the design and fabrication of processors, creating devices that implement Instruction Set Architectures (ISA) only known by these companies.

The flowering of RISC-V, one of the first open ISAs, has democratised the processor design process, as it has brought to engineers a tool that enables processor and System-on-Chip (SoC) design without any kind of restriction, leveraging the interest in purpose-specific processors.

The design of a SoC based on a RISC-V processor has been carried out in this project. An analysis of the different open-source RISC-V processors available has also been made to choose the processor of the system. The aforementioned design will be the basis for the ASIC that the APERT research team is planning to create. For this reason, the design has been made to permit the inclusion of extra IP (Intellectual property) blocks, so that the Research Team can include extra functionalities in the final design.

Aurkibidea

Irudien zerrenda

Taulen zerrenda

Akronimoen zerrenda

ABI	*Application Binary Interface*
ALU	*Arithmetic Logic Unit*
APERT	*APplied Electronics Research Team*
ASIC	*Application Specific Integrated Circuit*
AXI	*Avanced eXtensible Interface*
BCD	*Binary-Coded Decimal*
BOOM	*Berkeley Out-of-Order Machine*
BRAM	*Block Random Access Memory*
BSD	*Berkeley Software Distribution*
BSS	*Block Starting Symbol*
CISC	*Complex Instruction Set Computer*
CSR	*Control Status Register*
CV32E40P	*Core-V 32 Embedded 40 Processor*
DARPA	*Defence Advanced Research Projects Agency*
DMA	*Direct Memory Access*
DSP	*Digital Signal Processor*
EDA	*Electronic Design Automation*
ELF	*Executable and Linkeable Format*
ESA	*European Space Agency*
FPGA	*Field Programmable Gate Array*
FPU	*Floating Point Unit*
FreeRTOS	*Free Real Time Operating System*
GCC	*GNU C Compiler*
GDB	*GNU Debugger*
GPIO	*General Purpose Input Output*
GPU	*Graphics Processing Unit*
HDL	*Hardware Description Language*
HPC	*High Performance Computing*
IEEE	*Institute of Electrical and Electronics Engineers*

I/O	*Input / Output*
IoT	*Internet of Things*
IP	*Intellectual Property*
IRQ	*Interrupt ReQuest*
ISA	*Instruction Set Architecture*
ISR	*Interrupt Service Routine*
JTAG	*Joint Test Action Group*
LLVM	*Low Level Virtual Machine*
LSU	*Load-Store Unit*
LUT	*Look Up Table*
MIPS	*Microprocessor without Interlocked Pipeline Stages*
OBI	*Open Binary Interface*
PC	*Program Counter*
PMP	*Physical Memory Protection*
PULP	*Parallel Ultra Low Power*
PWM	*Pulse Wave Modulation*
RAM	*Random Access Memory*
RISC	*Reduced Instruction Set Computer*
RISC-V	*Reduced Instruction Set Computer V*
ROM	*Read Only Memory*
RoT	*Root of Trust*
RTL	*Register Transfer Level*
SBC	*Single Board Computer*
SE	*Sistema Eragilea*
SIMD	*Single Instruction Multiple Data*
SoC	*System on Chip*
SPARC	*Scalable Processor ARChitecture*
SPI	*Serial Peripheral Interface*
UART	*Universal Asynchronous Receier-Transmitter*
UVM	*Universal Verification Methodology*
VHDL	*Very high speed integrated circuit Hardware Description Language*

1. Sarrera

Informazio eta Komunikazio Teknologien beharra inoiz baino garrantzitsuagoa da gaur egun. Gizarteak darabiltzan gailu elektronikoen kopurua asko handitu da azken urteetan zehar, izan ere, edonork ezin duelako gailu hauek gabe bere bizitza nolakoa izango litzatekeen pentsatu. Noski, berdina gertatzen da arlo guztietako enpresetan; oinarri digitala duten sektoreetan gailuak gero eta ugariagoak dira eta inoiz baino bezero gehiagori eman behar diete zerbitzua. Bestalde, oinarri ez-digitaleko arloetan informazioaren kudeaketa edo atazen automatizazioa bezalako ataza garrantzitsuen atzean aurki daitezke gailu hauek ere.

IoT edota 5G kontzeptuen agerraldiak hau guztia asko areagotu du, gailu hauen eskaria esponentzialki igoz. Agertoki hauetan gainera, erabili ohi diren sistemek ataza gutxi eta oso espezifiko batzuk betetzen dituzte, ataza hauen eraginkortasuna eta gailuen efizientzia energetikoa direla kezka nagusiak. Bestalde, egungo zerbitzu digital askoren atzean dauden datu zentroetan, jasaten diren informazio-prozesaketa lan-karga gero eta izugarriagoei erantzun ahalik eta onena eman ahal izateko, konputazio soluzioak gero eta sofistikatuagoak bihurtu dira beharraren beharrez, efizientzia energetikoa mantendu behar dutela ere bai.

Gainera, adimen artifizialaren gorakada dela eta, teknologia hau leku orotan inplementatzen hasia da, eta, jakina denez, teknika hauek oso intentsiboak dira konputazioaren aldetik, algoritmo oso konplexuak denbora-tarte oso txiki batean egitea eskatzen baitute. Beraz, arlo guzti hauetan prozesatze-gaitasun gehiago behar dela modu argi batean ikusi daiteke, alegia, konputazio-sistema gero eta eraginkorragoak behar dira.

Orain arte, teknologia digital guztiekin gertatu ohi den bezala, urtetik urtera edo hilabete batetik bestera aurrerapen izugarriak egon dira. Gailu elektronikoen kasuan, hauen fabrikazioan egindako aurrerapenak izan dira amaierako produktuen errendimendu hobekuntzen erantzule nagusienetako bat; material erdieroale batean inprimatutako transistoreen tamaina txikitzen joan den ahala, gailu elektronikoen prestazioak izugarri igo dira.

Hala ere, Mooreren legea dela eta, erdieroaleen industriari gero eta zailagoa egiten zaio haien diseinuetan aurrerapenak egitea. Olatetan aurki daitezkeen transistoreen dentsitateak garapen logaritmiko bat jarraitu du, eta gailu elektronikoen errendimendu eskariak, berriz, garapen esponentziala jarraitu du. Hori dela eta, gaur egungo teknologiak norabide berri baten beharra du, eskaera mailari erantzun bat emateko.

Arazo honi irtenbide bat eman ahal izateko, gero eta ohikoagoa bilakatu da gailuen espezializazioa. Ataza zehatz bat modu eraginkorrean bete ahal izango duen gailu pertsonalizatu baten erabileraz, ataza gutxi batzuetarako gailu oso eraginkorrak lortu daitezke, bai prozesatze gaitasunaren aldetik, bai efizientzia energetikoaren aldetik.

Ildo honetan, prozesatze gaitasuna asko hobetu daiteke ataza horiek optimizatzen dituzten operazioak gauzatzen dituzten prozesatzaileak erabiliz. Datu zentroetan nahiko argi ikusi daiteke hau; aipatu denez, prozesatze abiadura gero eta handiagoa behar da datuen emaria gero eta handiagoa delako, hori dela eta ataza ezberdinak azeleratzen dituzten gailu espezializatuez (azeleragailuak) osatutako sistemak izaten ari dira erabilienak.

Bestalde, diseinu digitalaren munduan jarraitu den eredu klasikoa zaharkituta geratu da. Tradizionalki, prozesatzaile edo dena delako sistema digitalen diseinatzaileak enpresa handi gutxi batzuk izan dira. Hauek, helburu jakin baterako gailu baten diseinu bakar bat egiten zuten, ondoren beste enpresa asko eta askori saltzeko.

Baina gaur egun, eredu honek ez ditu merkatuaren beharrak betetzen. Aipatu den espezializazioa ezin daiteke negozio eredu hori jarraitzen duen merkatu batean lortu; honek aldatu egin behar du ataza ezberdinetara egokitzen diren gailuak eskaintzeko.

Gainera, sistema digitalen kasuan, aipatutako diseinu horiek kode itxiko diseinu bezala salduak izan dira, askotan diseinuak moldatzea galarazten duten lizentziak ezarriz. Orain ikusiko den bezala, hau oso eragozpen handia da diseinu berrien garapenerako, bereziki prozesatzaileen kasuan, kasu honi buruz hitz egingo dela.

1.1 ISA ezberdinak eta hauen garapena

Prozesagailuek edozein programa exekutatu ahal dezaten, programa bakoitza agindu edo instrukzio sinple batzuetara itzuli beharko da, ondoren instrukzio horiek bit mailan prozesagailura eramateko. Instrukzio bakoitzaren bit mailako edukiaren arabera, hardwareak eragiketa ezberdinak egingo ditu.

Argi dago beraz, prozesagailu baten hardwarearen diseinua instrukzioen egituraren araberakoa izango dela. Hori dela eta, prozesagailu bat diseinatzerako garaian, gailuak bete ahalko dituen ekintzak definituko dituen instrukzioen multzo bat definitzea da lehenengo pausua. Instrukzioen multzo honi Instrukzio Multzoen Arkitektura deritzo, ISA bezala ezagutua bere Ingelesezko siglak direla eta (*Instruction Set Architecture*). Prozesagailuaren hardware diseinua edo mikroarkitektura beraz, ISAren inplementazioa da. ISAk definitutako instrukzioen bit zabalerak arkitekturaren bit kopurua definitzen du ere, bit zabalera ohikoenak 32 eta 64 direla. 128 bitetako zabalerak errendimendu altuko sistemetan aurki daitezke.

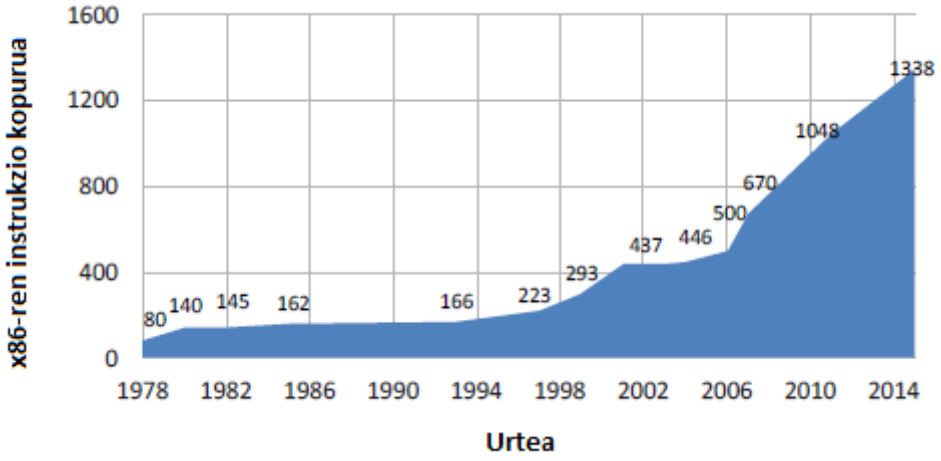
Hardware diseinuak bezala, historian zehar egon diren ISA garrantzitsuenak kode itxikoak izan dira, hauek sekretupean mantendu direla edo hauen moldaketa galarazia izan dela bezeroentzat. Gainera, ISA hauen erabilera lizentziak oso garestiak izan dira. Honela, prozesatzaileen moldaketa ia ezinezkoa izan da orain arte.

ISA historiko hauen artean gainera, ezaugarri ohiko bat izaera inkrementala izan da. Adibidez, x86-k, ISA itxi guztien artean seguruenik ezagunena, filosofia hau jarraitu du bere garapenean zehar. ISA inkrementalek, ezaugarri berriak inplementatzeko zabaltzen direnean, aurretik zeuden instrukzio zaharren gainean berriak inplementatzen dituzte, hau da, ISAren eboluzio bakoitzean, ISA birdiseinatu beharrean agindu berriak gehitu izan zaizkio zaharrak mantenduz.

Honen arrazoia softwarearen atzerako bateragarritasuna (*backwards compatibility*) mantentzea izan da. Prozesagailu berriek zaharrek zituzten instrukzioak exekutatu ahal baditzakete, konpilatutako programa zaharrak martxan jarri ahalko dituzte. Hala ere, honek hainbat desabantaila dakartza inplizituki.

Alde batetik, bere garaian erabilgarriak ziren baina denborarekin zaharkituta geratu diren hainbat instrukzio inplementatu behar dituzte ISA inkrementalean oinarritutako prozesagailuek. Esate baterako, x86 ISAn [1] BCD (*Binary-Coded Decimal*) kodearekin eragiketa aritmetikoak gauzatzeko lau instrukzio ageri dira. Nahiz eta kode hori gaur egun ez erabili, ISA hau inplementatzen duten prozesagailuek agindu horiek ulertu beharko dituzte.

Honela, x86 ISA gero eta instrukzio gehiago pilatzen joan da urteetan zehar, ISA oso konplexua bihurtu dela. Hurrengo grafikoan, x86-ren jatorritik 2014. urtera arte bere instrukzio kopuruan jasan duen eboluzioa ikusi daiteke. Irudian gainera, ez dira kontuan hartzen azkeneko urteetan gehitutako instrukzio kopurua, honen hazkundea esponentziala izan dela azkeneko urteetan. 2015. urterako hain zuzen ere, ISAra ia 4000 instrukzio gehituko zirela aurreikusten zen Intel etxeak publikatutako artikulu baten arabera [2].



1. Irudia: x86-ren instrukzio kopuruaren eboluzioa urteetan zehar

Horretaz gain, ISA askok biltzen dituen instrukzio asko konplexuak dira haien baitan. x86 bezalako ISA konplexuek hainbat operazio sofistikatu gauzatzeko instrukzio konplexu bakarrak erabiltzen dituzte, hala nola SIMD (*Single Instruction Multiple Data*) motako instrukzioak. Honela, ataza zailak instrukzio bakar batekin ebazten dira, normalean erloju ziklo ugari behar dituzten instrukzioak direla.

Noski, agindu konplexuen edota ISA zabal eta konplexu baten erabilerak hardware konplexu batekin ordaintzen dira, emaitza bezala prozesagailuen inplementazioak zailagoak eginez. Amaierako produktuen energia erabileran eragin handia du ere bai diseinu estrategia honek, hardware gehigarria dela eta. Hau kostuetan islatzen da, prozesadoreen diseinua eta garapena luzeagoa eta garestiagoa bihurtzen baita.

1.2 RISC Motako ISAk

Azkeneko ezaugarri honen aurka, instrukzio konplexuak ekiditen dituzten beste mota bateko ISA batzuk sortu izan dira. Hauek, ataza konplexuak instrukzio bat baino gehiagotan ebazten dituzte, aginduak inplementatzeko behar den hardware konplexua aurreztuz eta ISAren instrukzio kopurua murriztuz. Gainera, agindu sinpleagoak erabiltzen direnez, bakoitza azkarrago exekutatu daiteke, nahiz eta agindu gehiago behar ataza konplexuak betetzeko.

ISA hauek inplementatzen dituzten sistemak RISC edo *Reduced Instruction Set Computers* deritzo, x86 bezalako instrukzio konplexuz osatutako ISAen antonimo bezala, zeintzuk CISC edo *Complex Instruction Set Computers* bezala ezagutzen diren.

ARM enpresaren RISC prozesagailuak mota hauetako erabilienak direla esan daiteke. Haien prozesagailuak hauen aplikazioen arabera 3 multzotan sailkatuz, ISA ezberdinak aurki daitezke; A motakoak Sistema Eragile konplexuen erabilerarako, R motakoak denbora errealeko aplikazioetarako eta M motakoak mikrokontrolagailuen ekintzetarako.

M motak orain arte aurkeztu diren ISA guztiek ez duten ezaugarri bat du, ISA modularra dela. ARMen M ISAn, oinarrizko ISA bat aurki daiteke, M ISAren ezaugarri sinpleenak inplementatzen dituenak. Hala ere, oinarrizko ISA hau luzatzeko aukera existitzen da, bestelako ataza espezifikoagoak egitea ahalbidetzen dituzten instrukzioen ISA gehigarrien bitartez.

Diseinurako estrategia honekin, prozesagailuak beteko dituen atazak kasuan kasu aztertuz, prozesagailuaren diseinua beharretara hobekien egokituz gauzatu daiteke, nahi diren luzapenak aplikatuz eta erabiliko ez diren luzapenak ez inplementatuz. Honek kostu eta errendimenduaren aldetik onura handiak dakartza, diseinu eta baliabide gastuak kasura egokitu ahal baitira.

Hala ere, ARMen M ISA ez da guztiz bateragarria bertsioen artean, hainbat luzapenek ez baitute ezaugarri hau betetzen. Eta noski, ARMen ISA guztiak bezala, itxia da hau ere bai. Ondorioz, nahiz eta luzapenak eskura izan, bezeroek prozesagailuak puntu batera arte pertsonaliza ditzakete, ARMek eskaintzen dituen luzapenak erabiliz soilik. Azkeneko hau merkatuan aurkitu daitezkeen bai RISC eta bai CISC prozesagailu ia guztiek ezaugarri hau duten ISAetan oinarritzen dira, hau da, ISA itxietan oinarritzen dira.

1.3 RISC-V ISA

Azkeneko urteetan oso popularra bihurtu da 2011. urtean publikoki azaldutako ISA alternatibo bat, prozesagailuen industriak dituen aipatutako arazo guzti hauek konpontzeko gai bezala ikusten dena: RISC-V [3]. ISA honen izaerak industriara abantaila asko ekartzeko potentziala du, gehienbat hiru ezaugarri direla eta:

1. **Kode irekiko ISA bat da.** Hori dela eta, ISA guztiz eskuragarria da edonorentzat, edonori ISAn oinarritutako prozesatzaile pertsonalizatuak diseinatzeko aukera emanez.
2. **ISA berri bat da.** Gaur egungo ISA nagusiak 1970 edo 1980eko hamarkadetan sortuak dira, eta normala denez, urte horietan zehar zenbait akats izan dituzte. RISC-V akats horietatik ikasiz joan da garatzen.
3. **ISA Modularra da.** RISC-V-k oinarrizko ISA bat aurkezten du, luzapenekin osatu daitekeena. Honela, helburu bakoitzerako beharrezkoa inplementatzen duten prozesagailuak diseinatu daitezke.

Ikusi daitekeenez, ezaugarri hiru hauek industriaren arazo nagusiei ematen diete erantzuna, prozesadore pertsonalizatuen sorrera errazten duelako, edonork edozein atazarako moldatutako prozesagailu bat diseinatzea ahalbidetzen duela.

1.3.1 Jatorria

RISC-V-ren hastapenak 2010. urtean eman ziren, Berkeley Unibertsitateko Krste Asanovic irakaslearen eta, Yunsup Lee eta Andrew Waterman Graduko ikasleen eskutik. ISAren garapena hainbat enpresek (Intel eta Microsoft beste batzuen artean) eta Californiako Estatuak ordaindutako *Konputazio Paralelorako Laborategia* proiektuaren parte zen. Unibertsitate hau izan zen hain zuzen ere 1980an RISC kontzeptuaren sortzailea, RISC-V unibertsitate honetan egindako RISCen bosgarren inplementazioa dela.

ISA ireki baten onurak ikusita, eta RISC-V beraren posibilitateak ikusita, Estatu Batuetako DARPA erakunde militarrak haien proiektuetarako RISC-V ISAn oinarritutako prozesagailuak sortzeko proiektu bat finantziatu zuen, finantziazio hau oso garrantzitsua izan zela ISAren garapenerako.

Nahiz eta hasiera batean RISC-V erabilera akademikorako pentsatua izan, bere sortzaileek estandar ireki gisa publikatu zuten gizarteak ISA honen beharra zuela ikusi ondoren. Honela, 2015. urtean RISC-V fundazioa sortu zen. Erakunde honen helburua RISC-V-n oinarritutako software eta hardware garatzaileen komunitate baten sorrera zen, RISC-V-ren garapena eta erabilera bultzatzeko.

RISC-V-k ez du inolaz prozesatzaileen diseinu itxiekin amaitu nahi; enpresa eta gizabanakoei prozesagailuen diseinuak egiteko aukera ematen die ondoren hauek haien IP produktu itxiak sortu ditzaketela.

Aipagarria da RISC-V ez dela lehenengo ISA irekia; eredu irekiak sortzeko bestelako saiakera batzuk egin ziren ere iraganean. Esate baterako OpenSPARC, Sun Microsystemsen SPARC ISA itxia irekitzen zuenak; OpenRISC ISA irekia; edota erabilera askeko MIPS arkitektura, esate baterako. Hala ere, saiakera hauek ez dute denboran zehar arrakasta oso handia izan, seguruenik aipagarriena MIPS izan zela.

ISA ireki hauen arrakasta falta bi arrazoi nagusik justifikatzen dute, alde batetik industriaren adopzio eza eta bestetik komunitate zabaldu baten sorrera ez zelako eman. Hau guztiz beharrezkoa da produktu baten biziraupenerako, produktuaren erabilera eta garapena posible egiten baitu.

1.3.2 RISC-V Gaur egun

RISC-V arrakasta lortzen ari da, aurrekari diren ISA irekiek jasan zituzten arazo horiei erantzuna ematen dion RISC-V Fundazioa daukalako oinarrian. Erakunde honen helburua RISC-V-n oinarritutako software eta hardware garatzaileen komunitate baten sorrera da, RISC-V-ren garapena eta erabilera bultzatzeko. Erakundea da ISAren espezifikazio ezberdinen sortzailea, honek argitaratzen dituela ISAra gehitutako luzapen berriak.

Erakundeak modu irekian garatzen ditu bere ISA ezberdinak, hauen garapenean Fundazioko edozein kidek har dezakeela parte, edozein izan daitekeela kide. Honela, komunitate osoaren parte hartzea ahalbidetzen du, teknologiaren garapena bermatuz. Gainera, RISC-V Fundazioak honek sortutako ISA eta luzapenei aske eta dohainekoak izan daitezen babes juridikoa ematen dio, fundazioaren kezka nagusienetako bat ISAren ezaugarri hauek mantentzea baita.

Fundazioak kideak ezberdintzen ditu, ekarpen ekonomiko eta teknologikoak egin ahal dituzten enpresak, unibertsitateak eta gizabanakoak bereiziz. Maila altueneko kideek komite nagusian eta komite teknikoetan errepresentazioa lortzen dute, gizabanakoek ere errepresentazioa dutela komitean haiek bozkatutako ordezkari batekin.

Honela, Fundazioaren laguntzaz, RISC-V-n oinarritutako inplementazioak heltzen joan dira, ISAn oinarritutako soluzio gero eta sofistikatuago eta konpetenteak sortzen joan direla. RISC-V-ren garapen mailaren adierazle oso argia da SiFive enpresa, RISC-V-ren sortzaileek fundatutakoa. Sistema txertatuetarako coreak (*E cores*) edota Sistema Eragile (SE) konplexuak exekutatzeko multicore plataformak (*U Core)* aurkitu daitezke haien katalogoan, denak RISC-V-n oinarrituak.

Baina seguruenik RISC-V-n oinarritutako soluzioen aurrerapenak baino garrantzitsuagoa izaten ari da ISAk industria mailan gradualki jaso duen adopzioa. Softwarea garatzen duten enpresa teknologiko garrantzitsu askok ISA irekian oinarritutako hardwarearentzako bateragarriak diren bertsioak hasi dira garatzen, eta hardware produktuak sortzen dituzten enpresa askok ere RISC-V prozesagailuak gehitu dituzte haien produktuetan. Adibidez, Googlek (RISC-V fundazioko kidea) bere Android SEa RISC-V sistemetan hedatzea lortzeko proiektua abiatu zuela iragarri zuen 2023ko *RISC-V Summit*-ean, proiektu hau GitHuben publikoa eginez [4].

1.4 Lan honen helburua

Dokumentu honetan aurkezten den proiektua RISC-V-ren erabilera kasu oso argi bat da. APERT ikerkuntza taldeak SoC baten diseinua gauzatu nahi du, hainbat IP bloke ezberdin batzen dituen SoC baten ASIC bat sortu ahal izateko. Duela urte batzuk, SoC-aren prozesagailua diseinatzeko aukera gutxietako bat Intel edo ARM bezalako enpresa bati prozesagailu baten diseinuaren lizentzia bat erostea litzateke, honen kostuak proiektua seguruenik atzera eramango lukeela. RISC-V-rekin aldiz, diseinu hori egitea posiblea da, hau, gainera, taldearen eskakizunetara nahi bezain beste moldatu ahal dela.

Horretarako, existitzen diren RISC-V-ren kode-irekiko inplementazio ezberdin batzuk aztertu dira, SoC baten diseinurako egokiena aukeratzea izan dela helburu. Azterketa egiteaz gain, inplementazioak IP gehigarriekin nola luzatu aztertu da, APERTek eskura dituen IP bloke ezberdinak RISC-V SoC-era lotzeko etorkizunean. Azkenik, SoC-en diseinu ezberdinak Xilinxen FPGAtan inplementatu dira hauek balidatzeko.

1.5 Garapen iraunkorreko helburuak

Proiektu honekin, garapen iraunkorreko hainbat helburu betetzera lagundu nahi da. Zehazki, jarraian zerrendatzen diren helburuetara egin nahi da ekarpena.

- **Desberdintasunak murriztea:** RISC-V ISA irekiaren erabilera sustatuz, prozesagailuen diseinua demokratizatu daiteke. Honen ondorioz, edozein gizabanako edo erakundek haien diseinu propioak garatu ditzakete, hauei eta, ondorioz, herrialdeei gaur egun ataza hau betetzen duten enpresa gutxiekiko independentzia emanez. Beraz, ISA honen erabilera herrialde ezberdinen garapen ezberdintasunak txikiagotzen lagundu dezake.

  Bestalde, RISC-V-n oinarritutako prozesagailuek hardware gutxiago erabiltzea dute helburu, hauen eta, ondorioz, hauek erabiltzen dituzten produktuen prezioak murriztuz. Honek ezberdintasun sozialak gutxiagotu ditzake, baliabide gutxiko gizabanakoei produktuak eskuratzea erraztuz.

- **Kalitatezko hezkuntza:** RISC-V bezalako ISA ireki batek prozesagailuen eta konputazioaren irakaskuntza asko hobetu eta erraztu dezake, irekia den lehenengo ISA baita. Ondorioz, ikasleei prozesagailu baten funtzionamendua eta egitura guztiz erakutsi ahal izatea ahalbidetzen duen tresna baten aurrean gaudela esan daiteke.

  Proiektu honetan ezerezetik prozesagailu bat duen SoC bat diseinatu nahi izan da, prozesu hau nola egin bildu nahi izan dela ere. Ezagutza hau irakaskuntzarako baliagarria izan daitekeela uste da, ondorioz arlo honetan irakaskuntza errazteko dokumentazioa sortu dela.

- **Lan duina eta hazkunde ekonomikoa:** Prozesagailuen demokratizazioa bultzatzen duen ISAren erabilera sustatu nahi izan da lan honetan. ISA hau bultzatuz, honetan oinarritutako hardwarea diseinatzen duen enpresa berrien sorrera sustatu nahi da. Honela, lehiakortasuna bultzatuko lukeen eszenario bat sortu nahi da, hazkunde ekonomikoa bultzatzea helburu.

- **Hiri eta komunitate jasangarriak:** RISC-V prozesagailuen erabilerak prozesagailu espezializatuen sorrera duela helburu esan daiteke, hardware gutxiago erabiliz ataza espezifikoak betetzen dituzten gailuak sortuz. Honela, fabrikazio prozesuaren eragina gutxiagotu daiteke, baina batez ere, amaierako gailu hauek gaur egun erabiltzen diren beste gailu askok baino kontsumo energetiko baxuagoa dute, ingurumenean eragin txikiagoa dutela.

  Hau bereziki garrantzitsua izan daiteke hiri jasangarriak sortzeko, RISC-V-n oinarritutako prozesagailuak hirietan masiboki aplikatu nahi diren IoT edota 5G bezalako aplikazioetan aurkitu ahalko direla aurreikusten baita.

2. Testuingurua

2.1 RISC-V-ren fokatzea

RISC-V kode irekiko ISA modular bat da, eta gaur egun erabiltzen diren ISA asko ez bezala, duela gutxi sortutako ISA bat da. Azkeneko hau oso garrantzitsua izan da ISAren garapenerako, jada existitzen diren ISAek sortutako akatsak kontuan hartu zirelako ISAren diseinuan.

ISAren helburu nagusia unibertsala bihurtzea da [5]. Lehen aipatu den moduan, gaur egungo prozesatzaileen ISA asko helburu jakin batzuetara mugatzen da soilik, aldiz, RISC-V-ren fokatzea orokorra da, sistema txertatuetatik superkonputazio sistementzako egokiak diren prozesagailuen diseinua ahalbidetzea dela helburua.

Ez hori bakarrik, bere inplementazioa edozein motako teknologietan ahalbidetzeko diseinatu da, ASIC, FPGA edo dena delako gailuetan. Azkeneko honek garrantzi handia du, RISC-V sistema heterogeneoetan inplementatu ahal izatea ISAren sortzaileen helburuetako bat baitzen, hauetan edo bestelako sistemetan *multicore* inplementazioak egin ahal izatea beste helburu bat izanda. Birtualizazioa ahalbidetzea beste helburuetako bat zen ISAren jatorrian.

Bestalde, espezializazio maila oso altu bat ahalbidetzen du; ISA honetan oinarrituta prozesagailu orokorrak diseinatzeko aukera emateaz gain, helburu jakin bat optimizatzeko azeleragailuen diseinua ahalbidetzeko pentsatuta dago RISC-V. Hau oso garrantzitsua da lehen aipatutako Mooreren legearekin lotutako arazoa dela eta.

ISAren jatorrizko beste helburu oso garrantzitsu bat honen sinpletasuna lortzea zen. Alde batetik, RISC-V irakaskuntza munduan erabili ahal izatea, eta bestetik, inplementazio berrien erraztasuna areagotzea lortu nahi zen honekin.

Azkenik, ISA egonkorra izatea RISC-V-ren beste helburu bat da. Honek ez du ISAren eguneraketak egonkorrak izan behar direla esan nahi, baizik eta RISC-V erakundeak kaleratutako espezifikazio bakoitza behin-betikoa izan behar dela.

Aipatu den bezala, RISC-V ISA modular bat da. Ezinbesteko agindu batzuk definitzen dituen oinarrizko instrukzioen multzo batzuk existitzen dira, RISC-V gailuek derrigorrez inplementatu behar dituztenak. Multzo hauek gehienbat zenbaki osoekin lan egiteko oinarrizko instrukzioak definitzen dituzte. Horietaz gain, aukerazko instrukzio multzo ezberdinak existitzen dira, oinarrizko multzoei gehitu dakizkiokeenak ISA luzatzeko. Ezaugarri hau dela eta, RISC-V ISA ezberdinez osatzen dela esan daiteke.

Modulartasunaren helburua efizientzia da, sistema bakoitzari beharrezkoak dituen gaitasunak emateko aukera sortzen du eta; sistema txikiei beharrezko ezaugarri murritzak eskainiz eta sistema ahaltsuei optimizaziorako behar bezain beste modulu gehitzeko posibilitatea sortuz.

Aukerazko instrukzio multzo hauek beharren arabera gehitu daitezke. Esate baterako, F luzapena oinarri-multzo bati gehituz, komadun eragiketentzako instrukzioak gehitzen dira, hauek optimizatuz. Honek ez du aldaera hori gabe komadun eragiketak egin ezin direnik esan nahi, hauek software bidez inplementatu baitaitezke, baina inplementazioa ez da hain optimoa izango. F luzapena exekutatzen duen hardwarea inplementatzen duen prozesagailu batean aldiz, askoz ere azkarrago gauzatuko dira eragiketak, agindu hauek exekutatzeko gai den hardwarea gehitzearen trukean.

2.2 RISC-V ISAren egitura eta moduluak [6]

Konbentzioz, RISC-V-ren luzapen ezberdinak inplementatzen dituen prozesagailuei buruz hitz egiterakoan, luzapenen identifikazio hizkiak bata bestearen atzean ipintzen dira, sistemak instrukzioetan darabilen bit luzeraren ondoren. Honela, I oinarrizko instrukzio multzoa eta F luzapena inplementatzen dituen 32 biteko RISC-V prozesadore bat RV32IF bezala identifikatzen da. Bestalde, RISC-V-ren I edo E oinarrizko multzoak eta A, M, F eta D luzapenak inplementatzen dituzten 32 bitetako prozesagailuak RV32G bezala ezagutzen dira, hauen orokortasuna dela eta, G-ak ingelesezko *General* hitzari egiten baitio erreferentzia.

RISC-V coreen barneko erregistroak eta helbideratze espazioa erabilitako oinarrizko ISAren arabera aldatzen dira. RV32I eta RV64I-ren kasuan, 32 edo 64 biteko helbideratze espazioa lortzen da hurrenez hurren, bit kopuru horri XLEN deritzola. Honela, RISC-V core batek 2^{XLEN} biteko helbideratze espazioa du, espazioa zirkularra dela. Memoria helbide batek gainera, memoria nagusira, I/O gailu batera edo ezerrera apuntatu dezake. Erregistro asko inplementatzen ditu RISC-V-k, oinarri moduluaren arabera 32 edo 16 dituela hain zuzen ere; ezaugarri honek ISAren sinplifikaziora laguntzen duela. Perspektiban jartzeko, x86-k 8 erregistro darabiltza soilik.

Bestalde, ISA bi multzotan banatzen da. Alde batetik, ISA ez-pribilegiatua aurki daiteke, sistemaren ikuspegi mugatu bat duten instrukzioak biltzen dituenak, eta bestetik, ISA pribilegiatua existitzen da, sistemaren ezaugarri guztietara sarbidea duen modu pribilegiatua definitzen duenak.

Jarraian, RISC-V ISA modu sakonago batean azalduko da, bere derrigorrezko eta aukerazko luzapen garrantzitsuenak azalduko direla. Hasteko, ISA ez-pribilegiatuari buruz hitz egingo da, bertan aurkitzen baitira prozesagailu baten alderdi funtzionala definitzen duten oinarrizko eta aukerazko ISA nagusiak. Ondoren, ISA pribilegiatua azalduko da. Proiektuan erabili diren moduluen inguruko azalpen sakonagoak 1. eranskinean aurkitu daitezke, azalpen sakonagoak RISC-V prozesagailuen funtzionaltasunak erakusteko azaltzen direla.

2.2.1 RV32I – Zenbaki osoentzako oinarrizko RISC-V ISA

Aipatu den moduan, ISA honek zenbaki osoekin eragiketak gauzatzeko instrukzioak definitzen ditu, bestelako guztiz oinarrizko agindu askorekin batera. ISAren oinarrizko modulu hau Fundazioak berretsi du, beraz Fundazioaren espezifikazioa denboran ez dela aldatuko bermatuta dago.

Eragiketa hauen artean, gehiketa, kenketa, and, or, xor edota desplazamendu aritmetiko zein logikoak aurki daitezke, biderketarako agindurik ez dela gehitzen. Balio ezberdinen arteko konparaketarako aginduak definitzen dira ere.

Bestalde, operazio aritmetiko-logikoetaz gain, erregistroekin operazioak gauzatzeko aginduak definitzen dira ere RV32I-n. Memoriatik balioak kargatzeko eta gordetzeko aginduak espezifikatzen ditu ere.

Exekuzioaren kontrolerako aginduak definitzen dira ere ISA honetan, baldintzapeko edota baldintza gabeko jauziak definitzen direla. *ecall* edo *ebreak* bezalako aginduak ere aurkitu daitezke, hauek etendura bat sortu dezaketela ISAren modu pribilegiatua inbokatuz.

Aipatu den bezala, oinarrizko ISA bakoitzak bere erregistro kopurua ditu, ISA honen kasuan 32 erregistro definitzen direla. Lehen aipatu den bezala, erregistro kopuru altua da hau, honen arrazoi nagusia ISAren sinplifikazioa lortzea dela.

Azkenik, aipagarria da *Program Counter*-a edo PC erregistroa ez dela erregistro hauen parte. Erabaki honen arrazoia *branch*en aurresatea erraztea da, horrela jakina delako erregistroak erabiltzen dituzten instrukzioak ez direla *branch*ak.

Lehenengo eranskinean ISA honi buruzko xehetasunak ematen dira.

2.2.2 RV32M – Biderketa eta zatiketentzat aukerazko ISA

Luzapen honek biderketa eta zatiketa gauzatzeko instrukzioak gehitzen dizkio RV32I-ri. Luzapen hau RISC-V Fundazioak berretsi zuen ere, beraz honen gainean ez dira aldaketak espero. M aginduen kasuan, definitzen diren instrukzio guztiek kodifikazio bera jasotzen dute.

Biderketaren kasuan, 32 biteko biderketek 64 biteko emaitza sortzen dute. Nahiz eta eragiketa hau agindu bakar batekin egitea posiblea izan, ISA honek biderketak egiteko bi aginduen erabilera egitera behartzen ditu sistemak, hardwarea sinpleagoa mantentzeko.

Lehenengo eranskinean ISA honi buruzko xehetasunak ematen dira.

2.2.3 RV32F eta RV32D – Doitasun sinple eta bikoitzeko komadun zenbakientzako aukerazko ISAk

RISC-V-ren jatorrizko helburuetako bat IEEEren 754-2008 komadun zenbakien aritmetikarako estandarrarekin betetzea zenez sortu ziren aukerazko ISA hauek. Ohikoa izaten da F eta D ISA biak batera inplementatzen dituzten sistemak.

ISA hauen gehikuntzak aldaketa handi bat dakar, eragiketa hauentzako erregistroak gehitzea eskatzen baitu, *f* hizkiarekin identifikatzen direnak. Erregistro berrien gehikuntzak inplementazioak konplikatzen ditu, baina hauen errendimendua hobetzen du eta komadun ISA hauek sinplifikatzen du, ez baitago erabiltzen diren erregistroak instrukzioetan ezberdintzeko beharrik, bit gehigarriak ekidinez.

ISAren F eta D aldaerak batera inplementatzen badira, *f* erregistroak 64 bitekoak izan beharko dira. Honela, doitasun sinpleko eragiketek erregistroen esangura gutxieneko 32 bitak erabiliko dituzte, doitasun bikoitzekoek 64 bitak erabiliko dituztela.

Instrukzioek komadun zenbakien arteko gehiketak, kenketak, biderketak, zatiketak edota konparaketak egiteko aukera ematen dute, konparaketak eta *load/store* aginduak zehazten dituztela ere balioak memorian gorde edo bertatik kargatzeko. Hondarraren kalkulua ez da inplementatzen.

Operazio aritmetikoetan, erroketak gauzatzeko instrukzio bat inplementatzen da gehigarri gisa. IEEEren 754-2008 estandarra betetzeko.

Bestalde, ISA honetan 32 biteko zenbaki osoak *float* sinple edo bikoitz bihurtzeko, *float* sinpleak bikoitz bihurtzeko eta aipatutako eragiketak alderantziz egiteko aginduak definitzen dira ere bai.

2.2.4 RV32A – Operazio atomikoetarako aukerazko ISA

ISA honek atazen exekuzio paraleloa ahalbidetzea du helburu, core ezberdinen sinkronizazioa ahalbidetzen duten operazio atomikoak aurkitu daitezkeela, atazek baliabideak atzitzeko lehia egoerak ekiditeko balio dutenak. Aukerazko modulu hau ere Fundazioak berretsi du. Bi operazio atomiko mota definitzen dira RV32A-n, jarraian aurkezten direnak.

Alde batetik, *Atomic Memory Operations* edo AMO motako operazioak bereizten dira, *read-modify-write* motakoak. Memoriako helbide batetik balio bat irakurtzen da erregistro batera, balio horri operazio aritmetiko bat aplikatzeko beste erregistro bateko balio batekin, azkenik emaitza erregistroan gordetako memoriako helbidean berriro idazteko. Operazioa atomikoa denez memorian idazketa/irakurketa egitean ezin da etenik egon edo beste core batek ezin du memoriako balio hori aldatu. AMO operazioak multiprozesatzea gauzatu dezaketen sistema konplexuetan erabilgarriak direlako gehitu ziren, baita ere I/O gailuekin komunikazioa busean atomikoa den transakzio bakarrarekin gauzatzea ahalbidetzen duelako.

Bestetik, operazio atomiko bat bi instrukziotan gauzatzen duten *load reserved* eta *store conditional* aginduak definitzen dira. Lehenengoak memoriatik hitz bat irakurri eta erregistro batean gordetzen du, memoria helbide horretan erreserba bat eginez. Bigarrenak word bat gordetzen du memoria helbide berean soilik helbide horretan *load reserved*-ek egindako erreserba existitzen bada. Arrakasta badu “0” idatziko du emaitza erregistroan, bestela “0” ez den errore kode bat gordeko du.

2.2.5 RV32C – Instrukzio konprimatuetarako aukerazko ISA

Fundazioak berretsitako luzapen hau oso praktikoa eta erabilia da, softwarearen aldetik abantaila oso handiak ematen baititu. ISA honek 32 biteko instrukzioak 16 bitekoak bihurtzen ditu, kodearen tamaina asko murrizten. Hau guztia konpiladorean eta mihiztatzaile lengoaian eraginik izan gabe lortzen da gainera, mihiztatzailea eta linkerra arduratzen baitira instrukzioen itzulpenaz. Beste ISA askotan gertatzen ez den bezala, konprimitutako instrukzio bakoitza RISC-V-ren 32 biteko estandar bakarrarekin mapeatzen da.

Konprimatutako instrukzioak hainbat premisa aintzat hartuz diseinatu ziren. Alde batetik, erregistro jakin batzuk beste batzuk baino askoz gehiago erabiltzen direla, bestetik instrukzio askok haien eragingaietako bat berridazten dutela eta azkenik instrukzio batzuk berehalako balio jakin batzuk erabiltzen dituztela kontuan hartu zen.

Hardwareari dagokionez, ohikoa da konprimatutako – ez konprimatutako instrukzioen arteko hardware deskodetzaileen inplementazioa. Gailu honek 16 bitetako instrukzioak 32 bitetako homonimora itzultzen ditu instrukzioa exekutatu baino lehen. Deskodetzaile honek ez du eragin esanguratsurik hardwarean, gutxi gora behera 400 ate logikorekin inplementatu baitaiteke.

Noski, C modulua I edota E bezalako oinarri modulu baten gainean aplikatu behar da, konprimatutako instrukzioak beste modulu bateko instrukzioetara deskonprimatzen baitira.

2.2.6 RV32E – *Embedded* motako oinarrizko ISA

Sistema txertatuetarako pentsatutako oinarrizko ISA honek erregistro kopurua 32tik 16ra (x0 – x15) gutxiagotzen du. Erregistroek RV32I coreetan diseinuaren tamainaren laurdena betetzen zutela ikusita, ISA hau kaleratzea erabaki zen.

Aipatutako sistemetan, tamaina, kostua eta energia efizientzia dira alderdi garrantzitsuenak, ahaltsuak diren coreak ez direla interesgarriak. Hori dela eta, tamaina edo arearen murrizketa ahalbidetzen duen ISA baten diseinua gauzatu zen, horrek amaierako diseinuen energia-kontsumoa gutxiagotzea dakarrelako, sistemen kostua eta efizientzia hobetzeaz gain. Abantaila horien trukean, E oinarrizko ISA duten coreek konputazio-gaitasun murritzagoak dituzte.

Oinarrizko ISA berretsi honek RV32I-k erabiltzen dituen instrukzio berdinak biltzen ditu, hauek x0 – x15 erregistroak erabiltzen dituztela bakarrik.

2.2.7 RV64I – 64 Bitetako instrukzioen oinarrizko ISA

RV64 RV32-rekiko ISA ezberdin bat da, 64 bitekin lan egin ahal izateko instrukzio gutxi batzuk gehitzen direla, erregistroen zabalera 64 bitetara handiagotzeaz gain. Erregistro kopurua 32n mantentzen da. Oinarrizko ISA hau berretsia dago ere.

Nahiz eta funtzionalitateen aldetik berdin-berdinak izan, RV32I eta RV64I-k ezberdintasunak dituzte bit mailan, hainbat instrukzio gehitzen dituela eta errepikatzen diren instrukzioen funtzionamendua ezberdina dela.

Nahiz eta RV64I-k 64 bitetako helbideekin eta, defektuz, 64 bitetako datuekin lan egiten duen, 32 bitetako *word*-ak onartzen ditu, RV32I-k *halfword* eta byte-ekin egiten duen bezala. Modu berean, RV64I-k batuketak, kenketak, desplazamenduak eta memoriatik irakurketak/idazketak inplementatzen ditu 32 bitetako datuekin.

Aukerazko luzapenek berdina egiten dute 32 biteko datuekin bateragarritasuna mantentzeko, RV64C luzapenean izan ezik, honek ez dituela agindu batzuk inplementatzen.

2.2.8 Bestelako moduluak

Orain arte deskribatutako ISAz gain, RISC-V-n beste hainbat ISA daude. Oinarrizkoen artean, 128 bitetako RV128I ISA aurki daiteke. Honen helburua konputazio baliabide handiak behar dituzten aplikazioak lirateke, hala nola kriptografia, aplikazio zientifikoak edota multimedia. Espezifikazio hau irekia dago oraindik, berretsi gabe dirauela.

Aukerazko aldaerei dagokionez, Fundazioak hainbat luzapen berretsi ditu. Kriptografia eskalarrerako ISA aurki daiteke, 32 eta 64 bitetako inplementazioetara zuzendua [7]. Bestalde, arkitektura bektorizatuen inplementaziorako V luzapena aurkitu daiteke. Luzapen honek datuen paralelismoa inplementatzea ahalbidetzen du, errendimendu altuko plataformetan aurki daitekeen ezaugarri bat. Aipagarria da hau dela luzapenik luzeena, 190 instrukzio inguru dituela.

Hauetaz gain, alderdi ezberdinak optimizatzeko luzapen gehiago aurkitu daitezke RISC-V-ren espezifikazioetan. Hauetako gehienak berretsiak egon arren, beste batzuk oraindik irekiak daude, helburua uneko konputazio sistemen beharrei erantzuten duten luzapenak kaleratzea dela.

2.2.9 RISC-V ISA Pribilegiatua [8]

Orain arte aurkeztu diren instrukzio eta erregistroak pribilegio gabeko edo erabiltzaile-mailari (*user-level*) dagozkio, aplikazio arrunten kodeak erabiliko dituen aginduak alegia. Hala ere, RISC-V-n pribilegio handiagoko modu bi daude; makina modua (*machine mode)* eta begirale modua (*supervisor mode)*. Modu biak eta Fundazioak berretsiak daude.

Modu bi hauek pribilegio gutxiagoko moduen baliabide berdinetara sarbidea izateaz gain, funtzionaltasun berriak inplementatzen dituzte: Eten edo interrupzioen kudeaketa, I/O kontrola, etab. Normalean, prozesadoreek denbora gehiago ematen dute erabiltzaile moduan, pribilegiodun modura gertaera gutxi batzuk aldarazten dutela.

Modu pribilegiatuak beraz, ezinbestekoak dira RISC-V estandarrarekin lerrokatzeko, sistema sinpleenek ere inplementatzen baitituzte etenak bezalako ezaugarriak. Noski, modu pribilegiatuaren inplementazioa sistema sinpleago batengatik ordezkatu daiteke, baina kasu hauetan, prozesagailuak ez dira *RISC-V compliant* izango.

Instrukzio gutxi batzuk eta kontrolerako erregistro batzuk (*Control Status Register* edo CSR erregistroak) osatzen dituzte modu hauek, bi modu pribilegiatu existitzen direla: Makina eta Begirale moduak. Jarraian, modu hauek aurkezten dira, makina modua sakonago aztertuko dela, begirale modua SEak daramatzaten sistema konplexuetan inplementatzen dela soilik.

Makina modua – M (*Machine)* modua

Makina edo M moduak memoria eta I/O-ra sarbide osoa du, sistema abiarazi eta konfiguratzeko funtzionaltasunak dituela. Hori dela eta, RISC-V prozesagailu estandar guztiek inplementatzen duten pribilegio modu bakarra dela esan ohi da, oinarrizko atazak betetzen baititu.

Ezaugarri aipagarriena salbuespenen eta etenen kudeaketa gauzatzea da. RISC-V-n etenak bi taldetan banatzen dira: Salbuespen sinkronoak eta asinkronoak. Sinkronoak kodearen instrukzio jakin batzuen exekuzioak eragiten dituzte, eta ondokoak izan daitezke:

- Sarbide akatsak: Memoria helbide fisiko batek sarbide mota ez badu onartzen (Adibidez, ROM batean idaztea)
- *Ebreak* instrukzioen exekuzioa
- *Ecall* instrukzioen exekuzioa
- Instrukzio ilegalak: Opcode baliogabeak dituzten aginduen exekuzioa edota soilik ‘0’-z edo ‘1’-ez osatutako aginduak
- Instrukzioaren helbide ez-lerrokatua

Asinkronoei dagokionez, hiru mota ezberdintzen dira:

- Software etenak: Memorian mapeatutako erregistro batean idazketak gertatzean.
- Denboragailu etenak: *mtimecmp* hardware erregistroko balioa *mtime* kontagailuaren berdina denean.
- Kanpoko etenak: Kanpoko gailuetara atzipena duen etenen kontrolagailu batek eten seinaleak sortzean.

Bestalde, RISC-V-n salbuespenen kudeaketarako hainbat erregistro definitzen ditu M moduak, lehen aipatutako CSRak. Zortzi dira, eta ondoko egoera-informazioa gordetzen dute.

- *mstatus*: Etenen gaitzaile globala gordetzen du, bestelako makinaren uneko informazioarekin batera.
- *mip:* Kudeatzeke dauden etenak gordetzen ditu.
- *mie:* Etenen maskara, zein eten kudeatu behar diren eta zeintzuk ez adierazten du.
- *mcause:* Exzepzio bat gertatzean, zein izan den adierazten du.
- *mtvec:* Exzepzio bat gertatzean zein helbidetara egin behar den jauzi adierazten du.
- *mtval:* Exzepzioen informazio gehigarria ematen du.
- *mepc:* Exzepzioa sortu duen instrukzioa adierazten du.
- *mscratch:* Exzepzioen kudeatzaileentzat XLEN luzerako erregistro tenporal bat eskaintzen du. Prozesadorearen erregistroetan idaztea ekiditen laguntzen du.

Salbuespen sinkrono bat gertatzean, salbuespena sortu duen instrukzioari dagokion PC balioa *mepc*-n gordetzen da, eten asinkrono batean aldiz etena kudeatu ostean exekuzioa zein helbidetan jarraitu behar den gordetzen dela. Jarraian *mtvec* PCan idazten da, *mtcause* eta *mtval*en salbuespenaren inguruko informazioa idazten dela. Etenak desgaitu egiten dira *mstatus* bidez eta pribilegio modua Mra aldatzen da.

Etenaren kudeaketa gauzatu ondoren, M moduko *mret* instrukzioa exekutatzen da, erabiltzaile modura itzultzeko *mepc* idazten duela PCan eta CSRen balioak berrezartzen dituela.

Eten bat beste bat gertatzen ari den bitartean gertatuz gero, kudeaketa inplementazioaren esku geratzen da. Azkenik, *wfi* instrukzioak RISC-V corea kontsumo baxuko moduan jartzeko balio du, modu honetatik itzartzeko corea eten bati itxaroten geratzen dela.

Aipatu den bezala, modu pribilegiatuek ez-pribilegiatuen ezaugarri berdinak dituzte, beraz sistema oso sinpleak soilik M modua inplementatuz diseinatu ohi dira. Hau ez da orokorrean gomendagarria baina, M moduak hardwarera muga gabeko sarbidea baitu. Exekutatu nahi den kodea guztiz fidagarria bada gomendatzen da soilik M moduaren inplementazioa.

Gainontzeko kasuetan, erabiltzaile (U) eta makina (M) moduen ezberdintzea gomendatzen da. Ezberdintze honekin, U moduak M moduaren *mret* agindua exekutatzean edo CSR erregistroetara sartzen saiakerak egitean salbuespenak sortzen dira. Bestalde, M moduak U moduaren memoriara sarbidea mugatu behar du, M moduak *Physical Memory Protection* edo PMP izeneko mekanismo batekin U moduak memoriako zein helbideetara duen sarbidea adierazten duela.

Begirale modua – S (*Supervisor*) modua

PMP mekanismoa egokia da sistema txikientzat, baina ez da eskalagarria aplikazio konplexuentzat. Kasu hauetan, prozesadoreek SE konplexuak exekutatzen dituzte, begirale edo S moduaren oinarria den orrietan oinarritutako memoria birtuala inplementatzen dela. S modua U modua baino pribilegiatuagoa da, baina M moduak baino pribilegio-maila gutxiago du.

Defektuz, M moduak kudeatzen ditu salbuespen guztiak, baina salbuespen sinkrono jakin batzuk S modura birbideratu daitezke *Makina Etenen Delegatzailearen* bidez (CSR *mideleg*).

Interrupzioen delegatzailearen bidez, M moduak SEari dagozkion salbuespen sinkronoak bidali ahal dizkio S moduari, hala nola SEari deiak diren *ECALL* motakoak.

2.2.10 RISC-V Mihiztatzaile lengoaia

Aipatu den bezala, RISC-V-ren erabiltzaile moduan 32 erregistro definitzen dira. Honi esker, aldagai asko mantendu daitezke erregistroetan, memoriara sarbideak ekidinez eta softwarearen errendimendua hobetuz. Erregistro hauetan erregistro tenporalak eta ez-tenporalen inplementazioak ere, funtzio laburrak mihiztatzaile lengoaiara itzultzea asko errazten du. Honekin, RISC-V ISAren diseinua softwarearen garapena errazteko pentsatuta dagoela ikusi daiteke.

Bestalde, RISC-V mihiztatzaileak ISAren aginduak luzatzen ditu mihiztatzaile lengoaian programatzea errazteko, pseudoinstrukzioak sortuz. Erabilienetako bat *li* edo *load immediate* instrukzioa da. Pseudoinstrukzio honen bitartez, berehalako balio bat erregistro batera eramaten da, RISC-V-ren berezko aginduekin hainbat agindu behar direla ataza honetarako.

RISC-V-n ere hainbat ABI definitzen dira, RV32ren kasuan *ilp32* eta RV64arenean *ilp64* direla. Komadun zenbakientzat luzapenak erabiltzen direnaren arabera ABI ezberdinak daude; RV32I-ren kasurako, *ilp32* ABIak softwareko *float* motako aldagaiak erregistro normaletan gordetzen dira, aldiz F eta D luzapenak inplementatu badira, *ilp32f* edo *ilp32d* erabiliz aldagai hauek komadun zenbakien erregistroetan gordeko dira.

2.3 Softcore prozesagailuak

Aurrerago ikusiko den bezala, lan honetan aipatu den RISC-V sistema bat FPGAtan inplementatu da. Gailu hauetan *VHDL, Verilog* edota *SystemVerilog* bezalako HDL (*Hardware Description Language*) lengoaia sintetizagarrien bidez inplementatzen diren prozesagailuei *softcore* deritzo, HDL softwarean oinarritzen baitira.

Noski, hau ez da FPGA batekin batera prozesagailu bat erabiltzeko modu bakarra. Alde batetik, saltzaile askok mikroprozesatzaile arrunt bat eta FPGA bat duten SoC-ak saltzen dituzte, mikroprozesatzaile hori *hardcore* bezala ezagutzen dela. Sistema hauetan, mikroprozesatzaileak FPGAk baino erloju frekuentzia handiago batean lan egin dezake.

Hala ere, *softcore*ak oso baliagarriak izan daitezke FPGAk erabiltzen dituzten sistemetan, hauen erabilerarekin, gailuan bertan inplementatu baitaiteke prozesadorea, hainbat abantaila emanez.

Alde batetik, prozesagailuek FPGAren baliabideak erabili ditzakete on-chip RAM memoria bat erabili ahal izateko eta honela memoriara atzipena asko hobetuz. Nahiz eta FPGA chipetik lortu daitekeen memoriaren tamaina txikia izan, nahikoa izan daiteke prozesadoreak ataza sinpleak betetzea nahi bada, hala nola FPGAren gainontzeko ataletan egon daitekeen diseinu baten kontrola. Honela, asko aprobetxatu daitekeen area gutxiko prozesadore bat inplementatu daiteke FPGAn, beste diseinuentzat lekua utziz.

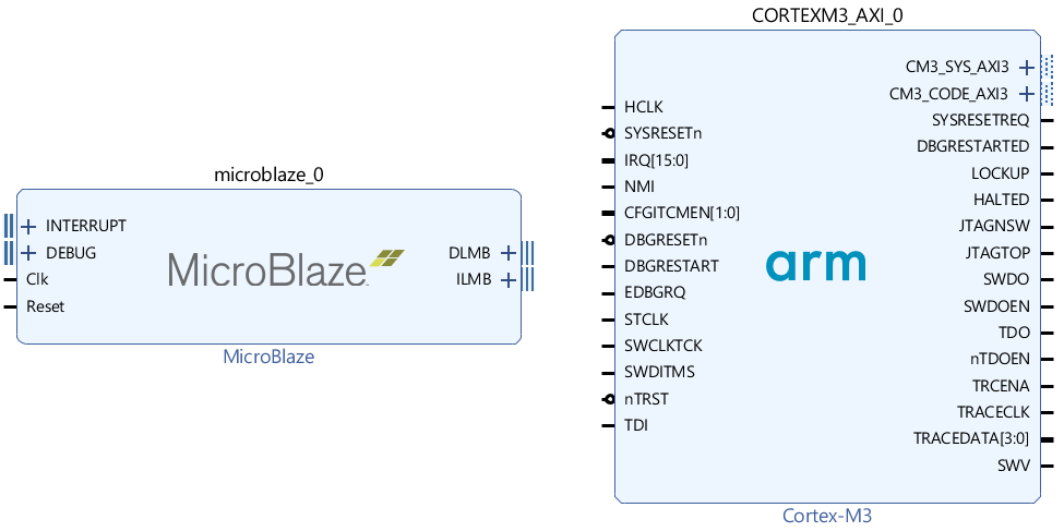
Bestalde, kontuan hartzekoa da prozesadore hauek ASICetan aurkitu daitzekeenak baino erloju abiadura txikiagoa dutela, hauen errendimendua orokorrean baxuagoa dela. Hala ere, desabantaila hau *multi-softcore* soluzioekin leundu daiteke, FPGAn lekua dagoen artean.

Tamalez, FPGA gailuak eta hauekin lan egiteko software tresnak historikoki bizpahiru enpresek eskainiak izan dira. Honen ondorioz, FPGAen mundua oso itxia izan da tradizionalki, garatzaileak enpresa hauek ematen dituzten tresnak erabiltzera mugatuta egon direla.

Enpresa hauek eskaintzen dituzten software tresnen artean, *softcore* prozesagailuen IPak egon dira. Xilinx/AMD enpresa estatubatuarrak *Microblaze* IPa eskeintzen du mota hauetako prozesagailuak behar dituzten FPGA diseinuentzat. Harvard arkitekturadun core hau 32 edo 64 biteko RISC makina pertsonalizagarria da, baina itxia denez, bere pertsonalizazioa mugatua da.

Xilinxen Vivado HDL diseinu tresna erabiliz instantziatu daiteke Microblaze, corearen konfigurazio ezberdinak eskaintzen direla alderdi ezberdinak optimizatuz. Errendimendua hobetzen duen diseinua eskaintzen da, bost etapako *pipeline* bat inplementatzen duenak. Area ahalik eta txikiena eskaintzen duen beste aldaera bat eskaintzen da ere, eta azkenik maiztasun maximoa hobetzen duen aldaera existitzen da, SE konplexuen inplementazioa ahalbidetuz, MMU edo cachedun kanpoko memoriekin lotzea posible eginez.

Bestalde, tradizionalki *hardcore* prozesadoreak diseinatzen dituzten enpresa askok haien diseinuak FPGAtan *softcore* gisa inplementatzeko IP produktuak eskaintzen dituzte. Hauxe da ARMen kasua, zeinak bere prozesagailuen zerrenda luze bat formatu honetan eskaintzen dituen. Nahiz eta Cortex M3 bezalako prozesagailuen diseinu batzuk lizentziarik gabe eskaini, beste modelo asko lortzeko lizentzia bat eskatzen da. Lizentziadun bertsioetan gainera, *core*aren informazio gehiago eskaintzen da, hala nola simulazio ereduak edota RTL fitxategiak. IP hauen pertsonalizazio maila oso txikia da gainera, haien egitura itxia baita.



2. Irudia: ARMen Cortex-M3 eta Xilinxen Microblaze prozesagailuen IPak FPGAk programatzeko Xilinx Vivado programan

Azkenik, soilik HDLan oinarritutako *softcore* irekiak aurkitu daitezke. Atal honetan aipatu den bezala, FPGA batean egin nahi den diseinu bat guztiz deskribatu dezake edozein HDL lengoaiak. Hori dela eta, VHDL edo Verilog bakunean oinarritutako *soft* prozesagailuen diseinua edonork egin dezake, pertsonalizazio maila hobezina dela. Noski, mota hauen barnean HDL lengoaia batean deskribatutako RISC-V coreak aurkitu daitezke.

3. Helburuak eta irismena

Atal honetan, proiektu honek dituen helburuak azaldu nahi dira, hauen deskribapenarekin proiektuaren irismena adierazi nahi dela aldi berean. Horretarako, lehenik eta behin proiektuaren helburu nagusia azalduko da, ondoren helburu nagusi hau betetzeko bigarren mailako helburuak deskribatuko direla.

3.1 Helburu nagusia

Proiektu honen helburu nagusia RISC-V ISA darabilen prozesagailu batean oinarritutako SoC baten diseinu bat gauzatzea da. Diseinua egiteko, hau etorkizunean ASIC batera eraman nahi dela kontuan hartu da, diseinuaren garapena asmo horren baldintzapean gauzatu dela.

Proiektu honetan ez da RISC-V prozesagailu bat diseinatu nahi, ataza honek denbora eta baliabide asko eskatuko lituzke eta. Honen ordez, jada sortutako eta egiaztatutako kode-irekiko prozesatzaile baten alde egin da.

Helburu nagusiarekin jarraituz, RISC-V prozesagailua lortu ondoren, honetan oinarritutako oinarrizko sistema bat sortu nahi da, hau nola luzatu eta moldatu aztertu nahi dela. Honela, ASICean inplementatu nahi diren gainontzeko IPak SoC-era akoplatzea ahalbidetu nahi da.

Azpimarratu beharra dago proiektu honen helburua ez dela ASIC horren gauzatzea, baizik eta bertan inplementatuko den SoC-aren HDL diseinua egitea, horretarako honen prozesatze-sistema diseinatu nahi izan dela. Proiektu hau beraz, aipatutako ASICaren sorreraren lehenengo pausua da.

3.2 Bigarren mailako helburuak

Helburu nagusia bete ahal izateko eta proiektuaren irismena guztiz zehazteko, jarraian ikusi daitekeen zerrendan bigarren mailako helburuen deskribapen xehea erakusten da.

1. **RISC-V ISAren aztertzea:** Lanarekin hasi baino lehen, RISC-V ISAren inguruko ezagutzen oinarri sendo bat izatea guztiz beharrezkoa ikusi da.

   Ezagutza hau guztiz ezinbestekoa izango da sistemaren prozesagailua aukeratzerako garaian, aukera bakoitzak eskaintzen dituen funtzionaltasunak ondo ulertzea ahalbidetzen baitu. Gainera, diseinatu nahi den ASICaren beharretara hobekien egokitzen diren RISC-V ISAren moduluen aukeraketarako guztiz beharrezkoa ikusi da moduluen eskaintza ondo ezagutzea, ezagutza hau RISC-V ISA sakonki aztertuz lortu daitekeela soilik.

   ISAren ulermena sisteman hedatu nahi den softwarea idazteko ere oso baliagarria eta ia derrigorrezkoa da, sistemara eramaten diren programen aginduak nahi direnak direla ziurtatu behar baita maila baxu batean, mihiztatzaile lengoaia dela sistemara eramaten den maila-baxueneko eta ulergarria den kodea. Hau noski, RISC-V instrukzioz osatzen da.

   Azkeneko honek garrantzi bereziki handia du *debugging* prozesuan, bertan exekuzioa instrukzioz instrukzio erakusten dela, hauek RISC-V mihiztatzaile lengoaian bistaratzen direla.

2. **Beharrezko moduluen aukeraketa:** RISC-V ISA eta bere modulu ezberdinak aztertu eta gero, APERT ikerkuntza taldeak diseinatu nahi duen ASICa buruan izanda, prozesagailuak inplementatu beharko dituen RISC-V moduluen aukeraketa gauzatu behar da.

   Horretarako, prozesagailuak ASICean izango duen funtzioa guztiz argitu behar da, horrela ikusi baitaiteke coreak zein operazio optimizatu beharko dituen. Gainera, SoC-aren frekuentzia eta area bezalako eskakizunak gutxi gora behera zehaztu beharko dira ere bai, SoC-a diseinatzerako garaian kontuan izateko.

3. **RISC-V-ren HDL inplementazio ezberdinen azterketa eta aukeraketa:** RISC-V-ren oinarri teorikoak ezarrita eta inplementatu nahi diren moduluak argi izanda, jada sortutako inplementazio ezberdinak aztertu beharko dira. Kode irekiko teknologia izanda, RISC-V *core* eta sistemen eskaintza oso zabala da, beraz existitzen diren inplementazioen artean bat aukeratu behar da.

   Aurreko pausuan egindako analisia oso garrantzitsua da, RISC-V modulu horiek inplementatzen eta bestelako eskakizunak asetzen dituen soluzio bat hautatu behar baita.

   Inplementazio bakoitzaren dokumentazioa sakonki aztertzea guztiz beharrezkoa da, erabaki bat hartzeko ezinbestekoa baita analisi hau. Ondoren, diseinua deskribatzen duen HDL kodea aztertu behar da, sistemaren inplementazioa nola egin den hobeto ulertzeko. Hau bereziki garrantzitsua da guztiz estandarrak ez diren *coreen* kasuan; RISC-V-ren malgutasuna dela eta, diseinu askok ez dituzte ISAen alderdi guztiak inplementatzen. Kasu horietan, dokumentazio aberats bat izatea oso garrantzitsua izango da.

   Prozesagailuaren HDL analisiaren barnean, kodea sintetizatu eta inplementatzen dituzten software tresnak erabili beharko dira ere, diseinu bakoitzak erabiltzen dituen baliabideak eta honen ezaugarriak (azalera, *timinga,* etab.*)* aztertu ahal izateko.

   Bestalde, software *toolchain* estandarrarekin bateragarriak izatea edo moldatutako *tooclhain* fidagarri bat eskaintzen den aztertu behar da, moldatutakoaren kasuan honen funtzionamendua eta instalazioa nola egin aztertu behar izan dela.

   HDL diseinuen funtzionamendua ulertu eta balidatzeko, diseinu hauen gainean hainbat simulazio gauzatu beharko dira ere, funtzionamendu seinale-mailan aztertu ahal izateko.

4. **SoC-aren garatzea eta funtzionaltasuna frogatzea:** Behin inplementazio bat aukeratuta, jada existitzen den edo aukeratutako prozesagailuan oinarrituz sortutako froga SoC diseinu sinple batetik abiatu beharko da.

   Behin diseinu sinple hau lortuta, honen garapena egin beharko da. Horretarako, periferiko berriak lotzea errazten duen bus bat gehitu beharko da, diseinura IP bloke berriak modu erraz eta fidagarri batean lotu ahal izateko.

   Sortutako SoC luzatua frogatzeko, HDL kodea FPGA batean sintetizatu beharko da, honela sistema martxan ikusi ahalko dela. Honetarako, diseinua FPGA konkretu batean erabiltzeko egin beharreko aldaketak gauzatu beharko dira.

   Pausu honetan garrantzitsua da inplementazioaren baliabide-erabileren emaitzen azterketa, erabilitako area bezalako parametroak esanguratsuak analisi honetan ondorioztatzen baitira.

5. **Standalone moduan abiatzeko mekanismo eta firmwarearen inplementazioa:** Garapenaren baitan, sistema bere kabuz abiatu ahal izateko mekanismoak gehitu beharko dira sistemaren *on-chip* memoria konfigurazioan.

   Bestalde, programa berriak modu erraz batean kargatu ahal izatea ezinbestekoa dela kontuan hartuz, firmwarearen diseinua garatu beharko da, honek abiaraztea modu automatiko batean egiten duela bermatuz.

   Abiarazte prozesu honetan, firmwareak prozesagailua modu egokian ezarri eta exekutatu nahi den programa modu egokian kargatzen duen aztertu behar da. Softwareari dagokionez, ASICean erabili nahi diren periferikoen erabilera zuzena lortzen dela bermatu beharko da, froga-programa sinple baten bidez.

4. Proiektuak dakartzan onurak

Atal honetan proiektu honek eta bertan erabilitako teknologia nagusien erabilerak arlo ezberdinetara ekarri ditzakeen onurak deskribatzen dira. Onurak hiru motetan sailkatu dira eta hurrenez hurren aurkezten dira; onura teknikoak, onura ekonomikoak eta onura sozialak.

4.1 Onura teknikoak

Lehenik eta behin, proiektu honetan egindako RISC-V inplementazio ezberdinen azterketa eta garatu den sistema, APERT taldeak sortu nahi duen ASICaren atal nagusietako bat izango dela azpimarratu behar da. Sistemara gainera, bestelako IP bloke batzuk gehitu nahi dira etorkizun batean.

Honela, lan honetan garatutako RISC-V sisteman oinarritzen den SoC bat sortuko litzateke, sistema honetan RISC-V coreak kontrolatutako bestelako core batzuk aurkitu ahalko direla. Sistema honen balioa ez litzateke prozesagailu sisteman egongo soilik, baizik eta aplikazio espezifiko batzuk betetzeko IP ezberdinak lotzen dituen SoC baten sorreran.

ASICaren prozesatze-sistemaren inplementazio honek RISC-V-ren abantaila nagusiak jarauntsiko lituzke ere bai. Erabiliko den RISC-V corearen inplementazioa kode irekikoa izango denez, garatu nahi den SoC-aren atal honen kode osoa eskuragarri egongo da sistema diseinatzean. Honek malgutasun handia emango du sistemaren diseinu digitala ASICera eramaterakoan, sistema xehetasun txikienera arte moldatu ahalko baita, IP propietarioen oztoporik gabe sortu nahi den SoC-erako guztiz egokia eta optimoa den prozesagailu bat erabili ahal dela.

Bestalde, prozesatze-sistemaren HDL kodea goitik behera eskuragarri izateak begi bistan ikusten ez den onura bat dakar ere bai: Segurtasuna. “Propietarioa” edo itxia den halako sistema baten diseinu bat erabiltzean, erabiltzaileak ez dauka begi bistan sistema horren barruan ezkutatuta egon daitezkeen atal maltzurrak detektatzeko baliabiderik.

Lan honetan erabilitako prozesatze sistemaren diseinua RISC-V Fundazioak berretsitako ISAren luzapenekin egin denez, etorkizunean garatu ahal diren luzapenak gehitzeko oinarri oso ona ematen du proposatzen den SoC-ak ere.

Azkenik, nahiz eta proiektuaren amaierako helburuetako bat FPGAtarako eraginkorra den RISC-V core baten diseinua ez izan, egindako diseinuak simulazioetatik haratago eraman dira FPGAtan inplementatuz. Honela, diseinu hau FPGA baten gainean *softcore* gisa erabiltzeko oso aukera interesgarria izan daitekeela argi ikusi da, bere pertsonalizazioa guztizkoa dela.

Berriro ere, HDL kodea guztiz eskuragarri izanda, diseinuaren eramangarritasuna bermatzen da, diseinua FPGA ezberdinetara eramatea posiblea dela. Hortaz, ia edozein FPGAtan sintetizatu daitekeen RISC-V SoC bat lortu da, *soft* prozesadore edo SoC-entzako prototipo-zonalde gisa erabili daitekeena.

4.2 Onura ekonomikoak

Proiektu honetan inplementatutako prozesadoreen erabilerak hainbat abantaila ekonomiko dakartza. Alde batetik, ISA irekia da, beraz ISA honetan oinarritutako inplementazioak gauzatzeko ez da inongo lizentzia-kosturik ordaindu behar. Lizentzia hauek oso garestiak izaten dira egun, eta ez dute RISC-V-k ematen duen diseinurako askatasun maila ematen.

Enpresei prozesamendu-sistemak diseinatzeko aukera ireki bat emanez, hauek gaur egun erabiltzen diren prozesadoreen atzean dauden enpresa gutxi horiekiko independentzia lortu dezakete. Honela, enpresen arteko lehiakortasuna asko sustatu daiteke, kasuan kasu arlo ezberdinetarako prozesadore ezberdinak diseinatu ahalko direla. Ondorioz, enpresek diseinu hauek komertzializatu edota haien diseinuetan oinarritutako sistemak kaleratu ahalko dituzte.

Aurreko azpiatalean aipatu den moduan, RISC-V-n oinarritutako prozesagailuak nahi diren atazak betetzeko pertsonalizatu daitezke. Honek beste ISA askotan gertatzen den arazo bat ekiditen du: Nahiz eta ISAren instrukzio asko ez erabili, hauek inplementatzeko hardwarea gehitu egin behar da, txiparen tamaina alferrik handiagotuz eta ondorioz, kostuak handiagotuz. RISC-V-n oinarritutako diseinuekin aldiz, prozesagailuek soilik beharrezkoak dituzten moduluak inplementa ditzakete, gailuen area eta errendimendu konputazional eta energetikoa hobetuz.

Bestalde, RISC-V ISA oso sinplea da erlatiboki. Ezaugarri horrek eragin zuzena du hardwarearen diseinuan, hau ere sinplea bihurtzen baitu. Berriro ere, honek txiparen tamaina hobetzen du, baina horretaz gain hardwarearen diseinurako esfortzuak murrizten ditu, garapen kostuak murriztuz.

Softwarea diseinatzerako garaian berdina gertatzen da, instrukzio sinpleek memoria erabilera txikiago bat suposatzen baitute. Txipetan inplementatu beharreko memoria azalerak eta ondorioz, kostuak, murriztu egiten dira beraz.

Gainera, ISA horren diseinuak denboran aldaezintasuna lortzea du helburu. Honela, RISC-V-n egindako diseinuek etorkizuneko software eta hardware inplementazioekin bateragarritasuna lortuko dela bermatzen dela. Hau bereziki onuragarria da software diseinatzaileentzat, haien softwarea ez dutelako printzipioz RISC-V inplementazio ezberdinentzat diseinatu behar eta.

Azkenik, RISC-V-ren garapena RISC-V Fundazio irekiaren esku dago. Fundazioaren kideak askotarikoak direnez, teknologia honen garapena ez dago enpresa bakar baten erabakien menpe, industria osoaren garapenaren menpe dagoela.

4.3 Onura sozialak

RISC-V-n oinarritutako SoC-en sorrerak onura oso garrantzitsuak eman diezazkioke gizarteari. Prozesagailuak gailu elektroniko askotan aurki daitezkeenez, proiektu honetan aurkezten den prozesatze-sistema nonahi aplikatu daiteke.

Lehen aipatu den moduan, teknologia honen onura teknologikoetako bat pertsonalizazioa da. Prozesagailuak osorik moldatzeko aukera ematen du, hauen eraginkortasuna ataza espezifikoetarako asko hobetu daitekeela. Honek gizartearentzako onura zuzen asko suposatzen ditu, errendimendu hobea eskaintzen dituzten produktuak kaleratzen lagundu dezake eta.

Errendimenduaren baitan, energia kontsumoaren alderdia kontuan hartzekoa da. Ataza ezberdinetarako neurrira egindako prozesagailuek errendimendu energetikoa optimizatzea ahalbidetzen dute, hauek ingurumenean duten inpaktua gutxiagotuz. Hau bereziki garrantzitsua da gaur egun, gailu elektronikoen energia kontsumo oso altua dela eta.

Bestalde, RISC-V-n oinarritutako prozesamendu-sistemek produktuen garapena merkatu dezakete, lehen aipatutako sinpletasunari esker. Honek produktu elektronikoen amaierako prezioen jaitsiera suposa dezake, produktu hauek gizarteari eskuragarriago ipintzen.

Azkenik, ISA hau oso baliagarria izan daiteke gizarteari prozesagailuen mundua gerturatzeko. Orain arte, sektore hau guztiz itxia izan da enpresen interesak direla eta, baina kode irekiko ISA batek prozesagailuen inguruan ikasi nahi duen pertsona orori ezagutza hau gerturatzen dio, edonork ikasi eta esperimentatu dezakeela ISA honen inplementazioarekin. Hau batez ere baliagarria izan daiteke irakaskuntza zentro espezializatuetan, hala nola Unibertsitate edo Lanbide Heziketa zentroetan.

Kode irekikoa izateak ere, teknologiaren garapenean pertsona gehiagoren parte-hartzea suposatzen du. Honek teknologia honen garapena ziurtatzen du, ondorioz errendimendu hobea duten produktuak denbora laburrago batean bultzatuz, gizarteari lehen aipatutako onurak emanez.

5. Artearen egoera

Sarrera atalean aipatu den bezala, RISC-V ISAk adopzio handia lortu du industria mailan, ISA honetan oinarritutako prozesagailuen inplementazio asko aurkitu daitezkeela aplikazio-eremu ezberdin askotan. Honen arrazoi nagusia RISC-V-ren malgutasuna eta bere atzean dagoen komunitatea dira, teknologia hau bultzatzen duen erakundearen atzean industria anitzetako enpresa garrantzitsuak daudela.

5.1 RISC-V Inplementazio itxi edo propietarioak

Kode itxiko inplementazioek RISC-V-ren egoera argi islatzen dute. Kasu aipagarrienetako bat RISC-V-ren sortzaileen SiFive enpresarena da, zeinak munduko RISC-V core aurreratuenetakoak sortzen dituen. *Fabless* eredua jarraituz, enpresaren eskaintzen artean IP coreak aurkitu daitezke gehienbat, haien prozesadoreen inplementazio ezberdinak dituzten ebaluazio plakak eskaintzen direla ere.

Enpresa honen coreen eskaintza oso zabala da. Alde batetik, prozesagailu ahaltsuen IP coreak eskaintzen ditu, hauen artean erabilera orokorrerako sistemetara zuzendutako prozesatze-sistema ezberdinak aurki ditzakegula. Sistema hauek, multi-core konfigurazioak eskaintzen dituzte, 13 pipeline etapako eta 16 corera arteko sistemak eskaintzen direla.

Errendimendu altuko koreen barnean ere, *machine learning* aplikazioetara zuzendatuko IP *core*ak eskaintzen ditu enpresak. IP hauek gehienez 8 *core*z osatutako konfigurazioak eskaintzen dituzte, hauek 64 bit eta 8 etapetako *pipeline*a duten RISC-V coreak direla. Core hauek RISC-V bektoreen luzapenak inplementatzen ditu (V luzapena), konputazio bektorialaren inplementazio honen bidez sare neuronalen prozesatzea azeleratuz. Bestalde, *machine learning* edota errealitate birtualerako 64 bitetako bestelako coreen eskaintza oso zabala da, errendimendu – area konpromisoaren arabera gaitasun gutxiagoko baino area txikiagoko inplementazioak eskaintzen direla ere.

Bestalde, *embedded* aplikazioetarako 32 bitetako *core* mugatuagoak eskaintzen dira ere. Hemen ere*, multi-core* (4 corerarte) sistemak aurki daitezke, IP core sinpleena soilik 13.500 ate logikoz osatuta dagoela.

Nahiz eta SiFive-ren kasuak RISC-V prozesagailu aurreratuenen ezaugarriak ikusten utzi, ez du RISC-V-ren adopzio maila erakusten. Hau ondo ikusteko, adibide oso interesgarria da Espressif enpresarena. Enpresa honek sortutako ESP8266 edo ESP32 SoC-ak mundu-mailan erabilienak bilakatu dira mikrokontrolagilu aplikazioetarako, haien efizientzia eta prezioa direla eta.

Honela, Espressifek ESP32-C3 SoC-a kaleratu zuen 2020. urtean, SoC-aren prozesagailua 32-biteko RISC-V core bakar bat dela [9]. ESP32-C3-arekin sinpletasuna maximizatu nahi zuen enpresak, horretarako soilik IMC luzerak inplementatzen dituen prozesagailu txiki baten alde egin zuela. Orduz geroztik, enpresak RISC-V-ren alde eginez jarraitu du, 2023. urtean RISC-V-n oinarritutako ESP32-P4 errendimendu altuko *dual-core* mikrokontrolagailua kaleratuko dela adierazi zuela.

RISC-V haien prozesagailu nagusi bezala inplementatzen duten kasu famatuez gain, munduko enpresa garrantzitsu askok RISC-V sistemak osagai laguntzaile edo azeleratzaile bezala inplementatu dituzte haien diseinuetan. Esate baterako, NXP, NVIDIA edota Apple bezalako enpresek haien produktuetan RISC-V coreak inplementatzeko asmoak erakutsi baitituzte azken urteetan.

Biltegiratze sistemetan espezializatutako Western Digitalen produktuetan jada RISC-V coreak aurkitu daitezke, *SweRV Core EH1* prozesagailua hain zuzen ere. Enpresak corea kode ireki bezala eskaintzen du bere GitHubeko biltegian [10].

5.2 Kode irekiko RISC-V inplementazioak

Bestalde, enpresa bakar baten menpe ez dauden RISC-V-ren kode irekiko inplementazio anitz aurkitu daitezke. Kasu honetan, alde batetik prozesagailuen inplementazio irekiak aurkitu daitezke, eta bestetik, prozesagailu hauetan oinarritutako SoC osoen diseinuak aurki daitezke ere.

5.2.1 Berkeley Unibertsitatearen inplementazioak

Arlo honetan, inplementazio aurreraturikoenetakoak, ISA beraren sortzailea den Berkeley unibertsitateak gauzatu ditu. Alde batetik, *out-of-order* motako BOOM (*Berkeley Out-of-Order Machine)* kode irekiko corearen hardware inplementazioa eskaintzen du Unibertsitateak [11], honen hirugarren bertsioa (SonicBOOM [12]) 7 etapako 64 biteko RV64GC prozesadore bat dela. Inplementatzen dituen ISAren luzapenek adierazten dutenez, SonicBOOM makina helburu orokorrerako konputaziorako pentsatutako errendimendu altuko corea da, bere erabilera datu zentro edota PC aplikazioetara zuzenduta dagoela.

Unibertsitate berak *in-order* inplementazio "sinpleago" bat sortu zuen, *Rocket Core* izenekoa. 5 etapako core hau RV32G edota RV64G bertsioetan aurkitu daiteke, bere erabilera nagusia helburu orokorrerako konputazioa dela. Datu eta instrukzio L1 *cache*arekin batera, *Rocket tile* bat osatzen da, *Rocket* core multi-prozesagailu arkitektura bat sortzeko unitatea alegia.

Unibertsitateak berak, bai *Rocket* corean bai BOOM corean oinarritutako SoC diseinuak automatizatzeko *Rocket chip* plataforma sortu zuen [13]. Tresna honek, definitutako liburutegi batzuetako osagaiekin SoC-en diseinuak *plug-and-play* moduan egiteko aukera ematen du, diseinu prozesu hau automatizatu eta erraztuz. Plataformak *Rocket* corean, BOOM corean edo beste open source core bitan (CVA6/Ariane edo Ibex/zero-riscy) oinarritutako SoC-en diseinurako liburutegiak ematen ditu. Azkeneko core bi hauen erabilera, SoC-en sorrerarako plataforma honetan prozesagailu sinpleagoetan oinarritutako sistemen diseinuetara zuzendua dago.

Aipagarria da *Rocket chip* plataformaren oinarria Berkeley unibertsitateak ere garatutako *Chisel* lengoaiak osatzen duela, lengoaia hau erabiltzen baita sistema hauen diseinuan. Hau *Scala* lengoaian txertatutako HDL lengoaia bat da, FPGA eta ASICetarako zirkuitu digitalen diseinu aurreratuak ahalbidetzen dituenak, hauen berrerabilera errazten duela ere. Nahiz eta *Rocket Chip* eta *Chisel* Berkeley Unibertsitatean sortuak izan, gaur egun irabazi-asmorik gabeko *CHIPS Alliance* erakundeak kudeatzen ditu.

5.2.2 PULP Plataforma

Ikusi daitekeen moduan, Berkeley Unibertsitatearen kode irekiko softwarearen bitartez, core ezberdinen inplementazioa lortzeaz gain SoC-ak garatzeko ingurune bat jartzen da eskuragarri garatzaileentzat. Antzeko modu batean, ETH Zürich eta Boloniako Unibertsitateak 2013. urtean PULP (*Parallel Ultra Low Power*) plataforma sortu zuen, honen helburua energia kontsumo baxuko prozesamendu arkitektura berrien garapena zela.

Urteak pasa ahala, proiektua asko zabaltzen joan da bere arrakasta izan dela eta. Honela, PULP proiektuak core bakarreko, multi-core edota errendimendu altuko konputaziorako (HPC) azpi-proiektuak gauzatu dituela.

Azkeneko arlo bi hauetan, aipagarria da PULP plataformak eskaintzen zuen Ariane corea, gaur egun OpenHW Group taldeak kudeatzen duela CVA6 izenarekin. 6 Etapetako core hau erabilera orokorrerako dago pentsatuta, multi-core sistemen parte izateko pentsatuta dagoela. Ez hori bakarrik, core honetaz osaturiko hainbat multi-core konfigurazioez osatutako konputazio-clusterrak sortzeko dago diseinatua.

Bestalde, konputazio-ahalmen baxuagoko sistemetarako coreak aurki daitezke ere PULP plataforman. Core hauek mikrokontrolagailu edota kontsumo baxuko aplikazioetarako oso egokiak dira, haien tamaina eta ezaugarriak direla eta. Honela, arlo hauetan kode-irekiko aukera interesgarrienetako bat bihurtu dira kore hauek.

Zehazki, PULP proiektuak *RI5CY* eta *zero-riscy* 32 biteko RISC-V coreak (gaur egun *CV32E40P* eta *Ibex,* hurrenez hurren) eskaintzen ditu. Lehenengoak IMCF luzapenak inplementatu ditzake, bigarrenak IMC luzapenak eskaintzen dituela. Core bakoitzak dituen luzapenak erabiltzailearen beharren arabera aukeratu daitezke inplementatzerako garaian.

Coreak soilik eskaintzeaz gain, PULP proiektuan core hauetan oinarritu daitekeen SoC oso baten diseinu bi aurkitu daitezke: PULPino eta PULPissimo. SoC hauek hainbat periferiko dituzte memoria eta prozesagailuaz gain, haien diseinua ASIC batera eramateko pentsatuta dagoela, testing modura FPGAra eramateko aukera ematen dela ere. SoC-aren osagai guztiak PULP plataformako kideek eginak dira.

Core bi hauen potentzial handia arrazoi gisa, ETH Zürich eta Boloniako Unibertsitateak haien garapena beste erakunde batzuen esku utzi zuen. RI5CY-ren kasuan, corearen garapena OpenHW Group erakundearen esku dago gaur egun, zeinak CV3240P izena eman dion. Zero-riscyren kasuan, LowRISC konpainiak hartu du erreleboa, Ibex bezala berrizendatu duenak.

Erakunde bi hauek irabazi-asmorik gabekoak dira, haien helburua kode irekiko hardware diseinu eta hauentzako software tresnak garatzea eta mantentzea dela, hauen biziraupena mantentzeko helburuarekin. RISC-V fundazioaren kasuan bezala, erakunde hauek enpresa pribatuak dituzte kide bezala, biziraute hori posible egin ahal izateko. Enpresa hauen artean erdieroaleen industrian oso garrantzitsuak diren Intel edota Google bezalako enpresak aurki daitezke.

5.2.3 Bestelako inplementazioak

RISC-V-ren izaera irekia dela eta, erakunde hauetatik at enpresa txiki edota profesional ezberdinek egindako RISC-V core asko aurkitu daitezke GitHub bezalako biltegietan. Nahiz eta alternatiba hauek ez duten aipatutako erakundeen “babes-industriala” edota kalitate-maila, aukera interesgarriak dira helburu sinpleetarako. Ez hori bakarrik, mota hauetako prozesagailu gehienek ez dute RISC-V estandarrarekin guztiz betetzen, eskaintzen dituzten luzapenak ez dituztela guztiz betetzen edota RISC-V-ren modu pribilegiatua bezalako alde konplexuak ez dituztela inplementatzen.

5.3 RISC-V Prozesatzaileentzako softwarea

Hardwareaz gain, RISC-V gailuetan softwarea exekutatzeko tresnen sorrera ere guztiz oinarrizkoa da. Azken finean, programak RISC-V-n oinarritutako prozesagailuen hardwareak exekutatu ditzakeen instrukzioetara itzultzea guztiz beharrezkoa da, honetarako konpiladoreak bezalako tresnak behar direla.

RISC-V prozesagailuentzako softwarea sortzeko eskuragarri dauden tresnetan ikusi daiteke ere teknologia honen adopzio maila. Noski, prozesagailu hauen amaierako helburua softwarea exekutatzea denez, oso garrantzitsua da softwarearen aldetik hainbat tresna eskura izatea; RISC-V prozesagailuak gaur egun erabiltzen diren aplikazioak exekutatzen ikusi nahi badira, software hauek prozesagailu hauetan exekutatzeko modu bat egon behar da.

Softwarearen garapenerako *toolchain*ei dagokionez [14], RISC-V munduan erabiliena GNU proiektuak garatutako GCC *toolchaina* da, *toolchain estandarra* bezala ezagutzen dela *de facto*. *Toolchain* honek C edo C++ lengoaietan idatzitako kodea RISC-V ISAn oinarritutako prozesagailuetarako konpilatzeko aukera ematen du, 32 eta 64 bitetako arkitekturekin bateragarria dela.

Toolchain honek IMAFDC luzapenak onartzen ditu ofizialki, sistema txertatuetan erabili ohi den 32 bitetako E luzapena onartzen dela ere. Hala ere, *toolchaina* bestelako luzapenak onartzeko aldatu daiteke, nahi bezain beste pertsonalizatu daitekeela RISC-V prozesagailu espezifikoetan eraginkortasun maximoa lortzeko, esate baterako instrukzio pertsonalizatuak sortuz.

Bestalde, GCCrekin batera erabili ohi diren GNU proiektuaren bestelako softwareak eskuragarri daude RISC-V-rentzat, software garapenean ezinbestekoak diren tresnak hain zuzen ere. Esate baterako, GNUren GDB debugger aski ezaguna, GNUren *linker* eta mihiztatzailea eskaintzen duen GNU Binutils tresna; GNUren GLIBC C lengoaiako liburutegia eta bestelako asko.

Ikusi daitekeenez, GNU proiektuak RISC-V-rentzako software *suite* zabala garatu du, tresnak gainera egunean mantentzen dituztela funtzionaltasun berriak gehitzen eta maiz akatsen zuzenketarako eguneraketak kaleratzen. Hau, GNU proiektuak *open-sourcean* eta teknologia munduan duen garrantzia kontuan hartuta, oso esanguratsua da eta asko dio RISC-V-ren heldutasun mailari dagokionez.

LLVM *toolchaina* eskuragarri dago ere RISC-V-rentzat. *Toolchain* honek GCCk onartzen dituen luzapen berdinak onartzen ditu, E luzapena izan ezik. Hala ere, LLVMk RISC-V estandarrean definitutako beste luzapen batzuk gehitzen ditu, hala nola bit-manipulaziorako, bektore operazioetarako edota zfh luzapena bektore operazioetan komadun zenbakietarako. LLVMren gehikuntza oso garrantzitsua izan da, RISC-V prozesagailuetan *Rust* lengoaiaren erabilera ahalbidetzen baitu.

Bestalde, Go, edo PHP lengoaiek edota V8 nabigatzaile motorrak haien *port*ak egin dituzte RISC-V-n erabili ahal izateko, software ekosistema nahiko osoa lortuz teknologia honetan [15].

Haratago joanda, hainbat sistema eragile nagusik ere haien produktuak RISC-V-ra eraman dituzte. BSD sistema eragileari dagokionez, FreeBSD eta OpenBSD BSD sistema eragile irekiek haien RISC-V bertsioa eskaintzen dute. Bestalde, oso garrantzitsua den Linux Kernela RISC-V-ra eraman da ere, ondorioz Linux distribuzio askok haien RISC-V bertsioak eskaini dituztela, bai *embedded* mundurako zuzenduak (Buildroot, Yocto) baina baita ere konputazio orokorrera edota errendimendu altuko konputaziora zuzendutako distribuzio garrantzitsuak, hala nola *openSUSE*, *Fedora*, *Arch Linux* edota *Debian* distribuzioak.

6. Alternatiben analisia

Atal honetan proiektuan sortu nahi den diseinua osatzen duten osagaien alternatibak aurkeztu eta aztertuko dira, helburua proiektuaren xedera gehien egokitzen diren teknologien aukeraketa gauzatzea dela.

Zehazki, proiektu honetan aztertu behar izan den atal nagusia SoC-aren oinarri izango den prozesatze-sistema izan da, hori dela eta, lanaren atalik garrantzitsuenetako bat irudikatzen du alternatiben analisiak.

Aurrerago ikusiko den bezala, aurkitu izan diren alternatiba batzuk prozesagailua eskaintzeaz gain honen inguruan bestelako blokez osatutako sistemak proposatzen dituzte, prozesagailua nola integratu argitzen dutela. Proposamen hauek oso interesgarriak izan ohi dira, alternatiba batekin lanean hastea asko errazten baitute.

Dokumentu osoan zehar aipatu den bezala, RISC-V-k duen ezaugarri erakargarrienetako bat bere *open-source* izaera da. ISA aske batek edonori ematen dio hau inplementatzeko aukera, aldi berean, inplementazio hauek publiko orokorrari eskura jartzea ahalbidetzen duela. “Artearen egoera” atalean ikusi den bezala, hau errealitate bihurtu da gaur egun; GitHub bezalako biltegietan hainbat RISC-V-n oinarritutako prozesagailu eta sistemen eskaintza zabala aurkitu daitekeela.

Noski, hau, abantaila izateaz gain, desabantaila bezala ulertu daiteke ere. Eskaintza hain zabal batek analisi sakon bat eskatzen dio RISC-V prozesatze-sistema bat aukeratu nahi duen pertsona orori. Hasteko, prozesatze-sistema bakoitzak eskaintzen dituen ezaugarriak kontuan hartu behar dira: Luzapen zehatzen inplementazioa, tamaina, abiadura, memoria-interfazeak, etab.

Ez hori bakarrik, diseinu batzuk beste batzuk baino fidagarriagoak izan daitezke. Fidagarriagoa den RISC-V prozesagailu batek diseinu ahalik eta optimoena izango du eta bere hardwarearen deskribapenean akats gutxiago izango ditu. Akats hauek murrizteko, ezinbestekoa da diseinuaren *verification* edo egiaztapena gauzatzea, ezinezkoa baita diseinu bat fidagarria dela segurtasunez esatea honek egiaztapen prozesu sakon bat jasan ez badu. Egiaztapen honek baliabide asko behar ditu; denbora asko eskatzen du prozesagailuaren atal bakoitza simulatu behar baita eta simulaziorako garestiak diren tresna eta hauek exekutatzeko ekipo kapazak behar baitira.

Gainera, prozesagailuaren hardware diseinuari etekin maximoa atera nahi baldin bazaio, honetan hedatuko den softwareak hardwarea ahalik eta modu eraginkorrenean erabili beharko du, pertsonalizatutako instrukzioak erabiliz. Honek C bezalako maila altuko lengoaietan idatzitako softwarea optimoak diren instrukzioetara itzultzen duen software *toolchain* pertsonalizatu baten sorrera eskatzen du, oso konplexua izan daitekeen ataza bat.

Aipatu beharra dago RISC-V-n oinarritutako alternatiba itxiak ere existitzen direla. Aurreko ataletan aipatu den moduan, RISC-V prozesagailuen diseinuak egiten dituzten enpresa pribatu askok haien produktuak IP izeneko kutxa beltzetan eskaintzen dituzte, pertsonalizazio aukera gutxi batzuk eskainiz.

Hala ere, proiektu honetan kode-irekiko alternatiben alde egin da, alde batetik, hauen erabilera bultzatu nahi izan delako. Bestalde, kode guztia eskuragarri izanda, diseinuak hobeto aztertu eta ulertu izan ahal dira, prozesagailuak guztiz pertsonalizatzeko aukera emanez ere bai. Azkenik, kode osoa eskura izatea oso ezaugarri garrantzitsua izan da aukeraketarako, jakina denez, proiektuarekin lortu nahi den diseinua ASIC batera eraman ahal izateko onuragarria bezala baloratu baita iturri kode osoa izatea.

6.1 Prozesagailuaren aukeraketarako irizpideak

Eskuragarri dauden kode-irekiko aukeren artean bat aukeratzeko, alternatiba bakoitzaren hainbat alderdi hartu dira kontuan, jarraian aurkezten direnak. Gainera, alderdi edo aukeraketa-irizpide hauen pisua adieraziko da, aurrerago azalduko den bezala, erabaki bat hartzeko kontuan hartu dena.

- **Proiektuarekiko egokitasuna (% 50)**: ASICera eraman nahi den SoC-aren diseinuan periferiko batzuk kontrolatzen dituen sistema bat nahi denez, ez da beharrezkoa izango konputazionalki oso kapaza den prozesadore bat, baizik eta kontrol ataza hauek modu eraginkorrean egiten dituen softwarea exekutatzeko gai den prozesagailu bat beharko da. Beraz, helburu honetarako mikroprozesagailu bat izango litzateke aukera egokiena.

  Hala ere, mikroprozesagailua ezingo da edozein motakoa izan. Honen eta sistemaren azalera edo area txikia izan beharko da, ASICaren kostua murriztuz ahal den heinean. Noski, ezaugarri hau prozesatze-sistemaren errendimendu konputazionalarekin orekatu beharko da, prozesagailu azkar eta efiziente bat aukeratu nahi dela ere.

  Sistemaren azalera eta gaitasunarekin lotuta, prozesagailu alternatiba ezberdinek inplementatzen dituzten RISC-V luzapenak aztertu beharko dira. Prozesagailuari eman nahi zaion erabilera kontuan hartuz, IMC luzapenak egokiak direla baloratu da, B luzapena interesgarria izan daitekeela ere bai. Nahiz eta F motako luzapenek programa jakin batzuk errendimendu hobean exekutatzeko ahalmena eman, luzapenaren inplementazioak dakarren hardware gehikuntza gehiegizkoa bezala baloratu da.

  Ezaugarri hauek analizatu ahal izateko, diseinuaren inplementazioaren baliabideen erabilera edota sistemak programak exekutatzerako garaian duen errendimendua behatu beharko da. Prozesagailuaren arkitekturaren azterketa garrantzitsua izango da ere.

  Diseinuek FPGA gailuetan frogatzeko aukera eman behar dute, diseinua fisikoki martxan ikusteko. Hau batez ere garrantzitsua izango da softwarea frogatzeko, hau EDA tresnen bidez frogatzea astuna eta konplikatua izan daitekeelako. Gainera, HDL diseinuak FPGA gailuak programatzeko tresnekin aztertuz, diseinuen baliabideen erabilera edota *timinga* bezalako alderdiak inplementazio fisiko batekiko modu hurbildu batean aztertu daitezke.

  Ez da erabakigarria izango, baina kontuan hartu beharko da diseinuak zein FPGA gailu edota plakarako dauden eskuragarri, gailu garesti hauek erostea ekidin daiteke eta. Kasu honetan, proiektua egiterako garaian Avnet etxearen ZedBoard (Xilinx Zynq SoC-a) eta Arty (Xilinx Artix-7 35T FPGA) plakak zeuden eskura.

Hori bai, FPGA gailuetarako zuzendutako diseinuek ezingo dituzte gailu hauetara zuzendutako IPak eduki, bloke normalean FPGA gailu zehatzetan aurkitu daitekeen hardware espezifikoa erabiltzeko pentsatuta daudelako.

Pisu altua eman zaio irizpide honi, azkenean proiektuaren bideragarritasuna eta zentzua definitzen baituelako; handiegia den prozesagailu bat ASICera eramatea garestia izango litzateke eta helburu diren aplikaziorako kontsumo handiegia izango luke.

- **Fidagarritasuna (% 30):** Lehen aipatu den bezala, diseinu ezberdinek fidagarritasun maila ezberdinak ematen dituzte hauen osotasunaren inguruan. Kode-irekiko alternatiba bakoitzaren atzean dagoen erakundea edota alternatiba bakoitzak duen arrakasta kontuan hartzea oinarrizko irizpide bat bezala hartu da aukeraketa bat egiterako garaian.

  Alternatiba bakoitza aktiboa den edo ez aztertu beharko da; baliteke kode-irekiko prozesatzaile baten diseinua urteetan aldatu ez izana honi lotutako proiektua abandonatuta utzi bada, esate baterako.

  Alderdi hau neurtzeko beste adierazle kritiko bat diseinuak duen egiaztapen-maila izango da. Aurretik azpimarratu den bezala, egiaztapena diseinu bat egokia dela bermatzeko prozesu ezinbestekoa da, honek izan ditzakeen akatsen identifikazioa errazten baitu, sakonki egiaztatutako diseinu baten fidagarritasun maila asko igoz.

  Egiaztapen-maila kuantifikatzeko, diseinuak duen egiaztapen azpiegitura aztertu beharko da, helburu honetarako eskaintzen dituzten tresnak behatu beharko direla ere.

  Atal honetan diseinuak softwarearen aldetik duen fidagarritasuna baloratu da ere. Diseinuak pertsonalizatutako *toolchain* baten erabilera egiten badu, honen funtzionamendua zuzena dela ere egiaztatuta egon beharko da, *toolchain* akasdun baten erabilerak programen konpilazioan erroreak ekarri ditzake eta, hauen exekuzioa galaraziz.

  Irizpide honi ere pisu handia eman zaio, ASICera eramaten den diseinuak ahalik eta akats gutxien izan behar baititu. Esan den bezala, akats esanguratsuak dituen diseinu batek lan honetan eta ASICa sortzeko emango diren etorkizuneko pausuetan egingo diren esfortzu guztiak bertan behera utziko bailituzke eta.

- **Inplementaziorako gida-lerroak izatea (% 20):** Alde batetik, erreferentzia-gida zehatz baten eskaintza guztiz ezinbestekoa izango da diseinu baten analisia egiteko. Gida argi baten bitartez, diseinu batek eskaintzen dituen aukerak modu azkar batean ezagutu daitezke, diseinuarekin lanean jartzerako garaian zalantza gehienak argitzen laguntzeaz gain.

  Bestalde, oso lagungarria da ere prozesagailuaren inplementazioa nola gauzatzen den ulertzeko erreferentziazko diseinu bat eskaintzea. Honek prozesagailuaren interfaze ezberdinen funtzionamendua argitzen du, bestelako helburuetarako berrerabili daitekeen kode zatiak erakusten direla ere.

  Nahiz eta aurkeztutako irizpideen artean pisu gutxiena jaso, hau ere garrantzitsua da diseinu zurrun eta sendo bat sortzeko. Hori dela eta hartu da kontuan, bere pisua esanguratsua dela ere.

6.2 Aukeren deskribapena

Aipatutako irizpideen azalpenetan esandakoan oinarrituta, eskuragarri dauden kode-irekiko alternatiba gutxi batzuk aukeratu dira. Zehazki, PicoRV32, CV3240P eta Ibex prozesagailuak eta RI5CY edota Zero-riscy-n oinarritu daitekeen PULPino sistema hautatu dira, haien artean alderatu direla aipatutako irizpideetan oinarrituta hauek hobekien betetzen dituen hautagaia aukeratu dela.

Nahiz eta “artearen egoera” atalean Berkeley Unibertsitateko prozesagailuak aipatu izan diren, hauek guztiz baztertu dira proiektu honetatik, konputazio orokor edota errendimendu altuko konputaziora zuzendutako diseinuak baitira. Unibertsitate hau RISC-V ISAren jaioleku izanda, bertan sortutako mikroprozesagailu baten diseinua lortzea interesgarria litzateke, baina haien ikerkuntza esfortzuak beste motako prozesagailuetan jarri dituztenez, ezin da mikroprozesagailurik aurkitu Unibertsitatearen kode-irekiko diseinuen biltegietan.

Aldiz, hautagai gehienak ETH Zürich eta Bolognako Unibertsitateak sortutako elkartean dute jatorria: PULP plataforman. Bertan IoT edota baliabide gutxiko inguruneetara pentsatutako hainbat prozesagailu aurkitu daitezke, PULP taldeak urteak daramatzala prozesagailu hauen garapenarekin. Lehen aipatu den bezala, urteak aurrera egin ahala, PULP taldeak haien diseinuen mantentzea eta garatzea bestelako erakunde batzuen esku utzi du. Erakunde hauek ataza hauek etengabe gauzatzen dituztenez, prozesagailuen diseinu egonkorren eskaintza dute, hauen eguneraketak maiz aurki daitezkeela.

Bestalde, PicoRV32 bezalako erakunde handi batetik kanpo dagoen aukera baten alde egin nahi izan da, RISC-V-ren kode-irekiko izaera erakusten duen aukera bat aztertzeko asmoz.

Hurrengo orrietan alternatiba hauek aztertuko dira, kandidatu bakoitzak aipatutako hiru alderdi horiek zeinen ongi betetzen dituen aztertzea izango dela helburua. Amaieran, proposatutako aukeren artean bat erabakitzeko, irizpide hauen betetze-maila kuantifikatuko da, taula baten bitartez zenbatetsiko dela alternatiba bakoitzaren irizpideen betetze-maila, 0tik 10-era nota bat ipiniz irizpide bakoitzeko. Amaitzeko, irizpide bakoitzari lehen definitutako pisua aplikatuko zaio, alternatiba bakoitzak duen amaierako nota irizpide guztien betetze-maila batuz lortuko dela.

6.2.1 PicoRV32 [16]

Aurkezten den lehen aukera PicoRV32 prozesagailu edo CPU corea da. 32 Bitetako (RV32) prozesagailu hau bi helburu oso ezberdinetarako dago pentsatua; alde batetik, FPGAtan erabili daiteke gailu hauetan hedatutako diseinuetan prozesagailu laguntzaile gisa, atazak modu sekuentzialean exekutatzeko adibidez. Bestalde, diseinua ASICera eraman daiteke ere bai, nahiz eta diseinua bera ez egon helburu horretarako guztiz optimizatua. Corea ez dago errendimendu altuko prozesagailu bat sortzeko pentsatua; tamaina txikia eta eraginkortasuna bilatzen du diseinuak.

Hasieran, corearen garapena *Claire Wolf*-ek egiten zuen nagusiki, beste garatzaile batzuen laguntzaz. Honela, erakunde batetik kanpo garatutako core baten aurrean gaudela esan daiteke. Gaur egun, corea aipatutako garatzaileak sortutako YosysHQ taldeak mantentzen du, baina proiektuak ez du eguneraketarik jaso azkeneko 2 urteetan.

Egokitasuna

Core honek hainbat defektuzko aukera ematen ditu bere diseinua modu azkar batean helburu ezberdinetara moldatzeko. Esate baterako, coreak RISC-V-ren I, M, C eta E luzapenak onartzen ditu.

Luzapenak aukeratzeaz gain, prozesagailuak beste berehalako konfigurazio aukera interesgarri batzuk ematen ditu. Esate baterako, prozesagailuaren memoria interfazea aldatzeko aukera. Coreak interfaze natibo oso sinple bat, WishBone interfaze bat eta AXI Lite Master interfaze baten artean hautatzeko aukera ematen du, corearekin egin nahi den diseinuaren komunikazio bus ezberdinekin bateragarritasuna ematen. Kasu honetan, AXI Literekin lan egiteko aukera interesgarritzat hartu da, SoC-aren diseinuan erabili nahi diren periferikoek interfaze hau darabilte eta.

Bestalde, corearen tamaina eta honen lan-maiztasun maximoa (fmax) hobetzen duten bestelako aukera asko eskaintzen ditu, hala nola M luzapeneko DIV motako zatiketa instrukzioak azkartzeko hardwarea, etenak gaitzeko aukera edota ALUaren pertsonalizazio aukerak. Honela, dagokion aukerak gaituz eta desgaituz, txikiagoa den eta fmax handiagoa duen diseinu eraginkorrago bat lortu daiteke, kritikoa izan daitekeena FPGA diseinuetan eta baliagarria izan daitekeena ASIC txiki baten diseinuan.

Nahiz eta coreak aipatutako RISC-V luzapenak inplementatu, coreak ez ditu etenak eta salbuespenak RISC-V estandarrak dion bezala kudeatzen, ez baitu modu pribilegiaturik inplementatzen. Hori dela eta, prozesagailuak ez ditu CSRrik (kontrol eta egoera erregistroak), etenak eta salbuespenak modu pertsonalizatu eta sinple batean kudeatzen dituela.

Etenak kudeatzeko modua sinplea da teorian; prozesagailuak 32 biteko “IRQ” edo interrupzio eskaera (Interrupt ReQuest) portu bat du, eta portu honetan jasotako datuaren arabera identifikatzen dira interrupzio ezberdinak, 32 IRQ ezberdin egon daitezkeela asko jota. Behin interrupzioa jasota, hau software bidez aztertzen da, dagokion IRQ kodea jasotzen dela eta horri dagokion interrupzio zerbitzu errutina (ISR) aplikatzen zaiola. IRQ kodea jaso eta tratatu ahal izareko, RISC-V estandarretik kanpo dauden instrukzio batzuk sortu behar direla.

Honek etenak kudeatzeko modu azkarra dakar, helburu honetarako hardware gehigarria oso txikia dela. Ezaugarri hau PicoRV32-ren filosofiarekin bat dator, core ahalik eta txikiena sortzea baita helburua. Hala ere, eten hauek konpiladorearentzat definitu behar dira, diseinu konplexua konplexuagoa egiten dela.

Fidagarritasuna

PicoRV32 proiektuak eskuragarri duen informazioak ez du corearen egiaztapenari buruzko ezer aipatzen. FPGA sistemetan erabiltzeko pentsatuta dagoenez, gailu hauetan diseinuak garatzeko tresnen simulatzaileen erabilera sustatzen da, horretarako Verilog HDL lengoaian idatzitako *testbench* fitxategi batzuk eskaintzen direla. Fitxategi hauetan, hainbat RISC-V instrukzio dituzten fitxategien edukia eramaten da prozesadorera, instrukzio ezberdinak frogatzeko. Baina argi dago proiektuak ez duela egiaztapenerako azpiegitura oso sendo bat izan.

Softwareari dagokionez, prozesagailuak ez dator *toolchain* berezi batekin, RISC-V fundazioaren GCC *toolchain* estandarra erabili behar da prozesagailura zuzendutako kodea konpilatzeko. Beraz, softwareari dagokionez, ez dago fidagarritasun arazorik, *toolchain* estandarraren funtzionamendua sakonki egiaztatua dagoelako. Hori bai, IRQak inplementatzekotan, sortu beharreko instrukzioen egitura adierazi behar zaizkio konpiladoreari, hauen software garapenean akatsak sortu daitezkeela.

Inplementaziorako gida-lerroak

Bestalde, coreak ez dauka dokumentazio oso zabal bat, GitHub biltegian aurkitu daitekeen ReadMe fitxategia soilik dago eskuragarri. Gainera, corearen deskribapena HDL lengoaian ez dago egituratua, diseinuaren deskribapena oso luzea den fitxategi bakar batean dagoela eskuragarri. Honek corearen egitura zein den ulertzea asko zailtzen du, ondorioz bere pertsonalizazioa ere zailtzen. Azkeneko honen arrazoia corearen diseinuan erabilitako lengoaian datza, ikusiko diren beste alternatiba guztiak ez bezala, PicoRV32 Verilog lengoaian idatzita dago, gainontzekoak aberatsagoak eta objektuetara zuzenduagoa dagoen SystemVerilog HDL lengoaian idatzita daudela.

Bestalde, bere egitura sinplea dela eta, FPGAn inplementazioa oso erraza da. Nahiz eta ez adierazi FPGA zehatz baterako erabilera, diseinua Xilinxen hainbat FPGAtan frogatua izan da, Artix edota Zynq FPGAtarako guztiz egokia dela.

Proiektuak corea inplementatzen duen adibide sinple bat eskaintzen du, SPI (*Serial Periphal Interface*) kontrolagailu, RAM memoria eta UART banaz osatzen dela. Adibidea oso sinplea da, PicoRV32 prozesagailua bere memoria interfaze natiboarekin eskaintzen dela.

Prozesagailu honekin lan egiteko era frogatu ahal izateko, ez da adibide sinple hori erabili, PicoRV32-ren AXI Lite Master interfazea frogatu nahi baitzen. Beraz, frogapena azkarrago egin ahal izateko, corea Xilinxen FPGA batera eraman da, SoC diseinu sinple bat sortuz. Hala ere, diseinu honen abiapuntua corea soilik izanda, SoC-aren gainontzeko elementuak Xilinxen IPekin lortu dira, ASIC inplementazio baterako gehitu nahi diren osagai bakoitzaren inplementazioa egin edo kode-irekiko alternatiba bat bilatu beharko litzatekeela. Diseinu hau 1. eranskinean aurkitu daiteke, core honen azalpen gehigarriekin batera.

Ondorioak

PicoRV32 alternatiba nahiko malgua da. Bere tamaina bereziki txikiak eta coreak eskaintzen dituen ezaugarriek oso egokia egiten dute mikrokontrolagailu helburuetarako, hainbat defektuzko aukera ezberdinek bere pertsonalizazioa ahalbidetu eta errazten dutela.

Bere inplementaziorako, PicoRV32 darabilen erreferentziazko diseinu oso sinple eta bakarra aurkitu daiteke proiektuaren GitHub biltegian. Hau lagungarria izan daiteke frogetarako, coreak AXI interfazeak erabiltzeko aukera ematen duenez diseinuen sorrera oso erraza da.

Bestalde, corean softwarea hedatzea prozesu oso motela dela ondorioztatu da. Nahiz eta prozesua ez izan bereziki konplikatua, coreak ez duenez liburutegi lagungarririk eskaintzen, memoria erabilerarako edota interrupzioen kudeaketarako liburutegien osatzea lan astuna da. Gainera, RISC-V-ren arkitektura pribilegiatuaren inplementazio ezak, interrupzioak ez-ohiko modu batean kudeatzea eskatzen du, hauen inplementazioa konplikatuz.

Aipatu den bezala, corearen dokumentazioa eskasa da, eta gainera, diseinuaren HDL deskribapena ez da batere argia, corearen osagai guztiak fitxategi bakar batean aurkitu daitezkeela. Atal ezberdinen bereizketa oso garrantzitsua da diseinuaren ulermenerako, SystemVerilog bezalako HDL lengoaia aberatsago baten erabilera faltan botatzen dela. Ildo beretik, HDL deskribapena FPGAra zuzenduagoa da, ez baitu ASICetara eramateko erraztasunik gehitzen diseinuaren aldetik.

Azkenik, fidagarritasunari dagokionez, nahiz eta GitHub-eko kode-irekiko RISC-V inplementaziorik arrakastatsuenetako bat izan, badirudi corearen diseinuak ez duela egiaztapen sakonik jaso. Aipatutako *testbench*-ak corearen funtzionamendu orokorra zuzena dela bermatu dezakete, baina prozesu honekin frogatu beharreko hainbat kasu interesgarri kanpoan geratu daitezke probabilitate handiz.

6.2.2 CV32E40P [17]

Bigarren alternatiba PULP proiektuan sortutako RI5CY prozesagailuaren ondorengoa da, CV32E30P corea hain zuzen. Aurreko alternatiba ez bezala, prozesagailu hau ASIC diseinuetarako FPGA diseinuetarako baino zuzenduagoa dago, nahiz eta bigarren gailu hauetan inplementazioa posiblea den.

Nahiz eta bere jatorria PULP proiektuan aurkitu, gaur egun irabazi-asmorik gabeko OpenHW Group erakundeak kudeatzen du. Erakunde honek bere bazkide eta banakako laguntzaileek diseinatutako kode-irekiko hardware eta software diseinuak gauzatzen ditu, erakundearen azpiegituraren laguntzaz.

Egokitasuna

Corearen ezaugarriei dagokionez, honek RV32IMC luzapenen inplementazioa eskaintzen du, komadun zenbakiekin lan egiteko instrukzioak exekutatzea eskura jartzen duela ere F edo *Zfinx* aukerazko luzapenak inplementatuz, azkeneko hau RISC-V erakundeak ofizialtzat hartzen duen komadun zenbakien instrukzioetarako luzapen bat dela.

Zfinx luzapenaren bitartez, RISC-V prozesagailuek komadun zenbakiekin zenbait operazio gauzatu ditzakete, F luzapenak inplementatzen dituen HW erregistro gehigarriak erabili gabe. Hala ere, softwareari dagokionez, F luzapena edo *Zfinx* luzapena jasotzeko konpilatutako programak ez dira bateragarriak.

Bestalde, prozesagailuak X deituriko luzapen pertsonalizatua gehitzeko aukera ematen du. Luzapen honek PULP plataformak sortutako instrukzio bereziak onartzea du helburu, instrukzio hauek prozesagailuaren hardware atal espezifiko batzuk erabiltzen dutela. Esate baterako, hardware-bidezko begizten erabilera ahalbidetzen dute luzapen hauek.

Azkenik, coreak *Zicntr*, *Zicsr* eta *Zifencei* RISC-V luzapenak inplementatzen ditu ere. Luzapen hauek errendimendu-kontagailuak, CSR instrukzioak eta *FENCE* instrukzioak inplementatzen dituzte, RISC-V-ren arkitektura pribilegiaturako estandarra betetzeko beharrezko luzapenak alegia. Beraz core honek RISC-V-ren modu pribilegiatua inplementatzen du ere.

CV32E40P-ren luzapenetan kusi daitekeenez, core honekin bete nahi den amaierako helburua aurreko alternatibarenarekiko ezberdina da, hau aukera ahaltsuagoa dela. Nahiz eta eskaintzen dituen luzapen ia guztiak berdinak izan, prozesagailu honek F edo *Zfinx* instrukzioak exekutatu ditzake. Hau oso onuragarria izan daiteke operazio berezi hauen erabileraren bitartez errendimendu onurak lortu ditzaketen software programa edo aplikazioentzat, prozesagailuaren tamaina handitzea izango litzatekeela desabantaila nagusia.

Prozesagailuaren egitura sakonago aztertuz, CV32E40P-k ohikoa den *pipeline* batez osatuta dagoela ikusi daiteke. Jakina denez, instrukzio-mailan paralelismoa ahalbidetzen duen egitura honek prozesagailuaren aprobetxamendua handiagotzen du, helburua honen atal ezberdinak uneoro lanean egotea dela. Kasu honetan, errendimendu altua lortzeko 4 etapako *pipeline* bat inplementatzen da.

Fidagarritasuna

Softwarearen aldetik, kasu honetan bi *toolchain* ezberdin behar dira; PULP luzapenak erabiltzekotan *pulp-riscv-gcc tooclhain*a eta bestela, RISC-V-ren *toolchain* estandarra. Aipagarria da OpenHW Group erakundea bere GCC eta LLVM konpiladore propioak garatzen ari dela, bertsio egonkorra ez dagoela eskuragarri momentuz. Gainera, erakundea bestelako software aplikazio interesgarri batzuk garatzen ari da bere koreentzat, hala nola FreeRTOS kernel edota SDK oso bat.

OpenHW Group taldeak esfortzu handiak egin ditu ere bere coreen egiaztapen funtzionala betetzeko. Haien tresna nagusia UVM metodologia jarraitzen duen CORE-V-VERIF deritzo, eta haiek sortutako hainbat coreren egiaztapena betetzeko dago diseinatua, haien artean CV32E40P dagoela. Tresna honen bitartez, RISC-V instrukzioak ausaz sortzen dira, hauek ondoren, alde batetik simulatzaile komertzial batekin simulatutako CV32E40P bezalako core batera eta bestetik Imperas enpresak sortutako eta egiaztatutako kode-itxiko RISC-V ISA simulatzaile batera eramateko, ondoren emaitza biak alderatuz.

Tresna honen erabileraz, CV32E40P-ren bertsioa egonkor bat kaleratzea lortu zuen taldeak, 1.0.0 bertsioa hain zuzen ere. Gaur egun eskuragarri dagoen bertsiorik berriena, 2.0.0 alegia, ez dago guztiz egiaztatua momentuz. Dena dela, honek guztiak proiektua “bizirik” dagoela erakusten du, etorkizunerako bertsio berri eta hobetuen kaleratzeak aurreikusi daitezkeela.

Inplementazio gida-lerroak

Core honen dokumentazioa oso luzea eta sakona da xehetasunei dagokionez. Bertan corearen egitura eta inplementazio aukera ezberdinen inguruko informazio asko ematen da, prozesagailuaren funtzionamendua guztiz ulertu daitekeela. Hala ere, dokumentazioaren atal asko *work in progress* bezala agertzen dira, askotan atal horri dagokion garapena amaitu gabe dagoelako. Bestalde, software *toolchain*aren inguruko informazioa eskasa da, seguruenik hau amaitu gabe dagoelako.

Faltan botatzen den beste alderdi bat corea frogatzeko ereduzko diseinu bat da. OpenHW Group taldeak CORE-V-MCU diseinua eskaintzen du adibide bezala, baina honekin lan egitea ez da posible izan, jarraian aurkezten diren arrazoiengatik.

- Diseinuak erabiltzen dituen osagaiak oso konplexuak direnez, corearekin lan egitea zaila da, egitura ez dela argia ere ez. Ondorioz, froga sinpleak egiteko asmoarekin osagaiak diseinutik ezabatzea ez da berehalakoa.
- Nahiz eta diseinuak FPGAn hedatzeko aukera eman, sistemaren konplexutasuna dela eta, proiektu hau egiterako garaian eskuragarri ez zegoen FPGA modelo oso espezifiko bat behar da.
- Proiektua ez dago guztiz amaitua, oraindik eguneraketak jasotzen dituela.

Ondorioak

Argi dago CV3E40P oso aukera interesgarria dela. Corea egokia da proiektuaren helburuekiko, eskaintzen dituen RISC-V luzapenak bilatzen zirenak baitira. Gainera, software programa askotan aurki daitezkeen atazak eraginkortzeko PULP taldearen luzapen gehigarriak inplementatzen ditu, eraginkortasuna hobetuz. Luzapen pertsonalizatuak erabili ahal izateko eta diseinuaren bestelako alderdiak hobetzeko, corearentzat egindako *toolchain* pertsonalizatu bat eskaintzen du OpenHW Group taldeak, hardwarearen erabilera optimizatzeko helburuarekin.

Gainera, nahiko sakonki egiaztatu da corearen funtzionamendua zuzena dela OpenHW Group erakundearen esfortzuei esker. Lan honen emaitza teorian egonkorra den CV32E40P-ren 1.0.0 bertsioa da, aukera eta ezaugarri gehiago dituen 2.0.0 bertsioa etorkizun hurbil batean kaleratuko dela.

Hala ere, egia da CV32E40P proiektuak amaitu gabeko alderdi batzuk dituela. Alde batetik, aipatutako *toolchain*a amaitu gabe dago oraindik, lortu daitekeen azkeneko bertsioa ez dela amaierakoa. *Toolchain* honekin ez dago programa sinpleak konpilatzeko arazorik, baina konplexutasuna handitzen doan heinean, akatsak aurkitzea probableagoa da.

Bestetik, corearen dokumentazioan hainbat atal azaldu gabe daude, *work in progress* bezala markatuta daudenak zehazki. Egia da jada eskuragarri dagoen dokumentazioa nahiko osatuta dagoela, hala ere, egiaztapena bezalako ataletan honelako arazoak aurkitu daitezke.

Azkenik, CV32E40P darabilen erreferentziazko diseinu sinple bat faltan botatzen da. CORE-V-MCU diseinua eskuragarri dago [18], baina diseinu hau oso konplexua da IP konplexuen erabilera dela eta. Gainera, diseinua amaitu gabe dago oraindik.

Beraz, proiektuarentzat oso interesgarria izan daitekeen aukera baten aurrean gaudela esan daiteke, bere egokitasuna eta funtzionaltasunen egiaztapena dela eta. Hala ere, gogoan izan behar da guztiz amaitu gabeko proiektu bat dela, honek kontuan hartzeko oztopo batzuk sor ditzakeela.

6.2.3 Ibex [19]

Core honek ere PULP proiektuan du jatorria, bertan sortutako zero-riscy prozesagailuaren ondorengoa dela. Hau ere ASIC diseinuetara dago zuzendua, bere diseinua helburu hau betetzeko pentsatu zela. Hala ere, FPGAn ere inplementatu daiteke.

Gaur egun, LowRISC erakundeak kudeatzen du Ibex proiektua, irabazi-asmorik gabeko erakunde bat. Erakundeak kode-irekiko hainbat software baina batez ere hardware diseinuak kaleratzen ditu, hauen egiaztapenean enfasi handia jartzen dutela. Aurkeztutako beste erakunde asko bezala, LowRISC-ek enpresa eta unibertsitate laguntzaileei esker lortzen du finantziazioa, erakundea osatzen duten ingeniari eta azpiegitura-kostuak mantentzeko.

Egokitasuna

LowRISC-en diseinu arrakastatsuenetako bat da Ibex corea. 32 bitetako RISC-V core honek produkzio-mailako kalitatea eskaintzen du bere egiaztapen sakonari esker, prozesagailuaren aplikazio-eremu egokiena sistema txertatuen arloa dela, kontrol aplikazioetarako. Coreak *pipeline* egitura jarraitzen du.

Coreak RISC-V-ren hainbat espezifikazio jarraitzen ditu; Erabiltzaile-maila ISA, arkitektura pribilegiatuaren ISA, kanpoko *debug support*a, bit manipulaziorako luzapenaren ISA eta modu pribilegiatuaren babeserako PMP (Physical Memory Protection) hobekuntzak inplementatzen dituen ISA.

Honela, Ibexek I edo *embedded* aplikazioetarako egokia den E oinarri luzapenak inplementa ditzake, hauen gainean aukerazko C, M edota B (bit-mailako manipulaziorako instrukzioak) gehitu daitezkeela. Bestalde, CSR erregistroen *Zicsr* luzapena, *Fence* instrukzioen *Zifencei* luzapena eta aipatutako PMP hobekuntzen *Smepmp* luzapena [20] inplementatzen ditu Ibexek.

Berriro ere, Ibexen helburua luzapenak soilik aztertuz ikusi daiteke; ohikoenak eta oinarrizkoak diren IMC luzapenak inplementatzeaz gain, corearen tamaina modu esanguratsu batean txikiagotu dezakeen E luzapena edota sistema txertatuetan eraginkortasuna hobetzeko erabiltzen diren bit manipulaziorako instrukzioetarako B luzapena aurkitu daitezke. Bestalde, F bezalako espezifikoak diren eta hardwarea handiagotzen duten luzapenak ez dira gehitzen.

Inplementatu nahi diren luzapenen artean aukeratzeaz gain, Ibexek hainbat konfigurazio aukera ematen ditu, aukera interesgarrienetako batek corearen *pipeline*aren etapa kopurua erabakitzea ahalbidetzen duela. Izatez, coreak bi etapa ditu soilik, baina hirugarren *writeback* aukerazko etapa bat gehitu daiteke.

Beraz, coreak tamaina oso txiki bat lortzeko aukera ematen du, bi etapako pipeline sinple bat inplementatzeko aukera emanez, baina sistema txertatuetarako egokiak diren luzapenak erabiltzeko aukera emanez.

Fidagarritasuna

Corearekin bateragarria den softwareari dagokionez, LowRISC taldeak Ibexentzat egokitutako GCC eta LLVM konpiladoreak dituen *toolchain* berezi bat eskaintzen du. Dokumentu hau idazterako garaian, konpiladore bientzat bertsio egonkorrak eskaintzen dituzte, nahiz eta hobekuntzak eta *bug* jakin batzuk konpontzen dituzten beste bertsio batzuk oraindik ez dauden amaituta. Honela, Ibexera guztiz egokitzen den software *toolchain* baten eskaintza du proiektuak, C edo Rust lengoaian programatzea ahalbidetzen duenak. RISC-V-ren GCC *toolchain* ofiziala erabili daiteke ere, baina LowRISC-ek pertsonalizatutako *toolchain*a gomendatzen du hau corera gehiago egokitzen baita.

Egiaztapena Ibex corearen puntu fuerteetako bat da. LowRISC enpresak, *regression* motako testak gauzatzen ditu Ibexen gainean, kodearen ehuneko 94a baino gehiago frogatu dutela metodo honekin. Kasu honetan ere simulazio teknika berezi bat erabiltzen da; RISC-V instrukzioen ausazko sorgailu bat erabiltzen da, instrukzio hauek simulatzaile batekin simulatutako Ibex coreari pasatzen zaizkiola, honen emaitzak ausazko instrukzio berak jaso dituen kode-irekiko Spike RISC-V ISA simuladorearen emaitzekin alderatzen direla.

Bestalde, Ibex proiektuak erabiltzaileei haien simulazioak gauzatzeko prozesua errazten du. Alde batetik, Ibex kode-irekiko Verilator simulatzailearekin frogatzeko aukera eskaintzen da, GTK Wave bezalako bestelako kode-irekiko tresnekin batera. Verilator LowRISC-etik kanpoko simulatzailea da; Verilog edota SystemVerilog bezalako HDL lengoaiak C++ lengoaian modelatzen ditu, ondoren, erabiltzaileak adierazitako balioak aplikatzen zaizkiola modeloari honen irteerak eta barne-seinaleak aztertu ahal izateko.

Hala ere, Verilator gertaeretan oinarritutako simulatzaile bat da, kasu honetan gertaera nagusia erloju seinalearen aldaketak direla. Hori dela eta, simulatzaile honek ez du simulazio oso sakon bat egiteko aukera ematen, erloju zikloen arteko egoerak ez baititu kontuan hartzen. Hori dela eta, Ibex SystemVerilog HDL lengoaiarentzat bateragarritasun ona duen edozein simulagailurekin simulatu daiteke, LowRISC-ek Cadence Xcellium, Mentor Questa edota Synopsys VCS bezalako simulagailuekin. Programa hauen alde txarra haien lizentzien prezio altua da.

Inplementaziorako gida-lerroak

Ibexen inguruko informazio asko eskaintzen du LowRISC-ek honen dokumentazio zehatzaren bidez. Bertan, corearen alderdi asko modu oso sakonean azaltzen dira, egitura eta coreak eskaintzen dituen aukerak argitzen dituela.

Proiektu hau gainera guztiz bizirik dago. Ibexen bertsio egonkorraz kanpo, LowRISC-eko garatzaileak etengabe ari dira lanean prozesagailua hobetzeko ezaugarri berriak gehituz eta bera perfekzionatzeko akatsak konponduz eta optimizazioak eginez. Gainera, komunitatearen iritzia kontuan hartzen dute; erabiltzaile batek arazo bat aurkitzen badu, Ibexen GitHub orrian *issue* bat ireki dezake eta Ibex taldeko langileek erantzuna emango diote, prozesagailua hobetzea baita helburua.

Corearen erabilera argitzeko, LowRISC-ek proiektu bi argitaratu ditu; *ibex-demo-system* eta OpenTitan, 1 eranskinean sakonago azaltzen direnak. *Ibex-demo-system* diseinu sinple bat da [21], Ibexekin frogak egiteko edo diseinu abiapuntutzat hartzeko egokia dena, bere izenak dioen bezala. Sistema honetan, Ibexen memoria-interfaze natiboekin bateragarria den komunikazio bus bat gehitzen da, bertara hainbat perfieriko konektatzen direla: SPI Master bat, UART bat eta GPIO periferiko bat. Datuen memoria busaren bidez atzitzen da ere, instrukzioen memoria memoria-interfazetik zuzenean atzitzen dela. *Ibex-demo-system* gainera, proiektua egiterako garaian eskuragarri zegoen Arty plakarako dago pentsatua.

Bestalde, LowRISC erakundeak OpenTitan proiektua eskaintzen du Ibexen inplementaziorako [22]. Sistema hau *Root of Trust* edo RoT bat da, segurtasun funtzio espezifiko eta kritiko bat edo gehiago betetzen dituen hardware-sistema oso fidagarria alegia, zeinaren prozesagailua Ibex den. OpenTitanek Ibex prozesagailuaren heldutasuna oso argi erakusten du. Hainbat enpresa pribatu oso garrantzitsuk parte hartu dute proiektu honetan, Ibexen oinarritu direla RoT fidagarri eta ireki bat sortzeko, corearen prestutasun-mailaz fidatu direla eta, Ibexen fidagarritasuna islatuz. Hala ere, sistema konplexuegia eta handiegia da proiektu honen helburuentzat.

Ondorioak

Ibex oso aukera interesgarria da proiektu honetan. Gainontzeko aukerek aurkezten dituzten RISC-V luzapenaz izateaz gain, sistema txertatuetarako egokia izan daitekeen B luzapena gehitzen du, eta baliabide gutxi behar dituzten aplikazioetara dago zuzendua, horretarako area txikia hartzen duela bere diseinuak.

Software *toolchain* egonkor eta pertsonalizatu bat eskaintzen du eta eguneraketa konstanteak jasotzen ditu, bai software bai hardwarearen aldetik, aldi berean bertsio egonkor bat eskaintzen duela. Dokumentazioa osoa da eta diseinuaren egiaztapen oso sakon eta aurreratu bat gauzatzen dute bere garatzaileek.

Diseinu pertsonalizatuak gauzatzeko demo sistema sinple bat eskaintzen da, eta sistema hau simulatu ahal izateko kode-irekiko zein komertzialak diren tresnak eskaintzen dira. Honela, argitaratutako dokumentazio osoarekin batera, Ibexen funtzionamendua sakon ulertzeko baliabide asko eskaintzen dira. Bestalde, Ibexen oinarritutako sistema baten diseinurako abiapuntu oso ona da, oinarrizko osagaiak gehitzen baititu soilik.

Beraz, argi dago Ibex alternatiba sendo bat dela, proiektuak egokitasun ona, fidagarritasun altua eta inplementaziorako lagungarria den diseinua eskatzen dituelako.

6.2.4 PULPino – RI5CY edo Zero-Riscy [23]

Aztertu den azken alternatiba hau ere PULP plataformak sortutakoa da. Hala ere, PULPino ez da core baten diseinua, baizik eta bi coretan oinarritutako sistema baten diseinua. Core hauek RI5CY eta zero-riscy dira, lehen aipatu den bezala, gaur egungo CV32E40P eta Ibexen aurrekariak, hain zuzen ere.

Egokitasuna

Alternatiba hau sistemak eskaintzen duen osotasun eta egokitasuna dela eta aukeratu da, mikrokontrolagailu aplikazioetara zuzendutako SoC baten diseinua eskaintzen baitu. Gainera, diseinua ASICera eramateko dago zuzendua, 65 nm-ko teknologian inplementatu zuela PULP taldeak. Hala ere, diseinua FPGAra eraman daiteke egiaztapen helburuetarako.

Sistema core ezberdin birekin inplementatu daiteke, beharren arabera egokitasuna lortzeko. Nahiz eta core hauek lehen aztertu diren, bertsio zaharragoekin lan egiten du PULPinok, core hauen ezaugarriak ezberdinak direla.

RI5CY-ri dagokionez, CV32E40P-ren antzeko kasu bat da, IMC eta F luzapenak inplementatzen dituela PULPen X luzapen pertsonalizatuez gain. RISC-V arkitektura pribilegiatua inplementatzen du ere bai. Egitura aldetik oso antzekoa da ere, 4 etapako *pipeline* batekin diseinatuta dagoela. Hala ere, RI5CY-ren memoria-interfazeek protokolo natibo bat erabiltzen dute, OBIren antzekoa dena baina ez dena honekin bateragarria. Bestalde, coreak darabilen komadun zenbakientzako FPU unitatea CV32E40P-rena baino zaharragoa da eta nahiz eta corearen bertsio egonkor bat erabili, RI5CY-k konpondu gabeko arazo batzuk ditu, bere garapena izoztuta geratu baitzen.

Zero-riscy-k aldiz Ibexek baino aukera murritzago bat eskaintzen du, IMC eta E luzapenak eskaintzen baititu soilik, B luzapena ez duela inplementatzen. Bestalde, ez du bere pipelinean Ibexen aukerazko hirugarren etapa gehitzen, bi etapako *pipeline*a erabiltzeko aukera ematen duela soilik, Ibexen oso antzeko mikroarkitektura jarraitzen dela dena den. Berriro ere, core honek RISC-V-ren arkitektura pribilegiatuaren 1.9 bertsio zaharra inplementatzen du.

Aipatu den moduan, PULPino sistemaren arkitektura oso egokia da mikrokontrolagailu gisa erabiltzeko, prozesagailu, memoria, periferiko eta sistema-barneko osagaien arteko komunikazioak gauzatzeko elementuak eskaintzen dituelako, SoC bat sortzeko oinarri oso sendo bat emanez. Osagai hauen ezaugarririk garrantzitsuena kode-irekiko IPak direla da, diseinuko atal ia guztiak SystemVerilog HDL lengoaian deskribatuta daudela.

PULPino-rekin frogak egin ahal izateko, FPGA gailuak erabiltzea proposatzen du PULP plataformak. Horretarako, proiektua egiterako garaian eskuragarri zegoen Avneten Zedboard plaka erabiltzea proposatzen da, plakak duen FPGAdun Zynq SoC-az baliatuz. Hedatze prozesua errazteko asmoz, PULPino “emulatzeko” plataforma konplexu bat eskaintzen da, non Zynq SoC-aren *Processing System* (PS) ataleko ARM prozesagailua Zynqeko FPGAn PULPinoren diseinua hedatzeko erabiltzen den. Prozesagailuaren bitartez gainera, PULPinoren SPI Slave portura egiten da atzipena, bertatik instrukzio eta datuen memoria erabili nahi diren programen edukiekin betetzeko.

Fidagarritasuna

Egiaztapenari dagokionez, PULPino-k ez dauka arlo honen inguruko informaziorik bere dokumentazioan. PULPino simulatzaile komertzial ezberdinekin simulatu daiteke, Siemensen ModelSim programarekin modu sakon batean simulatu daitekeela diseinu osoa. Aukera honen eskaintzak diseinuak egiaztapen prozesu minimo bat jaso duela adierazten du, baina ez dago hau frogatzen duen informazio ofizialik.

Softwareari dagokionez, PULP plataformak *toolchain* pertsonalizatu bat du, *RI5CY GNU toolchain* izenekoa. Tresna hau RI5CY eta zero-riscy coreekin bateragarria da, hauentzako egokitutako kodea konpilatu dezakeela. Honetaz gain, PULPino SoC bat denez eta periferiko propio asko dituenez, periferiko hauek eta sistemaren alderdi ezberdinak software bidez erabili edo konfiguratzeko C lengoaian idatzitako hainbat liburutegi eskaintzen dira, sistemaren kontrolerako oso lagungarriak direla.

Hala ere, bai hardwarearen diseinua, bai software *toolchaina* guztiz zaharkituta daude. PULPinoren core biak zaharrak dira, haien garapena izoztuta geratu zela duela urte batzuk. Honela, core biek adibidez, ISA pribilegiatuaren 1.9 bertsio zaharra inplementatzen dute, gaur egungoa 1.12 bertsioa dela. Berdina gertatzen da *toolchainarekin,* hau ere izoztuta geratu denez, programen konpilazioa ISA zaharrentzat egiten du.

Inplementazio gida-lerroak

Tamalez, PULPino-ren dokumentazioa ez da batere sakona. GitHub-en aurkitu daitekeen *readme*an, oinarrizko informazioa biltzen da, baina ez da sistema sakonki aztertzen. Bestalde, PULPino-k *datasheet* txiki bat du, diseinuaren osagai bakoitzaren xehetasun batzuk (Periferikoen erregistroak, hauen helbideak, etab.) aurkezten direla. Hala ere, *datasheet* hau ez da batere sakona, azalpen asko oso azalekoak direla.

Ondorioak

PULPino oso alternatiba interesgarri bezala planteatu da, sortu nahi den SoC-aren diseinurako abiapuntu ezin hobe bat ematen baitu. SoC bezala, ez da oso konplexua edo aurreratua, baina bere ezaugarriak proiektuaren helburuekin guztiz lerrokatzen dira, mikrokontrolagailu sistema oso egokia baita. Periferiko berriak gehitzea oso erraza da diseinu honetan ere, erabiltzen den busa oso ezaguna den AXI motakoa baita.

Gainera, diseinuaren gainean egin diren aldaketei esker, ASICetara guztiz zuzendutako diseinu funtzional baten bilakatu da, emulatzailearen oztopoa gainditzen baita. Ez hori bakarrik, PULPino ASICetara eramateko pentsatua dagoenez, bere HDL deskribapena helburu honetarako paregabea da.

Sistemak erabiltzen dituen IP guztiak kode-irekikoak dira eta ia denak PULP plataformak garatutakoak dira, ondorio bezala guztiz bateragarriak diren blokeez osatutako sistema bat sortuz. Berriro ere, honek diseinua ASICera eramateko prozesua errazten du, ez baitago “kutxa beltzik”.

Bestalde, PULPino-ren alde negatibo pisutsuena zaharkituta geratu den sistema bat dela da. Nahiz eta informazio hau ezagutu, PULPino teorian diseinu egonkor bat delako eta sistema oso bat delako hartu zen alternatiba bezala. Hala ere, praktikan lehenengo arrazoi hau zalantzan jarri da.

Alde batetik, PULPino proiektuan froga gisa eskaintzen den FreeRTOS softwarea exekutatu da sisteman, software eta hardware diseinu egonkorretan aldaketarik egin gabe. Froga sinpleenekin sistemak ondo erantzuten zuen, baina sistema eragilearen oinarrizko osagai batzuk erabili ostean, sistemak arazo larriak aurkezten zituen.

Bestetik, PULP taldeko garatzaile nagusienetako batek, gaur egun OpenHW Group erakundean lan egiten duenak, RI5CY eta zero-riscy coreei egiaztapen sakonagoak egin eta gero hainbat bug aurkitu zituztela aitortu zuen 2023ko RISC-V-ren Europako *summit*ean. *Bug* hauek arriskutsuak izan daitezke; nahiz eta hauek ez ezagutu, diseinua ASIC bihurtu ostean ez usteko larriak sor bai ditzakete eta.

Arazo hauek espero zitezkeen, PULPino proiektuak duela bost urte jaso baitzituen azkeneko eguneraketak, guztiz hilda dagoen proiektu bat dela berresten. Hori dela eta, egonkortzat eman zuten azkeneko bertsioak dituen arazoak konpondu gabe geratu dira.

Hau guztia esanda, PULPino aurkezten den alternatiba guztietatik, erabiltzeko prest dagoen SoC baten diseinu egoki bat eskaintzen duen bakarra da, inplementatu nahi diren funtzio gehienak biltzen dituen ASICetara zuzendutako sistema bat dela. Hala ere, azken urteetan ez dituenez aldaketarik jaso, zaharkituta geratu da sistema, bere akatsak zuzendu gabe daudela. Akats hauek, coreetan daude gehienbat, bestelako blokeak ez dutela zer akasdunak izan.

6.3 Aukeren baliabideen erabilera

Aipatu den bezala, core guztiak FPGA gailuetara eraman dira, kasu gehienetan corea integratzen duen SoC diseinu sinple bat FPGAra eraman dela ere bai. Honek diseinu bakoitzaren baliabide-erabileraren estimazio bat ematea ahalbidetu du.

Erabilitako baliabideak ez dira berdinak izango FPGA edo ASIC inplementazio batean; FPGA gailu batek baliabide jakin batzuk ditu,sintesia eta *place and route* prozesua gauzatzen duen programa arduratzen dela diseinuaren inplementazioa egin ahal izateko eskura dauden baliabide hauen kudeaketaz. Hala ere, estimazio hau erreferentzia fidagarri bezala hartu da diseinu bakoitzaren baliabideen erabilera estimatzeko.

6.3.1 Coreen analisia

Jarraian aurkezten den taulan, coreen sintesia egin ondoren estimatutako baliabideak erakusten dira, sintesia Artix-7 35T FPGAra zuzenduta egin dela. PULPino ez da kontuan hartu, hau SoC baten diseinu osoa baita eta analisi honetan coreen baliabideen erabilera soilik analizatu nahi baita. CV32E40P eta Ibex IMC luzapenekin inplementatu dira.

	Slice LUTs	Slice Registers	F7 Multiplexoreak	DSPak	BUFGCTRL
PicoRV32	2221	1238	-	-	-
CV32E40P	4348	2273	268	5	12
Ibex	3412	1371	18	10	1

1. Taula: Aztertutako coreen baliabideen erabilera

FPGA osatzen duten oinarrizko bloke konfiguragarrietako (*Slice*) *Look Up Table* edo LUT *sliceak* lotzeko F7 multiplexoreei dagokionez, argi dago CV32E40P-k baliabide gehien erabiltzen dituen corea dela. Honek zentzua du, bere deskribapena egin denean ikusi baita core konplexuena dela.

Bestalde, PicoRV32 baliabide gutxien darabiltzan corea dela ikusi daiteke, LUT oso gutxi erabiltzen dituela beste diseinuekin alderatuz.Erregistro eta LUT kopuru oso txikia erabiltzen duela ikusi daiteke, DSPak bezalako bloke konplexuagoen erabilera ez duela egiten. Analisi honek core honen sinpletasuna konfirmatzen du.

Nahiz eta PicoRV32-ren emaitzak harrigarriak izan, Ibexenak oso deigarriak dira ere. Erregistro oso gutxi erabiltzen dituela ikusi daiteke, honen LUT kopurua ere ez dela bereziki handia. Hori bai, DSP blokeen erabilera kopurua deigarria da.

6.3.2 Froga sistemen analisia

Jarraian coreak integratzen dituzten eta froga helburuetarako erabili diren sistemen FPGA inplementazioek erabilitako baliabideak erakusten dira. PicoRV32-ren kasuan, sistema 2. eranskinean erakusten den sistema hartu da kontuan. *Ibex-demo-system* eta *PULPino* aldaketa gabe inplementatu dira, *PULPino RI5CY* corearekin inplementatu dela IMC luzapenak ahalbidetuz.

	Slice LUT	Slice Registers	F7 Mux	F8 Mux	BRAM Tile	DSP	BUFGCTRL	Bonded IOB
PicoRV32 sistema	2759	1690	-	-	8	-	2	13
Ibex-demo-system	6496	6002	346	128	16	10	3	42
PULPino	13789	9981	823	134	16	8	0	0

2. Taula: Aztertutako froga sistemen baliabideen erabilera

Ikusi daitekeenez, oso deigarria da PicoRV32-ren sistemak baliabide oso gutxi erabiltzen dituela sistemaren osagaiak gehitu ondoren. Osagai hauek Xilinxen IPak izanda, baliteke kontuan hartu ez izatea edo hauen optimizazioa maximoa izatea.

Bestalde, oso argi dago PULPino dela baliabide gehien darabilen diseinua, baina periferiko gehien dituen diseinua dela kontuan hartu behar da, besteekin alderatuz askoz ere elementu gehiago dituela.

Hau kontuan hartuta, *ibex-demo-system*-aren LUT kopuru bikoitza eta 4000 erregistro gehiago erabiltzen dituela, aipatu den beste diseinuak baino periferiko askoz ere gehiago inplementatuz. Beraz, PULPino-k tamaina nahiko txiki bat lortzen duela ondorioztatu daiteke. F7 eta F8 multiplexore gehiago erabiltzea normala da, haien artean interkonektatu beharreko *slice* gehiago erabiltzen direlako. DSP kopurua nahiko txikia dela ikusi daiteke ere, bloke konplexu gutxi dituela ondorioztatu daitekeela.

Ibex-demo-system-ari dagokionez, baliabide gutxi erabiltzen dituen diseinu bat dela ikusi daiteke. Hau espero zen zerbait da, sistema bera oso sinplea delako. Hala ere, erabilitako baliabide kopurua oso txikia dela azpimarratu behar da.

Erabilitako BRAM elementuen kopurua sistema bakoitzaren memoria kopuruarekin dago lotua zuzenean. PULPino eta *ibex-demo-system* diseinuek 32 kB-eko memoria dute, PicoRV32-k 16 kB-ekoa darabilela. Memoria hauek FPGA barneko elementuekin inplementatzen dira, Block RAM egituren bitartez.

6.4 Amaierako konparaketa

Aurkeztu diren aukera guztiak analizatu ondoren, jarraian alternatibek irizpide bakoitza zenbateraino betetzen duen erakusten da, aukeren artean bat aukeratzeko amaierako nota ere ikusi daitekeela. Noski, emandako notak atal honetan aurkeztutako arrazoietan daude oinarrituta.

	Proiektuarekiko egokitasuna (%50)	Fidagarritasuna (% 30)	Inplementaziorako gida-lerroak (%20)	Guztira
PicoRV32	6	5	5	5.5
CV32E40P	7	8	7	7.3
Ibex	**9**	**8**	**9**	**8.7**
PULPino	9	6	5	7.3

3. Taula: Aztertutako coreen amaierako konparaketa

Ikusi daitekeenez, Ibex corea aukerarik egokiena dela ondorioztatu da. Bere diseinu eta egitura oso ondo moldatzen da proiektuaren eskakizunetara, honen egiaztapen prozesuak eta heldutasun maila asko hartu dela kontuan ere.

CV32E40P aukera interesgarri bezala ikusi da, bestelako helburuetara zuzendutako ASIC baten diseinuaren kasuan kontuan hartu beharko litzatekeela. Hala ere, kasu honetan bere diseinua ez da guztiz egokia aurkitu, inplementazioa errazteko diseinu sinple baten faltak ere erabakian eragin duela.

PicoRV32-ri dagokionez, FPGA gailuen diseinuetarako oso aukera baliagarria bezala ikusi da, bere tamaina guztien artean txikiena baita, alde handiarekin gainera. Hala ere, ASIC baten diseinurako ez da nahikoa bezala baloratu, honen diseinua besteena baino eskasagoa delako eta bere dokumentazioa oso urria baita.

Azkenik, nahiz eta hasiera batean PULPino aukera nagusi bezala hartu, honek dituen akatsak eta proiektua abandonatua dagoela jakiteak bere aukeraketa baztertzea eragin du proiektu honetarako. Bestalde, RISC-V edota prozesagailu-arkitektura bezalako gaien inguruen ikasketarako oso aukera egokia bezala ikusten da.

7. Arriskuen analisia

Dokumentuaren atal honetan, proiektua garatzen den bitartean agertu daitezkeen arrisku ezberdinak aztertuko dira. Arriskuen identifikazioa eta azterketa guztiz garrantzitsua den ataza bat da, izan ere, honi esker proiektuan zehar gertatu daitezkeen ezustekoak eta eragozpenak ekidin daitezkeelako, proiektua planifikatutako epeetan eta aurreikusitako baliabideekin gauzatzeko probabilitatea handiagotuz.

Hortaz, argi dago komenigarria dela arreta handia jartzea analisi honetan. Hori dela eta, analisia modu sakonean gauzatuko da. Horretarako, lehenengo arriskuak identifikatu eta banan-banan deskribatu egingo dira. Behin arazo bakoitza ondo ezagututa, arazo bakoitza ekiditeko argibideak emango dira. Jarraian, deskribatutako arrisku bakoitzaren gertaera probabilitatea eta honen eragina kuantifikatuko da, arrisku potentzial bakoitzaren kaltea zenbatesteko.

Azkenik, analisi osoa biltzen eta laburtzen duen probabilitate-eragin matrizea aurkeztuko da, azaldutakoa irudikatzeko asmoarekin.

7.1 Arriskuak

A1. Industriak RISC-V Teknologian interesa galtzea

Proiektu honetan egindako alternatiben analisia eta honen ondorioz sortutako SoC diseinua proiektu handiago baten atal bat dela kontuan hartu behar da arrisku hau ulertu ahal izateko. Noski, industriak teknologia batengan duen interesa ez da hilabete batzuetan aldatzen, denbora asko behar da horrelako norabide aldaketa bat hartzeko. Hala ere, arrisku hau guztiz kontuan hartu behar da, proiektu hau luzeagoa izango den beste proiektu baten parte baita.

Arrisku honen potentzial-maila oso argi islatzen du Europear Agentzia Espazialaren (ESA) LEON proiektuak. Erakunde honek 1997. urtean abiarazi zuen SoC propio bat sortzeko proiektua, hainbat arrazoi direla eta SPARC ISAn oinarritu zirela sistemaren prozesagailuaren diseinua gauzatzeko. Arrazoien artean ISAk bere garaian izan zuen gorakada aurki zitekeen. Urteak aurrera egin ahala, industriak SPARC alde batera utzi zuen, teknologiaren komunitate-babesa galduz. Ondorioz, ESAk SPARCetik RISC-V-ra migratu behar izan du, inbertsio oso handiak berriro egin behar izan dituela [24].

Ikusi daitekeen bezala, oso zaila da gertaera hau ekiditea, baina epe labur edo ertainean gerta daitekeena aurreikusi daiteke RISC-V-ren egungo egoera eta etorkizunerako duen norabidea aztertuz. Bestalde, ezinezkoa da epe ertainetik haratago hobeak diren ISAk agertuko diren aurreikustea, uneko ISAen ikuspegi orokorra aztertzea izan daitekeela tresna baliagarriena.

RISC-V-k industriaren arreta irabaztea lortu duela argi dago. Hau gaur egun aurki daitezkeen RISC-V-ren erakundeen atzean dauden enpresa pribatuak aztertuz ikusi daiteke; erdieroaleen industrian liderrak diren enpresek diru eta baliabide asko jarri ditu RISC-V-ren garapena bultzatzeko.

Ez hori bakarrik, gaur egun oso garrantzitsuak diren hainbat eta hainbat enpresek RISC-V-n oinarritutako HW eta SW produktuak kaleratzen hasi direla edo prozesuan daudela ikusi daiteke. Esate baterako, Linux kernelak RISC-V-rako bertsio ofizial bat izatea oso esanguratsua da, gaur egun erabiltzen diren gailu informatikoen ehuneko handi batek erabiltzen baititu sistema eragile hauek, RISC-V-ren etorkizuna aurresaten laguntzen duela honek.

Gainera, nahiz eta teknologiaren garapena eta, batez ere, honen hedapena, ARM bezalako "aurkariekiko" oso urrun egon, RISC-V-k bestelako ISAk konpontzen ez dituzten arazoak konpontzen ditu, honen espezializazio-maila dela eta. Beraz, baliteke RISC-V-ren etorkizuna konputazio orokorrean lausoa izatea, baina azeleragailu edota baliabide gutxiko sistemetan etorkizun oso argia du.

Aipatutako aurrekariengatik eta teknologiaren egungo eta etorkizuneko egoera direla eta, A1 arazoaren gertaera probabilitatea "baxua (% 30)" bezala klasifikatu da, RISC-V-k industriaren arreta handia bereganatu duelako jada eta etorkizunera begira adopzioak jarraituko duelako dirudi eta. Eraginari dagokionez, "kritikoa (% 90)" bezala sailkatuko da, Departamentuaren epe luzerako proiektua bertan behera utzi dezake eta.

- **Gertaera probabilitatea:** % 10
- **Eragina:** % 90

A2. ASICetarako desegokiak diren praktiken erabilera diseinuan

Berriro ere, arazo hau kokatu ahal izateko, aurkezten den proiektuan garatutako diseinua ASIC batera eraman nahi dela gogoratu behar da. Hala ere, diseinuaren balidazio funtzionala egin ahal izateko, hau FPGA gailuetan hedatuko da diseinuaren deskribapenerako erabili den HDL lengoaien bidez.

FPGA gailuak eta ASICak ezberdinak direnez, nahiz eta lortu nahi den diseinua FPGAtan inplementatu, hau ez da gailu hauetarako optimizatu behar. Arrisku hau diseinuaren etapa guztietan argi izan beharko da, izan ere, diseinuaren osagaiak FPGAn errendimendua optimizatzea helburu bezala eginez gero, ASICera eramatean errendimendua gutxitu baitaiteke.

Diseinuan erabiltzen diren IP blokeak kontu handiz aukeratu beharko dira ere. FPGAtan eskaintzen diren IP asko hauek eskaintzen dituzten konpainiaren produktuen teknologiaren menpe daude, eta baliteke IP bloke horiek ASICera eramatea ezinezkoa izatea, diseinuaren aldaketa beharrezkoa izango litzatekeela.

Baliteke beste IP askok arazo hau ez aurkeztea, baina IPak kutxa beltz bat direnez, IP baten sekretutasun-mailak ASIC batera eramateko informazio nahikoa ez ematea gertatu liteke. Berriro ere, proiektu honetatik at geratzen diren hurrengo pausuetan arazo bat sortuko luke honek, egindako goi-mailako diseinuaren atal hau ordezkatzera behartuz.

Arazo hau ekiditeko, IP itxien erabilera saihesten saiatuko da. Alternatiba irekirik aurkitu ezean, eta blokearen sorrera konplikatua izan daitekeela sumatzean, IP itxien erabilera gauzatu daiteke, betiere diseinua ASICera eramateko informazio nahikoa eskura baldin badago.

Lege hauek diseinu egoki bat aurkezten duen kode irekiko oinarri-diseinu bati aplikatu beharko zaizkio, hauetan aurkitu baitaitezke aipatutako IP itxiak. Gida-lerro hauek gogoan eduki beharko dira, ohikoa baita IP itxi hauen erabilera. Hauek jarraituz gero, arrisku hau ez litzateke garapenean agertu behar, ekiditea posiblea dela eta, ondorioz, gertaera-probabilitatea baxua izan daitekeela.

Aipatutako guztia kontuan hartuta, A2 arriskuaren gertaera probabilitatea “baxua (% 30)” bezala sailkatu da, bere eragina “altua (%70)” bezala katalogatu dela, arriskua egi bihurtzekotan diseinuaren aldaketa eskatuko lukeela.

- **Gertaera probabilitatea:** % 30
- **Eragina:** % 70

A3. Diseinuan akatsa esanguratsuak

Argi dago ezinezkoa dela akatsik gabeko hardware diseinu baten osaketa; nahiz eta simulazio eta kodearen azterketa sakonak egin, beti egongo da *bug* bat topatzeko aukera. Baina kasu honetan, HDL lengoaiarekin egindako diseinuan akatsak aurkitzea oso garrantzitsua izango da, behin diseinua ASIC baterako egokitu ondoren ASICa bera fabrikatu ostean ez baita egongo atzera egiteko modurik eta.

Proiektu honen kasuan, diseinuaren egiaztapena bi ataletan banatu daiteke. Alde batetik, RISC-V ISAn oinarritutako instrukzioak exekutatzen dituen prozesadorea egongo litzateke. Honen funtzionamendua zuzena dela bermatzeko modu oso eraginkor bat, prozesadoreak inplementatzen dituen RISC-V luzapenen espezifikazioak barne hartzen dituen instrukzio guztiak exekutatzeko kapaza dela egiaztatzea da.

Hau berebizikoa izan daiteke; prozesagailua ez bada luzapen baten instrukzio guztiak exekutatzeko gai, prozesagailu horretara zuzendutako softwarea RISC-V software tresna estandarrarekin konpilatzean, baliteke prozesagailuak ezagutzen ez duen instrukzio bat gehitzea, prozesagailuak ezingo duela programa exekutatu.

Egiaztapenaren beste atala prozesagailutik at dagoen SoC-eko osagai ororen funtzionamendua aztertzea litzateke. Bertan inplementatutako periferiko edota komunikazio mekanismoak guztiz zuzenak direla bermatu beharko da, seinale mailaraino egin beharko dela azterketa. Egiaztapen hau ez bada egiten, ezingo da diseinuaren funtzionamendu zuzena guztiz bermatu.

Proiektu honetan egin nahi den diseinuan jada egindako kode-irekiko prozesagailu bat erabili nahi denez, aukeraketa egiterako garaian guztiz oinarrizkoa izango da diseinu horrek atzean duen egiaztapena zein puntura arte den fidagarria aztertzea. Diseinuak egiaztapen azpiegitura bat izan behar du atzetik, honek segurtasun handia eman dezake eta bere erabilera egiterako garaian.

Bestalde, SoC-aren gainontzeko osagaien egiaztapena egitea ere beharrezkoa izango da. Osagai hauek kanpoko kode-irekiko diseinuak bezala hartzen badira, prozesagailuaren kasuan bezala egiaztapen-maila ezagutu beharko da. Osagaiak diseinatu behar badira, hauen funtzionamendua egiaztatu beharko da, simulazio edota FPGAn hedapenen bidez.

Noski, arrisku hau diseinuaren garapenean zehar gertatu daiteke ere diseinua inplementatzen joan ahala, bai simulazio bai FPGAn akatsak aurkitu daitezke eta. Akats hauek proiektuaren garapena oztopatu dezakete, hauek konpontzen denbora gehigarria eman beharko delako.

Aurkeztutako arrazoiak direla eta, argi dago arrisku hau oso garrantzitsua dela, diseinuaren funtzionalitatea mugatu dezake eta. Aipatu den bezala, akatsak lortzea oso probablea da, baina egiaztatutako diseinu batekin akats esanguratsuen kopurua asko murriztu daiteke. Hori dela eta, A3 arazoaren gertaera probabilitatea “baxua (% 30)” bezala sailkatu da, bere eragina “kritiko (%90)” bezala katalogatu dela, arriskua egi bihurtzekotan diseinuaren aldaketa sakonak beharko liratekeela.

- **Gertaera probabilitatea:** % 30
- **Eragina:** % 90

A4. Epeak ez betetzea

Proiektu hau gauzatzerako garaian, dokumentu honetan aurrerago proposatzen den lan plana jarraituko da. Hala ere, hainbat arrazoiengatik baliteke epe hauek ez betetzea, edozein atazaren atzerapenak proiektu osoaren atzerapena ekarri dezake eta. Proiektua ez bada adierazitako epean amaitzen, litekeena proiektuaren aurrekontuak aldaketa negatiboak jasatea da eta hori guztiz ekidin behar den zerbait da.

Arrisku potentzial hau egia bihurtu ahal izateko bi arrazoi nagusi daude. Hauek lan planaren diseinua desegokia izatea edota ataza batzuetan behar dena baino denbora gehiago ematea izan daitezke. Hau gertatu ez dadin, arreta handia jarri beharko da lan planaren diseinuan eta garatzaileek hau jarraitzearen garrantzia kontuan izan beharko dute momentu orotan.

Honengatik, A4 arazoaren gertaera probabilitatea “posiblea (% 50)” bezala klasifikatuko da plangintzari emango zaion garrantzia dela eta. Eraginari dagokionez, “ertaina (% 50)” bezala sailkatuko da aurrekontuan sortu ditzakeen aldaketak direla eta, baina proiektuak ez dituenez baliabide garestiak erabiltzen, kostuan igoera ez litzateke oso larria izan beharko proiektua luzatzeagatik.

- **Gertaera probabilitatea:** % 50
- **Eragina:** % 50

7.2 Probabilitate-eragin matrizea

"Arriskuen analisia" atalaren hasieran aipatu den bezala, analisi hau aztertutako arazoak modu bisual batean batera jartzen dituen probabilitate-eragin matrize batekin amaituko da. Matrize honek arrisku bakoitza gertaera probabilitate eta eraginaren arabera kokatzen du. Kokapenaren arabera arrisku maila antzeman ahalko da, gelaxkak koloreen arabera taldekatu dira eta: kolorea geroz eta gorriagoa bada, arrisku maila orduan eta altuagoa izango da eta kolorea geroz eta berdeagoa bada, arrisku maila orduan eta baxuagoa izango da.

		Eragina				
		%10	%30	%50	%70	%90
Probabilitatea	%10					**A1**
	%30				**A2**	**A3**
	%50			**A4**		
	%70					
	%90					

4. Taula: Probabilitate-eragin matrizea

Taulan ikusi daitekeenez, kontuan izan beharko da A3 arriskua, honen eragin altua dela eta. Diseinuaren egiaztapena oso garrantzitsua izango da, alde batetik amaierako emaitzaren funtzionamendua egokia dela bermatzeko eta bestetik erabiltzen diren osagaiek akatsak izatekotan hauek kokatu eta konpontzen laguntzeko.

Eragin bereko A1 arriskuari dagokionez ezingo da neurri handirik hartu, RISC-V-ren gaur egungo egoera eta etorkizunera begira proiekzioa nolakoa izango den aztertzeaz gain. Hala ere, azaleko analisi bat egin eta gero, momentuz behintzat argi dago arrisku hau ez dela oso probablea.

Aldiz, A2 arriskua ekiditeko, nahikoa izango da ASIC baterako diseinu bat egin behar dela gogoratzea eta kasu honetan aplikatu beharreko kodetze teknika gogoan izatea, IP itxien erabilera saihesteaz gain.

Azkenik, proiektu orotan aurki daitekeen A4 arriskua dago, gertaera-probabilitate eta eragin ertainarekin. Kasu honetan, lehen egindako analisian aipatu diren ekidite neurriak aplikatu beharko dira, buruan izanda arazo hauek gertatuz gero konponketa erlatiboki erraza izango dutela baina ondorio bezala atzerapenak ekarriko dituztela.

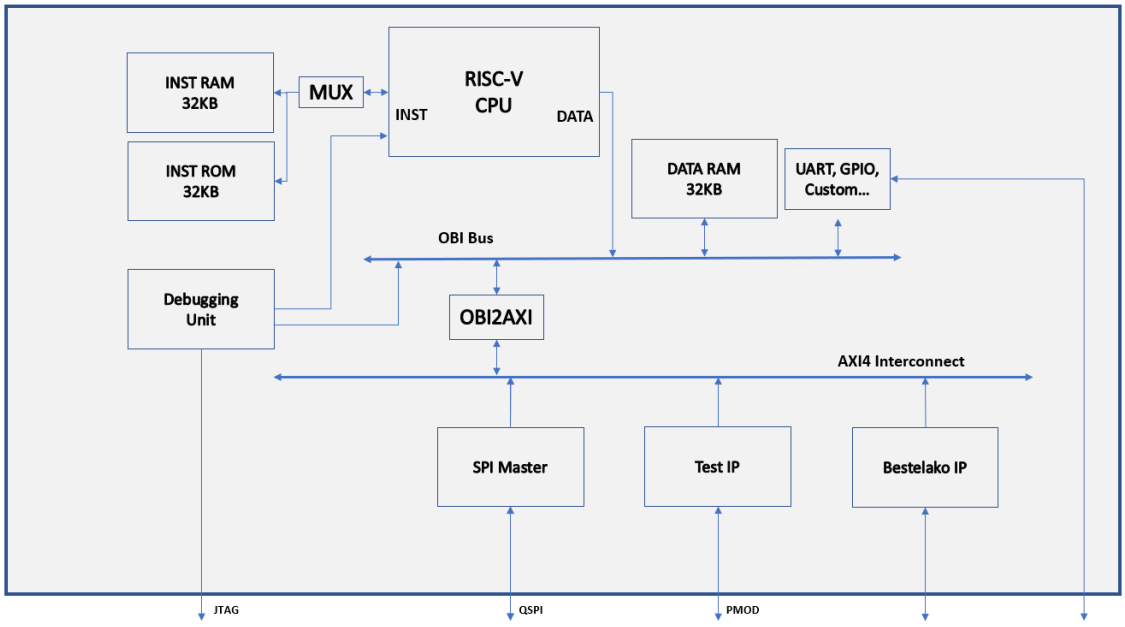
Orokorrean arriskuak zonalde laranja-gorrian kokatzen dira. Ondorioz, gainbegiratzea oso beharrezkoa izango da arazo hauek ekiditeko, arrisku potentzial batzuk larriak baitira. Hala ere, arrisku hauen gertaera probabilitatea baxua da orokorrean, proiektuaren garapen egonkorra litekeena dela ondorioztatu daitekeela.

8. Proposatutako ebazpenaren deskribapena

Atal honetan, garatutako sistemaren sorrera prozesuaren inguruko xehetasunak emango dira. Horretarako, lehenik eta behin sistemaren ikuspegi orokor bat emango da goi mailako diseinua aurkeztuz. Ondoren, sistemaren atal bakoitza deskribatuko da eta, azkenik, honen funtzionamendua argituko da memorien konfigurazioa erakutsiz.

8.1 SoC diseinuaren ikuspegi orokorra

Aipatu den bezala, Ibex corearen alde egin da SoC-aren prozesagailu gisa. Gainera, Ibex proiektuak eskaintzen duen *ibex-demo-system*-a hartu da diseinuaren abiapuntu bezala, honi dagokion aldaketak aplikatu zaizkiola lortu nahi den sistema lortzeko. Jarraian erakusten den irudian, inplementatu den diseinuaren bloke-diagrama aurkezten da.



3. Irudia: Diseinatutako sistemaren bloke-diagrama

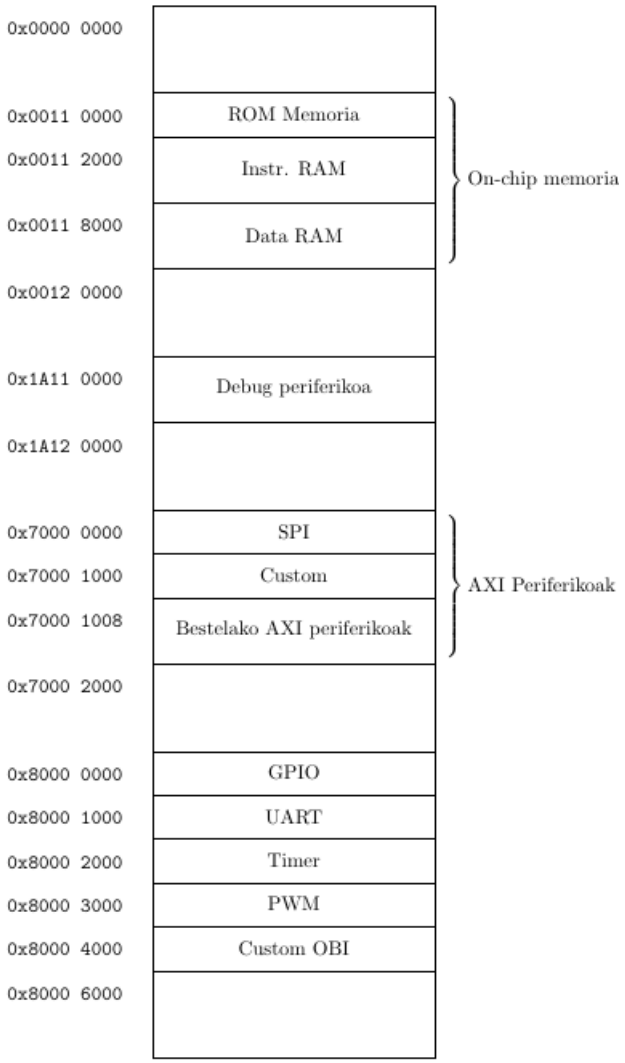
Diagraman ikusi daitekeen bezala, sistema prozesagailu bat, *debug* unitate bat, memoria hiru, interkonexio bus bi eta hainbat periferikok osatzen dute. Alde batetik, UART edo GPIO bezalako periferiko orokorrak eta datuen memoria OBI bus natiboan ipini dira [25], bestelako IP konplexuagoak AXI interkonexio busean jarri direla. Honela, Ibexen OBI interfaze natiboa errespetatu da, abiadura altuko periferikoei bus egoki bat eskainiz eta bestalde, industria mailan hain hedatuta dagoen AXI busa eskaini dela ere.

Prozesadoreak periferiko zehatzetara atzipena memoria helbideen bidez lortzen du, hauek beraz *memory-mapped* periferikoak direla. Periferiko bakoitzari 32 biteko helbideratze-espazio osoko helbide-tarte bat dagokio. Datuen memoria gainontzeko periferikoak bezala atzitzen da, honetara atzipena *read* edo *write* operazioetarako egin daitekeela. Honela, datu memoria edota bestelako periferikoetara atzipena datu memoriaren interfazearen bidez egiten dira.

Bestalde, aginduak edo instrukzioak gordeko dituzten instrukzioen ROM eta RAM memoriak prozesagailuaren instrukzioen memoriarako interfazera konektatuta aurkitu daitezke.

Azkenik, *debug*-erako IP bloke bat gehitu da sisteman. Bloke honek SoC-eko osagai guztietara du atzipena, akatsak bilatzerako garaian sistemaren puntu guztiak aztertu ahal izateko.

Jarraian sistemaren memoria mapa erakusten da. Bertan bloke diagraman aurkitzen diren periferiko bakoitzak hartzen duen helbide-tartea ikusi daiteke.



4. Irudia: Diseinatutako sistemaren memoria-mapa

Bai bloke-diagraman bai memoria mapan ikusi daitekeenez, sistema nahiko sinplea da; ez da errendimendua igo dezaketen *cache* edota DMA bezalako mekanismoen inplementaziorik egin. Honen arrazoi nagusia proiektu honen emaitza ASIC batera eraman nahi den lehenengo diseinua dela da, beraz kostu eta konplexutasun baxuko diseinu bat lortu nahi zela. Bestalde, prozesatze-sistema kontrol ataza sinpleetarako erabili nahi denez, hau ez da errendimendu altukoa izan behar ere ez.

8.2 Diseinuaren xehetasunak

8.2.1 Prozesagailuaren konfigurazioa

Aipatu den bezala, Ibex kontrol atazetarako erabiliko da soilik diseinu honetan. Hori dela eta, helburu honetarako hobekien egokitzen den Ibexen konfigurazioa aukeratu da.

Ibexen SystemVerilog kodeak pertsonalizazioa errazten du prozesatzailea instantziatzerako garaian. Prozesagailuak hainbat ezaugarri izan ditzake aukeran, ezaugarri guztiak ez direla bateragarriak haien artean. Hori dela eta, LowRISC-ek hainbat instantziatzeko defektuzko modu gomendatzen ditu, hauek funtzionamendu egokia bermatzen duten ezaugarriak biltzen dituztela. Aukeran dauden moduetatik, interesgarriena *small* modua dela ondorioztatu da.

Modu hau aukeratuta, Ibexek 2 etapetako *pipeline* eta hiru zikloko biderkatzaileak inplementatzen ditu, RV32IMC luzapenak inplementatzen dituela. Moduaren izenak adierazten duen bezala, hau da Ibexen inplementazio minimoena ezaugarriei dagokionez, baina baita ere area txikiena erabiltzen duena.

Modu honen alde egin da kontrol atazetarako ez delako errendimendu altu bat behar, eta tamaina txikia izatea garrantzitsua baita proiektu honetan. Hala ere, *maxperf-pmp-bmbalanced* modua aipatu beharra dago, etorkizunerako interesgarria izan baitaiteke; modu honek ziklo bakarreko biderkatzailea eta hiru etapetako *pipeline*a inplementatzen du, Ibexen konfigurazio ahaltsuena lortuz. Hala ere, horretaz gain RISC-V-ren B luzapena inplementatzen du, sistema txertatuetarako egokia dena. Bestalde, PMP segurtasun mekanismoa inplementatzen du ere.

8.2.2 OBI Busa

Prozesagailura zuzenean lotuta, periferikoetara atzipena ematen duen OBI busa dago. On-chip komunikazio protokolo arruntek bezala, OBIk *master/slave* eredua inplementatzen du, aktore bi hauen artean puntutik-punturako komunikazioa ematen dela. Hainbat *master* edo *slave* haien artean konektatzeko, *interconnect* elementu bat erabiltzen da.

OBI protokoloa bera oso sinplea denez (kontrol seinale zazpi ditu soilik), bere erabilera erraza da eta hardware sinplea behar du. Gainera, *Ibex-demo-system*-ean erabiltzen den busa da, corearen datu-memoriarentzako interfazearen seinale berdinak dituela.

Bus hau mantentzea erabaki da, bere erabilerarekin lehen aipatutako abantailak aprobetxatu daitezke eta, bertara abiadura altuko periferikoak konektatu daitezkeela.

Busaren *master* bloke bakarrak Ibex corea bera eta debug unitatea dira, gainontzeko periferiko guztiak *slave* motakoak direla. Datuen memoria OBI busean kokatzen dela aipagarria da, honek, OBI buseko gainontzeko *slave*-ek bezala, OBI slave interfaze bat duela.

Busaren funtzionamendua ulertu eta frogatzeko, *Ibex-demo-system*-ean PWM, timer, UART eta GPIO periferikoak gehitzen dira. Hala ere, *custom* izeneko periferiko sinple bat gehitu da, idazketa edo irakurketarako 8 biteko erregistro bat gordetzen duenak. Frogazko periferiko hau OBIren seinaleen funtzionamendua guztiz ulertzeko gehitu da, etorkizunean OBI busera beste periferiko bat gehitu nahiko balitz prozesu hau nola egin jakiteko.

8.2.3 AXI Busa

Hasiera batean, diseinatu beharreko SoC honetan AXI busa erabili nahi zen arrazoi bi nagusi zirela eta. Alde batetik, on-chip komunikazio protokolo erabilienetariko bat delako, periferiko estandar askoren komunikazio-protokoloa dela; bestetik, APERT departamentuarentzako aski ezaguna den protokoloa delako. Honela, diseinura bestelako periferiko batzuk gehitu nahiko balira, AXI bus bat izateak asko erraztuko luke prozesu hau.

Hori dela eta, AXI bus bat gehitzea erabaki da. Hau OBI busera konektatu da, OBI – AXI Full bihurgailu bat erabiliz. Honela, AXI busa eta bere barneko periferikoak OBI busaren ikuspuntutik periferiko bakar bat balira bezala gehitu dira, AXI buseko periferikoek erabiltzen duten helbide-espazio osoa hartzen duen periferiko bat bezala, alegia.

Noski, sistema zehatz honetan, AXI busean *master* bakarra OBI – AXI bihurgailua da, gainontzeko AXI periferikoak *slave* motakoak direla.

OBI – AXI bihurgailura AXI Interconnect bat gehitu da, bertara AXI darabilten periferiko bi gehitu direla:

- **Test IP bat:** Bloke hau OBI busean aurkitu daitekeen *custom* blokearen oso antzekoa da, irakurketa edo idazketarako erregistro bat inplementatzen duenak. Erregistroaren edukia prozesatze-sistematik kanpora eraman da.
- **SPI Master bloke bat:** Quad SPI protokoloa *master* bezala inplementatzen duen IP bat gehitu da, kanpoko gailuekin komunikazioa ahalbidetzeko.

8.2.4 SPI Master AXI periferikoa

Periferiko honen helburua Quad SPI protokoloaren inplementazioa da, SoC-etik kanpo dauden zirkuitu integratuekin komunikatzeko. Protokolo honen erabilera oso zabaldua dago, komunikazio bus bakar batean hainbat zirkuitu integratuekin ahalbidetu dezakeelako elkarrekintza.

Hardwarearen aldetik, periferiko honek hainbat atal ditu bere barnean. Jarraian, erregistro hauek zerrendatzen dira, periferikoaren funtzionamendua hobeto ulertzeko asmoarekin.

- **AXI Erregistroak:** Periferikoaren atal hau arduratzen da AXI protokoloaren logikaren inplementazioaz, sistemaren AXI busetik iristen zaizkion irakurketa/idazketa esakerak kudeatzeko erabiltzen direla.
- **SPI Master kontrolagailuak:** Atal honetan inplementatzen da SPI protokoloa, hemen gauzatzen dela SPI logika guztia. Honela, kanpoko mundura joango diren seinaleak aurkitu daitezke hemen, hala nola SPI busean konektatuko diren seinaleak edota busean erabiliko den master erloju seinalea.
- **Transmisio eta harrera FIFO ilarak:** Ilara hauek guztiz beharrezkoak dira datuak nahi diren formatuan bidaltzeko. SPI Buseko transakzio bakoitzak 32 bit baino gutxiagoko datuak erabiltzen dituenez, esate baterako, bidalketan, 32 bit hauek ezingo dira batera bidali. FIFO ilara batean sartu beharko da datua, 8naka bidaliko dela. Berdina gertatzen da harreraren kasuan.

8.2.5 *Debug* periferikoa

Lehen aipatu den bezala, periferiko honek *master* rola du OBI busean, bere *debug* eginkizuna dela eta, corera atzipena behar duelako honen edukia aztertzeko eta periferikoetara atzipena behar duelako hauek ere kontrolatu ahal izateko.

Periferiko hau *ibex-demo-system*-ean aurkitu daiteke, eta JTAG protokoloa erabiltzen du kanpoko munduarekin komunikatzeko. Aukeraketa hau oso egokia da demo sisteman, hau JTAG USB bidez inplementatzen duen Arty plakara zuzenduta baitago, baina JTAG protokolo oso ezaguna denez helburu hauetarako, *debug* unitate hau guztiz baliagarria da proiektu honetarako ere.

Gainera, LowRISC taldeak garatutako softwareari esker, GDB *debuggerra* erabili daiteke *debug* prozesuan, akatsak aurkitzea erraztuz.

8.2.6 On-chip memoriak

Sistemaren bloke-diagraman ikusi daitekeen moduan, SoC-aren datu eta instrukzioen memoriak banatuta daude. Honen arrazoia Ibex prozesagailu beraren diseinua da, instrukzioak soilik irakurtzeko interfaze bat eta datuak irakurri eta idatzi ahal izateko beste memoria interfaze bat duela, Harvard arkitektura jarraituz. Instrukzioen interfazean memoriak prozesagailura interfaze natibo baten bitartez zuzenean lotzen direla ikusi daiteke, datuen interfazean aldiz OBI protokoloa erabiltzen dela, datuen memoria OBI busera lotuta baitago.

Datuen memoriaren kasuan funtzionamendua sinplea da. Lehen aipatu den moduan, datuen memoria OBI buseko bestelako periferiko arrunt bat bezalakoa da; prozesagailuak hau irakurketa edo idazketarako atzitu ahal dezan, datuen memoriari dagokion helbidea adieraziz egin beharko du prozesagailuak bere datuen memoriarako interfazean, OBI master bezala irakurketa edo idazketa transakzioak egiteko erabiltzen duena.

Bestalde, instrukzioen memoriarentzat interfaze natibo bat definitzen da. Hau OBI protokoloaren antzekoa da, hau baino oraindik sinpleagoa dela. *Ibex-demo-system* originalean, prozesagailuaren instrukzioen memoriarako interfazea instrukzio memoriara zuzenean lotuta dago. Hau erabilgarria izan daiteke FPGA baterako froga gisa; *Ibex-demo-system*-ean exekutatu nahi den programa konpilatua sistemaren instrukzio-memorian “erre” ondoren, diseinua FPGAn inplementatu daiteke. Gailua abiatzean, coreak instrukzio-memoriako aginduak irakurriko ditu zuzenean, kodea exekutatuz. Hala ere, hau ez da ASIC baterako egokia, gailu hauetan ezin baita chiparen barnean dagoen memoria baten edukia hain erraz aldatu.

SoC-a bere kabuz piztu ahal izateko, garatzaileak chip barruko instrukzioen RAMaren edukia aldatu behar gabe alegia, abiatze edo *boot* prozesu generiko bat definitzen duen programa bat behar da, programa hau *read-only* memoria batean gorde behar dela.

Boot programa honek SoC-etik kanpo dagoen biltegiratze gailu ez-hegazkor batetik irakurri beharko du exekutatu nahi den programa, esate baterako, Flash memoria batetik. Kanpoko memoriatik programa irakurtzen den heinean, edukia SoC-eko instrukzioen memorian idatzi beharko du *boot* programak. Boot programaren amaieran, instrukzio memoriaren lehenengo helbidera (ez literalki, baizik eta programaren *entry point*-era) jauzi egin beharko da.

Prozesu honetan, Flash memoriara atzipena honekin komunikatzeko egokia den protokolo bat darabilen periferiko bat behar da. Kasu honetan, AXI busean kokatutako SPI master periferikoa hautatu da, flash memoria batekin komunikatzeko. SPI *master* periferikoa AXI busean kokatua dagoenez, prozesagailuak memoriara mapeatutako beste edozein periferikotatik bezala egingo ditu irakurketak flashetik.

Ondorioz, instrukzioen RAM eta *boot* ROM memoria biak atzitu behar ditu prozesagailuak. Memoria hauek soilik aginduak gordetzen dituztenez, prozesagailuak instrukzioen memoriarako duen interfaze natibotik atzitu behar ditu. Interfaze batetik memoria bi atzitu nahi direnez beraz, multiplexore moduko bloke bat gehitu behar izan da.

Multiplexore honetan, prozesagailuaren instrukzioen memoriarako interfazetik bidaltzen diren seinaleak aztertzen dira. *Boot* eta aginduen memoriak helbide-mapan helbide-tarte ezberdin dutenez, multiplexoreak prozesagailuak adierazitako helbidearen arabera memoria bati edo besteari bidaltzen dio irakurketa-eskaera. Honetaz gain, bloke multiplexore honek prozesagailuaren interfaze natiboko gainontzeko kontrol-seinaleak kudeatzen ditu.

Planteatutako sistema honek beste konfigurazio berezi bat behar du memoriaren atalean. Aipatu den moduan, *boot* ROM memoria *read-only* motakoa izan beharko da, beti berdina izango den programa bat gordeko du eta. Programa hau 3. eranskinean erakusten da, bere atal ezberdinen inguruan azalpenak ematen direla.

Hala ere, instrukzioen RAM memorian idazketa ahalbidetu behar da ere. Printzipioz, programa baten exekuzioan ez dira idazketak egongo memoria honetan, baina *boot* prozesuan exekutatzen den programak, flashetik irakurritako programa alegia, instrukzioen RAM memoria honetan idatzi beharko da. Idazketa gainera datuen memoriaren interfazetik egiten du prozesagailuak.

Arazo hau konpontzeko soluzio sinple bat proposatu da; datu eta aginduen memoria elementu fisiko bakar batean biltzea *dual port* RAM baten bidez. Honela, fisikoki bakarra den memoria elementu bat erabiltzen da, logikoki bitan banatutako memoria bat dela. Izenak adierazten duen bezala, memoria honek portu bi izango ditu, bata datuen memoriarentzako eta bestea instrukzioen memoriarentzako.

Instrukzioen memoriaren portua memoria multiplexorera dago lotuta, irakurketarako seinaleak soilik biltzen dituela. Datuen memoria aldiz, OBI busera dago lotuta, irakurketa eta idazketa ahalbidetzen duela. Hala ere, *dual port* RAM memoria baten izaera dela eta, portu batetik beste portuak irudikatzen duen memoria logikora atzipena lortu daiteke ere. Beraz, instrukzioen memorian idazketak datuen memoriari dagokion portutik egin ahalko dira.

8.2.7 Softwarean moldaketak

Hardwarea erabiltzeko liburutegiak

Software programak idaztean, beharrezkoa da edozein periferiko erabiltzeko hainbat definizio eta funtzio biltzen dituzten liburutegien sorrera. Orokorrean, periferiko bakoitzeko, hauen erregistro ezberdinek dituzten helbideak eta hauetan irakurketak edo idazketak egiteko funtzioak definitzen dituzten liburutegiak behar dira.

Kasu honetan, *ibex-demo-system* proiektua hartu denez abiapuntutzat, liburutegi gehienak idatzita zeuden jada. Hauen artean, sistemaren definizio eta funtzio orokorrak biltzen dituen *demo_system* edota *dev_access* liburutegiak daude, ondoren sisteman bildutako hainbat periferikoen liburutegiak aurkitu daitezkeela ere.

Bestalde, gehitu den SPI master periferikoaren liburutegiak gehitu behar izan dira. Honetan hainbat definizio gauzatu dira, hala nola lehen aipatutako erregistroen kokapena. Funtzioei dagokionez, periferikoa flash memoria batekin erabiliko dela hartu da kontuan, helburu honetarako egokiak diren funtzioak definitu direla. Esate baterako, SPI bidez memoriari komandoak bidaltzeko funtzioa edota periferikoaren harrera/transmisio bufferretan irakurketak edo idazketak (hurrenez hurren) egiteko funtzioak definitu dira.

Konpilazio prozesuan aldaketak

Sistemaren memoria eskeman aldaketak sortzeak memoria bakoitzera zuzendutako softwarearen konpilaziorako aldaketak eskatzen ditu ere. Konpiladoreak programa bakoitza zein memoriatan gordetzen den jakin behar du, programaren aginduei dagokion sistemaren helbide-tarteko helbideak esleitu ahal izateko.

Memoria-mapa aztertuz, adibidez ROM memoriaren helbide tartea 0x0011 0000 – 0x0011 1FFF dela ikusi daiteke. Memoria honetan gordetzen diren programen aginduak helbide-tarte horretako helbideekin egin beharko dira lerrokatuta.

Ibexen kasuan, LowRISC-en GCC *tooclhaina* erabili denez, GCCri helbide tarte hauek *linker script* baten bidez adierazi zaizkio. GCCren konpilazio prozesuan, C lengoaian idatzitako kodea lau etapatatik pasatzen da, emaitza bezala programa formatu bitarrean lortzeko: Aurreprozesagailua, direktibak prozesatzeko; konpiladorea eta mihiztatzailea, kodea RISC-V mihiztatzaile lengoaiara itzultzeko eta objektu fitxategiak sortzeko; eta azkenik linkerra, kode bitarra lortzeko. Linkerra *linker script* batekin hornituz gero, kodearen barnean segmentu ezberdinak definitu daitezke, segmentu bakoitza zein helbide-tartean kokatu nahi den adierazi daitekeela.

Bestalde, linker scriptean *stackaren* erabilerarako datu memoriaren helbide-tartearen zati bat adierazi zaio linkerrari. Hau ere beharrezkoa da, konpilatzaileak stackean egin beharreko eragiketak prozesatzailearentzat irakurketa eta idazketarako erabilgarri dagoen zonalde batean egin behar baitira. Bai ROM eta bai instrukzioen RAMean gordetako programei *stackerako* zonalde bera adierazi zaie.

Honela, *linker script* bi sortu behar izan dira: bata ROM memoriara zuzendutako programentzat eta bestea RAM memorietara zuzendutako programentzat. Script hauen hasieran zonalde nagusiak adierazi dira, jarraian adierazten den bezala. Hona hemen ROMera zuzendutako programen *linker script*aren direktibak.

```
MEMORY
{

    rom          : ORIGIN = 0x00110000, LENGTH = 0x2000 /* 8 KiB */
    stack        : ORIGIN = 0x0011E000, LENGTH = 0x2000  /* 8 KiB */

}
```

5. Irudia: Gauzatutako diseinuaren ROMera zuzendutako programen linker scripta

Eta jarraian RAMera zuzendutako programen *linker script*aren direktibak.

```
MEMORY

{

    ram          : ORIGIN = 0x00112000, LENGTH = 0x6000 /* 24 KiB */
    stack        : ORIGIN = 0x0011E000, LENGTH = 0x2000  /* 8 KiB */

}
```

6. Irudia: Gauzatutako diseinuaren RAMera zuzendutako programen linker scripta

Zonaldeak adierazi ondoren, kodearen atal edo segmentu bakoitza identifikatu da linker scriptean, bakoitza zonalde batera edo bestera esleitu dela, bestelako aukera batzuk adierazteaz gain. Zonalde hauek programa batean aurkitu daitezkeen segmentu tipikoak dira: *text, rodata (read-only data), data, BSS* eta *stack* segmentuak alegia.

*Linker script*ak definitu ondoren, C lengoaian idatzitako kodea modu arrakastatsuan konpilatu da, kode bakoitzari dagokion formatu bitarreko programen aginduek helbide zuzenak dituztela. Hau modu argian egiaztatu da *toolchain*aren *objdump* tresnarekin.

Tresna honek konpilatutako eta *elf* (*Executable and Linkable Format*) formatuko fitxategi batetik abiatuta, programa horri dagokion informazioa biltzen duen testu fitxategi bat sortzen du. Bertan, programaren agindu bakoitza aurkitu daiteke, honen helbidea, agindua bera hexamartar gordinean eta honen mihiztatzailera itzulpena adierazten direla. Adibide gisa, ROMera zuzendutako programa konpilatu bati *objdump* aplikatuta lortutako informazioren zati bat erakusten da.

```
00110084 <main>:

110084:   1101              c.addi  x2,-32
110086:   ce06              c.swsp  x1,28(x2)
110088:   cc22              c.swsp  x8,24(x2)
11008a:   1000              c.addi4spn  x8,x2,32
11008c:   00000517          auipc   x10,0x0
110090:   7f850513          addi    x10,x10,2040 # 110884 <sleep_loop+0x6>
110094:   28ad              c.jal   11010e <puts>
110096:   fe042623          sw  x0,-20(x8)
11009a:   a039              c.j 1100a8 <main+0x24>
11009c:   0001              c.addi  x0,0
11009e:   fec42783          lw  x15,-20(x8)
1100a2:   0785              c.addi  x15,1
1100a4:   fef42623          sw  x15,-20(x8)
1100a8:   fec42703          lw  x14,-20(x8)
1100ac:   67e1              c.lui   x15,0x18
1100ae:   69f78793          addi    x15,x15,1695 # 1869f <_stack_len+0x1669f>
1100b2:   fee7d5e3          bge x15,x14,11009c <main+0x18>
1100b6:   bfd9              c.j 11008c <main+0x8>
```

7. Irudia: Konpilatutako programa baten *objdump*a

Ikusi daitekeenez, agindu hauek *main* etiketadun programaren zatiaren parte dira. Lehenengo zutabean aginduari dagokion helbidea ikusi daiteke, kasu honetan ROM memoriaren tartekoak direla.

Azkenik, bitar formatuan lortutako kodea sistemaren osagaietan sartu behar izan da. Horretarako, hexamartar formatura itzuli behar izan dira formatu bitarreko kodeak eta gero, ROMaren kasuan, Verilog lengoaiaren *readmemb* direktiba erabili da hexamartar formatuko programa Verilogen deskribatutako memorian kargatzeko. RAMaren kasuan, Arty plakak duen flash memorian gorde da hexamartarrean idatzitako programa, Vivadok memoria hauek populatzeko duen tresna erabiliz.

9. Lan plana

Dokumentuaren atal honetan, proiektua aurrera eraman ahal izateko jarraitutako lan plana aurkezten da. Jarraian esandako plangintza xehetasun osoz aurkezten da, horretarako lehenik eta behin proiektua gauzatu duen lan-taldea eta baliabideak aurkeztuz. Ondoren, proiektu hau gauzatzeko beharrezkoa den garapen prozesua atal ezberdinetan banatu denez, atal edo fase bakoitzaren xehetasunak deskribatuko dira. Atal bakoitzari lan-pakete (LP) deitu zaio eta bakoitzaren baitan hainbat ataza (A) bereizten dira.

9.1 Lan-talde eta baliabide materialak

Proiektu hau osatzen duen lan-taldea jarraian aurkezten den taulan ikusi daitekeen pertsonek osatzen dute.

Kodea	Izena	Erantzukizuna	Rola
I1	Jesús Lázaro Arrotegui	Senior ingeniaria	Proiektuaren gainbegiratzea eta zuzenketa
I2	Armando Astarloa Cuéllar	Senior ingeniaria	Proiektuaren gainbegiratzea eta zuzenketa
I3	Unai Galicia Zabala	Junior ingeniaria	Proiektua gauzatzea

5. Taula: Proiektuaren lan-taldea

Baliabide materialei dagokionez, beharrezko elementu hiru identifikatu dira. Alde batetik, Vivado edota HDL simulatzaileak bezalako EDA (*Electronic Design Automation)* tresnak dituen zerbitzari ahaltsu bat beharrezkoa izango da, programa hauen exekuzioa ordenagailu pertsonal batean oso motela izango bailitzake eta. Zerbitzaria proiektu osoan zehar piztuta dagoela hartu da kontuan.

Bestalde, diseinuen portaera aztertzeko FPGAdun plaka bat behar da, baliabide gutxiko eta Xilinx/AMD etxearen FPGA baten alde egin dela Artix-7 35T modeloa duen Arty plaka aukeratuz.

Azkenik, zerbitzaria erabiltzeko eta dokumentazioa aztertzeko ordenagailu bat beharko da. Hurrengo taulan aurkezten dira aipatutako baliabide materialak:

Kodea	Materiala	Ezaugarriak	Kopurua
SRV	EDA zerbitzaria	EDA tresnak	1
FPGA	FPGAdun Arty plaka	Artix-7 35T FPGA, flash memoria	1
PC	Ordenagailua	-	1

6. Taula: Proiektuan erabilitako baliabide materialak

9.2 Lan-paketeak eta atazak

Dokumentuaren "Lan plana" atal honen hasieran esan den moduan, proiektuaren plangintza lan-pakete (LP) ezberdinez osatua dago, zeintzuk aldi berean ataza (A) ezberdinez osatuta dauden. Jarraian lan-pakete eta ataza bakoitza deskribatuko dira, hauen xehetasunak, epeak, etab. deskribatuz.

LP1 - Proiektuaren kudeaketa eta dokumentazioaren garapena

Fase honetan aurki daitezkeen ataza biak proiektu osoan zehar gauzatuko dira, izan ere ataza hauetan proiektuaren garapena jarraitu eta honen dokumentazioa bete beharko delako. Proiektuaren garapena eta honi dagokion dokumentazioa zuzena dela bermatzeko asmoarekin, hainbat bilera egingo dira ingeniari junior eta seniorren artean. Jarraian, aipatutako ataza biak deskribatzen dira:

- **A101 – Proiektuaren garapenaren jarraipena:** Jarraipenaren bitartez, proiektuaren garapenean emandako aurrerapausoen edota arazoen berri emango da. Gai hauek lan-talde osoarekin aztertuko dira, bileren bitartez. Ataza honen bitartez proiektuaren garapena egokia dela bermatu nahi da.

  Iraupena: *32 ordu proiektu osoan zehar*

- **A102 – Dokumentazioaren garapena:** Atazan honen bitartez proiektua eta honen garapena deskribatzen duen dokumentazioa osatuko da. Behin proiektua amaituta, fase honetan zehar bildutako informazioarekin amaierako memoria bat idatziko da. Lan-pakete honen deskribapenean aipatu den moduan, ataza hau proiektua garatzen den aldi berean beteko da.

  Iraupena: *56 ordu proiektu osoan zehar*

Ataza hauek amaitzean ondoko mugarria beteko da.

- **M1 – Proiektua amaituta:** Proiektu osoa biltzen duen dokumentazioa osatzean, proiektua amaitutzat hartu daiteke.

Fase honetarako beharko diren giza baliabideak, baliabide materialak eta bakoitzaren beharrezko lan-orduak jarraian aurkezten dira:

- ***I1, I2:*** *32 ordu bakoitzak*
- ***I3:*** *88 ordu*
- ***PC:*** *88 ordu*

LP2 - Proiektuaren definizioa

Fase hau hasieran kokatzen da, bertan proiektua abian jarriko delako, helburu nagusiak edota irismena bezalako oinarrizko kontzeptuak zehaztuko dira eta. Proiektuaren oinarriak atal honetan zehaztuko direnez fase hau oso garrantzitsua da, puntu honetan hartutako erabakiak jarraian datozen faseak definituko ditu eta. Lau ataza ezberdindu dira lan-pakete honetan:

- **A201 – Proiektuaren helburuen definizioa:** Ataza honetan proiektuaren betebeharrak definituko dira. Helburuen definizioak proiektua bukatzen denean sortutakoak izan behar dituen ezaugarriak zeintzuk izan beharko diren zehazten du, beraz oso garrantzitsua izango da atal honen gainean arreta handia jartzea, lana amaitzerakoan emaitza ona izan dadin.
  ***Iraupena**: 56 ordu*
- **A202 – Proiektua zuzen garatzeko teknologien ikerketa:** Azterlan honen bitartez, proiektuarekin ekin baino lehen adostutako helburuak betetzeko erabiliko diren teknologien inguruko azterketa bat egingo da; helburu nagusia SoC-aren prozesagailuaren ezaugarriak edota sistema-barneko komunikazio teknologien artean egokienak aukeratzea dela.
  ***Iraupena:** 24 ordu*
- **A203 – Arriskuen analisia:** Proiektuaren garapenean zehar agertu daitezkeen arazo ezberdinak identifikatu beharko dira atal honetan. Identifikazioaz gain, arrisku mailaren araberako sailkapen bat egingo da, arrisku bakoitzaren gertaera probabilitatea eta, gertatzekotan, izango lukeen eragina kuantifikatuz. Gainera, arrisku bakoitza ekiditeko argibideak zehaztu beharko dira.
  ***Iraupena:** 32 ordu*

Ataza hauek amaitzean ondoko mugarria beteko da.

- **M2 – Proiektua definituta:** Proiektuaren helburuak eta hauek betetzeko tresnak aukeratu ondoren, garapenean gerta daitezkeen arriskuak aurreikusita ere, proiektuaren irismena eta helburuak guztiz argituta egongo dira.

Fase hau burutzeko beharrezkoak izango diren baliabideak jarraian aurkezten direnak dira:

- ***I1, I2:** 56 ordu bakoitzak*
- ***I3:** 112 ordu*
- ***PC:** 112 ordu*

LP3 – RISC-V Prozesagailuen alternatiben analisia

Fase honetan, proiektuaren alderdi orokorrak aztertu ondoren, diseinatu nahi den SoC-aren oinarri izango den RISC-V prozesagailua aukeratu beharko da. Ataza hau lan pakete gisa definitu da, aukera bat hartzeko hainbat azpi-ataza bete behar baitira. Prozesagailu baten diseinu egokia aukeratu ondoren, sistemaren gainontzeko atalak gehitu ahalko dira. Bost ataza definitu dira lan-pakete honetan:

- **A301 – Alternatiba egokien zerrendatzea:** Jakina denez, proiektu honetan kode-irekiko prozesagailu bat erabili nahi izan da. Mota honetako prozesagailuen eskaintza zabala dela eta, proiektura gehien egokitu daitezkeen diseinuak aukeratzeko helburuarekin azterketa bat egingo da, eskuragarri dauden aukera anitzen arteko azaleko konparaketa bat eginez.
  Iraupena: *40 ordu*

- **A302 – Alternatiba bakoitzaren dokumentazioaren azterketa:** Alternatiben kopurua aukera gutxi batzuetara mugatu ondoren, bakoitza sakonki aztertu beharko da. Honetarako, prozesagailu bakoitzak bera deskribatzen duen dokumentazioa irakurri beharko da, aukera banak eskaintzen dituen ezaugarriak eta hauen egitura ulertu ahal izateko.
  Iraupena: *72 ordu*

- **A303 – HDL analisia eta simulazioa:** Aurreko atazan alternatiba bakoitzaren egitura ondo aztertu ondoren, diseinuaren HDL deskribapena aztertu beharko da. Azterketa hau diseinua nola inplementatuta dagoen ulertzeko guztiz beharrezkoa izango da, baita ere diseinuaren gaineko moldaketak nola egin zehazteko. Ulermen hau bermatzeko, simulazioak egin beharko dira, sistemaren funtzionamendua seinale-mailan ikusi ahal izateko.
  Iraupena: *72 ordu*

- **A304 – Froga sistemaren diseinua aztertzea:** Corearen HDL deskribapena integratzen duen sistema bat diseinatu edo honelako eredu bat beharko da ataza honetan, core bakoitzaren funtzionamendua sistema batean aztertu ahal izateko. Corearen proiektuak diseinu bat eskaintzen badu, hau sakonki aztertu beharko da.
  Iraupena: *48 ordu*

- **A305 – Software hedapena eta FPGAn inplementazioa:** Corea erabiltzea ahalbidetzen duen sistemaren HDL deskribapena lortu edo sortu ondoren, hau martxan ipini beharko da. Honetarako, software bat hedatu beharko da sistemaren memorietan, ondoren honen funtzionamendu zuzena FPGA gailuetan inplementatuz frogatuko dela. FPGAren alde egin da softwarearen simulazioa oso astuna izan baitaiteke eta, software *debug* prozesua abiatu beharko dela froga programekin akatsak izatekotan.
  Iraupena: *56 ordu*

A302 eta A303 atazen arteko elkarlotura sakona dela eta, ataza biak paraleloan egitea erabaki da. Honen arrazoi nagusia dokumentazioaren azterketa HDL analisiarekin batera eginez, bien ulermen sakonagoa lortu daitekeela da.

Ataza hauek amaitzean ondoko mugarria beteko da.

- **M3 – Prozesagailua aukeratuta:** Puntu honetan, kode-irekiko prozesagailua aukeratuta egongo da, SoC diseinuaren oinarri den diseinua aztertua egongo dela ere.

Fase hau gauzatzeko beharrezkoak izango diren baliabideak jarraian zehazten dira.

- ***I1, I2:*** *30 ordu bakoitzak*
- ***I3:*** *288 ordu*
- ***PC:*** *288 ordu*
- ***FPGA:*** *108 ordu*

LP4 – Sistemaren diseinua eta inplementazioa

Sortu nahi den sistemaren prozesagailua aukeratuta eta honen funtzionamendua frogatzen duen oinarrizko sistema baten diseinua eginda, sistemaren hedapena gauzatu behar da. Hedapen honen helburua, amaierako sisteman izan nahi diren periferiko edo IP blokeen integrazioa erraztea da, horretarako interfaze ezagun bat eskainiz.

- **A401 – Diseinuan AXI bus baten integrazioa:** SoC-ean gehitu nahi diren periferikoek sistemako prozesagailu eta bestelako *master* elementuekin komunikatu ahal izateko, hauen arteko interkonexio bus ezagun bat beharrezkoa da. Taldeak gehitu nahi dituen periferikoek AXI4 interfaze bat darabiltenez eta taldeak berak bus honekin esperientzia handia duenez, bere alde egin da. Busaren funtzionamendua balidatu beharko da, periferikoak honen bitartez atzigarriak direla egiaztatuz.
  Iraupena: *32 ordu*

- **A402 – Periferiko berriak gehitu eta balidatu:** Behin periferikoen atzipenerako interkonexioa zuzena dela bermatu ondoren, hauek diseinura gehituko dira. Funtzionamendua egokia dela ikusteko, hauetara atzitu beharko da, hauen funtzionaltasun ezberdinak konprobatzeko software test batzuk exekutatu beharko dituela sistemaren prozesagailuak.
  Iraupena: *32 ordu*

- **A403 – Diseinua *standalone* eran lan egiteko moldatu:** Kasu honetan periferikoak dituen prozesatze-sistema oso bat diseinatu nahi denez, ezinbestekoa izango da bere kabuz abiatzeko kapaza den diseinu baten erabilera. Honetarako, beharrezkoak diren ROM memoriak eta memoria ezberdinen kudeaketa gauzatzen duten blokeak diseinatu eta gehitu beharko dira diseinura. Noski, elementu hauek gehitu eta gero, funtzionamendua zuzena dela bermatu beharko da.
  Iraupena: *48 ordu*

Ataza hauek amaitzean ondoko mugarria beteko da.

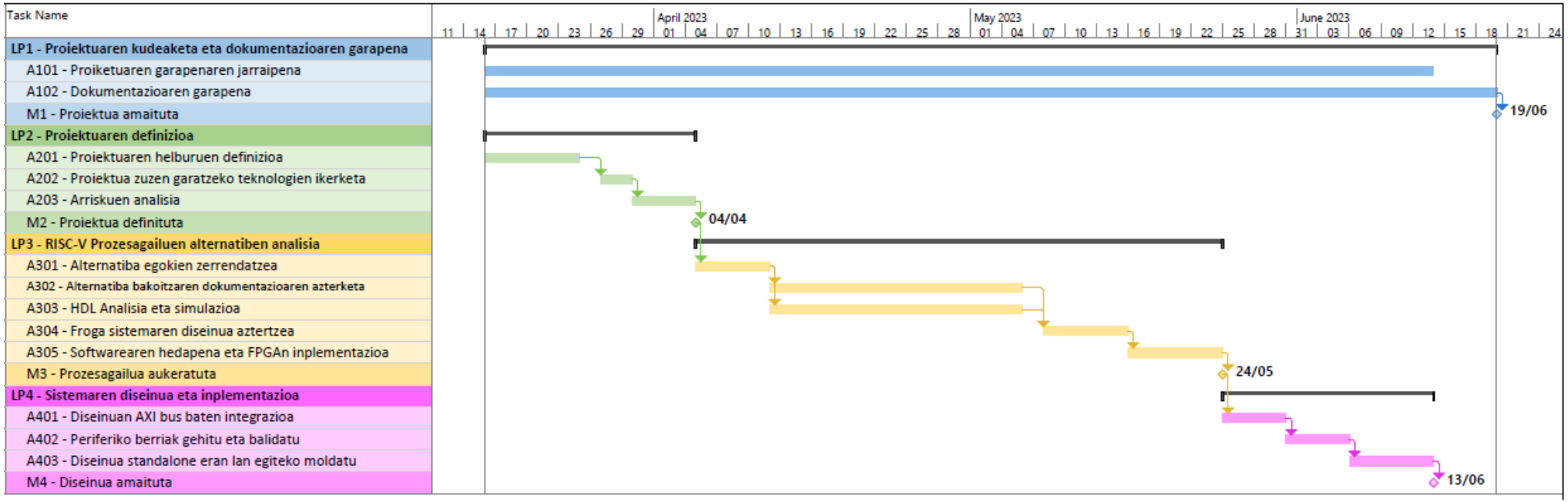
- **M4 – Diseinua amaituta:** Aipatutako atazak beteta, funtzionalki egiaztatutako eta beharrezko luzapenak dituen SoC-aren diseinua bukatuta egongo da, proiektuaren alde teknikoari amaiera emanez.

Fase hau gauzatzeko beharrezkoak izango diren baliabideak jarraian zehazten dira.

- ***I1, I2:*** *30 ordu bakoitzak*
- ***I3:*** *112 ordu*
- ***PC:*** *112 ordu*
- ***FPGA:*** *60 ordu*

9.3 Gantt diagrama

Dokumentuaren atal honetan, deskribatutako plangintza Gantt diagrama baten bitartez irudikatzen da, diagrama hurrengo irudian erakusten dela. Proiektuaren hasiera data bezala 2023. urteko martxoaren 16a hartu da, amaiera data bezala 2023 urteko ekainaren 19a hartu dela.



8. Irudia: Proiektuaren plangintza islatzen duen Gantt diagrama

10. Gauzatutako aurrekontuaren deskribapena

Dokumentuaren atal honetan, proiektu honen garapenak eragin dituen kostuen laburpen bat egingo da. Kostuak zenbatetsi ahal izateko, giza baliabideek eta baliabide materialek eragindako kostuak kontuan hartuko dira. Kostu hauen kalkulua egin ahal izateko, “lan plana” atalean zehaztutako ordu kopuruak hartuko dira kontuan.

10.1 Giza baliabideak

Plangintzan azaldu den moduan, hiru langilez osatutako lan-talde batek eraman du proiektu hau aurrera: bi ingeniari senior eta ingeniari junior bat. Hau kontuan hartuko da aurrekontua gauzatzerakoan, izan ere ingeniari seniorren orduko tasa handiagoa delako.

Beraz, jarraian aurkezten den 7. taulan giza baliabideen kostuen desglosea aurkezten da, bertan langile bakoitzaren orduko tasa, plangintzan adostutako ordu kopurua eta guztirako kostua adierazten direla.

KODEA	ERANTZUKIZUNA	ORDUKO TASA	ORDU KOPURUA	GUZTIRA
I1	Senior ingeniaria	60 €/h	148 h	8.880 €
I2	Senior ingeniaria	60 €/h	148 h	8.880 €
I3	Junior ingeniaria	25 €/h	600 h	15.000 €
			Guztira:	**32.760 €**

7. Taula: Barne-orduen kalkulua

10.2 Baliabide materialak

Proiektua aurrera eraman ahal izateko "lan-plana" atalean zehaztu diren baliabide materialak beharrezkoak izan dira. Hori dela eta, baliabide materialen kostua kalkulatu ahal izateko, hauen amortizazioa aztertu da.

Horretarako, analisia bi ataletan banatu da. Lehenik eta behin, beharrezkoa izan diren ordenagailu, FPGA plaka eta zerbitzariaren amortizazioak kalkulatu dira, honen hasierako kostua eta bizitza erabilgarria kontuan hartuz. Bestalde, proiektua garatzerakoan zerbitzaria uneoro piztuta egonen delako, honen kontsumo energetikoa kontuan hartu da ere. Proiektuaren luzapen absolutua, jaiegunak barne, 95 egunetan kalkulatu denez, denbora-tarte hau hartu da kontuan zerbitzariaren gastuak kalkulatzeko.

Ondorengo 8. taulan ordenagailuaren amortizazioaren kalkulua erakusten da.

KODEA	IZENA	KOSTUA	BIZITZA ERABILGARRIA	ORDU KOPURUA	GUZTIRA
PC	Ordenagailua	1.000 €	10.000 h	600 h	60 €
FPGA	FPGAdun plaka	260 €	5.000 h	168 h	8,74 €
SRV	Zerbitzaria	3.500 €	30.000 h	2.280 h	266 €
				Guztira:	**334,74 €**

8. Taula: Baliabide materialen amortizazioak

Erabilitako zerbitzariak 450 W-ko orduko kontsumoa duela eta elektrizitatearen batez besteko kostua 0.19 €/kWh-koa dela aintzat hartuz, zerbitzariaren erabilerak sortutako elektrizitate-kostuak aurkezten dira jarraian. Erabili diren ordenagailu eta FPGA plakaren elektrizitate kostuak ez dira kontuan hartu, gastu hauek gailu hauen kontsumo baxua dela eta baztertu baitira.

IZENA	ORDU KOPURUA	POTENTZIA	KOSTUA	GUZTIRA
Zerbitzaria	2.280 h	450 Wh	0,19 €/kWh	194,94 €
			Guztira	**194,94 €**

9. Taula: Elektrizitate kostuen kalkulua

Gastu hauetaz gain ez dagoenez azpikontratazioak bezalako bestelako gasturik, proiektu honen garapenaren gastu totala kalkulatuko da atal honetako gastu guztiekin. Ondorengo taulan gastu guzti hauek biltzen dira.

KONTZEPTUA	GUZTIRA
Barne-orduak	32.760 €
Amortizazioak	334,74 €
Elektrizitatea	194,94 €
Guztira:	**33.289,68 €**

10. Taula: Kostuen kalkulu totala

Hortaz, barne orduak, amortizazio kostuak eta elektrizitate gastuak kontuan hartuz, proiektuak guztira 33.289,68 €-ko gastua izan duela kalkulatzen da.

11. Ondorioak

11.1 Proiektuari estuki lotutako ondorioak

Proiektu honen helburu nagusia RISC-V-n oinarritutako SoC baten diseinua gauzatzea izan da, diseinu honen funtzionamendu egokia FPGA bidez ziurtatu nahi zela. Proiektuaren amaieran, gauzatu nahi zen SoC diseinua lortu da, honen luzapena errazten duen AXI Bus bat gehitu dela gehitu nahi diren IP gehigarrietan pentsatuz.

Gainera, amaierako emaitza diseinu fidagarri bat dela egiaztatu da, SoC-aren funtzionaltasuna frogatu baita hainbat alderdietatik; bai SoC-aren simulazio bidez, bai FPGAn hedatutako diseinuan softwarearen hedapenaren bidez. Honela, APERT ikerkuntza-taldearen xedeentzat interesgarria den diseinu bat lortu da, ASIC baten diseinurako erabiltzeko prest dagoena.

Helburu nagusi honetaz gain, SoC-aren sorreraren prozesuan bestelako helburu batzuk lortu dira ere. Alde batetik, ISA hau sakonki aztertu da, honen ezaugarri eta modulu nagusiak guztiz ulertu direla.

RISC-V-ren modulu ezberdinak identifikatzeak asko erraztu du diseinuak zein modulu inplementatu behar dituen aukeratzea. Sistemak mikrokontrolagailu izaera izan behar zuela kontuan hartuta, erraza izan da modulu egokien identifikazioa.

Amaierako SoC-a lortzeko, kode-irekiko RISC-V prozesagailuen eskaintza sakonki aztertu da, aukera zabal honetatik proiektuaren helburuetara hobekien egokitzen den diseinua aukeratu dela. Gainera, hartutako erabakia diseinuaren fidagarritasunean, dokumentazioaren osotasunean edota software-babesean bezalako irizpideetan oinarrituta egin da, alderdi garrantzitsuenak ebaluatu direla. Aukera ezberdinen azalera bezalako alderdiak aztertzeko EDA tresnak erabili dira ere.

Ondoren, aukeratutako prozesagailu honetan oinarritutako diseinu sinple bat egin da, hau FPGA gailu batean sintetizatu dela. Oinarri diseinu hau ondo frogatu da, simulazioak eginez eta FPGA inplementazioetan *test* programa ezberdinak exekutatuz. FPGAra eramateko tresnen laguntzaz, diseinuaren azalera, erabilitako baliabideak eta denbora-analisiak egin dira, egokitasuna bermatzeko.

Oinarri diseinu sendo bat lortu ondoren, honen luzapena gauzatu da, IP bloke berriak modu erraz eta fidagarrian gehitzea ahalbidetzen duen AXI bus bat gehitu dela sistemara. Busaren funtzionamendua zuzena dela bermatu da ere, berriro ere simulazio eta FPGA inplementazioen bidez. AXI busean ere SPI Master periferiko bat gehitu da, hau kontrolatzeko beharrezkoak diren software liburutegiak sortu direla.

Azkenik, SoC-a bere kabuz abiatzea ahalbidetzen duen hardware mekanismo bat inplementatu da, *on-chip* memoria berri bat gehituz, boot ROMa hain zuzen ere. ROMaren eduki gisa, abiatze firmware bat diseinatu da, kanpoko memoria batetik irakurritako programa bat sistemaren *on-chip* memorietara kargatzen duenak, hori dena SPI Master periferikoaren erabilera eginez. Honela, SoC-a bere kabuz abiatzea lortzeaz gain, sisteman programa berriak modu erraz batean kargatzea ahalbidetzen duen azpisistema sortzea lortu da, ASIC batera eramateko egokia den diseinu bat lortuz.

11.2 Bestelako ondorioak

Bestalde, proiektua garatu den heinean, honetaz haratago doazen hainbat ondorio atera dira lan honetatik, gehienak RISC-V-rekin lotuak daudela.

11.2.1 RISC-V-ren egoeraren inguruan

Proiektu honetan egindako alternatiben analisiarekin RISC-V ekosistema irekia aztertu da, ISA honek etorkizun argia duela ondorioztatu dela. RISC-V-k prozesagailuen industriak dituen arazo garrantzitsu batzuk konpontzeko potentziala duela ikusi da, hau izan daitekeela bere adopzio industrialerako arrazoi sendoena. Hori dela eta, RISC-V ISAk erdieroaleen industrian eragin gero eta handiagoa izango duela aurreikusi daitekeela ondorioztatu da ere.

Bestalde, prozesagailuentzako hardware irekia oraindik berria den kontzeptu bat da eta merkatu honen arrisku handiena fragmentazioa dela ikusi da; teknologia bultzatzen duten erakundeek ez badute norabide berean aurrera egiten, ISAren erabilera arriskuan jarri daiteke, bestelako hardware irekiaren aldeko iniziatiba askotan gertatu izan den bezala.

Hala ere, RISC-V Fundazioaren sorrerarekin, ISAren erabilera modu efektibo batean bultzatzen ari dela egiaztatu da, alde batetik erdieroaleen industriako enpresa garrantzitsuek Fundazioa bultzatu dutelako eta bestetik industria-mailako inplementazioak gero eta maizago ikusi ahal direlako enpresa bultzatzaile hauen eskutik.

Enpresa hauetatik kanpo, RISC-V ISA irekia izateak, OpenHW Group edota LowRISC bezalako irabazi-asmorik gabeko erakundeen sorrera ekarri duela ikusi da. Erakunde hauek orain arte beste inork egin ez duen zerbait egiten dute: Sakonki frogatutako prozesagailuen diseinu irekiak edonori eskura ipintzea. Hau oso berritzailea eta azpimarragarria den zerbait dela ondorioztatu da, RISC-V-ren erabilera izugarri sustatzen baitu, ISAren biziraupenera ekarpen handi bat eginez.

11.2.2 RISC-V Irakaskuntzan

Proiektuaren garapenean ere, bestelako ondorio bat atera da. ISA honek prozesagailu baten egitura eta funtzionatzeko era modu oso argi eta sinple batean ikustarazten duela ikusi da, bere erabilera Konputazio Ingeniaritza irakasteko aukera ona izan daitekeela ondorioztatu dela ere.

Lan honetan lortutako diseinua eta honetara iristeko aztertu diren alternatiben azterketak Teknologia Elektronikoko Departamentuarentzat baliagarriak izan daitezkeela uste da, konputazioaren oinarriak eta diseinu digitala irakasteko material interesgarria izango delakoan.

11.2.3 APERT Taldearentzako onurak

Amaitzeko, ISA berritzaile honetan oinarritutako SoC baten sorrera APERT taldearentzat oso baliagarria izan daitekeela ondorioztatu da, etorkizun argia duen teknologia honekin lehenengo pausu bat izango litzakeelako taldearentzat. Gainera, teknologia honetan oinarritutako beste proiektu batzuen hastapena bultza dezake, taldeari prozesagailuen diseinuaren munduan jarraitzeko atea irekiz.

12. Bibliografia

[1] Intel Corporation, *Intel(R) 64 and IA-32 Architectures Software Developer's Manual, Combined Volumes*, no. March. 2023.

[2] S. Rodgers and R. A. Uhlig, "x86: Approaching 40 years and still going strong," *Intel newsroom*, 2017.

[3] S. Greengard, "Will RISC-V revolutionize computing?," *Commun. ACM*, vol. 63, no. 5, pp. 30–32, 2020, doi: 10.1145/3386377.

[4] "riscv-android-src/riscv-android." https://github.com/riscv-android-src/riscv-android (accessed Sep. 18, 2023).

[5] A. Waterman and K. Asanovic, *The RISC-V Instruction Set Manual Volume I: Unprivileged ISA - Document Version 20191213*, vol. I. 2019.

[6] D. Patterson and A. Waterman, *The RISC-V Reader: An Open Architecture Atlas*. 2017.

[7] RISC-V Foundation, "RISC-V Cryptography Extensions Volume I," vol. I, 2022.

[8] A. Waterman, K. Asanović, and J. Hauser, *The RISC-V Instruction Set Manual, Volume II: Privileged Architecture Version 20211203*, vol. II. RISC-V Foundation, 2021.

[9] Espressif Systems, "ESP32-C3 Technical Reference Manual," 2022.

[10] "westerndigitalcorporation/swerv_eh1: A directory of Western Digital's RISC-V SweRV Cores." https://github.com/westerndigitalcorporation/swerv_eh1/tree/master (accessed Sep. 18, 2023).

[11] C. Celio, P. F. Chiu, K. Asanović, B. Nikolić, and D. Patterson, "BOOM: An Open-Source Out-of-Order Processor with Resilient Low-Voltage Operation in 28-nm CMOS," *IEEE Micro*, vol. 39, no. 2, pp. 52–60, 2019, doi: 10.1109/MM.2019.2897782.

[12] J. Zhao, B. Korpan, A. Gonzalez, and K. Asanovic, "SonicBOOM: The 3rd Generation Berkeley Out-of-Order Machine," *Carrv*, 2020.

[13] K. Asanovic *et al.*, "The Rocket Chip Generator," *EECS Dep. Univ. California, Berkeley, Tech. Rep.*, no. UCB/EECS-2016-17, 2016.

[14] "Toolchain Projects - Home - RISC-V International." https://wiki.riscv.org/display/HOME/Toolchain+Projects#ToolchainProjects-Binutils (accessed Sep. 18, 2023).

[15] "Language Runtimes - Home - RISC-V International." https://wiki.riscv.org/display/HOME/Language+Runtimes#LanguageRuntimes-V8 (accessed Sep. 18, 2023).

[16] "YosysHQ/picorv32: PicoRV32 - A Size-Optimized RISC-V CPU." https://github.com/YosysHQ/picorv32 (accessed Sep. 18, 2023).

[17] "openhwgroup/cv32e40p: CV32E40P is an in-order 4-stage RISC-V RV32IMFCXpulp CPU based on RI5CY from PULP-Platform." https://github.com/openhwgroup/cv32e40p (accessed Sep. 18, 2023).

[18] "CORE-V-MCU User Manual — CORE-V MCU documentation." https://docs.openhwgroup.org/projects/core-v-mcu/ (accessed Sep. 18, 2023).

[19] "lowRISC/ibex: Ibex is a small 32 bit RISC-V CPU core, previously known as zero-riscy." https://github.com/lowRISC/ibex (accessed Sep. 18, 2023).

[20] N. Kossifidis, J. Xie, B. Huffman, A. Baum, G. Favor, and T. Kurd, *PMP Enhancements for memory access and execution prevention on Machine mode (Smepmp)*. RISC-V Foundation, 2021.

[21] "lowRISC/ibex-demo-system: A demo system for Ibex including debug support and some peripherals." https://github.com/lowRISC/ibex-demo-system (accessed Sep. 18, 2023).

[22] "Documentation | OpenTitan." https://opentitan.org/documentation/index.html (accessed Sep. 18, 2023).

[23] "pulp-platform/pulpino: An open-source microcontroller system based on RISC-V." https://github.com/pulp-platform/pulpino (accessed Sep. 18, 2023).

[24] R. Weigand and F. Siegle, "Fast Forward for RISC-V in Space: The Coyote IP Core," 2014. [Online]. Available: http://microelectronics.esa.int/riscv/rvws2022/presentations/01-RVWS-WeigandSiegle-2022-12-14.pdf.

[25] Silicon Labs, "OBI Bus Specification," p. 28, 2020.

[26] "Link Options (Using the GNU Compiler Collection (GCC))." https://gcc.gnu.org/onlinedocs/gcc/Link-Options.html (accessed Sep. 18, 2023).

[27] "RISC-V Options (Using the GNU Compiler Collection (GCC))." https://gcc.gnu.org/onlinedocs/gcc/RISC-V-Options.html (accessed Sep. 18, 2023).

13. Eranskinak

I. Eranskina: RV32I eta RV32M moduluen inguruko azalpen gehigarriak

Dokumentuan aipatu diren luzapen nagusiek biltzen dituzten funtzionaltasunak argitzeko asmoz, eranskin honetan hauen aginduak deskribatzen dira gainetik, agindu mota bakoitzaren deskodetzea nola egiten den ez dela azaltzen.

Diseinuan IMC luzapenak erabili direnez, eranskin honetan luzapen hauek azalduko dira, sortutako sistemak daraman prozesagailuak egin ditzakeen operazioak erakusteko asmoarekin. C luzapena ez da gehitu, honek instrukzioen kodifikazio eskema bat definitzen baitu.

Aginduak aurkeztu baino lehen, luzapen guztietarako orokorra den alderdi bat azaldu behar da. Luzapen bakoitzean, aginduen formatu mota ezberdinak aurkitu daitezke, bakoitzak aginduen eremu jakin batzuk definitzen dituela. Eremu hauek gordetzen duten informazioa ezberdina izan daiteke, nagusiki ondorengo edukia dutela eremuek:

- ***Operation code* edo *Opcode*:** Aginduen multzoak ezberdintzeko eremua
- ***Function* edo *funct*:** Opcode bereko agindu ezberdinak ezberdintzeko eremua
- ***Immediate value* edo *Imm*:** Berehalako balio baten atal bat adierazten duen eremua
- ***Source Register* edo *rs1/rs2*:** Iturri erregistroak adierazteko eremua. Gauzatu nahi den eragiketa erregistro hauen edukiaren balioekin egingo da
- ***Destination Register* edo *rd*:** Helburu erregistroa adierazteko eremua. Gauzatu nahi den eragiketaren emaitza bertan gordeko da
- ***Shift amount* edo *shamt*:** Desplazamendu eragiketetan, aginduak adierazten duen norabidean desplazatu beharreko bit kopurua
- ***PC Relative Address* edo *pcrel*:** Jauzi-aginduetan iritsi nahi den helbidera jauzi egiteko PCari gehitu nahi zaion balioa

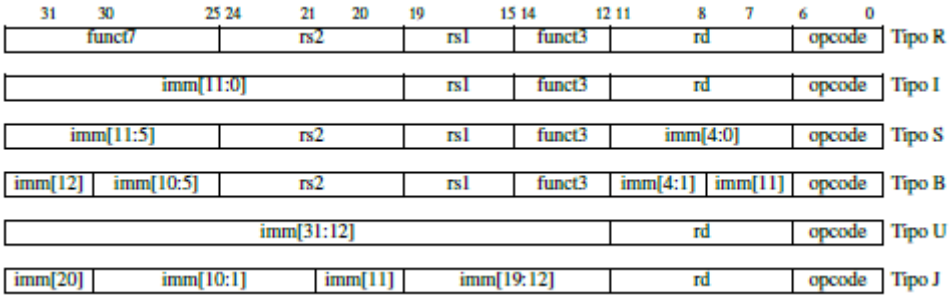
Hau azalduta, RV32I eta RV32M luzapenen xehetasunak aurkezten dira jarraian.

RV32I

I edo *integer* oinarrizko ISA, zenbaki osoen konputaziora eta prozesagailu baten oinarrizko atazetara zuzenduta dago. ISA honetan, instrukzio guztiak 32 bitekoak dira, sei formatu ezberdin definitzen dituela, gainontzeko RISC-V moduluetan erabiltzen direnak ere. Hona hemen definitutako formatuak:

- R motakoak: Erregistroen arteko operazioak egiteko.
- I motakoak: Berehalako balio motzak eta memoriatik *load*-ak egiteko.
- S motakoak: Memorian *store* eragiketak egiteko.
- B motakoak: *Branch*-ak gauzatzeko.
- U motakoak: Berehalako balio luzeen kudeaketarako.
- J motakoak: Baldintza gabeko jauzientzat.

Formatu bakoitzak eremu ezberdinak definitzen ditu, deskodetzeko era ezberdin bat izateaz gain. Hona hemen formatu bakoitzaren 32 biten egitura, bere eremuekin.



9. Irudia: RISC-V aginduen formatu ezberdinen egitura

Formatu mota jakin bat jarraitzen duten agindu guztiek eremu berak dituzte. Formatu bera jarraitzen duten agindu ezberdinak eta aginduak orokorrean, *opcode* eremuaren bidez ezberdintzen dira, *opcode* bakoitzak helburu bereko instrukzio multzo bat irudikatzen duela. Multzo honen barruan, agindu ezberdinak daude ere, *funct* eremuaren bidez ezberdintzen direnak.

Esate baterako, memoriatik balio bat kargatzeko aginduak *load* motako aginduak dira, haien formatu mota I dela eta haien *opcodea* 0000011 dela. Hala ere, *load* aginduen barnean byte bat, 16 bit edo 32 bit kargatzeko aginduak daude (*lb, lh, lw* dela agindu hauen izena hurrenez hurren). *Opcode* bereko instrukzio ezberdin hauek *funct* eremuaren bidez ezberdintzen dira. Hurrengo irudian agindu hauen formatua ikusi daiteke.

31 25	24 20	19 15	14 12	11 7	6 0	
imm[11:0]		rs1	000	rd	0000011	lb (I formatua)
imm[11:0]		rs1	001	rd	0000011	lb (I formatua)
imm[11:0]		rs1	010	rd	0000011	lb (I formatua)

10. Irudia: *load* instrukzioen formatu zehatza

Eragiketa aritmetiko-logikoetarako aginduak

Operazio hauetan, orokorrean agindu bakoitza aldaera bitan eskaintzen da: Bi erregistrok gordetzen dituzten balioen arteko operazioa (*add,* adibidez*)* edo erregistro batek eta berehalako balio baten arteko operazioa (*addi,* adibidez). 'i' letrak berehalako balio edo *immediate value*ari egiten dio erreferentzia.

Jarraian RV32I-n eragiketa aritmetiko-logikoak gauzatzeko eskaintzen dituen aginduak aurkezten dira.

Operazioa	**Aginduaren RISC-V Izena**	
	Erregistro biren artean	**Erregistro eta berehalako balio baten artean**
Gehiketa	*add (rd, rs1, rs2)*	*addi (rd, rs1, imm)*
Kenketa	*sub (rd, rs1, rs2)*	*subi (rd, rs1, imm)*
And	*and (rd, rs1, rs2)*	*andi (rd, rs1, imm)*
Or	*or (rd, rs1, rs2)*	*ori (rd, rs1, rs2)*
Xor	*xor (rd, rs1, rs2)*	*xori (rd, rs1, rs2)*
Ezkerrera desplazamendua	*sll (rd, rs1, rs2)*	*slli (rd, rs1, shamt_i)*
Eskuinera desplazamendu logikoa	*srl (rd, rs1, rs2)*	*srli (rd, rs1, shamt_i)*
Eskuinera desplazamendu aritmetikoa	*sra (rd, rs1, rs2)*	*srai (rd, rs1, shamt_i)*

11. Taula: Eragiketa aritmetiko-logikoetarako RV32I aginduak

Desplazamenduen kasuan, *rs2*-ren esangura gutxieneko bost bitek duten balioa izango da *rs1* desplazatzeko bit kopurua. Berehalako balioa erabiltzen denean aldiz, desplazamendua egiteko balio honen bit guztiak hartzen dira kontuan.

Konparaketetarako aginduak

Konparaketaren kasuan, erregistro bi edo berehalako balioa erabili nahi den arabera agindu bi definitzen dira ere (*rd, rs1, rs2*) edo (*rd, rs1, imm*) balioak jaso ditzakeela aginduak. Hala ere, konparaketetan *rs1*, *rs2* eta *imm* balioak zeinurik gabekoak direnerako agindu ezberdinak definitzen dira ere, hauek 'u' letra daramatela amaieran *unsigned*i erreferentzia eginez.

Agindu hauek bakoitzak ebaluatzen duen baldintza ebaluatzen dute, eta baldintza betetzekotan, '1' balioa gordeko da *rd* helburu erregistroan, bestela '0'. Adibidez, *set less than i* (*slti rd, rs1, XX)* aginduan *rs1*-en edukia *xx* (berehalako balio edo erregistro barten edukia izan daitekeena) baino txikiagoa bada, '1' bat gordeko da *rd*n.

Operazioa	**Aginduaren RISC-V Izena**			
	Erregistro biren artean		**Erregistro eta berehalako balio baten artean**	
	Zeinudun operandoak	**Zeinurik gabeko operandoak**	**Zeinudun operandoak**	**Zeinurik gabeko operandoak**
Set less than, txikiagoa bada	*slt (rd, rs1, rs2)*	*sltu (rd, rs1, rs2)*	*slti (rd, rs1, imm)*	*sltiu (rd, rs1, imm)*

12. Taula: Konparaketak egiteko RV32I aginduak

Datuak idatzi eta irakurtzeko aginduak

Agindu hauetan bi mota bereizi daitezke, memorian eta erregistroetan irakurketak eta idazketak ahalbidetzen dituzten aginduetan alegia.

Erregistroen kasuan bi agindu aurkitu daitezke, biak U formatua duten agindu bakarrak direla. *Load Upper Immediate* edo *lui* aginduak *rd* erregistro baten balioa *imm* berehalako balio batera ezartzen du. *Add Upper Immediate to PC* edo *auipc* aginduak aldiz, agindu horren helbidea, PCaren uneko balioa alegia, *imm* berehalako balioarekin batzen du, emaitza *rd* erregistroan gordez.

Operazioa	Aginduaren RISC-V Izena
Load Upper Immediate	*lui (rd, imm)*
Add Upper Immediate to PC	*auipc (rd, imm)*

13. Taula: Datuak erregistroetan idazteko RV32I aginduak

Memorian idazteko eta irakurtzeko, hiru agindu definitzen dira kasu bakoitzean. Agindu hauek hitz bat (32 bit), erdi-hitz bat (16 bit) edo byte bat (8 bit) gordetzeko aukera ematen dute. Kasu hiruetan, *rs1* erregistroaren edukian eta *imm* berehalako balio baten gehiketa egiten da memoriako irakurketa edo idazketa helbidea kalkulatzeko.

Idazketaren kasuan, *rs2* erregistroaren 32 biteko edukia idazten da kalkulatutako memoriaren helbide horretan, eta irakurketaren kasuan, *rd* erregistroan idazten da memoriatik irakurritakoa.

Idazketa byte bat edo erdi-hitz bat bada, *rs2*-ren esangura gutxieneko 8 edo 16 bitak hartuko dira, hurrenez hurren. Irakurketaren kasuan, zeinuan luzatuko da irakurritakoa.

Bestalde, zeinurik gabeko memoriako byte edo erdi-hitzen irakurketetarako beste agindu bat definitzen da, kasu honetan zeinuan luzatu beharrean 0-z luzatzen dela datua (*Zero extended*).

Zeinua	Operazioa					
	Memoriatik irakurketa			Memoriatik idazketa		
	Byte bat	Hitz-erdi bat	Hitz bat	Byte bat	Hitz-erdi bat	Hitz bat
Zeinuduna	*lb rd,imm(rs1)*	*lh rd,imm(rs1)*	*lw rd,imm(rs1)*	*sb rs2,imm(rs1)*	*sh rs2,imm(rs1)*	*sw rs2,imm(rs1)*
Zeinurik gabekoa	*lbu rd,imm(rs1)*	*lhu rd,imm(rs1)*	-	-	-	-

14. Taula: Datuak memoriatik irakurri edo memorian idazteko RV32I aginduak

Jauziak egiteko aginduak

Instrukzio hauen bidez, programa baten instrukzioen artean jauzi egin daiteke, exekuzioa guztiz sekuentziala izan ez dadin. Agindu hauek goi-mailako kodeetan ezinbestekoak diren funtzioen edota begizten exekuzioa egiten dute posible.

Jauzien kasuan ere, bi agindu mota ezberdin aurkitu daitezke: Baldintza-gabeko eta baldintzapeko jauziak. Bigarrengoek *slt* bezalako aginduak ekiditen dituzte, baldintzapeko jauzien operazioak atomiko bihurtuz.

Instrukzio hauek PCarekiko erlatiboak diren helbideak erabiltzen dituzte jauzi egiteko, hau da, *pcrel* helbideak ez du nora jauzi egin adierazten, baizik eta PCari zer gehitu behar zaion jauzi egin nahi den helbidera iristeko.

Baldintza gabeko jauzietan agindu bi bereizten dira. Sinpleena *jump and link* edo *jal* agindua da. Agindu honek *rd* erregistro batean *jal* agindu horren hurrengo agindua gordetzen du (Jauzitik itzuli nahi bada), PCari *pcrel* balioa gehitzen diola ere bai nahi den jauzia betetzeko. *Jump and link register* edo *jalr* aginduak aldiz, *rs1* erregistroaren edukia eta *imm* berehalako balio bat batzen ditu, jauzia eragiketaren emaitza horrek ematen duen helbidera izango dela. *Jal* aginduak bezala, agindu honek *rd* erregistro batean hurrengo agindua gordetzen du.

Baldintzapeko jauzietarako aginduen kasuan, funtzionamendu orokorra berdina da denetan: *rs1* eta *rs2* erregistro batzuk alderatzen dira, agindu bakoitzak ebaluatzen duen baldintza betetzekotan, *pcrel* balio bat gehitzen zaiola PC erregistroari. Mota honetako agindu guztiek B formatua jarraitzen dute.

Jarraian aurkezten den taulan aipatu berri diren aginduen RISC-V formatua erakusten da, baldintzapeko jauzi guztiak biltzen direla.

Operazioa	Aginduaren RISC-V Izena
Jump and link	*jal (rd, pcrel)*
Jump and link register	*jalr (rd, imm, rs1)*
***Branch if equal,* jauzi berdina bada**	*beq (rs1, rs2, pcrel)*
***Branch if greater or equal,* jauzi handiagoa edo berdina bada**	*bge (rs1, rs2, pcrel)*
***Branch if greater or equal unsigned,* jauzi handiagoa edo berdina bada, *rs1* eta *rs2* zeinurik gabekoak**	*bgeu (rs1, rs2, pcrel)*
***Branch if lower than,* jauzi txikiagoa bada**	*blt (rs1, rs2, pcrel)*
***Branch if lower than unsigned,* jauzi txikiagoa bada, *rs1* eta *rs2* zeinurik gabekoak**	*bltu (rs1, rs2, pcrel)*
***Branch not equal,* jauzi ezberdinak badira**	*bne (rs1, rs2, pcrel)*

15. Taula: Jauziak egiteko RV32I aginduak

Azkenik, aipatutako agindu guztiak biltzen dituen *opcodeen* mapa erakusten da hurrengo irudian. Bertan, agindu bakoitzaren eremuen balioak ikusi daitezke, eskuineko zutabean agindu bakoitzaren formatua ikusi daitekeela ere. ingurune-deiak egiteko *fence, ecall* eta *ebreak* aginduak, edota *csr* erregistroekin lan egiteko aginduak ikusi daitezke ere, sinplifikatzeko asmoarekin eranskin honetan azaldu ez direnak.

31 25	24 20	19 15	14 12	11 7	6 0	
imm[31:12]				rd	0110111	U lui
imm[31:12]				rd	0010111	U auipc
imm[20|10:1|11|19:12]				rd	1101111	J jal
imm[11:0]		rs1	000	rd	1100111	I jalr
imm[12|10:5]	rs2	rs1	000	imm[4:1|11]	1100011	B beq
imm[12|10:5]	rs2	rs1	001	imm[4:1|11]	1100011	B bne
imm[12|10:5]	rs2	rs1	100	imm[4:1|11]	1100011	B blt
imm[12|10:5]	rs2	rs1	101	imm[4:1|11]	1100011	B bge
imm[12|10:5]	rs2	rs1	110	imm[4:1|11]	1100011	B bltu
imm[12|10:5]	rs2	rs1	111	imm[4:1|11]	1100011	B bgeu
imm[11:0]		rs1	000	rd	0000011	I lb
imm[11:0]		rs1	001	rd	0000011	I lh
imm[11:0]		rs1	010	rd	0000011	I lw
imm[11:0]		rs1	100	rd	0000011	I lbu
imm[11:0]		rs1	101	rd	0000011	I lhu
imm[11:5]	rs2	rs1	000	imm[4:0]	0100011	S sb
imm[11:5]	rs2	rs1	001	imm[4:0]	0100011	S sh
imm[11:5]	rs2	rs1	010	imm[4:0]	0100011	S sw
imm[11:0]		rs1	000	rd	0010011	I addi
imm[11:0]		rs1	010	rd	0010011	I slti
imm[11:0]		rs1	011	rd	0010011	I sltiu
imm[11:0]		rs1	100	rd	0010011	I xori
imm[11:0]		rs1	110	rd	0010011	I ori
imm[11:0]		rs1	111	rd	0010011	I andi
0000000	shamt	rs1	001	rd	0010011	I slli
0000000	shamt	rs1	101	rd	0010011	I srli
0100000	shamt	rs1	101	rd	0010011	I srai
0000000	rs2	rs1	000	rd	0110011	R add
0100000	rs2	rs1	000	rd	0110011	R sub
0000000	rs2	rs1	001	rd	0110011	R sll
0000000	rs2	rs1	010	rd	0110011	R slt
0000000	rs2	rs1	011	rd	0110011	R sltu
0000000	rs2	rs1	100	rd	0110011	R xor
0000000	rs2	rs1	101	rd	0110011	R srl
0100000	rs2	rs1	101	rd	0110011	R sra
0000000	rs2	rs1	110	rd	0110011	R or
0000000	rs2	rs1	111	rd	0110011	R and
0000 pred	succ	00000	000	00000	0001111	I fence
0000 0000	0000	00000	001	00000	0001111	I fence.i
000000000000		00000	000	00000	1110011	I ecall
000000000001		00000	000	00000	1110011	I ebreak
csr		rs1	001	rd	1110011	I csrrw
csr		rs1	010	rd	1110011	I csrrs
csr		rs1	011	rd	1110011	I csrrc
csr		zimm	101	rd	1110011	I csrrwi
csr		zimm	110	rd	1110011	I csrrsi
csr		zimm	111	rd	1110011	I csrrci

11. Irudia: RV32I Oinarrizko moduluaren instrukzio guztien formatu zehatza

RV32I Erregistroak

Oinarrizko ISA honetan, 32 erregistro dituzte coreek. Bestelako ISAekin alderatuz, erregistro kopuru handia da, diseinuen area nabariki handiagotzen duela ezaugarri honek. Hala ere, erregistro asko izateak, operazioak asko errazten ditu, abantaila garrantzitsuak emanez. Abantaila nagusia ISAren sinplifikazio izugarria da, bestelako abantaila bat ataza jakin batzuk agindu gutxiagorekin betetzeko aukera dela.

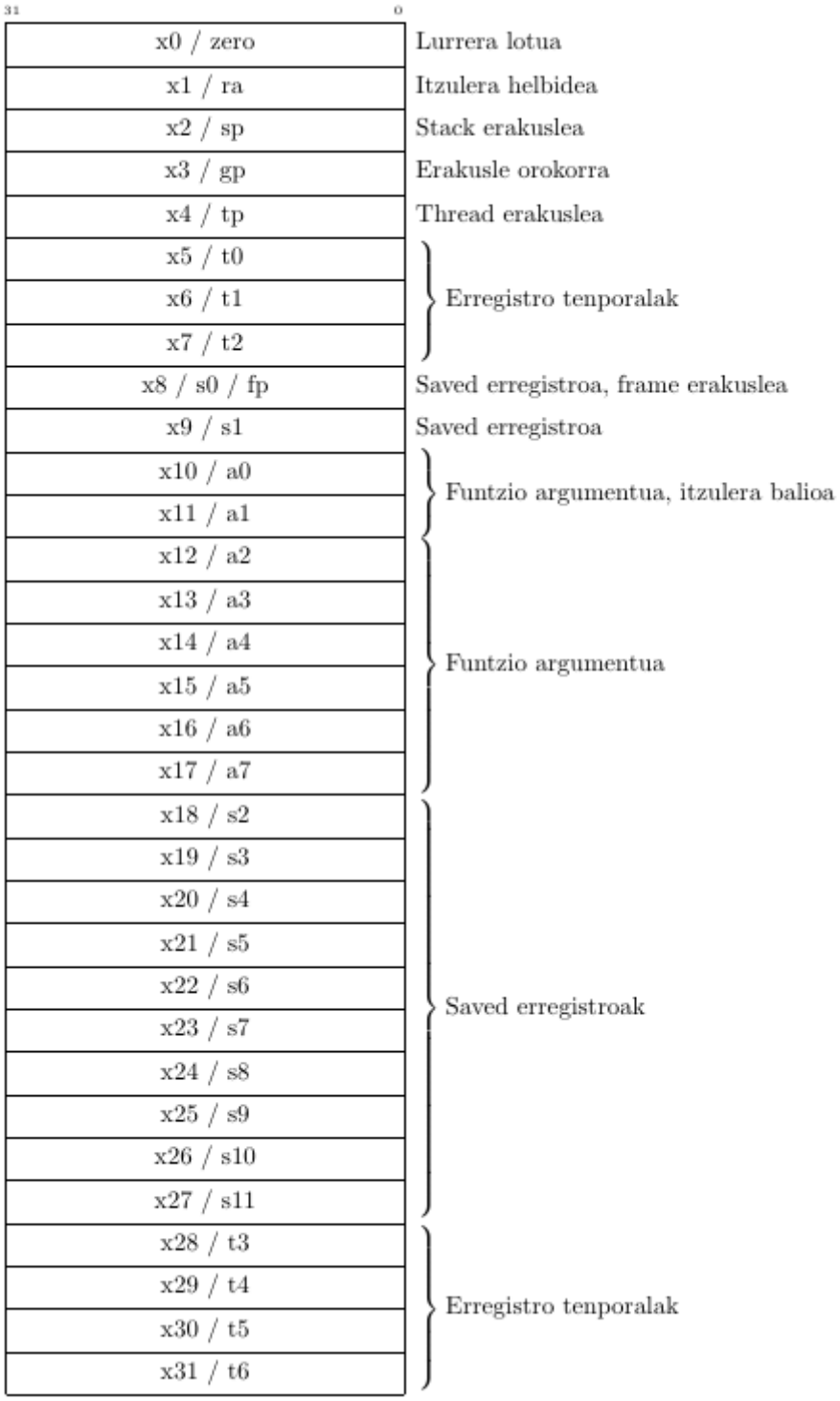
Erregistroek x0-x31 formatua jarraitzen dute, baina RISC-V-ren ABIak izen esanguratsuagoak ematen dizkie hauei. PC erregistroa hauetatik kanpo dagoela aipatu beharra dago. Jarraian 32 erregistroen artean aurkitu daitezkeen mota ezberdinak azaltzen dira.

x0 edo zero erregistroa oso berezia da. Lurrera lotutako erregistro bat dela esaten da, erregistro honen irakurketa beti izango baita '0', bere idazketa galarazten dela. Erregistro honek asko sinplifikatzen du ISA, adibidez balioak baztertzeko metodo oso erosoa eskaintzen baitu: x0-n balio bat gordetzea balio hori baztertzea bezala da.

Bestalde, RV32I-n hainbeste erregistro izanda, balioak memorian gorde eta bertatik kargatzea ekidin nahi da ahal denean. Horretarako, erregistro tenporal eta *saved* motakoak ezberdintzen dira. Lehenengoen kasuan, funtzioetara dei bat egotekotan ez da bermatzen bertan gordetzen diren balioak erregistro hauetan mantenduko direnik, *saved* erregistroen kasuan hau bai bermatzen dela.

Azkenik, erregistro hauen artean ohikoak diren bestelako erregistro batzuk aurkitu daitezke, hala nola *stack*era erakuslea, erakusle globala edota funtzioen argumentuak edo itzulera balioak gordetzeko erregistroak.

Hurrengo irudian RV32I-ren erregistro guztiak erakusten dira, hauen formatu normala eta ABIan definitzen den izena adierazten dela, bakoitzaren funtzio laburbildua azaltzen dela ere.



12. Irudia: RV32I Oinarrizko moduluaren erregistroak

RV32M

ISA Honek biderkatze eta zatikatze operazioak gauzatzeko aginduak gehitzen ditu. Bere inplementazioak hardware gehigarria dakar, operazio hauek azkarrago betetzea ahalbidetzen duela.

32 biteko datuekin lan egiten duenez ISA honek, bit kopuru hori duten zenbakien arteko biderketak 64 biteko emaitza bat ematen du. Ondorioz, zenbait kasutan, biderketa bi pausutan egin beharko da, horretarako instrukzio ezberdinak erabili beharko direla. Hardwarea sinpletzea da bi pausutan egitearen arrazoia, dena pausu batean egiteak asko konplikatuko bailuke hardwarea.

Agindu guztiek *insn (rd, rs2, rs1)* formatua jarraitzen dute, parametro bezala helburu erregistroa eta biderkatu nahi diren datu biak dituzten jatorri erregistro bi adierazten direla. Kasu honetan beraz, ez dago berehalako balioen erabilera, hauek M ISAko agindua exekutatu aurretik aipatutako erregistro bietan kargatu behar direla.

Zatiketaren kasuan, *rs1 rs2*-rengatik zatitzen da, borobiltzea zerorantz egiten dela.

M ISAk definitzen dituen eragiketak ondokoak dira:

- *mul*: Datu biren biderketaren emaitzaren esangura gutxieneko 32 bitak lortzen ditu
- *mulh*: Datu biren biderketaren emaitzaren 32 bit esanguratsuenak lortzen ditu
- *mulhsu*: Datu biren biderketaren emaitzaren esangura gutxieneko 32 bitak lortzen ditu, datuetako bat zeinuduna eta bestea zeinurik gabekoa denean dela egokia bere erabilera
- *mulhu:* Datu biren biderketaren emaitzaren esangura gutxieneko 32 bitak lortzen ditu, datu biak zeinurik gabekoak direnean dela egokia bere erabilera
- *div: rs1* zati *rs2* gordetzen du *rd*n, zatigaiak biren osagaiean adierazita daudela hartzen duela kontuan
- *divu: rs1* zati *rs2* gordetzen du *rd*n, zatigaiak zeinurik gabekoak direla hartzen duela kontuan
- *rem: rs1* zati *rs2* operazioaren hondarra gordetzen du *rd*n, zatigaiak biren osagaian adierazita daudela hartzen duela kontuan
- *remu: rs1* zati *rs2* operazioaren hondarra gordetzen du *rd*n, zatigaiak zeinurik gabekoak direla hartzen duela kontuan

Jarraian, ISA honen instrukzioen kodetzea erakusten da. ISA honetako instrukzioek R formatua dute, non 3 bitetako *funct* eremu bakarrarekin RV32M osatzen duten zortzi instrukzioak bereizten dituen, hasierako 7 bitek beti '1' balioa dutela M motako instrukzioak RISC-V ISA osoan identifikatzeko.

31 25	24 20	19 15	14 12	11 7	6 0	
0000001	rs2	rs1	000	rd	0110011	mul (R formatua)
0000001	rs2	rs1	001	rd	0110011	mulh (R formatua)
0000001	rs2	rs1	010	rd	0110011	mulhsu (R formatua)
0000001	rs2	rs1	011	rd	0110011	mulhu (R formatua)
0000001	rs2	rs1	100	rd	0110011	div (R formatua)
0000001	rs2	rs1	101	rd	0110011	divu (R formatua)
0000001	rs2	rs1	110	rd	0110011	rem (R formatua)
0000001	rs2	rs1	111	rd	0110011	remu (R formatua)

13. Irudia: RV32M Moduluko instrukzio guztien formatu zehatza

II. Eranskina: Alternatiben inguruko azalpen gehigarriak

Eranskin honetan, “alternatiben analisia” atalean aurkeztutako aukera ezberdinak modu xeheago batean aurkeztu nahi dira, bakoitzaren egitura modu sakonago batean aurkezteko asmoarekin. Gainera, alternatibek eskaintzen dituzten froga sistemak sakonago azaldu nahi dira ere.

PicoRV32

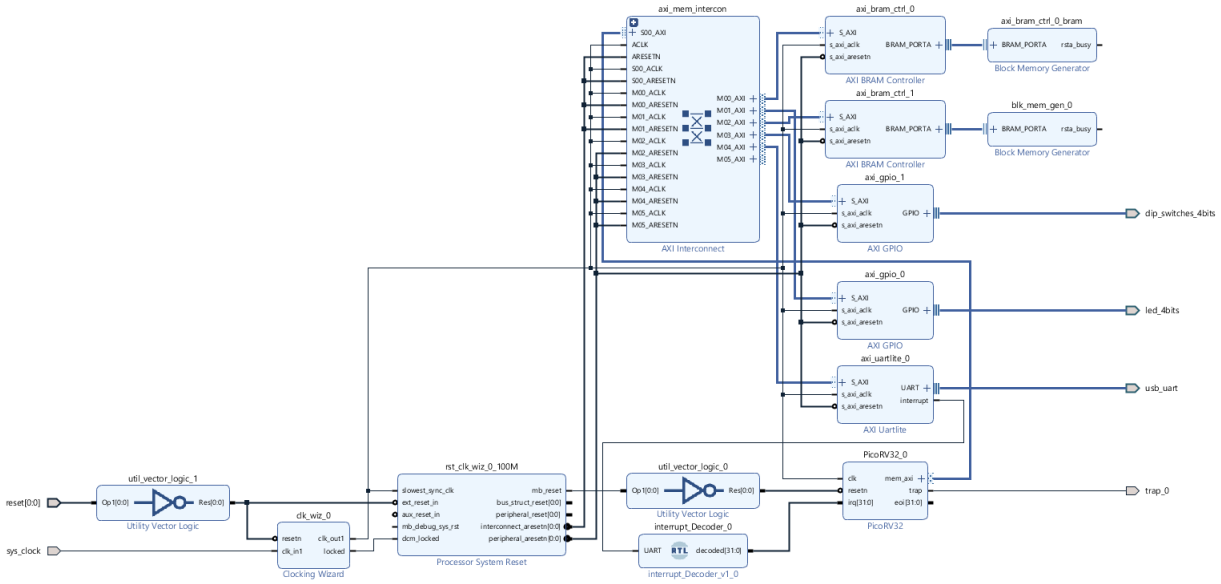
Core honen kasuan, soilik berau frogatzeko erabili den inplementazioa azalduko da, corearen egitura xehetasunez azaltzeko eskura dagoen informazioa urriegia baita.

Corea frogatzeko hardware diseinua

PicoRV32 proiektuak corearen interfaze natiboa darabilen PicoSoC sistema sinplea eskaintzen du corearen funtzionamendua erakusteko. Hala ere, ez da adibide sinple hori erabili, PicoRV32-ren AXI Lite Master interfazea frogatu nahi baitzen. Beraz, frogapena azkarrago egin ahal izateko, corea Xilinxen FPGA batera eraman da, SoC diseinu sinple bat sortuz. Diseinuan ondoko osagaiak gehitu dira, periferikoak Xilinxen AXI IPak erabiliz hautatu direla prozesua errazteko:

- PicoRV32 corea - AXI Lite Master
- UART periferikoa – AXI Lite Slave
- GPIO periferiko bi – AXI Lite Slave
- AXI Timer periferikoa – AXI Lite slave
- AXI Block RAM bi – AXI Lite slave
- Reset eta erloju sortzaileak
- IRQentzat dekodifikadore bat

Jarraian sortutako demo sistemaren eskema orokorra erakusten da, aipatutako osagai bakoitza bloke moduan ikusi daitekeela.



14. Irudia: PicoRV32 corearekin eta Xilinxen IPekin egindako SoC diseinu sinplea

Block RAM bat instrukzio memoria gisa erabili da, konpilatutako programak bertan kokatu direla. RAM hau praktikan ROM bat bezala erabili da, memoria exekutatu beharreko instrukzioak irakurtzeko erabili baita soilik. Beste block RAMa datu memoria gisa erabili da, bertan exekutatzen den programaren stacka etab. idatzi eta irakurtzeko. GPIO periferikoak froga moduan erabili dira.

AXI Timer eta UART periferikoak PicoRV32-ren etenekin lan nola egiten den frogatzeko gehitu dira, eta hauen kudeaketa deserosoa dela ondorioztatu da, proiektuak ez baitu hauek software aldetik kudeatzen goi mailako lengoaia baten bidez; gehiena mihiztatzaile lengoaiaz egin behar dela.

Gainera, IRQen kudeaketa hain sinplea dela eta, etenei buruz lortu daitekeen informazioa RISC-V estandarrak espezifikatzen duen modu pribilegiatuarekin lortzen denarekiko alderatuz oso mugatua da.

Gainera, aipatu den bezala, froga sisteman erabilitako osagai guztiak Vivadon eskuragarri dauden IPak dira. Hau arazo bat da, ASIC batera eraman nahi den diseinu batean IPen erabilera sahiestu nahi izan ohi da eta. Noski, askotan behar diren osagaien kode-irekiko inplementazioak aurkitu daitezke, baina proiektuarekiko independenteak izango direnez, bateragarritasun arazoak gerta daitezke.

Diseinua frogatzeko softwarea

PicoRV32-k ez duenez softwarea garatu eta sistema batean kargatzeko inongo *script* edo tresnarik, prozesu guztia eskuz egin behar izan da. Hori dela eta, GCC konpiladoreari beharrezko fitxategi guztiak adierazi behar izan zaizkio, hauek jarraian zerrendatzen direla. Hauek RISC-V prozesagailu bat 0tik abian jartzeko zer egin behar den erakusten dutelako azaldu dira, interesgarria izan daitekeelakoan.

- **Instrukzio pertsonalizatuak gehitu (custom_ops.S):** Mihiztatzaile lengoaian idatzitako fitxategi baten izena adieraziz, konpiladoreari mihiztatzaile lengoaian erabilitako funtzio jakin batzuk bit mailan nola itzuli behar diren definitzen da.

  Hau oso interesgarria da, RISC-V-k ematen duen askatasunaren erakusle baita. Jarraian erakusten den kode zatian, *custom_ops.S* fitxategiaren zati bat, ondo ikusi daiteke ezaugarri honen erabilera.

```
#define r_type_insn(_f7, _rs2, _rs1, _f3, _rd, _opc) \
.word (((_f7) << 25) | ((_rs2) << 20) | ((_rs1) << 15) | ((_f3) << 12) | ((_rd) << 7) | ((_opc) << 0))

#define picorv32_getq_insn(_rd, _qs) \
r_type_insn(0b0000000, 0, regnum_ ## _qs, 0b100, regnum_ ## _rd, 0b0001011)

#define picorv32_setq_insn(_qd, _rs) \
r_type_insn(0b0000001, 0, regnum_ ## _rs, 0b010, regnum_ ## _qd, 0b0001011)
```

15. Irudia: IRQei dagokien instrukzio pertsonalizatuak definitzeko mihiztatzaile lengoaian idatzitako kodea

Ikusi daitekeenez, lehenengo lerro bietan *r_type_insn* funtzio bat definitzen da. Bigarren lerroan, funtzio hau bit mailan nola definitu behar den adierazten da *.word* mihiztatzaile direktiba erabiliz, RISC-V agindu baten formatua definituz. Lerro hauen emaitza beraz, hurrengo irudian erakusten den formatuaren definizioa da.

Hurrengo lerroetan, funtzio biren izenak definitzen dira: *picorv32_getq_insn* eta *picorv32_setq_insn.* Funtzio hauen eginkizuna sortutako eten baten kodea lortzea da, PicoRV32-ren Q interrupzio erregistro bereziekin lan egitea ahalbidetzen dutela.

Eten bat egotekotan, etena identifikatzen duen kodea Q erregistro hauetan gordetzen da. Interrupzio-kode hau hardwareak sortzen du, bloke-diagraman ikusten den *interruptDecoder* blokeak bidaltzen diola coreari informazio hau honen *irq* portura.

Funtzio bakoitzeko, *r_type_insn-ri* egiten zaio dei, eremu bakoitzaren edukia adierazten dela. Ondorioz, lehen azaldutako kodeak hurrengo irudian erakusten diren aginduak sortu ditu.

31 25	24 20	19 15	14 12	11 7	6 0	
funct	0	rs1	funct	rd	opcode	Instrukzio pertsonalizatuen formatua
31 25	24 20	19 15	14 12	11 7	6 0	
0000000	0000	rs1	100	rd	0001011	getQ instrukzioa
0000001	0000	rs1	010	rd	0001011	setQ instrukzioa

16. Irudia: Pertsonalizatutako instrukzioen formatu zehatza

- **crt motako ftxategia (start.S):** Mihiztatzaile lengoaian idatzitako kode zati hau izango da prozesagailuak exekutatuko duen lehenengo atala. Hau oso ohikoa da sistema txertatuetan, eta sistema berrezartzeko erabili ohi da, kodeak ondoko atazak gauzatzen dituela.
    - Etenen kudeaketarako funtzioen deiak definitu, *interrupt handler* bat definitu alegia. Coreak eten bat jasotzen duenean, honen hardwareak exekuzioa kodearen *interrupt handler* atal honetara eramaten du jauzi batez.

      Interrupt handlerrak sortutako interrupzioari kasu egin baino lehen egin beharreko atazak betetzen ditu, etena gertatu baino lehen corearen erregistroen balioak memorian gordetzea adibidez, interrupzioa tratatu ondoren corearen aurreko egoera berrezarri ahal izateko.

      Kasu honetan, etenen kudeaketarako instrukzioak berriak definitu direnez, *interrupt handler*-ean *setQ* eta *getQ* funtzio berriak erabiliko dira, interrupzio kausak erregistro jakin batzuetara eramateko.

      Honela, Q erregistroek gordetzen duten “interrupzio kausa” balioa interrupzioa benetan kudeatzen duen eta C kodean definitutako funtzioari pasa ahalko zaio argumentu bezala. Azkenik, funtzio honetara egingo da jauzi.

    - Prozesagailuaren erregistroak 0-ra ipini. Hau da errealitaten coreak exekutatzen duen lehenengo zatia, *interrupt handlerra* eten bat gertatzean exekutatzen dela soilik.
    - Stack pointerraren balioa definitu, honek dagokion helbidea erakusten duela ziurtatzen duela.
    - Main funtziora jauzi egin.

- **Etenen kudeaketarako funtzioa (irq2.c):** C lengoaian idatzitako Kode hau da *interrupt handlerrak* bere amaieran deitzen duena. *Interrupt handlerrak* utzitako argumentuak hartuz daki C kode honek zein interrupzio gertatu den, horren arabera honi dagokion funtzioa exekutatuko dela. Azkenik, exekuzioa berrezartzen du.
- **Coreak exekutatu behar duen kodea (firmware.h eta firmware.c):** Kode hau izango da coreak exekutatu behar duena. Fitxategi bat baino gehiago gehitu daiteke, kasu honetan iturri-kode eta liburutegi fitxategi bana adierazi dela.
- **Linker scriptak (firmware.lds eta firmware.map):** Konpiladoreari sistemaren helbideratze-espazioaren inguruko informazioa ematen dieten fitxategiak.

GCC konpiladoreari informazio guzti hau ondoko komandoarekin adierazi zaio, ondoko informazioa pasa zaiola [26], [27]:

- **march=rv32im: Helburu coreak inplementatzen dituen RISC-V luzapenak eta arkitekturaren bit kopurua. Kasu honetan, 32 biteko IM luzapenak inplementatzen dituen corea.**
- **mabi=ilp32: Erabili nahi den ABIa.**
- **-Os: Prozesua modu optimoan egin (O) eta objektuen sinbolo ezabatu (s)**
- **ffreestanding: SE gabeko sistema bat dela adierazteko.**
- **nostdlib: Sistemen abiatzerako fitxategi estandarren erabilera ekiditeko.**
- **-o firmware.elf:** Konpilazioaren emaitza fitxategiaren izena adierazteko.
- **std=gnu99:** GCCren estandar zehatz bat aukeratzeko.
- **-Bstatic, -T, -Map:** Fitxategi horiek pasa *linkerrari,* eta hau modu estatikoan egin
- **Strip-debug:** Debug sinboloak ezabatu

Hona hemen exekutatutako kodea. Emaitza bezala *firmware.elf* fitxategia lortu da.

```
riscv32-unknown-elf-gcc -march=rv32im -mabi=ilp32 -Os -ffreestanding -nostdlib -o firmware.elf start.S custom_ops.S firmware4.c irq2.c firmware.h --std=gnu99 -Wl,-Bstatic,-T,firmware.lds,-Map,firmware.map,--strip-debug -lgcc
```

17. Irudia: Kodea konpilatzeko Bash komandoa

Ondorengo pausuak, lortutako elf fitxategia *bin* motako fitxategi bitarrera pasatzea izan da, *objcopy* tresnarekin lortu dena. Azkenik, *elf* fitxategiaren edukia ulertzeko *lst* motako testu fitxategi bat sortu da, kodearen aginduak mihiztatzaile lengoaian erakusten direla agindu bakoitzaren memoria-helbidearekin bat. Fitxategi bi hauek komando hauekin lortu dira.

```
riscv32-unknown-elf-objcopy -O binary firmware.elf firmware.bin
riscv32-unknown-elf-objdump -Mnumeric,no-aliases -dr firmware.elf > firmware3.lst
```

18. Irudia: Objektu fitxategia eta fitxategi bitarra lortzeko Bash komandoak

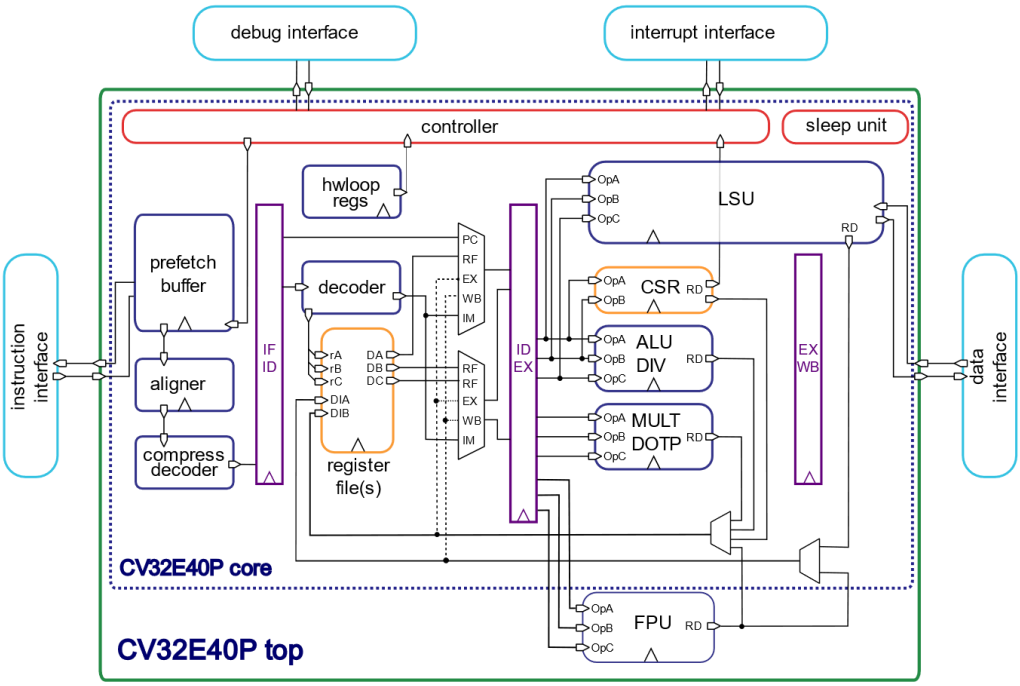
Bin motako fitxategia hardware diseinuaren memorietara eraman da, ondoren diseinua FPGA gailu batean erabiltzeko sintetizatu dela, emaitza egokiak lortuz.

CV32E40P

Jarraian CV32E40P corearen inguruko azalpenak emango dira. Kasu honetan, corearen egitura azalduko da lehenengo, kasu honetan ez dela froga-sistemarik azalduko proiektuan ez delako OpenHW Group taldeak proposatzen duen CORE-V-MCU SoC diseinua erabili honen konplexutasuna dela eta.

Corearen egitura

CV32E40P-k ohikoa den *pipeline* batetaz osatuta dago. Jakina denez, instrukzio-mailan paralelismoa ahalbidetzen duen egitura honek prozesagailuaren aprobetxamendua handiagotzen du, helburua honen atal ezberdinak uneoro lanean egotea dela. Kasu honetan, corearen egitura azaltzen duen bloke-diagraman ikusi daitekeenez, 4 etapako *pipeline* bat inplementatzen da. Jarraian aurkezten den irudian corearen bloke-diagrama ikusi daiteke.



19. Irudia: CV32E40P Corearen bloke-diagrama

Nahiz eta irudian esplizituki ez adierazi, etapa bakoitzaren osagaiak argi aurkitu daitezke, coreak inplementatzen dituen 4 etapak prozesagailu askotan oso ohikoak diren etapak direla. Hona hemen etapa bakoitza eta bere osagaiak.

- **Instrukzioaren irakurketa:** Instrukzioak memoriatik irakurtzeaz arduratzen den hardware-multzoa, instrukzioentzat buffer eta lerrokatzaile batetaz osatua. Aukeran konprimatutako instrukzioen pre-dekodetzailea gehitu daiteke etapa honetan, RISC-V-ren C instrukzioak I instrukzio bilakatzeko.

- **Instrukzioaren dekodetzea:** Dekodetzaileak irakurritako instrukzioak dekodetzen ditu eta prozesgailuaren erregistroetan beharrezko irakurketak gauzatzen ditu, saltoak dakartzaten instrukzioak hemen exekutatzen direla ere.

- **Instrukzioaren exkuzioa:** ALU, bidertzaile edota zatitzaile unitatearen erabileraz dekodetutako instrukzioak exekutatzen dira, emaitzak prozesagailuaren erregistroetan idazten direla ere. Exekuzioaren ondoren memoriara sarbidea bada, memoriako helbidearen sorrera etapa honetan egiten da ere, LSU edo *Load Store Unit*-aren erabileraz.

  CV32E40P F instrukzioak exekutatzeko konfiguratzen bada, FPUaren erabilera eta honekin komunikazioa etapa honetan egingo litzateke ere.

- **Emaitzaren idazketa:** Memoriatik irakurketak egotekotan, emaitzak erregistroetan idazten dira LSUaren bidez. FPUa erabiltzen bada, bere emaitzak erregistroetan etapa honetan idatziko dira.

Berriro ere, 4 etapen erabilera errendimendu hobea lortzeko asmoarekin inplementatu da, tamaina onargarri bat mantenduz. Etapa bakoitzaren gehikuntzak eragin handia du diseinuan, izan ere etapa bakoitza gehitzean, hauen hardwarearen sinpletasuna murriztu egiten da, lortu daitekeen fmax-a handiagotuz. Hala ere, area gehiago erabiltzen da eta sistemaren latentzia handiagotzen da, sistema konplexuagoa eginez ere. Bestalde, *pipeline* bat inplementatzean gerta daitezkeen arazo edo *hazard*ak ekiditeko, hardware gehigarria behar da ere.

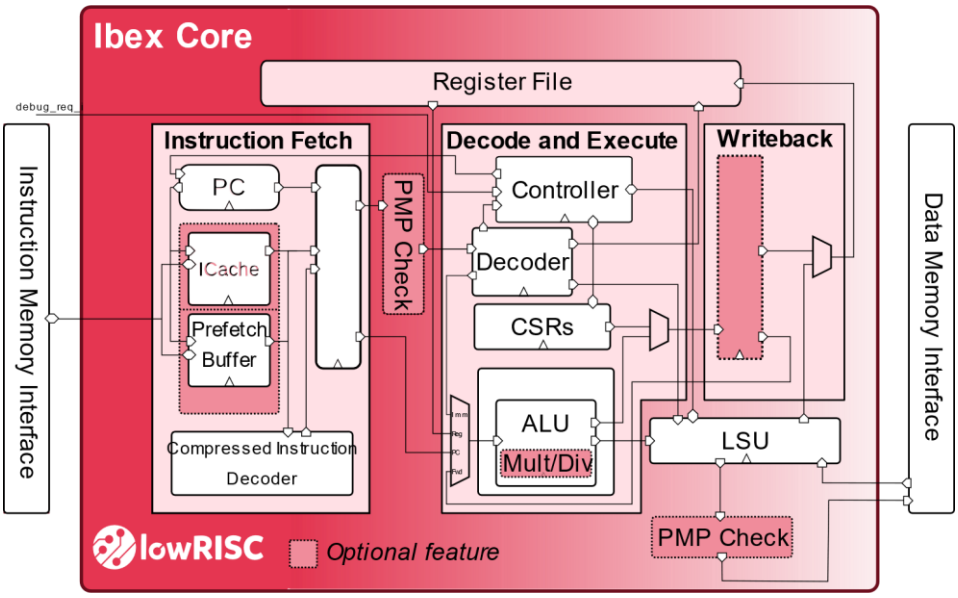
Prozesagailuaren memoria interfazeei dagokionez, interfaze bana erabiltzen da datu eta instrukzioen memoriak atzitzeko. Kasu honetan, interfaze hauek asko hedatu gabeko OBI (*Open Bus Interface*) estandarra jarraitzen dute. OBI AXI bezalako *request-grant* protokolo bat da, OBI sinpleagoa dela. Bestalde, prozesagailuak debug prozesuan laguntzeko interfaze bat du, RISC-V-ren kanpoko debug supporterako estandarra jarraitzen duenak. Azkenik, coreak pertsonalizatutako etenen interfaze bat du, bertatik ematen direla interrupzioekin elkarrekintzak.

Ibex

Ibexen coreari dagokionez, honen egitura eta bera erabiltzen duten *ibex-demo-system* eta *OpenTitan* erakutsiko dira.

Corearen egitura

Ondorengo irudian Ibex corearen bloke-diagrama erakusten da. Bertan hiru etapak ikusi daitezke, aukerazko osagaiak lerro etenekin adierazita daudela.



20. Irudia: Ibex corearen bloke-diagrama

Ikusi daitekeenez, hiru etapa nagusiak jarraian zerrendatzen direnak dira.

- **Instrukzioaren irakurketa (IF):** Memoriatik instrukzioak irakurtzen dira etapa honetan, irakurketa erloju ziklo batean egin daitekeela memoria sistemak ahal badu. Instrukzioak FIFO motako ilara baten gordetzen dira prefetch buffer elementuan, errendimendua hobetzeko. Konprimatutako instrukzioen kasuan, hauen deskonprimazio etapa honetan egiten da ere.

  Instrukzioentzako aukerazko cache baten erabilera eskaintzen da ere, errendimendua asko hobe dezakeen elementu gehigarria. Bestalde, diagramatik kanpo, errendimendua hobetzen duen *branch* prediktore bat inplementatu daiteke etapa honetan. Elementu hauek fase esperimentalean daude oraindik.

- **Instrukzioaren dekodetzea eta exekuzioa (ID/EX):** Aurreko etapan lortutako instrukzioa dekodetzen da eta jarraian hau exekutatu egiten da. Ataza biak etapa bakarrean izateak instrukzio batzuk erloju ziklo batean prozesatu daitezke, baina konplexuak diren instrukzioak ziklo bat baino gehiago iraun dezakete etapa honetan, etapa hau izozten.

  Instrukzioen dekodetzea kontrolagailu eta dekodetzaile batekin gauzatzen da. Kontrolagailuak prozesagailuaren exekuzio orokorra maneiatzen duen estatu-makina bat gordetzen du, dekodetzaileak instrukziotik informazioa ateratzen duela, horren arabera beharrezko seinaleak bidaltzen dituela beste blokeetara.

  Exekuzioari dagokionez, ALU unitate konbinazionalak zenbaki osoen operazioak edota kontrol helburuetarako konparazioak bezalako operazioak gauzatzen ditu. Biderkatzaile/Zatitzaile blokeak biderketa eta zatiketa operazioak gauzatzen ditu, biderkatzailea tamaina-errendimendu konpromisoarengan eragina duten hiru konfiguraziotan eskaintzen dela.

  Azkenik, CSR erregistroak aurkitu daitezke atal honetan, corearen egoera eta honen errendimendu metrikak gordetzen direla egitura honetan.

  Nahiz eta Pipelinetik kanpo kokatu, LSU unitateak etapa honekin du elkarrekintza. Bloke honek datuen memoriara gauzatzen ditu atzipenak, ID/EX etaparen agindupean. Memorian irakurketa edo idazketa batek ID/EX etapa blokeatzen du gutxienez ziklo batez, memoriaren erantzun-denboraren arabera.

- **Emaitzaren idazketa:** Aukerazko etapa honek memoriatik egindako irakurketak erregistroetan idazten ditu zuzenean. Nahiz eta informazio gutxi egon, Ibexen dokumentazioaren arabera errendimendua asko hobetu dezake.

Ikusi daitekeenez, coreak tamaina oso txiki bat lortzeko aukera ematen du, bi etapako *pipeline* sinple bat inplementatzeko aukera emanez, baina sistema txertatuetarako egokiak diren luzapenak erabiltzeko aukera emanez. Gainera, tamaina pixka bat handiagotzearen truke, hainbat alderdiren errendimendua hobetzeko aukera dago ere, nahiz eta ezaugarri hauetako asko oraindik garapen-prozesuan egon.

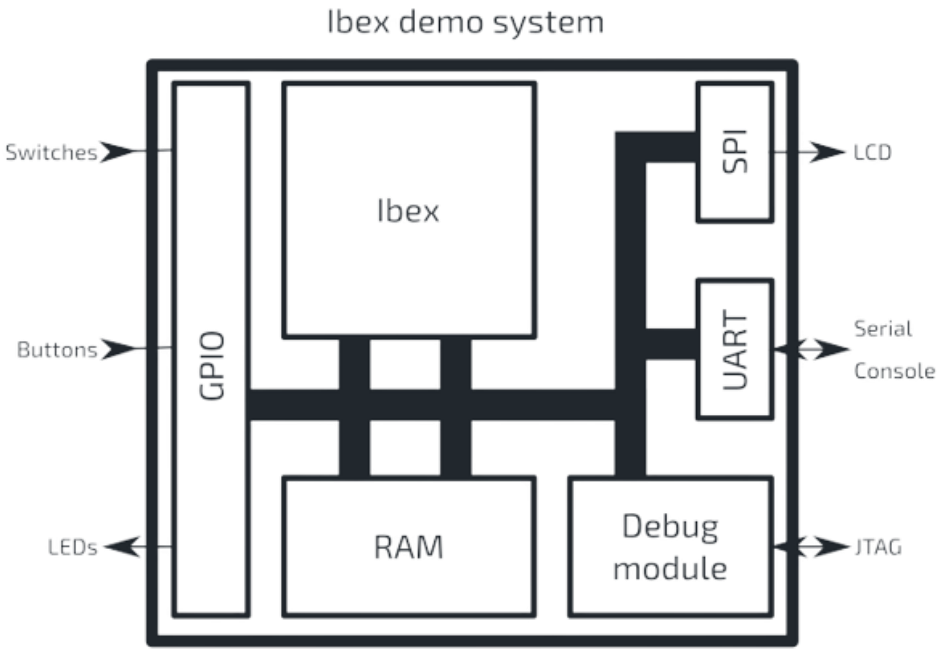
Corea frogatzeko diseinuak

Atal honetan *ibex-demo-system*a azalduko da, hau erabili baita Ibex frogatzeko. Nahiz eta ez den erabili, oso interesgarria ikusi da OpenTitan, Ibexen oinarrituta segurtasun aplikazioetarako prest dagoen maila profesionaleko SoC baten diseinua baita. Hori dela eta, diseinu hau gainetik azalduko da ere.

Ibex-demo-system

Ibexekin frogak egiteko edo diseinu abiapuntutzat hartzeko diseinu bat da hau, bere izenak dioen bezala. Sisteman OBI bus bat aurkitu daiteke, zeinak prozesagailua eta sistemaren periferikoak lotzen dituen. Sistemak SPI Master bat, UART bat eta GPIO periferiko bat gehitzen ditu, nahiz eta SPI eta GPIO periferikoak amaitu gabe egon. Datuen memoria busaren bidez atzitzen da ere, instrukzioen memoria memoria-interfazetik zuzenean atzitzen dela.

Sistemak debug interfaze bat du ere, JTAG bidez atzitu daitekeena. Interfaze hau dela eta, ibex-demo-systemak GDB *debugging* programa erabiltzea ahalbidetzen du, hau LowRISC-en *toolchain*-aren bitartez erabili daitekeela. Verilator sistema simulatzeko erabili daitekeela kontuan hartuz ere, demo-systemak *debugging* eta simulaziorako *suite* egoki bat eskaintzen duela ondorioztatu daiteke. Jarraian *ibex-demo-system-aren* bloke-diagrama aurkezten da, azalpen hauetan aipatu diren blokeak eta haien konexioak identifikatzea errazteko.



21. Irudia: *Ibex-demo-system*-aren bloke-diagrama

Ibex-demo-system-aren tamaina txikia dela eta, diseinua FPGA gailu txikietara eraman daiteke, proiektuak oso eskuragarriak diren Xilinxen Artix-7 FPGAn oinarritutako Arty plakara zuzendu duela diseinua. Hau abantaila handia da frogak FPGA merke batean egin ahal izateko.

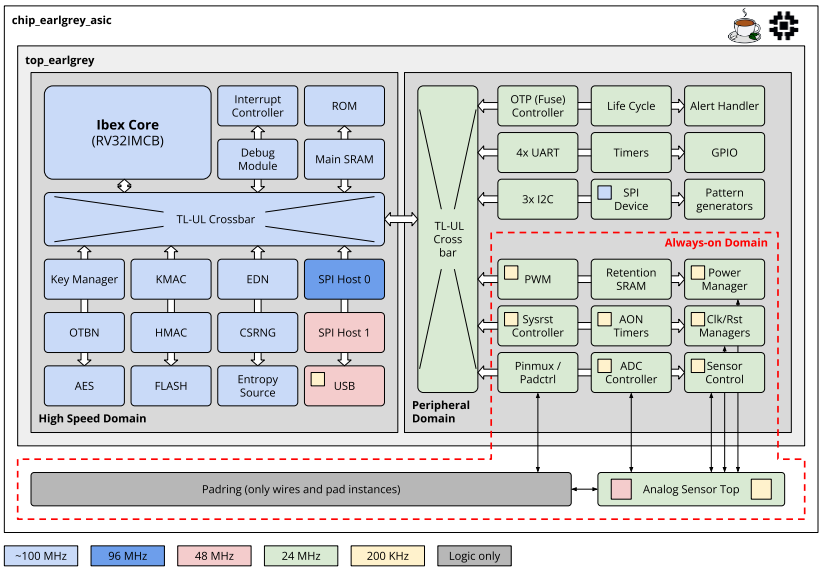
Gainera, diseinuaren proiektuak softwarea bertan hedatzeko erraztasun handiak ipintzen ditu. Alde batetik, corea FPGA batean hedatu ondoren, konpilatutako programen karga ahalbidetzen duten script batzuk jartzen dira eskura, prozesu hau errazten. Bestetik, GDB *debuggerra* erabiltzeko azpiegitura eskaintzen du proiektuak, lehen aipatu den *debug* unitatearen bitartez.

OpenTitan

Bestalde, LowRISC erakundeak OpenTitan proiektua eskaintzen du. Sistema hau *Root of Trust* edo RoT bat da, segurtasun funtzio espezifiko eta kritiko bat edo gehiago betetzen dituen hardware-sistema oso fidagarria alegia, zeinaren prozesagailua Ibex den. Segurtasun funtzio hauentzat eta bestelako funtzio orokorretarako periferiko asko ditu, IP moduan eskaintzen direnak. IP hauek kode-irekikoak dira eta hauen egileak erakunde ezberdinak dira.

Proiektu honek Ibex prozesagailuaren heldutasuna oso argi erakusten du. Hainbat enpresa pribatu oso garrantzitsuk parte hartu dute proiektu honetan, Ibexen oinarritu direla RoT fidagarri eta ireki bat sortzeko, corearen prestutasun-mailaz fidatu direla eta, Ibexen fidagarritasuna islatuz.

Segurtasunera zuzendutako Periferikoen eskakizun ezberdinak direla eta, sistemak erloju domeinu hainbat ditu, nagusienak 100 eta 24 MHz-koak direla. Domeinu bi hauetan, periferikoen interkonexiorako TL-DL bus bi aurki daitezke. Bus hau SiFive enpresak garatutakoa da, honen espezifikazioa irekia dela. Hona hemen proiektuaren bloke-diagrama.



22. Irudia: OpenTitan sistemaren bloke-diagrama

Ikusi daitekeenez, aukera hau oso interesgarria izan daiteke segurtasun-aplikazioetarako. Eskaintzen dituen periferiko irekiek egiaztapen prozesu sakonak jasan dituzte, sektorean liderrak diren enpresek eraman dutela garapen eta egiaztapen prozesua. Honela, diseinuaren helburua ez da tamaina bezalako alderdiak optimizatzea, segurtasun-aplikazio hauen errendimendua hobetzea baizik. Hori dela eta, tamaina aldetik diseinu oso handi bat izan da emaitza.

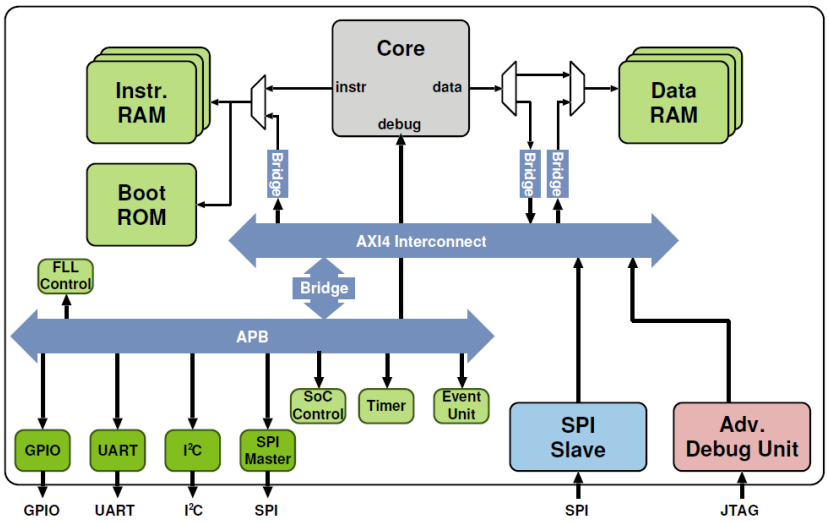
Argi dago hau ez dela proiektu honetan lortu nahi den helburua, baina hainbat osagai kenduz, OpenTitanen diseinu sinplifikatu bat lortzea beste aukera interesgarri bat izan daiteke, beharrezkoak ikusten diren periferikoak mantenduz. Hala ere, diseinuaren egitura dela eta, ataza hau ez da berehalakoa. Bestalde, erabilitako busa ez da oso ezaguna, periferiko berriak sartzeko eragozpen bat izan daitekeena hauek bus honetara egokitu beharko bailirateke eta.

Bestalde, diseinuaren tamaina dela eta, diseinua FPGA handietan inplementatzeko pentsatua dago, zehazki Xilinxen Kintex-7 FPGA daraman Avneten Genesys-2 plaka izugarri garestira zuzenduta dagoela OpenTitan.

PULPino

PULPino-ren kasuan, honen diseinuaren inguruko azalpen gehigarriak emango dira; bere coreak Ibex eta CV32E40P-ren ia berdinak direnez egitura aldetik, hauen inguruko azalpenak ez dira beharrezkoak ikusten. PULPino-rekin egindako lana dokumentu honetan aurkezten ez den eskuliburu zabal batean bildu da.

Jarraian sistemaren bloke-diagrama aztertzen da, gero bere osagaiak banaka aztertuko direla.



23. Irudia: PULPino sistemaren bloke-diagrama

Diseinuaren osagaiak jarraian zerrendatzen dira. Bloke guztiak kode-irekikoak direla.

- **Corea (AXI Master):** RI5CY edo zero-riscy-ren artean aukera. Memoria interfazeez gain, debugerako interfaze bat du, sistemaren bus nagusira lotua, *debug* prozesuan bertatik corera atzipena gauzatu daitekeela.
- **Instrukzio eta datu memoriak:** Tamaina konfiguragarria duten memoria bi.
- **Boot ROMa:** ASIC inplementaziorako pentsatua, memoria honetan sistema hasieratzerakoan exekutatuko den programa gordetzen da.
- **AXI Interconnect:** Sistemaren bus nagusia, AXI-4 estandarraren inplementazioa.
- **APB Busa (AXI Slave):** AXI busera lotuta, sistemaren periferiko ezberdinentzat erabiltzen den bigarren bus bat da.
- **Periferikoak:** GPIO, denboragailu edota UART bezalako erabilera orokorreko hainbat periferiko daude sisteman, kanpoko gailuekin komunikatzeko I2C eta SPI Master bat aurkitu daitekeela ere. Bestalde, SoC-aren kontrolerako erregistro batzuez osatutako periferiko bat aurki daiteke ere.
- **SPI Slave (AXI Master):** PULPino-ren kontrol osoa lortu daiteke SPI interfaze honen bidez. Bertara konektatutako SPI Master gailu batek sistema osora du atzipena; periferikoetatik corearen erregistroetara.
- **Debug unitatea:** *Debug* prozesua gauzatzeko portua, JTAG bidez sistemaren kontrola lortzea ahalbidetzen duenak.

PULPino ZedBoard plakan erabiltzeko prest dago, honek Zynq SoC-a duela; Artix FPGA bat eta ARM Cortex hardcore prozesadore batez osatutako SoC-a alegia. PULP plataformak plaka hori SoC-arengatik aukeratu zuen, FPGAn PULPinoren diseinua modu eroso batean kargatzeko prozesagailutik. Honela software frogak erraztu nahi ziren, prozesagailuak FPGAra duen sarbideaz baliatuz.

III. Eranskina: SoC diseinuarentzat abiatze kodea edo firmwarearen deskribapena

Dokumentuan zehar azaldu den bezala, SoC-a bere kabuz abiatzeko memorien sistema bat inplementatu da. Eranskin honetan, *boot* ROMean gordeko den kodea aurkezten da. Noski, programak Ibexen hedatzeko *LowRISC*-en GCC *toolchain*a aukeratu denez, kode hau C lengoaian dago idatzia.

Kode hau da SoC-eko prozesagailua piztean exekutatuko den lehen kodea, horretarako konfiguratu dela prozesagailua. Bere eginkizuna sinplea da: alde batetik, kodeak SPI Master periferikoa erabiltzen du SoC-etik kanpo dagoen flash memoriaren edukiak irakurtzeko eta, bestetik, flash memoriatik irakurritako programa SoC-eko instrukzioen memorian idazten du. *Boot* ROMaren programak amaieran, exekuzioa instrukzioen memoriako programari ematen dio, programa honen *entry point*era jauzi eginez.

Programak SPI Master periferikoari loturiko hainbat funtzio erabiltzen ditu, honi loturiko *spi.h header* eta *spi.c* iturri fitxategietan definituta daudenak. Funtzio hauek periferikoak dituen kontrol erregistro ezberdinetara irakurketak edo idazketak egiteko aukera ematen dute.

Nahiz eta kodearen funtzionamendu orokorra edozein kasutarako baliagarria den, kode hau Arty plakan dagoen N25Q128A flash memoriarentzat pentsatuta dagoela gogoan izan behar da, flasharekin komunikazioa modelo batetik bestera aldatu daiteke eta.

Kodea eranskinaren amaieran aurkezten da. Jarraian honen atal esanguratsuenak azalduko dira, *main* funtzioaren atal ezberdinetan sakonduz gehienbat.

Kodearen azalpena

Periferikoak abiatzea

29 eta 36 lerroetan SPI eta UART periferikoak abiatzen dira, hauen berrezartzerako beharrezkoak diren funtzioak exekutatuz.

Flashetik *headerra* irakurtzea

Flash memoriaren lehen irakurketa SPI bidez 51 eta 71 lerroen artean gauzatzen da. Bertan, 0. helbidetik hasita, 32 biteko irakurketa sei egiten dira. Sortutako *boot* sistema honetan, *header* edo goiburu bat eskatzen da flash memoriaren hasieran, goiburuak ondoko eremuak dituela:

- **0., 3., 4. eta 5. eremuak**: Erreserbatuak
- **1. eremua:** Kargatu nahi den programaren edukiaren lehen helbidea flashean
- **2. eremua:** Kargatu nahi den programaren 32 bitetako instrukzio kopurua

Eremuen edukiak *hdr* izeneko sei posizioko array estatikoan gorde dira.

Gainontzeko eremuak erreserbatu gisa utzi dira, esate baterako, helbide hauetako batean kargatu nahi den kodearen *checksumaren* emaitza gorde nahi bada. Honela kodearen integritatea egiaztatu daiteke: *headerra* irakurri ondoren, *boot* kodean flashak gordetzen duen programaren *checksuma* kalkulatu daiteke, emaitza goiburuaren eremu honekin konparatuz. Ezaugarri hau *boot* kodean inplementatu da, baina ez da kodean erakusten, hau sinplifikatzeko asmoarekin.

66. lerroan ikusi daitekeenez, irakurketa-helbidea lau unitatetan handiagotzen da. Honen arrazoia flash memoriatik irakurtzeko era da; irakurketa bakoitzean 32 bit irakurri nahi dira, baina memoriak helbide bakoitzean 8 bit gordetzen ditu. Ondorioz, 32 bit irakurri ondoren, hurrengo 32 biten hasiera lau helbide aurrerago egongo da. Hau dela eta, SPI Masterrak FIFO ilara bat inplementatzen du, prozesagailuak bertatik *spi_read_fifo* funtzioan adierazten diren bit kopurua irakurri dezakeela.

flashetik exekutatu nahi den programa irakurtzea

Irakurri nahi diren instrukzio kopurua *num* aldagaian gordeta, eta instrukzioen RAM memorian kargatu nahi den programaren kokapena flashean ezagututa, *addr* aldagaian gordetzen dena, nahikoa izango da *addr* helbidetik hasita, 32 bit *num* aldiz irakurtzea begizta batean, irakurritako 32 bit bakoitzak instrukzioen RAMean gordetzen direla. Prozesu hau 73 eta 96. lerroen artean gauzatzen da.

Prozedura lehen azaldutako begiztaren oso antzekoa da, soilik flashetik irakurritako edukiak instrukzioen RAMera apuntatzen duen erakusle batean gordetzen direla. Erakusleak erakusten duen helbidea unitate batean handiagotzen da lau unitateetan beharrean, erakuslea 32 bitekoa baita hau *int* bezala deklaratu delako eta konpiladoreari 32 biteko arkitekturarekin lan egitea adierazi baitzaio.

Exekutatu nahi den programara jauzia

Begizta amaitu ondoren, prozesagailuak *boot* ROMeko kodetik irten eta honek instrukzioen RAMean kargatutako programara jauzi egin behar du. Horretarako, C kodean RISC-V-ren mihiztatzaile lengoaiako *j* edo *jump* pseudoinstrukzioa txertatu da 101. lerroan. Instrukzioen RAMean kargatutako programak bere *entry pointa* 80. instrukzioan duenez, *boot* kodeak programa honen instrukzio honetara egin beharko du jauzi, zeina, instrukzioen RAMak sisteman duen helbidea dela eta, 0x00112080 helbidean kokatzen den.

Kodea

Jarraian, azaldutako C kodea erakusten da.

```
001 // Copyright lowRISC contributors.
002 // Licensed under the Apache License, Version 2.0, see LICENSE for details.
003 // SPDX-License-Identifier: Apache-2.0
004 // Unai Galiciak moldatua ROMetik boota egiteko

006 #include "demo_system.h"
007 #include "gpio.h"
008 #include <stdbool.h>
009 #include "dev_access.h"
010 #include "spi2.h"

012 void test_uart_irq_handler(void) __attribute__((interrupt));
013 void send_flash_command(char quad, char cmd, int addr, int addr_len, int datalen, char read);
014 int check_spi_flash();

016 //UART bidez interrupzio bat jasotzean exekutatu beharreko froga-funtzioa
017 void test_uart_irq_handler(void) {

019   int uart_in_char;
020   while ((uart_in_char = uart_in(DEFAULT_UART)) != -1) {
021     uart_out(DEFAULT_UART, uart_in_char);
022     uart_out(DEFAULT_UART, '\r');
023     uart_out(DEFAULT_UART, '\n');
024   }
025 }



027 int main(void) {

029   //UARTari dagokion IRQari aurreko funtzioa esleitu
030   install_exception_handler(UART_IRQ_NUM, &test_uart_irq_handler);

032   //UARTa baimendu
033   uart_enable_rx_int();

035   //SPI periferikoaren erloju-zatitzailea konfiguratu
036   *(volatile int*) (SPI_REG_CLKDIV) = 0x4;

039   puts("SPI Reset irakurtzen hasi baino lehen...\n");

041   //Write enable komandoa bidali flashera
042   send_flash_command(0, 0x06, 0, 0, 0, 0);
043   while ((spi_get_statuss() & 0xFFFF) != 1);

045   // QSPI Gaitzeko komandoa
046   spi_setup_cmd_addr(0x61, 8, 0x5F, 8);
047   spi_set_datalen(0);
048   spi_start_transaction(SPI_CMD_WR, SPI_CSN0);
049   while ((spi_get_statuss() & 0xFFFF) != 1);
```

```
051   puts("flasheko headerra irakurtzen...\n");
053   int hdr[6], addr = 0x0;

055   // Irakurketa komandoa: 0xEB --> fast read
056   // 8 ziklo nagi behar ditu flash memoriak
057   spi_setup_dummy(8, 0);

059   //Headerrak 6 eremu dituenez, 32 bit 6 aldiz irakurri
060   for (int i=0; i<6; i++){

062     //Fast read irakurketa komandoa
063     send_flash_command(1, 0xEB, addr, 32, 32, 1);
064     spi_read_fifo(&hdr[i], 32);

066     addr += 0x4;

068     spi_start_transaction(SPI_CMD_SWRST, SPI_CSN0);
069     while ((spi_get_statuss() & 0xFFFF) != 1);
071   }

073   //Instrukzioen RAM memoriaren helbidera erakuslea
074   int *mem = (int *)0x112000;
075   int num = hdr[2];
076
077   addr = hdr[1];
078
079   // Irakurketa komandoa: 0xEB --> fast read
080   // 8 ziklo nagi behar ditu flash memoriak
081   spi_setup_dummy(8, 0);
082
083   // num aldiz 32 bit irakurri
084   for(int i = 0; i<num; i++){
085
086     // Fast read irakurketa komandoa
087     send_flash_command(1, 0xEB, addr, 32, 32, 1);
088     spi_read_fifo(mem, 32);
089
090     addr += 4;
091     mem += 1; //32 biteko erakuslea
092
093     // SPI FIFOa badaezpada berrezarri
094     spi_start_transaction(SPI_CMD_SWRST, SPI_CSN0);
095     while ((spi_get_statuss() & 0xFFFF) != 1);
096   }

098   puts("OK; Exekuzioa instrukzioen RAMera eramaten...\n");

100   //RISC-V jump pseudo instrukzioa kodearen entry-pointera
101   __asm__ __volatile__("j 0x112080" );

103 }
```

```
104 // SPI Masterretik atzigarri dagoen flasha espero dena dela egiaztatzeko funtzioa
105 int check_spi_flash() {
106   int err = 0;
107   int rd_id[2];

109   // flasharen IDa irakurri
110   spi_setup_cmd_addr(0x9F, 8, 0, 0);
111   spi_set_datalen(64);
112   spi_setup_dummy(0, 0);
113   spi_start_transaction(SPI_CMD_RD, SPI_CSN0);
114   spi_read_fifo(rd_id, 64);

117   // Arty plakan dagoen flashAren IDa dela egiaztatu (0x0102194D)
118   if (((rd_id[0] >> 24) & 0xFF) != 0x01)
119     err++;

121   // flasharen kapazitatea 128MB edo 256 MBk den egiaztatu, 1.8 Veko elikadurarekin
122   // Informazio hau flasharen IDaren araberakoa da
123   if ( (((rd_id[0] >> 8) & 0xFFFF) != 0x0219) &&
124        (((rd_id[0] >> 8) & 0xFFFF) != 0x2018) )
125     err++;

127   return err;
128 }

130 // flashera komando bat bidaltzeko funtzioa
131 // Parametroak:
132 //   - quad:          QuadSPI erabili nahi den (>0) edo ez (0)
133 //   - cmd:           Komandoaren IDa
134 //   - addr:          Komandoa flasharen zein helbideetan exekutatu nahi den
135 //   - addr_len:      Adierazitako helbidearen luzera bitetan
136 //   - datalen:       Irakurri/idatzi nahi den dataren luzera bitetan
137 //   - read:          Irakurketa (>0) edo idazketa (0) egin nahi den
138 //
139 void send_flash_command(char quad, char cmd, int addr, int addr_len, int datalen, char read)
140 {
141   spi_setup_cmd_addr(cmd, 8, ((addr << 8) & 0xFFFFFF00), addr_len);
142   spi_set_datalen(datalen);

143   if (quad > 0)
144     if (read > 0)
145       spi_start_transaction(SPI_CMD_QRD, SPI_CSN0);
146     else
147       spi_start_transaction(SPI_CMD_QWR, SPI_CSN0);
148   else
149     if (read > 0)
150       spi_start_transaction(SPI_CMD_RD, SPI_CSN0);
151     else
152       spi_start_transaction(SPI_CMD_WR, SPI_CSN0);
153 }
```

24. Irudia: Boot firmwarearen kodea