Universiteti i Prishtinës
Fakulteti i Inxhinierisë Elektrike dhe Kompjuterike
Arkitektura e Kompjuterëve, 2022/2023
Prof. Valon Raça & Synim Selimi

Florian Saqipi, 210756100117
Gentrit Kryeziu, 210756100031
Fisnik Hazrolli, 210758100011
Adriatik Krasniqi, 210756100011

Detyra e dytë

Dizajnimi i një CPU 24-bitëshe (Single-Cycle)

1. Hyrje

Detyra e dytë ka kërkuar implementimin e CPU 24-bitëshe (Single-Cycle). Ne e kemi realizuar këtë implementim me anë të Verilog HDL dhe për ekzekutim, testim dhe simulim kemi përdorur Vivadon. Implementimi i CPU-së është bërë në bazë të instruksioneve të kërkuara në detyrë. Gjithashtu, në bazë të specifikave të cekura mbi arkitekturën 24-bitëshe. Së pari, vlen të ceket se për dizajnimin e CPU-së jemi bazuar në CPU-në 32-bitëshe (gjithashtu Single-Cycle), të ndërtuar gjatë ushtrimeve. Mirëpo, gjatë krijimit të moduleve të ndryshme të Single-Cycle-DataPath kemi modifikuar ato në mënyrë që të përputhen me kërkesat e cekura. Gjithashtu kemi vërejtur ndryshime në mënyrën se si këto module do të lidhen me njëra tjetrën. Qëllimi ynë ka qenë që këto module ti realizojmë e ti bashkojmë në mënyrë sa më efiçiente. Në figurën e mëposhtme shihet një verzion i thjeshtuar i CPU 24-bitëshe.

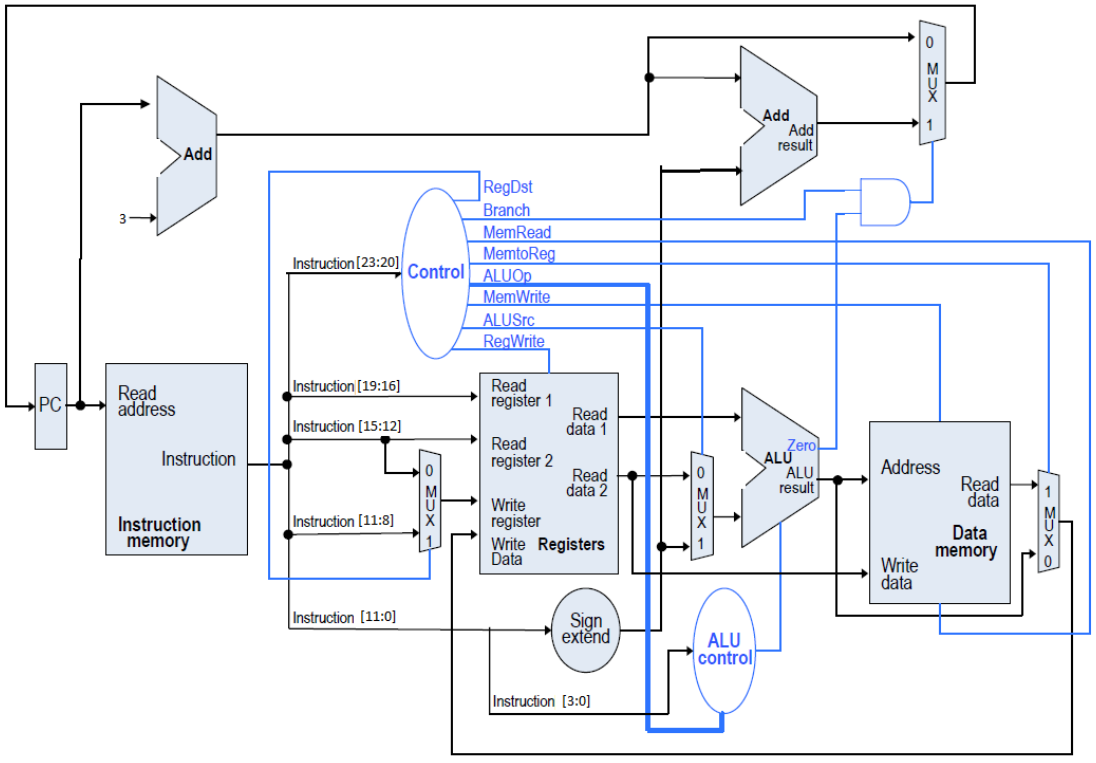

Figura 1. Single Cycle DataPath 24 bitësh(i thjeshtuar)

2. Dizajni

Fajllat e dorëzuar që gjenden përbrenda ZIP file ndahen në tre Foldera. Në Folderin Kodi gjenden të gjitha modulet si dhe dizajni i tyre ne Verilog. Këtu file kryesorë është CPU ku përmes tij bëhen ndarjet në të gjitha nënmodulet. Në folderin Memory gjenden gjendem Memory files si .mem files, pra DataMemory dhe InstructionMemory. Kurse në Folderin Test gjinden të gjitha modulet testuese të cilat kanë qenë të nevojshme.

ALU 1 bit

Së pari i gjithë dizajnimi ka startuar nga ALU 1 bitëshe fillimisht kemi bërë hulumtime në bazë të instruksioneve të cilat na kërkohen për mënyrën adekuate të implementimit të ALU 1 bitëshe. Pastaj, kemi filluar me kodin bazë në ushtrime si dhe nga ALU 1 bit e paraqitur në figurën e mëposhtme.

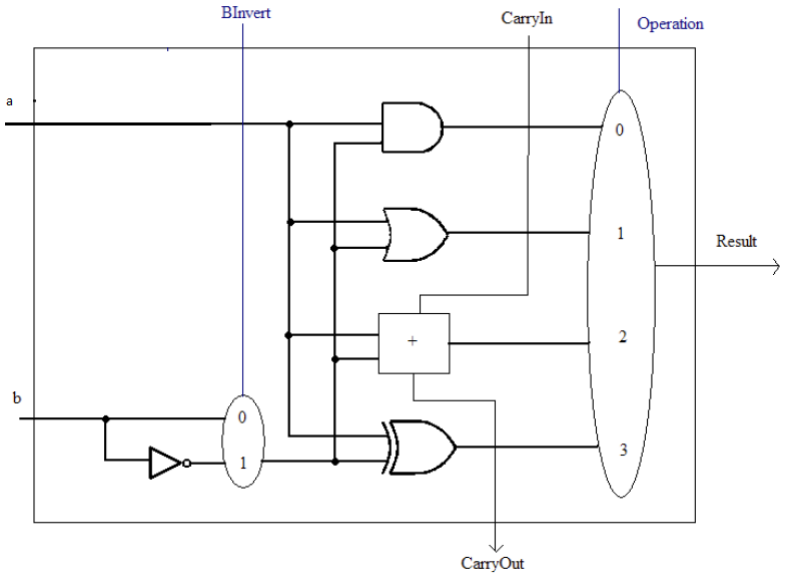

Figura 2. ALU 1 bitëshe

ALU 1 bitëshe të treguar në figurë ne e kemi modifikuar për nevojat e detyrës. Së pari ALU 1 bitëshe ka hyrje A dhe B. Gjithashtu Binvert dhe CarryIn i kemi bashkuar si Bnegate së implementim dhe 3 bit të Operation për selektim në multiplekser. ALU 1 bitëshe e implementuar përkrah operacionet AND, OR, ADD, SUB, LESS(SLT), XOR dhe SLL. Kurse MUL kemi vendosur ta implementojmë jashtë ALU 1 bitëshe. Gjithashtu kemi përdorur një multiplekser 8 në 1 (daljet) dhe multiplekserin 2 në 1 për invertimin e B.

- Portat AND, OR dhe XOR kryejnë operacionet përkatëse pra bitwise.
- LESS është gjithashtu hyrje në ALU 1 bit dhe me anë të këtij biti përcaktojmë se një hyrje a është me e vogël se tjetra (bazuar në bitin most significant).
- ADD dhe SUB realizohen përmes modulit Mbledhësi, ku zbritja bëhet sipas mbledhjes së 2 komplementit.
- SLL është një hyrje e cila na jep rezultatin për secilin bit në bazë të shifterit në ALU 24.

ALU 24 bit

ALU-në 24 bitëshe e kemi implmenetuar bazuar në kodin e ALU-së 32 bitëshe të ndërtuar gjatë ushtrimeve dhe e kemi përshtatur me kërkesat e detyrës. Struktura e ALU-së do dukej e ngjajshme me figurën poshtë por me disa përjashtime (nga kërkesat e detyrës).

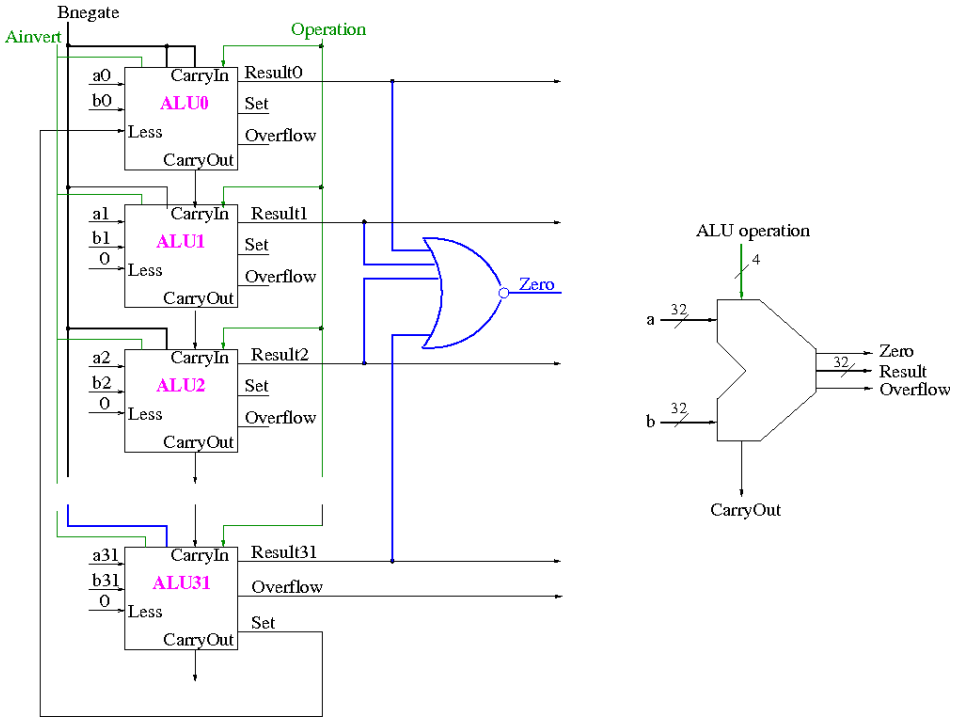

Figura 3. Struktura e ALU-së

ALU-ja jonë pranon: 2 hyrje A dhe B 24 bitëshe, hyrjen për Bnegate, 3 bit nga AluControl, 4 bit për SHAMT (për implementin e SLL), ndersa në dalje biti për Zero, rezultati 24 bitësh, Overflow dhe së fundi CarryOut. ALU-ja në vetvete përmban 24 njësi të ALU-ve 1 bitëshe të cilat implementojnë të gjitha operacionet aritmetiko-logjike , përveq implementimit të operacionit SLL me anë të shifterit, i cili realizohet në nivel të ALU-së 24-bitëshe. 4 Bitët e SHAMT në hyrje të ALU-së zgjerohen në 24 bit, ashtu që me anë të një Mbledhësi 24 bitësh të jetë e mundur të mblidhen me bitët e B (sipas kërkesës së detyrës). Hyrja A dhe shuma në mes të bitëve të SHAMT dhe B hyn në modulin e SHIFTER-it, ndërsa rezultati 24 bitësh i SHIFTER-it i ndarë në 24 pjesë futet si hyrje në 24 ALU-të 1 bitëshe.

ALU Control

ALU Control pranon 3 hyrje: ALUOp 2 bit, OpCode 4 bit, dhe Funct 4 bit, ndërsa dalja është ALUCtrl 4 bitëshe e cila futet pastaj futet në ALU.

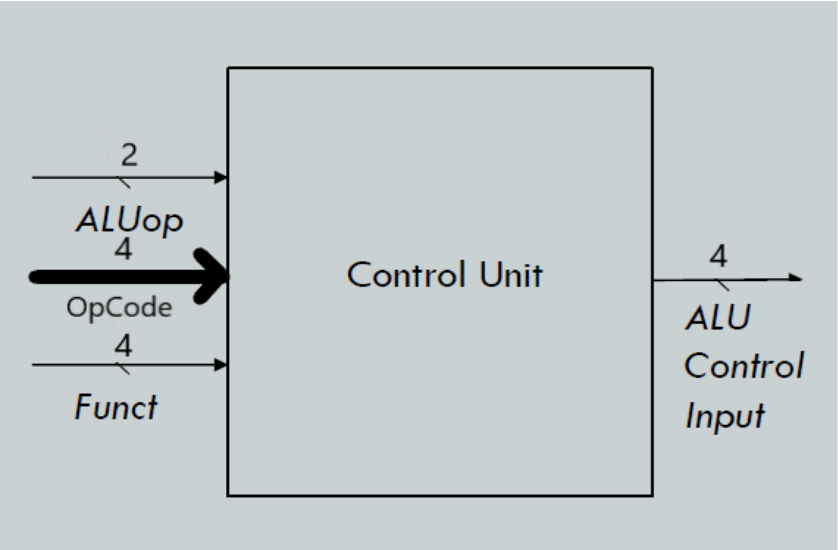

Figura 4. Struktura e ALU Control

Struktura e ALU Control është e bazuar në cases , përkatësisht në vlerat që kanë hyrjet në këtë njësi. Bazuar në vlerën e ALUOP kemi rastet për instruksionet për I-format, R-format, dhe për operacionin e MUL-it. Case për R-format në vetvete përmban një nën-case e cila pyet për vlerën e funksionit, dhe në bazë të saj cakton vlerën e ALUCtrl. Ndërsa case për operacionin e MUL-it gjithashtu përmban një nën-case e cila pyet për vlerën e OpCode. Arsyeja e kësaj strukture janë kërkesat që janë bërë në detyrë.

Control Unit

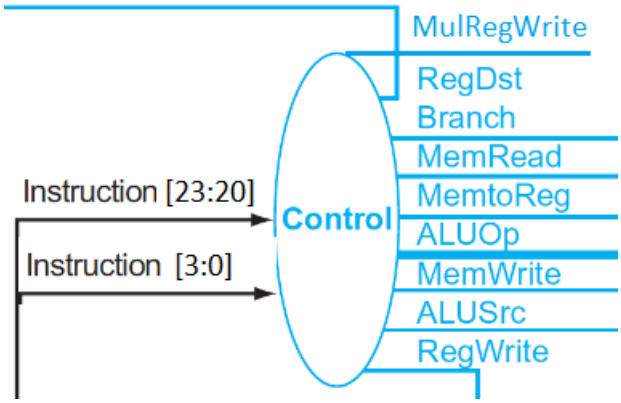

Figura 5. Moduli Control Unit

Control unit është një prej pjesëve themelore të CPU, pasi kombinuar me Datapath pra formohet vet CPU. Atëherë, Control Unit i merr si hyrje bitat e Opcode të instruksionit pra 4 bitat e parë(ne kemi shtuar edhe 4 bitat e funksionit) dhe e thënë më thjeshtë bën dekodimin e këtyre bitave në hyrje dhe në bazë të tyre sinjalet në dalje marrin vlerat përkatëse për çfardo instruksioni. Këto dalje janë të gjitha hyrje në DataPath. Në bazë të kësaj, DataPath kryen veprimet e nevojshme që të realizohet instruksioni. Ne kemi shtuar në hyrje bitat e funksionit dhe kemi shtuar në dalje MulRegWrite, në mënyrë që të dallojmë instruksionin MUL i cili shkruan në regjistrin special MULREG.

Data Memory

Data Memory mundëson funksonalitetit e leximit dhe të shkrimit në memory. Ne e kemi implementuar si një .mem file në të cilën mund të shkruhet dhe lexohet. Data memory si modul merr 5 hyrje: Adress 24 bitëshe, për adresën ku duhet shkruar, WriteData 24 bitëshe, për data e cila duhet shënuar, MemWrite, që lejon shkrimin në memorje, MemRead, që lejon leximin nga memorja, dhe Clock, në mënyrë që të mund të shkruajmë në teh, Këto hyrje janë nga 1 bit dhe si dalje është ReadData që jep të dhënën në adresën e kërkuar pra 24 bitëshe.

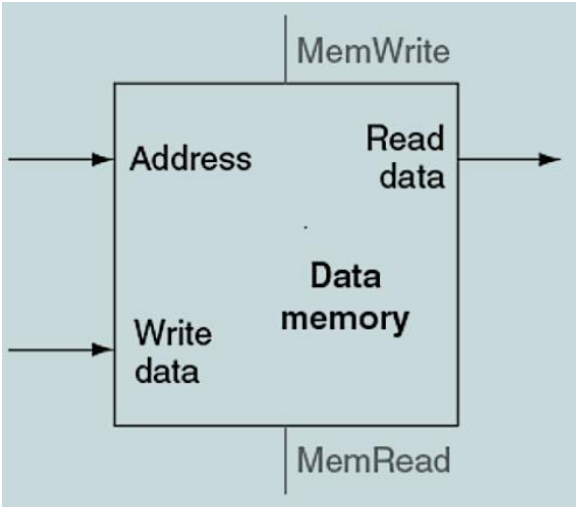

Figura 5. Struktura e Data Memory

DataPath

DataPath paraqet rrugëtimin e të dhënave nëpër CPU. Pra, çdo intstruksion do të kalojë nëpër DataPath dhe do kryej një funksion të caktuar. DataPath paraqitet si bashkim i të gjitha moduleve paraprake. Ne kemi implementuar DataPath në bazë të ushtrimeve, por me ndryshimet e nevojshme. Në hyrje DataPath ka të gjitha daljet e Control Unit, pra sinjalet të cilat kontrollojnë se me cilat module do të veprojnë të dhënat. Gjithashtu, njëkohësisht DataPath ka daljet që janë opcode dhe funksioni, funksionin e kemi përdorur për të dalluar instruksionin MUL. Kemi bërë mbledhjen përmes moduleve të Mbledhësve. Pra mund të vërehet DataPath në figurën e mëposhtme.

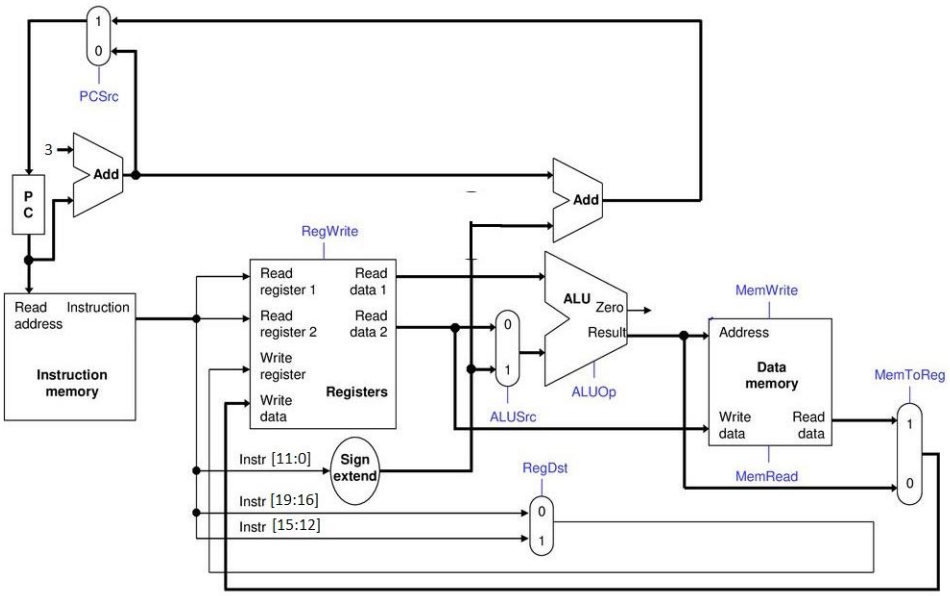

Figura 6. Struktura e Data Path

3. Ekzekutimi

Fajllat e Verilogut me anë të të cilave mund të bëjmë testimin dhe simulimin gjenden në Folderin Test. Këto kanë qenë fajllat që ne i kemi parë të nevojshme të testohen, përgjatë krijimit dhe bashkimit të moduleve.

Testimi i CPU

Testimi i CPU është bërë në Vivado dhe është bërë për 30 cikle. Në secilen prej cikleve mund të

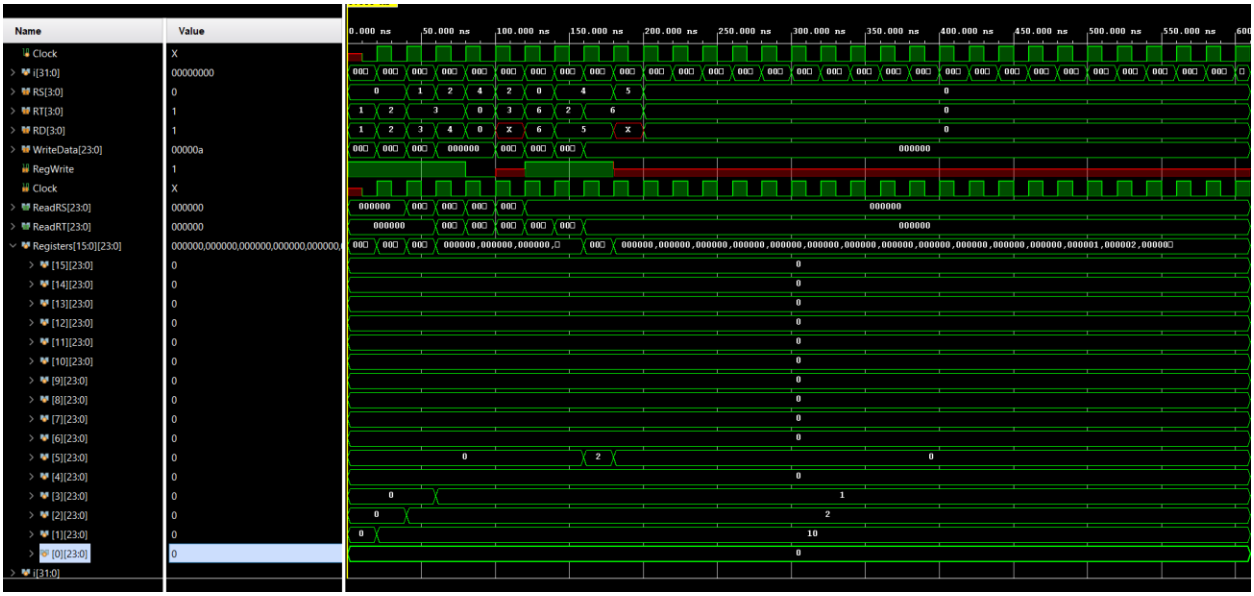

Figura 7. Struktura e Data Path

observojmë regjistrat si në figurën më lartë gjithashtu dhe daljet e Control Unit dhe me anë të këtyre ne mund të sigurohemi se janë ekzekutuar të gjitha instruksionet në Instruction Memory me sukses. Gjithashtu, mund të vërehet se kemi mundur të bejmë shkrim-lexim të memorjes duke shikuar files të gjeneruara. Si shihet në figurën me poshtë.

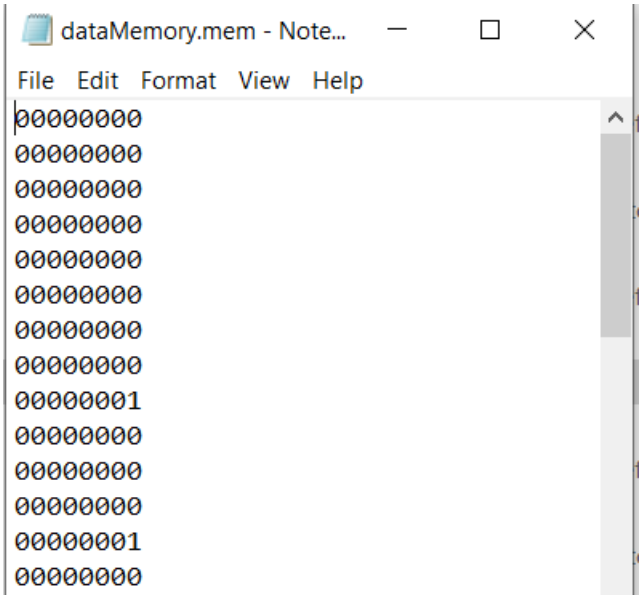

Figura 8. dataMemory pas instruksioneve

Testimi i Instruction Memory

Testimi i Instruction Memory është bërë gjithashtu me Vivado me anë të diagrameve. Ne kemi bërë

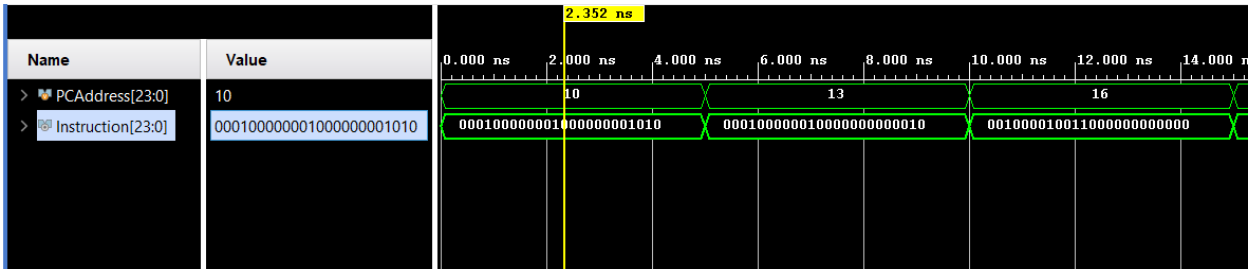

Figura 9. Testimi i Instruction Memory

leximin e Instruction Memory dhe kemi inkrementuar PC në mënyrë që të lexojë secilin instruksion. Në figurën më lartë mund të shihen vlerat e lexuara nga InstructionMemory.mem. Pra, kemi siguruar që mund të lexojmë nga kjo memorje.

Testimi i ALU 1 bitëshe

ALU 1 bitëshe e kemi testuar ngjashëm si modulet më lartë. Me anë të diagrameve kohore kemi konstatuar se ALU 1 bitëshe është tërësisht funksionale. Kemi bërë testimet e të gjitha veprimeve të cilat mund të bëhen përmes ALU 1 bitëshe.

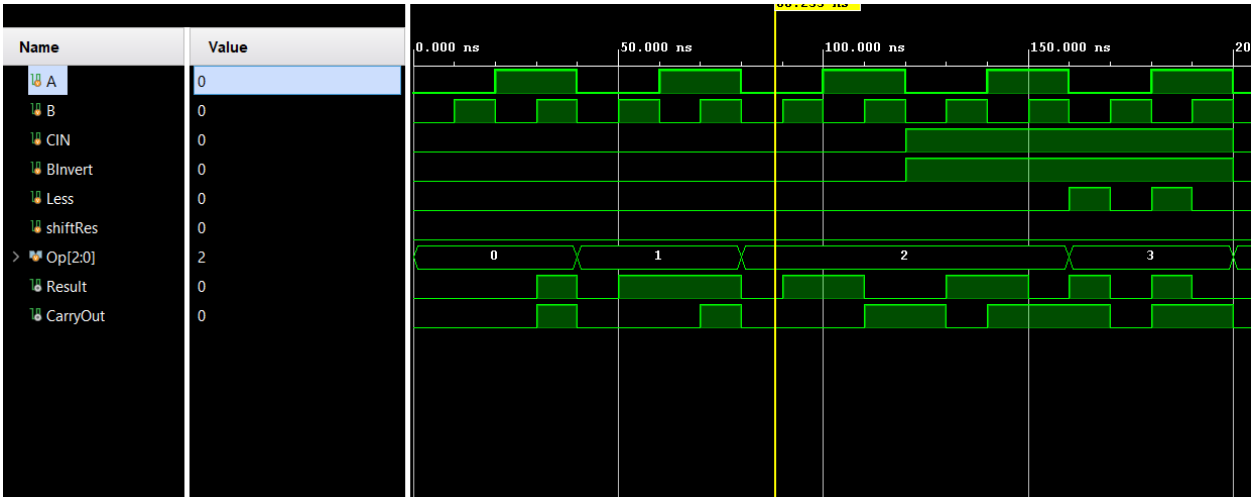

Figura 10. Testimi i ALU 1- bitëshe

Në figurën më lartë mund të shihet për operacionet AND, OR, ADD, SUB, LESS daljet janë korrekte. Pra, pas testimeve kemi bërë implementimin e modulit për 24 bit dhe pra e kemi ripërdorur.

Testimi i Register File

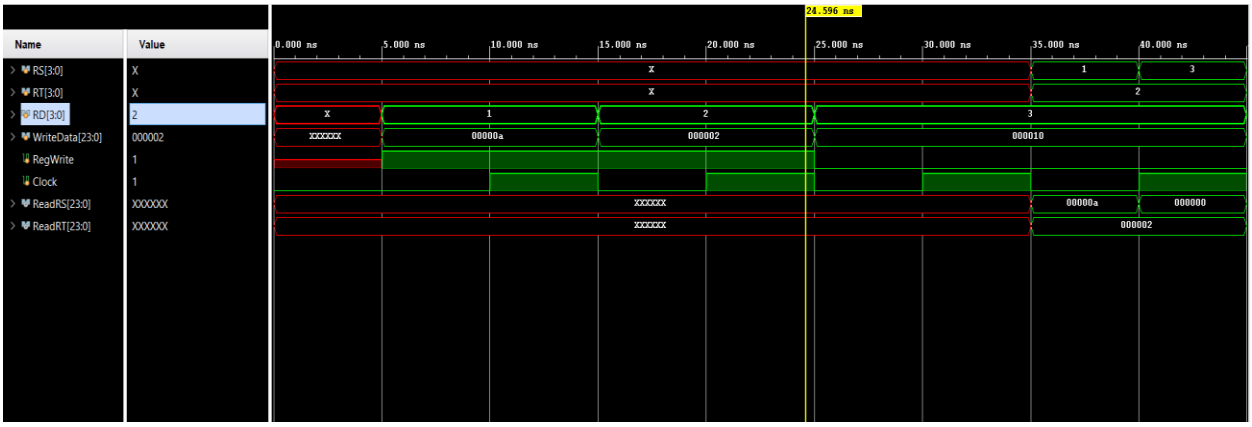

Figura 11. Testimi i Register File

Testimin e Register File gjithashtu e kemi bërë ngjashëm me anë të Vivados. Në figurën e mësipërme mund të shihet se në tehun rritës është bërë shkrimi i vlerës në Destination Register. Pas shkrimit të disa regjistrave i kemi vendosur vlera RD dhe RT në mënyrë që të lexohen këta regjistra në të cilët kemi shkruar.

4. Përfundimi

Detyra për ndërtimin e një CPU-je 24 bitëshe ka qenë mjaft sfiduese, mirëpo në fund ne i'a arritëm që ta përfundojmë me sukses atë. Implementimi i disa moduleve të detyrës na ka mësuar se si të mendojmë në mënyrë kreative për zgjidhjen e problemeve, dhe se ndonjëherë zgjidhja më e mirë është zgjidhja më e thjeshtë. Që të arrihet deri tek zgjidhja duhët këmbgulësi dhe fokus, gjë që kjo detyrë besoj na e ka dëshmuar më së miri. Qëllimi ynë si grup ka qenë që të implementojmë të gjitha kërkesat e detyrës në mënyrën sa më të thjeshtë dhe efiqiente të mundëshme. Në aspektin social, implementimi I detyrës na ka ndihmuar në koordinimin mes anëtarëve të grupit, dukë bërë kështu që ne të jemi më të aftë ne projektet grupore dhe gjithashtu të besojmë në aftësitë e njërit tjetrit.