Computer Architecture

Project 1 Report – Spring 2025

Pipelined Datapath

John Ebrahim

900226449

Zeina El Sawy

900223285

Ziad Gaballah

900221960

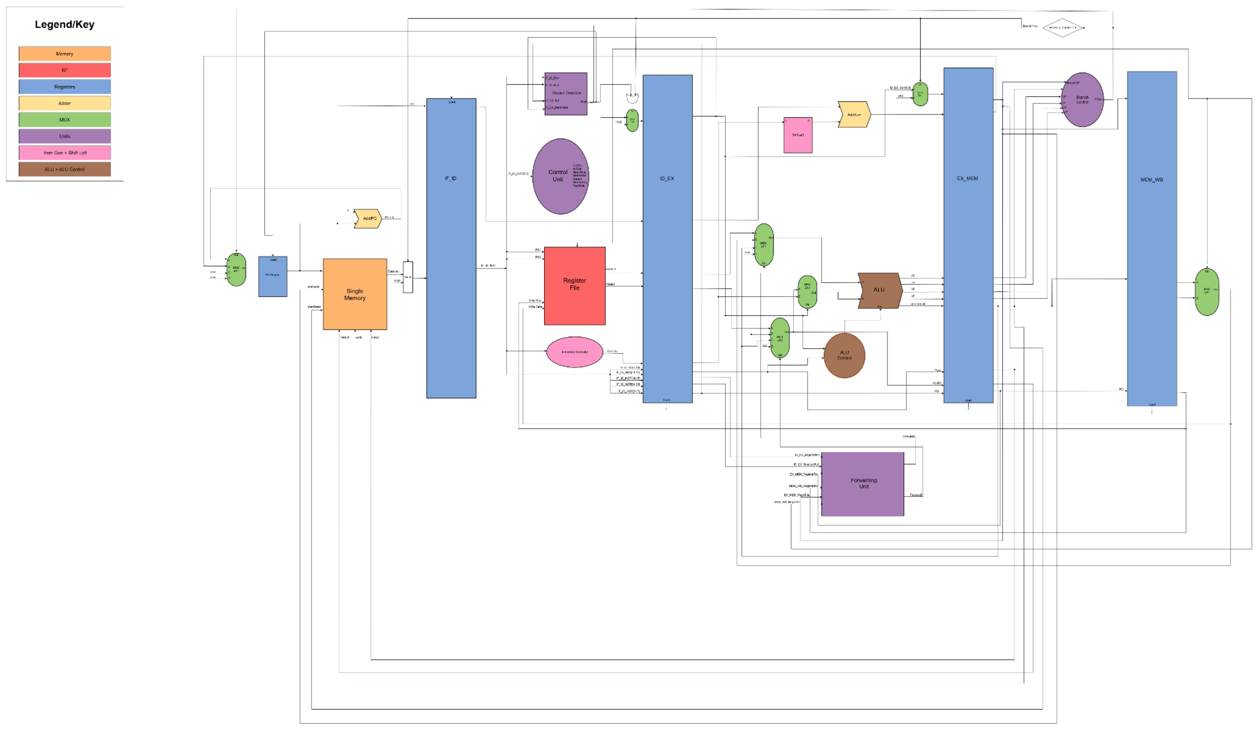
Supervised by

Dr. Cherif Salama

Introduction

In this project for CSCE 3301 – Computer Architecture, we will design and test a simple RISC-V processor on the Nexys A7 FPGA board. The processor must support the RV32I instruction set and use a pipelined design with shared memory for both instructions and data. The goal is to learn how modern CPUs work, especially how they handle instructions in stages and manage hazards. For this milestone, we moved beyond the single-cycle approach and implemented a complete pipelined datapath, with forwarding and hazard detection units, and verified the design through simulation.

Schematic Design



Implementation

- We added the registers IF/ID, ID/EX, EX/MEM, MEM/WB to turn our DP from single-cycled to pipelined.
- Implemented both the Forwarding Unit and the Hazard Detection Unit and their corresponding MUXs.
- Modified the load signal of the PC and IF/ID registers to be the NOT stall signal.
- We replaced the instruction memory and the data memory with a single memory.
- Split the memory into 256 bytes for instructions and the rest for data (maximum program

size = 64)

- We implemented all RISC-V multiplication and division instructions.
- We made the ALUsel 5 bits for the multiplication and division.

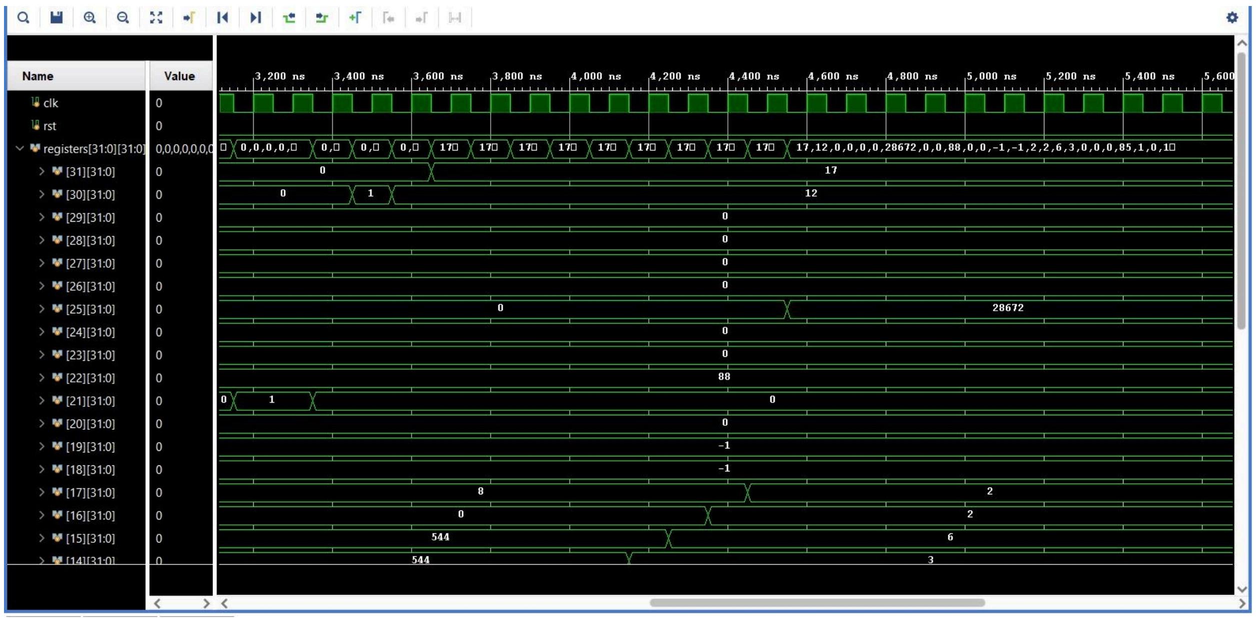
Issues

- A problem we encountered was while we were trying to build a pipelined datapath with the branch in the ID stage, we accidentally had another module as top and we had several errors in our code without realizing and it cost us a lot of time.
- A major issue we faced was the wiring between components. It cost us a lot of time to debug the errors that we would face.
- When starting the pipeline, we called the parameters of the registers incorrectly so some registers would work and others would not.
- Instruction SRAI was not operating correctly, it was being read as a load instruction because bit inst[25].
- SLL and SRL were not functioning properly until we realized they operate in the opposite fashion (SLL acts as SRL and vice versa).
- The JAL increments the PC incorrectly (pc + imm - 4) incorrectly stores the address in the register (Same for branch instructions).
- When trying to test data hazards, we were facing an undefined problem when testing RAW dependencies and we tried tracing it back to the Hazard Unit; however, the problem was unrelated to the unit and more to the structure of the instructions.

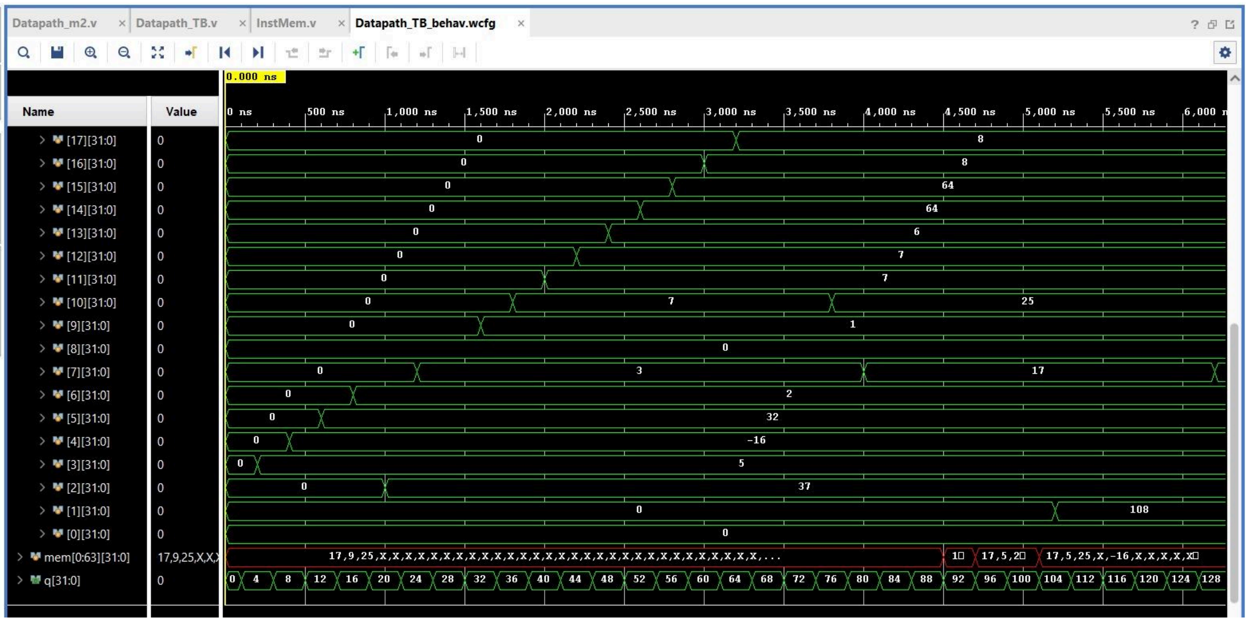
Solutions

- Realizing our pipeline with an early branch was faulty, we migrated back to a regular pipelined datapath and moved on from there. Although the wiring errors cost us time, we started to properly trace and debug the errors and fix them. Instructions SLL and SRL were fixed by switching the ALUsel signal between them to generate their expected output. To properly test data hazards, we implemented unit testing where we tackled each type of hazard separately such as Load-Use, Regular RAW, Double Data Dependencies. We also figured out that both JAL and branch instructions would jump by the (offset - 4), so we added 4 to the multiplexer input that corresponds to the address of the PC after a jump or branch.

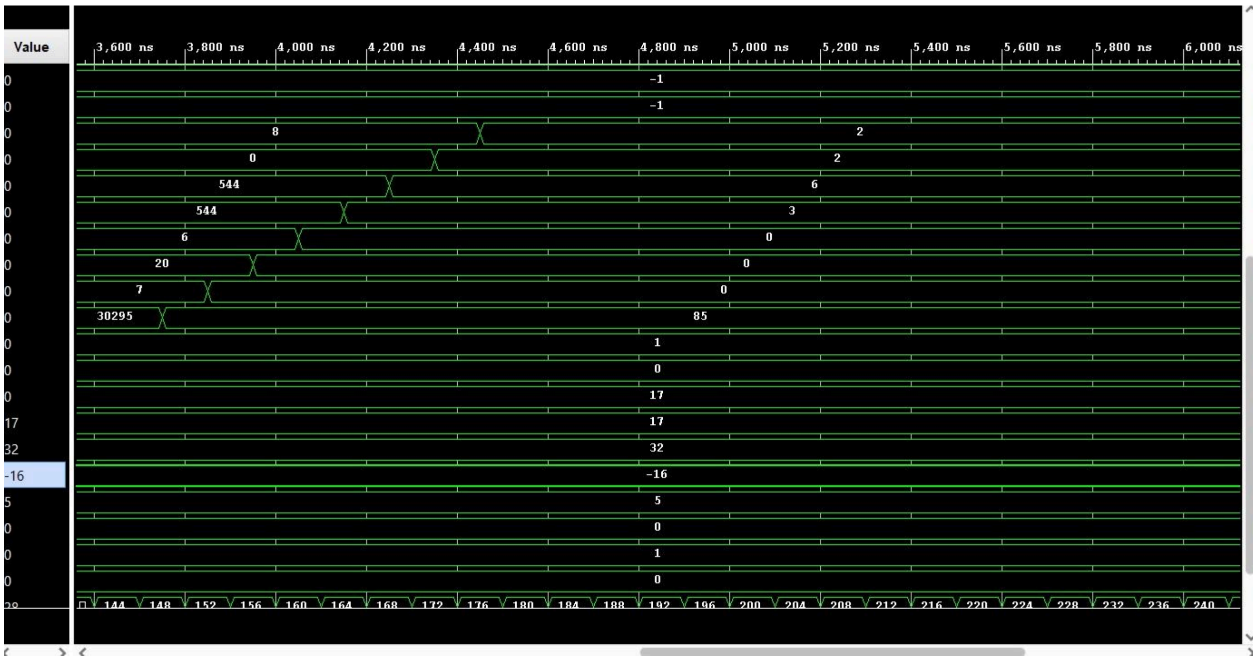
Test Waveforms

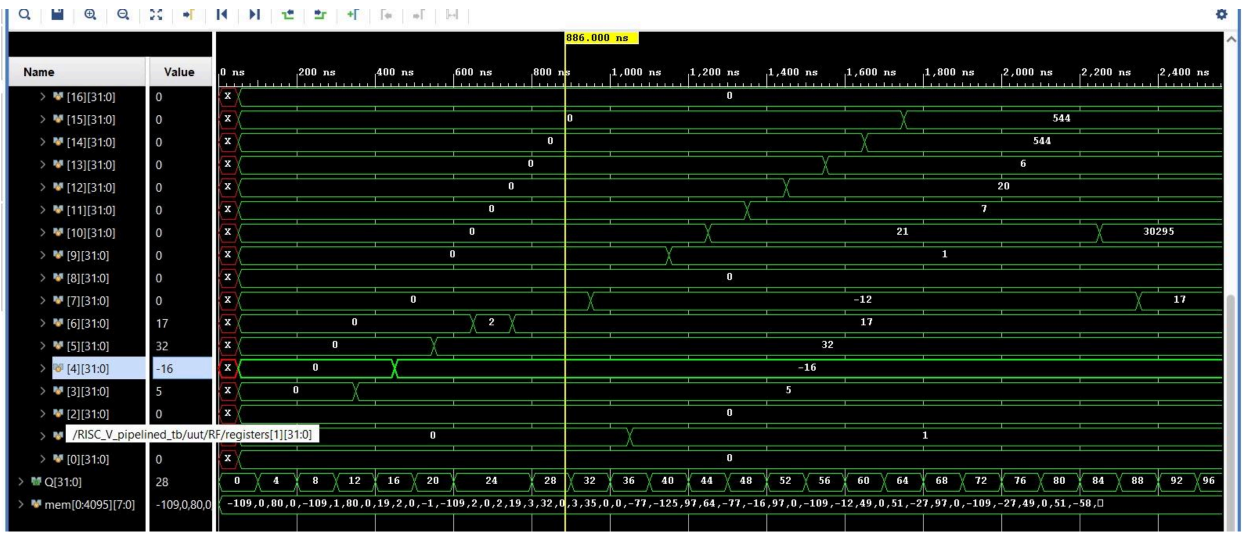


```
Name                      Value
clk                       0
rst                       0
registers[31:0][31:0]     0,0,0,0,0,0,0   0,0,0,0,0 | 0,0 | 0,0 | 0,0 | 17 | 17 | 17 | 17 | 17 | 17 | 17 | 17 | 17 | 17,12,0,0,0,0,28672,0,0,88,0,0,-1,-1,2,2,6,3,0,0,0,85,1,0,1
  [31][31:0]              0       0 | 17
  [30][31:0]              0       0 | 1 | 12
  [29][31:0]              0       0
  [28][31:0]              0       0
  [27][31:0]              0       0
  [26][31:0]              0       0
  [25][31:0]              0       0 | 28672
  [24][31:0]              0       0
  [23][31:0]              0       0
  [22][31:0]              0       88
  [21][31:0]              0       0 | 1 | 0
  [20][31:0]              0       0
  [19][31:0]              0       -1
  [18][31:0]              0       -1
  [17][31:0]              0       8 | 2
  [16][31:0]              0       0 | 2
  [15][31:0]              0       544 | 6
  [14][31:0]              0       544 | 3
```



```
Name                 Value
  [17][31:0]         0       0 | 8
  [16][31:0]         0       0 | 8
  [15][31:0]         0       0 | 64
  [14][31:0]         0       0 | 64
  [13][31:0]         0       0 | 6
  [12][31:0]         0       0 | 7
  [11][31:0]         0       0 | 7
  [10][31:0]         0       0 | 7 | 25
  [9][31:0]          0       0 | 1
  [8][31:0]          0       0
  [7][31:0]          0       0 | 3 | 17
  [6][31:0]          0       0 | 2
  [5][31:0]          0       0 | 32
  [4][31:0]          0       0 | -16
  [3][31:0]          0       0 | 5
  [2][31:0]          0       0 | 37
  [1][31:0]          0       0 | 108
  [0][31:0]          0       0
mem[0:63][31:0]      17,9,25,X,X,X   17,9,25,X,X,X,X,X,X,X,X,X,X,X,X,X,X,X,X,X,X,X,X,X,X,X,X,X,X,X,X,X,X,X,... | 1 | 17,5,2 | 17,5,25,X,-16,X,X,X,X,X
q[31:0]              0       0 | 4 | 8 | 12 | 16 | 20 | 24 | 28 | 32 | 36 | 40 | 44 | 48 | 52 | 56 | 60 | 64 | 68 | 72 | 76 | 80 | 84 | 88 | 92 | 96 | 100 | 104 | 112 | 116 | 120 | 124 | 128
```





- All instructions (tested or not tested): https://docs.google.com/spreadsheets/d/13qWn1fRsMHk_KRIKud-0jo2AdAoMOfh_JQ6Mubzvhk0/edit?usp=sharing