Bert Yu

25067519

March 21, 2019

Lab 4 Report

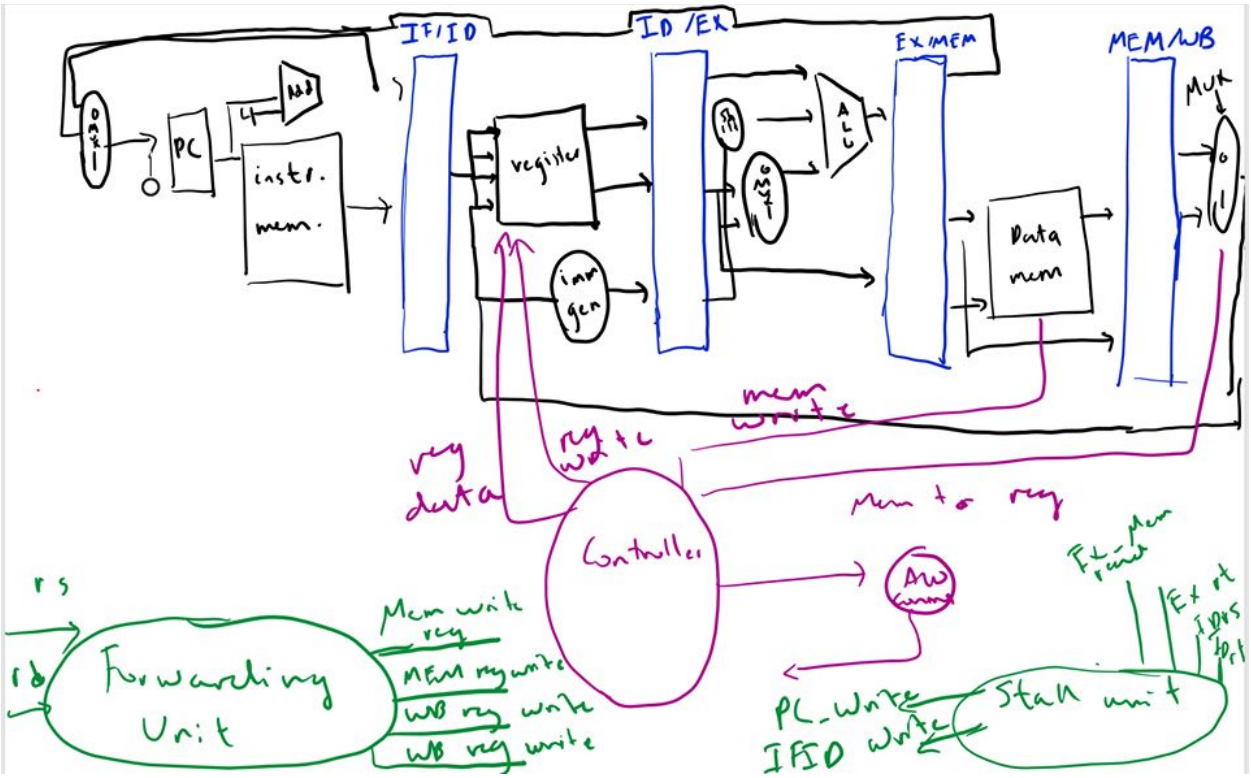

Fig 1 Shows the pipelined process block diagram with the Controller. Discard instruction and flush functions were not implemented. Only Forwarding and hazard detection were implemented

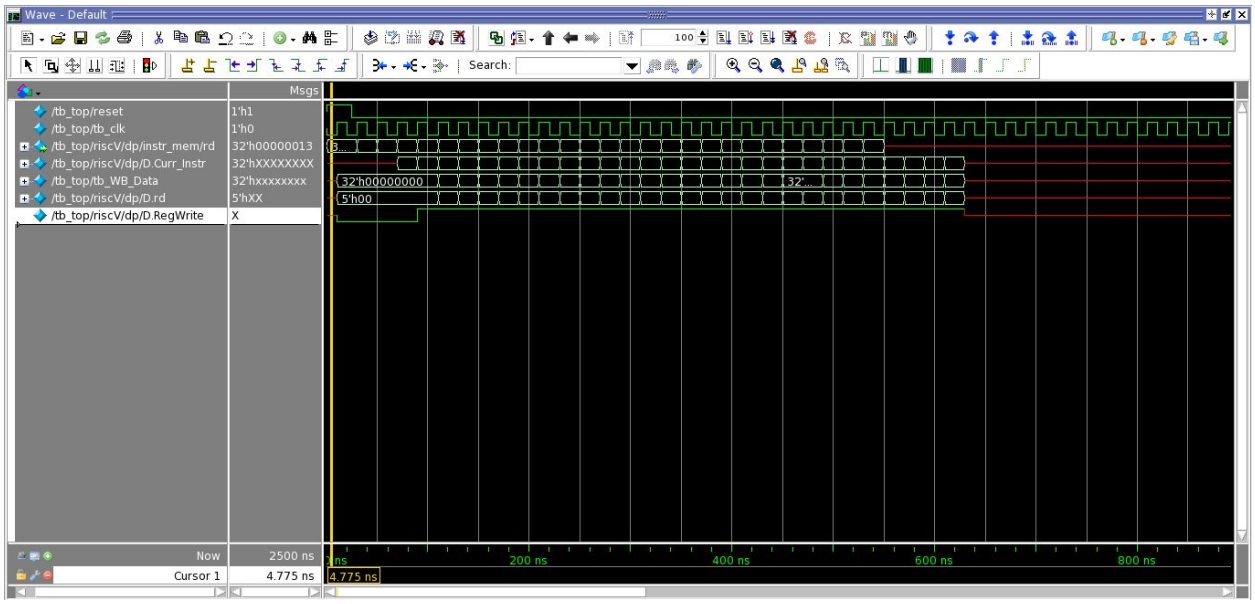

Fig 2 Shows the simulated results of the instructions run in the instruction memory.

Registers were created between IF and ID, ID and EX, EX and MEM, and MEM and WB.

Block Modules:

ALUController:

The ALUController takes the entire instruction as an input and then decides exactly which ALU instruction operation to take. This module splits bits 1 to 2, 25 to 31 and 12 to 14 of the instruction and then uses conditionals to determine which instruction to perform.

Controller:

The Controller operates in a very similar way to the ALUController. It splits the instructions up into different parts and then determines which segments of the instructions will be active in the datapath. By using conditionals, this module determines whether the data will go from memory to register, writing data to a register, reading from memory or writing to memory. To account for the forwarding and hazard detection, changes were made here.

Datapath:

The datapath determines instruction classes in the ISA. It also moves the data from place to place within the modules in design. Essentially this module is like the main a function that calls other modules to do work. Program counter, flopr, instruction memory, register file, immediate generate, and data memory are all called within this module.

RISC V:

This module behaves similarly to the datapath. This module calls the Datapath to perform its functions .Essentially this module calls controller, ALUController, and the datapath itself. In this module, parameters were changed to match the changes made in the controller and the datapath.

Parameters were changed.

Adder:
This module speaks for itself. It takes in two data types adds them together, and outputs the result. No changes were made here.

ALU:
This module performs any arithmetic operation found within the design. It runs a case statement which determines either a logical and, logical or, add, subtract or xor. The only changes made here were the additions of possible operations such as shifting, and branch instructions.

Data Memory:
Data memory takes parameters from the controller and then either performs a read from memory or a write to memory, depending on the results from the controller.

Flopr:
The flopr determines whether or not there is a positive clock edge to perform operations. Also, if the reset is switched on, the output will also be switched off.

The option to stall the clock was changed here.

Immediate Gen:
This module involves and I type load, I type addi, or S-type that involves an immediate value.Here the input is a 32 bit number, which converts to a 64 bit number.

Instruction Memory:
This module reads the address of the instruction memory, and then outputs the instruction that is located in the inst.bin file, allowing for the other modules to perform the needed operations.

No branch or jump instructions were implemented.

MUX2:
This mux allows for a choice between two different data outputs, depending on whether the inputs are true or not. This is used multiple times within the design.

MUX4:
This mux allows for a choice between two different data outputs, depending on whether the inputs are true or not. This is used multiple times within the design. However this mux is 4 input.

ForwardingUnit:
Forwarding selects muxs that are added to ALU Src A and B, however if there is no hazard mux will select RD1 and RD2 .

HazardDetection:
Hazard detection unit occurs when the source register in an early stage of the pipeline is also the same as the destination of another instruction at another stage of the pipeline. Once a hazard is

detected a NOP, or a stall occupies the execution stage of the processor, sent to the PC, and Register A and B.

PC, Register A and Register B is stalled every positive clock edge

RegPack:
This module is initialized to be the structure that forwards data to and back from the forwarding unit within the module.

Synthesis

```
Timing Path Group 'clk'
-----------------------------------
Levels of Logic:              18.00
Critical Path Length:          4.05
Critical Path Slack:          -0.07
Critical Path Clk Period:      4.00
Total Negative Slack:         -1.05
No. of Violating Paths:       41.00
Worst Hold Violation:          0.00
Total Hold Violation:          0.00
No. of Hold Violations:        0.00
-----------------------------------
```

```
Cell Count
-----------------------------------
Hierarchical Cell Count:          4
Hierarchical Port Count:        302
Leaf Cell Count:              54639
Buf/Inv Cell Count:            3566
Buf Cell Count:                 780
Inv Cell Count:                2786
CT Buf/Inv Cell Count:            0
Combinational Cell Count:     36915
Sequential Cell Count:        17724
Macro Count:                      0
-----------------------------------
```

```
Area
-----------------------------------
Combinational Area:    92153.885706
Noncombinational Area:
                      117075.503741
Buf/Inv Area:           5505.521512
Total Buffer Area:          1947.51
Total Inverter Area:        3558.02
Macro/Black Box Area:      0.000000
Net Area:             155358.965418
-----------------------------------
Cell Area:            209229.389447
Design Area:          364588.354865
```

Single Cycle:
Critical Path Length: 6.83
Critical Path Slack: -5.95

Compared to the single cycle processor, the pipelined processor has a shorter critical path and a overall, better performance.