# 3-Stage Pipelined RISC Processor

I have successfully implemented a **3-stage pipelined RISC processor**, consisting of the following pipeline stages:

1. **Instruction Fetch (IF)**
2. **Decode & Execute (ID/EXE)**
3. **Memory & Writeback (MEM/WB)**

All modules were developed separately in Verilog, and the complete processor was integrated and simulated on **Modelsim**.

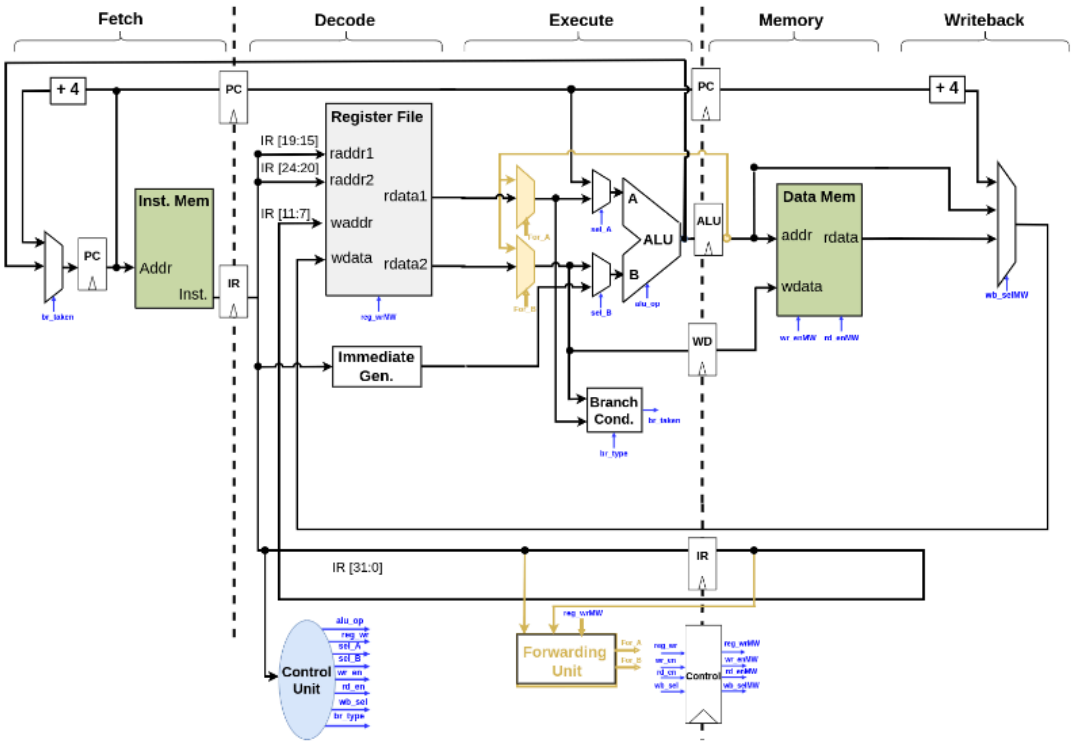
---

## 1. Processor Architecture & Modules

The pipelined datapath is created by inserting pipeline registers between IF → ID/EXE and ID/EXE → MEM/WB stages.
Individual modules implemented include:

- Program Counter
- Instruction Memory
- Register File
- ALU
- Control Unit
- Immediate Generator
- Pipeline Registers
- **Forwarding Unit** (for resolving RAW data hazards)

**Figure 1:3-Stage Pipelined Processor Block Diagram**




---

## 2. Data Hazard Handling (Forwarding Only)

I implemented **forwarding** between the MEM/WB stage and the Decode/Execute stage to resolve simple **read-after-write (RAW)** hazards.

**Forwarding Features:**

- Detects when destination register of previous instruction matches source register of current instruction
- Automatically forwards updated data to ID/EXE stage inputs
- Removes the need for stalling in many common arithmetic dependencies

---

## 3. Simulation & Verification (ModelsimS)

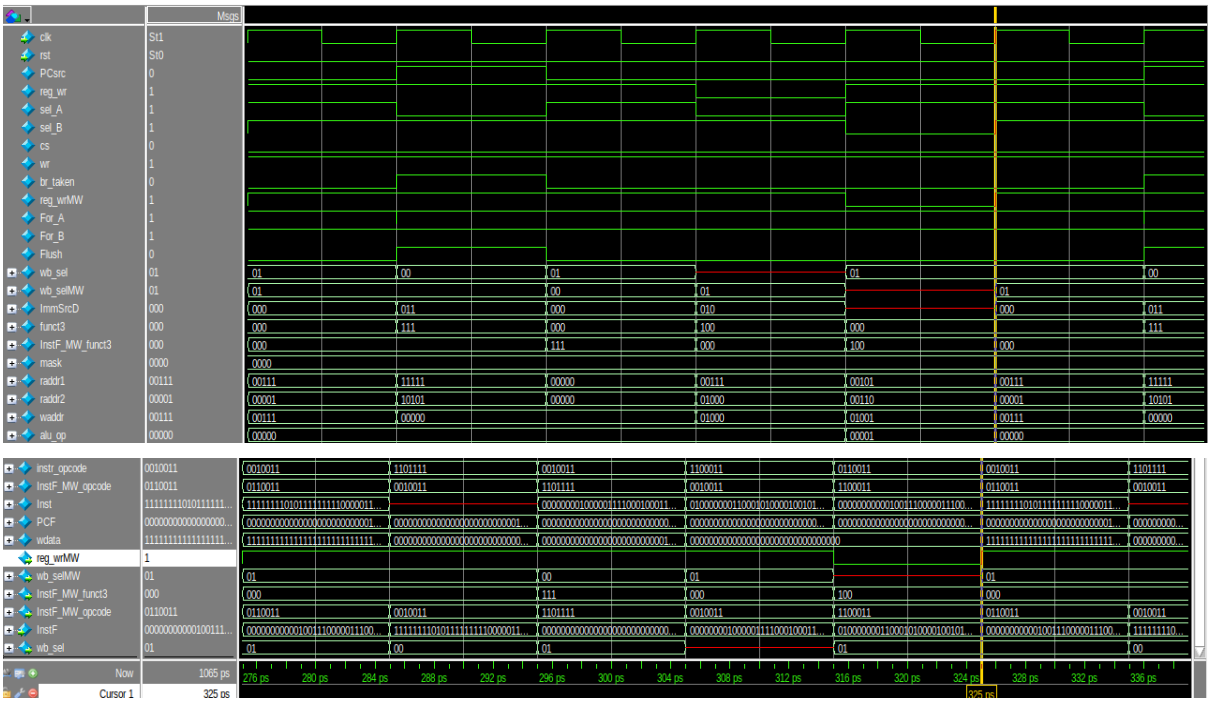
The complete pipelined CPU was simulated using QuestaSim.
Forwarding operation was tested using dependent instructions such as:

add x3, x1, x2

add x4, x3, x5  // dependent on x3 → forwarding required

The waveform confirms correct forwarding from MEM/WB back to the EXE stage.

**Figure 2 – Simulation Waveform**




## 4. FPGA/Vivado Work

Vivado synthesis, implementation have been completed. Below i  have attached the resources utilization report and schematic of the project.

---

**Schematic of the processor:**

Below is the schematic of the RTL generated from Xilinx Vivado.

**Resources utilization report:**

The Slice Logic section details the usage of the fundamental programmable resources: **LUTs** (for logic) and **Registers** (for storage).

1. Slice Logic

| Site Type | Used | Fixed | Available | Util% |
|---|---|---|---|---|
| Slice LUTs* | 1350 | 0 | 63400 | 2.13 |
| LUT as Logic | 838 | 0 | 63400 | 1.32 |
| LUT as Memory | 512 | 0 | 19000 | 2.69 |
| LUT as Distributed RAM | 512 | 0 | | |
| LUT as Shift Register | 0 | 0 | | |
| Slice Registers | 734 | 0 | 126800 | 0.58 |
| Register as Flip Flop | 690 | 0 | 126800 | 0.54 |
| Register as Latch | 44 | 0 | 126800 | 0.03 |
| F7 Muxes | 320 | 0 | 31700 | 1.01 |
| F8 Muxes | 160 | 0 | 15850 | 1.01 |

1.1 Summary of Registers by Type

| Total | Clock Enable | Synchronous | Asynchronous |
|---|---|---|---|
| 0 | _ | - | - |
| 0 | _ | - | Set |
| 0 | _ | - | Reset |
| 0 | _ | Set | - |
| 0 | _ | Reset | - |
| 0 | Yes | - | - |
| 0 | Yes | - | Set |
| 76 | Yes | - | Reset |
| 0 | Yes | Set | - |
| 658 | Yes | Reset | - |

4. IO and GT Specific

| Site Type | Used | Fixed | Available | Util% |
|---|---|---|---|---|
| Bonded IOB | 210 | 0 | 210 | 100.00 |
| Bonded IPADs | 0 | 0 | 2 | 0.00 |
| PHY_CONTROL | 0 | 0 | 6 | 0.00 |
| PHASER_REF | 0 | 0 | 6 | 0.00 |
| OUT_FIFO | 0 | 0 | 24 | 0.00 |
| IN_FIFO | 0 | 0 | 24 | 0.00 |
| IDELAYCTRL | 0 | 0 | 6 | 0.00 |
| IBUFDS | 0 | 0 | 202 | 0.00 |
| PHASER_OUT/PHASER_OUT_PHY | 0 | 0 | 24 | 0.00 |
| PHASER_IN/PHASER_IN_PHY | 0 | 0 | 24 | 0.00 |
| IDELAYE2/IDELAYE2_FINEDELAY | 0 | 0 | 300 | 0.00 |
| ILOGIC | 0 | 0 | 210 | 0.00 |
| OLOGIC | 0 | 0 | 210 | 0.00 |

**IO and Clocking:**

This section shows how the physical interfaces and clock signals are managed.

5. Clocking

| Site Type | Used | Fixed | Available | Util% |
|---|---|---|---|---|
| BUFGCTRL | 2 | 0 | 32 | 6.25 |
| BUFIO | 0 | 0 | 24 | 0.00 |
| MMCME2_ADV | 0 | 0 | 6 | 0.00 |
| PLLE2_ADV | 0 | 0 | 6 | 0.00 |
| BUFMRCE | 0 | 0 | 12 | 0.00 |
| BUFHCE | 0 | 0 | 96 | 0.00 |
| BUFR | 0 | 0 | 24 | 0.00 |

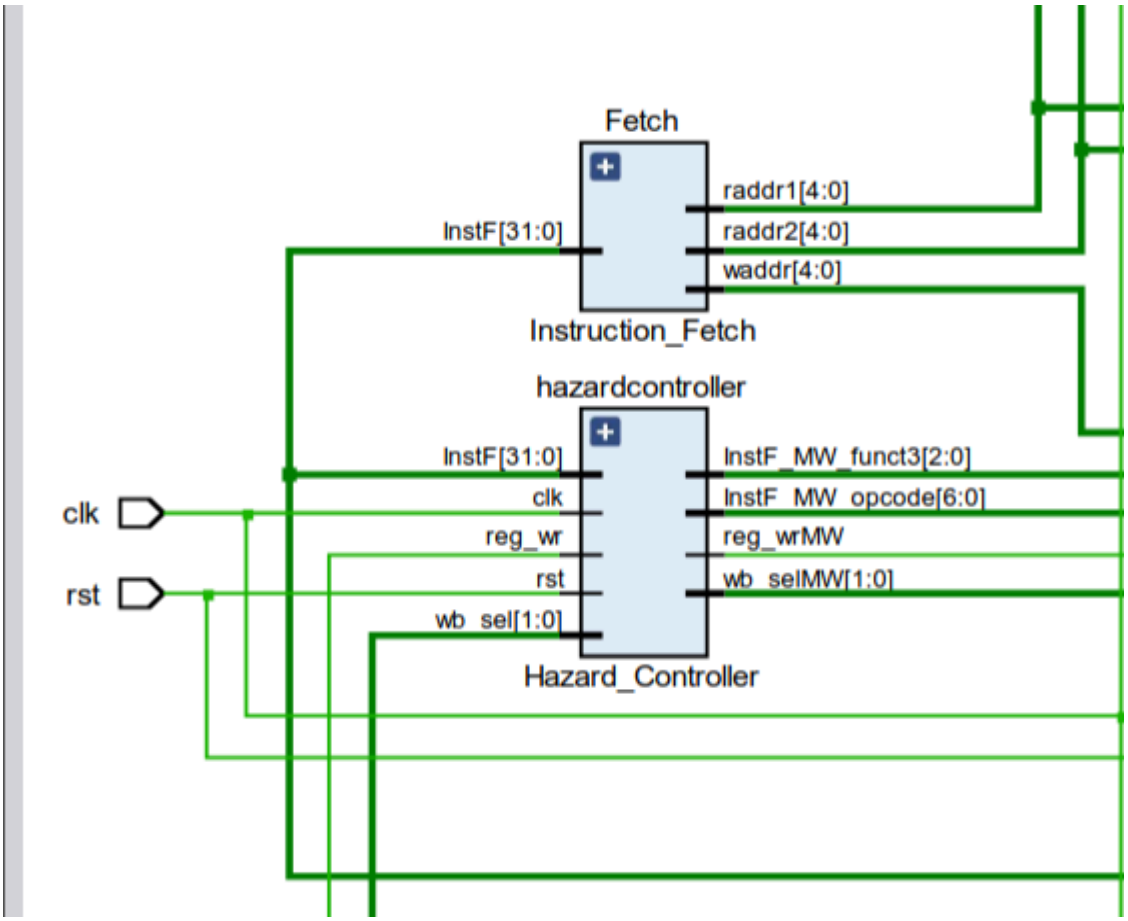
Primitive Used:

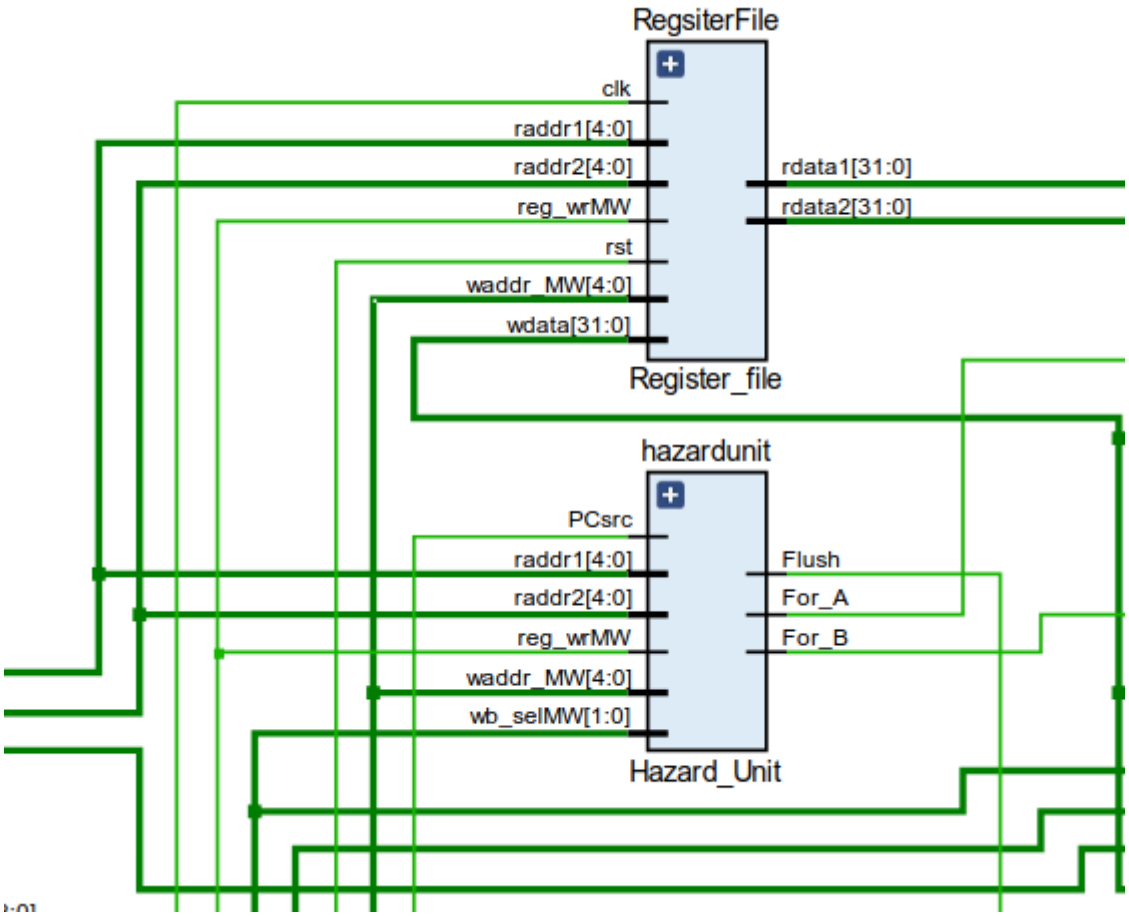
The Primitives table lists the fundamental FPGA building blocks inferred by the Synthesis tool from our SystemVerilog code:

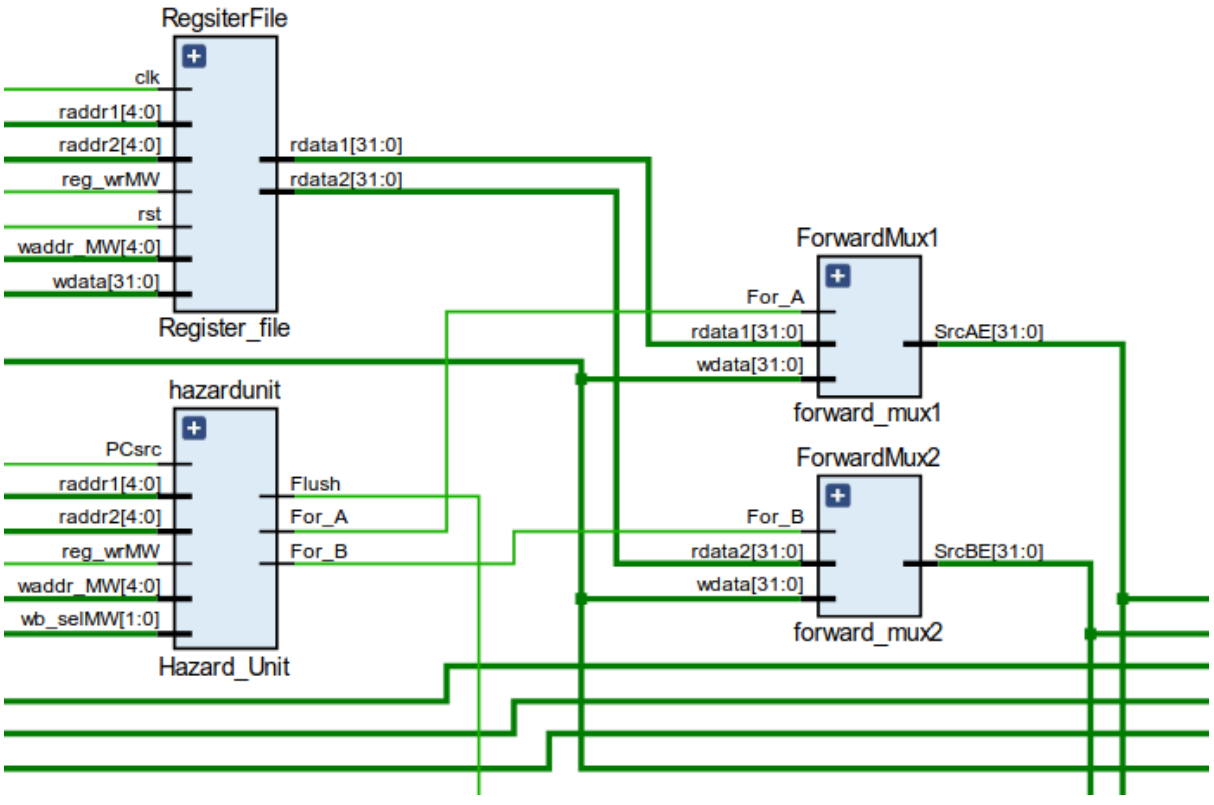
7. Primitives

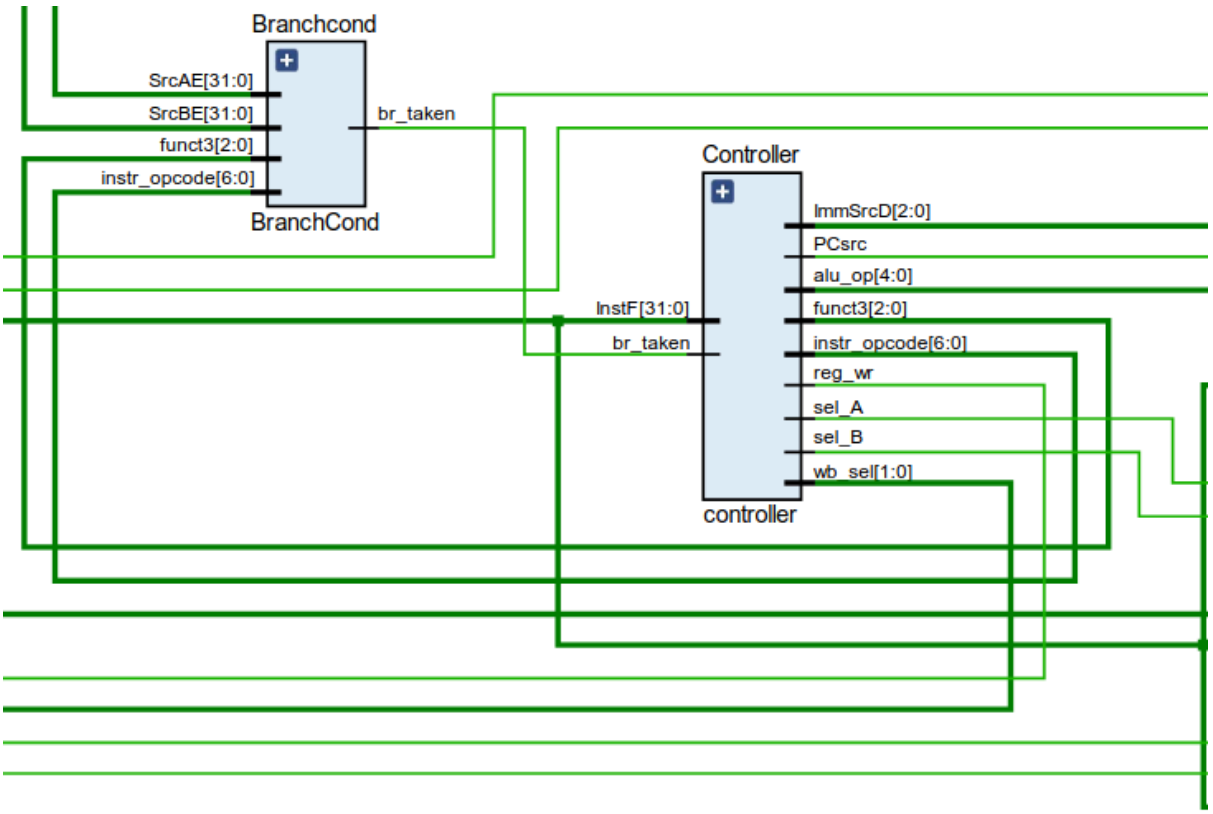
| Ref Name | Used | Functional Category |
|---|---|---|
| FDRE | 658 | Flop & Latch |
| RAMS64E | 512 | Distributed Memory |
| LUT6 | 347 | LUT |
| MUXF7 | 320 | MuxFx |
| OBUF | 208 | IO |
| LUT4 | 208 | LUT |
| MUXF8 | 160 | MuxFx |
| LUT5 | 150 | LUT |
| LUT2 | 99 | LUT |
| LUT3 | 92 | LUT |
| LDCE | 44 | Flop & Latch |
| CARRY4 | 40 | CarryLogic |
| FDCE | 32 | Flop & Latch |
| LUT1 | 30 | LUT |
| IBUF | 2 | IO |
| BUFG | 2 | Clock |

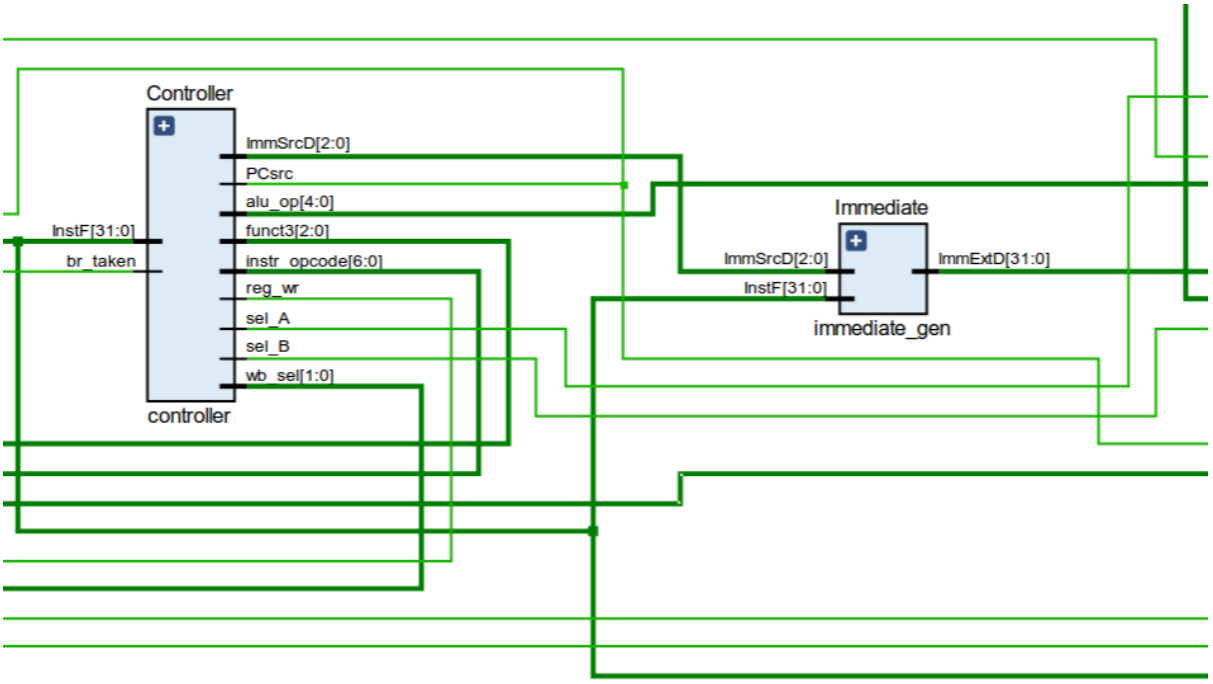
**Schematics of Processor:**

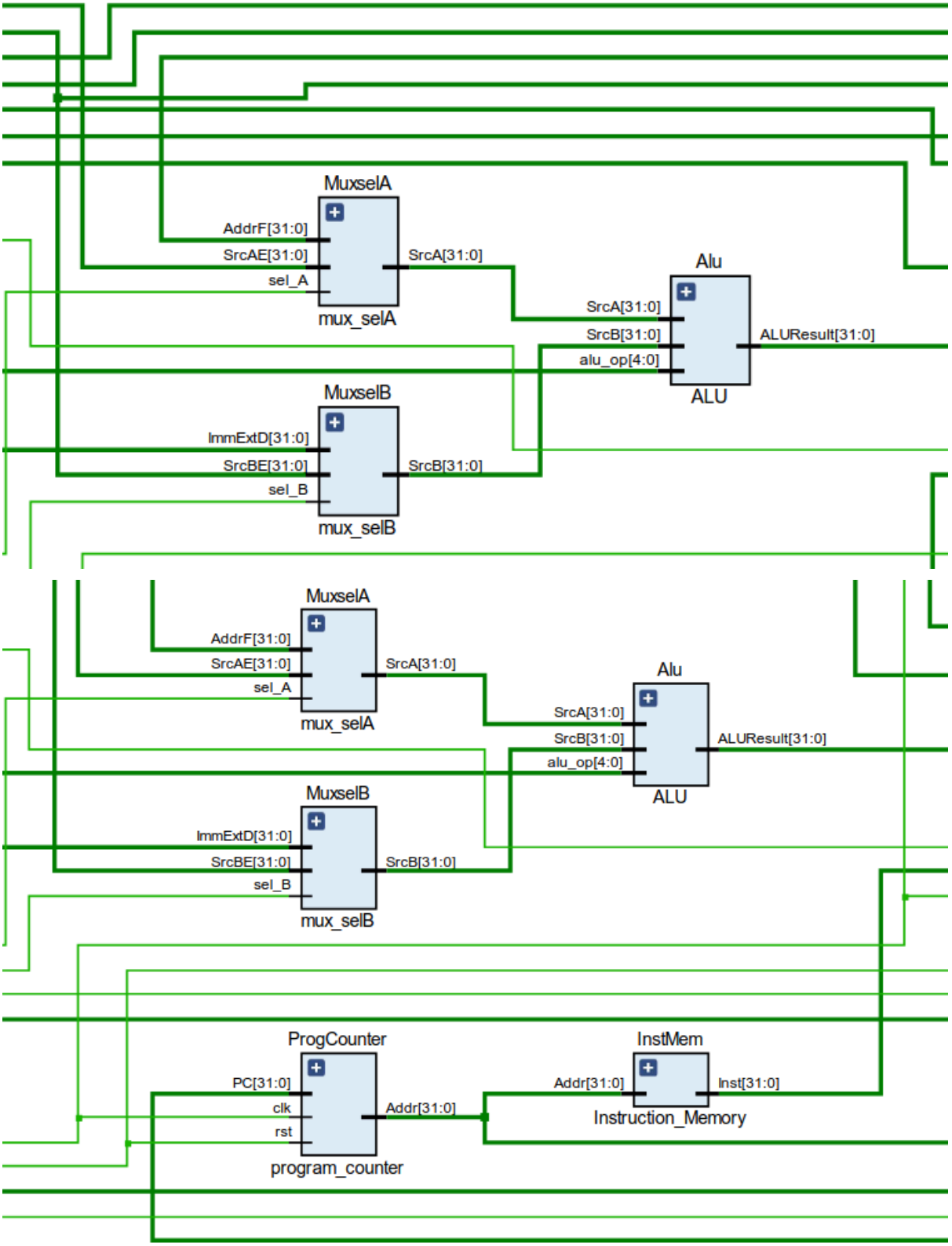


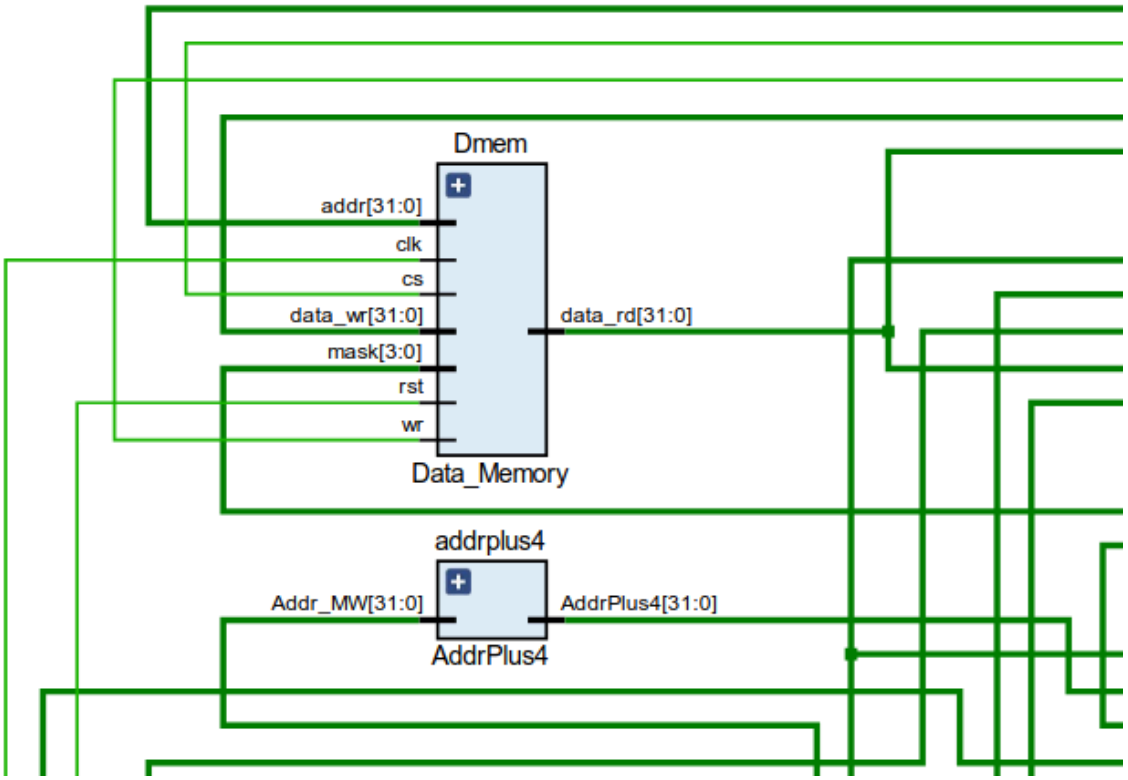

RegsiterFile
clk
raddr1[4:0]
raddr2[4:0]
reg_wrMW
rst
waddr_MW[4:0]
wdata[31:0]
rdata1[31:0]
rdata2[31:0]
Register_file
hazardunit
PCsrc
raddr1[4:0]
raddr2[4:0]
reg_wrMW
waddr_MW[4:0]
wb_selMW[1:0]
Flush
For_A
For_B
Hazard_Unit

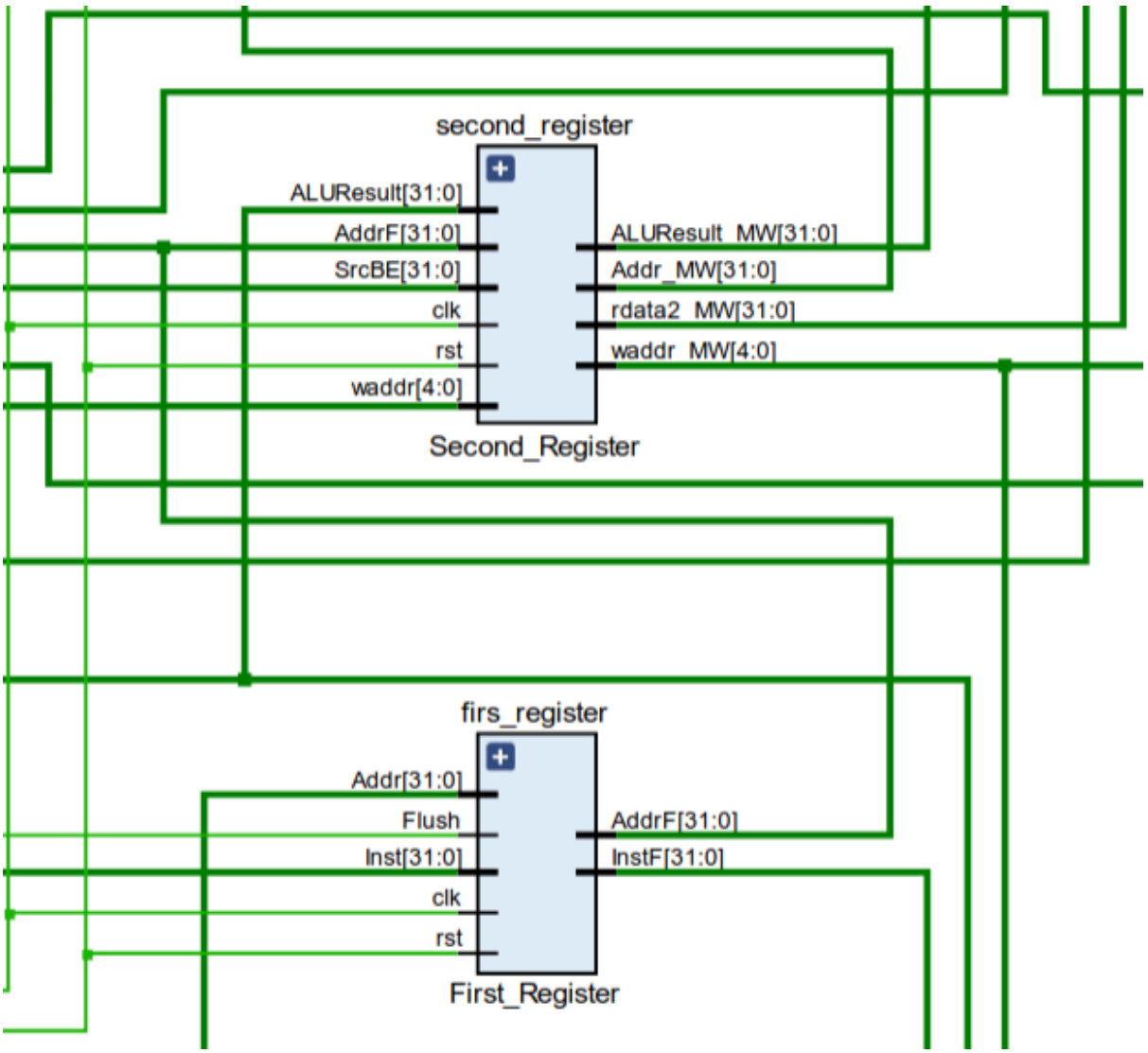

RegsiterFile
clk
raddr1[4:0]
raddr2[4:0]
reg_wrMW
rst
waddr_MW[4:0]
wdata[31:0]
rdata1[31:0]
rdata2[31:0]
Register_file
ForwardMux1
For_A
rdata1[31:0]
wdata[31:0]
SrcAE[31:0]
forward_mux1
hazardunit
PCsrc
raddr1[4:0]
raddr2[4:0]
reg_wrMW
waddr_MW[4:0]
wb_selMW[1:0]
Flush
For_A
For_B
Hazard_Unit
ForwardMux2
For_B
rdata2[31:0]
wdata[31:0]
SrcBE[31:0]
forward_mux2

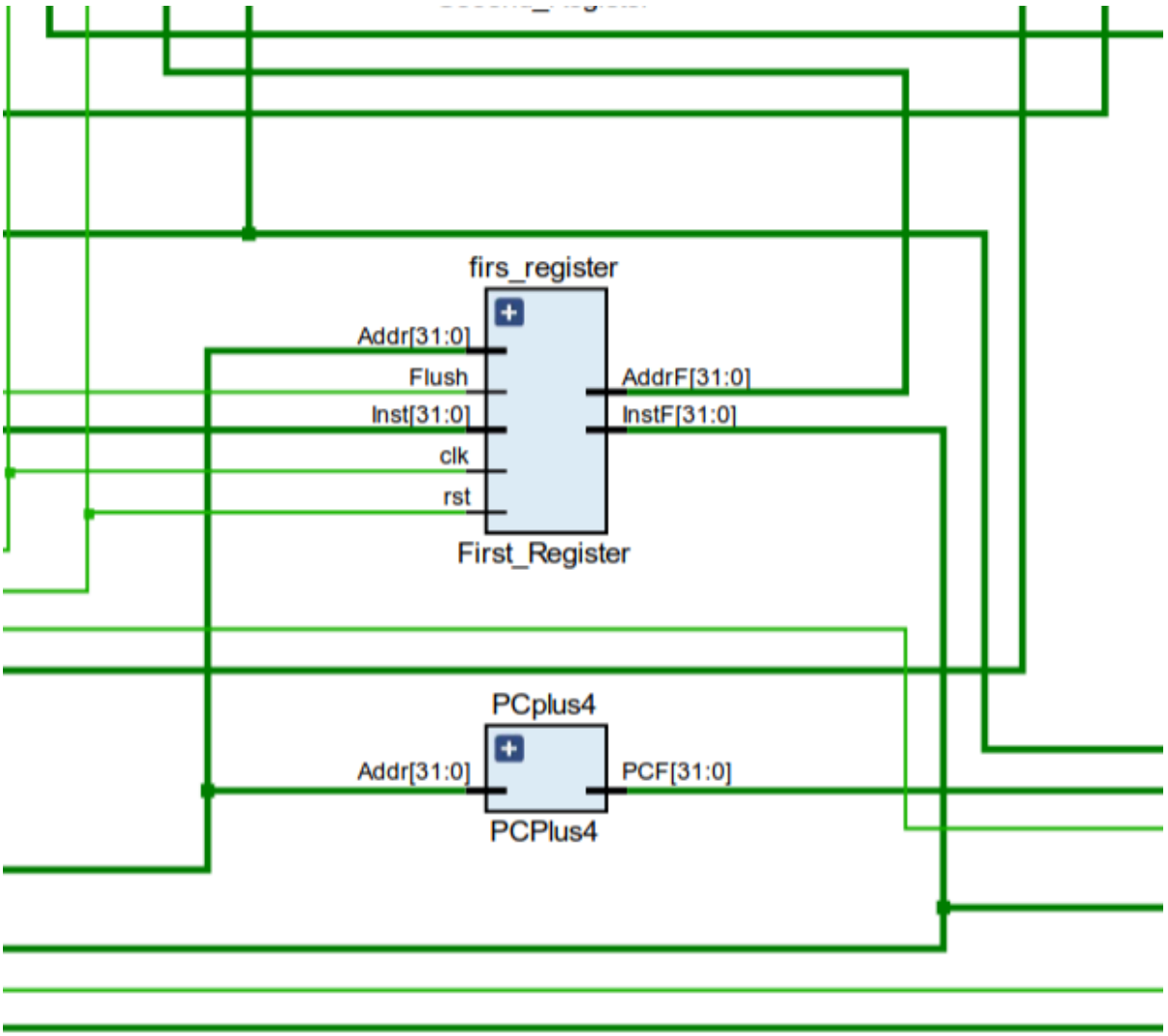

Branchcond
SrcAE[31:0]
SrcBE[31:0]
funct3[2:0]
instr_opcode[6:0]
br_taken
BranchCond
Controller
InstF[31:0]
br_taken
ImmSrcD[2:0]
PCsrc
alu_op[4:0]
funct3[2:0]
instr_opcode[6:0]
reg_wr
sel_A
sel_B
wb_sel[1:0]
controller

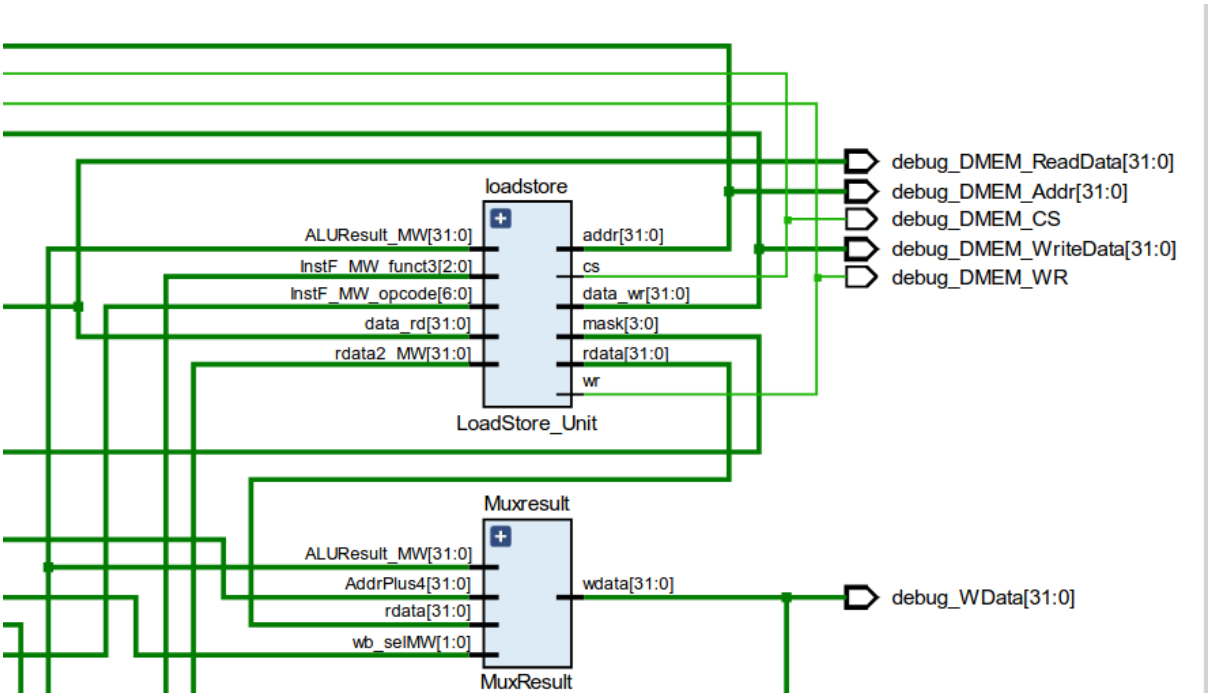

Controller
InstF[31:0]
br_taken
ImmSrcD[2:0]
PCsrc
alu_op[4:0]
funct3[2:0]
instr_opcode[6:0]
reg_wr
sel_A
sel_B
wb_sel[1:0]
controller
Immediate
ImmSrcD[2:0]
InstF[31:0]
ImmExtD[31:0]
immediate_gen

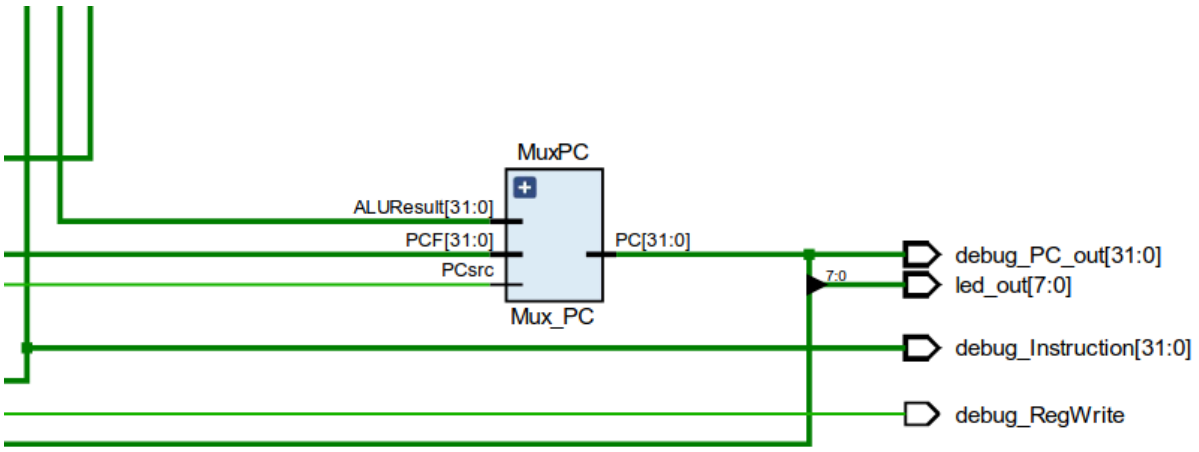

MuxselA
AddrF[31:0]
SrcAE[31:0]
sel_A
SrcA[31:0]
mux_selA
Alu
SrcA[31:0]
SrcB[31:0]
alu_op[4:0]
ALUResult[31:0]
ALU
MuxselB
ImmExtD[31:0]
SrcBE[31:0]
sel_B
SrcB[31:0]
mux_selB
MuxselA
AddrF[31:0]
SrcAE[31:0]
sel_A
SrcA[31:0]
mux_selA
Alu
SrcA[31:0]
SrcB[31:0]
alu_op[4:0]
ALUResult[31:0]
ALU
MuxselB
ImmExtD[31:0]
SrcBE[31:0]
sel_B
SrcB[31:0]
mux_selB
ProgCounter
PC[31:0]
clk
rst
Addr[31:0]
program_counter
InstMem
Addr[31:0]
Inst[31:0]
Instruction_Memory


Dmem
addr[31:0]
clk
cs
data_wr[31:0]
mask[3:0]
rst
wr
data_rd[31:0]
Data_Memory
addrplus4
Addr_MW[31:0]
AddrPlus4[31:0]
AddrPlus4


second_register
ALUResult[31:0]
AddrF[31:0]
SrcBE[31:0]
clk
rst
waddr[4:0]
ALUResult_MW[31:0]
Addr_MW[31:0]
rdata2_MW[31:0]
waddr_MW[4:0]
Second_Register
firs_register
Addr[31:0]
Flush
Inst[31:0]
clk
rst
AddrF[31:0]
InstF[31:0]
First_Register


firs_register
Addr[31:0]
Flush
Inst[31:0]
clk
rst
AddrF[31:0]
InstF[31:0]
First_Register
PCplus4
Addr[31:0]
PCF[31:0]
PCPlus4


loadstore
ALUResult_MW[31:0]
InstF_MW_funct3[2:0]
InstF_MW_opcode[6:0]
data_rd[31:0]
rdata2_MW[31:0]
addr[31:0]
cs
data_wr[31:0]
mask[3:0]
rdata[31:0]
wr
LoadStore_Unit
debug_DMEM_ReadData[31:0]
debug_DMEM_Addr[31:0]
debug_DMEM_CS
debug_DMEM_WriteData[31:0]
debug_DMEM_WR
Muxresult
ALUResult_MW[31:0]
AddrPlus4[31:0]
rdata[31:0]
wb_selMW[1:0]
wdata[31:0]
MuxResult
debug_WData[31:0]


MuxPC
ALUResult[31:0]
PCF[31:0]
PCsrc
PC[31:0]
Mux_PC
7:0
debug_PC_out[31:0]
led_out[7:0]
debug_Instruction[31:0]
debug_RegWrite