EC605: Computer Engineering

Lab 5: Single Cycle ARM Datapath

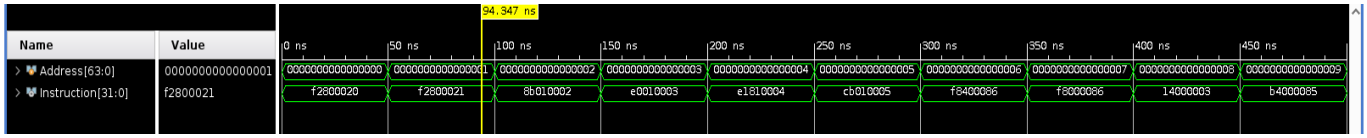
Group member: Xiaoyu An (anxiaoyu@bu.edu)

Yiming Miao (mym1031@bu.edu)

1. Waveform and Description

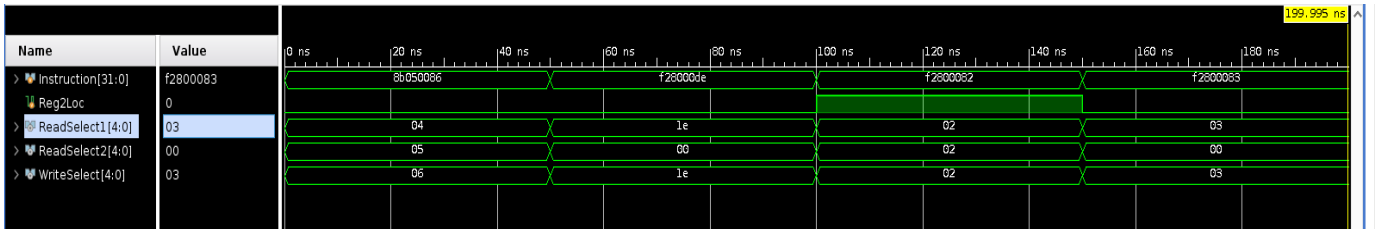
<1> Instruction_Memory

Instruction memory will output the instructions depends on PC address.



<2> MUX_2_1_5

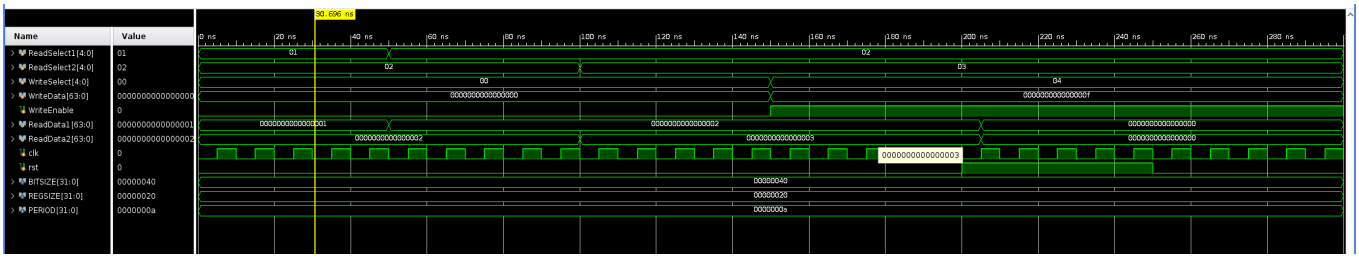
This is a selection of instruction output. The inputs are instruction and Reg2Loc signal, and the output are ReadSelect1, ReadSelect2 and WriteSelect. To realize the MOVK instruction, the Rd should be selected and output at ReadData1 in order to keep [63:48] bit the same. So, the ReadSelect1 will choose Instruction [4:0] when the instruction is MOVK and will select Instruction [9:5] during other instruction.



<3> Register_File

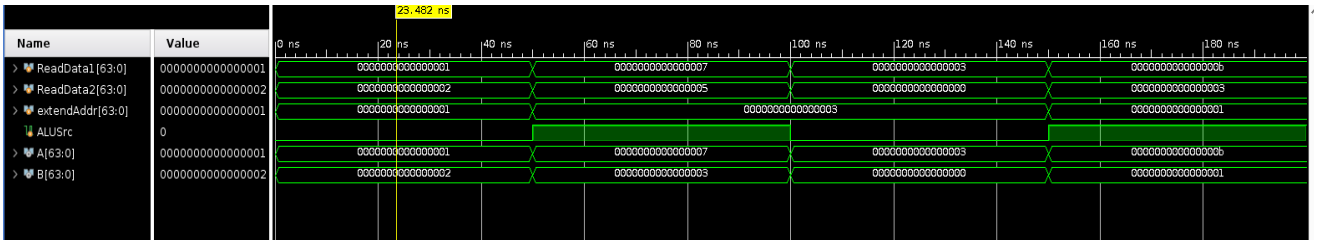
Register file contains 32 register and each size is 64 bits. It has five inputs, the read and write selection, the write data from data memory and a control signal WriteEnable. The write process is

driven by the clk and rst. Also has two outputs ReadData1and ReadData2.



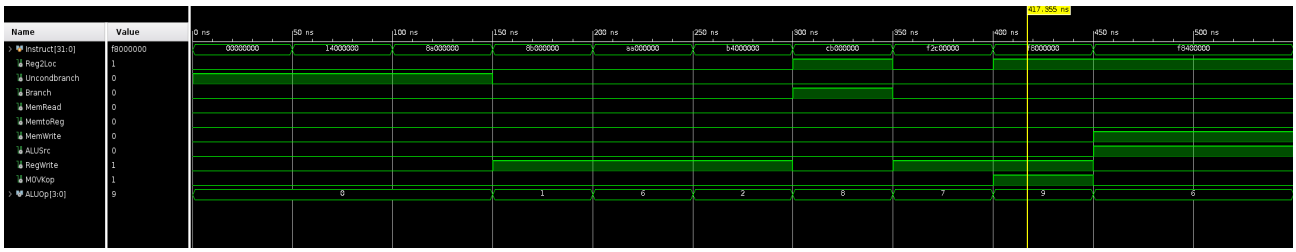
<4> MUX_2_1

This module is used for choosing the data that sent to ALU. It has four inputs ReadData1, ReadData2 from register file, extendAddr from sign extend and ALUSrc from control signal. And two output A and B that sent to A and B.



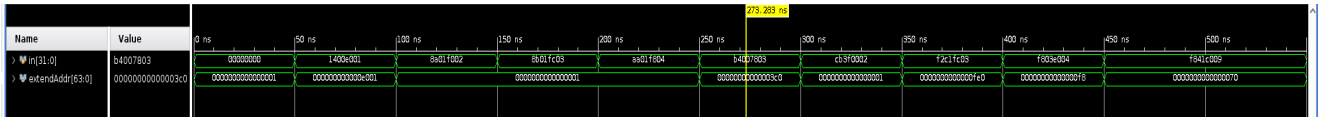
<5> Main_Control

This module is a control unit. The 10 control signals are depending on the first 11 bit of instruction to realize 9 instructions (Branch, And, Add…). The output of main control will be sent to each module as control signal. The control signal should be changed at the same time when instruction changed.



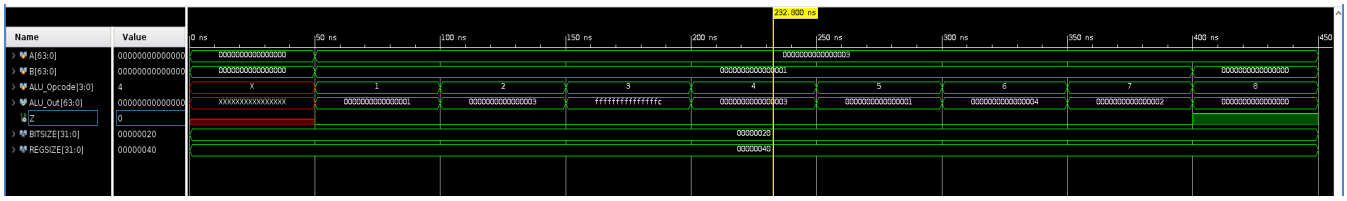
<6> Sign_extend

This module done the sign extend part. There are five instruction should do the sign extend part, BRANCH, CBNZ, MOVK, LDUR and STUR. This part needs to extend the address or immediate number to 64 bits. And the tricky part is that the Branch and CBNZ instruction should realize the function that jump back in the address. So, the sign extend part should also consider the situation that when Instruction [20] is equal to one.



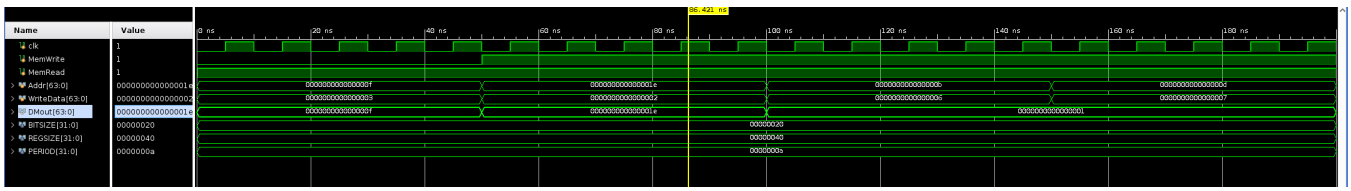
<7> ALU_64

This module is an ALU. The inputs are two data source A and B, the three bits ALU opcode. Outputs are ALU_Out and ZeroFlag. When receive the ALU_Opcode, the ALU can do 8 kinds of calculations. To realize the instruction MOVK, which ALU_Opcode is 111, the output of ALU should A[63:16] combined with B[15:0]. This will keep the first 48 bits of Rd not change and move the immediate number to the last 16 bits of Rd.



<8> Data_memory

This module is data memory, which is controlled by the clk and rst signal. My data memory has 64 memory and each size is 64 bits. Data can be read any time which will be read to DMout when Addr, data_mem or MemRead change. Data is written sequentially. And only can write at the positive edge of clk.

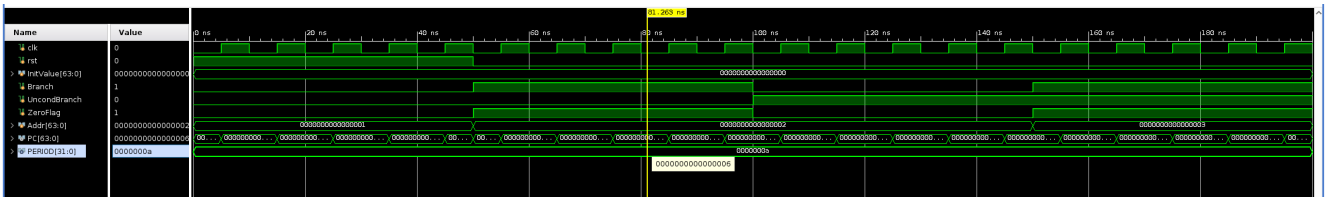


<9> MUX_To_Reg

The MUX_To_Reg will send the output to the register file. It will select between data memory output and ALU output depend on the control signal MemtoReg.

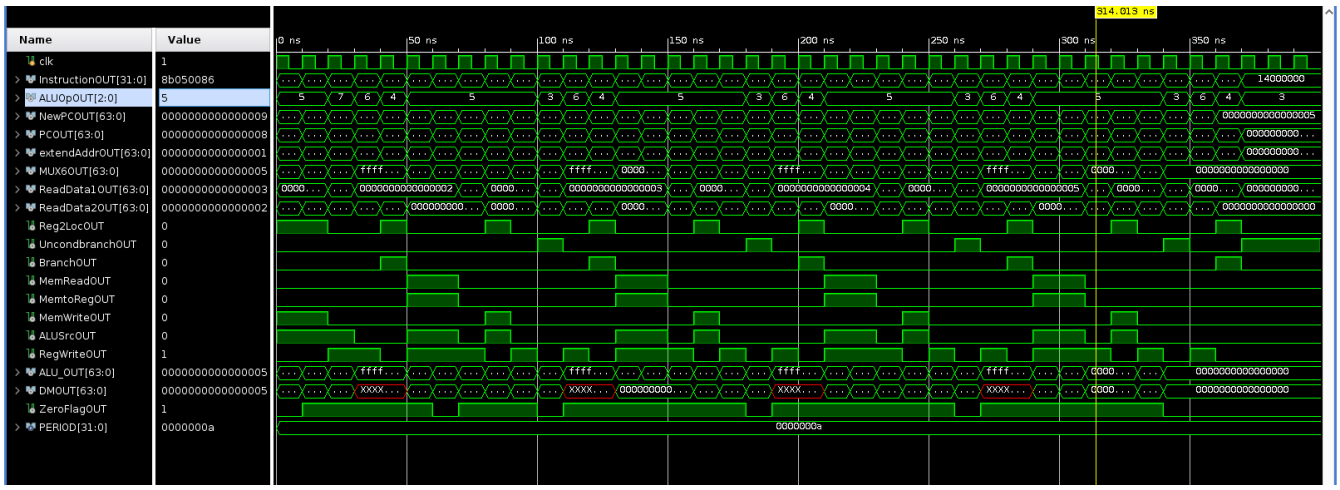
<10> ProgCounter

The PC counter will add address only at CBNZ and Branch instruction, in all other situation PC will directly add 1.



<11> Top_Module

The Top_Module will connect all the module. The PC is defined as a register and will add 1 at positive edge of clk. I use the Fib array in lab4 to test this file and got a correct answer!

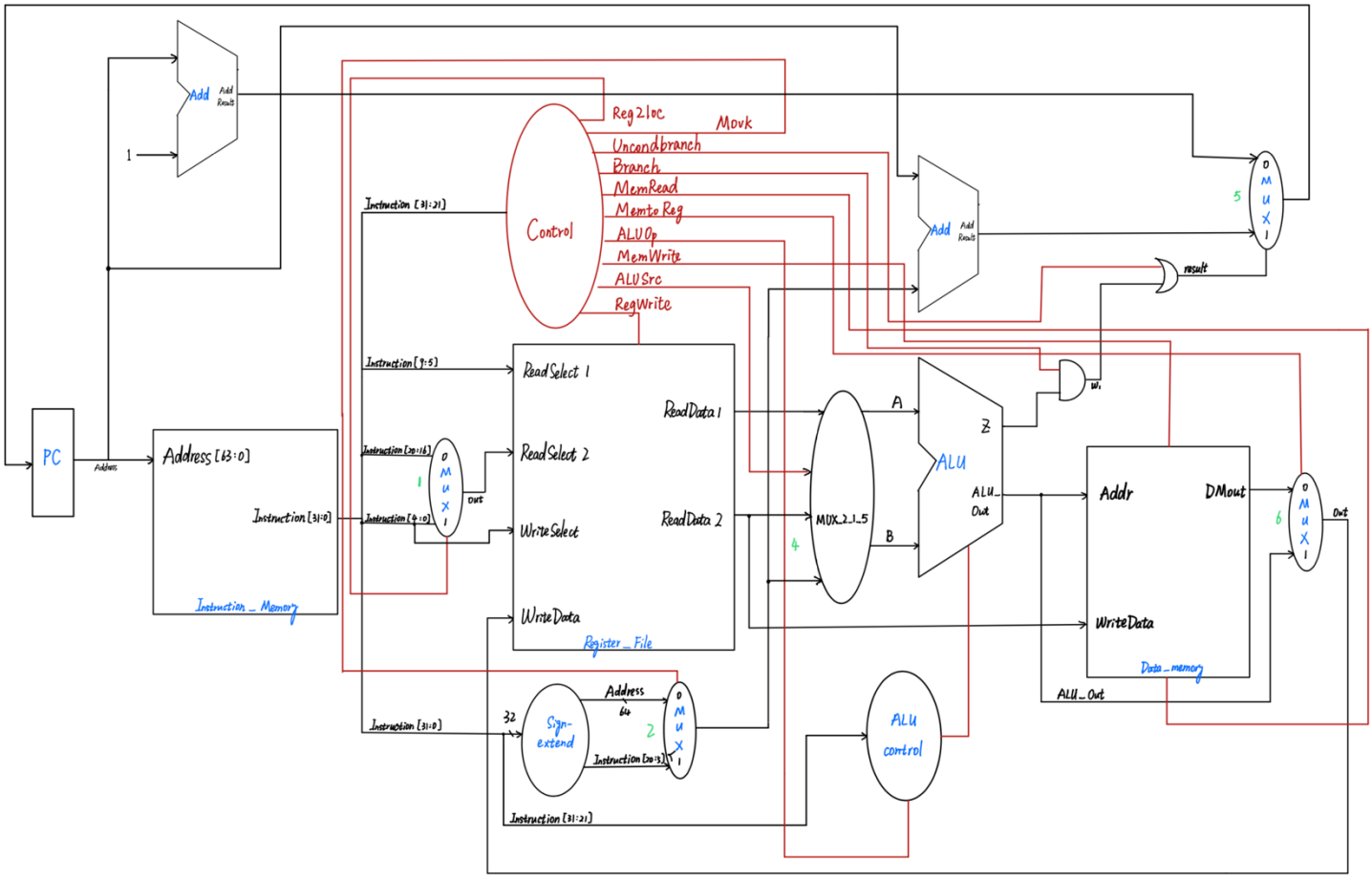


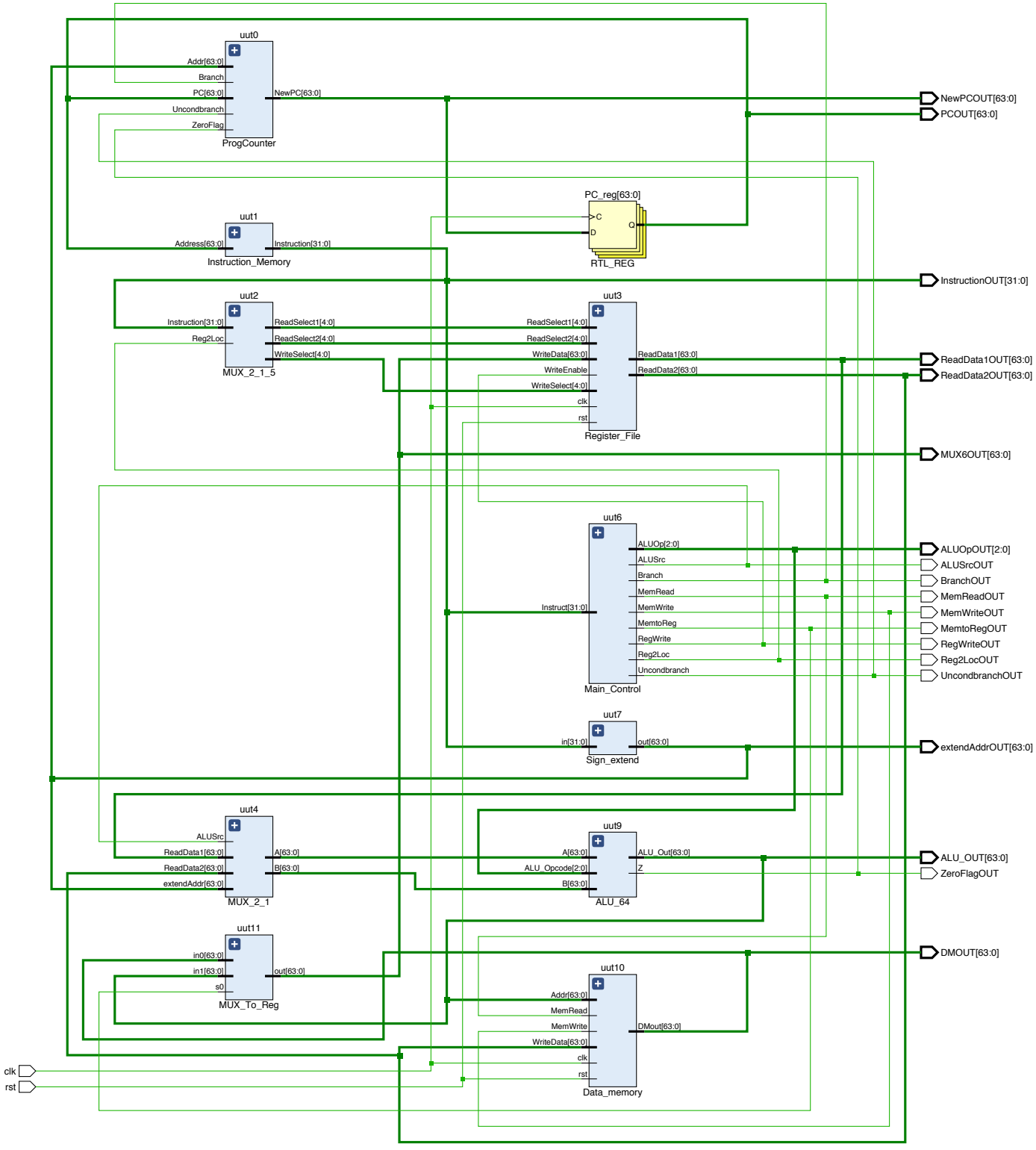
2. Integrated Datapath and MOVK

In milestone 2, we designed the MOVK in a wrong way. Now we design it as follow. Because we cannot read and write at the same time. So, we add a multiplexer MUX_2_1_5 to choose the instruction bit to select register. At MOVK instruction, ReadSelect1 will choose the Rd and read the data of Rd and output to another multiplexer MUX_2_1, this multiplexer will send Rd data to

Alu. At the same time the lowest 16 bit that need to move to Rd will go through sign extend. One of the ALU function will combine the highest 48 bits of Rd data and lowest 16 bits of sign extend and then the output of ALU will be written to register file. Our new block diagram will be attached at the end of lab report.

3. Milestone 2 Block Diagram




uut0
Addr[63:0]
Branch
PC[63:0]
Uncondbranch
ZeroFlag
NewPC[63:0]
ProgCounter
NewPCOUT[63:0]
PCOUT[63:0]
PC_reg[63:0]
C
D
Q
RTL_REG
uut1
Address[63:0]
Instruction[31:0]
Instruction_Memory
InstructionOUT[31:0]
uut2
Instruction[31:0]
Reg2Loc
ReadSelect1[4:0]
ReadSelect2[4:0]
WriteSelect[4:0]
MUX_2_1_5
uut3
ReadSelect1[4:0]
ReadSelect2[4:0]
WriteData[63:0]
WriteEnable
WriteSelect[4:0]
clk
rst
ReadData1[63:0]
ReadData2[63:0]
Register_File
ReadData1OUT[63:0]
ReadData2OUT[63:0]
MUX6OUT[63:0]
uut6
Instruct[31:0]
ALUOp[2:0]
ALUSrc
Branch
MemRead
MemWrite
MemtoReg
RegWrite
Reg2Loc
Uncondbranch
Main_Control
ALUOpOUT[2:0]
ALUSrcOUT
BranchOUT
MemReadOUT
MemWriteOUT
MemtoRegOUT
RegWriteOUT
Reg2LocOUT
UncondbranchOUT
uut7
in[31:0]
out[63:0]
Sign_extend
extendAddrOUT[63:0]
uut4
ALUSrc
ReadData1[63:0]
ReadData2[63:0]
extendAddr[63:0]
A[63:0]
B[63:0]
MUX_2_1
uut9
A[63:0]
ALU_Opcode[2:0]
B[63:0]
ALU_Out[63:0]
Z
ALU_64
ALU_OUT[63:0]
ZeroFlagOUT
uut11
in0[63:0]
in1[63:0]
s0
out[63:0]
MUX_To_Reg
DMOUT[63:0]
uut10
Addr[63:0]
MemRead
MemWrite
WriteData[63:0]
clk
rst
DMout[63:0]
Data_memory
clk
rst