UM-SJTU Joint Institution

VE370 Introduction to Computer Organization

Project Report 2

Prepared by Team 05

Chen Yinfan (5123709275)

Song Shiwei (5123709279)

Wei Yi (5123709286)

Date: November 19, 2014

Contents

Introduction

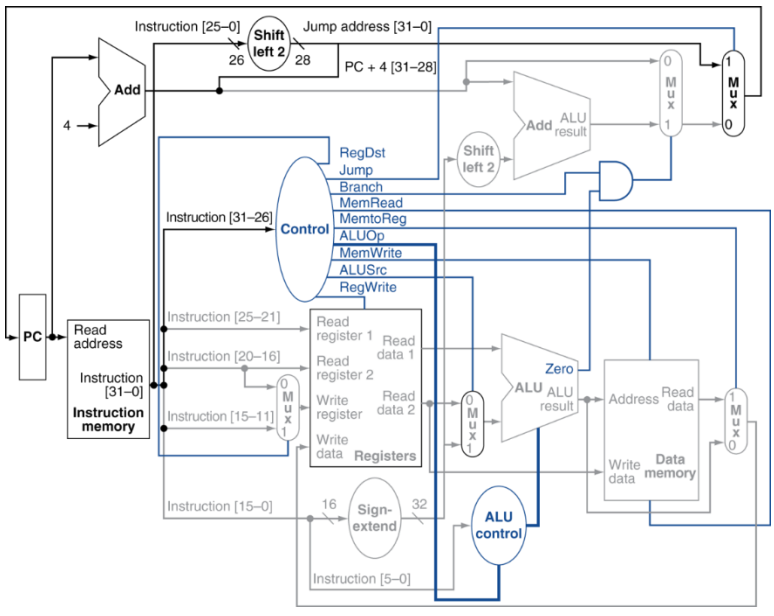
The central processing unit (CPU) is one of the most significant components of computer. The major function of CPU is to interpret instructions and process the data. We have learnt how to program in MIPS code and translate it to machine code.

In this project, we are required to implement a simple prototype of processer to realize some simple functions. There are 3 types of MIPS instructions. One is memory-reference instructions, like load word (lw) and store word (sw). Another is arithmetic-logical instructions like add, addi, sub, and, andi, or, and slt. Also the jumping instructions such as branch equal (beq) and jump (j).

As we have learned in the course, we could have designed a circuit which complete these procedures in one clock cycle. However, it waste a lot of time. So we would like to implement it in a pipeline way to make it more time-efficiency. As a consequence, pipeline processor will cause kinds of data hazard and control hazard. We will think over these hazards and obtain solutions in our design.

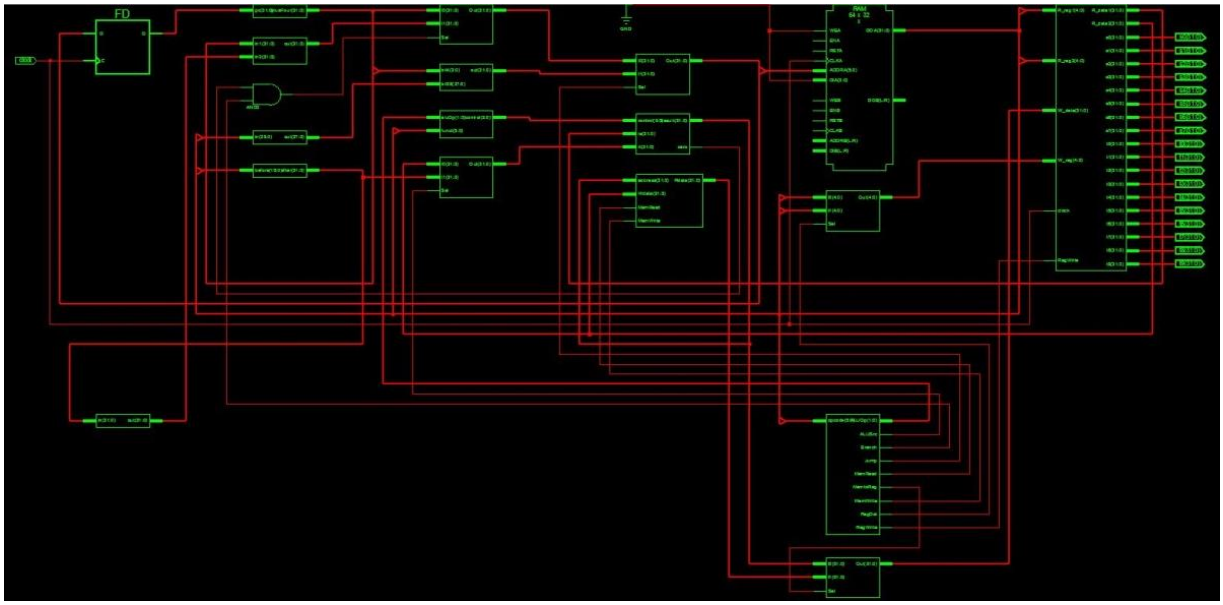
Design (Single Cycle)

For single-cycle part, we refer to the figure as following, which is given in the project-two description. We first build several blocks, such as PC, add, ALU Control , ALU, connect, control, Data Memory, Instruction Memory, register files, several mux and so on. We also value the Instruction Memory and build a test bench to test its functions.



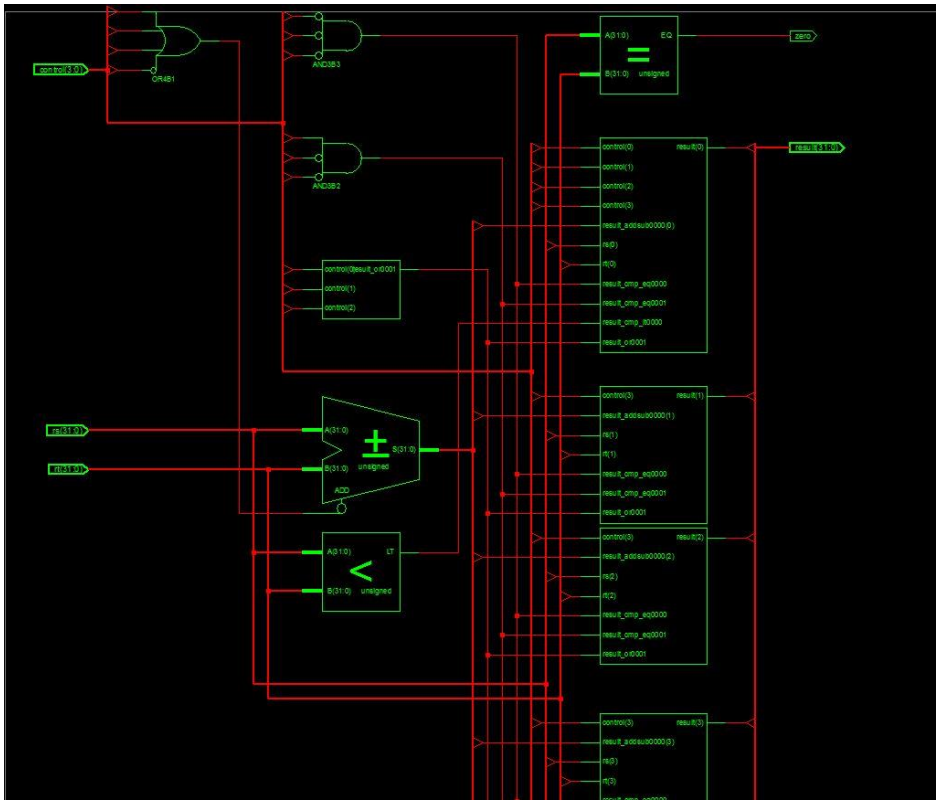
Implementation & RTL Schematic (Single Cycle)

The RTL schematic for single-cycle



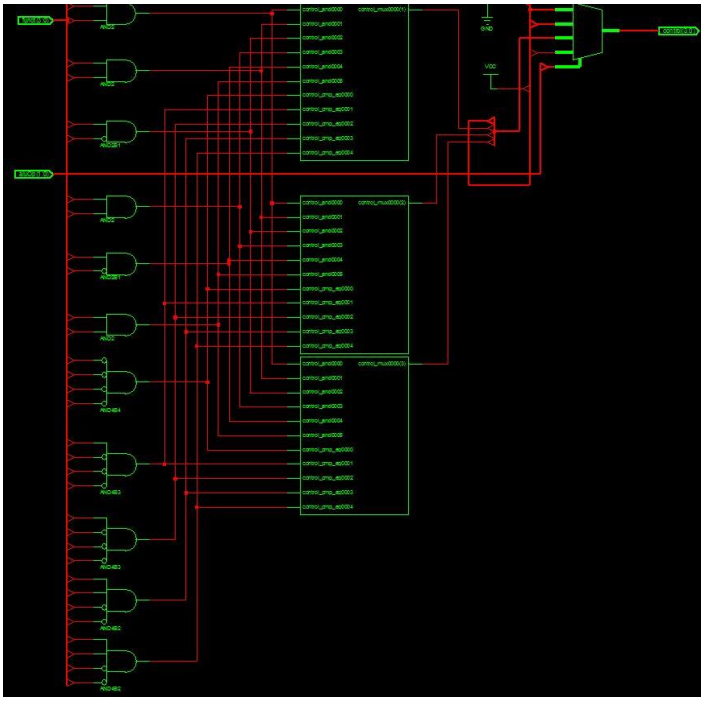
ALU

The following is RTL schematic of ALU. The specific code is attached in appendix. It has three inputs and two outputs. The inputs are 32 bits Rs, 32 bits Rt(or immediate number) and 4 bits control. It outputs whether Rt and Rs are equal and the calculation result.



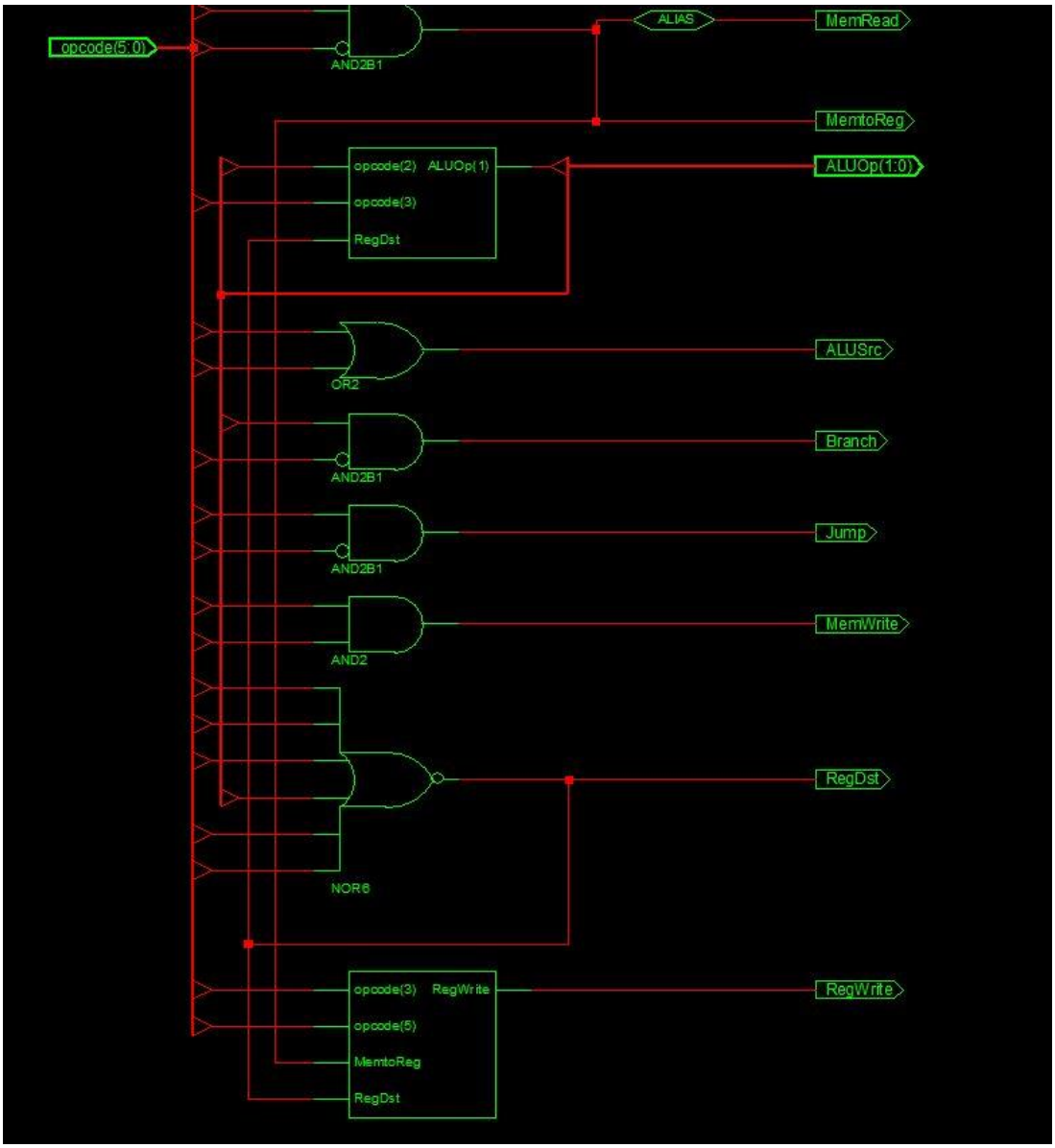
ALU Control

The following is RTL schematic of ALU control. The specific code is attached in appendix. It has two inputs and one output. The inputs are 2 bits ALUOp from control and last 6 bits of instruction. It outputs 6 bits funct.



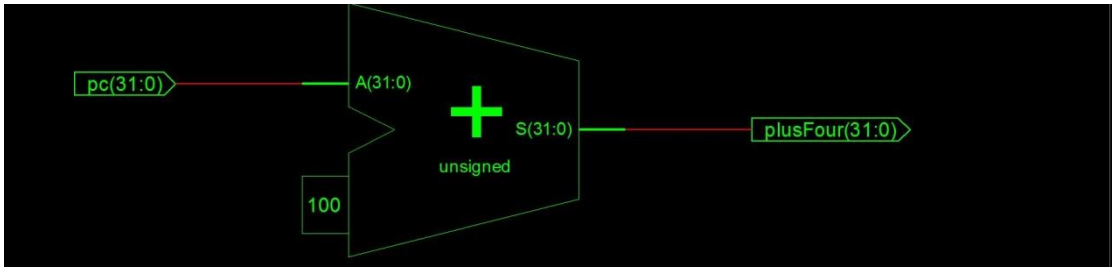
Control

The following is RTL schematic of control. The specific code is attached in appendix. It has one input and nine outputs. The inputs are first six bits of instruction. The outputs includes 8 one-bit control signal (jump, branch and so on) and 2 bits ALUOp



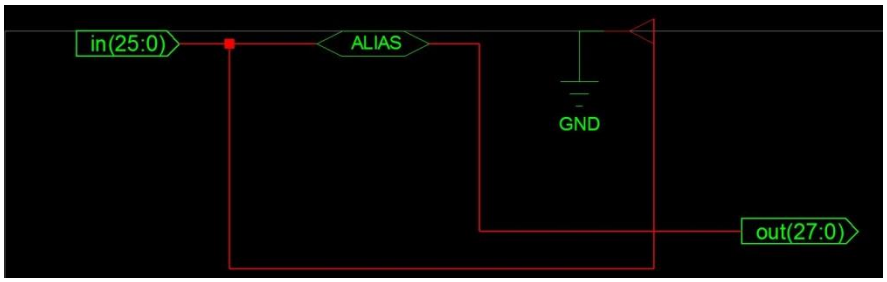
PC+4

The following is RTL schematic of pc+4. The specific code is attached in appendix. It is used to add four to present pc address.



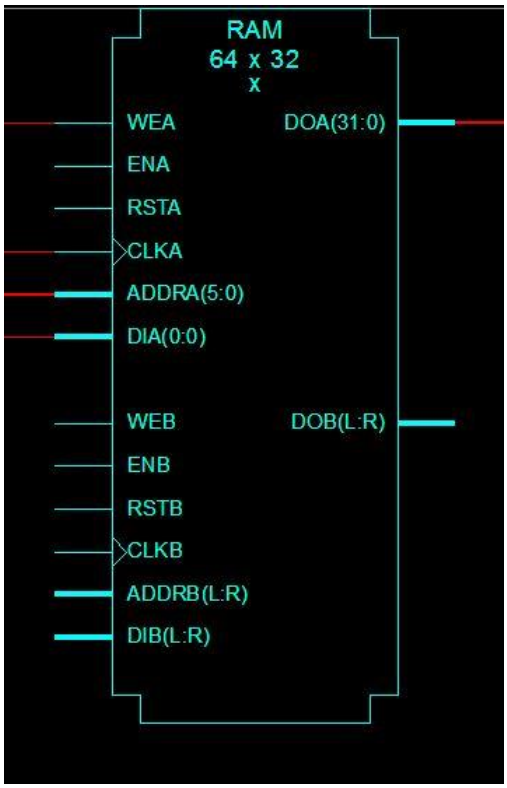
Shift 2

The following is RTL schematic of shift 2. The specific code is attached in appendix. It is used to get four times address.(from 26 bits to 28 bits)



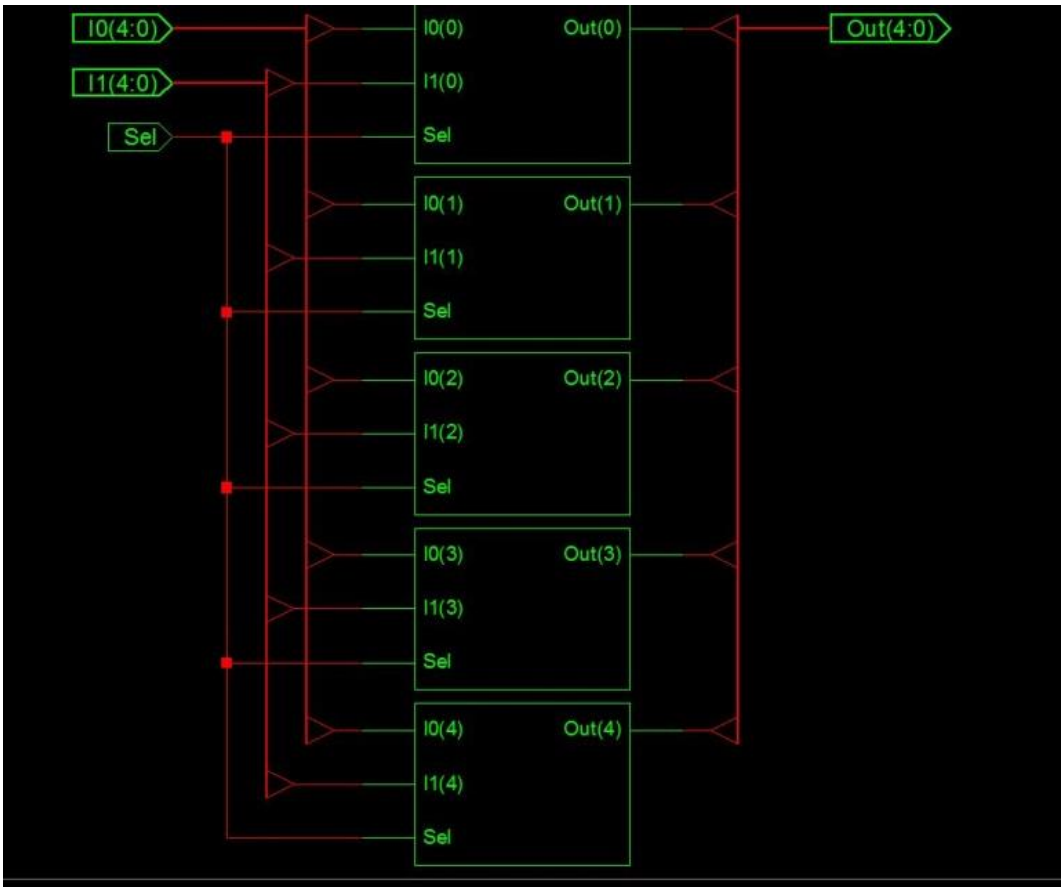
IM

The following is RTL schematic of RAM. The specific code is attached in appendix. It includes Instruction Memory and we use it to initialize the memory to test and implement.



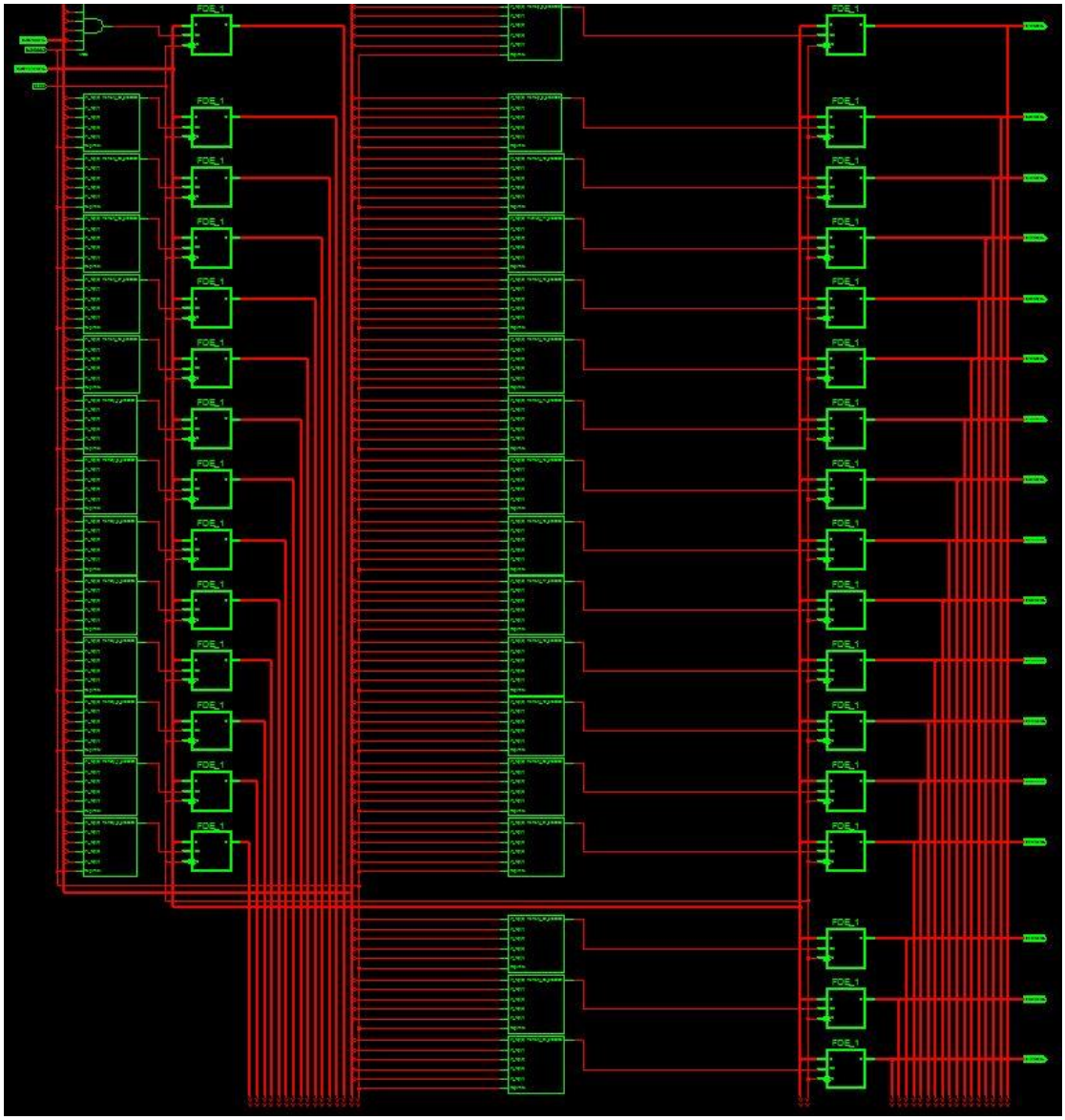
4-to-1 Mux

The following is RTL schematic of 4-to-1 mux. The specific code is attached in appendix. It is representative of multiple mux of different bits we use.



Register File

The following is RTL schematic of register files, one general graph and one specific graph. The specific code is attached in appendix. The block is used to read data stored in specified registers and write data to specified one. It writes back at negative edge of clock. We also add outputs (18 memory registers) in this block for test.



Simulation Result & Demonstration (Single Cycle)

The instruction we use: same with given in specification

Imemory[0] = 32'b00100000000010000000000000100000; //addi $t0, $zero, 32
Imemory[1] = 32'b00100000000010010000000000110111; //addi $t1, $zero, 55
Imemory[2] = 32'b00000001000010011000000000100000; //add $s0, $t0, $t1
Imemory[3] = 32'b00000001000010011000100000100010; //sub $s1, $t0, $t1
Imemory[4] = 32'b00000001000010011001000000100100; //and $s2, $t0, $t1
Imemory[5] = 32'b00000001000010011001100000100101; //or $s3, $t0, $t1
Imemory[6] = 32'b00000001000010011010000000101010; //slt $s4, $t0, $t1 (LOOP)
Imemory[7] = 32'b00010010100000000000000000000110; //beq $s4, $zero, EXIT
Imemory[8] = 32'b00000001000010010101000000100000; //add $t2, $t0, $t1
Imemory[9] = 32'b00000001010010010101000000100000; //add $t2, $t2, $t1
Imemory[10] = 32'b00000001010010000101100000100000; //add $t3, $t2, $t0

```
Imemory[11] = 32'b10001100000101010000000000000100; //lw $s5, 4($zero)
Imemory[12] = 32'b00000001010101011011000000100000; //add $s6, $t2, $s5
Imemory[13] = 32'b00000001010101011101100000100000; //add $s7, $s5, $t3
Imemory[14] = 32'b10101100000101010000000000001000; //sw $s5, 8($zero) (EXIT)
Imemory[15] = 32'b00001000000000000000000000000110; //j LOOP
```

The complete result is attached in Appendix. We choose several instructions to see whether our model is right.

After cycle one, the result is

[$s0] = 00000000 [$s1] = 00000000 [$s2] = 00000000

[$s3] = 00000000 [$s4] = 00000000 [$s5] = 00000000

[$s6] = 00000000 [$s7] = 00000000 [$t0] = 00000020

[$t1] = 00000000 [$t2] = 00000000 [$t3] = 00000000

[$t4] = 00000000 [$t5] = 00000000 [$t6] = 00000000

[$t7] = 00000000 [$t8] = 00000000 [$t9] = 00000000

We can see that first instruction is operated, t0 = 32. Addi instruction works well

After cycle four, the result is:

[$s0] = 00000057 [$s1] = fffffe9 [$s2] = 00000000

[$s3] = 00000000 [$s4] = 00000000 [$s5] = 00000000

[$s6] = 00000000 [$s7] = 00000000 [$t0] = 00000020

[$t1] = 00000037 [$t2] = 00000000 [$t3] = 00000000

[$t4] = 00000000 [$t5] = 00000000 [$t6] = 00000000

[$t7] = 00000000 [$t8] = 00000000 [$t9] = 00000000

The first four instructions are operated, t0 = 32; t1 = 55; s0 = 87; s1 = -23 The addi, add, sub instructions work well.

After cycle fifteen, the result is:

[$s0] = 00000057 [$s1] = fffffe9 [$s2] = 00000020

[$s3] = 00000037 [$s4] = 00000001 [$s5] = 00000000

[$s6] = 0000008e [$s7] = 000000ae [$t0] = 00000020

[$t1] = 00000037 [$t2] = 0000008e [$t3] = 000000ae

[$t4] = 00000000 [$t5] = 00000000 [$t6] = 00000000

[$t7] = 00000000 [$t8] = 00000000 [$t9] = 00000000

The first fifteen instructions are operated, t0 = 32, t1 = 55, s0 = 87, s1 = -23, s2 = 32, s3 = 55, s4 = 1, t1 = 142, t3 = 174, s5 = 0, s6 = 142, s7 = 174. The all instructions work well, including addi, add, sub, and, or, slt, beq, lw, sw.

Design (Pipeline)

As we learnt from the course, we separate this procedure into 5 stages and complete step by step. To make it happen, we need four stage register which update every clock cycle. Also, to solve data hazards and control hazards, we need components like hazard detection unit and forwarding unit. The flush signal and Write signal is required to play the role of inserting bubble and clearing wrong instructions.

Based on the pipelined implementation of MIPS architecture provided in project instruction (Figure. 2), we sketched our own draft schematic, shown as following.

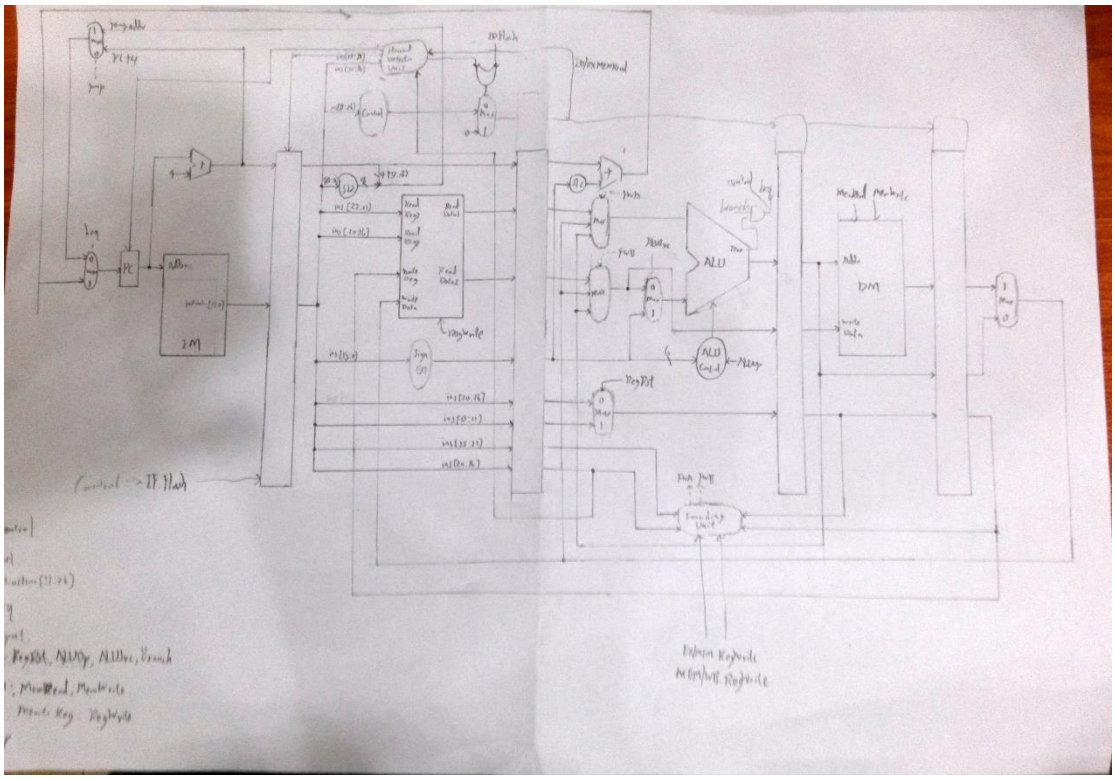


Figure Pipelined implementation

As we can see, different with the Figure 2 in project instruction, we complete with hazard detection unit and forwarding unit also some wires carrying flush and write signals to control the pipeline registers. Besides, another difference is we place the branch in the EX stage. We considered

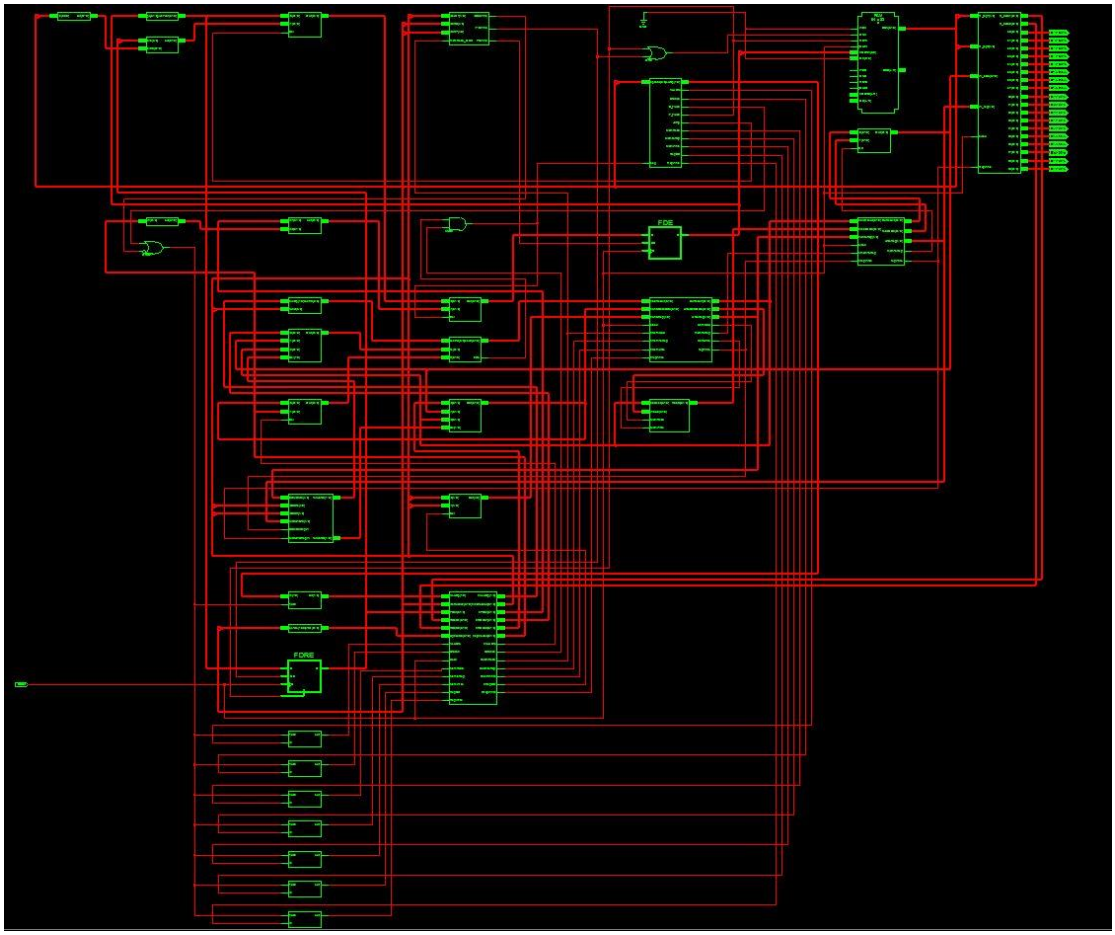
to place a comparator between two read data in ID stage. Though it may be time efficiency, but it will cause other data hazard, which leads to more complex hazard detection logic and more component supporting forwarding. If we place branch in MEM stage, we need to flush 3 previous stages when branch happens. After a balanced consideration, we decided to place branch in the EX stage, which won't cause more data hazard and be more time efficiency than placing it in MEM stage. When branch happens, we need to flush two pipeline registers (IF/ID, ID/EX). Another control hazard is about jump. Since we need to determine jump or not in the ID stage, one more instruction is fetched to the IF stage. So when jump happens, we need to flush the IF/ID register just like what we do when branch happens.

Implementation & RTL Schematic (Pipeline)

Note: All codes is shown in the Appendix.

General schematic

This is the general schematic of our design. There is one input, clock, and 18 outputs to show the values of registers.

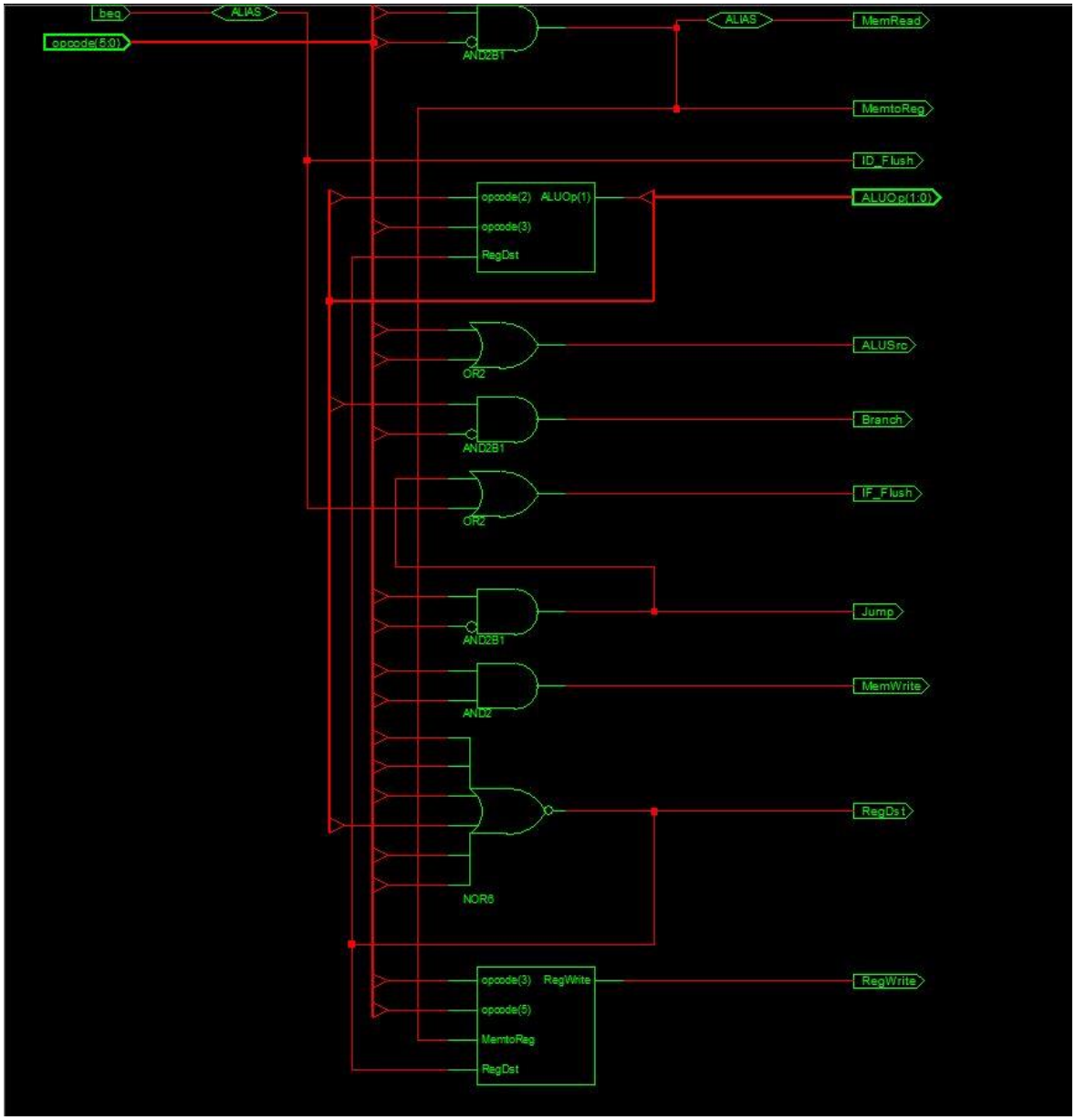


Control

Similar as the control unit in single cycle processor, the controller generate control signals according to the instruction. Since we need to control flush, so we add input (beq) and outputs (ID_Flush, IF_Flush) to this module. The schematic below shows the basic logics for inputs to outputs.

```
assign IF_Flush = beq | (opcode[1] & (~opcode[0]));
assign ID_Flush = beq;
```

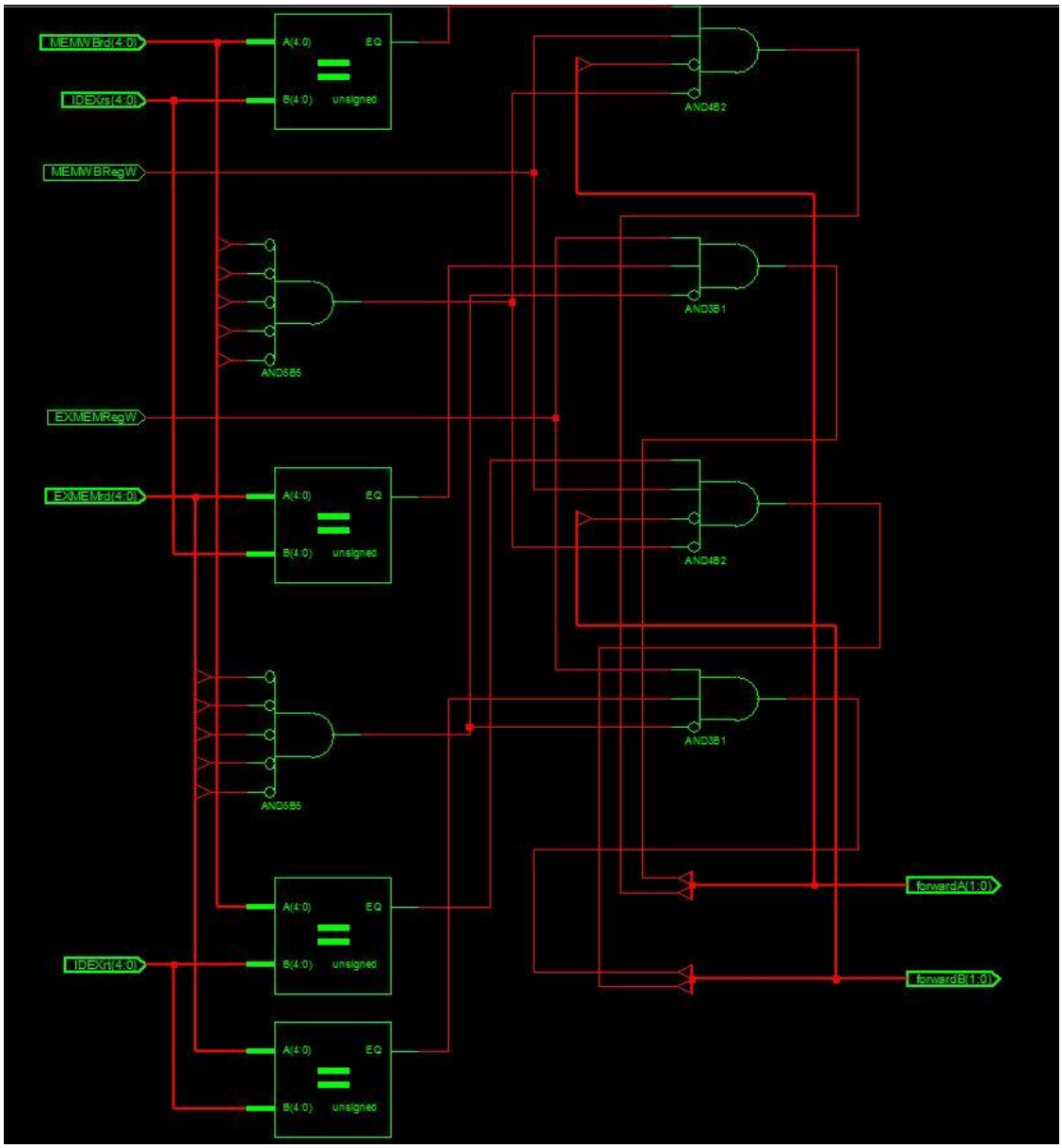
If branch happens we need to flush IF and ID stage. If jump happens, we need flush IF stage. Other output signals is same as them in single cycle processor.



Forwarding Unit

The forwarding unit detect the source registers and destination registers around three stages, to check if there is data hazard and lead right data to EX stage by control the mux in front of ALU. We have fully considered kinds of data hazards which could happen in our processor and use following logic for MEM hazard.

- **If (MEM/WB.RegWrite and (MEM/WB.RegisterRd ≠ 0)and (MEM/WB.RegisterRd = ID/EX.RegisterRs) and not (EX/MEM.RegWrite and (EX/MEM.RegisterRd ≠ 0)and (EX/MEM.RegisterRd = ID/EX.RegisterRs)))**
  - **ForwardA = 01**
- **If (MEM/WB.RegWrite and (MEM/WB.RegisterRd ≠ 0)and (MEM/WB.RegisterRd = ID/EX.RegisterRt) and not (EX/MEM.RegWrite and (EX/MEM.RegisterRd ≠ 0)and (EX/MEM.RegisterRd = ID/EX.RegisterRt)))**
  - **ForwardB = 01**

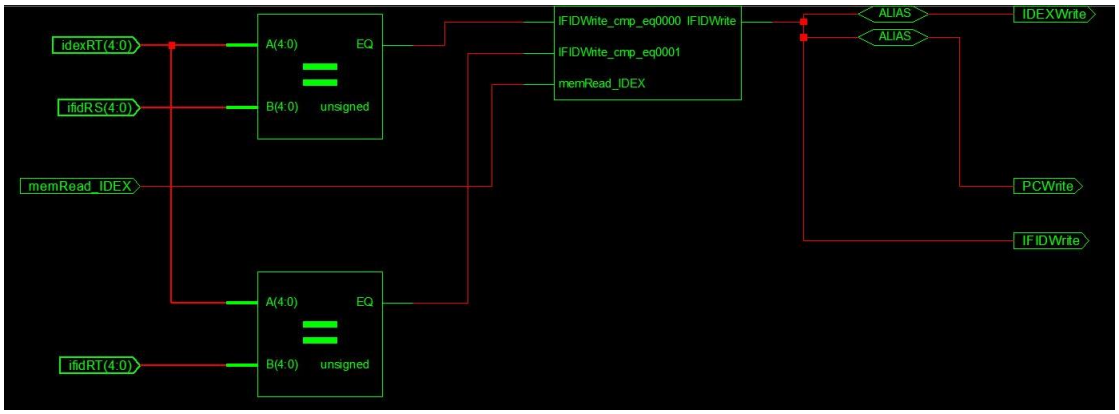


Hazard Detection Unit

When some data we need to calculate should be loaded at the just previous instruction, it is impossible to forward. So we have to stall for a cycle. To decide stall or not, we need "Write" signal for PC and pipeline registers. If "Write" signal is 1 then it goes normally. Else, it stalls, which means we hold the present value and not update. We use the logic shown as below.

ID/EX.MemRead and ((ID/EX.RegisterRt == IF/ID.RegisterRs) or (ID/EX.RegisterRt == IF/ID.RegisterRt))

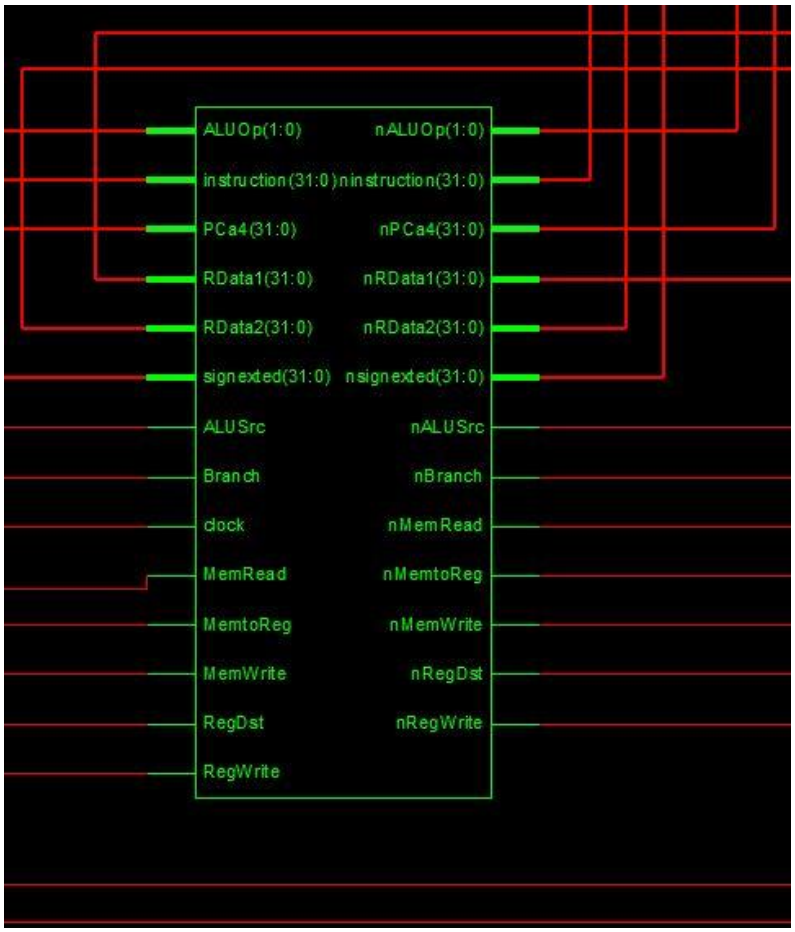
If this logic is true, PCWrite = IFIDWrite = IDEXWrite = 0;



Pipeline Registers

Since there are five stages, so we need four registers to memorize all the data in this clock cycle and be capable to use by next stage at next clock cycle. So the registers should be updated every clock cycle (at positive edge of clock). Also, several of them has inputs of flush and write signal. If flush signal is 1, clear all the reg on this module as 0. If write signal is 0, hold all the values as what they were. Besides, all the registers should be initialized as 0 to avoid something bad happen.

The IDEX_Reg interface is shown as below. The inner structure is too complicated to show. They are available upon request.



Simulation Result & Demonstration (Pipeline)

The instruction we use: modification on the base of specification. The load and beq instructions can really realize their function instead of never happening or equaling to zero.

Imemory[0] = 32'b00100000000010000000000000110110; //addi $t0, $zero, 54
Imemory[1] = 32'b00100000000010010000000000110111; //addi $t1, $zero, 55
Imemory[2] = 32'b00000001000010011000000000100000; //add $s0, $t0, $t1 datahazard for $t0,$t1
Imemory[3] = 32'b00000001000010011000100000100010; //sub $s1, $t0, $t1 datahazard for $t1
Imemory[4] = 32'b00000001000010011001000000100100; //and $s2, $t0, $t1
Imemory[5] = 32'b00000001000010011001100000100101; //or $s3, $t0, $t1
Imemory[6] = 32'b00000001000010011010000000101010; //slt $s4, $t0, $t1 (LOOP)
Imemory[7] = 32'b00010010100000000000000000000110; //beq $s4, $zero, EXIT datahazard for branch $s4
Imemory[8] = 32'b00000001000010010101000000100000; //add $t2, $t0, $t1
Imemory[9] = 32'b00000001010010010101000000100000; //add $t2, $t2, $t1 datahazard for $t2
Imemory[10] = 32'b00000001010010000101100000100000; //add $t3, $t2, $t0 double datahazard for $t2
Imemory[11] = 32'b10001100000101010000000000000100; //lw $s5, 4($zero)
Imemory[12] = 32'b00000001010101011011000000100000; //add $s6, $t2, $s5 loaduse harzard for $s5
Imemory[13] = 32'b00000001010101011101100000100000; //add $s7, $s5, $t3
Imemory[14] = 32'b10101100000101110000000000001000; //sw $s7, 8($zero) (EXIT)
Imemory[15] = 32'b10001100000101010000000000001000; //lw $s5, 8($zero)
Imemory[16] = 32'b00100001000010000000000000000001; //addi $t0, $t0, 1
Imemory[17] = 32'b00001000000000000000000000000110; //j LOOP

The complete result is attached in appendix.

We choose several cycle results to see whether our model is right.

After four cycles, the $t0 get its value

[$s0] = 00000000 [$s1] = 00000000 [$s2] = 00000000

[$s3] = 00000000 [$s4] = 00000000 [$s5] = 00000000

[$s6] = 00000000 [$s7] = 00000000 [$t0] = 00000036

[$t1] = 00000000 [$t2] = 00000000 [$t3] = 00000000

[$t4] = 00000000 [$t5] = 00000000 [$t6] = 00000000

[$t7] = 00000000 [$t8] = 00000000 [$t9] = 00000000

We can see that first instruction is operated, $t0 = 54. Addi instruction works well.

After five cycles, the result is:

[$s0] = 00000000 [$s1] = 00000000 [$s2] = 00000000

[$s3] = 00000000 [$s4] = 00000000 [$s5] = 00000000

[$s6] = 00000000 [$s7] = 00000000 [$t0] = 00000036

[$t1] = 00000037 [$t2] = 00000000 [$t3] = 00000000

[$t4] = 00000000 [$t5] = 00000000 [$t6] = 00000000

[$t7] = 00000000 [$t8] = 00000000 [$t9] = 00000000

We can see that the first two instructions are operated, t0 = 54, t1 = 55. It is pipeline for the first instruction runs for 4 cycles and the second instruction runs over after another cycle.

After six cycles, the result is:

[$s0] = 0000006d [$s1] = 00000000 [$s2] = 00000000

[$s3] = 00000000 [$s4] = 00000000 [$s5] = 00000000

[$s6] = 00000000 [$s7] = 00000000 [$t0] = 00000036

[$t1] = 00000037 [$t2] = 00000000 [$t3] = 00000000

[$t4] = 00000000 [$t5] = 00000000 [$t6] = 00000000

[$t7] = 00000000 [$t8] = 00000000 [$t9] = 00000000

We can see that the first three instructions are operated, t0 = 54, t1 = 55, s0 = 109. Next instruction finishes operating after last cycle. It is pipeline. In addition, data hazard happens. We can see that t1 changes its value in instruction two and it is used in instruction three. The final result is true and s0 = 109. Hence, our model can deal with hazard situation.

After fourteen cycles, the result is:

[$s0] = 0000006d [$s1] = ffffffff [$s2] = 00000036

[$s3] = 00000037 [$s4] = 00000001 [$s5] = 00000000

[$s6] = 00000000 [$s7] = 00000000 [$t0] = 00000036

[$t1] = 00000037 [$t2] = 000000a4 [$t3] = 000000da

[$t4] = 00000000 [$t5] = 00000000 [$t6] = 00000000

[$t7] = 00000000 [$t8] = 00000000 [$t9] = 00000000

The first eleven instructions are operated. The last three instructions of operated ones are add

$t2, $t0, $t1, add $t2, $t2, $t1 and add $t3, $t2, $t0. Double hazard happen. The first two instructions both change t2's value and the third instruction use t2's value. It works well and t2, t3 get its correct value.(t2 = 164, t3 = 218) Hence, our model can deal with double hazard situation.

After seventeen cycles, the result is:

[$s0] = 0000006d [$s1] = ffffffff [$s2] = 00000036

[$s3] = 00000037 [$s4] = 00000001 [$s5] = 00000000

[$s6] = 000000a4 [$s7] = 00000000 [$t0] = 00000036

[$t1] = 00000037 [$t2] = 000000a4 [$t3] = 000000da

[$t4] = 00000000 [$t5] = 00000000 [$t6] = 00000000

[$t7] = 00000000 [$t8] = 00000000 [$t9] = 00000000

The instructions operated after 14 cycles are lw $s5, 4($zero) and add $s6, $t2, $s5. Load use hazard happens and two instructions finish operating after another three cycles. One bubble exists. The result is right.(s5 = 0, s6 = 54) Hence, our model can deal with load use hazard situation.

After 25 cycles, instruction slt $s4, $t0, $t1 finishes operating and s4 = 0. Next instruction is beq $s4, $zero, EXIT. We can find that data hazard for branch $s4 happens. After 27 cycles, the result is:

[$s0] = 0000006d [$s1] = ffffffff [$s2] = 00000036

[$s3] = 00000037 [$s4] = 00000000 [$s5] = 000000da

[$s6] = 000000a4 [$s7] = 000000da [$t0] = 00000037

[$t1] = 00000037 [$t2] = 000000a4 [$t3] = 000000da

[$t4] = 00000000 [$t5] = 00000000 [$t6] = 00000000

[$t7] = 00000000 [$t8] = 00000000 [$t9] = 00000000

The value of t2 is still 164 instead of 110.(t0 + t1) It means the beq instruction works well and the PC jump to sw $s7, 8($zero) instead of operating add $t2, $t0, $t1. Hence, Hence, our model can deal with beq instruction.

Conclusion

In general, we have complete this project successfully.

We design and implement all the sub modules, and work hard to overall consider all the data hazards and control hazards. During the process, we are discovering bugs and debugging all the time. Fortunately, we solve all the bugs we ever found. This project motivate us to understand the working principle of processor better. Plus, it is really a good experience to work on a project as a team and solve problems under pressure which enhance our ability.

Appendix (code and simulation results)

I. Single Cycle Codes

```
module single_cycle(clock, s0, s1, s2, s3, s4, s5, s6, s7, t0, t1,
t2, t3, t4, t5, t6, t7, t8, t9);
    input clock;
    wire [31:0] w2, instruction, PCa4, Jaddr, signExted, rData1,
rData2, alusrc2, ALUresult, rData, addsrc2, Baddr, wbData, addr,
newPC;
    wire zero;
    wire [27:0] add28;
    wire [3:0] alucontrol;
    wire [4:0] wReg;
    wire RegDst, Jump, Branch, MemRead, MemtoReg, MemWrite, ALUSrc,
RegWrite;
    wire [1:0] ALUOp;
    wire [31:0] at0, at1, at2, at3, at4, at5, at6, at7, at8, at9,
as0, as1, as2, as3, as4, as5, as6, as7;
    output [31:0] t0, t1, t2, t3, t4, t5, t6, t7, t8, t9, s0, s1, s2,
s3, s4, s5, s6, s7;

    PC PC(clock, newPC, w2);
    IM IM(w2, instruction);
    control control(instruction[31:26], RegDst, Jump, Branch,
MemRead, MemtoReg, ALUOp, MemWrite, ALUSrc, RegWrite);
    Add4 PCplus(PCa4,w2);
    sl2_26b sl2_26b(instruction[25:0], add28);
    connect_4_28 connect_4_28(PCa4[31:28], add28, Jaddr);
    Adder Adder(PCa4, addsrc2, Baddr);
    signExt signExt(signExted, instruction[15:0]);
    sl2_32b sl2_32b(signExted, addsrc2);
    ALUControl ALUControl(alucontrol, ALUOp, instruction[5:0]);
    ALU ALU(zero, ALUresult, rData1, alusrc2, alucontrol);
    regfile regfile(clock, instruction[25:21], instruction[20:16],
rData1, rData2, wReg, wbData, RegWrite,
    at0, at1, at2, at3, at4, at5, at6, at7, at8, at9, as0, as1, as2,
as3, as4, as5, as6, as7);
    DM DM(clock, ALUresult, rData2, MemRead, MemWrite, rData,Dmem);
    mux_5b mux_regdst(wReg, instruction[20:16], instruction[15:11],
RegDst);
    mux_32b mux_alusrc2(alusrc2, rData2, signExted, ALUSrc);
    mux_32b mux_branch(addr, PCa4, Baddr, Branch & zero);
    mux_32b mux_jump(newPC, addr, Jaddr, Jump);
```

```
    mux_32b mux_wb(wbData, ALUresult, rData, MemtoReg);

     assign t0 = at0;
     assign t1 = at1;
     assign t2 = at2;
     assign t3 = at3;
     assign t4 = at4;
     assign t5 = at5;
     assign t6 = at6;
     assign t7 = at7;
     assign t8 = at8;
     assign t9 = at9;
     assign s0 = as0;
     assign s1 = as1;
     assign s2 = as2;
     assign s3 = as3;
     assign s4 = as4;
     assign s5 = as5;
     assign s6 = as6;
     assign s7 = as7;

endmodule


module PC(clock, in, out
    );
    input clock;
    input [31:0] in;
    output [31:0] out;
    reg [31:0] out;

    initial begin
    out = 32'hFFFFFFFC;
    end

    always @ (posedge clock)
        out <= in;

endmodule


module IM(address, instruction
    );
    parameter baseAddr = 8'h00000000;
```

```
    input [31:0] address;
    output [31:0] instruction;
    reg [31:0] Imemory [63:0];
    integer k;

    initial begin
for(k=0;k<64;k=k+1)begin
Imemory[k] = 32'b0;
end
//You can give your own code block here.
Imemory[0] = 32'b00100000000010000000000000100000; //addi $t0, $zero,
32
Imemory[1] = 32'b00100000000010010000000000110111; //addi $t1, $zero,
55
Imemory[2] = 32'b00000001000010011000000000100000; //add $s0, $t0,
$t1
Imemory[3] = 32'b00000001000010011000100000100010; //sub $s1, $t0,
$t1
Imemory[4] = 32'b00000001000010011001000000100100; //and $s2, $t0,
$t1
Imemory[5] = 32'b00000001000010011001100000100101; //or $s3, $t0, $t1
Imemory[6] = 32'b00000001000010011010000000101010; //slt $s4, $t0,
$t1 (LOOP)
Imemory[7] = 32'b00010010100000000000000000000110; //beq $s4, $zero,
EXIT
Imemory[8] = 32'b00000001000010010101000000100000; //add $t2, $t0,
$t1
Imemory[9] = 32'b00000001010010010101000000100000; //add $t2, $t2,
$t1
Imemory[10] = 32'b00000001010010000101100000100000; //add $t3, $t2,
$t0
Imemory[11] = 32'b10001100000101010000000000000100; //lw $s5,
4($zero)
Imemory[12] = 32'b00000001010101011011000000100000; //add $s6, $t2,
$s5
Imemory[13] = 32'b00000001010101011101100000100000; //add $s7, $s5,
$t3
Imemory[14] = 32'b10101100000101010000000000001000; //sw $s5,
8($zero) (EXIT)
Imemory[15] = 32'b00001000000000000000000000000110; //j LOOP
end

    assign instruction = Imemory[((address - baseAddr) / 4) % 64];
```

```
endmodule


module control(opcode, RegDst, Jump, Branch, MemRead, MemtoReg,
ALUOp, MemWrite, ALUSrc, RegWrite
    );
    input [5:0] opcode;
    output RegDst, Jump, Branch, MemRead, MemtoReg, MemWrite, ALUSrc,
RegWrite;
    output [1:0] ALUOp;

    assign RegDst = ~(opcode[0] | opcode[1] | opcode[2] | opcode[3] |
opcode[4] | opcode[5]);
    assign Jump = opcode[1] & (~opcode[0]);
    assign Branch = opcode[2] & (~opcode[3]);
    assign MemRead = opcode[5] & (~opcode[3]);
    assign MemtoReg = opcode[5] & (~opcode[3]);
    assign ALUOp[1] = ~(opcode[0] | opcode[1] | opcode[2] | opcode[3]
| opcode[4] | opcode[5]) | (opcode[3] & opcode[2]);
    assign ALUOp[0] = opcode[2];
    assign MemWrite = opcode[5] & opcode[3];
    assign ALUSrc = opcode[5] | opcode[3];
    assign RegWrite = (~opcode[5] & opcode[3]) | (opcode[5] &
~opcode[3]) | ~(opcode[0] | opcode[1] | opcode[2] | opcode[3] |
opcode[4] | opcode[5]);

endmodule

module regfile(clock, R_reg1, R_reg2, R_data1, R_data2, W_reg,
W_data, RegWrite,
        t0, t1, t2, t3, t4, t5, t6, t7, t8, t9, s0, s1, s2, s3, s4,
s5, s6, s7

    );
     input [4:0] R_reg1, R_reg2, W_reg;
     input [31:0] W_data;
     input RegWrite;
     input clock;
     output [31:0] R_data1, R_data2;
     reg [31:0] R_data1, R_data2;

     output [31:0] t0, t1, t2, t3, t4, t5, t6, t7, t8, t9, s0, s1, s2,
s3, s4, s5, s6, s7;
     reg [31:0] memory [31:0];
```

```
    integer k;
    initial begin
   for(k=0;k<32;k=k+1)begin
   memory[k] = 32'b0;
   end
    end

    always @(clock or R_reg1 or R_reg2)
    begin
       R_data1 = memory[R_reg1];
       R_data2 = memory[R_reg2];
    end
    always @(negedge clock)
    begin
      if (RegWrite) memory[W_reg] = W_data;
       else begin end
    end

    assign t0 = memory[8];
    assign t1 = memory[9];
    assign t2 = memory[10];
    assign t3 = memory[11];
    assign t4 = memory[12];
    assign t5 = memory[13];
    assign t6 = memory[14];
    assign t7 = memory[15];
    assign t8 = memory[24];
    assign t9 = memory[25];
    assign s0 = memory[16];
    assign s1 = memory[17];
    assign s2 = memory[18];
    assign s3 = memory[19];
    assign s4 = memory[20];
    assign s5 = memory[21];
    assign s6 = memory[22];
    assign s7 = memory[23];

endmodule

module ALU(zero, result, rs, rt, control);
    output zero;
    output reg [31:0] result;
```

```
    input [31:0] rs;
    input [31:0] rt;
    input [3:0] control;

    assign zero = (rs == rt)?1:0;
    always @(rs, rt, control)
    begin
        case(control)
        4'b0000: result = rs & rt;
        4'b0001: result = rs | rt;
        4'b0010: result = rs + rt;
        4'b0110: result = rs - rt;
        4'b0111: begin
            if (rs[31] == 0 & rt[31] == 0) result = (rs<rt)?1:0;
            else if (rs[31] == 1 & rt[31] == 0) result = 1;
            else if (rs[31] == 1 & rt[31] == 1) result =
(rs[30:0]<rt[30:0])?0:1;
            else result = 0; end
        default: result = 0;
        endcase
    end
endmodule

module ALUControl(control, aluOp, funct);
    output reg [3:0]control;
    input [1:0]aluOp;
    input [5:0]funct;

    always @(aluOp or funct)
    begin
        if(aluOp == 2'b00)
            control = 4'b0010;
        else if(aluOp == 2'b01)
            control = 4'b0110;
        else if(aluOp == 2'b10)
        begin
            case(funct[3:0])
            4'b0000: control = 4'b0010;
            4'b0010: control = 4'b0110;
            4'b0100: control = 4'b0000;
            4'b0101: control = 4'b0001;
            4'b1010: control = 4'b0111;
            default: control = 4'b0000;
            endcase
```

```
        end
        else
            control = 4'b0000;
    end
endmodule

module DM(clock, address, Wdata, MemRead, MemWrite, Rdata,out
    );
     input [31:0] address;
     input [31:0] Wdata;
     input MemRead, MemWrite;
     input clock;
     output [31:0] Rdata;
     reg [31:0] Rdata;
     reg [31:0] Dmemory [63:0];

     output[31:0] out;

     integer k;
     initial begin
    for(k=0;k<64;k=k+1)begin
    Dmemory[k] = 32'b0;
    end
     end

     always @ (clock or MemWrite or Wdata or MemRead)
       begin
       if (MemWrite) Dmemory[(address/4) % 64] = Wdata;
       else if (MemRead) Rdata = Dmemory[(address/4) % 64];
       else Rdata = 32'b0;
       end

     assign out = Dmemory[2];
endmodule

module signExt(after, before);
    output [31:0] after;
    input [15:0] before;

    assign after[15:0] = before[15:0];
    assign after[31:16] = {16{before[15]}};

endmodule
```

```
module Add4(plusFour, pc);
    output [31:0] plusFour;
    input [31:0] pc;

    assign plusFour = pc + 4;
endmodule

module Adder(in1, in2, out
    );
    input [31:0] in1, in2;
    output [31:0] out;

    assign out = in1 + in2;

endmodule

module connect_4_28(bit4, bit28, out
    );
     input [3:0] bit4;
     input [27:0] bit28;
     output [31:0] out;

     assign out[31:28] = bit4;
     assign out[27:0] = bit28;

endmodule

module mux_5b(Out, I0, I1, Sel
    );
     input [4:0] I0, I1;
    input Sel;
     output [4:0] Out;
     reg [4:0] Out;

     always @(I0, I1, Sel) begin
     case (Sel)
        1'b0: Out = I0;
        1'b1: Out = I1;
        default: Out = 5'b0;
     endcase
     end

endmodule
```

```
module mux_32b(Out, I0, I1, Sel
    );
     input [31:0] I0, I1;
    input Sel;
     output [31:0] Out;
     reg [31:0] Out;

     always @(I0, I1, Sel) begin
     case (Sel)
        1'b0: Out = I0;
        1'b1: Out = I1;
        default: Out = 32'b0;
     endcase
     end

endmodule

module sl2_26b(in, out
    );
    input [25:0] in;
    output [27:0] out;

    assign out[27:2] = in[25:0];
    assign out[1:0] = 2'b00;

endmodule

module sl2_32b(in, out
    );
    input [31:0] in;
    output [31:0] out;

    assign out = in << 2;

endmodule
```

II. Pipeline Codes

Note: Some module is same as the module we use in the single cycle processer. So if any module is undefined here, please refer to single cycle codes. And in pipeline processer, we defined new modules “control”, “DM” and “regfile”, not the same as in single cycle processer.

```
module pipeline(clock,s0, s1, s2, s3, s4, s5, s6, s7, t0, t1, t2, t3,
t4, t5, t6, t7, t8, t9
```

```
    );
    input clock;
    wire [31:0] nPC, PCaddr, IFPCa4, IFinstruction, IDPCa4,
IDinstruction, RData1, RData2, WBwritedata;
    wire [31:0] jumpaddr, IDsignexted, offset, branchaddr, jaddr,
ALUs1, ALUs2, ALUts, EXALUResult;
    wire [31:0] EXPCa4, EXRData1, EXRData2, EXsignexted,
EXinstruction;
    wire [31:0] MEMALUResult, MEMwritedata, MEMreaddata;
    wire [31:0] WBALUResult, WBreaddata;
    wire [4:0] EXwbReg, MEMwbReg, WBReg;
    wire PCWrite, IFIDWrite, WBRegWrite, beq, IF_Flush, ID_Flush,
jump, flush, hazFlush, IDEXWrite, zero;
    wire [1:0] FWA, FWB;
    wire IDRegDst, IDBranch, IDMemRead, IDMemtoReg, IDMemWrite,
IDALUSrc, IDRegWrite;
    wire fIDRegDst, fIDBranch, fIDMemRead, fIDMemtoReg, fIDMemWrite,
fIDALUSrc, fIDRegWrite;
    wire EXRegDst, EXBranch, EXMemRead, EXMemtoReg, EXMemWrite,
EXALUSrc, EXRegWrite;
    wire MEMMemRead, MEMMemWrite, MEMMemtoReg,MEMRegWrite;
    wire WBMemtoReg;
    wire [1:0] IDALUOp, fIDALUOp, EXALUOp;
    wire [27:0] sl28;
    wire [3:0] alucontrol;
    wire [31:0] at0, at1, at2, at3, at4, at5, at6, at7, at8, at9,
as0, as1, as2, as3, as4, as5, as6, as7;
    output [31:0] t0, t1, t2, t3, t4, t5, t6, t7, t8, t9, s0, s1, s2,
s3, s4, s5, s6, s7;

    PC PC(clock, nPC, PCaddr, PCWrite);
    Addfour Addfour(IFPCa4, PCaddr);
    IM IM(PCaddr, IFinstruction);
    IFID_Reg IFID_Reg(clock, IFPCa4, IFinstruction, IDPCa4,
IDinstruction, IF_Flush, IFIDWrite);
    regfile regfile(clock, IDinstruction[25:21],
IDinstruction[20:16], RData1, RData2, WBReg, WBwritedata, WBRegWrite,
    at0, at1, at2, at3, at4, at5, at6, at7, at8, at9, as0, as1, as2,
as3, as4, as5, as6, as7);
    signExt signExt(IDsignexted, IDinstruction[15:0]);
    sl2_26b sl2_26b(IDinstruction[25:0], sl28);
    connect_4_28 connect_4_28(IDPCa4[31:28], sl28, jumpaddr);
    control control(IDinstruction[31:26], beq, IDRegDst, jump,
IDBranch, IDMemRead, IDMemtoReg,
```

```
    IDALUOp, IDMemWrite, IDALUSrc, IDRegWrite, IF_Flush, ID_Flush);
    not(hazFlush, IDEXWrite);
    or(flush, ID_Flush, hazFlush);
    FlushMux_1b fMux_RegDst(flush, IDRegDst, fIDRegDst);
    FlushMux_1b fMux_Branch(flush, IDBranch, fIDBranch);
    FlushMux_1b fMux_MemRead(flush, IDMemRead, fIDMemRead);
    FlushMux_1b fMux_MemtoReg(flush, IDMemtoReg, fIDMemtoReg);
    FlushMux_1b fMux_MemWrite(flush, IDMemWrite, fIDMemWrite);
    FlushMux_1b fMux_ALUSrc(flush, IDALUSrc, fIDALUSrc);
    FlushMux_1b fMux_RegWrite(flush, IDRegWrite, fIDRegWrite);
    FlushMux_2b fMux_ALUOp(flush, IDALUOp, fIDALUOp);
    IDEX_Reg IDEX_Reg(clock, IDPCa4, RData1, RData2, IDsignexted,
IDinstruction,
    fIDRegDst, fIDALUOp, fIDALUSrc, fIDBranch, fIDMemRead,
fIDMemWrite, fIDMemtoReg, fIDRegWrite,
    EXPCa4, EXRData1, EXRData2, EXsignexted, EXinstruction,
    EXRegDst, EXALUOp, EXALUSrc, EXBranch, EXMemRead, EXMemWrite,
EXMemtoReg, EXRegWrite);
    hazDetectUnit hazDetectUnit(IFIDWrite, PCWrite, IDEXWrite,
    EXinstruction[20:16], IDinstruction[25:21], IDinstruction[20:16],
EXMemRead);
    sl2_32b sl2_32b(EXsignexted, offset);
    Adder Adder(EXPCa4, offset, branchaddr);
    mux_32b_2x1 mux_jump(jaddr, IFPCa4, jumpaddr, jump);
    mux_32b_2x1 mux_branch(nPC, jaddr, branchaddr, beq);
    mux_5b_2x1 mux_RegDst(EXwbReg, EXinstruction[20:16],
EXinstruction[15:11], EXRegDst);
    ALUControl ALUControl(alucontrol, EXALUOp, EXinstruction[5:0]);
    mux_32b_3x1 mux_FWA(ALUs1, EXRData1, WBwritedata, MEMALUResult,
FWA);
    mux_32b_3x1 mux_FWB(ALUts, EXRData2, WBwritedata, MEMALUResult,
FWB);
    mux_32b_2x1 mux_ALUSrc(ALUs2, ALUts, EXsignexted, EXALUSrc);
    ALU ALU(zero, EXALUResult, ALUs1, ALUs2, alucontrol);
    and(beq, EXBranch, zero);
    ForwardingUnit ForwardingUnit(EXinstruction[25:21],
EXinstruction[20:16], MEMRegWrite, WBRegWrite, MEMwbReg, WBReg, FWA,
FWB);
    EXMEM_Reg EXMEM_Reg(MEMMemRead, MEMMemWrite, MEMMemtoReg,
MEMRegWrite, MEMALUResult, MEMwritedata, MEMwbReg,
    EXMemRead, EXMemWrite, EXMemtoReg, EXRegWrite, EXALUResult,
ALUts, EXwbReg,clock);
    DM DM(MEMALUResult, MEMwritedata, MEMMemRead, MEMMemWrite,
MEMreaddata);
```

```
    MEMWB_Reg MEMWB_Reg(WBMemtoReg, WBRegWrite, WBALUResult,
WBreaddata, WBReg,
    MEMMemtoReg, MEMRegWrite, MEMALUResult, MEMreaddata, MEMwbReg,
clock);
    mux_32b_2x1 mux_MemtoReg(WBwritedata, WBALUResult, WBreaddata,
WBMemtoReg);

    assign t0 = at0;
    assign t1 = at1;
    assign t2 = at2;
    assign t3 = at3;
    assign t4 = at4;
    assign t5 = at5;
    assign t6 = at6;
    assign t7 = at7;
    assign t8 = at8;
    assign t9 = at9;
    assign s0 = as0;
    assign s1 = as1;
    assign s2 = as2;
    assign s3 = as3;
    assign s4 = as4;
    assign s5 = as5;
    assign s6 = as6;
    assign s7 = as7;

endmodule

module PC(clock, in, out, PCWrite
   );
   input clock, PCWrite;
   input [31:0] in;
   output [31:0] out;
   reg [31:0] out;

   initial begin
   out = 8'hFFFFFFFC;
   end

   always @ (posedge clock) begin
      if (PCWrite) out = in;
      else out = out;
   end
```

```
endmodule

module control(opcode, beq, RegDst, Jump, Branch, MemRead, MemtoReg,
ALUOp, MemWrite, ALUSrc, RegWrite, IF_Flush, ID_Flush
    );
    input [5:0] opcode;
    input beq;
    output RegDst, Jump, Branch, MemRead, MemtoReg, MemWrite, ALUSrc,
RegWrite;
    output [1:0] ALUOp;
    output IF_Flush, ID_Flush;

    assign RegDst = ~(opcode[0] | opcode[1] | opcode[2] | opcode[3] |
opcode[4] | opcode[5]);
    assign Jump = opcode[1] & (~opcode[0]);
    assign Branch = opcode[2] & (~opcode[3]);
    assign MemRead = opcode[5] & (~opcode[3]);
    assign MemtoReg = opcode[5] & (~opcode[3]);
    assign ALUOp[1] = ~(opcode[0] | opcode[1] | opcode[2] | opcode[3]
| opcode[4] | opcode[5]) | (opcode[3] & opcode[2]);
    assign ALUOp[0] = opcode[2];
    assign MemWrite = opcode[5] & opcode[3];
    assign ALUSrc = opcode[5] | opcode[3];
    assign RegWrite = (~opcode[5] & opcode[3]) | (opcode[5] &
~opcode[3]) | ~(opcode[0] | opcode[1] | opcode[2] | opcode[3] |
opcode[4] | opcode[5]);
    assign IF_Flush = beq | (opcode[1] & (~opcode[0]));
    assign ID_Flush = beq;

endmodule


module regfile(clock, R_reg1, R_reg2, R_data1, R_data2, W_reg,
W_data, RegWrite,
        t0, t1, t2, t3, t4, t5, t6, t7, t8, t9, s0, s1, s2, s3, s4,
s5, s6, s7
    );
     input [4:0] R_reg1, R_reg2, W_reg;
     input [31:0] W_data;
     input RegWrite;
     input clock;
     output [31:0] R_data1, R_data2;
     reg [31:0] R_data1, R_data2;
```

```
    output [31:0] t0, t1, t2, t3, t4, t5, t6, t7, t8, t9, s0, s1, s2,
s3, s4, s5, s6, s7;
    reg [31:0] memory [31:0];


    integer k;
    initial begin
   for(k=0;k<32;k=k+1)begin
   memory[k] = 32'b0;
   end
    end

    always @(clock or R_reg1 or R_reg2)
    begin
       R_data1 = memory[R_reg1];
       R_data2 = memory[R_reg2];
    end

    always @(RegWrite or W_data or clock)
    begin
      if (RegWrite & clock) memory[W_reg] = W_data;
       else begin end
    end

    assign t0 = memory[8];
    assign t1 = memory[9];
    assign t2 = memory[10];
    assign t3 = memory[11];
    assign t4 = memory[12];
    assign t5 = memory[13];
    assign t6 = memory[14];
    assign t7 = memory[15];
    assign t8 = memory[24];
    assign t9 = memory[25];
    assign s0 = memory[16];
    assign s1 = memory[17];
    assign s2 = memory[18];
    assign s3 = memory[19];
    assign s4 = memory[20];
    assign s5 = memory[21];
    assign s6 = memory[22];
    assign s7 = memory[23];

endmodule
```

```
module ForwardingUnit(IDEXrs, IDEXrt, EXMEMRegW, MEMWBRegW, EXMEMrd,
MEMWBrd, forwardA, forwardB);

    input [4:0]  IDEXrs, IDEXrt, EXMEMrd, MEMWBrd;
    input  EXMEMRegW, MEMWBRegW;
    output [1:0] forwardA, forwardB;


    assign forwardA[1] = ( EXMEMRegW & (EXMEMrd != 5'b0) & (EXMEMrd
== IDEXrs) )? 1'b1: 1'b0;
    assign forwardA[0] = ( MEMWBRegW & (MEMWBrd != 5'b0) & (MEMWBrd
== IDEXrs) & (~(EXMEMRegW & (EXMEMrd != 5'b0) & (EXMEMrd ==
IDEXrs))) )? 1'b1: 1'b0;
    assign forwardB[1] = ( EXMEMRegW & (EXMEMrd != 5'b0) & (EXMEMrd
== IDEXrt) )? 1'b1: 1'b0;
    assign forwardB[0] = ( MEMWBRegW & (MEMWBrd != 5'b0) & (MEMWBrd
== IDEXrt) & (~(EXMEMRegW & (EXMEMrd != 5'b0) & (EXMEMrd ==
IDEXrt))) )? 1'b1: 1'b0;

endmodule

module
hazDetectUnit(IFIDWrite,PCWrite,IDEXWrite,idexRT,ifidRS,ifidRT,memRea
d_IDEX);
    output IFIDWrite, PCWrite, IDEXWrite;
    input [4:0] idexRT, ifidRS, ifidRT;
    input memRead_IDEX;

    assign IFIDWrite = (memRead_IDEX & ((idexRT == ifidRS) | (idexRT
== ifidRT)))?1'b0:1'b1;
    assign PCWrite = (memRead_IDEX & ((idexRT == ifidRS) | (idexRT ==
ifidRT)))?1'b0:1'b1;
    assign IDEXWrite = (memRead_IDEX & ((idexRT == ifidRS) | (idexRT
== ifidRT)))?1'b0:1'b1;

endmodule

module DM( address, Wdata, MemRead, MemWrite, Rdata
    );
     input [31:0] address;
     input [31:0] Wdata;
     input MemRead, MemWrite;
     output [31:0] Rdata;
```

```
    reg [31:0] Rdata;
    reg [31:0] Dmemory [63:0];

    integer k;
    initial begin
   for(k=0;k<64;k=k+1)begin
   Dmemory[k] = 32'b0;
   end
    end

    always @ (MemWrite or Wdata or MemRead)
      begin
      if (MemWrite) Dmemory[(address/4) % 64] = Wdata;
      else if (MemRead) Rdata = Dmemory[(address/4) % 64];
      else Rdata = 32'b0;
      end

endmodule

module IFID_Reg(clock, PCa4, instruction, nPCa4, ninstruction,
IFFlush, IFIDWrite
   );
    input [31:0] PCa4, instruction;
    input IFFlush, IFIDWrite;
    input clock;
    output [31:0] nPCa4, ninstruction;
    reg [31:0] nPCa4, ninstruction;

    initial begin
    nPCa4 = 32'b0;
    ninstruction = 32'b0;
    end

    always @ (posedge clock)
    begin
      if (IFFlush) begin nPCa4 = 32'b0;
                          ninstruction = 32'b0; end
      else if (IFIDWrite == 0) begin nPCa4 = nPCa4;
                                      ninstruction =
ninstruction; end
      else begin nPCa4 = PCa4;
                 ninstruction = instruction; end
    end
```

```
endmodule

module IDEX_Reg(clock, PCa4, RData1, RData2, signexted, instruction,
      RegDst, ALUOp, ALUSrc, Branch, MemRead, MemWrite, MemtoReg,
RegWrite,
       nPCa4, nRData1, nRData2, nsignexted, ninstruction,
      nRegDst, nALUOp, nALUSrc, nBranch, nMemRead, nMemWrite,
nMemtoReg, nRegWrite);
    input clock;
    input [31:0] PCa4, RData1, RData2, signexted, instruction;
    input RegDst, ALUSrc, Branch, MemRead, MemWrite, MemtoReg,
RegWrite;
    input [1:0] ALUOp;
    output reg [31:0] nPCa4, nRData1, nRData2, nsignexted,
ninstruction;
    output reg nRegDst, nALUSrc, nBranch, nMemRead, nMemWrite,
nMemtoReg, nRegWrite;
    output reg [1:0] nALUOp;

    initial begin
    nPCa4 = 32'b0;
    nRData1 = 32'b0;
    nRData2 = 32'b0;
    nsignexted = 32'b0;
    ninstruction = 32'b0;
    nRegDst = 0;
    nALUOp= 0;
    nALUSrc = 0;
    nBranch = 0;
    nMemRead = 0;
    nMemWrite = 0;
    nMemtoReg= 0;
    nRegWrite = 0;
    end

    always @(posedge clock)
    begin
        nPCa4 = PCa4;
        nRData1 = RData1;
        nRData2 = RData2;
        nsignexted = signexted;
        ninstruction = instruction;
        nRegDst = RegDst;
        nALUOp = ALUOp;
```

```
        nALUSrc = ALUSrc;
        nBranch = Branch;
        nMemRead = MemRead;
        nMemWrite = MemWrite;
        nMemtoReg= MemtoReg;
        nRegWrite = RegWrite;
    end

endmodule

module EXMEM_Reg(Memread, Memwrite, memToReg, regWrite, aluResult,
writeDataData, writeReg,
nMemread, nMemwrite, nmemToReg, nregWrite, naluResult,
nwriteDataData, nwriteReg,clock);
    output reg memToReg, regWrite, Memread, Memwrite;
    output reg [31:0] aluResult, writeDataData;
    output reg [4:0] writeReg;
    input nmemToReg, nregWrite, clock, nMemread, nMemwrite;
    input [31:0] naluResult, nwriteDataData;
    input [4:0] nwriteReg;

    initial begin
    Memread = 0;
    Memwrite = 0;
    memToReg = 0;
    regWrite = 0;
    aluResult = 32'b0;
    writeDataData = 32'b0;
    writeReg = 5'b0;
    end

    always @(posedge clock)
    begin
        memToReg = nmemToReg;
        regWrite = nregWrite;
        aluResult = naluResult;
        writeDataData = nwriteDataData;
        writeReg = nwriteReg;
        Memread = nMemread;
        Memwrite = nMemwrite;
    end
endmodule
```

```
module MEMWB_Reg(memToReg, regWrite, aluResult, readData, writeReg,
nmemToReg, nregWrite, naluResult, nreadData, nwriteReg,clock);
    output reg memToReg, regWrite;
    output reg [31:0] aluResult, readData;
    output reg [4:0] writeReg;
    input nmemToReg, nregWrite, clock;
    input [31:0] naluResult, nreadData;
    input [4:0] nwriteReg;

    initial begin
    memToReg = 0;
    regWrite = 0;
    aluResult = 32'b0;
    readData = 32'b0;
    writeReg = 5'b0;
    end

    always @(posedge clock)
    begin
        memToReg = nmemToReg;
        regWrite = nregWrite;
        aluResult = naluResult;
        readData = nreadData;
        writeReg = nwriteReg;
    end

endmodule

module FlushMux_1b(flush, in, out
    );
     input flush, in;
     output out;
     reg out;

     always @(flush or in) begin
     if (flush) out = 0;
     else out = in;
     end

endmodule

module FlushMux_2b(flush, in, out
    );
     input flush;
```

```
    input [1:0] in;
     output [1:0] out;
     reg [1:0] out;

     always @(flush or in) begin
     if (flush) out = 2'b00;
     else out = in;
     end

endmodule

module mux_5b_2x1(Out, I0, I1, Sel
    );
     input [4:0] I0, I1;
    input Sel;
     output [4:0] Out;
     reg [4:0] Out;

     always @(I0, I1, Sel) begin
     case (Sel)
        1'b0: Out = I0;
        1'b1: Out = I1;
        default: Out = 5'b0;
     endcase
     end

endmodule

module mux_32b_3x1(Out, I0, I1, I2, Sel
    );
     input [31:0] I0, I1, I2;
    input [1:0] Sel;
     output [31:0] Out;
     reg [31:0] Out;

     always @(I0, I1, I2, Sel) begin
     case (Sel)
        2'b00: Out = I0;
        2'b01: Out = I1;
        2'b10: Out = I2;
        default: Out = 32'b0;
     endcase
     end
```

```
endmodule
```

III. Single Cycle Simulation Result

```
module testbench;
    parameter half_period = 50;
    reg clock;
    wire [31:0] s0, s1,
s2,s3,s4,s5,s6,s7,t0,t1,t2,t3,t4,t5,t6,t7,t8,t9;
    single_cycle UUT(clock, s0, s1,
s2,s3,s4,s5,s6,s7,t0,t1,t2,t3,t4,t5,t6,t7,t8,t9);

    initial begin
        $display("************************************************");
        $display("The textual simulation results:");
        $display("************************************************");
        $monitor("Time : %0d Clock: %d \n[$s0] = %h  [$s1] = %h  [$s2]
= %h \n[$s3] = %h  [$s4] = %h  [$s5] = %h \n[$s6] = %h  [$s7] = %h
[$t0] = %h \n[$t1] = %h  [$t2] = %h  [$t3] = %h \n[$t4] = %h  [$t5]
= %h  [$t6] = %h \n[$t7] = %h  [$t8] = %h  [$t9] = %h \n",
            $time, clock, s0, s1,
s2,s3,s4,s5,s6,s7,t0,t1,t2,t3,t4,t5,t6,t7,t8,t9);
    end

    initial begin
        #0 clock = 1;
    end

    always #half_period clock = ~clock;

    initial #3000 $stop;
endmodule
```

```
************************************************
The textual simulation results:
************************************************
Finished circuit initialization process.
Time : 0 Clock: 1
[$s0] = 00000000  [$s1] = 00000000  [$s2] = 00000000
[$s3] = 00000000  [$s4] = 00000000  [$s5] = 00000000
```

```
[$s6] = 00000000  [$s7] = 00000000  [$t0] = 00000000
[$t1] = 00000000  [$t2] = 00000000  [$t3] = 00000000
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 50 Clock: 0
[$s0] = 00000000  [$s1] = 00000000  [$s2] = 00000000
[$s3] = 00000000  [$s4] = 00000000  [$s5] = 00000000
[$s6] = 00000000  [$s7] = 00000000  [$t0] = 00000000
[$t1] = 00000037  [$t2] = 00000000  [$t3] = 00000000
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 100 Clock: 1
[$s0] = 00000000  [$s1] = 00000000  [$s2] = 00000000
[$s3] = 00000000  [$s4] = 00000000  [$s5] = 00000000
[$s6] = 00000000  [$s7] = 00000000  [$t0] = 00000000
[$t1] = 00000037  [$t2] = 00000000  [$t3] = 00000000
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 150 Clock: 0
[$s0] = 00000037  [$s1] = 00000000  [$s2] = 00000000
[$s3] = 00000000  [$s4] = 00000000  [$s5] = 00000000
[$s6] = 00000000  [$s7] = 00000000  [$t0] = 00000000
[$t1] = 00000037  [$t2] = 00000000  [$t3] = 00000000
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 200 Clock: 1
[$s0] = 00000037  [$s1] = 00000000  [$s2] = 00000000
[$s3] = 00000000  [$s4] = 00000000  [$s5] = 00000000
[$s6] = 00000000  [$s7] = 00000000  [$t0] = 00000000
[$t1] = 00000037  [$t2] = 00000000  [$t3] = 00000000
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 250 Clock: 0
[$s0] = 00000037  [$s1] = ffffffc9  [$s2] = 00000000
[$s3] = 00000000  [$s4] = 00000000  [$s5] = 00000000
[$s6] = 00000000  [$s7] = 00000000  [$t0] = 00000000
[$t1] = 00000037  [$t2] = 00000000  [$t3] = 00000000
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 300 Clock: 1
[$s0] = 00000037  [$s1] = ffffffc9  [$s2] = 00000000
[$s3] = 00000000  [$s4] = 00000000  [$s5] = 00000000
[$s6] = 00000000  [$s7] = 00000000  [$t0] = 00000000
[$t1] = 00000037  [$t2] = 00000000  [$t3] = 00000000
```

```
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 350 Clock: 0
[$s0] = 00000037  [$s1] = fffffffc9  [$s2] = 00000000
[$s3] = 00000000  [$s4] = 00000000  [$s5] = 00000000
[$s6] = 00000000  [$s7] = 00000000  [$t0] = 00000000
[$t1] = 00000037  [$t2] = 00000000  [$t3] = 00000000
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 400 Clock: 1
[$s0] = 00000037  [$s1] = fffffffc9  [$s2] = 00000000
[$s3] = 00000000  [$s4] = 00000000  [$s5] = 00000000
[$s6] = 00000000  [$s7] = 00000000  [$t0] = 00000000
[$t1] = 00000037  [$t2] = 00000000  [$t3] = 00000000
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 450 Clock: 0
[$s0] = 00000037  [$s1] = fffffffc9  [$s2] = 00000000
[$s3] = 00000037  [$s4] = 00000000  [$s5] = 00000000
[$s6] = 00000000  [$s7] = 00000000  [$t0] = 00000000
[$t1] = 00000037  [$t2] = 00000000  [$t3] = 00000000
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 500 Clock: 1
[$s0] = 00000037  [$s1] = fffffffc9  [$s2] = 00000000
[$s3] = 00000037  [$s4] = 00000000  [$s5] = 00000000
[$s6] = 00000000  [$s7] = 00000000  [$t0] = 00000000
[$t1] = 00000037  [$t2] = 00000000  [$t3] = 00000000
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 550 Clock: 0
[$s0] = 00000037  [$s1] = fffffffc9  [$s2] = 00000000
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 00000000
[$s6] = 00000000  [$s7] = 00000000  [$t0] = 00000000
[$t1] = 00000037  [$t2] = 00000000  [$t3] = 00000000
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 600 Clock: 1
[$s0] = 00000037  [$s1] = fffffffc9  [$s2] = 00000000
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 00000000
[$s6] = 00000000  [$s7] = 00000000  [$t0] = 00000000
[$t1] = 00000037  [$t2] = 00000000  [$t3] = 00000000
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
```

```
Time : 650 Clock: 0
[$s0] = 00000037  [$s1] = ffffffc9  [$s2] = 00000000
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 00000000
[$s6] = 00000000  [$s7] = 00000000  [$t0] = 00000000
[$t1] = 00000037  [$t2] = 00000000  [$t3] = 00000000
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 700 Clock: 1
[$s0] = 00000037  [$s1] = ffffffc9  [$s2] = 00000000
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 00000000
[$s6] = 00000000  [$s7] = 00000000  [$t0] = 00000000
[$t1] = 00000037  [$t2] = 00000000  [$t3] = 00000000
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 750 Clock: 0
[$s0] = 00000037  [$s1] = ffffffc9  [$s2] = 00000000
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 00000000
[$s6] = 00000000  [$s7] = 00000000  [$t0] = 00000000
[$t1] = 00000037  [$t2] = 00000037  [$t3] = 00000000
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 800 Clock: 1
[$s0] = 00000037  [$s1] = ffffffc9  [$s2] = 00000000
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 00000000
[$s6] = 00000000  [$s7] = 00000000  [$t0] = 00000000
[$t1] = 00000037  [$t2] = 00000037  [$t3] = 00000000
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 850 Clock: 0
[$s0] = 00000037  [$s1] = ffffffc9  [$s2] = 00000000
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 00000000
[$s6] = 00000000  [$s7] = 00000000  [$t0] = 00000000
[$t1] = 00000037  [$t2] = 0000006e  [$t3] = 00000000
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 900 Clock: 1
[$s0] = 00000037  [$s1] = ffffffc9  [$s2] = 00000000
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 00000000
[$s6] = 00000000  [$s7] = 00000000  [$t0] = 00000000
[$t1] = 00000037  [$t2] = 0000006e  [$t3] = 00000000
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 950 Clock: 0
[$s0] = 00000037  [$s1] = ffffffc9  [$s2] = 00000000
```

```
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 00000000
[$s6] = 00000000  [$s7] = 00000000  [$t0] = 00000000
[$t1] = 00000037  [$t2] = 0000006e  [$t3] = 0000006e
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 1000 Clock: 1
[$s0] = 00000037  [$s1] = fffffffc9  [$s2] = 00000000
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 00000000
[$s6] = 00000000  [$s7] = 00000000  [$t0] = 00000000
[$t1] = 00000037  [$t2] = 0000006e  [$t3] = 0000006e
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
run 1000 ns
Time : 1050 Clock: 0
[$s0] = 00000037  [$s1] = fffffffc9  [$s2] = 00000000
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 00000000
[$s6] = 00000000  [$s7] = 00000000  [$t0] = 00000000
[$t1] = 00000037  [$t2] = 0000006e  [$t3] = 0000006e
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 1100 Clock: 1
[$s0] = 00000037  [$s1] = fffffffc9  [$s2] = 00000000
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 00000000
[$s6] = 00000000  [$s7] = 00000000  [$t0] = 00000000
[$t1] = 00000037  [$t2] = 0000006e  [$t3] = 0000006e
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 1150 Clock: 0
[$s0] = 00000037  [$s1] = fffffffc9  [$s2] = 00000000
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 00000000
[$s6] = 0000006e  [$s7] = 00000000  [$t0] = 00000000
[$t1] = 00000037  [$t2] = 0000006e  [$t3] = 0000006e
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 1200 Clock: 1
[$s0] = 00000037  [$s1] = fffffffc9  [$s2] = 00000000
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 00000000
[$s6] = 0000006e  [$s7] = 00000000  [$t0] = 00000000
[$t1] = 00000037  [$t2] = 0000006e  [$t3] = 0000006e
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 1250 Clock: 0
[$s0] = 00000037  [$s1] = fffffffc9  [$s2] = 00000000
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 00000000
```

```
[$s6] = 0000006e  [$s7] = 0000006e  [$t0] = 00000000
[$t1] = 00000037  [$t2] = 0000006e  [$t3] = 0000006e
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 1300 Clock: 1
[$s0] = 00000037  [$s1] = ffffffc9  [$s2] = 00000000
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 00000000
[$s6] = 0000006e  [$s7] = 0000006e  [$t0] = 00000000
[$t1] = 00000037  [$t2] = 0000006e  [$t3] = 0000006e
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 1350 Clock: 0
[$s0] = 00000037  [$s1] = ffffffc9  [$s2] = 00000000
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 00000000
[$s6] = 0000006e  [$s7] = 0000006e  [$t0] = 00000000
[$t1] = 00000037  [$t2] = 0000006e  [$t3] = 0000006e
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 1400 Clock: 1
[$s0] = 00000037  [$s1] = ffffffc9  [$s2] = 00000000
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 00000000
[$s6] = 0000006e  [$s7] = 0000006e  [$t0] = 00000000
[$t1] = 00000037  [$t2] = 0000006e  [$t3] = 0000006e
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 1450 Clock: 0
[$s0] = 00000037  [$s1] = ffffffc9  [$s2] = 00000000
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 00000000
[$s6] = 0000006e  [$s7] = 0000006e  [$t0] = 00000000
[$t1] = 00000037  [$t2] = 0000006e  [$t3] = 0000006e
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 1500 Clock: 1
[$s0] = 00000037  [$s1] = ffffffc9  [$s2] = 00000000
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 00000000
[$s6] = 0000006e  [$s7] = 0000006e  [$t0] = 00000000
[$t1] = 00000037  [$t2] = 0000006e  [$t3] = 0000006e
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 1550 Clock: 0
[$s0] = 00000037  [$s1] = ffffffc9  [$s2] = 00000000
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 00000000
[$s6] = 0000006e  [$s7] = 0000006e  [$t0] = 00000000
[$t1] = 00000037  [$t2] = 0000006e  [$t3] = 0000006e
```

```
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 1600 Clock: 1
[$s0] = 00000037  [$s1] = ffffffc9  [$s2] = 00000000
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 00000000
[$s6] = 0000006e  [$s7] = 0000006e  [$t0] = 00000000
[$t1] = 00000037  [$t2] = 0000006e  [$t3] = 0000006e
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 1650 Clock: 0
[$s0] = 00000037  [$s1] = ffffffc9  [$s2] = 00000000
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 00000000
[$s6] = 0000006e  [$s7] = 0000006e  [$t0] = 00000000
[$t1] = 00000037  [$t2] = 0000006e  [$t3] = 0000006e
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 1700 Clock: 1
[$s0] = 00000037  [$s1] = ffffffc9  [$s2] = 00000000
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 00000000
[$s6] = 0000006e  [$s7] = 0000006e  [$t0] = 00000000
[$t1] = 00000037  [$t2] = 0000006e  [$t3] = 0000006e
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 1750 Clock: 0
[$s0] = 00000037  [$s1] = ffffffc9  [$s2] = 00000000
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 00000000
[$s6] = 0000006e  [$s7] = 0000006e  [$t0] = 00000000
[$t1] = 00000037  [$t2] = 00000037  [$t3] = 0000006e
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 1800 Clock: 1
[$s0] = 00000037  [$s1] = ffffffc9  [$s2] = 00000000
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 00000000
[$s6] = 0000006e  [$s7] = 0000006e  [$t0] = 00000000
[$t1] = 00000037  [$t2] = 00000037  [$t3] = 0000006e
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 1850 Clock: 0
[$s0] = 00000037  [$s1] = ffffffc9  [$s2] = 00000000
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 00000000
[$s6] = 0000006e  [$s7] = 0000006e  [$t0] = 00000000
[$t1] = 00000037  [$t2] = 0000006e  [$t3] = 0000006e
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
```

```
Time : 1900 Clock: 1
[$s0] = 00000037  [$s1] = fffffffc9  [$s2] = 00000000
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 00000000
[$s6] = 0000006e  [$s7] = 0000006e  [$t0] = 00000000
[$t1] = 00000037  [$t2] = 0000006e  [$t3] = 0000006e
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 1950 Clock: 0
[$s0] = 00000037  [$s1] = fffffffc9  [$s2] = 00000000
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 00000000
[$s6] = 0000006e  [$s7] = 0000006e  [$t0] = 00000000
[$t1] = 00000037  [$t2] = 0000006e  [$t3] = 0000006e
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 2000 Clock: 1
[$s0] = 00000037  [$s1] = fffffffc9  [$s2] = 00000000
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 00000000
[$s6] = 0000006e  [$s7] = 0000006e  [$t0] = 00000000
[$t1] = 00000037  [$t2] = 0000006e  [$t3] = 0000006e
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
run 1000 ns
Time : 2050 Clock: 0
[$s0] = 00000037  [$s1] = fffffffc9  [$s2] = 00000000
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 00000000
[$s6] = 0000006e  [$s7] = 0000006e  [$t0] = 00000000
[$t1] = 00000037  [$t2] = 0000006e  [$t3] = 0000006e
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 2100 Clock: 1
[$s0] = 00000037  [$s1] = fffffffc9  [$s2] = 00000000
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 00000000
[$s6] = 0000006e  [$s7] = 0000006e  [$t0] = 00000000
[$t1] = 00000037  [$t2] = 0000006e  [$t3] = 0000006e
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 2150 Clock: 0
[$s0] = 00000037  [$s1] = fffffffc9  [$s2] = 00000000
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 00000000
[$s6] = 0000006e  [$s7] = 0000006e  [$t0] = 00000000
[$t1] = 00000037  [$t2] = 0000006e  [$t3] = 0000006e
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 2200 Clock: 1
```

```
[$s0] = 00000037  [$s1] = fffffffc9  [$s2] = 00000000
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 00000000
[$s6] = 0000006e  [$s7] = 0000006e  [$t0] = 00000000
[$t1] = 00000037  [$t2] = 0000006e  [$t3] = 0000006e
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 2250 Clock: 0
[$s0] = 00000037  [$s1] = fffffffc9  [$s2] = 00000000
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 00000000
[$s6] = 0000006e  [$s7] = 0000006e  [$t0] = 00000000
[$t1] = 00000037  [$t2] = 0000006e  [$t3] = 0000006e
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 2300 Clock: 1
[$s0] = 00000037  [$s1] = fffffffc9  [$s2] = 00000000
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 00000000
[$s6] = 0000006e  [$s7] = 0000006e  [$t0] = 00000000
[$t1] = 00000037  [$t2] = 0000006e  [$t3] = 0000006e
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 2350 Clock: 0
[$s0] = 00000037  [$s1] = fffffffc9  [$s2] = 00000000
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 00000000
[$s6] = 0000006e  [$s7] = 0000006e  [$t0] = 00000000
[$t1] = 00000037  [$t2] = 0000006e  [$t3] = 0000006e
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 2400 Clock: 1
[$s0] = 00000037  [$s1] = fffffffc9  [$s2] = 00000000
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 00000000
[$s6] = 0000006e  [$s7] = 0000006e  [$t0] = 00000000
[$t1] = 00000037  [$t2] = 0000006e  [$t3] = 0000006e
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 2450 Clock: 0
[$s0] = 00000037  [$s1] = fffffffc9  [$s2] = 00000000
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 00000000
[$s6] = 0000006e  [$s7] = 0000006e  [$t0] = 00000000
[$t1] = 00000037  [$t2] = 0000006e  [$t3] = 0000006e
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 2500 Clock: 1
[$s0] = 00000037  [$s1] = fffffffc9  [$s2] = 00000000
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 00000000
```

```
[$s6] = 0000006e  [$s7] = 0000006e  [$t0] = 00000000
[$t1] = 00000037  [$t2] = 0000006e  [$t3] = 0000006e
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 2550 Clock: 0
[$s0] = 00000037  [$s1] = ffffffc9  [$s2] = 00000000
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 00000000
[$s6] = 0000006e  [$s7] = 0000006e  [$t0] = 00000000
[$t1] = 00000037  [$t2] = 0000006e  [$t3] = 0000006e
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 2600 Clock: 1
[$s0] = 00000037  [$s1] = ffffffc9  [$s2] = 00000000
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 00000000
[$s6] = 0000006e  [$s7] = 0000006e  [$t0] = 00000000
[$t1] = 00000037  [$t2] = 0000006e  [$t3] = 0000006e
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 2650 Clock: 0
[$s0] = 00000037  [$s1] = ffffffc9  [$s2] = 00000000
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 00000000
[$s6] = 0000006e  [$s7] = 0000006e  [$t0] = 00000000
[$t1] = 00000037  [$t2] = 0000006e  [$t3] = 0000006e
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 2700 Clock: 1
[$s0] = 00000037  [$s1] = ffffffc9  [$s2] = 00000000
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 00000000
[$s6] = 0000006e  [$s7] = 0000006e  [$t0] = 00000000
[$t1] = 00000037  [$t2] = 0000006e  [$t3] = 0000006e
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 2750 Clock: 0
[$s0] = 00000037  [$s1] = ffffffc9  [$s2] = 00000000
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 00000000
[$s6] = 0000006e  [$s7] = 0000006e  [$t0] = 00000000
[$t1] = 00000037  [$t2] = 00000037  [$t3] = 0000006e
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 2800 Clock: 1
[$s0] = 00000037  [$s1] = ffffffc9  [$s2] = 00000000
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 00000000
[$s6] = 0000006e  [$s7] = 0000006e  [$t0] = 00000000
[$t1] = 00000037  [$t2] = 00000037  [$t3] = 0000006e
```

```
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 2850 Clock: 0
[$s0] = 00000037  [$s1] = fffffffc9  [$s2] = 00000000
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 00000000
[$s6] = 0000006e  [$s7] = 0000006e  [$t0] = 00000000
[$t1] = 00000037  [$t2] = 0000006e  [$t3] = 0000006e
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 2900 Clock: 1
[$s0] = 00000037  [$s1] = fffffffc9  [$s2] = 00000000
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 00000000
[$s6] = 0000006e  [$s7] = 0000006e  [$t0] = 00000000
[$t1] = 00000037  [$t2] = 0000006e  [$t3] = 0000006e
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 2950 Clock: 0
[$s0] = 00000037  [$s1] = fffffffc9  [$s2] = 00000000
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 00000000
[$s6] = 0000006e  [$s7] = 0000006e  [$t0] = 00000000
[$t1] = 00000037  [$t2] = 0000006e  [$t3] = 0000006e
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
```

IV. Pipeline Simulation Result

```
module testbench;
    parameter half_period = 50;
    reg clock;
    wire [31:0] s0, s1,
s2,s3,s4,s5,s6,s7,t0,t1,t2,t3,t4,t5,t6,t7,t8,t9;
    pipeline UUT(clock, s0, s1,
s2,s3,s4,s5,s6,s7,t0,t1,t2,t3,t4,t5,t6,t7,t8,t9);

    initial begin
        $display("************************************************");
        $display("The textual simulation results:");
        $display("************************************************");
        $monitor("Time : %0d Clock: %d \n[$s0] = %h  [$s1] = %h  [$s2]
= %h \n[$s3] = %h  [$s4] = %h  [$s5] = %h \n[$s6] = %h  [$s7] = %h
[$t0] = %h \n[$t1] = %h  [$t2] = %h  [$t3] = %h \n[$t4] = %h  [$t5]
= %h  [$t6] = %h \n[$t7] = %h  [$t8] = %h  [$t9] = %h \n",
            $time, clock, s0, s1,
s2,s3,s4,s5,s6,s7,t0,t1,t2,t3,t4,t5,t6,t7,t8,t9);
```

```
    end

    initial begin
        #0 clock = 1;
    end

    always #half_period clock = ~clock;

    initial #3000 $stop;
endmodule
*************************************************
The textual simulation results:
*************************************************
Finished circuit initialization process.
Time : 0 Clock: 1
[$s0] = 00000000  [$s1] = 00000000  [$s2] = 00000000
[$s3] = 00000000  [$s4] = 00000000  [$s5] = 00000000
[$s6] = 00000000  [$s7] = 00000000  [$t0] = 00000000
[$t1] = 00000000  [$t2] = 00000000  [$t3] = 00000000
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 50 Clock: 0
[$s0] = 00000000  [$s1] = 00000000  [$s2] = 00000000
[$s3] = 00000000  [$s4] = 00000000  [$s5] = 00000000
[$s6] = 00000000  [$s7] = 00000000  [$t0] = 00000000
[$t1] = 00000000  [$t2] = 00000000  [$t3] = 00000000
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 100 Clock: 1
[$s0] = 00000000  [$s1] = 00000000  [$s2] = 00000000
[$s3] = 00000000  [$s4] = 00000000  [$s5] = 00000000
[$s6] = 00000000  [$s7] = 00000000  [$t0] = 00000000
[$t1] = 00000000  [$t2] = 00000000  [$t3] = 00000000
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 150 Clock: 0
[$s0] = 00000000  [$s1] = 00000000  [$s2] = 00000000
[$s3] = 00000000  [$s4] = 00000000  [$s5] = 00000000
[$s6] = 00000000  [$s7] = 00000000  [$t0] = 00000000
[$t1] = 00000000  [$t2] = 00000000  [$t3] = 00000000
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 200 Clock: 1
[$s0] = 00000000  [$s1] = 00000000  [$s2] = 00000000
```

```
[$s3] = 00000000  [$s4] = 00000000  [$s5] = 00000000
[$s6] = 00000000  [$s7] = 00000000  [$t0] = 00000000
[$t1] = 00000000  [$t2] = 00000000  [$t3] = 00000000
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 250 Clock: 0
[$s0] = 00000000  [$s1] = 00000000  [$s2] = 00000000
[$s3] = 00000000  [$s4] = 00000000  [$s5] = 00000000
[$s6] = 00000000  [$s7] = 00000000  [$t0] = 00000000
[$t1] = 00000000  [$t2] = 00000000  [$t3] = 00000000
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 300 Clock: 1
[$s0] = 00000000  [$s1] = 00000000  [$s2] = 00000000
[$s3] = 00000000  [$s4] = 00000000  [$s5] = 00000000
[$s6] = 00000000  [$s7] = 00000000  [$t0] = 00000036
[$t1] = 00000000  [$t2] = 00000000  [$t3] = 00000000
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 350 Clock: 0
[$s0] = 00000000  [$s1] = 00000000  [$s2] = 00000000
[$s3] = 00000000  [$s4] = 00000000  [$s5] = 00000000
[$s6] = 00000000  [$s7] = 00000000  [$t0] = 00000036
[$t1] = 00000000  [$t2] = 00000000  [$t3] = 00000000
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 400 Clock: 1
[$s0] = 00000000  [$s1] = 00000000  [$s2] = 00000000
[$s3] = 00000000  [$s4] = 00000000  [$s5] = 00000000
[$s6] = 00000000  [$s7] = 00000000  [$t0] = 00000036
[$t1] = 00000037  [$t2] = 00000000  [$t3] = 00000000
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 450 Clock: 0
[$s0] = 00000000  [$s1] = 00000000  [$s2] = 00000000
[$s3] = 00000000  [$s4] = 00000000  [$s5] = 00000000
[$s6] = 00000000  [$s7] = 00000000  [$t0] = 00000036
[$t1] = 00000037  [$t2] = 00000000  [$t3] = 00000000
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 500 Clock: 1
[$s0] = 0000006d  [$s1] = 00000000  [$s2] = 00000000
[$s3] = 00000000  [$s4] = 00000000  [$s5] = 00000000
[$s6] = 00000000  [$s7] = 00000000  [$t0] = 00000036
```

```
[$t1] = 00000037  [$t2] = 00000000  [$t3] = 00000000
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 550 Clock: 0
[$s0] = 0000006d  [$s1] = 00000000  [$s2] = 00000000
[$s3] = 00000000  [$s4] = 00000000  [$s5] = 00000000
[$s6] = 00000000  [$s7] = 00000000  [$t0] = 00000036
[$t1] = 00000037  [$t2] = 00000000  [$t3] = 00000000
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 600 Clock: 1
[$s0] = 0000006d  [$s1] = ffffffff  [$s2] = 00000000
[$s3] = 00000000  [$s4] = 00000000  [$s5] = 00000000
[$s6] = 00000000  [$s7] = 00000000  [$t0] = 00000036
[$t1] = 00000037  [$t2] = 00000000  [$t3] = 00000000
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 650 Clock: 0
[$s0] = 0000006d  [$s1] = ffffffff  [$s2] = 00000000
[$s3] = 00000000  [$s4] = 00000000  [$s5] = 00000000
[$s6] = 00000000  [$s7] = 00000000  [$t0] = 00000036
[$t1] = 00000037  [$t2] = 00000000  [$t3] = 00000000
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 700 Clock: 1
[$s0] = 0000006d  [$s1] = ffffffff  [$s2] = 00000036
[$s3] = 00000000  [$s4] = 00000000  [$s5] = 00000000
[$s6] = 00000000  [$s7] = 00000000  [$t0] = 00000036
[$t1] = 00000037  [$t2] = 00000000  [$t3] = 00000000
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 750 Clock: 0
[$s0] = 0000006d  [$s1] = ffffffff  [$s2] = 00000036
[$s3] = 00000000  [$s4] = 00000000  [$s5] = 00000000
[$s6] = 00000000  [$s7] = 00000000  [$t0] = 00000036
[$t1] = 00000037  [$t2] = 00000000  [$t3] = 00000000
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 800 Clock: 1
[$s0] = 0000006d  [$s1] = ffffffff  [$s2] = 00000036
[$s3] = 00000037  [$s4] = 00000000  [$s5] = 00000000
[$s6] = 00000000  [$s7] = 00000000  [$t0] = 00000036
[$t1] = 00000037  [$t2] = 00000000  [$t3] = 00000000
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
```

```
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 850 Clock: 0
[$s0] = 0000006d  [$s1] = ffffffff  [$s2] = 00000036
[$s3] = 00000037  [$s4] = 00000000  [$s5] = 00000000
[$s6] = 00000000  [$s7] = 00000000  [$t0] = 00000036
[$t1] = 00000037  [$t2] = 00000000  [$t3] = 00000000
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 900 Clock: 1
[$s0] = 0000006d  [$s1] = ffffffff  [$s2] = 00000036
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 00000000
[$s6] = 00000000  [$s7] = 00000000  [$t0] = 00000036
[$t1] = 00000037  [$t2] = 00000000  [$t3] = 00000000
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 950 Clock: 0
[$s0] = 0000006d  [$s1] = ffffffff  [$s2] = 00000036
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 00000000
[$s6] = 00000000  [$s7] = 00000000  [$t0] = 00000036
[$t1] = 00000037  [$t2] = 00000000  [$t3] = 00000000
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 1000 Clock: 1
[$s0] = 0000006d  [$s1] = ffffffff  [$s2] = 00000036
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 00000000
[$s6] = 00000000  [$s7] = 00000000  [$t0] = 00000036
[$t1] = 00000037  [$t2] = 00000000  [$t3] = 00000000
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
run 1000 ns
Time : 1050 Clock: 0
[$s0] = 0000006d  [$s1] = ffffffff  [$s2] = 00000036
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 00000000
[$s6] = 00000000  [$s7] = 00000000  [$t0] = 00000036
[$t1] = 00000037  [$t2] = 00000000  [$t3] = 00000000
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 1100 Clock: 1
[$s0] = 0000006d  [$s1] = ffffffff  [$s2] = 00000036
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 00000000
[$s6] = 00000000  [$s7] = 00000000  [$t0] = 00000036
[$t1] = 00000037  [$t2] = 0000006d  [$t3] = 00000000
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
```

```
Time : 1150 Clock: 0
[$s0] = 0000006d  [$s1] = ffffffff  [$s2] = 00000036
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 00000000
[$s6] = 00000000  [$s7] = 00000000  [$t0] = 00000036
[$t1] = 00000037  [$t2] = 0000006d  [$t3] = 00000000
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 1200 Clock: 1
[$s0] = 0000006d  [$s1] = ffffffff  [$s2] = 00000036
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 00000000
[$s6] = 00000000  [$s7] = 00000000  [$t0] = 00000036
[$t1] = 00000037  [$t2] = 000000a4  [$t3] = 00000000
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 1250 Clock: 0
[$s0] = 0000006d  [$s1] = ffffffff  [$s2] = 00000036
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 00000000
[$s6] = 00000000  [$s7] = 00000000  [$t0] = 00000036
[$t1] = 00000037  [$t2] = 000000a4  [$t3] = 00000000
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 1300 Clock: 1
[$s0] = 0000006d  [$s1] = ffffffff  [$s2] = 00000036
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 00000000
[$s6] = 00000000  [$s7] = 00000000  [$t0] = 00000036
[$t1] = 00000037  [$t2] = 000000a4  [$t3] = 000000da
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 1350 Clock: 0
[$s0] = 0000006d  [$s1] = ffffffff  [$s2] = 00000036
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 00000000
[$s6] = 00000000  [$s7] = 00000000  [$t0] = 00000036
[$t1] = 00000037  [$t2] = 000000a4  [$t3] = 000000da
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 1400 Clock: 1
[$s0] = 0000006d  [$s1] = ffffffff  [$s2] = 00000036
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 00000000
[$s6] = 00000000  [$s7] = 00000000  [$t0] = 00000036
[$t1] = 00000037  [$t2] = 000000a4  [$t3] = 000000da
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 1450 Clock: 0
[$s0] = 0000006d  [$s1] = ffffffff  [$s2] = 00000036
```

```
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 00000000
[$s6] = 00000000  [$s7] = 00000000  [$t0] = 00000036
[$t1] = 00000037  [$t2] = 000000a4  [$t3] = 000000da
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 1500 Clock: 1
[$s0] = 0000006d  [$s1] = ffffffff  [$s2] = 00000036
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 00000000
[$s6] = 00000000  [$s7] = 00000000  [$t0] = 00000036
[$t1] = 00000037  [$t2] = 000000a4  [$t3] = 000000da
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 1550 Clock: 0
[$s0] = 0000006d  [$s1] = ffffffff  [$s2] = 00000036
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 00000000
[$s6] = 00000000  [$s7] = 00000000  [$t0] = 00000036
[$t1] = 00000037  [$t2] = 000000a4  [$t3] = 000000da
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 1600 Clock: 1
[$s0] = 0000006d  [$s1] = ffffffff  [$s2] = 00000036
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 00000000
[$s6] = 000000a4  [$s7] = 00000000  [$t0] = 00000036
[$t1] = 00000037  [$t2] = 000000a4  [$t3] = 000000da
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 1650 Clock: 0
[$s0] = 0000006d  [$s1] = ffffffff  [$s2] = 00000036
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 00000000
[$s6] = 000000a4  [$s7] = 00000000  [$t0] = 00000036
[$t1] = 00000037  [$t2] = 000000a4  [$t3] = 000000da
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 1700 Clock: 1
[$s0] = 0000006d  [$s1] = ffffffff  [$s2] = 00000036
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 00000000
[$s6] = 000000a4  [$s7] = 000000da  [$t0] = 00000036
[$t1] = 00000037  [$t2] = 000000a4  [$t3] = 000000da
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 1750 Clock: 0
[$s0] = 0000006d  [$s1] = ffffffff  [$s2] = 00000036
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 00000000
[$s6] = 000000a4  [$s7] = 000000da  [$t0] = 00000036
```

```
[$t1] = 00000037  [$t2] = 000000a4  [$t3] = 000000da
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 1800 Clock: 1
[$s0] = 0000006d  [$s1] = ffffffff  [$s2] = 00000036
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 00000000
[$s6] = 000000a4  [$s7] = 000000da  [$t0] = 00000036
[$t1] = 00000037  [$t2] = 000000a4  [$t3] = 000000da
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 1850 Clock: 0
[$s0] = 0000006d  [$s1] = ffffffff  [$s2] = 00000036
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 00000000
[$s6] = 000000a4  [$s7] = 000000da  [$t0] = 00000036
[$t1] = 00000037  [$t2] = 000000a4  [$t3] = 000000da
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 1900 Clock: 1
[$s0] = 0000006d  [$s1] = ffffffff  [$s2] = 00000036
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 000000da
[$s6] = 000000a4  [$s7] = 000000da  [$t0] = 00000036
[$t1] = 00000037  [$t2] = 000000a4  [$t3] = 000000da
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 1950 Clock: 0
[$s0] = 0000006d  [$s1] = ffffffff  [$s2] = 00000036
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 000000da
[$s6] = 000000a4  [$s7] = 000000da  [$t0] = 00000036
[$t1] = 00000037  [$t2] = 000000a4  [$t3] = 000000da
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 2000 Clock: 1
[$s0] = 0000006d  [$s1] = ffffffff  [$s2] = 00000036
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 000000da
[$s6] = 000000a4  [$s7] = 000000da  [$t0] = 00000037
[$t1] = 00000037  [$t2] = 000000a4  [$t3] = 000000da
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
run 1000 ns
Time : 2050 Clock: 0
[$s0] = 0000006d  [$s1] = ffffffff  [$s2] = 00000036
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 000000da
[$s6] = 000000a4  [$s7] = 000000da  [$t0] = 00000037
[$t1] = 00000037  [$t2] = 000000a4  [$t3] = 000000da
```

```
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 2100 Clock: 1
[$s0] = 0000006d  [$s1] = ffffffff  [$s2] = 00000036
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 000000da
[$s6] = 000000a4  [$s7] = 000000da  [$t0] = 00000037
[$t1] = 00000037  [$t2] = 000000a4  [$t3] = 000000da
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 2150 Clock: 0
[$s0] = 0000006d  [$s1] = ffffffff  [$s2] = 00000036
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 000000da
[$s6] = 000000a4  [$s7] = 000000da  [$t0] = 00000037
[$t1] = 00000037  [$t2] = 000000a4  [$t3] = 000000da
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 2200 Clock: 1
[$s0] = 0000006d  [$s1] = ffffffff  [$s2] = 00000036
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 000000da
[$s6] = 000000a4  [$s7] = 000000da  [$t0] = 00000037
[$t1] = 00000037  [$t2] = 000000a4  [$t3] = 000000da
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 2250 Clock: 0
[$s0] = 0000006d  [$s1] = ffffffff  [$s2] = 00000036
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 000000da
[$s6] = 000000a4  [$s7] = 000000da  [$t0] = 00000037
[$t1] = 00000037  [$t2] = 000000a4  [$t3] = 000000da
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 2300 Clock: 1
[$s0] = 0000006d  [$s1] = ffffffff  [$s2] = 00000036
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 000000da
[$s6] = 000000a4  [$s7] = 000000da  [$t0] = 00000037
[$t1] = 00000037  [$t2] = 000000a4  [$t3] = 000000da
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 2350 Clock: 0
[$s0] = 0000006d  [$s1] = ffffffff  [$s2] = 00000036
[$s3] = 00000037  [$s4] = 00000001  [$s5] = 000000da
[$s6] = 000000a4  [$s7] = 000000da  [$t0] = 00000037
[$t1] = 00000037  [$t2] = 000000a4  [$t3] = 000000da
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
```

```
Time : 2400 Clock: 1
[$s0] = 0000006d  [$s1] = ffffffff  [$s2] = 00000036
[$s3] = 00000037  [$s4] = 00000000  [$s5] = 000000da
[$s6] = 000000a4  [$s7] = 000000da  [$t0] = 00000037
[$t1] = 00000037  [$t2] = 000000a4  [$t3] = 000000da
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 2450 Clock: 0
[$s0] = 0000006d  [$s1] = ffffffff  [$s2] = 00000036
[$s3] = 00000037  [$s4] = 00000000  [$s5] = 000000da
[$s6] = 000000a4  [$s7] = 000000da  [$t0] = 00000037
[$t1] = 00000037  [$t2] = 000000a4  [$t3] = 000000da
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 2500 Clock: 1
[$s0] = 0000006d  [$s1] = ffffffff  [$s2] = 00000036
[$s3] = 00000037  [$s4] = 00000000  [$s5] = 000000da
[$s6] = 000000a4  [$s7] = 000000da  [$t0] = 00000037
[$t1] = 00000037  [$t2] = 000000a4  [$t3] = 000000da
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 2550 Clock: 0
[$s0] = 0000006d  [$s1] = ffffffff  [$s2] = 00000036
[$s3] = 00000037  [$s4] = 00000000  [$s5] = 000000da
[$s6] = 000000a4  [$s7] = 000000da  [$t0] = 00000037
[$t1] = 00000037  [$t2] = 000000a4  [$t3] = 000000da
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 2600 Clock: 1
[$s0] = 0000006d  [$s1] = ffffffff  [$s2] = 00000036
[$s3] = 00000037  [$s4] = 00000000  [$s5] = 000000da
[$s6] = 000000a4  [$s7] = 000000da  [$t0] = 00000037
[$t1] = 00000037  [$t2] = 000000a4  [$t3] = 000000da
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 2650 Clock: 0
[$s0] = 0000006d  [$s1] = ffffffff  [$s2] = 00000036
[$s3] = 00000037  [$s4] = 00000000  [$s5] = 000000da
[$s6] = 000000a4  [$s7] = 000000da  [$t0] = 00000037
[$t1] = 00000037  [$t2] = 000000a4  [$t3] = 000000da
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 2700 Clock: 1
[$s0] = 0000006d  [$s1] = ffffffff  [$s2] = 00000036
```

```
[$s3] = 00000037  [$s4] = 00000000  [$s5] = 000000da
[$s6] = 000000a4  [$s7] = 000000da  [$t0] = 00000037
[$t1] = 00000037  [$t2] = 000000a4  [$t3] = 000000da
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 2750 Clock: 0
[$s0] = 0000006d  [$s1] = ffffffff  [$s2] = 00000036
[$s3] = 00000037  [$s4] = 00000000  [$s5] = 000000da
[$s6] = 000000a4  [$s7] = 000000da  [$t0] = 00000037
[$t1] = 00000037  [$t2] = 000000a4  [$t3] = 000000da
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 2800 Clock: 1
[$s0] = 0000006d  [$s1] = ffffffff  [$s2] = 00000036
[$s3] = 00000037  [$s4] = 00000000  [$s5] = 000000da
[$s6] = 000000a4  [$s7] = 000000da  [$t0] = 00000037
[$t1] = 00000037  [$t2] = 000000a4  [$t3] = 000000da
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 2850 Clock: 0
[$s0] = 0000006d  [$s1] = ffffffff  [$s2] = 00000036
[$s3] = 00000037  [$s4] = 00000000  [$s5] = 000000da
[$s6] = 000000a4  [$s7] = 000000da  [$t0] = 00000037
[$t1] = 00000037  [$t2] = 000000a4  [$t3] = 000000da
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 2900 Clock: 1
[$s0] = 0000006d  [$s1] = ffffffff  [$s2] = 00000036
[$s3] = 00000037  [$s4] = 00000000  [$s5] = 000000da
[$s6] = 000000a4  [$s7] = 000000da  [$t0] = 00000038
[$t1] = 00000037  [$t2] = 000000a4  [$t3] = 000000da
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
Time : 2950 Clock: 0
[$s0] = 0000006d  [$s1] = ffffffff  [$s2] = 00000036
[$s3] = 00000037  [$s4] = 00000000  [$s5] = 000000da
[$s6] = 000000a4  [$s7] = 000000da  [$t0] = 00000038
[$t1] = 00000037  [$t2] = 000000a4  [$t3] = 000000da
[$t4] = 00000000  [$t5] = 00000000  [$t6] = 00000000
[$t7] = 00000000  [$t8] = 00000000  [$t9] = 00000000
```