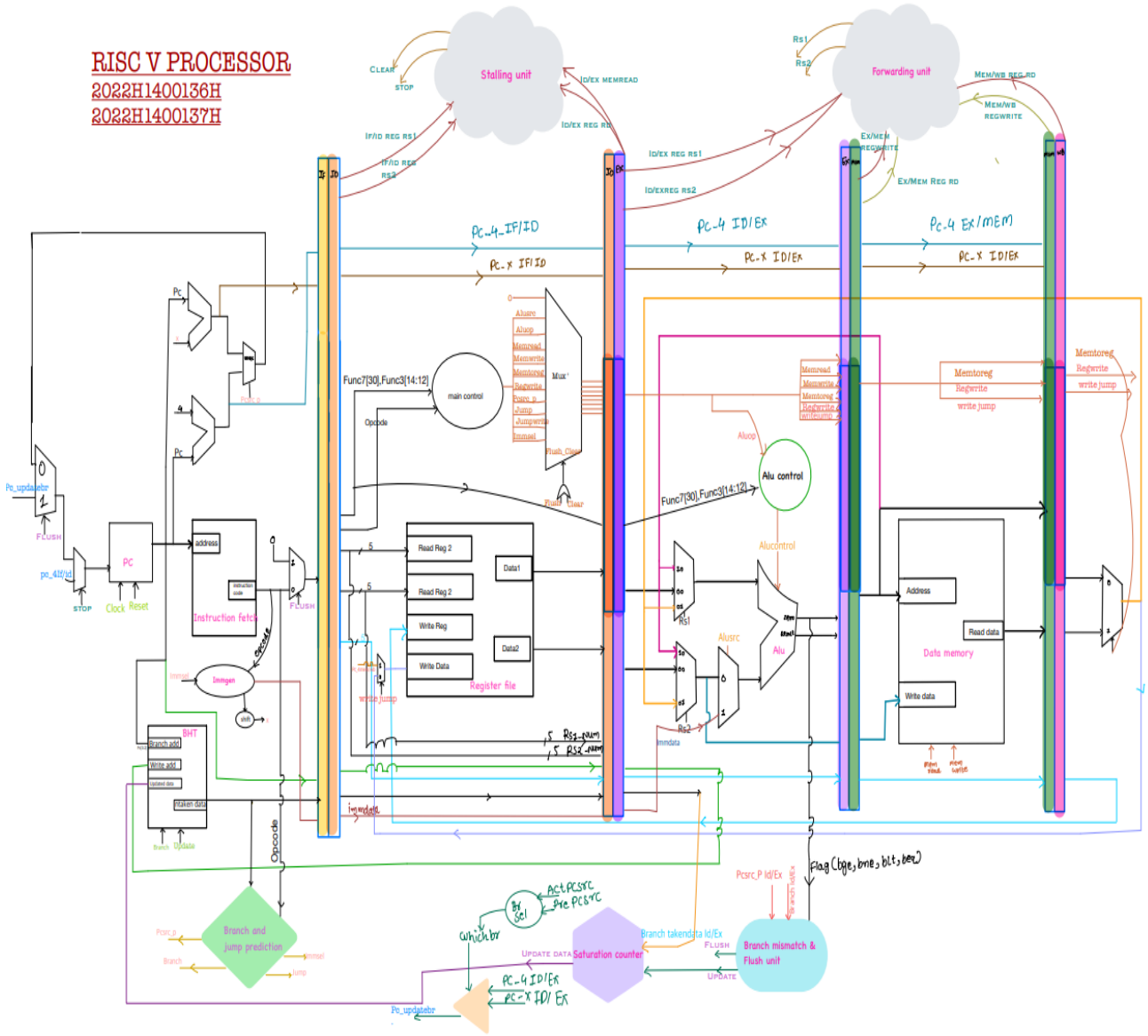
ASSIGNMENT 2:

Group no:6

Submitted By: *Soni Jay Ashokbhai(2022H1400136H)*
Jani Urvish Pankajbhai (2022H1400137H)

- **Implement Pipelined RISC-V integer/floating point processor (partially) such that it supports three different categories of instructions as mentioned below: 1. ALU Instructions (R-Type instructions) also called as computational instructions 2. Memory Instructions (load and Store) 3. Control Transfer Instructions (should include conditional and unconditional branch instructions) • The processor should support at least 12 instructions under the above specified categories. • This processor should support forwarding and stalling to avoid data hazards. • This processor should also support a 4-entry 2-bit saturation counter type branch predictor and flushing when the prediction is wrong.**

❖ ***Block level diagram of Processor:***



❖ ***Separate waveforms under data hazard and control hazard conditions***

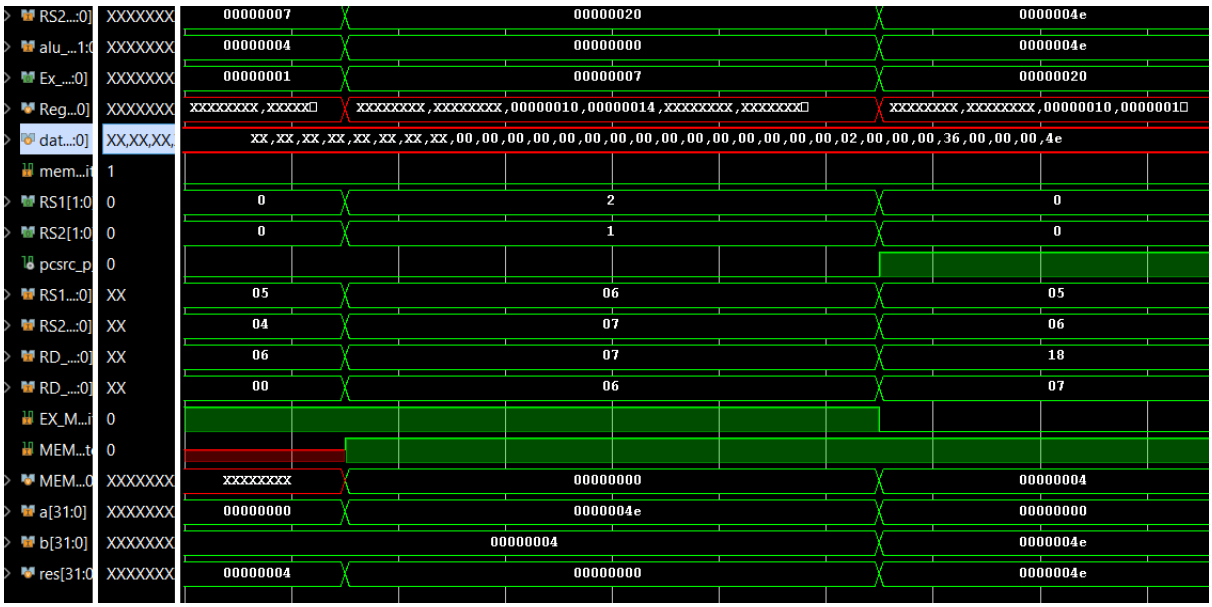
Forwarding:

1: Ex hazard and Mem hazards

add t3,t1,t2 t3=10+130=140

add t4,t2,t3 t4=130+140=270

add t5,t3,t4

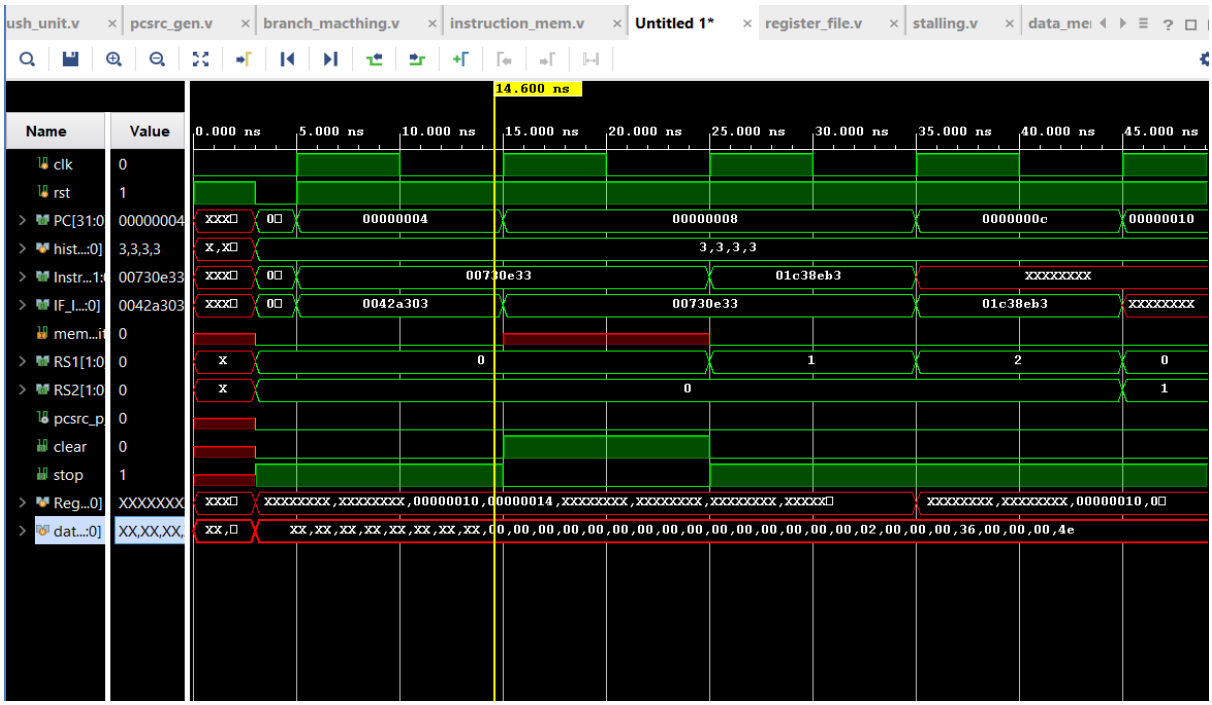


2: Stalling

lw t1,4(t0) t1=10

add t3,t1,t2 t3=10+130=140

add t4,t2,t3 t4=130+140=270

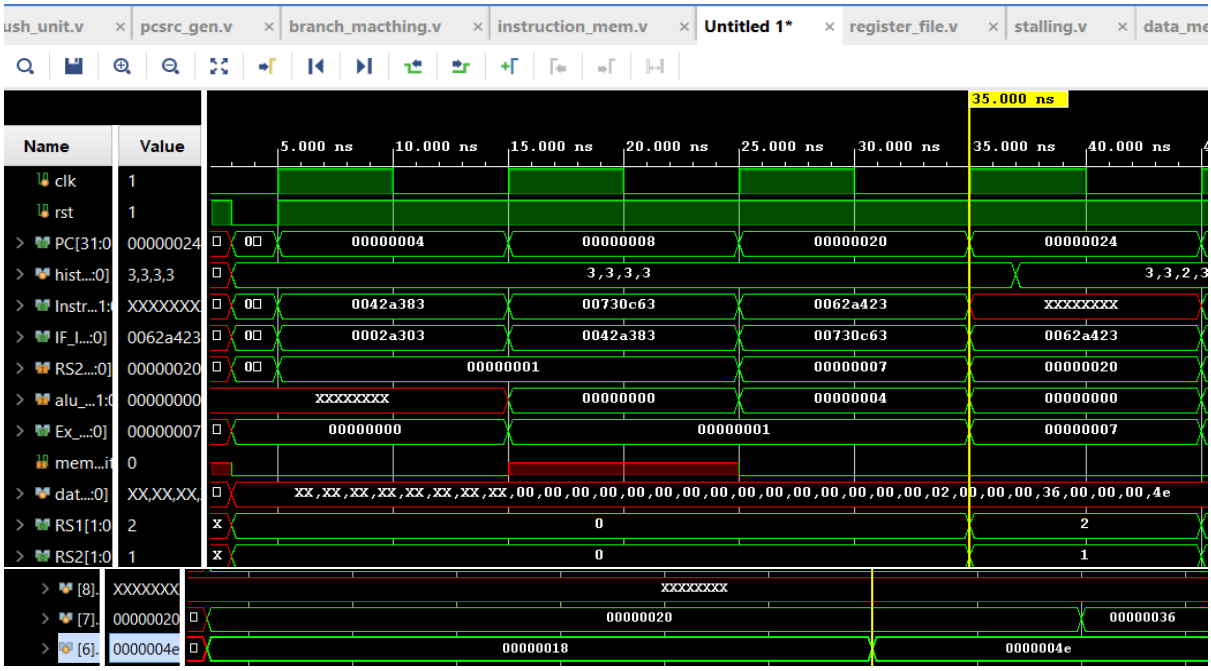


❖ ***Control signal values for all the 12 instructions and Separate waveforms showing the execution of the 12 instructions (show the changes in register file/memory/PC etc.)***

Load And Strore

1. load inst (lw t2,4(t0))

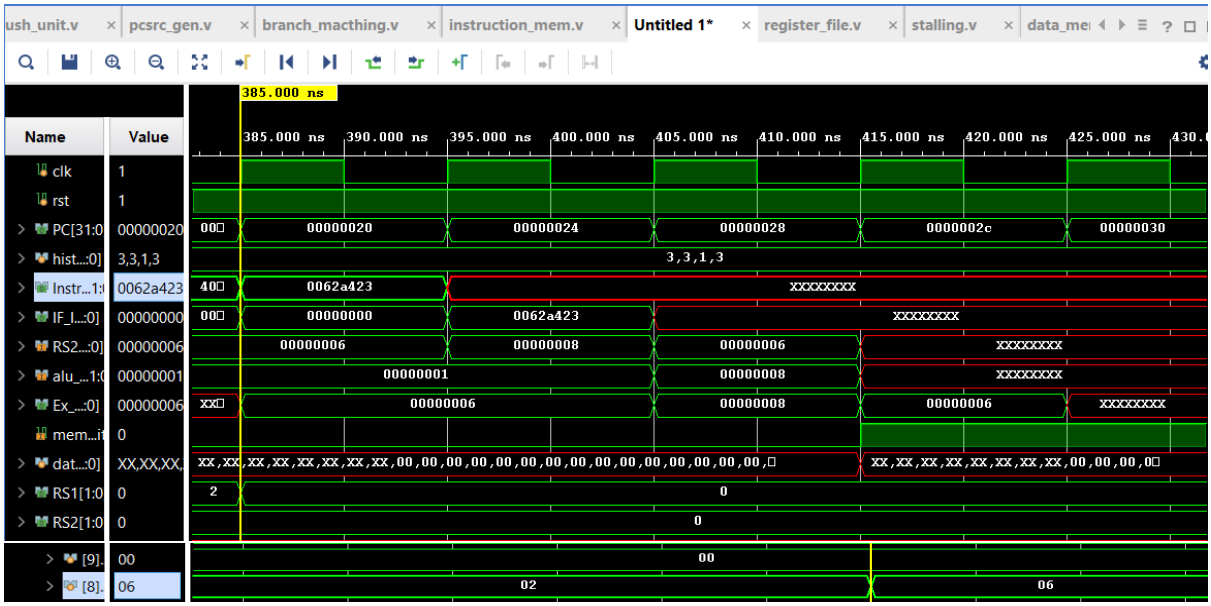
immsel=2'b00;
regwrite=1; //writing to reg
alusrc=1; //select sign extended one
memread=1; //memread 1
memwrite=0;
memtoreg=1; //mem to a register
pcsrc=0; //pc=pc+4
aluop=2'b00; //lw
jal=1'b0;
and according to aluop=2'b00,
AluControl=(funclines[2:0]==3'b010)?4'b0001:4'bxxxx;//lw and sw instructioni.e alu will perform add operation



2. sw t1,8(t0)

immsel=2'b01; //denotes {{20{InstructCode[31]}},InstructCode[31:25],InstructCode[11:7]}
regwrite=0; //not writing to reg
alusrc=1; //select sign extended one
memread=0; //memread 1
memwrite=1; //writing to mem
memtoreg=1'bx; //don't care mem to a register
pcsrc=0; //pc=pc+4
aluop=2'b00; //sw
jal=1'bx; and according to aluop=2'b00,
AluControl=(funclines[2:0]==3'b010)?4'b0001:4'bxxxx; //lw and sw instruction

i.e alu will perform add operation



Branch Instructions

3. blt t2,t1,subad (t2=#20,t1=#68)(Branch if less than)

immsel=2'b10; //denotes {{20{InstructCode[31]}},InstructCode[31],InstructCode[7],InstructCode[30:25],InstructCode[11:8]};

regwrite=0;

alusrc=0; //select data2

memread=0;

memwrite=0;

memtoreg=1'bx; //direct alu data to be written in reg file

case(fun3) //pc=pc+x

3'b000: pcsrc= zero; //beq

3'b001: pcsrc= ~zero; //bneq

3'b100: pcsrc= lt; //blt

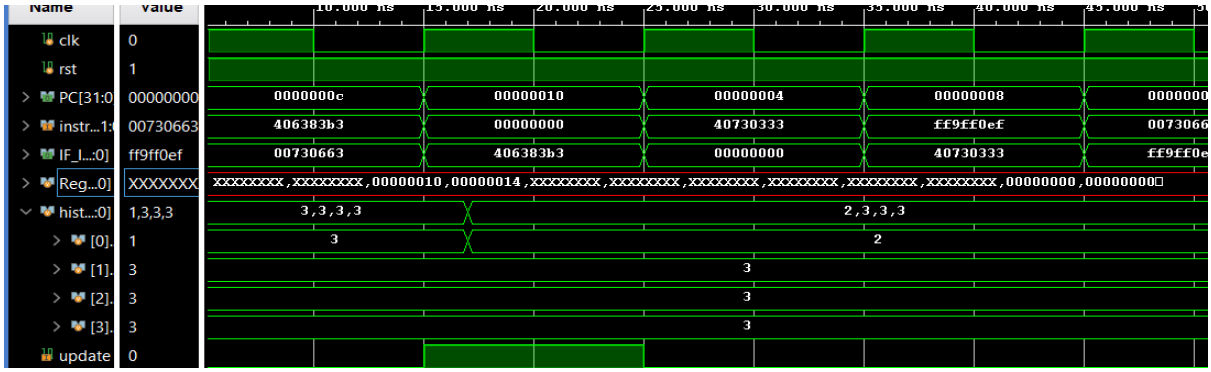
3'b101: pcsrc= gt; //bge

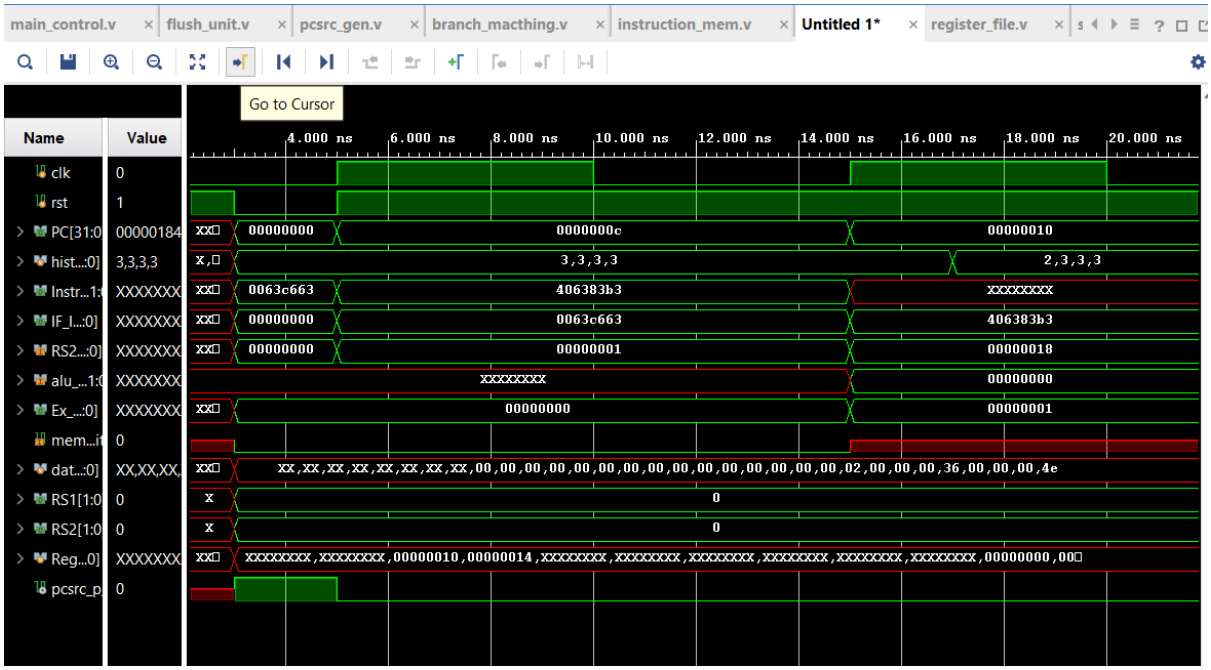
endcase

aluop=2'b01; //Btype

jal=1'bx;

#BHT



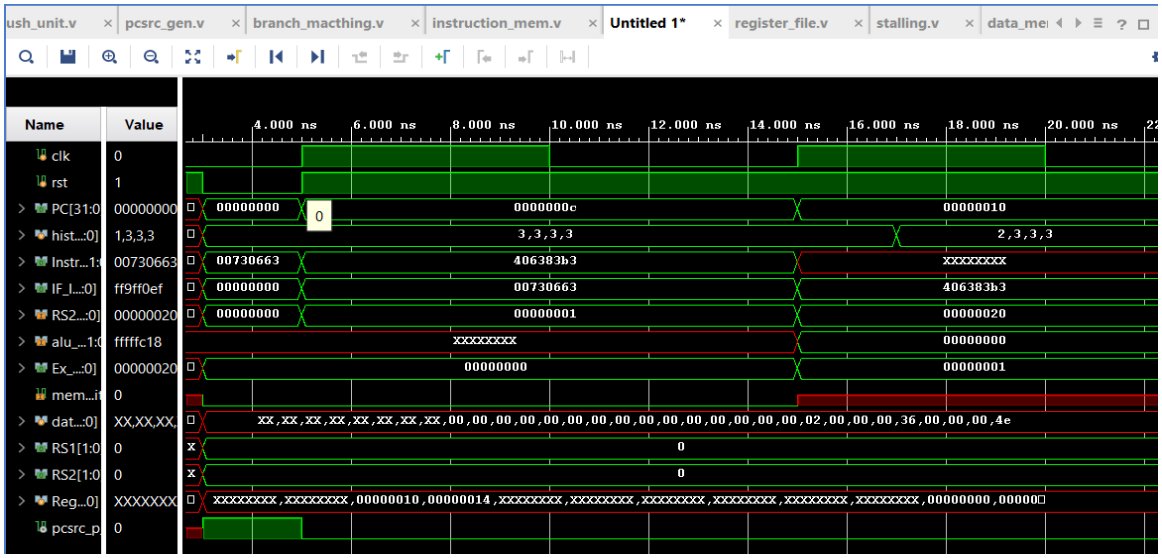


4.beq t1,t2,l1(Branch If equal)(t1=#8,t2=#8)

immsel=2'b10; //denotes {{20{InstructCode[31]}},InstructCode[31],InstructCode[7],InstructCode[30:25],InstructCode[11:8]};

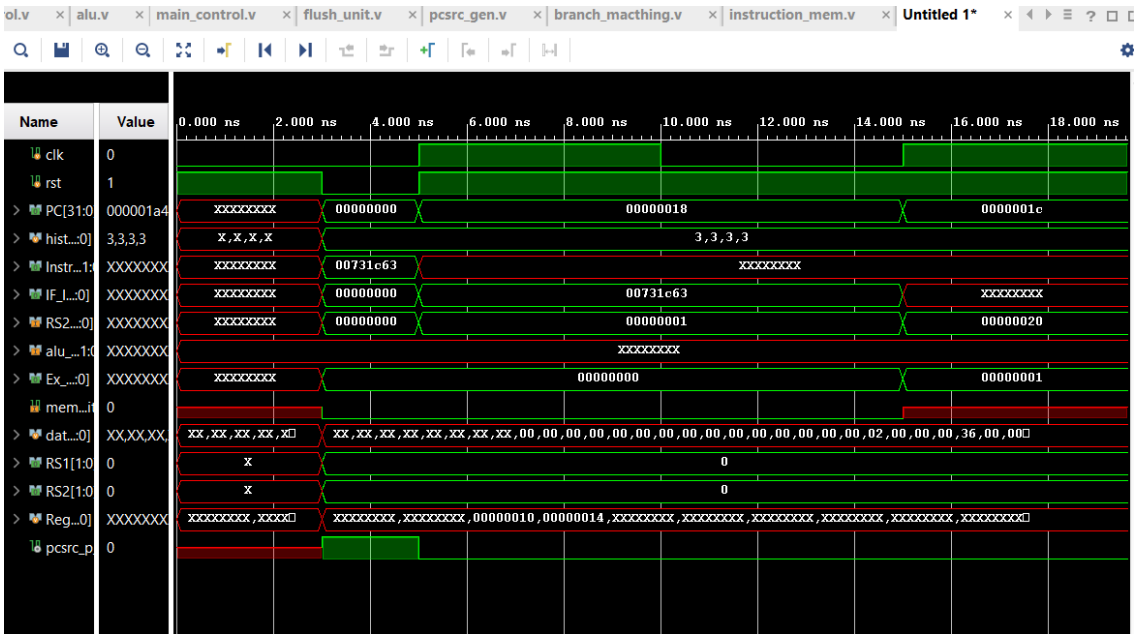
//beq

```
regwrite=0;
alusrc=0;            //select data2
memread=0;
memwrite=0;
memtoreg=1'bx;       //direct alu data to be written in reg file
case(fun3)           //pc=pc+x
3'b000: pcsrc= zero;   //beq
3'b001: pcsrc= ~zero;  //bneq
3'b100: pcsrc= lt;     //blt
3'b101: pcsrc= gt;     //bge
endcase
 aluop=2'b01;      //Btype
jal=1'bx;
```



5. Loop: bne t1,t2,subad (t1=#68,t2=#20)(Branch if not equal)

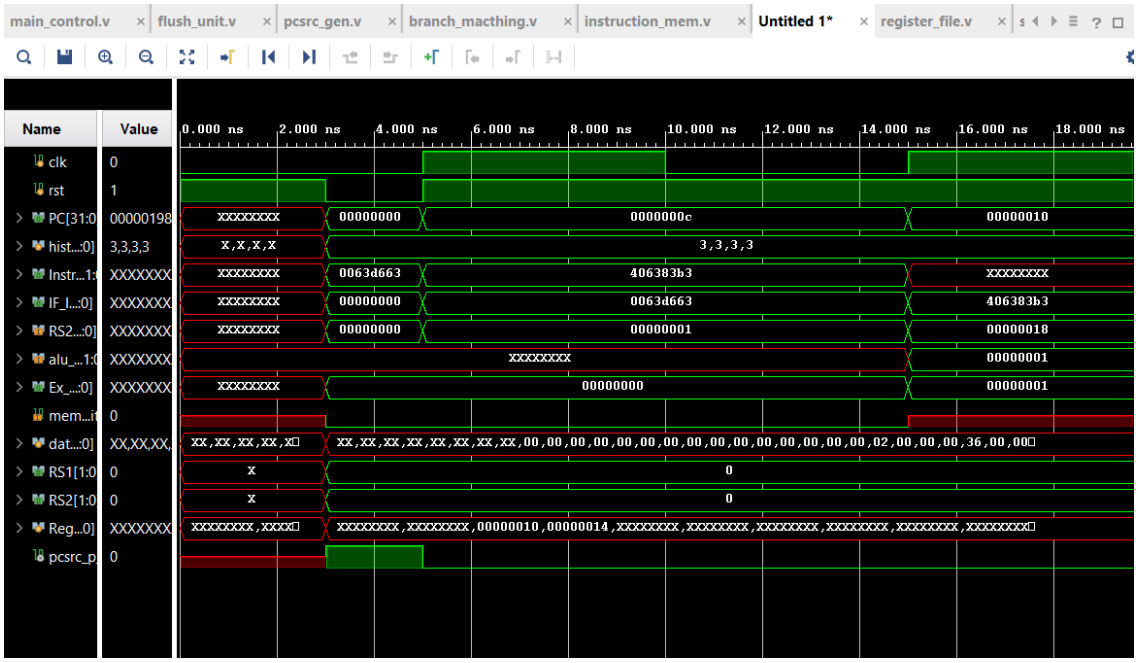
```
immsel=2'b10; //denotes
{{20{InstructCode[31]}},InstructCode[31],InstructCode[7],InstructCode[30:25],InstructCode[11:8]};
                regwrite=0;
                alusrc=0;           //select data2
                memread=0;
                memwrite=0;
                memtoreg=1'bx;      //direct alu data to be written in reg file
                case(fun3)          //pc=pc+x
                3'b000: pcsrc= zero;   //beq
                3'b001: pcsrc= ~zero;  //bneq
                3'b100: pcsrc= lt;     //blt
                3'b101: pcsrc= gt;     //bge
                        endcase
                aluop=2'b01;     //Btype
                        jal=1'bx;
```



6. Loop: bge t1,t2,subad(t1=#68,t2=#20)(Branch If greater than or equal)

immsel=2'b10; //denotes {{20{InstructCode[31]}},InstructCode[31],InstructCode[7],InstructCode[30:25],InstructCode[11:8]};

```
regwrite=0;
alusrc=0;            //select data2
memread=0;
memwrite=0;
memtoreg=1'bx;       //direct alu data to be written in reg file
case(fun3)           //pc=pc+x
3'b000: pcsrc= zero;   //beq
3'b001: pcsrc= ~zero;  //bneq
3'b100: pcsrc= lt;     //blt
3'b101: pcsrc= gt;     //bge
endcase
 aluop=2'b01;      //Btype
jal=1'bx;
```

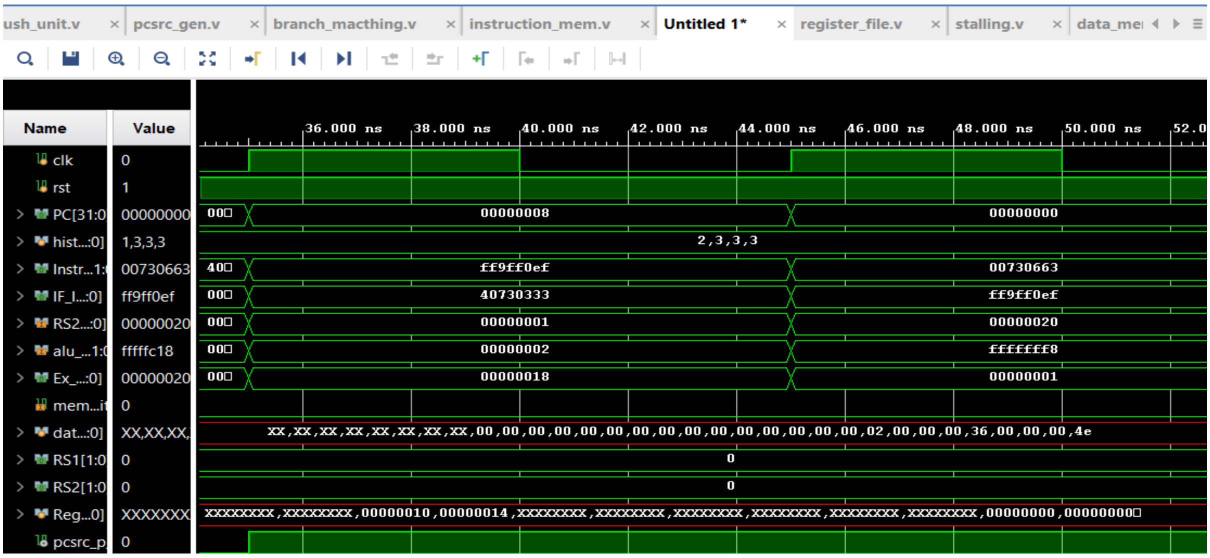


Jump and Link Instructions (Unconditional Jump)

7. jal subad

immsel=2'b11; //denotes {{12{InstructCode[31]}},InstructCode[31],InstructCode[19:12],InstructCode[20],InstructCode[30:21]}

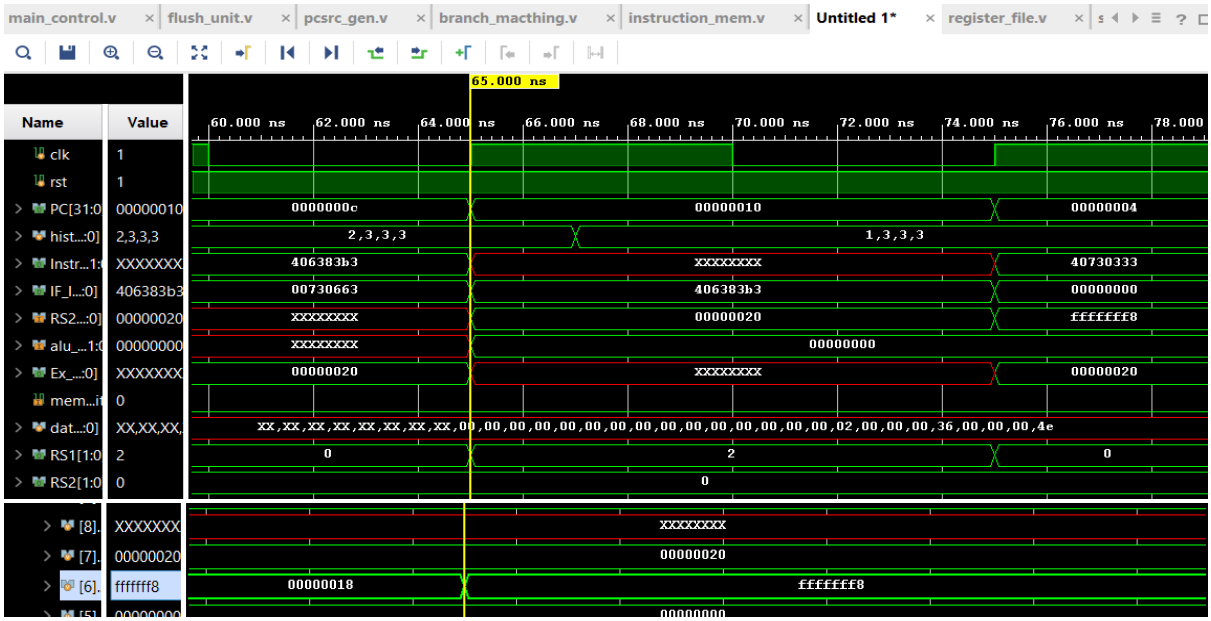
```
regwrite=1;       //writing to reg
alusrc=1'bx;        //don't care sign extended one
memread=1'bx;       //don't care memread
memwrite=1'bx;    //don't care memwrite
memtoreg=1'bx;    //don't care mem to a register
pcsrc=1;          //pc=pc+x

aluop=2'bxx;     //don't care
jump=1'b1;        //denotes jal instruction
end
```



R Type Instructions

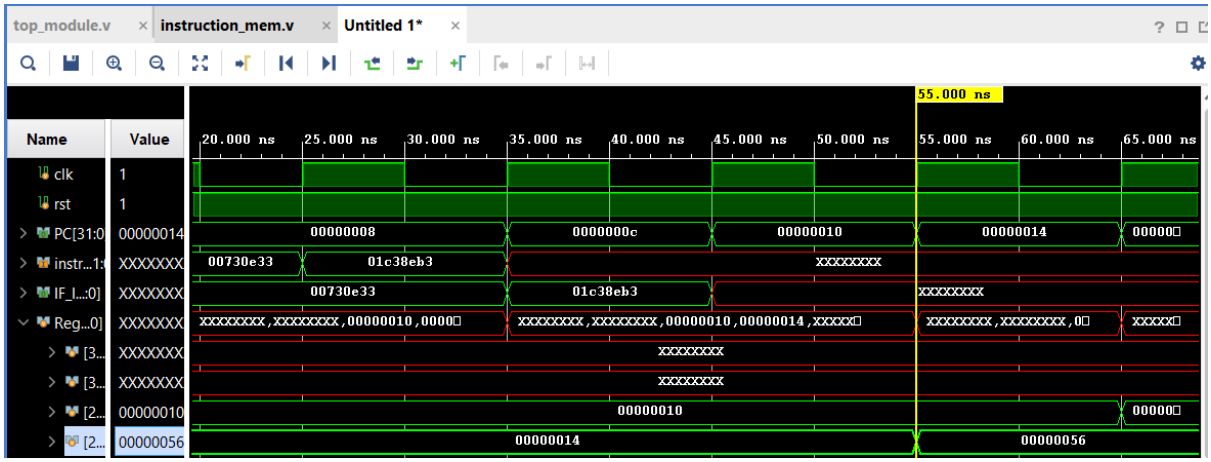
8. sub t1,t1,t2(t1=#28,t2=#20)

immsel=2'bxx;
regwrite=1;
alusrc=0; //select data2
memread=0;
memwrite=0;
memtoreg=0; //direct alu data to be written in reg file
pcsrc=0; //pc=pc+4
jal=1'b0;
aluop=2'b10; //rtype
and according to aluop=2'b10,
if my {instn_code[20],func3}=
4'b0000: AluControl=4'b0001; //add operation
4'b1000: AluControl=4'b0010; //subtraction operation
4'b0111: AluControl=4'b1011; //AND operation
4'b0110: AluControl=4'b1001; //OR operation
4'b0101: AluControl=4'b1101; //Shift right operation
4'b0001: AluControl=4'b1111; //Shift left operation
4'b0100: AluControl=4'b1110; //xor operation



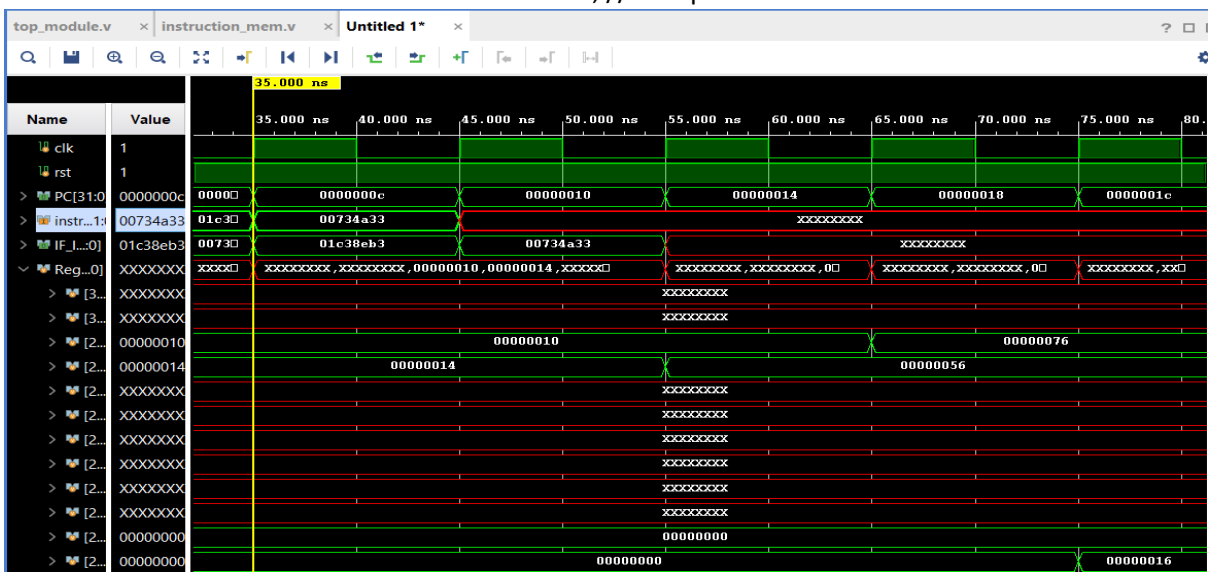
9. add s1,t1,t2 (t1=#68,t2=#20)

```
immsel=2'bxx;
  regwrite=1;
  alusrc=0;      //select data2
  memread=0;
  memwrite=0;
  memtoreg=0;    //direct alu data to be written in reg file
  pcsrc=0;       //pc=pc+4
  jal=1'b0;
 aluop=2'b10;    //rtype
                   and according to aluop=2'b10, if my {instn_code[20],func3}=
                     4'b0000: AluControl=4'b0001; //add operation
    4'b1000: AluControl=4'b0010; //subtraction operation
    4'b0111: AluControl=4'b1011; //AND operation
    4'b0110: AluControl=4'b1001; //OR operation
    4'b0101: AluControl=4'b1101; //Shift right operation
    4'b0001: AluControl=4'b1111; //Shift left operation
    4'b0100: AluControl=4'b1110; //xor operation
```



10. xor s4,t1,t2(t1=#68,t2=#20)(s4=#48)

```
immsel=2'bxx;
  regwrite=1;
  alusrc=0;      //select data2
  memread=0;
  memwrite=0;
  memtoreg=0;    //direct alu data to be written in reg file
  pcsrc=0;       //pc=pc+4
  jal=1'b0;
  aluop=2'b10;   //rtype
                    and according to aluop=2'b10, if my {instn_code[20],func3}=
                      4'b0000: AluControl=4'b0001; //add operation
    4'b1000: AluControl=4'b0010; //subtraction operation
    4'b0111: AluControl=4'b1011; //AND operation
    4'b0110: AluControl=4'b1001; //OR operation
    4'b0101: AluControl=4'b1101; //Shift right operation
    4'b0001: AluControl=4'b1111; //Shift left operation
    4'b0100: AluControl=4'b1000; //xor operation
```

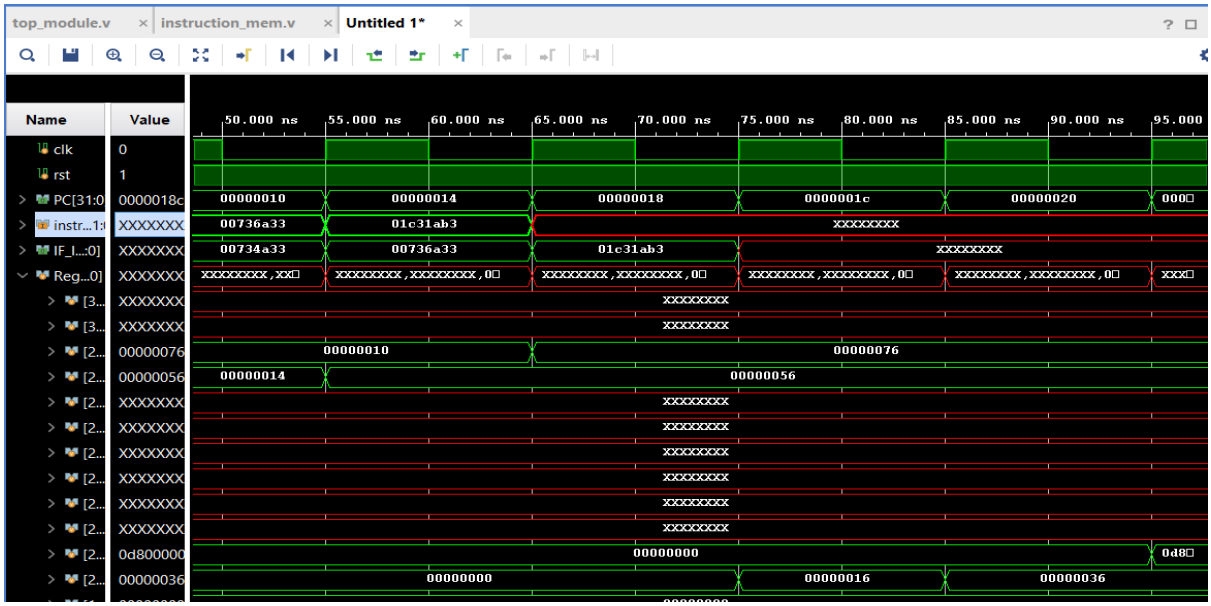


11. sll s5,t1,t3(t1=#15,t3=#02) (s5=#54)

```
immsel=2'bxx;
  regwrite=1;
  alusrc=0;      //select data2
  memread=0;
  memwrite=0;
  memtoreg=0;    //direct alu data to be written in reg file
  pcsrc=0;       //pc=pc+4
  jal=1'b0;
  aluop=2'b10;   //rtype
                    and according to aluop=2'b10, if my {instn_code[20],func3}=
                      4'b0000: AluControl=4'b0001; //add operation
    4'b1000: AluControl=4'b0010; //subtraction operation
    4'b0111: AluControl=4'b1011; //AND operation
    4'b0110: AluControl=4'b1001; //OR operation
    4'b0101: AluControl=4'b1101; //Shift right operation
```

4'b0001: AluControl=4'b0111; //Shift left operation

4'b0100: AluControl=4'b1110; //xor operation



12.or s4,t1,t2(t1=#15,t2=#10)(S4=#15)

immsel=2'bxx;

regwrite=1;

alusrc=0; //select data2

memread=0;

memwrite=0;

memtoreg=0; //direct alu data to be written in reg file

pcsrc=0; //pc=pc+4

jal=1'b0;

aluop=2'b10; //rtype

and according to aluop=2'b10, if my {instn_code[20],func3}=

4'b0000: AluControl=4'b0001; //add operation

4'b1000: AluControl=4'b0010; //subtraction operation

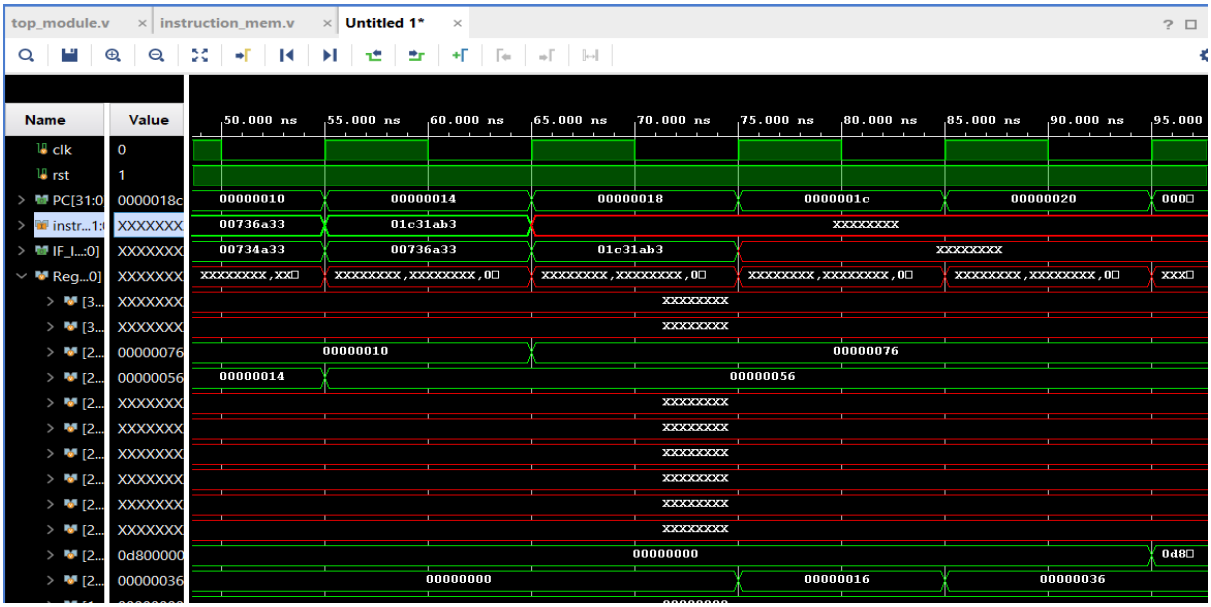
4'b0111: AluControl=4'b1011; //AND operation

4'b0110: AluControl=4'b0100; //OR operation

4'b0101: AluControl=4'b1101; //Shift right operation

4'b0001: AluControl=4'b1111; //Shift left operation

4'b0100: AluControl=4'b1110; //xor operation



❖ **Declaration:**

We hereby declared that we implemented this processor on by own.

During the implementing phase of the stalling, we were phasing problem in clock gating as in the same cycle clock being stop, but it should stop in next cycle, due to which stalling was not worked properly. How the control single need to be generate so stalling works properly on this issue we discussed with other groups. Moreover, we discussed on hardware implementation on the BHT and what are the control signal required and need to pass through pipeline register for detection of branch mismatch. These were topics on which we discussed with other group and solved our doubt.