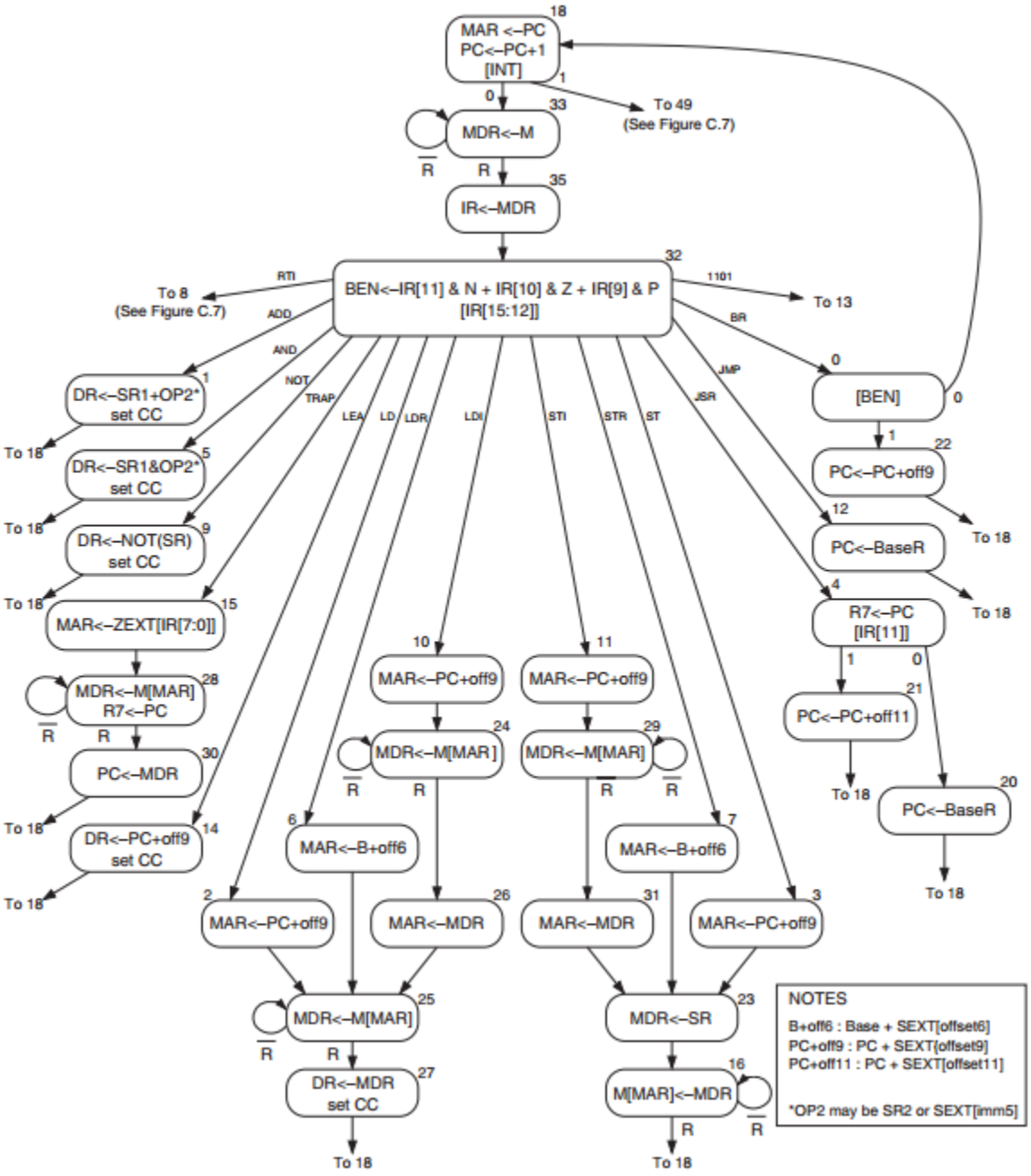
Lab 12

Isai Mercado Oliveros

December 6, 2014

State Diagram



Controller Verilog file (7pt)

```
module Control(output clk,output reset,output reg [1:0] aluControl,output reg enaALU,output reg [2:0] SR1,output reg [2:0] SR2,
               output reg [2:0] DR,output reg regWE,output reg [1:0] selPC,output reg enaMARM,output reg selMAR,output reg selEAB1,
               output reg [1:0] selEAB2,output reg enaPC,output reg ldPC,output reg ldIR,output reg ldMAR,output reg ldMDR,output reg selMDR,
               output reg memWE,output reg flagWE,output reg enaMDR,input N,input Z,input P,input[15:0] IR );

  wire JumpEnable;

  parameter Reset = 6'd0;
  parameter Fetch0 = 6'd1;
  parameter Fetch1 = 6'd2;
  parameter Fetch2 = 6'd3;
  parameter Decode = 6'd4;
  parameter Branch0 = 6'd5;
  parameter Branch1 = 6'd6;
  parameter Add = 6'd7;
  parameter Load0 = 6'd8;
  parameter Load1 = 6'd9;
  parameter Load2 = 6'd10;
  parameter Store0 = 6'd11;
  parameter Store1 = 6'd12;
  parameter Store2 = 6'd13;
  parameter JumpToSubroutine = 6'd14;
  parameter JumpToSubroutineOffset = 6'd15;
  parameter JumpToSubroutineBaseR = 6'd16;
  parameter And = 6'd17;
  parameter LoadRelative0 = 6'd18;
  parameter LoadRelative1 = 6'd19;
  parameter LoadRelative2 = 6'd20;
  parameter StoreRelative0 = 6'd21;
  parameter StoreRelative1 = 6'd22;
  parameter StoreRelative2 = 6'd23;
  parameter Not = 6'd24;
  parameter LoadIndirect0 = 6'd25;
  parameter LoadIndirect1 = 6'd26;
  parameter LoadIndirect2 = 6'd27;
  parameter LoadIndirect3 = 6'd28;
  parameter LoadIndirect4 = 6'd29;
  parameter StoreIndirect0 = 6'd30;
  parameter StoreIndirect1 = 6'd31;
  parameter StoreIndirect2 = 6'd32;
  parameter StoreIndirect3 = 6'd33;
  parameter StoreIndirect4 = 6'd34;
  parameter Jump = 6'd35;
  parameter LoadEffectiveAddress = 6'd36;


  reg[5:0] CurrentState;
  reg[5:0] NextState;
```

```
reg[5:0] CurrentState;
reg[5:0] NextState;

always @(posedge clk)
if (reset) CurrentState <= Fetch0;
else   CurrentState <= NextState;

assign JumpEnable = (N == 1'b1 || Z == 1'b1 || P == 1'b1)? 1'b1 : 1'b0;



always @(*)
begin

 aluControl = 2'b00;
 enaALU = 1'b0;
 SR1 = 3'b000;
 SR2 = 3'b000;
 DR = 3'b000;
 regWE = 1'b0;
 selPC = 2'b00;
 enaMARM = 1'b0;
 selMAR = 1'b0;
 selEAB1 = 1'b0;
 selEAB2 = 2'b00;
 enaPC = 1'b0;
 ldPC = 1'b0;
 ldIR = 1'b0;
 ldMAR = 1'b0;
 ldMDR = 1'b0;
 selMDR = 1'b0;
 memWE = 1'b0;
 flagWE = 1'b0;
 enaMDR = 1'b0;


      case (CurrentState)
          (Fetch0) : begin
                enaPC = 1'b1;
                ldMAR = 1'b1;
                NextState = Fetch1;
          end
          (Fetch1) : begin
                ldPC = 1'b1;
                ldMDR = 1'b1;
                selMDR = 1'b1;
                NextState = Fetch2;
```

```
end
(Fetch2) : begin
      ldIR = 1'b1;
      enaMDR = 1'b1;
      NextState = Decode;
end
(Decode) : begin
     if      (IR[15:12] == 4'b0000)  NextState = Branch0;
     else if (IR[15:12] == 4'b0001)  NextState = Add;
     else if (IR[15:12] == 4'b0010)  NextState = Load0;
     else if (IR[15:12] == 4'b0011)  NextState = Store0;
     else if (IR[15:12] == 4'b0100)  NextState = JumpToSubroutine;
     else if (IR[15:12] == 4'b0101)  NextState = And;
     else if (IR[15:12] == 4'b0110)  NextState = LoadRelative0;
     else if (IR[15:12] == 4'b0111)  NextState = StoreRelative0;
   //else if (IR[15:12] == 4'b1000)  NextState = ReturnFromInterrupt;
     else if (IR[15:12] == 4'b1001)  NextState = Not;
     else if (IR[15:12] == 4'b1010)  NextState = LoadIndirect0;
     else if (IR[15:12] == 4'b1011)  NextState = StoreIndirect0;
     else if (IR[15:12] == 4'b1100)  NextState = Jump;
   //else if (IR[15:12] == 4'b1101)  NextState = Reserved;
     else if (IR[15:12] == 4'b1110)  NextState = LoadEffectiveAddress;
   //else if (IR[15:12] == 4'b1111)  NextState = TRAP(System Call);

end
(Branch0) : begin
      selEAB2 = 2'b10;
     if (JumpEnable == 1'b0)  NextState = Fetch0;
     else                     NextState = Branch1;
end
(Branch1) : begin
      selPC = 2'b01;
      ldMAR = 1'b1;
      NextState = Fetch0;
end
 (Add) : begin
      aluControl = 2'b01;
      SR1 = IR[8:6];
      SR2 = IR[2:0];
      DR = IR[11:9];
      enaALU = 1'b1;
      regWE = 1'b1;
      flagWE = 1'b1;
      NextState = Fetch0;
end
 (Load0) : begin
      selEAB2 = 2'b10;
      enaMARM = 1'b1;
      ldMAR = 1'b1;
      NextState = Load1;
```

```
 (Load1) : begin
      ldMDR = 1'b1;
      selMDR = 1'b1;
      NextState = Load2;
end
 (Load2) : begin
      DR = IR[11:9];
      regWE = 1'b1;
      enaMDR = 1'b1;
      flagWE = 1'b1;
      NextState = Fetch0;
end
 (Store0) : begin
      selEAB2 = 2'b10;
      enaMARM = 1'b1;
      ldMAR = 1'b1;
      NextState = Store1;
end
 (Store1) : begin
      SR1 = IR[11:9];
      enaALU = 1'b1;
      ldMDR = 1'b1;
      NextState = Store2;
end
 (Store2) : begin
      memWE = 1'b1;
      NextState = Fetch0;
end
 (JumpToSubroutine) : begin
      DR = 3'b111;
      regWE = 1'b1;
      enaPC = 1'b1;
     if (IR[11] == 1'b0)  NextState = JumpToSubroutineBaseR;
     else                 NextState = JumpToSubroutineOffset;
end
 (JumpToSubroutineOffset) : begin
      selPC = 2'b01;
      selEAB2 = 2'b11;
      ldPC = 1'b1;
      NextState = Fetch0;
end
 (JumpToSubroutineBaseR) : begin
      aluControl = 2'b00;
      SR1 = IR[8:6];
      selPC = 2'b10;
      enaALU = 1'b1;
      ldPC = 1'b1;
      NextState = Fetch0;
end
```

```
(And) : begin
      aluControl = 2'b10;
      SR1 = IR[8:6];
      SR2 = IR[2:0];
      DR = IR[11:9];
      enaALU = 1'b1;
      regWE = 1'b1;
      flagWE = 1'b1;
      NextState = Fetch0;
end
(LoadRelative0) : begin
      SR1 = IR[8:6];
      selEAB1 = 1'b1;
      selEAB2 = 2'b01;
      enaMARM = 1'b1;
      ldMAR = 1'b1;
      NextState = LoadRelative1;
end
(LoadRelative1) : begin
      ldMDR = 1'b1;
      selMDR = 1'b1;
      NextState = LoadRelative2;
end
(LoadRelative2) : begin
      DR = IR[11:9];
      regWE = 1'b1;
      enaMDR = 1'b1;
      NextState = Fetch0;
end
(StoreRelative0) : begin
      SR1 = IR[8:6];
      selEAB1 = 1'b1;
      selEAB2 = 2'b01;
      enaMARM = 1'b1;
      ldMAR = 1'b1;
      NextState = StoreRelative1;
end
(StoreRelative1) : begin
      aluControl = 2'b00;
      SR1 = IR[11:9];
      enaALU = 1'b1;
      ldMDR = 1'b1;
      NextState = StoreRelative2;
end
(StoreRelative2) : begin
      memWE = 1'b1;
      NextState = Fetch0;
end
```

```
(Not) : begin
      aluControl = 2'b11;
      SR1 = IR[8:6];
      DR = IR[11:9];
      enaALU = 1'b1;
      regWE = 1'b1;
      flagWE = 1'b1;
      NextState = Fetch0;
end
(LoadIndirect0) : begin
      selEAB2 = 2'b10;
      enaMARM = 1'b1;
      ldMAR = 1'b1;
      NextState = LoadIndirect1;
end
(LoadIndirect1) : begin
      ldMDR = 1'b1;
      selMDR = 1'b1;
      NextState = LoadIndirect2;
end
(LoadIndirect2) : begin
      ldMAR = 1'b1;
      enaMDR = 1'b1;
      NextState = LoadIndirect3;
end
(LoadIndirect3) : begin
      ldMDR = 1'b1;
      selMDR = 1'b1;
      NextState = LoadIndirect4;
end
(LoadIndirect4) : begin
      DR = IR[11:9];
      regWE = 1'b1;
      enaMDR = 1'b1;
      NextState = Fetch0;
end
(StoreIndirect0) : begin
      selEAB2 = 2'b10;
      enaMARM = 1'b1;
      ldMAR = 1'b1;
      NextState = StoreIndirect1;
end
(StoreIndirect1) : begin
      ldMDR = 1'b1;
      selMDR = 1'b1;
      NextState = StoreIndirect2;
end
(StoreIndirect2) : begin
      ldMAR = 1'b1;
      enaMDR = 1'b1;
      NextState = StoreIndirect3;
end
```

```
            (StoreIndirect3) : begin
                  aluControl = 2'b00;
                  SR1 = IR[11:9];
                  enaALU = 1'b1;
                  ldMDR = 1'b1;
                  NextState = StoreIndirect4;
            end
            (StoreIndirect4) : begin
                  memWE = 1'b1;
                  NextState = Fetch0;
            end
            (Jump) : begin
                  aluControl = 1'b1;
                  SR1 = IR[8:6];
                  selPC = 2'b10;
                  enaALU = 1'b1;
                  ldPC = 1'b1;
                  NextState = Fetch0;
            end
            (LoadEffectiveAddress) : begin
                  DR = IR[11:9];
                  selEAB2 = 2'b10;
                  regWE = 1'b1;
                  enaMARM = 1'b1;
                  NextState = Fetch0;
            end
      endcase
   end
endmodule
```

Full LC3 Verilog file (7pt)

```
////////////////////////////////////////////////////////////////////////////////
module CPU(input clk,input reset,input[1:0] aluControl,input enaALU,input[2:0] SR1,input[2:0] SR2,
input[2:0] DR,input regWE,input[1:0] selPC,input enaMARM,input selMAR,input selEAB1,
input[1:0] selEAB2,input enaPC,input ldPC,input ldIR,input ldMAR,input ldMDR,input selMDR,
input memWE,input flagWE,input enaMDR,output N,output Z,output P,output[15:0] IR);

/*
dataPath( input clk,input reset,input[1:0] aluControl,input enaALU,input[2:0] SR1,input[2:0] SR2,
input[2:0] DR,input regWE,input[1:0] selPC,input enaMARM,input selMAR,input selEAB1,
input[1:0] selEAB2,input enaPC,input ldPC,input ldIR,input ldMAR,input ldMDR,input selMDR,
input memWE,input flagWE,input enaMDR,output N,output Z,output P,output[15:0] IR);
*/
dataPath path (clk,  reset,  aluControl,  enaALU,  SR1,  SR2,
  DR,  regWE,  selPC,  enaMARM,  selMAR,  selEAB1,
  selEAB2,  enaPC,  ldPC,  ldIR,  ldMAR,  ldMDR,  selMDR,
  memWE,  flagWE,  enaMDR,  N,  Z,  P,  IR);

Control control (clk,  reset,  aluControl,  enaALU,  SR1,  SR2,
  DR,  regWE,  selPC,  enaMARM,  selMAR,  selEAB1,
  selEAB2,  enaPC,  ldPC,  ldIR,  ldMAR,  ldMDR,  selMDR,
  memWE,  flagWE,  enaMDR,  N,  Z,  P,  IR);


endmodule
```

Procedure (20pts possible)

Master TCL file (2pt)

```
wave add clk
wave add reset

#add signals to view the PC and PC control
wave add selPC
wave add enaPC
wave add ldPC

#add signals to view the IR and IR control
wave add ldIR
wave add IR -radix hex

#add signals to view the EAB control
wave add selEAB1
wave add selEAB2

#add signals to view the MARMux control
wave add selMAR
wave add enaMARM

#add signals to view Register File control
wave add DR
wave add SR1
wave add SR2
wave add regWE
```

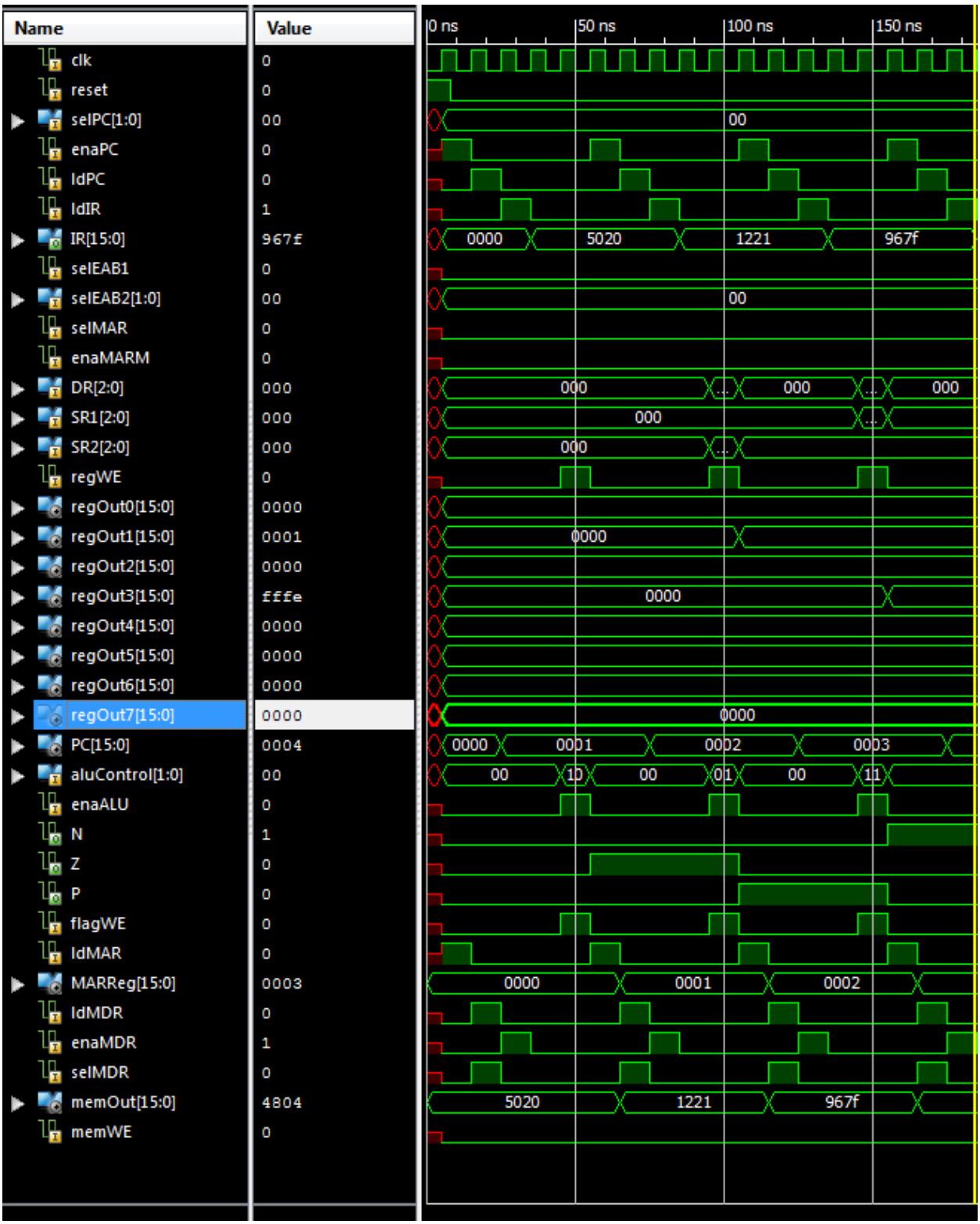
```
#add signals to view the Registers in the Register File
wave add /path/regfile/regOut0 -radix hex
wave add /path/regfile/regOut1 -radix hex
wave add /path/regfile/regOut2 -radix hex
wave add /path/regfile/regOut3 -radix hex
wave add /path/regfile/regOut4 -radix hex
wave add /path/regfile/regOut5 -radix hex
wave add /path/regfile/regOut6 -radix hex
wave add /path/regfile/regOut7 -radix hex
wave add /path/PC -radix hex

#add signals to view the ALU control
wave add aluControl
wave add enaALU

#view the condition flags
wave add N
wave add Z
wave add P
wave add flagWE

#add signals to view the Memory Registers and the Memory control
wave add ldMAR
wave add /path/memory/MARReg -radix hex
wave add ldMDR
wave add enaMDR
wave add selMDR
wave add /path/memory/memOut -radix hex
wave add memWE
```

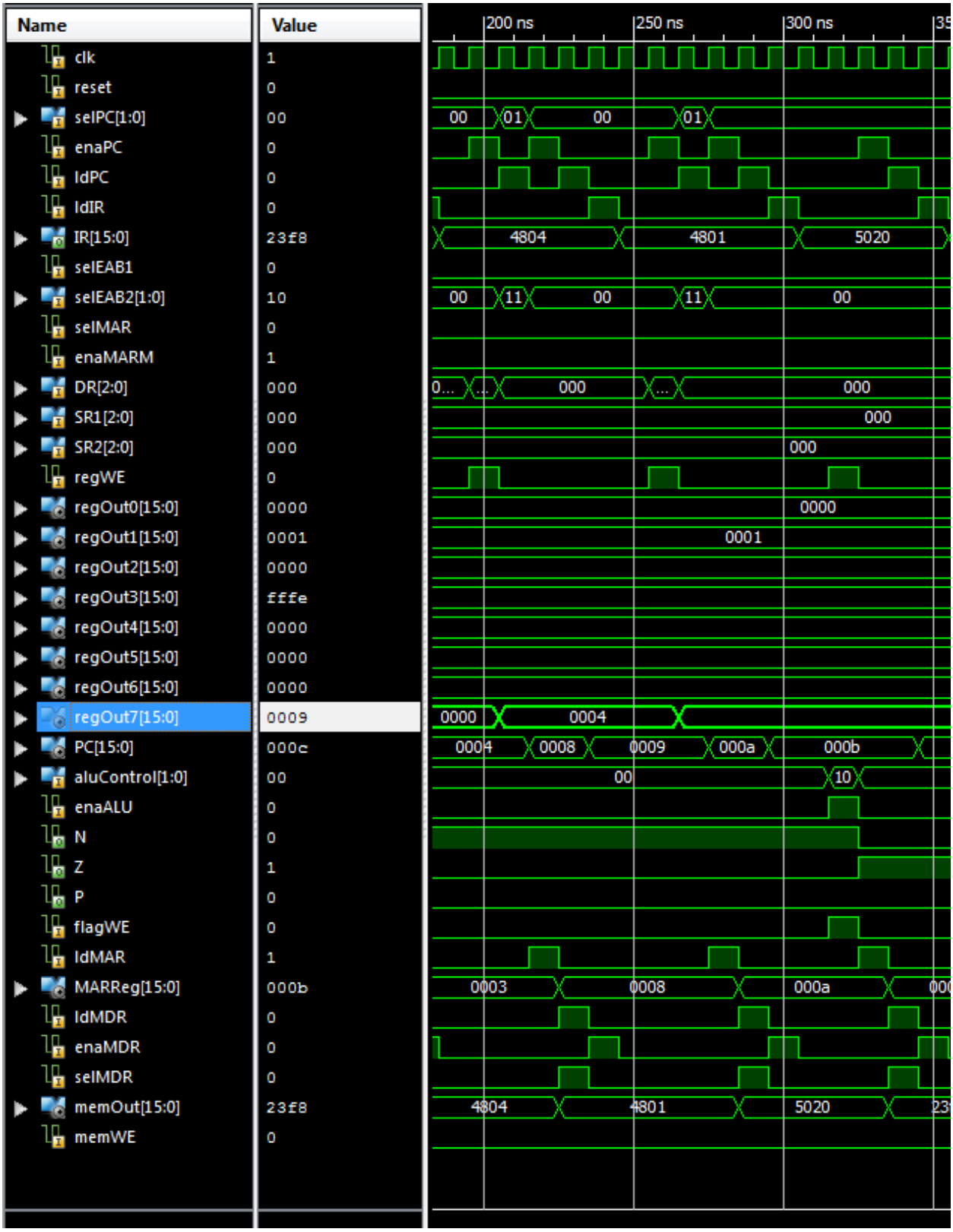
isim force add clk 0 -time 0 -value 1 -time 5ns -repeat 10ns

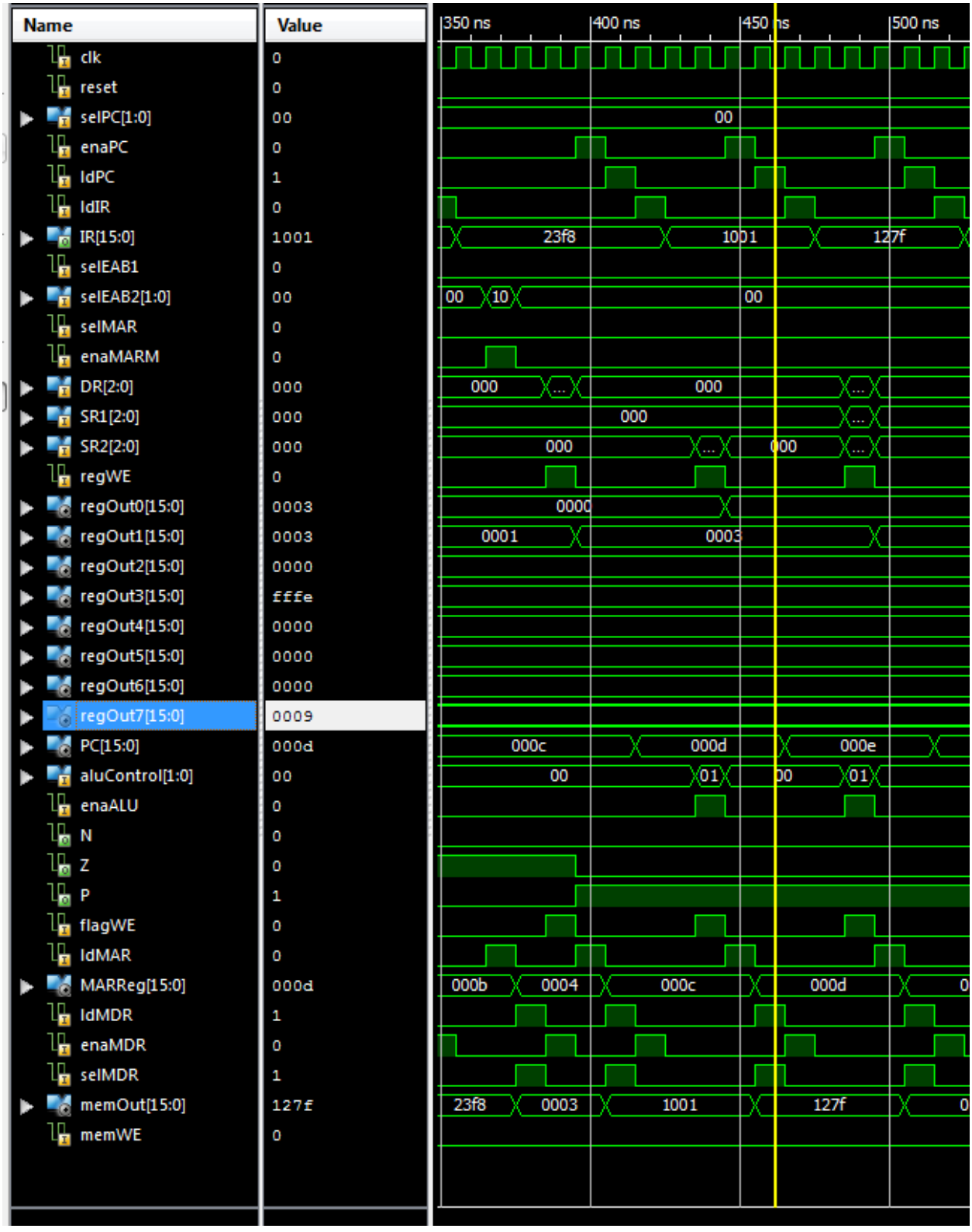
isim force add reset 1 -time 0 -value 0 -time 8ns

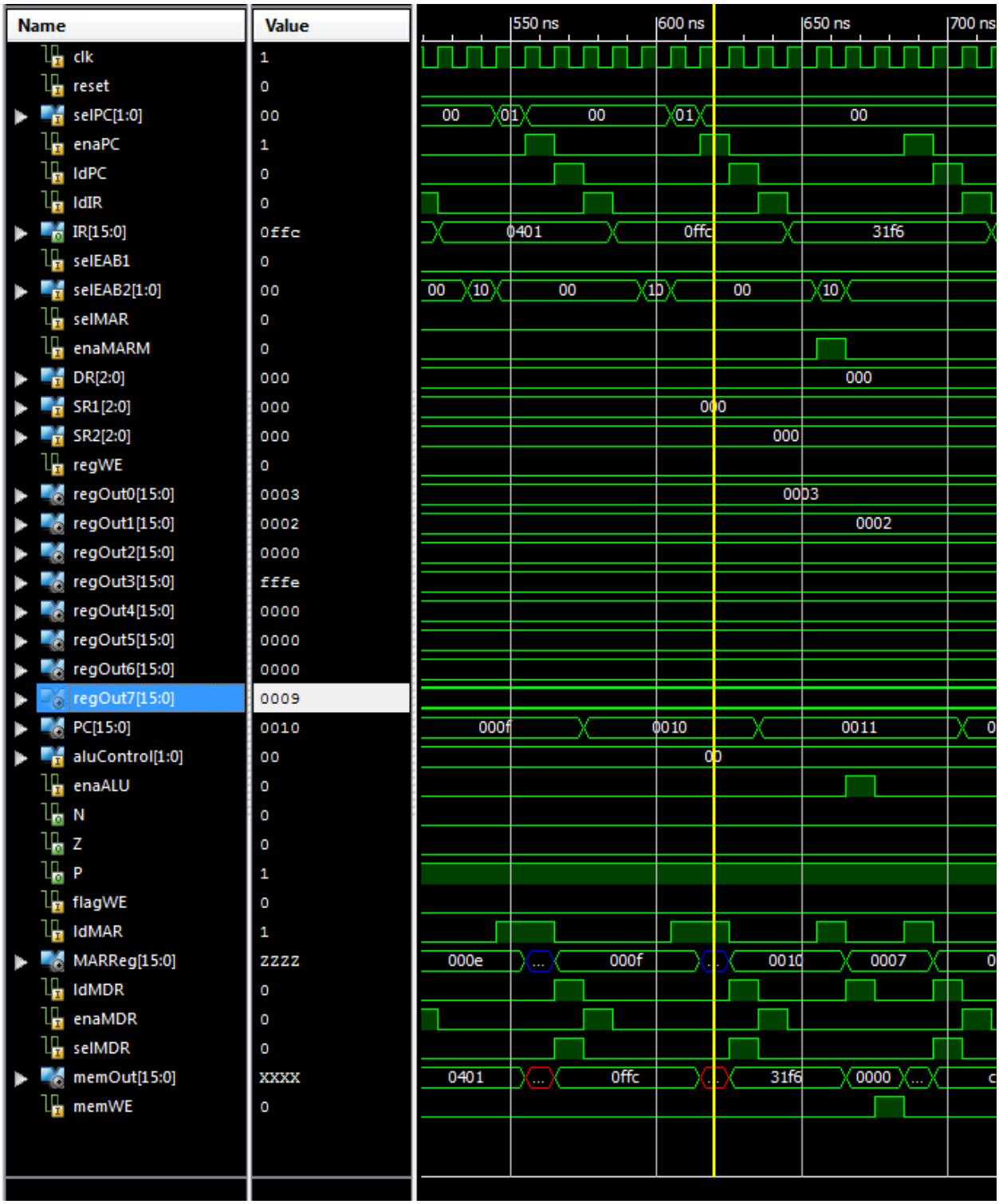
run 4us

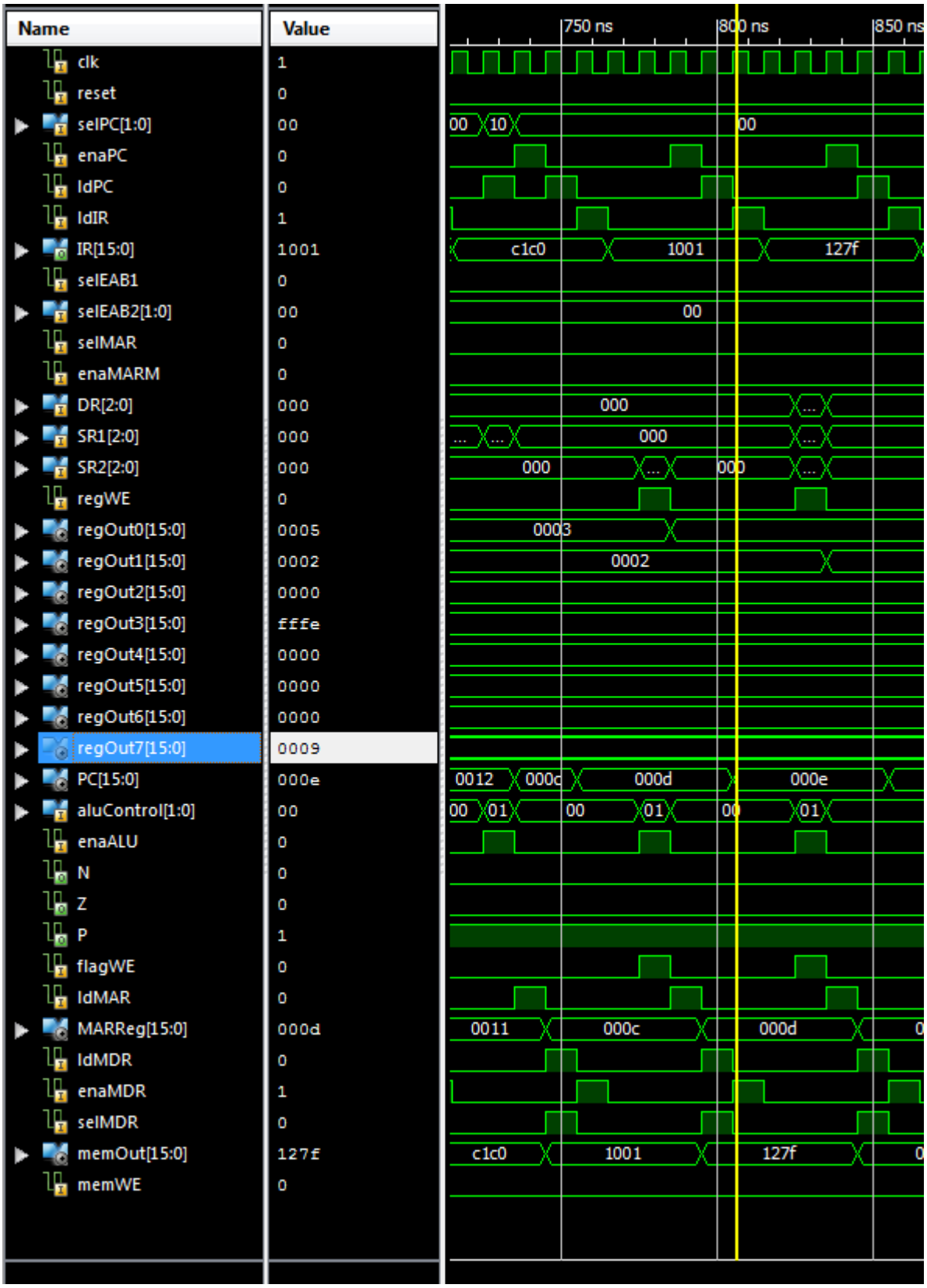
All Simulation waveforms(18pt)

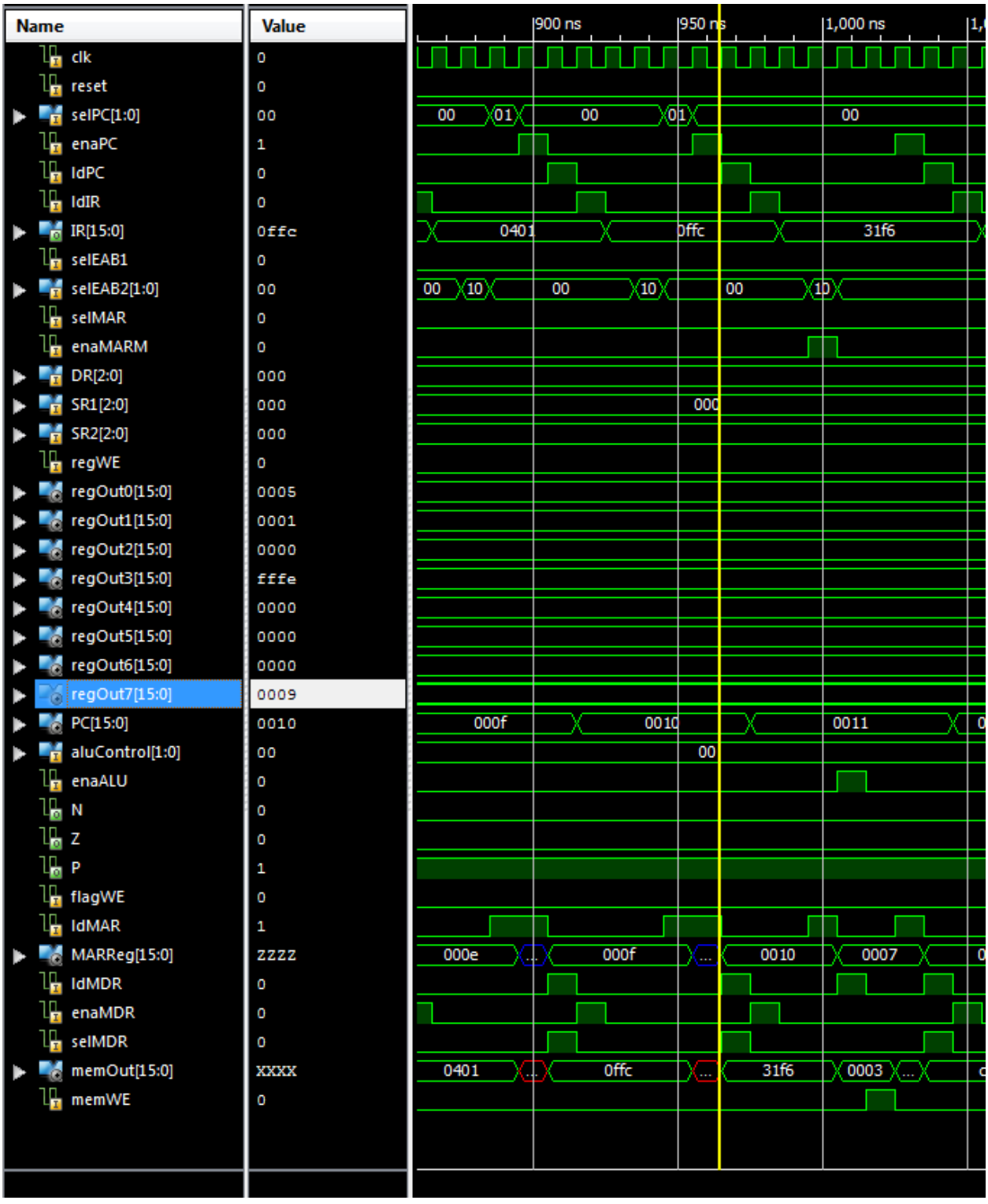

Name
Value
0 ns
50 ns
100 ns
150 ns
clk 0
reset 0
selPC[1:0] 00
enaPC 0
ldPC 0
ldIR 1
IR[15:0] 967f
selEAB1 0
selEAB2[1:0] 00
selMAR 0
enaMARM 0
DR[2:0] 000
SR1[2:0] 000
SR2[2:0] 000
regWE 0
regOut0[15:0] 0000
regOut1[15:0] 0001
regOut2[15:0] 0000
regOut3[15:0] fffe
regOut4[15:0] 0000
regOut5[15:0] 0000
regOut6[15:0] 0000
regOut7[15:0] 0000
PC[15:0] 0004
aluControl[1:0] 00
enaALU 0
N 1
Z 0
P 0
flagWE 0
ldMAR 0
MARReg[15:0] 0003
ldMDR 0
enaMDR 1
selMDR 0
memOut[15:0] 4804
memWE 0
0000
5020
1221
967f
0001
0002
0003
00
10
01
11
ffff
4804

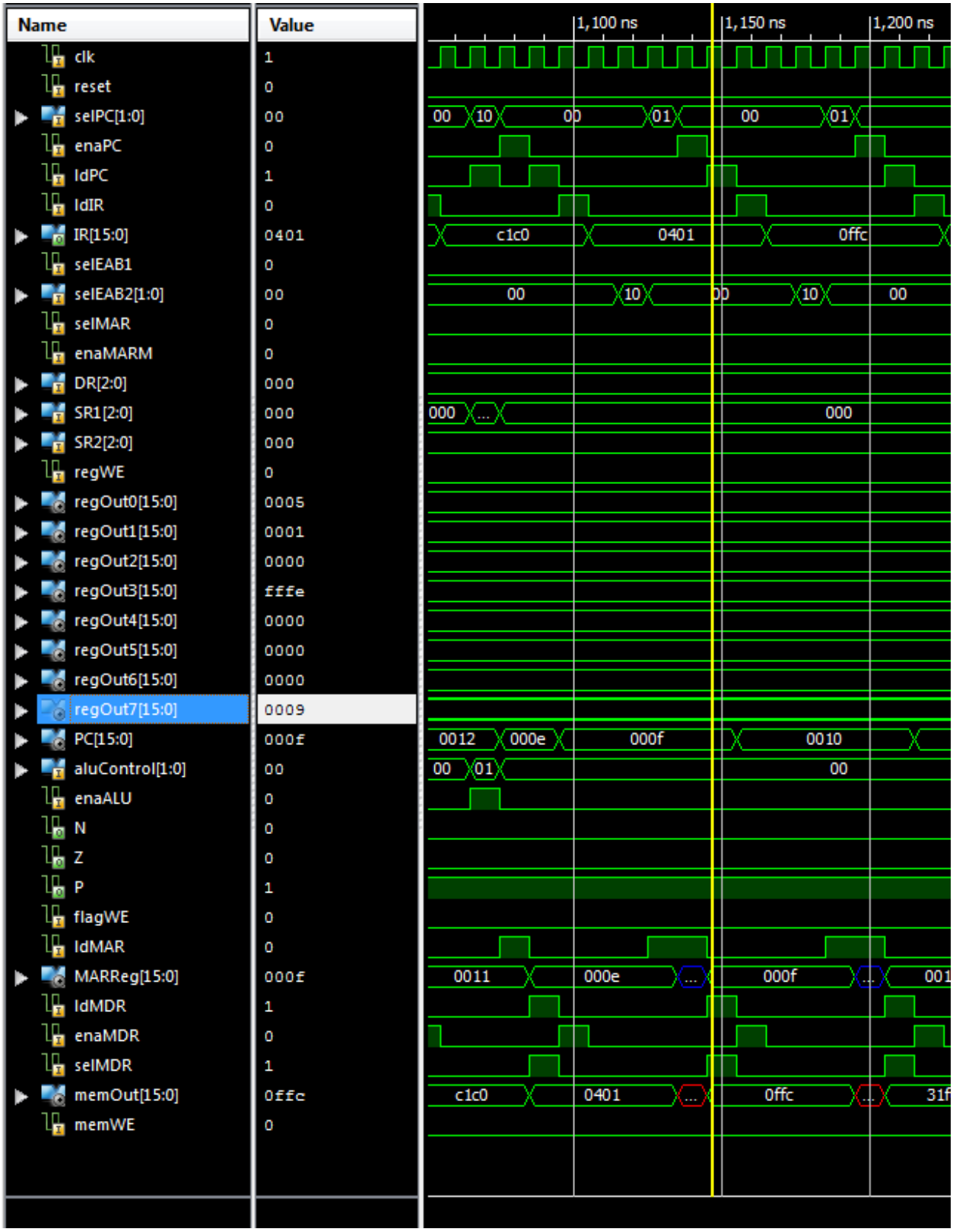

Name
Value
200 ns
250 ns
300 ns
clk 1
reset 0
selPC[1:0] 00
enaPC 0
ldPC 0
ldIR 0
IR[15:0] 23f8
selEAB1 0
selEAB2[1:0] 10
selMAR 0
enaMARM 1
DR[2:0] 000
SR1[2:0] 000
SR2[2:0] 000
regWE 0
regOut0[15:0] 0000
regOut1[15:0] 0001
regOut2[15:0] 0000
regOut3[15:0] fffe
regOut4[15:0] 0000
regOut5[15:0] 0000
regOut6[15:0] 0000
regOut7[15:0] 0009
PC[15:0] 000c
aluControl[1:0] 00
enaALU 0
N 0
Z 1
P 0
flagWE 0
ldMAR 1
MARReg[15:0] 000b
ldMDR 0
enaMDR 0
selMDR 0
memOut[15:0] 23f8
memWE 0
00 01 00 01
4804 4801 5020
00 11 00 11 00
000 000 000
000
000
0000
0001
0000 0004
0004 0008 0009 000a 000b
00 10
0003 0008 000a
4804 4801 5020

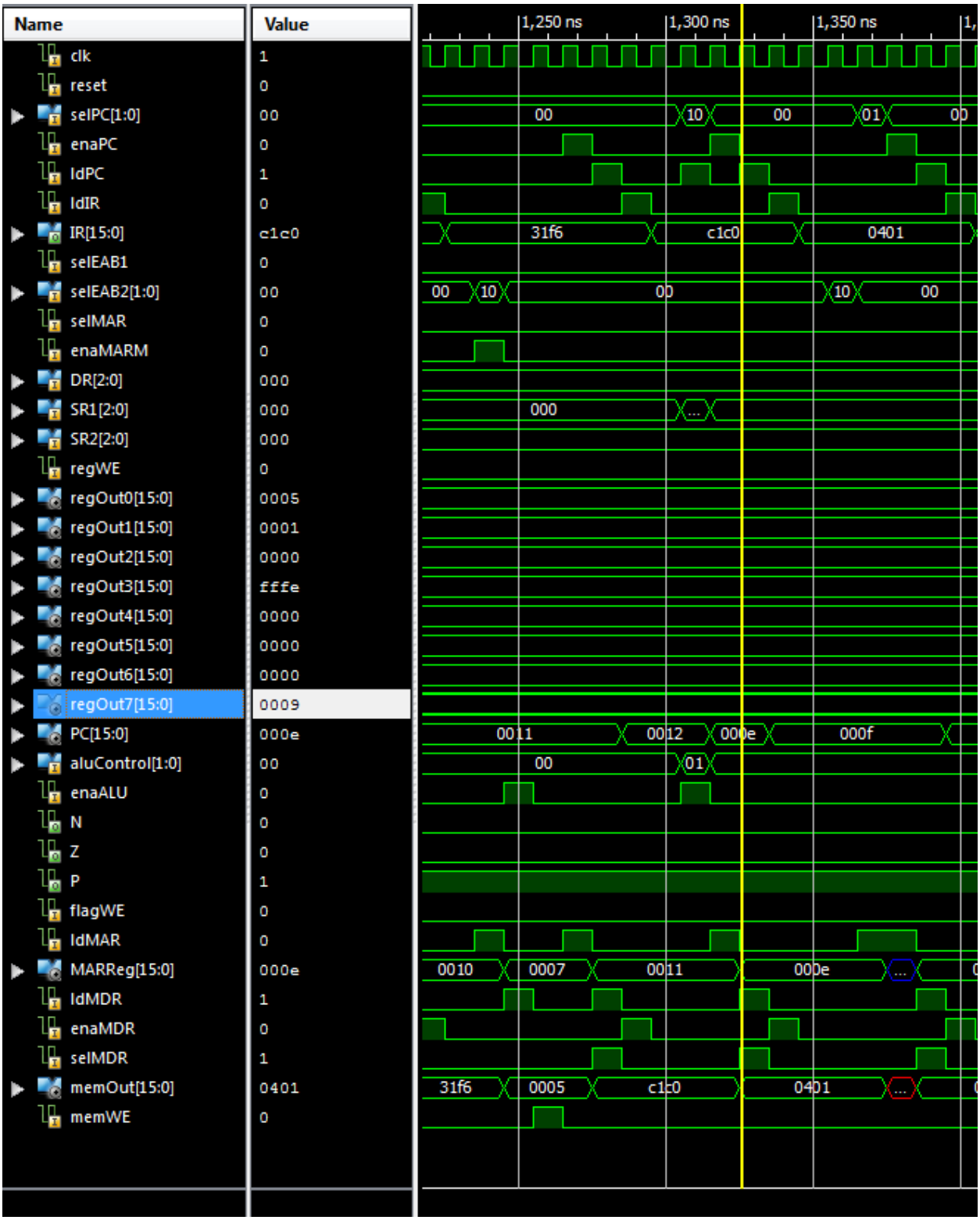

Name
Value
clk 0
reset 0
selPC[1:0] 00
enaPC 0
ldPC 1
ldIR 0
IR[15:0] 1001
selEAB1 0
selEAB2[1:0] 00
selMAR 0
enaMARM 0
DR[2:0] 000
SR1[2:0] 000
SR2[2:0] 000
regWE 0
regOut0[15:0] 0003
regOut1[15:0] 0003
regOut2[15:0] 0000
regOut3[15:0] fffe
regOut4[15:0] 0000
regOut5[15:0] 0000
regOut6[15:0] 0000
regOut7[15:0] 0009
PC[15:0] 000d
aluControl[1:0] 00
enaALU 0
N 0
Z 0
P 1
flagWE 0
ldMAR 0
MARReg[15:0] 000d
ldMDR 1
enaMDR 0
selMDR 1
memOut[15:0] 127f
memWE 0
350 ns
400 ns
450 ns
500 ns


Name
Value
550 ns
600 ns
650 ns
700 ns
clk
1
reset
0
selPC[1:0]
00
enaPC
1
ldPC
0
ldIR
0
IR[15:0]
0ffc
selEAB1
0
selEAB2[1:0]
00
selMAR
0
enaMARM
0
DR[2:0]
000
SR1[2:0]
000
SR2[2:0]
000
regWE
0
regOut0[15:0]
0003
regOut1[15:0]
0002
regOut2[15:0]
0000
regOut3[15:0]
fffe
regOut4[15:0]
0000
regOut5[15:0]
0000
regOut6[15:0]
0000
regOut7[15:0]
0009
PC[15:0]
0010
aluControl[1:0]
00
enaALU
0
N
0
Z
0
P
1
flagWE
0
ldMAR
1
MARReg[15:0]
zzzz
ldMDR
0
enaMDR
0
selMDR
0
memOut[15:0]
xxxx
memWE
0
0401
0ffc
31f6
000f
0010
0011
000e
0007
0000


Name
Value
750 ns
800 ns
850 ns
clk
1
reset
0
selPC[1:0]
00
enaPC
0
ldPC
0
ldIR
1
IR[15:0]
1001
selEAB1
0
selEAB2[1:0]
00
selMAR
0
enaMARM
0
DR[2:0]
000
SR1[2:0]
000
SR2[2:0]
000
regWE
0
regOut0[15:0]
0005
regOut1[15:0]
0002
regOut2[15:0]
0000
regOut3[15:0]
fffe
regOut4[15:0]
0000
regOut5[15:0]
0000
regOut6[15:0]
0000
regOut7[15:0]
0009
PC[15:0]
000e
aluControl[1:0]
00
enaALU
0
N
0
Z
0
P
1
flagWE
0
ldMAR
0
MARReg[15:0]
000d
ldMDR
0
enaMDR
1
selMDR
0
memOut[15:0]
127f
memWE
0
00
10
c1c0
1001
127f
0003
0002
0012
000c
000d
000e
01
0011
000c
000d
c1c0
1001
127f


Name
Value
clk
0
reset
0
selPC[1:0]
00
enaPC
1
ldPC
0
ldIR
0
IR[15:0]
0ffc
selEAB1
0
selEAB2[1:0]
00
selMAR
0
enaMARM
0
DR[2:0]
000
SR1[2:0]
000
SR2[2:0]
000
regWE
0
regOut0[15:0]
0005
regOut1[15:0]
0001
regOut2[15:0]
0000
regOut3[15:0]
fffe
regOut4[15:0]
0000
regOut5[15:0]
0000
regOut6[15:0]
0000
regOut7[15:0]
0009
PC[15:0]
0010
aluControl[1:0]
00
enaALU
0
N
0
Z
0
P
1
flagWE
0
ldMAR
1
MARReg[15:0]
zzzz
ldMDR
0
enaMDR
0
selMDR
0
memOut[15:0]
xxxx
memWE
0
900 ns
950 ns
1,000 ns
0401
31f6
000f
0010
0011
000e
0007
0003


Name
Value
clk
1
reset
0
selPC[1:0]
00
enaPC
0
ldPC
1
ldIR
0
IR[15:0]
0401
selEAB1
0
selEAB2[1:0]
00
selMAR
0
enaMARM
0
DR[2:0]
000
SR1[2:0]
000
SR2[2:0]
000
regWE
0
regOut0[15:0]
0005
regOut1[15:0]
0001
regOut2[15:0]
0000
regOut3[15:0]
fffe
regOut4[15:0]
0000
regOut5[15:0]
0000
regOut6[15:0]
0000
regOut7[15:0]
0009
PC[15:0]
000f
aluControl[1:0]
00
enaALU
0
N
0
Z
0
P
1
flagWE
0
ldMAR
0
MARReg[15:0]
000f
ldMDR
1
enaMDR
0
selMDR
1
memOut[15:0]
0ffc
memWE
0
1,100 ns
1,150 ns
1,200 ns


Name
Value
clk 1
reset 0
selPC[1:0] 00
enaPC 0
ldPC 1
ldIR 0
IR[15:0] c1c0
selEAB1 0
selEAB2[1:0] 00
selMAR 0
enaMARM 0
DR[2:0] 000
SR1[2:0] 000
SR2[2:0] 000
regWE 0
regOut0[15:0] 0005
regOut1[15:0] 0001
regOut2[15:0] 0000
regOut3[15:0] fffe
regOut4[15:0] 0000
regOut5[15:0] 0000
regOut6[15:0] 0000
regOut7[15:0] 0009
PC[15:0] 000e
aluControl[1:0] 00
enaALU 0
N 0
Z 0
P 1
flagWE 0
ldMAR 0
MARReg[15:0] 000e
ldMDR 1
enaMDR 0
selMDR 1
memOut[15:0] 0401
memWE 0
1,250 ns
1,300 ns
1,350 ns
00 10 00 01 00
31f6 c1c0 0401
00 10 00 10 00
000
0011 0012 000e 000f
00 01
0010 0007 0011 000e
31f6 0005 c1c0 0401

Anomalies

Fun lab but some TA do not even know the syntax of If statements and reg