# **Lab Report #3: Simple Processor**

Linh Ngo

Brian Dawson

ELC 363 - Computer Engineering Laboratory I

The College of New Jersey

Department of Electrical and Computer Engineering (ECE)



## **Introduction including Problem Description:**

In this lab the students were required to use the Xilinx design package for FPGAs to construct and implement a simple processor. The students needed to study the architecture of the processor in order to process a set a given instruction word formats with 16-bit word-length. The students used an architecture diagram and a controller flow diagram in order to create the processor. The students also used the Von Neuman approach to design the processor as well as implementing machine language code to test the processor.

## **Results:**

a. RTL for each instruction and addressing mode combination

### NOT

- 1.  $S0 : IR \leq M(PC)$
- 2.  $S1 : PC \le PC + 1$
- 3. S2 :  $AC \leq AC$

### INCA

- 1.  $S0 : IR \leq M(PC)$
- 2.  $S1 : PC \le PC + 1$
- 3.  $S3 : AC \le AC + 1$

## JPA (If $AC \leq 0$ )

- 1.  $S0 : IR \leq M(PC)$
- 2.  $S1 : PC \le PC + 1$

JPA (If AC > 0 AND AM = 0) (DIRECT)

- 1.  $S0 : IR \leq M(PC)$
- 2.  $S1 : PC \le PC + 1$
- 3. S7 : PC <= IR (Address Portion)

JPA (If AC > 0 AND AM = 1) (INDIRECT)

- 1.  $S0 : IR \leq M(PC)$
- 2.  $S1 : PC \le PC + 1$
- 3. S4 : MA <= IR (Address Portion)
- 4. S5 : MD <= M[MA] (Data in memory address)
- 5.  $S6 : PC \leq MD$

## STA (AM = 0) (DIRECT)

- 1.  $S0: IR \leq M(PC)$
- 2.  $S1 : PC \le PC + 1$
- 3. S8: MA <= IR (Address Portion)
- 4. S11:  $M[MA] \le AC$  and  $AC \le 0$

### STA (AM = 1) (INDIRECT)

- 1.  $S0: IR \leq M(PC)$
- 2.  $S1 : PC \le PC + 1$
- 3. S8: MA <= IR (Address Portion)
- 4. S9 :  $MD \le M[MA]$  (Data in memory address)
- 5.  $S10 : MA \le MD$
- 6. S11:  $M[MA] \le AC$  and  $AC \le 0$

### LDA (AM = 0) (Direct)

- 1.  $S0: IR \leq M(PC)$
- 2.  $S1 : PC \le PC + 1$
- 3. S8 : MA <= IR (Address Portion)
- 4.  $S12 : MD \le M[MA]$
- 5.  $S16 : AC \le MD$

### LDA (AM = 1) (Indirect)

- 1.  $S0: IR \leq M(PC)$
- 2.  $S1 : PC \le PC + 1$
- 3. S8 : MA <= IR (Address Portion)
- 4.  $S12 : MD \le M[MA]$
- 5.  $S13 : MA \le M[MD]$
- 6.  $S14 : MD \le M[MA]$
- 7.  $S16 : AC \le MD$

### ADC (AM = 0) (Direct)

- 1.  $S0: IR \leq M(PC)$
- 2.  $S1 : PC \le PC + 1$

3. S8: MA <= IR (Address Portion)

4.  $S12 : MD \le M[MA]$ 

5.  $S15 : AC \le AC + C + MD$ 

## ADC(AM = 1) (Indirect)

1.  $S0: IR \leq M(PC)$ 

2.  $S1 : PC \le PC + 1$ 

3. S8: MA <= IR (Address Portion)

4.  $S12 : MD \le M[MA]$ 

5.  $S13 : MA \le M[MD]$ 

6.  $S14 : MD \le M[MA]$ 

7.  $S15 : AC \le AC + C + MD$ 

## b. Number of clock cycles for each instruction and addressing mode combination

Instruction with addressing mode	Clock cycles
NOT	3
ADC Indirect	7
ADC Direct	5
JPA Direct	3
JPA Indirect	5
JPA if ACC <= 0	2
INCA	3
STA Direct	4
STA Indirect	6
LDA Direct	5
LDA Indirect	7

### c. Verilog design code

```
`timescale 1ns / 1ps
module alu(
   input wire [15:0]a,b,
   input wire [2:0]control,
   input wire ci,
   output reg [15:0]y,
   output reg co
   );
   always @(*) begin
       case(control)
           3'b000: y =~a; //NOTa = Not the Accumulator(Invert it)
           3'b001: {co,y} <= a+b+ci;//Add to input with carry
           3'b010: y = (a + 1); //Increment the Accumulator
           3'b011: y = 0; //Zeroing the Accumulator (Clearing accumulator)
           3'b100: y = b; //Passing B (Used in loading accumulator)
       endcase
    end
endmodule
/**********************************
**/
module controller(
   input [3:0]opcode,
   input pos, //most significant bit
   input clk, reset,
   //input am,
   output reg IR enable, MD enable, MA enable, PC enable, AC enable, C enable,
   output reg MUX1, MUX3,
   output reg [2:0] ALU control,
   output reg [1:0] MUX2,
   output reg RWn
);
reg [4:0] present_state, next_state;
   parameter S00 = 5'h00, S01 = 5'h01, S02 = 5'h02, S03 = 5'h03, S04 = 5'h04,
   S05 = 5'h05, S06 = 5'h06, S07 = 5'h07, S08 = 5'h08, S09 = 5'h09, S10 = 5'h0A,
   S11 = 5'h0B, S12 = 5'h0C, S13 = 5'h0D, S14 = 5'h0E, S15 = 5'h0F, S16 = 5'h1O;
   initial begin
       present state=S00;
    end
    always@(negedge clk or posedge reset) begin
       if (reset == 1'b1) begin
          present state=S00;
       else begin
           present state = next state;
       end
    end
```

```
always @(present_state) begin
   case(present_state)
       S00: begin
           IR enable = 1;
           MD enable = 0;
           MA enable = 0;
           PC enable = 0;
           AC enable = 0;
           C enable= 0;
           MUX1 = 0;
           MUX2 = 0;
           MUX3 = 0;
           RWn=1;
           next_state = S01;
       end
       S01: begin
           IR_enable = 0;
           MD enable = 0;
           MA enable = 0;
           PC_enable = 1;
           AC_enable = 0;
           C enable= 0;
           MUX1 = 0;
           MUX2 = 0;
           MUX3 = 0;
           RWn=1;
           if(opcode[3:1] == 3'b000) begin
               next state = S02;
           else if (opcode[3:1] == 3'b011) begin
              next state = S03;
           end
           else if (opcode[3:1] == 3'b010) begin
               if (pos == 0) begin
                   if (opcode [0] == 1) begin //look at lsb of opcode
                     next_state = S04;
                  end
              else
                  next_state = S07;
           end
           else
                   next state = S00;
          end
          else
          next state = S08;
          end
       S02: begin
```

```
IR enable = 0;
   MD_enable = 0;
    MA_enable = 0;
    PC_enable = 0;
   AC enable = 1;
   C enable= 0;
   MUX1 = 0;
   MUX2 = 0;
   MUX3 = 0;
   RWn=1;
   next state = S00;
   ALU control[2:0] = 0;
end
S03: begin
   IR enable = 0;
   MD_enable = 0;
   MA_enable = 0;
   PC_enable = 0;
   AC enable = 1;
   C_enable= 0;
   MUX1 = 0;
   MUX2 = 0;
   MUX3 = 0;
   RWn=1;
   next_state = S00;
   ALU control[2:0] = 2;
end
S04: begin
   IR enable = 0;
   MD enable = 0;
   MA enable = 1;
   PC enable = 0;
   AC_enable = 0;
   C_enable= 0;
   MUX1 = 0;
   MUX2 = 0;
   MUX3 = 0;
   RWn=1;
   next state = S05;
end
S05: begin
   IR enable = 0;
   MD enable = 1;
   MA_enable = 0;
   PC enable = 0;
   AC enable = 0;
   C enable= 0;
   MUX1 = 0;
   MUX2 = 0;
   MUX3 = 1;
   RWn=1;
   next_state = S06;
end
```

```
S06: begin
   IR_enable = 0;
   MD enable = 0;
   MA_enable = 0;
   PC enable = 1;
   AC enable = 0;
   C enable= 0;
   MUX1 = 0;
   MUX2 = 2;
   MUX3 = 0;
   RWn=1;
   next_state = S00;
end
S07: begin
   IR_enable = 0;
   MD_enable = 0;
   MA_enable = 0;
   PC_enable = 1;
   AC_enable = 0;
   C_enable= 0;
   MUX1 = 0;
   MUX2 = 1;
   MUX3 = 0;
   RWn=1;
   next state = S00;
end
S08: begin
   IR enable = 0;
   MD enable = 0;
   MA enable = 1;
   PC enable = 0;
   AC enable = 0;
   C_enable= 0;
   MUX1 = 0;
   MUX2 = 0;
   MUX3 = 0;
   RWn=1;
   if (opcode[3:1] == 3'b100) begin
      if (opcode[0] ==1) begin
           next_state = S09;
       end
       else if (opcode[0] ==0) begin
           next_state = S11;
       end
   end
   else
       next_state = S12;
  end
S09: begin
   IR_enable = 0;
   MD_enable = 1;
   MA enable = 0;
```

```
PC enable = 0;
    AC_enable = 0;
    C_enable= 0;
   MUX1 = 0;
   MUX2 = 0;
   MUX3 = 1;
   RWn=1;
   next state = S10;
end
S10: begin
   IR enable = 0;
   MD_enable = 0;
   MA_enable = 1;
   PC enable = 0;
   AC enable = 0;
   C_enable= 0;
   MUX1 = 1;
   MUX2 = 0;
   MUX3 = 0;
   RWn=1;
   next_state = S11;
end
S11: begin
   IR_enable = 0;
   MD enable = 0;
   MA_enable = 0;
   PC enable = 0;
   AC enable = 1;
   C enable= 0;
   MUX1 = 0;
   MUX2 = 0;
   MUX3 = 1;
    ALU control[2:0]=3;
    #3 RWn=0;
   next_state = S00;
end
S12: begin
   IR_enable = 0;
   MD enable = 1;
   MA_enable = 0;
   PC_enable = 0;
   AC_enable = 0;
   C enable= 0;
   MUX1 = 0;
   MUX2 = 0;
   MUX3 = 1;
   RWn=1;
   if (opcode[0] == 1) begin
      next state = S13;
   end
   else begin
       if (opcode[3:1] ==3'b001)
            next_state = S15;
```

```
else
       next_state = S16;
   end
   end
S13: begin
   IR_enable = 0;
   MD_enable = 0;
   MA enable = 1;
   PC_enable = 0;
   AC_enable = 0;
   C_enable= 0;
   MUX1 = 1;
   MUX2 = 0;
   MUX3 = 0;
   RWn=1;
   next_state = S14;
end
S14: begin
   IR enable = 0;
   MD enable = 1;
   MA_enable = 0;
   PC enable = 0;
   AC enable = 0;
   C enable= 0;
   MUX1 = 0;
   MUX2 = 0;
   MUX3 = 1;
   RWn=1;
   if (opcode[3:1] ==3'b001) begin
      next state = S15;
  end
   else
      next_state = S16;
  end
S15: begin
   IR_enable = 0;
   MD_enable = 0;
   MA_enable = 0;
   PC_enable = 0;
   AC enable = 1;
   C_enable= 1;
   MUX1 = 0;
   MUX2 = 0;
   MUX3 = 0;
   RWn=1;
   ALU_control[2:0]=1;
   next state = S00;
end
S16: begin
   IR enable = 0;
```

```
MD enable = 0;
             MA_enable = 0;
              PC_enable = 0;
              AC enable = 1;
             C enable= 0;
             MUX1 = 0;
             MUX2 = 0;
             MUX3 = 0;
             RWn=1;
             ALU_control[2:0]=4;
             next state = S00;
   endcase
   end
endmodule
*/
module data_path(
  input CLK,
   input reset,
   input MD_enable,
   input IR enable,
   input AC enable,
   input PC_enable,
   input MA enable,
   input C enable,
   input [2:0]alu_control,
   input mux1,
   input mux3,
   input [1:0] mux2,
   input [15:0] RD,
   output reg [15:0] WD,
   output reg [11:0] A,
   output reg [15:0] IR
);
    reg [15:0] AC;
    reg [15:0] MD;
    reg [11:0] PC;
    reg [11:0] MA;
    reg C;
    wire [15:0] y;
    wire co;
  initial begin
       if( reset == 1'b1) begin
          PC[11:0] = 12'h000;
```

```
MA[11:0] = 12'h000;
        MD[15:0] = 16'h0000;
        AC[15:0] = 16'h0000;
        IR[15:0] = 16'h0000;
        C = 1'b0;
       A[11:0] = 12'h000;
     end
end
always @ (posedge CLK ) begin
   if ( MD enable == 1'b1 ) begin
        #1 MD = RD;
    end
    else if ( IR enable == 1'b1 ) begin
       #1 IR = RD;
    end
end
//Call ALU... And reassigning AC when needed
alu op3 ( AC, MD, alu_control, C, y, co);
always @ (posedge CLK ) begin
   if ( AC enable == 1'b1) begin
        #1 AC = y;
    end
    if (C_enable == 1'b1 ) begin
        C = co;
    end
end
//Mux 1
always @ (posedge CLK) begin
   if ( mux1 == 1'b0 && MA enable == 1) begin
       MA = IR[11:0];
    else if ( mux1 == 1'b1 \&\& MA enable == 1 ) begin
       MA = MD[11:0];
    end
end
//Mux 2
always @ (posedge CLK) begin
    if ( mux2 == 2'b00 \&\& PC enable == 1) begin
       PC = PC + 1;
    end
    else if ( mux2 == 2'b01 && PC_enable == 1 ) begin
        PC = IR[11:0];
    else if ( mux2 == 2'b10 && PC enable == 1 ) begin
       PC = MD[11:0];
    end
end
//Mux 3
always @ (posedge CLK) begin
   if (mux3 == 1'b0) begin
       A = PC;
```

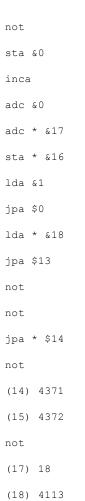
```
end
       else if ( mux3 == 1'b1 ) begin
         A = MA;
       end
   end
    always @ (posedge CLK) begin
     WD = AC;
    end
endmodule
/********************************
module simpleCPU(
   input CLK, reset,
   input [15:0] RD,
   output [15:0] WD,
   output RWn,
   output [11:0] A
);
   wire mux1, mux3, IR_enable, MD_enable, AC_enable, MA_enable, PC_enable, C_enable;
   wire [1:0] mux2;
   wire[2:0]alu control;
   wire [15:0] IR;
   data_path m2( CLK, reset, MD_enable, IR_enable, AC_enable, PC_enable, MA_enable,
C enable,
   alu control[2:0], mux1, mux3, mux2[1:0], RD[15:0], WD[15:0], A[11:0], IR[15:0]);
   controller m3( IR[15:12], WD[15], CLK, reset, IR_enable, MD_enable, MA_enable, PC_enable,
AC_enable, C_enable,
   mux1, mux3,alu control[2:0],mux2[1:0], RWn );
endmodule
```

## d. Verilog test bench code

endmodule

```
`timescale 1ns / 1ps
module procssor tb;
// Inputs
   reg CLK=0;
   reg reset=1;
   reg [15:0] RD;
   wire [11:0]A;
   wire [15:0] WD;
   wire RWn;
   reg [15:0] memory[0:18];
   initial begin
       $readmemh("C:/Users/ngol1.LIONS.006/Desktop/memory.dat", memory);
    end
   // Instantiate the Unit Under Test (UUT)
   simpleCPU uut (
   .CLK(CLK),
   .reset(reset),
   .RD(RD),
   .WD(WD),
    .A(A),
    .RWn(RWn)
   );
   initial begin
       reset <= 1;
       #1 reset <=0;
    end
    always begin
     #2 CLK = ~CLK;
    end
    always @* begin
       if (RWn == 1'b0) begin
           memory[A] = WD;
       else begin
          if (RWn == 1'b1)
              RD = memory[A];
       end
    end
    initial #250 $finish;
```

## e. Assembly language test program



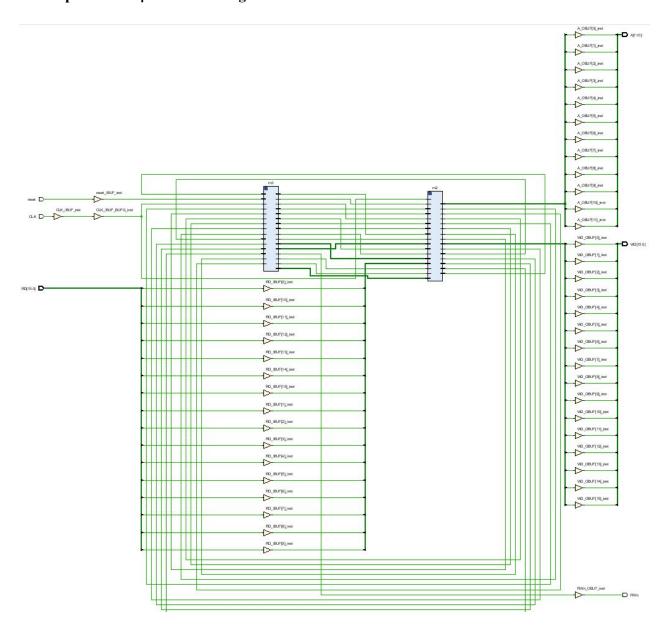
# f. Machine language test program

0000	0000
@001	8000
@002	6000
0003	2000
@004	3011
@005	9010
@006	A001
@007	4000
8009	В012
@009	400C
@00A	0000
@00B	0000
@00C	500D
@00D	0000
@00E	1113
@00F	1114
@010	0000
@011	0012
@012	1011

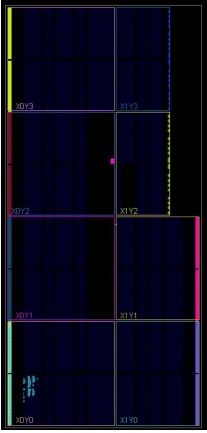
# g. Waveforms from verification of design

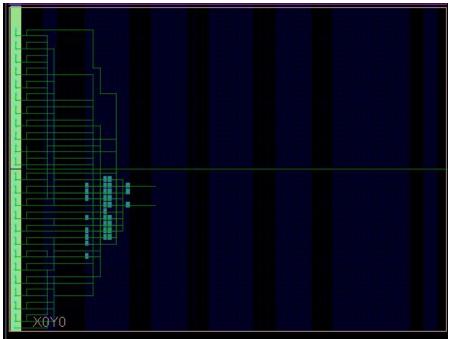
Name	Value	0.007 ns													
15 CLK	value	0 ns	5 ns	10 ns	15 ns	20 ns	25 ns	30 ns	35 ns	40 ns	45 ns	50 ns	55 ns	60 ns	65 ns
™ reset	1														
> MRD[15:0]	0000	0000	X	8000	X	6000	(0000 tfff	6000	20	000	Х 3(	11 X	ffff	3011	9010
> M A[11:0] > M WD[15:0]	000 XXXX	000	X	001	X	002	000	002		03	Χ	4 X	000	004	005
™ RWn	1	(2000)	0000	Χ		ffff			0000			0001			0000
∨ W memory[][	15:0 0000,800	0000,8000,600	0,2000,3011,901	0,a001,4000,b	012,400c,0000,	0000,5004,0000,	1130 X	tttt,	000,6000,2000,	,3011,9010,a001	,4000,b012,400	,0000,0000,50	d,0000,1113,1	114,0000,0012,	.1011
> 😼 [0][15:0]				0000			X				ttt				
> 🕨 [1][15:0]									8000						
> <b>1</b> [2][15:0] > <b>1</b> [3][15:0]									2000						
> 14][15:0]		$\vdash$							3011						
> 😼 [5][15:0]									9010						
> 🖼 [6][15:0]									a001						
> <b>1</b> [7][15:0] > <b>1</b> [8][15:0]									4000 b012						
> 1 [9][15:0]									400c						
> 😼 [10][15:0									0000						
> 🛂 [11][15:0									0000						
> 😼 [12][15:0 > 😼 [13][15:0									500d						
> 13[15](15)(									1113						
> 15][15]									1114						
> 🛂 [16][15:0	0000								0000						
> 17][15:0									0012						
> 😼 [18][15:0	0] 1011								1011						
Name	Value	70 ns	75 ns	80 ns	85 ns	90 ns	95 ns	100 ns	105 ns	110 ns	115 ns	120 ns	125 ns	130 ns	135 ns
<sup>™</sup> CLK	0														
le reset	1														
> M RD[15:0] > M A[11:0]	0000	9010	0012 X 90			010 X	a001	X 0000		000	a001 X	4000	X 8000	4000	
> W WD[15:0]	XXXX	005	000			5 X	006	1012	X 006	000	006	0000	V 001	V 007	8000
₩ RWn	1					1								^	
∨ ■ memory[][1		ttt	1,8000,6000,200	0,3011,9010,			\$00d,0000,1113,	1114,0000,001	,1011	1012,8000,	6000,2000,3011	9010,m001,4000		00,0000,5004,0	000,1113,1114,00
> 10][15:0]						ttt				X			1012		
> <b>1</b> [1][15:0] > <b>1</b> [2][15:0]									6000						
> 13[15:0]									2000						
> 😼 [4][15:0]	3011														
[4][10.0]									3011						
> 😼 [5][15:0]	9010								9010						
> <b>14</b> [5][15:0] > <b>14</b> [6][15:0]	9010 a001								9010 a001						
> <b>16</b> [5][15:0] > <b>16</b> [6][15:0] > <b>16</b> [7][15:0]	9010 a001 4000								9010						
> 14 [5][15:0] > 14 [6][15:0] > 14 [6][15:0] > 14 [7][15:0] > 14 [8][15:0] > 14 [9][15:0]	9010 a001 4000 b012 400c								9010 a001 4000						
> Md [5][15:0] > Md [6][15:0] > Md [7][15:0] > Md [8][15:0] > Md [9][15:0] > Md [10][15:0]	9010 a001 4000 b012 400c 0) 0000								9010 a001 4000 b012 400c 0000						
> = [5][15:0] > = [6][15:0] > = [7][15:0] > = [8][15:0] > = [9][15:0] > = [10][15:0	9010 a001 4000 b012 400c 0] 0000 0] 0000								9010 a001 4000 b012 400c 0000						
> = [5](15:0) > = [6](15:0) > = [7](15:0) > = [8](15:0) > = [9](15:0) > = [10](15:0 > = [11](15:0	9010 4001 4000 b012 400c 0] 0000 0] 0000								9010 a001 4000 b012 400c 0000 0000						
> = [5][15:0] > = [6][15:0] > = [7][15:0] > = [8][15:0] > = [9][15:0] > = [10][15:0	9010 4000 5012 4000 4000 0000 0000 0000 0000 0000 0000 0000								9010 a001 4000 b012 400c 0000						
> [5](15.0] > [6](15.0] > [6](15.0] > [6](15.0] > [6](15.0) > [6](15.0) > [10](15.0) > [11](15.0) > [12](15.0) > [13](15.0) > [13](15.0) > [14](15.0) > [14](15.0)	9010 9010 4000 1 4000 1 400c 1 400c 1 0000 1 0000 1 500d 1 0000 1 1113 1 1114								9010 a001 4000 b012 400c 0000 5004 0000 1113						
> [5]15.0] > [6]15.0] > [6]15.0] > [6]15.0] > [6]15.0] > [10]15.0] > [10]15.0 > [11]15.0 > [13]15.0 > [14]15.0 > [14]15.0 > [14]15.0	9010 9010 9011 9012 9000 9000 9000 9000 91113 9000 91114								9010 a001 4000 b012 4000 0000 0000 5004 0000 1113 1114						
> [5](15.0) > [6](15.0) > [6](15.0) > [6](15.0) > [6](15.0) > [11](15.0) > [12](15.0) > [14](15.0) > [14](15.0) > [14](15.0) > [14](15.0) > [14](15.0) > [14](15.0)	9010 4000 4000 5012 400c 00 0000 00 0000 01 500d 01 0000 01 1113 01 1114 01 0000 01 0000								9010 a001 4000 b012 400c 0000 500d 0000 1113 1114 0000 0012						
> [5]15.0] > [6]15.0] > [6]15.0] > [6]15.0] > [6]15.0] > [10]15.0] > [10]15.0 > [11]15.0 > [13]15.0 > [14]15.0 > [14]15.0 > [14]15.0	9010 4000 4000 5012 400c 00 0000 00 0000 01 500d 01 0000 01 1113 01 1114 01 0000 01 0000								9010 a001 4000 b012 4000 0000 0000 5004 0000 1113 1114						
> [5](15.0) > [6](15.0) > [6](15.0) > [6](15.0) > [6](15.0) > [11](15.0) > [12](15.0) > [14](15.0) > [14](15.0) > [14](15.0) > [14](15.0) > [14](15.0) > [14](15.0)	9010 4000 4000 5012 400c 00 0000 00 0000 01 500d 01 0000 01 1113 01 1114 01 0000 01 0000								9010 a001 4000 b012 400c 0000 500d 0000 1113 1114 0000 0012						
Sign	9010 4001 4002 400c 0) 0000 0) 0000 0) 0000 0) 500d 0) 500d 0) 1113 0) 1114 0) 0000 0) 011	146 ng	180 ma	las ne	leo ne	hes no	[170 ns	175 ns	9010 a001 4000 b012 400c 0000 500d 0000 1113 1114 0000 0012	les ne	190 ne	198 na		205 ns	210 ns
> m (5)(50) > m (5)(50) > m (5)(150) > m (5)(150) > m (5)(150) > m (5)(150) > m (10)(150)   Maximum   Value   Taylor   Cut   D	9010 a001 a001 b012 400c 0) 0000 0) 0000 0) 1113 0) 1114 0) 1000 0) 0000 0) 01 1013	jag ng	listy me	155 ne	, 1469 ns	lies ne	179 as	175 ne	9010 a001 4000 b012 400e 0000 0000 5004 0000 1113 1114 0000 0012	las ne	189 ns	, 135 ne	200 ns	205 ne	210 ns
M	9010 a001 4000 b012 400c 0000 0000 0000 0000 001 1114 00000 0001 00000 001 1114 00000 001 1011 1								9010 4001 4001 4000 6012 4000 6012 4000 6000 6000 1113 6000 6011 114 6000 6011 115 1114			105 24			
Marie   Mari	9010 4001 4001 4001 4002 000 000 000 000 000 000 01 1113 01 1114 00 01 1040 01 01 0000 01 01 01 0000 01 01 01 0000 01 01	2 \	400c	X 1011	400c X	0012	400c	X 0000 X	9010 4001 4001 4000 4000 4000 4000 4000	X	0000	135 ng	1012	X	8000
M   G(150)   M	9010 a001 4000 b012 400c 0000 0000 0000 0000 001 1114 00000 0001 00000 001 1114 00000 001 1011 1	z X		X 1011	400c X				9010 4001 4001 4000 6012 4000 6012 4000 6000 6000 1113 6000 6011 114 6000 6011 115 1114			195 ne			
March   Marc	9010 4001 4001 4001 4001 4002 01 0000 01 0000 01 0000 01 1113 01 1114 01 01 1140 00 01 0000 01 001 00	2 \	400c	X 1011	400c X	0012 X 011 X	400c 009	X 0000 X	9010 4001 4001 4000 4000 4000 4000 4000	X	0000 00d 0012	X	1012	X	8000
Marie   Mari	9010 4001 4001 4001 4001 4002 01 0000 01 0000 01 0000 01 1113 01 1114 01 01 1140 00 01 0000 01 001 00	2 \	400c	X 1011	400c X	0012 X 011 X	400c	0000 X	9010 4001 4001 4000 4000 4000 4000 4000	X	0000 00d 0012	X	1012	X	8000
Marie   Value   Name   Name   Value   Name	9010 4001 4001 4001 4001 4002 01 0000 01 0000 01 0000 01 1113 01 1114 01 01 1140 00 01 0000 01 001 00	2 \	400c	X 1011	400c X	0012 X 011 X	400c 009	0000 X 000a X 000a X 0000, 5012,400c, 1012	9010 4001 4001 4000 4000 4000 4000 4000	X	0000 00d 0012	X	1012	X	8000
Marie   Mari	9010 4001 4001 4001 4001 4002 01 0000 01 0000 01 0000 01 1113 01 1114 01 01 1140 00 01 0000 01 001 00	2 \	400c	X 1011	400c X	0012 X 011 X	400c 009	0000 X	9010 4001 4001 4000 4000 4000 4000 4000	X	0000 00d 0012	X	1012	X	8000
Name Value  Name (CL)	9010 4001 4001 4001 4001 4002 01 0000 01 0000 01 0000 01 1113 01 1114 01 01 1140 00 01 0000 01 001 00	2 \	400c	X 1011	400c X	0012 X 011 X	400c 009	0000 00a 00a 000,0012,400c, 1012 8000	9010 4001 4001 4000 4000 4000 4000 4000	X	0000 00d 0012	X	1012	X	8000
Name Value  Name (CL)	9010 4001 4001 4001 4001 4002 01 0000 01 0000 01 0000 01 1113 01 1114 01 01 1140 00 01 0000 01 001 00	2 \	400c	X 1011	400c X	0012 X 011 X	400c 009	0000 X 00a X 000, 5012, 400c, 1012 8000 6000 2000	9010 4001 4001 4000 4000 4000 4000 4000	X	0000 00d 0012	X	1012	X	8000
Name Value  Name V	9010 4001 4001 4001 4001 4002 01 0000 01 0000 01 0000 01 1113 01 1114 01 01 1140 00 01 0000 01 001 00	2 \	400c	X 1011	400c X	0012 X 011 X	400c 009	0000 X 00a X 000, 5012, 400c, 1012 8000 6000 2000 3011 9010	9010 4001 4001 4000 4000 4000 4000 4000	X	0000 00d 0012	X	1012	X	8000
Name Value  Name (CL)	9010 4001 4001 4001 4001 4002 01 0000 01 0000 01 0000 01 1113 01 1114 01 01 1140 00 01 0000 01 001 00	2 \	400c	X 1011	400c X	0012 X 011 X	400c 009	0000 X 000 X 000 X 000 X 1012 8000 6000 2000 3011 9010	9010 4001 4001 4000 4000 4000 4000 4000	X	0000 00d 0012	X	1012	X	8000
Name Value  Name V	9010 4001 4001 4001 4001 4002 01 0000 01 0000 01 0000 01 1113 01 1114 01 01 1140 00 01 0000 01 001 00	2 \	400c	X 1011	400c X	0012 X 011 X	400c 009	0000 X 00a X 000, 5012, 400c, 1012 8000 6000 2000 3011 9010	9010 4001 4001 4000 4000 4000 4000 4000	X	0000 00d 0012	X	1012	X	8000
Name Value  Name V	9010 4001 4001 4001 4001 4002 01 0000 01 0000 01 0000 01 1113 01 1114 01 01 1140 00 01 0000 01 001 00	2 \	400c	X 1011	400c X	0012 X 011 X	400c 009	0000 X 000 X	9010 4001 4001 4000 4000 4000 4000 4000	X	0000 00d 0012	X	1012	X	8000
M   G(15)	9010 4001 4001 4001 4001 4002 01 0000 01 0000 01 0000 01 1113 01 1114 01 01 1140 00 01 0000 01 001 00	2 \	400c	X 1011	400c X	0012 X 011 X	400c 009	0000 X 0000 X 000 0000 0000 0000 0000	9010 4001 4001 4000 4000 4000 4000 4000	X	0000 00d 0012	X	1012	X	8000
Name Value  Name (1915-0)  Name (191	9010 4001 4001 4001 4001 4002 01 0000 01 0000 01 0000 01 1113 01 1114 01 01 1140 00 01 0000 01 001 00	2 \	400c	X 1011	400c X	0012 X 011 X	400c 009	X 0000 X 000	9010 4001 4001 4000 4000 4000 4000 4000	X	0000 00d 0012	X	1012	X	8000
A	9010 4001 4001 4001 4001 4002 01 0000 01 0000 01 0000 01 1113 01 1114 01 01 1140 00 01 0000 01 001 00	2 \	400c	X 1011	400c X	0012 X 011 X	400c 009	0000 X 0000 X 000 0000 0000 0000 0000	9010 4001 4001 4000 4000 4000 4000 4000	X	0000 00d 0012	X	1012	X	8000
Name Value  Name (1915-0)  Name (191	9010 4001 4001 4001 4001 4002 01 0000 01 0000 01 0000 01 1113 01 1114 01 01 1140 00 01 0000 01 001 00	2 \	400c	X 1011	400c X	0012 X 011 X	400c 009	X 0000 X 000	9010 4001 4001 4000 4000 4000 4000 4000	X	0000 00d 0012	X	1012	X	8000
Name   Value	9010 4001 4001 4001 4001 4002 01 0000 01 0000 01 0000 01 1113 01 1114 01 01 1140 00 01 0000 01 001 00	2 \	400c	X 1011	400c X	0012 X 011 X	400c 009	000, 012, 400e, 000, 012, 400e, 00000, 000000	9010 4001 4001 4000 4000 4000 4000 4000	X	0000 00d 0012	X	1012	X	8000
March   Marc	9010 4001 4001 4001 4001 4002 01 0000 01 0000 01 0000 01 1113 01 1114 01 01 1140 00 01 0000 01 001 00	2 \	400c	X 1011	400c X	0012 X 011 X	400c 009	000 012,400e, 000 012,400e, 1012 8000 2000 3011 9010 4000 0000 0000 0000 1113	9010 4001 4001 4000 4000 4000 4000 4000	X	0000 00d 0012	X	1012	X	8000
Name   Value	9010 4001 4001 4001 4001 4002 01 0000 01 0000 01 0000 01 1113 01 1114 01 01 1140 00 01 0000 01 001 00	2 \	400c	X 1011	400c X	0012 X 011 X	400c 009	X 0000 X 000	9010 4001 4001 4000 4000 4000 4000 4000	X	0000 00d 0012	X	1012	X	8000

h. Design schematic from Xilinx synthesis of design with no area constraint and clock period of 1  $\mu S$  as the timing constraint



# i. Routed Design





## j. Post place and route timing report

```
Copyright 1986-2017 Xilinx, Inc. All Rights Reserved.
______
| Tool Version : Vivado v.2017.4 (win64) Build 2086221 Fri Dec 15 20:55:39 MST 2017
| Date : Sun Nov 3 13:12:13 2019
Host : STEM241-07 running 64-bit major release (build 9200)
| Command : report_timing_summary -max_paths 10 -file simpleCPU_timing_summary_routed.rpt
-rpx simpleCPU timing summary routed.rpx -warn on violation
| Design : simpleCPU
          : 7k70t-fbv676
| Device
| Speed File : -1 PRODUCTION 1.12 2017-02-17
______
_____
Timing Summary Report
______
| Timer Settings
| -----
______
 Enable Multi Corner Analysis : Yes
 Enable Pessimism Removal
                                 : Yes
                                 : Nearest Common Node
 Pessimism Removal Resolution
 Enable Input Delay Default Clock
 Enable Preset / Clear Arcs
 Disable Flight Delays
 Ignore I/O Paths
 Timing Early Launch at Borrowing Latches : false
 Corner Analyze
              Analyze
 Name Max Paths Min Paths
 Slow Yes Yes
Fast Yes Yes
check_timing report
Table of Contents
-----
1. checking no clock
2. checking constant clock
3. checking pulse width clock
4. checking unconstrained internal endpoints
5. checking no input delay
6. checking no output delay
7. checking multiple clock
8. checking generated clocks
9. checking loops
10. checking partial_input_delay
11. checking partial output delay
12. checking latch loops
1. checking no_clock
______
There are 107 register/latch pins with no clock driven by root clock pin: CLK (HIGH)
```

There are 17 register/latch pins with no clock driven by root clock pin:  $m3/ALU\_control\_reg[0]/Q$  (HIGH)

There are 17 register/latch pins with no clock driven by root clock pin: m3/ALU control reg[1]/Q (HIGH)

There are 17 register/latch pins with no clock driven by root clock pin: m3/ALU control reg[2]/Q (HIGH)

There are 19 register/latch pins with no clock driven by root clock pin: m3/FSM sequential present state reg[0]/Q (HIGH)

There are 19 register/latch pins with no clock driven by root clock pin:  $m3/FSM\_sequential\_present\_state\_reg[1]/Q$  (HIGH)

There are 19 register/latch pins with no clock driven by root clock pin: m3/FSM sequential present state reg[2]/Q (HIGH)

There are 19 register/latch pins with no clock driven by root clock pin:  $m3/FSM\_sequential\_present\_state\_reg[3]/Q$  (HIGH)

There are 19 register/latch pins with no clock driven by root clock pin:  $m3/FSM\_sequential\_present\_state\_reg[4]/Q$  (HIGH)

#### 2. checking constant\_clock

-----

There are 0 register/latch pins with constant clock.

#### 3. checking pulse width clock

\_ \_ \_

There are 0 register/latch pins which need pulse width check

#### 4. checking unconstrained\_internal\_endpoints

-----

There are 220 pins that are not constrained for maximum delay. (HIGH)

There are 0 pins that are not constrained for maximum delay due to constant clock.

#### 5. checking no\_input\_delay

\_\_\_\_\_

There are 17 input ports with no input delay specified. (HIGH)

There are 0 input ports with no input delay but user has a false path constraint.

#### 6. checking no output\_delay

-----

There are 29 ports with no output delay specified. (HIGH)

There are 0 ports with no output delay but user has a false path constraint

There are 0 ports with no output delay but with a timing clock defined on it or propagating through it

#### 7. checking multiple\_clock

\_\_\_\_\_

There are 0 register/latch pins with multiple clocks.

#### 8. checking generated clocks

\_\_\_\_\_\_

There a	re 0 genera	ated cloc	ks that are no	t conne	cted to a d	clock s	ource.		
0 aboats	ina loona								
9. check	ing loops								
There a	re 0 combin	national	loops in the d	esign.					
	king partia								
			th partial inp	ut dela	y specified	d.			
	king partia								
	re 0 ports		tial output de	lay spe	cified.				
	king latch <sub>_</sub>	_							
	 re 0 combin		latch loops in	the de	sian throug	nh latc	h inpu	t	
			-		-		-		
Design	Timing Sur	mmary							
1									
THS Fail	ing Endpoi al Endpoint	nts THS ts	FNS Failing End Total Endpoin	ts	WPWS(ns)	TPV	IS (ns)		
NA	NA	NA	NA	NA NA		NA	NA	NA	NZ NZ
NA									
All user	specified	timing o	onstraints are	met.					
Clock	Summary								
	- ,			_					
CTOCK M	aveform(ns)	)	Period(ns)	r'req	uency(MHz)				

Clock WHS(ns) THS(ns) TPWS Failing Endpoints	THS Failing Endpoints	TNS Failing Endpoints THS Total Endpoints	<del>_</del>
   Inter Clock Table 			
From Clock To Cloc Endpoints WHS(ns)	THS(ns) THS Failin	TNS(ns) TNS Failin ng Endpoints THS Total En	ndpoints
Other Path Groups Tab	le 		
	S(ns) THS(ns) THS		

## **Discussion:**

In this lab we were able to correctly implement all of the components of the lab. The various components of the design code, the ALU, controller, and datapath all worked correctly. They produced the correct waveforms. The testbench also was correct. The fully implemented processor's waveforms were correct. The RTL's were correct as well as the number of clock cycles for each instruction and addressing type. The assembly language program created the

correct machine language program. The design schematic from the synthesis with no area constraint was right as well as the routed design and timing report.

## **Conclusions:**

All of the components of the lab worked as expected. The ALU, control, and datapath all worked correctly individually. They produced the correct outputs and waveforms. They also then worked together once paired with the testbench. This shows that the processor works as intended. All of the diagrams were correct and showed the right information.