EE 443
Digital and Computer Analysis and Design Laboratory

Lab *#5*: *16-bit ALU and datapath for R-format instructions*
3/19/2013

Prepared by:
Matt Van Veldhuizen

Lab Participation
Provide the percent participation in lab by each lab partner (normally, 50%, 50% for 2 partners):

Lab member name	**Percent participation**
Matt Van Veldhuizen	100%

Does your solution work the way it's supposed to work? ☐YES ☐NO[1]

[1]If your answer is NO, please explain in your report.

Instructor/TA comments and grading:

1. Objective and Background

In this lab we were tasked to create a 16-bit ALU and datapath for R-format instructions. This ALU was based on the 32-bit RISC machine that was presented in our book. We had to go through the process of creating components that would add, or, and, and invert.

2. Equipment

Altera Quartus II, version 9.1 or newer.

3. Procedure

For each step in the procedure, we needed to record the timing delays for each component and test them, making sure they did what was needed. The first component that was created was a 4-bit adder, which was based on a commercial *Binary Adder with Fast Carry - 74LS283* which was given to us in a hand out. This component was named ADD4, which is shown in Figure 1 with the timing delays shown in Table 1 and the simulation shown in Figure 2. The next component to be created were the 4-bit bitwise AND (BWAND4) and OR (BWOR4), as seen in Figures 3 and 4 respectively. With the timing delays for BWAND4 and BWOR4 seen in Tables 2 and 3, with their simulations shown in Figures 5 and 6 .We were to reuse the 4-bit multiplexer (MUX4X4) that was created in the last lab as seen in Figure 7 with the timing delay shown in Table 4 and the simulation shown in Figure 8. The next component was a 4-bit programmable inverter with a control input, called PINV4. This inverter was to be used with the SUB and SLT operations, this can be seen in Figure 9, with the timing delay shown in Table 5 and the simulation shown in Figure 10. Once all of these components were created a component library was created to house them as seen in Figure 11. After the component library was created the 4-bit ALU (ALU4) was created using these components as seen in Figure 12 with the timing delay shown in Table 6 with the simulation shown in Figure 13. For this ALU we were to use the multiplexer approach when designing it. The last component of this lab was to create a 16-bit ALU (ALU16) which uses 4 ALU4s, as seen in Figure 14 with the timing delay in Table 7. After the ALU was created, the ALU was ran under several simulations, to check that all operations were working as expected this can be seen in Figure 15.

4. Results

All of the components that were needed to create the ALU4, were simple to implement. However the ALU4 gave me the most problems. For instance, it took me awhile to realize that port maps cannot be in a process, and another big issue was that I was thinking too much like a C++ programmer. With the help of some classmates I was able to figure out the problems I was having and was able to make it work.

5. Discussion and Questions

The estimated maximum clock rate for this ALU to sustain if it were clock would be 27Mhz. This is because of all of the time needed for all of the components of the ALU to compute and the setup times. The propagation delay is 15.990ns, so the clock rate is 27MHz.

6. Conclusion

The point of this lab was to create a 16-bit ALU and datapath for R-format instructions. This was done by making several components, an adder, or, and, and inverter.

7. Attachments

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4   entity ADD4 is
5      port( A, B : in std_logic_vector(3 downto 0);
6            CIN : in std_logic;
7            D : out std_logic_vector(3 downto 0);
8            COUT : out std_logic
9            );
10  end ADD4;
11
12  architecture logic of ADD4 is
13
14  signal sum : std_logic_vector(3 downto 0);
15  signal cG : std_logic_vector(3 downto 0);
16  signal cP : std_logic_vector(3 downto 0);
17  signal cI : std_logic_vector(3 downto 1);
18
19  begin
20     sum <= A xor B;
21     cG <= A and B;
22     cP <= A or B;
23     process(cG, cP, cI)
24     begin
25        cI(1) <= cG(0) or (cP(0) and CIN);
26        for i in 1 to 2 LOOP
27              cI(i+1) <= cG(i) or (cP(i) and cI(i));
28              end loop;
29        COUT <= cG(3) or (cP(3) and cI(3));
30     end process;
31
32     D(0) <= sum(0) xor CIN;
33     D(3 downto 1) <= sum(3 downto 1) xor cI(3 downto 1);
34  end logic;
```

Figure 1: ADD4.vhdl

Table 1: ADD4 Timing Delay

Progagation Delay						
	Input Port	Output Port	RR	RF	FR	FF
1	CIN	COUT	11.216			11.216
2	CIN	D[3]	11.213	11.213	11.213	11.213
3	A[1]	COUT	10.764			10.764
4	A[1]	D[3]	10.761	10.761	10.761	10.761
5	B[1]	COUT	10.594			10.594
6	B[1]	D[3]	10.591	10.591	10.591	10.591
7	CIN	D[2]	10.436	10.436	10.436	10.436
8	A[2]	COUT	10.404			10.404
9	A[2]	D[3]	10.401	10.401	10.401	10.401
10	B[2]	COUT	10.331			10.331
11	B[2]	D[3]	10.328	10.328	10.328	10.328
12	A[1]	D[2]	9.984	9.984	9.984	9.984
13	CIN	D[1]	9.933	9.933	9.933	9.933
14	B[1]	D[2]	9.814	9.814	9.814	9.814
15	B[3]	COUT	9.700			9.700
16	B[3]	D[3]	9.698	9.698	9.698	9.698
17	A[2]	D[2]	9.629	9.629	9.629	9.629
18	CIN	D[0]	9.492	9.492	9.492	9.492
19	A[1]	D[1]	9.485	9.485	9.485	9.485
20	B[2]	D[2]	9.434	9.434	9.434	9.434
21	A[3]	COUT	9.395			9.395
22	A[3]	D[3]	9.394	9.394	9.394	9.394
23	B[1]	D[1]	9.312	9.312	9.312	9.312
24	A[0]	COUT	8.120			8.120
25	A[0]	D[3]	8.117	8.117	8.117	8.117
26	B[0]	COUT	7.989			7.989

Figure 2: ADD4 Simulation

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4   entity BWAND4 is
5      port( X, Y : in std_logic_vector(3 downto 0);
6            D : out std_logic_vector(3 downto 0)
7            );
8   end BWAND4;
9
10  architecture logic of BWAND4 is
11  begin
12     D <= X AND Y;
13  end logic;
```

Figure 3: BWAND4.vhdl

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4   entity BWOR4 is
5      port( A, B : in std_logic_vector(3 downto 0);
6            D : out std_logic_vector(3 downto 0)
7            );
8   end BWOR4;
9
10  architecture logic of BWOR4 is
11  begin
12     D <= A OR B;
13  end logic;
```

Figure 4: BWOR4.vhdl

Table 2: BWAND4 Timing Delay

Progagation Delay						
	Input Port	Output Port	RR	RF	FR	FF
1	B[1]	D[1]	9.435			9.435
2	B[2]	D[2]	9.384			9.384
3	A[2]	D[2]	9.158			9.158
4	A[1]	D[1]	9.157			9.157
5	B[3]	D[3]	8.836			8.836
6	A[3]	D[3]	8.613			8.613
7	A[0]	D[0]	5.380			5.380
8	B[0]	D[0]	5.133			5.133

Table 3: BWOR4 Timing Delay

Progagation Delay						
	Input Port	Output Port	RR	RF	FR	FF
1	B[1]	D[1]	9.460			9.460
2	B[2]	D[2]	9.411			9.411
3	A[2]	D[2]	9.157			9.157
4	A[1]	D[1]	9.156			9.156
5	B[3]	D[3]	8.861			8.861
6	A[3]	D[3]	8.612			8.612
7	A[0]	D[0]	5.407			5.407
8	B[0]	D[0]	5.132			5.132

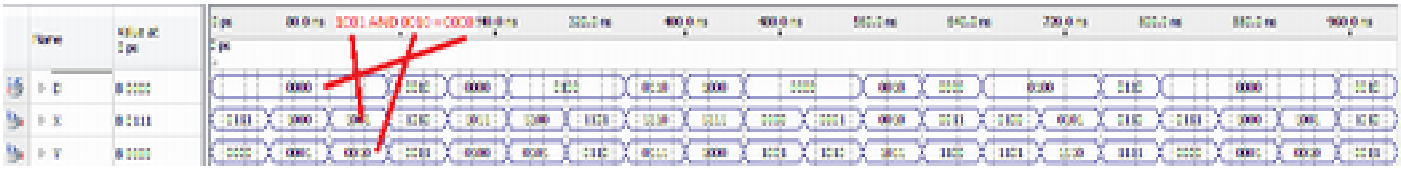

Figure 5: BWAND4 Simulation

Figure 6: BWOR4 Simulation

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4   entity MUX4x4 is
5      port(s : in std_logic_vector(1 downto 0);
6           d0, d1, d2, d3: in std_logic_vector(3 downto 0);
7           output: out std_logic_vector(3 downto 0)
8         );
9   end MUX4x4;
10
11  architecture logic of MUX4x4 is
12  begin
13     with s select output <= d0 when "00",
14                             d1 when "01",
15                             d2 when "10",
16                             d3 when "11";
17  end logic;
```

Figure 7: MUX4X4.vhdl

Table 4: MUX4X4 Timing Delay

Progagation Delay						
	Input Port	Output Port	RR	RF	FR	FF
1	d1[1]	output[1]	13.513			13.513
2	d0[0]	output[0]	13.445	13.445	13.445	13.445
3	d1[0]	output[0]	13.434	13.434	13.434	13.434
4	s[0]	output[1]	12.966	12.966	12.966	12.966
5	d0[1]	output[1]	12.925	12.925	12.925	12.925
6	d3[0]	output[0]	12.917			12.917
7	d2[1]	output[1]	12.832	12.832	12.832	12.832
8	s[0]	output[0]	12.788	12.788	12.788	12.788
9	d3[1]	output[1]	12.707			12.707
10	s[0]	output[2]	12.290	12.290	12.290	12.290
11	d0[2]	output[2]	12.289	12.289	12.289	12.289
12	d1[2]	output[2]	12.199	12.199	12.199	12.199
13	d2[2]	output[2]	11.553			11.553
14	d3[2]	output[2]	11.289			11.289
15	d1[3]	output[3]	11.203			11.203
16	s[0]	output[3]	10.639	10.639	10.639	10.639
17	d0[3]	output[3]	10.556	10.556	10.556	10.556
18	d2[3]	output[3]	10.530	10.530	10.530	10.530
19	d3[3]	output[3]	9.963			9.963
20	s[1]	output[1]	9.309	9.309	9.309	9.309
21	s[1]	output[0]	9.122	9.122	9.122	9.122
22	d2[0]	output[0]	8.716			8.716
23	s[1]	output[2]	8.457	8.457	8.457	8.457
24	s[1]	output[3]	6.967	6.967	6.967	6.967

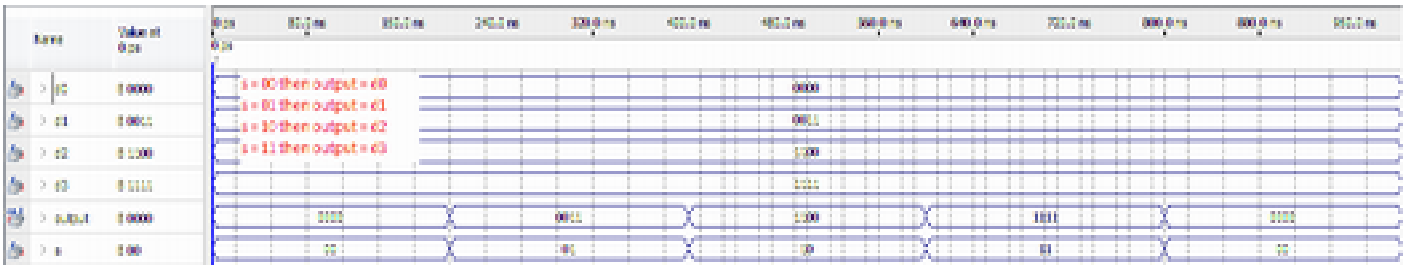

Figure 8: MUX4X4 Simulation

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4   entity PINV4 is
5      port( A : in std_logic_vector(3 downto 0);
6            EN : in std_logic;
7            D : out std_logic_vector(3 downto 0)
8            );
9   end PINV4;
10
11  architecture logic of PINV4 is
12  begin
13     process(EN)
14     begin
15        if EN = '1' then
16           D <= NOT A;
17        else
18           D <= A;
19        end if;
20     end process;
21  end logic;
```

Figure 9: PINV4.vhdl

Table 5: PINV4 Timing Delay

Progagation Delay						
	Input Port	Output Port	RR	RF	FR	FF
1	A[1]	D[1]	9.461	9.461	9.461	9.461
2	A[2]	D[2]	9.446	9.446	9.446	9.446
3	A[3]	D[3]	9.238	9.238	9.238	9.238
4	A[0]	D[0]	5.496	5.496	5.496	5.496
5	EN	D[1]	5.415	5.415	5.415	5.415
6	EN	D[0]	5.401	5.401	5.401	5.401
7	EN	D[2]	5.387	5.387	5.387	5.387
8	EN	D[3]	5.339	5.339	5.339	5.339

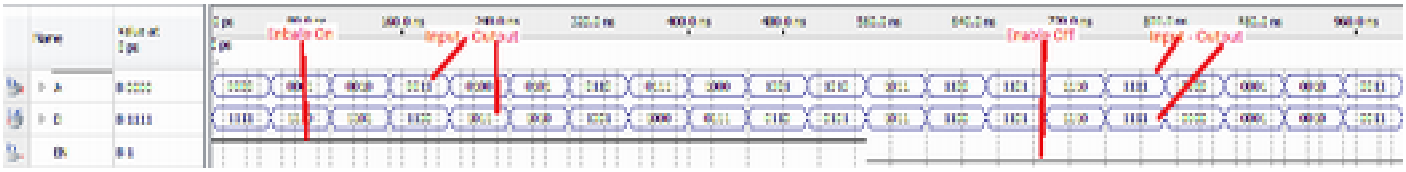

Figure 10: PINV4 Simulation

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4   package lib is
5      component ADD4
6         port( A, B : in std_logic_vector(3 downto 0);
7               CIN : in std_logic;
8               D : out std_logic_vector(3 downto 0);
9               COUT : out std_logic
10              );
11     end component;
12     component BWAND4
13        port( X, Y : in std_logic_vector(3 downto 0);
14              D : out std_logic_vector(3 downto 0)
15              );
16     end component;
17     component BWOR4
18        port( A, B : in std_logic_vector(3 downto 0);
19              D : out std_logic_vector(3 downto 0)
20              );
21     end component;
22     component MUX4x4
23        port( s : in std_logic_vector(1 downto 0);
24              d0, d1, d2, d3: in std_logic_vector(3 downto 0);
25              output: out std_logic_vector(3 downto 0)
26              );
27     end component;
28     component PINV4
29        port( A : in std_logic_vector(3 downto 0);
30              EN : in std_logic;
31              D : out std_logic_vector(3 downto 0)
32              );
33     end component;
34     component ALU4
35        port( A, B : in std_logic_vector(3 downto 0);
36              LESS, CIN : in std_logic;
37              SEL : in std_logic_vector(2 downto 0);
38              F : out std_logic_vector(3 downto 0);
39              COUT, OVERFLOW, SET, ZERO : out std_logic
40              );
41     end component;
42  end lib;
```

Figure 11: lib.vhdl

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3   use work.lib.all;
4
5   entity ALU4 is
6      port( A, B : in std_logic_vector(3 downto 0);
7            LESS, CIN : in std_logic;
8            SEL : in std_logic_vector(2 downto 0);
9            F : out std_logic_vector(3 downto 0);
10           COUT, OVERFLOW, SET, ZERO : out std_logic
11           );
12  end ALU4;
13
14  architecture logic of ALU4 is
15  signal s0, s1, s2, s3, s4, s5, s6, s7 : std_logic_vector(3 downto 0);
16  signal c0, c1 : std_logic;
17  begin
18
19  cADD: ADD4 port map(A, B, CIN, s0, c0);
20  cOR: BWOR4 port map(A, B, s1);
21  cAND: BWAND4 port map(A, B, s2);
22  cINV1: PINV4 port map(B, SEL(2), s4);
23  cSUB: ADD4 port map(A, s4, CIN, s3, c1);
24
25  process(LESS, s3, s6)
26  begin
27     if LESS = '1' then
28        s6 <= "0001";
29        SET <= s3(3);
30     else
31        s6 <= "0000";
32        set <= s3(3);
33     end if;
34  end process;
35
36  MUX0: MUX4X4 port map(SEL(1) & (SEL(2) or SEL(0)), s2, s1, s0, s3, s5);
37  MUX1: MUX4X4 port map('0' & (SEL(2) and SEL(0)), s5, s6, "0000", "0000", s7);
38
39  process(SEL, c0, c1)
40  begin
41     if SEL = "010" then
42        COUT <= c0;
43     elsif SEL = "110" then
44        COUT <= c1;
45     end if;
46  end process;
47
48  process(s7, A, B)
49  begin
50     if s7 = "0000" then
51        ZERO <= '1';
52     else
53        ZERO <= '0';
54     end if;
55
56     if ((A(3) = B(3)) and (s7(3) /= B(3))) then
```

```
57          OVERFLOW <= '1';
58      else
59          OVERFLOW <= '0';
60      end if;
61  end process;
62
63  F <= s7;
64  end logic;
```

Figure 12: ALU4.vhdl

Table 6: ALU4 Timing Delay

Progagation Delay

	Input Port	Output Port	RR	RF	FR	FF
1	CIN	ZERO	15.857	15.857	15.857	15.857
2	SEL[2]	ZERO	15.528	15.528	15.528	15.528
3	A[1]	ZERO	15.207	15.207	15.207	15.207
4	B[1]	ZERO	14.971	14.971	14.971	14.971
5	CIN	OVERFLOW	14.434	14.434	14.434	14.434
6	CIN	F[3]	14.119	14.119	14.119	14.119
7	SEL[2]	OVERFLOW	14.105	14.105	14.105	14.105
8	A[2]	ZERO	14.054	14.054	14.054	14.054
9	SEL[2]	F[1]	13.844	13.844	13.844	13.844
10	SEL[2]	F[3]	13.790	13.790	13.790	13.790
11	A[1]	OVERFLOW	13.784	13.784	13.784	13.784
12	B[2]	ZERO	13.753	13.753	13.753	13.753
13	B[1]	OVERFLOW	13.548	13.548	13.548	13.548
14	CIN	F[1]	13.505	13.505	13.505	13.505
15	A[1]	F[3]	13.469	13.469	13.469	13.469
16	SEL[1]	ZERO	13.349	13.349	13.349	13.349
17	B[1]	F[3]	13.233	13.233	13.233	13.233
18	B[3]	ZERO	13.212	13.212	13.212	13.212
19	A[3]	ZERO	13.107	13.107	13.107	13.107
20	A[1]	F[1]	12.887	12.887	12.887	12.887
21	CIN	F[2]	12.719	12.719	12.719	12.719
22	B[1]	F[1]	12.641	12.641	12.641	12.641
23	A[2]	OVERFLOW	12.631	12.631	12.631	12.631
24	SEL[1]	F[1]	12.475	12.475	12.475	12.475

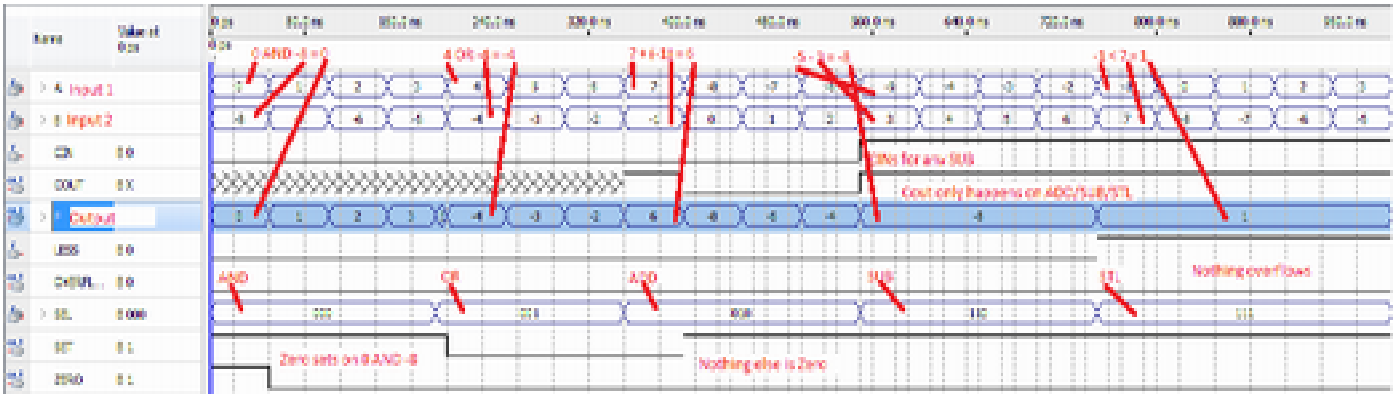

Figure 13: ALU4 Simulation

```
library ieee;
use ieee.std_logic_1164.all;
use work.lib.all;

entity ALU16 is
    port( A, B : in std_logic_vector(15 downto 0);
          SEL : in std_logic_vector(2 downto 0);
          F : out std_logic_vector(15 downto 0);
          COUT, OVERFLOW, ZERO : out std_logic
          );
end ALU16;

architecture logic of ALU16 is
signal tmp : std_logic_vector(15 downto 0);
signal C, G, Z : std_logic_vector(2 downto 0);
signal Z : std_logic_vector(3 downto 0);
signal L : std_logic := '0';
begin

cALU1: ALU4 port map(A(3 downto 0), B(3 downto 0), L, SEL(2), SEL, F(3 downto 0), C(0), G(0), Z(0), Z(0));
cALU2: ALU4 port map(A(7 downto 4), B(7 downto 4), '0', C(0), SEL, F(7 downto 4), C(1), G(1), Z(1), Z(1));
cALU3: ALU4 port map(A(11 downto 8), B(11 downto 8), '0', C(1), SEL, F(11 downto 8), C(2), G(2), Z(2), Z(2));
cALU4: ALU4 port map(A(15 downto 12), B(15 downto 12), '0', C(2), SEL, F(15 downto 12), COUT, OVERFLOW, L, Z(3));

zero <= (Z(0) and Z(1) and Z(2) and Z(3));

end logic;
```

Figure 14: ALU16.vhdl

Table 7: ALU16 Timing Delay

Progagation Delay						
	Input Port	Output Port	RR	RF	FR	FF
1	B[0]	ZERO	15.990	15.990	15.990	15.990
2	A[0]	ZERO	15.811	15.811	15.811	15.811
3	A[1]	ZERO	15.446	15.446	15.446	15.446
4	B[8]	ZERO	15.402	15.402	15.402	15.402
5	B[1]	ZERO	15.382	15.382	15.382	15.382
6	A[8]	ZERO	15.286	15.286	15.286	15.286
7	B[8]	F[11]	14.987	14.987	14.987	14.987
8	B[4]	ZERO	14.976	14.976	14.976	14.976
9	A[8]	F[11]	14.832	14.832	14.832	14.832
10	A[12]	ZERO	14.756	14.756	14.756	14.756
11	B[2]	ZERO	14.716	14.716	14.716	14.716
12	A[12]	OVERFLOW	14.683	14.683	14.683	14.683
13	A[2]	ZERO	14.637	14.637	14.637	14.637
14	A[4]	ZERO	14.579	14.579	14.579	14.579
15	B[12]	ZERO	14.479	14.479	14.479	14.479
16	B[12]	OVERFLOW	14.406	14.406	14.406	14.406
17	SEL[1]	ZERO	14.392	14.392	14.392	14.392
18	A[9]	F[11]	14.153	14.153	14.153	14.153
19	A[9]	ZERO	14.062	14.062	14.062	14.062
20	A[6]	ZERO	14.056	14.056	14.056	14.056
21	B[5]	ZERO	14.055	14.055	14.055	14.055
22	B[9]	F[11]	13.990	13.990	13.990	13.990
23	A[5]	ZERO	13.963	13.963	13.963	13.963
24	B[8]	F[10]	13.955	13.955	13.955	13.955
25	B[0]	F[3]	13.954	13.954	13.954	13.954
26	B[9]	ZERO	13.899	13.899	13.899	13.899

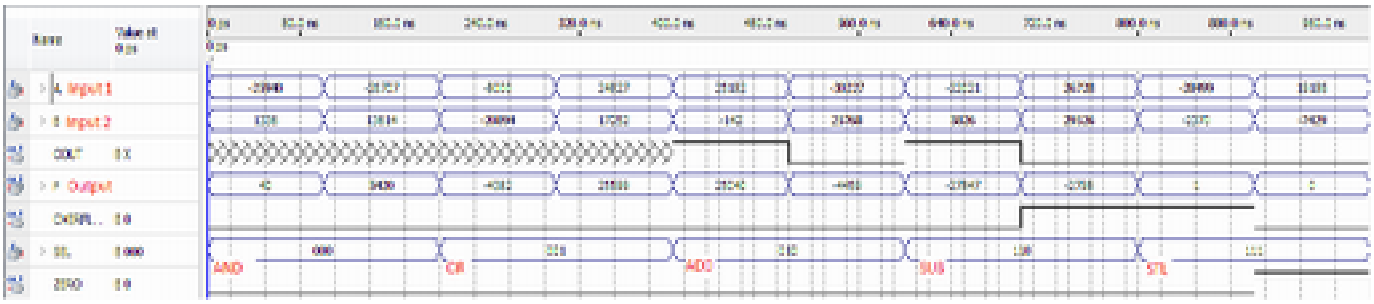

Figure 15: ALU16 Simulation