Lab 2: Design of Data Paths and Control Units Using VHDL

University of Guelph
ENGG*3050 Winter 2016
Professor Shawki Areibi

Graham Thoms 0795111
Vithursan Thangarasa 0821795

1 Introduction

This lab required the use of VHDL to be used as tool for design entry and design the following circuits:

- 4-Bit Algorithmic Logic Unit (ALU)
- 4 x 4 Binary Multiplier using Repeated Addition

The circuits were then tested and debugged by mapping on to the NEXYS3 Spartan-6 FPGA board, and performing simulations using Xilinx ISE Simulator.

2 Summary of Circuit Designs

This section outlines the preliminary work, implementation approaches, and simulation results of the respective circuits.

2.1 4-Bit Algorithmic Logic Unit

2.1.1 Truth Table for Specified ALU

The table presented in **Table 1** is the truth table for the required Algorithmic Logic Unit (ALU). The block diagram for the 1-bit ALU is depicted in **Figure 1**.

Table 1 Shows the truth table of the specified ALU.

Cin	S1	S0	F	X	Y
0	0	0	**A + B**	**A**	**B**
0	0	1	**(not A) + B**	**not A**	**B**
0	1	0	**A - 1**	**A**	**1**
0	1	1	**not A**	**Not A**	**0**
1	0	0	**A - B**	**A**	**not B**
1	0	1	**B - A**	**Not A**	**B**
1	1	0	**A + 1**	**A**	**0**
1	1	1	**(not A) + 1**	**Not A**	**0**

2.1.2 1-Bit Algorithmic Logic Unit

Inputs A and B were sent through a 8 to 1 bit multiplexers and into a 1 bit full adder in order to have various arithmetic operations shown in **Table 1**. These logic operations were synthesized into **Figure 1** to give the specified 1-bit Arithmetic Logic Unit. VHDL Implementation can be found in **Appendix A**.

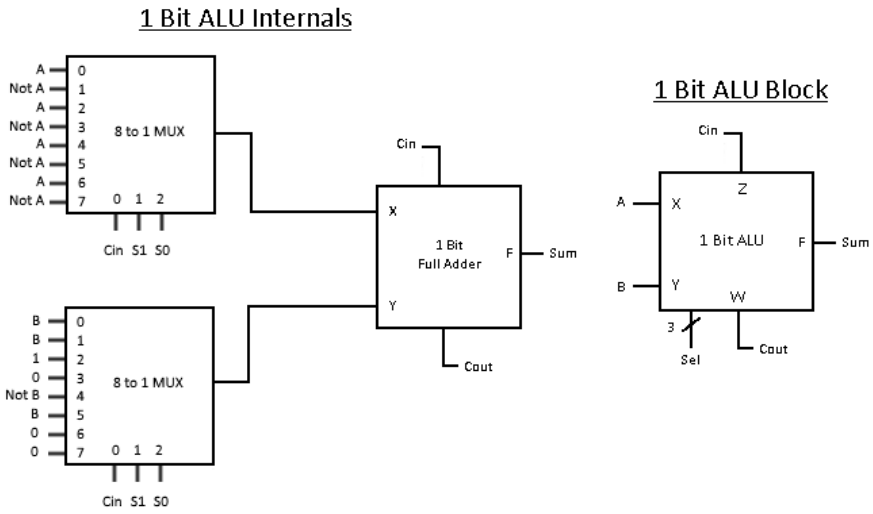

Figure 1 Shows the implementation of the 1-bit ALU

In **Figure 2** sample inputs a = 1 and b = 0 were used to run through each arithmetic operation using the select lines sel[2:0] = {S1, S0, Cin}. For example sel = “110” is the operation 1s complement of A, thus the result is 0, which is shown in the above figure. VHDL Implementation of the test bench can be found in **Appendix B**.

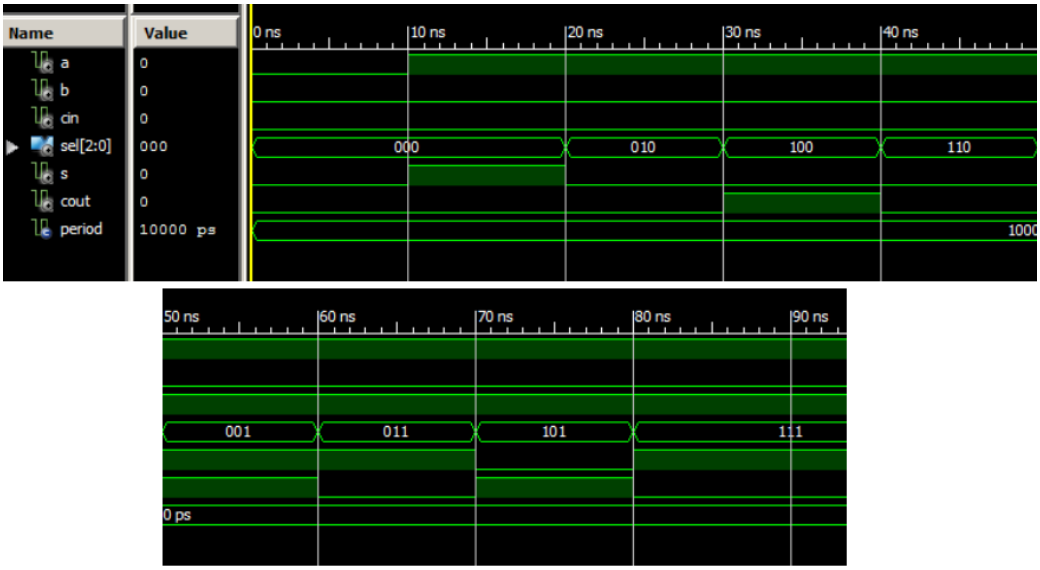

Figure 2 Shows the logic waveform of the 1 bit ALU.

2.1.3 Implementation Approach 4-Bit ALU

The 1-Bit ALU was repeated 4 times to create the 4-Bit ALU as shown in **Figure 3**. It comprises of 4 1-bit ALUs that are cascaded into the next bit and produce the logic operations shown in **Table 1** for two 4 bit numbers. VHDL implementation can be found in **Appendix C**. The 4-bit ALU outputted a binary coded decimal that was imported to a 7-Segment display, as well as the LEDs.

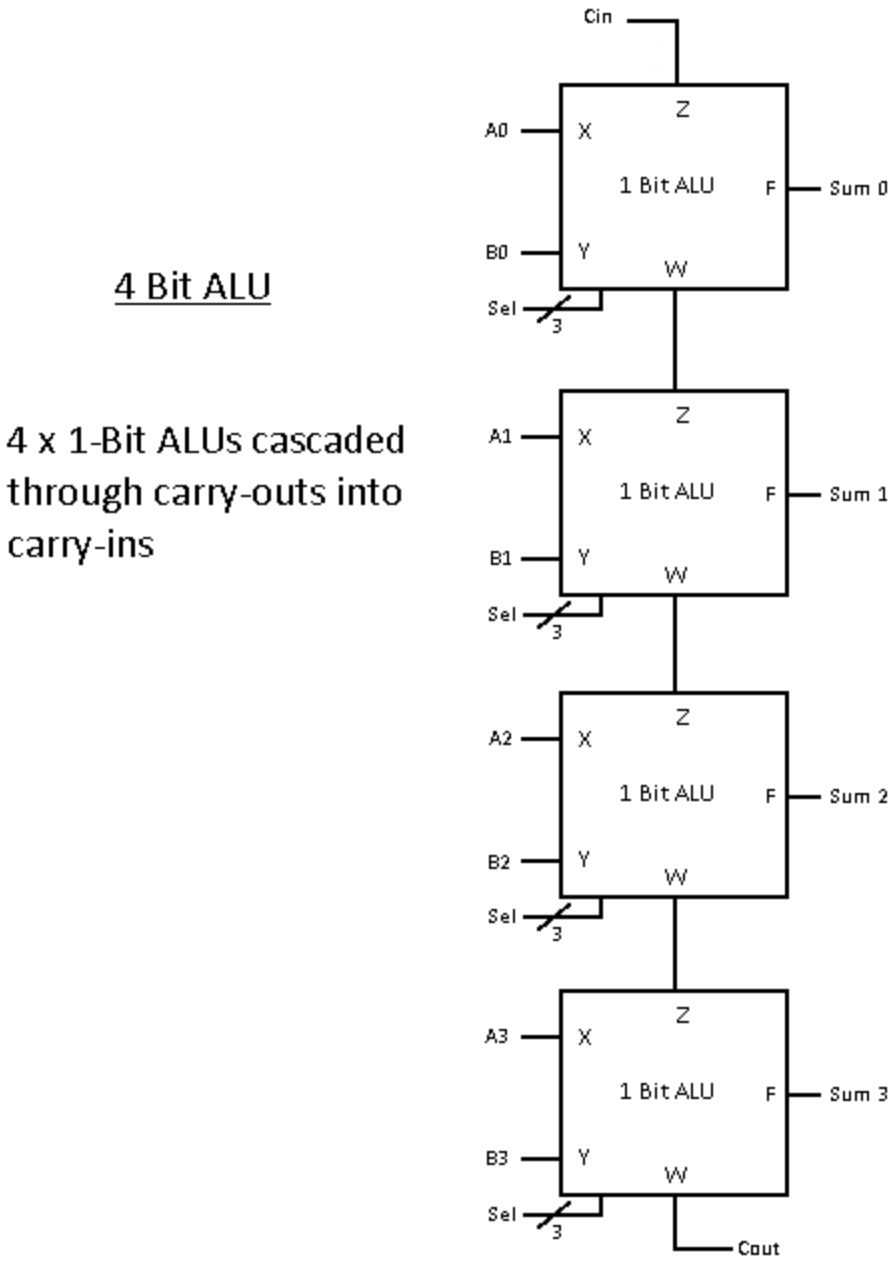

Figure 3 Block diagram of the 4-Bit ALU.

2.1.4 Testbench of 4-Bit ALU

In **Figure 4,** sample inputs a = 6 and b = 2 were used to run through each arithmetic operation using the select lines sel[2:0] = {S1, S0, Cin}. For example sel = “001” is the operation a - b, thus in this case 6 - 2 = 4 (100), which is shown in the above figure. VHDL Implementation can be found in **Appendix D**.

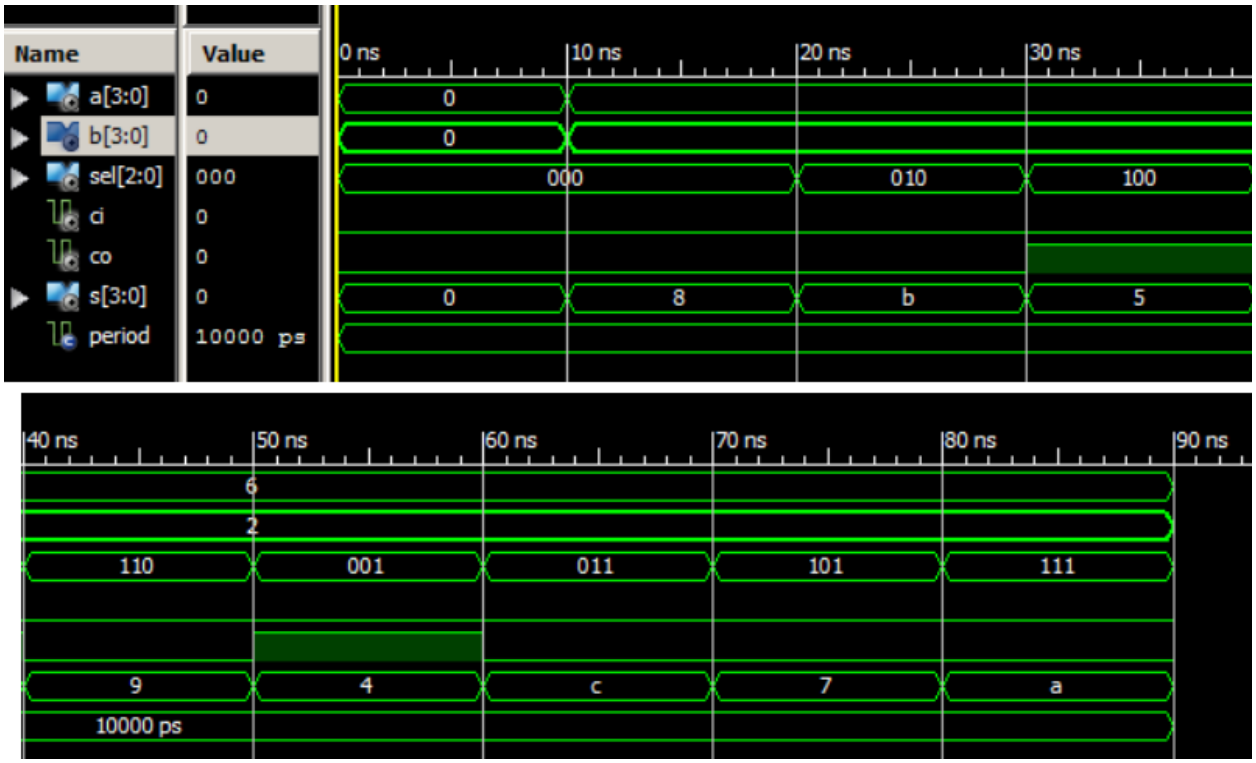

Figure 4 Shows the logic waveform of the specified 4-bit ALU.

2.3 4-Bit Multiplication Unit

2.3.1 Implementation Approach 4 x 4 Multiplier

The ASM chart for the 4 x 4 Multiplier is presented in **Figure 6**. The datapath circuit for the multiplier is presented in **Figure 7**. The ASM chart for the multiplier control unit is presented in **Figure 8**. The multiplicand is 4-bit wide (BR) and the multiplier (AR) is 4-bit wide, thus the product will need an 8-bit register (double width register). The multiplier is placed in the least significant half of the product register (PR), and the most significant half of the PR is cleared. Since there are 4 bits in a single width register, the following steps were performed 4 times:

1. If the least significant bit (LSB) of the product register (PR) contains a 1, the multiplicand is added to the most significant half
2. The whole register is shifted one bit to the right, thus discard the least significant bit. This also shifts the carry bit into the most significant bit.

The VHDL code for the 4 x 4 Multiplier using the Adder and Shifter is presented in **Appendix E**.

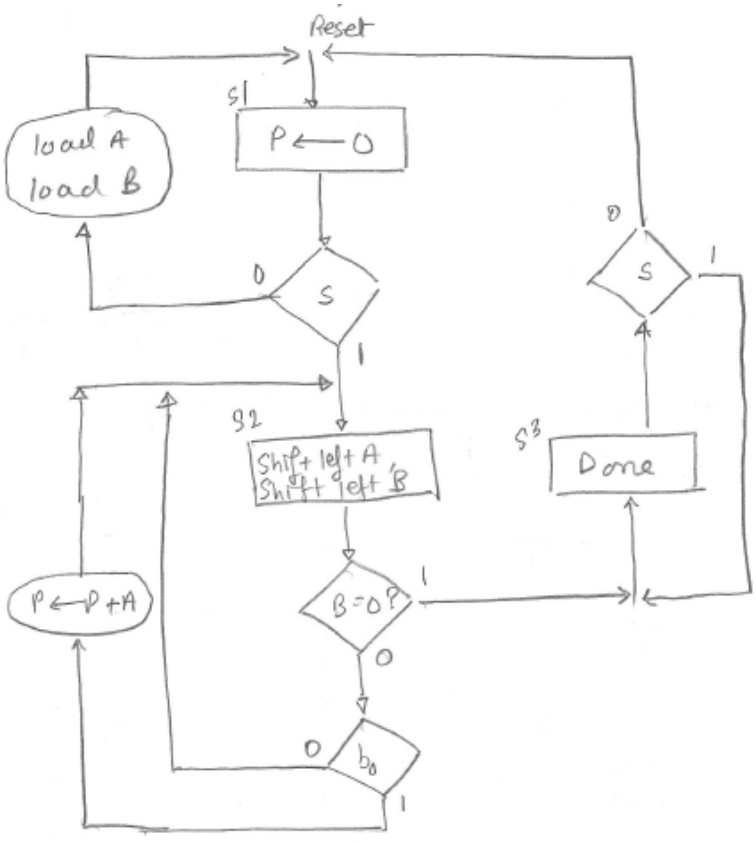

Figure 6 An ASM chart for the multiplier

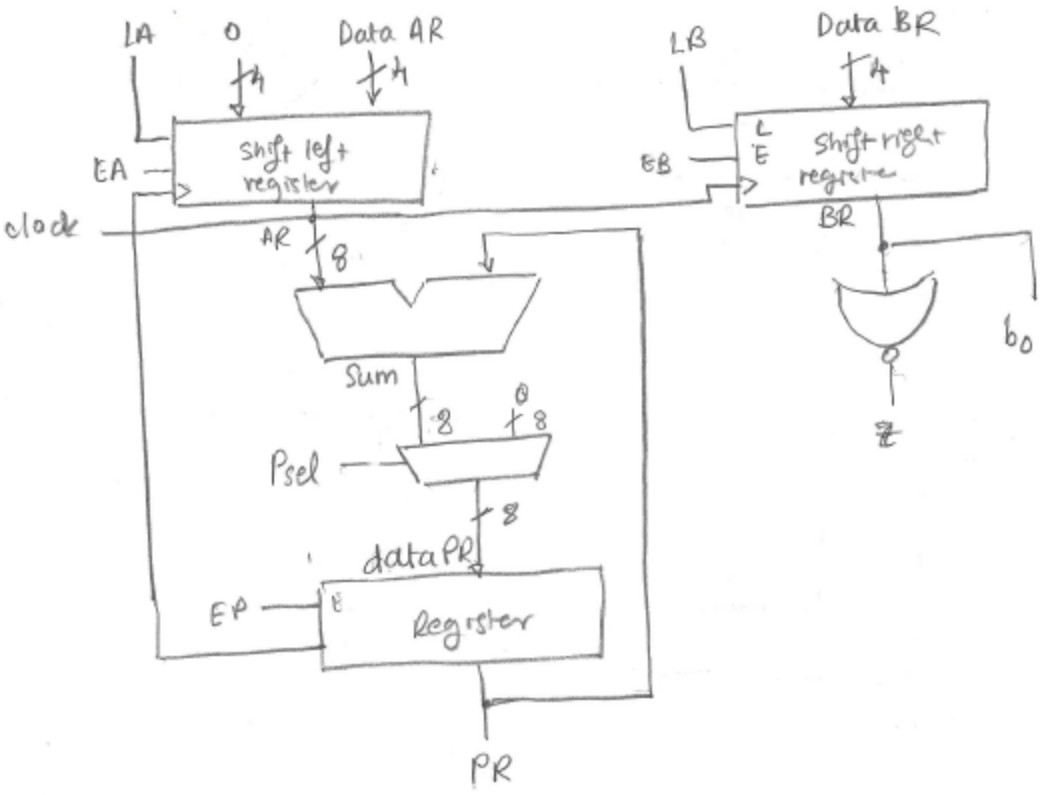

Figure 7 Datapath circuit for the multiplier.

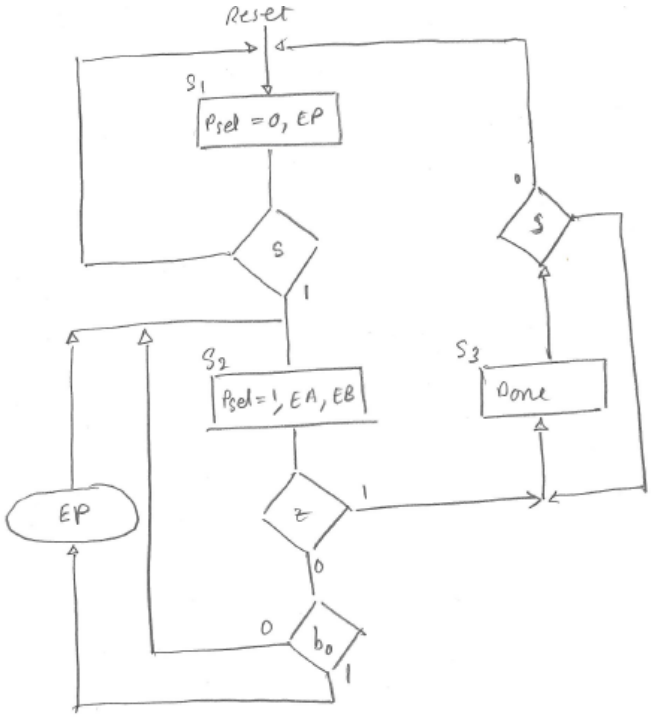

Figure 8 An ASM chart for the multiplier control unit.

2.3.2 Testbench for 4-Bit Multiplication Unit

The testbench simulation result for the 4-bit multiplication unit presented in a wave diagram shown in **Figure 9**. The start button is set to HIGH to activate the control unit. The input A vector is “1111” and the input B vector is “1111”, which resulted in an output of “11100001”. This output is correct as shown in **Figure 10**. For the second test, the input vector A is “1001” and the input vector B is “1111”, which resulted in an output of “01010001”. The output is also correct as shown in **Figure 11**. The test bench code is provided in **Appendix F**.

Figure 9 Simulation waveform for the 4 x 4 Multiplier.

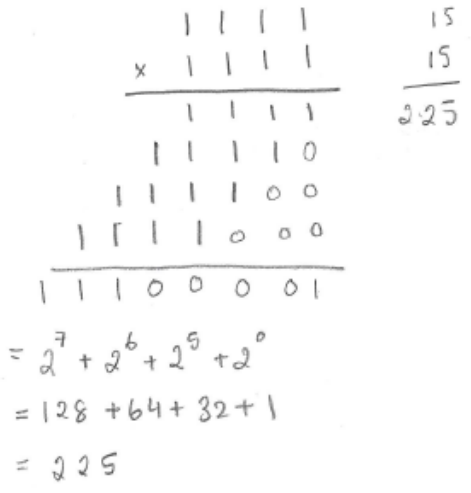

Figure 10 Expected Output where input A = “1111” and B = “1111”. The output is “11100001”.

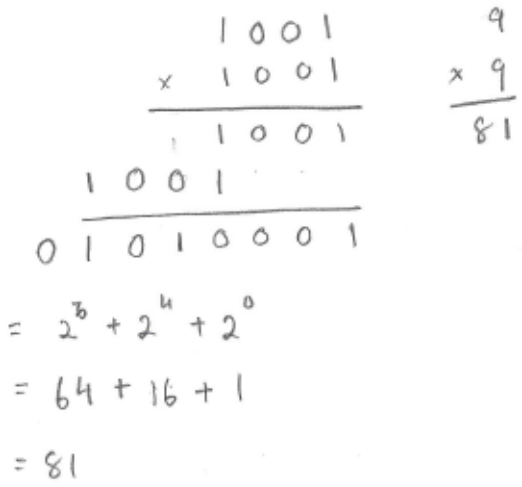

Figure 11 Expected output where input A = “1001” and B = “1001”. The output is “01010001”.

3 Conclusion and Future Considerations

In the final analysis, this lab helped develop a stronger understanding of the internal components in an ALU and Multiplier, create ASM charts, implementing state machines in VHDL, and using Xilinx ISE Design to map and simulate the circuits. This lab served as a beneficial tool with getting familiar with fundamental VHDL concepts. In the future, it may help to review VHDL programming concepts and styles, especially implementing registers, state processes, components, and etc. to give some background before beginning the lab.

Appendix A


```
1  ----------------------------------------------------------------------------------
2  -- Company:
3  -- Engineer: Graham Thoms
4  --
5  -- Create Date:    14:36:12 01/31/2016
6  -- Design Name:
7  -- Module Name:    alu_VHDL - Behavioral
8  -- Project Name:
9  -- Target Devices:
10 -- Tool versions:
11 -- Description:
12 --
13 -- Dependencies:
14 --
15 -- Revision:
16 -- Revision 0.01 - File Created
17 -- Additional Comments:
18 --
19 ----------------------------------------------------------------------------------
20 library IEEE;
21 use IEEE.STD_LOGIC_1164.ALL;
22
23 -- Uncomment the following library declaration if using
24 -- arithmetic functions with Signed or Unsigned values
25 --use IEEE.NUMERIC_STD.ALL;
26
27 -- Uncomment the following library declaration if instantiating
28 -- any Xilinx primitives in this code.
29 --library UNISIM;
30 --use UNISIM.VComponents.all;
31
32
33
34 ----------------------------------------------------------------------
35 --                          1 bit ALU
36 ----------------------------------------------------------------------
37
38
39 entity alu_VHDL_1bit is
40     Port ( A : in  STD_LOGIC;
41            B : in  STD_LOGIC;
42            S : out  STD_LOGIC;
43            Cin : in  STD_LOGIC;
44            Cout : out  STD_LOGIC;
45            SEL : in STD_LOGIC_VECTOR (2 downto 0));
46 end alu_VHDL_1bit;
47
48 architecture Behavioral of alu_VHDL_1bit is
49
50    component FullAdder is
51        Port ( x : in  STD_LOGIC;
52               y : in  STD_LOGIC;
53               g : out  STD_LOGIC;
54               ci : in  STD_LOGIC;
55               co : out  STD_LOGIC);
56    end component;
57
```

```
58      component Mux8to1 is
59          Port ( b0 : in  STD_LOGIC;
60                 b1 : in  STD_LOGIC;
61                 b2 : in  STD_LOGIC;
62                 b3 : in  STD_LOGIC;
63                 b4 : in  STD_LOGIC;
64                 b5 : in  STD_LOGIC;
65                 b6 : in  STD_LOGIC;
66                 b7 : in  STD_LOGIC;
67                 sel: in STD_LOGIC_VECTOR(2 downto 0);
68                 dout : out  STD_LOGIC);
69      end component;
70
71      signal S_s, Co_s, dout_s_a, dout_s_b : STD_LOGIC;
72
73      begin
74
75      -- connecting MUXs for inputs a and b to fulladder
76
77      A0: Mux8to1 port map (A, A,not A,not A,A, A, not A, not A, Sel, dout_s_a);
78      B0: Mux8to1 port map (B, not B,B,B,'1', '0', '0', '0', Sel, dout_s_b);
79      F0: FullAdder port map (dout_s_a, dout_s_b, S_s, Cin, Co_s);
80
81      -- outputs of 1 bit ALU
82
83      S <= S_s;
84      Cout <= Co_s;
85
86  end Behavioral;
87
88  ----------------------------------------------------------------------
89  --                          END 1 bit ALU
90  ----------------------------------------------------------------------
91
92  ----------------------------------------------------------------------
93  --                          FULL ADDER
94  ----------------------------------------------------------------------
95
96  library IEEE;
97  use IEEE.STD_LOGIC_1164.ALL;
98
99  entity FullAdder is
100     Port ( x : in  STD_LOGIC;                -- input x
101            y : in  STD_LOGIC;                -- input y
102            g : out  STD_LOGIC;               -- output g = x + y + ci
103            ci : in  STD_LOGIC;               -- input ci (carry in)
104            co : out  STD_LOGIC);             -- output co (carry out)
105 end FullAdder;
106
107 architecture dataflow of FullAdder is
108    begin
109
110    -- fulladder logic
111
112    g <= ((x xor y) xor ci);                  -- sum
113    co <= ((x and y) or ((x xor y) and ci));  -- carry out
114
```

```
115   end dataflow;
116
117   ----------------------------------------------------------------------
118   --                          END FULL ADDER
119   ----------------------------------------------------------------------
120
121   ----------------------------------------------------------------------
122   --                          8 to 1 MUX
123   ----------------------------------------------------------------------
124
125   library IEEE;
126   use IEEE.STD_LOGIC_1164.ALL;
127
128   entity Mux8to1 is
129       Port ( b0 : in  STD_LOGIC;                      -- MUX inputs
130              b1 : in  STD_LOGIC;
131              b2 : in  STD_LOGIC;
132              b3 : in  STD_LOGIC;
133              b4 : in  STD_LOGIC;
134              b5 : in  STD_LOGIC;
135              b6 : in  STD_LOGIC;
136              b7 : in  STD_LOGIC;
137              sel: in STD_LOGIC_VECTOR(2 downto 0);   -- select lines
138              dout : out  STD_LOGIC);                 -- output
139   end Mux8to1;
140
141   architecture Behavioral of Mux8to1 is
142      begin
143
144      -- behaviour of 8 to 1 MUX
145
146      dout <=     b0 when (sel="000")
147                else
148                   b1 when (sel="001")
149                else
150                   b2 when (sel="010")
151                else
152                   b3 when (sel="011")
153                else
154                   b4 when (sel="100")
155                else
156                   b5 when (sel="101")
157                else
158                   b6 when (sel="110")
159                else
160                   b7;
161   end Behavioral;
162
163   ----------------------------------------------------------------------
164   --                          END 8 to 1 MUX
165   ----------------------------------------------------------------------
166
167
168
169
170
```

Appendix B


```
1   --------------------------------------------------------------------------------
2   -- Company:
3   -- Engineer:
4   --
5   -- Create Date:   23:45:36 02/01/2016
6   -- Design Name:
7   -- Module Name:   T:/ENGG3050/ALU_1bit/ALU_1bit_TB.vhd
8   -- Project Name:  ALU_1bit
9   -- Target Device:
10  -- Tool versions:
11  -- Description:
12  --
13  -- VHDL Test Bench Created by ISE for module: alu_VHDL_1bit
14  --
15  -- Dependencies:
16  --
17  -- Revision:
18  -- Revision 0.01 - File Created
19  -- Additional Comments:
20  --
21  -- Notes:
22  -- This testbench has been automatically generated using types std_logic and
23  -- std_logic_vector for the ports of the unit under test.  Xilinx recommends
24  -- that these types always be used for the top-level I/O of a design in order
25  -- to guarantee that the testbench will bind correctly to the post-implementation
26  -- simulation model.
27  --------------------------------------------------------------------------------
28  LIBRARY ieee;
29  USE ieee.std_logic_1164.ALL;
30
31  -- Uncomment the following library declaration if using
32  -- arithmetic functions with Signed or Unsigned values
33  --USE ieee.numeric_std.ALL;
34
35  ENTITY ALU_1bit_TB IS
36  END ALU_1bit_TB;
37
38  ARCHITECTURE behavior OF ALU_1bit_TB IS
39
40      -- Component Declaration for the Unit Under Test (UUT)
41
42      COMPONENT alu_VHDL_1bit
43      PORT(
44           A : IN  std_logic;
45           B : IN  std_logic;
46           S : OUT  std_logic;
47           Cin : IN  std_logic;
48           Cout : OUT  std_logic;
49           SEL : IN  std_logic_vector(2 downto 0)
50          );
51      END COMPONENT;
52
53
54     --Inputs
55     signal A : std_logic := '0';
56     signal B : std_logic := '0';
57     signal Cin : std_logic := '0';
```

```
58      signal SEL : std_logic_vector(2 downto 0) := (others => '0');
59
60      --Outputs
61      signal S : std_logic;
62      signal Cout : std_logic;
63      -- No clocks detected in port list. Replace <clock> below with
64      -- appropriate port name
65
66      constant period : time := 10 ns;
67
68  BEGIN
69
70      -- Instantiate the Unit Under Test (UUT)
71      uut: alu_VHDL_1bit PORT MAP (
72             A => A,
73             B => B,
74             S => S,
75             Cin => Cin,
76             Cout => Cout,
77             SEL => SEL
78           );
79
80
81
82      -- Stimulus process
83      stim_proc: process
84      begin
85         wait for period;
86
87         A <= '1'; -- a = 1 (0110)
88         B <= '0'; -- b = 2 (0010)
89         Cin <= '0';
90
91         -- SEL(0) = Cin
92
93         -- SEL(0)|SEL(2)|SEL(1) | F
94         --------------------------------
95         --   0      0      0    | a + b
96         --------------------------------
97         --   0      0      1    | (not a) + b
98         --------------------------------
99         --   0      1      0    | a - 1
100        --------------------------------
101        --   0      1      1    | not a
102        --------------------------------
103        --   1      0      0    | a - b
104        --------------------------------
105        --   1      0      1    | b - a
106        --------------------------------
107        --   1      1      0    | a + 1
108        --------------------------------
109        --   1      1      1    | (not a) + b
110        --------------------------------
111
112        -- operations 0 - 3
113
114        SEL(0) <= '0'; --OPERATION: a + b
```

```
115         SEL(2) <= '0';
116         SEL(1) <= '0'; -- RESULT = 1
117
118         wait for period;
119
120         SEL(0) <= '0'; --OPERATION: (not a) + b
121         SEL(2) <= '0';
122         SEL(1) <= '1'; -- RESULT = 0
123
124         wait for period;
125
126         SEL(0) <= '0'; --OPERATION: a - 1
127         SEL(2) <= '1';
128         SEL(1) <= '0'; -- RESULT = 0
129
130         wait for period;
131
132         SEL(0) <= '0'; --OPERATION: not a , 1s complement
133         SEL(2) <= '1';
134         SEL(1) <= '1'; -- RESULT = 0
135
136         wait for period;
137
138         -- operations 4 - 7
139
140         SEL(0) <= '1'; --OPERATION: a - b
141         SEL(2) <= '0';
142         SEL(1) <= '0'; -- RESULT = 1
143         Cin <= '1';
144
145         wait for period;
146
147         SEL(0) <= '1'; --OPERATION: b - a
148         SEL(2) <= '0';
149         SEL(1) <= '1'; -- RESULT = 0
150         Cin <= '1';
151
152         wait for period;
153
154         SEL(0) <= '1'; --OPERATION: a + 1
155         SEL(2) <= '1';
156         SEL(1) <= '0'; -- RESULT = 0
157         Cin <= '1';
158
159         wait for period;
160
161         SEL(0) <= '1'; --OPERATION: not a + 1 , 2s complement
162         SEL(2) <= '1';
163         SEL(1) <= '1'; -- RESULT = 1
164         Cin <= '1';
165
166         wait for period;
167
168
169         wait for period;
170
171         wait;
```



```
172       end process;
173
174   END;
175
```

Appendix C


```
1  ----------------------------------------------------------------------------------
2  -- Company:
3  -- Engineer: Graham Thoms
4  --
5  -- Create Date:    14:36:12 01/31/2016
6  -- Design Name:
7  -- Module Name:    alu_VHDL - Behavioral
8  -- Project Name:
9  -- Target Devices:
10 -- Tool versions:
11 -- Description:
12 --
13 -- Dependencies:
14 --
15 -- Revision:
16 -- Revision 0.01 - File Created
17 -- Additional Comments:
18 --
19 ----------------------------------------------------------------------------------
20 library IEEE;
21 use IEEE.STD_LOGIC_1164.ALL;
22
23 -- Uncomment the following library declaration if using
24 -- arithmetic functions with Signed or Unsigned values
25 --use IEEE.NUMERIC_STD.ALL;
26
27 -- Uncomment the following library declaration if instantiating
28 -- any Xilinx primitives in this code.
29 --library UNISIM;
30 --use UNISIM.VComponents.all;
31
32 ----------------------------------------------------------------------
33 --                          4 bit ALU
34 ----------------------------------------------------------------------
35
36 entity alu_VHDL_4bit is                                -- entity for 4 bit ALU
37    Port ( a : in STD_LOGIC_VECTOR(3 downto 0);      -- input a
38           b : in STD_LOGIC_VECTOR(3 downto 0);      -- input b
39           SEL:in STD_LOGIC_VECTOR(2 downto 0);      -- select lines
40           ci : in STD_LOGIC;                        -- carry in
41           co : out STD_LOGIC;                       -- carry out
42           S : out STD_LOGIC_VECTOR(3 downto 0);     -- binary output
43           SS : out STD_LOGIC_VECTOR(6 downto 0);    -- 7 segment display output
44           Aout: out STD_LOGIC_VECTOR(3 downto 0);   -- A output for leds
45           Bout: out STD_LOGIC_VECTOR(3 downto 0);   -- B output for leds
46           POS : out STD_LOGIC_VECTOR(3 downto 0)); -- display digit select
47 end alu_VHDL_4bit;
48
49 architecture Behavioral of alu_VHDL_4bit is
50
51    component alu_VHDL_1bit is
52     Port ( A : in  STD_LOGIC;
53            B : in  STD_LOGIC;
54            S : out  STD_LOGIC;
55            Cin : in  STD_LOGIC;
56            Cout : out  STD_LOGIC;
57            sel : in STD_LOGIC_VECTOR (2 downto 0));
```

```
58      end component;
59
60      component SevenSeg is
61         Port (    BCD : in STD_LOGIC_VECTOR(3 downto 0); -- Binary coded decimal input
62                   SevenSegDisplay : out STD_LOGIC_VECTOR(6 downto 0));
63      end component;
64
65
66      signal S_one, S_two, S_three, S_four: std_logic;
67      signal C_one, C_two, C_three, C_four: std_logic;
68      signal gout : STD_LOGIC_VECTOR(3 downto 0);
69
70      begin
71
72      -- 4 x 1 bit ALUs cascaded through the carry-outs to the carry-ins
73
74      G0: alu_VHDL_1bit port map (a(0), b(0), S_one, SEL(0), C_one, SEL);
75      G1: alu_VHDL_1bit port map (a(1), b(1), S_two, C_one, C_two, SEL);
76      G2: alu_VHDL_1bit port map (a(2), b(2), S_three, C_two,C_three, SEL);
77      G3: alu_VHDL_1bit port map (a(3), b(3), S_four, C_three,C_four, SEL);
78
79      -- Operands to LEDs
80
81      Aout(0) <= a(0);
82      Aout(1) <= a(1);
83      Aout(2) <= a(2);
84      Aout(3) <= a(3);
85      Bout(0) <= b(0);
86      Bout(1) <= b(1);
87      Bout(2) <= b(2);
88      Bout(3) <= b(3);
89
90      -- 4 bit sum
91
92      gout(0) <= S_one;
93      gout(1) <= S_two;
94      gout(2) <= S_three;
95      gout(3) <= S_four;
96      co <= C_four;
97
98      S(0) <= S_one;
99      S(1) <= S_two;
100     S(2) <= S_three;
101     S(3) <= S_four;
102
103     -- seven segment display outputting 4 bit sum
104
105     POS <= "1110";
106
107     U1: SevenSeg port map (gout,SS);
108
109  end Behavioral;
110
111  ----------------------------------------------------------------------
112  --                          END 4 bit ALU
113  ----------------------------------------------------------------------
114
```

```
115  ----------------------------------------------------------------------
116  --                          1 bit ALU
117  ----------------------------------------------------------------------
118  library IEEE;
119  use IEEE.STD_LOGIC_1164.ALL;
120
121
122  entity alu_VHDL_1bit is
123      Port ( A : in  STD_LOGIC;
124             B : in  STD_LOGIC;
125             S : out  STD_LOGIC;
126             Cin : in  STD_LOGIC;
127             Cout : out  STD_LOGIC;
128             SEL : in STD_LOGIC_VECTOR (2 downto 0));
129  end alu_VHDL_1bit;
130
131  architecture Behavioral of alu_VHDL_1bit is
132
133     component FullAdder is
134         Port ( x : in  STD_LOGIC;
135                y : in  STD_LOGIC;
136                g : out  STD_LOGIC;
137                ci : in  STD_LOGIC;
138                co : out  STD_LOGIC);
139     end component;
140
141     component Mux8to1 is
142         Port ( b0 : in  STD_LOGIC;
143                b1 : in  STD_LOGIC;
144                b2 : in  STD_LOGIC;
145                b3 : in  STD_LOGIC;
146                b4 : in  STD_LOGIC;
147                b5 : in  STD_LOGIC;
148                b6 : in  STD_LOGIC;
149                b7 : in  STD_LOGIC;
150                sel: in STD_LOGIC_VECTOR(2 downto 0);
151                dout : out  STD_LOGIC);
152     end component;
153
154     signal S_s, Co_s, dout_s_a, dout_s_b : STD_LOGIC;
155
156     begin
157
158     -- connecting MUXs for inputs a and b to fulladder
159
160     A0: Mux8to1 port map (A, A,not A,not A,A, A, not A, not A, Sel, dout_s_a);
161     B0: Mux8to1 port map (B, not B,B,B,'1', '0', '0', '0', Sel, dout_s_b);
162     F0: FullAdder port map (dout_s_a, dout_s_b, S_s, Cin, Co_s);
163
164     -- outputs of 1 bit ALU
165
166     S <= S_s;
167     Cout <= Co_s;
168
169  end Behavioral;
170
171  ----------------------------------------------------------------------
```

```
172  --                          END 1 bit ALU
173  ----------------------------------------------------------------------
174
175  ----------------------------------------------------------------------
176  --                          FULL ADDER
177  ----------------------------------------------------------------------
178
179  library IEEE;
180  use IEEE.STD_LOGIC_1164.ALL;
181
182  entity FullAdder is
183      Port ( x : in  STD_LOGIC;                -- input x
184             y : in  STD_LOGIC;                -- input y
185             g : out  STD_LOGIC;               -- output g = x + y + ci
186             ci : in  STD_LOGIC;               -- input ci (carry in)
187             co : out  STD_LOGIC);             -- output co (carry out)
188  end FullAdder;
189
190  architecture dataflow of FullAdder is
191     begin
192
193     -- fulladder logic
194
195     g <= ((x xor y) xor ci);                  -- sum
196     co <= ((x and y) or ((x xor y) and ci));  -- carry out
197
198  end dataflow;
199
200  ----------------------------------------------------------------------
201  --                          END FULL ADDER
202  ----------------------------------------------------------------------
203
204  ----------------------------------------------------------------------
205  --                          8 to 1 MUX
206  ----------------------------------------------------------------------
207
208  library IEEE;
209  use IEEE.STD_LOGIC_1164.ALL;
210
211  entity Mux8to1 is
212      Port ( b0 : in  STD_LOGIC;                    -- MUX inputs
213             b1 : in  STD_LOGIC;
214             b2 : in  STD_LOGIC;
215             b3 : in  STD_LOGIC;
216             b4 : in  STD_LOGIC;
217             b5 : in  STD_LOGIC;
218             b6 : in  STD_LOGIC;
219             b7 : in  STD_LOGIC;
220             sel: in STD_LOGIC_VECTOR(2 downto 0);   -- select lines
221             dout : out  STD_LOGIC);               -- output
222  end Mux8to1;
223
224  architecture Behavioral of Mux8to1 is
225     begin
226
227     -- behaviour of 8 to 1 MUX
228
```

```
229       dout <=     b0 when (sel="000")
230                 else
231                    b1 when (sel="001")
232                 else
233                    b2 when (sel="010")
234                 else
235                    b3 when (sel="011")
236                 else
237                    b4 when (sel="100")
238                 else
239                    b5 when (sel="101")
240                 else
241                    b6 when (sel="110")
242                 else
243                    b7;
244    end Behavioral;
245
246    ----------------------------------------------------------------------
247    --                           END 8 to 1 MUX
248    ----------------------------------------------------------------------
249
250    ----------------------------------------------------------------------
251    --                           7 SEG DISPLAY
252    ----------------------------------------------------------------------
253
254    library IEEE;
255    use IEEE.STD_LOGIC_1164.ALL;
256
257    entity SevenSeg is
258       Port (
259          BCD : in STD_LOGIC_VECTOR(3 downto 0);              -- Binary coded decimal input
260          SevenSegDisplay : out STD_LOGIC_VECTOR(6 downto 0) -- 7 seg display output
261       );
262    end SevenSeg;
263
264    architecture Behavioral of SevenSeg is
265
266       begin
267
268       process(BCD)
269       BEGIN
270
271       case BCD is -- input for the LCD Display, allows user to visually see the number
       on a screen
272          when "0000"=> SevenSegDisplay <="0000001"; -- '0'
273          when "0001"=> SevenSegDisplay <="1001111"; -- '1'
274          when "0010"=> SevenSegDisplay <="0010010"; -- '2'
275          when "0011"=> SevenSegDisplay <="0000110"; -- '3'
276          when "0100"=> SevenSegDisplay <="1001100"; -- '4'
277          when "0101"=> SevenSegDisplay <="0100100"; -- '5'
278          when "0110"=> SevenSegDisplay <="0100000"; -- '6'
279          when "0111"=> SevenSegDisplay <="0001111"; -- '7'
280          when "1000"=> SevenSegDisplay <="0000000"; -- '8'
281          when "1001"=> SevenSegDisplay <="0000100"; -- '9'
282          when "1010"=> SevenSegDisplay <="0001000"; -- 'A'
283          when "1011"=> SevenSegDisplay <="1100000"; -- 'b'
284          when "1100"=> SevenSegDisplay <="0110001"; -- 'C'
```

```
285         when "1101"=> SevenSegDisplay <="1000010"; -- 'd'
286         when "1110"=> SevenSegDisplay <="0110000"; -- 'E'
287         when "1111"=> SevenSegDisplay <="0111000"; -- 'F'
288         when others => SevenSegDisplay <="1111111";-- ERROR
289      end case;
290
291   end process;
292
293   end Behavioral;
294
295   ----------------------------------------------------------------------
296   --                          END 7 SEG DISPLAY
297   ----------------------------------------------------------------------
298
299
300
```

Appendix D


```
1   --------------------------------------------------------------------------------
2   -- Company:
3   -- Engineer:
4   --
5   -- Create Date:   22:05:43 02/01/2016
6   -- Design Name:
7   -- Module Name:   H:/ENGG3050/ALU_VHDL/ALU_VHDL_TB.vhd
8   -- Project Name:  ALU_VHDL
9   -- Target Device:
10  -- Tool versions:
11  -- Description:
12  --
13  -- VHDL Test Bench Created by ISE for module: alu_VHDL_4bit
14  --
15  -- Dependencies:
16  --
17  -- Revision:
18  -- Revision 0.01 - File Created
19  -- Additional Comments:
20  --
21  -- Notes:
22  -- This testbench has been automatically generated using types std_logic and
23  -- std_logic_vector for the ports of the unit under test.  Xilinx recommends
24  -- that these types always be used for the top-level I/O of a design in order
25  -- to guarantee that the testbench will bind correctly to the post-implementation
26  -- simulation model.
27  --------------------------------------------------------------------------------
28  LIBRARY ieee;
29  USE ieee.std_logic_1164.ALL;
30
31  -- Uncomment the following library declaration if using
32  -- arithmetic functions with Signed or Unsigned values
33  --USE ieee.numeric_std.ALL;
34
35  ENTITY ALU_VHDL_TB IS
36  END ALU_VHDL_TB;
37
38  ARCHITECTURE behavior OF ALU_VHDL_TB IS
39
40      -- Component Declaration for the Unit Under Test (UUT)
41
42      COMPONENT alu_VHDL_4bit
43      PORT(
44           a : IN  std_logic_vector(3 downto 0);
45           b : IN  std_logic_vector(3 downto 0);
46           SEL : IN  std_logic_vector(2 downto 0);
47           ci : IN  std_logic;
48           co : OUT  std_logic;
49           S : OUT  std_logic_vector(3 downto 0);
50           SS : OUT  std_logic_vector(6 downto 0);
51           Aout : OUT  std_logic_vector(3 downto 0);
52           Bout : OUT  std_logic_vector(3 downto 0);
53           POS : OUT  std_logic_vector(3 downto 0)
54          );
55      END COMPONENT;
56
57
```

```
58      --Inputs
59      signal a : std_logic_vector(3 downto 0) := (others => '0');
60      signal b : std_logic_vector(3 downto 0) := (others => '0');
61      signal SEL : std_logic_vector(2 downto 0) := (others => '0');
62      signal ci : std_logic := '0';
63
64      --Outputs
65      signal co : std_logic;
66      signal S : std_logic_vector(3 downto 0);
67      signal SS : std_logic_vector(6 downto 0);
68      signal Aout : std_logic_vector(3 downto 0);
69      signal Bout : std_logic_vector(3 downto 0);
70      signal POS : std_logic_vector(3 downto 0);
71      -- No clocks detected in port list. Replace <clock> below with
72      -- appropriate port name
73
74      constant period : time := 10 ns;
75
76   BEGIN
77
78      -- Instantiate the Unit Under Test (UUT)
79      uut: alu_VHDL_4bit PORT MAP (
80             a => a,
81             b => b,
82             SEL => SEL,
83             ci => ci,
84             co => co,
85             S => S,
86             SS => SS,
87             Aout => Aout,
88             Bout => Bout,
89             POS => POS
90           );
91
92
93      -- Stimulus process
94      stim_proc: process
95      begin
96         wait for period;
97
98         a(3) <= '0'; -- a = 6 (0110)
99         a(2) <= '1';
100        a(1) <= '1';
101        a(0) <= '0';
102
103        b(3) <= '0'; -- b = 2 (0010)
104        b(2) <= '0';
105        b(1) <= '1';
106        b(0) <= '0';
107
108        -- SEL(0) = Cin
109
110        -- SEL(0)|SEL(2)|SEL(1) | F
111        ------------------------------
112        --   0      0      0    | a + b
113        ------------------------------
114        --   0      0      1    | (not a) + b
```

```
115         --------------------------------
116         --   0      1      0    | a - 1
117         --------------------------------
118         --   0      1      1    | not a
119         --------------------------------
120         --   1      0      0    | a - b
121         --------------------------------
122         --   1      0      1    | b - a
123         --------------------------------
124         --   1      1      0    | a + 1
125         --------------------------------
126         --   1      1      1    | (not a) + b
127         --------------------------------
128
129         -- operations 0 - 3
130
131         SEL(0) <= '0'; --OPERATION: a + b
132         SEL(2) <= '0';
133         SEL(1) <= '0'; -- RESULT = 8
134
135         wait for period;
136
137         SEL(0) <= '0'; --OPERATION: (not a) + b
138         SEL(2) <= '0';
139         SEL(1) <= '1'; -- RESULT = B
140
141         wait for period;
142
143         SEL(0) <= '0'; --OPERATION: a - 1
144         SEL(2) <= '1';
145         SEL(1) <= '0'; -- RESULT = 5
146
147         wait for period;
148
149         SEL(0) <= '0'; --OPERATION: not a , ls complement
150         SEL(2) <= '1';
151         SEL(1) <= '1'; -- RESULT = 9
152
153         wait for period;
154
155
156
157         -- operations 4 - 7
158
159         SEL(0) <= '1'; --OPERATION: a - b
160         SEL(2) <= '0';
161         SEL(1) <= '0'; -- RESULT = 4
162
163         wait for period;
164
165         SEL(0) <= '1'; --OPERATION: b - a
166         SEL(2) <= '0';
167         SEL(1) <= '1'; -- RESULT = C
168
169         wait for period;
170
171         SEL(0) <= '1'; --OPERATION: a + 1
```

```
172         SEL(2) <= '1';
173         SEL(1) <= '0'; -- RESULT = 7
174
175         wait for period;
176
177         SEL(0) <= '1'; --OPERATION: not a + 1 , 2s complement
178         SEL(2) <= '1';
179         SEL(1) <= '1'; -- RESULT = A
180
181         wait for period;
182
183
184         wait;
185      end process;
186
187   END;
188
```

Appendix E


```
1   ----------------------------------------------------------------------------------
2   -- Company:
3   -- Engineer:
4   --
5   -- Create Date:    16:15:32 01/31/2016
6   -- Design Name:
7   -- Module Name:    Multiplier_VHDL - Behavioral
8   -- Project Name:
9   -- Target Devices:
10  -- Tool versions:
11  -- Description:
12  --
13  -- Dependencies:
14  --
15  -- Revision:
16  -- Revision 0.01 - File Created
17  -- Additional Comments:
18  --
19  ----------------------------------------------------------------------------------
20  library IEEE;
21  use IEEE.STD_LOGIC_1164.ALL;
22  use IEEE.NUMERIC_STD.ALL;
23  use IEEE.STD_LOGIC_UNSIGNED.ALL;
24
25  -- Uncomment the following library declaration if using
26  -- arithmetic functions with Signed or Unsigned values
27  --use IEEE.NUMERIC_STD.ALL;
28
29  -- Uncomment the following library declaration if instantiating
30  -- any Xilinx primitives in this code.
31  --library UNISIM;
32  --use UNISIM.VComponents.all;
33
34  entity multiplier is
35
36  port(  a, b : in std_logic_vector(3 downto 0); --input vectors
37         c : out std_logic_vector(7 downto 0); --output
38         start: in std logic );
39  end multiplier;
40
41  architecture Behavioral of multiplier is
42     -- state machine in order to execute the operation
43     type statetype is (state0, state1, state2);
44     signal state: statetype;
45
46  begin
47  process(a, b)
48  variable PR:std logic vector(7 downto 0); --PR has to be twice the size (double with
    register)
49  variable BR:std_logic_vector(4 downto 0);
50
51  begin
52     case state is
53        when state0 => -- set output signals and next state
54           if start = '1' then
55              PR := (others => '0');
56              state <= state1;
```

```
57             else
58                state <= state0;
59             end if;
60          when state1 =>
61             c <= "00000000";
62             state <= state2;
63          when state2 => -- same here
64             PR := "0000"&b;
65             for Z in 1 to 4 Loop
66                if PR(0) = '1' then  -- if the LSB contains a 1
67                   -- add the multiplicand to the most significant half
68                   -- shift the whole register one bit to the right, throwing away the
    least significant bit
69                   -- and shifting the carry bit into the MSB
70                   BR := ('0' & a) + ('0' & PR(7 downto 4));
71                   PR := BR & PR(3 downto 1);
72                else
73                   PR := '0' & PR(7 downto 1);
74                end if;
75             end Loop;
76             c <= PR; -- put PR into output, c
77             state <= state0;
78       end case;
79    end Process;
80    end Behavioral;
```

Appendix F


```
  1  --------------------------------------------------------------------------------
  2  -- Company:
  3  -- Engineer:
  4  --
  5  -- Create Date:   10:58:27 02/02/2016
  6  -- Design Name:
  7  -- Module Name:
     C:/Users/vthangar/Documents/Lab2_Multiplier_4bit/Multiplier/Multiplier_TB.vhd
  8  -- Project Name:  Multiplier
  9  -- Target Device:
 10  -- Tool versions:
 11  -- Description:
 12  --
 13  -- VHDL Test Bench Created by ISE for module: multiplier
 14  --
 15  -- Dependencies:
 16  --
 17  -- Revision:
 18  -- Revision 0.01 - File Created
 19  -- Additional Comments:
 20  --
 21  -- Notes:
 22  -- This testbench has been automatically generated using types std_logic and
 23  -- std_logic_vector for the ports of the unit under test.  Xilinx recommends
 24  -- that these types always be used for the top-level I/O of a design in order
 25  -- to guarantee that the testbench will bind correctly to the post-implementation
 26  -- simulation model.
 27  --------------------------------------------------------------------------------
 28  LIBRARY ieee;
 29  USE ieee.std_logic_1164.ALL;
 30
 31  -- Uncomment the following library declaration if using
 32  -- arithmetic functions with Signed or Unsigned values
 33  --USE ieee.numeric_std.ALL;
 34
 35  ENTITY Multiplier_TB IS
 36  END Multiplier_TB;
 37
 38  ARCHITECTURE behavior OF Multiplier_TB IS
 39
 40      -- Component Declaration for the Unit Under Test (UUT)
 41
 42      COMPONENT multiplier
 43      PORT(
 44           a : IN  std_logic_vector(3 downto 0);
 45           b : IN  std_logic_vector(3 downto 0);
 46           c : OUT  std_logic_vector(7 downto 0);
 47           start : IN  std_logic
 48          );
 49      END COMPONENT;
 50
 51
 52     --Inputs
 53     signal a : std_logic_vector(3 downto 0) := (others => '0');
 54     signal b : std_logic_vector(3 downto 0) := (others => '0');
 55     signal start : std_logic := '0';
 56
```

```
57      --Outputs
58      signal c : std_logic_vector(7 downto 0) := (others => '0');
59      -- No clocks detected in port list. Replace <clock> below with
60      -- appropriate port name
61
62      --constant <clock>_period : time := 10 ns;
63
64   BEGIN
65
66      -- Instantiate the Unit Under Test (UUT)
67      uut: multiplier PORT MAP (
68             a => a,
69             b => b,
70             c => c,
71             start => start
72           );
73
74      -- Stimulus process
75      stim_proc: process
76      begin
77         -- hold reset state for 100 ns.
78         start <= '1';
79         wait for 10 ns;
80         start <= '0';
81         wait for 100 ns;
82         start <= '1';
83         a<= "1111";
84         b<= "1111";
85         wait for 100 ns;
86         a<= "1001";
87         b<= "1001";
88         wait for 100 ns;
89         wait;
90
91
92         wait;
93      end process;
94
95   END;
96
```