

Design of a Signed Number Parallel Multiplier

by Iman SayadiChoobari

FILE	2775_IMAN_SAYADICHOOBARI_DESIGN_OF_A_SIGNED_NUMBER_PARALLEL_MULTIPLIER_114255_1903176435.DOCX		
TIME SUBMITTED	11-MAY-2018 02:54PM (UTC+0100)	WORD COUNT	1190
SUBMISSION ID	88998519	CHARACTER COUNT	8046

2018

Digital Systems Synthesis

Design of a Signed Number Parallel Multiplier

IMAN SAYADI CHOOBARI P148275

Table of Contents

1. Introduction	3
2. Background Research to Multiplier	3
2.1. Braun Multiplier	4
2.2. Parallel Processing	4
3. Design Specification	5
3.1. What is VHDL?	5
3.2. What is Xilinx Design Software	6
3.3. What is test bench	7
4. Design.....	8
4.1. Half Adder	8
4.2. Full Adder	11
4.3. Line_1_8x8	15
4.4. Linex_8x8	21
4.5. Multi_8x8	32
5. Conclusion.....	45

1. Introduction

The purpose of this assignment is to design, simulate and synthesise an 8 bit x 8 bit signed number multiplier based upon a simple parallel Braun multiplier. The Xilinx design software has to utilize in order to overcome this goal.

Therefore, this assignment also briefly explains about multipliers, what are the purposes of them, why they important and different types of them.

2. Background Research to Multiplier

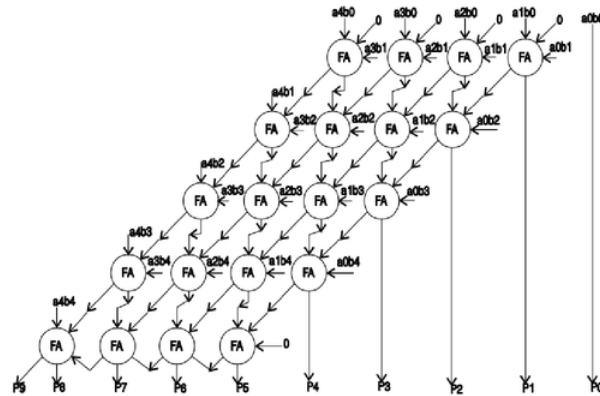
In the design of digital computing systems, especially in the design of discrete signal systems, one of the most important units is the multiplication unit. In fact, it can be surely stated that the slowest and most effective and efficient unit in the design of digital computing systems is the multiplication unit.

There are various of structures and topologies for designing the multipliers. These useful structures are such as:

- Bit Array Multiplier
- Bit Basic Braun Multiplier
- RCA Multiplier
- Row Bypassing based on RCA
- Bypassing Multiplier based on CSA

2.1. Braun Multiplier

Braun Multiplier also known as “carry save array multiplier” In this multiplier, with a special technique in transmission of output carry bit, power consumption and delay will reduce to minimum, therefore it has better swing.



The main factor of speed limitation on multiplier is the delay of passing the data through the circuit. Braun multiplier significantly reduces these delays by using pipeline techniques.

The combination of an AND gate and full address will produce the architecture of a Braun's multiplier.

2.2. Parallel Processing

Parallel array multipliers are widely used in order to produce more speed with low power demand in digital signal processing applications where most of the power is consumed by the multipliers. This is a term to specify a wide range of methods for processing the information simultaneously in order to increase the computational speed of a computer system. A parallel processing system can be used instead of series processing in order to processing information simultaneously, and thus the speed of system will be faster. For instance, while an instruction in Alu is running, the next command can be read from memory. The system can have two or more Alu and at the same time execute two or more instructions at same time. Additionally, it can have two or more processors that can work simultaneously.

The purpose of parallel processing is to accelerate the processing of the computer and increase its throughput. This means the amount of

processing is performing at a given time duration. In parallel processing the amount of hardware increases which also causes the cost of the system goes up. But technological advancements reduced this cost, therefore, parallel processing techniques are more sensible solution in term of economical perspective.

3. Design Specification

In the Design section of this assignment which is simulating an 8bit X 8bit signed number multiplier expected to prove the detail of design parameters and analysis of the design. In order to complete the design section, the VHDL model should be presented and data files have to be created, based on VHDL model. The Xilinx design software must utilized in order to achieve this aim.

3.1. What is VHDL?

VHDL is one of the hardware description languages that is used in the design of electronic circuits, more often to describe digital integrated circuits (ICs) or analogue and digital combinations. VHDL stands for "Very High Speed Description Language".

VHDL is one of the projects initially developed by the American Defence Forces and then it commercialized. VHDL is often used to design circuit blocks by programming, and after completing each block, the block can be tested and simulated. The blocks itself can be a part of larger blocks or work simultaneously with other blocks. Therefore, the system development and troubleshooting can be done easily.

A program or layout in VHDL is generally divided into three parts:

1. In the Library Declaration section all the libraries used in the code are introduced
2. The definition of the pins and the input / output ports and the signals (internal interfaces between the blocks), which is named ENTITY.
3. The section describing the operation of the circuit and the relationship between the inputs and outputs and the processing that should be performed on them, identified by the name Architecture.

Generally, with VHDL, simple and complex logical circuits can be implemented. Then convert this design to basic ICs using gates.

Among the VHDL simulators, the following software can be mentioned:

- Aldec Active HDL
- Cadence Incisive
- Mentor Graphics ModelSim
- Synopsys VCS-MX
- Xilinx Vivado based on the ISE tool-chain.

VHDL is one of the main programming language of FPGA.

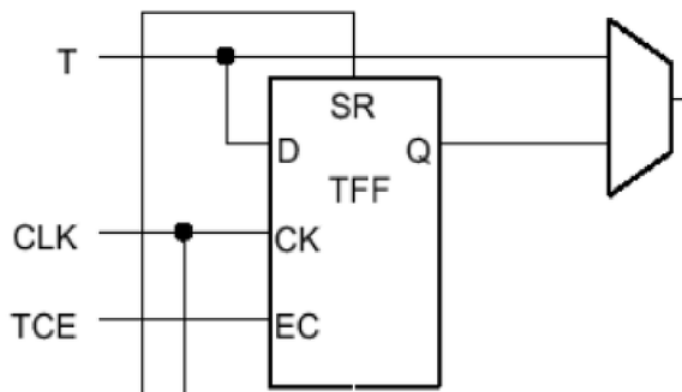
3.2. What is Xilinx Design Software

Xilinx is a manufacture company which provide Field Programmable Gate Array (FPGA) integrated circuits.

The advantages of using FPGAs are:

- Significant decrease in circuit size
- The integrity of the signals in the circuit increases
- The speed of the circuit increases
- High compression capacity.
- They are very cheap
- The supply of these ICs to the market is very low.
- The security of the circuit is high, and they cannot be copied.

Sequential circuits can also be implemented within a FPGA. In FPGA, the output of each LUT is transmitted directly or through a flip-flop to a multiplexer.



In some of the new Xilinx products, there are multiplier blocks available.

The multiplication function of these products can also be done at a higher speed by 18-bit internal multipliers.

These multipliers are commonly used in signal processing applications.

3.3. What is test bench

To simulate circuit performance, it is usually necessary to write another program other than the original program.

This program, which is for testing circuitry only, is called “test bench” for VHDL program.

Typically, the test program is that different waveform that contains all the possible inputs states is generated.

Usually, in writing the test bench program, the designer finds out a lot of negative points and modes that are neglected in the design of the main circuit.

4. Design

4.1. Half Adder

```
2  -- Company:
3  -- Engineer:      Iman Sayadi Choobari
4  --
5  -- Create Date:   09:44:22 03/08/2018
6  -- Design Name:
7  -- Module Name:   h_adder - 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 entity h_adder is                                --setting input / output of half adder
33     Port ( A : in  STD_LOGIC;                    -- A is input
34           B : in  STD_LOGIC;                    -- B is input
35           Sum : out STD_LOGIC;                  -- S is output
36           Carry : out STD_LOGIC);              -- C is output
37 end h_adder;
38
39 architecture Behavioral of h_adder is --starts half adder behaviour
40
41 begin
42
43     Sum <= A xor B;                                -- S is A xored by B
44     Carry <= A and B;                              -- C is A anded by B
45
46
47
48 end Behavioral;                                -- end of behaviour.
```

```

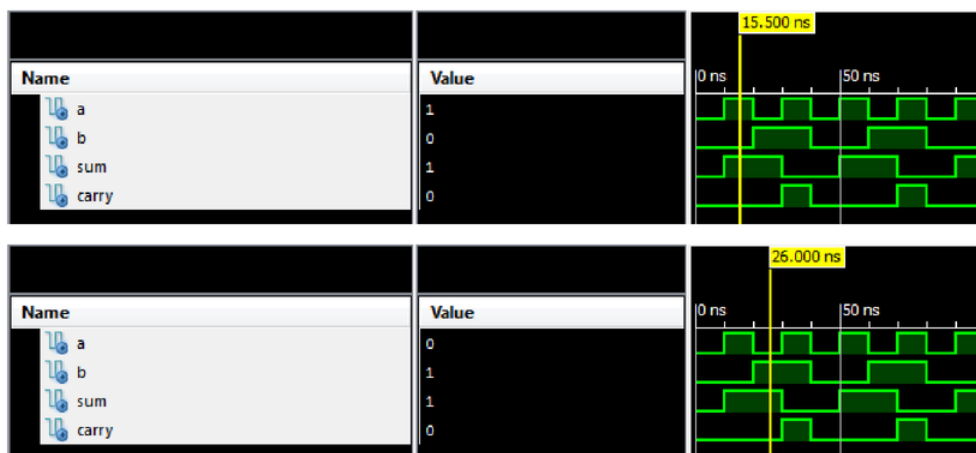
28 LIBRARY ieee;
29 USE ieee.std_logic_1164.ALL;
30
31 -- Uncomment the following library declaration if using
32 -- arithmetic functions with Signed or Unsigned values
33 --USE ieee.numeric_std.ALL;
34
35 ENTITY h_adder_tb IS
36 END h_adder_tb;
37
38 ARCHITECTURE behavior OF h_adder_tb IS
39
40     -- Component Declaration for the Unit Under Test (UUT)
41
42     COMPONENT h_adder                --calling component from half adder vhd
43     PORT(
44         A : IN  std_logic;           -- A is input
45         B : IN  std_logic;           -- B is input
46         Sum : OUT std_logic;          -- S is output
47         Carry : OUT std_logic        -- C is output
48     );
49     END COMPONENT;
50
51
52     --Inputs
53     signal A : std_logic := '0';     --set A as 0
54     signal B : std_logic := '0';     --set B as 0
55
56     --Outputs
57     signal Sum : std_logic;           --set signal S as S
58     signal Carry : std_logic;         --set signal |C as C
59     -- No clocks detected in port list. Replace <clock> below with
60     -- appropriate port name

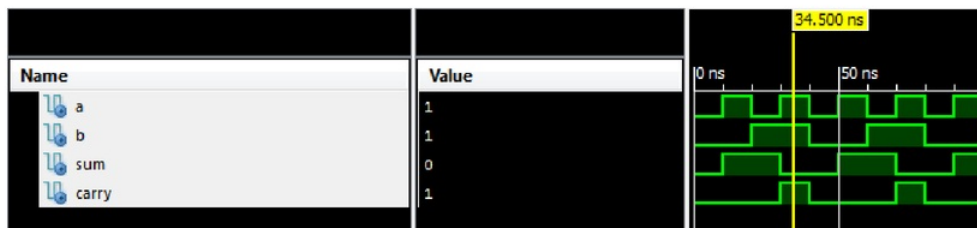
```

```

64 BEGIN
65
66     -- Instantiate the Unit Under Test (UUT)
67     uut: h_adder PORT MAP (           --port mapping for half adder
68         A => A,                       --assign A as A
69         B => B,                       --assign B as B
70         Sum => Sum,                   --assign S as S
71         Carry => Carry                --assign C as C
72     );
73
74
75     -- Stimulus process
76     stim_proc: process
77     begin
78         -- hold reset state for 100 ns.
79         wait for 10 ns;
80
81         A <= not A;
82
83     end process;
84
85     stim_proc1: process
86     begin
87         -- hold reset state for 100 ns.
88         wait for 20 ns;
89
90         B <= not B;
91
92     end process;
93
94 END;

```





4.2. Full Adder

```

2  -- Company:
3  -- Engineer:      Iman Sayadi Choobari
4  --
5  -- Create Date:   10:24:23 03/08/2018
6  -- Design Name:
7  -- Module Name:    f_adder - 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 entity f_adder is          --setting input / output of full adder
33     Port ( A : in  STD_LOGIC;  -- A is input
34           B : in  STD_LOGIC;  -- B is input
35           Cin : in  STD_LOGIC; -- Cin is input
36           Sum : out STD_LOGIC; -- S is output
37           Carry : out STD_LOGIC); -- Cout is output
38 end f_adder;
39
40 architecture Behavioral of f_adder is --start full adder behaviour
41
42 begin
43
44     Sum <= A xor B xor Cin;          -- S is A xored by B and xored by Cin
45     Carry <= (A and B) or ((A xor B) and Cin); -- Cout is A added by B or A xored by B
46                                           -- and added by Cin
47
48 end Behavioral;                  -- end of behaviour.

```

```

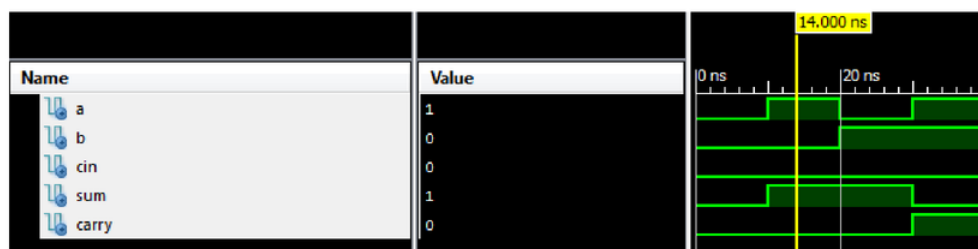
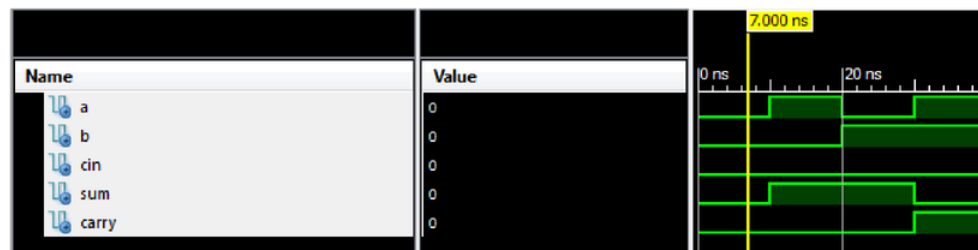
28 LIBRARY ieee;
29 USE ieee.std_logic_1164.ALL;
30
31 -- Uncomment the following library declaration if using
32 -- arithmetic functions with Signed or Unsigned values
33 --USE ieee.numeric_std.ALL;
34
35 ENTITY f_adder_tb IS
36 END f_adder_tb;
37
38 ARCHITECTURE behavior OF f_adder_tb IS
39
40     -- Component Declaration for the Unit Under Test (UUT)
41
42     COMPONENT f_adder          --calling component from full adder vhd
43     PORT(
44         A : IN  std_logic;      -- A is input
45         B : IN  std_logic;      -- B is input
46         Cin : IN  std_logic;    -- Cin is input
47         Sum : OUT std_logic;     -- S is output
48         Carry : OUT std_logic   -- Cout is output
49     );
50     END COMPONENT;
51
52
53     --Inputs
54     signal A : std_logic := '0'; --set A as 0
55     signal B : std_logic := '0'; --set B as 0
56     signal Cin : std_logic := '0'; --set Cin as 0
57
58     --Outputs
59     signal Sum : std_logic;        --set B out
60     signal Carry : std_logic;      --set Cout out
61     -- No clocks detected in port list. Replace <clock> below with
62     -- appropriate port name

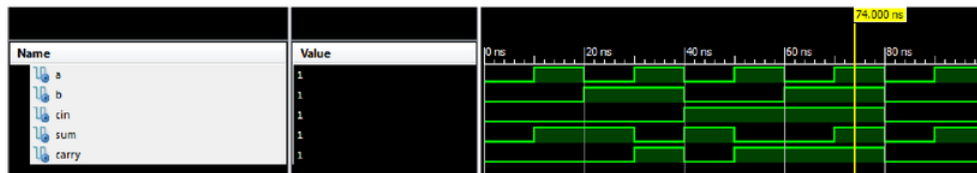
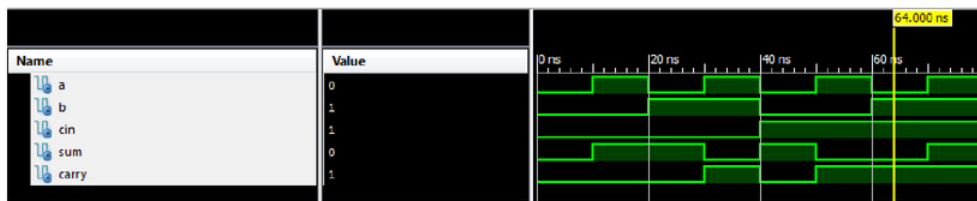
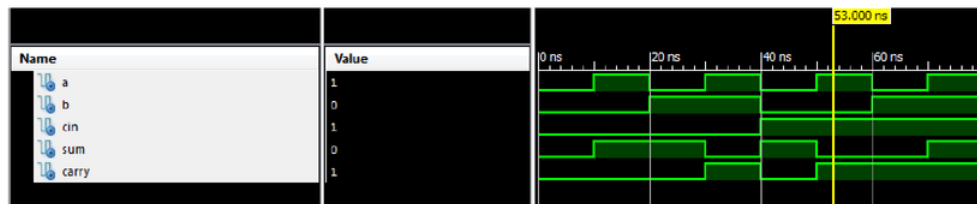
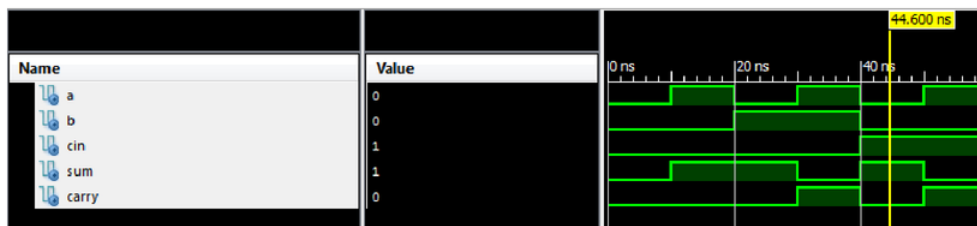
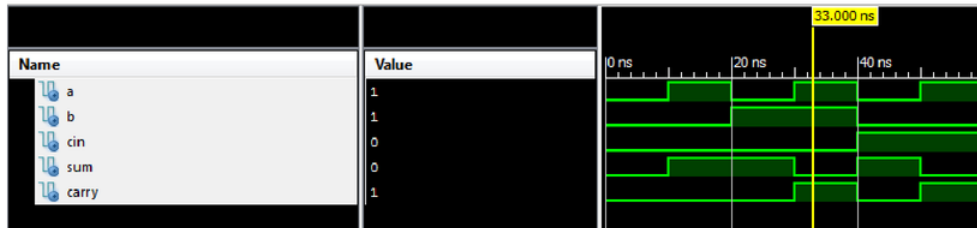
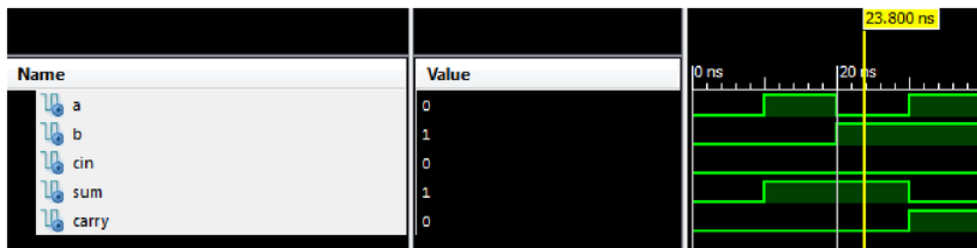
```

```

64 BEGIN
65
66     -- Instantiate the Unit Under Test (UUT)
67     uut: h_adder PORT MAP (           --port mapping for half adder
68         A => A,                       --assign A as A
69         B => B,                       --assign B as B
70         Sum => Sum,                   --assign S as S
71         Carry => Carry                --assign C as C
72     );
73
74
75     -- Stimulus process
76     stim_proc: process
77     begin
78         -- hold reset state for 100 ns.
79         wait for 10 ns;
80
81         A <= not A;
82
83     end process;
84
85     stim_proc1: process
86     begin
87         -- hold reset state for 100 ns.
88         wait for 20 ns;
89
90         B <= not B;
91
92     end process;
93
94 END;

```





4.3. Line1_8x8

```
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 entity Line1_8x8 is --setting input / output for Line1_8x8
33     Port ( A_0 : in STD_LOGIC; --A0 down to A7 is the input which is 8 bits
34           A_1 : in STD_LOGIC;
35           A_2 : in STD_LOGIC;
36           A_3 : in STD_LOGIC;
37           A_4 : in STD_LOGIC;
38           A_5 : in STD_LOGIC;
39           A_6 : in STD_LOGIC;
40           A_7 : in STD_LOGIC;
41           B : in STD_LOGIC; --B is a standard logic input
42           Q_0 : out STD_LOGIC;
43           Q_1 : out STD_LOGIC;
44           Q_2 : out STD_LOGIC;
45           Q_3 : out STD_LOGIC;
46           Q_4 : out STD_LOGIC;
47           Q_5 : out STD_LOGIC;
48           Q_6 : out STD_LOGIC;
49           Q_7 : out STD_LOGIC); --Q0 down to Q7 is the output which is 8 bits
50 end Line1_8x8;
51
52 architecture Behavioral of Line1_8x8 is --start 8x8 behaviour
53
54 begin
55     --Output Q is assigned with A0 to A7 inputs addded by B input
56     Q_0 <= A_0 and B;
57     Q_1 <= A_1 and B;
58     Q_2 <= A_2 and B;
59     Q_3 <= A_3 and B;
60     Q_4 <= A_4 and B;
61     Q_5 <= A_5 and B;
62     Q_6 <= A_6 and B;
63     Q_7 <= A_7 and B;
64
65 end Behavioral; --end of behaviour.
```

```

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 Line1_8x8_tb IS
36 END Line1_8x8_tb;
37
38 ARCHITECTURE behavior OF Line1_8x8_tb IS
39
40     -- Component Declaration for the Unit Under Test (UUT)
41
42     COMPONENT Line1_8x8          --calling Line_1_8x8 from vhd file
43     PORT(
44         A_0 : IN  std_logic;    -- setting A0 down to A7 as input
45         A_1 : IN  std_logic;
46         A_2 : IN  std_logic;
47         A_3 : IN  std_logic;
48         A_4 : IN  std_logic;
49         A_5 : IN  std_logic;
50         A_6 : IN  std_logic;
51         A_7 : IN  std_logic;
52         B : IN  std_logic;      -- setting B as input
53         Q_0 : OUT std_logic;    -- setting Q0 down to Q7 as output
54         Q_1 : OUT std_logic;
55         Q_2 : OUT std_logic;
56         Q_3 : OUT std_logic;
57         Q_4 : OUT std_logic;
58         Q_5 : OUT std_logic;
59         Q_6 : OUT std_logic;
60         Q_7 : OUT std_logic;
61     );
62     END COMPONENT;
63
64
65     --Inputs
66     signal A_0 : std_logic := '0';  --set signal A0 down to A7 as 0
67     signal A_1 : std_logic := '0';
68     signal A_2 : std_logic := '0';
69     signal A_3 : std_logic := '0';
70     signal A_4 : std_logic := '0';
71     signal A_5 : std_logic := '0';
72     signal A_6 : std_logic := '0';
73     signal A_7 : std_logic := '0';
74     signal B : std_logic := '0';    --set signal B as 0
75
76     --Outputs
77     signal Q_0 : std_logic;          --set signal Q0 down to Q7 as output
78     signal Q_1 : std_logic;
79     signal Q_2 : std_logic;
80     signal Q_3 : std_logic;
81     signal Q_4 : std_logic;
82     signal Q_5 : std_logic;
83     signal Q_6 : std_logic;
84     signal Q_7 : std_logic;
85     -- No clocks detected in port list. Replace <clock> below with
86     -- appropriate port name

```

```

90 BEGIN
91
92     -- Instantiate the Unit Under Test (UUT)
93     uut: Line1_8x8 PORT MAP (    --set port mapping for Line_1_8x8
94         A_0 => A_0,            -- assign A0 downto A7 as A0 downto A7
95         A_1 => A_1,
96         A_2 => A_2,
97         A_3 => A_3,
98         A_4 => A_4,
99         A_5 => A_5,
100        A_6 => A_6,
101        A_7 => A_7,
102        B => B,                -- assign B as B
103        Q_0 => Q_0,            -- assign Q0 downto Q7 as Q0 downto q7
104        Q_1 => Q_1,
105        Q_2 => Q_2,
106        Q_3 => Q_3,
107        Q_4 => Q_4,
108        Q_5 => Q_5,
109        Q_6 => Q_6,
110        Q_7 => Q_7
111    );

```

```

116  -- Stimulus process
117  stim_proc: process
118  begin
119      -- hold reset state for 100 ns.
120      wait for 10 ns;
121
122      A_0 <= not A_0;
123
124      -- insert stimulus here
125
126  end process;
127
128
129      stim_proc1: process
130  begin
131      -- hold reset state for 100 ns.
132      wait for 20 ns;
133
134      A_1 <= not A_1;
135
136      -- insert stimulus here
137
138  end process;
139      stim_proc2: process
140  begin
141      -- hold reset state for 100 ns.
142      wait for 40 ns;
143
144      A_2 <= not A_2;
145
146      -- insert stimulus here
147
148  end process;
149      stim_proc3: process
150  begin
151      -- hold reset state for 100 ns.
152      wait for 80 ns;
153
154      A_3 <= not A_3;
155
156      -- insert stimulus here
157
158  end process;

```

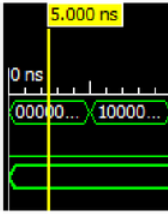
```

159     stim_proc4: process
160 begin
161     -- hold reset state for 100 ns.
162     wait for 160 ns;
163
164     A_4 <= not A_4;
165
166     -- insert stimulus here
167
168 end process;
169     stim_proc5: process
170 begin
171     -- hold reset state for 100 ns.
172     wait for 320 ns;
173
174     A_5 <= not A_5;
175
176     -- insert stimulus here
177
178 end process;
179     stim_proc6: process
180 begin
181     -- hold reset state for 100 ns.
182     wait for 640 ns;
183
184     A_6 <= not A_6;
185
186     -- insert stimulus here
187
188 end process;
189     stim_proc7: process
190 begin
191     -- hold reset state for 100 ns.
192     wait for 1280 ns;
193
194     A_7 <= not A_7;
195
196     -- insert stimulus here
197
198 end process;
199
200     stim_proc8: process
201 begin
202     -- hold reset state for 100 ns.
203     wait for 2560 ns;
204
205     B <= not B;
206
207     -- insert stimulus here
208
209 end process;
210 END;

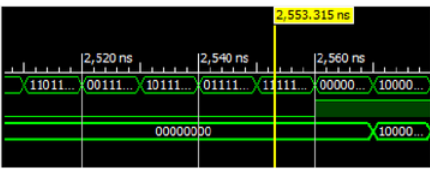
```

Name	Value
▼ New Virtual Bus	00000000
a_0	0
a_1	0
a_2	0
a_3	0
a_4	0
a_5	0
a_6	0
a_7	0
b	0
▼ New Virtual Bus	00000000
q_0	0
q_1	0
q_2	0
q_3	0
q_4	0
q_5	0
q_6	0
q_7	0

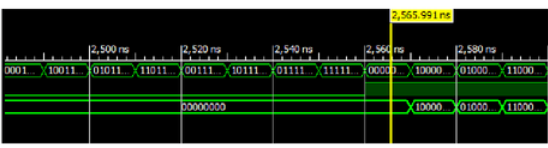
Name	Value
▶ New Virtual Bus	00000000
b	0
▶ New Virtual Bus	00000000



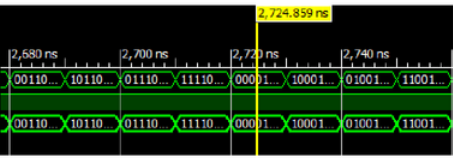
Name	Value
▶ New Virtual Bus	11111111
b	0
▶ New Virtual Bus	00000000



Name	Value
▶ New Virtual Bus	00000000
b	1
▶ New Virtual Bus	00000000



Name	Value
▶ New Virtual Bus	00001000
b	1
▶ New Virtual Bus	00001000



4.4. Linex_8x8

```
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 entity Linex_8x8 is                --setting input / output for Linex_8x8
33     Port ( A_0 : in  STD_LOGIC;    --A0 down to A7 is the input which is 8 bits
34           A_1 : in  STD_LOGIC;
35           A_2 : in  STD_LOGIC;
36           A_3 : in  STD_LOGIC;
37           A_4 : in  STD_LOGIC;
38           A_5 : in  STD_LOGIC;
39           A_6 : in  STD_LOGIC;
40           A_7 : in  STD_LOGIC;
41           B : in  STD_LOGIC;        --B is a standard logic input
42           X_0 : in  STD_LOGIC;      --X0 down to X7 is the input which is 8 bits
43           X_1 : in  STD_LOGIC;
44           X_2 : in  STD_LOGIC;
45           X_3 : in  STD_LOGIC;
46           X_4 : in  STD_LOGIC;
47           X_5 : in  STD_LOGIC;
48           X_6 : in  STD_LOGIC;
49           X_7 : in  STD_LOGIC;
50           Q_0 : out STD_LOGIC;      --Q0 down to Q7 is the output which is 8 bits
51           Q_1 : out STD_LOGIC;
52           Q_2 : out STD_LOGIC;
53           Q_3 : out STD_LOGIC;
54           Q_4 : out STD_LOGIC;
55           Q_5 : out STD_LOGIC;
56           Q_6 : out STD_LOGIC;
57           Q_7 : out STD_LOGIC;
58           Q_8 : out STD_LOGIC);
59 end Linex_8x8;
```

```

61 architecture Behavioral of Linex_8x8 is --start behaviour of Linex_8x8
62
63     COMPONENT h_adder
64     PORT(
65         A : IN  std_logic;    --A is input
66         B : IN  std_logic;    --B is input
67         Sum : OUT std_logic;   --S is output
68         Carry : OUT std_logic --C is output
69     );
70     END COMPONENT;
71
72     COMPONENT f_adder
73     PORT(
74         A : IN  std_logic;    --A is input
75         B : IN  std_logic;    --B is input
76         Cin : IN  std_logic;   --Cin is input
77         Sum : OUT std_logic;   --S is input
78         Carry : OUT std_logic  --Cout is input
79     );
80     END COMPONENT;
81
82     signal A0B : std_logic;    --assign A0B down to A7B as standard logic signal
83     signal A1B : std_logic;
84     signal A2B : std_logic;
85     signal A3B : std_logic;
86     signal A4B : std_logic;
87     signal A5B : std_logic;
88     signal A6B : std_logic;
89     signal A7B : std_logic;
90
91     signal C_0 : std_logic;    --assign C0 down to C6 as standard logic signal
92     signal C_1 : std_logic;
93     signal C_2 : std_logic;
94     signal C_3 : std_logic;
95     signal C_4 : std_logic;
96     signal C_5 : std_logic;
97     signal C_6 : std_logic;
98

```

```

99  begin
100
101      A0B <= A_0 and B;           -- A0B is the result of A0 added by B
102      A1B <= A_1 and B;           -- A1B is the result of A1 added by B
103      A2B <= A_2 and B;           -- A2B is the result of A2 added by B
104      A3B <= A_3 and B;           -- A3B is the result of A3 added by B
105      A4B <= A_4 and B;           -- A4B is the result of A4 added by B
106      A5B <= A_5 and B;           -- A5B is the result of A5 added by B
107      A6B <= A_6 and B;           -- A6B is the result of A6 added by B
108      A7B <= A_7 and B;           -- A7B is the result of A7 added by B
109
110      ha1 : h_adder
111      port map ( A0B , X_0 , Q_0 , C_0 );
112
113      fa1 : f_adder
114      port map ( A1B , X_1 , C_0, Q_1 , C_1 );
115
116      fa2 : f_adder
117      port map ( A2B , X_2 , C_1, Q_2 , C_2 );
118
119      fa3 : f_adder
120      port map ( A3B , X_3 , C_2, Q_3 , C_3 );
121
122      fa4 : f_adder
123      port map ( A4B , X_4 , C_3, Q_4 , C_4 );
124
125      fa5 : f_adder
126      port map ( A5B , X_5 , C_4, Q_5 , C_5 );
127
128      fa6 : f_adder
129      port map ( A6B , X_6 , C_5, Q_6 , C_6 );
130
131      fa7 : f_adder
132      port map ( A7B , X_7 , C_6, Q_7 , Q_8 );
133
134
135  end Behavioral;

```

```

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 Linex_8x8_tb IS
36  END Linex_8x8_tb;
37
38  ARCHITECTURE behavior OF Linex_8x8_tb IS
39
40      -- Component Declaration for the Unit Under Test (UUT)
41
42      COMPONENT Linex_8x8
43      PORT (
44          A_0 : IN  std_logic;
45          A_1 : IN  std_logic;
46          A_2 : IN  std_logic;
47          A_3 : IN  std_logic;
48          A_4 : IN  std_logic;
49          A_5 : IN  std_logic;
50          A_6 : IN  std_logic;
51          A_7 : IN  std_logic;
52          B : IN  std_logic;
53          X_0 : IN  std_logic;
54          X_1 : IN  std_logic;
55          X_2 : IN  std_logic;
56          X_3 : IN  std_logic;
57          X_4 : IN  std_logic;
58          X_5 : IN  std_logic;
59          X_6 : IN  std_logic;
60          X_7 : IN  std_logic;
61          Q_0 : OUT std_logic;
62          Q_1 : OUT std_logic;
63          Q_2 : OUT std_logic;
64          Q_3 : OUT std_logic;
65          Q_4 : OUT std_logic;
66          Q_5 : OUT std_logic;
67          Q_6 : OUT std_logic;
68          Q_7 : OUT std_logic;
69          Q_8 : OUT std_logic;
70      );
71  END COMPONENT;

```

```
74  --Inputs
75  signal A_0 : std_logic := '0';
76  signal A_1 : std_logic := '0';
77  signal A_2 : std_logic := '0';
78  signal A_3 : std_logic := '0';
79  signal A_4 : std_logic := '0';
80  signal A_5 : std_logic := '0';
81  signal A_6 : std_logic := '0';
82  signal A_7 : std_logic := '0';
83  signal B : std_logic := '0';
84  signal X_0 : std_logic := '0';
85  signal X_1 : std_logic := '0';
86  signal X_2 : std_logic := '0';
87  signal X_3 : std_logic := '0';
88  signal X_4 : std_logic := '0';
89  signal X_5 : std_logic := '0';
90  signal X_6 : std_logic := '0';
91  signal X_7 : std_logic := '0';
92
93  --Outputs
94  signal Q_0 : std_logic;
95  signal Q_1 : std_logic;
96  signal Q_2 : std_logic;
97  signal Q_3 : std_logic;
98  signal Q_4 : std_logic;
99  signal Q_5 : std_logic;
100 signal Q_6 : std_logic;
101 signal Q_7 : std_logic;
102 signal Q_8 : std_logic;
103 -- No clocks detected in port list. Replace <clock> below with
104 -- appropriate port name
```

```
108 BEGIN
109
110     -- Instantiate the Unit Under Test (UUT)
111     uut: Linex_8x8 PORT MAP (
112         A_0 => A_0,
113         A_1 => A_1,
114         A_2 => A_2,
115         A_3 => A_3,
116         A_4 => A_4,
117         A_5 => A_5,
118         A_6 => A_6,
119         A_7 => A_7,
120         B => B,
121         X_0 => X_0,
122         X_1 => X_1,
123         X_2 => X_2,
124         X_3 => X_3,
125         X_4 => X_4,
126         X_5 => X_5,
127         X_6 => X_6,
128         X_7 => X_7,
129         Q_0 => Q_0,
130         Q_1 => Q_1,
131         Q_2 => Q_2,
132         Q_3 => Q_3,
133         Q_4 => Q_4,
134         Q_5 => Q_5,
135         Q_6 => Q_6,
136         Q_7 => Q_7,
137         Q_8 => Q_8
138     );
```

```
140 -- Stimulus process
141 stim_procA_0: process
142 begin
143     -- hold reset state for 10 ns.
144     wait for 10 ns;
145
146     A_0 <= not A_0;
147
148 end process;
149
150
151 stim_procA_1: process
152 begin
153     -- hold reset state for 20 ns.
154     wait for 20 ns;
155
156     A_1 <= not A_1;
157
158 end process;
159
160
161 stim_procA_2: process
162 begin
163     wait for 40 ns;
164
165     A_2 <= not A_2;
166
167 end process;
168
169
170
171 stim_procA_3: process
172 begin
173     wait for 80 ns;
174
175     A_3 <= not A_3;
176
177 end process;
```

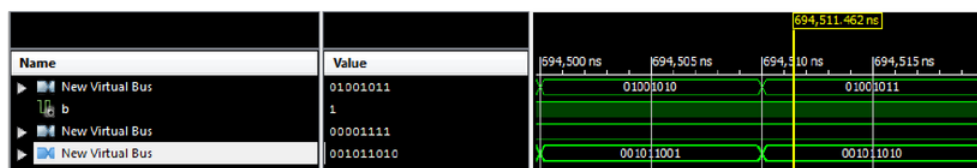
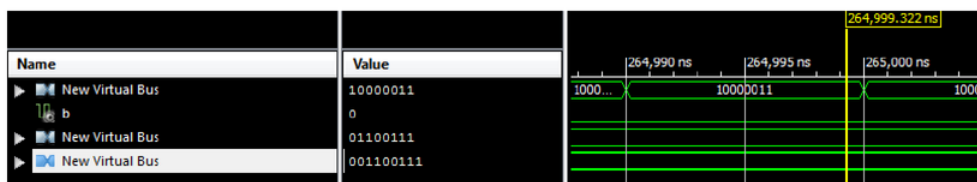
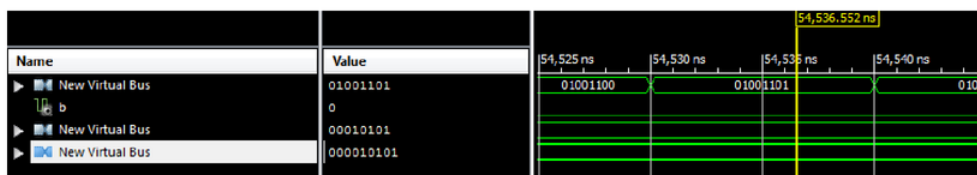
```
181 stim_procA_4: process
182 begin
183
184     wait for 160 ns;
185
186     A_4 <= not A_4;
187
188 end process;
189
190
191 stim_procA_5: process
192 begin
193
194     wait for 320 ns;
195
196     A_5 <= not A_5;
197
198 end process;
199
200
201 stim_procA_6: process
202 begin
203
204     wait for 640 ns;
205
206     A_6 <= not A_6;
207
208 end process;
209
210
211 stim_procA_7: process
212 begin
213
214     wait for 1280 ns;
215
216     A_7 <= not A_7;
217
218 end process;
219
220
221 stim_procX_0: process
222 begin
223
224     wait for 2560 ns;
225
226     X_0 <= not X_0;
227
228 end process;
```

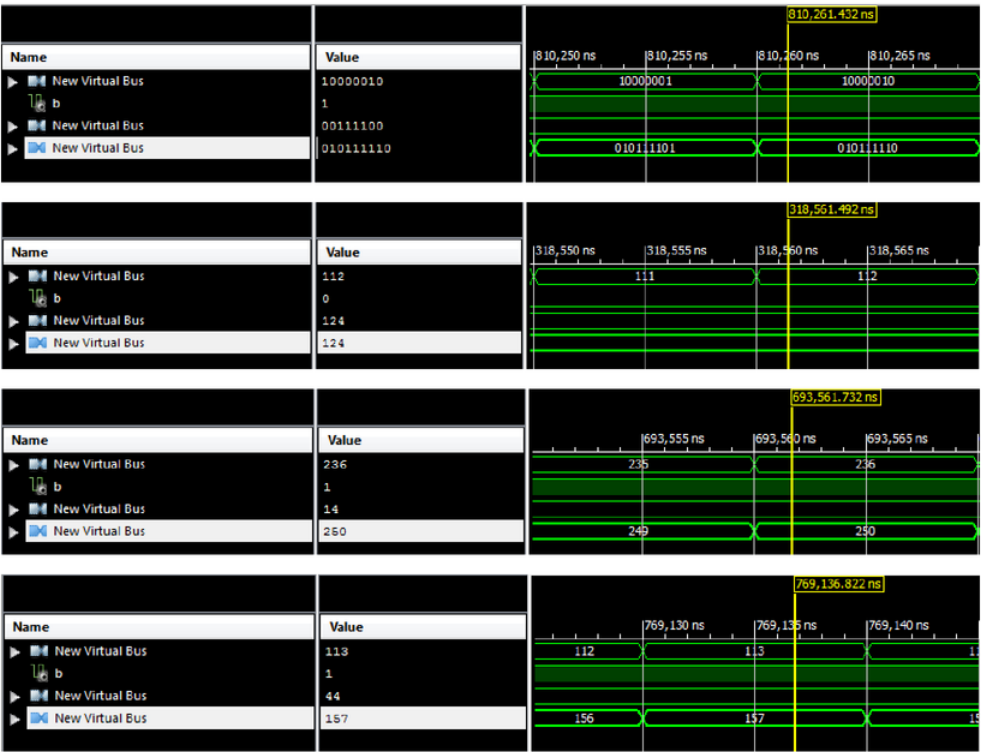
```
231 stim_procX_1: process
232 begin
233
234     wait for 5120 ns;
235
236     X_1 <= not X_1;
237
238 end process;
239
240
241 stim_procX_2: process
242 begin
243
244     wait for 10240 ns;
245
246     X_2 <= not X_2;
247
248 end process;
249
250
251 stim_procX_3: process
252 begin
253
254     wait for 20480 ns;
255
256     X_3 <= not X_3;
257
258 end process;
259
260
261 stim_procX_4: process
262 begin
263
264     wait for 40960 ns;
265
266     X_4 <= not X_4;
267
268 end process;
269
270
271 stim_procX_5: process
272 begin
273
274     wait for 81920 ns;
275
276     X_5 <= not X_5;
277
278 end process;
```

```

281 stim_procX_6: process
282 begin
283
284     wait for 163840 ns;
285
286     X_6 <= not X_6;
287
288 end process;
289
290
291 stim_procX_7: process
292 begin
293
294     wait for 327680 ns;
295
296     X_7 <= not X_7;
297
298 end process;
299
300
301 stim_procB: process
302 begin
303
304     wait for 655360 ns;
305
306     B <= not B;
307
308 end process;
309
310 END;

```





4.5. Multi_8x8

```
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 entity Multi_8x8 is
33     Port ( A_0 : in  STD_LOGIC;
34           A_1 : in  STD_LOGIC;
35           A_2 : in  STD_LOGIC;
36           A_3 : in  STD_LOGIC;
37           A_4 : in  STD_LOGIC;
38           A_5 : in  STD_LOGIC;
39           A_6 : in  STD_LOGIC;
40           A_7 : in  STD_LOGIC;
41           B_0 : in  STD_LOGIC;
42           B_1 : in  STD_LOGIC;
43           B_2 : in  STD_LOGIC;
44           B_3 : in  STD_LOGIC;
45           B_4 : in  STD_LOGIC;
46           B_5 : in  STD_LOGIC;
47           B_6 : in  STD_LOGIC;
48           B_7 : in  STD_LOGIC;
49           Q_0 : out STD_LOGIC;
50           Q_1 : out STD_LOGIC;
51           Q_2 : out STD_LOGIC;
52           Q_3 : out STD_LOGIC;
53           Q_4 : out STD_LOGIC;
54           Q_5 : out STD_LOGIC;
55           Q_6 : out STD_LOGIC;
56           Q_7 : out STD_LOGIC;
57           Q_8 : out STD_LOGIC;
58           Q_9 : out STD_LOGIC;
59           Q_10 : out STD_LOGIC;
60           Q_11 : out STD_LOGIC;
61           Q_12 : out STD_LOGIC;
62           Q_13 : out STD_LOGIC;
63           Q_14 : out STD_LOGIC;
64           Q_15 : out STD_LOGIC);
65 end Multi_8x8;
```

```
67 architecture Behavioral of Multi_8x8 is
```

```
68
```

```
69     COMPONENT Line1_8x8
```

```
70     PORT (
```

```
71         A_0 : IN  std_logic;
```

```
72         A_1 : IN  std_logic;
```

```
73         A_2 : IN  std_logic;
```

```
74         A_3 : IN  std_logic;
```

```
75         A_4 : IN  std_logic;
```

```
76         A_5 : IN  std_logic;
```

```
77         A_6 : IN  std_logic;
```

```
78         A_7 : IN  std_logic;
```

```
79         B : IN  std_logic;
```

```
80         Q_0 : OUT std_logic;
```

```
81         Q_1 : OUT std_logic;
```

```
82         Q_2 : OUT std_logic;
```

```
83         Q_3 : OUT std_logic;
```

```
84         Q_4 : OUT std_logic;
```

```
85         Q_5 : OUT std_logic;
```

```
86         Q_6 : OUT std_logic;
```

```
87         Q_7 : OUT std_logic
```

```
88     );
```

```
89     END COMPONENT;
```

```
92     COMPONENT Linex_8x8
```

```
93     PORT (
```

```
94         A_0 : IN  std_logic;
```

```
95         A_1 : IN  std_logic;
```

```
96         A_2 : IN  std_logic;
```

```
97         A_3 : IN  std_logic;
```

```
98         A_4 : IN  std_logic;
```

```
99         A_5 : IN  std_logic;
```

```
100        A_6 : IN  std_logic;
```

```
101        A_7 : IN  std_logic;
```

```
102        B : IN  std_logic;
```

```
103        X_0 : IN  std_logic;
```

```
104        X_1 : IN  std_logic;
```

```
105        X_2 : IN  std_logic;
```

```
106        X_3 : IN  std_logic;
```

```
107        X_4 : IN  std_logic;
```

```
108        X_5 : IN  std_logic;
```

```
109        X_6 : IN  std_logic;
```

```
110        X_7 : IN  std_logic;
```

```
111        Q_0 : OUT std_logic;
```

```
112        Q_1 : OUT std_logic;
```

```
113        Q_2 : OUT std_logic;
```

```
114        Q_3 : OUT std_logic;
```

```
115        Q_4 : OUT std_logic;
```

```
116        Q_5 : OUT std_logic;
```

```
117        Q_6 : OUT std_logic;
```

```
118        Q_7 : OUT std_logic;
```

```
119        Q_8 : OUT std_logic
```

```
120    );
```

```
121    END COMPONENT;
```

```
123     signal      R1_1 : std_logic;
124     signal      R1_2 : std_logic;
125     signal      R1_3 : std_logic;
126     signal      R1_4 : std_logic;
127     signal      R1_5 : std_logic;
128     signal      R1_6 : std_logic;
129     signal      R1_7 : std_logic;
130
131     signal      R2_1 : std_logic;
132     signal      R2_2 : std_logic;
133     signal      R2_3 : std_logic;
134     signal      R2_4 : std_logic;
135     signal      R2_5 : std_logic;
136     signal      R2_6 : std_logic;
137     signal      R2_7 : std_logic;
138     signal      R2_8 : std_logic;
139
140     signal      R3_1 : std_logic;
141     signal      R3_2 : std_logic;
142     signal      R3_3 : std_logic;
143     signal      R3_4 : std_logic;
144     signal      R3_5 : std_logic;
145     signal      R3_6 : std_logic;
146     signal      R3_7 : std_logic;
147     signal      R3_8 : std_logic;
148
149     signal      R4_1 : std_logic;
150     signal      R4_2 : std_logic;
151     signal      R4_3 : std_logic;
152     signal      R4_4 : std_logic;
153     signal      R4_5 : std_logic;
154     signal      R4_6 : std_logic;
155     signal      R4_7 : std_logic;
156     signal      R4_8 : std_logic;
157
158     signal      R5_1 : std_logic;
159     signal      R5_2 : std_logic;
160     signal      R5_3 : std_logic;
161     signal      R5_4 : std_logic;
162     signal      R5_5 : std_logic;
163     signal      R5_6 : std_logic;
164     signal      R5_7 : std_logic;
165     signal      R5_8 : std_logic;
166
```

```

167     signal      R6_1 : std_logic;
168     signal      R6_2 : std_logic;
169     signal      R6_3 : std_logic;
170     signal      R6_4 : std_logic;
171     signal      R6_5 : std_logic;
172     signal      R6_6 : std_logic;
173     signal      R6_7 : std_logic;
174     signal      R6_8 : std_logic;
175
176     signal      R7_1 : std_logic;
177     signal      R7_2 : std_logic;
178     signal      R7_3 : std_logic;
179     signal      R7_4 : std_logic;
180     signal      R7_5 : std_logic;
181     signal      R7_6 : std_logic;
182     signal      R7_7 : std_logic;
183     signal      R7_8 : std_logic;

```

```

185     begin
186
187     I1 : Line1_8x8
188     port map ( A_0, A_1, A_2, A_3, A_4, A_5, A_6, A_7, B_0, Q_0, R1_1, R1_2, R1_3, R1_4, R1_5, R1_6, R1_7 );
189
190     I2 : Line2_8x8
191     port map ( A_0, A_1, A_2, A_3, A_4, A_5, A_6, A_7, B_1, R1_1, R1_2, R1_3, R1_4, R1_5, R1_6, R1_7, '0' , Q_1, R2_1, R2_2, R2_3, R2_4, R2_5, R2_6, R2_7, R2_8 );
192
193     I3 : Line3_8x8
194     port map ( A_0, A_1, A_2, A_3, A_4, A_5, A_6, A_7, B_2, R2_1, R2_2, R2_3, R2_4, R2_5, R2_6, R2_7, R2_8, Q_2, R3_1, R3_2, R3_3, R3_4, R3_5, R3_6, R3_7, R3_8 );
195
196     I4 : Line4_8x8
197     port map ( A_0, A_1, A_2, A_3, A_4, A_5, A_6, A_7, B_3, R3_1, R3_2, R3_3, R3_4, R3_5, R3_6, R3_7, R3_8, Q_3, R4_1, R4_2, R4_3, R4_4, R4_5, R4_6, R4_7, R4_8 );
198
199     I5 : Line5_8x8
200     port map ( A_0, A_1, A_2, A_3, A_4, A_5, A_6, A_7, B_4, R4_1, R4_2, R4_3, R4_4, R4_5, R4_6, R4_7, R4_8, Q_4, R5_1, R5_2, R5_3, R5_4, R5_5, R5_6, R5_7, R5_8 );
201
202     I6 : Line6_8x8
203     port map ( A_0, A_1, A_2, A_3, A_4, A_5, A_6, A_7, B_5, R5_1, R5_2, R5_3, R5_4, R5_5, R5_6, R5_7, R5_8, Q_5, R6_1, R6_2, R6_3, R6_4, R6_5, R6_6, R6_7, R6_8 );
204
205     I7 : Line7_8x8
206     port map ( A_0, A_1, A_2, A_3, A_4, A_5, A_6, A_7, B_6, R6_1, R6_2, R6_3, R6_4, R6_5, R6_6, R6_7, R6_8, Q_6, R7_1, R7_2, R7_3, R7_4, R7_5, R7_6, R7_7, R7_8 );
207
208     I8 : Line8_8x8
209     port map ( A_0, A_1, A_2, A_3, A_4, A_5, A_6, A_7, B_7, R7_1, R7_2, R7_3, R7_4, R7_5, R7_6, R7_7, R7_8, Q_7, Q_8, Q_9, Q_10, Q_11, Q_12, Q_13, Q_14, Q_15 );
210
211
212
213
214     end Behavioral;

```

```

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 Multi_8x8_tb IS
36  END Multi_8x8_tb;
37
38  ARCHITECTURE behavior OF Multi_8x8_tb IS
39
40      -- Component Declaration for the Unit Under Test (UUT)
41
42      COMPONENT Multi_8x8
43      PORT(
44          A_0 : IN  std_logic;
45          A_1 : IN  std_logic;
46          A_2 : IN  std_logic;
47          A_3 : IN  std_logic;
48          A_4 : IN  std_logic;
49          A_5 : IN  std_logic;
50          A_6 : IN  std_logic;
51          A_7 : IN  std_logic;
52          B_0 : IN  std_logic;
53          B_1 : IN  std_logic;
54          B_2 : IN  std_logic;
55          B_3 : IN  std_logic;
56          B_4 : IN  std_logic;
57          B_5 : IN  std_logic;
58          B_6 : IN  std_logic;
59          B_7 : IN  std_logic;
60
61          Q_0 : OUT std_logic;
62          Q_1 : OUT std_logic;
63          Q_2 : OUT std_logic;
64          Q_3 : OUT std_logic;
65          Q_4 : OUT std_logic;
66          Q_5 : OUT std_logic;
67          Q_6 : OUT std_logic;
68          Q_7 : OUT std_logic;
69          Q_8 : OUT std_logic;
70          Q_9 : OUT std_logic;
71          Q_10 : OUT std_logic;
72          Q_11 : OUT std_logic;
73          Q_12 : OUT std_logic;
74          Q_13 : OUT std_logic;
75          Q_14 : OUT std_logic;
76          Q_15 : OUT std_logic;
77      );
78  END COMPONENT;

```

```
80  --Inputs
81  signal A_0 : std_logic := '0';
82  signal A_1 : std_logic := '0';
83  signal A_2 : std_logic := '0';
84  signal A_3 : std_logic := '0';
85  signal A_4 : std_logic := '0';
86  signal A_5 : std_logic := '0';
87  signal A_6 : std_logic := '0';
88  signal A_7 : std_logic := '0';
89  signal B_0 : std_logic := '0';
90  signal B_1 : std_logic := '0';
91  signal B_2 : std_logic := '0';
92  signal B_3 : std_logic := '0';
93  signal B_4 : std_logic := '0';
94  signal B_5 : std_logic := '0';
95  signal B_6 : std_logic := '0';
96  signal B_7 : std_logic := '0';
97
98  --Outputs
99  signal Q_0 : std_logic;
100 signal Q_1 : std_logic;
101 signal Q_2 : std_logic;
102 signal Q_3 : std_logic;
103 signal Q_4 : std_logic;
104 signal Q_5 : std_logic;
105 signal Q_6 : std_logic;
106 signal Q_7 : std_logic;
107 signal Q_8 : std_logic;
108 signal Q_9 : std_logic;
109 signal Q_10 : std_logic;
110 signal Q_11 : std_logic;
111 signal Q_12 : std_logic;
112 signal Q_13 : std_logic;
113 signal Q_14 : std_logic;
114 signal Q_15 : std_logic;
115 -- No clocks detected in port list. Replace <clock> below with
116 -- appropriate port name
117
```

```
120 BEGIN
121
122     -- Instantiate the Unit Under Test (UUT)
123     uut: Multi_8x8 PORT MAP (
124         A_0 => A_0,
125         A_1 => A_1,
126         A_2 => A_2,
127         A_3 => A_3,
128         A_4 => A_4,
129         A_5 => A_5,
130         A_6 => A_6,
131         A_7 => A_7,
132         B_0 => B_0,
133         B_1 => B_1,
134         B_2 => B_2,
135         B_3 => B_3,
136         B_4 => B_4,
137         B_5 => B_5,
138         B_6 => B_6,
139         B_7 => B_7,
140         Q_0 => Q_0,
141         Q_1 => Q_1,
142         Q_2 => Q_2,
143         Q_3 => Q_3,
144         Q_4 => Q_4,
145         Q_5 => Q_5,
146         Q_6 => Q_6,
147         Q_7 => Q_7,
148         Q_8 => Q_8,
149         Q_9 => Q_9,
150         Q_10 => Q_10,
151         Q_11 => Q_11,
152         Q_12 => Q_12,
153         Q_13 => Q_13,
154         Q_14 => Q_14,
155         Q_15 => Q_15
156     );
```

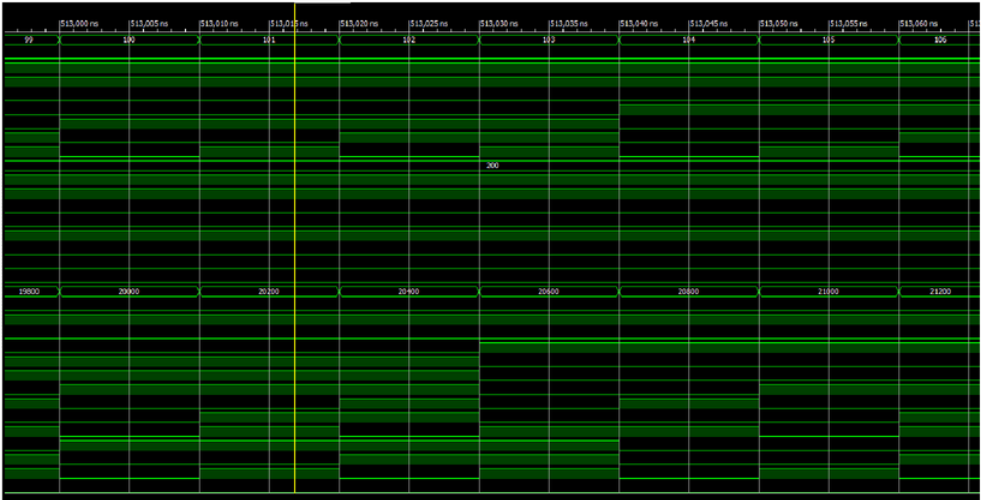
```
160 -- Stimulus process
161 stim_proc: process
162 begin
163     -- hold reset state for 100 ns.
164     wait for 10 ns;
165     A_0 <= not A_0;
166
167 end process;
168
169     stim_proc1: process
170 begin
171     -- hold reset state for 100 ns.
172     wait for 20 ns;
173     A_1 <= not A_1;
174
175 end process;
176
177     stim_proc2: process
178 begin
179     -- hold reset state for 100 ns.
180     wait for 40 ns;
181     A_2 <= not A_2;
182
183 end process;
184
185     stim_proc3: process
186 begin
187     -- hold reset state for 100 ns.
188     wait for 80 ns;
189     A_3 <= not A_3;
190
191 end process;
192
193     stim_proc4: process
194 begin
195     -- hold reset state for 100 ns.
196     wait for 160 ns;
197     A_4 <= not A_4;
198 end process;
199
200 stim_proc5: process
201 begin
202     -- hold reset state for 100 ns.
203     wait for 320 ns;
204     A_5 <= not A_5;
205
206 end process;
```

```
208     stim_proc6: process
209 begin
210     -- hold reset state for 100 ns.
211     wait for 640 ns;
212     A_6 <= not A_6;
213
214 end process;
215
216     stim_proc7: process
217 begin
218     -- hold reset state for 100 ns.
219     wait for 1280 ns;
220     A_7 <= not A_7;
221
222 end process;
223
224     stim_proc8: process
225 begin
226     -- hold reset state for 100 ns.
227     wait for 2560 ns;
228     B_0 <= not B_0;
229
230 end process;
231
232     stim_proc9: process
233 begin
234     -- hold reset state for 100 ns.
235     wait for 5120 ns;
236     B_1 <= not B_1;
237
238 end process;
239
240     stim_proc10: process
241 begin
242     -- hold reset state for 100 ns.
243     wait for 10240 ns;
244     B_2 <= not B_2;
245
246 end process;
247
248     stim_proc11: process
249 begin
250     -- hold reset state for 100 ns.
251     wait for 20480 ns;
252     B_3 <= not B_3;
253 end process;
```

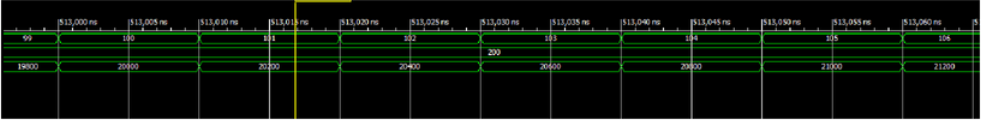
```
255     stim_proc12: process
256 begin
257     -- hold reset state for 100 ns.
258     wait for 40960 ns;
259     B_4 <= not B_4;
260
261 end process;
262
263     stim_proc13: process
264 begin
265     -- hold reset state for 100 ns.
266     wait for 81920 ns;
267     B_5 <= not B_5;
268
269 end process;
270
271     stim_proc14: process
272 begin
273     -- hold reset state for 100 ns.
274     wait for 163840 ns;
275     B_6 <= not B_6;
276
277 end process;
278
279     stim_proc15: process
280 begin
281     -- hold reset state for 100 ns.
282     wait for 327680 ns;
283     B_7 <= not B_7;
284
285 end process;
286
287 END;
```

Name		Value
 a_0		0
 a_1		0
 a_2		1
 a_3		0
 a_4		0
 a_5		1
 a_6		1
 a_7		0
 b_0		0
 b_1		0
 b_2		0
 b_3		0
 b_4		0
 b_5		0
 b_6		0
 b_7		0
 q_0		0
 q_1		0
 q_2		0
 q_3		0
 q_4		0
 q_5		0
 q_6		0
 q_7		0
 q_8		0
 q_9		0
 q_10		0
 q_11		0
 q_12		0
 q_13		0
 q_14		0
 q_15		0

Name	Value
▼ New Virtual Bus	100
a_7	0
a_6	1
a_5	1
a_4	0
a_3	0
a_2	1
a_1	0
a_0	0
▼ New Virtual Bus	200
b_7	1
b_6	1
b_5	0
b_4	0
b_3	1
b_2	0
b_1	0
b_0	0
▼ New Virtual Bus	20000
q_15	0
q_14	1
q_13	0
q_12	0
q_11	1
q_10	1
q_9	1
q_8	0
q_7	0
q_6	0
q_5	1
q_4	0
q_3	0
q_2	0
q_1	0
q_0	0



Name	Value
▶ New Virtual Bus	100
▶ New Virtual Bus	200
▶ New Virtual Bus	20000



Name	Value	29,140 ns	29,160 ns	29,180 ns	29,200 ns
▶ New Virtual Bus	102	98	99	100	101
▶ New Virtual Bus	11	102	103	104	105
▶ New Virtual Bus	1122	1078	1089	1100	1111

Name	Value	29,140 ns	29,160 ns	29,180 ns	29,200 ns
▶ New Virtual Bus	01100110	01100...	01100...	01100...	01101...
▶ New Virtual Bus	00001011	00000...	00000...	00000...	00000...
▶ New Virtual Bus	0000010001100010	00000...	00000...	00000...	00000...

5. Conclusion

The 8bit x 8bit unsigned parallel Braun multiplier was design, simulate and synthesise in this report.

The Field Programmable Gate Array (FPGA) produced the design software introduced from Xilinx company was utilized to complete the requirements.

The formal laboratory report and screen shots of different stages are included in order to prove the simulation results.

This report also highlights the VDHL and its functionality.

In overall five of different blocks have been implemented starting by half adder. Then it flows to full adder, Line_1_8x8, Linex_8x8 and finally Multi_8x8.

Test benches also was developed in order to making sure all the blocks are working perfectly fine before moving ahead.

6. References

Design of a Signed Number Parallel Multiplier

GRADEMARK REPORT

FINAL GRADE

GENERAL COMMENTS

Instructor

61/100

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13

PAGE 14

PAGE 15

PAGE 16

PAGE 17

PAGE 18

PAGE 19

PAGE 20

PAGE 21

PAGE 22

PAGE 23

PAGE 24

PAGE 25

PAGE 26

PAGE 27

PAGE 28

PAGE 29

PAGE 30

PAGE 31

PAGE 32

PAGE 33

PAGE 34

PAGE 35

PAGE 36

PAGE 37

PAGE 38

PAGE 39

PAGE 40

PAGE 41

PAGE 42

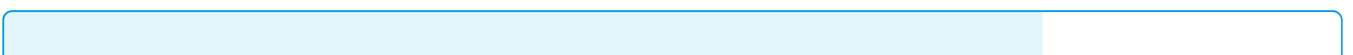
PAGE 43

PAGE 44

PAGE 45

PAGE 46

GRADING FORM: SMU GENERIC_GJ



RESEARCH-INFORMED LITERATURE

Very limited research with no other multipliers designs, could have been vastly improved more types and a comparison table. 45%

4.5**KNOWLEDGE AND UNDERSTANDING**

You have shown competency and understanding with your VHDL code. Your code is readable and remarked. 66%

19.8**ANALYSIS**

You have demonstrated your analysis of the design at each stage and commented on each section, with brief explanations of operation. 61%

12.2**PRACTICAL APPLICATION AND DEP.**

You have produced good simulation results, clearly demonstrating the operation of the circuit. Each section of the design has been tested before reuse in the subsequent sections. 65%

19.5**SKILLS FOR PROFESSIONAL PRAC.**

The report is reasonably presented with some good diagrams and illustrations. No bibliography or

4.8

reference section is included. 48%