Computer Logic - Practical 3

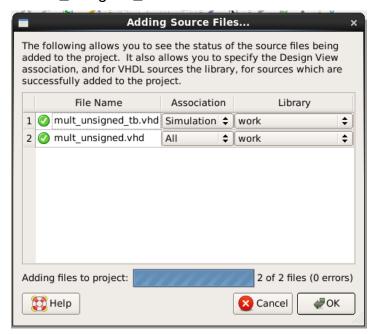
Objective:

To investigate unsigned and signed multipliers.

Tasks:

1) A new project called "Project3" was created and the provided files "mult_unsigned.vhd" and "mult_unsigned_tb.vhd" were added to it from Project:

Add Source. The association was set to All for the mult_unsigned.vhd file and to Simulation for the mult_unsigned_tb.vhd file.



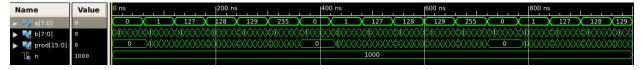
- 2) The file mult_unsigned.vhd was then opened. This included a shift_mux process which contained for loops and also if statements. However, this process was not complete and so was modified as follows:
- a) The contents of row corresponding to Algorithm 1, Line 3 were generalized to be true for all iterations by altering the code as shown.

b) Algorithm 1, Line 4 was modified to add the value of the row to the total in the if statement as can be seen.

```
-- Task 2b; Algorithm 1, Line 4
if b(i) = '1' then
```

3) The test bench *mult_unsigned_tb.vhd* was then opened and modified with an additional 4 hexadecimal values so that there are 6 in all.

After this was done, the test bench was simulated using *Behavioral Simulation* to test the multiplier as can be seen in the next snippet. The Radix was changed to *Unsigned Decimal* since the multiplication was unsigned and all 36 possible products were verified to give the expected output for correctness sake.



4) The hardware area of the multiplier was measured next. This was done automatically by the program when double clicking the *Synthesize* option found in the *Implementation view*, on the *Processes* pane when the multiplier is selected. The hardware footprint of the design was then found from the *Design Summary*, in the *Device Utilization Summary* when reading the *Number of Slice LUTs*. For this particular multiplier, the hardware area was found to be 94.

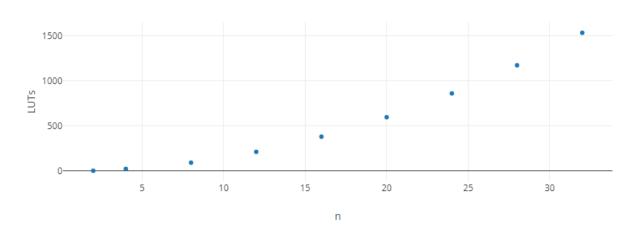
Device Utilization Summary (estimated values)												
Logic Utilization	Used	Available	Utilization									
Number of Slice LUTs	94	27288		0%								
Number of fully used LUT-FF pairs	0	94		0%								
Number of bonded IOBs	32	296		10%								

5) In this next step, n was changed in the range of $2 \le n \le 32$ inside the generic part of the entity and step 4 was repeated each time to obtain a measure of the hardware area for these different values. The results are listed on the table below

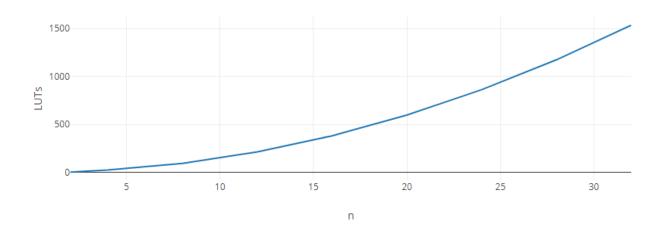
n	LUTs
2	4
4	25
8	94
12	214
16	382
20	598
24	862
28	1174
32	1534

6) A graph of LUTs (y-axis) against n (x-axis) was plotted and this graph seemed to relate mostly to a quadratic curve. The reason why is simple. The larger the value of *n*, the bigger the hardware area required so it would not make sense for such a graph to represent a square curve or a cubic one implying that for certain small values of *n*, the hardware area required is larger than for other values of *n* in which *n* is larger than before. The 2 snippets below represent this graph distribution, one with the plotted points and the other with the points connected.





Graph of LUTs against n



- 7) The whole *mult_unsigned.vhd* file was copied to a new project file entitled *mult_signed.vhd*. In this file any appearances of *mult_unsigned* were changed to *mult_signed*. The *shift_mux* process was also altered to perform signed multiplication instead of unsigned as explained in the following 3 steps:
 - a) All appearances of *mult_unsigned* were changed to *mult_signed*.
 - b) a was sign-extended when setting the row variable this time. This was done by adding an extra line as shown together with the 2 previously written lines.

This line copies the MSB of a and adds it to the sign extension bits

c) The line *total* := *total* + *unsigned(row)*; was replaced with the following conditional statement since in signed multiplication, both addition and subtraction are performed when calculating the total.

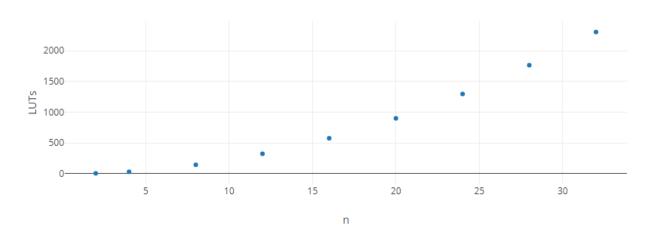
```
if i = n-1 then-- Algorithm 2, Line 5
   total := total - signed(row); -- Algorithm 2, Line 6
else -- Algorithm 2, Line 7
   total := total + signed(row); -- Algorithm 2, Line 8
end if; -- Algorithm 2, Line 9
```

- 8) The file *mult_unsigned_tb.vhd* was copied to a newly created file *mult_signed_tb.vhd* which was linked to this project. All appearances of *mult_unsigned* were once again changed to *mult_signed* and the test bench was simulated and checked to be correct. The radix of the simulation was changed to *Signed Decimal*.
- 9) The file $mult_signed.vhd$ was set as the **Top Module** and a measure of the hardware area in the range $2 \le n \le 32$ was obtained. A table with the possible values was written down. From this table, 2 versions of a graph of LUTs (y-axis) against n (x-axis) were drawn up as shown. The first version shows the plotted points and the second shows the points connected. This graph, just like the one shown in step 6 is a quadratic curve for the same reason explained already.

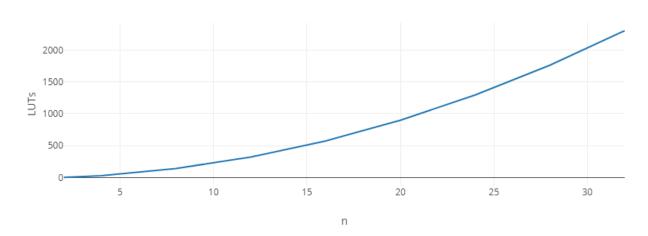
n	LUTs
2	4
4	30
8	143
12	323
16	575

20	899
24	1295
28	1763
32	2303

Graph of LUTs against n



Graph of LUTs against n



Appendix (Code):

mult_usnigned.vhd:

```
1 library ieee;
 2 use ieee.std_logic_1164.all;
 3 use ieee.numeric_std.all;
 4
 5 entity mult_unsigned is
 6
        -- Generic constants can be used to set the sizes of ports.
 7
        generic (
 8
            n : integer := 8);
 9
10
       port (
11
                : in std_logic_vector(n - 1 downto 0);
12
            a
                : in std_logic_vector(n - 1 downto 0);
13
            prod : out std_logic_vector(2 * n - 1 downto 0));
14
15
16 end entity mult_unsigned;
17
18 architecture behavioral of mult_unsigned is
19
20 begin -- architecture behavioral
21
        -- Whenever a signal in the process sensitivity list (a, b) changes,
22
        -- the outputs are changed according to the process contents.
23
24
        shift_mux : process (a, b) is
25
            -- Variables go here. Unlike signals, variables are sequential,
26
            -- that is, they can be changed multiple times in sequence.
27
            variable row : std_logic_vector(2 * n - 1 downto 0);
28
            variable total : unsigned(2 * n - 1 downto 0);
29
       begin -- process shift_mux
30
31
            total := (others => '0');
                                          -- Algorithm 1, Line 1
32
33
            -- iterate for each row
34
            for i in 0 to n - 1 loop
                                          -- Algorithm 1, Line 2
35
36
                -- Task 2a: Algorithm 1, Line 3
37
                row := (others => '0'); -- set row to 0
38
                row(n-1+i downto i) := a;
                                              -- set n bits to a
39
40
               -- Task 2b; Algorithm 1, Line 4
41
               if b(i) = '1' then
42
                   -- Since total is of type unsigned, we can add using '+',
43
                   -- but row needs to be cast to unsigned as well.
44
                   total := total + unsigned(row); -- Algorithm 1, Line 5
45
46
               end if; -- Algorithm 1, Line 6
47
48
            end loop; -- Algorithm 1, Line 7
49
 50
            -- Variables are local to the process, so the result
 51
            -- should be copied to the output signal prod.
            prod <= std_logic_vector(total);</pre>
 53
        end process shift_mux;
 55
 56 end behavioral;
57
```

mult_unsigned_tb.vhd:

```
1 library ieee;
 2 use ieee.std_logic_1164.all;
 3 use ieee.numeric_std.all;
 5 entity mult_unsigned_tb is
 7 end entity mult_unsigned_tb;
9 architecture behavioral of mult_unsigned_tb is
10
        component mult_unsigned is
11
           generic (
12
              n : integer);
13
14
            port (
                     : in std_logic_vector(n - 1 downto 0);
15
               b : in std_logic_vector(n - 1 downto 0);
16
               prod : out std_logic_vector(2 * n - 1 downto 0));
17
        end component mult_unsigned;
18
19
20
        -- generics
        constant n : integer := 8;
21
22
        -- inputs
23
        signal a : std_logic_vector(n - 1 downto 0) := (others => '0');
24
        signal b : std_logic_vector(n - 1 downto 0) := (others => '0');
25
26
        -- outputs
27
28
        signal prod : std_logic_vector(2 * n - 1 downto 0);
29
30 begin -- architecture behavioral
31
        uut : mult_unsigned
32
            generic map (
33
               n => n)
34
35
            port map (
               a => a,
b => b,
36
37
38
                prod => prod);
39
        -- stimulus process
40
41
        stim_proc : process
            type arr is array(0 to 5) of std_logic_vector(7 downto 0);
42
43
            constant values : arr := (
                x"00",
                                               -- hex 00
44
                 x"01",
45
                                              -- hex 01
                 x"7F",
46
                 x"80",
47
                 x"81",
48
                 x"FF"
49
50
                 );
        begin
51
            for i in 0 to 5 loop
52
                 a <= values(i);
53
                 for j in 0 to 5 loop
54
                     b <= values(j);
55
                     wait for 10 ns;
56
                 end loop; -- j
57
            end loop; -- i
58
        end process;
59
60
61 end architecture behavioral;
```

mult_signed.vhd:

```
1 library ieee;
 2 use ieee.std_logic_1164.all;
 3 use ieee.numeric_std.all;
 5 entity mult_signed is
 7
        -- Generic constants can be used to set the sizes of ports.
 8
           n : integer := 8);
 9
10
          a : in std_logic_vector(n - 1 downto 0);
b : in std_logic_vect
       port (
11
12
13
           prod : out std_logic_vector(2 * n - 1 downto 0));
14
15
16 end entity mult_signed;
17
18 architecture behavioral of mult_signed is
19
20 begin -- architecture behavioral
21
        -- Whenever a signal in the process sensitivity list (a, b) changes,
        -- the outputs are changed according to the process contents.
23
24
        shift_mux : process (a, b) is
25
            -- Variables go here. Unlike signals, variables are sequential,
26
            -- that is, they can be changed multiple times in sequence.
27
           variable row : std_logic_vector(2 * n - 1 downto 0);
28
           variable total : signed(2 * n - 1 downto 0);
29
       begin -- process shift_mux
3.0
31
            total := (others => '0');
                                           -- Algorithm 2, Line 1
32
33
            -- iterate for each row
34
            for i in 0 to n - 1 loop
                                            -- Algorithm 2, Line 2
35
36
                -- Algorithm 2, Line 3
37
               row := (others => '0');
38
                                               -- set row to 0
                row(n-1+i downto i) := a;
                                               -- set n bits to a
39
               row(2*n-1 downto n+i) := (others => a(n-1));
40
41
                -- Algorithm 2, Line 4
42
43
                if b(i) = '1' then
44
                   if i = n-1 then-- Algorithm 2, Line 5
45
                      total := total - signed(row);-- Algorithm 2, Line 6
46
                   else -- Algorithm 2, Line 7
47
                      total := total + signed(row); -- Algorithm 2, Line 8
48
                   end if; -- Algorithm 2, Line 9
49
50
                end if; -- Algorithm 2, Line 10
51
52
53
            end loop; -- Algorithm 2, Line 11
54
            -- Variables are local to the process, so the result
55
            -- should be copied to the output signal prod.
56
            prod <= std_logic_vector(total);</pre>
57
58
        end process shift_mux;
59
60 end behavioral;
```

mult_signed_tb.vhd:

```
1 library ieee;
 2 use ieee.std_logic_1164.all;
 3 use ieee.numeric_std.all;
 5 entity mult_signed_tb is
 7 end entity mult_signed_tb;
 8
 9 architecture behavioral of mult_signed_tb is
10
        component mult_signed is
11
12
           generic (
              n : integer);
1.3
           port (
14
               a : in std_logic_vector(n - 1 downto 0,,
b : in std_logic_vector(n - 1 downto 0);
15
16
               prod : out std_logic_vector(2 * n - 1 downto 0));
17
       end component mult_signed;
18
19
        -- generics
20
        constant n : integer := 8;
21
22
23
        -- inputs
        signal a : std_logic_vector(n - 1 downto 0) := (others => '0');
24
        signal b : std_logic_vector(n - 1 downto 0) := (others => '0');
25
26
27
        signal prod : std_logic_vector(2 * n - 1 downto 0);
28
29
30 begin -- architecture behavioral
31
        uut : mult_signed
32
           generic map (
33
34
                n => n)
35
            port map (
                a => a,
b => b,
36
37
                prod => prod);
38
39
40
        -- stimulus process
        stim_proc : process
41
            type arr is array(0 to 5) of std_logic_vector(7 downto 0);
42
43
            constant values : arr := (
               x"00",
                                              -- hex 00
44
                x"01",
                                              -- hex 01
45
                х"7F",
46
47
                x"80",
                x"81",
48
                 x"FF"
49
                 );
50
51
        begin
            for i in 0 to 5 loop
52
                 a <= values(i);
53
                 for j in 0 to 5 loop
54
55
                     b <= values(j);
                     wait for 10 ns;
56
                end loop; -- j
57
            end loop; -- i
58
59
        end process;
61 end architecture behavioral;
```

Simulated Behavioral Models:

Each snippet below shows only 200ns of the Simulated Model. One snippet leads onto the next 200ns.

mult_unsigned_tb.vhd:

Name Value 0 ns 0 n	عقسسا استعقاسا استعا		128 129 255 0 1	140 ns 160 ns 180 ns 127 (127 X 128 X 129 X 255 X 0 X 1
Ten Prod(15:0) 0 1000 1000	0	280 ns 300 ns	128	(16129 × 16256 × 16383 × 32385 × 0 × 128 × 128 × 1360 ns 1360 ns 1400 ns
128 (127 \ 128 \ 129 \ 255 (16256 \ 16384 \ 16512 \ 32640	129 0 1 127 128 0 129 16383 16512	129 X 255 O X	255 1 127 128 129 29 255 32385 32640 32895 650	0 0 1 127 128 129
400 ns	440 ns 460 ns	480 ns 500 ns	520 ns 540 ns	
0 129 (255) 0 (1	127 × 128 × 129 × 255 127 × 128 × 129 × 255		128 129 255 0 1 6256 16383 32385 0 1	
620 ns 620 ns 129	640 ns 660 ns	680 ns 700 ns 255	720 ns 740 ns 740 ns 0	760 ns 780 ns 800 ns
0 1 127 128 0 129 16383 (16512	129 \ 255 \ 0 \ 1 (16641 \ 32895 \ 0 \ 255	127 128 129 2 32385 32640 32895 65 1000	025 0 1 127 128	3 129 × 255 0 × 1 127 ×
800 ns 820 ns	840 ns 860 ns	880 ns 900 ns	920 ns 940 ns 128	960 ns 980 ns 1,000 ns
127 128 129 255 127 128 129 255	0 1 127 128 0 127 16129 16256		1 127 128 129 255 28 16256 (16384 (16512 (3264	

mult_signed_tb.vhd:

Name	Value	0 ns	2	0 ns	40	ns	60 n	s		80 ns	أحبينا	100 ns		120 ns		140	ıs	160 ns		180 ns	2
▶ ■ a[7:0]	0			0			\Rightarrow				1						127			Х	
▶ ■ b[7:0]	0	0 (1	127	.28	127 \ -1			1	127	128	-127 X	_1	0	X 1	12	7 \ -128	-127) 1	$\langle 0 \rangle$	
prod[15:0]	0				0			$=$ X \subset	1	127	128	-127 X	_1	0	12	7 161	29 \(-162!	6 -1612	-127	$\langle 0 \rangle$	-128
la⊓ n	8												8								
200 ns	220 ns		240 ns		260 ns		280 ns	5 .		300 ns		320 ns		34	0 ns		360 ns		380 ns		400 ns
-128						127							سبك								
			↑						=	=			-1	= =			_			0	
127 / 128	-127	X -1	0	<u> </u>	127	-128	-127	_X	-1	0	X 1	127		<u>3</u> _X_	127	-1	0	X <u>1</u>	127	128	-127
16256 16384	16256	128	0	X -127	-16129	16256	16129	9 X 1	27	0	X	-127	X 128	=X $=$	127	1				0	
										8				-							
																					1
	.430		440				.400			500		.530		.5.4			560		.500		.600
400 ns	420 ns	1	440 ns	1	460 ns	1	480 ns			500 ns		520 ns		54	0 ns		560 ns	<u> </u>	580 ns	<u> </u>	600 ns
0	*			1						1	27			_*			-1	28			12
127 / 1	¥ 0	χ 1	127	-128	-127	χ 1	0	-χ	1	127	-128	-127	χ -1	-∤-	0 X	1	127	-128	-127	χ 1	X 0
0		χ <u> </u>	127	X -128	-127	χī	0	χ 1	27	16129	-16256	-16129	X -127	-X-	0 X	-128	-16256	16384	16256	128	X 0
										8											

	600 ns	1	620 ns	1	640 ns	1	660 ns	1	680 ns	1,,,,,	700 ns	1	720 ns		740 ns		760 ns		780 ns		800 ns
D			-1	.27			Ж			-1			Ж			0			Х		1
Ď	0	1	127	-128	-127	X 1	0	X 1	127	-128	-127	\ 1	0	1	127	-128	-127	\ -1	0	\ 1	127
D	0	-127	-16129	16256	16129	127	X 0	X _ I	-127	128	127	X 1	\mathbf{x}			0				XΙ	127
Ī												8									
_																					
ľ	300 ns		820 ns		840 ns		860 ns		880 ns		900 ns		920 ns		940 ns		960 ns		980 ns		1,000 ns
		1					13	27					-1	28				-12	?7		
X	127	-128	-127		0	(1	127	-128	-127	-1	0	1	127	-128	-127		0		127	-128	
Х	127	-128	-127		0	127	16129	(-16256)	(-16129)	-127	0	-128	-16256	16384	16256	128	0	-127	-16129	16256	
										8											

Conclusion:

Signed and unsigned multipliers were successfully implemented in this practical.