

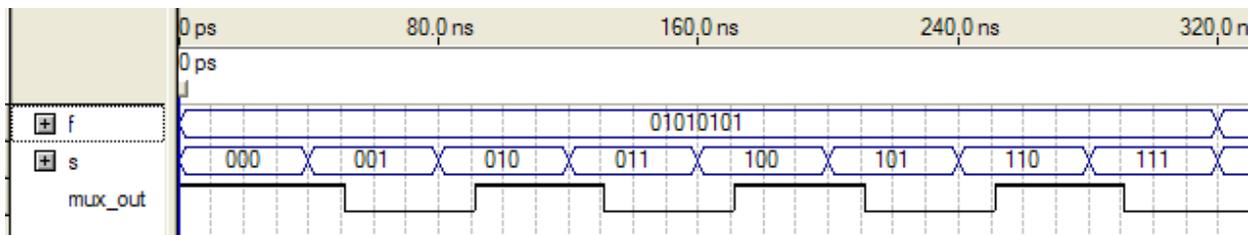
## EC262 Problem Set 9 Due: Friday 5 October 2012

**Exercise 1:** Implement an 8-to-1 multiplexer using logic operators (**Boolean expression**) in VHDL.

- a. Implement this circuit using Quartus II software. Attach your VHDL code.

```
1  -- 8-1 Multiplexer
2
3  -- Library declaration
4  library IEEE;                -- Declare which VHDL library
5  use IEEE.std_logic_1164.all; -- and packages to use
6
7  -- Entity declaration
8  ENTITY mux_8_1 IS
9  PORT (
10     -- Input ports
11     s      : IN STD_LOGIC_VECTOR (2 downto 0) ;
12     f      : IN STD_LOGIC_VECTOR (7 downto 0) ;
13
14     -- Output port
15     mux_out : OUT STD_LOGIC
16 );
17 END mux_8_1 ;
18
19 --Architecture body
20 ARCHITECTURE dataflow OF mux_8_1 IS
21 BEGIN
22
23     -- Implement function 8-to-1 mutiplexer;
24     mux_out <= ( f(0) AND NOT s(2) AND NOT s(1) AND NOT s(0) ) OR
25               ( f(1) AND NOT s(2) AND NOT s(1) AND     s(0) ) OR
26               ( f(2) AND NOT s(2) AND     s(1) AND NOT s(0) ) OR
27               ( f(3) AND NOT s(2) AND     s(1) AND     s(0) ) OR
28               ( f(4) AND     s(2) AND NOT s(1) AND NOT s(0) ) OR
29               ( f(5) AND     s(2) AND NOT s(1) AND     s(0) ) OR
30               ( f(6) AND     s(2) AND     s(1) AND NOT s(0) ) OR
31               ( f(7) AND     s(2) AND     s(1) AND     s(0) );
32
33 END dataflow ;
```

- b. Show a Quartus functional simulation to verify the correct operations of the circuit.



**Exercise 2:** Implement a 4-bit Binary-to-Gray code converter using a **WITH/SELECT/WHEN/ statement**. The binary encoded numbers and the equivalent Gray encoded numbers are shown in Table 1. The Gray code is based on the Hamming distance between 2 consecutive numbers, that is, only one bit changes between them. The circuit accepts a 4-bit binary encoded number and produces its equivalent number in Gray code.

Table 1. Binary encoded and Gray encoded numbers.

| Decimal number | Binary code | Gray code |
|----------------|-------------|-----------|
| 0              | 0000        | 0000      |
| 1              | 0001        | 0001      |
| 2              | 0010        | 0011      |
| 3              | 0011        | 0010      |
| 4              | 0100        | 0110      |
| 5              | 0101        | 0111      |
| 6              | 0110        | 0101      |
| 7              | 0111        | 0100      |
| 8              | 1000        | 1100      |
| 9              | 1001        | 1101      |
| 10             | 1010        | 1111      |
| 11             | 1011        | 1110      |
| 12             | 1100        | 1010      |
| 13             | 1101        | 1011      |
| 14             | 1110        | 1001      |
| 15             | 1111        | 1000      |

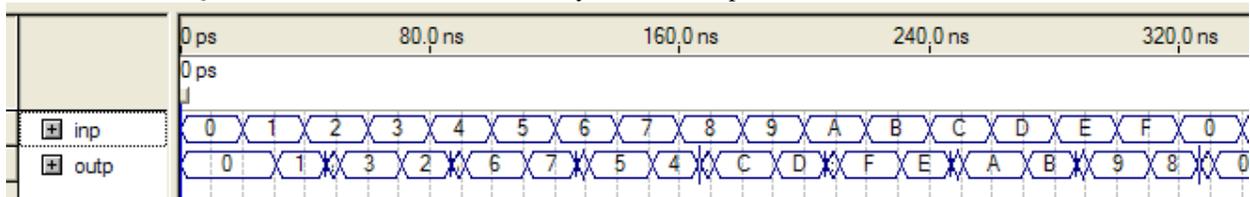
- a. Implement this circuit using Quartus II software. Attach your VHDL code.

```

1
2  -- Binary to Gray code converter
3  -- Library declaration
4
5  library IEEE;                -- Declare which VHDL library
6  use IEEE.std_logic_1164.all; -- and packages to use
7
8  -- Entity declaration
9  ENTITY Bin_2_Gray IS
10  PORT (
11
12      -- Input ports
13      inp          : IN BIT_VECTOR ( 3 downto 0 ) ;
14
15      -- Output port
16      outp         : OUT BIT_VECTOR ( 3 DOWNTO 0 )
17
18  ) ;
19  END Bin_2_Gray ;
20
21  --Architecture body
22  ARCHITECTURE dataflow OF Bin_2_Gray IS
23  BEGIN
24
25      -- Implement a Binary to Gray converter
26      -- Use WITH/SELECT/WHEN
27
28      WITH inp SELECT
29      outp  <=  "0000" WHEN "0000",
30                "0001" WHEN "0001",
31                "0011" WHEN "0010",
32                "0010" WHEN "0011",
33                "0110" WHEN "0100",
34                "0111" WHEN "0101",
35                "0101" WHEN "0110",
36                "0100" WHEN "0111",
37                "1100" WHEN "1000",
38                "1101" WHEN "1001",
39                "1111" WHEN "1010",
40                "1110" WHEN "1011",
41                "1010" WHEN "1100",
42                "1011" WHEN "1101",
43                "1001" WHEN "1110",
44                "1000" WHEN OTHERS;
45
46  END dataflow ;

```

b. Show a Quartus functional simulation to verify the correct operations of the circuit.



**Exercise 3:** Implement a 7-level priority encoder using a **WHEN/ELSE statement**. The block diagram for a 7-level encoder is shown in Figure 1. The circuit must encode the position of the input bit of the highest order that is active. For example, if the input bits are “0101011”, then the output bits should be “110” to indicate that 6 is the highest bit position that is active. “000” should indicate that there is no active bit.

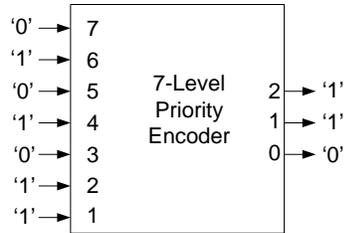


Figure 1. An example of 7-level priority encoder.

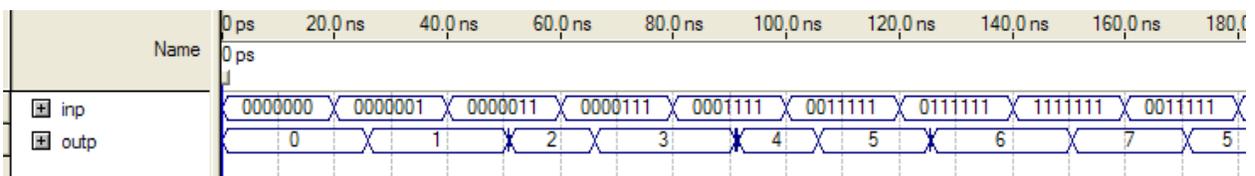
- a. Implement this circuit using Quartus II software. Attach your VHDL code.

```

2  -- 7-level priority encoder
3  -- Library declaration
4
5  library IEEE;                -- Declare which VHDL library
6  use IEEE.std_logic_1164.all; -- and packages to use
7
8  -- Entity declaration
9  ENTITY Priority_Encoder IS
10  PORT (
11
12      -- Input ports
13      inp          : IN BIT_VECTOR ( 6 downto 0 ) ;
14
15      -- Output port
16      outp         : OUT BIT_VECTOR ( 2 DOWNTO 0 )
17
18  ) ;
19  END Priority_Encoder ;
20
21  --Architecture body
22  ARCHITECTURE dataflow OF Priority_Encoder IS
23  BEGIN
24
25  -- Implement a 7-level priority encoder
26  -- output = address of the input bit of highest order that is active.
27  -- "000" indicates no bit active
28  -- For example: if inp = "0100110", outp should be "110"
29  -- Use WHEN/ELSE
30
31      outp  <=  "111" WHEN inp(6) = '1' ELSE
32                "110" WHEN inp(5) = '1' ELSE
33                "101" WHEN inp(4) = '1' ELSE
34                "100" WHEN inp(3) = '1' ELSE
35                "011" WHEN inp(2) = '1' ELSE
36                "010" WHEN inp(1) = '1' ELSE
37                "001" WHEN inp(0) = '1' ELSE
38                "000";
39
40  END dataflow ;

```

b. Show a Quartus functional simulation to verify the correct operations of the circuit.



**Exercise 4:** Implement a simple Barrel Shifter using a **GENERATE statement**. The block diagram for a simple Barrel Shifter is shown in Figure 2. The circuit must shift the input vector (8 bits) either 0 or 1 position to the left based on a controller signal, *shift*. When shifted by 1 position (*shift* = '1'), the least significant bit must be filled with '0' (shown in Figure 2). If *shift* = 0, then *outp* = *inp* (shifted by 0 position).

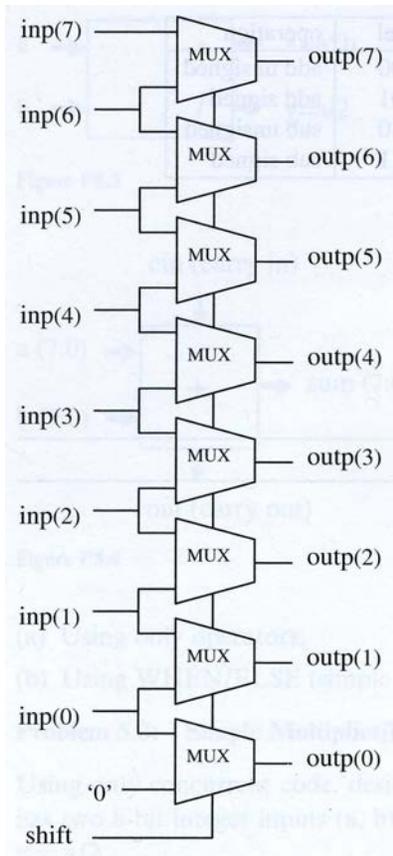


Figure 2. A block diagram of a simple Barrel Shifter (from Pedroni, *Circuit Design with VHDL*, 2<sup>nd</sup> ed.)

- a. Implement this circuit using Quartus II software. Attach your VHDL code.

```

1
2  -- Simple Barrel Shifter
3  -- Library declaration
4
5  library IEEE;           -- Declare which VHDL library
6  use IEEE.std_logic_1164.all; -- and packages to use
7
8  -- Entity declaration
9  ENTITY Barrel_Shifter IS
10  PORT(
11
12     -- Input ports
13     inp          : IN BIT_VECTOR ( 7 downto 0 ) ;
14     shift        : IN BIT;
15
16     -- Output port
17     outp         : OUT BIT_VECTOR ( 7 DOWNTO 0 )
18
19  ) ;
20  END Barrel_Shifter ;
21
22  --Architecture body
23  ARCHITECTURE dataflow OF Barrel_Shifter IS
24  BEGIN
25
26  -- Implement a simple Barrel Shifter
27  -- If shift = 1 --> shift left 1 bit
28  -- If shift = 0 --> shift left 0 bit
29  -- Use GENERATE
30
31     mux_gen:
32     FOR i IN 0 TO 7 GENERATE
33
34         mux_0:
35         IF (i = 0) GENERATE
36             outp(i) <= inp(i) WHEN shift = '0' ELSE '0';
37         END GENERATE;
38
39         mux_1_to_7:
40         IF (i /= 0) GENERATE
41             outp(i) <= inp(i) WHEN shift = '0' ELSE inp(i-1);
42         END GENERATE;
43
44         END GENERATE;
45
46  END dataflow ;

```

b. Show a Quartus functional simulation to verify the correct operations of the circuit.

