1 Binary to Decimal Converter

Due to the low complexity and purely combinational nature of the problem, the component can quickly be implemented directly using gate-level logic operators rather than any higher level code constructs. This is shown in the code below.

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity bin2dec is
5      port( SW : in std_logic_vector(3 downto 0);
6            LEDR: out std_logic_vector(9 downto 0);
7            LEDG7: out std_logic);
8  end entity;
9
10 architecture behv of bin2dec is
11 begin
12         LEDR(0) <= NOT SW(3) AND NOT SW(2) AND NOT SW(1) AND NOT SW(0);
13         LEDR(1) <= NOT SW(3) AND NOT SW(2) AND NOT SW(1)   AND SW(0);
14         LEDR(2) <= NOT SW(3) AND NOT SW(2) AND SW(1) AND NOT SW(0);
15         LEDR(3) <= NOT SW(3) AND NOT SW(2) AND SW(1) AND SW(0);
16         LEDR(4) <= NOT SW(3) AND SW(2) AND NOT SW(1) AND NOT SW(0);
17         LEDR(5) <= NOT SW(3) AND SW(2) AND NOT SW(1) AND SW(0);
18         LEDR(6) <= NOT SW(3) AND SW(2) AND SW(1) AND NOT SW(0);
19         LEDR(7) <= NOT SW(3) AND SW(2) AND SW(1) AND SW(0);
20         LEDR(8) <= SW(3) AND NOT  SW(2) AND NOT SW(1) AND NOT SW(0);
21         LEDR(9) <= SW(3) AND NOT SW(2) AND NOT SW(1) AND SW(0);
22
23         LEDG7 <= SW(3) AND (SW(2) OR SW(1));
24 end architecture;
```

The code was simulated and the output was evaluated using an incrementing binary input. As shown in figure 1, When the input is 9 (or $(1001)_2$), the output is a high signal on LEDR(9) and a low signal on the rest.

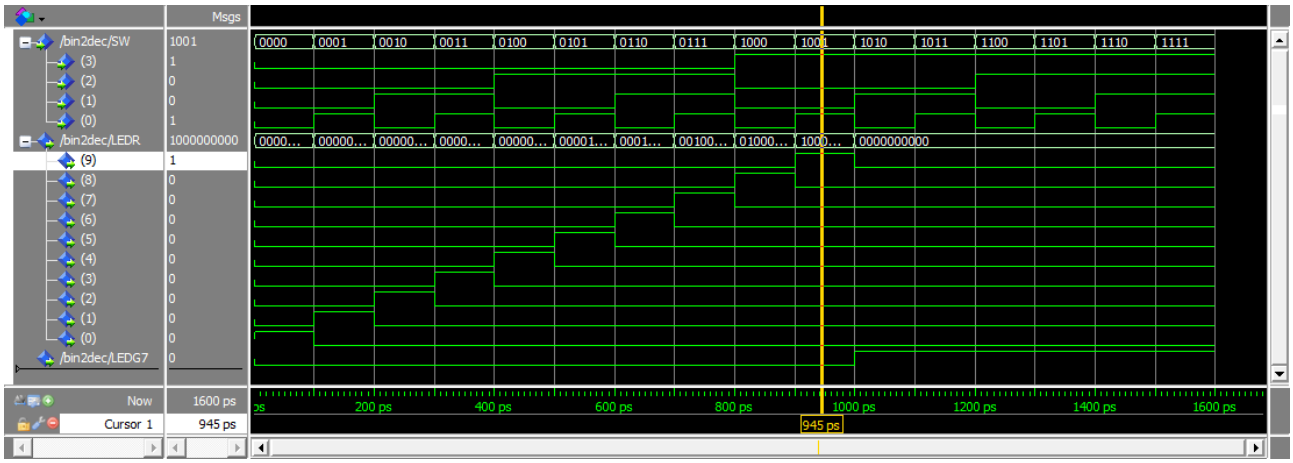

Figure 1: *Waveforms showing switch inputs and corresponding LED outputs*

2 7-Segment Encoder

Like the first problem, this component is purely combinational, though rather more complex and not as easy to implement using gate-level logic. Instead, a *when/else* statement is used, as shown in the source code below, making the program quicker to write and easier to understand.

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity sevenseg_enc is
5      port(SW : in std_logic_vector(3 downto 0);
6           HEX0 : out std_logic_vector(6 downto 0));
7  end entity;
8
9  architecture behv of sevenseg_enc is
10 begin
11         HEX0 <= "1000000" when SW="0000" else  --DISPLAY 0
12                 "1111001" when SW="0001" else  --DISPLAY 1
13                 "0100100" when SW="0010" else  --DISPLAY 2
14                 "0110000" when SW="0011" else  --DISPLAY 3
15                 "0011001" when SW="0100" else  --DISPLAY 4
16                 "0010010" when SW="0101" else  --DISPLAY 5
17                 "0000011" when SW="0110" else  --DISPLAY 6
18                 "1111000" when SW="0111" else  --DISPLAY 7
19                 "0000000" when SW="1000" else  --DISPLAY 8
20                 "0011000" when SW="1001" else  --DISPLAY 9
21                 "0100111" when SW="1010" else  --DISPLAY 10
22                 "0110011" when SW="1011" else  --DISPLAY 11
23                 "0011101" when SW="1100" else  --DISPLAY 12
24                 "0010110" when SW="1101" else  --DISPLAY 13
25                 "0000111" when SW="1110" else  --DISPLAY 14
26                 "1111111" when SW="1111";      --DISPLAY 15
27 end architecture;
```

One extra thing to consider in this particular circuit is that the displays on the DE1 board have a common anode configuration and are thus active low, in other words they use inverted logic. Table 1 shows the exact input and output signals for each display state.

Table 1: 7-segment display signals as functions of the input switch signals.

Display	SW3-SW0				HEX0[6]-HEX0[0]						
0	0	0	0	0	1	0	0	0	0	0	0
1	0	0	0	1	1	1	1	1	0	0	1
2	0	0	1	0	0	1	0	0	1	0	0
3	0	0	1	1	0	1	1	0	0	0	0
4	0	1	0	0	0	0	1	1	0	0	1
5	0	1	0	1	0	0	1	0	0	1	0
6	0	1	1	0	0	0	0	0	0	1	1
7	0	1	1	1	1	1	1	1	0	0	0
8	1	0	0	0	0	0	0	0	0	0	0
9	1	0	0	1	0	0	1	1	0	0	0
10	1	0	1	0	0	1	0	0	1	1	1
11	1	0	1	1	0	1	1	0	0	1	1
12	1	1	0	0	0	0	1	1	1	0	1
13	1	1	0	1	0	0	1	0	1	1	0
14	1	1	1	0	0	0	0	0	1	1	1
15	1	1	1	1	1	1	1	1	1	1	1

The pin configuration needed by the FPGA on the DE1 board is given in Table 2.

Table 2: Pin configuration.

Name	**Location**	**Enabled**	**I/O Standard**
SW[0]	PIN_L22	Yes	LVTTL
SW[1]	PIN_L21	Yes	LVTTL
SW[2]	PIN_M22	Yes	LVTTL
SW[3]	PIN_V12	Yes	LVTTL
HEX0[0]	PIN_J2	Yes	LVTTL
HEX0[1]	PIN_J1	Yes	LVTTL
HEX0[2]	PIN_H2	Yes	LVTTL
HEX0[3]	PIN_H1	Yes	LVTTL
HEX0[4]	PIN_F2	Yes	LVTTL
HEX0[5]	PIN_F1	Yes	LVTTL
HEX0[6]	PIN_E2	Yes	LVTTL