

# Lab 3

## Decoder/Encoder

Jeff Morrison  
CSCE 3730  
11/4/08

## **Description**

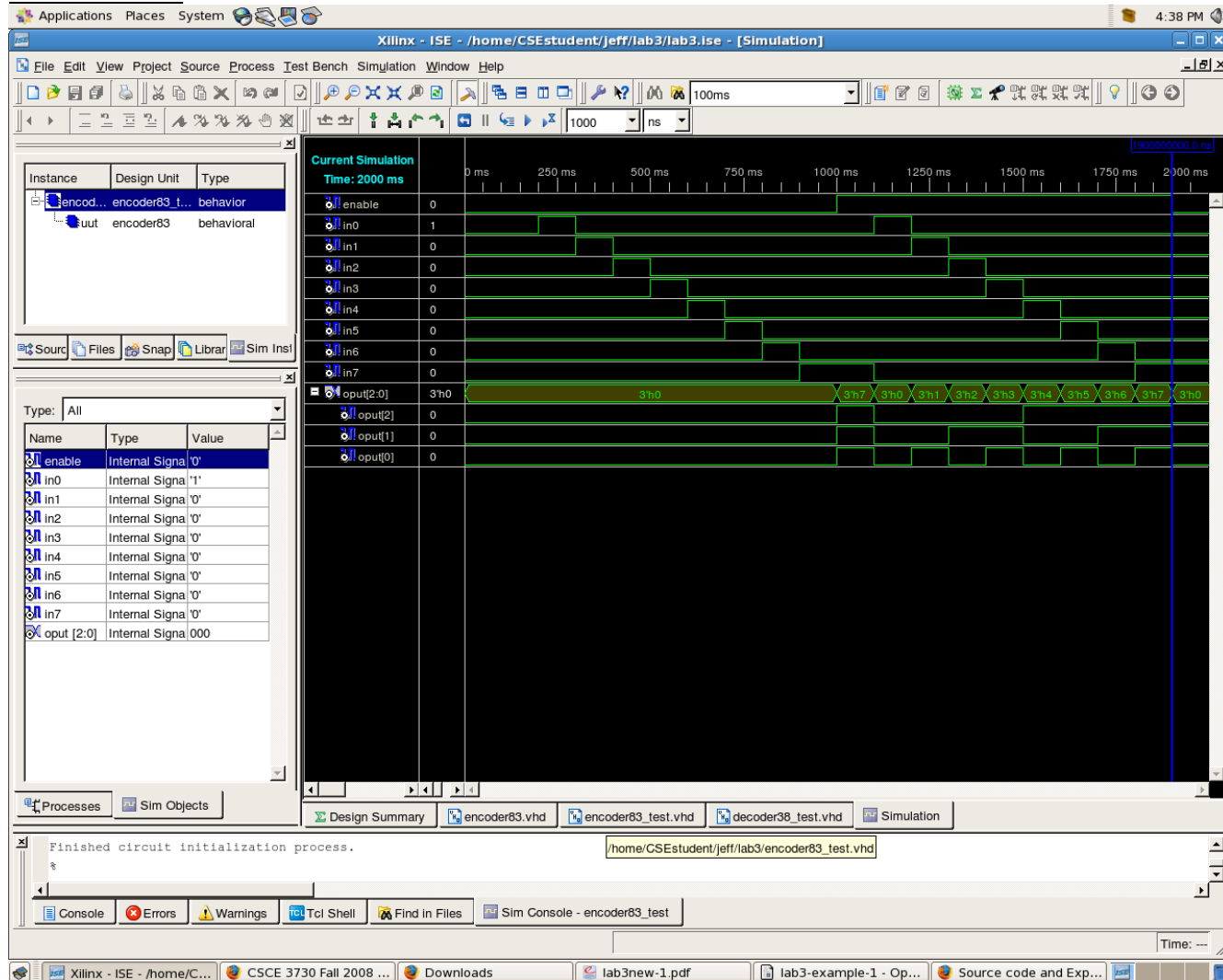
The purpose of this project was to design an 8-3 octal-to-binary encoder and a 3-8 binary-to-octal decoder using VHDL.

## **Analysis**

It is fairly simple to design the decoder/encoder project using VHDL. When we did the same project using schematics, there was much more processing involved on the part of the computer to be able to synthesize. VHDL allows us to make simple or broad changes to our design very quickly and easily based on a coding language.

# Waveforms

## Encoder 8-to-3



# Decoder 3-to-8

The screenshot shows the Xilinx ISE simulation environment for a 3-to-8 decoder. The main window displays a timing diagram for signals enable, inp[2:0], and outputs f0 through f7. The enable signal is high from 1000ms to 2000ms. The inp[2:0] signal is a 3-bit binary counter from 3'h0 to 3'h7. The outputs f0-f7 are active-low signals that are low when the corresponding input combination is present.

Signal	Value
enable	0
inp [2:0]	3'h0
f0	0
f1	0
f2	0
f3	0
f4	0
f5	0
f6	0
f7	0

The console window shows the message: "Finished circuit initialization process."

## Source

### Encoder

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity encoder83 is
    Port ( Enable : in  STD_LOGIC;
          in0  : in  STD_LOGIC;
          in1  : in  STD_LOGIC;
          in2  : in  STD_LOGIC;
          in3  : in  STD_LOGIC;
          in4  : in  STD_LOGIC;
          in5  : in  STD_LOGIC;
          in6  : in  STD_LOGIC;
          in7  : in  STD_LOGIC;
          oput : out  STD_LOGIC_VECTOR (2 downto 0));
end encoder83;

architecture Behavioral of encoder83 is
    signal inputs : std_logic_vector(7 downto 0); -- Defines internal signals
begin

    inputs <= in7 & in6 & in5 & in4 & in3 & in2 & in1 & in0;
    process(Enable, inputs) begin

        if (Enable = '1') then

            case inputs is

                when "00000001" =>
                    oput<="000";

                when "00000010" =>
                    oput<="001";

                when "00000100" =>
                    oput<="010";

                when "00001000" =>
                    oput<="011";
                when "00010000" =>
                    oput<="100";
                when "00100000" =>
                    oput<="101";
                when "01000000" =>
                    oput<="110";
                when "10000000" =>
                    oput<="111";
                when others =>
                    oput<="000";

            end case;
        else
            oput<="000";
        end if;
    end process;
end Behavioral;
```

## Decoder

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity decoder38 is
    Port ( Enable: in STD_LOGIC;
          inp : in  STD_LOGIC_VECTOR (2 downto 0);
          f0 : out  STD_LOGIC;
          f1 : out  STD_LOGIC;
          f2 : out  STD_LOGIC;
          f3 : out  STD_LOGIC;
          f4 : out  STD_LOGIC;
          f5 : out  STD_LOGIC;
          f6 : out  STD_LOGIC;
          f7 : out  STD_LOGIC);
end decoder38;

architecture Behavioral of decoder38 is

begin

process(inp,Enable)
begin

if (Enable = '1') then

    case inp is

        when "000" =>
            f0<='1';
            f1<='0';
            f2<='0';
            f3<= '0';
            f4<='0';
            f5<= '0';
            f6<= '0';
            f7<='0';

        when "001" =>
            f0<='0';
            f1<='1';
            f2<='0';
            f3<= '0';
            f4<='0';
            f5<= '0';
            f6<= '0';
            f7<='0';

        when "010" =>
            f0<='0';
            f1<='0';
            f2<='1';
            f3<= '0';
            f4<='0';
            f5<= '0';
            f6<= '0';
            f7<='0';

    end case;

end if;

end process;

end Behavioral;
```

```
when "011" =>
f0<='0';
f1<='0';
f2<='0';
f3<= '1';
f4<='0';
f5<= '0';
f6<= '0';
f7<='0';
```

```
when "100" =>
f0<='0';
f1<='0';
f2<='0';
f3<= '0';
f4<='1';
f5<= '0';
f6<= '0';
f7<='0';
```

```
when "101" =>
f0<='0';
f1<='0';
f2<='0';
f3<= '0';
f4<='0';
f5<= '1';
f6<= '0';
f7<='0';
```

```
when "110" =>
f0<='0';
f1<='0';
f2<='0';
f3<= '0';
f4<='0';
f5<= '0';
f6<= '1';
f7<='0';
```

```
when "111" =>
f0<='0';
f1<='0';
f2<='0';
f3<= '0';
f4<='0';
f5<= '0';
f6<= '0';
f7<='1';
```

```
when others =>
f0<='0';
f1<='0';
f2<='0';
f3<= '0';
f4<='0';
f5<= '0';
f6<= '0';
f7<='0';
```

```
end case;
```

```
else
```

```
f0<='0';  
f1<='0';  
f2<='0';  
f3<= '0';  
f4<='0';  
f5<= '0';  
f6<= '0';  
f7<='0';
```

```
end if;  
end process;  
end Behavioral;
```