SECTION D2: VHDL BASED STAND ALONE AND MODULAR DESIGNS

Ms. Mildred C. Zabawa & Mr. Vivek Jayaram Research Associates

Dr. Subbarao V. Wunnava, Professor

Florida International University
Department of Electrical and Computer Engineering
10555 West Flagler Street, Miami, FL 33174
Fall 2005: Updated Summer 2006

In this Section D2, very important aspects of VLSI system design, namely the stand along designs and modular designs have been introduced, using VHDL. Stand alone designs are extremely useful for gluing several different forms of digital circuits. Modular designs are very useful in expanding the size and scope of the stand alone designs. Also, the resource management is an important factor emphasized in Uyemura's book. In this section, using the Mentor Graphics tools, the gate level resources used in stand alone and modular designs have been presented. The information in this Section D3, will provide additional VLSI resource management capabilities to the students.

SECTION D2: VHDL BASED STAND ALONE AND MODULAR DESIGNS

Table of Contents **Page**

1 Objective

Two distinct approaches are being practiced in today's VLSI design industries: *stand-alone systems* and *modular systems*. Stand-alone systems are modules that are independent of other instantiate existing modules. Modular systems instantiate other existing modules for complex operations. Depending on the feature list requirements of the system being designed as well as the associated power and layout limiting criteria we can use either technique. Our objective is to design 4-bit and 8-bit systems in stand-alone and modular systems for *multiplexers, decoders, shift registers, rotate registers, static ram*, and *arithmetic logical units* (ALU). We will introduce each of the above system designs as well as our techniques, implementation, comparison of power and layout advantages and disadvantages.

2 Multiplexers

A Multiplexer (MUX) contains n-inputs lines and one output line. MUXs are highly used components which are often used in most designs in industry. We have design two different techniques shown below:

4-to-1 MUX: Stand-Alone System

The stand-alone system is based on behavior methodology built using case statements. The table below consists of the modules input/output ports:

Port	Type	Bit Width	Description
Sel(1:0)	In	2	Select signal If Sel = "00" select input0 If Sel = "01" select input1 If Sel = "10" select input2 If Sel = "11" select input3
inp0	In	1	Input0 signal
inp1	In	1	Input1 signal
inp2	In	1	Input2 signal
inp3	In	1	Input3 signal
mux_out	Out	1	MUX Output signal selected

Table 2.1 Input/Output Port signals for the 4-to-1 Multiplexer

The following is the VHDL code for the stand-alone module of the 4-to-1 Multiplexer:

```
-------------------------------------------------------------------------
--
-- Title    : VHDL style
-- File name      : mux_4_1.vhd
-- Authors  : Mildred C. Zabawa and Vivek Jayaram
-- Description    : This module is a stand-alone 4 to 1 Multiplexer.
-------------------------------------------------------------------------

library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.Numeric_Std.all;

entity mux_4_1 is
  port (Sel     : in unsigned ( 1 downto 0);
        inp0    : in std_logic;
        inp1    : in std_logic;
        inp2    : in std_logic;
        inp3    : in std_logic;
        mux_out : out std_logic);
end mux_4_1;

architecture stand_alone_mux4_1 of mux_4_1 is
begin

process (Sel, inp0, inp1, inp2, inp3)
begin
   case Sel is
       when "00"   => mux_out <= inp0;
       when "01"   => mux_out <= inp1;
       when "10"   => mux_out <= inp2;
       when "11"   => mux_out <= inp3;
       when others => mux_out <= inp0;
   end case;
end process;

end stand_alone_mux4_1;
```

Figure 2.1 VHDL Code for the 4-to-1 Multiplexer Stand-Alone System

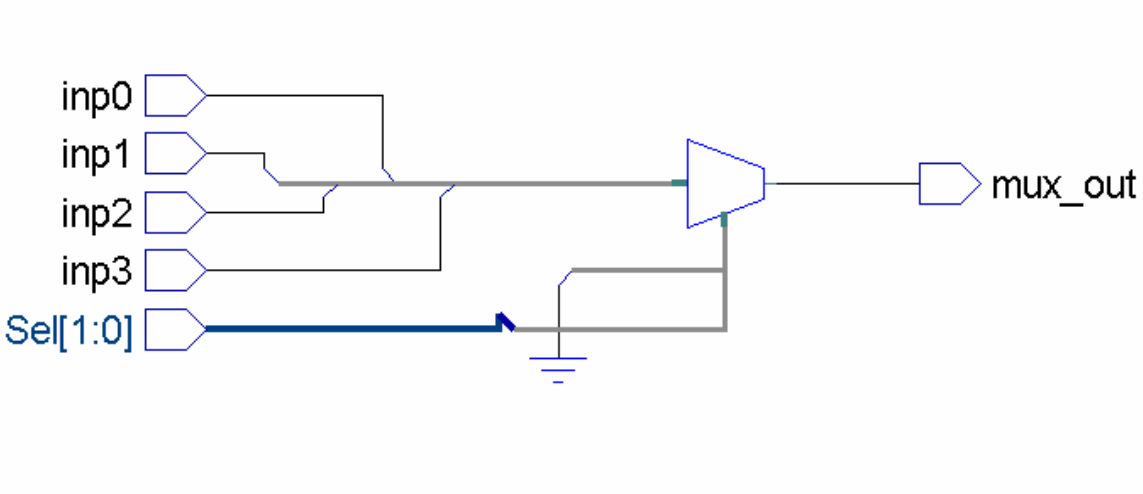

Figure 2.2 System Diagram of the 4-to-1 Multiplexer Stand-Alone System

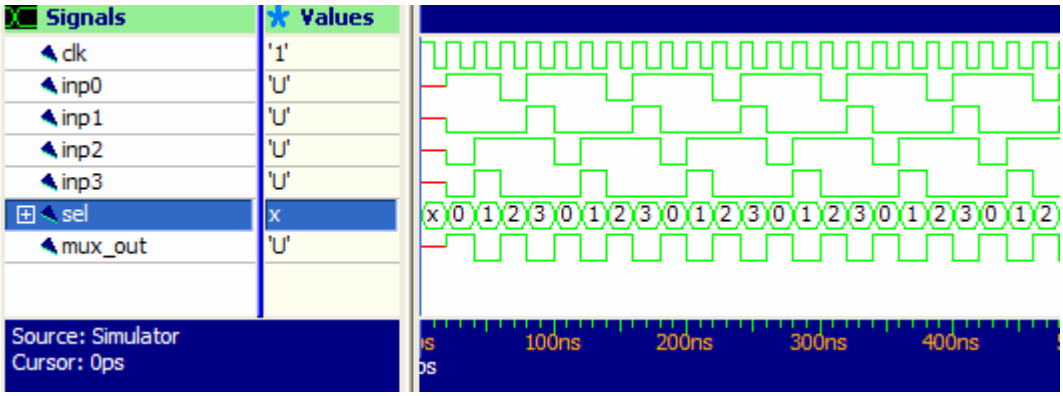

Figure 2.3 Timing Diagram of 4-to-1 Multiplexer (Stand Alone Module)

The total layout area for the 4-to-1 MUX was generated using Mentor Graphics Leonardo Synthesis Tools. The following are the results generated based on our synthesis module:

```
 ******************************************************

Cell: mux_4_1    View: stand_alone_mux4_1    Library: work

******************************************************

Total accumulated area :
 Number of IOs :                          7
 Number of LCs :                          3
 Number of accumulated instances :       10
 Number of ports :                        7
 Number of nets :                        16
 Number of instances :                   10
 Number of references to this view :      0


 Cell                   Library  References     Total Area

cyclone_io_input        cyclone     6 x      1      6 IOs
cyclone_io_output       cyclone     1 x      1      1 IOs
cyclone_lcell_normal    cyclone     3 x      1      3 LCs

*********************************************
Device Utilization for EP1C3T100C
*********************************************
Resource                Used    Avail   Utilization
---------------------------------------------
IOs                     7       117       5.98%
LCs                     3       2910      0.10%
Memory Bits             0       53248     0.00%

---------------------------------------------
Using default wire table: cyclone_default
```

8-to-1 MUX: Stand-Alone System

The stand-alone system is based on behavior methodology built using case statements. The table below consists of the modules input/output ports:

Port	Type	Bit Width	Description
Sel(2:0)	In	3	Select signal If Sel = “000” select input0 If Sel = “001” select input1 If Sel = “010” select input2 If Sel = “011” select input3 If Sel = “100” select input4 If Sel = “101” select input5 If Sel = “110” select input6 If Sel = “111” select input7
inp0	In	1	Input0 signal
inp1	In	1	Input1 signal
inp2	In	1	Input2 signal
inp3	In	1	Input3 signal
inp4	In	1	Input4 signal
inp5	In	1	Input5 signal
inp6	In	1	Input6 signal
inp7	In	1	Input7 signal
mux_out	Out	1	MUX Output signal selected

Table 2.2 Input/Output Port signals for the 8-to-1 Multiplexer

```
--------------------------------------------------------------------------
-- Title    : VHDL style
-- File name      : mux_8_1.vhd
-- Authors  : Mildred C. Zabawa and Vivek Jayaram
-- Description    : This module is a stand-alone 8 to 1 Multiplexer.
--------------------------------------------------------------------------
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.Numeric_Std.all;

entity mux_8_1 is
  port (Sel: in unsigned ( 2 downto 0);
        inp0     : in std_logic;
        inp1     : in std_logic;
        inp2     : in std_logic;
        inp3     : in std_logic;
        inp4     : in std_logic;
        inp5     : in std_logic;
        inp6     : in std_logic;
        inp7     : in std_logic;
        mux_out  : out std_logic);
end mux_8_1;

architecture data_flow of mux_8_1 is
begin

process (Sel, inp0, inp1, inp2, inp3, inp4, inp5, inp6, inp7)
begin
   case Sel is
       when "000" => mux_out  <= inp0;
       when "001" => mux_out  <= inp1;
       when "010" => mux_out  <= inp2;
       when "011" => mux_out  <= inp3;
       when "100" => mux_out  <= inp4;
       when "101" => mux_out  <= inp5;
       when "110" => mux_out  <= inp6;
       when "111" => mux_out  <= inp7;
       when others => mux_out <= inp0;
   end case;
end process;

end data_flow;
```

Figure 2.4 VHDL Code for the 8-to-1 Multiplexer Stand-Alone System

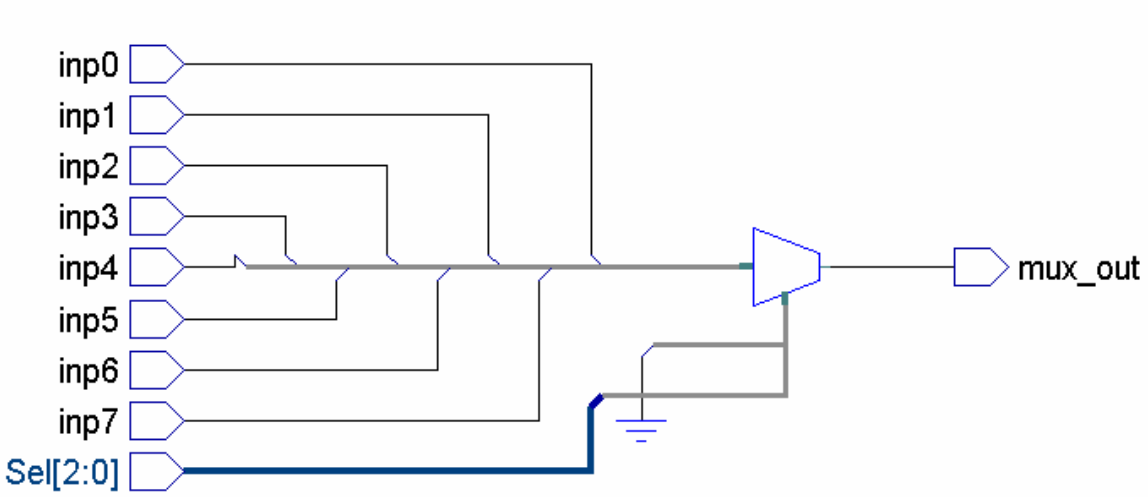

Figure 2.6 System Diagram of the 8-to-1 Multiplexer Stand-Alone System

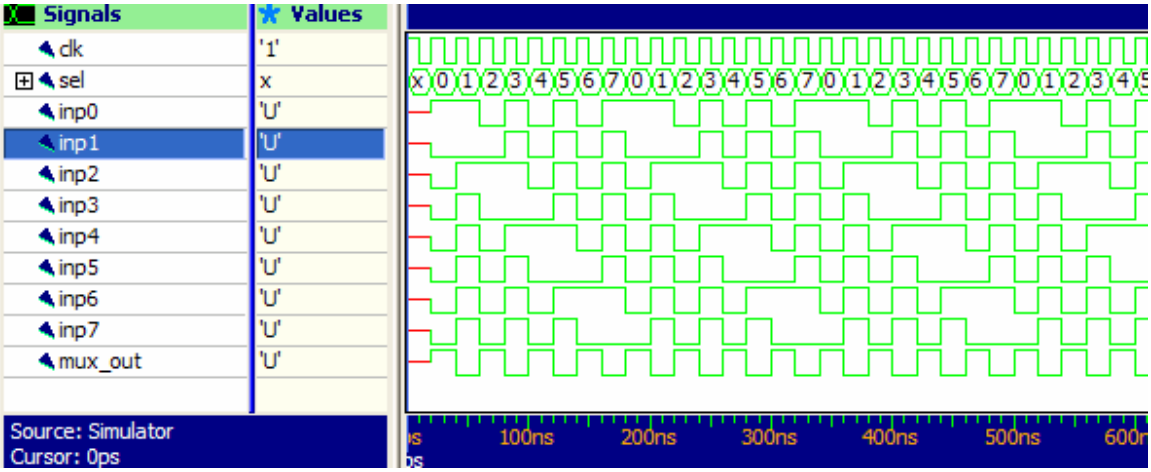

Figure 2.7 Timing Diagram of 8-to-1 Multiplexer (Stand Alone Module)

The total layout area for the 8-to-1 MUX was generated using Mentor Graphics Leonardo Synthesis Tools. The following are the results generated based on our synthesis module:

```
*******************************************************

Cell: mux_8_1    View: data_flow    Library: work

*******************************************************

Total accumulated area :
 Number of IOs :                          12
 Number of LCs :                           6
 Number of accumulated instances :        18
 Number of ports :                        12
 Number of nets :                         29
 Number of instances :                    18
 Number of references to this view :       0


 Cell                   Library  References     Total Area

cyclone_io_input        cyclone    11 x      1     11 IOs
cyclone_io_output       cyclone     1 x      1      1 IOs
cyclone_lcell_normal    cyclone     6 x      1      6 LCs

*********************************************
Device Utilization for EP1C3T100C
*********************************************
Resource                Used    Avail   Utilization
---------------------------------------------
IOs                     12      117      10.26%
LCs                     6       2910      0.21%
Memory Bits             0       53248     0.00%

---------------------------------------------
Using default wire table: cyclone_default
```

4-to-1 MUX: Modular System

The modular system is based on behavior methodology built using select statements. The table below consists of the modules input/output ports:

Port	Type	Bit Width	Description
Sel(1:0)	In	2	Select signal If Sel = "00" select input0 If Sel = "01" select input1 If Sel = "10" select input2 If Sel = "11" select input3
Inputs(3:0)	In	4	Input signal bit(0): Input0 signal bit(1): Input1 signal bit(2): Input2 signal bit(3): Input3 signal
mux_out	Out	1	MUX Output signal selected

Table 2.3 Input/Output Port signals for the 4-to-1 Multiplexer

```
-------------------------------------------------------------------------
---
-- Title    : VHDL style
-- File name      : mux4.vhd
-- Authors  : Mildred C. Zabawa and Vivek Jayaram
-- Description    : This module is the basic module of a 4 to 1
--                  Multiplexer.
--                This unit can be the bases of the expanded modules.
-------------------------------------------------------------------------
---
library IEEE;
use ieee.std_logic_1164.all;

entity mux4 is
port (Sel      : in  std_logic_vector(1 downto 0);
      inputs   : in  std_logic_vector(3 downto 0);
      mux4_out : out std_logic);
end mux4;

architecture modular_mux4 of mux4 is
begin

with Sel select
   mux4_out <= inputs(0) when "00",    --Input0 selected
               inputs(1) when "01",    --Input1 selected
               inputs(2) when "10",    --Input2 selected
               inputs(3) when  others; --Input3 selected

end modular_mux4;
```

Figure 2.8 VHDL Code for the 4-to-1 Multiplexer Modular System

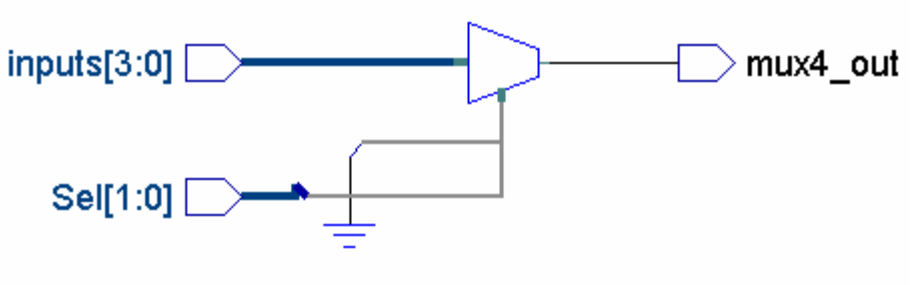

Figure 2.9 System Diagram of the 4-to-1 Multiplexer Modular System

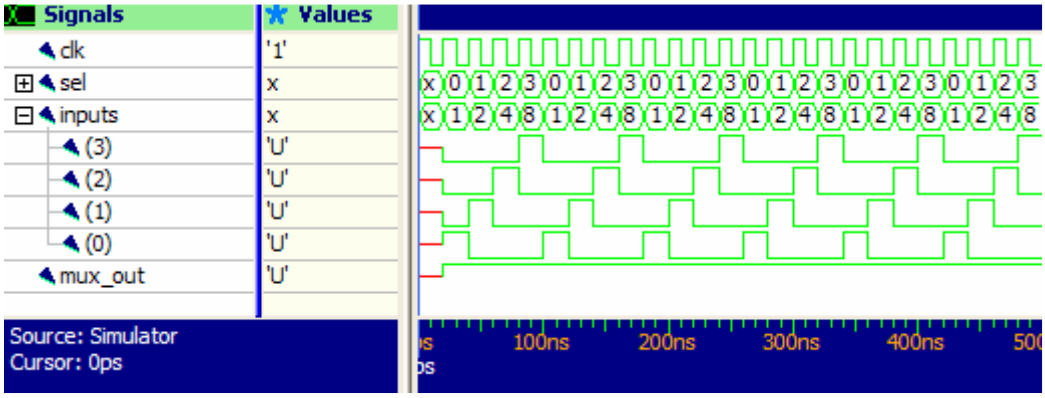

Figure 2.10 Timing Diagram of 4-1 Multiplexer (Modular Design)

The total layout area for the 4-to-1 MUX was generated using Mentor Graphics Leonardo Synthesis Tools. The following are the results generated based on our synthesis module:

```
*******************************************************

Cell: mux4    View: modular_mux4    Library: work

*******************************************************

Total accumulated area :
 Number of IOs :                          7
 Number of LCs :                          3
 Number of accumulated instances :       10
 Number of ports :                        7
 Number of nets :                        16
 Number of instances :                   10
 Number of references to this view :      0


 Cell                   Library  References     Total Area

cyclone_io_input        cyclone     6 x      1      6 IOs
cyclone_io_output       cyclone     1 x      1      1 IOs
cyclone_lcell_normal    cyclone     3 x      1      3 LCs

*********************************************
Device Utilization for EP1C3T100C
*********************************************
Resource                Used    Avail   Utilization
---------------------------------------------
IOs                     7       117       5.98%
LCs                     3       2910      0.10%
Memory Bits             0       53248     0.00%

---------------------------------------------
Using default wire table: cyclone_default
```

8-to-1 MUX: Modular System

The modular system is based instantiations of two 4-to-1 MUX built for modular design. The table below consists of the modules input/output ports:

Port	Type	Bit Width	Description
Sel(2:0)	In	3	Select signal If Sel = “000” select input0 If Sel = “001” select input1 If Sel = “010” select input2 If Sel = “011” select input3 If Sel = “100” select input4 If Sel = “101” select input5 If Sel = “110” select input6 If Sel = “111” select input7
Inputs(7:0)	In	8	Input signal bit(0): Input0 signal bit(1): Input1 signal bit(2): Input2 signal bit(3): Input3 signal bit(4): Input4 signal bit(5): Input5 signal bit(6): Input6 signal bit(7): Input7 signal
mux_out	Out	1	MUX Output signal selected

Table 2.4 Input/Output Port signals for the 8-to-1 Multiplexer

```
--------------------------------------------------------------------
---
-- Title    : VHDL style
-- File name      : mux8.vhd
-- Authors  : Mildred C. Zabawa and Vivek Jayaram
-- Description    : This module is a 8 to 1 Multiplexer expanded from a
--                basic 4 to 1 multiplexer.
--------------------------------------------------------------------
---
library IEEE;
use ieee.std_logic_1164.all;

entity mux8 is
port (Sel     : in  std_logic_vector(2 downto 0);
      inputs  : in  std_logic_vector(7 downto 0);
      mux8_out: out std_logic);
end mux8;

architecture modular_mux8 of mux8 is

component mux4
port (Sel      : in std_logic_vector(1 downto 0);
      inputs   : in std_logic_vector(3 downto 0);
      mux4_out : out std_logic);
end component;

signal out1, out2 : std_logic;

signal sel1, sel2 : std_logic_vector(1 downto 0);

signal inp1, inp2 : std_logic_vector(3 downto 0);

begin

  sel1 <= Sel(1 downto 0);
  sel2 <= sel1;
  inp1 <= inputs(7 downto 4);
  inp2 <= inputs(3 downto 0);

  mux1 : mux4
    port map (Sel      => sel1,
              inputs   => inp1,
              mux4_out => out1);

  mux2 : mux4
    port map (Sel      => sel2,
              inputs   => inp2,
              mux4_out => out2);

 mux8_out <= out1 after 1 ns when Sel(2) = '1' else out2 after  1 ns;

end modular_mux8;
```

Figure 2.11 VHDL Code for the 8-to-1 Multiplexer Modular System

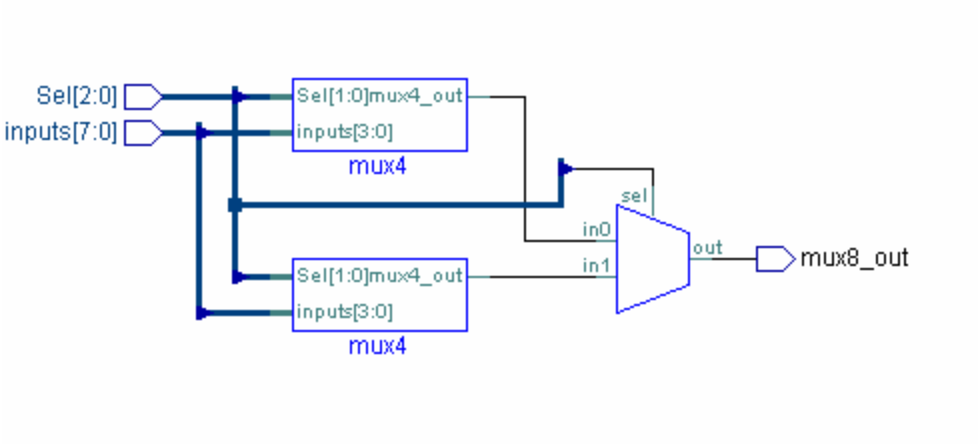

Figure 2.11 System Diagram of the 8-to-1 Multiplexer Modular System

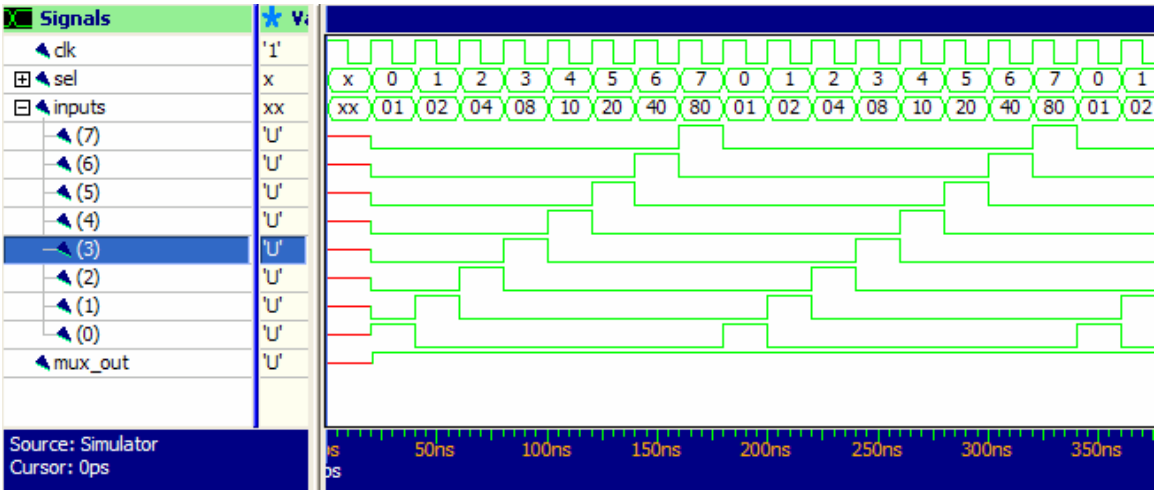

Figure 2.12 Timing Diagram of 8-1 Multiplexer (Modular Design)

The total layout area for the 8-to-1 MUX was generated using Mentor Graphics Leonardo Synthesis Tools. The following are the results generated based on our synthesis module:

```
*******************************************************

Cell: mux8    View: modular_mux8    Library: work

*******************************************************

Total accumulated area :
 Number of IOs :                          12
 Number of LCs :                           6
 Number of accumulated instances :        18
 Number of ports :                        12
 Number of nets :                         29
 Number of instances :                    18
 Number of references to this view :       0


 Cell                   Library  References     Total Area

cyclone_io_input        cyclone    11 x      1     11 IOs
cyclone_io_output       cyclone     1 x      1      1 IOs
cyclone_lcell_normal    cyclone     6 x      1      6 LCs

*********************************************
Device Utilization for EP1C3T100C
*********************************************
Resource                Used    Avail   Utilization
---------------------------------------------
IOs                     12      117      10.26%
LCs                     6       2910      0.21%
Memory Bits             0       53248     0.00%

---------------------------------------------
Using default wire table: cyclone_default
```

3 Decoders

A Decoder contains a n-bit control lines and selects a m-output line. We have design two different techniques shown below:

2-to-4 Decoder: Stand-Alone System

The stand-alone system is based on behavior methodology built using case statements. The table below consists of the modules input/output ports:

Port	Type	Bit Width	Description
A	In	Integer range from (0 to 3)	Control signal If '0' select 0-output line If '1' select 1-output line If '2' select 2-output line If '3' select 3-output line
Y(3:0)	Out	4	Selected Output Line

Table 3.1 Input/Output Port signals for the 2-to-4 Decoder

```
-------------------------------------------------------------------------
---
-- Title    : VHDL style
-- File name      : decoder_24.vhd
-- Authors  : Mildred C. Zabawa and Vivek Jayaram
-- Description    : This module is the stand alone module of a
--                2 to 4 decoder.
-------------------------------------------------------------------------
---
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity decoder_24 is
 port ( A: in  integer range 0 to 3;
        Y: out unsigned(3 downto 0));
end decoder_24;
```

Figure 3.1 VHDL Code for the 2-to-4 Decoder Stand-Alone System (Con't)

```
architecture stand_alone_decoder24 of decoder_24 is
begin
      process(A)
      begin
        case A is
            when 0 => Y <= "0001";
            when 1 => Y <= "0010";
            when 2 => Y <= "0100";
            when 3 => Y <= "1000";
        end case;
      end process;

end stand_alone_decoder24;
```

Figure 3.2 (Con't) VHDL Code for the 2-to-4 Decoder Stand-Alone System

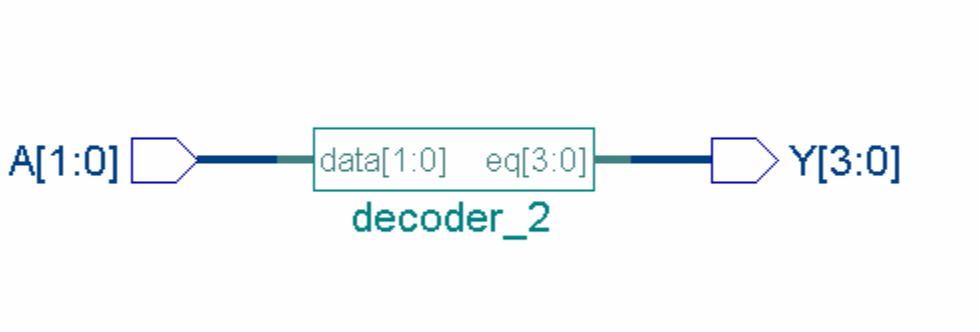

Figure 3.3 System Diagram of the 2-to-4 Decoder Stand-Alone System

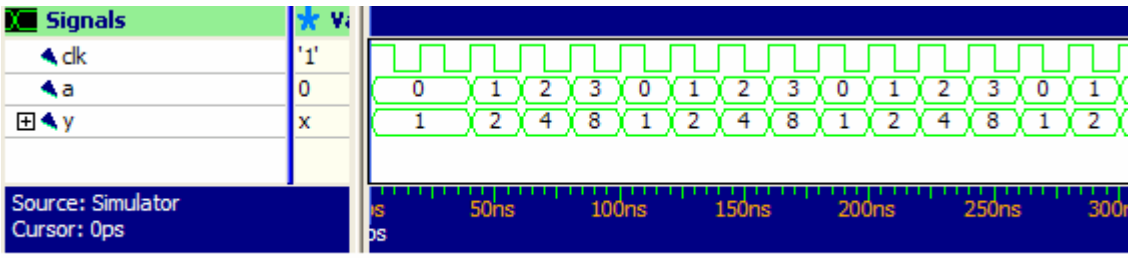

Figure 3.4 Timing Diagram of 2-4 Decoder (Stand Alone Module)

The total layout area for the 2-to-4 Decoder was generated using Mentor Graphics Leonardo Synthesis Tools. The following are the results generated based on our synthesis module:

```
*******************************************************

Cell: decoder_24    View: stand_alone_decoder24    Library: work

*******************************************************

Total accumulated area :
 Number of IOs :                          6
 Number of LCs :                          4
 Number of accumulated instances :       10
 Number of ports :                        6
 Number of nets :                        12
 Number of instances :                   10
 Number of references to this view :      0


 Cell                   Library  References     Total Area

cyclone_io_input        cyclone     2 x      1      2 IOs
cyclone_io_output       cyclone     4 x      1      4 IOs
cyclone_lcell_normal    cyclone     4 x      1      4 LCs

*********************************************
Device Utilization for EP1C3T100C
*********************************************
Resource                Used    Avail   Utilization
---------------------------------------------
IOs                     6       117       5.13%
LCs                     4       2910      0.14%
Memory Bits             0       53248     0.00%

---------------------------------------------
Using default wire table: cyclone_default
```

3-to-8 Decoder: Stand-Alone System

The stand-alone system is based on behavior methodology built using case statements. The table below consists of the modules input/output ports:

Port	Type	Bit Width	Description
A	In	Integer range from (0 to 7)	Control signal If '0' select 0-output line If '1' select 1-output line If '2' select 2-output line If '3' select 3-output line If '4' select 4-output line If '5' select 5-output line If '6' select 6-output line If '7' select 7-output line
Y(7:0)	Out	8	Selected Output Line

Table 3.2 Input/Output Port signals for the 3-to-8 Decoder

```
------------------------------------------------------------------------
---
-- Title    : VHDL style
-- File name      : decoder_38.vhd
-- Authors  : Mildred C. Zabawa and Vivek Jayaram
-- Description    : This module is the stand alone module of a
--                3 to 8 decoder.
------------------------------------------------------------------------
---
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity decoder_38 is
 port ( A: in  integer range 0 to 7;
        Y: out unsigned(7 downto 0));
end decoder_38;
```

Figure 3.5 VHDL Code for the 3-to-8 Decoder Stand-Alone System (Con't)

```
architecture stand_alone_decoder38 of decoder_38 is
begin
      process(A)
      begin
        case A is
            when 0 => Y <= "00000001";
            when 1 => Y <= "00000010";
            when 2 => Y <= "00000100";
            when 3 => Y <= "00001000";
            when 4 => Y <= "00010000";
            when 5 => Y <= "00100000";
            when 6 => Y <= "01000000";
            when 7 => Y <= "10000000";
        end case;
      end process;

end stand_alone_decoder38;
```

Figure 3.6 (Con't) VHDL Code for the 3-to-8 Decoder Stand-Alone System

Figure 3.7 System Diagram of the 3-to-8 Decoder Stand-Alone System

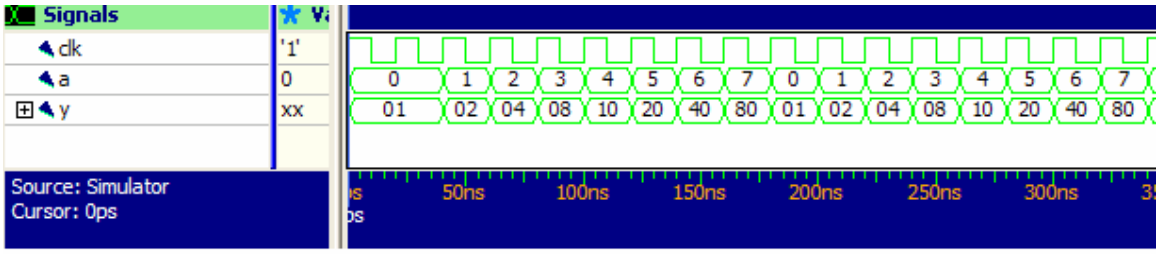

Figure 3.8 Timing Diagram of 3-8 Decoder (Stand Alone Module)

The total layout area for the 3-to-8 Decoder was generated using Mentor Graphics Leonardo Synthesis Tools. The following are the results generated based on our synthesis module:

```
*******************************************************

Cell: decoder_38    View: stand_alone_decoder38    Library: work

*******************************************************

Total accumulated area :
 Number of IOs :                          11
 Number of LCs :                           8
 Number of accumulated instances :        19
 Number of ports :                        11
 Number of nets :                         22
 Number of instances :                    19
 Number of references to this view :       0


 Cell                   Library  References     Total Area

cyclone_io_input        cyclone     3 x      1      3 IOs
cyclone_io_output       cyclone     8 x      1      8 IOs
cyclone_lcell_normal    cyclone     8 x      1      8 LCs

*********************************************
Device Utilization for EP1C3T100C
*********************************************
Resource                Used    Avail   Utilization
---------------------------------------------
IOs                     11      117       9.40%
LCs                     8       2910      0.27%
Memory Bits             0       53248     0.00%

---------------------------------------------
Using default wire table: cyclone_default
```

2-to-4 Decoder: Modular System

The modular system is based on behavior methodology built using select statements. The table below consists of the modules input/output ports:

Port	Type	Bit Width	Description
Sel(1:0)	In	2	Select signal If Sel = “00” select 0-output If Sel = “01” select 1-output If Sel = “10” select 2-output If Sel = “11” select 3-output
Y(3:0)	Out	4	Output signal

Table 3.3 Input/Output Port signals for the 2-to-4 Decoder

```
------------------------------------------------------------------------
---
-- Title    : VHDL style
-- File name      : decoder24.vhd
-- Authors  : Mildred C. Zabawa and Vivek Jayaram
-- Description    : This module is a modular design of a
--                2 to 4 decoder.
------------------------------------------------------------------------
---
library ieee;
use ieee.std_logic_1164.all;

entity decoder24 is
 port ( Sel: in  std_logic_vector(1 downto 0);
        Y: out std_logic_vector(3 downto 0));
end decoder24;

architecture modular_decoder24 of decoder24 is
begin
      with Sel select
        Y <= "0001" when "00",
             "0010" when "01",
             "0100" when "10",
             "1000" when "11",
             "0000" when others;

end modular_decoder24;
```

Figure 3.9 VHDL Code for the 2-to-4 Decoder Modular System

Figure 3.10 System Diagram of the 2-to-4 Decoder Modular System

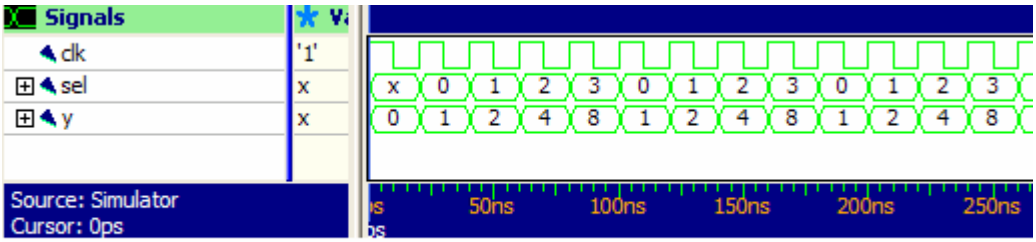

Figure 3.11 Timing Diagram of 2-4 Decoder (Modular Design)

The total layout area for the 2-to-4 Decoder was generated using Mentor Graphics Leonardo Synthesis Tools. The following are the results generated based on our synthesis module:

```
*******************************************************

Cell: decoder24    View: modular_decoder24    Library: work

*******************************************************

Total accumulated area :
 Number of IOs :                          6
 Number of LCs :                          4
 Number of accumulated instances :       10
 Number of ports :                        6
 Number of nets :                        12
 Number of instances :                   10
 Number of references to this view :      0


 Cell                   Library  References     Total Area

cyclone_io_input        cyclone     2 x      1      2 IOs
cyclone_io_output       cyclone     4 x      1      4 IOs
cyclone_lcell_normal    cyclone     4 x      1      4 LCs

*********************************************
Device Utilization for EP1C3T100C
*********************************************
Resource                Used    Avail   Utilization
---------------------------------------------
IOs                     6       117       5.13%
LCs                     4       2910      0.14%
Memory Bits             0       53248     0.00%

---------------------------------------------
Using default wire table: cyclone_default
```

3-to-8 Decoder: Modular System

The modular system is based instantiations of two 2-to-4 Decoder built for modular design. The table below consists of the modules input/output ports:

Port	Type	Bit Width	Description
Sel(2:0)	In	3	Select signal If Sel = "000" select input0 If Sel = "001" select input1 If Sel = "010" select input2 If Sel = "011" select input3 If Sel = "100" select input4 If Sel = "101" select input5 If Sel = "110" select input6 If Sel = "111" select input7
Y(7:0)	Out	8	Output signal

Table 3.4 Input/Output Port signals for the 8-to-1 Multiplexer

```
-------------------------------------------------------------------------
---
-- Title    : VHDL style
-- File name      : decoder38.vhd
-- Authors  : Mildred C. Zabawa and Vivek Jayaram
-- Description    : This module is a 3 to 8 decoder based on
--                two 2 to 4 decoders.
-------------------------------------------------------------------------
library IEEE;
use ieee.std_logic_1164.all;

entity decoder38 is
 port ( Sel: in  std_logic_vector(2 downto 0);
          Y: out std_logic_vector(7 downto 0));
end decoder38;

architecture modular_decoder38 of decoder38 is

component decoder24
port ( Sel: in  std_logic_vector(1 downto 0);
        Y: out std_logic_vector(3 downto 0));
end component;

signal out1, out2 : std_logic_vector(3 downto 0);
signal y1, y2     : std_logic_vector(7 downto 0);
signal sel1, sel2 : std_logic_vector(1 downto 0);
```

Figure 3.12 VHDL Code for the 3-to-8 Decoder Modular System (Con't)

```
begin
  sel1 <= Sel(1 downto 0);
  sel2 <= sel1;

  dec1 : decoder24
    port map (Sel        => sel1,
              Y          => out1);
  dec2 : decoder24
    port map (Sel        => sel2,
              Y          => out2);

  y1 <= out1&"0000";
  y2 <= "0000"&out2;

  Y <= y1 when Sel(2) = '1' else y2;

end modular_decoder38;
```

Figure 3.13 (Con't) VHDL Code for the 3-to-8 Decoder Modular System

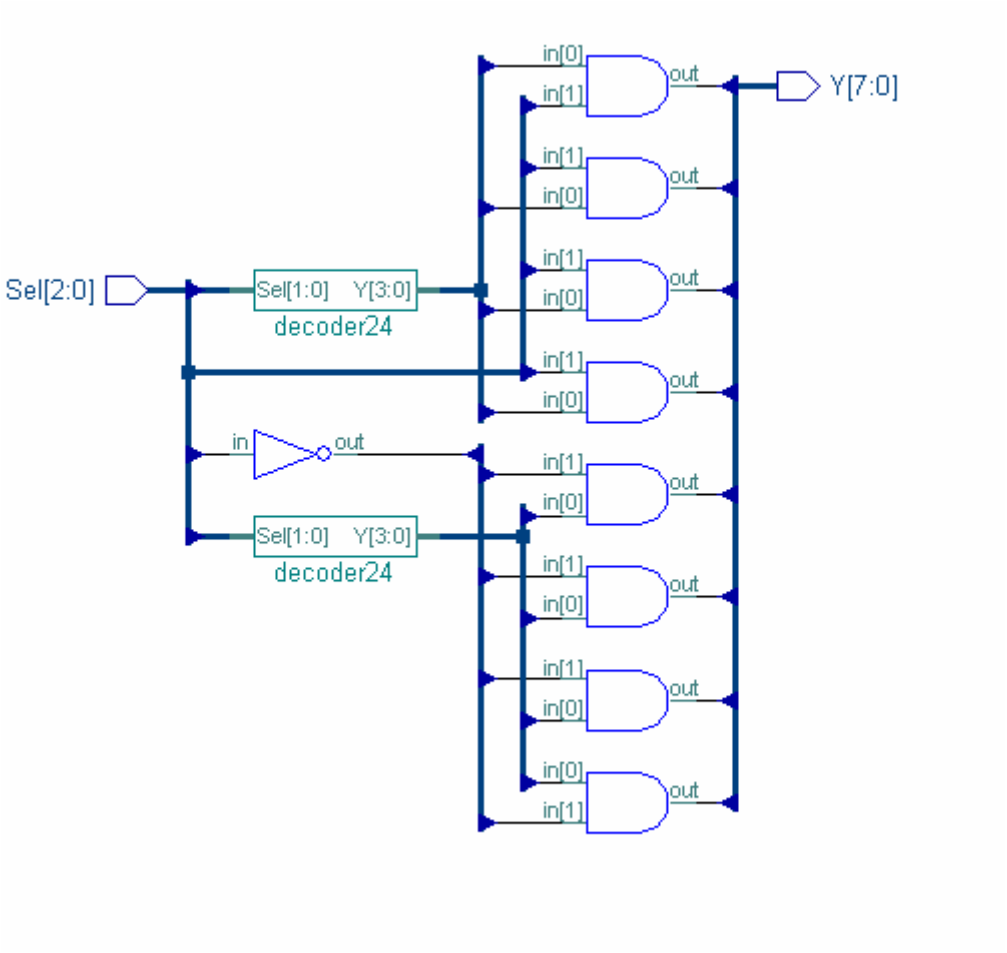

Figure 3.14 System Diagram of the 3-to-8 Decoder Modular System

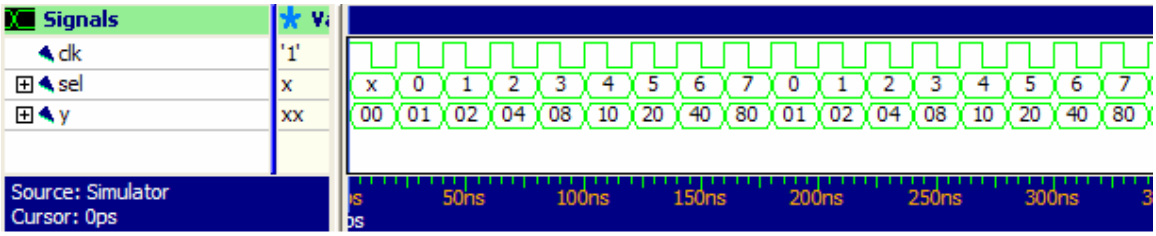

Figure 3.15 Timing Diagram of 3-8 Decoder (Modular Design)

The total layout area for the 3-to-8 Decoder was generated using Mentor Graphics Leonardo Synthesis Tools. The following are the results generated based on our synthesis module:

```
*******************************************************

Cell: decoder38    View: modular_decoder38    Library: work

*******************************************************

Total accumulated area :
 Number of IOs :                          11
 Number of LCs :                           8
 Number of accumulated instances :        19
 Number of ports :                        11
 Number of nets :                         22
 Number of instances :                    19
 Number of references to this view :       0


 Cell                   Library  References     Total Area

cyclone_io_input        cyclone     3 x      1      3 IOs
cyclone_io_output       cyclone     8 x      1      8 IOs
cyclone_lcell_normal    cyclone     8 x      1      8 LCs

*********************************************
Device Utilization for EP1C3T100C
*********************************************
Resource                Used    Avail   Utilization
---------------------------------------------
IOs                     11      117       9.40%
LCs                     8       2910      0.27%
Memory Bits             0       53248     0.00%

---------------------------------------------
Using default wire table: cyclone_default
```

4 Logical Shift Register

A Logical Shift Register with a Serial_In bit, Control Logic indicating direction of shift, and a n-output word. We have design two different techniques shown below:

4-bit Logical Shift Register: Stand-Alone System

The stand-alone system is based on behavior methodology built using if-else statements. The table below consists of the modules input/output ports:

Port	Type	Bit Width	Description
Clock	In	1	Clock signal
Serial_In	In	1	Serial Input signal
Ctl_Lt_Rt	In	1	Control Left/Right shift direction
Serial_Out	Out	4	Serial Output signal

Table 4.1 Input/Output Port signals for the Logical Shift Register

```
------------------------------------------------------------------------
---
-- Title    : VHDL style
-- File name      : logic_shift_4.vhd
-- Authors  : Mildred C. Zabawa and Vivek Jayaram
-- Description    : This module is a stand alone design of a
--                4-bit logic shift.
------------------------------------------------------------------------
---
library ieee;
use ieee.std_logic_1164.all;

entity logic_shift_4 is
port(Clock     : in  std_logic;
     Serial_In : in  std_logic;
     Ctl_Lt_Rt : in  std_logic;
     Serial_Out: out std_logic_vector(3 downto 0));
end logic_shift_4;


architecture stand_alone_shift4 of logic_shift_4 is
  signal tmp: std_logic_vector(3 downto 0);
  begin
    process (Clock)
```

Figure 4.1 VHDL Code for the Logical Shift Register Stand-Alone System (Con't)

```
      begin
        if (Clock'event and Clock='1') then
          if (Ctl_Lt_Rt='0') then
            tmp <= tmp(2 downto 0) & Serial_In;
          else
            tmp <= Serial_In & tmp(3 downto 1);
          end if;
        end if;
    end process;
    Serial_Out <= tmp;

end stand_alone_shift4;
```

Figure 4.2 (Con't) VHDL Code for the 4-bit Logical Shift Reg. Stand-Alone System

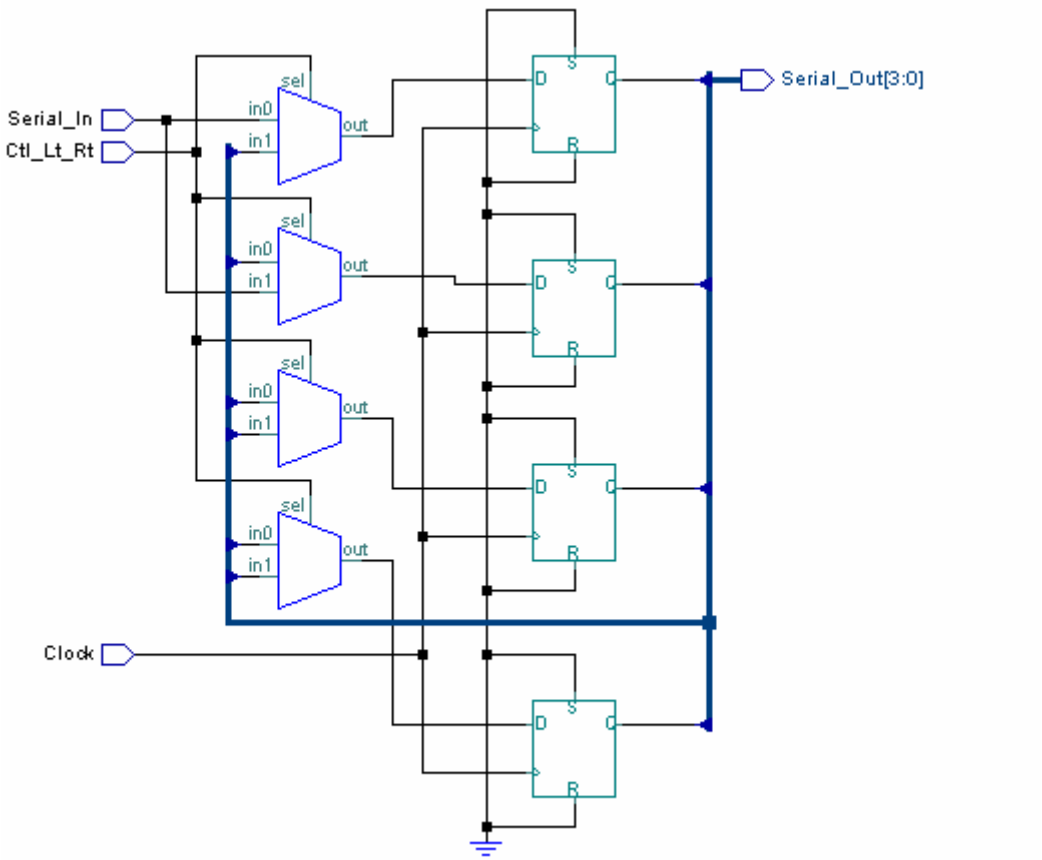

Figure 4.3 System Diagram of the 4-bit Logical Shift Register Stand-Alone System

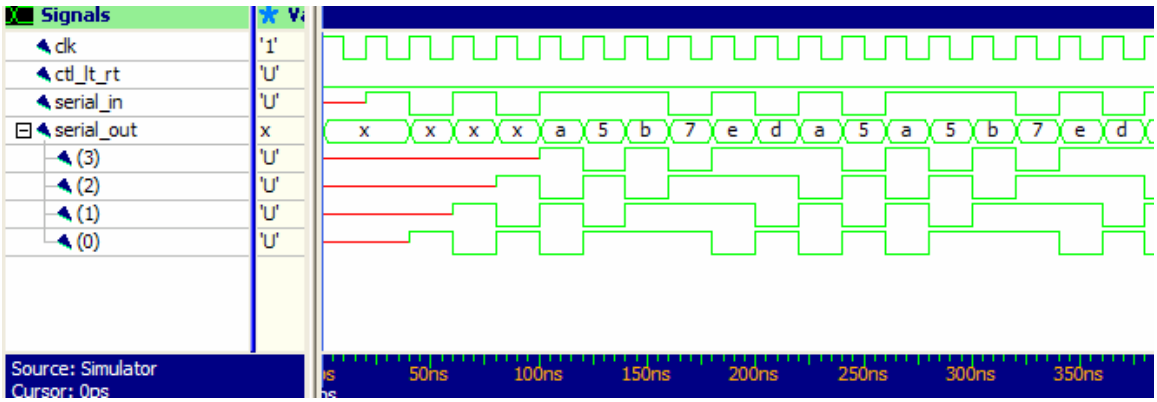

Figure 4.4 Timing Diagram of 4-bit Right Logical Shift Reg. (Stand Alone Module)

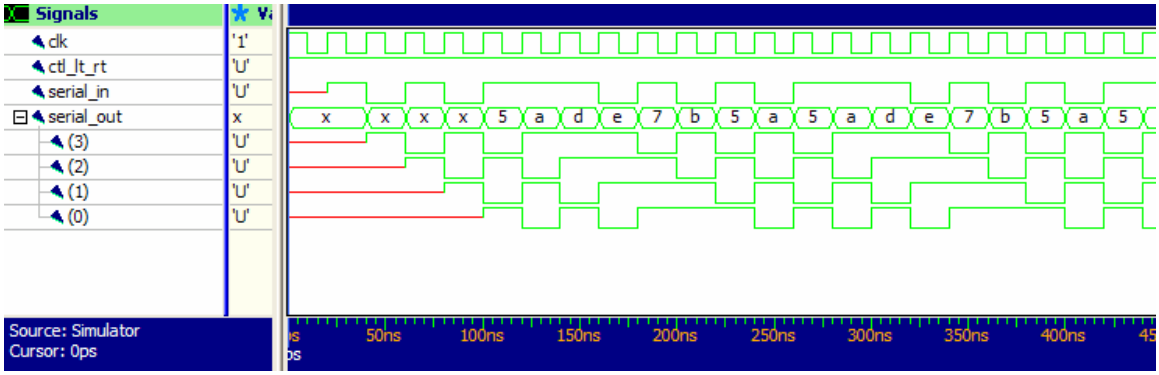

Figure 4.5 Timing Diagram of 4-bit Left Logical Shift Reg. (Stand Alone Module)

The total layout area for the 4-bit Logical Shift Register was generated using Mentor Graphics Leonardo Synthesis Tools. The following are the results generated based on our synthesis module:

```
*******************************************************

Cell: logic_shift_4    View: stand_alone_shift4    Library: work

*******************************************************

Total accumulated area :
 Number of GND :                          2
 Number of IOs :                          7
 Number of LCs :                          4
 Number of VCC :                          1
 Number of accumulated instances :       14
 Number of ports :                        7
 Number of nets :                        15
 Number of instances :                   14
 Number of references to this view :      0


 Cell                     Library  References     Total Area

GND                       cyclone     2 x      1       2 GND
VCC                       cyclone     1 x      1       1 VCC
cyclone_io_input          cyclone     3 x      1       3 IOs
cyclone_io_output         cyclone     4 x      1       4 IOs
cyclone_lcell_sync_reg    cyclone     4 x      1       4 LCs

*********************************************
Device Utilization for EP1C3T100C
*********************************************
Resource                 Used    Avail   Utilization
---------------------------------------------
IOs                      7       117       5.98%
LCs                      4       2910      0.14%
Memory Bits              0       53248     0.00%

---------------------------------------------
Using default wire table: cyclone_default
```