Department of Electrical and Computer Engineering

Spring-2004
Updated : Summer 2006

Article on Implementation of Digital Circuits

Professor: **Dr. Subbarao V. Wunnava**
Vivek Jayaram/ Pavani Rao

Contents

Abstract

In this article various digital circuits like ALU, Parallel to Serial Converter, Multiplexers, Decoders, Shift Registers and Barrel shifter unit are presented in Hardware description languages like Verilog HDL and VHDL, simulated using Mentor Graphics ModelSim and synthesized using Mentor Graphics Leonardo Spectrum tools.

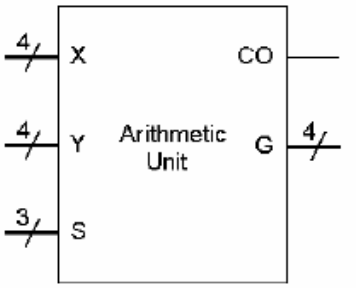
Coding of Digital Circuits

Arithmetic and Logic Unit

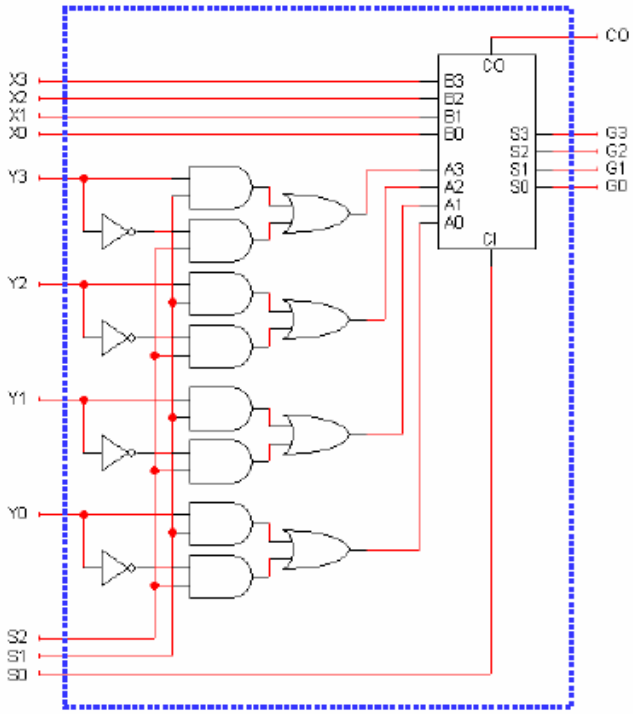
Functionality

Selection				Output	Function
S0	S1	S2	Cn		
0	0	0	0	in1	Transfer in 1
0	0	0	1	in1 +1	Increment in 1
0	0	1	0	in1 + in2	Addition
0	0	1	1	in1+ in2 + 1	Add with carry
0	1	0	0	in1 - in2 - 1	Subtract with borrow
0	1	0	1	in1 - in2	Subtraction
0	1	1	0	in1 - 1	Decrement in1
0	1	1	1	in1	Transfer in 1
1	0	0	x	in1 OR in2	OR
1	0	1	x	in1 XOR in2	XOR
1	1	0	x	in1 AND in2	AND
1	1	1	x	NOT in1	complement in1

Complete arithmetic unit

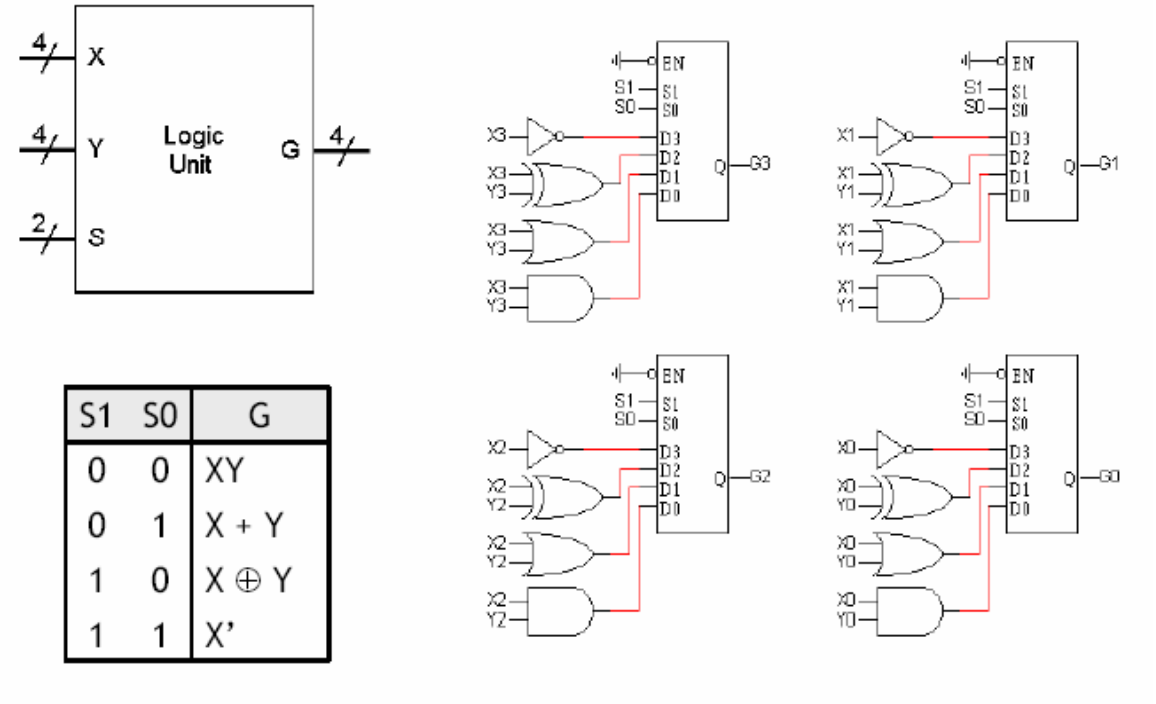


S2	S1	S0	G
0	0	0	X
0	0	1	X + 1
0	1	0	X + Y
0	1	1	X + Y + 1
1	0	0	X + Y'
1	0	1	X + Y' + 1
1	1	0	X - 1
1	1	1	X



(Ref: Arithmetic-logic units by Howard Huang 2000-2003)

Complete logic unit



S1	S0	G
0	0	XY
0	1	X + Y
1	0	X ⊕ Y
1	1	X’

Verilog HDL code for Arithmetic and Logic Unit

```
//transfer,increment,decrement,addition,add with carry,subtraction,subtract with borrow,
//OR,XOR,AND,NOT etc.

module gen_alu(out,cout,s0,s1,s2,cin,in1,in2);
output [1:0]out;
output cout;
input s0,s1,s2,cin;
input [1:0]in1,in2;
wire [1:0]out,in1,in2;
wire s0,s1,s2,cin;
wire
n1,n2,n3,n4,n5,n6,n7,n8,n9,n10,n11,n12,n13,n14,n15,n16,n17,n18,n19,n20,n21,c2,c3;
faddh f1(out[0],c2,n9,n13,n6);
faddh f2(out[1],c3,n16,n20,n21);
not g1(n1,s2);
not g2(n2,s0);
```

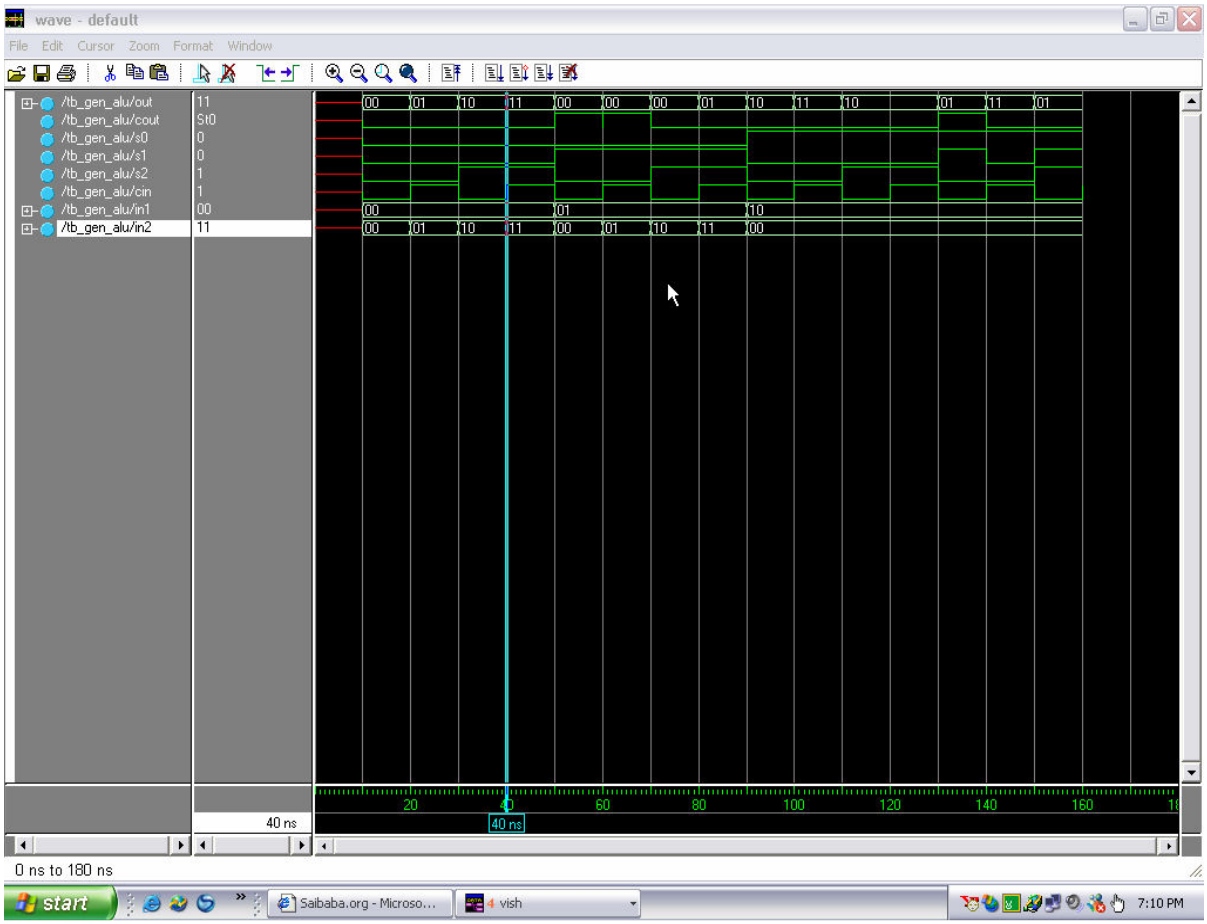
```
not g3(n3,s1);
and g4(n4,n2,s1,s2);
and g5(n5,n2,n3,s2);
and g6(n6,n1,cin);
and g7(n7,n5,in2[0]);
and g8(n8,n4,n10);
or g9(n9,n7,n8,in1[0]);
and g10(n11,s0,in2[0]);
and g11(n12,n10,s1);
or g12(n13,n11,n12);
not g13(n10,in2[0]);
and g14(n21,n1,c2);
and g15(n14,n5,in2[1]);
and g16(n15,n4,n17);
or g17(n16,n14,n15,in1[1]);
not g18(n17,in2[1]);
and g19(n18,in2[1],s0);
and g20(n19,n17,s1);
or g21(n20,n18,n19);
and g22(cout,n1,c3);
endmodule
```

Test bench for ALU in Verilog HDL

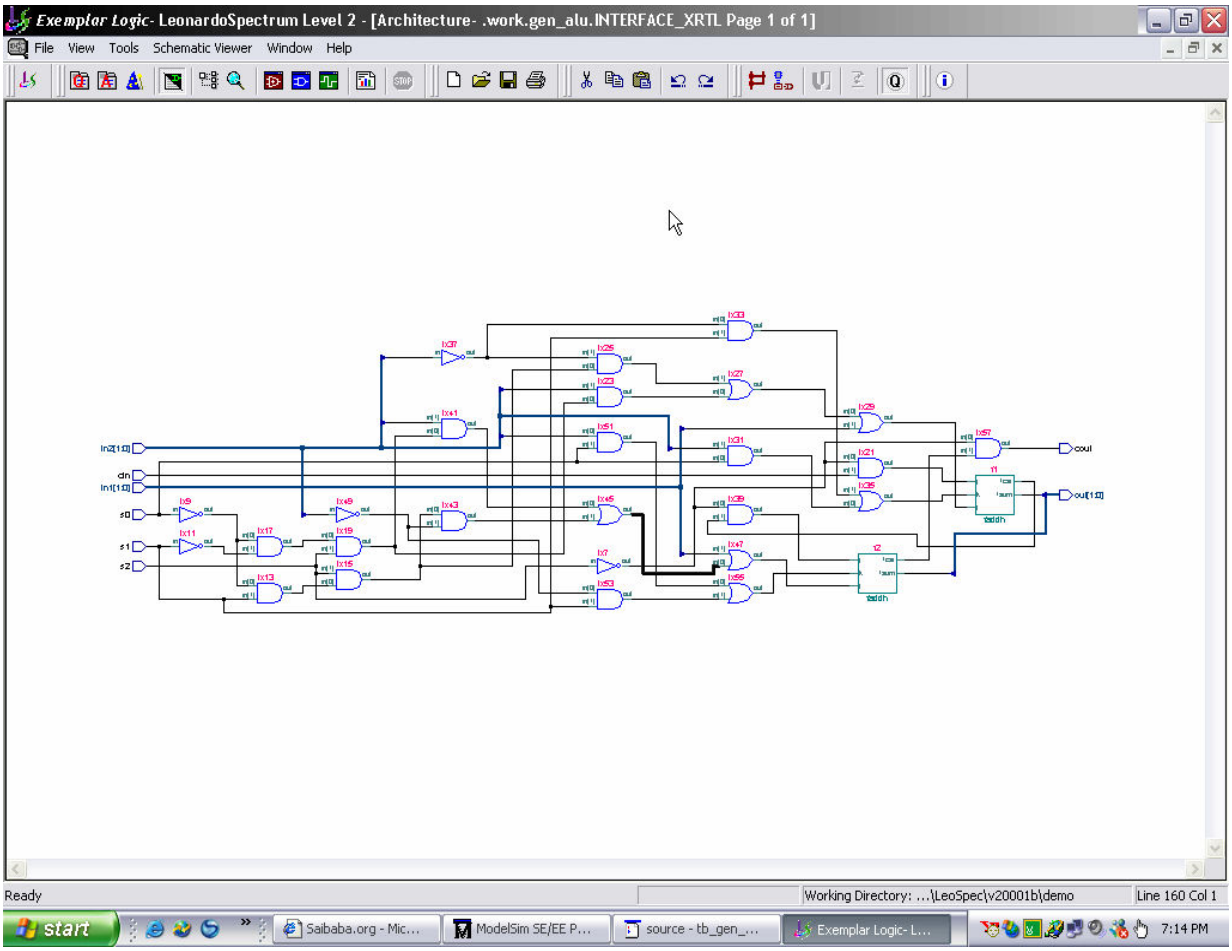
```
module tb_gen_alu;
wire [1:0]out;
wire cout;
reg s0,s1,s2,cin;
reg [1:0]in1,in2;
gen_alu ge1(out,cout,s0,s1,s2,cin,in1,in2);
initial
begin
$monitor($time,"out=%b,cout=%b,s0=%b,s1=%b,s2=%b,cin=%b,in1=%b,in2=%b",out,
cout,s0,s1,s2,cin,in1,in2);
//transfer in1
#10 s0=1'b0;s1=1'b0;s2=1'b0;cin=1'b0;in1=2'b00;in2=2'b00;
//increment in1
#10 s0=1'b0;s1=1'b0;s2=1'b0;cin=1'b1;in1=2'b00;in2=2'b01;
//addition
#10 s0=1'b0;s1=1'b0;s2=1'b1;cin=1'b0;in1=2'b00;in2=2'b10;
//add with carry
#10 s0=1'b0;s1=1'b0;s2=1'b1;cin=1'b1;in1=2'b00;in2=2'b11;
//subtract with borrow
#10 s0=1'b0;s1=1'b1;s2=1'b0;cin=1'b0;in1=2'b01;in2=2'b00;
//subtraction
#10 s0=1'b0;s1=1'b1;s2=1'b0;cin=1'b1;in1=2'b01;in2=2'b01;
```

```
//decrement in1
#10 s0=1'b0;s1=1'b1;s2=1'b1;cin=1'b0;in1=2'b01;in2=2'b10;
//transfer in1
#10 s0=1'b0;s1=1'b1;s2=1'b1;cin=1'b1;in1=2'b01;in2=2'b11;
//OR
#10 s0=1'b1;s1=1'b0;s2=1'b0;cin=1'b0;in1=2'b10;in2=2'b00;
#10 s0=1'b1;s1=1'b0;s2=1'b0;cin=1'b1;in1=2'b10;in2=2'b00;
//XOR
#10 s0=1'b1;s1=1'b0;s2=1'b1;cin=1'b0;in1=2'b10;in2=2'b00;
#10 s0=1'b1;s1=1'b0;s2=1'b1;cin=1'b1;in1=2'b10;in2=2'b00;
//AND
#10 s0=1'b1;s1=1'b1;s2=1'b0;cin=1'b0;in1=2'b10;in2=2'b00;
#10 s0=1'b1;s1=1'b0;s2=1'b0;cin=1'b1;in1=2'b10;in2=2'b00;
//complement in1
#10 s0=1'b1;s1=1'b1;s2=1'b1;cin=1'b0;in1=2'b10;in2=2'b00;
#10 s0=1'b1;s1=1'b1;s2=1'b1;cin=1'b1;in1=2'b10;in2=2'b00;
end
endmodule
```

Simulated output using Mentor Graphics ModelSim tool



Synthesized output using Mentor Graphics Leonardo Spectrum tool



Parallel to Serial Converter

Functionality

- Parallel data is loaded when data_rdy =1
- Each bit is shifted serially when clk signal is high

VHDL code for Parallel to Serial Converter

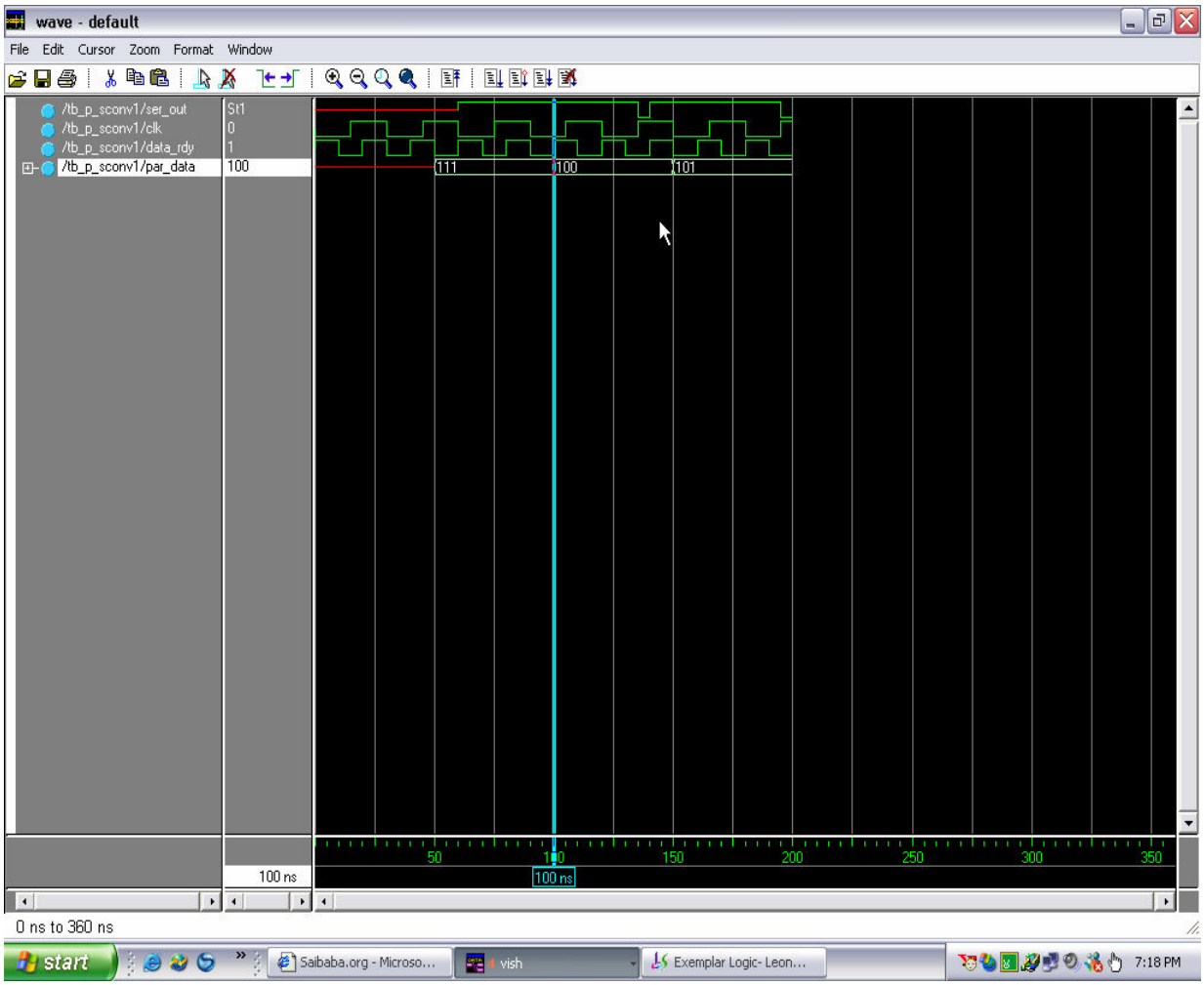
```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity p_sconv1 is
generic(N:POSITIVE:=3);
port(PAR_DATA:in STD_LOGIC_VECTOR(1 to 3);
CLK,DATA_RDY:in STD_LOGIC;
SER_OUT:out STD_LOGIC);
end;
architecture SHIFT of p_sconv1 is
signal SAVE_REG:STD_LOGIC_VECTOR(1 to 3);
begin
process(CLK,DATA_RDY,par_data)
begin
if DATA_RDY='1'then
SAVE_REG<=PAR_DATA;
elsif CLK'EVENT and CLK='1' then
SAVE_REG(1 to 2)<=SAVE_REG(2 to 3);
END IF;
END PROCESS;
SER_OUT<=SAVE_reg(1);
end;
```

Test bench for Parallel to Serial Converter in Verilog HDL

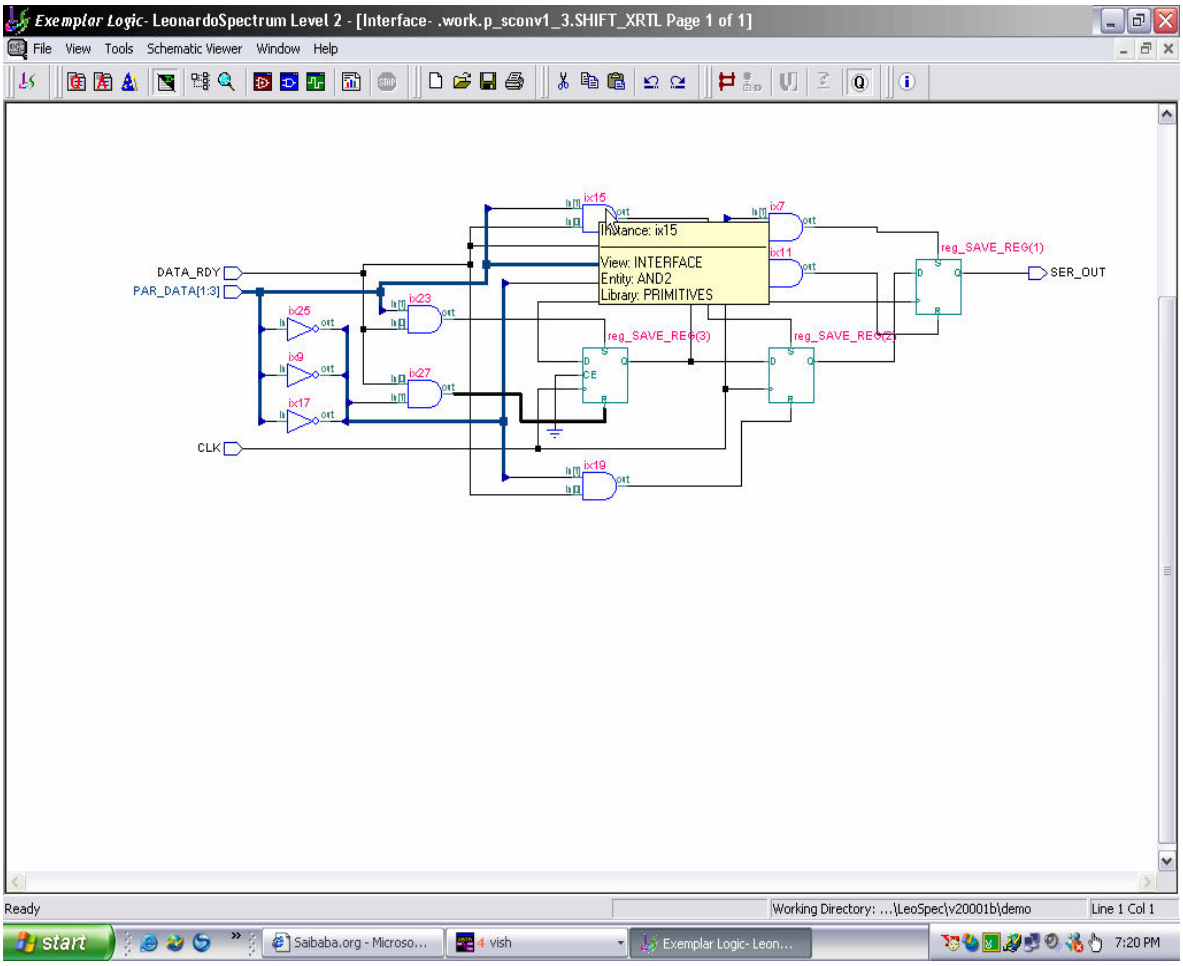
```
`define F $finish
module tb_p_sconv1;
wire ser_out;
reg clk,data_rdy;
reg [1:3]par_data;
```

```
p_sconv1 p1(par_data,clk,data_rdy,ser_out);
initial
begin
clk=1'b0;
data_rdy=1'b1;
end
initial
forever #10 data_rdy=~data_rdy;
initial
forever #15 clk=~clk;
initial
#200`F;
initial
begin
$monitor("time=%d,par_data=%b,clk=%b,data_rdy=%b,,ser_out=%b",$time,par_data,cl
k,data_rdy,ser_out);
#50 par_data=3'b111;#50 par_data=3'b100;#50 par_data=3'b101;
end
endmodule
```

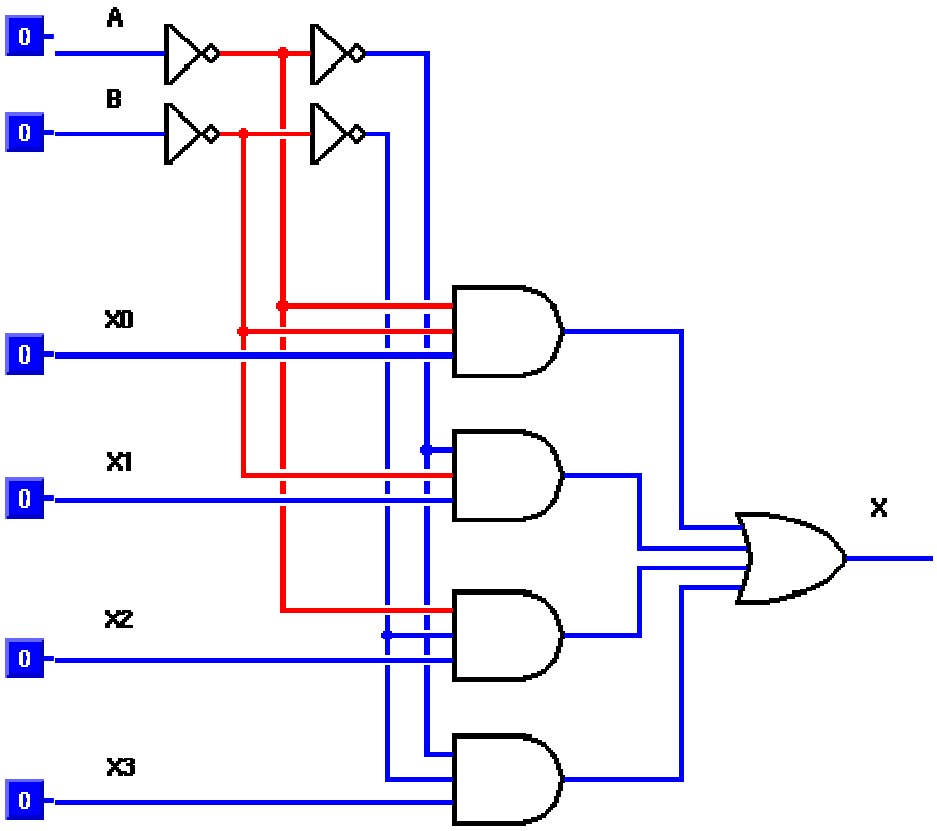
Simulated output using Mentor Graphics ModelSim tool



Synthesized output using Mentor Graphics Leonardo Spectrum tool



Multiplexer



Functionality

S1	S0	Out
0	0	in0
0	1	in1
1	0	in2
1	1	in3

Verilog HDL code for 4 X 1 Multiplexer

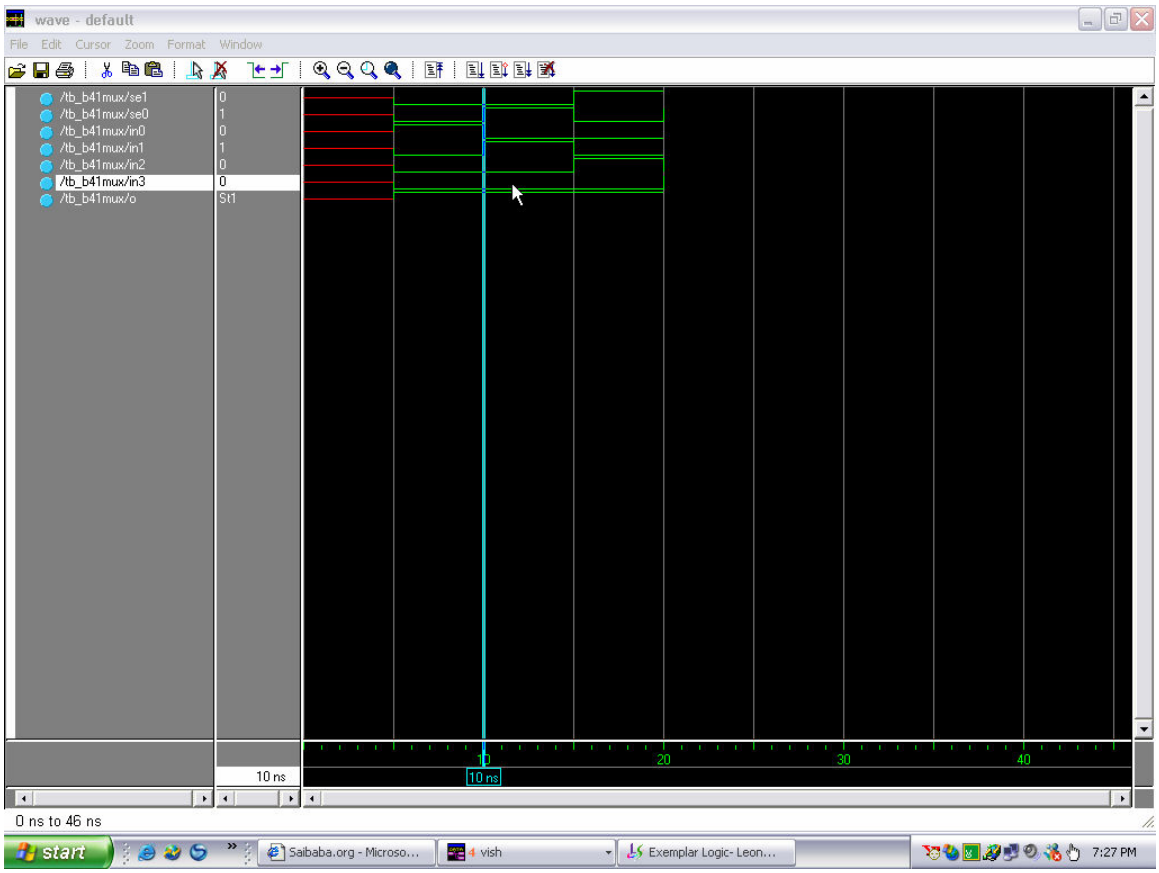
```
module  b41mux(out,i0,i1,i2,i3,s1,s0);

output out;
input i0,i1,i2,i3,s0,s1;

reg out;
wire i0,i1,i2,i3,s0,s1;
```

```
always@(s0 or s1 or i0 or i1 or i2 or i3)

begin

if (s1==1'b0 && s0==1'b0)
out=i0;

else if(s1==1'b0 && s0==1'b1)
out=i1;

else if(s1==1'b1 && s0==1'b0)
out=i2;

else
out=i3;

end

endmodule
```

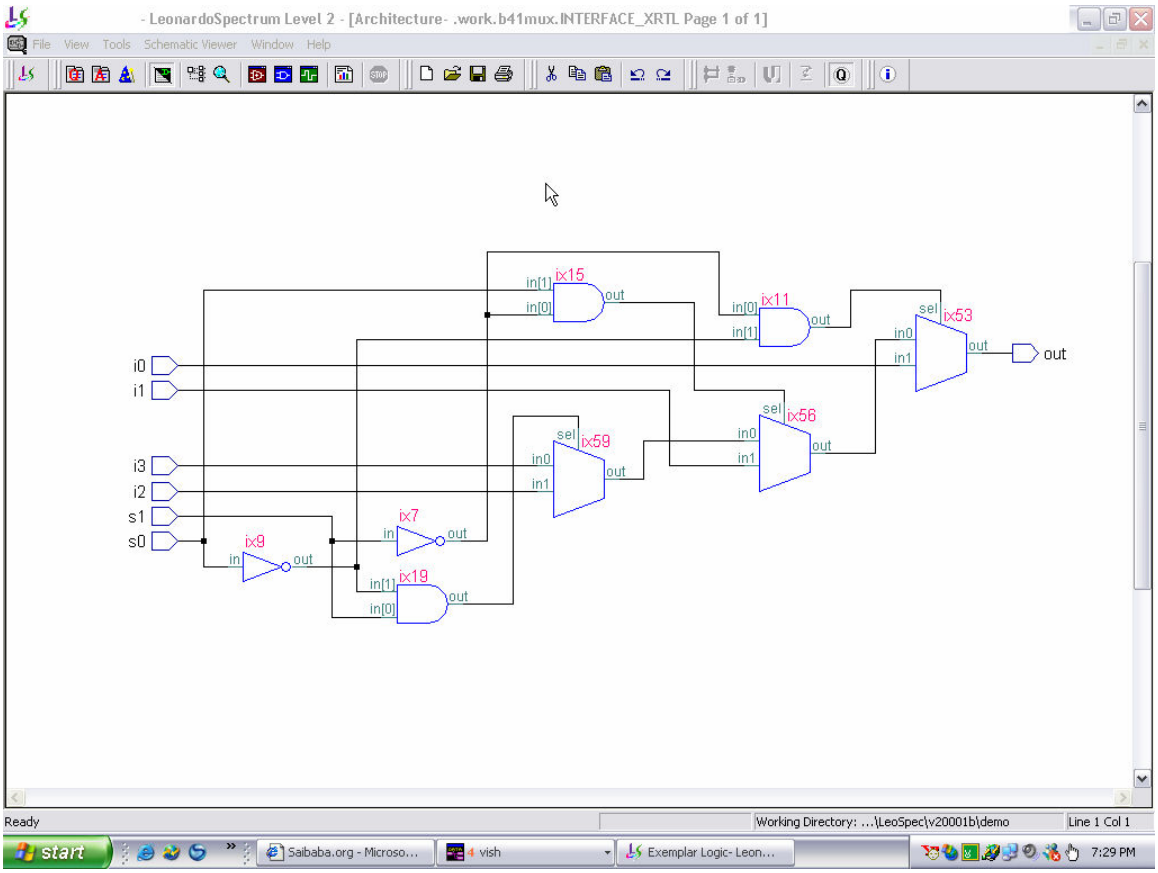
Test bench for 4 X 1 Multiplexer in Verilog HDL

```
module  tb_b41mux;

reg se1,se0,in0,in1,in2,in3;
wire  o;
b41mux m1(.out(o),.s1(se1),.s0(se0),.i1(in1),.i2(in2),.i3(in3),.i0(in0));
initial
begin
$monitor($time,"o=%b,se1=%b,se0=%b,in3=%b,in2=%b,in1=%b,in0=%b",o,se1,se0,
in3,in2,in1,in0);
#5 se1=1'b0;se0=1'b0;in3=1'b0;in2=1'b0;in1=1'b0;in0=1'b1;
#5 se1=1'b0;se0=1'b1;in3=1'b0;in2=1'b0;in1=1'b1;in0=1'b0;
#5 se1=1'b1;se0=1'b0;in3=1'b0;in2=1'b1;in1=1'b0;in0=1'b0;
#5 se1=1'b1;se0=1'b1;in3=1'b1;in2=1'b0;in1=1'b0;in0=1'b0;
end
endmodule
```

Simulated output using Mentor Graphics ModelSim tool



Synthesized output using Mentor Graphics Leonardo Spectrum tool



Verilog HDL code for 8 X 1 Multiplexer

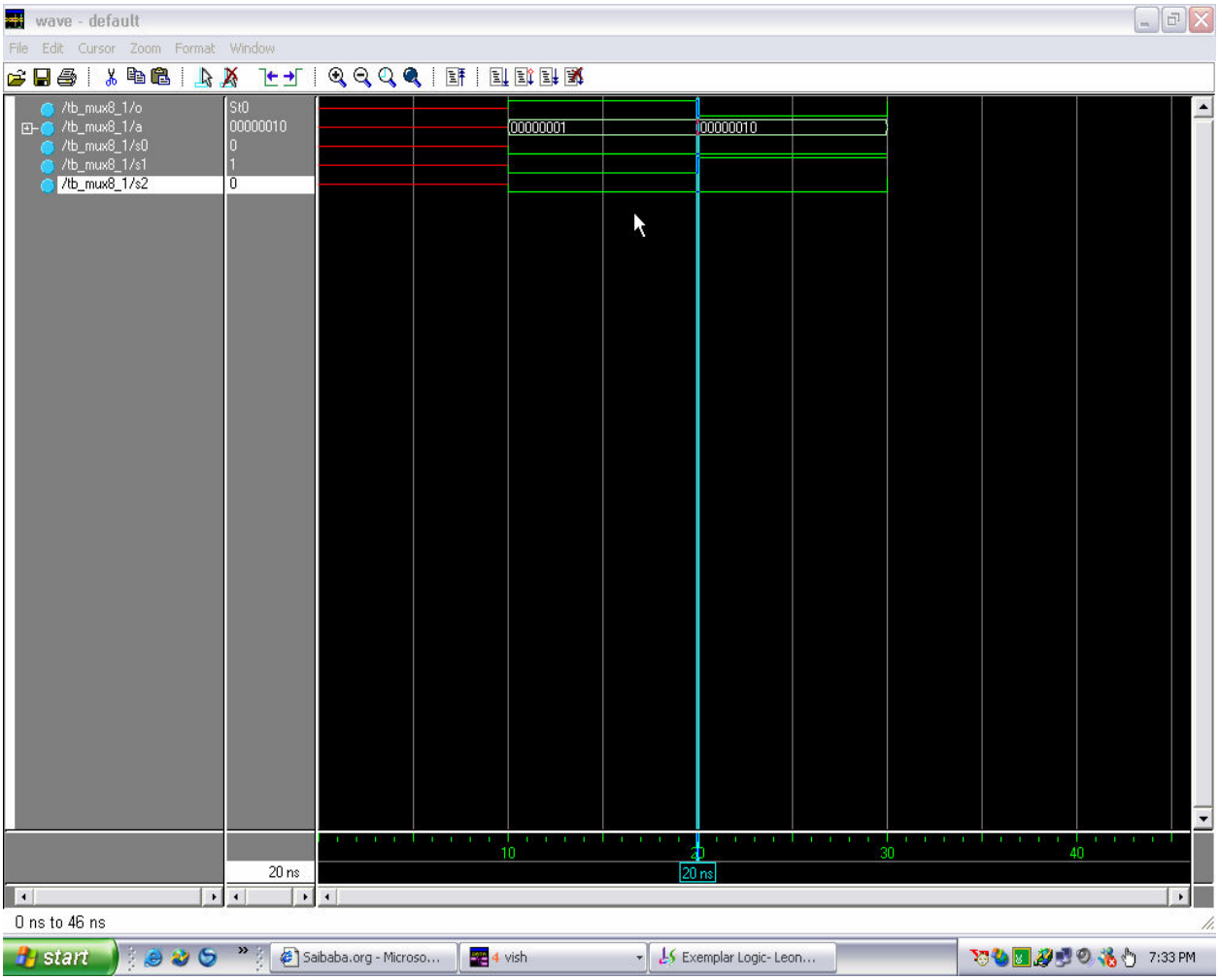
```
module mux8_1 (out,s0,s1,s2,i);

input [7:0]i;
input s0,s1,s2;
output out;

wire [7:0]i;
wire s0,s1,s2;
wire out;

assign out=s2?(s1?(s0?i[7]:i[6]):(s0?i[5]:i[4])):(s1?(s0?i[3]:i[2]):(s0?i[1]:i[0]));

endmodule
```

Test bench for 8 X 1 Multiplexer in Verilog HDL

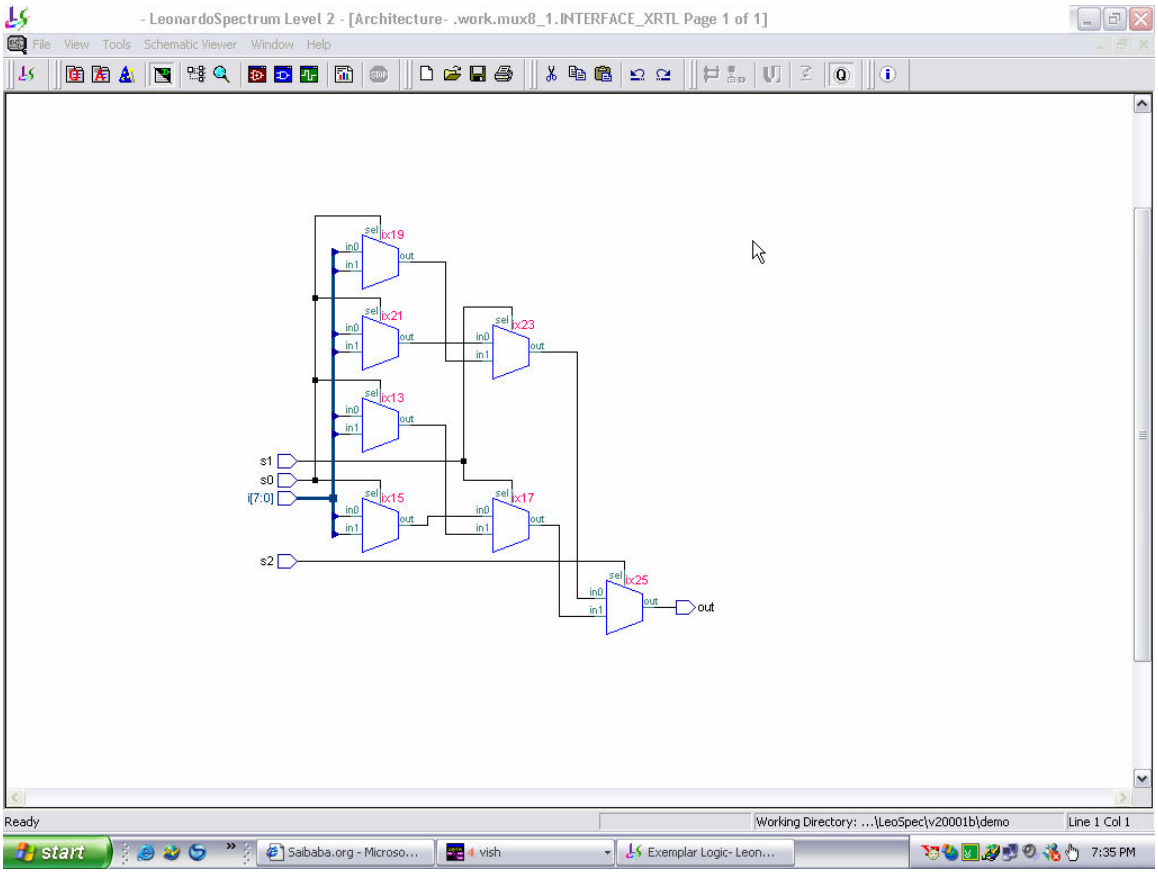
```
module tb_mux8_1;
```

```
wire o;
reg [7:0]a;
reg s0,s1,s2;

mux8_1 m1(o,s0,s1,s2,a);

initial
begin
$monitor($time,"o=%b,s0=%b,s1=%b,s2=%b,a=%b",o,s0,s1,s2,a);
#10  a[0]=1'b1;  a[1]=1'b0; a[2]=1'b0; a[3]=1'b0; a[4]=1'b0; a[5]=1'b0; a[6]=1'b0;
a[7]=1'b0; s0=1'b0;s1=1'b0; s2=1'b0;

#10  a[0]=1'b0;  a[1]=1'b1; a[2]=1'b0; a[3]=1'b0; a[4]=1'b0; a[5]=1'b0; a[6]=1'b0;
a[7]=1'b0; s0=1'b0;s1=1'b1;s2=1'b0;


#10  a[0]=1'b0;  a[1]=1'b0; a[2]=1'b0; a[3]=1'b0; a[4]=1'b0; a[5]=1'b0; a[6]=1'b1;
a[7]=1'b0; s0=1'b0;s1=1'b1; s2=1'b1;

end

endmodule
```

Simulated output using Mentor Graphics ModelSim tool



Synthesized output using Mentor Graphics Leonardo Spectrum tool



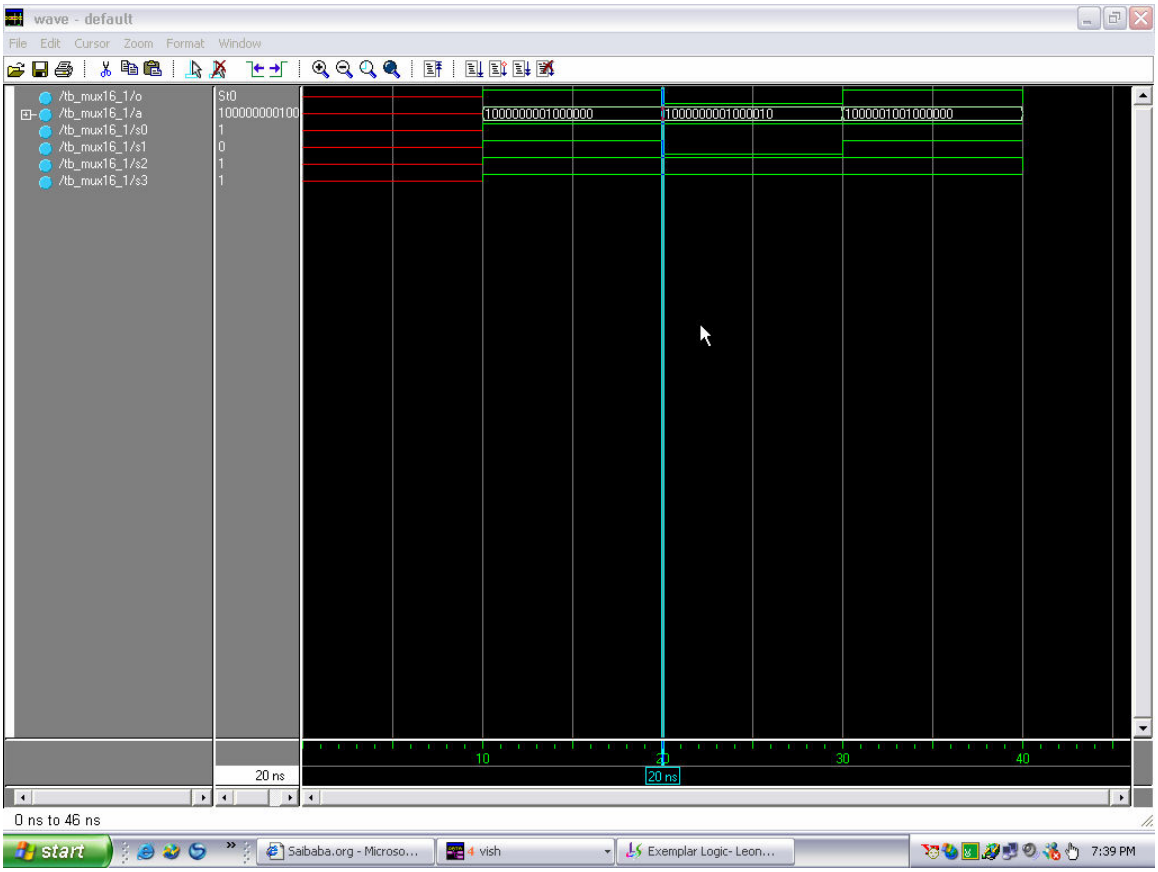
Verilog HDL code for 16 X 1 Multiplexer

```
module mux16_1 (out,s0,s1,s2,s3,i);

input [15:0]i;
input s0,s1,s2,s3;
output out;

wire [15:0]i;
wire s0,s1,s2;
wire out;
assign
out=s3?(s2?(s1?(s0?i[15]:i[14]):(s0?i[13]:i[12])):(s1?(s0?i[11]:i[10]):(s0?i[9]:i[8]))):
(s2?(s1?(s0?i[7]:i[6]):(s0?i[5]:i[4])):(s1?(s0?i[3]:i[2]):(s0?i[1]:i[0])));

endmodule
```

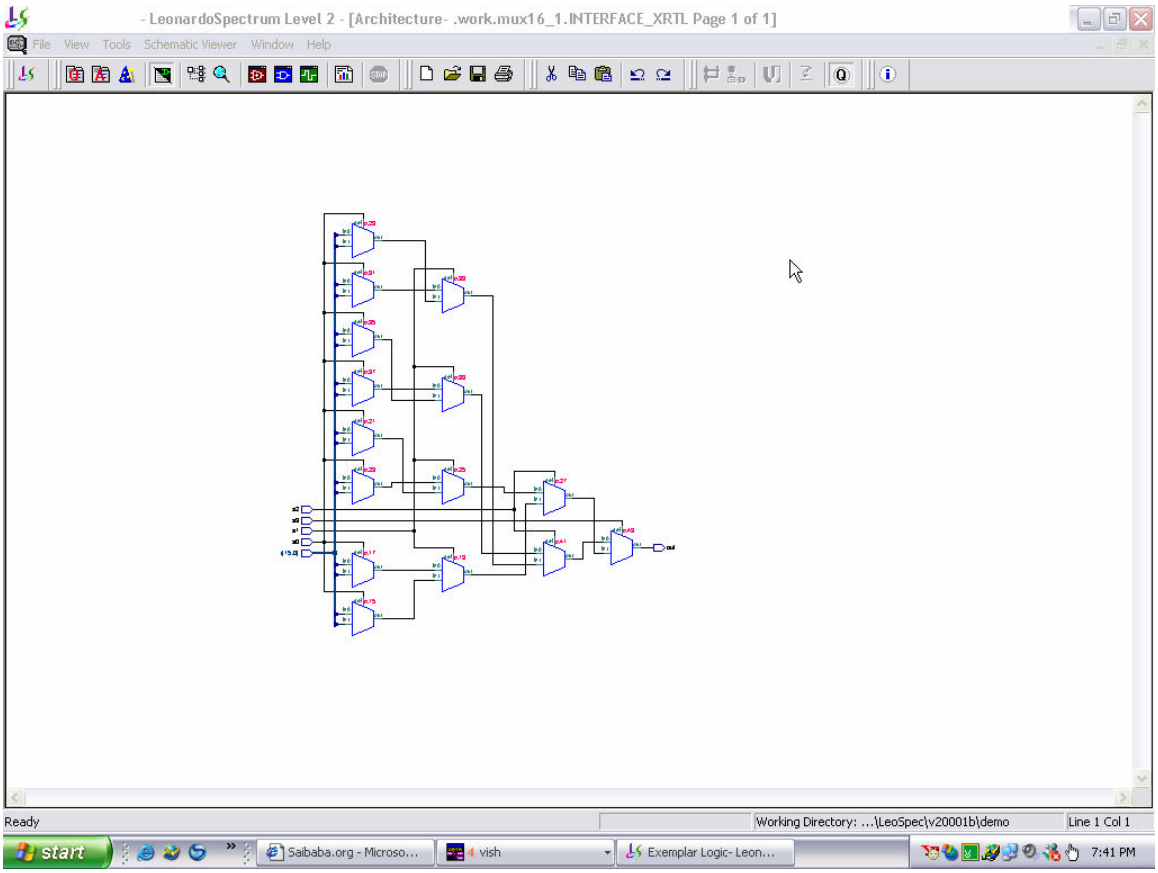
Test bench for 16 X 1 Multiplexer in Verilog HDL

```
module tb_mux16_1;
wire o;
reg [15:0]a;
reg s0,s1,s2,s3;
mux16_1 m1(o,s0,s1,s2,s3,a);
initial
begin
$monitor($time,"o=%b,s0=%b,s1=%b,s2=%b,a=%b",o,s0,s1,s2,a);
#10 a[15:0]=16'b1000000001000000;s0=1'b1;s1=1'b1;s2=1'b1;s3=1'b1;
#10 a[15:0]=16'b1000000001000010;s0=1'b1;s1=1'b0;s2=1'b1;s3=1'b1;
#10 a[15:0]=16'b1000001001000000;s0=1'b1;s1=1'b1;s2=1'b1;s3=1'b1;
#10 a[15:0]=16'b1000000001000000;s0=1'b1;s1=1'b1;s2=1'b0;s3=1'b1;
end
endmodule
```

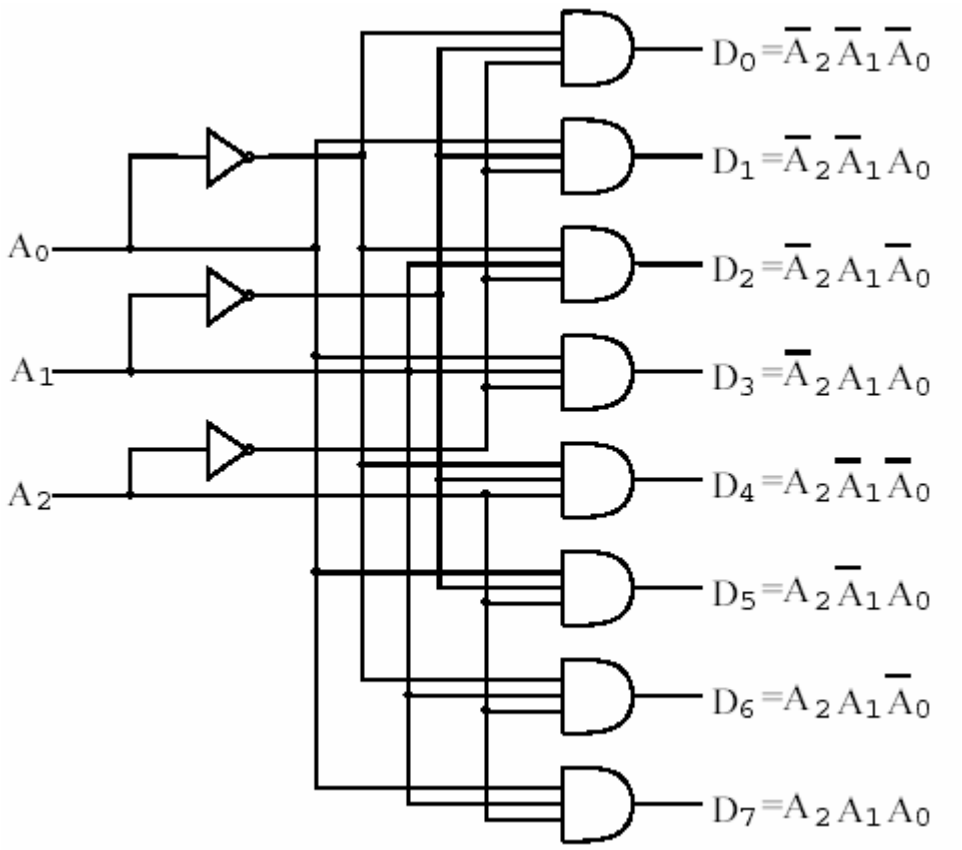
Simulated output using Mentor Graphics ModelSim tool



Synthesized output using Mentor Graphics Leonardo Spectrum tool


- LeonardoSpectrum Level 2 - [Architecture- .work.mux16_1.INTERFACE_XRTL Page 1 of 1]
File View Tools Schematic Viewer Window Help
Ready
Working Directory: ...\LeoSpec\v20001b\demo
Line 1 Col 1
start
Saibaba.org - Microso...
4 vish
Exemplar Logic- Leon...
7:41 PM

Decoder



Functionality

Inputs			Outputs							
A2	A1	A0	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

Verilog HDL code for Decoder 3 to 8

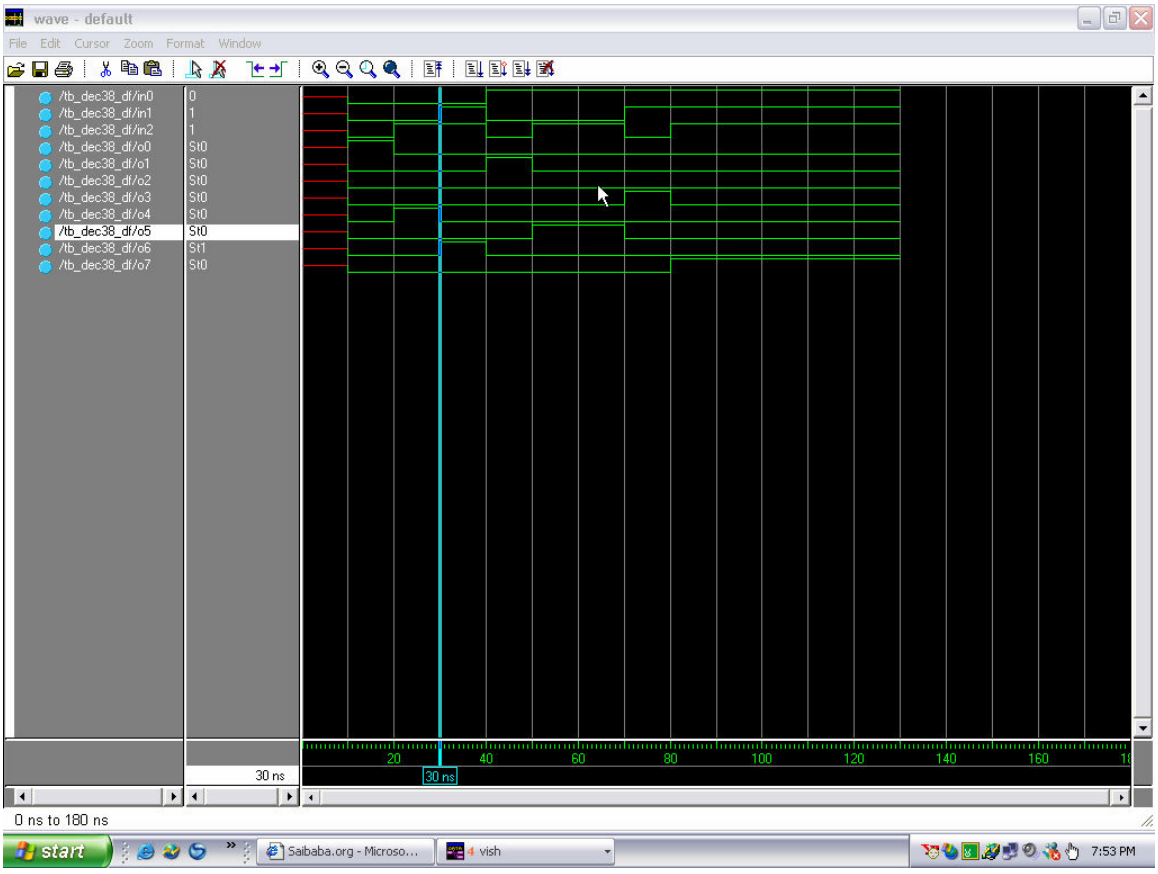
```
module dec38_df(o1,o2,o3,o4,o5,o6,o7,o8,i1,i2,i3);

output o1,o2,o3,o4,o5,o6,o7,o8;
input i1,i2,i3;
wire o1,o2,o3,o4,o5,o6,o7,o8,i1,i2,i3;
// local vars
 wire not_i3;
 assign not_i3 = ~i3;
dec24_df d1 (.out1(o1),.out2(o2),.out3(o3),.out4(o4),.in1(i1),.in2(i2),.sel(not_i3));

dec24_df d2 (.out1(o5),.out2(o6),.out3(o7),.out4(o8),.in1(i1),.in2(i2),.sel(i3));

endmodule
```

Test bench for Decoder 3 to 8 in Verilog HDL

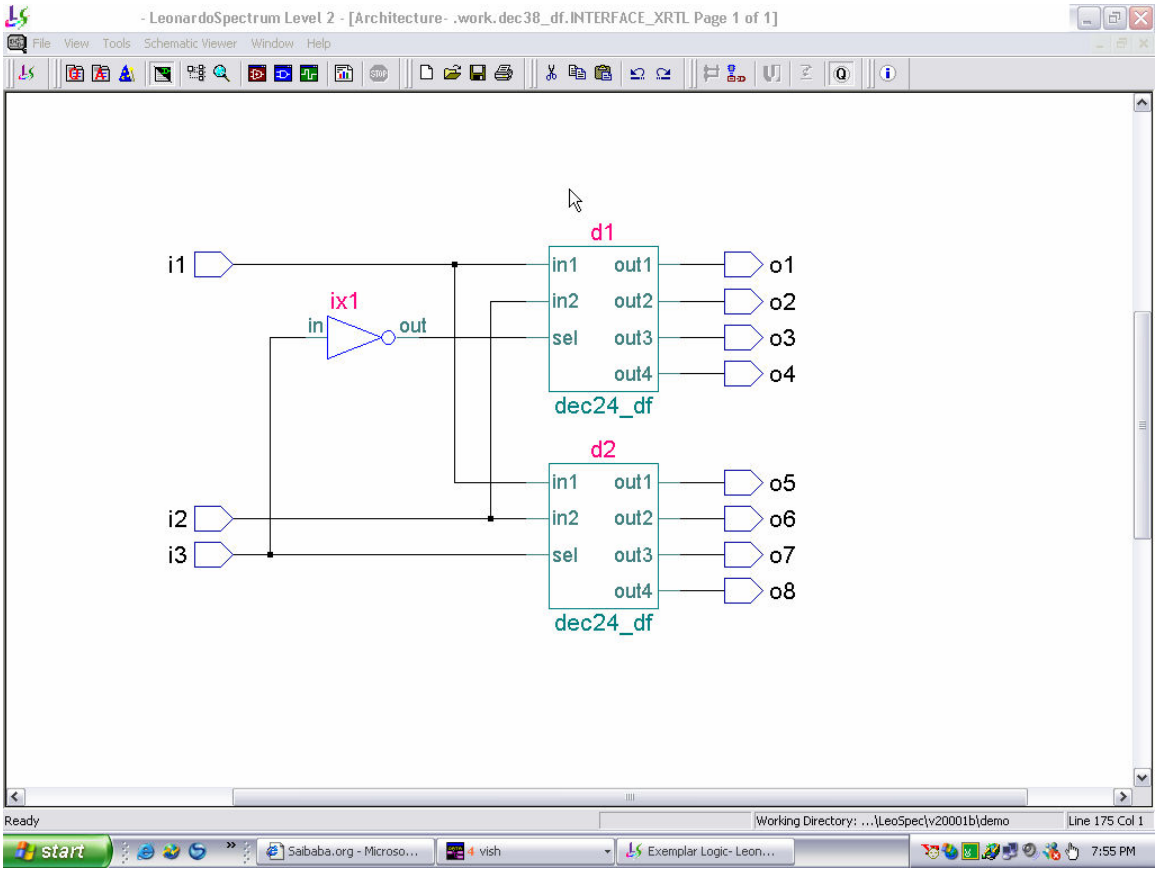
```
module tb_dec38_df;

reg in0,in1,in2;
wire o0,o1,o2,o3;

 // module instantiation
  dec38_df
d1(.o1(o0),.o2(o1),.o3(o2),.o4(o3),.o5(o4),.o6(o5),.o7(o6),.o8(o7),.i1(in0),.i2(in1),
.i3(in2));

//test bench

initial

begin

$monitor("time=%d,in1=%b,in2=%b,in2=%b,out1=%b,out2=%b,out3=%b,out4=%b,out
5=%b,out6=%b,
out7=%b,out8=%b",$time ,in0,in1,in2,o0,o1,o2,o3,o4,o5,o6,o7);



   #10 in0=1'b0 ; in1=1'b0;in2=1'b0;
   #10 in0=1'b0 ; in1=1'b0;in2=1'b1;
   #10 in0=1'b0 ; in1=1'b1;in2=1'b1;
   #10 in0=1'b1 ; in1=1'b0;in2=1'b0;
```

```
    #10 in0=1'b1 ; in1=1'b0;in2=1'b1;
    #10 in0=1'b1 ; in1=1'b0;in2=1'b1;
    #10 in0=1'b1 ; in1=1'b1;in2=1'b0;
    #10 in0=1'b1 ; in1=1'b1;in2=1'b1;

    #50  $stop;
end
endmodule
```

Simulated output using Mentor Graphics ModelSim tool



Synthesized output using Mentor Graphics Leonardo Spectrum tool



Verilog HDL code for Decoder 4 to 16

```
module dec416(i1,i2,i3,i4,en,out);

input i1,i2,i3,i4,en;
output [15:0]out;

reg [15:0]out;
wire i1,i2,i3,i4,en;

always @ ( en or i1 or i2 or i3 or i4 )

begin


case({en,i1,i2,i3,i4})
```

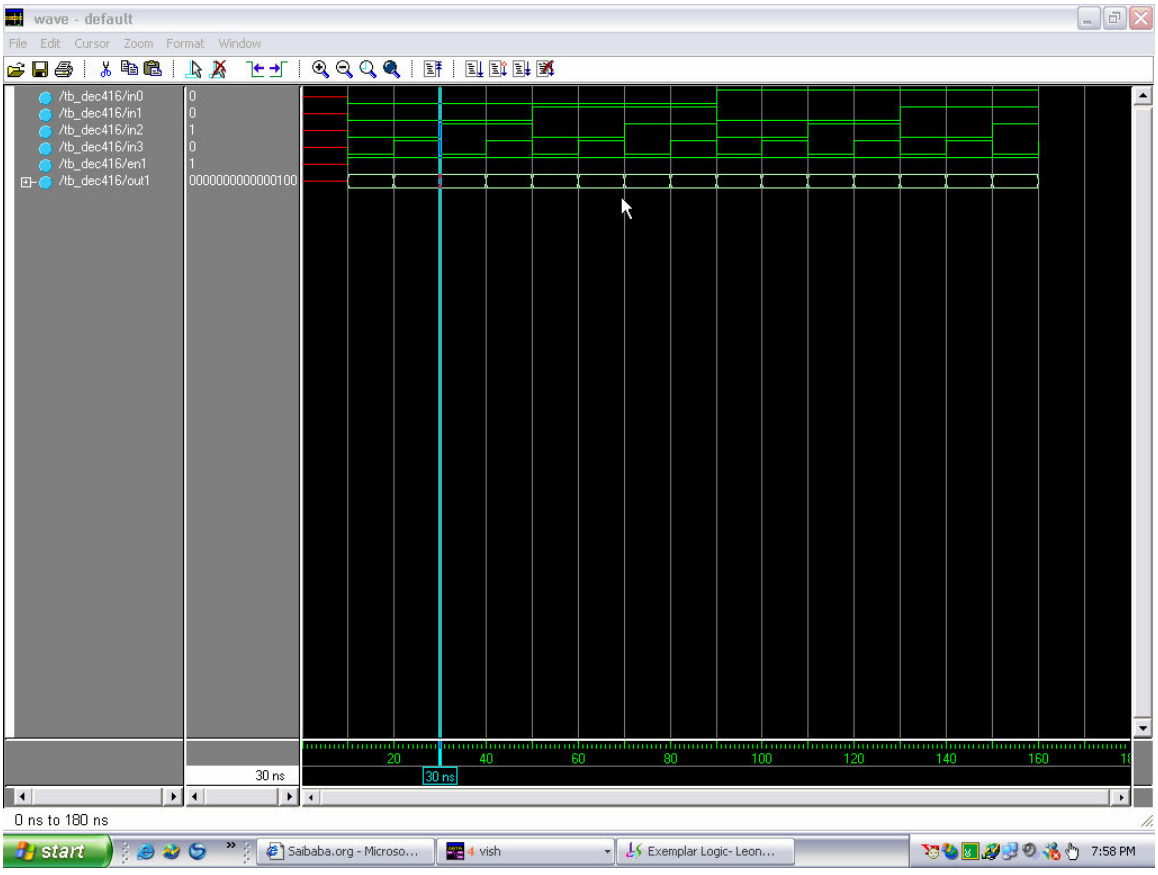
```
5'b10000:out=16'b0000000000000001;
5'b10001:out=16'b0000000000000010;
5'b10010:out=16'b0000000000000100;
5'b10011:out=16'b0000000000001000;
5'b10100:out=16'b0000000000010000;
5'b10101:out=16'b0000000000100000;
5'b10110:out=16'b0000000001000000;
5'b10111:out=16'b0000000010000000;
5'b11000:out=16'b0000000100000000;
5'b11001:out=16'b0000001000000000;
5'b11010:out=16'b0000010000000000;
5'b11011:out=16'b0000100000000000;
5'b11100:out=16'b0001000000000000;
5'b11101:out=16'b0010000000000000;
5'b11110:out=16'b0100000000000000;
5'b11111:out=16'b1000000000000000;

endcase
end

endmodule
```

Test bench for Decoder 4 to 16 in Verilog HDL

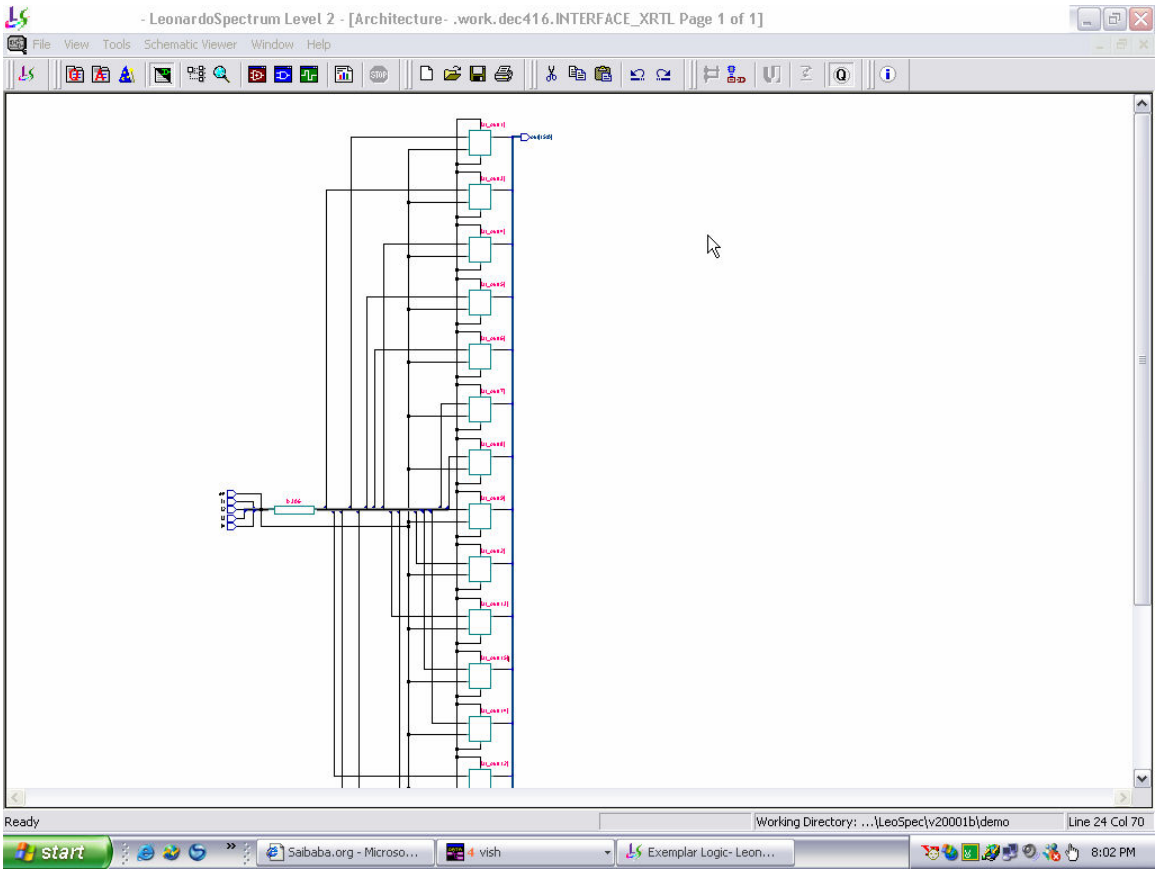
```
module tb_dec416;

reg in0,in1,in2,in3,en1;

wire [15:0]out1;

 // module instantiation
  dec416 d1(.i1(in0),.i2(in1),.i3(in2),.i4(in3),.en(en1),.out(out1));
//test bench

initial

begin

//
$monitor("time=%d,in1=%b,in2=%b,in2=%b,out1=%b,out2=%b,out3=%b,out4=%b,out
5=%b,out6=%b,out7=%b,out8=%b",$time ,in1,in2,in3,o1,o2,o3,o4,o5,o6,o7,o8);
    #10 in0=1'b0 ; in1=1'b0;in2=1'b0; in3=1'b0; en1=1'b1;
```

```
    #10 in0=1'b0 ; in1=1'b0;in2=1'b0; in3=1'b1; en1=1'b1;
    #10 in0=1'b0 ; in1=1'b0;in2=1'b1;in3=1'b0; en1=1'b1;
    #10 in0=1'b0 ; in1=1'b0;in2=1'b1;in3=1'b1; en1=1'b1;
    #10 in0=1'b0 ; in1=1'b1;in2=1'b0;in3=1'b0; en1=1'b1;
    #10 in0=1'b0 ; in1=1'b1;in2=1'b0;in3=1'b1; en1=1'b1;
    #10 in0=1'b0 ; in1=1'b1;in2=1'b1;in3=1'b0; en1=1'b1;
    #10 in0=1'b0 ; in1=1'b1;in2=1'b1;in3=1'b1; en1=1'b1;
    #10 in0=1'b1 ; in1=1'b0;in2=1'b0;in3=1'b0; en1=1'b1;
    #10 in0=1'b1 ; in1=1'b0;in2=1'b0;in3=1'b1; en1=1'b1;
    #10 in0=1'b1 ; in1=1'b0;in2=1'b1;in3=1'b0; en1=1'b1;
    #10 in0=1'b1 ; in1=1'b0;in2=1'b1;in3=1'b1; en1=1'b1;
    #10 in0=1'b1 ; in1=1'b1;in2=1'b0;in3=1'b0; en1=1'b1;
    #10 in0=1'b1 ; in1=1'b1;in2=1'b0;in3=1'b1; en1=1'b1;
    #10 in0=1'b1 ; in1=1'b1;in2=1'b1;in3=1'b0; en1=1'b1;
    #10 in0=1'b1 ; in1=1'b1;in2=1'b1;in3=1'b1; en1=1'b1;
    end
endmodule
```

Simulated output using Mentor Graphics ModelSim tool



Synthesized output using Mentor Graphics Leonardo Spectrum tool



Shift Register

Functionality

function	inputs		Next state		
	S0	S1	Q[2]	Q[1]	Q[0]
hold	0	0	Q[2]	Q[1]	Q[0]
Shift left	0	1	Q[1]	Q[0]	RightIn
Shift right	1	0	Left In	Q[2]	Q[1]
load	1	1	ParIn[2]	ParIn[1]	ParIn[0]

Verilog HDL code for 4 bit Shift Register

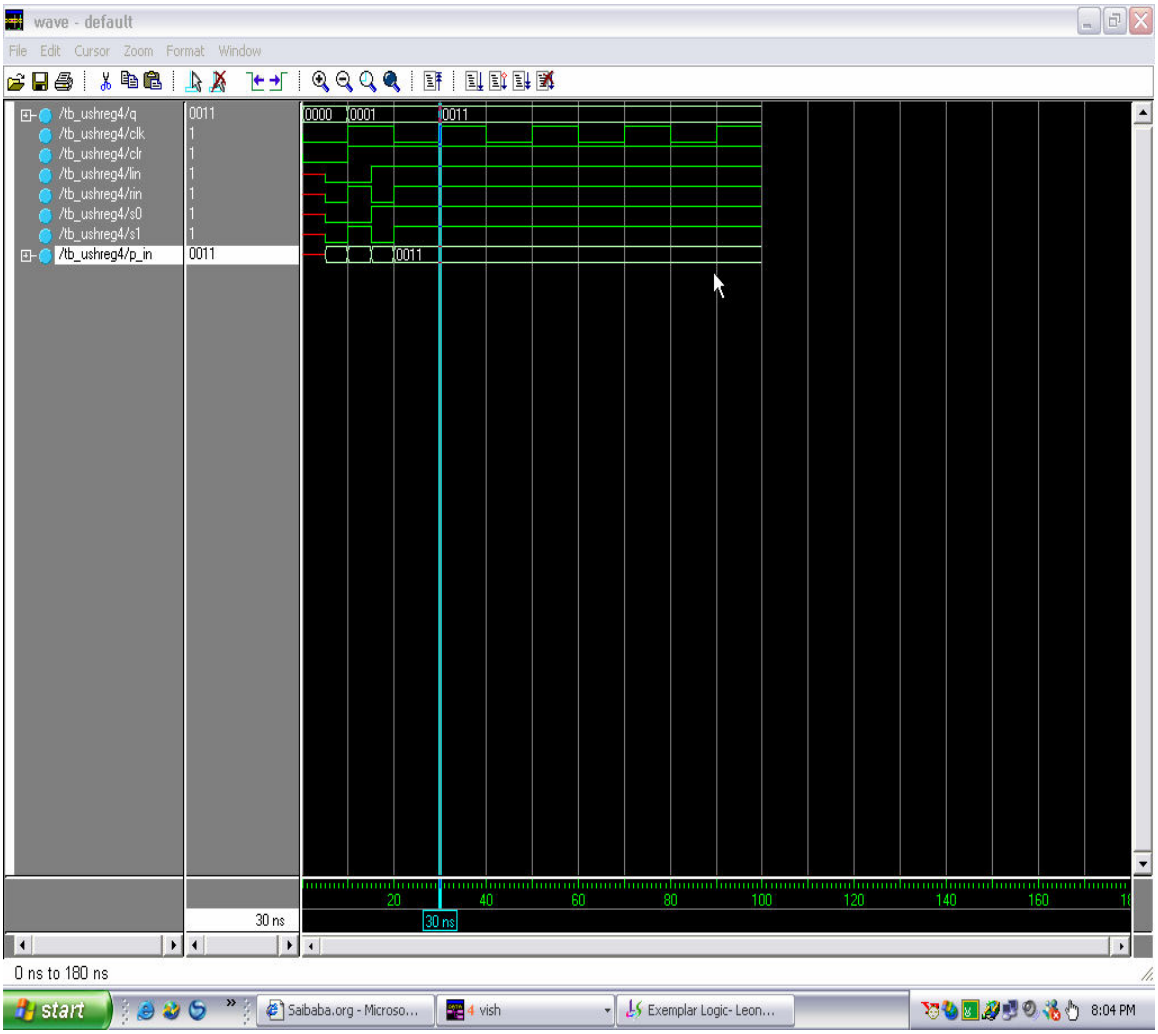
```
module ushreg4(clk,clr,lin,rin,s0,s1,p_in,q);
input clk,clr,lin,rin,s0,s1;
input[3:0]p_in;
output [3:0]q;
reg [3:0]q;
always@(negedge clr or posedge clk)
if(!clr)
q<=4'b0;
else
case({s0,s1})
2'b00:;
2'b01:
q<={q[1:0],rin};
2'b10:
q<={lin,q[2:1]};
2'b11:
q<=p_in;
endcase
endmodule
```

Test bench for 4 bit shift register in Verilog HDL

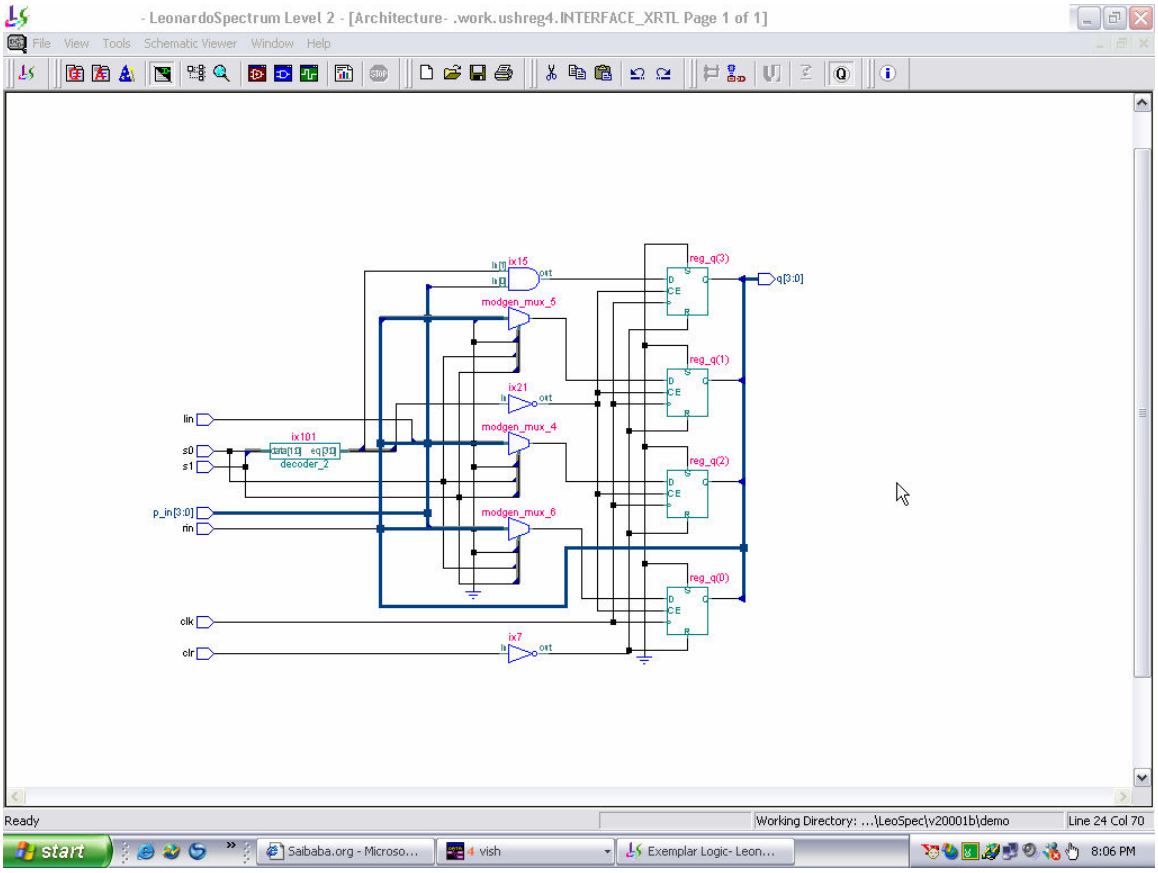
```
module tb_ushreg4;
wire [3:0]q;
reg clk,clr,lin,rin,s0,s1;
reg [3:0]p_in;
ushreg4 u1(clk,clr,lin,rin,s0,s1,p_in,q);
initial
$monitor($time,"clk=%b,clr=%b,lin=%b,rin=%b,s0=%b,s1=%b,p_in=%b,q=%b",clk,clr,
lin,rin,s0,s1,p_in,q);
initial
begin
clk=1'b0;
clr=1'b0;
#10 clr=1'b1;
end
initial
forever #10 clk=~clk;
initial
#100 $finish;
initial
begin
#5 lin=1'b0;rin=1'b0;s0=1'b0;s1=1'b0;p_in=4'b0;
```

```
#5 lin=1'b0;rin=1'b1;s0=1'b0;s1=1'b1;p_in=4'b0001;
#5 lin=1'b1;rin=1'b0;s0=1'b1;s1=1'b0;p_in=4'b0010;
#5 lin=1'b1;rin=1'b1;s0=1'b1;s1=1'b1;p_in=4'b0011;
end
endmodule
```

Simulated output using Mentor Graphics ModelSim tool



Synthesized output using Mentor Graphics Leonardo Spectrum tool



Verilog HDL code for 8 bit Shift Register

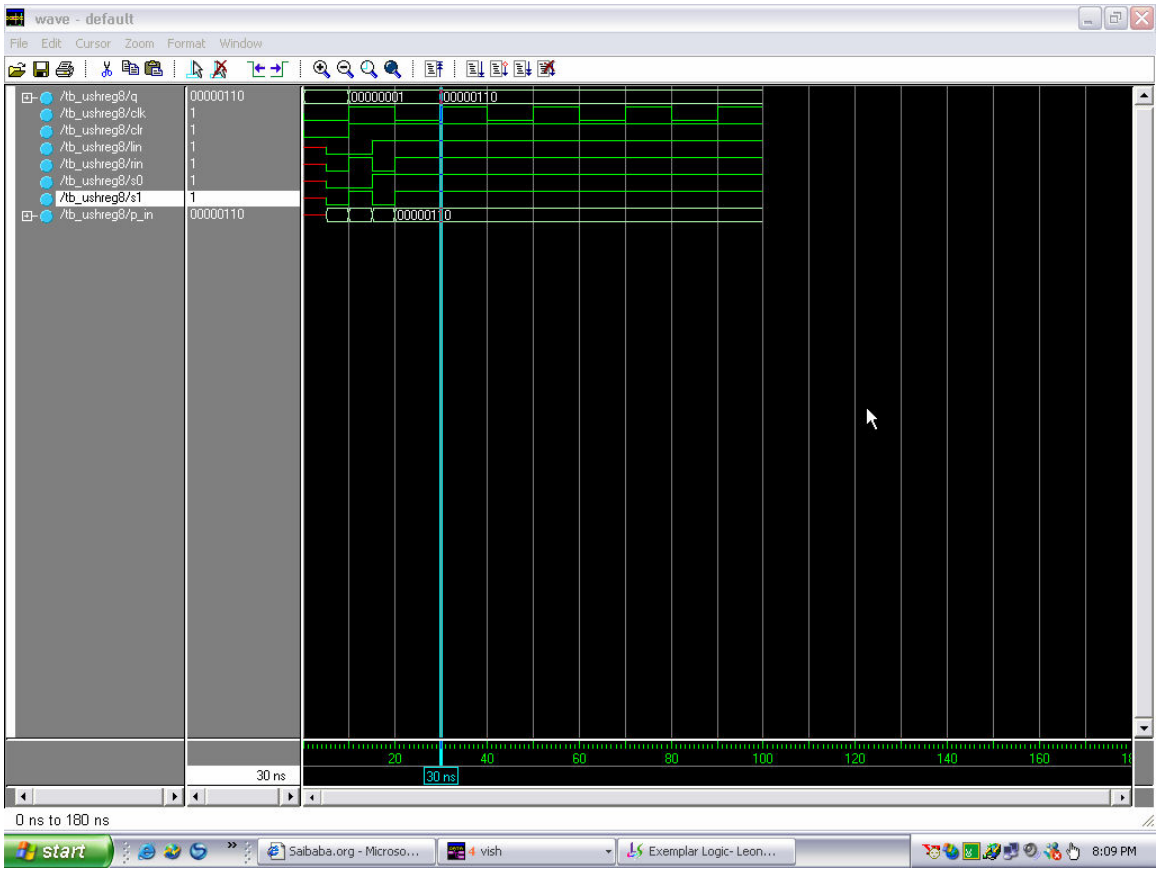
```
module ushreg8(clk,clr,lin,rin,s0,s1,p_in,q);
input clk,clr,lin,rin,s0,s1;
input[7:0]p_in;
output [7:0]q;
reg [7:0]q;
always@(negedge clr or posedge clk)
if(!clr)
q<=8'b0;
else
case({s0,s1})
2'b00:;//hold
2'b01://shift left
q<={q[1:0],rin};
2'b10://shift right
q<={lin,q[2:1]};
```

```
2'b11://load
q<=p_in;
endcase
endmodule
```

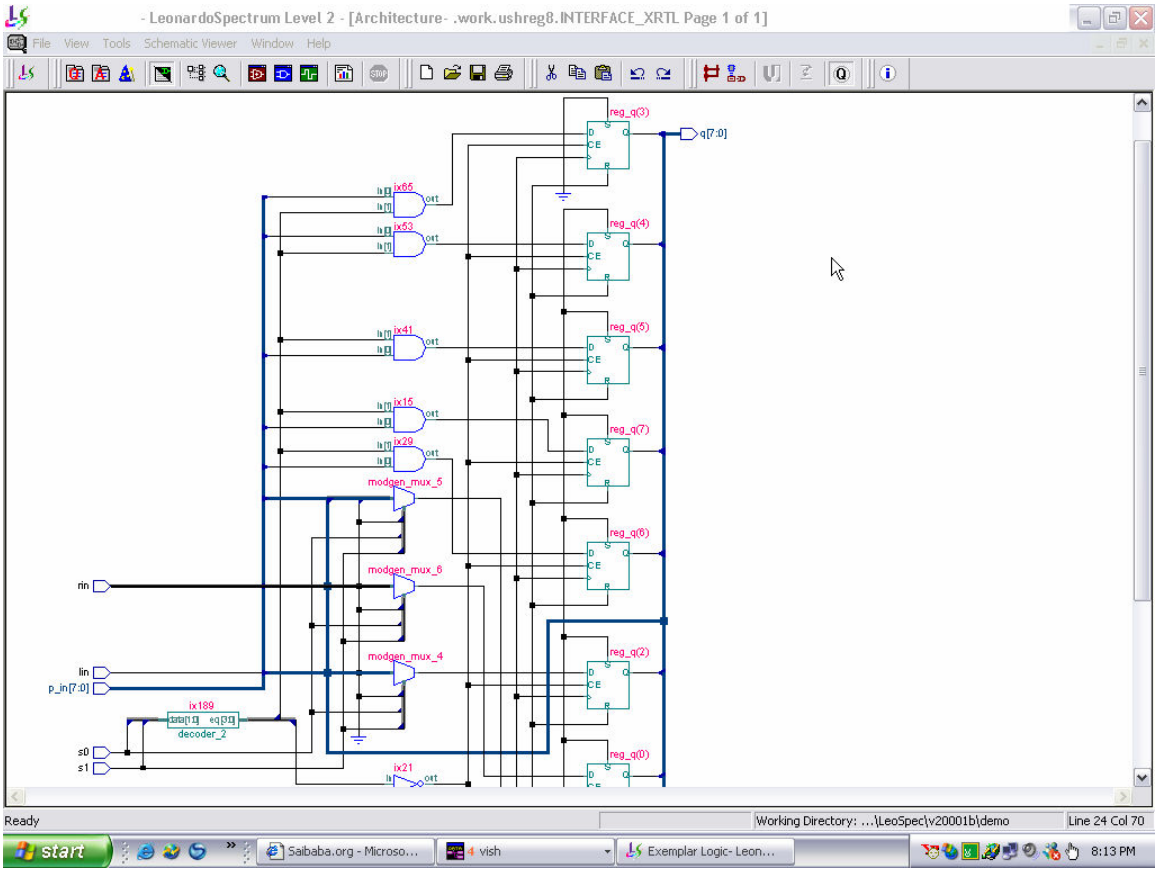
Test bench for 8 bit shift register in Verilog HDL

```
module tb_ushreg8;
wire [7:0]q;
reg clk,clr,lin,rin,s0,s1;
reg [7:0]p_in;
ushreg8 u1(clk,clr,lin,rin,s0,s1,p_in,q);
initial
$monitor($time,"clk=%b,clr=%b,lin=%b,rin=%b,s0=%b,s1=%b,p_in=%b,q=%b",clk,clr,
lin,rin,s0,s1,p_in,q);
initial
begin
clk=1'b0;
clr=1'b0;
#10 clr=1'b1;
end
initial
forever #10 clk=~clk;
initial
#100 $finish;
initial
begin
#5 lin=1'b0;rin=1'b0;s0=1'b0;s1=1'b0;p_in=8'b01;
#5 lin=1'b0;rin=1'b1;s0=1'b0;s1=1'b1;p_in=8'b10;
#5 lin=1'b1;rin=1'b0;s0=1'b1;s1=1'b0;p_in=8'b11;
#5 lin=1'b1;rin=1'b1;s0=1'b1;s1=1'b1;p_in=8'b110;
end
endmodule
```

Simulated output using Mentor Graphics ModelSim tool



Synthesized output using Mentor Graphics Leonardo Spectrum tool



Barrel Shifter

Functionality

- shifts(actually rotates) the input data
- rotates by the specified number bits in a combinational manner

VHDL code for Barrel Shifter

```
--Develop a VHDL model for a Barrel shifter that rotates the input data by the
--specified number of bits.
library ieee;
use ieee.std_logic_1164.all;
entity Barrel is
port (clk,reset,load :in std_logic;
      data_in:in std_logic_vector (7 downto 0);
      data_out:out std_logic_vector (7 downto 0));
```

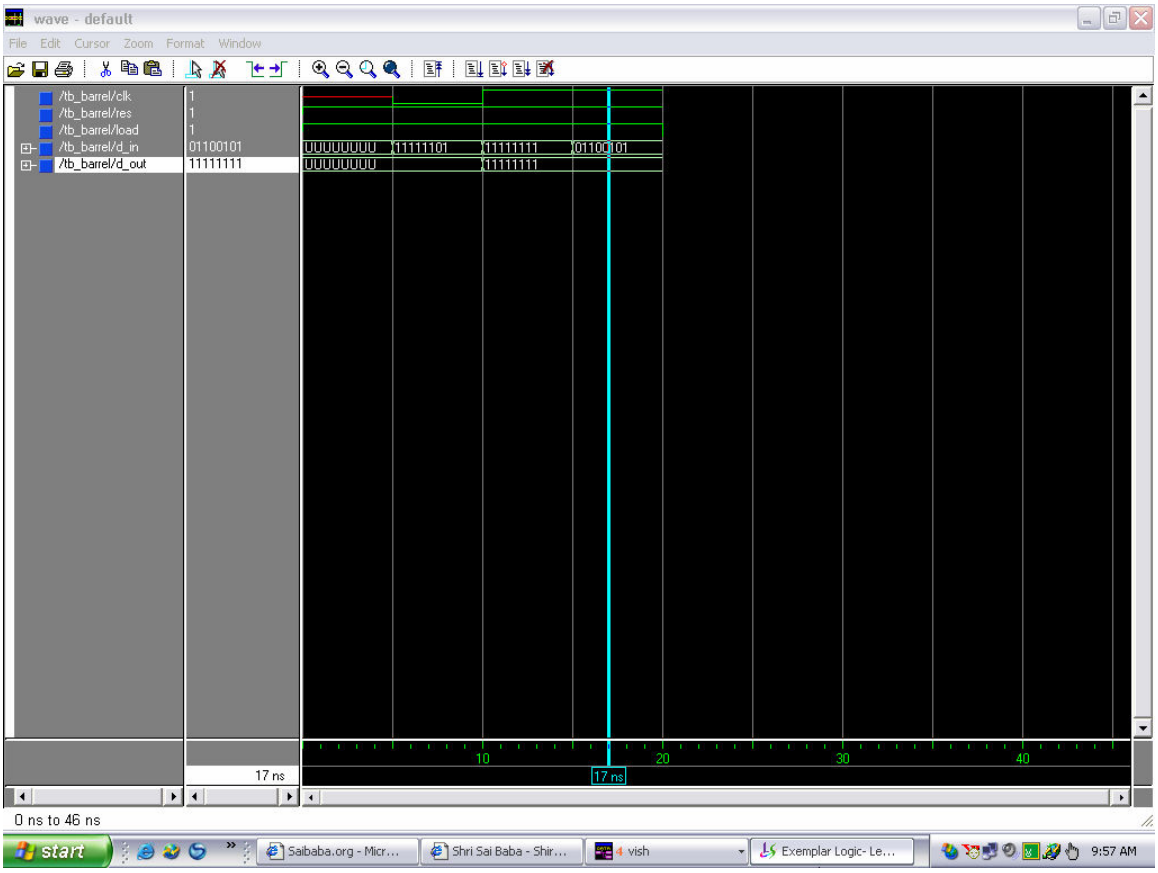
```
end Barrel;

architecture  Barrel of Barrel is
begin
process (reset,clk,data_in)
variable d: std_logic_vector (7 downto 0);
begin
  if (reset = '0') then
     d:=(others => '0');
   elsif (clk'event and clk = '1') then
      if (load ='1') then
       d:=data_in;
        else
      d:= d(0) & d(7 downto 1) ;
      end if;
   end if;
data_out<= d;
end process;
end Barrel;
```

Test bench for Barrel Shifter in VHDL

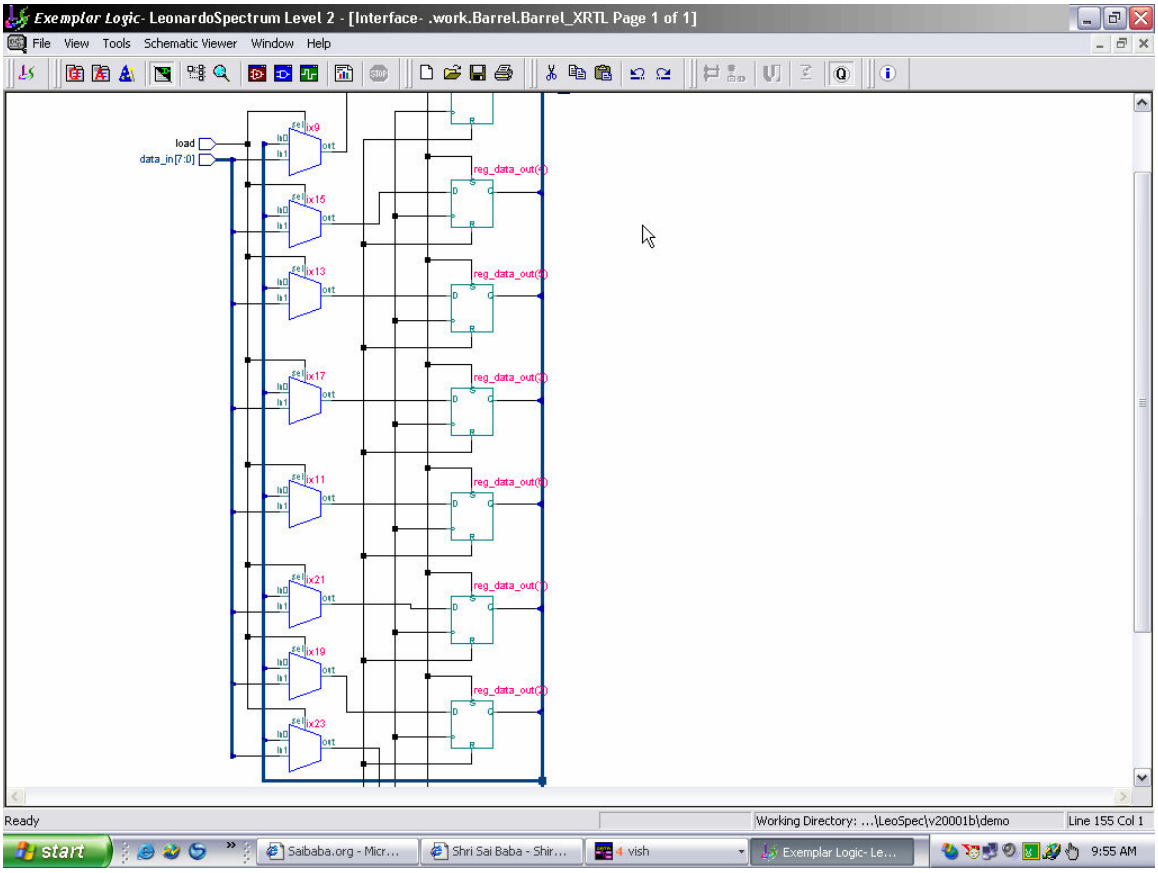
```
--test bench for barrel shifter
library ieee;
use ieee.std_logic_1164.all;

entity tb_Barrel is
end tb_Barrel;

architecture tb_Barrel of tb_Barrel is
component Barrel
port (clk,reset,load :in std_logic;
      data_in:in std_logic_vector (7 downto 0);
      data_out:out std_logic_vector (7 downto 0));
end component;
signal clk,res,load : std_logic;
signal d_in,d_out : std_logic_vector (7 downto 0);
begin
br: Barrel port map (clk,res,load,d_in,d_out);
process
begin
res <= '1'; load <= '1';

wait for 5 ns;
```

```
load <= '1';
clk<='0';
d_in<="11111101";
wait for 5 ns;
clk<='1';
d_in<="11111111";

wait for 5 ns;
res <= '1';
d_in <= "01100101";
load <= '1';
wait for 5 ns;
load <= '0';
wait;
end process;

end tb_Barrel;
```

Simulated output using Mentor Graphics ModelSim tool



Synthesized output using Mentor Graphics Leonardo Spectrum tool



Conclusions

Digital circuits like ALU, Parallel to Serial Converter, Multiplexers, Decoders, Shift Registers and Barrel shifter unit are successfully implemented in Hardware description languages like Verilog HDL and VHDL, simulated using Mentor Graphics ModelSim and synthesized using Mentor Graphics Leonardo Spectrum tools.