# **FPGA Based System Design**

## Lab #6: Synthesis of Verilog code with Leonardo Spectrum

### **Objectives:**

• To synthesize Verilog code using Leonardo Spectrum from Mentor Graphics.

### **1. Introduction**

Verilog HDL is a high level description language for system and circuit design. The language supports various levels of abstraction. Where a regular netlist format supports only structural description, Verilog supports a wide range of description styles. This includes structural descriptions, data flow descriptions and behavioral descriptions.

The structural and data flow descriptions show a concurrent behavior. All statements are executed concurrently, and the order of the statements does not matter. On the other hand, behavioral descriptions are executed sequentially in always blocks, tasks and functions in Verilog. The behavioral descriptions resemble high-level programming languages.

Verilog allows a mixture of various levels of design entry. LeonardoSpectrum synthesizes all levels of abstraction, and minimizes the amount of logic needed, resulting in a final netlist description in the technology of your choice.

### 2. Synthesis (Leonardo Spectrum)

1. The Verilog code used for this lab is as follows:

module mux\_21 (a,b,sel,y);

input a, b; output y; input sel; wire y;

assign y = (sel) ? b : a;

endmodule

- Open Leonardo Spectrum. (Start>All Programs>Leonardo Spectrum LS2009a\_6> Leonardo Spectrum LS2000a\_6)
- 3. Choose Level 3.

P	FPGA Synti Multi-Million-Gate ease select a license to check out:	
L E	<ul> <li>LeonardoSpectrum Level 1</li> <li>Actel</li> <li>LeonardoSpectrum Level 2</li> <li>LeonardoSpectrum Level 3</li> <li>Run license selection next time</li> <li>OK Cancel</li> </ul>	n
	Copyright 1990-2009 N	

4. Click on the 'Toggle Advanced FlowTabs' icon in the toolbar near the top of the window to enter the Advanced mode.

🎸 <u>F</u> ile	Edit View Tools	<u>W</u> i
45	🖄 🛓 🖻	[ ]
	$\overline{\bigcirc}$	

- 5. Type the command "set exclude\_gates {PadOut PadInC}" in the command window in the bottom right panel. This command excludes the gates PadInC and PadOut from being imported when we load the library in the next step.
- 6. The Technology tab should be selected. Choose TSMC 0.35u (typ) ( or AMI 0.5u) library and load it.

Technology Input Constraints	Optimize Report Output
technology logo to open the ven	vice tree and select a library. Click on the sel dor website. Use all defaults.Set advanced te ps". Press "Apply" or "Load Library" to apply : Temperature: Voltage: Process: -1
	Load Library Help

7. Click the Input tab. Set the working directory to the directory with the Verilog file, and then click on Open Files button to open the *mux2\_1.v* file. Load the file by clicking the Read button.

Technology	Input	Constraints	Optimize	Report	Output		
Click Working Directory -> Set Working Directory to establish a path. Click Op Set Input File(s) to open a file. Click LMB on file name to open file information. on file name to open pulldown with more file choices. Double click LMB on file open HDLInventor. Use Windows drag and drop attributes to arrange files. Us defaults.							
Working Dire	ectory: [ .	\Desktop\D	)E.lab3 ]				
mux2_1.v							
				Open fil	es: 🗾	5	00 00 00
Encoding Style: © Binary © Onehot © Twohot © Random © Gray © Auto							
🔽 Resource Sharing 🛛 🔽 Run Elaborate 🔽 Run Pre-Optimization							
Input File Se	arch Pat	h:					
<b>2</b>	🗖 Ad	d Before					
			Ç	Read		Help	

- 8. Move on to the Optimize tab, and click optimize.
- Go to the Output tab, specify the name of the output file as 'syn\_mux2\_1.v', and choose the output format as Verilog. Check *Down-to* selection as Primitives. Click Write to export a synthesized Verilog netlist to your working directory. You are done with synthesis.

10. The synthesized verilog netlist will read:

```
//
// Verilog description for cell mux_21,
// 02/14/12 10:51:43
//
// LeonardoSpectrum Level 3, 2009a.6
//
module mux_21 ( a, b, sel, y ) ;
    input a ;
    input a ;
    input b ;
```

```
input sel ;
output y ;
mux21_ni ix7 (.Y (y), .A0 (a), .A1 (b), .S0 (sel)) ;
endmodule
module mux21_ni ( Y, A0, A1, S0 ) ;
output Y ;
input A0 ;
input A1 ;
input S0 ;
wire NOT_S0, nx2, nx4;
assign NOT_S0 = ~S0 ;
and (nx2, A0, NOT_S0) ;
and (nx4, A1, S0) ;
or (Y, nx2, nx4) ;
endmodule
```

#### 3. What to hand-in

Submit The synthesized verilog netlist for the following codes.

1. Following is the Verilog code for a combinational logic.

```
module comb1(a,b,c);
input a,b;
output c;
reg c;
always @ (a or b)
begin
    if ((a==1) && (b==1))
        c = 1'b1;
    else
        c = 1'b0;
end
endmodule
```

2. Following is the Verilog code for 1-bit adder

```
module adder(a, b, ci, sum,carry);
input a;
input b;
input ci;
output sum,carry;
assign {sum,carry} = a + b + ci;
```

endmodule

3. Following is the Verilog code for an unsigned 4-bit greater or equal comparator.

```
module compar(a, b, cmp);
input [4:0] a;
input [4:0] b;
output cmp;
assign cmp = (a >= b) ? 1'b1 : 1'b0;
endmodule
```

4. Following is the Verilog code for a logical shifter.

```
module lshift (di, sel, so);
       input [2:0] di;
 input [1:0] sel;
 output [2:0] so;
 reg
      [2:0] so;
 always @(di or sel)
 begin
    case (sel)
       2'b00 : so = di;
       2'b01 : so = di << 1;
       2'b10 : so = di << 2;
       default : so = di \ll 3;
    endcase
 end
endmodule
```

5. Following is the Verilog code for an unsigned comparator

```
module synthesis_compare_xz (a,b);
output a;
input b;
reg a;
always @ (b)
begin
    if ((b == 1'bz) || (b == 1'bx)) begin
        a = 1;
    end else begin
        a = 0;
    end
end
endmodule
```

6. Following is the Verilog code for combinational logic

```
module comb2(a,b,g,q);
input a,b,g;
output q;
reg q;
always@(g, a, b)
begin
if (g == 1'b1)
q = 0;
else if (a == 1'b1)
```

```
q = b;
end
endmodule
```

7. Write a code using *for loop* and synthesize it.