

Assignment #1

Objectives:

The aim of this laboratory is to introduce to VERILOG Dataflow modelling using the Xilinx Vivado 2015.2

Introduction

This lab introduces the Xilinx Vivado 2015.2 tools to create a simple digital circuit using VERILOG. The design flow consists of creating the VERILOG model and perform simulation (functional and timing).

Steps:

1. Create a Vivado Project

- Open Vivado2015.2 by double clicking the icon.
- Click **Create New Project** to start the wizard. You will see *Create A New Vivado Project* dialog box. Click **Next**.
- Click the Browse button of the *Project location* field of the **New Project** form, browse to **D/DSDLab/ur_rollNos**, and click **Select**.
- Enter **lab1** in the *Project name* field. Make sure that the *Create Project Subdirectory* box is checked. Click **Next**.
- Select **RTL Project** option in the *Project Type* form, and click **Next**.
- Select **VERILOG** as the *Target Language* and *Simulator Language* in the *Add Sources* form and click **Next**
- Since we do not have any IP to add, click **Next** at the *Add Existing IP* form
- Click **Next** at the *Add Constraints* form.
- In the *Default Part* form, use the **Parts** option and various drop-down fields of the **Filter** section select the **xc7a100tcsg324-1** part. Click **Next**

(Note: This is not important here as the design will not be ported to the hardware).

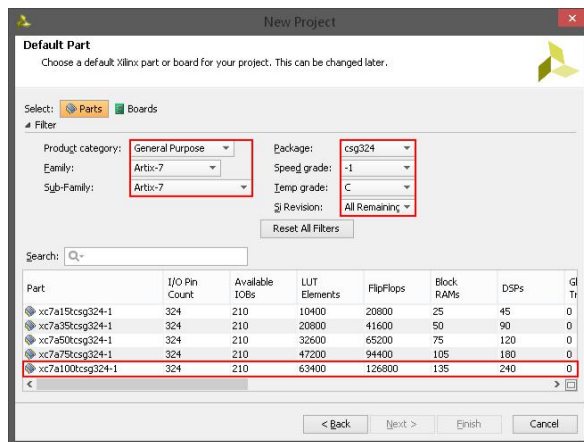


Fig. 7.1 Part Selection

- Click **Finish** to create the Vivado project
- Use Windows Explorer and look at the **D/DSDLab/ur_rollNos/lab1** directory. You will find that the lab1.cache, lab1.hw and lab1.sim directories and the lab1.xpr (Vivado) project file have been created. The lab1.cache directory is a place holder for the Vivado program database.
- The project manager is the base for anything you will want to do with your project. From it you can do circuit design, simulate/test your design, prepare your circuit for downloading to hardware, and quite a few other things which we will cover in future labs. It consists of five sub windows.

1. Creating the design

- **Project Manager – Add Sources – Select** *Add or Create Design Sources*
- Click on **+** and then select *Create File*
- Select *File type - VERILOG* *File name – CombCkt*, and *File location – Local to project –* Click *Finish*
- In *Define Module* form
 - Module – CombCkt
 - Ports Name - A,B, C, D : in ; F1, F2, F3: out.

The Wizard creates the ports and gives you a template in VERILOG in which you can enter your design

Open the file CombCkt.v and edit the architecture portion module .

```
module CombCkt(A,B,C,D,F1,F2,F3);  
    input A, B, C,D;  
    output F1, F2, F3;  
  
    assign F1 = A ^ B;  
    assign F2 = A | B & C | D;  
    assign F3 = F1 & F2 ;  
endmodule
```

Fig. 7.2 Example Verilog Program

- *In case there are any syntax errors it will be shown in Messages subwindow when you save the file.*

2. Simulation

In this section, we will introduce the concept of test bench and show how to verify the function of our CombCkt by behavioral simulation.

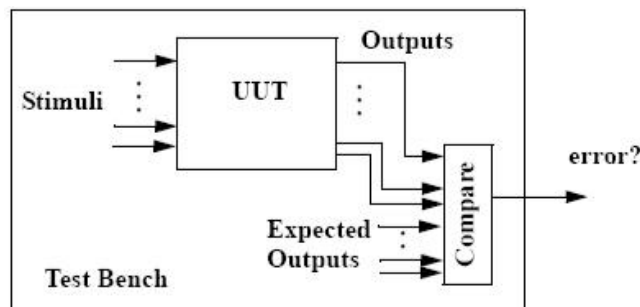


Fig. 7.3 Conceptual diagram of test bench

A test bench is an entity (usually a VHDL/Verilog program) which is used to verify the correctness of a design. The design to be verified is called Unit Under Test (UUT). The test bench supplies stimuli to the design, observes the outputs of the design, and compares the observed outputs with the expected values. If any mismatch happens, the test bench issues certain messages signifying that there are errors in the design. Fig. 3 shows the concept of test bench.

Create Testbench

- Click Add sources under *Project Manager* and select *Add or Create Simulation Sources*
- Click on **+** and then select *Create File*
- Select *File type - VERILOG* *File name - fulladd_tb*, and *File location – Local to project –* Click *Finish*.
- In *Define Module* form Click **Finish**.
- Open the file *fulladd_tb.v* and edit the architecture portion

```


module CombCkt_tb;
    reg a, b, c, d;
    wire f1, f2, f3;

    CombCkt uut(
        .A(a), .B(b), .C(c), .D(d), .F1(f1), .F2(f2), .F3(f3) );

    initial
    begin
        a = 0; b = 0; c = 0; d=0;
        #10
        a = 0; b = 0; c = 0; d=1;
        #10
        a = 0; b = 0; c = 1; d=0;
        #10
        a = 0; b = 0; c = 1; d=1;
        #10
        a = 0; b = 0; c = 1; d=1;
        #10
        a = 1; b = 0; c = 1; d=0;
    end
endmodule

```

Fig. 7.4 Example Verilog Testbench Program

- Select **fulladd_tb** under the **Simulation sources** group and click on **Run Simulation** under the *Project Manager* tasks of the *Flow Navigator* pane. Select the **Run Behavioral Simulation** option.
- The testbench and source files will be compiled and the Vivado simulator will run (assuming no errors). You will see a simulator output similar to the one shown below. The simulation is run for 1000ns (default). This can be changed in *Simulation Settings – Simulation*
- Click on the zoom-fit button  located left of the waveform window to see the entire waveform. Check the outputs.

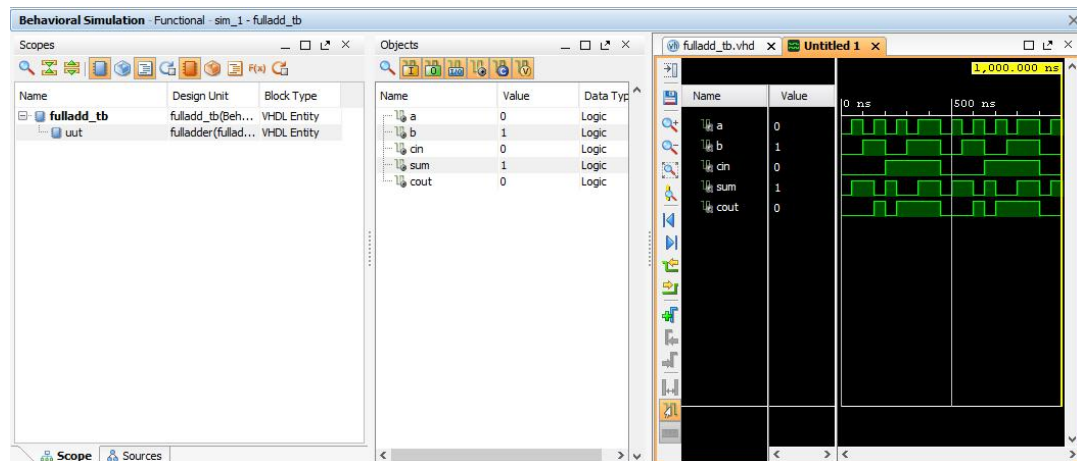


Fig. 7.5 Simulator output

3. Synthesizing the Design

After the design files have been successfully analysed, the next step is to translate the design into gates and optimise it for target architecture.

- Select **fulladder** under the *Design Sources* group and Click on *Run Synthesis* under the *Synthesis tasks* of the *Flow Navigator* pane.

- The synthesis process will be run on the fulladder.vhd file (and all its hierarchical files if they exist). When the process is completed a *Synthesis Completed* dialog box with three options will be displayed.
- Select the *Open Synthesized Design* option and click **OK** as we want to look at the synthesis output before progressing to the implementation stage.
- In The *Flow Navigator*, under *Synthesis* (expand *Synthesized Design* if necessary), click on **Schematic** to view the synthesized design in a schematic view.
- Notice that IBUFs and OBUFs are automatically instantiated (added) to the design as the input and output are buffered.

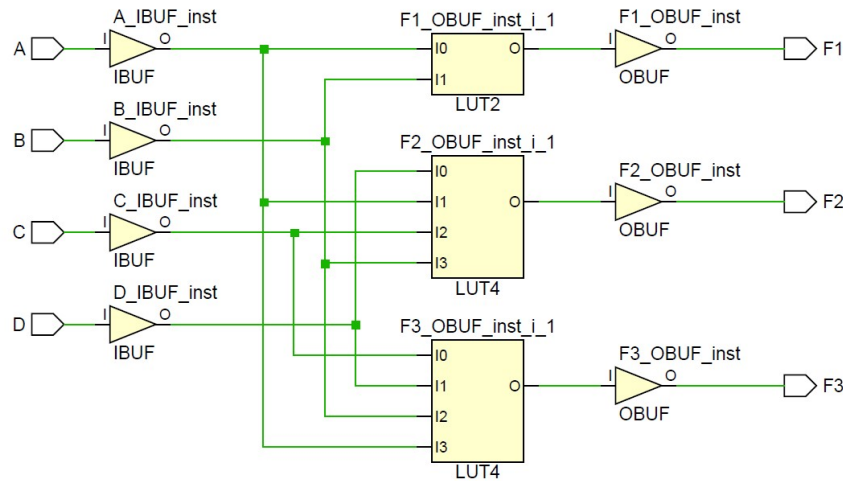


Fig. 7.6 Synthesized design's schematic view

Exercise

1. Write a Verilog data flow model using binary operators for implementing a full adder. Also check the function of the above circuit by writing testbench program.
2. Write a Verilog structural model using logic gates for implementing a full adder. Perform behavioral simulation to check the functionality of the designed circuit.
3. Design a 4:1 MUX
 - (a) Using continuous assignment & conditional operators.
 - (b) Using continuous assignment & binary operators.
 Perform behavioral simulation to check the functionality of the designed circuit.
4. Design a 4:2 Priority Encode using continuous assignment & conditional operators. Perform behavioral simulation to check the functionality of the designed circuit.
5. Design a 4 bit adder as a structural model by instantiating the full adder module 4 times. Simulate and test the code.