Building Combinatorial Circuit Using Data Flow Modeling Lab

Overview:
In this lab you will learn how to model a combinatorial circuit using Data-flow modeling style of Verilog HDL.

Outcome:
You will understand how to use Verilog logical operators in data-flow modeling style constructs. You will use ISE simulator to simulate the design. You will add user constraint file (ucf) to assign pins so the design can be targeted to National Instruments (NI) Digital Electronics FPGA Board. You will implement the design and create a bitstream file using ISE's implementation tools. Once bitstream is created, you will download using ISE's iMPACT program and verify the design functionality.

Background:
The Dataflow modeling style in Verilog uses continuous assignment statements. It is used to model a combinatorial circuits or expressions. In the continuous assignment statement the destination must of type net (not reg). You can have delay parameter value if you want to model the delay that net may be experiencing. The syntax for the assignment statement is

```
assign [delay] destination_net = source expression;
```

As an example,

```
assign #5 my_or2 = ain | bin;
```

In the above example, my_or2 is the destination_net and ain | bin is the source expression in which "|" is a logic OR operator. The source expression will be evaluated whenever its operand (ain or bin) value changes, and the result be assigned to the destination_net after 5 time units. If the delay parameter is not assigned then the result will be updated instantly. Note that the delay parameter value is useful for simulation purpose only. It is ignored during synthesis.

In a model, one can have as many dataflow statements as necessary. They all execute concurrently, thus the order of their appearance in the model does not matter. A statement is evaluated whenever its one or more of the operands change. The source expression uses logical operators. The language defines the following operators:

Operator	Precedence
+, -, !, ~ (unary)	Highest
*, / %	
+, - (binary)	
<<. >>	
<, < =, >, >=	
=, ==. !=	
===, !==	
&, ~&	
^, ^~	
|, ~|	
&&	
||	
? :	Lowest

[Verilog Quick Reference]

References:

1. National Instruments' Digital Electronics FPGA Board user manual
2. Verilog HDL books

   Stephen Brown, Zvonko G. Vranesic, "Fundamentals of Digital Logic with Verilog Design", 2002
   Zainalabedin Navabi, "Verilog Digital Systems Design: RT Level Synthesis, Testbench, and Verification", 2005
   Samir Paltinkar, "Verilog HDL: A Guide to Digital Design and Synthesis", 2003
   Joseph Cavanagh, "Verilog HDL: Digital Design and Modeling", 2007
   Michael D. Ciletti, "Modeling, Synthesis, and Rapid Prototyping with Verilog HDL", 2003
   Douglas J. Smith, "HDL Chip Design: A Practical Guide for Designing, Synthesizing and Simulating ASICs and FPGAs using VHDL or Verilog", 1996
3. On-line references:

   Verilog HDL Reference Card: http://www.stanford.edu/class/ee183/handouts_win2003/VerilogQuickRef.pdf

Problem Statement:

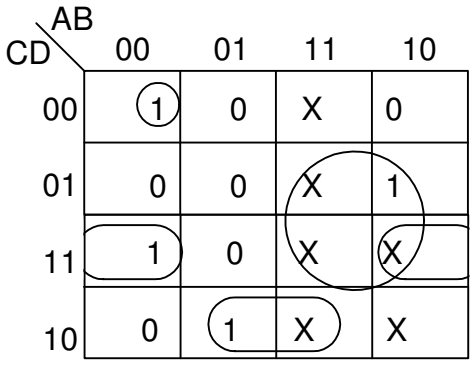
Design a combinatorial circuit that looks at its BCD input and output true when the input combination is divisible by 3 (0 is considered divisible by 3).

Implementation:

The circuit to be designed consists of four inputs and one output. Typically such circuit can be implemented using a combinational network. A truth table is created and then some Boolean minimization technique (e.g. K-Map) may be used to reduce the number of logic gates used. The truth table for the design at hand is shown next along with K-Map and minimized Boolean expression.

Inputs A B C D	Output F
0 0 0 0	1
0 0 0 1	0
0 0 1 0	0
0 0 1 1	1
0 1 0 0	0
0 1 0 1	0
0 1 1 0	1
0 1 1 1	0
1 0 0 0	0
1 0 0 1	1
1 0 1 0	X
1 0 1 1	X
1 1 0 0	X
1 1 0 1	X
1 1 1 0	X
1 1 1 1	X

Since the input is BCD, input combinations of 1010 to 1111 are not possible and hence they will have output is don't care (X). The K-Map will look like:



The Boolean expression will be

$$F = \overline{A}\,\overline{B}\,\overline{C}\,\overline{D} + A\,D + B\,C\,\overline{D} + \overline{B}\,C\,D$$

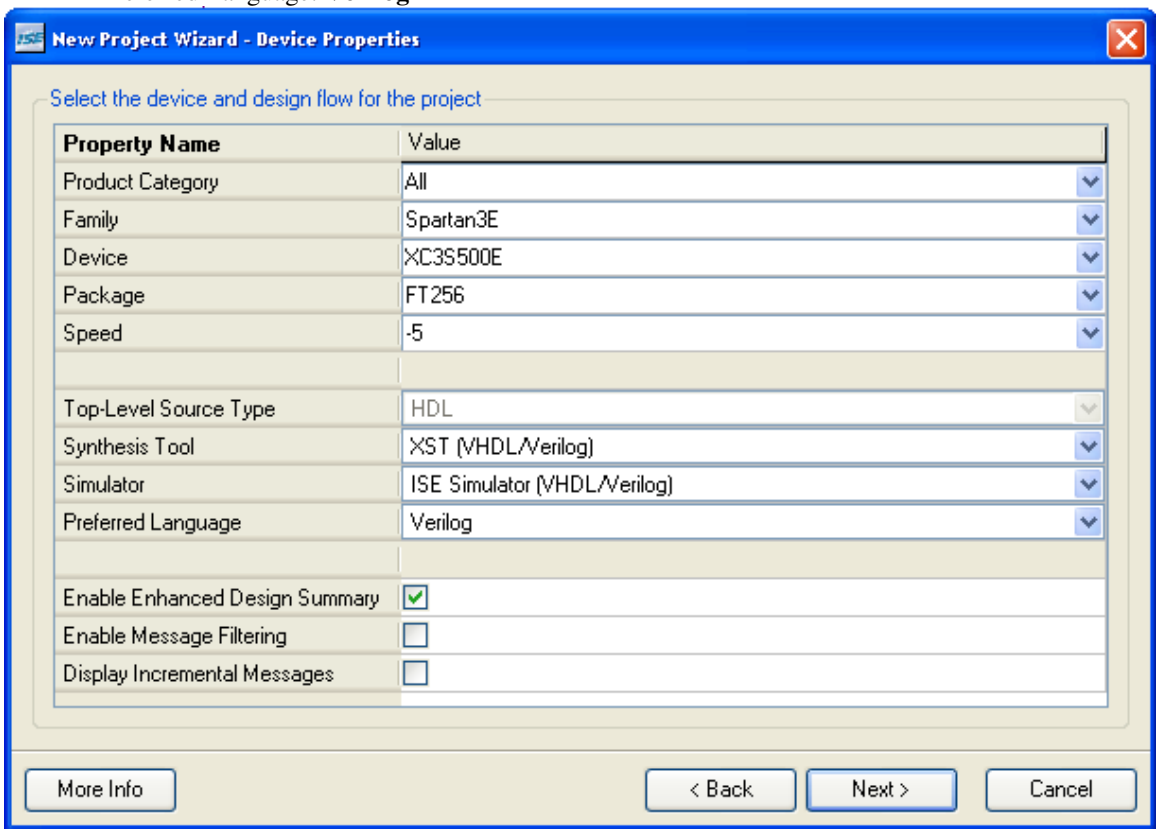
Procedure:

Extract resources.zip file in c:\NI\Verilog_Labs folder

1. Create a ISE project

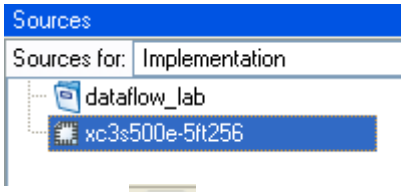
- Launch ISE: Select **Start → Programs → Xilinx ISE Design Suite 10.1 → ISE → Project Navigator**
- In the Project Navigator, select **File → New Project.** The New Project Wizard opens
- For Project Location, use the "**...**" button to browse to C:\NI\Verilog_labs, and then click **OK**
- For Project Name, type ***dataflow_lab***
- Click **Next**
- Select the following options and click **Next**

  - ✓ Device Family: **Spartan3E**
  - ✓ Device: **xc3s500E**
  - ✓ Package: **ft256**
  - ✓ Speed Grade: **–5**
  - ✓ Synthesis Tool: **XST (VHDL/Verilog)**
  - ✓ Simulator: **ISE Simulator (VHDL/Verilog)**
  - ✓ Preferred Language: **Verilog**



- The **Create New Source** dialog will appear. Click **Next**
- A *Add Existing Sources* form will be displayed. Click **Next** as we do not want to add

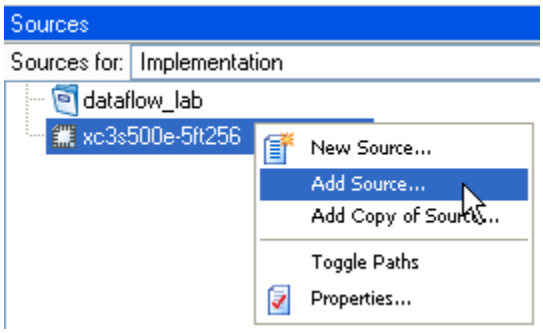
  Click **Finish**. An project will be created.



- Click  or **File** → **New** to create a *blank text* file.
- Enter the following model for the divide by 3 function using data-flow modeling style

```
1   module dataflow (
2      input a,
3      input b,
4      input c,
5      input d,
6      output dataflow_out
7      );
8
9   wire a_n, b_n, c_n, d_n, n1, n2, n3;
10
11  assign #2 a_n = ~a;
12  assign #2 b_n = ~b;
13  assign #2 c_n = ~c;
14  assign #2 d_n = ~d;
15  assign #2 n1 = a & d;
16  assign #2 n2 = b_n & c & d;
17  assign #2 n3 = a_n & b_n & c_n & d_n;
18  assign #2 n4 = b & c & d_n;
19  assign #2 dataflow_out = n1 | n2 | n3 | n4;
20
21  endmodule
22
```

- Save the file as **dataflow.v** and notice the text changes into context driven (Verilog language sensitive) format
- Close the file
- Note that even though the file has been created, it is not automatically added to the project as it was created as blank text file
- To add the files, select the chip in the *Sources* window and right-click and then select **Add Source…**



- Select the verilog model file (dataflow.v) and click **Open**
- Click **OK** to add the file

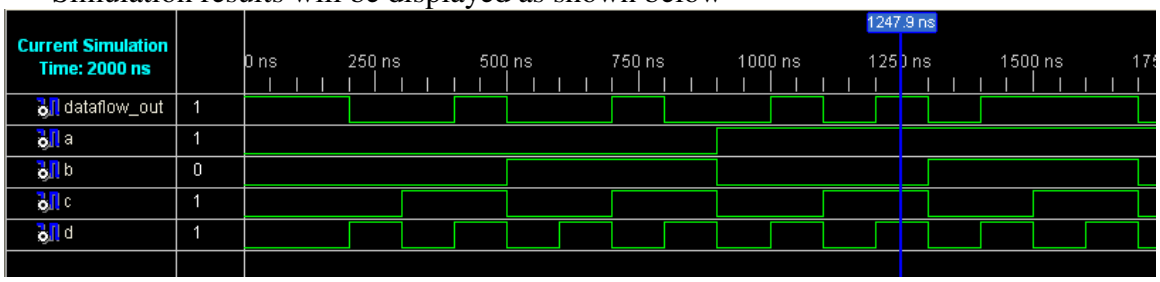
- Create a new blank file and enter the following location constraints

```
1  NET "d"  LOC = "J11" | IOSTANDARD = LVCMOS33 ; #switch[0]
2  NET "c"  LOC = "J12" | IOSTANDARD = LVCMOS33 ; #switch[1]
3  NET "b"  LOC = "H16" | IOSTANDARD = LVCMOS33 ; #switch[2]
4  NET "a"  LOC = "H13" | IOSTANDARD = LVCMOS33 ; #switch[3]
5  NET "dataflow_out"  LOC = "C11" | IOSTANDARD = LVCMOS33 ; #LED[0]
6
```

- Save and close the file, giving **dataflow.ucf** as the filename and *UCF(*.ucf)* as the *Save as type:*
- Add the ucf file to the project

2. *Simulate the design using ISIM*

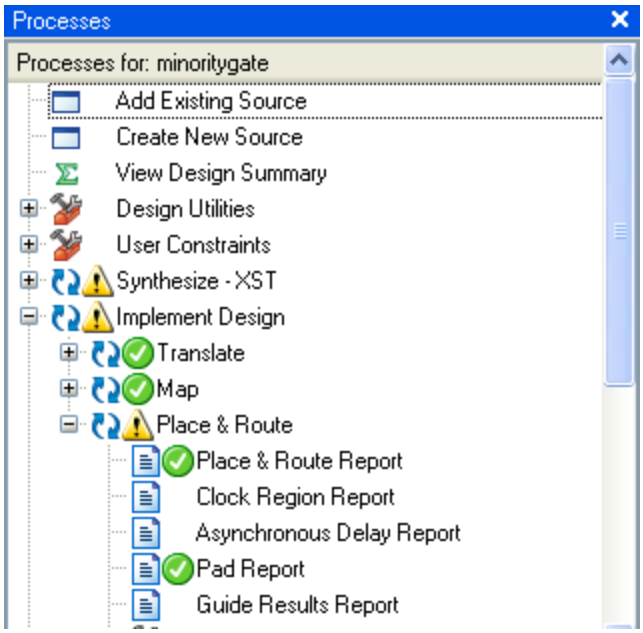
- Right-click on the **dataflow** entry in *Sources* window, right-click and select **Add Copy of Source…**
- Browse to C:\NI\Verilog_labs\resouces\dataflow_lab and select **dataflow_tb.v**. Notice that the Sources for window changes to Behavioral Simulation from Implementation
- Select **dataflow_tb** in the *Sources* window, expand the **Xilinx ISE Simulator** process in *Processes* window, and double-click **Simulator Behavioral Model**
- The model will be compiled and the simulator will be run
- Simulation results will be displayed as shown below



- Analyze and understand the simulation results. Observe that the output becomes TRUE when input combination is 0, 3, 6, and 9. It also becomes true when the input is > 9 (invalid input combinations which were don't care during creating truth table
- Close the simulator

3. *Implement the design*

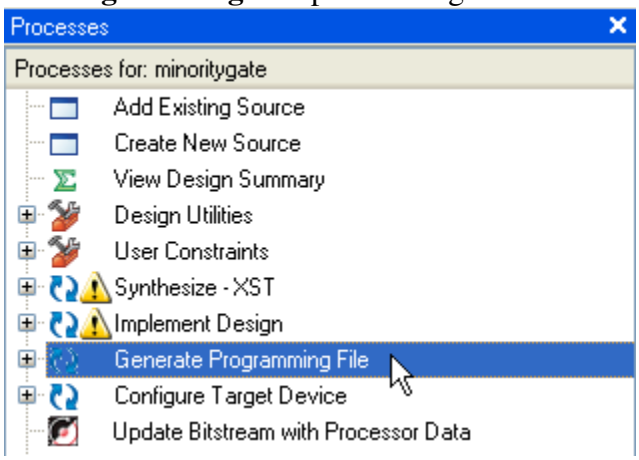
- Select **implementation** in *Sources for* window
- Select **dataflow** module in *Sources* window and double-click on **Implement Design** process in *Processes* window. This will go through Synthesis, and Implementation stages
- When the implementation is completed, expand Implement Design process to view the Place & Route report



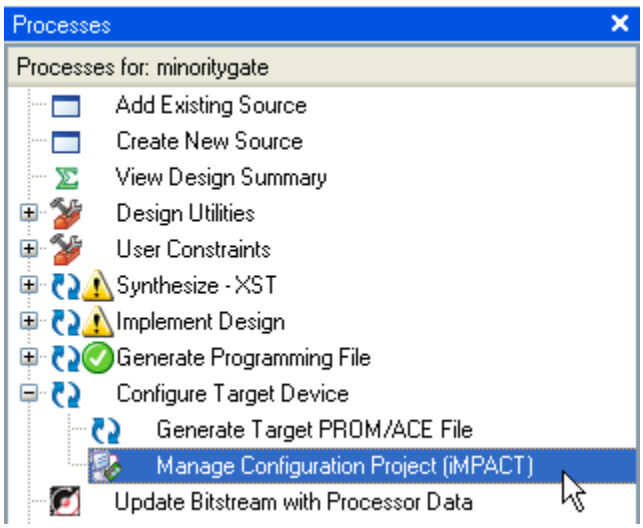
- Double-click on the Place & Route report to view the report. Look at the resource utilization and note that **1** slice is being used as the function has one output and 4 inputs. This shows that the number of slices (LUT) being used is a function of number of input variables and not the number of product and sum terms
- You can see similar information by clicking on Design Summary tab and looking at the various information

4. Verify the design in hardware

- Select **dataflow** in *Sources* window and double-click on **Generate Programming File** process to generate the bit file for the design



- Expand **Configure Target Device** process and double-click on **Manage Configuration Project (iMPACT)** process



- Connect the board with the USB-JTAG cable
- Power ON the board
- Click Finish to use the JTAG chain
- Select dataflow.*bit* file to be assigned to xc3s500e device and click **Open**
- Click **Bypass** button for *xcf04s* and then **OK** to use FPGA device programming
- Right-click on the FPGA and select **Program**
- This will program the FPGA and DONE light will lit on the board
- Once programmed successfully, verify the functionality by using SW0 thru SW3 and monitoring LD0 output
  - SW0 thru SW3 are BCD input, with SW0 being the least significant
  - To test the design, first set all switches to OFF position and observe that the LD0 is ON
  - Now change settings of input and verify that the design is behaving correctly
- Once confirmed the functionality, power down the board and close ISE saving project changes

Conclusion:

In this lab exercise you learned how to model a combinatorial circuit using dataflow style of Verilog HDL. You also saw what a continuous assignment statement is and its syntax, You also learned the operators it supports. You were able to simulate the design and then verify the complete design in hardware board.