Nikhil Gupta

FPGA Challenge

Takneek 2012

RECAP

FPGA

- Field Programmable Gate Array
  - Matrix of logic gates
  - Can be configured in any way by the user
  - Codes for FPGA are executed in parallel
- Configured using a Hardware Description Language (HDL)
- Verilog – A Hardware Description Language

A Practical Example

- MCU
  - ISR (Interrupt Service Routine)
  - One needs to take care of its length (Remember! Use of flags)

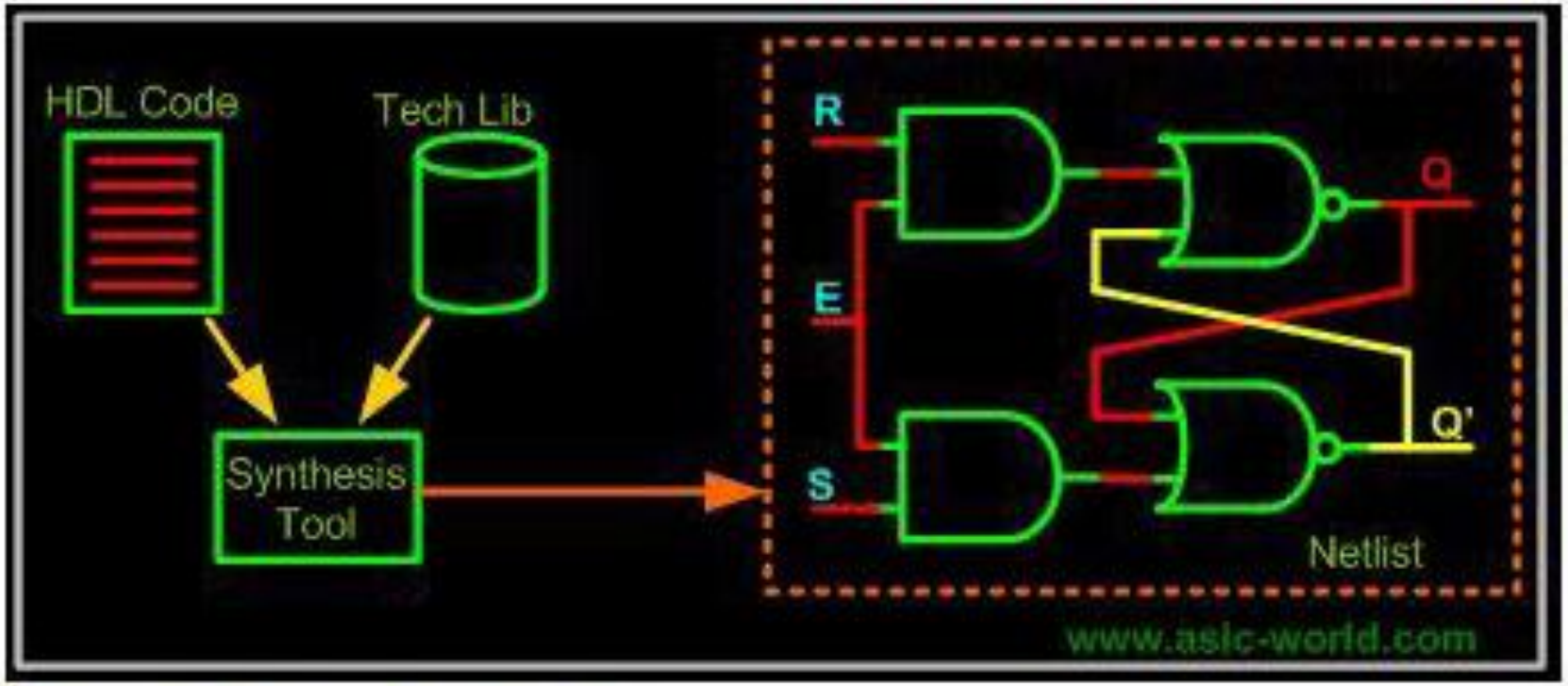
- FPGA
  - Little clouds of control logic (called blocks/modules) keep running simultaneously
  - No chance of missing an interrupt

Verilog

- Not a programming language despite the syntax being similar to C
- Describes an electronic model (for our purposes, the circuit is digital in nature)
- Synthesized (analogous to compiled for C) to give the circuit logic diagram

Synthesis of Verilog

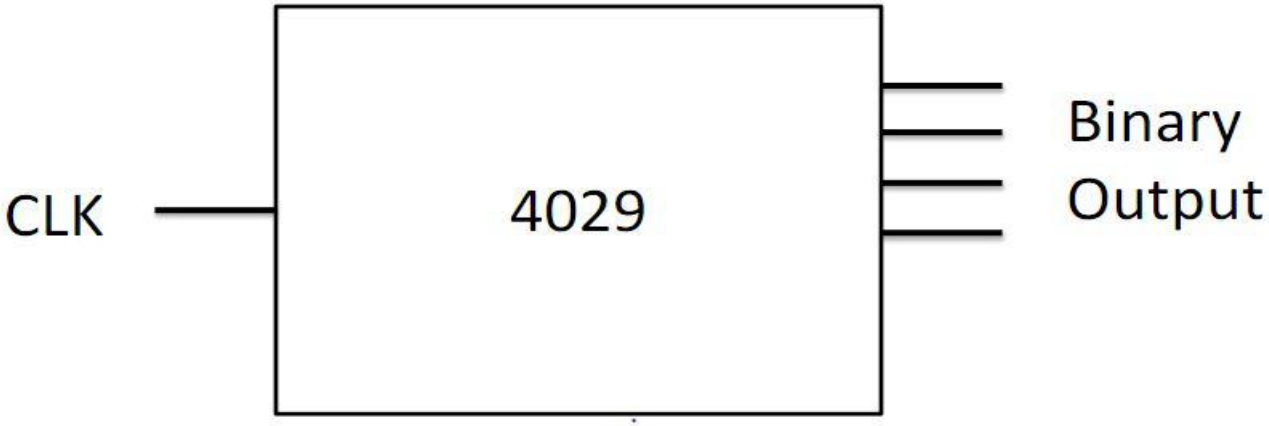
Synthesis Flow



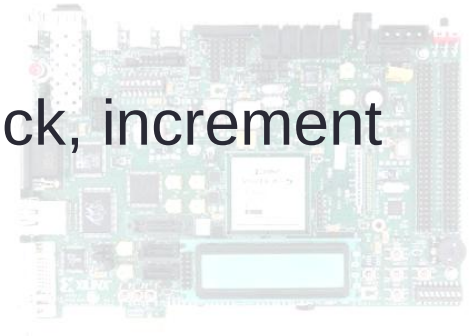
Coding in Verilog

- Consists of various building blocks called **Modules**
- Communication between a module and its environment is achieved by using **Ports**
- **Ports** are of three types: input, output, inout

Basic Logic Diagram for a 4029 Counter



- Name: 4029
- Input Ports: One
- Output Ports: Four
- Internal Logic: At every rising edge of the clock, increment the output by one



Module

- A “Black Box” in Verilog with inputs, outputs and internal logic working.
- So, a module can be used to implement a counter.
- A module is defined as

  module <specific type>(<port list>);

Defining 4029 module

- Way 1:
  - module 4029(clk,out,reset,enable); //Assuming two more input pins, reset and //enable with their corresponding functioning

OR

- Way 2:
  - module 4029(clk, a, b, c, d, reset, enable);

- Size of Port out?
- Input and Output Ports in each of the above?

- Every module ends with the statement endmodule

Declaring Ports

- Way 1:

```
input clk;
input reset;
input enable;
output a,b,c,d;
```

- Way 2:

```
input clk;
input reset;
input enable;
output [3:0] out;
```

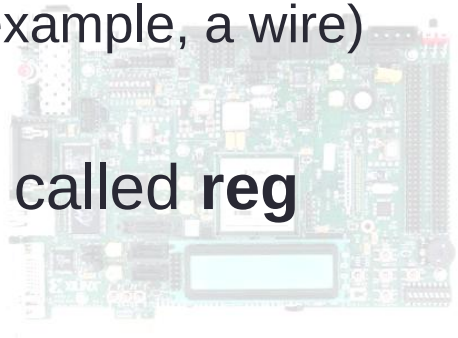
Drivers in Verilog

- We need drivers for this module in order to interact with the ports and describe its logical working.

- Driver is a way of defining something which can drive a load

- Two types of drivers:
  - Can store a value (for example, flip-flop)
  - Cannot store a value, but connects two points (for example, a wire)

- In Verilog, a driver which can store a value is called **reg** and the one which cannot is called **wire.**



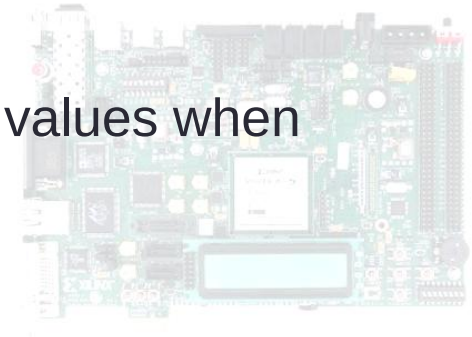
Drivers for 4029 module

- Ports defined as wires?
  - clk
  - reset
  - enable

  We do not need to stores the values of these ports in our logical block.

- Ports defined as reg?
  - a,b,c,d
  - out

  We need to store them so that we could modify their values when required.



Defining Drivers for 4029 module

- Way 1:

```
wire clk;
wire reset;
wire enable;
reg a,b.c,d;
```

- Way 2:

```
wire clk;
wire reset;
wire enable;
reg [3:0] out;
```

Operators and Conditional Statements

- All the arithmetic as well as logical operators in Verilog are similar to C, except ++ and – which are not available in Verilog.

- Conditional statements are also similar to C with following modifications:
  - { is replaced by begin.
  - } is replaced by end.

initial block

- It is executed only in the beginning of the simulation.
- One can have many initial blocks. In this case, all of them will be executed in the beginning of the simulation.
- Syntax

```
initial begin
//Code
end
```

always block

- Syntax

  always @(condition)

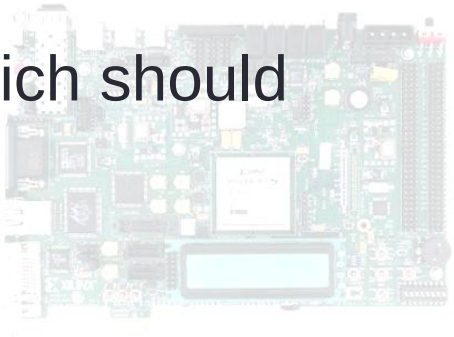
  begin

  //Code

  end

- Blocks starting with keyword **always** run simultaneously.

- @ symbol is used to specify the condition which should be satisfied for the execution of this block.



Usage of always block

- always

  The code in this block will keep on executing.

- always @(a)

  The code in this block will be executed every time the value of a changes.

- always @(posedge clk)

  This block is executed at every positive edge of clk.

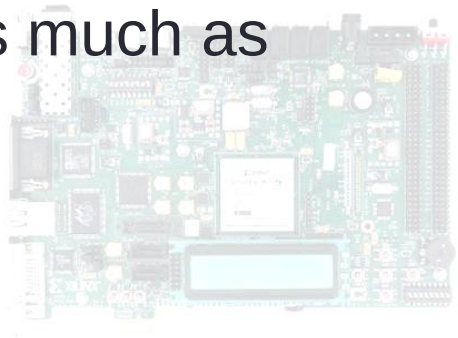
Blocking and Non-blocking assignments

- A blocking statement (consists of an = sign) must be executed before the execution of statements following it in a manner.

```
1  a = b;
2  out_d = 0;
3  {carry,sum) = in + sum_in;
```

- Non-blocking statements (consist of <= sign) within a block are executed simultaneously.

- We will try to use non-blocking statements as much as possible.



```
module 4029 ( input wire clk,
              input wire reset,
              input wire enable,
              output [3:0] reg out);   //You can declare direction as well as data type
                                        //in the module definition.
initial
begin
out <= 4'b0000;
end

always @(posedge clk)
begin
        if (reset == 0 && enable == 0)
        begin
                out <= out +1;
        end
end
```

```
always @(reset or enable)
begin
        if (reset == 1'b1)
        begin
                out <= 0;
        end
        if (enable == 1'b1)
        begin
                //Code if you want it to do something.
        end
end

endmodule
```

Assign statement

- Used to control the output wires which cannot be done using a blocking statement.
- It is executed continuously.
- Syntax:
  - module Repeater(input wire A, output wire B);
  - assign B = A;
  - endmodule;

Modular Circuits

- Various modules are interconnected to make a larger circuit (or module).
- Each sub-module has a separate Verilog file.
- A sub-module may have another sub-module in its circuit.
- One needs to indicate the top level module before synthesis.

MODULAR CIRCUITS

Example

- Sub-module 1
  - module Sub1 (input wire a1, output reg [7:0] b1);
- Sub-module 2
  - module Sub2 (input wire [7:0] a2, output reg [7:0] b2);
- Top Module
  - Module Top (input wire a, output reg [7:0] b);

Instantiation

- Used to interconnect various modules.
- In the above example, we need to instantiate the two sub-modules in the top level module.
- This is done as follows:
  - wire [7:0] c;
  - Sub1 Encoder (.a1(a), .b1(c));
  - Sub2 Decoder (.a2(c), .b2(b));

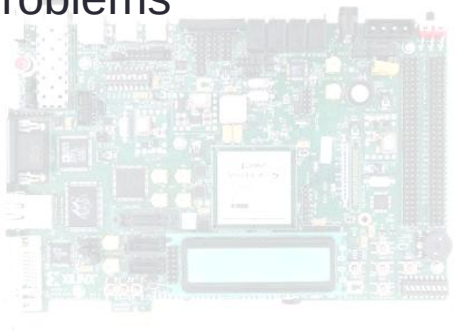
PROBLEM STATEMENT

FPGA Challenge

- On the Spot Event

- Prelims
  - Half an hour quiz
  - Based on fundamentals of Verilog

- Finals
  - Level 1:
    - Implement Verilog modules to solve the given design problems
  - Level 2:
    - Simulation of these Verilog modules



JUDGING CRITERIA

- Prelims:
  - 10 top scoring teams will move to the finals.

- Finals:
  - Level 1: Judging shall be done on the basis of:
    - Number of problems solved
    - Number of modules used
    - Ease of understanding of the code
    - Extra features if any
  - Level 2:
    - Teams will be provided points for this level only when their simulations of the problems solved by them work perfectly.

Important Note

- Team strength should be 2.
- At least one member of each team should belong to Y11 batch.
- The Team could comprise of two Y11 Students but it could not be comprised of two non Y11 students.

Contacts

Anurag Dwivedi
156 / Hall 2
anuragdw@iitk.ac.in
8960482723

Nikhil Gupta
138 / Hall 3
nikgupta@iitk.ac.in
9005671866

Rudra Pratap Suman
F107 / Hall 5
rpsuman@iitk.ac.in
9450003098

Takneek Website :

http://students.iitk.ac.in/takneek/ > events > Electronics > FPGA

FB Group : www.facebook.com/eclub.iitk