Introduction to FPGA and Verilog

Embedded Week '14

by

Anurag Dwivedi

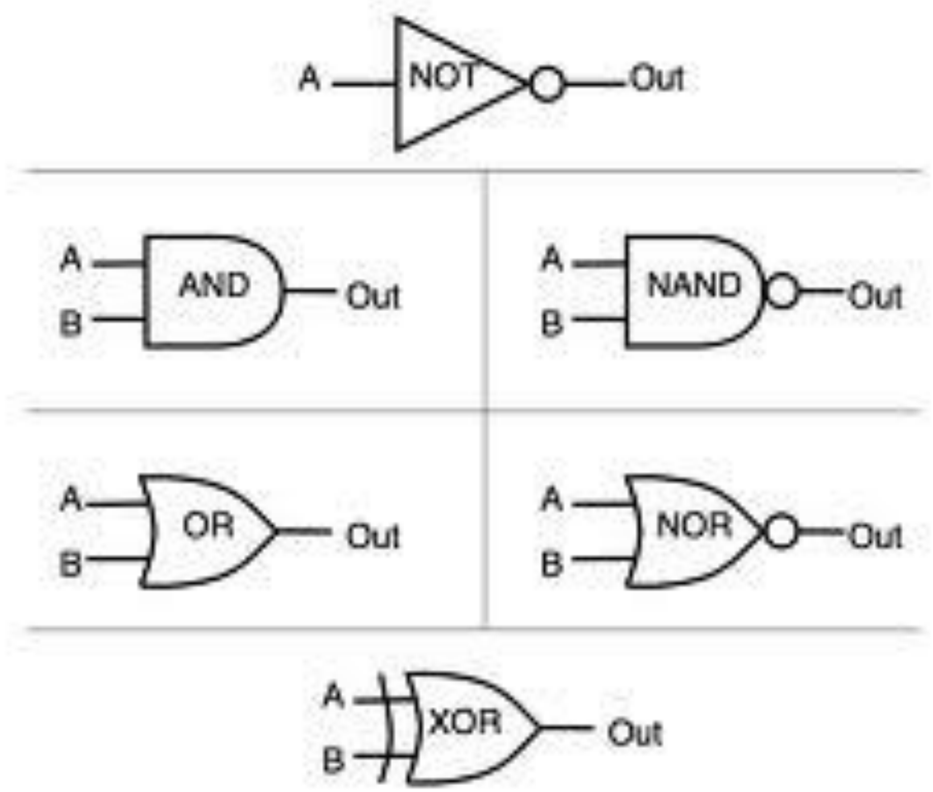
Basics

- Acquaintance with digital circuits.
- Logic gates, flip flops, counters, multiplexers.
- A micro-controllers consists of various smaller units such as counters, timers, memory units etc.
- Familiarity with any programming language.

Digital Design : Bottom Up Approach

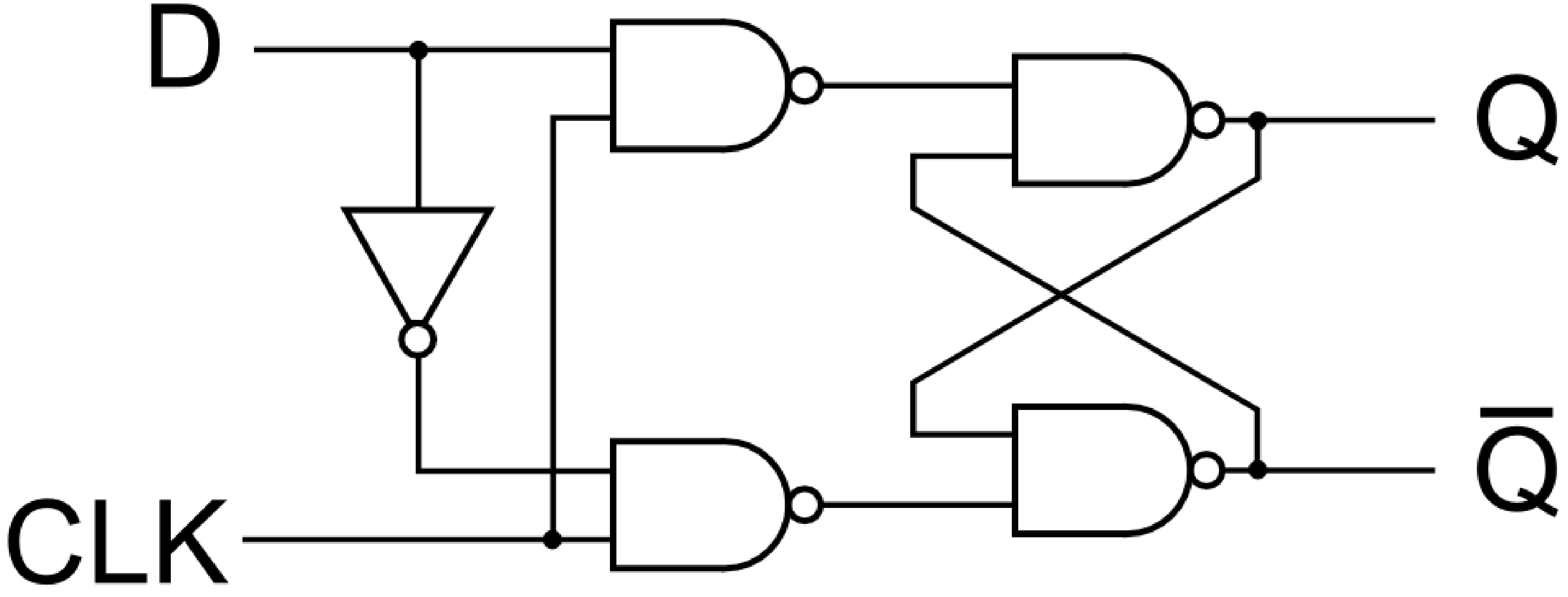
- What are the basic building blocks of all digital devices.
- Central Processing Unit ??
- Counters ?? Timers ??
- Flip-flops ??
- Gates !!

Digital Design : Gates



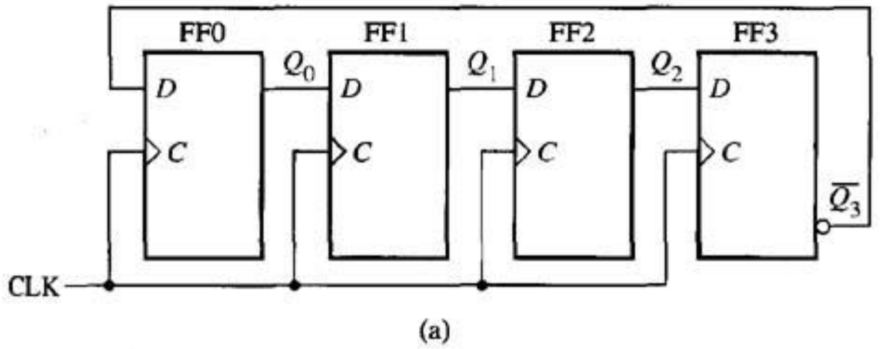
Most fundamental unit of any digital device

Digital Design : Flip Flops



Gates can be combined together to construct flip-flops

Digital Design : Counter



(a)

Clock	Q_0	Q_1	Q_2	Q_3
0	0	0	0	0
1	1	0	0	0
2	1	1	0	0
3	1	1	1	0
4	1	1	1	1
5	0	1	1	1
6	0	0	1	1
7	0	0	0	1

(b)

Various flip-flops can be connected to each other to form counters

Digital Design : Processors

Finally a processor can be designed from various smaller building blocks

From Designing To Testing

- Digital designing >> Testing and Validation
- Fabrication of ICs and testing on it ??
- Not an efficient solution
  - Testing involves many iterations
  - Wastage of money
  - Wastage of time
- FPGA is the solution

FPGA

- Field Programmable Gate Array

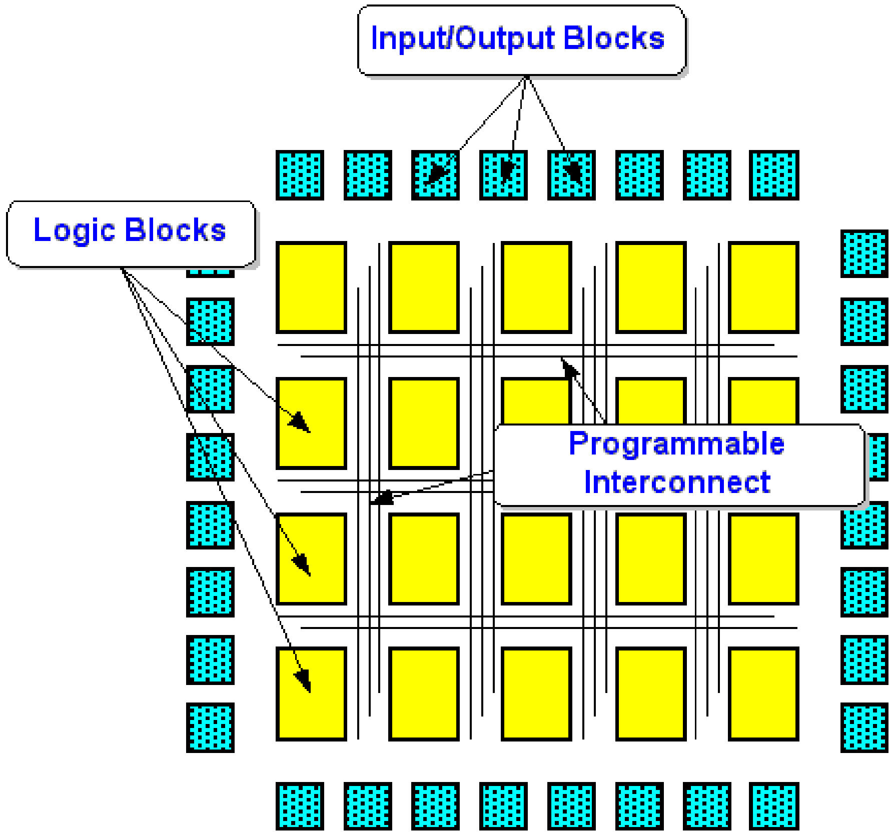
- A fully configurable IC
  - Can be made to work as a XOR gate, a Counter or even bigger- an entire Processor!

- FPGAs contain **programmable logic** components called logic blocks.

- Hierarchy of reconfigurable interconnects that allow the blocks to be **wired together.**

- FPGA has reprogrammable hardware as opposed to reprogrammable software in micro-controllers (when you feed a different hex file to your Atmega)

FPGA Architecture



Both the logic blocks and the interconnects can be programmed

Applications of FPGA

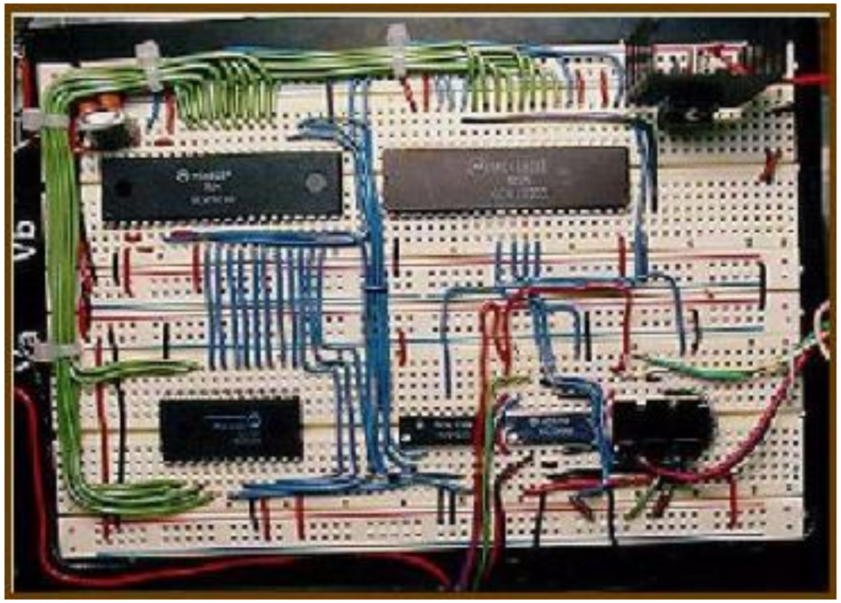
- Various other uses apart from being used as a testing device

- Due to reconfigurable nature, immense applications in
  - Space
    - Mars Rover uses Xilinx's FPGAs
  - Defense
  - Medical Science

- FPGA can do parallel processing also.

How to design the circuit?

- Circuit elements of the order of 10
  - Can be designed directly on paper or breadboard



How to design the circuit?

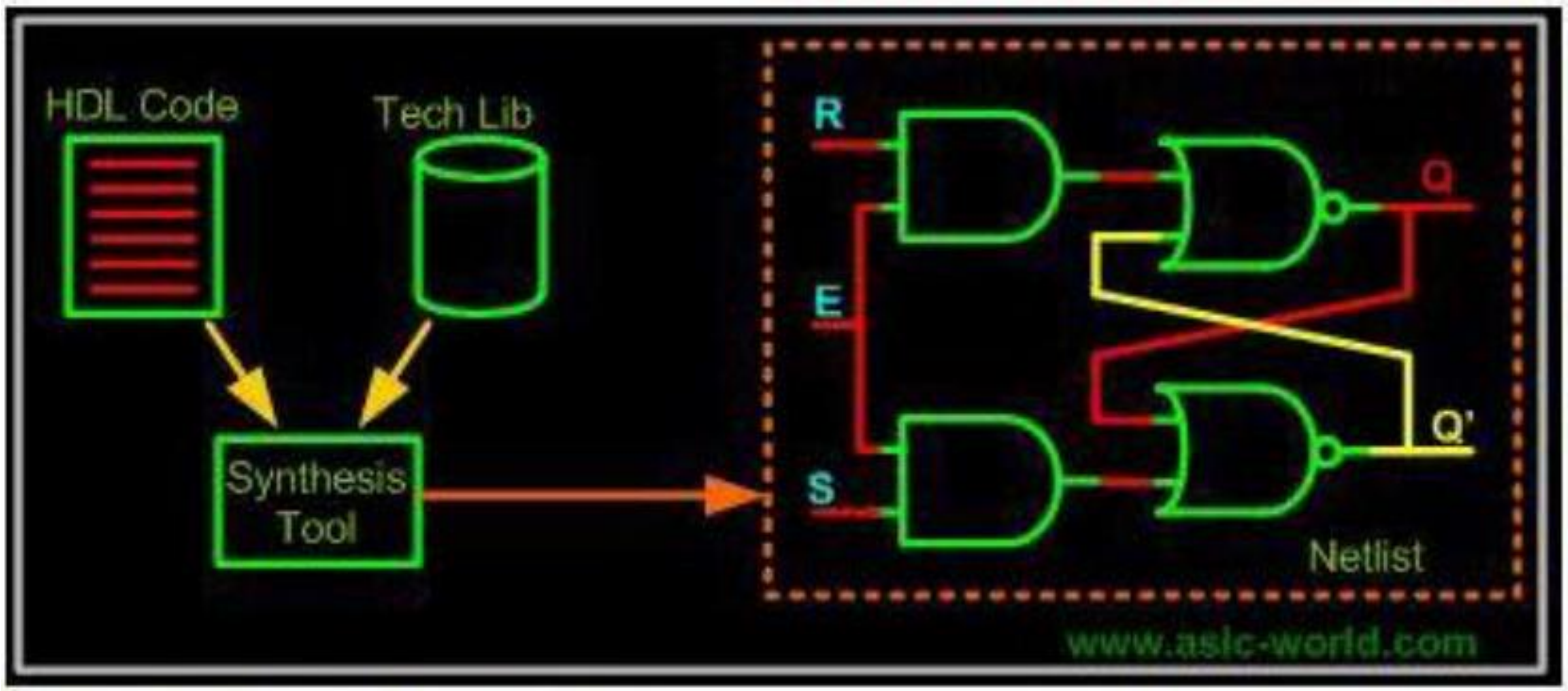
- What if number of circuits elements increases exponentially ??
  - Designing on paper or breadboard – not an efficient solution

- Need to describe the circuit in another syntax
  - Hardware Descriptive Languages (HDLs) developed for this purpose
  - Describes the digital circuits in terms of its **structure** and **functionality**.

Verilog

- Verilog is A Hardware Descriptive Language used to describe a circuit.
- Syntax is similar to C, but is not a programming language
- Synthesized (analogous to compiled in C) to give the circuit logic diagram
- VHDL is another HDL commonly used

Synthesis of Verilog



C code is compiled to give an executable, Verilog Code is synthesized to give a hardware

FPGA and Verilog

- Description of the circuit is written in Verilog

- It is then synthesized

- Finally it is implemented on FPGA

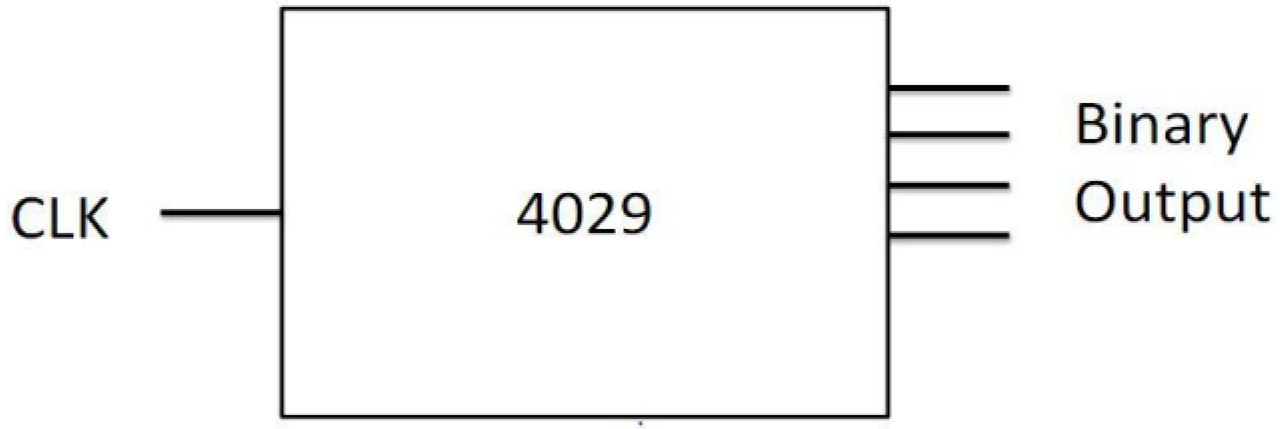
CODING IN VERILOG

Modules

- Consists of various building blocks called Modules
- Each module is a separate entity in itself. Eg. counter, OR gate, multiplexer etc.
- Communication between a module and its environment is achieved by using Ports
- Ports are of three types: input, output, inout

4029 counter

- One Input port for CLK
- Four binary output ports
- At every rising edge of clock, increment output by 1



Declaring Module

- Way 1:

```
module 4029(clk, a, b, c, d, reset, enable);
//Assuming two more input pins, reset and
//enable with their corresponding functioning
```

- Way 2:

```
module 4029(clk, out, reset, enable);
```

What is the difference in the two?

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;
```

Types of Ports

- We need drivers for this module in order to interact with other modules

- A driver is a data type which can drive a load.

- Basically, in a physical circuit, a driver would be anything that electrons can move through

- 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.

Complete definition for module

```
module 4029( clk, out, reset, enable);
    input wire clk;
    input wire reset;
    input wire enable;
    output reg [3:0] out;
 endmodule
```

Wire vs Reg

- reg can store a value, wire simply connects

- Most of the times, inputs are wire and outputs are reg

- Output of flip flop – wire or reg ?
- Output of XOR gate – wire or reg ?
- Output of multiplexer – wire or reg ?

What now ?

- We have seen how to define the outer structure of the modules we will use.
- Time to define the internal structure and functioning?

Combinatorial Circuits

- Combinational circuits are acyclic interconnections of gates.
  - And, Or, Not, Xor, Nand, Nor ……
  - Multiplexers

- **OUTPUT DEPENDS ON THE PRESENT INPUT ONLY.**

Assign Statements

- Continuous assignment statement.

- Used for modeling only combinational logic.

```
module BusInverter(  input wire A,  output
   wire B );
   assign B = ~A;
endmodule
```

- Basically B is shorted to ~A.
- **RHS should have variable of wire type.**

Example : 1 bit half adder

```
module half_adder(
   input wire a,
   input wire b,
   output wire sum,
   output wire carry   );

assign sum = a & ~b | ~a
   & b ;

assign carry = a & b ;

endmodule
```

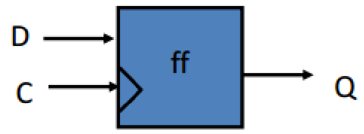
Gate Level Description

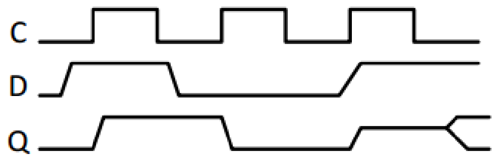
```
module full_adder(
   input wire a,
   input wire b,
   output wire sum,
   output wire carry );

assign { carry, sum } =
   a+b;

endmodule
```

Behavioural Description

Sequential Circuit

- Circuits containing state elements are called sequential circuits
- **OUTPUT DEPENDS ON THE PRESENT INPUT AS WELL AS ON ITS PRESENT STATE.**
- The simplest synchronous state element: Edge triggered D Flip Flop





- How do you implement such an element in Verilog?

always @ block

- It is an abstraction provided in Verilog to mainly implement sequential circuits.

- Also used for combinational circuits.

```
always @(#sensitivity list#)
begin
    ……….  //No assign statements inside always@
end
```

- Execution of always block depends on the sensitivity list.

Sensitivity list

- Run continuously.
  always

- Run when any variable changes its value.
  always @(*) //for combinational ckts

- Run when the variables `a' or `b' change their value.
  always @(a, b)

- Run when a positive edge is detected on CLK.
  always @(posedge CLK) //for sequential ckt

Counter Example

```
module Counter(
  input wire CLK,
  output reg [3:0] OUT  );

always @(posedge CLK)
  OUT <= OUT + 1;

endmodule
```

What next ??

- Only the minimal Verilog syntax is introduced.
- A lot is there to learn.
- Modular circuits – Using a module inside another bigger module.
- Test Benches – Test your circuit by simulations before deploying on the FPGA Board.