

# **VLSI & FPGA**

Nahin Ul Sadad  
Lecturer  
CSE, RUET

**Question:** How is real-life processor designed and implemented step by step?

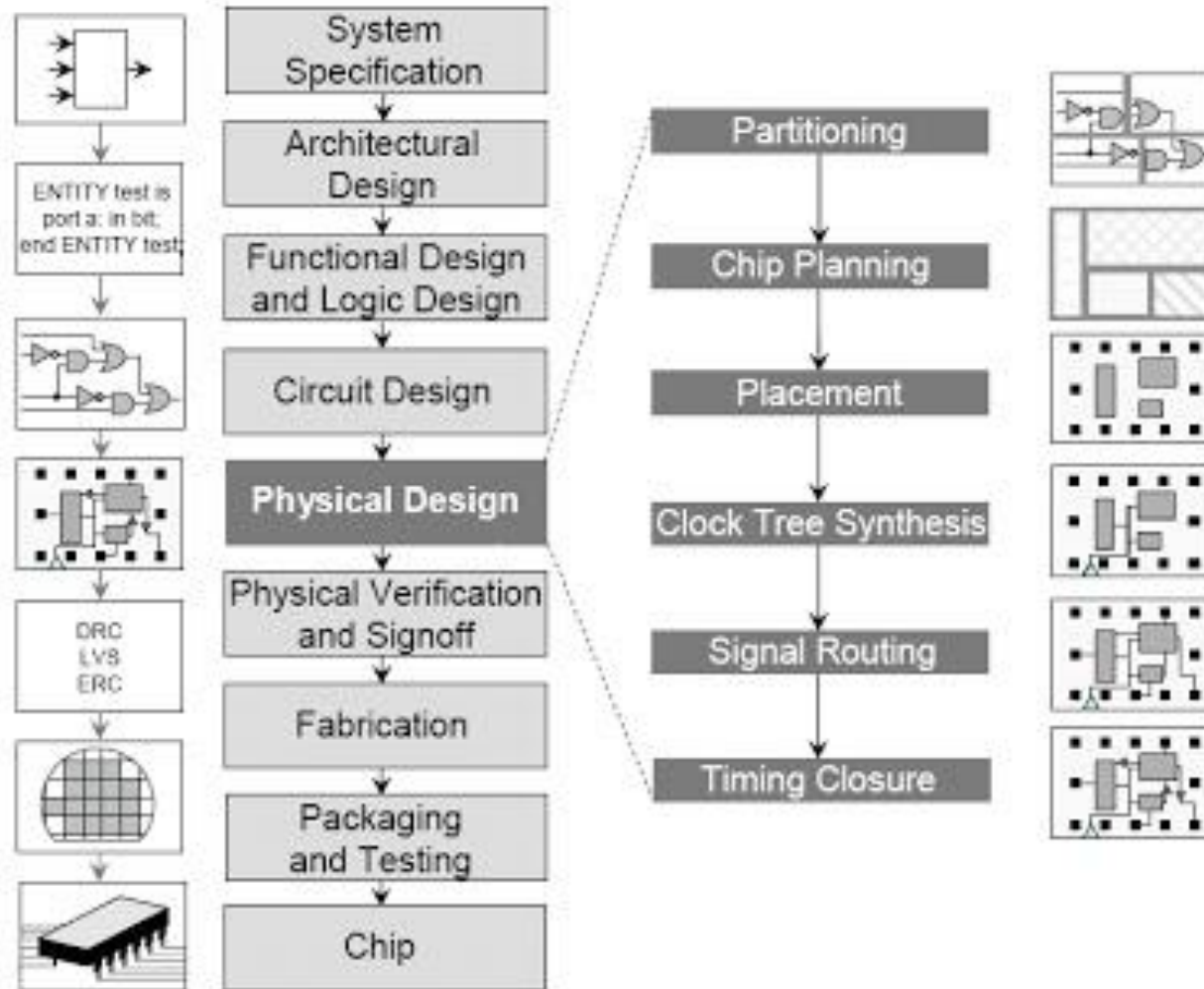
Or,

Explain design and implementation of real-life processor step by step.

Or,

Draw and explain VLSI design flow.

Ans:



1. **System specification:** Chip designers/Hardware engineers collectively define the overall goals and high-level requirements of the system.
2. **Architectural design:** A basic architecture must be determined to meet the system specifications. Architecture includes types of core (CPU, GPU, DSP, Cache etc.), pinout, power requirements etc.
3. **Functional and logic design:** Once the architecture is set, the functionality and connectivity of each module (such as a processor core) must be defined. During functional design, only the high-level behavior must be determined. That is, each module has a set of inputs, outputs, and timing behavior.

Logic design is performed at the register-transfer level (RTL) using a **Hardware description language (HDL)** by means of programs that define the functional and timing behavior of a chip. Two common HDLs are Verilog and VHDL. HDL modules must be thoroughly simulated and verified.

4. **Circuit Design:** Logic synthesis tools are used to automate the process of converting HDL into equivalent low-level circuit elements (transistor circuits).

5. **Physical design:** During physical design, all design components are **instantiated** with their geometric representations. It means all components are assigned spatial locations (placement) and have appropriate routing connections (routing).
6. **Physical verification:** After physical design is completed, the layout must be fully verified to ensure correct electrical and logical functionality.
7. **Fabrication:** The final layout is sent for manufacturing at a dedicated silicon foundry (fab). At the end of the manufacturing process, the ICs are **separated**, or **diced**, by **sawing** the **wafer** into smaller pieces.
8. **Packaging and Testing:** After dicing, functional chips are typically packaged. Package types include **DIPs, PGSSs, BGAs** etc. After packaging, the finished product may be tested to ensure that it meets design requirements such as function, timing and power dissipation.

**Question:** How does a Hardware Engineer design Digital Circuits in real life?

**Answer:** Hardware Engineer designs Digital Circuits through programming in real life!

Because it is neither feasible nor practical to design/draw a digital circuit like CPU/Digital IC/Chip with billions/millions of transistors.

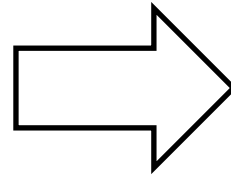
So, Hardware Engineer describes the behavior of digital circuit through programming which in turn generate equivalent Transistor circuits.

# Question: What is HDL? What is the importance of HDL?

## Answer:

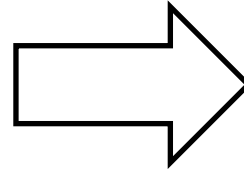
Hardware description language (HDL) is a specialized computer language used to describe the structure and behavior of digital logic circuits to create equivalent transistor circuit which is used to create an Integrated Circuit (IC). HDL can be simulated in software and can be physically verified in FPGA.

Hardware Description Language (HDL)  
(High Level Language)  
For example: Verilog, VHDL etc.

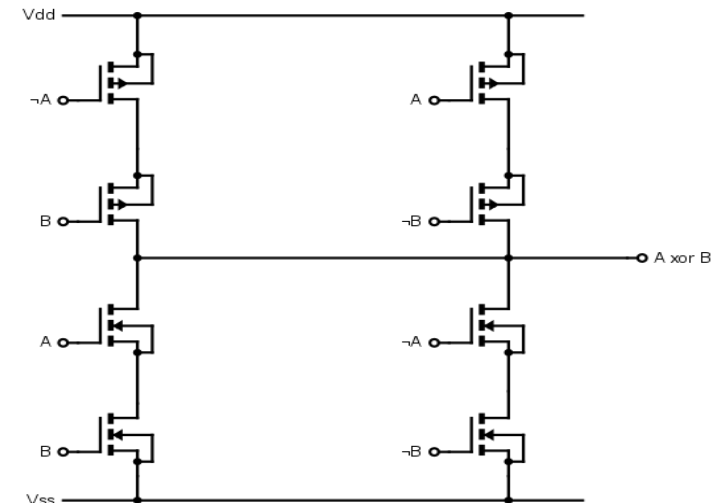


Generate Equivalent Transistor Circuit

```
module XOR(output Y, input A, B);  
  assign Y = A ^ B;  
endmodule
```



**Figure:** HDL Code of XOR to  
equivalent CMOS transistor circuit



**Question:** Is it possible to physically verify softcore of digital circuits (digital IC) written in HDL?

**Ans:** Yes, it is possible to physically simulate **softcore** of digital circuits/digital IC written in HDL using Field Programmable Gate Array (**FPGA**).



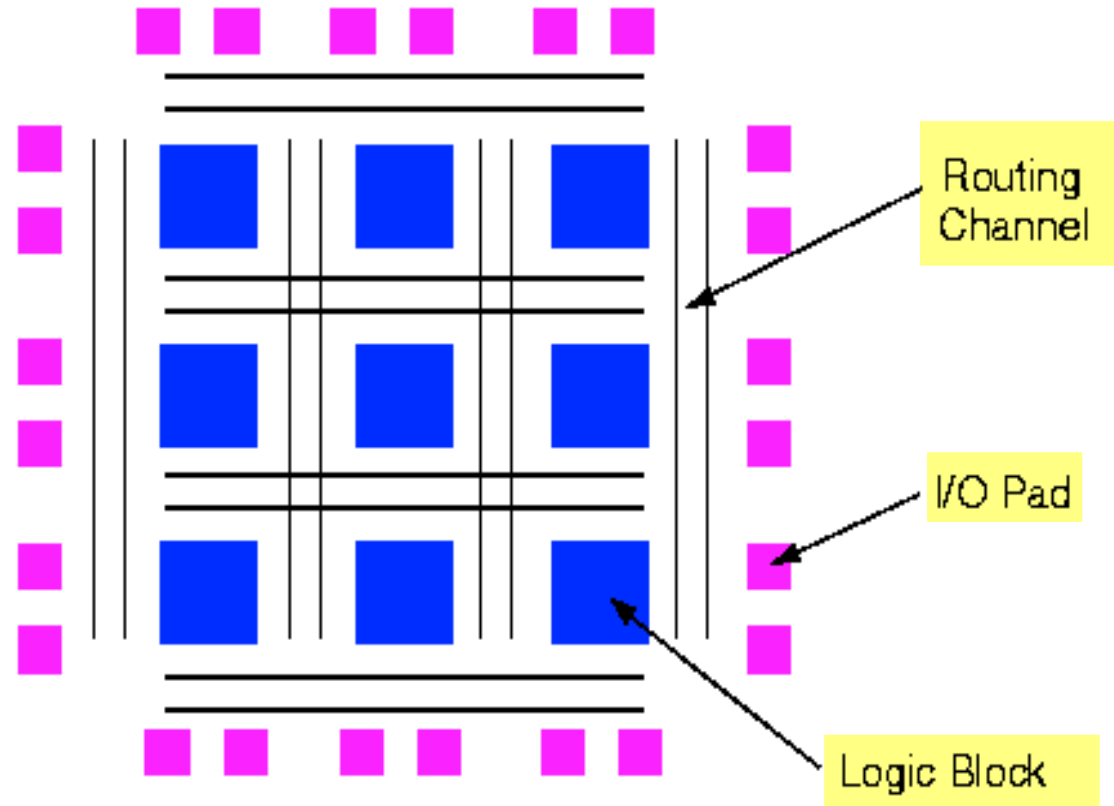
**Question:** What is Field Programmable Gate Array (FPGA)?

**Ans:** A Field Programmable Gate Array (FPGA) is an integrated circuit designed to be configured by a customer or a designer after manufacturing – hence the term "field-programmable".

The FPGA configuration is generally specified using a hardware description language (HDL).

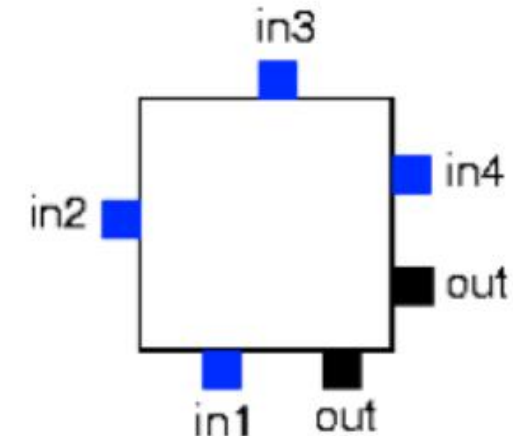
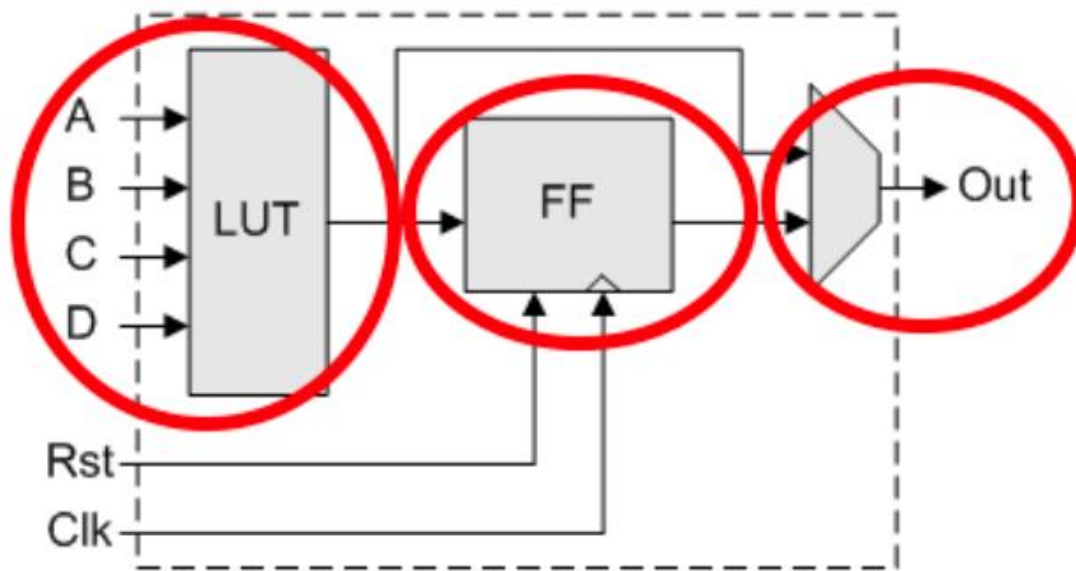
## Field-Programmable Gate Array (FPGA)

- ◆ First introduced by Xilinx in 1985
- ◆ Arrays of logic blocks (to implement logic functions)
- ◆ Lots of programmable wiring in routing channels
- ◆ Very flexible I/O interfacing logic core to outside world
- ◆ Two dominant FPGA makers:
  - Xilinx and Altera
- ◆ Other specialist makers e.g. Actel and Lattice Logic



## Configurable Logic Block (CLB)

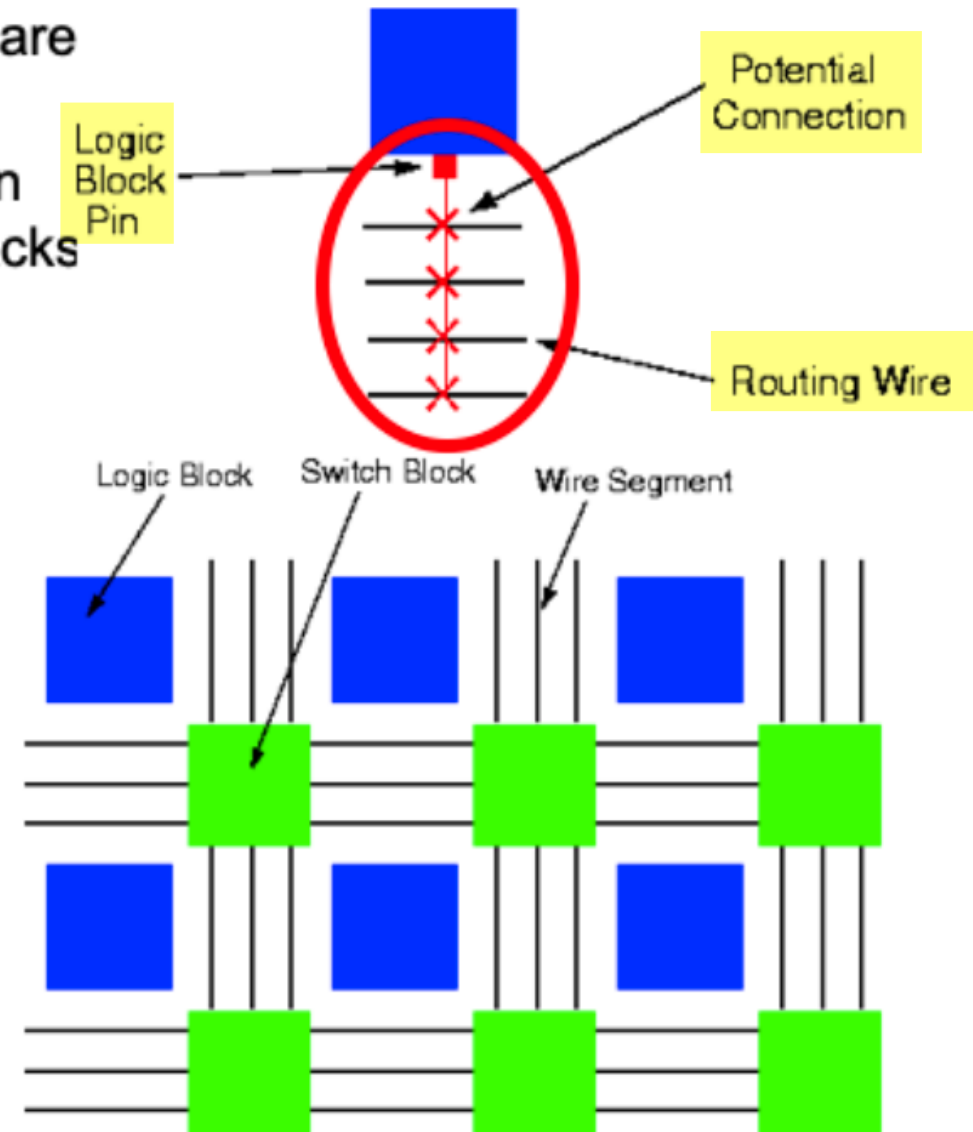
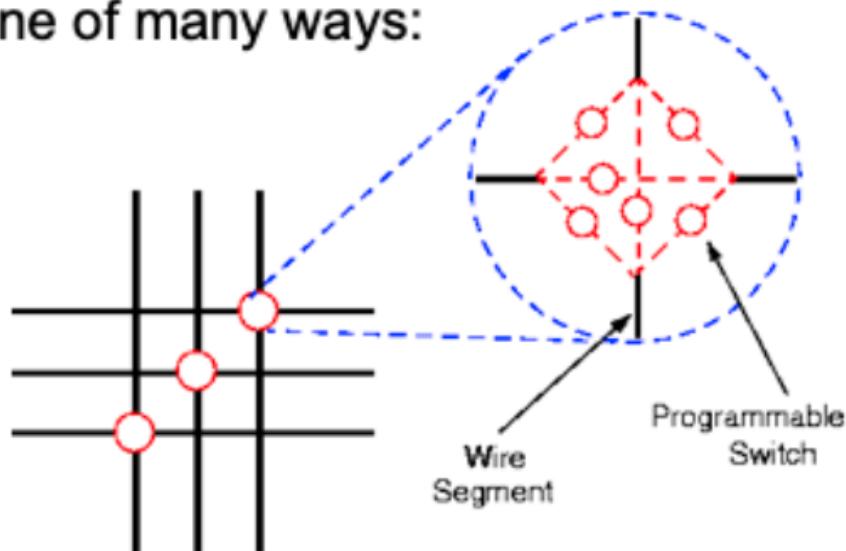
- ◆ Based around **Look-up Tables (LUTs)**, most common with 4-inputs
- ◆ Optional **D-flipflop** at the **output of the LUT**
- ◆ 4-input LUT can implement **ANY 4-input Boolean equation** (truth-table)
- ◆ Special circuits for cascading logic blocks (e.g. carry-chain of a binary adder)



- ◆ Each logic block has pins located for easy access:

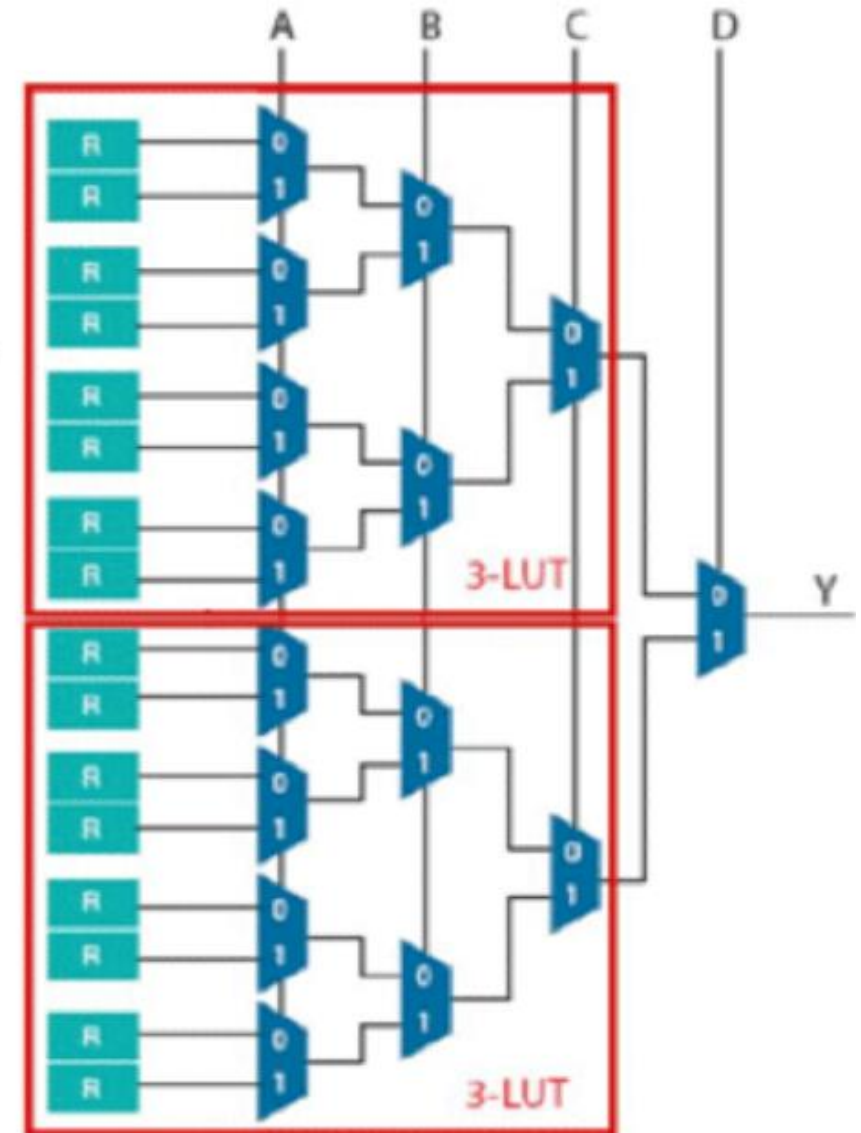
## Programmable Routing

- ◆ Between rows and columns of logic blocks are **wiring channels**
- ◆ These are programmable – a logic block pin can be connected to one of many wiring tracks through a programmable switch
- ◆ Xilinx FPGAs have dedicated switch block circuits for routing (more flexible)
- ◆ Each wire segment can be connected in one of many ways:



## The idea of configuring FPGA

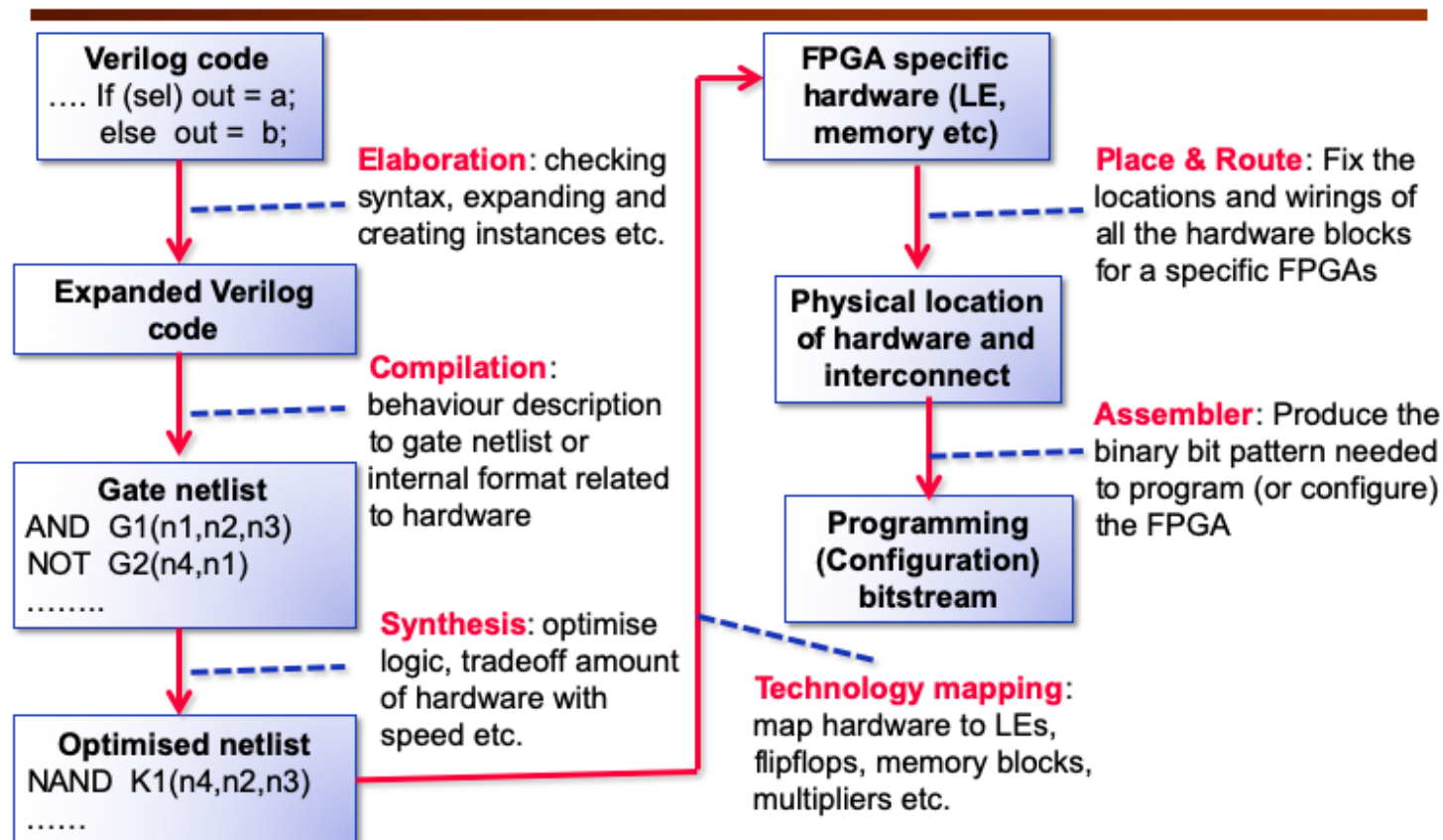
- ◆ Programming an FPGA is **NOT the same** as programming a **microprocessor**
- ◆ We download a **BITSTREAM** (not a program) to an FPGA
- ◆ Programming an FPGA is known as **CONFIGURATION**
- ◆ All LUTs are configured using the BITSTREAM so that they contain the correct values to implement the Boolean logic
- ◆ Shown here is a typical implementation of a 4-LUT circuit
  - ABCD are the FOUR inputs
  - There is four level of 2-to-1 multiplexer circuits
  - The 16-inputs to the mux tree determine the Boolean function to be implemented as in a truth-table
  - These 16 binary values are stored in registers (DFF)
  - Configuration = setting the 16 registers to 1 or 0



**Question:** How is a HDL description of a hardware module turned into FPGA configuration?

**Answer:**

### From Verilog code to FPGA hardware



**প্রশ্ন:** How to install Verilog?

উত্তরঃ To download and install Iverilog, Goto this site:  
<http://bleyer.org/icarus/>

Follow this YouTube tutorial:  
[https://www.youtube.com/watch?v=hg9splN\\_83Y](https://www.youtube.com/watch?v=hg9splN_83Y)

Make sure to click “Add Icarus Verilog executables  
folder to the system PATH”



উত্তরঃ For Linux installation,

```
sudo apt-get update
```

```
sudo apt-get install iverilog
```

```
sudo apt-get install gtkwave
```

**প্রশ্ন:** What should be IDE to write Verilog code?

উত্তরঃ My suggestion is VSCode. Goto to this link to download:

<https://code.visualstudio.com/download>

Follow this YouTube tutorial:

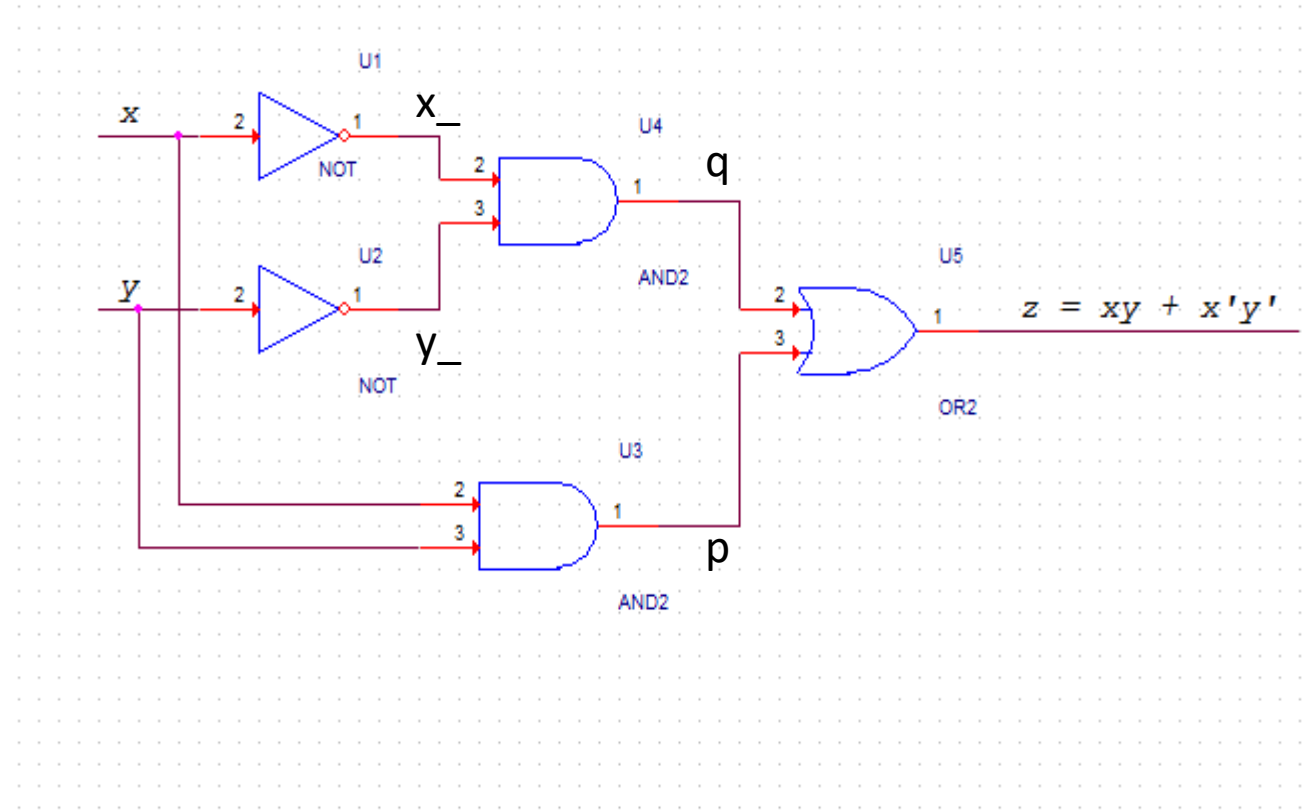
<https://www.youtube.com/watch?v=rwVFDfy2xVI>

Make sure to install Verilog addon for syntax highlighting.

**প্রশ্ন:** Write a simple comparator logic circuit in Verilog?

Input x	Input y	Output z
0	0	1
0	1	0
1	0	0
1	1	1

# উত্তরঃ



# উত্তরঃ

```
module comparator
(
input x,
input y,
output z
);

wire x_, y_, p, q;
not(x_, x);
not(y_, y);
and(p, x, y);
and(q, x_, y_);
or(z, p, q);

endmodule
```

comparator.v

# উত্তরঃ

```
module comparator
(
  input x,
  input y,
  output z
);

  wire x_, y_, p, q;
  not(x_, x);
  not(y_, y);
  and(p, x, y);
  and(q, x_, y_);
  or(z, p, q);

endmodule
```

→ Logic function name

→ Input port names

→ Output port names

→ Internal wires

→ NOT/AND/OR gate (output, input)

Here we have used Gate Level Modelling.

**প্রশ্ন:** Verify a simple comparator logic circuit written in Verilog?

Input x	Input y	Output z
0	0	1
0	1	0
1	0	0
1	1	1



উত্তরঃ Verilog এ লেখা কোড verify করতে আমরা test bench লিখি।

```
`timescale 1ns / 1ps
module stimulus;
// Inputs
reg x;
reg y;
// Outputs
wire z;
// Instantiate the
//Unit Under Test
//(UUT)
comparator uut
(
.x(x),
.y(y),
.z(z)
);
```

```
initial begin
    // Initialize Inputs
    x = 0;
    y = 0;

    #20 x = 1;
    #20 y = 1;
    #20 y = 0;
    #20 x = 1;
    #40;
end
```

```
initial begin
    $monitor("x=%d,y=%d,z=%d \n",x,y,z);
end
```

```
endmodule
```

**stimulus.v**

উত্তরঃ Verilog এ লেখা কোড verify করতে আমরা test bench লিখি। এই test benchকে পরে simulation করা হয়।

### Timescale

Simulation will run in steps of 1ns and has a precision value of 1ps.

### Test bench name

**Register**  
Inputs will be defined as reg.

### Instantiation

It creates an instance of the comparator module.

```
`timescale 1ns / 1ps
module stimulus;
  // Inputs
  reg x;
  reg y;
  // Outputs
  wire z;
  // Instantiate the
  //Unit Under Test
  //(UUT)
  comparator uut
  (
    .x(x),
    .y(y),
    .z(z)
  );
```

```
initial begin
  // Initialize Inputs
  x = 0;
  y = 0;
  #20
  x = 1;
  #20
  y = 1;
  #20
  y = 0;
  #20
  x = 1;
  #40;
end
```

```
initial begin
  $monitor("x=%d,y=%d,z=%d \n",x,y,z);
end
endmodule
```

Initializing inputs at the beginning of simulation.

# 20 means  
Wait for 20ns.

x = 1 means  
X value will be  
changed to 1 after  
waiting 20ns.

\$monitor means  
It will display the  
values of x, y and z.

**প্রশ্ন:** How to simulate Verilog program with test bench.

# উত্তর: Go to terminal in VSCode and write:

```
iverilog -o out.vvp comparator.v stimulus.v
```

```
# comment: out.vvp is the name output file after  
# compilation. "-o" defines output parameter.
```

```
vvp out.vvp
```

```
# comment: Show the output of the stimulus
```

```
#output:
```

```
x=0,y=0,z=1
```

```
x=1,y=0,z=0
```

```
x=1,y=1,z=1
```

```
x=0,y=1,z=0
```

প্রশ্ন: How to show graphical  
simulation/waveforms of Verilog program with  
test bench.

উত্তর: We will be using gtkwave.

First, add these two new lines in test bench:

```
`timescale 1ns / 1ps
module stimulus;
// Inputs
reg x;
reg y;
// Outputs
wire z;
// Instantiate the
//Unit Under Test
//(UUT)
comparator uut
(
.x(x),
.y(y),
.z(z)
);
```

```
initial begin
$dumpfile("test.vcd");
$dumpvars(0,stimulus);
// Initialize Inputs
x = 0;
y = 0;

#20 x = 1;
#20 y = 1;
#20 y = 0;
#20 x = 1;
#40;
end

initial begin
$monitor("x=%d,y=%d,z=%d \n",x,y,z);
end

endmodule
```

**stimulus.v**

উত্তর: Go to terminal in VSCode and write:

```
iverilog -o out.vvp comparator.v stimulus.v
```

```
# comment: out.vvp is the name output file after  
# compilation. "-o" defines output parameter.
```

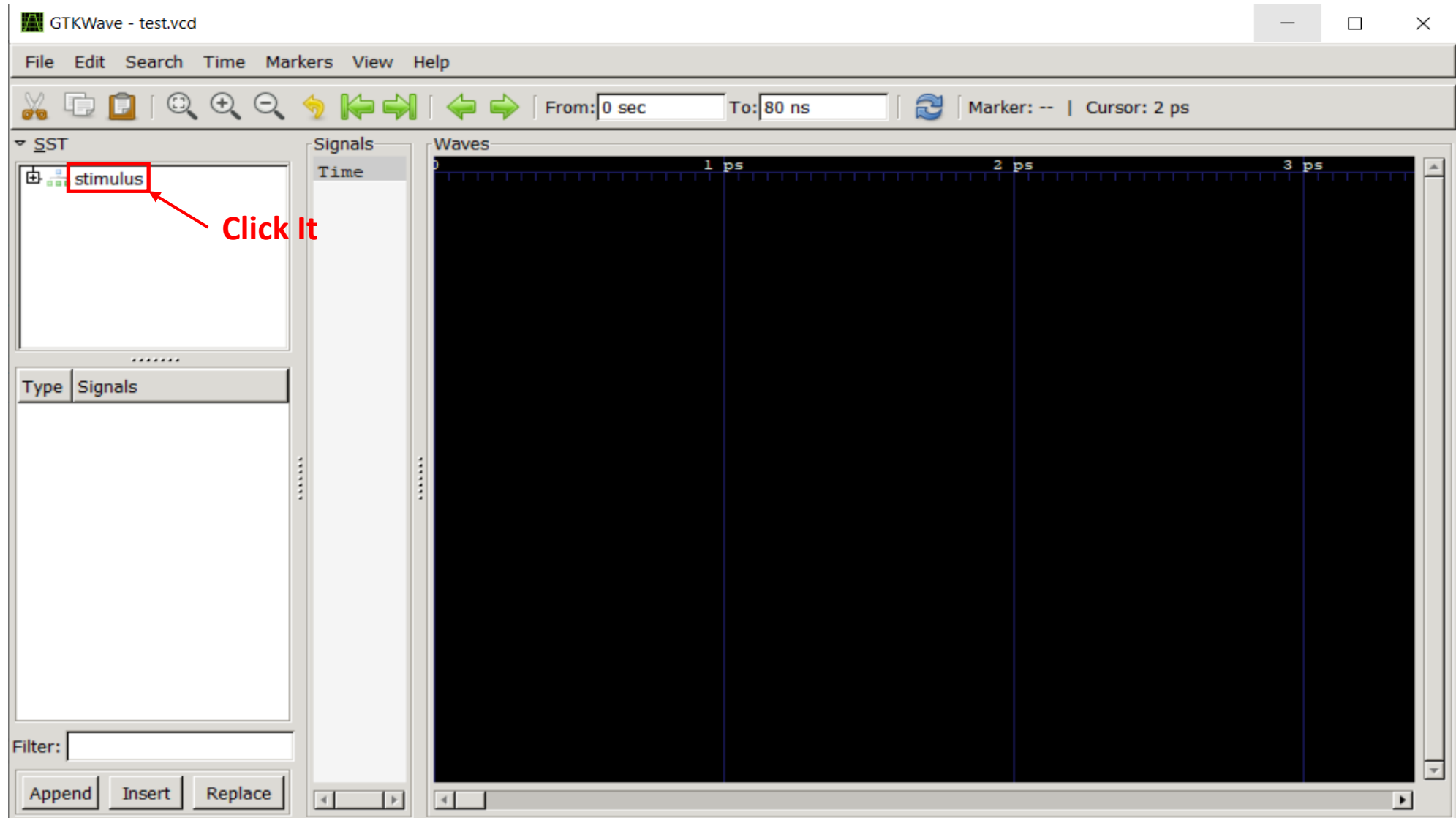
```
vvp out.vvp
```

```
# comment: Show the output of the stimulus
```

```
gtkwave test.vcd &
```

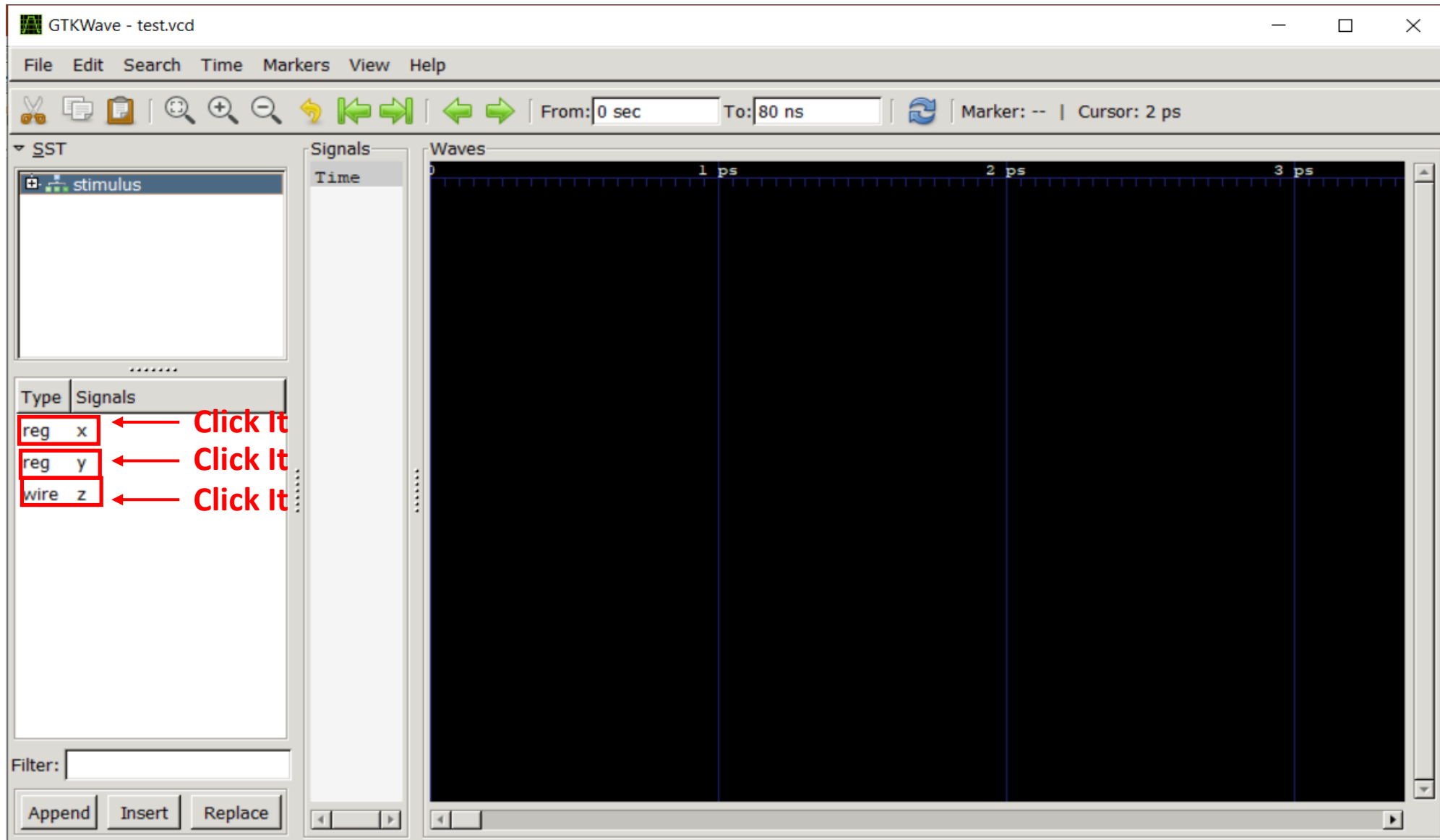
```
# It will show graphical simulation.
```

# উত্তরঃ

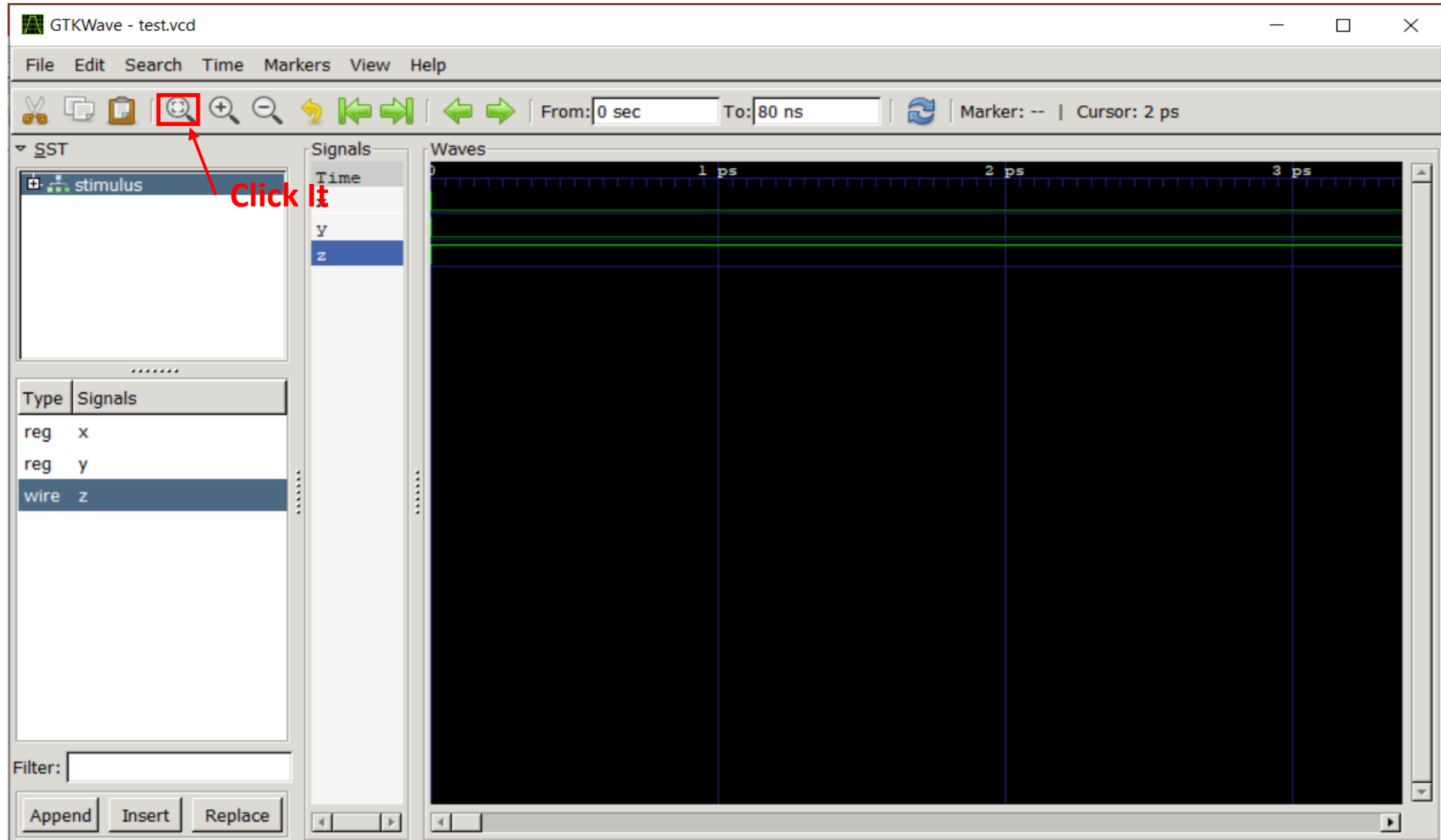




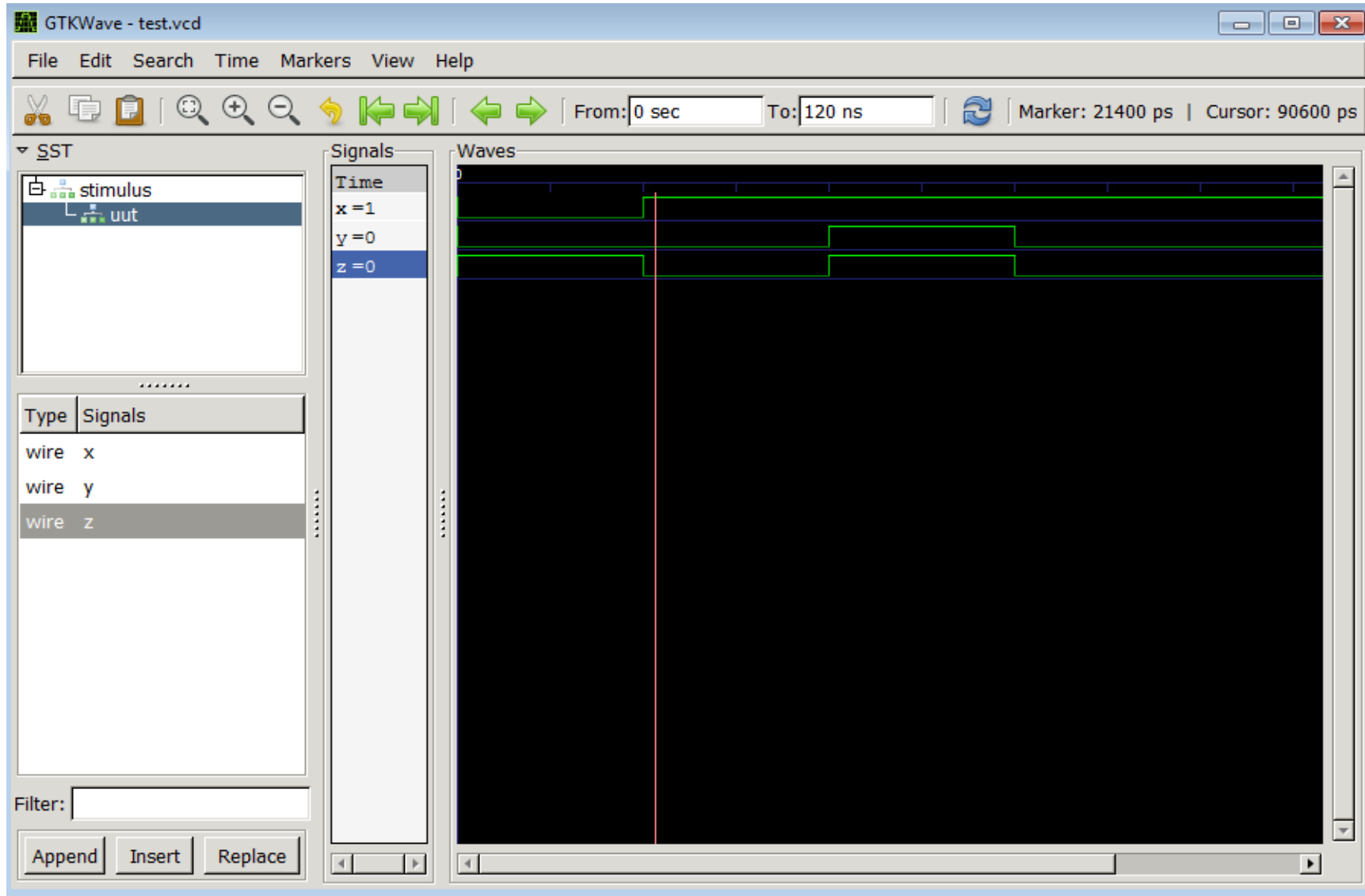
# উত্তরঃ



# উত্তরঃ



# উত্তরঃ



# Example: Verilog

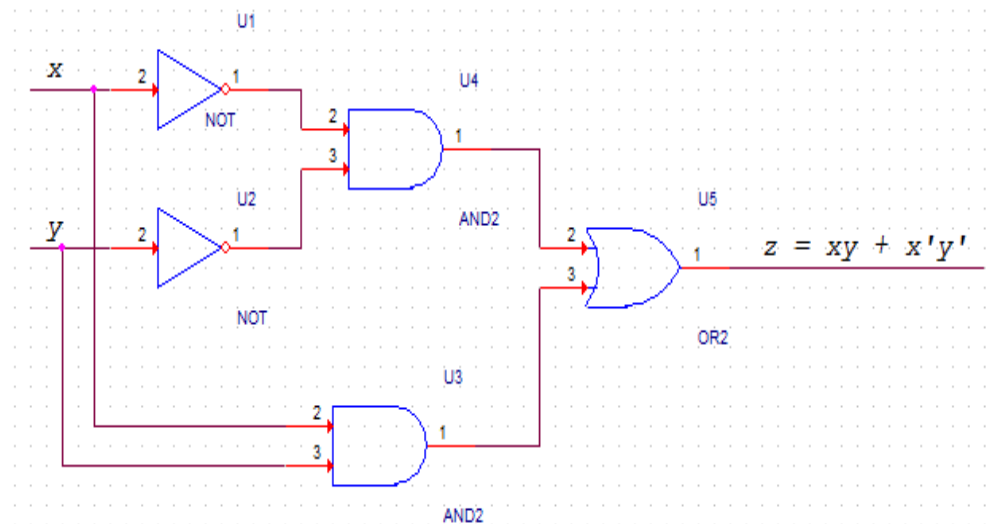
**Question:** Implement a **Comparator** in Verilog HDL along with a test bench.

**Answer:**

Boolean Table is shown below:    **Input x   Input y   Output z**

0	0	1
0	1	0
1	0	0
1	1	1

Resulting logic circuit which will be converted to Verilog is shown below:



# Example: Verilog

Verilog code of Comparator is shown below:

## comparator.v

```
module comparator
(
  input x,
  input y,
  output z
);

  wire x_, y_, p, q;
  not(x_, x);
  not(y_, y);
  and(p, x, y);
  and(q, x_, y_);
  or(z, p, q);

endmodule
```

# Example: Verilog

Test bench of Comparator is shown below:

## stimulus.v

```
`timescale 1ns / 1ps
module stimulus;
    // Inputs
    reg x;
    reg y;
    // Outputs
    wire z;
    // Instantiate the //Unit Under Test //(UUT)
    comparator uut
    (
        .x(x),
        .y(y),
        .z(z)
    );

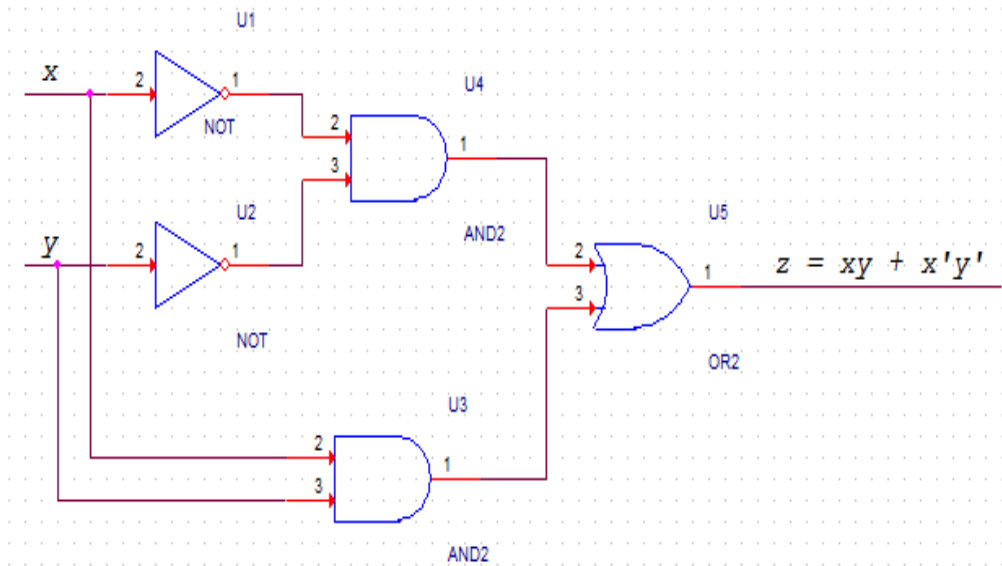
    initial begin
        // Initialize Inputs
        x = 0;
        y = 0;

        #20 x = 1;
        #20 y = 1;
        #20 y = 0;
        #20 x = 1;
        #40;
    end

    initial begin
        $monitor("x=%d,y=%d,z=%d \n",x,y,z);
    end
endmodule
```

# Exercises

1. Implement a Comparator in Verilog HDL along with a test bench.
2. Implement Boolean equation,  $Z = AC + BD$  in Verilog HDL along with a test bench.
3. Consider following circuit,



Implement this circuit in Verilog HDL. Provide a test bench to simulate this circuit.

Thank You 😊