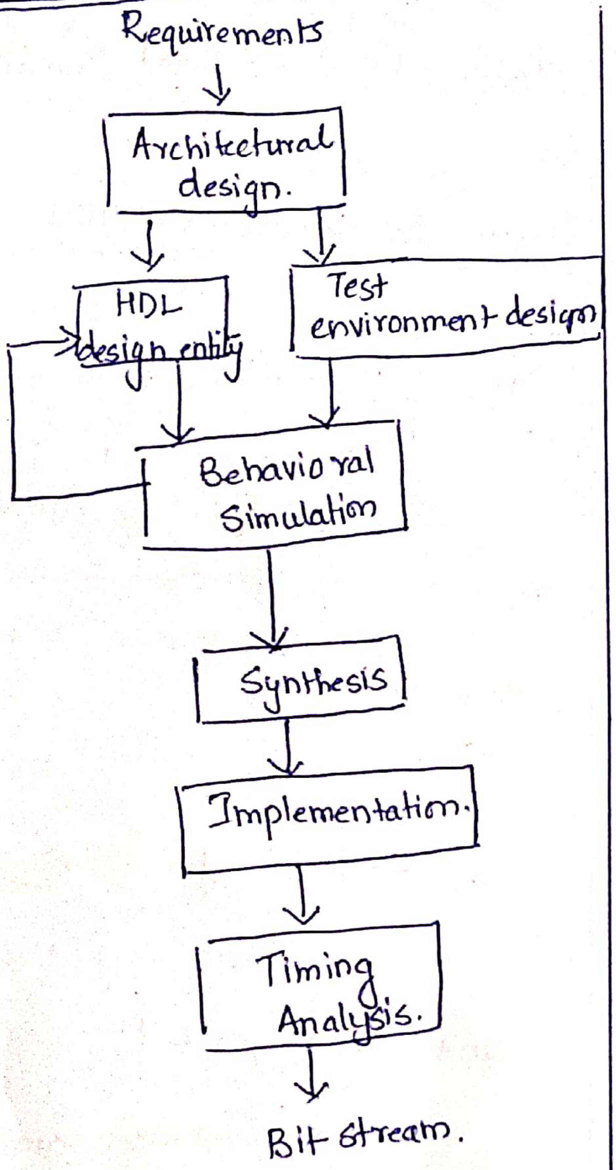
Vinay Reddy
ECE, PESU-EC

# HDL

## Hardware Description Language.

## FPGA DESIGN FLOW



## FEW TERMINOLOGIES

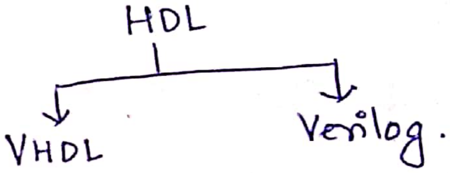
1. Design entity
   ↳ HDL.

2. Logic Simulation
   ↳ functional Verification of the design.
   ↳ We use the design file and the test bench file

3. Logic Synthesis
   ↳ Generation of netlist

4. Timing Verification
   ↳ Confirming that the fabricated, integrated circuit will operate at a Specified Speed.

5. Fault Simulation
   ↳ Comparing the behavior of an ideal circuit with the behavior of the Circuit that Contains a process-induced flaw.



## Verilog Code Syntax

VHDL
↳ VHSIC HDL.

Verilog

### Verilog syntax

Keyword

module example1 (a,b,y);

→ Ports

↳ name of the module (user defined) OR identifiers.

endmodule

Keyword

ⓐ Module decleration.

## (b) Verilog Ports.

1. Input port
2. Output Port

} unidirectional

3. inout port → Bidirectional.

### Port decleration

```
module example (a,b,y);
  input a, b;   } → Port decleration.
  output y;
     :
endmodule
```

## (c) Types of Verilog Styles.

1. Data Flow Style
2. Behavioral Style.
3. Structural Style.
4. Mixed Style.

Note : Verilog Code is case Sensitive.

## (d) Verilog Primitives.

Verilog Provides a standard set of Primitives, such as

- * and
- * nand
- * or
- * nor
- * not

} These are also Called as built-in Primitives. or System Primitives

**Note**

Verilog provides the ability to define User-Defined Primitives (UDP)

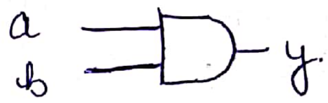
### Circuit to demonstrate HDL.

**Note**

and (y, a, b);

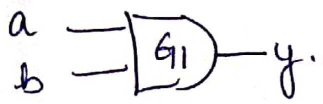
↳ always output port should be put first followed by the respective inputs.

equal to

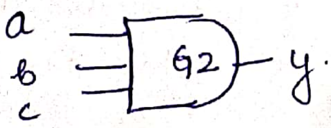


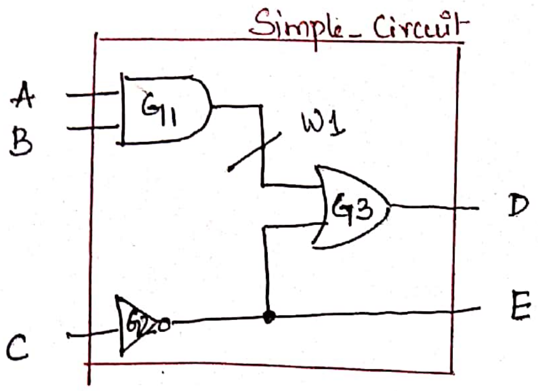
and G1 (y, a, b);

↳ instance name.



and G2 (y, a, b, c);





// Verilog model for simple_circuit

```
module simple_circuit (A,B,C,D,E);
  input A,B,C;
  output D,E;
  wire W1;

  and G1 (W1, A, B);
  not G2 (E, C);
  or  G3 (D, W1, E);

endmodule
```

→ Port decleration

→ Functionality in Structural style using Primitives

**Note:**

Please be clear with the terms instantiation and decleration.

## Gate Delays

`timescale 1ns/100ps

↳ Compiler directive.

When a HDL model is Simulated, It is sometime necessary to specify the amount of delay from the input to the output of its gates.

In Verilog propagation delay of a gate is specified in terms of time units and by the symbol #.

The number associated with the time delays in Verilog are dimensionless. The association of a time unit with physical time is made with the `timescale Compiler directive.

## Gate-level model with propagation delay.

```
'timescale 1ns/100ps.
Module simple_circuit (A,B,C,D,E)

Input A,B,C;
Output D,E;
Wire W1          → Gate delay
and #30 G1 (W1,A,B);
not #10 G2 (E,C);
or  #20 G3 (D,W1,E);

endmodule.
```

Compiler Directive.

```
'timescale 1ns/100ps.
```

```
'timescale <ref-time_unit>/<time Precision>
```

Test Bench code:

It is just a syntax, it dosent mean a sequential circuit.

```
module simplecircuit_tb;
reg A,B,C;
Wire D,E;
simple_circuit T1 (A,B,C,D,E); } → Design under Verification
initial → initial block.
```

```
begin
 A=1'b0; B=1'b0; C=1'b0;
#10 A=1'b0; B=1'b0; C=1'b1;
#10 A=1'b0; B=1'b1; C=1'b0;
 ↳ delay
        :
end
endmodule
```

Note: 1

* Initial block is a non-synthesizable block
* Initial block executes only once
* The initial block starts the execution at time zero.
* Initial blocks are sequential.

Note: 2

* If I have multiple statments, I should enclose them between a "begin" and "end" statments.

## User Defined Primitives (UDP)

The user can create additional Primitives by defining them in tabular form. These type of Circuits are referred as user defined Primitives.

General Rules :

1. It is declared with the Keyword primitive followed by the name and portlist.

2. There can be only one output, and it must be listed first in the port list and declared with the Keyword Output.

3. There can be any number of inputs. The order in which they are listed in the input decleration must confirm to the order in which they are given Values in the table that follows.

4. The truth table is enclosed within the Keywords table and endtable

5. The values of the inputs are listed in order, ending with a colon (:) The output is always the last entry in a row and is followed by a semicolon (;)

6. The decleration of a UDP ends with the Keyword endprimitive.

Example

Write an UDP for the function.

$f(A,B,C) = \Sigma(0,2,4,6,7)$;

// Verilog code for UDP example

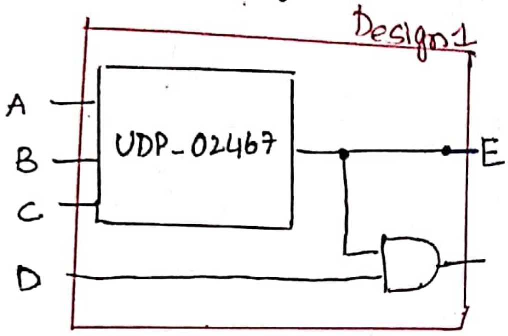
```
Primitive UDP_02467 (D, A, B, C);
Output D;
input A, B, C;

table

   000 : 1;
   001 : 0;
   010 : 1;
   011 : 0;
   100 : 1;
   101 : 0;
   110 : 1;
   111 : 1;

endtable
endprimitive
```

Eg: Write the verilog code for the circuit given below.



Solution

```
module design1 (A,B,C,D,E,F);
input A,B,C,D;
output E,F;
UDP_02467 U1(E,A,B,C);
and       U2(F,E,D);
endmodule
```

Structural Style or gate level Style

Eg: Write a verilog Code for the given boolean expression.

$D = AB + \overline{C}$

$E = \overline{C}$

## Verilog Operators

Logical operators.

! (logical negation)

&& (logical and)

|| (logical or)

Bit-wise operators

~ (negation)

& (and)

| (or)

^ (exor)

^~ or ~^ (xnor)

## Dataflow Style Verilog Coding

```
module sample (A,B,C,D,E);
 input A,B,C;
 Output D,E;

 assign D = (A&&B) || (!C);
 assign E = !C;

 endmodule
```