Module #01 : Introduction to Verilog HDL

- Verilog HDL is one of the two most common Hardware Description Languages (HDL) used by integrated circuit (IC) designers. The other one is VHDL.
- Designs described in HDL are technology-independent, easy to design and debug, and are usually more readable than schematics, particularly for large circuits.
- Verilog is used as an input for synthesis programs which will generate a gate-level description (a netlist) for the circuit.
- The way the code is written will greatly effect the size and speed of the synthesized circuit
- Some Verilog constructs are not synthesizable. (eg. delay)
- Non-synthesizable constructs should be used only for test benches.

Note: "Verilog is NOT a Programming Language. It is a Hardware Description Language !!!"

Programming Language : Let the hardware perform a particular function
HDL : Design that Hardware !!

Module #01 : Introduction to Verilog HDL

- There are two types of code in most HDLs:

  1) ***Structural***, which is a verbal wiring diagram without storage.
  Example 1.1:
  assign a=b & c | d;
  assign d = e & (~c);
  2) ***Procedural***, which is used for circuits with storage, or **as a convenient way to write conditional** logic.
  Example 1.2:
  always @(posedge clk) // Execute the next statement on every rising clock edge.
  count <= count+1;

- **if** and **case** statements are only allowed in procedural code. As a result, the synthesizers have been constructed which can recognize certain styles of procedural code as actually combinational.

Module #01 : Introduction to Verilog HDL

Verilog can be used to describe designs at four levels of abstraction:

(i) Switch level (the switches are MOS transistors inside gates).
(ii) Gate level (interconnected AND, NOR etc.).
(iii) Register transfer level/Data Flow Level (RTL uses registers connected by Boolean equations).
(iv) Algorithmic level/Behavioral Level (much like c code with if, case and loop statements).

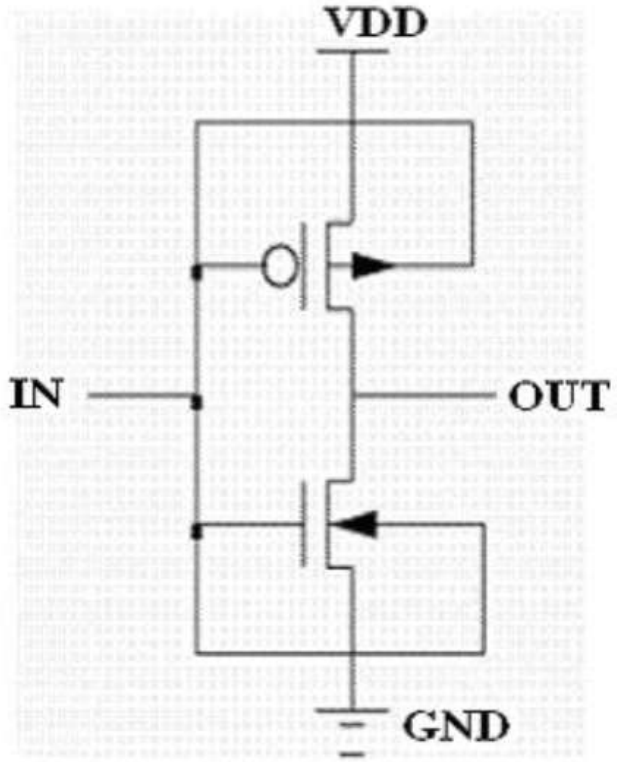
Module #01 : Introduction to Verilog HDL

(i) Switch level (the switches are MOS transistors inside gates).

Example 1.3:

```
module inverter (IN, OUT);
   input IN;
   output OUT;
   supply0 VSS;
   supply1 VDD;

    pmos p(OUT, VDD, IN);
    nmos n (OUT, VSS, IN);
endmodule
```

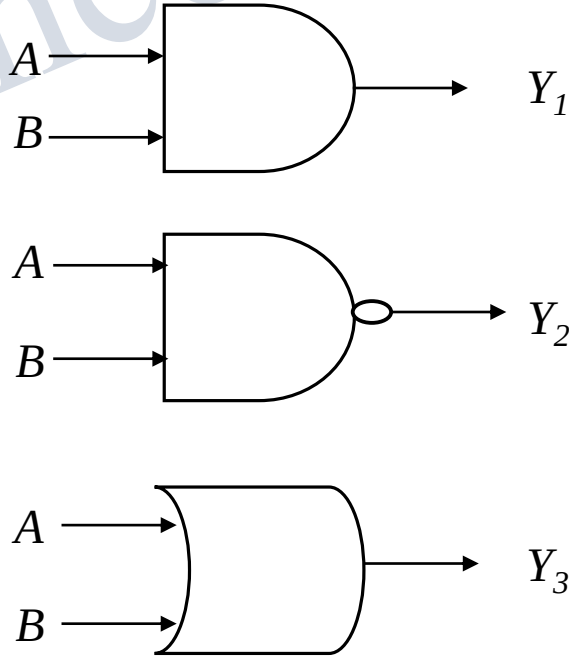


Module #01 : Introduction to Verilog HDL

(ii) Gate level (interconnected AND, NOR etc.).

Example 1.4:

```
module myDesign(A, B, Y);
    input A;
    input B;
    output Y;
    and A1(Y1, A,B);
    nand NA1(Y2, A, B);
    or O1(Y3, A,B);
endmodule
```

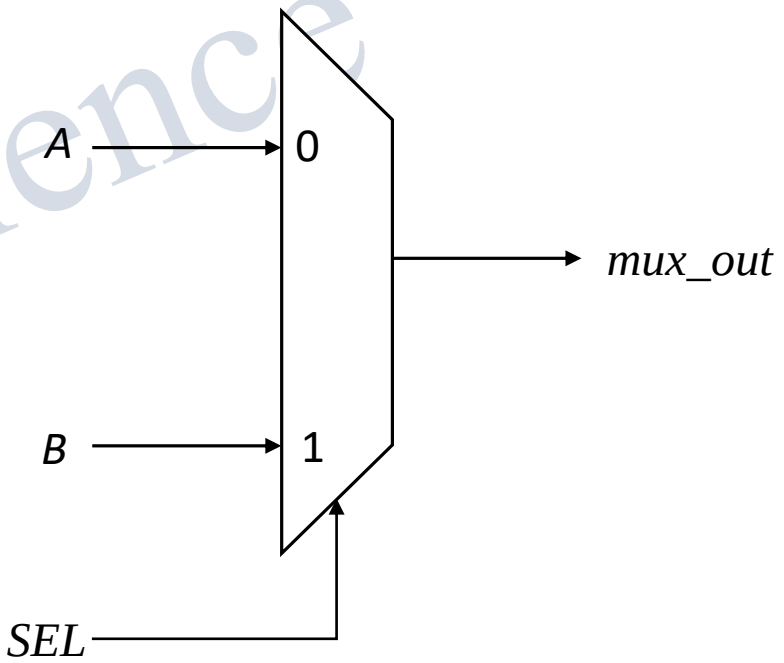


Module #01 : Introduction to Verilog HDL

(iii) Register transfer level/Data Flow Level

Example 1.5:

```
module myDesign(A, B, SEL, mux_out);
    input A, B, SEL;
    output mux_out;
assign mux_out = (SEL` & A) + (SEL & B);
endmodule
```



Module #01 : Introduction to Verilog HDL

(iv) Algorithmic Level/Behavioral Level

Example 1.6:

```
module myDesign(A, B, SEL, mux_out);
      input A, B, SEL;
      output  mux_out;
      reg mux_out;
 always @(A, B, SEL)
   begin
      if(SEL)
            mux_out = B;
      else
            mux_out = A;
    end
```

Verilog Modeling

Design Behavior

```
if (SEL = 1)  mux_out = B;
else mux_out = A;
```