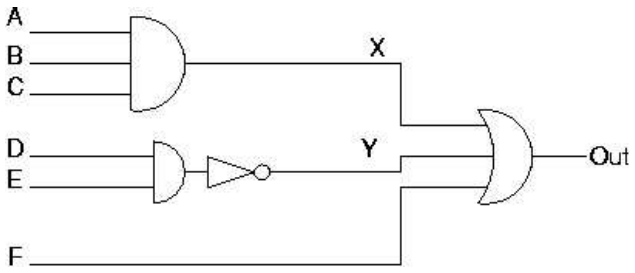
Intro to Verilog

A quick tutorial introduction to verilog

Outline

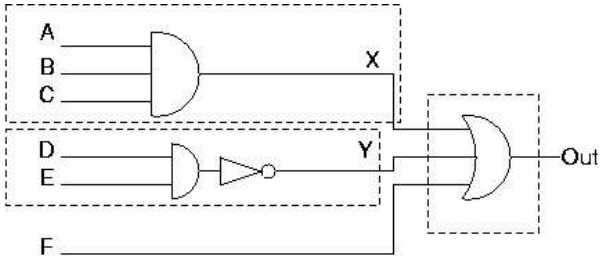
- Logic Overview
  - Logic Networks
- Verilog
  - Structural Verilog
    - Description
    - Example
  - Behavioral Verilog
    - Description
    - Example
  - Testing

Basic Logic Network



Out = A*B*C + (DE)' + F

Network Subdivision



X = A*B*C Y = (D*E)' Out = X + Y + F

Verilog

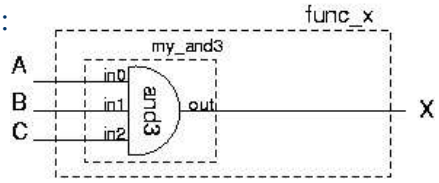
- Unlike other programming languages, verilog is a formalized description of hardware.
- Verilog statements are concurrent (unless otherwise noted)
- Two types of verilog:
  - Structural: Allows interconnection of pre-built blocks.
  - Behavioral: Allows use of high level constructs to describe function of hardware w/o having to describe the details.

Structural Verilog

- Defines the interconnections between blocks.
- Can be thought of as a “Text Schematic Diagram”

Structural Verilog Example

Schematic:

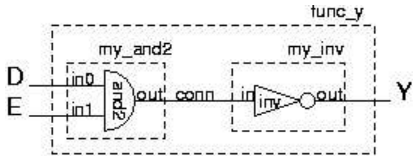


Structural Verilog:

```
module func_x ( X, A, B, C ) ;
   output X;
   input  A, B, C;
   and3 my_and3 ( .out(X), .in0(A), .in1(B), .in2(C));
endmodule
```

Structural Verilog Example Cont.

Schematic:

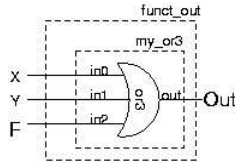


Structural Verilog:

```
module func_y ( Y, D, E ) ;
   output Y;
   input  D, E;
   wire   conn;
   and2 my_and2 (.out(conn), .in0(D), .in1(E));
   inv my_inv (.out(Y), .in(conn));
endmodule
```

Structural Verilog Example Cont.

Schematic:

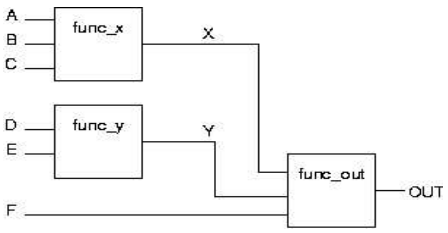


Structural Verilog:

```
module func_out ( OUT, X, Y, F ) ;
   output OUT;
   input  X, Y, F;
   or3 my_or3 ( .out(OUT), .in0(X), .in1(Y), .in2(F));
endmodule
```

Structural Verilog Example Cont.

Schematic:



Structural Verilog:

```
module top ( OUT, A, B, C, D, E, F ) ;
   output OUT;
   input  A, B, C, D, E, F;
   wire   X, Y;
   func_x my_x ( .X (X), .A (A), .B (B), .C (C));
   func_y my_y ( .Y (Y), .D (D), .E (E));
   func_out my_out ( .OUT (OUT), .X (X), .Y (Y), .F (F));
endmodule // top
```

Behavioral Verilog

- Allows the designer to specify the function of the device being designed.
- Designer works at a higher abstraction level.
- Synthesis tools produce structural verilog from this.

Behavioral Verilog Example

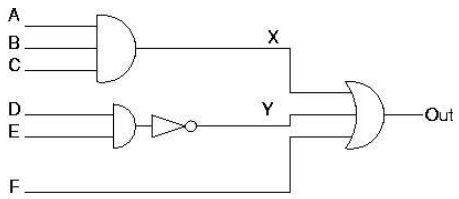
Three Input And Gate:

```
module and3 ( out, in0, in1, in2 ) ;
   output out;
   input  in0, in1, in2;
   assign out = in0 & in1 & in2;
endmodule // and3
```

(p92 in Palnitkar for other operators)

Behavioral Verilog Example Cont.

Schematic:



Behavioral Verilog:

```
module top_2 ( OUT, A, B, C, D, E, F ) ;
   output OUT;
   input  A, B, C, D, E, F;

   assign OUT = (A & B & C) | (~(D & E)) | F;
endmodule
```

Testing

- Create a verilog file that wraps around the top level of your design called a “testbench” or “stim file”.
- Provides inputs to your design that change with time.
- Checks that the output’s are correct for those inputs at any given time.
- You will be provided with a test bench for the first couple assignments.