Lecture - 4

☆ Recap: · Register Data Types
- → reg
- → integer
- → real
- → time

· Vectors
· Multidimensional Arrays & Memories
· Constant values
· Parameters (Equivalent to #define)

☆ Predefined Gates in Verilog

· Verilog supports 4 logic values : 0, 1, X, Z

· Gates : & (AND), | (OR), ~ (NOT), ^ (XOR)

☆ Laboratory Assignment (To be submitted via mail)

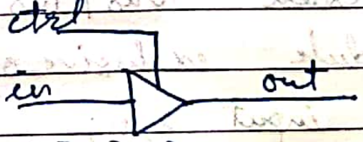
* Write verilog codes to verify o/p of the following:

1 & x	1 | x	1 ^ z
0 & x	1 | z	0 ^ x
1 & z	0 | x	1 ^ x
0 & z	0 | z	0 ^ z

☆ List of Primitive Gates

```
and   G(out, in1, in2);
nand  G (out, in1, in2);
or    G (out, in1, in2);
nor   G (out, in1, in2);
xor   G (out, in1, in2);
xnor  G (out, in1, in2);
not   G (out, in);
buf   G (out, in);
```

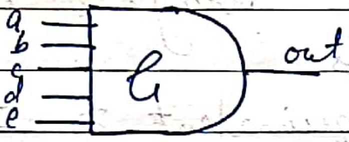
```
bufif1 G(out, in, ctrl);
```



```
if ctrl=1
    out = in
else
    out = Z
end
```

```
bufif0 G(out, in, ctrl);
notif0 G(out, in, ctrl);
notif1 G(out, in, ctrl);
```

* Example: 5 i/p AND gate:

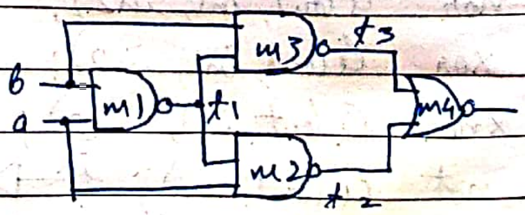


```
AND  G(out, a, b, c, d, e);
```

★ Restrictions while instantiating primitive gates
- output port must be connected to a net (Eg: a wire) (output signal is a wire by default)
- input port may be connected to NET or REGISTER type variable.
- primitive gates (except not/buf) can have a single o/p but any no. of i/ps.
- optional delay can be added as follows:
  and #5 G1(f, a, b);
  ↳ this delay is used only for simulation (#5 means time specified on time scale)

★ Example:

```
`timescale 10ns/1ns
module exclusive_or (f, a, b);
    input a, b;
    output f;
    wire t1, t2, t3;
    nand #5 m1 (t1, a, b);
    and  #5 m2(t2, a, t1);
    and  #5 m3 (t3, t1, b);
    nor  #5 m4(f, t2, t3);
endmodule
```



• 'timescale 10ns / 1ns

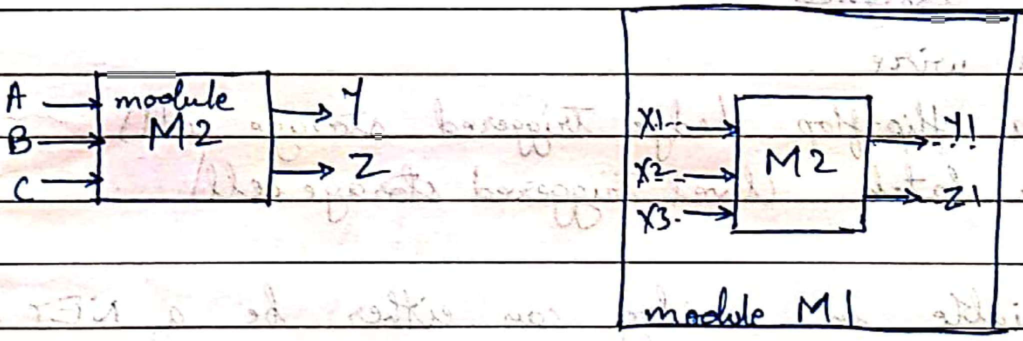
$\xrightarrow{}$ precision of simulation (it is the precesion to which delays are rounded)

basic unit of time so #1 = 10ns & #5 = 50ns

• valid values for specifying time unit & time precision are 1, 10 and 100.

• valid units: s, ms, μs, ns, ps, fs;

★ Specifying Connectivity during Instantiation



• instantiation of one module inside another.

• Two ways to specify the connections:

a) Positional Association

The parameters of M2 are listed in same order as in the original module description

Eg:

```
module  M2(A,B,C,Y,Z)
________________________________

module  M1(                )
   :
   M2  DUT(X1,X2,X3,Y1,Z1);
   :
endmodule
```

DUT: Device Under Test

b) Explicit Association

Parameters can be in arbitrary order. Here, chance of error is also less.

Eg:

```
module M1 (          )
  :
  :
  M2  DUT(Y1(Y), A1(A), C1(C), Z1(Z), B1(B));
endmodule
```

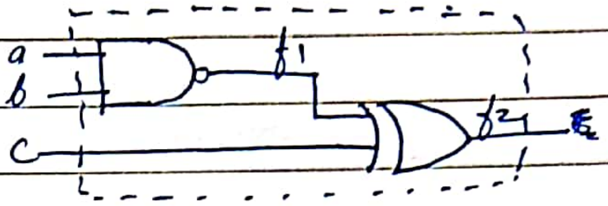
☆ Hardware Modeling Issues

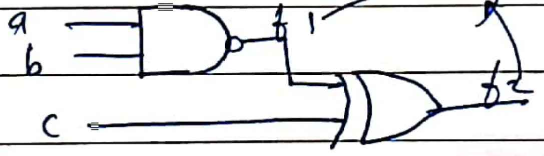
* In terms of h/w realization, the computed value can be assigned to:
  - a wire
  - a flip-flop (edge triggered storage cell)
  - a latch (level triggered storage cell)

* A variable in verilog can either be a NET or a REGISTER.
  - NET always maps to a wire.
  - REGISTER maps either to a wire or a storage cell.

★ Example:

```
module logic_ckt (a, b, c, f1, f2)
    input a, b, c;
    output f1, f2;
    reg f1, f2;
    always @ (a or b or c)    //Execute this block if
    begin                     //there is event on a, or b, or c.
          f1 = ~(A && B);
          f2 = f1 ^ c;
    end
endmodule
```

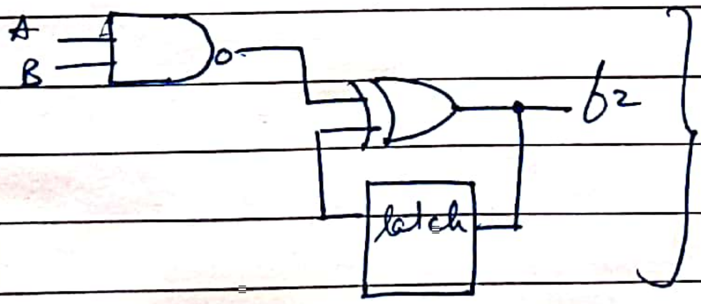




→synthesized as wire as no value is being stored.

★ Example:

```
f2 = f1 ^ f2;
f1 = ~(A & B);
```



Equivalent h/w