VERILOG

HDL : hardware discription lang

VHDL

written as a descriptive lang.

behaviour of essential system to be verified advance of the synthesis tools translation.

- wordy
- easy to understand

Verilog

written as hardware modelling language.

similar to C;

lower level of programming construct.

succinct

less code to write

Verilog:

4 levels of Abstraction:

(basically depending upon the purpose of designing internals can be designed at 4 different levels).

→ Behavioral level: Just focusing on the input & output. (procedural statements) (Str dont care):

Always @: happen recurs
Initial @: once done.

→ Dataflow level:

ASSIGN Keyword

→ Gate level

structures are VVIMP;

Predefined gates in V.log

→ Switch level

help of transistors:

coding @ transistor levels:

input [3:0] a;

Learn TTL

Verilog datatype: → Vectors:

- Registers
  - ↳ any assining will be of reg type
  - → Used for data storage: (tank)
- Nets → must be continously driven (pipe)

→ reg
→ integer
→ real
→ time

Nets:
- wire
- wand
- wor
- tri
- Supply0
- Supply1.

→ Same when multiple drivers are driving them.
→ insert and/or gate at the junction. (Multiple type)

Default z

Why integration:

1) More processing (DSP, ALU, BUS, DMA)
2) Lesser power consumption (C_L decreases)
3) Low cost (Lesser area)
4) Reliability (Same fabrication)
5) Portability (Smaller ~~footprint~~)

Registers:

1) Reg : Default X (don't cares).
behavioural me assign ke liye reg is used.

2) Int : 32 bit integers

3) Real : Real no when typecasting → Roundoff.

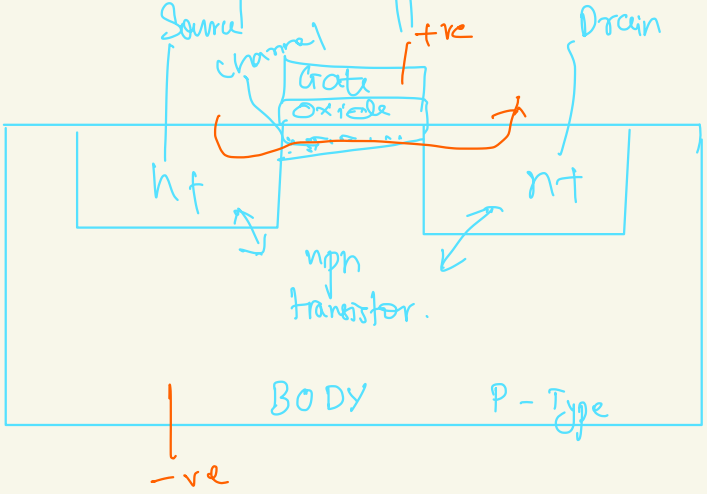
4) Time : Use current simulation time.

beh : always@
dataflow : assign
gate :
Switch :

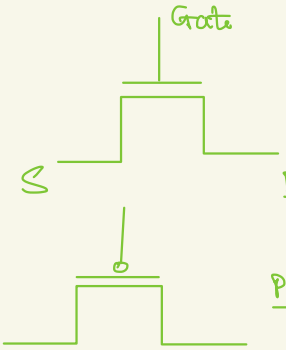
Verilog Course:

1) Intro
2) Verilog Basics
3) Combinational Logic
4) Sequential Logic
5) Fundamental Circuits
6) Static Timing Analysis
7) Good/Bad HDL Analysis
8) FPGA Implementation.

CMOS: Complimentary Metal Oxide Semiconductor.



So Gate Voltage acts as a switch.



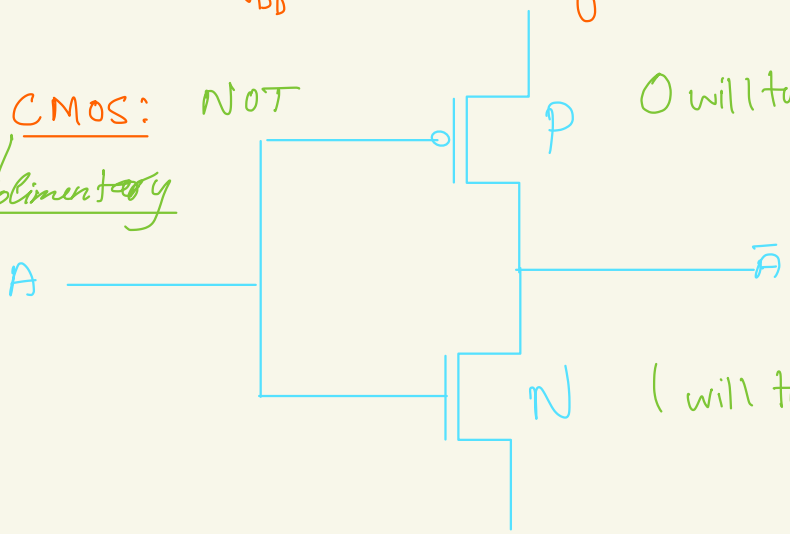
NMOS.
NPN transistor.
Active high.

PMOS.
PNP transistor
Active low.

Oxide: It is present to give an indirect effect of the gate Voltage.

V_{DD} : Gate Voltage :

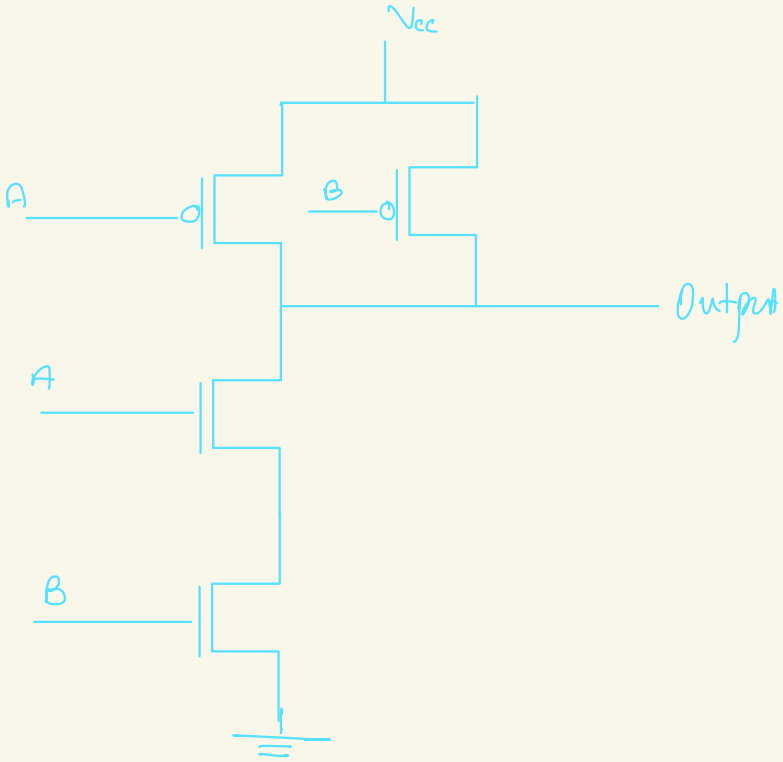
CMOS: NOT
Complimentary



0 will turn this on

1 will turn this on.

NAND:



CMOS:

- Low power
- Fully restored logic levels.
- On & off times (similar biasing times).
- High integration
- High performance.

Why integration?

1) More processing
2) Less area
3) Higher speed
4) Low power
5) Portability
6) Reliability.

Minimum feature size:

→ technology:

→ the distance between S & D.

VLSI

- Analog VLSI [behaviour analysis] of oscillators.
  → ADC's, DAC's
  ↓
  Analog to Digital Converter.
- Digital VLSI
  ALU: Arithmetic Logic Unit.
- EDA tool design.

ALV: Arithmetic Logic Unit

RTL: Register transfer level

ASIC: Application Specific Integrated Circuits.

- Design Specifications: Getting
- Architecturing
- RTL — Jobs
- Verification
- Synthesis — Same Team.
- DFT: Design for testability. — Jobs.
- Timing analysis.
- Floorplanning
- Placement
- Routing (DRC: Design Rules) → Jobs:
- Formal Verification: Checking functionality.
- Power Estimation: Switching causes large power consmp^{n}.
- Fabrication: Lithography
- Packaging

Design spec: → func^{n}; power; speed, size, input, costs; etc.
→ logic placement.

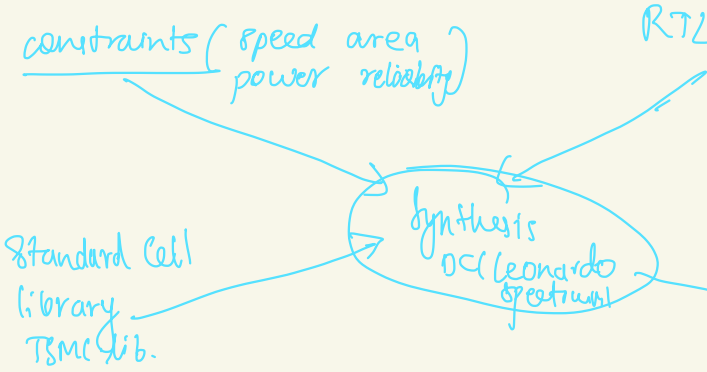
Architecturing: → IP's; master & slave processor, memory; DSP; RF; DAC; ADC;
Planning → own peripherals
Clocks number ??

BUS ARCHITECTURE

RTL Coding: We can write in C also and use HSU: high level encoder.

Verification: Writing testbenches, simulations. (Model frm)

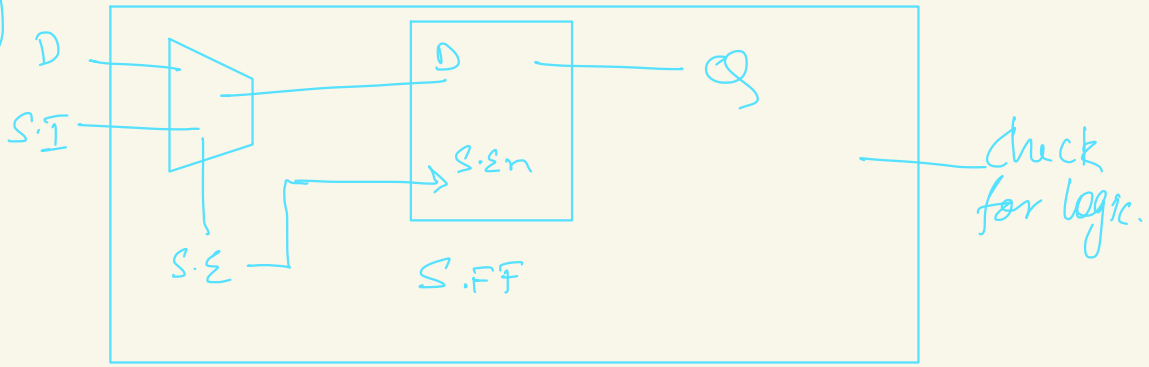
Synthesis: Code to hardware:



Design for testability
- DFT insertion (design phase) — add circuits to check for
- Test applic^{n} (after manf.)

TESSENT

Controlability
Observability.



Timing Analysis
- setup time
- hold time

Max delay: critical path.

CLOCK TREE NTHD

Floor Planning, Placement & Routing: Just basic stuff.

Design

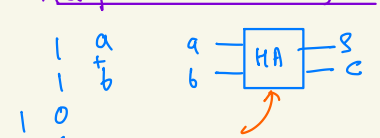
1) Dataflow:
   - → Boolean
   - → comb logic
   - → Circuits
   - → Muxes
   - → ASSIGN

2) Behavioral
   - → Circuit / Bool exp not known
   - → All seq elements latches / FlipFlops

3) Str:
   - M1
   - M2
   - used to integrate blocks.

while naming module (outputs first, then inputs)

Half Adder: Dataflow:



$S = a \oplus b$

$C = a \cdot b$

module: output to input.

```
module HA_df (S,C, a,b );
    input  a,b;
    output S,C;
    assign S = a^b ;
    assign C = a&b;
endmodule
```

both are concurrent in nature all are calculated at the same time.

Half Adder : Behavioral

```
module HA_Bh (S,C,a,b)
input a,b;
output reg S,C;
always @ (a,b)
begin
    S = a^b;
    C = a&b;
end
endmodule
```

helps to hold value:

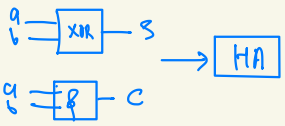
sensitivity list; only executes when a,b changes

one block

reg: helps hold value:

apart from reg; default is wire:

Half Adder : Structural




```
module HA_St (S,C,a,b)
input a,b;
output S,C;
primitives  xor  xor1 (S, a,b);
            and  and1 (C,a,b);
                 ↳ my copies instances;
endmodule
```

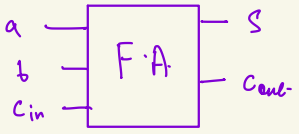
Verilog is case Sensitive.

Synthesis will be the same;

Structural : FA;

$S = a \oplus b \oplus C_{in}$

$C_{out} = a \cdot b + b \cdot c_{in} + c_{in} \cdot b$.



df:

```
module FA_df (S, Cout, a, b, cin)
    input a, b, cin;
    output S, cout;
    assign S = a ^ b ^ cin;
    assign Cout = a&b | b&cin | cin&a;
endmodule
```

bh:

```
module FA_bh (S, Cout, a, b, cin)
    input a, b, cin;
    output reg S, Cout;
    always @( a, b, cin);
    begin
      S = a ^ b ^ c;
      Cout = a&b | a&cin | b&cin;
    end
endmodule.
```

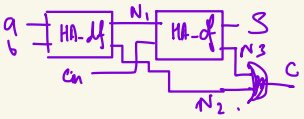
st:

```
module FA_St (S, Cout, a, b, cin)
    input a, b, cin;
    output S, Cout;
    wire S1, c1, c2, c3;
    xor xor1 (S1, a, b);
    xor xor2 (S, S1, cin);
    and and1 (c1, a, b);
    and and2 (c2, a, cin);
    and and3 (c3, b, cin);
    or or1 (Cout, c1, c2, c3);
endmodule.
```

when not a lot of names

Testbench:

→ time



HA_df(S, C, a, b).

```
module FA_St3 (ss, cc, aa, bb, cin)
    inp aa, bb, cin;
    Out ss, cc;
    wire N1, N2, N3;
    HA_df HA_df1 (.S(N1), .C(N2), .a(aa), .b(bb));
    HA_df HA_df2 (.S(ss), .C(N3), .a(N1), .b(cin));
    or or (cc, N2, N3);
endmodule.
```

When lot of names to be connected :

```
module FA (a, b, cin, S, cout)
    input a, b, cin;
    output S, cout;
    assign S = a ^ b ^ cin;
    assign Cout = a&b | b&cin | cin&a;
endmodule:
```