1/1

Introduction to VHDL

What is VHDL ?

* VHDL is a short form of VHSIC Hardware Description Language where VHSIC Stands for Very High Speed integrated circuits.
* it's a hardware description language — means it describes the behavior of a digital ckt and also it can be used to derive or implement a digital circuit / system.
* it can be used for digital circuit Synthesis as well as Simulation.
* it is used to build digital System / ckt using Programmable logic device like CPLD (Complex Programmable Logic Device) or FPGA (Field Programmable Gate Array).
* VHDL Program (Code) is used to implement digital circuit inside CPLD / FPGA, or it can be used to fabricate ASIC (Application Specific integrated circuit)
* it is very useful in developing high end, Sophisticated microprocessor or micro Controller like ASIP (Application Specific instruction Processor) or PSoC (Programmable System on Chip)

__/__/__

Now before going into more details about VHDL, let us first see how and why there was a need for VHDL.

Why VHDL ?

In 1980's US DoD (Department of Defence) initiated the VHSIC Program.

* Different hardware designing Companies started developing their ICs with their own HDL. All Companies had their own & different HDL.
* So the Problem was all these different Companies Cannot exchange their code & designs with another.
* Also, all Companies Provide their chip design to DoD with different HDL.
* So there was a requirement to Standardized hardware description language for digital circuit / system design, documentation, and Verification.

Advantages of VHDL →

* its Vendor-Indepedent.
* it is Portable.
* it is reusable.
* it Supports hierachical design — whole big and Complex System Can be modelled as an interConnection of Small Components, and

again Components Can be further modelled as an interConnection of subComponents.

* All Statements of VHDL Program are executed Concurrently (unless & untill the Statement are placed inside Procedure, function or Process)
* it is human - readable as well as machine readable.
* it is IEEE & ANSI Standard.
* Supports different design methodologies like top - down, bottom-up, mix etc.
* Can be used to design Combinational, Sequential or mixed digital circuits using three different methods.

1) data flow 2) behavioural 3) Structural

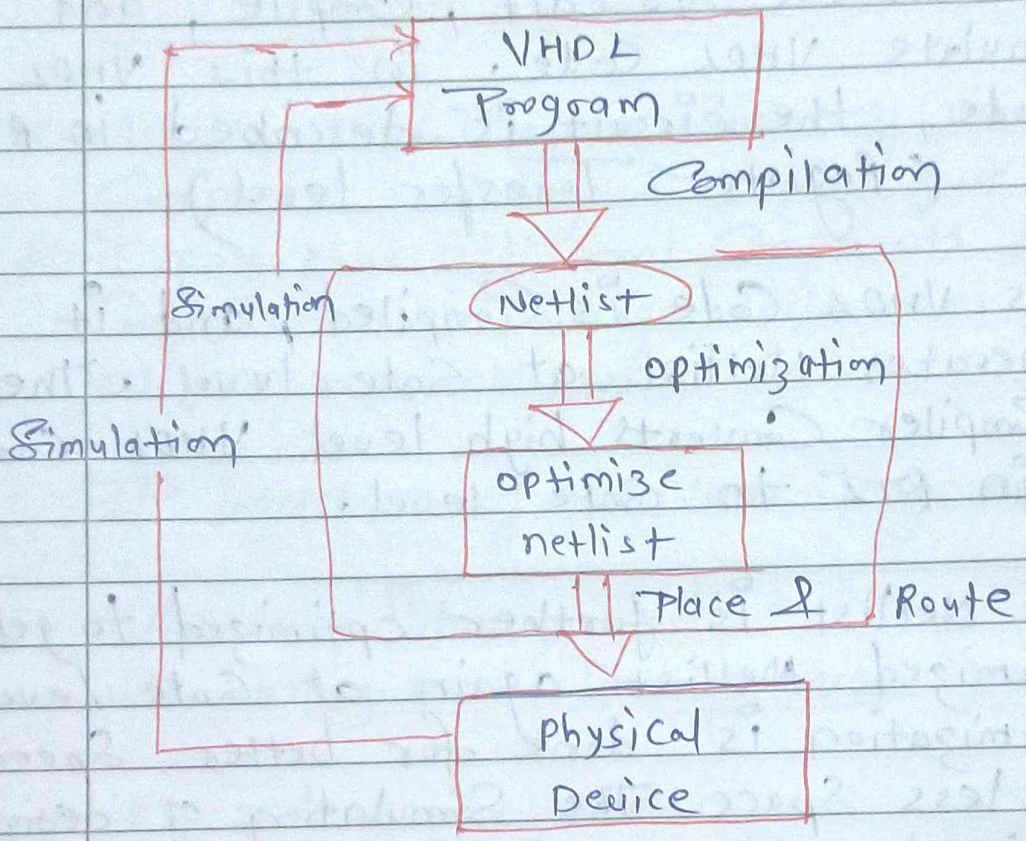
Brief history of VHDL origin →

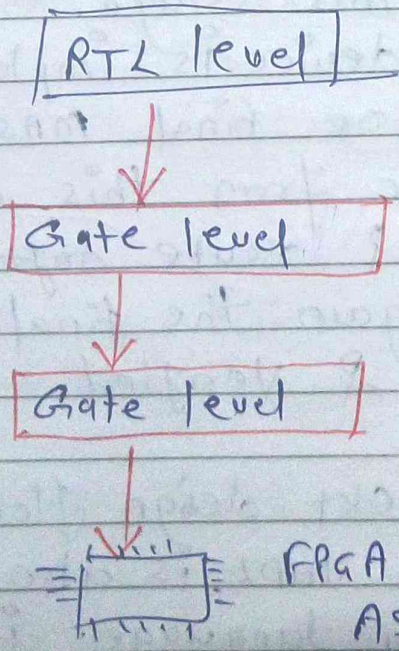
1) in 1985 the first Version of VHDL 7.2 was developed by IBM, TEXAS INST, and intermatrix under a Contract of DOD.
* in 1987 it was Standardized by IEEE with IEEE 1076 Standard. After than new Standard IEEE 1164 was given to VHDL that is now a day's used every where.
* ANSI also recognizes VHDL, and Standard VHDL reference manual is made available by IEEE that has an official description of this VHDL.

__/__/__

Now after getting enough information about VHDL. let us move ahead with designing digital circuits using VHDL.

VHDL Design Flow →





1) VHDL design flow starts with writing the VHDL program. Various manufacturing Companies likes XILINX, Altera etc. Provide their own Software development tools like XILINX ISE, Altera Quartus etc. to edit, Compile, and Simulate VHDL Code. in this VHDL Code, the circuit is described in RTL (Register Transfer level)

2) This VHDL Code is Compiled, and it generates Netlist at Gate level. The Compiler Converts high level VHDL Code in RTL to Gate level.

3) This Netlist is further optimized to get optimized Netlist again at Gate Level. optimization is done for better Speed and less Space. The Simulation of design is done at this Stage.

4) finally a physical device is implemented on CPLD / FPGA, or final MASK is Prepared for ASIC from this optimize Netlist by place & route Software (fitter), once again the final device Can be Simulated & Verified.

This is the digital ckt design flow using VHDL. Now because VHDL is also one Kind of Programming language. it also

__/__/__

has its Program Structure (similar to other Programming language like C Program Structure) So as a Next step, let us learn what the VHDL program structure is ?

VHDL Program Structure →

* All the VHDL Program Consist of at least two Component. Entity & Architecture.
* it may have additional Components like Configuration, Package declaration, body, etc. as per requirements.

The Structure of the VHDL Program is like.

```
LIBRARY library name;
USE library name. Package name. Package parts;

ENTITY entity name is
  PORT ( PORT name : Port mode Port type;
         Port name : Port mode Port type;
         Port name : Port mode Port type;
           :
           :
         );
END entity name;

ARCHITECTURE archi name OR entity name is
   declarations
   BEGIN
```

__/__/__

Code (sequential or Concurrent Statements)

:

:

END [architecture name]

library declaration →

* The library Contains all the piece of code that is used frequently. it will allows us to reuse them again & again. Also, this Can be shared with other designs.
* it Starts with keyword LIBRARY followed by library name.

There are three libraries usually used in all VHDL Codes.

1) IEEE — Specifies multilevel logic system.
2) Std — resource library for VHDL design enviironment.
3) Work — used for Saving our Project work & Program file (.vhd)

* However, in Program Code, we need to declare only the IEEE library because the other two libraries are default libraries.
* Now to add library Package and its Part USE keyword is used with library name. library Package and Package parts.

for example in the IEEE library, the Package is Std_Logic_1164 and to add all its Part we can write.

```
LIBRARY ieee
    USE ieee.std_logic_1164.all;
```

So all VHDL Program start with above two statements for library declaration.

Entity declaration:

- Entity defines input-output connections of the digital circuit with which it can interact with other components / circuit.
- it declares the no. of input given to the ckt and the number of outputs taken out form the circuit.
- Also, it declares any intermediate signals that are used within the circuit itself.
- Entity declaration starts with the keyword ENTITY. The user has to give desire name to entity often related to a circuit that is being designed like 'mux', 'decoder', 'adder' 'Counter' etc. (the rule of thumb for any VHDL Program is the Program file name must be same as entity name)
- inside entity, input-output pins of a circuit are declared using keyword PORT.
- PORT (means interfacing pins) are declared with Port_name, Port_mode, and Port_type.

1) Port_name — it's a user-defined name of the input-output pin.

2) Port_mode — There are four types of port mode IN, OUT, INOUT and BUFFER. IN indicates as input pin, that can be read-only. OUT indicates as o/p pin and its write-only. Both these pins are uni-directional. INOUT pin is bidirectional that can be read as well as write. BUFFER is used for the intermediate o/p.

3) Port_type — it can be BIT, BIT_VECTOR, STD_LOGIC, etc.

* After declaring all interfaces, the entity declaration ends with keywords END followed by entity name.

* Let us see an entity example for two-input AND Gate.

```
ENTITY and_gate is
  PORT (a, b : IN BIT;
        y : OUT BIT);
End and_gate;
```

* Similarly, we can write an entity for half adder as

__/__/____

```
ENTITY half adder IS
  PORT (a, b : IN BIT;
        Sum, Carry : OUT BIT);
End half adder;
```

Architecture →

1) Architecture declares the funtionalities of the digital circuit.
2) it gives internal details of an entity that means how input - outputs are interConnected.
3) it describes behavior of the circuit means how the circuit generates required output from given inputs.
4) The architecture declaration Starts with keyword ARCHITECTURE followed by architecture - name and entity - name.
5) The BEGIN keyword indicates the Starting of the architecture body. The body includes Sequential or Concurrent Statements that describe circuit functionality.
6) The architecture body ends with keyword END followed by architecture - name.
7) Here is the architecture of 2 input AND gate (the entity is as given above)

```
ARCHITECTURE and gate arch. OF and
        gate is
BEGIN
    y <= a and b;
END and gate architecture;
```

__/__/__

★ Similarly, let us write architecture to half adder.

ARCHITECTURE half adder architecture of half adder is

```
BEGIN
    Sum <= a XOR b;
    Carry <= a and b;
END half adder architecture;
```

★ There are three different modelling styles for architecture body.

1) Data flow Style → in this modelling Style the circuit is described using Concurrent Statements.

2) Behavioral Style → in this modelling style the circuit is describe using Sequential Statements.

3) Structural Style → in this modelling style the circuit is described using different inter Connected Components.

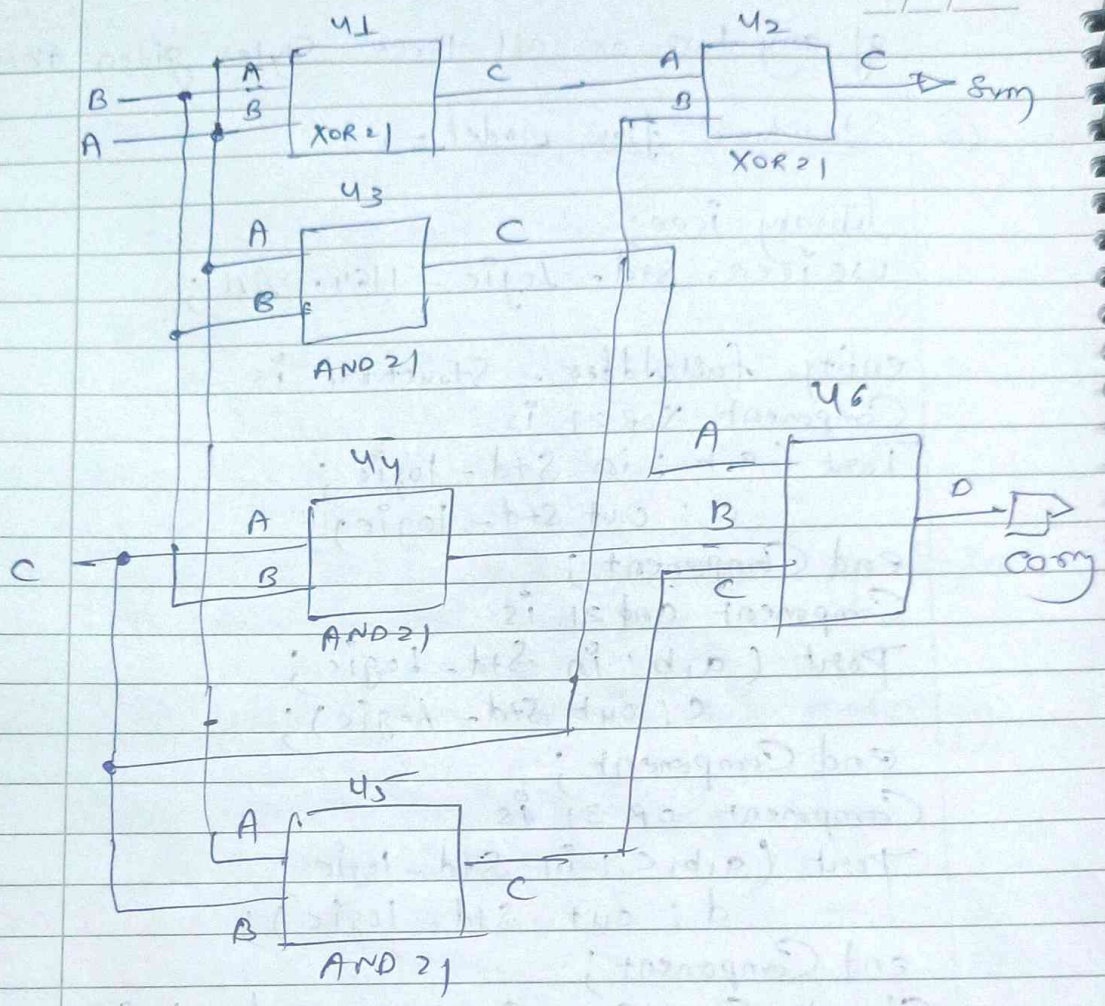
★ There Can be one more modelling style also that is mix modelling style - a Combination

of any two or all three styles given above.

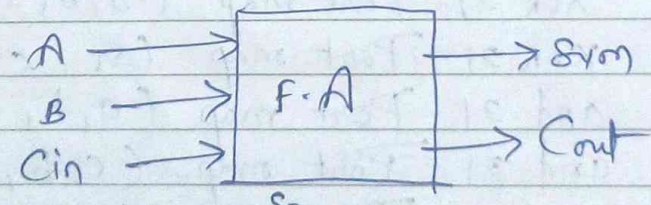
③ Structural flow model →

```
library ieee;
use ieee. std_logic_1164. All;

entity fullAdder_Structural is
Component XOR 21 is
Port (a,b : in Std_logic;
      c : out Std_logic);
end Component;
Component and 21 is
Port (a,b : in Std_logic;
      c : out Std_logic);
end Component;
Component OR 31 is
Port (a,b,c : in Std_logic;
      d : out Std_logic);
end Component;
Signal S1, C1, C2, C3 : Std_logic;
  Begin
U1 : XOR 21 Port map (a,b,S1);
U2 : XOR 21 Port map (S1, c, Sum);
U3 : and 21 Port map (a,b, C1);
U4 : and 21 Port map (c,b, C2);
U5 : and 21 Port map (a,c, C3);
U6 : OR 31 Port map (C1, C2, C3, Carry);
  end Behavioral;
```



Full Adder



$Sum = A \oplus B \oplus Cin$

$Cout = AB + B\,cin + A\,Cin$

S_1 S_2 S_3 S_4

internally generated signals are required.

```
Library IEEE;
use IEEE. Std_logic_1164. all;
entity fullAdder is
Port (A, B, Cin : in_ Std_logic;
  Sum, Cout ; out - Std - logic);
end entity;

Architecture Structural of fullAdder is
Component XOR 2 is
Port (A, B : in Std- Logic
        C : out Std- Logic);
End Component;
Component and2 is
Port (A, B : in Std-Logic ;
        C : Out Std - logic);
End Component;
Compment OR 2 is
Port (A, B : in Std- Logic ;
        C : out Std- Logic);
End Compment
Begin
  Signal S0, S1, S2, S3, S4 : bit;
X1 : XOR 2 Port map (A, B, S0);
X2 : XOR 2 Port map (S0, Cin, Sum);
A1 : And 2 Port map (A, B, S1);
A2 : And 2 Port map (B, Cin, S2);
A3 : And 2 Port map (A, Cin
O1 : OR 2 Port map ( ' ' (2 down to 0);
```

```
01 : or2 Port map (S4, S3, Cout);
```

2X4 Decoder

```
library IEEE;
use IEEE. STD_LOGIC_1164. ALL;

entity Decoder_2X4 is
    Port ( A,B : in STD_LOGIC;
           y0, y1, y2, y3 : in out STD_LOGIC);
end Decoder_2X4;

Begin
    y0 <= (not A) and (not B);
    y1 <= (not A) and B;
    y2 <= A and (not B);
    y3 <= A and B;
end Behavioral;
```

Behavioral modelling →

```
Architecture Behavioral of Decoder_
        2X4_ Behavioral is
begin
    P1 : Process (A,B)
  begin
if (A = '0' and B = '0') then
    y0 <= '1';
y1 <= '0'; y2 <= '0'; y3 <= '0';
elsif (A = '0' and B = '1') then
```