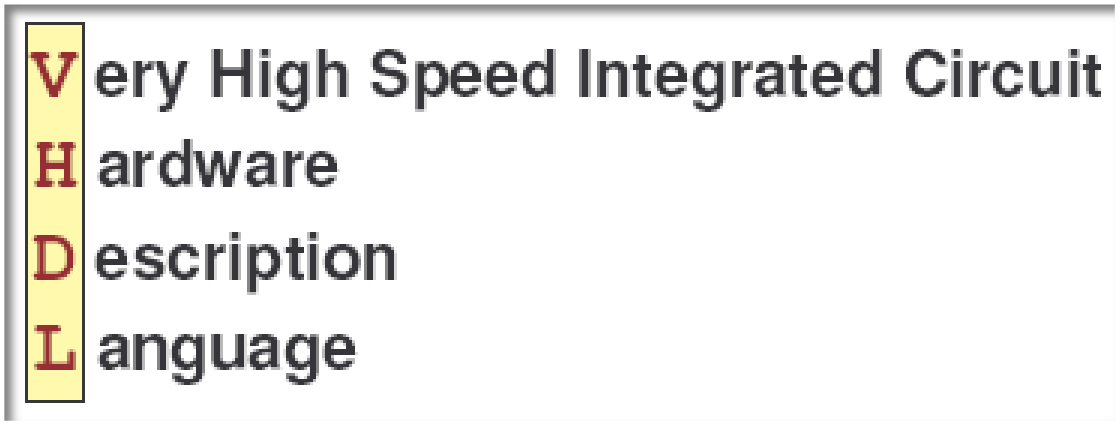
Introduction to VHDL

VHDL

- **VHDL is a language that is used to describe the behavior of digital circuit designs.**



– **Language to describe hardware.**

- **VHDL designs can be simulated and translated into a form suitable for hardware implementation.**

History of VHDL

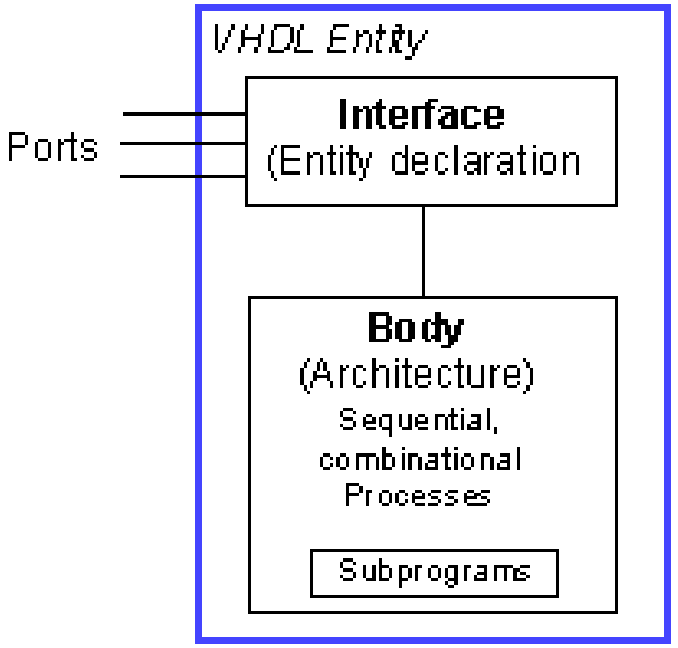
- Developed by Department of Defense (DoD) between 1970s and 80s, it was officially standardized as IEEE 1076 in 1987.
- IEEE 1164 is the latest standardization that bring about the interoperability between the common packages used by EDA venders.
- VHDL is now used extensively by industry and academia for the purpose of simulating and synthesizing digital circuit designs.

Basic Structure of a VHDL File

- Structural Elements of VHDL
  - Entity
  - Architecture
  - Signals

Basic Structure of a VHDL File

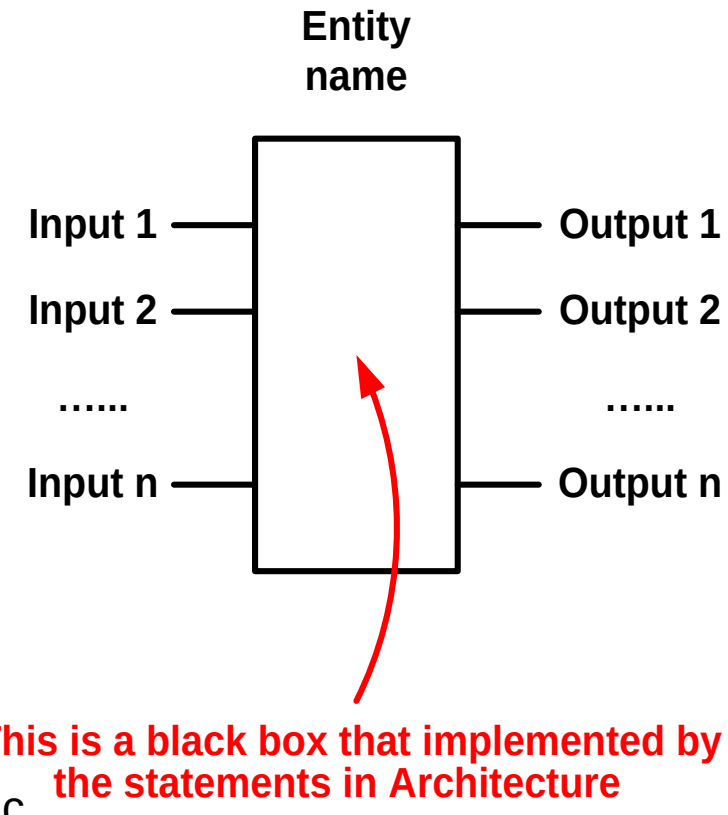
- Entity:-A hardware abstraction of the digital system is called an *entity.*
- An entity is modeled using an entity declaration and at least one architecture body.
  - Entity declaration: interface to outside world; defines input and output signals
  - Architecture: describes the entity, contains processes, components operating concurrently



Entity Declaration

- Interface for communication among different modules / components
- Define inputs and outputs

entity NAME_OF_ENTITY **is**
    port (*signal names*: **mode** *type*;
        signal_names: **mode** *type*;
            :
        signal_names: **mode** *type*);
end [NAME_OF_ENTITY] ;



- NAME_OF_ENTITY: user defined
- signal_names: list of signals (both input and output)
- mode: in, out, buffer, inout
- type: boolean, integer, character, bit, std_logic

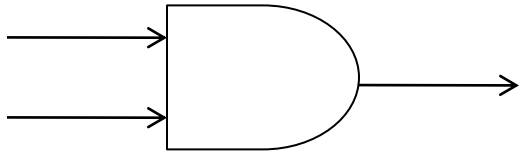
- **Type**: a built-in or user-efined signal type. Examples:
  - **bit** – can have the value 0 and 1.
  - **bit_vector** – is a vector of bit values (e.g. bit_vector (0 to 7)
  - **std_logic, std_ulogic, std_logic_vector, std_ulogic_vector**: can have 9 values to indicate the value and strength of a signal. Std_ulogic and std_logic are preferred over the bit or bit_vector types.
  - **boolean** – can have the value TRUE and FALSE.
  - **integer** – can have a range of integer values.
  - **real** – can have a range of real values.
  - **character** – any printing character.
  - **time** – to indicate time.

MVL - 9			
Uninitialized	'U'	Weak 1	'H'
Don't Care	'-'	Weak 0	'L'
Forcing 1	'1'	Weak Unknown	'W'
Forcing 0	'0'	High Impedance	'Z'
Forcing Unknown	'X'		

Entity Declaration

```
Entity and1 is
Port( a,b: in
   std_logic;
        c: out
   std_logic);
End and1;
```

and gate



B. Architecture body:

- The architecture body specifies how the circuit operates and how it is implemented.
- An entity or circuit can be specified in a variety of ways, such as **behavioral**, **structural** (interconnected components), or **dataflow**.
- The architecture body looks as follows:

```
ARCHITECTURE architecture_name OF NAME_OF_ENTITY IS
        -- Declarations
        BEGIN
        -- Statements
END architecture_name;
```

```
ARCHITECTURE dataflow OF and1 IS
        BEGIN
                c <= a AND b;
END dataflow;
```

Definitions of the Description Methods

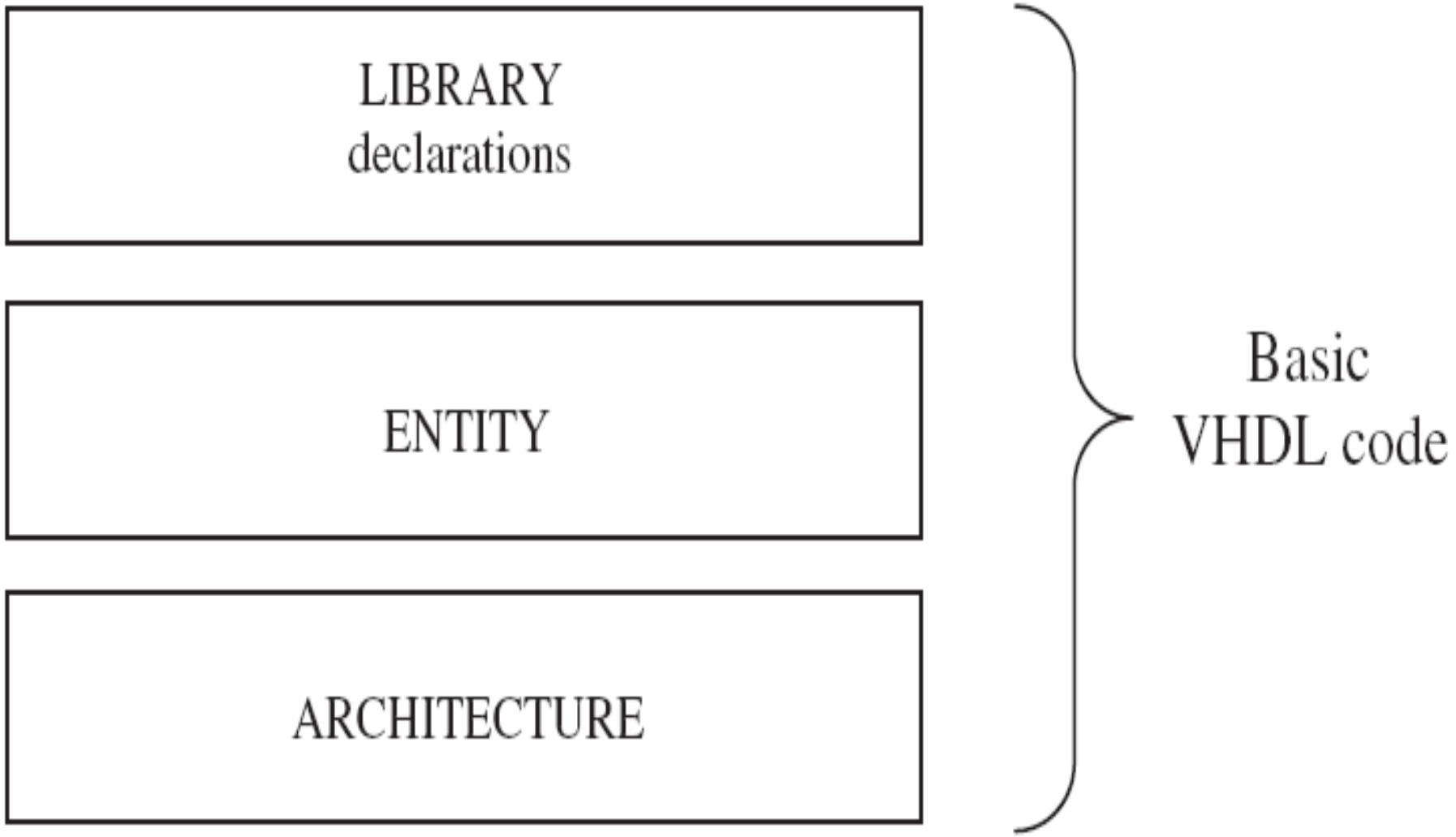
Data-Flow Description Method: It is similar to a register-transfer language.

- This method describes the function of a design by defining the flow of information from one input or register to another register or output .

- *Behavioral Description Method:*
  - The hardware behavior is described algorithmically

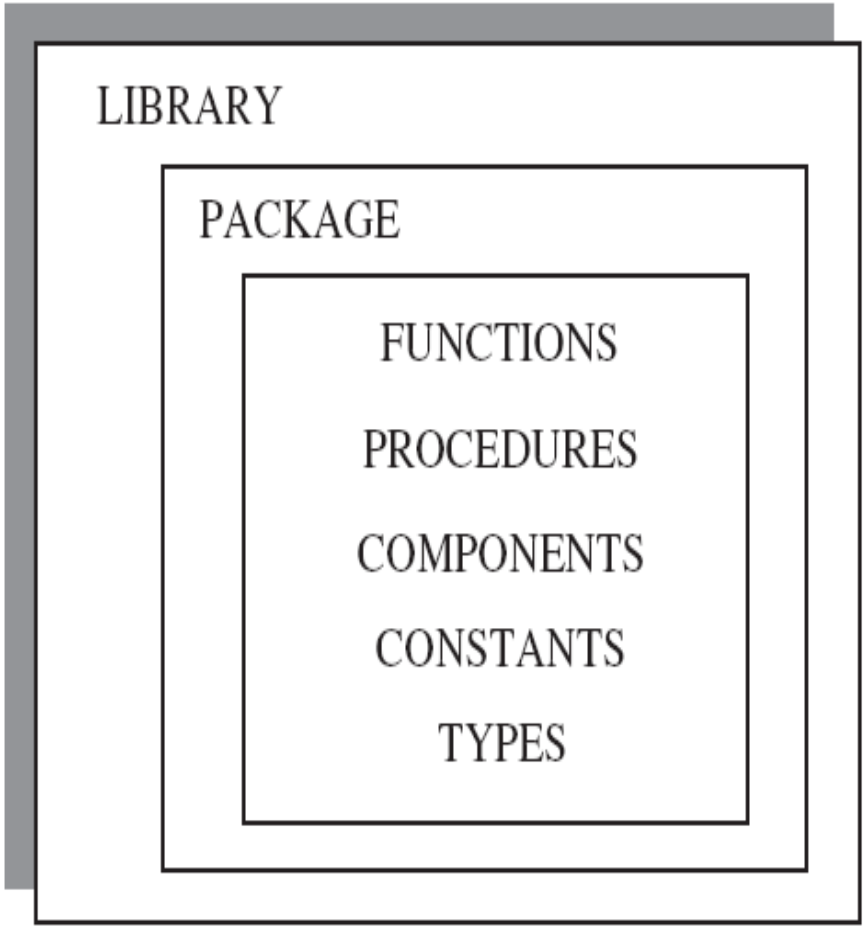
- *Structural Description Method:* expresses the design as an arrangement of interconnected components.
  - It is basically schematic

Fundamental sections of a basic VHDL code.



LIBRARY

- A LIBRARY is a collection of commonly used pieces of code. Placing such pieces inside a library allows them to be reused or shared by other designs.
- The typical structure of a library is shown aside.



Library Declarations

- To declare a LIBRARY (that is, to make it visible to the design) two lines of code are needed, one containing the name of the library, and the other a use clause.

The syntax is as follows

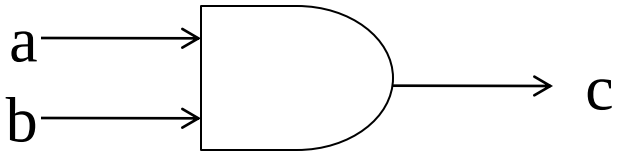
- LIBRARY library_name ;
- USE library_name.package_name.package_parts ;

Library Declarations

LIBRARY ieee; -- A semi-colon (;) indicates the end of a statement or a declaration

- USE ieee.std_logic_1164.all ;
- LIBRARY std ; - a double dash (--) indicates a
- comment.
- USE std . Standard . all ;
- LIBRARY work ;
- USE work. all;

and gate

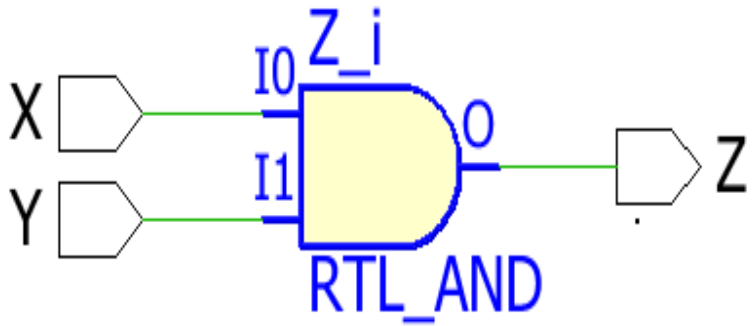


VHDL code:

```
entity and_dataflow is
    Port ( X : in STD_LOGIC;
           Y : in STD_LOGIC;
           Z : out STD_LOGIC);
end and_dataflow;
architecture Dataflow of and_dataflow is
begin
    Z <= X and Y;
end Dataflow;
```

AND Gate

- RTL Diagram



TBW Code:

```
entity and_d_tb is

--    Port ( );

end and_d_tb;

architecture Behavioral of and_d_tb is

component and_dataflow is

    Port ( X : in STD_LOGIC;

           Y : in STD_LOGIC;

           Z : out STD_LOGIC);

end component;

signal X1 : STD_LOGIC := '0';

signal Y1 : STD_LOGIC := '0';

signal Z1 : STD_LOGIC;
```

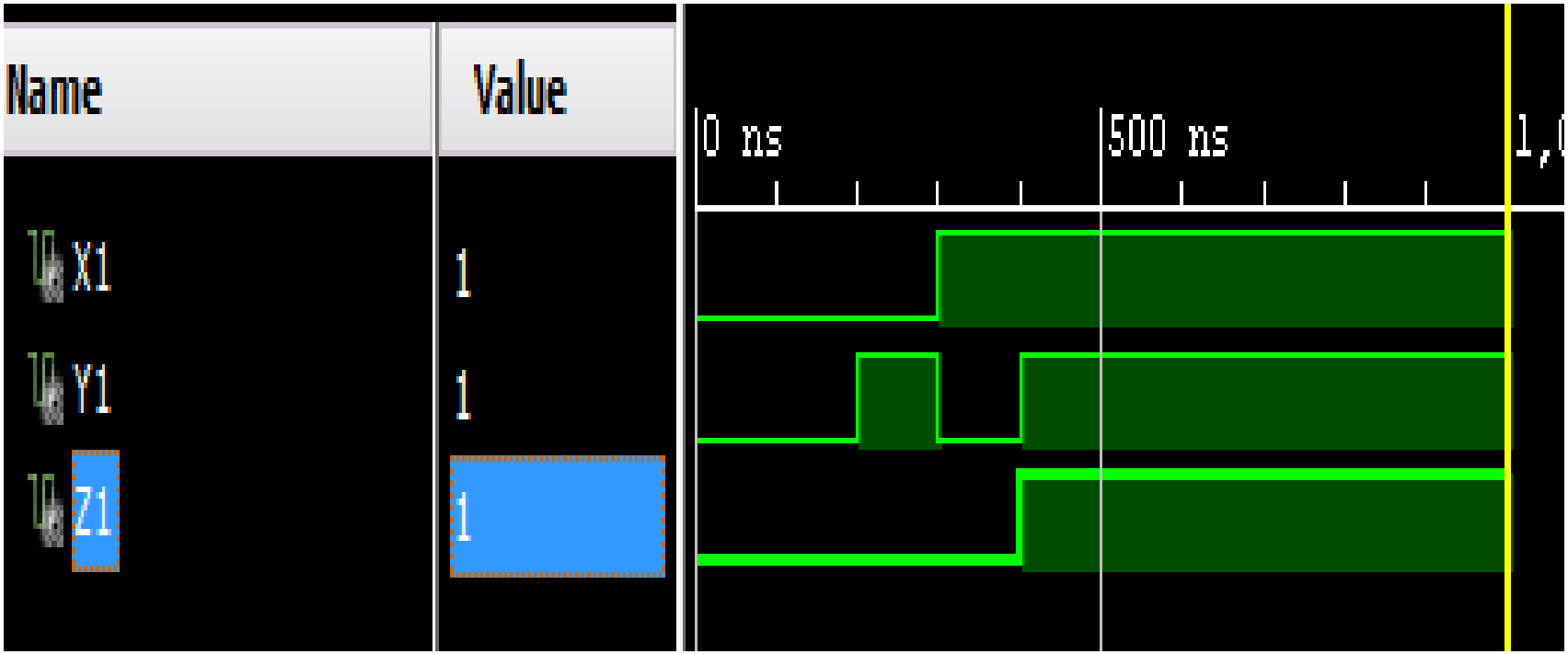
```
begin
uut: and_dataflow port map (X=>X1, Y=>Y1, Z=>Z1);
stim_proc: process
begin
wait for 100ns;
X1 <= '0';
Y1 <= '0';
wait for 100ns;
X1 <= '0';
Y1 <= '1';
wait for 100ns;
X1 <= '1';
Y1 <= '0';
wait for 100ns;
X1 <= '1';
Y1 <= '1';
wait;
end process;

end Behavioral;
```

AND Logic		
A	**B**	**Y = A • B**
0	0	0
0	1	0
1	0	0
1	1	1

Table 1: Truth table of ANDgate

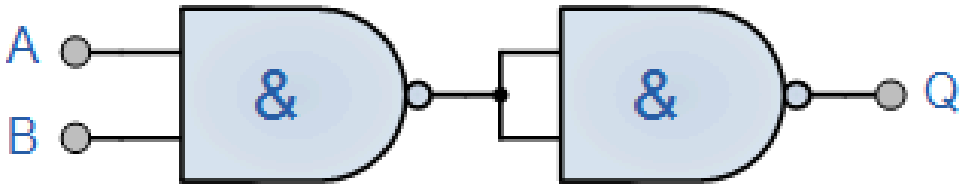
Waveform:



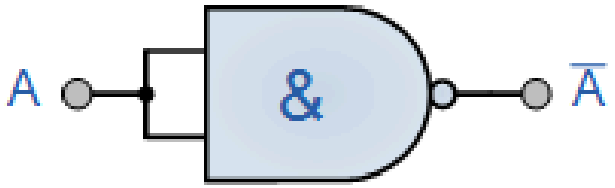
Assignments-1 Day 1

- Design Data Flow models of following logic gates.
- AND,OR,NAND,NOR,XOR,NOT
- Simulation of AND, NOT, OR gate using universal gates(NAND and NOR)

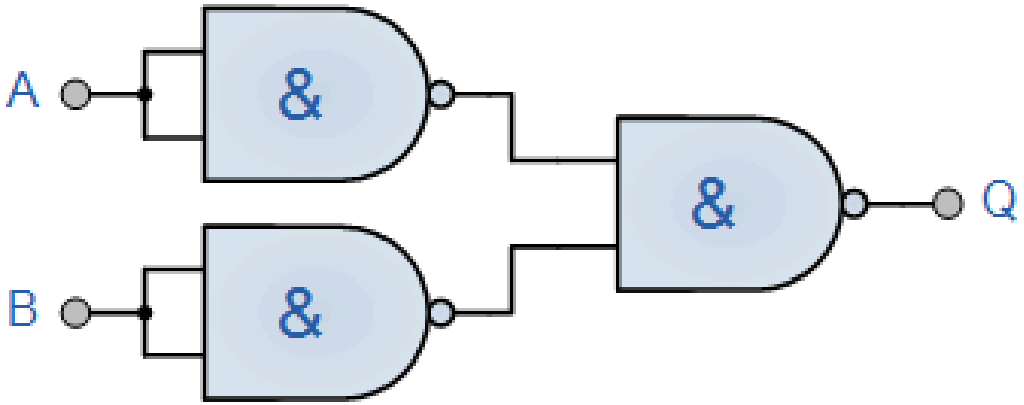
AND,NOT , OR using NAND



2-input AND Gate



NOT Gate

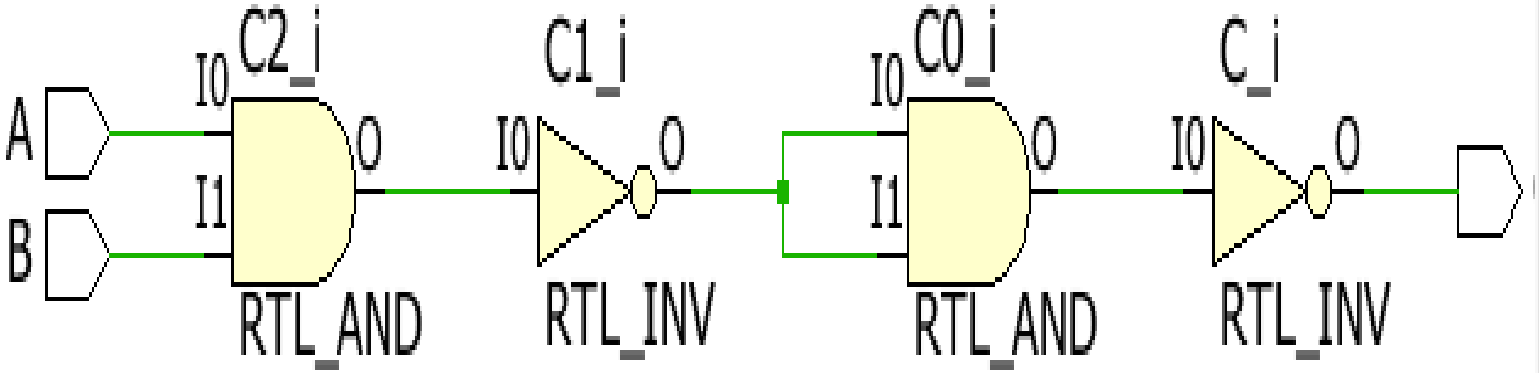


2-input OR Gate

AND using NAND

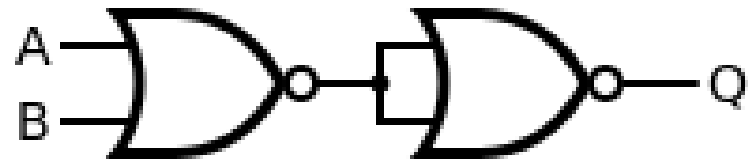
```
entity and_using_nand_dataflow is
    Port ( A : in STD_LOGIC;
           B : in STD_LOGIC;
           C : out STD_LOGIC);
end and_using_nand_dataflow;
architecture Dataflow of and_using_nand_dataflow is
 begin
    C <= (A nand B) nand (A nand B);
end Dataflow;
```

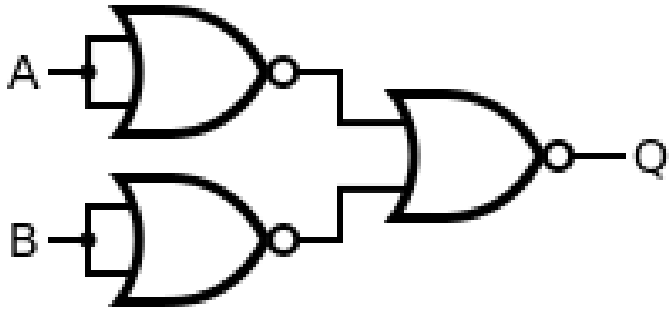
RTL Diagram:


A
B
I0
I1
C2_i
O
RTL_AND
C1_i
I0
O
RTL_INV
I0
I1
C0_i
O
RTL_AND
C_i
I0
O
RTL_INV

OR, AND,NOT Using NOR

A OR **B** = (**A** NOR **B**) NOR (**A** NOR **B**)





A AND B= (**A** NOR **A**) NOR (**B** NOR **B**)