INTRODUCTION TO VHDL

Lecture 5 & 6

Dr. Tayab Din Memon
Assistant Professor
Department of Electronic Engineering, MUET

VHDL Resources

- Other Sources
  - manufacturers web pages
    - http://www.xilinx.com
    - http://www.altera.com
    - http://www.vantis.com/
    - http://www.actel.com
    - plus LOTS of others
  - IP
    - www.opencores.org
    - http://www.fpgacpu.org/
  - Discussion groups
    - http://groups.google.com/group/comp.lang.vhdl/topics
  - VHDL International
    - http://www.vhdl.org/

Introduction to VHDL

- General introduction to basic language structure of VHDL
- Entities and Architecture
- Using VHDL to create a HDL “program”

What is VHDL?

- VHDL documentation contains behavioral and structural descriptions of an electronic system, subsystem, or device. The primary purpose of these data items is to document hardware designs in a machine processable, simulatable, and hierarchical format.
- Purpose – Specifying, modeling, designing, and simulation digital systems
- Advantage – Test Bench flexibility
- Standardization of VHDL urged to create synthesizer tools that reduces design time

Why VHDL?

- Simple PLDs of fewer than 500 gates are implemented by using Karnaugh map equations
- But Karnaugh Map (KM) equations are not suitable for larger circuits
- Schematic capture offers several advantages over KM approach
- But again this technique has many disadvantages as well
  - Control logic must still be implemented using traditional
- Needs a design methodology that increases the efficiency of designers

Disadvantages of Schematic

- Control logic must still be generated using traditional design techniques
- Schematics can be difficult to maintain because the intent of the design is often clouded by its implementation
- Schematic capture environments are proprietary, so a designer who works in a schematic capture environment for one project may not be able to reuse material when working one a new project that requires the use of a new schematic capture environment
- The simulation environment supported by the PLD schematic capture tool may not fit with the system design environment, making design verification difficult at best.

Why VHDL?

- VHDL is the language that fulfils the requirement like:
  - a standard medium for interchanging digital design information including a consistent method of providing communications during the procurement cycle.
  - a standard pathway for re-implementing electronics parts
  - a standard approach to determining which commercial off-the-shelf parts to use
  - a portable, reusable pool of hardware description language synthesis and simulation models for digital systems
  - a replacement pathway for obsolete components.

VHDL strengths and limitations

- Strengths:
  - Power and Flexibility – supports design libraries and the creation of reusable components. It can be used for design and simulation
  - Device-Independent Design – doesn't matter about device architecture and allows multiple styles of design
  - Portability – simulation before synthesize saves time
  - Benchmarking Capabilities – can be used with different device architectures and synthesize tools to evaluate results and choose the device that is the best
  - ASIC Migration – VHDL facilitates the development of AISC after Complex PLD (CPLD) or FPGA testing
  - Quick Time-to-Market and Low Cost – Saves time and earn more revenue

VHDL strengths and limitations

- Limitations:
  - verbose representation
  - does not always produce optimal (or even synthesizable) implementation
  - inefficient code can result in unneeded, repetitive, or sub-optimal logic.

VHDL Brief History /Standards

- VHDL stands for – Very High-Speed Integrated Circuits (VHSIC) Hardware Description Language
- Initially, VHSIC chips were developed for Department of Defence (DoD) USA
- Description was restricted to gate level design tools
- Large scale design description was needed
- Three companies IBM, Texas Instruments, and Intermetrics worked together for standardization
- These companies worked together to bring a standard called version 7.2, in 1985

History of VHDL

- IEEE 1076
  - VHDL first publicly available in 1985
  - adopted by IEEE in 1987 as IEEE1076-1987
  - enhanced version released as IEEE 1076-1993
- IEEE 1164
  - Improved portability
  - defines a standard package defining ***std_logic***
  - handles signal strengths, unknowns and high-Z

VHDL Standards - Extensions

- IEEE 1076.3
  - Numeric or Synthesis Standard
  - defines VHDL data types as they relate to actual hardware
  - Packages defined are: NUMERIC_BIT and NUMERIC_STD
  - NUMERIC_BIT – bit type vectors
  - NUMERIC_STD
    - vectors with elements that are type std_logic
    - Defines two arithmetic data types, unsigned and signed, along with arithmetic, shift, type conversation, and logical operators
    - Functions in the packages perform arithmetic operations on unsigned and signed types, and return these types.
  - Replaces many non-standard vendor packages

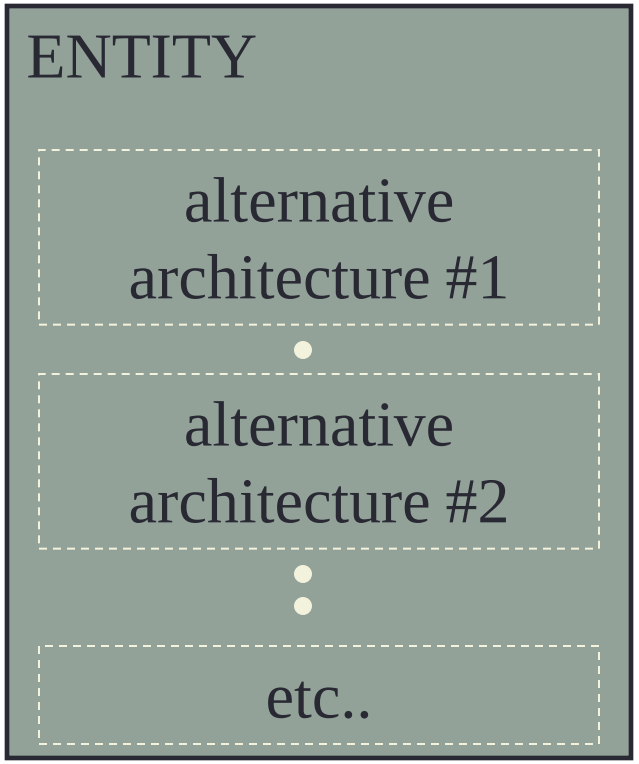
VHDL Structural Elements

- Main units in VHDL:
  - Entity
  - Architecture
  - Configuration
  - Package

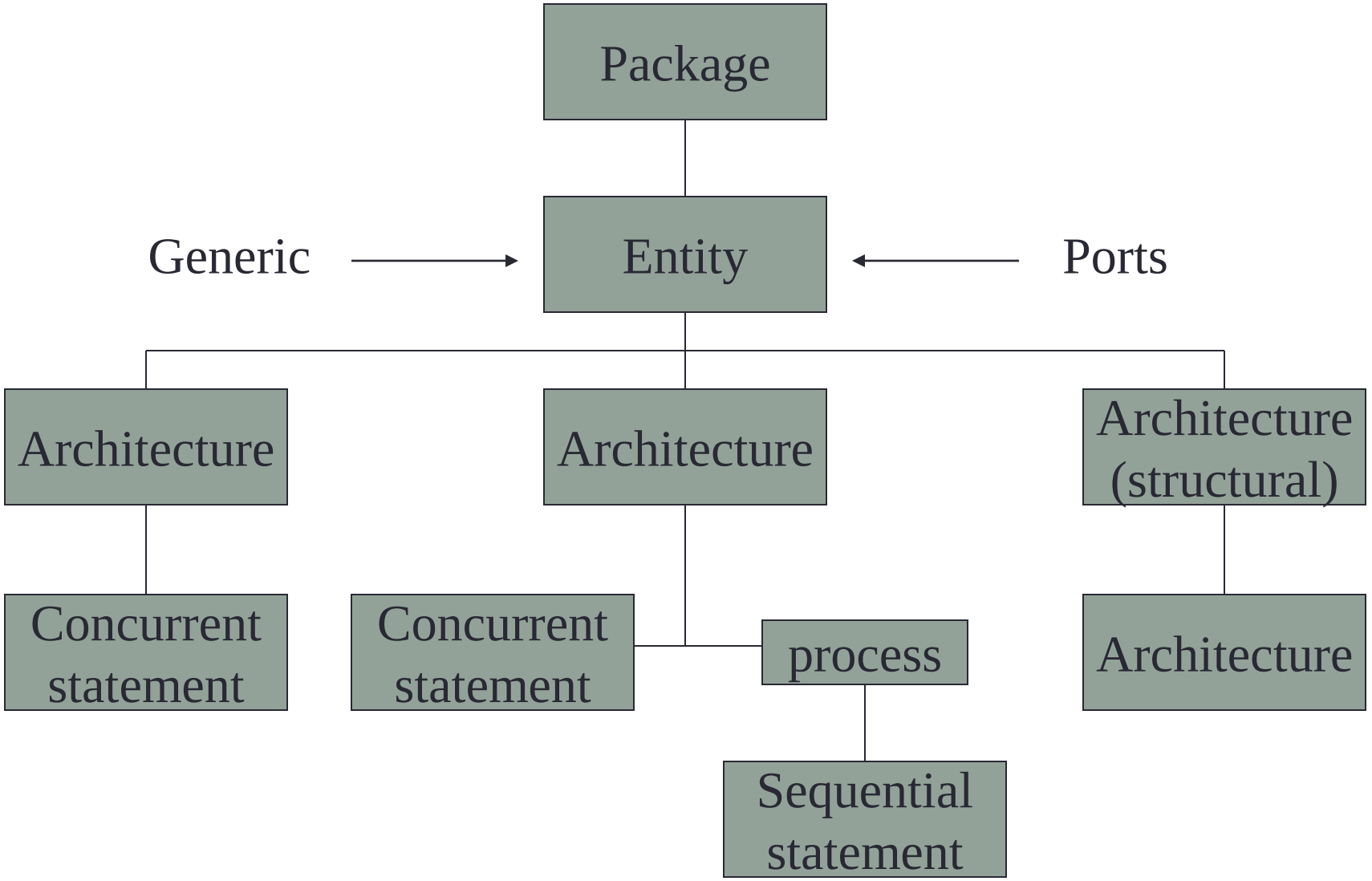
VHDL Entities and Architecture

- Every VHDL design description has one Entity/Architecture pair.

- *Entity*
  - describes circuit as it appears from outside
- *Architecture*
  - describes function *(contents)* of entity
  - can be numerous alternative arch's



VHDL Entities and Architecture



Entity Declaration

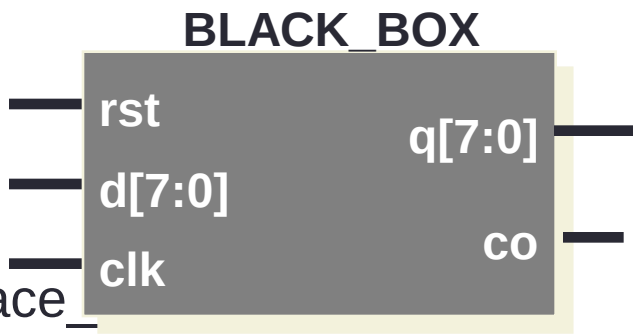
- Provides complete interface for circuit
  - defines I/O for connection and verification

syntax:

entity *identifier* **is**

port (port_interface_

end *identifier* ;



Example Entity declaration

```
-- eight bit comparator
entity compare is
        port (
            A, B : in  bit_vector ( 7 downto 0 );
            EQ  : out bit );
end compare;
```

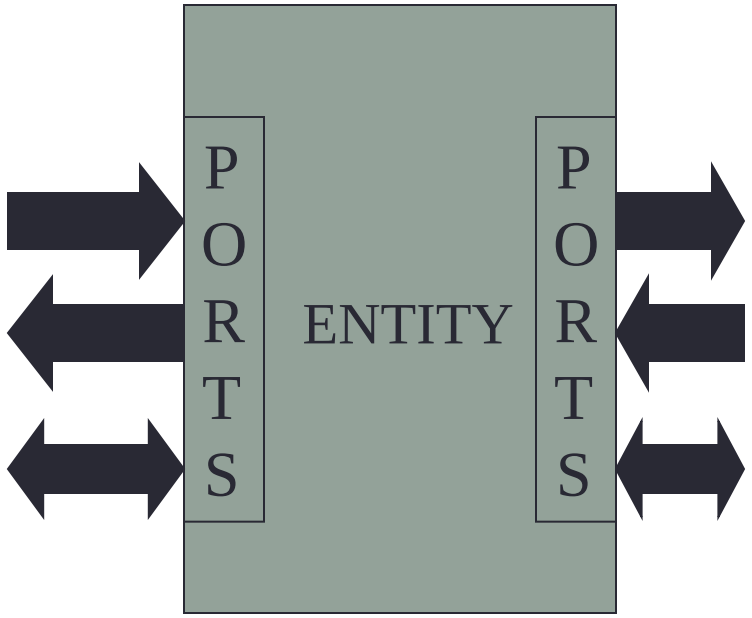
Entity name

List of inputs and outputs

Port types

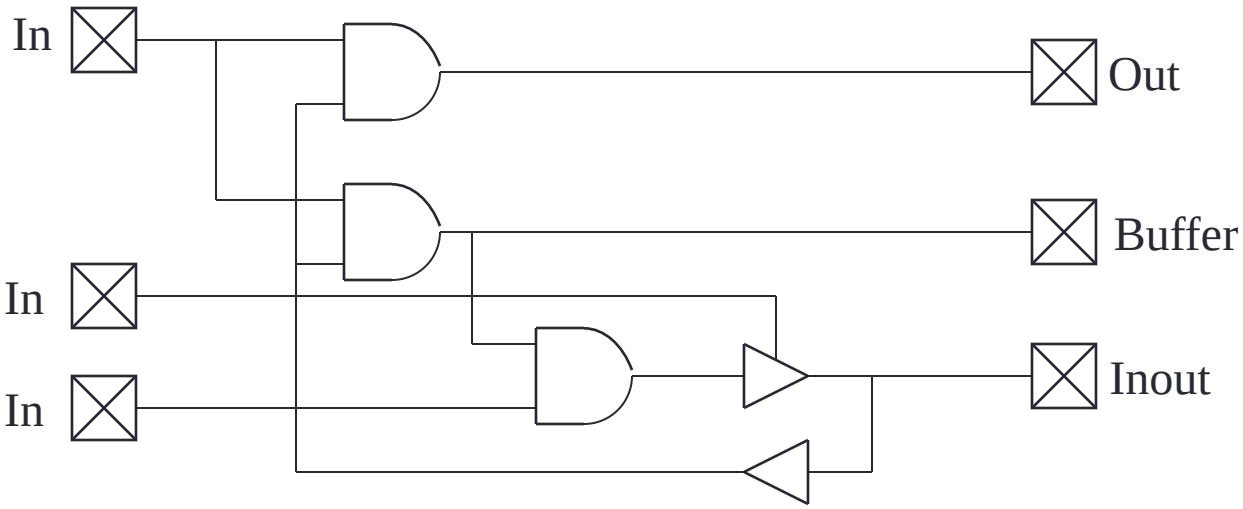
Basic VHDL: Ports

- Used to define inputs and outputs ports of an entity
- Each port defined by:
  - name
  - direction (mode)
  - data type



Port Modes

- Describes direction of data transfer
  - Port **in** : data flows only into circuit
  - Port **out** : data flows only out of circuit
  - Port **buffer** : for internal feedback or driver ***NOT*** bi-directional
  - Port **inout** : bi-directional signal, allows internal feedback



Architecture declaration

- Describes ***how*** circuit is implemented
- Describes the internal operation of a module
  - Every entity must have at least one architecture
  - Declared by the predefined word '***architecture***' followed by user selected name; this name follows the same name-selecting as for entity

    syntax:

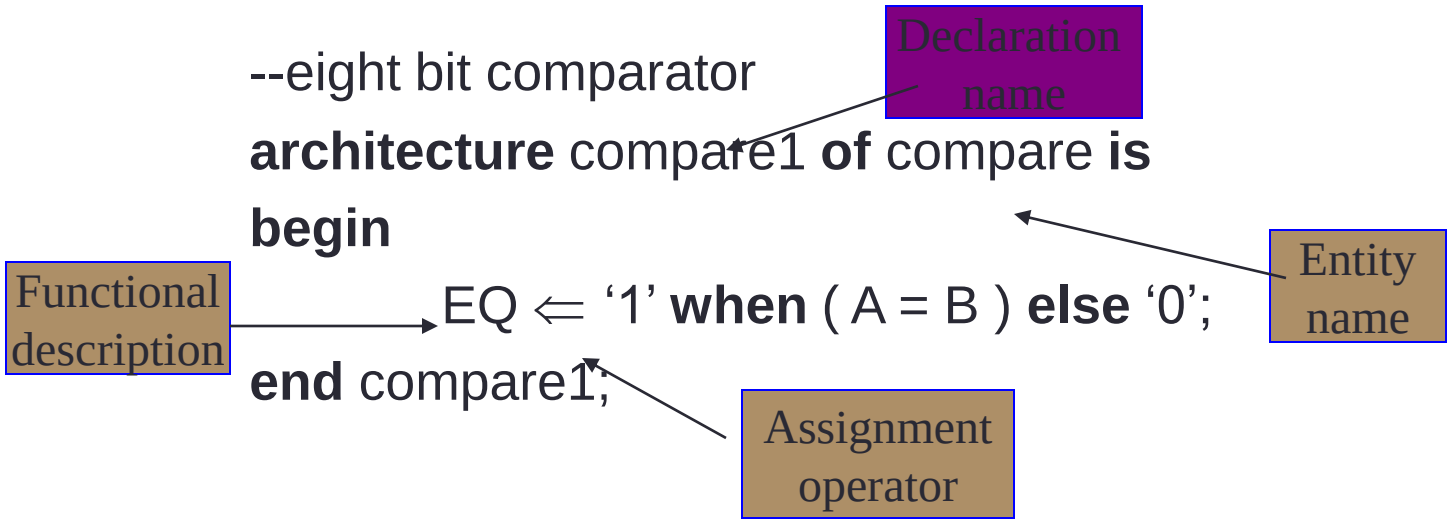
    architecture *identifier* **of** entity_name **is**

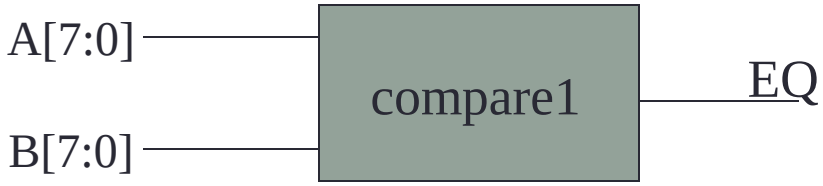
    begin

    [statements];

    end *identifier ;*

Example Architecture declaration





Example Entity/Architecture

```
-- eight bit comparator
entity compare is
port (    A, B : in  bit_vector ( 7 downto 0 );
          EQ  : out bit );
end compare;

--eight bit comparator
architecture compare1 of compare is
begin
  EQ ⇐ ‘1’ when ( A = B ) else ‘0’;
end compare1;
```

VHDL Hardware Models

- VHDL models three important facets of digital hardware
  - Behavior
    - A behavioral architecture uses only process statement
  - Structure
    - Structural architecture uses only component instantiation statements
  - Dataflow
    - A dataflow architecture uses only concurrent signal assignment statements
- VHDL combines all three facets of hardware description into a cohesive language

Behavioural Description

- Models the system as to how the outputs behave with the inputs
- Described in terms of High Level Language
- Process statement specifies the behaviour of the circuit, when executed in sequence
- Sensitivity list identifies which signals will cause the process to execute

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  entity behavtest is port
4      (x,y: in std_logic_vector(3 downto 0);
5       z: out std_logic);
6  end behavtest;
7  architecture behavioral of behavtest is
8  begin
9  behtst: process (x,y)
10       begin
11           if x=y then
12               z <='1';
13             else
14               z<= '0';
15             end if;
16           end process;
17         end behavioral;
```

Dataflow Description

- Here architecture specifies how data flows through the system i.e. from signal to signal and input to output without the use of sequential statements

It posses one or more concurrent signal assignments (i.e., when-else or with-select-when) rather then sequential statements inside the process

```
library ieee;
use ieee.std_logic_1164.all;
entity dataflowtest is port
    (x,y: in std_logic_vector(3 downto 0);
     z: out std_logic);
end dataflowtest;
architecture dataflow of dataflowtest is
begin
  z <= '1' when (x=y) else '0';
end dataflow;
```

Structural Description

```
library ieee;
use ieee.std_logic_1164.all;
entity structtest is
  port (x,y: in std_logic_vector (2 downto 0);
          z: out std_logic);
        end structtest;

architecture structure of structtest is
signal t: std_logic_vector (2 downto 0);
component AndTest is
  port (a,b: in std_logic;
          c: out std_logic);
        end component;
component OrTest is
  port (a,b,c: in std_logic;
            d: out std_logic);
          end component;
begin
  and0: AndTest port map (x(0), y(0), t(0));
  and1: AndTest port map (x(1), y(1), t(1));
  and2: AndTest port map (x(2), y(2), t(2));
  Org0: OrTest  port map (t(0), t(1), t(2), z);
end structure;
```

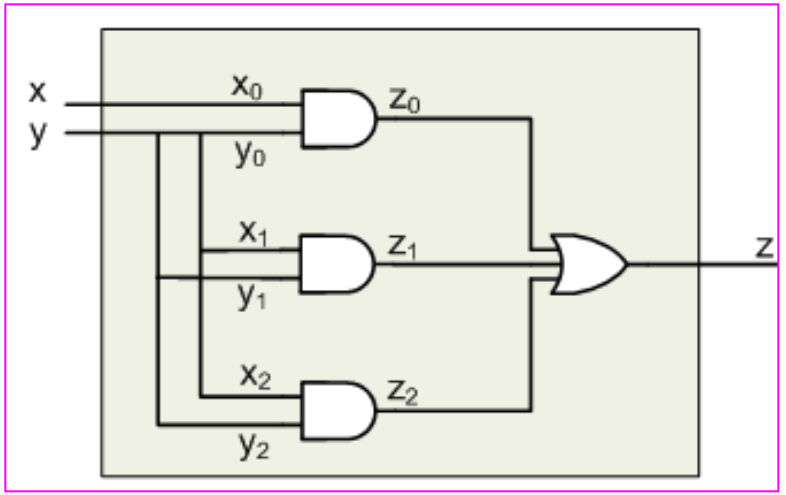


Fig: AND-OR Logic

Structural designs consists of VHDL netlists

Structural designs are hierarchical

Component design entities

- Components are instantiated and connected together with signals
- To instantiate a component is to place it in a hierarchical design

```
library ieee;
use ieee.std_logic_1164.all;
entity AndTest is
  port (a,b: in std_logic;
          c: out std_logic);
        end AndTest;
architecture dataflow of AndTest is
begin
c<=a and b;
end dataflow;

library ieee;
use ieee. std_logic_1164.all;
entity OrTest is
  port (a,b,c: in std_logic;
          d: out std_logic);
        end OrTest;
architecture dataflow of OrTest is
begin
d<=a or b or c; end dataflow;
```

END OF THE LECTURE 5 & 6

Thanks for your patience