Introduction to VHDL

Yvonne Avilés
Colaboration: Irvin Ortiz Flores
Rapid System Prototyping Laboratory (RASP)
University of Puerto Rico at Mayaguez

What is VHDL?

- **V**ery High Speed Integrated Circuit **H**ardware **D**escription **L**anguage
- A programming language designed and optimized for describing the behavior of digital systems.
- Allows for automatic circuit synthesis and system simulation.
- A standard in the electronic design community.
  - IEEE Standard 1076 defines the complete VHDL language
  - The IEEE 1076-1987 and IEEE 1164 standards together form the complete VHDL standard in widest use today

Entities and Architectures

- Every VHDL design description consists of at least one entity/architecture pair.
- **A VHDL *entity* is a statement that defines the external specification of a circuit or sub-circuit.**
- Provides the complete interface for a circuit
  - Names, data types, direction of ports

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

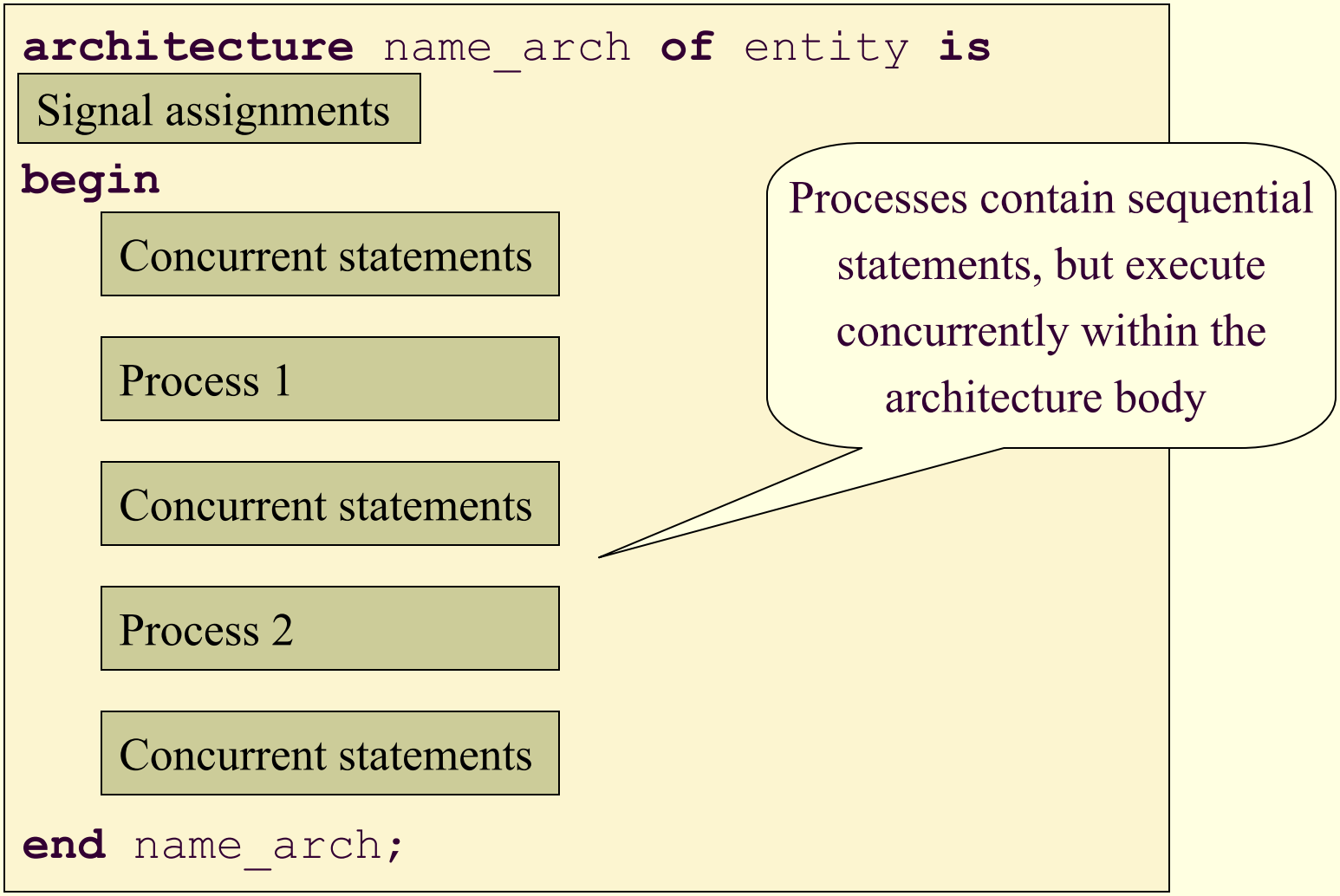
Architecture Declaration

- Every referenced **entity** in a VHDL design description must be bound with a corresponding architecture.
- The architecture describes the actual function—or contents—of the entity to which it is bound.

```
architecture compare1 of compare is
begin
                    EQ <= '1' when (A = B) else '0';
end compare1;
```

- Hierarchy and subprogram features of the language allow lower-level components, subroutines and functions in architectures; a *process* allows complex registered sequential logic as well.
- VHDL allows more than one alternate architecture for an entity.

VHDL Architecture Structure



VHDL Process Syntax

```
P1: process (<sensitivity list>)
<variable declarations>
begin
    <sequential statements>
end process P1;
```

- Within a process: Variables are assigned using := and are updated immediately. Signals are assigned using <= and are updated at the end of the process.

Signals Vs Variables

- Signals
  - Used to connect components or to carry information between processes
  - When inside of a process, its value is updated when the process suspends
  - Signal assignment operator: <=
- Variables
  - Local to a process
  - Not visible outside the process
  - Values are updated immediately after the assignment
  - Variable assignment operator: :=

Signals Vs Variables

- Signals
  - Initial values: A=5, X=10, B=15
  - Final values: A=10, B=5

```
Sigproc:
  process(A,X)
Begin
 A <= X;
 B <= A;
End process Sigproc;
```

- Variables
  - Initial values: X=10
  - Final values: A=10, B=10

```
Sigproc: process(X)
Variable A,B :
  integer;
Begin
 A := X;
 B := A;
End process Sigproc;
```

Data Types

- ***Like a high-level software programming language, data is represented in terms of high-level data types.***
- Every data type in VHDL has a defined set of values and valid operations.

Data Type	*Values*	*Example*
Bit	**‘1’,‘0’**	**Q<=‘1’;**
Bit_vector	**(array of bits)**	**DataOut <= "00010101";**
Boolean	**True, False**	**EQ <= True;**
Integer	**-2, -1, 0, 1, 2, 3, 4 . . .**	**Count <= Count + 2;**
Real	**1.0, -1.0E5**	**V1 = V2 / 5.3**
Time	**1 ua, 7 ns, 100 ps**	**Q <= ‘1’ after 6 ns;**
Character	**‘a’, ‘b’, ‘2, ‘$’, etc.**	**CharData <= ‘X’;**
String	**(Array of characters)**	**Msg <= "MEM: " & Addr**

Design Units

- Entity : Interface
- Architecture : Implementation, behavior, function
- Configuration : Model chaining, structure, hierarchy
- Process : Concurrency, event controlled
- Package : Modular design, standard solution, data types, constants
- Library : Compilation, object code

Levels of Abstraction

- Behavior
  - Relies on processes to implement sequential statements
- Dataflow
  - *Describe circuit in terms of how data moves through the system.*
  - *Also referred to as register transfer logic*, or RTL.
- Structure
  - Used to describe a circuit in terms of its components
  - Requires hierarchical constructs
- Mixed Method
  - Any combination of methods

Entity Declaration

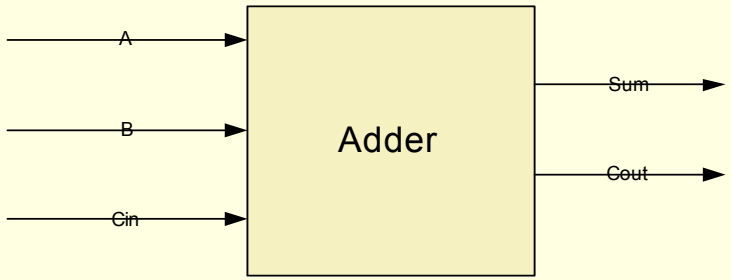
- Specifies the unit' s ports
- States the port's name type, mode
- Ports can be *in*, *out* or *inout*
- Port size can be from a bit to a bit vector

Entity name

Port length

```
Entity Adder is
 port (A,B :  in   std_logic(4 downto 0);
       Cin  : in   std_logic;
       Sum  : out  std_logic(4 downto 0);
       Cout : out  std_logic);
End Adder;
```

Port names Port mode Port type



An Architecture Example

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.STD_LOGIC_ARITH.all;
Entity Adder is
      port (A,B :  in   std_logic_vector(4 downto 0);
            Cin  : in   std_logic;
            Sum  : out  std_logic_vector(4 downto 0);
            Cout : out  std_logic);
End Adder;

architecture a_adder of adder is

signal AC,BC,SC : std_logic_vector(5 downto 0);

begin
     AC <='0' & A;
     BC <='0' & B;
     SC <= unsigned(AC) + unsigned(BC) + Cin;
     Cout <= SC(5);
     Sout <= SC(4 downto 0);
end a_adder;
```

Library declaration section

Architecture declaration

Associated entity

Signal declaration. Also can be placed component, constants, types, declarations.

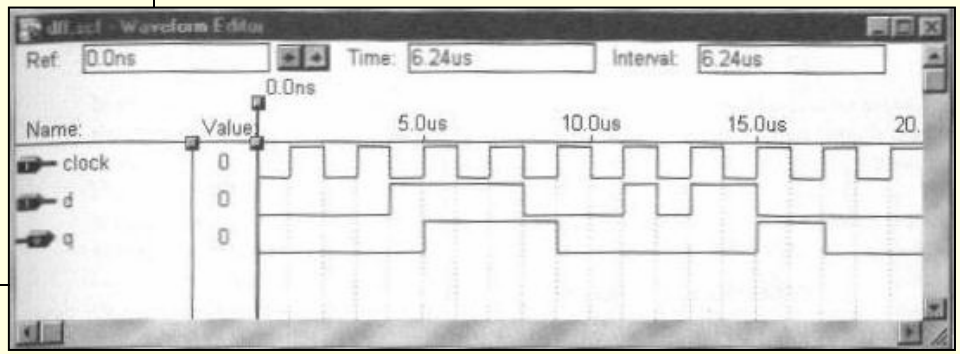
Concurrent Statements:Processed at the same time. Also component instantiations, and processes can be placed.

A “D” Flip-Flop

```
entity dff is
port ( d,clock : in bit;
       q: out bit);
end dff;

architecture arch of dff is
begin
    process (clock)
    begin
        if(clock'event and
    clock=1) then
            if(d='1') then
                q <= '1';
            else
                q <= '0';
            end if;
        end if;
    end process;
end arch;
```

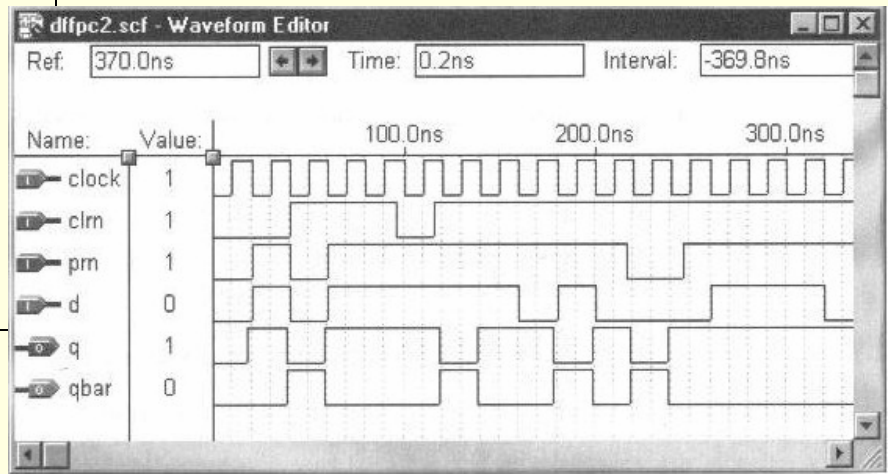
Refers to the rising edge of the clock

Explicit comparations and assigments to port and signals uses ‘ ’ for one bit and “ ” for multiple bits



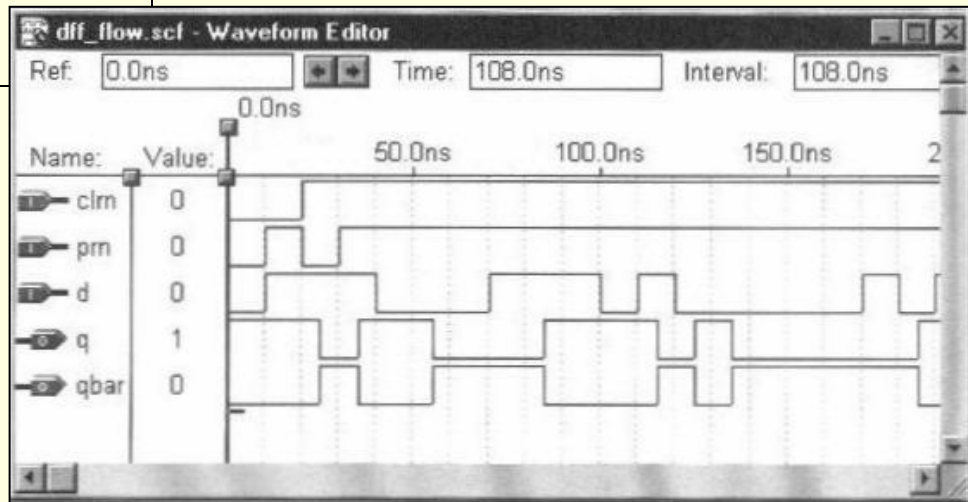
D Flip-Flop Behavioral

```
--Active low preset and clear inputs
entity dffpc2 is
   port(d,clock,clrn,prn:in bit;
        q,qbar:out bit;
end dffpc2;

architecture arch of dffpc2 is
begin
   process(clock,clrn,prn)
      begin
      if(clock'event and clock =
      '1') then
         q <= not prn or (clrn and
         d);
         qbar <= prn and (not clrn
         or not d);
      end if;
   end process;
end arch;
```



D Flip-Flop Dataflow

```
--D flip-flop dataflow
--Includes preset and clear
entity dff_flow is
   port (d, prn, clrn: in bit;
         q,qbar: out bit);
end dff_flow;

architecture archl of dff_flow is
begin
   q <= not prn or (clrn and d);
   qbar <= prn and (not clrn or not
   d);
end archl;
```



D Flip-Flop Structural

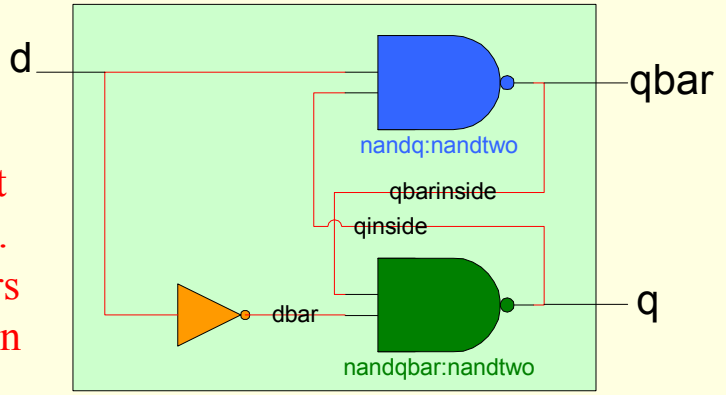
```
entity dff_str is
   port (d :in bit;
         q,qbar:out bit);
end dff_str;

architecture adff_str of dff_str is
component nandtwo
   port(x, y: in bit;
        z:out bit);
end component;
signal qbarinside, qinside, dbar: bit;
begin
   nandq:nandtwo
   port map(qinside, d,qbarinside);

   nandqbar:nandtwo
   port map(qbarinside,dbar, qinside);

   dbar <= not d;
   q <= qinside;
   qbar <= qbarinside;
end adff_str;
```

Component declaration. Port appears exactly as in the entity declaration.

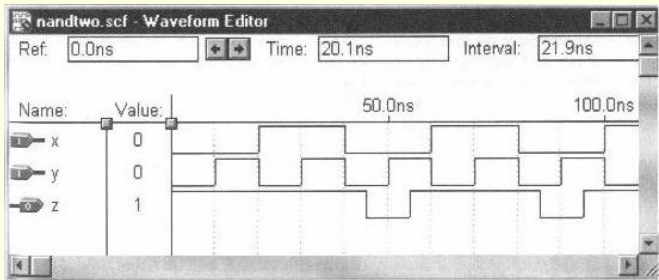
Entity name

Component instantiation label

Component instantiation. Conections are made by correspondence




```
--A two input nand gate
entity nandtwo is
   port(x, y:in bit;
        z :out bit);
end nandtwo;
architecture anandtwo of nandtwo
is
begin
   z <= not(x and y);
end anandtwo;
```



Three-Bit Binary Counter

```
entity countl is
   port( clock, enable: in bit;
         qa: out integer range 0 to 7);
end countl;

architecture countarch of countl is
begin
   process (clock)
   variable count: integer range 0 to 7;
   begin
      if (clock'event and clock ='1') then
            if enable = '1' then
               count:=count + 1;
            end if;
      end if;
      qa <= count after 10 ns;
   end process;
end countarch;
```

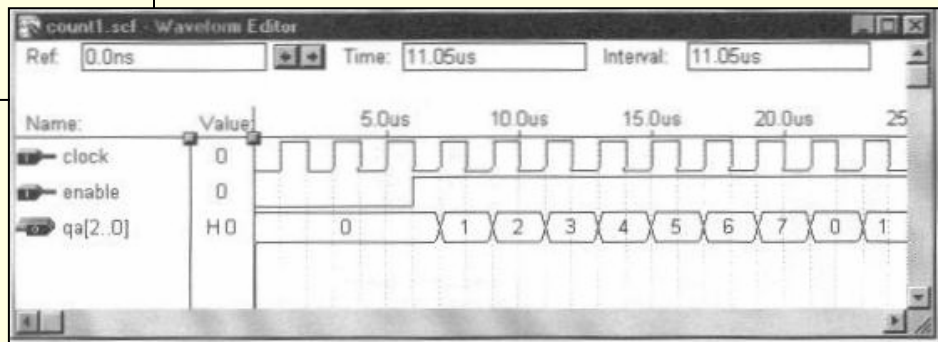
Sensitivity list. Process is executed everytime one of this parameters change.

Variable declaration

Variable assignment operator

Sequential statements.

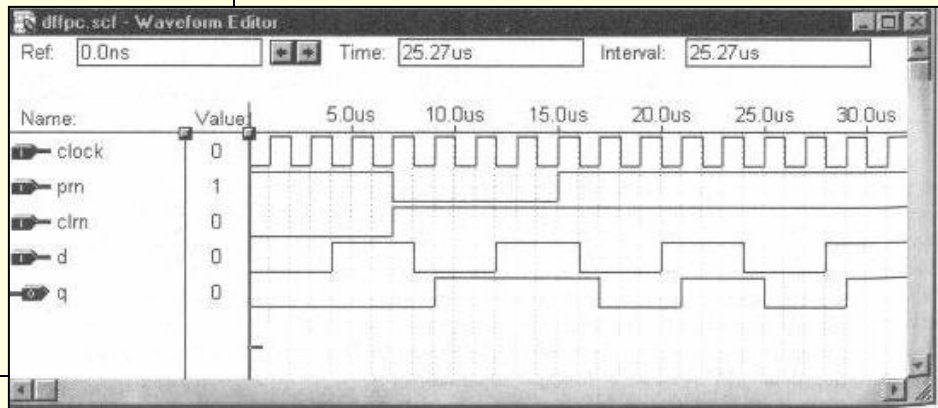
Signal assignment operator



D Flip-Flop with Asynchronous Preset and Clear

```
entity dffapc is
   port(clock, d, prn, clrn : in bit;
      q : out bit);
end dffapc;
architecture archl of dffapc is
begin
   process(clock, clrn, prn)
   variable reset, set: integer range 0 to
   1;
   begin
      if(prn='0') then
         q <= '1';
      elsif (clrn='0') then
         q <= '0';
      elsif (clock'event and
      clock='1') then
         q <= d;
      end if;
   end process;
end archl;
```

Integer range definition. *Range 0 to 1* defines one bit.

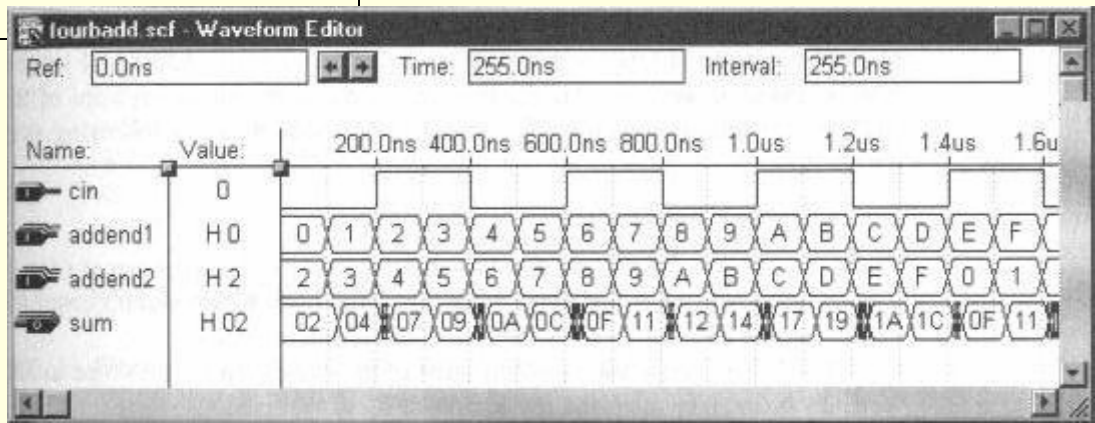


Four Bit Adder

```
--A VHDL 4 bit adder
entity fourbadd is
   port (cin: in integer range 0 to 1;
         addend1:in integer range 0
      to 15;
         addend2:in integer range 0
      to 15;
         sum: out integer range 0 to
      31);
end fourbadd;
architecture a4bitadd of fourbadd is
begin
    sum <= addend1 + addend2 + cin;
end a4bitadd;
```

Integer type allows addition, subtraction and multiplication. Need the following statement at the library declaration section:

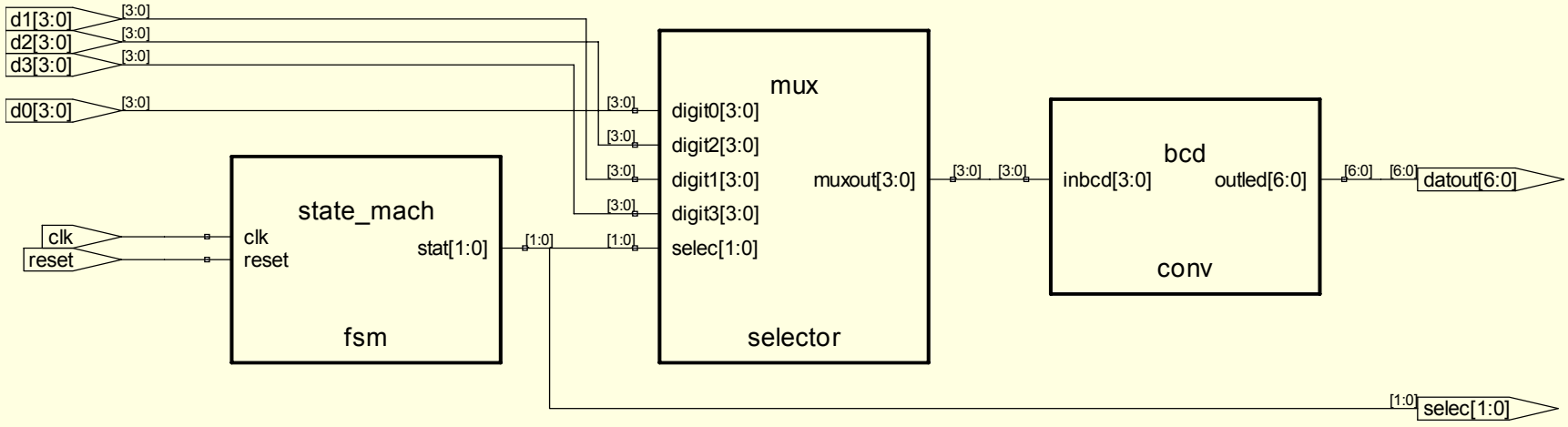
use IEEE.STD_LOGIC_ARITH.all



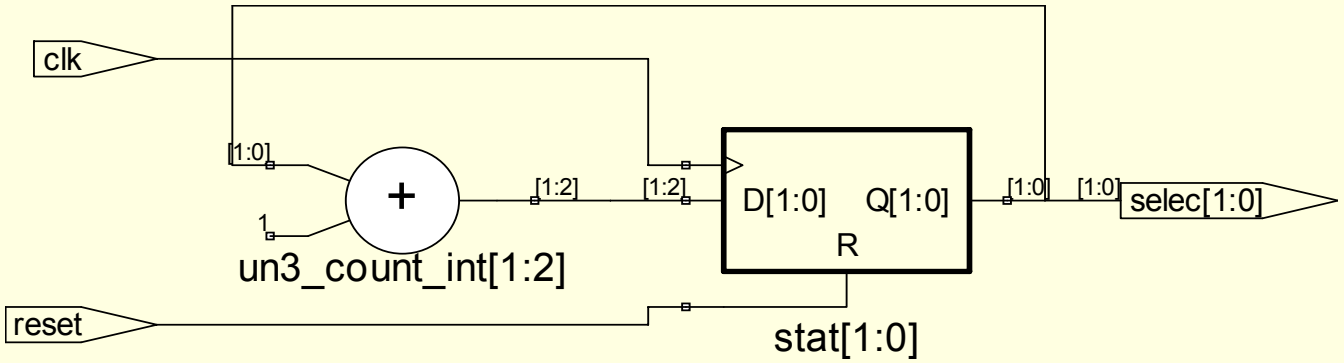
Synthesis

- Synthesis is a process of translating an abstract concept into a less-abstract form.
- The highest level of abstraction accepted by today's synthesis tools is the **dataflow** level.

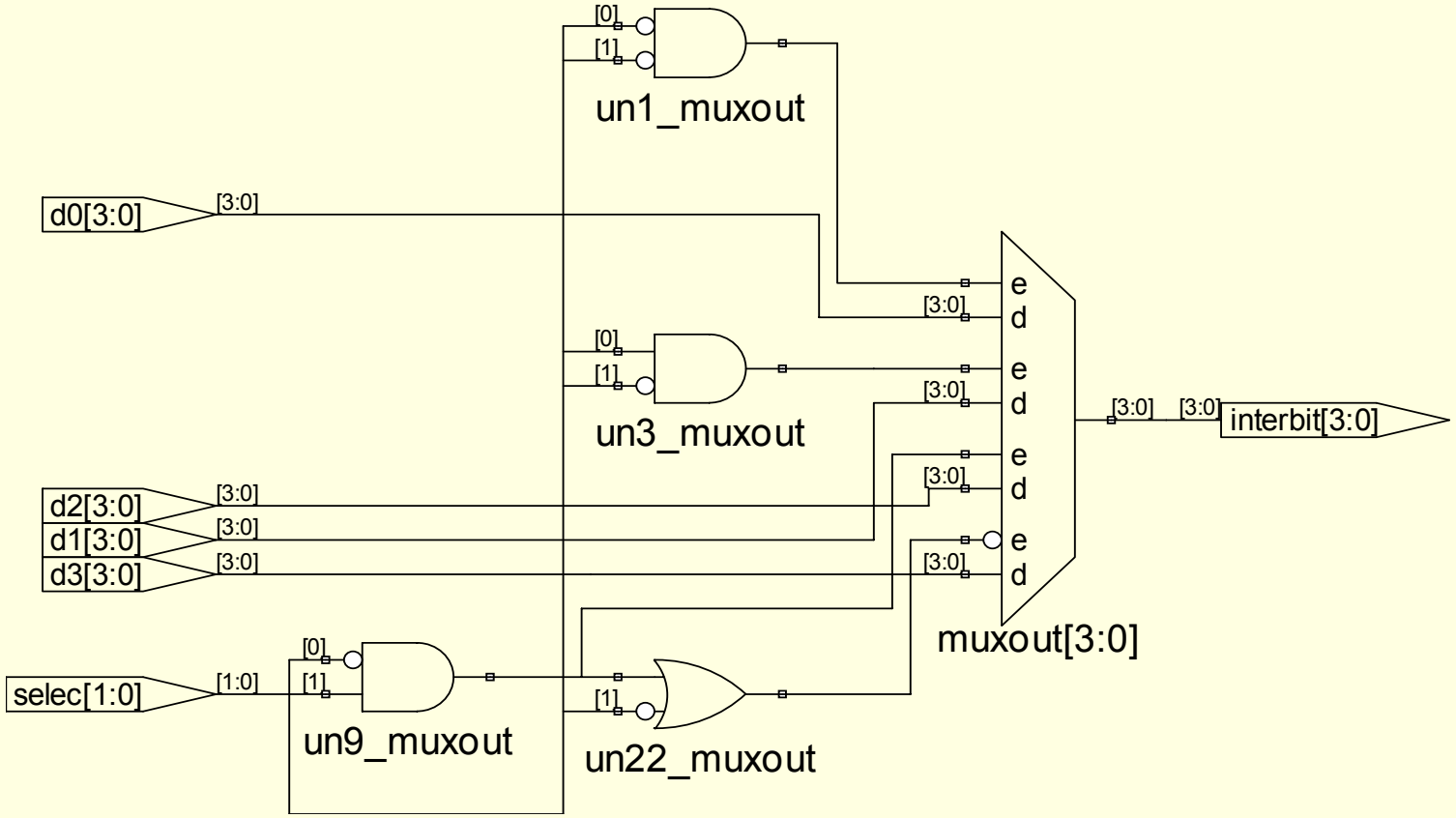
Display Driver



State Machine

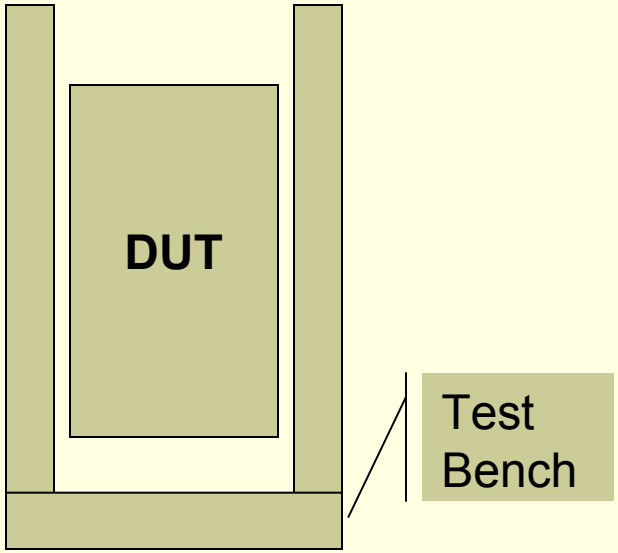


Data Selector



Test Benches

- VHDL can capture performance specification for a circuit, in the form of a test bench.
- Test benches are VHDL descriptions of circuit stimuli and corresponding expected outputs that verify the behavior of a circuit over time. Test benches should be an integral part of any VHDL project and should be created in tandem with other descriptions of the circuit.



Questions?

Link for VHDL Tutorial

- http://www.aldec.com/Downloads/