Course Name: Digital Hardware Design
Course Code: 17B1NEC741

VHDL-2

Dr. Arti Noor
Dean, Academic Affairs
Electronics and Communication Engineering,
Jaypee Institute of Information Technology, Noida

Types of Architectural declaration

The functionality of any system in VHDL can be modeled by the following methods:

- Behavioral Model
- Structural Model
- Physical/ Data Flow Model

Behavioral method:

- It describes a system's behavior or function in an algorithmic fashion. The most abstract style.
- It consists of one or more process statements. Each process statement is a single concurrent statement that itself contains one or more sequential statements.
- Sequential statements are executed sequentially by a simulator.

Structural method:

- In structural style of modelling, an entity is described as a set of interconnected components.
- The top-level design entity's architecture describes the interconnection of lower-level design entities and so on.
- This is most useful and efficient style when a complex system is described as an interconnection of moderately complex design entities.

Physical / Data Flow Model

- Dataflow style describes a system in terms of how data flows through the system.
- A dataflow description directly implies a corresponding gate-level implementation.
- It consist of one or more concurrent signal assignment statements.

Dataflow Modeling

Consider design of 2X1 multiplexers

$$F = I_0.S' + I_1.S$$

```
Entity mux is
    Port (I0,I1,S: in bit;
          F: out bit);
End mux;

Architecture t1 of mux is
  Begin
    F<= (I0 and (not S)) or ( I1 and S);
  End t1;
```

Half adder is having two inputs A,B and two outputs, say, S and C.

```
Entity HA is
   Port (A, B: in bit;
         S, C: out bit);
End entity HA;
 Architecture t2 of HA is
   Begin
      S<= A Xor B;
      C<= A and B;
  End t2;
```

Sequential and Concurrent Statements

1. Sequential statements:

- Sequential statements are compiled line by line.
- Order of writing program will affect system behavior as program is executed sequentially.
- Sequential statements are written inside process command. The if-then-else command and case command are most commonly used sequential command.

2. Concurrent statements:

- Concurrent statements are compiled at a same time or executed parallelly.
- Order of statements in writing program will not affect system behavior.
- Concurrent statements must be written outside process command.
- With-select command and while command are most commonly used.

Concurrent Statements

- Signals are concurrent objects S<= A Xor B; --no delay.
- It is possible to define several values in one signal assignment.

X<= ‘1’,
‘0’ after 5 ns,
B after 7 ns;

Concurrency: Example

Event

Signal assignment is **event controlled.**

Let’s take b<= c; The above statement is executed only when the value of c is changing.

```
Begin
 a<= b;
 b<= c;
end s1;
```

Both are same

```
Begin
b<= c;
a<= b;
end s1;
```

Delay Model in VHDL

There are three types of delay model in VHDL

1. Inertial Delay model

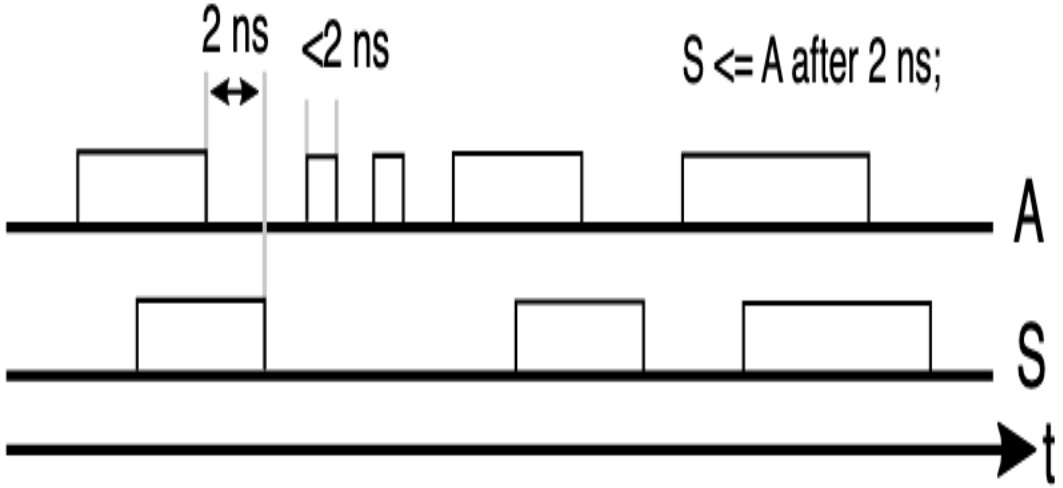
1. Transport Delay model

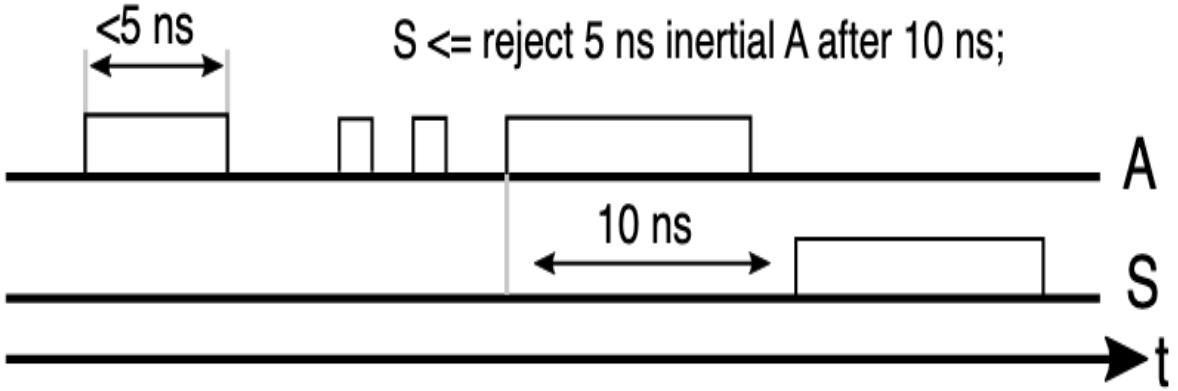
2. Delta Delay model

Delay Model in VHDL

- **Inertial delay** model (default delay mechanism) **:** This type of delay models inherent latency of the logical gate or wire.
- Captures the time it takes for a signal to transition from one state to another.
- Spikes can be handled effectively. Reject command is used to handle the spike of specified duration.

Example: This is used along with **after** command.



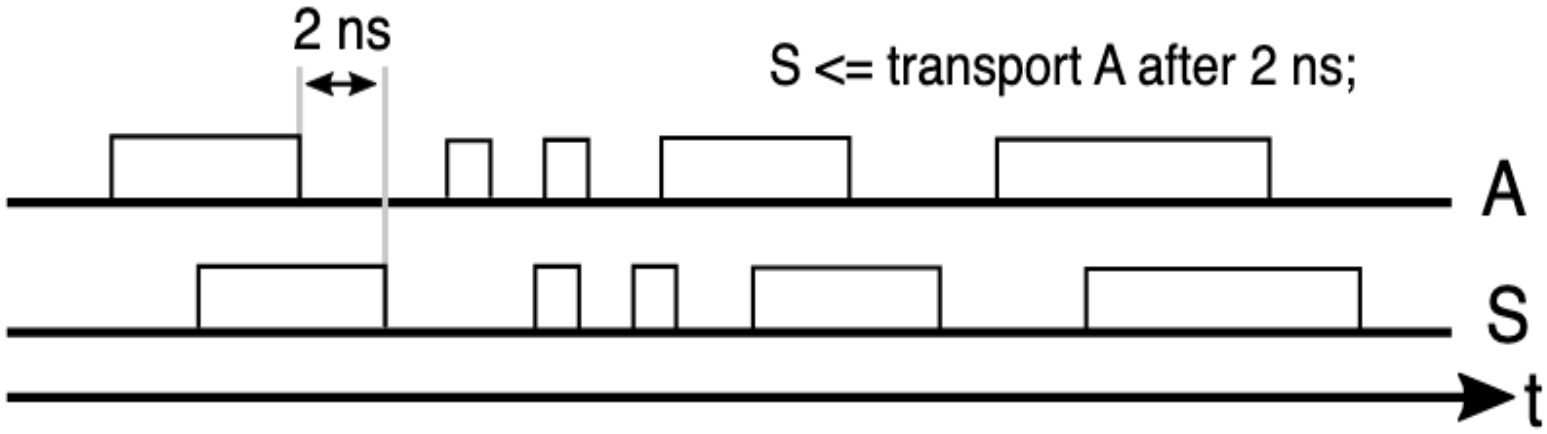


- **Transport delay:** This type of delay modeling is used to model propagation or interconnection delay from one point to other point.

  Q1<= a transport after 5 ns;

The above statement indicates that Q1 will be getting the value of a after 5ns irrespective of spikes.

Example:



Delta Delay

- A *delta delay is* a very small delay (infinitesimally small).
- It does not correspond to any real delay and actual simulation time does not advance.
- This delay models hardware where a minimal amount of time is needed for a change to occur, for example, in performing zero delay simulation.
- Delta delay allows for the ordering of events that occur at the same simulation time during a simulation.
- For example, events can occur at 15 ns, 15 ns+lA, 15 ns+2A, 15 ns+3A, 22 ns, 22 ns+A, 27 ns, 27 ns+A, and so on.

CONCURRENT COMMANDS

- **When-else statements**: This command executes operations depending on the value of expression.

Syntax : target signal <= expression **when** condition **else**

```
Entity ex is
Port (A: in bit_vector (0 to 1);
      F: out bit);
End ex;
Architecture t1 of ex is
Begin
F<='1' when A="11" else
    '0' when A="00" else
    '0' when A="10" else
    '0';
End t1;
```

It is permissible to use several conditional expressions in when-else command.

```
F<=A when enable ='1' else
    B when input ="00" else
    C when clock ='1' else
   '0';
```

With Select statements:

Syntax **With** expression **select**

Example:

```
signal selector : std_logic_vector ( 1 downto 0 ) ;
signal output : std_logic_vector( 3 downto 0 ) ;

with selector select
output <=
"0001" when "00" ,
"0010" when "01" ,
"0100" when "10" ,
"1000" when "11" ;
```

- Make sure that all of the select signal possibilities are covered.

Sequential statements

- In the sequential statements all statements are executed sequentially i.e. one by one.
- In VHDL, sequential statements are written only inside process command.

- If- then- else command: Format

```
If (conditional expression) then
    Statements   ;
Else
    Statements;
End if;
```

Example:

For 2x1 Mux,

```
If (Sel='0') then f<= I0;
else f<=I1;
End if;
```

Similarly, we can write the VHDL model for 4X1 Mux,

```
If (Sel="00") then f<= I0;
elsif (Sel="01") then f<=I1;
elsif (Sel="10") then f<=I2;
else   f<=I3;
End if;
```

Since there are four conditions on Sel, so elsif command is used.

Case command

- The basic format for using case command is given below

Case expression is

When condition1 => statements;

When condition2 => statements;

---

end case;

Example:

Case Sel is

When “00” => f<= I0;

When “01” => f<= I1;

When “10” => f<= I2;

When “11” => f<= I3;

End case;

- Every case command must have its end
- All the choices must be listed or must have <u>when others=></u> command at the end.

- It’s also permissible to define ranges in the choice list. This can be done by using to or downto.
- If a is input of type integer and q is output of type integer then

```
Case a is
 When 0 => q<=3;
 When 1 to 100 => q<= 4;
 When 200 downto 101 => q<=5;
 When others => q<=0;
 End case;
```

- To include several vectors in a choice list in the case statements, it is not possible to use concatenation command (&) to combine the vectors. This is because the case expression must be static.

Case a&b is

The solution is to use a variable or Signal to which the value a & b is assigned. For example

```
Variable temp: std_logic_vector (5 downto 0) ;
Begin
temp:= a & b;
case temp is
```

- Null statements: In VHDL there is a statement that means do noting.

This command can be used when no assignments or any output value to change for signal assignments.

In this case null is used for the case “when others” command

```
case temp is
        When “000000” => q<= “100”;
        When “010010” => q<=”101”;
        When others      => null ;
        End case;
```