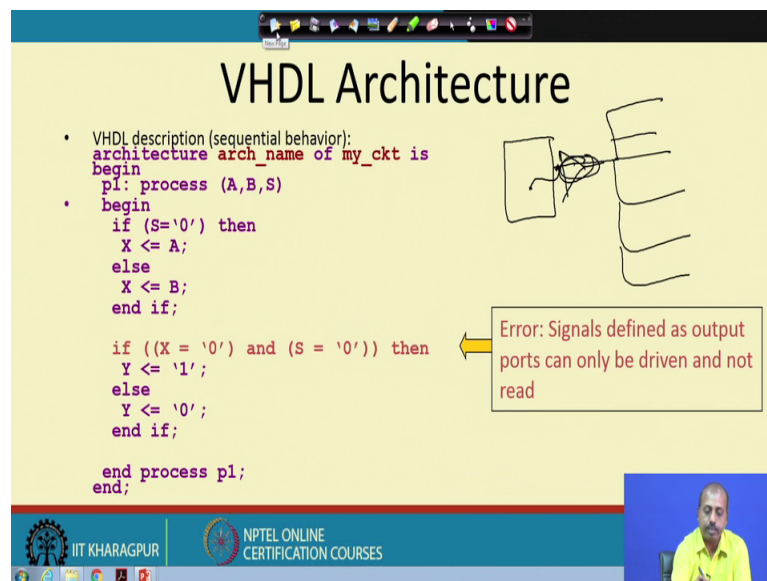


Digital Circuits
Prof. Shantanu Chattopadhyay
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur

Lecture – 47
VHDL
(Contd.)

(Refer Slide Time: 00:20)



VHDL Architecture

- VHDL description (sequential behavior):

```
architecture arch_name of my_ckt is
begin
  p1: process (A,B,S)
  • begin
    if (S='0') then
      X <= A;
    else
      X <= B;
    end if;

    if ((X = '0') and (S = '0')) then
      Y <= '1';
    else
      Y <= '0';
    end if;
  end process p1;
end;
```

Error: Signals defined as output ports can only be driven and not read

The slide features a diagram of a circuit block with multiple output lines on the right. A yellow arrow points from the error message box to the VHDL code line: `if ((X = '0') and (S = '0')) then`. The bottom of the slide includes the IIT Kharagpur and NPTEL Online Certification Courses logos, and a small video inset of the professor.

So, this type of situations are to be avoided that we I cannot take the output signals, output ports, I cannot read it, so we cannot it cannot be reads, they can only be driven. So, outputs can only be driven, so that actually is a good thing, because what happens is that if you are having circuit like this ok, and this is an output ports, now you need to. So, suppose this point it goes to 10 different places in my circuit in my in my whole design that goes to 10 different places.

So, what will be the delay of this line or what whether you need a driver, here to be put or not whether you need a driver here to be put or not, so that depends about the number of panels. But, if I say that this line is also coming inside ok, then that calculation is become is becoming very difficult, so I cannot find out this driver strength.


(Refer Slide Time: 01:12)

VHDL Architecture

- VHDL description (sequential behavior):


```
architecture arch_name of my_ckt is
begin
  p1: process (A,B,S)
  begin
    if (S='0') then
      X <= A;
    else
      X <= B;
    end if;

    if ((X = '0') and (S = '0')) then
      Y <= '1';
    else
      Y <= '0';
    end if;
  end process p1;
end;
```



Error: Signals defined as output ports can only be driven and not read

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES



So, if this is the block and this is the line going out, so depending upon the load that you have here, so you can define determine whether I need to put a driver here or not. But, if you are taking this line back inside the circuit, then I cannot do that calculation. So, that is why this VHDL designers, so they have said that this is not a good design practice, and it should be it should not be allowed, it should be avoided.

(Refer Slide Time: 01:42)

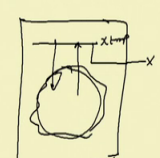
VHDL Architecture

```
architecture behav_seq of my_ckt is
  signal Xtmp: bit;
begin
  p1: process (A,B,S,Xtmp)
  begin
    if (S='0') then
      Xtmp <= A;
    else
      Xtmp <= B;
    end if;


    if ((Xtmp = '0') and (S = '0')) then
      Y <= '1';
    else
      Y <= '0';
    end if;

    X <= Xtmp;
  end process p1;
end;
```

Signals can only be defined in this place before the **begin** keyword



IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES



So, if we follow that paradigm, then we can modify the design like this. So, we can say that instead of X, so we take another signal X t, X tmp or X t m p. And then we say that

the process description becomes like this that if A s equal to 0, then X tmp equal to else X tmp equal to B end if. If X tmp equal to 0 and S equal to 0, then we make Y equal to 1 else Y equal to 0. And finally, we transfer this X tmp value to X, so that is a legal description. So, you can do this thing.

So, signals can only be defined in this place before the begin keyword. So, this is another restriction. So, it is defining like this that I am I am defining a block. So, this is my architecture ok. And within this architecture, at the very beginning, I have to tell: what are the signal lines I have. So, this X tmp is a new signal line that I am defining. So, this is the X tmp line. And whatever circuitry I have here after this, so they may take input from X tmp or they may drive this X tmp. For example, here, so this X tmp line is driven by A, here X tmp line is driv driven by B or sometime later this X is copied this X X tmp is going to be connected to X here so, that that may be possible.

But, I cannot define this X tmp line inside the description of this process. So, you cannot take this signal line inside, so that is not allowed that is not allowed. So, signals can be defined can only be defined in this place before the begin keyword. So, before begin keyword I have to define all the signals.

(Refer Slide Time: 03:36)

VHDL Architecture

```
architecture behav_seq of my_ckt is
  signal Xtmp: bit;
begin
  p1: process (A,B,S,Xtmp)
  begin
    if (S='0') then
      Xtmp <= A;
    else
      Xtmp <= B;
    end if;

    if ((Xtmp = '0') and (S = '0')) then
      Y <= '1';
    else
      Y <= '0';
    end if;

    X <= Xtmp;
  end process p1;
end;
```

Signals can only be defined in this place before the **begin** keyword

General rule: Include all signals in the sensitivity list of the process which either appear in relational comparisons or on the right side of the assignment operator inside the **process** construct.

In our example:
Xtmp and S occur in relational comparisons
A, B and Xtmp occur on the right side of the assignment operators

So, this is the general, so now if you as I said that this process A, B, S, X tmp. So, what I mean is that if you look into this particular description, you will see that whenever any of these values are changing like you see if A value is changed, then this X tmp has to be

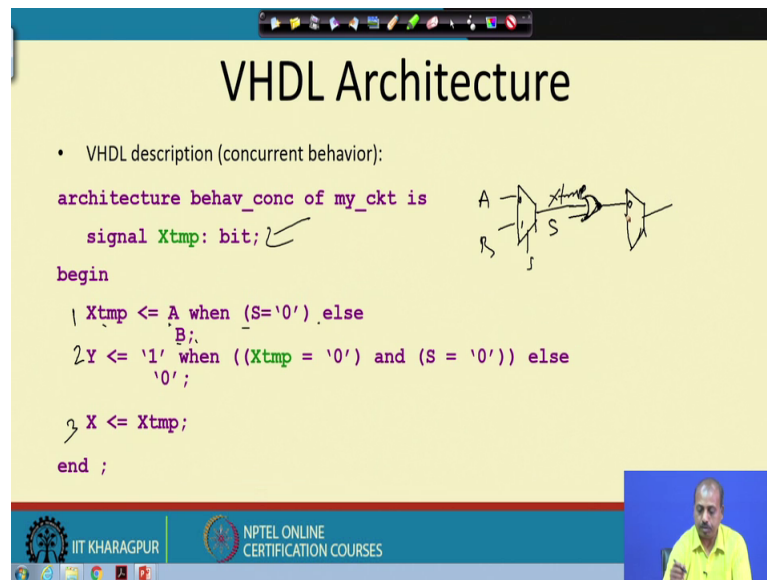
recalculated. If B value is changed, then the X tmp has to be recalculated. Y is Y value has to be recalculated, and X value has to be recalculated; so, any of this signal values changing I need to recalculate these things recalculate the values of X and Y.

So, this is this is done by means of this sensitivity list. So, this signals that we are putting within bracket after the process stay keyword. So, they are called the sensitivity list for the process. So, they will include all signals in the sensitivity list in of the process, which either appear in relational comparisons or on the right side of the assignment operator, inside the process construct. So, you see this is the general rule, so the which appear in the relational comparison. So, whenever I am doing these comparisons, so this will be done ok; or whenever it appears on the right side of this assignment A, B etcetera, so then also this is done.

So, if you do not do this, then what will happen? So, the for example, if I do not include A, include S in this at this point that means, my design is not dependent on the value of S. So, it is not sensitive to the value of S. So, even if S changes; my design is expected not to change, but that is wrong. Because in my description I see that I need I need to have this um value when this value of S changes, there is a chance that is, so instead it was pre previously outputting A and now it should output B. So, this value of X tmp has to change as a result X has to change, so it is not correct.

So, any signal that on which this description is going to depend, so they are those names must be appearing in this sensitivity list. Otherwise, there will be there is problem. So, we cannot the description is not correct ok, so that is the point of sensitivity list. So, in our example, this X tmp, S and the occur in relational comparison A, B and X tmp occur on the side of the assignment operators. So, they are all part of the they are all part of this symbol, so that was the sequential description.

(Refer Slide Time: 06:15)



The slide is titled "VHDL Architecture" and contains the following VHDL code:

```
architecture behav_conc of my_ckt is
  signal Xtmp: bit;
begin
  1 Xtmp <= A when (S='0') else
    B;
  2 Y <= '1' when ((Xtmp = '0') and (S = '0')) else
    '0';
  3 X <= Xtmp;
end ;
```

To the right of the code is a hand-drawn circuit diagram of an SR flip-flop. It shows two inputs, A and B, and a clock input S. The output is Xtmp. The diagram shows the internal logic of the flip-flop, including cross-coupled NAND gates and a clock input.

The slide also features the IIT KHARAGPUR logo and the NPTEL ONLINE CERTIFICATION COURSES logo at the bottom. A small video inset shows a man in a yellow shirt speaking.

So, I was describing the behavior of I was defining the behavior of the process in terms of my circuit in terms of a sequential statement. So, first it will execute this statement; then it will execute this f statement; then it will execute this statement. There are three main statements that we have. So, they will be executed one after the other.

However, there can be another style of description which is known as concurrent description. So, and this is more um close to the hardware, because we know that that in hardware everything is in parallel in nature. So, this in as far as the hardware description is concerned, so it is written line this. So, again this has to be architecture, this is another architecture of this my circuit. So, you see the entity name remains same. However, name of the architecture changes, so behave concurrent. So, this is the name given.

And, here also the signal is defined, so signal is defined as this one X tmp. And then I am writing a set of concurrent statements, so this is so this is one first concurrent statement; this is the second concurrent statement; this is the third concurrent statement. So, these three concurrent statements, so they are put one after the other. So, and they are as if they are three parallel piece of hardware, so all of them are executing in parallel. So, this X tmp getting aim when S equal to 0 else B. So, it is, so it is as it will be it it will be taken like this. So, this S is there, and this is 0 and 1. So, if S equal to 0, so this is A. And otherwise this is B, so this is the line X tmp.

Now, second line say the Y is 1, when X tmp equal to 0 and S equal to 0, so this, so this is my Y. So, this is another multiplexer, X tmp equal to 0 and S equal to 0. So, both of them being 0, so that is your NOR gate. So, X tmp and Y and sorry X tmp and S they will be coming there. And, so this is both of them being 0, sorry this is the both of if both of them are 0, then it should be 1. So, this I need to be this should be an OR gate; this is not an NOR gate, this is an OR gate. Both the inputs are 0, then this is equal to 1, this is wrong sorry.

(Refer Slide Time: 09:06)

The slide is titled "VHDL Architecture" and contains the following VHDL code:

```

• VHDL description (concurrent behavior):
architecture behav_conc of my_ckt is
    signal Xtmp: bit;
begin
    Xtmp <= A when (S='0') else
           B;
    Y <= '1' when ((Xtmp = '0') and (S = '0')) else
         '0';
    X <= Xtmp;
end ;

```

Hand-drawn circuit diagram: A multiplexer with inputs A and B, and select line S, outputs Xtmp. A NOR gate with inputs Xtmp and S, outputs Y. A buffer with input Xtmp, outputs X.

So, this is the first part. This is my S is the select line. And this is A and B. When S equal to 0, then A so this is my X tmp and this X tmp and S, so they are to be connected to form the input to the select input of the marks. So, this is 0 and 1; and 0 will be selected when this is Y is equal to 1, when X tmp equal to 0, and this is Y is S equal to also is equal to 0. So, this is an NOR gate.

So, you put an NOR gate, X tmp is connected, here and S is connected here. So, both of them being 0, the output is 1, as a result this will be selected; otherwise this will be 0. So, this is your Y. And, then X equal to X tmp. So, from this point, so it is going to X also. So, you see it is some sort of a circuit, you can you can at most see look at there is a buffer here, and it goes like this. So, you see that this is a type of circuit that you get.

Now, all these three all these modules, so they are parallel they are working in parallel. So, there are three different statements, and they are working in parallel, so that is why it

is a concurrent behavior of the same block. So, we can do this, so this is this is also valid. So, somebody may describe the circuit from a sequential angle, somebody may describe the circuit from a concurrent angle, but, both of them are correct. So, this way I can have several architectures for the same behavior.

(Refer Slide Time: 10:46)

Signals vs Variables

- Signals
 - Signals follow the notion of 'event scheduling'
 - An event is characterized by a (time,value) pair
 - Signal assignment example:
 - $X \leq X_{tmp}$ means
 - Schedule the assignment of the value of signal X_{tmp} to signal X at (Current time + delta)
 - where delta: infinitesimal time unit used by simulator for processing the signals

Handwritten notes on the slide include:

- A wavy line representing a signal between points X and Y.
- A circle containing the expression $t \in X$.
- The expression $Y := X$.
- Two small diagrams showing a signal being assigned to a variable.

So, far we have talked about signals. Sometimes we are talking using some variables also in our description. So, normally when you are writing a sequential description in our programming software programming languages, we are more familiar with variables not with signals. But, in case of VHDL, we have got two types of temporaries, one some of them are called signals; some of them are called variables.

So, signals are the temporaries for which there are dedicated signals lines or wires that will be there in the system. So, as a result, there will be some delay associated with the values assigned to a signals like if I have got a single line like this. So, if this side you say X; and this side you say Y; and if I say Y gets the value of X, then there is a certain delay associated with this. So, this value of X cannot come to Y immediately, so it takes some time.

Whereas other type of temporary this X and Y, they happen to be variable, and if I say Y equal to X, Y assigned to X that this assignment is immediate, because it is just you can say it is another name of that variable. So, I even if even if I do not use even if I do not

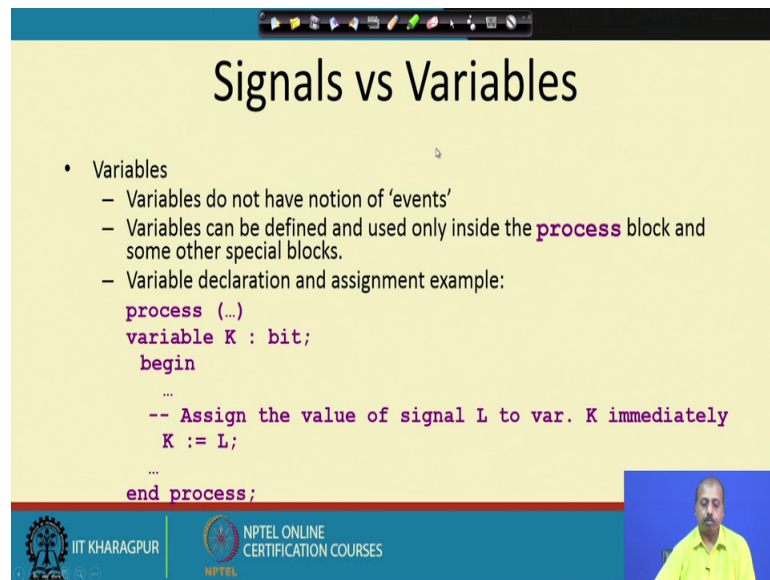
use a new variable Y, so I can take the value X directly and use it. But just for the sake of simplicity in my writing, so I have I am using a new variable name.

So, the signals are they follow the notion of event scheduling. So, once this and event is characterized by a time value pair. So, this X equal to X tmp means schedule the assignment of the value of signal X tmp to signal X at current time plus delta. So, this X has got the um, so this X tmp value will be coming to X the it is scheduled at current time plus delta, some time will be, so will be spent. So, this may be infinitesimally small time amount used by the simulator for processing the signals, but it is not immediate.

Like if there are um if there are if the two places one place, so if there is a one place which is generating this X tmp to to this block is executing the this statement X. And this X is used by another block, then you see that when this X has got the value of this X tmp, so it is not available immediately. So, as far as this block is concerned, so this will see the old value of X for some very small amount of time. So, that actually helps in the simulation process. Otherwise, if that was not there, then the simulation will not terminate. So, it will go on doing the thing again and again, so because it will be always getting a new value of X tmp. So, as a result it will be going into an infinite loop.

Just to avoid that the simulator designers they have said that this X X signal assignment like X assigned the value X tmp, so this will schedule the assignment to take place after time delta, after the current time after some delta time unit. Where this delta is infinitesimally small amount of time, so this is this is this has there is no hard and fast rule like how small it should be is determined by the simulator designer.

(Refer Slide Time: 14:11)



Signals vs Variables

- Variables
 - Variables do not have notion of 'events'
 - Variables can be defined and used only inside the **process** block and some other special blocks.
 - Variable declaration and assignment example:

```
process (...)  
variable K : bit;  
begin  
...  
-- Assign the value of signal L to var. K immediately  
K := L;  
...  
end process;
```

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

On the other hand, variables they do not have the notion of events. And variables can be defined and used only within the process block. So, only within process you can define a variable. And variable declaration assignment, so it is it is done like this. So, this variable K bit, so the begin. So, assigned values of K assigned as L. So, this K assigned as L; so K is a variable, so it gets the value of L. So, it gets it immediately, so this L is a signal. So, K gets the value of L immediately.

But if it had it been a signal assignment, then it will take some infinite single small amount of time for doing the assignment. So, we have got this signals and variables unimplemented. So, they have got there are two different types of notions that we have in VHDL. So, signals means they are say physical words, which will be connecting as a result there will be delays and all; but for variable, so it is immediate.

(Refer Slide Time: 15:16)

Signals vs Variables

- Variables
 - Variables do not have notion of 'events'
 - Variables can be defined and used only inside the **process** block and some other special blocks.
 - Variable declaration and assignment example:

```
process (...)  
variable K : bit;  
begin  
...  
-- Assign the value of signal  
K := L;  
...  
end process;
```

Variables can only be defined and used inside the **process** construct and can be defined only in this place

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, it is just a short hand for some complex expression, so that is done here. So, variable can only be defined and used inside the process construct. So, signals can be defined in any architecture we before the begin statement. But, this signals these variables they can be defined only within the process. So, they cannot be defined outside the process.

(Refer Slide Time: 15:34)

Simulation

- Simulation is modeling the output response of a circuit to given input stimuli
- For our example circuit:
 - Given the values of A, B and S
 - Determine the values of X and Y
- Many types of simulators used
 - Event driven simulator is used popularly
 - Simulation tool we shall use: ModelSim

A → my_ckt → X
B →
S → → Y

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, how do we simulate a circuit? So, simulation is the modeling of the output response of a circuit to the given stimuli. So, we have got this values of A, B and S you need to determine the values of X and Y. There are many types of simulators that are available;

one is called event driven simulator; so that is most popular. So, this event driven simulator is this is s a when event occurs. So, the event occurs on the simulator on the lines A, B and S, so this circuit will be simulated and we will get the outputs X and Y. So, modelsim is a very popular simulation tool that is available. It is also available freely for download. So, you can check it, and you can use it for your simulation.

(Refer Slide Time: 16:19)

The screenshot shows a simulation window titled "Simulation". On the left, there is Verilog code for a process named 'pl'. The code includes signal declarations for Xtmp and XtmpVar, and logic for updating Xtmp based on inputs A, B, and S, and updating Y based on Xtmp and S. The code also shows XtmpVar being assigned the value of Xtmp.

In the center, a table shows the simulation results over time:

Time	A	B	S	Xtmp	Y	XtmpVar	X
0-	U	U	U	'X'	'X'	'X'	'X'
0	0	1	0	'X'	'X'	'X'	'X'
0+d	0	1	0	0	0	0	'X'
0+2d	0	1	0	0	1	0	0
1	0	1	1	0	1	0	0
1+d	0	1	1	1	0	0	0
1+2d	0	1	1	1	0	0	1

Below the table, two boxes provide additional information:

- Scheduled events list:**
 - Xtmp = (0,0+d)
 - Y = (0,0+d)
 - X = ('X',0+d)
- Assignments executed:**
 - XtmpVar = 'X'

The bottom of the slide features logos for IIT KHARAGPUR and NPTEL ON CERTIFICATION, along with a small video feed of a presenter.

So, how does it simulate? So, if this is the description that I have. Now, at time t 0 minus, so 0 minus means just before starting the simulation; the values of A, B and S they are unknown. Then the value of X tmp is undefined; value of Y is undefined; X tmp variable is also undefined; then X the variable the signal X is also undefined. So, this is the condition just before starting the simulation. So, at time 0, if I say that I have applied a pattern 0, 1, 0 in A, B, S at times given simulation starts at this time.

Then a time 0, so this is the situation, so 0, 1, 0. And this X tmp is getting, so this evaluation is done. So, X tmp will be assigned the value A, but after an infinite after a small amount of delay. So, as a result this X tmp Y, X tmp var and X they continue to hold the value X. Then, after some small delay d, this, this this assignments will take place as a result is X tmp gets the value since S equal to 0, X tmp gets the value of A, so X tmp becomes equal to 0. Similarly, X tmp equal to 0; and S equal to 0. So, this Y also gets the value 1. So, Y is also getting the value, so Y sorry. So, at this time X tmp is so it actually while evaluating, so it takes the previous value of this X tmp, and that was equal

to x. So, this condition was not satisfied. So, it got Y equal to 0. So, you get Y equal to 0. And this X tmp var assigned to the value of X tmp, so X tmp value was 0, so that value is coming here.

Whereas, X the variable of the signal X, it continues to hold the previous value because this assignment has been scheduled, but this is the this is the this is with the value of X X tmp value. So, this old value is taken, so it is using this. Whereas, this X tmp var since it is the variable assignment, it takes the value of X tmp immediately and becomes 0 here, whereas X continues to be a unknown. At 0 plus 2 d, A, B, S remains unchanged. Now, this Y becomes equal to 1, because now X tmp value has become 0, so Y becomes equal to 1, so that is changed. And, then this X is getting the value X tmp, so X is getting the value 0.

Now, at time 1, if we say that we change the input pattern say to 0, 1, 1, then this previous value of this um this thing they continue to hold. And it in a similar fashion the rest of the simulation is done. So, you can the whatever way we were describing here, so it is done here and. So, at time 0, so scheduled events lists, so it has got X tmp will be assigned 0; at time 0 plus d, Y will be assigned 0, at time 0 plus d. X will be assigned X; at time 0 plus d, so that is the scheduled event. So, of and the, but the variable assignment is executed. This X tmp var equal to X, so that is executed.

(Refer Slide Time: 19:31)

Simulation

```

architecture behav seq of my_ckt is
  signal Xtmp: bit;

begin
  pl: process (A,B,S,Xtmp)
    variable XtmpVar: bit;
  begin
    if (S='0') then
      Xtmp <= A;
    else
      Xtmp <= B;
    end if;

    if ((Xtmp = '0') and (S = '0')) then
      Y <= '1';
    else
      Y <= '0';
    end if;

    X <= Xtmp;
    XtmpVar := Xtmp;
  end process pl;
end;

```

Time	A	B	S	Xtmp	Y	XtmpVar	X
0-	U	U	U	'X'	'X'	'X'	'X'
0	0	1	0	'X'	'X'	'X'	'X'
0+d	0	1	0	0	0	0	'X'
0+2d	0	1	0	0	1	0	0
1	0	1	1	0	1	0	0
1+d	0	1	1	1	0	0	0
	1	1	1	0	0	1	1

Scheduled events executed:
Xtmp = 0
Y = 0
X = 'X'

Scheduled events list:
Xtmp = (0,0+2d)
Y = (1,0+2d)
X = ('0',0+2d)

Assignments executed:
XtmpVar = 0

Now, at this next time at 0 plus d, so these are the scheduled events executed X tmp becomes equal to 0; Y equal to 0; and X equal to X. Assignment executed; X tmp var equal to 0 and these are the scheduled events. So, X tmp equal to A 0 plus 0 0 plus 2 d; Y equal to 1, at 0 plus 2 d. So, these are the scheduled events. So, it is at this time so this value should be assigned to the variables and signals.

(Refer Slide Time: 20:00)

Simulation

```

architecture behav_seq of my_ckt is
  signal Xtmp: bit;
begin
  pl: process (A,B,S,Xtmp)
    variable XtmpVar: bit;
  begin
    if (S='0') then
      Xtmp <= A;
    else
      Xtmp <= B;
    end if;

    if ((Xtmp = '0') and (S = '0')) then
      Y <= '1';
    else
      Y <= '0';
    end if;

    X <= Xtmp;
    XtmpVar := Xtmp;
  end process pl;
end;

```

Time	A	B	S	Xtmp	Y	XtmpVar	X
0-	U	U	U	'X'	'X'	'X'	'X'
0	0	1	0	'X'	'X'	'X'	'X'
0+d	0	1	0	0	0	0	'X'
0+2d	0	1	0	0	1	0	0
1	0	1	1	0	1	0	0
1+d	0	1	1	1	0	0	0
1+2d	0	1	1	1	0	0	1

Scheduled events executed:

Xtmp = 0

Y = 1

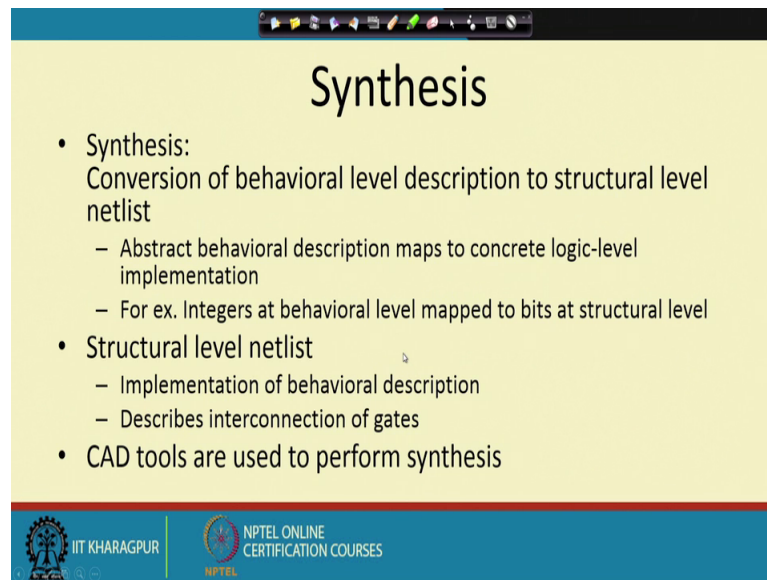
X = 0

Scheduled events list:

(empty)



Next at 0 plus 2 d, so this is the situation. So, these assignments are to be this is the scheduled list of events. And there is a scheduled events have been executed and there is no further scheduled event, because this A, B, S, X tmp, so they have not changed their values. As a result, no further simulation will be required. However, at the next time at time 1, when this schedule will be um taking place this A, B, S value changes, then again the simulation will take place.

(Refer Slide Time: 20:28)



Synthesis

- Synthesis:
Conversion of behavioral level description to structural level netlist
 - Abstract behavioral description maps to concrete logic-level implementation
 - For ex. Integers at behavioral level mapped to bits at structural level
- Structural level netlist
 - Implementation of behavioral description
 - Describes interconnection of gates
- CAD tools are used to perform synthesis

 IIT KHARAGPUR |  NPTEL ONLINE CERTIFICATION COURSES

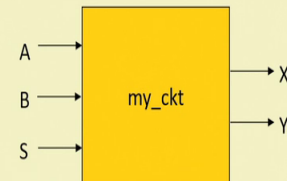
Now, as far as the synthesis is concerned, so synthesis is the process of converting a behavioral description to structural level netlist. The abstract behavioral description maps to concrete logic-level implementation. So, from the description, so it is converted into physical implementation. So, integers these are behavioral level you might have defined something as integer, they will be mapped to bits at the structural level, so that gives rise to a structural level netlist.

So, it will be in terms of logic elements connection between them and all, so that will give us a structural level netlist. Implementation of so structural level netlist is nothing but an implementation of behavioral description. So, it describes interconnection of gates, how this gates are inter connected and all. And there are large number of CAD tools. So, there are CAD design companies that have come up with CAD tools for doing this conversion from behavioral level to the structural level, so that helps when you have got this complex design, so that is going to help you.

(Refer Slide Time: 21:31)



Structural level netlist

- Behavior of our example circuit:
 - Behavior for output X:
 - When $S = 0$
 $X \leq A$
 - When $S = 1$
 $X \leq B$
 - Behavior for output Y:
 - When $X = 0$ and $S = 0$
 $Y \leq 1$
 - Else
 $Y \leq 0$



Logic functions

- $Sbar = \sim S$
- $Xbar = \sim X$
- $X = A*(Sbar) + B*S$
- $Y = (Xbar)*(Sbar)$

So, let us see how this structural level netlist is derived. So, coming back to our behavior, so it was like this when S equal to 0; X equal to A; X say though S equal to 1; X equal to B, and then this output was when X equal to 0, so this was the description. So, logic functions that are associated with it is we say that S bar. So, this is calculated as naught of S, X bar is naught of X. So, X is defined as A and S bar or B and S. And Y is defined as X bar and S bar. So, these are the logic functions for this behavioral description, so that can be derived.

(Refer Slide Time: 22:20)

Structural level netlist

```

architecture behav_conc of my_ckt is
-- component declarations

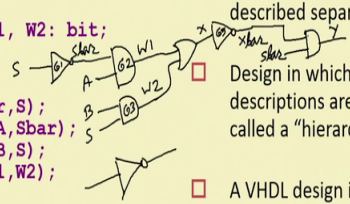
signal Sbar, Xbar, W1, W2: bit;
begin

G1: not port map(Sbar,S);
G2: and port map(W1,A,Sbar);
G3: and port map(W2,B,S);
G4: or port map(X,W1,W2);


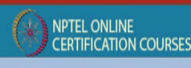

G5: not port map(Xbar,X);
G6: and port map(Y,Xbar,Sbar);

end ;

```



- Gate level VHDL descriptions (**and**, **or**, etc) are described separately
- Design in which other design descriptions are included is called a "hierarchical design"
- A VHDL design is included in current design using port map statement

And once we have derived that, so we can write down the corresponding structural level description. So, you see that this is um this is another description. And then we can have the, we can we can define some components and then we can we can use those components onto this. So, here this component in this case we do not need to define separately, because this not, and, or, so they are available as library modules in the in most of the VHDL synthesizers. So, you do not define these component declarations separately. But, we define the signal lines \bar{S} , \bar{W} , S , so \bar{S} , \bar{X} , W_1 , W_2 , they are of type B.

Now, next what we say is that G_1 , next, we take help of this statements like G_1 port map it is a NOT gate port map \bar{S} S . So, it means that I have got a got an inverter whose name is G_1 , in 1 side this is over. So, it has got S as input, and \bar{S} as output. So, this \bar{S} as output. So, it is the gate G_1 . G_2 is an AND gate, so that has got a port map of, so AND gate has got so one output and two input. So, this is feed to an AND gate, this AND gate is named G_2 . And in G_2 , so I have got this \bar{S} line connected here, the A line comes here and this output is W_1 .

Similarly, I have got G_3 as another AND gate. So, G_3 is another AND gate, where I have got W_2 as the output, and we have got this B and S as two inputs, B and S as two inputs. And then it says that G_4 is another OR gate. So, G_4 is another OR gate. And in this OR gate I have got X as the output and this W_1 and W_2 they are the two inputs. So, this W_1 and W_2 , they happen to be two inputs. And then G_5 is a NOT gate or inverter. So, this G_5 is a NOT gate.

This is G_5 , where output is \bar{X} . And finally we have got G_6 which is an AND gate. This G_6 is an AND gate, where we have got this Y . \bar{X} as 1 input; \bar{S} as another input. So, this is the \bar{S} . And we have got Y as the output. So, this is the final circuit that we get so ok. So, this is so this circuit realizes the same functionality that we have described previously. And you see that the same the description, so I am writing in terms of circuit components ok.

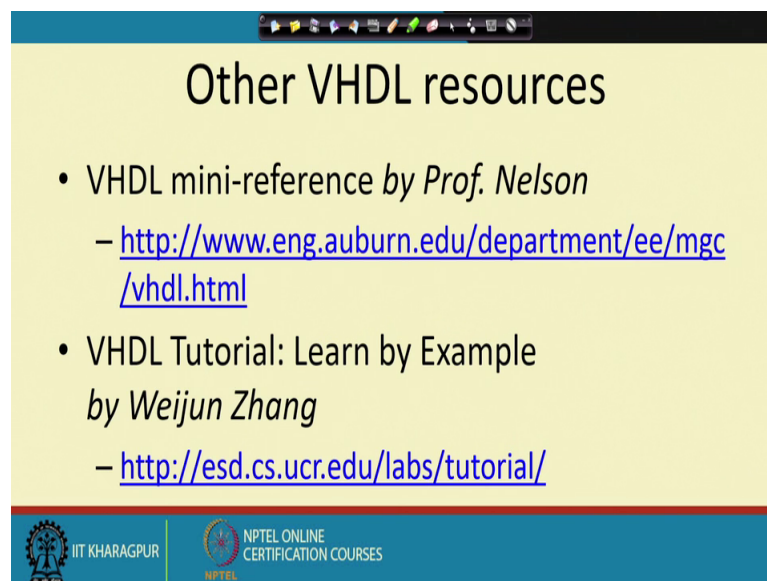
So, I have done the design myself. So, I have done the design in terms of some logic gates. And then I am describing the circuit behavior in terms of those logic gates. So, I have say it says that I have also these are the components that I need NOT gate, AND gate, OR gate. So, these are the three types of components we need. So, in this

component declarations parts, so you can define those component, but here it is not necessary as I said that they are part of VHDL library.

So, then I have to say how the ports are mapped like say an inverter, so it has got this NOT, so this is this is the NOT component. And it has got inverter, so it has got this output comma input. So, this output is mapped to S bar, and input is mapped to S ok, so that way this component is instantiated. So, this is called these are called component instantiation, so that way can be done.

So, so, then, so this is similarly this AND gate, so this is another one G 2 is one instance of AND gate and then the port mapping is like this. G 3 is another instance of AND gate; and this is the port mapping. So, this way we have we can describe the whole circuit in terms of these components and their connections.

(Refer Slide Time: 27:00)



The slide is titled "Other VHDL resources" and contains two bullet points. The first bullet point is "VHDL mini-reference by Prof. Nelson" with a sub-point linking to <http://www.eng.auburn.edu/department/ee/mgc/vhdl.html>. The second bullet point is "VHDL Tutorial: Learn by Example by Weijun Zhang" with a sub-point linking to <http://esd.cs.ucr.edu/labs/tutorial/>. At the bottom of the slide, there are logos for IIT Kharagpur and NPTEL Online Certification Courses.

Now, so these are the some of the further references for this VHDL. So, whatever we have done, so this is just a very preliminary description of this VHDL language, so that is just a start point. So, I wanted to give you a feel like how can we describe the circuits and systems using VHDL language. So, the very vast language and there are very good simulators available.

So, you can download those simulators. And you can refer to some of these mini reference and tutorial. So, they are very good, so you can look into them for getting

further reference for that of and text books are also available. So, you can go through that for digital design.