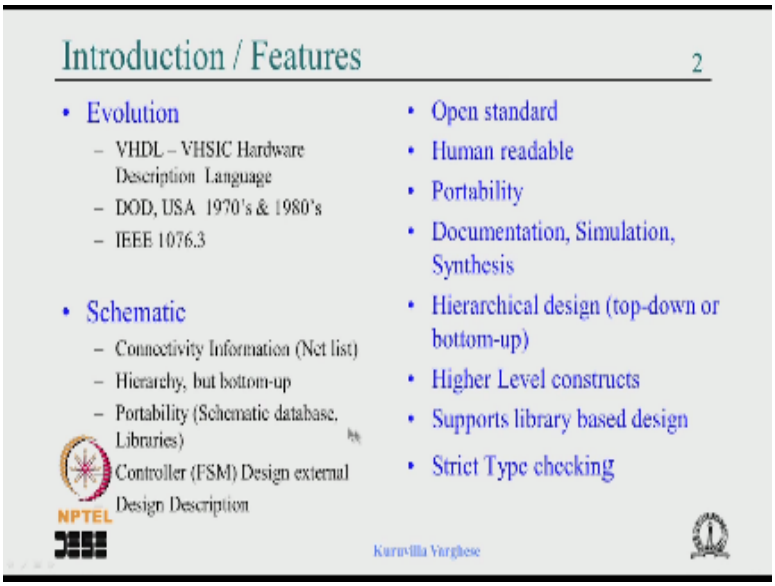
Digital Systems Design with PLDs and FPGAs
Kuruvilla Varghese
Department of Electronic Systems Engineering
Indian Institute of Science – Bangalore

Lecture-10
Concurrency, Data flow and Behavioural models

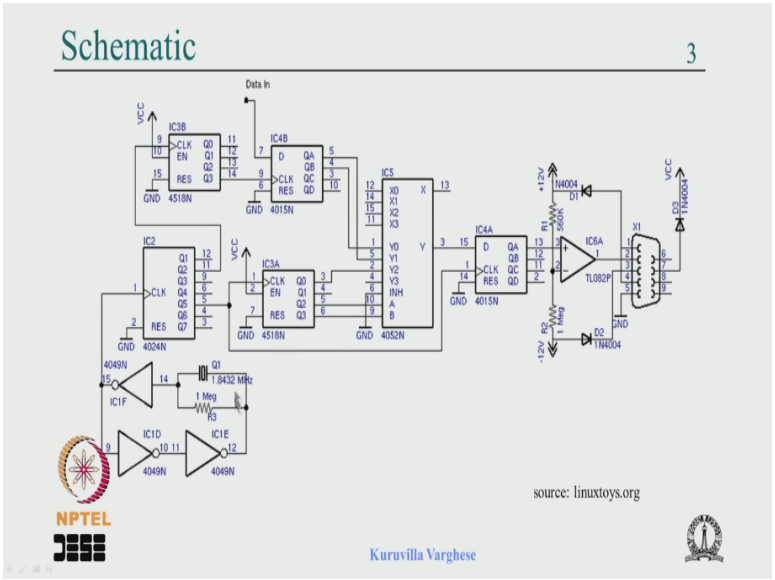
Welcome to the second lecture on VHDL as part of the course digital system design with PLDs and FPGAs. In the last class I give an introduction to the language VHDL, so we will have a quick run through the slides and come to reasonably serious part of the VHDL.

(Refer Slide Time: 00:44)



So let us go to the slide, so basically we have looked at how it is involved.

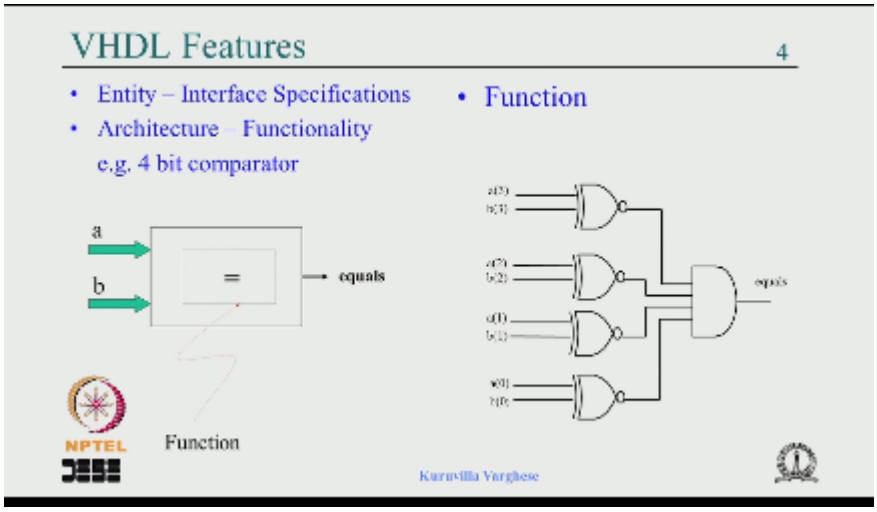
(Refer Slide Time: 00:50)



The stem from the sum lath in the schematic in the sense that it does not it only represent the network I mean the interconnection of the components or necklace and there is no hierarchy supported it was proprietary and it needed at design description because from the schematic you cannot make out the function of the circuit and when you have a company has multiple vendors using multiple CAD tools, integration understanding all becomes difficult.

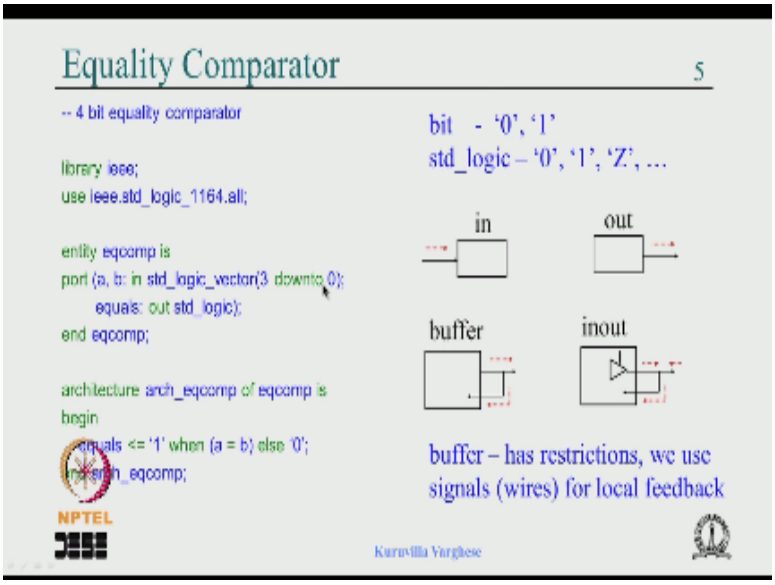
So for a documentation and describing the circuit, so for documenting describing the circuit documenting that started subsequently tools used for simulation because the description was unambiguous ultimately people thought why not used to generate the circuit itself, so 3 purposes this is human readable as well as the computer algorithms work on to do all these you are not simulate and synthesis. It support hierarchical design higher level constructs, library base design, types to type checking in all that.

(Refer Slide Time: 02:01)



Basically BHK last two description of a compound and has two part 1 is entity which is basically the name of the block inputs outputs, the direction of it weather in or out and the data type and architecture is a functionality in terms of the input and output. We have taken an example of for better quality comparative we know the circuit for example-nor gates and then AND gates.

(Refer Slide Time: 02:29)



We have seen an example code as I said the inbuilt data type is bit which support only 01, to be able to use something serious like a tri-state and I do not care and all that we use a data-type call STD underscore logic and we call it standard logic. So the standard to be able to use that we use this library and used as I said I will use the keywords in green all other thing in blue.

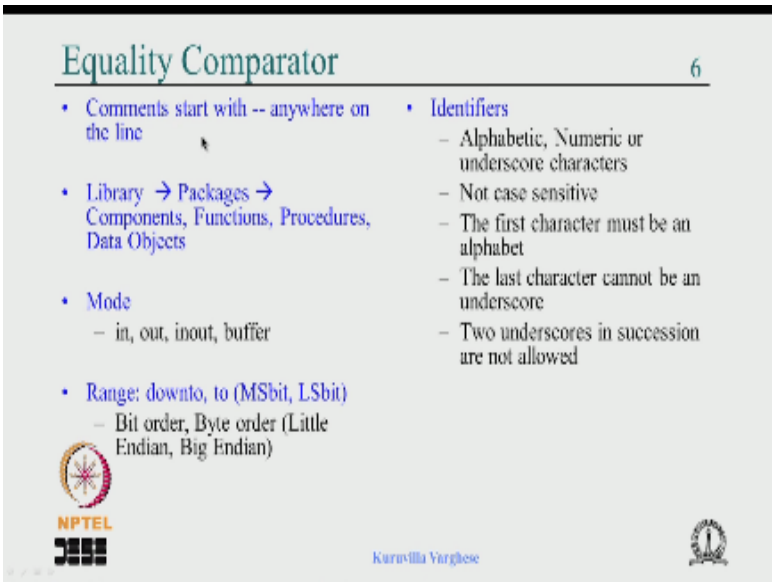
So that is why does library and use case and library has the the like various packages and packages as components and so on inside and this is a comment which cannot be nested and this is the body of entity where the ports specify the input and output, name, the direction and data type and STD logic vector is an array of standard logic and we have seen in, out, in means.

The input should come on right hand side of the assignment and cannot come on the left out is output it should come on the left-hand side in assignment and we are some requirement if there is a local feedback or this output is used as input to the further circuit then you have to declare this is buffer then only you can use output on right hand side ok, but this is restriction in the sense that cannot be used with multiple components.

So essentially use signal to circum that problem and will declared as a signal and assign the signal to and output pin, ultimately there is in out which essentially needs this particular signal or the port can drive out and also somebody when this is tri-state cut off something and drive in and that is different from this and when you have this picture only used in out and this a specified the bit order.

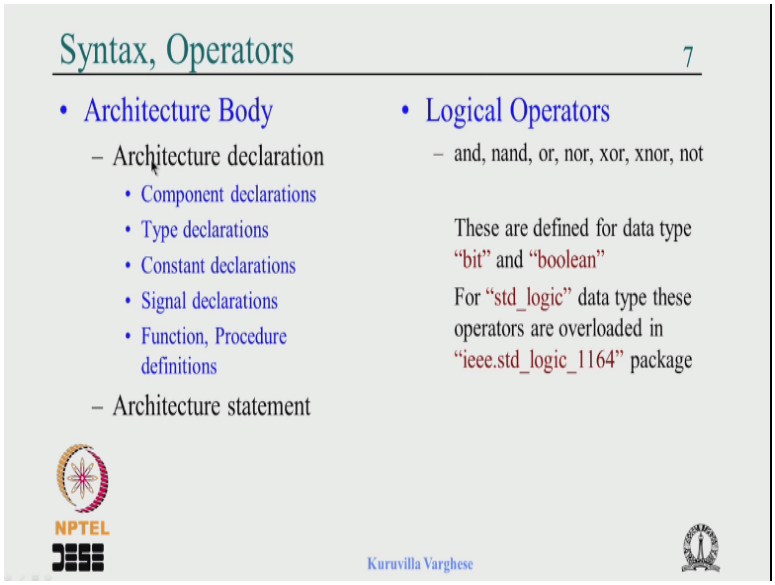
This is Ms bit, Ls bit so depending on little Endian of big Endian of use. This is the assignment operators, this a equality operator which say that equal is get one and one is defined with quotes because in the library it is defined like that when a=b else it is 0, so that is what is the kind of a sample example code.

(Refer Slide Time: 05:00)



And we looked at all these is the command at the libraries, the mod bit order, and alphanumeric and underscore can be used k not case sensitive first character should be alphabet the last character cannot be underscore and 2 underscore in success and success is not allowed all that.

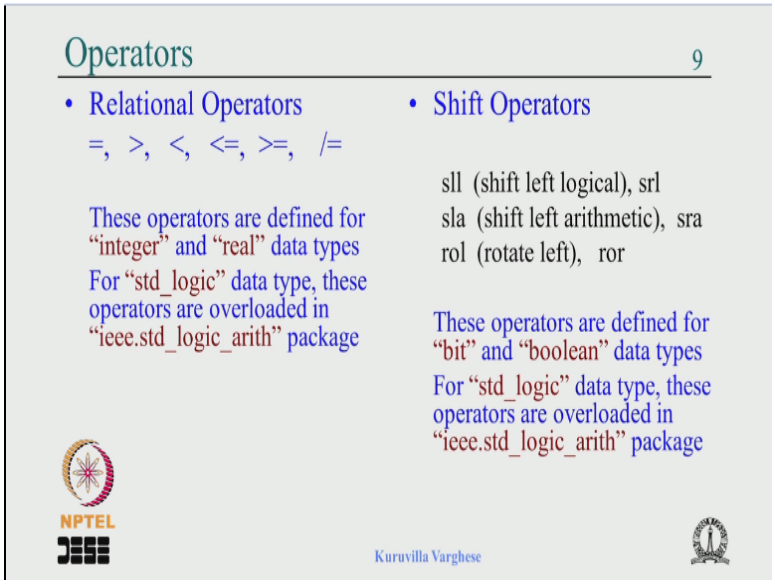
(Refer Slide Time: 05:21)



And the architecture body before you begin yet you can declare many things and after the beginning of the statements. These are the logical operators which is the fine for bed bath in the standard logic 1164 package, this is overload for overloaded for the standard logic data type and arithmetic and arithmetic operators and there is mod and rem we have seen.

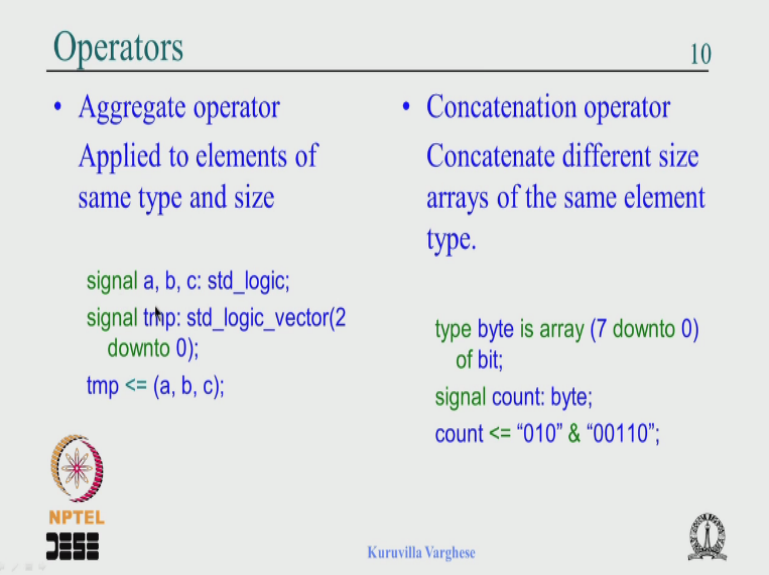
The difference this rem is the real works with the negative number the real mode of the computer science and once again this define for integer and float for standard logic these operators are overloaded in this particular package standard logic unsigned.

(Refer Slide Time: 06:07)



Then you have relational operators again this is overloaded for standard logic and standard logical arith and again shift operations, shift logical, shift arithmetic which is used for sign extension and 2 compliment numbers and rotate all of and rotate all right and these are overloaded once again in standard logic arith.

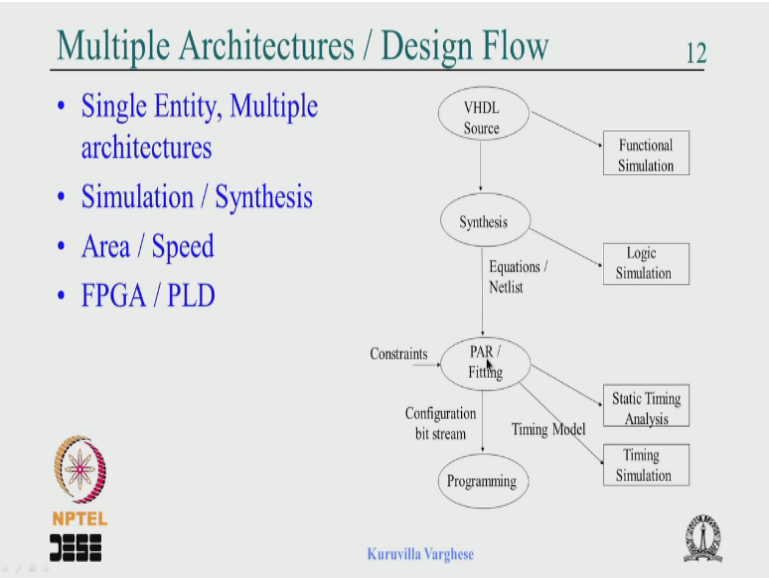
(Refer Slide Time: 06:29)



And you seen aggregate operator if you have some signals of the same type and same size that can be aggregated together to a bus or a multiplayer, but more useful thing is to have an concatenation operation which will work for the same type but different size elements and we have looked at the presidents of various operators the priority episode is ranging from 1 to 6.

That mean this is the highest precedence but when the same precedence is there it is left to right evaluation, but as I said it is better to use the appropriate parenthesis or bracket for easy readability and for understanding because maybe some people were used some other language comes and read it is not be confused for simple cases we can probably when you have a multiplication, addition you can use without brackets.

(Refer Slide Time: 07:29)



And now let us start todays part, if you look at the sample code see we have given and name for a name for an entity, entity some name is and end that me ok that is enough syntax, does a semicolon is a kind of termination character but when it comes to architecture there is a surname for architecture ok at the beginning it looks little awkward because I need to find some entity name why not use the architecture of it.

You cannot say why give a name, so this suggests that in principle you can write multiple architecture for a single entity okay. In VHDL you may have a single entity but you could write multiple architecture and at the time of synthesizing or compiling for simulation you can say which particular architecture you need to use ok. So how to specify that we will see later. So at least we will look at what are what is a kind of idea behind multiple architecture ok.

what is the use of specifying multiple architectures, so definitely one can think of this I like you can have an architect of assimilation ok I said I said at the beginning the VHDL anything you write using the syntax can be simulated. If you know the syntax of VHDL you write any code you go to the simulator and that whatever you have written can be simulated, but unless you take care may not be synthesized ok.

So these 2 are different games all together it is unlike the sequential languages like a C program or any other computer languages where are you can write the code that get executed that can be simulated but in VHDL there are changing your right you know syntax correct, logically correct can be simulated but in may not be able to synthesise that. So maybe you want to write you know you are developing a system and you want to quickly check whether it is the ideas are ok.

Maybe can write a without much thinking much about synthesis, you can write a quick code and make sure that your idea is correct and then you can work on how to write a code which can synthesise that so, that could be one use of the multiple architectures . There is another use you know that you again the kind of overview lectures we have looked at this part, there are constraint which is area and speed.

And we mention there that it is not that always it is possible to get a minimum area and maximum speed when you go for minimum area you might end up large delay or less speed,

when you give implement something with a large area you might get less speed and we mentioned the ripple adder and carry look ahead adder as an example simple example. So maybe when you write some entity and architecture.

You might write some architecture which gives a very minimal area but less speed or which gives a really good speed but the area may not be minimal and things like that and once again maybe you can write an architecture for you know suited for FPGA in a particular way we will see maybe how to write optimal code for FPGA and PLD and all that, maybe you write something specific to FPGA some code and something specific to PLD.

Maybe something even specific to ASIC you know that whatever code you have written the FPGA has lot of inbuilt components and you might instantiate this components in the code which may not work in the case of ASIC and you go to the ASIC, so sick when you go to the 6 so you could have architecture for FPGA 1 for PLD and 1 for ASIC and so on. So you see the VHDL you know idea providing multiple architecture is a really good use it as and you should be able to use it.

And but in any case when you have multiple architecture before whether you stimulate a synthesizer beginning have to choose which particular architecture you are going to use it ok that has to be specified and that syntax is called configuration as we proceed little ahead we will see that the configuration syntax. Now let us look at the design floor we will keep the JPG and PNG in mind and the VHDL as a friend and ok.

So we let us look at the how the design methodology of severe assuming that we are designing for FPGA and PLD and we are describing the design in VHDL or Verilog. Some kind of hardware description language that is the basic ideas. So initially you start with the VHDL source code, so you can imagine and some kind of editor here, you enter the code normally editors give you some templates some give you I too like you specify the inputs and output.

And name of the entity that create some template for kind of making life easy, so that it is available so you edit the VHDL code, once you do that before proceeding with any of the further processes you do a functional simulation okay that means that whatever VHDL code

you have written is simulated in a VHDL simulator. So here we are not dealing with a circuit, we are dealing with the code ok.

So it is convenient because at the code level itself we there are some kind of logical errors that can be quickly corrected without going for the further step. So you might start with VHDL do the functional simulation ok. Now these words know the functional simulation there are words like behavioral simulation, this might differ, you know people maybe you refer to some text book somebody else might call behavioral simulation.

And they might make a difference between function simulation and behavioral selection and so on. My advise you that you do not get kind of in a box by the particular terminology used you get you understand the concept you understand idea generally like you go for an interview somebody say that it not be call function simulation it should be cal as behavior simulation agree with him or her so that you can get the job.

Do not worry too much about the terminology ok. So these are not let you know something 10 commands written on the tablet. So let us call functions relation for all course, so you stimulate maybe you find error you come back correct, you tried quite a bit and you get the correct code in the first place, then you go for synthesis where you generate from the written code the necklace or the equation.

We are generating the logic circuit in terms of gates and flip flops sometime the synthesis tool can be smart can be quiet link to the next processor sometime it will generate the circuit which is very specific to the device used maybe the FPGA are some kind of combinational circuit elements are the synthesis tool can generate so cute in terms of those. So I am giving a kind of generic description.

But this can be very general or specific and things like that. So and once again you may not unless you write properly which we are going to learn how to write properly that, this may not happen the synthesis may not happen that case have to get back and it right and make sure that in synthesise the proper circuits it but after practice if you follow the strict rules there should not be any problem at this point you can go for a logic simulation.

That means here what your simulated was a code, hear what your simulating is a circuit which implement the equal and functionality whatever written here in the code the equal and functionality in terms of the logic circuit simulated here, mind you at this point there are not delay, suppose you have an AND gate, you have written here and you give the input, you can see that when you simulate 100 nanosecond.

The one of the input change the output also change at 100 nanosecond because we are not used any device yet. So at this point there's no delay the no mapping to the device. So the functional simulation and logic simulation are kind of 0 dealer simulation because it is just adjust the logic that logic which is not kind of fitted into a particular device. So the next step is something call PAR or place and route and forfeiting okay.

Normally the keys of FPGAs we call PAR and the case of PLD we called fitting and you have to give the constrains ok. There are two type of constraints, IO constraints and the timing control. You know that the FPGAs are field programmable, the PLDs are field programmable device. So you can there are a lot of generic Io pins, so you can map your signals the input and output to the any of the user IP pins conveniently ok.

And so you have to specify suppose you have an address line A0-A15 you have to say which been A0 comes which A1 comes and so on, that is IO confines and we have seen the expression for minimum clock period in a register to register path, so it is possible that you can specify the timing constraints, you can say at this point when the algorithm do the place and route.

I would like to have registered to register path with the delay of a nanosecond because we are planning to clock the circuit at 200 megahertz. So the clock period the minimum clock period can be specified, so depending on how you specify this place and route to will keep the components closer did used shorter wire to interconnect minimum number of switches to interconnect and so on.

As it interconnect as it routs it keeps track of the timing and any time it takes it say you specified 5 nanosecond when it comes back and find an alternate path, so the timing constraints also can be specified at this point in time and if everything become successful you

will get a configuration with which can be used by a programming tool to program the FPGA or PLD.

But before doing that now having places particular secured in FPGA or PLD we will be able to do the timing simulation, but now the device now that you knows that a particular register to register path take so much delay or with the combination circuits from the input output how much delay it takes in terms of the resources of the FPGA, the wire delays of the FPGA.

The delay of the switches that in the connect the wires and all that. So there is this tool can generate a timing model and instead of logic and function of simulation in the same simulator many times the VHDL simulator used to simulate with the timing model can simulate the timing now if you have an iron gate and you give the input 100 nanosecond. You will see the output is coming after some delays maybe will come after say 102 nanosecond depending on the delays.

You know kind of incurred within the FPGA, so but only thing is at so complex circuit is very time consuming because we have kind of detail model of the wires, switched and the various blocks within the device. So to do a timing simulation with take long time is quite compare, so many times to the people at beginning do something call Static Timing Analysis. In all the simulation you know that you have the model of your other circuit.

You specify various input and you verify the output ok that is the basic game, suppose you have clock reset input you give the clock you reset, you give various input and you check the sequence of output ok, that is how it is verified. In the case of timing simulation all the delays will be shown on this occasion, but the Static Timing Analysis is a very quick analysis where suppose you have register to register path.

It is not that you give clock reset and input and try to check the output waveform but what is done as that all the delays from the source register to the destination register is added together and that is reported ok. So it is very far but the real problem is Static Timing Analysis is that any not be accurate because in we need a complex circular path there may be some say 6 registers and 12 destination registers.

And some of the path may not be active at all in real life, but when you do Static Timing Analysis there is no way for you to know which is the path is active because you are not simulating, you are not giving clock and the state machine is not the controller is not active it is just the addition of the various block delays and the real behaviour of the circuit is not known that is not only if you really give the important simulated.

So the Static Timing Analysis is a quick estimate and you have to tell the tools some paths are not use that is call false path maybe from source to destination everything may not happen in the next clock cycle, maybe there is a source register which is clock at 1 clock, it goes through some kind of combination circuit and after truth like clock cycle it may be talked into the destination register.

But this tool has no way to know that, so but when you do the timing simulation it is very clear but in Static Timing Analysis as need to be told all these things that this path is not use this particular part 1 to Clock cycles and so on. But then it can be accurate and this is a very quick kind of estimate before going for timing simulation, supposed to do the park and you do Static Timing Analysis and it is satisfied then you can go for timing simulation maybe of written all these steps and at the end you can program it.

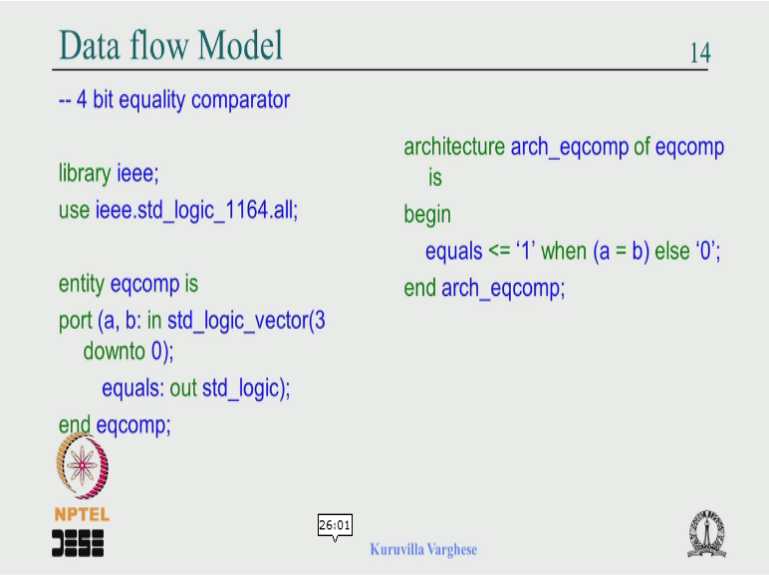
So that is the design methodology of FPGA with VHDL as a source and for that you have an editor, you have a synthesis tool, you have a place and route of fitting tool, you have a constrained editor, you have Static Timing Analysis tool, you have a programming tool and have a simulator which does the function simulation logic simulation and the timing simulation.

And most of component vendor the FPGA or the PLD vendor they give them separate tools but they give a GUI integrated GUI from which you can call the individual you know they are the components are using the graphical user interface, so it is convenient but they also allow you to write some kind of self script to use separate, so that you can run it without the GUI when you have complex design it may not be very convenient to sit in front of GUI and start clicking many selections.

(Refer Slide Time: 25:26)

There will be good whatever is the commonly used whatever is used for a project can be in a in a script form and you run the script and it writes output to the file various file may be at the end of the 6 hours 8 hours a day you can check the various output how it goes and things like that so that's design methodology as I said it uses editors in the stool and standard IT Park to door fitting tool a simulator VHDL simulator which can do all these and Static Timing Analysis tool. So that is the design methodology.

(Refer Slide Time: 25:46)



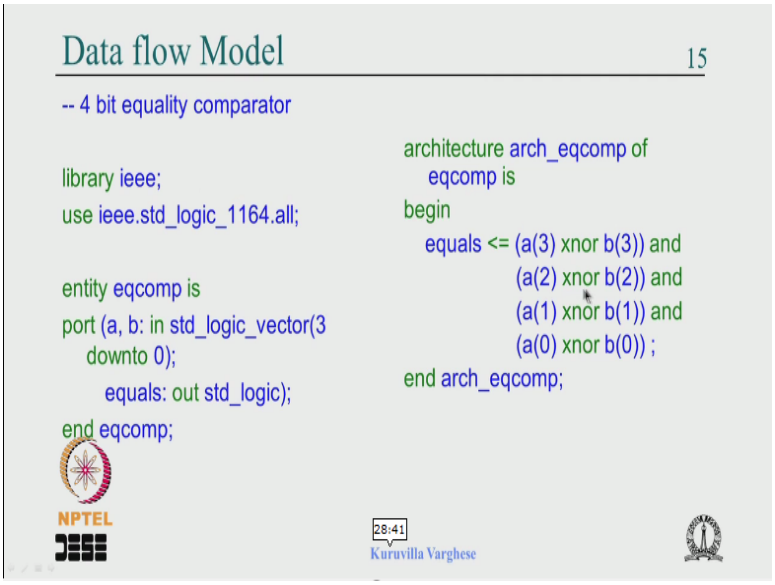
So now let us look at the various ways of disc arriving the components of circuit, the first thing is the data flow model this is what we have seen the we have already seen the data from model and we take as an example for better quality comparator, so this is the library declaration, use declaration, entity declaration with the port, so you have a, b as import which is a 4 bit vector.

Equal is a single bid output we have seen that and architecture we given name of this particular entity and you can declare many things with simple quotes of any not declared the begin you say = is assigned of get one when if a and b are equal else 0 that means a when A and B are equal get 1 otherwise will get 0, so this is a construct which is a concurrent state when else and this is assignment operator.

This is the kind of relation operators, checking other a=b and this is enough and this is call data flow because it shows the floor of the data from input to the output as once again as I said maybe some other literature book my show this is this why some other name but then you basically understand what is the idea behind it than latch on to the particular name because in our field we will have we will be forced to use the metaphor.

Like you know we call a hardware, a platform and many things like that we use we borrow from the English some equivalent word, so do not bother too much about the real meaning you do not ask why this platform whether there is some kind of train is coming to the platform also some something like that no point. So maybe some text book may call behavioral, I do not bother about it. So there is another way of describing the data from model.

(Refer Slide Time: 28:01)



So let us see that the same thing exactly same thing there for better quality comparator, now instead of using a when else statement we are writing the literally the equation using the logical operators, so we are writing = this is the same circuit we have describe for exclusive

NOR gate for a3b, a2b to a1b1, a0, b0, the output of which is ended together. So that is we are just fighting a3 xnor b3 and a2 xnor b2 and a1 xnor b1 and so on ok. So most people at least at the beginning this looks a better call them this ok because you have we are describing the circuit, so you might think that this will end up with a very nice circuit and this might end up with something kind of complex.

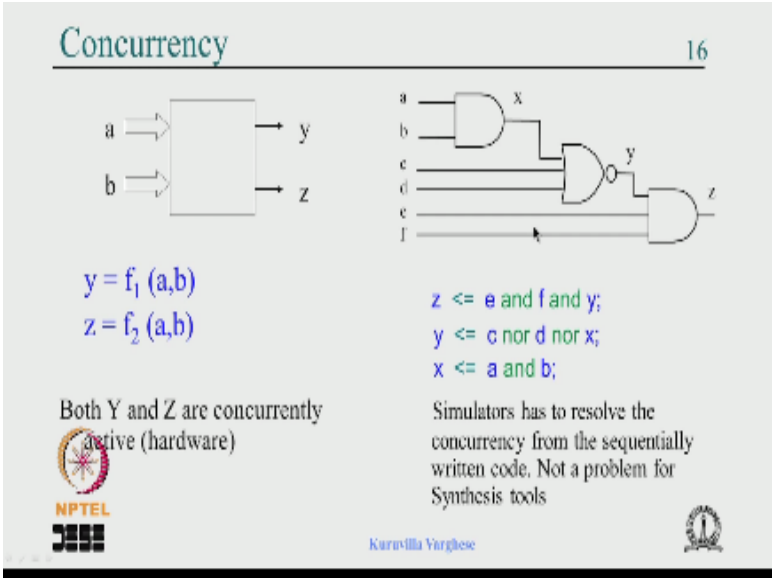
But you need not worry you just think of say a scenario user offer better quality comparator. If you are using say 31 bit comparator for whatever reason, so then we will write ab in standard logic vector 31.0 and you know that this goes down all the way a31, b31, a30, b30 and so on all the way up a0 b0, so this can be quite difficult to even if you copy-paste edit all that and do it.

But if you come here the previous code you just you want to 30 bit quality comparator just write 31 down to 0 and no changes required. So if you ask me whether the when else or the equation is better then definitely this is much more concise and we will see that you know this is for the tool this will generate the exactly same circuit as that and we will see how the synthesis tool kind of generator circuit out of the description.

So we have seen 2 data flow model, one is using the when else concurrence statement and 1 uses the kind of logical operators like xnor and all that and we will see how the synthesis really works later how these operators work what are the various concurrence statement, how it can be used and what is the meaning of photosynthesis is concerned all that we will handle.

Now you are looking at the various models of description for the first one was a data flow model. So let us move on before going further it is very important to consider this particular fact you know concurrency.

(Refer Slide Time: 31:02)



So if you take a digital circuit I suppose you take the keys of equality comparator there are two input signal a and b which of 4 bit, assume that the two outputs maybe this is an equality comparator output, this is a greater than output ok. So assume that if a changing at 100 nanosecond, so maybe say a was less than b then a becomes equal to b then z changes, so essentially if a changes both y and z can change.

Like suppose a was greater than b, so y was 1, z was 0 ok. Now you changed a that is equal to b, then y transit from 1 to 0, and z transfer from 0-1. So for a change in input both can change at the same time, so but when you write an expression in a sequential language, you have to write something first and something else second okay. So but you should know that is not like a c language.

In a c language you write something sequentially and execution happens from the top to bottom. You can use the model is at that you take the first statement executed and update the result take the second instruction do the execution and update the results and so on. But in the case of a VHDL when something changes both output may have to change at the same time ok same time that does not mean you compute at the same time.

But at the same simulation time when you simulator a circuit both been concurrent has to be computed at the same simulation time, that you should keep in mind it is not like a normal you know sequential language, the VHDL, both y and z are concurrent their active. So now let us look at little more serious example of that say we have a circuit like this, we have any

be going to an AND gate which produces an internal signal call x which along with the CD goes to an XOR gate, output of XOR gate is called y.

And that along with ef go to z okay. Now suppose we have written in VHDL code the expression like this have a look at the expression you see that the z is written first, z is written first, z is described as e and f and y and f y is written as cx nor d nor x. x is written as a and b. Suppose the simulator is trying to simulate by going from top to bottom as in a c language say like initially suppose there was an event on a.

And it is not computing, so z is computed fast using value of ef1 and y so the y will not show the value as before and z will remain same. Then y is computed using the old value of x, current value of x and z and d. So y is going to remain same and now the x is x computed the a has changed, a 10 nanosecond. So x will change but nothing happens to y and so ok.

So this is the currency problem as far as stimulate this concerned ok, when somebody write the code like that simulator as somehow have to figure out saying that if you sequentially compute from top to bottom the real time behaviour of the circuit cannot be simulated. We will see how the similar to handle said but you get, you try to understand the issue.

And you might say in a symbol case like this why do you write like this is not deliberately have shown that was by far second hand x you know the 3rd but you could as well you know your end xyz but you think of a real life scenario when you design a CPU will break down into pieces and give one piece 1 designer, second piece to another team, third piece to another team and 4 tram is trying sitting there on top and integrating interconnecting it together.

So when you interconnected the complex code you know this particular block itself has lot of codes and when the top level everything is interconnected together and it is very difficult to analyse which way the data flows and it is not keep the order and things will be Messy and so simulator as a little tough job of resolving the concurrency from the sequential written code.

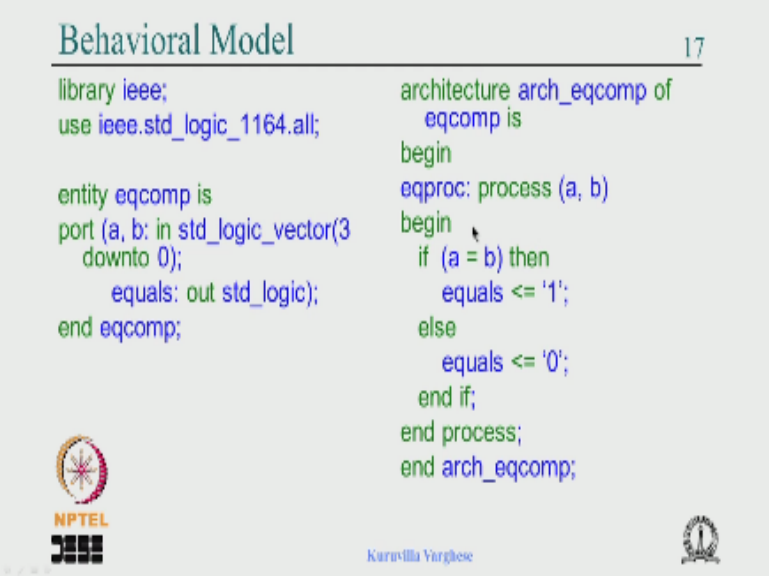
But if you think of synthesis is absolutely not problem whichever way you write it does not matter because they like if you are trying to generate the circuit of this from this code it does not matter whether z is written first, Y is written first or x is written first or in any particular

order because this a x is a and b and y is c nor d nor x, so the moment x is there you just put this.

And when it comes to this place why you get this z nor d nor x, so that is one. So it really does not matter the how the code is written which order the code is written as for the synthesis is concerned because synthesis tool is worried about the structure of the circuit not the real time behaviour, but they simulator has to simulate the real time behaviour any it has to worry about the concurrently ok.

So we will see how this is handle, but I bring this particular issue to the front is not the job of simulator and the job of synthesis tool is different, synthesis tool looks at the structure of the circuit, but the simulation tool worry about the real time behaviour of the circuit. We will see how it is handle but that you keep this in mind and let us move forward.

(Refer Slide Time: 37:36)



Let us come to the second level of description model which is called behavioral model and we are taking the same example equality comparator. So the library declaration is same, entity parts everything is same, but when it comes to the architecture it uses a particular Syntax of the body call process ok, so the syntax is you can give some label for the process because there could be multiple processes in a particular architecture.

So you can given name and keyword is process and you open bracket and you write some signals in the bracket and this is called sensitivity list that means this particular body of the process is sensitive to changes or even on this particular signal ok. So basically normally

these are input signals as above the block with input a and b, the input signals are written in the sensitivity list little more detail is required but then we will see the later.

So you have in Boots return in the sensor with us when an event happens in the signals in the sensitivity less we are talking about the simulator will run from the top to bottom and whatever is written as description these computer from the top to bottom wants, so that is the ideal of process and the process used the construct like if K you know the loop 4 loops and all that. So basically this allows this process allows you to describe the behaviour of the circuit in a behavioral fashion using higher level construct like if k is 4 loops and all that.

That is why it is called behavioral model, once again this terminology may might be different in different text box, different literature, do not worry too much about it and do not again make you know kind of big out of this processes, do not compare this with the multi process of the operating system, how then great you know kind of parallel with that process, maybe there is a kind of some kind of similarity in a in a computer science operating system processor.

The process become active when there is some event on a particular a signal, so maybe that is why it is called process. Here also this computation happens when there is an event on the signals in the sensitivity less then it goes from top to bottom. So here it simple if something changes a changes it comes to this with says if a=b equal get one else equal gets 0 end if, that is index, if something condition then output was assigned, else output assigned.

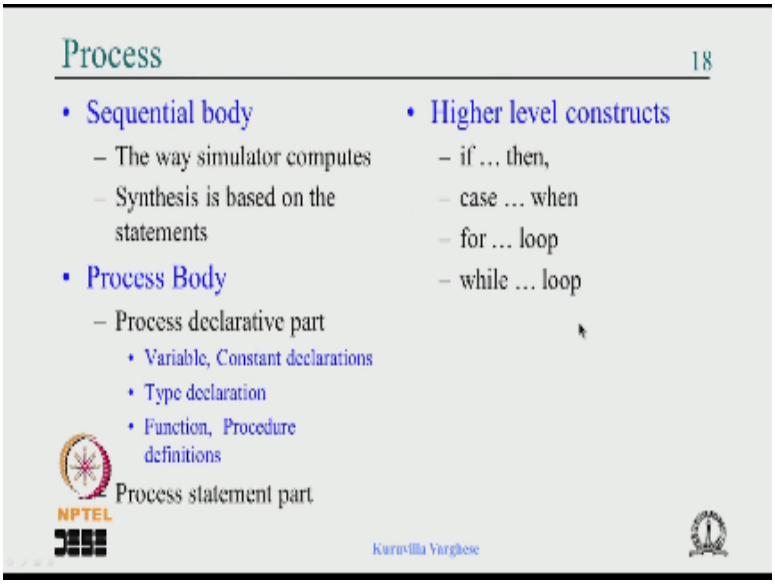
And you say end process the processes end that is all very simple and as I said that goes from sequentially from top to bottom, so when the simulator compute the behaviour whenever there is anyone it goes from top to bottom ones for a particular event ok. Otherwise if the second even happens again it goes from top to bottom ok. Now for the synthesis tool it is not going to bother about all these sensitivity less even happening and all that.

Because synthesis tool is not worried about the real time behaviour. So it looks at this quote what is written in the code, the code say if a=b equal is 1, else equal is 0, so generate the circuit matching this behaviour that is all ok. So it is equal and do the when else code when else said that equals get 1 when a=b else it is 0, so it is identical to the when else if then else if then else is identical to when else only thing is that the if else is used in a process and mind

you process is a sequential body you can use only if is or you know for loop you cannot write the when else or with select there is a Syntax call with select within a process board you can only use if and kind of the keys and all that.

And definitely you cannot write if and case and all that in an architecture directly it has to be used within a sequential body like processes, functions and procedures. So that you keep in mind, that idea process and I told about how the simulated looks at it, how the synthesis tool looks at it and use higher level construct for the behaviour model.

(Refer Slide Time: 42:50)



So even in this process there is a declaration region before the begin and the statement region after the begin and here you can declare you cannot declare a signal as in the case of architecture you can declare a variable now ok. We will see all the variable works, it works similar to you know like a sequential language like C program, but what is the hardware significance.

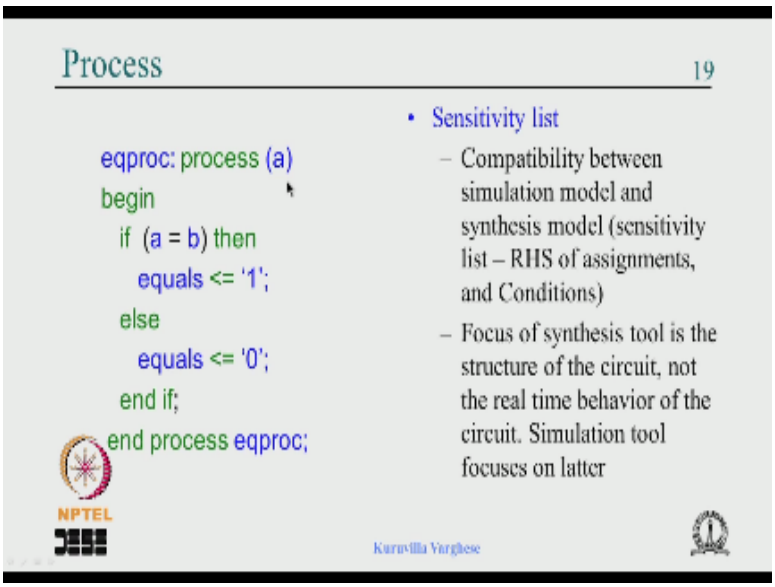
We will try to make out that how the synthesis to handle that variable we will see later, but I you can declare variable you can defined functions and procedures you can declare constants here in this kind of depletion region and this is a statement region which uses these construct, so let us look at the process is sequential body and when I say sequential body is a way simulator computer when anyone happens it goes from top to bottom once.

Synthesis tool is based on the bottom of behaviour you stayed within the statement reason there is nothing like top to bottom sequential, execution and things like that. And as I said the

process body has two parts one before begin, one after begin, before the beginning you can have variable and constant declarations, data type declaration, function and procedure definition.

In the statement party can use high level construct if then case 1, for loop and while loop, none of the concurrence statement can be used here like with select when else and some generate and all that cannot be used here and this cannot be used in a concurrent part of the code, outside the process, outside the functions of procedures, this cannot be used that you should keep in mind. So that is about process.

(Refer Slide Time: 44:56)



And we will see some kind of the issues with the process if you are not careful, so look at this this is the same process for the quality comparator, but if a and b we write only a ok, now it means that it is an equality comparator, but which is sensitive only to a, that means you have any quality comparator with inputs a and b. But if b changes the output would not change.

Maybe a was 4, b was 3, output was 0, but b has become 4, but output would not become you know 1. So that basic idea and if you write such a code and try to simulate it will show that behaviour it mean it will be sensitive only to a, but you know that given to a synthesis tool there is an issue, because this is supposed combinational circuit, you cannot have in a combinational circuit kind of 2 inputs which respond to only changes in one input.

If you have to have such a behaviour you need to have some kind of latches, which is you know latching the b input and keeping it you know kind of on the edge of a hands-on, you

can imagine a circuit but it was some kind of memory, so when you give such a code to synthesis tool and mind you as I said synthesis tool is not going to bother about even happening on the sensitivity list and so on.

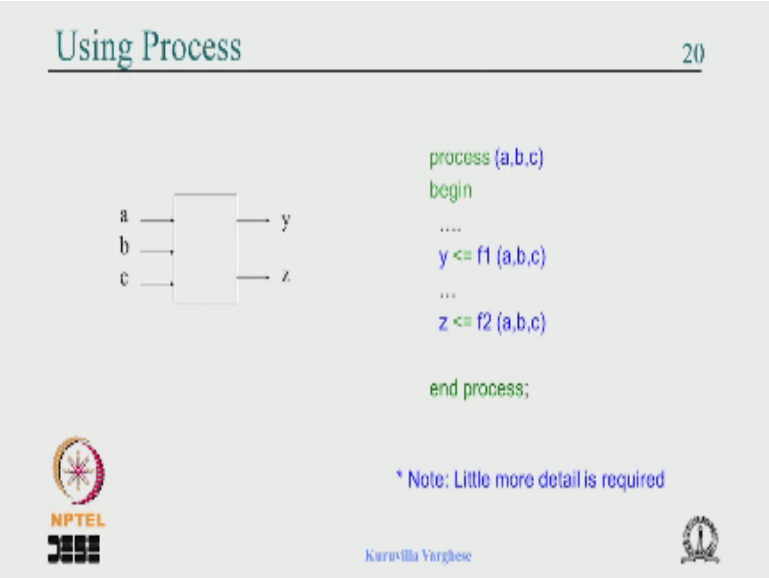
So synthesis tool basically looks at the code, the behaviour a code written, so if given to synthesis tool it look at this and try to generate and equality comparator, but given to a simulation tool will generate an equality comparator in simulator and equality comparator which is sensitive only a, so there is an compatibility between the simulation and synthesis when you write such a code.

And normally you are not supposed to write a code like that maybe you have forgotten something, so synthesis tool is not going to generate a circuit with sensitive only to a, it is going to report many a times saying that something is missing in the sensitivity less, that is what is going to do. So that is what is what happens either it ignores, and try to generate ah circuit using this kind of behaviour or it just ignore and you know want saying that something is missing in the sensitivity less.

So it means that you know it essentially means that you cannot use VHDL to invent a new kind of circuit, you know you cannot do that, you know you just play with the syntax of the VHDL and you are thinking that you are inventing some kind of new circuit that not going to happen, that is so kind of trying to discover some new land looking at the Google map or the globe or the maps and things like that.

What is in the map is already kind of discovered, it is already marked. So you cannot discover something new with some kind of work soft the VHDL Syntax that should be kept in the mind, many people do that many times some kind of the kind of write something and they claim that kind of new kind of the circuit is invented, but not possible. So keep that in mind, so this is a kind of an error which you need to correct. This is not a new circuit, but this is an error under the compatibility between synthesis and simulation model.

(Refer Slide Time: 48:50)



So let us move forward, now so the question is that do not get a kind of dot by this kind of keywords, it is very simple suppose you have a block where the input is a, b, c can be a single bid or kind of multi array of bits and you have 2 output y and z and you want use process to write this block it is very simple you write process and within the sensitivity less write the input a, b and c.

And as I begin using if or case or for or mixture of these you write the behaviour of the circuit and assign the output then you implement the circuit. So do not worry too much about the terminology like process and behavioral and things like that it is nothing you have a block, you have some input, write the input in the sensitivity less you see if k is for loops and all to describe the behaviour and find out food from the input.

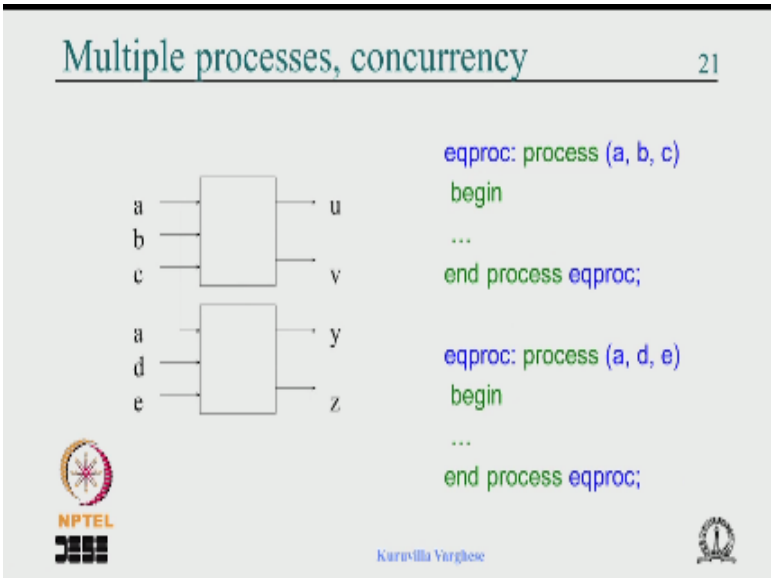
And we will see how to do that it is not that we are going to have a detailed look at this construct like When else with select if case and all that, how to write combinational circuit, how to write sequential circuits using this construct we will see that ok. But now for the time being you take it that you have a block of some input output you write a process with sensitivity list with inputs in the sensitivity list.

Describe the behaviour using the higher level construct, sequential construct, then you are implementing it, one point though I said so like a let us turn to the slide there is a little more detail when I say you just write the input in the sensitive less, it not completely correct, something is missing here, but we will handle that as we go ahead in the lecture.

We will see that as we go along because now mentioning that I will not bring clarity, as we go along we need to add something more we will see that what need to be added what kind of things need is missing here we will see that, but little more detail is required but this is kind of bringing clarity to the whole scenario because when you read a textbook which I be able model process if and.

More students kind of worry what is this all this about process sensitivity less, sequential execution. So when you say sequential execution b mean the simulator not the synthesis tool, so you have to make a distinction between simulator and synthesis tool.

(Refer Slide Time: 51:46)



So let us move to the second part of it so which now when you say you have two blocks like this say you have a 1 block where it takes input like a, b, and c and produce output like u and v and another block which takes the same a and d and e and it produces output like y and z. Now you can write a process for this like I am trying to show this process okay eq process maybe one.

Because I have used the same label twice this is not correct for this has to be corrected maybe there is a one here just to here. So I write process a, b, c and trying to write the code for this and I write some if then to assign u and v, we will see that later and you write a second process eqproc 2 process with sensitivity less a, d and e and I write the code for y and z. Now you should know that if an even happens on a, both process has to be computed okay.

That means like we have written, we have seen concurrency of the 2 statements earlier I have shown that we have written a and we have written expressions for x, y and z and we said that all are concurrent like when something happens everything has to be executed. Similarly 2 process are concurrent, if a changes both has to be computed in the same simulation time and the result has to be brought in here.

So when ask many like you have 3 concurrent statement as 5 processes something happened everything is concurrent like you have the same signals appear in the statement as well as in the sensitivity list of the process of when the simulator handles assimilation everything need to be computer at the same stipulation time, something changes is not like sequential language whole computer is not be done.

So that is how the currency is handled by the simulator but for synthesis tool if you look at the like the 2 processor which is in a kind of simulating like for synthesis told it does not matter it comes look at the quadroon here and generate the equation or the circuit for U and V, comes here look at the code for y and z and generate the kind of the circuit which is written for this block that is all.

So when you say can currency is mainly the simulator we mean of the can currency. So with that I try to I mean I like to wind up, so todays class we have looked at the different models of description and we have seen data flow model using concurrent statement like when else and we have also seen the data flow model using kind of equations and we have seen the when else is much better than this kind of you know the equations.

Then we have looked at the concurrent basically if you write 2, 3 statements the similar as an issue if the order is not correct it has to resolve concurrency, but synthesis tool does not matter looks at the description and generate the circuit, then we looked at the behavioral model, there is a syntax call processes and we use a sensitivity list where the inputs of a specified.

When something happens to the even happens on a signal on the sensitivity list, the whatever return behaviour is computed from top to bottom once and for synthesis tool it does not matter I know it describe looks at the description and if something is missing then the sensitivity less simulator show some behaviour and synthesis tool shows another behaviour

you are expect to write all the input signal which comes in the expression which comes on the right hand side of the assignment in the sensitivity less.

And we have seen how to use processes you write the at least you write the input signals and sensitivity less, why the behaviour using higher level construct, and little more detail is there we will handle it and multiple processes are concurrent with each other with multiple concurrence statement. So that is about concurrently as far as stimulator concerned, synthesis to look at the code and try to generate the circuit for the block.

So with that I wind up lecture today and in the next class we will see the remaining the model which is called structural model which is important in describing the hierarchy and we will see how the simulator handle this concurrent basically in the next class. So please revise today's portion, try to understand various concept refer to some text book and so I end the lecture here, thank you and wishing you all the best.