Digital Systems Design with PLDs and FPGAs

Kuruvilla Varghese
Department of Electronic Systems Engineering
Indian Institute of Science – Bangalore

Lecture-09
Entity, Architecture and Operators

Welcome to this lecture of digital system design with PLDs and FPGAs. In last 8 lectures we had a look at the advanced digital design. We started with an overview of the field or revision of the field, and given a brief overview of the current state of the art the basic what is happening in the field, then we started with synchronous sequential circuits, started with asynchronous counter etc, how to design a Timing Analysis.

Then we get got onto the design itself. We have seen the hierarchical design of a CPU, will see what is datapath, what is a controller, how to hierarchically design the datapath and using CPU as an example we have seen the behaviour of the controller of finite state machine, what is the structure of it, then we have seen Timing Analysis of it and how to design a control algorithm.

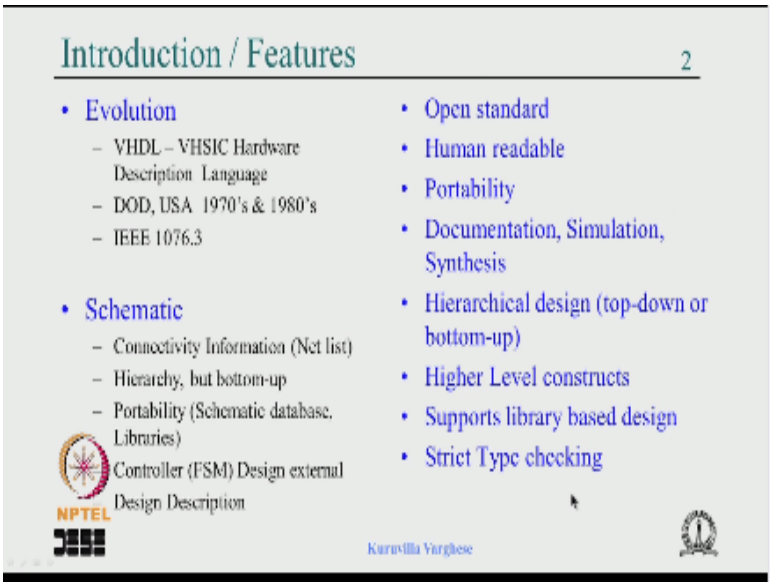
And finite state machine using state diagram and we have wound up the whole process with Case Study, somewhat realistic is a real life case study of data acquisition system. We started with spec and went through all the way like the partition, data path design, controller design, how to implement the in other state diagram, the next state logic output logic and all the way down.

And what is the part of the kind of the designer main part at what are what are the tools + designer and so on. But now we have some portion left in the advanced digital design but before proceeding I thought it is better we look at this hardware description language, so that when now onwards with me go with the digital design or anything PLDs or FPGA, wherever it is appropriate we can put the connected VHDL or the hardware description language.

That is the basic idea that is why we go. So now for quite a few lectures we will have the VHDL lectures with my plan is to at least complete the combination circuit description of models and sequential circuit test ranges and some demonstration using the tools available,

free tools available you know how to write the code out to synthesise how to simulate and so on. So let us turn to the slide of this VHDL ok.

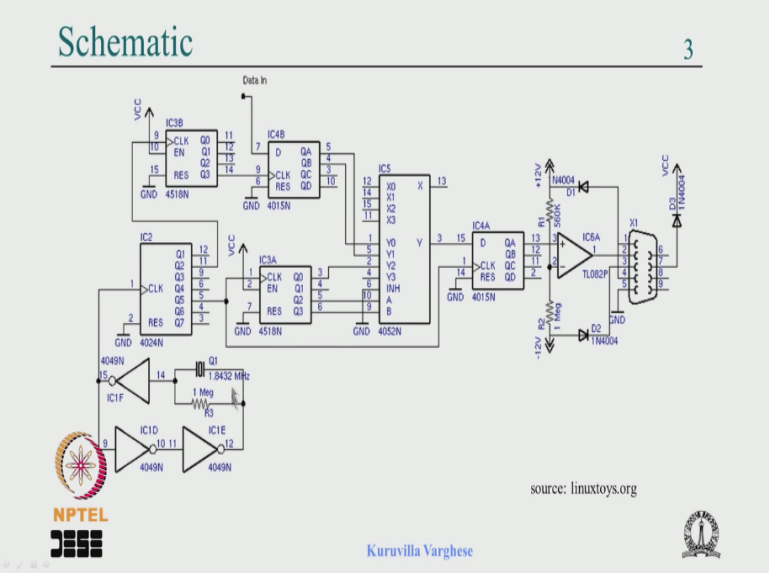
(Refer Slide Time: 03:17)



So let us look at the slide out the VHDL, the full form of the VHDL is VHSIC hardware description language that means at the very high speed in the greatest circuit hardware description language. I just started in 1970s and 80s by department of Defence USA for maybe I will give a brief on the contacts of that sometimes the contact make things clear, you do not grumble or complain why certain things are like that.

And it is an accurately standard IEEE1076.3, there are various years y know 93, 97, 2012, 2002 and all that. I do not exactly remember but then these are the various versions of the VHDL, essentially it started with some kind of limitation of the earlier design methodology which was using sounding call schematic capture ok. So I do not know whether you are familiar with the schematic capture.

(Refer Slide Time: 04:37)



But I will show a slide with which is taken from the internet of some kind of hardware digital mainly digital hardware but then you see this is kind of some flip flops are there, some Muxs are there, invertors are there and all that. A schematic would always involved some symbol of the IC you are using or the other Gate or the flip flop, many times in terms of the IC you are using.

Suppose this is a flip flop which is written IC 3D, so and there are pin numbers and it shows how it is interconnected ok that was the earlier days that was starting point of a design you could like you want an AND and invertor you put the inverter symbol which you mark which is a part of the IC maybe there are inverters in a IC, so our 6 invertors may be IC1A, 1B, !c, ! d, !E, like that the numbering goes.

And the PIN number you can see that is output is 15 pin number, 14 pin number and all that and people would put symbols and interconnect with the wires ok. So that was basically the schematic capture and when one designer printed circuit board will be an associate at package with it ok if it is in the standard library the package will be part of the library if the user create yes to create what is a package maybe old times or may times you will be make package.

So this was to be precisely created so that the PCB when one make PCB it is to size and the chip goes into the is a pinholes and the tracks alignments are so on, but basically essentially a schematic I would tell how is the interconnect or or the it is called Necklace ok, just shows

the symbol and interconnection between IC and it does not give you what is the function of the circuit.

Many times I have to refer to the data sheet of each IC try to understand internal in simple case gates in all simple but if there is a processor one has to read the data sheet of all the complete processor to understand a memory process of everything need to be understood and sometime a design description need to be returned saying that this is a particular IC, this is the function of that IC and so on.

So that was one big problem that it just does not convey anything other than kind of connectivity information, that was the first problem with the scheme capture and the second problem was that there were lot of schematic capture or the tattoo and everybody had many times a proprietary format that means the tattoo if you are using tattoo from my vendor A the files stored in that format may not be operable.

You cannot you know edit using some other tools okay, that was big problem a proprietary standards for fallout and many times this kind of tool did not support something called a black box design, go back to our hierarchy design of the CPU and initially at the level 0 P put a black box and set the clock and reset and read bar, write bar where the inputs outputs and so on.

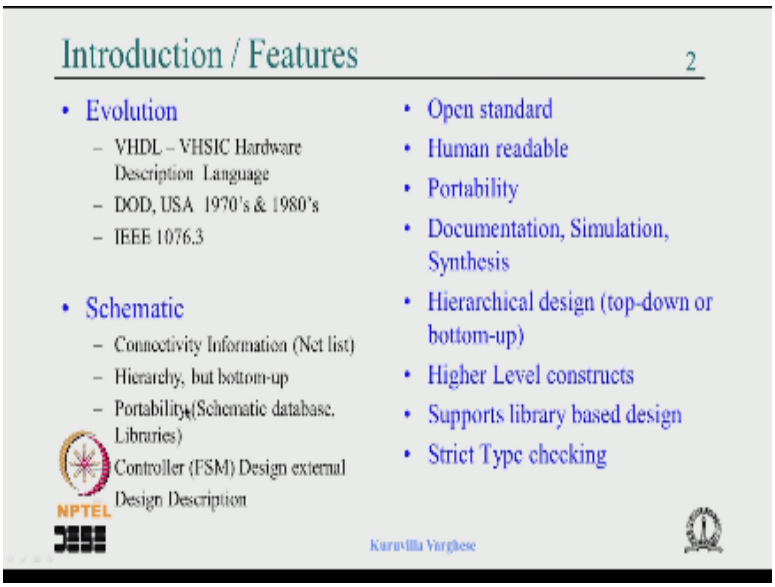
And then we have partition it to a level one kind of partition still it was black box you know there is a ALU, register program counter, ultimately when we broke down in 1 pieces it as it has come to some kind of gates and flip flop or some known blocks. So this was one difficulty with the schematic a hierarchical design was not possible or top down approach was not possible in this thing.

So this you that if you had to design a finite state machine, now when we describe the finite state machine we said you could kind of describe that in hardware description language and with tool will make the next year logic and output logic this was not possible with schematic capture, you need to do you know the state diagram you need to make the truth table minimise it and then implement those minimise equations using the gate and flip flops and so on.

So this was kind of external you walk externally and bring the design into the schematic capture and they also support some kind of simulation because there is a flip flop if the model of flip flop was available then you could simulate the whole circuit then model of all the chips used are available then using some kind of the later the behaviour of this could be kind of simulated.

But again this for proprietary like the when they will give some models but that was not compatible with the other tools and so on. There were right not that is completely prepared, there were standard for you know the cat file exchanges you know the schematic exchanges and the simulation models and so on, but not everyone was following in the work problems issues with it and there was a huge problem was that these were binary files.

(Refer Slide Time: 10:36)



If there is one bit goes on everything is corrupted and said that were the problems with the schematic capture now the department of Defence a lot of vendors making big systems and they have huge rack of cards comprising it system, maybe each subsystem was designed by a separate vendor and they all used different tools and put it together and to understand kind of coordinator activity was a big problem.

That is why this particular standard came up and they thought of basically documenting the schematic, know that started with documenting, so basically to describe in human readable language English language the description of the circuit that was a basic idea. So that it is portable like all the vendors use the same language to describe the circuit, so that one can read and understand.

That was a basic idea, so it started with documentation in a document it well, you know describe it unambiguously, so that the human can understand and there is it does not open standard human readable is portable and so on. Then late 1 people thought anyway we are describing the behaviour of subsystems a small circuits using some language, why not use that description to simulate the behaviour of the circle.

Still the design was largely done on schematic capture, but then the simulation came second ultimately when everybody started using simulation people started thinking why not generate, the circuit itself from the description ok. So these are the three uses of the VHDL language documentation simulation synthesis which is and which happened at different points in time, so you will see all these features in the language ok.

So there are documentation features you know you when you look at the VHDL people will you not see that the entity is or architecture a name is off and all that and people often complain saying that what does if and of and why the computer algorithm need is human readable thing, but you know you should remember that this is meant for documentation and easy way you can you stand it templates.

You can make your own template or copy paste old template and so on, so do not complain that much about it is a little bit verbos language so do not complain about that part of it and similarly there are when you come back that there are you know that the syntax for simulation there are some contracts in the language which is only used for simulation, the no point in using it for synthesis ok.

So this should be kept in mind is not at all like anything you write can be synthesize, you know there are different there are something made for synthesis something went for simulation, some part of the language is nearly made for documentation. So do not complain too much and the language support hierarchical design are you can do the top down design with a black box.

Absolutely no issues you can do bottom up whatever and much to the to the convenience there are higher level constructs you know you can describe the behaviour not just Boolean equations but you can use a higher level construct like if then, case when loop, so it kind of

give up power to describe the language at many levels at a very detailed level abstract level, as we go long we will see what is how we can efficiently use this contract to describe.

And what are the best scenarios to use at and construction so on and it supports library base design that means that you can put suppose you have designed a counter a generic counter you can put that in a library and late on you do not have to redesign that, so any anything which model or generic can be put in the library similarly various operators various functions procedures everything can be put in the library.

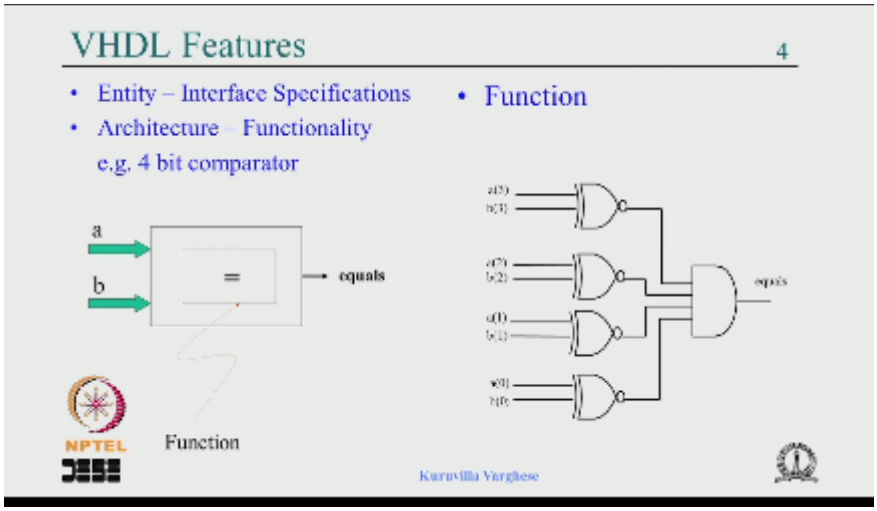
So that that can be reused, so the great help you are in an organisation in a particular business and you are making designs over the years. So you can have a good library of the various module SAP system components your using and so that when a new design need to be made you can quickly make it using already well design component interconnect them or modify them quickly and put it together.

So that is a big advantage and 100000 or you know or feature of the VHDL is that it is straight type checking t means a data type is strict you suppose you declare something called bit, bit is something takes 1 and 0 and there is a another kind of a data type call standard logic, ok we will see that but that also support 1 and 0 and many more things, but in VHDL suppose there is a signal which is of a data type bit.

This cannot be assigned to the something off data type thing that looks like a restriction but is a very good thing you because if you do not know what you are doing because you are in the connecting wireless signals and if you are not careful like you declared something as an integer of some range and there is some kind of bus which is 8 bit and you try to connect this integer that kind of bus 8 bit bus, maybe that your range of integers is more than 8 bit.

And then there is a confusion like you end up 12 bits and how to connect this 12 bits, 8 bits and if the tool design that least significant bit will be connected then you are in trouble. So that is why this spec type checking enforce a good thing, because we are making hardware not make mistake the VHDL enforces spec type checking and there are that you cannot assign a bit for standard logic you up to forcefully convert the data type then you are aware of it and then you will take caution that is why it is provided.

(Refer Slide Time: 17:31)



So let us move on to the slide and let us look at the main design unit of the VHDL. So the VHDL design or any block has two parts when you describe , it has 2 parts any component as an entity which is nothing but the interface specification and architecture which is functionality ok this and we will take an example of a bit comparator. This part is taken basically from the reference book Kevin (()) (18:03) the starting part is taken from there.

So I say you look at an equality comparator, so you have A and B which are 4 bit and there is a single bit signal call equal when the numerical value of A is exactly equal to B that means if it is say 1,0 1,0 and B is 1,0 1,0. This will be 1 that meaning of it, so NDT means a name for the block some name, what are its input port ok. So you say input part A and B which is 4 bit and what is a data type is bit or standard logic vector and so on.

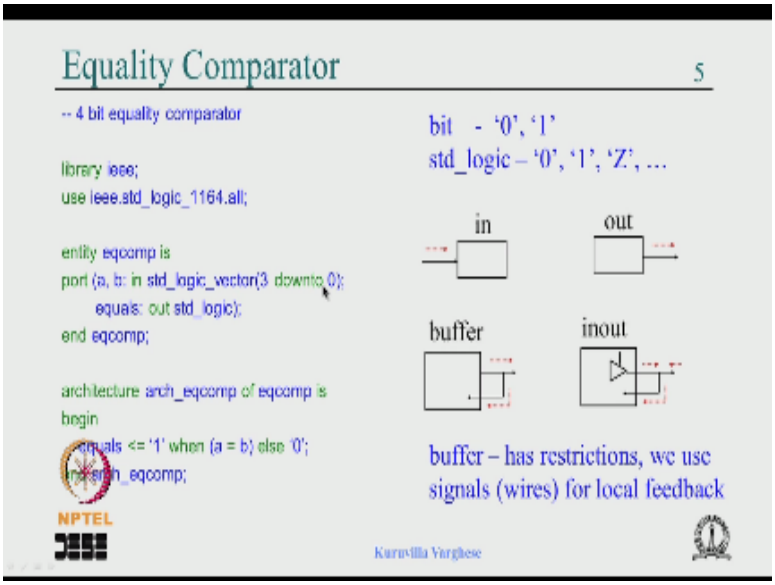
So and similarly what is output how many bits are there, is a single bit or multibit and what is a data types, so that comprises reality. So give a name for the block, what are the inputs, what are the outputs. So you have to clearly say what is the name whether the input output and so on and what is a data type. Now comes architecture having define the NDT you define what is inside.

The architecture is nothing but the functionality, so you have to define what is the function of the circuits in terms of the input and output, that means you somehow you say that equals will be 1 if A is equal to B in some there are many ways of describing it. But that is the functionality. So you describe the functionality of this blog equality comparator in terms of the input and output ok. So you say output, how the output is a assign from the input that is the basic cracks of the VHDL, NDT and architecture.

NDT means the name input output, basic the data types and architecture means function in terms of the input output and you know the circle, you know the when you come to the basic implementation we need to have each bit you know suppose you have a A3, A2, A1, A0, B3, B2, B1, B0, you can see that for this number to be equal. A3 should be equal to B3, A2 should be equal to B2 and so on.

So we use and exclusive NOR gate because exclusive OR gate as 01 and 1,0,1 but here your 00 and 1,1 as 1. So unless all these are equal this one by one and if all these bits are 1 then we have design gate is making it 1. So you know the circuit for exclusive OR gate and exclusive NOR gate and one AND gate is a function. So that is the VHDL way of describe main kind of component kind of sub description NDT and architecture.

(Refer Slide Time: 21:12)



So let us look at this equality comparator VHDL code straight away and tried to kind of see how to write a VHDL code and what are the basic components and so on. So say when you like I have returned here 2 dashes and 4 bit equality comparator that means this is a comment. If you start at dash anywhere you could I put 2 dash here and write something say anything after the dash is a comment.

And you know that there is no way to kind of make it no nest with multiple lines and not a big problem earlier when people use the keyboard that was a big problem, now suppose you have 10 lines to command it. If you have to put everywhere the big problem nowadays with

your eyes very easy you select everything and click on an icon and then everything become commended.

So this is a comment character only for a line, you cannot list it okay and the next thing is interesting it say ok now that that the I have shown that code in two colours basically green and blue and whatever is written in the green colour is a VHDL keyword and rest is what we are writing ok. So that is to make out so, like library is a keyword of the VHDL use is a keyword of the VHDL and ok.

So here we are saying we are going to use library Ieee, when there are Ieee standard library is severe we are saying that we are going for this NDT and architecture, we are going to use library Ieee. So when you declare that this Ieee libraries are visible only to this entity and this architecture. Suppose in the same file you write another entity and architecture you have to and unit used a library then you have to write again library before the entity and architecture.

And just write in the library that makes a library visible to the code but not the packages library as lot of packages within it, so the hierarchy of library, so you have library your packages and within the packages have components functions procedure data type and so on. Ok so that hierarchy and we are going to say that that we are going to use particular package within this library call standard logic, standard underscore logic underscore 164.all ok.

So we are going to use this particular package for this entity and architecture that is the meaning of this and now onwards we are going to use the standard logic library, the reason is that if you look at the VHDL inbuilt data type is bit, okay as for as the logic is concerned and it supports only two values you know 0 and 1 and this is a great limitation and for digital design we just cannot do with only 0 and 1.

We need many other things, at least you know that we have Tri-state gate which we use and you want to use a and that describe saying that some output need to be Tri-stated, so and that is described in a standard underscore logic and that is a kind of data type which is used which is available in this particular library and package. So that is why we use this. It support you know the data type like 0,1 values like 01 Z.

And you know there is many times in minimization used do not care, think support do not care and all that many more we will see what are the values in the standard losing data type but for the time being understand that at least support a Z which is represent the Tri-state is very much required for the design. So now onwards all the code we will not get into bid we will use the data type, standard logic that is what it shows.

Now this comes the entity description, so the keyboard is entity give some name, it does not matter and this is what I said is a kind of for documentation and you say end this particular name if you call ok. So that is a body of the entities, so you start with the keyboard entity a name is and end that mean that that is kind of that is the body of the entity and within that we have to describe the inputs and outputs.

And keyword used for that is port you open the port with the parenthesis left and right and semicolon, you can see that all the kind of syntax is terminated by semicolon. So if you describe something a complete entity then you put a semicolon complete port then you put a semicolon everything every statement, every kind of the body is terminated by a semicolon. Now you see this is the input specification and this is output pacification.

So we have two types of input to input A, b. So if it is of the same size, same data type which used in the same line, so here a, b, the new product column this is the name, now is a what is the direction of the port it is in it means it is an input and standard underscore logic underscore vector 3 down to 0 that means that the a is a 4 bit vector which is a of 3 bracket you know 3 like we have seen here.

A(3) a(2) and a(1), a(0) similarly b(3) b(2) and b(1), b(0) ok, that is the meaning of the moment you say standard logic vector 3 down to 0 that means it is a(3) a(2), a(1), a(0). There is another syntax you can say 3 down to 0 you can say 0-3 it is possible to write like that and there is a difference we will soon say what is the difference and let us come to this equals which is the name and the mood of the day or the direction is out and the data type is standard logic ok.

So when you describe the port you have given name, you have to tell the direction, you have to tell the data type and now when you say there has to be clarity with this kind of direction of the code, so when you say in it means that it is an input okay. It cannot be treated as an

output that means that when you write some assignment like here we are writing assignment = get 1 when a=b.

So a and b are the inputs in any assignment the input can come only other right hand side of the assignment ok that is you should kept in mind when you have something different as there is a direction in it can come only on the right hand side of the assignment and their similarly there is out and out means it is an output port something drives you know the reason has some kind of JK flip flop is output is driving a signal which is outside.

And when you declare something is out like = that can that should come on the left hand side only. You can alright I know something like Z gets 1 when to something, now you cannot write on right hand side ok, but that could be a restriction and you look at this kind of thing like we have some many times and output top is taken from the output and used as an input to the sum circuit inside this is quite, you know you have something coming like some gate is driving and output but that output is going to the output pin.

Also it is used as an input to the father suck if you know within this block ok but by definition it views out such a thing cannot happen with VHDL. So there is a roundabout there is a kind of way out that is declared as output as buffer like using buffer you know standard logic, but this has a problem because as standalone it is ok but when you are multiple components buffer it cannot be connected together.

You know there is a there is an issue with buffer, so I do not think that you should we will not be using this particular direction of the output signal at all and what we are going to do is that there are internal signal, you know you can use signal so what we're going to do that we will do normally in our codes we will declare a signal here which can you know which can definitely connect to the summer output and can connect to some input.

And ultimately the signal we cannot sign it to some pin, now that is what we are going to do in our code I so literally forget about this particular direction of the port buffer, now there is one other direction specification which is nothing but in or out or an IU. So now that is not a same as buffer in the case of buffer this still as an output, the pin is an output, but there is a tapping which is used in that.

But when you say something is in out it literally means sometime there is a definite have Tri-state gate and when the enabled is 1, this circuit will drive the circuit which is outside ok and when it is cut off when it is not disable this part is inactive there is nothing you know it is stated and this can be used as an input pin that means literally something can drive the pin ok. So there is a difference between this structure and this.

And you should not use in out for out or buffer ok and I have seen sometime people you know use out and something goes wrong and they somehow find that if you write in out that can be circumvented because of some kind of simulator difficulties maybe at the appropriate time I will tell, people tend used in out where in out is not meant to be used ok, so that is not happen, you should not used input in place a buffer.

That in out is used only when a pin is used as both as output and input unnecessarily there has to be Tri-state gate without with no wait use an IO pin, now that is the kind of various direction in, out buffer and in out and as I said forget about buffer we will use the wires I will show the proper time and standard logic buffer is a repetition of the standard logic like you know this is a for bit standard logic is a standard logic vector and as I said you could write three down 0-0 or 0-3 ok.

There is a difference when you write 3 down to 0 the most significant bit is 3 and least significant bit is 0, when you write 0-3 the most significant bit is 0, then whichever comes on the left hand side and the least significant bit is the 3 which is on the right hand side, so the rules symbols whatever you write on the left hand side is the most significant bit and whatever you write on the right hand side easily significant.

Now you might ask what is the big deal you know what does not matter ok. It matters if you studied kind of the processes from the Intel actually this is a mystery you know behind the Intel processors used to treat suppose your database which is going from 16 like 16 bit data bus, D15 was treated say the data 15 was treated as the most significant bit and D0 was treated as the least you can be done this was called little engine.

That means the small number will end the you know is the end part and then the Motorola which was a kind of a competitor to Intel, when they came out with this processor subsequently that is become free scale, but when they came out the numbering was kind of

opposite the b is 15 was LFT as D0 was a must be ok. Now you can imagine this is something to do with this 3 down to 0-3 I something to do with the kind of the bit order and byte order and so on.

So if you are using little end then you should use terminology like 15.0, we are using big end then you should use 0-3, so you may even end up with you are designing a chip on one side your connecting to some other chip from another vendor which has a bus in order and the second side has a bus which is in the big in order, then you should appropriately a kind of define this copy.

Otherwise things can go wrong, so you should keep that in mind is not that I know is not your convenience but if you are designing a complete chip in your own which is not be in the connected anywhere then you can choose what you want to do like whether the you adopt a little like big in kind of style for your multibit signals that has been designed, so that is all about the entity.

So we have looked in detail what is entity, what are the directions, little end and big end in the case of multi based on so on, now comes the functionality which is defined as architecture, so you say architecture which is a keyboard give some name of underscore if you come that's my style of doing at of this particular entity. So you whatever the entity name is is written here.

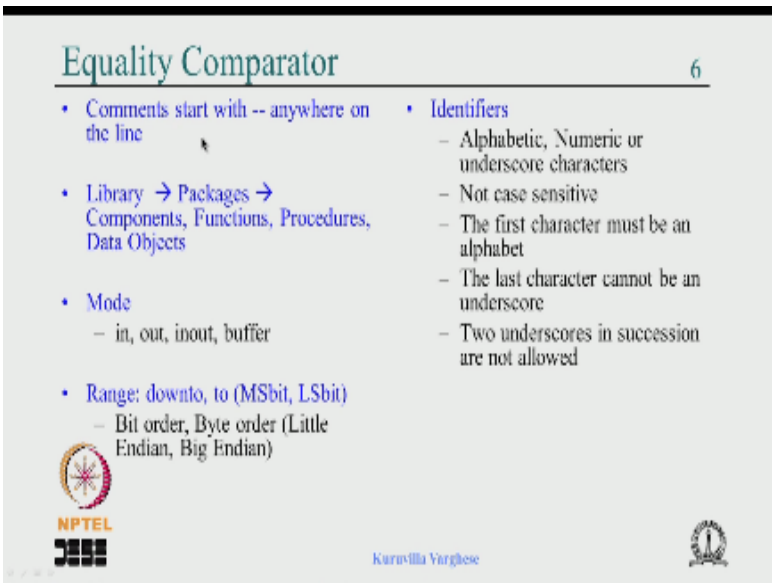
So this is the kind of link between the architecture using architecture name of which name have to say is ok. Now you say big end that is where you describe the functionality ok and before that before the begin you could declared many things, may you declared components we can define functions procedure and so on ok. So we will see what could be done before that.

And when you write this statement I we will see what are the statement and so on. So this is an output signal and this is a assignment operator which is similar to which is exactly similar to less than or equal to the VHDL, also use same for the less than or equal to so depending on the contacts, I know the meaning of that is right, so one, now 1 is written with quotes left coat and right code because in the library the standard library in the values are define with quotes like you know one with the left coat and right coat.

When a=b else it is 0 ok so = get one when equal to be else gets 0 ok, very simple the description is over. We say n this architecture whatever is mean, so that architecture is over. So that is the code in actually you have a comment, you have a library you are we use package kind of construct, your entity with port mainly within the entity port direction the data type the multi bit decoration.

Then the architecture before the begin the reservation in the statement region you write various statement then the description is over ok, that is in a nutshell a kind of the basic VHDL code and you can use various keywords, various names and let us move on to the slide summarise whatever I have said.

(Refer Slide Time: 39:06)

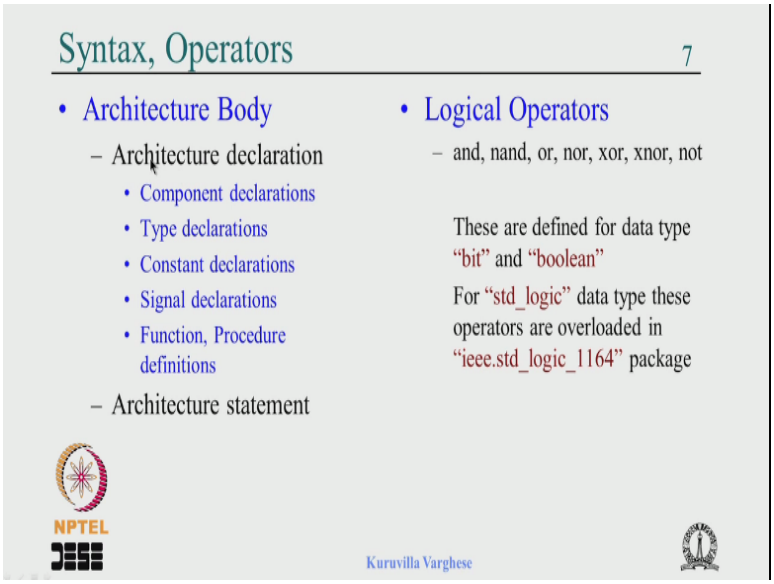


So I have said come and start with anywhere on the line and library has hierarchy library, packages and it contains component functions, procedures, various data object, data types and things like that then you have different mode of direction in out in out and buffer, we said this is very limited usage and in out and out is different or buffer is different, then we have down to and 2.

So I have to worry about little Endian and big Endian bit order, byte order and things like that and when you define some name you can use alphabets, you can use numbers, you can you underscore and it is not case sensitive first character should be an alphabet, the last character should not be underscore and you should not have to underscore in succession and I do not remember the exact number of characters you can use for the maybe it is 32 characters.

You can use you can check with us and it because standards keep changing but for practical purposes your normal and names you can give quite a long names that is not be problem but better to check with your with VHDL latest standard and tool compatibility what the tool support maybe you refer to the VHDL, latest standard but the tool support the earlier standard then you will be in trouble. Then you should be careful with that so that is in summary about the code.

(Refer Slide Time: 40:43)



So let us move on the architecture body, so you we have said that in an architecture before the begin you can declare many things out there is lot of things you can declare that is what is shown here the architecture of 2 parts before the begin you can declare something, after the begin you can write the statements, you can describe the function. So what all you can declare is that you can declare components.

Components are in like a quality comparator gates of flip flops of multiplexers and decoders all that is components that type data type like standard logic with Boolean and things like that and constant like you have a bus would you know define as a constant like size of a it, signal we have described like you want to interconnect kind of two pots or output of some block with an input of another block then new signals.

You can also use functions and procedures for the time being we will keep it aside but then you can literally defined functions and procedures not declared you can literally define in the architecture declaration region which is visible only to the architecture statement region ok.

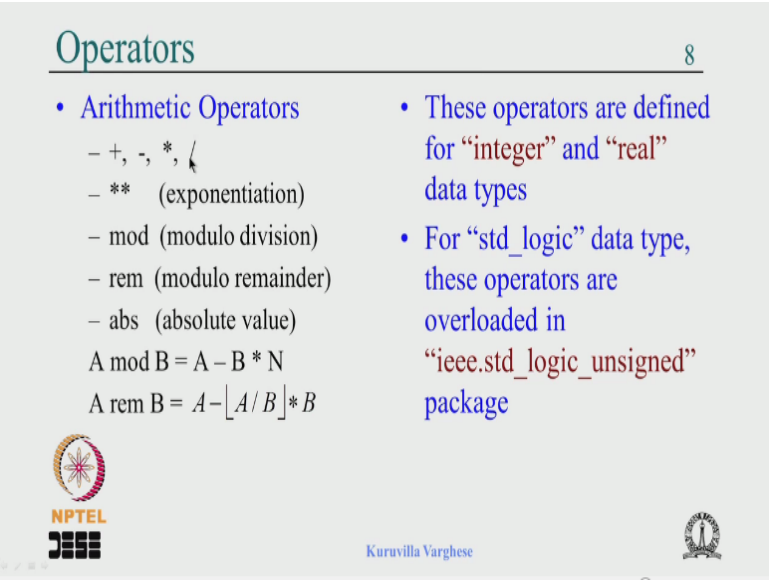
So after architecture statement is where you put all the description in terms of various construct.

So let us turn to the logical operators, so you have all the logical operator and AND or NOR or XNOR and NOR, but the trouble with the vehicles are the basic operate , within VHDL support only the bit and Boolean ok, but as we describe in the in the first light we are not going to use the bit because it is respected by 1 and 0, we are going to use the data type call standard STD underscore logic we call as standard logic.

So STD underscore logic is called standard logic and this is but you do not have to worry about the logical operators for this is data type because in this particular package Ieee standard logic 1164 package with you have used in our code the first example code this logical operators are overloaded, overloaded means whatever was written for the bit is re written for the standard logic data type.

So the moment you declare use ieee.standard_logic_1164, but all means we could use all that and all that here in the statement we can and the all means everything in it like that means the entity and architecture following should use constructor can use whatever there is within this particular package, that meaning of all. So that is a logical operator, then you have arithmetic operators.

(Refer Slide Time: 43:59)



You have the plus you can I add minus multiplication integer division exponentiation because many times we walk to the power of 2 then you suppose you have a data bus which is it but 8

bit, but then you know that it can go take values from 0 to 2 raise to 8 -1 ok. So where these kind of exponent station operators are useful than ever modulo division kind than the absolute value there is a little confusion with this terminology like mode is different from the computer science Mod.

The more definition is a A mod B is A-B-N like suppose you say 13 put 3 that means you know that this 13-3x4, so which is call the highest number input before it crosses over, then you get 1 ok, so if you know that when you do a modN the result has to be 0-N-1 but there is an issue with this kind of the expression because it does not work well with the negative values.

Because you know that at the modN has to map any number to 0-N-1, so that is why does rem operator is given to which is it actually this is equal into the computer science definition of the mod, A rem B is nothing but A=the floor of A/BxB ok positive numbers it makes no difference because you say again 13 mod 3 it is 13-13/3 integer division is 3 then kind of the real division will give you 13/3 will give you some you know 4 point something which is the floor of that is the decimal part of thrown off.

The fraction ball is thrown off, so you will get 3 and 3x4 is 12 and 13-12 is 1 and you get it correctly but if you say -13 then you will end up with you this confusion but with this formula but with this formula there is no issue because -13/3 you will get -3.7 something the floor of it is the lesson number so which is – 4x3 is -16 and you have -3-(-16) you get +3 ok you do not get a one but because you are so it is 13 like -13/4 is 3.4, 4x3 is 12.

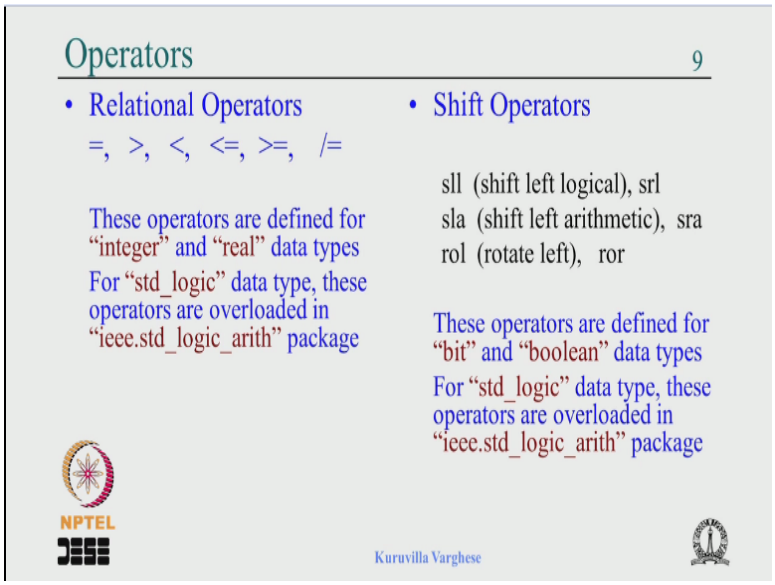
So this is -12 minus, so – and - + 12 and you will get you will end up with a with a positive number that is the of head are you can work out with an appropriate example but very important thing to note is that all these operators are defined internally for the data type in integer and real, integer is the integer as you know, the real is the real numbers on the kind of floating point data type this internal operators cannot be you will not work for the standard logic data type.

But if use standard logic unsigned package in the used things like you say library it and you say use it Ieee.standard and unsigned that all then you can use plus minus multiplication division and all that mind you if you write a code with this with the standard logic it might

work for simulation in and out from the size and generate a proper circuit for you have to keep that in mind everything does not work.

Sometime we have to write your own low level design for some of these to work or give some rudimentary circuit which is not that you want and things like that we have to keep that in mind.

(Refer Slide Time: 48:31)



And let us come to the relational operators. So relation operators are equal to greater than less than less than or equal to greater than or equal to and not equal to ok. These are the relation operation operators you say when equal a=be or if a less than b and so on and you see that less than or equal to operator is same as the assignment operator. So depending on the kind of contacts use the tools will in for what is the kind of operator and will find it do not worry once again in the only these are define for integer and real for standard logic data type.

This is overloaded these operators are overloaded for standard logic function in the package Ieee.standard logic arith ok. So in principle like if you use three packages like Ieee standard logic 1164, Ieee standard logic unsigned and Ieee standard logic arith you can do many things with the standard operators at least you can work with the standard losing data type.

So keep that in mind and when you come to shift operator you have logical shift, arithmetic shift and rotate. So you have shift left logical sll. So that is just left shift, shift by logic symbol right shift and shift left arithmetic and shipwright arithmetic work with 2 complement

number so into is complement number you know that the most significant bit represent the MSB or the most significant bed.

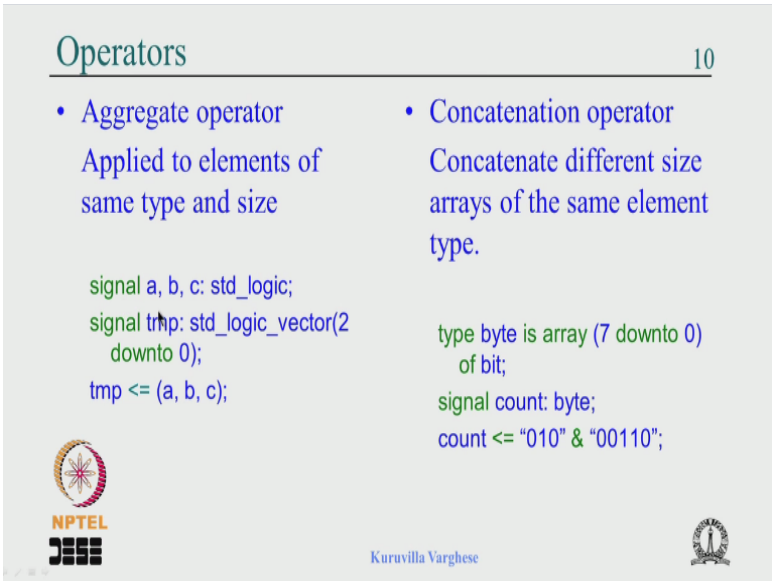
And that is a sing bit ,so for negative numbers will be 1 and when you extend suppose you have an 8 bit number with the sign because one when you convert this into 16 bit then all the numbers starting from and number 8 bit 15 bit has to be one so when you do a shift left arithmetic this sign extension will be automatically taken care that the meaning of left shift or arithmetic shift.

So if you do some arithmetic using the shift operators like you know that shifting left is like multiplying by 2 shifting right is like dividing by 2 and if you are working with unsigned integer then the shift sll srl like logical shift work but if use arithmetic like views kind of signed integers then you have to essentially do the ship Clipart American flight automatic to preserve the sign otherwise things will go on and you can work out.

You know you take a 4 bit number work out the negative numbers, you work out the you know if you extend it into it but make sure that you get the same value with the sign extension that should convince you and that will you will clear understanding of the 2 to 7 numbers you can you know work then even the decimal numbers and how this necessary but that bring clarity.

And once again shift left arithmetic and shift left logic and arithmetic all these are define for the bit and Boolean data type, for the standard logic data type this is overloaded in the standard logic arith package. So that is what I said you use these three packages most things will work you know standard logic 1164 and arith.

(Refer Slide Time: 52:31)



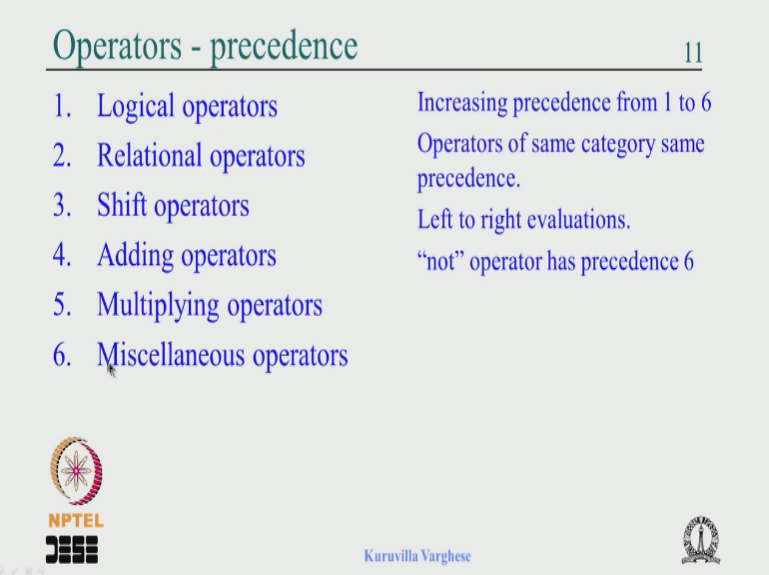
And you have an operator call aggregate operator essentially it is shown here suppose you have a signal like standard logic signal with a, b, c these are single bit single ok now I am using bit do not confuse bit I mean the real bit of the digital system not the bit data type of the VHDL ok. So here we are trying to make it is combined a, b, c into a 3 bit kind of the bus.

So I have a signal is a keyboard is the name which is of type standard logic vector 2 down 2,0 that mean just 10.0, now we say temp is nothing but assign in the brackets is a, b, c that means tmp 2 is a, tmp 1 is b, tmp 0 is company So this is call aggregate operators and aggregate individual signals which can be single bid multibit to a bus but the restriction is that these elements individual elements be of the same data type and should be same size.

It is not a can be 3 can be to bid and this is single but it has to be single bed to bit together but even better flexible operator is a kind of concatenation operator which will work for the same data type for different size you can see here we are declaring a data-type new data type which is keyword is typed bite is array is a keyboard, it is a array of single bit that is the meaning of it 7 down to 0 of bit the bit is.

The byte is nothing but an array of 8 bit array of with is the meaning of it and we say signal count is a bite and we say now the byte is an 8 bit kind of signal, it is count get 0, 1, 0 and 0, 0,1, 0, so this is a 3 bit kind of signal and this is a 5 bit signal which is combined it is and is much more flexible concatenation operator that I passed by saying but it will join things together very useful in digital design you have some kind of some parts of the bus come from different places which is come into the to make a bigger bus and things like that.

(Refer Slide Time: 55:13)



So that is very useful these operators and there is a president of the operators so you have the highest precedence is 6 as you go down 654321 one of the lowest president, miscellaneous operators are highest precedence and multiplying edition shift relational and logical operators and not operator as president 6 in when you write the same precedence operator from left to right it is evaluated left right.

But it is very difficult to remember all these I suggest you remember the kind of use the brackets possible it is good for you know understanding me when u p y you write the code somebody read it is very easy to understand. So maybe we will stop here todays lecture, so we stop with a VHDL how did it kind of involve, what was the basic contacts, the department of Defence has trouble with the schematic.

So it started with documenting the news for simulation synthesis in human readable it works with the computer language support, hierarchy support, high level constraint library base design and so on it and open it please send it to which over tool does not matter now the different tool when does everybody uses the same language and we have seen basically and entity and architecture in the interface architecture as a function.

Then we have seen an example code we have looked at the entity code the direction in the moment the data types bit order byte order then architecture declaration reinstatement region a statement and various operators. We have seen the logical operators and their particular data type question in logic and which are the library is which is used for the standard logic.

And we have looked at the arithmetic operators we have made found what is the difference between mod and rem then you have we have looked at the relational operators as to special operators called aggregate and concatenation operator. So now in the next class we will see a little bit how to what is the design tool floor you know various kind of different models of description and so on ok.

So I suggest you look simple to start with, so please go back brush up understand it so you can refer to some good books to get a grip on the on this particular VHDL, there a lot of good book as I said earlier maybe you use some book with synthesis as the emphasis, so that is that I wind up this lecture wish you all the best and thank you.