

**Synthesis of Digital Systems**  
**Dr. Preeti Rajan Panda**  
**Department of Computer Science & Engineering**  
**Indian Institute of Technology, Delhi**

**Lecture – 25**  
**Introduction to Timing Analysis**

We will take just a brief look at SPGS. Just to understand a little bit, how some of the algorithms for synthesis may change, if you use a different kind of target technology, just a brief definition; before we can talk about how the algorithms would change.

(Refer Slide Time: 00:40)

**Field Programmable Gate Array**

- “Field Programmable”
  - Customised by designer
  - vs. “Mask Programmable”: customised by foundry
- Customisation process simple/cheap
- FPGA chips produced in bulk
  - independent of functionality

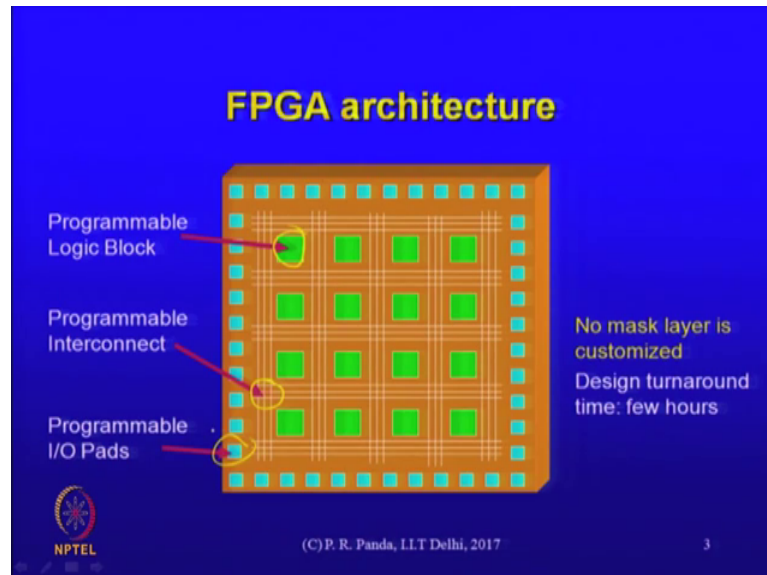
NPTEL (C) P. R. Panda, IIT Delhi, 2017 2

These are field programmable; meaning that the hardware is customized by the designer, it is customized in the field as opposed to mask programmable which means that it is customized at the foundry; what is this customization? Anyway you start off with a blank wafer irrespective of what the design is going to be and then you go through a various set of steps determined by a mask in the foundry, to customize the wafer into a functional, but here; we are saying that this functionality is being customized at the sight of the designer by the designer himself; that is what the field here means; it means that it is being done by the customer himself.

This customization process is supposed to be at least relatively simple and cheap and the FPGA chips themselves are produced in bulk, of course, it is independent of the actual

functionality to which the FPGA is going to be put much like a processor that is produced in bulk and is independent of what software is going to run on the processor.

(Refer Slide Time: 02:16)



Some very basic ideas about the internal architecture of (Refer Time: 02:22) of an FPGA; lots of things are programmable here; that logic block will take a little bit deeper look into what it might consist of, but it can be programmed to do different things that interconnect could be programmed to realize different connectivities and finally, that IO pad could also be programmed into an input or output and so on. So, there is a lot of things that can be customized in there. So, ultimately ultimately; you have a chip that can be customized to do very different things.

Student: (Refer Time: 03:09) software settings (Refer Time: 03:12) going to the logic connection.

Yes, the logic is being programmed meaning that now it can do some piece of logic, but in a reprogrammable architecture which is what this modern FPGAs are you could change that function to something else, the hardware function is being changed to something else as opposed to a processor in which the hardware is not changing, it is the memory into which the programmed instructions are being written and we are reading those instructions and interpreting them in a hardware; that is not changing. In the FPGA the intention is that the hardware is changing whether it is changing at runtime is a different

matter, but at least from one customization to the next customization, one used to the next use it is changing.

Student: (Refer Time: 04:11).

Yes, this partial reconfigurability is a feature in modern FPGAs, you can do it. So, essentially, we are reconfiguring means that the configurations also it has to be dynamically accessible somehow. So, for some time, you run it on one configuration and in later stages, you can run it on a different configuration; modern FPGAs do have that facility.

Student: (Refer Time: 04:37) FPGA is giving me a flexibility kind of (Refer Time: 04:41) designs. So, the kind of (Refer Time: 04:43) logic (Refer Time: 04:44) will be much more than an ethic, let us say if I would have an let us say for a particular functionality (Refer Time: 04:53) generic law ok, you can make (Refer Time: 04:55) this basically re programming so on, FPGA, if I realize my logic, let us say only subset of some logic is being used of the complete general logic, then R area power at timing analysis equally good as would have been on the (Refer Time: 05:16).

All the parameters will not be equally good, there is a price that we have to for the pay for the programmable hard ware that is obvious, right, if you design an AND gate by hand as opposed to some piece of logic which can be programmed to be an AND gate today or an OR gate tomorrow, then you have to give up something, you have to give up area because its capable of doing multiple things as compared to a customized AND gate, it would be more expensive. Similarly, there is an impact on timing on power on everything, yes. So, there is a price to pay for.

Student: (Refer Time: 06:01).

Yeah.

Student: (Refer Time: 06:02).

Yeah. So, yeah there is a price to pay, but it is a powerful additional feature that the hardware itself is programmable.

Student: (Refer Time: 06:13).

Yes, it is as opposed to for example, a simulation if you were to synthesize the same design into an FPGA; just to do an emulation, the chip is not being sold as an FPGA, but the emulation is being done on the FPGA that can be orders of magnitude faster than the simulation.

Student: (Refer Time: 06:35) powerful tool for testing.

Yeah. So, the emulation is that it is a much faster simulation you could fix a lot of things functional bugs could be caught, it is not as fast like we said, we are giving up on timing. So, you will not get the same timing that you are ultimately customised ASIC is giving you, still it is in like a 1000 times slower, it may be only 10 times slower; 5 to 10 times slower, but that may be enough to do a much more comprehensive and extensive validation of your system, before it goes into production, a lot of very powerful utilizations are there of the programmability feature.

Student: (Refer Time: 07:20) when people use let us say simulation on FPGA or simulation in the design of the chip.

People do both early stages, you might do simulation, but it is very common that after you are fine with their simulation result, you might actually go for an FPGA implementation of the same thing. So, that for example, if you are designing a processor, you could actually bring up an operating system and applications and so on because the speed permits you to do. So, in a simulation, you are hopeless mostly, you cannot do very serious applications that run for hours together in real time which would become once if it is on a.

Student: So, when we are (Refer Time: 08:06).

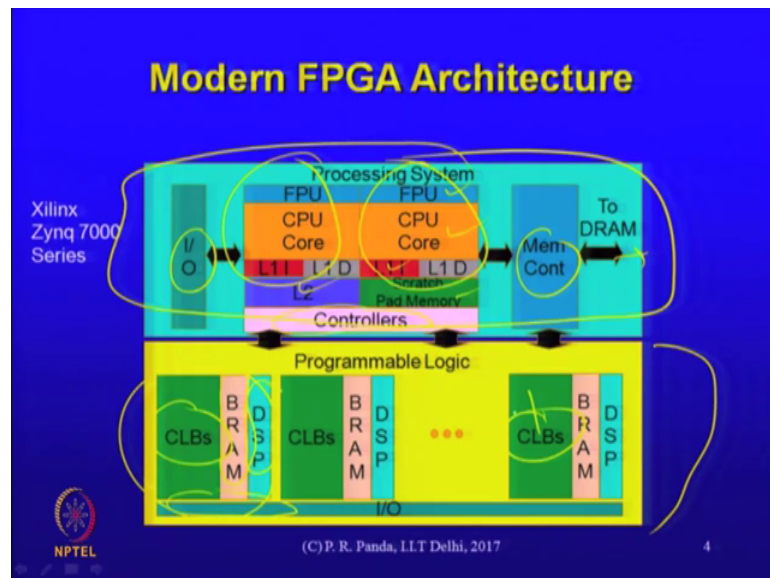
You to be as close to the chip as possible higher abstraction, you would always do of course so, simulation does not go away you would do all those abstractions even before, but you are going into more and more details of the design. So, the simulation is getting more and more expensive you would like to simulate everything your logic level, netlist is being simulated your layout is being simulated, there is limited amount of simulation you can do, but you would like to do that simulation FPGAs do permit a lot more aggressive validation than you would otherwise could effort.

Student: Here, it is for forming emulation is that I need to have my (Refer Time: 08:48) logic radiation.

Student: It is also can be differentiate simulation process (Refer Time: 08:55).

Ye, ultimately, this is an implementation where it comes in is you have for example, if you are doing synthesis you have done the synthesis, it is the last stage of the synthesis that is changing (Refer Time: 09:10). So, you are much closer to the implementation, but the advantage of course, is that the design turnaround time is just a few hours. So, you could do a lot of exploration a lot of validation and so on using this kind of an architecture.

(Refer Slide Time: 09:37)



Just to put up a high level picture of what a modern FPGA might look like.

So, what we saw in the previous picture was this and the IOs there, but there is a lot more in modern half FPGA as than this the most of the area might still be occupied by this combinational logic block they are the heart of the FPGA, they are what defines the FPGA, but there is the realization that by itself is not good enough for a number of other things that we would like to do. So, there usually would be some kind of a memory on chip memory that you would like to utilize otherwise it might be too inefficient if you have lot of data, then well; we will look at what the CLB is looked like, they may at best carry some flip flops or something like that nothing more complex than that, but you

may want to store large amounts of data. So, what is called Blockram in these Xilinx terminology is usually, they are distributed throughout the chip for whatever use you to put it to there are DSP class of components multipliers and adders those are important.

When we say combinational logic block; we are essentially talking about discreet logic, of course, you want to augment that with some powerful arithmetic components also. So, the discreet logic block might already have within it, some adder logic at least because it is so common, but multipliers are more expensive and they may be not necessarily part of that standard combinational logic block, they may be externally present distributed again throughout the fabric. So, such stuff are there; if this you call one tile or a slice or whatever the terminology is those are distributed all over the chip.

So, that you have arithmetic components close to your logic you have memory also close to your logic. So, this part is what is the standard programmable logic, but the direction in which these a FPGAs have evolved is that there are also some hard coded components particularly processors that are present inside the FPGA itself the intention of programmable logic is of course, very different as supposed to a processor; that is programmable in a different sense, here we want a hardware to be programmable, but it just turns out that systems that we design when ASIC systems that we design, actually we would like to put processors into them just because there may be parts of the system that are designed in software that would go into a processor implementation anyway high level control might be performed in software and so on.

A software component is usually there in any complex modern digital system. So, instead of leaving it to the designer to get a synthesizable processor from somewhere and element it on to the programmable hardware; what FPGA architects have done is essentially, they put some hard coded macros which are just processors ok, these look like standard processors, these are not programmable in the sense of hardware being programmable, right. So, this is a floating point unit CPU pipeline L 1 cache, L 2 cache as scratchpad, memory standard controllers and peripherals and so on memory controllers deal.

So, these are what a typical processor environment looks like so, but you might have a couple of processor cores or any number of process of cores embedded inside the FPGA fabric to help with all kinds of features, essentially, if you want to implement a design in

which part of the component is there in software part of the component is there in hardware, then this kind of an architecture helps us implement that in a reasonable way, most of the area might still be occupied by this program of the logic part, but that software and the processor component is considered essential at least high end, PFGAs and FPGAs targeting SOCs would usually have some kind of processors, of course, even if that hard coded processor was not there.

There is nothing that is stopping us from synthesizing an FPGA down into these CLBs, you could do that, but as we know there is a price to be paid, it will not be as good as a hard coded processor ok. So, that is just the high level picture of an FPGA all of that of course, we will not be targeting in the synthesis course.

(Refer Slide Time: 15:00)

**FPGA Programming: SRAM**

- Normal SRAM cell
- Programming: writing 0 or 1 into SRAM cell
  - this, in turn, causes selections, connections, etc.

The slide contains two diagrams. The top diagram, labeled 'Programming the logic', shows a 'Mem' block connected to a 'MUX' block, which is then connected to a logic circuit. The bottom diagram, labeled 'Programming the interconnection', shows a 'Mem' block connected to a switch-like structure representing interconnections.

NPTEL (C) P. R. Panda, IIT Delhi, 2017 5

But let us go down to the lowest level here and talk just a little bit about how this programming is done we ought to know that before we can synthesize into an FPGA an SRAM is usually used as the programming element this is just a normal SRAM cell which stores a 0 or A 1 and the programming essentially might translate to just writing a 0 or A 1 into that SRAM cell, but with a very definite purpose, this kind of programming that you do will have some implications, it may cause selections and connections and so on.

That is what is essentially part of the customization of hardware; there is nothing fancy going on here, although, this is only one kind of technology, there are other technologies

that have been tried programming technologies of hardware, there is people have tried and what is called an anti fuse element where if you pass a current; that is above a certain threshold, then that fuse breaks and that distinguishes a short from an open, for example, that is inherently how programming is done on such a design such a system may or may not be reprogrammable multiple number of times the implementation as a memory of course, has a nice feature; that tomorrow you changed that value to 0 and the meaning is something very different, yeah.

Student: But this typically when we compare this (Refer Time: 16:35) like quite costly and then it will be a very bulky device, if I realize a very big piece FPGA. So, did we ever get researchers also tried (Refer Time: 16:45), they would need to kind of refresh that, but that would be very small and very cheap because cost would be a challenge cost and area both.

Digital cell; I do not think has been tried out as a programming element although you could have an on chip DRAM for storing your data that is a different matter and that certainly has been their FPGA, I am not sure, but certainly there are ASICs in which on chip embedded DRAMs have existed for a very long time now, but the purpose has always been storing of data because of its compactness not really programming using it as a programming element probably has not been there because the SRAM is a faster device DRAM is slower its compact to, but it is slower what is the DRAM? Anyway, it is a capacitor.

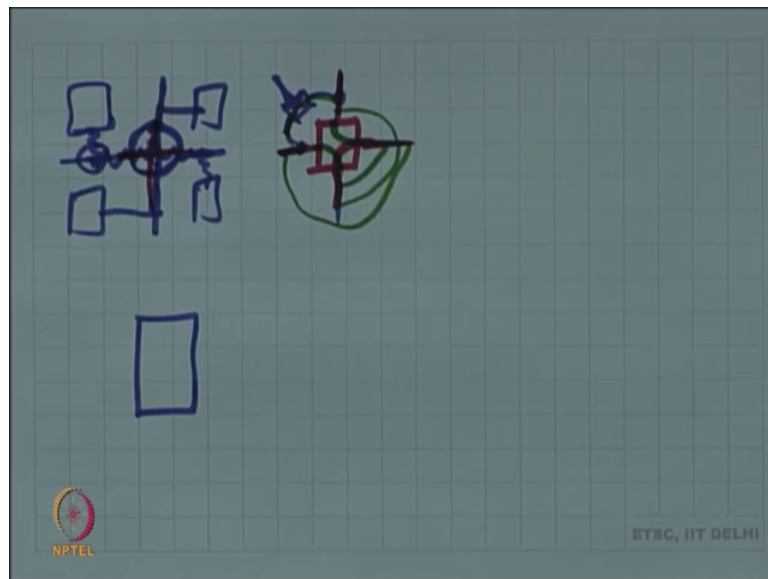
Student: Yes.

Right, but the SRAM cell, if you know, these are bigger cell, it is A 6 transistor circuit while it is bigger, it does have the advantage of faster also, we want this to be distributed all over the chip, this, it is not as though it is all there in one place from that point of view, also some of the advantages of DRAM go away because the compactness arises out of a tight integration, you, if you distribute it everywhere, you may lose the advantage of density anyway what kind of programming, it is it could be that you have such a selection kind of logic there is one logic here, there is some other logic there and this programming of that bit into 0 or A 1 might mean that z has this kind of function or that kind of function, of course, this is very elementary.



For this, you do not need any big support you could write code that does exactly this point is that both x and y had to be implemented and it is not particularly novel. This looks like just a standard design other interesting applications are this, for example, the interconnection could be programmed in ways that you have two terminals that are connected by a transmission gate and whether you write a 0 or a 1, essentially determines whether those two are shorted or not that could be used to realize a programmable interconnect in the following way you have some such design.

(Refer Slide Time: 19:21)



This is a programmable box where you would like to establish any kind of connection, either this or maybe simultaneously this or maybe, you want some yeah some other kind of connection maybe you want this to be connected or that to be connected and so on. So many different connectivities could be theoretically possible and interconnection fabric might actually look like this. So, while this box is there, you could realize any kind of connection maybe you do this and you just leave these two unconnected.

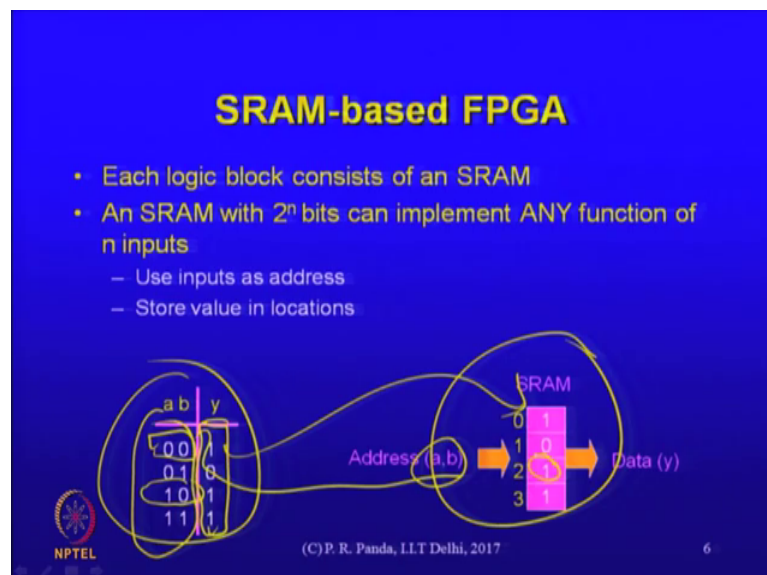
So, that is the kind of programming that could be achieved by this kind of a structure; how would you do this essentially between every pair of terminals that you would like to offer the programmability feature, either, they are connected or not connected essentially, you would have may be a kind of yeah a transmission within which the control is determined by our memory cell, right. So, this kind of a thing is common in the interconnection part of an FPGA. If let us say, I have these are my logic blocks and I

want to establish a path between this through this and this right maybe also simultaneously establish a path between this, this and this; all of this is ok.

So, there is one programmable interconnect there that I need to program which is decide whether that block is connected to that line or not. So, that is where that programmable interconnect is useful, but the more interesting interconnect is here where I have these 4 things coming in and I decide which one is connected to which. So, all the pairs between which I would like to establish connections now could possibly use that elementary programmable interconnect structure. So, how many such pairs are there? There are 6 pairs here, right, this could be one pair; that could be one pair, this could be one pair, this could be one pair; that could be one pair and this could be one pair, each of these; you might want to connect or might not want to connect ok.

And so, it is essentially 6 transistors that you need and 6 bits that control each of the gates of the transistors, the programmable interconnect would be just that for one bit if it is a bus that is coming in of 4 bits or whatever the same structure has to be repeated 4 times the logic block itself again consists of an SRAM based structure in the following way.

(Refer Slide Time: 22:35)



So, what we have is just a piece of memory at the heart of the CLB, there is nothing fancier it is just a memory, it is yeah.

Student: (Refer Time: 22:56) SRAM; what will it take in the program after (Refer Time: 23:04) would have to be programmed.

Ye, we would have to be go through the just programming, what is involved in the programming, there is a bit file that is the configuration file that we store elsewhere. So, when on power up that configuration file has to be imported into the FPGA, it does not take too much time considering how much time it remains; that might not take too much time, but yes if you power it down, then you have to read that configuration again before you continue, but the idea is actually very simple in the SRAM based FPGAs, suppose, I ask you to design a generic piece of logic that can implement any function of two bits, how would you do that any function of two bits means essentially, I should be able to implement any truth table; the two input function with one output function essentially translates to a truth table in which there are 4 rows.

Student: (Refer Time: 24:21).

Right, all of this is already known the left side of the truth table is all known. So, what is remaining is this; essentially that one column is what is known. So, if when I say any function; what do I mean in terms of realization? You just need to give me a way to realize that column. So, you can see there is an application of a memory element like an SRAM.

Student: This kind of that look of truth table (Refer Time: 24:53).

These are called look up tables why. So, how would you perform it if I gave you some memory and asked you to build any function of two inputs; two bits how would you do that so that.

Student: (Refer Time: 25:09).

Yeah. So, that column there just needs to be stored in the SRAM that is all and that input here just is the address the whole thing is just a combinational logic it see, it is realized using sequential elements using memory elements, but there is nothing fancy here is just the a b that is the two bit address to the SRAM structure, just corresponds to whatever is the input in the corresponding location, we store the appropriate bit whatever that truth table says.

Student: Do not use any gate level (Refer Time: 25:45) on the FPGA (Refer Time: 25:47).

This is one way of doing it, but actually this is what is the prevalent way.

Student: in the softwares like when we try to say that elaborated designed part, it gives us the gate level (Refer Time: 26:03) of (Refer Time: 26:04) attain in the logic.

,But that is still a little higher level, what is there inside the combinational logic block is not exposed to you see that.

Student: (Refer Time: 26:13).

Yeah right ye, but we will get to what the steps are, but that is still a little abstract view, if it is showing a netlist to you that is a little abstract.

Student: (Refer Time: 26:24).

Right, what is there inside, particularly, how this translation is made to what is there inside is proprietary information that will not be visible to the user, right.

Student: Yes sir, it is not like that gates are already implemented on the FPG, burn the inputs according to the program.

The burning means writing into those SRAM cells.

Student: But writing in the SRAM, we are not writing at the input of the (Refer Time: 26:47).

Right, well why do not we see a little more complex example, but at the heart it is this idea is this the look up table why it called lookup table, it is because much of the implementation is through a lookup like there is a little bit of logic also, there are some flip flops also separate flip flops, these are also memory elements, but some D flip flops are might also be there as part of the.

Student: But there is some memory element stored in the start of the program, no matter how our program will (Refer Time: 27:19).

Yes, yes, also that is one part of the FPGAs memory which is a configuration memory. This memory is not visible to the programmer to the designer in our picture there that block RAM is what is available to us for storing data and retrieving data the block RAM structure, we can control that configuration memory identically show in this picture, it has to be embedded as part of the CLBs somewhere, but we do not get to see that modern Xilinx vertex.

(Refer Slide Time: 27:55)

**Xilinx 7-Series Architecture**

- 64-bit Look-up Tables (LUT)
  - All functions of 6 inputs
- What if we wanted 2 functions of 5 inputs instead?

The slide features a diagram of a 6-input LUT, represented as a blue rectangular block with six input lines on the left and one output line on the right. The text '6-input LUT' is written on the block. At the bottom left is the NPTEL logo, and at the bottom center is the copyright notice '(C) P. R. Panda, IIT Delhi, 2017'. A small number '7' is in the bottom right corner.

Seven kind of series architecture looks conceptually pretty much like this like, if you extend that two input argument to A 6 input argument, what do you need there in that lookup table? What is this structure?

Student: 64.

It is 64 bits in the structure that is all given that you can implement any function of 6 inputs well that is the basic idea; of course, people are always working on tweaking and improving that lookup table structure and that CLB structure. So, there are a few other things that are useful and the FPGA designer would make those things also be part of the CLB structure, we said that there is a need for storing data that storing data does not actually form part of the CLB, you would have to store data in block RAMs, if you wanted to do that, but there are some flip flops that they would include in addition to these LUT structures that are used for combinational logic only.

Student: (Refer Time: 29:04).

So two things; one is flip flops will be there, other is some adder kind of circuits would be there because they are very useful; what they might give is maybe a 4 bit adder and expect you to realize a larger adder by chaining multiple 4 bit adders that are spread across different CLBs. So, this is used for discrete logic you just have some random Boolean function you would implement it using the LUT, but there are a few other things you may want flip flops output you may want to register for some reason and they may give some flip flops there.

But they might also give some adder bits either two bits of an adder or 4 bits of a simple carry chain that might be there, the way you would implement a bigger adder wider adder is that you make sure that the carry chain is appropriately set up output from one CLB goes into the appropriate input of the next CLB, but here too, there is a little bit of variation that you can think of what if instead of f function of 6 inputs, you want to realize two functions of 5 inputs each 6 inputs maybe the most general structure, but it could be that often I do not have 6 variables, maybe I have only 3 variables, but I still want an efficient implementation you can see that if your functions were all of Boolean functions were of 4 variables each, then the 6 input LUT is sort of an overkill, of course, the larger that number of inputs is the more the delay is of this structure and the more the area is and so on.

But can we think of a simple variation? Imagine; what that 6 input LUT looks like? This is just a simple memory structure, nothing more than that, but what if I wanted instead two functions of 5 inputs each, idea is I should be able to reuse that structure sometimes to perform any function of 6 inputs, but at other times to perform maybe 2 functions of 5 inputs each or maybe there are thousands of these elements spread over the FPGA some of them, I can personalize as one Boolean function of 6 inputs, but some other of those blocks could be customized as 2 Boolean functions of 5 inputs each.

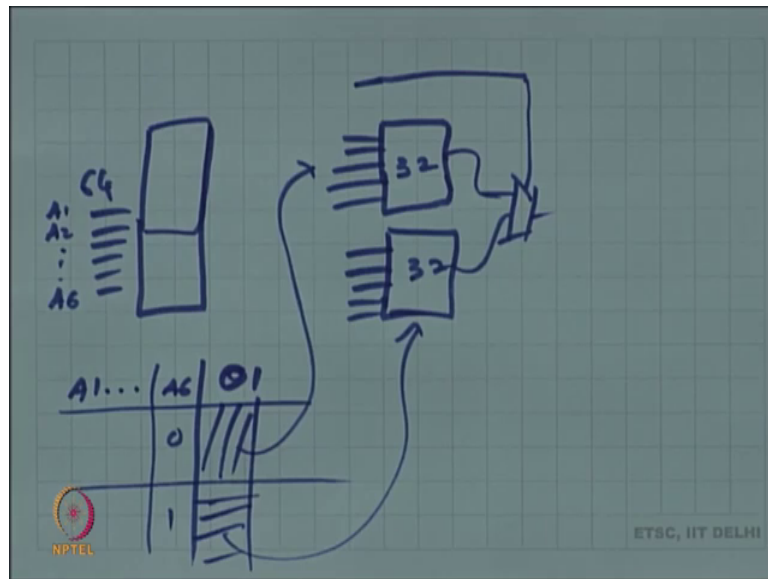
Student: (Refer Time: 31:43).

Those complement; how many bits are needed in an LUT which is 5 input; how much storage is needed as use the truth table?

Student: (Refer Time: 31:57).

32 bits ok. So, this one requires 64.

(Refer Slide Time: 32:05)



Student: (Refer Time: 32:05) two different functions.

So, one idea that suggests itself is let me divide this into 32 make this the same size, instead of having one 64, let me have two 32 that at least is the heart of the logic and that should not be hard to do, but in addition, I can do just a little bit. So, there are 6 inputs here. Now this one takes 5, this one takes 5, what else?

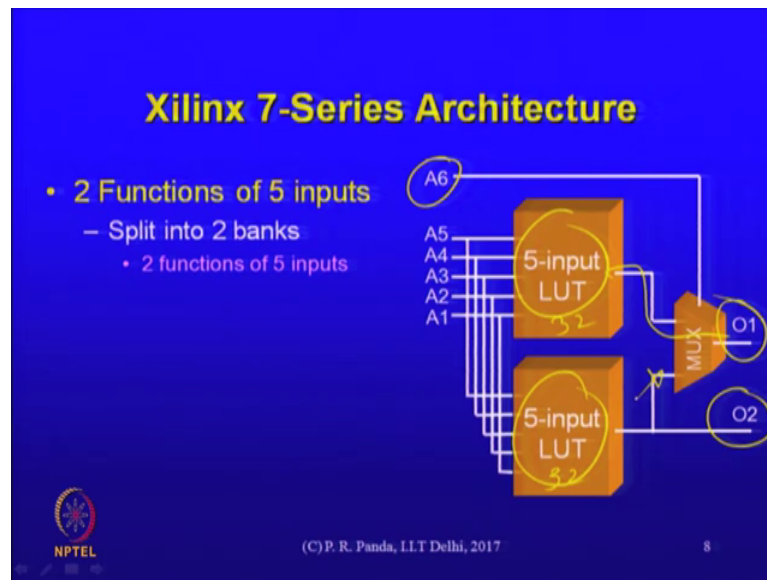
Student: (Refer Time: 32:41) I think if it is this one or this (Refer Time: 32:43).

So, the outputs of these can go to a MUX in which we say; it is that sixth bit a sixth input could actually select between 1 or 2; what about the connections here? These were my  $A_1, A_2$  and  $A_6$ . So, here what are the connections?

Student: (Refer Time: 33:10) common to both the (Refer Time: 33:13).

Right so, that structure is actually very simple, it is just this, what it is showing is that output  $O_1$  is which this is the normal 6 input function, right.

(Refer Slide Time: 33:24)



So, that is the sixth bit and these are my 5 bits and I have some such structure like that and this is my 6 function; any function of 6 inputs, I can spread it over the 32 bits that are here and I could choose one or the other depending on ah.

Student: Now 30; 30 addresses are kind of not used right; so, the first 5.

So, this 32 and that is 32, right, this has this is 32 bits and this is 32 bits.

Student: But the address will be (Refer Time: 34:16) address, right (Refer Time: 34:18) if I keep the same bit logic there then my address selection memory is continuous (Refer Time: 34:25).

This what was a truth table the truth table was A 1 through A 6 say and I am saying have O 1 as the output and these correspond to our A 6 being 0 this corresponds to A 6 being 1 and whatever was there that is what goes into that LUT and these elements go into this LUT, this is a standard way, it is just that what would have otherwise been part of that decoder is now explicitly done as I done what about the other output O 2, O 1 is the output of the mode where I am using this logic as A 6 input, right, but I could also be using that logic as two functions of 5 inputs each so; that means, that this is one function that is a totally different function and that is when the O 2 is relevant, otherwise the in the other mode the O 2 is not relevant and this one anyway can be selected here this is not selected and that A 6 can be used as a selected.



Student: (Refer Time: 35:44) CLB, I am not saying one, but I was guessing that if I have a 62 bit LUT, all my decoder of memory is sitting before the memory not after the memory when this case, we are saying that this (Refer Time: 35:57) something, but within the CLB itself I could shift that decoder that is the last decoder of A 6.

Ye, yeah.

Student: MSB after my memory which is counted into (Refer Time: 36:10), I mean because the hardware phases (Refer Time: 36:12).

, No, no, this is what the hardware looks this is not theoretical one, this is taken from the Xilinx hardware this is as close to.

Student: This is how they implement scene, right.

This is the CLB for the reason that most of the time your functions may not be very general sometimes you need smaller functions right ah. So, this is more efficient otherwise, you would need another CLB for the smaller functions which may be wasting area.

Student: (Refer Time: 36:37).

Ye, you could think of other variations why to you can have 4 different and so on, but that is what the current CLB looks like, but that is what the Xilinx CLBs have always looked like the size has been changing over the years they started with maybe 3 to 4 to 5 and so on, but the philosophy has been this only all the time there is difference in the way; they have evolved has been in what other components they are putting, there maybe they had a few flip flops.

So, some of that also has been changing that number has been changing these, but the evolution is really more along the peripheries and putting processor and such things to make it more attractive for proper system designs than the CLB itself which has looked like this conceptually forever, right.

Student: (Refer Time: 37:30) if this is 5 when we have A 6 input LUT and (Refer Time: 37:35) functions.

Right.

Student: We can similarly have a criteria for 3 equal function two bit functions also. So, the question is that do we have only (Refer Time: 37:45) one mark outside or inside that 5 (Refer Time: 37:48).

See somewhere, there is a tradeoff between efficiency of implementation of that CLB there will be tens of thousands of these CLBs. So, while it is to have a very general structure you are paying a price in terms of area all the time. So, this was arrived at statistically there is no theoretical reason why it should be just this, but it is just that you have a large number of functions to implement they have various characteristics with respect to how many inputs they are sometimes they are 2 and 3 inputs in which case this is an overkill sometimes they have 10 inputs or something like that and then you have to use many instances of these CLBs to realize them.

Now 5 and 6 these have been arrived at by looking at this statistics the way these things are distributed. So, indeed if you have very small functions of only 2 or 3 inputs, you are wasting some space you do need a CLB. So, it is not a general structure it is this kind of a structure usually where you have only two functions not any number of functions.

Student: So, this structure we arrived after we like had A 6 input LUT and we look inside it. So, we solve this structure that is true, right.

Say again.

Student: We like we were looking at a logic block of 6 input of LUT, right. So, when we look inside how it is implemented; it is how this structure?

See A 6 input LUT otherwise would not be implemented this way, right, it would be implemented with a standard decoder and 1 SRAM that is a lot more efficient if you put MUXs there and area increases.

Student: (Refer Time: 39:34).

This is what this CLB the selected CLB by Xilinx is that you do not have to do it that way, but this is what the structure.

Student: If we again look inside the structure the 5 input LUT that is again visible here will be again see.

No, no, you will see a memory structure standard memory structure, but in your (Refer Time: 39:52) you could do it that way that you could split it into instead of the 5 what there is nothing sacrosanct about the 5 number of course, you could split it further and instead have 4 different 4 input tables if you would like to that is a designers choice and a trade off, but it fundamentally impacts our synthesis algorithms because they have to be aware of this structure. It is not as though you can have 4 functions or 4 inputs each, you can only have two functions of 5 inputs each for this kind of the CLB structure.

So, when we say that this is my general logic and you synthesize it with that FPGA as the target that tool has to be aware of this internal representation.

Student: They can combine smaller components into bigger with aligning all the inputs and making it 6 input or 5 input.

Smaller components could be combined into bigger one, for example, if they occur in sequence output of one is going as input to the other, then it the whole thing might form a different function of maybe 4 inputs or 5 inputs, but still fit within one of these that is the intention of this kind of a design that often you can pack smaller logic into other, but if you need the independent outputs of each of them, then this does not work, this limits the number of outputs that are there with that kind of a logic structure.

(Refer Slide Time: 41:17)

**Technology Mapping in FPGAs**

- More complex "library cells"
- Library cells can be customised
  - cannot enumerate fully

NPTEL (C) P. R. Panda, IIT Delhi, 2017 9

How would logic optimization work? First of all, we have two phases, right, in the multi level logic minimization, there is nothing particularly two level about this this whole thing is that you can take output from one of CLBs and you can take it to a different CLBs, it is in a very natural way amenable to multi level logic implementation.

So, multi level logic implementation is done in two phases, we had that first technology independent logic optimization phase and then there was a technology dependent phase often, it might be that a technology independent phase is not different at all it is the same there we are minimizing literals. So, whatever we are doing, we do it the same way there is nothing different, but that second phase would be very different because the realization is different the technology is different, the target is different, you want to optimize for this kind of a target. Now, what is the library in that tech map phase, there were two inputs one was the netlist the generic netlist, but the other input was the library, here what is the library?

Library had a bunch of standard cells; if it was a standard cell implementation all the cells were enumerated there here what is the equivalent.

Student: LUT.

LUT is the equivalent; how do I incorporate it as part of my tech map.

Student: LUT d flip flops.

D flip flops are fine, ye, they are there just like they were there as cells in the standard cell library and in fact, we have embedded some D flip flops in each of these CLB. So, so that is fine. So, just the logic itself the logic there was implemented in terms of standard cell library elements here they are LUTs, but remember the way we were doing the tech map, there is series of steps where first we did a decomposition into inverters and NAND gates.

Right and then there was a mapping of that structure and there was a particular matching and covering strategy that we had what would the equivalent here be first of all LUT is our library elements, but how do I go about the matching. In fact, there we had decomposed a more complex netlist into simpler elements primarily because the

matching becomes easier right here what does matching means that LUT can be programmed to be any function therefore.

Student: (Refer Time: 44:05) the number of matching can be.

Whatever can be imaged right, it is a matter of me customizing that LUT to be whatever I want it to be. So, there is no more real matching to be done here is there any covering to be done?

Student: (Refer Time: 44:21).

If we do neither matching nor covering then there is nothing to do in that technology matching phase.

Student: (Refer Time: 44:30) reuse (Refer Time: 44:31).

(Refer Slide Time: 44:41)

**Matching**

- **Matching is trivial**
  - ANY function can be implemented
  - make sure number of inputs match

NPTEL (C) P. R. Panda, LLT Delhi, 2017 10

First of all, since any function can be implemented, there is nothing to do with respect to matching, actually, even in the other case, we had simplified the matching by just saying that we just look at how many inputs are there, there is only one gate which is a two input NAND gate, I just have to make sure that the number of inputs match just like I did there; here also that is all that I need to make sure.

(Refer Slide Time: 45:04)

**Covering for 4 i/p block**

- **Treat library cell as**
  - Black Box with 4 inputs, 1 output
    - no need to decompose into NAND/INV
- **Technology Mapping**
  - Cover the network by sub-graphs of 4 inputs, 1 output

NPTEL (C) P. R. Panda, IIT Delhi, 2017 11

Now, covering actually is not really complicated you treat that library cell as a black box with some number of inputs if that LUT has 5 inputs and one output it can be programmed to be any function of 5 inputs and one output we are just looking at a black box with 5 inputs and one right output whatever it is and there is no need for us to decompose into NAND inverter and so on because I know that any function of 5 inputs or 4 inputs can be realized. So, the technology mapping just consists of covering that network which is that subject graph in terms of sub graphs of 4 to 1; what logic is there inside that structure is not important because I know, I can realize it anyway right with an appropriate configuration of that LUT.

So, the only thing that I want to do here is cover the entire network by sub graphs of that property that it should have 4 inputs and one output maybe I want to minimize the number of such boxes in terms of which I realize the logic if it is timing that I want to optimize then it would be the depth of that network that I realize that in terms of that is what I would like to minimize. So, this is all I wanted to indicate that would be a variation for technology mapping there is the other issue that I have a fixed number of functions in each of the CLBs and so on.

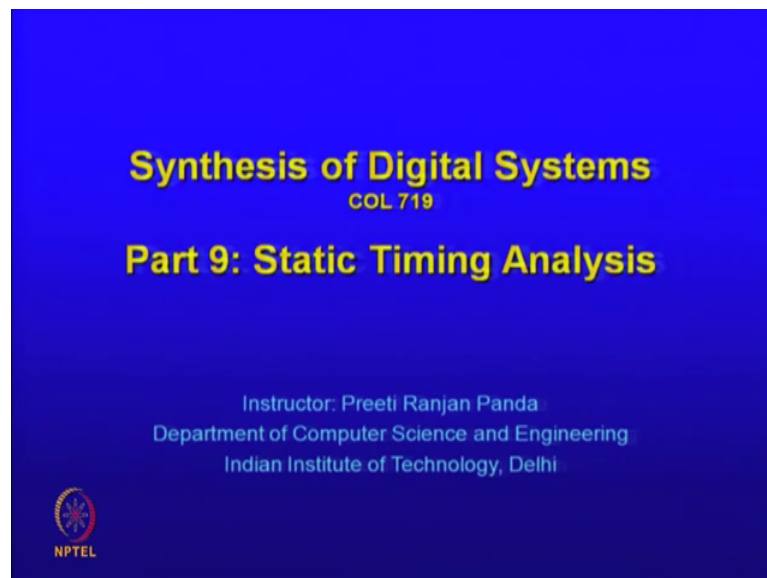
So, other complications are there, but the principle is just this; what changes from the standard logic optimization procedure it is just this. In fact, it looks like it is a simpler problem for us, but what does complicate it is that the CLB is not just this there are a few

other things there are arithmetic logic and so on. So, there are other things to be done and overall the synthesis process is indeed complicated, but some of the elementary processes are simple variations.

Student: (Refer Time: 47:31).

Ye, we did not talk too much about timing for instance here and the timing is influenced by the route that the wire take while this step itself might not be too complex the place and route is a complex process that does influence timing in other technologies as also in FPGA, right.

(Refer Slide Time: 48:04)



We will conclude the synthesis discussion with a couple of adjacent topics that cannot be completely divorced from synthesis, we have been talking about timing in various forms that is why I thought; there is a need to look at some of the very basics of timing analysis the algorithms are all variations of things that we already know, but this topic by itself is a complicated one it deserves a course on its own, really, there is no way to do justice in the 2 hours that remain, but there is an interplay of some of these topics with what goes on in synthesis that is why this is included ok.

(Refer Slide Time: 48:57)

**What is Timing Analysis?**

- Compute the delay of a combinational circuit
  - Worst case
- Why?
  - So we can decide clock rate
- Why not just simulate?

NPTEL (C) P. R. Panda, IIT Delhi, 2017 2

What are we trying to do we are trying to statically compute the delay first of just a combinational circuit the delay is the worst case irrespective of what data you give on its input, you say that this is the worst delay; why is this useful? Of course, it is of fundamental importance if I know that the delay does not exceed a certain amount, then I can use a clock period of that duration to run the circuit. Of course, the job of synthesis to make sure that that delay is as small as possible, but there would be a different analysis engine that at the end of the synthesis you could run and say that this is the delay. Why cannot I just simulate it? Simulation a gate level simulation does tell us; what is the delay, if you set the input the changes at the inputs of the gates will propagate to the changes in the outputs and when you see that the output has stabilized, that is how much time the logic has taken. So, I could use this as an alternative or not.

So, simulation requires what other input than the netlist itself. Synthesis has just the netlist or whatever is your functional specification that is what it has.

Student: (Refer Time: 50:20) constraints.

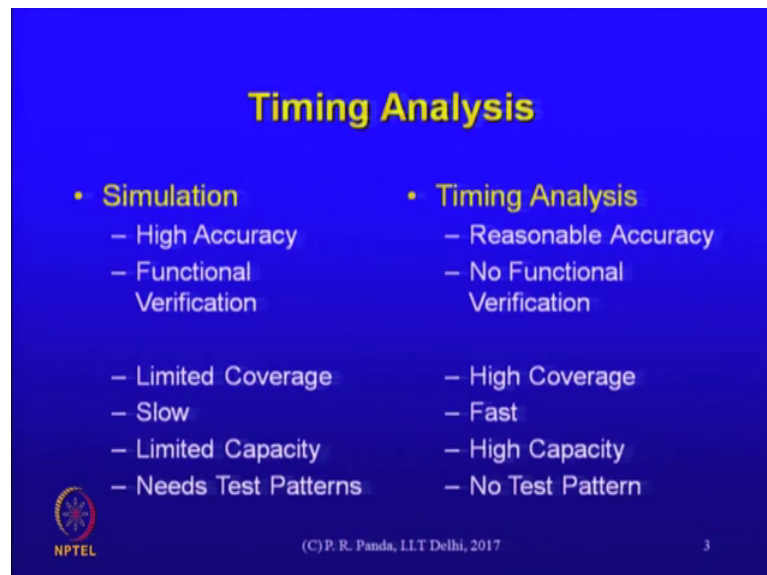
Constraints; it is cannot impose constraints on synthesis tools constraints are there those are input if we have used constraints as part of all our synthesis algorithms, you cannot say at the end of the synthesis course that I do not know about constraints, we do impose events a resource constraint that the constraint deadline is a constraint, but a simulation has one other input.



Student: Stimulus (Refer Time: 50:46).

The input vector; the input vectors are provided the simulator does not run without you saying what is the data on the input to the synthesis tool do you say; what is the data, you do not. The point of timing analysis or static timing analysis is to compute that delay without taking any data at least in a purer form, it would not rely on information about whether that particular input pin is 0 or 1. The result that it gives is a powerful one in that; it covers any input that you might give. Simulation would cover only those inputs that you actually provide.

(Refer Slide Time: 51:32)



The slide is titled "Timing Analysis" in yellow text on a blue background. It compares Simulation and Timing Analysis in two columns. Simulation is listed with: High Accuracy, Functional Verification, Limited Coverage, Slow, Limited Capacity, and Needs Test Patterns. Timing Analysis is listed with: Reasonable Accuracy, No Functional Verification, High Coverage, Fast, High Capacity, and No Test Pattern. The NPTEL logo is in the bottom left, and the copyright notice "(C) P. R. Panda, IIT Delhi, 2017" and the number "3" are in the bottom center and right respectively.

Simulation	Timing Analysis
– High Accuracy	– Reasonable Accuracy
– Functional Verification	– No Functional Verification
– Limited Coverage	– High Coverage
– Slow	– Fast
– Limited Capacity	– High Capacity
– Needs Test Patterns	– No Test Pattern

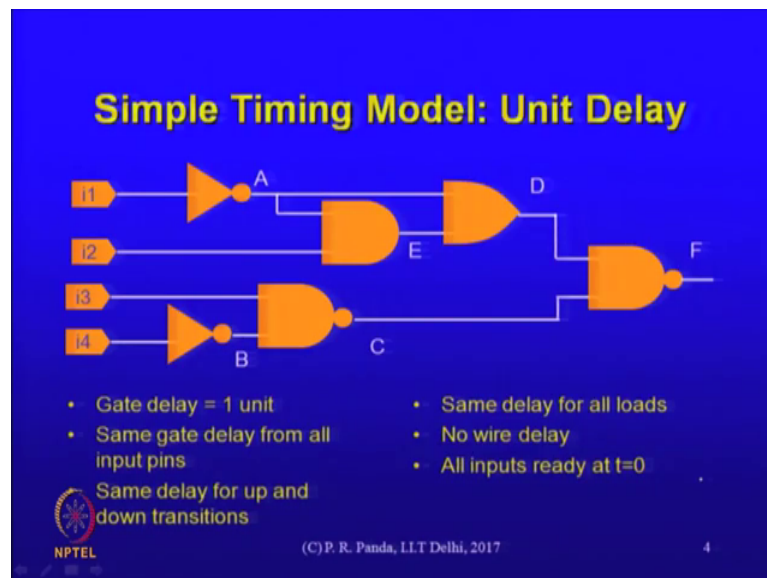
So, a quick differentiation could be made between these two mechanism; both of them could be used to validate timing, but if I do a simulation versus timing analysis, there is one thing about accuracy that maybe; we get back to later, we are saying the simulation could be more accurate than timing analysis without knowing what test timing, it is hard to resolve that, but all the other should be clearer. Simulation actually does a lot more than just the timing.

It does functional verification right you look at the logic output and decide whether it is working or not the functionality is working or not; timing analysis has no such feature it only tells you timing your everything may be wrong, but your critical path might look good. It actually might be very fast circuit that does something, that is not required. Coverage; we say limited coverage versus high coverage; what do we mean?

Student: The vector that we need to synthesize the (Refer Time: 52:29).

Yea, you need to provide input vectors for the simulation to work and of course, we are limited by how extensively, we can generate those vectors; how comprehensively, we can simulate the circuit with the vectors just because there are way too many vectors to test it for. But in the timing analysis the coverage could be much better because you are making a strong statement that irrespective of what the data is your circuit has only so much delay. Because of the same reason there is a speed difference simulation is relatively slower, but timing analysis could be fast capacity, again, we are limited by capacity just because these are all related to each other all these 4 are related to each other. Let us get on to a very simple timing model using which we can illustrate some things.

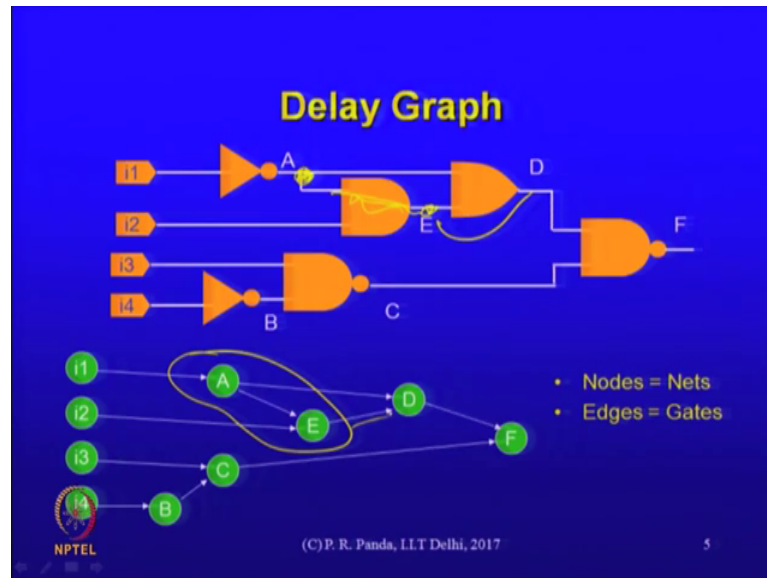
(Refer Slide Time: 53:23)



Let us assume that the gate delay is one unit, let us assume that we have the same gate delay from all the input pins, we are just indicating generalizations that later need to be done that is all. The reality is that that delay might be different in general from this delay right. From delay means an input changes and how much time later the output changes. It depends on what the circuit is doing the result of one input changing might propagate much faster to the output than some other input changing in general of course. Same delay for both low to high transitions and high to low transitions that two need not be the case in general, same delay across all the loads that of course, is not true in general, there is no wire delay.

So, lots of things are simplified here just so, that we can put up the most basic algorithm. And the other thing is all the inputs are ready at a particular time, time equal to 0. That also might not be true in general if it is not true then you could exploit that to perform a more aggressive timing analysis. This analysis is with a graph in which I need to define only two things; what are nodes and what are edges.

(Refer Slide Time: 54:48)

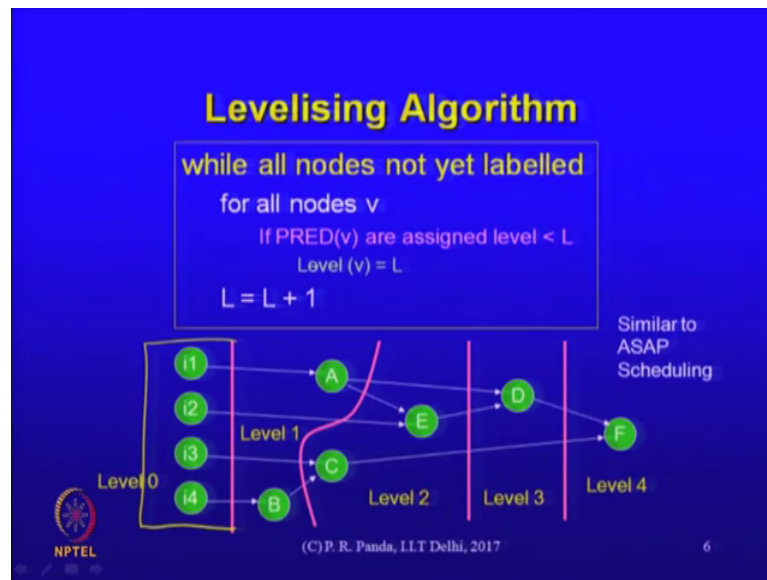


But the way we are treating this problem is that the nodes are the nets and the edges are the gates. So, why do I have that edge between A and E? That edge refers to the gate and there is a node corresponding to A, there is a node corresponding too. So, E ok; so, E to D; there is an edge because of the same reason that is my. And this is not done just to do something different, but it is useful in some ways. So, that is why the model is done in this way.

Student: (Refer Time: 55:44).

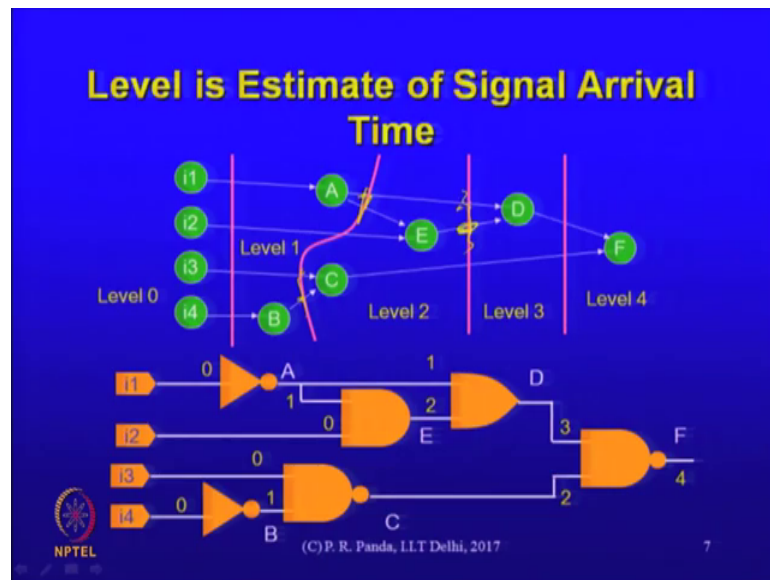
There are delays associated with the nodes, as of now we just said it is one, but in general I can put any delay there and that can be A included in some appropriate way.

(Refer Slide Time: 56:00)



A trivial Levelising algorithm is something that we can start with let us just assign levels; levels are useful because they tell us when a signal arrives at some inputs or some output this is a dag kind of a circuit anyway. So, I should be able to assign these levels. So, all of these that do not have any inputs, I can pick up and that is my first level, level 0 all the others that have inputs from what is there at level 0 can be put into 1. So, that is how level 2 would be defined 3, 4 and so on. This as you can see is very similar to that ASAP scheduling algorithm. Algorithm is very obvious, you just say that while all the nodes are not labeled, you consider the nodes one at a time and if all its predecessors have been labeled then you are able to label this; what label will you give it, it would be one plus whatever is the max level of any of its predecessors that is all right.

(Refer Slide Time: 57:20)

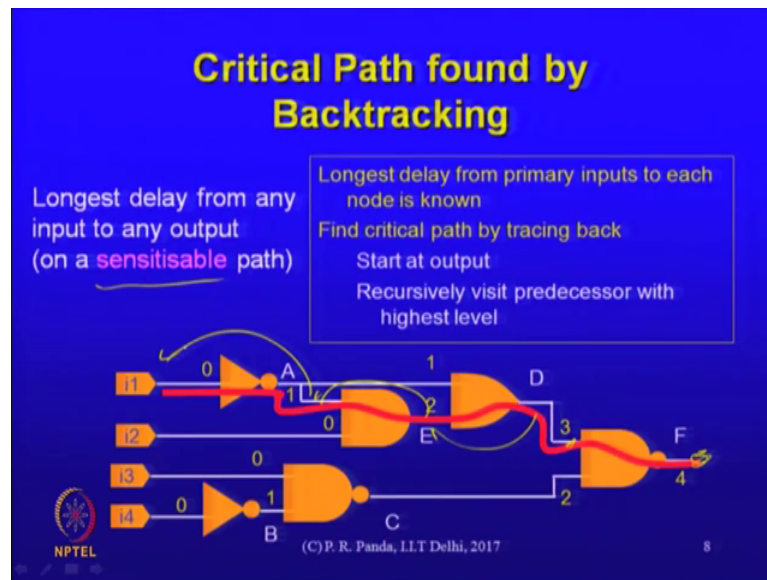


The level is just an estimate of the signal arrival time. All the input signals are arriving at time 0, then these signals are ready at the end of time 0 for C, the inputs arrive at the end of time 1. For D, you have all the inputs well when I say inputs arrive all the inputs are ready at time 0 that is what the.

Student: Their assumption is that the same delay (Refer Time: 57:55).

No, no, paths are differ; oh we are saying that at that time all the inputs are ready. So, for D, you can see that one of its inputs got ready earlier ok, it is this one that arrived later the computation of D essentially starts after all its inputs are there. The timing analysis of the circuit means that you are finding what is the maximum level of this.

(Refer Slide Time: 58:24)



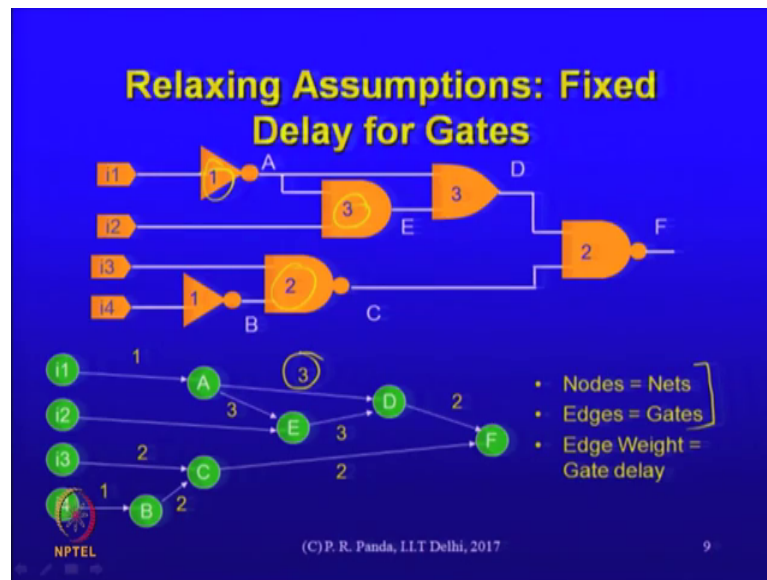
Having done that level analysis first then you could determine what is the critical part of this circuit critical delay is that level number itself critical path, we may be interested in what would be the point of knowing about the critical path having already known the critical delay. The critical delay tells us at what the clock I can run this circuit the path why is the path useful?

Student: (Refer Time: 58:55) We can re time it if necessary.

This helps in identifying any bottlenecks that are there, that if it is not good enough that is where we should focus right maybe I can do something about it. So, what is that critical file? It is the longest delay from any input to any output, that is what the longest such path would determine the level anyway. Modulo that path being sensitizable in ways that we will look at later for now, it is just a structural computation right. The path could be easily obtained by just backtracking from the output; if this is the level you just proceed to that inputs yeah corresponding to the maximum arrival time.

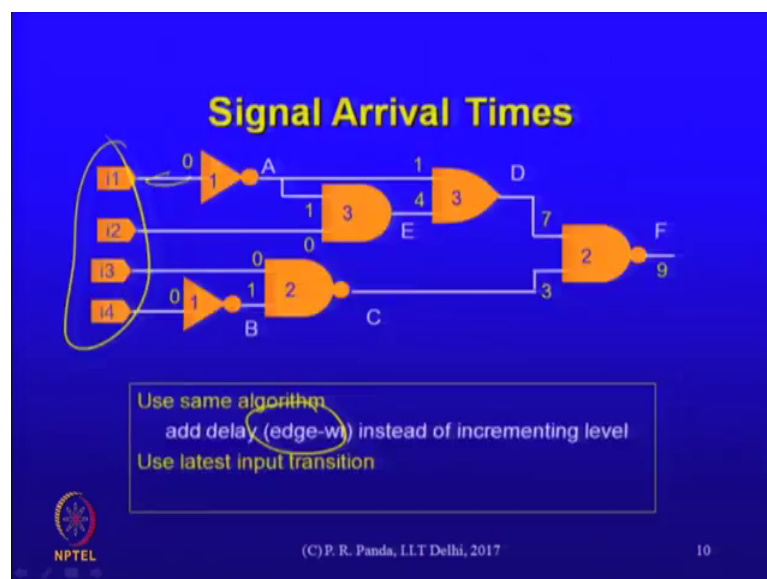
So, you would pick that go back from here you would pick this, from here you would pick this from here and you would pick this that is the path that we are interested recursively you visit the predecessor with the highest level. So, this is a simple computation.

(Refer Slide Time: 60:00)



We can very easily relax the assumptions that we started off with our nets and gates still our or our nodes and edges, but the gate delay need not be unit, I can make it whatever I want instead of 1, I can say it is 3 and 2 and so on and I can represent that in my graph here with the corresponding edge weight the earlier model, I just had an edge, I did not have weights on the edge, but it could capture the delay. The delay of the gate remember the edge represents the gate ok.

(Refer Slide Time: 60:43)



Now I can compute the signal arrival times using just a variation of that same algorithm, where I was calculating level plus 1 instead of level plus 1, you would just take that edge weight whatever is there and you were to appropriately set. So, that is fine as far as the computation of the level is concerned. There are a number of other adjustments that we could make number of other features that we can capture in this formalism; I could have different pin to pin delays within a gate, right.

(Refer Slide Time: 61:13)

### Other Extensions

- **Different pin-to-pin delays within a gate**
  - use same graph, but different edges representing same gate have different delays
- **Finding earliest arrival times**
  - same algorithm, use MIN instead of MAX
- **Finding shortest path**
  - backtrace on delay graph using min. arrival times
- **Non-zero arrival times at primary inputs**
  - initialise appropriately

NPTEL (C) P. R. Panda, IIT Delhi, 2017 11

(Refer Slide Time: 61:30)

NPTEL ETSC, IIT DELHI



In general, I have such a structure that delay maybe in general different from this delay, it could it depends on what that circuit is doing. This is just a block right inside that block, there may be other things that are happening, if you look at the transistor diagram, you will find that some paths are faster some paths are slower. Can I incorporate that in our representation?

Student: Yes.

Yea, each of these are nodes right. So, I have 3 nodes and earlier this was my graph; how can I change that to present the fact that these delays are different. So, earlier we used a fixed gate delay there. So, that same delay would be annotated on both of them those delays are instead they are different like that; then I could have something like this. I could use the same structure to find other answers like the earliest arrival time instead of latest arrival time, I can say earliest with just a small change you use the min instead of the max.

So, you could do a lot of things instead of the max delay path, you can find the shortest path by doing the appropriate backtracking using the minimum arrival time instead of max. If you had a non zero arrival time at the primary inputs; that is also one of the assumptions we said that all primary inputs are available at time 0, if it is non zero, then how would you adjust this? Actually the way we have designed this graph there is a placeholder already, these represent the inputs right. So, we can put numbers there also instead of just being 0, this can be something else and the rest of the graph does not change.


So much of this can be done of course, the problem with timing analysis is that it is not as simple as this; we have already alluded to this in other discussions. In general there are other dependencies, it is not constant right if it is constant then it is fine we could use all of these strategies, but the reality is that there is a more complex function for example, it is a function of the output load right. So, there are other dependencies there is a function of the input slew.

The rate at which the input is changing that too is determining the load. So, all of this does need to be taken into account, but I am just pointing out some very simple extensions using the same formalism nothing different. We could also discuss in this context a required arrival time for a signal.

(Refer Slide Time: 64:34)

## Required Arrival Times

- Also relevant to discuss required arrival times
  - Earliest time when a signal is required to arrive at a node
- Algorithm
  - start at output node
  - traverse delay graph backwards



(C) P. R. Panda, IIT Delhi, 2017

12

So, the earliest time when a signal is required to arrive at a node that you just start at the output node instead of just computing the delay let us say you want to find some bottlenecks or something like that you want to say, if that is the max delay that I can afford then what is the required arrival time at each of the inputs of the gates, you start at the output node and you traverse the delay graph backwards, you can annotate the required time instead of propagating from 0 and from the inputs towards the outputs you could also do the reverse ok.


(Refer Slide Time: 65:18)

## Slack

Slack = (Earliest Required Arrival Time) - (Latest Arrival Time)

Slack indicates Sensitivity of Circuit to Signal

- If positive Slack T at node V
  - Signal at V can be delayed by T without affecting critical path of circuit
- If Slack = 0, V is on critical path
- If negative Slack at V
  - Signal will not meet requirement: will arrive too late
  - Indicates desired speed-up



(C) P. R. Panda, IIT Delhi, 2017

13

In the process, you could compute; what is called a slack which is the difference between the earliest required arrival time and the latest arrival time at the input of a gate say that slack indicates the sensitivity of a circuit to a signal in the sense that if there slack is positive that is required arrival time is something, but the latest arrival time according to the circuit that you have designed and according to the numbers that we have for the gates is something that is less than the required arrival time ah; that means, that there is a positive slack at that node and that node means and that net that corresponds to that node that signal that is there at the node could be delayed a little bit without affecting the critical part of the circuit ok.

So, there is a gap it means that this is a good situation to be in or at least it is a situation where we are not too much concerned about that particular node because we have our signal at the right time. In fact, it is before what we actually need. So, if a slack is positive it means that that signal as of now we are not worried about that particular net and signals going on that net if that slack is 0, then that node is on the critical path gets delayed a little bit then that delay will propagate all the way to the output if it is negative it means that that signal will not meet the timing requirement right it will arrive too late what does that tell us. So, that is important information we have to do something about it there is some speed up some acceleration that is needed as of now because we are not meeting the required time.

Student: Would it would it (Refer Time: 67:20) because.

Yea, this is defined as every node of these every node of these circuits, right.

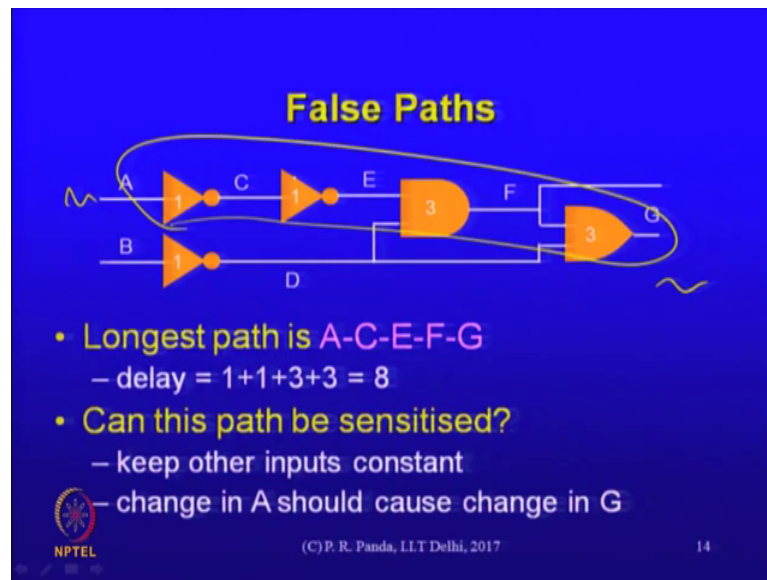
Student: If an intermediate node itself not able to meet the slack; that means, the subsequent will not be able.

Yeah.

Student: To recover in this.

Yeah.

(Refer Slide Time: 67:48)



So, the slack will be 0 for all nodes of the critical path if it is 0 for one node (Refer Time: 67:48) there are some things that complicate the timing analysis computation this should be expected if it was that simple then it would not be a valid subject of analysis and discussion here, this has to do with a very interesting phenomenon called false paths consider the circuit longest path according to just the structural computation is this right it is this.

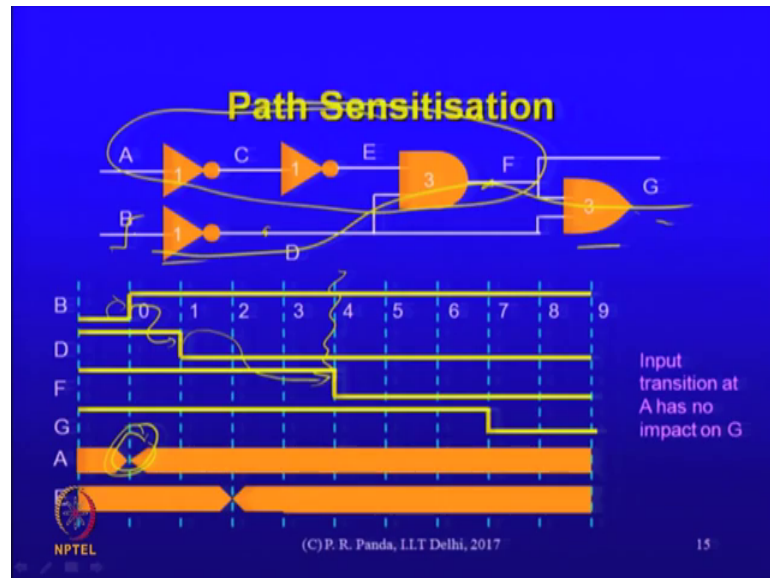
So, while it seems fine the question is can this path be sensitized? So, for example, if you run a simulation with data will you actually get that delay or not you keep other inputs constant delay of course, means that if that is my critical path then I should be able to recreate a situation where the rest of the circuit is constant and because of a change in that input there is a change that is reflected at the output after a certain time and time hopefully should be one plus one plus 3 plus 3 if that is the critical path do you think that is possible for this circuit.

Student: Or followed by the AND is where the limitation is B will not be. So, the B has to be the B bar has to be 1 at the same time for or to function, it has to be 0. So, it can be true. So, that is why G cannot be.

Yea, there is some analysis say this is not merely a structural analysis there is problem is it depends on what that gate is doing and depending on; what it is doing a path that

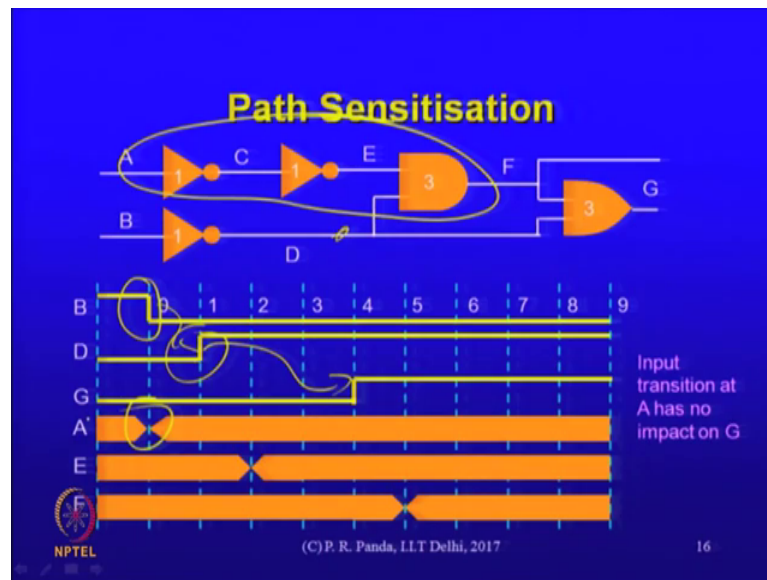
seems structurally as though it is a critical path might not actually be sensitive; this is what we mean by a sensitizable path.

(Refer Slide Time: 69:56)



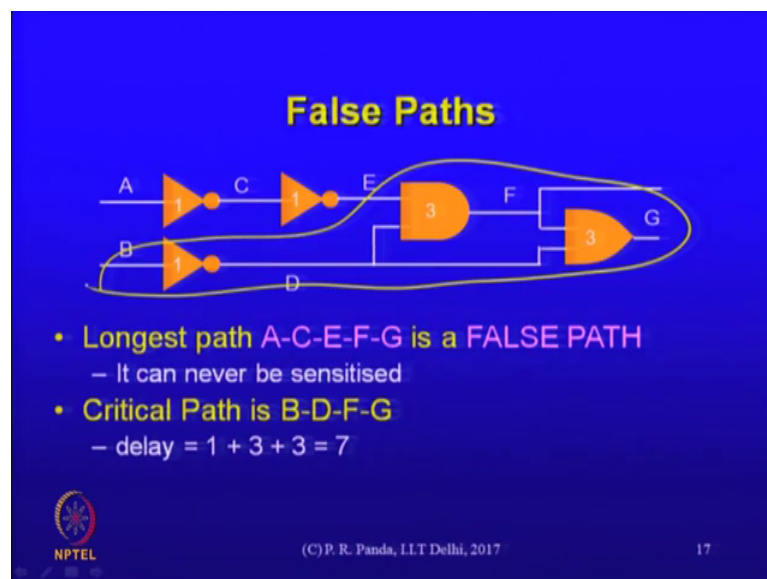
So, the issue is the following suppose b goes from 0 to 1, but if you look at b going from 0 to 1, then that goes from 1 to 0 because this went from 1 to 0 that output tend to 0 to 3, later, basically at that at time for F got set, remember this it is irrespective of the fact that something happened at A or did not happen at A. So, it is that path that got sensitized really if the transition on B was 0 to 1. So, the A has no impact input transition at A has no impact; what if it was the other way around, B changed from 1 to 0.

(Refer Slide Time: 70:40)



Then D changed from 0 to 1 which immediately cause G to change irrespective of anything that happened there, ok, either way, the transition in was not really dominating this computation of the critical path.

(Refer Slide Time: 71:04)



So, this longest path that we had identified structurally is a false path because it can actually not be sensitized, right. So, the critical path actually is this in this circuit.

Student: Sir, sensitized mean that it cannot be evaluated as (Refer Time: 71:26) the signal trough.

So, you should be able to control the situation in a way that you hold the rest of the circuit stable and a change in the input should reflect in a change at the output.

Student: (Refer Time: 71:39).

Right; that is sensitization, we will get to a clearer definition of the sensitization informally this is what it is just let us suspend it here.