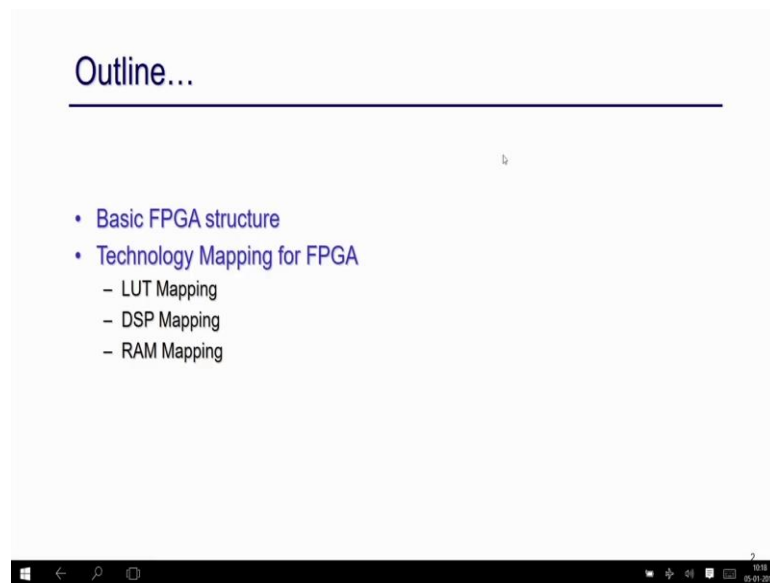


C-Based VLSI Design
Dr. Chandan Karfa
Department of Computer Science and Engineering
Indian Institute of Technology, Guwahati

Module - 12
Recent Advances in C-Based VLSI Design
Lecture - 40
Technology Mapping for FPGA

Welcome everyone, so today we are going to discuss about this Technology Mapping for FPGA.

(Refer Slide Time: 00:52)



Specifically, we are going to talk about the basic FPGA structure what are the components of FPGAs. And then we will talk about the technology mapping for FPGA specifically LUT mapping, DSP mapping and RAM mapping.

(Refer Slide Time: 01:02)

FPGA

- Field programmable Gate Arrays (FPGAs) are pre-fabricated silicon devices that can be electrically programmed in the field to become almost any kind of digital circuit or system.
- For low to medium volume productions, FPGAs provide cheaper solution and faster time to market as compared to Application Specific Integrated Circuits (ASIC) which normally require a lot of resources in terms of time and money to obtain first device
- FPGAs takes less than a minute to configure and they cost anywhere around a few hundred dollars to a few thousand dollars.
- Programmable routing interconnect of FPGAs which comprises of almost 90% of total area of FPGAs larger, slower, and more power consuming than their ASIC counterparts.

So, if you look into the FPGA field programming programmable gate array which is basically a pre-fabricated silicon. So, it is already fabricated, but the advantage is that it is electrically programmed in the field and it can almost implement any kind of digital circuit.

So, it already has a fixed structure component all are already fabricated in the board and you can actually map any kind of logic circuit into that, advantage of this FPGA is that it is very fast. So, if the time whenever your RTL you can easily just map into this FPGA using some synthesis tool provided by the industry like Altera or Synopsys, Cadence, and you just use it.

So, on the other hand in the ASIC flow the whole process is very time consuming and it does not make sense to go for it, when the number of chips required is very less. If you just need one or two kinds of thing, chip or design you do not need go for ASIC because it is a time-consuming process. And it is a costly also.

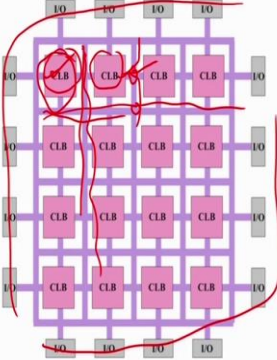
So, that is why FPGA is very popular because low time to get it done and it is very popular in prototyping also before making the actual ASIC design or ASIC (Refer Time: 02:24) chip we just map it to FPGA and see whether the simulation is giving correct results or not whether the design is working correctly or not. So, FPGA is very popular.

The disadvantage of FPGA is that it is little bit slow because it is a programmable interconnection. So, it has all the possibilities inbuilt, so it is slow, also power consuming and also larger compared to the ASIC design. So, for a given design, ASIC design will be much smaller than a FPGA design. But because of the other advantages FPGA is widely used in the EDA industries.

(Refer Slide Time: 03:02)

Basic FPGA structure

- Normally FPGAs comprise of:
- Programmable logic blocks which implement logic functions.
- Programmable routing that connects these logic functions.
- I/O blocks that are connected to logic blocks through routing interconnect and that make off-chip connections
- SRAM-Based Programming Technology
- Flash Programming Technology
- Anti-fuse Programming Technology



The diagram illustrates the basic structure of an FPGA. It features a central grid of Configurable Logic Blocks (CLBs) arranged in a 4x4 pattern. Each CLB is represented by a pink square. Surrounding this grid are I/O blocks, shown as grey squares, located on the top, bottom, left, and right edges. Red circles highlight the CLBs and I/O blocks, and red lines indicate the routing interconnect between them, showing how they are interconnected.

So, if you just look into the FPGA in details. So, this is the kind of overall structure of FPGA it consists of configurable logic blocks they are just run-in arrays and they are interconnected through this interconnection units. So, these are the interconnection units and the IOs are placed usually in the outer interface. And this logic block is basically is configurable, so where you can actually map your design.

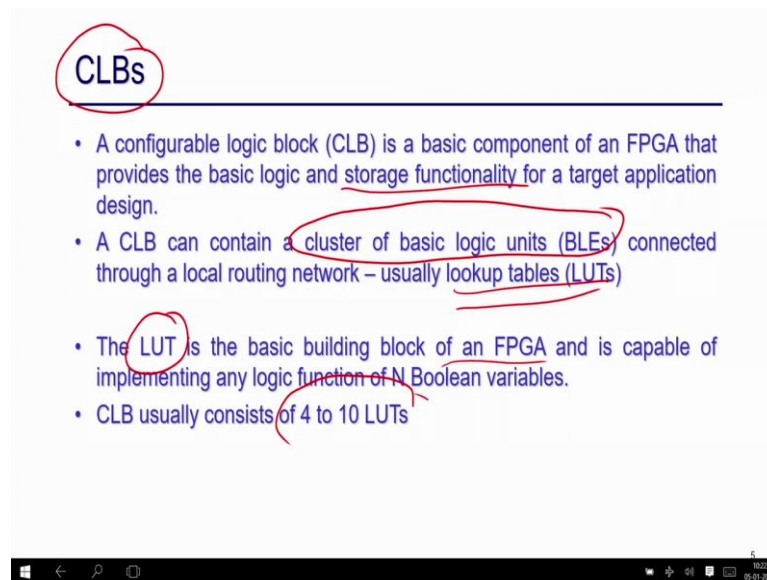
So, it consists of both storage as well as the logic units. So, your design has to be mapped here and the interconnection has to be done through this interconnection unit. And all this logic block is programmable as well as this interconnection is also programmable. So, you can make connection from anywhere to anywhere. So, anywhere the connection can be made through these interconnections.

And primarily this programmable component is mapped through SRAM based programming technology, also in some scenario this can be flash programming technology or anti fuse programming technology.

(Refer Slide Time: 04:05)

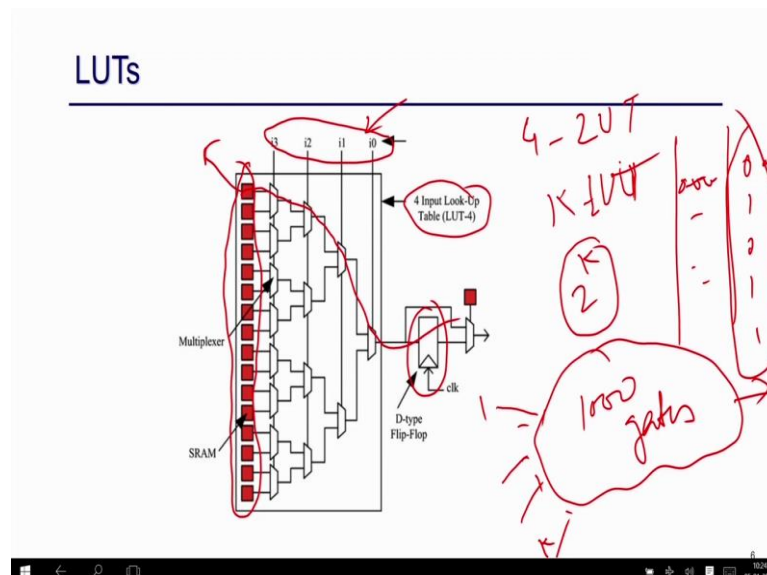
CLBs

- A configurable logic block (CLB) is a basic component of an FPGA that provides the basic logic and storage functionality for a target application design.
- A CLB can contain a cluster of basic logic units (BLEs) connected through a local routing network – usually lookup tables (LUTs)
- The LUT is the basic building block of an FPGA and is capable of implementing any logic function of N Boolean variables.
- CLB usually consists of 4 to 10 LUTs



So, if you go into the CLBs, configurable logic blocks, this is the primary component where your design has to be mapped. So, it consists of primarily a cluster of basic logic units which is nothing but lookup table. And also, it consists of some memory also just for storage like flip flop and RAM and all other components. So, essentially this LUT lookup table is the building block of an FPGA because this CLB consists of several 4 to 8 LUTs or maybe more in modern FPGAs.

(Refer Slide Time: 04:46)



So, if you just go into this LUTs it is basically lookup table. So, it consists of a memory which is SRAM based 1 bit memory and it has some inputs and it decides which memory bits should go at the output. And it also has a register at the output. So, based on the number of such input select line we say it is a 4-input lookup table, it is a 4 LUT. So,

basically K LUT depends on the number of inputs. So, what does it mean? So, a LUT if it has a K input, it can implement any Boolean function of 2 to the power K.

So, possibilities of bits are there, so any Boolean functions of K input can be implemented here. So, if you just think about a Boolean circuit it consists of maybe 100 to 200 gates it has K inputs, 1 to K and 1 output. So, that Boolean circuit it consists of may be consists 1000 gates, it does not matter how many gates are there. So, you have to just capture the input output co-relations. So, if the input is 0 0 0, then what is the output, 0 0 1 what is the output and so on.

So, basically you need that table for all possible inputs and corresponding output for those inputs. So, that is the output bit and this output will be stored in this SRAM. And when your input is say, 0 0 0 then the output is 0. So, this bit will be coming at the output. So, this is how that your Boolean circuit will be implemented.

So, it is not that all gate will be mapped here, we capture the functionality of any Boolean circuit where we have K inputs 1 output that particular possibilities of output we will capture and we will store that data pre store in that SRAM the memory bits. And this select line the inputs will actually select which bit has to be the output, which output should come at the output. So, that is something is lookup table, and so primary component of this CLBs are lookup table.

(Refer Slide Time: 06:57)

Interconnections

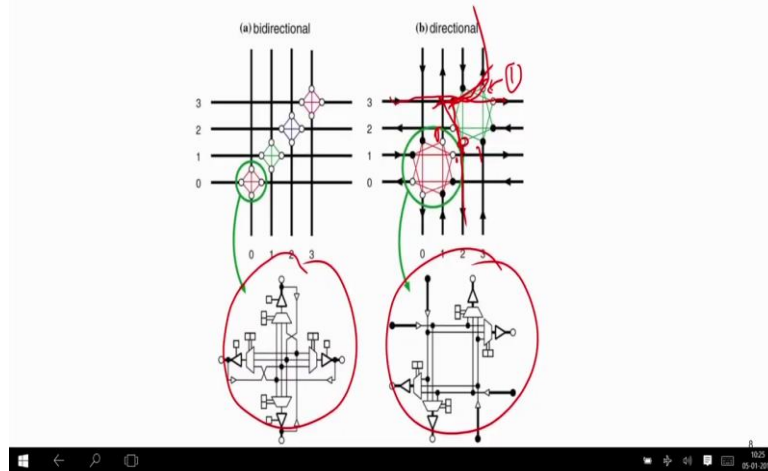
- The routing interconnect of an FPGA consists of wires and programmable switches that form the required connection.
-
- These programmable switches are configured using the programmable technology

The diagram illustrates the routing architecture of an FPGA. It shows a grid of Configurable Logic Blocks (CLBs) and Switch Boxes (SBs). The routing channels are labeled as Channel Width (W), I/O Block, Configurable Logic Block (CLB), Horizontal Routing Channel, Vertical Routing Channel, and Connection Box. A red path is highlighted through the grid, showing the connection between various components.

And also, if you just look into the interconnections, I also said these interconnections are also programmable if you look closely on that interconnection. So, these are the CLBs and the interconnections we have switch box as well as circuit box.

(Refer Slide Time: 07:14)

Interconnections

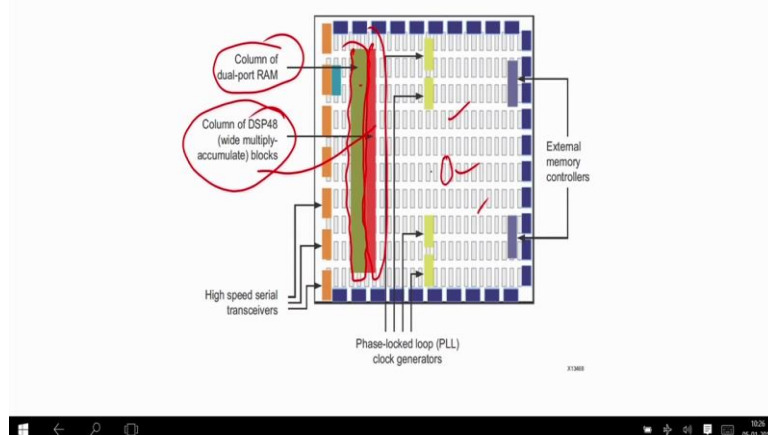


So, the switch box if you go into this there are interconnections either bidirectional or all possible directions, so you can understand here. So, whenever some data come it can go this way, this way, that way and all the possible ways, so this these are the control bit. So, if you just set this bit as 1 then this data will go this way, if you set this bit as 1 then this data will go this way and so on. So, if you just set this one 1 and this to 0. That means, this input will come and go through this way.

So, this is how the whole circuit is implemented like this bidirectional and unidirectional. So, this interconnection has to be programmed also, based on the actual interconnection we are going to set these bits 1 or 0 to make the actual interconnections. So, this is how the overall structure of this FPGA.

(Refer Slide Time: 08:06)

DSPs and RAMS

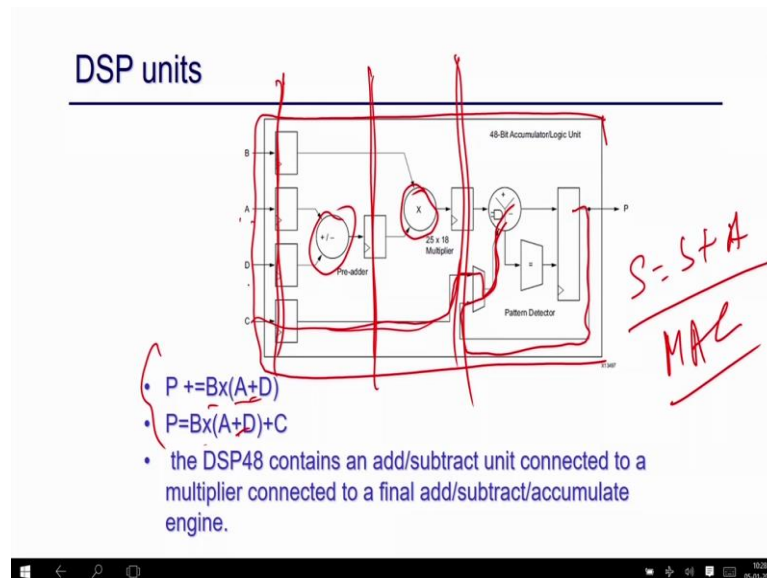


So, if you look into an FPGA, it does not only consist of LUTs. So, CLBs consist of several LUTs and also some of the CLBs are DSP unit. If you just look into this one specific structure here you can see that a column specifically used for dual port RAM there is another column which is specifically used for DSP 48.

So, that means, in this overall structure as I said these are all CLBs, some of them will be used specifically for DSPs, some of them is for RAM and some of the other is for logic units where is basically, we have this lookup table-based structures stored. And in a CLB, we do not have only one we may have say, 10, 12, 14 number of logic units are stored together and they are interconnected.

And also, one register is there just that we can use that register to store some data. So, this is the overall structure of a DSP if you just look into the connection of FPGA.

(Refer Slide Time: 09:13)



If you go into the DSP structure, so DSP is basically the very first implementation of a multiplier. So, if you just look closely in the DSP structure you can see here in the DSP block there is a multiplier. So, the primary purpose of this DSP unit is to perform the multiplication in very first way. If we just map a multiplier into this lookup table-based design it will be very big and it will be slow, on the other hand it is a first implementation of a multiplier and this is fabricated within the FPGA board itself. But in the DSP unit it is not only a multiplier it can have a pre adder, it has also have set of latencies.

So, we have three set of latencies here and this is MAC there is a sum, S equal to S plus A, accumulator. So, this is what is called accumulator. So, you can multiply and accumulate, MAC. So, this is what is happening here also we can do a post addition. So, this can also come here and you can do a post addition.

So, you can do specific type of operation like you can do B into A or you can do A plus D into B or you can do A plus D plus minus, this is plus minus and then you multiply this with B and then you add C. So, all kind of possibilities are there. So, all this kind of

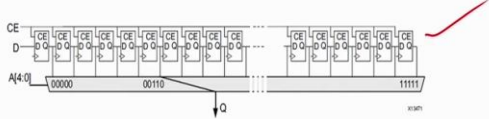
structure when you have in your design those can be mapped to a DSP unit and also you can pack some of the setup register also. 3 layers of registers are there.

So, it has 3 latencies either you can bypass or you can use them. So, this is the abstract structure in in in design you can actually bypass any set of registers. So, this is what this DSPs and also it has some memories RAM.

(Refer Slide Time: 10:54)

BRAM and other Memory units

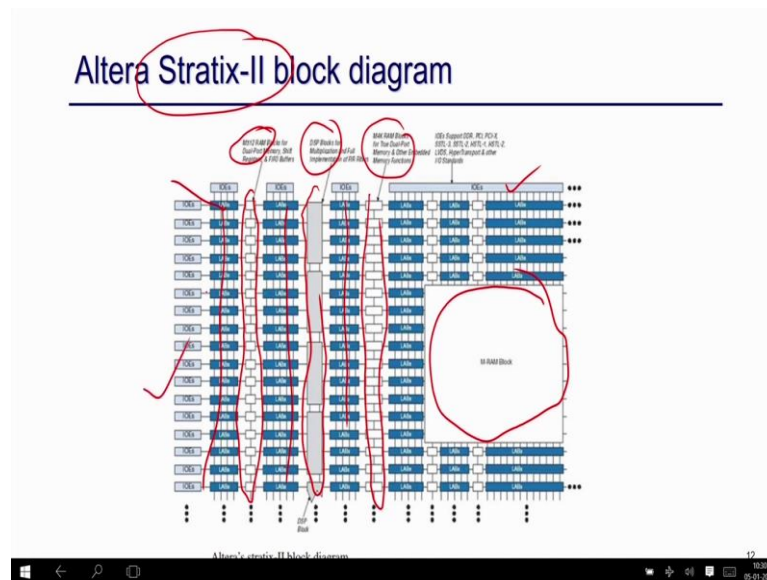
- The FPGA fabric includes embedded memory elements that can be used as random-access memory (RAM), read-only memory (ROM), or shift registers. These elements are block RAMs (BRAMs), LUTs, and shift registers.
- The shift register is a chain of registers connected to each other. The purpose of this structure is to provide data reuse along a computational path, such as with a filter.



The diagram illustrates a shift register structure. It consists of a chain of registers connected in series. The input is labeled 'D' and the output is labeled 'Q'. The diagram shows a sequence of registers with 'Q' outputs of one register connected to the 'D' inputs of the next. A red checkmark is visible to the right of the diagram.

So, where you can store the large arrays, large memories, large arrays RAM and ROM also it has some shift registers. So, shift register in hardware you know there is a specific kind of register where you have the data in every cycle you shift the data by one bit by left or right, left shift or right shift. So, in specifically in filter application the shift register is widely used. So, some of the registers I mean some of the memory is also shift register inside the DSPs inside the FPGAs.

(Refer Slide Time: 11:30)



So, this is the overall configuration of a FPGA and now if you just look at specific case an example Altera Stratix II Board you can see here the LAB is nothing but the CLB. So, in Altera this CLB is called as LAB. LAB means logic array block, so which is a CLB. So, inside that, so you have series of LABs and some of them are actually smaller RAM.

So, these are the smaller RAM you can see this is 512 bits RAM, this is also RAM, this is DSPs, this is DSP block these are LAB. And again, there are smaller RAM, 4K bits RAMs are there and there is a bigger RAM also we can see and the I/Os are here. So, these are also I/Os. So, this is a specific structure for Stratix II block.

(Refer Slide Time: 12:22)

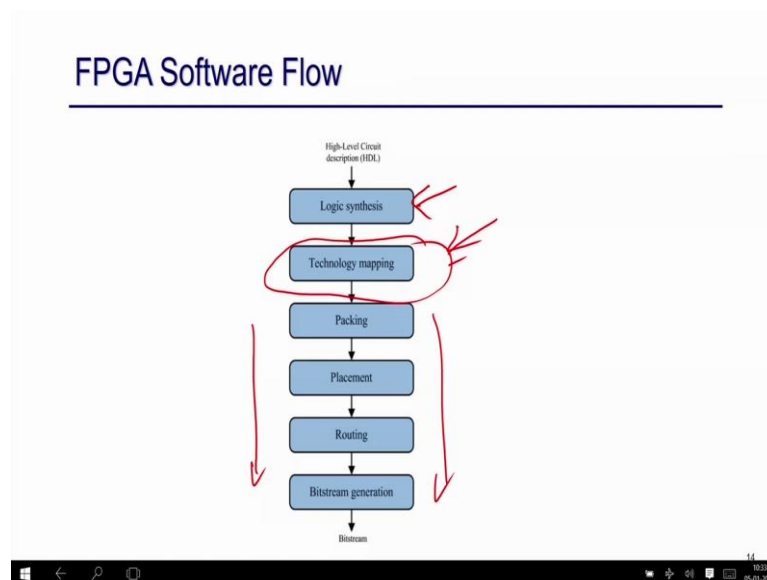
- ### Altera Stratix-II
- Each Stratix II LAB consists of eight Adaptive Logic Modules (ALMs).
 - An ALM consists of 2 adaptive LUTs (ALUTs) with eight inputs altogether. Construction of an ALM allows implementation of 2 separate 4-input Boolean functions.
 - In addition to lookup tables, an ALM provides 2 programmable registers, 2 dedicated full-adders, a carry chain, and a register-chain.
 - Full-adders and carry chain can be used to implement arithmetic operations, and the register-chain is used to build shift registers.

And if you have some detailed data like so, this LAB which is basically CLB, CLB consist of 8 ALM and each ALM consists of 2 LUTs. So, that means, effectively in a CLB we have 16 kinds of LUTs also we have other input like that two dedicated full adder is also implemented there a carry chain and a register chain.

So, what is the importance of this carry chain? What is the utilization of this? Whenever, you have a adder circuit, so it has a carry and if the carry has to route through this circuit block to somewhere else it will slow, you want to execute the add fast. So, what it actually happens inside the LAB whatever the 16 LUTs are there, they are connected.

So, their outputs are connected to each other locally not through the other circuit block, so this is carry chain. So, whenever you implement an adder inside a CLB that carry will not be routed through the switch box or circuit box. It will be internally connected and it will very fast connection path, so that actually you can do these additions fast. So, this is how this adder is also implemented in the LUT in faster manner. So, this is something the detail about the Stratix block.

(Refer Slide Time: 13:45)



So, this is all I just talked about just to give a brief idea, what is FPGA and what are the different components there inside the FPGA. Because when you are going to talk about the technology mapping, we need to know what are the components there which can be used efficiently for your designing purpose. Now, if we just look into our software flow. So, EDA software flow we know this high-level synthesis, there are logic synthesis and physical synthesis. So, after logic synthesis you know in the physical synthesis we have this floor planning, placement, routing.

So, in FPGA whenever after the logic synthesis is done, now you have to map your circuit into this LUTs in DSPs and this RAMs because you do not have any gates in the FPGA board. So, whatever the circuit is there, how many gates are there you have to find out. So, if you know your design has a K input LUT, so 4 input LUT.

So, you have to figure out all possible such 4 is to 1 kind of component or the pattern in your design and that has to be mapped into in a single LUT. So, you cannot map all the gates individually in one LUT, that will be more costly. So, we have to find a pattern and we have to map them into the LUTs, we have to find the multiplier, we have to map them into DSPs, we have to find the big arrays you have to mapped them into RAMs.

So, that is what is called technology mapping and once this is done then it will go through the normal procedure like placement routing and all other stuff. Because finally, you have to map them into the actual hardware, you have to place them into specific LUTs and then you have to also make the connections which is routing.

So, those are the normal steps as like ASICs, but in technology mapping we have to do a specific task which is specific to FPGA and that is the discussion topic for today. So, we are going to discuss more on how to map a big circuit into FPGA into LUTs into DSPs into RAM efficiently. So, that is what we are going to talk about today more.

(Refer Slide Time: 15:38)

FPGA Technology Mapping

- Technology Mapping:
- ASIC - Cell Library (e.g., NAND, NOT, FF)
- FPGA - fixed architecture
 - LUT
 - DSP
 - RAM/ROM
- In FPGA:
 - Gate level design → LUTs
 - Multiplier/MAC → DSP
 - Memory/Large Registers → RAM/ROM

So, that is what is explained again here that for ASIC the difference is that we have a cell library. So, in a cell library typically we consider a complete set of gates, say like NAND gate and NOT gate. So, through which you can implement any kind of gate XOR, XNOR, NAND, NOR and any gate you can represent using NAND and NOT. So, and we have also set of registers like D flip flops.

So, whenever you have a circuit where you might have XOR gate you have some other gates those has to be mapped through this set of gates. So, actually for ASIC you are effectively replacing all your gates using a standard library gate, so that those can be finally fabricated because the library fabrication fully supports only this kind of gates.

So, that is for normal ASIC flow, but in FPGA as I mentioned we have LUT, DSP and this RAM and ROM. So, we have that gate level design, that has to be mapped to LUTs. So, it is not one is to one mapping that 1 gate will go to 1 LUT then it is a huge design

and that is wastage of your resources. Similarly, it is not that you have to identify the multiplier and that has to map into DSPs.

It is not only a multiplier as I mentioned it can be MAC, some pre addition post addition and different combination can be mapped into DSP. So, that has to be done. If you do not do it and if you map your multiplier to a logic gate or LUT then your area will be exponential and maybe if you have large number of multipliers, you may not be able map your whole design into a FPGA.

So, that has to be taken care. Similarly, this RAM mapping. So, the difference from this ASIC to FPGA in technology mapping is in ASIC we have the cell library mapping, from the normal gates to cell library gates mapping and in FPGA during technology mapping we have LUT mapping, DSP mapping and RAM mapping.

So, this is where it primarily differs the ASIC flow versus FPGA flow. Otherwise, this placement, routing all of them more or less has to follow a similar kind of algorithm. So, we are going to discuss more on these three today.

(Refer Slide Time: 17:51)

DSP inference

- In pre-map stage.
- Before logic synthesis
- Identify MAC structure (along with registers) in RTL
 - Sum += (B * (A +/- D)) +/- C
 - Retime nearby registers to bring them into MAC structure
- Bypass (preserve) MAC structure during logic synthesis
- Map the preserved structure during technology mapping stage

The diagram illustrates the internal structure of a 48-bit Accumulator Logic Unit. It features a 28x18 Multiplier block, two adders, and several registers. Red circles and arrows highlight the MAC structure and the re-timing of registers. The diagram shows the flow of data from inputs A, B, C, and D through the multiplier and adders to the output P. The registers are connected to the multiplier and adders to store intermediate results.

So, let us start with the DSP. So, as I mentioned that the DSP unit is capable of doing many things. So, it has a pre adder, we have a post adder, we have this MAC, summations and also, we have three set of registers. So, the objective is to map maximum kind of sub structure into this structure because if you are just doing an addition in this block then also occupying the whole DSP. If you want to do only one multiplication here then also you are actually occupying the whole DSP, because this DSP cannot be used for something else.

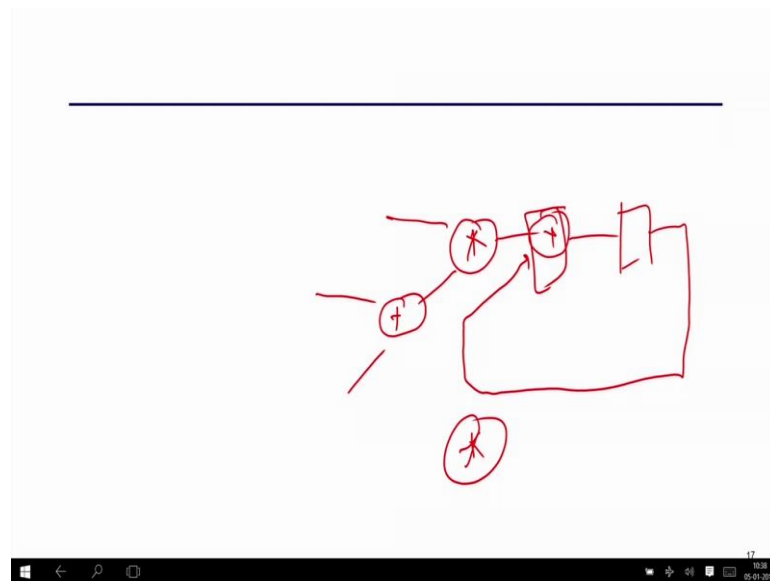
Or if you just do the whole thing, if you just do B into some pre addition and then post addition and then you do the MAC summation then also it is going to occupy the whole thing. And also, if you just pack some register inside then what will happen? The delay, because if you just have a path the which consists of adder multiplier followed by adder it will be a big path though it is very fast, but it is a big path.

So, it is advisable that you pack at least some registers into this structure also. Now the question is, how we can do this efficiently? So, that is something is called DSP inference in the FPGA flow. So, you can understand once you have done the logic synthesis, after logic synthesis what happens, everything is represented by gates. So, in that case it is very difficult to identify which is multiplier, which is adder and something like that.

So, the problem is that we have to identify this multiply and accumulate structure before logic synthesis which is called pre map stage. So, before logic synthesis we have to identify this MAC structure efficiently and then you have to preserve those structures, so that logic synthesis bypasses those structures. So, they should not map those adders, multipliers, those structures into gate level design. Then we are not able to map those things into DSPs.

So, the idea is here is that you identify those MAC structure before logic synthesis, preserve them. Inform logic synthesis that you should not do any kind of synthesis in this part of the design. And then during technology mapping you just map those circuits into DSP state. This is the overall flow. And again, I just mentioned you try to find maximum structure possible. So, basically the way the algorithm works is something like you identify all your multiplier in the RTL design.

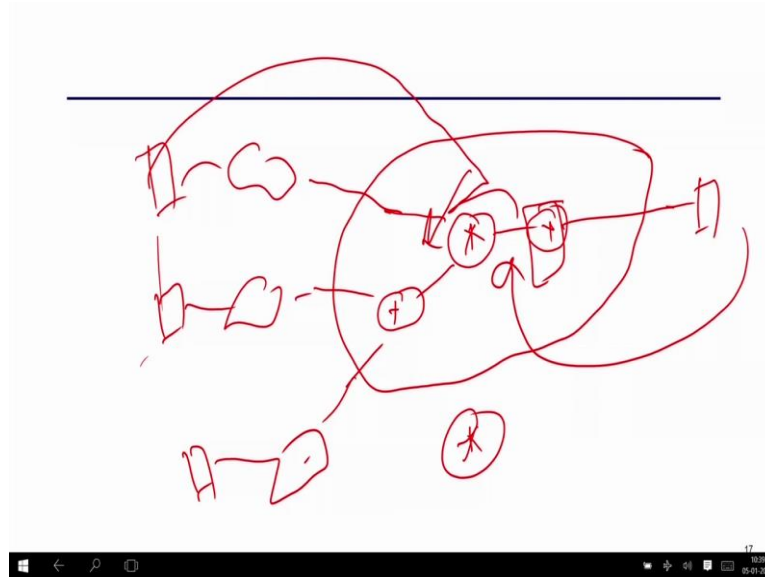
(Refer Slide Time: 20:12)



So, you identify all your multiplier in your design. So, then you can map that, but you try to expand that structure. So, then you try to find out some structures. So, initially you find only a MAC, only a multiplier then you try to find out whether there is any pre adder or not. So, if you find a pre adder then your structure will be like this and then you try to find out is there any post adder or not. So, this is also you try to find out and then you try to find out whether is there any multiply and accumulate options are there or not.

So, you basically just try to start from a multiplier and try to enhance their structure as much as possible. So, you figure out this structure that I have this a pre adder as well as a post adder.

(Refer Slide Time: 21:07)



Then you can actually have this structure. So, this is the MAC structure we are going to preserve but there is no register here. Then the next objective will be to try to do some local retiming just to move some register, nearby register. There may be some registers here some register here maybe there are some other logics here, but finally, there is some register. You try to move these registers into this structure, you move these registers here, you move these registers here, so register here. So, that you can pack those registers as well in the MAC structure.

So, this is the overall idea that we start from a multiplier we try to expand that structure which is supported by a DSP which is basically have a pre adder, post adder or a accumulate those structure once you figure out the structure you try to figure find out the nearby locations. What are the registers available? Can we move the register inside this structure? If you move this structure then the whole structure will be having faster execution in the DSP block.

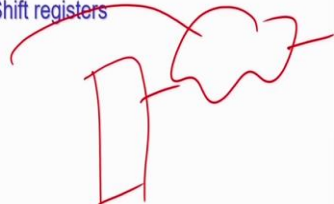
So, this is something is the overall approach in DSP inference and when you find this big structure you preserve it, you do the normal logic synthesis and then you map this preserved structure into the DSP block during technology mapping state. So, this is the overall DSP inference flow in any normal area synthesis FPGA synthesis tool.

(Refer Slide Time: 22:30)

Memory Inference

- In Pre-map stage
- Identify large arrays for mapping them into RAM
- Check for access consistencies
 - RAM has at most 2 ports
- Identify shift registers and preserve
- Maps the preserve structures into RAM/Shift registers

Handwritten notes: 1W, 1R, 2R, 2W



Similarly, if you just go for memory inference again it has to be done in the pre map stage because if you just map that whole array into this logically because of the registers then it will be a problem. So, identify the big arrays and then you preserve that, so that you can map them into RAM. Again, the problem here is something like the RAM has a fixed number of ports, maximum two ports may be one write port one read port or maybe two read port or two write port, so this is the possibilities.

So, based on the access pattern of that register we have to decide whether this can be mapped to a register or any RAM or not if it has a lot of accesses in a same clock maybe you have say 4 access to that particular array then probably it cannot be directly mapped to a RAM maybe you have to do some kind of modification. Some synthesis tool does that you try to break the whole array into two part or you break column wise or row wise.

So, that access can be differentiated and we can create a block where we have maximum access at a clock and then we can map that whole thing into a RAM. So, those all the intricacies are there I am not going to detail of that, but the bottom line here is something that you identify those structures you make sure that your access is something which is supported by the RAM block in the FPGA and then you try to preserve that and map it into the RAM block during technology mapping.

Also, there are other possibilities here like if you have some asynchronous reset to that particular memory then it cannot be mapped to a FPGA block. Then probably you have to add some extra logic around the RAM, you map that particular big array into the RAM and just to overcome that asynchronous reset you add some extra circuit.

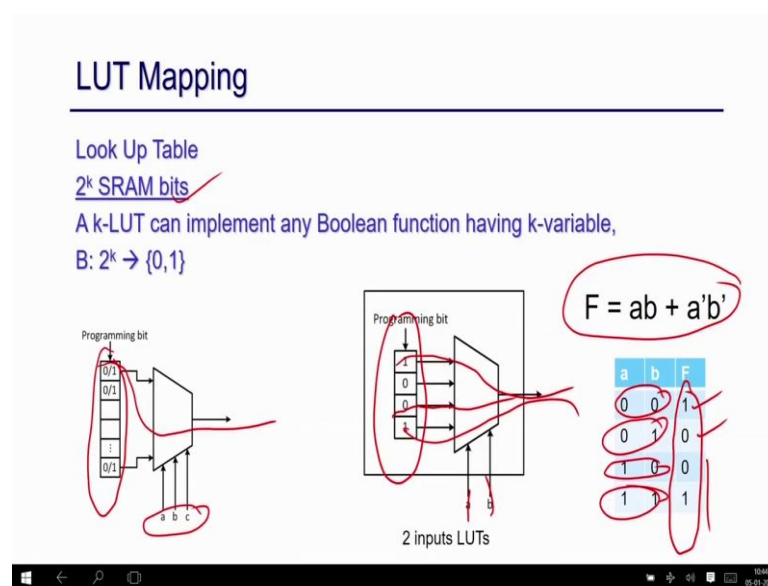
So, this is your RAM map and some extra circuit that will talk about that asynchronous reset. So, that we will discuss in separate section, but the overall idea here is not very obvious, just find and map it to RAM. There may be lot of intricacies like asynchronous reset, how to handle that you might have some extra logic that has to be created just to

handle that. Or, you have maybe a more accesses than you have probably split just to make sure that you have sufficient number of accesses which is supported by FPGA.

And then you map it to the RAM block of this FPGA. Similarly, that you have to find out the shift register identification shift register pattern in the RTL you preserve it and you map it to shift register. So, this is all is done. So, usually all these things are done before logic synthesis preserved because after logic synthesis this structure may not be very obvious to identify.

So, usually most synthesis tool does all these things before logic synthesis, but actual mapping happens after logic synthesis during mapping stage, but identification of those kind of structure is done before logic synthesis.

(Refer Slide Time: 25:27)

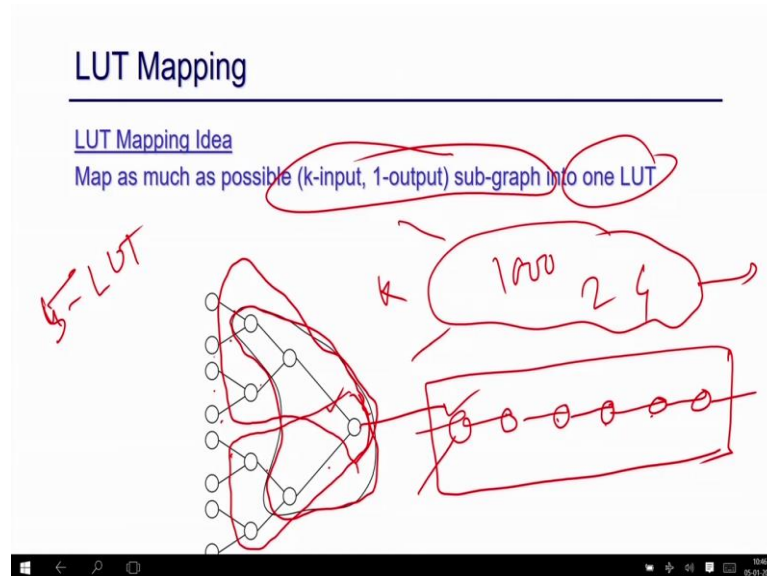


So, now, we are going to talk about this LUT mapping. So, LUT mapping is something lookup table as I mentioned earlier that it has 2 to the power k number of SRAM bits and based on the input value, we decide which output should go at the input. So, this is how the lookup table looks like. So, if you just take a specific example suppose I have this function. So, my output is like this if a and b is 0 0 then output is 1, 0 1 then output is 0 and so on.

So, I am going to store these outputs in this RAM, 1 0 0 1 and this is the select line of this MUX when this is 0 0 this bit will come at the output. The MUX structure is something like that when this is 1 0 this bit will come at the output. So, when the output is 1 1 then this bit will come at the output, this is how we are going to map. So, now, this is the primary part of the technology mapping because you can understand in a big logic circuit where we have millions of gates and our objective is to map all the gate into some LUT.

So, this is the overall objective and as I mentioned earlier also, I mean you just map 1 gate into 1 LUT it is not acceptable. Because finally, you are not going to implement any gate into this LUT we are just implementing that input output relation.

(Refer Slide Time: 26:55)



So, the idea here is like you identify a big structure where it has k number of input and 1 output. So, that you can map this k input into 1 output sub graph into 1 LUT. So, size does not matter here it may be a circuit where we have only k inputs and 1 output and maybe consisting of 1000 gates, it may be consisting only 2 gates or maybe 4 gates, but the big structure all the things can be mapped to a single LUT.

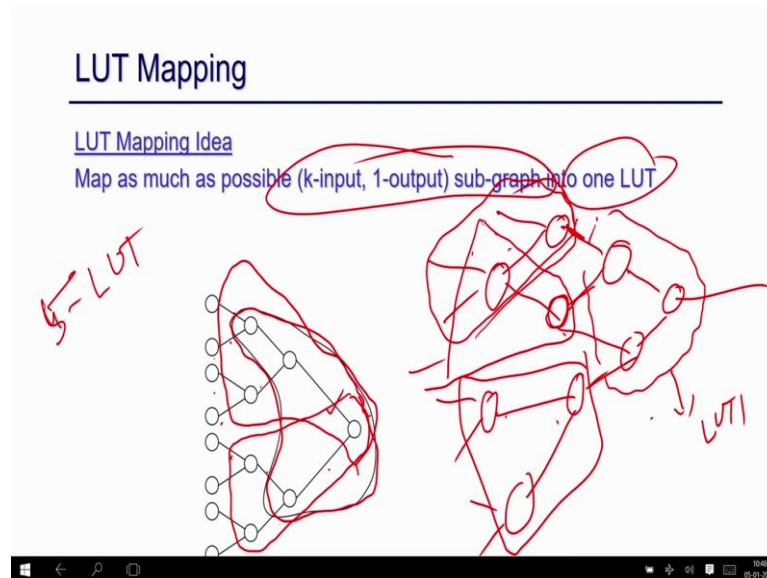
And also, there is an important point here is that so initially suppose there is a structure like this. So, this is a structure like this. So, the delay of combinational delay is 1, 2, 3, 4, 5, 6, 6 gate delay, but if you map in this whole structure into 2 k LUT. That means, 2 input LUT then the delay of that particular thing will become delay of 1 LUT.

Because finally, when you are actually executing this into FPGA it is just 1 LUT. So, it is very important to identify the maximum structure which can be mapped to a single LUT. So, this is something we have to do. So, if you just think about how the whole mapping idea works suppose you have a circuit like this you try to find out this kind of a sub structure.

You try to find out a sub structure and here input is 1, 2, 3, 4, 5. So, you can map this whole structure into a 5 input LUT, so it is a 5 LUT. And you can understand here there may be various possibilities here. So, you can suppose if you think about a 5 input LUT. So, you can actually consider a structure like this also. So, this is also a 5 input LUT.

So, here it is had been 1 input 1, 2, 3, 4, 5. So, this is one possibility or you can actually choose this. So, you can choose this also this is also then 1, 2, 3, 4, 5. So, you can understand that given a big logic circuit the possibilities are various. So, you can have different kind of this sub structure possible and also important that if you select 1 substructure the next substructure will be different. So, if you just think about a circuit is like this.

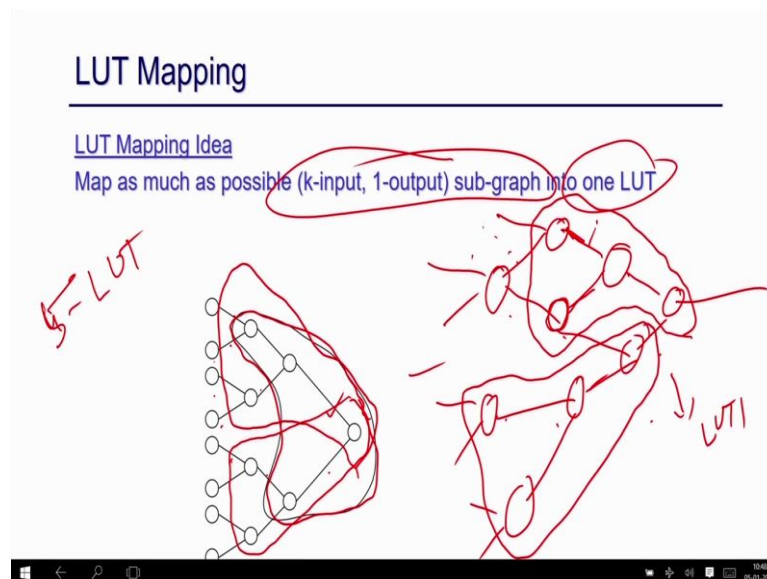
(Refer Slide Time: 29:21)



Suppose you have this I am just considering some arbitrary circuit like this you can see here suppose I want to map this into 4 input LUT. So, if you just select this one then this maps to 1 LUT. So, this is LUT 1, now I have to map some because this one output is coming, I have to create another LUT here. So, just to map because this is 1 output it has to map here. So, there are 2 out, so this is another output. So, I have to map this structure.

So, since this is already occupied here there may be some duplication as well here. So, because I want to get another 4 input 1 output LUT structure here. So, there may be some duplication as well and also similarly since this is 1 output, I can choose this one. So, this is one of the possibilities, now if you just think about instead of doing this I can instead of selecting this.

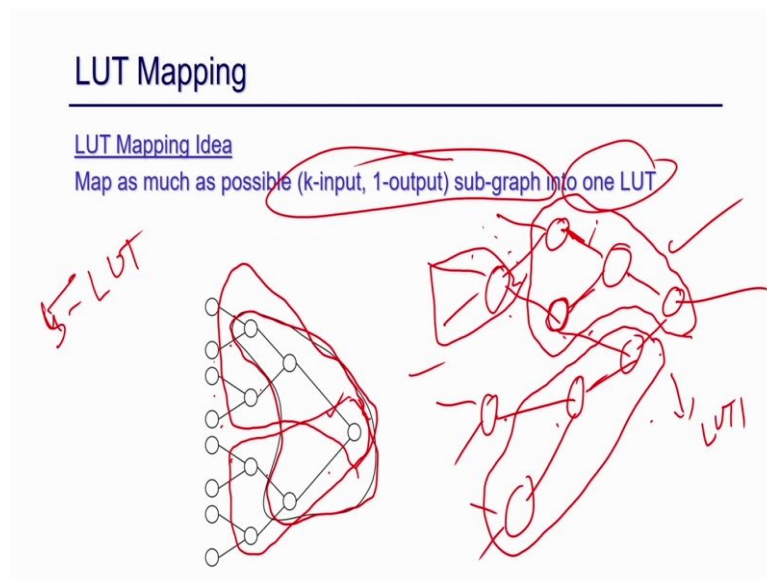
(Refer Slide Time: 30:38)



Suppose I do not select this and I select. So, I do not select this one. So, this is my structure. So, instead of selecting this suppose I select this, so this is also a 4 input LUT, so this is 1, 2, 3, 4. So, now, the next structure will be different. So, initially I select one possibility I select this one then I have to select something here and something here and something here.

Now, if I select this one now my choice will be different now, I have to select another LUT here. So, this is also one possibility, so this is one possibility. So, this is also not exactly correct because one output is here.

(Refer Slide Time: 31:06)



So, I have to select this, so this is 1, 2, 3, 4. So, similarly I have to select something here like this. So, I have to select something here. So, the main point here is that there are huge number of possibilities and one choice decide the next choice. So, the idea is that it is not the choice which is not unique there are various possibilities and it is not like that if you choose something you have a unique choice. If you select some other one way the next choice will be different. So, the idea here is that there may be various possibilities.

(Refer Slide Time: 31:47)

LUT Mapping

Objective

- Area minimization: map into minimum no of LUTs ✓
- Delay minimization: Minimize LUT delay
- Delay-area minimization: Minimize both delay and area

And your objective is to try to map those whole logic structure. So, that all the gates of your design are mapped to some LUT and your objective might be area minimization. May be that mean you try to use the whole of these things using minimum number of LUTs or you try to minimize the delay, the total LUT delay is minimum or you have both you try to optimize area as well as delay. So, your objective can be different, so this is something very important.

(Refer Slide Time: 32:21)

LUT Mapping

- For a given circuit, there is multiple mapping exists and may differ in area and delay.

2-LUT
1 LUT
5 LUT
3 LUT delay
1 LUT
7 LUT
1 LUT delay

And this is in general NP hard problem. So, usually all the synthesis tool you apply has some kind of heuristic approach just to get a good mapping. So, this is something which is important that we cannot have an optimal solution which will give you the optimal mapping always, but some good solution from the heuristic-based approach. And this

also I mentioned that there may be multiple solution existing, multiple mapping exists and may differ in area and delay.

So, we have to apply some kind of heuristic, some cost function just to decide that this is a good selection, this is a bad selection this is better than this selection and then we have to finalize the actual mapping. And also, there is another important factor here, I will just give an example suppose I have this design it is an interesting example. Suppose this structure I have is given to you and my LUT size is 2 input LUT, means the LUT I have only 2 inputs.

So, how many LUT structure is here? And how many LUT is required to map this circuit? And what is the delay? So, one possibility is that you map this into 1 LUT you map this into 1 LUT because again 2 input 1 map into this LUT then you can map the whole this structure into 1 LUT because now this is also 2 inputs and also similarly because this is also into 1 LUT. So, how many LUTs are there, 1 2 3 4 5.

So, in implementation one, I have 1 2 3 4 5 LUTs and on the other hand the delay is 3. So, delay is 3 LUT delay three LUT unit. So, this is the possibilities, but on the other hand you can actually map the whole circuit into 1 LUT because if we just think about the whole circuit, I have 2 inputs 1 output.

So, in implementation 2, only one LUT is required and 1 LUT delay. So, given such circuit if you just choose this way then I need 5 LUTs, the delay of the circuit is 3 LUT delay and here 1 LUT is required so 1 LUT delay. So, the primary point here I am trying to say is that, if you start from here and if you just identify node and whenever you find a 2 input and if you stop here.

So, that means, this is one cut. Cut is something, you start from something and you find out whenever there is a k input is coming to that circuit, so then this is a cut. So, if you find a cut of size k or say 2 here if you stop there then the kind of solution you are going to get may not be the optimum because it may be at this point it grows high. So, if you just find a cut here the input is 1 2 3 4.

So, I have a cut here which is basically of size 4, 4 input cut, but which is not supported by my target architecture because I assume my LUT structure is 2 input 1 output. So, if you just stop here then you will end up having solution of this, but if you ignore that and if you go further, then you might have a reconvergence of path. So, this is what is happening.

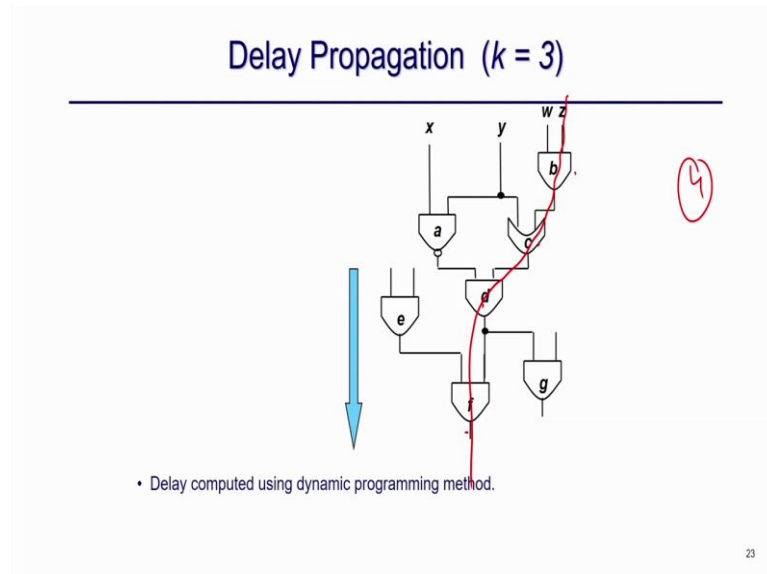
So, it grows and then it shrinks. So, if you have this reconvergence and finally, you end up in a cut of this the whole thing is also a cut and it has 2 inputs. So, this is something also important. Sometime even if your target is 2 you may have to go further because after sometime you might have a reconvergence of path and through which you can actually map a bigger circuit a bigger logic into a single map.

So, this is something also possible. So, that is the core point or the key point is there, there are various possibilities and various ways to actually identify the number of LUTs. And the reconvergence is also important and sometime some duplication also may happen. So, duplication when happens if some node has fanouts, then this node may have

been copied into 2 LUTs. So, this node is actually placed in this LUT as well as this LUT. So, this is called duplication.

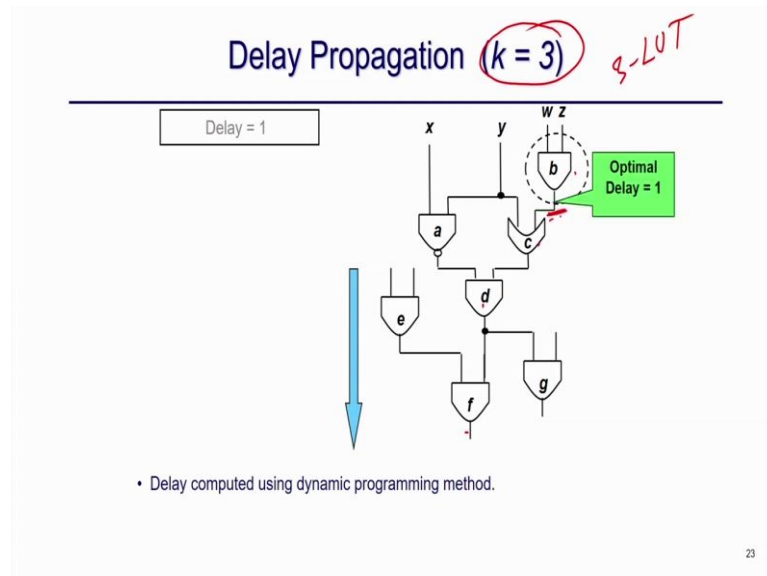
So, duplication is something that should be avoided because unnecessarily I am mapping these things to multiple nodes. So, this is something we will discuss further how to take care of the duplication how to take care of the reconvergence and the other factors. So, we will discuss all those things in detail.

(Refer Slide Time: 37:26)



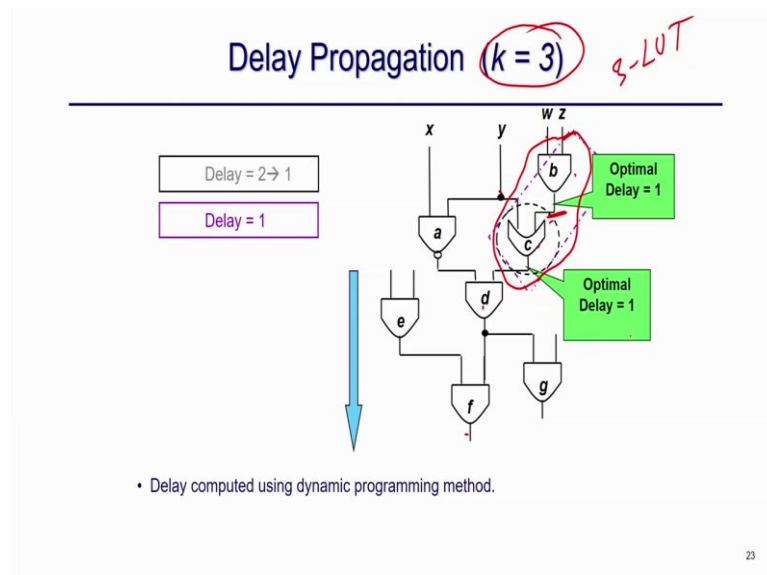
So, before that how to calculate the delay, what is the LUT delay. So, in this circuit you can see there is a path from here to here. So, this is the maximum longest path and 1 2 3 4, so 4 unit of logic delays. So, logic gate delays in the normal RTL circuit, but if you map these 2 into LUTs what will be the delay. So, this is something we are going to understand.

(Refer Slide Time: 37:52)



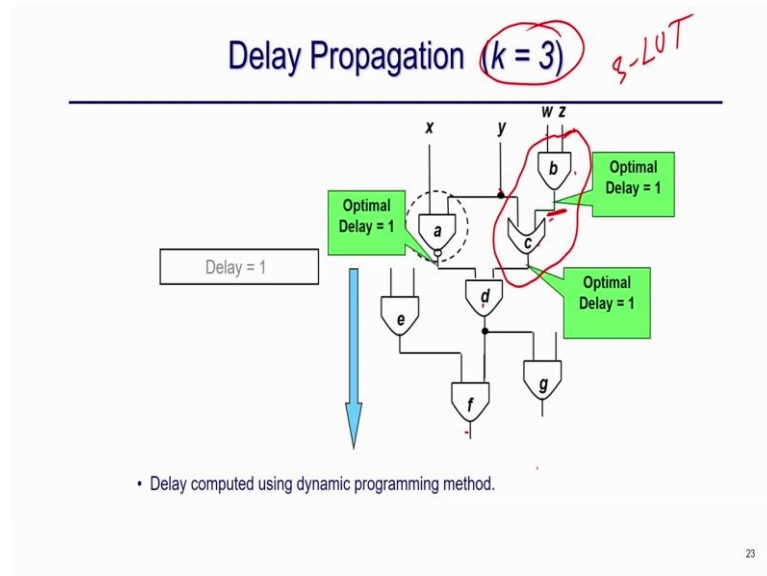
So, suppose I map this into this, so suppose my target is 3 input LUT. So that means, a LUT can have maximum 3 inputs, so suppose I map this into 1. So, at this point delay of this is 1, so the optimum delay at this point is 1 because this is 1 LUT delay.

(Refer Slide Time: 38:11)



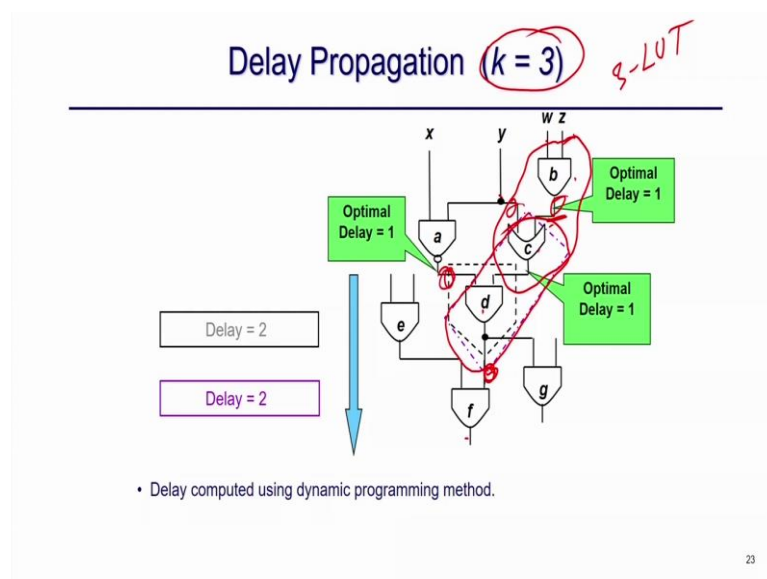
So, now suppose I map this also into another LUT. So, I map the whole structure into 1 LUT because now I have 3 input, 1 2 3 then at this point also delay is 1. Because though there are 2 gates, but at this point the whole thing is mapped to a single LUT. So, optimal delay at this point is 1.

(Refer Slide Time: 38:33)



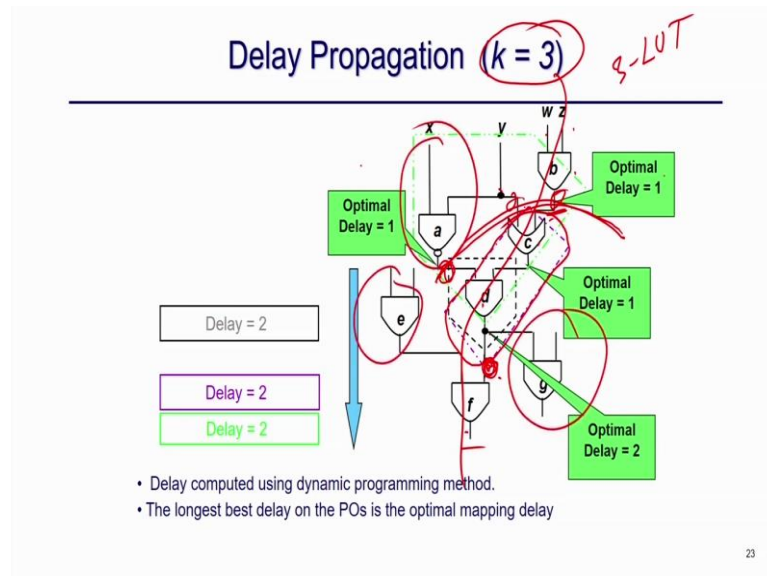
Similarly, if I just take this example. So, here also the delay is one because this is mapped to 1 LUT.

(Refer Slide Time: 38:38)



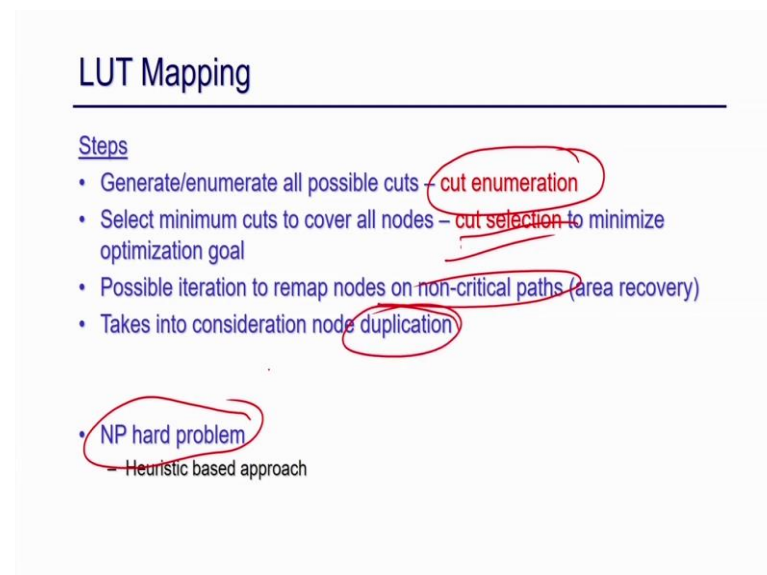
And now if I just take this, I decided to map this whole thing into 1, then how we calculate the delay here? I will just find out the delay of all the inputs maximum delay and I just add one more delay here. So, this is optimal delay at this point 1. So, this is not there at this point optimal delay is 1.

(Refer Slide Time: 39:04)



So, this should be optimal delay of 2. So, this is what is talked about, so at this point optimal delay would be 2. So, this is how I can actually identify the optimum delay of the whole circuit at the output, so what is the maximum possible? So, given a mapping what is that delay of that particular LUT mapping. So, this is we can actually identify and which is basically nothing but the maximum delay of all the inputs. So, the all the maximum possible delay of the all the input plus 1. This is how we can actually estimate the delay of your mapping.

(Refer Slide Time: 39:38)



So, now I am going to talk about this LUT mapping in detail. So, as I mentioned that LUT mapping is something we have to find out the cuts, cut means what? A cut of size k means there are k inputs 1 output. So, you have to first do all possible cuts in your

design, so cut enumeration. So, first step is something generate or enumerate all possible cuts that is called cut enumeration and the second step is that you try to cut select.

So, there may be lot of overlapping and lot of duplication cuts and all those things. So, I have to find out the minimum number of cuts which actually cover all your gates where we have minimum number of duplications. So, this is something called cut selection. So, there are two primary steps here cut enumeration and cut selection. And as I mentioned this problem is NP hard, we do not have an exact solution, that the solution is the minimum possible mapping of the gates into a k LUT.

So, we always have a heuristic base approach or which is basically kind of iterative. So, you do one mapping and then you try to do something, try to change little bit well you do another iteration try to do some modification in the mapping and try to get a better result. So, this is what is called iterative approach. So, what actually is happening here? So, if you have this non critical path, so if there is a critical path here.

So, suppose this is the critical path I cannot change anything here, maybe something can be done for this node, this node or some other nodes, so which is not in non-critical path. So, their mapping can be little bit of change, so that I can have a better mapping whether I can minimize more areas. So, this is something the overall idea that I have one mapping and then I iteratively try to improve the area over that particular mapping. So, one of the key factors is the duplication. How to remove the duplication as I mentioned earlier.

(Refer Slide Time: 41:39)

CUT Enumeration

Cut Enumeration

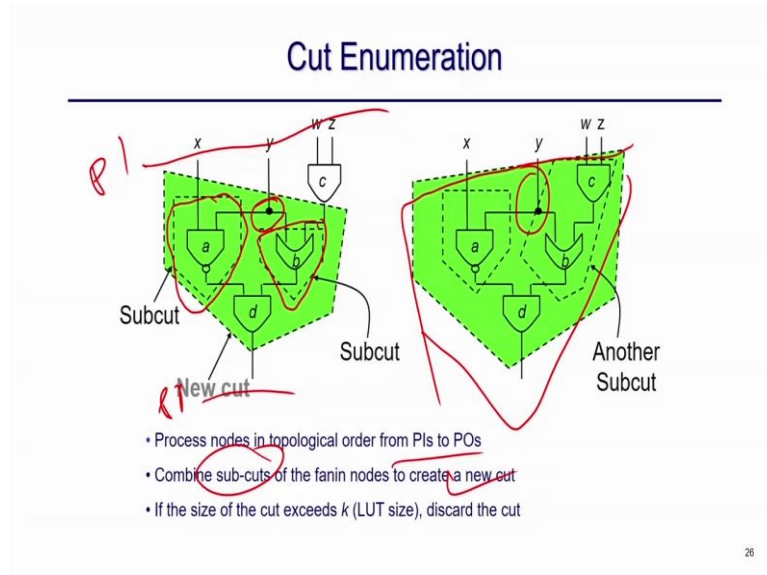
- Process nodes in topological order from PIs to POs
- Combine sub-cuts to create a new cut
- Search for re-convergence

25

So, let us go into this cut enumeration steps. In cut enumeration the basic steps are like this. So, we process the node in topological order from the primary inputs to the primary outputs. And then combine sub cut into create new cuts and also search for reconvergence. So, reconvergence example I have already given that even if your target is to map the things into k LUT, but you should not stop whenever you find a cut of size k because that may not be end of it.

So, you might grow a little bit further and then finally, you might find a reconverging path and that will give you a better mapping, so which I already gave an example.

(Refer Slide Time: 42:20)



So, I will talk about the other aspects. So, suppose this is my circuit. So, I will start from the input. So, this is my primary inputs and this is the primary output, so this is PI and this is PO. So, from where this I will going to start. So, as I mentioned topological order from the primary input to output and then I try to find a cut of my given size k cut. So, suppose I found a sub cut like this.

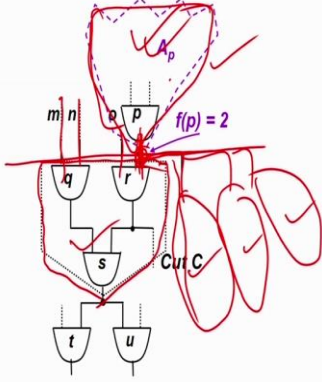
So, I found a sub cut like this where I have 2 input 1 output. Similarly, I found another sub cut of 2 input 1 output. And then I can actually combine these 2 cut because they have some input sharing and I tend to create a cut of size 3. So, there are 2, 2 input 2 is to 1 cut I can combine them to create a cut of size 3 is to 1. So, we try to combine the sub cut just to create new cut because our objective is to create all possible cuts of the design.

So, and this is one way of doing this. So, similar one example here, so this is 1 cut and this is 1 cut and since there is an once input sharing here, I can actually combine these 2 to get a bigger cut of size 4. So, this this is how the new kind of cut will be generated. So, I will start from the inputs and then try to find out some cuts and our maxim possible cut is k . So, I will find all the cut of size k or less than k and I can actually combine smaller cut where there is some input sharing to create a bigger cut as well. So, this is how we are going to generate all possible cut in your design.

(Refer Slide Time: 43:55)

Area Estimation

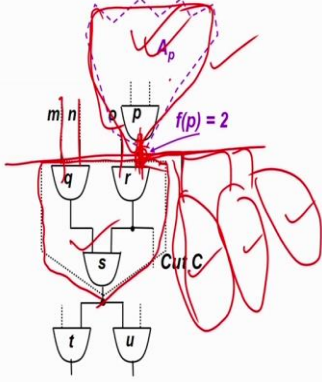
Tries to estimate area considering fanout effect



$A_c = \sum [A_i / f(i)] + U_c$

- A_i : estimated area of the fanin cone of signal i
- $f(i)$: fanout number of inputs
- U_c : area of the cut itself

- Can't underestimate area due to node duplication



27

So, now the important factor is that as I mentioned the number of cuts may be huge and there may be lot of overlapping, so which ones to select? So, there maybe you say 10000 possible cuts created and only 5, 10 or 20 has to be selected. So, which one has to be selected? So, that is something is important and for that we should have some cost factor, we have to decide this is a bad cut, this is a good cut based on some cost value.

If the cost is less; that means, it is a good cut if the cost is big; that means, this is a bad cut, so how to decide a cost of a cut. So, based on that we are going to choose the better cut, so that is given by this. So, there are 2 factors, costs contributed by the specific cut and the cost of the input that I am going to discuss. So, suppose I have this cut. So, cut means what? 1 output and k output, so cuts of size k mean there are k input 1 output.

So, suppose this is my cut. And so, I am going to talk about this in the next slide that, what is the component of the total cost of that cut contribution from this particular cut and this is from the input. So, what is this? What it says that the size of the input. So, this is say 4 input cut, so there is 1 input, 2 input, 3 input, 4 input. So, now, I am going to talk about the area of the input I. So, this is the area of the input I, because the input I will also map to some other LUT.

So, what is the area of that particular input and how many fanouts are there from this input. So, if this particular input is coming here and if this fanout have some multiple fanouts then this area has a contribution in this another cut here, another cut here, another cut here. So, this input will be going to map into another cut also, so that the effect of this area will be distributed among this 4.

So, that is how this comes. That area of the particular Ith input divided by the fanout value of that particular input. So, if the fanout is more this will be less if the fanout is less then this value will be high. So, this is something how to tackle the input and how to handle the fanout of the input also. So, this is how we are going to handle the area.

So, here one point to be noticed here is that, only the cost is not associated to one cut because as this is interdependent if we select one cut somewhere, we have to select some other cut from their input, so from the input of that cut. So, since the dependency between the cuts are there, so we have to take care of the cost of the cut itself along with the associated inputs and outputs and the surrounding cuts also. And what is the surrounding cuts? Those are the inputs.

So, from the inputs what is the area and their impact on that particular cut and if that input has a higher fanout, that means, has a less impact because that is a good selection. Because now I can use this cut output to multiple cuts, so this is something we can do, so this is how this first factor comes. So, then we will talk about this cost of that particular area of the cut itself.

(Refer Slide Time: 47:31)

Area Estimation

Tries to estimate area considering fanout effect

$$A_c = \sum_{i \in \text{input}(C)} [A_i / f(i)] + U_c$$

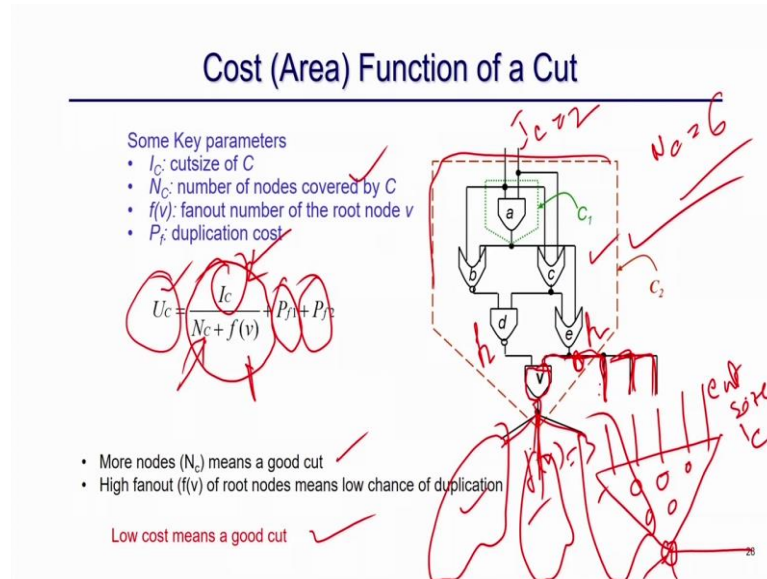
- A_i : estimated area of the fanin cone of signal i
- $f(i)$: fanout number of inputs
- U_c : area of the cut itself

• Can't underestimate area due to node duplication

27

So, that I am going to discuss first.

(Refer Slide Time: 47:36)



So, suppose this is my cut. So, key parameter is the cut size; a cut size of IC. So, that means, cut size is the number of inputs. So, if I just select a cut like this, this is the cut size. So, this is the cut size IC; this is the number of inputs.

And then NC is the number of nodes inside by the cut and then the $f(v)$ is the fanout number of the root node. So, for the output node, what is the fanout? So, there may be multiple fanouts here. So, this is the fanouts this is $f(v)$ and this is IC. So, for this cut I have 2 inputs, so this IC is equal to 2 here. And what is the NC for this cut? NC for this cut 1, 2, 3, 4, 5, 6. So, NC is 6 for this cut and $f(v)$ is 3 for this cut.

And then that $P(f)$ is the duplication cost I will talk about that. So, the cost contribution from this particular cut is given by this factor, this is the primary factor. So, its saying that IC by NC plus $f(v)$, what does it mean? So, if there are more nodes in the graph that means, it is a big good cut. So, if you pack more nodes in a particular cut that means, a good cut because you are trying to pack many or big number of gates into 1 cut.

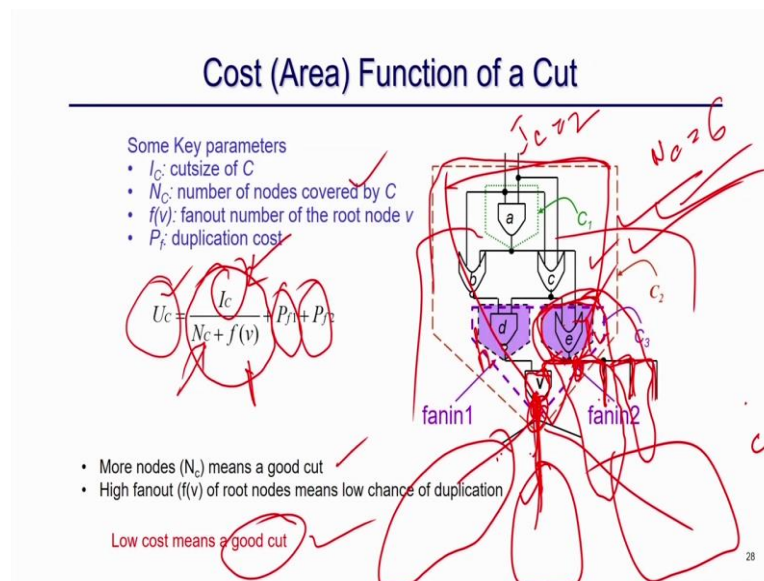
So, this is a good cut and as I mentioned the if the value of UC is low that means, it is a good cut. So, that is why this come into the output in that I mean here. Also, as I mentioned here if that there are more fanouts from this output, this cut will be shared among multiple cuts from here, because I will have another cut here, I will have a another cut here, I will have a another cut here. So, this cut will be shared among all this, so that means, it has a positive impact because this 1 cut I am going to reuse in multiple places.

So, that also reducing the cost factor. So, this IC is the number of cut size. So, the cost of that particular cut is given by IC by NC plus $f(v)$, you can understand if the number of nodes is big. Now $f(v)$ or the fanout from this node is big. So, this value will be less. So, there is something that will become a good cut and the other two factor is something I will talk about, now which is basically duplication cost. What is the duplication cost?

So, you can see here suppose this is my f node, which is the root node. So, root node is the last node and if the input of this is f1 and this is f2. So, the input of that root node. And from here if there is a fanout, the problem here is that if there is a fanout there is no output from this cut because your LUT has 1 output and k input. So, since this is my output, this output is not available, once you map this cut into a single LUT this output is not available, so this is very important.

But this is going to be used somewhere else. So, the problem here is that, this node has to be duplicated in some other cut. So, for this whatever the cut I am going to generate I will just draw it. So, you will understand. Let remove this.

(Refer Slide Time: 51:13)



So, the problem here as I mentioned the only output is available is this. So, this is not an output available to us. So, now, but this output is use going to use some other nodes. So, whenever I am going to decide a cut here this node has to be duplicated for this cut because this node is not available for me. Similarly, I am going for this also I am going to have this node duplicated, so this is the problem.

So, if that particular root node, the parent node, the previous node of the root node has multiple fanout that is actually a bad thing, because it has multiple fanout that has to duplicate in different other cuts also which is actually unnecessary. So, that has a negative impact.

(Refer Slide Time: 51:57)

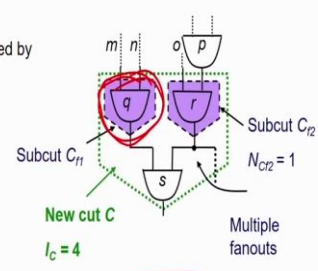
Duplication Cost Adjustment

- Considers potential node duplications
- Check the sub-cuts for multiple fanouts
- Area adjusted by addition of duplication cost

Duplication Cost:

- N_{C_i} : number of nodes contained by subcut C_i
- I_c : cutsize of C
- f_i : fanout number of subcut

$$P_f = \begin{cases} \frac{N_{C_i}}{I_c} & \text{if } f(i) > 1 \\ 0 & \text{otherwise} \end{cases}$$



- Larger N_{C_i} is, larger possibility of duplications. Means larger duplication cost
- I_c is the normalizing factor.

29

So, this P_f is given by this. So, the f_1 and f_2 I talked about. So, this in general is f . So, the number of nodes in that particular cut at the input and this I_c is the size. So, the point here is that if the number of nodes here is high then the possibility of duplication is also high, that means, this has a larger duplication cost and this I_c is the normalization factor.

So, as I mentioned here if that particular in the cut. So, they so we always consider sub cut here. So, if the in the sub cut of this input f_1 has a greater number of nodes that there is a high chance that some of them have a fanouts and that will create a duplication. So, that means, it will increase the cost of that particular cuts that means, is a bad cut. The basic idea is that if you select a cut and if the internal node has some fanout that is basically a redundant thing.

So, that node has to be duplicated for some other cut. So that means, you should not choose this kind of cut and try to avoid choosing this kind of cut because this is something there are lot of nodes will be duplicated in some other places. So, that will increase the cost and as I mentioned the low cost is good. So, if there is no such fanout this will be 0. So, there is no impact of that. So, the cost will be less, but if there are lot of nodes then the cost will be high and there is a high chance of duplication.

So, we try to increase this cost of this particular cut and try to mention that this may not be a good cut. So, this is how we are going to handle this duplication. So, this is something we talked about. So, there are 3 aspects here you know the first thing we will start from this, we start from this primary input to output and we choose the sub cut we combine the sub cut to generate a new cut as I mentioned here. And then also we try to remove the duplication, avoid the duplication this way just to increase the cost.

And also, as I told the reconvergence also has to be taken care, that example I already have given. So, these are all the things we have done iteratively just to find out all possible cut. So, that I get all possible cut of your design. So, once you have generated this, the next step as I mentioned is the cost which something is given by this factor

which has impact of that own factor the size from the cut size and that by further the fanout of the internal nodes that will come into picture.

As well as the factor is from the input also if there are some inputs which has multiple fanouts, which is good. So, as I mentioned again that if the output of a cone has multiple fanout which is a good thing. Because this this is a good cut because this result is going to use somewhere else because none of the node will be duplicated here. So, since this is 1 output this will be 1 cut, this will be 1 cut, this will be 1 cut. So, this is going to be a good cut.

So, in a good cut if the fanout of that root node is high that means, is a good cut, but if there are some internal or have lot of fanouts which is bad thing because those nodes effectively will be copied to somewhere else. And which will result in lot of duplications and most likely have number of modes LUTs, so this is basically a bad cut. So, once you have done this we have to select, so we are going to select the cut.

(Refer Slide Time: 55:22)

Cut Selection

- Once cuts are generated, traverse networks from POs to PIs and select cuts that map into LUTs
- Select cuts such that timing is met and the area is minimized
- Node in critical path have to be selected.
- Choices lies on non-critical path.
 - Nodes in non-critical paths have the luxury to select different cuts.
- Iterative Cut Selection Procedure
 - Local Cost Adjustment
 - Input Sharing
 - Slack Distribution
 - Cut Probing

30

And the cut selection as I mentioned, we again have some heuristic algorithm which is happening. So, our objective is, we will try to cover all the gates of your design using minimum number of cuts or maybe you try minimize the area or minimize the timing So, that is something has to be there and the choice here is that if there is a critical path, I just show how to calculate the delay once you have some choice you just select some cuts you can find out the delay.

So, whatever there is in the critical path that cannot be change, but if something lies in the non-critical path there, I can actually make some choices, some alternate, so that I can improve the area. So, usually it happens in iterative process. So, I create one solution then try to improve the solution in every iteration and there are some other factors also which is important, as I mentioned that these cuts are interdependent. So, once you select any cut, I associate some cost with it.

So, using the logic I just defined previously. So, every cut has a cost, and once you select a cut and that actually makes some changes in the surrounding cuts because maybe some fanout is now going to be considered and some will be ignored. So, they have some impact on the surrounding cuts. So, that is something called local cost adjustment. So, once you select a cut, we enumerate all the cuts and I associate a cost of that cut using this function.

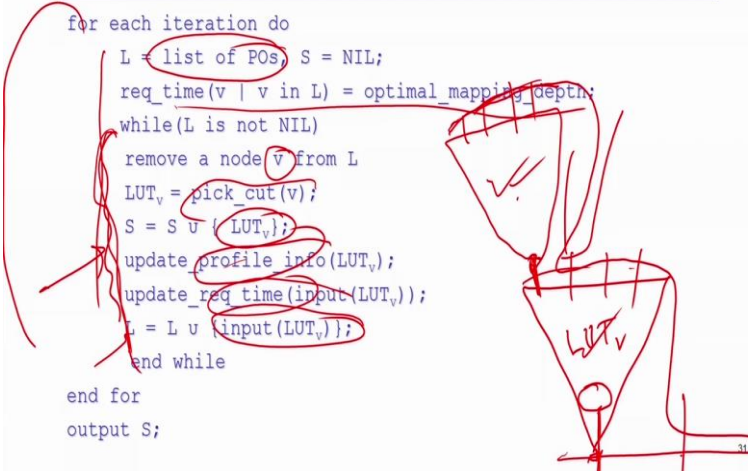
So, every cut has a cost, but once we select a cut, because I generate all the cuts. Now, I am going to, select all the cuts I am going to select some of them. So, once I going to select some cut then it has some impact in the surrounding cuts, as I mentioned the actual cost also involve the input fanout of the internal node, fanout of the output node. So, all are actually making effect on the actual cost.

So, once I select some node it may have some impact on the surrounding nodes. So, that has to be adjusted. So, that local cost adjustment is that. So, for that I have to consider the input sharing, slack distribution and cut probing that I am going to discuss. Those have an impact on the cost and I am going to update the cost of the surrounding cut using this logic.

(Refer Slide Time: 57:57)

Iterative Cut Selection Algorithm

```
for each iteration do
  L = list of POs; S = NIL;
  req_time(v | v in L) = optimal_mapping_depth;
  while(L is not NIL)
    remove a node v from L
    LUTv = pick_cut(v);
    S = S ∪ {LUTv};
    update_profile_info(LUTv);
    update_req_time(input(LUTv));
    L = L ∪ {input(LUTv)};
  end while
end for
output S;
```



So, here is the overall algorithm I am going to going to briefly explain. This is one iteration. So, you can see within the iteration this is one loop in which I am going to choose one solution or I am going to get a solution. And in the outer loop I am going to generate multiple solution and I am going to improve the solution. As I mentioned the start is starting from the POs, the primary outputs the cut selection start from the primary output.

So, my L is that list of POs primary output of your circuit and then I just find out the required time using some analysis that what is the optimal timing depth and all those things using that logic. And then what I am going to write? I take one output and then I pick a cut because from that cut. So, suppose this is my circuit I took an output node. So,

this is my output, so from this in the enumerated list there may be many cuts maybe 10, 20, 50 cuts possible which has the output as this.

So, from that I am going to choose 1 cut which has the least cost, which is a good cut. So, once I choose that, so this is one LUT I have already decided. So, suppose I choose this cut, so suppose I have 2 input output initially, so my PO has only 2, this and this. So, I decided to choose this and it has say four output what will happen now. So, my list will include this input of this LUT v , so this is my LUT v .

So, I am going to choose this LUT v because this is a good cut starting from this. So, once I have done that now I have to choose some cut from this input. So, now, my L will consist this. So, initially I have 2 these two once I have selected this cut all the input will also become the primary output now because this is already decided now I have to select some cut starting from this node of the cut where the output of this. So, this is how the whole thing works.


So, whenever I am going to choose this one, my cut set will become this followed by this and this. This is how the whole thing works. So, every iteration I am going to select one cut and I am going to update that PO list that is the logic I define. And also I have to also do this update profile as well as this update required time as I mentioned using this local cost adjustment because some of the other cut will improve with this selection.

So, this I am going to discuss, so this is what is the one iteration. So, as I mentioned once you select a cut, we have to update the surrounding cuts update this timing profile info using this local cost factor that I am going to discuss now. And so, after this while loop I have one solution. So, this is 1 LUT this one may be selected and so on. And once we have this then in the next iteration when I am going to talk about this timing information and all I am going to talk about the timing information of the previous selection. So, that I can decide whether my current iteration is better or not. This is how the whole algorithm works.

(Refer Slide Time: 61:13)

Input Sharing ✓

Local Cost Factor
Input sharing
Select C_a that share input with some of the already selected cuts
- High chance of reduction in node duplication



C_a and C_b share C_w

Now, if you just think about that local cost factor one of there is input sharing which is very important and I mentioned earlier also. So, suppose I have decided this is one cut and this is one cut. So, this is one cut and this cut is shared in many places. Now, once I start from this, so just to elaborate this example suppose I have this, so I have decided this cut.

And then say suppose from this input I have selected this cut as well. So, this is something I never know which order it will go. Suppose I have selected this one. So, now, if I start from this node and if I saw a cut where one of the inputs is coming from another cut which is already selected. So, that CW, then this cut is a better cut because this particular input is shared among multiple cut which is already decided.

So, there is a high chance that the duplication will be less here. So, in this case I am going to reduce the cost of this cut further. Because I am now in this particular cut where I already selected this 2 cut and I found for this cut, one of the inputs is coming from already a cut which is selected. That means, if I select this cut there is a high chance there is no duplication through this. So, then this is something is a good cut.

So, I am not going to improve the cost of this particular cut further. So, that the chance of getting selected of this cut is high, so this is what is called input sharing. So, if this input is coming from already selected cut which should be a good cut. And as I mentioned also that if cut is selected and it has multiple fanout that means, it is a good cut.

So, it has a positive impact in LUT mapping. So, this is what is called input sharing. So, if we found something then I am going to reduce the cost of the particular cut further. So that the chances of getting selected of this cut getting high. This is called input sharing.

(Refer Slide Time: 63:10)

Local Cost Adjustment – Slack Distribution

- $Slack_C = Req_C - 1 - \text{MAX}_{i \in \text{input}(C)} (Arr_i)$
- If $Slack_C < 0$, C is not a timing feasible cut
- The larger the $Slack_C$, the better for C in terms of slack distribution effect

$$C_{si} = C_i - \epsilon \cdot Slack_C$$

33

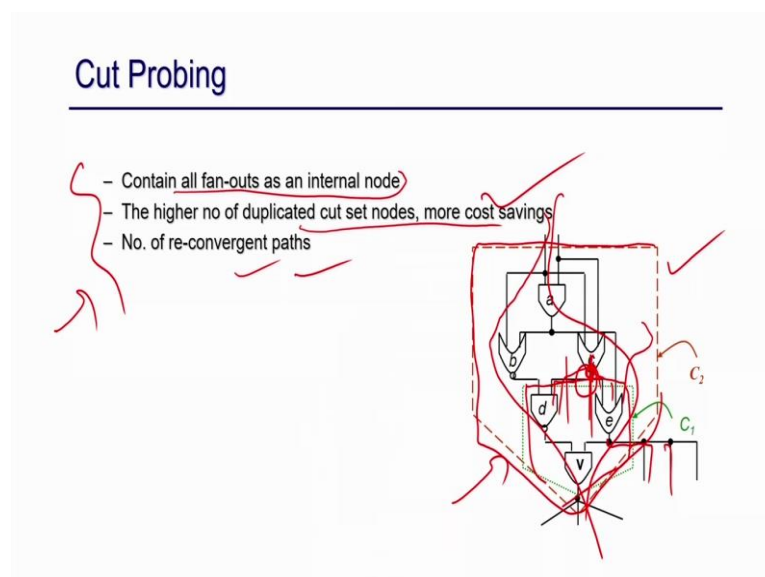
And then the slack distribution, so I know from this timing information that I need the required timing here, the data should come here by this time. And now if we find that the arrival time of the input, the max of arrival time is that, so this is the largest among these.

So, now the slack is what? The required time minus this. So, if I found this at this point if this is 0 or less than 0. That means, this is not a possible factor because you the data is coming later than what is expected. So, this cut should not be selected because this is not a feasible cut because if you select this way, the data will come here later than the expected required time.

But if it is positive that means, it is actually a good cut because I have some additional time here which I can utilize to have a different selection, so that I can reduce the number of areas. So, I am going to further reduce the cost of that particular cut if it has a positive slack by a constant factor. So, this is something is the uh the local aspect.

So, if I just find that local the timing of this particular cut is positive that means, if you are going to select, I have some extra time to select something else. So, I am going to increase the chance of getting it selected by reducing the cost of that particular cut further. This is what is called slack distribution using local cost during local cut adjustment.

(Refer Slide Time: 64:46)



This cut probing is also an important factor. I mean if I just found that a particular node as I mentioned if there are multiple fanout which is bad thing which is already taken care of that, but if I found some node where all the fanouts are part of the same cut. So, this is one cut and from this fanout both are coming here or you can think about this is my whole cut I have selected and where the internal load has fanout which is a bad thing, but fortunately all the fanouts are a part of the same cut.

So, which is a good thing so that means, it is not going to increase the size, it will reduce the duplication and also positive thing is this is a kind of a reconvergence.

So, it will improve the number of nodes getting selected inside that. So, that is the point that if all the fanouts of internal is part of a cut which is a good thing. So, it I am going to reduce the cost of that particular cut further, so that this make it better cut.

Also, the other example that I just gave here for this cut if I found the multiple input coming from the single source which is also a good. So, I am not going to select all of them are coming here. So, the high chance is that this the number of inputs of this is less because I know I can consider both of them as a single input and I can select something else as well. So, this is also most cost saving.

So, this also have a positive impact on the cut similarly the number of reconvergence path. So, if I found something which is going further and then slow down similarly there are two reconvergence paths here. So, if the number of reconvergence convergent path is high that means, the circuit has grown and then reduced, so which is a good thing. So, that means, the possible of number gates that is getting mapped into this particular cut is high.

So, again all these three aspects are actually improving the chance of getting selected of this. So, I am going to reduce the cut cost of that particular cut further using all this scenario. So, this is something I am going to do it locally and now this also decides about the current scenario. Because once I have selected something, in the next iteration I am going to update this because I know now the surrounding things what is happening there.

So, these are the three factors, this cut probing, this slack distribution and the input sharing, through which I am going to improve the cost of the cut locally and during the iteration and this is how that this is how the whole selection algorithm works. So, this is how the whole thing works that I decide start from the output I am going to select a base cut.

But when I am going to select this peak cut, I am going to update the all I consider all this local factor like input sharing slack distribution and this cut probing to update the cost and I am going to select the best cut. Once I have selected it, their input will become the PO now because this is selected their output will become PO. And then once this is selected their output or input will become the next starting point. So, this is how the whole algorithm works and finally, I am going to select cut which will cover all the gates of your circuit.

(Refer Slide Time: 68:13)

Good choice

- A node with high fanouts as root of a cut ✓

Bad choice

- A node with high fanouts as internal node as a cut
- High chance as duplication

Goal

- Reduce duplication as much as possible

So, this is what this LUT mapping all about and as I mentioned that if a node has high fanout as a root of a cut which is a good choice, a node with high fanout at internal node is a bad choice. Because high chance of duplication is there and our objective is trying to reduce the duplication as much as possible and try to map the whole thing in minimum number of LUTs and minimum number of delays.

(Refer Slide Time: 68:36)

Summary

- FPGA technology mapping involves mapping
 - Gate level design → LUTs
 - Multiplier/MAC → DSP
 - Memory/Large Registers → RAM/ROM
- DSP and RAM inference happens in pre-map stage, i.e., before logic synthesis
- LUT mapping involves mapping as much as possible log into one LUTS.
 - CUT enumeration
 - CUT selection
- Several factors
 - Re-convergent paths
 - Avoid duplication
 - Input sharing

36

So, here is the summary that FPGA technology mapping is little bit different from ASIC mapping because here we have dedicated resources like LUTs, DSPs and RAM and our objective is to map the whole thing in this particular blocks. And this DSP and RAM inferences usually happen during pre-mapping stage, so that because we have to preserve

those structure. So, that does not get flattened or may resolved into the gates, so that DSP or RAM inference cannot happen. And LUT mapping is a hard problem.

So, we usually happen we apply the heuristic base approach and there are two steps primarily here one is cut enumeration second is cut selection. And we also apply several kinds of other factors cost factor like reconvergence of path, avoid duplication, input sharing, cut probing and those kinds of techniques just to readjust the cost and try to select the best cut. So, this is the overall summary of this technology mapping of FPGA.

Thank you.