

## **Optimization Techniques for Digital VLSI Design**

**Dr. Chandan Kafra**

**Dr. Santosh Biswas**

**Department of Computer Science & Engineering**

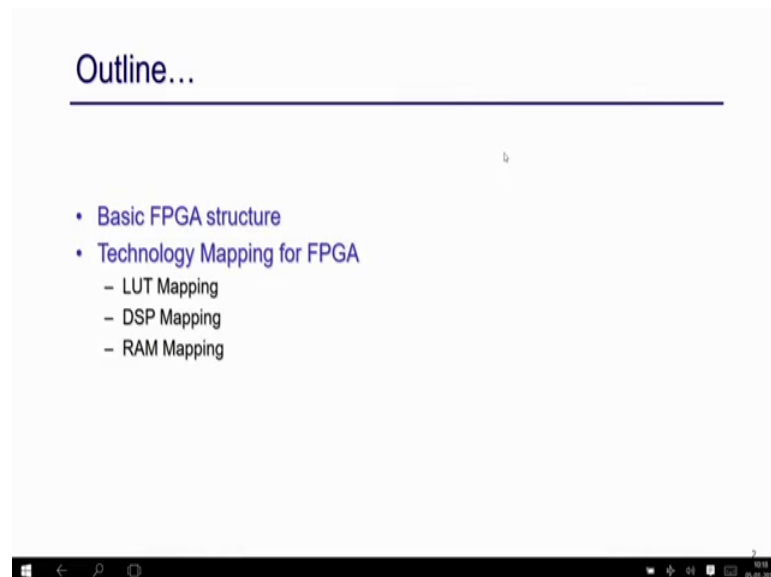
**Indian Institute of Technology, Guwahati**

### **Lecture – 11**

### **Technology Mapping for FPGA**

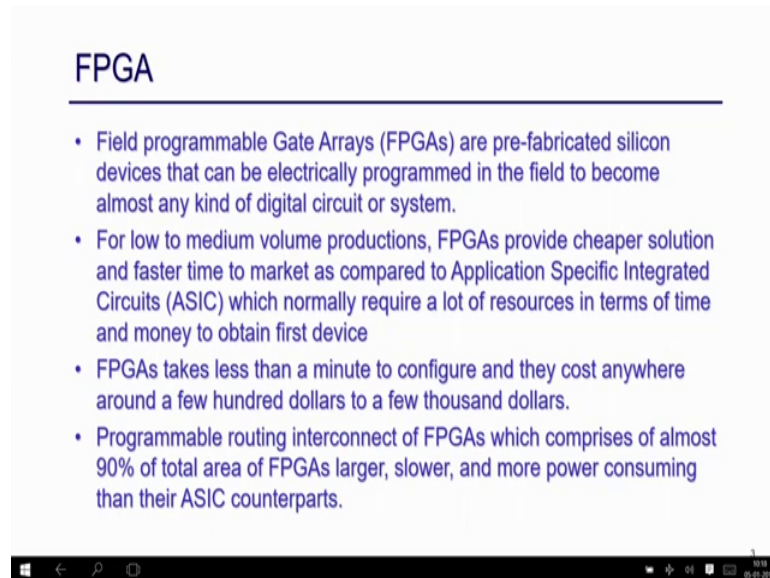
Welcome everyone. So, today, we are going to discuss about this technology mapping called F P G A.

(Refer Slide Time: 00:38)



Specifically, we are going to talk about the basic structure, what are the components of F P G A s and then will talk about the technology mapping called for F P G A specifically L U T mapping and D S P mapping and RAM mapping ok.

(Refer Slide Time: 00:48)



## 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.

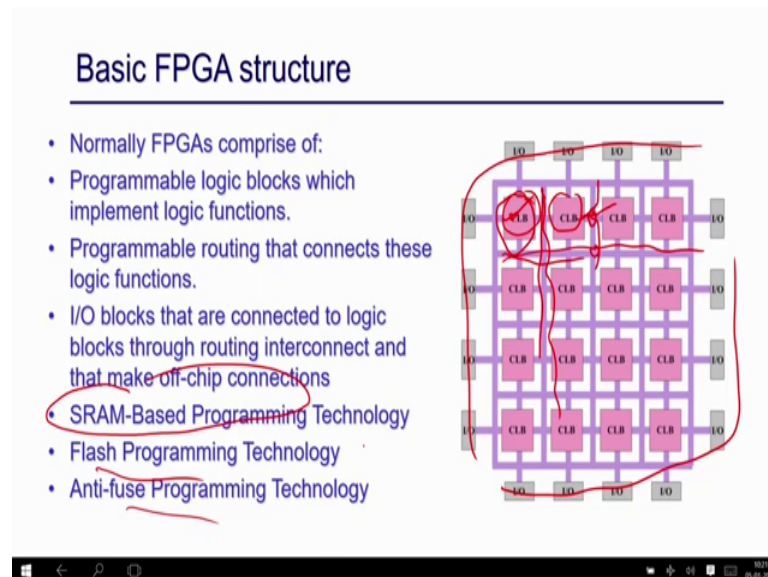
So, we look into the F P G A field [programming/programmable] programmable gate array, which is basically a pre-fabricated silicon right. So, it is already fabricated, but advantage is that it is electrically programmed in the field and it can almost implement any kind of digital circuit right. So, it is already have a fixed structure component, all are already, fabricated in the, fabricated in the board and it can actually map any kind of logic circuit into that.

So, advantage of this, F P G A is that it is very fast right. So, the time whenever you have R T L, you can easily just map into this F P G A using some synthesis tool, provided by the industrialized (Refer Time: 01:31) or synopsis schedules and you just, use right. So, you do not have to. On the other hand, in the ASIC flow, you have the whole process, is very time consuming and it does not make sense to have a, when you have, the number of chip, you require is very less right, if you just need one or two kind of thing, kind of chip or design, you do not go for ASIC, because it is time consuming process and it take, it is costly also right.

So,. So, that is why F P G A (Refer time: 02:00) 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 actual chip, we just map it to F P G A and see, whether the simulation is giving correctly or not, whether the, the your design is working correctly or not right. So, F P G A is very popular, the disadvantage of F P G A is that it is little bit slow, because it is a

programmable interconnections. 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, much more, much more smaller than A F P G A design, but because of the other advantages, F P G A is widely used in India industries ok.

(Refer Slide Time: 02:48)



So, if you just look into the F P G A in details. So, it consists of. So, this is the kind of overall structure of F P G A, it consistence of logic blocks configurable logic blocks. They are just arranged in arrays and they are interconnected through these interconnection units, right. So, these are the interconnections units and the I O S are placed usually in the outer interface right.

(Refer Slide Time: 03:10)

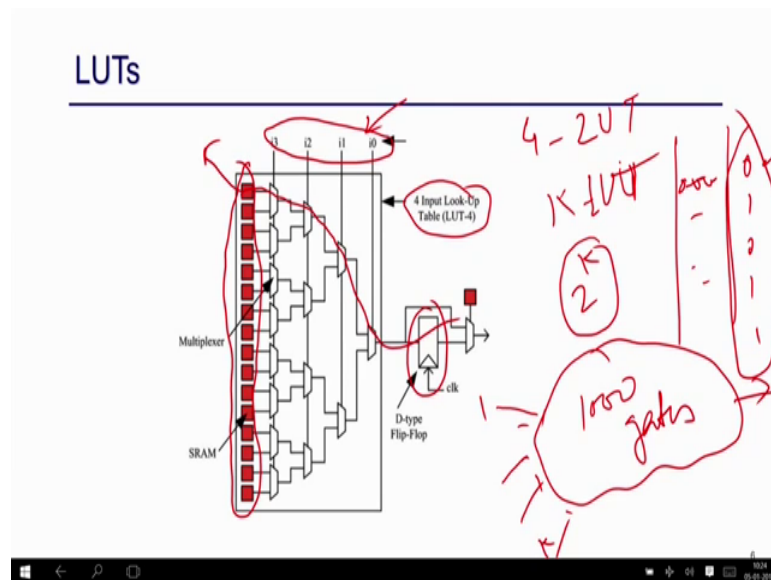
**CLBs**

- 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

And this configurable logic block is specifically is configurable. So, I where you can actually map yours 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 right. So, any, anywhere the connection can be made through this interconnections ok. And primarily, this programmable component are mapped through s RAM based, programming technology also in some scenario, it is, can be flash based programming technology or anti fuse programming technology ok.

So, if you, look into this C L Bs configurable logic blocks, it consists of, this is the primary component, where your design has to be mapped right. So, it consists of primarily a cluster of basic logic units, which is nothing, but look up table ok. So, and also it consist of some memory also, just to, for storage functionality like flip fops and all RAM and all other components right. So, essentially this L U T look up, table is the building block of an F P G A, because this C L B consist of several 4 to 8 L U Ts or may be more in modern F P G as ok.

(Refer Slide Time: 04:28)



So, if you just look into this L U Ts. So, L U Ts is basically look up table right. So, it consists of a memory, which is s RAM based 1 bit memory right and it has some inputs and it decides, which memory bit should go at the output right and it also have a register at the output. So, based on the number of such inputs select line, we say it is a four input look up table right. So, four L U T, basically, K L U T depends on the number of inputs right. So, what does it mean? So, a L U T, if it has a K input, if K input L U T can implement any Boolean functions of 2 to the power K right, so possible for possibilities of bits are there. So, any Boolean functions of, K input can be implemented here right.

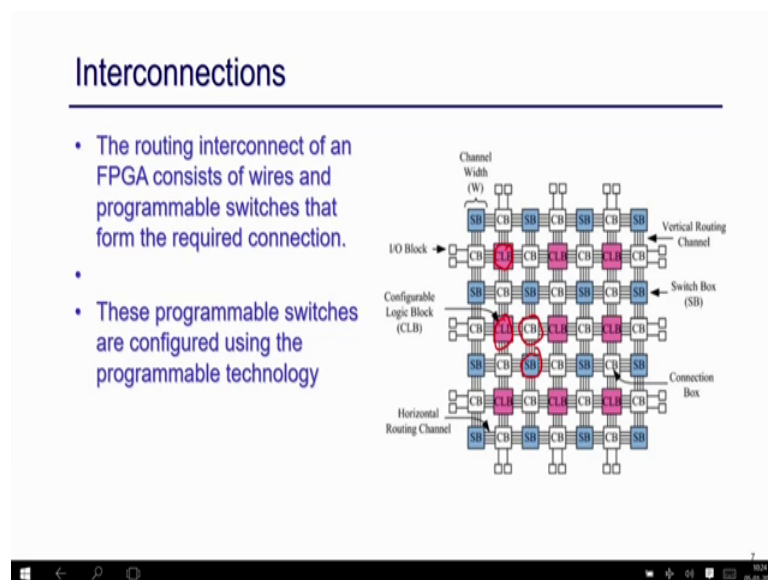
So, if you just think about a Boolean circuit, it consists of, may be about 100-200 gates, it has a K inputs 1 to K and 1 output. So, that Boolean circuit, it consists may be, consists 1000 gates, it does not matter how many gates are there. So, it is, you have to just capture the input, output correlation right. 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 did that, table for all possible input possibilities are there and corresponding that output right. So, that is the output bit and this output will be stored in this s RAM right and, when your input is say 0 0 0, all 0es then the output is 0. So, this bit will be coming at the output right. 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 possible of all possibilities of output will capture and we store that data [vocalize noise] will store in that s RAM, the memory bits and the select line, the inputs will actually select, which bit has to with the output, which output should cover the output right.

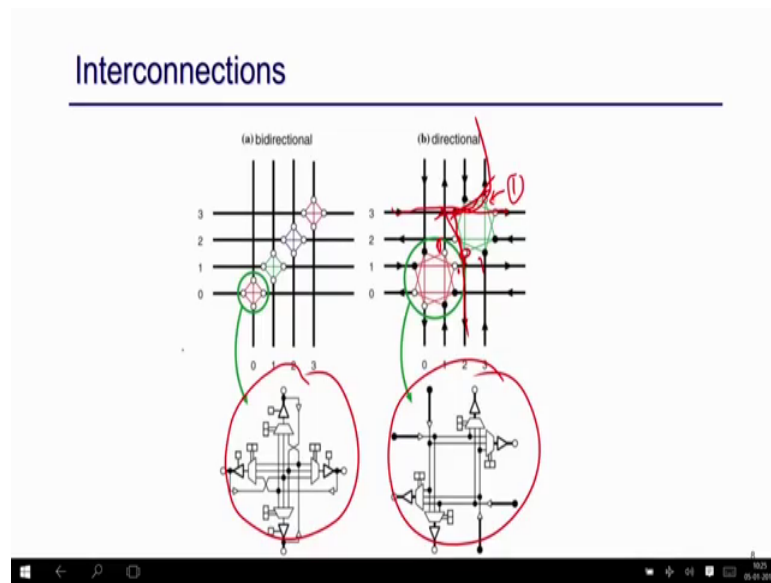
So, that is something is look up table and. So, primary components of this C L Bs are look up table and also, if you just look into the interconnections, I also said [FL], this interconnections are also programmable, if you, look closely on that interconnections.

(Refer Slide Time: 06:51)



So, these are the C L Bs right and the interconnections, we have switch box as well as circuit box right.

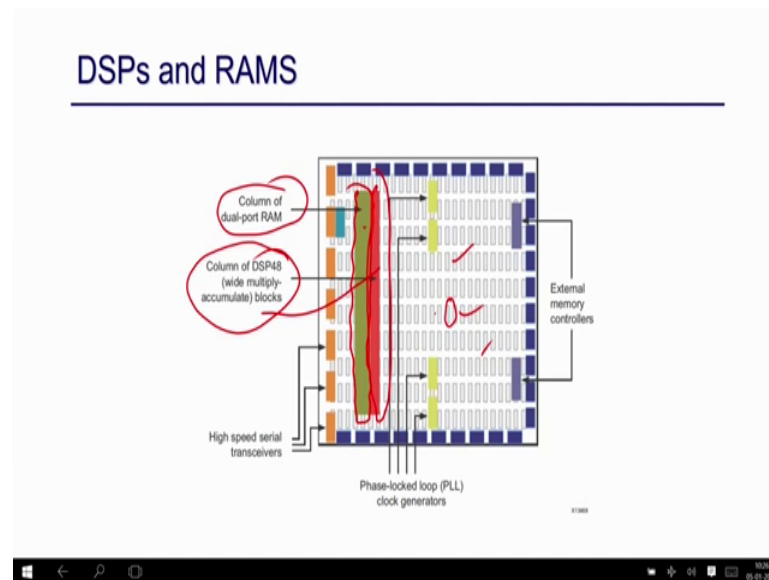
(Refer Slide Time: 07:00)



So, the switch box, if you look into this they are interconnections, either bidirectional or normal the all possible directions. So, so we can understand here. So, whenever some data come, it can go this way, this way, that way and all the possible ways right. So, this, these are the contraval. So, if you just said this bit is 1 then this data will go this way, if you said this bit is 1, then this data will go this way and so on. So, if you just said this 1 1 and this 2 0 then this input will come and go through this way right.

So, this is, this is how the whole circuit is implemented, like this bidirectional and unidirectional. So, this interconnections are, has to be programmed also right, based on that actual interconnections, we are going to set this bits 1 or 0 to make the actual interconnections right. So, this is how the overall, structure of this, of this, F P G A ok.

(Refer Slide Time: 07:52)

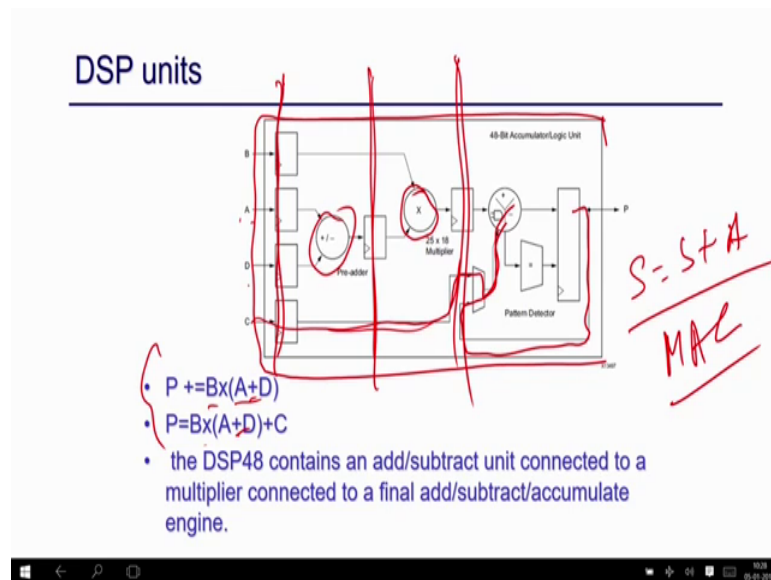


So, if this, if you will look into an F P G A, it does not only consist of L U Ts. So, this C L Bs consists of several L U Ts and also some of the C L Bs are D S P units right. If you just look into this 1, specific structure here, you can see that a column specifically used for dual port RAM.

There is a another column, which is specifically used for D S P right. D S P 48 so; that means, in this overall structure, as I said, these are all C L Bs. Some of them will be used specifically for D S P, some of them used for RAM and some of the other is for logic units, where is basically, we have this logic, this look up table based structure are stored.

And in a C L B I, we do not have only 1, we may have say 10 12 14 number of, logic units are stored together and they are interconnected right and also, and also one register is where, just to, you can use that register also to store some data also right. So, this is the overall structure of a D S P.

(Refer Slide Time: 08:54)



If you just look into the sorry, connection of F P G A, if you go into the D S P structure. So, D S P is basically, the 2 is a very first implementation of a multiplier. So, if just look into look closely in the D S P structure, if you can see here, in the D S P block, there is a multiplier right. So, the primary purpose of this, D S P unit is to perform the multiplication, in very faster right, if you just map a multiply into this, look up 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 fabricated within the, F P G A board itself, but in the D S P units, it is not only a multiplier, it can have a pre adder, it has also have set of latencies right. So, we have 3 set of latencies here and this is MAC, there is a, I mean sum right S equal to S plus a accumulator right. So, this what is call accumulator. So, you can multiply and accumulate right. So, multiply and accumulate MAC right. So, this is what is happening here, also we have a, we can also do a, post addition also right. So, this can also come here and you can do a post addition.

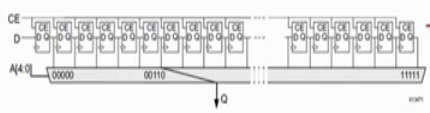
So, you can do specific type of operation like you can do B into A or you can do A plus D into B or you can do A plus D plus minus, this is plus minus and then you multiply this with B and then you add C right. So, all kind of possibilities are there. So, all these kind of structure, when you have in your design, those can be mapped to your D S P unit and also you can pack some of the set up registers. So, three layers of registers are there. So, it has three latency, either you can bypass or you can use them. So, this is the abstract

structure in, in, in design, you can actually bypass any, any set of registers also. So, this is what this D S Ps and also it has some memories RAM.

(Refer Slide Time: 10:40)

### BRAM and other Memory units

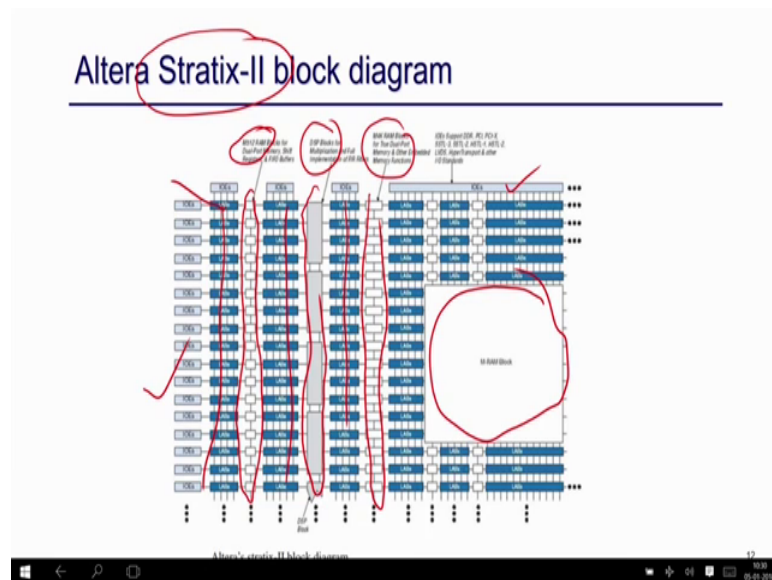
- The FPGA fabric includes embedded memory elements that can be used as random-access memory (RAM), read-only memory (ROM), or shift registers. These elements are block RAMs (BRAMs), LUTs, and shift registers.
- The shift register is a chain of registers connected to each other. The purpose of this structure 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 address 'A[4:0]' is shown with values 00000, 00110, and 11111. The output 'Q' is shown with a value of 11111. A red checkmark is next to the output 'Q'.

So, here you can store the large arrays large memories, large arrays RAM and ROM also, it has some shift registers ok. So, shift register in hardware, you know there is a specific kind of register, where you have, have that data in every cycle, you shift the data you want be, by left or right, left shift or right shift right. So, in specifically, in filter application, the shift register is wide use. So, some of the, registers I mean some of the memories also, shift register inside the D S Ps inside the F P Gs. So, this is the overall, configuration of a of a, F P G A.

(Refer Slide Time: 11:16)



And now, if just look as specific case an example, altera stratix 2 board, you see, you can see here the, this lab is nothing, but the C L Bs. So, the in altera or this lab is called as lab sorry C L Bs are called as lab, lab logic array block. So, which is C L Bs. So, inside that. So, you have series of labs and these are the and some of them are actually smaller RAM right. So, these are the smaller RAM, you can see this is a M procedure of M figure, 5 on 2 bits RAM. This is also RAM sorry, this is D S Ps, this is D S P block, these are again lab and again, there are smaller RAM M for 4 K 4 K bits RAMs are there and there is a bigger RAM also right, you can see and the I O S are here. So, this is, this are also I O S.

(Refer Slide Time: 12:08)

### Altera Stratix-II C L B

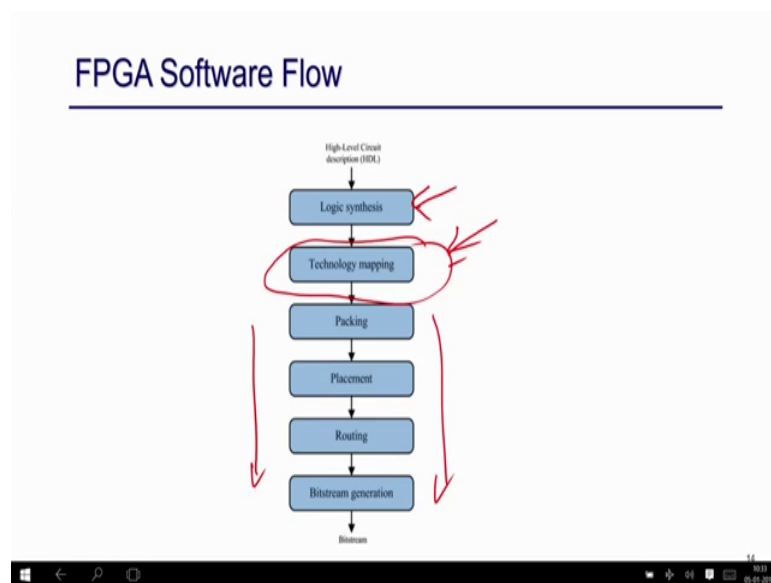
- Each Stratix II LAB consists of eight Adaptive Logic Modules (ALMs).
- An ALM consists of 2 adaptive LUTs (ALUTs) with eight inputs altogether. Construction of an ALM allows implementation of 2 separate 4-input Boolean functions.
- In addition to lookup tables, an ALM provides 2 programmable registers, 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.

So, this is a specific structure for stratix 2 block and if you just have some detail data like this, this lab the, which is basically C L B C L B consists of 8 AL, A L M right and each A L M consists of 2 L U Ts so; that means, effectively in a inside a, we have 16 kind of L U Ts ok, also we have other, other input like the two dedicated full adder is also implemented, there a carry chain and a register chain right. So, what is the importance of this carry chain, what is the utilization of this? Whenever you have a adder circuit. So, it have a carry right and if the carry has to, has to route through this, this circuit block to somewhere else, it will slow right, you want to, execute that in fastly right.

So, what it actually happen inside the lab, whatever the 16, 16 as HM, L U Ts are there, they are connected right. So, their output are connected each other, locally not through the other circuit block. So, this is carry chain. So, whenever you implement a, adder inside a C L B that carry does not will not be routed through the switch box or circuit box right. It will be internally connected and it will be very fast connection for.

So, that actually you can do, this, this additions, fast right. So, this is how, this adder is also can be implemented in the L U T, in faster manner ok. So, this is something, the detail about the stratix block.

(Refer Slide Time: 13:30)



So, this is all I just talked about just to give a brief idea, what is F P G A and what are different components are there inside the F P G A, because when you are going to talk

about the technology mapping. We need to know, what are the components are there, which can be used efficiently, for your design purpose right ok.

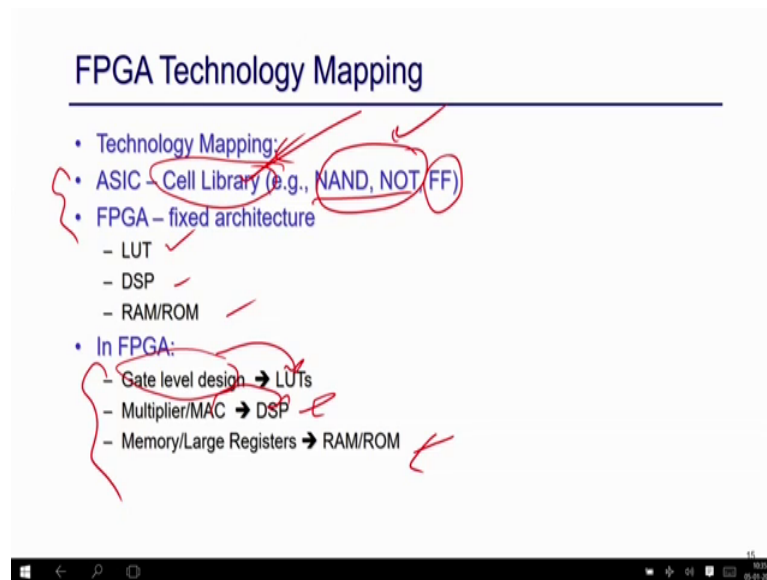
Now, if you just look into our software flow right. So, E D A software flow, we know this high level synthesis, then logic synthesis and physical synthesis right. So, after logic synthesis, you know in the physical synthesis, we have this flow planning placement routing. So, and so, in F P G A, whenever the after, the logic synthesis is done. Now, you have to map your circuit into this, this L U Ts in D S Ps and this RAMS right.

Because you do not have any gates in, in the F P G A 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 K input L U T right. So, four input L U T. So, you have to figure out all possible, such four, four is to one kind of component or the pattern in your design and that has to be mapped into that, in a single, single L U Ts right.

So, you cannot map all the gates individual 1 L U T that will be more costly. So, you have to find a pattern and we have to map them into the L U Ts. We have to find the multiplier, we have to map them into D S Ps, we have to find the big arrays, we have to map 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 HM in the actual hardware, you have to place them into specific L U Ts, because 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 F P G A and that is, that discussion topic for today. So, we are going to discuss more on how to map a big circuit into F P G A into L U Ts into D S Ps into RAM efficiently right.

(Refer Slide Time: 15:24)



So, that is what we are going to talk about today more, so that is, what is, explained again here, that for ASIC the different is that we have a cell library right. So, in a cell library typically we consider a complete set of, set like and gate and not gate. So, through which we can implement any kind of gate X or X not nand nor and any gate, you can represent using nand and not right.

So, and we have also set of registers like D flip flop. So, whenever you have a circuit where you might have X or gate, we have some other gates, those has to be mapped through this set of gates.

So, that you actually for ASIC, you are effectively replacing all your gates, using a standard, library gates so that those can be. So, those can be finally, fabricated, because the library, the fabrication fully support only this kind of gates. So, that is for normal ASIC flow, but in F P G A as I mentioned, we have L U T D S P and this RAM and ROM.

So, we have to that gate level design has to be mapped to L U Ts right. So, it is not 1 is to 1 mapping that 1 gate will go to 1 L U T then it is a huge design and that is wastage of your resources right. Similarly, it is not that, you have to identify the multiplier and that has to be mapped into D S Ps, it is not only a multiplier as I mentioned, it can be MAC right. A some pre addition, post addition and different combination can be mapped into D S P. So, that has to be done, if you do not do this and if you map your multiplier to a

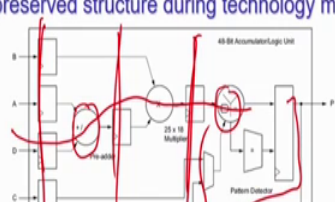
logic gate or L U T, then it will be a, I mean your area will be exponential and may be, if you have large number of, multiplier you, you cannot, may not map your whole design into a F P G A. So, that has to be taken care.

Similarly, this RAM, RAM mapping so. So, the difference from this ASIC to F P G A in technology mapping is, is. So, in ASIC, we have this cell library mapping from the normal gates to cell library gates mapping and in F P G A in during technology mapping, we have L U T mapping, D S P mapping and the RAM mapping. So, this is where, primarily differs the ASIC flow versus F P G A flow on the otherwise, this placement routing all, are kind of, kind of different more or less similar, algorithm has to follow ok. So, we are going to discuss more on this three today ok.

(Refer Slide Time: 17:37)

### DSP inference

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



So, let us start with the D S P. So, as I mentioned that the D S P unit is capable of doing many things right. So, it, it has a pre adder, we have a post adder, we have this MAC summations and also we have three set of registers right. So, the objective is to, map maximum kind of substructure into this structure right.

Because if you have also, you just do a addition in this block, then also occupying the whole D S P, if you want to multiply only 1 to do a 1 multiplication here, then also you are actually occupying the whole D S P, because these D S P cannot be used for something else or if you just do the whole thing right, if you just do B into some pre addition and then post addition and then you do the MAC summation, then also it going

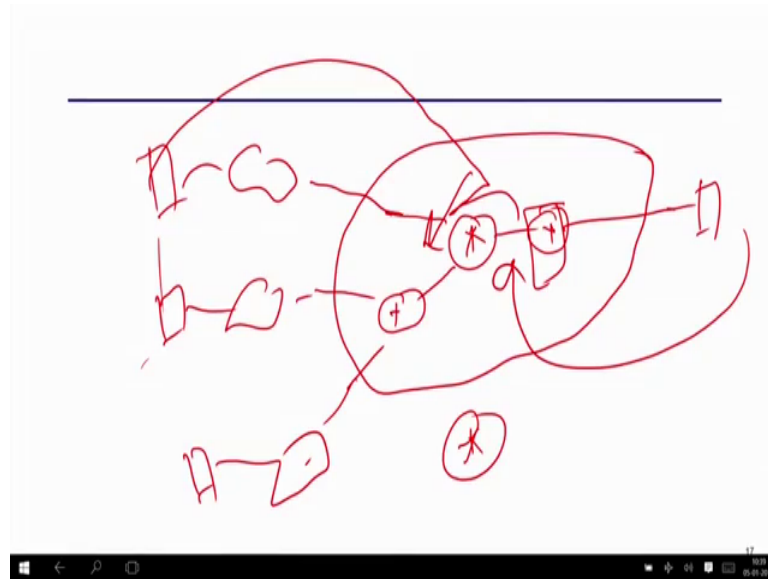
to occupy the whole thing and also, if you just pack some register inside, then what will happen the delay, because if you just have a path, which consist of adder multiplier, followed by a adder, it will be big path, though it is free path, but it is a big path. So, it is advisable that you pack at least some registers into the structure also.

Now, the question is how we can, do this efficiently right. So, that is something is called D S P inference in the F P G A flow. So, you can understand, once you have done the logic synthesis, in the after logic synthesis, what will happen? Everything represent by a gates. So, in, in that case it is very difficult to identify, which is multiplier, which is adder and something like that right. So, the problem is that we have to identify this multiply and accumulated structure before logic synthesis, which is called pre map stage right.

So, before logic synthesis, we have to identify this MAC structure efficiently and, and then you have to preserve those structures. So, that logic synthesis, bypass those structures. So, they should not map those, adder multiplier, those structure into gates, gate level design, then we are not able to map those things into D S Ps. So, the idea is here is that you identify those MAC structure.

Before logic synthesis preserve them, you 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 circuit into D S Ps right. This is the overall flow and the again, I just mention, you try to try to find them as MAC structure possible right. So, the basic, the way algorithm walks is something like you identify all your multiplier in the R T L design right.

(Refer Slide Time: 19:58)



So, you identify all your multiplier in your design right. So, then you, you can map that, but you try to expand that structure right. So, then you try to find out some. So, you find initially, you find a only a MAC, only a multiply, 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 right and then you try to find out is, there any post adder or not right. 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 right.

So, this is, whether this is a multiply and accumulate options are there, are not. So, if you basically, you just try to start from a multiplier and try, try to enhance the structure as much as possible. So, suppose you, you figure out this structure right. So, you figure out this structure, that I have, this a pre adder as well as a post adder, then you can actually have this structure right.

So, this is the MAC structure, we are going to preserve and, but there is no register here, then the next objective will be, you try to do some local retiming, just to move some register nearby, register there may be some registers here right. Some register here may be, there are some other logics here, but finally, HM, there is some register you try to move this registers into this structure, you move this registers here, you move this registers here right. So, register here.

So, that you can pack those register as well as 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 D S P, which is have a pre adder, post adder or a accumulate HM, those structure once you figure out, the structure, you try to fig, find out the nearby locations, what are the, register available? Can we move the register inside this structure? If you move this structure then the whole structure will be, you have a faster HM faster execution in the D S P block right.

So, this is something the overall approach in D S P inference and we, when you find the, this big structure, you preserve it, you do the normal logic synthesis and then you map this preserve structure into the, D S P block during technology mapping stage ok. So, this is overall the D S P inference, flow in any normal E D A synthesis F P G A synthesis tool.

(Refer Slide Time: 22:15)

### 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

Handwritten diagram: A red rectangle representing a RAM block with two ports, one on the left and one on the right, connected by a wavy line.

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 logic array, because logics in the registers then it will be a problem. So, identify the big, big arrays and then you preserve that. So, that you can map them into, into RAM again, the problem here is something like the RAM has a fixed number of port maximum, two ports may be, one write, one write port, one read port or may be 2 read port or 2 write port right. 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 may RAM or not, if it has a lot of accesses in a same clock, may be here say four 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, does that, it try to break the whole array, into two part, or it break column wise or row wise. So, that, that access can be differentiated and you can create a block, where we have maximum two access, at a clock and then we can map that whole thing into a RAM.

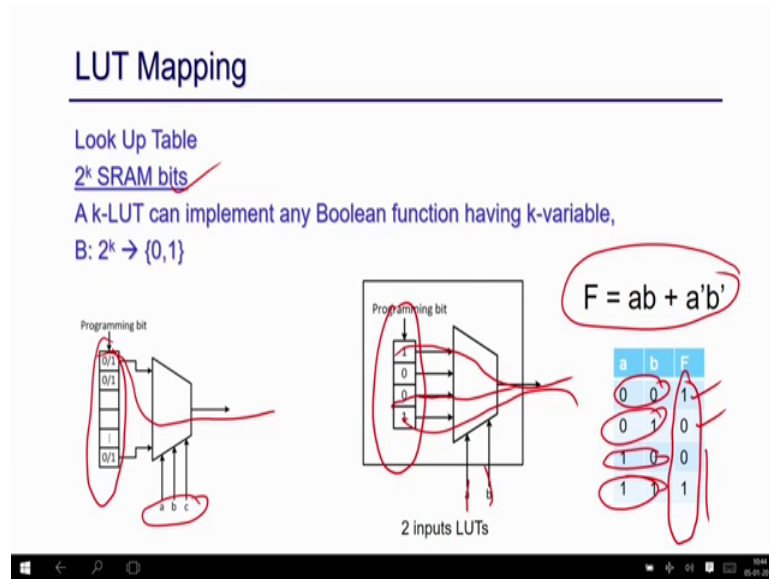
So, those all the integrities are there I am not going to detail of that, but the bottom line here is something that you identify those structure, you make sure that your access is something, you know which is supported by that RAM block in the F P G A and then you try to preserve that and you map it to, into the RAM block during technology mapping.

Also there are other possibilities are also here like, if you have some asynchronous reset to that particular, memory right then it cannot be mapped with F P G A block then probably you have to add some extra logic HM around the RAM, you map that particular big array into the RAM and just to overcome that a asynchronous reset, you add some extra circuit right so.

So, this is your RAM map and some extra circuit that will talk about that asynchronous reset. So, those will discuss in another separate section, but the overall idea is, here is it is not very obvious, just find and map it to RAM, there may be this lot of integrities like asynchronous reset, how to handle that, you might have some extra logic has to be created just to handle that or you have may be a more accesses then you have to probably split HM just to make sure that you have, sufficient number of access which is supported by F P G A and then you map it to the, to the RAM this, RAM block of this F P G A.

Similarly, that you, you have to find out the shift register, identified shift register pattern in the R T L, you preserve it and you have to do shift register. So, this is all is done. So, usually all this things are done before logic synthesis preserve, because after logic synthesis uh, this structure may not be very obvious to identify. So, usually most synthesis tool does all these things, before logic synthesis, but actual mapping happen after, logic synthesis during mapping stage, but identification of those kind of structure is done before logic synthesis ok.

(Refer Slide Time: 25:13)

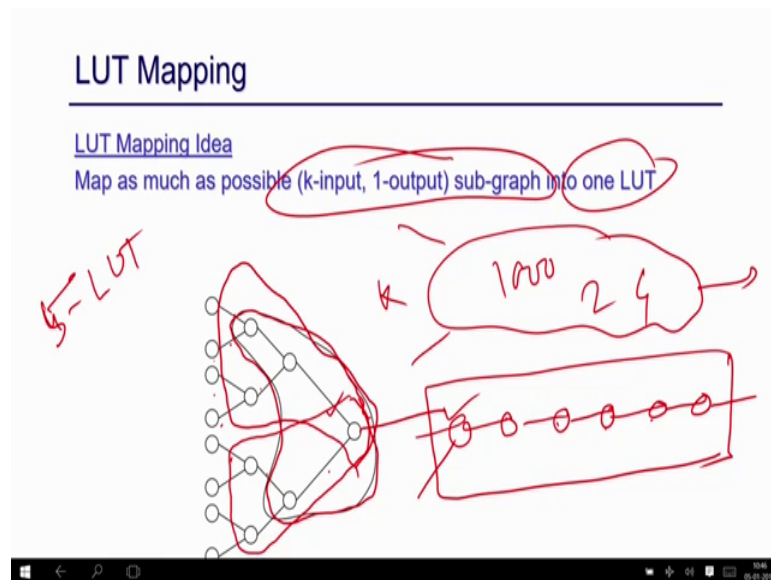


So, now we are going to talk about this LUT mapping. So, LUT mapping is something, look up table as I mentioned earlier that, this is, it has  $2^k$  number of SRAM bits and based on the input volume, we decide the output, which output should go, at the output right. So, this is how the look up table looks like right.

So, if we just take an 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 this outputs in this RAM right, 1 0 0 1 and when my, input and this is the select lines of this marks, when it is 0 0, this bit will come at the output, the mark structure is something like that, when this is, 1 0, this bit will come at the output right. So, when the output is 1 1 then this bit will come at the output right. This is how, we are going to map.

So, now if, 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 right. So, this is the overall objective and as I mention earlier also, I mean you just map 2 lut into 1, 1 gate into 1 LUT is not a, 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 right.

(Refer Slide Time: 26:41)



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

And also there is important point here, it is that HM. So, initially suppose, there is a structure like this right, so this is a structure like this ok. So, the delay of combinational delay was, delay of 1 2 3 4 5 6, 6 gate delay right, but if you map in this whole structure into 2 is 2 K 2 K L U T right 2 L U T; that means, 2 input L U T, then the delay of that particular thing will become delay of 1 L U T, because finally, when you are actually executing this into F P G A, it is just 1 L U Ts. So, it is very- very, important to identify the maximum structure, which can be map to a single L U Ts right.

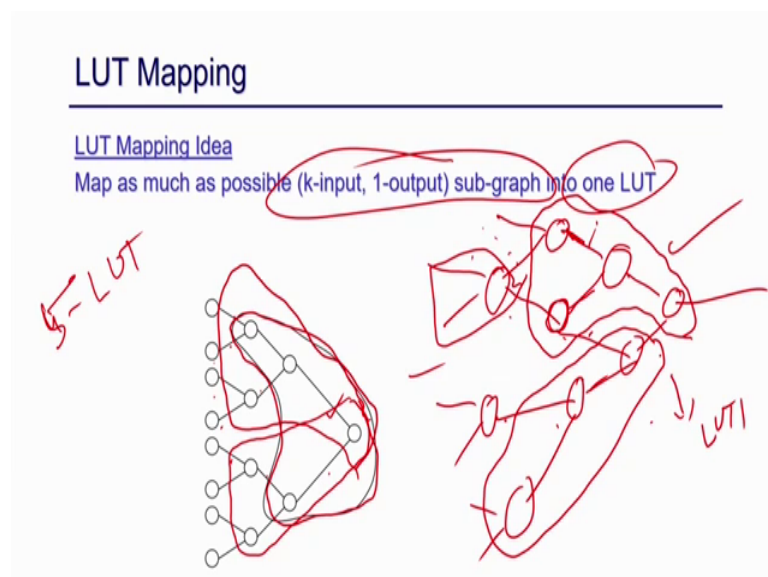
So, this is something HM, we, we have to do. So, if you just think about, the how, the whole mapping idea works, suppose, you have a circuit like this, you try to find out this kind of a substructure right. So, you try to find out a substructure and here input, you want to 3 4 5. So, you can map this whole structure into a 5 5 input LUT. So, 5 L U T and you can understand here, there may be various possibilities here right. So, you can suppose, if you think about a 5 input L U Ts. So, you can actually consider a structure

like this also right. So, this is also a 5 input LUT right. So, here it is having A into 1 input 1 2 3 4 5.

So, this is one possibility or you can actually choose this right. So, you can choose HM, this also, this is also then 1 2 3 4 5.

So, you can understand that given a big logic circuit, the possibilities is very, various right. So, you can have different kind of, this substructure possible and also important that if you select one substructure the next substructure will be different right. So, if you just think about a, a circuit is. So, if you just consider a, circuit like this.

(Refer Slide Time: 29:07)



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 L U T right. So, if you just select this one; I select this one right, then I have to select. So, this is a map 2 1 L U T right. So, this L U T 1 now, I have to map some, because this is 1, output is coming, I have to create another L U Ts here right. So, just to map, because this is 1 output, it has to map here.

Similarly, I have to map. 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 right. So, because I want to get another 4 input 1 output L U T structure here. So, there may be some duplication as well and, and also similarly, since this is 1 output, I can choose this one. So, this is one possibilities.

Now, if you just think about instead of doing, this I can instead of selecting this suppose, I do not select this and I select. So, I, I do not select this one right. So, this is my structure. So, instead of selecting this suppose, I select this right. So, this is also a 4 input L U T right. So, this is 1 2 3 4. So, now, the next structure, it will be different right. So, initially I select one possibility, I select this one then I have to select something here, and something here, and something here right.

Now, if I select this one, now my choice will be different. Now, I have to select another L U T here right. So, this is also one possibility. So, this is one possibilities right. So, this is, this is also, not exactly correct, because one output is here. So, I have to select, this right. So, this is 1 2 3 4. So, similarly I have to select something here, like this right. So, I have to select something here. So, the idea, the main point, here it is that there are huge number of possibilities and one choice decide the next choice right.

So, the idea is that it is not the choice is not unique, there may be various possibilities and it is not like that, if you choose something, you have always a unique choice right. If you select some other way, one way, the next choice will be different. So, the idea here is that, there may be various possibilities.

(Refer Slide Time: 31:33)

### LUT Mapping

---

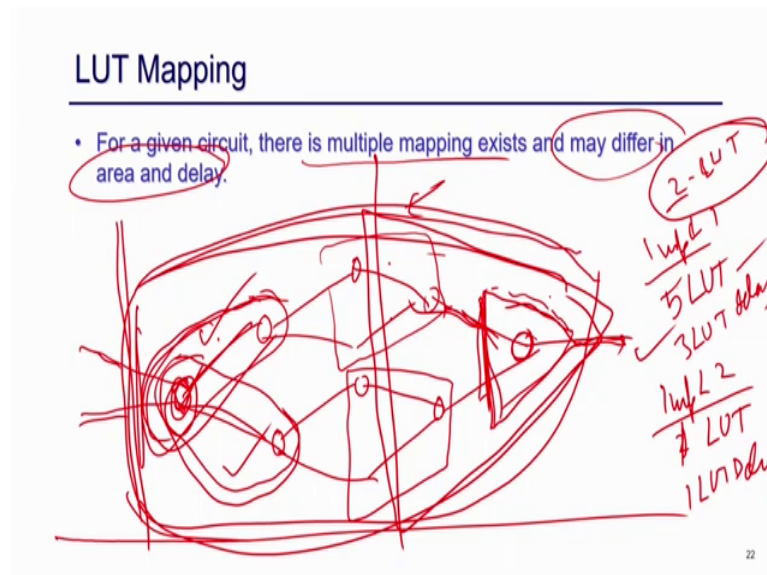
Objective

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

And your objective would be try to map, those whole logic structure. So, that you all the gates of your design is map to some L U T and your objective might be area minimization, may be that mean, you try to use the whole this things using minimum

number of L U T or you try to minimize the delay, the total L U T delay is minimum or you have, both right, you try to optimize area as well as delay. So, your objective can be different right. So, this is something very important.

(Refer Slide Time: 32:07)



And this is in generally N P hard problem. So, usually all the synthesis tool you, you apply some kind of heuristic, just to get a good mapping right. So, as I. So, this is something (Refer Time: 32:20) is important that we cannot have a optimal solution, which will give you the optimal mapping always, but some good, good kind, good solution right. Somehow heuristic ways approach.

And this is also I mentioned that there may multiple solution exist, multiple mapping exist and may differ in area and delay. So, we have to apply some kind of heuristic some, cost, cost function just to decide that this is a good selection, this is a bad selection, this is better than this selection and this way we have to finalize the actual mapping right.

So, and also and 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 given to you and I, my L U T size is 2 input L U T right, 2 L U T; that means, the L U T have only 2 input right. So, how many L U T structure is here and.

How many L U T is required to map this circuit and what is the delay right. So, one possibility is that, you map this into 1 L U T right. You map this into 1 L U T, because

again 2 input 1 map into this L U T, then you can map the whole this structure into 1 L U T, because now, this is also 2-2 input and also similarly, because and this is also into 1 L U T right. So, how many L U Ts are there 1 2 3 4 5. So, implementation 1, I have 1 2 3 4 5 L U Ts right, 5 L U Ts. On the other hand and the delay is 3, there are 3 delay in any (Refer Time: 34:16) right. 1 2 3 1 2 3. So, delay is 3 L U T delay 3 L U T unit right. So, this is the possibilities.

But on the, other hand, you can actually map the whole; whole circuit into 1 L U T right, because if you just think about the whole circuit, I have two inputs one output right. So, implementation 2 is 2 only 1 L U T is required and 1 L U T delay. So, you can understand. So, given such circuit, if you just choose this way then, I need 5 L U Ts 3 L U T in delay of is the circuit is 3 L U T delay and here 1 L U T and 1 L U T delay right.

So, the primary point here, try to, I try to say is that. So, if you start from here and if you just identify nod and whenever you find a 2 input and if you stop here so; that means, this is 1 cut right. Cut is something you start from something and you find out whenever there is a K input is coming to that circuit right. 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 optimal.

Because may be at this point it is grows high right. So, if you just find a cut here the input is 1 2 3 4. So, I have a cut here, which is of size 4 4 input cut, but which is not supported by my target architecture, because I assume [FL] my L U T 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 right.

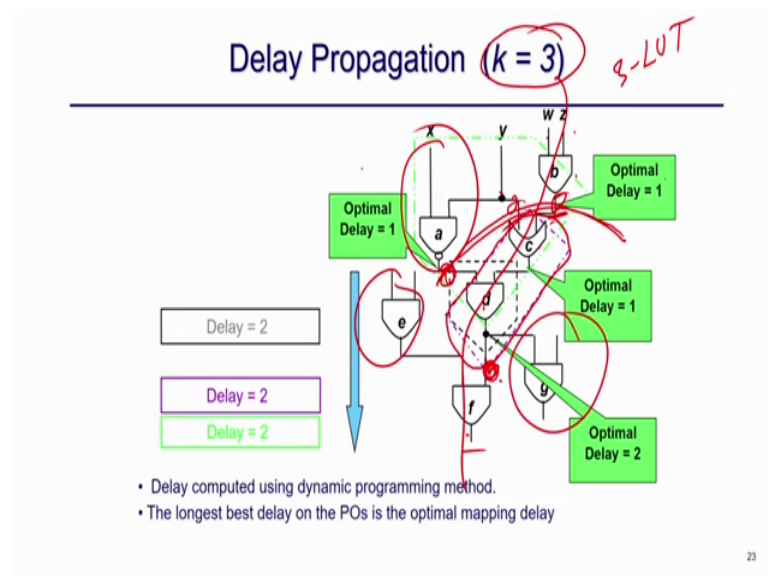
So, this is what is happening. So, it grows and then it sink right. So, if you have this reconergent and finally, you end up in a cut of this, the whole thing is also a cut and this is a whole thing is also a cut and it has two inputs right.

So, this is something also important sometime, even if your target is, is 2. You may have to go further, because after sometime, you might have reconvergence of path and through which you can actually map a bigger circuit, a bigger logic into a single LUT right. So, this is something also possible. So, that is the, the core point or the key point is there, there are various possibilities and various way to actually identify, the number of L, L U Ts and and reconvergence is also important and sometime some duplication also may

happen. So, duplication, when happen, if some node has a, fanouts then this may node, has node, may have copied into 2 L U T s right. So, this, his node is actually place in this L U T as well as this L U T. So, this is called duplication.

So,, duplication is something should be avoided, because unnecessary I am mapping this things to multiple node right. 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 ok. So, we will discuss all those things detail.

(Refer Slide Time: 37:12)



So, before that how to calculate the delay, what is the L U T delay. So, in this circuit, you have, you can see, there is path from here to here. So, this is the maximum longest path and 1 2 3 4. So, 4 unit of logic delays right. So, logic gate delays in the normal R T L circuit, but if you map this to into the L U T s, what will be the, what is the delay right. So, this something, we are going to understand.

So, suppose I map this into, this into suppose, my target is 3 input L U T right. So, 3 L U T so; that means, a L U T can have maximum 3 input. So, suppose I map this into 1. So, at this point delay of this is one right. So, this is the delay, optimum delay at this point is 1, because this is 1 L U T delay. So, now, suppose I map this also into another L U T. So, at this point sorry. So, I map the whole thing, this whole structure into 1 L U T, because now, I have three input right. 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. Similarly, if I just take this example. So, here also the delay is 1, because this is map to 1 L U T and now, if I just take this, this, I map, decided to map this whole thing into 1, then how we calculate the delay here, the I will just find out the delay of the all the inputs, max, maximum delay and I just add one more delay here right.

So, this is optimal delay at this point one of this at this point is. So, this is not there, at this point optimal delay is 1. So, this should be optimal delay of 2 right. 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 right. So, what is the maximum possible? So, given a mapping, what is the delay of that particular, LUT mapping right. So, this is we can actually identify and which is basically nothing, but the maximum delay of, all the inputs right. So, all the maximum possible delay of the, all the input plus 1 this is how we can actually estimate the delay of your mapping ok.

(Refer Slide Time: 39:23)

## LUT Mapping

---

Steps

- Generate/enumerate all possible cuts - cut enumeration
- Select minimum cuts to cover all nodes - cut selection to minimize optimization goal
- Possible iteration to remap nodes on non-critical paths (area recovery)
- Takes into consideration node duplication

• NP hard problem  
- Heuristic based approach

So, now, I am going to talk about this LUT mapping detail. So, as I mention that LUT mapping, is something we have to find out the cuts. Cut is something, cut means, what a cut of size K means? There are K inputs 1, output.

So, we have to first do all possible, 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 type 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, minimum number of cuts, which actually cover all your gates, where we have minimum number of duplication right. So, this is something called cut selection. So, there are two primary steps here, cut enumeration and cut selection and as I mention this problem is N P hard, we do not have a exact solution. That the solution is, this is the minimum possible, mapping of the gates into a, K L U T right.

So, we always have heuristic base approach or which is basically, kind of iterative. So, you do some one mapping and then you try to do something, try to change little bit, do another iteration, try to do some modification in the mapping, try to get a better, better result right. So, this is what is called, this, this iterative approach right.

So, what is happening here? So, if you have this non critical path. So, if there is a critical path here. So, suppose, this is a critical path, I cannot change anything here, may be some, something can be done for this node, this node or something other nodes right. So, which is not in non critical path.

So, there mapping can be little bit of change. So, that I can have a better mapping, whether I can, minimize uh, more areas right. So, this is something the overall idea that I have 1 mapping and then I iteratively try to improve the area over that particular mapping right. So, and one more, one of the key factory is the duplication. How to remove the duplication? As I mention earlier also ok.

(Refer Slide Time: 41:25)

## CUT Enumeration

---

### Cut Enumeration

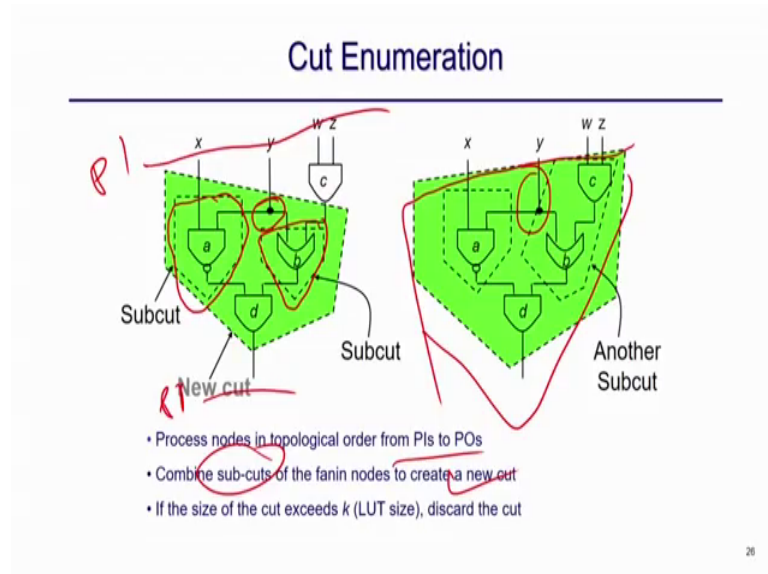
- Process nodes in topological order from PIs to POs
- Combine sub-cuts to create a new cut
- Search for re-convergence

25

So, let us go into this cut enumeration steps. In cut enumeration, the basic steps is like this. So, we process the node 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 you, even if your target is to map the things into the k L U T, but you should not stop whenever you find a cut of size K, because that may not be end of it. So, you might go little bit further and then finally, you might find a reconvergent path and that will give you a better mapping. So, which I already give an example so, I will talk about the other, other, other aspect.

(Refer Slide Time: 42:10)

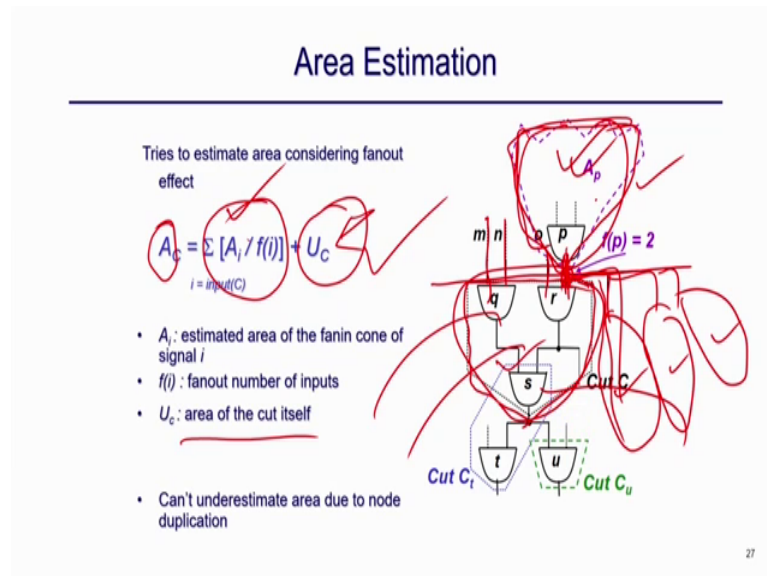


So, suppose this is my circuit. So, I will start from the input right. So, this is my primary inputs and this is primary outputs. So, this is P I and this P O. So, from where this, I will going to start right. So, as I mention, topological order from the primary input, output and then I try to find a cut, of my given size  $K$  cut right. 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 right and then I can actually combine this 2 cut, because they have same input sharing and I try to create a cut of size 3 right. 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 right.

So, this is what is called, this try to combine the sub cut, just to create new cut, because our objective is to get all possible cut of your design right. So, as, and this is one of one of doing this. So, similar one example here so, this is 1 cut and this is 1 cut and since there is a 1 input sharing here, I can actually combine these two, to get a bigger cut of size 4 right.

So, this, this is how the HM, the new kind of cut will generated. So, I will start from the inputs and then try to find out some cuts and our maximum 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 are 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  $K$ .

(Refer Slide Time: 43:41)



So, now the important factor is that as I mentioned the number of cut may be huge right and there may be lot of overlapping. So, which has to select. So, there may be say 10000, possible cut has created only 5, 10 or 20 has to be selected. So, which one has to be selected right 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 right, cost value.

If the cost is less; that means, 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, that is something, and based on that we are going to choose the better cut right. So, that is given by this. So, there are two factor cost contributed by the specific cut and the cost of the input that I am going to discuss right.

So, suppose I have this cut right. So, cut means what 1 output and K output. So, cuts of size K means, there are K input 1 output. So, suppose, this is my cut, this is the cut and. So, so I am going to talk about, you see in the next slide that what is the component, 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, I say [FL], this is a 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, input I right. So, this is the area of the input I, because the input, I also will be map to some other, other L U T right. So, what is the area of that particular input and how many fanouts are there for from this

input right. 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 right.

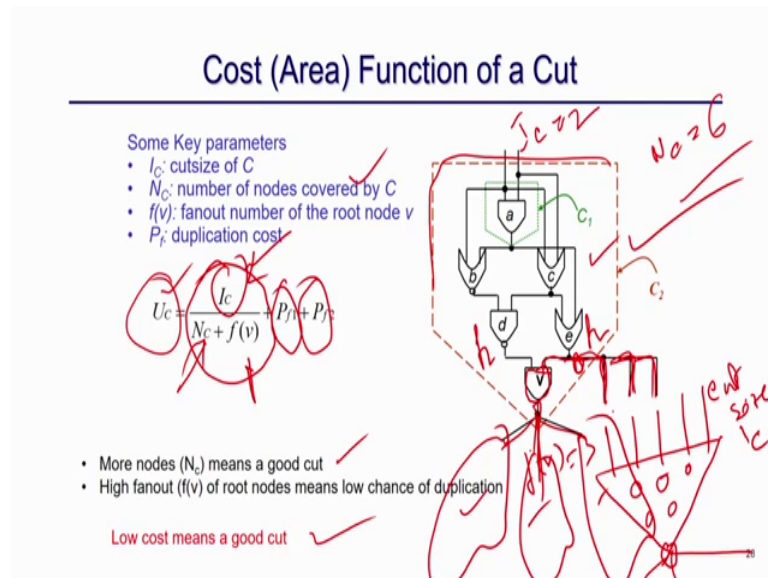
So, this. So, this input will going to map into another cut also. So, that the effect of this area will be distributed among this 4 right. So, that is how this, this comes, that the area of that particular  $I_{th}$  input divided by the fanout value of that particular input right. So, if the fanout is more, this will be less, if the fanout is less, then this, this value will be high right.

So, this is something, how to tackle the input and, how to handle the fanout of the input also right. So, this is how, we are going to, handle the area. So, here, one point to be noticed here is that the only, the cost is not associated 1 cut, because as this is interdependent, if you select 1 cut somehow their input, we have to select some other cut from their input right 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 the inputs and outputs, the associate or the surrounding, cuts also and what is the surrounding, surrounding cuts those are the inputs right.

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

(Refer Slide Time: 47:22)



So,, the first thing is here is like suppose, is, the factor is like suppose, this is my cut ok. So, key parameters, this is the cut size a cut size of C  $I_C$ . So; that means, cut size is the number of input right. So, if I just select a cut like this, this is the cut size right.

So, this is, this. So, this is the cut size  $I_C$ . So, the number of inputs right and then  $N_C$  is the number of node inside right. The number of node inside is the  $N_C$  by the cut and then the  $f_V$  is a fanout number of the root node. So, the output node, what is the fanouts. So, there may be multiple fanouts here, right. So, this is the fanouts, this is the  $f_V$  right and this is  $I_C$ . So, for this cut I have 2 input right. So, this  $I_C$  equal to 2 here and what is the  $N_C$  for this cut?  $N_C$  for this cut 1 2 3 4 5 6 right. So, there are  $N_C$  6 for this cut and  $f_V$  there are 3. So,  $f_V$  equal to 3 for this cut and then that  $P_f$  is the duplication cost, I will, I will talk about that.

So, the cost contribution from this particular cut is given by this factor ok, this is the primary factor. So, it is, its saying that  $I_C$  by  $N_C$  plus  $f_V$ , what does it mean. So, more nodes in the graph; that means, its a big good cut right. So, if you pack more nodes in a in a particular cut; that means, a good cut, because we are trying to pack more, many or big number of gates into one cut.

So, the, this is a good cut and as I mention the if the value is  $U_C$  is low; that means, a good cut. So, that is why this come into the output right in the, I mean here and also as I mention here, if the there are more fanouts from this output.

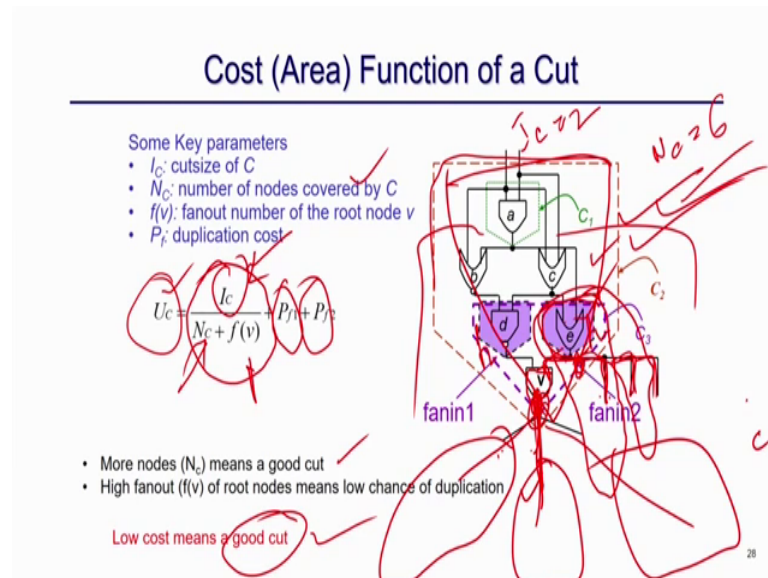
So, this cut will be shared among multiple cuts from here, because I will have another cut here, I will have another cut here, I will have another cut here. So, this cut will be shared among all this. So; that means, it has a positive impact, because one cut I am going to reuse in multiple places. So, that also reducing the cost factor, and if and this is. So, this is  $I_c$  is the number of cut size.

So, the you, cost of that particular cut is given by  $I_c$  by  $N_c$  plus  $f_v$  you can understand if the number of node is big. Now  $f_v$  the fanout from this node is big. So, this value will be less right. So, that is something. So,. So, that will become a good cut, and the other two factor is something I will talk about, now which is basically duplication cost ok.

What is the duplication cost, so you can see here suppose this is my  $f$  node, so root node right. So, root node is the last node and if the input of this is  $f_1$  this is  $f_2$  right. So, the input of that root node and from here this is very important things to notice, if there is a fanout right, 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 right.

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 right. So, this is very important. So, same, but this is going to use in somewhere else right. So, this fanout is going to use somewhere else. So, the problem here is that; that means, this node has to be duplicated in some other cuts. So, for these, whatever the cut I am going to generate, I will just draw it, so you will understand. So, let remove this.

(Refer Slide Time: 50:59)



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

So, this is the problem. So, if that particular that root node, the parent node that previous node of the root node has multiple fanout; that is actually bad thing, because if has multiple fanout that has to be duplicated in different other cut also, which is actually unnecessary, so that has a negative impact, so this  $P_f$  is given by this.

(Refer Slide Time: 51:42)

### 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.

So, as I mention here, so this is given by this, so this, this. So,  $f_1$  and  $f_2$  I talked about. So, this is. So, in general this is  $f$ . So, the number of node in that particular cut at the input and this  $I_c$  is the size right. So, if the point is here that if there are number of nodes is here is high, then the possibility of duplication is also high right; that means, there is a larger duplication cost and this  $I_c$  is the normalization factor.

So, as I mention here, if that particular in the cut. So, So, you always consider sub cut, where as if the in the sub cut of this input  $f_1$  has more number of nodes, that there is high chance some of them have a fanouts and that will create a duplication. So; that means, it will improve, increase the cost of that particular cuts; that means, its a bad cut, the basic idea is that if if select a cut and if the internal node has some fanout that is basically redundant thing.

So, that node has to be duplicated for some other cut. So; that means, you should not choose this kind of cut, you try to avoid this, choosing this kind of cut, because this is something there are not or node will be duplicated in some other places. So, that will increase the cost, as I mention 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, there is a high chance of duplication. So, we try to increase this cost of this particular cut and we try to mention that this, this may not be a good cut ok.

So, this is how we are going to handle this duplication. So, this is something we talked about. So, there are three aspect here uh, the first thing is that we generate, we 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 mention here and then also we try to remove the duplication, avoid the duplication this way just to increase the cost and also I as I told the reconvergence also, has to taken care that example I already given.

So, these are the all this things we are done iteratively just to find out all possible cut, so that I get all possible cut of your in a design right. So, once you have generated this the next step, and as I mention that cost is something is given by this factor, which has impact of that own, own factor, the size from the cut size and that (Refer Time: 54:10) that fanout of the internal nodes that will come into picture, as well as the factor is from the input also.

If there are some input which has multiple fanouts which is good right. So, if as I mentioned again that if the output of a cone, if the output, output of a cone has multiple fanout which is good thing, because this, this is a good cut, because this result we going to use somewhere else, because none of the node will be duplicated here.

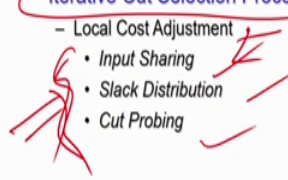
So, since this is one output, this will be one cut, this will be one cut, this will be one cut. So, this is going to be good cut. So, in a good cut if the fanout of that root node is high; that means, a good cut, but if there are some internal, would have lot of fanouts which is bad thing, because those node effectively will be copied to somewhere else and which will result in lot of duplications and most likely have number of more LUTS right. So, this is basically a bad, bad cut ok. So, once you have done this we have to select right. So, we have to, going to select the cut.

(Refer Slide Time: 55:09)

### 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
- Nodes 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 mention we have again, this is something heuristic algorithm is happening. So, our objective is try to cover all the, all the gates of your gates of your design using minimum number of cut, or maybe you try to area is minimize or your timing is also minimize ok. So, that is something has to be there, and the choice here is that if there is a critical path, I just 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 in the critical path that cannot be change, but if something lies in the non critical path, there I can actually make some choice some alternate, so that I can improve the area ok. So, usually it happens in iterative process. So, I create 1 solution, then try to improve the solution in every iterations, and there are some other factor also which is, which is important right, as I mention that this cut are interdependent. So, once you. So, once you select any cut, I associate some cost adju[stment]- with it right. So, using the logic I just define in just previously right.

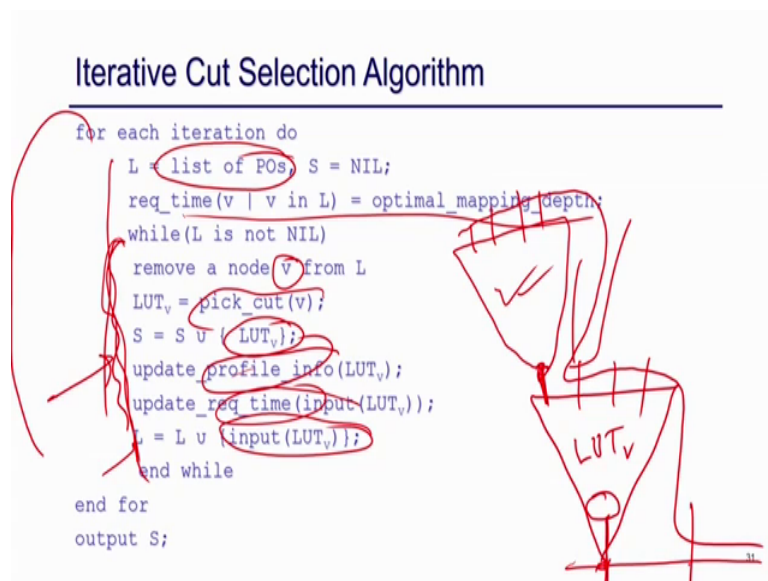
So, every cut has a cost, but, and once you select a cut and, so that actually make some changes in the surrounding cuts, because may be some fanout is not going to consider, some will be ignored, some will be considered. So, they have some impact on the surrounding cuts. So, that is something called local cost adjustments

So, once you select a cut we enumerate all the cut associate a cost of that, cut using this function right. So, this just function using this function ok, sorry using this function ok.

So, every cut has a cost, but once we select a cut. So, because I generate all the cut after now I am going to, I am not going to select all the cuts, I am going to select some of them.

So, once I am going to select some cut then it has some impact in the surrounding cuts, as I mention the actual cost also involve the input fanout in 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 has may have some impact on the surrounding nodes, so that has to be adjusted. So, that. So, that local cost, cost assumption is that. So, for that I have to consider the input sharing slack distribution and cut probing. I am going to discuss those have a impact on the cost and I am going to update the cost of the surrounding cut using this logic ok.

(Refer Slide Time: 57:43)



So, here is the overall algorithm I am going to go, going into briefly. So, this is that iteration right. So, you can see in the within the iteration this is the one one loop in which I am going to, I am going to do one choose 1 solution. I am going to get 1 solution and the outer loop I am going to select multiple. In iteration I am going to generate multiple solution, I am going to improve the solution ok

As I mentioned the start, its starting from the POs the primary outputs, the cut selections start from the primary output ok. So, my L is that list of P Os primary output of your circuit and then I just find out the required time using some analysis that what is the

optimal timing date and all those things using that logic. And then what I am going to do? I take one output and then I pick a cut, because from that cut. So, suppose this is my circuit, I took a output node right, so this is my output. So, from this enumerated list, there may be many cut right may 10, 20, 50 cuts possible starting which has the output this.

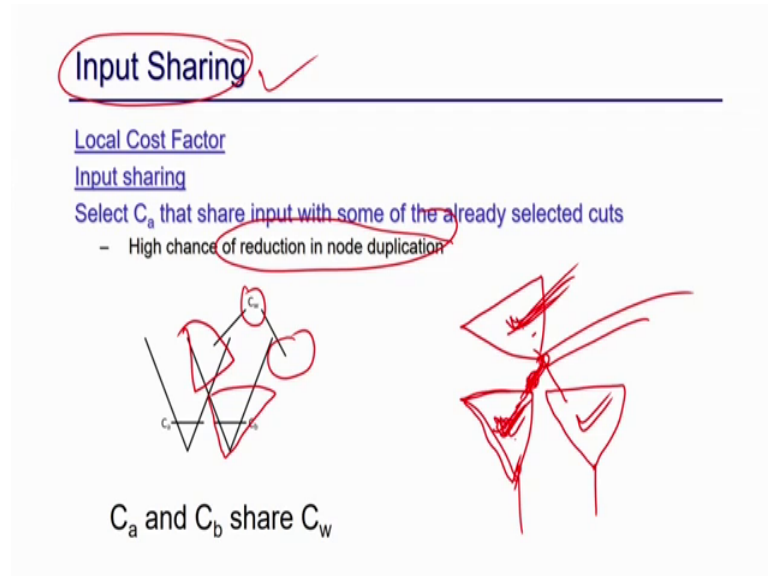
So, from that I am going to choose one cut which has the less cost, I am going to chose the cost which has the less cost, which is a good cut. So, once I choose that, so this is the 1 LUT I have already decided right. So, suppose I choose this, this cut. So, suppose I have two input output initially. So, my P O has only 2, this and this. So, I decided to choose this right, and it has say four input what will happen now, so I have, and now my list will include by this input of this L U T v. So, this is my L U T v right.

So, I am going to choose this L U T 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, this input right. So, now, my L will consists this right. So, initially I have 2, this 2. 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 from starting from this node of the cut, where the output of this is this right. So, this is how the whole thing works. So, whenever I am going to choose this one, my cut set will become this, this followed by this and this right.

This is how the whole thing works. So, every iteration I am going to select one cut and I am going to update that P O list that is the logic I define and also this, I have to also do this, this h m, I have to do this update profile, update profile as well as this update required time as I mention 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, through which I am going to ok. So, as I mention once you select a cut we have to update the surrounding cuts, update this timing profile info using this local cost factor, where I am going to discuss now and this. So, after this 5 loop I have one solution right. So, this is 1 LUT, this is 1, may be selected and so on. And once we have this, then in the next iteration whatever I am going to, when I am going to talk about this timing information or not I am going to talk about the timing information of the previous selection, so that I can decide whether the my current iteration is better or not right, this is how the whole algorithm works.

(Refer Slide Time: 60:59)



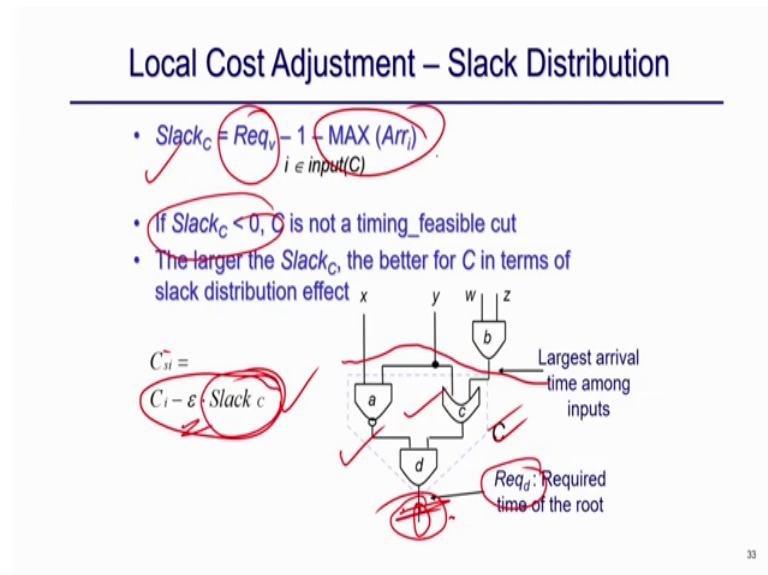
Now, if you just think about that local cost factor one of them is input sharing which is very important and I also mention earlier also. So, suppose I have decided this is one cut and this is one cut right. So, this is one cut and this cut is here in many place. Now once I start from this, just to elaborate 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 ok. So, this is something I never know which order it will go. So, suppose I have selected this one. So, now, if I start from this node and if I found that one of the, I saw a cut where one of the input is coming from one of the cut which is already selected right.

So, that  $C_w$  then, then this cut is a better cut right, because this particular input is sharing among multiple cut which is already decided. So, there is a high chance that the duplication will be less here. So, so in this case, I am going to improve, reduce this cost of this cut further right, because I am now in this particular cut, where I have already selected this two cut and I found for this cut. One of the input 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 right. 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 chance of getting selected of this cut is high right. So, this is what is called input sharing. So, if input is coming from already selected cut, which is should be a good cut and I mention also that if a cut is selected and it has a multiple fanout; that means, it is a good cut. So, it

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

(Refer Slide Time: 62:57)



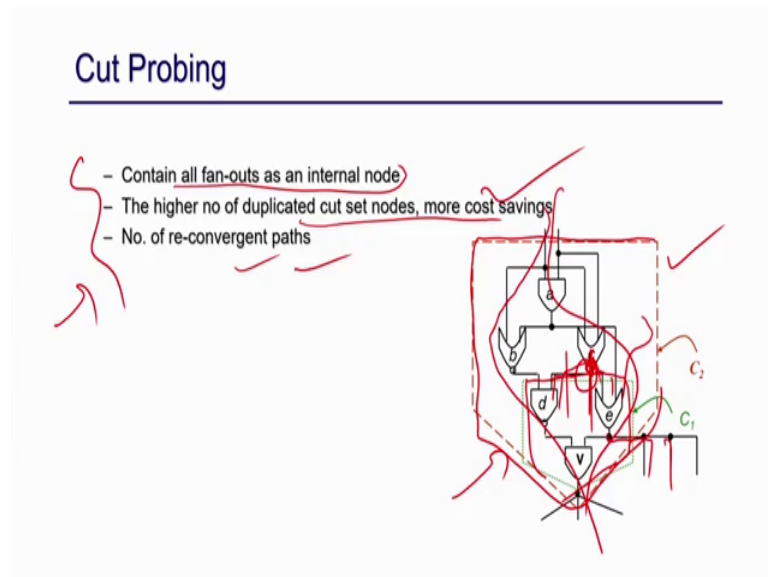
So, how so, suppose I know for this time, from the timing information I need the required timing here, the data should come at here by this time right. And now if we if we. So, how if we found that the arrival time of the input the max of the arrival time is that right, so that max of the arrival time. So, this is the largest among this is this? So, now, the slack is what the required time minus this right.

So, if I found this at this point, if this is 0 less than 0; that means, this is not a possible factor, because you the data is coming later than what is the expected. So, this cuts should not be selected right, because this is not a feasible cut, because if you select this way, because the data will come here later than the expected required time. So, if, but if it is positive this, this is that positive this, this value 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 area.

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 local the aspect right. So, if I just find [FL] local, the timing of this particular cut is positive; that means, if you see 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 right, this is what is called slack distribution using local cost during local cut adjustment and also this cut probing which is also an important factor.

(Refer Slide Time: 64:32)



I mean if I just found [FL] if a particular node as I mention. 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 right or you can think about. This is my whole, this is the bigger cut I have selected and where the internal node has fanout, which is a bad thing, but fortunately all the fanouts are part of the same cut, so which is a good thing right. So; that means, its not going to increase the size, it will reduce the duplication and also positive thing is, this is a kind of reconvergence right. So, it will improves the number of node getting selected inside that will be high right.

So, that is the point that if all the fanout have a internal and its part of a cut, which is good thing. So, it, I am going to reduce the cost of that particular cut further. So, that this make it better cut also if the. So, the other example that I just give here for this cut, if I found the input, multiple input coming from single source, which is also a good right, so; that means, which is also good; that means, I am not going to, all of them are coming here. So, the high chance is that, this the number of input of this is less, because now I

can consider both of them as a single input and I can select something else as well right. So, this is also most cost saving.

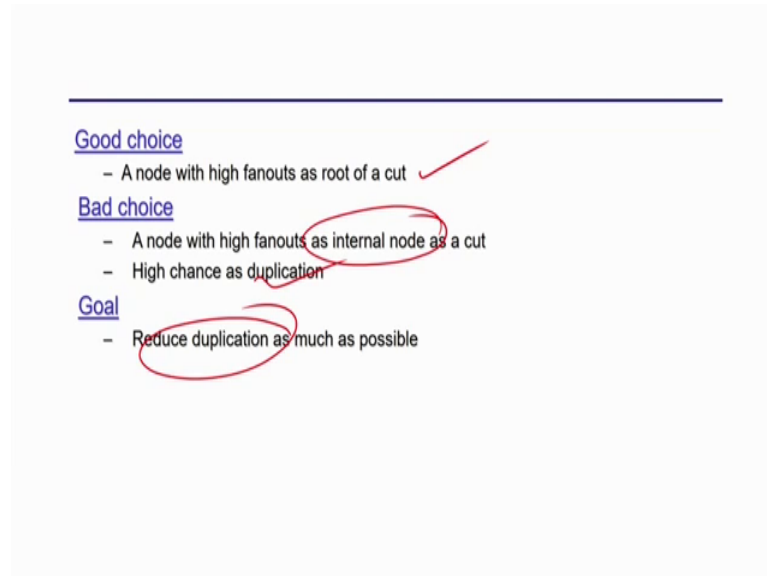
So, this is also have a positive impact on the cut, similarly the number of reconvergence path. So, if I found something which is going further and then slow down, similarly there are two reconvergent path here. So, if the number of reconvergent path is high; that means, circuit has grown and then reduced. So, which is good thing.

So; that means, the possible of, number of number of gate that is getting map into this particular cut is high. So, again this all, this three aspect actually improving the getting chance of, getting selected of this cut is high. So, I am going to reduce the cut cost of that particular cut, further using all this scenario right. So, this is something I am going to do it locally, because, and now this is also decide about the current scenario, because once I have selected something the next iteration. I am going to update this, because I know now the surrounding things what is happening there right.

So, this is all this three this cut probing, this slack distribution and the input sharing is the three factor through which I am going to improve the factor, the cost of the cut locally right and during the iteration, and this is how the this is how the whole selection algorithm works ok.

So, this is how the whole thing works that I decide. Start from the output I am going to select a best cut, but I am going to select this peak cut, I am going to update. 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 that their input will become the P O now right, because this its selected their output will become P O, and then once this is selected their output also 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. So, this is what this LUT LUT mapping all about.

(Refer Slide Time: 68:59)



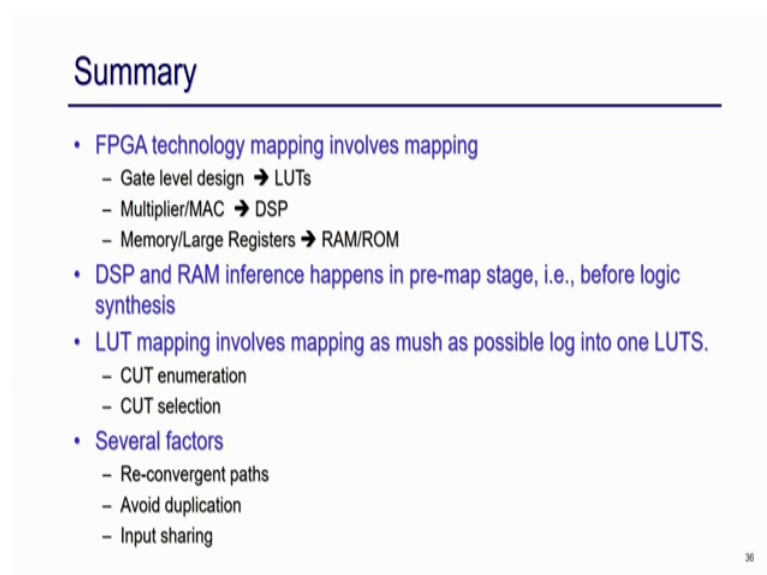
The slide content is as follows:

- 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

Handwritten red annotations include a checkmark next to the 'Good choice' item, a circle around the 'Bad choice' section, and a circle around the 'Goal' item.

And I as I mentioned that if a node have a high fanout as a root of a cut, it 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 try to reduce the duplication as much as possible and try to map the whole thing in minimum number of LUT and minimum number of delay ok.

(Refer Slide Time: 68:22)



The slide content is as follows:

### 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 F P G A technology mapping, is little bit different from ASIC mapping, because here we have dedicated resources like L U T S D S P S and

RAM, and our objective is to map the whole thing in this particular blocks and this D S P and RAM inferences, usually happen during pre mapping stress, so that because we have to preserve those structure. So, that that does not get flatten or may dissolve resolve into the gates.

So, that D S P or RAM inference cannot happen and LUT mapping is a hard problem, a computationally hard problem. So, usually happen, we apply the heuristic base approach and there are 2 steps; primarily here, one is cut enumeration, second is cut selection and we also apply several kind of other factors, cost factor; like reconvergent of path avoid duplication input sharing, cut probing and those kind of techniques just to re adjust the cost and try to select the best cut ok. So, this is the overall summary of this technology mapping of F P G A.

Thank you.