Digital Switching

Prof. Y. N. Singh
Department of Electrical Engineering
Indian Institute of Technology, Kanpur

Lecture – 21

(Refer Slide Time: 00:18)

Last time what we were looking at was delta network, and I had not given the formal definition so far, but we actually I was trying to give the fundamental reasoning why, a delta network is going to be a delta network, and I told you that when I am actually connecting the inputs or the outputs from the previous stage, and since the output, this is being selected based on a certain digit in the address. So, if I am going to select all the inputs, if any one input is being selected from say some value x, and if I choose from some other value y, then there is going to be a problem, actually because this onward path will be decided by different set of digits. So, same digit cannot be use to reach to the same output, actually; same combination of digits; that is not possible.

So, all the inputs; this is x. Then, this also has to be x, and wherever, you are connecting; this also has to be x. This also has to be x; that is the condition which, need to be satisfied. If you can build up a banyan network with this condition, you will get a digit control routing network or delta network. For that 8 by 8 case, I think that is what we were discussing last time. I had drawn something, but something like this. Now, this is 0; this is 1. So, I can actually take connect in any way I want. I need to build up only a multiplexer tree; that is very important. I should not be able to isolate; I should not lend up with the situation where from a input, you are not able to reach to all the outputs; that condition should never happen. So, there has to be a demultiplexer tree.

So, if it is 0, for example, I can connect here; one I can connect here; I can do whatever way I want, actually. So, I am just connecting something arbitrary, actually, and from here, there are again two things. These are again, two. I can connect it any fashion I wish, something like this. So, I am just making an arbitrary demultiplexer tree, and based on that delta network. Next one, which I will take; this will also have now two outputs. I can actually, use these or I can use this system material for me, as far as I am going to share multiplexer tree, it is fine.

For example, I can actually, choose. When I am connecting here, it is here connected to 0. So, this one cannot come here, remember; because then it is not going to be delta network. So, I have to connect only a 0 here and 1 here. So, this leads to me only one choice, but maybe, I can actually, connect; this is 0. So, I connect this thing here, and I am able to cover up these three. So, part of the multiplexer tree from here, is this to this; these two, I have not covered, actually so far. For covering these two, I have to choose from these, actually from anyone of these two or I can choose this one. If I choose this, these are like 4 x 4 by 4; this will be another 4 by 4 and then mapping. So, I can do it another way around, also.

For example, if I choose this one, this one cannot be connected to these two, because then I would not be able to form demultiplexer tree. I have to connect this mandatorily to only, these two conditions; no other options. So, there is a 0, and there is a 1. So, this one is connected to 1. So, this one has to go to here, if it has to be delta network. I am just creating a random topology; I have I have not preplanned it; I am just doing it here, in the class, just to show as an example; how does it work.

The next one is this, because these two has been already covered for this. This and this was not there. This is also 0. This will also be 0; you can check, actually, if I put certain number, this 0 from here, 0 here and a 0 here; if all three digits are 0, I will always land up with that box; any one of these four inputs. All three 0s will always give; I will come to this output. That is the only map, which is possible. If I put 0, 0, 1, I already take 0 here, 0 and this will be 0, 0, 1. So, I am actually, assuming there are three bits; d 2 will be use for routing here; d 1 will be here and d 0 will be here.

I can now, explain for the other one. There is a 0, and there is a 1. I have to build up a de-multiplexer tree. I can choose this one. Now remember, this is taking input from 1. So, if I choose this 1 option for a de-multiplexer tree, I am covering these two, and this guy or this guy has to be chosen; this cannot be chosen in that case. But of course, later on, you can find out that you can do shuffling of these things, move here and there, twist around, and you will be able to build up some nice topology. So far, it looks very bad. If I connect, this is 1. So, I can only connect 1 here. That is the condition for delta network. This 0, because I have connected here, which is connecting these two. So, I have to either, use this or this. This is already occupied. So, I can only use this option. I will take this, and this will take 0.

Student: (( )).

This one.

Student: Yes connecting one.

This is 1 and this is 1.

Student: Yes sir, my doubt is that (( )).

The bottom one.

Student: Yes sir.

Can I connect this 0 here?

Student: No, not that sir. Can I connect 1, not connecting that 1 down, sir?

This one?

Student: (( )).

Here?

Student: Yes sir.

How? Because 1 can only be fed here, and one can only be fade here. The two 0s have been taken; two 0s have to come here. If you do this, you would not get a 0, 1. See, you have to understand the balance, because this is a binary output. So, half of the nodes will take the inputs from all 0s; half of them will take from all 1s. So, you have no choice actually, in that but you can make arbitrary lot of combinations. Each one will give a different permutation. Different permutation will be set of permutations, which will be permitted.

Now, take from here. This is 0; this is 1. This 0 has to go obviously, here; there is no other option. You cannot connect a 1 here, and this 1 will be taken by this guy. So, where this has to go? There are only this and this, which are available. This is taking 1; this is taking 0. So, this 0 has to go here, and this 1 has to go here; this is another example actually, now. You put a 0, 0, 0 here, where you will go; go to top; top comes here; put 0, 0 here; top, again, you will come here. Only thing, the numbering is not 0, 1, 2, 3; is not like that but numbering is fixed. So, the moment you give address as 0, 0, 0; it will always come at this output. Of course, if there is a packet going for 0, 0, 0, direction, this switch; this line, and this line, and this line will be busy. There will be some other pattern, which may not be permitted; only certain set of patterns will be permitted, and that is a specific permutation, which is allowed.

This is another delta network of 8 by 8, but usually, we can keep life simple; this has been too complicated. So, best way is to go for that inverse bean configuration, or then the; we will actually, not talk about another things, which is known as shuffle net. Because usually, what happens is whenever, you are going to create, fabricate this stuff; each one in a VLSI or in optics; this is nothing but a small processor in itself; it is buffer and everything, and I have to interconnect. So, if it is an optical thing, for example, lot of buffers, lot of this processes have been built; I stack one on top of other; these stages. I have to make an interconnection between them.

This interconnection has to be something, which is uniform, and I can repeat it. This interconnection is different; this interconnection is different. Actually, it is not desirable; neither for VLSI, nor for optical computing implementation. So, both ways; this will not be suitable, actually.

Student: (( )) number of delta network?

Yes.

Student: Delta network (( )).

Here also. Here, you might actually, I do not know; I have not checked.

Student: So, it varies from?

It varies from each. What happens? Total number of permutations is 8; 8 factorial.

Student: (( )) different kind of combination or not?

Right, because only 2 raise power 12 possible combinations can be set, because you can either, be in cross or bar state; there are totally, 12 switches, and each can take 2 states. So, 2 raise power 12 possible permutations can be set. 2 raise power 12 is smaller than 8 factorial. So, 8 factorial minus 2 raise power 12; those combinations are not permuted by a switch. 8 factorial is higher than 2 raise power 12.

I can prove that. This is 8 factorial, fine. 2 cube; 3 [fl]. No, this three will also, 2 into 3, 6 is being broken. So, how much is this number? [fl] or 3.

Student: Sir, without calculating also (( )). 77, 2 raise to the power 2, 5 is greater than 2 raise to the power 2.

Yes [fl]. 2 raise power 7, 2 raise power 5, 2 raise power 5 is 32, and this number is greater than 32, actually. So, this much is the difference, which you will have in the permutations, but I am not interested in the permutation as of now. I am only interested in kind of, what is a blocking probability; that is basically the issue.

Now, coming to the actual definition; this is 2 by 2; all life is very simple. If you want a simplified version, I think, go with the Ben's network. So, I think the most important thing, which is there is you have figured out that how to connect, to create a delta network. Make sure, you always create a de-multiplexer tree, and the inputs to a switch should always come from the same output number, from the switches in earlier stage. There has to be exactly, one connection from a switch in the previous stage, to the switch in the next stage. Another important thing; none of the input and outputs will be left free in the intermediate stages. All outputs in the output stage will be the outputs, and all inputs in the input stage will be the input or to the switch matrix.

(Refer Slide Time: 14:30)

Another one, 8 by 8; I think this, I had done earlier, but let me do it again. I will make, give a general proof for a shuffle thing; shuffle is a generalize shuffle, I will be talking about; not for this one. This one is pretty simple. I can take 0 here, 1 here. This is 0, 1, 0, 1. Remember, this can only take input from 1. This can only take input from 0. So, that condition has to be satisfied. So, this 1, pre, I can make another 1. I am actually, drawing a shuffle net here. This is 0. So, I can go here. This is 1; so this can go here. This is 0; so this can go here. There are two trees; no links are common between these two trees actually, so far. There is no link, which is common in these two trees. That is the beauty of a shuffle net, actually.

Now onwards, we will get; in fact, you can very easily convert to a Ben's network, if you are smart enough. I will; I think, you think about it and I will ask how, that will be done. So, this 0; this is 1; so this can go here, and I create a tree here, this one can go here, created a multiplexer tree. This is also a delta network, and how to create, convert it to inverse Ben's? I can visualize that from here. You move it here, and you move this thing here; you get an inverse Ben's. These are like strings. You just take this block; move it here; this one, you move it here; it is an inverse Ben's; nothing else. Sometimes, the things are equivalent, but they do not seem to be equivalent, actually. This is equivalent to inverse Ben's. Now, let us go to generalized definition for what, we call the delta network. How formally, it will be defined?

Student: (( )) sub class of delta network?

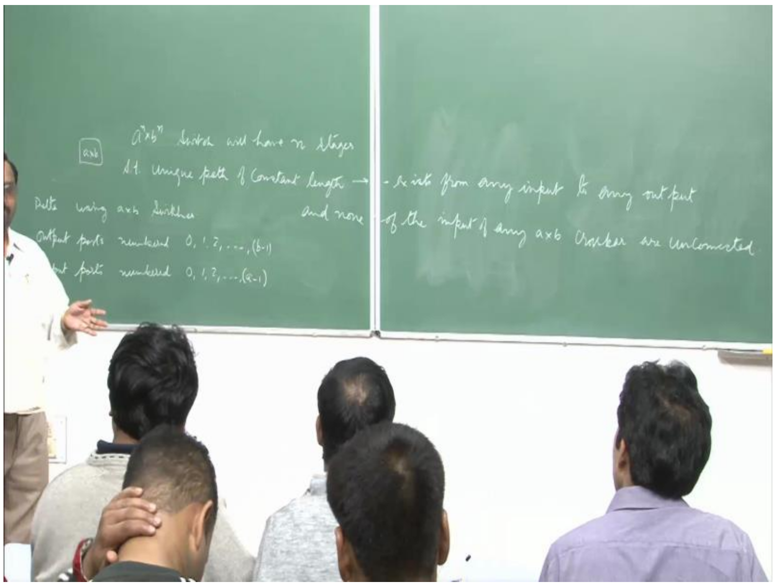
It is a sub class of delta network; this is not shuffling. I will define what the shuffle is, actually. This is not shuffling, but it is a delta network.

Student: The link between first and second stage is similar to second and third stage?

Link between first and second stage is exactly, same; ditto; there is no change.

Ditto; there is a shuffle. This particular thing is a shuffle. Shuffle is one particular example, which can be repeated between multiple stages. You require log 2 n stages, actually. So, this is 8 log 2 n is 3. Log 2 of 8 is 3 in this case, because we are using 2 by 2 switches, but this is not a general definition. General definition is that I will use the switch, which will be of a by b, and the number of inputs to a switch will be a raise power n, number of outputs will be b raise power n; generalized delta network. Now, let us go to this structure, because this is what, I will be using for proving that shuffle net actually, in general; a generalized shuffle net is a delta network. We have to formally, prove it. Actually, it is not evident. Here, it is fine. For 2 by 2, it was very simple, but you cannot keep on drawing for a very large size dimensional switch.

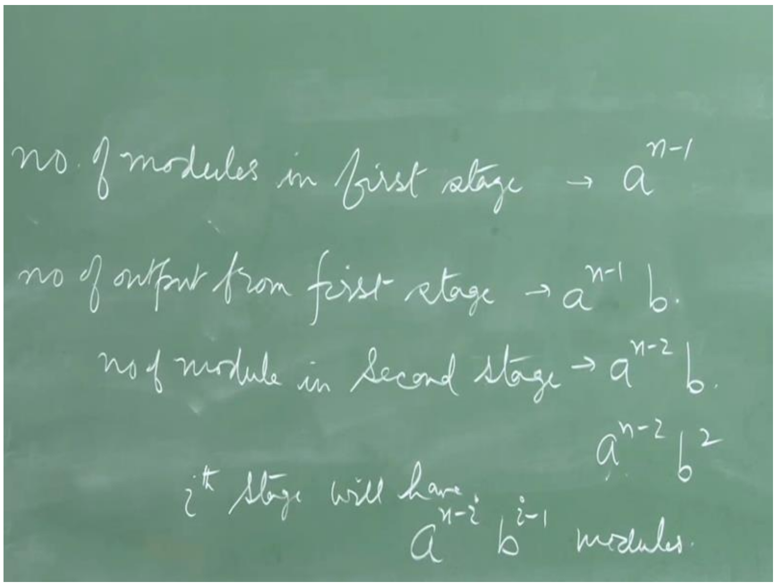
(Refer Slide Time: 19:08)



$\boxed{a \times b}$ $a^n \times b^n$ Switch will have n stages
s.t. Unique path of Constant length → exists from any input to any output
Delta using a×b Switches and none of the input of any a×b Crossbar are Unconnected
Output ports numbered 0, 1, 2, ..., (b-1)
Input ports numbered 0, 1, 2, ..., (a-1)

The switch structure will be something like this. Now, I am coming to that. This will be made using a by b cross bar switches. So, I have to write actually, many statements. These are all, from the part of the definition. So, delta network will be in general, will be filled by a by b switches; a by b is a cross bar switch, technically. It is equivalent to that. It is a strictly, non-blocking switch. Output ports are numbered as 0, 1, 2, and b minus 1; the total be output ports in every switch. Input ports will be numbered as what I should write; a minus 1 perfect. So, a raise power n, by b raise power n switch will contain n stages. So, we were having 2 cube by 2 cube switch; three stages, because of that. Because it was built by 2 by 2 switches. So, it will have n stages, such that unique path; that is the condition for banyan network; of constant length, I will slow this, exist from any input to any output. None of the inputs of any a by b switch or I call it cross bar, are unconnected. This actually, implies that from one stage whatever, is the number of output has to be equal to number of inputs to the next stage. This is obvious, and from here, actually you can and of course we also know that now you will have a raise power n sources; they will be mapped on to b raise power n destinations or the output ports. The stages will be numbered from 1, 2 till n; they are starting from source side.

(Refer Slide Time: 23:35)



no. of modules in first stage → a^{n-1}

no of output from first stage → $a^{n-1} b$.

no of module in Second stage → $a^{n-2} b$.

$a^{n-2} b^{2}$

i^{th} Stage will have $a^{n-i} b^{i-1}$ modules.

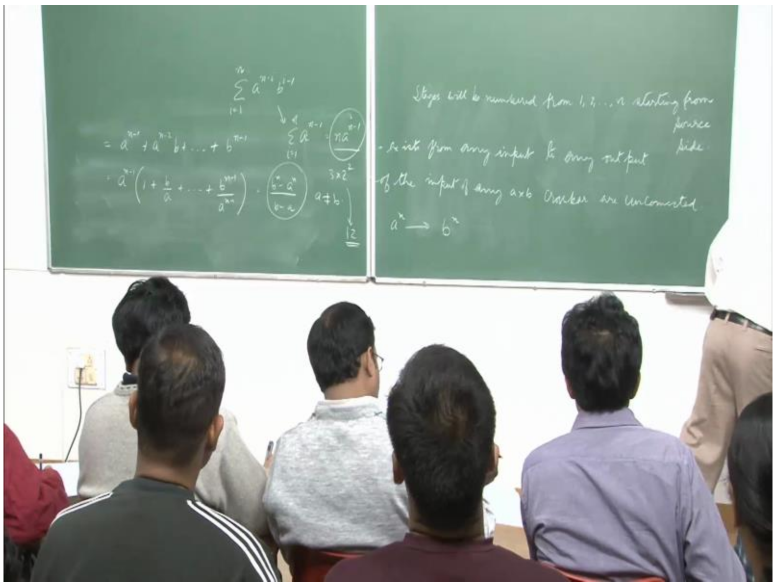
How many modules will be there in the first stage?

Student: (( )).

Yes, perfect.

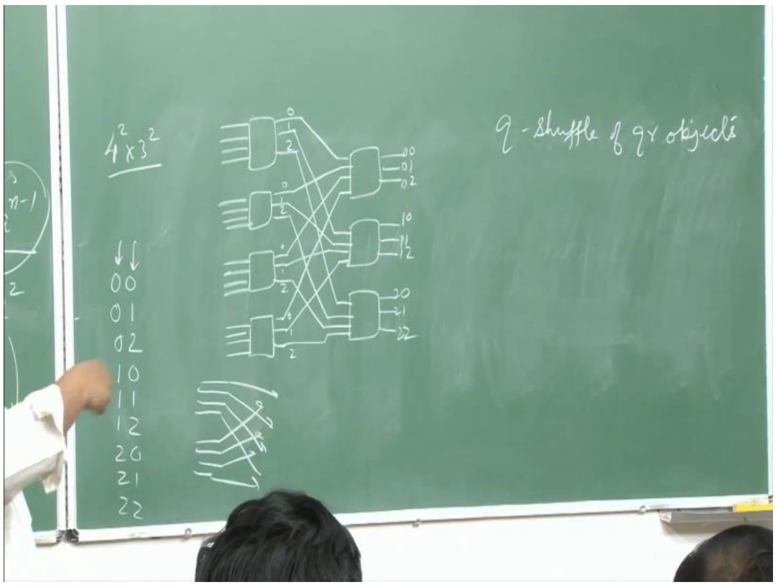
Because a inputs are there to each switch; total number of inputs is a raise power n. So, a raise power n divided by a; it gives a raise power n minus 1. What is the number of outputs, which are coming from first stage? Each switch is the output of b. These are number of modules, has to be a raise power n minus 1 into b. So, this should be equal to number of inputs to the second stage. How many numbers of modules in the second stage? You again, divide by a; number of outputs coming out of second stage, and so on. Once you understand this, i th stage will have a raise power n minus i; these many number of modules. Each module will be of size, a by b. What is the total number of modules? I can do summing up of all these with index i; total number of modules will be this.

(Refer Slide Time: 25:41)


$\sum_{i=1}^{n} a^{n-i} b^{i-1}$
$= a^{n-1} + a^{n-2}b + \ldots + b^{n-1}$
$= a^{n-1}\left(1 + \frac{b}{a} + \ldots + \frac{b^{n-1}}{a^{n-1}}\right) = \frac{b^n - a^n}{b - a}$
$a \neq b$
$\sum_{i=1}^{n} a^{n-1} = na^{n-1}$
3×2^2
12
Stages will be numbered from 1, 2, ... n starting from source side
exists from any input to any output
of the input of any a×b crossbar are unconnected
$a^n \longrightarrow b^n$

I can now sum it up. This will give me, and this of course obviously is when a is not equal to b, because when a is equal to b, this actually, becomes 0 by 0 form. When of course, a is equal to b in that case, this summation will be nothing, but n minus 1, which will be nothing but this obviously, the constant will come out. Either, these many modules will be there, when size is equal or this. So, for our case, when n was 3, a was 2; it is 3 into yes, which was 12 for 8 by 8 delta network. But in general, for a by b, because 2 by 2 was simpler; in general, for a by b, this is what will be the expression. Now, let us look at the rule of how, you are going to construct with a by b; 2 by 2 was fine; a by b also, you will follow the same rule. You will every time, create a demultiplexer tree. Every time, and you will still follow the same rule. So, I can do it for a very small example of 3 by 2 or something, and then take care of that.

(Refer Slide Time: 28:32)

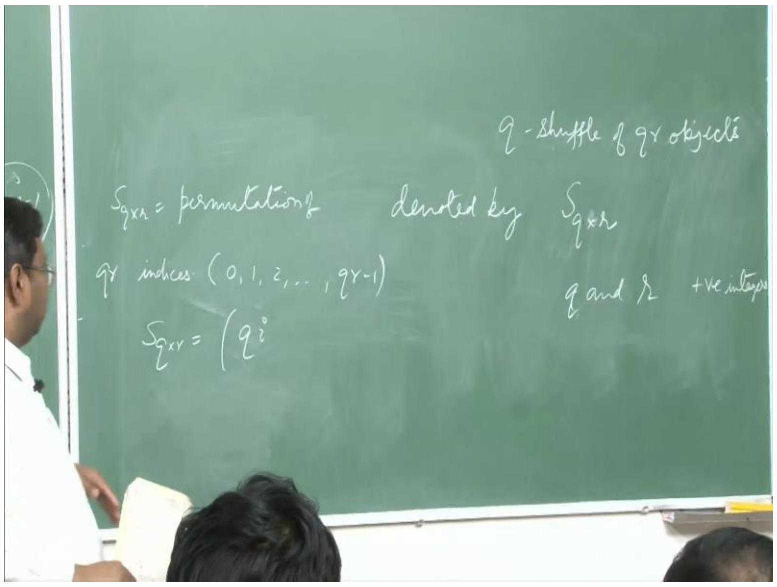


I am going to take a 4 square by 3 square switch. So, it will be built with 4 by 3 switching elements, and it will be having 2 stages. If you want, you can create for anything, but just this is an example of the principle, which are followed for building up the topology. I create, and I require 4 raise power n minus 1; 4 is actually, modules or 4 by 3 in the first stage. Second stage, I will be requiring 4 into 3 is 12; 12 divided by 4; I require 3 more modules, obviously. Let us look at the first input 0, 1, 2. Obviously, there only has to be only 0 input should come. Anyway, I am only creating a demultiplexer tree; I got the demultiplexer tree. Since, there are only 2 stages, actually, it is simpler. Same is here; this 0 has to go here; this 1 has to go here, because all the inputs have to come from the same numbers; got it? It is also a delta network, and the addresses, which you will be keeping is; output addresses will be decide by, the number there. There are only three outputs; remember. So, I require a de array digit. If it was 0 and 1, a binary bit. Now, I require 0, 1, 2; can only be the number, and there are two stages. So, I can use an address like 0, 0; 0, 1; 0, 2; 1, 0; because then I cannot have more than 2; these will be the addresses.

First one will be used for routing in the first stage; second one in the second stage, and you will reach to the destination. These will be the numbers, like 0, 0, 0, 1, 0, 2, 1, 0, 1, 1, 1, 2, 2, 0, 2, 1 and 2, 2. So, you can build up any kind of complex things with this kind of structure. Make sure, you make a de multiplexer tree, every time, for every input, which are going to connect, while maintaining the same condition. Of course, there can be, in general, can be many possible ways, you can create. Now, let us look at shuffle network, which is one of the regular interconnection patterns, which can be periodically, put between all n stages. So, generalized shuffling; we have to identify. So, q shuffle of q r objects; how that will be defined?

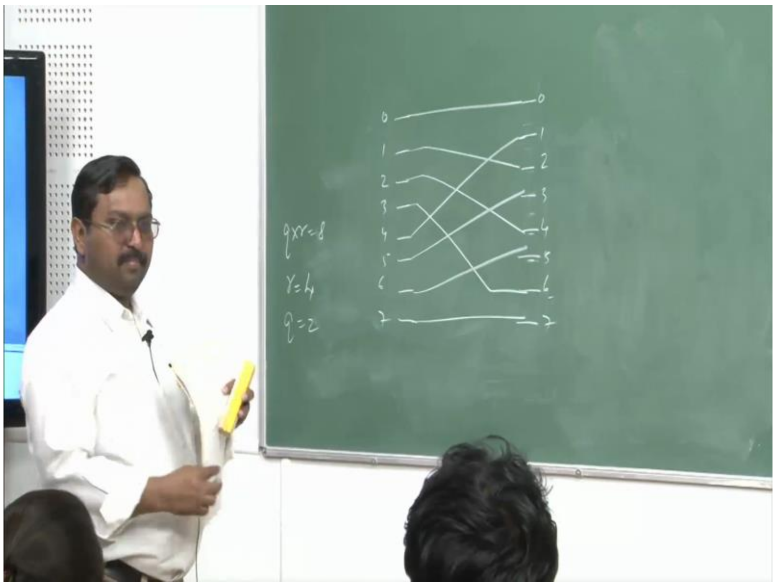
So far, shuffling is technically, I have got a stack; I build up another stack; I can build any number of stacks. Shuffling means, take first from here; second from here; first from here; second from here; I did the shuffling. Usually, when you are playing cards, you are always making to actually; portions and then you are taking one from here, one from here; that is how you do the two shuffles, actually there. In general, there need not be two shuffles. There can be k shuffle. So, you can actually, put all the cards into k separate packets take one from here; one from me each packet, every time in Round Robin fashion, and you create a k shuffle in that case.

(Refer Slide Time: 33:28)



We want to actually, know them give a generalized definition. This is required for me for doing the address mapping, because the proof is by induction. So, q shuffle of q r objects will be defined as this will be denoted by where q and r are positive integers. This is nothing but I am generating a permutation of indices, q r indices, which are 0, 1, 2, q r minus 1, and this will be defined by s of q r q into i. So, i is basically, the input side. So, there is an interconnection pattern from input to output.

(Refer Slide Time: 34:52)



You have numbers like this; 0, 1, 2, 3 till, q r minus 1. I am now, trying to generate a map to this. So, this interconnection pattern is what to shuffle. For if you take this i th pattern, which is sitting here, this i is mapped to work; that is what we are trying to figure out through a formula. So, it will be always q into I; it is a q shuffle. If there are two shuffles, I multiply by 2 and then do it, actually. Two shuffles means, there are two cards. So, if I take the first one, the next one has to go to the; you have to leave something in between; that is why q into i will be there. So, 0 will be connected to 0; 1 will be connected to 2, not to 1; 1 will be connected to 4, and so on, but then the number will saturate. Then, you have to do a modular operation. This is actually, for that.

Student: (( )) here, I am assuming that q shuffles of i (( ))?

q shuffling is what I am implementing now, of q r objects.

Student: But if I have a stamp of q r the maximum possible.

q r minus 1 is not the stack; stack is of q r objects. They are numbered from 0 to q r minus 1. This 0 also, you have to count. Actual number of object is how much? 1, 2, 3, q r; they are numbered in this fashion. They are actually, numbered in this fashion, because in computing, we always, start 0, is always included as the first object. So, if you have n address line, they will be numbered always from 0, 1, 2, 3 till, n minus 1.

Student: The number of shuffling can be greater than q to i.

I am not able to understand.

Student: (( )).

i, q into i can be greater than q r.

Student: Yes.

Yes, that is fine; there is a modular operation. I am going to give an example with this. There are two expressions, which actually, can be used. One of them is this, actually, and of course, the equivalent expression, which also can be used, is this. See, these both are equivalent expressions. For the 8 by 8, let me build up a two shuffles with this. So, 1, 2, 3, 4, 5, 6, 7, 8 numbered from 0, 1, 2, 3, 4, 5, 6, 7. We will do with the first formula first. So, q into r is 8. What is r; is 4, q is 2. So, I have correspondingly, the numbers here; 0, 1, 2, 3, 4, 5, 6, 7. Let me just do it parallel, actually. So, i is 0; 0 into q is 0; i divided by r; I have to find out an integer, which is lower than i by r; highest integer, which is lower than i by r, actually. That is what it means; i by that is a lower floor; that is the floor function; this, which I have wrong.

This actually, means the integer, which is lower; highest integer, which is lower than i by r. There is similarly, a sealing function, which is the lowest integer, which is higher than the number. So, 0 into 2 is 0; 0 by r is 0. So, highest integer lower than r equal to 0 is 0, actually, and modular q r; you will get a 0 to 0, has to be connected to 0. When it is 1, 1 into q, q is 2 here. You will get 2 and 1 divided by r, r is 4, actually; remember. So, 1 by 4 is a number, which is a fraction. So, integer, highest integer lower than that is 0, actually. So, 1 into 2 will give me 2. What 2 will give? 2 into 2 is 4; 2 by 4 neglect the other part; that four thing. So, this will be connected to 4; 3 will be connected to 6; when the 4 will come; 4 into 2 is 8, which is fine; 8 plus 4 by r is? 4 by 4 is 1. So, that number will come out to be 1, when you take the floor function; 4 by 4 is 1. So, that will be 1. So, 8 plus 1 is 9; 9 modulo of 8, is how much?

Student: 1.

So, I have 4 will be connected to 1. Try for the fifth, similarly. So, 5 into 2 is 10; 10 plus 5 by 4; floor function will give you 1. So, 10 plus 1 is 11; 11 modulo 8 is 3. Similarly, for 6; this was what exactly, the shuffle net, which I had drawn, but binary was easier to remember, but in general, you can do for q shuffling.

Now, use with the second formula; this will also give the same result, actually, not an issue. So, 0 will give you 0; q into i modulo q r, modulo 7 now; 1 will give you 2; 2 will give you 4; 3 into 2 will give you 6; 6 is actually, modulo, because 7, I am taking; 4 will give you 4 into 2 is 8; modulo 7 will give you 1; right connection. 5 into 2 is 10; modulo 7 will give you 3; 6 into 2 is 12; modulo 7 will give you 5, which is perfect, and when i is equal to q r minus 1, that time you have to use this i; the lower part. It also gives. So, that is a basically, thing, which we will be using now, for doing transformation of address lines, actually, ultimately.

Student: Sir, what is the advantage of (( ))?

Shuffle will give you the same connection pattern between multiple stages. Why you are implementing in silicon; the same mask can be simply; repeated three times; or it is a larger switch; same thing, you repeat multiple times. If you do not do that your cost of fabrication actually, will go up. You require n different kinds of interconnections, and n different kind of masks. You cannot simply, repeat. Well, that is I think, is the issue. If it is an optical thing, and I am making a kind of a planer array of transmitter, then I put another one on top of it. So, I would require somewhere, between holographic interconnect. Every time you have to try a different holographic interconnect between stages. If it is the same thing repeated, you only make it once; make multiple copies, push it. Mass production reduces the cost. Yes, that is all optical implementations will be something, like this. You will have transmitters here. Each one is a processor; it will connect two inputs.

On top of this, you will actually have a slab, which is a holographic thing; computational holography actually, is used. You create a phase grading, technically, by depending on what interconnection pattern is required, but remember; it is a 2-dimensional thing; it is not one dimensional. Here, I am actually, drawing 1-dimensional. So, 2-dimensional also, can be done in the same fashion, but I am not moving to 2-dimensional thing here. But that is actually, altogether, a different domain. That is I think, some other course probably, this can be discussed, but we do not have a course on optical computing. So, you do not get opportunity. Any course on optical computing will talk about this; the structures. May be, some other semester, I will try. I think, we will close, here. Shuffle now we have understood. I will take exactly, this thing and we will start a formal proof that shuffle net actually, is a delta network. A generalized shuffle net, which will follow this interconnection pattern from the output of i th stage to i plus first stage; mapping is done as per this.