

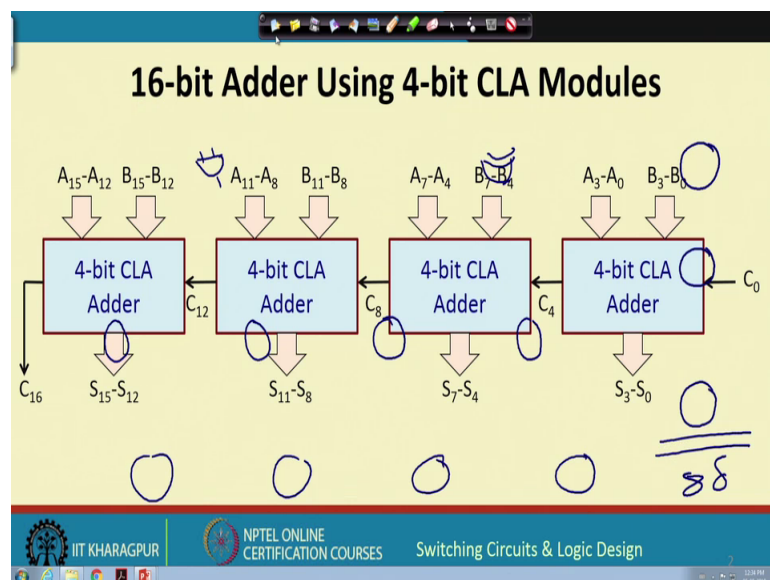
Switching Circuits and Logic Design
Prof. Indranil Senguta
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture - 23
Design of Adders (Part - III)

So, in the last couple of lecture if you recall, we had talked about the design of adders, we talked about half adder, full adder. Then we saw how we can combine them to create parallel adders, two kinds of parallel adders if seen so far ripple carry adder where the delay increases with the number of bits and carry look ahead adder where the delay is constant. But the hardware complexity number of gates, size of gates they increased very rapidly with the number of bits.

So, in practice we need some kinds of trade off we cannot design a 64 bit carry look ahead adder because the gates would become too complex. But we cannot also design a 64 bit ripple carry adder, because its delay will have to ripple through all these stages. So we have to have some kind of compromise, let us look into it this is our third part of adder design.

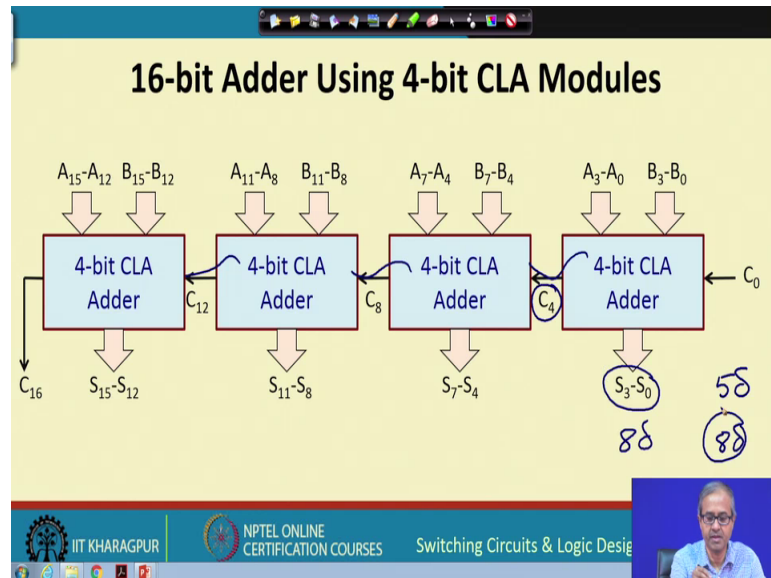
(Refer Slide Time: 01:22)



So, here I shall not be going into too much detail, just I will try to give you the basic idea in the concept. See we are trying to design a 16 bit adder let us say and suppose we have already designed 4 bit carry look ahead adder modules as we had discussed in the

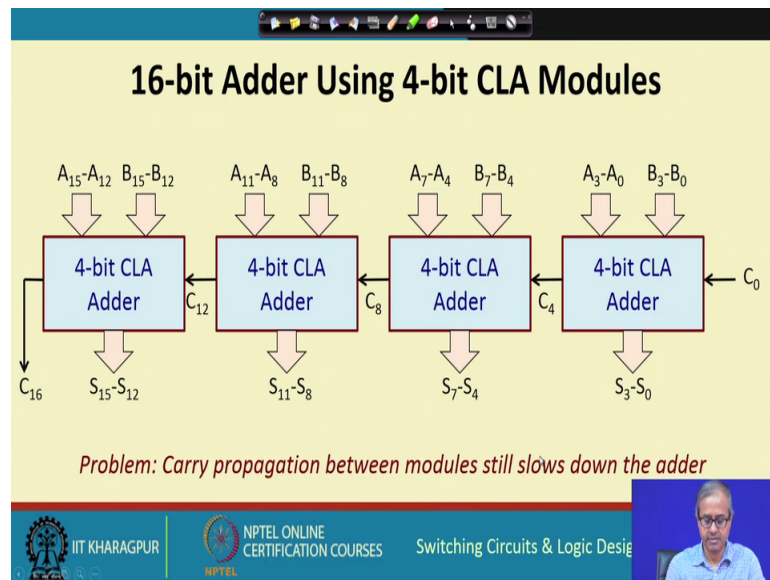
last lecture. Suppose we have such 4 bit carry look ahead adder modules available with us so we can add two 4 bit numbers and generate of 4 bit sum. Now, one thing you can always do we can cascade this 4 bit look ahead adders like this you see this 4 bit carry look ahead adder if you recall earlier that we need 5 delta time to generate the carry and 8 delta time to generate the sum.

(Refer Slide Time: 02:18)



So, this sum will be generated after time 8 delta and this carry is being generated here so again after some delay this will be generated. So you can calculate the total time taken, so there will be a rippling of carry still going on across the 4 bit carry look ahead adders stages, but within each of these blocks the time is constant this is basic idea.

(Refer Slide Time: 03:01)



So, carry propagation between modules are still there.

(Refer Slide Time: 03:05)

- Solution:
 - Use a second level of carry look-ahead mechanism to generate the input carries to the CLA blocks in parallel.
 - The second level of CLA generates C_4 , C_8 , C_{12} and C_{16} in parallel with two gate delays (2δ).
 - For larger values of n , more CLA levels can be added.
- Delay calculation of a 16-bit adder:
 - a) For original single-level CLA: 14δ
 - b) For modified two-level CLA: 10δ

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | Switching Circuits & Logic Design

Let us do a simple calculation. So, one thing you can do, we can use a second level of carry look ahead mechanism like in the previous diagram we said that we have 4 bit carry look ahead adder modules where there will be carry propagation between stages or between these modules. So what we can do we can have a higher level carry look ahead adder kind of a scheme again where all this carries can be generated in parallel.

So, you do not need to have this carry propagation again, same concept which you had for carry look ahead adder where using it in the higher level also to generate C_4 , C_8 , C_{12} and C_{16} like if you just go to the previous slide this C_4 , C_8 , C_{12} and C_{16} . If I have a higher level carry look ahead module which can generate these in parallel then this rippling affect will no longer be required.

(Refer Slide Time: 04:22)

• Solution:

- Use a second level of carry look-ahead mechanism to generate the input carries to the CLA blocks in parallel.
- The second level of CLA generates C_4 , C_8 , C_{12} and C_{16} in parallel with two gate delays (2δ).
- For larger values of n , more CLA levels can be added.

• Delay calculation of a 16-bit adder:

- For original single-level CLA: 14δ
- For modified two-level CLA: 10δ

Handwritten annotations on the slide include a wavy line under 2δ , a circle around 8δ next to the original CLA delay, and a circle around 5δ next to the modified CLA delay.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | Switching Circuits & Logic Design

So, a simple delay calculation I am showing here. Now this carry look ahead adder as you know that requires only two stages and or it will require to get delay twice delta and for a 16 bit adder 4 4 4 4 for the original single level carry look ahead if you do it, it would be fourteen delta or for modified two level it will be reduced 10 delta. So, here I am not going in to the detail of this, but I am just telling you; that if you have a modified two level carry look ahead scheme and the delay can be reduced; so the basic idea I give you.

(Refer Slide Time: 05:05)

Delay of a k-bit Adder

n	T_{CLA}	T_{RCA}
✓ 4	8δ	9δ
✓ 16	10δ	33δ
✓ 32	12δ	65δ
✓ 64	12δ	129δ
✓ 128	14δ	257δ
✓ 256	14δ	513δ

$T_{CLA} = (6 + 2\lceil \log_4 n \rceil) \delta$
 $T_{RCA} = (2n + 1) \delta$

4
16
64
256

256

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | Switching Circuits & Logic Design

So, this table gives a quick comparison that for certain values of n typical value 4 16 up to 256 I am showing, if we have a normal ripple carry adder the total delay will go if you recall total is 2 n plus 1 delta.

So, it will go up like this 933, 65 up to 513, but for carry look ahead adder this 4 4 4 in term so it will go like that you build up 4 bit adder; 4 such you get 16, 4 such you get 64, 4 such you get 256 and so on. So if you with a calculation will see that the delay are much less between 16 and 64 they do not change. So, you will see so above 16 if you go it will become 12.

(Refer Slide Time: 06:05)

Delay of a k-bit Adder

n	T_{CLA}	T_{RCA}
4	8 δ	9 δ
16	10 δ	33 δ
32	12 δ	65 δ
64	12 δ	129 δ
128	14 δ	257 δ
256	14 δ	513 δ

$$T_{CLA} = (6 + 2 \lfloor \log_4 n \rfloor) \delta$$

$$T_{RCA} = (2n + 1) \delta$$

$4 \rightarrow 16 \rightarrow 64 \rightarrow 256$
 $8\delta \quad 10\delta \quad 12\delta \quad 14\delta$

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Switching Circuits & Logic Design

Up to 64 and above 64 if you will go it will be 14 up to 250 so it will increase by 2 2 2 each. So for every part of 2 like you go from 4 to 16 to 64 to 256 your delay increases by 2 2 each so for 4 it is 8 delta, for 16 it is 10 delta, 64 it is 12 delta and 256 it is 14 delta. And if you have any other value between like 128 that will also be this 14 delta as you can see so 32 for example, 32 is also 12 delta.

So, you see the carry look ahead adder delay is much less much much less as compared to ripple carry adder this is one point.

(Refer Slide Time: 06:58)

Carry Select Adder

- Basically consists of two parallel adders (say, ripple-carry adder) and a multiplexer.
- For two given numbers *A* and *B*, we carry out addition twice:
 - With carry-in as 0 ✓
 - With carry-in as 1 ✓
- Once the correct carry-in is known, the correct sum is selected by a multiplexer.

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Switching Circuits & Logic Design

Now, let us look at some other kinds of adder of course, we shall not be going in to too much detail on these, these are the slightly advanced kind of an adder which can generate the sum with the average delay which is much smaller as compared to the convention adders. Let us see the basic concept the first kind of adder we talk about is called a carry select adder. Well, carry select adder the idea is very simple let us talk about an addition module let us say we have several modules of addition which are cascaded like we have talked about several 4 bit adders connected in cascade.

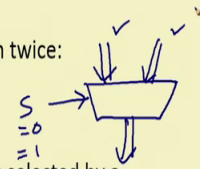
So, for one stage of a 4 bit adder the carries coming from the previous stage and the carry out will be going to the next stage, let us consider one such stage. Now in that stage the input carry can be either equal to 0 it can also be equal to 1, so we do not know what the carry will be at time t equal to 0 will have to wait for certain time for the carry to get generated only then we will be knowing what the carry is. Now the idea is that we have two concurrent versions of the at the that is two hardware we were using two adders one adder is doing addition in advance assuming carry is 0, another adder is doing addition in advance assuming carry is 1, but do not know have the actual carries ok.

So, we carry out addition twice; once with carry as 0, once with carry at 1. Later on when the previous stage has generated the actual carry because you see already these two additions had been complete they have added and we waiting. Now we can have a multiplexer this multiplexer can be selected by this actual carry either this will get selected or this will get selected the correct sum is selected by a multiplexer right.

(Refer Slide Time: 09:25)

Carry Select Adder

- Basically consists of two parallel adders (say, ripple-carry adder) and a multiplexer.
- For two given numbers A and B , we carry out addition twice:
 - With carry-in as 0
 - With carry-in as 1
- Once the correct carry-in is known, the correct sum is selected by a multiplexer.



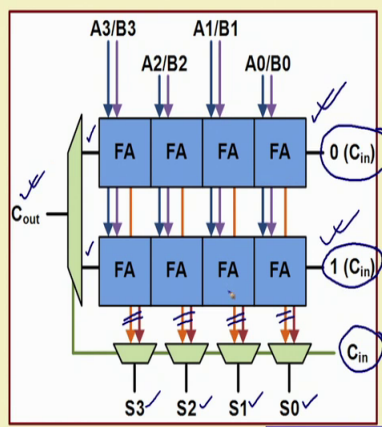
IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | Switching Circuits & Logic Design

So, a multiplexer if you recall multiplex is any circuit where there are two inputs, there is one output, there is one select line. It will either select this or select this depending on the value of the select line. If S is 0 this will be selected, if S is 1 this will be selected.

(Refer Slide Time: 09:51)

Basic building block of a carry-select adder, with block size of 4.

- For a multi-bit adder, the number of bits in each carry select block can be either uniform or variable.



IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | Switching Circuits & Logic Design

I mean showing you how a carry select adder looks like considered a 4 bit adder model that is like what we saw here.

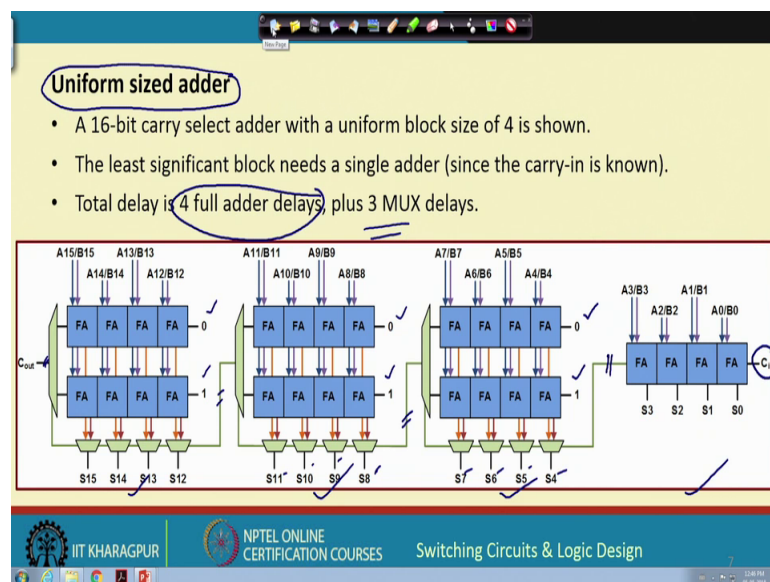
So, now we have two parallel 4 bit adders one is here, one is here you see one of them is adding assuming carry in 0 other is adding assuming carry in is 1. Later on when the

carry in actually becomes available from the previous stage, but this full adders have already completed their addition, so their sums are already available here their sums are already available here.

So, depending on the actual value of carry in you either select the top most adder or the bottom most adder depending on the actual value of carrying on either this or this will get selected as the final sum. Similarly the carry you select either this carry or this carry depending on C in, so you will be getting the final carry out. So the idea is this way investing on hardware we are using double the amount of hardware and we are doing some speculation we are doing addition both for carries 0 also for carry 1. And we are being ready with the two sums and once when the carries available here selecting one of the two sums using a multiplexer.

So, we are avoiding the addition delay, only the multiplex delay which is much smaller that will be n got ok. This is the item so this is an example where we have 4 bit modules, now the second thing is that if we have a multi such stage addition the number of full adders in every stage can be either uniform or variable let us see.

(Refer Slide Time: 11:52)

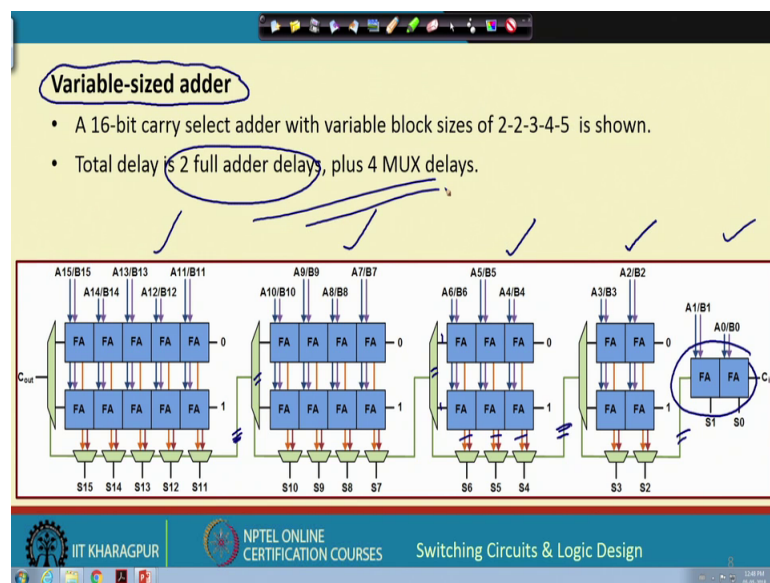


This is an example of a uniform size adder where I am saying this is a 16 bit adder, this is stage first stage, second stage, third stage and fourth stage 4 bit 4 bit 4 bit adder for the first stage there is no confusion because carry in is directly available.

So, you do not need two copies of adder only one adder, but for the other stages there are two adders and they are adding with carry 0 and carry 1 in parallel. So once the carry gets available say for a example the first stage computes this carry this carry will be selecting these S 4 5 6 and 7 also it will be selecting the correct carry here and this correct carry will now you selecting S 8, S 9, S 10, S 11 it will be selecting here and again 12 13 14 15 and C out.

So, you see here also there is a rippling effect, but this rippling effect is much faster because this is only through a multiplexer, not through full adders. So here the total delay of the 16 bit adder will now will be only 4 full adder delays, because each of these stages are taking 4 bit full adder delay assuming it is a rippling carrier. Let say, assuming ripple carry adder it will 4 full adder delays, if it is a carry look ahead adder it will even faster plus this 3 multiplexor delays right.

(Refer Slide Time: 13:44)



Now you can also improve upon this using a very intelligent way by using variable sized adders; variable sized means you see we use 2 bit adder, 3 bit adder, 4 bit adder, 5 bit adder because the input bits you do not divide equally this is also 2.

So, it is A 0 and B 0, B 1 these also two A 2, A 3, A 4, A 5, A 6 this is 4 bit this is 5 bit, but ultimately you are getting a 6 bit adder stage. See here the advantages that see when you the first stage has completed two bit addition. Let us say this two bit addition is also completed so after the one bit multiplexer delay will be getting here, so this two full

adder delay plus one. Now this adder by this time has finished addition because assuming one full adder delay and one mux delay are equal, here the delay was two full adder delay plus one mux delay by that time this carry is generated this full adder is also ready with the sum and the carry.

So, you can directly applied here and by the time it is ready plus one mux delay this 4 bit adder is also ready because 3 plus 1 is 4. Similarly 4 plus one mux delay when that comes here this is also ready, so you do not have to wait any more for the later stages because when the carry comes the adders are already ready with their sums. So you need only two full adder delays corresponding to the first stage plus the mux delays.

So, the delay gets reduced right I am not going into too much detail about this as I told you just trying to give an idea. This is sometime called 2 2 3 4 5 configuration 2 2 3 4 and 5, this is how the number of bits are divided.

(Refer Slide Time: 15:57)

Carry Save Adder

- Here we add three operands (say, X, Y and Z) together.
- For adding multiple numbers, we have to construct a tree of carry save adders.
 - Used in combinational multiplier design.
- Each carry save adder is simply an independent full adder without carry propagation.
- A parallel adder is required only at the last stage.

The slide contains two hand-drawn diagrams of a carry save adder. Each diagram shows a rectangular box with three arrows pointing down into the top of the box, representing three inputs. From the bottom of the box, two arrows point downwards, representing the sum and carry outputs.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | Switching Circuits & Logic Design

The last kind of adder that I talk about here is something called a carry save adder; now it is a little beyond this scope of this a course to talk about the application. It is very useful for the design of design multipliers carry save adders, but we shall not discussing this here we shall be telling you what the basic concept of carry save adder is and how you can design it. See a normal adder adds two numbers right. You have two input numbers you generate some and carry, but in a carry save adder you are adding 3 numbers X Y and Z.

So, you are adding 3 numbers and you generating sum and carry these are not 1 bit number this can be n bit numbers. Now you shall see how it works because we are seeing that there are 3 operands, but the carry save adder is nothing but an independent full adder without carry propagation, there is no carry propagation. Here let us try to understand this; what is meant by no carry propagation and that parallel adder may be required at the last stage.

(Refer Slide Time: 17:21)

• An illustrative example:

X: 10011	X: 10011	$\begin{array}{r} X: 10011 \\ Y: +11001 \\ Z: +01011 \\ \hline S: 00001 \\ C: 11011 \\ \hline \text{Sum: } 110111 \end{array}$
Y: +11001	Y: +11001	
Z: +01011	Z: +01011	
C: 10000	S: 00001	
S: 00001	C: 11011	

A set of full adders generate carry and sum bits in parallel
The sum and carry vectors are added later (with proper shifting)

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | Switching Circuits & Logic Design | 10

Let us take an example consider an addition; suppose I have 3 members X Y Z which you have adding because as I said for a carry save adder we talk about adding 3 numbers so we are showing the carry and sum separately.

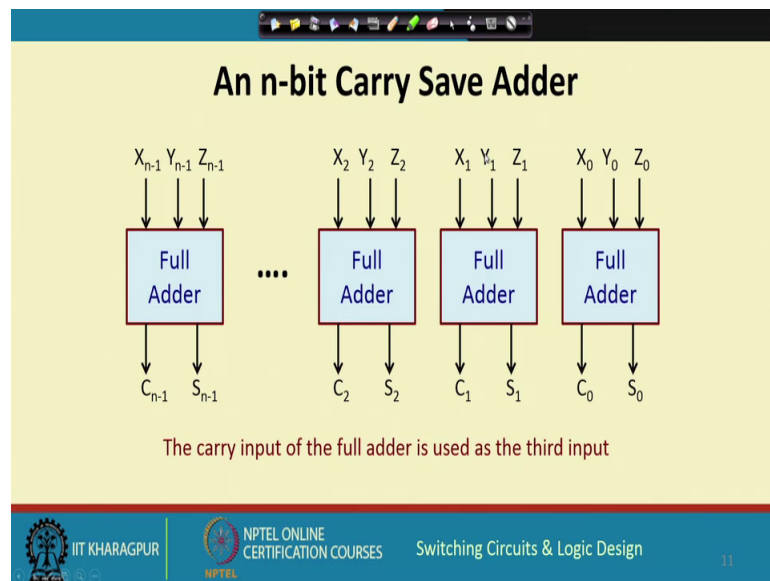
So, when you add X Y Z 1 1 1 it is 1 with the carry of 1 we are not just adding with carry we are independently adding the different stages 1 0 0 is 0 with the carry of 1, 0 0 0 is 0 with the carry of 0, 0 1 1 is 0 with the carry of 1 and 1 1 0 is 0 with the carry of 1. Now what I do you see you have accommodated the sum 0 0 0 0 1 parallelly without carry propagation and also carry you 1 1 0 0 1 1 the carry out from different stages we shift the carry by 1 position and add with S whatever we get is the actual sum of X Y and Z.

So, the idea is that we are not doing the conventional kind of addition instead of using the carries from one stage to the other we are actually doing the addition like this we are doing parallel addition without carry propagation, we are generating this sum separately generating the carry separately. A set of full adders independent full adders; that means,

each full adder will be like this there are 3 inputs and just 2 outputs sum and carry no carry propagation from one stage to the next

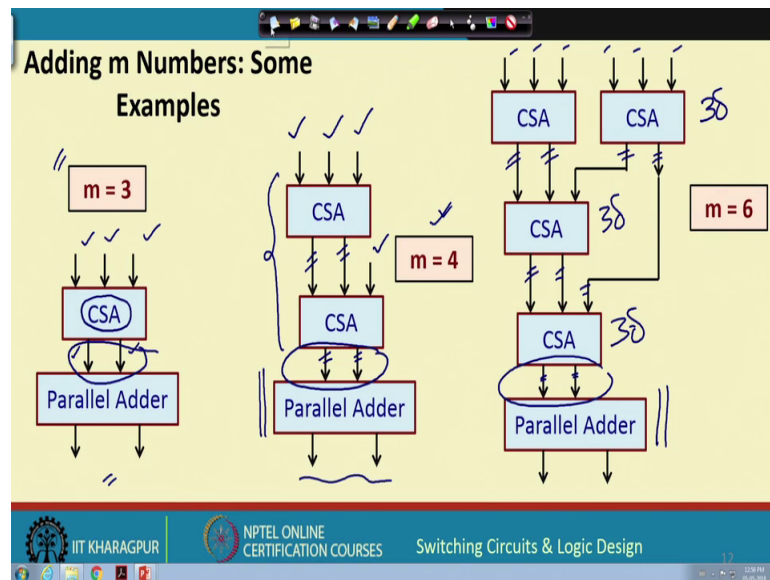
So, this can be then in constant time and after this constant time the last stage you need an adder to add S and C after 1 shift so you need an adder of course at the end, but not before that, but why do you need this what is the advantage.

(Refer Slide Time: 19:37)



The advantage will be if you have many numbers to add like n bit carry save adder just look like looks like this set of independent n full adders the each full adders takes 3 bits of the inputs X Y and Z, and generates carry and sums separately right.

(Refer Slide Time: 20:01)



So, let us take some examples suppose I have 3 numbers just like I show to previous just in the previous you see there are 3 numbers X Y Z in the output I generate a sum and a carry so it is like this. There are 3 numbers m equal to 3, I use 1 carry save adder the 3 numbers are applied in input.

So, sum and carry are generated so as I said at the last stage after shifting I need one parallel adder I get the final sum. Now for m equal to 3 we do not see advantage unnecessary we are doing it complicated because ultimately we need an adder at the end, the advantage will be it would be apparent if you use some higher value of m . Suppose you were wanting to add 4 numbers first 3 numbers are applied here the fourth number is applied here, you see we use a carry save adder in the first stage you generate a sum and a carry independent this sum carry and the fourth number if you need it to another carry save adder, it will generate a sum and carry.

And finally, you applied to a parallel adder with carry propagation to generate the final result. So these two CSA and CSA will be recurring constant time without carry propagation only the last stage will be using carry propagation. Let us say m equal to 6 there are 6 numbers I want to add, see first stage I use 2 carry save adder so 1 2 3 4 5 6 I am adding so 1 sum and 1 carry 1 sum and 1 carry.

So, I can use another carry save adder to add 2 of these and 1 of these I have finally, 1 sum and 1 carry and also this sixth one, you see ultimately you will have to bring it down

to 2 that is why we needed a tree of such carry save adders, you see here this parallel adder will take only 2 inputs at the last stage, so you have 6, 6 down to 4 down to 3 down to 1 will have a chain of such carry save adders. The advantage is that each carry save adder will be just independent full adders they will have a delay of 3Δ each 3Δ 3Δ 3Δ only at the end there is a rippling of the parallel adder, but the advantage is that you are able to add 6 numbers together, 9Δ plus the delay of 1 parallel adder this is the main advantage here. But, as I mentioned these are some kind of slightly advanced kind of adders, and we do not need to go too much detailed about this just to give you an idea that this kind of carry select and carry save adders are also possible, and they have very interesting applications in some other areas may be.

So, with this we come to the end of this lecture you see over the past 3 lectures we basically talked about various kind of adders. Now in the next few lecture we shall be looking at how we can design other basic combinational circuit modules and how they can be used in certain applications. Logic design is of course one application we want to realize or implement some function, but there can be other kinds of applications as well. This we shall be discussing in our next lectures.

Thank you.