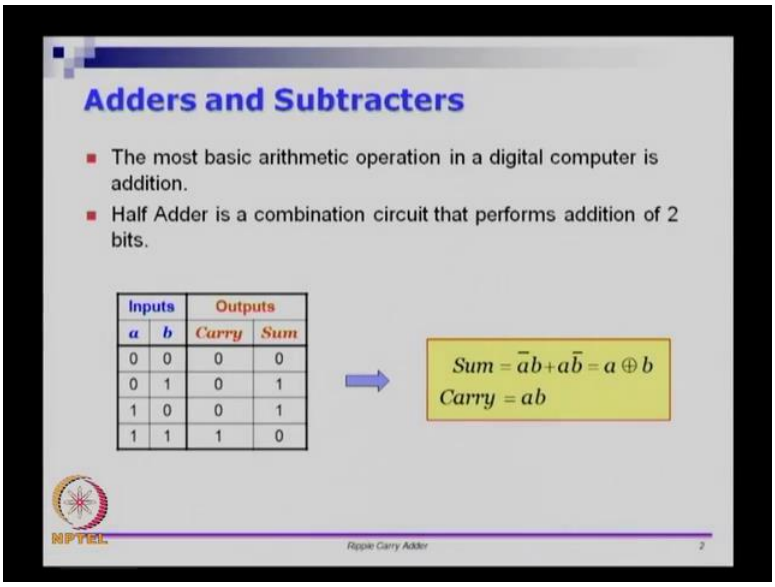
Digital Circuits and Systems
Prof. Shankar Balachandran
Department of Electrical Engineering
Indian Institute of Technology, Bombay
And
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Module – 50
Ripple Carry Adder

Welcome to module 50, so we have hit half a century now, in this module we are look at Ripple Carry Adder. So, in the last module I said, you will do this little thing that for 2's complement adder I said if there is an unsigned adder, I will be able to make a 2's complement adder out of it by doing some extra logic. So, I am going do how to design the ripple carry adder now or an unsigned adder now. So, this discussion is only about unsigned calculation, it is not about signed calculations.

(Refer Slide Time: 00:52)

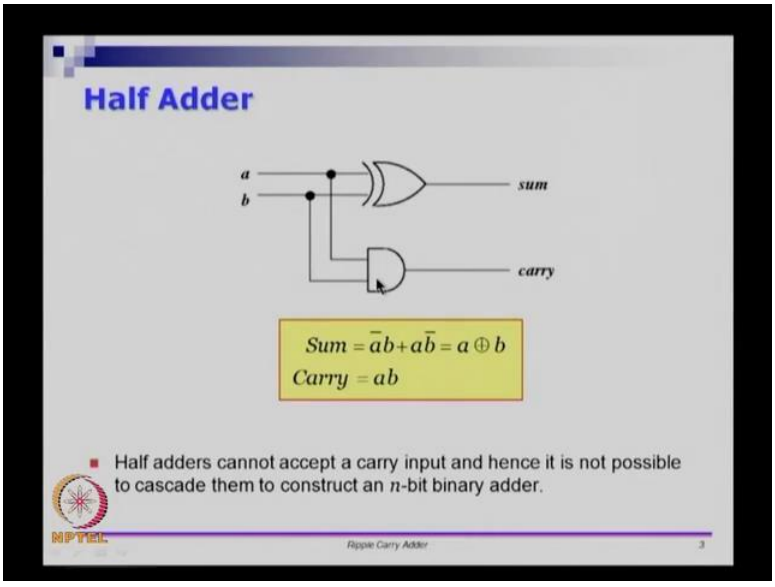


Let us look at adders and subtractors, these are the most basic operations in any digital computer. And in the homework and earlier also, I already talked about what a half adder is and what a full adder is. A half adder is a circuit which takes 2 input bits a and b, and produces 2 output bits carry and sum. So, the carry and sum if I take that as carry comma sum as a vector and if I interpret that in the unsigned form, that will give me the result of adding a to b.

So, 1 plus 1 is number 2 and number 2 gives me carry equal to 1 and sum equals to 0 and

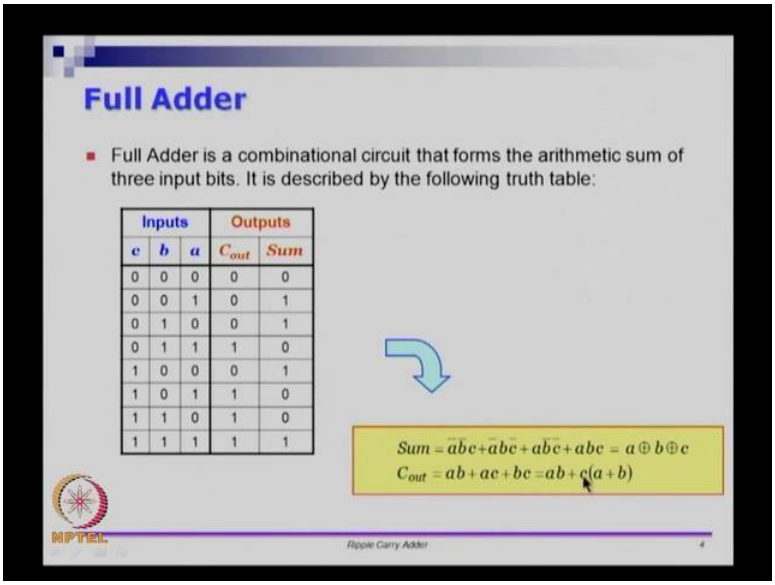
if you look at the truth table here. So, with respect to a and b this truth table carry column is only a and b and the sum column is a XOR b. So, sum is a XOR b, carry is a b, I believe that you have already done this in your Verilog assignment from a few weeks ago.

(Refer Slide Time: 01:49)



So, half adder is a fairly simple circuit, you have a XOR gate and you have a AND gate and remember this can add only two bits.

(Refer Slide Time: 02:00)

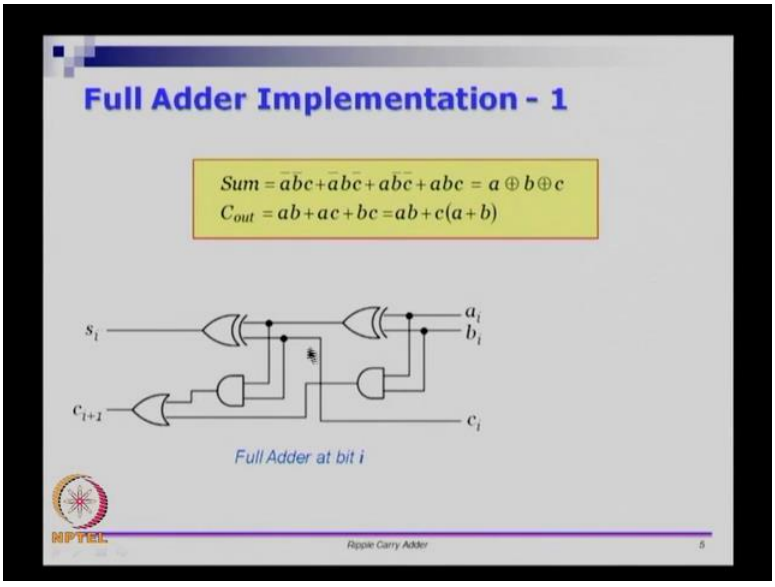


A full adder is a circuit which can take 3 bits, you can take 3 bits a, b, c and it can give you 2 numbers C out and sum, since these are 3 bits a, b and c there is only 2 bits sum

and 0s. So, these are 3 numbers at most it can be 3, so 3 is 1 1 we can represent that using only 2 bits. So, this is carry out and this is sum, so 0 plus 0 plus 0 is 0, so 0 is 0 0 similarly 1 plus 0 plus 1 is number 2. So, that 1 0 1 1 1, 1 plus 1 plus 1 is number 3 that is 1 1 and so on.

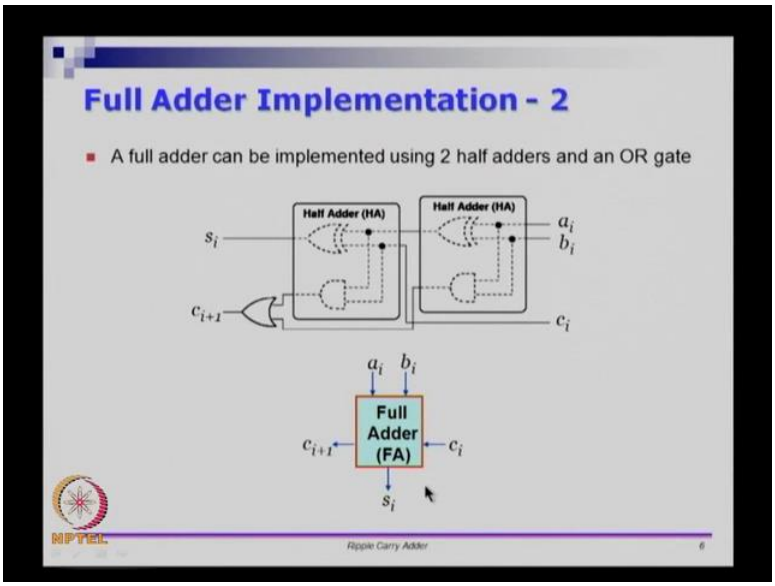
If we go and do the Boolean minimization for that, you will see that sum is a XOR b XOR c and C out is a b plus a c plus b c. So, C out is actually the majority of the three and sum is the XOR of the three, we have already talked about this at length at several locations before. We did not call it full adder then but now we know that it is a full adder. A full adder has sum which is XOR of all the three and carry out is majority of the 3 bits. So, we can do a slight simplification here a b plus a c plus b c is the same as a b plus c times a plus b or c and a plus b.

(Refer Slide Time: 03:19)



So, full adder is one of the implementation is directly taking this and implementing it like before. So, you take a and b, you XOR it and XOR that with c. So, let us coming from here, take a XOR b XOR c that is sum s and this is the carry. So, we have sum and carry as it was before and if you want to go and looked at this full adder, the full adder I use the suffix i, this a i, b i, c i which results in s i and c i plus 1. So, this s i and c i plus 1 are supposed to be in the intermediate stages. So, we have stages, at the i'th stage we take the inputs a i and b I, take the carry i and the results should be sum i and carry i plus 1.

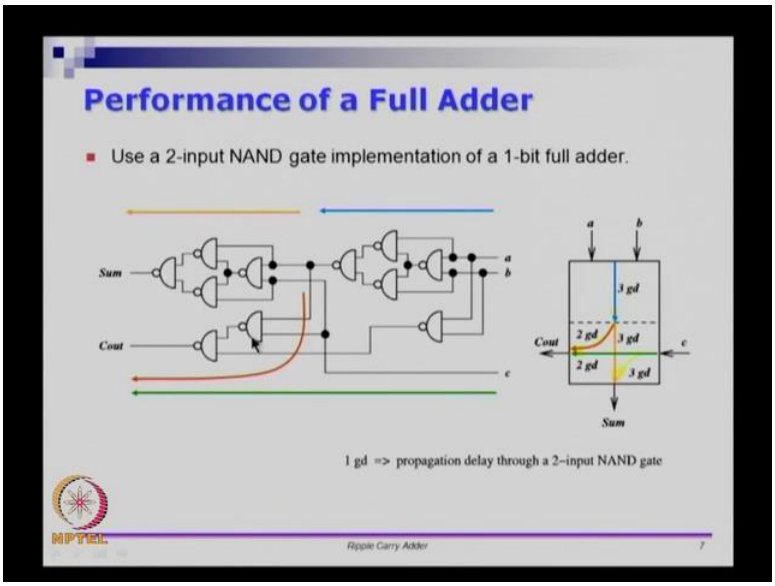
(Refer Slide Time: 04:14)



So, there is another implementation that we can do for a full adder and that is actually using 2 half adders, you can take 2 half adders and connect them using this form. You take the sum from one of the ones and you take c I, give it as the inputs to... So, remember half adder can take only 2 inputs. So, this half adder is taking a i and b i and this half adder is taking the sum bit of a i and b i and c i. This is going as the inputs and the carry that is coming out of here and the carry that is coming out of here is all together that gives you c i plus 1.

So, this is another way to do a full adder, using 2 half adders you can do a full adder. The key thing is that as I mentioned before, at stage i there would be a carry that is coming in from the right side, there are input bits that are coming in from the current stage, it may give you a sum and a carry out. So, we have already discussed this, when we looked at addition of numbers and so on.

(Refer Slide Time: 05:20)



Let us look at how to analyze the performance of a full adder, so this is the circuit that actually has a full 2 input NAND implementation of the complete thing that we have seen, so far ((Refer Time: 05:37)). So, it is actually you look at this picture, so this is an XOR gate, there is an AND gate, this is an XOR gate, this is AND gate and this is OR gate, what if we implemented all of them using only NAND gates, so that is this representation.

So, this circuit is a half adder and part of this is another half adder and the carry out is also absorbing the OR here. So, this is the complete NAND representation of a 1 bit full adder, a 1 bit full adder is 1 which will take 2 inputs a i and b i along with the carry input c i. So, there are 3 inputs in the circuit and it produces sum i and carry i plus 1. So, if you go and look at the delays in the circuit, so this slide is about the performance of the full adder.

Let us assume that all the NAND gates have 1 unit delay, just to keep things simple, we will assume that all the NAND gates have 1 unit delay. What we will do is, we are going to look at analyzing the performance of this. So, let us look at the circuit here, so this is supposed to be for a full adder, it takes 2 bits a and b, it takes a bit c, it gives sum and it gives carry out. So, I have drawn this dash line here, so top of this is the first half adder and below that is the other half adder.

So, let us see how this whole thing works, so if I change a and b and c let us say in one step I am changing a, b and c at time t equal to 0. So, the changes in a and b will have to

go through at least 3 NAND gate delays for this wire to change, you already know how to analyze delays. So, if I start from here it is at least 1 unit here, 2 unit here, 3 unit here for this path, if I start from a 1, 2, 3 on this path also, if I start from b again 1, 2, 3, 1, 2, 3 and so on.

So, there are 4 parts starting from a and b ending in this wire are actually here, so ending in this wire. So, this takes 3 gate delays, so that is what is written as blue here, the blue one is 3 gate delays. Now, if you go and look at this path, this is taken only one gate delay that is going to the next stage circuit, it is going to the next stage. Now, let see the next path if I go and look at this path from here, let us say the input settled down at some point here if I go and look at what is the delay here, then it goes through 1 plus 2, 2 NAND gates, so that is this path here.

So, starting from a, b to change C out you need 3 gate delays plus 2 gate delays or 5 gate delays, if I go through from a, b to this one. So, any change in a, b may take 5 gate delays for C out to change. So, however C out is also taking in from c, so let us see what happens to that. So, c is coming from here, it goes through 1 gate delay and another 2 gate delays. So, the longest path or the critical path is coming from a or b through this NAND network coming in through this and through these two finally to C out.

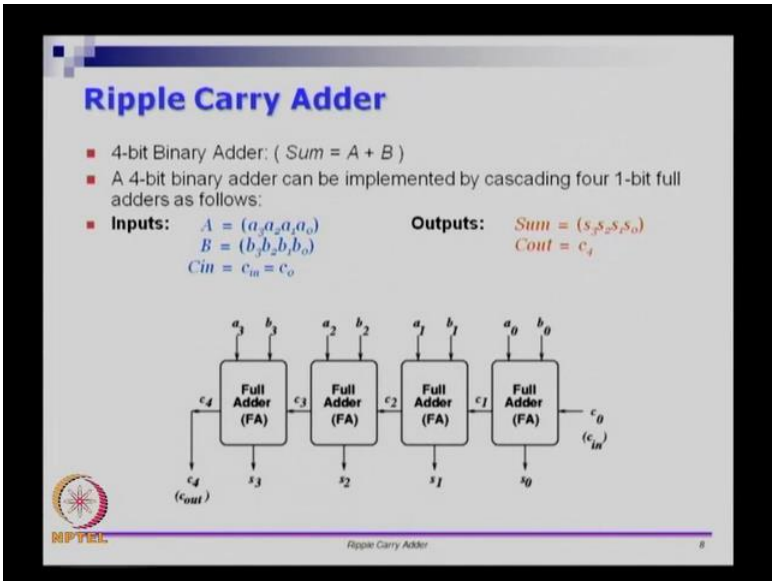
So, changing a or b will result in at least 5 units of delay to change C out, but the change in c, it requires only two units of delay. So, if I have a, b, c at time t equal to 0 at we can expect at 5 gate delay time C out will settled down. And if we go and look at sum, this gets settled down at 3, this was available at 0 itself. So, this input is available at 3 time units and this is available at 0 and there are 3 gate delays here, so sum will settled down at 6 time units.

So, in a single full adder if I give a, b and c at time t equal to 0, in t equal to 5 gate delays C out will settle down and t equal to 6 gate delays, sum will settle down. So, the carry gets generated faster than the sum and that is useful, we will come to that in a little while, the carry is getting generated faster than the sum. And finally, this last path is c going in directly through this and so on, that is only two units of delays.

So, any way let us the green path is not so interesting, the blue one followed by the orange one gives me the delay from change in input a, b, c to change in output C out. Then, the blue line followed by the orange line, this light orange one is change in a, b, c resulting in change of sum, change of c however is not a big deal, it is going through

fewer gate delays anyway.

(Refer Slide Time: 10:52)



So, now, let us get into ripple carry adder, this is also called carry ripple adder in a few books. So, you can implement a 4 bit binary adder by cascading four 1 bit full adders, so let us see the inputs and outputs. The inputs are a 3, a 2, a 1, a naught, b 3, b 2, b 1, b naught and C in is, if there is any input to the ripple carry adder you can give that; otherwise, you can make it 0 and the result would be sum s 3, s 2, s 1, s naught and a C out called s 4, these will be the inputs and outputs.

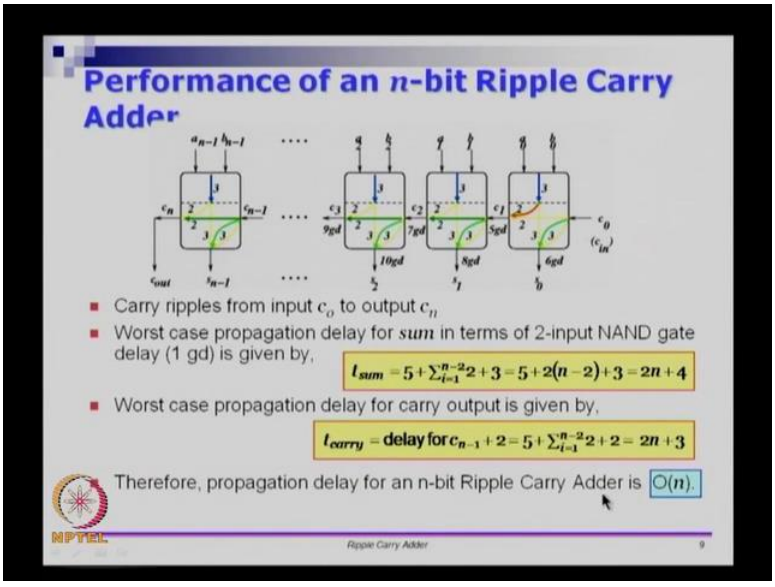
Let us see how this will be built, you take a full adder and you give a naught, b naught to it, if there is a carry input you give c naught, if there is no carry input it put a 0 here. This will give you sum naught and the carry one, this carry goes as the next stage carry, this what we do in decimal addition also. So, if I have the 1's, the 1's position if I add two numbers, the carry from that will go to the 10's stage. So, if I in the 10 stage whatever carry comes out will go to the 100 stage and so on.

So, imagine this as 0's, 1's, so this is 1's, so in decimal this would have been 1's, 10's, 100's and 1000's, here it would be 1's, 2's, 4's and 8's those are the 4 stages. So, this c 1 and a 1 and b 1 will change s 1 and c 2, then when s 2 comes a 2 and b 2 along with c 2 will result in s 2 and c 3 and so on. So, the key thing that you have to remember here is, there is something called rippling of carry. So, the carry that is generated from c 1 may go and change the values for s 1 and c 2, c 2 in turn may go and change the values for s 2 and c 3 which in turn may change the value of s 3 and c 4 and so on.

So, the carry that is generated from this right most stage may ripple through this rest of the circuit and may result in a change of c 4, so imagine if I added 0 0 0 1 to 1 1 1 1. So, a 3, a 1, a naught would all have be in 1's and b 3, b 2, b 1, b naught would be 0 0 0 1, let us say c naught is 0. So, all these are 1's, a is 1 and b naught alone is 1, so a naught and b naught will result in a sum of 0 and a carry of 1, that 1 here is coming now, that will flip this sum bit and change the carry bit which will in turn flip this bit and generate a carry bit, which will in turn flip this bit and generate a bit here, the result will be one followed by four 0s.

So, that carry generated from here ripple through to all the way here and there is a reason, why we are looking at the circuit, the delay that you get from the circuit is quite long.

(Refer Slide Time: 13:56)



So, let us see what the delay is, so if I go and look at the internal diagram. So, this is a and minus 1 and this is b n minus 1, let us say there is a stage a 2, b 2 and there are other stages which are not shown a 1, b 1, a naught, b naught and so on. Let us see what is the longest path in the circuit or the critical path in the circuit. If you go and look at to the critical path, let us say all the inputs are given simultaneously, all the bits a's and b's are given in one go.

You will see that a time unit 3 it have pass through the first adder and in time unit 6 s n minus 1 all the way to s naught would get some intermediate values. So, everything from s n minus 1 to s naught would get some values at time unit 6. However, at time unit 5 we

get c 1 which is coming through, this c 1 can potentially change both sum and carry. So, this came in through a delay 2, so 3 plus 2 is 5 and if you look at this path, this is already coming at 5 unit, this would have settled down at 3 units. So, this is waiting 5 units plus 2 units the 7 gate delays at 7 gate delays c 2 will stabilize, c 2 will not change anymore.

So, c 2 is stabilizing at 7 units this first half adders output will stabilize at 3 units itself, so 7 plus 2 more NAND gates. So, this is we are talking about these NAND gates, these 2 NAND gates ((Refer Time: 15:32)). So, if this comes at 7 units and this is send at 3 at 9 this will get settled down, if this comes at 9 and this input is settled at 3, this will come at 11 and so on, that is what we are looking at.

So, at 7 if this settle at 7 after 2 time units this will get settled down, after 2 time units the next stage is getting settle down and so on and every stage will settled down in the next 2 time units and so on. So, overall if you look at the calculations, so sum if you look at it, so this path this will take 3 plus 3 which is 6 time units, then this got settled at 5. So, this settled at 5 plus 3 which is 8, this will settled at 7 plus 3 which is 10, this will settled at whatever c n minus 1 times something plus 3.

So, that will be the time at which it will settled down, so the worst case prorogation delay for some term. So, sum will come the last s n minus 1 will be the very last bit that will come out and if you look at the path, then the longest path will be this one going through this set of carries all the way to this a naught, b naught that will be the longest path you can see that this carry chain or ripple carry.

So, a carry that is changing may eventually come and change here and from there it will take 3 times units for s n minus 1 to settle down. The worst case prorogation delay for sum in terms of 2 input NAND gates is given by 5 time units which is in the very first stage itself. So, the very first stage this carry will come out at 5 plus i equals 1 to n minus to 2 that is through the carry logic.

So, this summation is because of the carry logic, this is the delay taken for the very first carry to be generated c 1 to be generated, this is the delay for c n minus 1 to be generated. So, i equal to 1 to n minus 2 times 2, so we are adding, so many times 2 this is 2 gate delays of NAND gates that is because of generating the c n minus 1. But 1 c n minus 1 is generated you still need 3 more time units to settle s n minus 1.

So, t sum is 2 n plus 4, so where n is number of stages and 2 n plus 4 is the number of gate delays in terms of 2 input NAND gates that we require for this one. So, the worst

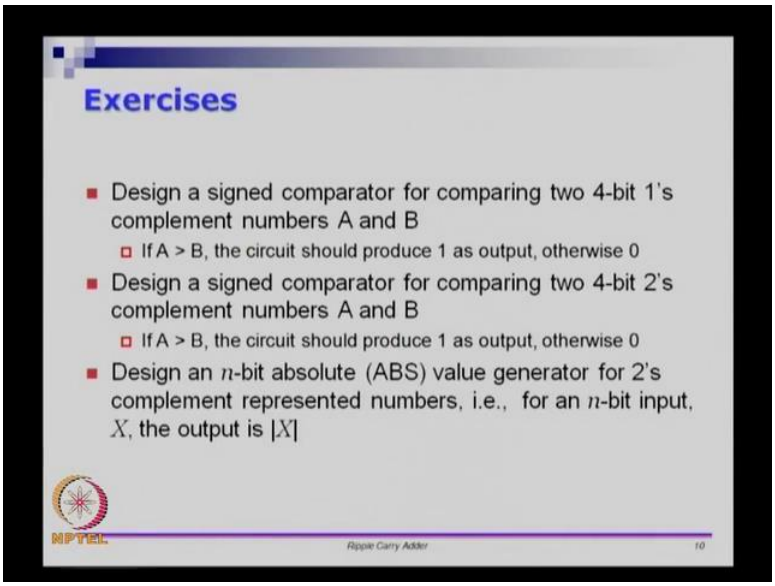
case propagation delay for carry output however is delay for c n minus 1 plus 2 or it is actually 1 unit less than 2 n plus 4. So, you still need to generate c n minus 1 to generate c n minus 1 it is this factor plus 5, but this path only has 2 NAND gates. So, instead of adding 3 we will add only 2 that will be 2 n plus 3.

So, both of them as you keep widening the width of addition operation that you want, you will see that implementing using ripple carry adder is increasing linearly with respect to the size of the adder. So, that is the final bullet, the propagation delay for a n bit ripple carry adder is linear in terms of the number of stages, this bigo of n means this is the function we do not have to say what that function is, but the function is growing linearly with respect to the n.

So, 1 bit adder we may take some time unit, if I want it to 2 inputs it will depend on this number 2, if I want to add 2 5 bit numbers it is going to be a function of n, it is going to be a function of n function of 5, 6 and 7 and so on, if you plot the delays for different n you will see that they are in a line that is why it is order of n. So, ripple carry adder if I want to make a bigger and bigger adder it add the delay increases, the delay increases proportionally with respect to the number of bits.

So, one final thing that I want you to think about is, the carry that is coming out here right it has a interesting property. If I have a 4 bit ripple carry adder and let us say I have another 4 bit ripple carry adder I can take the carry out that is coming from 1 stage and give in as c 0 for the next stage, you can make a 8 bit adder out of that, you can take 2 n bit adders and attach them back to back and you can get a 2 n bit adder. So, that is a very interesting property of a ripple carry adder. So, this is the one of the few circuits were you can take two circuits and out them together and without doing any extra work you can actually get bigger operation done.

(Refer Slide Time: 20:18)



So, I want you to do the, these are the exercises, designed a signed comparator for comparing tow 4 bit 1s complement numbers a and b. So, we are given two numbers a and b, the numbers already only in 1s complement format, compare and give 1 or 0 if you A is greater than B give 1 if it is A is equal to B or A is less than B with 0 similarly do that for 2s complement. And finally, design a circuit were you have a 2s complement number and you have given x you have to give modulo of x.

So, the number is positive it should return a number directly, if the number is negative it should give me the absolute value of the number. So, go and design circuit for this, so to do this do not think in terms of gates think in terms of the additions and subtractions and all these thinks that we thinks done, so for think in terms of that, so you use all the properties as... So, the very first bit if it is a one in 2s complement or in 1s complement the number is negative, use all of that information and see whether you can come up with circuits which can use adders to do this comparator and this absolute value generator.

So, this brings me to the end of module 50, in the next module we will... So, what we will have a look at is we still have to look at a multipliers. So, we have adders now with adders you know how to do this sub tractors, we will look at multipliers dividers are beyond this cope up with this course, dividers take a lot more work and this not something that we can do easily in a under graduate second year level of course, I suggest that if you are interested in dividers, you go and read a materials that is out there.

So, thank you and I will see you in module 51.