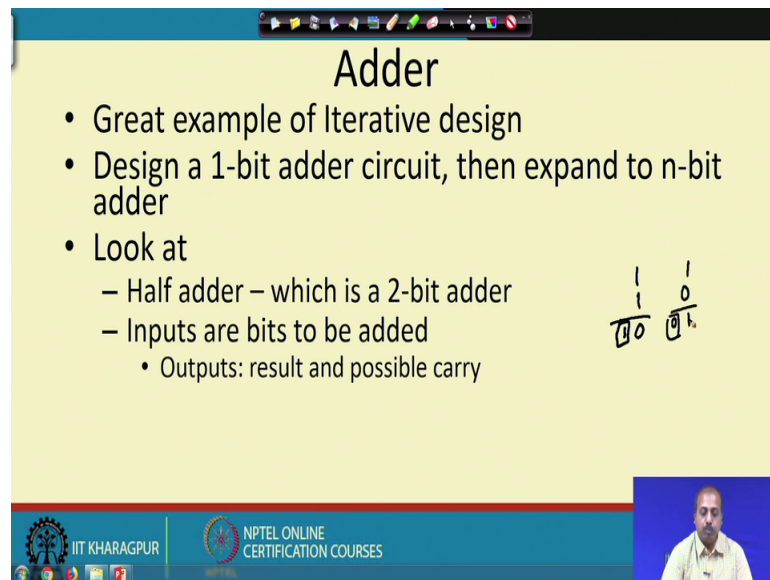


Digital Circuits
Prof. Santanu Chattopadhyay
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur

Lecture - 20
Arithmetic Circuits

So, next we will be looking into Arithmetic Circuits. So, the circuits there are many digital circuits that perform arithmetic operation like addition, subtraction, multiplication, division etcetera and they by themselves form one special class of circuits ok. So, this is arithmetic circuits are looking into that.

(Refer Slide Time: 00:40)



Adder

- Great example of Iterative design
- Design a 1-bit adder circuit, then expand to n-bit adder
- Look at
 - Half adder – which is a 2-bit adder
 - Inputs are bits to be added
 - Outputs: result and possible carry

$$\begin{array}{r} 1 \\ \hline 10 \\ \end{array} \quad \begin{array}{r} 1 \\ \hline 01 \\ \end{array}$$

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

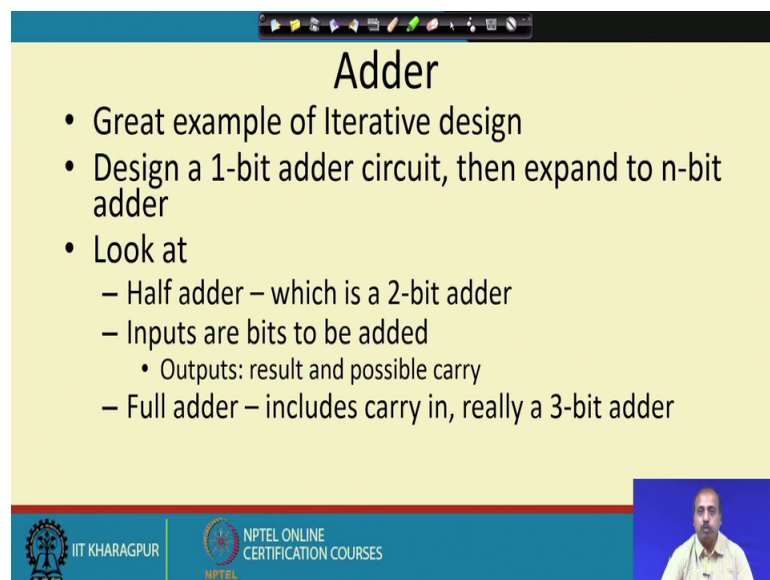
So, the first example of this arithmetic circuit that we will consider is that of an adder this is an example of iterative design. So, iterative design means so, you have got a basic block and that basic block is repeatedly used for getting the more complex designs ok. So, that way this adder is an example of this iterative design. So, we design a 2-bit adder and using the 2-bit adder. So, we can expand it to 3 bit, 4 bit, 5 bit like that.

So, we design an 1-bit adder circuit and then expand to n-bit adder circuit. So, this 1-bit adder; so it has got 2 1-bit inputs and it produces the some of them sum and carry bit and accordingly and then this 1-bit adder can be expanded to have any arbitrary n-bit adder

So, we will look into half adder first; so which is a 2-bit adder and inputs are bits to be added and the outputs are result and possible carry. So, when you are adding 2 1-bit numbers say; so, if the bits are say 1 and 1 then the sum is 0 and there is a carry of 1. So, this has to be generated; whereas if it is 1 and 0 then the sum is 1 and the carry there is no carry; so, carry is 0.

So, we can have this type of situation or say 0 0; sum is 0 and the carry is also 0. So, this way we have got half adder; so, this is a 2-bit adder and then output and this sum and carry; so these are the 2 outputs.

(Refer Slide Time: 02:17)



Adder

- Great example of Iterative design
- Design a 1-bit adder circuit, then expand to n-bit adder
- Look at
 - Half adder – which is a 2-bit adder
 - Inputs are bits to be added
 - Outputs: result and possible carry
 - Full adder – includes carry in, really a 3-bit adder

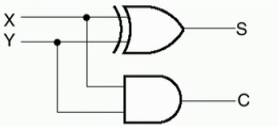
IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, on the other hand we have got full adder which includes a carry in also. So, it has got these 2 bits to be added and it has got a carry input. So, that is this is actually all this 3 bits will be added to produce the sum and the carry.

(Refer Slide Time: 02:33)

Half Adder




- $S = X \oplus Y$
- $C = XY$



Logic Diagram of Half Adder

Truth Table of Half Adder

Inputs		Outputs	
X	Y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0



So, the half adder is something like this; so, this is the logic diagram of half adder. So, this is the; so, truth table is like this. So, as I have said that it has got inputs X and Y and it has got 2 outputs sum and carry. So, how; so, if 0 and 0 are added these 2 bits are added then sum equal to 0 and carry is also equal to 0. If it is 0 and 1 sum is equal to 1 and carry is equal to 0, 1 0 sum equal to 1 carry equal to 0; 1 1 sum equal to 0 and carry is equal to 1.



So, if I look into the corresponding circuit then this sum output it is nothing, but the X OR function you see whenever X is 0 1 or Y is 1 then only the output is 1 and that is in the exclusive OR fashion. So, this S output is the X OR of X and Y and the carry output is the AND of X and Y. So, C is equal to 1 when X and Y both the inputs are at 1. So, we have got the logic diagram of half adder consisting of one X OR gate and one AND gate. So, S equal to $X \oplus Y$ and C equal to XY

(Refer Slide Time: 03:45)

Full Adder

- Three inputs. Two are operand bits, third is C_{in}
- Two outputs: sum and carry

Inputs			Outputs	
X	Y	Z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

The other one full adder; so, in case of full adder we have got this inputs like this. We have got this X Y and Z as input and then when this 0 0 0 0 are added. So, sum is 0 and carry is 0 when 0 0 1 in that case sum is 1 and carry is 0. So, for 0 1 1 sum is one sum is 0 and carry is 1 and when it is 1 1 1 both sum and carry are 1. So, in this way we have got the truth table of the full adder.

Now, it has got 3 inputs the full adder has got 3 inputs 2 are the operand bit say X and Y may be the operand bit and the third is the carry input C in.

(Refer Slide Time: 04:36)

K Map for S

YZ		Y			
		00	01	11	10
X	0		1		1
	1	1		1	

Z



- What is this?

$$S = \bar{X}\bar{Y}Z + \bar{X}Y\bar{Z} + X\bar{Y}\bar{Z} + XY Z$$

$$= X \oplus Y \oplus Z$$

In a half adder the sum bit:
 $S = X \oplus Y$

Inputs			Outputs	
X	Y	Z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

It has got 2 outputs sum and carry and so, for the Karnaugh map for sum is like this. So, if you if you draw the Karnaugh map then it will be like this that we have got this. So, this is basically the X OR ok; so, this is the X OR of XY and Z. So, this is basically this a sum equal to $X \bar{Y} \bar{Z}$ plus $X \bar{Y} Z$ plus $Z Y \bar{Z}$ plus xyz which is nothing, but $X \bar{X} \text{ OR } Y \bar{X} \text{ OR } Z$. So, the sum is equal to $X \text{ OR } Y \text{ OR } Z$ and in case of half adder you remember that the sum is equal to $X \text{ OR } Y$.

So; that means, if I use this half adder and we can make a full adder out of that how? We will see that later.

(Refer Slide Time: 05:20)

K Map for C

Inputs			Outputs	
X	Y	Z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

$C = XY + XZ + YZ$
 $= XY + Z(X\bar{Y} + \bar{X}Y)$
 $= XY + Z(X \oplus Y)$

In a half adder the carry bit:
 $C = XY$

But this K map for Karnaugh map K map for the carry output is like this. So, carry output is 1 whenever this Y and Z equal to 1 then or this X Z equal to 1 XY equal to 1 or XYZ equal to 1. So, I if I draw the truth table and then make the grouping.

So, it will be like this and so, I can say that is C equal to XY plus XZ plus YZ. So, it can be written as XY plus Z into X plus Y and this X plus Y can be written in this in this X OR form also because if X and Y both are equal to 1; then this term actually takes care of this thing. So, of this C output but if this only one of them is equal to 1, then this Z; Z being equal to 1; so output should be equal to 1.

So, I can I can this I can change this XY term X plus Y term into this X OR form. And then it terms out to be XY plus Z into X X OR Y; so, Z and X X OR Y. So, in a half

adder the carry output was XY and in case of full adder; so, you have got this as the carry output.

(Refer Slide Time: 06:37)

Using two half Adders to Build a Full Adder

- Full adder functions

$$S = \bar{X}\bar{Y}Z + \bar{X}YZ + X\bar{Y}Z + XYZ$$

$$= X \oplus Y \oplus Z$$
- Half adder functions

$$S = X \oplus Y$$

$$C = XY$$

$$C = XY + XZ + YZ$$

$$= XY + Z(X\bar{Y} + \bar{X}Y)$$

$$= XY + Z(X \oplus Y)$$

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, if we consider the full adder function; so, sum equal to $X \oplus Y \oplus Z$ and carry equal to C equal to XY plus Z into $X \oplus Y$. So, and the half adder function was S equal to $X \oplus Y$ and C equal to XY .

(Refer Slide Time: 06:53)

Two Half Adders (and an OR)

Logic Diagram of Full Adder

The diagram shows two half adders and an OR gate. The first half adder takes inputs X and Y and produces sum $X \oplus Y$ and carry XY . The second half adder takes inputs $X \oplus Y$ and Z and produces sum $Z \oplus (X \oplus Y)$ and carry $Z(X \oplus Y)$. The final sum S is $X \oplus Y \oplus Z$ and the final carry C is $XY + Z(X \oplus Y)$.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, what we can do? We can use 2 half adders to get a full adder. So, here you see that the in the first half adder we have given X and Y as input. So, we have got XY as the

carry output and $X \oplus Y$ as the sum output. So, with that if we put this Z as the second input. So, second half adder has got this sum as one input and the Z as another input; so, this $X \oplus Y \oplus Z$.

So, that gives me the sum S equal to $X \oplus Y \oplus Z$ and this carry is generated so, this carry part. So, what we have done? We have taken a separate OR gate. So, this first this half adder; first half adder; so it gave XY as output in the carry, but what we need is XY plus Z into $X \oplus Y$.

So, this AND gate will be realizing that Z into $X \oplus Y$ because $X \oplus Y$ is coming here. So, it is in the half adder form; so, it is connected to the AND gate. So, it is connected here and then this Z input is connected. So, here we get Z into $X \oplus Y$; so, ultimately if I use an OR gate to OR between these 2 outputs. So, you get C equal to XY plus Z into $X \oplus Y$. So, that is the full adder logic diagram; so, using this half adder, so we can realize a full adder.

(Refer Slide Time: 08:20)

Ripple-Carry Adder

- Straightforward – connect full adders

Handwritten notes:
 $C_3 \ C_2 \ C_1 \ C_0 = 0$
 $A_3 \ A_2 \ A_1 \ A_0$
 $B_3 \ B_2 \ B_1 \ B_0$
 $\hline S_4 \ S_3 \ S_2 \ S_1 \ S_0$

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Next, we will be looking into another extension like how can I have more number of bits in an adder? So, so for wherever we have seen a full adder; so, it can have 2 inputs is A and B and a carry input and then it is producing the sum output and carry output just in the previous slide we have seen that a full adder. So, it is having XYZ as the 2 input as Z may be considered as carry input and it is producing a sum output and a carry output.

Now, if you have got a number of if you have to if you have got to add a number bits word size of individual numbers are more say 4 bit numbers; then what we can do we can connect this full adders in a cascaded fashion ok. So, this C_0 is the carry in coming from the first adder.

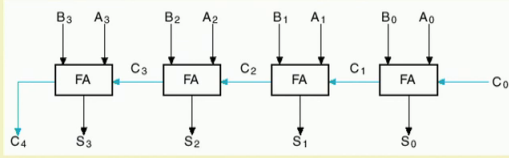
So, it may be if there is no carry for the addition; so, basically I am interested to do this addition this A_3, A_2, A_1, A_0 this bit pattern I want to add with B_3, B_2, B_1, B_0 . So, and there may be some previous carry coming into the thing; so, that is a C input. And if this carry is not there then we can set it to 0 or this C_0 can be set to be equal to 0 if there is no carry coming from the previous stage; so, carry coming from the input side

So, that way we can; so, what we are doing? For the first full adder; so, we are giving A_0 and B_0 and this C_0 . So, it produces the S_0 bit and the carry 1. So, this A_0 plus B_0 ; so the C_0 plus sum 0 sum comes here and the carry output goes here. The second full adder will add the C_1, A_1 and B_1 ; it will produce S_1 and it will produce another carry C_2 ok. So, C_2, A_2 and B_2 ; so they will added by the third full adder. So, this will giving S_3 and C_3 and this will be giving me S_4 and C_4 .


So, that is what is happening ok. So, we have got two 4 bit numbers they are getting added and it is producing a 5 bit number consisting of 4 sum bits and 1 carry bit. So, this way in a straight forward fashion; so, we can connect full adders and get the circuit for higher number bits.

(Refer Slide Time: 10:50)


Ripple-Carry Adder




- Straightforward – connect full adders
- C_4 : Chain carry-out to carry-in of FA of bits A_4 & B_4
 - C_0 in case this is part of larger chain
 - otherwise just set to zero



IIT KHARAGPUR

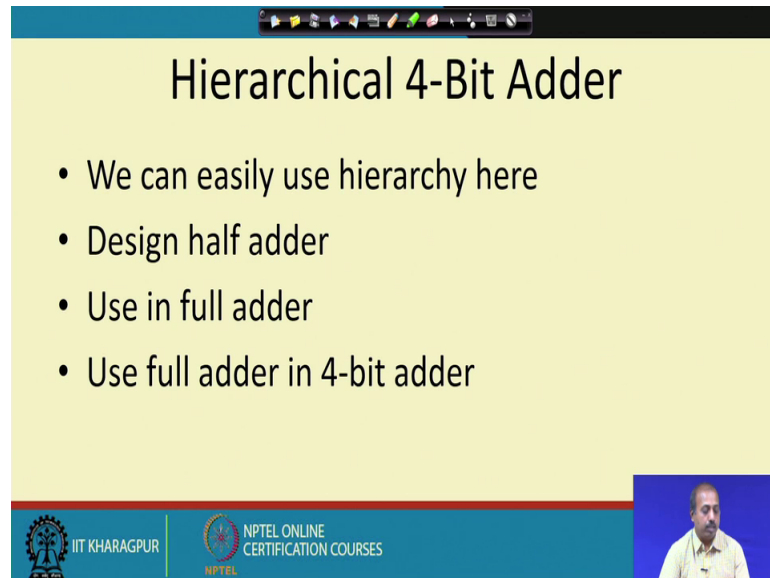


NPTEL ONLINE
CERTIFICATION COURSES



So, C_4 is the chain carry out to the carry in of a full adder of bits A_4 and B_4 . So, we can if we have got next stage A_4 and B_4 ; so, we can we can chain it to the next stage and C_0 in this case is part of larger design. So, if it is coming from another such some such adder then C_0 may be the carry chain output of that adder otherwise we can just set it to 0.

(Refer Slide Time: 11:17)



Hierarchical 4-Bit Adder

- We can easily use hierarchy here
- Design half adder
- Use in full adder
- Use full adder in 4-bit adder

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, we can now we can see the hierarchy ok. So, for 4 bit adder; so, we first design the half adder and use that half adder to design full adder and then use this full adders; it will get say 4 bit adder. So, that way we can have a clear cut hierarchy in the adder design.

(Refer Slide Time: 11:37)

Subtractor

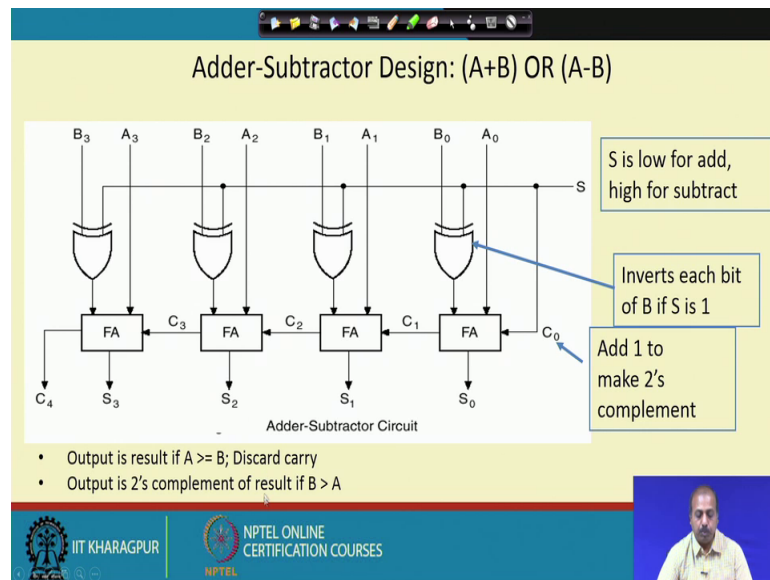
- Compute M-N
 - Add 2's complement of N to M:
 $M + (2^n - N)$
 - If $M \geq N$, need only “one adder” and a “complementer of N” to do the subtraction.
 - If $M < N$, we also need to take 2's complement of adder's output to produce magnitude of result
 $2^n - \{M + (2^n - N)\} = N - M$

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

On the other hand subtractor: if you want to compute M minus N and from our knowledge of this number theory. So, we know that we have to add 2's complement of N to M. So, we should have this so that is M plus 2 power N minus; so, that is to be done. Now if M is greater or equal N. So, in that case we need only one adder and complementer of N to do the subtraction. So, if M is greater or equal N then we just; so, we take the 2's complement of N and do the addition that will give me the subtraction.

However, if M is less than N ok; so, then we need to take 2's complement of adders output to produce the magnitude of the result. So, if M is less than N then the after doing the; so, the addition part is same or the sub same as the 2 complement addition part subtraction part. So, M plus 2 power N minus; so, that is done in both the cases, but for getting the 2s complemented result. So, we have to do this 2 power N minus this that is that will give us N minus N ok.

(Refer Slide Time: 12:50)



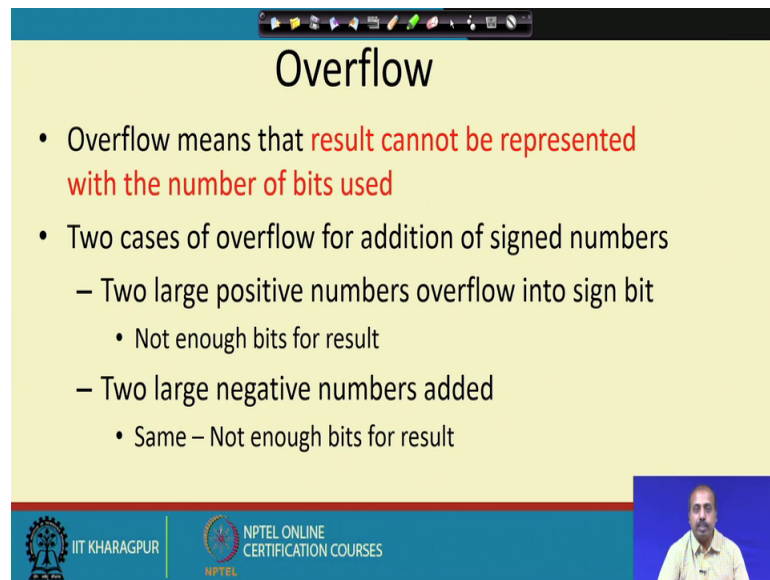
So, this is the adder subtracted design A plus B or A minus B. So, either of them can be done so this. So, here this S is it is the S is a control. So, this S is low for addition and high for subtraction. So, if S is low; so here I am getting a 0 C 0 is 0. So, it will be simply doing and this X OR gates. So, they will be having this S input has 0. So, as a result they will be passing it as 0 the B 0 will be passed here B 1 will be passed here, B 2 will be passed here and B 3 will be passed here. So, this will do the simple addition.

Now, if you are doing subtraction in that case this S will be equal to 1. So, this B 0, B 1, B 2, B 3 so, they will get complemented via this X OR gate; in X OR gate if one of the input is tied high we know that it acts as an inverter. So, at this point I will get B 0; bar here I will get B 1 bar, B 2 bar and B 3 bar.

So, I will get the complemented versions there and then this C 0 is equal to 1. So, this is the I am getting A A plus complemented 2's complement of B plus 1. So, that way I sorry 1's complement of B plus 1; so, that gives us the 2s complement of B and ultimately that results in A minus B.



So, this adder subtracted design is very simple; so, on top of this basic adder circuit. So, we have got a complements circuit which is consisting of a few X OR gates and that gives us the solution. Output is a result if A greater or equal B; so we just discard the carry bit and output is 2's complement of result, if B is greater than A. So, if you want to get back the original value then we have to take another 2's complement at this point.


(Refer Slide Time: 14:43)



Overflow

- Overflow means that **result cannot be represented with the number of bits used**
- Two cases of overflow for addition of signed numbers
 - Two large positive numbers overflow into sign bit
 - Not enough bits for result
 - Two large negative numbers added
 - Same – Not enough bits for result

 IIT KHARAGPUR |  NPTEL ONLINE CERTIFICATION COURSES



Now, overflow may occur; so, what is an overflow? So, when the result cannot be represented within the number of bits used. So, doing some 2 additions and then the I am doing 2 4 bit number addition the result can go to 5 bits. So, that way they if I am using only 4 bits for their realization; so there is an overflow. Particularly, for 2's complement number system when I am doing the subtraction; so, it can occur quite often.

So, there can be 2 cases of overflow for addition of signed numbers 2 large positive numbers overflow into sign bit. So, not enough bits for the result or 2 large negative numbers are being added and same that is a not enough bits for the result. So,; so, as I said that 4 bit numbers being added it gives 5 bit result and the 5th bit is actually for the sign. So, if I say that way then that will be effect in the sign bit.

(Refer Slide Time: 15:40)

• Consider $7 + 7$

• **Overflow: cannot represent 14 using only 4 bits**

• Generates NO CARRY, $C_4 = 0$

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, 7 plus 7; so, this is giving me 14. So, if I am. So, in 4 bit; so, if I am representing both positive and negative numbers we know the range is minus 8 to plus 7. So, the plus 14 cannot be realized you cannot be represented in 4 bit; so that will give us an overflow.

(Refer Slide Time: 16:00)

Example 2

• Consider $-7 - 7$

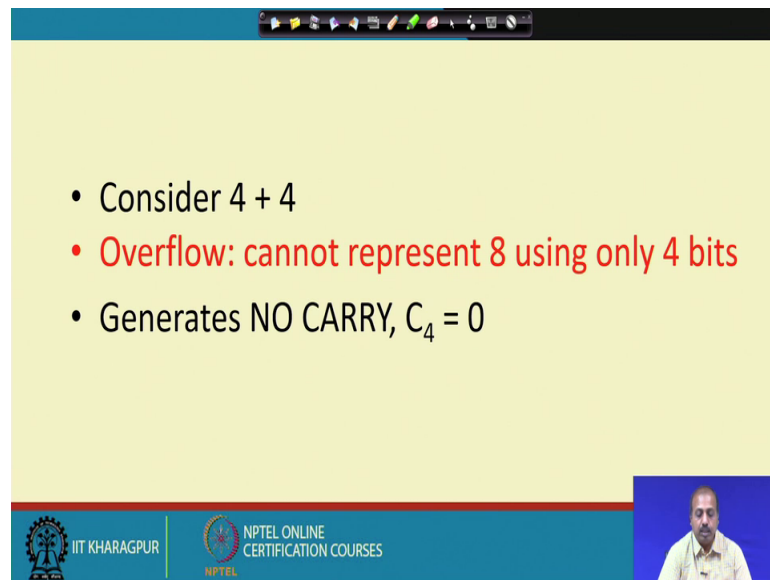
• **Overflow: cannot represent -14 using only 4 bits**

• Generates CARRY, $C_4 = 1$

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, it generates no carry and C_4 may remain equal to 0; on the other hand say you can consider minus 7 minus 7 that is the result is minus 14. So, here also it is an overflow because I cannot represent minus 14 in 4 bits, but in this case it generates a carry. So, C_4 equal to 1 if you look into that final carry generated; so, this C_4 will be equal to 1.

(Refer Slide Time: 16:25)

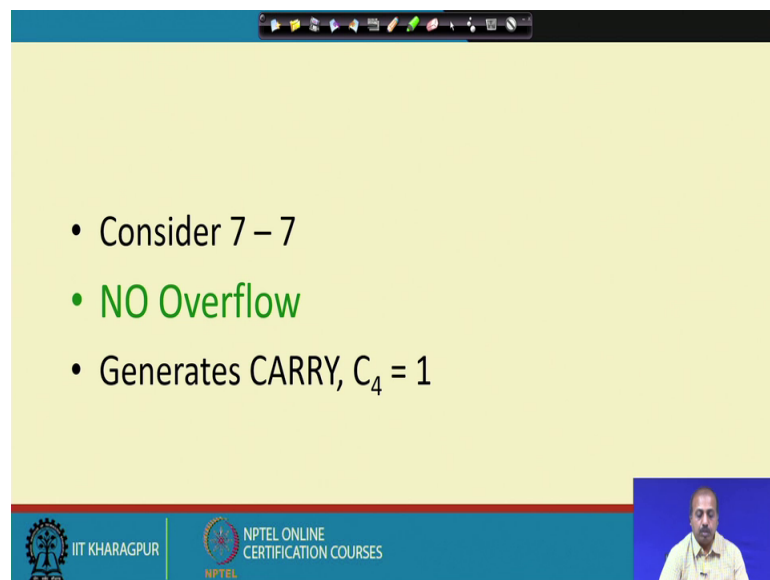


A slide from an NPTEL lecture. The slide has a yellow background and a blue footer. The footer contains the IIT Kharagpur logo, the text 'IIT KHARAGPUR', the NPTEL logo, and the text 'NPTEL ONLINE CERTIFICATION COURSES'. A small video inset of a man is in the bottom right corner. The slide content is as follows:

- Consider $4 + 4$
- **Overflow: cannot represent 8 using only 4 bits**
- Generates NO CARRY, $C_4 = 0$

Now, 4 plus 4; so, here the result is 8 here also I cannot represent 8 in 4 bits because the range is minus 8 to plus 7. However, it does not generate a carry C_4 equal to 0.

(Refer Slide Time: 16:38)



A slide from an NPTEL lecture. The slide has a yellow background and a blue footer. The footer contains the IIT Kharagpur logo, the text 'IIT KHARAGPUR', the NPTEL logo, and the text 'NPTEL ONLINE CERTIFICATION COURSES'. A small video inset of a man is in the bottom right corner. The slide content is as follows:

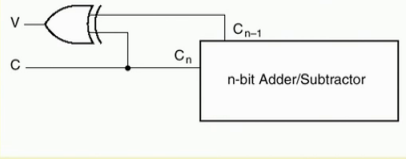
- Consider $7 - 7$
- **NO Overflow**
- Generates CARRY, $C_4 = 1$

And the 7 minus 7; there is no overflow and, but it generates a carry C_4 equal to 1. So, if you trace through the addition process then you can understand.




(Refer Slide Time: 16:47)

Overflow Detection

- Condition for overflow:
 - either C_{n-1} or C_n is high, but not both



- 7 + 7; only C_{n-1} (C_3) is high ; **overflow**
- -7 - 7; only C_n (C_4) is high ; **overflow**
- 4 + 4; only C_{n-1} (C_3) is high ; **overflow**
- 7 - 7; BOTH C_{n-1} & C_n (C_4 & C_3) are high; **no overflow**

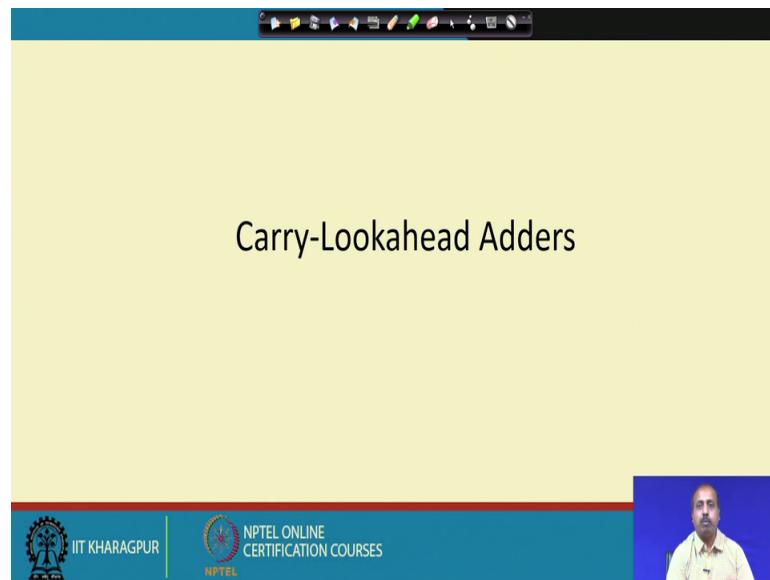
  

So, this overflow detection is basically if there is an addition at adder subtractor. So, you have to compare between the carries generated at nth stage and n minus 1th stage. So, if the if these 2 carry if you if you take an X OR of these 2, then that will be giving us the overflow bit; so, whether it is overflow or not

So, the condition for overflow is either C_{n-1} or C_n is a high, but not both. So, that way it; it this is the overflow condition. So, all 4 4 5 cases that we have seen previously, if you if you make a summary of them; then we will see that it comes to some such conclusion that is a C_{n-1} or C_n should be high, but not both; so, that is the condition for overflow

So, this 7 plus 7 only C_3 is high; so C_4 is not high. So, that is an overflow minus 7 minus 7. So, C_4 is high, but C_3 is not high; so, that is also an overflow plus 4 plus 4. So, that is C_{n-1} is high that is C_3 is high so again there is overflow. And 7 minus 7; so, both the carries are generated, but there is no overflow. So, that way this overflow detections circuit is the X OR of C_n and C_{n-1} .

(Refer Slide Time: 18:08)



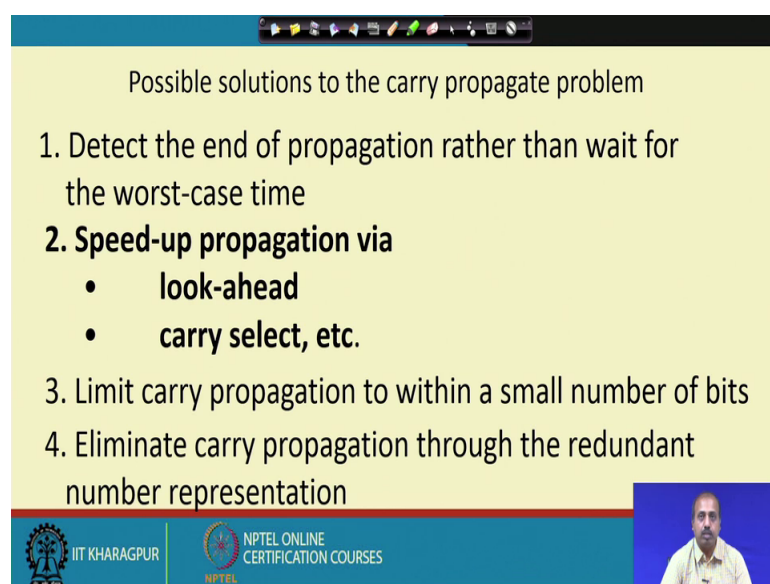
Carry-Lookahead Adders

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Now, one difficulty with this type of adder that we have seen is that I have got; I have cascaded number of full adders. Now the second stage of the adder it cannot work till the first stage has completed the addition and the carry bit is available from the first stage.

So, that way if we go ahead then for further and further stages; there will be problem in terms of this carry not yet appear not yet come to the adder stage ok. So, that makes this addition process or these ripple carry addition process pretty slow ok.

(Refer Slide Time: 18:49)



Possible solutions to the carry propagate problem

1. Detect the end of propagation rather than wait for the worst-case time
2. **Speed-up propagation via**
 - **look-ahead**
 - **carry select, etc.**
3. Limit carry propagation to within a small number of bits
4. Eliminate carry propagation through the redundant number representation

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, another type of adder which is quite fast is known as carry look ahead adder that we will see in this. So, this carry propagation is a problem; so, detect the end of propagation rather than wait for the worst case time to occur. And speed up the propagation while via look ahead carry select etcetera, and it will limit the carry propagation to a small number of bits and eliminate carry propagation through the redundant number redundant number representation. So, these are some of them so we will see them slowly.

(Refer Slide Time: 19:15)

Basic Signals

Generate signal: $g_i = x_i y_i$
 Propagate signal: $p_i = x_i \oplus y_i$
 Anihilate (absorb) signal: $a_i = \overline{x_i} \overline{y_i} = \overline{x_i + y_i}$
 Transfer signal: $t_i = g_i + p_i = a_i = x_i + y_i$
 $c_{out} = 1$ given $c_{in} = 1$
 Carry recurrence

$$c_{i+1} = g_i + c_i p_i = g_i + c_i t_i$$

The diagram shows a full adder stage with inputs x_i , y_i , and c_i , and outputs s_i and c_{i+1} . The truth table for c_{i+1} is:

x_i	1	1
y_i	1	0
c_i	0	0
c_{i+1}	1	1

Now, you see that for a particular stage ok; so. So, if I take an example then suppose this is a particular adder stage ok. So, it has got this x_i and y_i as the bits to be added and c_i is the carry, c_i is the carry. And it produces the sum output and the carry output; now we try to see when a carry will be generated from this stage. So, the carry will be generated only when this x_i and y_i both are equal to 1; so, that is.

So irrespective of whatever be the carry coming from the previous stage; if x_i and y_i both are equal to 1; so, this is x_i and y_i if both are equal to 1 then irrespective of whatever be the carry coming from the previous stage, so it will generate a carry; this carry will be generated. So, if I have got a previous stage; so then this carry will be coming to this stage. So, this carry will be generated from this stage.

And other condition is other possibility of generating a carry at getting a carry at this point is there was a carry coming to the previous stage and that propagated through this stage and when this will propagate? So, this will propagate when only one of this x_i and

y_i value is equal to 1 because in that case that one will be added. So, suppose x_i equal to 1 and this carry i equal to 1; so, as a result this sum will be 0 and this carry will be generated at this stage. So, I can say that whatever carry was coming as input to the stage that is getting propagated to the next stage.

However, if both of them are equal to 1; so, x_i and y_i both are equal to 1. So, the sum is like this; so, anyway this, so this will be the output and anyway this X this x_i and y_i being equal to 1; so, carry is already generated. So here, the generation and propagation does not have any separate meaning, so they are all the same. So, we can say in summary that this x_i and y_i . So, they are they will generate a carry if both the bits are equal to 1. So, x_i generate is equal to $x_i y_i$ and propagate is equal to $x_i \oplus y_i$. So, if only one of them is equal to 1 then it will propagate.

Now, when a carry will not propagate? So, it will not propagate or it is absorb if a_i is $\bar{x}_i \bar{y}_i$. So, if both x_i and y_i are 0 then the carry that was coming from previous stage will get absorb there.

(Refer Slide Time: 22:00)

Basic Signals

Generate signal:	$g_i = x_i y_i$
Propagate signal:	$p_i = x_i \oplus y_i$
Anihilate (absorb) signal:	$a_i = \bar{x}_i \bar{y}_i = \overline{x_i + y_i}$
Transfer signal:	$t_i = g_i + p_i = \bar{a}_i = x_i + y_i$

$c_{out} = 1$ given $c_{in} = 1$

Carry recurrence

$$c_{i+1} = g_i + c_i p_i = g_i + c_i t_i$$

x_i	0	
y_i	0	
	1	←
	1	

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, what I mean; what I say is if this x_i and y_i both are 0. So, if there is a carry coming from the previous stage then this will be get getting absorbed in it sum and the carry generated is 0 from this stage so that is the absorbed signal.

So, we can say that there is an absorbed signal a_i which is x_i NOR of. So, \bar{x}_i and \bar{y}_i or the nor of x_i and y_i and there is a transfer signal which is g_i or p_i . So, it will be transferred the; if it is if the carry is generated or carry is propagated. So,; so, this g_i plus p_i , so it can be written as a_i which is \bar{x}_i or \bar{y}_i and C_{out} is 1 given C_{in} equal to 1 and the carry recurrence is like this.

So, so I can say that c_{i+1} is either the carry is generated at this stage or there was a carry coming from the previous stage and this stage allowed propagation ok. So, this was the expression is g_i plus $c_i p_i$ and this expression can be simplified to be written as g_i plus $c_i t_i$ instead of $c_i p_i$. So, I can write is a write it as $c_i t_i$ also the transfer signal. So, this way I can say the how the carry recurred carry recurrence occurs.

(Refer Slide Time: 23:30)

Unrolling Carry Recurrence

$$\begin{aligned}
 c_i &= g_{i-1} + c_{i-1}p_{i-1} = \\
 &= g_{i-1} + (g_{i-2} + c_{i-2}p_{i-2})p_{i-1} = g_{i-1} + g_{i-2}p_{i-1} + c_{i-2}p_{i-2}p_{i-1} = \\
 &= g_{i-1} + g_{i-2}p_{i-1} + (g_{i-3} + c_{i-3}p_{i-3})p_{i-2}p_{i-1} = \\
 &= g_{i-1} + g_{i-2}p_{i-1} + g_{i-3}p_{i-2}p_{i-1} + c_{i-3}p_{i-3}p_{i-2}p_{i-1} = \\
 &= \dots = \\
 &= g_{i-1} + g_{i-2}p_{i-1} + g_{i-3}p_{i-2}p_{i-1} + g_{i-4}p_{i-3}p_{i-2}p_{i-1} + \dots + \\
 &\quad + g_0p_1p_2 \dots p_{i-2}p_{i-1} + c_0p_0p_1p_2 \dots p_{i-2}p_{i-1} = \\
 &= g_{i-1} + \sum_{k=0}^{i-2} g_k \prod_{j=k+1}^{i-1} p_j + c_0 \prod_{j=0}^{i-1} p_j
 \end{aligned}$$

The diagram shows a chain of logic blocks representing carry propagation. The input carry is c_0 . Each stage k has a generate signal g_k and a propagate signal p_k . The carry signal c_k is generated at stage k if g_k is true, or it propagates from the previous stage if p_k is true. The final carry output is c_i .

So, I have got this c_i is g_i plus c_{i-1} into p_{i-1} . So, if you just expand it; so, you will get the c_{i-1} is g_{i-2} plus c_{i-2} into p_{i-1} ; so, it is like this. So, if you go on expanding like this. So ultimately it will be coming to an expression in this format. So, you can you can get the series sum like this. So, it is g_{i-1} ; so c_i is equal to g_{i-1} that is the; i minus one stage has generate a carry or this stage k from 0 to $i-1$.

So, k -th stage generated the carry and the remaining stages; they propagated the carry ok; so this is the situation. So, this is what we say is that; so, we have got this stages we have got this stages. So, whenever we are talking about so, this is the i -th stage; so either at

this point I have got the c i. So, either this, so this is the stage i minus 1. So, from i minus 1th stage I have got the carry c i. So, c i is either at i minus 1th stage it was generated; so, that is given by this g i minus 1.

Or in i minus 2 stage the carry was generated and this stage just propagated it. So, this is g; so, k equal to k equal to i minus 2. So, it is generated here and it is propagated it is generated at i minus 2 stage and propagated at i minus 1 stage or it is generated at any stage in this site. So, it generated at any k-th stage; so some k-th stage some stage k at which the carry was generated and then it got propagated through all this remaining stages; so, that is taken care of by this expression.

So, this is telling that it is generated at stage k and the stages k minus. So, k plus 1 to i minus 1; they are just propagating the carry. So, that way we have got this carry propagated and then other possibility is that initially c 0 was there. So, it was given the carry was the input carry was 1 and then all the stages 0 to i minus 1 they propagated that carry. So, these are the 3 alternatives that we can have for the carry generation.

(Refer Slide Time: 25:55)

4-bit Carry-Lookahead Adder

$$C_4 = g_3 + g_2 p_3 + g_1 p_2 p_3 + g_0 p_1 p_2 p_3 + c_0 p_0 p_1 p_2 p_3$$

$$C_3 = g_2 + g_1 p_2 + g_0 p_1 p_2 + c_0 p_0 p_1 p_2$$

$$C_2 = g_1 + g_0 p_1 + c_0 p_0 p_1$$

$$C_1 = g_0 + c_0 p_0$$

$S_0 = x_0 \oplus y_0 \oplus c_0 = p_0 \oplus c_0$	$S_1 = p_1 \oplus c_1$
$S_2 = p_2 \oplus c_2$	$S_3 = p_3 \oplus c_3$

IIT KHARAGPUR
 NPTEL ONLINE CERTIFICATION COURSES

Now,; so, if you have just looking into the 4 bit carry look ahead adder. So, this c 4 will be equal to something like this if you expand that expression then c 3 c 2 c 1 c 0 like this. Now S 0 the sum 0 is given by x 0 x or y 0 x or c 0 which is nothing, but p 0 x or c 0 similarly p S 2 is p 2 X OR c 2 S 1 is p 1 X OR c 1. So, this way this sum and carries of individual stages can be written.

(Refer Slide Time: 26:36)

4-bit Carry-Lookahead Adder (2)

$$c_4 = g_3 + c_3 p_3$$

$$c_3 = g_2 + g_1 p_2 + g_0 p_1 p_2 + c_0 p_0 p_1 p_2$$

$$c_2 = g_1 + g_0 p_1 + c_0 p_0 p_1$$

$$c_1 = g_0 + c_0 p_0$$




3 gates less

$$s_0 = x_0 \oplus y_0 \oplus c_0 = p_0 \oplus c_0$$

$$s_2 = p_2 \oplus c_2$$

$$s_1 = p_1 \oplus c_1$$

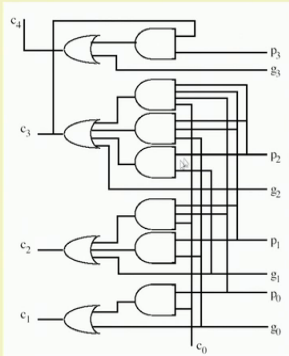
$$s_3 = p_3 \oplus c_3$$








Now, we can just optimize it a bit having a requiring 3 less gates where the C_3 expression this c_4 is g_3 plus $c_3 p_3$ then c_3 is g_2 plus $g_1 p_2$ like this. So, we can just modify this previously written expression optimize it a bit; so, that it will take 3 gates less

(Refer Slide Time: 26:48)

4-bit Carry Network with Full Lookahead



So, the circuit that we get is a something like this c_0 goes to all the stages then this is g_0 . So, this is actually this carry network. So, where this c_1 c_2 c_3 and c_4 they are generated. So, they are there is a based on this generation generate a propagate logic. So,

this is a coming from this expression; so ok. So, this is straight away implementation of this expression.

(Refer Slide Time: 27:22)

Equations for 4-bit Lookahead Carry Generation

$$c_{i+3} = g_{i+2} + g_{i+1} p_{i+2} + g_i p_{i+1} p_{i+2} + c_i p_i p_{i+1} p_{i+2}$$

$$c_{i+2} = g_{i+1} + g_i p_{i+1} + c_i p_i p_{i+1}$$

$$c_{i+1} = g_i + c_i p_i$$

$$g_{[i..i+3]} = g_{i+3} + g_{i+2} p_{i+3} + g_{i+1} p_{i+2} p_{i+3} + g_i p_{i+1} p_{i+2} p_{i+3}$$

$$p_{[i..i+3]} = p_i p_{i+1} p_{i+2} p_{i+3}$$

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Now, so for carry look ahead generation; so, we have got this expression in terms of I, so i plus c i plus 3 I plus 2 i plus 1; so, we have go this expressions.

(Refer Slide Time: 27:34)

4-bit Lookahead Carry Generator Schematic

The schematic shows the implementation of the carry propagation equations. It features a hierarchy of gates:

- Inputs: $c_{i+1}, c_{i+2}, c_{i+3}$ (carry signals) and $g_{[i..i+3]}, p_{[i..i+3]}$ (block signals).
- Block Signal Generation: Intermediate carries $c_i, c_{i+1}, c_{i+2}, c_{i+3}$ are generated from $g_i, p_i, g_{i+1}, p_{i+1}, g_{i+2}, p_{i+2}, g_{i+3}, p_{i+3}$ using a series of AND and OR gates.
- Final Carry: The final carry-out c_{i+3} is the output of a large 4-input AND gate that combines $c_i, p_i, p_{i+1}, p_{i+2}$ with the OR gate output of $g_{i+2}, g_{i+1}p_{i+2}, g_i p_{i+1} p_{i+2}$.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, in general I will have this type of situation. Now the difficulty is that you see the number of gates it is increasing significantly. So it is somehow at some point of time I do not want to generate further carries, ok. So, I want to do some ripple carry and that way I

will make it simple so that so many, so much of gate is gates are not required in the carry generation process.