


Digital System Design
Professor Neeraj Goel
Department of Computer Science Engineering
Indian Institute of Technology Ropar
Ripple Carry Adder

Hello students, today we are going to discuss how to design addition circuits. So in this lecture we will see how to design single bit addition, multi bit additions or faster additions. We also do delay and area analysis, which could be generically applicable to all other circuits also.

(Refer Slide Time: 0:41)



Addition in Binary

- Similar to decimal additions


$1 + 1 = (10)_2$	1 0 1 1 1
$1 + 1 + 1 = (11)_2$	1 1 0 1 1 1
$1 + 0 = (1)_2$	+ 1 1 0 1 0 1


	1 1 0 1 1 0 0

Each adder unit has three inputs:

- one bit from operand 1,
- One bit from operand 2
- Carry from previous adder

Two outputs: Sum and carry





Digital Logic Design: Introduction 59

So, let us quickly revise. We have seen this in of the previous lecture that a addition in binary is very similar to the way we do addition in decimal systems. So that means 1 plus 1 is actually 2 that means one 0 in binary, 1 plus 1 plus 1 is equal to 11 that means 11 in binary and 3 in decimal. Similarly, 1 plus 0 is equal to 1 in binary.

So, if we take an example let us say we want to add couple of bits, then first we take is the least significant bit and we add it and after that sum is produced as well as a carry would be there. Now, when we start adding the second bit, then we have to add these 3 bits then there would be sum as well as a carry would be generated.


So on so forth the third bit, fourth bit and fifth bit and sixth bit. At the end of sixth bit you have a sum as well as a carry which is generated. So, generically we can say that this whole addition process can be done by having the single bit addition and this single bit addition would require 3

inputs, 1 from the operand 1, the other from the operand 2 and 1 which is a carry from the previous adder and they are 2 outputs 1 is sum for this particular bit and carry which would be used as a carry for the next adder unit.

Now, also there is one important point to note here that, usually for all of this bit we require this full adder which has 3 inputs and 2 outputs but for the first unit there is only 2 inputs and still 2 outputs, one is sum and another is carry. But to make things generic we also assume that this first bit, whenever we are adding first bit then there is some carry in which could be because of some other circuit or there are some different applications we have already seen in our when we did binary addition and subtraction. So, for example it could be used for subtraction or it could be used to calculate two's complement etcetera.

So that is why we here assume that our addition bit, so this addition unit is 1 bit addition unit will have a full circuit which is 3 inputs and 2 outputs.

(Refer Slide Time: 3:26)



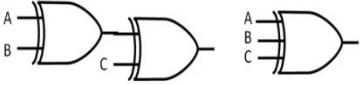
Single Bit Adder

A	B	C	Sum	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1


Sum = $A'B'C + A'BC' + AB'C' + ABC$
 $= A \oplus B \oplus C$

Cout = $AB + AC + BC$

Delay = max (2 XOR delay, AND-OR delay)



Digital Logic Design: Combinational Circuits. 60



Now, the same circuit let us put in a cascade manner and we see that how logically we can determine this are the logic for sum and C out. So, this also we have seen earlier that if the input is a b c and this is the truth table, this would be the sum, so that means when all 3 would be 0 then sum is 0 C out is 0. So, we have seen that sum is 1 when either one of a b c is 1 or all three of them is 1. So we have seen earlier the sum can be written as when one of the input is 1, so for example a b c, c is 1, b is 1 or a is 1 rest of them has to be 0 or all 3 of them is 1 then sum would

be 1 and we have also seen in our previous lectures that this can also be represented like a 3 input XOR gate.

Similarly, C out is equal to when at least 2 of them is equal to 1 ab or ac or bc , so at least 2 of them has to be more than 1 then the output would be, C out would be 1. So, this particular representation of sum we can do if we have, let us say we assume that we are not going to do it using 2 level gates but we want to use it, we want to double, we want to implement it using XOR gates.

So, there is 1 possible implementation where we implement it using 2 levels of XOR gates, first we XOR a and b then we XOR whatever the output of and b then we XOR a with c. Or we have 3 input XOR gate, which of that would be more beneficial or which would be more efficient? So, it looks like that this would be more efficient, where we are taking all the 3 inputs at same time.

So you see the implementation of this, the basic implementation would be something like this. There would be a 4 AND gates each of them will have 3 inputs and there would be 1 OR gate which will have 4 inputs. So, the delay of this 3 input XOR gate would be definitely more than 2 input XOR gate.


So, why people would prefer this? So, the only reason to prefer this implementation where we will have 2 level of XOR gates is that a and b both are actually primary input. Primary inputs means they would be available at time 0 but c depends on the previous adder. So, now you see that adder is a cascade of 1 bit adders, so the c would be coming from the previous adder.

So it would be, from the delay perspective it is more beneficial to have this kind of a combination because as soon as c will come then we will have to do only 2 input XOR gate. If we go for this approach then whenever c will come then this 3 input XOR gate would come into play and will have larger delay.

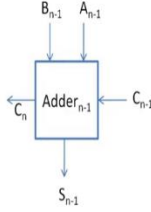
So, the delay for this would be 2 level circuit like basically delay of AND gate plus delay of OR gate. So, the overall delay of this particular adder we can say that either it is the maximum of this 2 level of XOR gates or either whatever is the maximum. So either the maximum is this 2 level XOR gates or it could be this AND and OR combination.

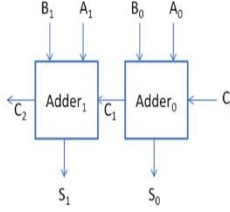
There is a more likelihood that 2 level of XOR gate will have higher delay than AND OR combination. So AND OR combination we can say will have lesser delay because the and number of inputs so and is only 2 and number of inputs so and is or is only 3.

(Refer Slide Time: 7:38)



Structure of N Bit Adder






$$S_0 = A_0 \oplus B_0 \oplus C_0$$

$$C_1 = A_0 \cdot B_0 + A_0 C_0 + B_0 C_0$$

$$S_1 = A_1 \oplus B_1 \oplus C_1$$

$$C_{i+1} = A_i \cdot B_i + A_i C_i + B_i C_i$$

Digital Logic Design: Combinational Circuits.
61


So, using this single bit adder now let us see that if we want to design an n bit adder so how can we do that? So, let us define this 1 bit adder like this where the 2 inputs A₀ and B₀ and carry in is a input here. It is generating an carry and there is a sum which is given as output. So, this single bit adder would, we have that this is how it would be implemented. So this would be cascading, so the carry input would be cascaded to the another single bit adder.

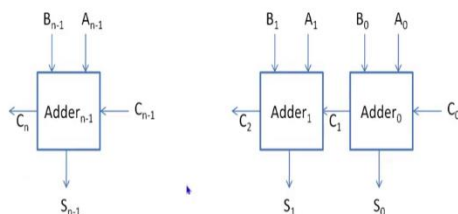
So, here I am putting the substitute 1 because the input a and b, so the first bit a and b and carry would be we will get carry from the previous adder 0 and sum 1 will be produced as well as carry to the second stage would be produced.

So this carry change will keep on propagating and we will have n such adders and in the end we will have this n minus 1 bit of a and minus 1 bit of b and the n minus 1 carry would be there and sum and minus 1 would be there and n carry would be generated. So generically for any i th adder we can say that sum is equal to a_i, XOR b_i, XOR c_i and carry to the next stage would be equal to a_i and b_i or a_i and c_i or b_i and c_i.

(Refer Slide Time: 9:14)



Delay and area analysis



Area: N (Two 2-input XOR gate + 3 two input AND + one Three input OR)
Delay N bit adder: $(N-1)$ (2 level Gate delay) + $\max(\text{XOR delay, AND-OR delay})$
 $\approx N$ (2 level Gate delay)

Ripple carry adder

Digital Logic Design: Combinational Circuits.

62



So, what would be the delay of this or what would be the area of this particular adder? Now, area we can say that whatever be the area of the single bit adder we can multiply it that with n , that would be the total area. So, essentially the area would be n into 2, 2 inputs XOR gates three 2 input and gates and one 3 input or gates. So, sum of all of these area into multiplied by n would be the total area of this adder.

What about delay? So, delay if you see that all of these adders would be able to do some part like $A_0 \text{ XOR } B_0$, so XOR of the primary inputs $a_i b_i$ can be done in parallel and then, now carry, the other sum part it depends on like carry of the second input depends on the carry of the first one.

Similarly, any n carry depends on n minus 1 carry. So, that is why the carry has to be propagated through all of them, so because we are assuming this as a primary input. So, whatever carry would be required to generate whatever delay would be required to generate the c_{n-1} that much amount of delay would be there.

So, I can say that n minus 1 into 2 level gate delay. So, I am assuming that let us say 2 input AND gate as well as 3 input OR gate are consuming let us say same delay. Then the total delay is 2 of this n delay plus OR delay that means 2 gate delay into n minus. So this n minus would be there because this we are assuming the C_0 is available at as a primary input and this to generate

C_{n-1} would require $n-1$ such delays and once C_{n-1} is there. Then the additional delay to this sum would be a maximum of XOR delay or AND OR delay.

So basically it would be, I can say this XOR delay is also a kind of 2 level logic. So we can approximately it like a n into 2 gate level delays or 2 gate delays. Let us define this 1 delay of 1 particular gate either AND or OR as Δ , so it would be $2N\Delta$. So that is why the delay here depends on the carry propagation and carry is rippling through all the time. So, basically first C_1 is generated then C_2 is generated then C_3 then C_4 then C_N . So that is why this particular adder is also called ripple carry adder.

So, this ripple carry adder has also couple of more interesting characteristics that your total sum S_0 to S_{n-1} so this sum will not be stable until this complete delay is there. So till that your output will keep on fluctuating sometime because as soon as 1 particular carry will go to the 1 stage to another stage it will change the input to certain other adders. So for example initially this C_{n-1} is definitely going to be 0.

Now, as it is 0 then S_{n-1} will give certain different output but because of some other propagation this C_{n-1} will keep on changing, so my S_{n-1} will keep on changing until the total delay which is $2N\Delta$ would be there, then it would be stable.