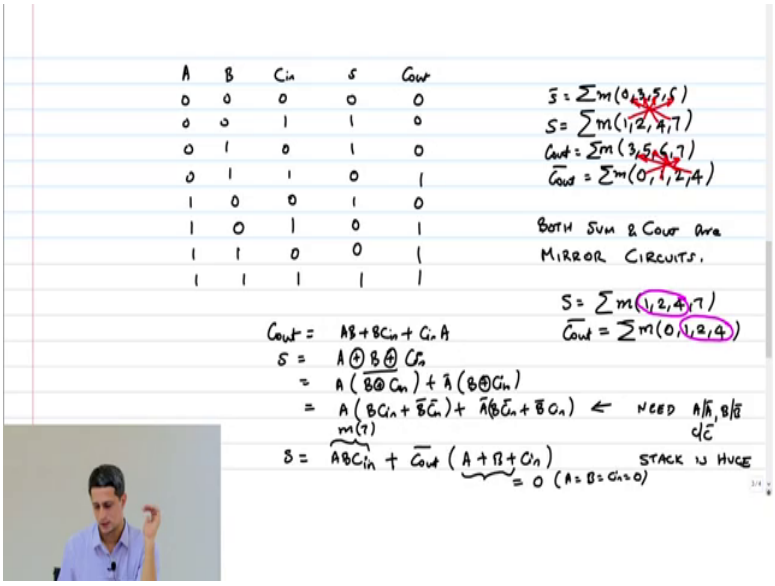
Digital IC Design
Prof. Janakiraman Viraraghavan
Department of Electrical Engineering
Indian Institute of Technology, Madras

Lecture – 55
Full Adder Circuit Implementation

(Refer Slide Time: 00:17)



So, now let us go back again to our truth table A B C in sum carry out; 0 0 0, 0 1 0, 0 1 1, 1 0 0, 1 1 1 sorry, 1 0 1, 1 0 1 1, ok. So, this is 0 1 1 0 1 0 0 1, 0 0 0 1 1 sorry, 0 1 1 1. So, what is my sum circuit in terms of the mean terms? It is summation of mean terms what? 1, 2, 4 and 7, ok. C out is what? Summation of mean terms 3, 5, 6, 7. Now, what about S bar? Ok, let me write S bar here, this is what summation of mean terms.

Student: (Refer Time: 01:38).

0, 3, 5, 6 C out bar summation of mean terms.

Student: 0, (Refer Time: 01:53).

C out bar 0, 1, 2, 4 ok. Now clearly, we this is something we discussed earlier this carry out circuit has the mirroring property, if I invert the inputs C out also inverts itself, right. So, 0 1 1 becomes this and this happens if I invert 1 you will get 6, clear there is no doubt about it. What about the some part, if I invert 1 what do I get; 0 0 1 you invert what you will get?

Student: 6.

6, right. So, this also seems to be inverting 0 will invert to 7; what about 3 and 4 if I invert 3 0 1 1 I get 1 0 0. So, that also is inverting 2 and 5, 0 1 0 1 0 1 right. So, clearly conclusion is both sum and carry out are mirror circuits, yes.

Student: (Refer Time: 03:23).

It is not always true.

Student: (Refer Time: 03:31).

Yeah, the necessary condition is there has to be equal number of 1s and 0s.

Student: When (Refer Time: 03:40).

Correct.

Student: We get this (Refer Time: 03:46).

No, no why if I do just 0 to 7 if let us say I have a 3 bit input 0 to 4 or 0s and 1 to 4 to 4 I mean 0 to 3 are 0s 4 to 7 are ones. So, why should this always be 2 that is what I do not understand.

Student:(Refer Time: 04:20).

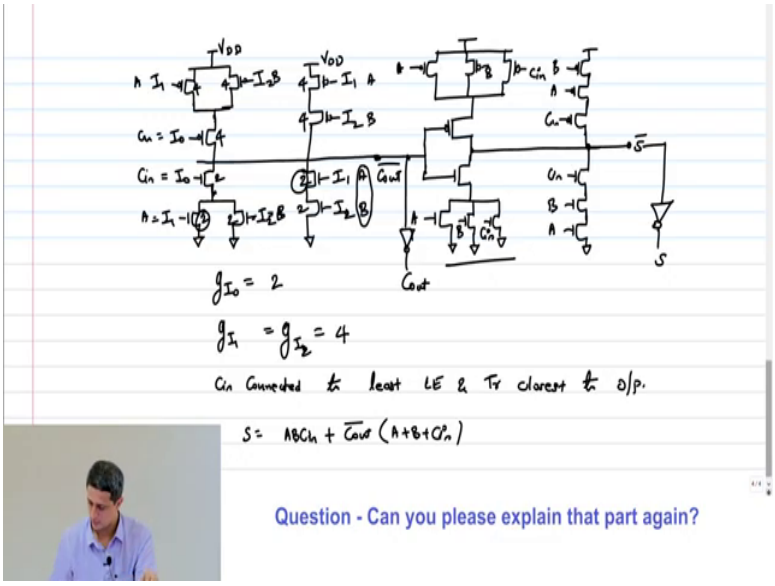
Input yeah, yeah.

Student: You get 1.

Correct; so, simple thing to do is to if I want to just counter your example, I will negate 1 of the inputs 0 0 0 I will get 7 I will make sure these 2 in outputs are the same my truth table then it will not hold, right. I am just saying that it is not always true only a subset of cases this is true, then necessary condition is yes I need to have equal number of 1s and 0s in my output if there are eight combinations 4 have to be ones, 4 have to be zeros only then that immersion will happen and mirroring can happen, but it is not always true, ok.

Here you just turns out that x or because the sum is an x or right and the carry out is also of the form AB plus BC plus CA it happens that the mirroring property indeed holds, ok. So, what is my carry out expression now, AB plus BC C in plus C in into A, correct.

(Refer Slide Time: 05:47)



So, we discussed this in the last module itself. So, I will go to straight to the circuit and just implement it in 1 shot. This is A we will come to that, right. So this by virtue of mirroring property I want to simply mirror the NMOS network and I will get exactly the same stack on top otherwise, you will see that the PMOS stack is going to go up by 1 number, and it will make your logical effort very bad ok, right.

And what are the sizes now; obviously, this is just 2 2 2 2 4 4 4 4 right, now I have to assign the inputs right because it is all interchangeable AB plus BC plus CA right, I could assign any input anywhere and logically they are all the same.

So, how do I assign the inputs here? How do I fix A B and C to the transistor inputs? Yeah, first tell me the logical effort of let us say this is my input I naught I 1 I 2, ok. So, this is also I naught I sorry I 1 I 2; I 1 I 2 I 2 what is the logical effort of I naught, g I naught how much?.

Student: 2.

2 right, 2 plus 4 by 3 what is the logical effort of I 1 I 2? Yeah, g I 2 equals how much, 4. Because I 1 is now connected to this two it is also connected to this 2 yeah 4 plus 4 plus 4 12, right; 12 by 3 and you get 4, clearly one input is going to be faster than the other input right, therefore, now amongst A B and C in which will you assign where?

Student: C in (Refer Time: 08:43).

C in has to be assigned to.

Student: I naught.

I naught right, this has to be CA, this can be AB does not matter AB it can be the order there does not matter and; obviously, this is A and B A and B, note also we have assigned C in to the transistor that is closes to the output. So, think about it when let us say that this full adder is being used in stage k of the ripple adder, A and B have already come in at times 0 and would have discharged this intermediate node either this or this node.

Now, the carry in its just waiting for the carry in to come and finish its work right; of course, if the carry had been generated then what would a happened this stack would have taken over and settled out to 0 already if g was 1 for that k th stage then it that carry out would have already got set by this parallel stack here, right.

So, we have also accounted for the fact that the signal coming last is connected to the transistor that is closest to the output there by having to discharge list number of capacitors in

the circuit, ok. So, maybe that is not the right place to write C in connected to least logical effort and transistor closest to output, ok.

Now, let us come to the sum generation part what is sum expression A x or B x or C in. Now, can you expand this and if you were to implement this in static CMOA logic in terms of A B and C in, how complex do you think the circuit would be um? What is the let us I mean just as an exercise let us first write out this expression right, it is A times B x nor C in right or B x or C in bar plus a bar times B x or C in. A times B C in plus B bar C in bar plus A bar times B C in bar plus B bar C in; yeah, anyone had a problem here have a written something wrong, it is correct know, yeah.

So, clearly you see that you know there is A into BC in plus you know; so, what will be the PMOS stack size, what will be the NMOS stack size. What will be the NMOS stack size?

Student: 3.

3 right, because there is just ABC type of terms; PMOS stack size yeah, it is basically going to be a very complex circuit and therefore, it is not and also you need to have A and A bar now, only then I can even implement that static CMOA logic correctly, right. So, therefore, it is not even why is to go this way you need A A bar B B bar and so on C C bar and also stack is huge.

So, now, we come back to our earlier discussion where we said it is for the sum to be generated a little later than the carry out, while the sum is being generated the carryout can propagate no problem. So, you actually have time to generate the sum after the carry out has been generated and propagated, right. What we are going to do is we are going to see if we can use this carry out that we created in order to create the sum one; so, that we can simplify our expression.

So, now, let us look at the expression for sum and carry out bar ok, the mean term expression sum is equal to summation m of 1, 2, 4, 7 and C out bar is what summation of m 0, 1, 2, and 4. So clearly, the carry out bar is equal to the sum in 3 out of 4 cases, right. So, we have to

just now take the C out bar that was generated by the way right and remember this expression that we got here is what C out bar, ok. So, we can take that C out bar and use it as an input to our sum circuit if I can express the sum in terms out of the C out bar correctly.

So, it turns out that the sum is basically the 1, 2, 4 is common right, these are common no problem, if it is any of these cases then carry sum is equal to carry out bar, no issue only in the other cases I have to create for example, when the input is 1, 1, 1 my carry out bar will be 0, but I need a thing for that. So, therefore, I have to add that mean term into my thing, ok. So, what I am going to do is I can say that sum is A, B, C in right this is the mean term 7, m of 7, plus the not S, C out bar ok; suppose, I say this S is equal to A B C in plus C out bar, we will this expression be correct, why?

Student: (Refer Time: 15:57).

Which case is it missing, the case I mean team 0. So, when the input is 0 0 0 I have to kill C out bar right; when the input is 0 0 0 I just have to kill the C out bar then I will get my in sum to be the right answer. So, this I will simply multiply by A A bar no, A plus B no, I think no A bar B bar C in bar, right. So, now, what happens is if the input is 0 0 0 then this term equal to 0 0 oh no, no.

Student: (Refer Time: 16:51) now.

It should just be ABC.

Student: (Refer Time: 16:54).

Yeah, if all 3 are 0 then it should be 0 your right, A plus B plus CA yeah, if all A B and C in r 0 then this will be equal to 0 for A equal to B equal to C in equal to 0, then C out bar is also killed you get the answer to be 0. In the remaining cases, in the remaining cases A plus B plus C in will be 1 because at least 1 bit will be 1 and therefore, the answer will be C out bar right

and except that last case unfortunately C out bar is wrong in the case of 7 mean term 7 and therefore, that mean term alone I am going to add here, correct.

So therefore, now I can express this guy my sum let me write that here is A B C in plus C out bar into A plus B plus C in, correct. So, all I have to do is take this circuit and now again remember on the sum I can do mirroring, mirroring is still possible. So, therefore, I am simply going to generate this circuit like this, right.

So, this is A B C in and now I am going to apply my mirroring property here and just mirror that NMOS stack on the PMOS as well, right. A B C in and I have to create that last case which is basically A B into C in right therefore, that is nothing, but down here, A A B C in this is S bar. In order to now generate the carry out and the sum you basically have to take out these nodes and add an inverter to it you will get the C out and the sum. In this portion tell me why I did not connect C in to the transistor closes to the output.

Student: (Refer Time: 20:20).

Sorry.

Student: (Refer Time: 20:28).

No, it is not only affect the logic itself will change, I cannot even do that. See you can do all this transistor reassignment if it is equivalent to connect any input to any transistor, AB plus BC plus CA was like that, otherwise I cannot do it. So, C out bar is this thing and additionally C out bar will come after C in now because C in comes C out bar gets generated.

So, C out bar is the signal that is actually arriving last. So, that is going to the transistor closest to the output anyway ok, any questions here, yeah.

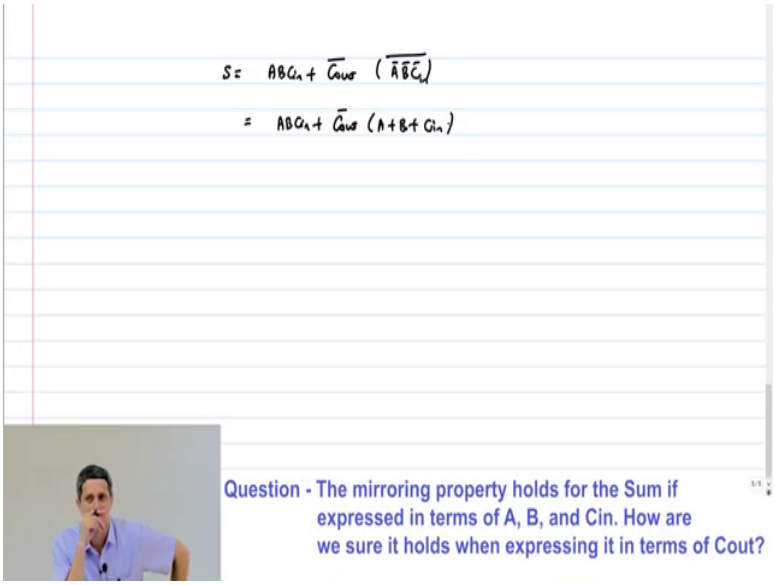
Student: (Refer Time: 21:18).

Student: (Refer Time: 21:19).

Yeah, see basically what we said is the sum is a summation of mean terms of what 1 2 4 and 7 right, the C out bar is summation of mean term 0 1 2 4. So, clearly the 1 2 4 part is the same. So, I can reuse C out bar to generate the sum, ok.

In 3 3 out of 4 cases 1 2 4, now in order to take care of the case of 7 right, I need to add that mean term in what is the mean term for 7 ABC right, ABC in. So, I say it is ABC in plus C out bar, but C out bar will give you the wrong answer for mean term 0, right. So, I have to ensure because C out bar equal to 1 when the mean term is 0 that is 0 0 0 I get a carry out bar of 1, but my sum should be 0. So, in that case I have to kill the carry out bar, how do I kill the carry out bar it is basically I will tell you what we are doing.

(Refer Slide Time: 22:41)



Sum is basically A B C in plus C out, this will work in 7 out of 8 cases for the case when a in A equal to B equal to C in equals to 0 this will be wrong, because C sorry this is C out bar because C out bar is 1, but sum should be 0, right. So, therefore, what do I do I just take it and actually it should be this right, this is my mean term 0 A bar B bar C bar is my thing I have to multiplied by that compliment, I am just killing it mean term A bar B bar C bar is will go high when all are 0.

So, if I compliment it will go exactly low during that time and it will kill the carry out bar. So, this is nothing, but A B C in plus C out bar into A plus B plus C in, ok. Any other questions, yeah.

Student: (Refer Time: 23:59).

How I used.

Student: (Refer Time: 24:06).

Yeah.

Student: (Refer Time: 24:09) u at get it.

Student: C (Refer Time: 24:13).

That is, so, that is a very good question you are asking me if I express a logic function in terms of some other variables will this mirroring property always hold.

Student: It is the to it we need to write (Refer Time: 24:31) and (Refer Time: 24:35).

Correct.

Student: (Refer Time: 24:37).

Right. So, I urge you to go back and try it ok, enumerate these inputs, but I will tell you what the catches. How many combinations do I have now A B C in and how many combinations do I have that is not true because C out bar itself is a function of A B and C. And therefore, only a subset of those combinations will work will actually happen in reality and this mirroring should hold only for that subset that is how you get in to work.

You followed this question right, his question is I said that the mirroring property I gave a hand waving statement that here I said that the mirroring property holds by showing that the sum you know 1 goes to 1 goes to 6, 0 goes to 7, 2 3 3 goes to 4, 2 goes to 5; question is how do you know that the mirroring holds when C out bar is an input to this function?

So, now I actually have to so, mirroring holds here; see, mirroring will hold in this expression without any problem because that is what we showed there, question is how will it hold when we do it here how do I know it holds here. So, his point is you should enumerate the cases and show that the mirroring holds even for this condition, but now C out bar is an additional input.

So, how many inputs do you have you have A B C in and C out bar, but C out bar is now depending on A B and C in d. So, it is not that it hold due to enumerate this for 16 combinations, it is only a subset of combinations because when A B C in equal to 0 0 0, C out bar is 1. You have to enumerate and see that the mirroring holds only for those subset of combinations. So, go back enumerate these cases, find out how many cases there are and see if the mirroring holds or not, ok.

Student: (Refer Time: 26:55) whether take 0 bar is already a mirror.

Yeah.

Student: So, if the input invert A B C.

Yeah.

Student: 0 bar or not you get it (Refer Time: 27:02) same as take that if the input invert ABC C o dot (Refer Time: 27:06).

Correct.

Student: So, the mirroring will hold.

Again, you have to show that for example, you take this expression here right, S equal to A B C in plus in C out bar into A plus B plus C. So, now, I say that ok, this is S bar is this you know S bar is this whole bar then you write the Boolean insight right, then you can say that when the C out becomes C out bar, you can say that I can invert if my inputs invert. But still you have to show that the mirroring holds is all I am saying, you either have to enumerate true table or you have to show it here, clear.

Student: (Refer Time: 27:50).

Yeah.

Student: (Refer Time: 27:54).

I know, but it is we have not proved anything like that, right. So, you have to be careful where is all he was pointing out and he is right, he want to show that it works for that, ok. Because it is true for example, let me tell I if I give you 4 independent inputs A B C D and said AB plus ABC plus B into A plus B plus C. mirroring does not hold there you cannot do it. It holds only because it is a subset of inputs on which it is going to work. So, therefore, it is you have to show that this still works, ok.