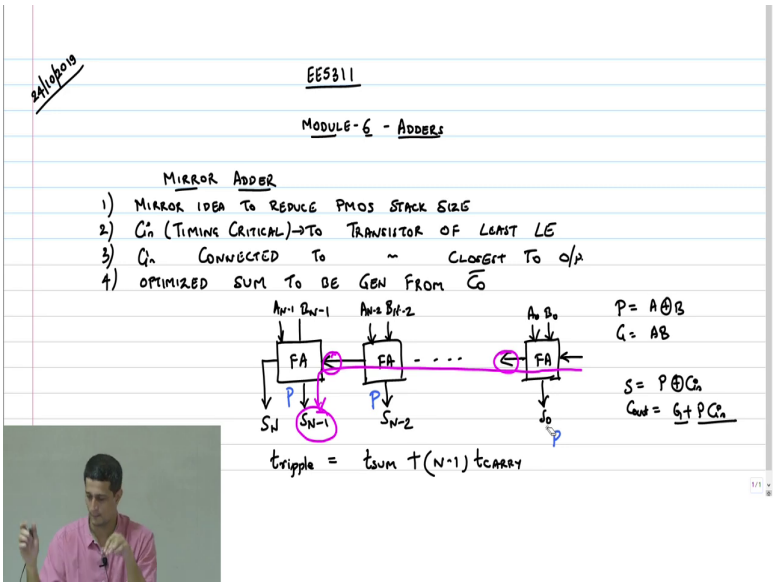
Digital IC Design
Prof. Janakiraman Viraraghavan
Department of Electrical Engineering
Indian Institute of Technology, Madras

Lecture – 56
Full Adder Optimization

So, yeah good morning and let us proceed with our discussion on the Adders, right.

(Refer Slide Time: 00:23)



So, yesterday we looked at the mirror Adder right and where we sort of applied all the concepts we learnt in module 4 in order to design this adder very effectively, ok. And what are the things we did? We used the mirror principle, right mirror idea to reduce PMOS stack size right.

Then we assigned tcarry which is the timing critical Single, right, C in which is timing critical to transistor of least logical effort right, then it also so happened that the transistor with these logical effort was the transistor that was connected closest to the output. So, C in right connected to transistor closest to output right and then, we optimized sum circuit to be generated; generated from C out or C o out bar right, otherwise the three input xor implementation is going to be very very complex.

Instead of that we are able to reuse the C o out bar and why was that because we showed that in the ripple adder, the C out is; the C out is actually the signals that on the critical part only for that last adder the sum is on the critical path, right. So, if you remember that we said that when I cascade full adders like this in a ripple adder form, right.

So, this is my C in A 0, B 0, S 0, A n minus 2 B n minus 2 S n minus 2 S n minus 1. Again A n minus 1, B n minus 1 and this is s n. So, it turns out that only for the last adder is the sum actually going to be on the critical path. For all the remaining n minus 1 adders on that ripple chain it is a carry out that is going to be on the critical path, right. So, this is the critical path.

So, if I want to optimize this path, then I have to make sure carry out is fast, the sum can be slow and that is what we saw when we wrote out the delay expression, right. Ripple adder we said was t sum for the last guy plus n minus 1 into t carry, right. That is why we chose to optimize t carry so much rather than sum, ok.

So, with this we got a very nice circuit implementation in static CMOS logic, right that you could you know have good drive strength and all those stuffs. So, there is no problem. Today we will see a few more may be alternate circuit implementations just briefly. I am not going to go into details. I will just show you a circuit may be a quick analysis and I will just tell you what the key advantage of that circuit is, right.

So, before we go in to that let us also figure out when we mark this critical path, in what mode should each of the n adders be? It should it be in generate, propagate or delete mode. So, let us start with the bit zero, should it be in generate or propagate or delete mode?

Student: Delete

Should it be in delete mode? What is delete Mode mean? It means a carry out to be set to zero does not matter. What the C in is will that be the worst case? No, right. So, what mode should it be?

Student: Propogate.

Yeah propagate or it could also be in generate because both to propagate C in it is going to take one unit of time to generate the carry, also it is going to take one unit of time. So, it really does not matter there.

So, this I would say can be in G or P mode, no problem. What about the other n minus 2 adders; N minus 1 adders? Actually everything has to be in propagate mode, ok. This is important because when we go to the other architectures of adders, this thing will slightly change. You have to be very careful. Once you carefully identify which adder is in which mode, then it is very easy to figure out the critical path, ok.

So, what we are trying to do is for the worst case delay what mode of operation should each adder be in? This is what you have to first identify, then it is very trivial to write out the expression of the delay, yeah.

Student: Generate mode.

It could be generate because see generate is let us say the carry is being generated or the carry is being propagated, both of them take one unit of time. See to generate you have to do that A and B operations, right that is a Logic Gate. The carry propagate is also like a gate, right. So, may be since you asked, so what was the expressions for propagate A x or B generate AB? Now, can you write the sum and carry in terms of this in terms of P C in and G. The sum is what?

Student: P x or C a.

P x or C a correct. A x or B x or C in A x or B is P therefore, it is P x or C in what about carry out.

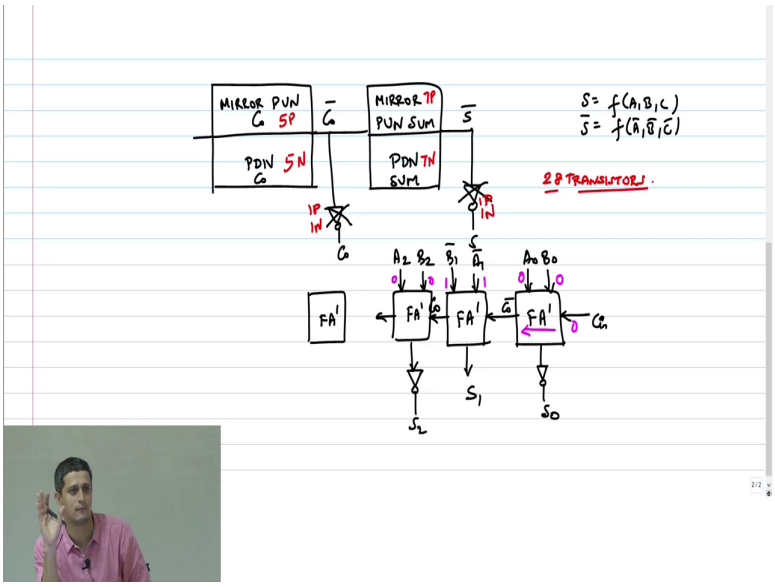
Student: A plus B.

Yes either the carry is going to be propagated which is controlled by the signal P or it will be generated. The delete you do not have to take care because delete is a case when both P and propagate are P and G are 0, output will automatically be 0. So, this is nothing, but G plus P into C in.

So, to answer your question basically to the for the C out to be ready, either you have to do this operation P into C in or you have to do this operation. Right now we are saying assuming these two operations are of similar time, it does not matter if it is generate or propagate.

Of course generation of P itself is an A x or B operation, then you have do the and with C in. So, technically that will take slightly longer, but assuming these two delays are reasonably comparable it is G or P. What I am saying otherwise it is fine to even say that this has to be P right, but the others cannot be in generate mode, that is all I am saying ok.

(Refer Slide Time: 09:12)



So, we can it turns out that we can do one more optimization you know to do this. So, if you remember yesterday we said that you know we wrote out the mirror adder circuit, right. Let me just write that here this is basically the pull down network, this is the mirror pull up network. This gave me C out bar, then we said we have to put an inverter here and get C out. Now, similarly we did something for the sum right. We had a pull down network for sum and mirror pull out network for sum, this is C out and we got S bar and we have to generate S.

Now, look at what happens when I cascade this is ok. This is full adder I am going to put it in the ripple form, ok. So, what is happening is in order to generate C out here I have to go through to that extra inverter right and then feed it into my next inverter. So, if you look at the number of inverters on this path, we are going to have one inverter per full adder, right.

So, you are effectively adding n more inverters on the path and you are going to increase the delay, ok. The difference remember is because this is like an array structure. I cannot do gate sizing like algorithm where I make this full adder, very small next full adder large you know, I cannot go like that. It is better to have a nice array structure especially when you go to multipliers and all that. It is better to keep the lay out same.

So, it is effectively same full adder that will get repeated. So, it is not like I can use those inverters as buffers and speed up that path, it is a same inverter, right. So, effectively I am going to add n inverters in to this path, and the problem is that is going to increase my delay. What I want to do is, I want to get rid of this completely right or at least it is, ok.

I do not mind the sum part. I want to get rid off the carry out delay because that is what going to go up as n minus one times delay on the critical path, right. So, question is how do I eliminate this? Any any suggestions? You use the complement idea in the sense we already know that the mirror property was used here because if you complimented the inputs, the output of this full adder will also complement itself, right.

So, we said that S is equal to f of A B C, S bar is equal to basically f of A bar B bar C bar, that is why we could do this mirroring technique in the first place. So, which means that if I want to give the complement inputs all I have to do is, I have to complement all the inputs, then the output will also get complemented, no problem right. So, what you have to do is, here is the first thing you make another full adder block FA prime which is basically the full adder without those two inverters. No inverters there you put inverters later to get this sum S0.

Now, this is my C in. So, I will leave this as A naught B naught, ok. I have removed the inverter now. So, what do I get here? C out bar. Now, C out bar is going to the next ripple adder. What should the other inputs be? Exactly Right A 1 bar B 1 bar.

What will you get here? You will now get S correctly because I have removed the inverter from that FA prime is this circuit without those two inverters. So, you will get S1. What

about the carry out? Again it is C out, right. So, the next stage you can basically put FA prime. Again give the right inputs A 2 B 2 and what you will get here is again you have to invert S 2 and you do this yeah that is a very good question.

So, the point is yes for a A 1 B 1 and all that I am using is another inverter and increasing the delay, but remember that all the signals arrive together. So, every alternate stage I am going to invert all those inversion going to happened in one shot. So, if you look at the arrival time of these signals, this will arrive at 0, this will arrive at 1, this will arrive at 0, the next A3 B3 will also arrive at 1.

So, in some sense by the time this full adder has evaluated and propagated the carry, this guy has already inverted the inputs, clear. So, this is another technique which you can use in order to optimize the delay and it is very useful in specifically in the multipliers. I will show you that when you come there. Clear? Any questions here?

Student: No.

So, with that we have you know done a lot of optimization. How many transistors were there by the way in this full adder circuit that we made, yeah?

Student: Total.

Totally how many transistors in the mirror in the pull down network of the carry out there were, how many transistors? 5, right. So, I will put that 5 transistors here, 5 N MOS, here therefore, this is also 5 PMOS. What about the sum?

Student: 7.

7 NMOS; 7 PMOS assuming that I also had to generate the inverted I mean the actual C out and sum I would need one NMOS here, one PMOS here, one NMOS here, one PMOS here. So, total number of transistors is how many? 14 plus 10 plus 4, right. 28 transistors.

Student: Yes.

And by the way earlier I also you know when I gave you the assignment, I asked you to the carry out with the carry out circuits, upsize the little bit. The reason for that is again what I just told you the full adder is just repeated n times. It is not a thing.

So, if I have all unit sized devices with this load, it may not be able to drive it. So, it is not that I can optimize that sizing and figure out what the correct sizing is. So, as a thumb rule you upsize it up by a factor of 3 or 4 and then repeat that adder your delay will be ok. This is your empirical thing that has been found and it is reasonably work. That is why I asked you to do up size the carry out circuit alone, sum I do not have to do it because it appears only once in the critical path delay term. So, the sum is left as a as a regular circuit in the sense the sizing just the usual sizing that we get, ok.