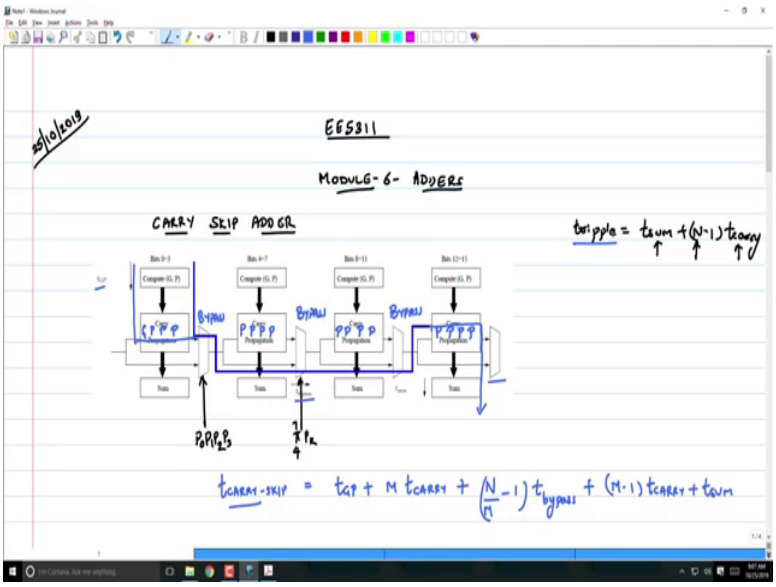
Digital IC Design
Prof. Janakiraman Viraraghavan
Department of Electrical Engineering
Indian Institute of Technology, Madras

Lecture – 58
Carry Select Adder

(Refer Slide Time: 00:15)



So, yesterday we started discussing about various architectures of adders which is the only way in which you can be speed up be performance of an adder right beyond a point. You can do circuit techniques to speed up little bit, but beyond the point the ripple delay right is t sum plus N minus 1 t carry right and if this is the case then it is delay is determined primarily by N.

So, it does not matter how much you can you know decrease the t sum and t carry at a circuit level. So, it is you are limited by N and therefore, you have to start thinking of various

architectures of ripple adders and in that pursuit we discuss the first adder called a the carry skip adder right. And, the idea of a carry skip adder was to simply bypass the carry bypass all the full adders in a single stage if all the propagates are 1.

So, we basically said that we will put a select line here called P 0, P 1, P 2, P 3 right this select line is basically product 4 to 7 P K right like that you just put a multiplexer and the worst case delay which is the case when everything when all the adders are in propagate mode, you by pass through the multiplexer right. So, with this what we are able to do is the critical path right.

Actually this blue line is slightly wrong, it has to go through the carry propagation it has to be like this right. So, you do a generation of G and P terms for the all the adders in time t GP right and then you propagate through the first stage. However, in the first stage you do not have to wait for sum to be ready. So, you just have to allow the carry is to propagate through that. So, which means that the first adder has to be in generate, the remaining have to be in propagate mode right.

Similarly, but after that everything has to be in propagate mode right and therefore, we got the expression for the rip carry skip adder, as what? T GP plus M times t carry. Now, you have to bypass N by M minus 1 multiplexers right. So, if this delay here the multiplexer delay is shown here as t by pass right then you have to do N by M minus 1 t bypass. Why M by N minus 1 because if this stage also as going to bypass then the carry will be ready.

However, that does not mean the sum is ready. Ultimately we want all the sum bits and the carry out bit to, but ready only then our?

Student: (Refer Time: 04:08).

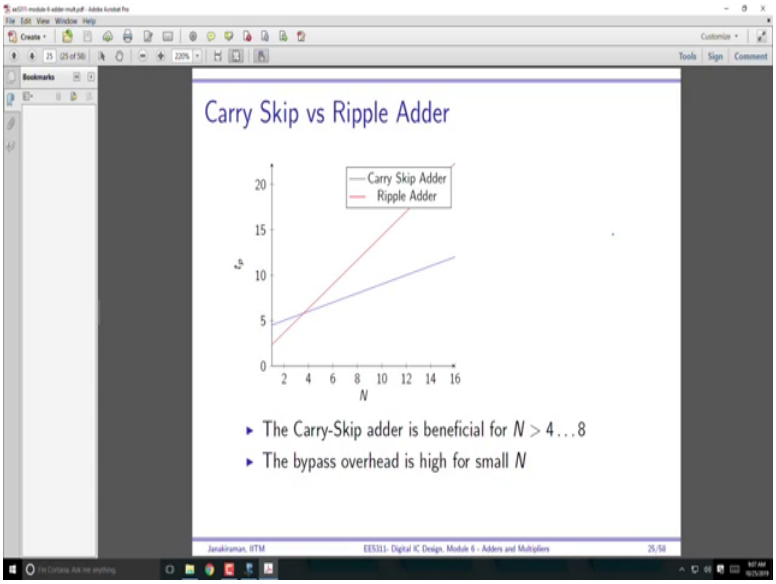
Computation is done right. So, therefore, the last stage cannot be in bypass mode right. So, this is going to be in bypass right, this will be in bypass, but this is not going to be in the bypass mode I mean that is not the delay. It can be in bypass mode, but we have to now consider the delay going into the sum here or in fact, I need to draw this on that side

propagate and bring it out here right. So, therefore, we said with the last stage is going to be what M minus 1 t carry plus t sum ok.

So, now how do this compare this delay term how does it compare with the ripple adder term right. Clearly what we have done is wherever there was N we have brought that down to N by M. So, clearly there is sum saving right, but definitely and quite obviously, it cannot give you savings if you have very few bits because now, I have actually added a multiplexer in the critical path.

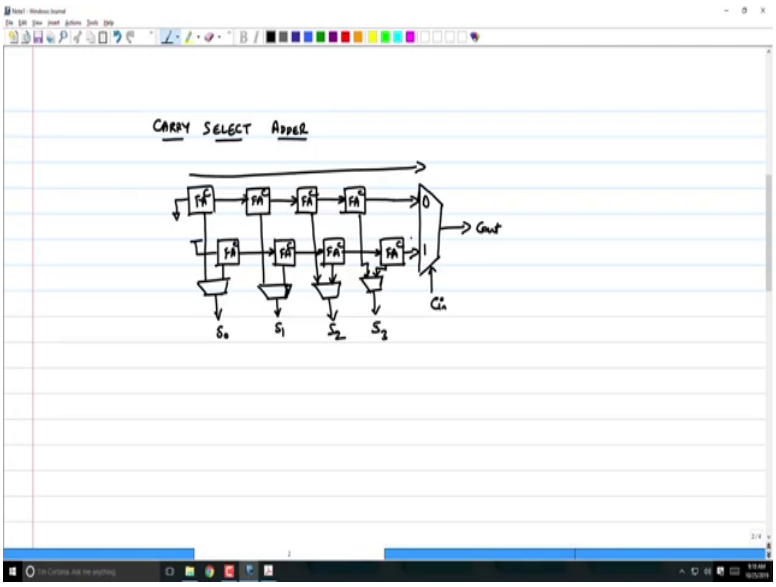
The ripple adder did not have the multiplexer, now I have put a multiplexer. So, it is bound to give you larger delay if there are fewer bits, but as you go higher it looks like beyond about 4 bits also the carry skip adder starts giving you good advantage ok.

(Refer Slide Time: 06:00)



This is what shown here. Ripple adder is basically the line in red right. It starts off with lower delay up to about 4 bits somewhere around 4 bits. Then the and up to 4 bits the carry skip adder has slightly higher delay. Beyond that the carry skip adder starts having better delay than the ripple adder ok. So, in summary we are still limited by we rippling though we brought in down to N by M ok. So, let see how we can improve on this further ok.

(Refer Slide Time: 06:58)



So, now you have let me get the name right I forgetting this name carry selector adder code. So, we would do a carry select adder. So, in the last stage of this carry skip adder right here we had to wait for the carry to actually ripple through and then produce the sum outputs. Can we do better than that? Right, can we avoid that rippling is the question ok.

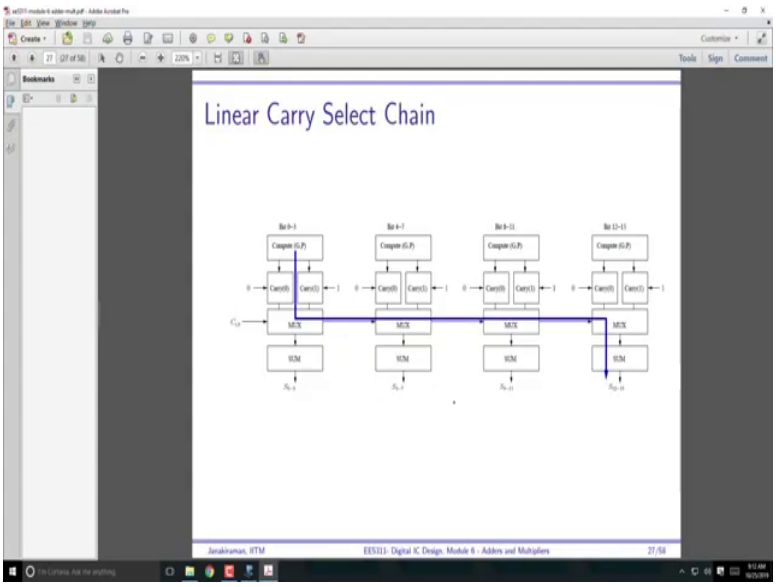
So, if that were the case how we do it? So, the point is very simple the carry in that is going to come to this final stage here this C in right is going to be either 0 or 1 right it is a bit.

Therefore, you double the hardware and evaluate the sum and carry for both C in equal to 0 and C in equal to 1 and keep it ready. The aim here is to improve performance, my power can go up, my area can go up. There is no problem.

If area is goes up power will go up because width goes up and therefore, capacitance goes up dynamic power C V DD square will go up right, but it is ultimately I want performance. So, the idea is instead of just bypassing you make a full adder ripple adder in that each block like and this is a full adder, carry in will be grounded in one stage. This will go like this, then I have another full adder right maybe I should put it like this.

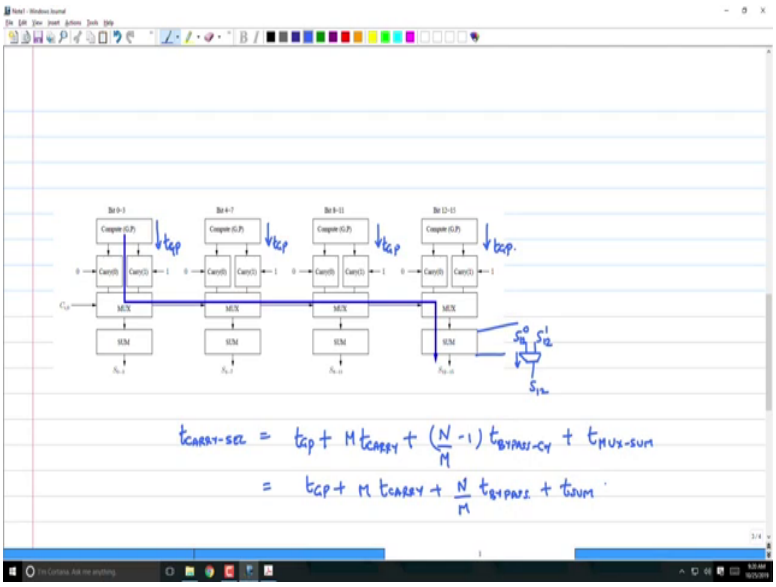
Here you connect it to V DD and then as usual I am going to put a mux. What is my select line? C in of this stage. This is 0, this is 1 we will get the C out right. Now, I also have to multiplex here and get my sum S 0, S 1, S 2 and S 3. So, now I have significantly increased the hardware I will double the number of full adders have also added more multiplexers right, but the advantage is that stage can actually evaluate that bit and be ready up front ok.

(Refer Slide Time: 10:54)



So, now we are going to use this and construct the thing there. So, in interest of time let me just sort of ok.

(Refer Slide Time: 11:15)



So, as usual we have a computing the G and P block first right. So, this is going to give you let me use blue there, the delay here is as usual t GP. Then it goes to the full adder rippling block that does evaluation of carry in equal to 0. In parallel there is another block which does the evaluation when carry in is equal to one then I am going to send it through a multiplexer and the carry out will go.

The sum of course, will have its own multiplexer and all that right. So, now, what is the first what mode of operation should each of these full adders be in?

Student: (Refer Time: 12:33).

Sorry.

Student: (Refer Time: 12:40).

It can be?

Student: (Refer Time: 12:45).

It can technically it can be anything because we are evaluating this for both cases right. So, what is the critical path delay now right t?

Student: (Refer Time: 13:02).

Select obviously, it has to take a time t GP, of course this is in parallel right. So, this is t GP plus what about the first stage?

Student: (Refer Time: 12:33).

Do you have to wait for it ripple through that or not?

Student: (Refer Time: 13:35).

See till now up to this point we only generated the GP signals you have created a GP signals and only one small amount of time is elapsed, in that time do you think this ripple adder could have ripple through already no right. So, technically you have to have that rippling thing to go through correct. So, I expect we will check let us just write it these things and then we will go back and see if the expression matches if not we see why. M times t carry right. Now, how many multiplexers do I have to bypass?

Student: (Refer Time: 14:23).

At least M by N minus 1 like the last case right. Only thing is what about that last multiplexer?

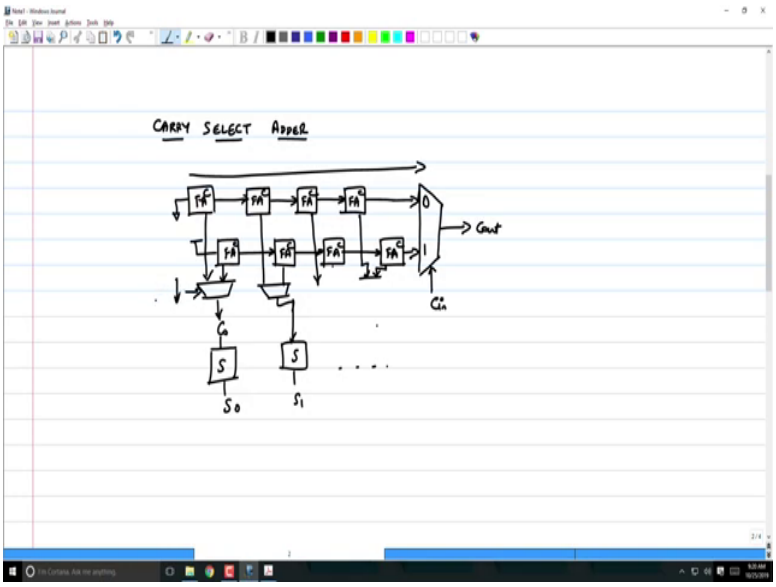
Student: (Refer Time: 14:49).

Correct. So, this is actually N by M minus 1 t bypass of carry, but this sum now you need to multiplex right you have to put the multiplexer there and get the sum output also multiplexed right which means if this blocks if you see there is actually similarly 2 is to 1 bypass mux blocks here that is going to select between the sums.

So, this will be S 12 right and this is going to come from the 0 block S 12 of 0, S 12 of 1. So, it going to. So, therefore, this bypass is plus t bypass or t mux of sum just to differentiate between bypass and sum multiplexer I will write it like this. Of course, if all this is the same then I have expect it to be t GP plus M t carry plus N by M into t bypass this was the expression right?

So, we can go and check if you know if I made a mistake N by M t mux plus there is a t sum ok this is interesting. So, now it depends on what this full adder block is ok. If my aim is to only propagate the carry right in this particular block if my aim is to propagate the carry then what I will do I will not put the full adder here I will only put the carry out circuit of the full adder there, correct? Because this path among this path we are only trying to propagate the carry the sum can be evaluated in parallel correct.

(Refer Slide Time: 17:43)



So, here I will call it FA C which means it has only the carry out part. What I will do is instead of this whole multiplexer thing, you are going to get the carry out here right and then I will create the sum block right. You remember when you made the full adder circuit in at a transistor level we use the carry out to create the sum.

You can now split that block you do not have to put that entire because you remember now you are doubling hardware. So, you double only the necessary hardware. So, here this sum block I am removing and I have only if one per full adder set right. So, I have reduced the hardware here, but of course, the sorry right the sum will then get created her S 0, S 1 and so on.

So, that is why after you bypass right what has happened this t by N the last bypass is actually through this multiplexer. Then I have to create the sum right I have that thing to create and therefore, I will add one more term to this called plus t sum ok.