Architectural Design of Digital Integrated Circuits
Prof. Indranil Hatai
School of VLSI Technology
Indian Institute of Engineering Science and Technology, Shibpur, Howrah

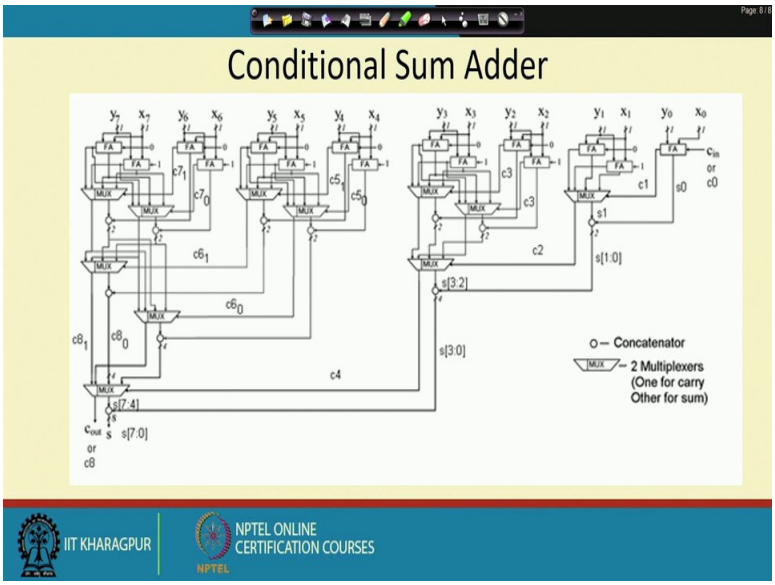
Lecture – 21
Efficient Adder Architecture (Contd.)

Hello everyone. Welcome back to the course on Architectural Design of ICS. So, in that last class, we are discussing about this carry select adder and from that carry select adder what we have that means, come to know that again we can that means, share the input and based on the; that means, predicting the carry input which will be nothing but that 0 and 1.

So, we will pre compute the values of this, using this full adder cell one for 0 considering this carry input bit of 0 and another considering for carry input bit of 1. So, we will pre compute that and but that means, whenever this the previous carry, previous stage carry based on that we will take the decision whether that previous k that means, stage carry, whether there is 0 or 1 based on that we will select, which input or that means, what combination that will be passed.

So, based on that, we have developed or we have; that means, tries to develop this conditional sum adder which is that means, the fastest adder among all these adders what we have that means, learnt till now. So, we are discussing about this conditional, 8 bit conditional sum adder.

(Refer Slide Time: 01:33)



So, what that means, what we were seeing that that in the last section. So, as there is this c in, this c in we know from the beginning. So, that is why actually, I need only one full adder cell for this LSB position which is x 0 and for addition adding this way x 0 with y 0.

So, whenever we are adding x 0 and y 0, at that time what I will get I will get one carry output from this particular full adder cell and the sum 0. Then for the y 1 and x 1, you see there are two full adders. So, 1 and these two inputs are shared among that; one actually full adder cell that computes the considering the carry input bit 0 to it and another full adder cell actually, computes the sum and carry based on this considering the carry input bit of 1. That means, whenever I am computing this particular cell, at the time I am also computing this two full adder cell as these inputs are already available to me, all the inputs.

So that means, for this particular y 1 and x 1, addition of y 1 and x 1, I will get these 2 full adder cell running in parallel considering 0 carry for one; 1 carry for another one. So, basically in addition in ripple carry addition or in any other addition what happens, the previous stage generated carry that effects the current stage k; that means, sum and the carry both.

So, here what we are doing? We are previously from the beginning we are; that means, calculating the values of considering as carry input bit that is of 1 bit that will be that can

and we are doing this binary addition. So that means, that cannot go beyond 0 or 1. So, that is why what we are doing we are pre computing these values considering the carry input bit to 0 and 1.

So, whenever I am getting at this previous stage; that means, carry is c 1, based on that we are taking the decision that which output from this 2 particular full adder cell that will pass. So, if the carry signal is 0, so at that time this full adder cell will pass these values. Otherwise, if this value is one then this full adder cell values will pass that is, values means this sum and carry will pass from this multiplexers.

So that means, now if I just this indicates the concatenation. So, from here now I will get this s 1. Here from this, I am getting the sum s 0 and from this particular multiplexer, I am getting sum 1. And another of that I will get the; that means, this stage generated carry which is this c 2.

Now, next go to the, that means, third bit. Now, for y 2 and x 2, so then again you consider this y 2 and x 2, y 2 considering 0 and y 1 considering sorry y 2 and x 2, two full adder cell 1, 1 considering 0 carry in and another considering one carry in right. Then again, for y 3 and x 3 again I have to do two that means, I have to use two full adder cell considering one of 0 another of 1.

So, then for these particular stage, these also can generate the carry. Here what is happening? For this only, as there is only one full adder cell. So, that can only generate one carry. But, here what is happening, I am getting 1 and 1. Suppose, I am getting for this 2, 1 and 1 and the carry from here that is 1, so that means, now I am adding 1, 1, 1, three 1's and then again for y 2 and x 2, if I am getting again 1 and 1, that will also produce another 1; that means, for this particular case, 2 carry will be come to this effecting these particular case.

So, that is why actually the previous stage carry, initially that will effect, so that means, one carry that will be generated from this particular stage and one carry will be generated from this particular case. So, then one multiplexer I will use for y 3 and x 3, which will considering this particular that means, 0; that means, carry and another one is for the if the that means, the for considering y 2 and x 2 of 1.

So, now we are doing this 2, two particular multiplexers for y 2 and x 2 which is just the previous carry of this y 3 and x 3. So, we have computed these two; that means, for multiplexer of these two. We have calculated for the previous stage carry, but again, another carry what I said that another carry can generate which is this c 2 that will also effect to this particular stage.

So, based on that, again these two multiplexer will passes through pass through another multiplexer where c 2 will take the decision; if this c 2 is 1, then this particular group will pass; otherwise if this is 1, then this group will pass. So that means, now I can generate the carry output bit and another input which is this 2 bit basically, what I am getting I am getting 2 bit here. Why 2 bit, one sum from this, another sum from this.

So that means, I have to; that means, as this is this sum is also concatenated and this sum is also concatenated with this particular sum. So that means, I will get two sums here and that will concatenate with the previous two and from here, I can calculate or I can compute this last 4 bit of sum. And this is the that means, corresponding carry out bit. Then again add this is for actually if I just talk here, this is for 4 bit conditional sum adder.

Now, if I just go to the that means, then what I actually my intention is to design that is 8 bit conditional sum adder. So, then again for y 4 and y 5; so, y 4 and y 5 again this kind of this kind of architecture bit; that means, the same logic will be applicable for y 4 and x 4, sorry y 4 and y 5 or x 5 or that means x 4.

So, this architecture then I will just copy here. Then, same thing will happen for this sixth position and 7 th position. So, if I say that this is the zeroth position, this is 1, then 2, then 3, then 4, then 5, then 6 and 7, so in a group, so this will effect this, this will effect this, this will effect this, this will effect this.

So, in the beginning we can that means, draw this particular circuit for this for even two of them and then from these for the odd one, will have 2 full adder cell along with these multiplexers. So, then the carry which is generated from this particular cell, what logic we have followed here; that means, this particular carry that will effect this. So, here this particular cells carry, that will again effect to this particular cell.

So, that is why I need this multiplexers, these two multiplexers for these two multiplexers. So, then again these two multiplexer again, that means actually where from that another carry will be generated; another carry which is that, that means, from this forth bit there is another carry. So, then that again that will take the final output or that will be effecting the carry output generated from this particular blocks.

So, we will take the final output which is nothing but this 4 bit for these particular 4; that means, considering this x 4 to; that means, x 7, I will get 4 sum bit here and this carry bit work 1 here. So, based on this particular this carry output bit this will select which of this that sum we will choose and how we are coming to this, that means, concluding to this particular point that which one will be selected? If you just go back and that means, reverse computation, if you do, you just see that based on this that, that means, the at the adjacent stage that effects the next stage and that next that means, previous to previous stage carry also that means, that also effects the current stage sum and carry computation.

So, based on this particular circuit now, we can calculate the corresponding sum which is. So, 4 bit I am getting from here and another 4 bit, I am getting from this particular block. So, if I concatenate both of that means, oh these two then I will get a 8bit sum and one carry output bit. So, this is the 8 bit conditional sum adder. So, if you see; that means, in these two particular case, so how what is the level of mux I need finally,, so, the level of mux which I need that is log to base if I consider that means, n.

So, for n bit conditional sum, how many that that how many this multiplexer cells that I require. That will be log to this n. So, if n is 8, so; that means, 3 level of multiplexer I require. So, if n equals to 16, so at that time I require what, I require 4 levels of multiplexes.

So, at that time what will happen; that means, if I want to that means, this circuit if I just want to; that means, convert from to the 16 bit conditional sum adder, so, at the time what will happen at the very these particular carry out the, the carry out which is; that means, generated from these particular cell so, that will effect the fourth multiplexer if I consider 16 bit conditional sum adder. This rest of the circuit will be; that means not will be like just like same.

So; that means, for this just next to next, it will follow something like this, then again like this and then again like this. So that means, 2, 3, 4 2, 3, 4, this again it will continue and finally, this carry will select the final output bit; that means, whenever I will compute these particular things as I am following this 3 level at the maximum that is MSB side; so, for the fourth 16 bit for y 15 and x 15 along with this y 14 and x 14, I will get 4 levels of multiplexer over here. So, here what is there; that means, this if I consider the this delay term. So, then what is that; that means, maximum delay over here? What is the maximum delay over here?

So, one of the part which is following that is from y 0 to the corresponding particular path. So, one path is this full adder, then one multiplexer, then 2 multiplexer, then 3 multiplexer, then again, if I choose this particular path then from this input, one full adder cell, then multiplexer, then multiplexer, then multiplexer.

So, if I from these particular cell if I just compute what is the maximum critical path one full adder cell, then this two, then again this is coming to that; that means, this is coming to this; that means, one full adder cell to one multiplexer, two multiplexer, three multiplexer. So that means, again I am following that the corresponding that we will this delay for this particular 8 bit conditional sum adder is one full adder cell delay plus 3; that means, 2 is to yes 2 is 2 1 multiplexer delay.

So, if I consider 16 bit, so at that time, one full adder delay and 4 multiplexer delay. That means, though I am increasing the number of; that means,, if I am increasing the number of bits but my delay that is not increasing that much. What I am increasing? Only I am for 32 bit there will be only that; that means the delay increment that will be just 1 multiplexer delay.

But the thing is that, at the time, I am getting the benefit in terms of delay, but at what cost? I am putting extra hardware that means, I am putting extra multiplexer in this circuit or this full adder cell, so that I am that means, making it redundant. The computation whenever we are doing this pre computation or pre assumption based computing we are doing, so, at the time, we are putting redundancy to the circuit.

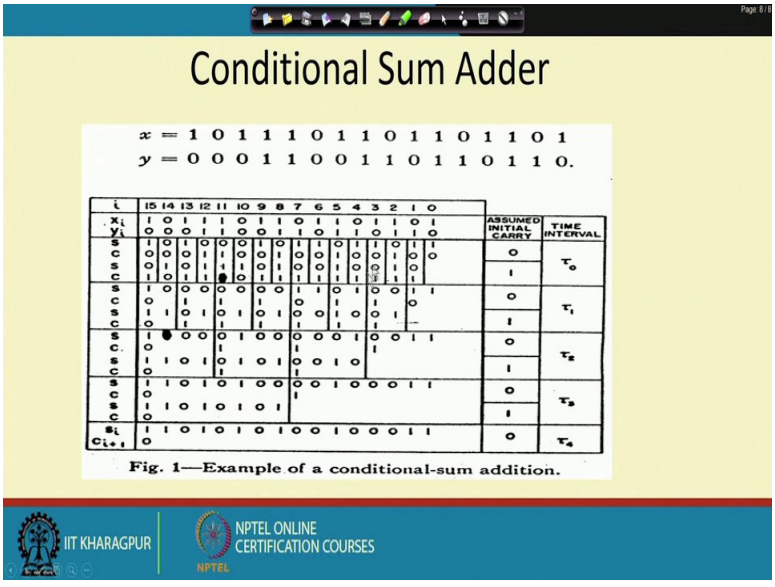
So that means, whenever we are putting redundancy; that means, it consumes more area than the original one, so that means, in these two in this particular case, I am getting the benefit of delay with respect to losing the area. So, if I am increasing the area, I can say

that or I can; that means, roughly say that I also can increase the corresponding power consumption, that we switching power consumption as I am putting more number of; that means, logically means here.

So that means, switching power consumption is one is one of the part of this dynamic power consumption right. So, switching power consumption if it is increases; that means, at the time I can say that dynamic power consumption is also increasing for this particular circuit. But for this parallel implementation of this, as I am doing parallel implementation, so that has again some of it is; that means, advantage for power consumption making the power consumption at the lower side.

So, from this particular unless and until, we are simulating it that means, we are implementing it in hardware, we cannot say that the power consumption is basically increasing. But just from the observation point, I can say that as switching power is increasing. So, dynamic power is increasing. So, power is increasing that I can say only.

(Refer Slide Time: 18:52)



So, if I take one example of this, so if I take one example how it comes; that means, computes. So, at that time you, it will be much more easier to you. So, this is considering what this is considering 4 bit, sorry 16 bit addition operation where x value is this one and the y value is this one. So, here you see the corresponding that means, number of levels which is required, that is only 4. In the fifth, I will get the output.

So; that means fourth, if I just like if I just mention it like a cycle, so at the time fourth cycle 4 cycle, I need to compute the 16 bit addition using this conditional sum adder. So, whenever if I consider this particular example, so at that time, for the LSB position, if I just go back; that means, sorry if I just go back for this is y 0 and x 0 and this will be added with c in. So, here c in is 0. So that means, whenever 1 and 0 is there as the input of x 0 and y 0 at that time sum will be 1 and carry will be 0. For the y 1 position, y 1 x 1 position, what will happen, one will compute if I just again go back.

So, this is this again will get two set of sum and carry; one considering carry input bit of 0 and one considering the carry input bit of 1. So that means, now I will get two set as this x 1 and y 1, they are 0, 1. So, for 0 sum is 1, carry is 0. If carry input bit if 1 say at the time sum is you, carry output bit that is 1. And from this particular point from this particular point, if I see; that means, now if you just go back, so, based on this what it will select? It will select as this carry output bit it is 0.

So, it will choose which value? These values, considering the carry input bit of 0. So; that means, now in this particular case, 1 and 0 that will be selected for from this particular point; that means, this c 2 will be 0 and s 1 will be 1.

Then, that means, s 1 is 1 and carry c 2 is 0. Then for the; that means, second position ok, so, y 2 and x 2, for y 2 and x 2, what is the value y 2 and x 2? I am getting for 1 1, I will get sum equals to 0 carry equals to 1. For 1, 1 n, carry 1. So, both are 1. So now, whatever is then me; that means, now if you just go back, so this c 3 and this c 3, now it will take the decision.

So that means, it will take the decision of what? It will take the decision for y 3 and x 3 sum selection or the carry generation selection; that means, for y 3 and x 3, consider as this is 1, 0, so, considering 0, I will get 1, 0 considering 1 in carry input, I will get 0 and 1 right. So, whenever I will choose whenever I will come to this particular point, so at the time, what will be selected here? What it will be selected from here? It will be selecting this as the previous carry here, there is 1 sorry.

Yes, the previous carry is 1; that means it will choose 0 and 1. And for this particular case is also, the carry output bit that is 1; that means, again it will also choose 0 and 1. So that means, in both of this case as the c 3 is 1, so it will choose 0 and 1, for this also 0 and 1, for this also 0 and 1, then 0 and 1, then 0 and 1. So, now, this again this will be

selected based on what among which set I will choose which set I will choose it will be selected this 0, 1 or 0, 1, this will be selected based on these particular signal.

So, what is my, what was my c 2? c 2 is basically coming from this particular point. So, c 2 is 0; that means, it will select 0 and 1 over here. So that means, the carry output bit, the carry output bit or c 4 value is 1 here, whether this is 3 as this is of 2 bit. So, here to bit means, both of this is basically 0 and 0, this is 0 and 0. So, I will get s 3 and s 2 of 0 and 0.

So that means, s 3 of 0 and 0 and for this 2 particular case the sum is 1 and 1; so that means, now I am getting s 3 down to 0, they are 0, 0, 1, 1. So, then again come to this particular point, for computation of this particular circuit where from we have to start? We have to start from this particular set. So that means, here for y 4 and x 4, for y 4 and x 4, so, this is up to this.

So, 4 y 4 and x 4, for 0, 1 this is sum is 1, this is 0 other case considering the carry input bit of 1, there 0 and 1, right. Then again, this will effect what to the 5, 5 position. So, in 5 position, what are they? 1, 1 means 0 and 1, 1 and 1. So, that means, considering the carry input bit of 1, there is 1, 1, 1, and for considering carry input bit of 0, that is 0, 1. So, now, based on these for carry input bit of 0, what I will choose for this, there is 0, 1. For considering the carry out in output of this particular stage 1, we will choose one one over here.

So, then again, you just come to this point. So that means, now these two bit, if I consider these two bit, then what will be the bit, what bit, I will consider they are that means, for 0 (Refer Time: 27:00) and 1. Why because, this bit is the output of carry off from this if I just go back the output carry from this particular point, it is from this is computing 0, 1 and for this is choosing 1, 1. And based on this, I will get what? I will get 1 that means, these two combination will be what 1, sum for sum from here I am getting 1 and from here also I am getting 1.

That means 1 and 1 and for this, this is 0. This will be for sorry, 4, 5; this will be 0 and 1. So, the sum is 1 over here. So that means, I will choose here and for fifth, I will choose 0 here, for what for this particular stage. Now, again I will generate the carry output bit 1 here and 1 here. If you see that for both the case for both that case fifth, these to carry output bit of carry there both are 1.

So, now, that means, now this again it will effect what this 1 and 1, that will effect to the 6 and 7 position. So, again something in this manner, we have to find out; that means, this x 6 and x 7, sorry x 6 and y 6, they will affect this y 7 and x 7. Then again from this multiplexer we will get two bit group two bit group means, one from this particulars sum and one from the sum of this particular stage.

So, then again, they will be selected based on the carry which is generated from if I that means, if I take this as a group, so from this group whatever the carry is generated. So, that will effect to this and then if I consider these four as a group, so the combination of this the carry generated from this 4 that will come to effect at this final multiplexer stage.

So, I will get concatenate if I concatenate 1 and 1, then I will get 2. If I concatenate these 2 and these 2, I will get from these 2 multiplexer, I will get 4 out 4 output bit. So, then these 4 output bit will come here and I am getting 4 output bit here. So, they can again they will concatenate and we will produce the 8 bits sum over here. And the carry output bit from this particular stage which is dependent on these whatever signal I am producing for this particular group which is nothing but this c 4.

So, in this manner, we can compute this 16 bit or any bit that means, here it is 8 bit. So, if I just want to; that means, make it 10 bit, so for 10 bit also, it will have 4 of these multiplexers. So, in this manner, we have to design the corresponding conditional or we can design the corresponding conditional sum adder.

So, this is the; that means, we are taking are taking the one example, so at that time yes. So, in this manner as this is of 16 bit, so in this manner it has been generated then based on that this is of; that means, this is the first level of output. Then, this is the second level of output and this is the third level of output and in the fourth level, we are getting the final output bit. That means, the addition results of this considering the value of x and y of these two ok.

So, for today's class, up to this one, then again there are various type of; that means, fast adder architecture which are already there in there; that means, literature. So, we will discuss with that what type of other architecture, adder architectures are available and; that means, how there; that means, how they are implemented considering the speed or power as the major constraint or area as the major constraint. That we will see in the future classes.

Thank you for today.