Digital Circuits and Systems
Prof. S. Srinivasan
Department of Electrical Engineering
Indian Institute of Technology Madras
Lecture # 12
Carry Look Ahead Address

In the last lecture we introduced the concept of propagation delay. Any hardware has a delay. So when you draw a block diagram or a circuit diagram we give inputs and you design outputs, you sort of assume that the outputs are available as soon as inputs are given. But however fast the hardware is the gates and other things you have inside the circuit there is to be a finite time delay for the transmission of signals. This time is called the propagation delay and of course depends on the length of the signal path.

As soon as signals are given the gates start switching. Basically these are switching activity from one level to the next level. Sometimes switching is not present as switching based on the input conditions. Based on the input conditions and function of the gate we have a switching activity. So the switching activity is going to take a finite amount of time and this propagates through this circuit. So the total propagation delay of any circuit or subsystem or even a big system depends on the length of the path that signals have to traverse travel from the input to the output.

So when you want to design fast circuits or fast systems of course one is the technology solution. What we have today is much superior to what we had thirty years ago, twenty years ago, ten years ago even five years ago so speeds are improving because of technology, that is one aspect. The other aspect is can we reduce the path of transmission. The path that the signals have to travel from input to output if you can reduce the path you can reduce the propagation delay or increase the speed of operation of a circuit. This is a concept with people use all the time. For a given technology if you want to maximize the speed or you want to improve the speed then you have to go to this type of options reducing the propagation delay by cutting the path length. But then a given job has to be done. you take the case of the full adder a multi-bit adder so in each full adder the carry has to go from one stage to next stage so stage to stage the carry has to travel we saw the carry circuit yesterday.

We have a $_2$ level gate structure input AND gates feeding into an OR gate output of an OR gate of the carry for that particular bit. This input is required in the next stage before the next stage can start switching.

So naturally as you go on increasing the number of bits the propagation delay from bit to bit is going to increase and finally the time at which the reliable results are available at the output will depend on the number of stages and the delay of each stage.

Now, if you don't have to depend on this transmission of the carry we can sort of project the carry output of each stage or predict the carry of each stage. We can reduce the propagation delay. There are several techniques. Fast carry propagation is one of the standard problems in arithmetic circuit design and you know that the speeds of computers today are very high we talk of Giga hertz computers Giga hertz means 10 power 9 so we are talking about such speeds. Naturally people have thought about this problem and have solved this problem so as I said this is the first level course in digital design and that is not my intension to go to all those design techniques to improve the speed of performance, all the various techniques but conceptually should know that we can always decrease the delay or increase the speed by having fewer stages.

Two levels of gate I said for each carry bit and depending on the number of stages there are to be that many two levels. That means there are 8-bit addition each bit requires two level carry circuit so 8 into 16 levels of transmission gates. If you can reduce that level there are two ways of doing it I can predict it and see whether without having to wait for the previous carry can I start my addition for the next stage. Of course which case you will have to always wonder what is going to happen when the real carry comes how are you going to take care of them.

As I said these are all very many techniques we are not going to look into what I can do is sort off given As and Bs the bits to be added bits of number A and bits of number B, you can sort of predict or look at what would be the carry for each stage and then feed those carries directly into those respective stages.

In other words do a carry computation, instead of having to wait for the carry to propagate do a carry computation and feed them all at the same time into the respective stage which means there has to be an additional logic. that means I need to have extra hardware given As and Bs if I want to calculate the carrys of all these different stages ahead of time I need to have extra hardware extra logic to accomplish this. So it clearly means that at the cost of extra logic or extra hardware I can speed up the operation of a circuit. This is something which is conceptually not very difficult to understand, intuitively, more hardware means faster you can do things.

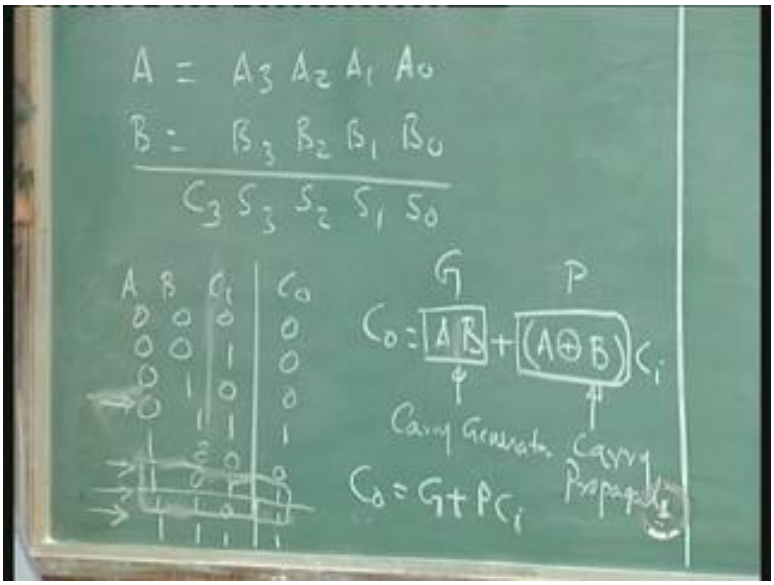
If you want to do something with a single unit of hardware it takes certain amount of time and if you have many pieces of hardware you can do it faster of course not always if probably for a partition like that but in most applications. In other words anything you want, the improvement in performance has to come in at the cost of something else and usually that something else is the extra logic.

Extra logic means extra cost and extra cost means extra power so it is all trade off, at what speed you want and what is the prize you are willing to pay. You want to pay more money get more gates and dissipate more power if you don't mind dissipating more power I can possibly get a better speed up. This is the concept I want to bring in rather than the actual technique there are so many techniques as an illustration of this concept we will only look at one of the techniques that is called Carry Look Ahead adder. So we

will talk about Carry Look Ahead adder for speed improvement of multi-bit adder. Also, some people call it CLA Carry Look Ahead adder.

As I said the differences can be looked ahead. As soon as the bits are known as soon as the bits are known ahead of time can you do a computation of the carry and then use it when it's required, it is sort of a prediction by computation in advance by using extra logic. That is why I said extra logic can give you extra speed and this technique is called Carry Look Ahead because of that reason. So we are going to have an extra hardware which is go into generate the carry for different stages keep it ready and when it's required it will be used. I will give you a simple example now.

(Refer Slide Time 17:04)



Let us use the 4-bit adders again A and B A is equal to A_3 A_2 A_1 A_0 and B is equal to B_3 B_2 B_1 B_0 these are the two bits you are going to add to get the sum S_0 S_1 S_2 S_3 and C_3 would be the fifth bit that we need. When you add two 4-bits you need a fifth bit and that fifth bit comes under this C_3.

Now, if you look at the truth table what are the conditions of the input for which there is a carry output. Carry output if you remember if you want to quickly do this only for Co there is the carry for this, there is the carry for these three. Now if you look at these two these are the four rows or four conditions of input for which you have an output carry. You look at these two conditions these arrows I have marked here these two conditions have one thing in common 0 1 1. Now we will put it this way, we put the carry differently, I will remove the arrow. The last two rows I will put under one category and these two rows I will put under another category. When both A and B are 1 we have a carry irrespective of whether C_i is 1 or 0 so that is one condition for which we have a carry. We are going to design logic to predict or look ahead. We have to design logic to look ahead. That means I should understand the carry mechanism. I should understand

the carry generation mechanism of the adder. If you want to really design a circuit which can generate my carrier in advance ahead of time that is why I am looking at it this way.

So, carry will be generated whenever both A and B are one irrespective of whether C is one or 0, the other condition where carry comes is when A or B is one of them Exclusive OR condition provided C_i also happens to be one. That means I can now sum up the carry mechanism of a full adder as; if both A and B are one I get carry that means A AND B if both are one the output carry is one OR when A Exclusive OR B is true but only when A Exclusive OR B true only when C_i is 1. So A Exclusive OR B and C_i is 1 condition in which output carry is one, A AND B is one is another condition for which output carry is one. so there are two conditions, look at A and B and the moment they are one you put a carry condition or look at A Exclusive OR B and C s 1 and see if C_i is 1 and if it is 1 make it one and if it is 0 it is not. so this is called carry generate, carry is generated in that stage whatever is the stage you have multi stage addition 4-bit addition in this case it will be 8-bits, it will be 16 bits, 32-bits so at each stage or each bit the carry will be generated if the corresponding bits are both one that is why it is called carry generate irrespective of the fact whether the previous carry was 1 or 0. So a fresh carry is generated in that stage if both A and B happen to be one in that stage. You don't have look at what is the carry pushed from the previous stage. The other condition is when A Exclusive OR B of that stage is one the carry would be generated only when the previous carry is pushed into that so this one I will call carry propagate.

Ci is propagated from previous stage to next stage if A Exclusive OR B is one do you understand this concept propagation. A carry from previous stage is propagated or transmitted to the next stage if A Exclusive OR B is one and A AND B is one a fresh carry is generated and transmitted to the next stage. So output carry will depend on whether A AND B is one or A Exclusive OR B is one but input carry is also one. Under these two conditions I have never exhausted all the possibilities of the carry generation mechanism of a 1 bit. This bit could be the eighth bit, seventh bit, eighteenth bit, thirty seventh bit I need to know what is the previous bit and the current value of A and B then I know whether the next bit is going to be 1 or 0 for carry.

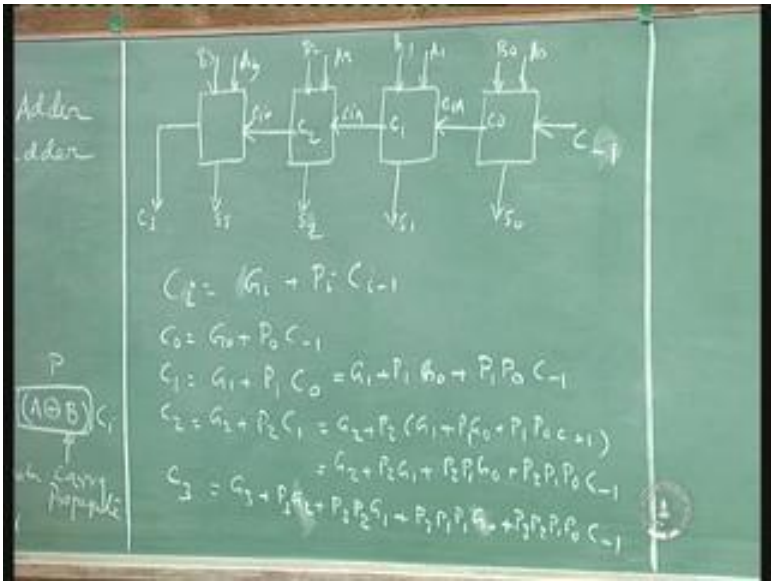
That means I can write it as, I will call this as G this as P, G for generate carry and P for propagate carry, carry generation and carry propagation. So I will have output carry as generate condition which is A AND B condition, I am going to call this G (Refer Slide Time: 16:54) and I am going to call this term P the propagate condition is A Exclusive OR B and Ci. So output carry will be G or PCi that G stands for A AND B G stands for A AND B and P stands for A Exclusive OR B.

Now let us look at our 4-bit adder. (Refer Slide Time: 17:55) this is my C minus 1, this is my A_0 B_0, C_0 is the output carry feeding as input carry for this, C_{in} A1 B1 output of this is called C1 feeding as input into this A_2 B_2, C_2 will be the output of this feeding as input to this A_3 B_3 this circuit I drew yesterday also in the last lecture. So this is A_3 B_3 and the output is C_3. I am only interested in C_3 finally. I am interested in S_3 S_2 S1 and S0 this is the sum and C_3 the carry. I am not interested in intermediate carry. Intermediate carry are C_0 C1 C_2 I am not interested but I need them for proceeding with my final computation.

in order to complete my additional process I need C_0 C1 C_2 but what I am interested in is to find out S_0 S1 S_2 S_3 and C_3 that is my final result.

So if I can now speed up the mechanism of generation of C_0 C1 C_2 and feed into the appropriate stages of the full adder I can get the result faster than that instead of having to wait for this carry to propagate from one to next to next to next to next to next to next like that I can get it faster than that.

(Refer Slide Time $_2$5:59)



We have already written for any given stage the C output is given as, so I will write this as a generalized term output carry I will call it C_i is equal to G_i P_i C_i minus 1. That means the input carry is called I minus 1 the previous ith stage, for ith stage output carry is C_i C subscript i and G_i is A_i AND B_i for that stage, P_i is A_i Exclusive OR B_i for that stage, C_i minus 1 is the carry coming from the previous stage. Now I can substitute for each stage.

For the first stage C_0 is equal to G_0 plus P_0 C minus 1 minus 1 I am putting usually it is the first 4-bit C minus 1 is 0 there is no input carry but as I said S in the last lecture it could be a stage in between a multi-stage 4-bits I can have several 4-bits this can be all of those intermediate 4-bits stage so I cannot assume C minus 1 to be 0 all the time so I am putting C minus 1.

Or you want to call this C_{in} if you are not comfortable with the word C minus 1 put the C_{in}. This C_{in} should not be confused. And C_1 is equal to G_1 plus P_1 C_0 this is where the looking ahead comes now. C_0 we have G_1 P_1 no problem, we are assuming that As and Bs are available at 1s in the beginning of the problem. As soon as you are asked to add you are given the values are A_0 A_1 A_2 A_3 B_0 B_1 B_2 B_3 so I can always compute G_0 G_1 G_2 G_3 P_0 P_1 P_2 P_3 all in one particular stage propagation delay will be there but it is very small. But what is not there is the value of C_0 which I have to get from this so now I can

substitute for this C_0 in this equation, can I not do that? So now this will become G_1 P_1 what is C_0 it is G_0 plus P_0 P_1 C minus 1. I am substituting for the value of C_0 from this equation.

Now again I know G_1, I know P_1, I know G_0, I know P_1, P_0 etc so at this stage I do not require any extra time to wait for my carry generation. The carry generation is based on the propagation delay of this stage it is logic, two-stage logic. It is a sum of product a two-stage logic so there is a fixed amount of time delay propagation delay we have to accept that but we do not have to wait for some carry to propagate from one stage to next stage.

Continue on this C_2 will be G_2 P_2 C_1 and if you substitute for C_1 from here this will be G_2 plus P_2 (G_1 plus P_1 G_0 plus P_1 P_0 C minus 1) and if you expand it, it will become G_2 P_2 G_1 plus P_2 P_1 G_0 plus P_2 P_1 P_0 C minus 1. Again I don't depend on any value of C expect C minus 1, again it is a $_2$-stage logic AND gates followed by OR gate. One set of AND gates feeding into an OR gate and then you get the output and C minus 1 is already known so there is no question of waiting for the carry to come through and then we can write by induction C_3 is equal to G_3 plus P_3 G_2 plus P_3 P_2 G_1 plus P_3 P_2 P_1 G_0 plus P_3 P_2 P_1 P_0 C minus 1.

So we have the equations for carry for stage one which has to feed into carry of this, the equation for carry of this has to be fed into this, the equation for carry of this is fed into this, equation for carry of this has been fed into this so I have the carry equations for all the stages I have the inputs for all the stages so all I need to do is to find out the propagation delay of the individual full adder which is of course that I am not saying it is 0 propagation delay I didn't say that. The propagation delay is minimized. This can be appreciated when we are talking of multi-bit adder in the sense of multi-bit adder the 4-bit adder, what is 4-bit maximum 16 numbers 0 to 15 but in computers when you are talking of number crunching applications we talk of 16 bits, 3_2 bits, 64-bits, $1_2$8-bits imagine the number of stages it has to travel in order to traverse to get to the output there you can appreciate the speeding up here.

Now what is this logic? All the G_s and P_s are known because G is nothing but AND combination and P is nothing but the Exclusive OR combination so I can have my Ps and G_s generated like this (Refer Slide Time: 26:10) A_i B_i gives me P_i I have four such gates for P_0 P_1 P_2 P_3, I have Gs A_i B_i G_i four such gates AND gates A_i and B_i. then I have this logic in which I have gone to feed my P_0 G_0 P_1 G_1 P_2 G_2 P_3 G_3 this is all been generated by that logic four sets of Exclusive OR gates and AND gates are fed into this, I am not showing it as a big drawing because it looks complex and then you lose track of connections.

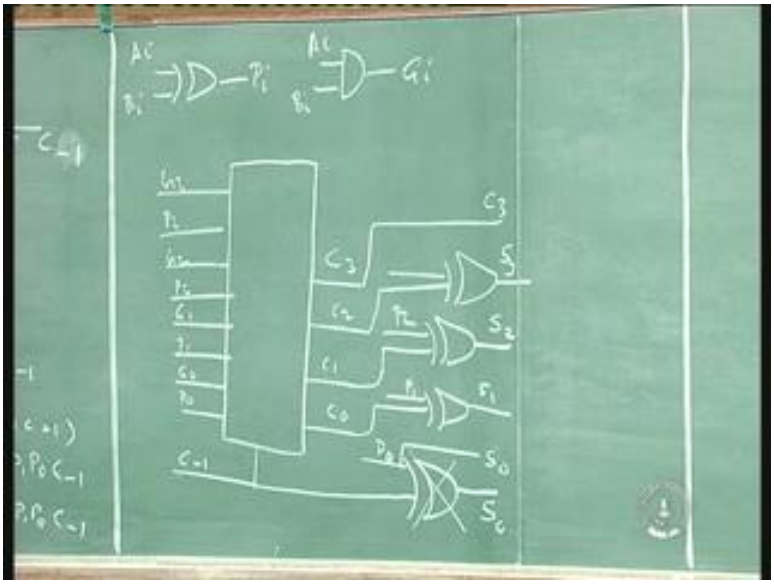
And all I have to do is to feed in my C minus 1 which is A_0, most of the time it is the first 4-bits and what would you get at the output? I get C_0 C_1 C_2 C_3 and what is inside this? Inside this is the logic for each one of these so there will be a gate structure to calculate this because this G and P are fed in, C minus 1 is fed in, this is an AND OR combination, there is going to be lots of AND OR combinations inside, I am showing in detail. The

number of gates will be more, for example this will require only two gates one AND gate for P_0 and C minus 1 and one OR gate for these two. This will require three gates one AND gate for this one, AND gate for this and one OR gate and this AND gate will have two inputs and this AND gate will have three inputs. The most complex circuit will be this which will have one two three four AND gates, this AND gate will have two inputs, this AND gate will have three inputs, this will have four inputs, this will have five inputs followed by an OR gate.

All these gates are only two levels that is what I am trying to tell you here the levels of transmission. The signal is given, As and Bs are given one level to get P_i and there is another level where the P_i and G_i are calculated simultaneously so after one level of computation or switching P_i gets computed G_i gets computed so that is fed here and after two levels I get C_s so 1 plus $_2$ is equal to 3 levels. Whereas if I have a multi-bit thing it is going to take several steps, each one of those things has two levels so going to two levels is a multiplication effect it goes all the way.

So I am buying speed the at cost of gates, I am paying for my speed in terms of gates, I pay gates get speed, it is always case, the trade off is always like that, good performance things are always more expensive, whatever it is, it is a general rule of course but there is always exception sometimes money cannot buy certain things it has to wait for this previous process to be completed just because when there is an extra person you have two persons, I can now say that if two persons can do a job in half the time but if the first person has to finished and handover the job to the second person two persons cannot do it faster than one person, there are exceptions like that we are not talking of those exceptions.

(Refer Slide Time 34:44)



In general, conceptually we pay by money in terms of gates or performance, performance is measured in terms of speed, cost is measured in terms of number of gates but number

of gates is not only the cost it also takes more power and more space and all that type of things. Anyway for one level two level three level I have C_0 C_1 C_2 C_3 and I need sum what is the sum of the full adder if you remember? S is A Exclusive OR B Exclusive OR C. So since I have A Exclusive OR B already and all the Cs one more level I get sum so P_0 Exclusive OR (Refer Slide Time: 31:00) this is C minus 1 this is C_0 the input to the second stage this is the carry input to the second stage, carry input to the third stage, carry input to the fourth stage, carry input to the output so this C minus 1 is not required so if you want that C minus 1 this is Pi P_0 Exclusive OR this gives you my S_0.

Sometimes this C minus 1 may be 0 as I said if the first stage if the first bit of a multi-bit addition input carry is 0 you don't need this gate but I am putting it as a general scheme in which this 4-bits can be slice of four intermediate bits of a much longer width. Suppose I have a 64-bit word in which I desire to put 4-bit at a time and this can be an intermediate 4-bit wherein the previous carry stage you will have to feed into this. This is a generalized drawing even though in this particular case for 4-bit addition the A_0 is the first bit and B_0 is the first bit this is not required. The sum is directly P_0, A Exclusive OR B is a half adder.

Half adder is A Exclusive OR B, we saw that in the last lecture half adder sum, full adder sum. Half adder sum is A Exclusive OR B which is P_0. Now P_1 is nothing but C_0 with P_1 S_1, C_1 with P_2 gives me S_2. This one will give me S_3 and this is C_3. So my final output is S_0 S_1 S_2 S_3 C_3. If C minus 1 is there this Exclusive OR will be there and if C minus 1 is not there this will be S_0. So now I have one more stage. Now the carry is generated so one gate is to transmit, one gate is to compute the sum, one gate is to compute this P_s and G_s, two levels of gates to compute carry. Now this doesn't depend on the number of inputs so the propagation is always same.

Therefore in a multi-bit adder the chain each of these carry would be generated after two stages that will be fed into the next stage as two stages, two stages etc and every bit addition requires two stages of computation and the last C_0 C_3 will come. We are talking of 8-bits or 16-bits, 3_2-bits even 4-bits we have eight stages. In this case that is one two three four of course sum I have not taken into account there also so it is one two three carrys available after the third stage itself. There the carry will be available after eight stages and sum of course will be parallely available but here the sum will be available after one extra stage. So sum will be available after four levels of gating which is again faster by four.

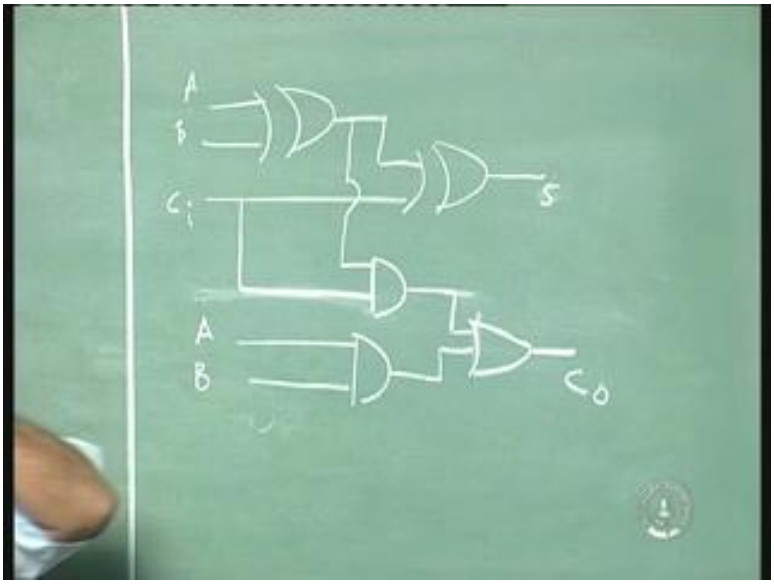
As I said I wanted to introduce a concept here. Carry Look Ahead adder is one of the most basic techniques to improve the speed of addition and arithmetic is not only addition, there are lots of arithmetic operations we have to do. We have to do subtraction, we have to do multiplication, we have to do division and large numbers we have to make we have to make exponential numbers whose magnitude is very large will have a radix and exponent that type of representation so we have to take care of all of that but the essence is all the same, do we speed up, when we need to speed up and what are the techniques used. There are several techniques but one of them is this. This is one of the simplest techniques to understand but by this technique I want to bring home the point

that it is always possible to have extra logic in order to accomplish certain things and make it faster.

Now of course you can always argue that I am going to have large number of gates of course I said large number of gates required I did not say it is free, you don't get anything free. If you can get the same number of gates as a 4-bit full adder that is if you can get a faster one then why would you think of the other one. It is a question of cost versus performance trade off, trade off or performance versus cost. So the cost comes in terms of these extra gates. That is why I put this separate black block (Refer Slide Time: 36:26) this is the extra logic unit called CLA logic Carry Look Ahead logic.

One other thing is the full adder concept which is a very modular concept. Without Carry Look Ahead adder if you remember the full adder was this and we have or you can even have this Exclusive OR, A Exclusive OR B and what is that carry, carry is A Exclusive OR B, A Exclusive OR B AND C A AND B this is my carry output. A full adder without Carry Look Ahead we saw that sum is A Exclusive OR B Exclusive OR C carry we saw can be written in two ways as AB plus AC_i plus BC_i OR AB plus C_i into A Exclusive OR B so I am using that option C_i times A Exclusive OR B C_i AND A Exclusive OR B OR A AND B.

(Refer Slide Time $38:3_2$)



Even though the propagation has to go from here the C_0 two levels and then you have two levels so it goes on one thing which is elegant here is the modularity of the circuit. That means whatever is the number of stages what I have to do is to build an identical block. Two bits and this is repeated twice, 4-bits repeated four times, 8-bits repeated eight times, 16 bits repeated sixteen times so sometimes it is easy to design circuits. Especially when you want to build it in a given a space we talked about space being a premium in

integrated circuit design we talked about that in the beginning the parameters are again the speed, the size, the cost and the power dissipation.

Therefore you can pack it nicely; these identical circuits are called modularity modular circuits whereas here this will be all irregular. This AND gate will have five inputs, this will have four inputs, this will have three inputs, this will have two inputs and one input here so it is an irregular structure of AND OR combination. At least if all the four carry circuits are the similar? No, they are not, this was five terms, this was four terms, this was three terms, this was two terms like that it goes on. So there is total irregularity you pay in terms of that so the optimization of the design in terms of space and everything that may in turn introduce some extra delay in wiring and all those types of things. Another big factor is I have put four stages A_0 A_1 A_2 A_3 B_0 B_1 B_2 B_3 suppose I had 8-bits A_0 to A_7, B_0 to B_7 so this equation will have how many terms, the last will be C C_0 C_1 C_2 C_3 C_4, C_5, C_6, C_7.

The last carry of the seventh bit that is the bit seven actually the eighth stage we call it bit seven because we started with 0, 0 to 7, so the eighth stage which is the seventh bit, bit seven C_7 would be we will start with G_7 this will be 4 plus 1 term here so that will be 7 plus 1 term 8 terms that will have eight terms and 4 plus 1 is equal to 5 so nine terms not only that how many inputs are to this gate? It is 9 so nine input AND gate, eight input AND gate, seven input AND gate, six input, five input, four input, three input, two input and all of them feeding into an OR gate with so many nine inputs so that will be a circuit which is sort of very unwieldy and again some of those hardware aspects I have not discussed, so is it any limit on the number of input that a gate can have?

Usually we have two inputs or three inputs or four inputs, is there any limit can I have seventeen input gates? Can I have 47 input gates naturally the question arises. There are some restrictions on that based on the current levels, voltage levels and the transistors that come in series switching elements which are inside these gates which have not touched upon, we are teaching this course at the system level not necessarily system level logic level. If you remember in the introductory lecture I talked about logic level, circuit level, subsystem level, system level like that.

So we are talking about logic level when you break logic you see actual elements which form the circuit like diodes and transistors, resistors or some capacitors. So when they are coming in series there is going to be some problem how many can be really put in series without suffering from the loss of the signal without your signal getting completely dropped to less than realizable less than recognizable values. So these are issues so there is a limit on the number of inputs this is called fan-in the number of inputs that you can have a gate is called a fan-in. the number of outputs you can have from the gate is called a fan-out.

Fan-out is the number of outputs you can take from a gate to feed similar gates without degradation in the performance of the output so that you can still maintain the 0 level and 1 level as per specifications. So you have 0 level specifications in terms of currents and voltage, one level specification in terms of current and voltage for any gate. Suppose I

say a gate is 0V and 5V it cannot be absolute 5V there will be a variation and it will also tell you how much current it will deliver. If you go on connecting many outputs to this gate a point will be reached when I will not be able to maintain that level that is required for driving the next gate that is called fan-out.

The maximum number of similar gates you can connect to the output of a gate so that the levels are reliably maintained both in currents or voltage. Likewise how many inputs can I give to a gate without any degradation in performance so that the levels are reliable is called fan-in. Hence there is a limit on this fan-in and fan-out for different families, different technologies. So I cannot go on arbitrarily adding the number of bits, I cannot say I will design a $_2$7-bit Carry Look Ahead circuit I cannot do that, somewhere it is going to fail.

Therefore these are the two negative points of this Carry Look Ahead adder. the Carry Look Ahead adder positive is speed up and all that so you don't have to wait for the carry to propagate you are looking ahead in sort of, I won't call it prediction it is not prediction but it is pre-computation, prediction is something which can go wrong, it is a pre-computation look ahead of what is going to happen. The negative points are the non-uniform circuit. Of course extra logic is one negative point that is why we are willing to pay for that otherwise we will not go for it.

Knowingly you go for I will pay more money give me a faster speed faster gate you are going to ask, I will give you more money give me a faster adder you ask for it you got it so that is not a disadvantage, I won't list it as a disadvantage, it is a technique of reducing the speed but from the disadvantage point of view there are two disadvantages; one is the non-uniformity non-modularity of the Carry Look Ahead adder and the second thing is there is a limit on number of bits you can do for Carry Look Ahead adder. People say 4 is ok sometimes 8 is a problem, between 4 and 8 somewhere like 6 or 7 may be optimum of 5 or 6 may be optimum. But unfortunately in digital we always do with 0 two powers always for some convenience and sometimes for practical reasons. you do 0 1 $_2$ 4 8 16 like that it is very rare to find a 7-bit adder in the market you can buy an 8-bit adder you cannot buy a 6-bit register you can buy a 8-bit register of course you can always build one that is not what I mean.

So 4-bits sometimes 8-bits are there. So what happens if I want to do really a multi-bit addition like 16 bits or 3_2 bits or 64-bits. There are two options. I can divide this into 4-bit 4-bit 4-bit in a carry propagation mode and within each of those 4-bits I can do a Carry Look Ahead. So first 4-bits Carry Look Ahead output carry will trigger the second stage so instead of having sixty four times the propagation delay I will have only sixteen times the propagation delay plus the extra propagation delay for each Carry Look Ahead stage that is one option. The second option is within that I can do a Carry Look Ahead. I do a Carry Look Ahead at the fourth bit, eighth bit, twelfth bit, sixteenth bit and use another Carry Look Ahead logic to generate higher level Carry Look Ahead. So I can go on doing it cascading in levels of Carry Look Ahead also can be done in different levels.

I can have a $_2$ level Carry Look Ahead, I can have a 3 level Carry Look Ahead, I can have a four level Carry Look Ahead. So Carry Look Ahead along with transmission serial, carry propagation combined with Carry Look Ahead is first option where I will have 4-bit Carry Look Aheads connected in carry propagation mode. So the speed is increased somewhat, propagation delay is reduced somewhat. The other one is if you want to do even better than that within the Carry Look Ahead output I look ahead the carry for the first 4-bits at the same time I can get carry for all of them. I can look ahead the carry for the eight stage, first 4-bits I can do, second 4-bits I can do, third 4-bits I can do so all of them can feed into another Carry Look Ahead stage, if it is not sufficient I can go to one more stage like that, that is more and more complex circuit that means when you have more logic and more non-modularity so again there is a trade off there is speed variation I can have a menu now, no Carry Look Ahead for me give me I will do a modular design with full adders I am happy, the speed is good enough take it. Ok I want to an improvement in speed I will do Carry Look Ahead but I don't more than 4-bits, give me any number of bits I will divide into 4-bit 4-bit 4-bit and each inside I will do Carry Look Ahead and outside they will connect as carry propagate, that is second option.

So more logic, increase in speed so you pay for it but you know that you pay for it and the modularity and all that extra fan-in considerations you have already taken into account. third is, I am not satisfied with this speed you are going to get with this, I want much faster then the output of the Carry Look Aheads the carrys of the four stages the fourth stage Carry Look Ahead can be used in a second level Carry Look Ahead to get a carry generation much faster than the other one. So this requires two levels Carry Look Ahead design which is going to be slightly more complex in terms of hardware, in design complexity, number of gates and thereby the non-modularity, size, etc so all those problems come again here also.

But let us not go into to all that, we will talk about at most Carry Look Ahead committed in propagation mode. Suppose you want to build a sixteen bit full adder I will probably do it at four Carry Look Ahead adders, C_3 feeding into second stage, C_7 feeding into third stage and C_{11} feeding into the fourth stage that is good enough for this class. But basically even though Carry Look Ahead as I said is the example to use for speeding up this is something which you should be sort of familiar with at the conceptual level. Arithmetic circuit I have lots of them, so that is the beauty of design of some analysis.

There is an analysis you have to lead that analysis, somebody else does analysis and you just understand and learn it. In design you have options, what are the various techniques you can speed up based on the consideration, based on the design requirement, based on the cost, based on the performance requirement you decide and that is where the design concept comes in. So I just thought I will introduce this concept to you. This is a general concept in terms of hardware, extra hardware can reduce the propagation delay or increase the speed or extra hardware can improve the performance in most cases, that qualifying class I want to always put. But there are some cases where this extra money cannot buy what you want so the […….51:12] is not always possible.