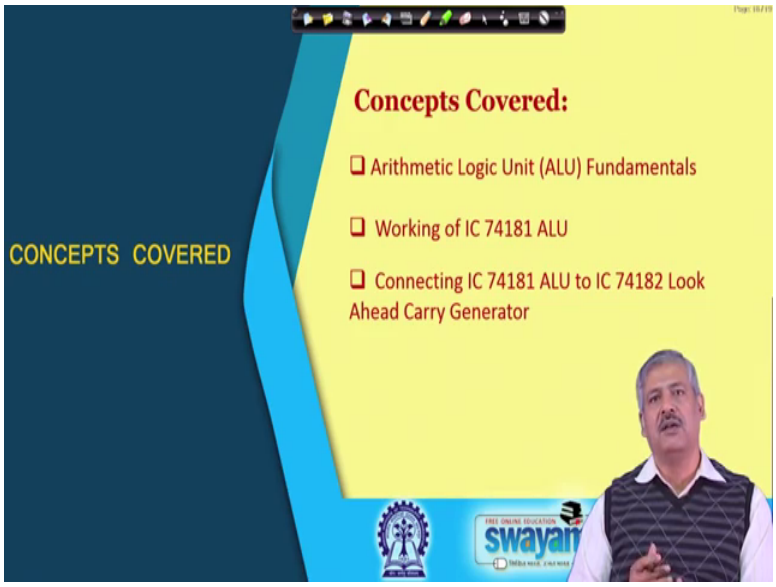
**Digital Electronic Circuits**
**Prof. Goutam Saha**
**Department of E & EC Engineering**
**Indian Institute of Technology, Kharagpur**

**Lecture – 27**
**Arithmetic Logic Unit (ALU)**

Hello, everybody. In this particular class, we shall discuss Arithmetic Logic Unit often we term it as ALU, the abbreviation of it.

(Refer Slide Time: 00:27)

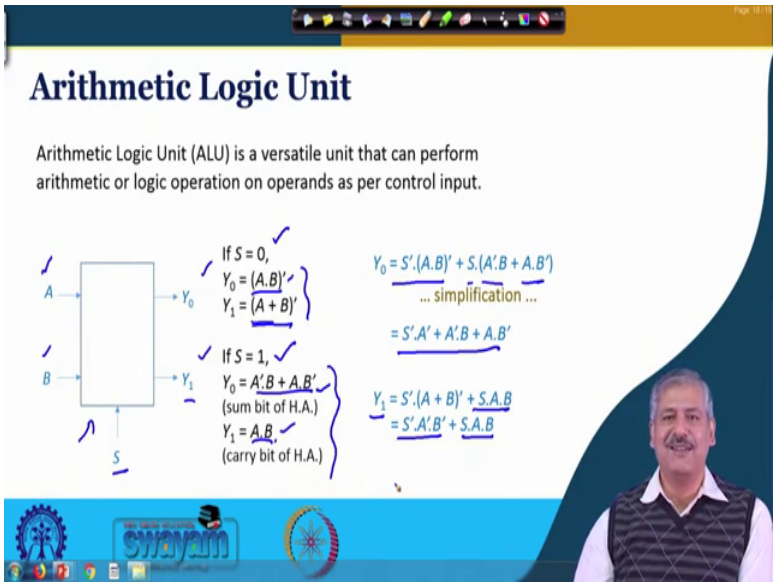



And, arithmetic logic unit as you understand from the very name of it that it is one unit within which both arithmetic and logic operation are to be done ah. The idea is that if you are looking for a circuit which is to be used in a general purpose device. So, you do not know for sure that what this circuit is you know going to do. So, when we require it will do some logic operation, in when required it will do it will do a different logic operation or it might do an arithmetic operation of one kind or the other.

So, would you like to have many such you know possible circuits and then invoke one of them when it is required and others are remaining you know idle at that time or would you like to have a device which can do all the different function many different functions and you have appropriate selection input by which you choose which arithmetic or logic operation you want to do, ok.

So, for such general purpose you know device you would be preferring an option like having a arithmetic and logic computing unit and which will be used by the main processor or the other you know processing units, ok. So, this is the idea. So, you can understand that when you are talking about such unit, it is relatively complex in design when you look for practical use of say using it for 4-bit, 8-bit kind of thing and many different functionalities are involved.

(Refer Slide Time: 02:11)




So, what we shall start, we shall start discussion on this by you know in step by step manner. So, that we you know do not get you know frightened by looking at the bigger circuit more complex circuit having you know about 70 logic gates or so.

So, first we look at a simple circuit where we are doing a 1 bit you know operation and the operation that we do here is either a logic operation NAND or NOT or a half addition, ok. So, this is the thing that we are doing, right and so, it has got two inputs A and B, A and B. So, these are the operands we can say and the result of you know the operation is coming from coming out in Y naught and Y 1, Y naught and Y 1 and S is the select input only one select input is there. So, it is selecting, if S is equal to 0, then Y naught output will be a logic operation AB at NAND of AB and at that time Y 1 if you look at Y 1 with a circle Y 1 you will get A naught B, ok.

But, if S is equal to 1, then what you get is the half addition. So, Y naught will be the sum bit which is XOR of A and B, right. So, A prime B plus A by AB prime and Y 1 will

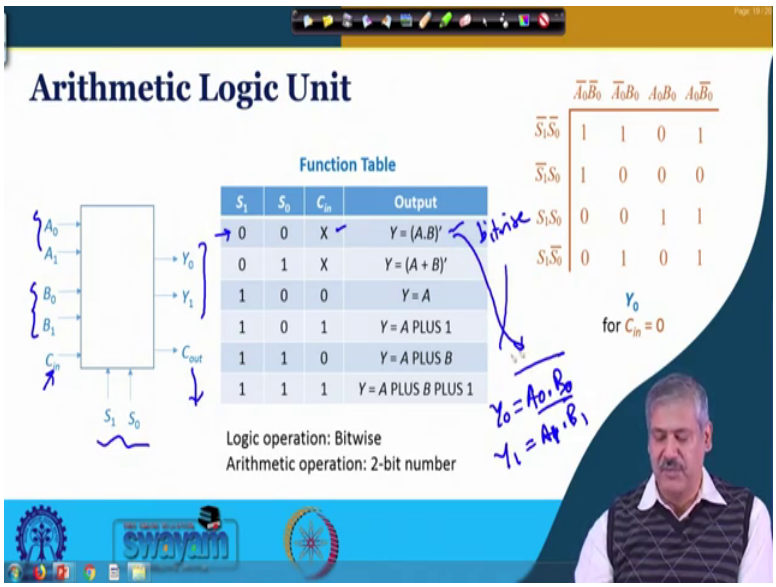
be the carry bit. So, this is the carry bit. So, that is AB you all know about it. So, we have seen this before, right.

So, how would you write the circuit inside the for this simple 1 bit unit? So, we very simple you know logic function and arithmetic function that we have seen. So, you will be writing S prime, when this is the half bit at the case for Y naught S prime AB prime for here for this particular case and S plus that is ORed with when S is equal to 1, A prime B ORed with AB prime. So, this is now we will write it, and if you do simplification then you can see this is the way Y naught will be related to S, A and B.

And, if you look at Y 1 how will you write it? So, S prime and A plus B prime, right and S A B this is coming from here, right. So, if you take it there. So, this you from the you know De Morgans theorem S A prime S prime A prime B prime then S A B. You can see that this is you know the simplified version of it. So, realize them. So, it does the operation. So, it is little bit simple.

Now, let us look at a you know more complex thing.

(Refer Slide Time: 05:11)




So, here we are having a 2-bit operands, right. So, A naught A 1, here and B naught B 1 and the corresponding outputs are Y naught Y 1, and we have additional input carry in and carry out, here additional inputs, and now we have got to select inputs S 1 and S naught ok. So, what we see here? So, we look at a function table ah, ok. So, for S 1 and S

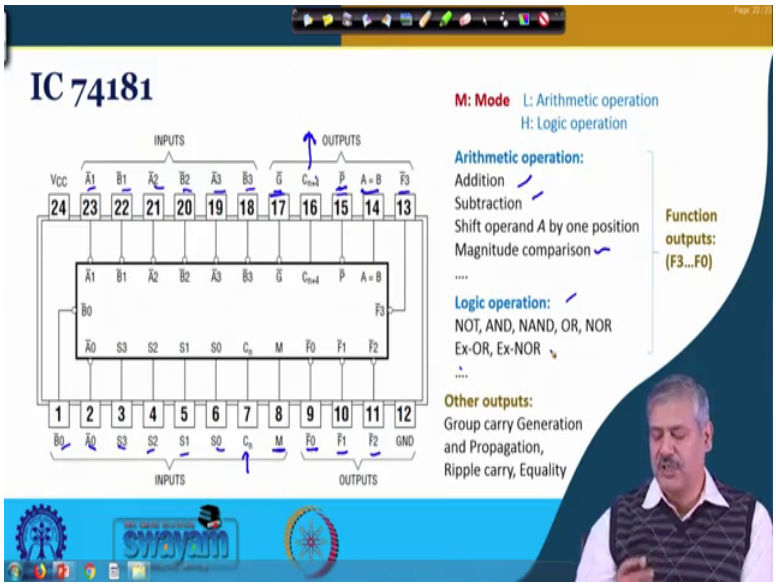
naught 0 and 0 0 0 so, we we ignore the carry in we do not take into consideration the carry in at that time and the output at that time is A NAND B; NAND of AB, ok. So, by this what we mean? We are meaning bit wise operation, ok, right. So, that means, Y naught is A naught NAND B naught Y 1 is A 1 NAND B 1. So, that is what we mean by this. Is it ok?

So, similarly for 0 1 we are having NOR operation bit wise NOR operation. So, these are the thing that is happening for you know these cases 0 0 and 0 1 and for 1 0, if carry in is 0 Y is equal to 1 and it carry is equal to 1, this is A plus 1. Now, we needs are introducing a term plus to differentiate it from the sum operation that we do it this symbol, ok. So, A plus 1 and when this selection inputs are 1 1, right if carry is 0 this is A plus B and carry is 1 this is A plus B plus 1, ok. So, that means, it includes the carry in right. So, basically what you are doing is adder full addition. So, this is what we are doing is it, ok.

Now, how to develop the circuit for this? So, we can see I mean basically it is a now 5 variable problems S 1 S naught and you know C in is there. So, of C in is 0. So, you know to make it understandable here. So, if C in is 0, ok. So, at that time for these options S 1 S naught 0 0, ok. So, when a naught B naught both are 0, this is NAND. So, this is this is the line we are talking about. So, this is 1, right when one of them 0; A naught is 0 and B naught is 1 NAND output means this is 1. When both of them are 1 NAND means this time it is 0; when A naught is 1 and B naught is 0 this is 1. So, this is the way you can complete this one and this is the corresponding truth table, right and you can have a relationship you know going forward for a five variable problem and minimize and you can the circuit, and you can see if it is now multiple outputs are there whether you can take some intermediate output to the next level.

So, this is the way you can visualize a 2-bit arithmetic logic unit which is doing you know two set of two logic operation. And essentially this is one mathematical operation; arithmetic operation this is 1 just carry part is included in one case and carrying is not included in the other case, basically it is not two different. So, basically with two inputs you can have four possibilities two is assigned to arithmetic and two are assigned to two are assigned to two are assigned to logic, ok.

(Refer Slide Time: 09:32)




Now, we go to something which is actual arithmetic logic unit IC that you might use in the lab this is IC 74181. So, you can see it is a 24-bit IC and in this IC this is a 4 bit arithmetic logic unit; 4-bit arithmetic logic unit and the inputs are A naught to A 3 where you can see them A naught through A 3, A 1, A naught, A 1, A naught, A 1, A 2 and A 3 this is active look right and B naught, B 1, B 2 and B 3, right. So, these are the inputs. So, selection; so, selection is designated by S, S naught, S 1, S 2, S 3. So, what it means. So, it tells us that some 2 to the power 4, 16 possible you know operations is possible, actually not you can do more than that because you have it has got as a additional unit input called M; M stands for mode. So, if M is equal to high it is logic operation and M is equal to low, it is arithmetic operation.

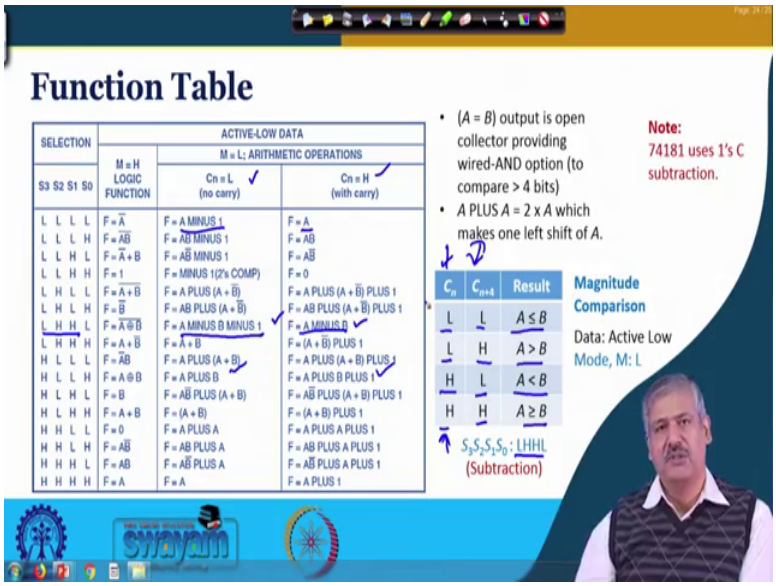
So, for each of this cases of m you have got sixteen possibilities. So, total thirty two different possibilities are there sixteen arithmetic operation and sixteen logic operation is possible through this particular ALU, ok. And, this is the carry input which can be taken from previous stage or you know another logic circuit and this is the carry output which we can take to the next circuit so it can be used for ripple carry addition if so, you know require, ok. And, other than that look at you have this group carry generation and group carry propagation term that is getting generated out of this which is useful which would be useful and we shall see one such you know frame work by which these two group carry terms are used for first addition.

So, internally it is doing first addition, but if it is to be connected to the next stage or other stage is if you want ripple carry then this is there C n plus 4 and if you what look at can be generation for which you need G and P term getting generated till they are also there, ok.

So, equality A is equal to B this output is generated, but not A greater than B or A less than B that I told, right and then these are the main outputs for these operation A and B, where A and B are operand, are generated from F naught generated at F 1, F naught, F 1, F 2 and F 3; is it clear right.

So, when you look at the function table we shall see the arithmetic operation that are being done are include addition subtraction shift of operand A. So, A is consider as a main operand here by one position, right. So, then magnitude comparison not directly because there is use say there is no direct output as A greater than B or A less than B; A equal to B is there, but not the other cases. So, by some other means we can do it and we shall see that logic operation like NOT, AND, NAND, OR, NOR, Ex-OR, Ex-NOR they all these common things are there, but more than that also it is there, ok. So, this we shall see.

(Refer Slide Time: 13:07)




So, this is the function table, ok. So, again it might look little bit you know heavy ah, but because there are you know thirty two different possibilities are there. So, this is M is equal to H right the logic function it is generating, ok. So, this these are the different

logic function for choices of S 3 to S naught S say L, L, L, L; that means, all of them are low. So, at that time the whatever at the it is functional puts F 3 to F naught will be the complement of A 3, A 2, A 1 and A naught. So, that is what it says, that is what you will get, right. That is the why the circuit is different. We shall look at one example also after few slides, is it, ok?

So, next one is similar you know the different logic functions we can see B 1 that I told that NOR will be available ex-NOR is there. So, many other things are there we shall see what are they and what is it is simplification and similarly for logic M is equal to low, so, depending on presence of carry and absence of carry and presence of carry the meaning is different if in one case if it is A minus 1 when carry is included it is F is equal to A, ok. So, we just input is passing through to the output. So, the different cases a plus B and really a plus B plus 1 right. So, these are so, A minus B and this is A minus B minus 1, right. So, these are the different functions that you can get and make use of, ok.

Now, 1 thing important here the A is equal to B the magnitude comparate comparison output the magnitude comparator output here is pin collector, ok. So, open collector we understand. What does it what does it mean basically you need a additional you know resistance and it is connected to the power supply through which actually the circuit offers. Other than that what we know and it can deliver more current? So, other than that we know that it can offer wired and connection also. So, if you have another say a you know 4-bit comparison being done by another say ALU which also has got open you know collector output for A is equal to B.

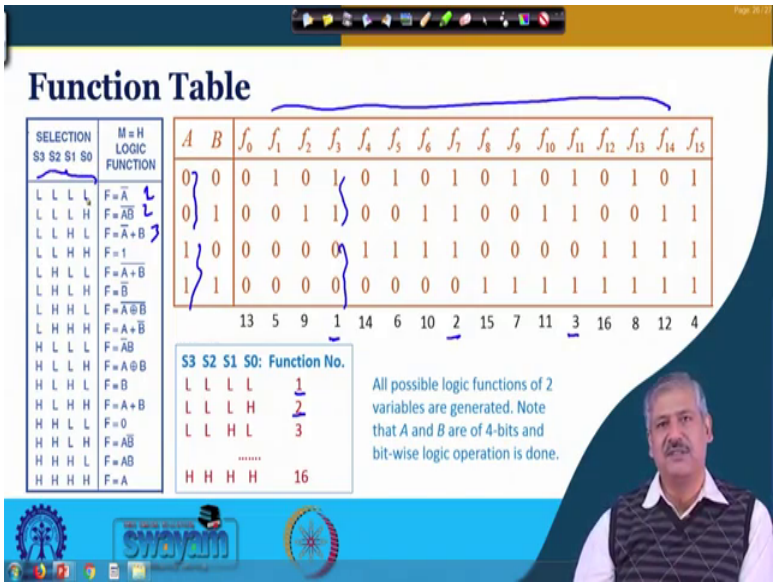
So, those 4-bits and these 4-bits, if you just add the, you know may provide connection I mean just connect the words there. So, it will be giving an AND logic. See both of them are high right then you can say as a whole this 8-bit number is high and oh as a whole 8-bit number is equal if one of them is low then of course, equality is not established. So, something else is happening, right. So, this you this is giving you a very quick you know comparison between two numbers which are being compared in two different ALUs, by making use of the wired-AND option of a open collector output, ok.

So, there is no multiplication as such, but A plus A means 2 into A which is basically is a left shit of one unit of A, ok. So, that is what we can see and for magnitude comparison there is no such you know is it separate input. So, you can look at the result, this carry

getting generated. So, it is noted what is the implication of the carry, right; when you are doing the subtraction when you are doing the subtraction ok. So, L H H L so, what is it? So, this is the one, ok.

So, in one case it is A minus B minus 1 and other is A minus B. In the last class for magnitude comparison we have seen that how to get it done through subtraction, right and so, we told at the time that ALU will be using subtraction for that. So, if it is low and this is low right then you can tell that this is A less than B less than equal to B; it is low, this is high, it is A greater than B you can see the 2's complementary arithmetic how it is done. This is high and this is low; then it is A less than B and if it is high it is high then it is A greater than B by which you can get the magnitude comparison in done, by understanding this carry output C n plus 4 in presence of or absence of C n .

(Refer Slide Time: 18:10)



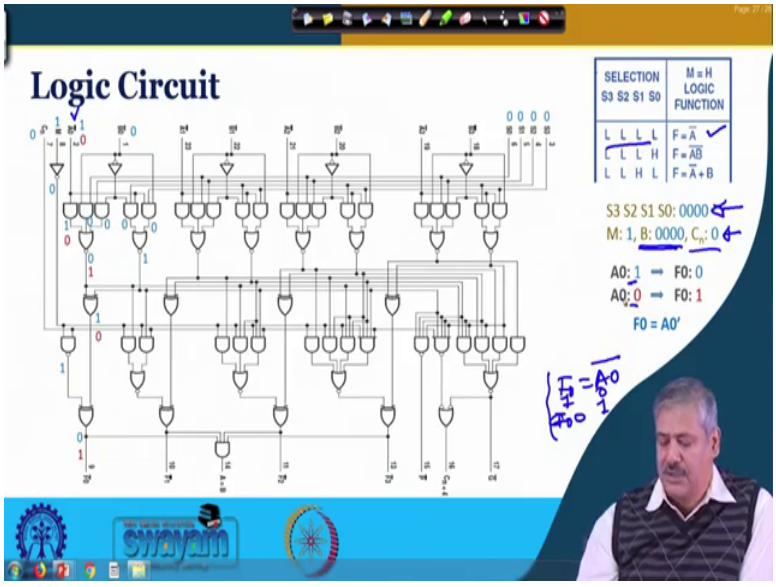

Now, the function table as I said they we had these logic functions, right. So, if you look at you know closely the closely the logic functions that are there we will see that the outputs here connects to this table on a one to one basis, ok. So, what is that and what is this table? So, this table we had seen in one of the you know earlier weeks in of this particular course. So, if two inputs what are the many different functions that are possible, ok. So, we found that 16 possible functions could be there with these two inputs 0 0 0 1, 1 0 1 1, ok. One possibility is that for each of this case is the output is 0 1

possibility is that only for 0 0 the output is 1 and rest of the cases it is 0. One possibility is that the for 0 1 it is 1, rest of the cases is 0.

So, with this four we can have again 2 to the power 4, 16 different combinations of 0's 0's and 1's, generating sixteen different functions sixteen different functions, right. We had seen where we had got AND, OR and those kind of cases, . So, if now what each of these cases L L L L the first one we designate it by number 1; second one we designate by number 2. This is number 1, this is 2, this is 3 and so on and so forth. You can identify that in this particular this function table this function table right 1 is somewhere. So, here is your 1 which is not there in other place because it give each one of them is unique, ok.

So, L L L L for this thing this is A bar and you can see this is 0 0 this is 1 and this is 1 this is 0 0. So, this is A bar. Similarly, you can find out 2 appearing here, 3 appearing here and that way each of this functions possible functions are actually covered by this logic operation. So, if you need any one of them, you can make use of them. So, required by invoking appropriate selection in the course by invoking appropriate selection inputs is it clear.

(Refer Slide Time: 20:51)




Now, this is the IC 74181 circuit. If you look at the you know manufacturers data sheet and all how it is there so, as I said it is a bit complex. So, is that is why we began with simpler you know version of it, ok, but now we are in a position to understand what it was represents and how it works or so, and we can look at one example case . Look at

one example case where the selection input is all 0 right, M is equal to 1; M is equal to 1 means it is logic function and B is all you know zeroes, right; C n is also 0.

So, if this is the case 0 0 0 0, then what we are expecting; F is equal to A prime this is bit wise you know compliment operation, right. So, we expect that F naught will be A naught F naught sorry A naught bar right; so, that means, when A naught is equal to 0, F naught will be 1; A naught is equal to 1, F naught will be 0. So, this is what we expect; similarly for other case, right.

So, and it does not depend on B and C n is immaterial. So, that is what we expect and we can see whether it is happening or not let us you know look at one such example case. So, this is where A naught is given. So, we have considered the case A naught given as 1 in the blue, and A naught given as 0 in the brown, ok. You can see the colors, right. So, when so, I am trying to make it cleaner. So, this A naught is 1, ok.

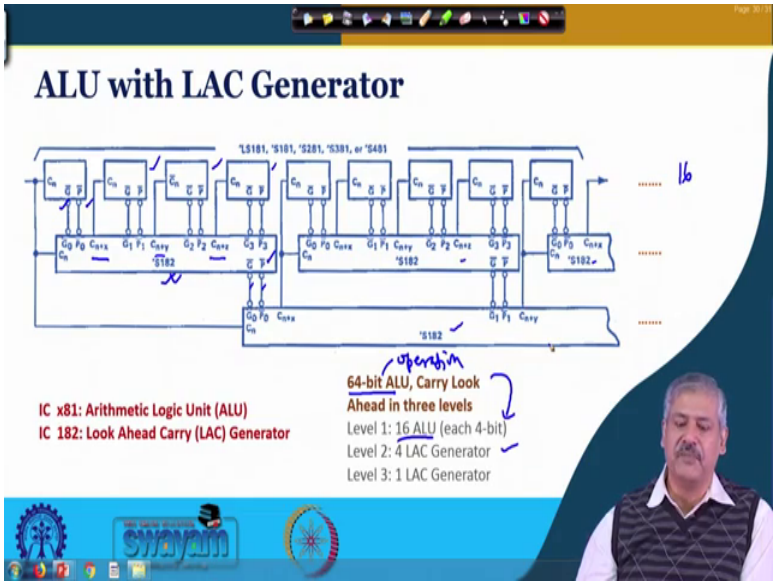
Then what is happening? This is 1 is coming over here; for NOR gate 1 is there means this is 0 the output is 0, and the other inputs this AND gates you can see it is coming from selection all selections are 0. So, these are 0 0. So, this is also 0 0 because one of the input is connected this the selection input which is 0 0 right. So, 0 0 this NOR gate this output is 1, ok.

Now, we look at the other cases ah. So, this is this ex or gate 0 and 1 this is the input. So, this is thus this output is 1. So, the for the final output for F naught this is the XOR gate which is giving an input 1, and we have to look at this NAND gate output. So, this NAND gate output it has got M is equal to 1 here because it is a logic operation. So, the no NOT gate output is 0. So, 0 to A NAND gate output means it is 1, ok. So, 1 1 for a no ex-OR gate you know the output is 0 this is clear. So, for A naught is equal to 1, F naught is equal to 0.

Now, if A naught is made 0. So, this is made 0, what is changing? What is different here? So, A naught is 0 means this is becoming 0. So, this NOR gate output, now all the inputs are 0 so, this output will now become 1. This NOR gate output is already 1 because of you know the selection lines are 0, right. So, this is 1 1 means this is becoming 0, right. So, this XOR gate 1 of the input is 0 and the other input is 1 because of you know the M being 1. So, NAND gate is output is 1, ok. So, 1 and 0 then this output is 1, is it fine?

So, you can see that this is happening. So, we can see for other cases and also and it works in that manner just.

(Refer Slide Time: 25:18)




Now, what we look at other than this the way the function table it works and all how ALU can be work in tandem with look ahead carry generator so, IC 74181 and IC 74182, ok. So, look ahead carry generator is 182 which we had seen before, right when we discussed first stage. So, here we show you a framework which the is available in the you know manufacturers data sheet if you go to the TI's excess instruments. So, you know website and all you can figure it out.

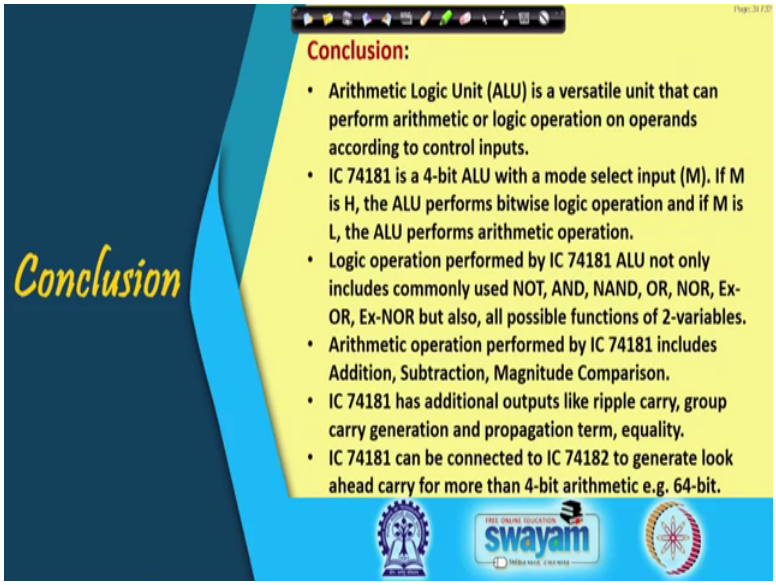
So, where we can see that the addition where we are using 64-bit addition. So, for this 64-bit addition we are using basically 64-bit operation we are doing using 16 ALU each is 4-bit, right each is 4-bit long. So, 74181 as you can see the first level 1 2 3 4 5 this way sixteen such cases sixteen such units will be there, ok.

And, the G n P term we had seen in the ALU the these are the terms that are there. So, four of them can go to 1 IC 74182, right and these IC 74182 is generating C n plus x C n plus y this carries in you know. So, that is getting connected here for each one of in the generation process right. So, this we have seen in the previous case how 782 works rather 182 works, right. So, similarly for this one, similarly for this one; so, four such look ahead carry generator will be there, ok. So, they are again generating this look carry generation and propagation term out of them because the circuit dissimilar, right and then

this will be again fed to next level of IC 74182 which has got which can take for inputs, and then we can get the final carry out of it.

So, basically this is the way you can use 74181 and 74182 and using multiple levels you can get the carry ahead of you know normal ripple carry based addition to that given, ok. And, you have we have already noted that the 74181 also generates it you know normal carrying which can be used for C n plus 4 which can be used for ripple carry addition if it is so required, ok.

(Refer Slide Time: 28:20)




So, to conclude we find that ALU is a versatile unit that can perform arithmetic and logic operation on operands according to control inputs which is coming through the selection lines. It is a 4-bit ALU IC 74181, with the mode select input and depending on whether it is high or low logic or arithmetic operations are done and it includes commonly used NOT and NAND these operations and in fact, it is it can generate all possible combinations of two variables that is 16 possible combinations.

And, arithmetic operation includes addition, subtraction and magnitude comparison by in indirect manner and then it has got additional inputs like ripple carry, group carry generation, propagation term equality and IC 74181 can be connected to IC 74182 you know the a combination can be prepared by which look ahead carry for more than 4-bit arithmetic is possible, ok.

Thank you.