

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

Lecture – 26
Magnitude Comparator

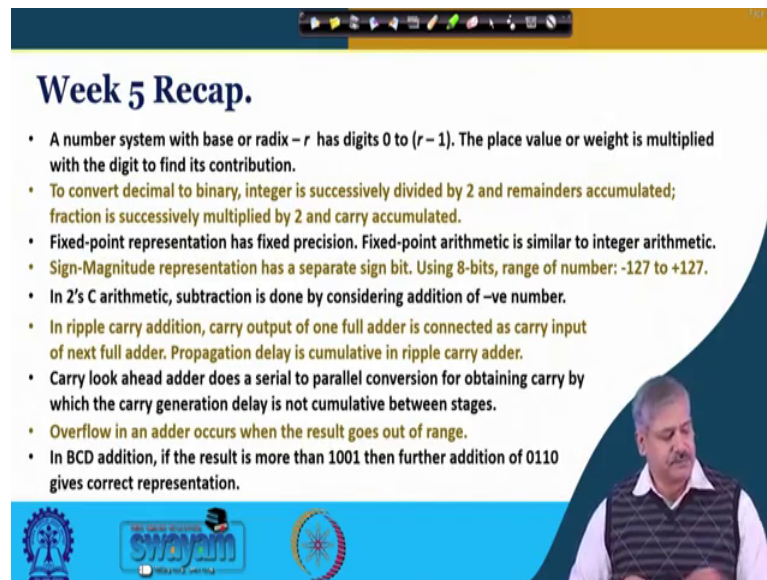
Hello, everybody. We are in week 6 of this particular course. In today's class, we shall look at Magnitude Comparator.

(Refer Slide Time: 00:24)



And, before that we shall have a quick review of what we discussed in week 5.

(Refer Slide Time: 00:27)



Week 5 Recap.

- A number system with base or radix r has digits 0 to $(r - 1)$. The place value or weight is multiplied with the digit to find its contribution.
- To convert decimal to binary, integer is successively divided by 2 and remainders accumulated; fraction is successively multiplied by 2 and carry accumulated.
- Fixed-point representation has fixed precision. Fixed-point arithmetic is similar to integer arithmetic.
- Sign-Magnitude representation has a separate sign bit. Using 8-bits, range of number: -127 to $+127$.
- In 2's C arithmetic, subtraction is done by considering addition of $-ve$ number.
- In ripple carry addition, carry output of one full adder is connected as carry input of next full adder. Propagation delay is cumulative in ripple carry adder.
- Carry look ahead adder does a serial to parallel conversion for obtaining carry by which the carry generation delay is not cumulative between stages.
- Overflow in an adder occurs when the result goes out of range.
- In BCD addition, if the result is more than 1001 then further addition of 0110 gives correct representation.

The slide also features a small video inset of a man in a white shirt and dark vest speaking, and a footer with logos for 'SWAYAM' and other educational institutions.

If you remember, we discussed number system and they representation of numbers in binary and the code that we arrived at were weighted codes in the sense that every place has got certain weight associated with it. When it is binary, before the binary point we had weights like 1, 2, 4, 8, 16 so on and so forth and after the binary point it was half, 1 upon 4, 1 upon 8, 1 upon 16, so on and so forth, ok.

And, to convert decimal to binary the integer part was successively divided by 2 and remainders were accumulated and for the fractional part; part coming after the binary decimal point it was successfully multiplied by 2 and the carry was accumulated. And, we had seen that fixed point arithmetic has fixed precisions and it follows integer arithmetic, and for negative number representation sign magnitude was discussed and also 2's compliment, 1's compliment all those things were there and then we discussed those compliment arithmetic.

And, then to design circuit so, for addition and subtraction we saw ripple carry addition and how we can convert a ripple carry other to a subtractor, ok. And, then we discussed carry look ahead adder for fast addition where the carry generation process was converted to parallel from serial by unlooping the iterative equation. And, then we discussed overflow detection, how to detect overflow if the number goes out of the range and then we also discussed BCD addition and the logic behind, it how to arrive at those logics.

(Refer Slide Time: 02:27)

Magnitude Comparison by Subtraction

- Magnitude comparison finds if (i) one number is same as the other number (identity / equality) or (ii) greater / smaller than the other number.
- An adder circuit performing 2's C subtraction ($X - Y$) can be used for this.
 - If $X = Y$ and $X > Y$ then $C_{out} = 1$.
 - If $C_{out} = 0$, then $X < Y$.
 - If C_{out} is 1 and any of $S_j = 1$ then $X > Y$, else $X = Y$.

$(X < Y) = C_{out}'$ $(X = Y) = C_{out}' \cdot S_3' \cdot S_2' \cdot S_1' \cdot S_0'$ $(X > Y) = C_{out} \cdot (S_3 + S_2 + S_1 + S_0)$

So, as I said in today's class we shall discuss magnitude comparison. So, magnitude to be compared is between two numbers and the numbers I mean whenever we talk about comparison there will be first thing that occurs to our mind why do not we subtract one number from the other. Say, one number is X another number is Y. So, if X minus Y is 0, then it is equal the numbers are equal and if X minus Y is positive the number is X is greater than Y; if X minus Y is negative then the number is X is less than Y, ok.

This is the thing that can that will occur to us and of course, we can get a magnetic comparison done by subtraction process, ok. So, that we can do and for that we can have a subtractor circuit developed from 2's complement by using 2's complement arithmetic and full adder circuit. So, in that the way we have seen it before how the subtraction works using 2's complement arithmetic. So, when X is equal to Y and X greater than Y, we see that carry out generated is 1, ok.

So, we have to refer back to which is complement arithmetic by which the subtraction is done those examples we studied that we had done in the previous class. From there we take the important point that if X is equal to Y and X greater than Y then carry out that is generated is equal to 1 and it carry out genera the generated is 0, then the number is negative that is X less than Y, I mean the sub result is negative. So, X is less than Y, ok. So, remember that, ok.

Then, from that we can get the logic for X less than Y very clear easily, that if the carryout is 0 no carry is generated then the result is X less than y. So, in that case we can the corresponding outputs X less than Y we can directly you know get from C out type complement of carryout, ok. And, when carry is generated then there are two possibilities one is X is equal to Y, another is X greater than Y, right. So, X is equal to Y will occur when all these sum bits are 0, right.

So, at the time we can follow this particular logic, that all the each of the sum bits are 0 no digit is 1. So, that means, that the numbers are equal result is 0 and carry it carryout generated is 1 that is following 2's complement subtraction. And, if any of the sum bit is 1 any of the sum bit is 1, so, that is that follow order logic by which we can hand it with C out, the carry is all already generated. So, that gives you X greater than Y log, ok.

So, this is the basic idea and of course, when you are generating multiple output function you can see that whether output of one can be used I mean some intermediate output can be used in two places two different places. So, accordingly certain minimization you can think of otherwise the basic logic in which this what we are arrived at.

(Refer Slide Time: 06:17)

1-bit Magnitude Comparison

- Magnitude comparison can be done without generating addition / subtraction result.

Input		Output		
X	Y	X > Y	X = Y	X < Y
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

Logic Circuit Diagram:

$(X > Y): G = X \cdot Y'$
 $(X = Y): E = X' \cdot Y' + X \cdot Y$
 $(X < Y): L = X' \cdot Y$

(If NAND gate is not used, two NOT gates will be required to get X' and Y')

$X' \cdot Y' + X \cdot Y = (X' + Y) \cdot (X + Y)$
 $= (X \cdot Y)' \cdot (X' \cdot Y)'$
 $= [(X \cdot Y) + (X' \cdot Y)]'$

So, but when we look at magnitude comparison and our objective is to generate only X greater than Y, X is equal to Y and X less than Y we do not actually need the subtraction result that is s 3, s 2, s 1, s naught, ok. So, we are doing more than what is expected from us in terms of magnitude comparison of two numbers, isn't it? So, can we have you

know can we use less (Refer Time: 06:50) you know another kind of you know circuit by which we generate only the result that compares two numbers, ok.

So, we will go 4-bit number comparison the way we had seen before or larger number of bits. So, begin with 1-bit magnitude comparison. So, two numbers are of 1-bit, ok. So, if X and Y are of 1-bit. So, the number possible you know the possible combinations the way we can compare them when X is equal to 0, Y is equal to 0; X is equal to 1, X is equal to 0; Y is equal to 1, X is equal to 1; Y is equal to 0 and X is equal to 1 and Y is equal to 1. So, these are the four possibilities by which you can do the comparison.

And, in these case you can see that 00 X and Y both are 0; that means, X is equal to Y. So, this will be 1, neither it is greater than Y nor X less than Y. So, they are 0 and 0 clear. So, X is equal to 0 and Y is equal to 1. Of course, Y is more and X is less. So, X is less than Y that is equal to 1 and rest 2 are 0. X is equal to 1 and Y is equal to 0, so, X greater than Y is 1 and when both of them are 1 that is they are equal, so other two are 0; is it fine?

So, by this we can figure out that X greater than Y is occurring for this combination. So, we can get it by $X'Y$ prime. So, G we are giving G for greater than X greater than Y. And, then X less than Y, right that is occurring for this particular case sorry, this particular case. So, that is X prime Y, X is equal to 0 and Y is equal to 1, clear and finally, X is equal to Y is occurring for these two cases, the other two cases left is this one and this one. So, this is X and Y both 0. So, that is X prime Y prime and X and Y both 1, so, XY so, you add them up sum them up to get the final expression. So, these are you know the basic logic by which you can get 1-bit comparison done. So, we do not need to do other things, ok.

Now, since we are generating $X'Y$ prime and X prime Y, right do we need to additionally generate X prime Y prime XY , so; that means, two other end AND gates and then OR it up, or we can make use of this and see whether you know another you know gate not more than one gate is required to get the final X is equal to Y output generated, ok so, that we can investigate.

So, for that we can see that X prime Y prime ORed with XY we can write these as X prime or Y and get with X or Y prime you can see that X prime and X it will become 0.

So, X prime Y prime will come from this particular term first this thing and YX will be there from these two and YY prime will be 0, ok.

So, once it is there then this particular term X prime Y came from DeMorgan's theorem you can write in this manner and this one also in this manner and finally. So, we have got our XY prime and X prime Y, and then again, we apply on this particular term DeMorgan's theorem. So, we can get a NOR relationship. So, a NOR of this two, these two term which are already generated or need to be generated we get X is equal to Y logic, we do not need to generate separately these two, ok.

And, the other thing that we can see that we can generate these XY prime and X prime Y by NOT gate and then AND gate which is possible, ok, it all already in some cases you are generating X you know inversion inverse of XX complement somewhere or Y from given you can make use of it or we can generate it. But otherwise also we can see that if we have a NAND gate between X and Y we are getting XY NAND which is nothing, but X prime plus Y prime, ok.

So, when you AND X with that so, XX prime will become 0. So, XY prime will be generated here and it will AND Y with that right, with this one so, X prime will be generated and Y and Y prime will become 0. So, we can get generated this one, you do not need then one and you know addition of those two NAND gates rather one NAND gate will do. So, by this we can get this made clear. So, that is for 1-bit comparison.

(Refer Slide Time: 12:07)

2-bit Magnitude Comparison

Numbers to be compared: $X: X_1X_0$ and $Y: Y_1Y_0$

Define:

$G_0 = X_0Y_0'$
 $E_0 = X_0'Y_0' + X_0Y_0$
 $L_0 = X_0'Y_0$

$G_1 = X_1Y_1'$
 $E_1 = X_1'Y_1' + X_1Y_1$
 $L_1 = X_1'Y_1$

$X > Y$:

- If $X_1 > Y_1$
- If $X_1 = Y_1$ and $X_0 > Y_0$

$(X > Y) = G_1 + E_1G_0$

$X = Y$:

- If $X_1 = Y_1$ and $X_0 = Y_0$

$(X = Y) = E_1 \cdot E_0$

$X < Y$:

- If $X_1 < Y_1$
- If $X_1 = Y_1$ and $X_0 < Y_0$

$(X < Y) = L_1 + E_1L_0$

Other than two 1-bit comparators, a logic circuit to generate final outputs.

The slide also features a logic circuit diagram with inputs X_1, X_0, Y_1, Y_0 and outputs G_1, E_1, L_1 . Handwritten annotations show a truth table for the 2-bit comparison:

X_1	Y_1	X_0	Y_0
1	0	0	1
0	1	0	1
0	0	1	0
0	0	1	1

Now, let us look at how to do 2-bit comparison, right. So, for 2-bit comparison then the number we are having is $X_1 X_0$ and $Y_1 Y_0$. So, X_1 is the more significant bit and X_0 is the less significant bit accordingly it is for Y , ok. Now, we know that X as a whole the number will be greater than Y , if X_1 is greater than Y_1 , isn't it. More significant bit; so, if X_1 is more than Y_1 , what is the option then; that is X_1 is 1 and Y_1 is 0, then irrespective of the rest of the bit 0 0, 0 1, 1 0, 1 1 all the time right you are having X greater than Y all these cases. So, this is your X_1 and this is your X_0 and Y_1 and Y_0 , and this is your Y_1 and this is your Y_0 right because more significant bit is larger than this, ok.

So, in this case we do not need to worry about the other things, ok. So, when X_1 is equal to Y_1 that both of them are 0 or both of them are 1, then whether X is greater than Y or not is decided by the you know X_0 Y_0 comparison, right. So, this is at that time only you look at it. So, this is we know between you know comparison of two numbers.

So, basically we are looking whether X_1 is greater than Y_1 and also we are looking at X_0 is greater than Y_0 where X_1 is equal to Y_1 . Now, if you remember the when had compared 1-bit generation, so, X_1 greater than Y_1 we can get if we do 1-bit comparison of X_1 and Y_1 . Whether X_1 is equal to Y_1 we can get by 1-bit comparison of again X_1 Y_1 where the equality impart is looked at X is equal to Y output is looked at and X_0 Y_0 we come from 1-bit comparison of the lower signal we can make X_0 and Y_0 .

So, what we do? We generate two 1-bit comparator, we use two 1-bit comparator, right. So, one is to generate G_0 , E_0 and L_0 by the same equation that we had done before and another is for G_1 , E_1 and L_1 , and so, we write. So, basically this is the generic 1-bit comparison you know in a circuit. So, instead of X and Y we are writing X_i , Y_i to indicate the bit position. So, 0 position you know position 1 and position 2 or so, ok. Is it clear? So, this is the way circuit is made.

Now, if we look at this X greater than Y thing, so, X_1 greater than Y_1 is your G_1 the it will be given by G_1 , right. So, if G_1 is high G_1 is true then X will be greater than Y irrespective of the other things, right. When G_1 is not high, right then two possibilities are there, that the number is equal or number X_1 is less. So, if the number is equal E_1

then you look at whether these lower significant bit G naught is high, I mean or not lower significant of X is more than Y or not, ok.

So, this way you get the logic for X greater than Y. So, similar thing you will get for X less than Y, where we look at L 1 if X the higher significant weight is lower of course, it will be less and if they are equal then you look at the significant bit lower significant bit whether it is less, then will be as a whole the number will be less and when it will be equal when both E 1 and E naught are equal, then only it will be equal, is it ok. So, two 1-bit comparator and it is additional logic circuit will give me 2-bit comparison, ok.

(Refer Slide Time: 16:43)

n-bit Magnitude Comparison

Numbers to be compared: $X: X_{n-1}X_{n-2} \dots X_1X_0$ and $Y: Y_{n-1}Y_{n-2} \dots Y_1Y_0$

$(X > Y) = G_{n-1} + E_{n-1}G_{n-2} + E_{n-1}E_{n-2}G_{n-3} + \dots + E_{n-1}E_{n-2} \dots E_1G_0$

$(X = Y) = E_{n-1}E_{n-2} \dots E_1E_0$

$(X < Y) = L_{n-1} + E_{n-1}L_{n-2} + E_{n-1}E_{n-2}L_{n-3} + \dots + E_{n-1}E_{n-2} \dots E_1L_0$

With cascading inputs coming from another block with lower significant bits:

$(X > Y)_{out} = G_{n-1} + E_{n-1}G_{n-2} + \dots + E_{n-1}E_{n-2} \dots E_1G_0 + E_{n-1}E_{n-2} \dots E_0(X > Y)_{in}$

$(X = Y)_{out} = E_{n-1}E_{n-2} \dots E_1E_0(X = Y)_{in}$

$(X < Y)_{out} = L_{n-1} + E_{n-1}L_{n-2} + \dots + E_{n-1}E_{n-2} \dots E_1L_0 + E_{n-1}E_{n-2} \dots E_0(X < Y)_{in}$

Handwritten example:
 $X: 0011$
 $Y: 0010$
 $X > Y: 1110$

Now, if we extend it we can get n-bit comparison which include 4-bit comparison the way we have seen subtraction and all of two 4-bit number as before. So, in this case X greater than Y and the numbers are X naught to X n minus 1 and Y naught to Y n minus 1 and we are looking at as a whole X greater than Y or equal to Y or less than Y or not, ok. So, we will be having n number of 1-bit you know comparison circuit the YX seen before and then we look at the most significant bit G n minus 1 if this is high then of course, X greater than Y, right.

If it is not high low then if the most significant these things equal, if it is less than the most significant bit of X is less than then you do not you know need to go to next step. If they are equal then you look at the next significant bit. If the next significant bit is also equal that is E n minus 1 and E n minus 2. So, that tells that both the most significant bit

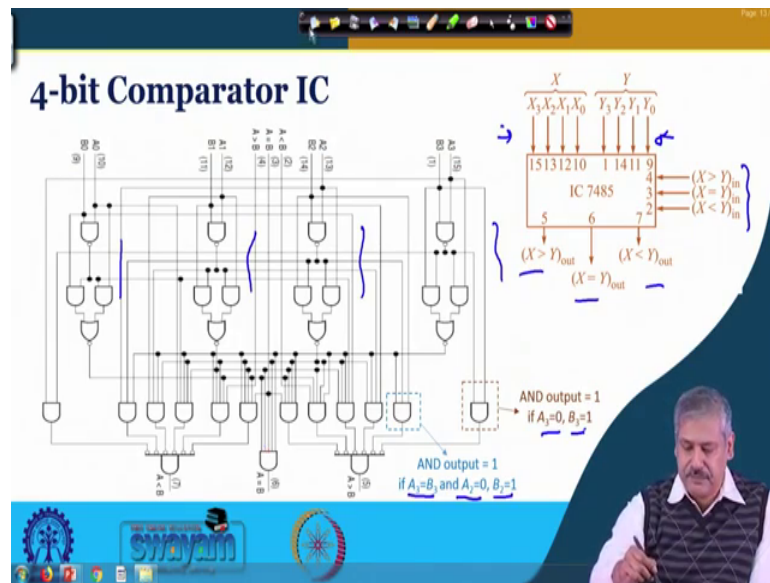
and next most significant bit they are equal. Then you look at the next significant bit, with that that is more than the other or not basically we are talking about in these case say 0 0, this is also 0 0 this is X and this is Y, right.

So, both are equal then you are looking at whether X this particular position is more than this or not for the other case that when both are 1, right then whether this is 1 or not. So, this is where E 1 when you are discussing we are looking at this case. So, if there again equal if you see that they are equal, so, it could be 0 0 could be 1 1, then you look at the next case whether this is 1 or this is 0. So, this is the way the equation can be understood, right. X is equal to Y will come in this manner and X less than v Y we look at where this $L_n - 1$ $E_n - 1$ $L_n - 2$. So, this is the similar to the other place greater than case is it.

Now, when you develop the n-bit circuit will be a 4-bit comparator which we shall see here later. So, it might require that you are looking for 8-bit comparison ok, but you have got you know 4-bit comparator IC or 4-bit comparator circuit available with you. So, you would like to cascade that you need like to connect to another IC, is it not. So, that is what your intention will be. So, in that case you need to have option for connecting inputs, ok. So, in this case when you do that you do it this way that the input is coming from lower significant bits. So, current IC is current circuit is comparing the more significant bits.

So, what we will do? So, everything is remaining same then all current one are equal; more significant bits are equal, then you look at whether X is greater than Y that is coming from the lower stage lower significant you know bit. So, it is greater than Y in or not, and if all are equal if X is equal to Y in is equal then the as a whole it will be equal, right and similarly for the less than cases. So, now, we are putting a out and in prefix in these cases because we are now in a position to accept output of comparison from lower significant bits from another logic circuit, ok.

(Refer Slide Time: 20:57)



Now, we come to a 4-bit comparator IC which is used in practice right and which is IC 741 IC 7485 and you can see what is there this is a 4-bit comparator IC, ok. So, these are the X 4-bits and these are the Y 4-bits these are the inputs that I said that it can come from lower significant bits where you know the lower significant bits are getting compared in another IC or another circuit and these are the output of these stage, which also considers the input, right.

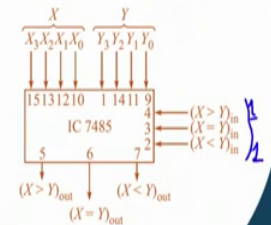
And, if you look at the circuit these are all these locks are 1-bit comparator we can see over here, right and we have used you know these bubbles naught option. So, we can just compare and see it. For example, this particular and output this particular and output is 1 when A 3 is equal to 0 and B 3 is equal to 1. How is it that? You can see that if A 3 is equal to 0, this output will be 1, right and B 3 is equal to 0, right B 3 is equal to 0 then this is B 3 is equal to 1 then this one is also coming. So, both of them are 1. So, this and gate output will be 1. So, this one this is 0. So, this output will be 1, sorry this output will be 0 because this is AND gate and at the input of it 1 it is inverse. So, that is 0 is there, ok.

So, if B 3 is more than A3 this cannot be 1. Similarly if you look at this particular AND gate, so, if it if A3 is equal to B 3 if you look at the connection A 2 is equal to 0, B 2 is equal to 1, then this output will be 1. So, that also ensures that in that case in that case the output will be 0. So, this is the way the circuit has been developed, right.

(Refer Slide Time: 23:08)

IC 7485 Function Table

COMPARING INPUTS				CASCADING INPUTS			OUTPUTS		
A3, B3	A2, B2	A1, B1	A0, B0	A>B	A<B	A=B	A>B	A<B	A=B
A3>B3	X	X	X	X	X	X	H	L	L
A3<B3	X	X	X	X	X	X	L	H	L
A3=B3	A2>B2	X	X	X	X	X	H	L	L
A3=B3	A2<B2	X	X	X	X	X	L	H	L
A3=B3	A2=B2	A1>B1	X	X	X	X	H	L	L
A3=B3	A2=B2	A1<B1	X	X	X	X	L	H	L
A3=B3	A2=B2	A1=B1	A0>B0	X	X	X	H	L	L
A3=B3	A2=B2	A1=B1	A0<B0	X	X	X	L	H	L
A3=B3	A2=B2	A1=B1	A0=B0	H	L	L	H	L	L
A3=B3	A2=B2	A1=B1	A0=B0	L	H	L	L	H	L
A3=B3	A2=B2	A1=B1	A0=B0	H	H	L	L	H	L
A3=B3	A2=B2	A1=B1	A0=B0	L	L	L	H	H	L
A3=B3	A2=B2	A1=B1	A0=B0	X	X	H	L	L	H



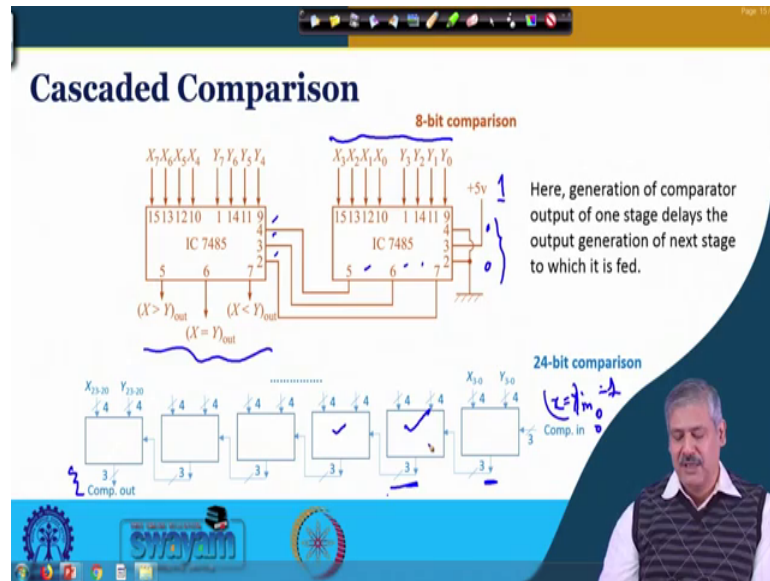
And, if you look at the function table of IC 7485, as we have discussed it is similar if A 3 is greater than B 3 right, irrespective of what are the other you know inputs as well as the cascaded inputs the output will be A greater than B, and if A3 is less than B 3, then it will be less than B 3, right. And, when A 3 is equal to B 3 at that time you look at whether A 2 is greater than B2 or not if A 2 is greater than B 2 irrespective of the other cases a greater than B, ok. So, this is follows what you know what basic comparator logic should follow, ok.

Now, comes you know when all of them are equal all these current bits. So, this is the case A 3 B A 2 A 1 A naught are equal to B 3 B 2 B 1 B naught; so, at that time if A greater than B right, then at the input, then A greater than B at the output. So, it is now looking at the lower significant bits it is comparison of that. Similarly, others and where A 3 is equal to B 3 then this is if this is high then this will be a equal to B.

Now, it is not expected that at the input side ok, both X greater than Y and X less than Y both will be equal to 1, because either it is high or it is less, ok. So, that is what is expected, right. But, in case it happens for some you know reason the circuit is made in such a manner even though it is not expected that if both of them are high then the output in this case A greater than B. So, both of them are low if both of them are low then both of them are high, this is what is there which we shall see can be used to our benefit in some configuration, ok.

So, this is not is going to happen in normal cases, but according to the logic circuit that has been developed this is what the function table gives us as such this is a don't care at the input side, ok. But, because this is there in the final truth table we can see that it can be put to some you know useful now configuration of comparator circuit for larger number of bit comparison and that part we shall take up in subsequent slides.

(Refer Slide Time: 25:55)

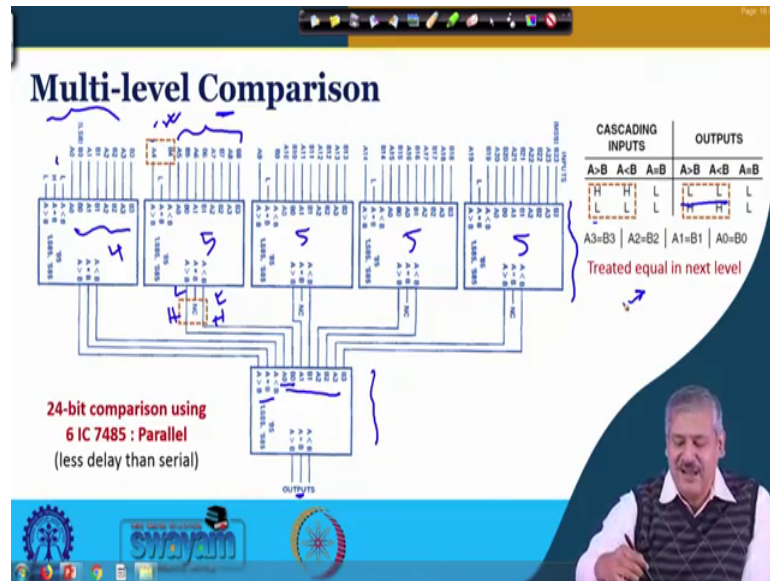


Now, as I said it can be used for comparing more number of bits because of the option that is available as input. So, if you look at 8-bit comparison. So, of course, the lower significant bits need to be placed here and here this X is equal to Y input is kept high the others are kept low, is greater than Y and less than Y, clear? And, so, these outputs X greater than Y equal to Y and X less than Y whatever is getting generated here in this three cases are fed as input to the next IC and the final output is taken from here, ok, is it clear?

Now, if you look for say 24-bit comparison, how will you go about? So, you will be requiring you know 6 such ICs. So, the first case we will be connecting in this manner, right that is the input pin number 3 that is X equal to Y in that will be high and other two are will be 0. So, for the first one, right and after that you are connecting the Y you are connected it. So, final output will be generated from here, right and in this case the final output generation will depend on the logic operation done in previous cases.

So, basically this IC comparator will wait till you know this previous comparator generates its output, this will wait till this one generates its output. So, that way you see the propagation delay means you know cumulative in nature, ok. This comparison result will come, but it will be the delay of which of this state will get a data which is how it is going to work.

(Refer Slide Time: 27:56)



But, we can have another configuration when we are using the same you know six comparators only for 24-bit comparison. So, there we can put we can put five such comparators here in the first place in one stage which will be the which is giving comparator output after stage unit delay that happens within the comparator and after that it is another level. So, it is a two level comparison, right and you see how intelligently the connections has been made, as such for 5 you can get 5 into 4, 20 inputs, but how 24 inputs have been placed there.

So, for this you know this significant 4-bits are connected the standard way, ok. So, the A is equal to B is high and the other two are low, right. So, there goes 4, 4 inputs and we have another 20 inputs 2 plus, right. So, that is placed 5 of them are placed in rest four in a first stage. Here 5, 5, 5, and 5. How? And, then this outputs I put over here. So, the first one is these two and rest this four A naught B naught to A3 B3 are coming from these four, ok.

So, how does it work? So, in the first place this $A_4 B_4$ we are placing it here and A_4 if it is greater than B_4 , then it is fine. It is A greater than B it will go in that manner. So, the corresponding output will be generated, right? A greater than B , otherwise if B_4 is more than A_4 , so, if the A less than B will be generated if other bits remaining same, if all of them are remaining same because this is a lowest you know lower significance compared to other.

Now, if they are equal all these are same, they are equal that is where the problem could have been there, right. But, if they are equal we have seen in the previous case in the function table as I was telling there if they are equal you see the outputs are also equal, if both of them are high and both of them are low output are also LL and HH in this particular case. So, both of them will be low or both of them will be high in that case.

So, when they are fed as A naught B naught here so, both of them are high and both of them in low. So, they do not contribute to the comparison that is equal and compare contribute in the sense that input is compare you know equal. So, if everything else remains same and the output is gene you know depends only on A_4 and B_4 comparison so as to say, ok, then if both of them are low and both of them high will generate same high-high and low-low output here and over here same high-high and low-low and the output will be A is equal to B in the final case place. So, the fifth bit can be inserted in that in a here like this.

So, similarly for the other cases; so, 4 into 5, 20 and this is 4, ok. So, that is a clever use of this thing and by which we can see this because of the parallelism the propagation delay can be reduced.

(Refer Slide Time: 31:25)

The slide features a dark blue background on the left with the word "Conclusion" in a yellow, cursive font. The main content is on a light yellow background. At the top right of the slide, there is a small toolbar with various icons and a timestamp "Time: 17:18". Below the toolbar, the word "Conclusion:" is written in red. A bulleted list follows, detailing methods for magnitude comparison and the characteristics of the IC 7485. At the bottom of the slide, there are three logos: the Indian Institute of Space Science and Technology (IIST) logo, the "swayam" logo with the text "FREE ONLINE EDUCATION" and "INDIAN INSTITUTE OF SPACE SCIENCE & TECHNOLOGY", and the logo of the Department of Space, Government of India.

Conclusion:

- Magnitude comparison of two numbers can be done by subtracting one number from the other and developing suitable logic circuit to generate $X>Y$, $X=Y$, $X<Y$ outputs.
- A direct approach for magnitude comparison of two numbers can avoid subtraction. In this, place value of the bits being compared is useful.
- IC 7485 is a 4-bit comparator which has inputs that can be connected to outputs of another IC comparing lower significant bits of two numbers.
- Delay of every stage adds up when larger number of bits are compared with IC 7485 is connected in serial.
- There can be parallel, multi-level arrangement of IC 7485 which can give lower delay.

So, with this we come to the conclusion of the particular discussion on comparator. Magnitude comparison of two numbers can be done by subtracting one number from the other and developing suitable logic circuit to generate this X greater than Y , X is equal to Y , X less than Y output. We shall see in the next class the ALU basically arithmetic logic unit does it in that manner only; we do not have separate comparator circuit within it. A direct approach for magnitude comparison of two numbers can avoid subtraction in this place value of the bits being compared is useful, I mean if it is more significant bit they differ then we can arrive at the decision from that directly.

IC 7485 is a 4-bit comparator with option for connecting input from lower significant bits that are getting compared in another IC 7485 or logic circuit. And, delay at every stage can add up for larger number of bit getting compared when you connect them in serial, but there can be an option of parallel multilevel arrangement, ok.

Thank you.