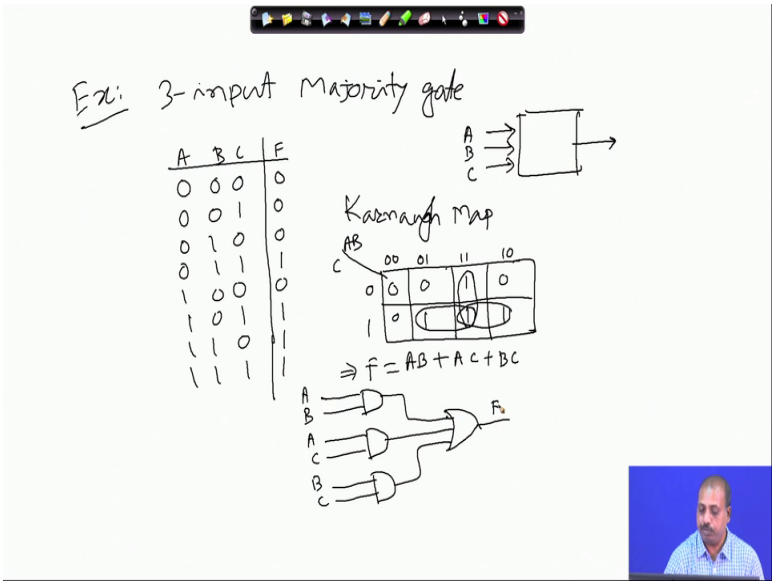
Microprocessors and Microcontrollers
Prof. Santanu Chattopadhyay
Department of E & EC Engineering
Indian Institute of Technology, Kharagpur

Lecture - 03
Introduction (Contd.)

So, as far as the design of combinational logic is concerned particularly this digital combinational circuit. So, we take help of certain design techniques. So, we will take a small example and elaborate it and that way you can you will be able to design any other circuit as well.

(Refer Slide Time: 00:34)



So, the example that we take is something called a three input majority gate, the idea is that we have got we have to design block where there are three inputs if this is the block that we design now there are three inputs say A B and C and there is a output. Now, this block we will produce a value 1 when the majority of it is inputs are equal to 1 otherwise it will output a 0. So, any combinational design it starts with the truth table of the design, so if we take the truth table concentrate to draw the truth table, so we have got these three inputs A B and C and the output has F. So, if you take this combination 0 0 0 0 0 1 0 1 0 0 1 1 1 0 0 1 0 1 1 1 0 and 1 1 1 then as at the as the statement of the problem said the majority of the inputs should be reflected so this majority is 0 so this is 0 this is also 0 this is also 0 so this is 1 this is also 0 this is 1 this is 1 and this is 1.

Now once we have drawn this particular table or the truth table the next important thing that we have to do is making something called a karnaugh map. So, we have to arrive at the logic expression that can that can be used for this particular block, so that is done by means of karnaugh map. So, for a three variable karnaugh map, so it will look something like this so on one side we put the variables A B other side we put the variable C and these are the combinations of 0 0 0 1 1 1 and 1 0 so this is 0 and this is 1.

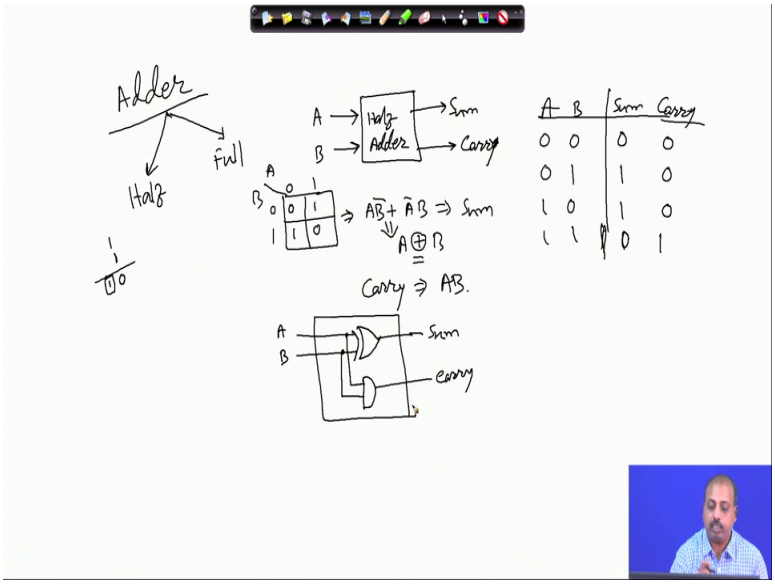
Now, where ever this is a the output is for 0 0 0 0 it is 0, for 0 0 1 this is again 0 then 0 1 0 is also 0, 0 1 1 is 1 then 1 0 0 is 0, 1 0 1 is 1, 1 1 0 is 1 and 1 1 1 is 1 so this is the karnaugh map. Now after drawing this karnaugh map from our digital logic classes, so you can find out that we have to make the groups of this ones and we will try to make a as large a group as possible. So, in this particular case so we can make groups like this so this is one group so this is another group and this is another group.

So, if we do this then this will be giving me the expression. So, when you are taking say this group, so this is A B this is a B then if you are taking say this group then it is the way the bit A is not changing A C and if we are taking say this group then it is B C; so A B plus A C plus B C. So, looking at the expression you can immediately find out that this is correct because our objective is to find majority of the inputs to be 1 then only the output will be 1 and each of the here the A and B both are 1. So, naturally that is the majority outputs and A and C both are 1 and here B and C both are 1. So, that is the majority function has been represented properly.

Now you can draw the corresponding digital logic circuit combinational circuit by if this is these are AND gates. So, for realizing this product comes A B, A C and B C so you can take it like this. So, A B A C and B C and there is an OR to sum them up so this is the OR gate where I am connecting all of them so this is the function F. So, this way for any digital designs if it is a combinational design so you can take help of this truth table and karnaugh map to arrive at the logic expression, sometimes it is this karnaugh map problem is that if the size the number of inputs is large more than four it is difficult to draw the karnaugh map on a piece of paper. So, it becomes a problem becomes problematic now of course there are other methods by which you can do all these type of minimizations or optimization so that you can find out in digital logic classes.

So, next thing that we will look into is the sequential part so sequential before sequential part. So, there is another very common circuit that is used in microprocessors and microcontrollers which is the adder circuit.

(Refer Slide Time: 05:48)



So, we will look into this adder circuit we carefully because this will be again used repetitively in any of these processors. So, what does then added do so, adder it will add the bits so this adder you can classify into two as far as the design is concerned so you can classify it into two classes, one is called half adder and the other is called full adder. This half adder it just adds to bit so this half adder block if you try to see this half adder it has got two inputs A and B the bits to be added and it produces two output one is sum other is carry so this is sum and this is carry ok.

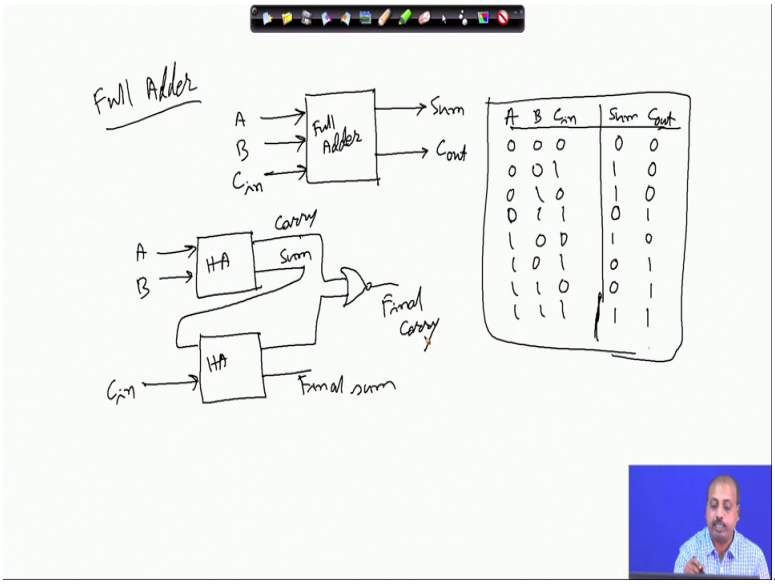
So, the truth table so this is a combinational circuit so I can represent it in the form of a truth table. So, if I want to do this thing so this A B and I have got the sum and carry as the two outputs, now combination as 0 0 0 1 1 0 and 1 1. So, this sum so in this case both are 0 here the sum is 1, carry is 0 here also the sum is 1 carry is 0 in this case sum is 1 and the carry is also 1, sorry this sum should be 0 because if I am adding 2 bits 1 and 1 when I am adding them I will get the result as 0 and with a carry generated so this is the carry and this sum should be 0.

So, you see that if you if you represent it in the form of a karnaugh map for the sum part so it will be like this A B. So, for 0 1 and 1 0 the output is 1 so these two are 1 and these

two are 0, so you cannot group this 1's in any way so you get the expression as A B bar or A bar B. So, that is nothing but A XOR B. So, this symbol we use for XOR so A XOR B, for the carry part is you see that it is it is basically the and function so this is the sum this is the sum and the carry part is nothing but A B the carry part is A B.

So, I can realize this half adder like this so there is 1 XOR gate, where this A and B inputs are coming A and B inputs are coming here there is one AND gate where this A and B inputs will be fed. So, this is the sum output and this is the carry output. So, we have got this as the half adder block ok, now this half adder the problem is that it can add only 2 bits now if you want to add 3 bits then we get something called a full adder.

(Refer Slide Time: 09:34)



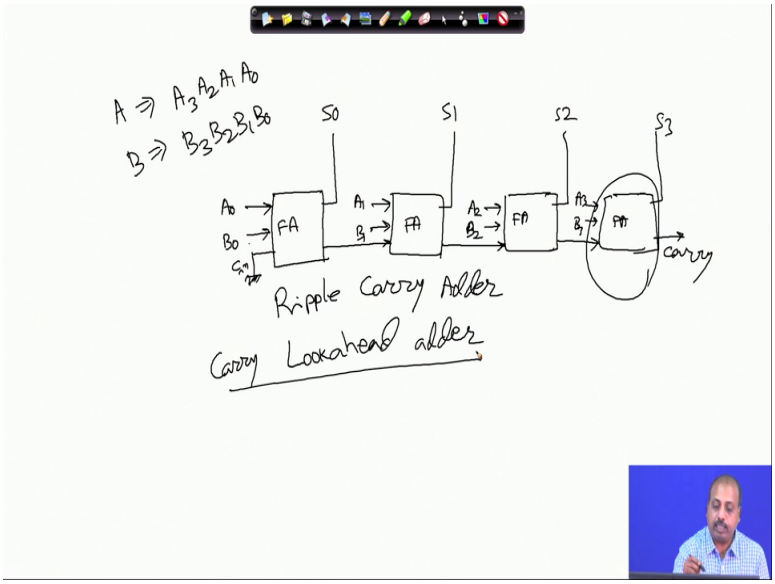
So, the full adder the full adder part so as far as the block is concerned. So, it has got the inputs like this A B there is a third input which we conventionally call carry in to indicate that this is the carry input from the previous stage of the adder and the sum we have got sum output and for this stage we have got the carry which we called carry out that is the carry of the current stage, so this is this is this the full adder.

Now, if you represent in the form of truth table so it will be like this A B and carry in and here you have got this sum and carry out, now if it is 0 0 0 0 0 0 1 0 1 0 0 1 1 1 0 0 1 0 1 1 1 0 1 1 1 so these are the combinations that we have, now for that so here the sum output will be 0 carry will also 0, here some will be 1 carry will be 0 here also some will be 1 carry will be 0 here sum is 0, but carry is 1 because when you add this A B and C in

so this 1 plus 1 will be 0 and the carry will be generated so carry output will be 1, here again the sum is 1 sorry here again the sum is 1 and carry is 0. Now here again you have got sum as 0 and carry as 1 here again you have got sum as 0 and carry as 1 and here you have got sum as 1 and carry also as 1. So, we have got the truth table like that so you can design you can design this function you can design this function and get the corresponding internal circuit the other way of doing the same operation is by means of using the half adder, like we have got one half adder that adds the numbers A and B and it produces the carry and the sum outputs.

Now, this sum output it can be put into the next stage of half adder so this sum comes here and the carry input that we have here the C in the; C in can be at put here and this gives me the final sum so this is the final sum and this carry you see I can take one or nor gate and I can take this carry generated like this. So, this is the final carry so ok this way I can have the final carry generated by doing the by doing that using a half adder so we can get this thing.

(Refer Slide Time: 13:34)



Now, so this way I can add two 1 bit number with the carry also now if you want to add numbers that are that has got the that have got more number of bits like say suppose I want to have say two numbers A and B where is say 4 bit so A 3, A 2, A 1, A 0 these are the 4 bits and the number B it has got again 4 bits B 3, B 2, B 1, B 0 and I want to add them.

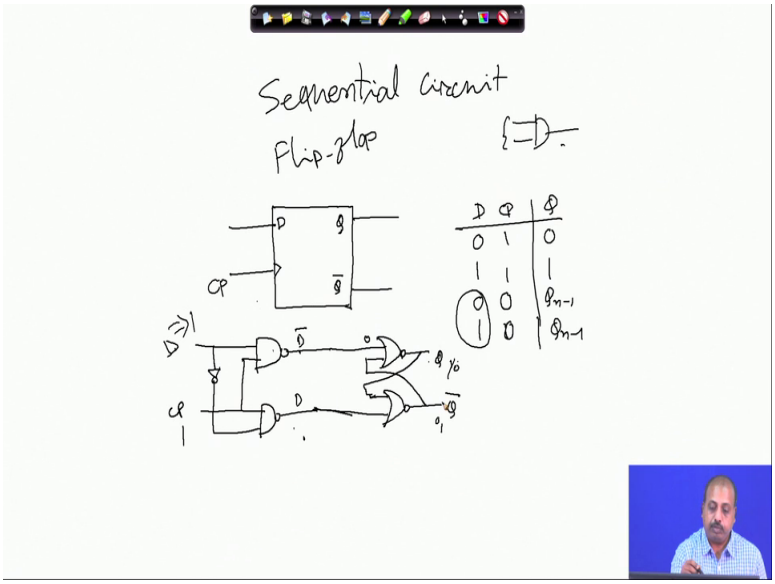
Now, how to do this addition so you can add you can have full adders for doing this job like you can have this the first stage of the full adder so it can be given the responsibility to add this A 0 and B 0 it can be given the responsibility to add A 0 by and B 0 so this is a full adder. So it will produce a sum which we called the 0 bit of the sum S 0, next stage of the full adder it can add the digit it can add the bits A 1 and B 1 and accordingly it can produce the bit S 1. The next stage can add A 3 and B 3, A 2 and B 2 and it can produce the output as S 2.

And finally I can have this A 3 and B 3 going there and producing the output S 3. And there is something the carry part so for this one the carry in is made equal to 0 so this is the carry in so this is grounded, because it does not have any value so this is no carry input here. Now this carry output can be fed to the carry input of the next stage similarly this carry output can be fed to the carry input of the next stage this can be fed as the carry input of the next stage and finally so from the last stage you can generate the carry signal for this entire 4 bit addition.

So, this way we can utilize this full adders combine them to a large size and accordingly get the adder circuit for larger number of bits, of course this is the this type of adder that I have discussed so this is commonly known as ripple carry adder, they are known as ripple carry adder because the carry bit is actually passing through all the full adder stages, now so that way the delay of this circuit is quite high and most of the processors they will use some improved version of this adder ok.

So, if you look into some book on V L S I so we can find out what are the different types of adders that are available, so the most low the fastest adder that we have is the carry look ahead adder the carry look ahead adder. So, where this carry bits have computed beforehand and that way it does not meet to they this so like this stage here so these stage so it has to wait for the other the previous stages to finish off and then only the carry is available here so that this full adder can do the operation, whereas in carry look ahead adder so this carry propagation is not required so it is calculated beforehand and that way the operation becomes faster though the amount of logic involved in the circuit is more so this way we can go for this adder circuits.

(Refer Slide Time: 17:27)



Next we look into another class of components that will be required in a in the design of these processors which are known as the sequential circuits. The basic block of any sequential circuit is called a flip flop ok because flip flop is a module that can remember the previous stage like unlike combinational circuit so like in case of say one AND gate So, if we take an AND gate so it AND gate whatever be the current input based on that it is giving the output so it is not remembering anything about what was the past output of the AND gate, but sometimes it is necessary but we need to remember the information about different the previous stage of the circuit, so that way it can be done by means of another class of circuit called flip flops.

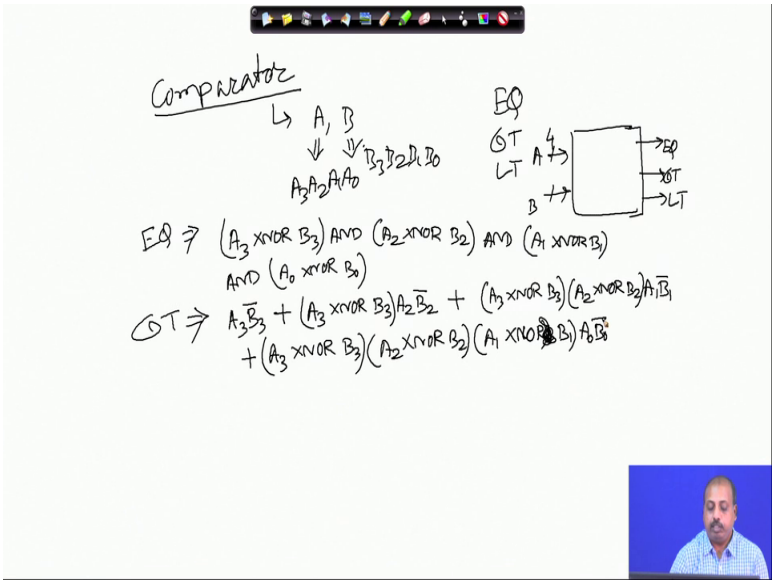
So, the simplest flavor type of flip flop that we have is that D flip flop, where there is a there is a D input or data input and there is a Q output or so normally there are two outputs Q and Q bar where Q bar is the complement of Q, now some was some of the cases you may not find a Q bar output explicitly but it is in many cases so Q and Q bar are there but in some cases some logic libraries may not have the Q bar output. Another very important input that we have here is the clock signal. So, this clock signal controls the operation of this flip flop ok, so you have got the clock signal so it is represented often like this so we call this the clock pulse or clock signal, so the operation is like this whenever this clock pulse comes then the value of this D will be copied on to Q and Q bar will get the complement of that and if this clock pulse is not there so it will remember the previous value, so in terms of truth table you can say that if it is D and it is Q if the D bit is 0 then this Q will also be 0 if d is 1 Q will be 1, but all of these are true provided the clock pulse value is equal to 1.

On the other hand if the clock pulse value is 0 whether it is 0 or 1 it does not matter. So, the clock pulse value is 0 so it will continue to have the previous value so you can say Q n minus 1 or Q n minus 1 that is a previous value of Q. So, it is not dependent on the value of D so if the clock pulse is absent then nothing happens to the circuit so it continues with the it continues with the previous value remembers the previous value ok. Now how to realize this circuit so for realizing it so you can do it in terms of combinational gates so we can do it like this say, I will just draw the circuit then I will try to explain, so we have got this d input here and we have got this clock pulse going to both of them, so D is connected there and D bar line is connected here and then there are 2 cross coupled NOR gates, to realize the Q and Q bar lines, so this is connected like this ok this is Q and this is Q bar.

Now, you see that if the clock pulse is 0 then what happens is that both the lines will get to both the NAND gate outputs will be 1 and as a result this Q bar coming here will get inverted and it will remain as Q, similarly the Q comes here getting inverted and remaining as q bar, so that way it continues as q and q bar so if the clock pulse is 0 nothing happens to the circuit if the clock pulse value is equal to 1 then at this point you get the value of D bar and here you get the value of D, because D bar comes here that is inverted so you get the value of D. Now the D bar coming here so if the D value is equal to say 1 then this D bar is 0 and then this value is say 0 and whatever is the Q bar, so in that case it will be if this value was previously 1 this value was 0. Now so this 0 and 0 so this is not say you get a 1 here and if it is 1, then you will get a 0 there so that way it is it is getting the value of D bar at the Q bar line ok.

So, like say this D bar line D is 1, D bar is 0 now this is the NOR gate. So, this is 0 and this is a Q bar so you will get a the Q bar line will be coming here a Q 1 line will come here getting get inverted and it will remain as Q and otherwise it will change to Q bar.

(Refer Slide Time: 23:35)



Now so we can have this combinational circuit so it can be used for realizing various module like, we can you realize something called the comparator you can realize something called a comparator, so in a comparator what happens is that the basic comparator module so if I want to compare between the numbers a and b where A and B so both are 4 bit number so this is say A 3, A 2, A 1, A 0 and this is B is B 3, B 2, B 1, B 0.

Now, if I say that the comparator has got a number of outputs equal greater than less than like that no so as far as the block is concerned so it is like this that it has got this A and B inputs which are each of them are 4 bit, and then it has got three outputs EQ GT and LT. Now you see that you can always draw the truth table so the if you try to draw the truth table then there will be 8 columns here and total 256 rows will come in the truth table and you can realize it you can find out the condition of equality greater than less than and accordingly you can draw the circuit.

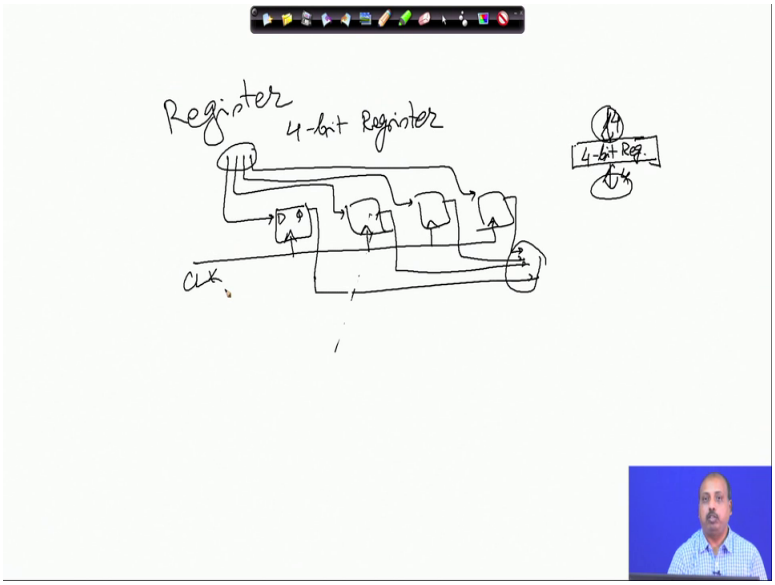
On the other hand you can also apply your intuition and try to come to the circuit, so the equation for say for the equality part, so equality will be true when all the bits are same and when all the bits are same so you have got this X NOR gate that will represent this operation, so if you say it is A 3 X NOR B 3 and A 2 X NOR B 2 and A 1 X NOR B 1 and A 0 X NOR B 0.

So, if all the bits are same then this x nor output will be 1 ok, so all the all these sub terms they will evaluate to 1 the EQ will become equal to 1, on the other hand if you are trying to get the value of expression for GT, so GT A is greater than B or the first condition is that if A 3 equal to 1 and B 3 equal to 0 then you have got a greater than B the first condition is A 3 B 3 bar A 3 is equal to 1 and B 3 equal to 0.

In the next condition that you have is A 3 and B 3 are same but A 2 is equal to 1 and B 2 equal to 0. So, the next condition that we have is A 3 X NOR B 3 so A 3 and B 3 are same and after that you have got A 2 B 2 bar so this is the next opera this is the next condition for this term or greater than term to be true or third condition is that A 3 B 3 A 3 and B 3 are same A 2 and B 2 are same but A 1 is 1 and B 1 is 0, so you can accordingly write this expression A 3 X NOR B 3, A 2 X NOR B 2 and then you have got A 1 B 1 bar.

And finally you have got all the bits same only the zero-th bit differ so A 3 X NOR B 3, A 2 X NOR B 2, A 1 X NOR B 1 sorry A 1 X NOR B 1 and then you have got A 0 B 0 bar, so these are the conditions for GT so this way you can write down the expression for less than as well and you can realize it using the circuit so the comparator circuit can be designed in this fashion.

(Refer Slide Time: 29:01)



So, you can also use this concept to realize sequential circuits like you can have. So, we have seen how to realize a single flip flop now in computer in processors there is one

module which is called a register which is very common in all these processors. So, register means it is a instead of having a single flip flop you have got a collection of flip flops. So, I can have a 4 bit register so if you want to get a 4 bit register so you need to have 4 flip flops, so 4 such d flip flops can be there so conceptually we will represent it in this fashion. So, this is my 4 bit register so this is the 4 bit register so it will have 4 input pins for input lines and can have four output lines.

But actually what happens is that if these are the 4 flip flops so the d the input actually the first input comes to the first flip flop second input goes to the second flip flop third input goes to the third flip flop and the fourth input goes to the fourth flip flop, similarly the as far as the output is concerned so this is the first output, this is the second output, this is the third output and this is the fourth output so that way they come so these 4 bits that are shown here so they are actually the four input lines here similarly the four output lines that are shown here the four output lines here.

So, what is common between all these, what is common between all these flip flops is that they are the clock line that they have so clock the clock line should be common. So, all of them have got the same clock line so they operate at the same clock they operate at the same clock. Fine so this type of register so they are called parallel load register so this 4 bit pattern can be loaded into these registers or you can you can read from this register so this register is another common component that we have in the processor design.