Digital Circuits and Systems

Prof. S. Srinivasan
Department of Electrical Engineering
Indian Institute of Technology, Madras

Lecture # 16

Introduction to Sequential Circuits

Until now we have been seeing circuits which are called combinational circuits. I think I mentioned this once to you earlier. A combinational circuit is defined as a circuit in which the output changes in the input that means the outputs are influenced only with the present inputs. On the other hand we have another class of circuit is called sequential circuits. That's what we will start today. As the name implies sequential circuits have some sequencing in it. What it means is the output of course depends on the input and in no circuit the output can not depend on the input there are some special circuits that, we will not talk about them now. But generally a circuit would like the output to be dependent on the input but not only of the input but also on the past behavior of the circuit.

As an example I give a count let us say count pulse count one that means you should count one from what it is right now. If the count now is seven this counts to be 8, if I give a count pulse again then from 8 you should go to 9, this is what I meant by sequential circuit. By the behavior of the circuit of counting depends on the input count pulse input pulse count has to come for this to count but it should count from where it left, this what it meant to the past behavior of the circuit or past outputs.

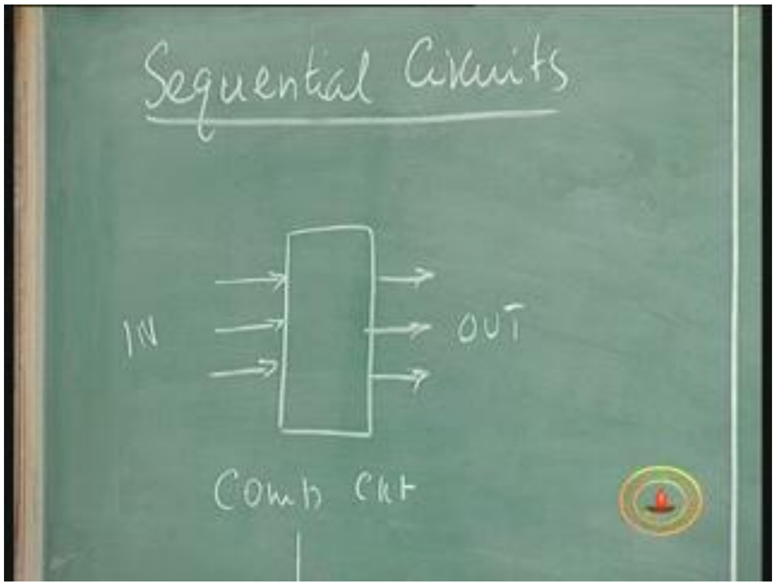
The present input as well as the previous output or previous outputs but you don't have to say outputs because previous output would have been this way one before that so it is a history that this is built-in. So the present input along with the past output or outputs determine the present output. Such a circuit is called a sequential circuit and it is very important to have such circuits. Why it is important to have these circuits is because we need to have a pattern of behavior. In a traffic light controller you are trying to design you have a timer which times the duration for which a particular light is on, you should know what light it should go to and that depends on the previous light, if I had a two lights red and green two roads let us call this road one and road two and in road one it has been red now and it will go to green based on whether it has been red for a while.

And next time after the next timing it should not change from green to red on the same road so the green for one road should be for certain duration and after that the next green should be for the second road only then the traffic from both roads can move. If you don't have the past output, as previously we changed road one from green to red and now the second road has changed then you don't want to go to the first road and then change it back to green the second road keeps red all the time so you do not want that, but you want some sort of a previous behavior, what was the previous pattern?

I already told you the example of the counter. I need to know the present count that can be given only by the past count value and then the present value whether the count has to be done or not. So it is important that any meaningful circuit, any design, any hardware, any system that you build will have to necessarily contain sequential circuits if you want to have a behavior pattern as per a sequence of events, elevator control. Even for a rocket launch a series of things a sequence of things have to happen, not abruptly something should happen. That means some sort of a past history of the output has to be available for the circuit to make a decision.

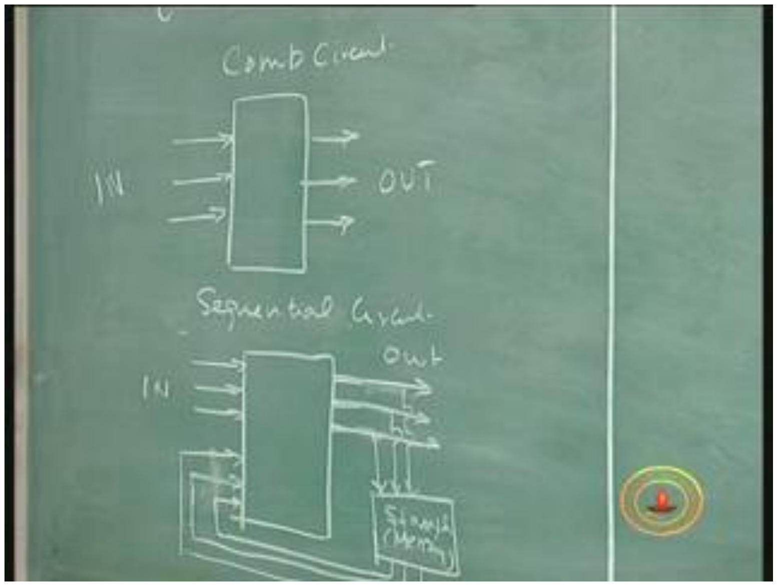
How can I say the circuit output has to depend on the present input and the past output, I have to have a way of remembering the past output. I should make available to the past output the circuit at the present time. At the present time I need to give the past output along with the present inputs. That means I need a storage element or a component which can store the value and give it at the required time, such an element is the memory you need to have a memory. The history has to be remembered. You can remember it only with a memory. So a sequential circuit will contain no doubt combinational circuit because once the history is given it will consider that as one of the input that's all. The combinational circuit takes the inputs and gives the output, based on the input it produces the output. Now the input is one more or many more depending on the previous output. So the previous outputs are also given as inputs. So if this was the combinational circuit model we give inputs and it gives outputs.

(Refer Slide Time 08:10)



The sequential circuit model will be inputs, outputs, these outputs will be stored there is some storage or memory we will call it memory or storage, what do you store in this? You store these previous outputs and make these available back onto this. So the model of the inputs deciding the output does not change, that is the model. For any circuit the input has to be given and the output will come and this circuit will still be a combinational circuit within this.

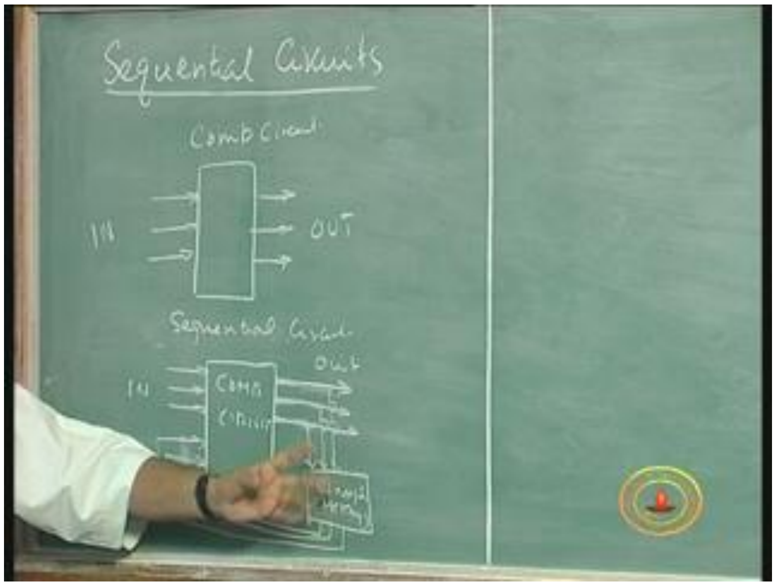
(Refer Slide Time: 9:46)



So this is combinational, this is sequential. But this block is a combinational block in the sense it combines. What is a combinational block? A combinational block combines the inputs to produce an output, combines in the form using gates or other functional units. So that is still that but I am going provide to it some more inputs which were not there in the previous case, these inputs being the past behavior of the circuit that's all that is to it.

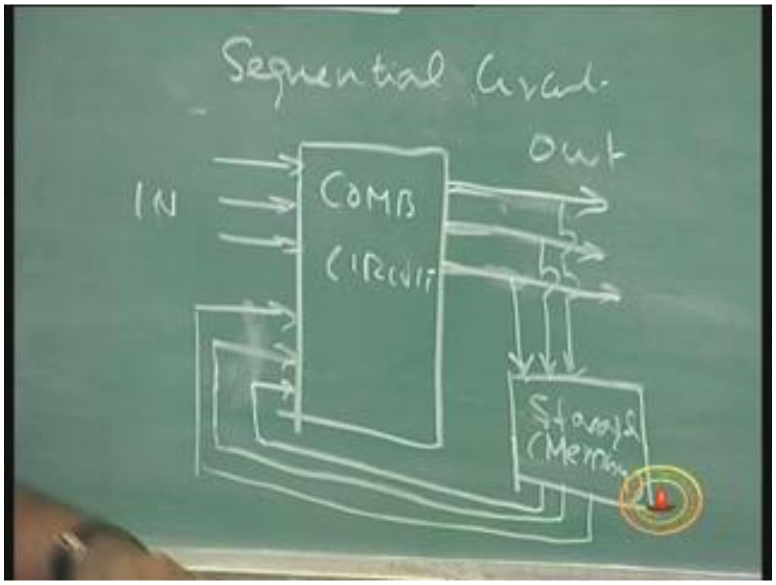
It is only an extreme model. It is not that every output has to be considered for every input. every past output has to be considered for considering the present output each time, and one of those outputs may determine whatever this is only a model a diagram so this is still a combinational circuit again which means there is nothing much here for us to do because we have already know how to design it given an input you do the truth table, Karnaugh Map, simplify, get the implementation using whatever type of gates you are asked to do it with and then we have the outputs. But in between you need to have these elements for storage.

(Refer Slide Time: 11:12)



This feedback component is what is new; the feedback concept is new so that the circuit will remember this behavior so that part is only the new thing here. Once we know how to do that or to implement this we can build this sequential circuit using this sort of a model and since we know how to do this as we have already done that part so this extra thing is all we need to do in order to realize a sequential circuit. And as I mentioned, in any practical circuit there will be lot of sequential thing, kind of a behavior, a computer for example. Supposing I need to give an instruction multiply, multiplication we did a little bit in our combinational circuit, we saw how to do a multiplication, is it purely combinational circuit using AND gates and adders?

(Refer Slide Time: 13: 46)



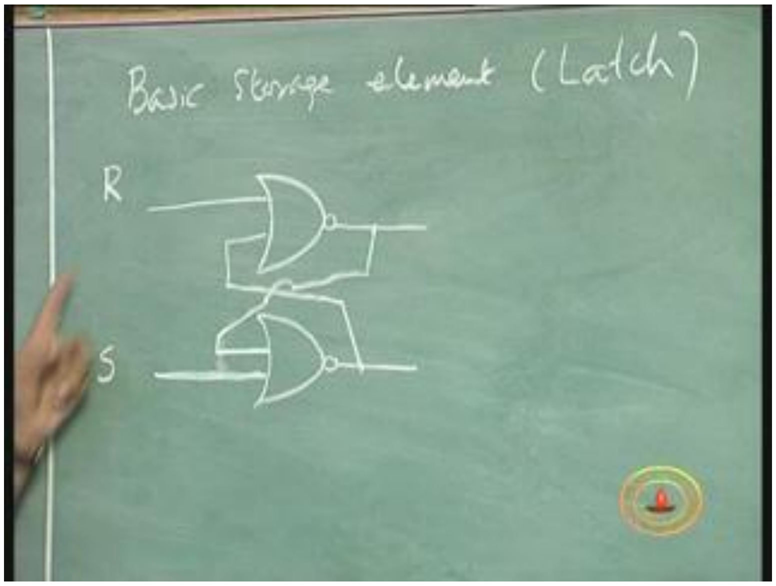
But I should tell the computer when to do that multiplication. Only when the correct values of As and Bs are available at the right inputs and at the right places and then I should say do the multiplication and then the result the sum or the product must be available at the right time to the next fixed unit. So there is a sequencing involved there also. Sequencing is an operation which is required in any system. A system cannot be done arbitrarily or it will become chaotic, I don't want a sciatic system where all circuits are right but they will function in their own way independent of each other. All of them will be working but they will not care about when the input is given and when the output is given. If that is not done then there is no logic there is no systematic thing and then the concept of system fails there.

So all we need to do is look at this small piece of hardware this block (Refer Slide Time: 13:01). This is the most common element to store, what do we store in digital circuits? there are only 0s and 1s all over, these are all binary everywhere it is binary it is a sea of bits, as if you break open a computer and see its brain on the side you will only see 0s and ones all over the place, sacks and sacks of 0s and 1s, that's all.

So I need to know how to storing 0 or 1. If I know basically know how to store 0 or 1 I can produce many such storage cells in order to store whatever values I want to store. It can be one bit it can two bit it can be multi bit four bit eight bit sixteen bit thirty two bits whatever number of bits I can do provided you know how to do one of them. If you remember my introduction to digital systems in this course I said the beauty of digital is you can replicate you can duplicate it without any effort.

That means all I need to do is find out how to store a single bit which is 0 or 1. If it is 0 I want it to remain 0 as long as 1 want it to remain 0 but if it is 1 I want to have it 1 and then have it as 1 as long as I want it. If you know how to build that basic building block then we know we have conquered the digital system because that's all is needed here, we have already know this so we can build any digital system. So the basic storage element is called a latch. As the name suggests it is the latch it latches on to the data, it latches on to 1 or to 0 and it is not a very complicated circuit this is a very simple circuit.

(Refer Slide Time: 15:48)



You know these gates, what are these gates? NOR gates, I will call this input for a reason which will be clear a little later on R and this input S. I am taking R as one of the inputs and here the other input to this NOR gate is the output of this NOR gate so the feedback is introduced here already. What did I say about this memory? Memory is feeding back putting the values back into the system the feedback concept this memory is nothing but a feedback concept, it is chewing on what happened earlier and that's what a memory does. So that's what is done here that's all. The feedback is built in so it should act as a latch to store a bit either 1 or 0. The second NOR gate has another input called S as I said we have specifically named it R and S for some reason, it will become evident pretty soon.

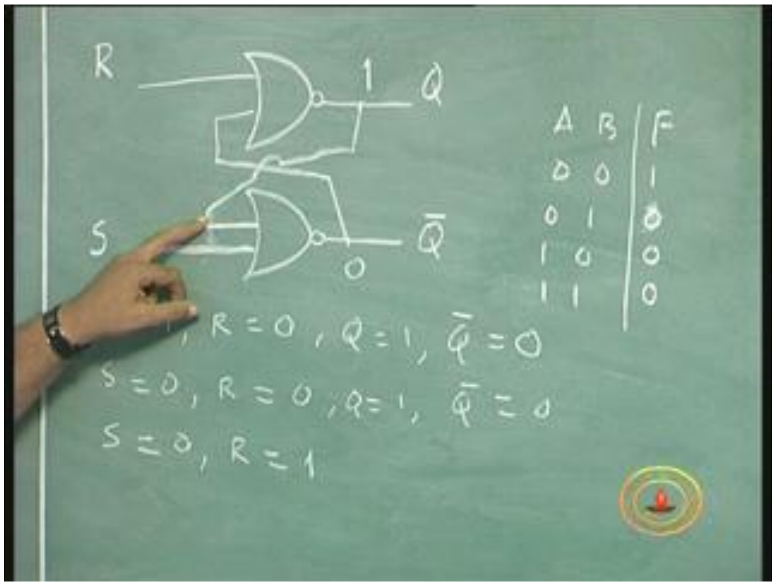
The other input of this NOR gate is from the other NOR gate. And then this is called Q and this as we will see is called Q bar (Refer Slide time: 16:58). That means this and this will be always complementing. I could have called it P and Q and then later on said P is same as this, whatever but it's all right. So let me put S is equal to 1 R is equal to 0. I do not know what the Q was because earlier these gates could have been having any outputs I didn't know that irrespective of that I am now going to introduce an input of S is equal to 1 and R is equal to 0. What is the truth table of NOR gate?

Let us write the truth table of NOR gate? A B F both are 0 and what is the output? It is 1. The complement of OR, NOR is the complement of OR. Now as long as one of the inputs is 0 since both inputs are not 0 as long as one of the input is 1 the output of that gate has to be 1. So when S is equal to 1 this gate has to have an output as 0. Whatever is the previous output the moment I make S is equal to 1 this output will become 0 and this gate will have input of 0 from here and since I made R is equal to 0 I am forcing this input to be 0 and this input is 0 so the output is 1, Q is one and Q bar is, already we are able to see that Q and Q bar are complementary, the right choice of the variables Q and Q bar. So we make S is equal to 1 and R is equal to 0 so Q becomes 1 and Q bar becomes 0.

Now I am going to make S is equal to 0 R is equal to 0. When S is equal to 0 R is equal to 0 that means this one I am making it 0, this R remains 0, (Refer Slide Time: 19:22) this one I am making 0 so what will happen now. This is 1 here and this is 0, 1 0 so output will be 0 here this will continue to be 0 and this 0 and this 0 will make this 1. That means this will continue to be 1 and this will continue to be 0.

Even though I made S is equal to 1 and R is equal to 0 earlier to make Q is equal to 1 and Q bar is equal to 0 when I remove S and made S is equal to 0 R is equal to 0 already we can see the effect of the previous output somehow. We will see that more clearly in the next stage. The previous output of Q is equal to 1 and Q bar is equal to 0 has affected it because I deduced this output as 0 because of this one I took this one into account (Refer Slide time: 20:12). Similarly I took this 0 into account determined this one. So Q continues to be 1 and Q bar continues to be 0. Now I am going to make a third combination I am going to make S is equal to 0 R is equal to 1.

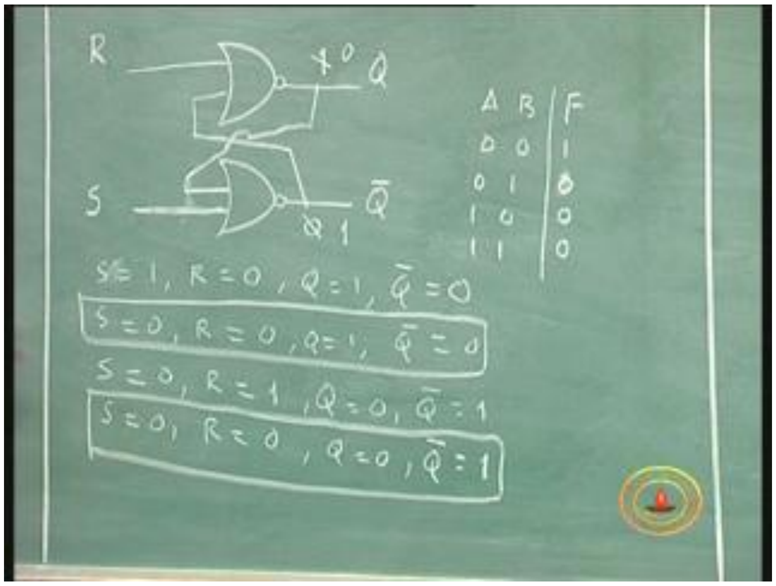
(Refer Slide Time: 21:02)


R
1
Q
S
0
$\bar{Q}$
A B F
0 0 1
0 1 0
1 0 0
1 1 0
, R = 0 , Q = 1, $\bar{Q}$ = 0
S = 0, R = 0 , Q = 1, $\bar{Q}$ = 0
S = 0, R = 1

Now it can be 0 1 or whatever, S is equal to 0 this is 1 so you put 0 1, the S is equal to 0, R is equal to 1 0 this has become 0 even if you assume the other way it will change again. So we can make R is equal to 1 and this is 0, 1 0 is going to make it 0 Q is going to become 0 and since S is 0 this is 0 this is going to become 1. So Q is going to become 0 Q bar is going to become 1. If we had taken this first I could have taken this 1 as this. this is 1 0 and this 1 0 would have resulted in 0 and 0 1 would have resulted in 0 so even though I have assumed it to be 1 it would have become 0 and then I have to do one more iteration and till it settles down I have to consider it.

Now again I will do this S is equal to 0 R is equal to 0. This will make it very clear that the present output is dependent on the previous output. Now I am considering in this case when S is equal to 0 R is equal to 1, Q is 0 Q bar is 1 after that I am one more time making S is equal to 0 R is equal to 0 so when you do that what happens? This was 0 now and this was 1 (Refer Slide Time: 22:39) so 0 0 continues to be 1 and 1 0 continues to be 0. Take these two cases (Refer Slide Time: 23:04). S is equal to 0 R is equal to 0, output is 0 1. In this case S is equal to 0 R is equal to 0 output is 1 0 because the previous to this input was 1 0 which results in output of 1 0 and in this case the previous output of 0 1 is resulting in output of 0 1.

(Refer Slide Time: 23:31)



A	B	F
0	0	1
0	1	0
1	0	0
1	1	0

$S = 1,\ R = 0,\ Q = 1,\ \bar{Q} = 0$

$\boxed{S = 0,\ R = 0,\ Q = 1,\ \bar{Q} = 0}$

$S = 0,\ R = 1,\ Q = 0,\ \bar{Q} = 1$

$\boxed{S = 0,\ R = 0,\ Q = 0,\ \bar{Q} = 1}$

So this condition of when S is equal to 0 R is equal to 0 corresponds to the condition where the output will remain unchanged from what it was earlier. This is the memory. When you want to save something when you store something when you give that information and remove this stimulus remove the excitation still it is retained. If I tell you something and go away you continue to remember it. The definition of memory is that when some information is given is retained until it is changed further some other way, erased and written over. I can change it and say no what I said yesterday was wrong and this is the new information then you will try to hold on to that new information.

So the information versus put was 1 0 and after this we left it removed that excitation which made it 1 0 made it 0 0 both the inputs were 0 the output remained 1 it retained the memory, the memory retained the information 1 0.

Likewise we put information 0 1 and made the output 0 1 after that if we removed this excitation of R is equal to 1 made it 0 then the output remained the same as previously 0 1. So this is memory stage (Refer Slide Time: 25:19). So what did we start with? Our requirement was to store a bit, that’s all. I said memory is required in our sequential circuit implementation because bits are always what is required to be stored in digital systems 0s and 1s. So if have a mechanism of storing 1 and keeping it there as long as I want it or if I have a mechanism of storing a 0 and keeping it there as long as I want it I have succeeded in building a memory in a binary system and I can replicate it any number of times to get whatever the size of memory I want. This is what I said earlier. Therefore have you now achieved that? Yes.

If I want to store 1 the latch is the memory element. it has two inputs R and S two outputs Q and Q bar so if I want to put 1 there I want to store 1 bit or bit 1 the value of a bit is 1 and I want to store it, I put 1 and S and 0 and R it will become 1 and 0 and even though if 1 is not available to me any more it will continue to retain it until of course I change it. Of course if I change it you have to retain what I wrote earlier even after changing my inputs I am not saying that, this is what I said a while ago about our memory. If I give you information if you remember it as long as I don't come and say don't have that information but have this information then naturally we will have to remove that information and put the new information memory.

Of course we being humans you will also tell you showed me that and you will immediately argue you told me that yesterday and you are an unreliable person and all that but that is not what I mean here, these are all electronics. They are better than us so they do not talk back fortunately. Similarly, when you want to store a bit of 0 all you need to do is to put 0 on this and 1 on this and that will register 0 in this latch and the 0 will remain as 0 even after removing the excitation 0 1 may be both 0 0. So this is the basic component of a memory component which is a fundamental building block of a sequential circuit in addition to all the combinational building blocks because combinational block cannot be dispensed with just because we are doing sequential. All circuits will still have lot of combinational blocks but then sequential thing is one which determines the operation the event driven, the sequence of operations, the output now depends on what output was there earlier so all that will be determined by the sequential circuits because there is memory unit so that is the basic stuff.

We have studied three cases we have two circuits actually it is combinational. You call it sequential circuit but as far I am concerned they are two NOR gates and I have not done anything different from what we know already. Did I introduce any concept here a new theory or anything? No. There are only two gates for which you know the behavior of and we used them in a way we wanted it. So, combinational logic is the basic building block NOR gate which is the universal gate by the way is used also for memory. So in every circuit including combinational and sequential all circuits can be formulated or designed or implemented using AND OR inverter as the general case or in a specific case of universal gates NAND or NOR.

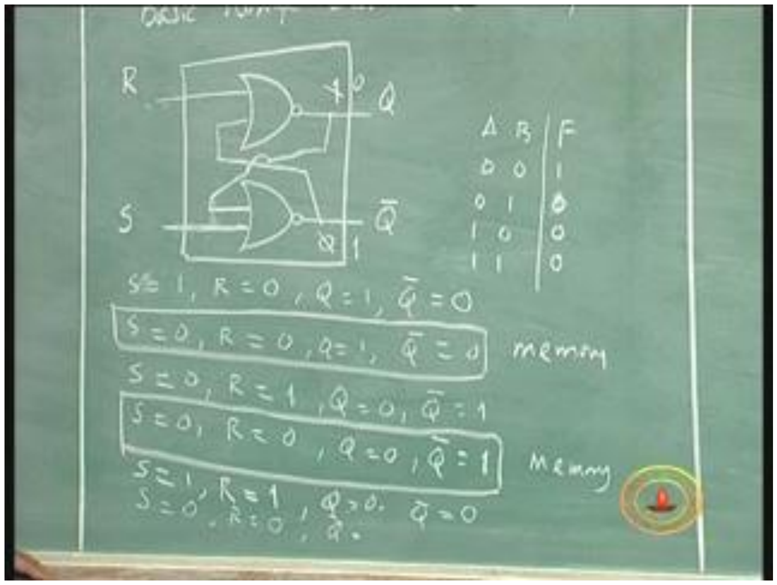
Combinational circuit basically even though there is a feedback inside it outside for me there are two inputs and two outputs, and if I give two inputs I get an output and I have a different combination of inputs I have different outputs so I have tried 0 0 combination, 1 0 combination I have tried but I have not tried the 1 1 combination.

What will be the output now when S is equal to 1 and R is equal to 1 both of the inputs are 1? The output has to be 0 0 because in a NOR gate as long as one of the inputs is 1 the output has to be 0 so in this case Q is equal to 0 so first of all it defines your idea the Q and Q bar I purposely put Q and Q bar as one which complement another that is violated here and when I say this and this also is 0 (Refer Slide Time: 30:24) when I say this is 0 and this is also 0 it is not Q and Q bar any more.

Now that is not a big problem though. After this I put 0 0 what happens? Because our interest is a memory component here, putting a data is important but keeping it there is also equally important that's what we said. Putting a bit is not the thing in a memory I can always give you some information. If you forget it the moment I give you the next instant of time what is the point in telling you something. So it is not the information that I give that is more important the information the storage capacity the capacity to retain it is more important. So let us see whether it has there. So you will put 0 0 here what will happen now? now both are 0 I put 0 here so let us assume this as 0 and this as 0 so 0 0 and the output will be 1 and 1 0 this will be 1 0 0 so this will be 0 this will be 1.

When I say S is equal to 1 R is equal to 1 Q become 0 Q bar is equal to 0 and then after that I put S is equal to 0 R is equal to 0 as I tried in these two cases but Q if I analyze this way that since this is 0, this is 0 this will produce 1 (Refer Slide Time: 31:50) and this one will go here and make 1 0 0. So this will be Q bar is equal to 1 Q is equal to 0.

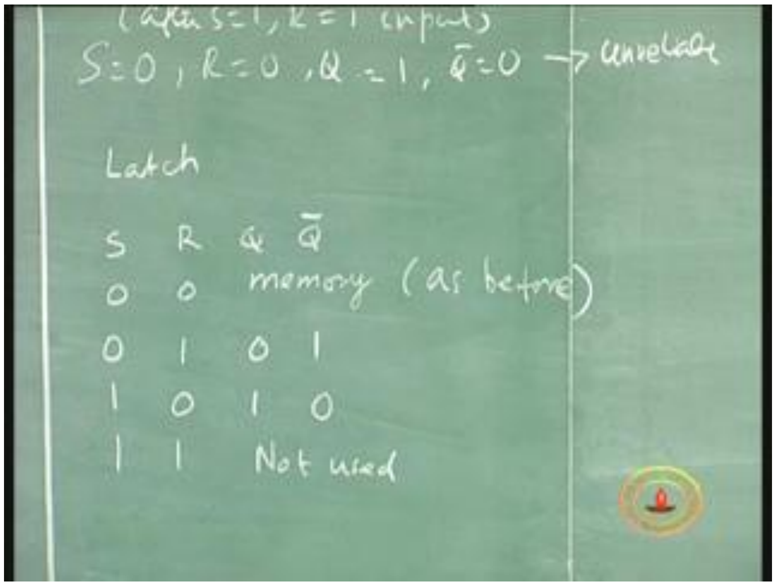
(Refer Slide Time: 31:59)


R
Q
S
$\bar{Q}$
A B F
0 0 1
0 1 0
1 0 0
1 1 0
S=1, R=0, Q=1, $\bar{Q}$=0
S=0, R=0, Q=1, $\bar{Q}$=0
memory
S=0, R=1, Q=0, $\bar{Q}$=1
S=0, R=0, Q=0, $\bar{Q}$=1
Memory
S=1, R=1, Q=0, $\bar{Q}$=0
S=0, R=0, Q

On the other hand if I started with this gate this is 0 0 earlier this is also 0 0 so this is 0, this is 0, this is 1, 0 0 will result in 1 here that 1 goes here and there is 1 0 and this is a 0. So if I started analyzing this gate first I get 1 here and 0 here. If I started my analysis with this gate first I get 1 here and 0 here. So output depends on whichever gate I start takes first to analyze. Would you like to have a circuit in which the output will depend on which gate you look at first or something like that? It is an unreliable operation. So this is Q bar is equal to 1 and Q is equal to 0 is not for sure no guarantee on that because if I have taken this way I would have got 0 1. So if I can continue this I can also write for S is equal to 0 R is equal to 0 after of course. After S is equal to 0 R is equal to 1 I make S is equal to 0 R is equal to 0. One possibility is output is 1 0 I mean Q bar is 1 Q is 0 and the other possibility is Q is 0 Q bar is 1 and here Q is 1 Q bar is 0 this is also possible. I can either get this or this. Either of the two combinations will be valid of course is not a question of my taking it but this is a question of which of the two gates acts faster.

I take this way or that way because I am only looking at as an observer. But the electronics inside one of them will act faster than the other. We cannot have two gates which are identical in terms of delay. Even if you nominally look at your data sheet and to the Pico second accuracy you match them and put them there will be a small minor variation of delay times of these two gates and one of them will be faster than the other. So whichever gate is faster that gate gets the output as 1 and the other gate gets the output as 0. And since we don't know which gate is going to be faster than the other the output of the latch with S is equal to 0 R is equal to 0 after the condition S is equal to 1 R is equal to 1 becomes unreliable. So this and this are unreliable outputs.

(Refer Slide Time 37:51)



(after S=1, R=1 inputs)

$S=0,\ R=0,\ Q=1,\ \bar{Q}=0 \rightarrow$ unreliable

Latch

S	R	Q	$\bar{Q}$
0	0	memory (as before)	
0	1	0	1
1	0	1	0
1	1	Not used	

When something is unreliable I will not use it as a memory at all so I will not use this combination so this combination (Refer Slide Time: 35:23) of S is equal to 1 R is equal to 1 there is no use, I have no use for this.

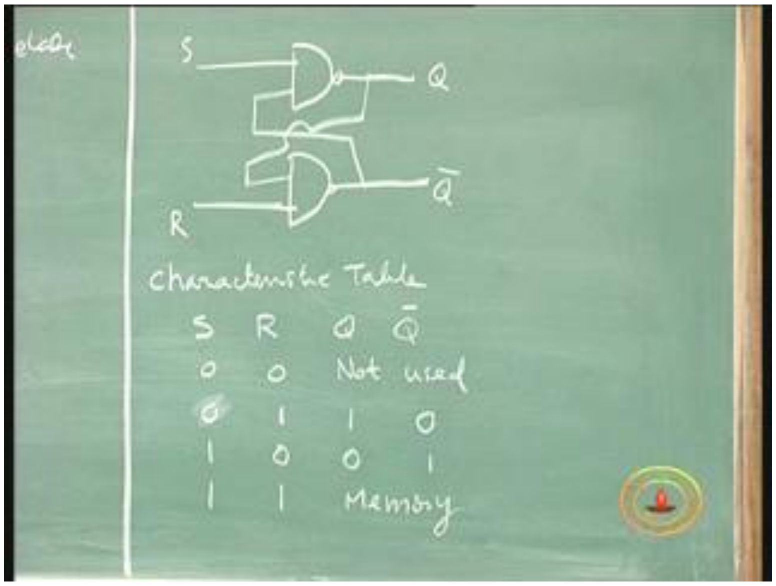
Since I have no use I will not use it at all or I can say not use it. Since I don't have a use for it I will not use it. So that combination is not used at all, never used. you don't put S is equal to 1 R is equal to 1 at all not because it is going to blow up or anything not going to blow on your face, nothing is going to happen no explosion don't worry, in the lab it will not happen, put 1 1 it will give 0 0. The moment you remove that 1 1 what is Q and what is Q bar is unpredictable. So you don't know what is going to be Q and what is going to be Q bar. Because if there is no point in having something which is not sure why would I put into my circuit something I am not sure about to start with so I will not use that combination.

That means I can write the table now because I put this analysis I write the table of the latch S R Q bar Q where S is equal to 0 R is equal to 0 both are 0 0 it acts as a memory. When I give S is equal to 0 R is equal to 0 I will not be able to say what is the output Q and what is the output Q bar. It will depend on what the output was before. That is why I call it memory in my table. Some people write it as" as before state" it will not change. In some books you will see it is written as "as before state" or "no change state" or "memory state" these are the different ways synonyms they use for this.

If it is 0 and 1 the output is 0 and 1 and if it is 1 and 0 then again output is 1 and 0, if it is 1 1 then it is not used. This table is not called a truth table. This is a truth table only it is truth it is not lie but truth table is generally defined for combinational logic where inputs are specified and a corresponding set of outputs are specified. Now we have dependency, here there is dependence like memory depends on the previous output so such a table is called a characteristic table but loosely lot of people call it truth table of latch no problem. If you call it truth table of the latch I have no problem but if you want to be very correct about it you call it a characteristic table. Characteristic because it is characteristic of, and I have to give a name for this latch, a latch is to hold on, the latch came from the fact that it holds on to the data 1 or 0 latches on to it. And since it uses two inputs S and R it is called characteristic of S R latch. And why it is called S R latch I have to tell you now. S is the condition where when you put S is equal to 1 the output remains gets 1. When you put R is equal to 1 output is 0. Something becoming 1 is called set condition in digital parlance, digital vocabulary or parlance 1 is called setting and 0 is called resetting. You make something 1 you set it to 1 and when you make something to 0 then you say reset it to 0.

Set is 1 reset is 0 so we can store 1 with an input to S is equal to 1 R is equal to 0 of course and reset it with an input of S is equal to 0 and R is equal to 1, that's why is called SR latch. I can also draw an AND gate NAND gate version of this because whatever you do with NOR gate you can do with NAND gate. So a NAND gate version of this is, this becomes S and this becomes R where the inputs are reversed and the characteristic table is S R Q Q bar.

(Refer Slide Time 42:42)



Characteristic Table

S	R	Q	$\bar{Q}$
0	0	Not	used
0	1	1	0
1	0	0	1
1	1	Memory	

Here 0 0 is not permitted state. Some people also write not used instead of not used it is not permitted. It is permitted of course provided you don't care for the output. This is not permitted in the sense you will not get reliable results. So not used is 0 0 it's a complement operation. A 0 and 1 will give you 1 and 0 and 1 0 will give you 0 1 and 1 1 will be memory. The operation is exactly complementary to the NOR operation. If you want to put 1 if you want to set the flip flop you need to put 0 and S and R then you can set the flip flop Q will become 1.

I am going to reset the flip flop make it 0 output you have to put S is equal to 1 and R is equal to 0 and then when you want to retain it in the memory state you make both inputs 1 and 0 0 is not permitted for the same reason and after that the next operation is unreliable. This SR latch also, all sorts of nomenclature you will find in the books. We can say SR latch with NAND gates is the simplest way of understanding this. Instead of SR latch NOR gates and since here S is 1 it sets flip flop is set to 1 and when S is 0 flip flop is reset to 0 so S corresponds to the setting condition. Here when S is equal to 0 the flip flop is really reset, S is equal to 1 the flip flop is reset, and when S is equal to 0 it is set and because it is opposite some people call this S bar R bar latch.

As long as you understand the functioning of the table functioning of this circuit it is good enough know that it is a NAND operation, using NAND gates this is NOR gates. So I call it SR latch with NAND gates SR latch with NOR gates. but if some books may call it S bar R bar latch doesn't matter, if you want to use that nomenclature it is fine with me because of the fact that S does not set it S bar sets it and R doesn't reset it but R bar resets it. So we now have a basic mechanism of storing a bit of information a bit is 1 or 0 and entire data whether it is a simple code converter or 7-segment display or a huge computer with lots of data be processed all data are in the form of 0s and 1s. So I have a basic mechanism of storing 0 or 1 so it is a question of replicating mass production of these latches in order to store whatever information I want and that information is needed to be placed as additional input to the circuit.

So when I want to design a digital circuit I have a set of inputs and I have a specific set of outputs requirement, I look at the input combination and look at the specifications and try to get the outputs sometimes output cannot be gotten like that so I need to know what happened earlier so I need to have this information ready and feed it back as an additionally input to the circuit. So our combinational logic theory whatever we have seen is not useless it is required, it is not that combinational logic can be forgotten from today and sequential logic is what is. Combinational logic along with sequential logic make digital systems stick, real systems will have both combinational and sequential. Sequence decides the sequence of operations. Combinational logic does the actual output requirement based on the inputs so both are required and the basis is done.

But of course you have a few glitches to sort out. One of them is this 1 0 0 1 being not permitted is something you would not like to do but sometimes occasionally there might be a circuit configuration which may make 0 0 come into the input of this so how would you prevent it? It is very difficult, in the case of wet point you say don’t touch it and you want to go and touch it, when you say don’t do it you do it, have you seen that? Whenever somebody says wet point you want to see how wet it is and sometimes it gets stuck to your hand and then you keep rubbing it. Therefore when you say something is not permitted you would like to try to do it and see what happens.

Of course not purposely sometimes the circuit behavior the previous stage which feeds on to this stage make read a glitch of 1 1 or 0 0 and you don’t know something may happen. Something may happen don’t be too nervous about it. It is only a condition of unreliability which I don’t want, you want certainty but you don’t want unreliability, uncertainty. So we have to remove that we will have to see how to remove that and if you want to really store 1 or 0 do you really want to store, to store 1 I will put 1 on S and R is equal to 0, to store 0 I have to put 0 and R is equal to 1. Why should I give two inputs to store one output so I have to remove that so I need to work lot more on this? How I am going to remove 1 1 or 0 0 case which is not a permitted case, how am I going to have one input, if I want to store 1 I should able to give 1 and store it and if I want to store 0 I should be able to give 0 and store it, don’t ask me if I want to store 1 and give 1 and 0 and then to store 1, if you want to store 0 you give 0 and 1 and then I will store 0, that’s all. So we will have to look at some of those extra things that we need to do. With that this is a very basic and fundamental building block like a gate, two input gate to be started with similar hierarchy here in this sequential or simplest R latch, we will modify it to make it a more useful circuit and then look at some other applications of these circuits.