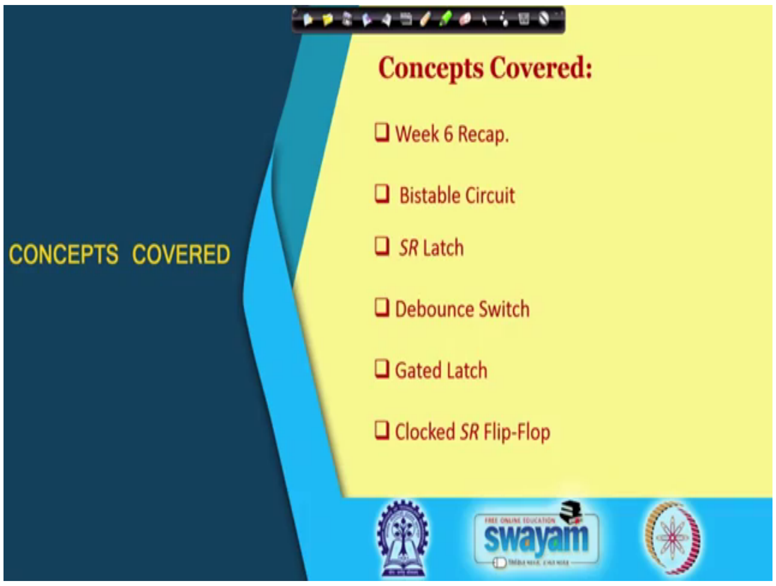
Digital Electronic Circuits

Prof. Goutam Saha
Department of E & EC Engineering
Indian Institute of Technology, Kharagpur

Lecture – 31
SR Latch and Introduction to Clocked Flip-Flop

Hello everybody. We are now in week 7 of this particular course. We shall begin Sequential Logic Circuit from today's class.

(Refer Slide Time: 00:25)



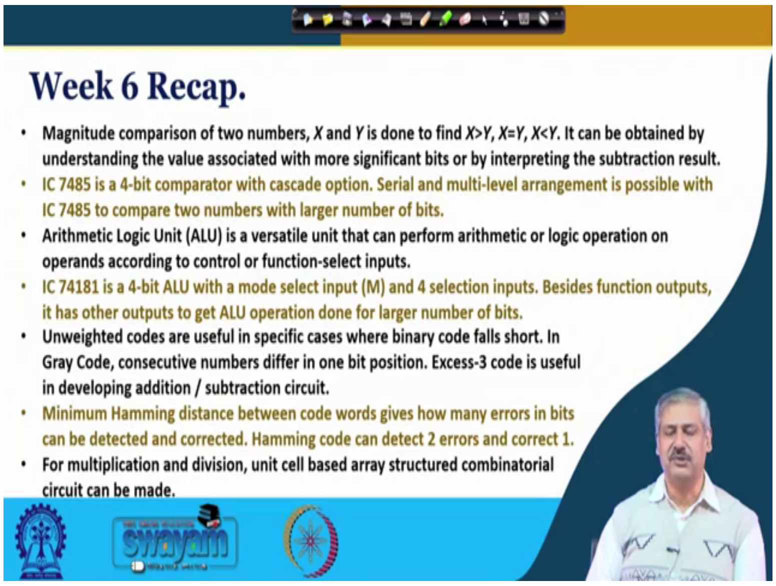
CONCEPTS COVERED

Concepts Covered:

- ❑ Week 6 Recap.
- ❑ Bistable Circuit
- ❑ *SR* Latch
- ❑ Debounce Switch
- ❑ Gated Latch
- ❑ Clocked *SR* Flip-Flop

And we shall have a quick recap of week 6, I believe it was little bit heavy. Because that was the last week of the combinatorial circuit discussion, so lot of advanced things were there. So, I hope you could assimilate them. But as we are done in case of combinatorial circuit, we made a soft start with introduction of basic gates and all. Then we moved on to build more complex circuit. So, similarly we shall start softly with basic components of these sequential logic circuit the by which we shall make more complex circuit in the a later classes, ok.

(Refer Slide Time: 01:10)



Week 6 Recap.

- Magnitude comparison of two numbers, *X* and *Y* is done to find *X*>*Y*, *X*=*Y*, *X*<*Y*. It can be obtained by understanding the value associated with more significant bits or by interpreting the subtraction result.
- IC 7485 is a 4-bit comparator with cascade option. Serial and multi-level arrangement is possible with IC 7485 to compare two numbers with larger number of bits.
- Arithmetic Logic Unit (ALU) is a versatile unit that can perform arithmetic or logic operation on operands according to control or function-select inputs.
- IC 74181 is a 4-bit ALU with a mode select input (M) and 4 selection inputs. Besides function outputs, it has other outputs to get ALU operation done for larger number of bits.
- Unweighted codes are useful in specific cases where binary code falls short. In Gray Code, consecutive numbers differ in one bit position. Excess-3 code is useful in developing addition / subtraction circuit.
- Minimum Hamming distance between code words gives how many errors in bits can be detected and corrected. Hamming code can detect 2 errors and correct 1.
- For multiplication and division, unit cell based array structured combinatorial circuit can be made.

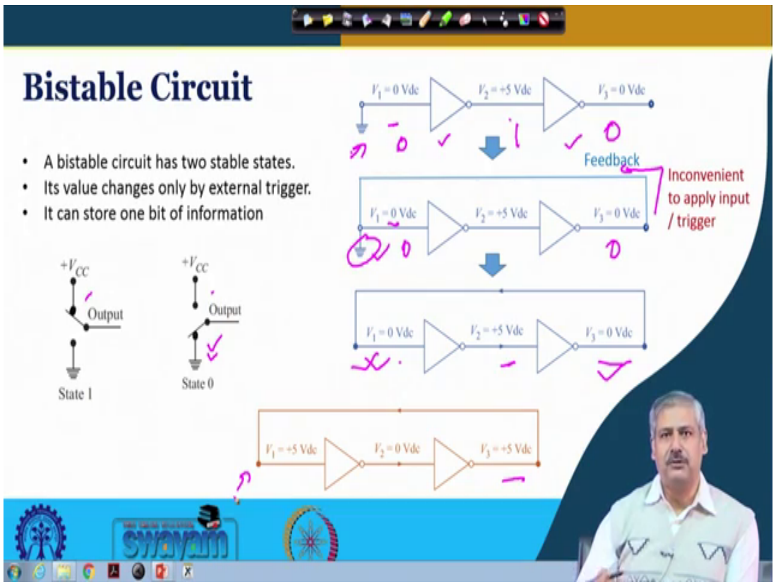
So, we discussed a magnitude comparison of two numbers, and how outputs like X greater than Y X is equal to Y X less than Y was generated while generated. And in that case we had to kind of an approach: one is of course by subtraction and another is by comparing the more significant bits of two numbers successively, ok. And then we discussed arithmetic logic unit, and we saw that one device which is versatile in nature depending on the selection unit it does perform one of the many different arithmetic or logicyou know function. And we saw that how all these devices can be cascaded and more number of bits can be handled together.

Then we discussed unweighted codes, ok. And then we saw that how gray code disadvantages where 2 consecutive number change by only 1-bit position. And we also discuss the excess 3 code, where we saw that the BCD arithmetic the way you had conducted before that can be made simpler by excess 3. And then we looked at error correction and error detection codes, and in that context we defined hamming distance. And we saw that minimum hamming distance in a particular coding paradigm how that can be useful in deciding or in understanding how many bits can be corrected or how many bits can be detected, ok.

So, in that context we discussed a parity code standard parity code even parity or parity code which we discussed in one of the earlier weeks. And also we discussed a new code hamming code which can detect 2 error and correct 1 error correct 1 error in bit position, ok.

So, this is was in brief what we discussed in the; and of course yes we discussed multiplication and division, right. And we saw bit you know seemingly complex circuit, but we had examples by which we could appreciate how the processing was done through the array the structure, in the unit cells, and finally the results were obtained, ok. So, each of the cases we have defined the unit cell properly and looked at the solution by taking specific examples.

(Refer Slide Time: 03:46)



Bistable Circuit

- A bistable circuit has two stable states.
- Its value changes only by external trigger.
- It can store one bit of information

So, with this we start this week’s discussion as I said, we shall start with defining basic units of sequential logic circuit which is memory element also called flip-flop. But before that we shall see what is a bistable circuit.

So, bistable circuit is something where the circuit is stable in one of the two position; bi for bi stands for 2, right. So, both the positions are stable, right. So, if you look at these you know a switch where you are applying a specific force by hand and all, and then you are connecting the switch to this particular place. Then you remove that hand pressure or you know this whatever force external force the switch will remain in that position.

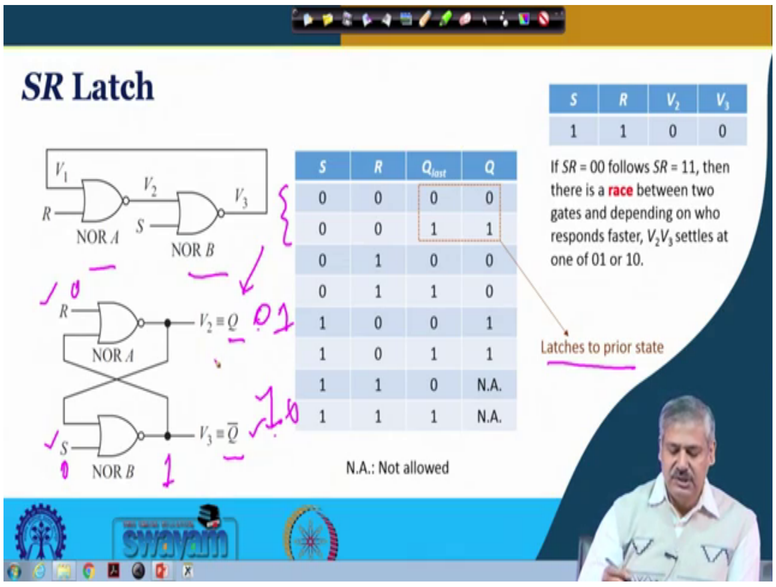
But you will again put a force and bring it here again it will remain there if you do not force it to go to another position. So, each of these state is stable, ok. So, in bistable circuit we can store 1-bit of information. That that is whether this output is high V CC or output is 0, right. So, together you can store in the context of binary representation of information 1-bit information we can store in the bistable circuit.

Now moving over to digital circuits. So, let us see if we have 2 inverters connected one after another right, and we place a 0 volt here what will happen to it to the inverter outputs in these cases. The first will be this 0 volt to the 5 volt, right. So, logic 0 will become logic over here and again it gets inverted it is logic 0, ok. This is understood. Now while this ground is here you connect a feedback. So, this feedback is keep; since feedback is very important in developing the basic cell and the sequential logic circuit, right. In the combinatorial circuit we did not discuss, we did not include feedback. So, the feedback comes in the sequential logic circuit discussion.

So, you connect the feedback the way you have seen from output to the input; why, there is no issue no ambiguity because this is 0 volt and this is also 0 volt. So, this is logic 0, this is also logic 0, this is logic 1. Now if you remove this ground like this circuit becomes like this. What will happen to the circuit? It will remain at this 0 volt, this 0 volt here is driving this one, and this 0 volt is driving this 5 volt 5 volt is 0 volt. So, it will remain in that place like you know as a stable formation right, this output will be 0 volt. Now instead of this ground if you had started with a 5 volt here, then a feedback, ok. And then you remove that you know 5 volt; how to do have been the output in this case, output would have the stable 5 volt, ok.

So, this is what we can see a bistable circuit in the context of digital logic representation, right. But what we find here that this way of you know applying external you know this ground of 5 volt this is a bit inconvenience. So, you would look for something which is a more convenient to trigger from one particular state to another.

(Refer Slide Time: 07:29)



SR Latch

S	R	V2	V3
1	1	0	0

S	R	Q_{last}	Q
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	N.A.
1	1	1	N.A.

If $SR = 00$ follows $SR = 11$, then there is a **race** between two gates and depending on who responds faster, V_2V_3 settles at one of 01 or 10.

Latches to prior state

N.A.: Not allowed

So towards that, let us look at this circuit where we are having instead of NOT gate we are having 2 NOR gate, ok. So, you can see the feedback happening here. And we can write, we can draw the same diagram in a different manner which is you know bit cross coupled, ok. But there is one feedback you remember and one is in the forward path, but this is the way you can draw it, ok. And we give this input some name S and R and more over that it will be clear very soon, ok. And this is Q and we shell see that it happens to be the cases where we are operating it the way we operate it this V 3 will be inverse of V 2. So, this is 0 volt it will be 5 volt and vice versa.

So, that way we can write it to be Q bar, ok. But we shall with the you know see more of it in the subsequent discussion, right. So, considered that both of them are 0 0 at a given time, ok. So, for a NOR gate 0 is a non n non-forcing input ok, for OR and NOR 1 is the forcing input, for OR gate if 1 is there output will be 1 irrespective of what is there in the other input. And for NOR gate if 1 is there output will be 0 irrespective of what is there in the other input; isn't it. So, these are forcing input, right. So, 0 0 is non-forcing input. So, it will look at what are the other inputs by which the output will be decided.

So, if to start with you consider that this was 0, right. So, then this 0 0 this is 1 right and this one, right is making it 0. So, this 0 and 1 if the previous value was 0, right then it will be 1. And from the symmetry you can say if this was 1 this would this would have been 0. So, if you look at this truth table this is what you can see; that 0 0 then 0 0 and 1

1. So, whatever had made the previous value 0 or 1 that will be continued with when this toward 0. So, the prior value of Q it is last into, ok. This prior value of Q is lasting, is it ok. So, this is what we see with 0 0 is present.

Now let us see if you present a 0 here and a 1 here, ok. So, 1 is a forcing input irrespective of what were was the pass value or so, because this is an coming from in an external. So, this is this will make it 0 right, and this 0 0 we will make it 1 even if the previous value was 0 1 1 or whatever. So, ultimately this 1 forces it to be 0. So, when this is 0 1 right, S is 0 and R is 1 irrespectively of what is the previous value the output is always 0, right. And similarly again from symmetry this is 1 and this is 0 this is the case, right. This 1 will force it to be 0, this 0 0 this is 1. So, output will be 1, right Q will be 1. So, this is what you can see, is it all right.

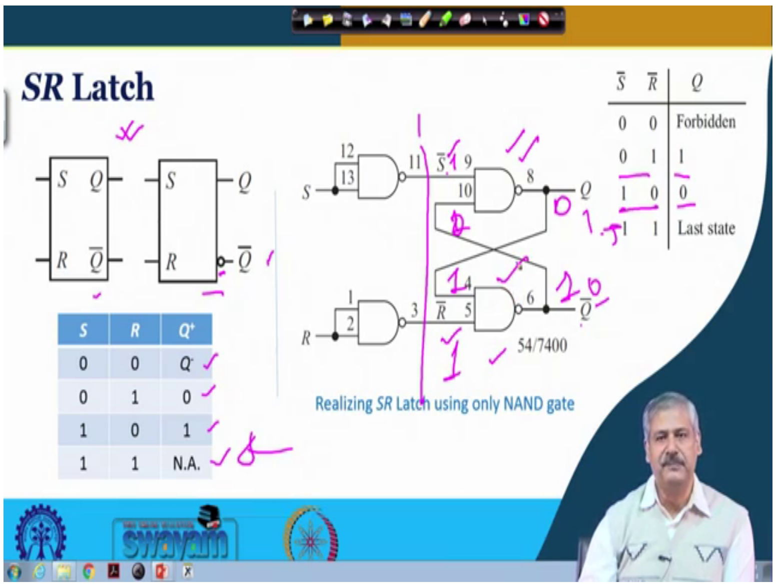
So if it was set, this now stands for S stands for set and R stands for reset. So, when S is 1 and R is 0 we say this particular latch this is basically it is called latch because it is latching whenever we are putting a 0 0 value. So, whatever the past value which made the Q 0 or 1 because of 0 1 or 1 1 1 0 that was present it latches on to the past stage; past you know value past state. So, that is why it is called latch. So, this S stands for set and a R stands reset. So, 1 0 is a setting of the latch and a 1 0 of S and R. And 0 1 of S and R is resetting of the latch, ok. Is it fine?

Now what would happen if 1 1 was presented here? So, the output would have been 0 0 which is fine, I mean as such that is nothing no ambiguity about it. But after 1 1 if you try to put into that resting state when the past value would be in latched on to. So, after that if you put 0 0 then what happens actually that this 0 and this is external 0, so both are now one input C, right. And this 0 is over here. Now depends I mean which one of them gets the 1 before the other. Ideally one can say that both of them will get 1 you know together. So, that 1 will go to the feedback to the respective cases. So, both 1 1 will become 0, ok.

So, that is the ideal case, but actually where the propagation delays are identical, but whatever be the fabrication process or so we will see that one of them which is not for sure because of the way it gets fabricated different transistors and other things are there in the circuit element passive element. So, one of them will be having a higher you know propagation delay than the other. So, that will correspond the slower which we cannot say as a designer what which one is you know have going to acquire the 1 value before the other, right.

So, that is something which becomes a bit you know unpredictable. And arrays between these 2 who means and that is something which we would like to avoid in a sorry; latch um use or application, ok. So, that is why this 1 1, we are saying as not allowed, clear.

(Refer Slide Time: 13:50)



SR Latch

S	R	Q+
0	0	Q-
0	1	0
1	0	1
1	1	N.A.

$\bar{S}$	$\bar{R}$	Q
0	0	Forbidden
0	1	1
1	0	0
1	1	Last state

Realizing *SR* Latch using only NAND gate

So, how do we again we will look into the you know the representation part. So, we will not write all those you know basic gates that we have used, instead we can prefer to put it in the form of a symbol which will help us in making bigger circuits, ok. Otherwise this circuit will become a bit you know clumsy too many you know logic gates and all.

So, when we say SR latch, so this is the way we can put it. And if you look at the textbook 2 different symbols have been used. In one case Q and Q bar are output inside in both the cases we will see SR is there, ok. And that means, this is Q and this is the invert of it that is what we have seen in the cases; the allowable cases, right. and In another representation Q bar Q and Q bar is taken kept outside, but now there is a inverter a bubble that is a not operation; that opposite operation that is being shown here. So that means, whatever is the value here it is opposite here, ok. So, this is these are the 2 conventions we will find in the textbook.

So, now we can whatever truth table we had seen before, now we can put it in a more compact form since we are trying to come up with a compact representation. So, 0 0, so previous value is written, previous state is written 0 1 of S and R output is 0 1 0 output 1, state becomes 1 and 1 1 is not allowed. So, we had seen the way the nor latch works. So, have you if you had connected you know NAND gates in the same manner right, so what would have been the corresponding you know truth table, ok.

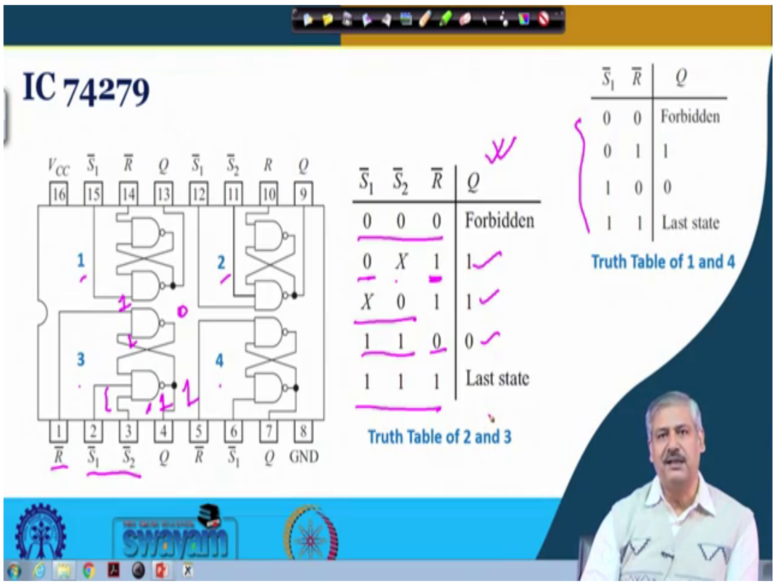
So, in the NAND latch if you look at it. So, NAND for AND and NAND what is the forcing input, 0 is the forcing input, ok. So, if AND gate 0 is there then respective of the other input the output will always be 0. And for NAND gate if 0 is there output will be one irrespectively of the other input,. So, in this case if you just look at the this first this particular block, so we will see that if this we are giving it in a S bar and R bar because that is the way we see we shall we will see that relationship with this particular table; the previous nor latch.

So, if both of them are 0 0 right, so the output will be become 1 1, right. And if both of them are 1 1, then it is one enforcing input. So, it depends on the previous value. So, if the previous value was say 0 and this was 1. So, 0 and 1 this will be 1 and 1 1 it will be 0. So, previous value will be returned. And previous value was 1 and 0. So, this will go to over and 0 1 1 it will become this 0 it will go to 0 and this will become 1, and 1 will come over here and 1 1 it will become 0 ok; the other case, right. You can see it from symmetry also, right.

So, this 1 1 is your previous state, right. Similarly you can see 1 0 will be output will be 0, 0 1 will be 1, and 0 0 is 1 1 because is forbidden because after that if 1 1 follows it is output becomes unpredictable and depends on the rest between this gate and this gate, ok.

Now if we are looking for a SR latch made out of this and we want a truth table like this so what we need to do; we need to put a NOT gate before it, isn't it. So, a NOT gate before it and this NOT gate you can get by a NAND gate also by joining the 2 inputs, right. So, this S bar becomes S here and R bar becomes R here. So, you can use IC 7400 which has got you know which is quite 2 input NAND gate to realise this one, fine.

(Refer Slide Time: 18:16)



IC 74279

$\bar{S}_1$	$\bar{S}_2$	$\bar{R}$	Q
0	0	0	Forbidden
0	X	1	1 ✓
X	0	1	1 ✓
1	1	0	0 ✓
1	1	1	Last state

Truth Table of 2 and 3

$\bar{S}_1$	$\bar{R}$	Q
0	0	Forbidden
0	1	1
1	0	0
1	1	Last state

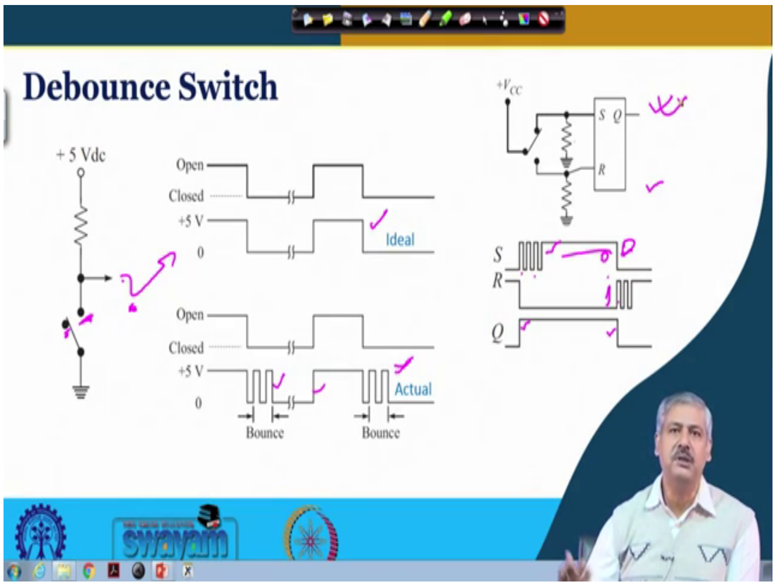
Truth Table of 1 and 4

So, this is one IC where within it the SR latch is already made, ok. So, this particular IC if you see there are 4 such latch one IC 74279 2, 3, and 4. So, 1 and 2 is normal SR latch, ok. Since it is made of NAND, so you can appreciate that the input will be S 1 bar it is given a name S 1 and R bar, ok. And in the other case you have got 2 inputs S 1 bar and S 2 bar. So, this is a 3 input NAND gate and this is R bar. So, for the other gate we have got only one input. And for that we shall have a truth table something like this; how, ok.

So, you can see that if this particular case all of them are 0 this is forbidden, because then the output will become 1, right. And then if S 1 is 0 then this output is 1; this output is 1, right irrespective of S 2 is 0 or not because one of the forcing input has become 0 for NAND gate. And at that time if R bar is 1. So, this R bar is 1 right, just by considering the way we had considered it before we can see that this is 1 this R bar is 1, so 1 1 it is 0 and 0 is (Refer Time: 19:56) here it is 1. So, the output is 1.

Similarly, from symmetry if S 2 was 0 at irrespective of S 1 the output will be 1. And when both of them are 1, right; that means, this is non-enforcing and R bar is 0, then the output becomes 0, right. And when all of them are 1 the output is in the last state, ok. So, this is what we can see for IC 74279, where the latch is already there inside the IC we do not need external connection using NAND gates.

(Refer Slide Time: 20:36)


Debounce Switch

Now, one application of, there will be many applications, we shall see later and complex sequential logic circuit will be made ah. So right now, we can see one application where a debounce switch can be made using SR latch, ok.

So, what do we mean by debounce switch? So, when we are basically connecting a mechanical switch which is kind of you know having a spring load kind of thing. So, whenever it is getting connected here, right. So it will make a connection disconnection connection discussion connection and then it will settle, so there will be a small vibration, right. And when it disengages from here, this is the output that we are talking about here when it disengages when it is connecting here it is making a oscillation, but it is connected to 0, so it does not make any difference here. So, only when it is getting connected at this place getting closed at that time there is a small vibration because of the you know spring action that is there.

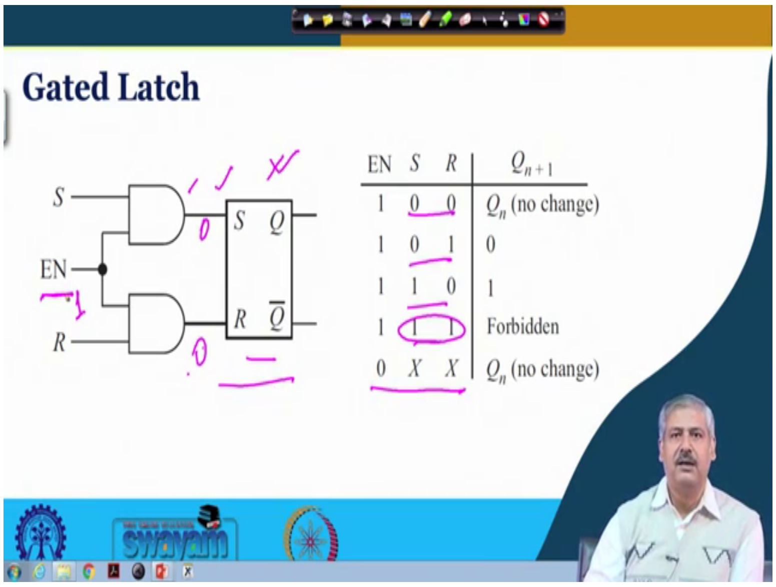
So, you will see you will see that whenever it is getting closed for example here, so there is a small vibration taking place. High-low, high-low it is getting disengaged getting engaged, right. When it is getting open and connecting to this place, so it is a immediate so there is no such issue again it is getting here. So, when ideal case we want this, but actually this is what is happening, clear. So, this ringing can create issues, because it might say that number of you know 0s 1s you know come into the picture which might create difficulty. So, it can trigger many events when, actually one event has taken place one such closer has taken place it may count that 1 2 3 such closer has taken place. And each closer might you know can get counted and lead to certain action going forward, ok.

So, that is something which may be needed to be I mean avoided in certain application. So, for that we can use SR latch, right. So, in the SR latch this which is the one is S and 1 that is r, right. So, when it is disengaging from R and connecting to S, so that is a ringing over there, ok. So, whenever that is happening, so 1 and 0 the output is becoming one because it is getting set after that when it is getting S because of the ringing, so it is becoming 0. So, this is a 0. So, previous state is written, ok.

So, there is no issue with that, is it clear. Similarly when it is coming to when it is getting disengage from S and coming to R right, it is getting disengage no issue, but when it is connecting to R there is a small vibration. So, R is becoming 1 0 1 0 for sometimes. So, whenever R becomes 1 for the first place and this is 0 right, so it will get reset; so it has got reset. After that during ringing this is this remains 0 and this is 0. So, 0 0 means previous state is written, ok.

So, if you now take output from here there will be no effect of bounce. So, effectively the got a debounce switch.

(Refer Slide Time: 24:04)


Gated Latch

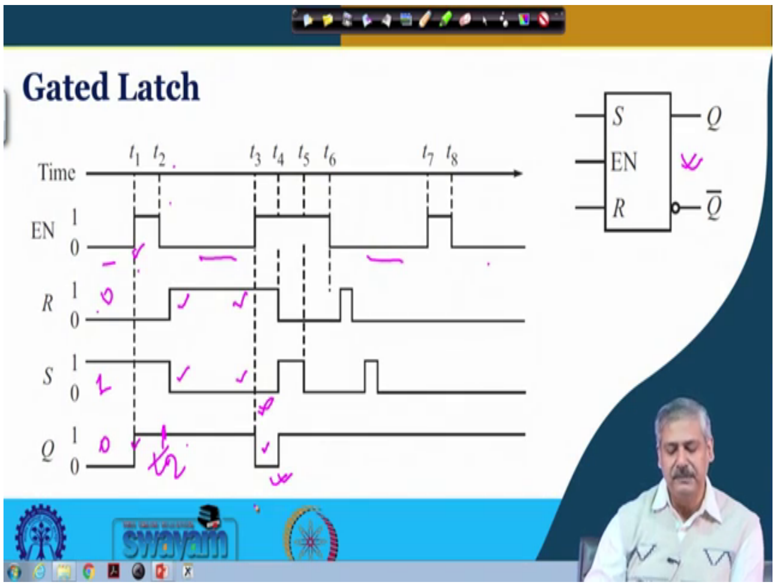
EN	S	R	Q_{n+1}
1	0	0	Q_n (no change)
1	0	1	0
1	1	0	1
1	1	1	Forbidden
0	X	X	Q_n (no change)

Now, we move little bit further, ok. We look at further use of latch the basic latch by introducing an enable input, ok.

So, this is the basic SR latch which could be made up of NOR or NAND, as the case might be, ok. So, that it that part we shall see more you know elaborate circuit later. So, what we are doing here that we are putting, we are making S go there through 2 AND gates like this, which has got a enable input. So, when enable is 0 so these 2 output are 0. So that means previous state is rewritten, ok. So, this is the case no change. And when enable is 1 when enable is 1 and that time depending on whatever we is this SR value this 1 1 is forbidden of course, the normal action takes place; is it clear.

So, we are not allowing the SR input to affect the final output at any time;at all the time. We are only allowing it when it is enabled, ok. It is like strobe or you know he kind of thing that we had used before; for multiplexer, demultiplexer those circuits if you remember, ok. So, the circuit this circuit is acting only when or make toward only when enable is at 1, ok.

(Refer Slide Time: 25:55)

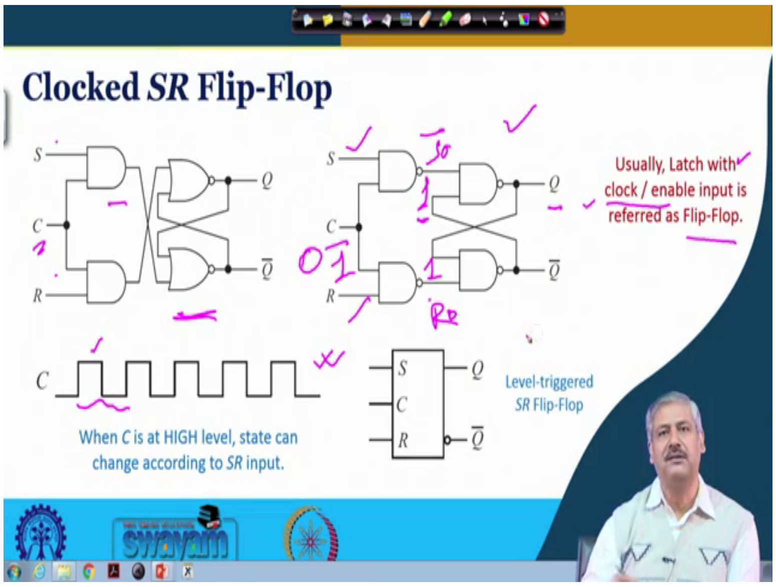

Gated Latch

So, what is its implication? So, it will look at a timing diagram, right. So, you can see that when enable is at 0 like this place, this place, this place, this place, right whatever changes are there it does not actually I mean the Q will remain at that value in those cases. So, t 1 t 2 is the window you which any a change can be accommodated. So, S was 1 and R was 0, and the Q was 0, so that makes I mean if enable was not there whenever this S and R; S was 1 and R was 0; so the Q would have been 1. But the enable comes over here only here at this point at t 1. So, at that time only Q will go to 1, right.

Now t 1 to t 2 there is no change in S and R. So, Q remains at 1 at t 2 at this point t 2 after that enable is 0. So, after that R becomes 1 and S becomes 0 at some point of time over here. But that affect does not get reflected in the output Q why, because enable is at low, right. So, again it gets trigger; again it gets the enable signal over here at t 3 at the time it sees it finds what is the value of S and R at that time. So, S is 0 and R is 1,. So, that time then it will go to it will become reset. So, in between before the enable was there if S and R had change their value, it would not be captured by the final output, right. So, only when the enable is there whatever is the value of S and R that gets transmitted and reflected in the final output. So, this is the idea, ok.

So, ah. So, whenever we want the SR flip-flop to work we put the enable, otherwise we allow the S and R input to settle if there is any transient and all. We do not want the final output gets you know changed before S and R is you know properly settled. So, enable is giving that you know option period for the inputs to get settled.

(Refer Slide Time: 28:28)


Clocked *SR* Flip-Flop

Usually, Latch with clock / enable input is referred as Flip-Flop.

When *C* is at HIGH level, state can change according to *SR* input.

Level-triggered *SR* Flip-Flop

And the this gives rise to what we end with the todays discussion is called clocked SR flip-flop, ok. So, the sequential circuit that we shall discuss it is called it is it will be actually synchronous sequential circuits. So, most of the sequential circuit are synchronous sequential circuit, where the state change takes place synchronous with a external clock, ok. So, these external clock actually totally decides how the these different the sequential operation of the different kind of elements inside the circuit, ok.

So, this particular clock would have you know a waveform something like this right, so it goes high and low and all. So, this is one clock cycle, right and in each clock cycle you are looking for one state change if it is ah. So, desired by the input to that particular bistable circuit. In this case this particular SR latch that we have discussed, ok.

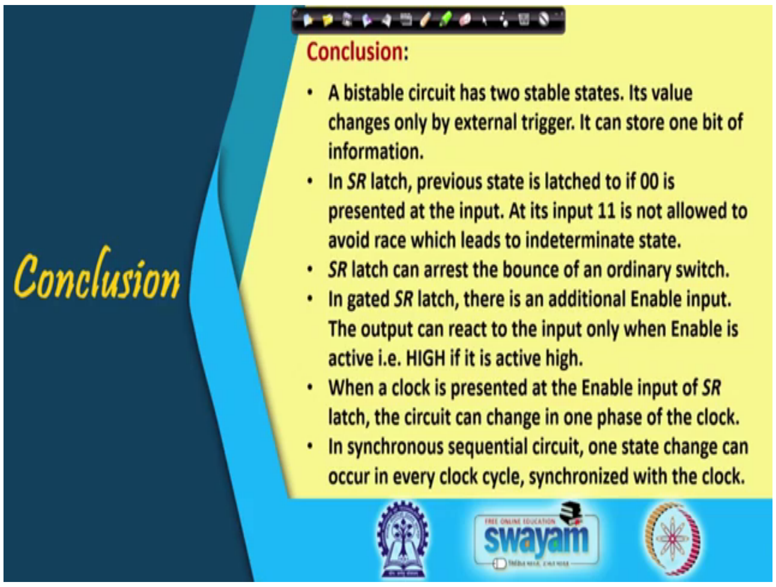
And usually latch with a clock is called a flip-flop, ok. But then you will see that in the text it is says latched SR flip-flop. So, basically a latch type is SR flip-flop. So, basically they are referring to latch SR flip-flop, but in other text general generally speaking presence of clock with I mean flip-flop means a presence of a clock, and latch means it is of the last stage of where the memory part is there, where their the value is latched into, ok. So, that is when distinction is often made in this particular field,.

So, then how this clocked SR flip-flop would look like. So, in place of now enable we are putting the clock, ok. So, whenever the clock is high, right. So, the input change in the input can be this S and R can be reflected in the output, ok. So, in this case for SR flip-flop this is a SR latch right, and this is the AND gate and this was enable before. So, we have put the clock over there, right. And for NAND base circuit we remember that this was a NOT gate, right 1 2 input of NAND gate where put together to get S and R otherwise this was S bar and R bar, right.

So, we can now use the other input of the NAND gate to put the clock, ok. And we can see the clock is 0 this output will be 1 1, these are the forcing input for the NAND gate. So, 1 1 means, right a for a NAND gate 1 1 means previous state will be retained because that this is non-enforcing input for the NAND gate. And only when it is 1, so this is now non-enforcing. So, based on S and R the value will be transmitted here and accordingly the final value will be arrived at, ok.

So, this is the basic SR clock flip-flop circuit. And the symbol for this is this the one that you see here. And this is also called level triggered flip-flop, because we will see S triggered flip-flop in the next class. So, level triggered flip flip-flop means whenever clock is at a particular level. So, that time it is allowed to trigger allowed to change the state, ok. And if we have put an inverter over here a NOT gate over there, then the change would have been in this stages of the clock when the clock is low; isn’t it. So, at that time would have would have said that if this is positive level triggered high level triggered, so that would have been negative or low level triggered, and at the time there would have been a bubble sign over here, ok. That is a difference. Is it clear?.

(Refer Slide Time: 32:45)



Conclusion

Conclusion:

- A bistable circuit has two stable states. Its value changes only by external trigger. It can store one bit of information.
- In *SR* latch, previous state is latched to if 00 is presented at the input. At its input 11 is not allowed to avoid race which leads to indeterminate state.
- *SR* latch can arrest the bounce of an ordinary switch.
- In gated *SR* latch, there is an additional Enable input. The output can react to the input only when Enable is active i.e. HIGH if it is active high.
- When a clock is presented at the Enable input of *SR* latch, the circuit can change in one phase of the clock.
- In synchronous sequential circuit, one state change can occur in every clock cycle, synchronized with the clock.

So, with this we conclude the today's class. A bistable circuit has 2 stable states; to summarize it value changes only be external trigger SR latch, 0 0 is the input when the previous state is latched into, 1 1 is not allowed and we can get bounce switch out of ordinary switches using SR latch. And in gated SR latch or a clocked SR latch there is additional input which enables SR input to go pass through the make changes in the final output. And in synchronous sequential circuit this clock is very important and one state change we are looking in every clock cycle synchronized with the clock, ok.

Thank you.