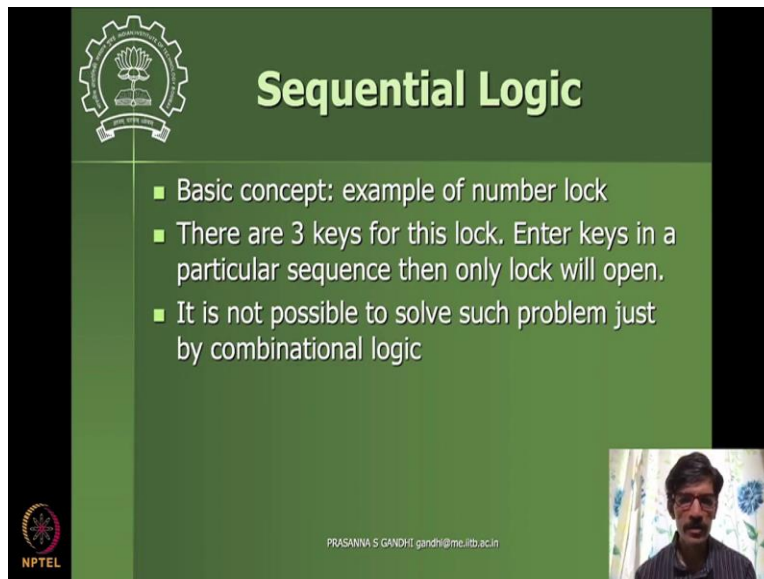


Design of Mechatronic Systems
Professor Prasanna S. Gandhi
Department of Mechanical Engineering
Indian Institute of Technology, Bombay

Lecture 14
Microprocessor Building Blocks II-Sequential Circuits

So, in today's class, we look at another building block of microprocessor, which is sequential logic circuits, we will build some kind of a fundamental concepts around sequential logic and slowly we will develop the need for the memory elements which are built from your flip flops. So that is how we will kind of structure this lecture.

(Refer Slide Time: 0:57)



The slide features a green background with a white gear icon containing the IIT Bombay logo in the top left corner. The title 'Sequential Logic' is displayed in a large, bold, white font. Below the title, there is a list of three bullet points in white text. In the bottom left corner, there is a small NPTEL logo. In the bottom right corner, there is a small video inset showing a man with glasses and a mustache, identified as Professor Prasanna S. Gandhi, speaking. The email address 'PRASANNA.S.GANDHI@me.iitb.ac.in' is printed in small white text at the bottom center of the slide.

- Basic concept: example of number lock
- There are 3 keys for this lock. Enter keys in a particular sequence then only lock will open.
- It is not possible to solve such problem just by combinational logic

So let us start with the, you can see the slides. So, the sequential logic, basic concept we will develop through an example, some of the problems in the domain can be solved without any use of microprocessor, I can kind of solve those problems directly by using some sequential logic circuits and these circuits can also have some kind of interfacing with the, with some other mechatronic interfaces.

So, that is why it is important to like understand what is possible with sequential logic. So that we can at time some solve some of the problems with lesser cost by using sequential logic rather than employing microcontrollers or microprocessors for such a task. So, we will develop this concept by using the example of a number lock.

So, this number lock has very simple primitive kind of number lock, has three keys and you are asked to enter the keys in some particular sequence and then only the lock will open, the lock does not may or may not have other keys. So right now we consider lock does not have other keys. So, can this problem be solved my combination logic is a question.

(Refer Slide Time: 2:27)

The slide features a green background with a white gear logo in the top left corner. The title 'Sequential logic design: Example' is in large white font. Below the title, a list of bullet points discusses a combination lock problem. To the right, there is a diagram showing three switches connected to a pink box labeled 'logic'. Above the switches, a green box contains the numbers '1 2 3'. Text next to it says 'Correct order to press is 123 for example'. A small video inset of a man is in the bottom right corner. The NPTEL logo is in the bottom left, and the email 'PRASANNA.S.GANDHI@me.iitb.ac.in' is at the bottom center.

- Combination lock
- As a first cut can we try the problem using combination logic?
- Say we just use AND of 3 key presses what is the problem?
- → we do NOT get the notion of sequence isn't it?
- How do we achieve this?

Correct order to press is 123 for example

logic

NPTEL

PRASANNA.S.GANDHI@me.iitb.ac.in

So let us say we can try 1 attempt. So you have these kind of keys 1 2 3 as given here, let me get my pointer right. So, if there is press in the sequence 1 first, two second and three last then only these lock should open, any other kind of sequence, lock cannot open. So, we are given this kind of problem to solve. Now, if we put our combination logic with the switches, switches to kind of give 1 or 0 when switch is on like no 1 input is given to the, to output of that switch anything like that.

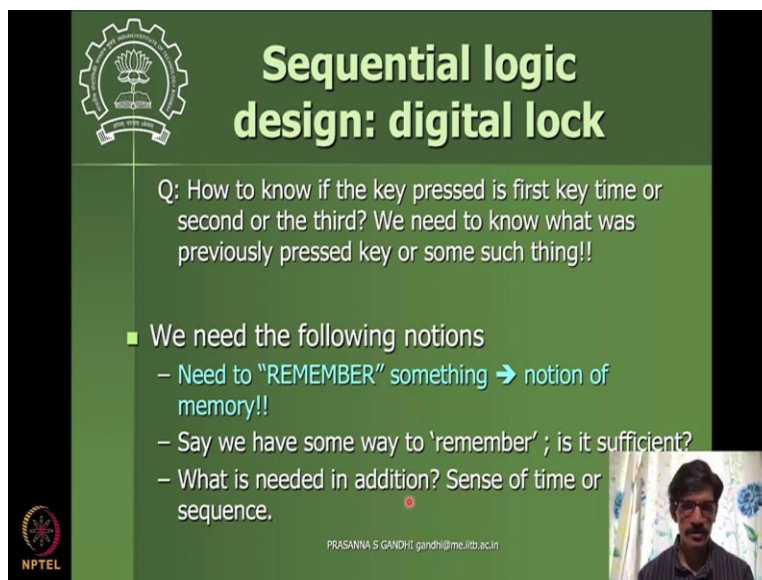
So, we have these three switches when the switch is pressed that means a key is pressed then the logic 1 will be there and we use a say AND gate, all the three switches are pressed then I can open the lock but now we do not get notion of sequence here like see if I press this 2 first and then like 3 and then 1 still my logic, AND logic will just kind of say that, I see all the three are pressed and it will open thing.

So what is missing is the notion of sequence that I have pressed this first and then I am pressing the other 1 that, like what is previously pressed that notion of the sequence is missing. So, how do we handle such a kind of a scenario for the person designing circuits which will achieve this

kind of a job, that is what is whole sequential logic about that where combinational logic fails combinational logic is only like given set of inputs, you have fixed set of outputs.

And there is notion of whether this is 1 case press first or the other key is pressed first, that notion of sequence is not there in combination logic. So far we have seen in the last class, the combination logic circuit class, all the circuits are of that kind. So, now, how do we do, develop these concepts here.

(Refer Slide Time: 4:44)



The slide features a green background with a white gear icon containing a book and a lamp. The title 'Sequential logic design: digital lock' is in large white font. Below it, a question asks how to know if a key is pressed first, second, or third, requiring knowledge of previously pressed keys. A list of concepts includes the need for 'REMEMBER' (notion of memory), the sufficiency of a 'remember' mechanism, and the need for a sense of time or sequence. The NPTEL logo is in the bottom left, and a small video inset of the presenter is in the bottom right.

Sequential logic design: digital lock

Q: How to know if the key pressed is first key time or second or the third? We need to know what was previously pressed key or some such thing!!

- We need the following notions
 - Need to "REMEMBER" something → notion of memory!!
 - Say we have some way to 'remember' ; is it sufficient?
 - What is needed in addition? Sense of time or sequence.

NPTEL

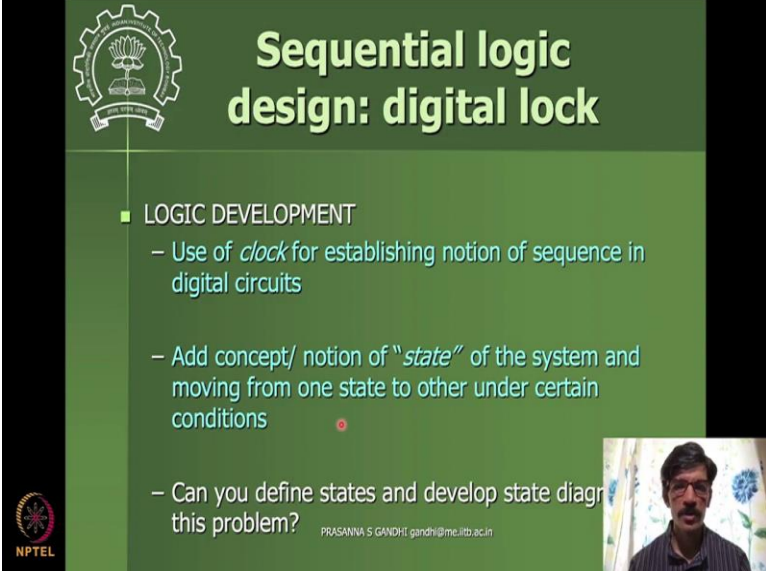
PRASANNA S GANDHI, gandhi@mc.itb.ac.in

So that is what we will look at. So, the question here is how to know if whatever key pressed is, is a first time you are pressing that key or some key has already been pressed and you are pressing this key after that. So, we need to know what was the previously press key and beyond that if this key is pressed then we have some kind of knowledge that is happening which will further transform into opening or not opening the lock.

So, we need first a notion an idea that we need to remember something, what is the activity that we need to remember, something what is like press before or something like that. So, this notion of memory is that here and then other notion that we will need is the time or sequence. So, just remembering is not sufficient we need to know, now sequence kind of notion or notion of time or sense of time it needs to be there for this kind of problems.

These are the two notions that we need a sense of time and this notion of memory. So, suppose we have that and we want to solve this problem.

(Refer Slide Time: 6:04)



The slide features a green background with a white gear icon containing a lamp and an open book in the top left corner. The title 'Sequential logic design: digital lock' is written in white. Below the title, the text 'LOGIC DEVELOPMENT' is followed by three bullet points. The first bullet point discusses the use of a clock for sequence, the second discusses the concept of state and transitions, and the third is a question about defining states and state diagrams. The NPTEL logo is in the bottom left, and a small video inset of a man is in the bottom right.

Sequential logic design: digital lock

■ LOGIC DEVELOPMENT

- Use of *clock* for establishing notion of sequence in digital circuits
- Add concept/ notion of "*state*" of the system and moving from one state to other under certain conditions
- Can you define states and develop state diagram for this problem?

NPTEL

PRASANNA S GANDHI | gandhi@me.iitb.ac.in

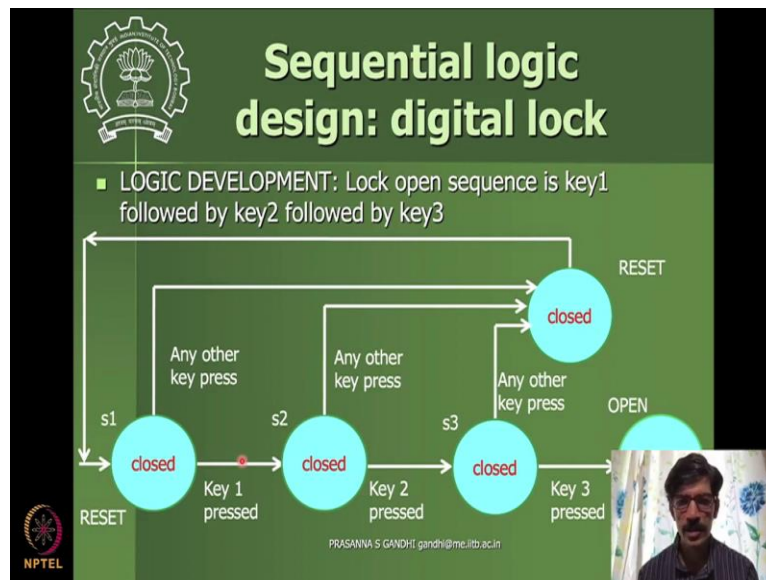
How do we go about doing that, how do we develop this now logic for say, we are given memory, we are given some notion of sequence or time, then how would we attack or attempt this problem? So, for this then there is, we need to define something called states. So, we introduce something called states so that we can introduce the notion of sequence for going from 1 state to other state to other state like that.

So, that is how we need to define states. That is how, so we will see how that can happen for this particular problem. And then this concept or these notions can be generalized to solve any other such problem by using a sequential logic concepts. So, we use clock for establishing some kind of notion of time and then we introduce this notion of states to say I need to remember now a previous state of the system. And then I can proceed for the problem.

So, this notion of state is for giving you this kind of a utility to use this notion of memory for solving such a logic problem. So, how do you develop with this what kind of logic 1 can develop? How do we can develop, so think about this, pause here for a moment and then like you can proceed to see the solution.

So, see, these pauses are important to develop something in your mind to attempt the problem, even if you do not get the problem is fine, but it is important to pause the video and think about and develop, do something on your own and then understand then something will open up, that is a way the understanding happens. So as we say like we have seen in the first class also. So, you need to ponder all these questions, can you define states and develop a state diagram for this problem. So, how do we go about doing that, you pause and then like proceed.

(Refer Slide Time: 8:27)



So, we develop like you know, these two states like for sure, we know that, the lock is open or lock is closed. Now, what we want actually is that when the lock should open, when say some sequence is there. So sequence is what key 1 is pressed, then followed by key 2 is pressed and followed by key 3 is pressed then my lock should be open. So when my key 1 is pressed, and is the first key that is right key, then I can change the state of the system.

So like somebody presses a key, from reset point,. It comes to the lock and he presses a key. Now if that key is right, first key is right, key 1 is pressed, and it is right, then I go to the second state. If that key is wrong in the first place itself, I will go back to this close state again. Then if a person from that state to when state 2 is there system is in state 2, and person presses a key 2, which is again the right key, then I move to the next state, state 3.

Otherwise, if the key is not correct, then again, I will go back to the reset kind of state, then, so suppose key 2 is also right. And now I am in state 3. At the state 3, if the case 3 is pressed which

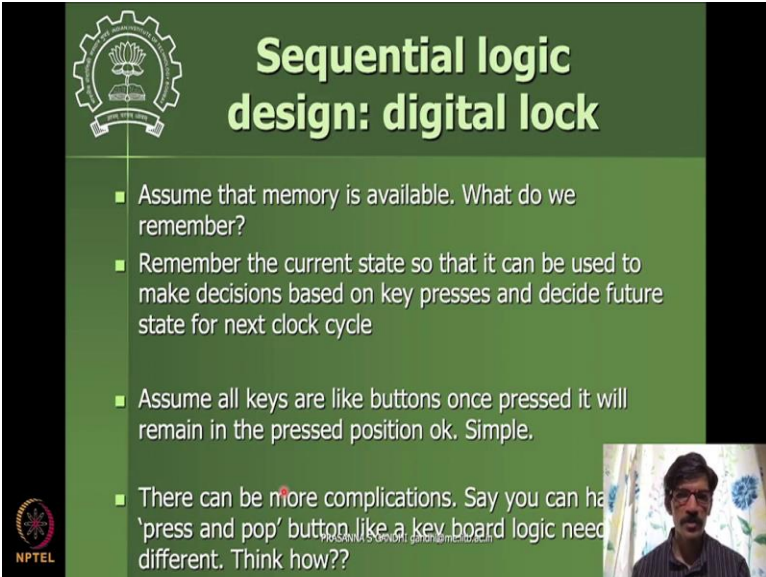
is correct key that is pressed then I will open the lock that is a kind of a way we define the states and we say that this is how like the system will progress from 1 state to other state to other state when the right keys are pressed, and my lock will be open.

So, now how do we complete this diagram like so right now this key pressed is correct key if other (k) key is pressed then we need to kind of introduce that also in the state diagram. So, any other key is pressed you get to the reset state which is closed lock. So, this is how one can start developing the logic about these kinds of problems.

So, you need to use judiciously this notion of state, what you define as a state is up to you, there are many different ways one can define what the state for a given problem is. And many different states, how many how many numbers of states that you want to use you can use and develop these kind of a state diagram based on which one can solve.

Once you have this diagram, so the way you have a table in the combinational logic, in the sequential logic you develop this kind of a state diagram, once you have a state diagram, then one can start writing a table. So, how do we develop now a table so, let us say this state S1, S2, S3 are there as it is like, we call them as a state S1 and then like we say there are these key 1, key 2, key 3 that are pressed and we will introduce in the state diagram a notion of previous state, so that is where, that is how we will use a memory.

(Refer Slide Time: 11:44)



Sequential logic design: digital lock

- Assume that memory is available. What do we remember?
- Remember the current state so that it can be used to make decisions based on key presses and decide future state for next clock cycle
- Assume all keys are like buttons once pressed it will remain in the pressed position ok. Simple.
- There can be more complications. Say you can have 'press and pop' button like a key board logic need different. Think how??

NPTEL

© 2011 NPTEL. All rights reserved.

So, memory is available, what do we use it for, to remember the current state or the current state, so that it will become a previous state when you start doing the thing. So, you remember this state of a system so that you can take a decision for the future then. So, now, there are this some kind of more nitty gritty facts that assume all keys like buttons once pressed they will remain pressed. So, 1 can have these kind of a way or if there is something button which is push button, then there is a different kind of way things are coming up.

So we will assume right now that , this is, this is that and there can be more complications, say you can press and pop the button, like keyboard logic. So, these are all more and more nitty gritties, we do not worry about that right now. So, our focus is to develop these ideas of defining states and solving the problem by using states and then like, how do you go about solving that problem is what we are looking at here now.

So, how do you develop a state diagram is what you need to focus on and pause here again you develop your own state diagram, the table of the as we have already develop the state diagram, you develop a table around it saying that, key 1 key 2 key 3 are the three inputs and state, previous state is another input. And then what we want as a current state, that is a kind of a table that we need to develop.

(Refer Slide Time: 13:30)

Sequential logic design: digital lock

Key1	Key2	Key3	STATE	Next STATE	LOCK
0	-	-	S1	S1	Closed
1	0	0	S1	S2	Closed
1	1	0	S2	S3	Closed
1	1	1	S3	OPEN	Open

- For all other combinations of keys and whatever be the state the next state should be s1 then lock will remain closed
- Memory block will make previous state available for given clock cycle to decide the next state AND of the state to the next state every clock cycle
- Now states (4) can be represented further as binary number to get the logic circuit with a memory block

So, you can see here the key 1 key 2 and key 3 are the three inputs and previous state another input and next state is the output of the system, output of this logic table. So, you remember your

combination logic has always this some inputs and some outputs. So here inputs have this additional input which is based on the memory, this is a previous state as a memory here.

So now say for example, now key 1 is pressed and you are in state S 1 and correct key is pressed like from the reset, then you go to state S2 then now key 1 is pressed and key 2 is also pressed. There like the next key is pressed is key 2, key 2 then you are in S 2 state then you move to S3 state. So like that you will have this logic of that.

So all the keys are pressed and you are in state S3 then you are in open. So now if the keys are pressed and in some other kind of a sequence then like, you will have this next state will always be S 1. So any other kind of a combination of previous states and the keys would lead to state S 1 as a next state, that is how we define this complete truth table for this sequential logic.

So, the first step is about developing the state diagram and after the state diagram is developed, then you introduce this previous state as a memory based that is what we remember in the truth table, and then you move on to developing the complete truth table. And then once you know, once you have this table, you can use Karnaugh map or whatever ways to get your outputs expressed in terms of inputs and then you draw the circuits.

Now, how do you draw the circuits? We will see that later. How do you kind of use this memory notion for that you need some kind of logic gates or which are having this memory notion in into them. So that is what we will worry about so for example, now the states can be, these are like some right now, defines as some kind of abstract thing that they need to be coded into some kind of binary numbers say I will say 00 is my S1, 01 is S2 and 10 is S3 like that I can define this this kind of a three states.

And then so, you can have then logic developed based on this and then you can do your output expressed as a function of inputs and so on and so forth, all the things can happen that way. Only thing right now, we are kind of not so clear about is how do you introduce this notion in the circuit, what is the element in the circuit that is required to introduce this previous state notion. So, we will see that in a minute now. So, how do you create memory is a basic question.

(Refer Slide Time: 17:05)

How do we get memory? Basic Concept

- Use feedback in digital circuits
- How and why feedback gives ability to remember a state? → we will see in the next slides
- These circuits form basic blocks for building memory elements

NPTEL

PRASANNA S GANDHI gandhi@me.iitb.ac.in

NPTEL

Using feedback to create memory

- Example: two inverters with feedback
- Will hold the value as long as power is applied
- How to get a new value in memory cell??
 - Selectively break feedback path

NPTEL

PRASANNA S GANDHI gandhi@me.iitb.ac.in

NPTEL

So, what gives you memory in these digital circuits and the simple kind of a way to create memory is used feedback. Using the feedback in digital circuits we will give you some ability to remember state, how does it happen? Very simple. So, you say for example, you have these two inverters and then this value you are giving as a feedback here.

So, this value say for example, this value is 0, this value is 1 here, this 1 is getting as a feedback and then this 1 is again producing 0 here that means, this is like a stable state here for this circuit, circuit is stable holding this value 0 here and 1 here. So, as long as the power is applied, this

value will be held. Now, we can kind of introduce the way to like to remove the value and store new value.

So, how do you get this new value, so you will break the path. So, this is I say remember kind of a switch here. So, you open the switch and put like a new value into this circuit and then moment you close this switch that new value will be remembered now, whether it is 0 or 1 does not matter, if it is 1 here also the same thing happens. That is a stable circuit. But now, we cannot do this switching on and off manually.

(Refer Slide Time: 18:45)

The slide features a green background with a white gear logo in the top left corner. The title "Using feedback to create memory" is written in a light green font. Below the title, there are two circuit diagrams. The top diagram shows a feedback loop with a switch labeled "Remember" that can be opened or closed. A "load" switch is also present. The bottom diagram shows a similar circuit but with a red circle containing a '0' on the feedback path. To the right of the diagrams is a list of steps: "To load value" followed by four bullet points: "Open remember switch", "Press load switch and give 0 or 1 on new data as need be", "Close remember switch", and "Open load switch". In the bottom right corner, there is a small video inset of a man speaking. The NPTEL logo is in the bottom left, and the email "PRASANNA.S.GANDHI@me.iitb.ac.in" is at the bottom center.

Using feedback to create memory

- To load value
 - Open remember switch
 - Press load switch and give 0 or 1 on new data as need be
 - Close remember switch
 - Open load switch

So, that is where one uses electronic switching and so, this is a way like load and value and remember the value but this manually we cannot do these. So that is where like you introduce this other gates to have electronic switching kind of action going on. And that is where the RS flip flop comes into picture.

You might have I mean, as a prerequisite you might have gone through some of the RS-flipflop literature and D-flipflop literature and you may know how they are working, we will just touch upon some of the aspects of them, we will not get into a lot of details about them. So, if you want to you can brush up that knowledge again, but the fundamentally from microprocessor perspective that is a first kind of a you know, the basic memory element into the into the system.

(Refer Slide Time: 19:44)

R-S Flip Flop (Latch)
Another way of using feedback without need of switches to load

Recall:
NOR

A	B	Z
0	0	1
0	1	0
1	0	0
1	1	0

- How do we analyze circuit shown?
- NOR gate with feedback from other
- Recall NOR truth table
- Observe: NOR gate with one 0 input \rightarrow inverter wrt other input

PRASANNA S GANDHI | gandhi@mc.itb.ac.in

NPTEL

R-S Flip Flop (Latch)
Another way of using feedback without need of switches to load

Recall:
NOR

A	B	Z
0	0	1
0	1	0
1	0	0
1	1	0

Case $R=0$ $S=0$

- Observe: NOR gate with one 0 input \rightarrow inverter wrt other input
- So if we consider case $R = 0$ $S = 0$ we have our memory block similar to two inverters in feedback
- So whatever value of Q was previously will be 'stored' or 'remembered'

Case similar to 'remember' switch on

PRASANNA S GANDHI | gandhi@mc.itb.ac.in

NPTEL

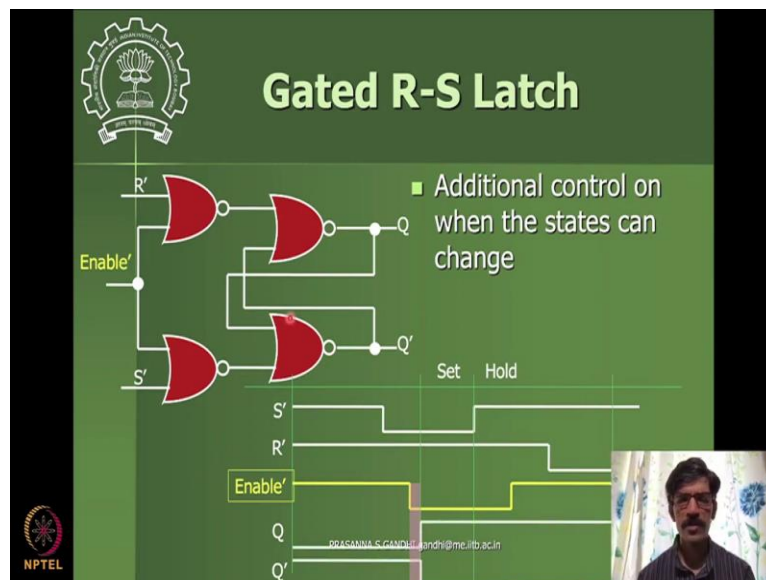
So, now, this way RS-flipflop works is very simple actually, it has these two NOR gates connected with this feedback. This is a recall for the, of the NOR gate with this truth table one can find out, figure out like with different inputs of R and S what is the output going to be and what is going to be the stored values like that one can kind of get sort of truth table done for this RS flip flop and a simple way to think is that this NOR gate with one input 0 it acts as a inverter with respect to other input, one input is 0 then like with respect to other input it is acting as a inverter.


So, one input say A is 00 here, so B is acting as B to Z or B to this output for the NOR gate is acting as a inverter or as a NOT gate here. So, say both R and S are 0 then we have exactly if you redraw the circuit when R and S are both 0 and this is acting as a inverter. This is acting also as an inverter then this simply becomes our two NOT gate connected in series and having a feedback kind of a circuit.

So, this is again like the same concept as what we had for the memory before, we had seen before, when R and S are 0 so when R and S are 0 the circuit is going to remember the values which is stable values and to introduce new values you change the value of R or S, that is a simple kind of a way one can understand this RS-flip flop or RS-latch.

So, we will not get into too many details about it, but one can brush up and like see how these values change can cause like changes in the value of outputs is Q and Q prime and when R and S are 0 then whatever the value is previously there it will be remembered, that is how one can see this as a memory.

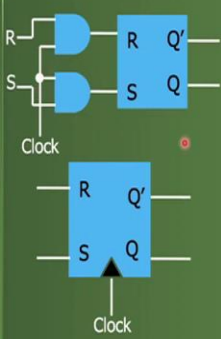
(Refer Slide Time: 22:00)








Gated R-S Latch

Compact Representation



- We will consider more compact representation shown because we may need to connect many such latches together to achieve desired application
- Recall that each nor gate further has so many CMOS transistors
- At hardware everything is a combination of CMOS transistors...

So, these are like some additional controls one can introduce now, to enable this flip flop or disable it and things like that. So, then the changes will happen only when this is enabled then one can add further the gated R S flip flop, where you now use this clock to introduce this clock in the circuit to say that or the changes will take place only when the clock pulse is positive. See, why this is introduced is basically there are so many kind of transistors that will be there now.

So, see so many, even this R S flip flop itself if you go to the circuit level it will have so many CMOS kind of transistors that are working and we want these timings to be happening like or these changes in the state from one R S flip flop in the next state they need to happen in some kind of controlled manner, they need should not be like a runaway kind of manner, they should switch. So, that is why this is introduced here to make sure that when this pulse goes high, then only like your switching is going to happen.

(Refer Slide Time: 23:28)

Connecting Latches

- Another approach: use different polarities of clock on two latches by introducing inverter as shown

so now second latch will be disabled during time the first is enabled so changes cannot rush through the circuit and will proceed in steps corresponding to half clock cycle

NPTEL

PRASANNA.S.GANDHI@mc.iitb.ac.in

So, then like you connect these latches together with the additional kind of inverter here to make sure like the status changes that are happened in the circuit will get reflected to the next circuit only when the next kind of clock comes. So, like that you introduce these gated R S latches along with the clock and is now inverted also. So, when this is acting, then this is disabled, so it will prevent like the changes to rush into this.

So, once these, all these changes have happened here then only this when clock goes whatever high or low, this will get enabled and changes from here will be passed on to the on to this latch and that will happen. Like that you do on the clock cycle, then the things are more robustly achieved when we have scaled kind of a versions of the circuits coming in place for microprocessors.

(Refer Slide Time: 24:44)

The slide features a green background with a white gear logo in the top left corner. The title "D Flip-Flop" is centered at the top in a large, bold, white font. Below the title, there are three bullet points in white text. At the bottom of the slide, there is a circuit diagram of a D flip-flop. The diagram shows a D input connected to two AND gates. The output of the first AND gate is connected to the R input of a first SR flip-flop. The output of the second AND gate is connected to the S input of a second SR flip-flop. The outputs of the two SR flip-flops are labeled Q and Q'. A clock input is shown at the bottom left of the diagram. In the bottom right corner of the slide, there is a small video inset showing a man with glasses and a mustache.

D Flip-Flop

- What if we use inverter between R and S, we will have only one input designated as D.
- Q: can you think how this circuit will behave?
- Since R and S cannot be both zero when clock is enable storing of value will happen only during clock cycle!!

PRASANNA S GANDHIL@meatlab.csi

NPTEL

Now, we now add more, one more kind of a modification. You might have already seen this called a D flip flop, where you put an inverter into R S flip flop, so that R and S here. So, this inverter is put if you see is on the gate, gates here. So, their output is the AND gates. So, this will make sure that the these R and S cannot be both 0 when the clock is enabled and this storing value will happen only during this clock cycle.

So, see this with this will make sure this, you can just kind of do a little bit analysis and you will find out that that R and S cannot be both 0 and then these changes will happen only during clock cycle. So, see when the clock is disabled for this this part then R and S are going to be 0 because these are AND gates here.

So, R and S are going to be 0 that 0 both 0 will store some value. But then this input cannot be put 0, either 0 or 1 value is going to get stored or like it is going to get stored in based on the input that is given at D. So, that is what is this inverter is doing here. So, and that is how these D flip flop is getting filled.

(Refer Slide Time: 26:37)

D Flip-flop

This is called D Flip-flop (flip-flop instead of latch since insensitive to momentary glitches at input)

- Characteristics equation is simple
$$Q^+ = D$$
- D-negative edge-triggered flip-flop
- Many other variants of flip-flop are available. Ex JK flip-flop

The diagram shows a D flip-flop circuit with inputs D and Clock, and outputs Q and Q'. The circuit consists of two SR flip-flops. The D input is connected to the S input of the first SR flip-flop and the R input of the second SR flip-flop. The Clock input is connected to the clock inputs of both SR flip-flops. The outputs of the SR flip-flops are Q and Q'.

NPTEL

PRASANNA S GANDHI, gandhi@mc.itb.ac.in

So, with this we get this D flip flop to remember. So, this is like Q plus state will be coming out here though this Q plus is 1 after one clock cycle this plus means after one clock cycle, so, if this input is D after one clock cycle that input will be reflected on this queue here. So, this is how this D flip flop will work and this is particularly negative edge triggered D flip flop.

So, there are many variants of this flip flop but for our case right now, what is most important is this memory notion that we are talking about, the notion of memory comes now here that a state will be remembered for one clock cycle. And then the next state will be like changes of the state will happen after one clock cycle.

So, at any point if you say this is input I am giving here, but output here is going to be previous state at any point of time, this input will, this output will become a new state only after one clock cycle gets passed. And after this, this is changed to say like new value and my value is changed here, this does not immediately change. So, this will be changing only again after next clock cycle. So, previous value will be remembered here till that time. So that is how this notion of a memory is introduced here. So, in this D flip flop.

(Refer Slide Time: 28:10)

Two stage shift register

- Positive edge Triggered

Can you complete timing diagram below?

NPTEL

PRASANN S GANDHI

Four Bit Ring Counter

- Simplest counter: timing diagram?

- How many states are getting represented? 1000, 0100, 0010, 0001
- Q: what if we start at 0000??
- Reset to 1000 needed to start
- How to reset to 1000???

NPTEL

PRASANN S GANDHI

So, these are like some variants here, but and some kind of a case for you to think about and consider how these input changes to the to the D flip flop, this D is input and how outputs will change and how the states will change in the clock that changes so think about this and then we can now put together this D flip flops to get different different kinds of circuits, the way you, we had some kind of a combination logic circuits we have now these four bits say for example this ring counter, so we want to count these numbers here.

So we are putting some kind of a these D flip flop in series and like looking at the output of these flip flop as the clock ticks and these in the output of the last one is again given as an input to this

first one. So under this scenario, like how the ring counter will work as the clock ticks. So if you start from say 0, 1000 state, so let us say somehow we have got the state inside here, after one clock cycle.

So, they say this is 1 this is 0, this is 0, this is 0. After one clock cycle, this 1 is going to get passed on to these. So this will become 1 what will come here is what is here, what is here is the output of this 4 so this 0 is output right now, the fourth like last part 1 0 here, that will come as a next state. So this is how like the next state will be 0100 and next, the next state will be 0010, anything like that. So, this is one of the ways one can be this kind of a counters.

(Refer Slide Time: 29:57)

Four Bit Ring Counter

RECALL

- D-FF with additional functionality (inputs: R,S)

■ Still this counter represents only 4 states.

■ How to increase number of states represented? Can you think of way? What we desire is increase of states!! What is maximum number of states represented by 4 bit binary number??

NPTEL

PRASANNA S GANDHI | gandhi@me.iitb.ac.in

Four Bit Johnson Counter

- One simple way: use inverter in feedback loop

- Q: can you now run through time and see how many states would get represented if we start at say

PRASANNA S GANDHI gandhi@me.iitb.ac.in

Four Bit Johnson Counter

- How to analyse the outputs when clock proceeds?

	Out1	Out2	Out3	Out4
T	1	0	0	0
	1	1	0	0
	1	1	1	0
	1	1	1	1
	0	1	1	1
	0	0	1	1
	0	0	0	1
	0	0	0	0
	1	0	0	0

PRASANNA S GANDHI gandhi@me.iitb.ac.in

These counters are very important in microprocessors, microcontrollers both. So, there are many variants that can be possible, you can look at some of these (variants) like variant like these, then you can introduce a NOT gate in feedback to enhance the number of states that are represented. So, say you think about for example, for this year if you run through this time like you as the clock ticks, what are the states that will be represented by this Johnson counter so to say when we start at the state 1000.

So, now this final output is going to get inverted, and it is going to come in the front. So, that is how it will look like 1100 as a second state like that you will be able to kind of see all this state

you can with this kind of thing you can see that we are able to now represent more number of states as a sequence in this counter.

(Refer Slide Time: 31:12)

Four Bit Johnson Counter

- Timing diagram: notice the sequence 1000, 1100, 1110, 1111, 0111, 0011, 0001, 0000 at each clock cycle

Clock
Out₁
Out₂
Out₃
Out₄
Time

NPTEL
PRASANNA S GANDHI gandhi@me.iitb.ac.in

The slide features a green background with a white gear icon containing a lamp. A timing diagram shows a square wave for 'Clock' and four digital signals for 'Out₁', 'Out₂', 'Out₃', and 'Out₄'. The signals change at each clock edge. A small inset video shows a man with glasses and a mustache.

**3 bit up counter:
Design Procedure**

- This all is fine ! But we want counter to behave the way we count numbers 0, 1, 2, so on..
- Q: how we can we develop such counter?
- First thing: Develop state diagram about what you want to do and represent states with binary numbers
- Can you do that!

NPTEL
PRASANNA S GANDHI gandhi@me.iitb.ac.in

The slide features a green background with a white gear icon containing a lamp. A small inset video shows the same man as in the previous slide.

Now, the question is, suppose we want to count 3 bit for example, but we want to count in a way that, we want the output to be binary 01234 like that up to 8 or 7 rather than the three bit up counter 1234 if you count 01234 eight numbers you will count. So these are binary numbers. So how do we develop such kind of a counter? So, again, we use the principle of state here. So, this is how like, this sequential logic design needs, like the definition of states.

And once you define states, then like, the problem becomes very easy. That is expressed now as a combinational logic problem, and then we will see how we use down D flip flops to get a circuit out of that combinational logic table. So first thing here to develop a state diagram. So you do that pause here, do the state diagram development that , if I am set at state 000 then what would be my next step, next state? So my next state will be 001 if I am at that state, what is the next state like that I developed the state diagram.

(Refer Slide Time: 32:32)

3 bit up counter: Design Procedure

- Observe the state diagram:
 - there are no external condition for transition from one state to other
 - No external inputs
- Can you next develop truth table with previous and next states?

NPTEL

PRASANNA S GANDHI gandhi@me.iitb.ac.in

3 bit up counter: Design Procedure

- Represent number by CBA
- What would be entries in truth table
 - Inputs: Current values of C B A
 - Outputs: Next values C⁺, B⁺, A⁺ respectively
- Truth table?

NPTEL

PRASANNA S GANDHI gandhi@me.iitb.ac.in

And you can see the simple state diagram you come up. So, there are no external conditions for transition here. You remember in our log problem, there are the conditions on the keys that will take us from one state to another state, but here there are no such kind of conditions.

So, there are no external inputs. So, again we can develop in the previous, the truth table in the similar way, where we do not have any other inputs, we are going from just one state to other state without any kind of a condition on any other kind of key or any other input. So this say number, let us say, I say this number is C for somebody C, second is B and third one is A now what should be the truth table entries, you can develop that.

(Refer Slide Time: 33:22)

3 bit up counter: Design Procedure

3-bit Up-counter

Q: Once you lay down various possible values of inputs, how will you generate Outputs?

PRASAN

Truth table:

	Present State			Next State			
	C	B	A	C+	B+	A+	
0	0	0	0				1
1	0	0	1				2
2	0	1	0				3
3	0	1	1				4
4	1	0	0				
5	1	0	1				
6	1	1	0				
7	1	1	1				



3 bit up counter: Design Procedure



3-bit Up-counter

■ Truth table:

	Present State			Next State			
	C	B	A	C+	B+	A+	
0	0	0	0	0	0	1	1
1	0	0	1	0	1	0	2
2	0	1	0	0	1	1	3
3	0	1	1	1	0	0	4
4	1	0	0	1	0		
5	1	0	1	1	1		
6	1	1	0	1	1		
7	1	1	1	0	0		

Q: Once you lay down various possible values of inputs, how will you generate Outputs?



PRASAN

So see now we have this C, B and A are the present state, then the next state should be what? If the present state C is 000 the next state should be what, think about that. So present state is 0 0 0, next state I am going to is 001. So I will write 001 here. So, these are like the next state C plus B plus and A plus our next states.

So we do not have any other outputs here. In the previous case, we had output of lock open or close, we do not have that kind of output here. So, we just have these three outputs and then we can complete this truth table. So, you complete it yourself first and then only like look at what is done, it will open up some things for you some mistakes small kind of nitty gritty, that will get corrected.

(Refer Slide Time: 34:24)

3 bit up counter: Design Procedure

3-bit Up-counter

■ Truth table:

	Present State			Next State		
	C	B	A	C+	B+	A+
0	0	0	0	0	0	1
1	0	0	1	0	1	0
2	0	1	0	0	1	1
3	0	1	1	1	0	0
4	1	0	0	1	0	0
5	1	0	1	1	1	1
6	1	1	0	1	1	1
7	1	1	1	0	0	0

3 bit up counter

Recall Template for K map →

C+

		C			
		00	01	11	10
A	0	0	0	1	1
A	1	0	1	0	1

B+

		C			
		00	01	11	10
A	0	0	1	1	0
A	1	1	0	0	1

A+

		C			
		00	01	11	10
A	0	1	1	1	1
A	1	0	0	0	0

■ K map: Develop expressions

	Present State			Next State		
	C	B	A	C+	B+	A+
0	0	0	0	0	0	1
1	0	0	1	0	1	0
2	0	1	0	0	1	1
3	0	1	1	1	0	0
4	1	0	0	1	0	0
5	1	0	1	1	1	1
6	1	1	0	1	1	1
7	1	1	1	0	0	0

Now, we have generated then we can use K map or like other kind of whatever techniques you want to use for getting like expression of our C plus as a combination of A B C. So, I want to express C plus as a combination of A B C because these are my next states, these are my outputs, and these are my previous states.

So, I would express them and these previous, this next state are only dependent upon previous state, no other inputs they are dependent upon, that is how I, so, I will express C plus as a binary or Boolean expression of A B C by using Karnaugh maps or by using sum of products rules or some kind of a way that you have done in the past.

(Refer Slide Time: 35:20)

3 bit up counter

■ K map: Develop expressions

$$\begin{aligned} C^+ &= CA' + CB' + C'BA \\ &= C(A' + B') + C'(AB) \\ &= C(AB)' + C'(AB) \\ C^+ &= C \oplus (AB) \end{aligned}$$


NPTEL PRASANNA S GANDHI gandhi@mc.itb.ac.in

3 bit up counter

■ K map: Develop expressions

$$\begin{aligned} B^+ &= BA' + B'A \\ &= A \oplus B \end{aligned}$$

NPTEL PRASANNA S GANDHI gandhi@mc.itb.ac.in



3 bit up counter

- K map: Develop expressions

$$A^+ = A'$$


$$= A'.1 + A.0$$

$$= A \oplus 1$$

Converted to have Expression with XOR Gate only!!
Since other expression Had XOR!!


		C			
		00	01	11	10
A+	A	0	1	1	1
	1	0	0	0	0
		B			

PRASANNA S GANDHI, gandhi@mc.itb.ac.in



So, this is more like a development of Karnaugh maps and I am not going to get into how they are like written there, but we get finally this expression by using this K maps. You can pause here and go through if you want to revise your Karnaugh map and get these expressions like that you will develop it for B plus also so B plus is expressed as some kind of function or Boolean expression of ABC so this, it happens that this is not dependent upon C here. And the same way for A plus.

(Refer Slide Time: 36:05)



3 bit up counter

- With all expressions available as below can you now develop circuit for implementation?


$$C^+ = C \oplus (AB)$$

$$B^+ = A \oplus B$$

$$A^+ = A' = A \oplus 1$$

You can put Inverter here
As well (its same thing)

In market usually same XOR chip will have multiple XOR gates. The implementation may reduce number of chips on PCB



3 bit up counter

- Recall the way to implement previous and current state using D Flip Flop

Current state Available here

A+ D Q A

Clock

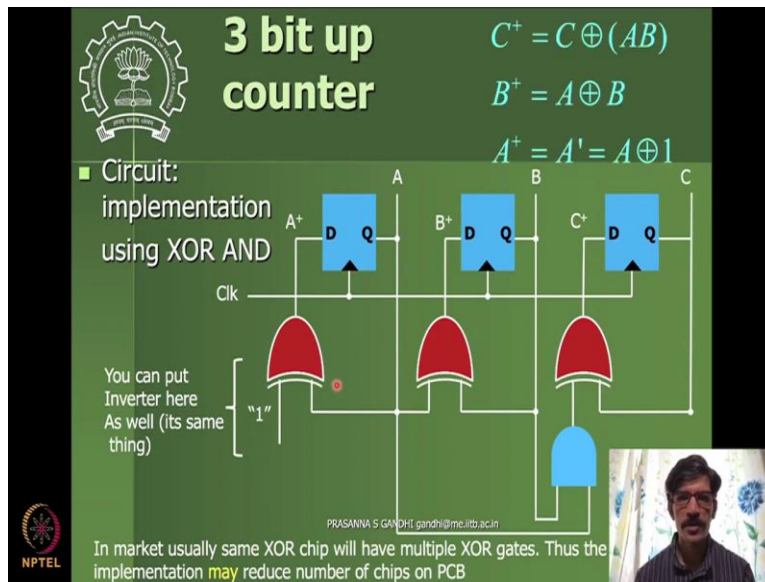
NPTEL

So, now, the question is that these are the expressions for C plus B plus and A plus and we want to now develop the circuit for implementing such a kind of a expressions by using now flip flops or memory elements. So, see now, the new state C plus is depend upon like is expressed as previous state some kind of a combination. So, how will you develop such a thing. So, this is what we want, the next state we want to be coming as a function of like current states.

So, current states are available as, so, we use D flip flop for this purpose. So, how do we use that is explained here. So, the way we implement the previous and current state using D flip flop, so, say some input A is current state is available here and I found the state after one clock cycle to be given as here. So, I use now depending upon the variables I have are three states I need to define. So, I need to use three D flip flops, one is for A one is for B and other is for C.

So, these are three D flip flops I will use and for these D flip flops at any point of time with the current state is available at output here and I use that state and then produce my A plus as a as per these relationships that I have produced already.

(Refer Slide Time: 37:52)



So, if I do that, then these circuits comes up. And then this becomes like my circuit for 3 bit up counter which will be counting exactly the way I want binary numbers to kind of get counted 001 000 001 010 like that it will come up. So, this is how these circuits are hard to developed, and in the same way, our problem of the key lock, the lock problem that we discussed can be also kind of converted into circuit now.

So what we need to remember is like you know the state, so, we need to define further state some kind of a binary representation, previous state and new state like that, it will come say, if you, if your state is A B, are two numbers that are defined, then like, you will have A plus B plus coming up, I mean, when the previous state and new state are expressed in your truth table. So you can do that exercises as a homework. So this is how we go about developing these counters or sequential logic circuits.

(Refer Slide Time: 39:12)

Building Memory

- Suppose we need to now construct memory using our D flip flops, how do we go about? What all we need.
- Lets start with putting 1 flip flop for storing one bit. Lets say for simplicity our one 'cell' or 'register' consist of 3 bit number
- Thus 3 flip flops for each cell would be needed and can be represented as the following

Register

Notice clock D Flipflops is

NPTEL

PRASHANT'S GANDHI

Now we want to build memory. So we will talk about a little bit about memory. So memory we will use now say 3 bit memory, we need to remember three numbers. So we need the like this 3 D flip flops to be combined together. Now, how do we kind of put the value in the memory and how do we get value out of memory?

So that is what is more nitty gritty to talk about. So we use this clock, which is going to the D flip flop judiciously to see when I want to store my value in the memory, when I want to get value out of the memory, how this is done, that is what we will see in the next class.

And then we will see some of the putting together now all these elements how we can kind of built some primitive microprocessor to get users the feel of how things work in general. So maybe we will stop at this point here and continue in the next class.