Digital VLSI System Design

Prof. S. Srinivasan

Department of Electrical Engineering

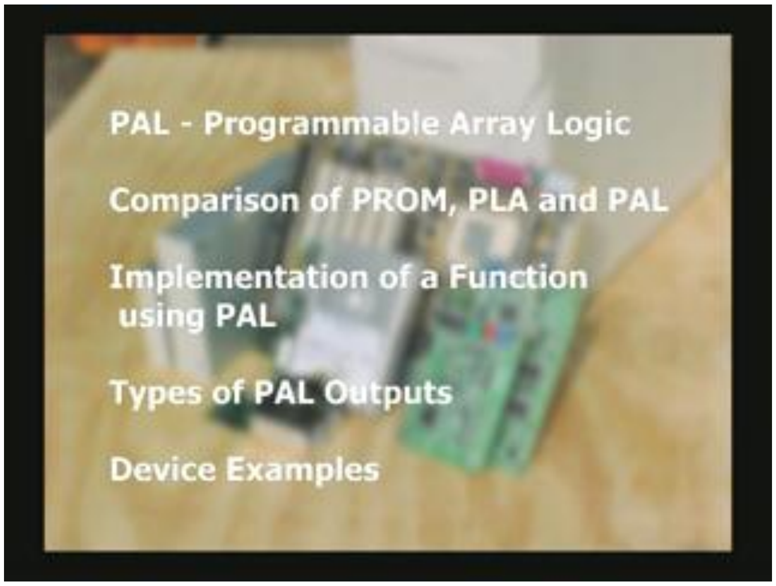
Indian Institute of Technology, Madras

Lecture - 5

Review of Flip – Flops

Slide – Summary of contents covered in the previous lecture.

(Refer Slide Time: 01:40)



PAL - Programmable Array Logic

Comparison of PROM, PLA and PAL

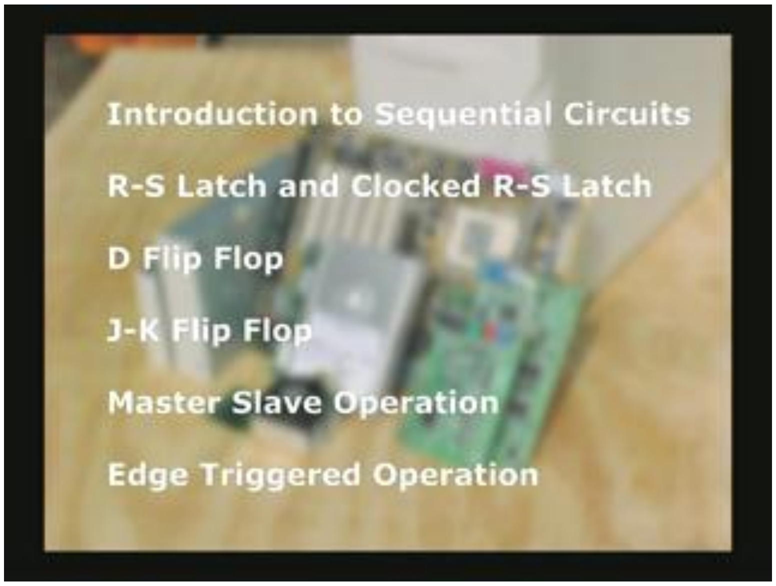
Implementation of a Function using PAL

Types of PAL Outputs

Device Examples

Slide – Summary of contents covered in this lecture.

(Refer Slide Time: 01:50)



Introduction to Sequential Circuits

R-S Latch and Clocked R-S Latch

D Flip Flop

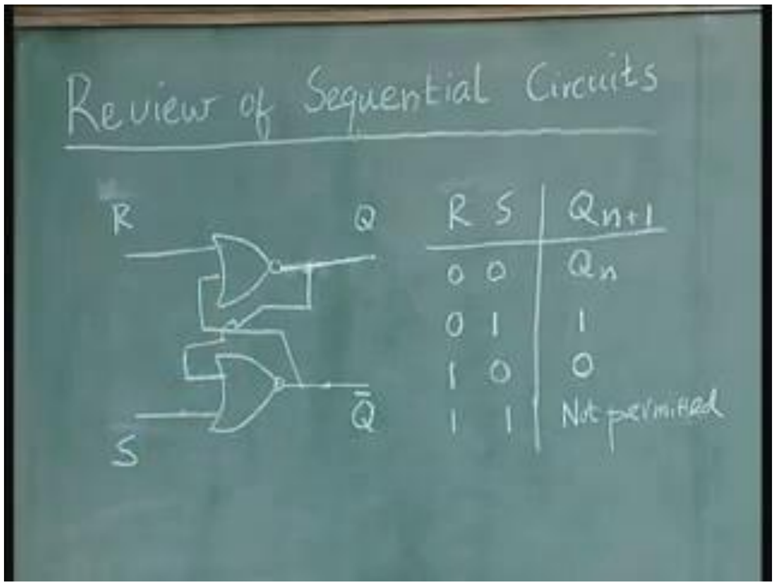
J-K Flip Flop

Master Slave Operation

Edge Triggered Operation

In the last few lectures, we have been looking at combinational logic. In particular, we took combinational logic which could be implemented using self devices such as multiplexers and decoders. Now very few digital circuits or digital systems are purely combinational. Most of the circuits, systems and digital are a combination of combinational sequential but it never says it is a combination of combinational sequential. Whenever the sequential element is involved, you call it a sequential circuit, even though there will be a combinational logic part of it. Let me not go in to the details of the combinational sequence logic differences and all that, which you would have seen in the first course. I can only say that no system can be designed without sequential block, let me say, no practical system can be designed without sequential elements. In this first lecture on the sequential circuits, I am going to introduce you to the building blocks which are already familiar to you, just as a review.

(Refer Slide Time: 03:45)


Review of Sequential Circuits

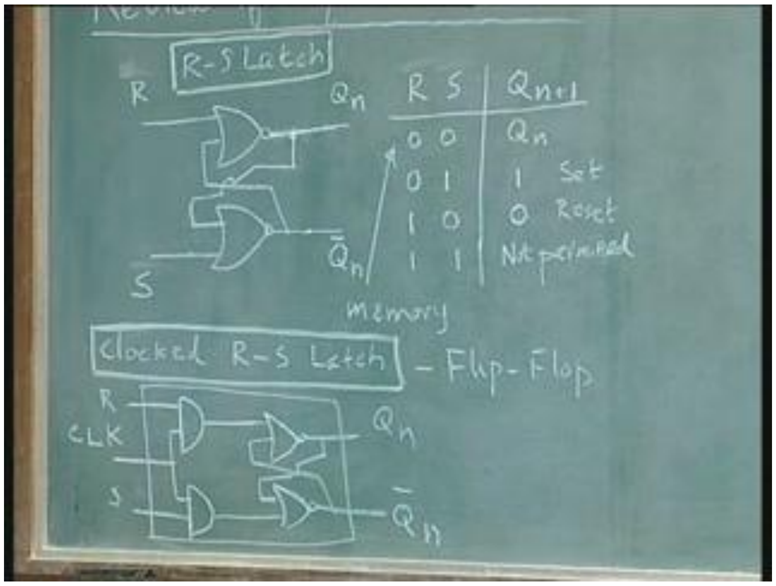
R S	Q_{n+1}
0 0	Q_n
0 1	1
1 0	0
1 1	Not permitted

All of you know that the most elementary building block of a sequence circuit is a latch called S R latch or R S latch depending on how you name it. It is 2 NOR gates connected back to back. You also have an equivalent version using 2 NAND gates connected back to back. All of you must be familiar with this table; the inputs are called R and S meaning Reset and Set. When you put 1 in the reset the output Q becomes 0; when you put 1 in the set S output becomes 1; and when you remove the inputs R and S and make them 0 0 outputs will remain what it was earlier, that is, if you want to make the output Q1 and Q bar as 0 of course, Q and Q bar are always complementary; you make S is equal to1 and R is equal to 0. That means you set the flip-flop by making S is equal to 1 and R is equal to 0 with the latch and then that become SQ is equal to 1 and Q bar is equal to 0. If you want to put a 0 on the output Q is called resetting operation; if you want to put a 1 in R-the reset input and 0 in the set input and then this become 0, this becomes 1 automatically.

Either way, whether it is a 1 0 or a 0 1, when the input is removed that means you make the input 0 0 either after 0 1 input or1 0 input, the output remains what it was before removing that input. That state is called memory state also called Q_n. So, now we are talking of before setting and resetting if the output as Q_n after applying 0 0 it still remain Q_n that was it means by Q_{n+1}. This state is also called as memory state, which is the basic feature of the characteristic of a sequential circuit a memory unit. This is the reset operation as I said (Refer Slide Time: 06:00); 1 1 is not applied because, after a 1 1 input when you give one of the 3 other inputs, the output is not determined. The output will be determined based on which of these 2 gates is faster. Since it does not have that information and also because we do not want this unreliability in the circuit, we never apply 11 as input to the R S latch.

When the input changes, the output changes but we would like to have the input changes and output controlled by another event called a clock. When you have a clocked latch we introduce an extra input, addition to S and R, certain reset inputs; (Refer Slide Time: 06:50) this is R S latch.

(Refer Slide Time: 07:14)



R	S	Q_{n+1}
0	0	Q_n
0	1	1 Set
1	0	0 Reset
1	1	Not permitted

When you have a clocked R S latch, we introduce extra input called clock input. The function of the clocked input is to make sure the changes to the output occur only when the clock is active or high. The input may change but the output will not change if the clock is low; output will change when input changes only if the clock is high. Such a latch is called clocked latch or a flip-flop. What you do? You put an extra gate on the input to control the clocking; this is my R S inputs. Both of them are controlled by an

AND gate whose other input, that is, the second input is called a clock, that is all (Refer Slide Time: 08:15).

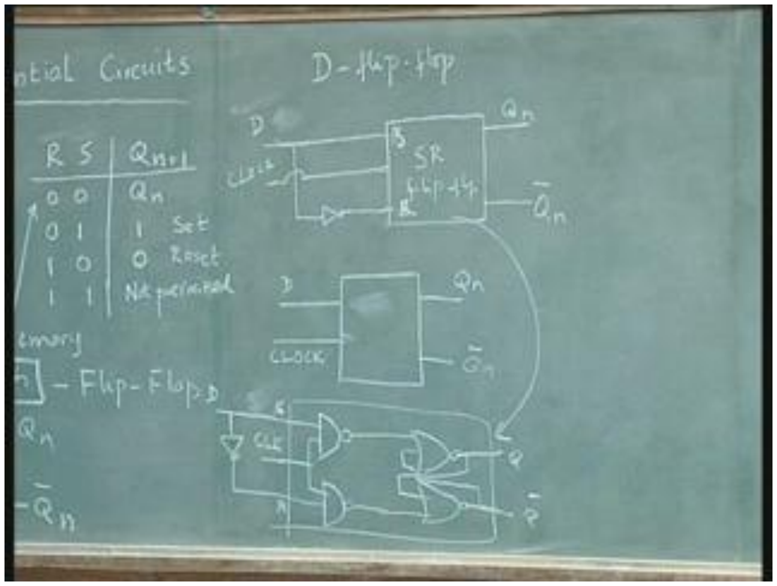
If you put an AND gate with an extra input connected to the clock, naturally, the output R or S will pass through this gate only if clock is1. When the clock is 0 the output remains 0 you know that. In AND gate when 2 inputs one of the input is 0 the output always 0 which means whenever the clock is 0, this is 0 0 and 0 0 corresponds to memory state with no change the output. There is no change in the output because inputs are 0 0. When inputs are 0 0 it does not make any change in the output.

On the other hand, if the clock is 1 whatever is there in R will pass through this gate, whatever in S will pass through this gate (Refer Slide Time: 08:49). Then the circuit operation will be determined by either a 0 1 or1 0 combination as we wanted. So in this case, this will be the additional…. we call it Q_{n+1}, so we will call it Q_n, n is suffix, say, in the n^{th} input, n+1 is the next input. So with a clocked operation there are 3 inputs and 2 outputs for this Flip-flop or a clocked latch.

Now, a modification of this Flip-flop is a D Flip-flop. Since we are only interested in either setting or resetting the Flip-flop and keeping it there we can do it to one of the inputs. For example, if you want to put 0 in the A, the output of the Flip-flop we need to make R is equal to1, S is equal to 0 and make clock high; then the output becomes Q_n is equal to 0, and Q_n bar is1. If you remove the clock output remains 0. Likewise, if we want to store a 1 1, I mean you want to store a 1 in the Flip-flop you make S is equal to1, R is equal to 0; clock it output changes to1; remove the clock and output continues to be 1.

Now, in order to make a 1 I need put a 1 here and 0 here; in order to get a 0 I will put a 1 here and 0 here; two inputs need not be done. It always complementary so why do not we do away with one is the inputs? We have only one input that is called D latch or D Flip-flop.

(Refer Slide Time: 11:00)



We call it a D Flip-flop because once you have introduced a clock concept we can continue to call all the other circuits as a Flip-flops; you will always have to mark a clock input. T his clocked Flip-flop would be same as the clocked S R Flip-flop with R and S.

So this input is called a D input. The D input, if D is1, D bar is this (Refer Slide Time: 12:00). So this is 1, this is 0, you get Q is equal to this is 1, this is 0, Q_n will be 0, this is 1 and this is 0, Q_n will be 1, this is a D latch. You do not want to show all this extra gates, you put a D here; I think, I made a little, out of the convention; I will tell you what it is (Refer Slide Time: 12:53). You go back to this R S latch with the clock which is called R S Flip-flop; R is 1, S is 0 so the output is 0; R is 0 S is 1 gives output is1. This is the clocked latch. Now when you want to have 1 stored, you like to have 1. Now by just connecting this input through an inverter; when this is 1, this become 0, that means I will have a 0 stored here; when this becomes 0 this becomes 1 and 1 get stored. So I will store the complement of what I give as input.

It is really not a D latch in the sense, D gets through R the output is if 0, the output is 1; if R is 1 output is 0. We do not like to call the D latch. Strictly speaking I want to call it as D bar. How can we solve that problem? By putting an inverter at the input here, that is, instead of changing this AND gate with a NAND gate, if I change this AND gate with a

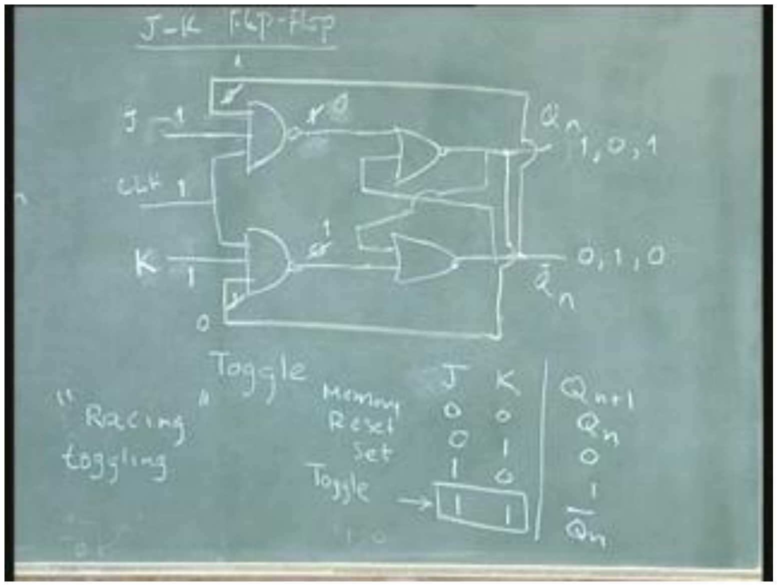
NAND gate what will happen is this R and S roles get reversed. Then if I put an inverter here (Refer Slide Time: 14:30), now I am going to do this. I am going to leave this circuit alone because I do not want to make too many changes in the original circuit. I want to show you what will happen. I am going to leave the circuit alone; you had in your notes I am going to make the change here. I am going to call this S and I am going to call this R, which means when D is1, Q is 1; when D is 0, Q is 0 (Refer Slide Time: 14:58). That will be more like data storing; 0 is stored or a 1 is stored. If you want to store 1 here put 1 here, clock it and 1 get stored; remove the clock and 1 continues to get stored (15:00 min). Similarly, if we want to store the 0, put a 0 and clock it, 0 get stored; remove the clock 0 continues to reside inside. So this is the D Flip-flop what you got earlier with an inverter of a simple R S Flip-flop would have been a Dbar Flip-flop would have what a D bar Flip-flop. Normally, it is not used, that is why it is not a mistake but it is a convention.

In the text book you will find D Flip-flop not D-bar Flip-flops. That is why I thought I will change it myself and call this S and R. That means the S and R Flip-flop is different from this Flip-flop, R S Flip-flop, where AND gate will be changed into a NAND gate. This S R Flip-flop that I have in mind here is this (Refer Slide Time: 16:10). This is S R Flip-flop and now if make this S and R as inverter, which I call D; it is a clocked D Flip-flop.

You are also familiar with the next Flip-flop called J-K Flip-flop. The problem with the S R Flip-flop is that one of the condition in this case, in an R S Flip-flop 1 1 or S R Flip-flop, whatever; 1 1 condition is not allowed because that I said least to the undefined operation, afterwards; 1 1 is not a problem. First of all, 1 1 will not give you a complementary output; Q and Q bar should be complementary in a Flip-flop. That will not happen if we put 1 1. The second problem is once you put a 1 1 and when you remove 1 1 and put some other input the operation is not defined. The operation is defined based on which of the two gates is the faster, which I do not want; I do not want an unreliable operation on a Flip-flop. Because of these things, we want to remove this possibility of this 1 1. Always remember not to give this input when something is restriction constraint which a circuit design does not want. You want to put some operation for this 1 1, which is why a J-K Flip-flop came about; we will see how it is done before that; we will see how the J-K Flip-flop is connected.

Now take this Flip-flop, (Refer Slide Time: 18:25) the S R Flip-flop with a NAND input; and Q AND Q bar or Q Q bar does not matter.

(Refer Slide Time: 18:40)



This is my S R clock. What I am going to do is, take the output of this Q_n and connect it to this input $Q_n$bar; and take Q_n and connect it to this input (Refer Slide Time: 19:50). That means I am making these 2 input NAND gates as 3 input NAND gates with an extra input derived from the feedback or the output of the Flip-flop. The condition is that S and R is equal to 1, which is not permitted in the previous case. We will see how that is eliminated in this case of J K Flip-flop. For that we need to make S is equal to 1 or R is equal to 1. Of course, we will analyse the circuit when clock is 1 because when clock is 0 input is not going to be effective in the sense that, the Flip-flop is going to store whatever value was put in it before the clock become 0. That means, we are going to analyse the condition 1 1 for these 2 inputs. Let us assume the condition of the output being 1 because, you have to start somewhere. Q_n1 and $Q_n$bar 0; when we set this back Q_n and $Q_n$bar, this is going to be become a 0, this is going to become 1. So this is 1, 1 1 and output is 0, 3 input NAND gate. 0 1 1 output is the 1 for a 3 input NAND gate. This condition is going to be the condition by which because of the inverse of the NAND gate becomes the reset condition. Earlier, we saw these 2 inputs NOR gate connected back to back as an R S Flip-flop, remember that? R S latch that means R is equal to 1; S is equal to 0 made a reset condition. That means it is going to become 0 output and 1 output. We have now made use of a condition of 1 1 from an unstable condition; I will not say unstable, but unpredictable condition. The condition which cannot be 1 1 input in a S R Flip-flop was not used because what happens further to that input is not clearly defined; that has been eliminated.

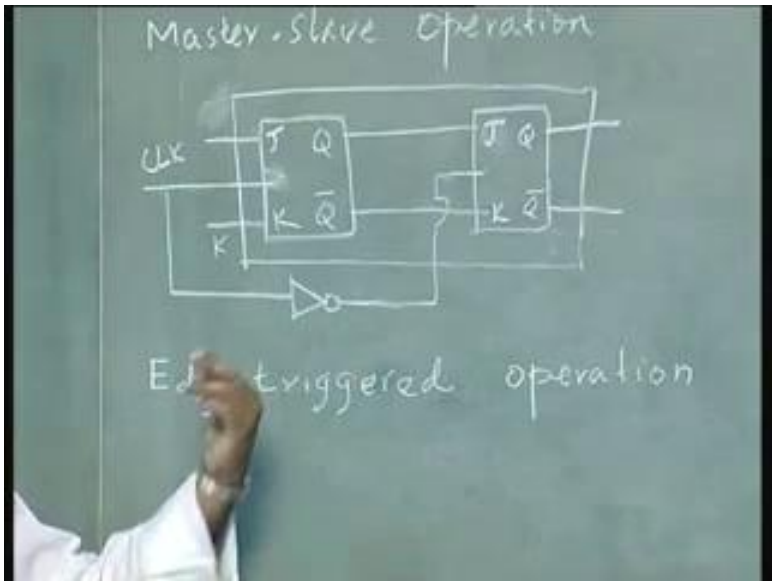
Now, 1 1 is the input, it will give you a change from 1 1 to 1 0 to 01. 1 1 earlier would have the 1 0 convert into 0 1. This condition is called toggling. Toggle stands for change of the output from 1 to 0 or 0 to1. But unfortunately, this is not going to be all that simple because, clock continuous to be 1, input continuous to be 1; input continues to be 1 but now we have changed the output from 1 0 to 0 1. Remember the feedback; so this feedback gets this 1 back into this and this feedback gets the 0 back into this. So instead of input being 1 0 because it is a 1 1 1, this input become 0, this becomes 1 and there is a reversal condition (Refer Slide Time: 23:00). Earlier with the 1 0 which results in the 01 and a reset condition have a 0 1 in the input. The R S latch uses a 1 0 R S set condition, this keeps going. When the clock is high and S is equal to1 R is equal to 1, these outputs keep feeding back to the input; the output keeps toggling. This is not what we want. We want an R S condition where both are 1 at a defined useful output. A defined useful output is alright that means it toggles with whatever the original output. If you write the truth table, I am going to introduce for the first time. Let us look at J and K; the 3 input NAND gate is an expanded version of the S R Flip-flop now called J K Flip-flops. We will call this J K instead of S R (Refer Slide Time: 24:20).

J K Flip-flop works like an S R Flip-flop for the 3 input 0 0, 0 1, 1 0 with a condition 11 which is not permitted in the S R Flip-flop, we have now established that the output changes. If the output is 0 here and 1 here it will become 1 here and 0 here or the output as 1 here and 0 here will become 0 here and 1 here, that means the output toggles for the condition of 11; we will call it $Q_n$bar. That means, whatever as a Q_n, the complementary will be the output. This condition is what I said the toggle condition. Toggle condition is alright as this is a memory condition; this is a reset condition. Toggle condition is the desirable condition as R S is equal to 1; R is equal to S is equal to 1 is not permitted in the S R Flip-flop. But the desirability does not stop there because, the constant feedback of Q_n and $Q_n$bar back on the input NAND gates keep toggling the output as long as the clock is high. Once the clock becomes 0, the activity stops because, once the clock is 0 the output 1; one of the inputs is 0, this becomes 1 and then 1 1. When it is 1 1 output corresponding to the R S Flip-flop 1 1, it means memory condition will return. (Refer Slide Time: 26:00). Now, this condition is called a Racing condition, constant changing or toggling of the output of the J K Flip-flop or any Flip-flop for that matter.

As long as the input clock is high, the output keeps toggling all the time. What is the time of toggle? The time of toggle is decided by the propagation delay of the gates. How much time it takes for the input to propagate from here to here to here and back here (Refer Slide Time: 26:46), that is the toggle time. At the end of this period it goes back. This toggle time is determined by the delay of the 2 gates; it will be very small. Especially in the modern gates, the delays are very small of nanoseconds order; whereas, we never have a clock with less than that. If you had a clock with less than the propagation delay of this Flip-flop then toggle will not happen. Because, suppose by the time the input reaches the output and changes the output, if I remove the clock then toggle will not happen; toggle condition requires clock to be high.

So, if we put a clock to high, put a set of inputs and let the output change and by the time output propagates back into the input if the clock is already removed, toggling cannot happen. That means, I need to operate a very high clock which is not possible; it is also not required. So, I should have some other way of controlling the toggle. That means, I need this desirable condition of 1 1 as a toggle condition which is an improvement from an S R Flip-flop and I do not like this racing where the outputs keep changing. The only way to avoid it is to make a clock which is very small with the clock frequency is to be extremely high which is also not required. Why should I operate my Flip-flop at that rate? What are the other ways in which I can remove this racing condition? There are two simple ways again. It is a review, you should have heard about these things earlier.

(Refer Slide Time: 28:44)



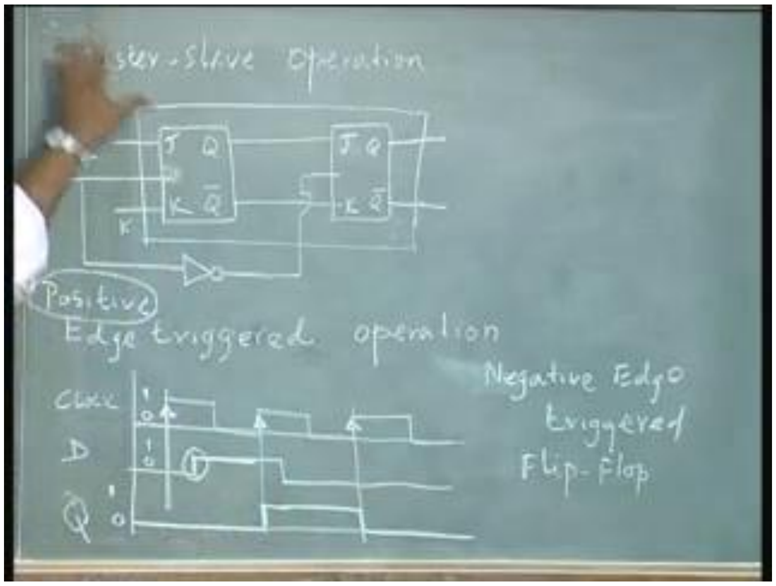
One is called master-slave operation; I will not go into the details of this. In a master-slave operation what happens is we have two identical Flip-flops, J K, S R or whatever you want to call it (Refer Slide Time: 29:32). Only difference is, I feed this Flip-flop with a clock and this is the clock which is inverted from this clock; that means, the clocks are inverted. One of the clocks is positive and the other clock is negative and vice versa, I mean positive negative, 1 and 0. So the clock is 1 here, 0 here; or a high here, low here; low here, high here (Refer Slide Time: 30:00). That means, even the input can change and toggle; it cannot get this input passed on to this. So the toggling is because of this overall inside. I do not have the access to these inputs. The feedback is overall; this input gets feedback from here and this input gets feedback from here. The 3 input NAND gate of this J, 3 input NAND gate of this K are getting feedback output from this Q and this Q bar. Even though I put a clock on the J K condition 11, output changes but this Flip-flop will not take through because the clock is low (Refer Slide Time: 30:55). So even though this is feedback, the feedback will not change as long as the clock is low. The moment the clock is high this feedback condition will reach here, the change in condition that is, the output will change.

By the time output changes here, when the clock is high here the clock is low here, the clock will be removed. We can operate any desired clock rate because these two Flip-flops are isolated in the operation. When this is effective, this is not effective; this is effective, this is not effective. So I am practically taking two steps. Any change in the input will change the output as long as this clock is high. At that time this Flip-flop will not have any change on the output because the clock is here low. When the clock is low the Flip-flop output does not change. When the clock high period is over for this Flip-flop, that will be put here; this Flip-flop becomes clock high. Any output at that time will be input to the Flip-flop and that will change. But no more change is possible in this Flip-flop because clock is already is removed. This way, we can eliminate the racing condition. It is called master-slave operation and requires an extra hardware, of course. An extra thing you need to do means, you have to pay something and that payment comes in terms of the extra Flip-flop. That way I do not have to worry about my clock width. I want it to be very narrow in order to avoid the racing as is required in a non-master-slave. But one objection to this should be that we have to use 2 hard blocks to get the same operation. 2 Flip-flops are required in order to get a master-slave operation and that is on problem.

The second problem would be the overall. Supposing I want to go a more way very high operation of the clock. This is an extremely unlikely case in today’s technology. In today’s technology we are talking about gates of nanoseconds or sub nanosecond delays. So when the delay of the gate is less than1 nanoseconds, a couple of the gates in this path and in this path of this total of 4 or 5 gates, 4 or 5 nanoseconds, it corresponds to very high clock rates. 10 nanoseconds corresponds to 100 mega hertz, 5 nanoseconds corresponds to 200 mega hertz operation. It is very unlikely to accept that; we are talking of today’s technology Pentium 3, Pentium 4, about giga hertz operation.

Whatever it is, I do not want to question your need. Suppose you want to operate a clock rate faster than the propagation delay, the 2 Flip-flops can allow you to do. That means should I be restricted by the speed of operation, by having to pass through two Flip-flops hardware? That means, why I should unnecessarily increase the propagation delay if it is not required? Plus, the fact that extra hardware is required, makes you think whether it is the best solution. Of course, this is one solution as I said. The other solution is Edge triggering; this also you are familiar with - edge triggered Flip-flops.

(Refer Slide Time: 34:24)



Another way of removing this is the same. As the condition, I do not want to change the input to propagate more than once during the clock period; that is the basic theory. That we did here by preventing the clock to the second Flip-flop. Another way of doing it is, I will let any change in the inputs change the output, only when the clock changes from 0 to 1 or 1 to 0. In between, even though the clock is high any change in input will not affect the output. This is called edge triggering operation, that means, what I am trying to say is if this is my clock and this my D input (Refer Slide Time: 35:35), let us say with the simple D Flip-flop I am talking about, let us say this is my D input, this is my Q output of the D Flip-flop - the simple D Flip-flop. Only when the positive clock edge or the positive transition of the clock from 0 to1, output will change.

Any change to the output will happen based on the input; output of the Flip-flop will change based on the input that is sure. But when will it happen? Only when the clock transition occurs from 0 to 1; rest of the time even though the input may change the Flip-flop will not recogonise the change in input. It will recognize the change in the input of the Flip-flop only when the clock transfer occurs from 0 to 1. This is called positive edge triggering. Why positive? Because it is going from 0 to1; clock edge going from 0 to 1is called positive edge. Let us assume the Flip-flop output was 0, even though at this point what will happen is it will look up at every transition of the clock the Flip-flop circuit will look at the input; input continues to be 0; so nothing happens. D input has changed from 0 to 1 at this point. Will this affect the output? No, it will not because, it is not occurring at the positive edge (Refer Slide Time: 37:15). It has already passed before they become 1. This change has to wait for the next positive edge until this positive edge which is the second positive edge. This input in D will not be recognized. At this point time again, the Flip-flop input will be looked at which D is equal to1; the output will be Q is equal to 1 for D Flip-flop; so this will become 1 (Refer Slide Time: 37:45). This Q has become 1 from 0; as soon as this edge disappears, a little time afterwards D disappears, D becomes 0 but this is not going to affect the output because output has already become 1; it will not be looked at all here; it will be looked at only when the third positive edge occurs.

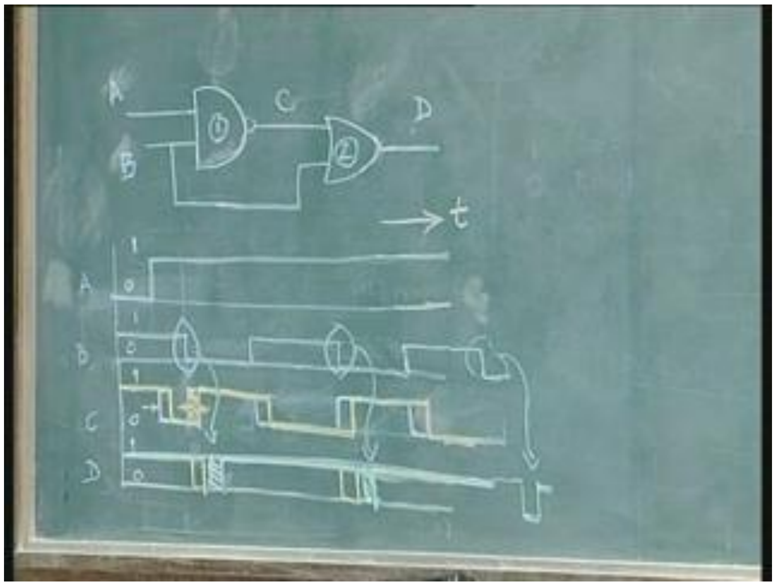
When that happens, look at the D again, D output will go back to 0. This is called positive edge operation. This is one way of preventing aliasing. Suppose you make a Flip-flop, whatever the hardware or logic inside, my inputs will be recognised or looked at only during the clock transitions, either 0 to 1. In this case, it is called positive edge trigger; or from 1 to 0 in which case it will be called negative edge trigger. Only at that time, output will change that way I can eliminate the possibility of a racing. I do not have to worry about two Flip-flops having to put in the circuit; this can occur at any speed.

Speed of operation is not a limitation in this case because I am going to have a single Flip-flop. Of course, the speed of limitation still depends on the propagation delay of the Flip-flop because there has to be hardware for any Flip-flop in couple of gates and some other extra logic, we will talk about little later on. Extra logic is also required. There is going to be propagation delay at least I do not have it double it. Here I am purposely doubling it; doubling the propagation delay I am avoiding, that is all. That does mean the propagation delay is 0; no hardware can have the propagation delay 0. There will be propagation delay but that is the only limitation of the speed and I have also filtered the possibility of output changing any time; it will happen only when clock changes. So it is called positive edge triggers. Similarly, you can draw a waveform for negative edge trigger. I am not going to talk about it here. You can also think of the negative edge trigger.

So, these are the two options: the master-slave option, where you require an extra Flip-flop; because of that, extra hardware and little slowing down the maximum frequency of operation, I will not worry about it too much. The maximum frequency of operation today is so high and most of the common circuit we have to design in digital circuits, except in very high performance circuits. This is not a problem.

On the other hand, we have edge triggered operation, where we will allow the output to change based on the input only once in clock period. It can be a positive edge of the clock or a negative edge of the clock; one case is called positive edge triggered Flip-flop; other case is called negative edge triggered Flip-flop. When doing so, we still have to worry of the speed of operation by deserving the propagation delay but, you do not have to worry about the toggling condition and all that racing condition.

(Refer Slide Time: 41:05)



It is said the edge of the clock determines the operation; only during the clock edge any input will be recognized. It is easy to say but how to implement it because you are going to have a Flip-flop of a JK type here.

We have inputs occurring and the clock occurring; you cannot suddenly remove some or one of them and say it is only for a short duration. So the actual way in which it is done is to put an extra circuit at the input of each of these Flip-flops. Each Flip-flop input that means a JK Flip-flop, 2 inputs, for each of the inputs you need to put the extra circuit so there will be an extra circuit. That has be a part of the input of each of the S. Let us just understand the circuit; this is one of the combination possibilities. This is not only ray in which the edge is triggering; this is one possibility.

Let us independently consider without Flip-flops or latches or RS or JK. We will independently consider the circuit, so simple gate NAND gate and OR gate. This NAND gate has 2 inputs: A and B, output is C. The C is one of the inputs of OR gate and other input as OR gate is same as the B, so C is A and B whole bar; D is C OR B.

Now, let us assume arbitrarily an input A; this is time axis, x-axis is the time axis for all the wave forms; inputs go from 0 to1 somewhere here and keeps as 1. B waveform remains 0 or remains 1 for a while, goes to 0 for a fixed duration, goes back to1, comes back to 0 for fix duration. Now the output C is the output of the NAND gates when A and B of the inputs is given by the S. C would be 1 and this is 1; when both are 1 the C will be 0. When both inputs are 1, it is going to be the last for this duration again (Refer Slide Time: 43:40). Since one of them is going to become 1, one of them becomes 0, this is going to become 1; again 1 1 combination results in 0 for this duration; back to 1 because this becomes 0 and so forth it continues. This output at C based on the simple NAND gate operation.

Now, I am giving D as the input to the output C or B, actually. What I meant was, output C and B of the 2 inputs of the OR gate D, so we have this. D would be based on B and C. We will see that all the time, I am looking at the white lines not these yellow lines which we draw later on, and will be explained later on; this is 1, this is always 1. This has become 0, this is still 1 and this is become 0, this is 1. So either of B or C is always 1, which means D will be 1 continuously.

Why I am explaining this I will tell you in a minute, but this is an idle operation. No gate is idle. What will happen is each of these gates is, C and D has a propagation delay. Let us assume a small delay of this duration. Even though A and B are applied A has been continuously there so there is no question of its change is the B; B changes from 0 to1 or 1 to 0 from etc; C will not immediately change even if there is a input change condition.

As far as C is concerned, it is a condition of the inputs. From A is equal to 0 and B is equal to 1 to, A is equal to1, B is equal to 1, that change will not immediately reflect the C output. It will reflect after a short time; that is why I use this colored chalk here in yellow. A short duration is because of the propagation delay of the gate C. Likewise when it goes back from 0 to 1, this same propagation delay will act so the propagation of delay of gate 1 will shift to the output C from the original point here and here to this point and this point.

The actual output would be practical output would be this yellow line for each cycle. Again there is a change here so this will not come here as repeated earlier it will come little later and again go back to 0 a little later. So the yellow line is a modified output line at C because of the propagation delay of gate 1. Now making OR gates inputs as this and this, the new line yellow there be a short durations during which they will be both 0s, for example this duration here is a duration in which this is B 0 and C 0; B and C are both 0 here so the output has to be 0. Originally we said D was 1, continuously. Now, I am saying because the propagation delay is 0. Again this 0 will not occur here, it will occur little shifted. If you want to be perfect about it, I can use another colour chalk and this will be really not happen here. All these exaggerated drawings are just to make you understand, so this would be the green line. If you call this green, some sort of light green, this is where output of D is; earlier the output is always 0.

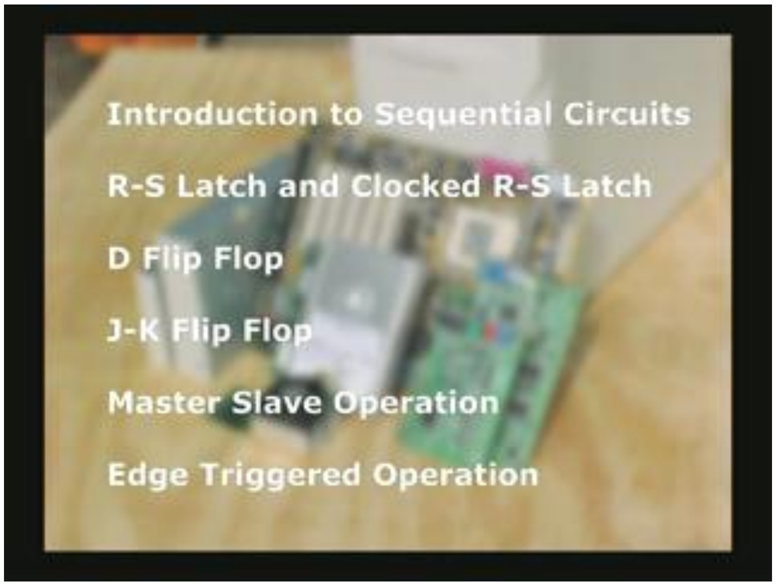
Now, I have a short duration. I mean earlier the D was constantly 1. Now I have a short duration of each output is 0. That is going to occur again here and again when this goes back to1 like that. If I want to give D as an input a Flip-flop, the Flip-flop input remains 0 for a short duration of a time. Now A can be the input, for example, for the Flip-flop, D Flip-flop may be, D could be the clock cycle and every time the clock goes from 1 to 0, this is negative edge. Every time the clock goes from 1 to 0 correspondingly, there is an output fall based on this input. Of course, it happens as a negative. I want a D input; if this is D and the input will be applied; so clock is high. Even though the clock is high the D does not remain 1 all the time; of course, D is 0 in this case. I will do some inversion. One of these gates has been inverted that is one of the concepts so I too invert one of these gates, may be this NAND gate can be AND gate. To make inversion this becomes a positive pulse and negative pulse so what I want is corresponding to the input which is constantly 1.

I want input to be narrow pulses every time the clock changes from positive to negative that means negative edge. Every negative edge, the output becomes a small pulse. I can change this into a positive pulse by changing the configuration of the gates. I can make this pulse into positive edge triggered by again making changes in the circuits. Changes in the circuit can lead to a positive pulse and at the positive edge trigger. I can have the positive pulse at a positive edge, negative pulse at a positive edge, negative pulse at a negative edge and negative pulse at a positive edge. Whatever this combination I want, I can make all by different types of gates. We have NAND gates, NOR gates, AND gates, OR gates, the combination of these gates. The idea is used to delay the gate. The delay of the gate is used to in order to make the input even though it is always available; the input of the Flip-flop when it actually reaches the Flip-flop gates, NOR gates which does the flopping operation. The Flip-Flop operation is because of that, remember RS latch we started with, the 2 NOR gate inverter that should be the one to which the input is considered. All other things are only external peripheral circuitry. So to that input circuit will change very narrowly like this.

What happens basically in edge triggered Flip-flop is at each input of the Flip-flop here is simply the JK Flip-flop, 2 master-slave Flip-flop has 2 identical Flip-flops. But in the edge triggered Flip-flop, I said simply it is very easy to say there is 1 Flip-flop, so reduce the hardware. This speed of course is a speed of propagation I said now, that is not all quite true. I need to add the input to the input of the each of the edge triggered Flip-flop whether is a JK Flip-flop, SR Flip-flop or a D Flip-flop. Whatever the edge triggered Flip-flop I am constructing, each of the inputs of a Flip-flop other than the clock input I need to add a little edge triggering circuitry. This circuitry shapes the input into narrow pulses which only be applied to the Flip-flop and only at that condition the output will change; the rest of the output will not change because the input is removed after that narrow pulse period. These have an edge triggered operation and become an input to the sort of a preprocessing if you want to call it. The preprocessing circuit to each of the inputs of any Flip-flop, if you want to have it as an edge triggered Flip-flop operation. We will stop here for today.

Summary of lecture-5:

(Refer Slide Time: 53:37)



Introduction to Sequential Circuits

R-S Latch and Clocked R-S Latch

D Flip Flop

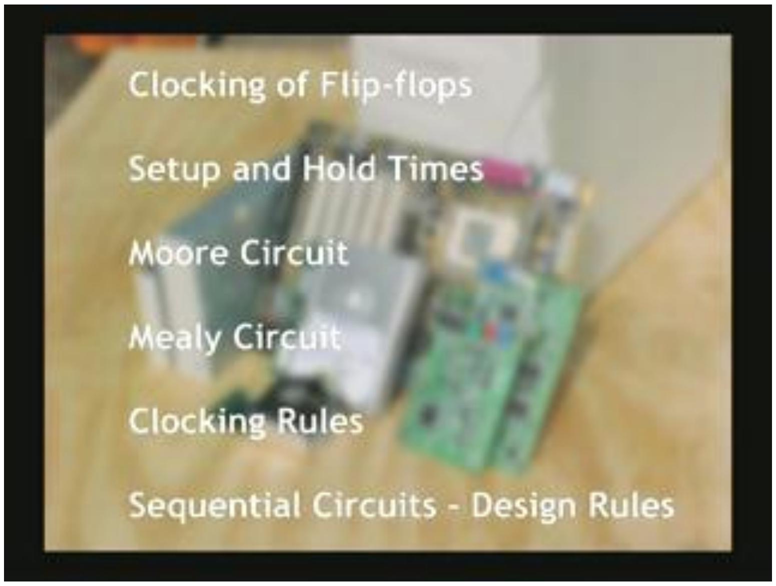
J-K Flip Flop

Master Slave Operation

Edge Triggered Operation

Sequential circuits:

(Refer Slide Time: 53:29)



Clocking of Flip-flops

Setup and Hold Times

Moore Circuit

Mealy Circuit

Clocking Rules

Sequential Circuits - Design Rules