Digital VLSI System Design

Prof. S. Srinivasan

Department of Electrical Engineering

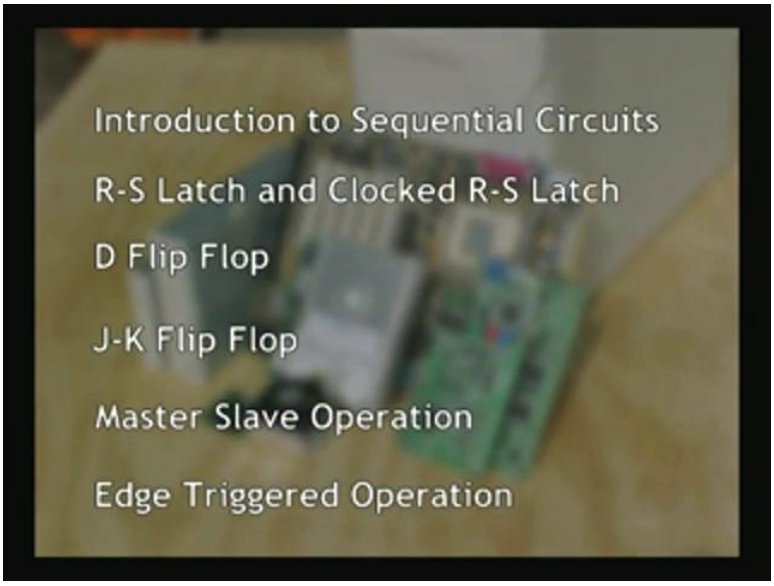
Indian Institute of Technology, Madras

Lecture - 6

Sequential Circuits

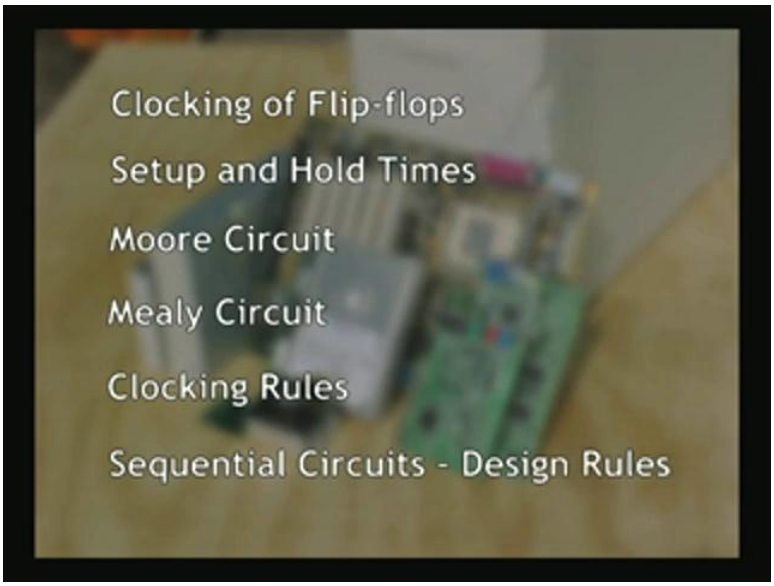
Slide – Summary of contents covered in the previous lecture.

(Refer Slide Time: 01:16)



Slide – Summary of contents covered in this lecture.

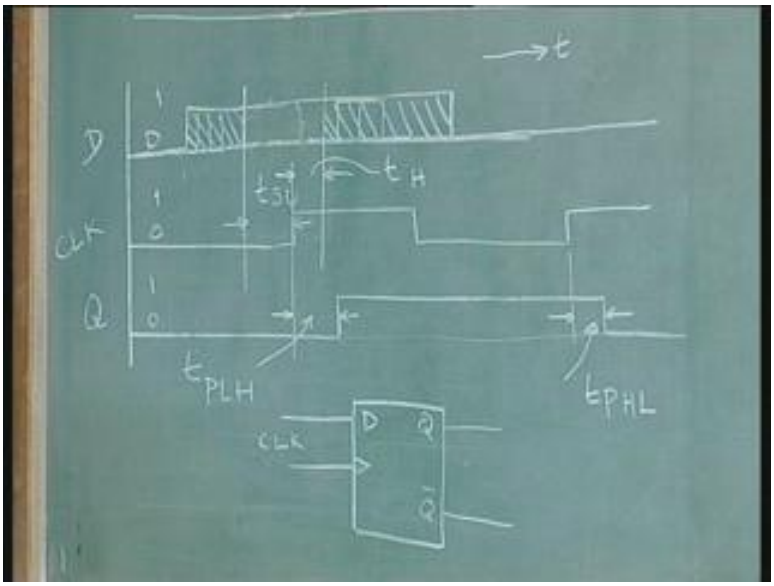
(Refer Slide Time: 01:38)



Today we will talk about some timing issues in clocking Flip-flop. In the last lecture, we saw the different types of Flip-flops and we also saw that the clock is an important component or an input signal of the Flip-flop. The Flip-flop changes state, output Flip-flop output changes state. With the clock it could be level triggered when the output changes; when the clock is 1 or edge triggered when the output changes, when clock edge transition occurs either from 0 to 1 or 1 to 0.

Whatever it is, the clock is a central signal around which the operation to Flip-flop is designed, operation Flip-flop takes place. We need to know how exactly it influences the performance of the Flip-flop. So in this lecture, we will talk about the timing issues. I call this lecture the clocking of the Flip-flop; that means, we will discuss all timing aspects connected to the Flip-flop operation.

(Refer Slide Time: 03:24)



Let us take an example of the simple D Flip-flop edge triggered. When you have the edge triggered Flip-flop, the clock is given a little arrow and this is the common practice (Refer Slide Time: 03:38). You may be familiar with this in your previous course and digital circuits. And if it a negative edge triggered we put an extra bubble in addition to the arrow head; output is Q. These arrow heads indicates a positive edge triggered Flip-flop. That means what happens is, the D signal input D which goes from 0 to 1 at this sense of time but there is no clock. The clock is 0, so the output does not change. Let us assume the initial output to be 0; the output does not change. When the clock goes from 0 to 1 assume this is a positive edge triggered Flip-flop, so the clock changes from 0 to 1. The output has to now change from 0 to 1 also because D is 1. Now this happens here. This should happen here (Refer Slide Time: 04:31) but it happens here usually a little later because of the propagation delay. This period is called t_{PLH}, propagation delay from low to high; the output has to change from low to high. The amount of time required for that propagation is called propagation delay t_{PHL}.

Likewise, when the Flip-flop clock has met 0 the output does not change because output condition is D and next time the positive edge occurs only when the input will be looked at. Then when it happens we see that the input is 0. D has already become 0 before the second clock edge at which time the input signal will be transmitted to the output. That

means, the output queue will go from 1 to 0 again not immediately but with a little delay called in this case t_{PHL}, propagation delay correspond to the variation transition from high to low.

Now, when you talk about the Flip-flop transitions generally for typical values you will take the average of these two. Some cases we will take the maximum of these two depending on the type of operation. Worst case delay is maximum of these two and typical value is the average of these two. This factor is important and that you should know before finding how fast, how frequently, you can clock the circuit. Before you can decide that we need to know these values. Another thing is that, this is purely the output propagation of the signal through the gates inside the Flip-flop to the output.

There are a couple of more timings we are to be concerned about, one is called setup time and the other is called the hold time. We will define these: when the clock transition occurs and we are already assuming that in this case. There is a D as 0 became 1 much before the clock transition occurred, so when the clock transition occurred this one was transmitted output of course with a propagation delay. What if D changes closer to clock transition? If D changing from 0 to 1 here suppose, it changes from here or here or here is there any restriction or condition which says how soon or how early should D change before the clock edge arrives in order for the clock edge to recognize this change and transmit it to the output?

Something like you want to board a bus; how early should you be in the bus stop or the bus terminal; before you are allowed to get in. You cannot just get into the bus when the bus is starting because you are not sure; you may run into an accident; you may fall from a foot board or because you do not know there is enough space inside. This of course is not a technical reason but technically, we will see how this.

What I am saying is you should know before any activity, we should be clear about the intension of the circuit. That means, the circuit intension is the input has to be change from 0 to 1 and that intension should be made very clear before the clock arrives and how early should happen that is called the setup time. Somewhere here this interval, we will call this t_{SU} setup. The setup time is the minimum time before the clock transition. The

input should be stable at the input of the Flip-flop; that is called the setup time. In this case of course, this is an extra time which has no meaning. I can afford to change my D here, here, here, (Refer Slide Time: 09:22) anywhere but not beyond this point. The problem is what will happen if you make it beyond this point? Suppose you let the D change here, can you positively say this input is not to be recognized? No. These are all typical values; it may or may not what we can say; it is not guaranteed. The transition being captured by the Flip-flop is not guaranteed but if you want a reliable operation Flip-flop you do not want output which is not guaranteed. Either you should say it is not changing or it is changing. Either way I am happy; if you say it is not changing fine I will try to do it next time; if it changes, fine I will use this now but if we say it may or may not change, that is a condition that we should avoid in any design because you do not want to any unreliable or unpredictable performance. So we have to avoid making D change occur here. We should never let this change anywhere here. We should make sure it happens before this. This is a condition.

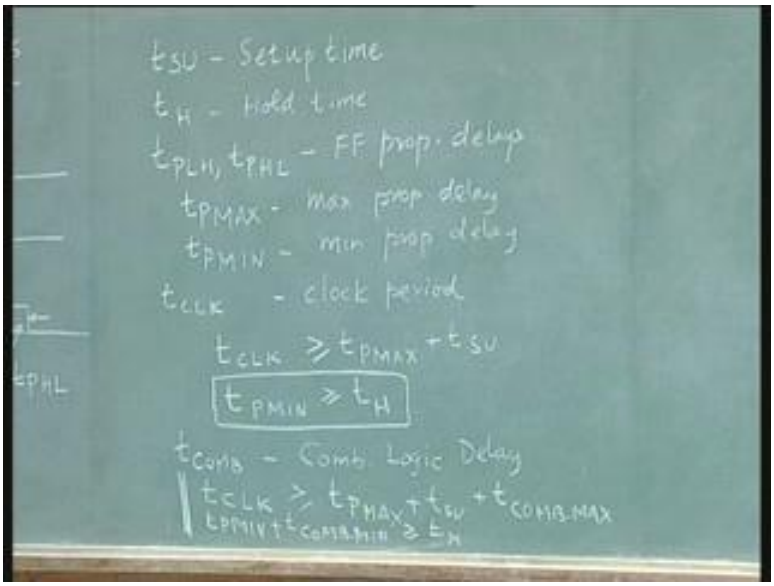
By the same argument, the signal to be captured by this, I said it has to be positive edge when the positive edge of the transition from 0 to 1 occurs clock; D should be high and not only it should be high at this point it should be high sufficiently before (Refer Slide Time: 11:03). Now the question is can you remove D as soon as the transition happens? Can I bring it back? Of course, in this case the D last so long for the long duration. But if I say is it alright if I remove D as soon as the clock transition occurs somewhere here. Again is it acceptable? No, there again there is a definition, there is a timing involved. There is a minimum time for which the output should be stable after the clock transition occurs for the output to be guaranteed again. This also guaranteed outputs something like a quality. If you do not do it nothing is going to happen. The world is not going to fall apart.

The point is you are not sure of the circuit performance. When we are not sure of the circuit performance, your output cannot be predicted properly. When your output cannot be predicted properly your system is unreliable. I do not want to design an unreliable system. That is why you got to make every point satisfy every aspect of the timing. So there is a minimum time for which the input should remain stable after the clock

transition occurs. That time I will put arbitrarily here (Refer Slide Time: 12:30), beyond which D can change any time. There is no need for D to extend up to this point; it can go here down, it can go here down, it can go here down and so forth but this period is something which you cannot eliminate. This period is called hold time (Refer Slide Time: 12:30). Now you have seen four timings; in fact, five timings. I did not explicitly say the fifth one being the clock frequency. We are all trying to do all these to find out what is the highest frequency at which I can operate the system if necessary; of course, I do not have to. When you are designing the traffic light signal in a major intersection of the roads you do not operate in giga hertz or mega hertz. You may operate so that there is enough time for the cars to flow. When you are operating a computer, you work in terms of hundreds of mega hertz and today in giga hertz. So we should know these things.

For example, some other things should not matter in a system you may design. You may be designing a traffic-like controller in which you are talking of milliseconds or even seconds as a minimum interval about which you are concerned. So where is the question of the microsecond delay in the signal occurring after the transition? Already it is a trivial issue. On the other hand, on designing a Pentium processor or any other microprocessor with 200 mega hertz clock speed, where we are talking of 5 nanoseconds clock period, these become comparable. Then you have to be extra careful in your design. So that is what I am trying to say, we have to know all these things. Five things are involved. One is the clock period- the maximum clock frequency which is implicitly stored; I have not mentioned it. I just assumed at clock wave form. Then there is this queue changing; there are two periods: one is called the transition propagation delay because of the transition propagation delay low to high; another one is called propagation delay because of high to low and the average of two is a typical value. The maximum is to use for worst case designs. And then we are talking about two other terms: one is called setup time- the period or the interval or the duration before the clock transition at which the input should be stable. Hold time is the period for which the input should be stable after the clock transition has occurred.

(Refer Slide Time: 15:25)



Let us write this here. t_{SU} - setup time, t_H - hold time, t_{PLH}, t_{PHL} are called Flip-flop propagation delays. Usually, we take the average of these two as typical value but I am going to define t_{PMAX} as the maximum propagation delay which is the larger of these two numbers or greater of these two numbers and t_{PMIN} as minimum propagation delay which is the smaller of these two numbers. Then, I will define the clock period t_{CLK} as the clock period.

With these definitions (Refer Slide Time: 17:10), let us write a couple of equations: t_{CLK} should be greater than or equal to t_{PMAX} plus t_{SU} setup time. Now you see naturally this t_{PMAX} need not come into the picture. What I am saying is the clock edge should occur, the setup time interval after the data change and hold time interval before the next data change. That means the data should change the t_{SU} before clock edge and should remain valid t_H after the clock edge. Naturally, we can get the frequency period in terms of setup time and hold time. But, deliberately I am introducing the t_{PMAX} because sometimes what happens when we use a Flip-flop in a feedback mode, the output of the Flip-flop fed back as input to the same Flip-flop or some other Flip-flop. If the propagation delay is very small because there is a maximum value, there can be a minimum value, typical value these values may be smaller. When that happens we do not want to change the input before the hold time has elapsed. I want to write one more equation which will say t_{PMIN}

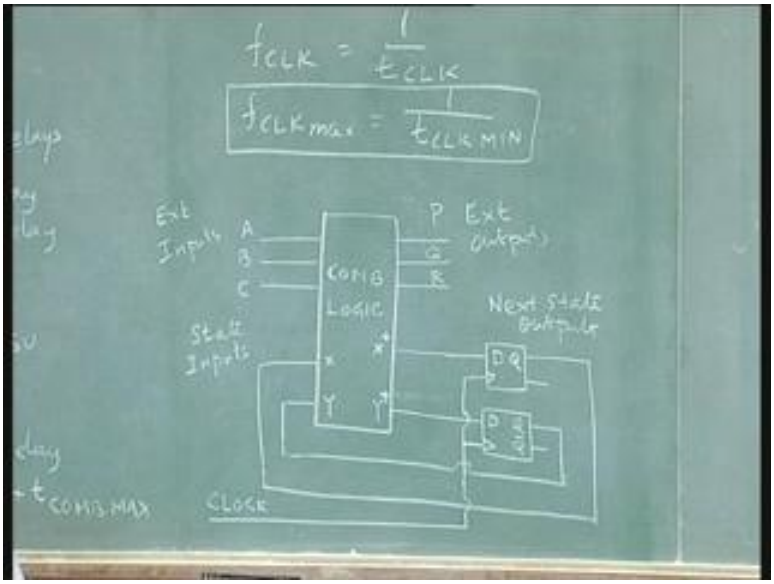
should be greater than or equal to t_H. What I mean is, if t_{PMIN} happens before t_H that means, its propagation delay is less than t_H, the output of the Flip-flop queue can go and change the input. That is what happens; the data should be valid; data validity is what we are looking at. The valid data at the input of the Flip-flop should not change for duration of t_H after the clock edge. But after clock edge t_{PMIN} is the minimum period at which the data is going to change and if this is immediately fed back as the input, the data would have changed before t_H has elapsed. That is why this additional condition (Refer Slide Time: 20:01) comes about; to make sure that the proper operation happens in case of feedback which is many times done. In sequential circuits, Flip-flop outputs are fed back as Flip-flop inputs for the next stage or the next clock cycle. Otherwise, what will happen is generally we will have to say the t_{CLK} should be…. substitute this condition; that is why I put t_{PMAX} here. Otherwise, we can simply say the clock period should be greater than t_{SU} setup plus t_H hold because of the extra thing we are putting this.

In practice though, the Flip-flop output does not get back into the Flip-flop input directly. There is usually a combinational logic in between. Let us say there is a combinational logic with this delay will define as t_{COMB} that should be factored into this. That means if there is going to be an additional time elapse in propagating the output back to the input because of the extra combinational logic which has to be passed or need not be the same Flip-flop. The Flip-flop of one Flip-flop can go to another Flip-flop through a combinational logic. So that is the time, t_{COMB} combinational logic. The equation will now get modified as t_{clock} should be greater than or equal to t_{PMAX} plus t_{SU} plus $t_{COMB.MAX}$. Again we will say maximum because combinational logic will not have all same delays so maximum delay of the combinational logic.

Likewise this equation also will get modified as t_{PMIN} plus that means now this is an extra restriction, that is, this restriction gets reduced little bit because I have extra combinational logic delay. What I said here was the propagation delay should be sufficient so that it does not reach the input in case of feed back before the hold time has expired. Now there is an additional delay; that delay also can be counted into this. So what I have to say is t_{PMIN} plus, in this case, minimum combination logic delay $t_{COMB.MIN}$ together should exceed t_H. These two equations are for the practical circuit in which

generally outputs or Flip-flops are fed into the next Flip-flop through a combinational logic; either to a same Flip-flop as a feedback output or a next Flip-flop as an input through the next stage. In either case there will be a combinational logic in between; the combinational logic delays are also taken into the curve, one case maximum value of that and the other case minimum value of that. In this case, it is just the Flip-flop alone. These are the limiting conditions for the proper Flip-flop operation. How does it help us?

(Refer Slide Time: 24:16)



We can find out what is the maximum frequency at which we can operate the circuit now t_{clock} is the period; the frequency of a clock is the inverse of clock period. The maximum frequency at which I can operate a Flip-flop will depend on the minimum period of the clock and that period determines the maximum frequency. So, $f_{CLK\ MAX}$ clock maximum will be now 1 by $t_{CLK\ MIN}$ minimum and this is the important equation in designing a system. Flip-flops have to operate at this speed (Refer Slide Time: 25:20).

You have seen the different types of Flip-flops and the clocking requirements like frequency of the clock and what is the maximum frequency at which you can operate the Flip-flop etc. Let us see how we use this Flip-flop in sequential circuits. As I said earlier, no circuit is entirely combinational; very few circuits can be fully combinational. That means, most of the digital circuits or digital systems design have both combinational

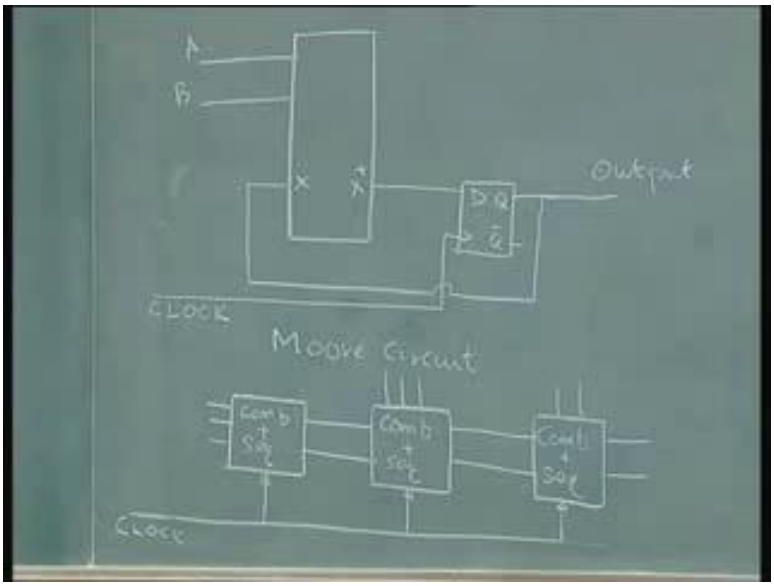
sequential components. They are purely sequential circuits; what is the purely sequential circuit? The circuit which uses only Flip-flops something like a register or a counter. Even there, you may have some resetting feature or some clearing feature and all that; but as I said there may be a pure sequential circuit, a very few combinational logic circuits. Most of the circuits are circuits which have both sequential and combinational elements in it even though they are called sequential circuits. Any circuit which sequences through various clock periods, various states and each state changes with clock period such circuits are called sequential circuits. We have a typical representation of a sequential circuit as a block which is called the combinational block really because as I said, this is actually a combinational logic, inputs to the combinational logic and then output to a combinational logic. Some of the outputs are required to be stored and used later on. You may have two types of output for a combinational logic: one is combinational outputs; the other is outputs which are to be stored and reused. These outputs go through Flip-flops because Flip-flop is a basic storage element and they need clocks. The simple model we will use a D Flip-flop here. It can have any number of any type of Flip-flop can be JK Flip-flop for example, in that case we need a J input and K input to be stored to get the next output.

In this case, we have D D. The advantage of the D Flip-flop is the input is same as output of the clock period. So whatever you want to store will be directly put here. In the case of J K, if you want to store something you need to find out the input combination J and K which has to generate the data 1 or 0 to be stored. So let us make this simpler by making this D Flip-flops clocked; we can have a single clock to clock both of them. Let us call this input A B C, this output P Q R (Refer Slide Time: 29:01). These Flip-flop outputs Q and this is a Q bar, which are normally required to be used again with another set of inputs, they are fed back. Let us call as X Y (Refer Slide Time: 29:55). This combinational logic has two set of inputs: one set of inputs are called combinational inputs - A B C are external inputs; another set of input in this case X and Y are called state inputs because they are internal inputs. These are the inputs generated to the previous clock period, stored and appear as inputs to the next, during the next clock period. These are called state variables. X and Y are called state variables which determine the state in which circuit is in and they will determine the next state X, next

state Y. So the present state of the variable X Y along the present inputs A B C, determine what should be the next state X and Y. In order to distinguish between these two, usually people put X plus Y plus indicating the next state. Given a set of inputs X Y, which are the state inputs for the given previous clock period and output of the previous clock period along the combinational inputs A B C to get X plus Y plus which are going to be the next state inputs or outputs as is the case may be. P Q R are the external outputs. These are called combinational outputs or external outputs; these are external inputs, these are external outputs, these were state inputs, these are next state outputs (Refer Slide Time: 32:08). Typically, if you give the present state inputs, the present state external inputs you get the present state external outputs at the next state outputs; the Flip-flops are used here. The whole thing becomes a sequential circuit which has a practical meaning with the A B C as inputs; may be in a traffic controller light situation; these can be different sensors or timing elements and it could be the state and circuit passes through and this can be different lights red, green, yellow lights which are required in the traffic intersection. These have to have a normal sequential circuit or any standard digital system configured or defined.

Combinational logic can be implemented as you know using gates or multiplexers or programmable logic elements as we saw in your lectures in combinational logic and this Flip-flop can be any type D or JK. As I said, D is simpler because the input becomes the output next time of the clock occurs. There is another variation of this input in this circuit which is some time seen. Sometimes the external outputs are not directly from combinational logic. They combine the state outputs along with the combinational inputs.

(Refer the Slide Time: 33:45)



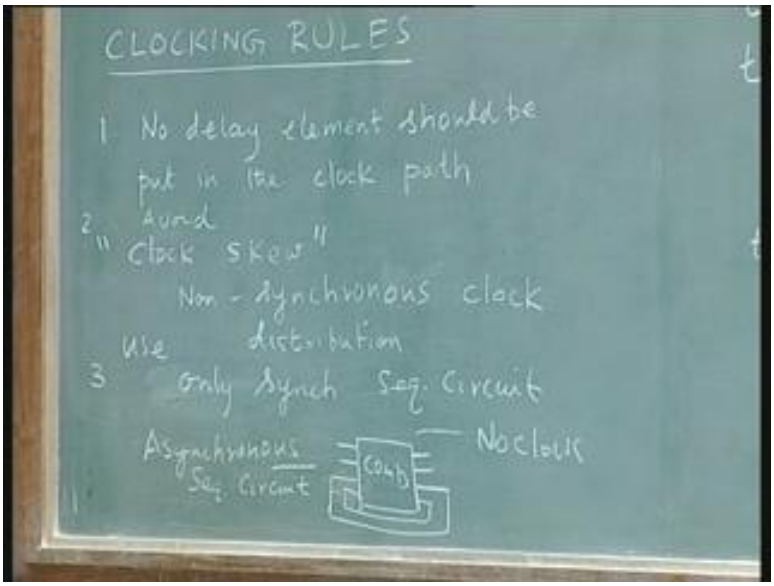
We will have a variation of these circuit in which we will have for a change, I will make it smaller and only one input. It is just an example what I am drawing so it does not need to be very complicated. Now these are directly getting the output (Refer Slide Time: 34:36). Here this combines them, the output combines the state variables here. We do not have these types of outputs here; these are directly used as output also. That means there is a variation in which the state input and the present input, decide set of external outputs and a set of next state outputs; that is one case. The other case, the combinational inputs and the present state input decide and the next state output which also determines the output of the circuits. This variation is called Moore circuit with outputs are tied to state. They are not tied in the sense of a physical tying or an electronic tying. Tied means outputs correspond to states; outputs are defined for the states. For a given state the output is defined; no output can occur independent of the state. Here there can be outputs which are based on the present state and the input variables and such a circuits is called Mealy circuits, in which we will have a set of present state inputs and a present input variables and combinational variables which determines not only the next state but also the current external output or combinational outputs. This circuit is called Mealy circuit. Both are possible in a system design. Here there will be output defined only for these states; here the output will be defined for states occurring for each state any input occurring at that time; that is defined. Whatever it is we are going to use these elements.

We have already seen enough combinational logic; we have seen Flip-flops; connecting them in this fashion realises the circuit. In order to do that, we need to go through a design procedure which we will see in subsequent lectures. I just thought I will give you these two variations, not only the basic architecture; that means, any digital system you are trying to design can be simplified or reduced to this format (Refer Slide Time: 37:33). What are the inputs, what are the outputs required externally, how many states are required, based on the state variables are defined and present state variables along the inputs designed in the next variable etc. This is the standard procedure that gives lots of design methodology which we will adapt in the subsequent lectures in order to design different types of digital systems.

Now, I want to go little out of this. We will come back to the design of the digital systems in the format using Flip-flops and combinational logic. When we use the same clock we can in this case, all the Flip-flops are clocked at the same time, only one stage a Flip-flop. Assuming the D's output goes to another similar circuit and we have to clock this also. We work stage by stage in a pipeline; there are several stages. Suppose your circuit has several stages of sequential circuits and all of these stages have Flip-flops and the output of one stage acts as input to the second stage; input to the second stage acts as input to the third stage and so forth. When the output of the first stage is decided, let me put it in as a sort of graphical way. Suppose my system has several modules or subsystems. This may be sequential circuits or as I said digital system containing both combinational and sequential; there is a clock which is connected to all of them. Of course, there are inputs, relays outputs, there may be external inputs and this is going to this, go back and then it goes back here if you want.

The problem is these clocks should all be synchronized; that is, it should occur at the same time. If I want to transfer, this output into input of this at the same time I should transfer this output of this into the input of this. So there must be perfect synchronism between all the stages. That can happen only when the clock edges that we know has no delay in the clock path. All the clock edges should be so perfectly synchronized. It should all be clocked at the same instance of time. That is one thing you need to know. You need to know a couple of these things before you can proceed in our design.

(Refer Slide Time: 41:08)



Let us state some basic rules, clocking rules. No delay element should be put into the clock path; that means; if I had a delay here (Refer Slide Time: 41:50) the instant at which this transition occurs is not same as this transition occurs. That means this output is delivered here but this input has not been transferred; that means it is going to be congested here. So overall this is a very simple statement. People define this as clock skew. Clock skew is a term which is used for non-uniform or non-synchronous distribution of clock. All the clocks should appear at the same time at all the modules; it does not happen. Clock skew is one of the reasons for non-synchronous clock distribution. Other thing is this delay may be purposeful or unequal matching of the Flip-flops and all that. Whatever it is, you can put a deliberate delay, you can avoid that; no delay elements should be there. So the second rule is to avoid clock skew which is part of the first role. Avoid clock skew. Then, to an extent, use only synchronized sequential circuit. What do you mean by this? That means a sequential circuit is defined as the circuit in which output is fed back along the input and we have the clock to determine when this new set of inputs are going to be recognised by that Flip-flop.

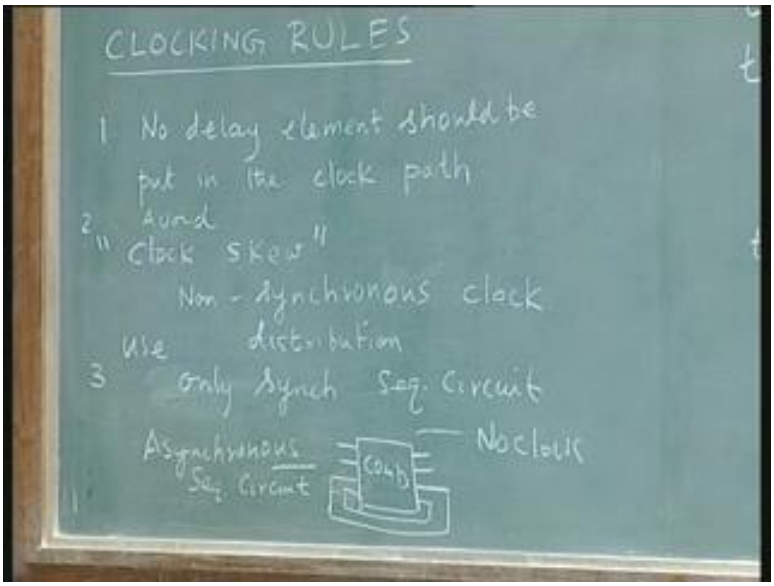
Now, suppose there is a no clock and just assume that this is a combinational logic in which there are some inputs and then some outputs. Some other outputs are fed back without any clock. This is a sequential circuit or not? By definition of sequential circuit

this is a sequential circuit. A sequence circuit is circuits in which outputs are determined by the present input and the previous outputs; the previous output is fed back into the gates which will again affect the present state, so this goes on. There is no control when these things will happen; it all depends on the path delays. The delay path of the combinational logic inside that will determine. So there is going to be a chaos inside this; especially when more and more inputs changes at the same time. The circuit behavior is not completely controlled by us; this is a problem. This problem can be avoided only by a putting a clock. If you have a clock the transition will occur at a given time. So, why cannot we always have the clock? Sometimes the clock slows down the operation. Suppose I want as fast as possible, this circuit is called asynchronous sequence circuit.

Asynchronous sequence circuit means a circuit is a combinational circuit, I mean a sequential circuit but there is no clock control. That means the inputs can change any time, output can change any time; based on the output change input can again change; that can again trigger another set of output changes, that can trigger another set of output changes. It goes on until the clock is stable until the circuit stabilises. Of course, there are some rules by which we can work with it in a satisfactory way. This is done in extreme cases where your speed of operation is so important or because you know the blocking strategy between you are designing two circuits which are totally independent of each other; whether there is no common clocking scheme your clocking scheme is different from the circuit which are going to interface; clocking scheme of that circuit is going to be different from your clocking circuit.

There may be several reasons why you may have to go for asynchronous sequential circuit. Of course, if it is the special case we have to go through this. I am not saying do not do this. Do it with extreme caution only under required conditions. Only when it is extremely important you go for a synchronous design. When you do it you have to carefully design; go through a set of rules which are independent, which are totally different from the set of rules we used in a clocking scheme. This is not in our scope of present lecture series. So, we will not talk about this but you should know. What I mean by this means, I just sort of haphazardly developed it; I am going to rewrite the same things.

(Refer Slide Time: 48:05)



I am not saying clocking rules but synchronous sequential circuits: make them synchronous as far as possible; 2. do not put any delay element in the clock path; 3. As I said, avoid clock skew. Clock skew can be part of this so the clock skew should be the result of these also, that is why we are putting it as a separate case. Even though this is contained in this, there may be other reasons of clock skew. Avoid it by proper design. What will we do next is of course, these are the rules we will follow in all our designs and what I am going to do next is to give you certain cases in which I cannot, as I said, we cannot afford to have this series of Flip-flops in a big circuit. This is going to consume lot of real estate and lot of power consumption. So is there any simpler scheme of getting Flip-flop implemented in IC? In my next lecture, we will talk about some Flip-flop implementation schemes adapted in integrated circuit design.

(Refer Slide Time: 50:49)

Sequential Circuit Design:

(Refer Slide Time: 51:15)