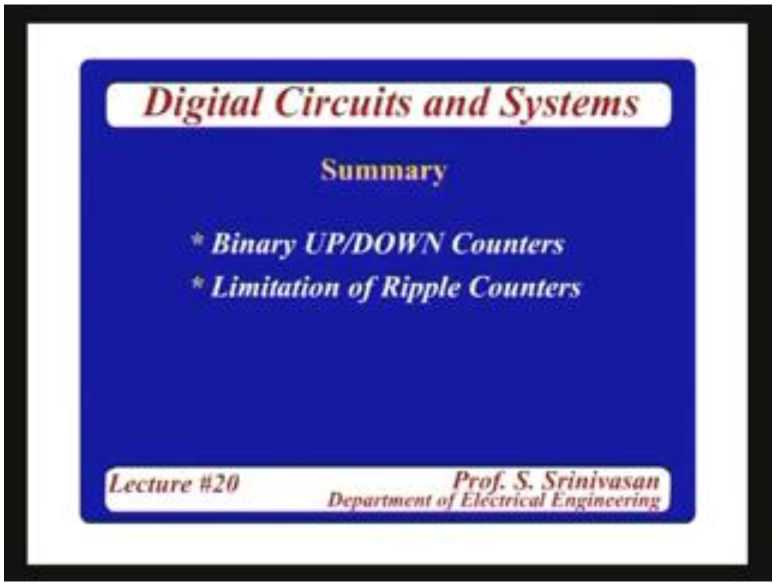
Digital Circuits and Systems

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

Lecture # 20

Up/Down Counters

(Refer Slide Time: 1:07)



Digital Circuits and Systems

Summary

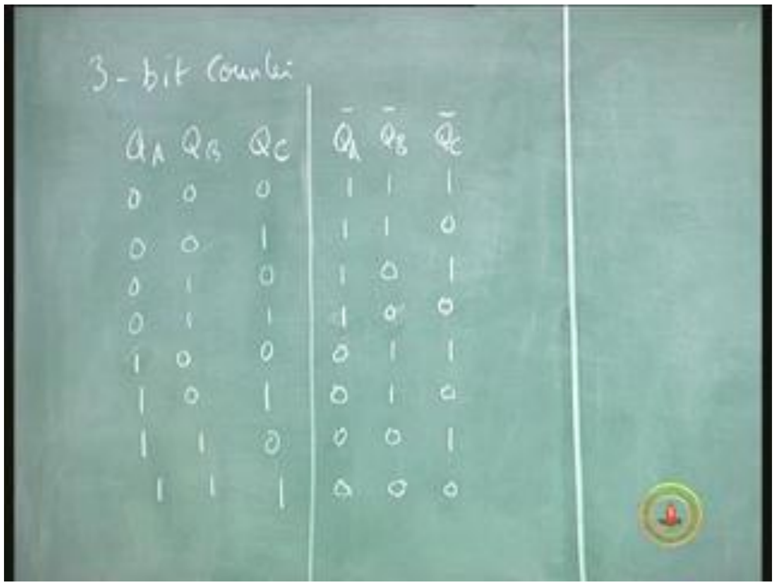
* *Binary UP/DOWN Counters*
* *Limitation of Ripple Counters*

Lecture #20 *Prof. S. Srinivasan* *Department of Electrical Engineering*

We were discussing the applications of flip-flops. In addition to being a storage element a flip-flop can also be used for division of the frequency by a factor of two each flip-flop divides the clock frequency by a factor of 2 and also can be used as a counter by stacking flip-flops serially by connecting series of flip-flops we can use it as a counter. We saw this in the last lecture. The counting depends on the number of counts or number of distinct states the counter has. We will depend on the number of flip-flops so there are n flip-flops the number of distinct states the counter can take is 2 power n so if we start the count with 0 it will go to 2 raised to n minus 1. You should also do a down counting. After all down counting and up counting is not very different because we know that Q and Q bar always complementary in any flip-flop. So when counting if you start the count with 0 0 0 0 let us say it is a three bit counter.

Let us assume these three flip-flops have the outputs A B C as QA QB and QC so the count starts from 0 0 0 then 0 0 1 and 0 1 0 (Refer Slide Time: 3:55). Now if you look at the complementary outputs this would be 1 1 1, this would be 1 1 0……… so all you have to do is to use the same counter whatever counter we had for up counting but take the flip-flop states from the Q bar.

(Refer Slide Time: 04:43)


3- bit Counter

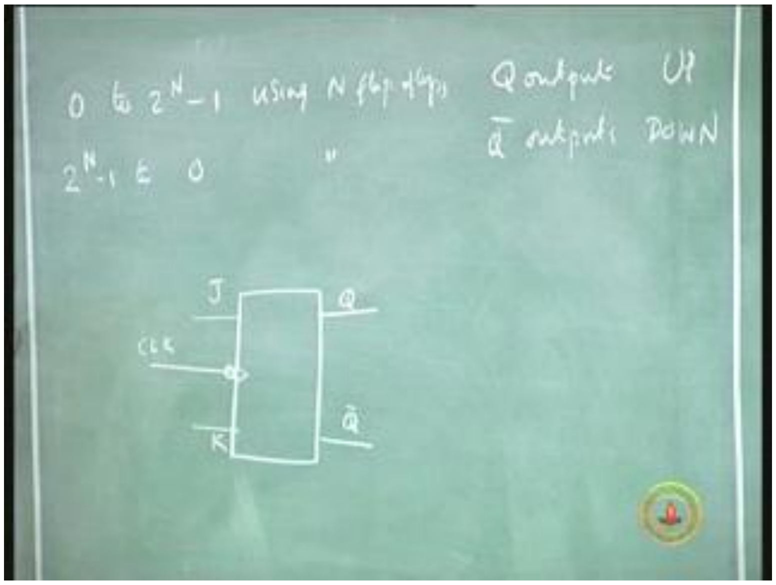
Q_A	Q_B	Q_C	$\bar{Q}_A$	$\bar{Q}_B$	$\bar{Q}_C$
0	0	0	1	1	1
0	0	1	1	1	0
0	1	0	1	0	1
0	1	1	1	0	0
1	0	0	0	1	1
1	0	1	0	1	0
1	1	0	0	0	1
1	1	1	0	0	0

So if you looked at Q bar instead of Q of the flip-flops you get a down counter. So I can have a 0 to 2 power n minus 1 counter using n flip-flops and 2 power n minus 1 to 0 counter also using the same n flip-flops depending on where I take the outputs from take the output from Q it becomes up counting (Q outputs up count) and Q bar output if you take it becomes down counting. So that is a very major use of a flip-flop, use flip-flops in a chain and for counting events.

Sometimes I may not want to count till the end of the last state that is possible within n bit counter we go from 0 to 2 n minus 1 and then go back to 0 after 2 n minus 1 count the flip-flop becomes 0 0 0 again. So I need to go through a full cycle the modulo as I said modulos the total number of states the counter has is called modulos. Suppose I want to terminate the count at a count which is less than maximum, if you want to terminate the count at the count which is less than maximum and go back to 0 I should be able to do it, example is the four bits we can have 0 to 15 but I may want to terminate with 9 that is 0 to 9 and start again with 0, there will be a decimal counter a decimal counter will come from 0 to 9 back to .Or for that matter I may want to start a count for a state which is greater than 0 I may want to start a counter for some reason 3 instead of 0 and then go out all the way up to 11 or 12 or whatever.

So I may be able to start at any count or end at any count within the possible range. With four bits you can only go from 0 to 2 n minus 1, with four bits we can only go from 0 to 2 power 4 minus 1 which is 0 to 15 and in n flip we can only go from 0 to 2 power n minus 1 and within this range I may want to start any count and I may want to end any count and start all over again. How do you do that? By terminating the count and starting it again at any point there is some extra mechanism required. In any flip-flop whether it is a D flip-flop or a JK flip-flop I am talking about, let us take the JK flip-flop because that is what we have been using for these counters, clock it as 0 and this as 0 so it becomes a negative edge triggered with an arrow and a bubble.

(Refer Slide Time: 8:30)



Q and Q bar is this and this is what we have been depicting as a flip-flop until now. Now I want to introduce two are inputs in this flip-flop. Of course there are two more inputs already which is the power supply and ground, no circuit will work without power supply and ground because these are active devices like transistors either Bipolar transistors or MOS transistors inside this.

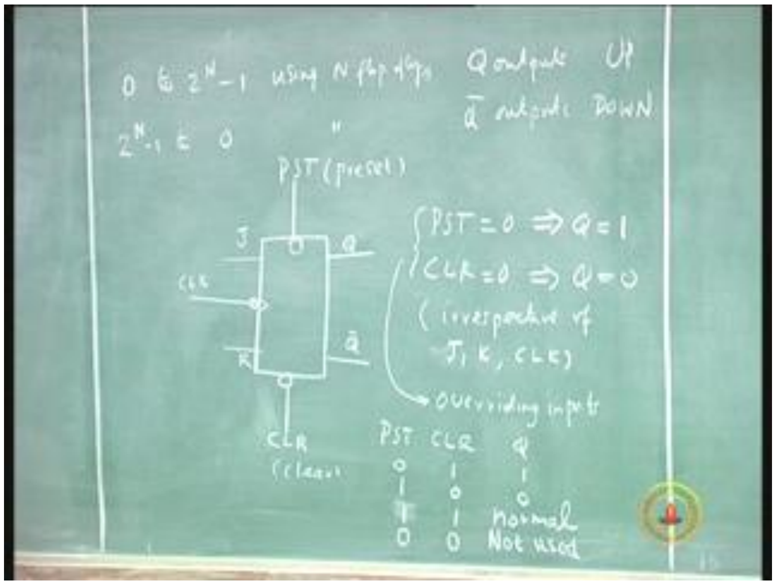
We have not gone into the circuit details but utmost we have gone into the gate level detail and not beyond that. Gates have active component like transistors and MOS devices. Let us not get into that but we need to power up those devices we need V_{cc} and ground or V_{dd} and ground depending on the nomenclature. Leaving those two inputs power supply and ground apart there are two more inputs defined for many of these flip-flops these are called preset and clear.

This input is called PST stands for preset and CLR stands for clear (Refer Slide Time: 9:52) clear input and preset input. What does a preset input do? As we see there is a bubble here that means this is an inverted input. An input which activates the circuit when it is low is called active low input. Such inputs we draw with a bubble. An input which would be affective when you make it 0 is called active low. An input which will be active or which will activate the required thing when you put 1 on it's called active high input. When you draw an input with the bubble it is called active low input. So the preset input when it becomes 0 what happens is it makes Q1 irrespective of the values of J and K and irrespective of the clock being positive, clock is 0 or 1. that means the preset input as called in electronic circuit parlance overrides all other inputs, PST is equal to 0 makes Q is equal to 1 irrespective of the value of J, K and clock at that instant of time and as long as preset is 0 output continues to be 1 and in between a clock may come and go, you may change J and K regardless of all that. Such an input is called an overriding input.

Likewise a clear input will make as the name suggests it clears the output means Q will become 0. These happen irrespective of JK and clock. So these two inputs are called as I said overriding inputs. So I need extra logic and to do that you can get into this little bit extra logic by regard to make sure that this has an overriding property of making the output 1 or 0 as the case may be. That means I can now write a table which is called preset clear table and preset clear Q, preset is 0, clear is 1, output is 1 and if preset is 1, clear is 0 and output is 0 and when preset is 1 and clear is 1 only then the flip-flop functions normally taking J and K as the inputs on the clock cycle. And at the clock cycle it goes past high or low all that will be determined and only when preset and clear are both made 1 they will allow influence on the normal functioning of the flip-flop. So the flip-flop acts normally as defined by the inputs and the clock.

When you say normal the behavior depends on the values of the input J and K and the clock. And if both are 0 which is predominant to the other because each of them is a pre-overriding input and when both are 0 then we won't know which one is overriding so one of those states is not permitted. Don't try to do both clear and preset to 0. That means there is uncertainty. And I always tell you the uncertainty should be avoided on all cost in any design. Make sure that preset and clear are not made 0 at the same time.

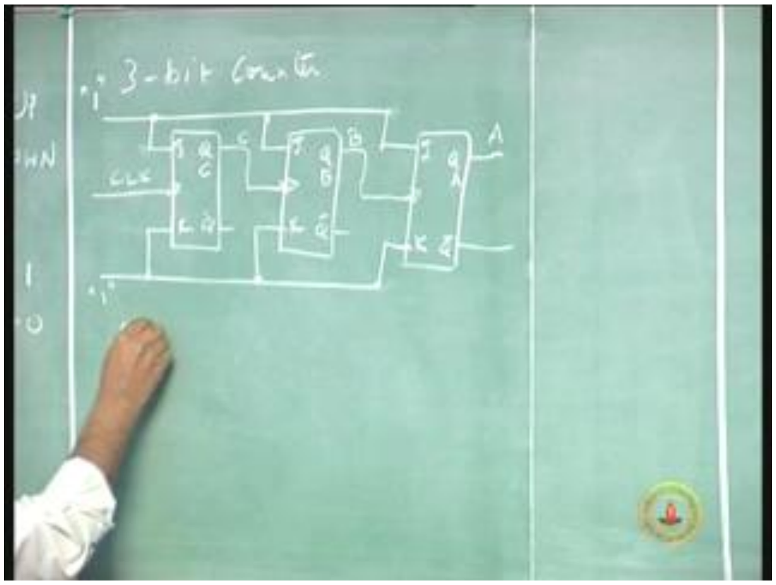
(Refer Slide Time: 15:35)



Now this property is going to help me to start or count anywhere I like and end my count anywhere I like. Suppose I go from 0 to 2 power n minus 1 and at some intermediate point I want to terminate and make it 0 0 0 0 I can do that by detecting that point and clearing all the flip-flops at that point. Similarly I want to start the count other than 0 I detect the point at which I want that to happen and use my preset inputs to get the flip-flops in different values. So use preset and clear inputs to get the flip-flops to different values other than the normal sequence that is where you get an arbitrarily counter, it is not going to start arbitrarily and end arbitrarily but of course defined by you and I didn't say it is random but it is arbitrarily that means you can decide what you want.

So let us take a simple example of a three bit counter and whether it is up or down it doesn't matter because as I said all you have to do is to take Q or Q bar for up and down respectively. So the three bit counter will be three flip-flops all of them are JK one clock, this is Q and this is Q bar so I will call this A B C three counts, A is the most significant bit and C is the least significant bit of this, this would be QA this is QB this is QC or I want to call this as A B C.

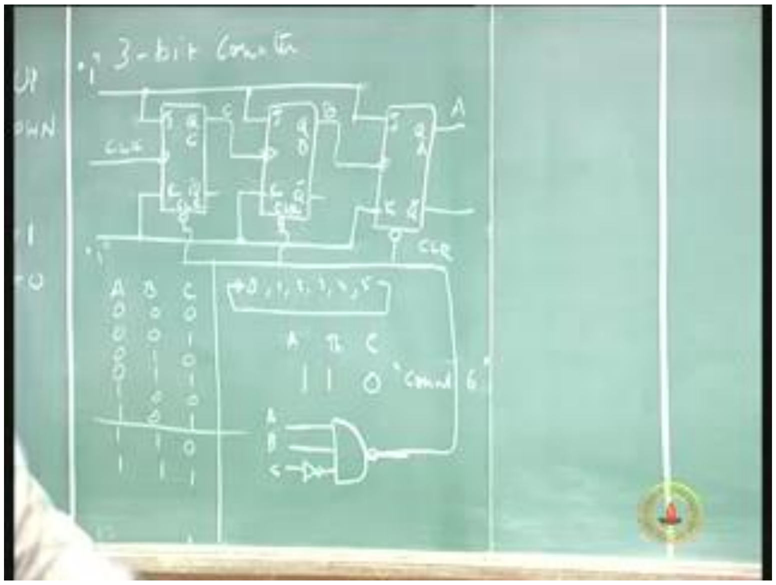
(Refer Slide Time: 17:26)



So I have A B C normally I go 0 0 0, 0 0 1, 0 1 0, 0 1 1, 1 0 0, 1 0 1, 1 1 0, 1 1 1 or if I take the complements it will be 1 1 1, 1 1 0, so forth 0 0 0. I have just drawn the up counter you can draw the down counter by taking Q bars instead of Qs as I said now. Now let us say I want after 5, 0 that means I want count from 0 1 2 3 4 5 and back to 0. I don't want the count 6 and 7 to go through for some reason but I want only 6 states to be counted. So all I have to do is to take this when 5 is reached clear all the flip-flops again by using the clear inputs. So all these flip-flops have clear inputs and preset inputs so I detect the count that I want to count and until that count reaches I will have to clear them. But I cannot clear it as soon as 5 is reached because if I clear it as soon as 5 is reached assuming there is no delay in the clearing operation I just reached 5 and it's cleared that means I will go 0 1 2 3 4 completely done, 4 is completely done, 5 has just started but I want each count to at least last for one clock period so up to 0 1 2 3 4 as soon as 5 is reached assuming there is no delay in the process of preset clearing it will become 0 that means I will not have one full clock period for 5. So really I should start looking at 6, when 6 reaches if I want to terminate it at 5 I should wait for 6 and as soon as 6 reaches assuming there is no delay involved in the presetting process and in the clearing process as soon as count 6 is reached I clear all the flip-flops. Therefore it will go from 0 1 2 3 4 5 and even 5 will last for one time period of clock and the moment it becomes 6 immediately I will clear it. so there will be a small glitch, I referred to glitch in one of my previous lectures, a glitch is a very narrow variation of the output a very short duration output variation from either 0 to 1 back to 0 or 1 0 back to 1 a pulse of going from 0 to 1 back to 1, 0 to 1 back to 0 or 1 to 0 back to 1. It could be that there is a small glitch because of the small time delay involved in the clearing circuitry, there is a circuitry involved. If you neglect that the other alternative is for me to have clock count 5 only for a very narrow period which is not correct, I don't want that I want full clock period for 5 also.

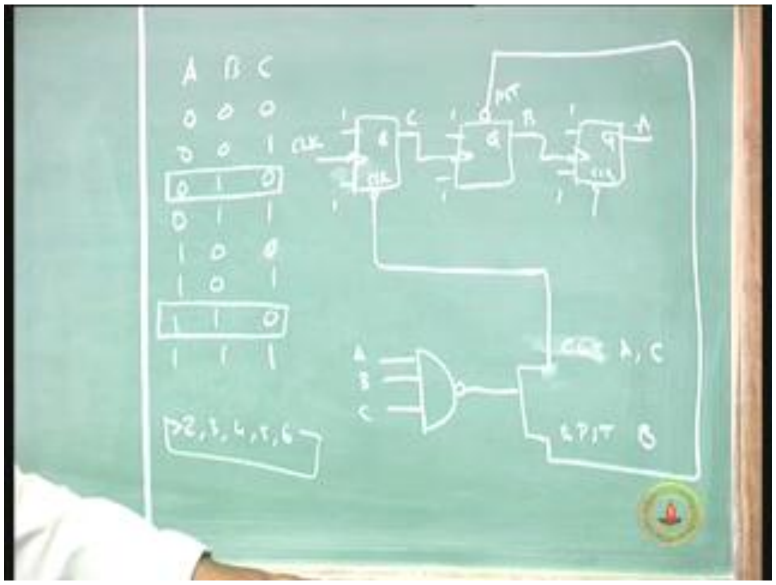
So actually if you want to terminate at count 5 you clear it as soon as 6 reaches. How do you do that? I detect state 6 and use the state 6 to clear that. So state 6 is given by 1 1 0. This is count 6, count 6 is 1 1 0. If 1 1 0 is reached I should clear immediately all the three flip-flops. that means I should take this into an AND gate and 1 1 0 becomes 1 1 1 this is 0 when 1 1 0 is reached this is 0 (Refer Slide Time:: 21:42) so a three input NAND gate which will give 0 output when this becomes 1 1 0 or this becomes 1 1 1 and this is what I will connect to the clear inputs of the three flip-flops.

(Refer Slide Time: 23:00)



So the clear inputs of this flip-flop…. and from where the A B C is coming from? They are coming from AC, in order to not clutter the drawing I put it here, this A B C are not different signals these are the outputs of the flip-flops, A is the output of this MSB flip-flop, this LSB flip-flop and this in between flip-flop. So assume the flip-flops A B C take the value 1 1 0 this becomes 0 so I would have a count 0 1 2 3 4 5 and momentary, momentary, momentary short duration count 6 which can be ignored as a glitch and then start all over again. If you do not want to start at 0 I connect it appropriately to preset and clear. Suppose I want to start with 3 so the second example is this (Refer Slide Time: 23:30) suppose I want to start at count 2 and yet end at count 6 I want counts 2 3 4 5 6 and back to 2 this is the cycle I want. I don't want to count 0 and 1 and don't want to count 7.

(Refer Slide Time: 27:30)



So again I will do the same thing. I will look at 6, 6 I have to complete I cannot detect when the flip-flop reaches the count of 6, I have to wait for count 7 to come up because 6 has to be completely allowed so when count 7 comes up which is 1 1 1 I detect that and not clear all the flip-flops in this case I will preset this flip-flop and clear these two flip-flops (Refer Slide Time: 24:27). So I will get the count 1 1 1 this is A B C this pulse will be used to clear A and C and preset B preset will become 1 so 2 will start and after that it will be 2 3 4 5 6 back to 2.

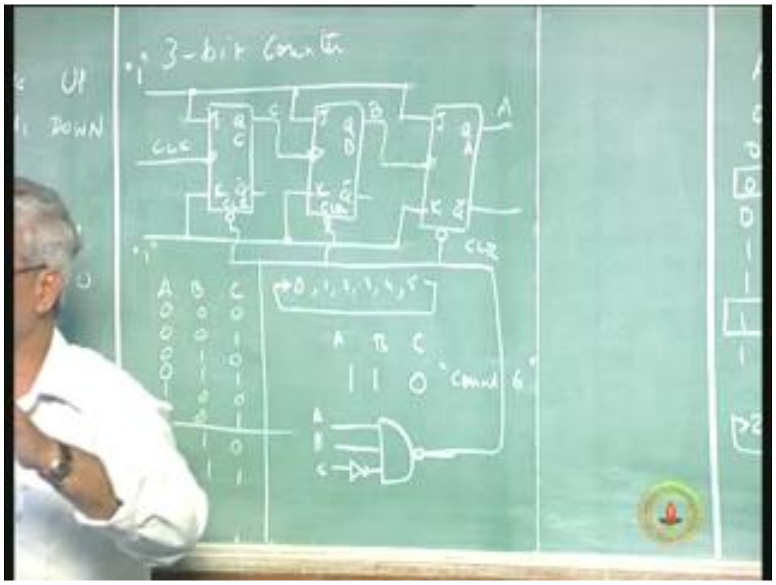
Hence, by using this preset and the clear inputs I can start the count at any point and I can terminate the count at any other point, at any count we can start. And if you want to down count all you have to do is to look at Q bars instead of Qs so I have an up count, a down count and arbitrary starting point, arbitrary ending point and number of flip-flops as I said the number of counts, total maximum number of counts possible is 2 power n when n is the number of flip-flops since 0 has to be counted as one state or first count the maximum count you can reach is 2 power n minus 1. So this is how you count as I have already said counters are used to count people, to count events, count things, clock division is used all the flip-flops are very very useful circuits, in many applications you will find the use of flip-flops. We will see many more of them, do you any questions?

So in order to complete the drawing if you want to draw A B C this is my C B A Q I will not show the rest of the drawing of JK etc, you know what it is all Js are 1 Ks are 1. Now A B C when it reaches 0, 1 1 1 becomes 0 we count these two, this is clear and clear (Refer Slide Time: 27:15) and preset, this is my input clock, this is the counter which counts from 2 to 6.

In any circuit there is a transient state, there is not even a guarantee that it is going to start with 0. When you switch on the flip-flop the first time arbitrarily the clock is applied arbitrarily the counter can reach where the first counter may be 1, second counter may be 0, third counter may be 1 or 1 1 0 it can be anything. The very first cycle of count you are not guaranteed of any starting value, it could all be 0, it could all be 1 or it could anything else but moment the second time that count reaches so there is a transient state, the circuit has to settle down, once the circuit settles down it does what you want. You never take into account the initial state after you switch on the power. The first time when you switch on the power there is no guarantee the flip-flop is going to start at 0 0 0, B may be 1, A and C may be 0 and so on.

On the other hand 1 and 2 may be 0 and the other one may be 1 this is 0 0 1 or 1 0 0 so you have to go for 1 0 0, 1 0 1, 1 1 0, and when 1 1 0 comes it goes to 2 so that is the cycle. So the initial portion in linear circuit is called transients. You don't talk about transient only with a steady state a stable state you analyze the circuit. In this case you can call it latency digital terminology digital parlance is called latency. Latency is the time you have to wait for the circuit to start working after you give the inputs. This is a very common word, latency is the time at which the circuit starts delivering the output after you switch on the power and after you give the inputs. So there is an in-built latency in any circuit.

(Refer Slide Time: 31:26)



Now we have seen counters and we can design any counter we can and count up to 128 if you want to or 0 to 127 or anything else also. For example; think of a clock, somebody gave an example of a clock being divided by 2, example of a division of frequency. We have a very high crystal frequency, these clocks run on redundant crystal the watches the watch chips, let us call it watch. A watch chip an IC Integrated Circuit which is used in the watches it has this clock. This clock is derived from a very high frequency stable source called a crystal oscillator, there is a crystal. Generally it is easy to build crystals of very high frequencies, it is very difficult to get a crystal of 100 Hz, 10 KHz or something like that.

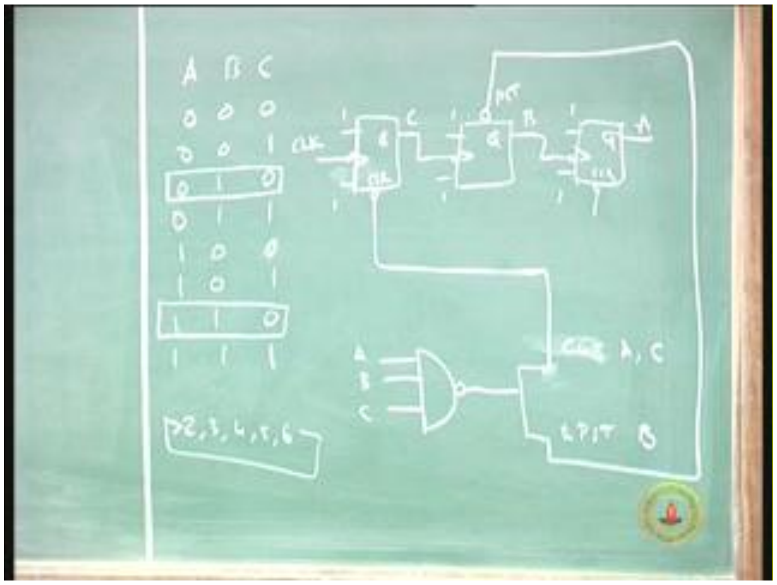
Usually megahertz and several megahertz is the crystal frequency that is available which stable frequency. What do you mean by stable the frequency? The frequency will not drift. Some of the wrist watches you buy in some of the shops, bazaar you know what happens is you think it is a good watch and you buy for 25 rupees and the moment you start wearing it and the next morning you look at the watch it is already 9 O'clock in morning when it is 5 or 6 or the other way it is going slow and showing 2 O'clock. The reason is because the frequency is not stable it counts down, but then if the original crystal is not a stable frequency it will either lose time or gain time which you do not want. So this division by a factor of whatever required to make a clock.

So I will give this as an exercise; assume a crystal clock of very high frequency megahertz, I want you to find out how many stages of counting is required so that I can have a count of seconds, minutes and hours not beyond that. I want an hour display, a minute display and second display in my clock. So the crystal frequency clock has to be brought down to the hours, brought down to the minutes I mean the other way very very high, you do several divisions to make it one cycle per second to count seconds then divide by sixty cycles per minute, divide by 24 divide by 60 minutes to make it one hour so I am not giving you a clock exact frequency because I want a whole number finally. So it is the multiple of the clock because 60 times 60 so 3600 seconds make an hour. So 60 minutes make an hour, 60 seconds make a minute so 3600 seconds make an hour. So make it 3.6 MHz I am having a 3.6 megahertz clock given to you, design a series of counters so that I can get my frequency divided at right numbers to have a count of seconds, count of minutes and count of hours. This is an exercise.

Now one disadvantage of this counter is, in these counters we feed a clock to the very first stage of the flip-flop, the output of the flip-flop is used as the clock for the second stage, the output of this flip-flop is used as the second stage for this clock and so on, the clock is passed from flip-flop to flip-flop. There is an inherent delay as I said. They also talked about small glitch. The moment I reset it it is not going to get reset that is going to be short time duration depending on this. once a one 1 0 whatever is reached here there is a small delay here and it takes its own time to become 1 1 1, it takes its own time to become 0 that 0 is applied here, here, here it has to activate the circuitry for clear to become 0 0 0 there is small amount of delay involved that is a short pulse as I said it is a glitch so you can ignore it.

Now each of these flip-flops has also the propagation delay. I told you the flip-flop have this initial stage, clocking stage and the next stage will be latching stage so there is a propagation delay involved in each of the flip-flops so I give this clock it takes a while a small amount of time though propagation delay time before the output changes that is fed into this and this is going to take its time before this is going to change (Refer Slide Time: 35:19) and this is going to be applied here, this is going to take its own time and this is going to change.

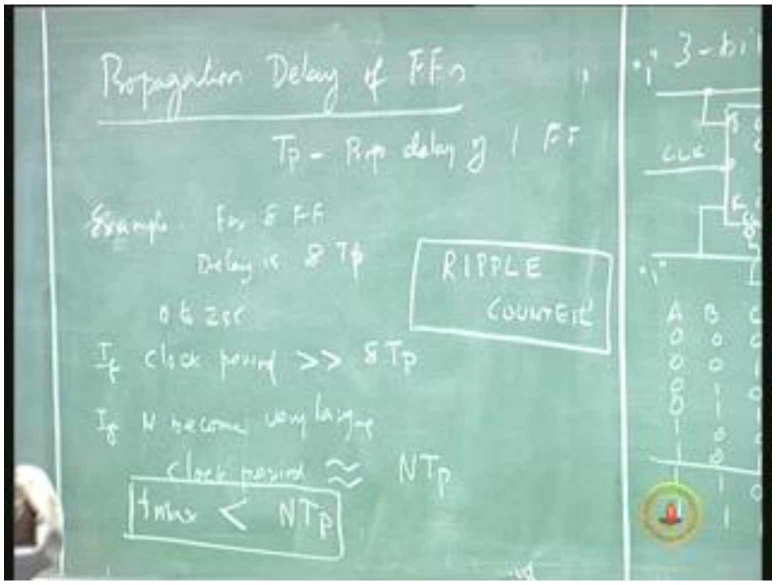
(Refer Slide Time: 37:40)



So let us go through this. supposing I have this count and I am applying the clock here the next count is this (Refer Slide Time: 35:40), this one has to become 0 first flip-flop C flip-flop has to become from 1 to 0, b flip-flop has to become from 1 to 0 and A flip-flop has to become from 0 to 1. I am feeding the clock to the C flip-flop as you can see there so it is going to take its own time for this to become 0 and after that delay only this flip-flop will go from 1 to 0 and after that delay only this 0 will go to this 1. So, from 0 1 1 to 1 0 0 to settle down it is going to take a final amount of time approximately equal to 3 into propagation delay of flip-flop we call T_p T subscript p as the propagation delay of the flip-flop, I want three propagation delays approximately, of course there are other things like reset, clear etc.

Once you have a number you can always compute but there is a concept here. You should know there is a number you will have to look for. Once a number is given to you then you can make it approximate, after two nanoseconds by looking at the data sheet you find that it is not two nanoseconds but it is three nanoseconds but still it's all right you have to do a computation but you should know there is something like that we have taken into account. So there is a delay in each flip-flop stage which affects the change and for more number of flip-flops there is a delay in counting so it becomes longer and longer.

(Refer Slide Time: 44:15)



So if you have eight stages with 8 flip-flops what is the count I can reach? Flip-flop propagation delay: let us assume T_p to be the propagation delay of a single flip-flop so we have 8 flip-flops 2 power 8 as an example for 8 flip-flops delay is $8T_p$ so what is the count I can reach from 0 to 2 power n minus 1 where n is 8, 2 power 8 is 256, so for a count from 0 to 255 each count takes $8T_p$ to materialize. If I am 0 after $8T_p$ 1 comes and after $8T_p$ 2 comes and so forth reliably. Of course something can happen faster than the other. For example here to here (Refer Slide Time: 39:10) there is no change in this flip-flop so it is going to immediately come, there is no change, there is no question of propagation delay. It is not that each count is going to change only then. This I should allow for a maximum propagation delay of 8 times T_p in order for a count to be reliable.

Now talking of these clocks the watches from 3.6 MHz you have to count down to the second. One second is 1 Hz per one cycle per second which is one second and megahertz is 10 power 6 times 3.6 times so 3.6 million to 1 you will have to find out the number of flip-flop stages that is required. When that is the number of flip-flop stages required each count is going to take that much longer to materialize. Now if I run it at a speed which is faster than that speed so my clock period that is starting from one point in the clock and stopping exactly in the same point in the next clock cycle.

If you take the positive edge of the clock then the time duration until the next positive edge comes is what is called the period. If I take the negative edge from one negative edge to the next negative edge is known as the clock period. So if this clock period is much much larger than $8T_P$ then you are not worried, it is only a small amount of time, anyway it is going down to one clock period to settle down. The count is not going to change for one clock period. For every clock period the count is going to remain from 0 to 1 to 2 to 3 to 4 to 5, each of these counts is going to remain for one clock period. What if it takes $8T_P$ to settle down? $8T_P$ is the insignificant part of my whole clock period. But supposing I have the number of stages more, if n becomes large clock period may approximate the value of the propagation delays of the n flip-flops. Each flip-flop has a propagation delay of T_P and n flip-flop stages will have n time T_P and n is very large. So I can not run this counter at a speed faster than this NT_P. Thus maximum clock period has to be less than NT_P. If I run my counter faster than this before the count settles down the next count has to come. All counts are arbitrary then in that case reliability is not there. This is what I have been stressing all the time. You can not reliably say the count has reached from 0 to 1, 1 to 2, 2 to 3, 3 to 4 it keeps changing constantly because the clock keeps coming and sometimes it may be right depending on the number of transition sometimes may be wrong but we don't know which is right and which is wrong. So I have to either use very small number of flip-flops there is a limitation or use extremely low frequency clock which is not always possible as I said the crystal oscillator stable oscillators.

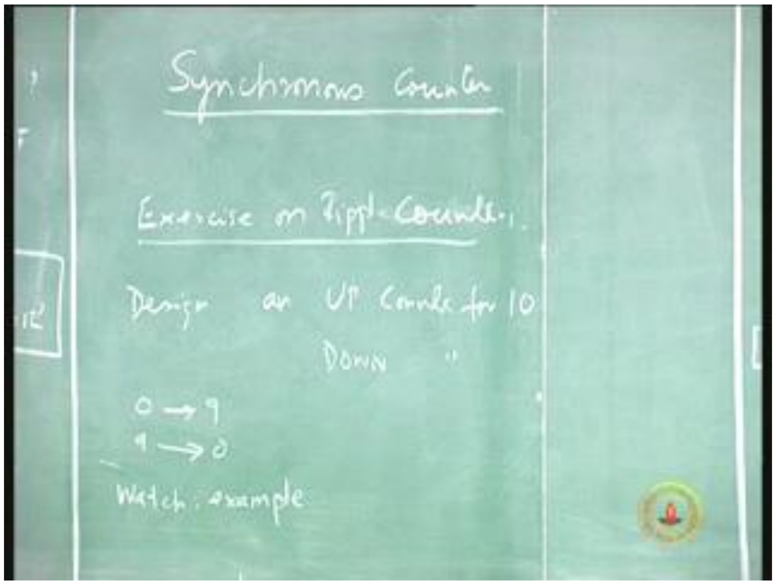
There has to be crystal and these are all only available in megahertz range. You cannot reduce the clock frequency for the reason of stability not always but sometimes it is possible or you want particular speed of counting, for my application I need to count this fast events keep coming the number of people I am going to count if people keep coming constantly or some small events some atomic events or something you keep counting at the rate at which it has come into account and I cannot say I can only count at this rate so please slow down I cannot say that. Wherever it is possible to slow down your event or increase the clock frequency or have less number of flip-flops there is no problem. But there may be situations where the events happen fast because you are ready to use a high frequency clock and the number of flip-flops used is also large because the count has to be reached with a very large value then this counter fails.

So this counter which is called a ripple counter suffers in the fact when the clock has to ripple through the flip-flops. Each flip-flop clock is the output of the previous clock. I give the clock to the original clock to the first flip-flop and the output of that flip-flop is given as the clock to the second flip-flop, and the output of second flip-flop is given as clock to third flip-flop, the clock doesn't directly get into the flip-flops the clock that is given to you as a circuit in the crystal oscillator whatever I said that clock doesn't directly go into different flip-flops it only goes to the first flip-flop and it has to ripple through like a wave in the ocean, it has go into this, into this, into this, into this and each stage it accumulates a delay because of the propagation delay of the flip-flop. So the clock accumulates delay and if the number of stages is larger than the events that is if the propagation delay is longer than the frequency of the events the distance between two events the time difference between two events then you lose track of the count.

Ripple counter: The very name suggests it is a ripple which means that the clock is not directly applied to all the flip-flops but the clock is applied to the first flip-flop which ripples through different flip-flops to different stages. The ripple counter files if the number of stages are very large such that the total propagation delay is of the order of, I won't say mathematically it should be larger, even in a practical electronic circuit when it approaches the order of this clock frequency clock period the total propagation delay approaches the order of the propagation delay it will fail to be a reliable circuit because you can never predict exact values. So I have to think of flip-flops being used as counters, usually a lot of things as I said, it can do a division by 2 you can count up count down start with any value end in any value and so on. Counting is one of the important things we do in many application systems but ripple counter has to be replaced by some other counter. So the improvement of this ripple counter is called a synchronous counter. A synchronous counter in principle is one which will take care of this problem. Ripple counter is suffers from the problem of too many stages in propagation delay affecting the total number of count you can use it for.

On the other hand a synchronous counter is more practically used. In a synchronous counter all flip-flops receive the clock at the same time, there is no question of clock ripple through. The clock that is given be it a crystal clock or stable clock or whatever clock you design for the circuit is given to all the flip-flops so they are all clocked all the time but the change will depend on the previous count because if all the flip-flops are given in the clock and all the flip-flops change up and down then I can only count from 1 1 1 1 to 0 0 0 0 if all are toggle flip-flops so one state is all 1 1 1 1 and other states are all 0 0 0 0 then that is something I don't want. I will be able to get all intermediate counts also. So we will see how to use flip-flops whose clocks are all tied together to the same clock that is the given clock is applied to all the flip-flops not to the first flip-flops alone but at the same time how to get a different count these are called synchronous counters we will see it in the next lecture.

(Refer Slide Time: 49:09)



Synchronous Counter

Exercise on Ripple Counter

Design an UP Counter for 10

DOWN "

$0 \rightarrow 9$

$9 \rightarrow 0$

Watch : example

Before that we can do a lot of work on these ripple counters. What I have given you is a flavor.

Exercise on ripple counters: I would like you to design an up counter for 10 and down counter for 10so these are called decimal counters it goes from 0 to 9 or 9 to 0 and any other value you can also do by using these gates. I also gave you the watch example. These are the assignments on this ripple counters. But actually in watch we will not use a ripple counter. Finally we will say that because watch has so many stages you will see that it may not be worthwhile to do the ripple counter but we may have to use a synchronous counter. So we will talk about synchronous counter in the next lecture.