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

Lecture – 42
Decoding Logic and Synchronous Counter

Hello everybody, in the in this week, we are discussing counter. In the last class, we had seen asynchronous counter. We had seen up counter; we had seen down counter; we had seen up and down counter put into one particular circuit. And also we look at looked at one particular IC. Today's class we shall look at decoding logic what it is and how it is useful in counter design context and also we shall discuss synchronous counter, up counter, down counter and up-down counter.

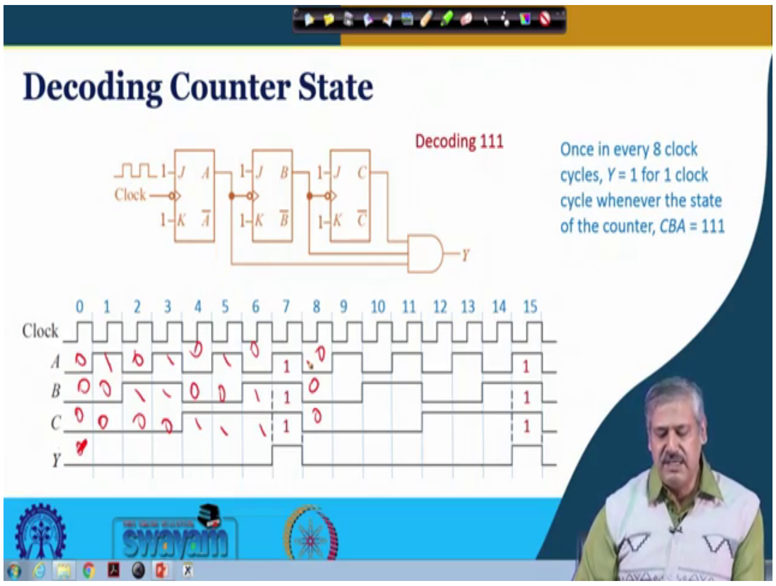
(Refer Slide Time: 00:47)

CONCEPTS COVERED

Concepts Covered:

- ❑ Decoding Logic
- ❑ Glitch and Strobed Decoding Gate
- ❑ Synchronous Up Counter
- ❑ Synchronous Down Counter
- ❑ Synchronous Up – Down Counter

(Refer Slide Time: 00:53)



Decoding Counter State

Decoding 111

Once in every 8 clock cycles, $Y = 1$ for 1 clock cycle whenever the state of the counter, $CBA = 111$

Decoding logic, why decoding a counter state is important? So, for example, we consider the mod 8 up counter asynchronous up counter that we had taken up in the previous class ok. So, the counter has got 3 flip-flop JK flip-flops each one of the input is having 1 and 1; that means, whenever the clock trigger comes to its input, it toggles ok.

And so the counting state as we had seen before in the last class moves from 0 0 sorry this is the output 0 0 0 to 0 0 1, then 0 1 0, then 0 1 1, then 1 0 0, then 1 0 1, 1 1 0, 1 1 1, and again it comes back to 0 0 0, and it continuous. This we have already seen in the last class we discussed this.

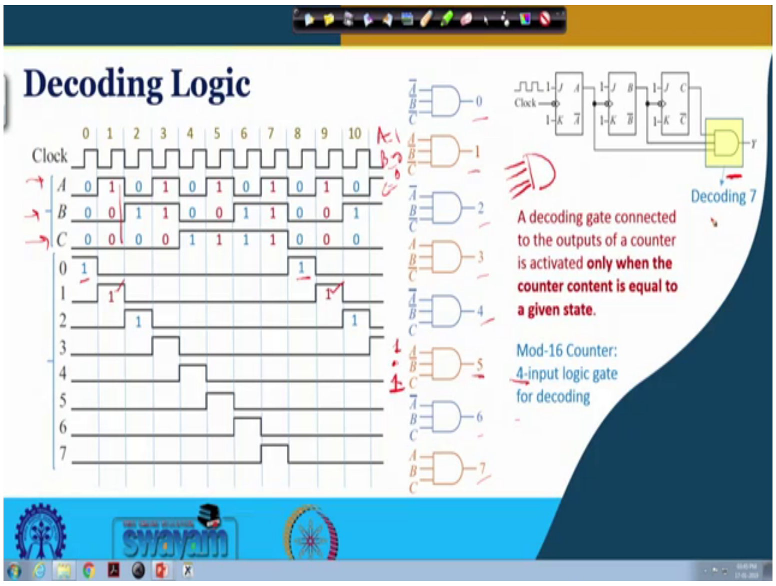
Now, so this is happening internally. If we look at all the three inputs, we can understand that a counting of a specific number of a clock at or the trigger has happened ok. But to the external world if we want to know that a count of 7 has taken place, do we need to sample all three of them, and you know figure it out or we can make a logic an output a separate output available which will go high whenever the count of 7 takes place ok. So, that we do not need an additional thing where we need to look at all three values; and from that infer whether a count of 7 has taken place or not or count of 5 has taken place or some such thing.

So, in this example we are showing that a count of 7 has taken place that can be understood by having a three input AND gate over here out in where A, B, C are connected in this manner. And when all of this A, B, C are high as is the case over here in

7th you know after the 7th point ok, so 7th clock cycle if we have consider these as 0th ok, then the output will go high for one clock period. Is it clear right?

So, this one this output is 1, that means, 7 clock edges starting from 0 has taken place 1, 2, 3, ok, 1, 2, 3, 4, 5, 6 and 7, 7 clock clocking, 7 triggering has happened Is it clear? So, this will again go high, when another 7 occurs right. So, this is the utility of having a decoding circuit by which we can figure out whether that particular a count has taken place or not ok.

(Refer Slide Time: 04:11)



Decoding Logic

A decoding gate connected to the outputs of a counter is activated **only when the counter content is equal to a given state.**

Mod-16 Counter: 4-input logic gate for decoding

Now, if we extend it, if we extend it, instead of say 7, if we are interested in decoding whether a count of 5 has taken place or not ok. So, what shall I do? So, we shall put a output gate, where the in input of which will be C is the MSB here, C is you know changing with a frequency which is less than that of A and that of B right. So, when C is 1, B is 0, and A is 1, right that indicates that a count of 5 has taken place, that means, five clock edges have come starting from 0. Is it ok, right?.

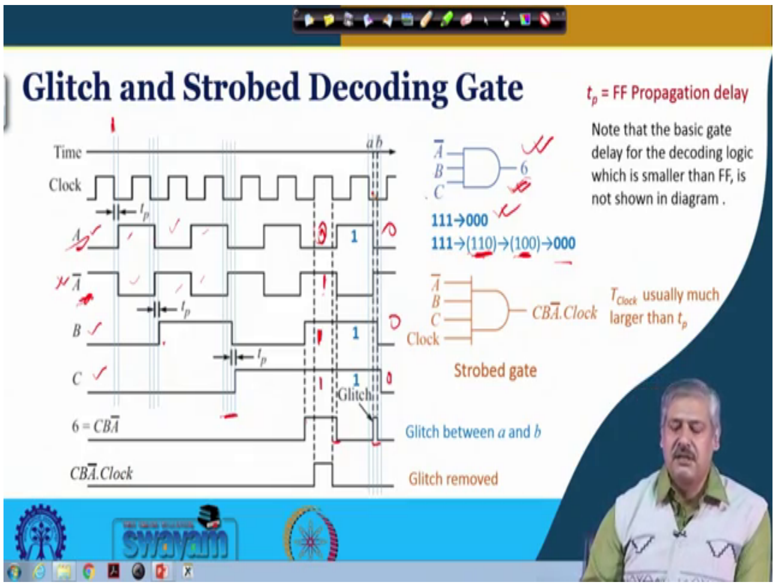
So, similarly 0, 1, 2, 3, 4, 5, 6, 7, 7, we have already seen. So, depending on our requirement, we can put this you know decoding logic at the output of the counter. And accordingly this output will go high, whenever the count the count value in the counter reaches that 1. And it remains high for one clock period.

So, in this example, you can see since this 0 we will go high here again that is A bar, B bar, C bar, all of them are 1 at this point; and again when it comes over here right, one this decoding decoder output ok, this logic gate output, so C B both are 0, and A is 1 right. C is 0, C is 0, B is 0 and A is one, so that is the occasion when it will go high, so that is the case when this is happening right this is going high here and again going again high here. So, after every 8 clock cycle, it will happen.

Now, similarly for the other cases. Is it clear, right?. So, in the circuit, in this place instead of 3 input AND gate, we shall put one of this connection right. And we can take wherever A bar is there, we will be to take it from the in inverted output instead of A the way it has been shown here right.

Now, instead of say mod 8, if we had a mod 16 counter right, so to have a count of say in that case if 15 or 14 depending on that in the requirement, so we need we would have required a four input logic gate. So, depending on in you know the number of the count value the mod 16 mod 32, whether a it will be 4-input 5-input, accordingly the number of input for the decoding circuit will increase.

(Refer Slide Time: 07:05)


Glitch and Strobed Decoding Gate

t_p = FF Propagation delay

Note that the basic gate delay for the decoding logic which is smaller than FF, is not shown in diagram .

$111 \rightarrow 000$

$111 \rightarrow (110) \rightarrow (100) \rightarrow 000$

T_{Clock} usually much larger than t_p

Now, we look at what happens one specific issue related to glitch that we discussed we introduced in the last class regarding discussion related to asynchronous counter so the effect of propagation delay. So, in the previous diagram, we did not considered it. Now, we are considering it. So, this is the time ok; this is the clock. And here it is A; here it is B; here it is C ok.

And we had seen in the previous class, this is the clock edge, this one this blue line you can see, and A is getting triggered right. And after one proper propagation delay, A is a value is changing from 1 to 0, because it is struggling in every clock in edge negative edge of the clock right. Now, B gets triggered from A; clock of B flip-flop is actually fed by fed by A, sorry this one ok, this part I shall discuss it later right.

So, after the negative edge of the clock, so another propagation delay is required for B to change state, and B will toggle every time A changes from high to low ok. Similarly, for C. So, there is a there are three propagation delays. So, this we have seen in the last class. So, this is the way A B C you know changes its value right, and the propagation delays are cumulative right.

Now, consider that we are looking for a count of 6. So, then the decoding logic required is C B A bar right. Say A bar is the complimented output of flip-flop A so it is changing. Whenever A is going high, it is going low and vice versa, so that is what is there in A bar ok, A bar wave form all right. And this will go high right when this is 0, this is 0, and A is sorry this is 1, C is 1, B is 1, and A is 0, that is A bar is 1 ok. So, this is happening where when it is reaches the count of reaches the count of 6 ok. So, this is the case C is 1, B is 1, and A is 0, so A bar is 1 ok. So, this three are 1, at that time A bar B and C, and the output will go high right that is what you see right.

And of course, this AND gate, there will be small propagation delay which has not been shown here in which is less compared to this flip-flop propagation delay so, but there will be small delay over here ok. But the idea here is that the output will go high, when these three are 1 1 and A is 0 that is A bar is 1 all right.

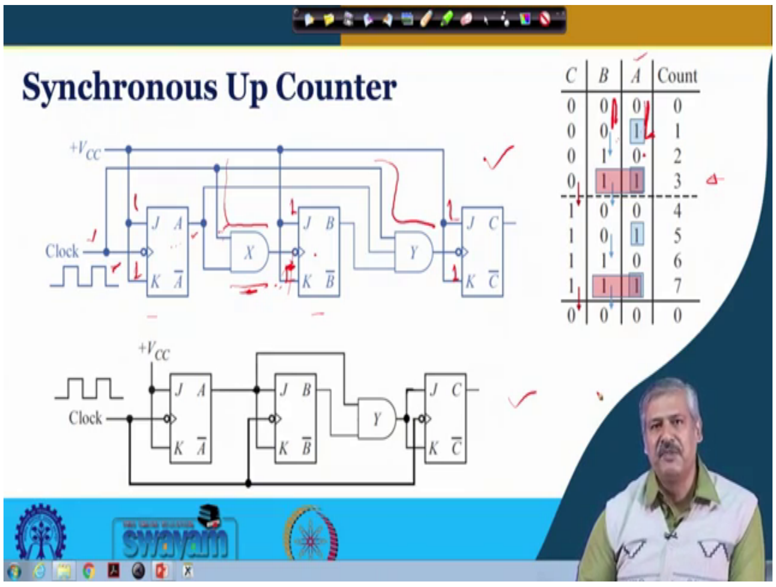
Now, moving forward, after 1 1 0, there is 1 1 1. And after 1 1 1, there is a it should be 0 0 and 0, this is the value that is expected all right. But because of the propagation delay all right, after 1 1 1 you can see for a small duration it still remain is remaining at 1 ok. B as B is 1, but A has already change from 1 to 0. So, for a small time for the duration of the propagation delay, you have got 1 1 0. After that B has changed right, but A has already changed, but C has not yet changed. So, for a small time 1 0 0 is there; and after that only you are getting the 0 0 0 right.

Now, whenever these 1 1 0 is there ok, so this logic decoding out decoder output will go high for a grip period ok. So, this is the glitch that you can see. And if this output is feeding, another counter or another say sequential logic circuit which is you know counting, the number of trigger that is happening number of changes happening here right. So, it will get a wrong message that a count of 6 has taken place. Again a count of 6 has taken place just after 1 clock cycle which is wrong right. So, this is something which is the which is a problem with a synchronous circuit, and we need to think of getting around solving this problem addressing this problem right.

So, one way to solve this problem is to associate the clock and increasing the number of input to the decoding gate to by a by 1. So, there were three inputs, now you are making it four inputs. So, when you associate the clock and of course, clock period is considered much larger right, then the, this propagation delay. Then what is happening over here, so this is getting ended right. So, this propagation delay is occurring when the clock is there in the low value all right, you can see this is the where the clock is changing. So, the change is initiated when clock is going from high to low ok, it is negative edge triggered. So, this is the time clock is remaining at low.

When you are ending it with the clock, then this glitch goes away ok. Of course, we are not considering clock to be so fast that this delay and delay so much that delay is coming when the clock is again becoming high right, so that is ruled out ok. So, propagation delay and the number of flip-flops considered and the clock period are such that it is occurring the glitch is occurring when the clock is low ok. So, then the glitch can get removed right. So, this is one way of handling this problem.

(Refer Slide Time: 13:13)



Synchronous Up Counter

C	B	A	Count
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7
0	0	0	0

Now, this brings up brings us to the discussion on synchronous counter ok. So, in the synchronous counter, this accumulated delay and resulting glitch this aspect are not there ok. See in the synchronous counter all the flip-flops are getting triggered by the same clock by the same clock ok. Now, we see here in this case two diagram two arrangements ok. So, this is one circuit, this is another circuit I mean there giving you will give you the same result.

So, in the first circuit what you see is that each of the flip-flop has got 1 and 1, 1 and 1, 1 and 1. So, both all of them can toggled right, all of them can toggle whenever they get the clock right. And you can see this clock is fed here in to flip-flop A. This clock is again fed here through an AND gate, we will come to that AND gate particular later, and this is also fed here. So, effectively the clock is being available to all the three flip-flops right which will make it a synchronous counter right.

So, of course, there is no AND logic here. So, every lock trigger, it toggles a flip-flop a toggles the way you know we expected to be the case the least significant bit part of it in a CBA configuration. So, A is toggling at every place 0 1, 0 1, 0 1, more about this boxes etcetera we shall discuss later in this diagram, in this table, all right.

Next for flip-flop B, clock has come right, but clock will clock has come up to this, but clock will go is allowed to go to this place only when A is high because of the AND logic right. So, only when A is high, only when A is high right, so we will get the clock for which it will trigger, it will sorry it will toggle ok. And when A is 0 like this case, when A is 0 initialised with 0 0 0, A is 0, clock has come here as well as here, but the clock has not advanced to this point right for which A has toggled, but B has not toggled. You can see A has toggled but B has not toggled ok.

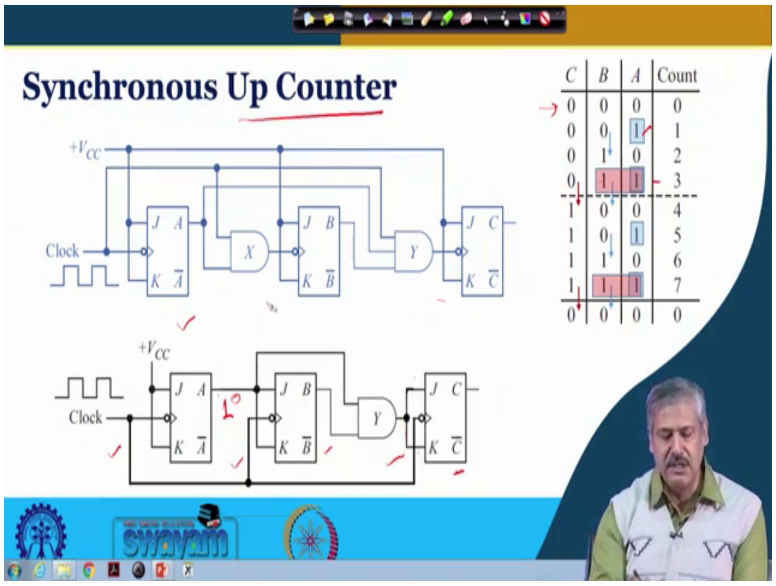
So, when A is 1 right, that means the next you know clock cycle right, then this is 1 and the clock has come. So, according to the clock at the negative edge of the clock, it will toggle, so it has become 0 1. So, then again A has become 0. So, when A A becomes 0 right, so this clock is not available it is coming up to the AND gate input right. So, it will remain at the same value, input is because in that clock cycle there is not toggling happening. So, this is remaining 1 1. And this continues that way it is 0 0 1 1 0 0 1 1, again it will go becomes 0 0, so that is for flip-flop B.

And for flip-flop C, what is happening, for flip-flop C clock again has come right, but when A and B any one of them is 0 right, this output is 0. And when both of them are 1, that is it has reached 1 1 right. It has reached 1 1, then only the clock is allowed to come over here and trigger the flip-flop C ok. And the this flip-flop C is getting triggered, and the output is its value changes from 0 to 1. Again it will change when both of them are 1, and it will it will value will change to 0 ok.

So, the value 1 1 or 1, it is made available in the previous clock triggered itself right. So, that way it this is not contributing to the delay ok. In the way that this flip-flop this is not actually triggering. A output is not triggering, it is only the clock is triggering a output is just ensuring that the clock is made available, so that way it is behaving like a synchronous counter ok.

Now, let us look at the other circuit ok. So, in this the clock is directly fed to each of the flip-flop right. The first flip-flop is toggling at every instance right.

(Refer Slide Time: 18:15)



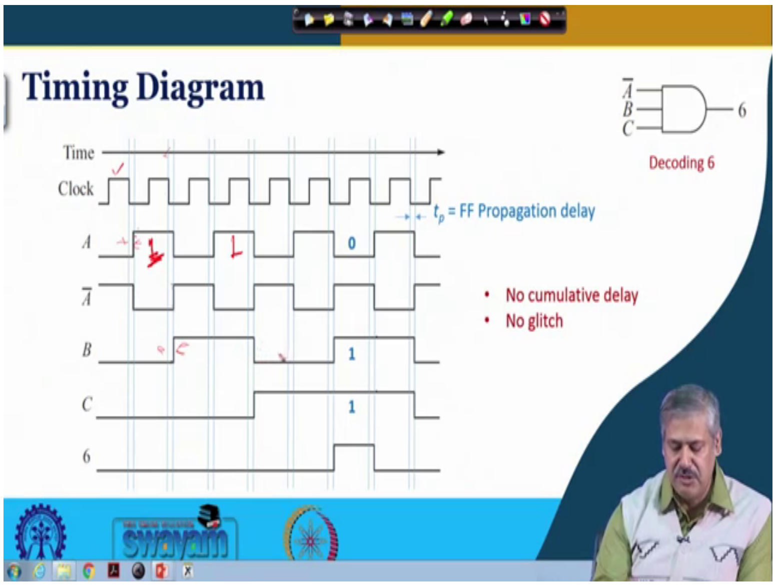
Synchronous Up Counter

C	B	A	Count
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7
0	0	0	0

And next flip-flop it when it A is 0, the output will remain 0 0 no change. When A is 1 right like this case, when A is 1 right, then this will both of them are 1 J and K. So, it will toggle. So, the output will become from 0, it will go to 1 right. So, next clocking instance, it is 0 all right, so it will get a 0 here, so no toggling right. Again it will get a 1 over here so that will make it change from 1 to 0 for B, and similarly it continuous right.

And for flip-flop C, what happens right, this is when both A and B are 1 right, then J and K both of them are 1 as is the case over here, then only when the clock comes it triggers ok, it toggles. So, it becomes 0 to 1. So, every clock instance, it gives gets a trigger, but it the value if it is 0, it will not change; if it is 1, it will toggle fine. So, this is the way these 3 flip-flops behave, and resulting you know values for CBA are 0 0 0, 0 0 1 to 1 1 1, and then 0 0 0. So, what is it, it is a up counter and it is synchronous ok, clear.

(Refer Slide Time: 19:47)

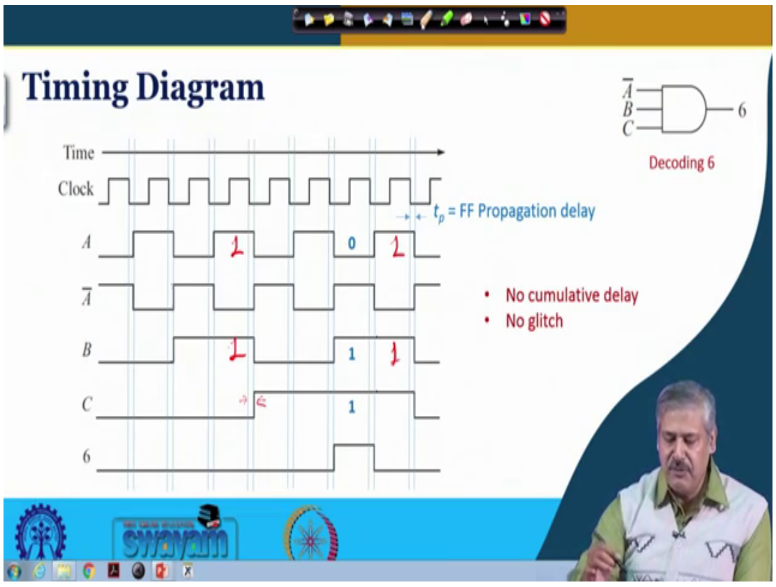

Timing Diagram

- No cumulative delay
- No glitch

And because of this if you look at the timing diagram, if you look at the timing diagram what is happening in this case, so this is your clock right this is the negative edge of the clock, A is changing at every clock the way you have seen. So, basically this becomes A the a after one propagation delay it changes right. So, it is moving like this ok and what about B right. So, B is changing again with the edge whenever A is 1, this is A is 1, then only B will change when the clock trigger comes ok.

So, clock trigger has come B has change because A is one right clear. So, it will change after one propagation delay only. So, A has changed the A value has taken from 0 to 1, after one propagation delay 1 to 0 1 propagation delay B has gone from 0 to 1 after one propagation delay right. So, there is no accumulated delay from A to B like asynchronous right. Again B will change when A is 1, then the clock is you know the clock triggered it will changed from 1 to 0 and so on and so forth. And what about C, C will change you have seen when both B and C are 1.

(Refer Slide Time: 21:11)

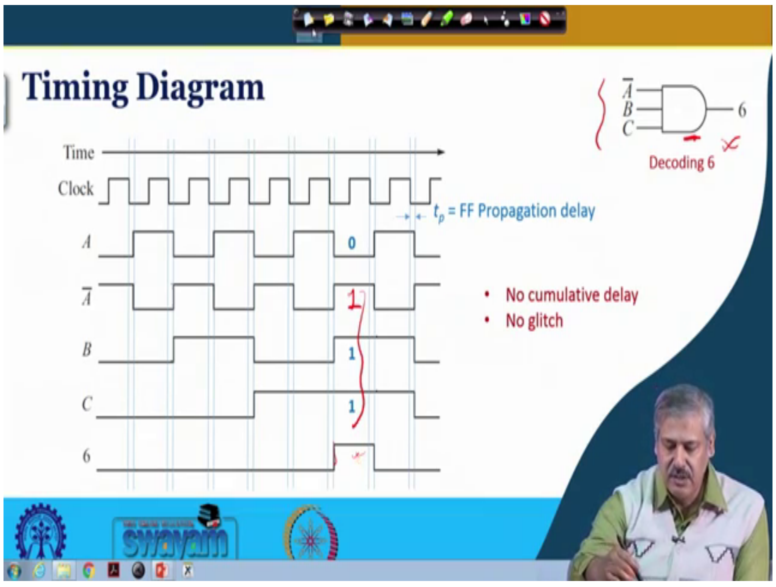


Timing Diagram

- No cumulative delay
- No glitch

So, this and this when both of them are 1 right at the clock trigger, it changes all right. And it changes after one propagation delay only right. So, this has become 0 to 1. Again it occurs in this place when both of them are 1, it has gone to 1 to 0, so that way you have got the timing diagram, where the flip-flops are all changing at the same clock trigger and the values are available after one propagation delay right. And because of which because of which if we have got a decoding, the way we were decoding asynchronous up counter say 1 1 0 right and in this case so A bar is just following A the complimented output.

(Refer Slide Time: 21:53)

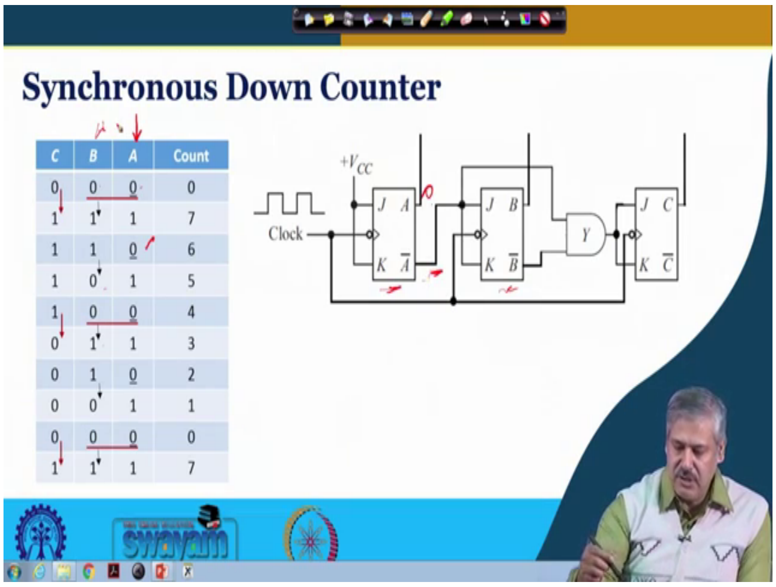


Timing Diagram

- No cumulative delay
- No glitch

So, at this time this is 1 1 1, this three one right. So, at that time this output will be 0. So, the propagation delay associated with this AND gate has not been shown in this diagram, but there will be small propagation delay as was the case for the last timing diagram. And it will remain high for one clock period right. So, this is the way it can the count of 6 can be decoded right.

(Refer Slide Time: 22:23)



Synchronous Down Counter

C	B	A	Count
0	0	0	0
1	1	1	7
1	1	0	6
1	0	1	5
1	0	0	4
0	1	1	3
0	1	0	2
0	0	1	1
0	0	0	0
1	1	1	7

Now, let us look at synchronous down counter ok, like we had asynchronous down counter following asynchronous up counter. Now, we have asynchronous down counter right. In asynchronous down counter, we remember that A bar was given as input ok, and then B bar was given as input you know to the next flip-flop the clock and all right. So, here we would be following the same logic right. So, first we look at the sequence ok, the down count sequence then we can see what would be the corresponding circuits. So, down count consequence is if you start from 0, then it goes to 7 we are talking about mod 8 counter only right, but it can be extended to mod sixteen and other values.

So, 7 6 1 1 0 5 1 0 1 4 1 0 0 0 1 1 0 1 0 0 0 1 0 0 0 again it is 1 1 1. So, this way it repeats is not it, so that is your down count ok. So, the first if we look at A, A is toggling at every you know a clocking clock cycle all right clock edge clock. So, this is how A is connected both of the both the inputs are 1 is fine right.

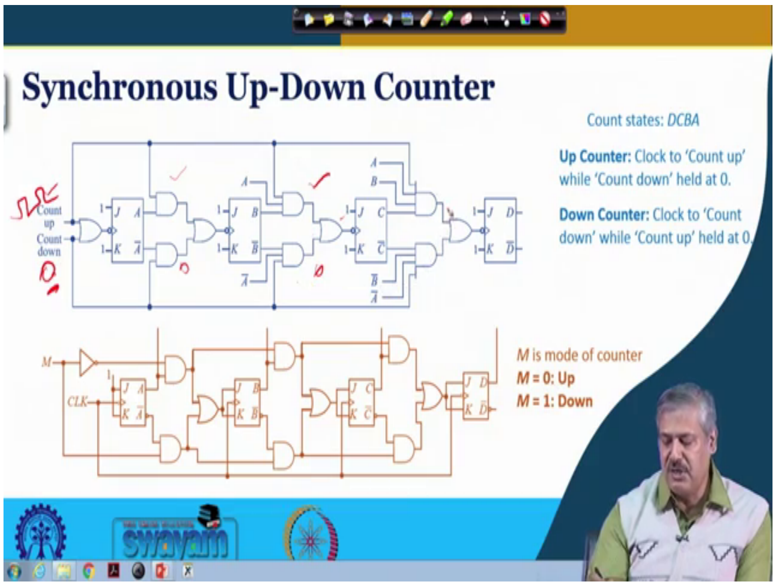
Now, when B is changing, B is changing you can see when this 0 black underline is there. Whenever there is A 0 ok, with the clock trigger A as value 0 A as value 0 right with the clock trigger right, we see that it is changing right. Again A has 0 with the clock trigger, it is changing right. So, this is changing from 0 to 1, this is changing from 1 to 0, again 0 to 1 and so on and so forth right.

Now, to make change happen in a flip-flop we need 1 1 at the input right, just in the toggle mode for a JK flip-flop. So, for 0, it is changing. So, instead of 0 A, we will put if we put A bar that means, that will make whenever A is 0 1 1 fed as the input to J and K, and with the clock it will toggle so that is where the connection has been made here clear.

So, similarly for C to change C to change what we see that follow this brown underline when both of B and A are 0, it is changing. Again 0 0, it is changing right; again 0 0 it is changing right. And both of the A and B 0 0 means A bar and B bar are 1, so that way right we can take this A bar and B bar ended and put them together, and we can get a output we can get a circuit like this ok, so that will give you a synchronous down counter all right.

And like what we discussed in the previous asynchronous case previous class asynchronous up and down counter, instead of A, B, C, if you take output if so possible from A bar, B bar, C bar, the output will be just opposite A A B C, if we gives down count A bar B bar C bar will be up giving you up count ok. And similarly in the previous case also for up counter design, if you take the output from the inverted terminal, complimented terminal, it will be giving you a down count.

(Refer Slide Time: 26:05)



Synchronous Up-Down Counter

Count states: *DCBA*

Up Counter: Clock to 'Count up' while 'Count down' held at 0.

Down Counter: Clock to 'Count down' while 'Count up' held at 0.

M is mode of counter

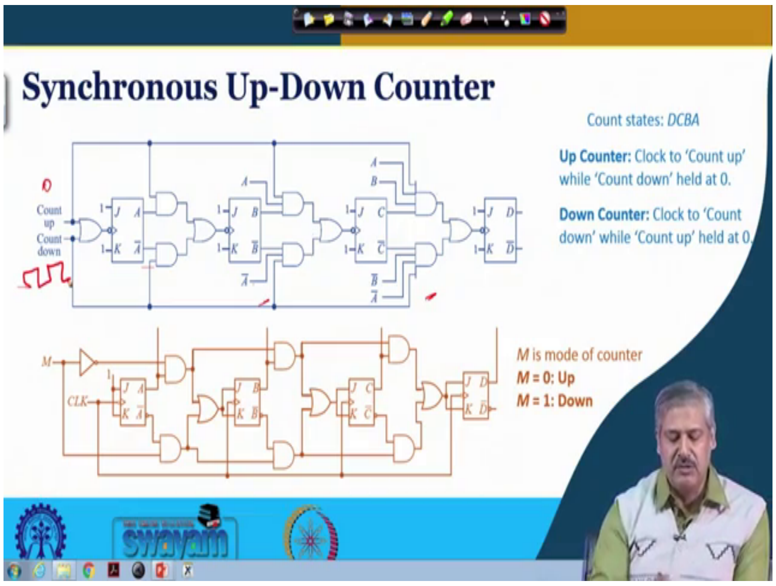
M = 0: Up

M = 1: Down

Now, we can move to a synchronous up and down counter right. So, again there can be two arrangement. So, in one arrangement this there are two clocks, one is a count up. So, when a when we are giving the clock through this, then the countdown clock is 0 ok. So, countdown clock is 0, this input is 0 right whenever we are feeding clock here right. So, this A is coming over here this is the first circuit first arrangement that we are seen in the discussion related to up counter; and its corresponding counterpart is coming here for the down counter right. So, this is coming, so basically then A ended with this clock is fed here right, and this will be 0 because this is 0.

Similarly, the A and B ended with the clock will come here, and this is this output will be 0 and so on, and it will continue right. And if we have if you are looking for down count, then you want to feed the clock through A bar B bar, A bar B bar here, A bar B bar C bar over here right, so that is what we will do.

(Refer Slide Time: 27:17)



Synchronous Up-Down Counter

Count states: *DCBA*

Up Counter: Clock to 'Count up' while 'Count down' held at 0.

Down Counter: Clock to 'Count down' while 'Count up' held at 0.

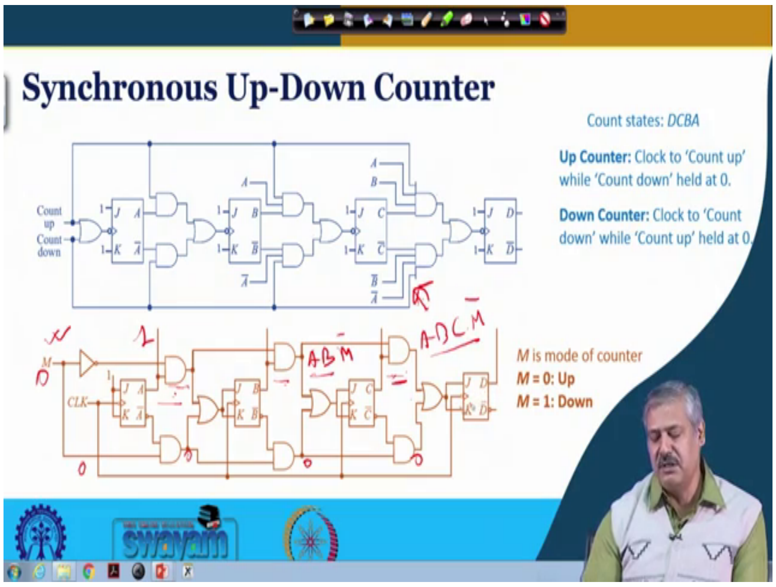
M is mode of counter

***M* = 0: Up**

***M* = 1: Down**

And we will put this to be 0 and this is the clock that will be fed here right. And in the other arrangement, again similar to these asynchronous up down counter, we can have a mod control input right when if it is 0 all right.

(Refer Slide Time: 27:35)



Synchronous Up-Down Counter

Count states: *DCBA*

Up Counter: Clock to 'Count up' while 'Count down' held at 0.

Down Counter: Clock to 'Count down' while 'Count up' held at 0.

M is mode of counter

***M* = 0: Up**

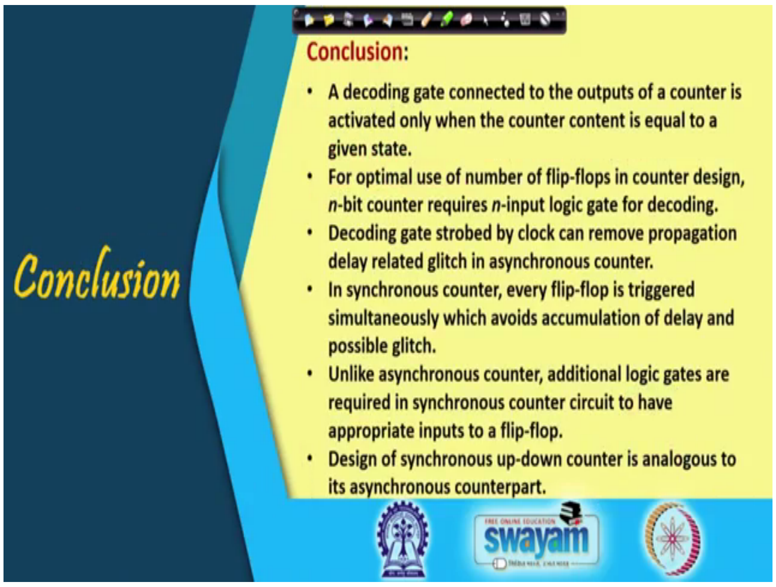
***M* = 1: Down**

So, then this is one and this is 0 all right and all this n gates outputs will be 0 0 0 right, and then A is fed to J-K input and in this case A and B ended is fed to the J-K input here, this is your this is your A right. So, this is A, B and of course, M bar is there with it right. And this is your A B is already there, A B C and M bar is there right. So, we can have a three input AND gate also like the one that you had seen before, in this case in state right here we are only having in all the cases because we are feeding the AND logic AND gate output of the previous stage. So, every case we have got a two input gate.

But remember in this case there is a small you know accumulation of delay right, because this gate, this gate this delay gates accumulated, so that issue may come up if you are clocking it too fast right so that need to be taken care of ok. Otherwise, it is suggest us only one propagation delay, but there too many if for a larger number of you know bit associated flip-flop associated in the counter design then the fanning could be an issue if we go for multi input AND gates.

And this arrangement there is only we are having two input AND gates, but the thing is that there is a at accumulation of delay before clocking of course, right not after clocking right. But so a flip-flop has changed, so after that the delayed value, we will make the appropriate clocking available to the next flip-flop ok, so that we need to take note of. And for n is equal to 1, we know that the down counter A bar B bar C bar, these are the one that will be fed here. So, this is how we can design a synchronous up down counter.

(Refer Slide Time: 29:51)



Conclusion

Conclusion:

- A decoding gate connected to the outputs of a counter is activated only when the counter content is equal to a given state.
- For optimal use of number of flip-flops in counter design, *n*-bit counter requires *n*-input logic gate for decoding.
- Decoding gate strobed by clock can remove propagation delay related glitch in asynchronous counter.
- In synchronous counter, every flip-flop is triggered simultaneously which avoids accumulation of delay and possible glitch.
- Unlike asynchronous counter, additional logic gates are required in synchronous counter circuit to have appropriate inputs to a flip-flop.
- Design of synchronous up-down counter is analogous to its asynchronous counterpart.

So, with this we conclude so quickly, a decoding gate connected to the outputs of a counter is activated only when a counter content is equal to a given state. For optimal use of a number of flip-flops in counter design n bit counter requires n input logic gate for decoding. So, for mod 8, we have seen that in three input logic gate will be required for mod 16 four input and so on and so forth. So, decoding gates strobed by clock can remove propagation delay related to glitch in asynchronous counter.

In synchronous counter, every flip-flop is triggered simultaneously which avoids the accumulation of delay and possible glitch. Unlike asynchronous counter, a additional logic gates are required in synchronous counter. So, in asynchronous counter, we have seen that output of one flip-flop has fed as clock to the next flip-flop. Here we are putting some additional logic gates in the synchronous counter circuit. And is design of synchronous up down counter is analogues to asynchronous counter, but where we are feeding for up count certain input and for down counts certain other input, and the intermediate logic or the box is to (Refer Time: 31:06) or multiplexer the way it works ok.

Thank you.