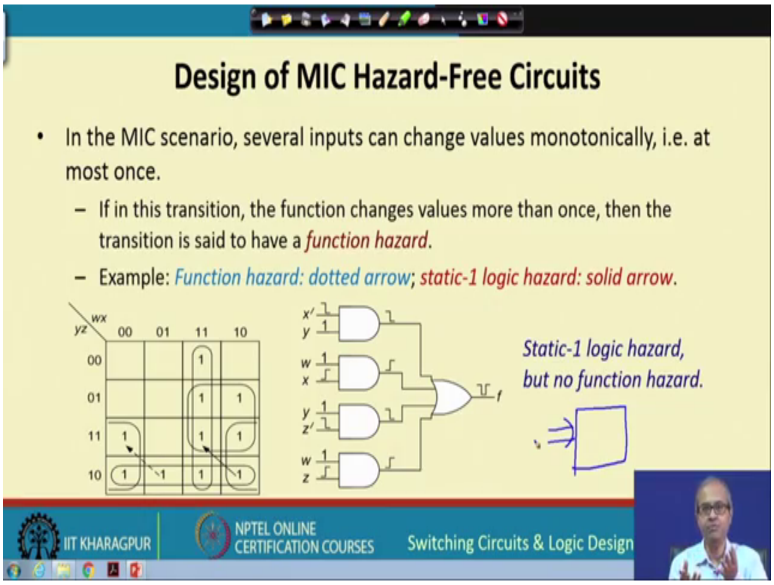
Switching Circuits and Logic Design
Prof. Indranil Sengupta
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture - 53
Asynchronous Sequential Circuits (Part II)

So, we continue with our discussion on Asynchronous Sequential Circuits and Hazards. So, in our last lecture we had some discussion about the different kinds of hazards, and particularly the; I say hazards we saw how to avoid that with respect to Karnaugh map. So, we continue with the discussion in this lecture. This is Asynchronous Sequential Circuits part II.

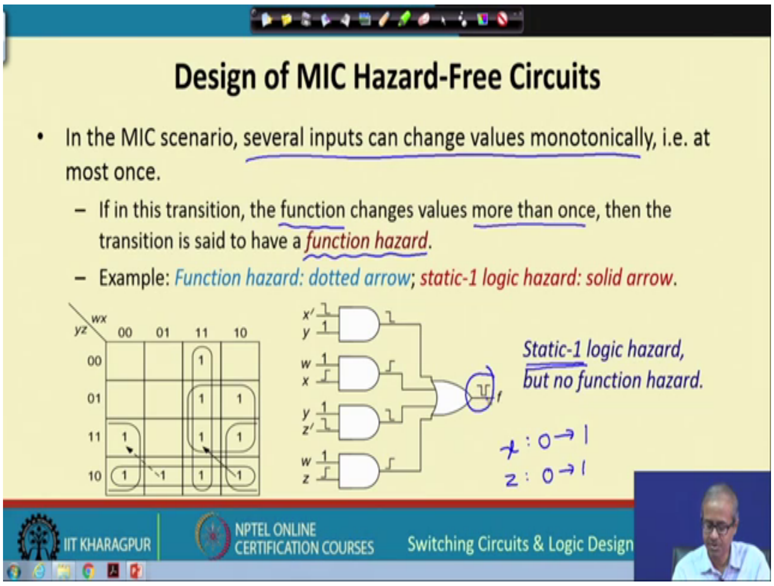
(Refer Slide Time: 00:39)



Now here we start by talking about multiple input change kind of situations, and how such hazards can be handled. Now one thing let me tell you here, when multiple inputs are changing then in addition to logic hazards you can also have function hazards. For that mean suppose I have realized a function, there are some inputs, and more than inputs can change.

Now, when more than in one input can change, the order in which the inputs will finally, change will depend on the delays, their variations. So, there can be multiple scenarios. So, if you look at the function at the functional level. At the functional level when this multiple inputs are changing, you can have this function changes occurring as a hazard, with respect to the function. So, even if you forget about the delays, if the input changes are not happening simultaneously, such kind of hazard situations can occur right. Like in this diagram I have shown here, in this diagram have shown that for multiple input changes there can be, there can be a hazard scenario. Now, we talk about a method for avoiding hazards in such kind of an MIC scenario when multiple inputs are changing at the same time.

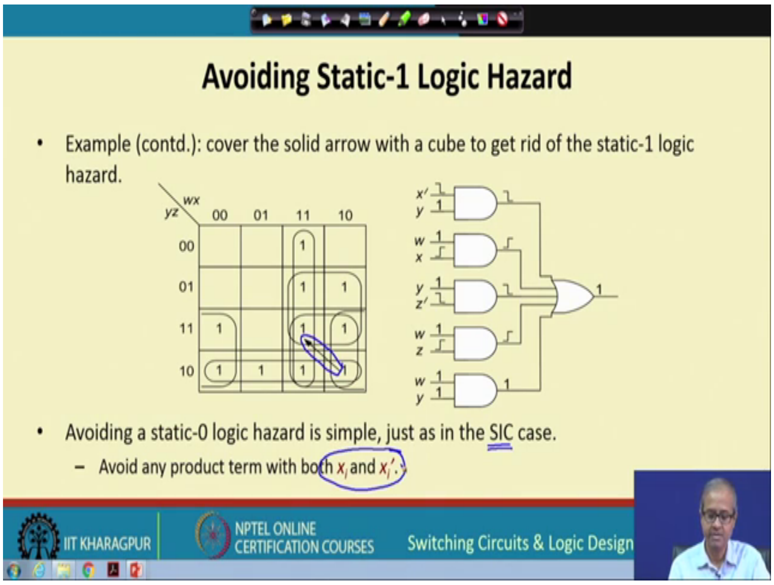
(Refer Slide Time: 02:35)



Let us see, several inputs can change values monotonically. Monotonically means they do not change more than once, at most once, but they may change one after the other. Because, of this as I have just now said, the function value may change more than once which may lead to a function hazard. Like, here in this Karnaugh map I am showing a 4 variable function, the true mean terms are shown. And, some of the changes are shown here you see in this realization; there are some cubes, and then implementations. You see here we have say, the value of x is changing from 0 to 1.

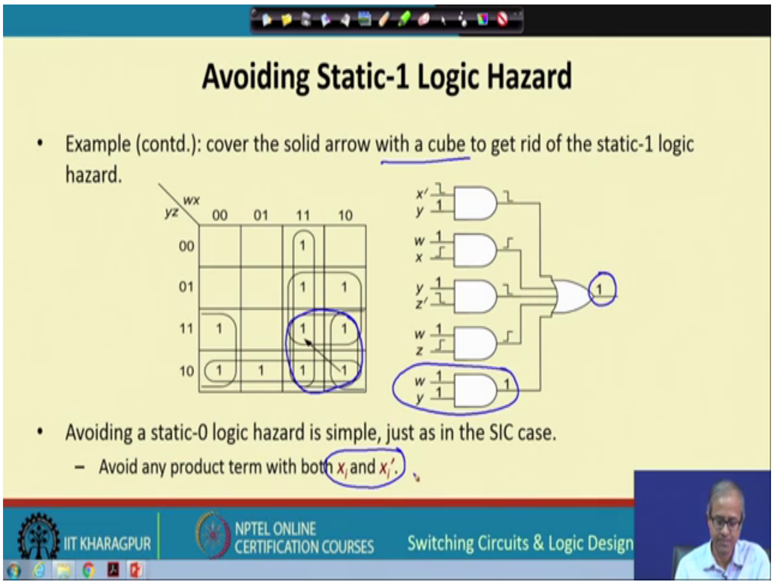
The value of z is change or is also changing from 0 to 1. So, 2 inputs are changing so, the transitions are shown. Now here again if you look into the unequal delays of the gates; so, I suggest you can work out that under what scenario it can happen. There can be a glitch occurring in the output, which is a static 1 logic hazard. But in this case there will be no function hazard. Function hazard will occur in case of this dotted line. If you show, but here we are not showing here we are only showing static 1 logic hazard, where it was 1 remains 1, but in between there is a glitch, right.

(Refer Slide Time: 04:31)



So, how to avoid such static one logic hazard? Here in this Karnaugh map it is represented by this arrow this change. So, both x and z are changing from 0 to 1, right. So, what we do just like this single input change case. So, you should avoid any product term that contains both x i and x i bar as the inputs. This is the trivial case I told earlier, right. So, what we do here is that, we cover the solid arrow by a cube to get rid of the hazard.

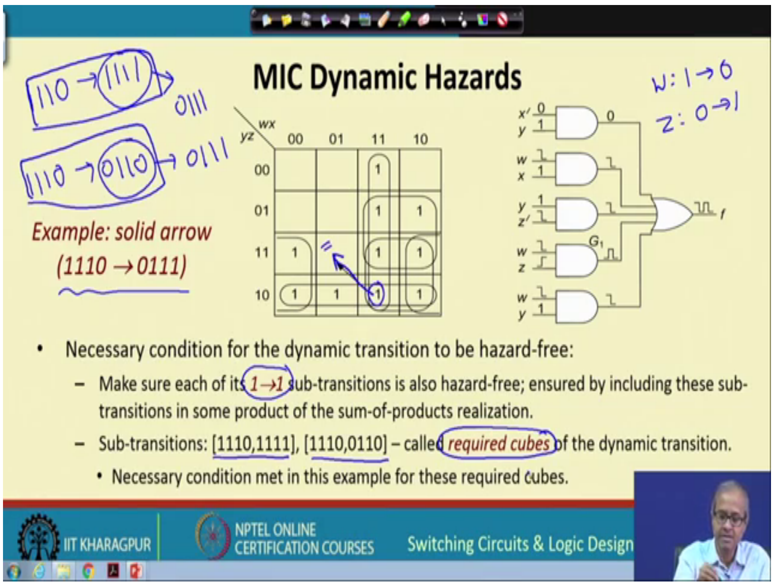
(Refer Slide Time: 05:21)



You see here we have used a cube like this. This is an addition cube we have added. And what does this cube indicate? Indicates w y so this last gate. So, if you add this one additional gate, in the earlier case there was a static 1 hazard coming. You see, now there will be clean output coming no glitch, right.

So, for static 1 hazard this kind of a thing will come, but for static 0 hazard you need not have to worry, just like in the single input change case, because static 0 hazard will occur only for this hypothetical scenario; which normally you will never do fine.

(Refer Slide Time: 06:13)

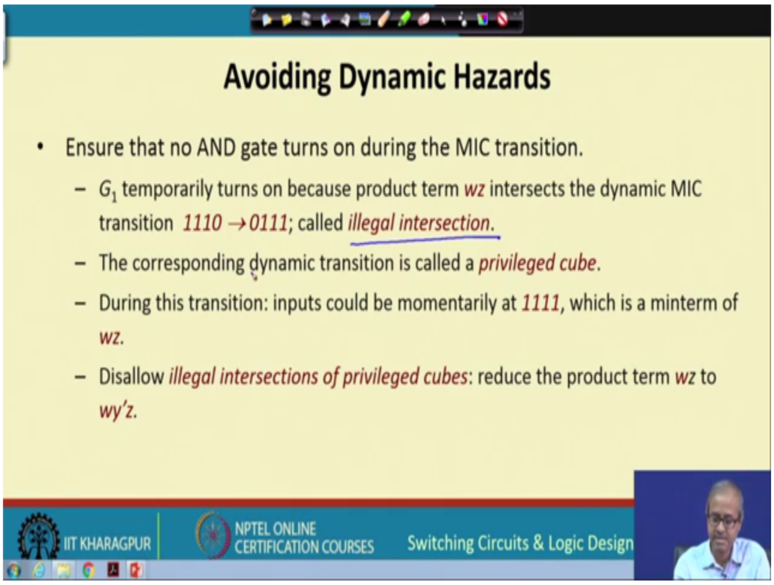


Now, talking about dynamic hazards in multiple input changes, here again we show a function in Karnaugh map, and you show this arrow. This arrow corresponds to this transition. It was in 1 1 1 0, this was 1 1 1 0. The input is changing to 0 1 0 1, this one is 0 1 0 1. Now such a dynamic transition will be hazard free, there are certain conditions to satisfy necessary conditions.

See the function was on because this was not a true mean term. So, if you make this transition so, for 1 1 1 0 it should be 0. So, you see you look at the sub transitions, you would have to ensure that all these sub transitions are hazard free. You look at this implementation, here you have a transition w going from 1 to 0. You have a transition z going from 0 to 1. You see w and z both are changing here, right. Now, for this, we look at the sub transitions; as I said to go from 1 1 1 0 to 0 1 0 1. So, you can have 1 1 1 0 to 1 1 0 1; like again let us see from 1 1 11 0 you can go to 1 1 1 1.

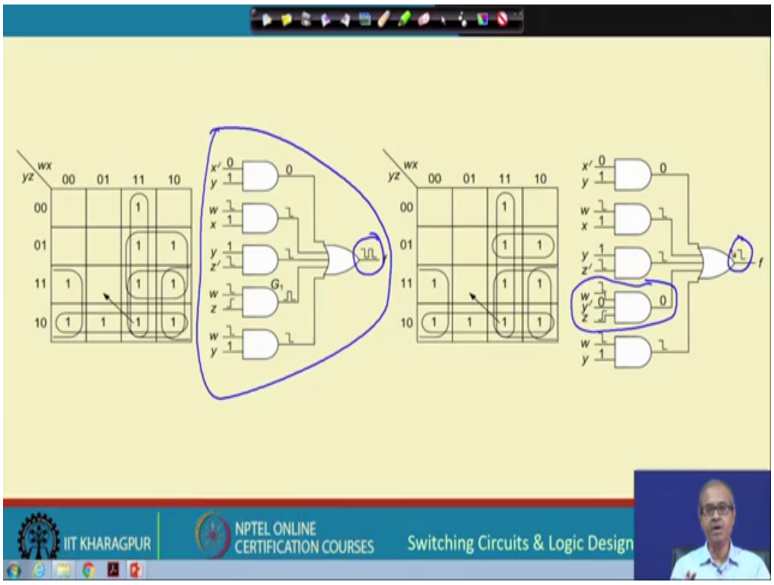
Then you can go to this 0 1 1, this is one possibility. Or you can go to 0 1 1 0, 0 1 1 0, then 0 1 0 1, one input changing at a time. So, there are 2 possible intermediate temporary states you can go through. Those will be the sub transitions I am talking about. So, what we are saying is that, to make an MIC dynamic hazard free, you identify all these sub transitions; like, here I have identified the sub transitions, I said that one sub transition can be this, another sub transition can be this. It leads to a temporary state, and all these sub transitions must be hazard free separately. They must correspond to required cubes. So, if you ensure this, then your implementation will be dynamic hazard free.

(Refer Slide Time: 09:07)



Like as I had said this 1 1 1 1 0 to 0 1 0 1, that makes 2 different transitions, this is sometimes called illegal intersection. And illegal intersection you avoid by just said by looking at the sub transitions, and just adding additional cubes; like I will show in an example to easier.

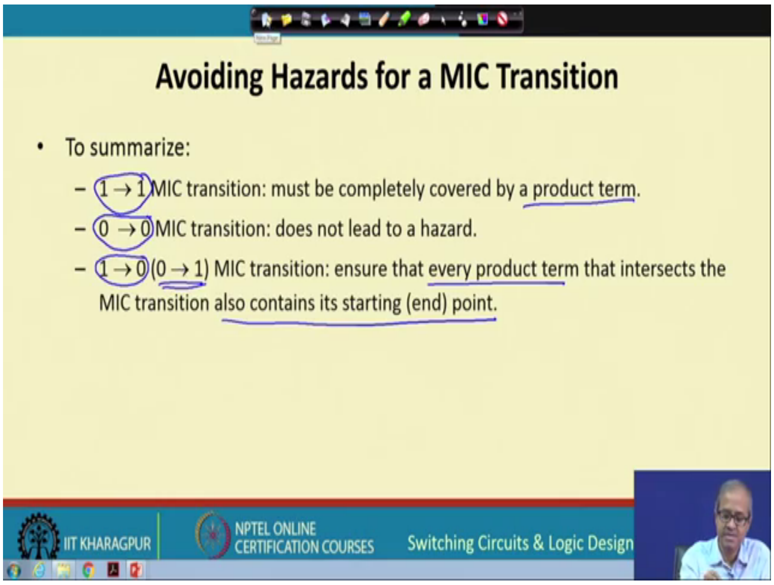
(Refer Slide Time: 09:43)



So, here as you can see, here you have added the cubes. This is, this was your original implementation; which was resulting in a dynamic hazard in the output ok. Now here what I am doing? Instead of this w z, here I have modified this. So, with this modification, you see in the output I am having a clean transition. So, the idea is this you must prevent some of the transitions from happening, if you can do this, then the dynamic hazard in the output will be avoided.

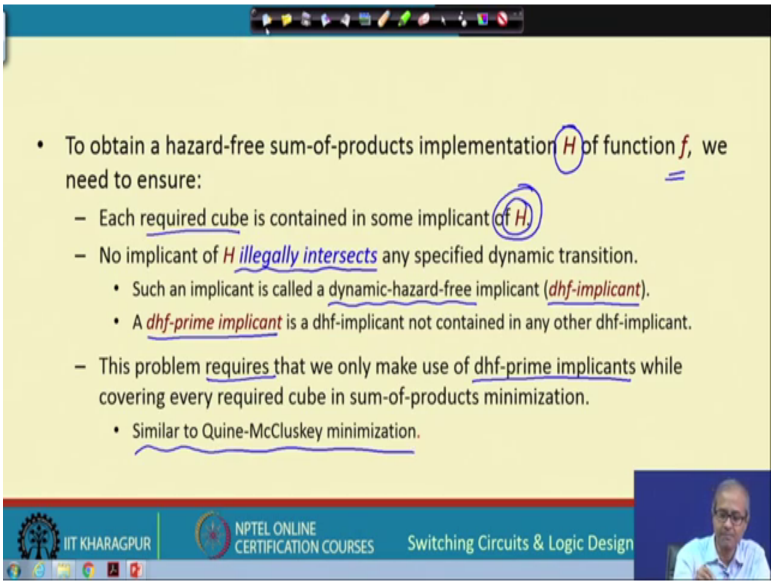
Well, I am not going into too much detail of this, just trying to give you an basic idea; that this is how it is done, but one thing I think you are appreciating, now that design of asynchronous circuits making them hazard free is much more complicated. You will have to look at all possible scenarios for hazards, and then you will have to add additional gates. You see, I am looking at Karnaugh maps here. But think of larger functions, how to handle larger functions? Very difficult, right?

(Refer Slide Time: 11:09)



So, that is the trouble here. So, to summarize for a multiple input change transition, if we have a 1 to 1 transition so, the q must be completely covered by a product term as we had said, 0 to 0 transition in the practical case will never happen. So, forget it, but for one to 0 or 0 to 1 transitions, this is the last example you took. We have to ensure that every product term, that intersects with the transition must also contains it starting for the first case and ending for the second case points. Then such hazards can be avoided.

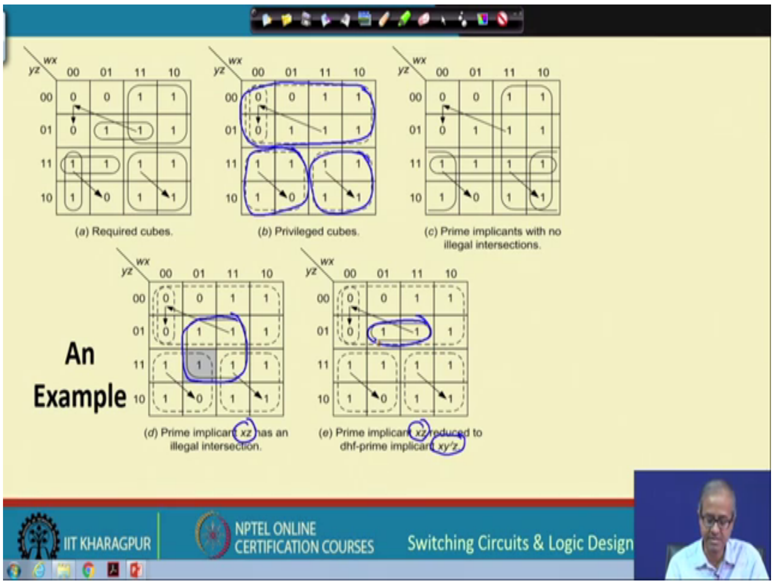
(Refer Slide Time: 11:55)



So, to obtain a hazard free implementation of a function f, let us say the hazard free implementation we call H, the condition is that each required cube, required cube must be contained in some of the implicants of this hazard free implementation. No implicant of H must illegally intersect the dynamic transitions. Such an implicant is called a dynamic hazard free implicant; that which does not illegally intersect, and in short I call it as dhf implicant, dynamic hazard free implicant. A dhf prime implicant just like the definition of a prime implicant is a dhf implicant, not properly contained within any other larger dhf implicant, ok.

So, here we require that we make use of dhf prime implicants only, and we need to cover all the required cubes; which is quite similar to Quine-McCuskey that tabular method of minimization. So, we are giving one example to just illustrate the idea, again not going into too much detail ok.

(Refer Slide Time: 13:29)

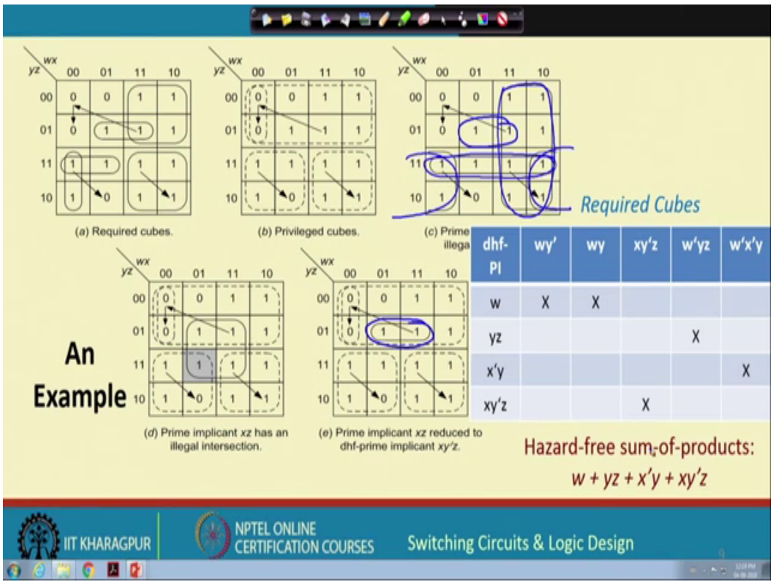


So, let us take an example here. Suppose, we have a function here where the true mean terms are shown and also; the required cubes are shown the required cubes are all shown here. And from the required cubes, you identify the privileged cubes; like, there was a transition, you include this transition also in the cube we make it a larger cube.

Then there was a transition here, a transition here, you make everything included, this transition can be smaller, but to include everything, you make it itself a large cube, include all transition inside it. And this last transition was already inside it. So, there is no problem, these are the privileged cubes. And the prime implicants with no illegal intersections, you can also compute the prime implicants from the original true mean terms. You see these are the prime implicants which are shown 3 of them, they are no illegal intersections with these transitions as per the definition.

Now, if you look at a cover of this, the prime implicant x z has an illegal intersection, what is x z? X z is this and z is this is x z. This x z has an illegal intersection with one of the privileged cubes; such illegal intersections are not allowed, right. So, if we have such an illegal intersection, what you do? You reduce the size of this cube, such illegal intersections are not allowed, this x z you make it x y or z this so that this intersection with privileged cube is not there anymore. This is the basic idea, ok.

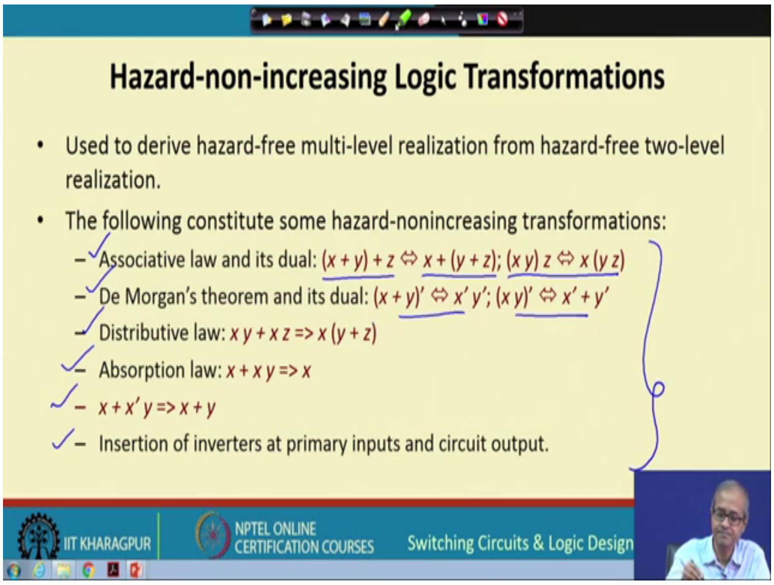
(Refer Slide Time: 15:51)



Now once you do this, then you have the dhf prime implicants, there are 4 here, you can check w y bar w y x y z bar; 5 in fact, w bar y z and w bar x bar y. And these are the coverings, these are the cubes, and these implicants are covered like this.

Now, you see all of them are essential, in these columns all of these check marks are single. So, all these product terms must be there in the hazard free sum of products implementation. So, all this product terms w, this is w, you say this large one was w. This w was this. This was w right. Then y z, y z was this one. This one was y z, x bar y was, x bar and y this, this was x bar y. And finally, x y bar z, x y bar z is the one which you have added separately, this one right, this one. These are the cubes and you need to include all of them. If you implement it like this, this will be a hazard free realization this is ensured.

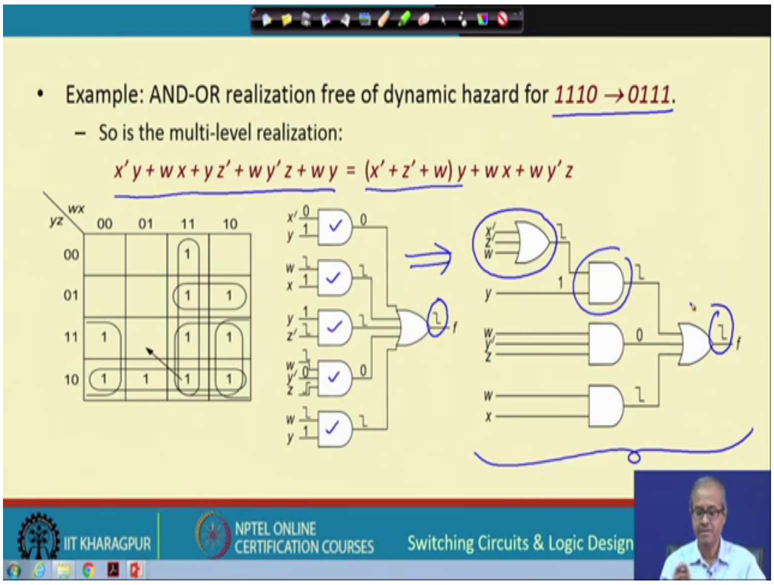
(Refer Slide Time: 17:41)



So and there is another thing this is something called hazard non increasing transformations. Like, suppose you have a 2 level hazard free realization. There are some rules Boolean algebra or switching algebra rules, if you apply them, you can convert this 2 level realization to multi-level realization with the guarantee that the multi-level realization will also be hazard free.

Now without any proof, I am just mentioning some of the transformations that can be used. First one is the Associative law, x plus y z is x plus y, and also it is dual. DeMorgan's theorem you can apply DeMorgan's theorem and it is dual. Distributive law x y plus x z you can take common. Absorption law x plus x x y equal to x and you can also use this law x plus x bar y equal to x plus y. And additionally you can insert NOT gates at the primary inputs and circuit outputs if you want. They will all ensure that if your original circuit was hazard free, and if we apply this rules it will remain hazard free.

(Refer Slide Time: 19:07)



Let us take an example, suppose I have a function; which was implemented in some of products terms like this, there are 5 product terms, and these are the 5 AND gates, they are implementing the product terms. And this implementation is free from the dynamic hazard 1 1 1 0 to 0 1 0 1. This is same example I took earlier. So, in the output there will be a clean transition. Now I am saying that well, I want a multi-level realization, the objective may be to use less number of gates. So, I can take do some factoring, I can use the distributive law, I can apply distributive law, I can take y common from 1 2 these 3 terms, and I can write it like this.

So, I can use an OR gate in the first level to implement this say this sum term, then end it with y. Then other 2 terms remain as it is. So, what the rule says, that if I apply this rules, and I modify this realization in to a multi-level realization, this will also be guaranteed to be free from the dynamic hazard. These are some circuit design techniques or rules available to the designer. So, once hazard free realization is obtained, this can be extended or reused for multi-level circuits, fine.

So, with this we come to the end of our discussion on hazards and asynchronous sequential circuits. So, again I am repeating, I have not gone into very much detail of this, I have not taken too many examples. Just try to give you a flavor of the problem. Design of asynchronous circuits making a circuit hazard free is quite complicated.

Just using Karnaugh map I have given you some techniques, but again Karnaugh map can be used only for up to 4 5 or 6 variables. What happens when the number of variables are more? The problem is really difficult. So, the application of a synchronous circuits are rather limited; even though there are some interesting benefits like higher speeds, lower power consumption and so on, but there are some genuine design problems. So, we shall be talking about a higher level sequential circuit description called ASM in our next lecture; which is also quite useful in designing complex or higher level sequential systems.

Thank you.