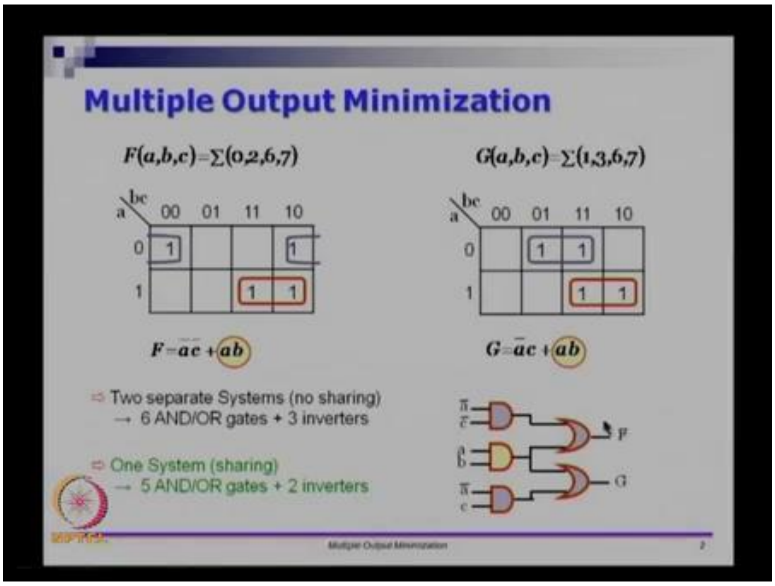
Digital Circuits and Systems

Prof. Shankar Balachandran
Department of Electrical Engineering
Indian Institute of Technology, Bombay
And
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Module - 11
Minimization of Multiple Output Functions

Hi there, welcome to module elven. In this module what we are going to do is, we are going to learn minimization of, what is called, multiple output functions. So, far we have seen several kinds of circuits. So, just the last module I introduced the notion of multiple outputs, till then we had only single outputs. And even in the last module we looked at, for example, BCD incrementer. We had four values W, X, Y and Z and we came up with the expressions, which are individual map, that are made one at a time. We made a map for W, we made a map for X, and so on. So, I am going to now make a case for minimization of multiple output functions, where you consult all the tables together and see if you can minimize the hardware.

(Refer Slide Time: 01:04)



Multiple Output Minimization

$F(a,b,c)=\Sigma(0,2,6,7)$

a \ bc	00	01	11	10
0	1			1
1			1	1

$F=\bar{a}\bar{c}+ab$

$G(a,b,c)=\Sigma(1,3,6,7)$

a \ bc	00	01	11	10
0		1	1	
1			1	1

$G=\bar{a}c+ab$

⇨ Two separate Systems (no sharing)
→ 6 AND/OR gates + 3 inverters

⇨ One System (sharing)
→ 5 AND/OR gates + 2 inverters

Multiple Output Minimization 2

So, let us look at this example. Let us say I have two outputs. So, I have 3 input bits, that are coming in, namely a, b and c and let us say I have a function. So, I have two outputs coming from the circuit one, one output called F, which is the sigma of 0, 2, 6 and 7 and another input called G, which is the sigma of 1, 3, 6 and 7 and we can go and put them in the K-maps. So, we, we have it here. So, one on the left side is for, sigma 0, 2, 0, 2, 6 and 7 and one of the right side is sigma of 1, 3, 6 and 7.

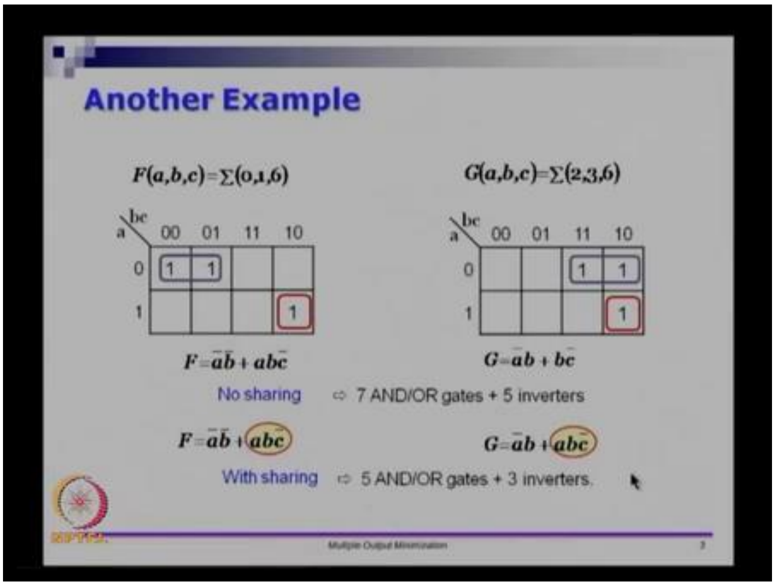
Now, you want to minimize this. This is sum that you can do for F. This is a bar, c bar plus ab. So, this is the term, which is a, b and this term is a bar, c bar. And for G, this term is ab and this term is a bar c. So, we can go and implement them separately. We can implement them as two separate systems, one circuit for implementing F and another separate circuit for implementing G. If you did that then we would have needed overall 6 AND or OR gates plus 3 inverters.

So, for example, you would not need inverter for a bar here or c bar here. You would have needed another inverter for a bar here because we did not share anything. We designed this as two separate circuits, then you would have needed one, 1 AND gate here, 1 AND gate here, 1 OR gate here. Similarly, 1 AND gate here, 1 AND here, 1 OR gate here. We have needed 6 AND or OR gates plus 3 inverters. But from the expression you can see, that ab is actually common to both F and G, right. So, it is probably apparent you because even in the minterms we have 6 and 7, which are in the, which are common to F and G. So, this 6 and 7 as in in turn resulted in one term called ab, which is common to both, right.

If we actually allow for sharing, then we can do something better. If you notice, that ab is being shared, then we can do something better. So, there is a circuit, which allows for sharing. So, in the top we have the function F in the bottom, we have the function G and you have a bar, that is given as input here and c bar given as input here. So, that is a bar, c bar and this is ab. So, a bar, c bar plus ab is derived as F and a bar c plus ab is derived as G. So, we do not have two copies of the AND gate. So, what is not shown is, that a bar itself can be shared using just one inverter. We do not need two inverters for the expression on the left hand, expression on the right we can use just one inverter. The inverter is not shown in this circuit. So, if you throw in the inverter also, then overall we would have needed 5 AND/OR gates plus 1 inverter.

So, with sharing you would have needed 5 AND/OR gates plus, oh, so, it is 2 inverters. c also needs an inverter because of c bar. So, we need 2 inverters, sorry about that. So, we need 2 inverters plus 5 AND/OR gates. So, clearly there is a reduction in the number of hardware circuits that we are using. So, 1 AND or OR gate less and 1 inverter less. Again, remember, in terms of cost this has lesser cost and it has exactly the same functionality as designing it has two separate circuits. So, clearly this is useful and we would like to do this in a more systematic manner. So, this is a simple example. We would like to do this in a more systematic manner. Let us see another example.

(Refer Slide Time: 04:43)



Another Example

$F(a,b,c)=\sum(0,1,6)$

a \ bc	00	01	11	10
0	1	1		
1				1

$F=\bar{a}\bar{b}+ab\bar{c}$

$G(a,b,c)=\sum(2,3,6)$

a \ bc	00	01	11	10
0			1	1
1				1

$G=\bar{a}b+b\bar{c}$

No sharing ⇨ 7 AND/OR gates + 5 inverters

$F=\bar{a}\bar{b}+ab\bar{c}$ $G=\bar{a}b+ab\bar{c}$

With sharing ⇨ 5 AND/OR gates + 3 inverters.

Multiple Output Minimization 3

So, in this example what we have is, I have given you the truth table and the expressions here. So, we have two 1s here and a loner 1 here. So, this gives us the term. So, this gives us the term a bar b bar and this gives us the term a b c bar. So, here we have two groupings of two each. So, this vertical one gives us the term b c bar and this horizontal one gives us the term a bar b. So, now if you go and look at these two, it is not quite apparent whether there is anything common at all.

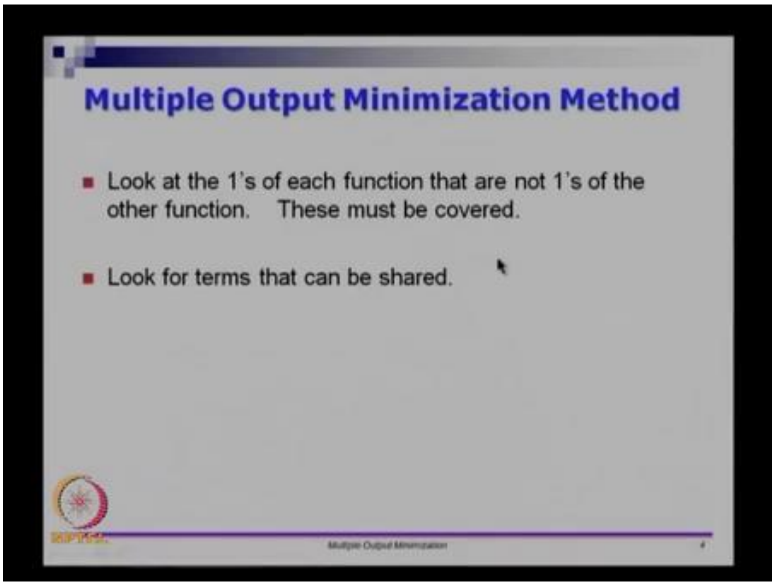
You notice here, at least in the previous one, there was a term ab, which is common to both here. We have a bar b bar. You do not have a bar b bar on this side. Then you have a b c bar. You do not have a b c bar on this side, instead we have a b c bar, right. So, it is not, it is not immediately clear where the sharing is coming from, right. So, if you do not do any sharing, you will need 7 AND OR gates plus 5 inverters. So, you will need 3 inverters for this side, 2 inverters for this side and you will need 7 either AND OR gates. So, in fact, you need 2 OR gates and 5 AND gates, but you can do something slightly better.

So, you notice, that we earlier had grouped these two together, right. We had grouped these two together resulting in a term called b c bar. So, now, if you do not group it and if you leave it as it is, you would have, you would have got a bar b for this term and this should have remained as a b c bar, right, and this is interesting.

So, when I talked about K-map, I always said, you should go and maximize the number of 1s that you are grouping. And argument is, if you maximize the number of 1s that you are grouping, the number of literals go down and therefore, the number of AND gates will go down if you are doing a product, sum of products. So, if I group eight 1s together, then I am knocking off variables, right. So, this is something that we saw earlier. We will knock off variables to this group's eight 1s together.

However, here I am making the case for not, not doing maximal grouping. So, this one was not grouped, this one, instead this if you keep it as a loner, then you have a bar b bar plus a b c bar for this one and for, for this one you have a bar b plus a b c bar. Now, with sharing you can see, that first of all the inverters go down in number. So, you do not need a bar and c bar on both sides. So, you, you need 3 inverters, one for a, one for b one for c. So, you get 3 inverters and otherwise, this term requires two AND gates. So, remember, a b c bar, it is 2 AND gates, 1 AND gate here and 1 AND gate for this one and this one requires 1 AND gate, this one requires 1 AND gate. So, that is 4 AND gates plus 2 OR gates plus 3 inverters. So, we have reduced the hardware cost by AND gate.

(Refer Slide Time: 07:44)



Multiple Output Minimization Method

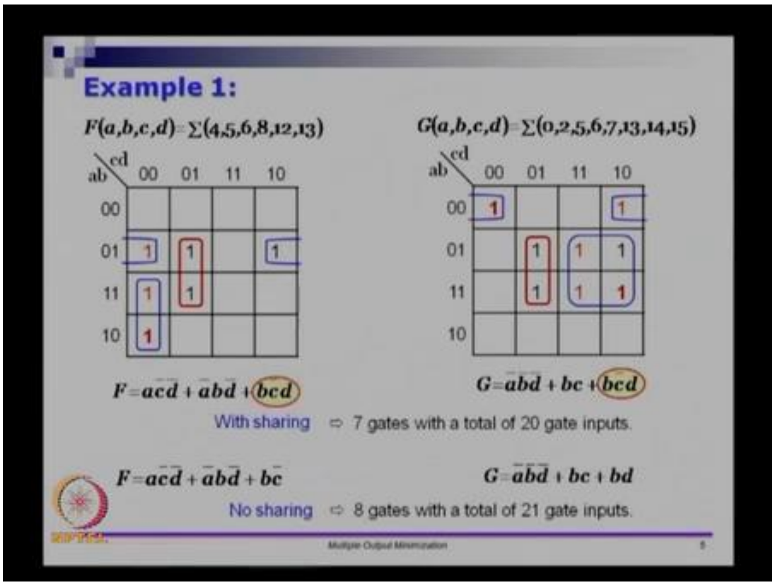
- Look at the 1's of each function that are not 1's of the other function. These must be covered.
- Look for terms that can be shared.

Multiple Output Minimization 4

So, the general philosophy of doing multiple output minimization is, look at the 1s of each function that are not 1s of the other function. These must be covered. So, you go to this example, you look at 1s. For example, these two 1s here are not here and these two 1s here are not here. So, there is possibly no sharing, that you can have with them anyway, right. So, this term is unique is only for F and this term is only for G. You cannot do anything about them, you have to have those terms anyway. So, that is the first bullet.

Look at the 1s of each function that are not 1s of the other function. They must be covered and then start looking for the terms that can be shared. And when you do that, ((Refer Time: 04:43)) so for example, you see, that these terms can be shared here and here. If I had not been careful I would have grouped this. However, if I see that it is been shared, may be just allowing for sharing it is better than cutting down the hardware only for G, but you will still have that number of gates for a. So, it is better to make only one copy of these terms here and leave it as it is.

(Refer Slide Time: 08:53)



Example 1:

$F(a,b,c,d) = \sum(4,5,6,8,12,13)$

ab \ cd	00	01	11	10
00				
01	1	1		1
11	1	1		
10	1			

$G(a,b,c,d) = \sum(0,2,5,6,7,13,14,15)$

ab \ cd	00	01	11	10
00	1			1
01		1	1	1
11		1	1	1
10				

$F = a\bar{c}\bar{d} + \bar{a}b\bar{d} + b\bar{c}d$ $G = \bar{a}\bar{b}\bar{d} + bc + b\bar{c}d$

With sharing ⇨ 7 gates with a total of 20 gate inputs.

$F = a\bar{c}\bar{d} + \bar{a}b\bar{d} + b\bar{c}$ $G = \bar{a}\bar{b}\bar{d} + bc + bd$

No sharing ⇨ 8 gates with a total of 21 gate inputs.

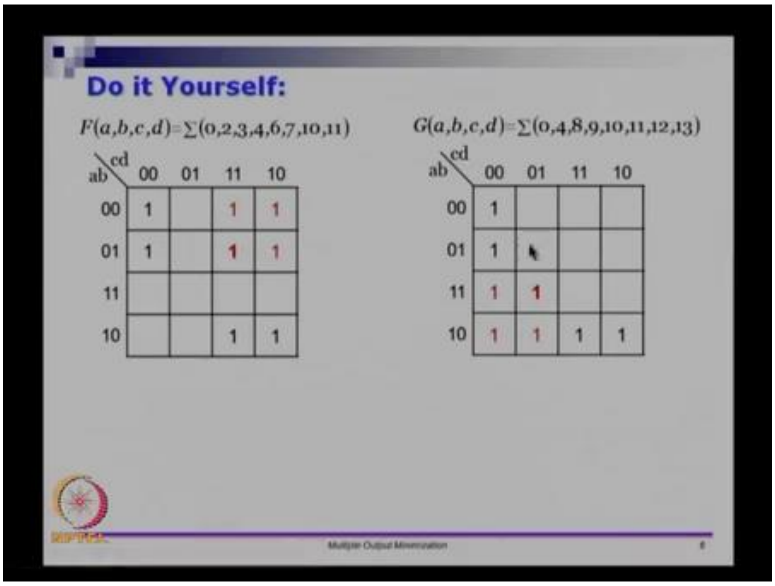
Multiple Output Minimization 5

So, a few more examples here. So, in the left side we have a function on four variables, a, b, c and d and on the right side you have function on four variables again on the same way, same into a, b, c and d, two different sets of terms. And if you notice, you have these 1s is red, the red colored 1s are unique. So, these three 1s appear only in F and these six 1s, 1, 2, 3, 4, 5, so these five of them appear only in G. They are, they are unique only to that their respective outputs. So, what we can do is, we can start with the red ones first.

So, these are red one and these two are red, we will start grouping the red ones first. So, you have all these 1s and these two taken care of. Similarly, these and these taken care of. So, you take, you do maximal grouping if you can for the red, red ones here. All the red ones are now taken care of. They are, they are peculiar or unique only to those and after that you are still left with two of these 1s. And if you notice, these two 1s are also in the same position in G. So, I do not have to make a separate circuit for this term and separate circuit for the same term in G. I can insert, share that circuit.

So, what we have is, in terms of an expression we have a c bar d bar, which comes from one of these terms and a bar b d bar, which comes from the other term, but the common red term is b c bar d. On the right side we have a quad here, that gives us bc. We have group of two, which gives us a bar b bar d bar and we have bc bar d, which is common. So, with sharing we actually get 7 gates and a total of 20 gate inputs. So, the number of inputs that come into the gates is actually 20. So, ignore this for a while. Now, you need only 7 gates in this. So, this is a very useful thing to have. So, without sharing you would had 8 different gates. So, let us just leave this for a while. You can see, that the b c bar d being shared is useful.

(Refer Slide Time: 11:11)



Do it Yourself:

$F(a,b,c,d)=\sum(0,2,3,4,6,7,10,11)$

ab \ cd	00	01	11	10
00	1		1	1
01	1		1	1
11				
10			1	1

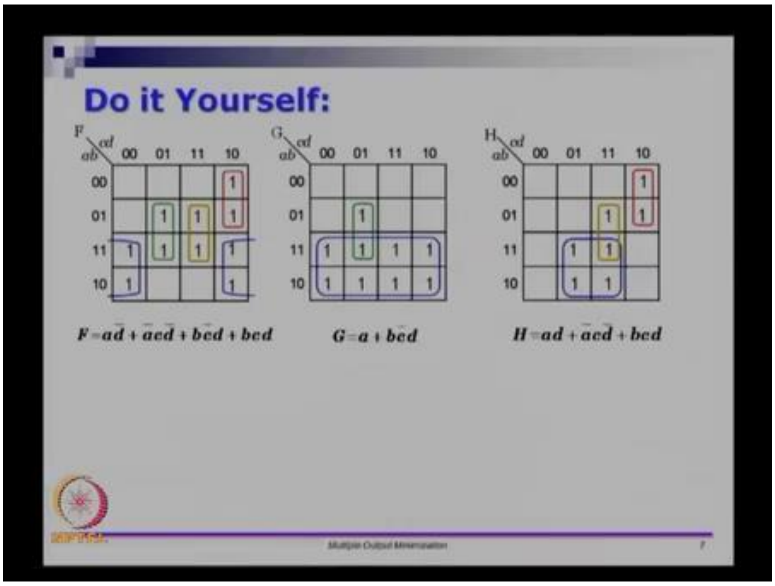
$G(a,b,c,d)=\sum(0,4,8,9,10,11,12,13)$

ab \ cd	00	01	11	10
00	1			
01	1			
11	1	1		
10	1	1	1	1

Multiple Output Minimization 6

So, these are some do it yourself excises with some marking already made. So, we have two sets of minterms here and the 1s in red here and the 1s in red here are unique to F and G, but the ones in blue are actually common across F and G. I want you to go and think about how you will do minimization for this.

(Refer Slide Time: 11:32)



Do it Yourself:

F: ab \ cd	00	01	11	10
00				1
01		1	1	1
11	1	1	1	1
10	1			1

G: ab \ cd	00	01	11	10
00				
01		1		
11	1	1	1	1
10	1	1	1	1

H: ab \ cd	00	01	11	10
00				1
01			1	1
11		1	1	
10		1	1	

$F = a\bar{d} + \bar{a}c\bar{d} + b\bar{c}d + bcd$

$G = a + b\bar{c}d$

$H = ad + \bar{a}c\bar{d} + bcd$

Multiple Output Minimization 7

So, there is another exercise, which is on three different outputs. This is slightly more tricky. So, you have to now see what is common across several outputs. So, earlier we said, go and look at what is shared between F and G, now we have three different outputs and the truth table is already given, I am also going to give you the groupings. So, the various groupings are given to you, right. These are the groupings that are possible and I would like you to go and think about what you want to share across what and so on. So, there is some color scheme that is used here for sharing. So, I would like you to go and think about why that color sharing makes sense and what the, what is the impact of that grouping, go on and count the number of gates that you have for each one of them.

So, in, in essence what we have done in this lecture is, we looked at the case for sharing terms across different outputs in what are called multiple output functions. So, minimization of the terms in the presence of multiple output functions is not an easy task. So, sometimes if you share, it is better; sometime if you share, it is actually going to be worse. So, you have to be careful about when to share and when not to share. In general, beyond a point this becomes very cumbersome for human beings to actually go and do, what is called, optimization.

So, if you want to go and look at the smallest number of gates that is needed to implement these three input, these three four input, three output circuit, it may not be possible at all, right. We may not be able to get the optimal one for every single combination of three outputs and so on. So, instead what we actually start looking for is, we start looking at synthesis of circuits. Synthesis means, we want to create a gate level description of these truth tables, however we may or may not have the smallest number of gates. So, this getting the smallest number of gates is not always possible. This is actually hard even for single output case. But even when you have multiple outputs, this problem is not easily solvable. It is a hard problem to solve. Instead we go and try and minimize as many as possible and not necessarily look for the optimal number of gates or the absolutes smallest number of gates. We may not look for that number. This brings me to the end of module 11 of week 2 and in module 12 we are going to see, so we will conclude this week with module 12.

Thank you very much.