

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

Lecture - 13
Cost Criteria and Minimization of Multiple Output Functions

Hello everybody, we are in lecture 13 of this particular course. In previous classes, we have seen how to minimise the logic expression and get a implementation for single input cases.

(Refer Slide Time: 00:32)

CONCEPTS COVERED:

- Cost Criteria
- Multiple Output Minimization
- Use of Don't Care in Multiple Output Minimization

THE GREAT EDUCATOR
swamyam
INDIA'S CHOICE

So, in this particular class, we shall look at multiple output minimization. And we shall see that there are very many different possibilities, and we have to weigh one option against the other for which we need to define certain cost criteria. So, we shall begin this particular lecture by defining certain cost criteria and then we shall look into the minimization aspect.

(Refer Slide Time: 01:00)

Simplest SOP and POS: An Example

A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

$Y = F(A,B,C)$
 $= \sum m(2,4,5,6,7)$

$Y = A + B.C'$

$Y = (A + B).(A + C')$

Here, realization of POS is costlier.

Y = F(A,B,C) = ∑ m(2,4,5,6,7)

Y = A + B.C'

Y = (A + B).(A + C')

Here, realization of POS is costlier.

So, the kind of implementation we had seen before we start from there. So, this is a representation of a truth table where the minterms are 2, 4, 5, 6, 7 ok, this is familiar we had already solved this particular kind of minimization problem before. And in the SOP case, we have the realization where one minterm comes from here which is A, and another minterm comes from here which is B C prime ok, this we have seen before

So, in the SOP realization we have Y is equal to A plus B C prime in this manner right. And for the same truth table, we can have a POS representation for which we are having two such sum terms one is A plus C prime, another is A plus B ok. Now, if you want to realize this in the form of a circuit for each of the operation, we have corresponding logic gate available. So, we consider that the inverter inverted input complemented and un complemented, both type of input is available if it is not the case we can consider it separately. So, we have a two level logic implementation.

So, in this case, we are having two gates; one gate is the OR gate combining A, and the other A and BC prime. Another gate is an AND gate which is denoting BC prime fine and compared to that we are having A plus A or B generated by this one A or C prime generated by this one. And this and gate is combining these two outputs of the OR gates ok.

So, if you look at the complexity and say that which one is more costly costlier implementation, of course, this one is the POS is the cost is a costlier implementation in

this case it is very clear very evident here is not it. So, we look at the number of inputs and you know number of logic gates that are used here ok, but it may not be the case, so simple we need to for more complex circuit specially for multiple output minimisation. So, for which we look at defining certain cost criteria.

(Refer Slide Time: 03:32)

Cost Criteria

- Literal Cost:** The number of times a literal appears in complemented or uncomplemented form (L).
- Gate Input Cost:** The number of inputs to the gates in the implementation. It may not count NOT gates (G) or it may include NOT gate count (GN).

If the number of terms excluding single variable term is T , and the number of distinct inversions is N , then

$$G = L + T$$

$$GN = L + T + N$$

Implementation	L	T	N	G	GN
$Y = A + B.C'$	3	1	1	4	5
$Y = (A+B).(A+C')$	4	2	1	6	7

So, one definition is that of literal cost. So, literal cost is the number of times a literal appears in complemented or uncomplemented form in the Boolean expression ok, so that is what is known as literal cost. So, if we look at the previous example, so how many times it is appearing A, B and C prime complemented or uncomplemented, both are you know to both are to go as literal cost, so 3 right. In this case, A, B, again A, again C prime, so four times literals are appearing. So, literal cost is 4 that is how the literal cost is arrived at.

Next is gate input cost. So, compared to literal cost, gate input cost gives a better appreciation of the complexity of the circuit that is getting implemented ok. So, in the gate input cost, the number of inputs to the gates in the implementation that is getting that gets counted. So, in the previous example, in this example, how many this SOP realization how many is the gate input cost, so 1, 2 over here, another here. So, these are the 3, this is 3. And what is the gate input cost for this POS representation 1, 2, 3, 4, 5, 6. So, these are 6 ok.

So, here we are not counting the inverter NOT gate ok. So, if you count the NOT gate,

then we get another measure one is defined by G another is defined by gate input cost including NOT gate count ok. So, in each of these cases see one inverter will be coming here. So, this will become 3 plus 1, and this will become 6 plus 1 ok. If we consider the inverter or if we do not consider the inverter, these are the two cases that is possible.

Now, we can very quickly without going for the diagram and getting the implemented circuit logic circuit in place right from the expression, we can get the gate input cost very easily by considering these two equations ok. See, if the number of terms excluding single variable term ok, excluding single variable term, because it does not require a gate is T . And the number of distinct inversions is T , number distinct inversions means that if a bar is used in two places, it will be only counted once.

A bar as literal is coming in two places distinct inversion required only one inverter one NOT gate is required, so that is the distinct that is defined by distinct inversion. So, if that number is T ok, then the gate input cost without considering the NOT gate count is literal cost plus number of terms ok. So, we can see what is the gate input cost by and literal cost by from this formula. So, in this example without looking at this gate, so literal is 3, 1, 2, 3, ABC prime, and number of terms excluding single variable term. So, excluding single variable term this is a single variable term which have to exclude. So, number of term is only one. So, 3 plus 1 – 4 is the gate input cost without not gate count. Is it fine? Right.

So, and we can count here 1, 2, 3, 4, four gates are there four gates inputs are there, in the other case how many literals are there four already we have seen right and how many terms are there are two terms one term one sum term another is another term sum term. So, 4 plus 2 – 6 is the gate input cost, and you have counted that 6, 6 is the gate input cost, so that is how we can get just by looking at the expression without going for a physical realization of the or drawing of the logic circuit we can get the gate input cost ok. And if we want NOT gate count additionally in addition then we shall at the number of distinct inversion for this ok. So, this is put in a tabular form here. So, this is 4 and this is 6 right, and 5 and 7, if we include the NOT gate count. So, mostly it is gate input cost that is considered in the study.

(Refer Slide Time: 08:20)

Cost Criteria: Examples

Implementation	L	T	N	G	GN
$Y = A.B + B.C + C.A$ ✓	6	3	0	9	9
$Y = (A+B).(B+C).(C+A)$ ✓	6	3	0	9	9
$Y = A.B.C + A'.B'.C'$ ✓✓	6	2	3	8	11
$Y = (A'+B).(B'+C).(C'+A)$ ✓	6	3	3	9	12
$Y = A.B + B'.C.D + A'.B'.D + B.C'.D$	11	4	3	15	18
$Y = (A+C).(A+D).(B'+C'+D).(B+C+D')$	10	4	3	14	17

Considered: 2-Level implementation, no restriction in fan-in or kind of logic gates to be used (as in K-Map, QM simplification)

Implementation using only 2-input NAND gate: An Example
 $Y = A.B + C.D.E = ((A.B)'.(C.D.E)')' = ((A.B)'.((C.D)')'.E)''$ → Cost: 5 Units (10 Gate inputs)

Total Cost (TC) = No. of gate i/p + No. of gates

So, we look at few more examples. And this realisation and this realisation both are equivalent ok; they represent same truth table ok. Now, we are not drawing any logic circuit for them. So, simply by the way we have calculated using that formula, we shall see what is the gate input cost here. So, number of literals are 1, 2, 3, 4, 5, 6 right, so 6 literals.

So, literal cost is 6. Number of term 1, 2, 3, so 3, and there is no inversion involved. So, n is 0. So, 6 plus 3 – 9 is the gate input cost without NOT gate count, and with not gate count is also 9, because there is no inverter used. And equivalent representation POS represented for this is A plus B and add with the B plus C and add with C plus A. Again number of literals is 6. You can see number of terms is 3 right, and no inversion, so 9 and 9. So, both are equivalent, in terms of complexity any one of them you can pick up ok.

There are again two equivalent representation of same truth table; one is in SOP form; another is in POS form ok. Now, how we see; we see how they are you know compared in terms of the cost that we have just defined. So, literal cost in the first place is 1, 2, 3, and 4, 5, 6, 6 ok; number of terms 2, 1 and 2 right. And inversion required A prime, B prime, C prime, so 3 right. So, gate input cost without not gate count literal cost and number of term 6 plus 2, this is 8. And in the other case including NOT gate count is 11. And for the other one the previous realization, we have got A prime plus B B prime plus C and C prime plus A three OR terms are getting ANDed ok.

So, 6 is the number of literals, three 3 OR terms are there, 3 terms OR there you can see right. And three inversions A prime, B prime, C prime are required ok. So, your gate input cost is 6 plus 3 – 9. And this is the 12. So, of course, now we can consider compare and you can say that this one is a, this one is a less costly realization, is it according to this definition. So, these are few more examples.

So, this is they are not equivalent. Just here number of terms are there 1, 2, 3, 4, 5, 6, 7, 8, 9 for 10, 11 number of literals 11 literals. 1, 2, 3, 4, four terms are there. Inversion B prime, A prime, B prime again and C prime, but B prime will be counted only once distinct inversion. So, it is 3, so that way you are getting 15 and 18 for these cases.

Similarly, for the other and when we do this we consider two level implementation, and there is no restriction in fan-in or the kind of logic gates that is to be used ok; the way we actually minimise circuit in Karnaugh map or QM algorithm using queen McCaskey algorithm. So that is what we have that is what is being considered here. But if there is a restriction in fan-in if there is a restriction in say the kind of gate that you can use, then of course this is not the way we have to calculate this one.

For example, if it is asked so this expression Y is equal to AB plus CD ok. So, this is to be realised. And the restriction put is we have to use only two input NAND gate then what is the cost ok. So now, two input NAND gate becomes one unit right. And we can use De Morgan's theorem right successively and we can see that 1, 2, 3, 4 and 5, so 5 two input NAND gates will be required ok. So, this is the realization.

So, there could be another realization possible realizations. So, we have to compare against them, and depending on number of units we will pickup with the one which is less in cost. And here you see the 10 gate inputs are required, two input NAND gates 5 into 2 10 gates. But if you look at the normal two level implementation with no restriction in fan-in and then the literal cost is A B C D E 5, and number of terms are 2, so it is 7. So, number of gates gate inputs required, if we go for no restriction two level implementation is 7, but because of the restriction we are now having 10 gates, but so and 5 2 input NAND gates used in this particular problem.

Now, what is meant by 2-level and 3-level. So, let us look at one more example, I mean 2-level and more than 2-level. So, here we are realising AB CD and CE, AB plus CD plus CE. So, if you go for 2-level implementation, then how what is the gate input cost 1,

2, 3, 4; 5, 6 6 literal and 3 terms, 6 plus 3 – 9. And you see 9 gate inputs are there 1, 2, 3, 4, 5, 6, 7, 8, 9, 7, 8, 9 right.

And what about this one, we see that if we take C common A B plus C ANDed with D or E, this expression. Then C D or E getting ANDed ok, which is or and then ANDed with C, and then finally this is. So, this is 3-level implementation right, and how many gate inputs are there 1, 2, 3, 4, 5, 6, 7, 8 right. So, instead of 9 we are having 8, so this is for 3-level implementation. So, our discussion is on 2-level implementation the way we usually do for AND-OR or NAND-NAND or for POS OR-AND or NOR-NOR circuit ok.

Now, another point here in some of the literature, we will find that the total cost an another new another definition is given for cost, which is number of gate input that is the gate input cost plus number of gates that are used. So, additionally number of gates that are used that are also put into consideration, which is defined as total cost in some of the literature ok.

(Refer Slide Time: 15:27)

Multiple Output Minimization Common P.I.

BC	00	01	11	10
A	0	1	0	0
A	1	1	1	1

$F_1 = B'C + A.B$

Individual implementation
 $G = 6 \times 2 = 12$
 $TC = 12 + 3 \times 2 = 18$

Individual

BC	00	01	11	10
A	0	1	1	1
A	1	1	0	0

Using common P.I.
 $G = 6 + 4 = 10$
 $TC = 10 + 3 + 2 = 15$

$F_2 = B'C + A'B$

Combined

Now, we come to the multiple output minimization problem and we will keep this cost aspect in our mind. And what we see is that therefore various possibilities. So, in one case the first case we look at the problem, where the between two functions that are to be realized the common term is a prime implicant, prime implicant in both the places.

So, you see this is these are the common terms between these two functions that are to be realized. So, B prime C B prime C is the common prime implicant right. And of course, in this case the decision is obvious. So, if you go for individual realization, so this is what we will be doing, so B prime C over here, again B prime C is again realized over there. And then this is the corresponding circuit, which will require 6-gates 1, 2, 3, 4, 5, 6 6-gates. And the gate input cost for each one of them is 1, 2, 3, 4, 5, 6 1, 2, 3, 4, 5, 6 here also you can count 1, 2 1, 2, 3, 4 for literal two terms 6. So, 6 into 2-12 is the gate input cost.

And if you look at the total cost I mean including the number of gates being used, it is 18. And of course, this B prime C is a candidate, which can be the common term shared term, which can be used for both the places. So, this is the corresponding circuit. In this case to realize this part, you would need six gate inputs 1, 2, 3, 4, 5, 6 and to realise this part which is common you need 4. So, 6 plus 4-10. And the gate count will be total cost will be also 10 3 for this, and the two for the other ones, so 15. So, is very you know simple. In this case, we do not have any issue in coming up with the minimized multiple input you know solution right. So this is the minimized solution.

(Refer Slide Time: 17:38)

Multiple Output Minimization

BC	00	01	11	10
A	0	1	1	0
	1	0	0	0

$F_1 = \sum m(0,1,3)$
 $= A'B' + A'C$

Individual implementation
 $G = 6 + 8 = 14$
 $TC = 14 + 3 + 3 = 20$

BC	00	01	11	10
A	0	0	1	0
	1	0	0	1

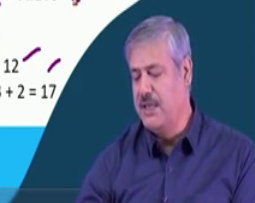
$F_2 = \sum m(3,6)$
 $= A'B.C + A.B.C'$

BC	00	01	11	10
A	0	1	1	0
	1	0	0	0

$F_1 = A'B' + A'B.C$
 $F_2 = A'B.C + A.B.C'$

Common term need to be generated

Combined
 $G = 7 + 5 = 12$
 $TC = 12 + 3 + 2 = 17$



Now, look at a problem where there is no common prime implicant as such ok. So, in this case, so F 1-0, 1, 3 minterms and F 2 has got 3 and 6 is minterm. So, you have got this two terms if you go for individual implementation A prime B prime, and A prime C,

these are the one that are to be realized ok. And in this case this is common right, but as I said in this particular case this is to be realised, but this is not the term is you know there in the other case as a prime implicant, it is already covered by the one of the term A prime C right.

So, individually if you realise, these are the two terms. And if you calculate the gate count in this case gate count cost, so for the first one 1, 2 3, 4 and 5, 6 this is the term. And for the second one 1, 2, 3 4, 5, 6 and two terms 8; so this is the gate input cost, so 14 right. And if you look at the total cost 14, and 3 gates are used here, 3 gates are used here, so total 20 ok.

But, since the common term over here is to be generated you have no choice, because the other term requires it ok. So, there is another way you can realize it, where you are not generating A prime C over here rather this A prime B C, which is generated that is being used ok. And in this case, you have got one common term one shared term, which is you know being utilized by both the expression.

So, in this case how does it help, it helps by reducing the gate count. In the first case, 1, 2, 3, 4, 5, 6, 7 in the second case, 1, 2, 3 right. And they are two terms 4, 5. So, 7 plus 5 12, because A prime B B C is already generated in the first case, right. So, total is 12 and the total gate count gate cost is 12 and this is 17, so if you compare you can see that you are in gain ok. So, it is less costly, so it is to be realised in this manner by this expression not by individual ok.

(Refer Slide Time: 20:33)

Multiple Output Minimization

	CD			
AB	00	01	11	10
00	0	0	1	1
01	1	1	1	1
11	1	1	1	1
10	0	0	0	0

	CD			
AB	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	1	1	1	1
10	0	1	1	0

Individual Minimization
 Combined Minimization

$F_1 = B.C' + B.D + A'.B'.C$
 $F_2 = B.D' + B.C + A.B'.D$

$F_1 = B.C' + B.C.D + A'.B'.C$
 $F_2 = B.D' + B.C.D + A.B'.D$

Common term part of bigger group in which other members already covered

Individual
 $G = 10 + 10 = 20$
 $TC = 20 + 4 + 4 = 28$

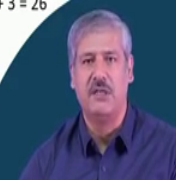
Combined
 $G = 11 + 8 = 19$
 $TC = 19 + 4 + 3 = 26$

✓

✓

✓

✓



Now, let us look at another example, where we have got common term right. But, this common term is already part of bigger groups, which is already covered ok. So, let us not look at this particular example, what is happening over here. So, between these two this is F 1 and this is F 2 ok, so between these two you can see this is this two ones are the common term right.

Now, these two 1's can be part of a bigger group (Refer Time: 21:14) 1, therefore if you go for individual minimisation, this is how you will realize it B C prime B D and A prime B prime C. So, this is this term A prime B prime C. And another one term is B C prime, which is coming from here. And the other one is your B D, so this is the term that we are talking about ok. So, this is how you will get it. And for the second case, this common term can be part of a bigger group ok. So, this is the corresponding realization of it, so B D prime B C A B prime D. So, this is A B prime D ok.

Now, the issue with this is that by making a bigger group, group of four the ones that you are covering, it is already covered by the previous group one term B C prime over here. Similarly, by forming this bigger group, you are covering this one this one is already covered by this you know group of four, which is covered in the blue right. So, there is no necessary there is no necessity of you know grouping them in that manner. And generating two distinct term right, one is B D, another is a B C over here rather you can generate only one of the term this term, which is B C D and which can be shared in both

the places ok.

If you do if you do does it help in anyway, we can see that it helps in a manner, which is depicted here if you if you go for individual you know minimization. So, 1, 2, 3, 4, 5, 6, 7 and three terms 10, and similarly the other one is 10 gate input cost is 20. Whereas, in this case 1, 2; 3, 4, 5; 6, 7, 8 and three terms are there 11 right. In the second case, 1, 2, 3, 4, 5 this is already generated and three terms that is 5 plus 3 eight, so this is 19. So, this is 20 versus 19 ok.

Similarly, if you look at the total count including the realization in some measure, they are considering also the number of logic gates that are used, so this is 28 versus this is 26 ok. So, this is A better you know less costly realization ok. So, this is what we find when the common term is a part of bigger group, where the members are already covered by other prime implicants in individual cases ok.

(Refer Slide Time: 24:06)

Multiple Output Minimization

Use of Don't Care

CD	00	01	11	10
AB 00	0	0	1	1
01	X	1	1	0
11	X	1	1	0
10	0	0	0	0

$F_1 = B.D + A'.B'.C$
 $F_2 = B.C + A.B'.D$

$G = 7 + 7 = 14$
 $TC = 14 + 3 + 3 = 20$

CD	00	01	11	10
AB 00	0	0	1	1
01	X	X	1	0
11	1	X	1	0
10	0	0	0	0

$F_1 = B.C'.D' + B.C.D + A'.B'.C$
 $F_2 = B.C'.D' + B.C.D + A.B'.D$

$G = 12 + 6 = 18$
 $TC = 18 + 4 + 2 = 24$

Individual

$G = 20, TC = 28$

Now, we look at the case where do not care is there ok, and we see that how do not care can be considered in the multiple output minimisation ok. So, we have taken two examples. So, in one example you see that these are the do not cares this is do not care do not care in for function 1, and this is do not care for the function 2 right.

Now, what we can see that this do not cares I mean you can you know come up with a expression like this and this can be a common term the way we had seen before, but it is

meter that if you have I mean a combination of this and combination of these rather than, you know individually going for so because that we generate additional term could get this. What I mean to say is that one option is this one right, and this one, this there are two ones right and then combining with this one, but that will require additional terms to be generated right.

So, this is the complexity of you know group of four, this is the complexity of group of four. And then A third term, which is common of which is of course shared, but which one we will required is C terms, but in this case what we are having these two terms are always required, and you are having only one term, which is you know group of four group of four with same complexity that is only getting generated ok. So, here this do not care cases, we are simply ignoring them and we are going in the manner as at as it has been shown, and by which we are reaching a gate counter gate count cost of 14 and total of 20 ok.

Now, let us look at little bit more you know complex thing, where the do not care cases are this one right ok. So, again there is a possibility of you know in the minimized expression having a group of this a group of this right, similarly a group of this, and a group of this. So, these are the possibilities right. And if you look compare all those possibilities, then you can see that considering this one and do not care over here, and this do not care and one over here, you can generate a shared term is not it. So, this term this group is shared.

And similarly, we can generate between these two a shared term and compared to individual minimisation the way we would have done in this case as I repeat again like this case right, and over here to cover these all these 1s ok. Compared to that if you way that realization of these two, which is shared realization of these two which is shared in this manner ok, you find we will find that the relative cost is coming to in the first place 18 gate input cost, and total cost is 24.

And if you tried individually, you can see it will go for larger value of 20 and 28 ok. So, this is the different way for multiple output minimization using the cost criteria that we have defined, we can look into it and pickup the one, which is favourable. And this was for two output. Similarly you can go extend it for three output and this is more complex, but the concept remains the same ok.

(Refer Slide Time: 28:43)

Conclusion:

- Comparison of different logic circuit implementation is done through cost criteria.
- Literal Cost considers number of times variables appear in an expression, complemented or uncomplemented.
- Gate Input Cost considers total number of inputs to different logic gates.
- A total cost is defined which consider Gate Input Cost and number of logic gates used.
- Consideration of shared terms in multiple functions can lead to an overall minimized circuit with less cost.
- Don't Care conditions are useful in deciding shared terms that can be generated for minimized circuit.

With this we come to the conclusion of this particular class, what we have discussed in briefly. We have seen that comparison of different logic circuit implementation can be done through defining certain cost criteria. We define literal cost, which is considers the number of times a variable appears in an expression either in complemented or in uncomplemented form.

And gate input cost considers total number of inputs to different logic gates. And there is a definition related to total cost, which considers gate input cost as well as number of logic gates that are used. And considered consideration of shared terms in multiple functions multiple output functions can lead to an overall minimized circuit with less cost. And do not care conditions can be useful in deciding shared terms that can be generated for minimized circuit.

Thank you.