

Multiple Output Networks

Definition: A single-output two-stage AND/OR network is a realization using a level of AND gates followed by a single OR gate of a switching function $f(x_1, \dots, x_n)$.

Definition: A k -output two-stage AND/OR network is a realization of k switching functions

$$f_1(x_1, \dots, x_n) \quad \dots \quad f_k(x_1, \dots, x_n)$$

all evaluated simultaneously on any given assignment of $\{0, 1\}$ to the variables x_1, \dots, x_n , using a level of AND gates followed by a level of k (or less) OR gates

Definition: A *minimal* two-stage AND/OR network is a realization which has the minimum number of gates, g , and among g gate realizations the minimum number of connections (input and internal).

Recall the implications of minimality for a single-output network.

- Minimizing gates only affected the number of AND gates which is equivalent to the number of prime implicants used in the sum-of-products expression.
- When an AND gate outputs 1, the network outputs 1, i.e., the product term for an AND gate is an implicant of f .
- The number of internal connections (between the AND gates and the OR gate) was fixed by the gate count optimization.
- The number of input connections was determined by minimizing the number of literals used in the products defining the first level, i.e., the products were *prime implicants* of f .

This changes when considering a k -output two-stage network.

This changes when considering a k -output two-stage network.

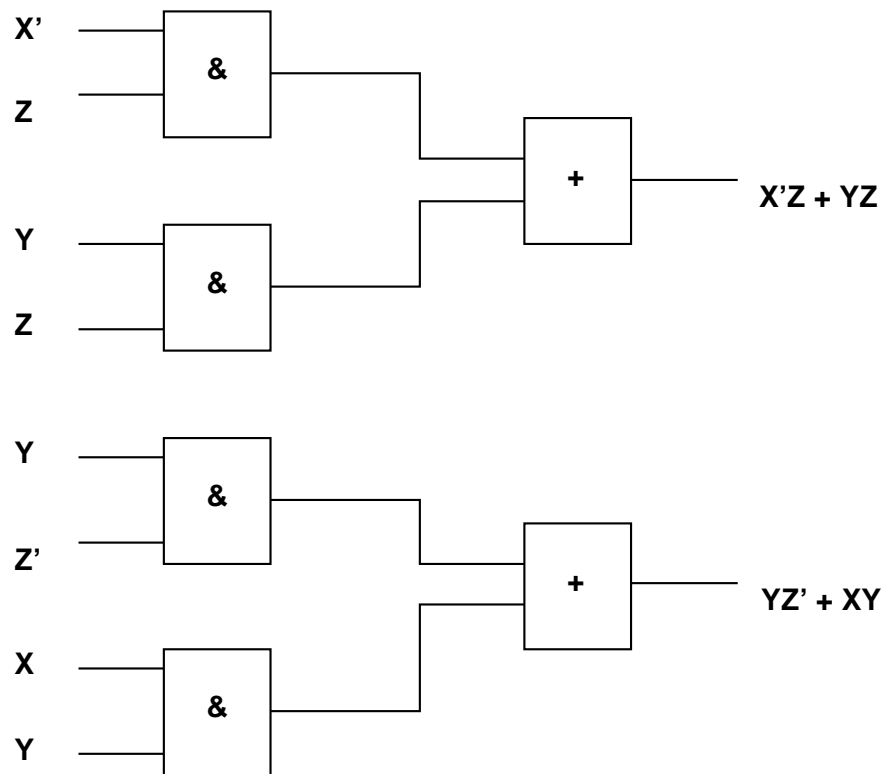
- Minimizing gates can also affect the number of OR gates at the second level.
- When an AND gate outputs 1, it does not necessarily affect all network outputs, i.e., the product term for an AND gate is not an implicant of all f_i .
- The number of internal connections depends upon how many OR gates each AND gate drives.
- The number of input connections is still determined by minimizing the number of literals used in the products defining the first level, but this does not necessarily imply that the products are prime implicants of all (or any) of the f_i .

TWO-OUTPUT EXAMPLE

$$F(X, Y, Z) = X'Z + YZ$$

$$G(X, Y, Z) = YZ' + XY$$

Optimize each separately



6 gates and 14 connections

We can do better. Recall *No-name* theorem: $V_1 + V'_1V_2 = V_1 + V_2$ Rather than algebraic simplification consider algebraic complexification (?)

$$\begin{aligned}
 F(x, y, z) &= x'z + yz \\
 \text{Dist.} &= (x' + y)z \\
 \text{No-name(reversed)} &= (x' + xy)z \\
 \text{Dist.} &= x'z + xyz
 \end{aligned}$$

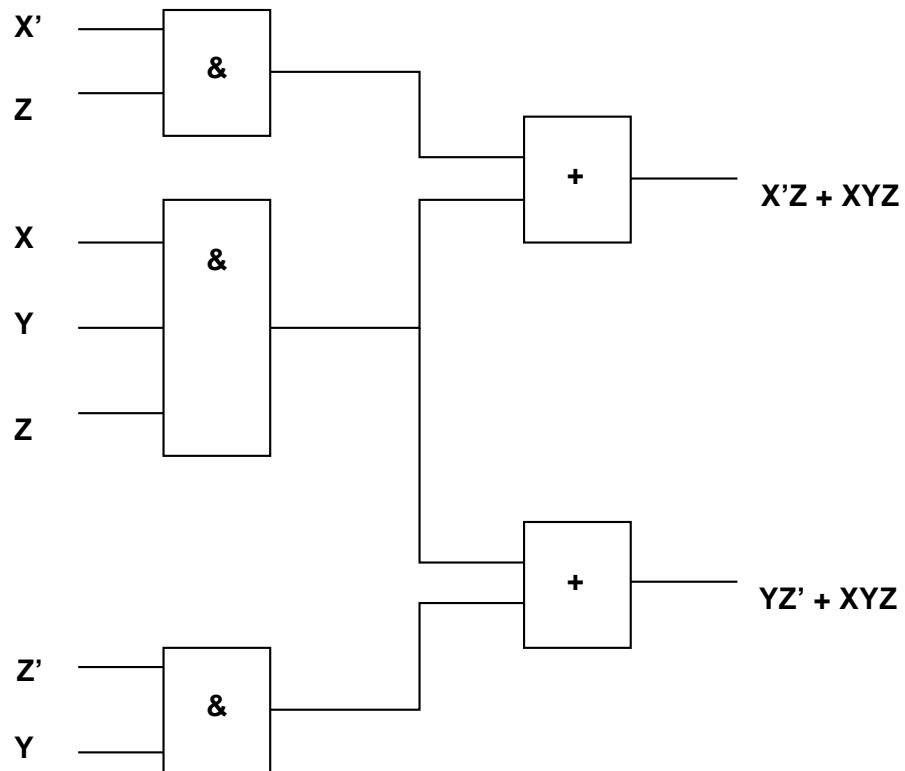
$$\begin{aligned}
 G(x, y, z) &= yz' + xy \\
 \text{Comm.\&Dist.} &= (z' + x)y \\
 \text{No-name(reversed)} &= (z' + xz)y \\
 \text{Dist.\&Comm.} &= yz' + xyz
 \end{aligned}$$

We have increased the complexity of one product in each function. But, there is now a common product term – shared AND gate.

Optimize Together

$$F(X, Y, Z) = X'Z + XYZ$$

$$G(X, Y, Z) = YZ' + XYZ$$



5 gates and 13 connections

An AND gate may go to $1, 2, \dots, k$ OR gates.

AND gates that drive 1 OR gate to contribute to f_i :

- if AND output is 1 then OR gate must output 1 – product must be an implicant of f_i
- to have minimal input connections the number of literals must not be reducible – product is a prime implicant of f_i

AND gates that drive 2 OR gates to contribute to f_i and f_j :

- if AND output is 1 then OR gate for f_i and OR gate for f_j must output 1 – product is an implicant for f_i , f_j and $f_i \bullet f_j$
- to have minimal input connections the number of literals, if one is removed the output of f_i or f_j or both must change, i.e., $f_i \bullet f_j$ must change – the product is a prime implicant of $f_i \bullet f_j$

- note the product does not have to be a prime implicant of f_i or a prime implicant of f_j since dropping a literal does not necessarily cause f_i or f_j to change (consider dropping the x from xyz , f remains the same, $g_{new} = y$)
- AND gates that drive k OR gates to contribute to $f_{i_1} \cdots f_{i_k}$:
 - product must be a prime implicant of $f_{i_1} \bullet \cdots \bullet f_{i_k}$ and an implicant of each.

Definition: A *multiple-output prime implicant* of a set of functions $\{f_1(x_1, \dots, x_n), \dots, f_m(x_1, \dots, x_n)\}$ is a product of literals which is:

- a prime implicant of one of the functions;
- or a prime implicant of a product of two or more of the functions.

Theorem (McCluskey p. 212) : There is at least one minimal two-stage multiple output network where the outputs f_i are all sums of multiple-output prime implicants. All product terms that occur only in expressions for f_j are prime implicants of f_j ; all the product terms that occur in the expression for both f_j and f_k are prime implicants of the product $f_j \bullet f_k$ etc.

So to design minimum cost two-stage multiple-output networks, we must not only consider the prime implicants of all output functions but also the prime implicants of all of the products of two or more of the functions.

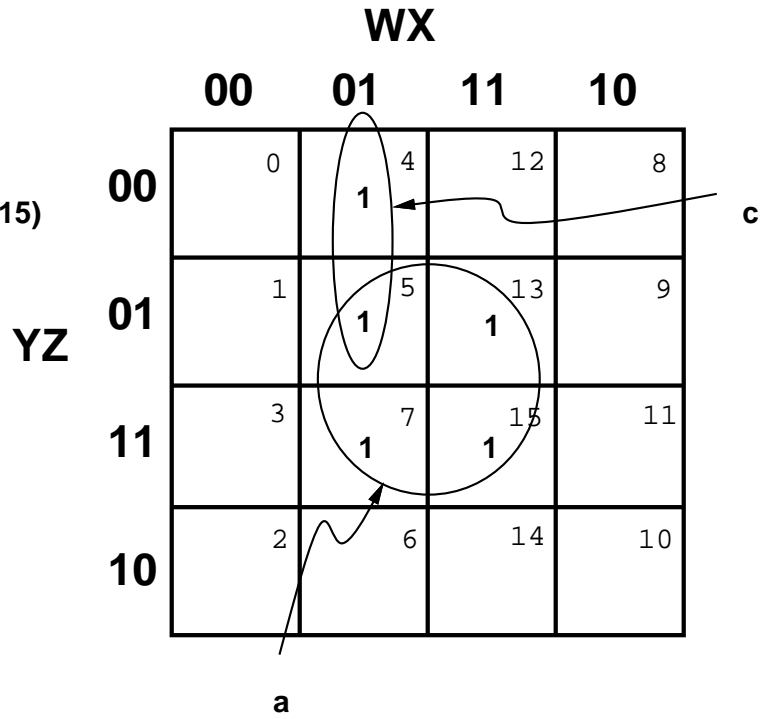
The techniques for enumerating prime implicants and choosing the prime implicants that generate a minimal network can all be generalized.

$$u(W,X,Y,Z) = \sum (4,5,7,13,15)$$

$$u = a + c$$

3 gates 5 inputs

1 output 2 internals

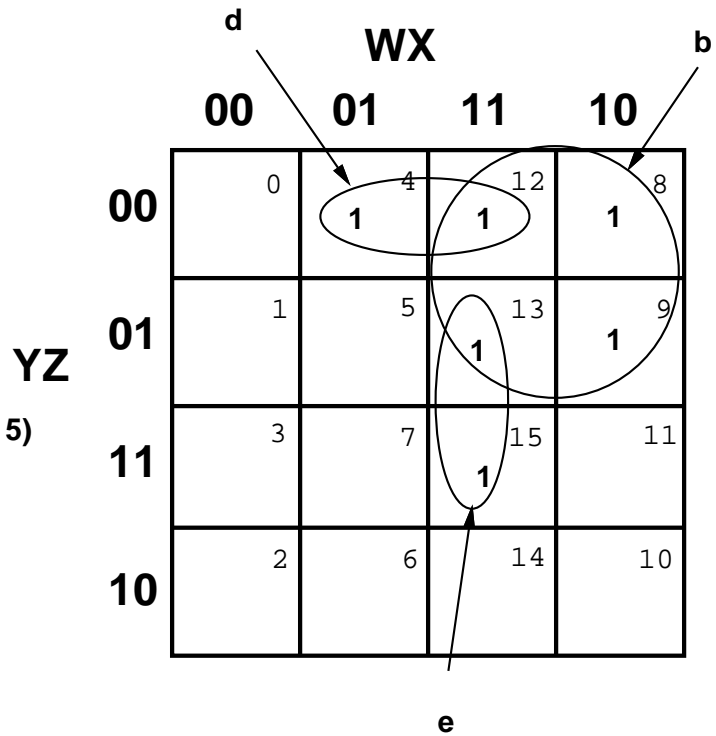


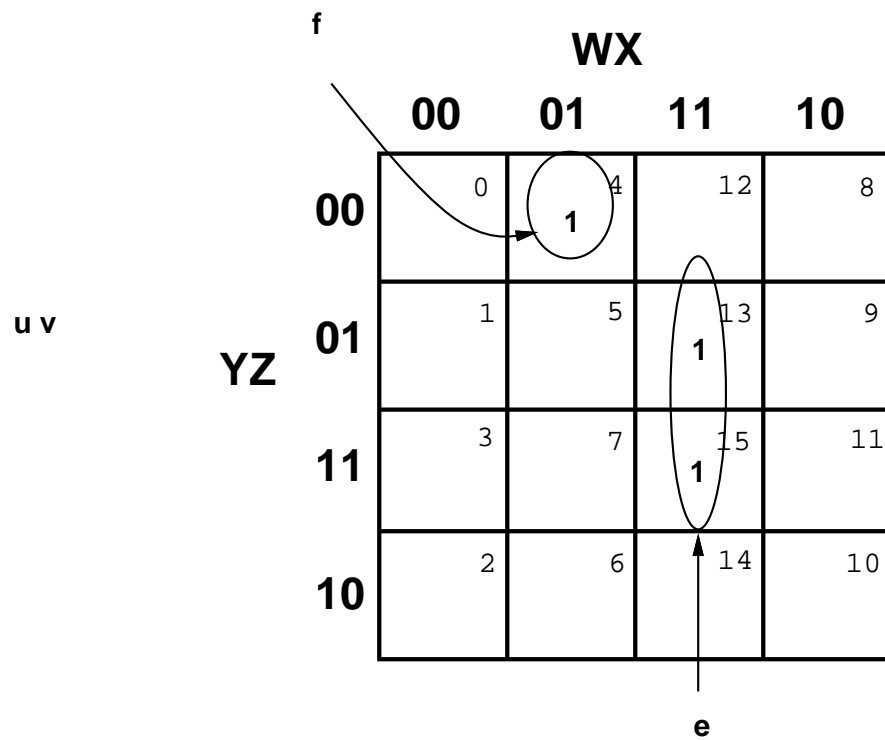
$$v(W,X,Y,Z) = \sum (4,8,9,12,13,15)$$

$$v = b + d + e$$

4 gates 8 inputs

1 output 3 internals





e is a PI of uv and also v

f is a PI of uv but of neither u or v

PI	u4	u5	u7	u13	u15	v4	v8	v9	v12	v13	v15
*a		x	x	x	x						
*b							x	x	x	x	
c	x	x									
d						x			x		
*e				x	x					x	x
f	x					x					

where $a = xz$, $b = wy'$, $c = w'xy'$, $d = xy'z'$, $e = wxz$, and $f = w'xy'z'$

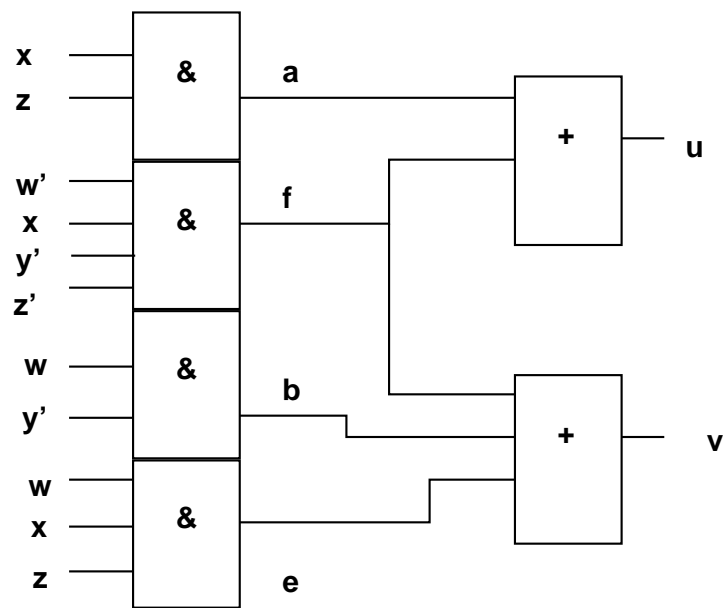
a is essential for u ; b and e are essential for v . Only columns $u4$ and $v4$ and PI's c , d , and f remain after removal.

Use Petrick expressions to combine essentials and remaining columns (Note that the individually optimized sequences are listed.)

$$u : A(C + F) = AC + AF$$

$$v : BE(D + F) = BED + BEF$$

u	v	AND gates
AC	BED	5
AC	BEF	5
AF	BED	5
AF	BEF	4



6 gates 18 connections

Doing MOPI Table manipulation by hand can be made simpler by using a Compact MOPI Table.

Easier for

- essential MOPI identification
- eliminating the resulting distinguished rows and columns from the appropriate part of the table
- column dominance via absorption
- row reductions via dominance can also be done but we will not consider it here

Turn the MOPI table on its side and use Petrick expressions

PI	f4	f5	f7	f13	f15	g4	g8	g9	g12	g13	g15
*a		x	x	x	x						
*b							x	x	x	x	
c	x	x									
d						x			x		
*e				x	x					x	x
f	x					x					

<i>f</i>		<i>g</i>	
4	$C + F$	4	$D + F$
5	$A + C$	8	B
7	A	9	B
13	$A + E$	12	$B + D$
15	$A + E$	13	$B + E$
		15	E

<i>f</i>		<i>g</i>	
4	$C + F$	4	$D + F$
5	$A + C$	8	B
7	A	9	B
13	$A + E$	12	$B + D$
15	$A + E$	13	$B + E$
		15	E

A appears as a singleton in f portion so it is essential to f . By absorption (column dominance in original MOPI table) and equality, 5, 13, 15 can be eliminated from f portion, i.e., $A(A + C)(A + E)(A + E) = A$.

B appears as a singleton in g portion so it is essential to g . By absorption (column dominance in original MOPI table) and equality, 9, 12, 13 can be eliminated from g portion, i.e., $BB(B + D)(B + E) = B$.

E appears as a singleton in g portion so it is essential to g .

We have:

$$\begin{aligned} f & : AX \\ g & : BEY \end{aligned}$$

Need X and Y from compact reduced MOPI table

f		g	
4	$C + F$	4	$D + F$

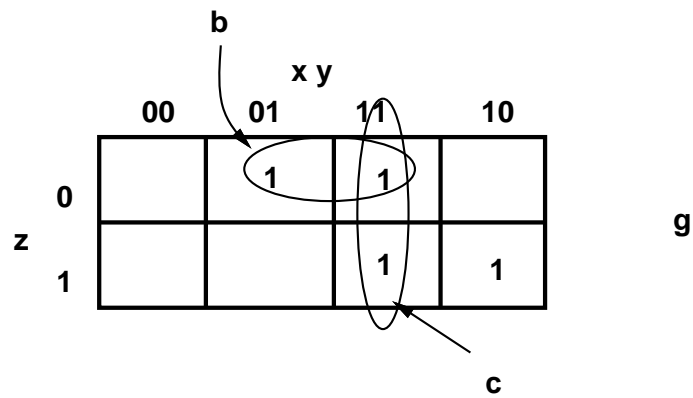
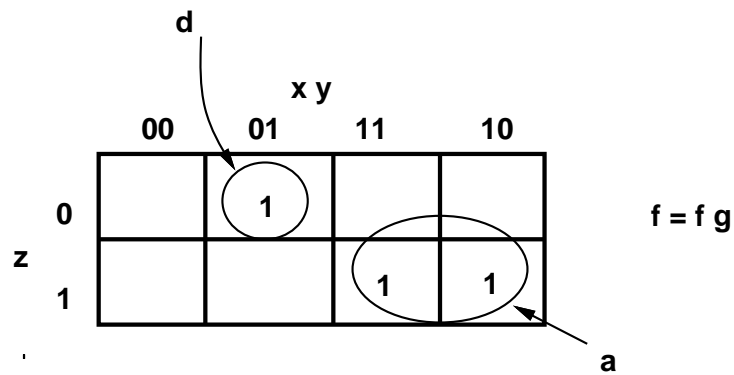
Therefore,

$$\begin{aligned} f & : AX = A(C + F) \\ g & : BEY = BE(D + F) \end{aligned}$$

and we proceed as before.

$$f(x,y,z) = \sum (2,5,7)$$

$$g(x,y,z) = \sum (2,5,6,7)$$



MOPI's

5,7 a = xz (fg)

2,6 b = yz' (g)

6,7 c = xy (g)

2 d = x'yz' (fg)

(increasing input connection costs)

PI	$f = fg$			g			
	f2	f5	f7	g2	g5	g6	g7
(fg) a*		x	x		x		x
(fg) d*	x			x			
(g) b				x		x	
(g) c						x	x

a is essential for f and g : remove row a and remove columns $f5$, $f7$, $g5$, and $g7$.

d is essential for f : remove column $f2$ and f is completely determined BUT d is NOT essential for g so row d must stay, i.e., column dominance is only applied between columns related to the same output function.

PI	g	
	g2	g6
(fg) d	x	
(g) b	x	x
(g) c		x

We could do the same reduction via compact MOPI tables (capital letters are selection switching variables)

<i>f</i>		<i>g</i>	
2	<i>D</i>	2	<i>B + D</i>
5	<i>A</i>	5	<i>A</i>
7	<i>A</i>	6	<i>B + C</i>
		7	<i>A + C</i>

A and *D* essential for *f* and all rows can be removed in *f* portion.

A essential for *g* and 5 and 7 can be removed by Absorption.

<i>g</i>	
2	<i>B + D</i>
6	<i>B + C</i>

Suppose we proceed with Petrick expressions

$$\begin{aligned} f &: AD \\ g &: A(B + D)(B + C) \\ &= A(B + BC + BD + DC) \\ &= A(B + CD) \\ &= AB + ACD \end{aligned}$$

Two possibilities:

$$\begin{aligned} f &: AD \quad \text{and} \quad g : ACD \\ f &: AD \quad \text{and} \quad g : AB \end{aligned}$$

The first is tempting since AD is already chosen for F so we merely add C . Both have only three AND gates.

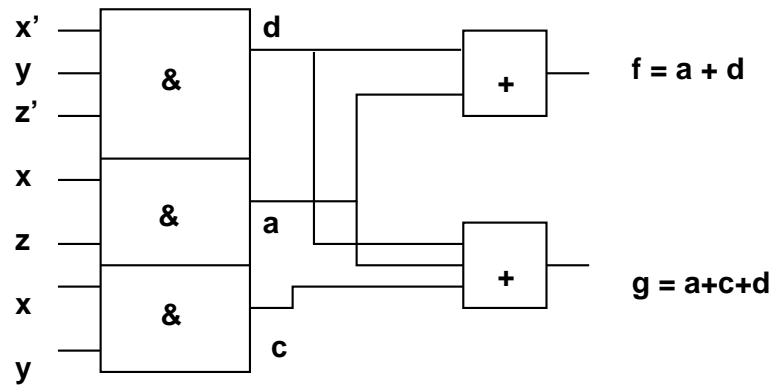
Consider the number of input and internal connections (inputs to the OR gates)

f: AD g: ACD

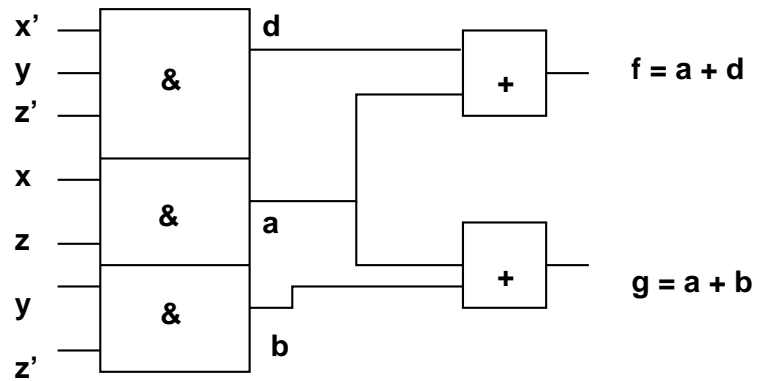
3 AND gates

7 input connections

5 internal connections



f: AD g: AB



only 4 internal connections

What about pairwise row dominance? We had $f : AD$ and $g : A$? so far with the reduced MOPI table

PI	g	
	g2	g6
(fg) d	x	
(g) b	x	x
(g) c		x

In terms of inclusion b dominates both c and d . What about incremental cost? Remember d is already chosen!

- incremental gate costs: $b = 1, c = 1, d = 0$
- incremental input costs: $b = 2, c = 2, d = 0$
- incremental internal input costs: $b = 1, c = 1, d = 1$

So costs are $b = (1, 3), c = (1, 3), d = (0, 1)$ and strict pairwise row dominance does not remove d due to cost failure. We would have to recognize that it takes c and d to give coverage of b and use an aggregate cost if we wanted to remove both at this stage.

We can remove row c however due to equal cost. This yields

PI	g	
	g2	g6
(fg) d	x	
(g) b	x	x

At last, b is essential for g and we can finalize g as

$$g : AB \rightarrow g = a + b$$

So the $g : ACD$ option takes two steps to eliminate from consideration when row dominance is applied.

Summary of MOPI Reduction Rules

- Identification of essential MOPI's and removal of corresponding rows and columns can be done only within the portion of the table corresponding to the output function f_i for which the MOPI is essential.
- Pairwise column dominance reductions can only involve columns that are both in the portion of the table corresponding to the same output function f_i .
- Pairwise row dominance comparisons must use a three term (incremental) cost: gates, input connections, and internal connections.
- A Petrick expression is used for each column of a MOPI table. If a minterm appears in more than one output function a Petrick function is required for each appearance.

How do we find MOPI's?

The most straightforward way to find the MOPI's is

- form $f_i \rightarrow$ PI's via QM tabulation
- form $f_i f_j, i \neq j, \rightarrow$ PI's via QM tabulation
- form $f_i f_j f_k, i \neq j \neq k, \rightarrow$ PI's via QM tabulation
- ETC.

This is unnecessarily wasteful.

The products need not be formed.

We can use the Tagged Tabulation Procedure due to Bartee (1961).

Tagged Tabulation Procedure

Initialization:

List minterms.

Associate with each a binary identifier indicating which variables appear complemented and which appear without complementation in the expression for the minterm. (Same as single-output case).

Associate with each a tag of k bits, one for each output function. The i -th bit is 1 if the minterm is present in f_i , 0 otherwise.

Example:

$$\mathcal{F}(w, x, y, z) = \sum(1, 2, 3, 9)$$

$$\mathcal{G}(w, x, y, z) = \sum(2, 3, 5, 7, 9, 10, 11)$$

$$\mathcal{H}(w, x, y, z) = \sum(1, 2, 3, 7, 9, 10, 11)$$

POP	minterm	w x y z	$\mathcal{F}\mathcal{G}\mathcal{H}$	covered
1	m_1	0001	101	x
1	m_2	0010	111	x
2	m_3	0011	111	x
2	m_5	0101	010	x
2	m_9	1001	111	i
2	m_{10}	1010	011	x
3	m_7	0111	011	x
3	m_{11}	1011	011	x

Next, as in single-output case, compare the identifier patterns which differ by 1 in the POP count.

An entry is formed in a new table any time two minterms are found whose identifiers differ in exactly one bit.

The identifier of the new entry is formed as in the single-output case – x in the position of the differing bit and the common bit pattern elsewhere.

The tag of the new entry is the bitwise AND of the tags of the two minterms, e.g., $t_1 = (110)$ and $t_2 = (011)$ then $t_{new} = (010)$.

The minterms that form a new entry are marked covered if they also have the same tag pattern as the new entry, i.e., they must appear in the same output functions in order to be covered by the expression of the new entry.

POP	minterm	w x y z	FGH	covered
1	m_1	0001	101	x
1	m_2	0010	111	x
2	m_3	0011	111	x
2	m_5	0101	010	x
2	m_9	1001	111	i
2	m_{10}	1010	011	x
3	m_7	0111	011	x
3	m_{11}	1011	011	x

POP	cells	w x y z	FGH	covered
1	1,3	00x1	101	c
1	1,5	0x01	000	remove
1	1,9	x001	101	d
1	2,3	001x	111	e
1	2,10	x010	011	x
2	3,7	0x11	011	f
2	3,11	x011	011	x
2	5,7	01x1	010	g
2	9,11	10x1	011	h
2	10,11	101x	011	x

This table is then used to form a third.

POP	cells	w x y z	\mathcal{FGH}	covered
1	1,3	00x1	101	c
1	1,5	0x01	000	remove
1	1,9	x001	101	d
1	2,3	001x	111	e
1	2,10	x010	011	x
2	3,7	0x11	011	f
2	3,11	x011	011	x
2	5,7	01x1	010	g
2	9,11	10x1	011	h
2	10,11	101x	011	x

POP	cells	w x y z	\mathcal{FGH}	covered
1	1,3,9,11	x0x1	001	a
1	1,3,5,7	0xx1	000	remove
1	2,3,10,11	x01x	011	b

MOPI's

MOPI	cells	w x y z	\mathcal{FGH}	max. input cost
a	1,3,9,11	x0x1	001	2 + 1
b	2,3,10,11	x01x	011	2 + 2
c	1,3	00x1	101	3 + 2
d	1,9	x001	101	3 + 2
e	2,3	001x	111	3 + 3
f	3,7	0x11	011	3 + 2
g	5,7	01x1	010	3 + 1
h	9,11	10x1	011	3 + 2
i	9	1001	111	4 + 3

\mathcal{F}		\mathcal{G}		\mathcal{H}	
1	$C + D$	2	$B + E$	1	$A + C + D$
2	E	3	$B + E + F$	2	$B + E$
3	$C + E$	5	G	3	$A + B + C + E + F$
9	$D + I$	7	$F + G$	7	F
		9	$H + I$	9	$A + D + H + I$
		10	B	10	B
		11	$B + H$	11	$A + B + H$

Essential MOPI's are

$$\begin{aligned} \mathcal{F} &: e \\ \mathcal{G} &: b \ g \\ \mathcal{H} &: b \ f \end{aligned}$$

Petrick expressions (essentials marked):

$$\begin{aligned} \mathcal{F} &: E(C + D)(D + I) = E^*D + E^*CI \\ \mathcal{G} &: BG(H + I) = B^*G^*H + B^*G^*I \\ \mathcal{H} &: BF(A + C + D) = B^*F^*A + B^*F^*D \\ & (A + D + H + I) + B^*F^*CH + B^*F^*CI \end{aligned}$$

$b, e, f,$ and g selected already.

We can complete the outputs with only 2 more gates. Three choices DH , DI , or CI .

This clear by inspection here but in general you can create a “remaining” function by simply noting that you must have $p_{\mathcal{F}} \bullet p_{\mathcal{G}} \bullet p_{\mathcal{H}} = 1$, where $p_{\mathcal{F}}, p_{\mathcal{G}}$, and $p_{\mathcal{H}}$ are the respective prime implicant functions, since each must be 1. Certain selection variables have already been set to 1 due to their prime implicants being essential, e.g., $E = 1$ in $p_{\mathcal{F}}$, $B = G = 1$ in $p_{\mathcal{G}}$ and $B = F = 1$ in $p_{\mathcal{H}}$. Note these are only set to 1 in the appropriate function. If E appeared in $p_{\mathcal{G}}$ but e was not essential for \mathcal{G} then it would not be set to 1 in $p_{\mathcal{G}}$ but would be carried along as a 0 cost gate.

After setting the values (and therefore restricting the function) we have

$$\begin{aligned}
 (p_{\mathcal{F}}p_{\mathcal{G}}p_{\mathcal{H}})_{restricted} &= 1 \\
 &= (C + D)(D + I)(H + I)(A + C + D) \\
 &= (C + D)(A + C + D)(D + I)(H + I) \\
 &= (C + D)(D + I)(H + I) \\
 &= (D + CI)(H + I) \\
 &= DH + DI + CI + HCI \\
 &= DH + DI + CI
 \end{aligned}$$

which yields the three possible completion choices. Note, however, in general after choosing the completion you must go back to the individual prime implicant functions to determine which connections are used when multiple connections are possible for a given MOPI.

CI c adds 3 inputs and 2 internals

i adds 4 inputs and 3 internals

DI d adds 3 inputs and 2 internals

i adds 4 inputs and 1 internal

DH d adds 3 inputs and 2 internals

h adds 3 inputs and 1 internal

