

(Lec 8) Multilevel Min. II: Cube/Cokernel Extract

▼ What you know

- ▶ 2-level minimization a la **ESPRESSO**
- ▶ **Boolean network model** -- let's us manipulate multi-level structure
- ▶ **Algebraic model** -- simplified model of Boolean eqns lets us factor stuff
- ▶ **Algebraic division / Kerneling**: ways to actually do Boolean division

▼ What you don't know

- ▶ Given a big **Boolean network**...
- ▶ ... how do we actually **extract** a good set of factors over all the nodes?
- ▶ What's a good set of factors to try to find?

▼ This is "extraction"

- ▶ Amazingly enough, there is a unifying "covering" problem here whose solution answers all these extraction problems

(Concrete examples from Rick Rudell's 1989 Berkeley PhD thesis)

© R. Rutenbar 2001

18-760, Fall 2001 1

Copyright Notice

© Rob A. Rutenbar 2001

All rights reserved.

You may not make copies of this material in any form without my express permission.

© R. Rutenbar 2001

18-760, Fall 2001 2

Where Are We?

▼ In logic synthesis--how *multilevel factoring* really works

	M	T	W	Th	F	
Aug	27	28	29	30	31	1
Sep	3	4	5	6	7	2
	10	11	12	13	14	3
	17	18	19	20	21	4
	24	25	26	27	28	5
Oct	1	2	3	4	5	6
	8	9	10	11	12	7
	15	16	17	18	19	8
	22	23	24	25	26	9
	29	30	31	1	2	10
Nov	5	6	7	8	9	11
	12	13	14	15	16	12
Thnxgive	19	20	21	22	23	13
	26	27	28	29	30	14
Dec	3	4	5	6	7	15
	10	11	12	13	14	16

Introduction
 Advanced Boolean algebra
 JAVA Review
 Formal verification
 2-Level logic synthesis
Multi-level logic synthesis
 Technology mapping
 Placement
 Routing
 Static timing analysis
 Electrical timing analysis
 Geometric data structs & apps

© R. Rutenbar 2001

18-760, Fall 2001 3

Readings & Deadlines

▼ DeMicheli does not have much relevant stuff

- ▶ 8.3.2 Extraction and algebraic kernels-- gives a few small examples like the ones in my lecture

▼ Deadlines

- ▶ Today: Project I JAVA BDDs due
- ▶ Thursday Oct 11: Paper I Review, Rudell's Dynamic Ordering due
- ▶ Thursday Oct 18: HW3 (2-level, multi-level synthesis) due
 - ▷ As always, check webpage for bugfixes, updates...

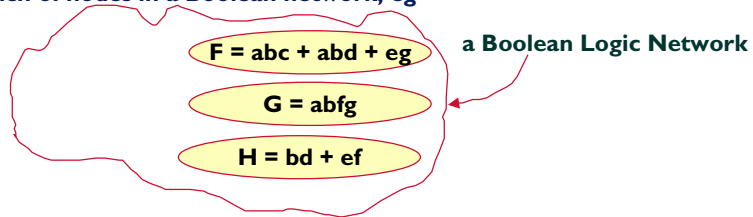
© R. Rutenbar 2001

18-760, Fall 2001 4

Strategy: Single Cube Extraction

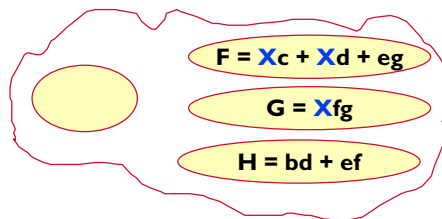
Input

- ▶ A bunch of nodes in a Boolean network, eg



Goal

- ▶ Find 1 cube we can extract, useful in many (not necessarily all) of nodes



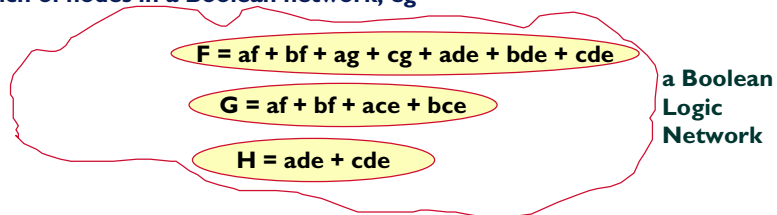
© R. Rutenbar 2001

18-760, Fall 2001 5

Strategy: Multiple Cube Extraction

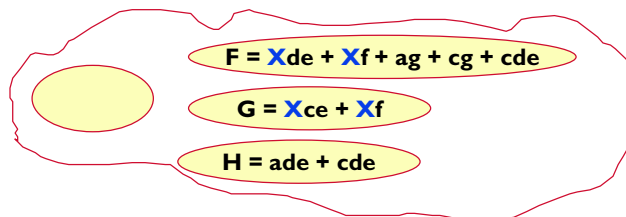
Input

- ▶ A bunch of nodes in a Boolean network, eg



Goal

- ▶ Find a *multiple cube expression* we can extract that is useful in many (not necessarily all) of these nodes



© R. Rutenbar 2001

18-760, Fall 2001 6

Single Cube Extraction: Matrix Representation

▼ Given: a set of SOP Boolean equations

$$F = abc + abd + eg$$

$$G = abfg$$

$$H = bd + ef$$

▼ Construct the *cube-literal matrix* as follows

- ▶ 1 row for each unique *product term*; 1 column for each unique *literal*
- ▶ A "1" in the matrix if this product term uses this literal, else a "."
- ▶ Number each row and col for convenience in referring to them, later

$$F = abc + abd + eg$$

$$G = abfg$$

$$H = bd + ef$$

		a	b	c	d	e	f	g
		1	2	3	4	5	6	7
abc	1	1	1	1
abd	2	1	1	.	1	.	.	.
eg	3	1	.	1
abfg	4	1	1	.	.	.	1	1
bd	5	.	1	.	1	.	.	.
ef	6	1	1	.

© R. Rutenbar 2001

18-760, Fall 2001 7

Covering this Matrix: Rectangle Definition

▼ A *rectangle* of this matrix...

- ▶ ...is a set of rows **R**, and a set of columns **C** -- denoted (R,C)--of the cube-literal matrix..
- ▶ ... such that if *r* is a row in **R**, *c* is a column in **C**, then the cube-lit matrix is "1" in row *r* and column *c*.
- ▶ **NOTE:** rectangles *don't* need to be made up of contiguous rows or cols in the cube-lit matrix. Any set of rows can be in **R**; ditto cols in **C**

Example rectangle

		a	b	c	d	e	f	g
		1	2	3	4	5	6	7
abc	1	1	1	1
abd	2	1	1	.	1	.	.	.
eg	3	1	.	1
abfg	4	1	1	.	.	.	1	1
bd	5	.	1	.	1	.	.	.
ef	6	1	1	.

© R. Rutenbar 2001

18-760, Fall 2001 8

Covering this Matrix: Prime Rectangles

▼ A *prime* rectangle of this matrix...

- ▶ ...is a rectangle of the matrix not strictly contained inside another rectangle of the matrix
- ▶ In English: cannot make a prime rectangle any *bigger* by adding another row or another column (like “primes” in a Kmap)

Example *prime* rectangle
(bigger than the rect on
previous page...)

		a	b	c	d	e	f	g
		1	2	3	4	5	6	7
abc	1			
abd	2		
eg	3	
abfg	4			.	.	.		
bd	5
ef	6

© R. Rutenbar 2001

18-760, Fall 2001 9

Rectangles: Who Cares?

▼ We do. *Cool* result:

- ▶ Prime rectangles are “biggest possible” common single-cube divisors
- ▶ Makes sense: columns of (R,C) tell you the lits in the product term divisor (ie. you AND these cols), and rows tell you which product terms you can divide!

$$F = abc + abd + eg$$

$$G = abfg$$

$$H = bd + ef$$

		a	b	c	d	e	f	g
		1	2	3	4	5	6	7
abc	1			
abd	2		
eg	3	
abfg	4			.	.	.		
bd	5
ef	6

I cube divisor

F =

G =

H =

X =

© R. Rutenbar 2001

18-760, Fall 2001 10

Matrix Update

▼ OK, suppose we extract $X=a \cdot b$, now what?

- ▶ Need to update the matrix, since things changed

▼ Update mechanics

- ▶ **Add a column** at end of matrix, label it with name of new factor (**X** in this case).
 - ▷ **Why?** **X** is like any other literal now (look at **F**, **G** to see it)
- ▶ **Add another row** at bottom of matrix, label with product term of factor (**ab** in this case)
 - ▷ **Why?** We added a new function to our set, $X=ab$, so we have to record this new product term in our matrix
- ▶ **Change all the "1" entries** of the prime rectangle we just extracted into "*" to indicate don't cares
 - ▷ **Why?** It's OK to cover these guys again with the next rectangle we extract, but if you don't cover them it's OK too
- ▶ **Redefine "rectangle"** to mean "rows, cols don't cover any "." entries, ie, it's OK to cover the "*" con't cares

© R. Rutenbar 2001

18-760, Fall 2001 11

Matrix Update: Before & After

Before

		a	b	c	d	e	f	g
	1	1	2	3	4	5	6	7
abc	1	1	1
abd	2	1	1	.	1	.	.	.
eg	3	1	.	1
abfg	4	1	1	.	.	.	1	1
bd	5	.	1	.	1	.	.	.
ef	6	1	1	.

After

		a	b	c	d	e	f	g	X
	1	1	2	3	4	5	6	7	8
abc	1	*	*
abd	2	*	*	.	1
eg	3	1	.	1	.
abfg	4	*	*	.	.	.	1	1	.
bd	5	.	1	.	1
ef	6	1	1	.	.
ab	7

© R. Rutenbar 2001

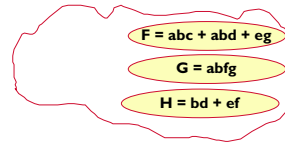
18-760, Fall 2001 12

Update: Why are We Doing This?

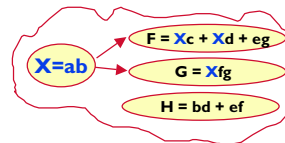
▼ Because we want to extract the *next* good factor...

- And so we have to update the matrix to represent the new network we got by yanking out the last factor

	a	b	c	d	e	f	g
	1	2	3	4	5	6	7
abc	1	1	1
abd	2	1	1	1	.	.	.
eg	3	.	.	.	1	1	1
abfg	4	1	1	.	.	1	1
bd	5	.	1	1	.	.	1
ef	6	.	.	.	1	1	1



	a	b	c	d	e	f	g	X
	1	2	3	4	5	6	7	8
abc	1	*	*	1	.	.	.	1
abd	2	*	*	.	1	.	.	1
eg	3	.	.	.	1	1	1	1
abfg	4	*	*	.	.	1	1	1
bd	5	.	1	1	.	.	1	1
ef	6	.	.	.	1	1	1	1
ab	7	1	1	1



© R. Rutenbar 2001

18-760, Fall 2001 13

Repeat on Updated Matrix: Get Next Cube Factor

Next good prime rect

	a	b	c	d	e	f	g	X
	1	2	3	4	5	6	7	8
abc	1	*	*	1	.	.	.	1
abd	2	*	*	.	1	.	.	1
eg	3	.	.	.	1	1	1	1
abfg	4	*	*	.	.	1	1	1
bd	5	.	1	1	.	.	1	1
ef	6	.	.	.	1	1	1	1
ab	7	1	1	1

▼ Hey, this one overlaps the last one (the “*”s)! Is this OK?

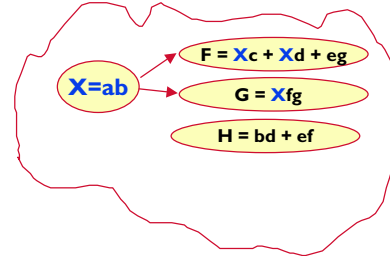
- Actually, not a problem
- It does mess up an assumption in the algebraic model...
- Look closely at what happens...

© R. Rutenbar 2001

18-760, Fall 2001 14

Second Rectangle (Cube) Extracted...

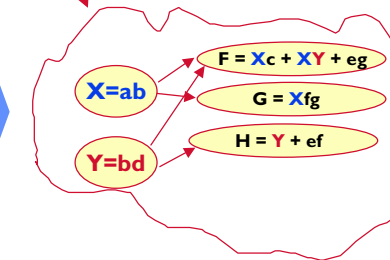
	a	b	c	d	e	f	g	X
abc	1	*	*		.	.	.	
abd	2	*	*		.	.	.	
eg	3	
abfg	4	*	*		.	.	.	
bd	5	
ef	6	
ab	7	



cube=

	a	b	c	d	e	f	g	X
abc	1	*	*		.	.	.	
abd	2	*	*		.	.	.	
eg	3	
abfg	4	*	*		.	.	.	
bd	5	
ef	6	
ab	7	

New Prime rect



© R. Rutenbar 2001

18-760, Fall 2001 15

Overlaps of Prime Rectangles? Look Closely...

▼ Started with...

$$\begin{aligned} F &= abc + abd + eg \\ G &= abfg \\ H &= bd + ef \end{aligned}$$

▼ Extracted $X=ab$ and got...

$$\begin{aligned} F &= Xc + Xd + eg \\ G &= Xfg \\ H &= bd + ef \\ X &= ab \end{aligned}$$

▼ Extracted $Y=bd$ and got

$$\begin{aligned} F &= Xc + XY + eg \\ G &= Xfg \\ H &= Y + ef \\ X &= ab \\ Y &= bd \end{aligned}$$

▼ Look again at F...

$$\begin{aligned} F &= abc + abd + eg \\ &\Rightarrow (ab)c + (ab)(bd) + eg \\ &\text{is what we really are implementing} \end{aligned}$$

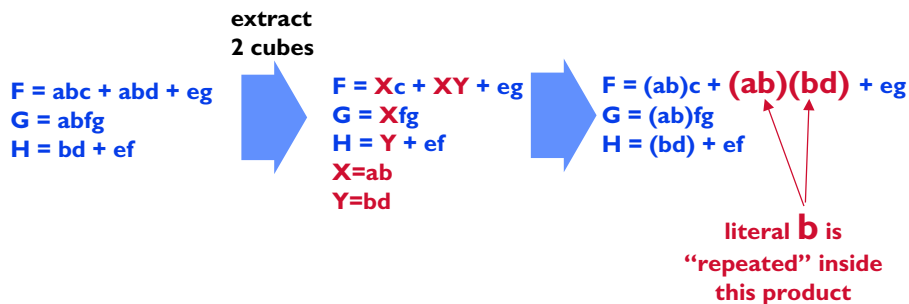
© R. Rutenbar 2001

18-760, Fall 2001 16

Overlaps of Prime Rectangles

Effect:

- ▶ Literals get *repeated* in the factoring of products
- ▶ This is a technical violation of the algebraic model, which said that if we factor $f = d \cdot q + r$, we should have d, q with *disjoint* support...
- ▶ ...ie, d, q have no common variables
- ▶ In this case, overlapping rectangles mean d, q do have common vars



© R. Rutenbar 2001

18-760, Fall 2001 17

How Do We Actually Find a Big, Prime Rectangle?

Lots of strategies

- ▶ Can do it one rectangle at a time, or try to find a cover with many rectangles simultaneously (ie, remember ESPRESSO irredundant?)
- ▶ Can try to do exact search, which is very expensive
- ▶ Or can try various heuristics that get you a good (not perfect) prime

Central idea: Value of a prime rectangle

- ▶ Value is how many literals you should save if you factor out the cube (product term) that corresponds to this rectangle
- ▶ It is your estimated *improvement* in the overall Boolean Logic Network by factoring this cube out
- ▶ How can we compute this easily?

© R. Rutenbar 2001

18-760, Fall 2001 18

Define: Weights & Values for a Cube-Lit Matrix

▼ Define: the *weight of a row* of the cube-lit matrix

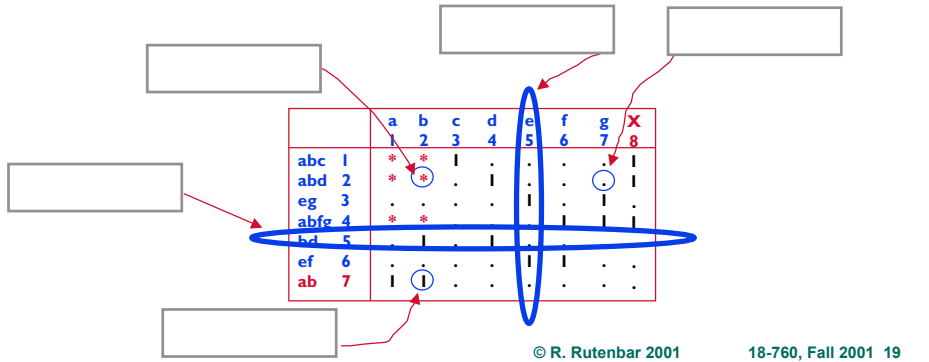
▶ It's just = 1 for each row

▼ Define: the *weight of a column* of cube-lit matrix

▶ It's just = 1 for each column

▼ Define: the *value of an element* of cube-lit matrix

▶ It's = 1 if there is a "1" in that entry, else it's = 0



Define: Weights & Values for a Rectangle

▼ Define: the *weight of a rectangle* of a cube-lit matrix

▶ If just 1 row in rectangle (R,C): $\sum_{c \text{ in } C}$ (weight of each column c)

▶ Else for (R,C): $\sum_{c \text{ in } C}$ (weight of each col c) + $\sum_{r \text{ in } R}$ (weight of each row r)

▼ Define: the *value of a rectangle* of a cube-lit matrix

▶ $[\sum_{r \text{ in } R} \sum_{c \text{ in } C}$ (value of element r,c in matrix)] - (weight of rectangle)

▼ In *English*, the *value* of a rect is:

▶ (# of "1"s covered by the rectangle) - (# of rows + # cols of the rectangle)

▼ Why is this a good number to know...?

Example: Value of a Rectangle

▼ Rect is

	a	b	c	d	e	f	g
abc 1							
abd 2							
eg 3							
abfg 4							
bd 5							
ef 6							



$$F = abc + abd + eg$$

$$G = abfg$$

$$H = bd + ef$$



$$X = ab$$

$$F = Xc + Xd + eg$$

$$G = Xfg$$

$$H = bd + ef$$

Observe: #1s covered by the rect =

Observe: #rows + #cols of rect =

Observe: value = (#1s) - (#rows + #cols) =

Change in #lits =

© R. Rutenbar 2001

18-760, Fall 2001 21

Value of a Rectangle

▼ So, (#1s covered) - (#rows + #cols)...

▶ ...is really a measure of change in #literals after we factor this cube!

▼ Why do we care?

- ▶ A lot of heuristic extraction strategies are “greedy” in nature
- ▶ They try to find a rectangle of maximal value first, then extract it, then update matrix, and repeat--find next max-value rectangle, etc
- ▶ We need these value numbers (and means to compute them) to be able to use strategies like this.

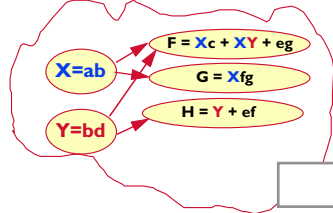
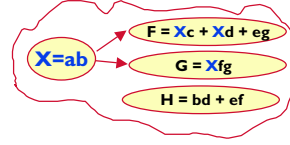
© R. Rutenbar 2001

18-760, Fall 2001 22

Don't Cares in Updated Matrix?

2nd Rect is

	a	b	c	d	e	f	g	X
abc	1	*	*	
abd	2	*	*	
eg	3	.	.	.	1	.	.	.
abfg	4	*	*
bd	5	.	1
ef	6	.	.	.	1	.	.	.
ab	7	1



Observe: #1s covered by the rect =>
don't care *s not counted! =

Observe: #rows + #cols of rect =

Observe: value = (#1s) - (#rows + #cols) =

Change in #1its =

© R. Rutenbar 2001

18-760, Fall 2001 23

Multiple-Cube Factors...?

Remarkably, a very similar matrix concept

Given

- ▶ A set of Boolean functions (nodes in a network)

$$F = af + bf + ag + cg + ade + bde + cde$$

$$G = af + bf + ace + bce$$

$$H = ade + cde$$

Find cokernels and kernels of each of these functions

- ▶ Why? Remember Brayton-McMullen Theorem
- ▶ All interesting multiple-cube factors are found as intersections of the product terms in the kernels for each of these functions

© R. Rutenbar 2001

18-760, Fall 2001 24

Kernels / Cokernels of Our F, G, H Example

▼ $F = af + bf + ag + cg + ade + bde + cde$

- ▶ cokernal (a) kernel (de + f + g)
- ▶ cokernal (b) kernel (de + f)
- ▶ cokernal (de) kernel (a + b + c)
- ▶ cokernal (f) kernel (a + b)
- ▶ cokernal (c) kernel (de + g)
- ▶ cokernal (g) kernel (a+c)
- ▶ cokernal (1) kernel (af + bf + ag + cg + ade + bde + cde) *trivial*

▼ $G = af + bf + ace + bce$

- ▶ cokernal (a) or cokernal (b) kernel (ce + f) (can have > 1 cokernal)
- ▶ cokernal (f) or cokernal (ce) kernel (a + b) (ditto)
- ▶ cokernal (1) kernel (af + bf + ace + bce) *trivial*

▼ $H = ade + cde$

- ▶ cokernal (de) kernel (a + c)
- ▶ cokernal (1) kernel (ade + cde) *trivial*

© R. Rutenbar 2001

18-760, Fall 2001 25

New Matrix: Cokernel-Cube Matrix for Our Ex

- F cokern (a) kern (de + f + g)
- F cokern (b) kern (de + f)
- F cokern (de) kern (a + b + c)
- F cokern (f) kern (a + b)
- F cokern (c) kern (de + g)
- F cokern (g) kern (a+c)
- G cokern (a) kern (ce + f)
- G cokern (b) kern (ce + f)
- G cokern (f) kern (a + b)
- G cokern (ce) kern (a + b)
- H cokern (de) kern (a + c)

One column for each unique co-kernal in problem

			a	b	c	ce	de	f	g
			1	2	3	4	5	6	7
F	a	1	What goes in the individual (row, col) entries here in the matrix?						
F	b	2							
F	de	3							
F	f	4							
F	c	5							
F	g	6							
G	a	7							
G	b	8							
G	ce	9							
G	f	10							
H	de	11							

One row for each unique (Function, Co-kernal) pair in prob

$F = af+bf+ag+cg+ade+bde+cde$

$G = af+bf+ace+bce$

$H = ade + cde$

© R. Rutenbar 2001

18-760, Fall 2001 26

Entries in the Cokernel-Cube Matrix

▼ Mechanics

- ▶ From the row, take the **cokernel**, go look at the associated **kernel**
- ▶ Look at the **product terms** in this kernel
- ▶ Put in a "1" in the columns that represent **cubes** in this kernel; put a "." in all the other columns

$F = af+bf+ag+cg+ade+bde+cde$			a	b	c	ce	de	f	g
			1	2	3	4	5	6	7
F cokern (a) kernel (de + f + g)	F a	1							
	F b	2							
	F de	3							
F cokern (g) kernel (a+c)	F f	4							
	F c	5							
	F g	6							
	G a	7							
	G b	8							
	G ce	9							
	G f	10							
	H de	11							

© R. Rutenbar 2001

18-760, Fall 2001 27

Complete Cokernel-Cube Matrix

		a	b	c	ce	de	f	g
		1	2	3	4	5	6	7
F	a	1	1	1
F	b	2	1	.
F	de	3	1	1	1	.	.	.
F	f	4	1	1
F	c	5	1	1
F	g	6	1	.	1	.	.	.
G	a	7	.	.	.	1	1	.
G	b	8	.	.	.	1	1	.
G	ce	9	1	1
G	f	10	1	1
H	de	11	1	.	1	.	.	.

Notice each row says "here is a function, if I divide it by this co-k cube, this is the result (kernel) I get..."

▼ OK, now what...?

- ▶ Turns out all the rectangle, prime, value stuff still works here!

© R. Rutenbar 2001

18-760, Fall 2001 28

Prime Rectangles in Cokernel-Cube Matrix

▼ A rectangle is a multiple cube factor

	a	b	c	ce	de	f	g
F a	1
F b	2
F de	3
F f	4
F c	5
F g	6
G a	7
G b	8
G ce	9
G f	10
H de	11

A prime rect is (R,C) = ({3,4,9,10}, {1,2})

Interpretation

$$F = (\text{co-k cube de}) \cdot (a + b + c) + \text{rem1}$$

$$F = (\text{co-k cube f}) \cdot (a + b + \text{stuff2}) + \text{rem2}$$

$$G = (\text{co-k cube ce}) \cdot (a + b + \text{stuff3}) + \text{rem3}$$

$$G = (\text{co-k cube f}) \cdot (a + b + \text{stuff4}) + \text{rem4}$$

OR together cubes of the columns of the prime rect -- that's the factor!

© R. Rutenbar 2001

18-760, Fall 2001 29

Multiple-Cube Extraction

	a	b	c	ce	de	f	g
F a	1
F b	2
F de	3
F f	4
F c	5
F g	6
G a	7
G b	8
G ce	9
G f	10
H de	11

$F = af + bf + ag + cg + ade + bde + cde$

$G = af + bf + ace + bce$

$H = ade + cde$

$X = a+b$

$F = Xde + Xf + ag + cg + cde$

$G = Xce + Xf$

$H = ade + cde$

cols of rect = {a, b} -> a + b is factor

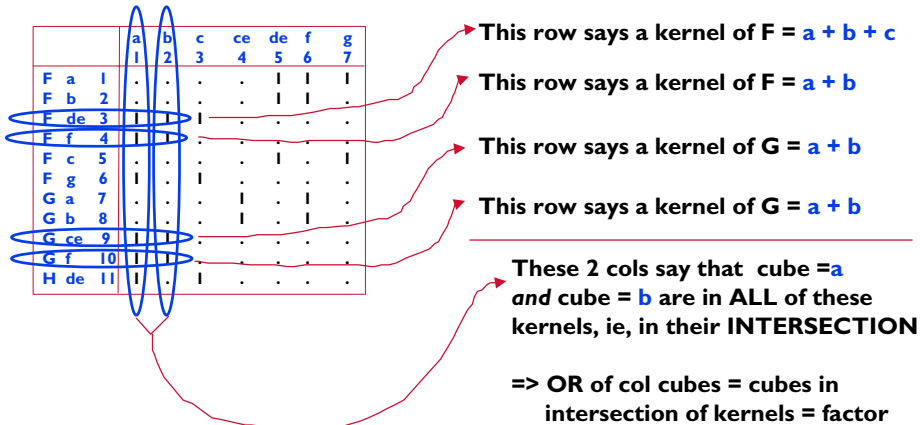
© R. Rutenbar 2001

18-760, Fall 2001 30

Aside: Brayton-McMullen Revisited

▼ This theorem said intersections of kernels yielded all the good multiple cube factors

► So, exactly how does the matrix get this...?



© R. Rutenbar 2001

18-760, Fall 2001 31

Cokernel-Cube Matrix Update

▼ Same problem as before

- You pulled out a factor, and now you want to pull out the next factor
- You have to update the matrix to represent the new situation

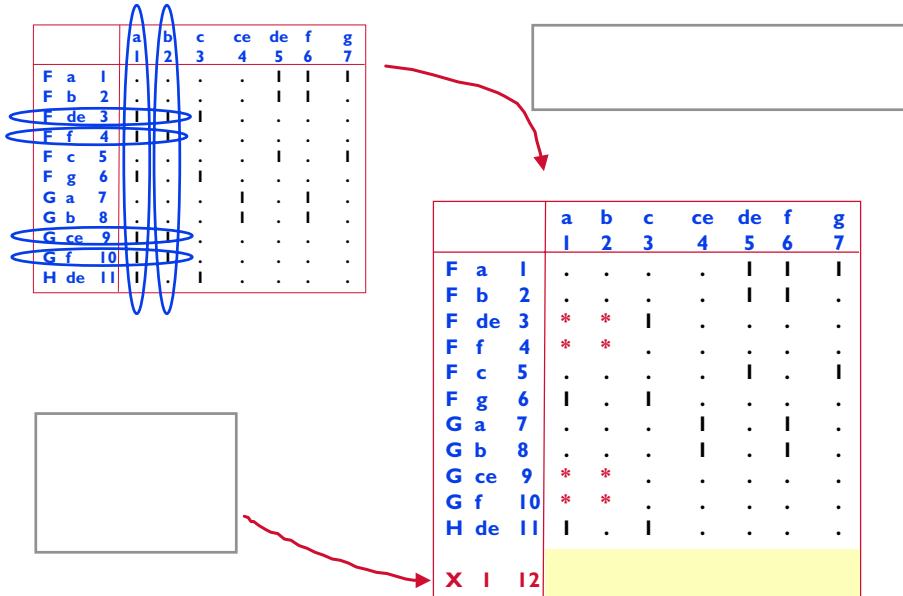
▼ Messier now: mechanically....

- First, *kernel* the multi-cube factor you just pulled out
- In our case $(a + b)$ just has trivial cokernel = (1) , kernel = $(a+b)$
- For each kernel you found here, add a new row (and label as before)
- No new columns - you are not going to find any new product terms in the kernels of the factor--these are just intersections of stuff you already have found before
- Change all the "1"s in the prime rectangle you found into "*"s, since they are don't cares now -- you don't need to worry about these any more
- Messy part: Other entries outside rectangle also become "*"s...

© R. Rutenbar 2001

18-760, Fall 2001 32

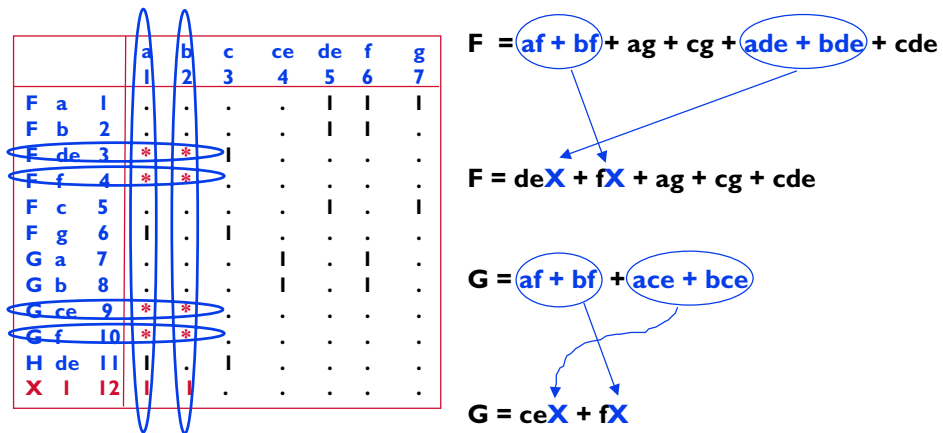
Matrix Update: Straightforward Parts



© R. Rutenbar 2001 18-760, Fall 2001 33

Matrix Update: Messy Part

Look closely at what we actually did with extracted (a+b)



© R. Rutenbar 2001 18-760, Fall 2001 34

Matrix Update: Messy Part

▼ Ideas

- ▶ Don't look for any more factors that involve the circled terms of the original functions
- ▶ We already took our "best shot" at factoring these
- ▶ Focus on the remaining, uncircled terms

$$F = (af + bf) + ag + cg + (ade + bde) + cde$$

$$F = deX + fX + ag + cg + cde$$

▼ How?

- ▶ Each original term in one of the original functions is the product of a cokernel (row) and one term of a kernel (column)
- ▶ Put "*"s on the elements where (row*column) term == one of the eqns circled, ie, subsumed in one of our new factored terms

$$G = (af + bf) + (ace + bce)$$

$$G = ceX + fX$$

© R. Rutenbar 2001

18-760, Fall 2001 35

Update: Elements Outside Prime Rectangle

▼ For each product term of each original function subsumed by the extracted factor, make "*" where (row*col) makes that term

We covered *ade* here... *...so don't need to cover this other ade term here*

		a	b	c	ce	de	f	g
F	a	1	
F	b	2
F	de	3	*	*		.	.	.
F	f	4	*	*
F	c	5	
F	g	6	
G	a	7	.	.	.	*	*	.
G	b	8	.	.	.	*	*	.
G	ce	9	*	*
G	f	10	*	*
H	de	11	
X	I	12		

$$F = (af + bf) + ag + cg + (ade + bde) + cde$$

$$F = deX + fX + ag + cg + cde$$

$$G = (af + bf) + (ace + bce)$$

$$G = ceX + fX$$

NOTE:

© R. Rutenbar 2001

18-760, Fall 2001 36

2nd Factor to Extract

→ Prime rectangle is (R,C) => $Y = (a+c)$

	a	b	c	ce	de	f	g
F a	1	.	.	.	*	*	
F b	2	.	.	.	*	*	.
F de	3	*	*		.	.	.
F f	4	*	*
F c	5	
F g	6		
G a	7	.	.	*	.	*	.
G b	8	.	.	*	.	*	.
G ce	9	*	*
G f	10	*	*
H de	11		
X	12		

After 1st factor of $X=(a+b)$:

$$F = deX + fX + ag + cg + cde$$

$$G = ceX + fX$$

$$H = ade + cde$$

$$X = a + b$$

After 2nd factor of $Y=(a+c)$:

$$F = deX + fX + deY + gY$$

$$G = ceX + fX$$

$$H = deY$$

$$X = a + b$$

$$Y = a + c$$

Hey, this prime rect circles a don't care! So, what happens?

© R. Rutenbar 2001

18-760, Fall 2001 37

Covering a Don't Care

▼ Means we covered an element twice

- ▶ First time with first factor it was a "1"
- ▶ Second time with second factor it was a "*"
- ▶ Effect...? Duplicates some logic

	a	b	c	ce	de	f	g
F a	1	.	.	.	*	*	
F b	2	.	.	.	*	*	.
F de	3	*	*		.	.	.
F f	4	*	*
F c	5	
F g	6		
G a	7	.	.	*	.	*	.
G b	8	.	.	*	.	*	.
G ce	9	*	*
G f	10	*	*
H de	11		
X	12		

After 1st factor of $(a+b)$:

$$F = deX + fX + ag + cg + cde$$

After 2nd factor of $(a+c)$:

$$F = deX + fX + deY + gY$$

$$= de(a+b) + f(a+b) + de(a+c) + g(a+c)$$

$$= ade + bde + af + bf + ade + cde + ag + cg$$

© R. Rutenbar 2001

18-760, Fall 2001 38

Weights & Values for Cokernel-Cube Matrix?

- ▼ Very similar to case of simpler cube-lit matrix
- ▼ Define: *weight of column of co-kernel cube matrix*
 - ▶ The number of literals in the cube that labels the column
- ▼ Define: *weight of row of co-kernel cube matrix*
 - ▶ $l +$ number of literals in the co-kernel cube that labels the row
- ▼ Define: *value of an element of the co-kern cube matrix*
 - ▶ If the matrix has a “1” there: number of literals in the product term you get by ANDing row cube label with column cube label
 - ▶ If matrix has a “.” there or a “*” there, $=0$

© R. Rutenbar 2001

18-760, Fall 2001 39

Weight, Value Examples

	a	b	c	ce	de	f	g
	1	2	3	4	5	6	7
F a 1	*	*	1
F b 2	*	*	.
F de 3	*	*	1
F f 4	*	*
F c 5	1	.	1
F g 6	1	.	1
G a 7	*	*
G b 8	.	.	.	*	.	*	.
G ce 9	*	*
G f 10	*	*
H de 11	1	.	1
X 1 12	1	1

© R. Rutenbar 2001

18-760, Fall 2001 40

Rectangle Weight & Value in Cokern-Cube Matrix

▼ Nicely enough, *same* as before with these definitions

▼ *Weight* of a rectangle

▶ $[\sum (\text{row weights}) + \sum (\text{col weights})]$

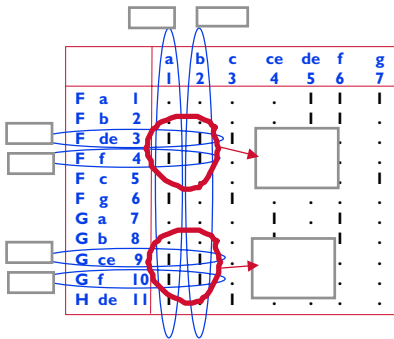
▼ *Value* of a rectangle

▶ $[\sum \text{element value of each "1" covered}]$
 - $[\sum (\text{row weights}) + \sum (\text{col weights})]$

▼ *Same interpretation*

- ▶ **Weight** tells you how many literals after you pull out this factor
- ▶ **Value** tells you how many lits affected in the original functions

Example

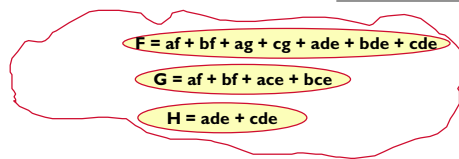


Rectangle is (a+b)

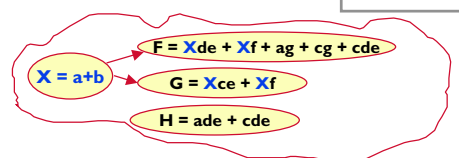
Weight is:

Value is:

Before



After



change in #lits =

Multi-Cube Extraction Strategy (Revisited)

▼ Just like with the single-cube extraction case!

- ▶ Can try to get a big, *simultaneous* set of prime rectangles
- ▶ Can try to get the *best*, biggest prime next, one at a time
- ▶ Can use a heuristic to get a *good*, big prime next, one at a time

▼ How can we really do this?

- ▶ All of the above work OK...
- ▶ ...but usually use heuristics that yank 1 prime rect (next best) at a time

© R. Rutenbar 2001

18-760, Fall 2001 43

Rudell's Ping-Pong Algorithm

▼ Ping Pong

- ▶ Idea: Try to find prime rectangles one at a time
- ▶ Try to find the next one with maximal value (== # lits saved)
- ▶ Grow the rectangles alternatively by adding more rows, more cols

▼ Outline

- ▶ Try to grow a big prime rectangle, from a good 1-row rectangle
 0. Pick the best single row (the one that makes a 1-row rectangle with the best, biggest possible value)
 1. Look at other rows that also have 1s in the same places (and more!) that you can add, one at a time, that each maximize the value of the growing rectangle. Add them until can't find any more.
 2. Now, look at other columns that also have 1s in same places (and more!) that you can add, one at a time, that max the value of the growing rectangle. Add them until you can't find any more.
 3. Goto 1.
 4. Quit when can't grow rectangle any more in any direction.

© R. Rutenbar 2001

18-760, Fall 2001 44

Ping Pong -- How Good?

▼ Very. Results from Rudell's 1989 Berkeley PhD

- ▶ **PING PONG** implements a version of the ping pong heuristic that grows a good next prime rect alternatively by rows, cols
- ▶ **BEST_RECT** uses an aggressive search to find the next **BEST** rect
- ▶ Examples are industrial circuits, goal is single cube extraction

Example	Inp	Outp	Initial #Lits	Using PING PONG #Lits	CPU	Using BEST_RECT #Lits	CPU
apex1	45	45	2,887	1,314	14	1,304	858
apex2	39	3	15,531	3,996	193	3,901	3,408
apex3	54	50	3,342	1,566	15	1,631	1,867
apex4	9	19	5,438	2,219	30	2,180	14,179
apex5	117	88	7,369	3,798	43	3,779	343
seq	41	36	3,497	1,268	14	1,254	564
TOTALS			38,064	14,161	309	14,049	21,219

© R. Rutenbar 2001

18-760, Fall 2001 45

Aside: How to We Really Do This Today

▼ ...not with rectangle covering on ALL kernels/cokernels

- ▶ Too expensive to do rectangle covering problem (**PING-PONG**) on really big circuits (10K+ gates) using complete set of kernels, co-k's

▼ Usually use heuristics to find a "quick" set of factors

- ▶ Do **NOT** completely kernel each node of the boolean network, since this could result in many, many cubes to consider in the factors
- ▶ Instead, do a "quick" factoring that gets a decent set of likely candidates for factors
- ▶ Then, can either do rectangle covering on these smaller problems (smaller since fewer factors to consider in covering problem)...
- ▶ ...or, just try to do simpler overall network restructuring, eg, try all pairwise substitutions of factors into nodes, keep the good ones, continue on in a greedy fashion

▼ We explore some of these ideas in HW3

© R. Rutenbar 2001

18-760, Fall 2001 46

Extraction: Summary

▼ Powerful, unifying theme here: *Rectangles in matrices*

▼ Single cube extraction

- ▶ Build the cube-literal matrix
- ▶ Each prime rectangle is a good single cube factor
- ▶ Weights, values allow us to estimate impact (#lits) on boolean network
- ▶ Update mechanisms for matrix to yank out one factor after another

▼ Multiple cube extraction

- ▶ Kernel all the expression in the network you start with
- ▶ Build the cokernal-cube matrix
- ▶ Each prime rectangle is a good multiple cube factor (a la Brayton-McMullen theorem about these being intersection of kernels)
- ▶ Weights, values allow us to estimate impact (#lits) on boolean network
- ▶ Update mechanisms for matrix to yank out one factor after another

▼ Mechanically, *another* covering problem!