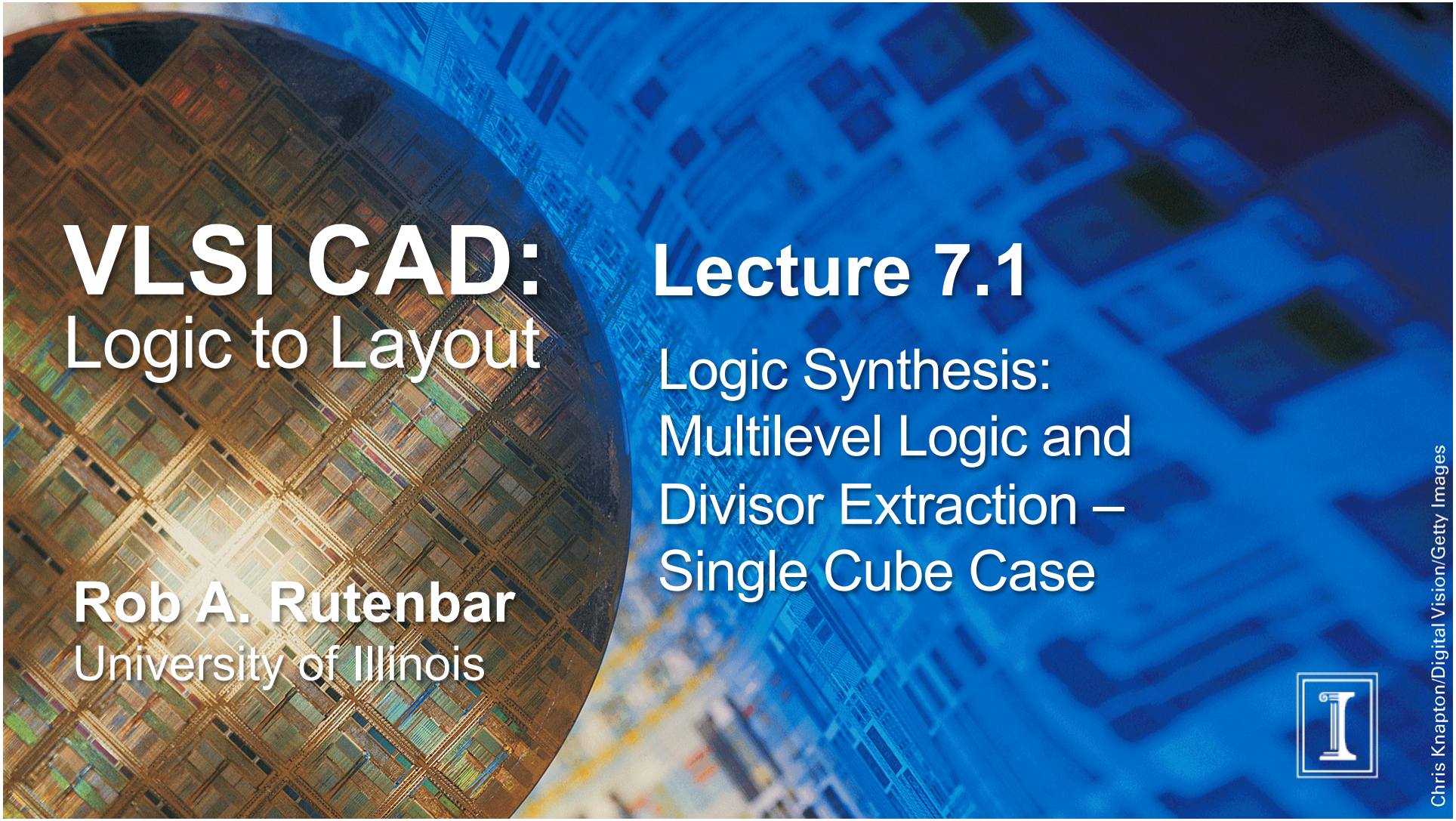


VLSI CAD: Logic to Layout

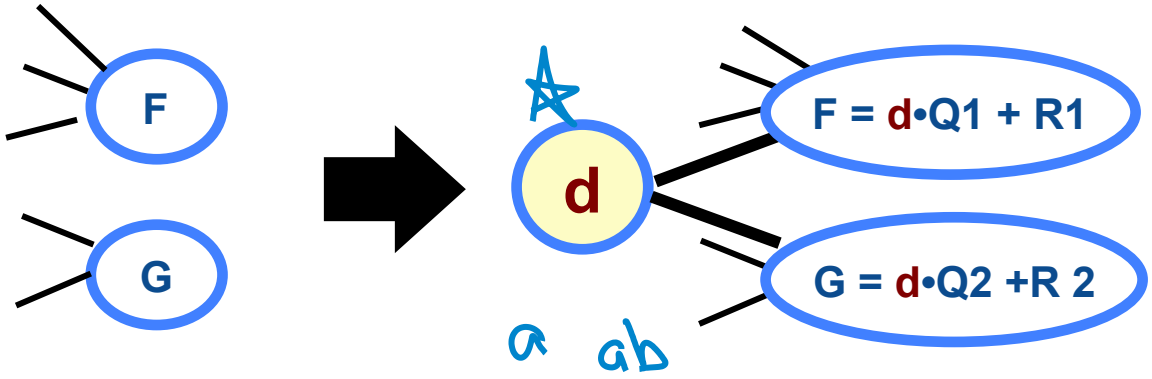
Lecture 7.1

Logic Synthesis: Multilevel Logic and Divisor Extraction – Single Cube Case

Rob A. Rutenbar
University of Illinois

How Do We *Find* Good Divisors?

- **Called: Extraction. How do we extract good divisors?**
  - Extraction of a **single-cube divisor** from multiple expressions?
  - Extraction of a **multiple-cube divisor** from multiple expressions?



- **Remember**: if we want a single-cube divisor: **Go look for co-kernels**
- **Remember**: if we want a multiple-cube divisor: **Go look for kernels**

How We Do It

- **For single cube extraction**
  - Build a very large matrix of 0s and 1s
  - Heuristically look for "prime rectangles" in this matrix
  - Each such "prime" is a good common single-cube divisor
- **For multiple cube extraction**
  - Build a (different) very large matrix of 0s and 1s
  - Heuristically look for "prime rectangles" in this matrix
  - Each such "prime" is a good multiple-cube divisor
- **Surprisingly, a lot like Kmaps(!)**
  - Except we do it all automatically algorithmically. Still, unifying and familiar idea…

Single Cube Extract: Matrix Representation

- **Given: a set of SOP algebraic model Boolean equations (P,Q,R)**
- **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 column for convenience in referring to them, later

P = abc + abd + eg
Q = abfg
R = bd + ef

	a 1	b 2	c 3	d 4	e 5	f 6	g 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	.

Example from: Richard Rudell, Logic Synthesis for VLSI Design, PhD Thesis, Dept of EECS, University of California at Berkeley, 1989.

Covering this Matrix: Prime Rectangles

- **A rectangle of this cube-literal matrix...**
  - Set of rows **R** and columns (cols) **C** that has a '1' in every row/column intersection
  - Need **not** be contiguous rows or cols in matrix. **Any set** of rows or columns
- **A *prime* rectangle:**
  - A rectangle that **cannot** be made any bigger by adding another row or a column

Prime rectangle example

	a 1	b 2	c 3	d 4	e 5	f 6	g 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	.

Result: Prime Rectangle Columns = Divisor!

- **Primes are "biggest possible" common single-cube divisors**
  - **Makes sense:** columns of (R,C) tell you the literals in the product term divisor (ie. If you AND these columns) , rows tell you which product terms you can divide!

P = abc + abd + eg
Q = abfg
R = bd + ef

	a 1	b 2	c 3	d 4	e 5	f 6	g 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	.

1 cube divisor

X = ab

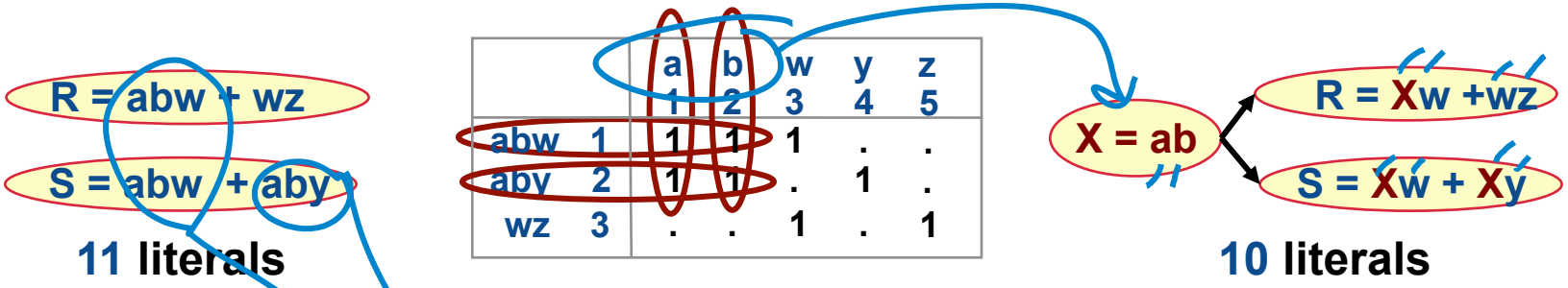
P = Xc + Xd + eg
Q = Xfg
R = bd + ef
X = ab

Simple Bookkeeping to Track Literals

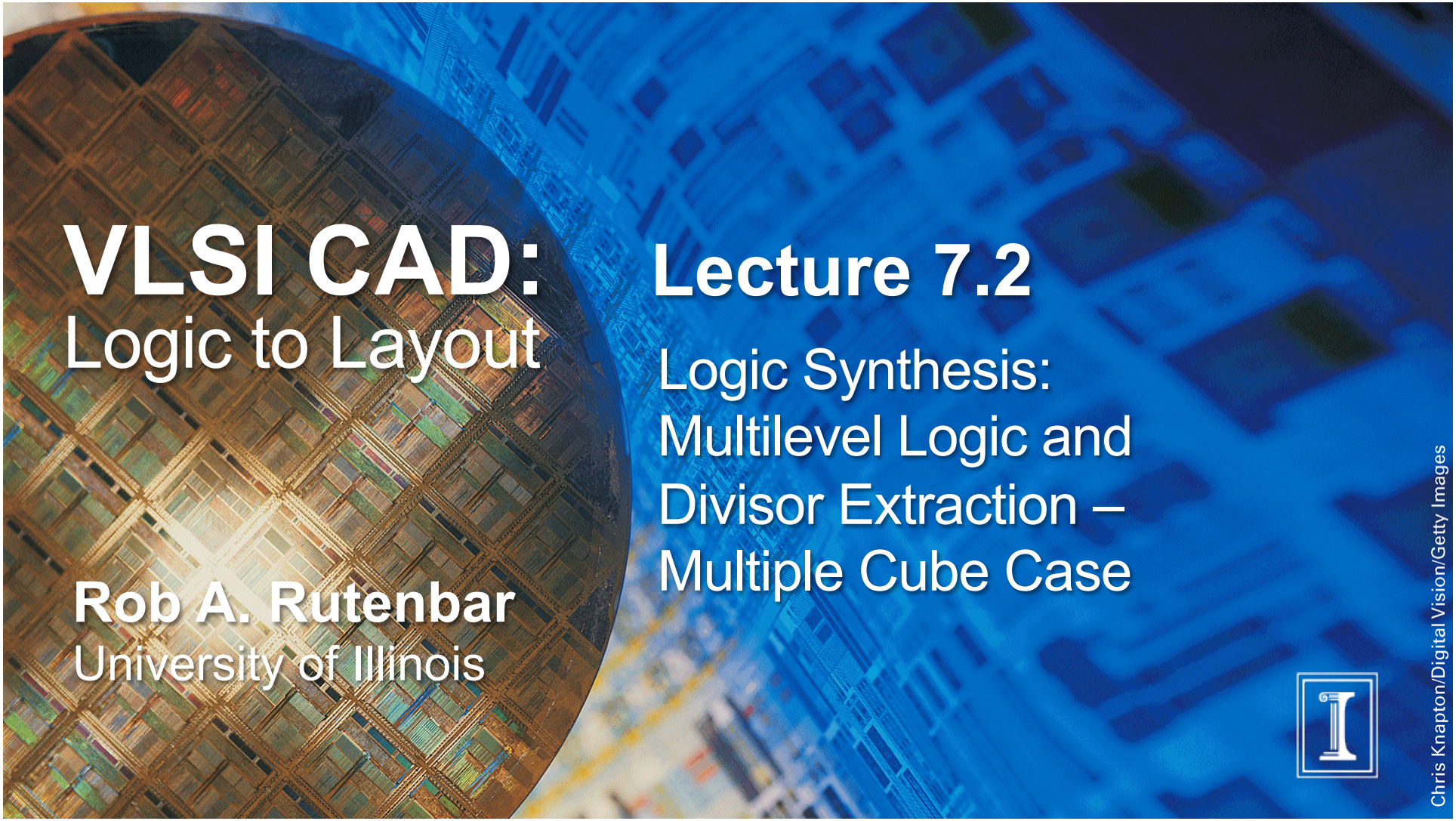
- **Recall: we factor – extract– to reduce literals in logic network**
  - Would be nice if there was a simple formula to compute this. (There is!)
  - Start with a prime rectangle
  - Let **C** = number of ("#") columns in rectangle
  - For each **row r** in rectangle: let **Weight(r)** = # times this product appears in network
  - Compute **L = (C-1) × [$\sum_{\text{rows r}}$ Weight(r)] – C**

- **Nice result: for a prime rectangle, L = # literals saved**
  - **To be precise:** if you count literals *before* extracting this divisor cube, and *after*, **L** is how many literals are saved

Compute Saved Literals, Single-Cube Extract



- **Literals saved formula:** Compute **L = (C-1) × [$\sum_{\text{rows r}}$ Weight(r)] – C**
  - **C = 2 columns**
  - **Weight(abw) = 2** (appears twice in network). **Weight(aby) = 1** (appears once)
  - **L = (2-1) (2 + 1) – 2 = 1** it works! We did, in fact, **save just 1 literal.**
  - Counts all places **ab** appeared (6 literals), the savings to replace each with **X** (3 literals), and new overhead of adding **X=ab** term (+2 literals). -6+3+2 = **1 saved**.



VLSI CAD:
Logic to Layout

Lecture 7.2

Logic Synthesis: Multilevel Logic and Divisor Extraction – Multiple Cube Case

Rob A. Rutenbar
University of Illinois

Multiple-Cube Factors...?

- **Remarkably, a very *similar* matrix-rectangle-prime concept**
  - Make an appropriate **matrix**. Find **prime rectangle**. Do literal count **bookkeeping** with numbers associated with rows/columns. It all just works! Nice, **unifying** idea.
- **Given: A set of Boolean functions (nodes in a network)**

P = af + bf + ag + cg + ade + bde + cde
Q = af + bf + ace + bce
R = ade + cde

- **First: find co-kernels and kernels of each of these functions**
  - Why? Brayton-McMullen says: Multiple-cube factors are **intersections of the product terms in the kernels** for each of these functions

Kernels / Co-Kernels of P,Q,R Example

- **P = af + bf + ag + cg + ade + bde + cde**
  - Co-kernel **(a)** kernel **(de + f + g)**
  - Co-kernel **(b)** kernel **(de + f)**
  - Co-kernel **(de)** kernel **(a + b + c)**
  - Co-kernel **(f)** kernel **(a + b)**
  - Co-kernel **(c)** kernel **(de + g)**
  - Co-kernel **(g)** kernel **(a+c)**
  - Co-kernel **(1)** kernel **(af + bf + ag + cg + ade + bde + cde)** trivial, **ignore**
- **Q = af + bf + ace + bce**
  - Co-kernel **(a)** or co-kernel **(b)** kernel **(ce + f)** (can have > 1 co-kernel)
  - Co-kernel **(f)** or co-kernel **(ce)** kernel **(a + b)** (ditto)
  - Co-kernel **(1)** kernel **(af + bf + ace + bce)** trivial, **ignore**
- **R = ade + cde**
  - Co-kernel **(de)** kernel **(a + c)**
  - (Note, **R** is not its own kernel; why not?)

Example from: Richard Rudell, Logic Synthesis for VLSI Design, PhD Thesis, Dept of EECS, University of California at Berkeley, 1989.

New Matrix: Co-Kernel--Cube Matrix

P	co-kernel	(a)	kernel	(de + f + g)
P	co-kernel	(b)	kernel	(de + f)
P	co-kernel	(de)	kernel	(a + b + c)
P	co-kernel	(f)	kernel	(a + b)
P	co-kernel	(c)	kernel	(de + g)
P	co-kernel	(g)	kernel	(a+c)
Q	co-kernel	(a)	kernel	(ce + f)
Q	co-kernel	(b)	kernel	(ce + f)
Q	co-kernel	(f)	kernel	(a + b)
Q	co-kernel	(ce)	kernel	(a + b)
R	co-kernel	(de)	kernel	(a + c)

One row for each **unique (Function, Co-kernel) pair** in problem

One column for each **unique cube** among all kernels in problem

			a 1	b 2	c 3	ce 4	de 5	f 6	g 7
P	a	1							
P	b	2							
P	de	3							
P	f	4							
P	c	5							
P	g	6							
Q	a	7							
Q	b	8							
Q	ce	9							
Q	f	10							
R	de	11							

What goes in the individual (row, column) entries here in the matrix?

Entries in the Co-Kernel--Cube Matrix

- From the **row**, take the **co-kernel**, go look at the associated kernel
- Look at **cubes** in this kernel; put **"1"** in **columns** that are cubes in this kernel; else put **"."**

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

P co-kernel **(a)** kernel **(de + f + g)**

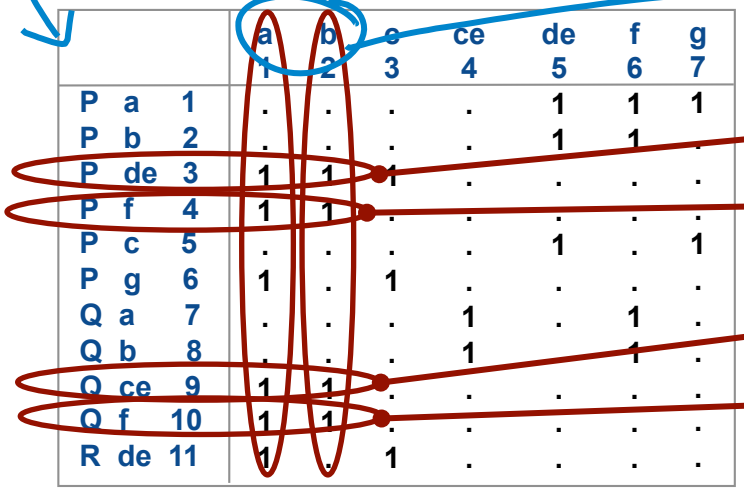
P co-kernel **(g)** kernel **(a+c)**

			a 1	b 2	c 3	ce 4	de 5	f 6	g 7
P	a	1	.	.	.	.	1	1	1
P	b	2	.	.	.	.	1	1	.
P	de	3	1	1	1	.	.	.	.
P	f	4	1	1	.	.	.	.	.
P	c	5	.	.	.	.	1	.	1
P	g	6	1	.	1	.	.	.	.
Q	a	7	.	.	.	1	.	1	.
Q	b	8	.	.	.	1	.	1	.
Q	ce	9	1	1	.	.	.	.	.
Q	f	10	1	1	.	.	.	.	.
R	de	11	1	.	1	.	.	.	.

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

Prime Rectangles in Co-Kernel--Cube Matrix

- **Prime rectangle is again a good divisor: now multiple cube(!)**
  - Create the multiple cube divisor as **sum** (OR) of cubes of prime rectangle **columns**



		a 1	b 2	c 3	ce 4	de 5	f 6	g 7
P	a 1	.	.	.	.	1	1	1
P	b 2	.	.	.	.	1	1	.
P	de 3	1	1	1	.	.	.	.
P	f 4	1	1	.	.	.	.	.
P	c 5	.	.	.	.	1	.	1
P	g 6	1	.	1	.	.	.	.
Q	a 7	.	.	.	1	.	1	.
Q	b 8	.	.	.	1	.	1	.
Q	ce 9	1	1	.	.	.	.	.
Q	f 10	1	1	.	.	.	.	.
R	de 11	1	.	1	.	.	.	.

Interpretation

P = (co-kernel **de**) • (a + b + stuff1) + rem1

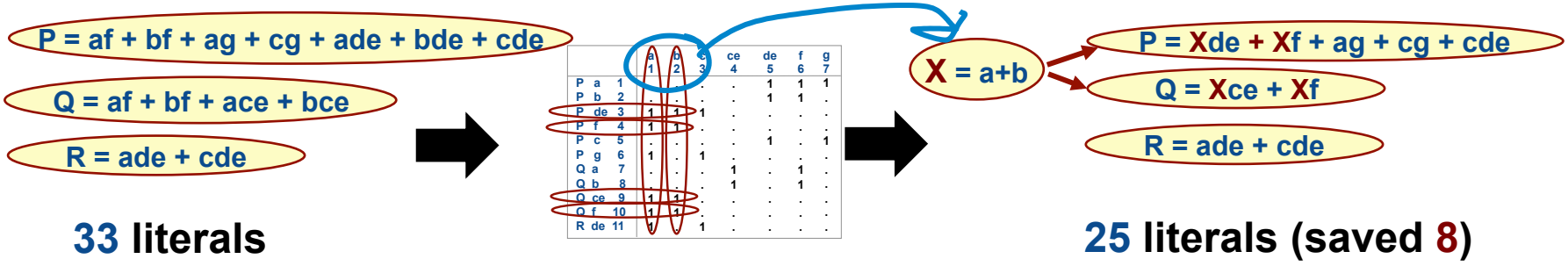
P = (co-kernel **f**) • (a + b + stuff2) + rem2

Q = (co-kernel **ce**) • (a + b + stuff3) + rem3

Q = (co-kernel **f**) • (a + b + stuff4) + rem4

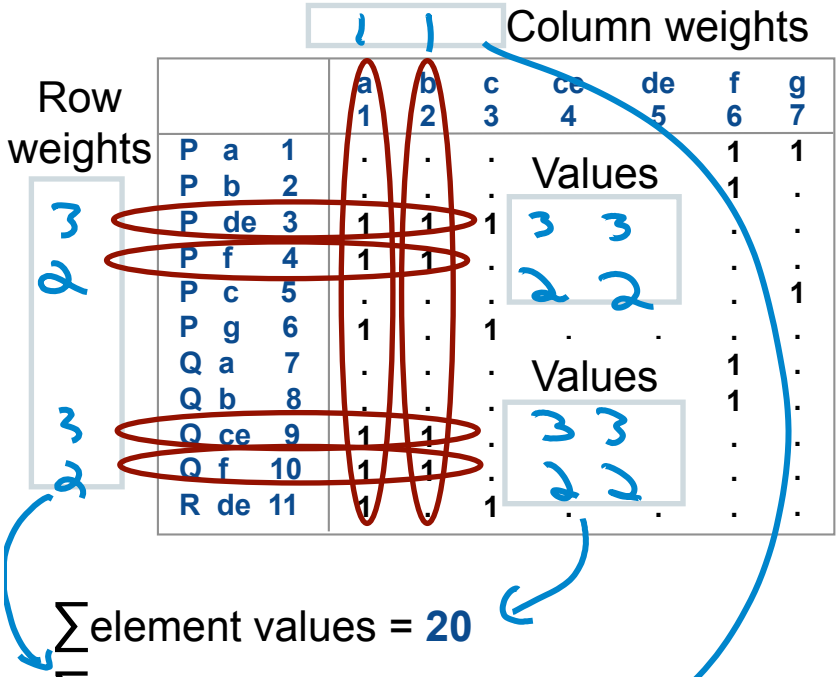
OR together cubes of the **columns** of the prime rectangle – **this** is the divisor **(a+b)**!

Multiple-Cube Extraction: # Literals Saved



- **Similar formula**
  - For each **column c** in rectangle: let **Weight(c)** = # literals in column cube
  - For each **row r** in rectangle: let **Weight(r)** = 1 + # literals in co-kernel label
  - For each **"1"** covered at **row r**, **column c**: AND row co-kernel and column cube. Let **Value(r,c)** = # literals in this new ANDed product
  - Compute $L = \sum_{\text{rows r, columns c}} \text{Value(r,c)}] - \sum_{\text{rows r}} \text{Weight(r)} - \sum_{\text{column c}} \text{Weight(c)}$
  - **Result: L** is the number of **saved literals** from this extraction

Multiple-Cube Extraction: # Literals Saved



∑element values = **20**

∑row weights = **10**

∑column weights = **2**

(value sum) – (row sum) – (column sum) = **20-10-2=8**

Before **33 literals**

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

Q = af + bf + ace + bce

R = ade + cde

After **25 literals**

X = a+b

P = Xde + Xf + ag + cg + cde

Q = Xce + Xf

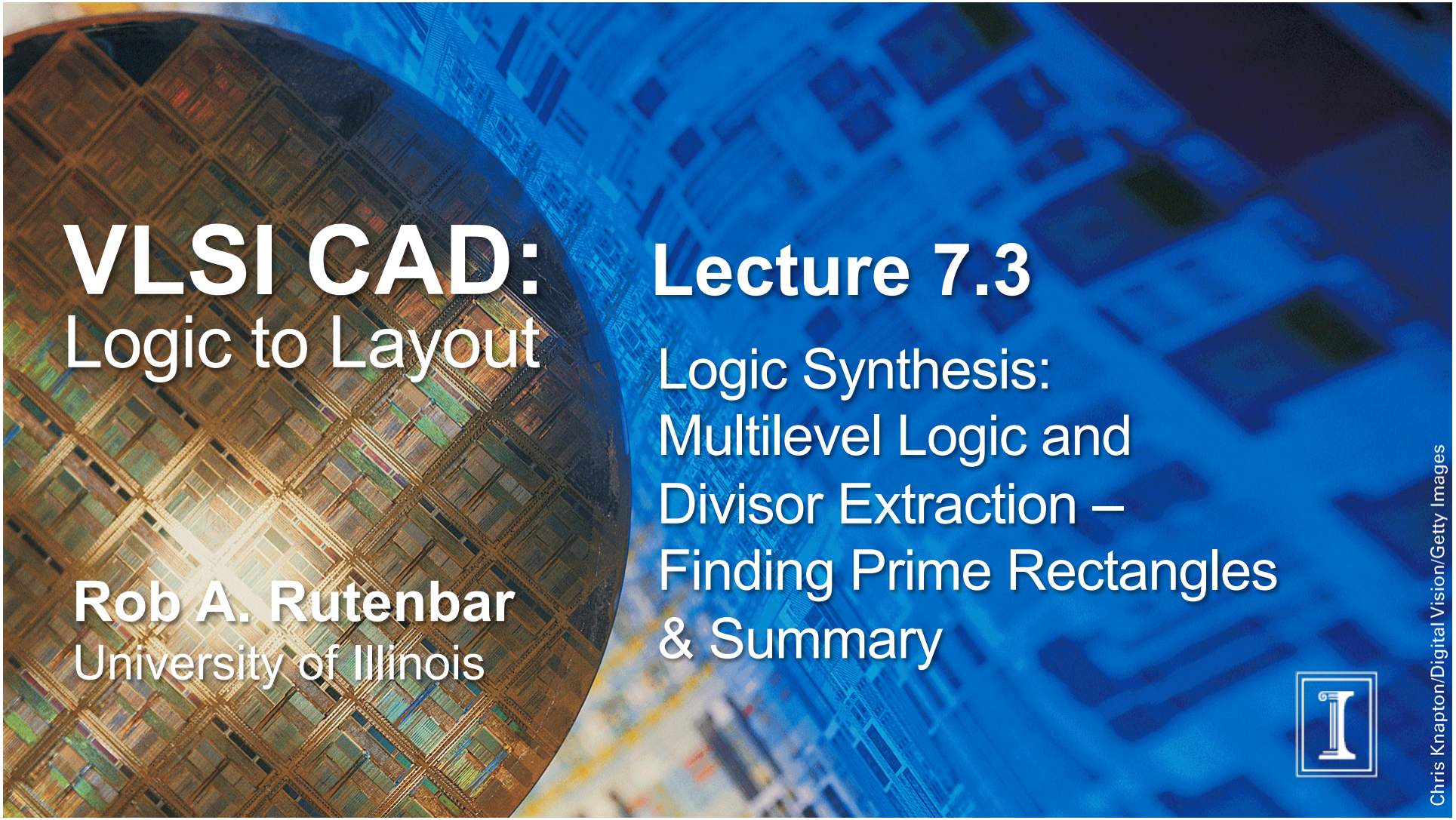
R = ade + cde

Change in #literals = 33 – 25 = **8** !

Details for Both Single/Multiple Cube Extract

- **You can extract a *second, third*, etc, divisor using same matrix**
  - Works for the single-cube divisors
  - Works for the multiple-cube divisors

- **…but must update matrix to reflect new Boolean logic network**
  - Because the node contents are different, and there is a new divisor node in network
  - Like adding "don't cares" to a Kmap: some things no longer essential to cover
  - And for multiple-cube case, must kernel new divisor nodes to update matrix
  - All mechanical. A bit tedious, to be honest. Just skip it for this presentation
  - **For us:** just know how to **extract first good divisor**, this is good enough



VLSI CAD: Logic to Layout

Lecture 7.3

Logic Synthesis:
Multilevel Logic and
Divisor Extraction –
Finding Prime Rectangles
& Summary

Rob A. Rutenbar
University of Illinois

How to *Find* a Prime Rectangle in Matrix?

- **Greedy heuristics work well for this rectangle covering problem**
  - Start with a single row rectangle with "good #literal savings"
  - Grow the rectangle alternatively by adding more rows, more columnss
- **Example: Rudell's Ping Pong heuristic (from his '89 Berkeley PhD)**
  - To grow a big prime rectangle, from a good 1-row rectangle
    1. Pick the **best single row** (the 1-row rectangle with best #literals saved)
    2. Look at **other rows with 1s** in same places (and more!) that you can add, one at a time, that each maximize #literals saved. Add them until can't find any more.
    3. Look at **other columns with 1s** in same places (and more!) that you can add, one at a time, that maximize # literals saved. Add them until you can't find any more.
    4. Goto 2.
    5. Quit when cannot grow rectangle any more in any direction.

Ping Pong Heuristic -- How Good?

- **Very. Results from Rudell's 1989 Berkeley PhD**
  - **PING PONG:** this heuristic that grows a good prime alternatively by rows, columns
  - **BEST RECT:** uses an aggressive search to find the ideal **best** rectangle
  - Examples are industrial circuits, goal is single cube extraction

				Using PINGPONG		Using BESTRECT	
Ex	Inputs	Outputs	Initial #Literals	#Literals	CPU	#Literals	CPU
apex1	45	45	2,887	1,314	14	1,304	858
apex2	39	3	15,531	3,996	193	3,901	3,408
apex5	117	88	7,369	3,798	43	3,779	343

  - Relevant result: within **<1% of optimal**, **10-100x faster** than brute force

Extraction: Summary

- **Single cube extraction**
  - Build the cube-literal matrix; prime rectangle is a good **single cube divisor**
  - Simple bookkeeping lets us **estimate savings in #literals** in Boolean network

- **Multiple cube extraction**
  - Kernel all the expressions in network; build the co-kernel--cube matrix
  - Each prime rectangle is a good **multiple cube divisor**
  - Simple bookkeeping lets us **estimate savings in #literals** in Boolean network

- **Mechanically, both rectangle cover problems (very like Kmaps!)**
  - Good **heuristics** for solving for a good prime rectangle, fast and effective
  - Ways to extract *more* than one divisor from network (but we didn't cover these)

Aside: How to We Really Do This?

- **…*not* with rectangle covering on *all* kernels/co-kernels**
  - Too expensive to do rectangle problem on big circuits (> 20K gates)
  - Too expensive to go compute **complete** set of kernels, co-kernels

- **Often use heuristics to find a "quick" set of likely divisors**
  - Don't fully kernel each node of network: too many cubes to consider here
    - Instead, can extract a ***subset*** of useful kernels quickly
  - Then, can either do rectangle cover on these smaller problems (smaller since fewer things to consider in covering problem)…
  - …or, try to do simpler overall network restructuring, eg, try all pairwise substitutions of one node into another node: keep good ones, continue in a greedy way

Notes:

- **Good references to read about all these ideas**
  - R.K. Brayton, R. Rudell, A. Sangiovanni-Vincentelli, A.R. Wang, “MIS: A Multiple-Level Logic Optimization System,” *IEEE Transactions on CAD of ICs*, vol. CAD-6, no. 6, November 1987, pp. 1062-1081.
  - Giovanni De Micheli, *Synthesis and Optimization of Digital Circuits*, McGraw-Hill, 1994.
  - R.K. Brayton, R. Rudell, A.S. Vincentelli, and A. Wang, "Multi-Level Logic Optimization and the Rectangular Covering Problem," Proceedings of the International Conference on Computer Aided Design, pp. 66-69, 1987.
  - Richard Rudell, *Logic Synthesis for VLSI Design*, PhD Thesis, Dept of EECS, University of California at Berkeley, 1989.
  - Srinivas Devadas, Abhijit Gosh, Kurt Keutzer, *Logic Synthesis*, McGraw Hill Inc., 1994.