ECE6703J

Computer-Aided Design of Integrated Circuits

Multi-Level Logic Synthesis:

Extraction

Outline

- Single-cube Extraction

- Multiple-cube Extraction

How Do We Find Good Divisors?

- The operator is called **extraction**.
  - Want to extract either **single-cube divisor** or **multiple-cube divisor** from multiple expressions.
- How do we **extract** good divisors?
- Solution:
  - When you want a **single-cube divisor**, go look for **co-kernels**.
  - When you want a **multiple-cube divisor**, go look for **kernels**.

Approach Overview

- For **single cube extraction**
  - Build a very large matrix of **0s and 1s**
  - Heuristically look for "**prime rectangles**" in this matrix
  - Each such "prime" gives 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" gives a good multiple-cube divisor

Single Cube Extract: Matrix Representation

- **<u>Given</u>**: a set of SOP Boolean equations
- Construct the **cube-literal matrix** as follows:
  - One row for each **unique** product term.
  - One column for each unique literal.
  - A "1" in the matrix if this product term uses this literal, else a "-".

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

		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	-

Covering this Matrix: Prime Rectangles

- A **rectangle** of a cube-literal matrix is a set of rows R and columns C that has a ‘1’ in **<u>every</u> row/column intersection**.
  - Need not be contiguous rows or columns in matrix. Any set of rows or columns is fine.

		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	-

Covering this Matrix: Prime Rectangles

- A **prime rectangle** is a rectangle that cannot be made any bigger by adding another row or a column.

		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	-

Prime Rectangle Columns = Divisor!

- **Primes** are "biggest possible" common single-cube divisors.
  - **Makes sense**: columns of the prime rectangle tell you the literals in the single-cube divisor, while rows tell you which product terms you can divide!

		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	-

Single-cube divisor:

$X = ab$

Prime Rectangle Columns = Divisor!

		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	-

Single-cube divisor:

$X = ab$

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

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

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 the number of reduced literals.
- Indeed, there is:
  - Start with a prime rectangle.
  - Let C = # columns in rectangle.
  - For each row r in rectangle: let $\text{Weight}(r)$ = # times this product appears in network.
  - Compute $L = (C - 1) \times [\sum_{\text{rows } r} \text{Weight}(r)] - C$.
- **Nice result**: for a prime rectangle, L = **# literals saved**
  - **To be precise**: if you count literals before extracting this single-cube divisor, and after, L is how many literals are saved.

Compute Saved Literals: Example

$R = abw + wz$

$S = abw + aby$

Build Matrix

	a 1	b 2	w 3	y 4	z 5
abw 1	1	1	1	-	-
wz 2	-	-	1	-	1
aby 3	1	1	-	1	-

Extraction

$$\begin{tikzcd} & R = Xw + wz \\ X = ab \arrow[ur] \arrow[dr] & \\ & S = Xw + Xy \end{tikzcd}$$

Original # literals: 11

# saved: 1

After extraction # literals: 10

Compute Saved Literals: Example

$R = abw + wz$

$S = abw + aby$

	a	b	w	y	z
	1	2	3	4	5
abw 1	1	1	1	-	-
wz 2	-	-	1	-	1
aby 3	1	1	-	1	-

Result by Counting:
# saved: 1

- Now apply formula $L = (C - 1) \times [\sum_{\text{rows } r} \text{Weight}(r)] - C$
  - C = # columns in rectangle $\Rightarrow 2$
  - Weight(abw) $\Rightarrow 2$ (appear twice in the network)
  - Weight(aby) $\Rightarrow 1$ (appear once in the network)
  - $L = (2 - 1) \times (2 + 1) - 2 = 1$ **Correct!**

Outline

- Single-cube Extraction
- Multiple-cube Extraction

How About Multiple-Cube Factors?

- Remarkably, a very similar matrix-prime-rectangle concept.
  - Make an appropriate **matrix**. Find **prime rectangle**. Do literal count **bookkeeping** with numbers associated with rows/columns.
- 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 **kernels** of each of these functions.
  - Why? Brayton-McMullen theorem: Multiple-cube factors are **intersections of the expressions 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 c, kernel $de + g$
  - Co-kernel de, kernel $a + b + c$
  - Co-kernel f, kernel $a + b$
  - Co-kernel g, kernel $a + c$
  - Co-kernel 1, kernel $af + bf + ag + cg + ade + bde + cde$ **(trivial, ignore)**

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

- $Q = af + bf + ace + bce$
  - Co-kernel a, kernel $ce + f$
  - Co-kernel b, kernel $ce + f$
  - Co-kernel ce, kernel $a + b$
  - Co-kernel f, kernel $a + b$
  - 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?

New Matrix: Co-Kernel-Cube Matrix

- One row for each **unique** (function, co-kernel) **pair** in problem.
- One column for each **unique cube** among all kernels in problem.

P: co-kernel a, kernel $de + f + g$
P: co-kernel b, kernel $de + f$
P: co-kernel c, kernel $de + g$
P: co-kernel de, kernel $a + b + c$
P: co-kernel f, kernel $a + b$
P: co-kernel g, kernel $a + c$
Q: co-kernel a, kernel $ce + f$
Q: co-kernel b, kernel $ce + f$
Q: co-kernel ce, kernel $a + b$
Q: co-kernel f, kernel $a + b$
R: co-kernel de, kernel $a + c$

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

Entries in the Co-Kernel-Cube Matrix

- For each **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: co-kernel a, kernel $de + f + g$
P: co-kernel b, kernel $de + f$
P: co-kernel c, kernel $de + g$
P: co-kernel de, kernel $a + b + c$
P: co-kernel f, kernel $a + b$
P: co-kernel g, kernel $a + c$
Q: co-kernel a, kernel $ce + f$
Q: co-kernel b, kernel $ce + f$
Q: co-kernel ce, kernel $a + b$
Q: co-kernel f, kernel $a + b$
R: co-kernel de, 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	c	3	-	-	-	-	1	-	1
P	de	4	1	1	1	-	-	-	-
P	f	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	-	-	-	-

Entries in the Co-Kernel-Cube Matrix

- Each row gives the kernel of the function (e.g., P) obtained by dividing the co-kernel (e.g., a).

P: co-kernel a, kernel $de + f + g$

			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	c	3	-	-	-	-	1	-	1
P	de	4	1	1	1	-	-	-	-
P	f	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	-	-	-	-

Prime Rectangles in Co-Kernel-Cube Matrix

- Prime rectangle is again a good divisor: now multiple cubes
  - 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	c	3	-	-	-	-	1	-	1
P	de	4	1	1	1	-	-	-	
P	f	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	-	-	-	-

(a+b) is a multiple cube divisor!

P = (de)•(a+b) + stuff1

P = (f)•(a+b) + stuff2

Q = (ce)•(a+b) + stuff3

Q = (f)•(a+b) + stuff4

Final Result

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

$Q = af + bf + ace + bce$

$R = ade + cde$

Original # literals: 33

(a+b) is a multiple cube divisor!

From Rectangle

P = (de)•(a+b) + stuff1

P = (f)•(a+b) + stuff2

Q = (ce)•(a+b) + stuff3

Q = (f)•(a+b) + stuff4

Extraction

$$\begin{tikzcd}
 & P = Xf + Xde + ag + cg + cde \\
X = a + b \arrow[ur] \arrow[r] \arrow[dr] & Q = Xf + Xce \\
 & R = ade + cde
\end{tikzcd}$$

After extraction # literals: 25

# saved: 8

Simple Formula to Get # Literals Saved

- For each column c in rectangle: let $\text{Weight}(c) = \#$ literals in column cube.
- For each row r in rectangle: let $\text{Weight}(r) = 1 + \#$ literals in co-kernel label.
- For each "1" covered at row r and column c: AND row co-kernel and column cube; let $\text{Value}(r, c) = \#$ literals in this new ANDed product.
- **# literals saved** is

$$L = \sum_{\text{row } r} \sum_{\text{col } c} \text{Value}(r, c) - \sum_{\text{row } r} \text{Weight}(r) - \sum_{\text{col } c} \text{Weight}(c)$$

Compute Saved Literals: Example

			a	b	c	ce	de	f	g
			1	2	3	4	5	6	7
P	a	1	-	-	-	-	1	1	1
P	b	2	-	-	-	-	1	1	-
P	c	3	-	-	-	-	1	-	1
P	de	4	1	1	1	-	-	-	-
P	f	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	-	-	-	-

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

$Q = af + bf + ace + bce$

$R = ade + cde$

# saved: 8

- Column weight
  - Weight(a) = #literals in "a" $\Rightarrow$ 1
  - Weight(b) = #literals in "b" $\Rightarrow$ 1
- Row weight
  - Weight((P, de)) = 1+ #literals in "de" $\Rightarrow$ 3
  - Weight((P, f)) = 1+ #literals in "f" $\Rightarrow$ 2
  - Weight((Q, ce)) = 1+ #literals in "ce" $\Rightarrow$ 3
  - Weight((Q, f)) = 1+ #literals in "f" $\Rightarrow$ 2

Compute Saved Literals: Example

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

Values

# saved: 8

- Column weight
  - Weight(a) = 1; Weight(b) = 1
- Row weight
  - Weight((P, de)) = 3; Weight((P, f)) = 2
  - Weight((Q, ce)) = 3; Weight((Q, f)) = 2
- Value(r,c): # literals in the **product** of **row co-kernel** and **column cube**.
- Apply formula $L =$

$$\sum_{\text{row } r}\sum_{\text{col } c}\text{Value}(r,c) - \sum_{\text{row } r}\text{Weight}(r) - \sum_{\text{col } c}\text{Weight}(c)$$

$$= 20 - 10 - 2 = 8$$

Correct!

Details for Both Single/Multiple Cube Extraction

- You can extract a **second**, **third**, etc., divisor using same matrix.
  - Works for both single-cube and 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.
  - For multiple-cube case, must **kernel** new divisor nodes to update matrix.
  - All mechanical. A bit tedious. Just skip it…
  - For us: just know how to **extract first good divisor** is good enough.

How to Find 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 by adding more rows, more columns.

- Example: **Rudell's Ping Pong heuristic**.
  - From his Berkeley PhD dissertation in 1989.
  - **Very good** heuristic:
    - **< 1%** of optimal result.
    - **10~100x faster** than brute force approach.

Extraction: Summary

- **Single cube extraction**
  - Build the cube-literal matrix.
  - Each prime rectangle is a good **single cube divisor**.
  - Simple bookkeeping lets us obtain savings in #literals.
- **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 obtain savings in #literals.
- Mechanically, both are **rectangle covering** problems
  - Good **heuristics** to obtain a good prime rectangle, fast and effective.

Aside: How do We Really Do This?

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

- Often use heuristics to find a "**quick**" set of likely divisors.
  - Don't fully kernel each node of network: too many cubes to consider. 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, e.g., try all pairwise **substitutions** of one node into another node: keep good ones, continue in a greedy way.

References

- 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.