ECE/CS 5740/6740
CAD of Digital Circuits: Logic Synthesis & Optimization

Boolean Function Decomposition

Priyank Kalla
Professor
Electrical & Computer Engineering
http://www.ece.utah.edu/~kalla

Multi-Level Logic Synthesis

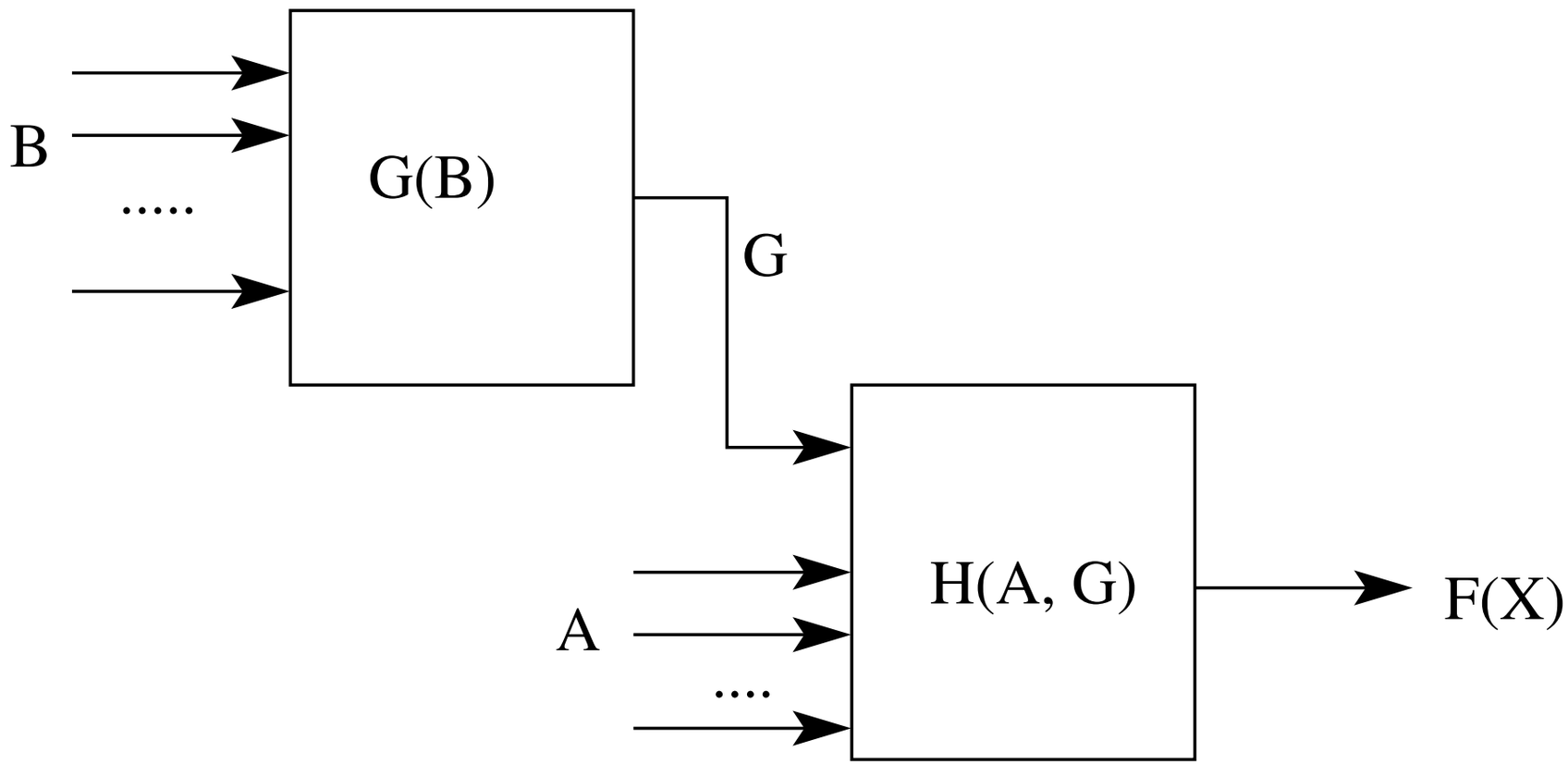
- Functional Decomposition: Given a Boolean function f, decompose it into sub-functions $g, h, \ldots$
- The total cost of the implementation of sub-functions should reduce (in terms of # of literals, delay of the circuit, etc.)
- It should be targeted to the technology: FPGAs, or CMOS
- Functional Decomposition techniques are either algebraic or Boolean
  - Algebraic: Treat literals $x, \overline{x}$ as distinct, algorithms mimic algebraic factorization as polynomials, not very good with don't cares. SIS mostly uses algebraic techniques, some Boolean techniques
  - Boolean: Exploit full-scope of Boolean algebras, more powerful. ABC uses more Boolean techniques, and a few algebraic ones

Boolean Functional Decomposition

- Functional Decomposition can be classified (roughly) into the following types
  - Simple Disjunctive Decomposition
  - Generalize to Ashenhurst-Curtis Decomposition
  - Bi-Decomposition

Simple Disjunctive (and Disjoint) Decomposition

Given function $F(X)$, variables $X = \{x_1, \ldots, x_n\}$
Disjoint partition of X as $X = A \cup B$
Compute $G(B)$ and $H(A, G(B))$ such that $F(X) = H(A, G(B))$



B = Bound–set, A = Free–set

Simple Disjunctive (and Disjoint) Decomposition

- Example of Simple Disjunctive Decomposition
  - f = w'x'z' + wx'z + w'yz + wyz'
  - (w'z')x' + (wz)x' + (w'z)y + (wz')y
  - f= F(ø(w, z), x, y)
    - Ø = XNOR(w, z);
    - Ø' = XOR(w, z)
    - Orthonormal Expansion of f w.r.t. Ø= ø x' + ø' y

- Does this remind you of something we have studied?

Simple Disjunctive Decomposition

- Draw a K-map corresponding to this function $f = w'x'z' + wx'z + w'yz + wyz'$
- Given: Bound-set = {w, z}, and Free-set = {x, y}
- Put bound-set on the columns, free-set on the rows

Bound set = {w, z}					
		00	01	11	10
Free set ={x,y}	00	1	0	1	0
	01	1	1	1	1
	11	0	1	0	1
	10	0	0	0	0

Simple Disjunctive Decomposition

Decomposition through partition matrix

Test for existance of a decomp:

Reduce partition matrix, get column multiplicity

If col. Mult. <= 2, you WILL FIND A DECOMP!

Bound set = {w, z}					
		00	01	11	10
Free set ={x,y}	00	1	0	1	0
	01	1	1	1	1
	11	0	1	0	1
	10	0	0	0	0

- Column multiplicity = number of distinct column patterns

Significance of Column Multiplicity

- If column multiplicity = 2
  - Bound-set function corresponds to two values 0 and 1, or the function $\phi, \overline{\phi}$
    - Which means a 1-bit encoding for ϕ exists
    - This gives a simple decomposition with an orthonormal basis $f = \phi A + \phi' B$
- How to compute ϕ, A, B?
- Merge identical columns, one column is ϕ, the other ϕ'

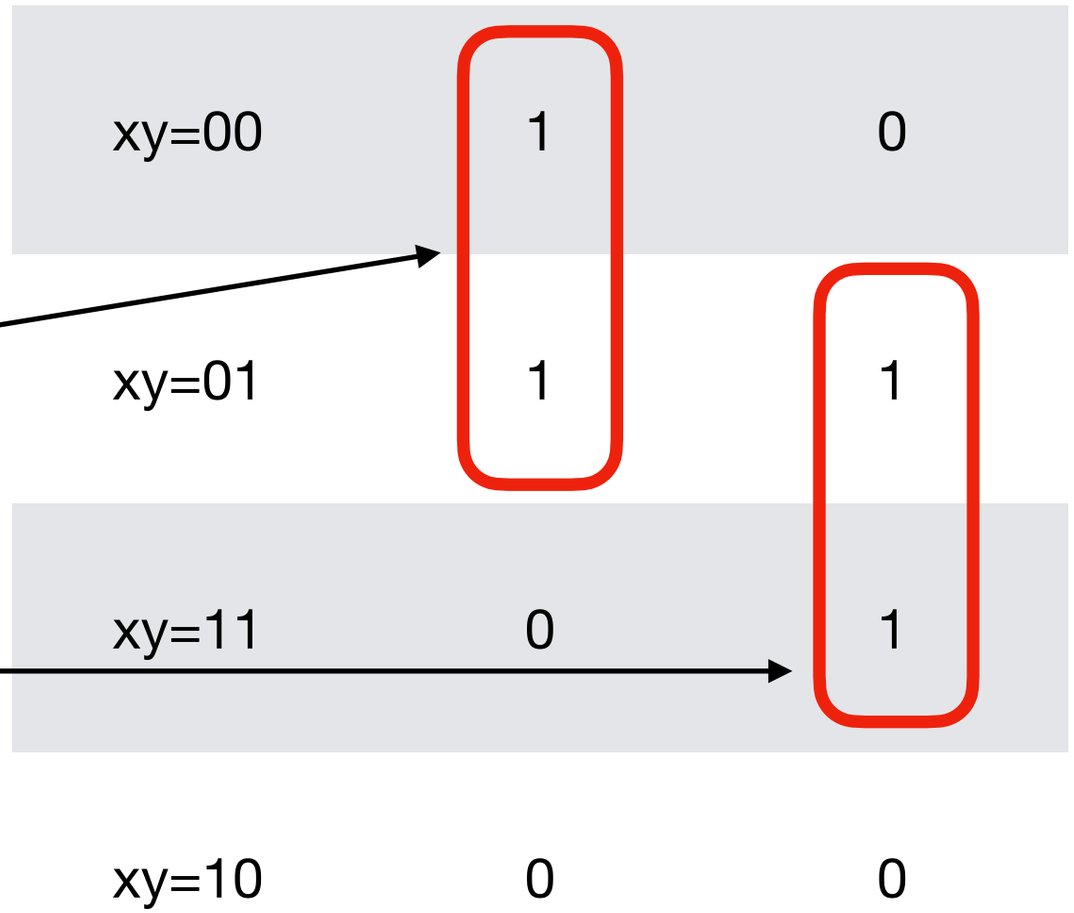
Compute Simple Decomposition

- Columns:
  $\phi = w'z' + wz = w \overline{\oplus} z$
- $\phi = w \oplus z$
- Rows = A, B

	wz={00,11}	wz={01,10}
xy=00	1	0
xy=01	1	1
xy=10	0	1
xy=11	0	0

Compute Simple Decomposition

- Columns:
  $\phi = w'z' + wz = w \overline{\oplus} z$
- $\phi = w \oplus z$
- Rows = A, B
- $A = x'$
- B = y
- $f = \phi A + \phi' B = \phi x' + \phi' y$



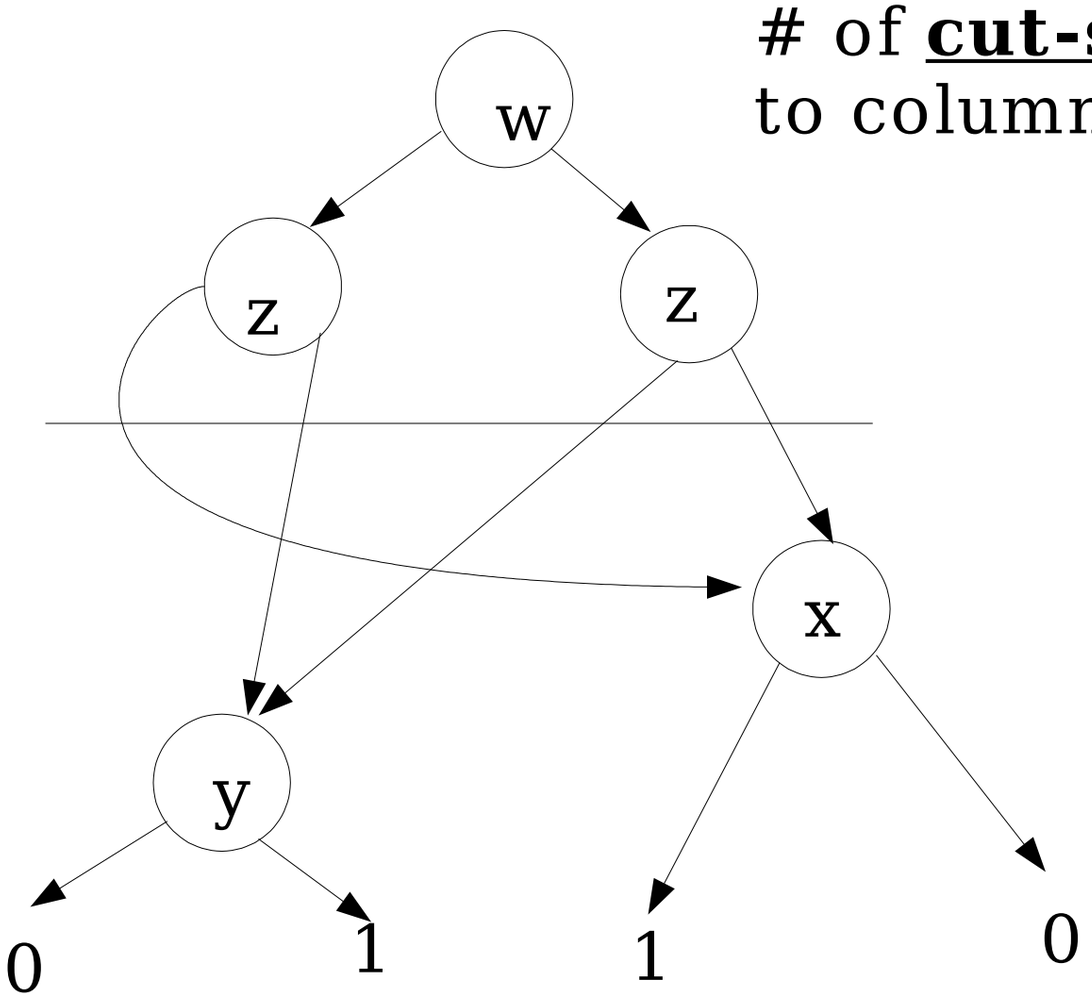
Simple Decomposition May or May Not Exist

0	1	0	1
0	0	1	0
0	0	1	0
0	1	0	1

- Column multiplicity? = 3
- What about row multiplicity? =2!! Transpose the matrix
- Sometimes, you have to try all bound/free-set variable combinations, still column multiplicity maybe more than 2. Then simple decomposition does not exist under any combination.
- Then you can generalize it to “generalized co-factor”

Simple Decomp on BDDs: Basic Concepts

F = w'z'x' + wzx' + w'zy + wz'y
F = H(G(w, z), x, y)



of **cut-set nodes** corresponds to column multiplicity: BDD MAGIC

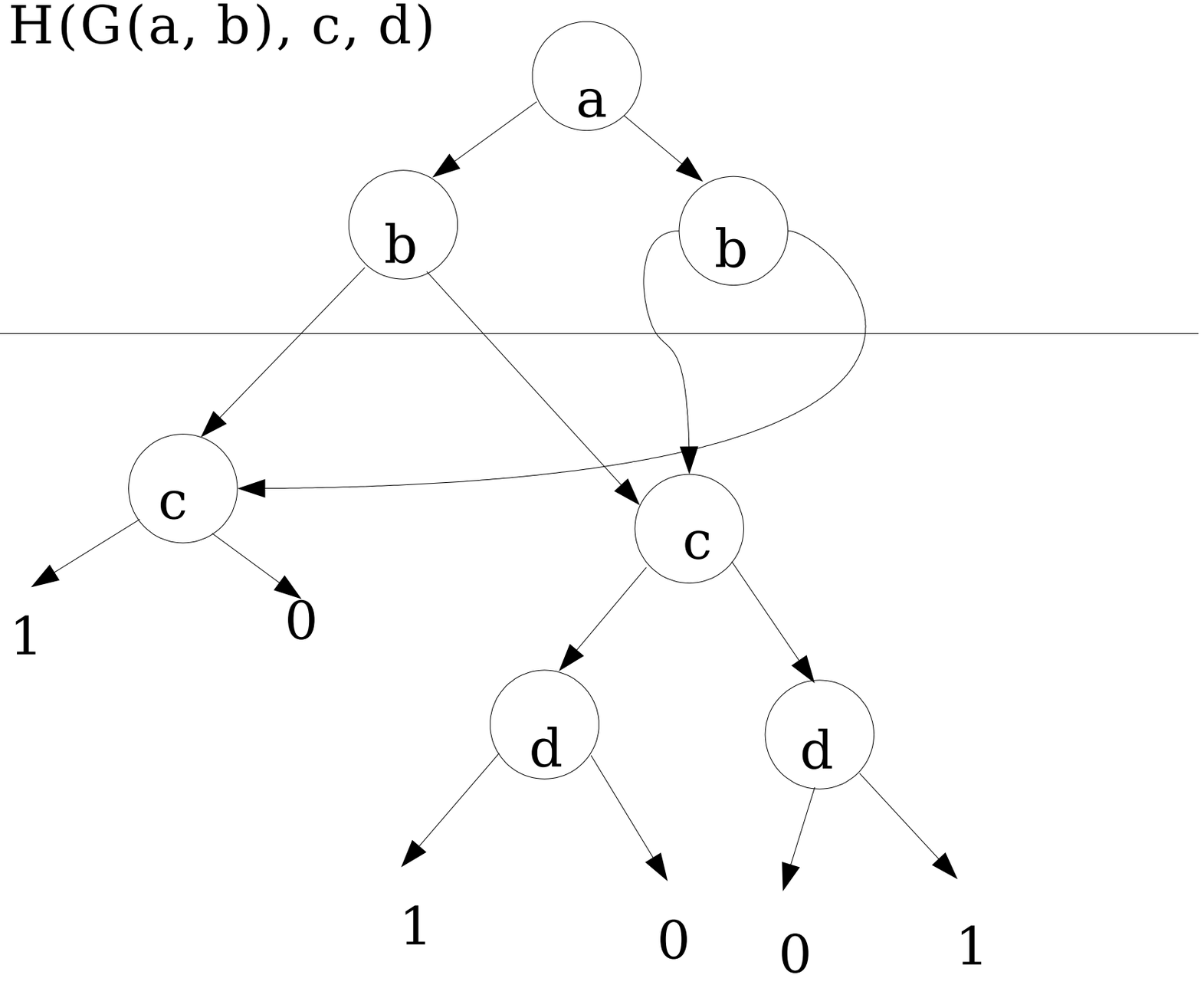
Free set term: x'
Associated bound-set cubes:

Free-set term: y
Bound-set cubes:

Simple Decomp on BDDs: Basic Concepts

F = a'b'c' + a'bc'd' + a'bcd + abc' + ab'cd' + ab'cd

F = H(G(a, b), c, d)



BDD-Based Simple Decomp. Algorithm

- F = H(G(w, z), x, y)
- Build BDD for F.
- Bound set vars together on top
- Free set on Bottom
- Perform a “CUT”: Partitioning of vars
- #cut-set nodes <= 2: orthonormal expansion corresponding to simple decomp exists! Problem solved!
- **But what if #cut-set nodes = 3 ?**