Binary Decision Diagram

Virendra Singh

Associate Professor

Computer Architecture and Dependable Systems Lab

Department of Electrical Engineering

Indian Institute of Technology Bombay

http://www.ee.iitb.ac.in/~viren/

E-mail: viren@ee.iitb.ac.in

EE-677: Foundation of VLSI CAD

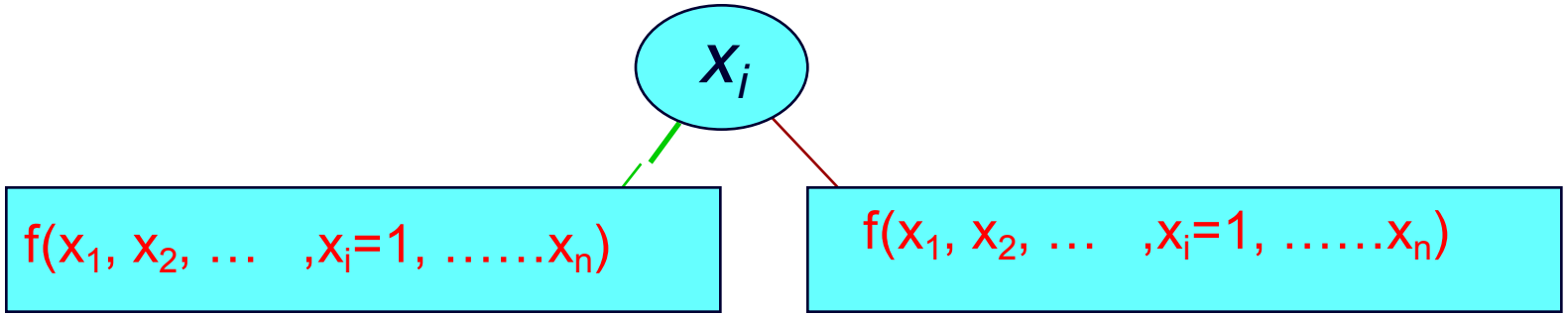
CADSL

Formal Equivalence Checking

- BDD is canonical form of representation
- Shanon's expansion theorem
- $f(x_1, x_2, \ldots x_i, \ldots\ldots x_n) =$

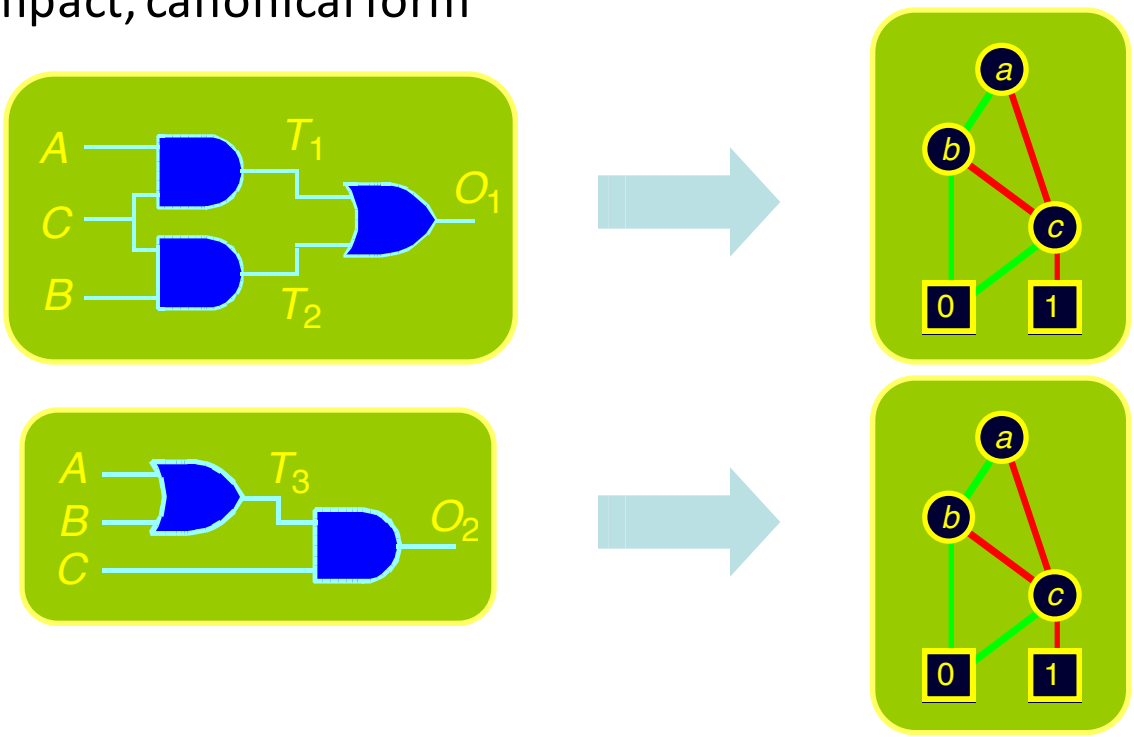
$x_i.f(x_1, x_2, \ldots \ ,x_i=1, \ldots\ldots x_n) +$

$x_i'.\ f(x_1, x_2, \ldots \ ,x_i=0, \ldots\ldots x_n)$



Binary Decision Diagram

- Generate Complete Representation of Circuit Function
  - Compact, canonical form



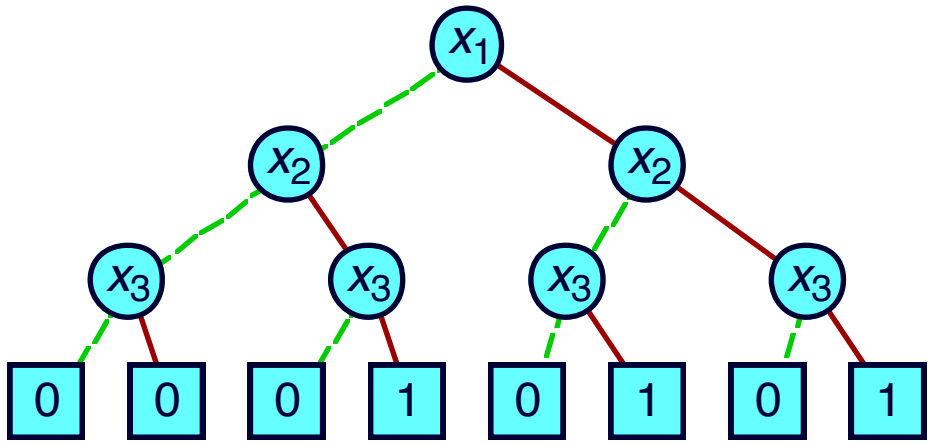
- Functions equal if and only if representations identical
- Never enumerate explicit function values
- Exploit structure & regularity of circuit functions

Decision Structures

Truth Table

x_1	x_2	x_3	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

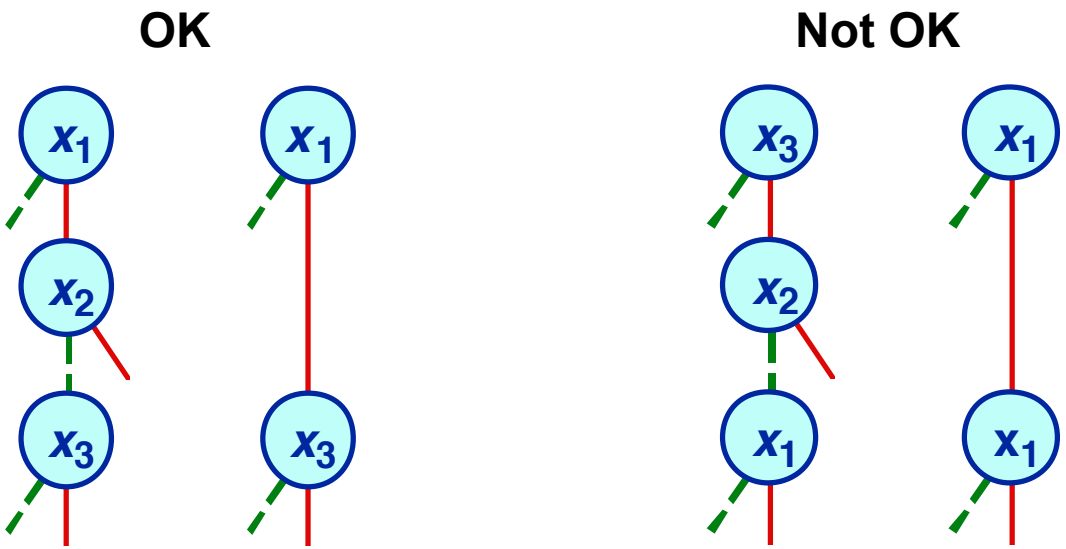
Decision Tree



- Vertex represents decision
- Follow green (dashed) line for value 0
- Follow red (solid) line for value 1
- Function value determined by leaf value.

Variable Ordering

- Assign arbitrary total ordering to variables
  - e.g., $x_1 < x_2 < x_3$
- Variables must appear in ascending order along all paths

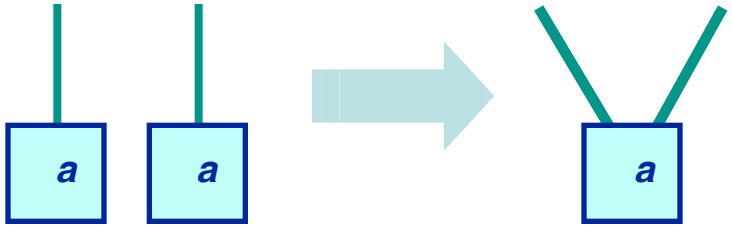


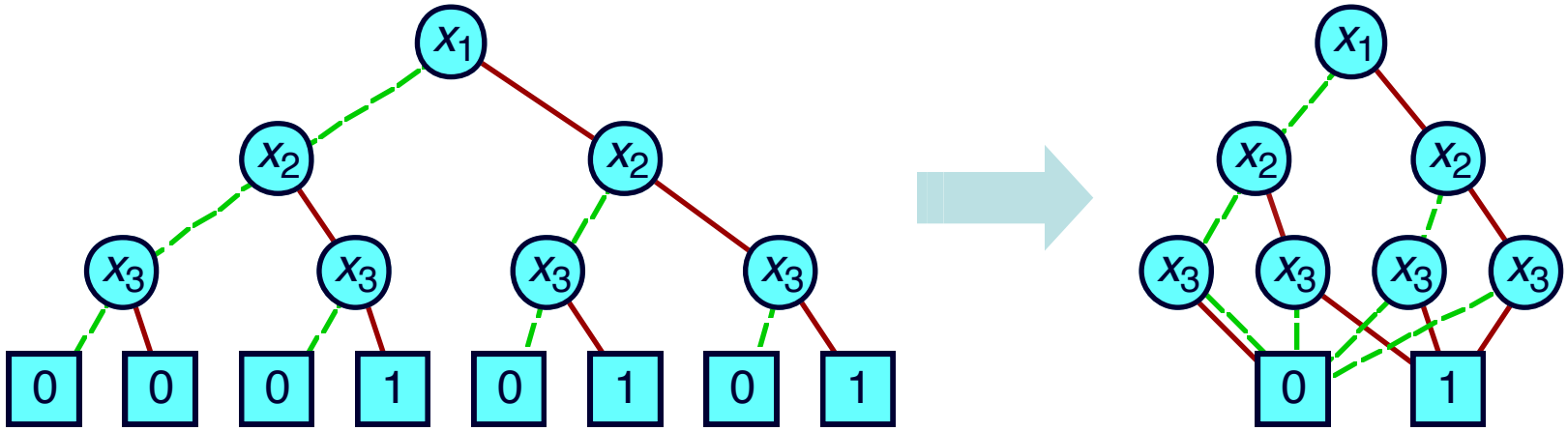
Properties

- No conflicting variable assignments along path
- Simplifies manipulation

Reduction Rule #1

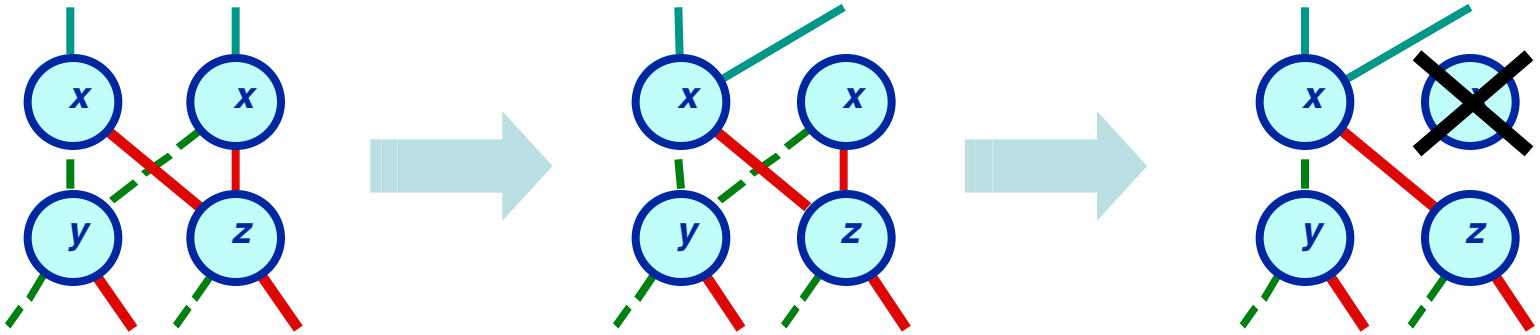
Merge equivalent leaves

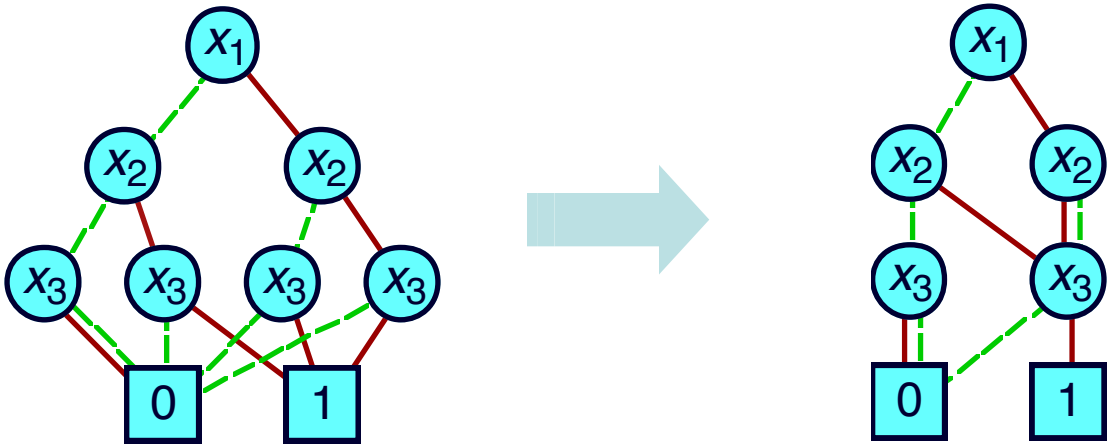




Reduction Rule #2

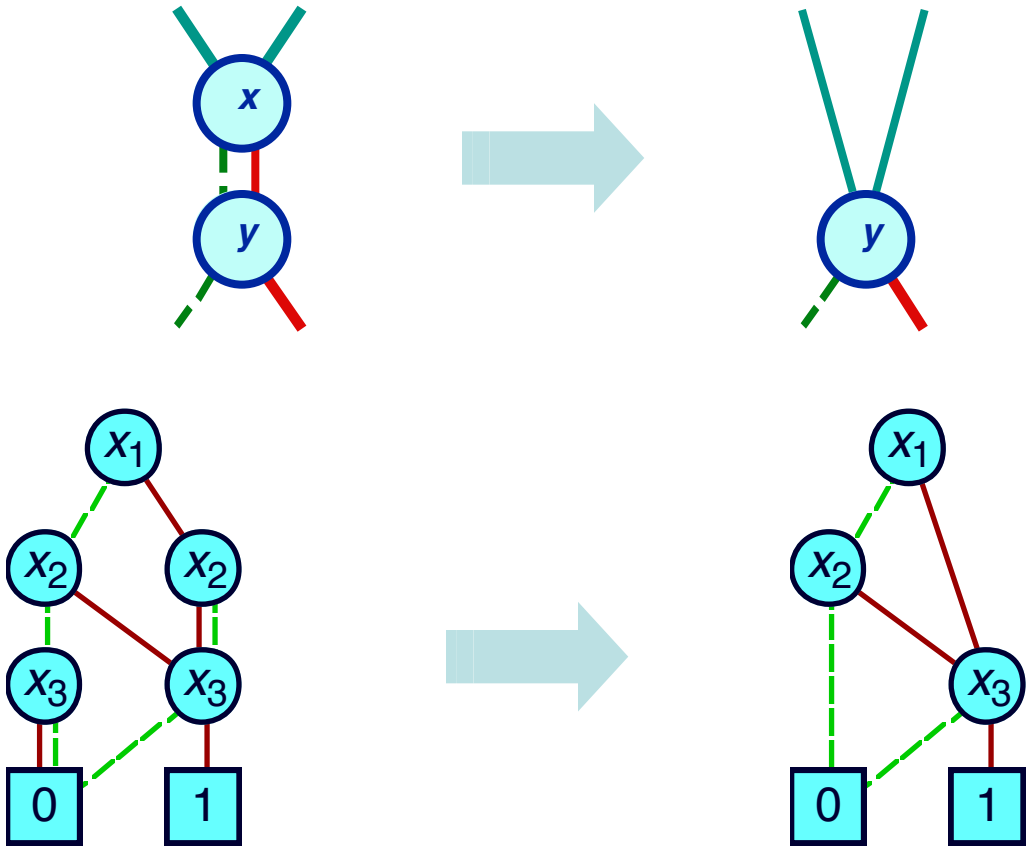
Merge isomorphic nodes



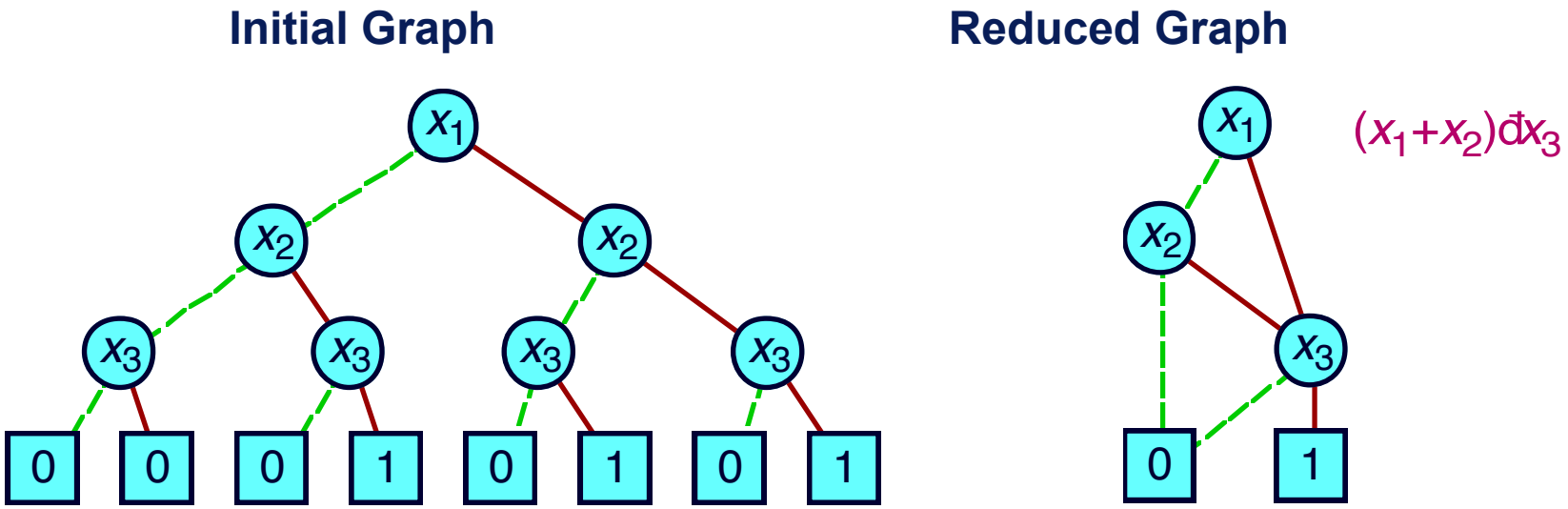


Reduction Rule #3

Eliminate Redundant Tests



Example OBDD



- Canonical representation of Boolean function
  - ❖ For given variable ordering
  - ➢ Two functions equivalent if and only if graphs isomorphic
    - o Can be tested in linear time
  - ➢ Desirable property: *simplest form is canonical*.

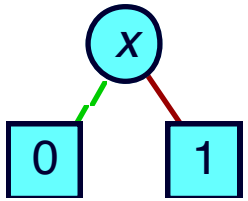
Example Functions

Constants

Unique unsatisfiable function

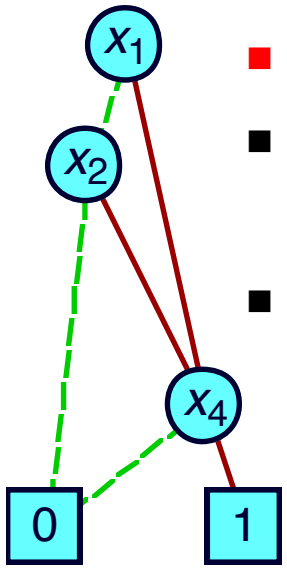
Unique tautology

Variable



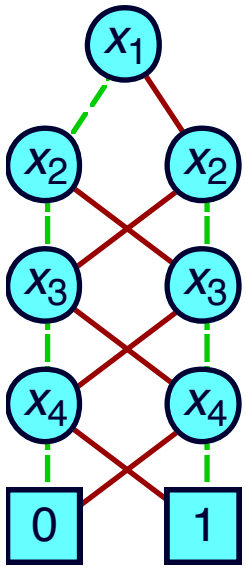
Treat variable as function

Typical Function



- $(x_1 + x_2) . x_4$
- **No vertex labeled x_3**
  - independent of x_3
- **Many subgraphs shared**

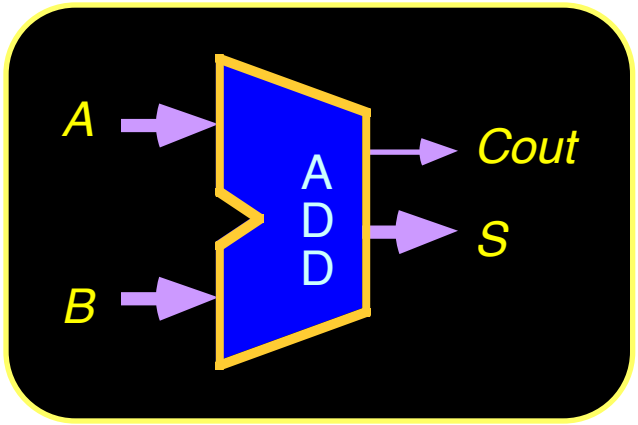
Odd Parity



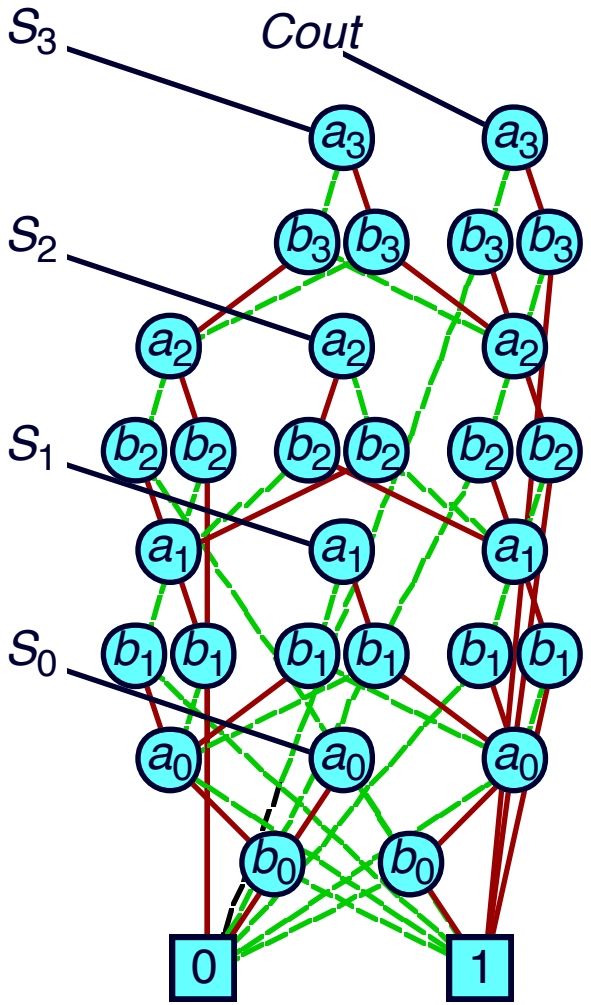
Linear representation

Representing Circuit Functions

- Functions
  - All outputs of 4-bit adder
  - Functions of data inputs



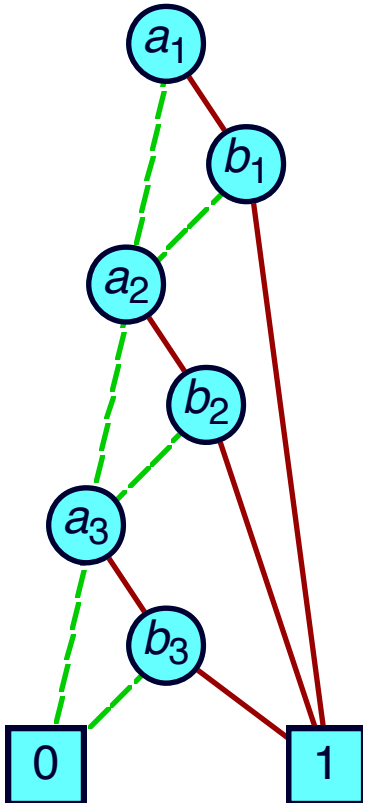
- Shared Representation
  - Graph with multiple roots
  - 31 nodes for 4-bit adder
  - 571 nodes for 64-bit adder
  - ☛ *Linear growth*



Effect of Variable Ordering

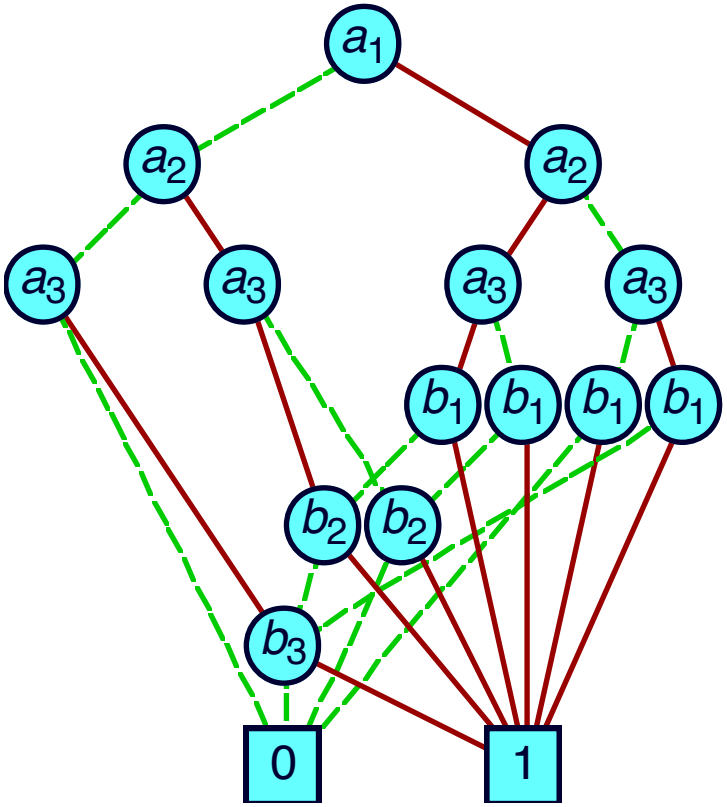
$$(a_1 \wedge b_1) \vee (a_2 \wedge b_2) \vee (a_3 \wedge b_3)$$

Good Ordering



Linear Growth

Bad Ordering



Exponential Growth

Selecting Good Variable Ordering

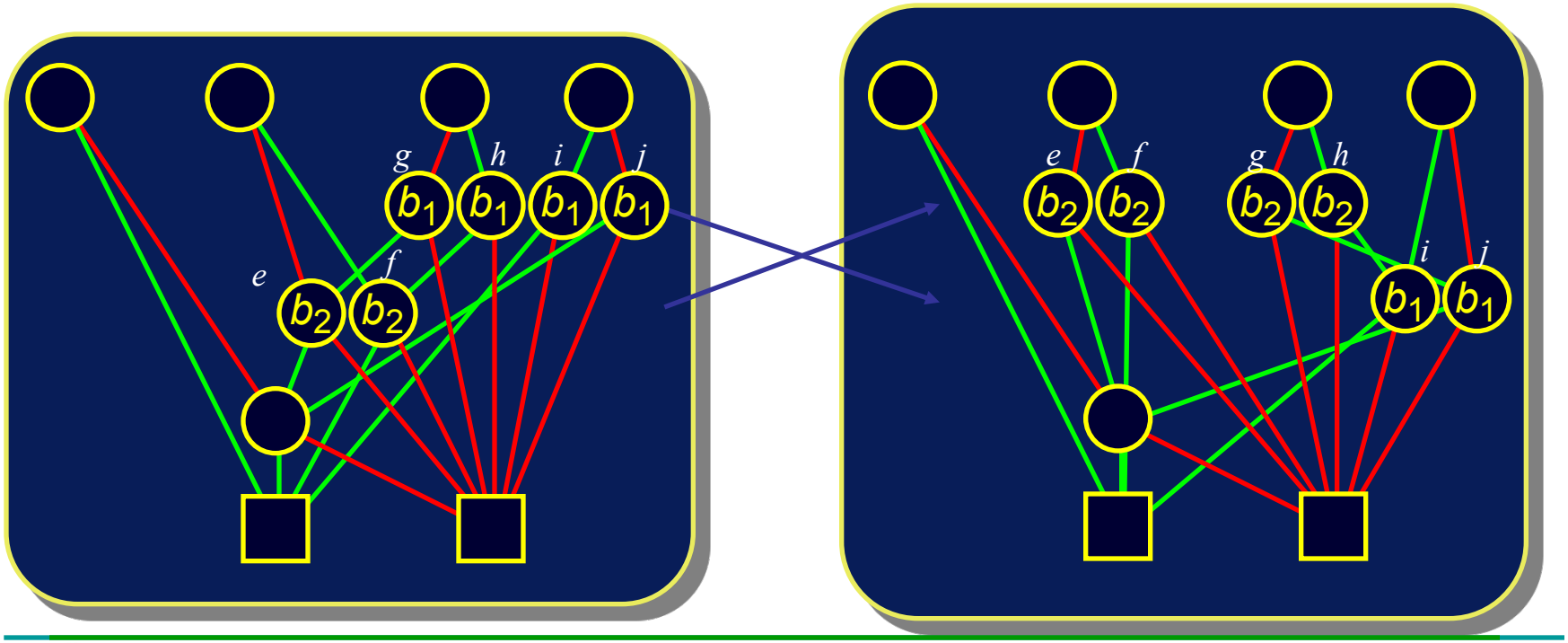
- Intractable Problem
  - Even when problem represented as OBDD
    - i.e., to find optimum improvement to current ordering
- Application-Based Heuristics
  - Exploit characteristics of application
  - e.g., Ordering for functions of combinational circuit
    - Traverse circuit graph depth-first from outputs to inputs
    - Assign variables to primary inputs in order encountered

Selecting Good Variable Ordering

- Static Ordering
  - Fan In Heuristic
  - Weight Heuristic
- Dynamic Ordering
  - Variable Swap
  - Window Permutation
  - Sifting

Swapping Adjacent Variables

- Localized Effect
  - Add / delete / alter only nodes labeled by swapping variables
  - Do not change any incoming pointers

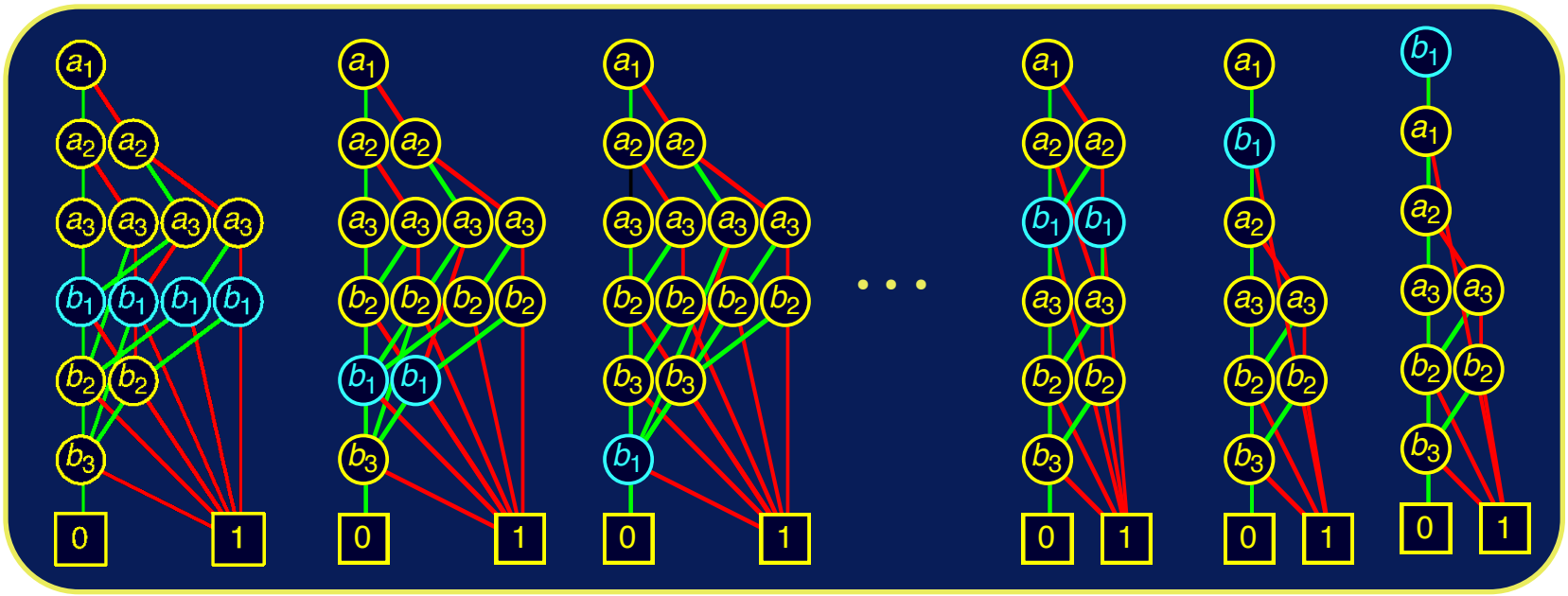


Dynamic Variable Reordering

- Richard Rudell, Synopsys
- Periodically Attempt to Improve Ordering for All BDDs
  - ❖ Part of garbage collection
  - ❖ Move each variable through ordering to find its best location
- Has Proved Very Successful
  - ❖ Time consuming but effective
  - ❖ Especially for sequential circuit analysis

Dynamic Reordering By Sifting

- Choose candidate variable
- Try all positions in variable ordering
  - Repeatedly swap with adjacent variable
- Move to best position found

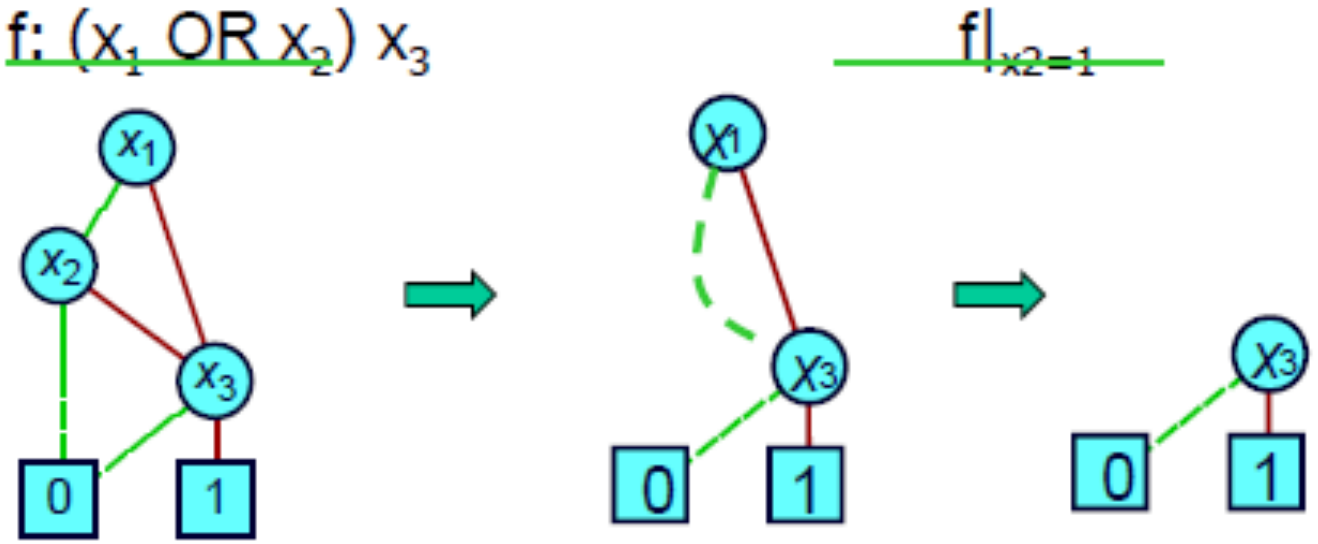


ROBDD Sizes & Variable Ordering

- Bad News
  - Finding optimal variable ordering NP-Hard
  - Some functions have exponential BDD size for all orders *e.g.* multiplier
- Good News
  - Many functions/tasks have reasonable size ROBDDs
  - Algorithms remain practical up to 500,000 node OBDDs
  - Heuristic ordering methods generally satisfactory
- What works in Practice
  - Application-specific heuristics e.g. DFS-based ordering for combinational circuits
  - Dynamic ordering based on variable sifting *(R. Rudell)*

Operations with BDD (1/5)

- Restriction: A restriction to a function to x=d, denoted $f|_{x=d}$, where $x \in var(f)$, and $d \in \{0,1\}$, is equal to f after assigning x = d.
- Given BDD of f, deriving BDD of $f|_{x=d}$ is simple



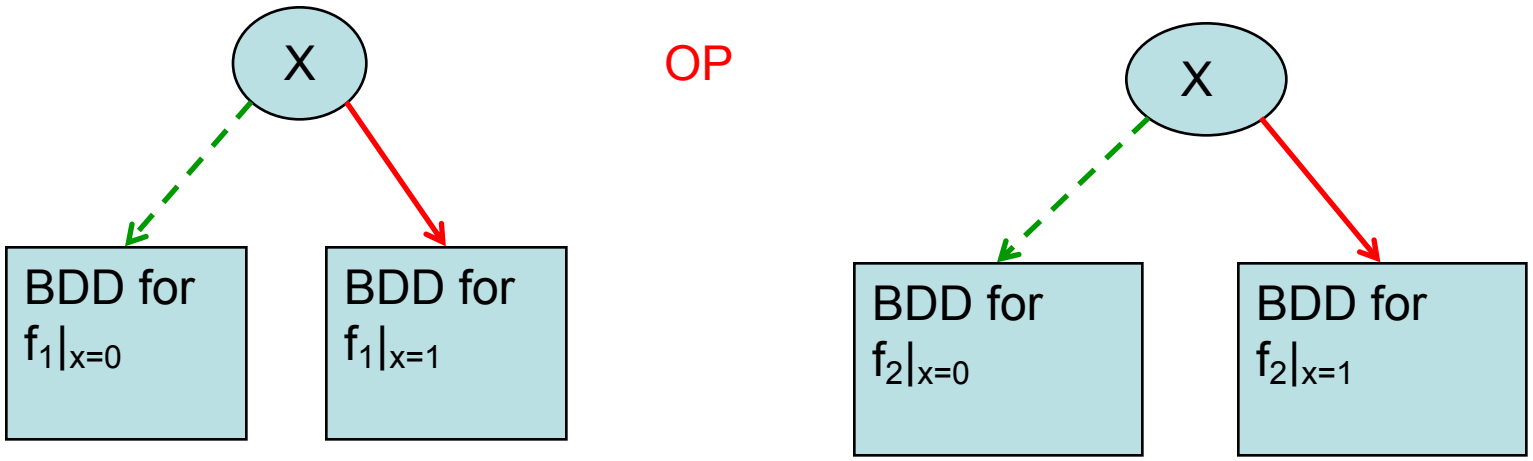
Operations with BDD (2/5)

- Let v_1, v_2 denote root nodes of f_1, f_2 respectively , with $var(v_1) = x_1$ and $var(v_2) = x_2$
- If v_1 and v_2 are leafs, f_1 OP f_2 is a leaf node with value $val(v_1)$ OP $val(v_2)$

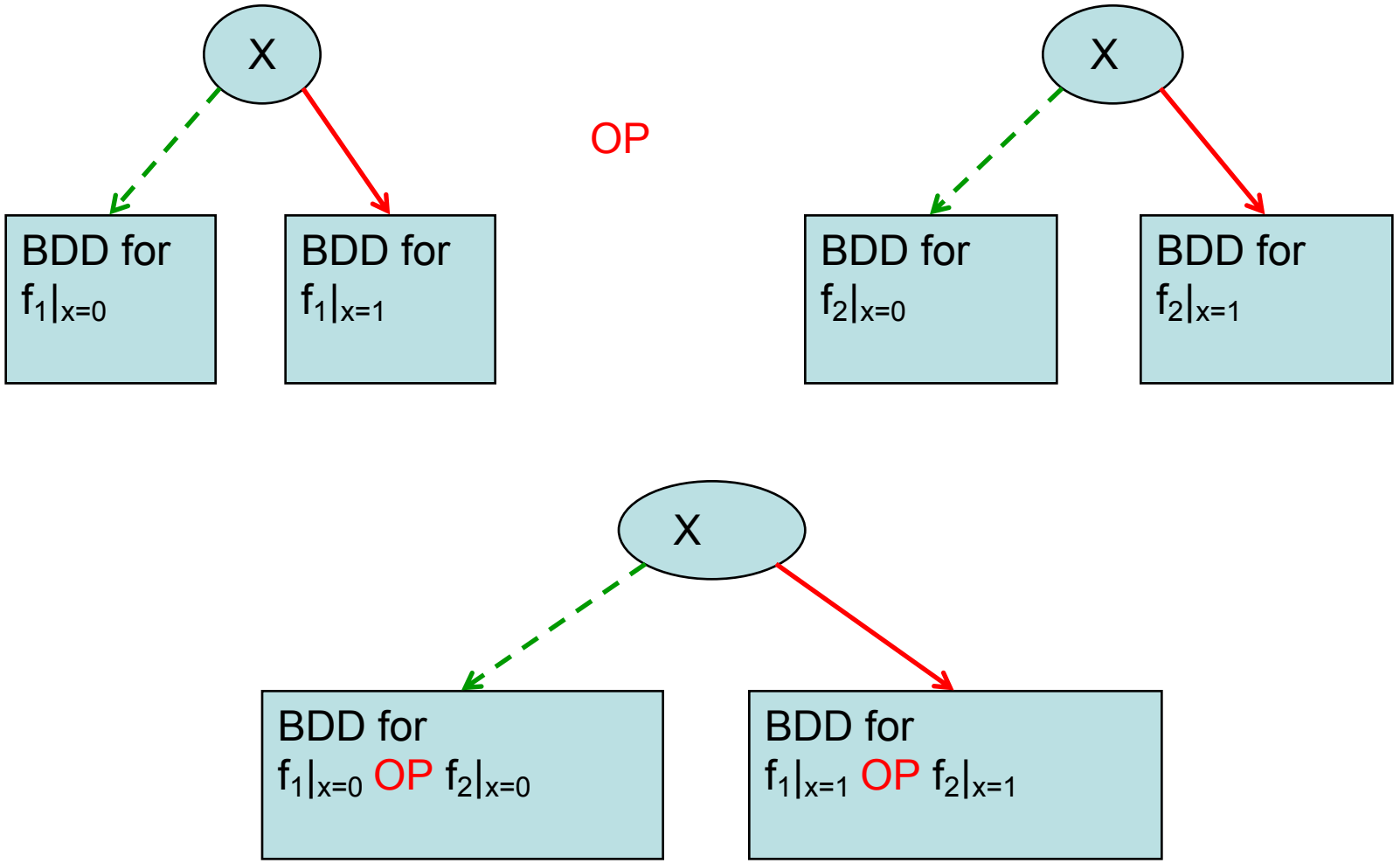
Operations with BDD (3/5)

- If $x_1 = x_2 = x$, apply shanon's expansion

$$f_1 \text{ OP } f_2 = x' \,.\, (f_1|_{x=0} \text{ OP } f_2|_{x=0}) + x \,.\, (f_1|_{x=1} \text{ OP } f_2|_{x=1})$$



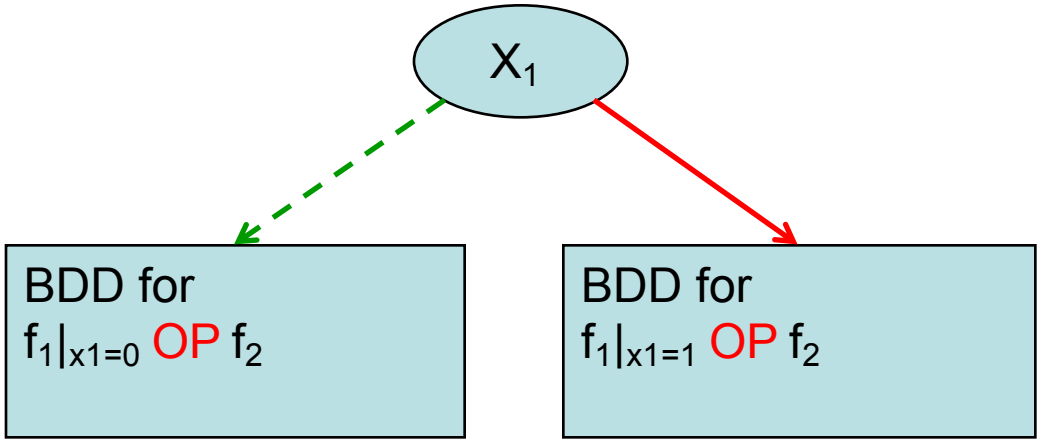
Operations with BDD (4/5)



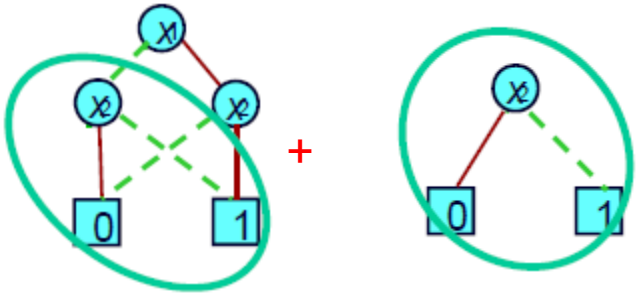
Operations with BDD (5/5)

- Else suppose $x_1 < x_2 = x$, in variable order

f_1 OP $f_2 = x'_1 (f_1|_{x1=0}$ OP $f_2) + x_1 (f_1|_{x1=1}$ OP $f_2)$

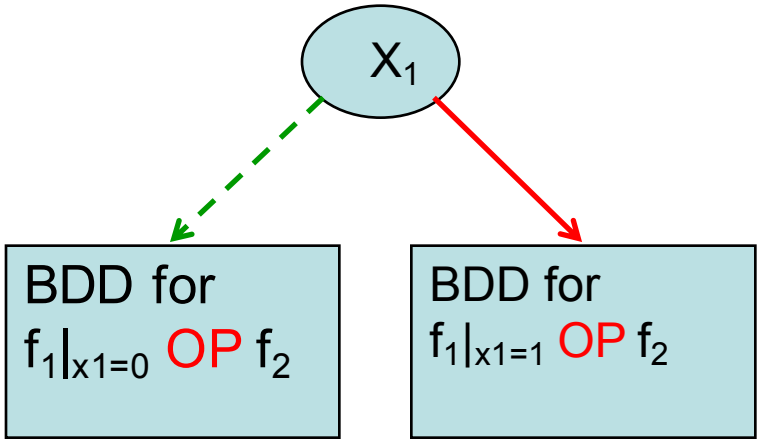


Operations with BDD: Example

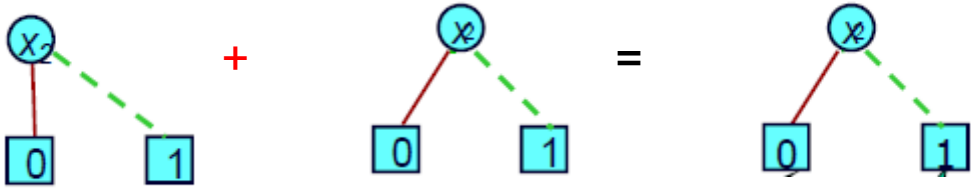


$f_1 = X_1 \text{ XNOR } X_2$

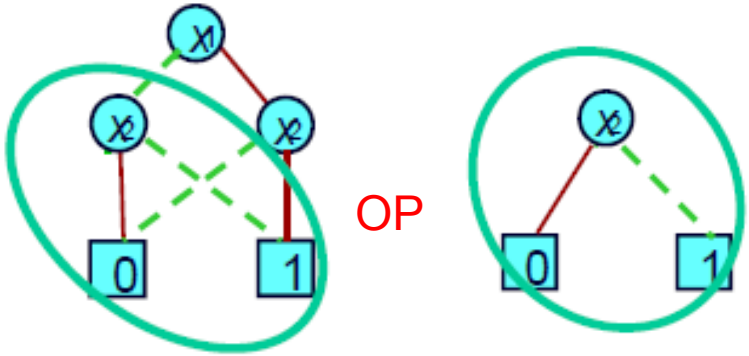
$f_2 = X'_2$



BDD for $f_1|_{x1=0}$ OP f_2 =

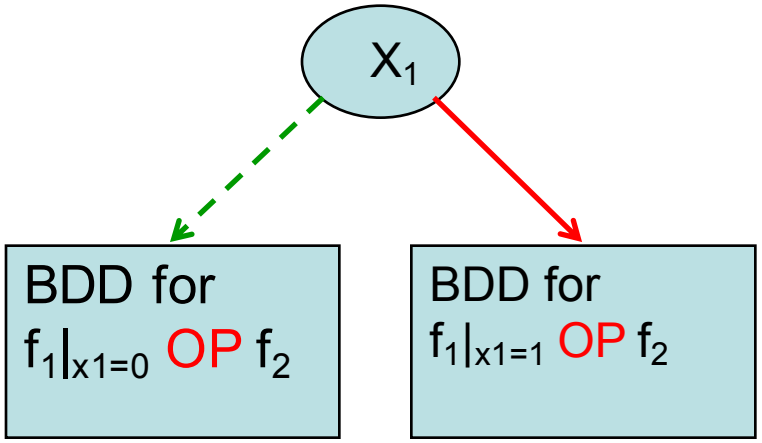


Operations with BDD: Example



OP

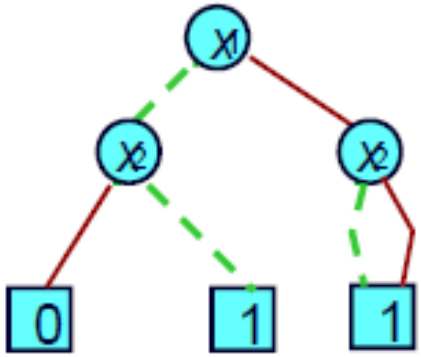
=



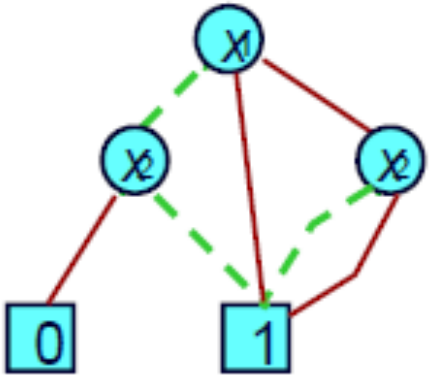
$f_1 = X_1$ XNOR X_2

$f_2 = X'_2$

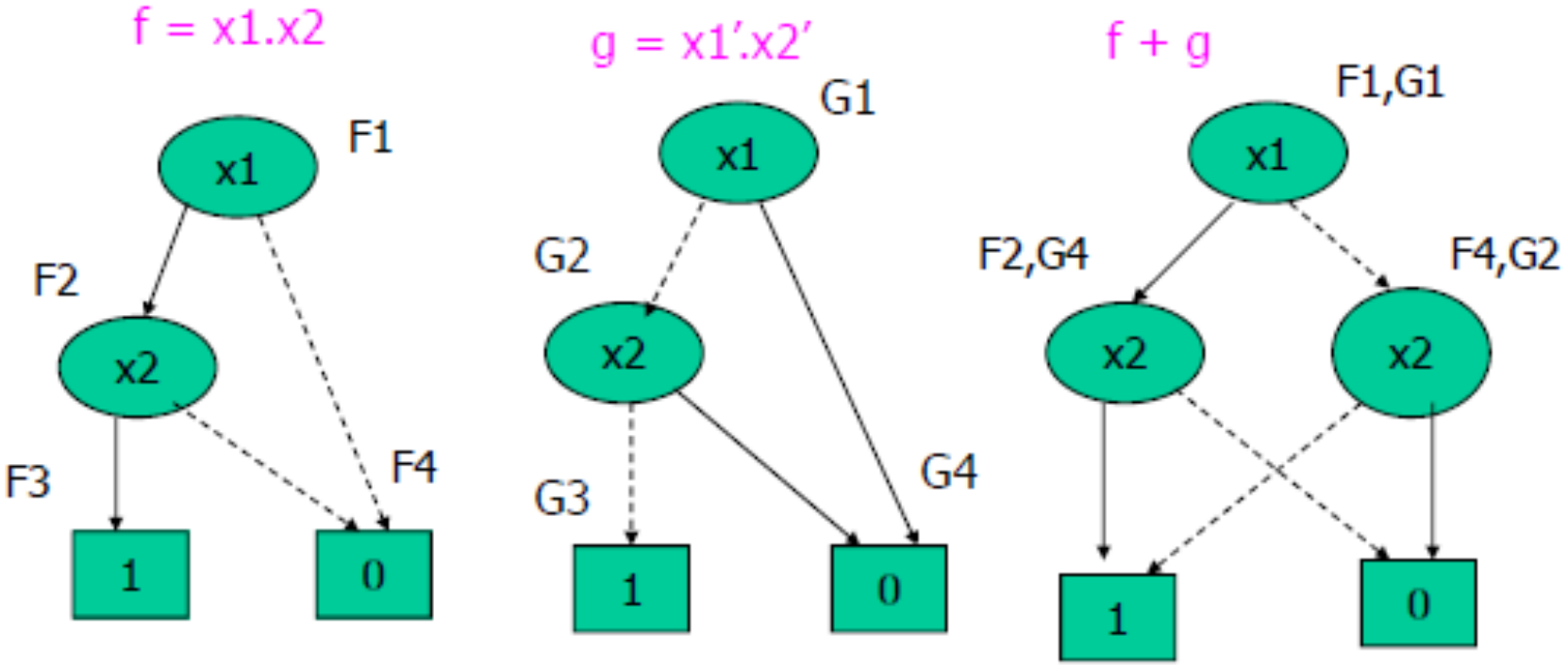
=



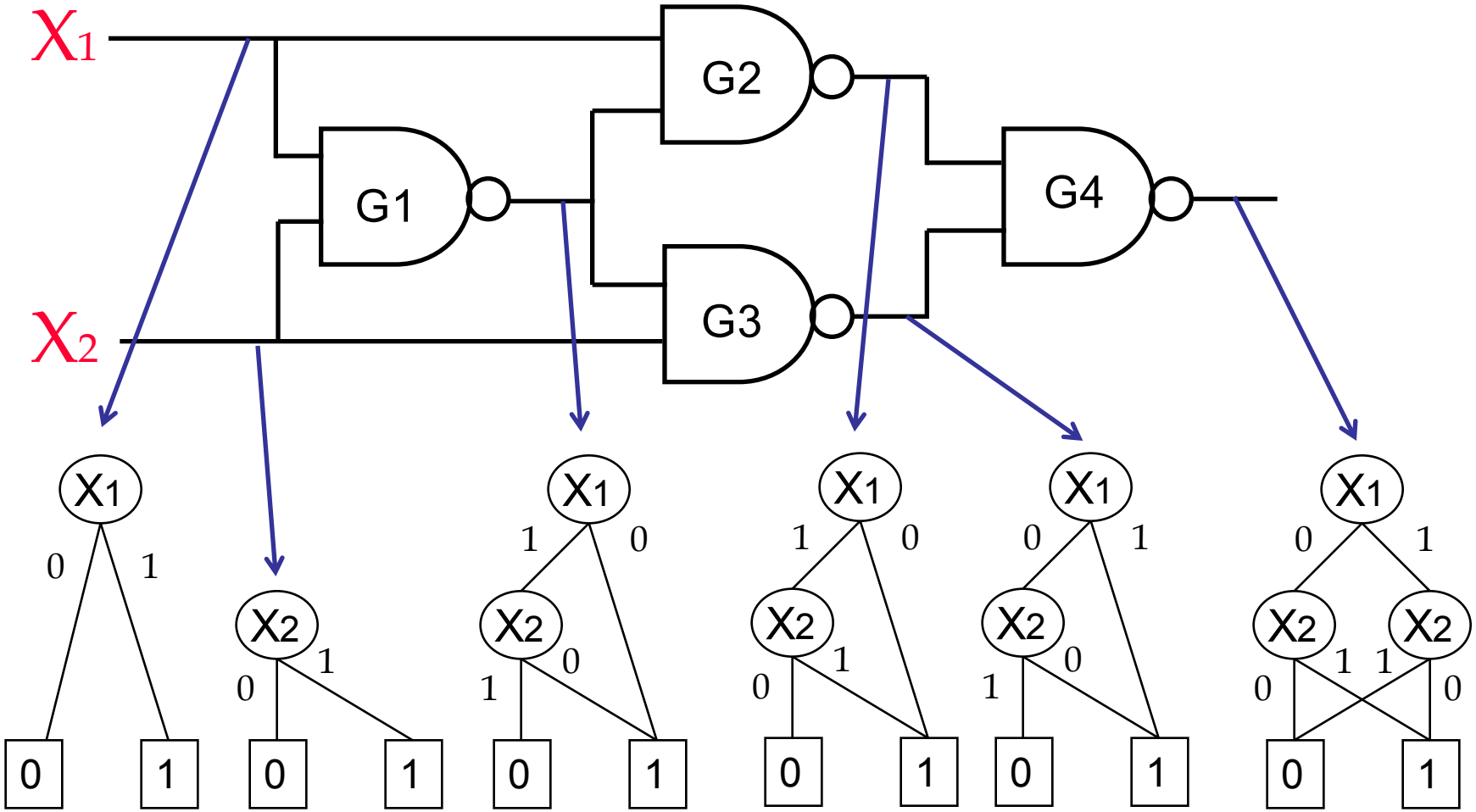
=



Operations with BDD: Example



From Circuits to BDD



Thank You