And Invert Graph (AIG)

Virendra Singh

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

CS-226: Digital Logic Design

Lecture 10-B: 08 February 2021

CADSL

Logic Representation

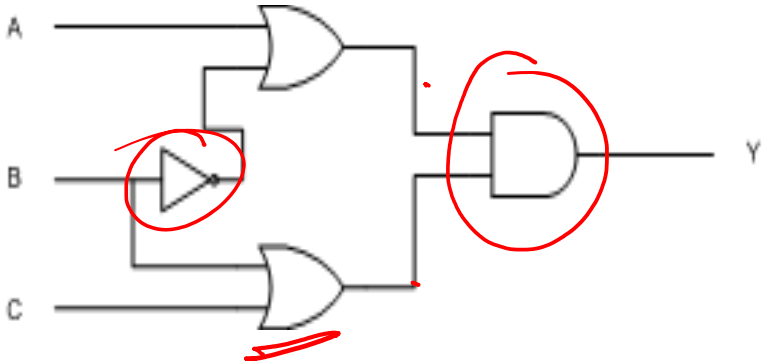
- Canonical Forms in common usage:
  - Truth Table
  - Sum of Minterms (SOM)
  - Product of Maxterms (POM)
  - Binary Decision Diagram (BDD)
  - Reed Muller Representation → Testable circuit, Quantum circuits
- Non canonical Representation
  - Sum of Product
  - Product of Sum
  - And Invert Graph - Graphical - Scalable

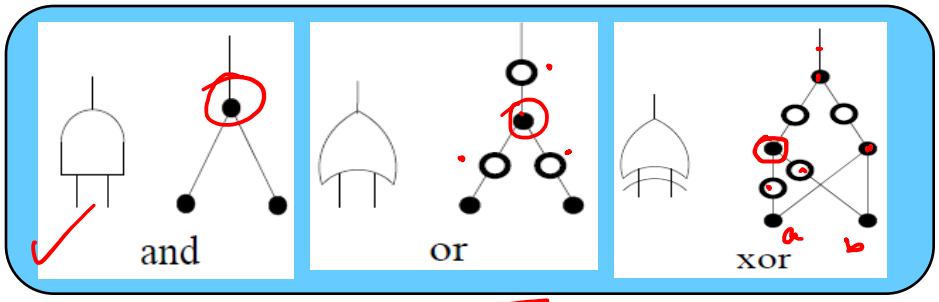
And-Invert Graph (AIG)

- AIG is a Boolean network with two types of nodes:
  - two-input ANDs, nodes
  - Inverters (NOT)
- Any Boolean function can be expressed using AIGs
  - For many practical functions AIGs are smaller than BDDs
  - Efficient graph representation (structural)
  - Very good correlation with design size
- AIGs are not canonical
  - For one function, there may be many structurally-different AIGs
  - Functional reduction and structural hashing can make them "canonical enough"

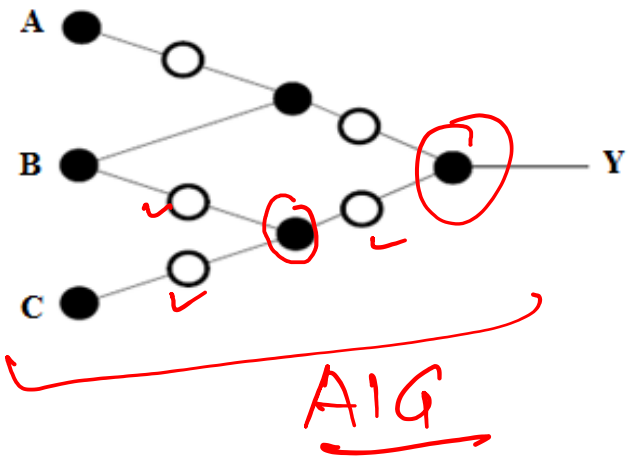
How to Create AIG ?

- AIGs are constructed from the Boolean network
- Constructed from the netlist available from technology independent logic synthesis

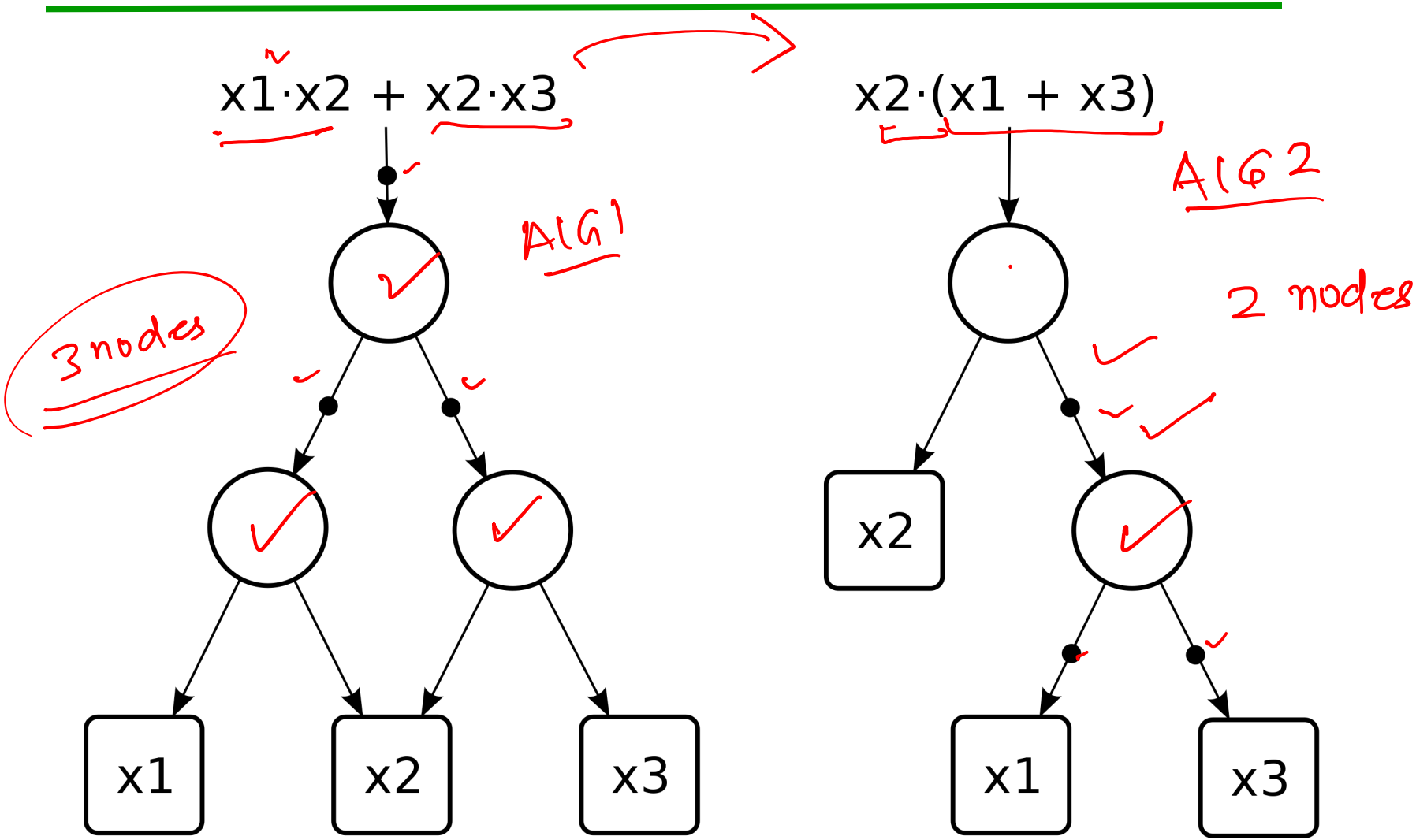




$a+b = \overline{\bar{a}\cdot\bar{b}}$ $\qquad \bar{a}\cdot\bar{b} + a\cdot b$

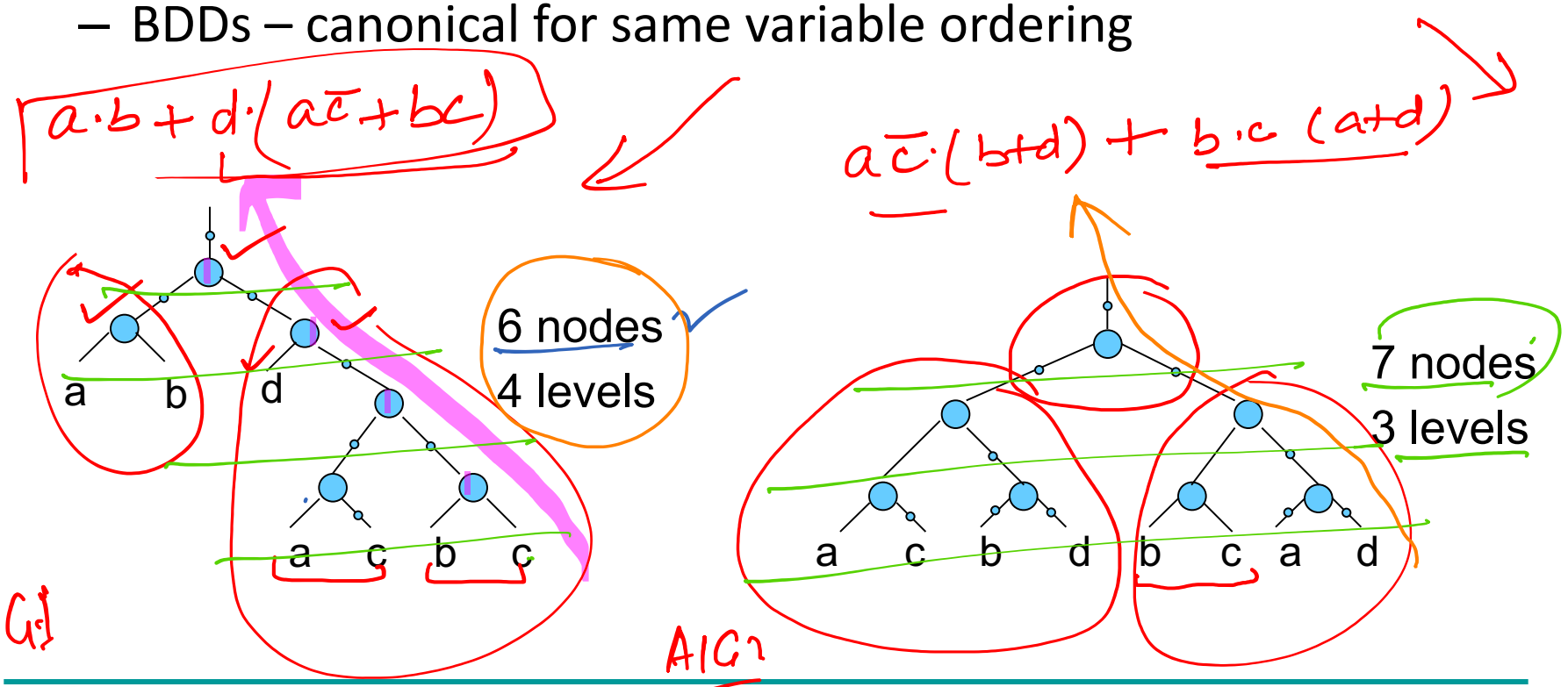


AIG Non-canonicity



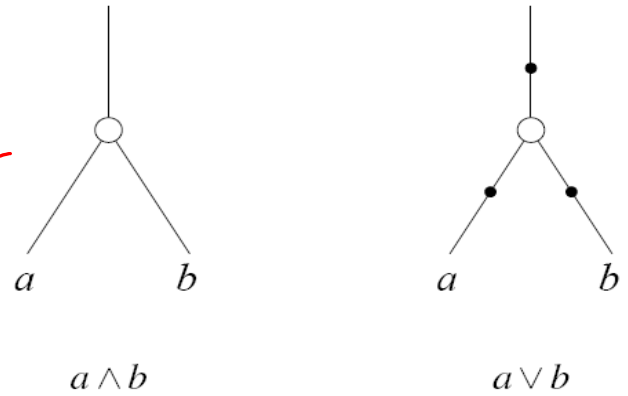
AIG Non-canonicity

- AIGs are **not** canonical
  - same function represented by two functionally equivalent AIGs with different structures
  - BDDs – canonical for same variable ordering



Basic Logic Operations

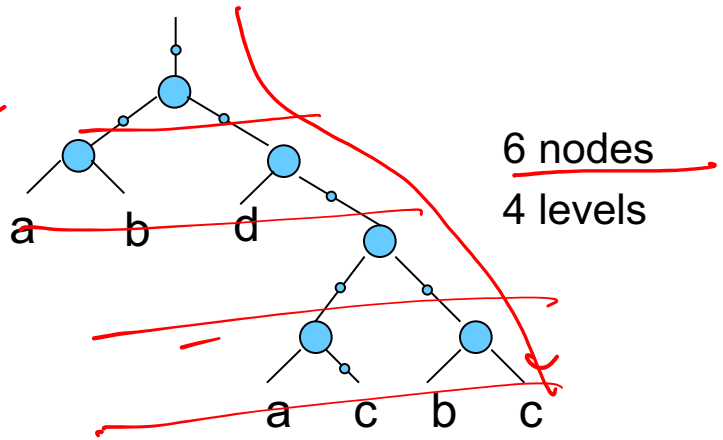
- Converting logic function into AIG graph
  - Inversion $\neg a$ $\neg a$
  - Conjunction $a \wedge b$ (ab) $a \wedge b$
  - Disjunction $a \vee b$ ($a+b$) $\neg(\neg a \wedge \neg b)$
  - Implication $a \Rightarrow b$ $\neg(a \wedge \neg b)$
  - Equivalence $a \Leftrightarrow b$ $\neg(a \wedge \neg b) \wedge \neg(\neg a \wedge b)$
  - a XOR b $\neg(\neg(a \wedge \neg b) \wedge \neg(\neg a \wedge b))$.



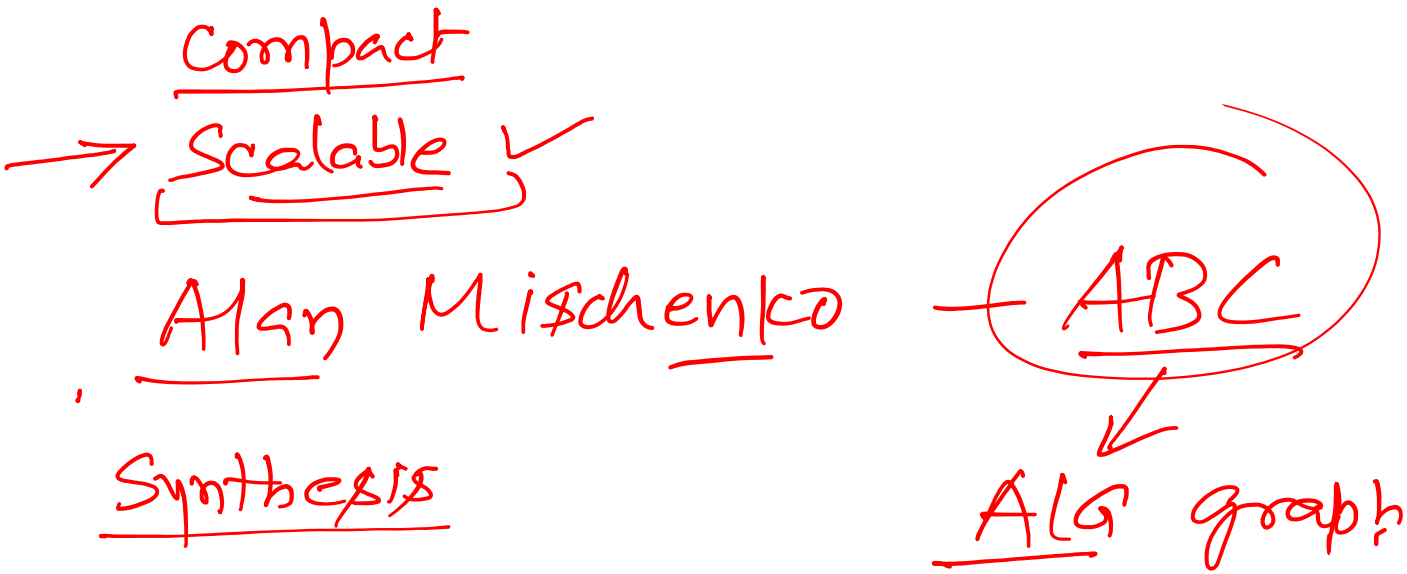
$a \wedge b$ $a \vee b$

AIG Attributes

- AIG size
  - Measured by number of AND nodes
- AIG depth
  - Number of logic levels = number of AND-gates on longest path from a primary input to a primary output
  - The inverters are ignored when counting nodes and logic levels



And Invert Graph (AIG)



Thank You