

# ECE 510 OCE BDDs and Their Applications

## Lecture 3. Manipulation Algorithms

April 4, 2000

Alan Mishchenko

### Overview

- ITE operator
- APPLY operator
- RESTRICT operator
- Derived operators
- The effect of variable ordering
- Deriving the upper bound on BDD size for boolean functions
- Dynamic variable reordering

## IF-THEN-ELSE (ITE) Operator

- Boolean operations over 2 arguments can be expressed as ITE of F, G, and constants

$$\text{ITE}(F, G, H) = F \& G + F' \& H$$

- Example:  $\text{AND}(F, G) = \text{ITE}(F, G, 0)$
- Computation of boolean operations is based on the Shannon expansion

$$\text{ITE}(F, G, H) = \text{ITE}(x, \text{ITE}(F_{x'}, G_{x'}, H_{x'}), \text{ITE}(F_x, G_x, H_x))$$

## APPLY operator

- **APPLY**(F, G) operator is a shorthand for any two-variable boolean operator
- **APPLY** is reducible to **ITE**
- It follows that **APPLY** can be computed recursively just like **ITE**

$$\text{APPLY}(F, G) = x' \& \text{APPLY}(F_{x'}, G_{x'}) + x \& \text{APPLY}(F_x, G_x)$$

## Pseudocode for APPLY operator

```

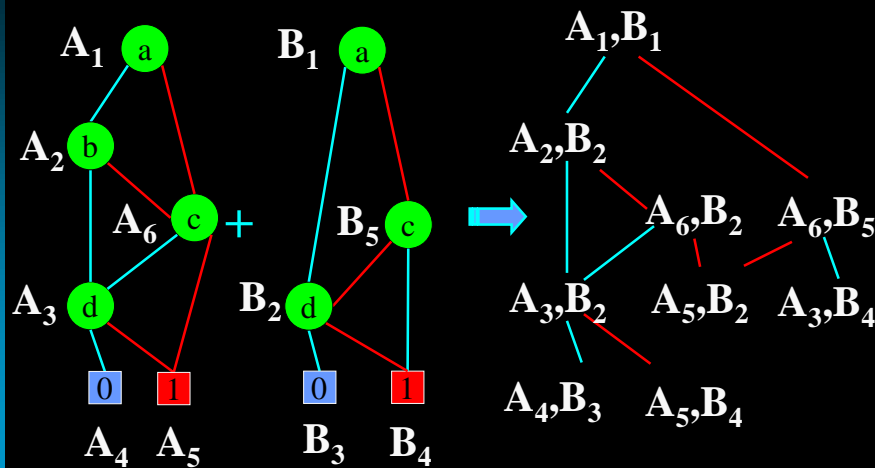
function Apply( F, G )
  if ( AlreadyComputed( F, G ) ) return the result;
  else if ( F=={0,1} && G=={0,1} ) return oper( F, G );
  else if ( Var( F ) == Var( G ) )
    u = CreateNode( Var(F), Apply(Fx',Gx'), Apply(Fx,Gx));
  else if ( Var( F ) < Var( G ) )
    u = CreateNode( Var(F) , Apply(Fx',G) , Apply(Fx,G));
  else /* if ( Var( F ) > Var( G ) ) */
    u = CreateNode( Var(F) , Apply(F,Gx' ) , Apply(F,Gx));
  InsertComputed( F,G,u );
  return u;
  
```

April 4, 2000

ECE 510 OCE: BDDs and Their Applications

5

$$F=ac+bc+d \quad G=ac'+d \quad F+G = ?$$

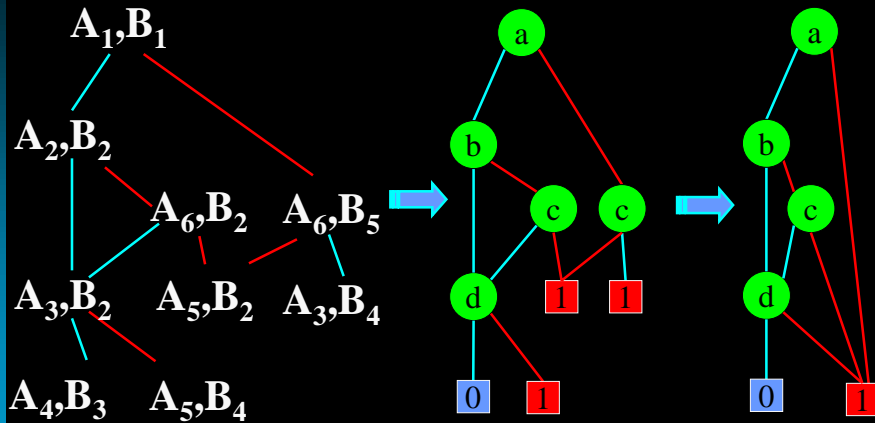


April 4, 2000

ECE 510 OCE: BDDs and Their Applications

6

$$F = ac + bc + d \quad G = ac' + d \quad F + G = a + bc + d$$



April 4, 2000

ECE 510 OCE: BDDs and Their Applications

7

## Pseudocode for RESTRICT operator

```

function Restrict( F, var, value )
if ( AlreadyComputed( F, var, value ) ) return result;
else if ( Var( F ) > var ) return F;
else if ( Var( F ) < var )
    u = CreateNode( Var(F), Restrict(Fx', var, value),
                  Restrict(Fx, var, value) );
    InsertComputed(F, var, value, u); return u;
else /* ( Var( F ) == var */ if ( value == 0 )
    return = Restrict(Fx', var, value);
else /* ( Var( F ) == var && value == 1 ) */
    return = Restrict(Fx, var, value);

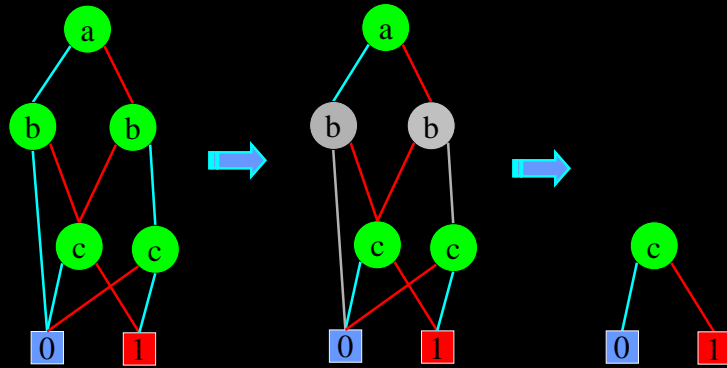
```

April 4, 2000

ECE 510 OCE: BDDs and Their Applications

8

$$F = bc + ab'c' \quad F(b=1) = ?$$



April 4, 2000

ECE 510 OCE: BDDs and Their Applications

9

## Derived Operations: COMPOSE

- Given  $F(x)$  and  $G(y)$ , find  $F(G(y))$
- Using Shannon Expansion
 
$$F(x) = x' \& Fx' + x \& Fx$$

$$F(G(y)) = G'(y) \& Fx' + G(y) \& Fx$$
- **COMPOSE** is reduced to two operations **RESTRICT** and three operations **APPLY**

April 4, 2000

ECE 510 OCE: BDDs and Their Applications

10

## Derived Operations: Quantification

- Given a function  $F(x_1, x_2, x_3)$
- **Existential quantification** of  $F$  w.r.t.  $x_1$  is
$$\exists_{x_1} F(x_1, x_2, x_3) = F(0, x_2, x_3) + F(1, x_2, x_3)$$
- **Universal quantification** of  $F$  w.r.t.  $x_1$  is
$$\forall_{x_1} F(x_1, x_2, x_3) = F(0, x_2, x_3) \& F(1, x_2, x_3)$$
- **Unique quantification** of  $F$  w.r.t.  $x_1$  is
$$\exists!_{x_1} F(x_1, x_2, x_3) = F(0, x_2, x_3) \oplus F(1, x_2, x_3)$$

## Summary of Operations on BDDs

- **Apply** – NOT, AND, OR, EXOR, etc.
- **Restrict**
- **Compose (Replace)**
- **Quantification** (existential, universal)
- **Specialized operators**
  - Generalized cofactor (constrain, restrict)
  - Compatible projection, etc.

## Worst-Case Complexity

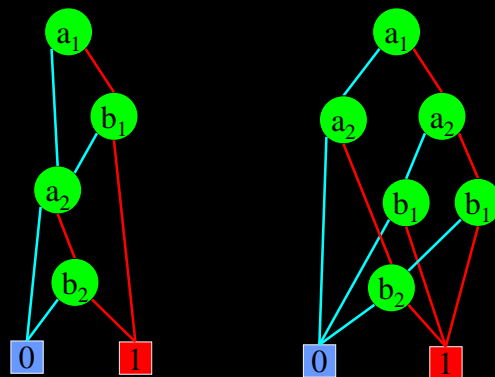
- CreateNode -  $O(1)$
- Build -  $O(2^n)$
- APPLY -  $O(|F| * |G|)$
- RESTRICT -  $O(|F|)$
- COMPOSE -  $O(|F|^2 * |G|^2)$

April 4, 2000

ECE 510 OCE: BDDs and Their Applications

13

## Variable Ordering for $F = a_1b_1 + a_2b_2$



$$a_1 < b_1 < a_2 < b_2$$

$$a_1 < a_2 < b_1 < b_2$$

April 4, 2000

ECE 510 OCE: BDDs and Their Applications

14

## Deriving the Upper Bound on BDD Size for Boolean Functions

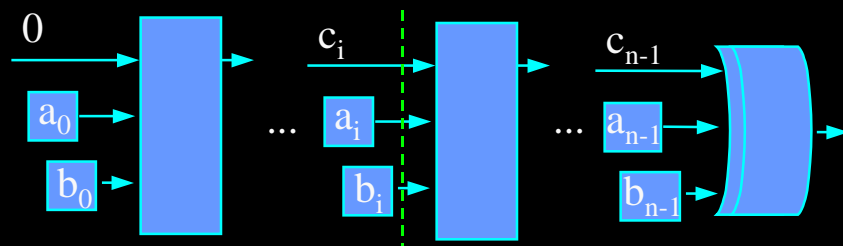
- **Linear arrangement** of the network is a numbering of its blocks from **1** to **n** such that the output block is numbered last
- **Forward cross section of a block** is the number of wires from the outputs of previous blocks to the inputs of this and following blocks
- **Forward cross section of the network ( $wf$ )** is the maximum forward cross over all blocks
- Similarly we introduce **backward cross section ( $wb$ )**
- The size of BDD for the network is  $n * 2^{wf} * 2^{wb}$

April 4, 2000

ECE 510 OCE: BDDs and Their Applications

15

## Example: Circuit for the Most Significant Bit of Integer Adder



Forward cross section  $wf = 3$

Forward cross section  $wb = 0$

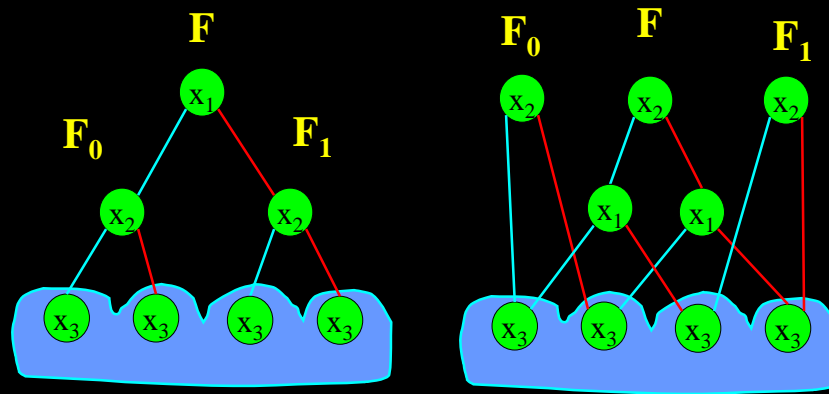
The upper bound is linear  $U < 8n$

April 4, 2000

ECE 510 OCE: BDDs and Their Applications

16

## Dynamic Variable Reordering



Variable reordering is a local operation

April 4, 2000

ECE 510 OCE: BDDs and Their Applications

17

## Properties of BDDs

- Canonicity
- Compactness (with some exceptions)
- Fast computation (with some exceptions)
- Represent a variety of discrete objects
  - Boolean functions and relations
  - Compositional sets and subsets
  - Partitions of states
  - Encodings and labelings
- Facilitate **symbolic** methods
  - Two-level minimization
  - State traversal of FSMs, etc.

April 4, 2000

ECE 510 OCE: BDDs and Their Applications

18