

ECE 510 OCE

BDDs and Their Applications

Lecture 13. (1) Compatible Projection (2) Implicit Formulation for Problems in Boolean Networks

May 9, 2000
Alan Mishchenko

Overview

- Compatible projection operator
 - Background
 - Recursive algorithm
 - Applications (FSM state minimization, FSM decomposition, functional decomposition, classification of functions, etc.)
- Representations of Boolean networks
 - functional
 - structural
- Problems in Boolean networks
 - test pattern generation
 - fault localization
 - optimization (for the number of nodes, area, delay, testability)
 - eliminating combinational loops, etc

Equivalence Relation

- **Equivalence relation** is a boolean function $E(x,y) : \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}$, which is true for codes $x=(x_1x_2\dots x_n)$ and $y=(y_1y_2\dots y_n)$ iff the corresponding encoded objects are equivalent

May 9, 2000

ECE 510 OCE: BDDs and Their Applications

3

Properties of Equivalence Relation

- Equivalence relation is **reflexive**, **symmetric**, and **transitive**
- Suppose the equivalence classes are $\{(00,01),(11),(10)\}$

	00	01	11	10
00	1	1	0	0
01	1	1	0	0
11	0	0	1	0
10	0	0	0	1

May 9, 2000

ECE 510 OCE: BDDs and Their Applications

4

Equivalence Class Characterization Relation

- **Equivalence class characterization relation** $\xi(x,y)$ selects exactly one representative from each equivalence class of states defined by equivalence relation $E(x, y)$
- $\xi(x, y)$ is a boolean function that is one for the codes x and y iff the equivalence class containing code x is represented by the state with code y
 $\xi(x, y) = CProjection(E(x, y), y_0),$
 y_0 is the reset state encoded using variables y .

May 9, 2000

ECE 510 OCE: BDDs and Their Applications

5

Compatible Projection Operator

- Given an equivalence relation $E(x,y)$, the **compatible projection** is a boolean function selecting exactly one representative from each equivalence class and relating it all members of this class
- In particular, the selection function may find the element which has the shortest distance from the given vertex y_0

May 9, 2000

ECE 510 OCE: BDDs and Their Applications

6

Compatible Projection Pseudocode

```

bdd CProjection( bdd E(x,y), bdd  $\alpha(y)$  ) {
  if (  $\alpha = 1$  ) return E;
  if ( E = 0 ) return 0;
  if ( E = 1 ) return  $\alpha$ ;
   $y_i$  is the top variable in  $\alpha$ ;
  if (  $\alpha_{y_i} = 0$  )       $\alpha_i = y_i'$ ;
  else /*if (  $\alpha_{y_i} = 0$  )*/  $\alpha_i = y_i$ ;

   $\gamma = \exists_{y_i} E_{\alpha_i}$ ;
  return  $\alpha_i$  & CProjection(  $E_{\alpha_i}$  ,  $\alpha_{\alpha_i}$  ) +
          $\gamma \alpha_i'$  & CProjection(  $E_{\alpha_i}$  ,  $\alpha_{\alpha_i}$  );
}

```

May 9, 2000

ECE 510 OCE: BDDs and Their Applications

7

Complete Source Code for CProjection

```

bdd CProjection( bdd F, bdd Ref )
{
  assert( Ref != bddfalse );
  if ( Ref == bddtrue ) return F;
  if ( F == bddfalse ) return bddfalse;
  if ( F == bddtrue ) return Ref;

  // check cache for ready-made results

  bdd NextRef, Literal;
  int CurVar = bdd_var( Ref );
  if ( bdd_low( Ref ) == bddfalse )
  { // the top var is positive
    NextRef = bdd_high( Ref );
    Literal = bdd_ithvar( CurVar );
  }
  else if ( bdd_high( Ref ) == bddfalse )
  { // the top var is negative
    NextRef = bdd_low( Ref );
    Literal = !bdd_ithvar( CurVar );
  }
  else // Ref is not a cube!
    assert( 0 );

  // cofactors of F with respect to this literal
  bdd PosCofF = bdd_restrict( F, Literal );
  bdd NegCofF = bdd_restrict( F, !Literal );

  // part of projection that is aligned with Literal
  bdd PosPart = CProjection( PosCofF, NextRef );

  // the domain where the projection
  // aligned with Literal exists
  bdd Domain = bdd_exist( PosCofF, AllYVars );

  // part of projection that is not aligned with Literal
  bdd NegPart = !Domain &
    CProjection( NegCofF, NextRef );

  bdd Result = bdd_ite( Literal, PosPart, NegPart );

  // insert the result into cache

  return Result;
}

```

May 9, 2000

ECE 510 OCE: BDDs and Their Applications

8

Applications of Compatible Projection

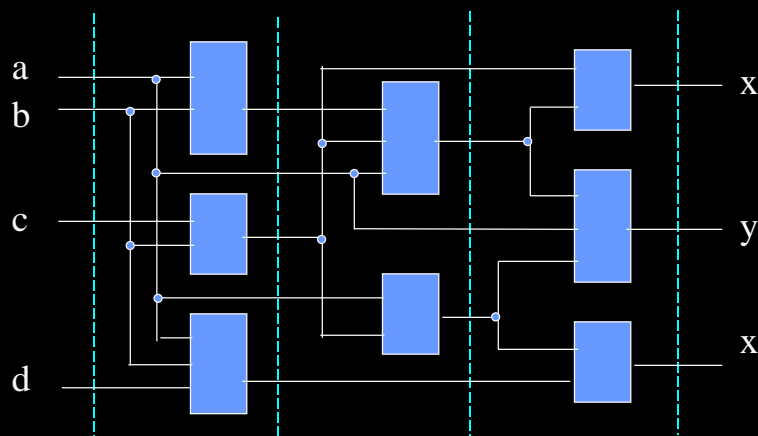
- **FSM state minimization**
 - selecting one state from a group of equivalent states
- **FSM decomposition**
 - counting number of blocks in implicitly represented partitions
 - finding partitions with maximum/minimum number of blocks
- **Functional decomposition**
 - finding exact or approximate column multiplicity
- **Classification of boolean functions**
 - deriving the number of NPN classes of functions depending on the given number of variables
- **Other applications dealing with equivalence classes**

May 9, 2000

ECE 510 OCE: BDDs and Their Applications

9

Boolean (Binary) Networks (Netlists)



May 9, 2000

ECE 510 OCE: BDDs and Their Applications

10

Functional/Structural Representations

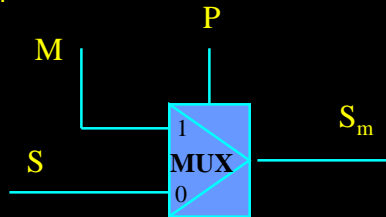
- **Functional representation** characterizes the output behavior in response to inputs (possibly in presence of faults)
- **Structural representation** characterizes the network as interconnection of blocks belonging to one of the finite number of types

A BDD-based approach to ATPG

- Deriving functional representation (sensitivity function)
 - Introducing modifiers
 - Encoding test points and fault models
- Solving testing problems using sensitivity function
 - Accounting for specific fault models
 - **Stuck-at-k** and **stuck-at-any** fault models
 - Other “combinational” fault models (**shifting**, **bridging**)
 - “Sequential” fault models (**delay**, **stuck-open**)
 - ATPG for multiple faults
 - Fault localization
 - Towards sequential ATPG

Introducing Modifiers

- A **modifier** is a multiplexer controlled by additional variable **P**



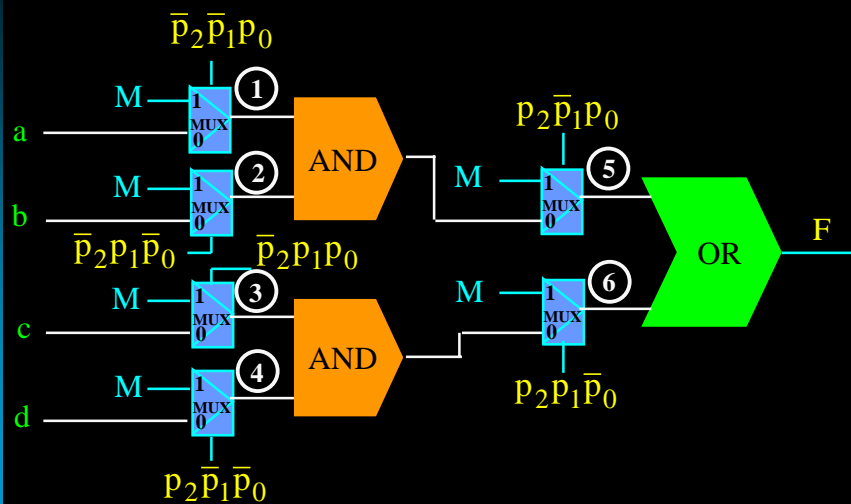
- Modifiers are *not* introduced in hardware. It is a way to formally describe selective change of signal **S** in the circuit by assuming that, under certain condition **P**, it is replaced by some other signal **M**

May 9, 2000

ECE 510 OCE: BDDs and Their Applications

13

Example: 2-AND-1-OR circuit



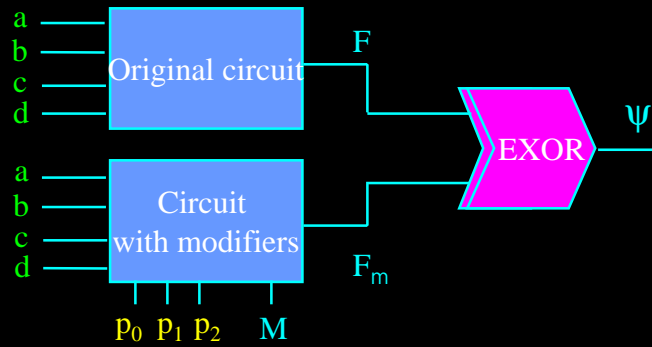
May 9, 2000

ECE 510 OCE: BDDs and Their Applications

14

Generalized Sensitivity Function

$$\psi(a,b,c,d,p_0,p_1,p_2,M) = F \oplus F_m$$



May 9, 2000

ECE 510 OCE: BDDs and Their Applications

15

$\psi(a,b,c,d,p_0,p_1,p_2,M)$ for $M = 1$

a	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
b	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0
c	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0
d	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0
p2	000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
p1	001	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0
p0	011	0	1	0	0	0	0	0	1	0	0	0	0	0	0	1
	010	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
	110	1	1	0	1	1	0	1	1	0	0	0	0	1	0	1
	111	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	101	1	1	0	1	1	0	1	1	0	0	0	0	1	0	1
	100	0	0	0	1	1	0	0	0	0	0	0	0	1	0	0

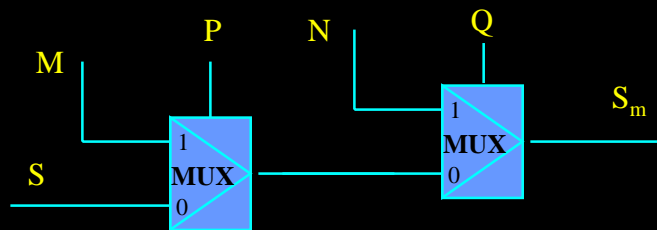
May 9, 2000

ECE 510 OCE: BDDs and Their Applications

16

Test Generation for Double Faults

- Let us introduce two modifiers controlled by signals P and Q



- This is a formal way to describe processes that occur in the circuit when two signals S_1 and S_2 in different locations are replaced by M and N

May 9, 2000

ECE 510 OCE: BDDs and Their Applications

17

Fault Localization

- Let $\Psi(a,b,c, p_0,p_1,p_2)$ be a vector of sensitivity functions for each output
- If $\Psi_i(a,b,c, p_0,p_1,p_2) = 1$, then observing the response of i -th output of the function to the test (a,b,c) detects fault (p_0,p_1,p_2)
- If $\Psi_i(a,b,c, p_0,p_1,p_2) = 0$, then observing the response of i -th output of the function to the test (a,b,c) does not detect fault (p_0,p_1,p_2) , assuming this fault exists in the circuit

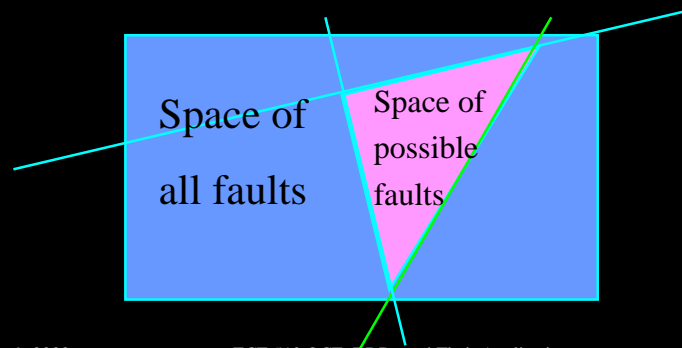
May 9, 2000

ECE 510 OCE: BDDs and Their Applications

18

Fault Localization for 3-Output Circuit

- After applying one test pattern, the search space is reduced



May 9, 2000

ECE 510 OCE: BDDs and Their Applications

19

Finding the Shortest Test Sequence to Localize a Fault

- Compute relation $F(a,b,c, p_0,p_1,p_2, y_0,y_1,y_2)$, which combines tests, faults, and output patterns
- For each test, group faults producing the same output pattern into equivalence classes
- Find next test by applying one of the following greedy heuristics at each iteration:
 - Select a test that splits faults into as many equivalence classes as possible
 - Select a test that splits faults into classes, such that the maximum number of faults in each class is minimized
- Reduce relation F after each iteration

May 9, 2000

ECE 510 OCE: BDDs and Their Applications

20

ATPG for Sequential Circuits

- Compute the sensitivity function for combinational logic
- Compute the set of reachable states
- Find the set of **combinationally redundant** and **sequentially non-excitable faults**
- Build the product machine of the fault-free FSM and the FSM with faults encoded by variables p
- Perform reachability analysis on the product machine to determine test sequences for all faults that can be detected (the remaining faults are **non-distinguishable**; they change STG but never manifest at the outputs)

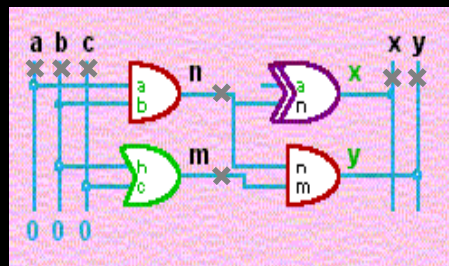
May 9, 2000

ECE 510 OCE: BDDs and Their Applications

21

Structural Representation of Networks

- a and n are in relation **AND2-direct**
- n and a are in relation **AND2-reverse**
- a and b are in relation **AND2-adjacent...**



May 9, 2000

ECE 510 OCE: BDDs and Their Applications

22

Encoding Gates and Signals

- Assign each signal ("cut point"), each type of gate, and relation a unique binary code (label)
 - $a - \bar{x}_2\bar{x}_1\bar{x}_0$ $b - \bar{x}_2\bar{x}_1x_0$ $n - \bar{x}_2x_1\bar{x}_0$
 - AND2 - $\bar{g}_1\bar{g}_0$ OR2 - \bar{g}_1g_0 EXOR2 - $g_1\bar{g}_0$
 - Direct - $\bar{a}_1\bar{a}_0$ Reverse - \bar{a}_1a_0 Adjacent - $a_1\bar{a}_0$
- Create products for each couple of objects (cut points in the network) and multiply these products by labels, defining their relationship
- Compute a bdd equal to the sum of these products

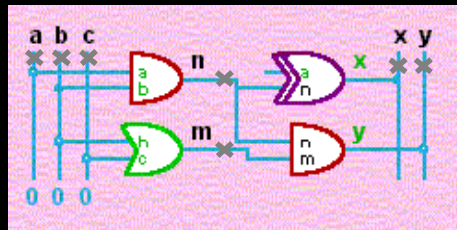
May 9, 2000

ECE 510 OCE: BDDs and Their Applications

23

Example

- $R(x_2, x_1, x_0, y_2, y_1, y_0, g_1, g_0, a_1, a_0) =$
 $\bar{x}_2\bar{x}_1\bar{x}_0\bar{y}_2\bar{y}_1\bar{y}_0\bar{g}_1\bar{g}_0\bar{a}_1\bar{a}_0 + \bar{x}_2x_1\bar{x}_0\bar{y}_2\bar{y}_1\bar{y}_0\bar{g}_1\bar{g}_0\bar{a}_1a_0 +$
 $\bar{x}_2\bar{x}_1x_0\bar{y}_2\bar{y}_1\bar{y}_0\bar{g}_1\bar{g}_0\bar{a}_1a_0 + \bar{x}_2x_1x_0\bar{y}_2\bar{y}_1\bar{y}_0\bar{g}_1\bar{g}_0\bar{a}_1a_0 +$
 $\bar{x}_2\bar{x}_1\bar{x}_0\bar{y}_2\bar{y}_1y_0\bar{g}_1\bar{g}_0a_1\bar{a}_0 + \bar{x}_2\bar{x}_1x_0\bar{y}_2\bar{y}_1y_0\bar{g}_1\bar{g}_0a_1\bar{a}_0 + \dots$



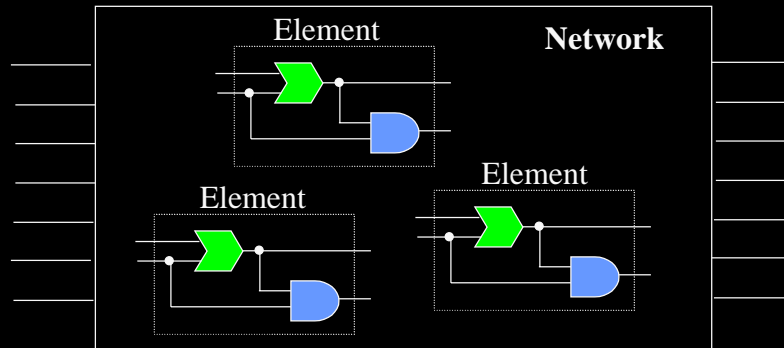
May 9, 2000

ECE 510 OCE: BDDs and Their Applications

24

Problem1: Locating Elements

- Find all occurrences of the combination of AND-OR gate in the network of logic gates



May 9, 2000

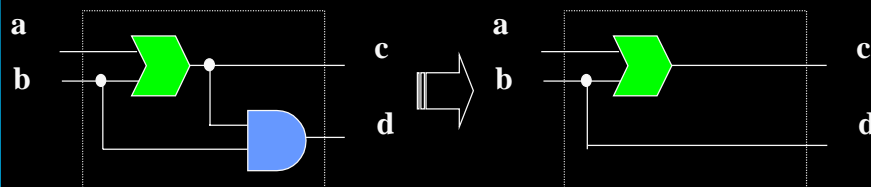
ECE 510 OCE: BDDs and Their Applications

25

Problem2: Replacing Elements

Replace a given element with another one in all places where it occurs. For example:

$$c = a + b \quad d = (a + b) \& b = a \& b + b = b$$



May 9, 2000

ECE 510 OCE: BDDs and Their Applications

26