Laboratory on Binary Decision Diagrams

Gianpiero Cabodi *Stefano Quer*

Politecnico di Torino

Torino, Italy

{gianpiero.cabodi,stefano.quer}@polito.it

http://staff.polito.it/{gianpiero.cabodi,stefano.quer}/

Outline

- BDD – Apply: Circuit Representation
- A Demo-BDD-WEB Package: TUDD
- A BDD Package: CUDD
  - Simple Usage
- A BDD Package: CUDD
  - Hints for Advanced Usage: How to use CUDD
  - Problem Solving
    - BDD calculator and equivalence verifier
    - N-queen Problem

    (select one)

Outline

- BDD – Apply: Circuit Representation
- A Demo-BDD-WEB Package: TUDD
- A BDD Package: CUDD
  - Simple Usage
- A BDD Package: CUDD
  - Hints for Advanced Usage: How to use CUDD
  - Problem Solving
    - BDD calculator and equivalence verifier
    - N-queen Problem

    (select one)

Increasing difficulty

(solutions available on the teacher WEB page)

(stop when desired … have fun!!!)

BDD – Apply: Circuit Representation

- Build the BDD (step-by-step) of the following functions:
  - $f_1 (x_1, x_2, x_3, x_4) = x_1 \cdot x_2 + x'_3 \cdot x_4 \oplus x_1 \cdot x'_2 \cdot x_3 \cdot x_4$
  - $f_2 (x_1, x_2, x_3, x_4) = x_1 \cdot (x_2 + x'_3 \cdot x_4) \oplus x_3 \cdot (x_1 + x'_2)$
  - $f_3 (x_1, x_2, x_3, x_4) = (x_1 + x_2) \cdot (x'_3 + x_2 \cdot x_3) \oplus x_4$
  - $f_4 (x_1, x_2, x_3, x_4) = x'_1 \oplus x'_2 \oplus x_3 \cdot x_4 + x_2 \cdot x'_3$
  - $f_5 (x_1, x_2, x_3, x_4) = (x_1 \cdot x_2)' + x'_3 \oplus x_1 \cdot x_2 \cdot (x_3 + x_4)$
  - $f_6 (x_1, x_2, x_3, x_4) = (x_1 + x_2 + x_3 \cdot x'_4) \oplus x_1 \cdot x_2 + x_3$
  - $f_7 (x_1, x_2, x_3, x_4) = x_1 \cdot (x_2 + x_3)' \oplus x_1 \cdot x_2 + x_4$
  - $g_1 = f_1 \cdot f_2$
  - $g_2 = f_3 + f_4$
  - $g_3 = (f_5 + f_6)'$
  - $g_4 = f_1 + f_7$
  - $g_5 = f_3 \cdot f_5$

TUDD

- Main features
  - Stephan Horeth - University TU Darmstadt
  - http://marple.rs.e-technik.tu-darmastadt.de/~sth/demo.html
  - Integrates different decomposition type
  - Demo-WEB page with good graphical interface
  - Package on request
- Laboratory duty
  - "Play" with TUDD, i.e., select
    - Function
    - Variable Order
    - Decomposition Type

  … toy-tool … have fun …

CUDD

- Main features
  - Fabio Somenzi - Boulder/Colorado WEB page
  - http://vlsi.colorado.edu/~fabio
    (copy in http://www.polito.it/~quer/teaching/phd/fv/laib)
  - Most widly used BDD package
  - Integrates BDD, ADD (Algebraic Decision Diagram), ZDD (Zero-Suppresed Decision Diagrams)
  - Very Efficient
  - Many Realeases over the years … now version -2.3.1
  - (Includes the dddmp package from G. Cabodi and S. Quer)

Simple Usage

- Laboratory Duty
  - Grab and uncompress it
  - Compile it
    - See Makefile in the root directory
    - Small modification IFF necessary (architecture parameters, directory positions, etc.)
    - "Run"

      Make
  - Check main features out
    - See documentation

      Directory cudd/doc - File cudd.doc (text file)

- Build BDD for standard ISCAS benchmarks (with nanotrav):
  - Combinational benchmarks
    - c17.blif, c..., etc.
  - Sequential benchmarks
    - s713.blif

      #PI=35, #PO=23, #FF=19, #Gate=393
    - s1512.blif

      #PI=29 , #PO=21, #FF=57 , #Gate=780
    - s1423.blif

      #PI=17, #PO=5, #FF=74, #Gate=657

      (from now on <c>)
    - s298, s1196, s1238, , s1488, s1494

    save output results …

Advanced Usage

- Hints to use CUDD:
  - DD Manager
    - Type
      - DdManager *
    - Functions
      - Cudd_Init
  - Elementary BDD Variables
    - Type
      - DdNode *
    - Functions (somehow similar)
      - Cudd_bddIthVar
      - Cudd_bddNewVar

- Build BDDs
  - Start from constant one (get zero from "not" (one))
  - Functions (somehow similar)
    - Cudd_ReadOne
    - DD_ONE
  - Proceed through the "circuit/function"
    - Cudd_Not
    - Cudd_bddAnd
    - Cudd_bddOr
    - etc.
  - Each new BDD has to be referenced
    - Cudd_Ref
  - Useless node must be dereferenced
    - Cudd_RecursiveDeref

- Check Results and Statistics
  - Functions
    - Cudd_CountMinterm
    - Cudd_PrintMinterm
    - Cudd_DagSize
- Quit the manager
  - Function
    - Cudd_quit

Problem solving 1

Boolean Function Manipulation and (combinational) Equivalence Checker

- Write Program
  - booleanOp
- Run it as
  - booleanOp <fileName1> <op> <fileName2>

  where
  - <fileName1> and <fileName2> are files containing a function description in PLA format
  - <op> is the operation

    a stands for and

    o stands for or

    x stands for xor

    e stands for (combinational) equivalence
- Result
  - report statistics on resulting function

    (e.g., print out BDD minterms, PLA format)

- **Example**

```
File 1              File 2
4                   1
0010 1              -0-0 1
0000 1
1010 1
1000 1
```

$F_1 = \neg a \cdot \neg b \cdot c \cdot \neg d + \neg a \cdot \neg b \cdot \neg c \cdot \neg d + a \cdot \neg b \cdot c \cdot \neg d + a \cdot \neg b \cdot \neg c \cdot \neg d$

$F_2 = \neg b \cdot \neg d$

- **Run as**
  booleanOp File1 e File2
- **Result**
  f in File1 == f in File2 !!!

Problem solving 2

The N-quenn problem (chess puzzle)

- **Write Program**
  - **8queen**
- **Run it as**
  - **8quenn <N>**

  where
  - **<N> specifies the board size (N x N)**
- **Result**
  - **Report number of solutions**
  - **(Eventually) the solution themselves (somehow coded)**

Coding the Problem

- **Chess Board NxN**
- **For each position in the variable create variable $x_{i,j}$**
  (i row index, j column index, from 1 to N)
- **Relations (constraints – no queen in conflict)**
  - **Iff there is a queen in i, j no other queen in row i**
    $X_{i,j} \Rightarrow \prod_k \neg X_{i,k}$, with $k = [1, N], k \neq j$
  - **Iff there is a queen in i, j no other queen in column j**
    $X_{i,j} \Rightarrow \prod_k \neg X_{k,j}$, with $k = [1, N], k \neq i$
  - **Iff there is a queen in i, j no other queen in same diagonal**
    $X_{i,j} \Rightarrow \prod_k \neg X_{k,j+k-i}$, with $k = [1, N], j+k-i = [1, N], k \neq i$
  - **Iff there is a queen in i, j no other quenn in same inverse diagonal**
    $X_{i,j} \Rightarrow \prod_k \neg X_{k,j+i-k}$, with $k = [1, N], j+i-k = [1, N], k \neq i$

- **Relations (constraints – enough queens)**
  - **There must be a queen for each row**
    $X_{i,1} \vee X_{i,2} \vee X_{i,3} \vee \ldots \vee X_{i,N}$, for all row $i = [1, N]$
- **Final Relation**
  - **Taking the conjunction of all the previous ones**
  - **When true there is a solution**

N.B.

A	B	A⇒B
0	0	1
0	1	1
1	0	0
1	1	1

$A \Rightarrow B = \neg A \vee B$