Computing Minimal DNF of Boolean Functions for Digital Implementations

Reni Banov

Zagreb University of Applied Sciences

20^{th} Central European Conference on Cryptology, June 2020

Table of Contents

Introduction

Modern communication systems impose strong constraints to digital circuit implementation of the Boolean function, such as

- reliability
- performance
- cost
- security

Notice

Constraints do not always appear in this order of importance.

A possible solution is to minimize the Boolean function expression form

$$f : \mathbb{B}^n \to \mathbb{B},$$

by which the number of gates is reduced.

Which form to choose for the minimization?

- **ANF** – Reed-Muller expansion

$$f(x_1, \dots, x_n) = \bigoplus_k \Big(\bigwedge_{i_k=1}^{n_i} x_{i_k} \Big), \qquad n_i \leq n$$

- **CNF** – Product-Of-Sums

$$f(x_1, \dots, x_n) = \bigwedge_k \Big(\bigvee_{i_k=1}^{n} \xi_{i_k} \Big), \qquad \xi_i \in \{x_i, \overline{x_i}\}$$

- **DNF** – Sum-Of-Products

$$f(x_1, \dots, x_n) = \bigvee_k \Big(\bigwedge_{i_k=1}^{n} \xi_{i_k} \Big), \qquad \xi_i \in \{x_i, \overline{x_i}\}$$

What is the minimal form of DNF?

The conjunction term p containing *some* literals

$$p = \wedge \xi_i,\ i = 1, \ldots, k \leq n$$

having no sub terms

$$q \subset p : q(\mathbf{x}) = 1 \implies f(\mathbf{x}) = 1$$

is called *prime implicants*.

Minimal DNF

The DNF with a minimum number of *prime implicants*.

Such a minimal DNF form, being implemented in the digital circuit, is optimal in the restricted class of two-level digital circuits (disjunctions of conjunctions of literals) gates [Weg91].

Why DNF is selected?

NAND implementation

It is easy to implement a Boolean function with NAND gates only if converted from a DNF form.

For example, $f(\mathbf{x}) = (x_1 \wedge x_2) \vee (x_3 \wedge x_4) = \overline{\overline{(x_1 \wedge x_2)} \wedge \overline{(x_3 \wedge x_4)}}$

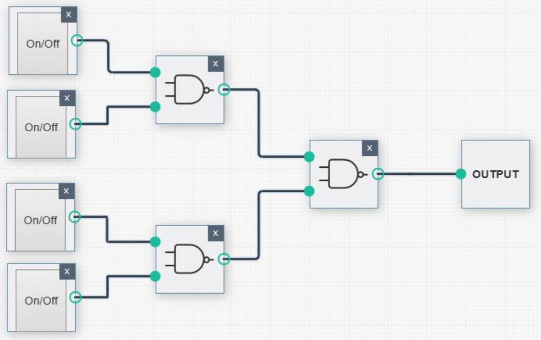


Table of Contents

How to find a minimal DNF of the Boolean function?

- Karnaugh maps – expression simplification; appropriate for functions defined on less than ≈ 6 variables
- Minterms (Quine-McClusky [Weg91]) – conjunction term simplification; appropriate for functions defined on less than ≈ 15 variables
- Heuristic (Espresso [MSBS93]) – prime implicants heuristic search; appropriate for functions with less than ≈ 50 variables
- BDDs (Coudert [CM94]) – set cover implicit computation; applicable for functions with ≈ 100 variables or more

Monotonic Boolean functions

Minimal DNF of monotic functions can be computed with BDDs according to the Rauzy approach [Rau93].

What are BDDs?

One of the only really fundamental data structures that came out in the last twenty-five years (D.E. Knuth, 2008)

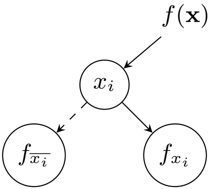
Bryant [Bry86]

The Boolean Decision Diagrams (BDDs) a variant of *directed acyclic graphs* (DAGs) used for Boolean functions representation.

Two-terminal DAG based on Shannon identity

$$f(\mathbf{x}) = \big(x_i \wedge f_{x_i}(\mathbf{x})\big) \vee \big(\overline{x_i} \wedge f_{\overline{x_i}}(\mathbf{x})\big)$$

built from vertices representing the *If-Then-Else* (ITE) construct.

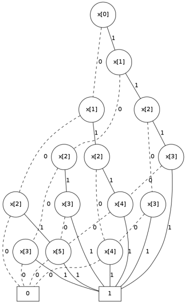


Why and when BDDs?

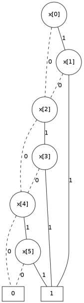
Canonicity

The BDDs are a canonical representation of the Boolean function.

but, they are sensitive to the selection of the variable order



Bad order[1]



Good order

[1] Source: PyEDA documentation [Dra20]

Any strong results for BDDs? I

Hardness of variable ordering

- The problem of finding an optimal order is *NP-hard* [TY00], and even by improving a variable order, the problem is *NP-complete* [BW96],
- There are Boolean functions which have an exponential size BDD for every ordering [Bry86]

Friedman-Supowit [FS90]

Let $I \subseteq \{1, \ldots, n\}, k = |I|, v \in I$, then there is a constant c such that for each $\pi \in \Pi(I)$ satisfying $\pi[k] = v$, we have

$$cost_v(f, \pi) = c.$$

Any strong results for BDDs? II

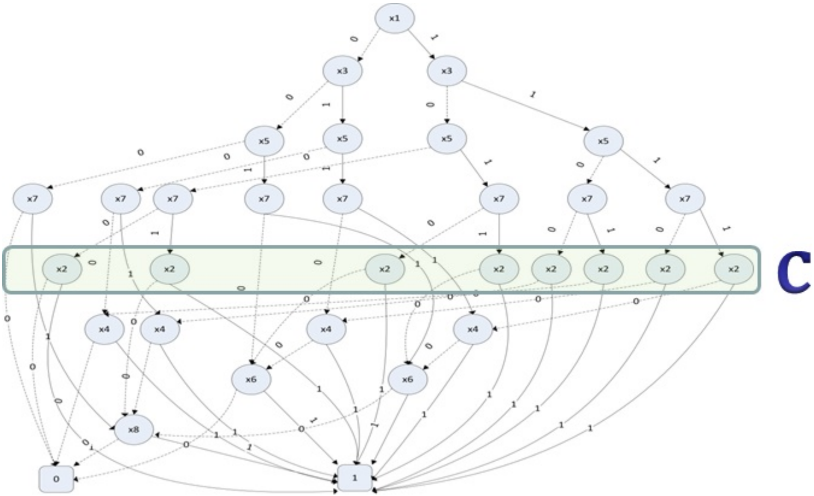


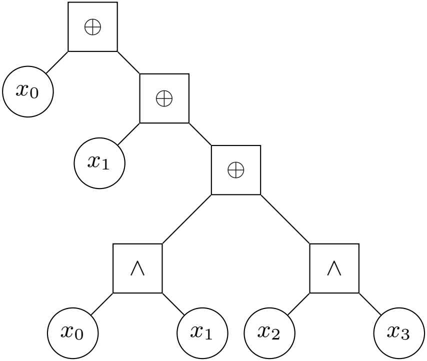
Table of Contents

Parse tree

Parse tree

First step: to build the parse tree of the Boolean function expression

Option: to build a Boolean Expression Diagram (BED) to remove redundant subexpressions [AH02]



$$f(\mathbf{x}) = x_0 \oplus x_1 \oplus (x_0 \wedge x_1) \oplus (x_2 \wedge x_3)$$

A new variable ordering heuristic

Input: f top node in parse tree or BED
Output: π a new variable order for top node

```
Orders ← ∅;                                  // Hash table for orders
for each node ∈ DFSOrder(f) do
    if node is Variable then
        x_i ← Variable(node)
        π_i ← {(x_i, count : 1)}
        Orders.Insert(key:node, value:π_i)
    end
    else
        /* node is boolean operator                          */
        π_l ← Orders.FindValue(key:LeftChild(node))
        π_r ← Orders.FindValue(key:RightChild(node))
        π ← MergeOrders(π_l, π_r)              // Merge heuristic
        Orders.Insert(key:node, value:π)
    end
end
π ← Orders.FindValue(key:f)
return π        // final order of variables for f node
```

$MergeOrders(\pi_l, \pi_r)$

```
Input: π_l, π_r two variable orders for merging
π ← ∅
(x_l, c_l) ← First(π_l)
(x_r, c_r) ← First(π_r)
while (x_l ≠ ∅) & (x_r ≠ ∅) do
    /* Heuristic criteria                                */
    if (c_l + 2 * c_r) ≤ (c_r + 2 * c_l) then
        Append_If_Not_Present(π, {x_l, count : c_l + 2 * c_r})
        (x_l, c_l) ← Next(π_l)
    end
    else
        Append_If_Not_Present(π, {x_r, count : c_r + 2 * c_l})
        (x_r, c_r) ← Next(π_r)
    end
end
if x_l ≠ ∅ then
    Append(π, from:x_l, π_l)                // Left finished?
end
if x_r ≠ ∅ then
    Append(π, from:x_r, π_r)                // Right finished?
end
return π                                    // Two orders merged
```

Results

Table: Preliminary results

example	variables	operators	$\lvert BDD\rvert$ [1]	$\lvert DNF\rvert$ [2]
$ex1$	13	12	15	11
$ex2$	70	53	254	36.292
$ex3$	43	32	57	1.043
$ex4$	66	50	281	32.369
$ex5$	44	35	49	784
$ex6$	98	141	264	960
$ex7$	18	19	31	46
$ex8$	61	84	2.481	46.188
$ex9$	194	158	3.048	34.477.555

[1] count of nodes in BDD built with heuristic order
[2] count of prime implicants for minimal DNF form

Table of Contents

Why minimal DNF and BDDs?

Some reasons for minimal DNF in digital implementations

- Reduced cost due to the minimal number of gates used
- Improved performance due to minimized total delay
- Design constraints fullfiled (die size, thermal, . . .)
- More reliable circuits

What else can be done with BDDs besides the DNF minimization?

- Expressing (and solving) of a $0/1$ optimization problems
- Representing a structure function of complex systems
- Could properties of the Boolean functions be derived from topological properties of the BDD DAGs
- Other ideas?

Thank You!

References I

Henrik Reif Andersen and Henrik Hulgaard, *Boolean expression diagrams*, Information and Computation **179** (2002), no. 2, 194 – 212.

Bryant, *Graph-based algorithms for boolean function manipulation*, IEEE Transactions on Computers **C-35** (1986), no. 8, 677–691.

B. Bollig and I. Wegener, *Improving the variable ordering of obdds is np-complete*, IEEE Transactions on Computers **45** (1996), no. 9, 993–1002.

O. Coudert and J. C. Madre, *Metaprime: an interactive fault-tree analyzer*, IEEE Transactions on Reliability **43** (1994), no. 1, 121–127.

References II

- C. Drake, *Python Electronic Design Automation*, 2020.
- S. J. Friedman and K. J. Supowit, *Finding the optimal variable ordering for binary decision diagrams*, IEEE Transactions on Computers **39** (1990), no. 5, 710–713.
- P. C. McGeer, J. V. Sanghavi, R. K. Brayton, and A. L. Sangiovanni-Vicentelli, *Espresso-signature: a new exact minimizer for logic functions*, IEEE Transactions on Very Large Scale Integration (VLSI) Systems **1** (1993), no. 4, 432–440.
- Antoine Rauzy, *New algorithms for fault trees analysis*, Reliability Engineering & System Safety **40** (1993), no. 3, 203 – 211.

References III

Yasuhiko Takenaga and Shuzo Yajima, *Hardness of identifying the minimum ordered binary decision diagram*, Discrete Applied Mathematics **107** (2000), no. 1, 191 – 201, SI Boolean Functions.

I. Wegener, *The complexity of boolean functions*, Wiley Teubner, ISBN: 978-0-471-91555-3, 1991.