Multiplicative Complexity of Boolean Functions

Meltem Sönmez Turan & René Peralta
National Institute of Standards and Technology

Objective

- Given a target Boolean function $\mathbf{f}$, find an implementation of $\mathbf{f}$ over the basis (AND, OR and NOT) that requires a minimum number of AND gates.

Introduction

- An $\mathbf{n}$-variable Boolean function is a mapping from $\{0,1\}^n$ to $\{0,1\}$.
- **Multiplicative Complexity** is the **minimum** number of multiplications (AND gates) that are necessary and sufficient to evaluate the function over the basis (AND, XOR, NOT).

Why do we count the number of AND gates?

- **Lightweight cryptography:** Efficient implementations are needed for resource-constrained devices (e.g., RFIG tags). The technique of *minimizing the number of AND gates, and then optimizing the linear components* leads to implementations with low gate complexity.
- **Secure multi-party computation:** Reducing the number of AND gates improves the efficiency of secure multi-party protocols (e.g. conducting online auctions in a way that the winning bid can be determined without opening the losing bids).
- **Side channel attacks:** Minimizing the number of AND gates is important to prevent side channel attacks such as differential power analysis.
- **Cryptanalysis of cryptographic primitives:** Primitives with low multiplicative complexity can be broken with algebraic attacks.

Example (Threshold function)

$$\mathbf{T}_3^5(x_1,x_2,x_3,x_4,x_5) = \begin{cases} 1, & \text{if majority of } \{x_1,x_2,x_3,x_4,x_5\} \text{ is } 1, \\ 0, & \text{otherwise.} \end{cases}$$

Its Algebraic Normal Form (ANF) is

$$\begin{aligned}\mathbf{T}_3^5 = {} & x_1x_2x_3 + x_1x_2x_4 + x_1x_2x_5 + x_1x_3x_4 + x_1x_3x_5 + x_1x_4x_5 + x_2x_3x_4 \\ & + x_2x_3x_5 + x_2x_4x_5 + x_3x_4x_5 + x_1x_2x_3x_4 + x_1x_2x_3x_5 + x_1x_2x_4x_5 \\ & + x_1x_3x_4x_5 + x_2x_3x_4x_5\end{aligned}$$

(where arithmetic is modulo 2). The equation contains 35 multiplications. **Surprisingly, only 3 multiplications are enough to compute it!**.

Affine Equivalence

- An *affine transformation* from $\mathbf{g}$ to $\mathbf{f}$ in $\mathbf{B_n}$ is a mapping of the form $\mathbf{f(x) = g(Ax + a) + b \cdot x + c}$, where $\mathbf{A}$ is a non-singular $\mathbf{n \times n}$ matrix over $\mathbb{F}_2$; $\mathbf{x, a}$ are column vectors over $\mathbb{F}_2$; $\mathbf{b}$ is a row vector over $\mathbb{F}_2$; and $\mathbf{c} \in \mathbf{F}_2$.
- $\mathbf{f, g}$ are *affine equivalent*, if there exist affine transformations between them.
- **Multiplicative complexity is affine invariant on Boolean functions**, i.e., applying affine transformations does not change multiplicative complexity.

Method

Precomputation

1. Find a "simple" (e.g., one with a small number of monomials in its ANF) representative for each equivalence class in $\mathbf{B_n}$.
2. For each representative, find an optimal implementation with respect to the number of AND gates.

Online Phase

1. Find the equivalence class $\mathbf{C_f}$ of $\mathbf{f}$.
2. Find the affine transformation from $\mathbf{f}^*$, the representative of $\mathbf{C_f}$, to $\mathbf{f}$.
3. Apply the affine transformation to the optimal implementation of $\mathbf{f}^*$. This yields an optimal implementation for $\mathbf{f}$.

Results on 4-variable Boolean Functions

There are **65 536** four-variable Boolean functions, **but only 8 equivalence classes**.

Class	*Representatives*
1	x_1
2	x_1x_2
3	$x_1x_2 + x_3x_4$
4	$x_1x_2x_3$
5	$x_1x_2x_3 + x_1x_4$
6	$x_1x_2x_3x_4$
7	$x_1x_2x_3x_4 + x_1x_2$
8	$x_1x_2x_3x_4 + x_1x_2 + x_3x_4$

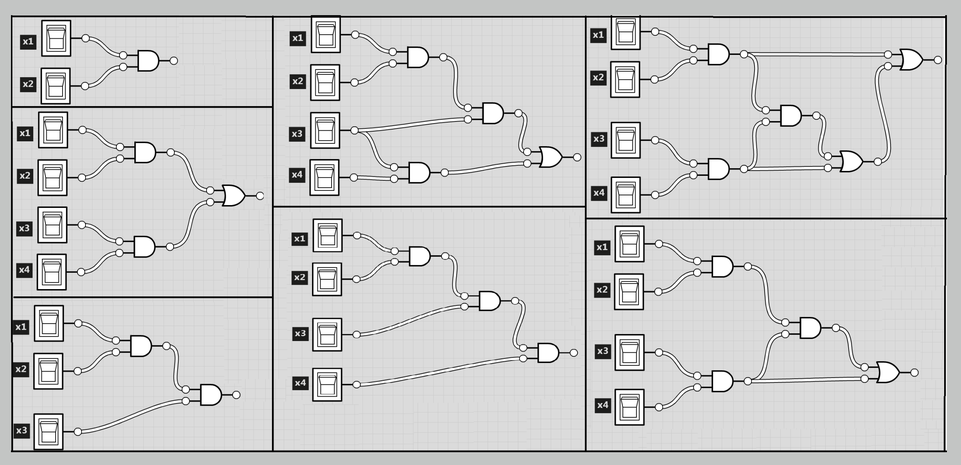

Figure 1: Circuits for all nonlinear representatives. No circuit uses more than 3 AND gates.

Four-variable Boolean functions can be implemented using at most 3 AND gates.

Example

Target function: $\mathbf{g} = x_1x_2x_3x_4 + x_1x_2x_3 + x_1x_2x_4 + x_1x_3x_4 + x_2x_3x_4 + x_1x_2 + x_1x_3 + x_1x_4 + x_2x_3 + x_2x_4 + x_3x_4 + x_1 + x_2 + x_3 + x_4 + 1$

$\mathbf{f} = x_1x_2x_3x_4$ and $\mathbf{g}$ are affine equivalent under the following transformation.

$$x_1 \to (x_1 + 1) \quad x_2 \to (x_2 + 1)$$
$$x_3 \to (x_3 + 1) \quad x_4 \to (x_4 + 1)$$

$\mathbf{g}$ can be implemented as $(x_1 + 1)(x_2 + 1)(x_3 + 1)(x_4 + 1)$ using only 3 AND gates.

Results on 5-variable Boolean Functions

There are **4 294 967 296** five-variable Boolean functions, **but only 48 equivalence classes**.

Five-variable Boolean functions can be implemented using at most 4 AND gates.

Conclusion and Future Work

- We provided efficient implementations of four and five-variable Boolean functions, in term of multiplicative complexity.
- We plan to extend the work to six-variable Boolean functions. The approach becomes impractical as the number of variables increases:
  - The number of equivalence classes increases exponentially with the number of variables.
  - Constructing optimal implementations for the representatives gets harder.
  - Finding an affine transformation from the representative to the target function gets harder.

Contact Information

- Web: http://cs-www.cs.yale.edu/homes/peralta/CircuitStuff/CMT.html
- Email: meltem.turan@nist.gov & peralta@nist.gov