Shanghai Jiao Tong University

School of Electronic Information and Electrical Engineering

# A new combinational logic minimization tech

Zhaole Li

Workshop of AES, 2022

Section 1

# Introduction

# Optimal combinational circuits

The meaningful metrics for constructing optimal combinational circuits are gate count, depth, energy consumption, etc.
The number of $n$-variable Boolean functions is $2^{2^n}$, so no known techniques can even find the optimal circuits for $8$-variable Boolean functions. Thus we build the implementations using some heuristics.

1. This work presented a new technique for circuit implementations with two steps:
    1. Reducing multiplicative complexity for the non-linear components;
    2. Then optimizing the linear components.
2. The metric is gate count with AND, XOR and 1;

# First Step

## Definition

The multiplicative complexity of a function is the number of $\mathbb{F}_2$ multiplications necessary and sufficient to compute it.

## Example

The multiplicative complexity of $f(x_1, x_2, x_3, x_4) = x_1x_2x_3x_4 + x_1x_2x_3 + x_1x_2x_4 + x_2x_3x_4 + x_1x_2 + x_1x_3 + x_1x_4 + x_2x_3 + x_3x_4$ is no greater than $3$ since $f = (x_1 + 1)(x_2 + 1)(x_3 + 1)(x_4 + 1) + x_1 + x_2 + x_3 + x_4 + 1$ and is $3$ due to $\deg(f) = 4$.

# Second Step



The second step is composed by finding the maximal linear components of the circuit and minimizing the number of XOR gates needed. A new heuristic for the second step is proposed.

Section 2

# Non-linear components of Sbox in AES[1]

[1]Boyar J, Peralta R. A new combinational logic minimization technique with applications to cryptology[C]//International Symposium on Experimental Algorithms. Springer, Berlin, Heidelberg, 2010: 178-189.

The only non-linear component in AES's Sbox is to compute the inverse in the finite field field $\mathbb{F}_{2^8}$. Canright built a circuit for inverses in $\mathbb{F}_{2^8}$ by giving a circuit for inverses in $\mathbb{F}_{2^4}$. Using the same general technique but in different bases[2] $\{W, W^2, Z^2, Z^8\}$ we can represent an element $\Delta = \left(x_0 W + x_1 W^2\right) Z^2 + \left(x_2 W + x_3 W^2\right) Z^8$ of $\mathbb{F}_{2^4}$, and the inverse of this element $\Delta' = \left(y_0 W + y_1 W^2\right) Z^2 + \left(y_2 W + y_3 W^2\right) Z^8$ can be calculated as the following:

[2] $W$ is a root of $x^2 + x + 1$ over $\mathbb{F}_2$, $Z$ is a root of $x^2 + x + W$ over $\mathbb{F}_{2^2}$.

$$
\begin{cases}
-y_0 = x_1x_2x_3 + x_0x_2 + x_1x_2 + x_2 + x_3 \\
-y_1 = x_0x_2x_3 + x_0x_2 + x_1x_2 + x_1x_3 + x_3 \\
-y_2 = x_1x_0x_3 + x_0x_2 + x_0x_3 + x_0 + x_1 \\
-y_3 = x_1x_2x_0 + x_0x_2 + x_0x_3 + x_1x_3 + x_1
\end{cases}
$$

For $\mathbb{F}_{2^4}$ inversion, we take the method:

1. pick an equation and build an efficient circuit for it;
2. store the intermediate functions used in above for possible usage in the other equations;
3. iterate until all equations have been computed.

**Remark:**
It turns out that $3$ multiplications are enough to compute any functions on four variables.

# Optimal non-linear components

Shanghai Jiao Tong University

$$\begin{cases} -y_1 = (x_0x_2 + x_1)(x_2 + x_3) + x_3 \\ -y_3 = (x_0x_2 + x_3)(x_0 + x_1) + x_1 \\ -y_0 = (x_0x_2 + y_1)x_3 + y_1 + x_2 + x_3 \\ -y_2 = (x_0x_2 + y_3)x_3 + y_3 + x_0 + x_1 \end{cases}$$

This circuit needs $5$ AND gates and $11$ XOR gates.

Section 3

# Minimizing linear components[3]

[3] Boyar J, Peralta R. A new combinational logic minimization technique with applications to cryptology[C]//International Symposium on Experimental Algorithms. Springer, Berlin, Heidelberg, 2010: 178-189.

### Example

$$\begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}$$

Actually written in the equations $x_1 + x_2; x_1 + x_2 + x_3; x_1 + x_2 + x_3 + x_4; x_2 + x_3 + x_4$.
It's easy to see that we only need $4$ XOR to compute the linear component $v_1 = x_1 + x_2; v_2 = v_1 + x_3; v_3 = v_2 + x_4; v_4 = v_3 + x_1$.

# A new heuristic

Let $S$ be a set of linear functions. For any linear predicate $f$, we define the distance $\delta(S, f)$ as the minimum number of additions of elements from $S$ necessary to obtain $f$. For the linear Boolean function, initially $S$ is just the set of all variables $x_1, x_2, ..., x_n$, then a new base element is the form of two old base elements, update the $\delta(S, f)$ until $\delta(S, f) = 0$.

1. For the $(n, m)$-linear Boolean functions, we use the $m \times n$ matrix over $\mathbb{F}_2$ such as $f(\mathbf{x}) = M\mathbf{x}$. $S$ still be just the set of all variables $x_1, x_2, ..., x_n$;
2. Denote $Dist[]$ the distance from $S$ to the linear function given by rows of $M$, in fact, $Dist[i] = \delta(S, f_i)$ where $f_i$ is the $i^{th}$ linear function given by $M$;
3. Pick a new base element by adding two old base elements and then update $Dist[]$;
4. Iterate the last step until $Dist[] = (0, 0, ..., 0)$.

1. pick those that minimize the sum of new distances;
2. pick one that maximizing the Euclidean norm of the vector of new distances;

This criterion seems strange for maximizing. But we want a distance $(0, 2, 1)$ rather than $(1, 1, 1)$.

# Example



We bulid the circuit of the following equation system:

$$\begin{aligned}
y_0 &= x_0 + x_1 + x_2 \\
y_1 &= x_1 + x_3 + x_4 \\
y_2 &= x_0 + x_2 + x_3 + x_4 \\
y_3 &= x_1 + x_2 + x_3 \\
y_4 &= x_0 + x_1 + x_3 \\
y_5 &= x_1 + x_2 + x_3 + x_4
\end{aligned}$$

so the matrix $M$ is $\begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 \end{pmatrix}$.

# Example

1. In above situation, the initial $S$ is the set $\{[00001], [00010], [00100], [01000], [10000]\}$;
2. The initial distance is $Dist = [2, 2, 3, 2, 2, 3]$;
3. First choose the two coloumn which have the most $1$ in the same row;
4. ...

Shanghai Jiao Tong University

# Thank You

Zhaole Li · A new combinational logic minimization technique