LINEARIZATION OF FUNCTIONS REPRESENTED AS A SET OF DISJOINT CUBES AT THE AUTOCORRELATION DOMAIN

Osnat Keren, Ilya Levin and Radomir Stankovic

The project is supported by BSF grant "Optimization of BDD by using Autocorrelation Function"
Outline

- Introduction
- Linearization algorithms and minimization of logic functions
- Linearization over the disjoint cubes domain
- Experimental results
- Conclusions
Linear Decomposition

The linearization allows implementation of a multi-output logic function $f : \mathbb{GF}(2^n) \rightarrow \mathbb{GF}(2^k)$ as a superposition of a linear function $\sigma$ followed by a non-linear part, $f_\sigma$, having the minimal complexity.

Complexity (simplicity) criterion:
The number of two input logical gates required for implementing the function is proportional to $\mu(f)$ (Shannon-49,Karpovsky-76)

$$\mu(f) = \left| \left\{ (x_1, x_2) \mid x_1, x_2 \in \mathbb{GF}(2^n), d(x_1, x_2) = 1, \begin{cases} f(x_1) = f(x_2) \
\end{cases} \right\} \right|$$
The autocorrelation function

• Let \( f : GF(2^n) \rightarrow GF(2) \) a logic function of a single output. 

  The value of the autocorrelation function \( R_f \) of \( f \) at point \( \tau \in GF(2^n) \) is defined as

  \[
  R_f(\tau) = \sum_{x \in GF(2^n)} f(x)f(x+\tau),
  \]

• Let \( f : GF(2^n) \rightarrow GF(2^k) \) a logic function of a \( k \) outputs. 

  The the total autocorrelation function \( R_f \) of \( f \) is

  \[
  R_f(\tau) = \sum_{u \in GF(2^k)} \sum_{x \in GF(2^n)} f_u(x)f_u(x+\tau) = \sum_{u \in GF(2^k)} R_{f_u}(\tau)
  \]

  where \( f_u \) is the characteristic function of \( u \in GF(2^k) \)

• Theorem (Karpovsky 76):

  \[
  \mu(f) = \sum_{||\tau||=1} R(\tau) = R(I)
  \]
Linear decomposition

Representation of an element of $GF(2^n)$

- An element of $GF(2^n)$ can be represented as a linear combination of elements $\{x_i\}$ in the initial basis with a coefficients vector $z$
- It can also be represented by a set $\{\tau_i\}$ of vectors defined by $T$, i.e. $\tau_i = Tx_i$, with the coefficient vector $\hat{z}$ determined by $\hat{z} = T^{-1}z$.
- Example:

$$T = (\tau_2, \tau_1, \tau_0) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}.$$ 

- The linear transform matrix $\sigma$ is $\sigma = T^{-1}$
The main idea of linearization is

- Replace the initial set by another set of variables derived by a linear transform $\sigma$ over the initial variables such that $\mu(f_{\sigma_{\text{opt}}})$ is maximal.

$$
\mu_{\text{max}} = \max_\sigma \mu(f_\sigma) = \max_\sigma R_{f_\sigma}(I) = \max_\sigma R_f(\sigma^{-1}) = R_f(T),
$$

where $T = \sigma_{\text{opt}}^{-1}$ is a nonsingular $(n \times n)$ matrix whose columns $(\tau_{n-1}, \ldots, \tau_1, \tau_0)$, $\tau_i \in GF(2^n)$, form a basis and $\sum_i R_f(\tau_i)$ is maximal.
Linear decomposition (cont’)

Example:

Base vectors: \( x_0 = (001), x_1 = (010), x_2 = (100) \)

<table>
<thead>
<tr>
<th>( x_2 x_1 x_0 )</th>
<th>( f(x_2, x_1, x_0) )</th>
<th>( R(x_2, x_1, x_0) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>001</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>010</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>011</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>100</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>101</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>110</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>111</td>
<td>0</td>
<td>6</td>
</tr>
</tbody>
</table>

\( \tau_0 = (111), \tau_1 = (100), \tau_2 = (110) \)

<table>
<thead>
<tr>
<th>( \tau_2 \tau_1 \tau_0 )</th>
<th>( f_{\sigma}(\tau_2, \tau_1, \tau_0) )</th>
<th>( R(\tau_2, \tau_1, \tau_0) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>001</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>010</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>011</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>100</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>101</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>110</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>111</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

\( \mu(f) = 2 + 0 + 2 = 4 \)

The corresponding PLA has 11 literals

The corresponding MOBDD has 9 nodes

\( \mu(f_{\sigma}) = 6 + 2 + 2 = 10 \)

The corresponding PLA has 7 literals

The corresponding MOBDD has 7 nodes
Linear decomposition - principles

The linearization procedure consists of three steps:

1. Calculation of the autocorrelation function

2. Construction of a set of base vectors that span $GF(2^n)$ and have maximal autocorrelation values.

3. Construction of the corresponding linearized function $f_\sigma$
Methods for calculation of the autocorrelation function

Two approaches:

- Calculation by definition
- Calculation by the Wiener-Khinchin theorem

\[ R_f = 2^n W^{-1} (W f)^2, \]

where \( W \) is the Walsh transform operator.

The complexity of calculation of \( R_f \) depends on the way \( f \) is represented:

- Calculations of \( R_f \) over truth-vectors and decision diagrams by using the Wiener-Khinchin is faster \( (O(n2^n)) \) than calculations by the definition \( (O(2^{2n})) \)
- In disjoint cube representations, calculations can be performed separately over each cube or a pair of cubes.
Calculation of the autocorrelation function (cont’)

- **Calculation of the Walsh transform over disjoint cubes**
  complexity of the method depends on the number of disjoint cubes.
  Example:

  \[
  f(x_3x_2x_1x_0) = \bar{x}_3\bar{x}_2 + x_3\bar{x}_2x_1 + \bar{x}_3x_2\bar{x}_1x_0.
  \]

  \[
  F = W_f = [7, -3, 1, -1, 5, -1, -1, 1, 3, 1, 1, -1, 1, 3, -1, 1] \]

  The vector \(|F|^2\) can be covered by 9 disjoint cubes.

- **Tabular Technique**
  (Almaini,Thomson and Hanson-1991)
  Method to convert representation of a logical function from a sum-of-product form into a Reed-Muller expression. The method involves bit-by-bit operations on **minterms** and can be used for any number of input variables with complexity \(O(2^n)\)
Linearization Algorithms and Minimization of Logic Functions

- Varma and Trachtenberg (1989) - A linearization algorithm for efficient minimization of logic functions on the disjoint cubes domain
  - The linearization algorithm runs over the cubes.
  - Heuristic procedure determines candidate $\tau$’s that is likely to have high correlation value
  - If it finds a $\tau$ which is independent in previous $\tau$’s, a single value of $R_f$ is calculated at a time directly by definition and if this value is higher than the values calculated so far $\tau$ is included in the basis.
  - Main drawback - the final set of $\tau$’s depends on the order of processing the cubes and on $\tau$’s of previously produced cubes.
Linearization Algorithms and Minimization of Logic Functions (cont’)

- Karpovsky, Stankovic and Astola (2003) - $K$-procedure
  - Reduction of sizes of decision diagrams by autocorrelation functions
  - Minimizing the number of nodes per levels, by starting from the bottom of the BDD
  - The BDD is folded after each step.
  - The $K$-procedure may be inapplicable for functions of many variables

Decomposition of $\sigma$ in $K$-procedure
Linearization Algorithms and Minimization of Logic Functions (cont’)

Example:

\[ f(x_3, x_2, x_1, x_0) = x_3 \overline{x}_2 x_1 \overline{x}_0 + x_3 \overline{x}_2 x_1 \overline{x}_0 + x_3 \overline{x}_2 x_1 x_0 + x_3 x_2 x_1 x_0 \]

Folding by using the truth-table is straightforward:

<table>
<thead>
<tr>
<th>( x )</th>
<th>( f(x) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>0</td>
</tr>
<tr>
<td>0001</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>0111</td>
<td>0</td>
</tr>
<tr>
<td>1000</td>
<td>1</td>
</tr>
<tr>
<td>1001</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>1110</td>
<td>0</td>
</tr>
<tr>
<td>1111</td>
<td>1</td>
</tr>
</tbody>
</table>

Folding the function when represented by two disjoint cubes requires extracting the cubes into minterms:

\[
\begin{align*}
\{(1, \phi, 0, 0), (1)\} & \quad \rightarrow \quad \{(1, 0, 0, 0), (1)\} \\
\{(1, 1, \phi, 1), (1)\} & \quad \rightarrow \quad \{(1, 1, 0, 1), (1)\} \\
\end{align*}
\]
Linearization over the autocorrelation domain

Definitions:

- Let \( f : GF(2^n) \rightarrow GF(2^k) \) a system of \( k \) logic functions of \( n \) variables or Multioutput Logic Function. Let \( G = \{0, 1, \phi\} \), where \( \phi \) stands for don’t-care. The representation of \( f \) at the cubes domain is a set of \( N \) pairs

\[
F = \{(P_i, Y_i)\}_{i=1}^{N}
\]

where \( P_i \in G^n \), is a product and \( Y_i \in GF(2^k) \) is the corresponding output.

- The characteristic set, \( F_u, (u \in GF(2^k),) \) is the set

\[
F_u = \{(P_i, Y_i)| (P_i, Y_i) \in F, Y_i = u\}
\]

- Two cubes are called disjoint if they do not have any minterm in common. If any pair of cubes is disjoint the function is said to be of a disjoint cubes representation.
Linearization over the autocorrelation domain

a. Calculation of the auto correlation function over disjoint cubes domain

The autocorrelation function of the characteristic set $F_u$ is:

$$R_u(\tau) = \sum_{x \in GF(2^n)} \left( \bigvee_{i=0}^{N_u} P_i(x) \right) \left( \bigvee_{i=0}^{N_u} P_i(x + \tau) \right)$$

$$= \sum_{i=0}^{N_u} \sum_{j=0}^{N_u} \sum_{x \in GF(2^n)} P_i(x) P_j(x + \tau) = \sum_{i=0}^{N_u} \sum_{j=0}^{N_u} R_{i,j}^{(u)}(\tau)$$

Autocorrelation of a single cube (Karpovsky-76):

**Theorem 1** Denote by $n_\phi$ the number of symbols of a product $P_i = (p_{n-1}^{(i)}, \ldots, p_1^{(i)}, p_0^{(i)}) \in G^n$ that carry don't care. The autocorrelation $R_{i,i}(\tau)$ of $P_i(x)$ equals $2^{n_\phi}$ for any $\tau$ of the form $(\tau_{n-1}, \ldots, \tau_1, \tau_0)$, where

$$\tau_k = \begin{cases} \phi & p_k^{(i)} = \phi \\ 0 & otherwise \end{cases}$$

$(k = 1, 2, \ldots n - 1)$ and is zero elsewhere.
Linearization over the autocorrelation domain (cont’)

Cross correlation of two cubes:

Let $P_i$ and $P_j \in \mathcal{G}^n$ two products, denote by $p_k^{(i)}$ and $p_k^{(j)}$ the $k$’th symbol of $P_i$ and $P_j$ respectively. There are nine possible $(p_k^{(i)}, p_k^{(j)})$ pair types:

\[
(p_k^{(i)}, p_k^{(j)}) \in \left\{ T_1 = (0,0), T_2 = (0,1), T_3 = (0,\phi), T_4 = (1,0), T_5 = (1,1), T_6 = (1,\phi), T_7 = (\phi,0), T_8 = (\phi,1), T_9 = (\phi,\phi) \right\}.
\]

**Theorem 2** Let $P_i$ and $P_j \in \mathcal{G}^n$. Denote by $n_\phi$ the number of pairs $(p_k^{(i)}, p_k^{(j)})$ of type $T_9$. For any $\tau$ of the form $(\tau_{n-1}, \ldots, \tau_1, \tau_0)$, where

\[
\tau_k = \begin{cases} 
0 & (p_k^{(i)}, p_k^{(j)}) \in \{T_1, T_5\} \\
1 & (p_k^{(i)}, p_k^{(j)}) \in \{T_2, T_4\} \\
\phi & \text{otherwise}
\end{cases}
\]

the cross-correlation $R_{i,j}(\tau)$ equals $2^{n_\phi}$ and is zero elsewhere.
The autocorrelation function can be represented in PLA-like format by a set of $M$ pairs,

$$R = \{(C_i, V_i)\}_{i=1}^{M},$$

where

$$M \leq \sum_u (N_u + \binom{N_u}{2}) \leq N^2$$

and

$$1 \leq V_i \leq 2^n$$

Equivalently the autocorrelation function can be represented as an arithmetic sum of cubes,

$$R(\tau) = \sum_{i=1}^{M} C_i(\tau)V_i$$
Linearization over the autocorrelation domain (cont’)

Example:

\[
F = \begin{Bmatrix}
(0 1 0 0), 0 \\
(0 0 1 1), 0 \\
(1 - 0 0), 1 \\
(0 - 1 0), 1 \\
(0 1 0 1), 2 \\
(0 0 0 -), 2 \\
(1 - 1 -), 2 \\
(1 - 0 1), 3 \\
(0 1 1 1), 4 \\
\end{Bmatrix}
\Rightarrow R = \begin{Bmatrix}
R_0 \\
R_1 \\
R_2 \\
R_3 \\
R_4 \\
\end{Bmatrix} = \begin{Bmatrix}
(0 0 0 0), 2^0 \\
(0 0 0 0), 2^0 \\
(0 1 1 1), 2 \cdot 2^0 \\
(0 - 0 0), 2^1 \\
(0 - 0 0), 2^1 \\
(1 - 1 0), 2 \cdot 2^1 \\
(0 0 0 -), 2^2 \\
(0 0 0 -), 2^2 \\
(0 1 0 -), 2 \cdot 2^0 \\
(1 - 1 -), 2 \cdot 2^0 \\
(1 -1 -), 2 \cdot 2^1 \\
(0 - 0 0), 2^1 \\
(0 0 0 0), 2^0 \\
\end{Bmatrix} = \begin{Bmatrix}
(0 0 0 0), 4 \\
(0 1 1 1), 2 \\
(0 - 0 0), 6 \\
(1 - 1 0), 4 \\
(0 0 0 -), 2 \\
(0 - 0 -), 4 \\
(0 1 0 -), 2 \\
(1 - 1 -), 6 \\
\end{Bmatrix}
\]

The value of the autocorrelation function \( R(\tau) \) for \( \tau = (0100) \) is

\[
R(0100) = \sum_{i=1}^{8} C_i(0100) \cdot V_i = 6 + 4 + 2 = 12
\]
Linearization over the autocorrelation domain (cont’)

b. Construction of the basis

- Greedy algorithm, constructs a set of $n$ base vectors in $n$ steps.
- Each step select $\tau$ of maximal autocorrelation that is not an element of the subspace spanned by previous vectors.
- Avoid the complexity of verifying that $\tau$ is linearly independent by restricting the range of possible values of $\tau$.

$\mu$ is invariant to the order of the base vectors $\Rightarrow$ The base vectors can be reordered by increasing decimal value.

Example:

\[
T = \begin{pmatrix}
1 & 0 & 0 \\
1 & 0 & 1 \\
1 & 1 & 0 \\
\end{pmatrix} = (7, 1, 2) \Rightarrow T = \begin{pmatrix}
1 & 0 & 0 \\
1 & 1 & 0 \\
1 & 0 & 1 \\
\end{pmatrix} = (7, 2, 1)
\]

Note the matrix $(7, 6, 2)$ is nonsingular but does not satisfy this property
Construction of the basis (cont’)

• The autocorrelation functions of $f(x)$ and $f_{\sigma}(x)$ carry the same values, but in different positions,

$$f(x) = f_{\sigma}(\sigma x) \Rightarrow R_{f_{\sigma}}(\tau) = R_f(\sigma^{-1}\tau)$$

• Perform instantaneous linear transforms $\sigma_i$ on the autocorrelation function

$$R_i = \sigma_i R_{i-1}, \quad i = 1, \ldots, n$$

where $R_0 = R$ and $R_n$ is the autocorrelation function of the transformed set of cubes that represents $f_\sigma$, and

$$R_i(\delta_k) = R_{i-1}(\sigma_i^{-1}\delta_k) = R_{i-1}(T_i\delta_k)$$

where $\delta_k$ is the binary vector corresponding $2^k$ in base 2.

$\Rightarrow$ At step $i$ the maximal autocorrelation values are located at the positions $\tau = 2^k, \ k < i$

$\Rightarrow$ All $\tau$'s of the value greater or equal to $2^{i-1}$ are linearly independent in previously chosen vectors

• The linear transform matrix $\sigma$ is a product of $n' \leq n$ matrices

$$\sigma = \sigma_{n'-1} \cdots \sigma_1 \sigma_0$$
Linearization over the autocorrelation domain (cont’)

c. Linear transform of cubes

- The problem: the matrix $\sigma$ may break a cube into many cubes of smaller order $\Rightarrow$ high computational complexity

\[ \sigma P = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ \phi \\ 1 \\ \phi \end{pmatrix} = \sigma \begin{pmatrix} (0, 0, 1, 0) \\ (0, 0, 1, 1) \\ (0, 1, 1, 0) \\ (0, 1, 1, 1) \end{pmatrix} = \begin{pmatrix} (0, 0, 0, 1) \\ (1, 0, 1, 1) \\ (0, 1, 1, 1) \\ (1, 1, 0, 1) \end{pmatrix} \]

- Solution: Perform instantaneous linear transforms on the set of cubes

\[ F_i = \sigma_i F_{i-1}, \ i = 1, \ldots, n \]

$\Rightarrow$ The matrix $\sigma_i$ can be represented as a product of two matrices

\[ \sigma_i = L_i P_i \quad \text{and} \quad T_i = P_i L_i \]

where $P_i$ is a permutation matrix, and $L_i$ has ones on its diagonal and a single column of Hamming weight greater or equal one

$\Rightarrow$ A cube may be halved
Linearization procedure by instantaneous linear transforms of $R$

Linearization procedure:
Set $i = 0$
For all $\tau \in GF(2^n)$, $||\tau|| \leq w$ calculate $R(\tau)$. Set $\sigma = I$
While $i \leq n - 1$

1. If $R(\tau) = 0$ for all candidate $\tau$'s then break.
2. Determine $\tau$, $\tau \geq 2^{i-1}$ that maximizes $R(\tau)$. In case there is more than one $\tau$ choose one randomly
3. Construct the instantaneous linear transform matrix $\sigma_i$
4. Perform an instantaneous linear transform on $R$ and on the set of products
5. Update $\sigma$, $\sigma = \sigma_i \sigma$
6. Increment $i$

**Theorem 3:** The value of the complexity measure $\mu$ does not decrease throughout the procedure, namely,

$$\mu(F_0) \leq \mu(F_1) \leq \cdots \leq \mu(F_n)$$
Experimental Results

Sensitivity to the restriction on the Hamming weight of $\tau$

The cost function $\mu$ versus the restriction $w$ on the Hamming weight of $\tau$.

The $\mu$ of the original function equals to the $\mu$ calculated with $w = 1$.

<table>
<thead>
<tr>
<th>benchmark</th>
<th>$n$</th>
<th>$k$</th>
<th>$w = 1$</th>
<th>2</th>
<th>3</th>
<th>5</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>sqrt8.pla</td>
<td>8</td>
<td>4</td>
<td>1164</td>
<td>1284</td>
<td>1268</td>
<td>1286</td>
<td>1286</td>
</tr>
<tr>
<td>radd.pla</td>
<td>8</td>
<td>5</td>
<td>824</td>
<td>1304</td>
<td>1304</td>
<td>1304</td>
<td>1304</td>
</tr>
<tr>
<td>root.pla</td>
<td>8</td>
<td>5</td>
<td>868</td>
<td>932</td>
<td>940</td>
<td>958</td>
<td>958</td>
</tr>
</tbody>
</table>
Experimental Results

The cost function of the original function $\mu_{\text{orig}}$, the linearized benchmark functions ($\mu_k$ and $\mu_{dc}$) and an upper bound $\mu_{\text{up}}$ on the cost function.

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>$n$</th>
<th>$k$</th>
<th>$\mu_{\text{orig}}$</th>
<th>$\mu_k$</th>
<th>$\mu_{dc}$</th>
<th>$\mu_{\text{up}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>rd53.pla</td>
<td>5</td>
<td>3</td>
<td>50</td>
<td>66</td>
<td>82</td>
<td>104</td>
</tr>
<tr>
<td>sqr6.pla</td>
<td>6</td>
<td>12</td>
<td>114</td>
<td>114</td>
<td>124</td>
<td>196</td>
</tr>
<tr>
<td>rd73.pla</td>
<td>7</td>
<td>3</td>
<td>308</td>
<td>476</td>
<td>566</td>
<td>644</td>
</tr>
<tr>
<td>radd.pla</td>
<td>8</td>
<td>5</td>
<td>824</td>
<td>1112</td>
<td>1304</td>
<td>1556</td>
</tr>
<tr>
<td>dist.pla</td>
<td>8</td>
<td>5</td>
<td>638</td>
<td>638</td>
<td>690</td>
<td>1058</td>
</tr>
<tr>
<td>f51m.pla</td>
<td>8</td>
<td>8</td>
<td>884</td>
<td>1076</td>
<td>1204</td>
<td>1536</td>
</tr>
<tr>
<td>adr4.pla</td>
<td>8</td>
<td>5</td>
<td>1040</td>
<td>1212</td>
<td>1340</td>
<td>1492</td>
</tr>
<tr>
<td>dc2.pla</td>
<td>8</td>
<td>7</td>
<td>820</td>
<td>820</td>
<td>894</td>
<td>1310</td>
</tr>
</tbody>
</table>
Experimental Results

Run Time improvement

Figure 1: Average Run Time of $K$-procedure and suggested algorithm for random PLAs of 4 outputs and 50 cubes
Experimental Results

Sensitivity to the number of disjoint cubes: Average execution time versus number of inputs for random PLAs of 4 outputs and 50 and 100 products

- The complexity is polynomial with the number of cubes ($N^2$)
- The complexity is increasing as $n^4$ with the number of inputs and not exponentially ($n^2 2^n$)
Conclusions

- A method for calculation and compact representation of the autocorrelation function of a function of high number of input variables defined as disjoint set of cubes was presented.

- A technique for constructing the corresponding linear transform matrix was proposed.

- The computational complexity of the linearization procedure is of order \(\max(n^{w+2}, nN^2)\).

- This technique is proved to derive a linearized function having a \(\mu\) which is not smaller than the \(\mu\) of the original function.

- Experimental results clearly demonstrate the efficiency of the presented techniques.
THANK YOU
Computational Complexity

- The complexity can be reduced by restricting the Hamming weight of $\tau_k$ to be less or equal $w$. There are

$$W = \sum_{j=1}^{w} \left( \binom{n}{j} - \binom{k}{j} \right)$$

such $\tau$’s. From simulations $w = 3$ is sufficient.

- The autocorrelation calculation complexity is $O(nN^2)$ bit operations, where $N$ is the number of disjoint cubes, the memory size required to store $R$ is of order $\min(nW, nN^2)$ bits.

- The basis is constructed in $O(n^2W)$ bit operations.

- The linearized function is obtained in $O(n^2\tilde{N})$ bit operations where $\tilde{N}$ is the maximal number of disjoint cubes throughout the procedure.

- The computational complexity of the linearization procedure is of order $\max(n^{w+2}, nN^2)$.
Further Research

- Minimization of BDD’s
- Linear decomposition with other cost functions, i.e. $\mu$ of higher order.
- Parallel decomposition
- Efficient calculation of the autocorrelation function for special applications, e.g. watermark
- How linearization effects the power consumption
- Linearization in respect to path length distribution and/or critical paths length and pipeline balancing
- The effect of linearization on security and testability