View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Calhoun, Institutional Archive of the Naval Postgraduate School

Calhoun: The NPS Institutional Archive

Faculty and Researcher Publications

Faculty and Researcher Publications

1994-04

A design method for look-up table type FPGA by pseudo-Kronecker expansion

Sasao, Tsutomu

T. Sasao and J. T. Butler, "A design method for look-up table type FPGA by pseudo-Kronecker expansion" IEEE ISMVL-94, May 1994, pp. 97-106.

http://www.nps.edu/library

Calhoun is a project of the Dudley Knox Library at NPS, furthering the precepts and goals of open government and government transparency. All information contained herein has been approved for release by the NPS Public Affairs Officer.

Dudley Knox Library / Naval Postgraduate School
411 Dyer Road / 1 University Circle
Monterey, California USA 93943

A Design Method for Look-up Table Type FPGA by Pseudo-Kronecker Expansion

Tsutomu Sasao

Kyushu Institute of Technology
Iizuka 820, Japan

Jon T. Butler

Naval Postgraduate School
Monterey, CA 93943, U.S.A.

April 15, 1994

1 Introduction.

Field programmable gate arrays (FPGA's) are very useful in rapid prototyping as well as small volume production [3]. A look-up table (LUT) type FPGA shown in Fig. 1.1 consists of LUT's and programmable interconnection. Both LUT's and programmable interconnections are controlled by static RAMs. We assume that each FPGA has q LUT's, and each LUT can realize an arbitrary logic function of k binary variables. We also assume that interconnection resources are sufficient, i.e., any logical network with q LUT's can be realized. Note that wiring is implemented by pass transistors, and the delay time for interconnection will often be larger than for the LUT's. Consider the realization of a moderately complex function. If k is small, e.g., $k = 2$ or 3, the LUT's are efficiently used, but the interconnections will be complex. This will be especially inefficient since interconnections are often more expensive than logic. If k is large, e.g., $k = 7$ or 8, the interconnections will be simpler, but the LUT's are not efficiently used. Thus, there exists an optimum value for k between 3 and 7. One study [20] shows that when $k = 3$ or 4, FPGA's require the smallest chip area. However, these results have not been used by manufacturers. For example, for the XILINX 4000 Series FPGA's, $k = 5$, and for the AT&T ORCA FPGA's, $k = 6$ [1]. Various design methods for LUT-based FPGA's are known:

1) Technology mapping from AND-OR multi-level logic circuits [3].
2) Technology mapping from binary decision diagrams (BDD's) [2].
3) Functional decomposition [27, 12, 32].

Commercially available FPGA's contain many LUT's, but the interconnections are much slower than other standard mask type gate-arrays. Thus, it is often more important to reduce the propagation delay than to reduce the number of LUT's [5, 16]. To reduce the propagation delay, we have to reduce the number of the levels in the networks. Also, if the interconnections and layout are regular, then the propagation delay will be smaller.

In this paper, we propose a design method for LUT-based FPGA's that produces LUT networks having the

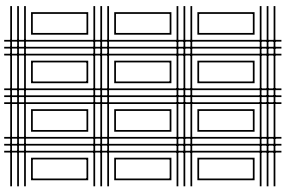

Figure 1.1: LUT type FPGA

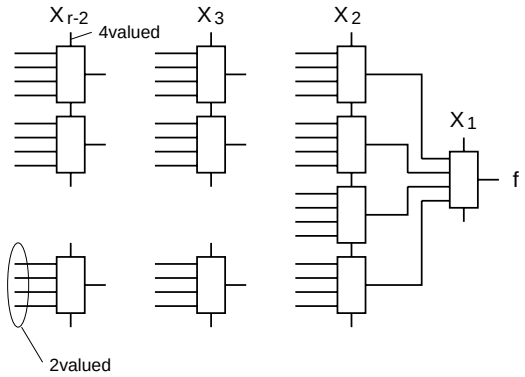

Figure 1.2: LUT network

structure shown in Fig. 1.2. We assume that each LUT has 6 binary inputs. By pairing variables of two-valued input functions, we can obtain four-valued input functions. By using four-valued logic, we can reduce the number of variables to consider, and can design a network having a regular structure. We use the pseudo-Kronecker expansion for four-valued input two-valued output functions to design compact and regular LUT networks. The proposed method is also promising for the four-valued LUT based FPGA's [26, 14, 31].

2 Realization of Two-Valued Input Functions With 3-input LUT's

Before studying the realization of logic functions with 6-input LUT's, it is convenient to review the

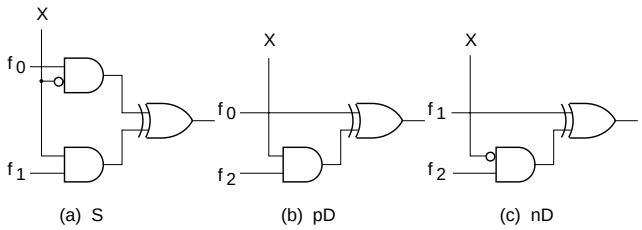

Figure 2.1: Logic circuits corresponding to three types of expansions

realization for two-valued input functions by 3-input LUT's. Consider three expansions:

$$f = \bar{x}_1 f_0 \oplus x_1 f_1, \tag{2.1}$$
$$f = f_0 \oplus x_1 f_2, \text{ and} \tag{2.2}$$
$$f = f_1 \oplus \bar{x}_1 f_2, \text{ where } f_2 = f_0 \oplus f_1. \tag{2.3}$$

(2.1), (2.2), and (2.3) are called the Shannon expansion, the positive Davio expansion, and the negative Davio expansion, respectively. Fig. 2.1 shows the logic circuits realizing these three expansions; they are denoted by S, pD, and nD, respectively. Note that a 3-input LUT directly realizes these circuits.

When the Shannon expansions are used to expand f_0 and f_1, we have

$$f_0 = \bar{x}_2 f_{00} \oplus x_2 f_{01}, \text{ and}$$
$$f_1 = \bar{x}_2 f_{10} \oplus x_2 f_{11}, \text{ respectively.}$$

In the similar way, we can expand the new sub-functions as follows:

$$f_{00} = \bar{x}_3 f_{000} \oplus x_3 f_{001},$$
$$f_{01} = \bar{x}_3 f_{010} \oplus x_3 f_{011},$$
$$f_{10} = \bar{x}_3 f_{100} \oplus x_3 f_{101}, \text{ and}$$
$$f_{11} = \bar{x}_3 f_{110} \oplus x_3 f_{111}.$$

Fig. 2.2 is the binary decision tree corresponding to the above expansion. If we replace each node by a two-input multiplexer (Fig. 2.1(a)), we have the network shown in Fig. 2.3. To reduce the number of the multiplexers, we use the following rules:

1) If a sub-function is a constant 0 or 1, terminate that branch.
2) If two sub-functions are the same, i.e., if $f_0 = f_1$, then extend the branch down to the next level (i.e., do not use a multiplexer).
3) If the current sub-function (f_i) is the same as one already generated (f_j), move the current branch (for f_i) over to the other branch (f_j). For example, if f_{01} is the same as f_{10}, move the f_{10} branch over, thus merging the two nodes.

A decision diagram simplified by the above rule is called a (reduced ordered) binary decision diagram (BDD). For a given function and a given order of the input variables, the BDD is unique.

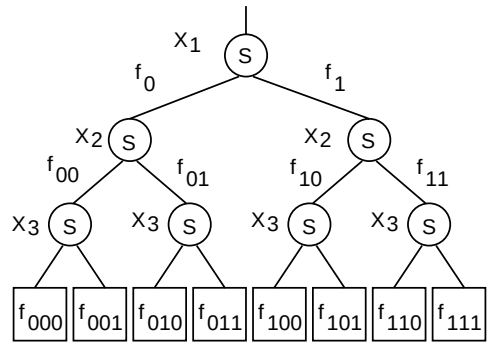

Figure 2.2: Representation of logic functions using the Shannon expansion

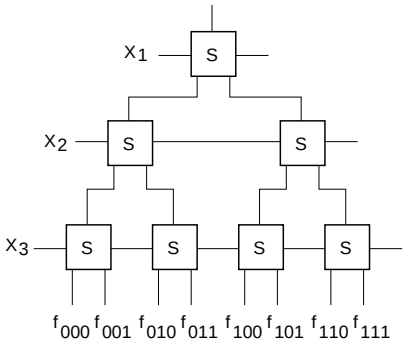

Figure 2.3: Realization of a logic function using multiplexers

If we use the positive Davio expansion instead of the Shannon expansion, then we have (2.2). When we use the same expansion for f_0 and f_2, we have

$$f_0 = f_{00} \oplus x_2 f_{02}, \text{ and}$$
$$f_2 = f_{20} \oplus x_2 f_{22}.$$

Fig. 2.4 is the tree corresponding to this expansion. Also, in this case, we can reduce the diagram. Such a diagram is called the functional decision diagram (FDD) [11]. For a given function and a given order of the input variables, the FDD is unique. In a similar way, we can consider the expansion using only the negative Davio expansion. Also, we can consider an expansion using either the positive or negative Davio expansions for each variable [13].

Next, consider the expansion where any one of the three expansions is permitted for each variable . For example, in Fig. 2.5, the Shannon expansion is used for x_1, the positive Davio expansion is used for x_2, and the negative Davio expansion is used for x_3. Such an expansion is called a Kronecker expansion [6, 28]. The reduced diagram is the Kronecker decision diagram (KDD) [18]. For a given function and a given order of the input variables, there are at most 3^n different KDD's. A minimum KDD is one with the fewest nodes.

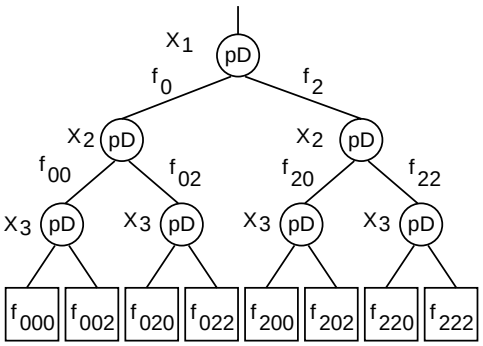

Figure 2.4: Representation of a logic function using the positive Davio expansion

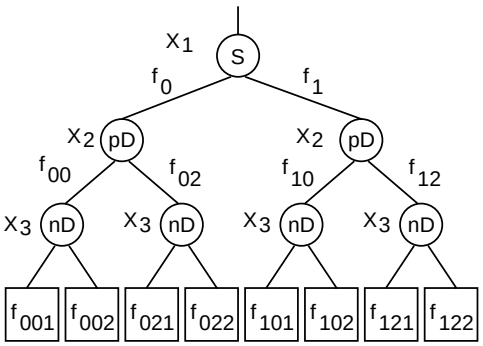

Figure 2.5: Representation of a logic function using the Kronecker expansion

Furthermore, we can consider the following expansion: Any one of the three expansions is permitted for any node. For example, in Fig. 2.6, the Shannon expansion is used for x_1, the positive and negative Davio expansions are used for x_2, and all the three expansions are used for x_3. Such an expansion is called a pseudo-Kronecker expansion [24, 28], and the reduced diagram is called a pseudo-Kronecker decision diagram (PKDD). For a given function f and the given order of the input variables, the PKDD with the minimum number of nodes (LUT's) is the minimum PKDD for f. There are at most $3^{(2^n-1)}$ different PKDD's.

The relation among BDD, FDD, KDD and PKDD is shown in Fig. 2.7. This shows that KDD's are a special case of PKDD's, and that BDD's and FDD's are each a special case of KDD's. It follows that when we realize a given function by 3-input LUT's, a method based on a PKDD requires the fewest LUT's. Fig. 2.8 is a realization of the function in Table 2.1 by 3-input LUT's. In this case, we used only the Shannon expansion for each LUT. However, in general, we can use any one of the three expansions to reduce the number of LUT's.

3 Definitions and Basic Properties

We assume that functions to be realized have $n = 2r$ inputs. Let $X = (x_1, x_2, \cdots, x_n)$ be the input variables,

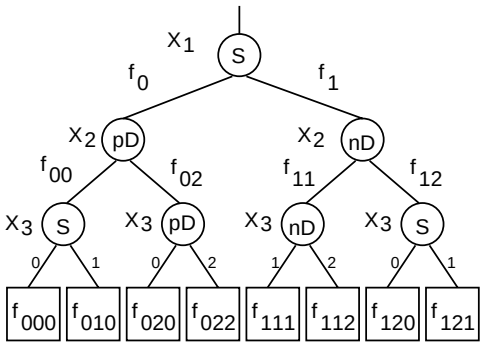

Figure 2.6: Representation of a logic function using the Pseudo-Kronecker expansion

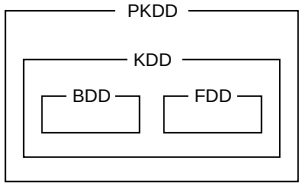

Figure 2.7: Relation among BDD's, FDD's, KDD's and PKDD's

where $x_i (i = 1, 2, .., n)$ takes on two values.

Definition 3.1 *Let $Q = \{0, 1, 2, 3\}$ and $B = \{0, 1\}$. $f: Q^r \rightarrow B$ is a* **four-valued input two-valued output function**.

An arbitrary logic function of n variables $f(X)$ can be converted into a four-valued input two-valued output function as follows: $f(X_1, X_2, \cdots, X_r) = f(X)$, where $X_i = (x_{2i-1}, x_{2i})$ takes either 0,1,2, or 3 iff $(x_{2i-1}, x_{2i}) = (0,0)$, $(0,1)$, $(1,0)$, or $(1,1)$, respectively.

Example 3.1 *The logic function shown in Table 2.1 can be converted into the four-valued input two-valued output function in Table 2.2, where $X_1 = (x_1, x_2)$ and $X_2 = (x_3, x_4)$.*

From now on, a four-valued input two-valued output function will be simply called a function, and an ordinary two-valued input function will be called a two-valued input function.

Definition 3.2 *Let $S \subseteq Q$. X^S is a* **literal** *of X, where*

$$X^S = \begin{cases} 0 & \text{if } X \notin S \\ 1 & \text{if } X \in S. \end{cases}$$

When S contains only one element, $X^{\{i\}}$ is denoted by X^i. A product of literals $X_1^{S_1} X_2^{S_2} \cdots X_n^{S_n}$ is **a product term** *that is the AND of the literals $X_1^{S_1}, X_2^{S_2}, \cdots,$ and $X_n^{S_n}$. A sum of products*

$$\bigvee_{(S_1, S_2, \cdots, S_n)} X_1^{S_1} X_2^{S_2} \cdots X_n^{S_n}$$

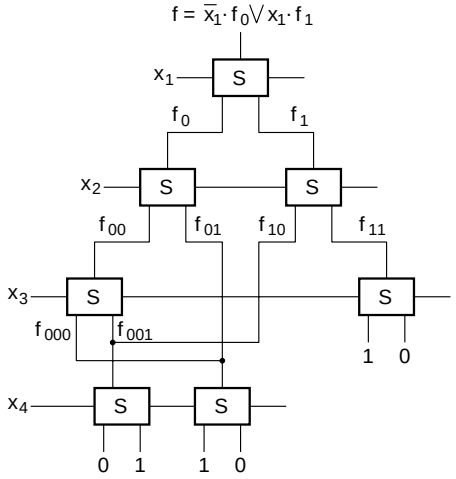

Figure 2.8: Realization of the function in Table 2.1

Table 2.1:

x_1	x_2	x_3	x_4	f
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	0
1	1	1	1	0

Table 2.2:

X_1	X_2	f
0	0	1
0	1	0
0	2	0
0	3	1
1	0	1
1	1	0
1	2	1
1	3	0
2	0	0
2	1	1
2	2	0
2	3	1
3	0	1
3	1	1
3	2	0
3	3	0

is a **sum-of-products expression** *(SOP), where* $\vee_{(S_1,S_2,\cdots,S_n)}$ *denotes the inclusive-OR of product terms. If the inclusive-OR is replaced by exclusive-OR, the result is an exclusive-OR of products*

$$\sum_{(S_1,S_2,\cdots,S_n)} \oplus \quad X_1^{S_1} X_2^{S_2} \cdots X_n^{S_n},$$

which is called an **exclusive-OR sum-of-products expression** *(ESOP).*

Lemma 3.1 *An arbitrary r-variable function can be expanded as*

$$f(X_1, X_2, \cdots, X_r) = X_1^0 f(0, X_2, \cdots, X_r) \vee X_1^1 f(1, X_2, \cdots, X_r) \vee X_1^2 f(2, X_2, \cdots, X_r) \vee X_1^3 f(3, X_2, \cdots X_r). \quad (3.1)$$

This is the Shannon expansion with respect to X_1.

Lemma 3.2 *f is uniquely represented as*

$$f = \bigvee_{(a_1,a_2,\cdots,a_n)} f(a_1, a_2, \cdots, a_n) X_1^{a_1} X_2^{a_2} \cdots X_n^{a_n},$$

where $\vee_{(a_1,a_2,\cdots,a_n)}$ *is the inclusive-OR for all the combinations such that* $a_i \in Q$, *and* $f(a_1, a_2, \cdots, a_n) = 0$ *or =1. f can be also uniquely represented by an expression:*

$$f = \sum_{(a_1,a_2,\cdots,a_n)} \oplus \quad f(a_1, a_2, \cdots, a_n) X_1^{a_1} X_2^{a_2} \cdots X_n^{a_n},$$

where $\sum \oplus_{(a_1,a_2,\cdots,a_n)}$ *represents the exclusive-OR for all the combinations of* $a_i \in Q$.

4 Design of FPGA's Using the Shannon Expansion

First, we will show a naive method to design LUT networks, where each LUT has six inputs. We can realize a function f by using the Shannon expansion,

$$f(X_1, X_2, X_3, X_4) = X_1^0 f_0 \vee X_1^1 f_1 \vee X_1^2 f_2 \vee X_1^3 f_3. \quad (4.1)$$

Fig. 4.1 shows a realization of the expansion in (4.1). It consists of one LUT that drives the output (called the output LUT) and four other LUT's, each realizing f_i. It is convenient to view the two binary control inputs as a single four-valued input. In this example, the control input of the output LUT is driven by X_1, while X_2 drives the control inputs of the other four LUT's. Fig. 4.1 shows that some combination of X_3 and X_4 drives the primary inputs of these four LUT's.

The following observations will reduce the number of LUT's:

1. If $f_i (i = 0, 1, 2,$ or 3) is a constant function, then the corresponding LUT's are not needed. That is, no special hardware is necessary for the constants.
2. If $f_i = f_j (i \neq j)$, then one LUT can be used for both f_i and f_j.

The diagram corresponding to the LUT network is called a QDD (Quarternary Decision Diagram). We can design an LUT network using an algorithm that is similar to that used for Binary Decision Diagrams (BDD's): If two sub-functions are the same, realize only one sub-function. If a sub-function involves 6 or fewer variables, stop (it is realized by a single LUT).

Example 4.1 *Let f be the following function:*

$$f(X_1, X_2, X_3, X_4) = X_1^0 (X_2^3 \oplus X_3^3) \vee X_1^1 (X_3^3 \oplus X_4^3) \vee X_1^2 (X_2^3 \oplus X_4^3) \vee X_1^3 X_4^3,$$

where $\oplus$ *is the modulo 2 sum.*

If f is represented in the form (4.1), we have

$$f_0 = X_2^3 \oplus X_3^3,\ f_1 = X_3^3 \oplus X_4^3,\ f_2 = X_2^3 \oplus X_4^3,\ f_3 = X_4^3.$$

In this case, all the sub-functions are non-constant and distinct. Fig. 4.1 shows a realization for f.

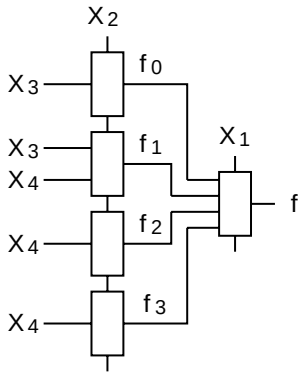

Figure 4.1: Realization of a logic function by the Shannon expansion

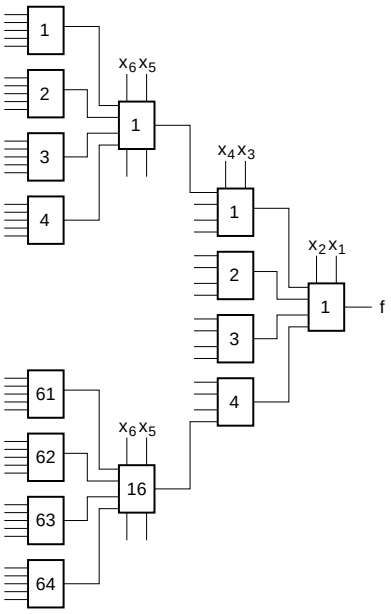

Figure 4.2: Network structure for $k = 6$

5 Enumeration of LUT's to Realize Given Functions.

Here, we consider the question of how many LUT's are required in the realization of a given function.

Theorem 5.1 *Let f be an arbitrary r-variable four-valued input two-valued output function. When $r \leq 3$, f can be realized by a single LUT with six two-valued inputs. When $r \geq 4$, f can be realized with at most $\frac{4^{r-2}-1}{3}$ LUT's.*

(Proof) Because an LUT realizes any function with 3 inputs, a single LUT is sufficient. Any r-variable function, where $r \geq 3$ can be realized by an LUT tree structure of the type shown in Fig. 4.2, where one four-valued variable is applied at each level, except the leftmost level, where three variables are applied. The number of LUT's required is at most

$$1 + 4^1 + 4^2 + \cdots + 4^{r-3} = \frac{4^{r-2}-1}{3}.$$

Q.E.D.

The above theorem gives an upper bound. However, in many cases, given functions can be realized with fewer LUT's.

Theorem 5.2 *Given a function $f(X_1, X_2, \cdots, X_r)$, let μ be the number of distinct functions of the form $f(a_1, a_2, \cdots, a_{r-3}, X_{r-2}, X_{r-1}, X_r)$, where $a_i \in Q$, $i = 1, 2, \cdots, r-3$. Then, f can be realized with at most*

$$\frac{4^{r-3}-1}{3} + \mu \text{ LUT's.}$$

(Proof) f can be expanded as

$$f(X_1, X_2, \cdots, X_r) =$$
$$\vee X_1^{a_1} X_2^{a_2} \cdots X_{r-3}^{a_{r-3}} f(a_1, a_2, \cdots, a_{r-3}, X_{r-2}, X_{r-1}, X_r),$$
$$\text{where } a_i \in Q,\ i = 1, 2, \cdots, r-3.$$

We can realize a selector circuit that selects one out of 4^{r-3} inputs by using $\frac{4^{r-3}-1}{3}$ LUT's. Because there are μ distinct functions of the form $f(a_1, a_2, \cdots, a_{r-3}, X_{r-2}, X_{r-1}, X_r)$, we need to realize only μ such functions. Q.E.D.

6 A Design Method Using the Pseudo-Kronecker Expansion.

The design method using the Shannon expansion is simple, but requires many LUT's. This is because each LUT is only used as a multiplexer, while it can realize any of 2^{64} different functions. To reduce the number of LUT's, we can use the theory of pseudo-Kronecker expansions, which was developed for the optimization of ESOPs with four-valued inputs [23, 24].

Definition 6.1 *Let S be a set of logic values, where $S \subseteq Q$. The* **characteristic vector** *of S is $\boldsymbol{a} = (\alpha_0, \alpha_1, \alpha_2, \alpha_3)$, where α_j is a logic value such that*

$$\alpha_i = \begin{cases} 0 & \text{if } i \notin S \\ 1 & \text{if } i \in S. \end{cases}$$

Definition 6.2 *A matrix M*

$$M = \begin{bmatrix} \boldsymbol{a}_0 \\ \boldsymbol{a}_1 \\ \boldsymbol{a}_2 \\ \boldsymbol{a}_3 \end{bmatrix}$$

of characteristic vectors $\boldsymbol{a}_i$ is **non-singular** *if there exists a unique matrix M^{-1} such that*

$$MM^{-1} = \begin{bmatrix} 1000 \\ 0100 \\ 0010 \\ 0001 \end{bmatrix} = I,$$

where multiplication is AND and addition is exclusive-OR.

If one expands a given function f: $Q^r \rightarrow B$ about variable X using the Shannon decomposition,

$$f = X^0 f_0 \oplus X^1 f_1 \oplus X^2 f_2 \oplus X^3 f_3,$$

then f_0, f_1, f_2, and f_3 are unique. The next theorem [23] shows that for certain other forms of X^S, the sub-functions are also unique. The same theorem can be also found in [19] and [24].

Theorem 6.1 *(Expansion Theorem) An arbitrary function $f : Q^r \rightarrow B$ can be expanded with respect to a variable X as*

$$f = X^{S_0} h_0 \oplus X^{S_1} h_1 \oplus X^{S_2} h_2 \oplus X^{S_3} h_3, \qquad (6.1)$$

where h_0, h_1, h_2, and h_3 are unique if and only if M is non-singular, where

$$M = \begin{bmatrix} \boldsymbol{a}_0 \\ \boldsymbol{a}_1 \\ \boldsymbol{a}_2 \\ \boldsymbol{a}_3 \end{bmatrix}, \text{ and } \boldsymbol{a}_i \ (i = 0, 1, 2, 3) \text{ are the characteristic vectors of } S_i. \qquad (6.2)$$

(Proof) By (6.1), we have

$$f = X^{S_0} h_0 \oplus X^{S_1} h_1 \oplus X^{S_2} h_2 \oplus X^{S_3} h_3.$$

On the other hand, by the Shannon expansion of f, we have

$$f = X^0 f_0 \oplus X^1 f_1 \oplus X^2 f_2 \oplus X^3 f_3.$$

Let I be the unit 4×4 matrix. Since the two expansions above represent the same function,

$$[h_0, h_1, h_2, h_3] M = [f_0, f_1, f_2, f_3] I.$$

(if) When M is non-singular, the inverse matrix M^{-1} exists, and the function can be uniquely represented with

$$[h_0, h_1, h_2, h_3] = [f_0, f_1, f_2, f_3] M^{-1}.$$

(only if) When M is singular, $[h_0, h_1, h_2, h_3]$ cannot be uniquely represented. Q.E.D.

Corollary 6.1 *Let A, B, C, D, A', B', C' and D' be subsets of $Q = \{0, 1, 2, 3\}$, and let $\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{c}$, $\boldsymbol{d}$, $\boldsymbol{a}'$, $\boldsymbol{b}'$, $\boldsymbol{c}'$ and $\boldsymbol{d}'$, respectively, be the characteristic vectors for them.*

$$\text{Let } M = \begin{bmatrix} \boldsymbol{a} \\ \boldsymbol{b} \\ \boldsymbol{c} \\ \boldsymbol{d} \end{bmatrix},$$

$R = [\boldsymbol{a}'^t, \boldsymbol{b}'^t, \boldsymbol{c}'^t, \boldsymbol{d}'^t]$, and $MR = I$ (unit matrix). Then, an arbitrary four-valued input two-valued output function f can be uniquely represented in the forms

$$\begin{aligned} f &= X^{\{0\}} f_{\{0\}} \oplus X^{\{1\}} f_{\{1\}} \oplus X^{\{2\}} f_{\{2\}} \oplus X^{\{3\}} f_{\{3\}} \text{ or} \\ &= X^A f_{A'} \oplus X^B f_{B'} \oplus X^C f_{C'} \oplus X^D f_{D'}, \text{ where} \\ f_{A'} &= \sum_{i \in A'} \oplus f_{\{i\}}, \ f_{B'} = \sum_{i \in B'} \oplus f_{\{i\}}, \ f_{C'} \\ &= \sum_{i \in C'} \oplus f_{\{i\}}, \ f_{D'} = \sum_{i \in D'} \oplus f_{\{i\}}. \end{aligned}$$

Example 6.1 *Consider the four-valued input two-valued output function f in Table 2.2. By the Shannon expansion, it can be represented as*

$$f = X^{\{0\}} f_{\{0\}} \oplus X^{\{1\}} f_{\{1\}} \oplus X^{\{2\}} f_{\{2\}} \oplus X^{\{3\}} f_{\{3\}}.$$

Let us consider the expansion of the form $f = X^A f_{A'} \oplus X^B f_{B'} \oplus X^C f_{C'} \oplus X^D f_{D'}$. In this case, the sub-functions obtained by EXORing sub-functions are as follows:

$$\begin{aligned} f_{\{0\}} &= X_2^{\{0,3\}}, \ f_{\{1\}} = X_2^{\{0,2\}}, \\ f_{\{2\}} &= X_2^{\{1,3\}}, \ f_{\{3\}} = X_2^{\{0,1\}}, \\ f_{\{0,1\}} &= X_2^{\{2,3\}}, \ f_{\{0,2\}} = X_2^{\{0,1\}}, \ f_{\{0,3\}} = X_2^{\{1,3\}}, \\ f_{\{1,2\}} &= X_2^{\{0,1,2,3\}}, \ f_{\{1,3\}} = X_2^{\{1,2\}}, \ f_{\{2,3\}} = X_2^{\{0,3\}}, \\ f_{\{1,2,3\}} &= X_2^{\{2,3\}}, \ f_{\{0,2,3\}} = 0, \\ f_{\{0,1,3\}} &= X_2^{\{0,1,2,3\}}, \ f_{\{0,1,2\}} = X_2^{\{1,2\}}, \\ f_{\{0,1,2,3\}} &= X_2^{\{0,2\}}. \end{aligned}$$

The number of products needed to represent the above sub-functions are 0 for $f_{\{023\}}$, and 1 for the others. The characteristic vectors for the four sub-functions $f_{\{023\}}$, $f_{\{1\}}$, $f_{\{2\}}$ and $f_{\{3\}}$ are (1011), (0100), (0010), and (0001), respectively. Note that these vectors are linearly independent of each other. Hence, f can be uniquely represented by these four sub-functions. Because

$$R = \begin{bmatrix} 1000 \\ 0100 \\ 1010 \\ 1001 \end{bmatrix} \text{ and } M = R^{-1} = \begin{bmatrix} 1000 \\ 0100 \\ 1010 \\ 1001 \end{bmatrix},$$

f can be represented as

$$\begin{aligned} f &= X_1^A f_{A'} \oplus X_1^B f_{B'} \oplus X_1^C f_{C'} \oplus X_1^D f_{D'}, \\ &= X_1^{\{0\}} f_{\{0,2,3\}} \oplus X_1^{\{1\}} f_{\{1\}} \oplus X_1^{\{0,2\}} f_{\{2\}} \oplus X_1^{\{0,3\}} f_{\{3\}}, \\ &= X_1^{\{1\}} X_2^{\{0,3\}} \oplus X_1^{\{0,2\}} X_2^{\{1,3\}} \oplus X_1^{\{0,3\}} X_2^{\{0,1\}}. \end{aligned}$$

Note that this expansion has only three product terms. *(End of Example)*

A computer enumeration shows that, out of the $(2^4)^4 = 2^{16}$ possible 4×4 matrices, 20160 are non-singular [23]. If we ignore the labeling of logic values for one of the variables, there are essentially $\frac{20160}{4!} = 840$ different expansions, of which the Shannon expansion is one. This leads to the question of which expansion reduces the complexity of the network. We can select an expansion that reduces the complexity of the network.

Definition 6.3 Pseudo-Kronecker decision diagrams *(PKDD's) of n-variable four-valued input two-valued output functions are defined recursively as follows:*

1) *A PKDD is a terminal node v labeled by value(v), where value(v) $\in \{0,1\}$.*
2) *A PKDD is a non-terminal node v that has four children, $h_0(v)$, $h_1(v)$, $h_2(v)$, and $h_3(v)$, that are also PKDD's, where v is labeled by*
(a) index(v), where index(v) $\in \{1,2,\cdots,r\}$, and by
(b) a 4×4 regular matrix M, where M is defined in (6.2).

The correspondence between a PKDD and a Boolean function is defined as follows:

A PKDD G having a root node v denotes a function f_v defined recursively as:

3) *If v is a terminal node:*
(a) If value(v) $= 1$, then $f_v = 1$.
(b) If value(v) $= 0$, then $f_v = 0$.
4) *If v is a nonterminal node with index(v) $= i$, then f_v is a function*
$X_i^{S_0} f_{h_0(v)} \oplus X_i^{S_1} f_{h_1(v)} \oplus X_i^{S_2} f_{h_2(v)} \oplus X_i^{S_3} f_{h_3(v)}$,

such that

$$M = \begin{bmatrix} \boldsymbol{a}_0 \\ \boldsymbol{a}_1 \\ \boldsymbol{a}_2 \\ \boldsymbol{a}_3 \end{bmatrix} \text{ is non-singular,}$$

where $\boldsymbol{a}_i$ is a characteristic vector for S_i $(i = 0,1,2,3)$. X_i is called the **decision variable** *for node v.*

Definition 6.4 *A* **reduced ordered PKDD** *is a PKDD such that:*

1) *In Definition 6.3, for any non-terminal node v, if $h_i(v)$ is also nonterminal, then index(v) <index($h_i(v)$).*
2) *No two subgraphs in the PKDD are identical.*

The PKDD for f is said to be minimum if it contains the least number of nodes.

It is very difficult to obtain an exact minimum PKDD, so we will consider a heuristic method which obtains a good solution quickly. In this case, we have to consider the following 15 sub-functions:

$$\begin{aligned}
&f_{\{0\}},\ f_{\{1\}},\ f_{\{2\}},\ f_{\{3\}},\\
&f_{\{01\}} = f_{\{0\}} \oplus f_{\{1\}},\ f_{\{02\}} = f_{\{0\}} \oplus f_{\{2\}},\ f_{\{03\}}\\
&\quad = f_{\{0\}} \oplus f_{\{3\}},\\
&f_{\{12\}} = f_{\{1\}} \oplus f_{\{2\}},\ f_{\{13\}} = f_{\{1\}} \oplus f_{\{3\}}\ , f_{\{23\}}\\
&\quad = f_{\{2\}} \oplus f_{\{3\}},\\
&f_{\{012\}} = f_{\{0\}} \oplus f_{\{1\}} \oplus f_{\{2\}},\ f_{\{013\}}\\
&\quad = f_{\{0\}} \oplus f_{\{1\}} \oplus f_{\{3\}},\\
&f_{\{023\}} = f_{\{0\}} \oplus f_{\{2\}} \oplus f_{\{3\}},\ f_{\{123\}}\\
&\quad = f_{\{1\}} \oplus f_{\{2\}} \oplus f_{\{3\}},\\
&f_{\{0123\}} = f_{\{0\}} \oplus f_{\{1\}} \oplus f_{\{2\}} \oplus f_{\{3\}}.
\end{aligned}$$

Algorithm 6.1 *(Generation of a PKDD : Simple method)*

0) *For $i = 1$ to $r-3$ do the following:*
1) *Let f be the function. Expand it in the form*
$$f = X_i^{S_0} h_0 \oplus X_i^{S_1} h_1 \oplus X_i^{S_2} h_2 \oplus X_i^{S_3} h_3$$
that minimizes the cost of the expansion, where
$$COST(f) = \sum_{j=0}^{3} COST(h_j),\ \text{and}$$
$$COST(h_j) = \begin{cases} 0 \text{ if } h_j \text{ is constant} \\ 0 \text{ if } h_j \text{ is already realized} \\ k \text{ if } h_j \text{ is not realized previously} \\ \quad \text{and depends on } k \text{ variables} \end{cases}.$$
2) *Create the nodes for new functions if they do not exist. Record that they are realized.*
3) *Expand the remaining nodes in the same way.*

Example 6.2 *Consider the 4-valued input function f in Example 3.1.*

$$\begin{aligned}
f(X_1,X_2,X_3,X_4) = {} & X_1^{\{0\}}(X_2^3 \oplus X_3^3) \vee X_1^{\{1\}}(X_3^3 \oplus X_4^3)\\
& \vee X_1^{\{2\}}(X_2^3 \oplus X_4^3) \vee X_1^{\{3\}} X_4^3.
\end{aligned}$$

This expansion has four non-zero sub-functions. However, we can reduce the number of non-zero sub-functions by considering an expansion with the form

$$f = X_1^A f_{A'} \oplus X_1^B f_{B'} \oplus X_1^C f_{C'} \oplus X_1^D f_{D'}.$$

To find a good expansion, we calculate the costs for all the 15 sub-functions.

$$\begin{aligned}
&f_{\{0\}} = f_{\{1,2\}} = X_2^3 \oplus X_3^3 && , cost = 2.\\
&f_{\{1\}} = f_{\{0,2\}} = X_3^3 \oplus X_4^3 && , cost = 2.\\
&f_{\{2\}} = f_{\{0,1\}} = X_2^3 \oplus X_4^3 && , cost = 2.\\
&f_{\{3\}} = f_{\{0,1,2,3\}} = X_4^3 && , cost = 1.\\
&f_{\{0,3\}} = f_{\{1,2,3\}} = X_2^3 \oplus X_3^3 \oplus X_4^3 && , cost = 3.\\
&f_{\{1,3\}} = f_{\{0,2,3\}} = X_3^3 && , cost = 1.\\
&f_{\{2,3\}} = f_{\{0,1,3\}} = X_2^3 && , cost = 1.\\
&f_{\{0,1,2\}} = 0 && , cost = 0.
\end{aligned}$$

Note that $f_{A'}$, $f_{B'}$, $f_{C'}$, and $f_{D'}$ have smaller costs, where $A' = \{2,3\}$, $B' = \{1,3\}$, $C' = \{3\}$, and $D' = \{0,1,2\}$. The characteristic vectors for A', B', C', and D' are: (0,0,1,1), (0,1,0,1), (0,0,0,1), and (1,1,1,0), and they are linearly independent. Thus, we have R and M as follows:

$$R = \begin{bmatrix} 0001 \\ 0101 \\ 1001 \\ 1110 \end{bmatrix} \text{ and } M = R^{-1} = \begin{bmatrix} 1010 \\ 1100 \\ 0111 \\ 1000 \end{bmatrix}.$$

Because $A = \{0,2\}$, $B = \{0,1\}$, $C = \{1,2,3\}$, and $D = \{0\}$, f can be represented as

$$\begin{aligned}
f &= X_1^A f_{A'} \oplus X_1^B f_{B'} \oplus X_1^C f_{C'} \oplus X_1^D f_{D'},\\
&= X_1^{\{0,2\}} f_{\{2,3\}} \oplus X_1^{\{0,1\}} f_{\{1,3\}} \oplus X_1^{\{1,2,3\}} f_{\{3\}}\\
&\quad \oplus X_1^{\{0\}} f_{\{0,1,2\}},\\
&= X_1^{\{0,2\}} X_2^{\{3\}} \oplus X_1^{\{0,1\}} X_3^{\{3\}} \oplus X_1^{\{1,2,3\}} X_4^{\{3\}}.
\end{aligned}$$

Note that the last expansion contains only three non-zero subfunctions. (End of Example)

7 Experimental Results [29]

7.1 For 3-input LUT's (2-valued case)

Table 7.1 compares the number of non-terminal nodes for BDD's, FDD's, KDD's, and PKDD's. For FDD's, the column headed with "pD" denotes the case where only positive Davio expansions are used, and "both" denotes the case where both positive and negative Davio expansions are used. For each diagram, we obtained an ordering of input variables that reduce the number of nodes. The orderings of the input variables are obtained by a simulated annealing method similar to [8]. Note that the orderings that minimize the size of BDD's do not always minimize the size of FDD's, KDD's or PKDD's.

Let size(BDD), size(FDD), size(KDD) and size(PKDD) be the numbers of non-terminal nodes for optimized BDD, FDD, KDD and PKDD for a function, respectively. The experimental results show that

$$\text{size(PKDD)} \leq \text{size(KDD)} \leq \text{size(BDD)},$$
$$\text{size(KDD)} \leq \text{size(FDD:both)} \leq \text{size(FDD:pD)},$$
$$\text{size(FDD:both)} \leq \text{size(FDD:nD)}.$$

These results are consistent with the relation of decision diagrams in Fig. 2.7. The bottom row of the Table 7.1 shows the relative sizes of the diagrams. On the average, PKDD's require 29% fewer nodes than BDD's. However, the nodes for BDD's and FDD's are comparable.

7.2 For 6-input LUT's (4-valued case).

The bit pairing algorithm [22] for PLA's with decoders was used to produce four-valued input two-valued output functions. Table 7.1 also compares the number of non-terminal nodes for QDDs (Quarternary Decision Diagrams) and 4-valued PKDD's. QDDs were obtained by the 4-valued extension of Shannon Expansion in (3.1). Details of the optimization algorithm is shown in Appendix. On the average, QDDs require 37% fewer nodes than BDD's, and 4-valued PKDD's require 51% fewer nodes than BDD's. Also, 4-valued PKDD's require 23% fewer nodes than QDDs.

8 Conclusions and Comments

In FPGA design, interconnections are often more expensive than logic. FPGA's using 3-input LUT's require many logical levels and complex interconnections. On the other hand, FPGA's using 6-input LUT's require fewer interconnections and fewer logical levels.

In this paper, we showed a method to represent logic functions by using pseudo-Kronecker diagrams (PKDD's). Experimental results show that 2-valued PKDD's require 29% fewer nodes than BDD's, and 4-valued PKDD's require 23% fewer than QDDs, the 4-valued extension of BDD's. Thus, this method is useful for the design of FPGA's with 6-input LUT's. However, when LUT's have less than 6 inputs, this method is not applicable.

In the practical design of FPGA's, we can further reduce the number of LUT's as follows:

1) Use complement edges [15, 7].
   Consider the decision diagrams where each edge can complement the sub-function. This corresponds the fact that LUT's can complement the inputs with no extra cost. In the case of 2-valued BDD's, complement edges can reduce the number of nodes by 30 to 50% [15].
2) Remove redundant LUT's.
   For a 3-input LUT, if both of two sub-functions are constants, then it realizes a variable or its complement. Thus, this LUT can be removed, and the variable is used as the output. If an LUT depends on at most three variables, then it can be realized with one LUT. For a 6-input LUT, if it depends on at most six variables, then it can be realized with one LUT.
3) Merge several LUT's into one.
   For 6-input LUT's, if any of the sub-functions is a constant, then the function is realize by an LUT with 5 or fewer inputs. In AT&T ORCA series FPGA's, an LUT can be configured to realize either a 6-input function, or a pair of 5-input functions, or four 4-input functions. Thus, LUT's with fewer inputs can be merged. Except for the 6-input case, some of the inputs must be shared among the functions.

We assumed that the FPGA's are all binary, but the control inputs for LUT's can be four-valued inputs instead of pairs of binary variables. This will significantly reduce the connections. Thus, this method is also useful for a design of multiple-valued LUT type FPGA's.

[9] shows a method to realize a logic function by using multiple-valued multiplexers, where only circuits with tree type structure are generated. Also, interconnections are more complex, since each level may have different control variables.

Acknowledgments

This work was supported in part by a Grant in Aid for Scientific Research of the Ministry of Education, Science and Culture of Japan, and the Naval Research Laboratory, Washington, DC through direct funds at the Naval Postgraduate School, Monterey, CA U.S.A. M. Matsuura, K. Hara, and Y. Yamashita developed software and did experiment.

Appendix

Algorithm 8.1 *(4-valued PKDD Simplification)*

1. *Simplify QDDs by permuting the input variables (Algorithm 8.2).*
2. *Simplify 4-valued KDD's by changing the method of expansion for each variable (Algorithm 8.3).*
3. *Simplify 4-valued PKDD's by changing the method of expansion for each node (Algorithm 8.4).*

Algorithm 8.2 *(Optimization of input ordering for QDDs).*

Table 7.1: Number of nodes in Decision Diagrams

function	in	out	BDD	FDD		KDD	PKDD	4-valued	
				pD	both			QDD	PKDD
9sym	9	1	33	27	27	27	26	17	17
add6	12	7	47	23	23	23	23	17	12
adr4	8	5	29	15	15	15	15	11	8
alu2	10	6	61	94	77	60	59	48	38
apla	10	12	103	188	124	103	94	78	76
bw	5	28	108	116	106	103	90	71	56
clip	9	5	97	87	85	85	69	52	34
co14	14	1	27	26	26	26	26	13	13
con1	7	2	15	17	17	15	14	11	11
dc2	8	7	64	77	71	63	54	44	37
dist	8	5	152	167	163	152	127	75	71
dk17	10	11	63	98	71	63	62	47	45
dk27	8	9	25	24	23	23	22	25	25
duke2	22	29	373	367	351	332	269	257	221
f51m	8	8	67	38	35	35	28	46	22
inc	7	9	70	78	75	65	58	41	37
misex1	8	7	36	42	39	35	31	27	24
misj	35	14	43	46	44	43	43	26	26
mlp4	8	8	141	108	108	108	99	81	57
radd	8	5	29	15	15	15	15	11	8
rd53	5	3	23	13	13	13	13	11	7
rd73	7	3	43	21	21	21	21	21	11
rd84	8	4	59	29	29	29	29	30	15
risc	8	31	67	62	59	59	56	57	51
rot8	8	5	75	89	83	74	60	41	36
sao2	10	4	85	98	97	84	69	50	36
sex	9	14	47	46	45	41	41	38	35
sqr8	8	16	233	201	201	201	169	147	109
t481	16	1	32	23	17	17	17	15	10
tial	14	8	690	543	494	456	381	392	212
ts10	22	16	146	156	156	132	132	148	148
xor5	5	1	9	5	5	5	5	5	3
z5xp1	7	10	68	53	41	41	33	40	24
total			3160	2992	2756	2564	2250	1993	1535
ratio			1.00	0.95	0.87	0.81	0.71	0.63	0.49

1. *Let X_1, X_2, $\cdots$, X_r be the original ordering for the QDD.*
2. *For $i = 1$ to $r - 1$*
   {For $j = 2$ to r
   exchange X_i with X_j, and count the number of nodes in QDD.
   Adopt the ones that minimize the number of nodes }
3. *Repeat 2. while the number of nodes is reduced.*

Algorithm 8.3 *(Simplification of 4-valued KDD's).*

1. *Let X_1, X_2, $\cdots$, X_r be the ordering obtained by Algorithm 8.2.*
2. *Construct a PDD (Pentadecimal Decision Diagram) that has 15 children for each non-terminal node, where each child corresponds to one of 15 sub-functions.*
3. *Let QDD be the initial 4-valued KDD.*
4. *For $i = 1$ to $r - 1$*
   Expand the sub-functions with respect X_i in 840 different ways, and find one that minimizes the number of nodes.
5. *Repeat 4. while the number of nodes is reduced.*

Algorithm 8.4 *(Simplification of 4-valued PKDD's).*

1. *Let X_1, X_2, $\cdots$, X_r be the ordering. Let the 4-valued KDD obtained by Algorithm 8.3 be the initial 4-valued PKDD.*
2. *For $i = 1$ to $r - 1$*
   For each node for X_i, expand the sub-functions in 840 different ways, and find ones that minimizes the number of nodes.
3. *Repeat 2. while the number of nodes is reduced.*

References

[1] AT&T, AT&T ORCA (Optimized Reconfigurable Cell Array) Series FPGA, Product Brief, April. 1993.

[2] H. Bouzouzou, G. Saucier, F. Pirot, and R. Roane, "Use of binary decision diagrams for logic design", *International Workshop on Logic Synthesis*, May 1993.

[3] S. D. Brown, R. J. Francis, J. Rose, and Z. G. Vranesic, *Field Programmable Gate Arrays*, Kluwer Academic Publishers, Boston 1992.

[4] R. E. Bryant, "Graph-based algorithms for Boolean function manipulation," *IEEE Trans. Comput.* Vol. C-35, No. 8, Aug. 1986, pp. 677-691.

[5] J. Cong and Y. Ding, "An optimal technology mapping algorithm for delay optimization in look-up table based FPGA designs", *Proc. ICCAD*, pp. 48-53, 1992.

[6] M. Davio, J-P. Deschamps, and A. Thayse, *Discrete and Switching Functions*, McGraw-Hill International, 1978.

[7] R. Drechsler, A. Sarabi, M. Theobald, B. Becker and M.A. Perkowski, "Efficient representation and manipulation of switching functions based on ordered Kronecker Functional Decision Diagrams", *Fachbereich Informatik*, Universitat Frankfurt, Interner Bericht 14/93.

[8] N. Ishiura, H. Sawada, and S. Yajima, "Minimization of binary decision diagrams based on exchange of variables", *ICCAD-91*, pp. 472-475, Nov. 1991.

[9] M. Kameyama and T. Higuchi, "Synthesis of multiple-valued logic networks based on tree-type universal logic module", *IEEE Trans. Comput.* Vol. C-26. No. 12, Dec. 1977.

[10] K. Karplus, "Xmap: a technology mapper for table-lookup field-programmable gate arrays", *DAC-91*, pp. 240-243, June 1991.

[11] U. Kebschull, E. Schubert and W. Rosenstiel, "Multilevel logic synthesis based on functional decision diagrams", *EDAC 92*, 1992, pp. 43-47.

[12] Y-T. Lai, M. Pedram and S. B. K. Vrudhula, "BDD based decomposition of logic functions with application to FPGA synthesis", *30th ACM/IEEE Design Automation Conference*, June 1993.

[13] L. McKenzie, L. Xu and A. Almaini, "Graphical representation of generalized Reed-Muller Expansions", *IFIP 10.5 Workshop on Applications of the Reed-Muller Expansion in Circuit Design*, Hamburg, Sept. 16-17, 1993.

[14] D. M. Miller,"Multiple-valued logic design tools", *ISMVL-93*, May 1993, pp. 2-11.

[15] S. Minato, N. Ishiura, and S. Yajima, "Shared binary decision diagram with attributed edges for efficient Boolean function manipulation", *27th Design Automation Conference*, pp. 52-57, June, 1990.

[16] R. Murgai, N. Shenoy, R. K. Brayton, and A. Sangiovanni-Vincentelli, "Performance directed synthesis for table look up programmable gate arrays", *ICCAD-91*, pp. 564-567, Nov. 1991.

[17] M.A. Perkowski, "The generalized orthnormal expansion of functions with multiple-valued inputs and some of its applications", *ISMVL-92*, Sendai pp. 442-450, May 1992.

[18] M. A. Perkowski, A. Sarabi, and F. R. Beyl, "XOR canonical forms of switching functions", *IFIP 10.5 Workshop on Applications of the Reed-Muller Expansion in Circuit Design*, Hamburg, Sept. 16-17, 1993.

[19] M. A. Perkowski, "A fundamental theorem for EXOR circuits", *IFIP WG 10.5 Workshop on Applications of the Reed-Muller Expansion in Circuit Design*, September 16-17, 1993.

[20] J. Rose, R. J. Francis, D. Lewis, and P. Chow, "Architecture of field-programmable gate arrays: the effect of logic block functionality on area efficiency", *IEEE Journal of Solid-State Circuits*, vol. 25, No.5. pp. 1217-1225, 1990.

[21] A. Sarabi, P. F. Ho, K. Iravanni, W. R. Daasch, and M. A. Perkowski, "Minimal multi-level realization of switching functions based on Kronecker functional decision diagrams", *International Workshop on Logic Synthesis*, Lake Tahoe, May 1993.

[22] T. Sasao, "Input variable assignment and output phase optimization of PLA's", *IEEE Trans. on Comput.* vol. C-33, No. 10, pp. 879-894, Oct. 1984.

[23] T. Sasao, "A transformation of multiple-valued input two-valued output functions and its application to simplification of exclusive-or sum-of-products expressions", *ISMVL-91*, May 1991, pp. 270-279.

[24] T. Sasao, "Optimization of multiple-valued AND-EXOR expressions using multiple-place decision diagrams", *ISMVL-92*, May 1992, pp. 451-458.

[25] T. Sasao and J. T. Butler, "On the Analysis of an FPGA Architecture", *International Symposium on Logic Synthesis and Microprocessor Architecture*, July 1992, pp. 162-168.

[26] T. Sasao, "Logic design of ULSI systems", *Post Binary Logic Workshop*, Sendai, Japan, May 1992.

[27] T. Sasao, "FPGA design by generalized functional decomposition", in (Sasao e.d.) *Logic Synthesis and Optimization*, Kluwer Academic Publishers, pp. 233-258, 1993.

[28] T. Sasao, "AND-EXOR expressions and their optimization", in (Sasao e.d.) *Logic Synthesis and Optimization*, Kluwer Academic Publishers, pp. 287-312, 1993.

[29] T. Sasao and M. Matsuura, (to be published)

[30] C. E. Shannon, "The synthesis of two-terminal switching circuits", *Bell Syst. Tech.* J. 28, 1, pp.59-98, 1949.

[31] K. C. Smith and P. G. Gulak, "Prospects for multiple-valued integrated circuits", *IEICE Trans. Electron.*, Vol. E.76-C, No.3, March 1993, pp.372-382.

[32] Wei Wan and M. Perkowski, "A new approach to the decomposition of incompletely specified multi-output functions based on graph coloring and local transformations and its application to FPGA mapping", *Euro DAC'92*.