# On the Cascade Implementation of Multiple-Output Sparse Logic Functions

*Václav Dvořák*
*Faculty of Information Technology*
Brno University of Technology
Brno, Czech Republic
dvorak@fit.vutbr.cz

*Petr Mikušek*
*Faculty of Information Technology, BUT*
and *KI-WI Digital, Ltd.*
Brno, Czech Republic
petr@mikusek.info

***Abstract*— Representation of multiple-output logic functions by Multi-Terminal Binary Decision Diagrams (MTBDDs) is studied for the useful class of sparse logic functions specified by the number of true min-terms. This paper derives upper bounds on the MTBDD width, which determine the size of look-up tables (LUTs) needed for hardware realization of these functions in FPGA logic synthesis. The obtained bounds are generalization of similar known bounds for single-output logic functions. Finally a procedure how to find the optimum mapping of MTBDD to a LUT cascade is presented and illustrated on a set of benchmarks.**

***Keywords- Boolean functions, multi-terminal binary decision diagrams MTBDDs, LUT cascades, area-time complexity***

## I. INTRODUCTION

The design of digital systems with look-up tables (LUTs) relies on the $K$-LUT basic module that implements an arbitrary function of $K$ variables. At present, LUTs with up to $K$=6 binary inputs and a single binary output are common components of FPGAs and in the future LUTs with $K$=7 or 8 may become available. LUTs with $M$ output bits ($K/M$-LUTs) can be implemented as a collection of $M$ single-output $K$-LUTs (at present if $K \leq 6$) or they can be embedded in RAM modules (if $K > 6$).

When designing digital systems on the basis of above modules, a synthesis problem can be stated as follows:

**Problem 1.1.** Given an $M$-output function $F_n$ of $n$ binary variables, find

1. the minimum number of $K$-LUTs needed to implement $F_n$, where $K = 6, 7, 8$.
2. the number of $K$-LUTs needed to implement $F_n$ with the shortest delay possible, where $K = 6, 7, 8$.
3. the number of $K$-LUTs needed for the best area-delay product (combination of two requirements above).

These optimization problems are very hard generally and we will restrict ourselves to the first and third problems and only to one configuration of $K/M$-LUTs, namely to the LUT cascade. The cascade of $K/M$-LUTs can be simulated by simple hardware or may provide support for reconfigurable architectures [1]; speed is competitive with other FPGA designs [2]. LUT cascades have been applied to many useful digital functions and their effectiveness and performance has been compared to benchmark circuits [1].

A direct synthesis of LUT cascades comes out easily from the known representation of multiple Boolean functions in a form of Multi-Terminal Binary Decision Diagrams (MTBDDs) [3]. The MTBDD cost (the number of nodes) grows for random functions exponentially with the number of function outputs. However, the MTBDD cost is severely bounded for functions with the specified number of true min-terms (the weight). Such functions are frequently used in digital design and the use of MTBDDs is then practical. If the number of outputs is large, one can use output grouping [4], i.e. partitioning outputs to smaller groups. A group of functions can be evaluated simultaneously with MTBDDs, whereas BDD for ECFNs evaluate one function at a time using auxiliary variables [1].

A cascade of LUTs implementing the given function can be obtained by partitioning a MTBDD into slices (layers). The question is how to order variables in the diagram, because the ordering influences dramatically the cost and shape of the MTBDD as well as that of the LUT cascade. Among all possible orderings of variables we should find one that produces a diagram and a LUT cascade optimal in some sense (e.g., diagram cost, width, or average path length and cascade area, length, or area-delay product). The cost of the MTBDD is very sensitive to variable ordering and finding a good order even for BDDs is an NP-complete problem [3]. There are heuristic approaches for MTBDD optimization, e.g. a sifting method or the application-specific variable ordering (ASVO) [5]. For LUT cascades, the minimization of the BDD width is more important than the minimization of the total number of nodes, because the the logarithm of BDD width directly influences the cascade width. A sub-optimal ordering of variables by heuristic iterative decomposition [6], [7] can be targeted for the local cost as well as width minimization. Here we will present optimization of LUT cascades as a separate technique and will assume that the MTBDD, optimized in some way, is already given.

The paper is structured as follows. Basic definitions and concepts are explained in Section II. Properties of general and sparse multiple-output logic functions and their profiles are analyzed in Section III and IV; here new theoretical results regarding number of LUTs needed to realize such functions are obtained. Finally, Section V deals with optimum mapping of MTBDDs to LUT cascades and application of the optimization technique to some benchmark circuits. Theoretical and experimental results are commented on in the Conclusion.

## II. BASIC DEFINITIONS AND NOTIONS

To begin our discussion, we define the following terminology. A system of $m$ Boolean functions of $n$ Boolean variables,

$$f_n^{(i)}: (Z_2)^n \rightarrow Z_2, \; i = 1, 2, ..., m \qquad (1)$$

will be described as a logic function $F_n$ with integer output values from $Z_R = \{0, 1, 2, \ldots, R\text{-}1\}$,

$$F_n: (Z_2)^n \rightarrow Z_R, \qquad (2)$$

where $R$ is the number of distinct $m$-bit output vectors enumerated by values from $Z_R$.

Function $F_n$ is incomplete if it is defined only on set $X \subset (Z_2)^n$; $(Z_2)^n \setminus X = D$ is the don't care set. We assume that all component functions (1) have the same don't care set $D$.

**Definition 2.1** Let $F_n$: $(Z_2)^n \rightarrow Z_R$ be the function of binary variables $x_1, x_2, \ldots, x_n$. **Sub-function** $f(x_{k+1}, \ldots, x_{n-1}, x_n)$ of $n$-$k$ variables is the function $f = F_n(v_1, v_2, \ldots, v_k, x_{k+1}, \ldots, x_{n-1}, x_n)$ for any given combination of binary values $v_1, v_2, \ldots, v_k$.

**Definition 2.2** The MTBDD for the function $F_n$: $(Z_2)^n \rightarrow Z_R$ is a DAG with $n$ levels of decision nodes and with one level of terminal nodes. A decision node controlled by variable $x_k$ has two outgoing edges, one followed when $x_k = 0$, and another when $x_k = 1$.The diagram is ordered, meaning that all nodes in the same level are controlled by the same variable. Input variables are renamed in such a way that variable $x_k$ controls the nodes at level $k$, $k = 1, 2, \ldots, n$.

A node in a MTBDD at level $k+1$ is reached from the root by substitution of the particular vector of binary values $v_1, v_2, \ldots, v_k$ for **bound** variables $x_1, x_2, \ldots, x_k$. If two nodes are reached by the same vector, they are equivalent and unified into one. In each node at level $k$+1 we may substitute arbitrary values for remaining $n$-$k$ **free** variables to reach a terminal node value. Then, according to Definition 2.1 and 2.2, each node at level $k$+1 corresponds to a distinct sub-function of $n$-$k$ variables $x_{k+1}, \ldots, x_n$ .

**Lemma 2.1** (Correspondence between MTBDD partition and circuit decomposition). Let $\mu_k$, $k = 1, 2, \ldots, n$ be the number of distinct sub-functions of $n$–$k$ variables $x_{k+1}, \ldots, x_n$ (column multiplicity in Curtis decomposition). Let $w_k$ be the **local width** of the MTBDD at level $k$, i.e. the number of distinct edges crossing the section of the graph between nodes controlled by $x_k$ and $x_{k+1}$ (edges incident with the same target node are counted only once). Then it holds $w_k = \mu_k$, $\max(\mu_k) = \max(w_k) = w$.

The above lemma follows trivially from the fact that the number $\mu_k$ of nodes in level $k$+1 (i.e. the number of distinct sub-functions of $n$-$k$ variables) is the same as the number of edges coming into this ($k$+1)-th level, counting edges pointing to the same node as one. (Note: in what follows, we prefer to use parameters $w_k$ and $w$ related to MTBDDs rather than equivalent parameters $\mu_k$ and $\mu$ used in recent literature [8]).

**Example 2.1** A sample MTBDD for the 4-valued function $F_4$ of 4 Boolean variables $x_1, x_2, x_3, x_4$ is in Fig.2.1. There are

$w_1 = 2$ sub-functions of 3 variables $x_2, x_3, x_4$
$w_2 = 3$ sub-functions of 2 variables $x_3, x_4$
$w_3 = 3$ sub-functions of single variable $x_4$
$w_4 = 4$ terminal values.

Some sub-functions do not depend on all variables, though. This is reflected in the fact that some decision nodes have a single output edge and the value of a control variable is irrelevant. These degenerated nodes are marked by black dots. We could shift some terminal nodes (2 and 1) up, closer to the root, over them. This makes sense for branching programs only, but it would be difficult to gather the outputs together in hardware implementation where we use mostly binary codes for terminal values. (However, one-hot encoding could benefit from such shifts).

Decomposition of the logic circuit that realizes function $F_4$ into two modules shown in Fig. 2.1b requires that the number of wires between them be at least $\log_2 \lceil w_3 \rceil$, see (4). (End of Example)

**Definition 2.3** Let: $(Z_2)^n \rightarrow Z_R$ be the function of binary variables $x_1, x_2, \ldots, x_n$. The **profile** of the function $F_n$ is the vector $(w_1, w_2, \ldots, w_n)$. Note that $w_1 = 2$, $w_n = R$.

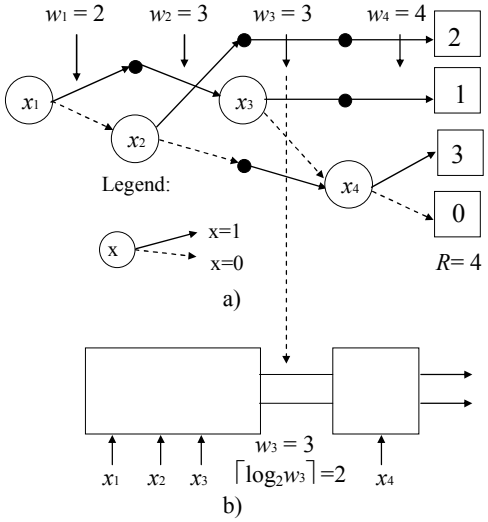


Figure 2.1. Sample MTBDD for the 4-valued function of 4 Boolean variables (a) and decomposition of the associated circuit (b).

**Definition 2.4** A generic binary cascade C of the form $Q \times 1$ is the system

$$C = [Q, H_1, H_2, \ldots, H_n],$$

where

- $H_i$: $(Z_2)^{k_{i-1}} \times Z_2 \rightarrow (Z_2)^{k_i}$, $1 \le i \le n$ is a function implemented by the $i^{th}$ logic device (cell) with $k_{i-1}$ horizontal inputs, one external (vertical) input $x_i$, and $k_i$ outputs; $(Z_2)^0 = \varnothing$.
- input variables are renamed in such a way that variable $x_i$ enters the $i^{th}$ cell in the cascade, $i$ =1, 2,

…, $n$; each input variable used at an external input enters one and only one cell (the so called non-redundant cascade).

- $Q = \max [k_i]$ is a cascade width,
- $n$, the cascade length, is the total number of cells.

Cascade cells have up to $Q$ horizontal inputs (rails) carrying Boolean values between cells and each cell has one additional vertical input. As first cells have $k_0 = 0$, $k_1 = 1$, $k_2 = 2$, $k_3 \in <1, 3>$, $k_4 \in <1, 4>$, …horizontal inputs, first cells are typically combined into a single cell until the combined cell has $Q$+1 external inputs and $Q$ outputs (see Fig. 2.1b where $Q = 2$). The cascade length is then $n - Q$. Cell functions $H_i$ are described by LUTs and the cascade is then referred to as the LUT cascade.

There is a practical reason why we need to reduce the generic cascade length even further, namely the overall delay and the resulting speed. We can combine several consecutive LUT cells easily into one cell to reduce the delay. The cost in bits of this single cell may be larger or smaller than the aggregate cost of individual cells. E.g., combining two cells, both with $Q$+1 inputs, and with the second cell having $q \le Q$ outputs, always reduces or does not increase the overall cost because it always holds

$$2^{Q+1}(Q+q) \ge q2^{Q+2} = 2^{Q+1}(2q) \tag{3}$$

If we combine $p$ consecutive cells into one, then the length of the cascade will become $\lceil (n - Q)/p \rceil$. We are going to study optimal clustering of cells later on in Section V.

Based on Lemma 2.1, the mapping between a MTBDD and the generic LUT cascade can be defined such that

- $w_i$ distinct edges crossing the section of the graph between variables $x_i$ and $x_{i+1}$, are mapped to rails between cell $i$ and $i$+1. Whereas for any input vector only one edge can carry the value 1 ("one-hot" coding), dense binary encoding is used on the rails, so that the number of rails $k_i$ is

$$k_i = \log_2 \lceil w_i \rceil \tag{4}$$

- by enumerating distinct input edges of the level $i$ and $i$+1 by as few integers as possible, a transformation of integer codes under the control variable $x_i$ can be described by a multi-bit LUT and implemented by the $i$-th cell in the cascade.

## III. Properties of Multiple-Output Logic Functions

Before focusing on sparse logic functions, let us shortly review properties of random logic functions and their profiles.

**Lemma 3.1** The logic function $F_n$: $(Z_2)^n \to Z_R$ has up to $\min (2^k, R^{2^{n-k}})$ sub-functions of $n$-$k$ variables, $k$ = 1, …, $n$, but not all of them are necessarily distinct.

(Proof) According to Def. 2.1, each ($n$–$k$)-variable sub-function $(Z_2)^{n-k} \to Z_R$ is related to a binary vector ($v_1$, $v_2$,…, $v_k$). There are $2^k$ such vectors and related sub-functions. On the other hand, the number of ($n$–$k$)-variable sub-functions is limited by the number of function values $R$. Maximum number of single variable ($n$–$k$=1) sub-functions is the same as the number of distinct pairs of function values, i.e. $R^2$. Two-variable sub-functions ($n$–$k$=2) are 4-tuples of function values and there are up to $R^{2^2}$ of them. Continuing in the same way, we have up to $R^{2^{n-k}}$ sub-functions of $n$–$k$ variables ($2^{n-k}$-tuples of function values). A lower value of the two limits gives the bound, QED.

**Corollary 3.1** The local width $w_i$ and the global width of the MTBDD for function $F_n$: $(Z_2)^n \to Z_R$ are upper-bounded by

$$\begin{aligned} w_k &\le \min (2^k, R^{2^{n-k}}) \\ w &\le \max_k (w_k) = \max_k \left[\min (2^k, R^{2^{n-k}})\right] \end{aligned} \tag{5}$$

where $k$ = 0, 1, …, $n$.

**Definition 3.1 Sparse functions**. Under the sparse functions $F_n$: $(Z_2)^n \to Z_R$ we will understand functions with the domain $(Z_2)^n$ divided into two subsets $X$ and $D$, $(Z_2)^n = X \cup D$, $|X| << 2^n$, if one of the following conditions holds:

1) $F_n$ is a fully specified function in $(Z_2)^n$,
   $F_n$: $[X \to Z_R \backslash \{0\}, D \to \{0\}]$
   (without loss of generality, value 0 is taken as the dominant value);
2) $F_n$ is an incomplete function in $(Z_2)^n$, $F_n$: $X \to Z_R$ and $(Z_2)^n \backslash X = D$ is the don't care set.

In this second case we can artificially define mapping $D \to \{0\}$ and come back to the first case. Further on we therefore consider only the first case.

**Definition 3.2** The **weight** of function $F_n$, denoted by $u$, is the cardinality of set $X$ in Def. 3, $u = |X|$.

**Lemma 3.2** Let sparse function $F_n$: $(Z_2)^n \to Z_R$ attains non-zero values 1, 2, …, $R$-1 in $|X| = u$ binary vectors, $X \subset (Z_2)^n$, $R \le u << 2^n$. Then $w \le u$+1 and the profile of the function is upper-bounded by

$$(2, 4, 8, \ldots, 2^h, u+1, u+1, u+1, \ldots, R^{2^i}, \ldots, R^{2^1}, R) \tag{6}$$

where $h = \lfloor \log_2 (u+1) \rfloor$ and $i = \lfloor \log_2 \lfloor \log_R (u+1) \rfloor \rfloor$.

(Proof) The beginning and end of the profile are found from Corollary 3.1. It is sufficient to show that $w \le u$ +1. Each of $u$ binary vectors defines a path from the root to a leaf of the MTBDD. Paths for two different vectors can share some edges or nodes. However, $u$ paths cannot enter more than $u$ nodes in any level of the MTBDD. One more node in each level may be needed for a path leading to the terminal node valued 0. Thus the number of nodes in any level is at most $u$+1, and this determines the values of $h$ and $i$ given above, QED.

**Example 3.1** Let us consider a 5-valued function of 12 variables, $R = 5$, $n = 12$. The profile is according to (5) upper bounded by

(2, 4, 8, 16, 32, 64, 128, 256, 512, 625, 25, 5)

and the MTBDD width $w$ is at most $w = 625$. The change in evaluating the minimum (5) occurs between

$w_9 = \min(\mathbf{2^9}, 5^8) = \mathbf{512}$ and $w_{10} = \min(2^{10}, \mathbf{5^4}) = \mathbf{625}$.

If the above function had weight $u = 39$, then according to (6) we would have $h = 5$, $i = 1$ and the profile would be upper-bounded by

(2, 4, 8, 16, 32, 40, 40, 40, 40, 40, 25, 5).

(End of Example).

**Lemma 3.3** In a LUT cascade, that realizes an $R$-valued logic function $F_n$, let $Q = \log_2 \lceil w \rceil$ be the maximum number of rails; $K = Q + p$ be the number of inputs for a cell, $p \geq 1$; $n$ be the number of input variables, $n \geq K + 1$. Then the LUT cascade with at most $\lceil (n\text{-}Q)/p \rceil$ cells does exist.

(Proof) Each cell, including the first one, has $Q + p$ inputs. The first cell has no left neighbor and no rails coming in, all $Q + p$ inputs are external. Therefore $n–Q–p$ remaining external inputs must be distributed among cells in groups of $p$ inputs. The last $1+\lceil (n\text{-}Q\text{-}p)/p \rceil = \lceil (n\text{-}Q)/p \rceil$-th cell may have less than $p$ external inputs, QED.

The similar Lemma restricted only to Boolean functions appeared in [8]. The restriction can be lifted, Lemma 3.3 extends the result for $R$-valued functions as well as for sparse $R$-valued functions with $w = u+1$. Similarly, the following Theorem 3.1 is generalization of theorems for realization of Boolean functions by 6-LUTs in [8 ].

**Theorem 3.1** The number of 6-LUTs (#LUTs) needed to realize an arbitrary $R$-valued, $n$-variable function having width $w$ (or a sparse function of weight $u = w$-1) is at most

A. case $w \leq 8$ (or $u \leq 7$), $n \geq 8$, $2 < R \leq 16$:
   #LUTs = $n – 6 + \log_2 R$ when $n = 3q$, and
   #LUTs = $n – n \bmod 3 - 3 + \log_2 R$ when $n \neq 3q$
B. case $w \leq 16$ (or $u \leq 15$), $n \geq 8$, $2 < R \leq 32$:
   #LUTs = $2n – 12 + \log_2 R$ when $n$ is even, and
   #LUTs = $2n – 2n \bmod 2 - 8 + \log_2 R$ when $n$ is odd
C. case $w \leq 32$ (or $u \leq 31$), $n \geq 8$, $2 < R \leq 64$:
   #LUTs = $5n – 30 + \log_2 R$.

(Proof) We will prove only the case C, other cases can be handled similarly. All cells have at most $K = \log_2 \lceil w \rceil + p = 5+1 = 6$ inputs and at most $\log_2 \lceil w \rceil = 5$ outputs. From Lemma 3.3, the number of cells is $\lceil (n\text{-}Q)/p \rceil = n\text{-}5$. The last level of decision nodes in the MTBDD can have up to $w_n = w = 32$ nodes (16 if $R = 4$), which can generate at most $R = 2w = 64$ distinct terminal values on at most $\log_2 R$ outputs. Therefore the total number of 6-LUTs is $5(n - 5) – 5 + \log_2 R$, QED.

**Example 3.2** Let us figure out the number of 6-LUTs to realize $R$-valued sparse functions of 16 variables, $R = 2, 4, 8, 16$ and 64 (groups of 1, 2, 3, 4 and 6 Boolean functions) with $w = 32$.

$R = 2$: #LUTs = $5n – 35 = 45$ [8]

$R = 4, 8, 16, 64$:

#LUTs = $5n – 30 + \log_2 R = 52, 53, 54, 56$.

(The same expression for #LUTs cannot be applied for $R = 2$, because 2 rightmost cells can be combined only for $R = 2$.)

(End of Example)

It is seen that theoretically the groups of functions are cascade realizable much more efficiently than a single function. It is the consequence of MTBDD profiles of groups of $\log_2 R$ functions. The size of the group has only a small influence on the profile (with the exception of the last value $w_n = R$).

## IV. Profiles of Sparse Multiple-Output Logic Functions

This Section derives upper bounds on the profiles of sparse logic functions and associated MTBDDs when the weight $u$ is specified. Functions with small weight $u$ have a narrow profile limited by width $w = u+1$. More accurate analysis shows, that the number $w_{n\text{-}k}$ of sub-functions of $k$ variables (and taking up to $t = 2^k$ values) is limited by the weight $u$ and depends also on the number of function values $R$, $w_{n\text{-}k} \leq \lambda(t, u, R)$. The case $w_{n\text{-}k} \leq \lambda(t, u, 2)$ has been analyzed in [8]; here we will address the case $R > 2$.

**Example 4.1** Let us consider sub-functions of single variable $x_n$ ($k$ =1, $t$ = 2) of an $R = 4$-valued function. There are up to $R^{2^k} = 16$ distinct sub-functions $Z_2 \rightarrow Z_R$, that can be displayed in 16 columns of $t = 2^k$ elements each. We can organize column vectors into groups of 1, 6, and 9 columns with 0, 1, and 2 non-zero elements, Table. 4.1. The given number of non-zero elements $u$ may limit the number of sub-functions to less than 16, Fig. 4.1. If we increase $u$, the value of $w_{n\text{-}1} = \lambda(2, u, 4)$ always includes the maximum possible number of columns. This is what we just need for estimating the upper bound of $w_{n\text{-}1}$. Note that $\lambda = u + 1$ for $u \leq t(R\text{-}1) = 6$, i.e. in the first group of sub-functions (with one non-zero value) and $\lambda = R^t = 16$ for $u \geq u_m = 24$. (End of example)

The value of $\lambda(t, u, R)$ is the number of *complete* columns that contain together $u$ non-zero elements; if only some of non-zero elements are counted into the value $u$, the column is incomplete and is not counted into $\lambda$.

Generally the number $u$ of $R$-1 non-zero values of $k$-variable sub-functions and the number of distinct sub-functions $\lambda(t, u, R)$ are in intervals

$$0 < u \leq \sum_{i=1}^{t} i\binom{t}{i}(R-1)^i = u_m, \qquad 2 < \lambda(t,u,R) \leq \sum_{i=0}^{t}\binom{t}{i}(R-1)^i = R^t.$$

TABLE 4.1 SUB-FUNCTIONS OF SINGLE VARIABLE $X_N$ ($k$=1, $t$=2), $R$=4.

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 0 | 0 | 0 | 1 | 1 | 2 | 2 | 3 | 3 | 1 | 2 | 3 |
| 0 | 0 | 0 | 0 | 1 | 2 | 3 | 2 | 3 | 1 | 3 | 1 | 2 | 1 | 2 | 3 |
| 1 | 6 | | | | | | | | | 9 | | | | | |

$u = 12$, $\lambda = 10$

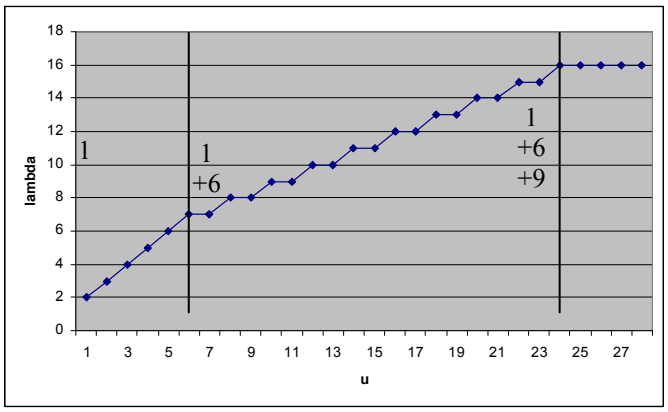


Fig. 4.1 $\lambda(t, u, R) = \lambda(2, u, 4)$

Values of function $\lambda(t, u, R)$ are given in Table 4.2. The values $\lambda = 2^i$, important for cascade realization, are highlighted. Crossing these values changes the number of rails in the cascade given by (4). Table 4.2 makes it possible to estimate the number of LUTs needed to realize $R$-valued functions of $n$ variables for the given weight $u$. The cascade cells with $K$ inputs and $M$ outputs are accounted for as $M$ $K$-input LUTs. Examples are given in Theorem 4.1 below.

**Example 4.2** Let us find the value $\lambda(t, u, R) = \lambda(4, 161, 8)$, not available in Table 4.2. Since

$$\sum_{i=1}^{1} i\binom{4}{i}7^i = 28 < 161 \le \sum_{i=1}^{2} i\binom{4}{i}7^i = 322,$$

the difference 161 – 28 belongs to the interval $i = 2$ where $\Delta\lambda = \Delta u/2$, so that the value of $\lambda$ is

$$\lambda = \sum_{i=0}^{1}\binom{4}{i}7^i + \left\lfloor \frac{161-28}{2} \right\rfloor = 1 + 28 + 66 = 94.$$

(End of Example)

**Theorem 4.1.** The number of 8-LUTs needed to realize an R-valued function of $n$-variables is

- 8 or less, when $R = 4$, $n = 10$, $u \le 102$
- 14 or less, when $R = 4$, $n = 12$, $u \le 78$
- 9 or less, when $R = 8$, $n = 10$, $u \le 63$
- 15 or less, when $R = 8$, $n = 12$, $u \le 63$
- 31 or less, when $R = 8$, $n = 12$, $u \le 127$
- 18 or less, when $R = 16$, $n = 10$, $u \le 224$
- 25 or less, when $R = 16$, $n = 11$, $u \le 134$
- 16 or less, when $R = 16$, $n = 12$, $u = 63$
- 32 or less, when $R = 16$, $n = 12$, $u = 127$
- 39 or less, when $R = 16$, $n = 13$, $u = 127$.

(Proof) Theorem 1 is based on the values of $\lambda$ in Table 4.2 . We will proof only one case for illustration. The profile of an 8-valued function of $n$ =12 variables, $u$ = 63 is, by means of $\lambda(t, 63, 8)$, upper-bounded by

(2, 4, 8, 16, 32, 64, 64, **64**, 60, **46**, 40, **8**).

TABLE 4.2 FUNCTION $\lambda(t, u, R)$, SELECTED VALUES.

| | $\lambda(t,u,R)$ $R=$ 4 | | | | | | | | | | | → $u$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ↓$t$ | 18 | 31 | 38 | 40 | 63 | 78 | 102 | 114 | 158 | 206 | 230 | 303 |
| 2 | 13 | **16** | 16 | 16 | 16 | 16 | 16 | 16 | 16 | 16 | 16 | 16 |
| 4 | **16** | 23 | 26 | **32** | 39 | 46 | 58 | **64** | 79 | 95 | 103 | **128** |
| 8 | 19 | 29 | **32** | 33 | 35 | 52 | **64** | 70 | 77 | 116 | **128** | 165 |
| 16 | 19 | **32** | 39 | 41 | 57 | **64** | 76 | 82 | 89 | **128** | 140 | 177 |
| 32 | 19 | 32 | 39 | 41 | **64** | 79 | 100 | 106 | **128** | 152 | 164 | 200 |

| | $\lambda(t,u,R)$ $R=$ 8 | | | | | | | | | | | → $u$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ↓$t$ | 16 | 31 | 34 | 48 | 63 | 71 | 98 | 112 | 142 | 198 | 227 | 286 |
| 2 | **16** | 24 | 25 | **32** | 40 | 43 | 57 | **64** | 64 | 64 | 64 | 64 |
| 4 | 17 | 30 | **32** | 39 | 46 | 50 | **64** | 71 | 86 | 114 | **128** | 158 |
| 8 | 17 | **32** | 35 | 49 | 60 | **64** | 78 | 85 | 100 | **128** | `42 | 172 |
| 16 | 17 | 32 | 35 | 49 | **64** | 72 | 99 | 113 | **128** | 156 | 170 | 200 |
| 32 | 17 | 32 | 35 | 49 | 64 | 72 | 99 | 113 | 143 | 199 | 226 | **256** |

| | $\lambda(t,u,R)$ $R=$ 16 | | | | | | | | | | | → $u$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ↓$t$ | 31 | 63 | 66 | 96 | 127 | 134 | 194 | 224 | 270 | 390 | 450 | 480 |
| 2 | 31 | 47 | 49 | **64** | 79 | 83 | 113 | **128** | 151 | 211 | 241 | **256** |
| 4 | 32 | 62 | **64** | 79 | 94 | 98 | **128** | 143 | 166 | 226 | **256** | 271 |
| 8 | 32 | **64** | 67 | 97 | 124 | **128** | 158 | 173 | 196 | **256** | 286 | 301 |
| 16 | 32 | 64 | 67 | 97 | **128** | 135 | 195 | 225 | **256** | 316 | 346 | 361 |
| 32 | 32 | 64 | 67 | 97 | 128 | 135 | 195 | 225 | 271 | 391 | 451 | 481 |

The cascade realization requires 3 cells specified by the number of [inputs, outputs]: [8, **6**], [6+2, **6**], [6+2, **3**] because $\log_2 64 = \lceil \log_2 46 \rceil = 6$, $\log_2 8 = 3$. The three cells have 8 inputs and 6, 6, and 3 outputs, see the highlighted values. The number of 8-LUTs is thus 6+6+3=15 or less, QED.

Similar theorems can also be derived for 7-LUTs.

**Example 4.3** The number of 6-LUTs needed to realize an arbitrary $n$-variable function with $w \le 32$ is $5n - 35$ or less, where $n \ge 8$. [8]. If we consider 16-valued function with $u \le 31$, the profile given by $\lambda(t, 31, 16)$ is limited by

(2, 4, 8, 16, 32, 32, …, 32, 31, 16)

and the number of 6-LUTs will be $5(n\text{-}5) - 1 = 5n - 26$ or less, because the last cell has only 4 outputs. Simultaneous realization of 4 functions with $u \le 31$ is thus demanding only 9 extra 6-LUTs over the realization of a single function with $u \le 31$. For $n$ = 10 the simultaneous and separate realization require 6 and 15 6-LUTs per function and for $n$ = 20 it is already 18.5 against 65 LUTs per function.

(End of Example)

**Example 4.4** Find the number of 7-LUTs needed to realize a 16-valued function of $n$ = 9 variables with $u \le 63$. Using the values $\lambda(t, 63, 16)$ from Table 4.2, we can construct the profile of the function as

(2, 4, 8, 16, 32, 64, 62, 47, 16)

and design the cascade, Fig. 4.2. The number of 7-LUTs is at most 16 (2048 bits in total).

For comparison, had we realized 4 Boolean functions of 9 variables with $u \le 55$ separately, we would need 40 6-LUTs, i.e. 2560 bits in total [8]. (End of example)

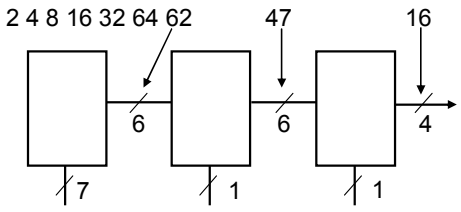


Fig. 4.2. LUT cascade for a 16-valued function of $n = 9$ variables, $u \le 66$.

Two above examples demonstrate that simultaneous realization of group of sparse Boolean functions with the same domain $X \subset (Z_2)^n$, $|X| = u$, may be more efficient than separate realization of individual functions.

## V. Optimal LUT Cascades and Experimental Results

The most important characteristics of logic functions targeted for cascade implementation is their profile. The random functions have the profile (the upper bound) in the shape of a mountain peak with slope at the root rising as powers of 2 and that at the leaves rising steeper, as $2^k$-powers of $R$, see Example 3.1. The profile of sparse functions then looks like a table mountain, with the height $u$+1.

Profiles of some benchmark functions which have been realized by LUT cascades are listed in Tab. 5.1. Four classes of circuits are bcu (branch control unit), rra (round-robin arbiter), lrs (arbiter with least recently serviced strategy) and pe (priority encoder) of different size (specified by the integer after the benchmark code). Profiles have been obtained by the MTBDD heuristic synthesis tool HEDIT with local width minimization [10].

The generic cascades obtained by HIDET with the cascade length equal to the number of variables are much too long. Some cells can be combined and more useful shorter cascades can be obtained. A separate optimization tool has been developed for this purpose, which explores all possible combinations of up to $n = 32$ cells. There are three optional optimization criteria, searching a minimum of

- memory area, regardless the number of cell inputs;
- the product of memory area and cascade length
- memory area when the number of cell inputs $K$ is equal or less than the given value $N$.

Experimental results of benchmark functions optimization are given in Table 5.2. The table gives memory requirements for the cascades with only a single cell, two cells, generic cascades with $n$ cells, and then cascades optimized for memory area or for the product memory area × cascade length. The fraction of a single memory [in %] obtained when dividing memory into #c cells is also given. The interesting result is that the generic cascades are not optimal as far as the memory consumption is concerned. The biggest drop in memory area comes from dividing a single cell into two. Further benchmark-specific subdivision of cascades to 4-12 cells produces some additional decrease in memory area, but after reaching a minimum, the memory area goes up again. This trend is illustrated on the example of lrs6 benchmark with 21 input variables in Fig. 5.1. Optimization for area-time product leads to slightly shorter cascades (2 – 6 cells) and slightly larger memory area.

When the number of cell inputs is given, often two solutions for $K$ and $K$+1 inputs with the same minimum memory appear, as can be anticipated from (3). E.g., Example 4.3 gives $5n - 35$ 6-LUTs for a single function with $w \le 32$. According to Lemma 3.3, in case of $n$ odd we can do with $5\lceil (n\text{-}7)/2 \rceil$ 7-LUTs with the same capacity in bits, but the cell count and thus the delay is cut to one half.

## VI. Conclusions

The obtained theoretical and experimental results can now be summarized. Cascade implementation of sparse $R$-valued functions of Boolean variables was based on MTBDDs. Theoretical results on profiles of such functions made it possible to determine the number of cells and LUTs needed for their realization. It was shown that cascade implementation of multi-valued sparse functions may be more effective than separate implementation of several Boolean functions.

Optimization of cascades for some benchmark circuits leads to a conclusion, that aggregate memory area attains a minimum mostly for only few cells and the product (memory area – time) for even less than that.

The use of the obtained results is possible in LUT-based FPGA synthesis, especially when LUTs with as many as 7 or 8 inputs will become available. Cascade cells are also directly realizable in embedded RAM modules. Several cells may be packed into one memory module and then accessed sequentially or in pipelined fashion by means of a very simple controller [1]. Such implementations may outperform programmable logic controllers which evaluate logic functions in digital control. At the present state of technology, groups of 4 or less Boolean functions ($R \le 16$) are adequate; larger groups are evaluated better by partition outputs into smaller groups.

The future research will target MTBDD synthesis oriented to minimization the number of true decision nodes. Clustering of binary decision nodes into larger $m$-ary decision nodes will be optimized under various criteria, such as speed or code memory area, for a multi-way branching program running on specialized decision diagram machines.

## Acknowledgment

This research has been carried out under the financial support of the research grants "Natural Computing on Unconventional Platforms", GP103/10/1517, "Mathematical and Engineering Approaches to Developing Reliable and Secure Concurrent and Distributed Computer Systems" GA 102/09/H042, and the research plan "Security-Oriented Research in Information Technology", MSM0021630528.

TABLE 5.1 PROFILES OF BENCHMARK FUNCTIONS WITH 8 TO 24 VARIABLES

| name | in | out | profiles |
|---|---|---|---|
| bcu4 | 8 | 4 | 2,2,4,4,8,8,16,16 |
| bcu6 | 12 | 6 | 2,2,4,4,8,8,16,16,32,32,64,64 |
| bcu7 | 14 | 7 | 2,2,4,4,8,8,16,16,32,32,64,64,128,128 |
| bcu8 | 16 | 8 | 2,2,4,4,8,8,16,16,32,32,64,64,128,128,256,256 |
| busarb4 | 8 | 4 | 2,3,4,5,7,8,7,4 |
| busarb6 | 12 | 6 | 2,3,4,5,6,7,11,13,14,14,11,6 |
| busarb8 | 16 | 8 | 2,3,4,5,6,7,8,9,15,19,21,22,22,20,15,8 |
| busarb12 | 24 | 12 | 2,3,4,5,6,7,8,9,10,11,12,13,23,31,37,41,43,44,44,42,38,32,23,12 |
| lrs4 | 10 | 4 | 2,3,3,4,5,4,5,6,6,4 |
| lrs6 | 21 | 6 | 2,3,3,4,5,4,5,6,6,5,6,7,8,7,6,7,8,9,9,8,6 |
| pe8 | 8 | 4 | 2,3,4,5,6,7,8,16 |
| pe12 | 12 | 5 | 2,3,4,5,6,7,8,9,10,11,12,32 |
| pe16 | 16 | 5 | 2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,32 |

TABLE 5.2 MEMORY REQUIREMENTS OF LUT CASCADES FOR BENCHMARK FUNCTIONS WITH 1, 2, IN, AND OPTIMUM NUMBER OF CELLS WITH RESPECT TO MEMORY AREA OR MEMORY X SPEED.

| | | | total cell memory in bits, % of single memory cell | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | #cells=1 | #cells=2 | | #cells=in | | | memory | | | memory * speed | | |
| name | in | out | bits | bits | % | #c | bits | % | #c | bits | % | #c | bits | % |
| bcu4 | 8 | 4 | 1024 | 288 | 28,13% | 8 | 294 | 28,71% | 4 | 196 | 19,14% | 2 | 288 | 28,13% |
| bcu6 | 12 | 6 | 24576 | 2560 | 10,42% | 12 | 1926 | 7,84% | 6 | 1284 | 5,22% | 3 | 1600 | 6,51% |
| bcu7 | 14 | 7 | 114688 | 8192 | 7,14% | 14 | 4614 | 4,02% | 7 | 3076 | 2,68% | 3 | 4352 | 3,79% |
| bcu8 | 16 | 8 | 524288 | 21504 | 4,10% | 16 | 10758 | 2,05% | 8 | 7172 | 1,37% | 3 | 10752 | 2,05% |
| busarb4 | 8 | 4 | 1024 | 320 | 31,25% | 8 | 258 | 25,20% | 4 | 224 | 21,88% | 2 | 320 | 31,25% |
| busarb6 | 12 | 6 | 24576 | 2560 | 10,42% | 12 | 914 | 3,72% | 5 | 816 | 3,32% | 3 | 1088 | 4,43% |
| busarb8 | 16 | 8 | 524288 | 18432 | 3,52% | 16 | 2338 | 0,45% | 8 | 2016 | 0,38% | 4 | 2944 | 0,56% |
| busarb12 | 24 | 12 | 201326592 | 475136 | 0,24% | 24 | 7970 | 0,00% | 12 | 7200 | 0,00% | 6 | 10112 | 0,01% |
| lrs4 | 10 | 4 | 4096 | 384 | 9,38% | 10 | 282 | 6,88% | 4 | 224 | 5,47% | 2 | 384 | 9,38% |
| lrs6 | 21 | 6 | 12582912 | 36864 | 0,29% | 21 | 986 | 0,01% | 9 | 832 | 0,01% | 7 | 992 | 0,01% |
| pe8 | 8 | 4 | 1024 | 320 | 31,25% | 8 | 258 | 25,20% | 4 | 224 | 21,88% | 2 | 320 | 31,25% |
| pe12 | 12 | 5 | 20480 | 1664 | 8,13% | 12 | 802 | 3,92% | 6 | 704 | 3,44% | 3 | 960 | 4,69% |
| pe16 | 16 | 5 | 327680 | 9216 | 2,81% | 16 | 1314 | 0,40% | 8 | 1216 | 0,37% | 4 | 1728 | 0,53% |

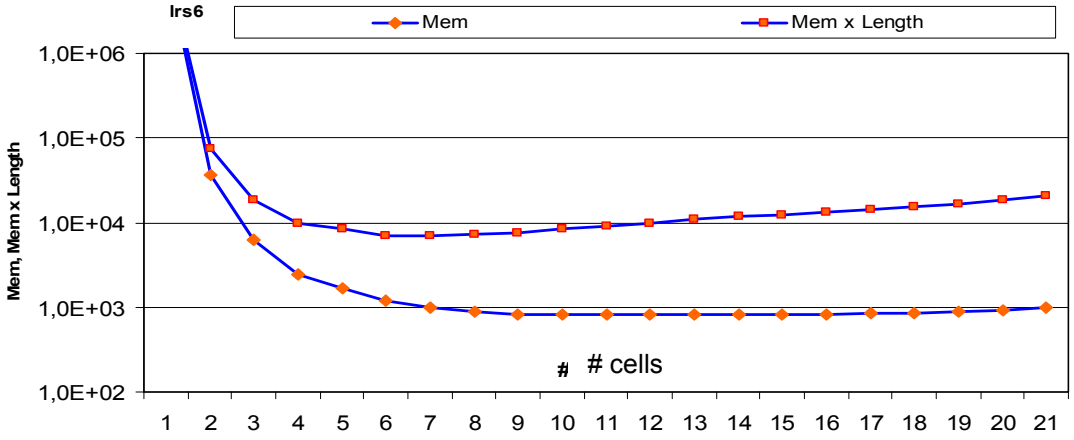


Figure 5.1 Memory area and memory area times cascade length vs the number of cells

## REFERENCES


[1] T. Sasao, M. Matsuura, and Y. Iguchi, "A cascade realization of multiple-output function for reconfigurable hardware," International Workshop on Logic and Synthesis (IWLS01), Lake Tahoe, CA, June 12-15, 2001. pp.225-230.

[2] K. Nakamura, T. Sasao, M. Matsuura, K. Tanaka, K. Yoshizumi, H. Qin, and Y. Iguchi, "Programmable logic device with an 8-stage cascade of 64K-bit asynchronous SRAMs," Cool Chips VIII, IEEE Symposium on Low-Power and High-Speed Chips, April 20-22, 2005, Yokohama, Japan.

[3] S. N.Yanushkevich, D. M. Miller, V. P. Shmerko, R. S. Stankovic, Decision Diagram Techniques for Micro- and Nanoelectric Design Handbook. CRC Press, Taylor & Francis Group, Boca Raton, FL, 2006.

[4] M. Matsuura, T. Sasao, "BDD representation for incompletely specified multiple-output logic functions and its application to the design of LUT cascades," IEICE Transaction on Fundamentals of Electronics, Communications and Computer Sciences, E90-A, No.12, 2007, pp.2770–2777.

[5] R. Drechsler, B. Becker: Binary Decision Diagrams - Theory and Implementation, Springer 1998.

[6] V.Dvořák, "LUT cascade-based architectures for high productivity embedded systems," International Review on Computers and Software, Vol.2, No. 4 , 2007, pp.357–365.

[7] V. Dvořák, "Efficient Eevaluation of multiple-output boolean functions in embedded software or firmware," Journal of Software, Vo. 2, No. 5, 2007, pp.52–63.

[8] T. Sasao, "On the number of LUTs to realize sparse logic functions," 18th International Workshop on Logic and Synthesis, (IWLS-2009), Berkeley, CA, U.S.A., July 31-Aug. 2, 2009, pp.64-71.

[9] T. Sasao, M. Matsuura, "BDD representation for incompletely specified multiple-output logic functions and its applications to functional decomposition," Design Automation Conference, June 2005, pp.373-378.

[10] V. Dvořák, P.Mikušek, "Design of arbiters and allocators based on multi-terminal BDDs," in: Journal of Universal Computer Science, Vol.16, No. 14, 2010, pp. 1826-1852.