# A New Power Estimation Technique with Application to Decomposition of Boolean Functions for Low Power

Peter H. Schneider[1,2] Ulf Schlichtmann[2] Kurt J. Antreich[2]

[1] Siemens ZFE BT SE 52
Otto-Hahn-Ring 6, 81730 Munich

[2] Institute of Electronic Design Automation
Technical University of Munich, 80290 Munich

**Abstract** – *Logic level circuit optimization for low power requires efficient estimation of the number of transitions occuring on signals internal to a circuit. We introduce a new technique based on Markov chains to estimate the transition probabilities of internal signals. Both temporal dependence and multiple concurrent transitions of primary inputs are taken into account.*

*Functional decomposition of Boolean functions is an important synthesis step, especially for look-up-table FPGAs. Typically, functional decomposition optimizes circuits for low area. In this paper we introduce modifications to this method to find solutions with low power consumption. The modified functional decomposition is controlled by the estimated transition probabilities.*

*Detailed experiments on benchmarks demonstrate a reduction of power consumption by 27% on average at a small cost of 5% area increase.*

## 1 Introduction

Two major trends shaping today's design of digital systems. The first trend is the increasing need to consider and drastically reduce power consumption. The second trend is the emergence of field programmable gate arrays (FPGAs).

One strategy to reduce the power consumption is to lower the supply voltage. The leading FPGA-vendor Xilinx already offers 3.3V-FPGAs [1]. Lowering the supply voltage often is not enough, however. Additional reduction of power consumption can be obtained through design improvements both on the architectural and the logic level.

In this paper we will focus on power estimation and optimization at the logic level. The general optimization strategy is to decrease the number of transitions occuring on the internal signals of a circuit. These are of crucial importance to power consumption. Since the number of signal transitions now is part of the objective function it must be estimated for each optimization step of a circuit. So, there is a need for fast yet accurate estimation techniques. Such estimation techniques consider the statistical behavior of primary inputs and derive the transition probabilities of internal signals.

In an early approach Najm [2] suggests a technique to estimate these probabilities. Temporal dependence of primary inputs can be accommodated. Multiple concurrent transitions of primary inputs are not considered, i.e. at each point in time all primary inputs but one are assumed to be constant. Gosh et al. [3] present a method which can handle concurrent transitions but they assume temporal independence for each primary input. They take glitches into account and present a first idea to estimate transitions in sequential circuits.

The estimation technique presented in this paper can take both concurrent transitions of multiple primary inputs and temporal dependence of each primary input into account. The overall technique of our method is a recursive formula which can efficiently be computed on reduced, ordered binary decision diagrams (ROBDDs) [4]. The problem of temporal dependence is solved by introducing Markov chains. To see the importance of temporal dependence, consider a read-enable signal for serial data transmission. While this signal might be 1 with a probability of 0.5, it might switch only every eighth cycle on average. However, methods based on temporal independence will assume that the signal switches every second cycle on average.

Several logic-level optimization methods have been presented to minimize power consumption for digital circuits. All of these methods are based on fast power estimation. Prasad et al. [5] suggest an extraction of common divisors for low power. Lin et al. [6], and Tsui et al. [7] consider the power consumption during technology mapping. Again Tsui et al. [7] propose a NAND-decomposition for low power. Monteiro et al. [8] present retiming for low power. Alidina et al. [9] use precomputation to switch off parts of a circuit if certain assignments of values to the inputs appear.

Disjoint functional decomposition is an effective method to break up Boolean functions. As suggested in [10], it can efficiently be performed on ROBDDs.

In this paper, we show how to perform disjoint functional decompositions that minimize the number of transitions. The new approach uses two key ideas to achieve this result. First, the transition probabilities of signals are taken into account when selecting a partitioning of the variables for decomposition. Furthermore, for signals created during decomposition degrees of freedom are exploited to select an encoding of these signals such that the transition probabilities are minimized. For this second step an additional technique is presented to estimate the transition probabilities of the newly created signals before the decomposition step itself is performed.

The paper is organized as follows. In Section 2 we introduce our new estimation technique. Functional decomposition for low power is described in Section 3. Section 4 presents experimental results.

## 2 Power estimation

### 2.1 Power modeling

In CMOS circuits power is mainly consumed by charging and discharging capacitances [11]. We will focus on this

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission. (c) 1994 ACM 0-89791-687-5/94/0009 3.50

main part of the power consumption. The average power consumption is then given by

$$P_{circuit} = \sum_{\text{signal } i} \frac{1}{2} C_i V_{dd}^2 f E_i \sim \sum_{\text{signal } i} C_i E_i \qquad (1)$$

where $C_i$ is the sum of all input capacitances of those transistors which are driven by signal $i$. $V_{dd}$ is the supply voltage and $f$ the clock frequency. $E_i$ is the probability of a transition on signal $i$ during one clock cycle i.e. the probability that there occurs a $1 \rightarrow 0$ or a $0 \rightarrow 1$ transition on signal $i$. We call $E_i$ the *transition probability* of signal $i$.

While $V_{dd}$, $f$, and $C_i$ are known from the used technology respectively cell library the transition probabilities $E_i$ must be derived from primary input vectors.

## 2.2 General strategy

Our approach is to determine the transition probability at an internal signal $i$ using transition probabilities of primary inputs. For this, we first build a transition function $\mathcal{TF}$ such that $\mathcal{TF} = 1$ holds if and only if a transition occurs on signal $i$ from one clock cycle to the next one. The transition function $\mathcal{TF}$ depends on primary inputs only. So, the required probability $E_i$ is the probability that $\mathcal{TF} = 1$ holds. To compute the probability of $\mathcal{TF} = 1$, we use a recursive technique which is based on the Shannon expansion. During this recursive computation, probabilities are employed which are derived from a Markov chain. ROBDDs are used to perform the recursive computation efficiently.

## 2.3 Transition function

Let $x_i$ be a Boolean variable which describes the Boolean behavior of signal $i$. Then the probability of signal $i$ being 1 is written as $p_1(i)$ $(= p_1(x_i))$. The vector $\underline{x}$ consists of Boolean variables $(\underline{x} = (x_1, x_2, \ldots, x_n))$. Let $y$ be a Boolean variable that corresponds to an output or an internal signal of a circuit. It can be obtained by $y = f(\underline{x})$ where $f(\underline{x})$ is a fully specified single output Boolean function $f(\underline{x}) : \{0,1\}^n \rightarrow \{0,1\}$. Cofactoring function $f$ with respect to variable $x_i$ yields $f_{x_i} = f(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n)$. Note that double cofactoring is commutative, $(f_{x_i})_{x_j} = (f_{x_j})_{x_i} = f_{x_i x_j}$.

Since we are dealing with transition functions, we have to take temporal dependencies into account. With $x_i^0$ we denote a Boolean variable that corresponds to $i(t^0)$ where $t^0$ is an arbitrary point in time. The variable $x_i^T$ corresponds to $i(t^0 + T)$ where $T$ is the duration of one clock cycle. So, the transition probability of signal $i$ can be written as $E_i = E(x_i) = p_1(x_i^0 \oplus x_i^T)$. Furthermore $\underline{x}^0 = (x_1^0, \ldots, x_n^0)$ and $\underline{x}^T = (x_1^T, \ldots, x_n^T)$. Now we can express $E(y)$ as

$$\begin{aligned} E(y) &= p_1(y^0 \oplus y^T) && (2)\\ &= p_1(f(\underline{x}^0) \oplus f(\underline{x}^T)) && (3) \end{aligned}$$

The function $\mathcal{TF} = f(\underline{x}^0) \oplus f(\underline{x}^T)$ is called *transition function*. Its value is 1 if a sequence of two primary input vectors sets $f$ to different values.

## 2.4 Recursive computation of transition probabilities

Primary inputs of a circuit are assumed to be mutually independent and each primary input is first order temporally dependent.

The transition function $\mathcal{TF}$ is decomposed by employing the Shannon expansion $(f = x_i f_{x_i} + \overline{x}_i f_{\overline{x}_i})$ where $\overline{x}_i$ denotes the negation of $x_i$. This way we obtain a recursive formula to compute $E(y)$.

$$\begin{aligned} E(y) &= p_1(\mathcal{TF}) && (4)\\ &= p_1(x_i^0 \mathcal{TF}_{x_i^0} + \overline{x}_i^0 \mathcal{TF}_{\overline{x}_i^0}) && (5)\\ &= p_1(x_i^0 x_i^T \mathcal{TF}_{x_i^0 x_i^T} + x_i^0 \overline{x}_i^T \mathcal{TF}_{x_i^0 \overline{x}_i^T} + \\ &\quad \overline{x}_i^0 x_i^T \mathcal{TF}_{\overline{x}_i^0 x_i^T} + \overline{x}_i^0 \overline{x}_i^T \mathcal{TF}_{\overline{x}_i^0 \overline{x}_i^T}) && (6)\\ &= p_1(x_i^0 x_i^T \mathcal{TF}_{x_i^0 x_i^T}) + p_1(x_i^0 \overline{x}_i^T \mathcal{TF}_{x_i^0 \overline{x}_i^T}) + \\ &\quad p_1(\overline{x}_i^0 x_i^T \mathcal{TF}_{\overline{x}_i^0 x_i^T}) + p_1(\overline{x}_i^0 \overline{x}_i^T \mathcal{TF}_{\overline{x}_i^0 \overline{x}_i^T}) && (7)\\ &= p_1(x_i^0 x_i^T) p_1(\mathcal{TF}_{x_i^0 x_i^T}) + p_1(x_i^0 \overline{x}_i^T) p_1(\mathcal{TF}_{x_i^0 \overline{x}_i^T}) + \\ &\quad p_1(\overline{x}_i^0 x_i^T) p_1(\mathcal{TF}_{\overline{x}_i^0 x_i^T}) + p_1(\overline{x}_i^0 \overline{x}_i^T) p_1(\mathcal{TF}_{\overline{x}_i^0 \overline{x}_i^T}) && (8) \end{aligned}$$

Expressions (6) and (7) are equivalent since $x_i^0 x_i^T$, $x_i^0 \overline{x}_i^T$, $\overline{x}_i^0 x_i^T$, and $\overline{x}_i^0 \overline{x}_i^T$ are mutually disjoint. Expressions (7) and (8) are equivalent since all primary inputs are mutually independent and $\mathcal{TF}_{x_i^0 x_i^T}$ is independent of both $x_i^0$ and of $x_i^T$. To obtain a compact representation we use

$$\begin{aligned} p_{11}(x_i) &= p_1(x_i^0 x_i^T) \qquad p_{10}(x_i) = p_1(x_i^0 \overline{x}_i^T)\\ p_{01}(x_i) &= p_1(\overline{x}_i^0 x_i^T) \qquad p_{00}(x_i) = p_1(\overline{x}_i^0 \overline{x}_i^T) \end{aligned} \qquad (9)$$

Then we obtain

$$\begin{aligned} E(y) &= p_1(\mathcal{TF})\\ &= p_{11}(x_i) p_1(\mathcal{TF}_{x_i^0 x_i^T}) + p_{10}(x_i) p_1(\mathcal{TF}_{x_i^0 \overline{x}_i^T}) + \\ &\quad p_{01}(x_i) p_1(\mathcal{TF}_{\overline{x}_i^0 x_i^T}) + p_{00}(x_i) p_1(\mathcal{TF}_{\overline{x}_i^0 \overline{x}_i^T}) \end{aligned} \qquad (10)$$

With Formula (10) each transition function can be decomposed recursively and the required $E(y)$ is obtained. In the next section we derive the computation of the probabilities of Equation (9).

## 2.5 Markov chain

To derive an accurate computation of the probabilities of (9) we consider each primary input as a discrete, stationary, first order Markov process. First the Markov chain of a primary input is introduced. Then we show the relation to the probabilities of (9).

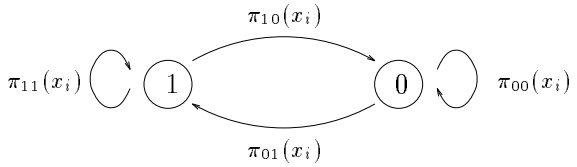


Figure 1: Markov chain

In Figure 1 the Markov chain corresponding to $x_i$ can be seen. From [12] e.g., we know that

$$p_1(x_i) + p_0(x_i) = 1 \quad (11)$$

$$\pi_{11}(x_i) + \pi_{10}(x_i) = 1 \quad (12)$$

$$\pi_{00}(x_i) + \pi_{01}(x_i) = 1 \quad (13)$$

$$p_1(x_i) = p_1(x_i)\pi_{11}(x_i) + p_0(x_i)\pi_{01}(x_i) \quad (14)$$

$$p_0(x_i) = p_0(x_i)\pi_{00}(x_i) + p_1(x_i)\pi_{10}(x_i) \quad (15)$$

where $p_0(x_i)$ is the probability of $x_i$ being 0 and $p_0(x_i) = p_1(\overline{x}_i)$ holds. The meaning of the other probabilities are given in Figure 1. The relation to the probabilities of (9) is

$$\begin{aligned} p_{11}(x_i) &= p_1(x_i^0 x_i^T) = p_1(x_i^0)p_1(x_i^T|x_i^0) \\ &= p_1(x_i)\pi_{11}(x_i) \qquad p_{10}(x_i) = p_1(x_i)\pi_{10}(x_i) \\ p_{01}(x_i) &= p_0(x_i)\pi_{01}(x_i) \qquad p_{00}(x_i) = p_0(x_i)\pi_{00}(x_i) \end{aligned} \quad (16)$$

The reason for (16) is as follows. In Equation (12), e.g., it is assumed that $x_i^0 = 1$ holds. In the next clock cycle either a $1 \rightarrow 1$ or a $1 \rightarrow 0$ transition occurs. However, the probabilities of (9) do not assume that $x_i^0 = 1$ holds. They must take $p_1(x_i)$ into account. Obviously, Equation (17) holds.

$$E(x_i) = p_{01}(x_i) + p_{10}(x_i) \quad (17)$$

Employing (12), (14), (16), and (17) yields

$$p_{10}(x_i) = p_{01}(x_i) = \frac{E(x_i)}{2} \quad (18)$$

From (12), (13), (16), and (18) we obtain

$$p_{11}(x_i) = p_1(x_i) - \frac{E(x_i)}{2} \quad (19)$$

$$p_{00}(x_i) = 1 - p_1(x_i) - \frac{E(x_i)}{2} \quad (20)$$

Since $p_1(x_i)$ and $E(x_i)$ are assumed to be known for primary inputs Formula (10) can now be evaluated.

## 2.6 Employing ROBDDs

In this section we introduce an efficient way to compute the recursive Formula (10). An ROBDD of a transition function $\mathcal{T}f$ is shown in Figure 2. The variables are ordered such that for each $i$, $x_i^T$ immediately follows $x_i^0$ in the variable ordering. An ROBDD with such a variable ordering, called *TFBDD* in the sequel, employs the Shannon expansion in the same way as Formula (10). The reason for this ordering is that $x_i^0$ and $x_i^T$ are mutually dependent but neither $x_i^0$ nor $x_i^T$ depends on any other variable. So, each box in Figure 2 can be handled independently of any other box. Therefore, Formula (10) can be computed on a TFBDD in linear time in terms of TFBDD nodes.

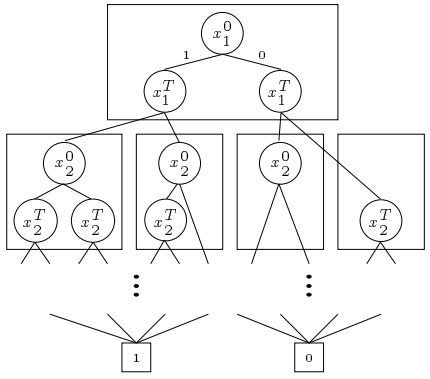

Figure 2: ROBDD of a transition function $\mathcal{T}f$

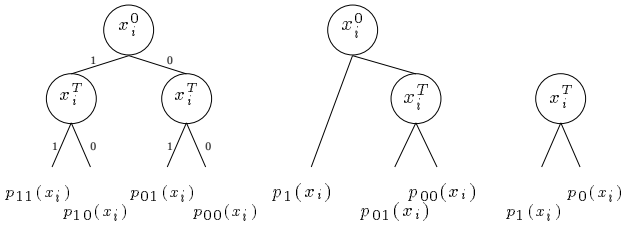

Figure 3: ROBDD structures inside boxes in Fig. 2

Now, we show in detail how this ROBDD based technique works. In Figure 3 some ROBDD structures are shown that can appear inside of a box of Figure 2. For each possible assignment of values to $x_i^0$ and $x_i^T$ the probability of its occurrence is given in Figure 3, also. If an assignment of values is required for only one of $x_i^0$ and $x_i^T$ the probability $p_1(x_i)$ respectively $p_0(x_i)$ is used. In this case, temporal dependence must not be taken into account since only one point in time is considered.

The probability of the terminal node $\boxed{1}$ is 1 ($p_1(1) = 1$) and the probability of the terminal node $\boxed{0}$ is 0 ($p_1(0) = 0$).

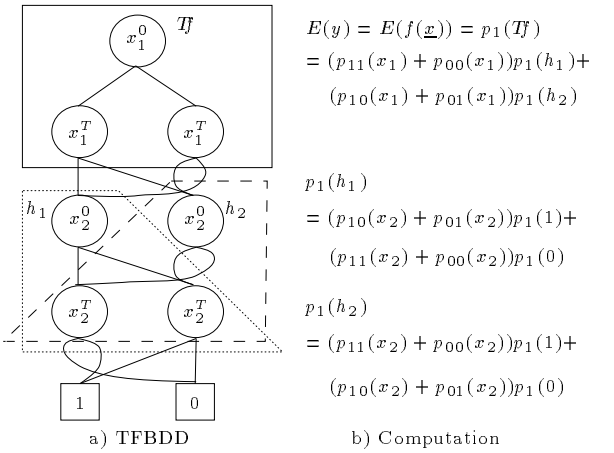

Figure 4: Example for $y = x_1 \oplus x_2$

An example for the ROBDD-based evaluation of the recursive Formula (10) is shown in Figure 4. Let $y = f(\underline{x}) = x_1 \oplus x_2$ hold. In Figure 4a) the TFBDD of $y$ is shown. The computation of the transition probability is performed on this ROBDD. It yields that $\mathcal{T}f_{x_1^0 x_1^T} = \mathcal{T}f_{\overline{x}_1^0 \overline{x}_1^T}$ which is the subfunction represented by the top node of the dotted box denoted as $h_1$. Furthermore, the subfunction $h_2$ represented by the top node of the dashed box is $\mathcal{T}f_{\overline{x}_1^0 x_1^T} = \mathcal{T}f_{x_1^0 \overline{x}_1^T}$. We recursively traverse the ROBDD and compute the probability of each subfunction on the fly by employing the probabilities of its own subfunctions. The resulting probabilities of the function and the subfunctions are given in Figure 4b).

We call the presented method *LinEstimator*. Both concurrent switching and temporal dependence of primary in-

puts are taken into account. The complexity of LinEstimator is linear in number of TFBDD nodes.

# 3 Functional decomposition for low power

## 3.1 Principle of functional decomposition

An important task in logic synthesis is the decomposition of a large Boolean function into a set of smaller functions. An important and popular application is in technology mapping for look-up-table (LUT) FPGAs.

In this section we present an efficient method to perform disjoint decomposition for area optimization [10]. In the next section we will modify this method to optimize power consumption instead of area consumption, using LinEstimator.

We perform decomposition on fully specified, single-output Boolean functions, $F(\underline{z}) : \{0,1\}^{|\underline{z}|} \rightarrow \{0,1\}$. The number of components of the vector $\underline{z} = (z_1, z_2, \ldots, z_{|\underline{z}|})$ is obtained by $|\underline{z}|$. A multi-output Boolean function is a vector $\underline{g}(\underline{x})$ of single-output Boolean functions.

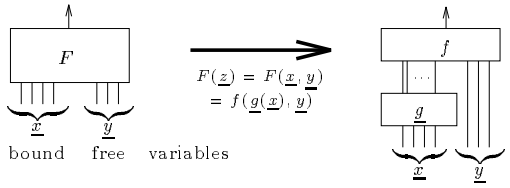


Figure 5: Principle of disjoint decomposition

The main idea of disjoint decomposition is shown in Figure 5. Vector $\underline{z}$ is partitioned into *bound* variables $\underline{x}$ and *free* variables $\underline{y}$. Obviously, $F(\underline{z}) = F(\underline{x}, \underline{y})$ holds. The problem is to find a function $\underline{g}(\underline{x})$ with $|\underline{g}(\underline{x})| < |\underline{x}|$ such that a function $f$ exists with $f(\underline{g}(\underline{x}), \underline{y}) = F(\underline{x}, \underline{y})$. If such functions $\underline{g}$ and $f$ can be found each of them depends on fewer variables than $F$.

The problem of finding such decompositions can be separated into two subproblems. One problem is to determine $\underline{g}$ and $f$, given $F(\underline{x}, \underline{y})$. The other problem is to choose an effective variable partitioning $\underline{z} = (\underline{x}, \underline{y})$, given $F(\underline{z})$, that results in a low area consumption. If the first problem can be solved efficiently, we can evaluate a large number of different variable partitionings in acceptable time.

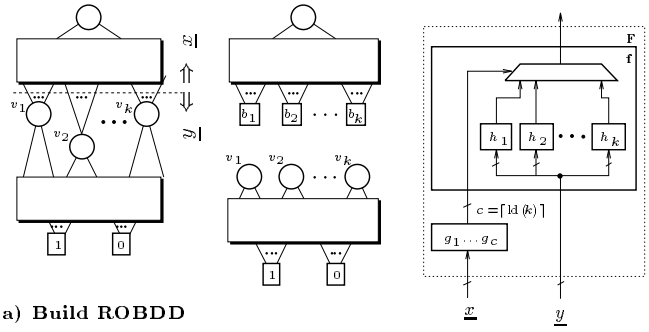


Figure 6: ROBDD-based disjoint decomposition

Recently, an efficient ROBDD-based approach was introduced to determine $\underline{g}$ and $f$, given $F(\underline{x}, \underline{y})$ [10]. We will briefly describe the main idea of this approach, as shown in Figure 6.

First, an ROBDD is constructed for $F(\underline{x}, \underline{y})$ with variables $\underline{x}$ appearing before variables $\underline{y}$ in the variable ordering. If a cutline is introduced to separate ROBDD nodes associated with bound variables from ROBDD nodes associated with free variables, the set $CutNodes(F, \underline{x}, \underline{y})$ is obtained. This set contains all nodes that are located below the cutline and have an incoming edge that originates above the cutline. Each sub-ROBDD with a top node $v_i \in CutNodes(F, \underline{x}, \underline{y})$ represents a Boolean function $h_i(\underline{y})$ depending on free variables only.

Now, imagine the ROBDD partitioned at these nodes. For each node $v_i \in CutNodes(F, \underline{x}, \underline{y})$ a new terminal node $b_i$ is introduced. The set of all these terminal nodes is $CutTerminals(F, \underline{x}, \underline{y})$. The upper ROBDD in Figure 6b) results. It is called *CutROBDD*.

For an usual ROBDD as well as for CutROBDD, each possible assignment of values selects only one terminal node. Therefore, we can think of the Boolean functions $h_i$ as connected to the data inputs of a multiplexer as shown in Figure 6c). Then, $c = \lceil \mathrm{ld}(k) \rceil$ selector inputs to the multiplexer are needed ($k = |CutNodes(F, \underline{x}, \underline{y})| = |CutTerminals(F, \underline{x}, \underline{y})|$). These can be obtained by assigning a $c$ bit wide binary encoding to each of the new terminal nodes $b_i$.

This procedure is described more formally in [10]. In [13] a two-step approach is proposed to find a good variable partitioning $\underline{z} = (\underline{x}, \underline{y})$.

First, a heuristic is employed to obtain an initial variable ordering that minimizes the number of ROBDD nodes. Then, an iterative variable exchange heuristic is used to swap variables as long as the number of $CutNodes$ improves for some cutline.

## 3.2 Minimization of power consumption

Modifications to the basic idea of disjoint functional decomposition will be introduced. The average power consumption is determined by Equation (1). Both the supply voltage $V_{dd}$ and the clock frequency $f$ are fixed for synthesis at logic level. Therefore, we consider the power consumption to be proportional to $\sum C_i E_i$, as can be seen from Equation (1). This expression highlights the difference between optimization for area and optimization for power. A solution with *minimal area* has few capacitances $C_i$. However, an optimization for *minimal power consumption* minimizes the product $C_i E_i$. This is an important difference since the transition probabilities of different signals can vary widely.

Since technology mapping for LUT-based architectures is an important application of disjoint decomposition let us consider this target technology now. For every LUT input we assume the same capacitance. Capacitances interior to a LUT are not considered. Our method reduces the transition probabilities of the LUT inputs as well as the transition probabilities of the LUT outputs. Therefore, on average smaller transition probabilities of the signals interior to a LUT can also be assumed. According to these assumptions the objective is to reduce $\sum fo_i E_i$ where $fo_i$ is the number

of LUTs driven by signal $i$.

To minimize $\sum fo_i E_i$, we propose two key modifications to the algorithm for disjoint decomposition of a function $F(\underline{z})$. First, in the ROBDD another initial order of variables is employed and secondly, the encoding of the terminal nodes $b_1, \ldots, b_k$ is chosen to minimize switching activity.

Each single-output function $g_1, \ldots, g_c$ of the multi-output function $\underline{g}(\underline{x})$ in Figure 6c) needs one single-output LUT to be implemented unless it requires further decomposition. Since the inputs of these LUTs correspond to the bound variables $\underline{x}$, each signal which corresponds to a bound variable possibly feeds more than one LUT input, but each signal that corresponds to a free variable feeds only one LUT input. Therefore, the variables with low transition probabilities are selected as bound variables respectively are ordered at the top of the ROBDD in Figure 6a). To apply this optimization step, first for all signals corresponding to the variables $\underline{z}$ the transition probabilities are estimated using LinEstimator.

The cost function $\sum fo_i E_i$ shows that a large number of LUTs should be avoided. Therefore, starting from the new initial variable partitioning, the iterative variable exchange heuristic is employed as mentioned in Section 3.

The binary encoding of the terminal nodes $b_i \in CutTerminals$ has a significant impact on power consumption. Since each output of $\underline{g}$ feeds one LUT input the power consumption of these LUT inputs is proportional to the sum over the transition probabilities of $g_1, \ldots, g_c$. This sum can already be computed from the nodes in $CutTerminals$. As in the previous section assume $b_i, b_j \in CutTerminals$, $k = |CutTerminals|$, and $c = \lceil \mathrm{ld}(k) \rceil$.

$$\sum_{l=1}^{c} E(g_l) = \sum_{i=1}^{k} \sum_{j=1}^{k} p^{ij} \cdot (\text{HD of codes of } b_i,\, b_j) \qquad (21)$$

where $p^{ij}$ is the probability that such a path of CutROBDD is selected at time $t^0$ which ends in the terminal node $b_i$ and at time $t^0 + T$ a path is selected which ends in $b_j$. After binary encoding the terminal nodes, transitions will occur only for those signals that correspond to different binary values in the codes of $b_i$ and $b_j$. Therefore, the number of signals making a transition is equal to the Hamming distance (HD) of the codes of $b_i$ and $b_j$. The probabilities $p^{ij}$ are computed by the technique presented in section 3.3.

Our goal is to find an encoding that minimizes the sum (21). To achieve this goal we apply a greedy iterative procedure. In every iteration $i, j$ are selected such that $p^{ij}$ is maximized and not both $b_i$ and $b_j$ have already been assigned codes. Then $b_i$ or $b_j$ (respectively $b_i$ and $b_j$) are assigned available binary codes having minimal Hamming distance. This step is repeated until all terminal nodes are encoded. Then every pair of codes is exchanged. As long as any improvement can be achieved this latter step will be repeated.

This or similar techniques can only be employed if the probabilities $p^{ij}$ are known for all pairs of terminal nodes. An accurate but fast computation of these probabilities is the key to find a suitable encoding. To solve this task a variation of LinEstimator was developed. The next section presents this technique.

### 3.3 Estimation of the probabilities required for encoding

We compute $p^{ij}$ directly on CutROBDD. The probability that a sequence of two primary input vectors selects a certain pair of paths in CutROBDD is considered. While the complexity of this method depends on the square number of ROBDD paths, it is suitable for the rather small CutROBDDs encountered in decomposition.

Let $PTH_u$ be a path from the root node to a terminal node in CutROBDD. The set of all paths is $S_{PTH}$. Furthermore, $S^i_{PTH} = \{PTH_u | PTH_u \text{ ends in } b_i\}$ with $b_i \in CutTerminals$. Now, assume such a sequence of two primary input vectors at $t^0$ and $t^0 + T$ that path $PTH_u$ is selected at $t^0$ and path $PTH_v$ is selected at $t^0 + T$, denoted as $PTH_u \rightarrow PTH_v$. The probability of such a sequence of input vectors is equal to $p_1(PTH_u \rightarrow PTH_v)$. To obtain $p^{ij}$ we must take all pairs of paths into account that end in the terminal nodes $b_i$ and $b_j$. Therefore we obtain

$$p^{ij} = \sum_{(PTH_u \in S^i_{PTH}) \wedge (PTH_v \in S^j_{PTH})} p_1(PTH_u \rightarrow PTH_v) \qquad (22)$$

All pairs of paths can be found by a double traversal of CutROBDD. Each pair contributes to some $p^{ij}$. The probability $p_1(PTH_u \rightarrow PTH_v)$ is as follows.

Associated with each node in an ROBDD is some variable $x_m$. The set $V^u$ is the set of all variables associated with nodes in $PTH_u$. Let $V^u_+$ be the set of all variables which must be assigned 1 to obtain $PTH_u$. Similarly, $V^u_-$ is the set of all variables which must be assigned 0 to obtain $PTH_u$. Now we can compute $p_1(PTH_u \rightarrow PTH_v)$.

$$p_1(PTH_u \rightarrow PTH_v) = \prod_{x_m \in V^u \cup V^v} \begin{cases} p_{11}(x_m) & if \quad x_m \in V^u_+ \wedge x_m \in V^v_+ \\ p_{10}(x_m) & if \quad x_m \in V^u_+ \wedge x_m \in V^v_- \\ p_{01}(x_m) & if \quad x_m \in V^u_- \wedge x_m \in V^v_+ \\ p_{00}(x_m) & if \quad x_m \in V^u_- \wedge x_m \in V^v_- \\ p_1(x_m) & if \quad (x_m \in V^u_+ \wedge x_m \notin V^v) \vee \\ & \qquad (x_m \notin V^u \wedge x_m \in V^v_+) \\ p_0(x_m) & if \quad (x_m \in V^u_- \wedge x_m \notin V^v) \vee \\ & \qquad (x_m \notin V^u \wedge x_m \in V^v_-) \end{cases} \qquad (23)$$

Figure 3 may help to understand Equation (23). In Figure 3 the probability $p_{10}(x_i)$ is used if $x^0_i = 1$ and $x^T_i = 0$ hold. Now $p_{10}(x_m)$ is applied if $x_m = 1$ in $PTH_u$ and $x_m = 0$ in $PTH_v$. Since $PTH_u$ and $x^0_i$ relate to $t^0$ and $PTH_v$ and $x^T_i$ relate to time $t^0 + T$, we use $p_{10}$ in both cases. Furthermore, the probabilities $p_1(x_m)$ and $p_0(x_m)$ are used similarly to Figure 3.

## 4 Results

We applied our new approach on a large number of MCNC-benchmarks. Functions which depend on more than 5 inputs were decomposed as long as such functions existed. This corresponds to a technology mapping on FPGAs of the Xilinx XC3000 series. These FPGAs contain a number of identical LUTs with 5 inputs, i.e. each function of up to 5 variables can be implemented in any single LUT. Usually, additional steps are employed to improve the quality

| circuit | trans before | area optim | | power optim | |
|---|---|---|---|---|---|
| | | trans | LUT | trans | LUT |
| 9sym | 0.7 | 2.2 | 6 | 1.6 | 7 |
| t481 | 1.3 | 0.5 | 5 | 5.5 | 32 |
| rd84 | 2.9 | 3.1 | 13 | 1.9 | 13 |
| f51m | 4.1 | 1.7 | 16 | 1.9 | 16 |
| clip | 4.2 | 12.3 | 36 | 4.7 | 29 |
| cordic | 4.5 | 10.2 | 39 | 6.2 | 36 |
| vg2 | 8.7 | 17.7 | 91 | 7.9 | 88 |
| misex3c | 13.8 | 54.5 | 181 | 45.6 | 187 |
| x1 | 37.4 | 48.3 | 221 | 37.4 | 258 |
| ex4 | 14.9 | 53.7 | 247 | 53.9 | 274 |
| duke2 | 22.2 | 38.4 | 259 | 27.8 | 274 |
| alu2 | 30.9 | 112.2 | 373 | 103.0 | 381 |
| pdc | 40.6 | 68.6 | 381 | 55.0 | 414 |
| spla | 48.5 | 101.7 | 376 | 46.1 | 401 |
| ex1010 | 9.3 | 196.5 | 481 | 116.0 | 422 |
| too_large | 33.2 | 209.9 | 775 | 165.8 | 846 |
| $\sum$ | 277.2 | 931.5 | 3500 | 680.3 | 3678 |
| ratio | | 1.00 | 1.00 | 0.73 | 1.05 |

Table 1: Area vs. power optimization

of technology mapping, but we wanted to focus on evaluating the effects of our modifications to the decomposition procedure.

In real designs, $p_1$ and $E$ of primary inputs are known since primary input vectors are existing. Since the probabilities and purposes of primary inputs of MCNC benchmarks circuits are unknown we decided to select probabilities in such a fashion that the distribution of the transition probabilities is similar to real life [6] where an FPGA is only part of a system. For each input the same probability $p_1 = 0.5$ has been assumed, but the transition probabilities $E$ are square distributed in the range of 0 to 0.5.

In Table 1 we present benchmarks which contain a reasonably large number of functions to decompose. In the second column "trans before", for each circuit the sum of the transition probabilities of all those signals is given which already existed before decomposition is performed. Since these signals are still existing after the decomposition this sum is a constant offset.

The next two columns show results for the area oriented functional decomposition introduced in section 3.1. In the column "trans" for each circuit the sum $\sum fo_i E_i$ subtracted by the offset of the second column is given. The sum of the values of the second and the third column yields $fo_i E_i$ for all signals. In the column "LUT" the number of used LUTs can be seen.

The last two columns of Table 1 give the results of the modified procedure aiming at a low power consumption. An interesting result appears for circuit t481. It is well known that the sizes of ROBDD representations of some functions are quite sensitive to variable ordering. The variable partitioning chosen by the power optimization heuristic turned out to be extremely disadvantageous, leading to a very poor final result. This indicates a sensitivity problem of this heuristic in some cases. Otherwise, results are quite encouraging. On average, the area has risen by about 5% and CPU runtime by about a third. However, the transition probabilities of the processed signals have been reduced by 27% which yields a proportional power saving for LUT-based FPGAs. This estimation does not consider glitches which can only be estimated reasonably after placement and routing. Due to the significant reduction of the transition probabilities, a reduced number of glitches can be expected.

# 5 Conclusion

Logic optimization for low power differs from optimization for a traditional cost function as area insofar as the calculation of power consumption is a difficult task. The computation of power consumption must cope with various properties of real applications in short CPU-times. We have presented a novel and efficient technique to estimate transition probabilities in combinational circuits taking into account concurrent transitions and temporal dependence of primary inputs.

The capability to estimate power consumption of circuits fast yet accurately has been used to perform functional decompositions for low power. Estimated power consumption of internal signals controls variable partitioning and encoding of signals created during decomposition. Results for LUT-based architectures show a significant reduction of power consumption by 27% on average with an area penalty of about 5%, only.

# References


[1] "The Programmable Logic Data Book," *Xilinx Inc.*, pp. 2–169 – 2–176, 1994.

[2] F. N. Najm, "Transition Density, A Stochastic Measure of Activity in Digital Circuits," *IEEE Design Automation Conference DAC*, pp. 644 – 649, 1991.

[3] A. Gosh, S. Devadas, K. Kreutzer, and J. White, "Estimation of Average Switching Activity in Combinational and Sequential Circuits," *IEEE Design Automation Conference DAC*, pp. 253 – 259, 1992.

[4] R. E. Bryant, "Graph-Based Algorithms for Boolean Function Manipulation," *IEEE Transactions on Computers*, pp. 677–691, 1986.

[5] S. C. Prasat and K. Roy, "Circuit Activity Driven Multilevel Logic Optimization for Low Power Reliable Operation," *IEEE European Design Automation Conference EDAC*, pp. 368 – 372, 1993.

[6] B. Lin and H. de Man, "Low-Power Driven Technology Mapping under Timing Constraints," *International Workshop on Logic Synthesis IWLS*, pp. 9a–1 – 9a–16, 1993.

[7] C.-Y. Tsui, M. Pedram, and A. M. Despain, "Technology Decomposition and Mapping Targeting Low Power Dissipation," *IEEE Design Automation Conference DAC*, pp. 68 – 73, 1993.

[8] J. Monteiro, S. Devadas, and A. Ghosh, "Retiming Sequential Circits for Low Power," *International Conference on Computer Aided Design ICCAD*, pp. 398 – 402, 1993.

[9] M. Alidina, S. Devadas, J. Monteiro, A. Ghosh, and M. Papaefthymiou, "Precomputation-Based Sequential Logic Optimization for Low Power," *International Workshop on Low Power Design IWLPD*, pp. 57 – 62, 1994.

[10] Y.-T. Lai, M. Pedram, and S. B. K. Vrudhula, "BDD Based Decomposition of Logic Functions with Applications to FPGA Synthesis," *IEEE Design Automation Conference DAC*, pp. 642 – 647, 1993.

[11] L. Glasser and D. Dobberpuhl, "The Design and Analysis of VLSI Circuits," *Addison-Wesley Publishing Company*, 1985.

[12] A. Papoulis, "Probability, Random Variables and Stochastic Processes," *McGraw-Hill*, 1991.

[13] U. Schlichtmann, "Boolean Matching and Disjoint Decomposition for FPGA Technology Mapping," *International IFIP Workshop on Logic and Architecture Synthesis*, pp. 83 – 102, 1993.