Local Transformation Techniques for Multi-Level Logic Circuits Utilizing Circuit Symmetries for Power Reduction *

Ki-Seok Chung

Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801

C. L. Liu

Department of Computer Science
National Tsing Hua University
Hsinchu, Taiwan, ROC

Abstract

In this paper, we present several optimization techniques for power reduction utilizing circuit symmetries. There are four kinds of symmetries that we detect in a given circuit implementation. First, we propose an algorithm for detecting the four different types of symmetries in a given circuit implementation of a Boolean function. Several re-synthesis techniques utilizing such symmetries are proposed. These techniques enable us to optimize power consumption and delay with no (or very little) area overhead. We have carried out experiments on MCNC benchmark circuits to demonstrate the efficiency of the proposed techniques. The average power reduction is 14% with little or none area and/or delay overhead.

1 Introduction

Minimization of circuit area and delay has been the main design objective in VLSI/CAD design for many years. However, low power consumption has recently emerged as another important design objective. Often circuit designers try to reduce the power consumption of a circuit at a cost of increasing the area and delay of the circuit. Such a change of the design objectives often leads to the re-synthesis of an existing circuit to improve the circuit behavior in terms of the new design objective. Also, practically, re-synthesis has a huge advantage over synthesis from scratch in terms of verification cost and design time.

In this paper, we study the problem of re-synthesizing a given circuit utilizing symmetries for power reduction. We assume that the given circuit has been optimized for area and delay, and we want to further improve the circuit in terms of power consumption while maintaining the same functionality. This implies that the re-synthesis techniques employed should not increase the area or delay apparently while reducing the power consumption. We found that re-synthesis utilizing symmetries is an efficient and powerful technique.

To re-synthesize a circuit utilizing symmetries, we need to address the following issues: (i) how to detect symmetries in a given circuit implementation of a Boolean function and (ii) how to utilize the detected symmetries for power reduction.

The remainder of the paper is organized as follows. In the next section, we briefly summarize major ideas on multi-level logic synthesis for low power and those on symmetry detection. Then we present the definitions of four types of symmetries. Next, we propose optimization techniques for power reduction for each type of symmetry. Then we discuss our algorithm for detecting symmetries in a given implementation of a circuit. Experimental results and conclusion will follow.

2 Related Works

A good survey of logic synthesis techniques for multi-level combinational logic was presented in [1]. For low power design, [13] presents an excellent survey. There have been several studies focused on multi-level combinational logic synthesis for low power. In [5], the authors presented techniques for computing power relevant observability and satisfiability don't care conditions which guarantee monotonic power reduction. In [11, 15], the authors determine local transformation based on the computation of *permissible functions*([10]). In [14], the authors studied the problem of area minimization using logic perturbation. In [7, 18], the authors used a similar idea to optimize circuit for power. In [16], the authors studied the problem of identifying subcircuits that can be disabled under a certain condition for power reduction.

Detecting the total or partial symmetry in a logic circuit has been studied since the early 1950s ([2, 3, 6]). Total symmetry of a function can be detected in polynomial time. However, partial symmetry is much harder

*The work was partially supported by the National Science Foundation under grant MIP-9222408 and NSF Career Award MIP95-01615.

to detect. Since exhaustive search of all the possible symmetries is too costly in practice, several techniques have been proposed ([9, 12, 17]). In [9], a symmetry detection procedure using ROBDD was studied. In [12], an efficient method to reorder the variables in the BDDs dynamically so that symmetric variables can be located as close as possible was proposed. In [17], generalized Reed-Muller form was used for the detection of symmetry.

Our approach is to re-synthesize a multi-level logic circuit for power reduction utilizing symmetries. The difference between this work and the other previous works is that we detect circuit symmetry not only among primary inputs but among any primary input and internal signals. To the best of our knowledge, our work is the first attempt to detect symmetries from a given implementation of a circuit. Also we propose several novel techniques for re-synthesis of a given circuit utilizing the detected symmetries for power reduction.

3 Definitions of Symmetry

3.1 Cofactor

The *cofactor* of a Boolean function $f(x_1, \ldots, x_n)$ with respect to x_i is $f_{x_i} = f(x_1, \ldots, 1, \ldots, x_n)$. The *cofactor* of $f(x_1, \ldots, x_n)$ with respect to $\bar{x_i}$ is $f_{\bar{x_i}} = f(x_1, \ldots, 0, \ldots, x_n)$. The Shannon's expansion of a function f is defined by:

$$f(x_1, \ldots, x_n) = x_i f_{x_i} + \bar{x_i} f_{\bar{x_i}} \quad (1)$$

$f(x_1, \ldots, x_n)$ is said to *depend on* x_i if and only if $f_{x_i} \neq f_{\bar{x_i}}$. It is easy to see that the Equation 1 can be generalized with respect to two variables x_i and x_j as follows:

$$f(x_1, \ldots, x_n) = x_i x_j f_{x_i x_j} + x_i \bar{x_j} f_{x_i \bar{x_j}} + \bar{x_i} x_j f_{\bar{x_i} x_j} + \bar{x_i} \bar{x_j} f_{\bar{x_i} \bar{x_j}} \quad (2)$$

A function $f(x_1, \ldots, x_n)$ is *symmetric* in $\{x_i, x_j\}$ (or $\{x_i, \bar{x_j}\}$) if and only if the interchange of the variables x_i and $x_j(\bar{x_j})$ leaves the function invariant. A function f is symmetric in a subset X' of the support X if any permutation of the variables in X' leaves the function invariant. In this case, X' is called a *symmetry subset* of f.

3.2 Symmetry types and conditions

In [4], the notion of symmetric function was extended and five different types of symmetry were defined. We observe that some of them are essentially the same and re-classify them into three. We then introduce one more type of symmetry.

Nonequivalence and Equivalence Symmetry

A function f is *nonequivalence symmetric* in variables x_i and x_j if and only if $f_{x_i \bar{x_j}} = f_{\bar{x_i} x_j}$. This corresponds to the original definition of symmetry given above. Hence later on, the term "symmetry" will mean "nonequivalence symmetry". A function f is said to be *equivalence symmetric* in variables x_i and x_j if and only if $f_{x_i x_j} = f_{\bar{x_i} \bar{x_j}}$. Let $f \sqsubset S_{x_i, x_j}$ denote that function f is symmetric in x_i and x_j.

Multi-form Symmetry

A function f is *multi-form symmetric* in support variables x_i and x_j if and only if f is both nonequivalence and equivalence symmetric in them. In other words, $f_{x_i \bar{x_j}} = f_{\bar{x_i} x_j}$ and $f_{\bar{x_i} \bar{x_j}} = f_{x_i x_j}$. Let $f \sqsubset M_{x_i, x_j}$ denote f is multi-form symmetry in x_i and x_j.

Single-Variable Symmetry

A function f is *single-variable symmetric* in the variable x_j in the space $x_i = 1$ if and only if $f_{x_i \bar{x_j}} = f_{x_i x_j}$. Similarly, a function f is *single-variable symmetric* in the variable x_j in the space $x_i = 0$ if and only if $f_{\bar{x_i} \bar{x_j}} = f_{\bar{x_i} x_j}$. Let $f \sqsubset V_{x_i \to x_j}(V_{\bar{x_i} \to x_j})$ denote that f is single-variable symmetric in x_j under $x_i = 1 (x_i = 0)$.

Pseudo Symmetry

Suppose there are two Boolean functions f and g over n support variables. Suppose for some support variable set $X = \{x_1, \ldots, x_{n-1}\}$, the support of f is $X \cup \{x_f\} (x_f \notin X)$ and that of g is $X \cup \{x_g\} (x_g \notin X)$. Two functions f and g are said to be *pseudo symmetric* in support variable x_f and x_g, respectively, if and only if $f_{x_f} = g_{x_g}$ and $f_{\bar{x_f}} = g_{\bar{x_g}}$. Let $\{f, g\} \sqsubset P_{x_f, x_g}$ denote the pseudo symmetry of f and g with respect to x_f and x_g, respectively.

4 Re-synthesis Utilizing Circuit Symmetries for Power Reduction

First, we discuss how symmetries in a circuit can be utilized for power reduction. We will present our symmetry detection algorithm later. Remember that our primary goal is to reduce the power consumption while taking into account the trade-offs between power and delay. In all methods, the area overhead is either zero or very small.

4.1 Re-synthesis without Area Overhead - Swapping Connections

According to the definition of symmetry subsets, a function remains invariant under any permutation of variables in the symmetry group. In the corresponding circuit implementation, "permutation" means swapping connections. See Figure 1-(a). Through swapping, we might find a better circuit in terms of power consumption. Figure 2-(a) shows a circuit where $g \sqsubset S_{e,i}$. Therefore, we can swap the connections and obtain the circuit in Figure 2-(b) which is functionally equivalent to the circuit in Figure 2-(a).

Since a multi-form symmetry implies a nonequivalence symmetry ($S_{s,t}$), we can swap connections of s and t if $f \sqsubset M_{s,t}$. Furthermore, since a multi-form symmetry implies an equivalence symmetry ($S_{\bar{s},t}$), if there exists a signal which corresponds to $\bar{s}(\bar{t})$, then we can swap the connections to $\bar{s}(\bar{t})$ and $t(s)$ if $f \sqsubset M_{s,t}$. See Figure 3 for an example using a real circuit.

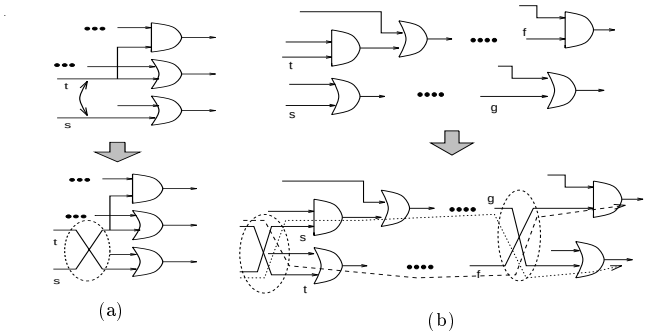

Figure 1: Swapping connections: (a) Symmetry $f \sqsubset S_{s,t}$ (b) Pseudo symmetry $\{f, g\} \sqsubset P_{s,t}$

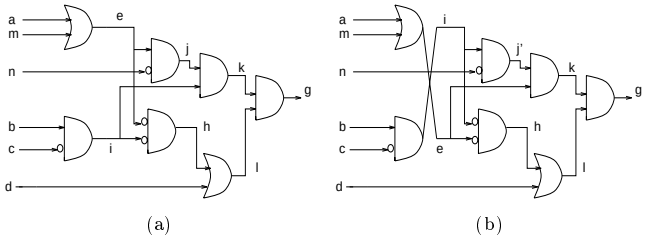

Figure 2: (a) Symmetry $g \sqsubset S_{e,i}$ (b) Equivalent circuit with connections swapped

A similar yet more interesting case can be seen in a circuit with pseudo symmetry. In this case, to ensure that the functionality of the circuit remains invariant, we need to swap two pairs of connections as shown in Figure 1-(b). See Figure 4 for a concrete example.

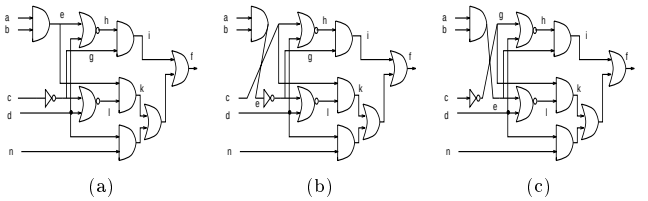

Figure 3: (a) Multi-form symmetry $f \sqsubset M_{c,e}$ (b) An equivalent circuit with connections swapped (c) Another equivalent circuit with connections swapped

4.2 Re-synthesis with Area Overhead - Structural Transformation

Besides connection swapping, following techniques can be employed for power reduction. It should be noted, however, that these techniques do not preserve the area of the initial implementation. In other words, it may reduce or may increase the total area of the circuit.

Multi-form Symmetry

Suppose a signal f is multi-form symmetric in two signals s and t. It means that if in clock cycle i, $s = t = 0 (s = t = 1)$, and in the next clock cycle $i + 1$, $s = t = 1 (s = t = 0)$, then the value of f is unchanged (because $f_{st} = f_{\bar{s}\bar{t}}$) and the transitions in s and t are immaterial. Similarly, when $s = 0$ and $t = 1$ in one clock cycle and $s = 1$ and $t = 0$ in the next clock cycle (or vice versa), the value of f is unchanged (because $f_{\bar{s}t} = f_{s\bar{t}}$). Consequently, reduction in switching activities can be obtained by removing some of the unnecessary transitions in the transitive fan-outs of the variables s and t. The redesign is carried out as follows: (We assume that f is multi-form symmetric in s and t with t having more transitions than s.)

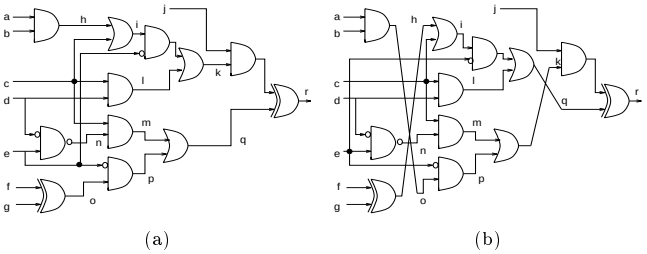

Figure 4: (a) Pseudo symmetry $\{k, q\} \sqsubset P_{o,h}$ (b) An equivalent circuit with connections swapped

1. Connect constant signal 0(or 1) in place of t. The choice between 0 and 1 is dependent on the type of the gates in the fan-out of t.
2. Add a two input XOR (XNOR) gate if 0(1) is selected for t, with s and t as the inputs to the XOR (XNOR) gate. Then, connect the output of the XOR (XNOR) gate to the fan-out of (the original) s.

Figure 5-(a) shows the transformation where 0 and XOR are chosen to be used. In this transformation, the area overhead is at most equal to the introduction of a new gate (`g0`), but there may be some removal of gates (`g1,g2`) in the transitive fan-out of t.

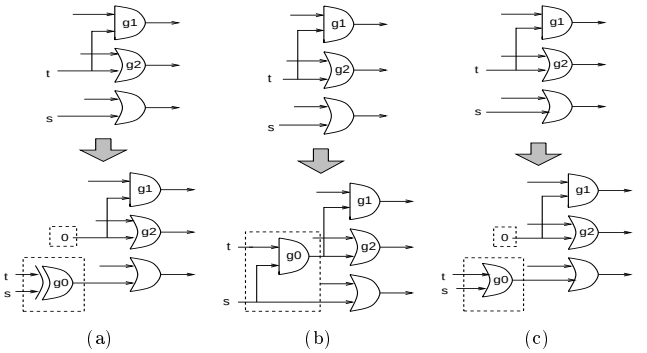

Figure 5: Structural transformation for (a) Multi-form symmetry $f \sqsubset M_{s,t}$ (b) Single-Variable symmetry $f \sqsubset V_{s \to t}$ (c) Both $f \sqsubset V_{s \to t}$ and $f \sqsubset S_{s,t}$

Single-Variable Symmetry

Recall that $f \sqsubset V_{s \to t}$ implies that $f_{st} = f_{s\bar{t}}$. This means that if $s = 1$, f does not depend on t, which

implies that if $s = 1$, then we can remove the spurious transitions on the transitive fan-out of t due to the value changes in t. This can be implemented by adding a gate as shown in Figure 5-(b). The gate added will be an OR gate for any $V_{s \to t}$ and an AND gate for any $V_{\bar{s} \to t}$. We can observe that the additional OR (AND) gate will be used for gating signals of t when $s = 1 (s = 0)$.

Single-Variable Symmetry with Nonequivalence Symmetry

Recall that if both $f \sqsubset V_{s \to t}$ and $f \sqsubset S_{s,t}$, then $f_{st} = f_{s\bar{t}} = f_{\bar{s}t}$. This means that as long as the values of s and t remain 1 and 1, 0 and 1 or 1 and 0 over consecutive clock cycles, respectively, the value of f will remain unchanged and spurious switchings in the transitive fan-out of s and t due to the value changes in signals s and t can be suppressed. Thus, we only need to be able to distinguish between the signal value combination of $s = 0$ and $t = 0$ from the other signal value combinations. Hence, an additional OR gate (g0) is introduced in the corresponding transformation as shown in Figure 5-(c). For the case where $f_{\bar{s}\bar{t}} = f_{s\bar{t}} = f_{\bar{s}t}$ (equivalently, if both $f \sqsubset V_{\bar{s} \to t}$ and $f \sqsubset S_{s,t}$), an additional AND gate will be used. To summarize:

1. Connect constant signal 0(1) in place of t. The choice between 0 and 1 is dependent on the type of the gates in the fan-out of t.
2. Introduce an OR (AND) gate with both s and t as inputs to the gate. The output of this OR (AND) gate becomes the fan-out of (the original) s.

Pseudo Symmetry

Let $sup(f)$ denote the set of support variables for a function represented by f. Pseudo symmetry implies that for some support set X, where $s, t \notin X$, there are two functions f, g where $sup(f) = \{s\} \cup X$ and $sup(g) = \{t\} \cup X$ and $g_{\bar{s}} = f_{\bar{t}}$ and $g_s = f_t$. This means that we can share the function evaluation when $s = t$. This observation leads to a re-synthesis structure as shown in Figure 6. In this structure, f always computes $f_{\bar{t}}$, and g computes g_s (Or, f computes f_t and g computes $g_{\bar{s}}$).

5 Selection of Transformations

In Sections 4.1 and 4.2, we showed how a circuit can be re-synthesized for the occurrence of each type of symmetry. However, in a circuit, there may be many occurrences of various types of symmetries. Thus, the question is: "How do we select one transformation over the others in order to maximize the effectiveness of the transformation?"

Remember that a transformation with respect to signals x and y utilizes a certain (set of) symmetry of the two signals in the circuit. We have found that depending on how the signals in the symmetric subset are correlated, a transformation can be more effective in terms

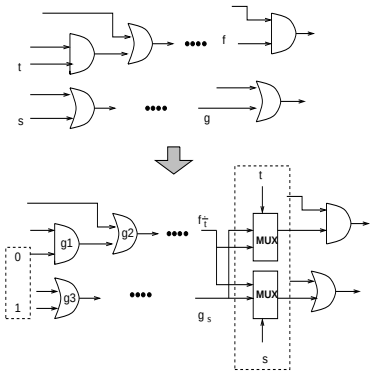

Figure 6: Transformation for pseudo symmetry

of the reduction in switching activity than the others. There are two different notions of correlations: temporal correlation and spatial correlation [8].

We have found that there is a direct relationship between a certain correlation and the effectiveness of a transformation. Suppose a circuit is symmetric in x and y. Let O_{ab} denote the occurrence probability of x being a and y being b (approximation to the spatial correlation), and let $T_{ab,cd}$ denote the transition probability between the two states where signal x goes from a to c and signal y goes from b to d (approximation to the temporal correlation)[1]. Table 1 summarizes the relations. $\uparrow$ represents that the corresponding value should be high, and $\downarrow$ represents that the value should be low in order for a transformation to be effective for power reduction. Remember that each transformation utilizes a certain type of symmetry. Therefore, we use the type of symmetry to indicate the corresponding transformation.

Type(x, y)	Type of symmetry	Correlation
Sect. 4.1	$S_{x,y}$	$O_{01} \uparrow$ or $O_{10} \uparrow$
	$S_{x,\bar{y}}$ (or $S_{\bar{x},y}$)	$O_{00} \uparrow$ or $O_{11} \uparrow$
Sect. 4.2	$M_{x,y}$	$T_{00,11} \uparrow \wedge T_{01,10} \uparrow$
Sect. 4.2	$V_{\bar{x} \to y} \wedge S_{x,y}$ (or $V_{\bar{y} \to x} \wedge S_{x,y}$)	$O_{11} \downarrow$
	$V_{x \to y} \wedge S_{x,y}$ (or $V_{y \to x} \wedge S_{x,y}$)	$O_{00} \downarrow$
	$V_{y \to x} \wedge S_{x,\bar{y}}$ (or $V_{\bar{x} \to y} \wedge S_{x,\bar{y}}$)	$O_{10} \downarrow$
	$V_{x \to y} \wedge S_{x,\bar{y}}$ (or $V_{\bar{y} \to x} \wedge S_{x,\bar{y}}$)	$O_{01} \downarrow$
Sect. 4.2	$V_{\bar{x} \to y}$	$T_{01,00} \uparrow$
	$V_{\bar{y} \to x}$	$T_{00,10} \uparrow$
	$V_{y \to x}$	$T_{01,11} \uparrow$
	$V_{x \to y}$	$T_{10,11} \uparrow$

Table 1: Desired correlation in order for a transformation to be effective for power reduction

[1] For the sake of brevity, we assume that $T_{ab,cd} = T_{cd,ab}$.

6 An Algorithm for Symmetry Detection in a Circuit Implementation

In this section, we describe our algorithm for detecting the four types of symmetries defined in Section 3.2. Even though there are several symmetry detection methods in the literature, they are not suitable for logic re-synthesis because they do not utilize the structural information provided by the given circuit structure and they only detect symmetry with respect to primary inputs.

Let s be the first signal considered and call it a *source*, and t be the second signal considered and call it a *target*. Let $FI(v)$ and $FO(u)$ be the set of fan-ins of v and the set of fan-outs of u, respectively. Let C_v represent the function of the cone that v represent. Let $TFI(v)$ and $TFO(u)$ represent *transitive fan-in* and *transitive fan-out*, respectively. Let *Level(v)* be the maximum distant from the primary inputs.

6.1 An Observation

The key observation that leads to our symmetry detection algorithm is as follows:

Observation 6.1 *Let f be a single output function. Suppose we want to check the symmetry between two given signals s and t. Assume that there is no multiple fan-out from s and t to f. Let c be a signal in $TFO(s) \cap TFO(t)$.(See Figure 7-(a).) Then if c is symmetric in s and t, then f is symmetric in s and t as well.*

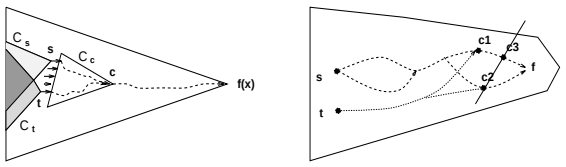

Figure 7: (a) Key observation of symmetry detection from a circuit, (b) Symmetry detection from a circuit with multiple fan-outs

The observation above implies the following: First, if $Level(c) << Level(f)$, the cost of symmetry detection at c will be much cheaper than at f. Second, we observe that the symmetry at c with respect to s and t does not depend on what C_s or C_t is (shaded regions in Figure 7-(a)). Finally, the switching activities for any signal in $TFO(c)$ or any signal outside of C_c are unchanged when the connections to s and t are swapped. Thus, our problem is to find the c which is as close as to s and t as possible.

When there are multiple fan-outs and reconvergent fan-outs in the transitive fan-out of s and t, $TFO(s) \cap TFO(t)$ will in general be a set of signals where the signals in the set may be independent of each other as $c1$ and $c2$ in Figure 7-(b). The extension is straightforward. For each fan-out path from s and t, there should be a signal c that is symmetric in s and t. Again, we try to find the set of such c's which are as close to s and t as possible.

6.2 Detection Algorithm

Our symmetry detection algorithm is given here. It detects the four types of symmetry with respect to signal t and all other signals which are independent of t.

```
Detecting Symmetry From a Circuit Implementation I
  Compute the primary input support for each signal
  Simulate I to compute switching activity
  Identify the signal t with the highest switching activity
  Update the support for signal n ∈ TFO(t)
  For every signal s that is independent of t
    Update the support for signal n ∈ TFO(s)
    Identify the set of check points CP for symmetry detection
    For every checkpoint c ∈ CP
      Construct a minimal cone with respect to c, s, t
      Check the four types of symmetries at c in s, t
      If All the necessary checkpoints are found
          Stop
    EndFor
  EndFor
END
```

7 Experimental Result

We have implemented the algorithm for symmetry detection and the re-synthesis techniques presented in this paper. The programs were written in C++ running on a Sun Sparc10 workstation. We carried out experiments on many MCNC benchmark circuits.

From a given circuit, we identify a signal which has the highest switching activity. The switching activity of each signal line is estimated on the basis of the onprobabilities of the primary inputs using SIS[2]. Then we try to detect the existence of any of the four types of symmetries in the circuit with respect to this signal and other signals. Then the circuit is re-synthesized using the techniques presented above. After the transformation, we re-evaluate the switching activities in the new circuit. And we iterate this process by selecting the signal with the highest switching activity. In this experiment, the number of iterations was 0.2 × the total number of gates in the circuit. The power consumption, area, and delay were computed after executing a SIS script: "`sweep; eliminate -l; map -m 0`." The power measurement was conducted using the power estimation tool in SIS 1.2 assuming a 20MHz clock and 5V. Furthermore, we measured the power of mapped circuits using the well-known MCNC generic library.

Table 2 summarizes the result. `#PI` and `#PO` represent the number of primary inputs and primary outputs, respectively. `Size` is the number of gates in the circuit when the circuit is decomposed using `tech_decomp`

[2] `power_estimate -d GENERAL` and `power_print` were used.

Circuit	#PI	#PO	Size	Before			After			% Power	Time
				Power	Delay	Area	Power	Delay	Area		(sec)
cm162a	14	5	83	203	16.7	78	182	13.7	78	10.7	21
pm1	13	10	93	299.2	14.5	109	277	15	114	7.4	14
cu	14	11	94	357.7	13	109	321.3	11.6	114	10.2	20
cm152a	11	1	36	119	11.3	32	82.4	11	43	30.7	2
pcler8	27	17	94	224.1	27.7	165	200.1	33.5	144	10.7	35
sct	19	13	237	548.2	18.1	232	517.4	23.7	237	5.6	329
cmb	16	4	78	241.2	8.4	77	218	16.5	77	9.6	17
i2	201	1	233	730.9	18	306	630.4	22.5	317	13.7	2304
mux	21	1	160	564.5	25.5	188	391.7	40.3	228	30.6	505
cm85a	11	3	76	123.7	21	72	100.3	26.7	84	18.9	18
9symml	9	1	243	1000.2	28.3	353	973.6	28.7	355	2.7	450
b9	41	17	219	514.0	16.8	191	394	16.7	198	7.7	324
cc	21	13	103	367.8	13.5	113	339.0	13.6	107	7.8	23
c8	28	17	315	924.6	18.1	375	822.5	18.8	378	11.0	1226

Table 2: Re-synthesis of logic circuit utilizing symmetries

command in SIS. The columns under `Before` show the power consumption, area, and delay of the circuits before re-synthesis and those under `After` show the result after applying re-synthesis techniques. `% Power` shows the percentage of power reduction achieved through re-synthesis. `Time` shows the user CPU time for the detection of the symmetries and the applications of corresponding transformations[3]. We were able to obtain the average power savings of 14% with very little area and delay overhead.

8 Conclusion

We proposed several techniques for re-synthesizing multi-level circuits utilizing symmetries for power reduction in this paper. We studied: (i) definitions of different types of symmetry, (ii) how to detect such symmetries in a given implementation of a circuit and (iii) how to utilize the detected symmetries for power reduction. We find that re-synthesis utilizing circuit symmetry is very efficient and powerful for power reduction.

We are currently investigating several challenging issues related to this work. Our future work includes:

- Considering internal don't care conditions in the circuit to detect more symmetries and extending the set of applicable transformations with the existence of such don't care conditions
- Developing an efficient technology decomposition algorithm which can be used with symmetry detection

References

[1] R. K. Brayton, G. D. Hachtel, and A. L. Sangiovanii-Vincentelli. Multilevel logic synthesis. *Proceedings of the IEEE*, 78(2):264–300, February 1990.

[2] S. R. Das and C. L. Sheng. On detecting total and partial symmetry of switching functions. *IEEE Transactions on Computers*, 20(3):352–355, March 1971.

[3] Donald L. Dietmeyer and Peter R. Schneider. Identification of symmetry, redundancy and equivalence of boolean functions. *IEEE Transactions on Electronic Computers*, EC-16(6):804–817, December 1967.

[4] Colin R. Edwards and S. L. Hurst. A digital synthesis procedure under function symmetries and mapping methods. *IEEE Transactions on Computers*, C-27(11):985–997, November 1978.

[5] Sasan Iman and Massoud Pedram. An approach for multilevel logic optimization targeting low power. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 15(8):889–901, August 1996.

[6] Zvi Kohavi. *Switching And Finite Automata Theory, 2nd Edition*. McGraw-Hill, 1978.

[7] Wolfgang Kunz and Dhiraj K. Pradhan. Recursive learning: A new implication technique for efficient solutions to cad problems = test, verification, and optimization. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 13(9):1069–1078, September 1994.

[8] Radu Marculescu, Diana Marculescu, and Massoud Pedram. Switching activity analysis considering spatiotemporal correlations. In *International Conference on Computer-Aided Design*, pages 294–299, November 1994.

[9] Dirk Möller, Janett Mohnke, and Michael Weber. Detection of symmetry of boolean functions represented by ROBDDs. In *International Conference on Computer-Aided Design*, pages 680–684, November 1993.

[10] Saburo Muroga, Yahiko Kambayashi, Hung Chi Lai, and Jay Niel Culliney. The transduction method-design of logic networks based on permissible functions. *IEEE Transactions on Computers*, 38(10):1404–1423, October 1989.

[11] Rajendran Panda and Farid Najm. Post-mapping transformations for low-power synthesis. In *Design Automation Conference*, pages 650–655, 1997.

[12] Shipra Panda, Fabio Somenzi, and Bernard F. Plessier. Symmetry detection and dynamic variable ordering of decision diagrams. In *International Conference on Computer-Aided Design*, pages 628–631, 1994.

[13] Massoud Pedram. Power minimization in IC design: principles and applications. *ACM Transactions on Design Automation of Electronic Systems*, 1(1):3–56, January 1996.

[14] Perturb and Simplify: Multi level Boolean Network Optimizer. Shih-chieh chang and malgorzata marek-sadowska. In *International Conference on Computer-Aided Design*, pages 2–5, 1994.

[15] Bernhard Rohfleisch, Alfred Kölbl, and Bernd Wurth. Reducing power dissipation after technology mapping by structural transformation. In *Design Automation Conference*, pages 789–794, 1996.

[16] Vivek Tiwari, Sharad Malik, and P. Ashar. Guarded evaluation: Pushing power management to logic synthesis. In *International Symposium on Low Power Design*, pages 221–226, September 1995.

[17] Chieh-Chung Tsai and Malgorzata Marek-Sadowska. Generalized reed-muller forms as a tool to detect symmetries. *IEEE Transactions on Computers*, 45(1):33–40, January 1996.

[18] Qi Wang and Sarma B.K. Vrudhula. Multi-level logic optimization for low power using local logic transformation. In *International Conference on Computer-Aided Design*, pages 270–277, 1996.

[3] It does not include the time for the power estimation.