# Reducing Power Dissipation after Technology Mapping by Structural Transformations

BERNHARD ROHFLEISCH ALFRED KÖLBL BERND WURTH

Institute of Electronic Design Automation
Technical University of Munich
80290 Munich, Germany

Synopsys, Inc.
700 E. Middlefield Rd.
Mountain View, CA 94043

**Abstract – Due to the increasing demand for low power circuits, low power dissipation has emerged as an important optimization goal in logic synthesis. In this paper, we show that the power dissipation of technology mapped circuits can be significantly reduced by ATPG-based structural transformations. Our approach performs a sequence of permissible signal substitutions, where each substitution reduces the power consumption of the circuit. Since timing constraints can be considered, we achieve a trade-off between power and delay.**

**The effectiveness of the proposed method is based on two facts. First, the power models for library gates are reasonably accurate. Thus, the power savings achieved by transformations of mapped circuits are also well modeled. Second, ATPG-based structural transformations effectively exploit don't care conditions after technology mapping even for large circuits.**

**Experimental results show power reductions of 26% on average with no area penalty. Substantial power reductions are also achieved if timing constraints are considered.**

## 1 INTRODUCTION

Power consumption has emerged as an important optimization goal in the design of VLSI circuits. This is due to two driving forces. First, modern portable applications require a low power consumption to increase battery lifetime. At the same time, high throughput of such devices is demanded. Second, power consumption limits the number of transistors that can be integrated on a single chip because of packaging and cooling problems.

Low power consumption can be targeted at various levels of the design process, e.g. at the system, architectural, logic and physical levels. In this paper, we tackle power minimization at the logic level.

Combinational logic synthesis is traditionally partitioned into two phases. First, a Boolean network is optimized independently of the chosen target library. Second, functions in the network are mapped to the library (technology mapping). This phase yields a logic netlist. Finally, gate re-sizing can further optimize the netlist before placement and routing. Recently, *structural netlist optimization* has been developed as an additional synthesis phase [1, 2, 3, 4, 5]. Structural netlist optimization methods have been shown to effectively reduce area and delay by exploiting don't cares after technology mapping. Figure 1 shows the complete logic synthesis flow.

Each of the traditional phases has significant impact on power consumption, and a variety of optimization methods has been proposed to reduce power dissipation in each phase. Iman et al. reduce power consumption during the technology independent optimization phase by exploiting don't cares [6] and extended algebraic techniques [7]. Initial work on this topic includes [8, 9].

Much research has been done to consider power consumption during technology mapping. This logic synthesis phase is appealing for power minimization since accurate power and delay models for gates exist. Both the decomposition [10, 11] and the covering step [10, 12, 13] have been optimized. In order not to deteriorate the circuit delay, timing constraints are taken into account by some of these mapping methods [10, 13]. Bahar et al. [14] also consider timing constraints during gate re-sizing for low power.

In this paper, we address the problem of reducing power dissipation after technology mapping by structural optimization of the netlist. Our approach performs a sequence of permissible signal substitutions, where each substitution reduces the power consumption of the circuit. Since our approach transforms mapped circuits, the power consumption can be modeled more accurately than before technology mapping. Accurate modeling guarantees a consistent power optimization. Similarly, due to accurate delay models timing constraints are considered easily. By varying the user-specified timing constraints, our approach is able to achieve a nice trade-off between power and delay. We use ATPG methods to identify permissible structural transformations. Therefore, we don't need global BDDs, which are required by other techniques to exploit functional don't cares.

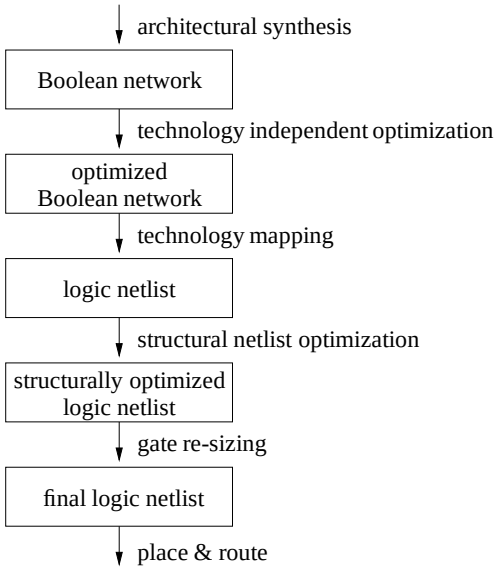


Figure 1: Logic Synthesis Flow.

**33rd Design Automation Conference ®**


DAC 96 - 06/96 Las Vegas, NV, USA

The rest of the paper is organized as follows. Section 2 provides basic terminology and introduces the power dissipation and timing model to be used. Our new power reduction method is presented in Section 3; we review ATPG-based structural transformations of the netlist and show their application to reduce the power consumption. Experimental results are given in Section 4.

## 2 Background

**Terminology.** We consider netlists of combinational gates. Each gate of the netlist has a unique label, which is also used to denote the signal at the output of the gate. If the signal at the output of a gate branches out to several fanout gates, we distinguish between the root of the branching signal, called *stem signal*, and the individual branches, called *branch signals*. The output signal of a gate with a single fanout is regarded as a stem signal. We associate an individual label with a branch signal. Gate **t** is said to be a *transitive fanout* of gate **s** if there is a directed path from **s** to **t** in the DAG representing the netlist. The set of all gates that are transitive fanouts of gate **s** is denoted by *TFO*(**s**). A gate **s** is said to *dominate* gate **t** if any path from **t** to a circuit output contains **s**. The set of all gates dominated by gate **s** is refered to as *dominated region* of **s** and is abbreviated by *Dom*(**s**). By *inputs*(*Dom*(**s**)) we denote the set of gates outside of *Dom*(**s**) that have a direct fanout gate within *Dom*(**s**).

**Power Estimation Model.** The amount of energy dissipated by a CMOS gate is dominated by charging and discharging its capacitive load whenever a transition occurs at the gate output. In this paper, we assume a *zero-delay* power estimation model, i.e, we don't consider the contribution of glitches to the power consumption. Glitches typically contribute about 20% to the total power consumption [9]. However, at the logic level glitching is difficult to model accurately. Glitching is caused by unbalanced path lengths, and especially in highly integrated circuits the path lengths strongly depend on wiring delays, which are unknown before the placement and routing step. Under this zero-delay assumption, the power consumed by a CMOS circuit is given by

$$P_{circuit} = \frac{1}{2} V_{dd}^2 \cdot f \cdot \sum_{\mathbf{i} \in signals} C(\mathbf{i}) \cdot E(\mathbf{i}), \qquad (1)$$

where $V_{dd}$ is the supply voltage, and $f$ is the clock frequency. Here, $C(\mathbf{i})$ denotes the total capacitive load driven by signal **i**, and $E(\mathbf{i})$ is the *transition probability* at signal **i**, i.e., the probability that signal **i** makes a $0 \rightarrow 1$ or a $1 \rightarrow 0$ transition from one clock cycle to the next.

At the logic level, the supply voltage $V_{dd}$ is fixed and the clock frequency is given by the performance specification. Therefore, power reduction at the logic level is achieved by reducing $\sum_{\mathbf{i}} C(\mathbf{i}) \cdot E(\mathbf{i})$. The transition probabilities of the signals are part of the cost function and therefore efficient and accurate techniques for their computation are required. Several approaches tackling this problem have been suggested [15].

As in [6, 12], we compute the transition probability of a signal assuming temporal independence of the primary inputs. Thus, the transition probability of a signal **s** is $E(\mathbf{s}) = 2 \cdot p(\mathbf{s})(1 - p(\mathbf{s}))$, where $p(\mathbf{s})$ is the signal probability of **s**. Note that in our approach other estimation methods considering temporal and spatial correlations could also be used.

**Timing Model.** After the technology mapping, the delay of a circuit can be computed based on the gate delays as specified in the library. We use a linear delay model where the delay $D$ of a gate **s** is given by $D(\mathbf{s}) = \tau(\mathbf{s}) + C(\mathbf{s}) \cdot R(\mathbf{s})$. Here, $\tau(\mathbf{s})$ denotes the intrinsic delay through gate **s**, and $C(\mathbf{s}) \cdot R(\mathbf{s})$ accounts for the additional delay due to the fanout load driven by gate **s**. Again, $C(\mathbf{s})$ is the capacitive load at the output of **s** and $R(\mathbf{s})$ is the drive resistance of **s**.

Based on the delay values for individual gates, the arrival and required times of all gates can be computed. The circuit delay is the maximum of the arrival times of all primary outputs.

## 3 Power Reduction by Structural Transformations

### 3.1 Example

An example to illustrate our method for power reduction is shown in Figure 2. Assuming a capacitive load of 1 for an AND-gate input and a load of 2 for an EXOR-gate input, the sum $\sum_{\mathbf{i}} C(\mathbf{i}) \cdot E(\mathbf{i})$ in circuit $A$ evaluates to 1.555. The Boolean function for output **f** in circuit $A$ is given as $\mathbf{f} = \mathbf{d} \cdot \mathbf{b} = (\mathbf{a} \oplus \mathbf{c}) \cdot \mathbf{b}$. In circuit $B$, **f** is given as $\mathbf{f} = \mathbf{g} \cdot \mathbf{b} = ((\mathbf{a} \cdot \mathbf{b}) \oplus \mathbf{c}) \cdot \mathbf{b}$, which - in terms of primary inputs - is equivalent to the Boolean function of **f** in circuit $A$, i.e., the global Boolean function of **f** has not changed. Note that the function of **d** in circuit $A$ is not equivalent to function **g** in circuit $B$. Therefore, without changing the circuits' output functions, circuit $A$ can be modified to circuit $B$ by reconnecting an input branch of the EXOR-gate. In circuit $B$ the sum $\sum_{\mathbf{i}} C(\mathbf{i}) \cdot E(\mathbf{i})$ evaluates to 1.132, which is lower than in circuit $A$.

In this example, the power reduction is caused by two effects. First, by disconnecting the fanout branch from **a** and connecting it to **e** instead, a fanout load was moved towards a signal with lower transition probability. Second, the global Boolean function computed at the output of the EXOR-gate was changed. The transition probability of the new signal **g** is lower than the transition probability of **d**.

### 3.2 Review of Structural Netlist Transformations

In the example of Figure 2, the substitution of a branch signal did not change the function computed at the primary outputs. A transformation that preserves the circuit's input/output behavior is called *permissible*.

**Definition 1** *An* output substitution OS2(**a**,**b**) *substitutes the stem signal* **a** *by signal* **b**. *An* input substitution IS2(**a**,**b**) *substitutes the branch signal* **a** *by signal* **b**. *Output and input substitutions with inverted signal* **b** *are defined analogously [5].*

An input substitution is illustrated in Figure 2. As seen in this example, such substitutions can be permissible although the function computed at the substituting signal (**e**) differs from the function of the substituted signal (**ã**). Provided that the input vectors for which the functions differ belong to the observability don't care set of the substituted signal, the global Boolean functions of the primary outputs will not change. In terms of *permissible functions* [16], a substitution is permissible if the global Boolean function of the substituting signal is contained in the set of permissible functions of the substituted signal.

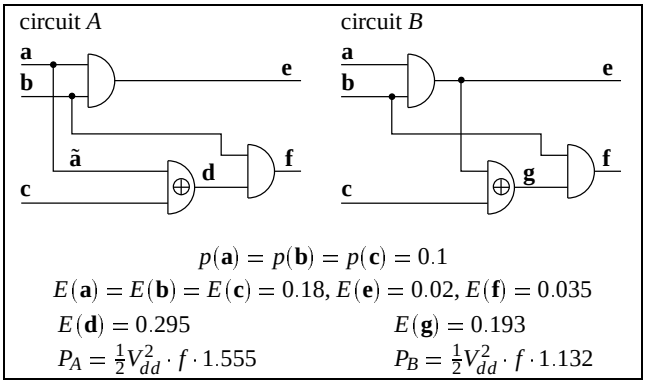


Figure 2: Power reduction by reconnecting a gate input.

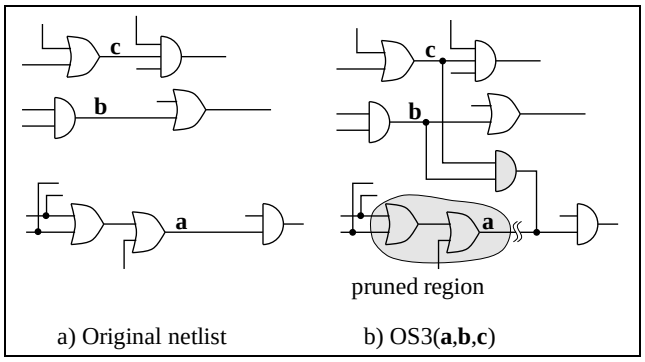


Figure 3: Output substitution OS3(**a**,**b**,**c**) with an AND-gate.

**Definition 2** *An* output substitution OS3$(\mathbf{a},\mathbf{b},\mathbf{c})$ *substitutes the stem signal* **a** *by the output of a new gate driven by the signals* **b** *and* **c**. *Similarly, an* input substitution IS3$(\mathbf{a},\mathbf{b},\mathbf{c})$ *substitutes the branch signal* **a** *by the output of a new gate driven by the signals* **b** *and* **c** *[5].*

An output substitution OS3 by an AND-gate is illustrated in Figure 3. All kinds of two-input gates including EXOR- and EXNOR-gates can be considered for the newly inserted gate. Note that only gates contained in the library are allowed to be added to the netlist. If a gate is not contained in the library, it is decomposed such that all components are contained. An OS3- or an IS3-substitution is permissible if the global Boolean function computed at the output of the new gate is a permissible function of the substituted signal.

The concept of input- and output-substitutions has been shown to be effective in reducing circuit area and circuit delay [16, 2, 5]. Several methods exist to compute permissible substitutions. In [5] substitutions have been related to combinations of *valid clauses*, which in turn can be found by ATPG-based methods. We use these methods to compute permissible substitutions.

### 3.3 Analyzing the Power Reduction of Transformations

In this section we analyze how a structural netlist transformation affects the power consumption of the circuit. Generally, the *power gain PG* of a transformation is given by

$$PG(trans) = \sum_{\mathbf{i} \in signals\ before\ trans} C(\mathbf{i}) \cdot E(\mathbf{i}) - \sum_{\mathbf{i} \in signals\ after\ trans} C(\mathbf{i}) \cdot E(\mathbf{i}). \quad (2)$$

In our approach we often have to deal with a large set of so called *candidate* substitutions, and we want to choose the one with highest *PG* from this set. Directly applying Equation (2) to compute the *PG* for each individual transformation would require large amounts of CPU-time since for each transformation a reestimation of the whole circuit is required. Instead, more efficient methods to estimate the *PG* are needed. The goal is to avoid as much reestimation as possible. The transformations given in Definition 1 and 2 change the circuit structure only incrementally. Therefore, we can expect that many signals are unaffected.

Figure 4 illustrates the power gain contributions in case of an OS2-substitution. The power gain for OS2- and IS2-substitutions is due to the following effects:

A) removing the dominated region of the substituted signal,

B) adding fanouts to the substituting signal,

C) changing the global Boolean function of the signals in the transitive fanout of the substituted signal.

Power reduction due to A) comes along with area reduction and is associated with the removal of physical capacitances. The power gain due to A) is always positive and evaluates to

$$PG_A = \sum_{\mathbf{i} \in Dom(\mathbf{a})} C(\mathbf{i}) \cdot E(\mathbf{i}) + \sum_{\mathbf{i} \in inputs(Dom(\mathbf{a}))} C'(\mathbf{i}) \cdot E(\mathbf{i}), \quad (3)$$

where $C'(\mathbf{i})$ is the portion of $C(\mathbf{i})$ in $Dom(\mathbf{a})$. The first sum in (3) accounts for the pruned gates. The second sum considers the reduced capacitive load for all gates that had a fanout gate in $Dom(\mathbf{a})$. Note that for an input substitution IS2$(\tilde{\mathbf{a}},\mathbf{b})$ the first sum in (3) evaluates to 0. In this case, $PG_A$ is simply $C(\tilde{\mathbf{a}}) \cdot E(\tilde{\mathbf{a}})$. The contribution $PG_A$ can always be computed without any reestimation.

The substituting signal **b** has to drive additional fanouts after the substitution. Therefore, the power gain due to B) is always negative and evaluates to

$$PG_B = -C(\mathbf{a}) \cdot E(\mathbf{b}). \quad (4)$$

To achieve a power reduction, this penalty must be overcompensated by the contributions $PG_A$ and $PG_C$. In the example of Figure 2, we have already seen the overcompensation in case of an input substitution IS2. The contribution $PG_B$ can be computed without reestimation, too.

Generally, the global Boolean function of all gates in the transitive fanout of the substituted signal (except for the primary output gates) may change. Since the signal probability and the transition probability depend on the Boolean function, a reestimation for all signals in this region is necessary. Power gain due to C) is given by

$$PG_C = \sum_{\mathbf{i} \in TFO(\mathbf{a})} C(\mathbf{i}) \cdot \left( E(\mathbf{i})|_{before\ trans} - E(\mathbf{i})|_{after\ trans} \right). \quad (5)$$

The contribution $PG_C$ can be positive or negative. Our experiments have shown that $PG_C$ can dominate the power gain of a substitution. Even if $PG_A + PG_B$ is positive for a substitution (as for almost all OS2-substitutions), a negative contribution by $PG_C$ may increase the circuit power.

The power gain of the 3-signal substitutions OS3 and IS3 can be analyzed in a similar way. In this case, $PG_B$ has to account for the fanout load on the two signals driving the newly inserted gate, and for the newly inserted gate itself.

### 3.4 Considering Timing Constraints

For many of today's designs, tight performance specifications are given. The task of low power synthesis then is to find a power-minimal implementation that satisfies the timing requirements. In this case, trading circuit delay for reduced power consumption is not allowed. Our approach easily incorporates timing constraints. We can predict the effect of each substitution on the circuit delay. There exist two situations where a substitution increases the delay. First, if the arrival time of the substituting signal (or the newly

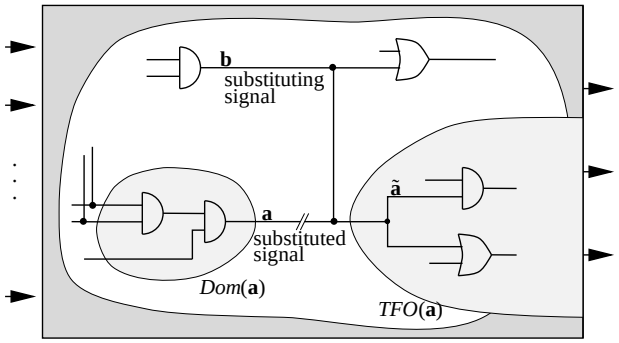


Figure 4: Power gain of a transformation.

```
power_optimize(netlist, repeat, delay_limit)
{
  power_estimate(netlist);
  do
  {
    cand_substitutions = get_candidate_substitutions(netlist);
    while((repeat > 0) ∧ (cand_substitutions ≠ ∅))
    {
      good_subst =
            select_power_red_subst(cand_substitutions, netlist);
      increases_delay = check_delay(good_subst, netlist, delay_limit);
      if (increases_delay) continue;
      allowed = check_candidate(good_subst, netlist);
      if (allowed)
      {
        perform_substitution(good_subst, netlist);
        power_estimate_update(good_subst, netlist);
        repeat = repeat - 1;
      }
    }
  } while(cand_substitutions ≠ ∅)
  return(netlist);
}
```

Figure 5: Algorithm for power optimization.

inserted gate in case of OS3/IS3-substitutions) is larger than the required time of the substituted signal, then the substitution creates a path with a delay larger than the circuit delay before the substitution. Second, there is a delay increase for gates with additional fanouts. By this effect, a previously uncritical path through such a gate can become critical.

Timing constraints are considered by discarding all substitutions that increase the circuit delay. This guarantees that the circuit delay never increases.

### 3.5 Overall Procedure

Our power optimization algorithm is outlined in Figure 5. First, we compute the power consumption of the initial netlist. During this step, the transition probabilities of all gates are stored. Function `get_candidate_substitutions` returns a set of potentially permissible substitutions, called *cand_substitutions*. This is done using techniques based on fault simulation as described in [2, 5].

From the set *cand_substitutions* we select a substitution with positive power gain. For that purpose, we have developed a heuristic based on the results of Section 3.3. First, we pre-select the substitutions with highest $PG_A + PG_B$. This pre-selection is fast since no reestimation is required. Then, we compute $PG_C$ for all pre-selected substitutions by reestimating the transitive fanout. The pre-selected substitution with highest $PG_A + PG_B + PG_C$ is chosen and returned by function `select_power_red_subst`.

Function `check_delay` checks if the chosen substitution satisfies the user-given delay constraint. In case of a violation, we discard the substitution and proceed by searching for the next best substitution. Otherwise we check if the substitution is permissible. Function `check_candidate` involves ATPG and returns TRUE for a permissible substitution and FALSE if the substitution is not permissible or in case that ATPG aborted. After a permissible transformation has been performed, we update the stored transition probabilities in the *TFO*-region of the substituted signal. To increase efficiency, the inner loop is repeated several times (specified by the parameter *repeat*) before a new set of candidate substitutions is computed. If power reducing substitutions are not found anymore, we exit the outer loop and return the optimized netlist.

## 4 Experimental Results

We implemented the described approach in the program POWDER, which is embedded into the synthese tool TOS[TUM]. The initial circuits used in our experiments had been synthesized with existing logic synthesis techniques targeting low power. They were obtained by the program POSE (Power Optimization and Synthesis Environment) developed at USC [17] using the library `lib2.genlib` from [18]. POSE includes logic optimization [6, 7] and technology mapping methods [10] for low power, which significantly reduce power consumption compared with standard techniques. The circuits in Table 1 are ordered according to their initial area. For the initial circuits obtained by POSE, we give the circuit *power* ($\sum_{\mathbf{i}} C(\mathbf{i}), E(\mathbf{i})$), the circuit *area* (total gate area), and the circuit *delay* (ns) in columns 2 to 4. The same signal probabilities for primary inputs were assumed during synthesis with POSE and optimization with POWDER.

### 4.1 Optimization of Low Power Circuits

First, we applied POWDER without delay consideration. The circuit power, the power reduction percentage, and circuit area after running POWDER are shown in columns 5, 6, and 7 of Table 1. We achieve an overall power reduction of 26.1%. This demonstrates the great potential exploited by structural transformations. Individual power reductions range from 4.3% (circuit `i2`) to 79.4% (circuit `t481`). The drastic power reducton for circuit `t481` is caused by a related area decrease of 87%. Note however that in several other cases, e.g. circuit `cps`, huge power reductions but only moderate area reductions are obtained. Circuit area is reduced by 8.9% on average, though there are some circuits with area increase.

We analyzed how the individual classes of substitutions (OS2, IS2, etc.) contribute to power and area reduction, respectively. For that purpose, we individually summed up the power and area savings of the substitutions performed during power optimization. The results, which are summarized in Table 2, show the relative contribution of the classes to the total power reduction. It turned out that IS2-substitutions are most valuable for power optimization, since 36.5% of the overall power reduction is due to this class. The OS2- and OS3-substitutions also significantly contribute to power optimization, whereas IS3-substitutions yield no more than 3.4%. For an IS3-substitution a new gate is added to substitute a single branch signal. Obviously, the power increase due to the new gate can be compensated only in rare cases.

It is interesting to relate the effect of the substitutions on power to their effect on area. The slight overal area reduction of 8.9% is solely caused by OS2-substitutions. All other substitution classes increase circuit area. Thus, some of the area saved by performing OS2-substitutions is lost again by IS2-, OS3-, and IS3-substitutions. Table 2 nicely illustrates that optimization for low-power substantially differs from area optimization. Power reduction may be associated with an area increase or decrease.

### 4.2 Satisfying Timing Constraints

In this experiment, we applied POWDER in its delay constraint mode. We used the initial circuit delay as constraint to be satisfied. Columns 8 to 12 of Table 1 refer to final power, power reduction percentage, area, delay, and CPU-time in seconds on an AlphaStation $250^{4/266}$. The power reduction is 21.4% on average and reaches up to 72.3%. The delay of the circuits decreases by 6.8% on average.

The CPU times of our implementation range from 2 minutes to more than 8 hours for example `apex1`. During the experiments we observed that most of the power reduction is achieved by the first couple of substitutions. Much of the CPU time is spent at the end to achieve negligible power reductions. Therefore, one could terminate the program when the power reduction by the current substitutions is below a threshold. This measure would substantially reduce the CPU times but only slightly degrade the results.

Table 1: Results of POWDER on a set of benchmark circuits.

| circuit | initial | | | POWDER no delay constraints | | | POWDER with delay constraints | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | power | area | delay | power | red.% | area | power | red.% | area | delay | CPU |
| comp | 3.36 | 118784 | 16.6 | 2.29 | 31.8 | 143376 | 2.80 | 16.7 | 126208 | 16.3 | 312 |
| Z5xp1 | 3.61 | 123888 | 32.0 | 2.12 | 41.3 | 89552 | 2.35 | 34.9 | 107648 | 22.8 | 118 |
| clip | 3.99 | 138736 | 23.1 | 2.45 | 38.6 | 107648 | 2.65 | 33.6 | 116928 | 15.9 | 251 |
| frg1 | 3.54 | 144304 | 15.1 | 2.13 | 39.8 | 135488 | 2.72 | 23.2 | 137808 | 15.1 | 357 |
| c8 | 3.54 | 145232 | 18.0 | 2.91 | 17.8 | 155440 | 3.09 | 12.7 | 162400 | 16.7 | 222 |
| term1 | 3.66 | 154048 | 14.0 | 2.81 | 23.2 | 141056 | 2.98 | 18.6 | 143840 | 12.7 | 143 |
| f51m | 3.45 | 158224 | 13.2 | 2.32 | 32.8 | 137344 | 2.44 | 29.3 | 120176 | 12.8 | 239 |
| rd84 | 3.04 | 160080 | 22.8 | 1.75 | 42.4 | 104400 | 1.84 | 39.5 | 107648 | 20.7 | 417 |
| bw | 2.93 | 195344 | 13.8 | 1.83 | 37.5 | 195344 | 1.98 | 32.4 | 187920 | 13.7 | 724 |
| ttt2 | 3.88 | 195344 | 16.8 | 2.62 | 32.5 | 168896 | 2.98 | 23.2 | 192560 | 16.0 | 708 |
| C432 | 5.61 | 204160 | 41.6 | 4.19 | 25.3 | 202768 | 4.30 | 23.4 | 203232 | 41.5 | 1092 |
| i2 | 4.90 | 232928 | 16.2 | 4.69 | 4.3 | 290928 | 4.73 | 3.5 | 278400 | 16.2 | 5425 |
| Z9sym | 7.90 | 238960 | 14.0 | 4.51 | 42.9 | 218080 | 5.31 | 32.8 | 224576 | 14.0 | 1911 |
| apex7 | 6.74 | 249632 | 16.7 | 5.85 | 13.2 | 238496 | 5.98 | 11.3 | 239424 | 16.6 | 614 |
| alu4tl | 4.67 | 262624 | 25.7 | 2.12 | 54.6 | 132240 | 2.50 | 46.5 | 169360 | 20.2 | 246 |
| 9sym | 7.02 | 263552 | 17.2 | 4.32 | 38.5 | 278400 | 5.11 | 27.2 | 258912 | 17.2 | 1549 |
| 9symml | 5.40 | 267728 | 26.6 | 2.60 | 51.9 | 266336 | 2.87 | 46.9 | 289072 | 23.7 | 3891 |
| x1 | 9.29 | 322016 | 12.4 | 8.10 | 12.8 | 341040 | 8.40 | 9.6 | 333616 | 12.4 | 2305 |
| example2 | 6.72 | 356352 | 18.4 | 5.56 | 17.3 | 354032 | 5.64 | 16.1 | 369344 | 17.4 | 2940 |
| ex5 | 8.64 | 358208 | 19.1 | 4.02 | 53.5 | 325264 | 5.18 | 40.0 | 328976 | 15.4 | 3430 |
| alu2 | 6.16 | 362848 | 46.5 | 3.78 | 38.6 | 360528 | 3.84 | 37.7 | 340576 | 46.1 | 10451 |
| x4 | 12.08 | 396256 | 17.9 | 11.11 | 8.0 | 380016 | 11.43 | 5.4 | 386512 | 17.1 | 316 |
| C880 | 10.82 | 405072 | 44.0 | 10.26 | 5.2 | 393008 | 10.51 | 2.9 | 390224 | 44.0 | 2661 |
| C1355 | 14.98 | 438480 | 27.3 | 12.24 | 18.3 | 360528 | 12.51 | 16.5 | 368880 | 27.2 | 1288 |
| duke2 | 6.26 | 444512 | 21.9 | 3.20 | 48.9 | 386976 | 3.40 | 45.7 | 412032 | 21.7 | 6212 |
| pdc | 10.01 | 477920 | 21.1 | 7.25 | 27.6 | 458432 | 7.36 | 26.5 | 471424 | 21.1 | 10470 |
| C1908 | 12.46 | 486272 | 43.6 | 8.69 | 30.3 | 439872 | 9.84 | 21.0 | 437088 | 43.4 | 4786 |
| ex4 | 13.27 | 559584 | 13.8 | 10.66 | 19.7 | 589280 | 11.80 | 11.1 | 554944 | 13.8 | 267 |
| t481 | 5.35 | 702960 | 26.3 | 1.10 | 79.4 | 93264 | 1.48 | 72.3 | 117856 | 14.8 | 428 |
| rot | 17.84 | 710848 | 34.3 | 15.86 | 11.1 | 709920 | 16.58 | 7.1 | 702032 | 32.9 | 939 |
| spla | 9.76 | 735904 | 26.2 | 4.31 | 55.8 | 543808 | 5.04 | 48.4 | 578608 | 22.8 | 8504 |
| vda | 6.02 | 798544 | 24.7 | 4.34 | 27.9 | 750752 | 4.94 | 17.9 | 769312 | 24.3 | 1835 |
| misex3 | 10.79 | 807824 | 36.5 | 4.21 | 61.0 | 627792 | 4.99 | 53.8 | 646816 | 29.8 | 2581 |
| frg2 | 13.40 | 819888 | 39.5 | 11.17 | 16.6 | 805968 | 11.80 | 11.9 | 813856 | 35.5 | 2753 |
| alu4 | 6.92 | 845872 | 54.7 | 4.09 | 40.9 | 836592 | 4.60 | 33.5 | 833808 | 46.4 | 4653 |
| apex6 | 23.55 | 845872 | 17.8 | 20.62 | 12.4 | 804576 | 21.77 | 7.6 | 814784 | 17.0 | 1388 |
| x3 | 23.84 | 901552 | 21.0 | 19.97 | 16.2 | 821280 | 20.19 | 15.3 | 848192 | 19.9 | 2372 |
| apex5 | 9.28 | 945632 | 21.1 | 5.42 | 41.6 | 904336 | 6.14 | 33.8 | 902480 | 21.0 | 3233 |
| dalu | 16.26 | 955840 | 48.3 | 12.61 | 22.4 | 892736 | 14.27 | 12.2 | 912688 | 45.8 | 2224 |
| i8 | 13.95 | 1074624 | 36.3 | 12.61 | 9.6 | 1086224 | 12.94 | 7.2 | 1075552 | 34.0 | 2018 |
| table5 | 11.85 | 1236096 | 29.2 | 4.85 | 59.1 | 1060704 | 5.30 | 55.3 | 1096896 | 27.0 | 7959 |
| cps | 13.25 | 1409632 | 35.5 | 3.77 | 71.5 | 1218000 | 5.07 | 61.7 | 1258368 | 30.9 | 14053 |
| k2 | 8.00 | 1455568 | 34.9 | 4.52 | 43.5 | 1394784 | 5.35 | 33.1 | 1413808 | 34.1 | 13032 |
| C5315 | 55.81 | 1699632 | 38.8 | 45.78 | 18.0 | 1537232 | 47.63 | 14.7 | 1544656 | 39.8 | 7474 |
| apex1 | 21.41 | 1741856 | 36.7 | 9.36 | 56.3 | 1421232 | 10.23 | 52.2 | 1469024 | 33.0 | 30121 |
| pair | 41.26 | 1814704 | 37.8 | 32.35 | 21.6 | 1698240 | 35.39 | 14.2 | 1772944 | 36.8 | 9807 |
| des | 90.73 | 3839600 | 124.5 | 80.69 | 11.1 | 3833568 | 81.17 | 10.5 | 3827536 | 124.5 | 20530 |
| $\sum$ : | 587.20 | 31203536 | 1353.5 | 434.01 | | 28435776 | 461.42 | | 28858944 | 1262.0 | |
| reduction: | | | | 26.1 | | 8.9 | 21.4 | | 7.5 | 6.8 | |

Table 2: Contribution of classes of substitutions to power and area reduction.

| substitution: | OS2 | IS2 | OS3 | IS3 |
|---|---|---|---|---|
| contribution to overall power reduction: | 32.5% | 36.5% | 27.6% | 3.4% |
| contribution to overall area reduction: | 171.5% | -11.6% | -27.7% | -32.2% |

### 4.3 Power-Delay Trade-off

Finally, we explored the power-delay tradeoff that can be achieved with POWDER. For a set of 18 circuits we ran POWDER with various delay constraints. We summed up the resulting power and delay values for all circuits and each delay constraint. The power-delay trade-off is shown in Figure 6. In this diagram, the power of the optimized circuits relative to the initial power is given on the y-axis. The delay relative to the initial delay is shown from left to right. The numbers next to the points refer to the delay constraint that was set (e.g., 20 means that an 20% increase in the circuit delay was allowed). The point for 0% delay increase is left

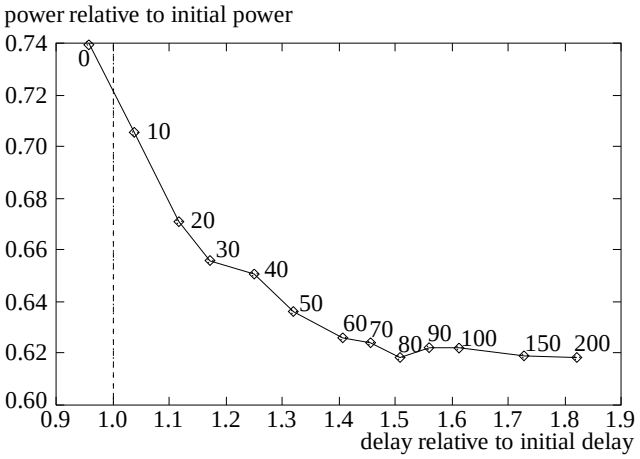


Figure 6: Power-delay tradeoff.

of the dashed line, which refers to the initial circuit delay. Figure 6 illustrates that the circuits produced by POWDER have on average less delay than specified by the constraint, i.e., the given constraint is not always fully exploited. Our algorithm handles delay constraints such that the delay after a substitution will never increase beyond the constraint, however it may be less then the specified constraint.

For the circuits used in this experiment, the overall power reduction ranges from 26% (0% delay constraint) to 38% (200% delay constraint). Two thirds of the additional power reduction exceeding 26% are obtained by a delay increase of only 15% (30% delay constraint). Further power reductions require a significant increase in the circuit delay. For delay constraints beyond 80%, no additional power reduction is achieved.

## 5 Conclusion

We have presented an algorithm that performs structural transformations of mapped netlists to reduce power consumption. Our experimental results demonstrate that power consumption can be significantly reduced in this logic synthesis phase even after previous power-oriented logic optimization and mapping. Thus, the new approach is value-added to existing low-power techniques.

The power reductions achieved by our approach are drastic (over 70%) for some circuits. On average, the circuit power is reduced by 26%. User-specified delay constraints are handled easily. We obtain average power reductions of 21% without degrading the circuit delay. By varying the delay constraints, we achieve a trade-off between circuit power and circuit delay.

## Acknowledgment

The authors are very grateful to Prof. Kurt J. Antreich and Peter Schneider for many valuable discussions and advice. Special thanks are to Sasan Iman from USC who provided the power optimized benchmark circuits used for the experiments.

## References

[1] K.-T. Cheng and L. A. Entrena, "Multi–level logic optimization by redundancy addition and removal," in *European Conference on Design Automation (EDAC)*, pp. 373–377, Feb. 1993.

[2] B. Rohfleisch and F. Brglez, "Introduction of permissible bridges with application to logic optimization after technology mapping," in *European Design and Test Conference (EDAC/ETC/EUROASIC)*, pp. 87–93, Feb. 1994.

[3] S.-C. Chang, K.-T. Cheng, N.-S. Woo, and M. Marek-Sadowska, "Layout driven logic synthesis for FPGAs," in *31st ACM/IEEE Design Automation Conference (DAC)*, pp. 308–313, June 1994.

[4] W. Kunz and P. R. Menon, "Multi-level logic optimization by implication analysis," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 6–13, Nov. 1994.

[5] B. Rohfleisch, B. Wurth, and K. Antreich, "Logic clause analysis for delay optimization," in *32nd ACM/IEEE Design Automation Conference (DAC)*, pp. 668–672, June 1995.

[6] S. Iman and M. Pedram, "Multi-level network optimization for low power," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 372–377, Nov. 1994.

[7] S. Iman and M. Pedram, "Logic extraction and factorization for low power," in *32nd ACM/IEEE Design Automation Conference (DAC)*, pp. 248–253, June 1995.

[8] S. C. Prasad and K. Roy, "Circuit activity driven multilevel logic optimization for low power reliable operation," in *European Conference on Design Automation (EDAC)*, pp. 368–372, Feb. 1993.

[9] A. Shen, A. Gosh, S. Devadas, and K. Keutzer, "On average power dissipation and random pattern testability of cmos combinational logic networks," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 402–407, Nov. 1992.

[10] C.-Y. Tsui, M. Pedram, and A. M. Despain, "Power Efficient Technology Decomposition and Mapping Under an Extended Power Consumption Model," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems CAD*, vol. 13, pp. 1110–1122, Sept. 1994.

[11] R. Panda and F. N. Najm, "Technology decomposition for low-power synthesis," in *IEEE Custom Integrated Circuits Conference (CICC)*, 1995.

[12] V. Tiwari, P. Ashar, and S. Malik, "Technology mapping for low power," in *30th ACM/IEEE Design Automation Conference (DAC)*, pp. 74–79, June 1993.

[13] B. Lin and H. D. Man, "Low-power driven technology mapping under timing contraints," in *IEEE International Conference on Computer Design (ICCD)*, pp. 421–427, Oct. 1993.

[14] R. I. Bahar, G. D. Hachtel, E. Macii, and F. Somenzi, "A symbolic method to reduce power consumption of circuits containing false paths," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 368–371, Nov. 1994.

[15] F. N. Najm, "Feedback, correlation, and delay concerns in the power estimation of VLSI circuits," in *32nd ACM/IEEE Design Automation Conference (DAC)*, pp. 612–617, June 1995.

[16] S. Muroga, Y. Kambayashi, H. C. Lai, and J. N. Culliney, "The transduction method - design of logic networks based on permissible functions," *IEEE Transactions on Computers*, vol. 38, pp. 1404–1424, Oct. 1989.

[17] S. Iman, "personal communication," August 1995.

[18] S. Yang, *Logic Synthesis and Optimization Benchmarks User Guide, Version 3.0*. MCNC, Research Triangle Park, N.C. 27709, 1991.