# Automated Phase Assignment for the Synthesis of Low Power Domino Circuits

Priyadarshan Patra
Strategic CAD Labs, Intel Corporation
JFT-104, 211 NE 25th Avenue
Hillsboro, OR 97124-5961

Unni Narayanan
Design Technology, Intel Corporation
SC12-606, 2200 Mission College Boulevard
Santa Clara, CA 95052-8119

## Abstract

High performance circuit techniques such as domino logic have migrated from the microprocessor world into more mainstream ASIC designs. The problem is that domino logic comes at a heavy cost in terms of total power dissipation. For mobile and portable devices such as laptops and cellular phones, a high power dissipation is an unacceptable price to pay for high performance. Hence, we study synthesis techniques that allow designers to take advantage of the speed of domino circuits while at the same time to minimize total power consumption. Specifically, in this paper we present three results related to automated phase assignment for the synthesis of low power domino circuits: (1) We demonstrate that the choice of phase assignment at the primary outputs of a circuit can significantly impact power dissipation in the domino block (2) We propose a method for efficiently estimating power dissipation in a domino circuit and (3) We apply the method to determine a phase assignment that minimizes power consumption in the final circuit implementation. Preliminary experimental results on a mixture of public domain benchmarks and real industry circuits show potential power savings as high as 34% over the minimum area realization of the logic. Furthermore, the low power synthesized circuits still meet timing constraints.

## 1 Introduction

The advent of portable digital devices such as laptop computers and cellular phones has made low power circuit design an increasingly important research area [4, 13, 14, 12, 6, 11]. For example, laptop computers have a limited battery life, and so the circuitry in the computer must be designed to dissipate as little power as possible without sacrificing performance in terms of speed. Furthermore, simultaneous low power and high performance designs are needed beyond the realm of the microprocessors. For example, ASICs in computer chipsets or cellular phones must also approach microprocessor-like frequency targets, but are constrained by even tighter power budgets [4]. The problem, of course, is that the objectives of low power and high performance are often contradictory. Consider, for example, the use of domino or dynamic logic which is a necessity in high speed designs.

Figure 1 contains a schematic for a basic N-type domino gate.


DAC 99, New Orleans, Louisiana


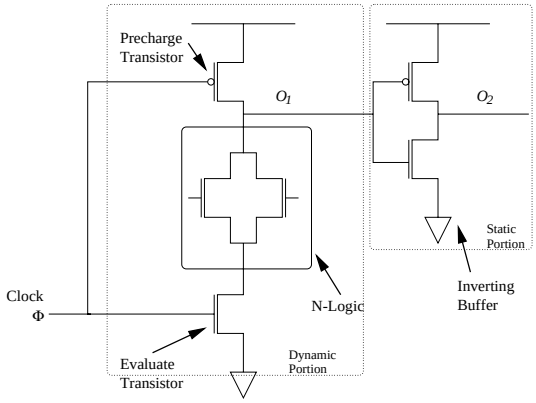


Figure 1: A basic domino gate.

The domino gate consists of a dynamic component and a static component. During the precharge phase (when $\Phi = 0$ ), the output of the dynamic gate (at $O_1$) is precharged high, and the output of the buffer is low. When the gate is evaluated, the output will conditionally discharge and result in the output $O_2$ conditionally becoming high. Observe that domino gates are inherently noninverting. The reason is that a block of domino logic can only function correctly if each gate makes a monotonic "0" to "1" transition. For additional details about domino design the reader is referred to [16, 7, 5, 15].

We note that although domino logic greatly enhances performance, this benefit comes at a high cost in terms of power consumption. Due to clock loading and the precharging every clock cycle, domino gates can consume up to four times the power of an equivalent static gate [16]. However, most efforts in lower power logic synthesis have focused on static CMOS gates [8, 9, 10, 11]. In this paper, we study the problem of low power logic synthesis for domino circuitry. Specifically we present three results: (1) We demonstrate that the output phase assignment affects power dissipation in domino circuits (2) We propose methods for efficiently measuring power consumption in domino circuits (3) We show how these methods can be be incorporated in an algorithm for determining a phase assignment that minimizes power consumption in domino circuits. In Section 2, we discuss some interesting properties about power consumption in domino gates. In Section 3, we explain the problem of phase assignment and show that the choice of output phase can affect power consumption. In Section 4, we present an approach for determining phase assignments for low power domino circuits. In Section 5, we present experimental re-

sults that indicate the promise of our approach. Finally, in Section 6, we propose future directions for this work.

## 2 Power Consumption in Domino Circuitry

Recall that in CMOS technology a large portion of power dissipation on chip is due to dynamic power consumption at the gates which is computed according to the formula:

$$\sum_{i=1}^{N} \frac{1}{2} C_i V_{dd}^2 f_i$$

where $C_i$ is the output capacitance of the $i$th gate, $V_{dd}$ is the supply voltage, $f_i$ is the number of transitions at the output of the $i$th gate, and $N$ is the total number of gates on the chip [4]. Hence power consumption is linearly related to the switching activity $f_i$ of a gate, and clearly a reduction in $f_i$ will lead to a corresponding reduction in the total power consumption of the circuit. We say that the signal probability of a gate is the probability that the logical output of a gate is high and the switching probability of a gate is the probability that the output experiences a transition. We can relate the signal probability and the switching probability of a domino gate such as the one in Figure 1 in the following manner:

**Property 2.1** *Let $p_g$ be the signal probability of logical output $O_2$ of gate $g$. Then $S_g$, the switching probability, at both $O_1$ and $O_2$ is exactly $p_g$.*

**Property 2.2** *Domino gates never glitch or experience spurious transitions at their outputs.*

Property 2.1 becomes apparent if we trace the behavior of a domino gate. Suppose the logical output of of $O_2$ is high. Then, the output $O_1$ must be low. This means that the dynamic portion of the gate discharged the precharged current. Furthermore, the output will need to be precharged during the next clock cycle. Thus, the probability of a transition at $O_1$ is precisely the signal probability at $O_2$. Furthermore, $O_2$'s output experiences a transition if and only if $O_1$ experiences a transition. Hence, the switching probability at $O_2$ is also the signal probability at $O_2$. Now suppose that the logical output of the gate at $O_2$ remains "0". In this case, no charging or discharging takes place anywhere in the gate, and so, no power is dissipated. Property 2.1 is interesting because domino gates, in contrast to static gates, experience an asymmetry in switching activity with respect to signal probability. Figure 2 compares the two types of gates. In Section 3, we will show how this asymmetry can be exploited during the phase assignment portion of domino synthesis in order to reduce power consumption.

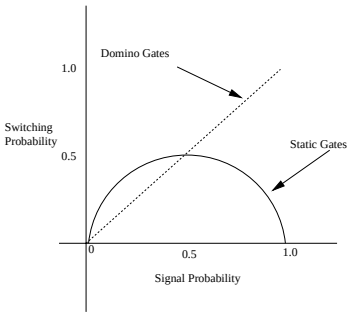


Figure 2: Signal probability and switching for domino and static CMOS logic

Property 2.2 is true because once a gate discharges current, its output cannot be recharged until the next clock cycle. Hence, any glitch that appears at the inputs of a domino block sets a chain of monotonic transitions that cannot be reversed (until the next clock cycle). The consequence is that since domino gates never glitch, the switching activity can be modeled correctly under a zero delay assumption.

## 3 Phase Assignment for Domino Circuits

Recall from Section 1 that domino logic is inherently noninverting. Hence, domino blocks must be synthesized without logical inverters. In [15], Puri proposes the following flow for synthesizing inverter free blocks: (1) Perform a standard technology independent synthesis. Inverters will appear at arbitrary points in this initial realization (2) Systematically remove inverters by changing the phase of primary outputs and applying DeMorgan's Law. In Figure 3, we illustrate an example of this procedure. Suppose we wish to synthesize the following logic functions: $f = \overline{(a+b)+(c \cdot d)}$ and $g = (a+b)+(c \cdot d)$. Referring to Step 1 in Figure 3, we generate an initial synthesis. This realization cannot be implemented in domino logic because of the internal inverters. Hence, we try "changing the phase" of output $g$. We say an output is in *positive phase* if *no* inverter appears at the output boundary. We say an inverter is in *negative phase* if an inverter appears on the output boundary. We note that a "negative phase" assignment does not mean that we are changing the polarity of the output. **In other words, a negative phase assignment does not mean that we are implementing the complement of the original output**. In the initial synthesis, $f$ is implemented in the negative phase, and $g$ is implemented in the positive phase. In Step 2, we change the phase of $g$ (and preserve the logical value of $g$) by placing two "logical" inverters on the output $g$. In Step 3, we push the inverter back, and apply DeMorgan's law to transform the OR gate into an AND gate. Finally, in Step 4, we remove the chained inverters. In general, phase assignment is not as straightforward as this example. Figure 4 illustrates how various phase assignments result in trapped inverters which in turn result in significant logic duplication. The reader is referred to [15] for a detailed discussion on the trapped inverter phenomenon and how it relates to logic duplication.

### Phase Assignment and Switching Activity

We make the key observation that different phase assignments affect the switching activity and hence power in the final domino circuit implementation. Additionally, we show that the phase assignment for minimum area is not necessarily the same as the phase assignment for minimum power. For example, Figure 5 contains circuits corresponding to two different phase assignments. If the primary input signal probabilities are 0.9, we see that the second realization has 75% fewer transitions including the transitions in the static CMOS inverters at the boundaries. This is true even though the second implementation is clearly not the minimum area implementation.

## 4 Major Results

Figure 6 contains a description of our overall approach for power minimization. First, we generate an arbitrary initial phase assignment. We impose this phase assignment on the outputs of the logic network. Next we measure the power. The power measurement steps involve two parts: (1) Partitioning the sequential domino blocks into disjoint combinational blocks (2) Computing the signal probabilities at each node. If we have not exhausted all our candidate phase assignments, we then heuristically generate a new candidate phase assignment based upon the power measurements. If

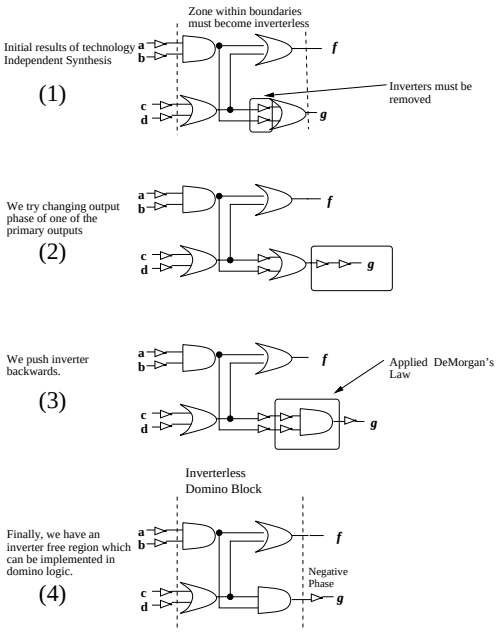


Figure 3: An example of how phase assignment can be used to remove inverters.

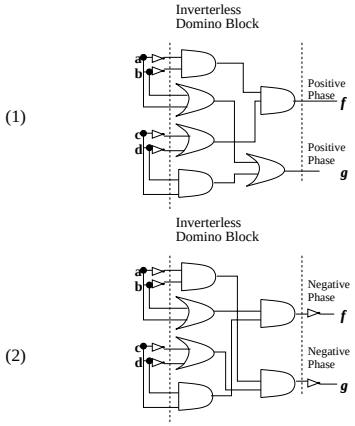


Figure 4: Phase assignments result in logic duplication.

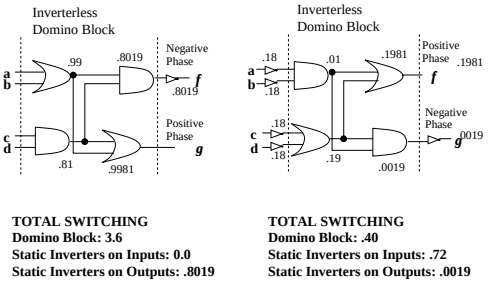


Figure 5: A Comparison of switching in circuits from different phase assignments

we have exhausted all of our candidate phase assignments, then the algorithm terminates. In Section 4.1, we describe our algorithm for determining a candidate phase assignment. Then in Section 4.2, we describe how we compute the power of the domino blocks. Specifically, we describe an enhanced minimum feedback vertex set heuristic that takes advantage of the properties of domino logic blocks to effectively partition sequential blocks into combinational blocks. Finally, we apply a variable ordering heuristic to reduce the complexity of BDD computations.

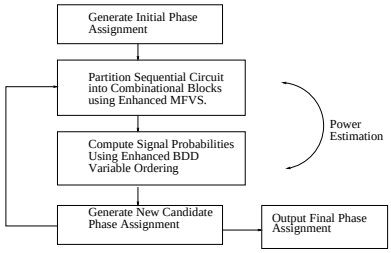


Figure 6: Overall Power Minimization Paradigm.

### 4.1 Generating Candidate Phase Assignments

In this section we describe how we generate a candidate phase assignment. Our heuristic is guided by the following critical observation:

**Property 4.1** *Suppose we change the phase assignment of a primary output. If the signal probability of an individual node in the transitive fanin is p, then its new signal probability will be* $1-p$.

We also note that a particular choice of phase assignment might be globally worse in terms of power because of area duplication. Recall that the area duplication is due to conflicting phase assignments, and this is related to some extent to the degree of total overlap of the transitive fanin cones of the primary outputs [15]. Hence, we define the quantity:

$$O(i,j) = \frac{|D_i \cap D_j|}{|D_i| + |D_j|}$$

where $i$ and $j$ are primary outputs and $D_i$ and $D_j$ are the set of nodes in the transitive fanin of $i$ and $j$ respectively. We call $O(i,j)$ the overlap of primary outputs $i$ and $j$ and it represents the worst possible duplication penalty for incompatible phase assignments $i$ and $j$ respectively. Next, we define the average signal probability for the current phase assignment:

$$A_i = \frac{\sum_{n \in |D_i|} S_n}{|D_i|}$$

where $S_n$ is the signal probability of node $n$. Finally, we can define a cost function for various combinations of phase assignments:

$$\begin{aligned}
K(i^+, j^+) &= |D_i|A_i + |D_j|A_j + 0.5 \cdot O(i,j)(A_i + A_j) \\
K(i^-, j^-) &= |D_i|(1-A_i) + |D_j|(1-A_j) + \\
&\quad 0.5 \cdot O(i,j)((1-A_i) + (1-A_j)) \\
K(i^+, j^-) &= |D_i|A_i + |D_j|(1-A_j) + \\
&\quad 0.5 \cdot O(i,j)(A_i + (1-A_j)) \\
K(i^-, j^+) &= |D_i|(1-A_i) + |D_j|A_j + \\
&\quad 0.5 \cdot O(i,j)((1-A_i) + A_j)
\end{aligned}$$

Observe that Property 4.1 is incorporated into the cost function. We note that $i^+$ refers to retaining the current phase (it does not mean that we are making the output phase positive) and $i^-$ refers to inverting the current phase (it does not mean that we are making the output phase negative). Thus the heuristic for determining a phase assignment that reduces power is: *(1) Generate an arbitrary initial phase assignment for all the primary outputs (2) For each pair of primary outputs determine the cost for each of the four possible phase assignments (3) Choose choose the output pair and phase assignment combination of minimum cost. (4) Synthesisze circuit with that particular phase assignment (5) Measure the power using the techniques described in Section 4.2 (6) If the power decreases with this output pair and phase assignment combination, commit to that combination and remove the pair from the candidate set. Otherwise do not commit to that combination, and still remove pair from candidate set (7) Return to Step 2 if there are still outputs in candidate set* We observe that this heuristic can be extended to capture a greater degree of interaction between phase assignments by extending the definition of the cost function $K$ to more than a pair of outputs. If the cost function is extended to all of the primary outputs in the circuit, the heuristic essentially reduces to a greedily ordered exhaustive search.

### 4.2 Power Computation in Domino Blocks

We estimate the power consumption in a domino block by the following equation:

$$\sum_{i=1}^{N} S_i \times C_i - P_i$$

where $S_i$ is the signal probability at the output of gate $i$, $C_i$ is the load capacitance at the output of gate $i$, and $P_i$ is a penalty for a particular gate type. The quantity $P_i$ arises because we wish to balance the tradeoff between power savings and circuit performance. It is well known that certain logic structures such as domino AND gates are slower than other structures such as domino OR gates. The reason is that AND gates have transistors in series. For aggressive circuit designs, the performance penalty for using an excessive number of AND gates may be too high. Hence, we account for this penalty.

The difficulty lies in estimating the power is the computation of $S_i$ for each node in a sequential domino block. The naive approaches of using exact symbolic simulation or a straightforward application of BDDs quickly break down in terms of computational complexity. This is depsite the fact that domino gate switching behavior can be modeled correctly under a zero delay assumption. Furthermore, the complexity increases due to the iterative nature of the algorithm. Hence, in this section we proposes methods for efficiently measuring the power consumption in a domino block. Specifically in Section 4.2.1 we propose a method for partitioning a sequential circuit into combinational blocks in order to simplify power estimation. Additionally, in Section 4.2.2 we propose a method for ordering the variables in the BDD signal probability computation.

#### 4.2.1 Minimum Feedback Vertex Set Partitioning

Sequential circuits contain cycles. Hence, the resulting state explosion makes it computationally expensive to compute the exact signal probability of each circuit nodes. One heuristic for reducing the computational complexity of computing signal probabilities at the nodes is to partition the circuit into combinational blocks at the expense of accuracy. For example, Figure 7 illustrates a preferred partitioning of the circuit. The reason is that this partitioning results in a combinational block with fewer primary inputs. Thus, the problem of computing the signal probability at an internal node is greatly simplified.

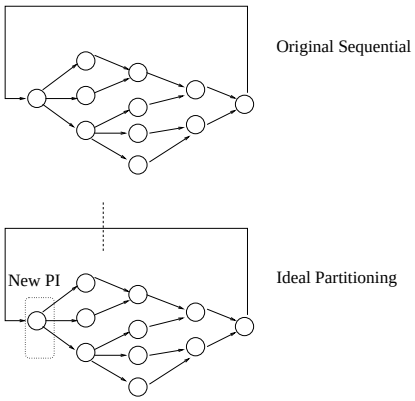


Figure 7: A sequential circuit with various partitions.

Ideally, we would compute the minimum feedback vertex set to generate the partitioning. Unfortunately, the MFVS problem is NP-Complete. However, there are several well known heuristics used in the testing domain that are potentially applicable for approximating the MFVS [2]. These techniques use the concept of transforming a sequential network into an *s-graph*. An *s-graph* is a directed graph representing structural dependencies (edges) among flip-flops (vertices). Figure 8 shows the graph transformations used in previous work.

**Enhanced Minimum Feedback Vertex Sets for Domino**

We make the observation that the process of phase assignment inevitably leads to some logic duplication. In fact, the logic duplication in domino can be quite high when compared to the equivalent static blocks [15]. This means that there are many nodes in a domino block that share common fanins and fanouts. Thus the corresponding *s-graph* contains nodes (flip-flops in the original circuit) that share common fanins and fanouts.

Hence, we propose a fourth transformation which is illustrated in Figure 9. In that figure, the s-graph is strongly connected and none of the original transformations illustrated in Figure 8 can further reduce it. However our fourth transformation, known as a *symmetry* based transformation, groups vertices which have identical fanins and identical fanouts into a *weighted supervertex*. Thus, the vertices $A, B$ and $E$ form supervertex $ABE$ with weight 3, and vertices $C$ and $D$ combine to form supervertex $CD$ with weight 2. The MFVS algorithm of [2] is then applied to this transformed s-graph of supervertices with the modification that the supervertices in an s-graph should be processed in descending order of their weights.

#### 4.2.2 Variable Ordering for BDDs

Once we have split the loops in the sequential circuit to form a combinational structure we are ready to use BDDs [1] to compute the signal probability at each circuit node [3, 14]. We can greatly reduce the complexity of BDD computations by maximizing sharing of nodes in the ROBDD. Domino blocks have the following properties that allow us to maximize BDD node sharing: (1) The circuits are highly flattened and a node's average fanout is high, (2) The overall circuit is highly convergent – nodes near the primary inputs have a greater number of fanouts than nodes near the primary outputs, (3) Most signals in a block of control domino logic feed gates at the *same topological level* in the circuit. Hence there is a heavy overlap of logic cones in the domino implementation. We observe

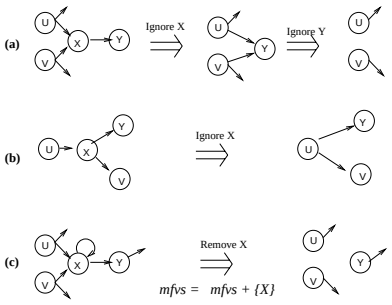


Figure 8: Transformations to generate MFVS.

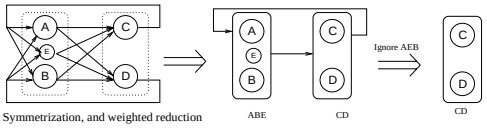


Figure 9: A new transformation to generate MFVS

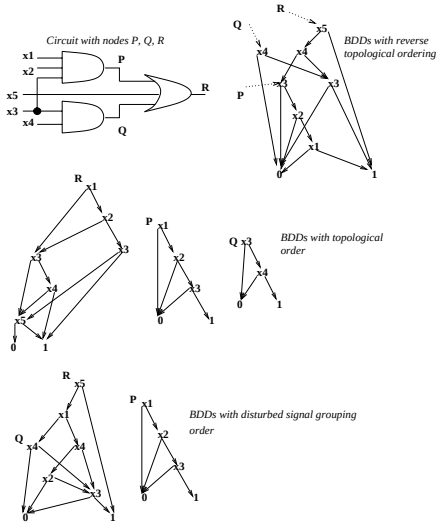


Figure 10: BDD ordering heuristic

that with a careful variable ordering we can exploit this overlap of logic cones to generate more compact BDD representations We order the BDD variables according to two principles: (1) Variables are ordered in the *reverse* of the order that the circuit inputs are *first visited* when the gates are *topologically* traversed, (2) Gates that are at the same topological level are traversed in the decreasing order of the cardinality of their fanout cones. These principles heuristically insure that a variables takes a lower position in the BDD ordering if it is near the primary inputs or has large fanout cones.

Consider the example in Figure 10 where a circuit with nodes $P, Q$ and $R$ is depicted. The two possible topological orders for visiting the gates are $P, Q, R$ and $Q, P, R$. Let's focus on the first order which implies that (primary) inputs $x1, x2, x3$ are used first and then $x4$, and finally $x5$ are used. We let the input names stand for the variables in the BDDs, which we construct for all (non input) circuit nodes, namely, $P, Q$, and $R$. According to our previous observations, the *initial* BDD ordering should be $x5, x4, x3, x2, x1$. In Figure 10 this ordering corresponds to the first row of BDDs. Note that it requires only 7 non-leaf BDD nodes to represent all the circuit nodes. In contrast, the second row of BDDs are obtained when the topological ordering $x1, x2, x3, x4, x5$ is used. This requires 11 BDD nodes. Finally, the bottom row shows the BDDs obtained if the "natural" grouping is violated and the primary inputs are arbitrarily combined. In this case the ordering is $x5, x1, x4, x3, x2$. The last BDD variable ordering, which requires 9 non-leaf BDD nodes, has the variable $x1$ "unnaturally sandwiched" between $x5$ and $x4$. In practice, the circuit nodes have much larger fanouts and convergence; thus, in practice, our heuristic is actually much more effective than what is depicted here.

## 5 Experimental Results

Our experimental framework was an in-house domino synthesis system and flow. We implemented the heuristic for efficiently computing signal probabilities for domino circuits and the algorithm for determining the phase assignment that results in minimum switching activity. In our optimization objective function, for each gate $i$ we set the penalty $P_i = 0$ and the capacitance $C_i = 1$. Hence, we effectively determined the phase assignment that minimized the total switching activity. We measured the power using the EPIC PowerMill circuit simulator which accounts for accurate delays and capacitances. Table 1 and Table 2 contain our experimental results for a mixture of internal proprietary control logic blocks and public benchmark circuits.

Table 1 contains synthesis results and power measurements when the signal probabilities assigned to the primary inputs were respectively 0.5 (different signal probabilities yielded similar results). We used the following flow: (1) We performed technology independent minimization (2) We independently applied either the minimum area phase assignment algorithm (identical to [15]) or the minimum power phase assignment algorithm (depending upon the experiment) (3) We performed technology mapping (to a proprietary cell library) (4) We measured the power using the Epic PowerMill tool with statistically generated input vectors with the appropriate signal probabilities. The results under the column "MA" refer to the **OPTIMAL** synthesis for minimum area with the algorithm specified by [15]. Similarly, the results under the column "MP" refer to the synthesis for minimum power based upon our heuristic. The power is reported in terms of total capacitive current, short circuit current, and leakage current (mAmps). The area is reported in terms of the total number of standard cells.

We note some interesting points. First, the average power savings is 18%. Second, the benchmark circuit *frg1* yields several insights about our algorithm: (1) The circuit *frg1* has only three primary outputs. Hence, there are only $2^3$ or 8 possible phase assignments. Despite this severely limited search space, the power savings of 34% that we achieved is quite high. (2) The area overhead for this particular synthesis was 48%. This example conclusively shows that the minimum area phase assignment and minimum power phase assignment are quite different.

Table 2 contain the results for the circuits that were run through the same flow with an additional step of transistor resizing (after technology mapping) in order to meet realistic timing constraints. This set of experiments is interesting because the power and area optimizations can potentially be "undone" by subsequent timing optimizations. We note that the power based phase assignment appears to be quite robust with an average power savings of 35%. Furthermore, the area penalty still is reasonably small, and in fact, there is a power optimized circuit that has a smaller area than the area optimized circuits.

| Ckt | Desc. | # PIs | # POs | MA | | MP | | % Area Pen. | % Pwr Sav. |
|---|---|---|---|---|---|---|---|---|---|
| | | | | Size | Pwr | Size | Pwr | | |
| Industry 1 | Control Logic | 127 | 122 | 1849 | 12.47 | 1970 | 9.65 | 6.5 | 22.6 |
| Industry 2 | Control Logic | 97 | 86 | 2272 | 13.74 | 2348 | 14.13 | 3.3 | -2.8 |
| Industry 3 | Control Logic | 117 | 199 | 1589 | 11.77 | 1699 | 8.56 | 6.9 | 27.3 |
| apex7 | Public Domain | 79 | 36 | 394 | 3.71 | 443 | 2.98 | 12.4 | 19.5 |
| frg1 | Public Domain | 31 | 3 | 98 | 1.30 | 145 | 0.86 | 48.0 | 34.1 |
| x1 | Public Domain | 87 | 28 | 404 | 2.57 | 421 | 2.34 | 4.2 | 8.9 |
| x3 | Public Domain | 235 | 99 | 1372 | 7.49 | 1390 | 6.25 | 1.3 | 16.6 |
| *Average* | | | | | | | | 11.8 | 18.0 |

Table 1: Synthesis when signal probabilities of primary inputs were 0.5

| Ckt | Desc. | # PIs | # POs | MA | | MP | | % Area Pen. | % Pwr Sav. |
|---|---|---|---|---|---|---|---|---|---|
| | | | | Size | Pwr | Size | Pwr | | |
| apex7 | Public Domain | 79 | 36 | 452 | 3.72 | 485 | 3.04 | 7.3 | 18.3 |
| frg1 | Public Domain | 31 | 3 | 98 | 3.20 | 147 | 1.91 | 50 | 40.3 |
| x1 | Public Domain | 87 | 28 | 406 | 7.67 | 433 | 6.10 | 6,7 | 20.5 |
| x3 | Public Domain | 235 | 99 | 2005 | 70.13 | 1601 | 26.61 | -20.0 | 62.0 |
| *Average* | | | | | | | | 8.6 | 35.3 |

Table 2: Timed synthesis when signal probabilities of primary inputs were 0.5

## 6 Conclusions

In this paper, we have studied the problem of low power logic synthesis for domino circuits. We have shown that the choice of output phase assignment can dramatically affect power consumption in a block of domino logic. Furthermore, we have presented heuristics for determining a phase assignment that minimizes power consumption in sequential domino blocks. Finally, we presented experimental results on variety of public benchmarks and industry circuits that show these techniques can be beneficial in practice. One promising direction for future work is in the area of integrating the choice of phase assignment with timing optimization. We believe that such a combination will lead to even greater power savings.

## Acknowledgements

We would like to thank Barbara Chappell, Rony Friedman, Jeff Parkhurst, Rob Roy, Prashant Sawkar, Prashant Saxena, Carl Seger, Naresh Sehgal, George Stamoulis, Xinning Wang, Tamar Yehoshua, and the team for their valuable feedback about this research.

## References

[1] R. Bryant. Graph-based algorithms for boolean manipulation. *IEEE Transactions on Computers*, C-35(8):677–691, 1986.

[2] S. T. Chakradhar, A. Balakrishnan, and V. D. Agrawal. An exact algorithm for selecting partial scan flip-flops. In *Design Automation Conference*, pages 81–86, 1994.

[3] S. Chakravarty. On the complexity of using bdds for the synthesis and analysis of boolean circuits. In *Allerton Conference on Communication, Control and Computing*, pages 730–739, 1989.

[4] A. Chandrakasan and R. Broderson. *Low Power Digital CMOS Design*. Kluwer Academic Publishers, 1995.

[5] H. Y. Chen and S. M. Kang. Performance optimization for domino cmos circuit modules. In *ICCD*, pages 522–525, 1997.

[6] J. C. Costa, J. Monteiro, and S. Devadas. Switching activity estimation using limited depth reconvergent path analysis. In *International Symposium on low power electronics and design*, pages 184–189, 1997.

[7] S. M. Kang. Data shifting and rotating apparatus. US Patent 4,396,994, August 1983.

[8] U. K. Narayanan, H. Leong, K. Chung, and C. L. Liu. Low power multiplexer decomposition. In *International Symposium on low power electronics and design*, pages 269–274, 1997.

[9] U. K. Narayanan and C. L. Liu. Low power logic synthesis for xor based circuits. In *International Conference on Computer-Aided Design*, 1997.

[10] U. K. Narayanan, P. Pan, and C. L. Liu. Low power logic synthesis under a general delay model. In *International Symposium on low power electronics and design*, 1998.

[11] R. Panda and F. Najm. Technology decomposition for low-power synthesis. In *IEEE Custom Integrated Circuits Conference*, pages 627–630, 1995.

[12] P. Patra. *Approaches to Design of Circuits for Low-Power Computation*. PhD thesis, The University of Texas at Austin, 1995.

[13] P. Patra and D. Fussell. Power-efficient delay-insensitive codes for data transmission. In *Proc. of 28th Hawaii International Conference on System Sciences*, Jan 1995.

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

[15] R. Puri, A. Bjorksten, and T. Rosser. Logic optimization by output phase assignment in dynamic logic synthesis. In *International Conference on Computer Aided Design*, pages 2–8, 1996.

[16] N. Weste and K. Eshraghian. *Principles of CMOS VLSI Design: A Systems Perspective*. Addison-Wesley, 1993.