# Timing Optimization of Mixed Static and Domino Logic

Min Zhao Sachin S. Sapatnekar
Department of Electrical and Computer Engineering
University of Minnesota, 200 Union Street SE, Minneapolis 55455, USA.
contact: sachin@ece.umn.edu

## Abstract

*A timing optimization algorithm dealing with circuits containing mixed domino and static logic is described. Transistor-level node timing constraints of domino logic is described. The optimization procedure preserves the requirements of maintaining adequate noise margins by constraining the sizing procedure. After sizing, charge-sharing problems are identified with a new method and rectified.*

## 1 Introduction

Domino logic is one of the most effective circuit configurations for implementing high speed logic designs. Domino logic has the advantage of small area, fast operation and low power. However, it has drawbacks which include an inherently non-inverting nature, strict timing constraints, charge sharing and noise susceptibility. All of these factors have restricted applications of domino logic to the timing-critical regions of high-performance designs. However, recently there has been a vast amount of interest in using this logic style. The goal of the paper is to present a tool that performs optimal sizing for mixed domino and static logic circuits.

In this paper, we provide a treatment that considers the domino circuit as combinational logic embedded in a sequential circuit. The analysis technique developed here shows that techniques similar to the static combinational circuit analysis can be applied to domino circuits. Given a circuit consisting of flip-flops, domino gates and static gates, the circuit segment between flip-flops is considered here, and is optimized subject to timing constraints. This circuit segment may consist of static, domino or mixed logic.

Although several sizing algorithms have been published in the past (a survey is provided in [1]), most of them have not considered domino logic. The work by Chen and Kang in [2] and by Wurtz [3] perform sizing for domino circuits, treating them as combinational circuits. However, both techniques perform local optimizations, optimizing only one domino block at a time. This work extends the timing analysis technique used in Venkat *et al.* [4] and van Campenhout *et al.* [5] to identify timing problems, and develops a sizing technique to rectify them.

[1]This work was supported in part by the National Science Foundation under award MIP-9502556 and a gift from Intel Corp.

## 2 Domino Logic Timing Constraints

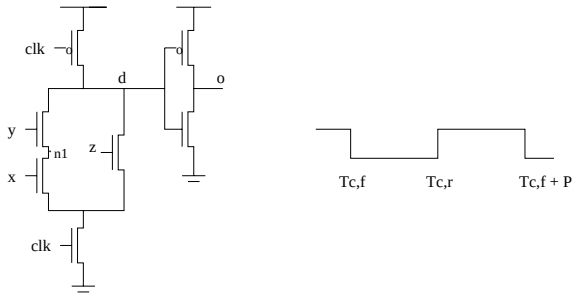


Figure 1: A Typical Domino Circuit

A representative domino gate configuration is shown in Figure 1. When the clock input is low, the gate precharges, charging the dynamic node $d$ to logic 1. In the next half-cycle of the clock when it goes high, the domino gate evaluates, i.e., the dynamic node either discharges or retains the precharged state, depending on the values of the input signals. The two-step mode of operation with a precharge and an evaluate phase causes the timing relationships in domino logic to be more complex than those for static logic.

We list the node timing constraints for domino logic as follows in terms of the signal arrival times and the clock arrival time. In case of multiple clocks for the domino logic, the clock signal $c$ should be set to be the clock signal that feeds the gate that is currently under consideration.

(i) Any falling event at a data input should meet the setup-time requirement to the rising edge of the evaluate clock. If $T_f(in)$ refers to the falling event time of the input node, then we require that

$$T_f(in) \leq T_{c,r} - T_{setup} \quad (1)$$

where the setup time $T_{setup}$ is a constant that acts as a safety margin.
(ii) The rising event of the output node of the domino gate must be completed before the falling edge of evaluate clock. If $T_r(out)$ refers to the rising event time at the output node, then the circuit operates correctly only if

$$T_r(out) \leq T_{c,f} + P \tag{2}$$

In other words, before the beginning of the precharge for next cycle, the correct evaluation result must have traveled to the output node.

For example, in Figure 1, the rising event at the output node $o$ of the domino gate must satisfy (2). Since we can write

$$T_r(o) = \max((T_r(x) + D_f(x,d), T_r(y) + D_f(y,d), T_r(z) + D_f(z,d), T_{c,r} + D_f(c,d))) + D_r(d,o) \tag{3}$$

where $T_r(x)$, $T_r(y)$, $T_r(z)$ are the rising event times at inputs $x, y$ and $z$, respectively, $D_f(i,d)$ represents the delay of a falling transition at the dynamic node $d$ due to a rising transition at input $i \in \{x,y,z\}$, and $D_r(d,o)$ represents the rise delay of the inverter feeding the gate output node $o$. Therefore for $i \in \{x,y,z\}$, we get

$$D_f(i,d) + D_r(d,o) - P \leq T_{c,f} - T_r(i) \tag{4}$$
$$D_f(i,d) + D_r(d,o) - P \leq T_{c,f} - T_{c,r} \tag{5}$$

(iii) The rising event $d$ of the domino gate must be completed before the rising edge of the evaluation clock, i.e.,

$$T_r(d) \leq T_{c,r} \tag{6}$$

If we denote the rise time of the dynamic node through the $p$-transistor fed by the clock as $D_r(c,d)$, then the rising event time can be expressed as:

$$T_r(d) = T_{c,f} + D_r(c,d) \tag{7}$$

This leads us to the constraint given by

$$D_r(c,d) \leq T_{c,r} - T_{c,f} \tag{8}$$

This constraint implies that the pulse width of precharge clock must be capable of pulling up the output node.

# 3 Sizing Algorithm

## 3.1 Overview

The problem is solved in two phases. In the first phase, the sizing problem is solved subject to timing and noise margin constraints. In the second phase, any charge-sharing problems that were created as a result of the sizing procedure are resolved. The sizing problem is formally stated as follows:

$$\begin{aligned}
&\text{minimize} && Area && (9)\\
&\text{subject to} &&&&\\
&\max(T_r(o), T_f(o)) \leq T_{spec} && \forall o \in PO &&\\
&T_f(in) \leq T_{c,r} - T_{setup} && \forall in \in I_{domino} &&\\
&T_r(out) \leq T_{c,f} + P && \forall out \in O_{domino} &&\\
&T_r(d) \leq T_{c,r} && \forall d \in D_{domino} &&\\
&K_1 \leq \frac{W_p}{W_n} \leq K_2 && \forall \text{ gates in the circuit.} &&
\end{aligned}$$

where *Area* is the area of the circuit and, as in other work on transistor sizing [6], is approximated as a sum of transistor sizes, $PO$ is the set of primary outputs, $I_{domino}$, $O_{domino}$ and $D_{domino}$ are, respectively, the set of inputs, outputs dynamic nodes of all of the domino gates in the circuit.

## 3.2 Timing Analysis

The timing analysis procedure described here is based on the PERT procedure and uses an table-lookup delay model for delay calculation. The rising and falling event arrival times for each node $v$ are calculated as follows:

$$T_r(v) = \max(T_f(u) + D_r(u,v)) \tag{10}$$
$$T_f(v) = \max(T_r(u) + D_f(u,v) \tag{11}$$

where $T_r(u)$ and $T_f(u)$ are, respectively, the rising and falling event times for nodes $u$ and $v$, and $D_f(u,v)$, $D_r(u,v)$ are, respectively, the worst fall delay and rise delay from input $u$ to output $v$.

The domino clock input node is treated in the same way as any primary input node, and the rising or the falling edge of the clock provide the corresponding event times for the clock node. The rising and falling event arrival times at the output node of a domino gate can be obtained similarly to the static gate arrival time computations, using (10) and (11). The only difference is that the rising event at the dynamic node is related only to the falling edge of the domino clock and is independent of the other input nodes. The constraint graph is modified to capture the fact that (unlike static gates) the falling transition at an input node cannot influence the rising transition at the dynamic node by setting the value of $D_r$ from each input node of the domino gate to the output node as $-\infty$ .

## 3.3 Sizing Algorithm

The sizing algorithm used here is an adaptation of the TILOS algorithm [6]. Beginning with a circuit

where all transistors are minimum-sized, each iteration selects one transistor and increases its size by a constant factor.

In each iteration, a timing analysis is performed to identify the constraint $g(\mathbf{w}) \leq 0$ with the largest violation, where $g(\mathbf{w})$ denotes the fact that the constraint $g$ is a function of the vector $\mathbf{w}$ of transistor widths. The traceback procedure described above is used to determine the critical path of the circuit, which corresponds to that constraint. The sensitivity of the constraint function $g$ to each transistor width is computed, and the width of the transistor with the most negative sensitivity is increased. The iterations continue until the timing specifications are all met, or until no further improvement is possible.

### 3.4 Noise Margins

Noise margin constraints are applicable to both static and dynamic gates. In [2], Chen and Kang describe a technique for deriving bounds $K_1$ and $K_2$ on $W_p/W_n$ that will ensure that noise margin constraints are satisfied:

$$K_1 \leq Ratio = W_p/W_n \leq K_2 \tag{12}$$

For an inverter, it is a simple matter to verify whether *Ratio* satisfies the specified bounds or not. For complex gates, each domino gate is reduced to an equivalent inverter corresponding to the largest and smallest value of *Ratio*. During the sizing process, these are compared with $K_1$ and $K_2$, respectively, to ensure that during the sizing process, these bounds are not violated.

In other words, the constraint above corresponds to the following two constraints that are always maintained during sizing:

$$Ratio_{min} = W_{p(min)}/W_{n(max)} \geq K_1$$
$$Ratio_{max} = W_{p(max)}/W_{n(min)} \leq K_2$$

The value $W_{p(max)}(W_{n(max)})$ corresponds to the equivalent inverter width when all pmos (nmos) transistors in the complex gate are on, and $W_{p(min)}$ ($W_{n(min)}$) is the equivalent inverter width when only the largest resistive path [1] of the complex gate is on.

## 4 Charge Sharing Algorithm

### 4.1 Estimation of the Worst-Case

Charge-sharing noise is produced by charge redistribution between a dynamic evaluation node and internal nodes within the gate. The usual way [2, 7, 8] of estimating worst case charge sharing is as follows. During the precharge stage, the uppermost device of every $n$-stack is assumed to be off, so that only the capacitance at the dynamic node, $C_d$ is precharged. In the evaluate stage, the bottommost devices in the $n$-stack are configured to be off, and all devices above these in the $n$-stack are assumed to be on. The total capacitance that now shares charge with the dynamic output node is $C_d + C$, where $C$ is the sum of all internal node capacitors.

However, this may be too pessimistic. If the worst-case arrival time for each input is known, and if we can identify a node $n$ such that there is a path from the dynamic node $d$ to $n$ on which the rise transition on all transistors is guaranteed to arrive sufficiently before time $T_{c,r}$, then node $n$ will be precharged and will not trigger charge-sharing. If $C_{pre}$ is the total capacitance of all such nodes $n$, then we can arrive at a less conservative estimate of charge sharing that states that

$$V_{worst} = V_{dd} \cdot \frac{C_d + C_{pre}}{C_d + C} \tag{13}$$

The calculation of $C_{pre}$ is illustrated by the example of Figure 2, which is taken from a fast adder [9]. The value of $C$ is $C_1+C_2+C_3+C_4+C_5+C_6+C_7+C_8$. If we know that signal a4 arrives before $T_{c,r}$ and that the arrival of a3, b3 is later than $T_{c,r}$, then we know $C_1$, $C_3$ should be precharged and $C_2$ may not be precharged. Therefore, the value of $C_{pre}$ in (13) is $C_1 + C_3$. If instead, b3 were to arrive before $T_{c,r}$, then $C_2$ can also be added to $C_{pre}$ and this would correspond to a smaller value of $V_{worst}$.

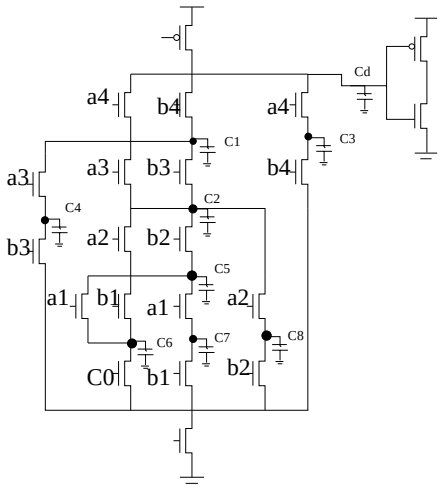


Figure 2: An Example for Charge Sharing

### 4.2 Reducing Charge Sharing

We note that if two nodes are connected by a transistor whose input arrives before evaluation, then

precharging one node will also precharge the other node; otherwise, then precharging one node would not precharge the other. We refer to any set of such nodes as a *channel-connected precharge set*. Our algorithm finds the channel connected precharge set with the largest total capacitance and connects a pmos transistor to a node in that set. This procedure requires one traversal of the graph representing the channel-connected component [1]. This total capacitance is then added to $C_{pre}$ and the worst case voltage due to charge sharing is calculated.

## 5 Experimental Results

The CAD tool is implemented in C++, and takes an input in the form of a transistor netlist. The constraints applied on the circuits include specifications on the clocks, output timing specifications, technology parameters, upper and lower bounds on the size of each transistor and constraints on the worst-case voltage $V_{worst}$ due to charge sharing .

A summary of the results on several sample circuits is shown in Table 1. For each circuit, the number of transistors $|T|$ is listed. For various specifications on the output arrival time, $T_{spec}$ (in ns) and for various domino clock specifications listed in the "Clk" column, the results of sizing are listed. The clock specification is in the format $(T_{c,r}, T_{c,f}, T_{c,r} + P)$, as in Figure 1, where all numbers are specified in ns. The area is reported as "-" if the specifications are too tight to be satisfied.

Table 1: Results of Transistor Sizing

| Circuit/ Unsized Area | $T_{spec}$ (ns) | Clk (ns) | Optimized Area | CPU Time |
|---|---|---|---|---|
| test2/ 28 | 0.39 | (0,0.15,0.39) | 47 | 0.09s |
| | 0.36 | (0,0.15,0.36) | 71 | 0.10s |
| | 0.35 | (0,0.11,0.35) | 52 | 0.09s |
| | 0.32 | (0,0.11,0.32) | 93 | 0.11s |
| adder2/ 160 | 0.55 | (-0.2,0,0.55) | 177 | 0.18s |
| | 0.45 | (-0.2,0,0.45) | 222 | 0.55s |
| | 0.40 | (-0.2,0,0.45) | 317 | 0.81s |
| | 0.38 | (-0.2,0,0.38) | - | 0.77s |

The results of the application of the charge sharing algorithm on the example in Figure 2 is illustrated in Table 2. The set of signals that arrive before time $T_{cr}$ are listed as the early signals. The second column shows the parameter $\frac{V_{worst}}{V_{dd}}$, which is defined in (13). The node at which a pmos transistor is added is defined in column Pn, and the updated value of $\frac{V_{worst}}{V_{dd}}$ is listed after the addition of the first pmos and the second pmos, respectively.

Table 2: Charge Sharing Algorithm Result

| # N($< T_{cr}$) | Ratio | 1st time | | 2nd time | |
|---|---|---|---|---|---|
| | | Pn | Ratio | Pn | Ratio |
| NULL | 0.210 | c5 | 0.361 | c2 | 0.514 |
| a1,b1,a3,b3,a4,b4 | 0.635 | c5 | 0.939 | c8 | 1 |
| a1,b1,a2,b2,a3,b3 | 0.210 | c1 | 0.939 | c3 | 1 |
| a1,b2,a3,a4,b4 | 0.939 | c8 | 1 | - | - |
| a1,a2,a3,a4 | 1 | - | - | - | - |

## References


[1] S. S. Sapatnekar and S. M. Kang, *Design automation for timing-driven layout synthesis*. Boston, MA: Kluwer Academic Publishers, 1993.

[2] H. Y. Chen and S. M. Kang, "A new circuit optimization technique for high performance CMOS circuits," *IEEE Transactions on Computer-Aided Design*, vol. 10, pp. 670–676, May 1991.

[3] L. T. Wurtz, "An efficient scaling procedure for domino CMOS logic," *IEEE Journal of Solid-State Circuits*, vol. 28, pp. 979–982, Sept. 1993.

[4] K. Venkat, L. Chen, I. Lin, P. Mistry, and P. Madhani, "Timing verification of dynamic circuits," *IEEE Journal of Solid-State Circuits*, vol. 31, pp. 452–455, Mar. 1996.

[5] D. van Campenhout, T. Mudge, and K. A. Sakallah, "Timing verification of sequential domino circuits," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pp. 127–132, 1996.

[6] J. P. Fishburn and A. E. Dunlop, "TILOS: A posynomial programming approach to transistor sizing," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pp. 326–328, 1985.

[7] K. Venkat, L. Chen, I. Lin, P. Mistry, P. Madhani, and K. Sato, "Timing verification of dynamic circuits," in *Proceedings of the IEEE Custom Integrated Circuits Conference*, pp. 271–274, 1995.

[8] K. L. Shepard and V. Narayanan, "Noise in deep submicron digital design," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pp. 524–531, 1996.

[9] Z. Wang, G. A. Jullien, W. C. Miller, J. Wang, and S. S. Bizzan, "Fast adders using enhanced multiple-output domino logic," *IEEE Journal of Solid-State Circuits*, vol. 32, pp. 206–213, Feb. 1997.