Monotonic Static CMOS and Dual V_T Technology

Tyler Thorp, Gin Yee and Carl Sechen
Department of Electrical Engineering
University of Washington, Seattle, WA 98195
{thorp,gsyee,sechen}@twolf.ee.washington.edu

Abstract

We developed a methodology and tools for synthesizing monotonic static CMOS networks, which consist of alternating low-skewed and high-skewed static gates. When used with a dual V_T process, monotonic static CMOS can simultaneously reduce standby static power and increase performance by using low V_T devices in the evaluation networks and making all other devices high V_T. Experimental results show monotonic static CMOS to be 1.67 times faster than traditional static CMOS.

1. Introduction

Logic gates in a traditional static CMOS network are not usually skewed to favor a certain direction because their outputs can either switch high or low. However, logic gates in a monotonic static CMOS network can be skewed to favor a certain direction because each gate is guaranteed to exclusively make only a pull-up or a pull-down transition in a clock cycle. This is done by alternating low-skewed and high-skewed static gates such that every other gate output in a path is monotonically rising or monotonically falling [9]. In order for a gate's output to monotonically rise or fall, the concept of a precharge and evaluation time for the logic network is needed. During precharge, the output of each skewed static gate is initialized. In evaluation, each output will either stay at its precharged value or make a monotonic transition.

Since monotonic logic, such as monotonic static CMOS, is inherently non-inverting, it must be mapped to a network that does not contain intermediate inversions. This is in contrast to a static CMOS network which can allow intermediate inversions at the expense of only an inverter. Any random logic network can be transformed into a network of non-inverting functions (where inversions are only allowed at network boundaries) by finding a unate network representation. Generating a unate network from a binate random logic network may require logic duplication since separate logic cones for both positive and negative signal phases may be needed. Algorithms for transforming a binate random logic network into an inverter-free unate network have been developed [6] [7]. Binate to unate network conversion will at most double the amount of original logic and will not increase the number of logic levels.

With the growing use of portable electronics, which are typically used for event driven applications, static power dissipation has become an important design issue since it affects battery lifetime. Though dynamic power is important when circuits are active, static power dominates when circuits have significant standby time. As both the supply voltage (V_{DD}) and device threshold voltages (V_T) continue to be scaled down to reduce dynamic power dissipation and maintain performance, the static power dissipation has increased since reducing V_T leads to an exponential increase in subthreshold leakage current. Lowering V_{DD} also helps with device reliability and hot-electron effects in short channel devices. Recently, dual V_T processes have been developed to help circuit designers maintain performance and reduce static power dissipation.

There has been some previous work in the area of optimizing static CMOS networks for dual V_T technology. One method reduces leakage by placing a high V_T "gating" transistor in series with the normal circuitry [5]. However, this large "gating" transistor will increase area, delay and sensitivity to noise. More recent methods reduce leakage while minimizing performance loss by assigning a high V_T to transistors along non-critical paths and a low V_T to transistors along critical paths [10] [11]. Monotonic static CMOS can simultaneously reduce standby leakage power and increase performance by using low V_T devices in the evaluation networks and making all other devices high V_T.

The paper is organized as follows. Section 2 introduces monotonic static CMOS and provides definitions of terms frequently referred to in this paper. Design considerations for monotonic static CMOS are then discussed in Section 3. This is followed by Section 4 which provides a methodology for synthesizing monotonic static CMOS circuits. In Section 5, eight MCNC combinational logic benchmarks are used to compare monotonic static CMOS and traditional static CMOS. Results comparing the use of high V_T and low V_T devices are also discussed. Finally, the concluding remarks are given in Section 6.

2. Monotonic Static CMOS

An example of a low-skewed (LS) monotonic static CMOS gate is shown in Fig. 1 (a). The devices are sized so that the gate has a fast fall delay at the expense of a slow rise delay. Likewise, a high-skewed (HS) gate is designed to provide a fast pull-up, as shown in Fig. 1 (b).

A “precharge” device tied to *clk* or $\overline{clk}$ can be added to speed up the slower edge, if necessary, when *clk* is low.

When *clk* is low, the output of the LS (HS) gate is precharged to 1 (0). During the evaluation phase of *clk*, the output of the LS (HS) gate will either switch from 1 to 0 (0 to 1) or remain at the precharged value. A path of monotonic static gates must alternate with LS and HS gates, just like domino logic must alternate with inverting dynamic and static gates [9]. Thus, a LS gate can only drive a HS gate and vice-versa. Once this constraint is met, the skewed direction of a path can be very fast.

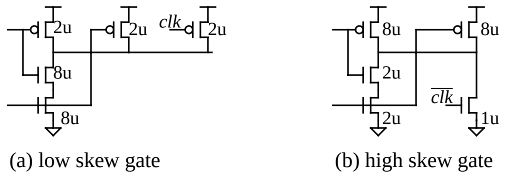

Fig. 1 Example of low and high skew NAND gates.

Monotonic static is very similar to dynamic logic families such as domino and Zipper CMOS [4] logic. In a LS gate, the nMOS pull-down network is the same as that in a domino network without the series evaluation transistor. The pull-up path of a LS gate is a complex keeper controlled by the inputs of the gate. The keeper can be sized to improve noise margins at the expense of speed. These similarities allow monotonic static CMOS to achieve higher performance compared to static CMOS. The performance gain is due to both the ability to skew gates to favor a certain direction and the ability to use fast wide fanin NOR gates (LS) and fast wide fanin NAND gates (HS). One key difference between monotonic static CMOS and domino logic is that non-monotonic switching can be tolerated in monotonic static CMOS, with only a delay penalty. Slowing down the frequency of a block that uses monotonic static CMOS prevents non-monotonic switching from causing setup time failures to memory elements. On the other hand, dynamic logic will fail if non-monotonic switching occurs during the evaluation phase and it cannot be recovered from at any reduction in clock frequency.

3. Design Considerations

Static CMOS gates are sized for roughly equal delays for either direction a path can take to minimize worst case delay. Monotonic static CMOS provides performance gains over static CMOS because only the skewed path is allowed to switch during evaluation. A gate can be skewed to favor one direction for speed at the expense of the other. The path delays for an inverter chain of alternating low-skewed and high-skewed inverters that are skewed by some amount to favor their evaluation direction are shown in Fig. 2. The results, using a 0.8μm MOSIS process, show that changing the skew factor for each inverter from 1 to 8 can increase performance by a factor of 2.4. Further skewing gives diminishing returns.

To prevent the slow path of a gate from affecting a path’s delay, clocked precharge devices can be added to LS and HS gates, as shown in Fig. 1. The precharge transistors can be sized to meet the desired precharge time requirement. During precharge, primary inputs to monotonic static gates must be initialized to prevent precharge race conditions from occurring. This can be done using a monotonically transitioning memory element such as a dynamic pulse latch or flop [3]. Like dynamic logic circuits, the precharge time can be hidden by using multi-phase clocks [2] [12].

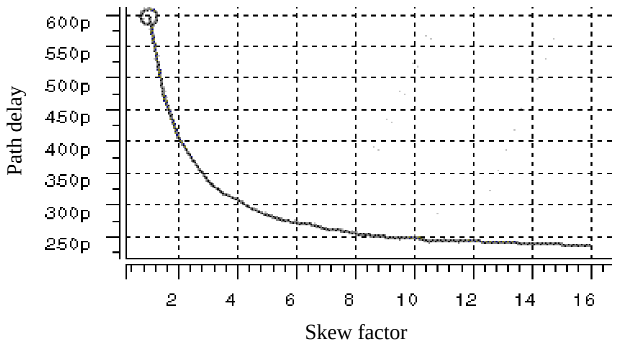

Fig. 2 Inverter chain delay vs. skew factor.

Like footless domino [1], if consecutive monotonic static CMOS gates use the same evaluate/precharge clock, then a DC path to ground through the precharge devices will exist whenever a gate whose inputs are still high is in precharge. To eliminate the DC path to ground from occuring, two constraints must be met. The first constraint is a gate must be in evaluation before a conducting path in its evaluation network is created. The second constraint is a conducting path in a gate’s evaluation network must not exist when going into precharge. To enforce these two constraints, a clocking scheme in which precharge clocks are delayed can be used.

The difficulty with assigning high V_T devices in a traditional static CMOS network is due to the inability of knowing whether a certain device will be on or off. Given stable inputs, a static gate will have a conducting network and a non-conducting network. A conducting network provides a path from either V_{DD} or ground to the output node. In order to reduce leakage, all paths through the non-conducting network must be through a high V_T device that is off. Otherwise, a path from V_{DD} to ground would exist through low V_T devices. Since a given device in the non-conducting network cannot be guaranteed to be off, making it a high V_T device may not reduce static power dissipation.

However, the monotonic nature of monotonic static allows all devices in either the precharge or evaluation networks to be turned off. Forcing all devices in precharge networks, which are high V_T, to be off will reduce standby static power. This can be accomplished by forcing all monotonic static CMOS gates to evaluate true. Leakage is further reduced since the devices in the precharge network are very small. Reducing standby static power with monotonic static CMOS does not affect performance because all devices in the evaluation network can be low V_T. An example of a LS gate using dual V_T technology is shown in Fig. 3. The clocked precharge device should be high V_T to reduce leakage power during standby operation.

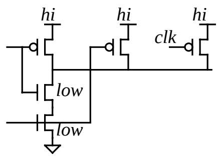

Fig. 3 Use of high and low V_T devices in a LS gate.

Because monotonic static CMOS allows for a very slow pull-up for LS gates, fast wide fanin NOR and AOI gates can be used. Likewise, fast wide fanin NAND and OAI gates can be used for HS gates. The use of these types of gates helps remove the need for extra precharge devices in the critical evaluation path, but it may still be necessary to insure that internal nodes in the precharge path are fully precharged to ground or V_{DD} in order to prevent charge sharing problems, as shown in Fig. 4.

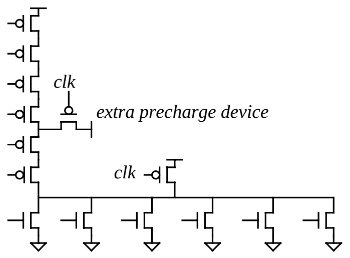

Fig. 4 Example of fast LS NOR6.

4. Synthesis of Monotonic Static CMOS

Monotonic logic is inherently non-inverting and must be mapped to a network that does not contain intermediate inversions. The removal of intermediate inverters within a logic network can be accomplished by finding a unate representation for the network. However, generating a unate network from a binate random logic network may require logic duplication since separate logic cones for both positive and negative signal phases may be needed.

After a unate network representation has been created, the network can be technology mapped to monotonic static CMOS gates. Our approach uses a concurrent two-coloring and technology mapping algorithm [9] to merge a unate network's non-inverting functions into an alternating pattern of LS and HS gates. There are two ways to implement a non-inverting function. One way is to pair a LS gate with an inverter (non-inverting function 1 in Fig. 5), another is to pair an inverter with a HS gate (non-inverting function 5 in Fig. 5). Fig. 5 shows a network of non-inverting functions 1-6 and the resulting colored network of inverting functions. Since the internal inverters cancel out, only the gates labeled with a L or H will remain. Notice that the original non-inverting functions 1-6 are preserved throughout the network (i.e. we maintain logical correctness, since inverter cancellation takes place between every L-H connection).

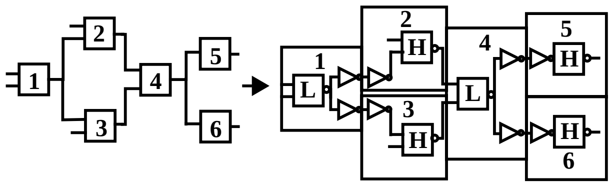

Fig. 5 Coloring a non-inverting network.

Unfortunately, not every network can be two-colored. A modification to the two-coloring algorithm is needed to resolve color conflicts caused by an uneven difference in the number of gates between two reconvergent fanout paths. The conflict resolution scheme for monotonic static CMOS is shown in Fig. 6. The non-inverting functions in Fig. 6 are represented by squares (the gates embedded within each square form a non-inverting function). Color conflicts are resolved by placing a dual output gate at the root of the reconvergent fanout paths. This allows a valid two-coloring of the network since both LS and HS gates may be driven. The logic network maintains its correctness since the original non-inverting functions are still preserved.

The two-coloring and mapping algorithm, shown in Fig. 7, ensures that each LS (HS) gate will have no pull-down (pull-up) path longer than a user specified limit. Limits can also be placed on the number of devices in parallel and the number of inputs to a gate.

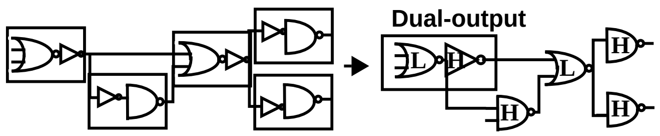

Fig. 6 Resolving color conflicts.

```
procedure Merge_Nodes_and_Color(Network N)
  while performing a postorder traversal of network N from its outputs
    for all predecessors of current_node that are not multi-output nodes
      predecessor_node = current_node's predecessor with
      a) the greatest distance from a primary input and, to break ties,
      b) the fewest number of transistors in series
      if merging predecessor_node and current_node satisfy node limits
        merge_predecessor_with_current_node();
      update_node_colors();
      resolve_color_conflicts();
end
```

Fig. 7 Algorithm for the coloring and mapping of a network.

5. Experimental Results

Using the monotonic static CMOS synthesis methodology described in the previous section, eight MCNC combinational logic benchmark circuits were implemented using monotonic static CMOS (MS) and static CMOS (STA). Logic minimization for all circuits was performed using SIS [8]. For the static CMOS circuits, a library similar to mcnc.genlib was used for technology mapping. For the monotonic static CMOS circuits, our library-free technology mapping algorithm was used to map LS monotonic static gates with limits up to three devices in series and eight devices in parallel. For the HS gates, we allowed two devices in series (for pull-up) and eight in parallel. Fanout optimization for all circuits was used to limit the number of fanouts to eight. Synopsys was used to verify that the logic networks before and after synthesis were functionally equivalent.

All logic devices in the nMOS (pMOS) networks for static CMOS were 12 (30) μm. For LS monotonic static CMOS gates, the pMOS pull-up network had devices of 2μm and the nMOS pull-down network had devices of 12μm. The HS gates had nMOS devices of 1μm and pMOS devices of 30μm. Worst-case paths were found using Pathmill and Spice simulations were performed using MOSIS's 0.8μm scalable CMOS process parameters.

The worst-case delays for each benchmark are summarized in Table 1. These results show monotonic static CMOS to have an average speed improvement of 1.67 times over static CMOS.

Simulations of monotonic static CMOS networks using high and low V_T devices were done using a state-of-the-art dual V_T very deep submicron process. Table 2 shows the performance and leakage power benefits. For the delay comparisons given in the second column, all devices in the precharge networks were high V_T. The evaluation networks used either low V_T devices (E_L) or high V_T devices (E_H). For the standby static power comparisons given in column three, all devices in the evaluation networks were low V_T. The precharge networks used either low V_T devices (P_L) or high V_T devices (P_H). From Table 2, the results show an average speed improvement of 33% when using low V_T devices compared to high V_T devices in the evaluation networks. Standby static power was shown to be reduced by a factor of 6.2 when using high V_T devices compared to low V_T devices in the precharge networks.

6. Conclusion

This paper introduces monotonic static CMOS and describes a methodology for synthesizing circuits with it. By consistently skewing static gates along a path and allowing more complex functionality in those gates, monotonic static CMOS can improve performance by an average of 1.67 times. With a dual V_T process, monotonic static CMOS can simultaneously reduce standby leakage power and increase performance. This is accomplished by using low V_T devices in the evaluation networks and making all other devices high V_T. These results show monotonic static to be an attractive option for high performance and low power circuits.

Table 1: Worst-case delay comparisons

Ckt.	STA [ns]	MS [ns]	STA/MS
k2	7.9	5.3	1.49
des	10.4	5.4	1.92
t481	7.5	4.9	1.53
dalu	8.8	5.6	1.57
term1	3.6	2.1	1.71
x3	4.7	3.3	1.42
rot	8.0	3.9	2.05
C1355	7.1	4.2	1.69
avg.			1.67

Table 2: Normalized Dual V_T Delay and Leakage Power

Ckt.	E_H/E_L delay	P_L/P_H power
k2	1.31	6.35
des	1.32	5.81
t481	1.35	6.43
dalu	1.35	5.62
term1	1.36	6.21
x3	1.31	6.33
rot	1.34	7.39
C1355	1.32	5.42
avg.	1.33	6.20

7. References

[1] A. Dharchoudhury, et. al., "Transistor-level sizing and timing verification of domino circuits in the Power PC microprocessor," *IEEE Int. Conf. on Computer Design*, pp. 143-148, 1997.

[2] D. Harris and M. Horowitz, "Skew-tolerant domino circuits," *IEEE J. Solid-State Circuits*, vol. 32, no. 1, pp. 1702-1711, 1997.

[3] F. Klass, et. al., "A new family of semi-dynamic and dynamic flip-flops with embedded logic for high-performance microprocessors," *IEEE J. Solid-State Circuits*, vol. 34, no. 5, pp. 712-717, 1999.

[4] C. Lee and E. Szeto, "Zipper CMOS," *IEEE Circuits and Devices Magazine*, pp. 10-16, May 1986.

[5] S. Mutoh, et. al., "1-V power supply high-speed digital circuit technology with multithreshold-voltage CMOS," *IEEE J Solid-State Circuits*, vol. 30, no. 8, pp. 847-854, 1995.

[6] M. Prasad, D. Kirkpatrick, R. Brayton, and A. Sangiovanni-Vincentelli, "Domino logic synthesis and technology mapping," *Proc. Int. Workshop on Logic Synthesis*, vol. 1, 1997.

[7] R. Puri, A. Bjorksten, and T. E. Rosser, "Logic optimization by output phase assignment in dynamic logic synthesis," *IEEE/ACM Int. Conf. on Computer-Aided Design*, pp. 2-8, 1996.

[8] E. Sentovich, et. al., "SIS: a system for sequential circuit synthesis," Technical Report UCB/ERL M92/41, University of California, Berke-

ley, CA, May 1992.

[9] T. Thorp, G. Yee, and C. Sechen, "Domino logic synthesis using complex static gates," *IEEE/ACM Int. Conf. on Computer-Aided Design*, pp. 242-247, 1998.

[10] Q. Wang and S. Vrudhula, "Static power optimization of deep micron CMOS circuits for dual Vt technology," *IEEE/ACM Int. Conf. on Computer-Aided Design*, pp. 490-496, 1998.

[11] L. Wei, Z. Chen, M. Johnson, and K. Roy, "Design and optimization of low voltage high performance dual threshold CMOS circuits," *Design Automation Conf.*, pp. 489-494, 1998.

[12] G. Yee and C. Sechen, "Clock-delayed domino for adder and combinational logic design," *Proc. IEEE Int. Conf. on Computer Design*, pp. 332-337, 1996