Output Prediction Logic: a High-Performance CMOS Design Technique

Larry McMurchie, Su Kio, Gin Yee, Tyler Thorp, and Carl Sechen
University of Washington, Seattle, Washington, USA
larry@ee.washington.edu / sechen@ee.wasington.edu

Abstract

We present Output Prediction Logic (OPL), a technique that can be applied to conventional CMOS logic families to obtain considerable speedups. When applied to static CMOS, OPL retains the restoring character of the logic family, including its high noise margins. Speedups of 2X to 3X over (optimized) conventional static CMOS are demonstrated for a variety of circuits, ranging from chains of gates, to datapath circuits, and to random logic benchmarks. Such speedups are obtained using identical netlists without remapping. When applied to pseudo-nMOS and dynamic families, in combination with remapping to wide-input NORs, OPL yields speedups of 4X to 5X over static CMOS. Since OPL applied to static CMOS is faster than conventional domino logic, and since it has higher noise margins than domino logic, we believe it will scale much better than domino with future processing technologies.

I. Introduction

Dynamic circuit families such as domino [1] are commonly used in today's high-performance microprocessors [2][3][4][5] for obtaining timing goals that are not possible using static CMOS circuits. Their increased performance is due to reduced input capacitance, lower switching thresholds, and circuit implementations that typically use fewer levels of logic due to the use of efficient and wide complex gates. It has been shown that dynamic logic can be used to realize average speed improvements of about 60% over static CMOS for random logic blocks when using synthesis tools tailored specifically for dynamic logic [6].

However, dynamic circuits have notable disadvantages. In the case of domino, logic must be mapped to a unate network, which usually requires duplication of logic. Perhaps the main disadvantage going forward is its increased noise sensitivity (compared to static CMOS). The only way to increase its noise margin is to sacrifice some of its performance gain. How to retain the good attributes of static CMOS, namely high noise immunity and easy technology mapping, while obtaining greater speed is an elusive goal.

Output prediction logic (OPL) is a new technique that can be applied to a variety of inverting logic families to increase speed while retaining the attributes of the underlying family. OPL relies on the alternating nature of logical output values for inverting gates on a critical path. That is, for any critical path, the logical output values of the gates along that path will be alternating ones and zeros. By correctly predicting exactly one half of the gate outputs, OPL obtains significant speedups (at least 2X) over the underlying logic families (e.g. static CMOS, pseudo-nMOS and dynamic logic).

When applied to static CMOS, OPL yields circuits that are typically 2 to 3 times faster than conventional static CMOS implementations. Although OPL employs clocks, OPL-static is inherently restoring logic and has the same noise margins as conventional static CMOS. OPL-static is also highly tolerant to clock skew, guaranteeing functionally correct results regardless of skew. Additionally, OPL-static uses the same synthesis tools as static CMOS (e.g. Synopsys). OPL can be applied to the same netlists as conventional static CMOS with a simple cell-for-cell substitution.

For the efficient implementation of wide NOR gates, designers often choose gates from pseudo-nMOS or dynamic logic families. OPL can be applied to these families as well. For example, a CLA adder implementation using OPL-pseudo-nMOS for wide-input NORs obtained a speedup of 5.4X over an optimized static CMOS implementation. These speedups were obtained while employing very conservative noise margins.

In Section II, we introduce OPL and show how it is applied to static CMOS circuits. In Section III we apply OPL to pseudo-nMOS and dynamic circuit styles. A means of generating the required clocks is outlined in Section IV. Section V quantifies the performance enhancement obtained with OPL when applied to chains of gates, a CLA adder and random logic benchmarks. We present conclusions in Section VI.

0-7695-0801-4/00 $10.00 © 2000 IEEE

II. Output Prediction Logic Applied to Static CMOS

In static CMOS logic, every gate is an inverting logic gate. Because of this inverting property, every output on a critical path must fully transition from 0 to 1, or 1 to 0 in the worst case. This is shown in Fig. 1, where we assume the primary input transitions high. This is why static CMOS is inherently slow. A circuit designer must take into account this worst-case delay scenario for a static CMOS critical path.

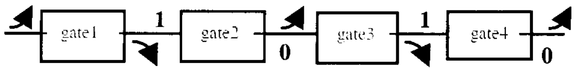


Figure 1. Static CMOS worst-case behavior.

Output Prediction Logic (OPL) greatly reduces the worst-case behavior of a critical path. OPL predicts that every inverting gate output on a critical path will be a logic one after the transitions are completed. Since all gates are inverting, as in static CMOS, the OPL predictions will be correct exactly one-half the time. Every other gate will not have to make any transition (see Figure 2).

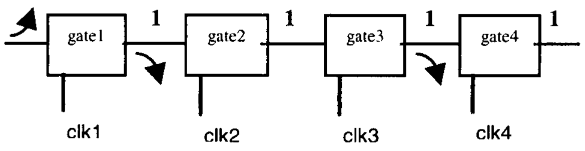


Figure 2. OPL predicting ones

There is, however, a key problem with this idea. A one at every output (and therefore input) is not a stable state for an inverting gate. The one will erode (possibly going to zero) in the latter gates of a critical path. The solution to this problem is to tri-state each gate with a clock, in which case ones at inputs and a one output is no longer a contradiction for an inverting gate. The gates remain tri-stated until their inputs are ready for evaluation. In this manner, predicted output values are maintained until new input values dictate otherwise. Successive clocks are delayed by a clock separation as shown in Fig. 3.

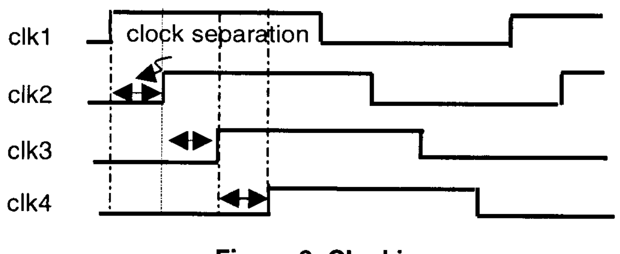
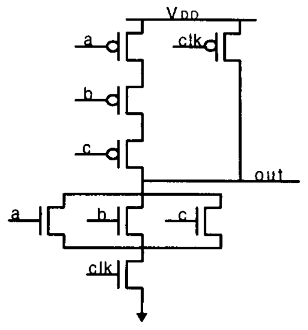


Figure 3. Clocking.

A tri-state, pre-charge-high static CMOS inverting gate implementing the above idea is shown in Fig. 4.



Fiaure 4. OPL-static CMOS NOR3.

When the clock (*clk*) is low, the gate is tri-stated, with the output being charged to a logic one. When the clock goes high, the gate becomes a conventional static CMOS gate.

While an actual circuit essentially follows this desired behavior, there are important nonidealities. Fig. 5 shows a chain of 3 OPL-static inverters and Fig. 6 shows the behavior of gate 2 as the clock arrival is varied. First, consider the case where the input to gate 2 in Fig. 5 is low, and therefore gate 2's output should remain high. If the clock arrives (goes fully high) at gate 2 after its input becomes stable at its low value, and if the clock to gate 3 is still low, gate 2's output will stay high at the pre-charged (predicted) value. If the clock arrives at gate 2 while its input is settling, a small glitch occurs, as shown in Fig. 6c. If the clock is earlier yet – when its 50% point occurs at the same time as the 50% falling transition point of the input to gate 2 -- the still falling (but not yet fully zero) inputs will cause a bigger glitch at the output of gate 2 as shown in Fig. 6b. If the clock is even earlier yet, the

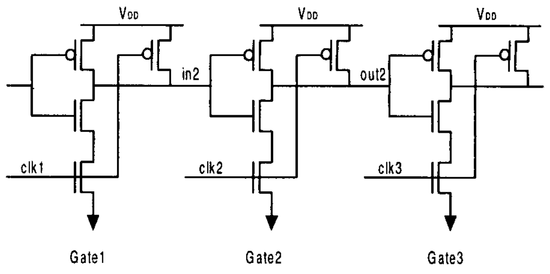
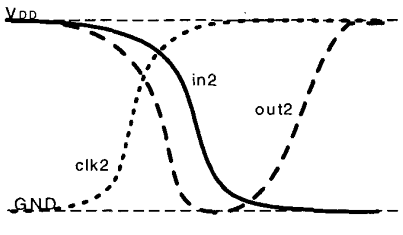
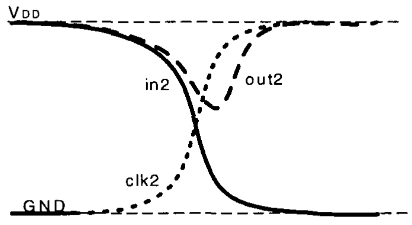


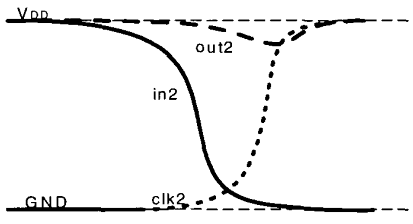
Figure 5. Chain of 3 OPL-static inverters.



a. Early Clock



b. Optimal Clock



c. Late Clock

Figure 6. Dependency of gate output upon clock arrival for gate 2 of Fig. 5.

precharged (predicted) value is completely lost, as shown in Fig. 6a.

The magnitude of the glitch is also enhanced by Miller kickback capacitance from the load gate 3. The kickback occurs when gate 3 also glitches to some extent (for exactly the same reasons as for gate 2) or falls all the way to 0. When this happens, gate 2's output load capacitance will be at least somewhat larger than what was seen by gate 2's precharge device when gate 3 was fully precharged. Should the clock to gate 3 arrive at almost the same time as the clock to gate 2, the kickback effect will be large since gate 3 fully transitions to zero, causing a significantly greater glitch at the output of gate 2.

If we view the glitch as highly undesirable, then we need to have the rising clock (or evaluation edge) arrive after the inputs to a gate have fully settled. If this is done for all gates, then the glitch will be very small. However, in so doing, we have created what is commonly called a clock-blocked circuit, in that the throughput of the circuit is limited by the clocks and not the data. Hence, the speedup achieved may not be impressive.

On the other hand, if we allow some glitching, then we can have the evaluation clock edges arrive somewhat earlier than the corresponding gate inputs. This will make the circuit throughput data-limited. However, if the glitches are all the way to zero as in Fig. 6a, the precharge (predicted) values are completely lost and there is no reason to believe that any speedup over a conventional static CMOS inverter chain would be achieved.

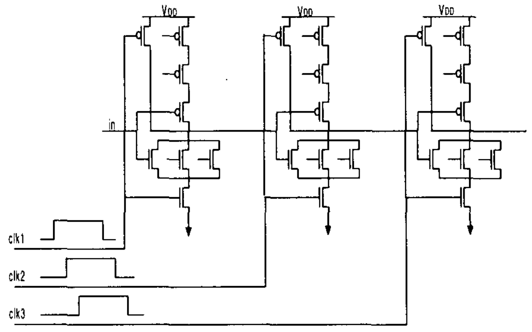


Figure 7. Chain of 3 OPL-static NOR3 gates.

Not surprisingly, we have found that there is an optimal point between the two extremes (fully clock-blocked and fully lost precharge values). As we shall see below, the minimum delay occurs when a modest amount of glitch occurs, as shown in Fig. 6b.

Fig. 7 shows a chain of OPL-static NOR3 gates. A chain of length 10, each gate having a fanout of four identical gates was simulated using parameters from the 0.25 micron 2.5V TSMC process [8]. An optimal clock separation was determined by sweeping over the clock separation in Hspice to find the separation (0.13ns) that gave the minimum delay. Waveforms for gate outputs at this separation are shown in Fig. 8. We see half the out-

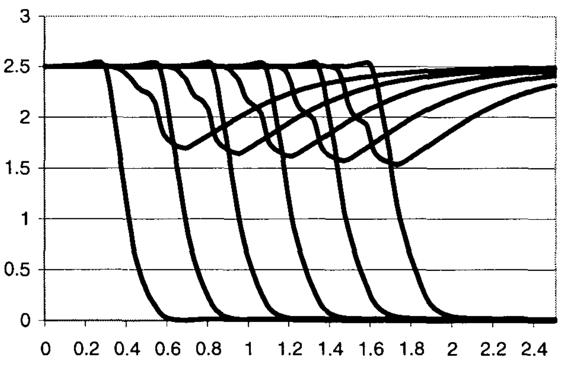


Figure 8. Waveforms (volts vs. ns) for OPL-static NOR3 chain at a clock separation of 0.13ns.

puts falling to zero and the other half dipping (or glitching) and then rising back up to V_{DD}, as seen in Fig. 6b. Note that the evaluation of successive gates overlaps considerably. The objective of OPL is to control this overlap so that low-going gates get a headstart in evaluating, while high-going gates do not glitch excessively. If the pull-up and pull-down strengths are the same, one would expect the minimum overall delay should occur when a high-going output dips to a voltage intermediate between zero and V_{DD}. This voltage should be near the maximum gain point, where a small change in the input will cause a large change in the output. By controlling the clock separation we effectively position the output near this critical voltage. This contrasts to the normal operation in static CMOS, where gates begin evaluation at either zero or V_{DD} where the gain is the smallest.

Positioning gate outputs at their maximum gain point in order to increase speed has been used previously in a limited context. Zhu and Carlson describe such a method called Critical Voltage Transition Logic (CVTL) [9]. In CVTL a chain of pseudo-nMOS inverters is precharged low, then allowed to float simultaneously to a critical voltage (the point of maximum gain). Propagation delay is greatly reduced by this "preconditioning" of gate outputs. Unfortunately, this scheme depends on a very delicate balancing of loading and drive strengths between stages in order for the preconditioning state to hold. In a chain of arbitrary gates, outputs will typically decay from a precharged value unless explicitly prevented by a method such as the OPL delayed clocks.

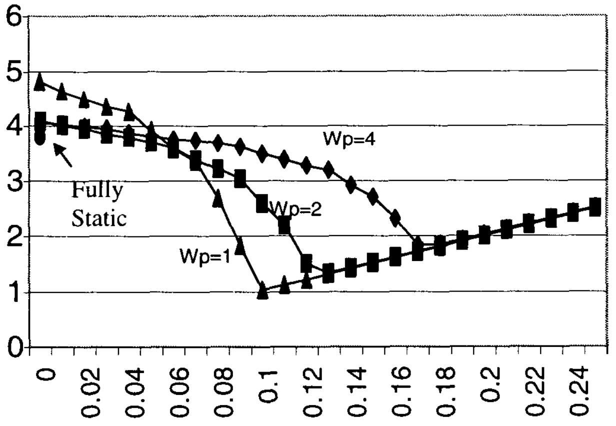


Figure 9. Total delay (ns) vs. clock separation (ns) for OPL-static NOR3 chains.

The dependency of total delay upon the clock separation can be seen clearly from Fig. 9. We show three curves, corresponding to OPL-static chains with different pMOS device sizes (W_p) in the pull-up network (pull-down devices were all sized with W_n=2μm). At zero separation, we have the case where every gate is precharged high and allowed to float at the same time. Nearly all the gates (except those near the beginning of the chain) will decay to alternating 1's and 0's before having to make a full swing to the opposite rail. Note that as the clock separations are increased from zero, more of the gate outputs approach an intermediate voltage before correcting. At the minimum in each curve, the clock separation is the effective gate delay. Eventually, as the clock separation continues to increase, the circuit (in effect) becomes clock-blocking and the delay increases linearly with clock separation.

The W_p=4μm curve corresponds to the same W_p and W_n as in the fully static gate. Note that the noise margin for this gate is exactly the same as for the fully static gate. The delay at zero separation is very close to that of the fully static gate (4.0ns vs. 3.8ns). As one would expect, the total delay at the minimum of this curve is about half the delay at zero separation (1.9ns vs 4.0ns). We can decrease the width of the pMOS devices in the pull-up network and thereby decrease the internal loading, speeding the gates up. As we decrease W_p from 4μm to 2μm, then to 1μm, we can see the minimums in the curves decrease. However, as we decrease this size we also reduce the ability of the gates to pull up and recover from a glitch. This results in a very steep rise in the delay-separation curve before the minimum point; the circuit will be sensitive to clock skew in this region. A W_p of 2μm was chosen as a reasonable compromise between increased speed, an ability to recover from glitches, and good noise margins. We note that the noise margin for W_p=2μm is only slightly less than that for W_p=4μm, and is much higher than nominal domino noise margins. The role of the pMOS pull-up network is analogous to the complex keeper in monotonic static CMOS circuits [6].

Note that highly accurate clocking is not required to achieve high speedups over fully static gates. In the W_p=2μm case, a 10% error in the overall average clock results in a speedup (over fully static) of 2.5X vs 2.8X if the clock were positioned at the exact minimum. Such a level of control (10%) in overall average clock skew is readily attainable today.

III. Output Prediction Logic Applied to Pseudo-nMOS and Dynamic Design

The OPL technique can be applied to pseudo-nMOS as well as dynamic circuits. A tri-state, pre-charge-high pseudo-nMOS gate is shown in Fig.10. When the clock (*clk*) is low, the gate is tri-stated, with the output being charged to a logic 1. When the clock goes high, it becomes a pseudo-nMOS gate. The pull-up serves both to precharge the gate and correct a high output when it glitches. This pMOS device is sized in accordance with the pull-down stack to yield an appropriate output-low voltage. Note that the output-low voltage can be set closer to zero than for conventional pseudo-nMOS since pull-up delay is less of a concern, thus lowering static power dissipation (as will be shown later). The behavior of the gate is similar to that of pseudo-nMOS. Once *clk* goes high, we would expect this gate to outperform OPL-static for wide input NORs, where the pull-up chains are not as effective as a single pull-up device in correcting a high output that has glitched.

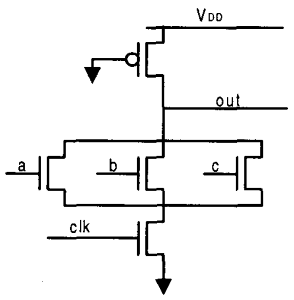


Figure 10. OPL-pseudo-nMOS NOR3.

We also tested the Output Prediction Logic concept with dynamic logic gates. As shown in Fig. 11, an OPL-dynamic gate looks exactly like a domino gate, but with the output inverter missing. Note that the gate precharges high, and that the keeper, if sized sufficiently large, will enable the output node to recover from glitches. If the clock arrives too early (keep in mind that the inputs precharge high), a gate may glitch so much that the keeper shuts off, causing the output voltage to remain at a value possibly well below V_{DD} (or even zero). Thus, in contrast to OPLstatic and OPL-pseudo, OPL-dynamic gates can fail functionally. One must ensure that the keeper is sized sufficiently large to correct for glitches arising from Miller coupling (kickback) of the output to fanout gates.

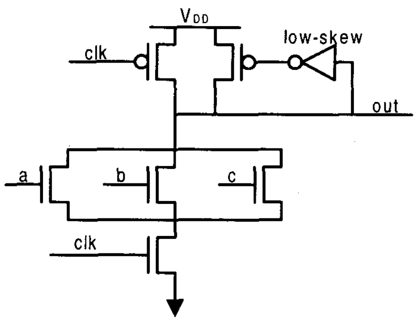


Figure 11. OPL-dynamic NOR3.

Note that the OPL-dynamic gate is very different from a conventional domino gate, as it does not have a following inverter. Domino circuits are positive unate and may have critical paths that require every gate to discharge. Such circuits will therefore be slower than OPL-dynamic where one can take advantage of the alternating nature of the logical output values of the gates on critical paths to speed up the circuit. Domino circuits also generally require logic duplication to map to positive unate functions, in contrast to OPL circuits.

IV. Generation of Clocks

The fast speed of OPL logic requires the clocks to be separated by a small amount, typically less than a buffer

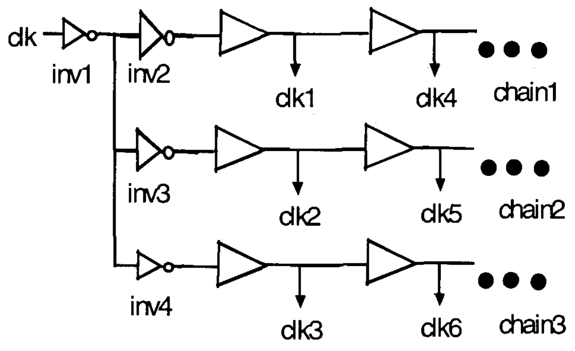


Figure 12. Generation of the clocks.

delay. Thus, a normal chain of delay buffers is not sufficient to generate these clocks since each clock will be separated by a buffer delay that is more than a gate delay. The required clock separations can be generated by using several buffer delay chains as shown in Fig. 12. For example, if we want a clock separation equal to 1/3 of a buffer delay, then inverters *inv2* and *inv3* are sized such that *chain2* lags *chain1* by 1/3 of a buffer delay. Therefore, *clk2* is 1/3 of a buffer delay behind *clk1* and *clk5* is 1/3 of a buffer delay behind *clk4*. Similarly, *inv3* and *inv4* are sized such that *chain3* lags *chain2* by 1/3 of a buffer delay. Thus, *clk3* is 1/3 of a buffer delay behind *clk2* and *clk6* is 1/3 of a buffer delay behind *clk5*. Since *clk4* is one buffer delay behind *clk1* and *clk3* is 2/3 of a buffer delay behind *clk1*, *clk4* is 1/3 of a buffer delay behind *clk3*. As a result, all clocks are separated by 1/3 of a buffer delay. To achieve arbitrary clock separations, we modify the buffer delay and size *inv2*, *inv3* and *inv4* accordingly, and/or increase or decrease the number of chains.

V. OPL Performance

To determine the performance possible with OPL, we simulated critical paths consisting of 10 identical gates, each gate in the path driving a load of four identical gates. We used nominal simulation parameters for the TSMC 2.5 volt 0.25 micron process [8]. The delay results in Table 1 compare static CMOS, OPL-static, OPL-pseudo-nMOS (OPL-pseudo) and OPL-dynamic. The numbers in parentheses are the speeds relative to the static CMOS chain. From the INV data, it is apparent that the fanout-of-four static CMOS inverter delay for this process is 160ps.

Table 1. Delays(ns) for chains of 10 gates (FO of 4).

Chain Type	Static CMOS	OPL Static	OPL Pseudo	OPL Dynamic
INV	1.62 (1.0)	0.43 (3.77)	0.42 (3.86)	0.43 (3.77)
NOR3	3.83 (1.0)	1.34 (2.86)	0.71 (5.39)	0.76 (5.04)
NAND2	2.45 (1.0)	0.94 (2.61)	0.93 (2.63)	1.02 (2.40)
NAND3	3.32 (1.0)	1.44 (2.31)	1.54 (2.16)	1.54 (2.16)
NAND4	4.24 (1.0)	1.97 (2.15)	2.16 (1.96)	2.15 (1.97)
AOI22	4.75 (1.0)	2.13 (2.23)	1.81 (2.62)	1.80 (2.64)
AOI222	6.75 (1.0)	3.04 (2.22)	2.63 (2.57)	2.49 (2.71)
Average	(1.0)	(2.59)	(3.03)	(2.96)

The nMOS devices for all versions of the above gates were sized to have an effective pull-down width of 2 μm. Thus, if the pull-down portion of a gate had stack of *k* transistors in series, the size of the transistors was 2*k* μm. The pMOS transistors for the static CMOS gates were uniformly sized by sweeping their size versus overall delay for the chain of 10 gates, and then selecting the size that minimized the worst case delay for the chain. As explained in Section II, the sizes of the pMOS devices in the OPL-static gates were selected to obtain high speed as well as suitably fast recovery from glitches at the output nodes. We used 2 μm for all pMOS devices except for NAND3 and NAND4 gates which used 3 μm. For OPL-pseudo, 1.4μm was used for the pMOS pull-up device. The keeper pull-up pMOS device was 2 μm for OPL-dynamic.

On average, OPL-static chains are 2.6 times faster than static CMOS and OPL-pseudo chains are 3 times faster than static CMOS. In fact, OPL-pseudo NOR3 chains are 5.4 times faster than static CMOS, and INV chains are 3.86 times faster. NOR gates with an arbitrary number of inputs are extremely fast with OPL-pseudo. On the other hand, OPL-static is faster for NAND gates than OPL-pseudo. OPL-dynamic has about the same performance as OPL-pseudo. The results so far have been for homogeneous chains of gates.

Table 2. Delays for heterogeneous chains of 8 gates.

Logic Family	Delay	Speedup
Static CMOS	2.13ns	1.0
OPL Static	910ps	2.34
OPL Pseudo	650ps	3.28
OPL Dynamic	688ps	3.10

We also constructed a chain of eight heterogeneous gates, which were (in order): NOR3, NAND3, AOI22, INV, INV, NOR3, NAND3, and AOI22. Each gate drives a load of four identical gates. The device sizes used were exactly those selected for the uniform chains. Having the gates so ordered means that each gate type will have to pull down once and stay high once. Table 2 shows the results. While it might be expected that tailoring the clock separations to the specific gate types would yield the best results, we have found that not to be true. A uniform clock separation, which is inherently easier to generate, yields results equivalent to specially tailored clock separations. The speedup obtained for the heterogeneous chain is quite similar to that obtained for the average homogeneous chain.

We also compared the total energy consumption for static CMOS, OPL-static and OPL-pseudo (including precharging energy), as well as OPL-dynamic. Table 3 shows that the average energy consumption of OPL-static gates is around 1.3 times of that for static CMOS. This energy includes the energy needed to generate the clocks, precharge outputs, and any energy consumed during evaluation. OPL-pseudo gates consume about 2.2 times as much energy as static CMOS gates, on average. However, the energy consumption for OPL-pseudo NOR gates is reasonably modest. This is because the output-low voltages of these gates are rather low (pull-up delay is less of a concern when predicting ones). With V_{OL} near zero, the static current drawn when the pull-down network is conducting is fairly small. Instead of the 50% duty cycle clocks that we used, we could have used clock pulses of smaller duration to obtain even smaller energy consumption with the OPL-pseudo gates. OPL-dynamic consumes less energy than OPL-pseudo but offers similar speed. Therefore, it could be a better candidate for NOR gates than OPL-pseudo. However, as pointed out earlier, functional failures are possible if output glitches are too large.

Table 3. Energy consumption(in pJ) for chains of 10 identical gates. Numbers in parentheses are relative to static CMOS.

Chain Type	Static CMOS	OPL Static	OPL Pseudo	OPL Dynamic
INV	2.00 (1.0)	3.80 (1.90)	4.97 (2.49)	4.41 (2.21)
NOR3	3.19 (1.0)	4.45 (1.39)	6.07 (1.90)	4.47 (1.40)
NAND2	3.83 (1.0)	5.00 (1.31)	8.39 (2.19)	5.60 (1.46)
NAND3	6.23 (1.0)	6.66 (1.07)	12.7 (2.04)	7.51 (1.21)
NAND4	8.65 (1.0)	12.7 (1.47)	19.3 (2.23)	10.0 (1.16)
AOI22	6.13 (1.0)	6.31 (1.03)	12.8 (2.09)	7.01 (1.14)
AOI222	7.08 (1.0)	7.70 (1.09)	16.7 (2.36)	8.09 (1.14)
Average	(1.0)	(1.32)	(2.19)	(1.39)

We now show results for complete logic networks. In Table 4, we first consider a 32-bit carry look-ahead adder (CLA32) network described in [5]. Two schemes were implemented. The first scheme uses 8 4-bit Full Adders (FA). Static CMOS can only realize this scheme since stack heights are limited to four, and therefore three levels of 4-bit CLA units are needed. We implemented the exact same network using static CMOS, OPL-static and OPL-pseudo. The speedup is close to the speedup of the NAND4 chain in Table 1, as expected, since NAND4 gates dominate the critical path of this scheme. The second scheme uses only two levels of CLA units by employing 8 4-bit full adders with the second level CLA being an 8-bit unit. Thus, this scheme saves one level of CLA units and cannot be implemented using static CMOS since 8 series transistors would be required.

Table 4 Delays for implementations of 32-bit CLA.

Logic Family	Delay	Speedup	CLA type
Static CMOS	3.0ns	1.0	Three levels
OPL Static	1.5ns	2.0	Three levels
OPL Pseudo	1.82ns	1.65	Three levels
OPL Pseudo	552ps	5.43	Two levels
OPL Mixed	781ps	4.38	Two levels

Since OPL-pseudo and OPL-dynamic also support wide fan-in NOR gates, they are best suited for the second scheme, which uses NOR gates up to NOR8. We can use NOR gates here because we can apply DeMorgan's law to all the equations in the CLA unit [7]. The advantage of the second scheme is that it only requires two levels of CLA units in contrast to three CLA levels in the first scheme. We implemented the second scheme using purely OPL-pseudo and using a mix of OPL-static and OPL-dynamic gates (OPL-dynamic was used only for the NOR gates in the network). The speedup using purely OPL-pseudo is about 5.4X. For the mixed OPL-static/OPL-dynamic case, the speedup is about 4.4X but the energy consumption is less than using purely OPL-pseudo.

We also implemented five ISCAS benchmark circuits in both static CMOS and OPL-static. The networks were obtained with Synopsys, using a library with functionality similar to lib2.genlib (26 logically different cells) with the addition of inverters and buffers of various drive strengths. All possible Synopsys scripts were used and the fastest static CMOS implementation was selected. The resulting network was then used for both static CMOS and OPL-static experiments. Note that the networks used in both experiments were identical; only the cell implementations were different. Both implementations were analyzed using Pathmill.

For OPL-static, all paths with delays within 10% of the critical path were selected in Pathmill. The pull-up delays of these gates were artificially set to be small in Pathmill to properly mimic the behavior of OPL-static paths in which the primary delay contributor is the pull-down delay of every other gate in a path.

We computed the logic level of each gate and assigned clock i to all gates at level i. The resulting set of OPL-static paths were then simulated using Hspice. The clock separation was varied and the separation selected that gave the minimum worst-case delay over all paths.

We noted from our experience with homogeneous chains of OPL gates that optimal delays are obtained when high gate outputs glitch down to approximately 60% of V_{DD}. We also noted from the initial implementations of the ISCAS benchmarks that some of the critical paths had gates glitching considerably more than this. We therefore surmised that better worst-case delays could be obtained if we prohibit glitches of more than 50% of V_{DD}. There are two methods that we identified to remove excessive glitches. One is to increase the pMOS transistor widths in the pull-up network for the relevant gate. Another is to increase the clock separation for the slow falling gates providing inputs to the glitching gate. We implemented the latter by assigning to the glitching gate the next higher clock phase and propagating the reassignment to their fanout cones.

```
Algorithm(Optimized_OPL)
For level = 1 to NumLevels
   Run Hspice on critical paths
   For each gate in level
      If a high gate output glitches below 1.25V
         Double pMOS widths in pull-up network
   End For
   Run Hspice on critical paths
   For each gate in level
      If a high gate output glitches below 1.25V
         Move gate clock ahead one phase
   End For
End For
```

Figure 13. Optimized OPL algorithm

The Optimized_OPL algorithm, which was employed to choose between these methods is shown in Fig. 13. The algorithm eliminates excessive glitches level by level. A glitch is corrected if it causes a voltage drop more than 50% of VDD (1.25V). First an attempt is made to correct the glitch by increasing the width of the pMOS devices. If this is not successful, the glitching gate is moved ahead a clock phase (separation) to allow more time for late falling inputs to arrive.

Table 5. Results for ISCAS benchmarks

Circuit	Static	OPL-static		Optimized_OPL-static	
	Delay (ns)	Delay (ns)	Clocks	Delay(ns)	Clocks
t481	0.91(1.0)	0.48(1.90)	7	0.46(1.98)	7
term1	1.38(1.0)	0.91(1.52)	10	0.70(1.97)	11
x3	2.20(1.0)	0.85(2.59)	10	0.67(3.28)	10
rot	2.19(1.0)	1.38(1.59)	16	1.05(2.09)	17
dalu	2.35(1.0)	0.96(2.45)	14	0.96(2.45)	15
Avg	(1.0)	(2.01)		(2.35)	

Results are shown in Table 5 for static, OPL-static, and Optimized_OPL-static. On average, OPL-static is 2.01 times faster than static CMOS. At 2.35 times static CMOS, Optimized_OPL-static shows a significant improvement over OPL-static. Circuit x3 exhibits a particularly large speedup over static because of the preponderance of NOR3, NOR4 and inverters on critical paths. Although Optimized_OPL-static could conceivably double the total number of clocks, in practice at most one additional clock gave the reported speed improvement. Even greater speedups over static CMOS should be possible for these circuits if they were mapped to wider gates (e.g. NORs) as was done with the CLA. Technology mapping tools targeting such wide gates have been demonstrated for CD domino logic [7].

VI. Conclusions

We presented Output Prediction Logic, a technique that can be applied to conventional CMOS logic families tobtain considerable speedups. Speedups of 2X to 3X over conventional static CMOS were demonstrated for a variety of circuits, ranging from chains of gates, to datapath circuits, and to random logic benchmarks. These speedups are obtained while retaining the noise margins and level restoring nature of static CMOS as well as the same netlists..

Acknowledgements

We would like to acknowledge the financial support provided by the Semiconductor Research Corporation, the National Science Foundation, and the NSF Center for the Design of Analog and Digital ICs (CDADIC). We would also like to acknowledge the contributions of Jovanka Ciric.

References

1. D. Harris, "Skew-tolerant circuit design," Ph. D. dissertation, Stanford University, Stanford, CA, 1999.
2. F. Klass, et. al., "A new family of semi-dynamic and dynamic flip-flops with embedded logic for UltraSPARC-III," *IEEE J. Solid-State Circuits*, vol. 34, no. 5, pp. 712-717, May 1999.
3. B. Benschneider, et. al., "A 300-MHz 64-b quad-issue CMOS RISC microprocessor," *IEEE J. Solid-State Circuits*, vol. 30, no. 11, pp. 1203-1214, Nov. 1995.
4. A. Charnas, et. al., "A 64b microprocessor with multimedia support," *ISSCC Dig. of Tech. Papers*, pp. 177-179, Feb. 1995.
5. R. Colwell and R. Steck, "A 0.6um BiCMOS processor with dynamic execution," *ISSCC Dig. of Tech. Papers*, pp. 176-177, Feb. 1995.
6. T. Thorp, G. Yee and C. Sechen, "Monotonic Logic and Dual Vt Technology", *Proc. of Int. Conf. on Computer Design (ICCD)*, Austin, TX, October 1999.
7. G. Yee and C. Sechen, "Clock-delayed Domino for Adder and Random Logic Design," Proc. IEEE Int. Conf. on Computer Design (ICCD), October 7-9, 1996, Austin, TX.
8. MOSIS, "MOSIS Parametric Test Results," http://www.mosis.org.
9. Z. Zhu and B. Carlson, "Critical Voltage Transition Logic: An Ultrafast CMOS Logic Family", Proc. IEEE Int. Conf. On Computer Design (ICCD), Austin, TX, October 1997.