Low-Power Data-Driven Dynamic Logic (D^3L)

R. Rafati[1], *S. M. Fakhraie*[1], *K. C. Smith*[2]

[1]ECE Department, University of Tehran, Tehran, IRAN
[2] ECE Department, University of Toronto, Toronto, Ontario, CANADA
(rafati | mehdif)@khorshid.ece.ut.ac.ir, lfujino@cs.utoronto.ca

ABSTRACT

In this paper a new family of low-power dynamic logic called Data-Driven Dynamic Logic (D^3L) is introduced. In this logic family, the synchronization clock has been eliminated, and correct sequencing is maintained by appropriate use of data instances. Then, it is shown that replacement of the clock with input data implies less power dissipation without speed degradation compared to conventional dynamic logic.

1. INTRODUCTION

In conventional CMOS circuits, for each n-type device there is a corresponding p-type device. In fact, the required logic function is implemented twice, both in a PDN (Pull-Down Network) and a PUN (Pull-Up Network). For increasing speed, in dynamic logic, the PUN is often replaced with a single transistor that is controlled by a global clock signal [1]. Correspondingly, circuit operation is divided into distinctive precharge and evaluate phases. In the precharge phase, the output node is precharged to a particular level (usually high for a PUN) while the current path to the other level (GND for a PDN) is turned off. Upon completion of the precharge phase (CLK='1'), the path to the high level is turned off by the clock and the path to ground is turned on. Therefore, depending on the state of the inputs, the output node will either float at the high level or will be pulled down.

A clear advantage of a dynamic CMOS gate is its reduced silicon area. Typically, there are 2n transistors in a conventional n-input CMOS gate, while the dynamic configuration needs only n+2 transistors. Also, due to the smaller area and consequently smaller parasitic capacitance, power dissipation and speed are, in principle, improved by the dynamic approach.

Unfortunately, however, such dynamic logic blocks cannot be cascaded straightforwardly [1]. A number of design styles for avoiding the associated problems have been developed. For example, in the Domino logic style [2], a static inverter follows every *CLK-N* block (Fig. 1). This ensures that all inputs to the next logic block are set low after the precharge period. Thus, a chain of Domino gates can be cascaded with no problem. Note that though the presence of a static inverter improves the output drive, it has the disadvantage that only noninverting logic can be implemented.

For resolving this noninverting problem of Domino logic, NP-CMOS logic has been developed [3]. In this style, the high-precharged output node is connected directly to a PUN (Fig. 2) which consists of PMOS switches driven by a $\overline{CLK}$ signal, and called a $\overline{CLK}$ -*P* block. In a similar way, a *CLK-N* block can follow a $\overline{CLK}$ -*P* block without introducing any false evaluation problems. Compared to Domino, NP-CMOS is more than 20% faster due to the elimination of the static inverters, and correspondingly the reduction of load capacitance [1]. However, the presence of PMOS blocks in NP-CMOS increases the overall area. In addition, we must take into account the problem of producing $\overline{CLK}$ and routing it throughout the circuit.

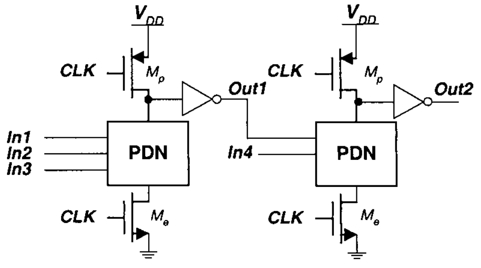


Fig. 1. Cascading dynamic gates in the Domino logic style.

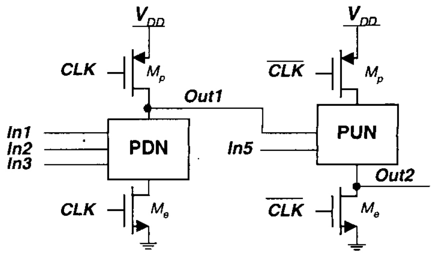


Fig. 2. Cascading dynamic gates in the NP-CMOS logic style.

Compared to static CMOS logic, the input capacitance of every dynamic gate can be reduced by 50% or more, but due to the presence of an additional transistor (the clocking transistor) that must be cascaded with the main block, the speed generally does not double. Moreover, the excessive loading of the clock signal that must be connected to every dynamic gate and the corresponding routing problems reduce the benefits of dynamic logic. Correspondingly, as well, the increasing frequency of today's circuits results in greater power consumption when implemented in dynamic fashion. For example in the Alpha 22164 microprocessor, the clock-distribution system consumes 20W, which is 40% of the total power dissipation of the processor [4].

In searching for an answer to the clock-power problem, we have developed the D^3L logic style to be introduced next. In this style, both the clocking signal and the associated clocking transistor are removed from the dynamic gate. In addition, inverting logic gates can be easily implemented.

0-7803-5482-6/99/$10.00 '2000 IEEE

2. DATA-DRIVEN DYNAMIC LOGIC (D^3L)

In creating conventional dynamic logic gates, in which one of the PDN or PUN of static logic is removed, a set of conditions must be imposed on the inputs. For example in a Domino logic block, all of the inputs must be low in the precharge phase, while in NP-CMOS all inputs to a P-block must be high in the precharge phase. This suggests that if we can precharge the corresponding gate with a combination of input data, then the need for a clock signal could be eliminated. We call such a use of data precharging instead of clock precharging Data-Driven Dynamic Logic or D^3L. While maintaining the usual conditions enforced on Domino and NP-CMOS, in D^3L we replace the clock signal by one or more inputs. In the following example, we demonstrate such a replacement process.

Consider a 2-input AND gate in Domino logic, implemented as shown in Fig. 3. As noted in this figure, both inputs *A* and *B* are held at the low level in the precharge phase. Awareness of this usual restriction enables us to eliminate the clock signal as shown in Fig. 4.

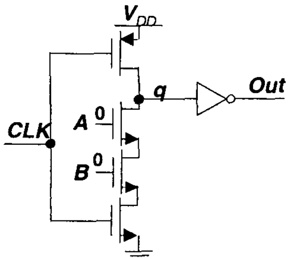


Fig. 3. A Domino AND gate.

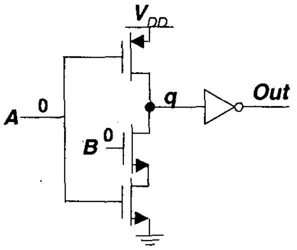


Fig. 4. A D^3L AND gate.

During the precharge phase when the input *A* is low, node *q* is precharged high. When signal *A* makes a possible transition from low to high, the evaluation phase starts. At this phase, depending on the value of *B*, node *q* conditionally discharges. In a similar manner using a p-logic block, an OR gate can be converted to D^3L (Fig. 5). The resulting operation is the same as for the implementation of a p-logic block in NP-CMOS logic, but without any need for the $\overline{CLK}$ signal.

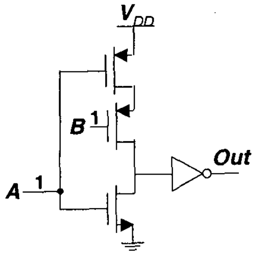


Fig. 5. A D^3L OR gate.

3. IMPLEMENTATION OF VARIOUS FUNCTIONS IN D^3L

In general, whenever we have a function F in the product-of-sums form $F=\prod_{i=1}^{n} S_i$, then the minimum S_i (the S_i with a minimum number of literals) in which all inputs have a low value (the Domino condition) is used to replace the clock. This replacement procedure results in a minimum number of series transistors that must be placed in the PUN. Examples of this operation are shown in Fig. 6 and Fig. 7.

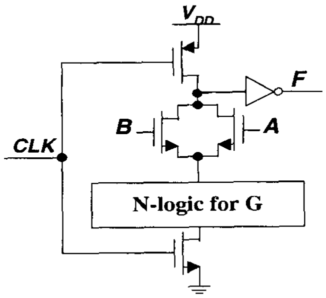


Fig. 6. Dynamic implementation of F=(A+B).G.

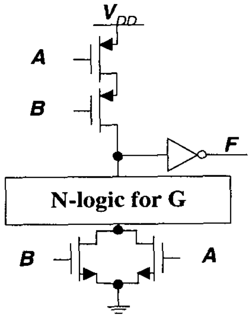


Fig. 7. D^3L implementation of F=(A+B).G.

Whenever all of the inputs have high values in the precharge phase, the replacement process is done in the PDN (corresponding to the $\overline{CLK}$ -section of NP-CMOS logic). As usual, the best case occurs when one of the S_i terms has only one literal. In this case, only one transistor is used in clock replacement.

4. DESIGN OF A 16-B BARREL SHIFTER

To investigate the advantages of the D^3L concept at a system-building-block level, we implement a 16-bit barrel shifter in static, dynamic and D^3L styles, and then compare their characteristics.

The desired barrel shifter has a combinational structure with two main modules: the shift-and-rotate array and the control section. While the shift-and-rotate array shifts and rotates the input data, its controlling signals which determine shift value, come from the control section [5].

The shift-and-rotate array occupies most of the area and determines the critical path delay of the barrel shifter while only a small percentage of the chip area belongs to the control section. For this reason, in the three designs simulated, only the shift-and-rotate arrays are compared.

4.1 Implementation of the shift-and-rotate array

I. Static

This module has been designed using six stages having sixteen cells each, as illustrated in Fig. 8a. The basic cell used in this array is an AO22 gate that is called *qmux* (Fig. 8b). The cell implements the function $F = Ci_1.In_1 + Ci_2.In_2$.

The first and sixth stages of the array are used for shifting or rotating data from 0 to 15 positions to the right. The four central stages of the array are used for shifting or rotating data from 0 to 15 positions to the left.

II. Dynamic

The potential advantage of a dynamic implementation rather than one using standard CMOS is a reduction both of the area and the total circuit delay. Since the shift-and-rotate array has non-inverting properties, the Domino style can be used directly in its dynamic implementation. In the precharge phase of Domino logic, inputs of each gate must be set to the inactive state. This means that in the precharge phase, all four inputs of each *qmux* cell must be set to a low level. This is easily done using the clock signal to arrange that the outputs of the control section are forced to the low level, and recognizing that the In_i inputs of each *qmux* cell are also set to the low level through the previous cell inverter. Such a dynamic cell configuration is shown in Fig. 9a.

III. D^3L

In order to eliminate the clock signals from the *qmux* cells, we must substitute them with suitable combinations of inputs. Each of the four groups (In_1,In_2), (Ci_1,Ci_2), (In_1,Ci_2) and (In_2,Ci_1) can be considered in a replacement strategy. For the first stage, we choose the (In_1,In_2) group so that stages of the barrel shifter are precharged with the external inputs $(I_0\text{-}I_{15})$. For this purpose, these inputs must be set to the low level in the precharge phase. These low values in the first stage create a precharge wave, which is transferred to the outputs through the second stage, third stage, and so on.

In a similar manner, control inputs (Ci_1,Ci_2) of all *qmux* cells that originate from the control module are set to the low level during the precharge phase. For each *qmux* cell, whenever the condition $In_1=In_2=$'0' is satisfied, the corresponding cell is precharged. A possible high transition on each input In_1 or In_2 initiates the evaluation phase. The configuration of the resulting *qmux* cell in D^3L design is shown in Fig. 9b.

4.2 Simulation Results

I. Power Consumption

For estimating the power consumption, a VHDL model was prepared for each of the three basic implementations. In each model, for every rising edge of any input of any *qmux* cell, a shared variable called Event Count or EC is incremented. This increment is weighted by consideration of the input capacitance. For example in the basic D^3L design, a rising edge for the inputs In_1 or In_2 results in EC incrementation by 3 ($Wp=2Wn$), whereas for inputs Ci_1 or Ci_2, this incrementation is by only one. For each of the three designs, 10,000 random input vectors with random shift numbers were generated and the simulation is performed. Total event counts weighted by input-capacitance considerations are shown in Table 1.

Table 1. Power estimation using activity factors (EC).

Circuit Type	EC	Ratio to static
Static	3,766,401	1
D^3L	4,993,912	1.33
Dynamic	5,908,815	1.57

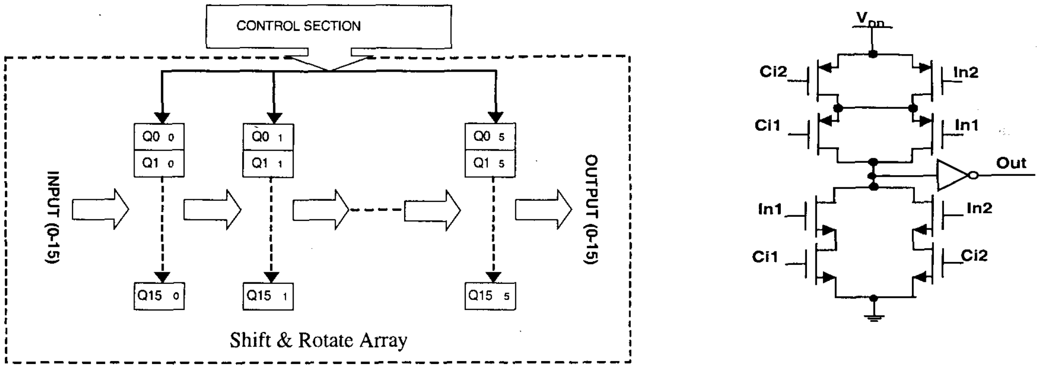
As we expected, dynamic and D^3L have greater activity factors, since the entire circuit is established in a predefined state in the precharge phase. As well, since in a dynamic implementation, input clock capacitance must be discharged every cycle, the event count further increases. A look at the Table 1 shows that the D^3L shifter has 33% and the dynamic shifter has 57% increase in event count when compared to a static design. Note that in this simulation, routing capacitance of the clock signal in the dynamic design is not considered.

II. Speed

At the transistor level, all three cells have been simulated with $V_{DD}=3.3v$, $L_p=L_n$ and $W_p=2W_n$ in a CMOS 0.5μm process. The simulation results are listed in Table 2.

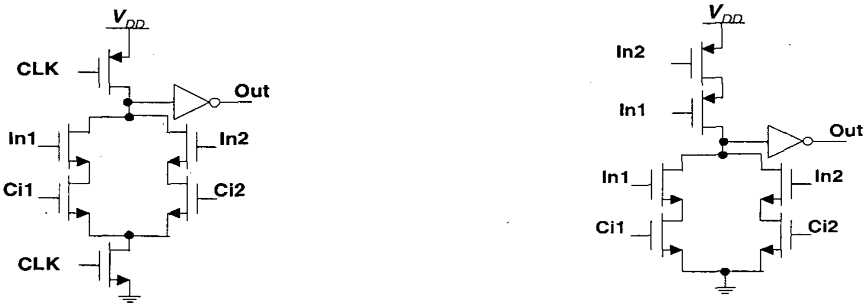
Table 2. SPICE simulation results (times in Pico-seconds).

Circuit Type	T_{rise}	T_{fall}	Tp_{LH}	Tp_{HL}
Static	200	170	210	210
Dynamic	140	95	185	120
D^3L	170	150	185	185



(a) (b)

Fig. 8. (a) Schematic of the barrel shifter (b) Transistor-level representation of a *qmux* cell.



(a) (b)

Fig. 9. *qmux* cells in (a) dynamic logic (b) D^{3}L.

Since only one PMOS device precharges the dynamic cell, its precharge time (Tp_{HL}) has the lowest value. In the evaluation phase of the dynamic cell, there are three series NMOS devices between the output and GND, while for D^{3}L, there are only two. However, in D^{3}L design, the input-output path consists of In_i inputs, which drive both PMOS and NMOS devices in each *qmux* cell. However in the dynamic style, the capacitance of each In_i is reduced to that of only one NMOS switch. These two opposing factors tend to equalize the evaluation delays of dynamic and D^{3}L designs. Note, of course, that this is in the local situation where the buffers and the capacitance of the clock signal in the dynamic design are ignored.

III. Robustness

It is clear that the static implementation gives us a high robustness. After this, D^{3}L takes its place ahead of the dynamic style. The reason is that at the rising edge of the clock, all dynamic *qmux* cells are disconnected from *Vdd* and remain in the float state. In this situation, noise immunity decreases through charge sharing and charge leakage. In contrast, the D^{3}L *qmux* goes into the evaluation phase only when one of the inputs In_1 or In_2 (replacing the clock) assigns a high value. Since normally the In_1 or In_2 inputs of D^{3}L *qmux* cells have greater delay than control signals Ci_1 and Ci_2, charge sharing and charge leakage problems are reduced compared to the situation in dynamic *qmux* cells.

5. CONCLUSION

A new dynamic logic style called Data-Driven Dynamic Logic, or D^{3}L has been discussed in this paper. D^{3}L is an improved type of dynamic logic in which precharge and evaluation phases are performed under control of input data and without an explicit clock. This mode of operation, besides eliminating the clock signal, allows inverting functions also to be implemented straightforwardly. For illustration, we have applied the D^{3}L concept to a barrel shifter. Simulation results show that we can obtain the same speed as acceptable in the Domino circuits but with a considerable reduction in power consumption.

6. REFERENCES

[1] J. M. Rabaey, *Digital Integrated Circuits*, Prentice Hall, 1996.

[2] R. H. Krambeck, "High-speed compact circuits with CMOS," *IEEE J. Solid-State Circuits*, vol. 17, no. 3, pp. 614-619, June 1982.

[3] N. Goncalvez and H. De Man, "NORA: A racefree dynamic CMOS technique for pipelined logic structures," *IEEE J. Solid-State Circuits*, vol. 18, no.3, pp. 261-266, June 1993.

[4] B. J. 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.

[5] R. Pereira, J. A. Michell and J. M. Solana, "Fully pipelined TSPC barrel shifter for high-speed applications," *IEEE J. Solid-State Circuits*, vol. 30, pp. 686-690, June 1995.