Data-Driven Dynamic Logic versus NP-CMOS Logic, A Comparison

R. Rafati[1], A. Z. Charaki[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@khorshid.ece.ut.ac.ir , fakhraii@chamran.ut.ac.ir , lfujino@cs.utoronto.ca

ABSTRACT

Data-Driven Dynamic Logic (D^3L) is an appropriate candidate for replacing conventional dynamic logic in many cases. In our previous paper, we have shown how a D^3L-based design can be used in place of a conventional Domino logic [1]. The designed circuit has been shown to have up to 30% less power dissipation compared to its Domino counterpart. This is in addition to the fact that no speed degradation was found. In this paper, after a brief introduction on NP-CMOS logic and D^3L, we show how to convert an NP-CMOS design to a D^3L one. Then the results of simulations performed on a 16-bit barrel shifter are demonstrated and compared for static, NP-CMOS and D^3L design styles.

1. INTRODUCTION

Historically, complementary logic came after NMOS and pseudo-NOMS styles to overcome static power consumption of NMOS logic family. However, CMOS logic comes with an area overhead, as a specific logic must be complementarily duplicated in both of a pull-up network (PUN) and a pull-down network (PDN). Improvements of CMOS technology and reduction of minimum feature sizes compensated part of this area overhead. However, one should note that logic duplication in CMOS has an effect on increased node capacitances, beside extra power consumption, that could not be ignored.

A major challenge is looking for solutions by which to avoid logic duplication in CMOS without static power consumption of NMOS logic family. Dynamic logic style provides an answer for this question [2]. In dynamic logic, logic function appears in only one of PDN or PUN branches, and a global clock signal provides the functionality of PUN or PDN branch. Thus, the circuit operation is divided in two distinct precharge and evaluate phases. In the precharge phase, charge is pumped from V_{DD} to the output node (M_p in Fig. 1), while M_e is turned off and the current path to ground is disconnected [2].

Compared to CMOS, dynamic logic can improve speed and reduce area. However, But these benefits do not come for free. Presence of a foot transistor increases logic evaluation time, the clock signal must drive every dynamic logic gates in addition to registers, and it will be faced with heavy loads. Besides this, it oscillates permanently resulting in high activity factor and considerable power consumption [3]. Moreover, for correct evaluation of dynamic gates, a set of conditions must be imposed on the inputs of every dynamic block. For example, in the Domino logic style [2], a static inverter follows every Φ-*N* block. This ensures that all inputs to the next logic block are set low after the precharge period. This means that it might be difficult to implement some functions and the designers must take additional procedures.

We summarize the advantages of dynamic logic styles compared to CMOS logic in two main categories. The first one is speed improvement and the second is area reduction. Its disadvantages are higher power consumption, especially on clock signal, presence of the foot transistor (Fig. 1) that adds some delay and power consumption, and a more difficult design process. We show that D^3L improves conventional dynamic logic functionality and reduces its disadvantages, namely, relaxes its excessive power consumption. Some details come in the next section.

2. DATA-DRIVEN DYNAMIC LOGIC

In this section, we describe D^3L design concept. In creation of conventional dynamic logic, a set of conditions is imposed on dynamic blocks. These conditions are arranged such that the logic transistors stay in *off* state during the precharge time. This condition is necessary for correct operation at the beginning of the evaluation phase and prohibits the output node from accidental discharge. In D^3L, we use these existing conditions to find a replacement for the clock signal.

For example, consider a 2-input *AND* gate in Domino logic, implemented as shown in Fig. 1. As noted in this figure, both of the inputs *A* and *B* are held at a low level in the precharge phase. Awareness of this usual restriction enables us to eliminate the clock signal as shown in Fig. 2.

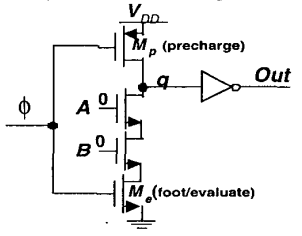

Fig. 1. A Domino *AND* gate.

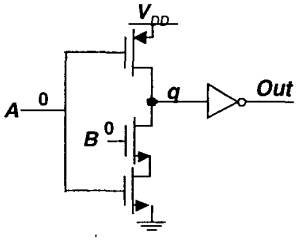

Fig. 2. A D^3L *AND* gate.

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 values of B, node q conditionally discharges.

3. IMPLEMENTATION OF VARIOUS FUNCTIONS IN NP-CMOS AND D^{3}L

In this part, we examine how an NP-CMOS design can be converted to a D^{3}L design. In NP-CMOS, instead of using a static inverter after each dynamic gate, the precharged output nodes are directly connected to PUNs, which are derived with $\bar{\Phi}$ [2]. For example, an *OR* gate in NP-CMOS is shown in Fig 3. Note that in contrast to Domino circuits, all of the inputs of PUN transistors must be set to a high level (*off* condition) in the precharge state ($\bar{\Phi}$=1). Thus a chain of succeeding Φ-*N* and $\bar{\Phi}$-*P* logic can be cascaded without any false evaluation.

For D^{3}L we can substitute one of two inputs A or B (Fig 4) in place of a $\bar{\Phi}$ signal. During the precharge phase, when the input A is high (NP-CMOS condition), node q is precharged low. When signal A makes a possible transition from high to low, the evaluation phase starts. At this phase, depending on the value of B, node q conditionally recharges.

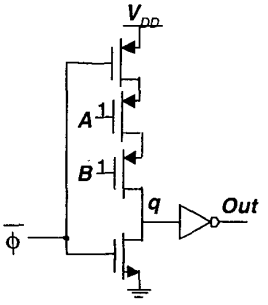

Fig. 3. An NP-CMOS *OR* gate.

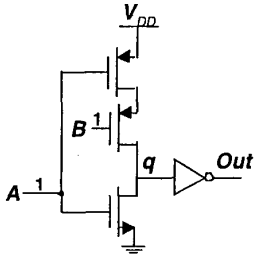

Fig. 4. A D^{3}L *OR* gate.

In general, for D^{3}L designs, when we have a function F in the sum-of-products form $F = \sum_{i=1}^{n} P_i$, then the minimum P_i (the P_i with a minimum number of literals) is selected in such a way that:

1- If in the precharge phase, all of inputs have a low value (the Domino condition) the minimum P_i is used to replace the clock in PUN and the main function is made in PDN [1].

2- If in the precharge phase, all of the inputs have a high value (the NP-CMOS condition) the minimum P_i is used to replace the $\bar{\Phi}$ in PDN and the main function is made in PUN. Examples of these operations are shown in Fig. 5 and Fig. 6.

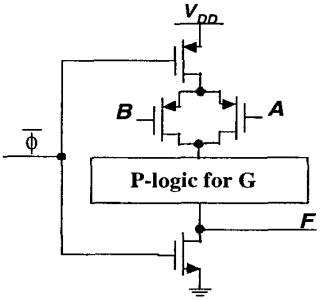

Fig. 5. NP-CMOS implementation of $\bar{F}$=AB+G.

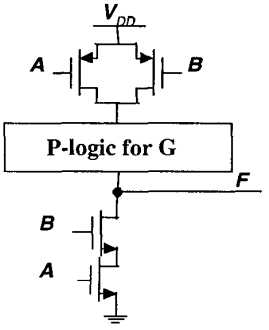

Fig. 6. D^{3}L implementation of $\bar{F}$=AB+G.

4. DESIGN OF A 16B BARREL SHIFTER

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

The desired barrel shifter has six columns, each with sixteen cells named as *qmux* [1]. Each *qmux* implements function $F = Ci_1.In_1 + Ci_2.In_2$. If Ci_1 and Ci_2 are complements of each other, then F behaves as a two-input multiplexer. These multiplexers are used for shifting the input data. If Ci_1=Ci_2=0 then F equals zero and this is especially useful for zero insertion in left shifting [5].

4.1 Static Implementation

In this section, we show how static implementation could be modified to better be prepared for NP-CMOS dynamic implementation. A straightforward implementation of every F function mentioned above requires an inverter at its output, as we have shown in [1]. For improving speed, one can eliminate

this inverter. Correspondingly, inverted outputs of F appear at the output of each *qmux*. Thus, $\overline{Out}$ is delivered to the next column of *qmuxes* (Fig. 7a). Note that if all of the inputs of a mux are inverted, then its output is also inverted. Therefore, correct result of the shift operation is restored after two successive inversions. This is true only for all of the cases in which $Ci_1 = \overline{Ci_2}$. However, for zero injection in left shifting, there are situations where Ci_1=Ci_2=0. In these cases, the output of corresponding *qmux* must be set to zero, ignoring the input values of that *qmux*. This requirement results in the static implementation of odd-column *qmuxes* as shown in Fig. 7b, which is dual of previous stage for even columns (Fig. 7a). Note that for zero injection in even columns, all control inputs Ci_1 and Ci_2 are set to low, and for odd columns, all control inputs Ci_1 and Ci_2 are set to high value. In this way, in even columns, the situation Ci_1=Ci_2=1, and in odd columns, the situation Ci_1=Ci_2=0 never occur.

4.2 NP-CMOS Implementation

The inverting function in even columns makes straightforward implementation of NP-CMOS without any problem. If the i'th even stage precharged high as shown in Fig. 7b, then the next $i+1$'th odd stage must be constructed with PUN and precharged to low with the aid of $\bar{\Phi}$. Since all of its inputs have high values (*off* state) at the onset of evaluation phase, no false evaluation occurrs. The resulting NP-CMOS structure is shown in Fig. 7c and Fig. 7d.

4.3 D^{3}L Implementation

After removing the clock signal, we must substitute it with a suitable combination of one or more inputs. For every design, we have either of two choices: 1. input precharging, and 2. control precharging. In the first scheme, the inputs In_1 and In_2 of each *qmux* cell, and in the second scheme, Ci_1 and Ci_2 control signals can be considered to replace the clock signals. Each of these has its own characteristics. For now, we select the first scheme in which inputs In_1 and In_2 replace clock of every NP-CMOS *qmux*.

At the even stages, the inputs In_1 and In_2 have low values in the precharge phase, and hence, we put the series combination of two PMOS in PUN (Fig. 7e). For odd stages, the inputs In_1 and In_2 have high values from the previous precharged *qmuxes*. Assume we replace the PDN logic with a series combination of two NMOS transistors. Then in the precharge phase (In_1=In_2=1) of the corresponding *qmux*, as in the $\bar{\Phi}$ section of NP-CMOS, the output node is precharged to low value. A possible transition on each of the In_1 or In_2 signals starts the evaluation phase on corresponding *qmux*. In this phase, based on input and control values, the output node either remains float at a low state or is set to high level via a PUN. Note that in D^{3}L, there is no need to $\bar{\Phi}$ signal. This greatly simplifies routing and decreases power consumption. The resulting D^{3}L structures of the barrel shifter are shown in Fig. 7e and Fig. 7f.

4.4 Simulation Results

A. Power Consumption

As in [1], for estimation of the power consumption, based on a VHDL model, we count the total event numbers, while considering relative input capacitances. For example, in a static design, every event on one input results in a total event count (EC) incrementation by 3 (assuming Wp=2Wn). For each of the three designs, the same random test vectors as in [1] are applied to the barrel shifter and simulation is performed. Total event counts weighted by input-capacitance considerations are shown in Table 1. Our previous results in [1] are also given for comparison.

As we expected, for elimination of one inverter from each gate output, the new design has some reduction in power consumption compared to the previous one reported in [1]. However, since about one-half of the *qmuxes* are made with PMOS switches, and in the precharge phase are set to a known state, NP-CMOS and D^{3}L designs have less EC reduction over static implementation, when compared to the previous design. As well, since in a NP-CMOS implementation, the Φ and $\bar{\Phi}$ signals oscillate unconditionally at every cycle, the event count further increases. A look at Table 1 shows that the NP-D^{3}L shifter has 64% and the NP-CMOS shifter has 122% increase in event count when compared to a static design. From this we can conclude that speed improvement of NP-CMOS logic comes with much more power consumption and this can be reduced by D^{3}L style.

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

	New results			Results of [1]		
Circuit Type	Static	D^{3}L	NP-CMOS	Static	D^{3}L	Domino
EC	2,503,428	4,110,130	5,555,429	3,766,401	4,993,912	5,908,815
Ratio to static	1	1.64	2.22	1	1.33	1.57

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

	P-stage following N-stage			N-stage following P-stage			Results of [1]		
Circuit Type	Static	D^{3}L	NP-CMOS	Static	D^{3}L	NP-CMOS	Static	D^{3}L	Domino
T_{rise}	450	340	150	455	420	420	200	170	140
T_{fall}	220	180	130	260	220	120	170	150	95
T_{PLH}	165	150^p	110^p	175	180^e	110^e	210	185^e	185^e
T_{PHL}	125	110^e	100^e	105	120^p	75^p	210	185^p	120^p

^{p}:precharge time, ^{e}:evaluate time.

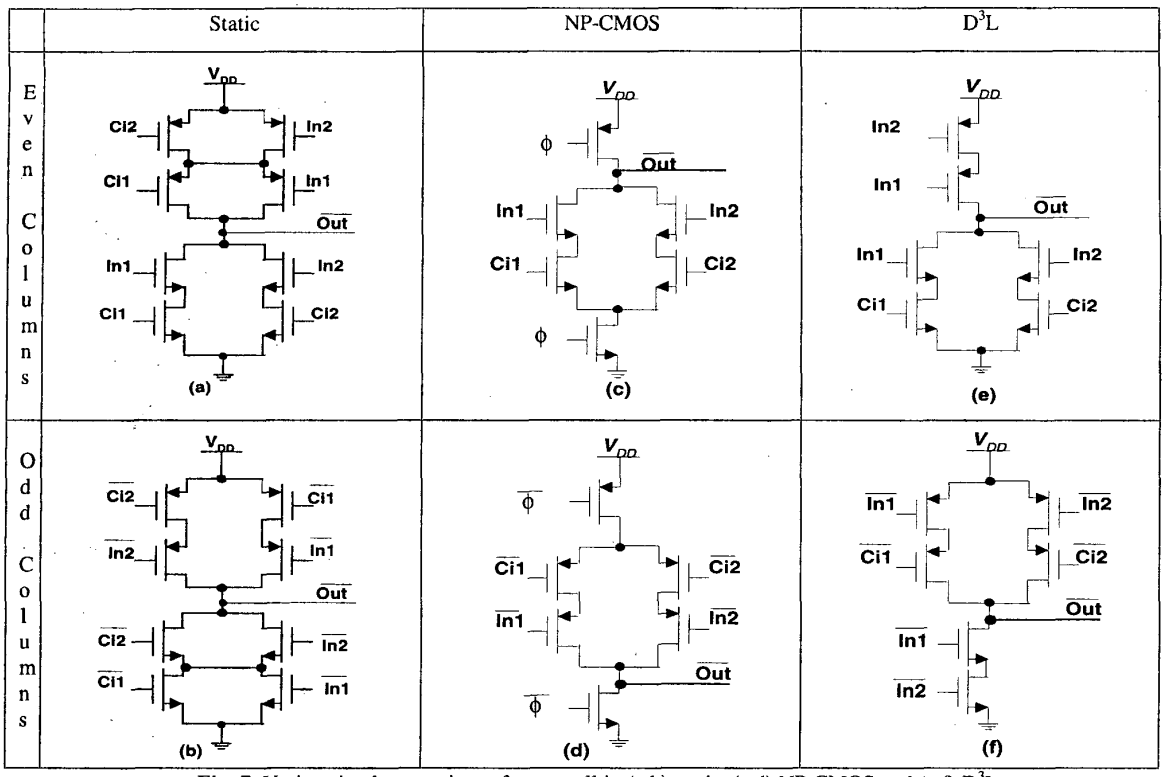

Fig. 7. Various implementations of *qmux* cell in (a,b) static, (c,d) NP-CMOS and (e,f) D^3L.

B. Speed

At the transistor level, we use the same test conditions (V_{DD}=3.3v, $L_p=L_n$, $W_p=2W_n$) as in [1] for a CMOS 0.5μm process. The simulation results are shown for different stages of PUN and PDN in Table 2. Beside this, the results of [1] are also shown for comparison.

Since only one PMOS device at even columns and one NMOS device at odd columns precharges each NP-CMOS cell, its precharge time has the lowest value. In the evaluation phase of the NP-CMOS cell, there are three series NMOS in PDN (three series PMOS in PUN for odd stages) devices between the output and GND, while for D^3L, there are only two. In D^3L design, the input-output path consists of In_i inputs, which drive both PMOS and NMOS devices in each *qmux* cell. However, in the NP-CMOS style, the capacitance of each In_i is reduced to that of only one NMOS (PMOS in odd stages) switch. The result of average evaluation time for even and odd stages is 105_{ps} and 145_{ps} for NP-CMOS and D^3L respectively. The results show a 30% reduction in NP-CMOS delay compare to D^3L, but we must take into account 35% more power consumption over D^3L. Note, of course, that this is in the local situation where the buffers and the capacitance of the Φ and $\bar{\Phi}$ signal in the NP-CMOS design are ignored.

5. CONCLUSION

This paper demonstrated how an NP-CMOS logic design could be converted to D^3L. First, we made necessary modifications on static implementation of a barrel shifter, and prepared it for implementation in NP-CMOS scheme. After this, correct sequence of Φ and $\bar{\Phi}$ sections was considered to build NP-CMOS logic structures. In this way, by elimination of an extra inverter, 45% ($105_{ps}/185_{ps}$) delay reduction was achieved when compared to previously reported Domino implementation [1]. We eliminated the global Φ and $\bar{\Phi}$ signals from NP-CMOS cells, and replaced them with a suitable combination of inputs. The resulting D^3L design has been shown to have 27% and 18% reduction in power consumption, compared to NP-CMOS and previous Domino implementations respectively.

6. REFERENCES

[1] R. Rafati, S. M. Fakhraie, K. C. Smith, "Low-Power Data-Driven Dynamic Logic," *ISCAS'* 2000, vol 1, pp. 752-755.

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

[3] 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.

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

[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.