

Static Implementation of Quasi-Delay-Insensitive Pre-Charge Half-Buffers

Liang Zhou and Scott C. Smith
Department of Electrical Engineering
University of Arkansas
Fayetteville, Arkansas, USA
lxz001@uark.edu and smithsco@uark.edu

Abstract— In the literature, quasi-delay-insensitive (QDI) asynchronous circuits utilizing Pre-Charge Half Buffers (PCHB) are based on either dynamic or semi-static implementations. In this paper, a static implementation of PCHB is presented, and compared to previous PCHB architectures and static NULL Convention Logic (NCL), using a full adder design. Transistor level simulation shows that the static PCHB architecture is faster, more energy-efficient, and can operate correctly at much lower supply voltage than its semi-static counterpart, although it requires more transistors. Compared to static NCL, static PCHB is faster and requires fewer transistors, but is less energy-efficient.

Keywords—Pre-Charge Half-Buffer (PCHB); NULL Convention Logic (NCL); asynchronous; delay-insensitive

I. INTRODUCTION

The development of synchronous circuits currently dominates the semiconductor design industry. However, there are major limiting factors to the synchronous, clocked approach, including the increasing difficulty of clock distribution, increasing clock rates, decreasing feature size, increasing power consumption, timing closure effort, and difficulty with design reuse. Asynchronous (clockless) circuits require less power, generate less noise, produce less electromagnetic interference (EMI), and allow for easier reuse of components, compared to their synchronous counterparts, without compromising performance, which is why the International Technology Roadmap for Semiconductors (ITRS) predicts a likely shift from synchronous to asynchronous design styles in order to increase circuit robustness, decrease power, and alleviate many clock-related issues.

Quasi-delay-insensitive (QDI) NULL Convention Logic circuits (NCL) [1] and QDI Pre-Charge Half-Buffers (PCHB) [2] are two primary delay-insensitive asynchronous paradigms. In NCL, computation and registration are separated and implemented independently, while PCHB integrates computation and registration. The NCL and PCHB protocols are also different in that PCHB uses the inputs and outputs of a registration stage to generate acknowledge signals, while NCL only uses registration outputs.

Both NCL and PCHB utilize hysteresis state-holding functionality to attain delay-insensitivity. Hysteresis can be implemented in one of three ways: dynamic, semi-static, or static [3]. The dynamic implementation relies on a gate's

output capacitance to hold the previous state; it does not use any feedback. Hence, the gate capacitance will lose its charge, and therefore the previous state, if left idle for long enough; however, it is fastest. Therefore, it is not delay-insensitive and its application is limited. Semi-static hysteresis utilizes a weak feedback inverter; however, this must be carefully sized to balance current strengths, hence, is more vulnerable to supply voltage and process variations than the static implementation. Static hysteresis feeds the output back into additional pull-up and pull-down networks specifically designed for state-holding. It is the most robust, but it requires the most transistors.

In the literature, PCHB circuits utilize either dynamic or semi-static hysteresis. Although sometimes static C-elements are utilized instead of semi-static C-elements to generate acknowledge signals, no scheme currently exists that also implements computational blocks statically. This paper presents a static implementation of PCHB, and compares the new QDI architecture to semi-static QDI PCHB and static NCL, for a dual-rail full adder circuit.

Section II provides an overview of NCL and PCHB, while Section III details the proposed static QDI PCHB architecture. Section IV compares the various implementations; and Section V provides conclusions and future work.

II. PREVIOUS WORK

A. Introduction to NCL

NCL circuits [1] utilize multi-rail logic, such as dual-rail, to achieve delay-insensitivity. A dual-rail signal, D , consists of two wires, D^0 and D^1 , which may assume any value from the set $\{\text{DATA0}, \text{DATA1}, \text{NULL}\}$. The DATA0 state ($D^0 = 1, D^1 = 0$) corresponds to a Boolean logic 0, the DATA1 state ($D^0 = 0, D^1 = 1$) corresponds to a Boolean logic 1, and the NULL state ($D^0 = 0, D^1 = 0$) corresponds to the empty set meaning that the value of D is not yet available. The two rails are mutually exclusive, such that both rails can never be asserted simultaneously; this state is defined as an illegal state.

NCL circuits are comprised of 27 fundamental gates [4]. These 27 gates constitute the set of all functions consisting of four or fewer variables. The primary type of threshold gate, shown in Fig. 1, is the TH_{mn} gate, where $1 \leq m \leq n$. TH_{mn} gates have n inputs. At least m of the n inputs must be asserted before the output will become asserted. NCL threshold gates are designed with *hysteresis* state-holding capability such that

all asserted inputs must be de-asserted before the output will be de-asserted. Therefore, a TH_mn gate is equivalent to an n-input C-element [5] and a TH1n gate is equivalent to an n-input OR gate. NCL threshold gates may also include a *reset* input to initialize the output. Circuit diagrams designate resettable gates by either a *d* or an *n* appearing inside the gate, along with the gate's threshold. *d* denotes the gate as being reset to logic 1; *n*, to logic 0. The static NCL threshold gate is shown in Fig. 2, where hold0 = set' and hold1 = reset'.

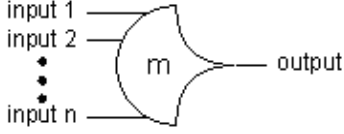


Figure 1. TH_mn threshold gate.

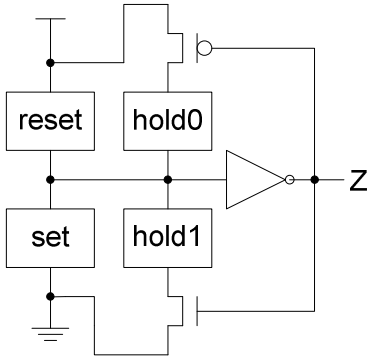


Figure 2. Static NCL threshold gate design.

QDI circuits communicate using request and acknowledge signals, K_i and K_o , respectively, to prevent the current DATA wavefront from overwriting the previous DATA wavefront, by ensuring that the two DATA wavefronts are always separated by a NULL wavefront. The acknowledge signal from the receiving circuit is the request signal to the sending circuit. When the receiver circuit latches the input DATA, the corresponding K_o signal will be logic0, indicating a *request-for-NULL* (*rfn*); and when it latches the input NULL, the corresponding K_o signal will be logic1, indicating a *request-for-DATA* (*rfd*). When the sending circuit receives a *rfd/rfn* on its K_i input, it will allow a DATA/NULL wavefront to be output, respectively. Request/acknowledge signals are usually generated using C-elements.

In NCL, computation and registration are separated and implemented independently, and only outputs of a registration stage are used to generate acknowledge signals.

B. Introduction to PCHB

PCHB circuits [2] are designed at the transistor level, utilizing dynamic CMOS logic, instead of targeting a predefined set of gates like NCL. PCHB circuits have dual-rail data inputs and outputs, and combine combinational logic and registration together into a single block, yielding a very fine-grain pipelined architecture.

The semi-static implementation of QDI PCHB is shown in Fig. 3. The dual-rail output is initially pre-charged to NULL. When request (R_{ack}) and acknowledgement (L_{ack}) are both *rfd*,

the specific function will evaluate when the inputs, X and/or Y , become DATA, causing the output, F , to become DATA. L_{ack} will then transition to *rfn* only after all inputs and the output are DATA. When R_{ack} is *rfn* and L_{ack} is *rfd*, or vice versa, the output will be floating, so weak inverters must be used to hold the current output value to maintain delay-insensitivity. After both R_{ack} and L_{ack} are *rfn*, the output will be pre-charged back to NULL. After all inputs become NULL and the output changes to NULL, L_{ack} will change back to *rfd*, and the next DATA wavefront can evaluate after R_{ack} becomes *rfd*. Note that R_{ack} and L_{ack} are equivalent to K_i and K_o , respectively.

Different from NCL, in PCHB computation and registration are integrated, and both the inputs and outputs of a registration stage are used to generate acknowledge signals.

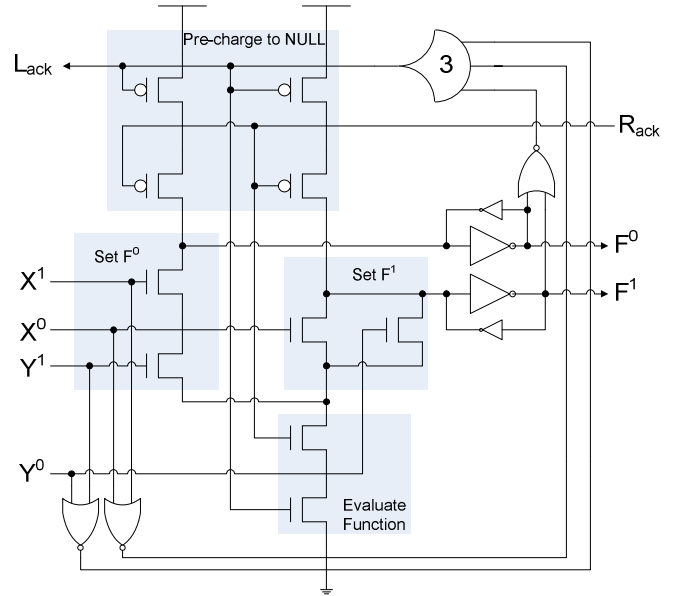


Figure 3. Semi-static PCHB NAND2 circuit.

III. STATIC IMPLEMENTATION OF PCHB

The proposed static implementation of PCHB is illustrated in Fig. 4. Just like in static NCL, the *hold0* block is the complement of its corresponding *set* block. *Sleep* is generated by K_i and K_o with a TH22 gate and an inverter. When K_i and K_o are both *rfn*, *sleep* is asserted and both rails of the output are deasserted. When K_i and K_o are both *rfd*, *sleep* is deasserted and the specific function will evaluate when the inputs become DATA, causing the output, Z , to become DATA. After Z becomes DATA, feedback transistors hold Z at DATA until *sleep* is asserted, causing Z to transition back to NULL. Hence, if Z is DATA and *sleep* is deasserted, Z will not become NULL even if all of the dual-rail inputs are NULL, until K_i becomes *rfn*, which is required by the PCHB handshaking protocol [6].

Fig. 5 shows the implementation of sum, S , for a static PCHB full adder. The *set* blocks are $S^1 = X^1Y^1Cin^1 + X^1Y^0Cin^0 + X^0Y^1Cin^0 + X^0Y^0Cin^1$; $S^0 = X^0Y^0Cin^0 + X^0Y^1Cin^1 + X^1Y^0Cin^1 + X^1Y^1Cin^0$. The *hold0* blocks are $S^{1'} = X^1Y^1Cin^1 + X^1Y^0Cin^0 + X^0Y^1Cin^0 + X^0Y^0Cin^1$; $S^{0'} = X^0Y^0Cin^0 + X^0Y^1Cin^1 + X^1Y^0Cin^1 + X^1Y^1Cin^0$.

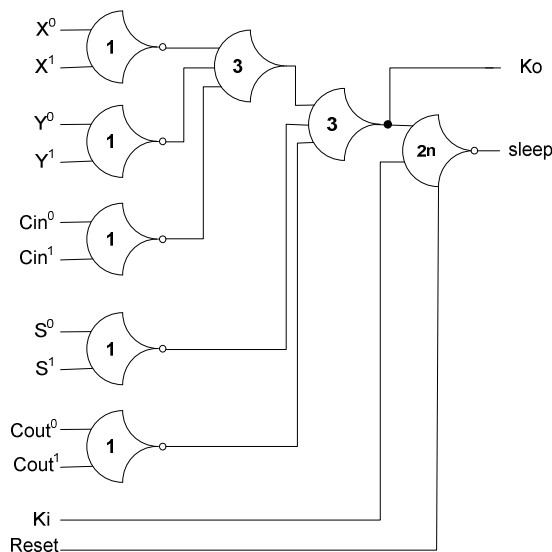


Figure 7. Generation of *sleep*/acknowledge signals for static PCHB full adder.

IV. SIMULATION RESULTS

Five implementations of the full-adder were simulated at the transistor level with Cadence's UltraSim simulator running a VerilogA controller in mixed-signal mode, using the 1.2V IBM cmos10lpe 65nm CMOS process, and compared in terms of number of transistors, average DATA+NULL cycle time (T_{DD}), energy per operation, and minimum supply voltage at which the circuit operates correctly, as listed in Table I. The dynamic and semi-static PCHB designs are from [7].

For fair comparison, all transistors are minimum length and width, except for the weak inverters, which require multiple times minimum length. The weak inverter transistor lengths are also chosen to be as small as possible, but still achieve correct circuit functionality. An input register and corresponding completion logic are included in the static NCL implementation to make yield a fair comparison.

As expected, the dynamic PCHB implementation gives the best performance and requires the least transistors. However, it is not delay-insensitive and its application is limited, because states will be lost without periodical refreshing and it is vulnerable to charge sharing problems. The proposed static PCHB is 24% faster and 8% more energy efficient than semi-static PCHB with static C-elements, and 143% faster and 36% more energy-efficient than semi-static PCHB with semi-static

C-elements, although it requires 20% and 29% more transistors, respectively. Additionally, the proposed static PCHB can operate correctly at a much lower supply voltage than its semi-static counterparts. Compared to static NCL, the proposed static PCHB has 24% fewer transistors and is 59% faster, although it is 17% less energy efficient.

V. CONCLUSION AND FUTURE WORK

This paper describes a static QDI PCHB architecture, which is shown to be faster and more energy efficient than the current state-of-the-art semi-static PCHB in the literature, and can operate correctly at a much lower supply voltage, although it requires more transistors. Note that the length of the weak inverters may be increased to reduce T_{DD} , energy/operation, and minimum operational supply voltage, at the expense of additional area; however, static PCHB is still better in all categories despite the increased size of the weak inverters. Compared to static NCL, static PCHB requires fewer transistors and is faster, although it is less energy efficient.

Future work consists of implementing and fabricating large designs using this new static PCHB architecture for comparison with the previous work.

REFERENCES

- [1] K. M. Fant and S. A. Brandt, "NULL Convention Logic: A Complete and Consistent Logic for Asynchronous Digital Circuit Synthesis," *International Conference on Application Specific Systems, Architectures, and Processors*, pp. 261-273, 1996.
- [2] A. J. Martin and M. Nystrom, "Asynchronous techniques for system-on-chip design," *Proceedings of the IEEE*, pp. 1089-1120, Vol. 94/6, 2006.
- [3] M. Shams, J. C. Ebergen, and M. I. Elmasry, "Modeling and comparing CMOS implementations of the C-element," *IEEE Transactions on VLSI Systems*, vol. 6, no. 4, pp. 563-567, 1998.
- [4] Gerald E. Sobelman and Karl M. Fant, "CMOS Circuit Design of Threshold Gates with Hysteresis," *IEEE International Symposium on Circuits and Systems (II)*, pp. 61-65, 1998.
- [5] D. E. Muller, "Asynchronous Logics and Application to Information Processing," in *Switching Theory in Space Technology*, Stanford University Press, pp. 289-297, 1963.
- [6] R. O. Ozdag and P. A. Beerel, "High-speed QDI asynchronous pipelines," *Int. Symp. Adv. Res. Asynchronous Circuits Syst.*, pp. 13-22, 2002.
- [7] A. J. Martin, A. Lines, R. Manohar, U. Cummings, and M. Nystroem, "Pipelined Asynchronous Processing," U.S. Patent: 6,658,550, Dec. 2, 2003.

TABLE I. PCHB AND NCL COMPARISONS

	# Transistors	T_{DD} (ps)	Energy/Operation (fJ)	Minimum Supply Voltage (V)
Static PCHB	132	463	28.9	0.22
Static NCL	164	735	24.7	0.22
Dynamic PCHB	68	340	15.5	0.74
Semi-static PCHB with Semi-Static C-elements	102	1127	39.2	1.08
Semi-static PCHB with Static C-elements	110	575	31.1	0.88