High-speed Hybrid Threshold-Boolean Logic

Marius Padure, Sorin Cotofana, Stamatis Vassiliadis
Computer Engineering Laboratory
Delft University of Technology
Mekelweg 4, 2628CD Delft, The Netherlands
Email: {marius,sorin,stamatis}@ce.et.tudelft.nl

Abstract— **In this paper we propose a high-speed hybrid Threshold-Boolean logic style suitable for Boolean symmetric functions implementation. First, we present a depth-2 hybrid implementation scheme for arbitrary symmetric Boolean functions, based on differential Threshold logic gates as circuit style. Finally, we present the hybrid logic design of a $7/3$ counter. The simulation results, suggest that the hybrid $7/3$ counter designed in $0.25\mu m$ CMOS technology, achieves between 25% and 53% higher speed when compared with traditional Threshold logic and Boolean logic counterparts, at expense of between 52% and 67% transistors.**

Keywords—**computer arithmetic, Threshold logic, symmetric functions, counters**

I. Introduction

The increasing demand for high-speed computer arithmetic in hardwired processors have shifted the research efforts toward highly customized alternative circuit techniques and specific arithmetic algorithms. Among them, Threshold logic (TL) has received increasing attention in recent years.

Threshold logic (TL) originally emerged in the early 60's as a generalized theory of logic gates, which includes conventional Boolean logic (BL) as its subset [5]. More formally, a Threshold Logic Gate (TLG) is defined as an n-input processing element such that its output performs the following Boolean function[1]:

$$F(X) = sgn\{\mathcal{F}(X)\} = \begin{cases} 1, & if \;\; \mathcal{F}(X) \geq 0 \\ 0, & if \;\; \mathcal{F}(X) < 0 \end{cases} \qquad (1)$$

$$\mathcal{F}(X) = \sum_{i=0}^{n-1} \omega_i \cdot x_i - T \qquad (2)$$

where $X = \{x_0, x_1, \ldots, x_{n-1}\}$, $\Omega = \{\omega_0, \omega_1, \ldots, \omega_{n-1}\}$ and T are the set of Boolean input variables, the set of fixed signed integer weights associated with data inputs, and the fixed signed integer threshold, respectively [5].

TL is fundamentally more powerful that Boolean logic since the TL gate can perform more complex and wider functions than the usual Boolean CMOS gates (e.g., Nand, Nor, Invert) can. Several recent theoretical investigations [7], [8] have indicated that computer arithmetic building blocks (e.g., adders and multipliers) can be implemented in TL with smaller number of logic gates and fewer logic stages when compared with traditional Boolean logic counterparts.

In this paper a novel high-speed Hybrid Threshold-Boolean Logic (HTBL), suitable for fast Boolean symmetric functions implementation, is presented. High-speed design is addressed in both electrical and logical directions. Electrical improvements are achieved by resorting to a fast differential latch-based TL gate [6] design style. Logical improvements are achieved via the introduction of a novel depth-2 hybrid Threshold-Boolean logic implementation for arbitrary symmetric Boolean functions. In order to evaluate the potential performance of the hybrid scheme the design of a HTBL $7/3$ parallel counter is presented. The simulation results, suggest that the hybrid $7/3$ counter designed in $0.25\mu m$ CMOS technology, achieves between 25% and 53% higher speed when compared with traditional Threshold logic and Boolean logic counterparts, at expense of between 52% and 67% transistors.

The paper is organized as follows: Section 2 briefly reviews the differential TL gate employed by the hybrid scheme; Section 3 introduces the hybrid depth-2 implementation scheme for arbitrary symmetric Boolean functions. Section 4 presents the complete design of a HTBL $7/3$ parallel counter. Moreover, in this Section there are presented the simulation results and comparisons in terms of delay and estimated area of this parallel $7/3$ counter implemented in traditional Boolean, Threshold, and HTBL. Section 5 presents some concluding remarks.

II. Differential TL gate description

The differential TL gate schematic [6] and its basic symbols utilized further in the paper are depicted in Figure 1. The gate is in principle a clocked differential cascode voltage switch (DCVS) circuit, operated with a single phase clock. It comprises a fast latched comparator and two parallel-connected sets of unit nMOS transistors, referenced herein as input data bank and threshold mapping bank (as its transistors have usually the gates hardwired to ground or power supply). The gate presented in Figure 1 has $\omega_i = 1$ for every input while the same holds true for every threshold mapping input (T_i). The total conductances of the transistor banks are compared each other by the latched comparator and therefore the output Y is logic one if the current generated by the data bank is greater than the current generated by the threshold mapping bank and logic zero otherwise. Please note that, by design, the data bank is prevented to have similar conductance with the threshold mapping bank, when the threshold is reached, since an nMOS transistor with weight 0.5 is always on. This prevents the latch comparator entering in a metastable state.

As described in [6], the TL gate from Figure 1 has several potential advantages over other CMOS TL gates, e.g., [3], which makes it suitable for high-speed symmetric functions designs, as follows:

- the gate achieves high-speed of operation since the latched comparator is very fast;
- the gate is suited for dual-rail implementations since it provides both Y and $\overline{Y}$ simultaneously;

[1] All the operators are algebraic.

Fig. 1. 7-input differential latch-based TL gate schematic and its symbols

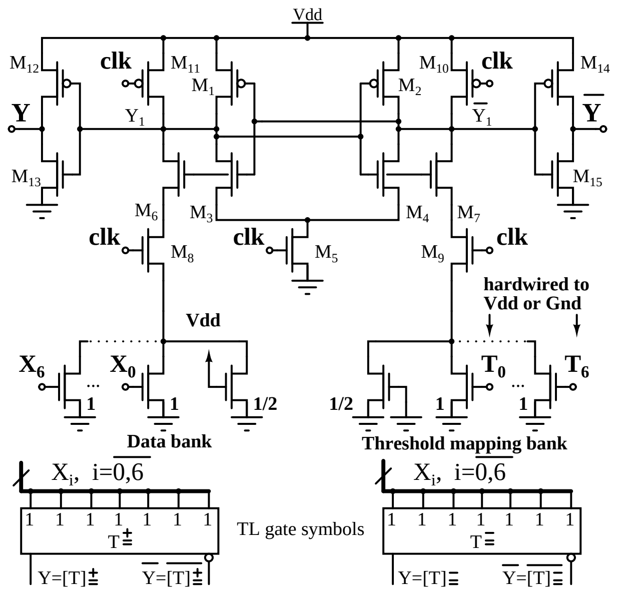
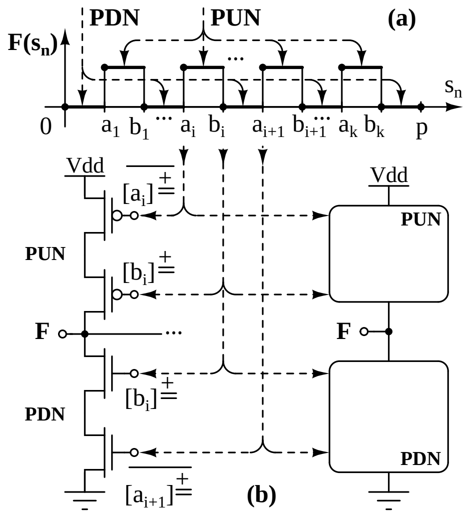


Fig. 2. a) Arbitrary symmetric function - interval description; b) synthesis rule example for Boolean stage



During the next section we use the following notations: n is the total number of primary inputs; $s_n = \Sigma_{i=0}^{n-1} x_i$, the total number of true inputs; $[T]_{=}^{+}$, $[T]_{=}^{-}$, $T = \overline{0,n}$, (as TL symbols from Figure 1 suggest for $n = 7$) denote $sgn\{s_n - T\}$ and $sgn\{T - s_n\}$ respectively; $'+'$ and $'\cdot'$ operators are the algebraic sum and product while $'\vee'$ and $'\wedge'$ denote a logical $'OR'$ and $'AND'$ operation, respectively.

III. High-speed hybrid logic

In this Section we present a depth-2 hybrid implementation scheme for arbitrary symmetric Boolean functions, based on differential Threshold logic gates as circuit style. We introduce first the definition of Boolean symmetric functions.

Definition 1 *A Boolean function of n variables, $F(x_1, \dots x_n)$, is called symmetric iff (if and only if) it is invariant to any permutation σ of $\langle 1, 2, \dots, n\rangle$ of its inputs $F(x_1, \dots x_n) = F(x_{\sigma(1)}, \dots x_{\sigma(n)})$.*

Because any input vector X with exactly i ones is a permutation of any other vector with exactly i ones, it can be said that F is symmetric *iff* $F(X)$ depends only on the number of ones in the input vector X, s_n. Consequently, instead of using a truth table with 2^n entries, a symmetric function can be described by stating the ranges of s_n for which the function is true [5], [1]. Any symmetric function can be specified either as a reunion of intervals for which $F = 1$, called symmetric function *on-set*, or as a reunion of intervals for which $F = 0$, called symmetric function *off-set*. Therefore, a symmetric Boolean function with k intervals in which $F = 1$, has the on-set $F^1(s_n) = \bigcup_{i=1}^{k} [a_i, b_i)$. Similarly, its off-set is the reunion of complementary intervals $F^0(s_n) = [0, n] - F^1(s_n) = \bigcup_{i=0}^{k} [b_i, a_{i+1})$, with $b_0 = 0$ and $a_{k+1} = n$.

We denote by *interval description*, both the on-set and off-set of the symmetric function. The interval description of an arbitrary symmetric function is depicted in Figure 2.a, where by the dot sign, $\bullet$, we mean a closed interval. The following theorem introduces the hybrid Threshold-Boolean symmetric function implementation scheme.

Theorem 1 *Given any arbitrary symmetric Boolean function of n variables, $F(s_n)$, having k intervals in its on-set, it can be implemented with $2k$ fan-in n TL gates and one AND-OR Boolean gate, in two stages as follows:*

Stage 1: the first stage comprises a total of $2k$ fan-in n differential TL gates, as those presented in Figure 1, computing in parallel $[a_i]_{=}^{+}$, $[b_i]_{=}^{+}$, $i = \overline{1,k}$, and their complements ($a_i, b_i \in [0, n]$).

Stage 2: the second Boolean stage receives the dual-rail outputs of the first stage and computes the function on-set using the following Boolean Equation:

$$F^1(s_n) = [a_1]_{=}^{+} \wedge \overline{[b_1]_{=}^{+}} \vee \dots \vee [a_k]_{=}^{+} \wedge \overline{[b_k]_{=}^{+}}. \quad (3)$$

Proof: Assume $s_n \in [a_i, b_i)$, for a specific $i = \overline{1,k}$. In this case $[a_j]_{=}^{+} = 0$ for $j = \overline{1, i-1}$, $[a_j]_{=}^{+} = 1$ for $j = \overline{i,k}$, $\overline{[b_j]_{=}^{+}} = 1$ for $j = \overline{1,i}$, $\overline{[b_j]_{=}^{+}} = 0$, for $j = \overline{i+1,k}$. Therefore $[a_i]_{=}^{+} \wedge \overline{[b_i]_{=}^{+}} = 1$, all $[a_j]_{=}^{+} \wedge [b_j]_{=}^{+} = 0$, $j \neq i$ and it follows that only one AND term in the Boolean Equation (3) is one, which implies $F^1(s_n) = 1$, as it should.

Assume now that there is no i, $i = \overline{1,k}$ such that $s_n \in [a_i, b_i)$. In this case there are three possibilities:

a) $s_n \in [b_i, a_{i+1})$ then $[a_j]_{=}^{+} = 0$ for $j = \overline{1,i}$, $[a_j]_{=}^{+} = 1$ for $j = \overline{i+1,k}$, $\overline{[b_j]_{=}^{+}} = 1$ for $j = \overline{1,i}$, $\overline{[b_j]_{=}^{+}} = 0$, for $j = \overline{i+1,k}$. Therefore $F^1(s_n) = 0$.

b) $s_n \in [0, a_1)$ then $[a_j]_{=}^{+} = 0$ for $j = \overline{1,k}$, $\overline{[b_j]_{=}^{+}} = 1$ for $j = \overline{1,k}$. Consequently $F^1(s_n) = 0$.

c) $s_n \in [b_k, n)$ then $[a_j]_{=}^{+} = 1$ for $j = \overline{1,k}$, $\overline{[b_j]_{=}^{+}} = 0$ for $j = \overline{1,k}$. Therefore $F^1(s_n) = 0$.

On the first stage, we compute each "rising transition", $[a_i]_{=}^{+}$ and "falling transition", $\overline{[b_i]_{=}^{+}}$, $i = \overline{1,k}$, i.e., we use $2k$ TL gates[2]. On the second stage we compute according to Equation (3) the symmetric function on-set, using a single AND-OR gate. □

[2]Please note that TL gates computing $[0]_{=}^{+}$ and $[n]_{=}^{+}$ are not needed in the first stage for the computation of off-set intervals $[0, a_1)$ and $[b_k, n]$ as there are no "transitions" in $s_n = 0$ and $s_n = n$, with respect to the interval description depicted in Figure 2. However, there are symmetric Boolean functions having such kind of transitions (e.g., parity).

Speaking from the circuit point of view, the design of the Boolean stage is a key point for achieving high-speed symmetric function implementations. Such an AND-OR Boolean gate has in CMOS a pull-up network (PUN) and a pull-down network (PDN). In HTBL designs PUN is responsible with the implementation of the symmetric function on-set while PDN implements its off-set.

Theorem (1) suggests an implementation method for both PUN and PDN from this Boolean gate. Each and every on-set/off-set interval is implemented by 2 serial connected MOS transistors, as depicted in Figure 2.b, for the particular case of the intervals $[a_i, b_i)$ and $[b_i, a_{i+1})$. Please note, the outer intervals $[0, a_1)$ and $[b_k, n]$ are implemented each with only one transistor in PDN as the gates $[0]^{+}_{=}$ and $[n]^{+}_{=}$ are not needed. Since generally, the interval descriptions are basically reunions of intervals and since each interval is implemented as in Figure 2.b, PUN and PDN can be extended easily by connecting in parallel such sets of no more than two series transistors.

Referring to Figure 2.b, the PUN and PDN work as follows: assuming that $s_n \in [a_i, b_i)$, it follows that $\overline{[a_i]^{+}_{=}} = 0$ and $[b_i]^{+}_{=} = 0$, therefore both pMOS transistors presented in Figure 2 are on and $F(s_n) = 1 = F^1(s_n)$, as it should. Similarly, assuming that $s_n \in [b_i, a_{i+1})$, it follows that $[b_i]^{+}_{=} = 1$, $\overline{[a_{i+1}]^{+}_{=}} = 1$, both PDN nMOS transistors are on and consequently $F(s_n) = 0 = F^0(s_n)$, as it should. Please note that no electrical conflict between PUN and PDN may exist as *only one* set of serial connected transistors is active at a time even in PUN or in PDN.

IV. Performance evaluation

To provide more inside in the method and to evaluate its potential performance and cost for computer arithmetic building blocks we choose a $7/3$ counter for our simulation comparisons. Since counters are well known examples of multiple-output symmetric functions, $7/3$ counters can be implemented in two stages using hybrid approach from Theorem (1).

In this Section a $7/3$ parallel counter design using HTBL is explained first. Then, in order to evaluate the counter performance when compared with traditional Boolean and Threshold Logic counters, the simulation results and comparisons are presented.

Generally, a $7/3$ counter can be described by the interval description depicted in Figure 3.a and equivalently by the following on-set [7] expressed as Boolean equations:

$$Y_2 = [4]^{+}_{=},\ Y_1 = [2]^{+}_{=} \wedge \overline{[4]^{+}_{=}} \vee [6]^{+}_{=}$$

$$Y_0 = [1]^{+}_{=} \wedge \overline{[2]^{+}_{=}} \vee [3]^{+}_{=} \wedge \overline{[4]^{+}_{=}} \vee [5]^{+}_{=} \wedge \overline{[6]^{+}_{=}} \vee [7]^{+}_{=}. \quad (4)$$

Figure 3.a and, equivalently, Equation (4) suggests that, for example, the Y_0 output has to be logic one in 4 intervals ($s_7 = 1, 3, 5, 7$) and logic zero in 4 intervals ($s_7 = 0, 2, 4, 6$). Consequently the Boolean gate implementing the Y_0 output, has 4 sets of pMOS and 4 sets of nMOS transistors in PUN and PDN respectively. More formally, the PUN can be synthesized assuming Y_0 equation, with negated terms and $\wedge/\vee$ operators meaning serial/parallel connection of pMOS transistors, as in the synthesis theory of static CMOS gates. Similarly, PDN can be synthesized taking into account the off-set descriptions. Figure 4 presents the Boolean stage implementation of the $7/3$ HTBL counter. Using the same way of reasoning, higher order hybrid counters e.g., $15/4$, $31/5$, can be designed.

Fig. 3. Interval descriptions: a) $7/3$ counter, b) interval implementation example

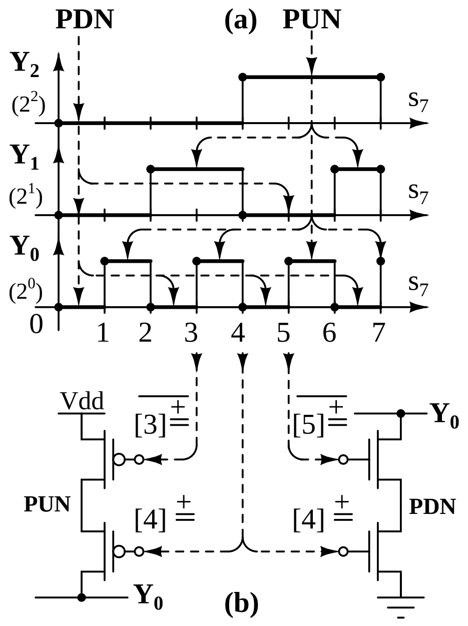
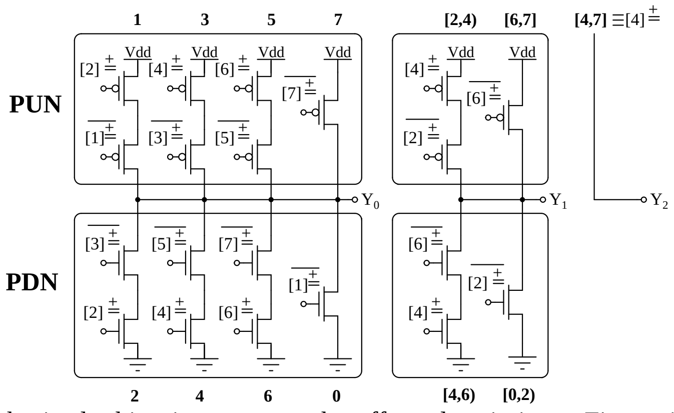


Fig. 4. $7/3$ HTBL counter—Boolean stage design



We designed and compared the HTBL $7/3$ counter, with traditional full-adder based $7/3$ counter from [2] (pp. 129-130), and with TL $7/3$ counter proposed in [4], respectively. We choose the full-adder based scheme for our comparison since it has been proved in literature to be the nearly optimal Boolean approach in terms of logic depth (and therefore speed) for the specific compression ratio of $7/3$ counter ($7 : 3 = 2.33$). A 7-input TL gate, as in Figure 1, was designed and optimized in $0.25\mu m$ feature size CMOS technology. It has $250ps$ worst case delay. Given that a full-adder has a worst case delay of $\Delta_{FA} \simeq 300ps$, the TL gate is with more than 17% faster in this particular technology.

The circuits were simulated with HSpice. Correct operation of TL and HTBL counters was verified by extensive simulations under the process corners. Every simulated counter was loaded with similar counters in order to have a more clear image of the delay improvement in a real partial product reduction matrix environment. We would like to stress out that the delay penalty

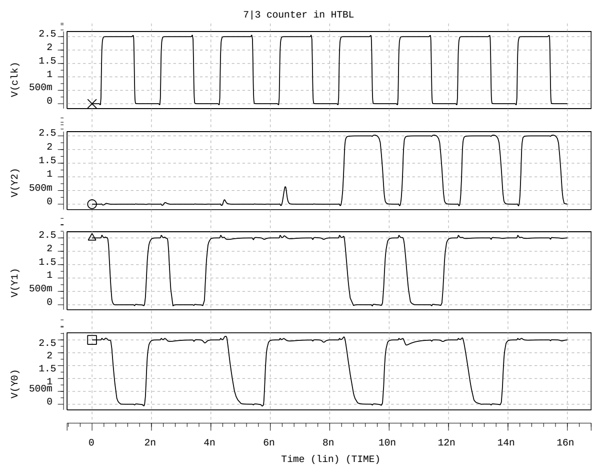


Fig. 5. HSpice waveforms of HTBL 7/3 counter

TABLE I
7/3 COUNTERS DATA ($0.25\mu m$ CMOS, TYPICAL CORNER)

Type	Delay [ps]	Norm. delay	Delay [Δ_{FA}]	# of Tran.
BL	**726**	1.00	2.42	136
TL	**460**	0.63	1.53	155
HTBL	**345**	0.47	1.15	237

caused by capacitive loading in the succeeding stages is minimized with our TL gate since the capacitive load per input is only half the value of a minimum inverter (see Figure 1). The output waveforms for HTBL design are presented in Figure 5.

Table I present the $7/3$ counters characteristics, in terms of absolute delay (@ 27°C, $V_{dd} = 2.5V$), normalized delay, and total number of transistors. For TL and HTBL counters "delay" signifies "latency" (data-output delay) as a clock is needed for their operation.

Table I suggests that $7/3$ HTBL counter, when compared with Boolean, is with 53% faster at the expense of 67% more transistors. When compared with TL $7/3$ counter [4], HTBL is 25% faster at the expense of 52% more transistors. HTBL counters compare favorably in terms of speed with either full-adder based Boolean and Threshold logic counterparts since they take the advantages of both Threshold logic (fast parallel processing of large number of inputs in the first stage) and Boolean logic (high-speed implementation of And-Or like Boolean functions in the second stage).

V. Conclusions

In this paper we proposed a high-speed hybrid Threshold-Boolean logic style suitable for Boolean symmetric functions implementation. First, we presented a depth-2 hybrid implementation scheme for arbitrary symmetric Boolean functions, based on differential Threshold logic gates as circuit style. Finally, we presented hybrid logic design of a $7/3$ counter. The simulation results, suggest that the hybrid $7/3$ counter designed in $0.25\mu m$ CMOS technology, achieves between 25% and 53% higher speed when compared with traditional Threshold logic and Boolean logic counterparts, at expense of between 52% and 67% transistors.

References

[1] S. Coţofană and S. Vassiliadis. Periodic symmetric functions, serial addition and multiplication with neural networks. *IEEE Trans. on Neural Networks*, 9(6):1118–1128, October 1998.

[2] I. Koren. *Computer arithmetic algorithms*. A. K. Peters, 2002.

[3] Y. Leblebici, H. Ozdemir, A. Kepkep, and U. Cilingiroglu. A compact high-speed (31,5) parallel counter circuit based on capacitive Threshold-logic gates. *IEEE Journal of Solid-State Circuits*, 31(8):1177–1183, August 1996.

[4] R. Minnick. Linear-input logic. *IRE Transaction on Electronic Computers*, EC-10:6–16, March 1961.

[5] S. Muroga. *Threshold logic and its applications*. Wiley and Sons Inc., 1971.

[6] M. Padure, S. Cotofana, C. Dan, M. Bodea, and S. Vassiliadis. A new latch-based Threshold logic family. *Proceedings of International Semiconductor Conference, CAS2001*, 2:531–534, October 2001.

[7] S. Vassiliadis and S. Cotofana. 7|2 counters and multiplication with Threshold logic. *Proceedings of* 30^{th} *Asilomar Conference on Signals, Systems and Computers*, pages 192–196, November 1996.

[8] S. Vassiliadis, S. Cotofana, and K. Bertels. 2-1 addition and related operations with Threshold logic. *IEEE Transactions on Computers*, 45(9):1062–1068, September 1996.