νMOS Enhanced Differential Current-Switch Threshold Logic Gates

K.C. Li(*), M. Padure(**), S.D. Cotofana(*)
(*) Delft University of Technology, The Netherlands
Mekelweg 4, 2628 CD, Delft, The Netherlands
email:{kwok, sorin}@ce.et.tudelft.nl
(**) Politehnica University of Bucharest,
Laboratory of Microelectronic Systems,
Blvd. Iuliu-Maniu 1-3, Bucharest, Romania
email:marius@messnet.pub.ro

Abstract— **This paper presents a way to enhance the Differential Current-Switch Threshold Logic gate (DCSTL) in order to allow the gate to perform more complex functions. This enhancement is achieved by replacing the MOS-transistors at the data- and threshold mapping bank of the DCSTL gate with neuronMOS transisors. First, we introduce the neuronMOS-enhanced DCSTL gate. Then, the results of HSPICE simulations of a 7-input parity and a 3-bit addition function implemented with the enhanced DCSTL gate is presented, along with a comparison with the same functions implemented using the original DCSTL gate. These simulations indicate that the designs based on enhanced DCSTL gates can achieve a 12.5% speed-up over the conventional DCSTL gate designs for both addition and parity. HSPICE power estimations also suggest that the standard DCSTL gate dissipates 30% more power when performing a 7-bit parity, and 5% in the case of a 3-bit addition.**

Keywords—**Threshold logic, neuron-MOS transistors, parity, addition**

I. Introduction

Threshold Logic (TL) constitutes an alterative way for implementing logic and arithmetic functions. The main theoretical advantage of TL [5][9] stands in its powerful computational capabilities when compared with standard Boolean gates. Therefore, TL based implementations require less gates and shallower networks when compared with traditional Boolean gate networks for a broad range of functions [5][9]. However, while TL gates are more powerful, their theoretical advantage comes at the expense of more basic circuit elements (e.g., MOS transistors). Therefore, in practice, depending on the specific TL gate circuit and manufacturing technology the theoretical advantage may be lost. Up to date, several TL implementations have been proposed in either CMOS [1][2] and emerging technologies, e.g., SET [6][7], and it was demonstrated that for some functions (mainly population counters - basic building blocks for multipliers) they outperform Boolean gate implementations [1][6][7][8].

Even though these studies suggests that TL may provide effective practical circuits, state of the art CMOS based TL gates are rather expensive in terms of area and one should seek ways to further reduce their area by proposing less complex circuits. In this paper we address this issue however form a different prospective. Instead of attempting to reduce the number of circuit elements in a TL gate implementation we propose a way to increase the gate computational capabilities. Our proposal is based on the observation that the up to date TL gate implementations in CMOS are differential in nature, and usually require comparators to perform a part of the computations. For TL gates with small fan-in and weights values the comparator may dominate the gate area and delay. While this overhead cannot be reduced as such, as the comparator is a crucial component in the TLG, one can embed more computations in the gate such that the overhead induced by the comparator, relative to the achieved computation power, is reduced.

In this paper we propose to increase in computational capability of a TL gate by combining two existing TL implementations: the Differential Current-Switch Threshold Logic gate (DCSTL) [1][3], and the neuronMOS transistor (νMOS) [2][10][11]. The basic idea is to use νMOS transistors to replace the standard NMOS transistors in the data- and threshold mapping bank of the DCSTL gate. In this way we obtain a νMOS enhanced DCSTL gate which can evaluate functions that are more complex than simple threshold functions. One single νMOS enhanced DCSTL gate can actually evaluate functions which would otherwise require depth-2 implementations built with more than one TL gate. Thus, simply speaking, more complex functions can be evaluated with just one comparator.

In order to evaluate the modified DCSTL gate, we assumed 7-input parity and 3-bit addition and implemented them both with standard DCSTL gates and with enhanced DCSTL gates. For comparison purposes we simulated all the designs with HSPICE. Our simulations indicate that the designs based on enhanced DCSTL gates can achieve a 12.5% speed-up over the conventional DCSTL gate designs for both addition and parity. HSPICE power estimations also suggest that the standard DCSTL gate dissipates 30% more power when performing a 7-bit parity, and 5% in case of a 3-bit addition.

This paper is organized as follows: Section II provides some Threshold logic background and briefly introduces the DCSTL gate and the νMOS transistor. Then, the νMOS-enhanced DCSTL gate is introduced in Section III. As examples, implementations of a 7-input parity and a 3-bit adder are presented in Section IV, together with results of HSPICE simulations and a comparison of the designs based on νMOS-enhanced DCSTL gates with the designs using standard DCSTL gate. Section V closes this paper with some concluding remarks.

II. BACKGROUND

A Threshold Logic Gate (TLG) is a device that is able to compute any linearly separable Boolean function given by:

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

where $\mathcal{F}(X) = \sum_{i=1}^{n} \omega_i x_i - \psi$, x_i are the n Boolean inputs and w_i are the corresponding n integer weights. The TLG performs a comparison between the weighted sum of the inputs $\Sigma_{i=1}^{n} \omega_i x_i$ and the threshold value ψ. If the weighted sum of inputs is *greater than or equal to* the threshold, the gate produces a logic '1'. Otherwise, the output is a logic '0'. Up to date, many gate implementations are able to evaluate functions lik in Equation (1) are available, but for the purpose of the current paper, two of them are relevant: the DCSTL gate, and the νMOS transistor.

The DCSTL gate is depicted in Figure 1. It features a comparator, and two sets of parallel connected NMOS transistors. The two sets of NMOS transistors are called the data bank, and the threshold mapping bank. These banks draw a certain amount of current, and the comparator compares the two currents. Each input is connected to a the gate of a transistor in the data mapping bank. The weights are implemented by dimensioning the transistor such, that, for example, a transistor that implements a weight of 2 should draw twice as much current as a transistor that implements a weight of 1. The total current drawn by the data mapping bank is

$$I_{data} = \sum_{i=1}^{n} I_i, \qquad (2)$$

where n is the number of inputs attached to the data mapping bank, and I_i is the on current controlled by the input x_i. The threshold is programmed by hardwiring the gates of the transistors in the threshold mapping bank either to the ground, or V_{dd}. The ouput of the comparator is high if the current drawn by the data mapping bank exceeds the current drawn by the threshold mapping bank, implementing a TL function as described in Equation (1).

The comparator itself works as follows: The comparator is in its precharge phase when the clock is low. Transistors M_{10}, M_{11} are on, and M_5, M_8 and M_9 are switched off, pulling nodes X and Y up. When the clock switches to high, the comparator enters its evaluation phase. M_{10} , M_{11} closes, while M_5, M_8 and M_9 switches on. Note that M_1, M_2, M_3 and M_4 now act as a pair of cross-coupled inverters. Because nodes X and Y were precharged high, inverter M_1, M_3 will try to pull down node Y, while inverter M_2, M_4 will try to pull down node X. At the beginning of evaluation phase, M_6 and M_7 are also on, due to node X and Y being precharged high in the precharge phase. This causes the data mapping bank to draw current from node X, while the threshold mapping bank draws current from node Y. The voltage at node X drops faster than the voltage at node Y, if the current I_{data} is stronger than I_T, causing Y to go up. In the reverse situation, if $I_T > I_{data}$, the exact opposite happens, and Y goes down, pulling X back up. The Nand gates X1 and X2 make up a latch, and are in place to hold the results of the current evaluation during the next precharge phase. We note here, that this gate is also capable of implementing TL functions with negative weights and/or a negative threshold. Inputs with negative weights are simply connected to transistors in the threshold mapping bank, and a negative threshold is programmed in the data mapping bank instead of in the threshold mapping bank.

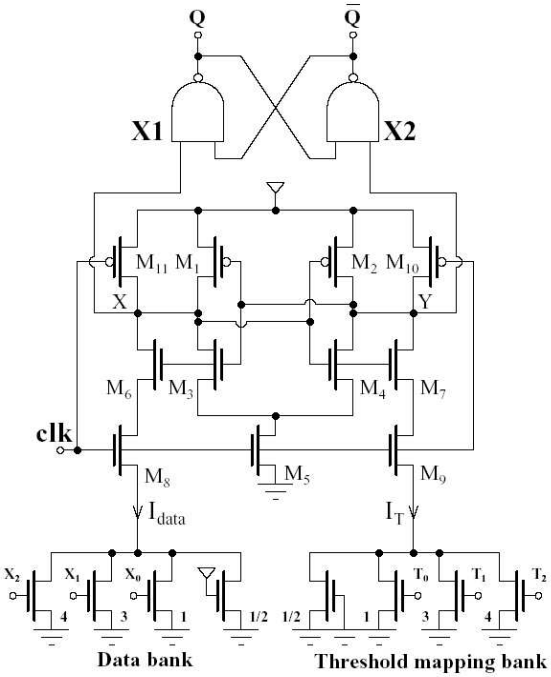

Fig. 1. DCSTLgate

A νMOS transistor [3], is basically a standard MOS transistor with an electrically floating gate. A number of inputs are coupled to the floating gate through capacitors. A schematic representation of the νMOS transistor with n inputs and input capacitors is illustrated in Figure 2. The transistor switches on when the voltage of the floating gate exceeds the transistor threshold voltage, and switches off otherwise. The voltage on the floating gate V_f, can be calculated as

$$V_f = \frac{\sum_{i=1}^{n} C_i V_i}{\sum_{i=1}^{n} C_i + C_0}, \qquad (3)$$

where C_0 stands for the capacitance between the floating gate and the substrate, V_i stands for the voltage of input i, and C_i for the capacitors associated with input i. Note that V_{th} is proportional to the sum of inputs, weighed by the capacitors associated with each input, thus the structure in Figure 2 is able to evaluate threshold functions.

III. NEURONMOS ENHANCED DCSTL GATE

We propose to enhance the DCSTL gate by replacing the standard NMOS transistors in its data- and threshold mapping bank with νMOS transistors. Due to this change, the current I_{data} and I_T, can not be any longer evaluated with an expression similar to Equation (2). Unlike the standard transistor, the νMOS accomodates more than one input, and the current drawn by a

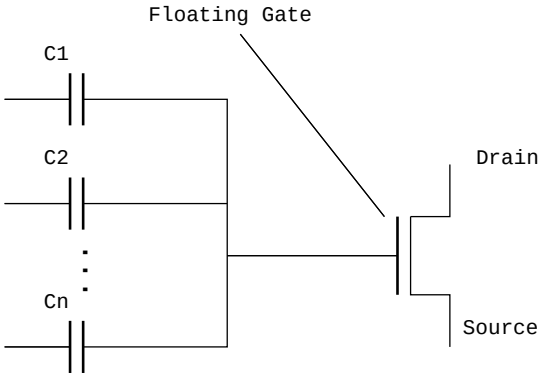

Fig. 2. neuronMOS transistor

νMOS is no longer limited to two values. Let us assume that the MOS transistor in the data mapping bank corresponding to input x_k is replaced by a νMOS transistor with n inputs X_k = [x_k^n,..., x_k^1, x_k^0], and that V_f^k and V_{th}^k are the floating gate voltage, and threshold voltage of the νMOS, respectively. The νMOS transistor draws no current if $V_f^k < V_{th}^k$. When $V_f > V_{th}$, the νMOS transistor will draw a current that is proportional to $V_f^k - V_{th}^k$. Recalling Equation (3), the current draw by the νMOS can actually be expressed as

$$I_k = \begin{cases} 0 & \text{for } V_f^k < V_{th}^k \\ \frac{\sum_{i=1}^{n} C_i x_k^i}{\sum_{i=1}^{n} C_k^i + C_k^0} - V_{th}^k & \text{otherwise} \end{cases} \quad (4)$$

In other words, the νMOS transistor is capable of computing the following function:

$$F^k(X_k, T_k) = \begin{cases} (X_k - T_k) & \text{for } X_k \geq T_k, \\ 0 & \text{for } X_k < T_k \end{cases} \quad (5)$$

where X_k is the sum of the weighted inputs of the νMOS, and T_k the threshold of the νMOS. Given this, the enhanced DCSTL gate is capable of evaluating the following function:

$$F = sgn\{\sum_{i=1}^{2k} w_k F^k(X_k, T_k)\} \quad (6)$$

w_k is a factor that corresponds to the dimensions of the transistor at position k. This value is negative, if the νMOS is connected to the threshold mapping bank instead of the data mapping bank. The enhanced DCSTL gate still behaves as a TL gate, as it evaluates an sgn function. However, the functionality has been enhanced by the νMOS transistors, which evaluate the weighted sum of its own inputs. For convenience, in the remainder of this paper, a νMOS transistor that performs the following function

$$F(X, T) = \begin{cases} (X - T) & \text{for } X \geq T \\ 0 & \text{for } X < T. \end{cases} \quad (7)$$

will be referred to as $(T)_{=}^{+}$.

Although the threshold value of the standard DCSTL gate can be programmed easily, the νMOS transistor as depicted in Figure 2 does not share this property. The threshold value of the νMOS transistor depends on its threshold voltage V_{th}, which in turn, depends on physical properties such as the type of gate metal, oxide thickness, and sillicon doping levels. Thus, changing V_{th} in order to alter the threshold value in Equation (5), will involve changing the named physical properties. This approach is not very attractive, as this doesn't allow the programming of the threshold values in a practical way. In [4], a scheme for a νMOS with threshold setting via inputs commutation is discussed. The basic idea is to use the input capacitors during its precharge phase to set the threshold value. This method was used in [4] for the implementation of a νMOS based threshold gate, and produces only two levels at the output: either a high, or a low voltage, instead of performing the function described in Equation (5) that we actually need in our case. Therefore, we slightly modified the scheme in order to obtain the required behaviour. An example schematic of our implementation of the νMOS transistor is depicted in Figure 3.

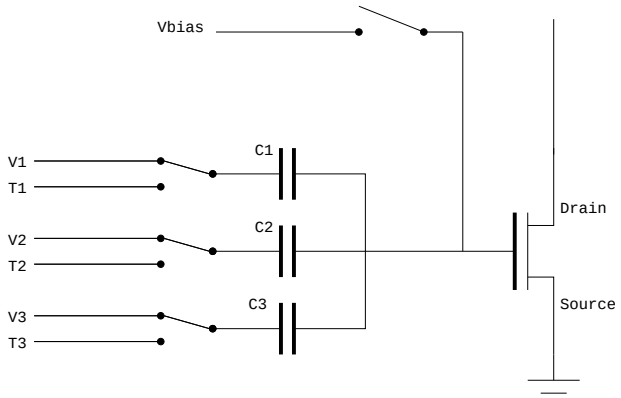

Fig. 3. neuronMOS transistor with programmable threshold

The structure in Figure 3 assumes 3 data inputs V1, V2 and V3. However, during the precharge phase, the switches connect the input capacitors to T1, T2 and T3 instead, and the floating gate to V_{bias}. T1, T2, and T3 are connected either to ground or VDD, and are used to set the threshold value we want to program for the νMOS transistor. In the evaluation phase, the floating gate is left floating, and the input capacitors are connected to normal data inputs V1, V2 and V3. Assuming the general case when the νMOS transistors has n inputs, the floating gate voltage at the end of the evaluation phase can be calculated as:

$$V_f = \frac{\sum_{i=1}^{n}(C_i V_i) - \sum_{i=1}^{n}(C_i T_i)}{\sum_{i=1}^{n} C_i + C_0} + V_{bias}. \quad (8)$$

If we take $V_{bias} = V_{th}$, the transistor in Figure 3 will now switch on when

$$\frac{\sum_{i=1}^{n}(C_i V_i) - \sum_{i=1}^{n}(C_i T_i)}{\sum_{i=1}^{n} C_i + C_0} + V_{bias} < V_{th} \quad (9)$$

or

$$\frac{\sum_{i=1}^{n}(C_i V_i) - \sum_{i=1}^{n}(C_i T_i)}{\sum_{i=1}^{n} C_i + C_0} < 0 \quad (10)$$

or

$$\sum_{i=1}^{n}(C_i V_i) - \sum_{i=1}^{n}(C_i T_i) < 0 \quad (11)$$

If we assume

$$T = \sum_{i=1}^{n} (C_i T_i), \tag{12}$$

then the circuit depicted in Figure 3 implements Equation (5).

Using this scheme, we can now use νMOS transistors, each programmed with a different threshold T, to augment the DCSTL gate in order to implement more complex functions.

For example, we can implement the following function.

$$(T-1)^{+}_{=} - 2(T)^{+}_{=} + (T+1)^{+}_{=} \tag{13}$$

To implement this function wit a certain value T, we have to attach one νMOS with threshold T-1, and one νMOS with threshold T+1 to the data mapping bank, and a νMOS transistors with multiplicity 2, and a threshold T to the threshold mapping bank of the DCSTL gate. A transistor connected to the data- or threshold mapping bank with multiplicity m, means that we dimension the transistor such, that the transistor draws m times the current of what it normally would draw. Figure 4a shows the function $(T-1)^{+}_{=}$, Figure 4b adds $-2(T)^{+}_{=}$, to the previous, and $(T+1)^{+}_{=}$ is added in Figure 4c.

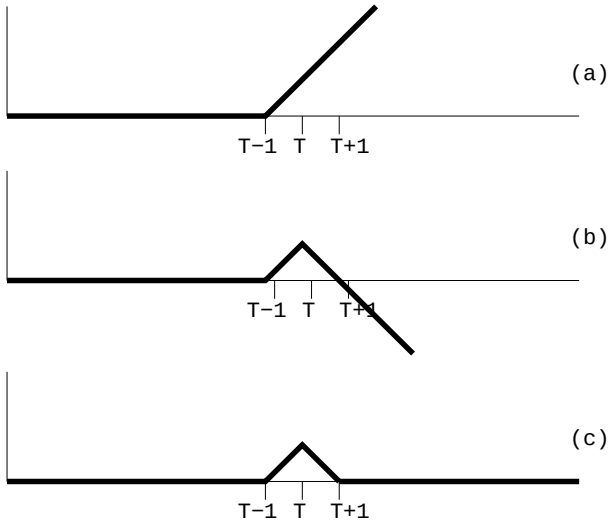

Fig. 4. $(T-1)^{+}_{=} - 2(T)^{+}_{=} + (T+1)^{+}_{=}$

Note that Equation (13) results in an output 1 at X=T, and 0 for any other input X. Using Equation (13), we can implement functions with 1's and 0's we desire. Consequently, in theory, using νMOS transistors instead of conventional ones at the data- and threshold mapping banks, any symmetric Boolean function can be implemented using only one (enhanced) DCSTL gate.

Unfortunately, a νMOS cannot accomodate too large a number of inputs. This is due to the fact, that Equation (5) is merely an approximation of the behaviour of the νMOS transistor. Figure 5 shows the behaviour of a short-channel νMOS transistor, together with the desired behaviour. The solid line is the desired behaviour, while the dashed line represents the actual current drawn by the transistor. The difference between the actual behaviour of the νMOS and the desired behaviour, is in the nonlinear region near the threshold of the gate. This difference becomes more significant as the number of inputs increase, and this prevents us from using too many inputs. Figure 6 a and b shows the situation of a νMOS transistor with few inputs, and a νMOS transistor with a large number of inputs. One can observe that in the case in Figure 6b, more values for input X fall within the nonlinear region. This prevents a proper implementation of functions such as Equation (13).

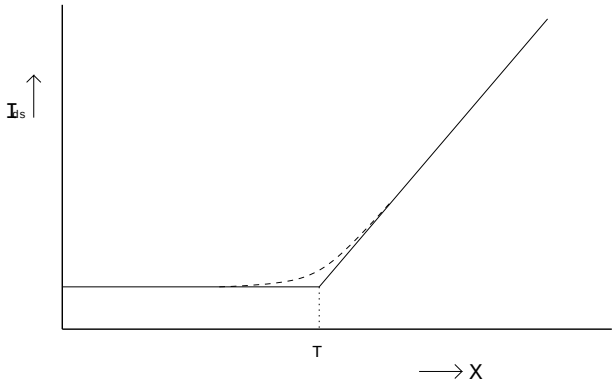

Fig. 5. Desired and actual behaviour of the νMOS

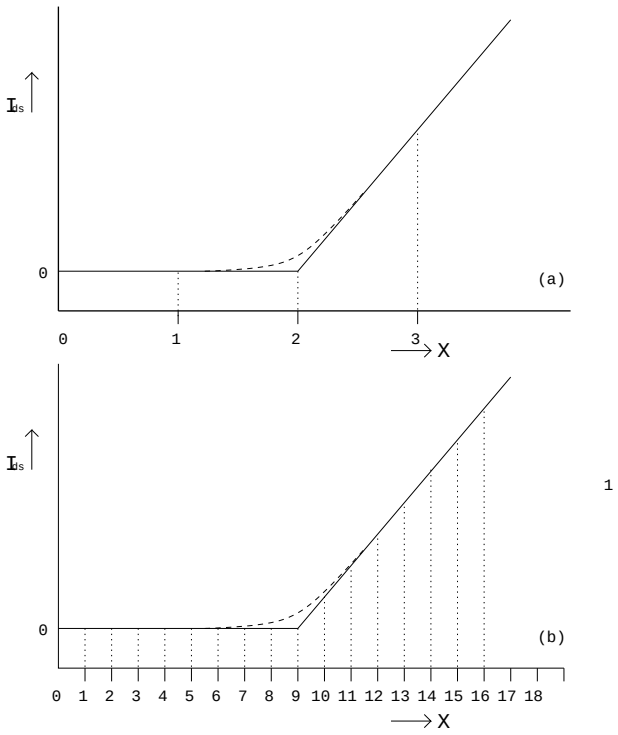

Fig. 6. Impact of a large fan-in on νMOS behaviour.

IV. Experimental results

To evaluate our proposal, a 7-input parity and a 3-bit binary adder are implemented in both standard, and enhanced DCSTL, and are compared in terms of delay and area. HSPICE was used to verify correct operation and evaluate the delays.

The 7-input parity function is '1' for X=1, X=3, X=5 and X=7, and '0' otherwise. Using Equation (13), the 7-input parity

can be implemented with the enhanced DCSTL gate as follows:

$$\begin{aligned} XOR7 = \quad & sgn\{(0)^{+}_{=} - 2(1)^{+}_{=} + (2)^{+}_{=} \\ & +(2)^{+}_{=} - 2(3)^{+}_{=} + (4)^{+}_{=} \\ & +(4)^{+}_{=} - 2(5)^{+}_{=} + (6)^{+}_{=} \\ & +(6)^{+}_{=} - 2(7)^{+}_{=} + (8)^{+}_{=}\} \end{aligned} \tag{14}$$

As the input X , where $X = \sum_{i=1}^{7} x_i$, is limited to 7, the terms -2(7)$^{+}_{=}$ and +(8)$^{+}_{=}$ are unnecesary. The observation leads to the simplification of the inplementation as follows:

$$\begin{aligned} XOR7 = \quad & sgn\{(0)^{+}_{=} + 2(2)^{+}_{=} + (4)^{+}_{=} \\ & +2(6)^{+}_{=} - 2(1)^{+}_{=} - 2(3)^{+}_{=} \\ & -2(5)^{+}_{=} - \tfrac{1}{2}\} \end{aligned} \tag{15}$$

The extra term -$\frac{1}{2}$ is added for two reasons: $sgn\{0\}$ is usually defined as 1. Furthermore, in case of the DCSTL gate, it also means that, in the case of evaluating $XOR7(0)$, the data bank, and threshold mapping bank both draw an equal amount of current, which may result in an unstable output of the DCSTL gate.

The 3-bit adder is implemented in a similar fashion, except that the inputs are now two 3-bit numbers, and a carry-in, and the inputs are weighted according to the significance of each bit:

$$X = \sum_{i=0}^{2} s^i(a_i + b_i) + Cin \tag{16}$$

The output of the 3-bit adder is a 3-bit sum $s[2-0]$ and a carry-out. We need one enhanced DCSTL gate for each output bit. The Carry-out can be generated without the need for an enhanced DCSTL gate, a standard one will suffice:

$$Carry - out = sgn\{X - 8\} \tag{17}$$

The highest bit $s[2]$ can be expressed as:

$$\begin{aligned} s[2] = sgn\{ & (3)^{+}_{=} - (4)^{+}_{=} - (7)^{+}_{=} + (8)^{+}_{=} \\ & +(11)^{+}_{=} - (12)^{+}_{=} - (15)^{+}_{=} - \frac{1}{2}\} \end{aligned} \tag{18}$$

$s[1]$ can be generated using the function:

$$\begin{aligned} s[1] = sgn\{ & (1)^{+}_{=} - (2)^{+}_{=} - (3)^{+}_{=} + (4)^{+}_{=} \\ & +(5)^{+}_{=} - (6)^{+}_{=} - (7)^{+}_{=} - \frac{1}{2}\} \end{aligned} \tag{19}$$

Note that the two highest bits are not needed to generate $s[1]$. These two bits are not connected to the gate that generates $s[1]$.

The least significant bit is the result of a 3-bit parity of the two lowest bits and the carry-in, and is implemented in much the same way as the 7-input parity above.

$$s[0] = sgn\{(0)^{+}_{=} - 2(1)^{+}_{=} + (2)^{+}_{=} - \frac{1}{2}\} \tag{20}$$

In order to evaluate our results, both the 7-bit parity, and 3-bit adder were also implemented using the standard DCSTL gate. The 7-input parity, and the 3-bit addition can be implemented in 2 levels of threshold logic with standard DCSTL. This is achieved either by using Minnick[5], or Muroga[5] schemes. Considering that the performance of both schemes are comparable in terms of delay, a Minnick network was chosen as it uses a smaller number of threshold gates when compared to Muroga. Equation (21), implements the 7-bit parity:

$$\begin{aligned} XOR7 = sgn\{ \quad & X - 2sgn\{X - 6\} \\ & -2sgn\{X - 4\} \\ & -2sgn\{X - 2\} \\ & -1\} \end{aligned} \tag{21}$$

Equation (21) also shows, that we need 4 standard DCSTL gates to implement the 7-input parity, compared to only 1, when implemented using the enhanced DCSTL gate. Equations (22), (23), (24), and (25) generate carry-out, and the sum bits.

$$carry - out = sgn\{X - 8\} \tag{22}$$

$$s[2] = sgn\{X - 8sgn\{X - 8\} - 4\} \tag{23}$$

$$s[1] = sgn\{X - 4sgn\{X - 4\} - 2\} \tag{24}$$

$$s[0] = sgn\{X - 2sgn\{X - 2\} - 1\} \tag{25}$$

According to Equations(22), (23), (24), and (25) indicate that 7 standard DCSTL gates are needed, while we only need 4 when using the enhanced DCSTL gate.

	precharge	evaluation	total delay	avg power
7-bit parity, eDCSTL	3ns	5ns	8ns	291mW
7-bit parity, sDCSTL	3ns	3ns	9ns	377mW
3-bit adder, eDCSTL	3ns	5ns	8ns	557mW
3-bit adder, sDCSTL	3ns	3ns	9ns	586mW

TABLE I
RESULTS OF HSPICE SIMULATIONS

Table I shows the time needed during the precharge phase and evaluation phase of the DCSTL gate, the total delay and estimated average power dissipation. Although we need 2 levels of threshold logic in standard DCSTL, it is possible to overlap the precharge phase of the second level with the evaluation phase of first level. The delay would be only $1\frac{1}{2}$ clock cycles instead of 2.

V. CONCLUSIONS

Aa way to enhance the Differential Current-Switch Threshold Logic gate (DCSTL) was presented in order to allow the gate to perform more complex functions. This enhancement was achieved by replacing the MOS-transistors at the data- and threshold mapping bank of the DCSTL gate with neuronMOS transisors. First, we introduced the neuronMOS-enhanced DCSTL gate. Then, the results of HSPICE simulations of a 7-input parity and a 3-bit addition function implemented with the enhanced DCSTL gate was presented, along with a comparison with the same functions implemented using the original DCSTL gate. These simulations indicated that the designs based on enhanced DCSTL gates can achieve a 12.5% speed-up over the conventional DCSTL gate designs for both addition and parity. HSPICE power estimations also suggested that the standard DCSTL gate dissipates 30% more power when performing a 7-bit parity, and 5% in the case of a 3-bit addition.

REFERENCES

[1] M. Padure, S. Cotofana, S. Vassiliadis, "A CMOS flip-flop featuring embedded Threshold logic functions", 13th ProRISC workshop on Circuits, Systems and Signal Processing, ProRISC2002, Veldhoven, The Netherlands, 2002

[2] T. Shibata, T. Ohmi, "A functional MOS Transistor Featuring Gate-Level Weighted Sum and Threshold Operations", IEEE Transactions on electron devices, vol. 39, no.6, june 1992

[3] D. Somasekhar, K. Roy, "Differential Current Switch Logic: A low Power DCVS Logic Family", IEEE journal of solid-state circuits, vol 31, No. 7, July 1996

[4] R. Lashevsky, K. Takaara, M. Souma, "The efficiency of neuron-MOS transistors in threshold Logic", Soft Computing 3, Springer-Verlag, 1999

[5] S. Muroga, "Threshold logic and its applications", New York, Wiley-Interscience, 1971

[6] C. R. Lageweg, S. D. Cotofana, S. Vassiliadis, "Binary addition based on single electron tunneling devices", Proceedings of the 2004 Fourth IEEE Conference on Nanotechnology, pp. (CD proceedings), Munich, Germany, August 2004

[7] C. R. Lageweg, S. D. Cotofana, S. Vassiliadis, "A linear threshold gate implementation in single electron technology" , Proceedings. IEEE Computer Society Workshop on VLSI 2001: Emerging Technologies for VLSI Systems, pp. 93-98, Orlando, USA, April 2001

[8] M. D. Padure, S. D. Cotofana, S. Vassiliadis, "High-speed hybrid threshold-Boolean logic counters", 45th International Midwest Symposium on Circuits and Systems, pp. 457-460, Tusla, Oklahoma, USA, August 2002

[9] S. D. Cotofana, "Addition Related Arithmetic Operations with Threshold Logic", PhD Thesis, Delft, January 1998, PhD Thesis

[10] T. Shibata, T. Ohmi, "Neuron MOS binary-logic integrated circuits. I. Design fundamentals and soft-hardware-logic circuit implementation", Electron Devices, IEEE Transactions on Volume 40, Issue 3, March 1993

[11] T. Shibata, T. Ohmi, "Neuron MOS binary-logic integrated circuits. II. Simplifying techniques of circuit configuration and their practical applications", Electron Devices, IEEE Transactions on Volume 40, Issue 5, May 1993

[12] H. Ozdemir, A. Kepkep, B. Pamir, Y. Leblebici, U. Cilingiroglu, "A capacitive threshold-logic gate", Solid-State Circuits, IEEE Journal of Volume 31, Issue 8, Aug. 1996