A COMPACT PARALLEL MULTIPLICATION SCHEME BASED ON (7,3) AND (15,4) SELF-TIMED THRESHOLD LOGIC COUNTERS

Peter Celinski, Troy Townsend, Said Al-Sarawi, Derek Abbott

Centre for High Performance Integrated Technologies & Systems (CHiPTec), The Department of Electrical and Electronic Engineering, Adelaide University, SA 5005, Australia.

José F. López

Research Institute for Applied Microelectronics, Universidad de Las Palmas de G.C., 35017-Spain.

ABSTRACT

This paper presents a new, a highly compact implementation of a 32×32 parallel multiplier based on parallel counters. The new multiplier is designed using the recently proposed Self-Timed Threshold Logic (STTL). The design is based on a direct multiplication scheme using depth 2 (15,4) and (7,3) STTL parallel counters and (4:2) compressors. The proposed parallel multiplier reduces the partial product matrix to two rows in only three stages, hence the effective multiplier logic depth is 6. It is shown that the presented scheme significantly reduces the gate count of known proposals for multiplication using threshold logic.

1. INTRODUCTION

As the demand for higher performance very large scale integration processors with increased sophistication grows, continuing research is focused on improving the performance, area efficiency and functionality of the arithmetic and other units contained therein. Low power dissipation has become a major issue demanded by portable and embedded applications and also the high performance processor market in order to meet the high density requirements of advanced VLSI processors. In addition, applications such as DSP and on-chip multipliers for processors have created the demand for fast, area efficient multipliers. The main goal of this work was to develop an area efficient multiplier architecture while maintaining the high performance of known threshold logic based designs.

Threshold logic (TL) was introduced over four decades ago, and over the years has promised much in terms of reduced logic depth and gate count compared to conventional logic-gate based design. However, lack of efficient realizations has meant that TL has, until recently, had little impact on VLSI systems. Efficient TL gate realizations have recently become available [1] [2] [3] [4], and a number of applications based on TL gates have demonstrated its ability to achieve high operating speed and significantly reduced area compared to conventional logic [5].

This paper presents a highly compact implementation of a parallel multiplier based on Self-Timed Threshold Logic counters. The organisation of the paper is as follows. Section 2 gives a brief overview of threshold logic, followed by a description of Self-Timed Threshold Logic in Section 3. Section 4 describes the (7,3) and (15,4) STTL counters, the proposed partial product matrix reduction scheme for a 32×32 multiplier and a comparison with known proposals for multiplication using TL. Finally, the results of this work are summarized in Section 5.

2. THRESHOLD LOGIC

A threshold logic gate is functionally similar to a hard limiting neuron. The gate takes n binary inputs x_1,x_2,…,x_n and produces a single binary output y, as shown in Fig. 1. A linear weighted sum of the binary inputs is computed followed by a thresholding operation.

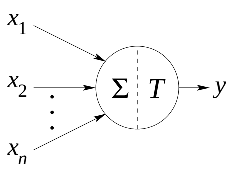

Fig. 1. Threshold Gate Model

The Boolean function computed by such a gate is called a threshold function and it is specified by the gate threshold T and the weights w_1,w_2,…,w_n, where w_i is the weight corresponding to the ith input variable x_i. The output y is given by

$$y = \begin{cases} 1, & \text{if } \sum_{i=1}^{n} w_i x_i \geq T \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

This function can be written in a more compact form using the sgn notation as

$$y = \operatorname{sgn}\left(\sum_{i=1}^{n} w_i x_i - T\right). \quad (2)$$

The sgn function is defined as $\operatorname{sgn}(x) = 1$ if $x \geq 0$ and $\operatorname{sgn}(x) = 0$ if $x < 0$. Alternatively, expressions of the type $\operatorname{sgn}(x - T)$ may be conveniently (and informally) written simply as T^+, where it is understood that the actual sgn function argument is $x - T$. This will allow us to easily describe feed-forward TL networks with composite expressions such as $y = \operatorname{sgn}(x - 2 - 3 \cdot 5^+)$.

A TL gate can be programmed to realize many distinct Boolean functions by adjusting the gate threshold T. For example, an n-input TL gate with $T = n$ will realize an n-input AND gate, while setting $T = n/2$ results in a majority function. This versatility means that TL offers a significantly increased computational capability over conventional logic. Significantly reduced area and increased circuit speed can therefore be obtained, especially in applications requiring a large number of input variables.

3. SELF-TIMED THRESHOLD LOGIC (STTL)

Both static and dynamic synchronous TL gate implementations have been devised. Purely static gates such as neuron-MOS suffer from limited fan-in [5], typically less than 12 inputs. Also, some of the existing dynamic gates have the disadvantages of relatively high short circuit and dynamic power dissipation, and some require multiple non-overlapping clock phases [2] [5].

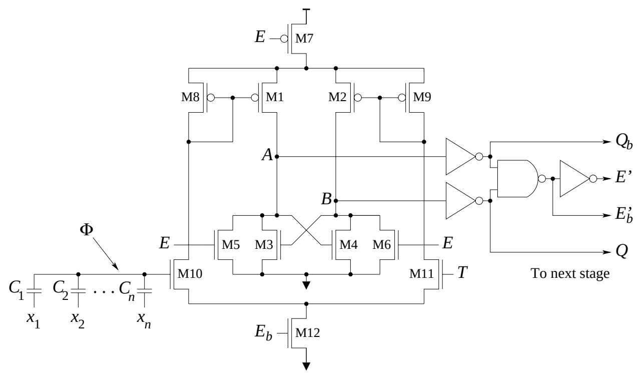

Fig. 2. The proposed Self-Timed Threshold Logic gate structure

Fig. 2 shows the proposed dynamic circuit structure for implementing a Self-Timed Threshold Logic gate. The main element is the cross coupled NMOS transistor pair (M3, M4) which generates the output Q and its complement Q_b after buffering by the two inverters. The gate operates in two phases. Precharge and evaluate are specified by the dual enable signals E and its complement E_b. The inputs x_i are capacitively coupled onto the floating gate ϕ of M10, and the threshold is set by the gate voltage T of M11. The potential ϕ is given by $\phi = \sum_{i=1}^{n} C_i x_i / C_{tot}$, where C_{tot} is the sum of all capacitances, including parasitics, at the floating node. Weight values are thus realised by setting capacitors C_i to appropriate values. Typically, these capacitors are implemented between the polysilicon 1 and polysilicon 2 layers, although alternatives, such as trench capacitors used in DRAM or MIM capacitors, are available in some processes.

The enable signals, E and E_b, control the precharge and activation of the sense circuit. When E is high the voltages at nodes A and B are discharged to ground. When E is low and E_b is high, the outputs are disconnected from ground and the differential circuit, formed by M10 and M11, draws different currents from the supply via M8 and M9. The currents in M8 and M9 are mirrored by M1 and M2 respectively, and the gates of M3 and M4 (nodes A and B) begin to charge at different rates. As the charging rates are different and the capacitances at those two nodes are the same (ensured by identical sizing of the two buffer inverters), a voltage difference begins to develop between nodes A and B. When this difference is sufficiently large, either M3 or M4 turns on, but not both. The outputs Q and Q_b are evaluated and passed to the next stage. In this way, the circuit structure effectively determines if the weighted sum of the inputs, ϕ, is greater or less than the threshold, T, thus realizing a thresholding operation. The two buffer inverters serve to provide a balanced capacitive load for nodes A and B and also to drive the inputs of the next stage.

The next STTL gate is held in precharge until the previous gate evaluates its output. The enable signals for the next stage are generated by the NAND gate output E'_b and its inverse E'. During the precharge phase of the first stage, the Enable signals for the next stage are E'_b=0 and E'=1, hence the second stage and all subsequent stages are also in the precharge phase, and only begins to evaluate after the outputs of the first stage (Q and Q_b) are established. Correct timing is ensured by setting the combined delay of the two buffer inverters and the NAND gate to be larger than the evaluation delay of the first gate. Thus, outputs of each gate propagate through the chain in a self-timed fashion.

4. THE PROPOSED 32×32 MULTIPLIER DESIGN

Parallel counters, or simply counters, are multiple-input circuits that count the number of inputs in a given state (normally logic 1). The most common application of counters is the reduction of the partial product matrix (PPM) in parallel multipliers. The bits forming the columns of the matrix are the inputs to a succession of counters which reduce the matrix to two rows. These two words are then added using a fast carry-propagate adder to produce the final result of the multiplication. The majority of the delay and area of multipliers is associated with the reduction of the PPM.

4.1. (7,3) and (15,4) Parallel Counters Using STTL

An (m,n) counter is a combinatorial network which generates a binary coded output vector of length n which corresponds to the number of logic 1's in the m-bit input vector. In conventional logic, counters such as (7,3) or (15,4) have traditionally been implemented by using trees of (3,2) counters (full adders) because of the disadvantages of a direct implementation [6]. However, counters consisting of such full adder trees have a relatively high delay and grow rapidly with input vector size in terms of the required number of full adders [7].

The truth table for the (7,3) counter, and the (7,3) Minnick counter [8] design are shown in Fig. 3. The Minnick implementation was chosen because it offers a good gate count-delay tradeoff. The input v consists of the seven input bit lines, each having a weight of 1, and is denoted by a thick black line to differentiate it from the single bit lines. In effect v represents the arithmetic sum of 1's in the 7 inputs. From the truth table, the MSB of the output, y_2, is 1 when $v \geq 4$, hence y_2 is the output of the first layer gate which has a threshold of 4. The y_1 output is 1 when $2 \leq v < 4$ and $v \geq 6$. Therefore the second layer gate which has threshold 2 computes y_1. This gate has an input weighted -4 from the first layer gate which has threshold 4. Similar reasoning can be applied to the output y_0. In the general case, the MSB will be computed by a first layer gate, and the lesser significance outputs are computed in the second layer. The second layer gates have as inputs, in addition to v, the negatively weighted outputs from the first layer to isolate the desired ranges of v where those outputs are 1.

The operation of the (7,3) Minnick counter can be described by the following expressions [9]:

$$\begin{aligned} y_2 &= \operatorname{sgn}(v-4) \;=\; 4^+ \\ y_1 &= \operatorname{sgn}(v-2-4\cdot 4^+) \\ y_0 &= \operatorname{sgn}(v-1-2\cdot 2^+ - 2\cdot 4^+ - 2\cdot 6^+). \qquad (3) \end{aligned}$$

The (15,4) Minnick counter can be designed by extending the previous arguments. The implementation is shown in Fig. 4 and can similarly be described by the following expressions:

$$\begin{aligned} y_3 &= \operatorname{sgn}(v-8) \;=\; 8^+ \\ y_2 &= \operatorname{sgn}(v-4-8\cdot 8^+) \\ y_1 &= \operatorname{sgn}(v-2-4\cdot 4^+ - 4\cdot 8^+ - 4\cdot 12^+) \\ y_0 &= \operatorname{sgn}(v-1-2\cdot 2^+ - 2\cdot 4^+ - 2\cdot 6^+ - 2\cdot 8^+ \\ &\quad - 2\cdot 10^+ - 2\cdot 12^+ - 2\cdot 14^+). \qquad (4) \end{aligned}$$

v	y_2	y_1	y_0
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

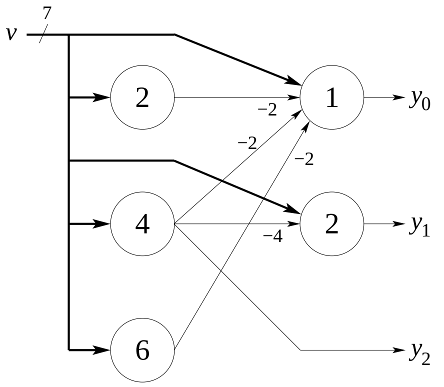

Fig. 3. The (7,3) counter truth table and the Minnick implementation

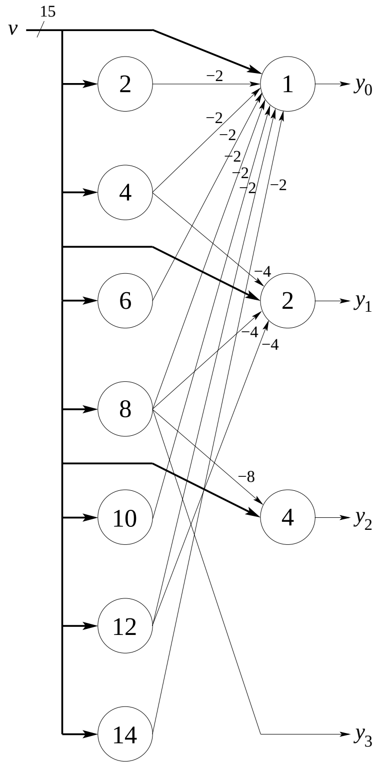

Fig. 4. The (15,4) Minnick counter

The circuit diagrams showing the (7,3) and (15,4) counter designs are shown in Fig. 5 and Fig. 6, respectively. The numbers next to the capacitors indicate the multiple of the unit capacitor. In both counters, the capacitive network which calculates the sum of the counter input bits is implemented

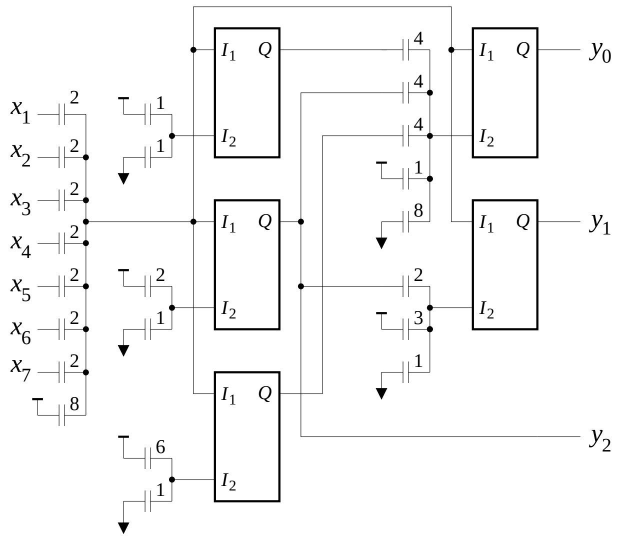
Fig. 5. Circuit diagram of the proposed STTL Minnick (7,3) counter

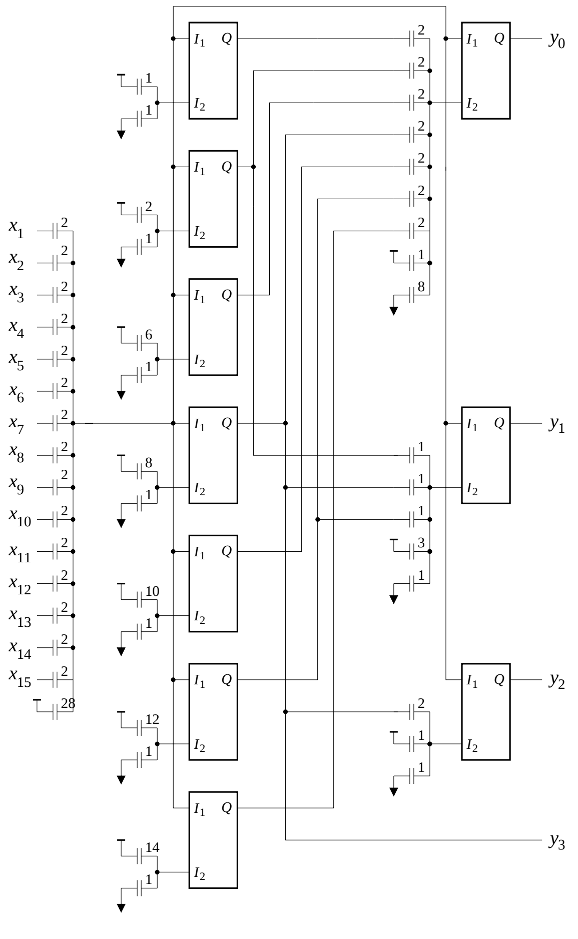
Fig. 6. Circuit diagram of the proposed STTL Minnick (15,4) counter

only once, and this value becomes one input of the sense amplifier in every STTL gate. The enable signals, E and E_b are not shown to improve clarity. The gates in the second layer are enabled after the outputs from the first layer are evaluated, as discussed in Section 3. The enable signals of one of the first layer gates drive the enable inputs of all second layer gates. The capacitors shown connected to Gnd and V_{dd} adjust the effective threshold of each STTL gate. The outputs of the first layer gates are connected to the capacitors that implement the negative weights. The inputs denoted by I_1 and I_2 in the Figures correspond to the ϕ and T inputs, respectively, in Fig. 2.

4.2. Partial Product Matrix Reduction Scheme

The reduction of the partial product matrix for a 32×32 direct multiplication of unsigned numbers is shown in Fig. 7. The partial product matrix in the top of the figure consists of 32 rows, each containing 32 bits, and is generated using an array of AND gates.

In the reduction scheme shown, Stage I employs the STTL (15,4), (7,3) counters and full adders to reduce the matrix to that shown in Stage II. Stage II has a maximum column height of 10 bits and is further reduced by using (15,4), (7,3) and (3,2) counters to obtain the matrix of height 4 shown in Stage III. Finally, two rows are obtained by using conventional (4:2) compressors [10] in Stage III. These two rows are usually the inputs to a fast carry-propagate adder (not shown) which calculates the final product. In Stage III it is more area efficient to use static CMOS (4:2) compressors than an equivalent threshold gate network. The (4:2) compressor proposed in [10] has a critical path delay of 3 XOR gates which is less than the delay of two STTL threshold gates. Hence the critical path delay of the partial product matrix reduction is less than 6 threshold gate delays.

During the reduction process, not all inputs in each counter are used, and the unused inputs are simply connected to logic 0. It is therefore possible to reduce the number of TL gates in some of the counters in the proposed scheme. For example, when using a (15,4) counter to add a column of 10 bits, the quantities 12^+ and 14^+ will be zero and need not be computed. This optimization was not considered in this evaluation.

Table 1 compares the TL gate count of this work with other 32×32 multiplication schemes that also have a critical path delay of six gates. For comparison, the full adder and (4:2) compressor are conservatively estimated as occupying the equivalent area of two and four threshold gates, respectively. The breakdown in terms of the number of each type of counter is only shown for this work. The total TL gate count is compared with two other known proposals for multiplication [11] [12] in the final row of the table. It shows that the scheme proposed here reduces the total number of

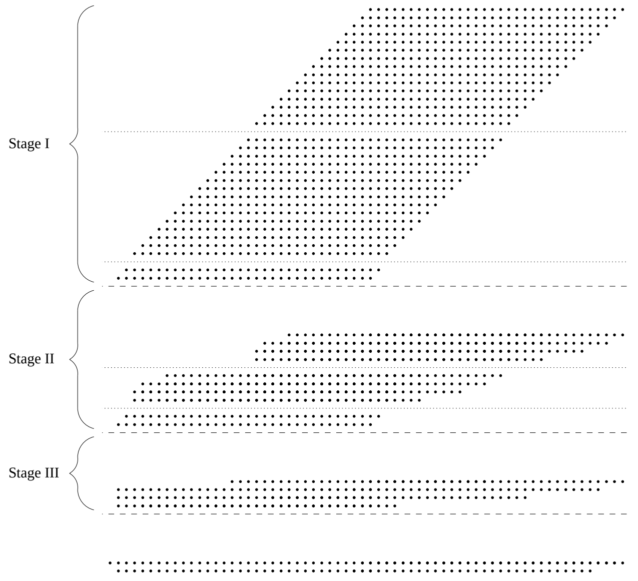

Fig. 7. Proposed reduction scheme for 32×32 partial product matrix

equivalent threshold gates by almost 50% compared to other schemes.

Table 1. Comparison of 32×32 TL multiplication schemes

	This work	Ref. [11]	Ref. [12]
Full Adders	17		
(4:2) Compressors	60		
(7,3) Counters	46		
(15,4) Counters	85		
Total TL Gate Count	1354	2678	3374

5. CONCLUSIONS

This paper presented compact designs for (15,4) and (7,3) counters based on Self-Timed Threshold Logic, and the counters were applied to the reduction of the partial product matrix in a 32×32 direct multiplication. The gate count of the resulting parallel multiplier was compared to other known schemes for threshold logic multiplication in depth 6 and it was shown that the proposed design requires almost 50% fewer gates.

Acknowledgments

The support of the Australian Research Council (ARC) and the Sir Ross and Sir Keith Smith Fund is gratefully acknowledged.

6. REFERENCES

[1] T. Shibata and T. Ohmi, “An intelligent MOS transistor featuring gate-level weighted sum and threshold operations,” in *IEDM, Technical Digest*, New York, NY, USA, December 1991, IEEE.

[2] M.J. Avedillo, J.M. Quintana, A. Rueda, and E. Jiménez, “Low-power CMOS threshold-logic gate,” *IEE Electronics Letters*, vol. 31, no. 25, pp. 2157–2159, December 1995.

[3] H. Özdemir, A. Kepkep, B. Pamir, Y. Leblebici, and U. Çiliniroğlu, “A capacitive threshold-logic gate,” *IEEE JSSC*, vol. 31, no. 8, pp. 1141–1149, August 1996.

[4] P. Celinski, J. F. López, S. Al-Sarawi, and D. Abbott, “Low power, high speed, charge recycling CMOS threshold logic gate,” *IEE Electronics Letters*, vol. 37, no. 17, pp. 1067–1069, August 2001.

[5] K. Kotani, T. Shibata, M. Imai, and T. Ohmi, “Clocked-neuron-MOS logic circuits employing auto-threshold-adjustment,” in *ISSCC Digest of Technical Papers*, 1995, pp. 320–321.

[6] P. J. Song and G. D. Micheli, “Circuit and architecture tradeoffs for high-speed multiplication,” *IEEE Journal of Solid State Circuits*, vol. 26, pp. 1184–1198, September 1991.

[7] E. E. Swartzlander, “Parallel counters,” *IEEE Transactions on Computers*, vol. C-22, pp. 1021–1024, 1973.

[8] R. C. Minnick, “Linear-Input Logic,” *IRE Transactions on Electronic Computers*, vol. EC-10, pp. 6–16, March 1961.

[9] Tijs Huisman, “Counters and multipliers with threshold logic,” M.S. thesis, Delft University of Technology, May 1995.

[10] V. G. Oklobdzija, D. Villeger, and S. S. Liu, “A method for speed optimized partial product reduction and generation of fast parallel multipliers using an algorithmic approach,” *IEEE Transactions on Computers*, vol. C-45, no. 3, pp. 294–305, March 1996.

[11] S. Vassiliadis and S. Cotofana, “Counters and multiplication with threshold logic,” in *Proceedings IEEE 30th Asilomar Conference on Signals, Systems and Computers*, California, USA, November 1996.

[12] R. Lauwereins and J. Bruck, “Efficient implementation of a neural multiplier,” in *Proc. 2nd Intern. Conference on Microelectronics for Neural Networks*, October 1991, pp. 217–230.