PRACTICAL LOW-COST CPL IMPLEMENTATIONS OF THRESHOLD LOGIC FUNCTIONS[1]

José M. Quintana[‡] María J. Avedillo[‡] Raúl Jiménez[†] Esther Rodríguez-Villegas[‡]
[‡]Instituto de Microelectrónica de Sevilla (IMSE-CNM), Avda. Reina Mercedes s/n, 41012 Sevilla, SPAIN
[†]Dpto. Ing. Electrónica de Sistemas Informáticos y Automática, Carr. Palos-La Rábida, 21071 Huelva, SPAIN
Phone: +34-955056666 Phone: +34-955056666 Phone: +34-959017368 Phone: +34-955056666
josem@imse.cnm.es avedillo@imse.cnm.es naharro@uhu.es esther@imse.cnm.es

[1]*This effort was partially supported by the Project 1FD 1997-2351 (TIC)*

ABSTRACT

This paper gives the rationale for the Complementary Pass-Transistor Logic (CPL) implementation of threshold gates for the design of complex logic functions. The interesting low-power properties of CPL circuits and the efficiency with which a broad class of functions can be implemented by threshold functions make the proposed methodology extremely useful. A number of implementation examples are given which illustrate the feasibility and versatility of the proposed technique and its potential as a low-cost design technique for CMOS technologies. Simulation results confirm our expectations.

1. INTRODUCTION

Circuit design based on threshold gates has been considered for many years an alternative to the traditional logic gate design procedure because of their acknowledged power. The power of the threshold gate design style lies in the intrinsic complex functions implemented by such gates, which allows for realizations that require less threshold gates than standard logic gates [1]. More recently, a number of theoretical results show that polynomial-size, bounded level networks of threshold gates can implement functions that require unbounded level networks of standard logic gates [2-4]. In particular, important functions like multiple-addition, multiplication, division, or sorting can be implemented by polynomial-size threshold circuits of small constant depth. Threshold gate networks have also been found useful in modeling nerve nets and brain organization, and with variable threshold (or weights) values they have been used to model learning systems, adaptive systems, self-repairing systems, pattern recognition systems, etc.

Threshold gates are based on the so-called majority or threshold decision principle, which means that the output value depends on whether the weighted arithmetic sum of its input values exceeds a threshold. Conventional simple logic gates, such as AND and OR gates, are special cases of threshold gates. Thus, threshold logic can treat conventional gates, as well as threshold gates in general, in a unified manner. However, the usefulness of threshold logic as a design alternative is determined by the availability, cost and capabilities of the basic building blocks, as well as by the existence of effective synthesis procedures.

Recently, a number of solutions for building CMOS threshold gates with integer weights and threshold levels have been reported [5-11]. All of them rely on representing each distinct weighted sum of inputs by an analogue voltage. Thus, they present limitations to implement threshold gates with many different sums, specially for low voltage operation. Pass-transistor logic styles have emerged as an attractive alternative to conventional CMOS logic [12-16]. In particular, Complementary Pass-Transistor Logic (CPL) is a well known low-power logic style [14-15]. Pass-transistor logic is attractive as fewer transistors are needed to implement important logic functions, smaller transistors and smaller capacitances are required, and it is faster than conventional CMOS. In this paper we link both concepts by presenting the rationale for CPL-based realizations of threshold gates. Implementation examples which demonstrate the feasibility and versatility of the proposed technique and its potential as a low-cost design technique for CMOS technologies are presented.

The following section presents a brief description of the threshold gate principle. Section 3 gives the rationale for steering realizations of threshold gates. Section 4 particularizes the procedure previously described for the case of complementary pass-transistor logic (CPL). Section 5 shows experimental results and, finally, some conclusions are given.

2. THE THRESHOLD GATE

A threshold gate has *n* binary input variables $x_1, x_2, \ldots, x_n$ and a binary output variable "y". The gate is specified by $(n+1)$ real numbers: the threshold T and a group of *n* weights $w_1, w_2, \ldots, w_n$, where each weight w_i (i=1,..., n) is associated to a specific input x_i. The output of a threshold gate is defined as $y = 1$ iff $\sum_{i=1}^{n} w_i x_i \geq T$ and $y = 0$ if this were not the case. Sums and products are the conventional, rather than the logical, operations, and the weights and threshold may have any real value. A threshold function with all input weights equal to 1 and threshold in m, $m \leq n$ is a special case of a TG and is represented as T_m^n, (m= 1, 2, ..., n), where $T_m^n = 1$ iff $\sum_{i=1}^{n} x_i \geq m$, and 0 otherwise.

In the following, we will specify the threshold element by means of a weights and threshold vector, TG; TG=$[w_1, w_2, \ldots, w_n; T]$. It can be deduced from the elemental properties [1] of the threshold functions shown in Figure 1, that if a function can be performed as a threshold gate, then selectively complementing the input and/or the output, a realization can be obtained using an element with positive values for both weights and threshold. Also, any threshold function may be obtained with integer weights [1].

$$\begin{tikzcd}
\substack{f(\mathbf{x})\\ [\mathbf{w}; T]} \arrow[r, "f'"] \arrow[d, "\mathbf{x}'"] & \substack{f'(\mathbf{x})\\ [-\mathbf{w}; 1-T]} \arrow[d, "f'(\mathbf{x}')"] \\
\substack{[-\mathbf{w}; T-\sum w_i]\\ f(\mathbf{x}')} \arrow[r, "f'"] & \substack{[\mathbf{w}; 1-T+\sum w_i]\\ f^d(\mathbf{x})}
\end{tikzcd}$$

Figure 1: Properties of threshold functions

3. STEERING LOGIC REALIZATIONS OF THRESHOLD GATES

In this section, a steering logic realization for threshold gates is discussed and compared with the traditional steering logic realization of symmetric functions. It is well known [17] that symmetric functions allow for a clever topologically regular implementation. They are the so-called Tally functions. A Tally circuit has n inputs $x_1, x_2, \ldots, x_n$ and $(n+1)$ outputs $z_0, z_1, z_2, \ldots, z_n$. The circuit counts the number of its inputs that are at logic 1 and asserts only the output with a subindex equal to this number. For example, if there are i inputs at logic 1, then z_i, and only z_i, is at logic one. The remaining n outputs are at logic 0. Traditional implementations for tally circuits use transmission gates as shown in Figure 2 for a single-input tally function.

3a. Threshold Functions with all Weights Equal to 1

Threshold gates could be implemented by considering that they can be formulated as particular symmetric functions, but this solution lacks efficiency concerning area overhead. The structure we propose for threshold gate implementations is simpler than traditional tally circuits. Figure 3 shows the proposed threshold gate basic structure when particularized to three input variables. Outputs z_1, z_2, z_3 are the threshold functions of three variables, with all the

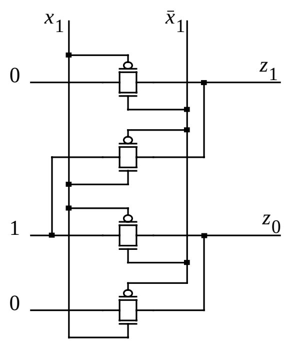
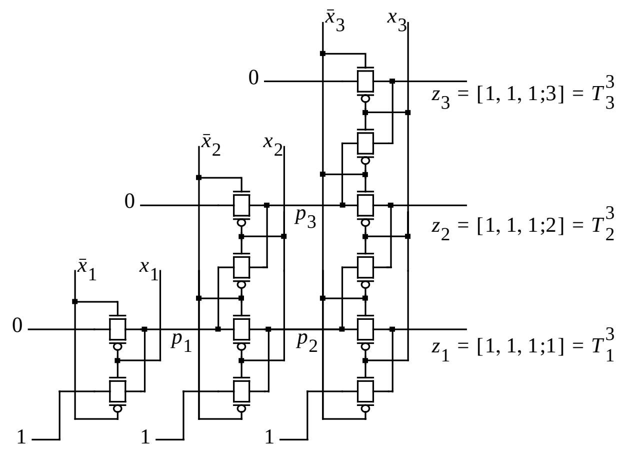


Figure 2: Single-input tally circuit implemented by using transmission gates



$\bar{x}_3$
x_3
0
$z_3 = [1, 1, 1;3] = T_3^3$
$\bar{x}_2$
x_2
0
p_3
$z_2 = [1, 1, 1;2] = T_2^3$
$\bar{x}_1$
x_1
0
p_1
p_2
$z_1 = [1, 1, 1;1] = T_1^3$
1
1
1

Figure 3: Proposed basic structure for 3-input threshold gates T_1^3, T_2^3, and T_3^3

weights equal to 1, and thresholds in 1, 2, and 3, respectively, *i.e.*, $z_i = T_i^3$, (i= 1, 2, 3). Depending on the setting of the control signals, the inputs are steered straight through the network (control signal at logic 0) or are shifted diagonally (control signal to logic 1). Checking that the proposed structure operates correctly can be done by considering the signal flow. For example, let us suppose input x_1 at a logic 1. Then, point p_1 is also at a logic 1. If now input x_2 is also at a logic 1, points p_2 and p_3 are also at a logic 1 (p_2 because the diagonal transmission of the logic 1 at the bottom, and p_3 through the diagonal transmission of logic 1 in p_1). Finally, if input x_3 is also at logic 1, z_1 is at logic 1 by diagonal transmission of logic 1 at the bottom, z_2 is at logic 1 by diagonal transmission of logic 1 in p_2, and z_3 is at logic 1 by diagonal transmission also of logic 1 in p_3. In the case that input x_3 is now at a logic 0, logic 1 in p_2 is steered to the z_1 output, logic 1 in p_3 is steered to the z_2 output, and a logic 0 from the left of the network is steered to the z_3 output. Finally, if the signal in point p_1 is considered, it can be easily shown that it is the same as input x_1 and so, transmission gates controlled by x_1 can be eliminated and input x_1 directly tied to transmission gates controlled by input x_2. The network is well formed, *i.e.*, for any combination of the control inputs, there is at least one conducting path from an input to the output. The proposed structure can be easily extended to implement the *n* threshold functions $T^n_1, T^n_2, \ldots, T^n_n$ of *n* inputs each, in such a way that each column is controlled by each input variable.

The example shown in Figure 3 simultaneously provides the network for a three-input OR gate (T^3_1), a three-input majority gate (T^3_2), and a three-input AND gate (T^3_3). If only a three-input majority gate is required, for example, the network is pruned as shown in Figure 4. Additionally, note that implementation obtained for the 3-input AND gate (OR gate) is the traditional pass-transistor one.

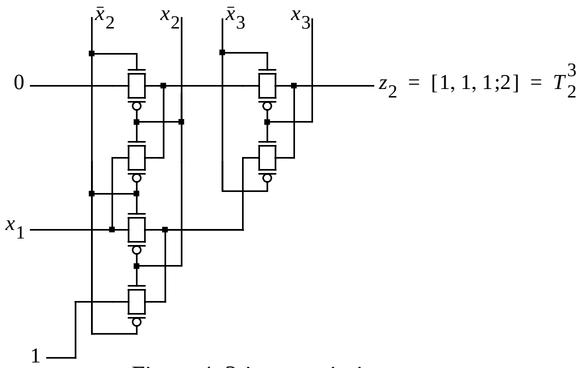
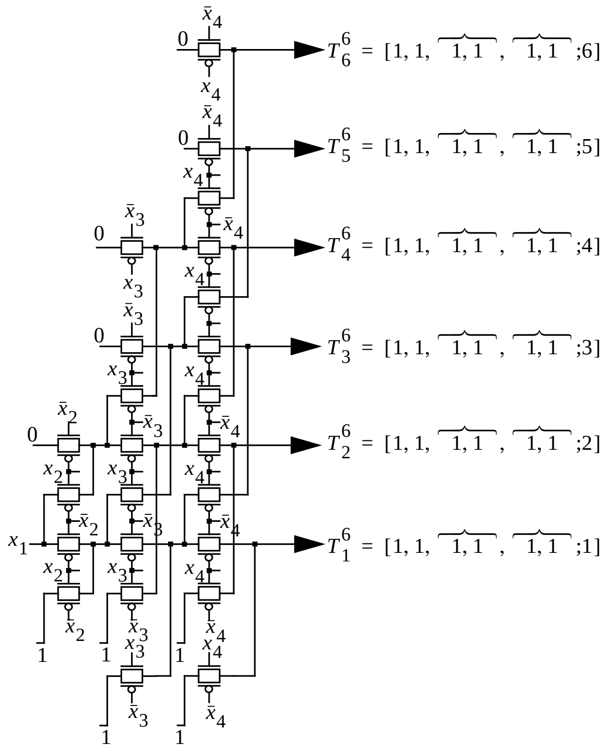


Figure 4: 3-input majority gate

3b. Threshold Functions with Arbitrary Weights

Now we will address the implementation of threshold functions with arbitrary integer weights. If input x_i has an associated weight of w_i, the problem can be solved by tying together as many inputs as necessary (given by w_i). This solution makes the design unnecessarily complex. It is easy to show that an arbitrary weight of w_i can be directly implemented in the column controlled by input x_i by replicating the transmission gates connected to logic 1 (at the bottom) and logic 0 (at the top), and altering by the diagonal connection pattern. Figure 5 shows the network which implements the six possible threshold functions which can be obtained with the following set of weights: $[1, 1, 2, 2]$. As the weight associated to variables x_3 and x_4 is 2, there are two transmission gates connected to logic 1 at the bottom, and another two transmission gates connected to logic 0 at the top for both columns. Also, the diagonal connection pattern has been modified. The normal diagonal pattern (for the column controlled by x_2) connects a transmission gate with the one placed in the row immediately above. For columns controlled by variables x_3 and x_4, this connection jumps one row and it is tied to the transmission gate placed two rows above. These considerations can be generalized and the complete structure for a network implementing an arbitrary threshold gate can be easily derived.



$T^6_6 = [1, 1, \overbrace{1, 1}, \overbrace{1, 1}; 6]$
$T^6_5 = [1, 1, \overbrace{1, 1}, \overbrace{1, 1}; 5]$
$T^6_4 = [1, 1, \overbrace{1, 1}, \overbrace{1, 1}; 4]$
$T^6_3 = [1, 1, \overbrace{1, 1}, \overbrace{1, 1}; 3]$
$T^6_2 = [1, 1, \overbrace{1, 1}, \overbrace{1, 1}; 2]$
$T^6_1 = [1, 1, \overbrace{1, 1}, \overbrace{1, 1}; 1]$
0
x_1
1
x_2
$\bar{x}_2$
x_3
$\bar{x}_3$
x_4
$\bar{x}_4$

Figure 5: Network implementing $[1, 1, 2, 2; k]$, $k = 1, \ldots, 6$

4. CPL-BASED THRESHOLD GATES

In this section we will prove that the steering logic structure proposed in the previous section can be easily transformed into a Complementary Pass-Transistor Logic (CPL) structure. The main concepts behind CPL are the use of complementary input/outputs, an nMOS pass-transistor logic network, and CMOS output inverters. Additionally, a pMOS latch can be added to decrease static power consumption and for swing restoration, as opposed to the conventional pull-up function. To obtain these characteristics, we will consider the circuits shown in Figure 6, which can be considered a “dual” of the other once because they are obtained by simply exchanging logic 0’s and logic 1’s.

Signals p_{21}, p_{22}, $\overset{*}{p}_{21}$, and $\overset{*}{p}_{22}$ are related by:

$$p_{21} = x_2 + \bar{x}_2 x_1 = x_2 + x_1$$

$$p_{22} = x_2 x_1$$

$$\overset{*}{p}_{21} = \bar{x}_2 \bar{x}_1 = \bar{p}_{21}$$

$$\overset{*}{p}_{22} = \bar{x}_2 + x_2 \bar{x}_1 = \bar{x}_2 + \bar{x}_1 = \bar{p}_{22}$$

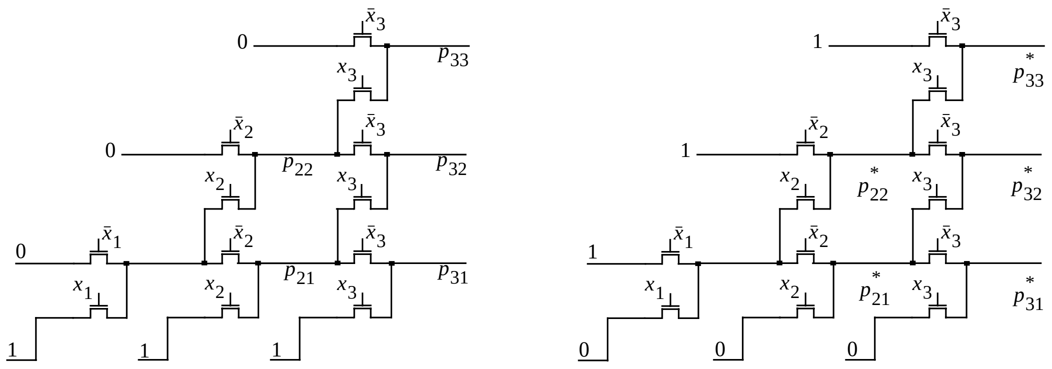


Figure 6: Basic structures for a CPL implementation of a threshold gate

In a similar way, and after simple logic manipulation, we have:

$$p_{31} = x_3 + \bar{x}_3 p_{21} = x_3 + p_{21}$$
$$p_{32} = x_3 p_{21} + \bar{x}_3 p_{22}$$
$$p_{33} = x_3 p_{22} = x_3 + p_{21}$$
$$\overset{*}{p}_{31} = \bar{x}_3 \overset{*}{p}_{21} = \bar{x}_3 \bar{p}_{21} = \bar{p}_{31}$$
$$\overset{*}{p}_{32} = x_3 \overset{*}{p}_{21} + \bar{x}_3 \overset{*}{p}_{22} = x_3 \bar{p}_{21} + \bar{x}_3 \bar{p}_{22} = \bar{p}_{32}$$
$$\overset{*}{p}_{33} = \bar{x}_3 + x_3 \overset{*}{p}_{22} = \bar{x}_3 + \bar{p}_{22} = \bar{p}_{33}$$

Through a simple exercise of mathematical induction we can show that the proposed structure makes it possible to obtain two sets of *n* output signals (for *n* input variables) of which one is the complement of the other. Moreover, these *n* outputs are the threshold gates $T^n_1, T^n_2, \ldots, T^n_n$, as it was proven in Section 3. Thus, a CPL implementation for a threshold function can be obtained by adding the CMOS inverters to the output lines and, if desired, the pMOS latch. Figure 7 shows a 3-input CPL majority gate (T^3_2), obtained from these considerations.

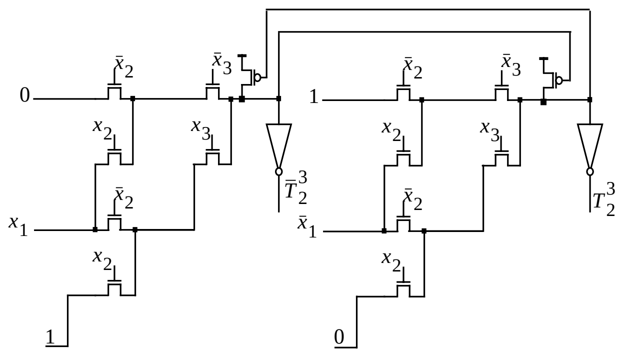


Figure 7: A 3-input CPL majority gate

To show the power of the proposed methodology, let us consider a more complex example. A set of functions which appear in the implementation of arithmetic circuits is given by:

$$F(x_1, \ldots, x_n) = x_n + x_{n-1}(x_{n-2} + x_{n-3}(x_{n-4} + x_{n-5}(\ldots + x_2(x_1)))) \quad , \qquad n \text{ odd} \tag{1}$$

All the functions in this set are threshold functions [18] with weights $w_k = F_k$ and threshold in F_n, where F_k is the *k*-th Fibonacci number, *i.e.*, each F_k is the sum of the two adjacent Fibonacci numbers on its immediate left, $F_k = F_{k-1} + F_{k-2}$ $(F_1 = F_2 = 1)$.

$$F(x_1, x_2, \ldots, x_n) = [1, 1, 2, 3, 5, \ldots, F_{n-1}, F_n; F_n] \tag{2}$$

The total weight for the threshold gate implementing F, $W_T(F)$, is given by:

$$W_T(F) = 1+1+2+\ldots+F_{n-1}+F_n = F_{n+2} - 1 \tag{3}$$

It is very difficult to implement these threshold gates by any of the other threshold gate implementation methods published in the literature. However, the CPL-based implementation we propose depends only on the number of variables, not on their associated weight. Therefore, when we particularize Eq. (2) to *n*=9, for example, function $F(x_1, \ldots, x_9)$ becomes $[1, 1, 2, 3, 5, 8, 13, 21, 34; 34]$ and the procedure developed in this section leads to a 36 transistor CPL implementation (without introducing the pMOS latch in the count), as shown in Figure 8.

Some practical considerations are in order. As the CPL-based design is a class of static pass-transistor logic, it inherits the design problems that are specific to that circuit class. In particular, as we have chains of nMOS transistors in the nMOS networks, after a careful study of the situation additional intermediate swing restoration may be needed (see the arrows in Figure 8 for their possible placement, after 4-transistor paths). This consideration does not introduce any conceptual problem in our implementation because intermediate signals in both nMOS networks are each one complemented to the other.

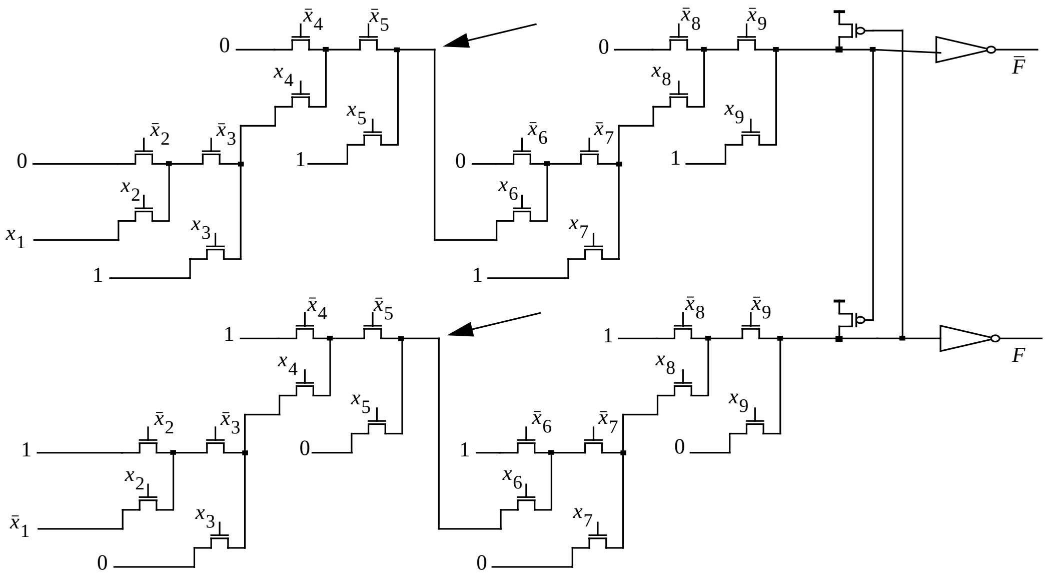


Figure 8: CPL-based threshold gate implementation of function $F(x_1, \ldots, x_9) = [1, 1, 2, 3, 5, 8, 13, 21, 34;34]$

5. PRACTICAL RESULTS

In order to prove both the power of the approach and the correction of the designs produced by our methodology, we have selected the 5-input majority function (*i.e.*, a function which is 1 if three, four or five inputs are at logic 1). A conventional CMOS implementation for this function is very expensive as it requires $\binom{5}{3}$ product terms of three literals each (60 transistors without counting the OR gate needed to add them). This function is efficiently implemented by the threshold gate $T_3^5 = [1, 1, 1, 1, 1;3]$ which needs 38 transistors to be realized, as shown in Figure 9. This circuit has been laid out in an 0.6µm double poly CMOS technology as shown in Figure 10. The occupied area is $50\mu\text{m} \times 30\mu\text{m}$. Operation under process and ambient parameters has been validated through extensive HSPICE simulations including Monte Carlo simulations and simulations using different standard worst-case device parameters at a power supply, V_{DD}, of 3.3V. Figure 11 depicts some of the simulation results for a typical load. The waveforms at the top trace are the circuit input signals. A new input combination is applied each 5ns every. Circuit outputs T_3^5 (middle trace) and $\bar{T}_3^5$ (bottom trace) are shown below. Pre- and post-layout simulations have been depicted.

6. CONCLUSIONS

A design methodology for implementing CPL-based threshold gates has been presented. It provides an easy and systematic way to build threshold functions. Implementation of threshold gates by the circuits thus obtained does not depend on the weight associated to each input. This is a distinguishing characteristic which differentiates these implementations from others which have been reported. Several examples of circuit implementations have been presented which take advantage of both the CPL logic design style and the power of the threshold gate based paradigm. Experimental results confirm the suitability of the proposed methodology.

7. REFERENCES

[1] S. Muroga, *Threshold Logic & its Applications*, Wiley-Interscience 1971.

[2] K.Y. Siu and V.P. Roychowdhury, "On Optimal Depth Threshold Circuits For Multiplication And Related Problems", *SIAM J. Discrete Math*, vol. 7, no. 2, pp 284-292, 1994.

[3] K.Y. Siu, V.P. Roychowdhury and T. Kailath, "Toward Massively Parallel Design of Multipliers", *J. of Parallel and Distributed Computing*, 24, pp. 86-93, 1995.

[4] R. Impagliazzo, R. Paturi and M. E. Saks, "Size-Depth Tradeoffs for Threshold Circuits", *SIAM J. Comput.*, vol. 26, no. 3, pp. 693-707, 1997.

[5] C.L. Lee, C-W. Jen: "Bit-sliced median filter design based on majority gate", *IEE Proc.-G,* vol. 139, No. 1, pp. 63-71, Feb. 1992.

[6] C.L. Lee, C-W. Jen, "CMOS Threshold gate and networks for order statistic filtering", *IEEE Tr. on Circ. and Syst.* -I, vol. 41, No. 6, pp. 453-456, June 1994.

[7] J.M. Quintana, M.J. Avedillo and A.Rueda, "Hazard-free edge-triggered D flipflop based on Threshold Gates", *Elect. Let*. vol. 30, no. 17, pp. 1390-1391, Aug. 1994.

[8] T. Shibata, T. Ohmi, "A functional MOS transistor featuring gate level weighted sum and threshold operations", *IEEE J. Solid-State Circ.,* vol. 39, pp. 1444-1445, 1992.

[9] W. Weber, et al., "On the application of the Neuron MOS Transistor Principle for Modern VLSI Design", *IEEE Tran. Electron Devices*, vol. 43, No. 10, pp. 1700-1708, October 1996.

[10] J.M. Quintana, M.J. Avedillo, A. Rueda y C. Baena, "Practical Low-Cost CMOS realization of Complex Logic Functions", *Proc. of European Conf. on Circuit Theory and Design*, ECCTD'95, pp. 51-54.

[11] E. Rodriguez, G. Huertas, M.J. Avedillo, J.M. Quintana, and A. Rueda, "A Practical Floating-Gate Muller-C Element Using vMOS Threshold Gates," *to appear in the Special Issue on Floating Gate Circuits and Systems, IEEE Trans. on Circuits and Systems -II: Analog and Digital Signal Processing.*

[12] A.P. Chandrakasan, S. Sheng, and R.W. Brodersen, "Low-Power CMOS Digital Design", *IEEE J. Solid-State Circ.*, vol. 27, pp. 473-483, April 1992.

[13] C.F. Law, S.S. Rofail, and K.S. Yeo, "A Low-Power 16×16-b Parallel Multiplier Utilizing Pass-Transistor Logic", *IEEE J. Solid-State Circ.*, vol. 34, no. 10, pp. 1395-1399, October 1999.

[14] K. Yano, T. Yamanaka, T. Nishida, M. Saito, K. Shimohigashi, A. Shimizu, "A 3.8-ns CMOS 16×16-b Multiplier Using Complementary Pass-Transistor Logic", *IEEE J. Solid-State Circ.*, vol. 25, no. 2, pp. 388-395, April 1990.

[15] A.P. Chandrakasan and R.W. Brodersen, *Low-Power Digital CMOS Design*. Norwell, MA: Kluwer, 1995.

[16] R. Zimmermann and W. Fichtner, "Low-Power Logic Styles: CMOS Versus Pass-Transistor Logic," *IEEE J. Solid-State Circuits*, vol. 32, no. 7, pp. 1079-1090, July 1997.

[17] C. Mead and L. Conway, *Introduction to VLSI Systems*, Addison-Wesley, 1980.

[18] E. Rodríguez-Villegas, M.J. Avedillo, J.M. Quintana, and A. Rueda, "Threshold Logic Based Adders Using Floating-Gate Circuits", *4th CSCC, 2000 World Multiconference*, July 2000.

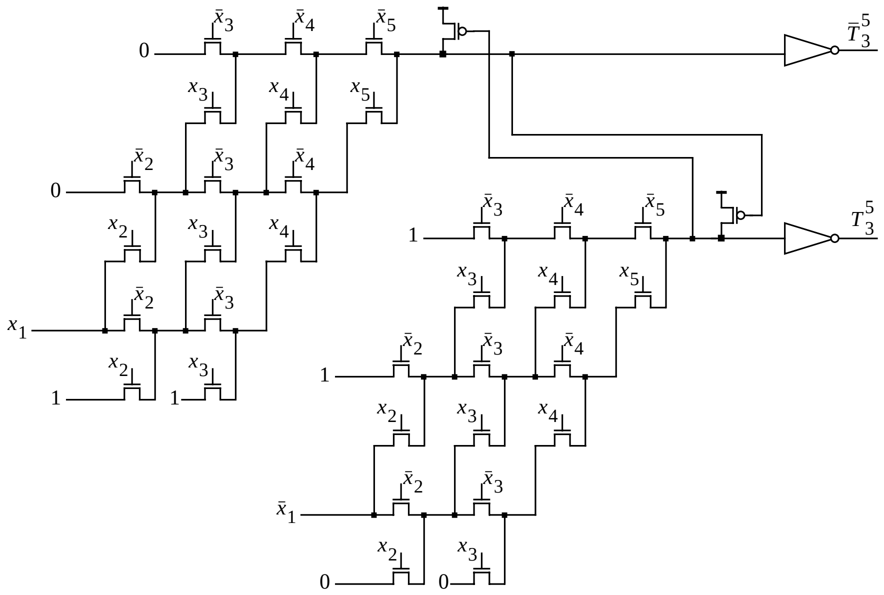


Figure 9: 5-input majority gate implemented as CPL-based threshold gate

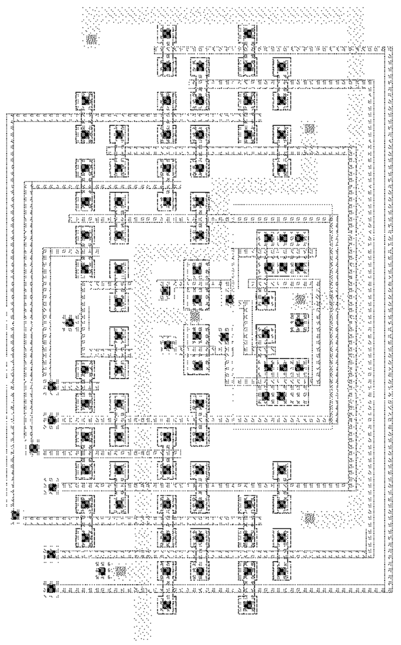


Figure 10: Layout of the CPL 5-input majority gate

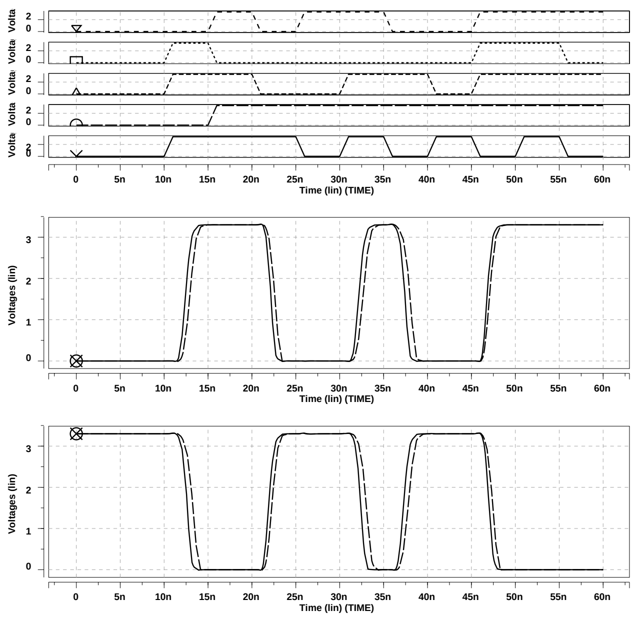


Figure 11: Simulated waveforms for the CPL 5-input majority gate