A FAMILY OF LOW DEPTH, THRESHOLD LOGIC, CARRY LOOKAHEAD ADDERS

Peter Celinski, 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

The main result of this paper is the development of a low depth carry lookahead addition technique based on threshold logic. The adders designed using this method are shown to have a very low logic depth and therefore reduced area and improved speed over conventional logic implementations.

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 the high performance processor market in order to meet the high density requirements of advanced VLSI processors. The importance of low power is also evident in portable and aerospace applications, and is related to issues of reliability, packaging, cooling and cost.

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 traditional AND-OR-NOT logic-gate based design. However, lack of efficient physical realizations has meant that TL has, until recently, had little impact on VLSI. Efficient TL gate realizations have recently become available, and a number of applications based on TL gates have demonstrated its ability to achieve high operating speed and significantly reduced area [1][2][3] [4].

We begin in Section 2 by giving a brief overview of threshold logic (TL). This is followed by a description of the proposed carry lookahead addition scheme in Section 3. Two adders based on the proposed scheme are shown in Section 4. Finally a brief conclusion is given 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.

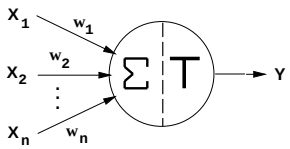

Fig. 1. Threshold Gate Model

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 may be written in a more compact form using the sgn function as

$$Y = \text{sgn}\left(\sum_{i=1}^{n} w_i X_i - T\right). \quad (2)$$

The sgn function is defined as $\text{sgn}(x) = 1$ if $x \geq 0$ and $\text{sgn}(x) = 0$ if $x < 0$. Any threshold function may be computed with positive integral weights and a positive real threshold, and all Boolean functions can be realized by a threshold gate network of depth at most two. A TL gate can be programmed to realize many distinct Boolean functions by adjusting the threshold T. For example, an n-input TL gate with $T = n$ will realize an n-input AND gate and

by setting $T = n/2$, the gate computes a majority function. This versatility means that TL offers a significantly increased computational capability over conventional AND-OR-NOT logic.

3. CARRY LOOKAHEAD ADDITION WITH THRESHOLD LOGIC

Addition is one of the most critical operations performed by VLSI processors. Adders are used in ALUs, floating-point arithmetic units, memory addressing and program counter updates. The critical requirement of the adder is speed, but low power dissipation and area efficiency have become increasingly important in recent years. The key factor in the proposed addition scheme is the introduction of high-valency threshold logic carry generate and propagate cells, which results in reduced logic depth addition networks, and hence reduced area and power dissipation.

Carry lookahead is a well-known technique for decreasing the latency of addition by reducing the logic depth to $O(log_2 w)$, where w is the word length of the addends. It is one of the fastest addition algorithms, and allows significant design trade-offs to be made in terms of latency, area and power. The addition problem can be expressed in prefix notation in terms of generate, G^i_j, propagate, P^i_j, and carry c_j signals at each bit position j for a width w adder as follows:

$$G^i_j = \begin{cases} a_j.b_j, & \text{for} \quad i = j \\ G^k_j + P^k_j.G^i_{k-1}, & \text{for} \quad j \geq k > i \end{cases} \tag{3}$$

$$P^i_j = \begin{cases} a_j + b_j, & \text{for} \quad i = j \\ P^k_j.P^i_k, & \text{for} \quad j \geq k > i, \end{cases} \tag{4}$$

where $j = 0, \ldots, w-1$, c_j denotes the carry generated at position i and c_{-1} denotes the carry into the LSB position. Assuming that $c_{-1} = 0$, then the carry signal at position j, c_j, is given by:

$$c_j \; = \; G^0_j. \tag{5}$$

The direct approach to implementing this scheme in, for example, static CMOS is not practical for any useful wordlength $w \geq 16$, since the amount of circuitry required to assimilate the MSB carry becomes prohibitive. For this reason, and also because of the associative nature of the expressions for G^j_i and P^j_i, carry lookahead adders are usually built using a parallel tree structure. The threshold logic approach can, however, be used to design circuits which implement carry lookahead addition in a more direct and efficient way than the static CMOS prefix-tree approach.

A modified set of the Boolean Equations (3)-(5) will now be derived in a form suitable for implementation in threshold logic. We will take advantage of the high fan-in capability of TL to design high valency threshold logic prefix-cells, that is, prefix-cells which compute group propagate, group generate and carry signals from a large number of input bits.

The input operands $(a_i,\ b_i)$, $i = 0, \ldots, w-1$, are grouped into n-bit blocks. The first stage starts with the computation of the group-generate and group-propagate signals (G^{j-n+1}_j and P^{j-n+1}_j) for each block, directly from the input operands. A carry is generated in a group if the sum of the n bits in the group exceeds (is strictly greater than) the maximum number representable by n sum bits. Therefore a group-propagate signal G^{j-n+1}_j is 1 if the sum of the n bits in the group exceeds the maximum number representable by n sum bits. Similarly, the group propagates a carry originating in the neighbouring group of lower significance and the group propagate signal P^{j-n+1}_j is 1 if the sum of the n bits in the group is equal to or greater than the maximum number representable by n sum bits. This may be written in general equation form as:

$$G^{j-n+1}_j = \text{sgn}\Big(\sum_{k=j-n+1}^{j} 2^{k-(j-n+1)}(a_k + b_k) - (2^{n+1} - 1)\Big) \tag{6}$$

$$P^{j-n+1}_j = \text{sgn}\Big(\sum_{k=j-n+1}^{j} 2^{k-(j-n+1)}(a_k + b_k) - (2^{n+1} - 2)\Big). \tag{7}$$

Equations (6) and (7) are exactly in the same form as Equation (2) which describes the operation of a threshold gate. The input weights for calculating G^{j-n+1}_j and P^{j-n+1}_j are the same, and the gate thresholds differ by 1.

An example will serve to illustrate the ideas. Consider a 3-bit grouping of the input bits $(a_5, b_5, a_4, b_4, a_3, b_3)$. The group generate signal G^3_5 is 1 if the sum of the inputs is greater than the largest number representable in the three sum bits (s_5, s_4, s_3), which is 7. The group propagate P^3_5 signal is 1 if the sum of the inputs is greater than or equal to 7. This can be expressed as:

$$G^3_5 \; = \; \text{sgn}(4a_5 + 4b_5 + 2a_4 + 2b_4 + a_3 + b_3 - 8) \tag{8}$$

$$P^3_5 \; = \; \text{sgn}(4a_5 + 4b_5 + 2a_4 + 2b_4 + a_3 + b_3 - 7). \tag{9}$$

An expression for calculating G^0_j may be written by combining the intermediate group generate and group propagate signals in the following way:

$$G^0_j \; = \; G^k_j + P^k_j G^l_{k-1} + P^k_j P^l_{k-1} G^m_{l-1} + P^k_j P^l_{k-1} P^m_{l-1} \ldots P^z_{x-1} G^0_{z-1}. \tag{10}$$

Equation (10) can be interpreted as expressing the partitioning of the w inputs into contiguous blocks in which it is determined where a carry signal is generated and propagated. Such an expression may easily be converted into TL form. This is illustrated by the following example for G^0_{15}, where

we partition the 16 bits into 4 groups, and use 4 bit group generate and group propagate signals as follows:

$$\begin{aligned} G_{15}^{0} &= G_{15}^{12} + P_{15}^{12}G_{11}^{8} + P_{15}^{12}P_{11}^{8}G_{7}^{4} + P_{15}^{12}P_{11}^{8}P_{7}^{4}G_{3}^{0} \\ &= \mathrm{sgn}\Big(8G_{15}^{12} + 4P_{15}^{12} + 4G_{11}^{8} + 2P_{11}^{8} + 2G_{7}^{4} + \\ &\qquad P_{7}^{4} + G_{3}^{0} - 8\Big). \end{aligned} \tag{11}$$

Finally, the sum bits are computed from the truth table for addition as follows:

$$\begin{aligned} s_j &= \mathrm{sgn}(a_j + b_j + c_{j-1} - 2c_j - 1) \\ &= \mathrm{sgn}(a_j + b_j + c_{j-1} + 2\bar{c}_j - 3), \end{aligned} \tag{12}$$

where we have used $-c_j = \bar{c}_j - 1$ so that all weights are positive.

4. LOW DEPTH CARRY LOOKAHEAD ADDERS

By exploiting the parallelism inherent in the computation of carry signals as expressed in Equation (10), we can construct carry lookahead adders of significantly reduced logic depth compared to previous prefix tree approaches.

One possible 16-bit carry lookahead tree structure is shown in Fig. 2. The black cells consist of two CRTL gates and compute GP_j^i signals, the grey cells compute the carry signals G_j^0 and the white cells compute the sum according to Equation (12). The adder has a depth of only 4 gates. This is a significant improvement compared to, for example, a conventional 16-bit Brent-Kung adder which has a critical path consisting of 9 gates (7 gates for the prefix-tree, one gate for generating p_i and g_i and finally one XOR gate for computing the sum bits).

To achieve bit-level pipelined operation, the input operands (a_j, b_j) must propagate through the CLA because the sum bits s_j are computed from the input operands as well as the two carries (c_{j-1}, c_j). Therefore each cell in the CLA must also include two D-latches, which are not shown in Fig. 2. This would result in a compact and potentially low-power pipelined adder suitable for DSP applications. It can also be said that the proposed CLA has a number of very desirable properties. The adder consists primarily of only one type of CRTL gate, which means only one gate requires careful design and optimization. The regularity of each cell also means that networks of the type shown in Fig. 2 are highly suitable for automated layout generation.

5. CONCLUSIONS

A low depth carry lookahead addition technique based on threshold logic was presented. A 16-bit adder was designed based on this technique and was shown to have a low logic depth and significantly reduced area compared to common static CMOS switch implementations.

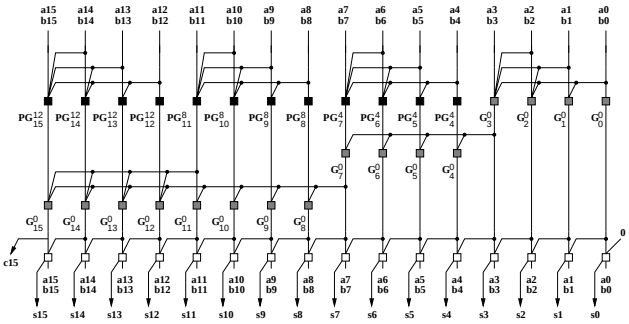

Fig. 2. Structure of a 16-bit carry lookahead adder with 4-bit grouping and no carry-in

References

[1] 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.

[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, Dec. 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.