A SYSTEMATIC METHODOLOGY FOR DESIGNING AREA-TIME EFFICIENT PARALLEL-PREFIX MODULO $2^n - 1$ ADDERS

G. Dimitrakopoulos[†] *, H. T. Vergos*[†‡]*, D. Nikolos*[†‡]*, and C. Efstathiou*[§]

[†]Computer Engineering and Informatics Dept., University of Patras, 26500 Patras, Greece
[‡]Computer Technology Institute, 3 Kolokotroni Str., 26221 Patras, Greece
[§]Informatics Dept., TEI of Athens, Ag. Spyridonos St., 12210 Egaleo, Athens, Greece.

ABSTRACT

In this paper a systematic methodology for designing parallel-prefix modulo $2^n - 1$ adders, for every n, is introduced. The resulting modulo $2^n - 1$ adders feature minimum logical depth and bounded fan-out loading. Additionally, an optimization technique is proposed, which aims to the reduction of redundant operators that appear on the parallel-prefix carry computation trees. Performance data reveal that the reduced structures achieve area×time complexity reduction of up to 46% when compared to previously reported designs.

1. INTRODUCTION

Modulo $2^n - 1$ arithmetic is used in a variety of applications, ranging from the Residue Number System [1], up to fault-tolerant computer systems [2], and checksum computations in high-speed networks [3]. Therefore, high-speed low-cost modulo $2^n - 1$ adders are essential building blocks for all these applications.

Several proposals have already appeared to the modulo $2^n - 1$ adder design problem. To cut down the addition delay, single and two level CLA modulo $2^n - 1$ adders have been proposed in [4]. Even faster designs, based on the parallel-prefix carry computation approach have appeared recently in [5] and [6]. In [5] the most-significant carry of a parallel-prefix integer addition is fed back to all the bits from 0 to $n - 1$ through an extra prefix level, thus increasing the delay, and leading to a maximum wire fan-out n. On the contrary, in [6] a new theory for the design of high-speed modulo $2^n - 1$ adders was introduced based on the idea of recirculating the carry generate and carry propagate signals, instead of the traditional end-around carry approach. Although the fundamental theory and a general architecture were presented in [6], no straightforward design method was given, when $n \neq 2^k$. This task was left to the intuition of the designer. When n is of the form 2^k, it was shown that the adders proposed in [6] outperform those proposed in [5].

The contribution of the current paper is twofold. First, a formal framework for the design of parallel-prefix modulo $2^n - 1$ adders, for every n, is introduced. The resulting adders feature minimum logical depth and bounded fan-out. Second, an optimization technique for reducing the number of operators required by the carry computation unit of the designed parallel-prefix adders, is proposed. The technique is based on identifying and removing redundant operators; therefore neither the logical depth nor the fan-out is increased by its application. The reduction in the number of operators can be of up to 20%, while experimental results based on static CMOS implementations reveal that area×time complexity reductions from 4% to 46% can be achieved.

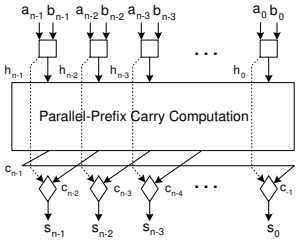

Figure 1: The structure of a parallel-prefix modulo $2^n - 1$ adder.

The rest of the paper is organized as follows. Section 2 revisits the basics of parallel-prefix formulation of modulo $2^n - 1$ addition. In Section 3 the proposed design methodology is introduced, while Section 4 presents the novel area-reduction technique. Performance data are provided in Section 5. Finally, conclusions are drawn in Section 6.

2. PARALLEL-PREFIX MODULO $2^n - 1$ ADDITION

In this section, we revisit the basics of the modulo $2^n - 1$ addition when it is treated as a parallel-prefix problem [6]. Let $A = \{a_{n-1}a_{n-2}\ldots a_0\}$ and $B = \{b_{n-1}b_{n-2}\ldots b_0\}$ represent the two operands to be added, and $S = \{s_{n-1}s_{n-2}\ldots s_0\}$ denote their sum. The addition can be performed by a three-stage circuit, as shown in Fig. 1. The preprocessing stage produces the carry generate g_k, the carry propagate p_k, and the half-sum h_k, bits, according to the relations $g_k = a_k \cdot b_k$, $p_k = a_k + b_k$, and $h_k = a_k \oplus b_k$, for $0 \leq k \leq n-1$, where $\cdot$, $+$, and $\oplus$ denote the logical AND, OR, and exclusive-OR operations, respectively. Using the carry generate/propagate pairs (g_k, p_k), the carry-computation stage computes the carries c_k using the associative prefix operator $\circ$, which is defined in [7] as $(g, p) \circ (g', p') = (g + p \cdot g', p \cdot p')$. According to [6], each carry c_k is produced by combining the carry generate and propagate pairs from 0 through k and the carry generate and propagate pairs from $k+1$ to $n-1$, using the formula,

$$(G_k, P_k) = (g_k, p_k) \circ (g_{k-1}, p_{k-1}) \circ \ldots \circ (g_0, p_0) \circ (g_{n-1}, p_{n-1}) \circ \ldots \circ (g_{k+1}, p_{k+1}), \quad (1)$$

where, $c_k = G_k$. The carries generated by the carry-computation unit, along with the half-sum bits produced by the preprocessing stage, are used in the final stage to compute the sum bits s_k, according to $s_k = h_k \oplus c_{k-1}$, for $0 \leq k \leq n-1$, while $c_{-1} = c_{n-1}$. It should be noted that in contrast to integer addition, the number of the carry generate/propagate pairs (g_k, p_k) that have to be associated for the generation of each carry is equal to n.

The parallel-prefix carry computation structures are often represented as Directed Acyclic Graphs (DAGs). In these graphs

0-7803-7762-1/03/$17.00 ©2003 IEEE

the prefix operators are represented by nodes which are placed on a grid of rows (prefix levels) and bit columns.

3. THE PROPOSED DESIGN METHODOLOGY

In this section, based on the theory presented in [6], a systematic methodology for the design of parallel-prefix modulo $2^n - 1$ adders, for every n, is introduced. The goals of the proposed design methodology are at first to generate regular structures of minimum logic depth, that is structures composed of m prefix levels, with $m = \lceil \log_2 n \rceil$, and secondly to maintain the fan-out bounded to 2.

On the prefix levels of the carry computation unit, group carry generate/propagate terms are produced.The number of carry generate/propagate pairs (g_k, p_k) that are associated in the ith prefix level, to form a group generate/propagate term, is called the *length* of the group generate/propagate term and is denoted here as V_i. For example the term $\{ (g_1, p_1) \circ (g_0, p_0) \}$ has length 2, while the term $\{ (g_2, p_2) \circ (g_1, p_1) \circ (g_0, p_0) \}$ has length 3. We are interested in the generation of valid group generate/propagate terms, that is, group terms that preserve the order of the carry generate/propagate pairs (g_k, p_k), as dictated by (2).

The connections between the operators of successive prefix levels can generate valid group carry generate/propagate terms of different lengths.

Example 1: Assume that on the 1st prefix level of a modulo $2^5 - 1$ adder, all the valid group generate/propagate terms of length 2, $\{(g_4, p_4) \circ (g_3, p_3)\}$, $\{(g_3, p_3) \circ (g_2, p_2)\}$, $\{ (g_2, p_2) \circ (g_1, p_1) \}$, $\{ (g_1, p_1) \circ (g_0, p_0) \}$, $\{ (g_0, p_0) \circ (g_4, p_4) \}$, have been implemented. Depending on the connections between the 1st and the 2nd prefix level, valid group generate/propagate terms of length 3 or 4 can be produced:

1. Carry generate/propagate terms of length 2 with one bit-position difference are combined to generate valid triplets, by sharing a common term and using the well-known idempodency property [8], i.e., $\{ (g_4, p_4) \circ (g_3, p_3) \} \circ \{ (g_3, p_3) \circ (g_2, p_2) \} \Leftrightarrow \{ (g_4, p_4) \circ (g_3, p_3) \circ (g_2, p_2) \}$.
2. Two terms with of length 2 with two bit-positions difference are combined to generate a valid quadruple, such as $\big\{ \{ (g_4, p_4) \circ (g_3, p_3) \} \circ \{ (g_2, p_2) \circ (g_1, p_1) \} \big\}$. □

In our design we assume that on each prefix level, group generate/propagate terms of the equal length are produced. Then, in order to maintain the bounded fan-out 2 loading, a method similar to recursive-doubling [9] is followed. In particular, the length of the group terms of the ith level is selected to be equal to $V_i = \left\lceil \frac{V_{i+1}}{2} \right\rceil$, which when unrolled gives, $V_m = n$, $V_{m-1} = \lceil \frac{V_m}{2} \rceil = \lceil \frac{n}{2} \rceil$, $V_{m-2} = \left\lceil \frac{V_{m-1}}{2} \right\rceil = \left\lceil \frac{\lceil \frac{n}{2} \rceil}{2} \right\rceil = \lceil \frac{n}{4} \rceil$, ... Therefore it can be derived that the length of the terms implemented on the ith prefix level, $1 \leq i \leq m$, equals to

$$V_i = \left\lceil \frac{n}{2^{m-i}} \right\rceil, \tag{2}$$

and V_0 is assumed to be equal to one ($V_0 = 1$).

Among the various connections between the nodes (operators) of the ith prefix level, the connections that guarantee the generation of valid group terms in the $(i+1)$st prefix level should be identified. Therefore, the *rotate distance* between the ith and $(i+1)$st levels is introduced, which is denoted as D_{i+1}^i. The rotate distance D_{i+1}^i between the ith and the $(i+1)$st prefix level is equal to $V_{i+1} - V_i$, irrespective of the number of carry generate/propagate pairs (g_k, p_k) that are shared through idempodency [8], as proven by Proposition 1.

Proposition 1. *The rotate distance D_{i+1}^i between any two successive prefix levels i and $(i+1)$ equals to $V_{i+1} - V_i$, with $0 \leq i \leq m$.*

Proof. Two cases are distinguished according to the way the group generate/propagate terms of the $(i+1)$st prefix level are produced.

- The group carry generate/propagate terms of the $(i+1)$st prefix level are generated without sharing any carry generate/propagate pairs (g_k, p_k) of the group terms of the ith level. Thus, it follows that the bit-position difference between the operators of the ith level that are combined in the $(i+1)$st prefix level, equals to V_i, or else $D_{i+1}^i = V_i$, and the length of the terms on the $(i+1)$st level is $V_{i+1} = 2V_i$, which implies that $V_{i+1} - V_i = V_i$. Therefore it follows that $D_{i+1}^i = V_{i+1} - V_i$.
- The group carry generate/propagate terms of the $(i+1)$st prefix level are generated after sharing r carry generate/propagate pairs (g_k, p_k) of the group terms of the ith level. Then, since all the terms that appear on the ith level are of equal length, a group carry generate/propagate term is produced in the $(i+1)$st prefix level by combining two terms with $V_i - r$ bit-positions difference. The length of the resulting term is equal to $V_{i+1} = 2V_i - r \Rightarrow V_{i+1} - V_i = V_i - r$, which implies that the rotate distance (bit-positions difference) is equal to $D_{i+1}^i = V_{i+1} - V_i$.

From both cases it is derived that $D_{i+1}^i = V_{i+1} - V_i$. □

The rotate distance determines the connection between the operators of two successive prefix levels, and the proposed connection topology is shown in Fig. 2. The notation $|x|_q$ is used to denote the operation x modulo q, and the coordinate pair (i, j) denotes the placement of an operator on the jth bit column of the ith prefix level, with $0 \leq j \leq n-1$ and $1 \leq i \leq m$.

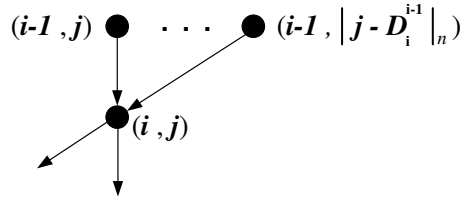

Figure 2: The connection of the operators between two successive prefix levels.

The coordinates of the operators connected via the lateral wires include the modulo operation $\left| j - D_i^{i-1} \right|_n$ since, according to Eq. (2), the generation of each carry c_k, with $0 \leq k \leq n-1$ imposes the association of both carry generate/propagate pairs (g_{k_1}, p_{k_1}) with $k_1 \leq k$ and carry generate/propagate pairs (g_{k_2}, p_{k_2}) with $k_2 > k$. In case that the operation $\left| j - D_i^{i-1} \right|_n$ produces a negative value, then, based on the properties of the modulo arithmetic, the value $n - \left| j - D_i^{i-1} \right|_n$ is used. The rotate distance is selected to be equal for all operators since this is sufficient to maintain the wire fan-out bounded to 2. This assumption is proven by the following proposition.

Proposition 2. *If all the operators of each prefix level have the same rotate distance then the maximum wire fan-out between successive prefix levels is two.*

Proof. Assume that there is at least one operator of the $(i+1)$st prefix level placed on the jth column that has rotate distance $D^* > D^i_{i+1}$, where D^i_{i+1} is the rotate distance of the rest $n-1$ operators. Thus, the operator with rotate distance D^* connects to the operators of the ith level with coordinates, (i,j) and $(i,|j-D^*|_n)$. Since $D^* > D^i_{i+1}$ there exist an operator placed on $(i+1, j-D^*+D^i_{i+1})$ that connects to the operators $(i, j-D^*+D^i_{i+1})$ and $(i, |(j-D^*+D^i_{i+1})-D^i_{i+1}|_n)$, or equivalently $(i,|j-D^*|_n)$. Therefore the operator placed on $(i,|j-D^*|_n)$ connects to $(i+1,j)$ and $(i+1, j-D^*+D^i_{i+1})$ through two lateral connections and to $(i+1,|j-D^*|_n)$ through a vertical connection, implying a wire fan-out equal to 3.

In a similar manner it can be proven that if there is at least one operator of the $(i+1)$st prefix level placed on the jth column that has rotate distance $D^* < D^i_{i+1}$, then there is one wire with fan-out 3. Hence, when the operators of each prefix level share the same rotate distance, the maximum wire fan-out is equal to 2. □

Design Procedure : Given the word length n of the modulo 2^n-1 adder, the proposed design methodology is described by the following steps:

1. Using Eq. (1) calculate the length of the terms on the ith level, for $1 \le i \le m$.
2. Calculate the rotate distance between any two successive prefix levels, according to Proposition 1.
3. Connect the operators between the $(i-1)$st and the ith level using the connection topology of Fig. 2. According to Proposition 2, in order to maintain fan-out 2 the same rotate distance should be used for each connection.

Example 2: In order the proposed methodology to be clarified a modulo 2^5-1 adder is designed. The minimum prefix levels required by a modulo 2^5-1 adder are $m = \lceil \log_2 5 \rceil = 3$. Following the proposed design methodology the length of the carry generate/propagate terms of each level are:

$$\{V_0, V_1, V_2, V_3\} = \left\{1, \left\lceil \frac{5}{2^{3-1}} \right\rceil, \left\lceil \frac{5}{2^{3-2}} \right\rceil, \left\lceil \frac{5}{2^{3-3}} \right\rceil \right\} = \{1,2,3,5\}$$

After computing the lengths V_i, the rotate distance between successive levels has to be determined. It follows that,

$$\{D^0_1, D^1_2, D^2_3\} = \{V_1 - V_0, V_2 - V_1, V_3 - V_2\} = \{1,1,2\},$$

which leads to the parallel-prefix computation unit of Fig. 3. □

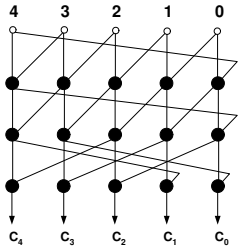

Figure 3: The carry-computation unit of a parallel-prefix modulo 2^5-1 adder designed according to the proposed methodology.

4. GENERATION OF REDUCED-AREA CARRY COMPUTATION UNITS

In the previous section a methodology for the implementation of the carry-computation unit of parallel-prefix modulo 2^n-1 adders, which assumed that n operators are present on each prefix-level, was presented. We will refer to these carry computation units as *full carry trees* in the rest of the paper. In this section we give an optimization procedure that removes redundant operators and produces reduced-area computation units, hereafter denoted as *reduced carry trees*. Note that the proposed optimization procedure does not increase either the prefix levels or the fan-out loading of the carry computation unit.

Fig. 4a presents a prefix operator with two inputs X_A and X_B that implements the relation $Y = X_A \circ X_B$. Since the wire fan-out is bounded to 2, then its output is propagated in two buses, namely Y_A and Y_B, where $Y_A = Y_B = Y$, as shown in Fig. 4a.

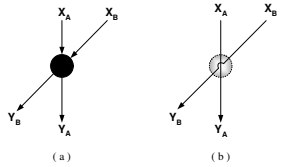

Figure 4: Operator Replacement

Our optimization procedure is based on the following observation: An operator placed on the ith prefix level can be removed if it does not affect the functionality of the next prefix levels. Therefore, starting from the first level of the full carry tree, the operators are conditionally examined for removal one-by-one. If the removal does not affect the group carry generate/propagate terms of the next level then the operator is removed. In case that the removal leads to non-valid terms in the following prefix level then the operator can not be removed.

The removal of a black-dot operator involves the elimination of its boolean-logic function while it fully maintains the wire connections. The removal is therefore performed according to the following rule: The vertical wire input directly connects to the vertical wire output, and the lateral wire input directly connects to the lateral wire output. Following Fig. 4b, and the introduced connection rule, it holds that $Y_A = X_A$ and $Y_B = X_B$. It should be noted that, when the removal of an operator leads to a valid prefix modulo 2^n-1 carry tree, the bounded wire fan-out property is also preserved, since no wire is either added or removed.

Example 3: A full-prefix tree of a modulo 2^5-1 adder, generated according to the methodology presented in Section 3, is shown in Fig. 5a. The group carry generate/propagate terms that are implemented on the nodes R_1 and R_2 are given by: $R_1 = \{\{(g_4,p_4)\circ(g_3,p_3)\}\circ\{(g_3,p_3)\circ(g_2,p_2)\}\} = \{(g_4,p_4)\circ(g_3,p_3)\circ(g_2,p_2)\}$, and $R_2 = \{\{(g_0,p_0)\circ(g_4,p_4)\}\circ\{(g_4,p_4)\circ(g_3,p_3)\}\} = \{(g_0,p_0)\circ(g_4,p_4)\circ(g_3,p_3)\}$ In Fig. 5b the $(1,4)$ operator has been removed, while its vertical and lateral connections have been preserved. In this case the group generate/propagate terms that are implemented on the nodes R^*_1 and R^*_2, are equal to, $R^*_1 = \{\{(g_4,p_4)\}\circ\{(g_3,p_3)\circ(g_2,p_2)\}\}$ and $R^*_2 = \{\{(g_0,p_0)\circ(g_4,p_4)\}\circ\{(g_3,p_3)\}\}$, which are equal to terms of the nodes R_1 and R_2 in the full-prefix tree. No other node needs to be checked since they are not affected by the removal of the specific operator. Therefore, the removal of node $(1,4)$ leads to a valid parallel-prefix modulo 2^5-1 tree carry

Table I: Area(μm^2) and Time(ns) results achieved by the reduced and full-carry trees, and the adders presented in [5]. The AT(%) columns present the Area×Time Savings achieved by the reduced-carry tree versus the other designs.

n	Reduced		Full Tree			[5] Ladner–Fisher			[5] Kogge-Stone		
	Area	Time	Area	Time	AT(%)	Area	Time	AT(%)	Area	Time	AT(%)
5	2094	1.46	2363	1.46	11%	1860	1.76	7%	2219	1.82	24%
6	2566	1.46	2835	1.46	9%	2333	1.84	13%	2781	1.85	27%
9	4612	1.75	5060	1.75	9%	3751	2.25	4%	4827	2.27	26%
12	6209	1.75	6747	1.75	8%	5463	2.69	26%	6693	2.37	31%
20	12142	2.03	13039	2.03	7%	9465	3.45	25%	12928	3.23	41%
24	14570	2.03	15646	2.03	7%	11714	3.82	34%	15626	3.48	46%

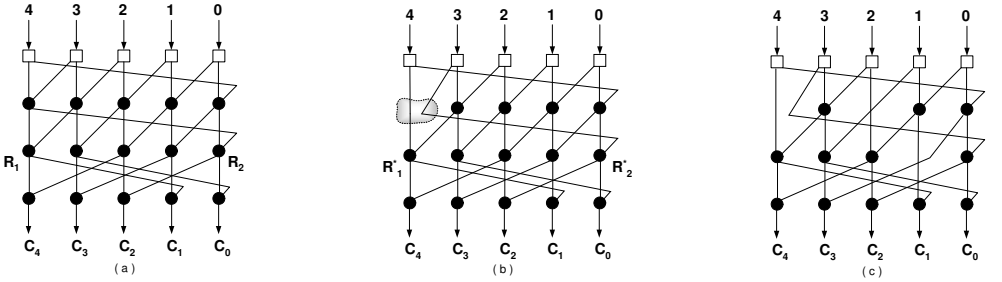

Figure 5: The reduction process for the modulo $2^5 - 1$ adder of Example 3.

generator. Following the same procedure for all remaining nodes, the resulting reduced prefix-tree is shown in Fig. 5c. □

Different reductions can be achieved, depending on the value of n. In case that n is in the range $2^{m-1}+1 \leq n \leq 2^{m-1}+2^{m-2}$, for a certain logic depth $m = \lceil \log_2 n \rceil$, then the number of operators removed from the full-carry tree is approximately equal to $\lceil \frac{n}{2} \rceil$. In the rest of the cases no reduction is possible.

5. EXPERIMENTAL RESULTS

The proposed full and reduced carry-tree adders were compared against the prefix adders proposed in [5], for several values of n, when either a Ladner-Fischer [10] or a Kogge-Stone [9] carry-computation unit is used for the adders of [5]. All adders were described in Verilog HDL and mapped in the UMC-VST 25 implementation technology ($0.25\mu m$, up to 5-metal layers, 1.8 / 3.3V), using the Design Compiler tool set of Synopsys®.

Table I summarizes the results. Since the proposed structures require one less prefix level than those of [5], both full and reduced carry tree adders offer smaller execution latency. Moreover, since their fan-out is kept constant at 2, their delay highly depends on the number of prefix levels required. Therefore adders with the same number of prefix levels (for example the modulo 2^9–1 and 2^{12}–1 adders) have the same execution latency. This is not the case for the adders proposed in [5] because the feedback carry of the last-stage prefix operators has a fan-out loading equal to n. We have also included in Table I the area×time savings achieved by the proposed reduced-carry trees compared to the full-carry trees and the adders of [5]. As Table I indicates, in all examined cases, the proposed reduced-carry adders exhibit the best performance, achieving area×time savings ranging from 4% to 46%.

6. CONCLUSIONS

High-speed parallel-prefix modulo $2^n - 1$ adders can be useful in a variety of computer applications. In [6] the fundamental theory was given, along with a method to design them, when $n = 2^k$. In the rest cases, the designer had either to use his intuition or to rely on the structures proposed in [5] that have degraded performance. In this paper we have presented a systematic way to design parallel-prefix modulo $2^n - 1$ adders, for every n, with the minimum logic depth and a constant fan-out loading of 2. We have further introduced an optimization procedure to eliminate redundant prefix operators in the carry computation logic. Experimental results indicate that the proposed adders, besides being faster than all known structures, they also offer the best performance under the area×time metric.

7. REFERENCES

[1] M. A. Soderstrand, W. K. Jenkins, G. A. Jullien, and Fred J. Taylor, *Residue Number System Arithmetic: Modern Applications in Digital Signal Processing*, IEEE Press, 1986.

[2] B. J. Johnson, *Design and Analysis of Fault-Tolerant Digital Systems*, AddisonWesley Publishing Company, 1989.

[3] F. Halsall, *Data Communications, Computer Networks and Open Systems*, Addison Wesley, 1996.

[4] C. Efstathiou, D. Nikolos, and J. Kalamatianos, "Area-Time Efficient Modulo $2^n - 1$ Adder Design," *IEEE Trans. Circ. Syst. II*, vol. 41, no. 7, pp. 463–467, Jul. 1994.

[5] R. Zimmerman, "Efficient VLSI Implementation of Modulo $(2^n \pm 1)$ Addition and Multiplication," in *Proc. of 14th IEEE Symposium Computer Arithmetic*, April 1999, pp. 158–167.

[6] L. Kalampoukas, D. Nikolos, C. Efstathiou, H. T. Vergos, and J. Kalamatianos, "High-Speed Parallel-Prefix Modulo $2^n - 1$ Adders," *IEEE Trans. Comp.*, vol. 49, no. 7, pp. 673–680, Jul. 2000.

[7] R. P. Brent and H. T. Kung, "A Regular Layout for Parallel Adders," *IEEE Trans. Comp.*, vol. 31, no. 3, pp. 260–264, Mar. 1982.

[8] T. Lynch and E. E. Swartzlander, "The Redundant Cell Adder," in *Proc. of 10th IEEE Symposium Computer Arithmetic*, Jun. 1991, pp. 165–170.

[9] P. M. Kogge and H. S. Stone, "A parallel algorithm for the efficient solution of a general class of recurrence equations," *IEEE Trans. Comp.*, vol. C-22, pp. 786–792, Aug. 1973.

[10] R. E. Ladner and M. J. Fischer, "Parallel prefix computation," *JACM*, vol. 27, no. 4, pp. 831–838, Oct. 1980.