$$U_i = \left\{v \in U \,\middle|\, n_1 \geq i, n_0 = i\right\} \text{ for all } 1 \leq i \leq \left\lfloor \frac{n+1}{3} \right\rfloor,$$
$$U_n = \left\{v \in U \,\middle|\, n_1 = n\right\} \cup \left\{v \in U \,\middle|\, n_1 = n-1, n_0 = 1\right\}.$$

For $v \in U_i$ and for all $1 \leq i \leq \left\lfloor \frac{n+1}{3} \right\rfloor$, there exists an L-path from vertex $f(i+1)$ to v and therefore, $d(f(i+1), v) \leq n - (2i+1)$. Hence we have $U_i \subseteq N_{n-(2i+1)}(f(i+1))$ for $1 \leq i \leq \left\lfloor \frac{n+1}{3} \right\rfloor$.

Furthermore, we have $d(f(2), v) \leq 2$ for $v \in U_n$ and therefore, $U_n \subseteq N_{n-3}(f(2))$. Hence the theorem follows. □

It follows from Theorem 3, the optimal transmitting time for $UB(2, n)$ is at most $n-1$. For $UB(3, n)$ and $UB(4, n)$, the optimal transmitting time is either n or $n+1$ due to Theorem 2. We have the following conjecture which appears to be difficult to prove:

CONJECTURE 1. *The optimal transmitting time is $n-1$ for $UB(2, n)$, and n for $UB(3, n)$ and $UB(4, n)$.*

5 CONCLUSIONS

We have solved the shortest path problem on undirected de Bruijn networks $UB(d, n)$. Using this result, we have obtained a transmitting scheme on binary de Bruijn networks $UB(2, n)$. We have also proved that the optimal transmitting problem is trivial for $UB(d, n)$ when $d \geq 5$. We believe that the result on the shortest paths in $UB(d, n)$ can be also extended to solve the shortest path problem on hierarchical networks constructed from undirected de Bruijn networks. Furthermore, this result can help in solving other broadcasting-related problems.

ACKNOWLEDGMENTS

The authors are grateful to Drs. Jean-Claude Bermond, Lih-Hsing Hsu, and Ming-Tat Ko for useful discussions of this work. Encouragement from Professor D. Frank Hsu is highly appreciated in preparing this paper.

REFERENCES

[1] N. Alon, "Transmitting in the n-Dimensional Cube," *Discrete Applied Math.*, vol. 37/38, pp. 9–11, 1992.
[2] J.C. Bermond, *Discrete Applied Math.*, special issue on *Interconnection Networks*, vol. 37/38, 1992.
[3] J.C. Bermond, Z. Liu, and M. Syska, "Mean Eccentricities of de Bruijn Networks," Rapport de Recherche INRIA, no. 2,114, 1993.
[4] J.C. Bermond and C. Peyrat, "de Bruijn and Kautz Networks: A Competitor for the Hypercube?" *Hypercube and Distributed Computers*, F. André and J.P. Verjus, (eds.), pp. 279–293. Elsevier Science Publishers, 1989.
[5] J. Bond, *Construction de Grands Réseaux d'Interconnexions*, Thése de Troisíeme Cycle, Université de Paris-Sud, Orsay, 1984.
[6] N.G. de Bruijn, "A Combinatorial Problem," *Proc. Akademe Van Wetenschappen*, vol. 49, pp. 758–764, 1946.
[7] A.-H. Esfahanian and L. Hakimi, "Fault-Tolerant Routing in de Bruijn Communication Networks," *IEEE Trans. Computers*, vol. 34, pp. 778–788, 1985.
[8] T.Y. Ho, L.H. Hsu, and T.Y. Sung, "Transmitting on Various Network Topologies," *Networks* vol. 27, pp. 145-157, 1996.
[9] D.F. Hsu, *Networks*, Special Issue on *Interconnection Networks and Algorithms*, vol. 23, 1993.
[10] Z. Liu, "Optimal Routing in the de Bruijn Networks," *Proc. 10th Int'l Conf. Distributed Computing Systems*, pp. 537–544, May 1990, Paris, France.
[11] C. Peyrat, *Vulnérabilit é dans les grand réseaux d'interconnexion*, PhD thesis, Université de Paris-Sud, Centre d'Orsay,, 1984.
[12] D.K. Pradhan and S.M. Reddy, "A Fault-Tolerant Communication Architecture for Distributed Systems," *IEEE Trans. Computers*, vol. 31, pp. 863–870, 1982.
[13] P. Weiner, "Linear Pattern Matching Algorithms," *Proc. IEEE 14th Annual Symp. Switching and Automata Theory*, pp. 1-11, 1973.

2-1 Addition and Related Arithmetic Operations with Threshold Logic

Stamatis Vassiliadis, Sorin Cotofana, and Koen Bertels

Abstract—In this paper we investigate the reduction of the size for small depth feed-forward linear threshold networks performing binary addition and related functions. For n bit operands we propose a depth-3 $O(\frac{n^2}{\log n})$ asymptotic size network for the binary addition with polynomially bounded weights. We propose also a depth-3 addition of optimal $O(n)$ asymptotic size network and a depth-2 comparison of $O(\sqrt{n})$ asymptotic size network, both with $O(2^{\sqrt{n}})$ asymptotic size of weight values. For existing architectural formats we show that our schemes, with equal or smaller depth networks, substantially outperform existing schemes in terms of size and fan-in requirements and on occasions in weight requirements.

Index Terms—Computer arithmetic, binary adders, binary comparison, majority circuits, threshold logic, neural networks.

1 INTRODUCTION AND MAIN RESULTS

A linear threshold gate with a Boolean output $F(X)$ is defined by:

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

with $\mathcal{F}(X) = \sum_{i=1}^{n} w_i x_i + \psi$.

Given that such a model can compute arbitrary Boolean functions it has been the subject of numerous studies concerning its theoretical capabilities, see for example [1], [2], [3], [4], [5], [6]. Furthermore, there is evidence of direct implementation of threshold logic at the device level, see for example [7], [8], [9].

In this paper we investigate feed-forward linear threshold gates based networks for addition and addition related operations. Regarding such operations the following has been established using threshold logic based parallel networks:

- For the binary addition, Siu et al. [4], [10] suggested that each bit of the sum is computable with depth-2 networks with a network size of $O(n^4)$ and that the network size can be reduced to $O(n^2)$ for depth-3 networks.
- In [4] it has been indicated that the comparison function, performed on two operands of length n, can be computed in depth-2 networks with size of $O(n^4)$. Further with depth-3 networks, it has been suggested that the comparison can be realized with size of $O(n)$. Roychowdhury and al [11] suggested that the comparison can be computed in depth-3 networks with size of $O(\frac{n}{\log n})$ and polynomially bounded weights.

We investigate the reduction of the network size for depth-3 networks for addition and depth-2 networks for comparison. The main theoretical conclusions of the paper can be summarized as follows:

- *S. Vassiliadis and S. Cotofana are with the Electrical Engineering Department, Delft University of Technology, P.O. Box 5031, 2600 GA Delft, The Netherlands. E-mail: {Stamatis, Sorin}@duteca.et.tudelft.nl.*
- *K. Bertels is with the University of Namur (FUNDP), Rempart de la Vierge 8, 5000 Namur, Belgium. E-mail: Bertels@chaos.eco.fundp.ac.be.*

Manuscript received Apr. 1, 1994; revised June 30, 1995.
For information on obtaining reprints of this article, please send e-mail to: transcom@computer.org, and reference IEEECS Log Number C96075.

- Addition can be performed by a depth-3 network with the size in the order of $O(\frac{n^2}{\log n})$ and polynomially bounded weights.
- Addition can be performed by a depth-3 network[1] with $6n + 2\left\lceil \frac{n}{\lceil \sqrt{n} \rceil} \right\rceil$ size (i.e., of optimal $O(n)$ size complexity), a maximum fan-in of $2\left\lceil \frac{n}{\lceil \sqrt{n} \rceil} \right\rceil + 3$ and a maximum weight size of $2^{\lceil \sqrt{n} \rceil}$. It is not known if optimal $O(n)$ size depth-3 networks with polynomial weights are possible.
- The comparison of two n-bit operands with carry can be computed by a depth-2 network with $2\left\lceil \frac{n}{\lceil \sqrt{n} \rceil} \right\rceil + 1$ size (i.e., of $O(\sqrt{n})$ size complexity), a maximum fan-in of

$$MAX\left\{2\left\lceil \frac{n}{\lceil \sqrt{n} \rceil} \right\rceil + 1,\ 2\lceil \sqrt{n} \rceil\right\}$$

and a maximum weight size of $2^{\lceil \sqrt{n} \rceil}$. It is not known if $O(\sqrt{n})$ size depth-2 networks with polynomial weights are possible.

Concerning practical situations, represented by existing architectural formats, we show that our schemes provide sizable advantages over other schemes. In particular we show the following:

- The proposed addition scheme with polynomially bounded weights requires up to 71% threshold gates and 28% fan-in for the realization of 32-bit adders and up to 47% threshold gates and 18% fan-in for the realization of 64-bit devices when compared to the Siu et al. scheme [4], known to be the best schemes thus far for small depth and size networks for addition.
- The proposed $O(n)$ size addition scheme requires up to 18% threshold gates for the realization of 32-bit adders and up to 9% for the realization of 64-bit adders when compared to the Siu et al. scheme [4]. Our scheme implies a maximum weight value twice (for 32 bit operands) or four times (for 64 bit operands) the maximum weight value deduced from [4], but it provides an 8.53, respectively 13.47 times lower fan-in.
- For equal delay our scheme requires up to 18% gates, 50% weights, and 28% fan-in for the realization of 32-bit comparators and up to 13% gates, equal weights, and 20% fan-in for the realization of 64-bit devices , when compared to the Siu et al. scheme [4]. When compared with Roychowdhury and al scheme [11] it requires up to 94% gates, 25% weights, and 75% fan-in for the realization of 32-bit comparators and up to 83% gates, 50% weights, and 92% fan-in for the realization of 64-bit devices.

The presentation is organized as follows: In Section 2 we present the proposed schemes for addition and addition related functions. Section 3 contains comparisons between our approaches and what is known as the state of the art for some usual dimensions of operands and Section 4 some concluding remarks.

2 Recursive Formulae for Binary Addition

Binary addition requires the computation of the carry and the sum. We assume that the operands are partitioned into groups. In order to produce the carry equations, for a group i of length l, we define two new quantities, α_i, (the carry-force quantity) and β_i, (the carry-preserve quantity) defined by the following:

- *carry-force.* $\alpha_i = 1$ when the group's sum has a value $[2^l]^{+}_{=}$ and 0 otherwise.[2]
- *carry-preserve.* $\beta_i = 1$ when the group's sum has a value $[2^l - 1]^{+}_{=}$ and 0 otherwise.

The theorem to follow introduces a carry computation using threshold logic.

THEOREM 1. *For any given group i, the carry-out of the group i, C_i can be computed by:*

$$C_i = sgn\{\gamma_i - 1\}$$

with $\gamma_i = 2^i[\alpha_i + \beta_i - 1] + \gamma_{i-1}$ for $0 \le i$ *and* $\gamma_{-1} = C_{in}$.

PROOF. By induction. Given that the expression for the carry presumably computes the true carry C_i it must be that $\gamma_i - 1 \ge 0$ when the true carry of the addition is $C_i = 1$ and that $\gamma_i - 1 < 0$ when the true carry for the addition is $C_i = 0$.

basis. Trivial with proper substitutions.

step. Assume that the theorem holds true for $k - 1$ prove that it is also true for k.

Assuming that the theorem holds true for $k - 1$ it is implied that:

- If the true carry for the addition $C_{k-1} = 1$ then $\gamma_{k-1} \ge 1$.
- If the true carry for the addition $C_{k-1} = 0$ then $\gamma_{k-1} < 1$.
- Further, by removing the recurrence and with substitutions it can be proven that the maximum value of γ_{k-1} is $MAX\{\gamma_{k-1}\} = 2^k$ and the minimum is $MIN\{\gamma_{k-1}\} = -2^k + 1$.

The carry C_{k-1} is the carry into the group i thus the logical expression for the carry-out is $C_k = \alpha_k + \beta_k C_{k-1}$, and it must be proven that $C_k = sgn\{2^k[\alpha_k + \beta_k - 1] + \gamma_{k-1} - 1\}$ is equivalent to this logical expression. The logical expression dictates to consider, after exclusion of irrelevant cases,[3] four distinct cases:

Case 1. (if $\alpha_k = 1$ then $C_k = 1$ (independent of the C_{k-1} value)). If $C_{k-1} = 1$ then $\gamma_{k-1} \ge 1$ and because $\alpha_k = 1$ implying also $\beta_k = 1$, $C_k = 1$. If $C_{k-1} = 0$ then $\gamma_{k-1} < 1$. Given that $MIN\{\gamma_{k-1}\} = -2^k + 1$, and because $\alpha_k = 1$ implying $\beta_k = 1$, γ_k in the worse case scenario is:

$$\gamma_k = 2^k[\alpha_k + \beta_k - 1] + \gamma_{k-1} = 2^k - 2^k + 1 = 1,$$

thus $\gamma_k - 1 \ge 0$ and $C_k = 1$. Consequently, independent of the C_{k-1}, $\gamma_k - 1 \ge 0$ and $C_k = 1$ when $\alpha_k = 1$.

Case 2. (if $\beta_k = C_{k-1} = 1$ and $\alpha_k = 0$, then $C_k = 1$). If $C_{k-1} = 1$ then $\gamma_{k-1} \ge 1$ and because $\alpha_k = 0$ and $\beta_k = 1$, $\gamma_k - 1 \ge 0$ and $C_k = 1$.

Case 3. (if $\beta_k = 1$, $C_{k-1} = 0$, and $\alpha_k = 0$,then $C_k = 0$). If $C_{k-1} = 0$ then $\gamma_{k-1} < 1$, because $\alpha_k = 0$ and $\beta_k = 1$, $\gamma_k - 1 < 0$ and $C_k = 0$.

Case 4. (if $\beta_k = \alpha_k = 0$, independent of C_{k-1}, $C_k = 0$). Because $\beta_k = \alpha_k = 0$, when $C_{k-1} = 0$ $\gamma_k - 1 < 1$ and obviously $C_k = 0$. When $C_{k-1} = 1$ because the maximum value $MAX\{\gamma_k - 1\} = 2^k$, $\gamma_k = 2^k[\alpha_k + \beta_k - 1] + \gamma_{k-1} = -2^k + 2^k = 0$, and consequently $\gamma_k - 1 < 0$ thus the carry-out $C_k = 0$ independent of the C_{k-1} value. □

1. It is interesting to note that this scheme allows also an implicit construction of a depth-2 network for the addition with the size in the order of $O(n)$.

2. We use $x^{+}_{=}$ and $x^{-}_{=}$ in order to denote greater or equal and less than or equal to x, respectively.

3. I.e., the exclusion of the cases where $\alpha_k = 1$ and $\beta_k = 0$.

COROLLARY 1. *Assuming that the carry-in into the addition is C_{in}, the carry-out of the group i, C_i, can be computed by:*

$$C_i = sgn\left\{\sum_{m=0}^{i} 2^m(\alpha_m + \beta_m) + C_{in} - 2^{i+1}\right\}$$

PROOF. Trivial. By removing the recursion on γ_i. □

Regarding the comparison function it can be observed that, using linear threshold feed-forward networks, it can be computed by

$$C(X, Y) = sgn\left\{\sum_{i=0}^{n-1} 2^i(x_i - y_i)\right\}.$$

Assume that the operation is performed with n-bit represented unsigned numbers then after the inversion of the operands and the "hot one" addition the following holds true:

COROLLARY 2. *The comparison can be computed by a depth-2 linear threshold network of size $2\left\lceil\frac{n}{\lceil\sqrt{n}\rceil}\right\rceil + 1$, with the weight values at most $2^{\lceil\sqrt{n}\rceil}$ and with an upper bound of $2\lceil\sqrt{n}\rceil + 1$ for the maximum fan-in.*

PROOF. Divide the operands into groups of length x. Thus, there are $\lceil\frac{n}{x}\rceil$ groups all (but possibly the last one) with the same length. The comparison can be computed by computing the carry-out of the 2 – 1 binary addition with carry thus it can be computed by:

First Level. Let $0 \le m \le \lceil\frac{n}{x}\rceil - 1$, compute

$$\alpha_m = sgn\left\{\sum_{r=0}^{l-1} 2^r(X_r + Y_r) - 2^l\right\}$$

and

$$\beta_m = sgn\left\{\sum_{r=0}^{l-1} 2^r(X_r + Y_r) - (2^l - 1)\right\},$$

where $l - 1$ is either $x - 1$ or the length of the last group minus 1.

Second Level. Let $i = \lceil\frac{n}{x}\rceil - 1$, compute the carry-out

$$C_{out} = sgn\left\{\sum_{m=0}^{i} 2^m(\alpha_m + \beta_m) + C_{in} - 2^{i+1}\right\}.$$

The maximum fan-in is due to either the computation of the carry out on the second level of threshold gates or the computation of the group α_m and β_m. The fan-in required for the carry is equal to $2\lceil\frac{n}{x}\rceil + 1$. The fan-in requirements for the α_m and β_m depends on the number of bits comprising a group. It is equal to $2x$ (the bits of both operands are required to compute the α_m and β_m for any given m). Consequently, the maximum fan-in required for comparison is $MAX\{2\lceil\frac{n}{x}\rceil + 1, 2x\}$. With appropriate considerations, the maximum weight value required can be computed to be equal to $MAX\{2^{\lceil\frac{n}{x}\rceil}, 2^x\}$. Consequently, the weight sizes are minimum when $2^{\lceil\frac{n}{x}\rceil} = 2^x$ implying a partition of $\sqrt{n}$ bits per group.

Because the number of blocks has to be an integer number we have to assume for x the value $\lceil\sqrt{n}\rceil$. This leads to the maximum weight of $MAX\{2^{\left\lceil\frac{n}{\lceil\sqrt{n}\rceil}\right\rceil}, 2^{\lceil\sqrt{n}\rceil}\}$, and to a maximum fan-in of $MAX\{2\left\lceil\frac{n}{\lceil\sqrt{n}\rceil}\right\rceil + 1, 2\lceil\sqrt{n}\rceil\}$. In order to be able to assume an upper bound for the result of the MAX operator we have to establish a relation between $\left\lceil\frac{n}{\lceil\sqrt{n}\rceil}\right\rceil$ and $\lceil\sqrt{n}\rceil$. If n is a perfect square then $\left\lceil\frac{n}{\lceil\sqrt{n}\rceil}\right\rceil = \lceil\sqrt{n}\rceil$, otherwise it can be proved, based on the fact that $\lceil x\rceil \le x + 1$ holds true for any x, that $\left\lceil\frac{n}{\lceil\sqrt{n}\rceil}\right\rceil \le \lceil\sqrt{n}\rceil$ and the difference between the two numbers could not be larger than 1. Therefore the weights are at most $2^{\lceil\sqrt{n}\rceil}$ and the maximum fan-in is upper bounded by $2\lceil\sqrt{n}\rceil + 1$.

Regarding the size the first level requires $2\left\lceil\frac{n}{\lceil\sqrt{n}\rceil}\right\rceil$ threshold gates for the computations of the α_m and the β_m quantities. One threshold gate is required to compute the carry-out on the second level. Consequently, the comparison requires $2\left\lceil\frac{n}{\lceil\sqrt{n}\rceil}\right\rceil + 1$ threshold gates. □

THEOREM 2. *The 2 – 1 addition of two n-bit binary numbers can be computed by an explicit depth-3 linear threshold network with $O(\frac{n^2}{\log n})$ size and polynomially bounded weights.*

PROOF. Assume that the operands have been subdivided into groups and that each group contains at most $m \log n$ bits[4] and there are $\left\lceil\frac{n}{m\log n}\right\rceil$ groups. For each group i, $i = 0, 1, \ldots, \left\lceil\frac{n}{m\log n}\right\rceil - 1$ we compute the group carry-force α_i and the group carry-preserve β_i both direct and inverted values as:

$$\alpha_i = sgn\left\{\sum_{k=0}^{m\log n-1} 2^k(X_k + Y_k) - 2^{m\log n}\right\},$$

$$\overline{\alpha}_i = sgn\left\{2^{m\log n} - 1 - \sum_{k=0}^{m\log n-1} 2^k(X_k + Y_k)\right\},$$

$$\beta_i = sgn\left\{\sum_{k=0}^{m\log n-1} 2^k(X_k + Y_k) + 1 - 2^{m\log n}\right\},$$

$$\overline{\beta}_i = sgn\left\{2^{m\log n} - 2 - \sum_{k=0}^{m\log n-1} 2^k(X_k + Y_k)\right\}.$$

All these quantities can be computed with $4\left\lceil\frac{n}{m\log n}\right\rceil$ threshold gates with a maximum fan-in of $2m \log n$. The maximum weight is given by $2^{m\log n} = 2^{\log n^m} = n^m$ and therefore the weights are polynomially bounded. Moreover for each bit position j, $j = 1, 2, \ldots, m \log n - 1$, for a group i, we compute the bit carry-force α_i^j and the bit carry-preserve β_i^j into the bit j both direct and inverted values as:

$$\alpha_i^j = sgn\left\{\sum_{k=0}^{j-1} 2^k(X_k + Y_k) - 2^j\right\},$$

4. m is a given integer constant and we will assume for the simplicity of notations that $m \log n$ is also an integer.

$$\overline{\alpha}_i^j = sgn\left\{2^j - 1 - \sum_{k=0}^{j-1} 2^k\left(X_k + Y_k\right)\right\},$$

$$\beta_i^j = sgn\left\{\sum_{k=0}^{j-1} 2^k\left(X_k + Y_k\right) + 1 - 2^j\right\},$$

$$\overline{\beta}_i^j = sgn\left\{2^j - 2 - \sum_{k=0}^{j-1} 2^k\left(X_k + Y_k\right)\right\}.$$

For the bits in position $j = 0$ inside the group i, the bit carry-force and the bit carry-preserve are the group carry-force and respectively the group carry-preserve that correspond to the group $i - 1$. We need $4n - 4\left\lceil \frac{n}{m \log n} \right\rceil$ gates for the computation of all the bit carry-force and bit carry-preserve quantities. Obviously the fan-in and the weight requirements for bit carry-force and bit carry-preserve are less than for the group quantities. All these group and bit carry-force and carry-preserve can be computed in parallel with the expense of $4n$ gates.

Given that the carry-out C_i of the group i (see Theorem 1) can be expressed with a logic recursive formulae as

$$C_i = \alpha_i + \beta_i C_{i-1},$$
$$C_i = \alpha_i + \beta_i\alpha_{i-1} + \beta_i\beta_{i-1}\alpha_{i-2} + \ldots + \beta_i\beta_{i-1} \cdots \beta_1\alpha_0 + \beta_i\beta_{i-1} \cdots \beta_1\beta_0 C_{in}.$$

This logic expression contains $i + 2$ products each one of at most $i + 2$ logic variables. The carry-out C_j from the bit position j inside the group i can be computed with a similar expression but in this case the variables α_i and β_i have to be replaced with α_i^j and β_i^j, respectively.

In order to compute $\overline{C}_i$ we can use the following recurrence:

$$\overline{C}_i = \overline{\alpha}_i\overline{\beta}_i + \overline{\alpha}_i\overline{C}_{i-1}$$

which leads to

$$\overline{C}_i = \overline{\alpha}_i\overline{\beta}_i + \overline{\alpha}_i\overline{\alpha}_{i-1}\overline{\beta}_{i-1} + \ldots + \overline{\alpha}_i\overline{\alpha}_{i-1} \cdots \overline{\alpha}_1\overline{\beta}_1 + \overline{\alpha}_i\overline{\alpha}_{i-1} \cdots \overline{\alpha}_1\overline{\alpha}_0\overline{C}_{in}.$$

This logic expression also contains $i + 2$ products each one of at most $i + 2$ logic variables and the inverted carry-out $\overline{C}_j$ from the bit position j inside the group i can be similarly computed but with $\overline{\alpha}_i^j$ and $\overline{\beta}_i^j$ instead of $\overline{\alpha}_i$ and $\overline{\beta}_i$.

Given that the sum bit in position j is equal to

$$S_j = X_j\overline{Y}_j\overline{C}_{j-1} + \overline{X}_jY_j\overline{C}_{j-1} + \overline{X}_j\overline{Y}_jC_{j-1} + X_jY_jC_{j-1}$$

we can rewrite this expression as:

$$\begin{aligned} S_j = {} & X_j\overline{Y}_j\overline{\alpha}_i\overline{\beta}_i + X_j\overline{Y}_j\overline{\alpha}_i\overline{\alpha}_{i-1}\overline{\beta}_{i-1} + \ldots + X_j\overline{Y}_j\overline{\alpha}_i\overline{\alpha}_{i-1} \ldots \overline{\alpha}_1\overline{\beta}_1 \\ & + X_j\overline{Y}_j\overline{\alpha}_i\overline{\alpha}_{i-1} \ldots \overline{\alpha}_0\overline{C}_{in} + \overline{X}_jY_j\overline{\alpha}_i\overline{\beta}_i + \overline{X}_jY_j\overline{\alpha}_i\overline{\alpha}_{i-1}\overline{\beta}_{i-1} \\ & + \ldots + \overline{X}_jY_j\overline{\alpha}_i\overline{\alpha}_{i-1} \ldots \overline{\alpha}_1\overline{\beta}_1 + \overline{X}_jY_j\overline{\alpha}_i\overline{\alpha}_{i-1} \ldots \overline{\alpha}_0\overline{C}_{in} + \overline{X}_j\overline{Y}_j\alpha_i \\ & + \overline{X}_j\overline{Y}_j\beta_i\alpha_{i-1} + \ldots + \overline{X}_j\overline{Y}_j\beta_i\beta_{i-1} \ldots \beta_1\alpha_0 + \overline{X}_j\overline{Y}_j\beta_i\beta_{i-1} \ldots \\ & \beta_1\beta_0C_{in} + X_jY_j\alpha_i + X_jY_j\beta_i\alpha_{i-1} + \ldots + X_jY_j\beta_i\beta_{i-1} \ldots \beta_1\alpha_0 \\ & + X_jY_j\beta_i\beta_{i-1} \ldots \beta_1\beta_0C_{in} \end{aligned} \tag{2}$$

All the products in (2) can be computed in parallel in one gate delay with $4(i + 2)$ threshold gates and after that the logical OR of these products can be done with one threshold gate.

Therefore, the entire addition can be performed by a depth-3 network. In the first level we compute the group and bit carry-force and carry-preserve quantities with $4n$ threshold gates. The second level computes the products in (2). Because each bit position j in the group i needs $4(i + 2)$ products and there are $m \log n$ bit positions in each group we need $4m \log n\ (i + 2)$ threshold gates in order to compute the products that correspond to the sum bits in group i. Because i spans from 0 to $\left\lceil \frac{n}{m \log n} \right\rceil - 1$ the global number of gates in the second level is given by:

$$4m \log n \sum_{i=0}^{\left\lceil \frac{n}{m \log n} \right\rceil - 1} (i + 2) = 8m \log n \left\lceil \frac{n}{m \log n} \right\rceil + 2m \log n \left\lceil \frac{n}{m \log n} \right\rceil \left(\left\lceil \frac{n}{m \log n} \right\rceil - 1 \right) \tag{3}$$

The third level of the network contains n threshold gates, one for each bit position. Therefore the entire size of the network is in the order of $O(\frac{n^2}{\log n})$.

Because all the gates in the second level compute logical ANDs the inputs' weights are 1 and the threshold values are at most $m \log n + 4$. All the gates on the third level perform logic ORs and therefore have all the inputs' weights and the thresholds equal to 1. As a consequence the weight values are dominated by the weights associated to the gates in the first level and therefore are in the order of $O(n^m)$, i.e., polynomially bounded. □

COROLLARY 3. *The maximum fan-in for the threshold gates in the network is given by*

$$MAX\left\{2m \log n,\ 4\left(\left\lceil \frac{n}{m \log n} \right\rceil + 2\right)\right\}.$$

PROOF. The maximum fan-in is equal to $2m \log n$ for the gates in the first level. By the inclusion in the products of the bits X_j and Y_j (normal or inverted) the products in (2) contain at most $i + 4$ variables and therefore the maximum fan-in for the gates that compute the AND terms is equal with $\left\lceil \frac{n}{m \log n} \right\rceil + 4$. Because (2) contains $4(i + 2)$ terms the maximum fan-in for the gates in the third level is $4\left(\left\lceil \frac{n}{m \log n} \right\rceil + 2\right)$. Therefore, the maximum fan-in is given by $MAX\left\{2m \log n,\ 4\left(\left\lceil \frac{n}{m \log n} \right\rceil + 2\right)\right\}$. □

We proved that the $2 - 1$ addition can be performed by a depth-3 network of size in the order of $O(\frac{n^2}{\log n})$ with polynomially bounded weights. In the following we keep the same depth of the network, we impose the size to be in the order of $O(n)$ and investigate the consequences such an imposition has on weight values and fan-in. The results are stated by the following theorem.

THEOREM 3. *The $2 - 1$ addition of two n-bit binary numbers can be computed by an explicit depth-3 linear threshold network with $6n + 2\left\lceil \frac{n}{\lceil \sqrt{n} \rceil} \right\rceil$ size, with the maximum weight value of $2^{\lceil \sqrt{n} \rceil}$ and the maximum fan-in of $2\left\lceil \frac{n}{\lceil \sqrt{n} \rceil} \right\rceil + 3$.*

PROOF. Assume that the operands have been subdivided into groups and that $\oplus$ indicates the Exclusive-or. The sum, S_j, of a bit j in a group i can be computed by

$$S_j = X_j \oplus Y_j \oplus C_{j-1},$$

where $C_{j-1} = sgn\left\{\sum_{k=0}^{i} 2^k(\alpha_k + \beta_k) + C_{in} - 2^{i+1}\right\}$ is the carry into position j of group i. The sum equation can be transformed to $S_j = sgn\{1_{=}^{+} + 1_{=}^{-} + 3_{=}^{+} - 2\}$.

It will be shown that $1_{=}^{+}$, $1_{=}^{-}$, and $3_{=}^{+}$ can be computed by:

$$1_{=}^{+} = sgn\left\{2^{i+1}(X_j + Y_j) + \sum_{k=0}^{i} 2^k(\alpha_k + \beta_k) + C_{in} - 2^{i+1}\right\}$$

$$1_{=}^{-} = sgn\left\{2^{i+1} - (2^{i+1}(Y_j + Y_j) + \sum_{k=0}^{i} 2^k(\alpha_k + \beta_k) + C_{in})\right\}$$

$$3_{=}^{+} = sgn\left\{2^{i+1}(X_j + Y_j) + \sum_{k=0}^{i} 2^k(\alpha_k + \beta_k) + C_{in} - 3 \cdot 2^{i+1}\right\}.$$

Where the α_k and β_k are computed for all k, except $k = i$, using the entire group of bits and for $k = i$ the quantities α_k and β_k are computed by considering the bits r of the group i where $0 \le r \le j-1$.

Case 1. ($1_{=}^{+}$) To prove that the ($1_{=}^{+}$) expression is correct we must prove that if any of X_j, Y_j, and C_{j-1} is equal to 1 then $1_{=}^{+} = 1$ and if none of X_j, Y_j, and C_{j-1} is 1 then $1_{=}^{+} = 0$. Clearly for C_{j-1} it holds true (proven earlier) and it can be trivially proven (with substitutions) that the case holds true for X_j, Y_j values.

Case 2. ($1_{=}^{-}$) Analogous to *Case 1*.

Case 3. ($3_{=}^{+}$) Analogous to *Case 1* with proper considerations.

The equations that compute the sum require an explicit depth-3 network computing: on the first level the α_k and β_k for all groups and bits for the group i. On the second level the network computes $1_{=}^{+}$, $1_{=}^{-}$, and $3_{=}^{+}$, and finally on the third level the network computes the S_j for all j.

In order to compute the cost we divide the addition, as we did for the comparison, into groups of length x. By following the same way of reasoning as in the Corollary 2 we obtain that the optimum value of the maximum number of bits in each group is $\lceil\sqrt{n}\rceil$. This partition leads to a maximum fan-in of $2\left\lceil\frac{n}{\lceil\sqrt{n}\rceil}\right\rceil + 3$ and to a maximum weight of $2^{\lceil\sqrt{n}\rceil}$.

Under the assumption that the partition of the operands is done in groups of $\lceil\sqrt{n}\rceil$ bits the following is required regarding the size of the network. In order to compute the group α_i and β_i it is required to have $2\left\lceil\frac{n}{\lceil\sqrt{n}\rceil}\right\rceil$ threshold gates in the first level. Further we require at most $2n$ threshold gates in the first level to compute all bit α_k and β_k. On the second level we require $3n$ gates to compute $1_{=}^{+}$, $1_{=}^{-}$, and $3_{=}^{+}$ and finally we require n gates on the third level to compute the sum S_j for all j. Thus the entire scheme requires at most $6n + 2\left\lceil\frac{n}{\lceil\sqrt{n}\rceil}\right\rceil$ threshold gates to compute the sum. □

REMARK 1. As a consequence of the fact that sum equation in Theorem 3 can be rewritten as $S_j = 1_{=}^{+} + 1_{=}^{-} + 3_{=}^{+} - 1$, the sum of two n-bit binary numbers can be computed by an implicit depth-2 linear threshold network with $5n + 2\left\lceil\frac{n}{\lceil\sqrt{n}\rceil}\right\rceil$ size, i.e., in the order of $O(n)$. This scheme will increase the fan-in for the next network that uses as input the computed sum by 2, because the value of each sum bit is carried by 3 signals instead of 1.

3 COMPARISONS

In the previous discussion we have determined the network requirements in general. Our scheme for addition presented in Theorem 2 provides polynomially bounded weights and a network size in the order of $O(\frac{n^2}{\log n})$ and it is superior to the scheme presented in [4] which has an $O(n^2)$ size. Consequently, we imposed an optimal $O(n)$ size for depth-3 networks for the addition and investigated the influence such an imposition had on the weight sizes and fan-in. It was established that the weight requirements are of $O(2^{\sqrt{n}})$ complexity and that the fan-in requirements are of $O(\sqrt{n})$ complexity. Consequently, we have proposed depth-2 networks for the comparison that have $O(\sqrt{n})$, $O(2^{\sqrt{n}})$, and $O(\sqrt{n})$ complexity for size, weight, and fan-in respectively. Our investigation leaves open the questions of $O(n)$ size depth-3 addition and $O(\sqrt{n})$ depth-2 comparison with polynomially bounded weights.

Given that asymptotic complexities need not apply to realistic scenarios we considered as a final exercise a comparison with other schemes assuming existing architectural formats. In particular we considered 32- and 64-bit architectures and estimated the requirements of the various schemes. The results of our estimations are reported in Table 1 and Table 2. For the evaluation of Siu and al and Roychowdhury and al schemes performance we used the formulas reported in [4], [11]. The PW Addition row corresponds to the addition scheme presented in Theorem 2 for $m = 1$ i.e., the division of operands in groups of $\log n$ bits. For the other rows we assumed that the subdivision of the operands is made using $\lceil\sqrt{n}\rceil$. The depth-3 comparison is done by first dividing the operands in two and after that in $\left\lceil\sqrt{\frac{n}{2}}\right\rceil$. The first level computes the carry-force and carry-preserve for all the groups of $\left\lceil\sqrt{\frac{n}{2}}\right\rceil$ bits. The second level produces the carry out of the least significant $\frac{n}{2}$ bits and the third level the result of the comparison

What is noticeable from the tables is the small amount of linear threshold gates to realize the addition-comparison for the common 32- and 64-bit operand sizes. Clearly, the improvement for the size over existing art is substantial. In particular our addition scheme with polynomially bounded weights requires up to 71% for the realization of 32-bit adders and up to 47% for the realization of 64-bit devices when compared to the Siu et al. scheme [4]. The fan-in reduction is also significant because our scheme requires up to 28% for the realization of 32-bit adders and up to 18% for the realization of 64-bit devices. As the tables suggest the scheme proposed in Theorem 3 can be realized with a very small fraction of gates for the 32- and 64-bit operand sizes. In particular, it requires up to 18% for the realization of 32-bit adders and up to 9.32% for the realization of 64-bit devices when compared to the Siu et al. scheme [4]. While, as it can be observed in Table 2 for 32-bit operands, our scheme implies a maximum weight value twice the maximum weight value deduced from [4], it provides an 8.5 times lower fan-in.

TABLE 1
COMPARISONS FOR SOME OPERAND LENGTHS

	Function	Depth	32-bit operands			64-bit operands		
			S	W	F	S	W	F
Proposed Schemes	Comparison	2	13	64	13	17	256	17
	Comparison	3	18	16	9	26	64	13
	PW Addition	3	796	32	36	2040	64	48
	EW Addition	3	204	64	15	400	256	19
Siu and al	Comparison	3	96	32	32	192	64	64
	Addition	3	1121	32	128	4289	64	256
Roychowdhury and al	Comparison	3	19	64	12	31	128	14

TABLE 2
RATIO BETWEEN SCHEMES

	Function	32-bit operands			64-bit operands		
		S	W	F	S	W	F
Ratios vs Siu and al	Comparison in depth 2	0.13	2	0.40	0.08	4	0.26
	Comparison in depth 3	0.18	0.50	0.28	0.13	1	0.20
	PW Addition	0.71	1	0.28	0.47	1	0.18
	EW Addition	0.18	2	0.11	0.09	4	0.07
Ratios vs Roychowdhury and al	Comparison in depth 2	0.68	1	1.08	0.54	2	1.21
	Comparison in depth 3	0.94	0.25	0.75	0.83	0.50	0.92

Regarding the comparison when we consider a depth-2 network, as expected, the weights requirements are greater for our scheme when compared to [4], [11] and superior in size. This conclusion however is reversed when the depths of the network are assumed to be equal. Our estimations indicate that the scheme we propose is better in all counts including the size of the weights (at the exception of 64 operands which the weight size of the Siu et al. scheme are equal to ours). In particular, our depth-3 scheme requires up to 18% gates, 50% weights and 28% fan-in for the realization of 32-bit comparators and up to 13% gates, equal weights and 20% fan-in for the realization of 64-bit devices, when compared to the Siu et al. scheme [4]. When compared with Roychowdhury and al scheme [11] it requires up to 94% gates, 25% weights and 75% fan-in for the realization of 32-bit comparators and up to 83% gates, 50% weights and 92% fan-in for the realization of 64-bit devices.

4 CONCLUDING REMARKS

The main concern of this paper was the reduction of the size of networks computing fixed point arithmetic operations while maintaining small network depths with bounded and unbounded weights. It was shown that the addition can be performed by a depth-3 network with: the size in the order of $O(\frac{n^2}{\log n})$ and polynomially bounded weights; with the size of $6n + 2\left\lceil \frac{n}{\lceil \sqrt{n} \rceil} \right\rceil$ (i.e., of optimal $O(n)$ complexity), a maximum fan-in of $2\left\lceil \frac{n}{\lceil \sqrt{n} \rceil} \right\rceil + 3$ and a maximum weight value of $2^{\lceil \sqrt{n} \rceil}$. Related to comparison it was shown that the comparison of two n-bit operands with carry can be computed by a depth-2 network with $2\left\lceil \frac{n}{\lceil \sqrt{n} \rceil} \right\rceil + 1$ size (i.e., of $O(\sqrt{n})$ complexity), a maximum fan-in of $MAX\{2\left\lceil \frac{n}{\lceil \sqrt{n} \rceil} \right\rceil + 1, 2\lceil \sqrt{n} \rceil\}$ and a maximum weight size of $2^{\lceil \sqrt{n} \rceil}$. The open questions left by the investigation are optimal $O(n)$ size depth-3 addition and $O(\sqrt{n})$ depth-2 comparison with potentially polynomially bounded weights.

REFERENCES

[1] S. Muroga, "The Principle of Majority Decision Elements and the Complexity of their Circuits," *Proc. Int'l Conf. Information Processing*, pp. 400-407, UNESCO House, Paris, June 1959.

[2] R. Minnick, "Linear-Input Logic," *IRE Trans. Electronic Computers*, vol. 10, pp. 6-16, Mar. 1961.

[3] S. Muroga, "Threshold Logic and its Applications," John Wiley and Sons, 1971.

[4] K. Siu, V. Roychowdhury, and T. Kailath, "Depth-Size Tradeoffs for Neural Computation," IEEE *Trans. Computers*, vol. 40, no. 12, pp. 1,402-1,412, Dec. 1991.

[5] N. Alon and J. Bruck, "Explicit Construction of Depth-2 Majority Circuits for Comparison and Addition," Technical Report RJ 8300 (75661), IBM Research Division, Aug. 1991.

[6] S. Cotofana and S. Vassiliadis, "Periodic Symmetric Functions with Feed-Forward Neural Networks," *NEURAP '95/'96 Neural Networks and their Applications*, pp. 215-221, Marseille, Mar. 1996.

[7] T. Shibata and T. Ohmi, "A Functional MOS Transistor Featuring Gate-Level = Weighted Sum and Threshold Operations," *IEEE Trans. Electron Devices*, vol. 39, pp. 1,444-1,455, June 1992.

[8] T. Shibata and T. Ohmi, "Neuron MOS Binary-Logic Integrated Circuits—Part I: Design Fundamantals for Soft-Hardware Circuit Implementation," *IEEE Trans. Electron Devices*, vol. 40, pp. 570-575, Mar. 1993.

[9] T. Shibata and T. Ohmi, "Neuron MOS Binary-Logic Integrated Circuits—Part II: Simplifying Techniques of Circuit Configration and their Practical Applications," *IEEE Trans. Electron Devices*, vol. 40, pp. 974-979, May 1993.

[10] K. Siu and J. Bruck, "On the Power of Threshold Circuits with Small Weights," *SIAM J. Discrete Math.*, pp. 423-435, Aug. 1991.

[11] V. Roychowdhury, A. Orlitsky, and K. Y. Siu, "Lower Bounds on Threshold and Related Circuits via Communication Complexity," *IEEE Trans. Information Theory*, vol. 40, no. 2, pp. 467-474, Mar. 1994.