Efficient Hamming Weight Comparators for Binary Vectors Based on Accumulative and Up/Down Parallel Counters

Behrooz Parhami, *Fellow, IEEE*

***Abstract*—New counting-based methods for comparing the Hamming weight of a binary vector with a constant, as well as comparing the Hamming weights of two input vectors, are proposed. It is shown that the proposed comparators are faster and simpler, both in asymptotic sense and for moderate vector lengths, compared with the best available fully digital designs. These speed and cost advantages result from a more efficient population counting, as well as the merger of counting and comparison operations, via accumulative and up/down parallel counters.**

***Index Terms*—Column compression, comparator, Hamming distance, multioperand addition, parallel counter, population count.**

I. Introduction

THE HAMMING weight of the binary vector $V = \{v_1, v_2, \ldots, v_n\}$ is a number that ranges from 0 to n, which is defined as $H(V) = \Sigma_{1 \le i \le n} v_i$. Certain applications require that $H(V)$, that is, the number of 1s in the vector V, be compared with a fixed threshold k or with $H(U)$, where $U = \{u_1, u_2, \ldots, u_m\}$ is another binary vector of arbitrary length m. Thus, the problems of interest here are determining whether $H(V) \ge k$ or $H(V) \ge H(U)$. Such applications abound, for example, in neural networks [1], threshold voting circuits [2], digital filtering [3], [4], pattern matching/recognition [5]–[7], and encoding for data compression [8], [9].

In this paper, we present new designs for Hamming weight comparators that are faster [$O(\log n)$ versus $O(\log^2 n)$], as well as less complex [$O(n)$ versus $O(n \log^2 n)$], than the best previously published fully digital designs for n-bit inputs. These advantages result from two improvements: a more efficient population counting (determining how many 1s there are among the inputs) by means of parallel counters, and a merger or overlapping of population counting with the comparison stage. Following a brief review of previous work in Section II, our designs are presented via separate discussions of the required parts (Sections III and V) and their use in our proposed new designs (Sections IV and VI). Speed and cost comparisons with the existing fully digital designs appear in Section VII. Section VIII concludes this paper.

Manuscript received December 20, 2007; revised April 8, 2008. First published February 10, 2009; current version published February 25, 2009. This paper was recommended by Associate Editor V. Gaudet.

The author is with the Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA 93106-9560 USA (e-mail: parhami@ece.ucsb.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSII.2008.2010176

II. Review of Previous Work

In view of the extensive applications cited in Section I, many researchers have offered designs for Hamming weight comparators, focusing on the optimization of speed, cost (very large scale integration (VLSI) area), and power consumption. These existing designs fall into two categories of mixed digital/analog and fully digital. Our focus here is on the latter category, but we provide a brief review of some mixed digital/analog designs for completeness. Clearly, digital designers must consider both classes of designs before making a choice for a particular application domain.

Fujino and Moshnyaga [10] offer a mixed digital/analog design based on two parallel circuits, each delaying a common signal by an amount proportional to the number of 1s in an n-bit input. Depending on which signal path wins the race, an output circuit is accordingly set to denote the comparison result between two input vectors or an input vector and a fixed threshold. Fujino and Moshnyaga contrast their design to those of Asada *et al.* [5], Yamashina *et al.* [11], Gurkaynak *et al.* [12], and Morie *et al.* [13]. The comparison shows the area and power advantages of 8-bit designs relative to those in [5] and [12], and the latency advantage over that in [5].

Most of these mixed analog/digital designs are said to outperform "conventional" designs based on the counting of the number of 1s, although the exact counting-based designs are not specified. It is also rather unclear how scalable the mixed digital/analog designs would be in the face of increasing process variations as we move up to higher values of n and deeper into submicrometer regimes [14].

In the realm of fully digital designs, King *et al.* [15] and Pedroni [16] have proposed Hamming weight comparators with $O(n)$ delay (in terms of logic gate levels) and $O(n^2)$ complexity (which is measured in number of gates). Piestrak [17] has recently improved on these results by showing that threshold functions, which are realized by means of bit-sorting networks [18], lead to Hamming weight comparator designs with $O(\log^2 n)$ delay and $O(n \log^2 n)$ complexity. The improvements reported by Piestrak are direct consequences of using sorting networks of lower time and cost complexities, given that sorting a pair of bits only requires an OR gate, to obtain the larger of the two bit values, and an AND gate, yielding the smaller of the two (see Section VI for further elaboration).

Pedroni's design is better than that of King *et al.*, so we will not discuss the latter any further. However, although Piestrak's designs supersede those of Pedroni, in the sense of being

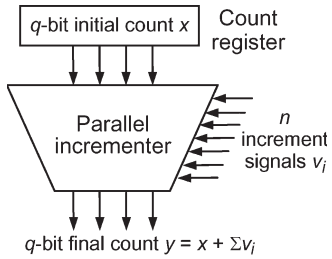

Fig. 1. Block diagram of an accumulative parallel counter.

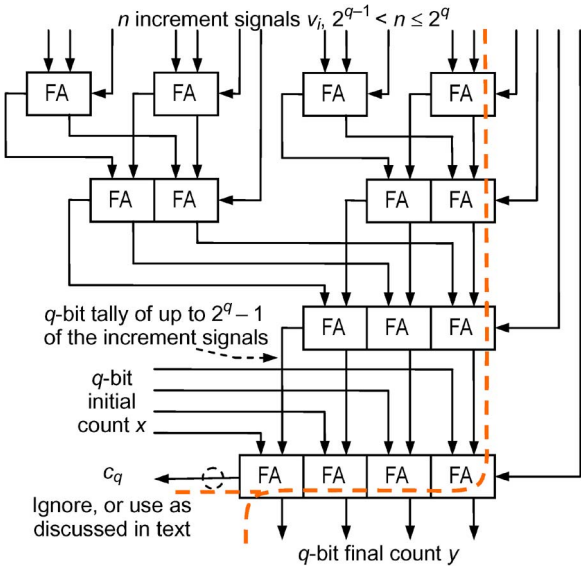

Fig. 2. Design of an unsigned mod-2^q parallel incrementer.

uniformly better in both speed and cost, we will compare our designs to both schemes to better point out the relationships of the two schemes and the progression of speed and cost improvements.

III. Accumulative Parallel Counters

Parallel counters [19], that is, circuits that count the number of 1s in an input vector, were proposed to generalize sequential counters. A more appropriate generalization would be a circuit that adds the sum of n increment signals to a stored count that is q bits wide, as shown in Fig. 1. When there is only one increment signal, the circuit degenerates into a conventional sequential up counter.

The parallel incrementer in Fig. 1 is a generalized form of a conventional parallel counter in the sense of including the extra count input entering it from the top. Designs for the parallel incrementers have been studied for arbitrary values of q [20].

For our domain of interest here, we can set $q = \lceil \log_2 n \rceil$, because the threshold k, which is smaller than n, is represented with at most $\lceil \log_2 n \rceil$ bits. Fig. 2 shows a parallel incrementer based on the foregoing assumptions. The population count part of the design consists of a tree of increasingly wider ripple-carry adders from top to bottom. It is readily seen that with $q-1$ levels of such adders, imposing a total latency of $2q-2$ levels, the q-bit sum of up to $2^q - 1$ increment signals is obtained. The final q-bit ripple-carry adder incorporates the initial count and perhaps the last increment signal. It adds a

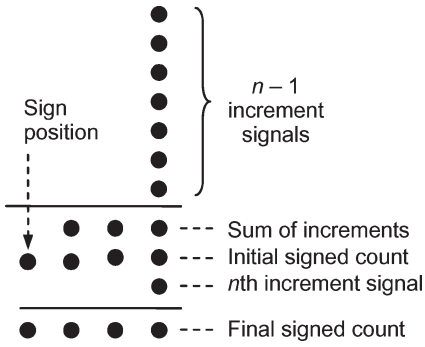

Fig. 3. Signed parallel incrementer in dot notation.

latency of only one level given that its carry propagation almost totally overlaps that of the $(q-1)$-bit adder directly above it. The total latency is thus $2q-1$ full-adder levels (see the dashed line).

Note that in terms of the number $n = 2^q - 1$ of input signals, the latency of the circuit in Fig. 2 is approximately $2 \log_2 n$. Although it might be possible to improve this latency to slightly over $\log_{1.5} n \cong 1.7 \log_2 n$ via the application of the Wallace-tree construction used in high-speed multipliers [21], very little, if any, performance would be gained due to the peculiar structure of such trees, which necessitates longer and more irregular wires. So, we proceed with the simple design in Fig. 2.

IV. Fixed-Threshold Comparators

A fixed-threshold Hamming weight comparator is obtained if the 2's-complement of k, that is, a representation of $-k$, is placed in the count register of Fig. 1, the bits of the vector V are supplied as increment signals from the right, and the complement of the result's sign bit is taken as the output. This is tantamount to checking the condition $-k + \Sigma_{1 \leq i \leq n} v_i \geq 0$.

The latter condition is equivalent to having a nonnegative final count in an accumulative parallel counter with initial count $-k$ and n increment signals v_i, $1 \leq i \leq n$. As shown in the dot-notation diagram in Fig. 3 for $n = 8$, this multioperand addition problem [21] can readily be solved by adding $n-1$ of the increment signals to form a $\lceil \log_2 n \rceil$-bit sum. The latter sum is then added to the $(q+1)$-bit initial count by using the nth increment signal as the carry-in into the addition.

A direct circuit realization of the method in Fig. 3 is slightly different from that in Fig. 2. We could modify the design by including an extra half-adder (in the location marked by a dashed circle near the bottom of Fig. 2) and extending the initial count to $q+1$ bits so as to accommodate a sign position in 2's-complement format. Then, the complement of the sum output of this extra half-adder indicates a nonnegative final count. However, if we start with a negative initial count and would like to know whether the updated count is nonnegative, the carry-out of the q-bit ripple-carry adder at the bottom of Fig. 2 directly supplies this information. This is because the sign bit 1 in the extra position to the left would change to 0, thus indicating a nonnegative result iff the carry-out bit c_q is 1. Put another way, instead of initializing to $-k$ and determining whether adding n increment signals would make it nonnegative, we can initialize an unsigned parallel counter to $2^q - k$ and determine whether

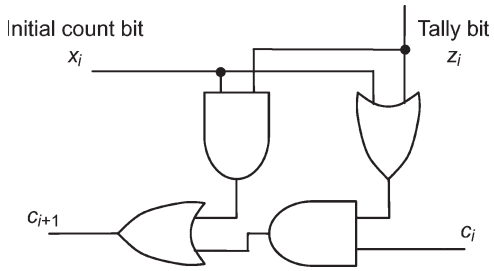

Fig. 4. Stage of carry network in lieu of the final adder.

adding the n increment signals would make the unsigned sum equal to or greater than 2^q (leads to overflow).

Because we are only interested in the carry-out bit c_q of the final addition, those parts of the parallel incrementer that are only used to generate its other q outputs can be removed to simplify the design. This is an output-side simplification. Furthermore, for any fixed threshold k, the circuit can be simplified by removing the unnecessary elements; these input-side simplifications are discussed later. Alternatively, we could forego the input-side simplifications to allow the use of a variable (adjustable) threshold that is externally supplied (set at system startup). This approach would make our design more flexible than those offered earlier. These more versatile designs are still faster than those previously published (see Section VII).

A critical path through the circuit realization in Fig. 2 is shown by the heavy dashed line. It consists of $2q-1$ full-adder levels. Using a more precise analysis, by distinguishing between the full-adder latency for its sum and carry outputs, $\delta_{\rm sum}$ and $\delta_{\rm carry}$, respectively, the critical path shown has a latency of $(q-1)\delta_{\rm sum}+q\delta_{\rm carry}$, assuming we focus on the c_q output, as discussed above. The output-side simplifications consist of removing the sum logic for the q full-adders at the bottom of Fig. 2, leaving in place only the carry network depicted in Fig. 4. This reduces the circuit complexity, but has no effect on the latency previously computed.

The carry network in Fig. 4 can also be used to assess the extent of input-side simplifications. In positions where the initial count bit is $x_i=0$, we have $c_{i+1}=z_ic_i$, which requires a single AND gate per stage. For $x_i=1$, the simplified stage will compute $c_{i+1}=c_i\vee z_i$ by using a single OR gate. Thus, the q full-adders and the associated delays at the bottom of Fig. 2 are replaceable with q gates and q gate delays. The critical path in Fig. 2 now goes through the full-adders on the next to the last row for a total latency of $(q-1)(\delta_{\rm sum}+\delta_{\rm carry})+\delta_{\rm gate}$.

V. Up/Down Parallel Counters

An up/down parallel counter [22] is a generalization of a sequential up/down counter. The latter receives a "count" signal along with a "direction" signal and accordingly updates the stored count ($+1$ for up, -1 for down, if count is asserted). One possible generalization involves using a single direction signal along with n parallel count signals. More useful for our purposes here is a generalization that views the inputs as signed count signals (increments and decrements), each encoded in 2 bits. The stored count is then updated by adding to it the sum of the signed count signals, that is, $y=x+\Sigma w_i$, where w_i's

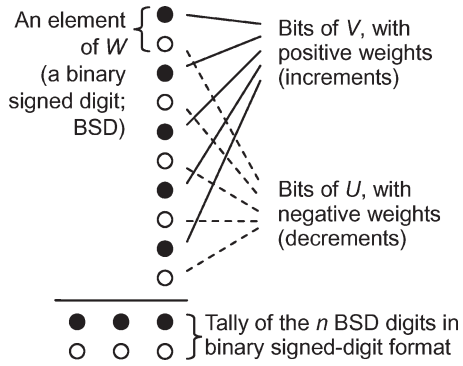

Fig. 5. Up/down parallel counting shown in extended dot notation.

are now the signed values in $\{-1,0,1\}$ encoded in 2 bits. With the latter view, the block diagram for such a counter would be identical to Fig. 1, except that each of the lines entering from the right would represent a 2-bit bundle.

The count values -1, 0, and 1 can be encoded in different ways by using 2 bits. An encoding that has been found to be quite efficient in the past is the (ν,π) encoding, which uses 10, 00, and 01 to represent -1, 0, and 1, respectively. The name "(ν,π) encoding" arises from the interpretation of the two bits as negative and positive flags [23]. We use here a simple variation of the (ν,π) encoding, which we call (ν',π) encoding, where the ν bit is inverted. This rather trivial change has been shown to produce significant savings [24] in cost and delay (and to a lesser extent, power consumption) of circuits that process binary signed digits (BSDs) in $\{-1,0,1\}$. These savings result from the avoidance of multiple inversions in the course of computation.

The realization of an up/down parallel counter can be based on the process depicted in Fig. 5, where an extended dot notation is employed. In this notation, the black dots represent the ordinary bits, or *posibits*, whereas the hollow circles stand for the negatively weighted bits, or *negabits*. A pair consisting of a posibit and a negabit form a signed digit in $\{-1,0,1\}$. It is readily shown in Fig. 5 that the sum of the BSD inputs with three-valued digits can be formed as a BSD number by using two separate tally circuits: one for the posibits to form the π bits of the BSD output, and another for the negabits to form the output's ν bits. All that remains is to detect the sign of the BSD result. This is done by a fast sign detection circuit [25], or as outlined below.

Fig. 6 shows a realization of an up/down parallel counter based on Fig. 5 and the discussion in the preceding paragraph. Each of the parallel counter blocks is identical to the design in Fig. 2 with its final row of full-adders removed. The final row of adders does the required 2's-complement subtraction. Note that the inverted encoding of the negabits obviates the need for complementation, and the carry-in of 1 supplies the $+1$ term needed for 2's complementation. It is clear from a comparison of Figs. 2 and 6 that the latency formula derived at the very end of Section IV is applicable to both.

VI. Two-Vector Comparators

In what follows, we shall assume that $m=n$, limiting our design to comparing the Hamming weights of vectors of

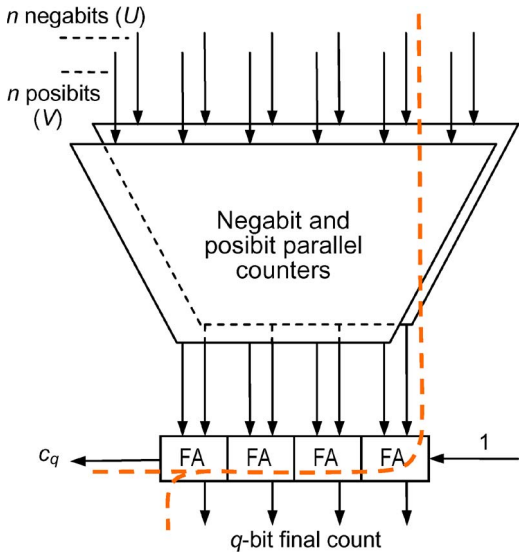

Fig. 6. Realization of an up/down parallel counter.

the same length. For $m \neq n$, all the derived time and cost complexity results become pessimistic upper bounds if n is taken to be the larger of the two values. This is tantamount to 0-extending the shorter of the two input vectors to obtain $V = \{v_1, v_2, \ldots, v_n\}$ and $U = \{u_1, u_2, \ldots, u_n\}$ as our comparands.

Defining the vector $W = V - U$, which is computed bitwise, the condition $H(V) \geq H(U)$ is equivalent to $\Sigma_{1 \leq i \leq n} w_i \geq 0$. The latter sum is the "weight" of W, with each element having the weight -1, 0, or $+1$. Thus, if we use bits of V as posibit inputs to the circuit in Fig. 6 and form the complements of the bits of U for use as its negabit inputs, the carry-out c_q yields the desired comparison result.

As noted earlier, the latency of the circuit in Fig. 6 is identical to that in Fig. 2; the circuit complexity, in terms of the number of full-adders, is $n - q$ greater. For example, when $n = m = 15$ $(q = 4)$, there are 11 extra full-adders in the circuit in Fig. 6 compared with that in Fig. 2.

VII. Speed and Cost Comparisons

We now compare our designs with those of Pedroni [16] and Piestrak [17] in terms of latency (number of gate levels) and cost (gate count). To facilitate our comparison, we note that these previous designs, although quite different at first sight, are instances of the same design method, with Piestrak's design being simpler and faster. To explain our unified formulation of the two design methods, we use the example of determining whether $H(V) \geq 3$ for a 5-bit vector V. Piestrak's design [17], which is shown in Fig. 7(a), is based on a sorting network that orders the input bits from largest to smallest. The sorted output vector starts with a number h of 1s (perhaps none) followed by $5 - h$ elements of 0. For $k = 3$, we simply examine the third output line. Each of the short vertical line segments in the diagram represents an OR gate and an AND gate, collectively sorting a pair of bits, with the larger bit output at the top.

Pedroni's design [16], which is formulated in terms of a rectangular cellular array that is formed of a building block with two NOR gates, can be recast in terms of a sorting network by complementing the horizontal signal into the cell [16, Fig. 2]

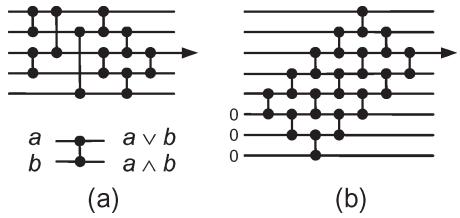

Fig. 7. Unified view of Piestrak's and Pedroni's designs. (a) Piestrak's design for $H(V) \geq 3$. (b) Pedroni's design for $H(V) \geq 3$.

TABLE I
Cost and Latency Expressions for Fixed-Threshold Hamming Weight Comparators

Design	Cost (gate count)	Latency (gate levels)
Pedroni [16]	$2kn$	$n + k - 1$
Piestrak [17]	$n(\log_2 n)^2/2$	$\log_2 n(\log_2 n + 1)/2$
Proposed	$(n - \log_2 n - 1)\gamma_{FA} + \log_2 n$	$(\log_2 n - 1)(\delta_{sum} + \delta_{carry}) + 1$

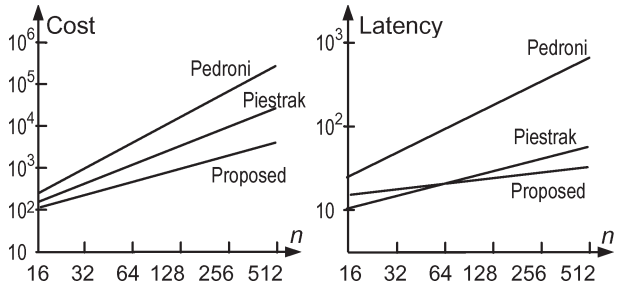

Fig. 8. Cost and latency comparisons.

and replacing the cell with the two-input sorter shown in Fig. 7. With this interpretation, Pedroni's design for $H(V) \geq 3$ would look as in Fig. 7(b). The three extra 0 inputs ensure that the third output from the top would equal 1 iff the input vector V has at least three 1s. The original design presented by Pedroni actually outputs the logical AND of the top three outputs, whereas the third output provides the same information given that the top two outputs cannot be 0s when the third one is 1.

Piestrak's fixed-threshold Hamming weight comparator is built of $O(n \log^2 n)$ gates and has a latency of $O(\log^2 n)$ gate levels. Note that the speed and cost of these networks are independent of the threshold k. The Pedroni fixed-threshold Hamming weight comparator requires $O(kn)$ gates and has a latency of $O(n + k)$ gate levels. Our design, which is depicted in Fig. 2, requires $O(n)$ gates and has a latency of $O(\log n)$ gate levels. The exact cost (number of gates) and latency (gate level) formulas for the three designs are listed in Table I, where γ_{FA} is the cost of a full-adder relative to a gate, and δ_{sum} and δ_{carry} are the full-adder delay parameters.

From the expressions in Table I, we find that our design is faster than Piestrak's when $\delta_{sum} + \delta_{carry} < 1 + (\log_2 n)/2$. Thus, taking $\delta_{sum} = \delta_{carry} = 2$, we find that our design would have a speed advantage for $n > 64$. Likewise, our design is simpler when $\gamma_{FA} < \log_2 n[n(\log_2 n)/2 - 1]/(n - \log_2 n - 1)$. A standard full-adder needs nine gates; thus, our design is more economical for $n > 8$. Fig. 8 shows the preceding cost/latency comparisons in graphic format. Note that we have taken Pedroni's design in its original form [16], whereas it is readily

TABLE II
COST AND LATENCY EXPRESSIONS FOR TWO-VECTOR HAMMING WEIGHT COMPARATORS

Design	Cost (gate count)	Latency (gate levels)
Pedroni [16]	$n(n+1)$	$2n$
Piestrak [17]	$n(\log_2 n)^2 + n + \log_2 n$	$(\log_2 n+1)(\log_2 n+2)/2$
Proposed	$2(n-\log_2 n-1)\gamma_{FA}+4\log_2 n$	$(\log_2 n-1)(\delta_{sum}+\delta_{carry})+1$

shown in Fig. 7(b) that all the comparators connected to the lowest k signal lines can be removed with no ill effect. However, this simplification does not change the asymptotic cost and latency formulas.

Of course, with modern implementation technologies, a gate-level comparison is unrealistic and, possibly, also unfair. Full-adders are such important components of arithmetic circuits that their designs have extensively been optimized with regard to cost, latency, and energy requirements. Scores of designs are available for use with different technological constraints, performance needs, and power characteristics. For example, if we use a ten-transistor full-adder [26] in our design, it might become superior to that of Piestrak [17] for smaller values of n as well. Thus, the data in Table I and Fig. 8 must be viewed as conservative for our designs. Note, however, that transistor-level circuit optimizations are also applicable to competing designs, although the resultant savings are unlikely to be as significant as for full-adders.

The corresponding cost and latency expressions for the two-vector Hamming weight comparators appear in Table II. It is readily shown from comparing Tables I and II that, compared with the state of the art, our proposed scheme offers even greater benefits for two-vector comparators.

It is also possible to compare the three designs in Tables I and II based on the product of cost and latency, which represents cost effectiveness. In such a comparison, our proposed designs are uniformly better, even for small values of n.

VIII. CONCLUSION

We have proposed a design method for Hamming weight comparators that allow the weight of a vector to be compared to a given threshold value. The constant threshold may be known *a priori* (fixed) or supplied as input (adjustable). Even when a completely variable threshold is not a requirement, it may be useful to have the facility to adjust the threshold based on system requirements or field experience. We have also applied our design strategy to circuits that determine which of the two vectors has a larger Hamming weight. Our designs are faster [$O(\log n)$ versus $O(\log^2 n)$] as well as less complex [$O(n)$ versus $O(n \log^2 n)$] than the best previously published results, due to Piestrak [17], for fully digital realizations.

Hamming weight comparators find applications in digital neural networks, threshold voting, digital filtering, pattern recognition/matching, and data compression. Further optimizing our designs and performing detailed cost/speed analysis based on actual circuit implementations for the named applications constitute possible directions for future work.

REFERENCES

[1] D. B. S. King, R. J. Simpson, R. J. Moore, and I. P. MacDiamid, "Hamming value comparator hierarchies," *Electron. Lett.*, vol. 35, no. 11, pp. 910–911, May 1999.

[2] B. Parhami, "Voting networks," *IEEE Trans. Rel.*, vol. 40, no. 3, pp. 380–394, Aug. 1991.

[3] K. Chen, "Bit-serial realizations of a class of nonlinear filters based on positive Boolean functions," *IEEE Trans. Circuits Syst.*, vol. 36, no. 6, pp. 785–794, Jun. 1989.

[4] M. Karaman, L. Onural, and A. Atalar, "Design and implementation of a general-purpose median filter unit in CMOS VLSI," *IEEE J. Solid-State Circuits*, vol. 25, no. 2, pp. 505–513, Apr. 1990.

[5] K. Asada, S. Kumatsu, and M. Ikeda, "Associative memory with minimum hamming distance detector and its application to bus data encoding," in *Proc. IEEE Asia-Pacific Appl.-Specific Integr. Circuits Conf.*, 1999, p. 16.1.

[6] C. Barral, J.-S. Coron, and D. Naccache, "Externalized fingerprint matching," in *Proc. 1st Int. Conf. Biometric Authentication*, 2004, pp. 309–315.

[7] S. Ullman, *High-Level Vision: Object Recognition and Visual Cognition*. Cambridge, MA: MIT Press, 1996, p. 5.

[8] S. Komatsu, M. Ikeda, and K. Asada, "Bus data encoding with coupling-driven adaptive code-book method for low power data transmission," in *Proc. 27th Eur. Solid-State Circuits Conf.*, 2001, pp. 297–300.

[9] M. Stan and W. Burleson, "Bus-invert coding for low-power I/O," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 3, no. 1, pp. 49–58, Mar. 1995.

[10] M. Fujino and V. G. Moshnyaga, "An efficient hamming distance comparator for low-power applications," in *Proc. 9th Int. Conf. Electron., Circuits Syst.*, 2002, vol. 2, pp. 641–644.

[11] T. Yamashina, T. Shibata, and T. Ohmi, "Neuron MOS winner-take-all circuit and its application to associative memory," in *Proc. Int. Solid-State Circuits Conf.*, 1993, pp. 236–237.

[12] F. K. Gurkaynak, Y. Leblebici, and D. Mlynek, "A compact high-speed hamming distance comparator for pattern matching applications," in *Proc. 9th Eur. Signal Process. Conf.*, 1998, pp. 1–9.

[13] T. Morie, T. Matsuura, S. Miyata, T. Yamanaka, M. Nagata, and A. Iwata, "Quantum dot structures measuring hamming distance for associative memories," *Superlattices Microstruct.*, vol. 27, no. 5/6, pp. 613–616, May 2000.

[14] O. S. Unsal, J. W. Tschanz, K. Bowman, V. De, X. Vera, A. Gonzalez, and O. Ergin, "Impact of parameter variations on circuits and microarchitecture," *IEEE Micro*, vol. 26, no. 6, pp. 30–39, Nov./Dec. 2006.

[15] D. B. S. King, R. J. Simpson, C. Moore, and I. P. MacDiarmid, "Digital n-tuple hamming comparators for weightless systems," *Electron. Lett.*, vol. 34, no. 22, pp. 2103–2104, Oct. 1998.

[16] V. A. Pedroni, "Compact fixed-threshold and two-vector hamming comparators," *Electron. Lett.*, vol. 39, no. 24, pp. 1705–1706, Nov. 2003.

[17] S. J. Piestrak, "Efficient hamming weight comparators of binary vectors," *Electron. Lett.*, vol. 43, no. 11, pp. 611–612, May 2007.

[18] B. Parhami, *Introduction to Parallel Processing: Algorithms and Architectures*. New York: Plenum, 1999, ch. 7.

[19] E. E. Swartzlander, "Parallel counters," *IEEE Trans. Comput.*, vol. C-22, no. 11, pp. 1021–1024, Nov. 1973.

[20] B. Parhami and C.-H. Yeh, "Accumulative parallel counters," in *Proc. Asilomar Conf. Signals, Syst., Comput.*, 1995, pp. 966–970.

[21] B. Parhami, *Computer Arithmetic: Algorithms and Hardware Designs*. New York: Oxford, 2000, ch. 8.

[22] B. Parhami, "Parallel counters for signed binary signals," in *Proc. Asilomar Conf. Signals, Syst., Comput.*, 1989, pp. 513–516.

[23] B. Parhami, "Generalized signed-digit number systems: A unifying framework for redundant number representations," *IEEE Trans. Comput.*, vol. 39, no. 1, pp. 89–98, Jan. 1990.

[24] G. Jaberipur, B. Parhami, and M. Ghodsi, "Weighted two-valued digit-set encodings: Unifying efficient hardware representation schemes for redundant number systems," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 52, no. 7, pp. 1348–1357, Jul. 2005.

[25] T. Srikanthan, S. K. Lam, and M. Suman, "Area–time efficient sign detection technique for binary signed-digit number system," *IEEE Trans. Comput.*, vol. 53, no. 1, pp. 69–72, Jan. 2004.

[26] J.-F. Lin, Y.-T. Hwang, M.-H. Sheu, and C.-C. Ho, "A novel high-speed and energy efficient 10-transistor full adder design," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 54, no. 5, pp. 1050–1059, May 2007.