Low-Complexity Tree Architecture for Finding the First Two Minima

Youngjoo Lee, *Member, IEEE*, Bongjin Kim, *Student Member, IEEE*, Jaehwan Jung, *Student Member, IEEE*, and In-Cheol Park, *Senior Member, IEEE*

***Abstract*—This brief presents an area-efficient tree architecture for finding the first two minima as well as the index of the first minimum, which is essential in the design of a low-density parity-check decoder based on the min–sum algorithm. The proposed architecture reduces the number of comparators by reusing the intermediate comparison results computed for the first minimum in order to collect the candidates of the second minimum. As a result, the proposed tree architecture improves the area–time complexity remarkably.**

***Index Terms*—Area-efficient design, digital integrated circuits, low-density parity-check (LDPC) codes, minimum value generation, tree structure.**

I. Introduction

DUE to the powerful error-correcting capability, low-density parity-check (LDPC) codes have widely been applied to wireless communication systems [1], [2], personal area networks [3], and solid-state drives [4], [5]. To eliminate the complicated hyperbolic computations required in the sum–product decoding algorithm, recent LDPC decoders are implemented based on the min–sum (MS) decoding algorithm [4]–[8]. In the MS algorithm, the check-node (CN) operation computes the first two minima and the index of the first minimum among many variable-to-check messages given as inputs. Generally, the hardware block that finds the first two minima, which is called a searching module (SM), can be implemented by employing the balanced tree structure [9]–[12]. The number of inputs to be compared in selecting the first two minima is increasing to achieve strong and long LDPC codes [4]–[6]. For example, a recent SM developed for storage applications deals with more than 100 inputs [4]. The hardware complexity of such a complex SM takes a significant portion in the overall complexity of an LDPC decoder. Moreover, the area taken by multiple SMs becomes more considerable in a high-throughput decoder, as massive CN operations are performed in parallel to increase the decoding throughput [11].

Manuscript received April 30, 2014; revised July 28, 2014; accepted September 24, 2014. Date of publication October 14, 2014; date of current version January 1, 2015. This work was supported by the IT R&D Program of MOTIE/KEIT [No. 10035202, Large Scale Hyper-MLC SSD Technology Development]. This brief was recommended by Associate Editor Y.-B. Kim.

Y. Lee is with Interuniversity Microelectronics Center (IMEC), 3001 Leuven, Belgium (e-mail: youngjoo.lee@imec.be).

B. Kim, J. Jung, and I.-C. Park are with the Department of Electrical Engineering, Korea Advanced Institute of Science and Technology (KAIST), Daejeon 305-701, Korea (e-mail: icpark@kaist.edu).

Digital Object Identifier 10.1109/TCSII.2014.2362663

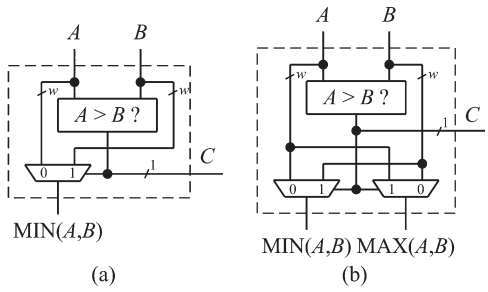

Fig. 1. Details of two component units. (a) C1M1 unit. (b) C1M2 unit.

A novel tree structure is proposed in this brief to minimize the number of comparators as well as the area–time (AT) complexity. Instead of finding the exact second minimum after finding the first minimum, the proposed algorithm collects the candidates of the second minimum while searching for the first minimum. The candidate set is easily constructed by reusing the comparison results performed for the first minimum. Compared to the previous SM, the proposed SM reduces the number of comparators by more than 40%.

The rest of this brief is organized as follows. Section II describes the previous works, and Section III explains the proposed low-complexity SM. The implementation results are compared to the previous works in Section IV. Finally, concluding remarks are made in Section V.

II. Previous Works

For a given set of k w-bit inputs, $X = \{x_0, x_1, \ldots, x_{k-1}\}$, the SM for k inputs produces three outputs: 1) the first minimum value $\text{MIN}_1 = \min\{X\}$, 2) the second minimum value, $\text{MIN}_2 = \min\{X - \{\text{MIN}_1\}\}$, and 3) the index of the first minimum IDX, which is i if x_i is MIN_1. Two 2-input primitive units, C1M1 and C1M2, are widely used to realize an SM. As shown in Fig. 1(a), the C1M1 unit that selects the smaller value from two inputs consists of one comparator and one w-bit 2-to-1 multiplexor. On the other hand, the C1M2 unit is made of one comparator and two w-bit 2-to-1 multiplexors to determine both the larger and smaller values, as depicted in Fig. 1(b). For the sake of simplicity, we focus in this brief on the generation of MIN_1 and MIN_2, as IDX can be obtained using the results of comparisons performed for MIN_1 [11]. In addition, let the number of inputs k be a power of 2, i.e., $k = 2^m$. When k is not a power of 2, such an SM can be achieved by pruning some leaf nodes of the balanced SM built with 2^m inputs where $2^m > k$, as described in the previous literatures [9]–[12].

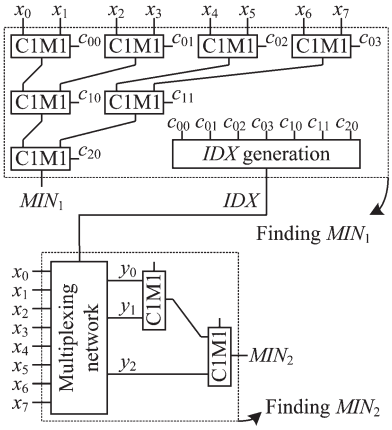

Fig. 2. Sorting-based searching module designed for eight inputs [9].

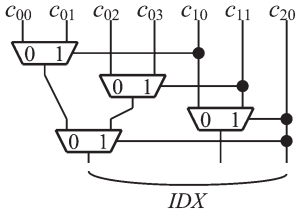

Fig. 3. Detailed structure for generating the index of the first minimum, i.e., *IDX*.

Fig. 2 depicts the conventional sorting-based SM, referred to as SM_{sort}, dealing with eight inputs [9], [10]. The overall process consists of two steps: 1) finding MIN_1 with the binary tree structure and 2) selecting MIN_2 by means of the multiplexing network controlled by IDX [9]. As shown in Fig. 3, IDX can simply be generated from the comparison results, where c_{ij} represents the jth comparison result at the ith step of the binary tree. The multiplexing network generates a candidate set of MIN_2, $Y = \{y_0, y_1, y_2\}$, by employing three 8-to-1 multiplexors. After choosing three candidates, two C1M1 units are used to determine MIN_2. As a result, the SM_{sort} necessitates nine comparators, three 8-to-1 multiplexors, and nine 2-to-1 multiplexors to process eight inputs and furthermore suffers from the long critical delay caused by the serially connected structure. Due to the miscellaneous control at the multiplexing network, the critical delay of SM_{sort} is slightly larger than $5T_C + 5T_{M2} + T_{M8}$, where T_C, T_{M2}, and T_{M8} stand for the delay of a comparator, a 2-to-1 multiplexor, and an 8-to-1 multiplexor, respectively [11].

For a high-speed realization, the tree-based SM architecture, referred to as SM_{tree}, was proposed in [11]. The SM_{tree} designed for eight inputs is exemplified in Fig. 4, where the processing times for MIN_1 and MIN_2 are almost the same as they are both based on the hierarchical tree structure. To calculate exact MIN_2 in each subtree, however, SM_{tree} requires more comparators than SM_{sort}. Three C1M1 units and one 2-to-1 multiplexor are additionally used to combine two subtrees, but the serially connected block required for finding MIN_2 in SM_{sort} is removed so that the critical delay of SM_{tree} is reduced to $3T_C + 5T_{M2}$. A faster tree-based SM, denoted as SM_{radix}, was achieved by adopting the mixed-radix scheme [12]. However, realizing the high-radix computation increases comparators and multiplexors drastically.

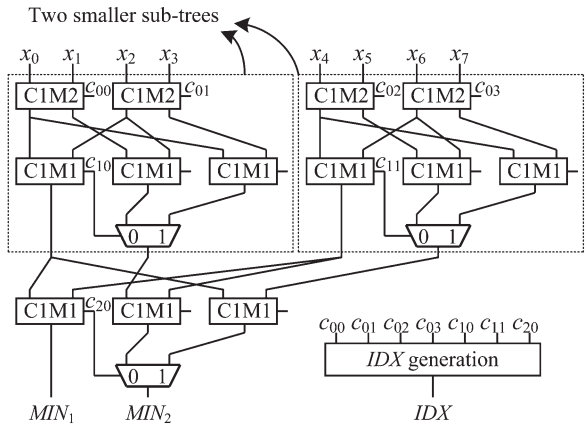

Fig. 4. Tree-based searching module for eight inputs [11].

As the hardware complexity of a comparator is considerable, the previous tree-based SM cannot be cost effective when the number of inputs is not small, particularly for recent strong LDPC codes targeting a row degree of more than 100 [4], [5]. Hence, it is necessary to develop a new SM that can reduce comparators while keeping the critical delay less than that of SM_{sort}.

III. Proposed Architecture

It is possible to reduce the number of comparators needed for the second minimum by reusing the comparison results performed for the first minimum. In the proposed architecture, a candidate set Y for MIN_2 is first constructed by using the prior comparison results, and then a comparison network is additionally constructed to select MIN_2. This two-step approach is conceptually similar to SM_{sort} [9], but the second step is much faster in the proposed architecture.

As previously discussed, SM_{sort} requires complex multiplexing networks to construct the candidate set and thus suffers from the long critical delay resulting from k-to-1 multiplexors. To eliminate the complex k-to-1 multiplexors, the proposed architecture introduces a basic unit, i.e., PRO_k, which produces the first minimum of k inputs and m $(= \log_2 k)$ candidates for the second minimum, as depicted in Fig. 5(a). Similar to SM_{tree}, a PRO_k unit can be recursively designed with two smaller $\text{PRO}_{k/2}$ units, as shown in Fig. 5(b). The first minimum of k inputs, i.e., MIN_1, is simply selected by comparing two minima, i.e., $\text{MIN}_1^{(L)}$ and $\text{MIN}_1^{(R)}$, produced in $\text{PRO}_{k/2}^{(L)}$ and $\text{PRO}_{k/2}^{(R)}$ units, respectively. Depending on the comparison result of the C1M2, the PRO_k decides $m-1$ candidates for the second minimum by selecting either the candidate set of $\text{PRO}_{k/2}^{(L)}$ or that of $\text{PRO}_{k/2}^{(R)}$. If $\text{MIN}_1^{(L)}$ is smaller than $\text{MIN}_1^{(R)}$, all the $m-1$ candidates of $\text{PRO}_{k/2}^{(R)}$ cannot be the second minimum, because $\text{MIN}_1^{(R)}$ is the smallest value among the 2^{m-1} inputs on the right side. Therefore, m candidates for the second minimum are simply formed by including one of $\text{MIN}_1^{(L)}$ and $\text{MIN}_1^{(R)}$ to the $m-1$ candidates selected by the result of the C1M2, as shown in Fig. 5(b). In short, a PRO_k unit can be realized with two $\text{PRO}_{k/2}$ units, one comparator and $m+1$ 2-to-1 multiplexors. It is apparent that a PRO_2 unit

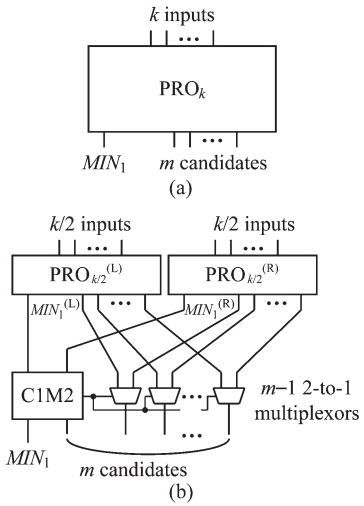

Fig. 5. (a) Proposed PRO_k unit. (b) Its recursive structure.

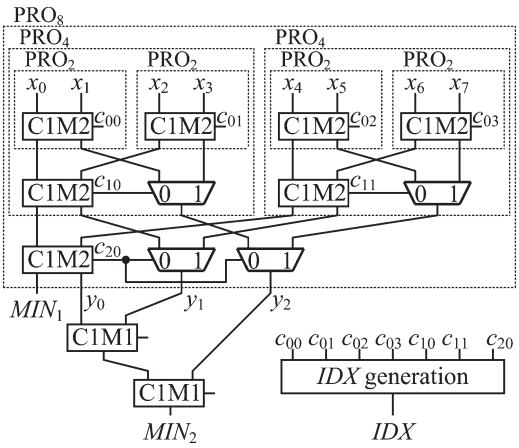

Fig. 6. Proposed searching module for eight inputs.

processing two inputs is identical to the C1M2 unit shown in Fig. 1(b).

The major point of the PRO_k is that the candidate set for the second minimum is constructed in parallel with the searching for the first minimum. After finding the first minimum, another tree structure that can be built with $m-1$ C1M1 units is used to find MIN_2 among m candidates.

Fig. 6 shows how the proposed SM, referred to as SM_{pro}, is constructed to process eight inputs, where a PRO_8 unit is followed by a tree structure composed of two C1M1 units to find MIN_2 among the three candidates produced in the PRO_8 unit. The SM_{pro} in Fig. 6 requires 9 comparators and 20 2-to-1 multiplexors to process eight inputs, whereas the SM_{tree} in Fig. 4 necessitates 13 comparators and 20 2-to-1 multiplexors for the same number of inputs. The critical delay of SM_{pro} is $5T_C + 5T_{M2}$, which is much smaller than that of SM_{sort}.

Table I compares the hardware complexities and critical delays of three different SM architectures, where the number of inputs is assumed to be a power of 2, $k = 2^m$. As there are m final candidates for MIN_2, SM_{sort} and SM_{pro} require $2^m + m - 2$ comparators in determining two minima, which is much smaller than that of SM_{tree}. As the proposed architecture completely removes the 2^m-to-1 multiplexors that are

TABLE I
COMPARISONS OF SEARCHING MODULES FOR 2^m INPUTS

Architecture		Sorting-based [9]	Tree-based [11]	Proposed
Number of components	Comparators	2^m+m-2	$2^{m+1}-3$	2^m+m-2
	2-to1 Multiplexors	2^m+m-2	$3\cdot 2^m-4$	$3\cdot 2^m-4$
	2^m-to-1 Multiplexors	m	0	0
Critical delay		$(m+\lceil\log_2 m\rceil)T_C+T_{Mk}+(m+\lceil\log_2 m\rceil)T_{M2}$	$mT_C+(2m-1)T_{M2}$	$(m+\lceil\log_2 m\rceil)T_C+(m+1+\lceil\log_2 m\rceil)T_{M2}$

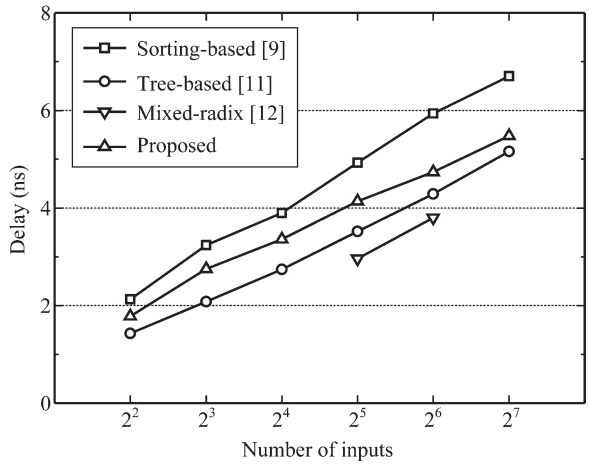

Fig. 7. Critical delay of searching modules for a range of inputs.

inevitable in SM_{sort}, the critical delay of SM_{pro} is much smaller than that of SM_{sort}. Note that the delay of SM_{pro} is quite comparable with that of SM_{tree}, as indicated in Table I. As the number of inputs increases, in addition, the proposed structure becomes more area efficient. If the number of inputs is increased to 64, for example, the proposed architecture eliminates 40% comparators of SM_{tree}, remarkably reducing the hardware complexity.

IV. IMPLEMENTATION RESULTS

For fair comparison, four types of SMs with a computational bit width of 8 bits are designed and synthesized in a 130-nm CMOS process: the sorting-based one [9], the tree-based one [11], the mixed-radix one [12], and the proposed one. Fig. 7 illustrates how the critical delays of the four SMs are affected by the number of inputs. Note that the critical delay of SM_{sort} is always larger than those of the other SMs due to the complex 2^m-to-1 multiplexing networks. As the proposed structure requires additional comparison steps to find the second minimum among m candidates, the delay of SM_{pro} is between those of the previous sorting- and tree-based SMs. The tree-based architectures described in [11] and [12] are attractive in terms of computational delay, but they necessitate a significant amount of hardware complexity. The proposed architecture leads to an area-efficient SM, as indicated in Fig. 8. When m is 6, which corresponds to 64 inputs, SM_{pro} saves 34%, 33%, and 68% area over SM_{sort}, SM_{tree}, and SM_{radix}, respectively. As a result,

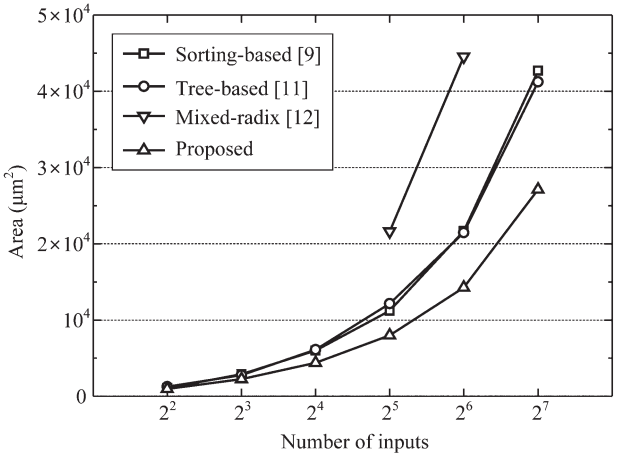

Fig. 8. Area complexity of searching modules for a range of inputs.

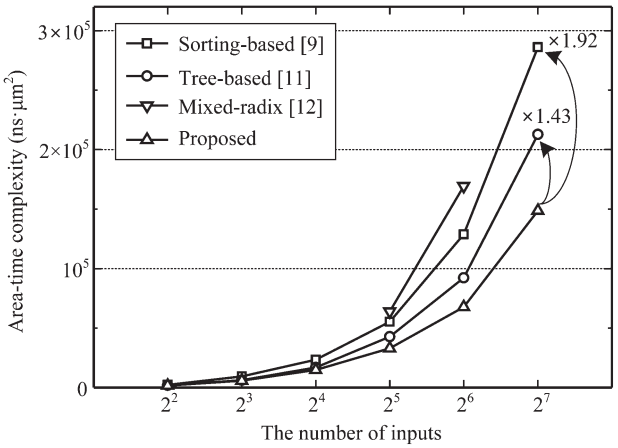

Fig. 9. AT complexity of searching modules for a range of inputs.

the proposed SM improves the AT complexity remarkably, as shown in Fig. 9. When m is 7, for example, the AT complexity of $\mathrm{SM}_{\mathrm{tree}}$ is 43% larger than that of $\mathrm{SM}_{\mathrm{pro}}$.

V. Conclusion

We have presented a novel tree structure that finds the first two minima among many inputs. In the proposed structure, the candidates of the second minimum are collected by utilizing the results of comparisons performed for the first minimum. Hence, the proposed structure minimizes the number of comparators, leading to a low-complexity realization. In addition, the second minimum is selected from the candidates by carrying out a few comparison steps. As the proposed structure eliminates the large-sized multiplexing networks, it improves the AT complexity significantly compared to those of the previous state-of-the-art SMs.

References

[1] *IEEE Standard for Local and Metropolitan Area Networks Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, IEEE Std. 802.11n-2009, Oct. 2009.

[2] *IEEE Standard for Local and Metropolitan Area Networks Part 16: Air Interface for Broadband Wireless Access Systems*, IEEE Std. 802.16-2009, May 2009.

[3] *IEEE Standard for Local and Metropolitan Area Networks Part 15.3: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for High Rate Wireless Personal Area Networks (WPANs)*, IEEE Std. 802.15.3c-2009, Oct. 2009.

[4] J. Kim and W. Sung, "Rate-0.96 LDPC decoding VLSI for soft-decision error correction of NAND Flash memory," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 22, no. 5, pp. 1004–1015, May 2014.

[5] J. Kim, D. Lee, and W. Sung, "Performance of rate 0.96 (68254, 65536) EG-LDPC code for NAND Flash memory error correction," in *Proc. IEEE ICC*, 2012, pp. 7029–7033.

[6] Y. Sun and J. R. Cavallaro, "VLSI architecture for layered decoding of QC-LDPC codes with high circulant weight," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 21, no. 10, pp. 1960–1964, Oct. 2013.

[7] F. Angarita, J. Valls, V. Almenar, and V. Torres, "Reduced-complexity min–sum decoding algorithm for decoding LDPC codes with low error-floor," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 61, no. 7, pp. 2150–2158, Jul. 2014.

[8] Y.-L. Ueng, B.-J. Yang, C.-J. Yang, H.-C. Lee, and J.-D. Yang, "An efficient multi-standard LDPC decoder design using hardware-friendly shuffle decoding," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 60, no. 3, pp. 743–756, Mar. 2013.

[9] D.E. Knuth, *The Art of Computer Programming*, 2nd ed. New York, NY, USA: Addison-Wesley, 1998.

[10] Q. Xie, Z. Chen, X. Peng, and S. Goto, "A sorting-based architecture of finding the first two minimum values for LDPC decoding," in *Proc. IEEE CSPA*, 2011, pp. 95–98.

[11] C.-L. Wey, M.-D. Shieh, and S.-Y. Lin, "Algorithms of finding the first two minimum values and their hardware implementation," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 55, no. 11, pp. 3430–3437, Dec. 2008.

[12] L. G. Amaru, M. Martina, and G. Masera, "High speed architectures for finding the first two maximum/minimum values," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 20, no. 12, pp. 2342–2346, Dec. 2012.