IEICE TRANS. FUNDAMENTALS, VOL.E94–A, NO.11 NOVEMBER 2011

PAPER Special Section on Smart Multimedia & Communication Systems

A Fast Systematic Optimized Comparison Algorithm for CNU Design of LDPC Decoders

Jui-Hui HUNG[†a)] ***and*** **Sau-Gee CHEN**[†b)]**,** ***Members***

SUMMARY This work first investigates two existing check node unit (CNU) architectures for LDPC decoding: self-message-excluded CNU (SME-CNU) and two-minimum CNU (TM-CNU) architectures, and analyzes their area and timing complexities based on various realization approaches. Compared to TM-CNU architecture, SME-CNU architecture is faster in speed but with much higher complexity for comparison operations. To overcome this problem, this work proposes a novel systematic optimization algorithm for comparison operations required by SME-CNU architectures. The algorithm can automatically synthesize an optimized fast comparison operation that guarantees a shortest comparison delay time and a minimized total number of 2-input comparators. High speed is achieved by adopting parallel divide-and-conquer comparison operations, while the required comparators are minimized by developing a novel set construction algorithm that maximizes shareable comparison operations. As a result, the proposed design significantly reduces the required number of comparison operations, compared to conventional SME-CNU architectures, under the condition that both designs have the same speed performance. Besides, our preliminary hardware simulations show that the proposed design has comparable hardware complexity to low-complexity TM-CNU architectures.

key words: *channel coding, LDPC decoder, comparison operation, algorithm, hardware*

1. Introduction

Low-Density parity check (LDPC) code [1] can achieve performance close to Shannon bound. As such, LDPC has been adopted by many state-of-the-art communication systems. It is a kind of binary linear block code whose parity check matrix is sparse which has much fewer 1s than a common matrix. A sparse parity check matrix facilitates simple decoding algorithms and low-complexity decoder designs. Check matrix of a LDPC code is often represented by a bipartite graph, called Tanner graph [2], which is composed of n variable nodes (realized by variable node units (VNU)) and m check nodes (realized by CNUs). Those variable nodes and check nodes are connected by edges defined by the nonzero entries of the parity-check matrix $\boldsymbol{H}$. Figure 1 shows an example with 4 check nodes and 8 variable nodes. The number of "1" in each column of $\boldsymbol{H}$ determines the number of edges for each variable node connected to check nodes, and the number of "1" in each row of $\boldsymbol{H}$ determines the connections from each check node to variable nodes. Tanner graph shows a clear picture of all the information exchange links in a decoding process.

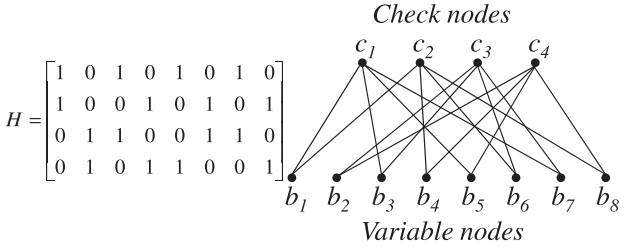


Fig. 1 Tanner graph of a parity check matrix.

Decoding of a LDPC code is generally carried out by Sum-Product algorithm (SPA) [3]. However, it is too complicated to be practically realized. Min-Sum algorithm (MSA) [4] is a popular, accurate and low-complexity approximation to SPA. Hence, the existing LDPC decoders are basically all based on MSA. According to MSA, for every decoding iteration, each CNU has to send updated messages to all its connecting variable nodes. The updated message a variable node receives from its connecting check node should be the minimum values among all the message values the CNU received from all its connecting variable nodes, except for the message sent from the target variable node to the check node. Hence, a CNU mostly is performing comparison operations in a decoding iteration and find out the minimum message values it needs to send to all its corresponding connecting variable nodes. As such, the most time-consuming operations in LDPC decoders are the comparison operations required by check nodes.

Due to the mentioned characteristic of MSA, in the literature [5]–[7], roughly there are two different CNU design approaches for LDPC decoders. Consider a check node and one of its connecting variable nodes. The first approach is that the CNU finds the minimum message value (for sending to the variable node) without including the variable node's message (as dictated by MSA). The CNU in [8] is based on this approach which has the advantage of high-speed, independent and parallel decision of all the minimum message values. However, it has the disadvantage of high area cost, because so many redundant comparison operations are performed. For the convenience of later discussion, we call this type of CNU design as "self-message-excluded CNU" (i.e., SME-CNU for simplicity).

On the other hand, the second approach finds out the global minimum and the second minimum values from all the message values the check node receives from all its

Manuscript received January 27, 2011.
Manuscript revised May 12, 2011.

[†]The authors are with the Institute of Electronics, National Chiao Tung University, 1001 University Road, Hsinchu, Taiwan 300, ROC.
a) E-mail: paholisi.ee99g@nctu.edu.tw
b) E-mail: sgchen@mail.nctu.edu.tw

DOI: 10.1587/transfun.E94.A.2246

connecting variable nodes, and then picks either the second minimum value or the minimum value, depending on if the particular variable node's message value is equal to the minimum value or not. This approach significantly reduces the amount of comparison operations, compared with the first approach. The CNU design in [9] adopts the approach, which effectively reduces the required number of the comparison operations but has a longer delay time than the first approach. Similarly, for convenience, we term this kind of CNU designs as "two-minimum CNU" (i.e., TM-CNU for simplicity).

Since this work focuses on high-speed applications, the proposed systematic optimization technique also assumes SME-CNU structure. In order to reduce the redundant comparison operations involved in SME-CNU, the approach in [8] proposes a 6-input SME-CNU architecture, by considering the design of reusable and sharable comparator cells. However, it only conducted a customized specific design with 6 inputs, without proposing a systematic and generalized design approach for arbitrary input numbers. For cases with long code lengths, it will be impractical to optimize the designs manually. Therefore, it is very desirable to have an efficient and automatic generator for the comparison operations and comparators of SME-CNU. To address this design challenge, this work is aimed to achieve the fastest comparison speed with the lowest comparator complexity as much as possible.

This paper is organized as follows. In Sect. 2, the widely used Sum-Product algorithm (SPA) and Min-Sum algorithm (MSA) LDPC decoding algorithms are introduced. Section 3 introduces key comparison algorithms used in CNUs, the associated CNU architectures, and some CNU design issues. Section 4 details the proposed optimized systematic comparison algorithm for SME-CNU design. Section 5 presents the performance of the proposed algorithm. Finally, Sect. 6 is the conclusion.

2. Decoding Algorithm for LDPC Codes

Decoding of a LDPC code is generally carried out by SPA, which is based on log-likelihood ratio (LLR) of the form: $L(x) = \log(P\{x = 0\}/P\{x = 1\})$. In the decoding process, the ith variable node v_i needs to update the following message $L_{vi\to cj}$, which will then be sent by the variable node to its connecting check node c_j,

$$L_{vi\to cj} = channel(v_i) + \sum_{c_k\in C\backslash c_j} L_{ck\to vi}, \tag{1}$$

where $C\backslash c_j$ is a set containing all the check nodes connected to variable node v_i (excluding check node c_j), and $channel(v_i)$ is the channel value of v_i. This equation is implemented with a VNU. On the other hand, check node c_j needs to update and send the following message $L_{cj\to vi}$ to variable node v_i,

$$L_{cj\to vi} = 2\tanh^{-1}\left(\prod_{v_k\in B\backslash v_i}\tanh\left(\frac{L_{vk\to cj}}{2}\right)\right), \tag{2}$$

where $B\backslash v_i$ is a set containing all the variable nodes connected to c_j (excluding variable node v_i). Since Eq. (2) is too complicated to be practically realized, MSA [4] is proposed which is a popular, accurate and low-complexity approximation to (2), as shown below.

$$L_{cj\to vi} \approx \left(\prod_{v_k\in B\backslash v_i} sign(L_{vk\to cj})\right) \times \min_{v_k\in B\backslash v_i}(|L_{vk\to cj}|) \tag{3}$$

MSA reduces decoder hardware complexities significantly, at the cost of a little performance loss. Like most existing designs, this work is also based on MSA. Decoding of a LDPC code is an iterative process composed of the following sequential decoding steps: 1) initialize the decoding parameters such as the maximum iteration count, iteration counter, channel values, and etc; 2) update check-node messages; 3) update bit-node messages; 4) hard decision of bit-node messages; 5) check if the iteration count exceed the maximum iteration number, or the decoded information bits satisfy the zero syndrome vector constraint, i.e., $\boldsymbol{x}\boldsymbol{H}^T = \vec{0}$ or not, where $\boldsymbol{x}$ denotes the decoded codeword in row vector form; if it is then output the decoded bits, otherwise go to step 2).

As can be seen, the key operation required in a LDPC decoding process is finding the minimum value from a set of samples as shown in (3). Therefore, it is crucial to achieve fast comparison operations with low hardware complexity. It is well-known that the fastest comparison operations of finding the minimum among a set of N_{in} samples can be achieved with a delay time of $\lceil\log_2 N_{in}\rceil T_{cmp}$ and an area of $(N_{in} - 1)$ comparators, by performing parallel binary comparison operations, where T_{cmp} is the delay time of a 2-input comparator and the ceiling function $\lceil\log_2 N_{in}\rceil$ is the number of comparison levels.

3. CNU Architectures & Comparison Algorithms

In this work, since we focus on the design of highest-speed LDPC decoders (while minimize the area) as much as possible, all the ensuing discussions assume unfolded CNU structures. As mentioned in Introduction, there are two different types of CNU structures, namely, SME-CNU and TM-CNU. The basic module of a complete SME-CNU is a direct realization of (3) as discussed below. Without loss of generality, consider check node c_j and its connecting ith variable node v_i. According to (3), c_j must find out the minimum value $L_{cj\to vi}$ (for sending it to v_i) from all its inputs excluding the input from v_i. Note that, a complete unfolded SME-CNU contains N_{in} parallel basic modules. Obviously, this design can achieve very high-speed performance, because it updates all the minimum message values in parallel, simultaneously and independently. However, it is at the cost of considerable hardware for parallel message generations and redundant comparison operations.

For the success of the SME-CNU architecture, one

needs to reduce the redundant comparators. The most intuitive way is to find out the common comparison terms in the parallel generation of minimum message values [8]. However, the design in [8] only did a customized optimization for a specific 6-input CNU architecture without proposing a systematic and generalized design approach for arbitrary input numbers. For cases with long code lengths, it will be impractical to optimize the designs manually.

By observing (3), it is obvious that the output value from the jth check node to the ith variable node will be the minimum value of all the input values to the jth check node if the minimum value is unequal to the input value from the ith variable node. Otherwise, output of the CNU to the ith variable node will be the second minimum value of all the CNU inputs. Thus, (3) can be rewritten as:

$$L_{cj\to vi} \approx \begin{cases} S \cdot \min\limits_{v_k\in B}(|L_{vk\to cj}|), \text{ if } L_{vk\to cj} \neq \min\limits_{v_k\in B}(|L_{vk\to cj}|) \\ S \cdot \min\limits_{v_k\in B} 2(|L_{vk\to cj}|), \text{ otherwise} \end{cases}, \quad (4)$$

where for simplicity, the overall sign term S is defined as

$$S = \prod_{v_k\in B\setminus v_i} sign(L_{vk\to cj}) \quad (5)$$

and min2(.) represents the function that finds out and returns the second minimum value from its input operands. By directly realizing (4), one can get the TM-CNU architecture as shown in Fig. 2. In the figure, XOR gates altogether generate the overall sign term S in (4) for all the outputs of CNU, while the two-minimum comparator (TMC) generates the minimum, the second minimum values, and indexing information of the minimum value [9] from CNU input values. The output "index of min" from TMC is simply the unsigned binary number of the variable node that has the minimum message value.

For example, if $N_{in} = 8$ and v_3 has the minimum message value, then "index of min" is equal to 011. The index decoder then expands the "index of min" value to a binary positional representation (similar to a de-multiplexer) that reflects the number of the variable node in position. In this example, since 011 is equal to 3, the output of index decode will be 00000100, where the value at the 3rd bit position is equal to one while all the other bit values are equal to zero. The decoded 8 output bits are served as the control signals to their respective multiplexers for selecting the appropriate minimum message values. In this case, the 3rd multiplexer will select the second minimum value, while all the other multiplexers will select the minimum value. Finally, the selected message values are sent to their corresponding variable nodes.

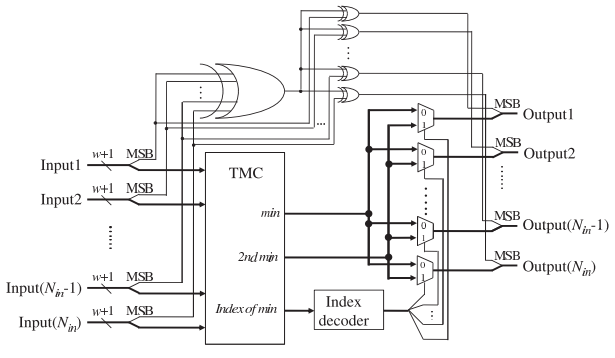


Fig. 2 A TM-CNU architecture.

Compared to SME-CNU architecture, TM-CNU architecture costs smaller area. However, since TM-CNU architecture needs to find out the second minimum value from all its input values, its delay time is inherently longer than that of SME-CNU architecture.

For finding the two minimum values in a TM-CNU, generally there are three approaches, namely, the double-elimination (DE) comparison scheme, the trace-back (TB) comparison scheme [10], and the tree-structure (TS) comparison scheme [11]. Any one the three schemes can be realized with TMC.

3.1 Double-Elimination (DE) Comparison Scheme

The concept of DE comparison scheme is similar to the double-elimination tournament in sport competitions. A double-elimination tournament is divided into two sets of brackets: winner's bracket (WB) and loser's bracket (LB). In the beginning, all the participants are in WB, and randomly (in a fair game) paired and entered the contests. For each paired contest, the winner will remain in WB and continue his/her random paired contests with the remaining players in WB, while the loser exits to LB. In WB, it is conducted as a single-elimination tournament. Similarly, all the players in LB conduct the similar single-elimination tournament as in WB. The loser of each paired contest in LB will be eliminated totally from the tournament. Finally, both WB and LB will generate their own winners, respectively. These two players then proceed to the final content for the championship. When applying the above game rules to CNU comparison operations, the winners of WB and LB respectively represent the minimum and the second minimum values among all the CNU inputs.

Consider an N_{in}-input CNU and each input of CNU has wordlength of $(w+1)$ bits (including a sign bit at the MSB). By adopting the DE comparison scheme, there will be N_{in} inputs (players) in WB and $N_{in}-1$ inputs (players) in LB. Hence, it needs $(N_{in}-1)$ and $(N_{in}-2)$ comparison operations to compute the minimum value and the second minimum value, respectively. For high speed consideration, parallel binary tree comparison operations can be adopted, which take $(\lceil \log_2 N_{in} \rceil + 1)$ comparison levels to get the final results. This number reflects the critical path delay time in a CNU. Figure 3 shows an example of a 6-input comparison contest chart using DE scheme to find out the minimum and second minimum values.

Note that since each comparison operation in WB needs to pass the loser to LB, it needs an additional 2-to-1 multiplexer. Note that it is easier to do comparison oper-

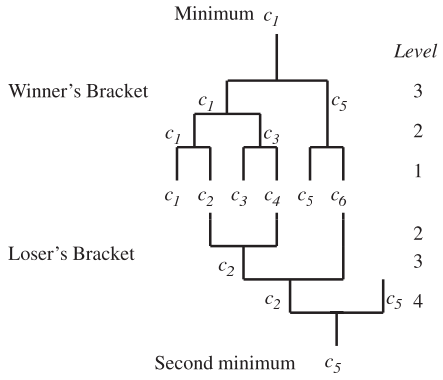


Fig. 3 An example of 6-input comparison operation chart based on DE comparison scheme.

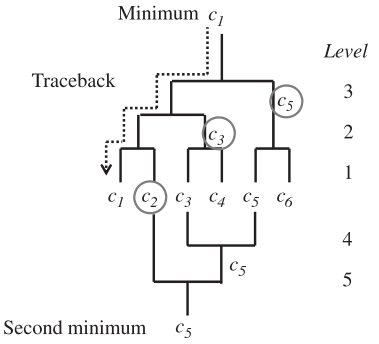


Fig. 4 Comparison chart based on the TB comparison scheme, of the same 6-input comparison example as in Fig. 3.

ations and find out the minimum value in (4), by adopting signed-magnitude data representations, as commonly done in the existing designs. Hence, the MSB of each CNU input is not required in the comparison operation, and only the w-bit magnitude part is involved in the comparison operation. In the ensuing discussion, all the comparators and multiplexers in a CNU assume w-bit operations, except for some specified cases otherwise.

If DE scheme is adopted, the TM-CNU architecture in Fig. 2 needs $(2N_{in} - 3)$ comparators, $(4N_{in} - 4)$ multiplexers, an index decoder, and some additional hardware (for indexing the minimum value in TMC). The critical path delay time of the design consists of $(\lceil \log_2 N_{in} \rceil + 1)$ units of a comparator delay time, $(\lceil \log_2 N_{in} \rceil + 2)$ units of a multiplexer delay time, a unit of an index decoder delay time and an additional delay time of the indexing circuit.

3.2 Trace-Back (TB) Comparison Scheme [10]

Although DE comparison scheme is simple in implementation, it may execute a large amount of redundant comparison operations in LB, because some members in LB may have already competed with some other members in LB when they were in WB. Consequently, one can eliminate those redundant operations by tracking those members in LB if they already met and compared before in WB so that one can immediately eliminate those losers without repeating the comparison operations done before. That is the main idea of TB comparison scheme. Take the same 6-input CNU design example as in Fig. 3, Fig. 4 shows the whole contest chart of TB scheme. In the figure, the contest chart in WB is exactly the same as that in Fig. 3 which produces the WB winner C_1 (i.e., the minimum value).

Next, one has to decide the second minimum value in LB as follows. Since C_5 lost to C_1 in the final-round contest of WB and it is the last one to be put into LB, it has very high potential of being the second minimum value. As such, it is enough that one can simply only compare C_5 with all those losers to C_1 in the previous contest rounds in WB. This can be done by tracing back the whole comparison tree branches of C_1 in WB, and at the same time only compare C_5 with those losers to C_1 in the tree comparator leaves. In Fig. 4, since C_3 is the first traced leaf member in the C_1 branch, it is compared with C_5 first. The winner (i.e., C_5 in this case) of the comparison operation will then be compared with the next traced leaf member (i.e., C_2 in this example), and so on. In the end of the trace-back comparison process in LB, the winner of LB represents the second minimum value of CNU operations.

TB comparison scheme can be easily shown to have the least amount of comparison operations. If realized with unfolded structure, an N_{in}-input CNU architecture [11] based on TB scheme needs $(N_{in} + \lceil \log_2 N_{in} \rceil - 2)$ comparators, $(\lceil \log_2 N_{in} \rceil 2^{\lceil \log_2 N_{in} \rceil} + 2N_{in} - 2)$ w-bit multiplexers, $(2^{\lceil \log_2 N_{in} \rceil}(2^{\lceil \log_2 N_{in} \rceil} - \lceil \log_2 N_{in} \rceil - 1)$ 1-bit multiplexers, $(\lceil \log_2 N_{in} \rceil)$ 1-bit inverters and an index decoder. The critical path of this design consists of $(\lceil \log_2 N_{in} \rceil + \lceil \log_2 \lceil \log_2 N_{in} \rceil \rceil)$ units of a comparator delay time, $(2\lceil \log_2 N_{in} \rceil + \lceil \log_2 \lceil \log_2 N_{in} \rceil \rceil + 1)$ units of a multiplexer delay time, $(\lceil \log_2 N_{in} \rceil - 1)$ units of a 1-bit multiplexer delay time, a unit of an inverter delay time and a unit of an index decoder delay time. Note that the indexing overhead in the TMC has already been included in these hardware cost and delay time quantities.

Although TB scheme costs fewer comparators than DE scheme, it needs some additional hardware to handle the trace-back operation and information recording. The overhead may contribute to a higher overall hardware cost than DE scheme. Besides, the delay time of a TB-based CNU is much longer than that of a DE-based CNU due to a larger number of comparator levels and some additional overhead for back tracing.

3.3 Tree-Structure (TS) Comparison Scheme [11]

The TS comparison scheme in [11] can be shown to be a compromised design between DE and TB schemes. TS scheme divides an N_{in}-input CNU into several minimum-value generators (mVG) with smaller numbers of inputs which can then generate their respective minimum values, second minimum values and indexing information. After that, the two global minimum values are decided from these local minimum values. A particular 4-input mVG [11] needs

only 5 comparators, 8 multiplexers and a 1-bit multiplexer with a total delay time consisting of 2 units of a comparator delay time and 3 units of a multiplexer delay time.

Generally, the critical path delay time of an N_{in}-input CNU based on TS scheme consists of $\lceil \log_2 N_{in} \rceil$ units of a comparator delay time, $2\lceil \log_2 N_{in} \rceil$ units of a multiplexer delay time and a unit of an index decoder delay time. Since TS scheme in [11] is specifically realized for different N_{in} cases, there is no general closed-form formula for the required hardware complexity. Here, we only discuss the case of assuming N_{in} is a power-of-2 number. For other cases, the results are similar. In this case, the design needs $(2N_{in} - 3)$ comparators, $(4N_{in} - 4)$ multiplexers, $(N_{in} - \log_2 N_{in} - 1)$ 1-bit multiplexers and an index decoder. Note that the indexing hardware overhead in TMC has already been included in these hardware cost and delay time quantities.

Compared to TB comparison scheme, although TS scheme needs more comparison operations, it has a much lower additional overhead. As a result, overall, TS scheme requires smaller area and shorter delay time than TB scheme when realized with the mentioned unfolded structures.

All the above discussed design techniques will be compared with the proposed design later. Specifically, as mentioned before we will focus on the design of highest-speed unfolded SME-CNU architecture. However, the biggest disadvantage of SME-CNU is its area cost which is much larger than TM-CNU. To lower the area cost of SME-CNU, we will present a systematic optimization algorithm for the synthesis of fast and parallel comparison operations of (3) so that the proposed design achieve the shortest delay time, while minimizes the required number of 2-input comparators. It can be done by designing sharable comparators as much as possible, while maintaining the highest operation speed as detailed next.

4. The Proposed Comparison Algorithm for SME-CNU [12]

To achieve the minimum comparison delay time, the proposed comparison algorithm is also based on (3) and fast parallel divide-and-conquer comparison algorithm. Additionally, to reduce the required total number of 2-input comparators, we recently proposed a vertical-and-horizontally-cyclic (VHC) set construction algorithm for the parallel comparison algorithm in [12]. The set algorithm can maximize sharable comparison operations so as to minimize the required number of comparators.

Before detailing the comparison algorithm and VHC set reconstruction algorithm, we will roughly introduce the design concept. Figure 5 shows a 7-input example (i.e., N_{in}=7). In the figure, the minimum value of each of the seven row sets is to be obtained in the top comparison level (i.e., Level $\lceil \log_2 N_{in} \rceil$=3). The seven minimum values are supposed to be generated in parallel and sent to their respective variable nodes. Each row set's minimum value can in turn be obtained by properly comparing two minimum values of the four subsets' minimum values in Level 2 (i.e., $\lceil \log_2 N_{in} \rceil - 1 = 2$). Those subsets are obtained in the VHC set reconstruction process. The process will generate some H-V, H and H-C sets (as will be defined later) which should cover all the elements in the seven sets of the top level. Note that these four subsets are maximized sharable subsets broken down from the row sets in the top level.

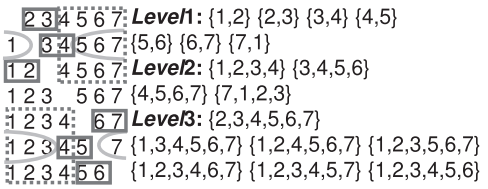


Fig. 5 An example of the proposed algorithm with 7 inputs.

For example, the minimum value of the first row set can be obtained by the operation of Min{Min{subset 2 in Level 1}, Min{subset 3 in Level 2}}, where Min{.} represents the operation which finds out the minimal value of its arguments. Finally, the four subsets can be broken down to the 7 primitive 2-input comparison sets in the bottom level (i.e., Level 1). Likewise, the four minimum values in Level 2 can be obtained by suitably comparing two minimum values of the 7 subsets in this level.

Next, the most key part of the proposed algorithm is how to construct and maximize sharable comparison sets so that the required number of comparators can be minimized. In the following, we will detail the set construction and optimization algorithm.

4.1 The Proposed VHC Set Construction Algorithms

To maximize sharable comparison sets across the input matrix rows as much as possible, we will utilize some symmetry properties discussed below. The first step of the proposed algorithm is to layout all the exclusive CNU input operand sets in a matrix, where the minimum value of each matrix row set is to be generated in parallel with all the other minimum row values. Observing the matrix pattern of Fig. 5, one can conclude the following two regular properties:

- First, the diagonal elements of the matrix are all blank (which correspond to the missing exclusive inputs). And there exhibits row set symmetry in the matrix.
- Second, each row of the matrix is almost the same as any others rows, except in two exclusive elements.

4.1.1 The Formation of H-V Sets

First due to the position symmetry (i.e., the first property) of the input matrix, the discussion can be limited to the upper right half part (i.e., the $\boldsymbol{U}$ part) of the matrix from row 1 to row 3. The lower left half part (i.e., the $\boldsymbol{L}$ part) from row 5 to 7 can be treated similarly, while comparison operations of the center symmetry row (i.e., row 4) is automatically solved once the upper and lower parts of the matrix are solved, as can be easily seen later.

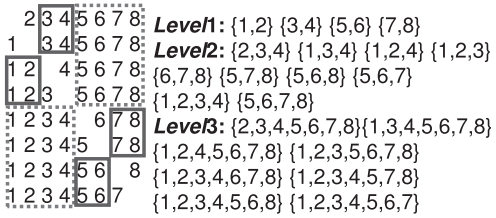


Fig. 6 An example of the proposed algorithm with 8 inputs.

Based on the second property, in the second step this algorithm finds out the intersection sets of the matrix that maximize the number of sharable compared results both in the horizontal and vertical directions of the input matrix. Moreover, in order to reduce the required number of comparators, each intersection set should contain 2^k elements, where k is a positive integer. We define this kind of sets as the H-V (horizontal-vertical) sets. For example, in Fig. 6, the set (in dotted square enclosing elements 5, 6, 7 and 8) is the initial maximum H-V set. Similarly, due to the mentioned symmetry, the dotted square enclosing elements 1, 2, 3 and 4 of rows 5 to 8 is another initial H-V set of the same size as the previous one. Next, one can find smaller H-V sets of size 2^{k-1} that contain the remnant elements. In Fig. 6, sets $\{3, 4\}$ (of rows 1 and 2), $\{1, 2\}$ (of rows 3 and 4), $\{7, 8\}$ (of rows 5 and 6) and $\{5, 6\}$ (of rows 7 and 8) include those remnant elements, as shown in the Level 1.

4.1.2 The Formation of H and H-C Sets

After formation of H-V sets, there is no more sharable elements in the vertical direction can be combined as an H-V set, and it can be easily shown that there are at most two remnant elements left in each row. Therefore, one can only form comparison sets in the horizontal (row) direction for those remaining elements. Figure 6 shows the case of only one element left in each row such as element 2 in the first row. In this condition, each one of those elements can be combined with the smallest H-V set and formed a H (horizontal) set, such as $\{2, 3, 4\}$ in the first row. Figure 5 shows the case of two elements left in each row, where elements 2 and 3, 1 and 3, 1 and 2 are remnant elements in row 1, row 2 and row 3, respectively. In this case, if the remnant elements are consecutive, then they can be formed an H set, for examples, $\{2, 3\}$ and $\{1, 2\}$ are two H sets formed in rows 1 and 3, respectively. Similarly, due to symmetry property, $\{5, 6\}$ of row 7 and $\{6, 7\}$ of row 5 are the two H sets formed this way.

Up to now, only non-consecutive remnant elements are left, such as elements 1 and 3, 5 and 7 in rows 2 and 6, respectively in Fig. 5. Of these elements, 3 and 5 are symmetric non-boundary elements which can be combined with their immediate neighbor elements and formed two additional H sets. In this case, element 3 is combined with element 4 in row 2, while element 5 is combined with element 4 in row 6.

Finally, only boundary remnant elements are left. In this case, there are the symmetric elements 1 (of row 2) and 7 (of row 6). We can then further define the H-C (horizontally cyclic) set of size 2^m which cyclically combines a boundary remnant element with its end-around neighbor elements in the same row. For example, element 1 can be combined with the end-around elements 5, 6 and 7 as the first H-C set $\{1, 7, 6, 5\}$ in row 2, while element 7 can be combined with its end-around elements 1, 2, and 3 as the second H-C set $\{7, 1, 2, 3\}$. For the consideration of least number of required 2-input comparators, we can take advantage of the mentioned symmetry property and form sharable sets by combining the boundary remnant element.

4.1.3 Detailed Flows of the Proposed Comparison Algorithm

The proposed complete comparison algorithm is divided into the following eight steps.

Step 1) Initialization: Layout all the exclusive input CNU operands in matrix form denoted as $\boldsymbol{M}$. Set the initial value of the iteration number variable N_{itr} and the row count variable N_r as $\lceil \log_2(N_{in} - 1) \rceil - 1$ and 1, respectively.

Step 2) Formation of H-V sets: Find out the maximum H-V sets which have $2^{N_{itr}}$ elements in the $\boldsymbol{U}$ part of $\boldsymbol{M}$. Due to symmetry, there are corresponding H-V sets in the $\boldsymbol{L}$ part which can be formed automatically, from the remnant elements (or all elements in the first iteration). Set $N_{itr} = N_{itr} - 1$.

Step 3) Update of H-V set size: Check if N_{itr}=0. If it is, go to Step 4, otherwise go back to Step 2.

Step 4) Formation of H sets: Form the H sets in the N_rth row of $\boldsymbol{M}$.

Step 5) Formation of H-C sets: Form H-C sets that cover those boundary remnant elements in the N_rth row of $\boldsymbol{M}$, after H set is formed. If no remnant element is left, go to Step 6.

Step 6) Check of termination condition: Set $N_r = N_r$+1. If $N_r > N_{in}$, go to Step 7, otherwise go back to Step 4.

Step 7) Execution of binary comparison operations: Perform parallel binary divide-and-conquer comparison operations for each row, based on the formed H sets, H-V sets and H-C sets in the previous steps.

Step 8) End of the algorithm: Output N_{in} parallel comparison operation results.

Figure 7 shows the implementation of a 7-input CNU architecture using the proposed VHC algorithm. The CNU is a realization of the design example in Fig. 5. The function block "Min" in Fig. 7 executes a comparison operation which is composed of a comparator and a 2-to-1 multiplexer.

5. Performance of the Proposed Design

The SME-CNU in [8] also considers the design of sharable comparators for parallel and simultaneous generation of all the necessary minimum message values. However, it only

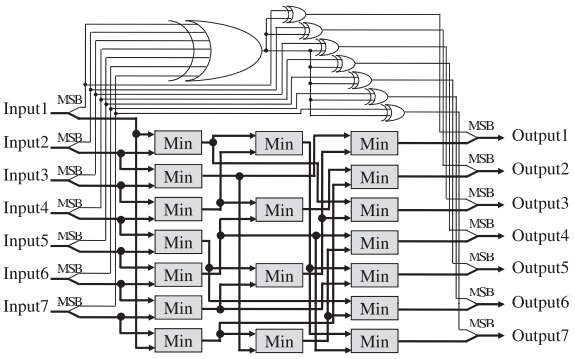


Fig. 7 The 7-input CNU architecture using the proposed VHC algorithm.

manually optimized the design of specific 6-input case, and did not present a general and systematic design methodology assuming arbitrary input numbers. Consequently, except for the case of N_{in}=6, there is no further data provided in [8] so that it can be compared with the proposed design. Coincidently, when N_{in}=6, the required hardware of the proposed design is the same as the design in [8]. It suggests that the proposed algorithm can systematically optimize the design with comparable performances to the manually optimized design.

Since the proposed design is for SME-CNU, while DE, TS schemes are for TM-CNU, it is hard to do a fair comparison of them with the proposed design. Nonetheless, we still can provide some insights on the advantages of the proposed design. Table 1 summarizes the performances of the proposed design and the existing designs mentioned before, where symbols A and T with subscripts denote the area and time measures of a particular hardware cell specified by the subscript, respectively. The subscript abbreviations *cmp*, *mux*1, *mux*2, *inv*, *xor* and *idx* represent a 2-input comparator, a 1-bit 2-to-1 multiplexer, a w-bit 2-to-1 multiplexer, an inverter, an XOR gate and an index decoder, respectively. In the table, N_{cmp} represents the number of 2-input comparators required in the CNU architecture based on the proposed VHC algorithm. As can be seen, the proposed VHC-based SME-CNU design has a much smaller area complexity than the direct SME-CNU design (because $N_{in} \times N_{in} \gg N_{cmp}$). The direct SME-CNU is a direct N_{in}-copy realization of Eq. (3) without optimization of sharable comparators. For fair comparison, the same indexing technique for TS scheme in [11] is also applied to generate the index information in TMC for DE scheme. Thus, the additional hardware and delay time of the indexing hardware in TMC as mentioned in Sect. 3.1 will be $(\lceil \log_2 N_{in} \rceil)A_{inv} + (2^{\lceil \log_2 N_{in} \rceil}(2^{\lceil \log_2 N_{in} \rceil} - \lceil \log_2 N_{in} \rceil - 1)A_{mux1}$ and $T_{inv} + (\lceil \log_2 N_{in} \rceil - 1)T_{mux1}$, respectively.

There is no closed-form formula for the required comparator number of the proposed design, because the comparator number does not consistently increase with the input number. The proposed design obviously needs more comparators than the other designs, because the design goal is to achieve a CNU design that has the shortest delay time

Table 1 Comparisons of time and area complexities for various unfolded CNU architectures.

Direct SME-CNU	Area	$(N_{in}(N_{in}-2))(A_{cmp}+A_{mux2})$
	Time	$(\lceil \log_2(N_{in}-1) \rceil)(T_{cmp}+T_{mux2})$
SME-CNU based on VHC algorithm	Area	$N_{cmp}(A_{cmp}+A_{mux2})$
	Time	$(\lceil \log_2(N_{in}-1) \rceil)(T_{cmp}+T_{mux2})$
TM-CNU based on DE scheme	Area	$(2N_{in}-3)A_{cmp}+(4N_{in}-4)A_{mux2}+((2^{\lceil \log_2 N_{in} \rceil}-\lceil \log_2 N_{in} \rceil-1)2^{\lceil \log_2 N_{in} \rceil})A_{mux1}+\lceil \log_2 N_{in} \rceil A_{inv}+A_{idx}$
	Time	$(\lceil \log_2 N_{in} \rceil+1)T_{cmp}+(2\lceil \log_2 N_{in} \rceil+1)T_{mux2}+T_{inv}+T_{idx}$
TM-CNU based on TB scheme	Area	$(N_{in}+\lceil \log_2 N_{in} \rceil-2)A_{cmp}+(\lceil \log_2 N_{in} \rceil 2^{\lceil \log_2 N_{in} \rceil}+2N_{in}-2)A_{mux2}+((2^{\lceil \log_2 N_{in} \rceil}-\lceil \log_2 N_{in} \rceil-1)2^{\lceil \log_2 N_{in} \rceil})A_{mux1}+\lceil \log_2 N_{in} \rceil A_{inv}+A_{idx}$
	Time	$(\lceil \log_2 N_{in} \rceil+\lceil \log_2 \lceil \log_2 N_{in} \rceil \rceil)T_{cmp}+(2\lceil \log_2 N_{in} \rceil+\lceil \log_2 \lceil \log_2 N_{in} \rceil \rceil+1)T_{mux2}+(\lceil \log_2 N_{in} \rceil-1)T_{mux1}+T_{inv}+T_{idx}$
TM-CNU based on TS scheme	Area*	$(2N_{in}-3)A_{cmp}+(4N_{in}-4)A_{mux2}+A_{idx}$
	Time	$(\lceil \log_2 N_{in} \rceil)T_{cmp}+(2\lceil \log_2 N_{in} \rceil)T_{mux2}+T_{idx}$

*Only list the case of power-of-2 N_{in}.

Table 2 The required numbers of 2-input comparators vs. input number for various designs.

N_{in}	6	7	8	9	10	11	12	13	14
Direct SME-CNU	24	35	48	63	80	99	120	143	168
Proposed VHC SME-CNU	12	18	22	28	25	41	36	49	38

while consumes as least hardware as possible. However, the number of comparators is much reduced than without optimization, as shown in Table 2. The table shows the required comparator number versus CNU input number for various design approaches.

Besides, in average one also can find that the proposed design consumes about 1.76-times comparators of those required by the CNU architectures based on DE and TS schemes. However, since the proposed CNU design is an SME-CNU, it doesn't need those indexing hardware (such as multiplexer, inverter and index decoder) required by TM-CNU (based on DE and TS scheme). The CNU areas (in gate counts) and delays versus CNU input numbers, due to the proposed and existing comparison schemes are shown in Fig. 8, where the number marked on the top of each bar is the maximal delay time of its corresponding CNU design. All the design results are synthesized by Synopsys™ Design Compiler, based on UMC CMOS 90-nm cell library, without setting any timing and area constraint for fair com-

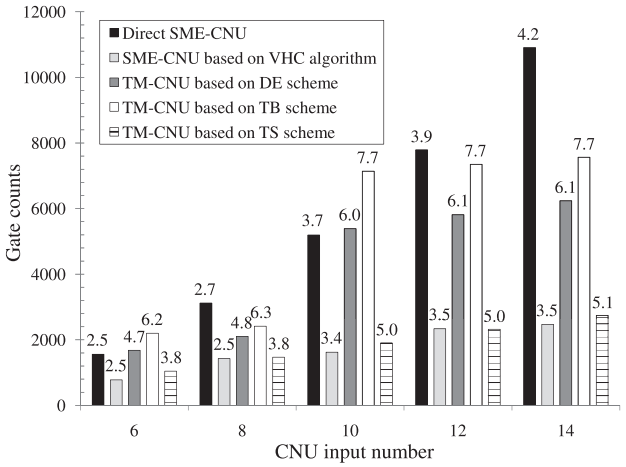


Fig. 8 The synthesized CNU areas (in gate counts) versus CNU input numbers, due to the proposed and existing comparison schemes. Those numbers on the bars indicate the corresponding synthesized delay times.

parison. As can be seen, the proposed CNU designs have lower delays and smaller areas than all the compared design. Thus, for high-speed applications, the proposed design achieves the best speed with low area cost.

6. Conclusion

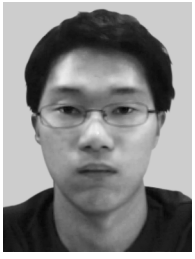
The presented systematically optimized comparison algorithm achieves efficient SME-CNU designs with the shortest critical path delays and low comparator counts. For very long LDPC code lengths, it will take immeasurable time to optimize the designs manually. For those cases, the proposed VHC technique can provide close to optimal solutions in a short time. Finally, the technique can give the designers reference results to help them further improve their designs when they consider the hand-crafted structures. The proposed algorithm can also be applied to VNU design by replacing all 2-input comparators with 2-input adders units.

Acknowledgments

This work is supported in part by the grants NSC 98-2220-E-009-029 and NSC 98-2219-E-009 -010, Taiwan.

References

[1] R.G. Gallager, Low-Density Parity-Check Codes, MIT Press, MA, 1963.
[2] R. Tanner, "A recursive approach to low complexity codes," IEEE Trans. Inf. Theory, vol.27, pp.533–547, Sept. 1981.
[3] D.J.C. MacKay, "Good error-correcting codes based on very sparse matrices," IEEE Trans. Inf. Theory, vol.45, no.2, pp.399–431, March 1999.
[4] X.Y. Hu, E. Eleftheriou, D.M. Arnold, and A. Dholakia, "Efficient implementation of the sum-product algorithm for decoding LDPC codes," IEEE Proc. GLOBECOM, vol.02, pp.1036–1036E, Nov. 2001.
[5] M.M. Mansour and N.R. Shanbhag, "Design methodology for high-throughput memory-efficient programmable decoder cores for architecture-aware low-density parity-check codes," IEEE Workshop on SiPS, pp.159–164, Aug. 2003.
[6] M.M. Mansour and N.R. Shanbhag, "High-throughput LDPC decoders," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol.11, no.6, pp.976–996, Dec. 2003.
[7] M.M. Mansour and N.R. Shanbhag, "Architecture-aware low-density parity-check codes," IEEE Proc. ISCAS, vol.2, pp.57–60, May 2003.
[8] M. Karkooti and J.R. Cavallaro, "Semi-parallel reconfigurable architectures for real-time LDPC decoding," IEEE Proc. ITCC, vol.1, pp.579–585, April 2004.
[9] C.-C. Lin, K.-L. Lin, H.-C. Chang, and C.-Y. Lee, "A 3.33 Gb/s (1200, 720) low-density parity check code decoder," IEEE Proc. ESSCIRC, pp.211–214, Sept. 2005.
[10] J. Snyders, "Reduced lists of error patterns for maximum likelihood soft decoding," IEEE Trans. Inf. Theory, vol.IT-37, pp.1194–1200, July 1991.
[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., vol.55, no.11, Dec. 2008.
[12] J.H. Hung and S.G. Chen, "A systematic optimized comparison algorithm for fast LDPC decoding," IEEE Proc. ISSPIT, pp.922–926, Dec. 2007.



Jui-Hui Hung received his B.S. degree from National Chi Nan University, Taiwan, in 2005 and M.S. degree in Electronics Engineering from National Chiao Tung University in 2007. He is currently a Ph.D. student in the Institute of Electronics, National Chiao Tung University. His research interests include digital signal processing, channel coding, VLSI architecture and bio-information.

Sau-Gee Chen received his B.S. degree from National Tsing Hua University, Taiwan, in 1978, M.S. degree and Ph.D. degree in electrical engineering, from the State University of New York at Buffalo, NY, in 1984 and 1988, respectively. Currently, he is a professor at the Department of Electronics Engineering, National Chiao Tung University, Taiwan. He was the director of Institute of Electronic at the same organization from 2003 to 2006. During 2004–2006, he served as an associate editor of IEEE Transactions on Circuits and Systems I. His research interests include digital communication, multi-media computing, digital signal processing, and VLSI signal processing. He has published more than 100 conference and journal papers, and holds several US and Taiwan patents.