International Journal of Electronics Signals and Systems

Volume 4 | Issue 1 Article 5

July 2014

A LAYERED DECODING ARCHITECTURE FOR LDPC DECODER WITH LOW ENERGY CONSUMPTION

G.SURESH KRISHNA SAGAR
GATES Institute of Technology, Anantapur, gsk.sagar@gmail.com

M. SANDEEP KUMAR
GITAM University, Vizag, mk_sandeep@yahoo.com

A LAYERED DECODING ARCHITECTURE FOR LDPC DECODER WITH LOW ENERGY CONSUMPTION

G.SURESH KRISHNA SAGAR[1] & M.SANDEEP KUMAR[2]

[1]GATES Institute of Technology, Anantapur
[2]GITAM university, Vizag

Abstract—Low-density parity-check (LDPC) decoder requires large amount of memory access which leads to high energy consumption. To reduce the energy consumption of the LDPC decoder, memory-bypassing scheme has been proposed for the layered decoding architecture which reduces the amount of access to the memory storing the soft posterior reliability values. In this work, we present a scheme that achieves the optimal reduction of memory access for the memory bypassing scheme. The amount of achievable memory bypassing depends on the decoding order of the layers. We formulate the problem of finding the optimal decoding order and propose algorithm to obtain the optimal solution. We also present the corresponding architecture which combines some of memory components and results in reduction of memory area. The proposed decoder was implemented in TSMC 0.18 m CMOS process. Experimental results show that for a LDPC decoder targeting IEEE 802.11n specification, the amount of memory access values can be reduced by 12.9–19.3% compared with the state-of-the-art design. At the same time, 95.6%–100% hardware utilization rate is achieved.

Index Terms- *Low power, low-density parity-check code, simu-lated annealing.*

I. INTRODUCTION

Recently,low density parity check codes (LDPC) have gained significant attention due to near shannon limit performance.They have been adopted in several wireless standards such as DVB S-2,IEEE 802.16e and 802.11n because of their excellent error correcting performance. A LDPC code is a linear block code defined by a sparse parity check matrix [Fig. 1(a)]. It can be represented by a bipartite graph, called Tanner Graph as shown in Fig. 1(b), which contains two sets of nodes: variable nodes that represent the bits of a codeword and check nodes that implement the parity-check constraints. The standard decoding procedure is the mes-sage passing algorithm, also known as "sum-product" or "belief propagation" (BP) algorithm , which iteratively exchanges the messages between the check nodes and the variable nodes along the edges of the graph. In the original message passing algorithm, the messages first are broadcasted to all the check nodes from the variable nodes and then along the edges of the graph the updated messages are fed back from the check nodes to the variable nodes to finish one iteration of decoding. There

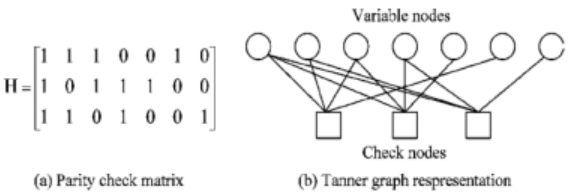

Fig. 1. Example of parity check matrix of a LDPC code and its Tanner graph representation.

are different ways to implement the LDPC decoder, based on the number of processing units available. In general, the LDPC decoder architecture can be classified into three types: fully par-allel architecture, serial architecture and partial parallel archi-tecture. In fully parallel architecture , a check node processor is needed for every check node, which usually results in large hardware cost and complicated routing, and hence is less flex-ible. The serial architecture uses just one check node processor to share the computation of all the check nodes and is too slow for most applications. For partial parallel architectures, multiple processing units are used allowing proper tradeoff between the hardware cost and the throughput and are commonly adopted in the actual implementation. In order to achieve higher convergence speed, i.e., to minimize the number of decoding it-eration, serial message passing algorithm, also known as layered decoding algorithm, has been proposed together with the corresponding partial parallel architecture. There are two types of layered decoding schemes: vertical layered decoding and horizontal layered decoding .In the horizontal layered decoding, a single or a certain number of check nodes (called layer) are first updated. Then the whole set of neighboring variable nodes are updated, and the decoding process proceeds layer after layer. Dually, in the vertical layered decoding, a single or a certain number of variable nodes (layer of variable nodes) may be up-dated first. Then the whole set of neighboring check nodes are updated. Because the serial check node processor is easier to be implemented in VLSI and therefore the horizontal layered decoding is preferable for practical implementations Be-cause of the faster convergence and regular architecture, layered decoders are commonly found in the LDPC decoder implementation . In this work, we focus on the

LDPC decoding implementation based on the layered decoding algorithm. High power consumption is one of the bottlenecks for LDPC decoder design. Recently low power design techniques and architectures have been proposed for fully parallel , and partial-parallel LDPC decoder architecture. The partial parallel architectures based on the layered decoding algorithm have efficiently reduced the hardware cost and sped up the convergence rate, and already led to energy-efficient design comparing with other architectures. However how to further reduce its energy consumption is still a challenging design problem. Due to the large amount of memory access, the power consumption of the memory access accounts for the major part of the total power consumption of the layer decoder. Reducing the energy consumption of the memories is the key issue to realize a low energy LDPC decoder. At the algorithmic level, the Min-sum decoding algorithm and its variants have been proposed, which greatly reduces the memory storage required for the check to variable messages, and also the energy consumption of the memories of the LDPC decoder with insignificant performance loss. At the architectural level, memory-bypassing scheme has been proposed in to reduce the amount of the memory access by utilizing the characteristic of the LDPC parity check matrix and the decoding algorithm for the layered decoding architecture. However, the scheme proposed in may not lead to optimal reduction of memory access. In this work, we propose several schemes that lead to the optimal reduction of the memory access. Compared with the previous works, this work has the following contributions. Firstly by de-coupling the read and write access order of the memory storing the soft posterior reliability values (we denote it as the Channel RAM), optimal amount of memory bypassing is achieved and the number of idle clock cycles is reduced. Secondly, we are not only considering memory bypassing between consecutive layers of decoding but extending it to multiple layers. By doing so, the number of memory bypass is maximized. Thirdly, the decoding order of the layers has significant impact on the amount of memory bypass that can be achieved. We formulate the problem of finding the optimal decoding order given a parity check matrix as a searching problem and propose algorithm to obtain the optimal decoding order. Fourthly, based on the decoupling of the read and write order of the Channel RAM, we propose a memory efficient architecture, in which the Channel RAM and the memory storing the intermediate data are merged into a single memory. By doing so, the overall area is reduced. The rest of the paper is organized as follows. Section I gives the background of LDPC decoding scheme, the traditional layered LDPC decoder architecture and the memory-bypassing scheme for the layered decoding architecture proposed in [10]. The proposed memory bypassing scheme is presented in Section III. A quick searching algorithm to find the optimal decoding order of the layers of the LDPC base matrix, which results in the maximum overlapping, is described in Section IV. The overall decoder architecture implementing the memory-by-passing scheme is described in Section V. In Section VI, experimental results and comparisons among different LDPC decoders are presented. Conclusions are drawn in Section VII. comparing with other architectures. However how to further reduce its energy consumption is still a challenging design problem. Due to the large amount of memory access, the power consumption of the memory access accounts for the major part of the total power consumption of the layer decoder . Reducing the energy consumption of the memories is the key issue to realize a low energy LDPC decoder. At the algorithmic level, the Min-sum decoding algorithm and its variants have been proposed, which greatly reduces the memory storage required for the check to variable messages , and also the energy consumption of the memories of the LDPC decoder with insignificant performance loss. At the architectural level, memory-bypassing scheme has been proposed in to reduce the amount of the memory access by utilizing the characteristic of the LDPC parity check matrix and the decoding algorithm for the layered decoding architecture. However, the scheme proposed in may not lead to optimal reduction of memory access. In this work, we propose several schemes that lead to the optimal reduction of the memory access. Compared with the previous works, this work has the following contributions. Firstly by de-coupling the read and write access order of the memory storing the soft posterior reliability values (we denote it as the Channel RAM), optimal amount of memory bypassing is achieved and the number of idle clock cycles is reduced. Secondly, we are not only considering memory bypassing between consecutive layers of decoding but extending it to multiple layers. By doing so, the number of memory bypass is maximized. Thirdly, the decoding order of the layers as significant impact on the amount of memory bypass that can be achieved. We formulate the problem of finding the optimal decoding order given a parity check matrix as a searching problem and propose algorithm to obtain the optimal decoding order. Fourthly, based on the decoupling of the read and write order of the Channel RAM, we propose a memory efficient architecture, in which the Channel RAM and the memory storing the intermediate data are merged into a single memory. By doing so, the overall area is reduced. The rest of the paper is organized as follows. Section I gives the background of LDPC decoding scheme, the traditional layered LDPC decoder architecture and the memory-bypassing scheme for the layered decoding architecture proposed in [10]. The proposed memory bypassing scheme is presented in Section III. A quick searching algorithm to find the optimal decoding

order of the layers of the LDPC base matrix, which results in the maximum overlapping, is described in Section IV. The overall decoder architecture implementing the memory-by-passing scheme is described in Section V. In Section VI, experimental results and comparisons among different LDPC decoders are presented. Conclusions are drawn in Section VII.comparing with other architectures. However how to further reduce its energy consumption is still a challenging design problem. Due to the large amount of memory access, the power consumption of the memory access accounts for the major part of the total power consumption of the layer decoder . Reducing the energy consumption of the memories is the key issue to realize a low energy LDPC decoder. At the algorithmic level, the Min-sum decoding algorithm and its variants have been proposed, which greatly reduces the memory storage required for the check to variable messages , and also the energy consumption of the memories of the LDPC decoder with insignificant performance loss. At the architectural level, memory-bypassing scheme as been proposed in to reduce the amount of the memory access by utilizing the characteristic of the LDPC parity check matrix and the decoding algorithm for the layered decoding architecture. However, the scheme proposed in may not lead to optimal reduction of memory access. In this work, we propose several schemes that lead to the optimal reduction of the memory access. Compared with the previous works, this work has the following contributions. Firstly by de-coupling the read and write access order of the memory storing the soft posterior reliability values (we denote it as the Channel RAM), optimal amount of memory bypassing is achieved and the number of idle clock cycles is reduced. Secondly, we are not only considering memory bypassing between consecutive layers of decoding but extending it to multiple layers. By doing so, the number of memory bypass is maximized. Thirdly, the decoding order of the layers as significant impact on the amount of memory bypass that can be achieved. We formulate the problem of finding the optimal decoding order given a parity check matrix as a searching problem and propose algorithm to obtain the optimal decoding order. Fourthly, based on the decoupling of the read and write order of the Channel RAM, we propose a memory efficient architecture, in which the Channel RAM and the memory storing the intermediate data are merged into a single memory. By doing so, the overall area is reduced. The rest of the paper is organized as follows. Section I gives the background of LDPC decoding scheme, the traditional layered LDPC decoder architecture and the memory-bypassing scheme for the layered decoding architecture proposed in [10]. The proposed memory bypassing scheme is presented in Section III. A quick searching algorithm to find the optimal decoding order of the layers of the LDPC base matrix, which results in the maximum overlapping, is described in Section IV. The overall decoder architecture implementing the memory-by-passing scheme is described in Section V. In Section VI, experimental results and comparisons among different LDPC decoders are presented. Conclusions are drawn in Section VII.

$$H \bullet x^T = 0 \quad \forall x \in C. \tag{1}$$

The code can also be described by means of a bipartite graph, known as Tanner graph (Fig. 1(b)) [6]. A Tanner graph is made up of two entities, variable nodes (VN) and check nodes (CN), and they are connected to each other through a set of edges. An edge links the check node m to the variable node n if the element $H_{m,n}$ of the parity check matrix is non-null. The optimal LDPC decoding is achieved by using a message passing algorithm, also known as "belief propagation" (BP) , which can be described as an iterative exchange of messages along the edges of the Tanner graph . The algorithm proceeds iteratively until a maximum number of iterations are elapsed or a stopping rule is met. Inputs of the algorithm are the intrinsic Log-Likelihood Ratios (LLRs) of the received bits (i.e., the vari-able nodes), also referred to as *a priori* information.

The following describes the belief propagation algorithm. is $R_{m,n}^{(q)}$ the check-to-variable message from check node m to variable node in the q th iteration. $Q_{m,n}^{(q)}$ is the variable-to-check message from variable node n to check node m in the qth iteration. Mn is the set of the neighboring check nodes of variable node n, and Nm is the set of the neighboring variable nodes of check node m. In the qth iteration, the variable node process and the check node process are computed as follows.

Variable node process: the variable node n receives the messages $R_{m,n}^{(q)}$ from the neighboring check nodes and propagates back the updated messages $Q_{m,n}^{(q)}$ as

$$Q_{m,n}^{(q)} = \lambda_n + \sum_{i \in \{M_n \backslash m\}} R_{i,n}^{(q)} \tag{2}$$

where λ_n is the intrinsic LLR of the variable node n . At the same time, the posterior reliability value, also referred to as the soft output for the variable node , is given by

$$\Lambda_n^{(q)} = \lambda_n + \sum_{i \in \{M_n\}} R_{i,n}^{(q)}. \tag{3}$$

Check node process: the check node m combines together messages $Q_{m,n}^{(q)}$ from the neighboring variable nodes to compute the updated messages $R_{m,n}^{(q+1)}$, which are sent back to the corresponding variable nodes. Updates are performed separately on signs and magnitudes as follows:

$$-\mathrm{sgn}(R_{m,n}^{(q+1)}) = \prod_{j \in \{N_m \backslash n\}} -\mathrm{sgn}(Q_{m,j}^{(q)}) \tag{4}$$

$$\left|R_{m,n}^{(q+1)}\right| = \Phi^{-1}\left\{\sum_{j \in \{N_m \backslash n\}} \Phi\left(\left|Q_{m,j}^{(q)}\right|\right)\right\} \tag{5}$$

where

$$\Phi(x) = \Phi^{-1}(x) = -\ln\left(\tanh\left(\frac{x}{2}\right)\right). \tag{6}$$

The layered decoding scheduling improves the convergence speed and reduces the number of iteration by viewing the parity check as a sequence of

check through horizontal or vertical layers. The intermediate updated messages are used in the up-dating of the next layer. The layered decoding principle for hor-izontal layers is expressed by [14]

$$-\mathrm{sgn}\left(R_{m,n}^{(q+1)}\right) = \prod_{j\in\{N_m\setminus n\}} -\mathrm{sgn}\left(\Gamma_{m,j}^{(q+1)}\right) \quad (7)$$

$$\left|R_{m,n}^{(q+1)}\right| = \Phi^{-1}\left\{\sum_{j\in\{N_m\setminus n\}} \Phi\left(\left|\Gamma_{m,j}^{(q+1)}\right|\right)\right\} \quad (8)$$

and

$$\Gamma_{m,n}^{(q+1)} = \Lambda_n^{(q+1)}[k-1] - R_{m,n}^{(q)} \quad (9)$$

$$\Lambda_n^{(q+1)}[k] = \Gamma_{m,n}^{(q+1)} + R_{m,n}^{(q+1)} \quad (10)$$

where $k \in \{1,2,\ldots,dv\}$ is the order of the check node up-dated in an iteration dv and is the variable node degree. After an iteration is finished, the posterior reliability values after the last check update will be used as $\Lambda_n^{(q)}[k]$ and $\Lambda_n^{(q+1)}[k]$ in the following iteration. Equations (7)–(10) are derived by merging the variable node process and the soft-output updating process (2)–(3) with the CN update process (4)–(5). The variable node process is spread on the check node updating and the posterior a $\Lambda_n^{(q+1)}$ value , is refreshed after every check node update. The key advantage of the layered schedule is that the immediate up-dates of the posterior messages are used and propagated to the next layers for its updating within the iteration [14], which in-creases the convergence speed and reduces the average number of iteration by up to 50% [9].

The computation of (6) and (8) are complicated and difficult for hardware implementation. Low complexity algorithms such as min-sum approximation have been proposed to reduce the computation complexity [24]. In the min-sum decoding algorithm, the computation of (8) is approximated and expressed by

$$\left|R_{m,n}^{(q+1)}\right| = \min_{j\in\{N_m\setminus n\}} \left|\Gamma_{m,j}^{(q+1)}\right|. \quad (11)$$

Thus for a check node to compute the magnitudes of the outgoing messages, only two of the incoming messages with the smallest magnitudes have to be determined. The computa-tion complexity of (8) is significantly reduced. Furthermore, the storage of the outgoing messages has been reduced to only two as opposed to dc, where dc stands for the check node degree (i.e., the number of the neighboring variable nodes of a check node), because dc-1 variable nodes share the same outgoing message [24]. In order to achieve better performance and maintain the similar computation complexity and storage requirement of the min-sum approximation, the variants of the min-sum, such as offset min-sum [16], [17] and two-output approximation [15], have also been proposed and adopted in the hardware design.

B. The Layered Decoding Architecture

The layered decoding algorithm has been adopted in many designs [9]–[18] due to their high convergence speed and easy adaptation to the flexible LDPC codes. In this section, the-

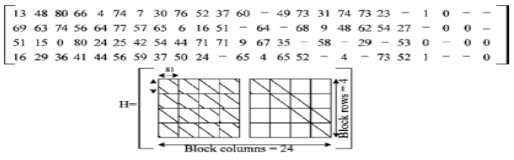

Fig. 2. Base matrix for a rate 5/6 with sub-block size of 81 LDPC code

defined in IEEE 802.11n.decoder architecture with layered decoding algorithm for the type of the architecture-aware LDPC codes (AA-LDPC) [9] is briefly introduced. Architecture-aware codes were proposed to facilitate the hardware design of the decoder. They are structured codes, whose parity-check matrix is built according to specific patterns. They are suitable for VLSI design, because the interconnection of the decoder is regular and simple and the trade-off between throughput and hardware complexity can be easily made. Since they support an efficient partial-parallel hardware VLSI implementation, AA-LDPC codes have been adopted in several modern communication standards, such as DVB-S2 [3], IEEE 802.16e and IEEE 802.11n [5]. Fig. 2 shows an example of such parity-check matrix which is a LDPC code defined in IEEE 802.11n . It s of rate 5/6 with sub-block size (i.e., the size of the identity sub-matrix) of 81. The parity-check matrix is composed of null sub-matrix or identity sub-matrix with different cyclic shifts. The numbers stand for the cyclic shift value of the identity sub-matrix, and the "–" stands for null sub-matrix. Several VLSI architectures have been proposed for he de-coder of these systems ,and layered decoding algo-rithm is commonly adopted in the design. A block diagram of these decoders is shown in Fig. 3 [13]–[19]. In the decoder, mul-tiple soft-in soft-out (SISO) units work in parallel to calculate multiple check node process for a layer. The Channel RAM is used to store the input LLR value of the received data initially. During the iteration of the decoding, it is used to store the pos-terior reliability values of the variable nodes. The shifter is used to perform the cyclic shift of the soft output messages so that the correct message is read out from the Channel RAM and sent to the corresponding SISO for calculation based on the base a-trix. The Sub-array is used to perform the subtraction of (9), and the results will be sent to the SISO unit and the memory used to store these intermediate results (i.e., FIFO in Fig. 3), at the same time. The SISO unit performs the check node process of (7) and (8). The two-output approximation [15] is used for the SISO computation, and two outgoing magnitudes are generated for a check node. One is for the least reliable incoming variable node, and the other is for the rest of the variable nodes. Thus, the SISO unit, for every check node, will generate the signs for the outgoing messages of all the variable nodes, two magnitudes and an index [23]. The index is used to select the two magnitudes for the update process in the Add-array. The data generated by the SISO will be stored in the Message RAM. The Add-array performs the addition of (10), by taking the output of the SISO and the intermediate results stored in the FIFO. The results of the Add-array will be written back to the Channel RAM. To increase the throughput, pipeline operation of the decoder is adopted in the design [15], [16]. To reduce the power consumption of the Channel RAM for a layered

decoding architecture, memory-bypassing scheme has been proposed in [10] by reducing the amount of memory access of the Channel RAM, which is briefly introduced in Section II-C.

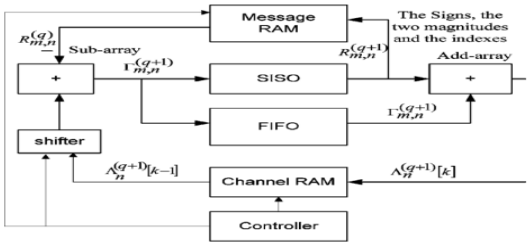

Fig. 3. Block diagram of the layered LDPC decoder.

C. The Memory Bypassing Scheme

In the layer decoding architecture, during the decoding, for every layer, the soft messages are read from and wrote into the Channel RAM and the FIFO every cycle. The Channel RAM stores the soft posterior reliability values of the variable nodes. The updated values are stored back from the Add-array and will be used in the decoding of the subsequent layer. In the memory-bypassing scheme [10], when two consecutive layers have non-null entry at the same column, the results of the Add-array can be directly sent to the cyclic shifter and used for the decoding of the next layer without the need of storing the in-termediate result in the Channel RAM. The memory bypassing scheme saves the write operation for the current layer and the read operation for the next layer. Fig. 4 shows an example. Fig. 4(a) shows a base matrix with three layers and Fig. 4(b) shows the timing diagram of the pipeline. Without any memory bypassing, the number of read and write access of the Channel RAM is equal to the non-null entries in the matrix. In this ex-ample, the total number of read and write operation is 12. If by-passing scheme is employed, instead of writing back the channel RAM, the updated soft output values are used directly for the decoding of the next layer, the number of memory access is re-duced. For example memory access for columns 0 and 2 can be bypassed when the decoding proceeds from layer 0 to layer 1; memory access for columns 0 and 1 can be bypassed for the second layer decoding and memory access for columns 0 and 3 can be bypassed for the third layer decoding. Thus, 6 out of the 12 read and write operations are eliminated, and 50% of
the power consumption of the Channel RAM can be saved. Assume the pipelined architecture takes two clock cycles for the cyclic shifter, Sub-array, the SISO and the Add-array to finish the computation after the last incoming variable node is read in, a detail timing diagram showing the operation of the de-coder is shown in Fig. 4(c). The order of read and write of the Channel RAM is following the natural order stated in the base matrix. Fig. 4(d) shows the memory bypassing scheme intro-duced in [10]. We denote it as simple memory bypassing. Here column 0 and 2 are written earlier for layer 0 and columns 0 and 2 are scheduled later for layer 1 so that overlap can be achieved. There exist some column orders that can result in max-imum overlap between two layers. However, for the subsequent layers the memory verlap may not be optimized. Due to data dependency, the memory write of a certain column for the ex-isting layer should finish before or at the same time with the reading of the same column for the subsequent layer. In order to achieve that, the decoding of the third layer has to be delayed to align the memory access and idling cycles are inserted in the decoding pipeline. We need to add idling cycle in order to maxi-mize the overlap and even that there is still one potential overlap (W3, R3) in the third layer that cannot be achieved. From that we can see that the simple memory bypassing scheme cannot realize all the potential memory bypass operation available for the decoding matrix or extra idling cycles are needed to be in-serted. In Section III, we will present an improved memory by-passing scheme which can achieve memory bypass for all the overlapped columns.

II. AN IMPROVED MEMORY BYPASSING SCHEME

In order to achieve the optimal amount of memory bypassing, we de-couple the read and write order of the Channel RAM and the memory storing the intermediate messages (i.e., FIFO in the traditional design) for a layer. This is shown in Fig. 5(a). We can see that all the potential bypassings are achieved and at the same time the idling cycles are minimized. As the messages read out from the Channel RAM will be stored in the intermediate data RAM (i.e., FIFO in the traditional design) after the subtraction check node update, the read order of the Channel RAM is the write order of the intermediate data RAM and the read order of the intermediate data RAM is the write order of the Channel RAM. The access sequence of the intermediate data RAM is shown in Fig. 5(b). In the above example, we limited our discussion to the fol-lowing assumptions.

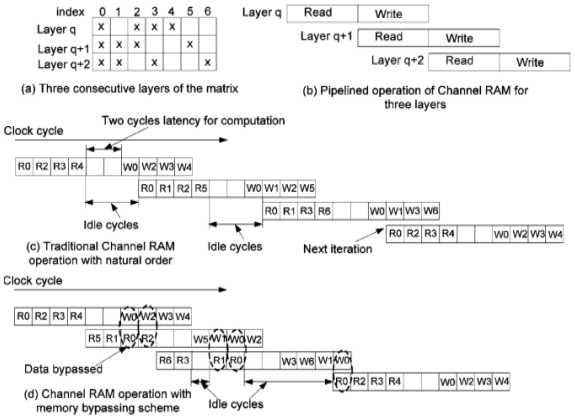

Fig. 4. The memory bypassing operation for the Channel RAM in the layered LDPC decoder.

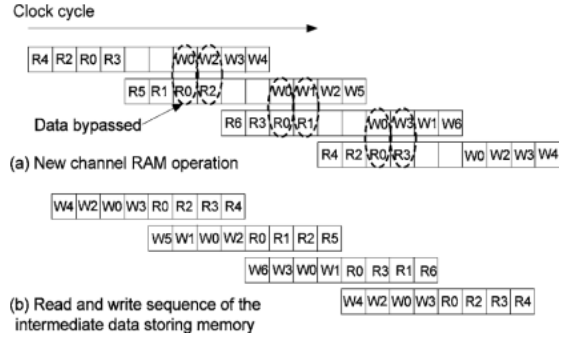

Fig. 5. Memory operations with different read and write order for the matrix shown in Fig. 4. and will be read out from the intermediate data RAM for the

1) Only overlapping between two consecutive layers are con-sidered for the memory bypassing.

2) The number of the latency cycles is 2. i.e., it takes two clock cycles for the cyclic shifter, Sub-array, the SISO and the Add-array to finish the computation after the last incoming variable node is read in. To optimize the number of memory bypass, we can consider the overlapping of more layers. For example, in Fig. 4(b), the first layer and the third layer have non-null entry at column 3, and this overlapping can be used for memory bypassing. The memory operations considering the overlapping of the three consecutive layers are shown in Fig. 6. For this example, if we also consider the overlapping of the first and the third layer, two more memory access can be by-passed (i.e.,the write operation W3 in the first layer and W2 in the second layer can be bypassed with the read operation R3 in the third layer and R2 in the first layer of the next de-coding iteration.)Considering the overlapping of three consec-utive layers, the memory-bypassing operation can be divided into two cases: memory-bypassing between layers and q+2.

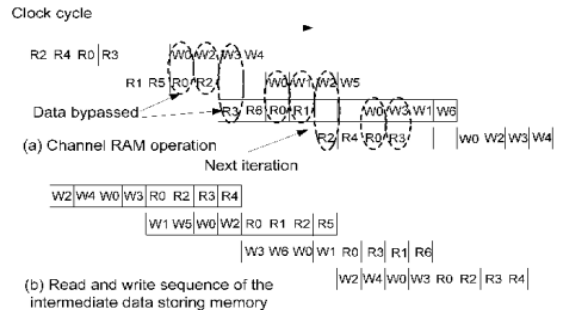

Fig. 6. Memory operations by considering the overlapping of three consecutive layers for the matrix shown in Fig. 4.

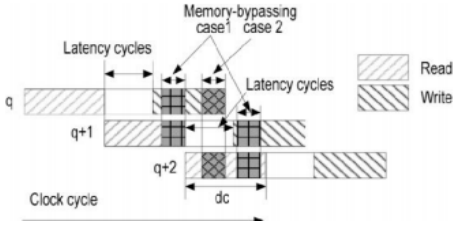

Fig. 7. Channel RAM operation when considering the overlapping among three consecutive layers.

$q+1$, memory-bypassing between layers $q+2$ and q . The potential number of memory-bypassing that can be achieved for layer $q+2$ is the number of the non-null entry that the layer $q+2$ are in common with the above two layers. However, this amount of the memory-bypassing may not always be achieved. The amount of memory -bypassing between layer $q+2$ and layer q (i.e., case2 shown in Fig.7)is limited by the latency cycles. When the latency cycles are smaller than the number of the non-null entry that the current layer $q+2$ are in common with the layers q but not in common with layer $q+1$,some of the memory operation for the overlapped columns cannot be bypassed. In this case, to increase the amount of memory-bypassing operation, the latency clock cycles have to be increased. We limit the memory-bypassing to three consecutive layers. Since the check node degree is in general larger than the latency cycles, the write operation of layer q will finish earlier than the read operation of the layer $q+3$ unless we insert many latency clock cycles. Thus, the memory-bypassing of four or more layers is not considered.

The amount of the overlapped columns indicates the number of memory-bypassing operation that we can achieve. The amount of the overlap directly depends on the order of the layer decoding. To achieve the optimal amount of memory bypassing, the optimal decoding order has to be determined, and also the non-null entries in every layer has to be properly scheduled such that the memory bypassing can be applied for every overlapped column. In Section IV, the algorithm for finding the

TABLE I
TOTAL NUMBER OF THE OVERLAPPED COLUMNS WHEN CONSIDERING THE OVERLAPPING OF TWO CONSECUTIVE LAYERS FOR THE IEEE 802.11N LDPC CODES

Order \ Rate	1/2	2/3	3/4	5/6
Best case order	49	55	59	65
Natural order	48	50	59	65
Worst case order	30	42	49	63

TABLE II
TOTAL NUMBER OF THE OVERLAPPED COLUMNS WHEN CONSIDERING THE OVERLAPPING OF THREE CONSECUTIVE LAYERS FOR THE IEEE 802.11N LDPC CODES

Order \ Rate	1/2	2/3	3/4	5/6
Best case order	61	67	70	78
Natural order	53	65	67	76
Time required (second)	68550.28	4.02	0.19	0.04

Optimal decoding order and the scheduling algorithm for the non-null entries inside each layer will be presented.

III. THE QUICK SEARCHING ALGORITHM

To obtain the optimum decoding order, we can use brute-force method to list out all the permutation orders of the layers and then compute the number of overlapped columns of each per- mutation. A few experiments have been done for the example codes defined in IEEE 802.11n. The results are shown in Tables I and II which also shows the time required for the calculation. Table I shows the results when we only consider the overlap- ping of two

consecutive layers. We also compare the number of overlapped column obtained using different decoding orders, namely the best, the natural and the worst order. Table II shows the corresponding results when we consider the overlapping of three consecutive layers. From the results we can see that the total number of overlapped columns when considering the over- lapping of three consecutive layers using the optimal order is in- creased by 12.9%–19.3% and the memory access of the Channel RAM is reduced by the same amount.

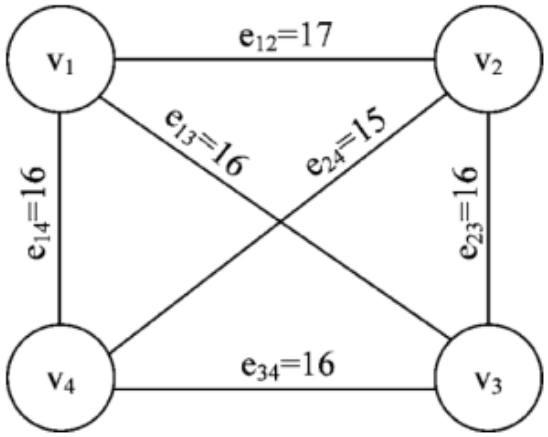

Fig. 8. Undirected graph + ₃ ₃₃₃ Γ₃ for the rate 5/6 code LDPC code shown

In Fig. 2.Table II shows that for the codes with small number of the layers, the brute-force method is still practical. However, when the base matrix becomes larger, the time required for the cal-culation will increase dramatically. It is infeasible to use the brute-force method to find the best order of the layers for the base matrix with large number of layers. For example, the LDPC codes defined in DVB-S2 have 180 layers! The complexity of the brute- ( )force method is and will quickly become imprac-tical when the n number increases. In order to reduce the computation time for the searching al-gorithm, the problem of finding the optimal decoding order of the layers which has the maximum amount of overlapping is an-alyzed. A quick searching algorithm is proposed and the esults for a few LDPC codes are presented.For better illustration, the algorithm finding the best order of the layers having the max-imum amount of overlapped columns between two consecutive layers is considered first.We denote it as two-layer overlapping. It can be formulated as a graph problem. Let be complete graph where V and E are the set of the nodes and edges of the graph, respectively. A node represents row of the base matrix and the cost of edge is the number overlap between rows and . The problem of finding the optimal orders is the same as that finding the path starting from any of the node in G, visiting all the other nodes exactly once and returning back to the starting node and that summation of costs of the edges of the path is maximum. Fig. 8 shows the graph representing the rate 5/6 code of IEEE 802.11n. The problem is equivalent to the well known traveling salesman problem (TSP) which is known to be NP-hard. For problems that have small number of nodes, such as the one shown in Fig. 8, brute force approach can be used to find the optimal solution. For problem with large size, heuristic algorithm is needed to generate a good solution. The problem of finding the optimal order of the layers that considers overlapping among more than two consecutive layers is similar to the two layer overlapping problem except that the calculation of the cost function is more complicated. Hence, the computation complexity is also NP-hard and heuristic algorithm is required to find a good solution for problem with large value of n. In this work, we used simulated annealing [28] to imple-ment the heuristic algorithm for the two-layer-overlapping and three-layer-overlapping problems. The cost function used in the

TABLE III
TOTAL NUMBER OF THE OVERLAPPED COLUMNS FOR THE IEEE 802.16E LDPC CODES

802.16e Sub-block size 96	Two layer		Three layer	
	Natural order	Solution found by simulated annealing	Natural order	Solution found by simulated annealing
1/2	23	38	35	50
2/3 A	35	42	55	59
2/3 B	9	52	73	73
3/4 A	51	51	70	75
3/4 B	55	62	75	75
5/6	67	67	77	79

TABLE IV
TOTAL NUMBER OF THE OVERLAPPED COLUMNS FOR THE IEEE DVB-S2 LDPC CODES

DVB-S2 Sub-block size 360	Two layer		Three layer	
	Natural order	Solution found by simulated annealing	Natural order	Solution found by simulated annealing
1/4	9	146	25	176
1/3	14	157	48	196
2/5	26	170	55	221
1/2	24	139	42	182
3/5	70	185	123	265
2/3	34	131	71	183
3/4	47	136	106	197
4/5	71	130	113	194
5/6	90	148	146	224
8/9	54	86	106	145
9/10	55	97	113	159

simulated annealing is the number of the overlapped columns for the base-matrix for a particular order of the layers. For each move, a pair of layers is randomly selected and their orders in the decoding sequence are swapped. The cost function is up-dated for the new sequence. The new sequence is accepted or rejected accordingly to the annealing schedule. Fig. 9 shows the detail flow diagram of the simulated

annealing algorithm used in this work. We applied the algorithm on different LDPC codes, including codes for IEEE 802.11n, IEEE 802.16e, and DVB-S2. The ex-perimental results are summarized in Tables III–IV. For small LDPC codes such as the codes for IEEE 802.11n and IEEE 802.16e, the simulated annealing algorithm always converges to the optimal solution. For large LDPC codes, such as the codes used in DVB-S2, the simulated annealing gives solutions that re-sult in significant increases in overlapping. Table III shows that for the codes used in IEEE 802.16e [4], 65.8%–98.8% of the ac-cess of the posterior reliability values in the Channel RAM can be bypassed. Table IV shows that for the codes used in DVB-S2 [3], 30.9%–65.9% of the access of the posterior reliability value in the Channel RAM can be bypassed. It can be seen that con-sidering overlapping among more layers and using optimum decoding order have a significant effect on the increase in the number of memory bypassing. In order to implement the memory bypassing scheme, after determining the optimal decoding order of the layers, the order of the column decoding of each layer has to be properly sched-uled. Since the read and write order of the Channel RAM for a layer are decoupled, they can be scheduled differently and in-dependently. As shown in Fig. 7, for the write order of layer , the non-null entries that the current layer are in common with the next layer are written first. The non-null entries that the layer are in common with the layer but not with the layer are written at the end. The other non-null entries are then scheduled in between randomly. From Fig. 7, we can also see that if the latency of the decoding data-path is zero, i.e., the SISO can start writing to the Channel RAM for a certain layer, e.g., layer , mmediately after all the inputs are read from the Channel RAM, then the memory bypassing between layers and cannot be done. It also means that if the number of overlapping between layers and is larger than the la- tency of the data-path, some of the potential memory bypassing cannot be realized. In this case, in order to achieve the maximum memory bypassing operation, we need to add idle clock cycles in the pipeline. When the latency is larger than the number of overlapping between layers and memory bypassing can always be achieved for all the overlapped columns without the need of inserting idle clock cycles. In Section V, the architecture of the LDPC decoder implementing the improved memory-by-passing scheme will be presented.

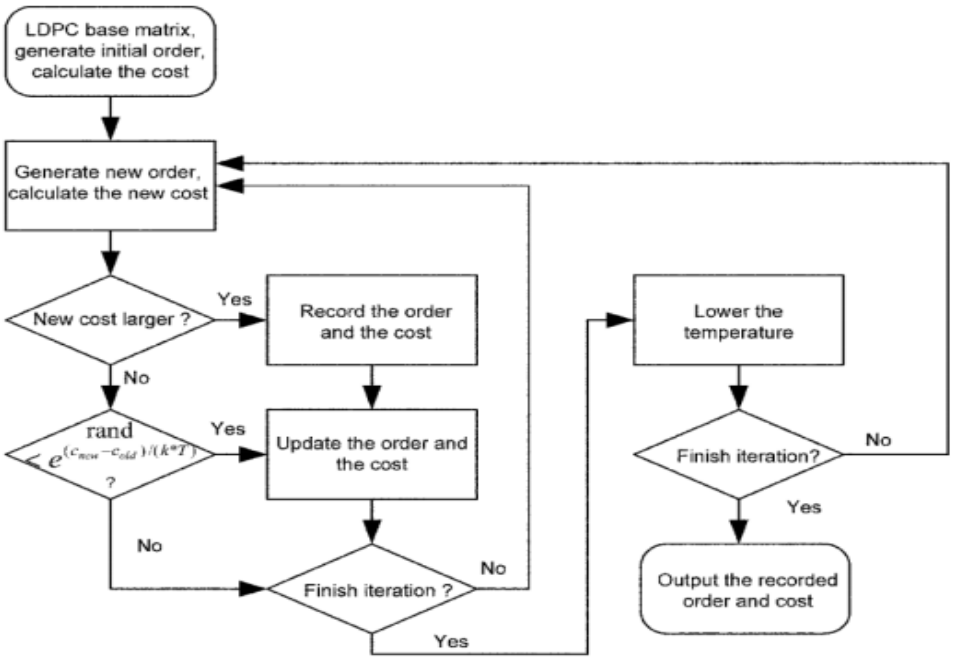

Fig. 9. The flow of the simulated annealing algorithm for finding the orderof the layers

IV. LDPC DECODER ARCHITECTURE IMPLEMENTING THE MEMORY BYPASSING SCHEME

The block diagram of the proposed LDPC decoder for IEEE 802.11n is shown in Fig. 10. Since the sub-block size is 81,

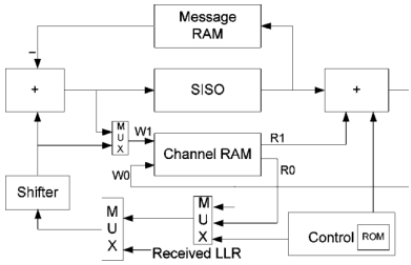

Fig. 10. Block diagram of the proposed LDPC decoder.

we use 81 soft-in soft-out (SISO) units working in parallel to calculate multiple check nodes processing for a layer at the same time. The operation of the shifter, the sub-array and the SISO is the same as the traditional layered decoding architec-tures [13]–[15]. The received messages are quantized into 5 bit and the bit-width of the soft output for the variable nodes is 6. The two-output approximation approach [15] is used to imple-ment the SISO. For every check node, two magnitudes, an index, and signs from all variable nodes are generated to represent the check node message and stored in the message RAM [23]. The message RAM is composed of one 88 88-bit dual port SRAM and 14 24 45-bit dual port SRAMs. The one 88 81-bit dual port SRAM is used to store the signs of the check node mes-sages, and the 14 24 bit dual port SRAMs are used to store the magnitudes and the indexes which represent the magnitudes of the check node messages. The decoder will stop decoding if the signs of the messages in one iteration during the e-coding satisfy all the parity checks or the number of iteration equals to a pre-defined value (In our design, the value is equal to 15). In order to minimize the memory access of the Channel RAM, the decoding order of the layers is determined by the al-gorithm introduced in the previous section. After determining the decoding order of the layers, the read and write order of the Channel RAM for the non-zero entries within a layer is sched-uled to achieve the memory bypassing for all the overlapped columns and minimize the idle cycles due to the data depen-dency of the layers. The read and write order of the intermediate data RAM is then fixed, as they are the same as the write and read order of the Channel RAM, respectively. In the traditional decoder, the FIFO and the Channel AM are implemented separately, because of the addressing of the FIFO and Channel RAM are different. By de-coupling the read and write order of the intermediate data RAM, the intermediate data RAM and the Channel RAM can be combined because the messages stored in the same address in the Channel RAM and the intermediate data RAM not needed at the same time. When the intermediate data RAM is storing the message at the location, the message stored at the location of the Channel RAM is not needed, and vice versa. Instead of using two separate memories, we use a single four port memory (two read ports and two write ports), to implement both the Channel RAM and the intermediate data RAM. By doing so, the area of the required memory is reduced. We de-note the combined memory as the new Channel RAM. The new Channel RAM is composed of 6 24 81 bit four port SRAMs and stores the input LLR values of the received data initially. Each entry of the new Channel RAM is dedicated to store the messages of the 81 variable nodes in the base-matrix. One pair of the read and write ports (R0 and W0) are used for the read and write access of the original Channel RAM and another pair of the read and write ports (R1 and W1) are used for the read and write access of the original intermediate data RAM. If the updated results will be used directly in the decoding of the fol-lowing layers, they will be sent to the shifter directly through a mux-array. The write port W0 and the read port R0 are dis-abled. Otherwise, the updated messages will be written into the new Channel RAM through the write port W0 and the messages needed in the next layer decoding are read out through the read port R0.

TABLE V COMPARISON CLOCK CYCLES REQUIRED PER ITERATION AND IDLE CYCLES IN PERCENTAGE

Rate		1/2	2/3	3/4	5/6
Cycle per iteration	Traditional design	158	135	121	102
	Memory bypassing	90	88	86	80
	Simple bypassing	133	120	107	91
	[18]	94	89	90	95
Idle cycle (%)	Traditional	45.7	34.8	29.8	22.5
	Memory bypassing	4.4	0.0	1.2	1.3
	Simple bypassing	35.3	26.7	20.6	13.2
	[18]	7.5	1.0	4.9	14.5

A bank of muxes is added to select the output of the Add-array and that of the Channel RAM and pipeline registers are added after the Add-array, to implement the memory bypassing scheme. Because the order of the messages entering the SISO (i.e., the read order of the read port R0) and the order of the messages updated in the Add-array (i.e., the read order of the read port R1) are different, the index generated in the SISO in-dicating the position of the least reliable incoming messages will be incorrect for the update process. A ROM containing the order of the updated process (i.e., read order of the read port R1) is added and it is used together with the index generated in the SISO to select the two magnitudes for the update process. The additional hardware is easy to implement and the overhead in area and power is very small. By de-coupling the read and write order of the Channel. RAM and using the memory bypassing scheme, the number of read and write access of the Channel RAM is reduced by 70.9%–

98.7% depending on the codes. At the same time, the idle cycles due to the data dependency of messages are minimized.

V. EXPERIMENTAL RESULTS

We implemented the proposed LDPC decoder to demonstrate the performance of the improved memory by-passing scheme. We also implemented the traditional layered decoding architec-ture [14], [15] and the LDPC decoder using simple memory bypassing scheme for comparison. For all the designs, the bit-width for the soft output messages is set to 6. The decoders were implemented and synthesized with Synopsys Design Com-piler using the Artisan's TSMC 0.18μm standard cell library. The power consumption of the embedded SRAM is character-zed by hspice simulation with the TSMC 0.18μm process. The power consumption of the decoder was simulated using Syn-opsys VCS-MX and Prime-Time. The supply voltage is 1.8V and the clock frequency is 250MHz. The first experiment compares the clock cycle required, the idle cycles and hence the throughput of the decoder. We also compared with the design in [18], which uses the reversal order of the read operation as the order of the write operation to re-duce the idle cycles. Table V summaries the results. The re-quired numbers of idle cycles reported in [18] are also included in Table V. Compared with the traditional decoder using the natural order (i.e., the order specified in the standard), the pro-posed decoder with the memory bypassing scheme decouples the read and write order of the memory and hence can reduce the number of idle cycles by 21.2%–41.3%. Compared with the simple memory bypassing scheme of which the read and write

TABLE VI ENERGY CONSUMPTION (PJ/BIT/ITERATION) OF THE THREE LDPC DECODERS WHEN OPERATED IN 250 MHZ

Rate		1/2	2/3	3/4	5/6
Message RAM	Traditional design	16.5	13.1	10.8	8.7
	Simple bypassing	16.5	13.1	10.8	8.7
	Our design	16.5	13.1	10.8	8.7
Logic units	Traditional design	62.3	54.6	49.4	43.9
	Simple bypassing	60.9	53.0	48.7	43.9
	Our design	57.7	50.0	47.5	43.8
Channel RAM + FIFO	Traditional design	64.7	66.6	65.5	62.4
	Simple bypassing	47.1	48.3	43.8	38.6
	Our design	40.4	40.2	38.4	32.7
Total	Traditional design	143.5	134.3	125.7	115.0
	Simple bypassing	124.5	114.4	103.3	91.2
	Our design	114.6	103.3	96.8	85.2

TABLE VII
COMPARISON THE DIFFERENT LDPC DECODER IMPLEMENTATIONS

	Memory bypassing LDPC decoder	Traditional LDPC decoder	IEEE 802.11n LDPC decoder [18]	IEEE 802.16e LDPC decoder [20]	TDMP LDPC Decoder [12]
Throughput	503 Mb/s (250MHz, 10 iter, synthesis result)	294.5Mb/s (250MHz, 10 iter., synthesis result)	356Mb/s (240MHz, 13~14 iter., synthesis result)	105Mb/s (150MHz, 20 iter.)	640Mb/s (125MHz, 10 iter.)
Area	2.67 mm^2 (1.26 mm^2 logic part)	2.67 mm^2 (1.24 mm^2 logic part)	0.74 mm^2 (0.45 mm^2 logic part)	6.25 mm^2	14.3 mm^2
Power	463 mW	339.5 mW	234.6 mW	264 mW	787 mW
Energy efficiency	92.05 (pJ/bit/iter.)	115.26 (pJ/bit/iter.)	—	125 (pJ/bit/iter.)	123 (pJ/bit/iter.)
CMOS technology	180 nm, 1.8V	180 nm, 1.8V	65 nm, 1.2V	90nm, 1.0V	180nm, 1.8V

order of the Channel RAM are not decoupled, the number of idle cycle is reduced by 11.9%–30.6%. Compared with the design in [18], the number of idle cycles is reduced by 1.0%–13.2%. The idle clock cycle in the decoder using the proposed memory by-passing scheme is only due to the irregular check node degrees. The idle clock cycles due to the data dependency, i.e., the up-dated message is computed before it can be used in another layer [21], [22] are all eliminated. The next experiment compares the power consumption of different decoders. Because the clock cycles required per iteration for the decoders are different, we used the energy efficiency of the decoders for comparison instead. The av-erage energy consumptions for decoding a block of data for different code rate modes are shown in Table VI. It can be seen that the decoder using the proposed memory bypassing scheme reduces the energy consumption of the channel RAM and FIFO by 37.6%–47.5%, comparing with the traditional

decoder. The corresponding reduction in energy is 27.2% to 38.1% when comparing with the traditional architecture using simple bypassing scheme. The overall energy reduction of the decoder is reduced by 20.1%–25.9% and 13.2%–20.7%. The reduction will be more for other LDPC codes such as DVB-S2 and IEEE 802.16e as the amount of memory bypassing that can be achieved by using the improved memory bypassing scheme is increased significantly as shown in Tables III and IV. Finally we compare the overall power consumption of the proposed decoder with other LDPC decoder implementations published in the literature. The comparisons are summarized in Table VII. Although it is hard to directly compare the different architectures since the LPDC codes are different and the imple-mentation technologies are also different, the comparison gives some idea on the energy efficiency of different designs.

VI. CONCLUSION

I have presented an improved memory-bypassing scheme to re-duce the memory access and hence the energy consumption of the LDPC decoder by exploiting the characteristic of the LDPC parity check matrix. Searching algorithm to find the optimal decoding order that results in maximum number of memory bypassing was proposed. The corresponding architecture sup-porting the proposed scheme was also presented. Experimental results show that optimum reduction in the memory access can be achieved for LDPC decoder targeting IEEE 802.11n specifi-cation and the memory access is reduced by 12.9%–19.3%. In addition, the idling cycle is reduced by 1.0%–13.2%, compared with the state-of-the-art design.

REFERENCES

[1] R. G. Gallager, “Low-density parity-check codes,” IRE Trans. Inf. Theory, vol. IT-8, pp. 21–28, Jan. 1962.

[2] D. J. C. MacKay, “Good error-correcting codes based on very sparse matrices,” IEEE Trans. Inf. Theory, vol. 45, no. 2, pp. 399–431, Mar. 1999.

[3] Digital Video Broadcasting (DVB); Second Generation Framing Struc- ture, Channel Coding and Modulation Systems for Broadcasting, Inter- active Services, News Gathering and Other Broadband Satellite Appli- cations 2004.

[4] LDPC coding for OFDMA PHY. 802.16REVe Sponsor Ballot Recircu- lation Comment 2004, IEEE C802.16e-04/141r2.

[5] Joint Proposal: High Throughput Extension to the 802.11 Standard: PHY. IEEE P802.11 Wireless LANs 2006, IEEE 802.11-05/1102r4.

[6] R. Tanner, “A recursive approach to low complexity codes,” IEEE Trans. Inf. Theory, vol. 27, no. 5, pp. 533–547, Sep. 1981.

[7] A. J. Blanksby and C. J. Howland, “A 690-mW 1-Gb/s 1024-b, rate 1/2 low-density parity-check code decoder,” IEEE J. Solid-State Circuits, vol. 37, pp. 404–412, Mar. 2002.

[8] J. Zhang and M. P. C. Fossorier, “Shuffled iterative decoding,” IEEE Trans. Commun., vol. 53, no. 2, pp. 209–213, Feb. 2005.

[9] M. M. Mansour and N. R. Shanbhag, “High-throughput LDPC de- coders,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 11, no. 6, pp. 976–996, Dec. 2003.

[10] E. Cavus, “Techniques for the decoding of low density parity check codes: efficient simulation, algorithm improvement and implementa- tion,” Ph.D dissertation, UCLA, Los Angeles, 2007.

[11] M. Rovini, F. Rossi, P. Ciao, N. L’Insalata, and L. Fanucci, “Layered decoding of non-layered LDPC codes,” in Proc. Euromicro Conf. Dig- ital Syst. Design, Aug.-Sep. 2006, pp. 537–544.

[12] M. Mansour and N. R. Shanbhag, “A 640-Mb/s 2048-bit pro-grammable LDPC decoder chip,” IEEE J. Solid-State Circuits, vol. 41, pp. 684–698, Mar. 2006.

[13] J. Dielissen, A. Hekstra, and V. Berg, “Low cost LDPC decoder for DVB-S2,” presented at the Design, Automation Test in Eur., Munich, Germany, Mar. 2006.

[14] T. Brack, M. Alles, T. Lehnigk-Emden, F. Kienle, N. Wehn, N. E. L’In- salata, F. Rossi, M. Rovini, and L. Fanucci, “Low complexity LDPC code decoders for next generation standards,” in Proc. Design, Automa- tion and Test in Europe, Apr. 2007, pp. 1–6.

[15] G. Gentile, M. Rovini, and L. Fanucci, “Low-complexity architectures of a decoder for IEEE 802.16e LDPC codes,” in Proc. Euromicro Conf. Digital Syst. Design Architectures, Methods Tools, Aug. 2007, pp. 369–375.

[16] Y. Sun, M. Karkooti, and J. R. Cavallaro, “VLSI decoder architecture for high throughput, variable block-Size and multi-rate LDPC codes,” in Proc. IEEE Int. Symp. Circuits Syst., May 2007, pp. 2104–2107.

[17] K. K. Gunnam, G. Choi, W. Wang, and M. B. Yeary, “Multi-rate lay- ered decoder architecture for block LDPC codes of the IEEE 802.11n wireless standard,” in Proc. IEEE Int. Symp. Circuits Syst., May 2007, pp. 1645–1648.

[18] M. Rovini, G. Gentile, F. Rossi, and L. Fanucci, “A scalable de- coder architecture for IEEE 802.11n LDPC codes,” in Proc. Global Telecommun. Conf., Nov. 2007, pp. 3270–3274.

[19] X. Y. Shih, C. Z. Zhan, C. H. Lin, and A. Y. Wu, “An 8.29 mm 52 mW multi-mode LDPC decoder design for mobile wimax system in 0.13 _m CMOS process,” IEEE J. Solid-State Circuits, vol. 43, no. 3, pp. 672–683, Mar. 2008.

[20] C. H. Liu, S. W. Yen, C. L. Chen, H. C. Chang, C. Y. Lee, Y. S. Hsu, and S. J. Jou, “An LDPC decoder chip based on self-routing network for IEEE 802.16e applications,” IEEE J. Solid-State Circuits, vol. 43, no. 3, pp. 684–694, Mar. 2008.