Low-Power Design of Variable Block-size LDPC Decoder using Nanometer Technology

[1]Chih-Hung Lin, [2]Alex Chien-Lin Huang, [1]Robert Chen-Hao Chang and [3]Kuang-Hao Lin
[1]Department of Electrical Engineering National Chung Hsing University, Taichung, Taiwan
[2]Design Service Department, National Chip Implementation Center, Hsinchu, Taiwan
[3]Department of Electronic Engineering, National Chin-Yi University of Technology, Taichung, Taiwan
ljh@ee.nchu.edu.tw, moonalex@icsl.ee.nchu.edu.tw, chchang@nchu.edu.tw and khlin@ncut.edu.tw

[1] ***Abstract*—This paper presents a low-power, variable block-size and irregular LDPC decoding. Our proposed LDPC decoder uses nanometer technology running the well-known TDMP and SMSA decoding algorithm. We further improved the design with pipeline structure, parallel computation and without any memory unit. Therefore, we can utilize only one routing network to route three different block-size data. The prototype architecture is being implemented on 90 nm VLSI technology. Because this VLSI technology has multi-Vth layers, we can make the design more effective. Compared to recent state-of-the-art architectures, the proposed variable block-size LDPC decoder has 450 MHz clock frequency, 349.48 K gate counts, 168 mW power dissipation, and 1.215 Gbps throughput.**

***Index Terms*—LDPC, Nanometer VLSI technology, IEEE 802.11n, TDMP**

I. Introduction

INCREASING demand of high data rate and reliability in modern communication systems is pushing next-generation standards toward error correction schemes allowing high throughput decoding with near Shannon limit performance. At present, low-density parity-check (LDPC) codes [1] are among the best candidates to meet these requirements. However, the first work on hardware implementation [2] pointed out the huge complexity associated to a LDPC decoder even for short-length codewords. The peculiarities of the decoding algorithm strongly affect traditional VLSI systems metrics (chip area, clock speed, and power consumption) making it difficult to meet feasible implementation requirements without spoiling communication performance [3].

With the advances in the VLSI technology, designers are able to use smaller cell with high clock rate and low power consumption. Recently, the nanometer VLSI technology has been adopted on Giga-scale communication baseband circuits and systems design. One of the designs is LDPC decoder. The nanometer VLSI technology offers low-k for manufacturing, giving designers the flexibility to choose the dielectric material best suited for their particular product application. Besides, the nanometer VLSI technology offers several different transistor options for its nanometer process to target a broad rang of semiconductor applications. Low leakage devices are offered for low-power, where power management is a priority. On the other hand, a high-speed option is available for critical path when utmost performance is needed.

All upcoming standards featuring the use of LDPC codes such as WiFi (IEEE 802.11n) [4], WiMax (IEEE802.16e) [5], 10GBASE-T (IEEE 802.3an) [6], and DVB-S2 [7], adopt architecture-aware LDPC codes. Despite the co-design approach, the need for a further reduction in complexity is still an appealing issue specially when coping with the variety of code lengths and rates exhibited by the above mentioned standards. A binary LDPC code is a binary linear block code that can be defined by a sparse binary parity-check matrix. A spare matrix is a matrix where only a small fraction of its entries are ones.

For any $m \times n$ parity-check matrix H, it defines a (n, k, j)-regular LDPC code if every column vector of H has the same weight j and every row vector of H has the same weight k. Here the weight of a vector is the number of ones in the vector. By counting the ones in H, it follows that $n \times j = m \times k$. Hence if $m < n$, then $j < k$. Suppose the parity-check matrix has full rank, the code rate of H is $r = (n-m)/n = (k-j)/k = 1 - j/k$. If not all the rows of the parity-check matrix H have the same number of ones, an LDPC code is said be irregular.

As suggested by Tanner [8], an LDPC code can be represented as a bipartite graph. An LDPC code corresponds to an unique bipartite graph and a bipartite graph also corresponds to an unique LDPC code. In a bipartite graph in Figure 1, one type of nodes, called the variable nodes, correspond to the symbols in a codeword. The other type of nodes, called the check nodes, correspond to the set of parity-check equations. If the parity-check matrix H were an $m \times n$ matrix, it would have m check nodes and n variable nodes. A variable node v_i is connected to a check node c_j by an edge, denoted as (v_i, c_j), if and only if the entry h_{ij} of H is one.

In this paper, we proposed a variable block-size LDPC decoder based on IEEE 802.11n specification [4]. The decoder should support three different block sizes at code rate $1/2$. The TDMP and SMSA (Scaling Min-Sum Algorithm) decoding algorithm are used in our design. We also use one data routing unit and no memory for hardware implementation that can increase the throughput and reduce power consumption. By introduced nanometer VLSI technology, we analyzed the design and use nanometer characteristics to improve the performance. A prototype of the decoder architecture is implemented in Verilog HDL and synthesized on UMC $90nm$ Multi-Vth

[1]This work was supported in part by the National Science Council, Taiwan. under Grant NSC NSC 97-2221-E-005-086-MY2 and in part by the Ministry of Education, Taiwan. under the ATU plan. The authors would like to thank the National Chip Implementation Center of Taiwan for technical support.

978-1-4244-5309-2/10/$26.00 ©2010 IEEE

$$H = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \end{bmatrix}$$

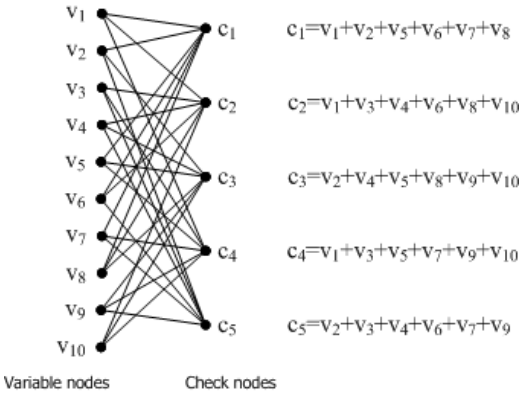

Figure 1. Example of a $(10, 3, 6)$-regular LDPC codes and its corresponding Tanner graph. There are 10 variable nodes (v_i) and 5 check nodes (c_j).

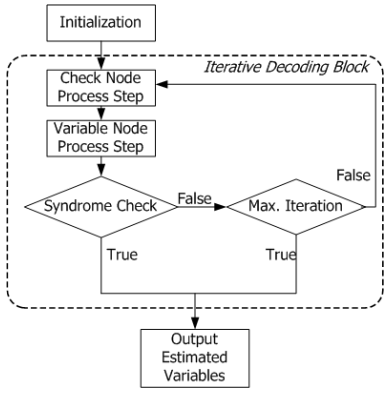

Figure 2. Iterative decoding flow chart for LDPC codes

VLSI technology. The logic synthesis report shows that the proposed algorithm not only can achieve required throughput that IEEE 802.11n specified but also have better area and power efficiency.

II. The LDPC Decoding algorithm

The LDPC can be decoded by Gallager's iterative two phase message passing algorithm (TPMP), which involves 4 steps: initalization, check node process step, variable node process step, and syndrome check, Figure 2 illustrates the iterative decoding flowchart of LDPC codes. The authors in [9] introduced the concept of turbo decoding message passing (TDMP, so-called layered decoding) using BCJR (named after its discoverers Bahl, Cocke, Jelinik, and Raviv) for their archiecture-aware LDPC (AA-LDPC) codes.

TDMP towards iterative decoding is based on decoding in layers. The parity-check matrix can be viewed as horizontal layers and each layer can represent a component code. The intersection of all these layers (codes) forms the full code.

Algorithm 1 The pseudo code of TPMP algorithm

//Initialization
$L(x_i)$ =Channel Values;
$L^{-1}_{v_i \to c_j}$ =Channel Values;
//Start iteration
for $l = 0$ to $l_{max} - 1$ do
//Check node process step
$\forall i \in c(G)$, $L^{l}_{c_i \to v_j} =$
$\beta \times \prod_{v_{i'} \in L(j) \backslash v_i} sign\left(L^{(l-1)}_{v_{i'} \to c_j}\left(e_{i'j}\right)\right) \times min\left(\left|L^{(l-1)}_{v_{i'} \to c_j}\right|\right)$;
//Vairable node process step
$\forall i \in c(G), L^{l}_{v_i \to c_j} = L\left(x_i\right) + \sum_{c_{j'} M(i) \backslash c_j}\left(e_{ij'}\right)$;
end for
$L^{post}\left(x_i\right) = L\left(x_i\right) + \sum_{c_j \in M(i)} L^{l}_{c_j \to v_i}$;

As each next layer starts decoding, its inputs are combined from the channel values and the extrinsic probability that is computed from the decoding on the last layer processed, or another prior layer if necessary. This overall process is repeated as many times as desired. Iterations within a layer via the decoder can be called sub-iterations and the overall process repetitions are labeled as iterations.

The layered approach to the scaling min-sum algorithm is given below. This is presented in a form easily implemented using time-overlapped column summation. In this approach, we divide the rows of H into G non-overlapping groups. The word non-overlapping indicates that each column of each group has weight of one at most. Denote the LLR of message passing form i^{th} check node to j^{th} variable node at the l^{th} iteration as $L^{l}_{c_j \to v_i}(e_{ij})$, and the opposite message as $L^{l}_{v_j \to c_i}(e_{ij})$. $L^{post}(x_i)^{(l \times G+g)}$is denoted as the a posteriori LLR of LLR of j^{th}vairable at the g^{th} sub-iteration of l^{th} iteration. $M(i)$ is the set of check nodes connected to variable node v_i and $L(j)$ is the set of variable nodes that associated with check node c_j, and $c(g)$ is the set of check nodes of group g. At the l^{th} iteration, the input of check node operation are the output of the last layer, with the most recently updated extrinsic message $L^{l}_{v_j \to c_i}(e_{ij})$ of variable nodes. After check node operation of this group, variable node update is calculated immediately. Since there is only one entry of each column at most, only one or none degree of each variable node has new input $L^{l}_{c_j \to v_i}(e_{ij})$. The $L^{(l-1)}_{c_j \to v_i}(e_{ij})$ of the last iteration will become a subtrahend and this new input will become an addend of the a posteriori value of the variable nodes. It can be organized as Algorithm 1 and Algorithm 2.

III. Proposed Decoder Architecture

A. Overall Decoder Architecture

In order to achieve the variable block-size design, we propose the reconfigure architecture. There are 5 components, that is, data route unit (DRU), variable node process unit (VNPU), check node process unit (CNPU), LLR calculation unit (LLR CU), and FIFO array. By using the pipeline technique, we separate 5 components into 4 stages. The pipeline technique can improve throughput but it needs lots of delay flip-flops (DFF). To reduce the hardware cost, we use the retiming

Algorithm 2 The pseudo code of TDMP algorithm

//Initialization

$L^{-1}_{c_j \to v_i}(e_{ij}) = 0$;

$L^{post}(x_i)^{-1}$ =Channel Values;

//Start iteration

for $l = 0$ to $l_{max} - 1$ do

//Start sub-iteration

for $g = 0$ to $G - 1$ do

$\forall i \in c(g), L^{l}_{v_i \to c_j}(e_{ij}) = L^{post}(x_i)^{(l \times G + g - 1)} - L^{(l-1)}_{c_j \to v_i}(e_{ij})$;

//Check node process step

$\forall i \in c(g), L^{l}_{c_i \to v_j} = \beta \times \prod_{v_{i'} \in L(j) \setminus v_i} sign\left(L^{(l-1)}_{v_{i'} \to c_j}(e_{i'j})\right) \times min\left(\left|L^{(l-1)}_{v_{i'} \to c_j}\right|\right)$;

$\forall i \in c(g), L^{post}(x_i)^{(l \times G + g)} = L^{l}_{v_i \to c_j} + L^{l}_{c_j \to v_i}$;

end for

end for

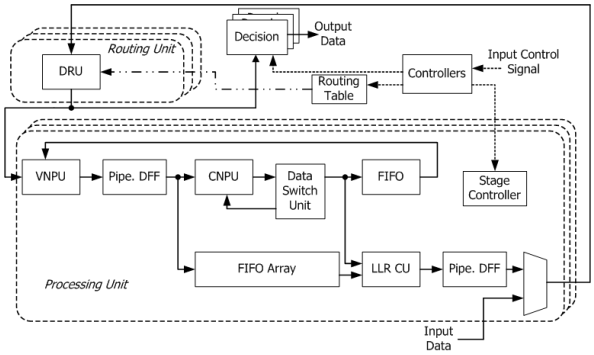

Figure 3. The improved architecture of our proposed design

technique to eliminate some pipeline's DFF. Figure 3 is the improved architecture.

In the upcoming standards featuring the use of LDPC codes such as WiFi (IEEE 802.11n) [4] has three different sizes of the sub-matrix to support multi-rate standard. In this paper, we use IEEE 802.11n LDPC specification to realize our proposed hardware architecture. In the IEEE 802.11n LDPC code, the size of the sub-matrix is 27, 54, or 81. Thus, the size of the smallest hardware unit that we proposed is 27. Because of the nanometer VLSI technology, we can increase the operation frequency and use the smallest hardware unit to route the data of sub-matrix 54 and 81.

B. Data Route Unit (DRU)

Because the IEEE 802.11n LDPC code has three different sizes of sub-matrix, the DRU must be flexible. Therefore, we use one data switch unit (DWU) , three FIFOs, and one FIFO select unit (SFU) to meet three different sizes of sub-matrix. The DWU has 27 inputs and 27 outpus; the functionality of DWU is to exchange two data bits at one cycle. And then, SFU will put data bits to the correct FIFO. Figure 4 is the architecture of DRU.

Because of the IEEE 802.11n LDPC code uses QC-LDPC, the sub-matrix is cyclic permutation. We only need one DRU

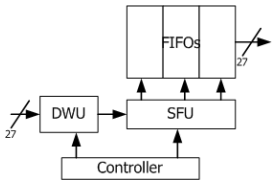

Figure 4. The architecture of DRU

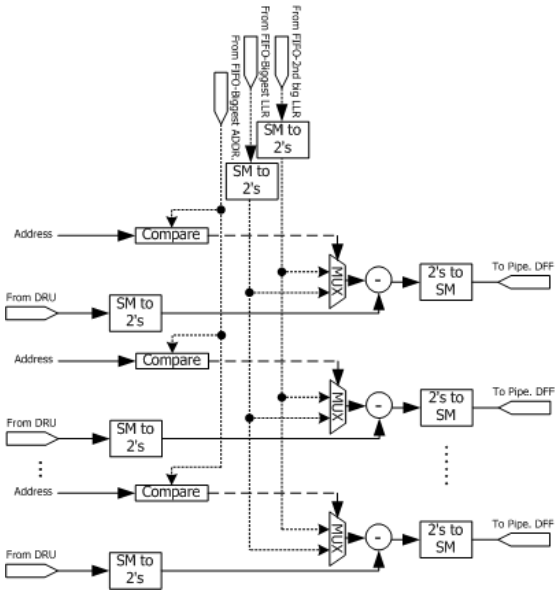

Figure 5. The illustration of VNPU and LLR CU architecture

to route the data by off-line re-calculating the route network, so that lots of routing network hardware is saved.

C. Variable Node Process Unit (VNPU) and LLR Calculation Unit (LLR CU)

Figure 5 is the illustration of VNPU and LLR CU architecture. We compare the address of previous iteration's biggest LLR values and the address of DRU input. If the address of previous iteration's biggest LLR is equal to the address of DRU data in, the multiplexer selects the second big LLR value to operation unit, otherwise the multiplexer selects the biggest LLR value. Because the format of all data is sign-magnitude, we convert the format back to SM (Sign-Magnitude). And the LLR CU is similar to VNPU, except the input data and mathematical operation.

D. Check Node Process Unit (CNPU)

In our proposed LDPC decoder, we just restore the biggest LLR value, 2nd big LLR value and the address of biggest LLR value at each layer. Therefore, we can save lots of memory or DFFs to restore the check node information. Figure 6 is the hardware architecture of CNPU. A CNPU contains 4 CMP4x2s (Comparator have 4 inputs and 2 outputs) and each CMP4x2 is constructed by two CMP2x2s. The CMP2x2 is a comparator of two values. In the CMP4x2, we use fully parallel architecture to shorten the critcal path of hardware.

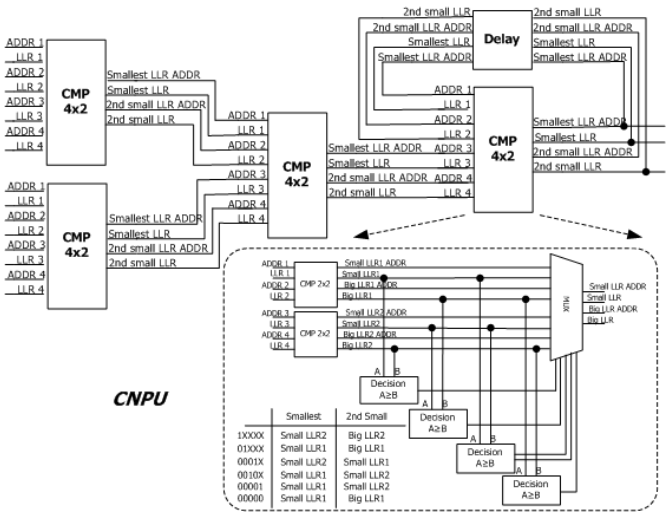

Figure 6. The hardware architecture of CNPU

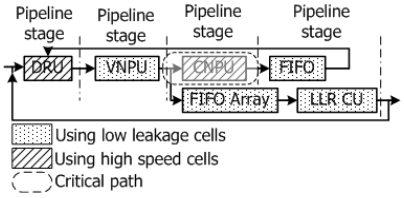

Figure 7. Critical path, pipeline and multi-threshold voltage design

E. Design Optimization and Nanometer VLSI technology

To speed up the clock frequency, we add pipeline DFF in every operations. Figure 7 shows the pipeline stages, critical path, and multi-threshold voltage design. It starts from the VNPU's pipeline DFF and ends at CNPU's data switch unit.

Low leakage devices are offered for non-critical path. On the other hand, a high-speed option is available for critical path when the performance is needed. Thus, we use multi-threshold voltage to enhance our design performance. At the critical path we use high-speed cells to make sure the time slack is meet. At the non-critical path we use low leakage cells to lower the power consumption. It results the design to have lower power consumption at same area cost and same clock frequency.

IV. Hardware implementation results

Table I gives the synthesis result of our proposed LDPC decoder. This table contains before optimization design and after optimization design where we can get better power consumption after optimization. The nanometer VLSI technology used is 90nm low-k multi-threshold voltage CMOS process.

Table II gives the hardware implementation results compared with some other decoders reported in recent papers. This throughput achieved is Gb/s at a maximum iteration of 10. The throughput value can meet the requirement of IEEE 802.11n standard.

Note that the area values given in Table II are after synthesis excluding the layout utilization factors. Compared with existing designs, our proposed LDPC decoder can get similar hardware efficiency plus some flexibility in coding rate.

Table I
Synthesis results

Module	Gate count	Numbers	Power w/o opt.	Power w opt.
DRU	3.76 K	27	2.56 mW	2.56 mW
VNPU	2.56 K	8	1.37 mW	1.26 mW
CNPU	3.31 K	27	2.33 mW	2.33 mW
LLR CU	2.70 K	8	1.42 mW	1.30 mW
TOP (Whole decoder)	349.48 K	-	177.51 mW	168.76 mW

Table II
Comparisons with other decoder architectures

	[10]	[11]	[12]	This paper
Code length	648~1944	648~1944	960	648~1944
Code rate	1/2~5/6	1/2~5/6	1/2	1/2
Quantization ($bits$)	5	6	-	6
Algorithm	TDMP	TPMP	TPMP	TDMP
Technology	65	65	90	90
Frequency (MHz)	240	400	100	450
Iterations	11~14	25~50	-	10
Throughput ($Mbps$)	178~140	53~281	-	1215
Area (mm^2)	0.74	1.024	3.4	1.11
Gate count (K)	355 (est.)	491 (est.)	825	349.48
Power dissipation (mW)	235	-	199.6	168.76

V. Conclusions

In this paper, we have presented a variable block-size LDPC decoder design for IEEE 802.11n codes. The decoder implements the TDMP decoding algorithm. By introducing a high parallel, pipeline and retime decoding schedule and multi-threshold voltage VLSI technology, the decoder can reach 1.215Gps throughput and support three different code lengths as well. The implementation result shows that our proposed design can be a competitive candidate among recent state-of-the-art designs.

References

[1] R. G. Gallager, *Low-Density Parity-Check codes*, Cambridge, MA: MIT Press, 1963
[2] A.J. Blanksby, C.J. Howland, "A 690-mW 1-Gb/s 1024-b, rate-1/2 low-density parity-check code decoder," *IEEE Journal of Solid-State Circuits*, vol. 37, pp. 404-412, March 2002.
[3] T. Brack, F. Kienle, N. Wehn, "Disclosing the LDPC code decoder design space," in *Proc. Design, Automation and Test in Europe*, March 2006, vol. 1, pp. 6.
[4] "IEEE 802.11n," ieeexplore.ieee.org.
[5] "IEEE Std 802.16e-2005," ieeexplore.ieee.org.
[6] "IEEE P802.3an, 10GBASE-T tack fore," ieeexplore.ieee.org.
[7] "DVB-S2," www.dvb.org.
[8] R. Tanner, "A recursive approach to low complexity codes" *IEEE Transactions Information Theory*, vol. 27, no. 5, pp. 533-547, Sep 1981.
[9] M.M. Mansour, N.R. Shanbhag, "A 640-Mb/s 2048-bit programmable LDPC decoder chip," *IEEE Journal of Solid-State Circuits*, vol. 41, no. 3, pp. 648-698, March 2006.
[10] M. Rovini, G. Gentile, F. Rossi, L. Fanucci, "A scalable decoder architecture for IEEE 802.11n LDPC codes," in *Proc. IEEE Global Telecommunications Conference*, Nov. 2007, pp. 3270-3274.
[11] T. Brack, M. Alles, T. Lehnigk-Emden, F. Kienle, N. Wehn, N.E. L'Insalata, F. Rossi, M. Rovini, L. Fanucci, "Low complexity LDPC code decoders for next generation standards," in *Proc. Design, Automation & Test in Europe Conference & Exhibition*, April 2007, pp.1-6.
[12] X. Shi, S. Yoshizawa, Y. Miyanaga, "Evaluation and implementation of quasi-cyclic LDPC codes for IEEE802.11n based MIMO-OFDM system," in P*roc. IEEE Conference on Soft Computing in Industrial Applications*, June 2008, pp. 227-280.