A LOW-POWER 1-Gbps RECONFIGURABLE LDPC DECODER DESIGN FOR MULTIPLE 4G WIRELESS STANDARDS

Yang Sun and Joseph R. Cavallaro
Department of Electrical and Computer Engineering
Rice University, Houston, TX 77005
Email: {ysun, cavallar}@rice.edu

Abstract— In this paper we present an efficient system-on-chip implementation of a 1-Gbps LDPC decoder for 4G (or beyond 3G) wireless standards. The decoder has a scalable datapath and can be dynamically reconfigured to support multiple 4G standards. We utilize a pipelined version of the layered belief propagation algorithm to achieve partial-parallel decoding of structured LDPC codes. Instead of using the sub-optimal Min-sum algorithm, we propose to use the powerful belief propagation (BP) decoding algorithm by designing an area-efficient soft-input soft-output (SISO) decoder. Two power saving schemes are employed to reduce the power consumption up to 65%. The decoder has been synthesized, placed, and routed on a TSMC 90nm 1.0V 8-metal layer CMOS technology with a total area of 3.5 mm^2. The maximum clock frequency is 450 MHz and the estimated peak power consumption is 410 mW.

I. INTRODUCTION

The approaching fourth-generation (4G) wireless systems are projected to provide 100 Mbps to 1 Gbps speeds by 2010, which consequently leads to orders of magnitude complexity increases in the wireless receiver SoC (System-on-Chip). As a core technology in wireless communications, FEC (forward error correction) coding has migrated from 2G convolutional/block codes to more powerful 3G Turbo codes, and LDPC (Low-density parity-check) codes [1] forecast for 4G systems because of their excellent error correction performance and highly parallel decoding scheme. To meet the power consumption constrains in wireless handsets, it is very challenging to design a flexible and high throughput LDPC decoder.

Most of the research on LDPC decoder design so far has focused on one particular system in which specific optimizations are made to improve the decoder performance. For example, authors in [2][3] discussed LDPC decoders for WiMax system, and authors in [4][5] presented custom designed, non-standard LDPC decoders. In this paper, we discuss a scalable and dynamically reconfigurable LDPC decoder targeting multiple 4G standards. For low power implementations, we introduce an early termination scheme and a distributed SISO decoding and memory banking scheme to reduce power consumption.

II. DECODING ALGORITHM

A binary LDPC code is a linear block code specified by a very sparse binary $M \times N$ parity check matrix: $H \cdot x^T = 0$, where x is a codeword ($x \in C$) and H can be viewed as a bipartite graph where each column and row in H represent a variable node and check node, respectively.

A. Block structured LDPC codes

Non-zero elements in H are typically placed at random positions to achieve good performance. However, this randomness is unfavorable for efficient VLSI implementation that calls for structured design. To address this issue, block-structured LDPC codes are recently proposed for several new communication standards such as IEEE 802.11n, IEEE 802.16e, DVB-S2 and DMB-T. As shown in Fig. 1, a block structured parity check matrix can be viewed as a 2-D array of square sub matrices. Each sub matrix is either a zero matrix or a cyclically shifted identity matrix I_x. Generally, a block structured parity check matrix H consists of a $j \times k$ array of $z \times z$ cyclically shifted identity matrices with random shift values x ($0 \leq x < z$). Table 1 summarizes the design parameters for H in several standards. In

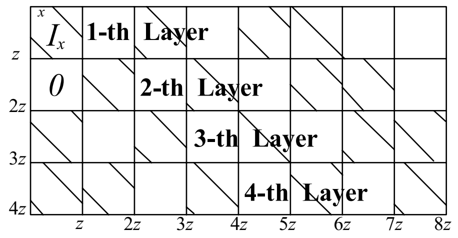


Fig. 1. A block structured parity check matrix with block rows (or layers) $j = 4$ and block columns $k = 8$, where the sub-matrix size is $z \times z$.

Table 1: Design parameters for H in several standards

LDPC Code	WLAN-802.11n	WiMax-802.16e	DMB-T
j	4-12	4-12	24-48
k	24	24	60
z	27-81	24-96	127

order to efficiently decode structured LDPC codes, we adopt the layered belief propagation (LBP) algorithm [6], which is described in Algorithm 1. In the description of the algorithm, $\mathcal{N}_m$ is the set of variable nodes that connected to check node m, and $\mathcal{N}_m \backslash n$ is the set $\mathcal{N}_m$ with variable node n excluded. Λ_{mn} and λ_{mn} denote the check and variable message, respectively. L_n denotes the *a posteriori* probability (APP) log-likelihood ratio (LLR) of variable node n: $L_n = \log(P(x_n = 0)/P(x_n = 1))$. H^l denotes l-th layer in H.

Algorithm 1 Layered belief propagation algorithm

```
Initialization:
  $\forall(m,n)$ with $H(m,n)=1$, set $\Lambda_{mn}=0$, $L_n=\frac{2y_n}{\sigma^2}$
for iteration $i=1$ to $I$ do
  for layer $l=1$ to $L$ do
    1) Read:
      $\forall(m,n)$ with $H^l(m,n)=1$:
        Read $L_n$ and $\Lambda_{mn}$ from memory
    2) Decode:
      $\lambda_{mn} = L_n - \Lambda_{mn}$
      $\Lambda_{mn}^{new} = \prod_{j\in\mathcal{N}_m\backslash n}\mathrm{sign}(\lambda_{mj})\Psi\big(\sum_{j\in\mathcal{N}_m\backslash n}\Psi(\lambda_{mj})\big)$
      $L_n^{new} = \lambda_{mn} + \Lambda_{mn}^{new}$
    3) Write back:
      Write $L_n^{new}$ and $\Lambda_{mn}^{new}$ back to memory
  end for
end for
Decision making: $\hat{x_n} = \mathrm{sign}(L_n)$
```

III. VLSI ARCHITECTURE

A. Block-serial scheduling algorithm

To implement Algorithm 1 in hardware, we propose a block-serial (BS) scheduling algorithm as shown in Fig. 2. In this algorithm, one full iteration is divided into j sub iterations. SISO decoding is applied to each layer in sequence. Each $z \times z$ sub-matrix is treated as a macro within which all the involved parity checks are processed in parallel using z number of SISO decoders. Each SISO decoder is independent from all others since there is no data dependence between adjacent check rows.

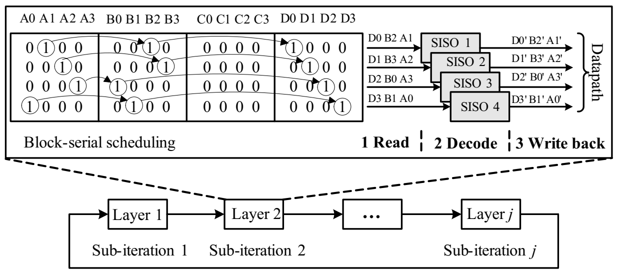


Fig. 2. Block-serial (BS) scheduling algorithm

B. Low-complexity implementation of BP algorithm

Conventionally, function $\Psi(x) = -\log(\tanh(|x/2|))$ is used for the decoding operations in Algorithm 1. However, the $\Psi(x)$ function is prone to quantization noise and can be numerically unstable [7]. Alternately, a different and numerically more robust way to compute the Λ_{mn} is shown as

$$\Lambda_{mn} = \sum_{j\in\mathcal{N}_m\backslash n}\!\!\boxplus\lambda_{mj} = \Big(\sum_{j\in\mathcal{N}_m}\!\boxplus\lambda_{mj}\Big) \boxminus \lambda_{mn}, \tag{1}$$

where the $\boxplus$ and $\boxminus$ operations are defined as $a \boxplus b \triangleq f(a,b) = \log\frac{1+e^a e^b}{e^a+e^b}$ and $a \boxminus b \triangleq g(a,b) = \log\frac{1-e^a e^b}{e^a-e^b}$ [8][9]. This computation method is especially suitable for the proposed BS scheduling algorithm in which the macro blocks are processed in sequential order. For hardware implementation, $f(\cdot)$ and $g(\cdot)$ functions can be simplified to

$$\begin{aligned} f(a,b) &= \mathrm{sign}(a)\,\mathrm{sign}(b)\Big(\min(|a|,|b|) + \\ &\qquad \log(1+e^{-(|a|+|b|)}) - \log(1+e^{-\left||a|-|b|\right|})\Big), \\ g(a,b) &= sign(a)sign(b)\Big(\min(|a|,|b|) + \\ &\qquad \log(1-e^{-(|a|+|b|)}) - \log(1-e^{-\left||a|-|b|\right|})\Big). \end{aligned} \tag{2}$$

In hardware, the non-linear correction terms $\log(1+e^{-x})$ and $\log(1-e^{-x})$ in (2) are approximated using low-complexity 3-bit lookup tables (LUTs) [9].

C. Radix-2 SISO decoder

Fig. 3 shows the proposed SISO decoder architecture for generating Λ_{mn}. We refer to it as Radix-2 (R2) recursion architecture since only one element can be processed in one clock cycle. The R2-SISO core consists of one $f(\cdot)$ recursion unit followed by one $g(\cdot)$ unit. Note that the $g(\cdot)$ unit would have the same structure as the $f(\cdot)$ unit but with a different LUT.

Fig. 4 shows the decoding schedule for check row m. During the first d_m[1] cycles, the incoming variable messages λ_{mn} ($\forall n \in \mathcal{N}_m$) are fed to the decoder sequentially and the $f(\cdot)$ unit is reused d_m times to obtain the intermediate $\boxplus$ sum S_m. Then, the outgoing messages Λ_{mn} ($\forall n \in \mathcal{N}_m$) are generated in a sequential order by the $g(\cdot)$ unit. Though the decoding is sequential for each check row, multiple (z) check rows within one layer can be processed in parallel by employing multiple (z) SISO decoders, which increases the throughput by a factor of z (see Fig. 2). Furthermore, the decoding throughput can be improved by overlapping the decoding of two layers as shown in Fig. 4. This scheduling would require dual-port memory for simultaneous read and write operations. Typically data dependencies between layers will occasionally stall the pipeline for one or more cycles. However the pipeline stalls can be avoid by shuffling the order of the layers [10].

D. Radix-4 SISO decoder via look-ahead transform

To increase the throughput of the R2-SISO decoder, a look-ahead transform can be used for the $f(\cdot)$ recursion. This transform leads to an increase in the number of data processed in each cycle as shown in Fig. 5, where two elements are processed in one clock cycle. We refer to this transform as Radix-4 (R4) recursion. Fig. 6 shows the corresponding Radix-4 SISO decoder architecture. Since

[1]d_m is the number of non-zero elements in check row m.

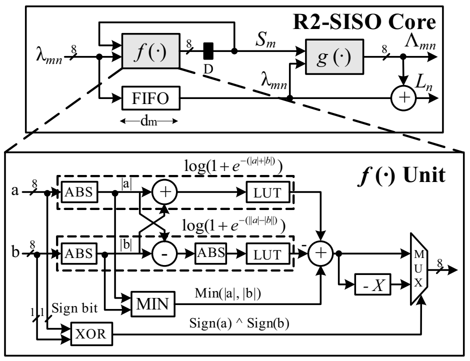
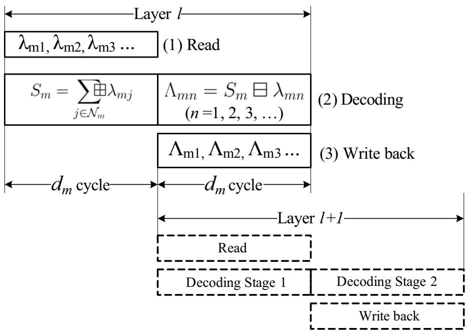


Fig. 3. Radix-2 (R2) SISO decoder architecture



Layer l
$\lambda_{m1}, \lambda_{m2}, \lambda_{m3} \dots$
(1) Read
$S_m = \sum_{j \in N_m} \boxplus \lambda_{mj}$
$\Lambda_{mn} = S_m \boxminus \lambda_{mn}$
(n =1, 2, 3, …)
(2) Decoding
$\Lambda_{m1}, \Lambda_{m2}, \Lambda_{m3} \dots$
(3) Write back
d_m cycle
Layer $l+1$
Read
Decoding Stage 1
Decoding Stage 2
Write back

Fig. 4. Pipelined decoding schedule

two elements can be processed in each cycle, it has a throughput speed up of 2. Table 2 summarizes the synthesis results (90nm CMOS technology) for the R4 and R2 SISO decoders. To compare these two architectures, we define an efficiency factor η as the throughput speed-up with R4-SISO divided by the area overhead. As can be seen, R4-SISO achieves throughput-area efficiency gains especially at lower clock frequency.

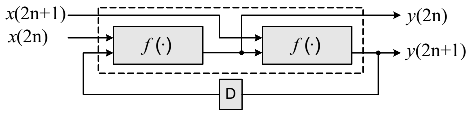


Fig. 5. One level look-ahead transform of $f(\cdot)$ recursion

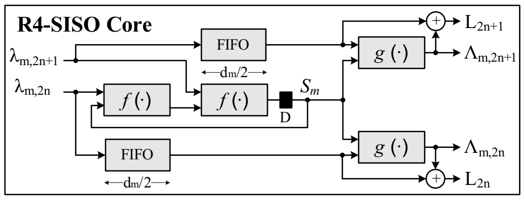


Fig. 6. Radix-4 (R4) SISO architecture

E. Scalable and reconfigurable LDPC decoder

To support multiple LDPC codes, the datapath has to be scalable and reconfigurable. Fig. 7 shows the proposed LDPC decoder architecture. In the proposed BS scheduling algorithm, the parallelism factor is equal to the sub-matrix size z. Since parameter z varies from code to code, i.e. 19 different sizes of z are defined in WiMax, we must design a datapath that is modular and scalable to support different code types. This is achieved by employing distributed SISO decoders and memory banks as shown in Fig. 7. This architecture can also reduce the overall power consumption by deactivating the memory banks and SISO decoders that are not being used. The L messages, on the other hand, are stored in a central memory bank for parallel accessing by z SISO decoders. This is achieved by grouping $[1 \times z]$ L messages (associated with each sub-matrix) into one memory word.

The decoding flow for one sub-iteration is as follows: at each cycle, $[1 \times z]$ L messages are first fetched from the L-memory and passed through a circular shifter to be routed to z SISO decoders. The soft input information λ_{mn} is formed by subtracting the old extrinsic message Λ_{mn} from the APP message L_n. Then the SISO decoder generates a new extrinsic message Λ_{mn} and APP message L_n, and stores them back to the Λ-memory and the L-memory, respectively.

Table 2: Comparison of two SISO decoder architectures

	450 MHz	325 MHz	200 MHz
R2 SISO area	6978 μm^2	6367 μm^2	6197 μm^2
R4 SISO area	12774 μm^2	10077 μm^2	8944 μm^2
$\eta = \frac{\text{Speedup}}{\text{Area overhead}}$	1.09	1.26	1.39

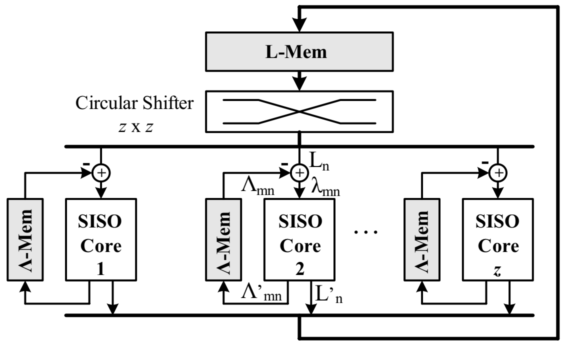


Fig. 7. LDPC decoder architecture with scalable datapath

By designing proper control logic, the decoder can be dynamically reconfigured to support multiple block-structured LDPC codes. With this partial-parallel architecture, the pipelined (Radix-4) decoding throughput is approximately equal to $\frac{2 \times k \times z \times R \times fclk}{E \times I}$, where k is the number of block-columns in H, z is the sub-matrix size, R is the code rate, E is the total number of non-zero sub-matrices in H, and I is the number of full iterations. Note that the latency of the circular shifter is not included in the throughput analysis, which may degrade the throughput by about 5-15%.

IV. RESULTS

A multi-mode LDPC decoder which supports both IEEE 802.11n and IEEE 802.16e has been synthesized on a TSMC 90nm 1.0V 8-metal layer CMOS technology. Fig. 8 shows the VLSI layout view of the LDPC decoder. Table 3 compares this decoder with the state-of-the-art LDPC decoders of [3] and [4]. The decoder in [3] has the flexibility to support 19 modes of LDPC codes in the WiMax standard, however it will not support the higher data rates envisioned for 4G and IMT-Advanced. The decoder in [4] has a throughput of 640 Mbps, but it does not have the flexibility to support multiple codes. As can be seen, our decoder shows significant performance improvement in throughput, flexibility, area and power.

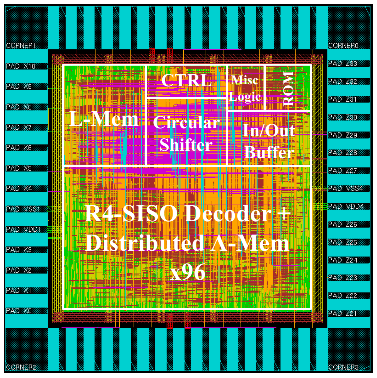
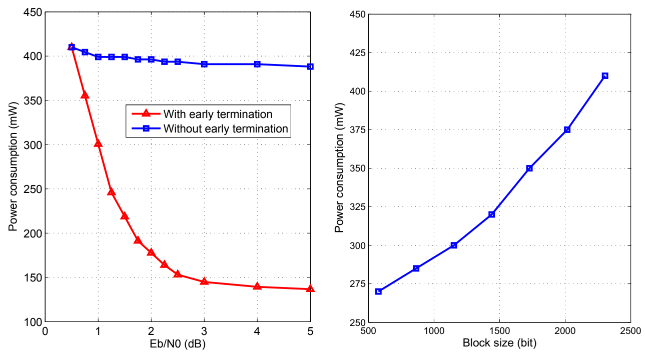


Fig. 8. VLSI layout view of the LDPC decoder

Table 3: LDPC decoder architecture comparison

	This Work	[3]	[4]
Flexibility	802.16e/.11n	802.16e	2048-bit fixed
Max Throughput	1 Gbps	111 Mbps	640 Mbps
Total Area	3.5 mm^2	8.29 mm^2	14.3 mm^2
Max Frequency	450 MHz	83 MHz	125 MHz
Peak Power	410 mW	52 mW	787 mW
Technology	90 nm	0.13 μm	0.18 μm
Max Iteration	10	8	10
Algorithm	Full BP	Min-Sum	Linear Apprx.

As low power design is critical for wireless receivers, in order to save power, we have implemented a simple and effective early termination criteria for stopping the iteration process. The decoding will stop if the following two conditions are satisfied: 1) the hard decisions for the information bits based on their LLR values do not change over two successive iterations, and 2) the minimum of the absolute values of the information bit LLRs is larger than a pre-defined threshold. As shown in Fig. 9 (a), when the wireless channel is good, the decoding needs fewer iterations to converge, which therefore saves substantial power (up to 65% power reduction). Another power saving technique is to use distributed SISO decoders and memory banks. Fig. 9 (b) shows the power reduction from deactivating the unused SISO decoders and memory banks when the LDPC code size is small.



(a) Early termination (Block size = 2304, Max iter. = 10)

(b) Distributed SISO decoding and memory banking

Fig. 9. Two power reduction techniques

V. CONCLUSION

A high performance LDPC decoder has been described that achieves a throughput of 1 Gbps. The decoder has a scalable datapath and can be dynamically reconfigured to support multiple 4G wireless standards.

VI. Acknowledgement

This work was supported in part by Nokia and by NSF under grants CCF-0541363, CNS-0551692, and CNS-0619767.

References

[1] R. Gallager, "Low-density parity-check codes," *IEEE Trans. Inf. Theory*, vol. 8, pp. 21–28, Jan. 1962.

[2] T. Brack, M. Alles, F. Kienle, and N. Wehn, "A Synthesizable IP Core for WIMAX 802.16e LDPC Code Decoding," in *IEEE 17th Int. Symp. Personal, Indoor and Mobile Radio Communications (PIMRC)*, 2006, pp. 1 – 5.

[3] X.-Y. Shih, C.-Z. Zhan, C.-H. Lin, and A.-Y. Wu, "A 19-mode 8.29mm^2 52-mW LDPC Decoder Chip for IEEE 802.16e System," in *2007 Symposium on VLSI Circuits*, June 2007.

[4] M.M. Mansour and N.R. Shanbhag, "A 640-Mb/s 2048-Bit Programmable LDPC Decoder Chip," *IEEE Journal of Solid-State Circuits*, vol. 41, pp. 684–698, March 2006.

[5] 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 Journal of Solid-State Circuits*, vol. 37, no. 3, pp. 404–412, 2002.

[6] D. Hocevar, "A reduced complexity decoder architecture via layered decoding of LDPC codes," in *IEEE Work. on Signal Processing Syst. (SIPS)*, Oct 2004, pp. 107–112.

[7] T. Zhang, Z. Wang, and K. Parhi, "On finite precision implementation of low density parity check codes decoder," in *Int. Symposium on Circuits and Systems (ISCAS)*, vol. 4, May 2001, pp. 202–205.

[8] J. Hagenauer, E. Offer, and L. Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Trans. Inf. Theory*, vol. 42, no. 2, pp. 429 – 445, 1996.

[9] X.-Y. Hu, E. Eleftheriou, D.-M. Arnold, and A. Dholakia, "Efficient implementations of the sum-product algorithm for decoding LDPC codes," in *IEEE GLOBECOM*, Oct. 2001, pp. 1036–1036.

[10] K. Gunnam, G. S. Choi, M. B. Yeary, and M. Atiquzzaman, "VLSI Architectures for Layered Decoding for Irregular LDPC Codes of WiMax," in *Int. Conf. Commun. (ICC)*, June 2007.