LOW-POWER IMPLEMENTATION OF A HIGH-THROUGHPUT LDPC DECODER FOR IEEE 802.11N STANDARD

Junho Cho[*]

Department of Electrical Engineering,
Seoul National University,
Seoul, 151-744, Korea

Naresh R. Shanbhag

Department of Electrical and Computer Engineering,
University of Illinois at Urbana-Champaign,
Urbana, Illinois 61801, USA

Wonyong Sung

Department of Electrical Engineering,
Seoul National University,
Seoul, 151-744, Korea

ABSTRACT

Flexible and scalable LDPC decoder architecture is developed for the IEEE 802.11n standard. The serial-parallel architecture is employed for achieving high throughput with low chip area, and triple-bank memory blocks are used for parallel factor expansion. Two low-power strategies using voltage over-scaling (VOS) and reduced-precision replica (RPR) are applied to the decoder. By applying these techniques, power saving of up to 35% is demonstrated when implemented in a 90nm CMOS technology.

1. INTRODUCTION

Among all known error correcting codes, low-density parity-check (LDPC) codes show the best performance approaching to channel capacity when decoded by iterative algorithms. The implementation challenges caused by the irregular interconnect have led to the development of implementation-oriented structural LDPC codes, such as quasi-cyclic (QC)-LDPC codes. Consequently, the LDPC codes have found applications in standards such as IEEE 802.3an (10GBASE-T), 802.16e (WiMAX) and 802.11n (Wi-Fi). Although the structural property eases implementation, there still remain many challenges including multi-mode design, high complexity, routing congestion, and high power consumption [1].

In this paper, flexible serial-parallel LDPC decoder architecture is designed for IEEE 802.11n standard, which defines the specification for enhanced high-throughput wireless local area network (WLAN) [2]. The proposed decoder architecture is easily scalable and highly flexible to support high throughput requirement and multi-mode operation upon various QC-LDPC codes. Two low-power strategies are proposed for the designed LDPC decoder architecture, using voltage over-scaling (VOS) [3] and reduced-precision replica (RPR) [4]. The decoder primarily operates on a slow data-path with small driving strength under VOS. A supplementary data-path with large driving strength and short path delay is activated in case of successive decoding failures to correct possible circuit errors. The proposed low-power strategies aim to reduce power consumption for majority of operating and channel conditions, while preserving functional robustness under deteriorated conditions by investing more power. Low-power implementation is especially desirable for mobile devices such as laptops employing IEEE 802.11n.

[*]The author performed this work while at University of Illinois at Urbana Champaign, USA. This work was supported in part by the Hynix Semiconductor Inc. and in part by the Ministry of Knowledge Economy (MKE), Republic of Korea, under the Brain Korea 21 Project

2. DECODING OF QC-LDPC CODES

2.1. LDPC Codes for IEEE 802.11n

An (N, K) LDPC code is specified by the parity-check matrix of size $M \times N$, where N, K and M are the codeword length, information length, and the number of parity-checks, respectively. With respect to structured QC-LDPC codes, the parity-check matrix is generally composed of γ rows and ρ columns of $\delta \times \delta$ permutation submatrices. There are 12 such parity-check matrices defined in IEEE 802.11n, which accommodate 4 code rates and 3 codeword lengths as listed in Table 1 [2]. Design of a flexible decoder is necessary for supporting these 12 different LDPC codes in a single decoder.

ρ	γ	Code rate (R)	Submatrix size (δ)	Codeword length (N)	Information length (K)
24	4	5/6	27	648	540
			54	1296	1080
			81	1944	1620
	6	3/4	27	648	486
			54	1296	972
			81	1944	1458
	8	2/3	27	648	432
			54	1296	864
			81	1944	1296
	12	1/2	27	648	324
			54	1296	648
			81	1944	972

Table 1. LDPC Codes for IEEE 802.11n

2.2. Offset Min-Sum Algorithm

The min-sum algorithm (MSA) approximates the numerically accurate sum-product algorithm (SPA) with simple arithmetic operations, thereby significantly reducing computational complexity [5]. Slight performance degradation due to the approximation error can be recovered by dividing the approximate value with a normalization factor $\alpha \geq 1$ or by subtracting it with an offset $\beta \geq 0$ [6]. In this paper, the offset-based calibration is used for MSA, which performs decoding as follows:

Denotation:

x_n: The n-th symbol of transmitted codeword

y_n: The n-th symbol of received word

$\mathcal{M}(n)$: Set of check nodes participating in bit node n

$\mathcal{N}(m)$: Set of bit nodes participating in check node m

Initialization:

Each bit node n is assigned an *a posteriori* log-likelihood ratio (LLR) for every check node $m \in \mathcal{M}(n)$ such that

$$\lambda^b_{n\to m} = \Lambda_n = \ln\big(\Pr(x_n = 0|y_n)/\Pr(x_n = 1|y_n)\big). \quad (1)$$

Step 1. Check node update:

For each check node m, and for each bit node $n \in \mathcal{N}(m)$, compute the bit-to-check messages

$$\lambda^c_{m\to n} = \prod_{n'\in\mathcal{N}(m)\backslash n} \operatorname{sgn}\Big(\lambda^b_{n'\to m}\Big) \times \max\left\{\min_{n'\in\mathcal{N}(m)\backslash n} |\lambda^b_{n'\to m}| - \beta, 0\right\}. \quad (2)$$

Step 2. Bit node update:

For each bit node n, and for each check node $m \in \mathcal{M}(n)$, compute the check-to-bit messages

$$\lambda^b_{n\to m} = \Lambda_n + \sum_{m'\in\mathcal{M}(n)\backslash m} \lambda^c_{m'\to n} \quad (3)$$

and the decision criterion

$$\Lambda'_n = \Lambda_n + \sum_{m\in\mathcal{M}(n)} \lambda^c_{m\to n}. \quad (4)$$

Step 3. Decision:

Obtain the hard-decision vector $\hat{\mathbf{x}} = [\hat{x}_1, \hat{x}_2, \cdots, \hat{x}_N]$, where $\hat{x}_n = 0$ if $\Lambda'_n \geq 0$, and $\hat{x}_n = 1$ if $\Lambda'_n < 0$. If $\hat{\mathbf{x}}\mathbf{H}^T = \mathbf{0}$ or the maximum iteration limit is reached, finish the decoding process with the decoded codeword $\hat{\mathbf{x}}$. Otherwise, go to *Step 1*.

3. DECODER ARCHITECTURE

The IEEE 802.11n standard supports the data rates of up to 600Mbps. Highly parallel processing is required in order to satisfy this high-throughput specification. Partially parallel decoder architecture can deliver sufficient parallelism while overcoming routing congestion by partitioning the design into several subsets. Moreover, it can offer flexibility to support various code rates and codeword lengths [7].

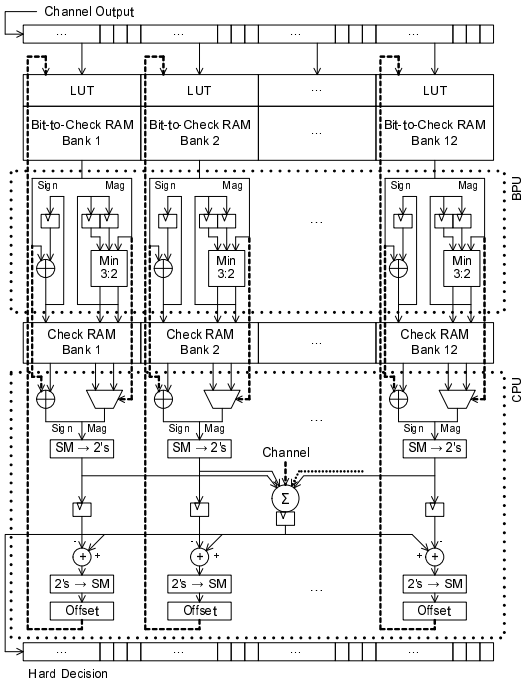


Fig. 1. Baseline serial-parallel architecture

3.1. Baseline Serial-Parallel Architecture

The "serial-parallel" architecture illustrated in Fig. 1 is adopted for our MSA decoder. This architecture is similar to the "parallel-serial" architecture for the SPA decoder proposed in [8], yet has some advantages such as simpler processing units, smaller memory overhead and extendability in parallelism. These advantages are mostly brought from elaborate memory mapping and customized address generation units (AGUs). The bit node processing unit (BPU) is serialized and the check node processing unit (CPU) is parallelized in this work (see Fig. 1), thus is called "serial-parallel" architecture. Note that the parallelization of the BPU for finding the two smallest magnitudes among $\rho = 24$ messages is more costly than that of the CPU for just summing up 24 messages. The proposed architecture operates as follows.

When decoding begins, the bit-to-check RAM loads initial *a posteriori* LLRs defined by (1) from the channel output buffer. Then, in order to compute (2), the BPU performs a modulo-two summation of the sign bits over all ρ (=24) bit nodes participating in the first check node. At the same time, it searches for the first and second minimum magnitudes among the 24 bit nodes. This operation is performed serially, in such a way that a MIN32 module enumerates two new minimum values, at every clock cycle, from the two previous minimum values and one input data. There are γ BPUs operating simultaneously for each row group. Therefore, 24(cycles for row processing) $\times$ δ(rows in a submatrix) $= N$ clock cycles are required to compute all check nodes. Since the parity-check matrices of the IEEE 802.11n standard have

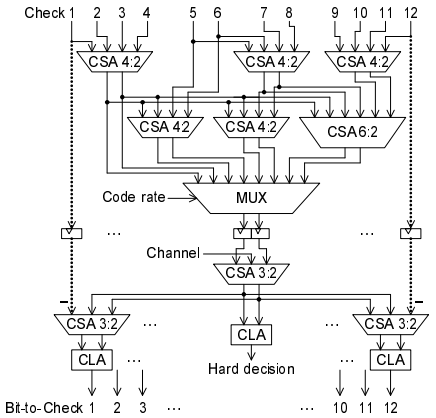


Fig. 2. Check-to-bit processing unit supporting various code rates

four γ values of 4,6,8 and 12 corresponding to the code rates of 5/6, 3/4, 2/3 and 1/2, respectively, the number of memory banks and processing lanes is set to 12, which is the maximum of the four γ values. If a code rate of greater than 1/2 is used, less than 12 lanes are needed, and hence the unused banks and lanes are turned off. In this stage, the check magnitudes are stored without offset calibration. It is delayed to the final stage of the bit node update step, which then makes it possible to apply logic optimization technique over several blocks of similar computation, and to reduce the size of MIN32 module and check RAM by not storing the address of the minimum magnitude. This delayed offset calibration does not degrade error correcting performance as will be shown in Section 4.2.

In the following step, the CPU loads a check node from the check RAM, and a bit-to-check message from the bit-to-check RAM. A new marginalized check-to-bit message described in (2) is obtained from these two data. The first minimum magnitude is selected if it is not equal to the magnitude of the loaded bit-to-check message, otherwise the second minimum is chosen. Note that the first and second minimums can be identical when there are two or more bit-to-check messages that have the same minimum reliability. Then, to compute (3), the γ check-to-bit messages are globally summed up with a channel output, and then again marginalized locally in each lane (see the lower part of Fig. 1), thereby supplying new γ bit-to-check messages $\lambda^b_{n\to m}$ for a bit node n. These bit-to-check messages are written back with the same address used in the read operation. It takes N clock cycles to update all bit-to-check nodes. The CPU is implemented by a tree of carry save adders (CSAs) that can support various code rates, as depicted in Fig. 2. One-stage pipeline is applied to the CPU, since the 13-input (including twelve check-to-bit messages and one channel message) parallel adder has long path delay.

Memory mapping should be manipulated to efficiently utilize the structural property of QC-LDPC codes. The bit-to-check messages are stored in the order that generates sequential check nodes. Both bit-to-check and check RAMs are thus accessed by sequential addresses at the check node update step. On the other hand, to compute the bit-to-check messages for a sequential series of bit nodes, LUTs are required to supply the irregular addresses. Size of these LUTs can be minimized using the cyclic shift property of submatrices; i.e., in each column group, the start addresses are only irregular but the following $\delta - 1$ addresses are regular and hence can be calculated by the increment-mod-δ modules. The addresses generated either by LUT search or by computation are shared between the bit-to-check and check RAMs. In this way, the memory size can be reduced as follows:

- Channel buffer: Nw, where w denotes the word length,
- Bit-to-check RAM: $N\gamma w\,(= M\rho w)$,
- Check RAM: $M(2w-1)$,
- Address LUT: $\rho\gamma\lceil\log_2\delta\rceil$,
- Output buffer: N

Note that the channel buffer, bit-to-check RAM, check RAM, and output buffer are shared by all 12 LDPC codes defined in the IEEE 802.11n, whereas the address LUTs are implemented individually. Total of 191,484bits of RAM and 12,960bits of ROM are required for the IEEE 802.11n.

3.2. Parallel Factor Extension Using Triple-Bank Memory

Additional parallelism is exploitable over the baseline architecture using the sequential memory access within each submatrix. The memory banks should contain $P \times w$ bits in a row to use a parallel factor of P. However, non-aligned memory accesses may arise when the requested messages are laid across two rows. This problem has been an important issue in increasing data-level parallelism of computing, and the dual-bank approach proposed in [9] seems a suitable method to resolve it. A triple-bank memory is employed in this paper instead of the dual-bank one, since the data consists of an odd number of elements when $\delta = 27$ or 81 so that bank conflicts are not avoidable across the cyclic round in the dual-bank one. The structure of triple-bank memory is illustrated in Fig. 3. The entire bit-to-check and check RAMs consist of 12 macro-banks, each of which is composed of 3 micro-banks, where a micro-bank contains P messages in a row. Parallel factor P of 3, 9 or 27 can be used for each micro-bank, where 9 is chosen for the particular throughput requirement of the IEEE 802.11n. For the parallel memory access, the AGU generates two addresses and two read/write strobes for the two micro-banks in which the required data resides. Then, the loaded two message vectors are combined, shifted and selected by the alignment hardware (see Fig. 3). In this way, a single cycle fetch of unaligned data is realized. Although the latency of memory access increases when pipelined, the throughput is maintained regardless of data alignment. In the mean time, the BPUs and CPU are just duplicated P times to process the

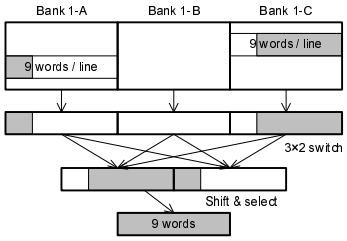


Fig. 3. Unaligned memory access using triple bank memory

data in parallel.

Using the parallel factor P over the proposed serial-parallel architecture, it takes δ/P clock cycles to initialize the bit-to-check memory, $(N/P + t_1)$ clock cycles to complete the check node update step, and $(N/P + t_2)$ clock cycles to finish the bit node update step, where t_1 and t_2 denote the clock cycles for the pipelining of each step. If there is no error in the received word, only a half iteration is needed to terminate decoding. In this case, the accumulation of the signs over all check nodes by an OR gate is zero (implying satisfied), hence the channel output can be just bypassed to the decoder output. Therefore, the decoding throughput is computed by $N/\{\delta/P + (N/P + t_1) + (2N/P + t_1 + t_2) \times n_{iter}\}$ bits per clock cycle, where n_{iter} denotes the number of iterations. Assuming that t_1 and t_2 are much smaller than N/P and $n_{iter} \approx 0$, which means that most decoding is finished after the first half iteration, approximately P bits are decoded at every clock cycle.

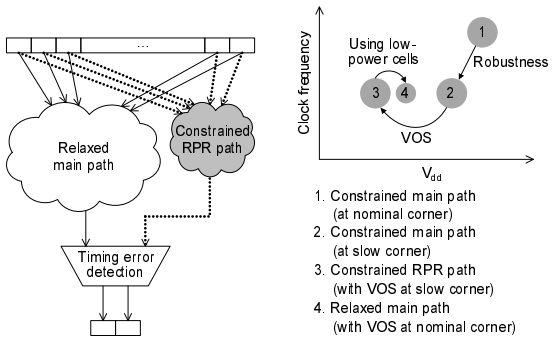
4. LOW-POWER IMPLEMENTATION

In this work, standard cell based design is performed with a gate-level synthesis tool using the IBM 90nm process technology.

4.1. Low-Power Strategies Using VOS and RPR

Attaining the design goals of both speed and power at the same time is difficult when using design automation tools, since they employ more power-hungry cells to reduce path delay; i.e., timing-oriented designs dissipate much power and power-oriented designs exhibit long delays. Furthermore, the modern design methodology based on the worst-case analysis is so pessimistic that the resulting circuits consume much more power than required in most cases. In order to deal with these problems, VOS [3] and RPR [4] techniques are applied.

The RPR path performs exactly the same function with the main path but of reduced precision. The VOS refers to reduction of supply voltage $\mathrm{V_{dd}}$ below the critical value $\mathrm{V_{dd-crit}}$ that ensures functional correctness at the slow corner. In the proposed design flow, the main path is given no timing constraint, thereby being synthesized with low-power cells. The RPR path, however, is synthesized under a strict timing constraint to prevent timing violations even under



(a) Hardware consists of main and RPR paths

(b) Voltage-frequency design space

Fig. 4. Low-power implementation strategy using VOS and RPR

harsh operating conditions. Fig. 4(a) shows an example hardware with a relaxed main path and a strictly constrained RPR path. Note that the timing error detector need not be implemented separately, but can be substituted by the decoding failure detector in this particular application; i.e., detection of successive decoding failure is considered to the occurrence of timing error. Based on this hardware, a low-power implementation strategy is established as illustrated in Fig. 4(b). In this figure, size of shaded circles reflects the average power consumption of each cell in the design. Circle 1 indicates the constrained main path at the nominal process, voltage and temperature (PVT) corner. In order not to decrease yield under process variations, as well as to make the circuit robust to voltage and temperature variations, the worst-case simulation is performed at the 44% slower process corner, together with $0.9 \times \mathrm{V_{dd-crit}}$ and 125°C operating condition (circle 2). At circle 3, the clock frequency of main path achieved at circle 2 is maintained in the RPR path while VOS applied. The final step at circle 4 is to release the timing constraint of main path until timing violations do not occur at the nominal corner. Note that the design at circle 4 is synthesized with low-power cells, whereas those at circles 1, 2 and 3 are built with power-hungry cells. The fast and power-consuming RPR path is only activated when timing errors are detected successively in the main path. The resulting circuits can effectively save power consumption in majority cases by operating on the low-power main path, while preserving resiliency against worst cases by operating on the fast RPR path.

Considering that success of decoding depends on both channel condition and circuit operating condition, a more aggressive power-saving strategy can be employed, which operates a relaxed RPR path for typical cases and a constrained main path for worst cases. Suppose that the decoder operates with good channel and operating conditions. Then it is desirable to assign as low precision and power as possible for those majority cases. In rare cases of successive decoding failure, more precision and power can be exploited to combat bad channel or circuit operating conditions. Because the high-

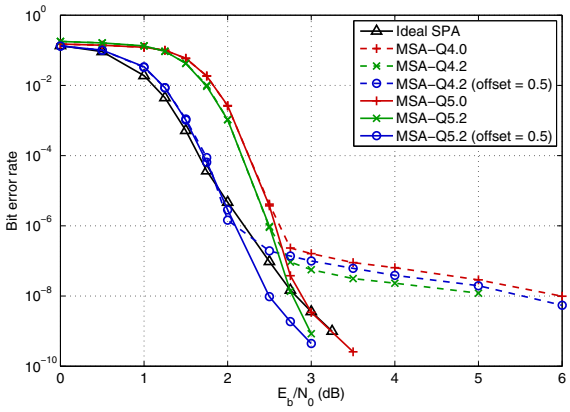


Fig. 5. BER performance of various MSA implementations (iteration limit = 30)

precision main path is expensive to accelerate, it consumes more power than the constrained RPR path of the first strategy. The two proposed low-power strategies will be simply called designs A (the former) and B (the latter), respectively, for convenience. The comparison of them will be discussed in Section 4.3.

4.2. Word Length Decision for Main and RPR Paths

The word length w for the bit-to-check and check messages in the high-precision path was determined to 7 bits under various simulations, for which the integer length w_i is 5 bits including a sign bit and the fraction length w_f is 2 bits. The low-precision path computes the 5-bit integer part only. The round-to-nearest-integer scheme is used for quantization in the channel buffer, and the round-to-zero scheme in the other parts. Messages with larger magnitude than that can be represented in w bits are saturated. For the most demanding code, which is the rate-1/2 (1944, 972) code, some selected performance estimation results are plotted in Fig. 5, where the quantization scheme is denoted by $\mathrm{Q}w_i.w_f$. As previously stated in [10] for different LDPC codes, when 4 bits are assigned to the integer part (i.e., when $\mathrm{Q}4.w_f$ is used), early error floors are introduced due to clipping effect. The early error floors disappear with the increased integer length of 5 bits. The offset value of $\beta = 0.5$, which is 10 in binary Q5.2 representation, shows the best error correcting performance among a number of binary Q5.2 offset values. By using offset calibration, the MSA attains approximately 0.4dB SNR gain at the bit error rate (BER) of 10^{-6}, and even outperforms the ideal SPA of floating point precision at the high SNR regime. With respect to the Q5.0 MSA, no offset value improves the non-calibrated MSA, since the least significant bit (LSB) still has a large magnitude in the Q5.0 quantization. The Q5.0 MSA shows worse performance than the Q5.2 offset-MSA by $0.4 \sim 0.5$dB at wide range of BER, but an apparent error floor is not observed until the BER of 10^{-9}.

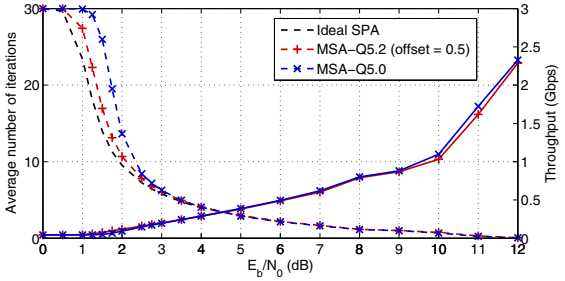


Fig. 6. Average number of iterations (dashed line) and decoding throughput (solid line)

The design A will show the error performance of the Q5.2 offset-MSA for the majority of operating conditions with the main path, and that of the Q5.0 MSA for the worst operating conditions with the RPR path. The design B functions sufficiently well at the high SNR regime above 4dB with the RPR path at nominal operating condition, while the frequent decoding failure caused by low SNR or bad circuit condition activates the high-precision main path to improve error correcting performance.

Fig. 6 shows the average number of iterations until decoding completion and the corresponding decoding throughput at the clock frequency of 294MHz. The maximum number of iterations is set to 30. If the assigned iterations are fully consumed at the low SNR regime, only 43Mbps is attained. However, in this regime, high throughput has no significance since most decoding is failure. The 600Mbps throughput is achieved near 7dB with the proposed architecture. The decoding throughput increases up to 2.54Gbps if the channel condition is very good and there is no contamination in all received words,

4.3. VLSI Implementation Results

Several VLSI circuits were implemented for the processing units of the proposed LDPC decoder as listed in Table 2. The maximum clock frequency was determined at the slow PVT corner. Power estimation is based on a sample decoding at 6dB of signal-to-noise ratio (SNR). Without supply voltage scaling (see the left half of table), designs with low clock frequency and high parallel factor consume less power. The decoder with 1.6 times faster clock frequency requires 21% more power for the same decoding throughput. By simply scaling down the supply voltage of the conventional designs, 10% power saving is achieved (see the right half of table). The power reduction due to low supply voltage overcomes the increase caused by using expensive cells to meet the rigorous timing constraint.

In the design A, by adding an RPR path with a strict timing constraint, the relieved main path can perform decoding with 22% less power. Moreover, 8% power saving is still achieved when the main and RPR paths operate simultaneously in bad conditions, since constraining the RPR path is

Implementation strategy		Without VOS (V_{dd}=1.2V)			With VOS (V_{dd}=1.0V)		
		Relaxed main path	Slightly constrained main path	Aggressively constrained main path	Slightly constrained main path	Relaxed main & constrained RPR paths (Design A)	Relaxed RPR & constrained main paths (Design B)
Critical path delay	At nominal corner	1.89 ns	1.62 ns	1.17 ns	1.59 ns	$1.88^{\dagger}$/ $2.70^{\ddagger}$ ns	$1.87^{\ddagger}$/ $2.69^{\dagger}$ ns
	At slow corner	3.14 ns	2.69 ns	1.90 ns	2.70 ns	3.20/ 4.72 ns	3.20/ 4.74 ns
Maximum clock frequency		294 MHz	345 MHz	476 MHz	345 MHz	294 MHz	294 MHz
Req. parallel factor (P)		9	7.7	5.6	7.7	9	9
Area	γ lanes	25,232 μm^2	26,098 μm^2	31,324 μm^2	33,369 μm^2	45,441 μm^2	47,006 μm^2
	γP lanes	0.23 mm^2	0.20 mm^2	0.18 mm^2	0.26 mm^2	0.41 mm^2	0.42 mm^2
Power	γ lanes at 294 MHz	2.87 mW	2.87 mW	3.56 mW	2.59 mW	2.23^{*}/ $2.64^{\S}$ mW	1.86^{*}/ $2.92^{\S}$ mW
	γ lanes at max freq.	2.87 mW	3.32 mW	5.56 mW	3.01 mW	2.23/ 2.64 mW	1.86/ 2.92 mW
	γP lanes	25.8 mW	25.6 mW	31.1 mW	23.2 mW	20.1/ 23.8 mW	16.7/ 26.3 mW
	at max freq.	(100 %)	(99 %)	(121 %)	(90 %)	(78^{i}/ 92^{ii}%)	(65^{iii}/ 102^{iv}%)

†: RPR path delay, ‡: Main path delay, *: Power consumption at nominal corner, §: Power consumption at slow corner,
i) Main path operated at nominal corner, ii and iv) Both paths operated at slow corner, iii) RPR path operated at nominal corner

Table 2. Area and Power Comparison of Various VLSI Implementations of Processing Units

less costly than constraining the main path. In the design B, the relaxed RPR path saves 35% power in the typical operating condition. However, the power saving diminishes in either bad channel or operating conditions. The longest path delays of the fast supplementary paths are 3.2ns at the slow corner in our two alternative designs. The slow primary paths for the typical cases reach the endpoint within 2.7ns at most in typical cases. The timing margin of these primary paths is not small, especially when considering the static timing analysis is pessimistic. Hence the activation of the power-consuming supplementary path will be quite infrequent. Notice that the circuit area is much increased in both designs at the expense of power saving.

The design A might be more beneficial if the channel or operating conditions vary frequently, which shows balanced low-power consumption across the operating conditions. The design B is advantageous in relatively stable and good conditions. Determination of better design depends on the statistics with respect to the variation of PVT and channel conditions.

5. CONCLUDING REMARKS

Two low-power strategies are proposed in this paper using VOS and RPR. The proposed strategies effectively reduce overinvested power for the majority of channel and circuit operating conditions. They could save up to 35% of nominal power consumption in the processing units of the designed flexible and scalable LDPC decoder. Power gating is expected to achieve more power saving than the clock gating used for deactivating the unused data-path in this paper, at the expense of longer activation time.

6. REFERENCES

[1] X-Y. Shih, C-Z. Zhan, C-H. Lin, and A-Y. Wu, "An 8.29 mm^2 52 mW multi-mode LDPC decoder design for mobile WiMAX system in 0.13μm CMOS process," *IEEE J. Solid-State Circuits*, vol. 43, pp. 672–683, Mar. 2008.

[2] *Wireless LAN medium access control (MAC) and physical layer (PHY) specifications: enhancements for higher throughput*, IEEE Std. P802.11n/D7.0, 2008.

[3] R. Hegde and N. R. Shanbhag, "A voltage overscaled low-power digital filter IC," *IEEE J. Solid-State Circuits*, vol. 39, pp. 388–391, Feb. 2004.

[4] B. Shim and N. R. Shanbhag, "Low-power digital signal processing via reduced-precision redundancy," *IEEE Trans. VLSI Syst.*, vol. 12, pp. 497–510, May 2004.

[5] M. P. C. Fossorier, M. Mihaljević, and H. Imai, "Reduced complexity iterative decoding of low-density parity check codes," *IEEE Trans. Commun.*, vol. 47, pp. 673–680, May 1999.

[6] J. Chen and M. P. C. Fossorier, "Near optimum universal belief propagation based decoding of low-density parity check codes," *IEEE Trans. Commun.*, vol. 50, pp. 406–414, Mar. 2002.

[7] S. S. Tehrani, S. Mannor, and W. J. Gross, "Fully parallel stochastic LDPC decoders," *IEEE Trans. Signal Process.*, vol. 50, pp. 5692–5703, Nov. 2008.

[8] Z. Zhang, L. Dolecek, B. Nikolić, V. Anantharam, and M. Wainwright, "Investigation of error floors of structured low-density parity-check codes by hardware emulation," in *Proc. IEEE GLOBECOM*, San Francisco, CA, USA, Nov. 2006, pp. 1–6.

[9] M. Alvarez, E. Salami, A. Rami, and M. Valero, "Performance impact of unaligned memory operations in SIMD extensions for video codec applications," in *Proc. IEEE ISPASS*, San Jose, CA, USA, Apr. 2007, pp. 62–71.

[10] J. Zhao, F. Zarkeshvari, and A. H. Banihashemi, "On implementation of min-sum algorithm and its modifications for decoding low-density parity-check (LDPC) codes," *IEEE Trans. Commun.*, vol. 53, pp. 549–554, Apr. 2005.