Architectures for ASIC Implementations of Low-Density Parity-Check Convolutional Encoders and Decoders

Ramkrishna Swamy, Stephen Bates and Tyler Brandon
Department of Electrical and Computer Engineering
University of Alberta, Edmonton, Canada.
Email: {swamy, stephen.bates, brandon}@ece.ualberta.ca

***Abstract*— Low-Density Parity-Check Convolutional Codes (LDPC-CCs) are an attractive alternative to their block-oriented counterparts, LDPC-BCs.**

In this paper, we introduce these codes and propose an encoder and decoder architecture that is implementable as an ASIC. We report upon a realization of this new architecture capable of an information throughput of 430 Mbps and 164 Mbps for the encoder and decoder, respectively. We discuss a top-level chip specification and then extend ideas to parallelize the design.

I. Introduction

Low-Density Parity Check (LDPC) codes have attracted a lot of attention in recent years. Implementations of LDPC Block Code (LDPC-BC) decoders have been reported in industry and academia [1], [2]. However, it has been shown that LDPC Convolutional Codes (LDPC-CCs) may be more suited to certain applications, such as streaming video and packet switching networks, than their LDPC-BC counterparts [3]. However, until encoder and decoder architectures have been implemented in ASICs and FPGAs, the real advantages of adopting them is unclear. We discuss the FPGA implementation of an LDPCC-CC decoder using a memory-based architecture in [4]. In this paper, we focus on a register-based LDPC-CC encoder and decoder designed for ASICs. This design is based upon the concept of processors, which is analogous to iteration cycles in LDPC-BC decoding. This becomes advantageous when multiple processors are tiled and wired, and leads to a design that is globally less complicated.

We introduce and describe LDPC-CCs in Section II. In Section III, we propose a top-level specification for the ASIC implementation of the encoder and decoder. We then present initial synthesis results for the encoder and decoder, utilizing the 0.18μm CMOS technology files, in Section IV. We go on to present results for a pipelined version of the LDPC-CC encoder in Section V. We finish in Section VI with conclusions and a discussion of ongoing and future work.

II. An Overview of LDPC-CCs

LDPC-CCs were first proposed by Felström and Zigangarov [5]. They are similar to LDPC-BCs in that they generate code-bits based on parity-check operations [6]. However they differ in that any given code-bit, $v(t)$, is generated using only previous information bits and previously generated code-bits. This implies that the parity-check matrix for LDPC-CCs is inherently lower diagonal and this simplifies the encoding process and reduces encoding latency [6].

An important parameter of an LDPC-CC is its memory, M. The performance of well designed LDPC-CCs increases with M but so too does the encoder and decoder complexity.

For example, the LDPC-CC code-bit generation equation, for a regular (3,6) rate 1/2 code, at time $t = nM + \phi$ can be written as

$$\begin{aligned} v(nM+\phi) &= v((n-1)M+\phi) + v(nM - \delta^{(v)}(\phi)) + \\ &\quad \textstyle\sum_{i=1}^{3} u(nM - \delta_i^{(u)}(\phi)). \end{aligned} \tag{1}$$

Here, $n \geq 0$, $\phi \in \{0, 1, 2, \cdots, M-1\}$, $\delta^{(v)}(.)$ and $\{\delta_i^{(u)}(.)\} \in \{1, 2, \cdots, M-1\}$ and $+$ implies addition in GF(2) (i.e. XOR operations). Note that $v(nM+\phi)$ is always generated as part of a parity-check with $v((n-1)M+\phi)$. Here $u(t)$ represents the information bit at time t and the δ terms are a function of the code and are related to its parity-check matrix. For a discussion on construction methods for LDPC-CCs see [7].

A. Encoding

There are several advantages to using LDPC-CCs. One advantage is an encoder structure that is simpler than that of LDPC-BCs. The outputs at time t are merely a function of the present input, $u(t)$, and the encoder state, S. The encoder state is the set of the last M information-bits, the last M code-bits and the phase term, ϕ.

One code-bit and one information-bit are generated with each input since we are implementing a regular, rate-1/2 parity check code. From Figure 1, we can see that the encoder complexity of LDPC-CCs is $O(M)$. This compares well with regular LDPCs, which have a worst case complexity $O(N^2)$ [6]. In addition, there is no need to implement a control mechanism to read in a block, shift the block into an encoder and then encode an entire block, all the while transmitting data.

As illustrated in Figure 1, the encoder architecture comprises first-in first-out (FIFO) delay lines for the code-bits and

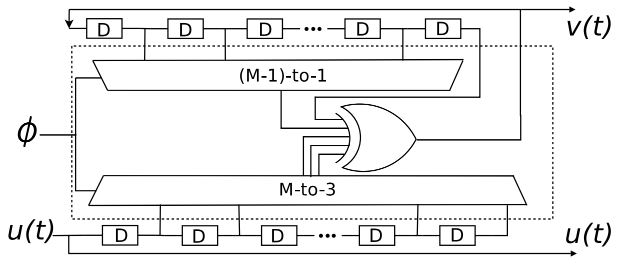

Fig. 1. The architecture for an LDPC-CC encoder. $u(t)$ is the information bit, $v(t)$ is the code bit and ϕ is the phase at time t. The multiplexing units implement a rate-1/2 (3,6) parity check matrix.

information-bits. The code-bit delay line can be implemented as a set of registers, where exactly two previous code-bits are selected as inputs to the XOR gate. The information-bit delay line is represented by another set of registers, where exactly three information-bits are selected.

It is noted that four of the five signals selected as inputs to the XOR gate are not located at fixed points on the delay lines. Two multiplexors, driven by the phase signal ϕ, are required to select these signals. The output of the encoder comprises the most recent information-bit and code-bit, which is the output of the XOR gate.

B. Decoding

LDPC-CC decoding utilizes the concept of processors. Each processor is a physically independent unit consisting of delay elements, a single parity node and a single variable node. In addition, a processor is comparable to an iteration in a regular LDPC-BC and as such, performance increases as we add more processors, until an upper bound is reached [8]. The architecture of each processor is identical and the outputs of processor $j-1$ are the inputs of processor j. These inputs and outputs are Log-Likelihood Ratios (LLRs) and for the first processor, the inputs are the channel LLRs.

The repetitive nature of the architecture suggests that LDPC-CCs are well-suited for ASIC and FPGA implementations. Efforts only have to be focussed on one processor and upon doing so, it can be tiled as many times as required or permitted by die size or resources. Moreover, the design then becomes very easy to parallelize and trade-offs between throughput and bit error rate (BER) can be made. For example, 30 processors can be configured as three decoders with 10 processors each or one large decoder using all 30 processors. The former has a $\times 3$ information throughput of the latter. However, the latter has improved BER performance. In addition, no high-speed global routing is required. This is still a major obstacle for LDPC-BC decoders where silicon utilization figures can be poor due to routing overhead [1].

Figure 2 depicts a simplified model of a processor in an LDPC-CC decoder. The processor module takes LLRs and phases as inputs. The LLR inputs to processor j can be the channel LLRs, if it is the first processor, or it can be the outputs of processor $j-1$. Once in the processor, these LLRs get routed to their prospective delay lines, namely, $\{\ell_u^{(i)}\}$ for information LLRs and $\{\ell_v^{(i)}\}$ for code LLRs. The word-length of each LLR is six bits. This sums to 48 bits for the LLR inputs and outputs of each processor.

As with the encoder, this approach is very register intensive. The registers are required to keep track of current and previous information-bits and code-bits. It is evident that the processor latency for this design is $N = 140$, even though the code only requires $M+1$ registers to be functional. However, extra registers are needed for pipelining. Such pipelining relaxes the critical path of the decoding operation and only impacts upon the latency of the decoder.

Exactly six inputs are taken from the LLR delay lines at a specific phase and the outputs from the parity-check block are then written back to the registers, from which they were read. The parity-check operation implements the sign-minimum function. In other words, each output evaluates the minimum of the five other inputs times the sign of the five inputs. This can be written as

$$\begin{gathered} y_k = \min_{\forall n \neq k} \{x_n\} \prod_{\forall k \neq n} sgn(x_n) \\ \text{where } n, k \in \{1, 2, \ldots, 6\}. \end{gathered} \tag{2}$$

Just prior to leaving the processor, a variable node operation is performed. The LLRs are summed as shown on the RHS of Figure 2. The new LLRs are then directed to the next processor or to the decision slicer if it is the last processor.

III. Top-level specification

In this section, we present an overview of the top-level specification of the ASIC. It describes some of the debugging features specifically implemented for this design.

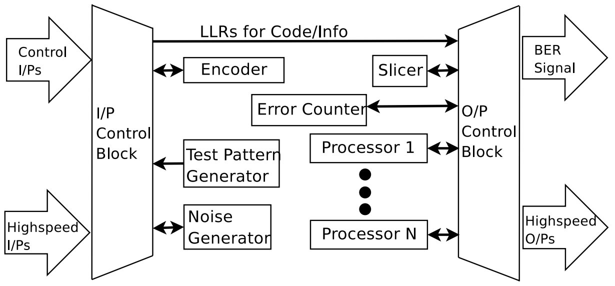

Fig. 3. A block diagram of the top-level chip. Note that an LDPC-CC encoder and decoder are both located on the same chip.

Figure 3 describes the overall architecture of the chip. The components involved are the test pattern generator, noise generator, encoder, decoder, slicer and an error counter for the BER.

The input and output control blocks have two-way communication with most of the components. This enables a built-in-self-test (BIST) mode as well as a fully configurable design.

In the BIST mode, we generate inputs using the test pattern generator and the encoder. We then add approximate Additive-White Gaussian Noise (AWGN) to avoid having to bring high-speed signals into the device. The processors are connected via the output control block. This allows us to move data from one

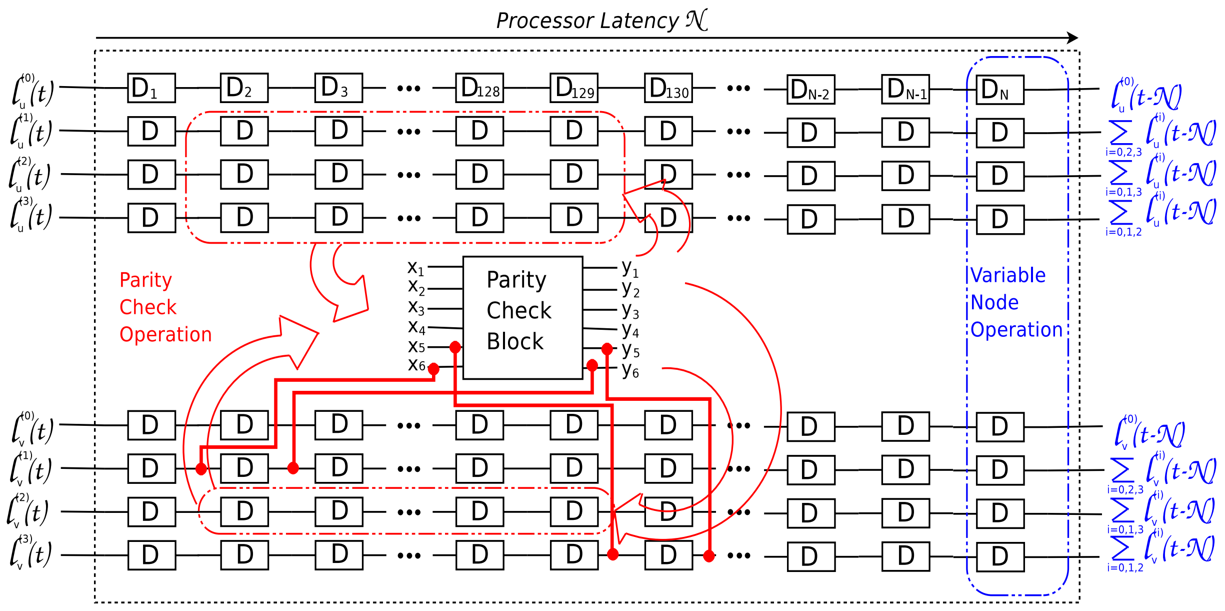

Fig. 2. A simplified model of a processor in the decoder. For this specific design, $N = 140$, hence it takes 140 clock cycles to see an output on the RHS. Being a (3,6) regular code, there are four delay lines - three for the variable nodes and one for original channel LLRs - and every parity-check operation takes in six inputs and two of them are hard-wired from the code-bit delay lines. The inputs and outputs of the parity-check block are $\{x_k\}$ and $\{y_k\}$, respectively.

processor to any other processor. This becomes important if defects are introduced during the fabrication process. Since the array of processors occupy most of the silicon space, defects could render the entire chip useless. The slicer block makes a hard decision based on the LLRs from the last processor. It deciphers the final decoded stream of information-bits. The error counter module allows us to obtain an accurate measure of the BER, upon decoding the information-bits and code-bits.

At the top-level, this chip requires 64 high-speed inputs, 16 low-speed control inputs, 32 power pins and 16 high-speed outputs. Therefore, a 208-pin Pin Grid Array (PGA208) package will be used. We are clocking this design at 125 MHz and this is, to the authors' knowledge, the first ASIC implementation of an LDPC-CC encoder and decoder. Note that the actual speeds reported during synthesis are higher (see Table I).

IV. Synthesis Results

An ASIC implementation of our LDPC-CC encoder and decoder is currently underway and is expected to be fabricated in 2005. Here, we present synthesis results obtained using 'Design Analyzer', a tool utilizing version 2003.06 of Synopsys' Design Compiler.

Upon analyzing, elaborating and compiling the designs, constrained by a given clock period, we obtained the area, power and timing estimates.

It is obvious from Table I that the area and power estimates for the full decoder is approximately 10 times than that of a single processor. Note the maximum operating frequency for a processor and the full decoder are the same. This illustrates the point that our decoder contains only serially concatenated processors, which we stated in the beginning of Section II-B.

TABLE I

Area and power estimates for a register-based architecture of an LDPC-CC encoder/decoder with $M = 128$ (3,6) code at the maximum attainable frequency. NOTE: Power estimates for the single processor and the full decoder are reported at a 125 MHz clock frequency. Only 10 processors were synthesized for the decoder.

Attribute	Encoder	Single Processor	Full Decoder
Area (μm^2)	52588	1006534	10085751
Max. Frequency (MHz)	250	164	164
Power (W)	0.0086	0.39	3.9
Normalized Power ($\frac{mW}{MHz}$)	0.034	3.1	31.2

The power figures stated may actually be overly pessimistic because the compiler uses only the clock to estimate the switching activity of the circuit. However in reality, the power dissipation should be highest for the first processor and should taper off as we reach the j^{th} processor. That is, the switching activity should be lower in processor $j+1$ than in processor j because the decoder tends to converge to the correct result and therefore, fewer LLR bits need to be switched. This effect is also observed in LDPC-BC decoders [1].

Due to limitations in silicon space, we restricted our decoder design to only 10 processors. This still gives us a reasonable performance and is large enough to validate our approach. It must be noted that additional techniques to speed up the design is the subject of ongoing research by the authors.

V. A pipelined version of the LDPC-CC Encoder

To speed up the encoder design, we analyzed the critical path after synthesis and noticed that it resided in the XOR circuitry (see Section II-A).

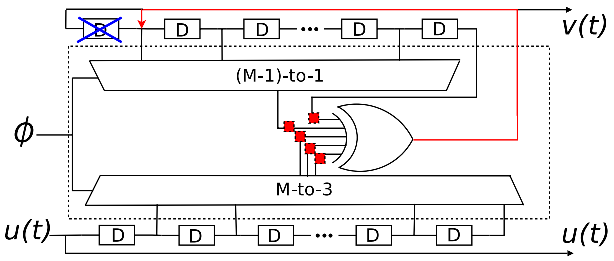

Fig. 4. The architecture of a pipelined LDPC-CC encoder. $u(t)$ is the information bit, $v(t)$ is the code bit and ϕ is the phase for the design. Changes made in the design are marked.

TABLE II
AREA AND POWER ESTIMATES FOR A REGISTER-BASED ARCHITECTURE OF A PIPELINED LDPC-CC ENCODER WITH $M = 128$ (3,6) CODE AT THE MAXIMUM ATTAINABLE FREQUENCY.

Attribute	Pipelined Encoder
Area (μm^2)	51523
Max. Frequency (MHz)	430
Power (W)	0.021
Normalized Power ($\frac{mW}{MHz}$)	0.049

It is possible to modify the encoder structure as shown in Figure 1 to introduce a pipelining stage prior to the XOR gate. Due to the structure of the code, this stage can be compensated for by removing the first delay element in the code-bit delay line (see Figure 4).

The design was synthesized, once again, using version 2003.06 of Synopsys' Design Compiler. The results are stated in Table II. We can note that the area has reduced with the increase in clock frequency. This is because the critical path has been shortened and the surrounding circuitry has been optimized further by the compiler. We also observe that power is proportional to frequency. Hence, the normalized power increases from 0.034 $\frac{mW}{MHz}$ to 0.049 $\frac{mW}{MHz}$.

It is worth noting that, with a single register stage at the input and output of this encoder, it has a latency of $4.65ns$ $(2 \times \frac{1}{430MHz})$ with nearly a two-fold increase in speed.

VI. Conclusions and Ongoing Work

In this paper, we have introduced LDPC-CCs and mentioned that they maybe more suitable for some applications than their block-counterparts. We then focused on implementation issues of LDPC-CC encoders and decoders.

We presented synthesis results which demonstrate a 164 Mbps and 430 Mbps LDPC-CC decoder and pipelined encoder, respectively. This work complements a memory-based approach to LDPC-CC decoder design for FPGAs [4].

The encoder for LDPC-CCs is based on a very simple structure with complexity $O(M)$. In this paper, we realized an implementation with high-throughput, low latency and low power and area figures. This makes LDPC-CCs a potential coding choice for high-speed communication systems. The register-based processor is however, approximately $\times 20$ larger than the encoder. In addition, we require 10-20 such processors in the decoder depending on the required performance. It is obvious that future research should focus upon reducing the area and power figures for the processor.

A circular buffer organization for the LLRs is currently being investigated. In addition, a memory-based architecture is also being investigated, which would be a replacement for the register approach. With the above plans, we hope to be able to reduce area significantly and increase the operating frequency of the circuit.

We are also investigating the parallelization of the design. We plan to develop an implementation that will output n information-bits per clock cycle superseding the one information-bit throughput per clock cycle, for the current design.

References

[1] C. J. Howland and A. J. Blanksby, "A 690-mW 1-Gb/s 1024-b, rate 1/2 low density parity check decoder," *Solid-State Circuits, IEEE Transactions on*, vol. 37, no. 3, pp. 404–412, March 2002.
[2] T. Zhang and K. Parhi, "A 54 MBPS (3,6)-regular FPGA LDPC decoder," in *Proceedings of the IEEE Workshop on Signal Processing*, 2002.
[3] Z. C. S. Bates and X. Dong, "Low-density parity check convolutional codes for packet switching networks," *Submitted to IEEE Communication Letters*, 2004.
[4] S. Bates and G. Block, "A memory-based architecture for FPGA implementations of low-density parity-check decoders," in *Proceedings of IEEE Symposium on Circuits and Systems*, 2005.
[5] A. J. Felström and K. S. Zigangirov, "Time-varying periodic convolutional codes with low-density parity-check matrix," *IEEE Trans. Information Theory*, vol. 45, no. 6, September 1999.
[6] C. Schlegel and L. Perez, *Trellis and Turbo Coding*, 1st ed. IEEE, 2004.
[7] A. Sridharan and D. Costello, "A new construction method for low density parity check convolutional codes," in *Proceedings of The IEEE Information Theory Workshop*, 2002.
[8] S. B. Z. Chen and X. Dong, "Performance of low-density parity-check convolutional codes," 2004, available from http://www.ece.ualberta.ca/∼sbates.