FPGA Implementation of a Clockless Stochastic LDPC Decoder

Chris Ceroici and Vincent C. Gaudet
Dept. of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada N2L 3G1
Email: cceroi@uwaterloo.ca, vcgaudet@uwaterloo.ca

Abstract—This paper demonstrates a clockless stochastic low-density parity-check (LDPC) decoder implemented on a Field-Programmable Gate Array (FPGA). Stochastic computing reduces the wiring complexity necessary for decoding by replacing operations such as multiplication and division with simple logic gates and serial processing. Clockless decoding increases the throughput of the decoder by eliminating the requirement for node signals to be synchronized after each decoding cycle. The design is implemented on an ALTERA Stratix IV EP4SGX230 FPGA and the frame error rate (FER), throughput, and power performance are presented for (96,48) and (204,102) LDPC decoders.

Keywords—stochastic computation, clockless decoding, hardware implementation.

I. INTRODUCTION

Low-Density Parity-Check (LDPC) codes are a family of forward-error-correcting (FEC) codes that are capable of operating near the Shannon capacity limit, and are used in many communications standards such as IEEE 802.16e (WiMax) and IEEE 802.11n (WiFi) [1,2]. Typically, hardware LDPC codes are decoded using the sum-product algorithm (SPA), an iterative soft decoding algorithm [3]. The SPA involves passing a probability, or soft-valued message known as a Log-Likelihood Ratio (LLR), between two sets of nodes, variable nodes and parity-check nodes.

Parallel hardware implementations of this decoding method tend to have high wiring complexity as well as large area consumption leading to longer wire lengths [4]. These longer wires introduce a delay that limits the throughput of the decoder. An alternative method of implementation is to perform node calculations in the probability domain using statistics-based signal processing, known as stochastic computing. Stochastic implementations of the SPA are significantly less complex resulting in greater area efficiency. However, node calculations in a stochastic decoder are done serially and therefore a high clock frequency is required to increase throughput. Stochastic LDPC decoders were demonstrated on a Field-Programmable Gate Array (FPGA) in [5], and on an integrated circuit in [4].

A large high-speed clock network results in high power dissipation and may pose routing challenges. Furthermore, the clock speed is limited by the varying wire delays across the interleaver limiting the throughput. Both asynchronous and Clockless implementations of stochastic decoders have been proposed to circumvent this problem [6,7]. Asynchronous decoders use sets of request and acknowledge control signs to eliminate the global clock. Clockless decoders do not require request and acknowledge control signals throughout the interleaver and therefore have a lower wiring complexity. In clockless decoding the LLRs are passed between the graph nodes with the throughput limited only by the wiring delay of the interleaver. Furthermore, because the size of wire delays will vary, the throughput of the asynchronous decoder will be limited by the longest delay while the Clockless decoder will be limited only by the average of the wire delays. A clockless stochastic LDPC decoder was simulated by Onizawa in [8] but to date there has been no hardware implementation.

II. STOCHASTIC DECODING

A. Overview

Stochastic computing is the process of using random numbers to perform calculations. A stochastic representation of a continuous-value probability is a sequence of random bits in which the probability of a single bit being ‘1’ is equal to the probability being communicated. The order of which these bits appear is not important and so there are many different sequences of bits to represent any probability. For example, the probability 0.6 can be represented by the stochastic sequences 10101 or 0110010111. Information is conveyed as the statistical mean of the stochastic stream. One benefit of using this type of probability representation is the simplification of multiplication and division operations. Multiplication of two stochastic sequences can be achieved simply using a single XOR gate while division can be done using a JK flip-flop. The SPA involves many division and multiplication operations between probabilities and so stochastic computing is well suited for this decoding method.

B. Edge Memories and Noise-Dependent Scaling

Two additional methods are applied to maximize the performance of the decoder: Noise-Dependent Scaling (NDS) and Edge Memories (EMs), which were proposed in [9]. NDS is the multiplication of the LLRs received from the channel by a factor proportional to the SNR. This increases the amount of switching activity in the decoder and makes the amount of switching activity similar across different signal-to-noise ratios (SNRs). EMs are memory elements located in the variable nodes whose purpose is to reduce the probability of the node becoming locked in a hold state (or degenerative state), in which there is no switching activity. This is the case where the inputs to a variable node are not all equal. JK flip flops are susceptible to this phenomenon and so EMs are used instead to increase the degree of randomness. When a hold state is detected the output of the node is chosen randomly from the EM thereby increasing the amount of random switching activity. If the node is not in a hold state (regenerative) the
equality bit calculated by the node is taken as the node output and is also stored in the EM.

III. CLOCKLESS STOCHASTIC DECODING

In synchronous decoding the decoder must wait for all nodes to complete their computations before beginning the next decoding cycle. Because of the distribution of wire lengths, the local update timing restriction results in the throughput being limited by the longest wire in the interleaver. In a clockless design each node begins its next calculation immediately after completing the previous one. The result is that the rate of communication along some graph edges are faster than others, creating instances where some nodes use outdated signals for their calculations. However, due to the nature of stochastic computation, where only the statistical means carry weight in the calculation, these small intermediate signals have little effect overall. By neglecting the synchronization constraint the throughput is limited only by the average of the wire lengths. In addition, no handshaking mechanism is necessary in a clockless decoder which reduces the wiring complexity compared to an asynchronous decoder.

IV. FPGA IMPLEMENTATION

Fig. 1 shows a block diagram of the decoder. The clockless stochastic decoder uses zero value code words with additive white Gaussian noise (AWGN) generated on-chip for testing purposes. The AWGN generator and NDS module consist of lookup tables (LUTs) and a set of linear feedback shift registers (LFSRs). The controller initiates the decoding frame by setting the initialization to ‘1’. This triggers the AWGN generator to generate 8-bit noise signals which are then sent through the NDS LUT which both scales the noise signals and also converts them into LLRs.

LLRs are then converted into the stochastic domain through the use of an 8-bit comparator where the second input is connected to an LFSR triggered by a local oscillator. During the initialization phase LLRs are preloaded into the variable nodes. When this is completed, the node operations begin and the probabilities are communicated across the interleaver.

The communication between variable and parity check nodes is entirely clockless and so involves no latches. Decoding continues until all parity check equations are satisfied from each parity check node or until a counter in the controller reaches a certain value which implies an error being detected. At this point the decoding frame ends and the next frame begins.

A. Variable Nodes

Fig. 2 shows the circuit diagram of a 3-input stochastic, clockless variable node proposed in [8]. U classifies the equality bit as either regenerative or degenerative. If the variable node switches from a degenerative to a regenerative state the rising edge of U generates a pulse triggering the EM to store this regenerative bit. While U is ‘1’ the output of the node is simply the equality bit. In the case of a degenerative state the equality bit is ignored and the output is chosen from the EM at a random address.

The EM consists of a pulse triggered 8-bit shift register...
with an output selected by a random address. These random addresses are generated by an LFSR driven by a local oscillator, such as a ring oscillator. During the initialization phase of the decoder the EM registers are preloaded with the LLRs. In practice decoding can also begin with the EMs in a zero state, eliminating the initialization phase but slowing the speed of convergence. However, this is not suitable for a test using only zero code words since the EMs will always start off preloaded with the correct bits at any SNR.

B. Parity-Check Nodes

Fig. 3 shows the circuit diagram of a 6-input parity-check node proposed in [8]. In contrast to synchronous and asynchronous decoders, the propagation delays of the input paths within the PCNs must be matched to prevent the generation of glitch signals. This is because there is no mechanism to wait until the computation finishes and a delay discrepancy would result in a glitch signal being produced at every input change which would affect the variable node calculations. In the PCN shown here, each input connection has at most one direct path to the last set of gates and so no glitch signal is produced.

Fig. 3. Parity-check node circuit with matched delays for all output lines.

V. INTERLEAVER TIMING ANALYSIS

Fig. 4 and Fig. 5 show the estimated wire delays of the connections between the nodes of the (96,48) and (204,102) decoders. This was obtained by analyzing the extracted decoder design’s timing constraints. The two peaks in Fig. 5 are likely due to the two different methods of routing used during synthesis: global routing, a technique used for wires spanning large quadrants of the chip, and local routing, a technique used for smaller wires.

The largest wire delay encountered in the (96,48) decoder is 12.306 ns. Therefore, if a synchronous decoder were to be implemented using this interleaver routing technique the clock speed would be limited to 81.26 MHz. Similarly, the (204,102) decoder would be limited to a 96.28 MHz. This limit on the maximum clock frequency does not apply to a clockless interleaver since the clockless decoder is able to tolerate node calculations involving outdated signals.

VI. RESULTS

The design was implemented on an ALTERA Stratix IV EP4SGX230 FPGA using the Quartus II set of tools. The design occupies 16% of the FPGA’s logic elements for a (96,48) decoder, and 35% for a (208,104) decoder. The asynchronous design was constructed by directly routing node wires to connected nodes. Care was taken to ensure that these logical loops were not simplified by the compiler. In some cases, such as the pulse mechanism in the EM, extra delay elements were added to increase the duration of the pulse such that the rising edge would be detected by the latch. Due to the asynchronous mechanisms and the tolerance for varying routing delays for different graph edges, specific timing constraints for wire delays were not used.

A. Frame Error Rate

Fig. 6 and Fig. 7 show the frame error rate (FER) performance of the (96x48) and (208,104) decoders using both
NDS and EMs compared with numerical simulations. The numerical simulations were conducted using the (non-stochastic) SPA equations assuming an AWGN channel. Calculations were continued until a valid codeword was obtained or until 50 iterations had been reached. The simulated FER was calculated with at least 200 accumulated errors.

**B. Throughput**

A decoding frame ends when either all parity check equations are satisfied or when the counter in the controller reaches a predefined value and an error is declared. For this reason, the throughput will increase by lowering this predefined value at the expense of error correction performance. This dependency is stronger at low SNR where errors are encountered more frequently. The throughput also depends heavily on the size of the EMs since a larger EM will require a longer initialization period to preload it with the initial channel probabilities. However, increasing the throughput by decreasing the size of the EMs will have a negative effect on the FER. The coded throughputs of the (96,48) and (204,102) decoders using a 300 maximum iteration count are shown in Fig. 8.

VII. POWER MEASUREMENTS

Since the throughput varies at different SNRs, so does the power consumption of the decoder. This is because new frames will be initiated more frequently and more test LLRs will need to be generated. The power consumptions of the (96,48) and (204,102) decoders are shown in Fig. 9. Rather than using the power estimation capabilities of the Quartus II design tools, the measurements were made directly on the FPGA board for greater accuracy. This technique was developed by Joyce Li in [10]. Two 0.01 Ω resistors are introduced to the power supply lines of the FPGA which allow for the measurement of the operating currents of the FPGA. A similar measurement is made while the FPGA is operating using minimal logic and this power measurement is subtracted from that of the decoder’s. This allows for a power measurement which does not include the leakage current from unused portions of the FPGA.

The termination criteria for these decoders uses the on-board 50 MHz clock. The controller counts 300 clock cycles before declaring an error.
At higher SNR the decoder converges more quickly. This means that the decoder must reinitialize and generate additional test noise bits more frequently. This additional switching activity is likely what causes the higher power consumption at higher SNR. By dividing the power by the coded throughput, the coded energy-per-coded bit is obtained. This is shown for both decoders in Fig. 10.

The (204,102) decoder has a lower energy-per-coded bit than the (96,48) decoder. This is expected as the coded throughput of the (204,102) decoder, 254 Mbps at 5 dB, is significantly larger than that of the (96,48) decoder, 152 Mbps at 5 dB. In addition, the difference in power consumption is relatively small. At 5 dB, the operating power of the larger decoder is only 0.0219 W larger than that of the smaller decoder. This small increase is likely due to the optimizations made during the synthesis process for the larger design.

VIII. CONCLUSION

We demonstrated a clockless stochastic LDPC decoder implemented on an FPGA. We reported the FER and throughput of both a (96,48) and a (204,102) decoder. This stochastic decoder design has area and wiring complexity advantages over a fully parallel decoder. In addition, the clockless design benefits from the elimination of synchronization requirements and a large global clock network. The (204,102) decoder has a throughput of 318.8 Mbps at an SNR of 5.5 dB and a FER of 3.47 x 10^-7. The energy-per-coded bit of the (96,48) and (204,102) decoders are 1.36 nJ/bit and 0.698 nJ/bit, respectively. While this performance is surpassed by traditional decoders using larger codes, our goal was to demonstrate this proof-of-concept design in order to study its decoding characteristics. In addition, the performance of LDPC decoders depends heavily on the maximum iteration number. Since this design does not have discrete iterations across all nodes, performance comparisons are difficult to make. In future designs, the clockless stochastic LDPC decoder could be optimized for power or throughput. To our knowledge this is the first stochastic clockless decoder implemented on an FPGA.

ACKNOWLEDGMENT

This work was supported by the Natural Sciences and Engineering Research Council, Discovery Grant. The FPGA board was donated by ALTERA.

REFERENCES