University of Alberta

Parallel-Node Low-Density Parity-Check Convolutional Code Encoder and Decoder Architectures

by

Tyler Brandon

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Department of Electrical and Computer Engineering

Spring 2010
Edmonton, Alberta

Examining Committee

Dr. Duncan Elliott, Electrical and Computer Engineering

Dr. Bruce Cockburn, Electrical and Computer Engineering

Dr. P. Glenn Gulak, Electrical and Computer Engineering, University of Toronto

Dr. Ioanis Nikolaidis, Computing Science

Dr. Vincent Gaudet, Electrical and Computer Engineering

Dr. Ivan Fair, Electrical and Computer Engineering

Give a man a fish; he eats for a day.
Teach a man to fish; he eats for a lifetime.
Teach a computer to fish; man becomes the fish.

– modified Chinese proverb

To the mystery of insight
and the beauty of innovation

Abstract

We present novel architectures for parallel-node low-density parity-check convolutional code (PN-LDPC-CC) encoders and decoders. Based on a recently introduced implementation-aware class of LDPC-CCs, these encoders and decoders take advantage of increased node-parallelization to simultaneously decrease the energy-per-bit and increase the decoded information throughput. A series of progressively improved encoder and decoder designs are presented and characterized using synthesis results with respect to power, area and throughput. The best of the encoder and decoder designs significantly advance the state-of-the-art in terms of both the energy-per-bit and throughput/area metrics. One of the presented decoders, for an E_b/N_0 of 2.5 dB has a bit-error-rate of 10^{-6}, takes 4.5 mm^2 in a CMOS 90-nm process, and achieves an energy-per-decoded-information-bit of 65 pJ and a decoded information throughput of 4.8 Gbits/s. We implement an earlier non-parallel node LDPC-CC encoder, decoder and a channel emulator in silicon. We provide readers, via two sets of tables, the ability to look up our decoder hardware metrics, across four different process technologies, for over 1000 variations of our PN-LDPC-CC decoders. By imposing practical decoder implementation constraints on power or area, which in turn drives trade-offs in code size versus the number of decoder processors, we compare the code BER performance. An extensive comparison to known LDPC-BC/CC decoder implementations is provided.

Acknowledgements

Special thanks goes to Zhengang Chen, without whose work this project would not have been possible.

Also, I'd like to thank Amirhossein Alimohammad, Leendert van den Berg, Zhengang Chen, Jason Klaus, John Koob and Ramkrishna Swamy for their contributions to the LP3 chip design.

For their guidance: Duncan Elliott, Bruce Cockburn, Stephen Bates, and Vincent Gaudet.

Thanks also goes to the VLSI and HCDC lab members.

Table of Contents

List of Figures

List of Tables

List of Symbols

ρ	- node-parallelization factor, in this case referring to the parallelization of the variable-node and parity-check node processing.
A priori	- prior knowledge about a population.
ϕ, ϕ'	- phase and the group phase.
AA-LDPC	- Architecture-Aware Low-Density Parity-Check
ASIC	- Application-Specific Integrated Circuit
API	- Application Programming Interface
AWGN	- Additive White Gaussian Noise
BCJR	- Balh, Cocke, Jelinik, Raviv
BER	- Bit-error-rate is the ratio of the number of incorrectly decoded information bits to the total number of transmitted information bits.
BIST	- Built-In Self-Test
BP	- Belief Propagation
Check Node	- The circuitry or method that combines LLRs to determine confidence in the sign value of the output LLR. The check node is also responsible for determining the sign of the output LLRs based on the XOR of the signs of the input LLRs. The check node is defined to have a degree, that corresponds to the number of input LLRs.
circulant matrix	- A matrix where each row is the previous row rotated by one element to the right.
code length	- The maximum potential distance over which symbols can be chosen for input to the parity-check operations.
code memory	- In LDPC-CCs, the shortest sequence of most recent information and parity-check bits that includes all of the bits that can be involved in a parity-check constraint involving the most recent check bit.
Cnode	- see Check Node
clk	- Clock
CRC	- Cyclic Redundancy Check
CVS	- Concurrent Versions System
DE	- Density Evolution

DUT	- Device Under Test
DRC	- Design Rule Check
E_b/N_0	- Energy-per-bit divided by the spectral noise density; also known as the SNR per bit.
encoder node	- the circuitry that is repeated for each row in the code. i.e. if the code length is 1152 then there are 1152 encoder nodes.
encv1, encv2, ...	- encoder versions 1, 2, ...
FEC	- Forward Error Correction
FIFO	- First-In First-Out
fflop	- Flip-flop
FPGA	- Field Programmable Gate Array
H	- Parity check matrix (PCM)
H^T	- Transpose of H
Hamming weight	- number of ones in a binary vector. In a bipartite graph, the number of edges connecting nodes in the two disjointed sets of nodes.
HDL	- Hardware Description Language
HDL2GDS	- An automated digital design flow, developed at the University of Alberta by Tyler Brandon
Helium	- A command line program designed to run tests on the HP81200 developed at the University of Alberta by Christian Giasson
J	- column weight of H^T.
K	- row weight of H^T.
LDPC	- Low-density parity-check
LDPC-BC	- Low-density parity-check block code
LDPC-CC	- Low-density parity-check convolutional code
LFSR	- Linear Feedback Shift Register
LLR	- Log-Likelihood Ratio
LP3	- Name of a LDPC-CC decoder chip designed and tested at the University of Alberta
LVS	- Layout Versus Schematic
m_s	- syndrome former memory, also known as the code memory.
MSMP	- Merged-Schedule Message-Passing
OMS	- Offset Min-Sum
null space	- given a matrix A, the set of all vectors x such that $xA^T = 0$

P&R	- Place and Route
PCM	- Parity check matrix. In the case of LDPC-CCs this is a periodic sparse matrix that describes edges between variable and check nodes (bipartite or factor graph).
period	- The number of phases before the phases start repeating. Also, the number of rows in the PCM.
PF	- Parallelization factor
phase	- refers to the specific row in the PCM. There are 1/T phases.
PLL	- Phase Locked-Loop
PN-LDPC-CC	- Parallel-Node Low-Density Parity-Check Convolutional-Code
puncture	- to throw away symbols from the output of the encoder to effectively raise the rate of a code. The lost symbols are reconstructed at the decoder.
register bank	- a set of registers.
rst	- Reset
RTL	- Register Transfer Level
SNR	- Signal-to-Noise Ratio
SISO	- Soft-Input Soft-Output
SoC	- System on a Chip
SVN	- Subversion, a source file version control system.
switching activity	- probability of a signal transition occurring, from logic 0 to logic 1, per clock period.
T_s	- the period of the code.
T_s'	- the group period, equal to T_s/ρ
TDMP	- Turbo-Decoding Message-Passing, also known as layered decoding
TPMP	- Two-phase Message-Passing, also known as belief propagation
Variable Node	- The circuit or method that sums LLRs from Cnodes to produce another LLR. The degree of the variable node is equivalent to the number of LLR inputs from the check node plus one.
Vnode	- see Variable Node

Chapter 1

Introduction

Forward error correction (FEC) is the method by which redundancy is added to information to protect the information from corruption when transmitted over a noisy channel. To successfully recover the original information, the amount and method of redundancy is adjusted to compensate for the expected level of corruption in the received transmission. If the FEC is sufficient to recover the original information, re-transmission is not necessary.

FEC has found its way into many applications, from hard-drives, to cell-phones to satellite communications. There are two main classes of FEC: block coding and convolutional coding. Block codes encode and protect fixed-length blocks of data, whereas convolutional codes encode and protect a stream of data of arbitrary length by inserting check bits at predictable intervals. Well known FEC code types include: Hamming codes, Reed-Solomon codes, Turbo codes and LDPC codes. Both Turbo and LDPC codes are capable of approaching the Shannon limit. The Shannon limit establishes an upper bound on channel capacity for a continuous-time analog communications channel in the presence of Gaussian noise [4]. The channel capacity is defined as the maximum rate at which digital data can be transmitted without error within a given bandwidth at a given signal-to-noise ratio. The Shannon limit for an analog system is expressed by

$$C = B \cdot log_2(1 + SNR) \tag{1.1}$$

where C is the channel capacity in bits of information per second, B is the bandwidth of the channel in Hertz, and SNR is the signal-to-noise ratio. The SNR can alternatively be expressed as the energy-per-bit divided by the spectral noise density (E_b/N_0), also known as the SNR normalized per bit, where the number of bits is equal to the number of bits represented in a modulation symbol. In BPSK modulation there is 1 bit per symbol. In QPSK modulation there are 2 bits per symbol. The relationship between SNR and E_b/N_0 is given by Equation (1.2).

$$SNR = \frac{E_b}{N_0} \cdot \frac{f_b}{B} \tag{1.2}$$

where f_b is the channel bit rate.

Substituting SNR with E_b/N_0 in Equation (1.1) gives

$$C = B \cdot log_2 \left(1 + \frac{E_b}{N_0} \cdot \frac{f_b}{B}\right) \tag{1.3}$$

"For transimission over a binary-input, continuous-output AWGN channel with BPSK signaling, the Shannon limit (in terms of E_b/N_0) as a function of code rate does not have a closed from" [5]. An important goal of the field of information theory over the past 60 years is to identify code constructions whose error correcting performance approaches the Shannon limit.

LDPC codes have been developed in each of the two FEC types: block codes (LDPC-BC) and convolutional codes (LDPC-CC). As an alternative to the LDPC-BCs, originally developed by Gallager in 1962 [6], LDPC-CCs were introduced in [7]. Well-designed LDPC-BCs and LDPC-CCs have similar capacity-approaching error-correcting performance [7]. However, LDPC-CCs have certain inherent advantages over LDPC-BCs. Compared to LDPC-BCs, LDPC-CC encoder implementations tend to be very simple [8, 2]. On the other hand, for fixed-length data streams, LDPC-CC encoders require the use of encoder termination, which adds to the complexity of the encoder [9]. With respect to power consumption, the added cost of the termination circuitry is at least minimized by the fact that it remains inactive during normal operation and only becomes active at the end of the data stream. LDPC-BC encoders, on the other hand, are inherently more complex than LDPC-CC encoders and require padding to the data stream to fully fill the data capacity of each coded block. For larger block codes this can lead to significant overhead [9]. Overall it would appear that LDPC-CC encoders would especially benefit those applications where one side of the communications link is sensitive to power and transmits far more than it receives. Figure 1.1 depicts the error-correcting performance of a Turbo and LDPC code in relation to an uncoded channel, and the Shannon limit.

This thesis deals with novel hardware encoder and decoder implementations for LDPC-CCs. More specifically, a new class of parallel-node LDPC-CC (PN-LDPC-CC) encoders and decoders is presented that takes advantage of the architecture-aware PN-LDPC-CCs introduced in [1]. Our PN-LDPC-CC encoder/decoder architectures improve the data throughput and decrease the energy-per-bit to the point that our new architectures outperform the state-of-the-art of both LDPC-CC encoders/decoders and LDPC-BC encoders/decoders. This work should demonstrate that LDPC-CC decoders can compare favorably to LDPC-BC decoders in terms of the energy-per-decoded-bit and the throughput/area.

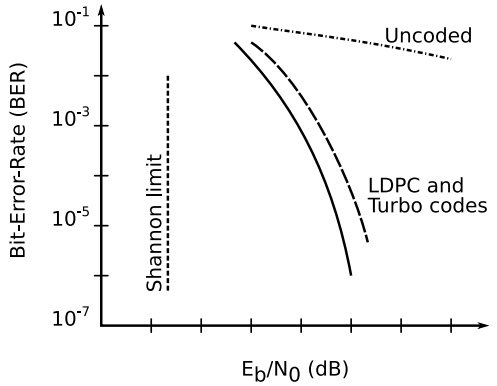

Figure 1.1: The bit-error-rate (BER) versus the ratio of the energy per transmitted bit to the spectral noise density (E_b/N_0, or the signal-to-noise ratio normalized to the number of bits represented per symbol). LDPC and Turbo codes both approach the Shannon limit. As E_b/N_0 increases, the BER typically decreases.

Even the most recent published implementations of LDPC-CCs have not been able to match LDPC-BCs in terms of the energy-per-bit and the throughput/area [10]. While many LDPC-BC encoder and decoder implementations have been described in the literature, their convolutional brethren, the LDPC-CC encoders and decoders, have made fewer appearances. This can be partially attributed to the LDPC-BC decoder's ability to be fully-parallelized with only area and routing being an issue [11]. Historically, the lack of node-parallelization in LDPC-CC decoders has limited their throughput [8, 2, 10]. Node-parallelization for time-varying LDPC-CCs has only recently been introduced in [1]. Another aspect of LDPC-BCs that benefits their hardware implementation is that LDPC-BC error-correcting performance is improved by iterating in the time domain; in contrast, LDPC-CC decoders typically require an additional decoder processor to perform each additional decoding iteration in space. The net effect of more iterations is that LDPC-BC decoders suffer lower data throughput, whereas LDPC-CC decoders require more area. As the number of iterations increases, both LDPC-BC and LDPC-CC decoders require more energy-per-decoded-bit.

In this thesis we throughly analyze a sequence of new PN-LDPC-CC encoder and decoder architectures using the common hardware metrics of power, throughput and area, as well as BER performance. We make comparisons with the characteristics of previously published encoder and decoders. Using the competing decoder's process technology, we match their BER performance and E_b/N_0 and then compare the energy-per-bit and throughput/area metrics. Our comparisons cover four different process technologies and 14 different LDPC decoders from the literature. Our PN-LDPC-CC encoders and decoders set new energy-per-bit and throughput/area benchmarks for both LDPC-CC and LDPC-BC implementations.

The rest of this thesis is organized as follows. The background review, Chapter 2, covers the general structure of LDPC-CC encoders and decoders, the variable-node and check-node operations, and the evolution of improvements made to LDPC-CC encoders and decoders. In Chapter 3 we discuss silicon results for a LDPC-CC encoder and decoder that targets high-throughput. In Chapter 4 we introduce a set of novel encoder and decoder ASIC architectures for parallel-node LDPC-CC codes. The new LDPC encoders and decoders are throughly quantified in terms of BER performance, power, area and throughput. The most advanced decoder architecture is throughly analyzed over process technologies, clock frequency, log-likelihood-ratio (LLR) bit-width and BER performance. The new encoders and decoders are shown to set efficiency benchmarks, not only for LDPC-CCs but for LDPC-BCs as well, by comparing them to the state-of-the-art decoders disclosed in the literature. In Chapter 6, a summary of our results is presented and the major conclusions are reviewed.

Chapter 2

Background

LDPC codes have been shown to approach the Shannon limit [12]. LDPC-CCs were first proposed in [7]. LDPC-CCs, compared to LDPC-BCs of the same *code length*, have better bit-error-rate (BER) performance [7]. In fact, LDPC-CCs have similar BER performance to LDPC-BCs with ten times the length [13]. *Code length* refers to the maximum distance between data when viewed from the perspective of the parity-check operations. LDPC-CC BER performance curves do not have the relatively high error floor of Turbo codes at large $E_{b/}N_0$ [7].

LDPC-CCs are linear codes that generate code bits based on parity-check operations [14]. Any given code bit, $v(t)$, is generated using the present and previous information bits, $u(t)$, and previously generated code bits. LDPC-CCs operate on a continuous stream of data bits. Usually the added code bits are interleaved among the original information bits, as with other convolutional codes.

An LDPC-CC is defined as the null space of a parity-check matrix, which is the same way in which other linear codes (including LDPC-BCs) are defined. If $\mathbf{H}$ is the parity-check-matrix (PCM), then all valid codewords $\mathbf{v}$ satisfy $\mathbf{vH}^T = \mathbf{0}$. For an LDPC-CC of rate b/c $(b < c)$, the corresponding infinitely-large parity-check matrix can be written as

$$\mathbf{H} = \begin{bmatrix} \ddots & \ddots & & & 0 \\ \mathbf{H}_{m_s}(t) & \cdots & \mathbf{H}_0(t) & & \\ & \ddots & \vdots & \ddots & \\ & & \mathbf{H}_{m_s}(t+m_s) & \cdots & \mathbf{H}_0(t+m_s) \\ 0 & & & \ddots & \vdots \quad \ddots \end{bmatrix}, \tag{2.1}$$

where $\mathbf{H}_i(t)$ $(i = 0, 1, \cdots, m_s)$ is a sub-matrix of size $(c-b) \times c$. The parameter m_s in (2.1) is called the *code memory*, not to be confused with the memory definition used in digital circuits. The *constraint length* of the rate-b/c LDPC-CC is defined as $\nu = c \cdot (m_s + 1)$. For *regular* LDPC-CCs, both the column

weight (K) and the row weight (J) of $\mathbf{H}^T$ are constant. We will call such a code an (m_s, J, K) LDPC-CC.

If the parity-check constraints defined by each row of $\mathbf{H}$ are fixed, then the LDPC-CC is called *time-invariant*. Otherwise, the code is a *time-varying* LDPC-CC. In general, time-varying LDPC-CCs can achieve better BER performance than time-invariant codes. Let the *period* be T_s, then $\mathbf{H}_i(t) = \mathbf{H}_i(t + T_s)$ for all i and t. For both encoding and decoding, due to the periodicity of $\mathbf{H}$, a *phase* parameter ϕ, $\phi \in [0, T_s)$, is used to specify the parity-check constraints at each instant in time. Given any time $t \in [0, \infty)$, $\phi = t \mod T_s$.

LDPC-CC encoders add parity-check bits to a stream of user data in accordance with $\mathbf{H}$. The encoded data, both user data and parity-bits, is transmitted though a channel. For evaluation purposes the channel consists of additive white Gaussian noise (AWGN). The noise corrupts the data stream in accordance with the power of the noise relative to the transmit signal strength. More specifically, the quality of the signal is expressed as the ratio of the energy-per-transmitted bit versus the spectral noise density (E_b/N_0). At the decoder, channel samples are scaled and quantized into log-likelihood-ratios (LLRs). In the case of LDPCs, the LLR is defined by Equation (2.2).

$$LLR_i(y) = ln\left(\frac{P(x_i = 0|y_i)}{P(x_i = 1|y_i)}\right) \tag{2.2}$$

where $P(x_i = 0|y_i)$ is the probability that x_i is zero given the known value of y_i. In the case where the data stream is transmitted using the values -1 or 1, the additive noise from the channel corrupts these values and thus creates a Gaussian distribution around -1 and 1, as illustrated in Figure 2.1. The greater the noise, the greater the standard deviation of the signal distributions. Those data values, originally transmitted as -1 or 1, that are offset by the noise and cross the mid-level 0, become errors in the data stream. A decoder's ability to correct errors is determined by a number of factors, which include: the theoretical code strength, channel sample scaling, LLR precision, LLR range, the number of decoder iterations and the check-node operation. The decoder attempts to correct errors in accordance with $\mathbf{H}$. The BER performance measures how well a decoder corrects errors in the received data. The BER performance is often represented as a graph, plotting how the BER declines with increasing E_b/N_0. As the energy-per-bit is increased relative to the spectral noise density (commonly called the signal-to-noise-ratio per normalized bit), fewer signal detection errors are caused by the noisy channel. Given a fixed E_b/N_0, a decoder's ability to correct errors within the data stream will determine its BER. If all the errors created by the channel noise are corrected by the decoder, the output stream from the decoder will match the original user data stream.

In a (m_s,J,K) code, the column weight (K) determines the number of LLRs entering the check-node from the variable-nodes. A common check-node op-

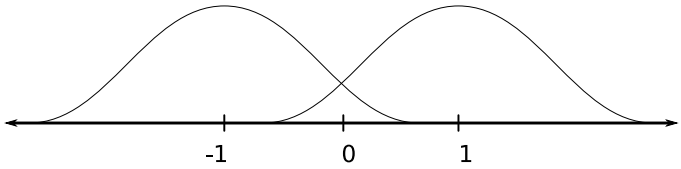

Figure 2.1: Transmitted -1s or 1s corrupted by noise form Gaussian distributions around -1 and 1. Those values that get bumped by the noise and cross 0, become errors in the data stream.

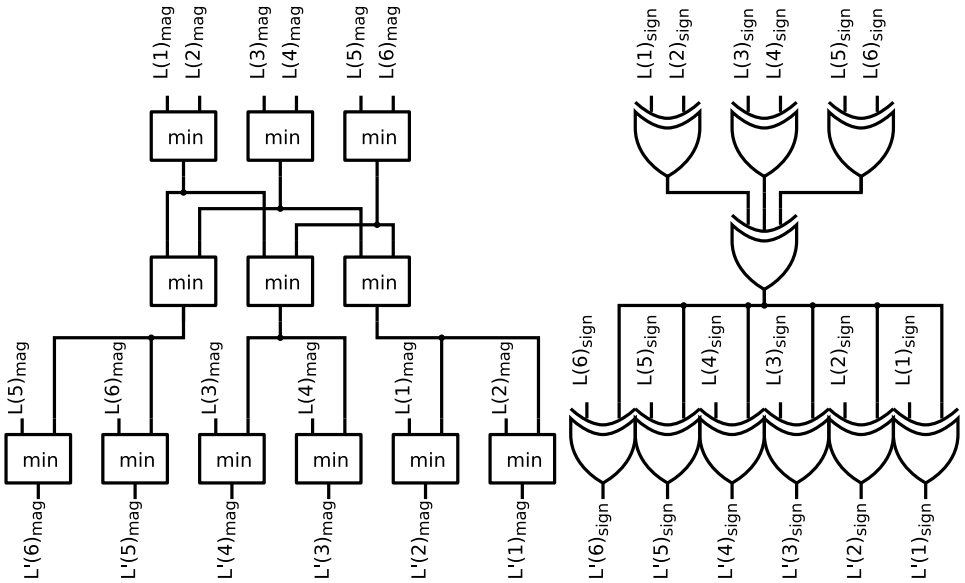

Figure 2.2: Check-node operation, when K = 6.

eration is given by equation (2.3) [7].

$$L'(k) = sign\left(\prod_{\substack{k'=1 \\ k' \neq k}}^{K} L(k)\right) \times \min_{\substack{k'=1,\ldots,K \\ k' \neq k}} |L(k')| \tag{2.3}$$

where $L'(k)$ is the k-th check-node output, $L(k')$ is the k'-th check-node input, where $k' \in \{1, \cdots, K\}$ and $k \in \{1, \cdots, K\}$. The sign bit of the k-th output of the check-node is the XOR of the input sign bits, excluding the k-th input's sign bit. The k-th check-node output's magnitude is the minimum of the input magnitudes, excluding the k-th input's magnitude. There are b check-nodes per decoder processor. Figure 2.2 depicts a check-node, where K = 6.

In a regular (m_s,J,K) code, the row weight (J) determines the number of LLRs entering each variable-node from the check-nodes. In the variable-node, for each LLR input, an LLR output is generated that is the sum of the original un-altered noisy channel sample and the other LLR inputs. The un-altered channel sample is output from the variable-node, un-altered. If the signs of the LLRs in the variable-node operation are all equal, then the output value

becomes more certain, as its magnitude is increased. If the signs of the LLRs in the variable-node operation differ, the output value becomes less certain. The magnitude of an LLR thus represents the probability, confidence or likelihood of the sign-bit being correct. The LLR sign-bit represents estimated the binary value of one bit in the stream, 0 or 1.

The variable-node operation is given by equations (2.4,2.5) [7]. This case shows a 1/2 rate code, where one information and one parity bit is decoded per phase ϕ.

$$L(0) = L'(0) \tag{2.4}$$

$$L(j) = L'(0) + \sum_{\substack{j'=1 \\ j' \neq j}}^{J} L'(j') \tag{2.5}$$

where $L(0)$ is the unaltered channel sample, L is the output from the variable-node, $j \in \{1, \cdots, J\}$ is the number of variable-node outputs (excluding $L(0)$), L' is the input to the variable-node from the check-nodes. There are c variable-nodes per decoder processor.

One row and one column are processed per phase ϕ. Rows are processed before columns. The J 1's in the **H** rows determine the LLR inputs to the check-nodes, and the K 1's in the **H** columns determine the LLR inputs to the variable-nodes.

The amount of LLR storage per processor is given by equation 2.6.

$$M_{LLRs} = (J+1) \cdot c \cdot T_s \cdot q \tag{2.6}$$

where M_{LLRs} is the number of LLR storage bits per processor, J is the row weight, c is the number of variable-nodes per processor, T_s is the code period, and q is the number of bits per LLR. For a (128,3,6) LDPC-CC decoder with 5-bit LLRs, the amount of LLR storage per processor is 5,120 bits.

There are a number of modifications to the min-sum check-node operation to improve BER performance [15], which include $normalizedBP-baseddecoding$ and $offsetBP-baseddecoding$. The $normalizedBP-baseddecoding$ divides the resulting minimum magnitude by a constant. The $offsetBP-baseddecoding$ subtracts a constant value from the minimum magnitude. Significant work goes into calculating the values of the normalized and offset terms [15].

In most cases, LDPC-CC encoding is applied to finite-length data streams. Encoder termination is used to ensure that the trailing bits at the end of the data stream are as well protected as the earlier bits. It has been determined that the size of the termination circuitry usually dominates the encoder complexity [9]. For a detailed discussion of parallel all-phase termination see [1].

For LDPC-CC decoders, the design effort can be focused to optimize one processor. Then a suitable number of copies of that processor can be concatenated together to form a decoder. Increasing the number of decoder processors generally improves the error correcting performance.

Hardware implementations have three common metrics for design comparison: area, power and throughput. Less common, but arguably more important, hardware metrics include the energy-per-bit and the throughput/area. Other hardware metrics include the clock frequency and the latency.

For LDPC encoders and decoders we define throughput as the information or user data rate. Throughput can be calculated as the number of information bits, excluding parity-bits, produced per clock cycle, multiplied by the clock frequency.

Area is typically presented in one of three ways: standard cell area, core area and chip area. The standard cell area is the sum of all the areas of the standard cells within a gate-level netlist. The gate-level netlist is a technology process specific representation of the design using the provided technology logic gates. The technology logic gates are a library of logical functions that have associated physical representations that are manufactureable as semiconductor circuits. The core area is the area required to place and route the standard cells from a gate-level netlist. Placement consists of instantiating a cell within a given area, such that the cells do not overlap with any other cells and such that the cell is relatively close to the other cells to which it will be connected. Routing is the process of "wiring-up" the design by connecting the standard cells together with physical representations of metal strips. The chip area includes the place and route area plus the area required for input-output cells used to communicate off-chip. A rule of thumb we use is that the core area is approximately 70% of the chip area and that the standard cell area is approximately 75% of the core area.

Power, like area, can be presented in three forms: measured power, post place and route power estimate, and synthesis power estimate. Measured power takes the chip operating voltage and multiplies it by the measured chip current consumption. Synthesis power estimates the power based on the switching-activity of a gate-level netlist combined with the standard cell library power consumption information along with operating conditions and wire-load models. Switching activity specifies how often a transition is made from 0 to 1 for a given net in the design. Standard cell library power consumption information consists of power consumed for various input transitions as well as drive strengths and input gate capacitances. The operating conditions specify different process technology corners and operating voltages. Wire-load models are generalized estimates of the parasitics due to the routing and scale with the size of the design. Post place and route power estimates are the same as synthesis power estimates, with the main difference being that extracted wiring parasitics are back-annotated onto the gate-level netlist.

Energy-per-bit and throughput/area are metrics of efficiency. The energy-per-bit takes the power consumption and divides it by the throughput. In this way, we can compare the power efficiency between various designs. Power consumption by itself can be misleading. For example, a circuit that consumes 100 mW and outputs 10 Mbits/s of information consumes less power but is less energy efficient than a circuit that consumes 1 W and outputs 1 Gbit/s of information. In the first case the energy-per-bit is 10 nJ; in the second case the energy-per-bit is 1 nJ. Throughput/area gives us a metric for how well the area is being used. Similar to power consumption, providing throughput or area by itself can be misleading.

Latency is the time it takes for information to travel through a design. Latency is the number of clock cycles information takes to propagate through a design, multiplied by the clock period. Latency only becomes an issue if it becomes too large for a given application.

Clock frequency, in many cases, is a secondary metric that can be used by designers familiar with the process technology to gauge how the design may be altered to better optimize target hardware metrics. If an unexpectedly high clock frequency is achieved, then power can be saved by lowering the operating voltage while sacrificing throughput and latency.

2.1 Evolution of LDPC-CC Encoders

LDPC-CC encoders are distinguished by their relatively simple implementations [8]. The classic serial-node direct encoder is described in [8, 16]. Figure 2.3 shows a LDPC-CC serial direct encoder [8]. M bits of history of the information and parity bit stream, $u(t)$ and $v(t)$, are stored in the bottom and top rows of flip-flops, respectively. The current parity-bit is the XOR of five values, two from the history of $v(t)$ and three from the history of $u(t)$. The phase ϕ determines which five historical values are selected.

We present a high-throughput serial-node LDPC-CC encoder in [2]. This design improves the power consumption and throughput of the encoder shown in Figure 2.3. The shift chains and multiplexors in 2.3, are replaced with addressed flip-flops and a pipelined XOR tree. Figure 2.4 shows the resulting high-throughput encoder [2], while Figure 2.5 shows the corresponding termination circuitry [2]. The result is an encoder with termination implemented in 62,304 μm^2 in a CMOS 90-nm process, capable of operating at 1.1 GHz with a power consumption of 23 mW with a 1.2 V supply.

A parallel encoder without termination is presented in [17]. This encoder generates ρ parity-bits in parallel. The design does not alter the parity-check matrix and the parity-check matrix itself does not guarantee that the ρ parity bits within each group will not depend on each other. In the cases where one parity bit is generated based on another parity bit within the same group,

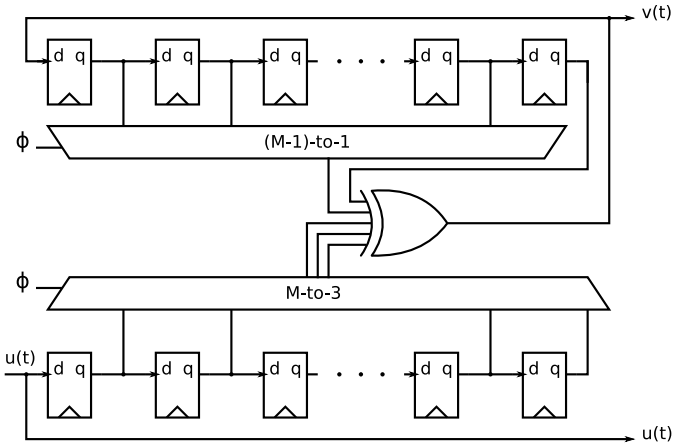

Figure 2.3: LDPC-CC direct encoder.

then parallel processing cannot happen because of the data dependency. In such cases, an internal serialization of the calculation may be needed within a group of ρ parity bits, which increases the latency and lowers the throughput. As ρ grows, this issue becomes more serious [17]. For a (128,3,6) code encoder implementation in a CMOS 90-nm process, with ρ=8, an information throughput of 5.4 Gbits/s is achieved in an area of 80,040 μm^2. For a (2048,3,6) code in a CMOS 90-nm process, with ρ=8, a throughput of 3.6 Gbits/s is achieved in an encoder with an area of 761,900 μm^2.

Altering the original encoder structure, a so-called partial syndrome encoder is presented in [18]. The partial syndrome encoder reduces the amount of storage to equal m_s by updating $J-1$ storage registers per phase ϕ. As the code size increases, the partial syndrome encoder's throughput scales better than the direct encoder, due to the growth of the direct encoder's multiplexors [16]. Figure 2.6 shows a partial syndrome encoder.

For rate-b/c LDPC-CCs, the number of state bits for the partial syndrome encoder is $(cb)/c = (1R)$ of that for the direct encoder, which reduces the termination complexity, where R is the code rate [1].

A partial syndrome parallel-node LDPC-CC encoder is introduced in [1]. The partial syndrome parallel-node LDPC-CC encoder increases the throughput and decreases the energy-per-encoded-bit by processing multiple information bits in parallel. Figure 2.7 depicts a partial syndrome encoder for the architecture aware PN-LDPC-CC codes described in [1].

2.2 Evolution of LDPC-CC Decoders

The first reported ASIC LDPC-CC decoder is presented in [8]. This decoder uses 10 cascaded processors to form the decoder. Each decoder processor contains multiple sets of shift registers for the LLRs. When LLRs reach a

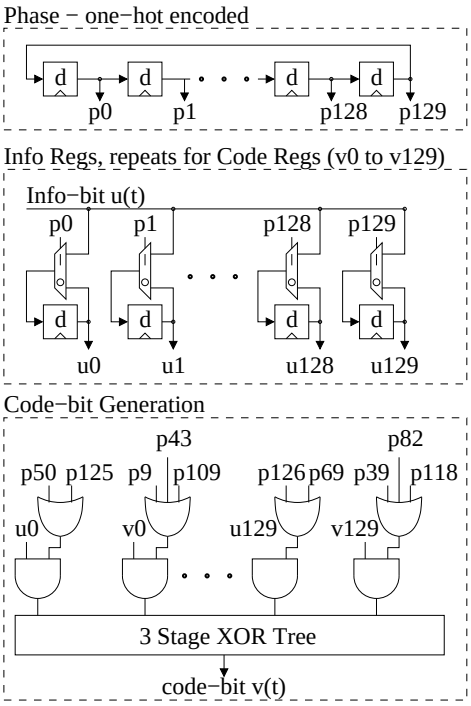

Figure 2.4: A 1.1-GHz encoder. Replacing the shift chains and multiplexors, found in the direct encoder, with addressed flip-flops and a pipelined XOR tree results in a high-speed encoder.

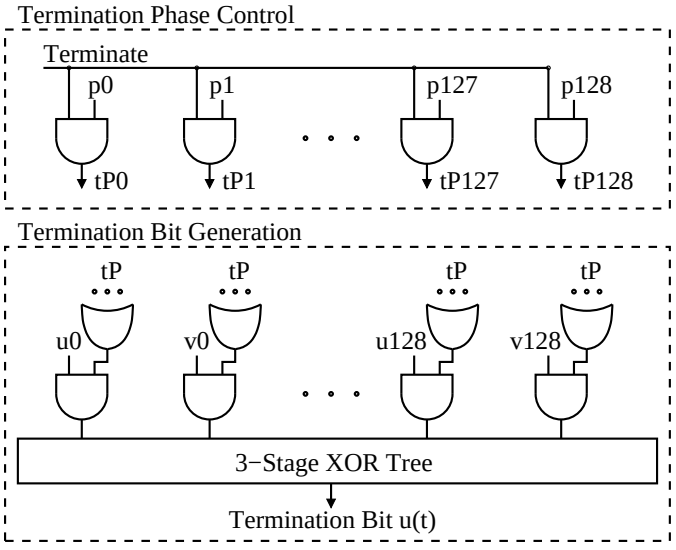

Figure 2.5: A 1.1-GHz encoder's termination circuitry.

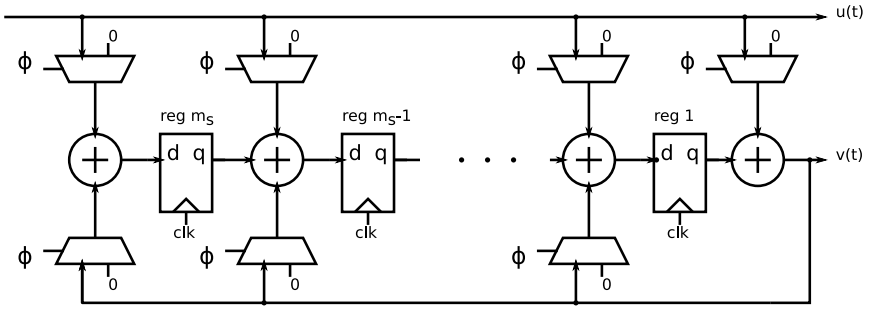

Figure 2.6: Partial syndrome encoder for LDPC-CC codes.

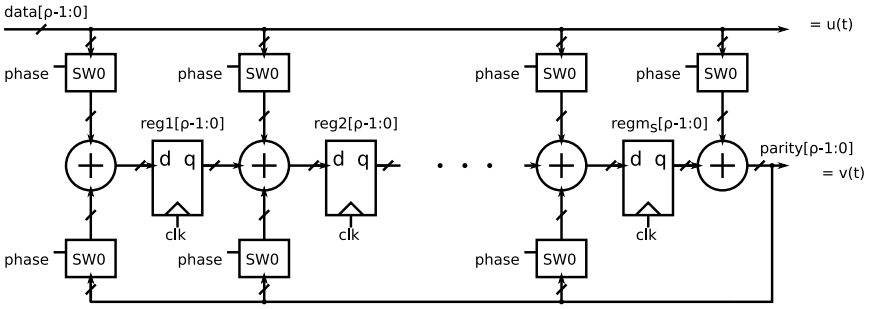

Figure 2.7: Partial syndrome encoder for the architecture-aware PN-LDPC-CC codes with ρ=8 [1].

certain point in the shift chain, as determined by the phase ϕ, they are routed to the parity-check node. The updated LLR outputs of the parity-check node are routed back to their originating positions in the shift chains. The parity-check-node updates each LLR exactly once. When the updated LLRs reach the end of the shift chain they go through the variable-node operation and on to the next processor.

A highly-pipelined LDPC-CC decoder is presented in [2]. Instead of energy-inefficient shift chains, as in [8], a memory structure is used. In order to facilitate conflict-free memory access, the number of memory banks is increased and the LLRs are assigned specific memories based on their access pattern. A relatively complex switch matrix is needed to connect the relatively large number of memory banks to the check-node unit and the variable-node units. The decoder processor has 7 pipe-line stages, allowing the decoder to run at 600 MHz. While these pipeline stages allow the decoder to reach higher throughputs, the overhead associated with the extra registers, in terms of area and power, is significant.

A serial LDPC-CC decoder is presented in [19]. This decoder uses a processor combined with memory to store a frame of soft data as it is being iteratively processed. Each iteration consists of the frame being read from memory, fed through the decoder and then written back to memory. The decoder achieves a throughput of less than 10 Mbits/s with a maximum of 100 iterations and occupies 15% of a Altera Stratix EP1S80 for a length-2048

code. The authors show that their design uses a very small number of logic elements, and claim that their design reduces "memory consumption" and is suitable for large LDPC-CC codes.

A tail-biting LDPC-CC flexible decoder, that is programmable like a specialized microprocessor, is presented in [10]. Consisting of memory, cross-bar switches, vector arithmetic units and very long instruction words, this LDPC-CC decoder is capable of processing codes of different lengths and rates. The processor takes 1152K gates, consumes 370 mW and achieves a throughput of 100 Mbits/s for 10 decoding iterations.

We present a parallel-node LDPC-CC decoder in [1]. The node-parallelization factor ρ is equal to the number of information channel samples processed in parallel. To achieve a high throughput, the memory architecture is designed to complete the node update cycle in one memory cycle. The decoder achieves a single memory cycle per phase by using a combination of single and dual-port asynchronous memories. The parallel-node decoder attains relatively high throughput while lowering the energy-per-decoded-bit. In addition, the decoding latency is also reduced significantly due to the node-parallelization.

2.3 LDPC-BCs versus LDPC-CCs

The decoders for both LDPC-BCs and LDPC-CC use parity-check nodes and variable nodes; however, the number of nodes and how they are connected differ [13].

Compared to LDPC-BCs, LDPC-CCs can achieve similar BER performance using a code length of approximately an order of magnitude less [13]. However, LDPC-BCs and LDPC-CC have the same computational complexity [13]. The theoretical memory storage for LDPC-BCs and LDPC-CCs is equal when the LDPC-BC code length (N) is equal to the LDPC-CC constraint length times the number of iterations ($\nu * I$) [13]. A comparison of LDPC-BC and LDPC-CC decoder hardware is attempted in [13], but in every case there are exceptions and caveats that complicate the argument. In the comparison a number of challenges became apparent: some decoders are implemented in FPGAs (of differing types), others in ASICs of various fabrication technologies, all with different BER performance with each decoder targeting various hardware performance metrics. Prior to this thesis research, a comprehensive or meaningful comparison of LDPC-BC and LDPC-CC decoders was not available.

For LDPC-BCs, the iterative message passing decoding algorithm based on the code graph representation (called a Tanner graph) leads to a natural parallelization in the decoder design. Multiple nodes in the graph can be physically realized. Figure 2.8 depicts a Tanner graph with check-nodes and variable-nodes. The parallelization in the node dimension signicantly increases

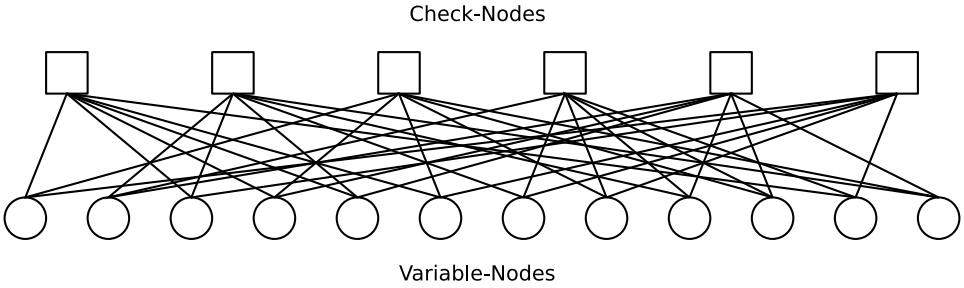

Figure 2.8: An example of a bipartite or Tanner graph for a regular LDPC-BC block code. The bipartite graph is regular because the number of connections to the check-nodes is the same for all the check-nodes.

the decoding throughput. However, fully-parallel LDPC-BC decoders tend to have a high silicon area cost in a VLSI implementation due to the large number of physically implemented nodes as well as the routing between nodes. Large LDPC-BCs can fall victim to routing congestion [11], which requires an increase in silicon area to fix. Many different approaches have been proposed to address the LDPC-BC design challenges, including partially-parallel decoders [20, 21] and layered decoders [22]. These approaches partition the decoding problem into stages rather than tackling a full decoder iteration all at once; the goal of partitioning is to achieve a better trade-off between area, throughput and power.

Conventional LDPC-CCs are not easily parallelized in the node dimension, due to the serial nature of the node processing [7]. This limits their implementations' throughput, compared to LDPC-BC implementations that can parallelize the node operations [11]. Recently new LDPC-CCs have been developed that allow parallelism in the node dimension [10, 1]. This thesis will explore LDPC-CC encoder and decoder implementations for the parallel-node LDPC-CCs (PN-LDPC-CCs) presented in [1]. When code design and decoder design are considered jointly, we will show that more efficient designs are achievable [1].

2.4 Parallel-Node LDPC-CCs

PN-LDPC-CCs were developed at the University of Alberta by Zhengang Chen [1]. These codes are the result of applying coding and hardware constraints. As a result, these codes naturally fit into efficient hardware implementations, while maintaining good BER performance. We will talk about the origins of PN-LDPC-CCs, give their BER performance and comment on the factors impacting BER performance.

The BER performance of a code, typically with an additive white Gaussian noise (AWGN) channel, allows us to compare it to other codes. The PN-LDPC-CCs have good BER performance compared to other LDPC codes [1].

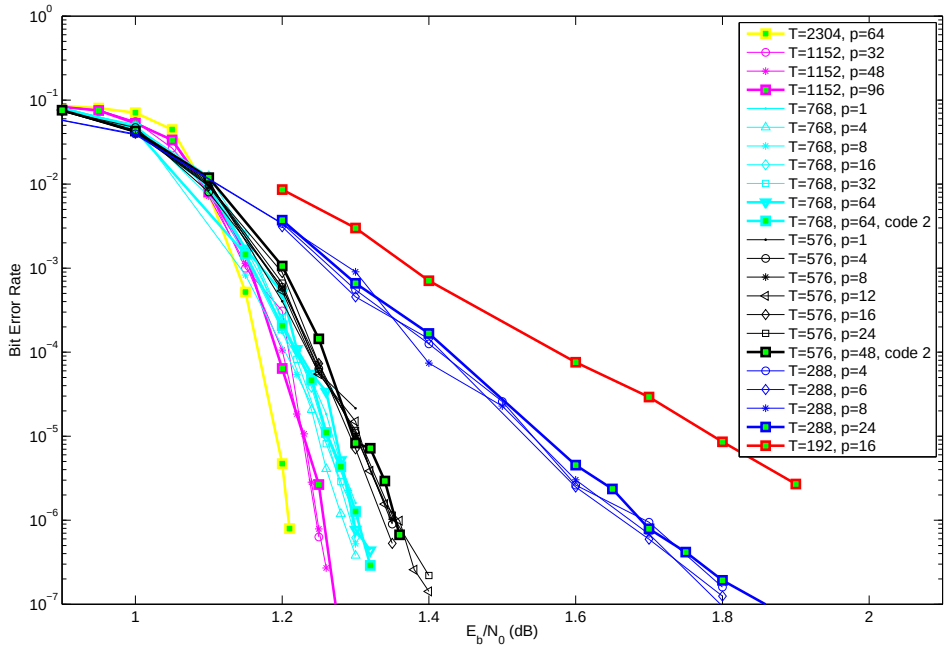

Figure 2.9: BERs for the PN-LDPC-CCs. Codes and BER performance data provided by Zhengang Chen [1]. Note the relative independence of the BER performance with ρ. Codes with the same T_s (T in the plot) and different ρ have similar BER performance.

Figure 2.9 shows the BERs for our set of PN-LDPC-CC codes, constructed by Zhengang Chen using the method presented in [1]. Two important factors of PN-LDPC-CCs are the code length, T_s, and the node-parallelization factor, ρ. In the figure, codes with the same T_s have the same shade. It is interesting to note that the ρ can be increased significantly with little impact on the BER performance of the code. The length of the code (T_s) is the dominant factor governing its BER performance. This is seen in the figure, by the grouping of lines of the same color. 1000 bit errors are gathered before calculating each BER point on the BER performance curves. The shown codes were chosen by trial and error. Typically, a number of codes of the same T_s and ρ are generated and run though simulations, and the best performing code is chosen. To recap, the BER performance of a code is significantly impacted by the length of the code (T_s).

The method of decoding also plays a large role in the BER performance. Figure 2.10 shows the impact of fixed-point versus floating-point and the effects of a variable number of iterations on BER performance. In the case of the T_s=480 code, variations in the number of iterations and the LLR bit-width, dramatically impact the BER performance. For a BER of 10^{-6}, using the T_s=480, ρ=8 code, compared to the ideal case, the fixed-point decoder using 30 iterations and a 6-bit LLR, loses an E_b/N_0 of 0.75 dB. To summarize, the major factors effecting BER performance are T_s and the decoding method.

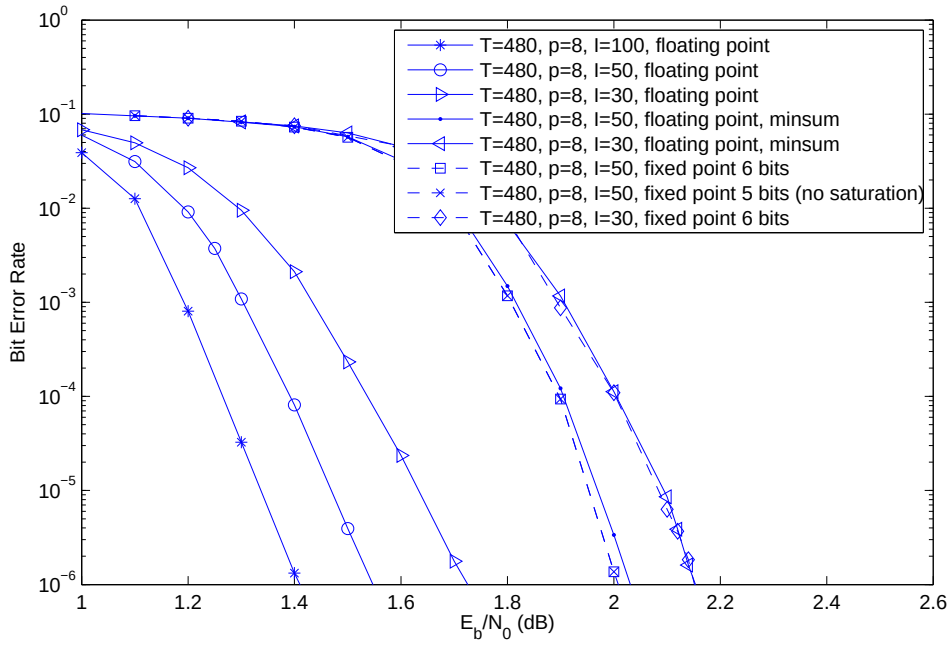

Figure 2.10: BER performance for the T_s=480, ρ=8 code for various bit-precision and iterations. The bit-precision and number of iterations has a large impact on the BER performance. We generated this figure using Zhengang Chen's BER data.

Taking a closer look at the clusters of T_s=768 codes reveals that the ρ has little predictable impact on BER performance. Figure 2.11 plots the BER of T_s=768 codes versus ρ for various values of E_b/N_0. Increasing ρ, to the limit that we are capable of generating, does not negatively impact the BER. While there are deviations in BER performance, no trends relating to ρ are present. Both high-ρ and low-ρ codes have similar BER performance. The same holds true for T_s=576 codes. Figure 2.12 plots the BER of T_s=576 codes versus ρ. Again, there is no observable relationship between ρ and the BER performance. This same relationship hold true for the other codes.

In Chapters 4 and 5, novel encoder and decoder architectures are presented for these PN-LDPC-CCs.

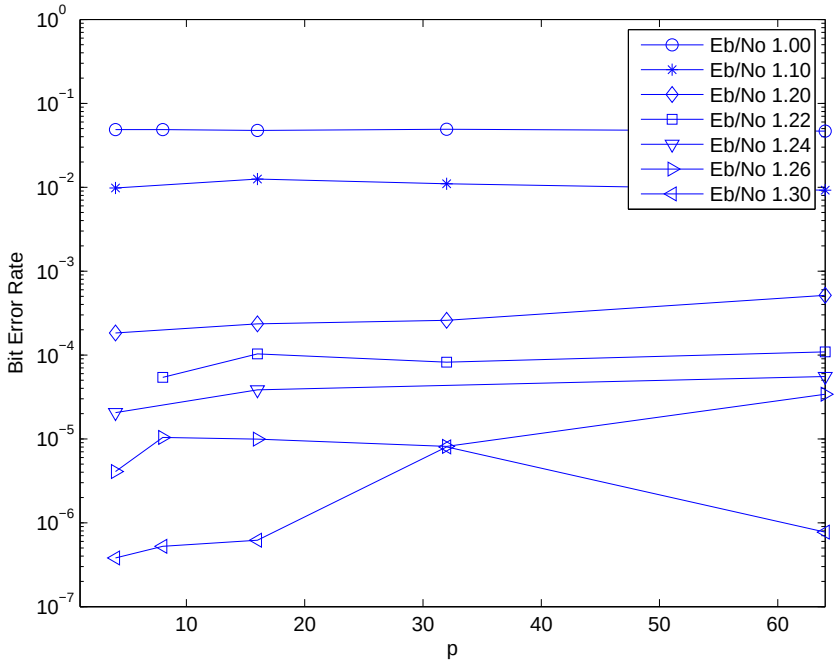

Figure 2.11: BER versus ρ for T_s=768 codes. Increasing the node parallelization ρ does not negatively impact the BER. We generated this figure using Zhengang Chen's BER data.

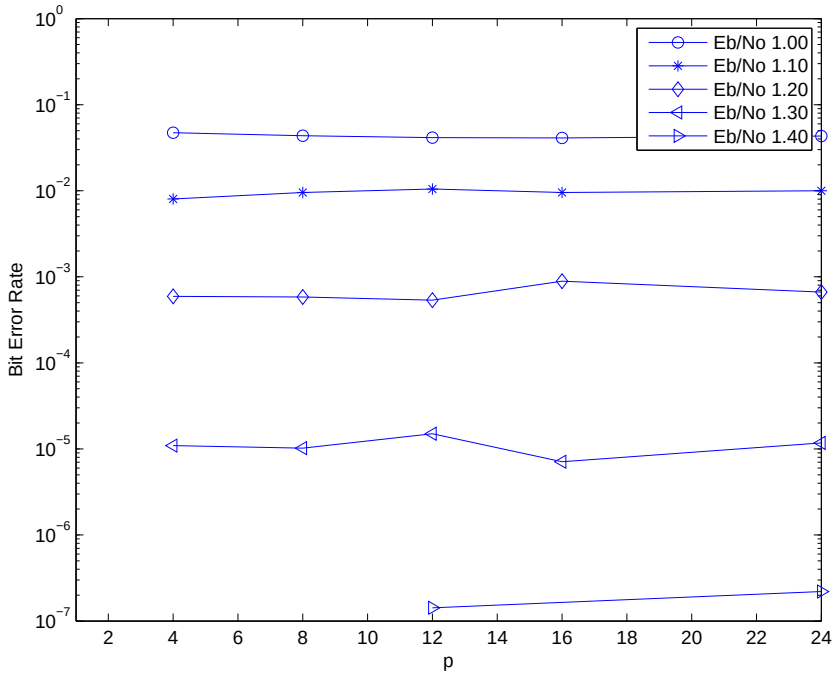

Figure 2.12: BER versus ρ for T_s=576 codes. Increasing the node parallelization ρ does not negatively impact the BER. We generated this figure using Zhengang Chen's BER data.

Chapter 3

A Highly Pipelined LDPC-CC Encoder and Decoder Implementation

3.1 Overview

This chapter describes the first LDPC-CC encoder and decoder design, called LP3, for which I was the principal designer. As later chapters present results based on synthesis results, it is important to present the experimental verification in silicon of synthesis results. LP3 is a relatively complex chip design that pushed the throughput envelope for encoder and decoder LDPC-CC implementations. This work is presented in [2]. The subsequently designed parallel-node LDPC-CC encoders and decoders, presented in Chapter 4 and Chapter 5, build upon the experience gained with LP3 and far surpass LP3 in terms of increased throughput, decreased area and decreased power consumption.

LP3 is a rate-1/2 (128,3,6) LDPC-CC 1.1-Gbits/s encoder and 600-Mbits/s decoder test chip fabricated in ST Microelectonics's 90-nm CMOS logic process (for a list of standard cell libraries used see A.5). LP3 is a self-testing design, capable of generating user-data on chip, encoding the data, adding noise, decoding the data and counting errors. To facilitate high-speed testing, despite our relatively slow 133-MHz digital tester, a PLL is included on-chip to generate the necessary high-frequency clock on-chip. Figure 3.1 depicts the major blocks. The flow of data originates with user data, generated in LP3 by the random number generator. The user data is encoded and redundancy added according to a sparse matrix that defines the "code". The encoded data is passed though an AWGN channel emulator that adds Gaussian noise to the encoded data. The resulting noisy baseband channel samples are then quantized and processed by the decoder. The results of the decoder are compared to the original user data, which is kept saved in a FIFO, and the resulting errors are counted.

There are a number of contributors to LP3. Leendert van den Berg de-

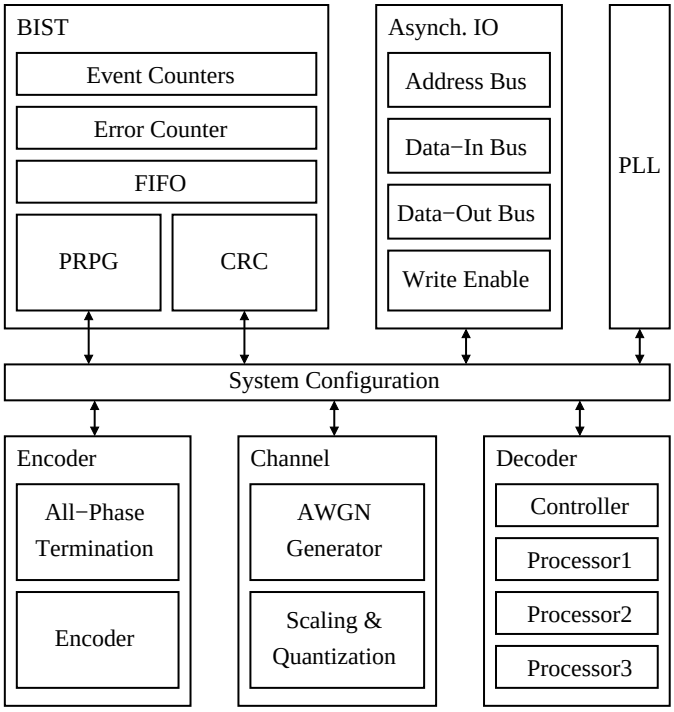

Figure 3.1: LP3 block diagram [2].

signed the PLL. Amirhossein Alimohammad designed the AWGN generator and random number generator. Zhengang Chen contributed to the introduction and provided the sections on the coding theory. Jason Klaus created a graph coloring algorithm that not only minimized the number of colors, but balanced the number of edges in each color thus balancing the sizes of the memory banks and therefore reducing the size of the largest required memory bank. Ramkrisha Swamy helped with specifications, discussions on the architectures and understanding the decoding algorithm. John Koob was one of the designers and the primary editor/writer for the LP3 paper. Both John Koob and Leendert van den Berg helped implement and verify LP3 during tape-out and assisted in testing LP3. I designed the encoder and decoder, integrated all components, designed the built-in self-test (BIST), helped write the LP3 paper and tested LP3. Figures in this section are © IEEE and are used with permission.

EDA tools used include: VCS 2005.06, DesignCompiler 2005.09, Conformal 50, Encounter 41, Formality 2005.12, Virtuoso 51, Calibre DRC 2006.2, Calibre LVS 2006.2, StarXtract 2005.06, and PrimeTime 2005.12.

3.2 The Challenge

The primary goal of implementing LP3 in silicon was to gather power vs. throughput measurements for a new high-speed, low-power LDPC-CC architecture. Every module LP3 was a new design that had never been tested in silicon. This project was therefore a high-risk scenario for the overall functionality of the design. In a conventional design, test features accessed through a serial JTAG interface could be used to test the functionality of individual modules, but in this design a JTAG testing interface would make power vs. throughput measurements difficult to achieve. As LP3 consists mostly of flip-flops, the area and power overhead required for a full JTAG interface would be considerable. Given that every module had the potential for failure, we needed a test method that would do more than just detect failures, we needed to be able to localize any failures by being able to test each module independently of the others. In addition, the LP3 modules were estimated to function correctly in the range of a few hundred MHz to just over a GHz. The available HP81200 tester is only capable of driving signals at a couple of hundred MHz. The available on-chip IO drivers and the re-use of a previous test PCB limits the chip-to-tester IO frequency to a maximum of 100 MHz.

3.3 Meeting the Test Challenge - Design for Test

LP3 has features that allow all internal digital components to be tested independently or together, at high-speed, on-chip and with only the results of the tests being relayed off-chip. This is accomplished with a PLL, an asynchronous memory interface, a custom system configuration module and a number of BIST components, that include a variable length FIFO, counters, a signature generator and an LFSR.

3.3.1 The Channel

The "channel" module consists of an additive-white-Gaussian-noise (AWGN) generator and a scaling function. The AWGN provides high quality noise that is added to the "info" and "code" bits produced by the encoder and is capable of achieving an range greater than 6σ. Noise is scaled according to the SNR value set in the configuration registers and added to the "info" and "code" bits. The combined signal and noise is then scaled by a linear function and quantized to 8-bit LLRs.

To eliminate the need for an external noise generator, a compact and accurate white Gaussian noise generator, as described in [23], was implemented on-chip. The AWGN provides two 16-bit noise samples per clock cycle. Accurately distributed and uncorrelated noise samples are important when verifying the behavior of very low BER systems, such as LDPC coded systems [23]. In

addition, the noise samples should be generated as fast as possible to keep up with other blocks in the system.

3.3.2 Random-Number-Generator

Instead of a using a conventional linear feedback shift register (LFSR) to generate random digital data for stimulus, a combined Tausworthe pseudo-random-pattern-generator (PRPG) was used to generate random numbers for testing on-chip modules. Without sufficiently good randomness properties, a set of generated vectors might never excite certain hard-to-detect faults [24]. We implemented a three-component Tausworthe generator to improve the randomness properties of the produced numbers [23, 25].

3.3.3 The Phase-Locked-Loop

To facilitate high speed testing, a custom phase-locked-loop (PLL) is included on-chip that allows us to create on-chip clock frequencies in the range of 100 MHz to over 2.5-GHz. The IO drivers on-chip are limited to a maximum toggle rate of 100-MHz. The PLL thus helps to make high-speed on-chip testing of the core modules possible.

3.3.4 System Configuration

The system configuration registers are used to configure all aspects of the design, including initialization variables, data-flow between modules, module configurations and timing events. The system configuration module is at the heart of the modularity and configurability of the LP3 design. This module is the hub of all on-chip inter-module data communication. The timing of every event within the LP3 is controlled from the system control registers. All module inputs and outputs are fed through this module. There are three types of control registers and two sets of addresses in this module. The first set of addresses correspond to control registers. The second set of addresses correspond to the system modules. The module addresses include the inputs and outputs of each module (and in some cases to sub-modules). The three types of system configuration registers include: system control registers, event timing registers and module configuration registers.

The system configuration registers are the same size as the off-chip databus, 16-bits. Multiple system configuration registers can be used together to form larger values. System configuration registers are written with the off-chip input-databus when the write-enable signal is asserted. The off-chip input and output databusses behaves like any other module in the design.

The system control registers control the flow of data between modules. The system can be configured such that almost any given module can be connected

with any other module. The interconnections are facilitated with multiplexors. The inputs to the multiplexors consist of the other module outputs and a zero-state register. The zero-state register allows the user to pass zeros into a module. The value of a control register determines which module output will go to a module's input.

System configuration timing-event-registers are used to control timing events. The most common event is the "reset" event. Each module has a reset signal. The module reset signals are tied to an associated event signal that is activated when the system configuration counter reaches the value stored in a timing-event-register. The off-chip "reset" signal only controls the system configuration reset. The assertion of the off-chip reset sets the system configuration registers to their default values.

The module-configuration-registers store configuration setting for the modules. For example the AWGN module takes a 128-bit seed value. This seed value is stored across 8 module-configuration-registers.

When the LP3 chip is reset, it enters an idle state. At this time the user can configure the system control registers to setup a desired test. Once the system control registers are set, an additional control register is cleared to start the system configuration counter, which will, in turn, trigger other events and thus perform the test.

3.3.5 Asynchronous Memory Interface

To simplify tester communication with the chip, we chose a memory style interface. The memory interface consists of an 14-bit address bus, a write-enable signal, a 16-bit input databus and a 16-bit output databus. A major advantage of this memory interface is that it can be easily made asynchronous. Nothing happens inside the chip unless the write-enable signal is asserted. The input databus and address bus are free to toggle and glitch, but the chip will remains unaffected until the write-enable signal is asserted. To write data into the chip, the user should wait until the address bus and input databus have settled then assert the write-enable signal. In this way the external clock and internal clock do not need to be synchronized in any way. Multiple writes of the same value to the same address location have no effect on the state of the circuit. The value of the output databus is solely dependent on the value of the address bus. Two points are important to emphasize. First, that only the 1-bit write-enable signal controls the latching of data into the chip. And second, provided that the address bus and input databus signals remain stable, a single write or multiple writes makes no difference to the internal state of the design. LP3's asynchronous interface makes getting data in and out of LP3 a relatively easy task.

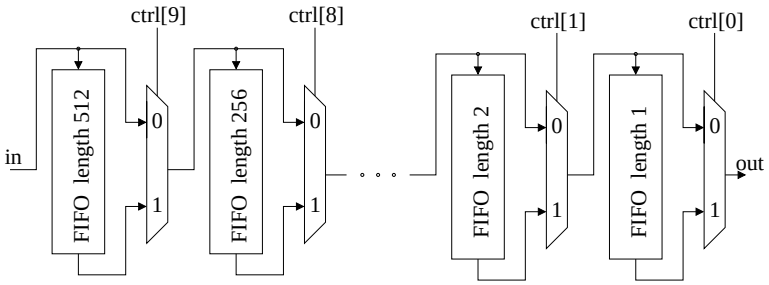

Figure 3.2: LP3 BIST Variable Length FIFO [2].

3.3.6 The BIST Modules

Our BIST module consists of several sub-modules used to test LP3 that include: a CRC, FIFOs, random number generators and counters instead of state-machines.

The LFSR is primarily used as a data generator whose output can be connected to the input of any other module in the design, including the output databus.

The CRC can be used to generate a 32-bit signature (hash) of the consecutive outputs of any module within the chip. The CRC uses the same polynomial as the IEEE 802.3 standard. The CRC can be programed, via the system configuration registers, to stop updating after a fixed number of cycles. Once stopped, the CRC will hold its value, until reset. An on-chip module is deemed to be behaving as expected if its on-chip signature matches its simulation signature.

The FIFO is a programmable variable-length FIFO, capable of buffering 1-bit inputs over a range of 0 to 1023 cycles. The variable-length FIFO is created by concatenating ten FIFOs of the following lengths: 1,2,4,8,16,32,64,128,256, and 512. A 10-bit control register acts as a bit-mask for the FIFOs. If a bit in the bit-mask is a "1" the associated FIFO is enabled, however, if the bit is "0" the associated FIFO is by-passed and input-bit is sent directly to the next FIFO. Figure 3.2 illustrates this configuration. The FIFO's primary use is to delay values from the BIST LFSR so that they can be later compared with the output of the decoder to calculate the BER.

An important difference in our BIST circuitry was the use of counters rather than state-machines. The desired degree of programmability in our design warranted the use of counters rather than state-machines. In LP3, there are a number of modules that required different initialization sequences. For example the encoder needed to be started after 10 cycles, the noise generator after 1 cycle and the decoder after 22 cycles. All of these events are programmable via the system control registers. In addition, there are a number of stop events that send control signals to individual components after a set number of cycles. For example, the encoder can begin its termination sequence

anywhere from 1 cycle to 2^{64} - 1 cycles (an approximate time of 1000 years, running at 1 GHz). The ability for the design to run for relatively long periods of time is important when power measurements are performed as our current-meter takes measurements at 150 ms intervals.

The LFSR and CRC together allow us to perform high-speed testing and take power measurements. The LFSR is a "random number generator" that we use to supply inputs to the various on-chip modules. The CRC is key in testing our design. The CRC generates a 32-bit hash value of the current 16-bit input and the previous 32-bit hash value. We can compare the hash values generated in simulation to the chip hash values generated in silicon to verify that the chip is behaving as expected. For example, if a simulated CRC and the on-chip CRC receive the same 1 million 16-bit values, we would get two identical 32-bit hash values. However, if any bit in the inputs were to be different, then the on-chip and simulated hash values would be different. This allows us to detect errors in any on-chip module. The chance of the simulated and on-chip CRC producing identical hashes from non-identical inputs is $1/2^{32}$ if the CRC is designed appropriately.

3.4 Encoder and Decoder Architecture

3.4.1 Encoder

The LP3 LDPC-CC encoder is a compact design with built-in all-phase termination. LDPC-CC encoder implementations are typically much simpler than an LDPC-BC encoders with similar capabilities, even though both types of encoders add redundancy to a stream of information bits. The LP3 encoder design has been improved compared to the previous implementation presented in [3]. The LP3 encoder was redesigned to consume less energy-per-bit while operating at higher frequencies of up to 1.1 GHz.

The structure of an LDPC-CC encoder consists of phase control, registers for information bits and code bits, and logic for code bit generation (see Figure 3.3). The phase signals are created by one-hot encoding logic. One-hot encoded phase signals reduces the amount of encoder logic that is active in any given cycle and simplifies the encoder logic, which reduces the power consumption and permits a higher frequency clock. As shown in Figure 3.3, the one-hot encoded phase signals control when the previous information and code bits are loaded into the appropriate registers. The current code bit is generated based on the XOR of a specific phase-dependent set of previous code and information bits. The three-stage XOR tree in Figure 3.3 is pipelined to increase the throughput.

Unlike block codes, convolutional codes require a termination scheme to ensure that the trailing information bits at the end of a transmission are protected. Truncating a data stream without termination results in loss of bit-

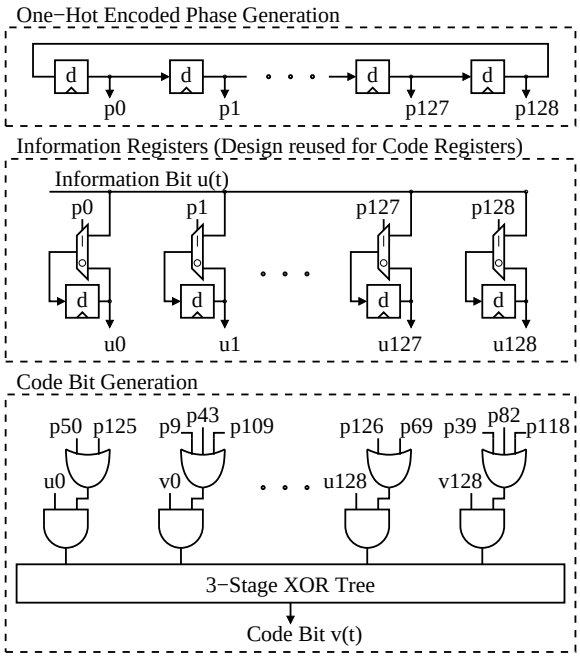

Figure 3.3: Architecture of the 1.1-GHz encoder [2]. As previously shown in the background section.

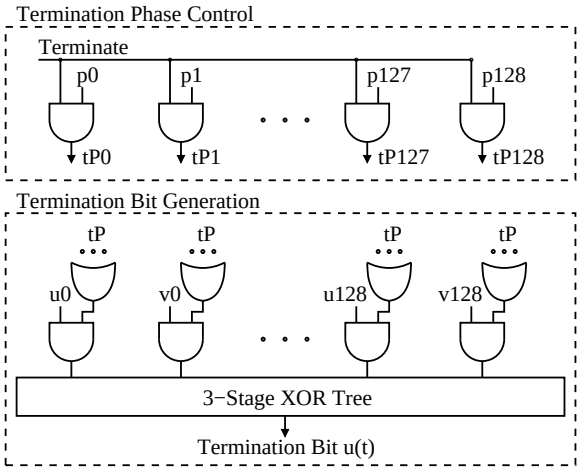

Figure 3.4: Termination mechanism for the encoder [2]. As previously shown in the background section.

error rate (BER) performance for those bits [26]. Termination is accomplished by returning the encoder to the all-zero state, where "0" value information bits result in code bits of value "0" from the encoder. Knowing that the encoder is in the all-zero state allows the decoder generate zeros at its input to complete the decoding process. Unfortunately, built-in termination increases the standard cell area of the encoder from 14,575 $\mu m^2$to 55,272 μm^2.

The LP3 LDPC-CC encoder implementation utilizes the all-phase termination scheme described in [26]. The LP3 encoder termination circuitry is shown

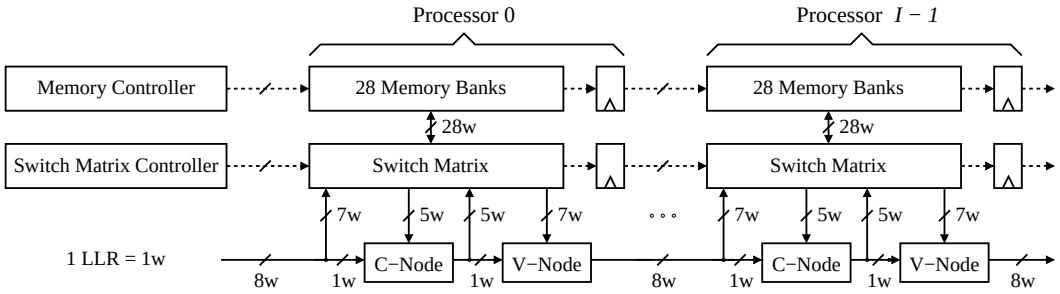

Figure 3.5: Decoder controller and the decoder processor data path [2].

in Figure 3.4. The value of the termination bit depends on the current state of the encoder. Based on the phase, approximately half of the information and code bit values are XORed to form the termination bit, which is then fed back into the encoder as the current information bit. The encoder will reach the all-zero state after approximately M cycles, where M is the encoder memory size. At this point, the termination bit will remain at '0'. The termination circuitry is inactive most of the time if the data segments are much longer than the termination sequence, so its contribution to the overall encoder power consumption is primarily leakage power.

3.4.2 Decoder

The LP3 decoder concatenates 3 identical decoder processors together. Processor concatenation helps reduce routing congestion. To save area, the LP3 decoder processor controller has been implemented separately and is shared among all the decoder processors. Since each processor performs the same set of operations as defined by the code, it is possible to share one set of control circuitry. Data and control signals are registered between decoder processors to create pipelined decoding logic.

3.4.2.1 Decoder Data Path

The LP3 decoder processor data path, as shown in Figure 3.5, processes 8-bit LLRs. For the (128,3,6) code, a variable-node consists of two variable-nodes each with a degree of three and a single check-node with a degree of six. A given LLR follows a predefined path through a processor. For example, an LLR flows from a memory bank, through the switch matrix, into a check-node or variable-node, back through the switch matrix or out of the processor and then returns to a memory bank. LLRs are registered at the memory output, the input and output of the switch matrix, the output of the check-node, the input and output of the switch-matrix again and in a memory bank. The memory write itself counts as the seventh register in the pipeline.

The critical path of the decoder starts at the registered output of the check-

node, passes through the switch-matrix and ends at the registered output of the switch-matrix.

Our LDPC-CC implementation used 8-bit LLRs, but simulations show that similar BER performance would be achievable with 6-bit LLRs. An 8-bit LLR was originally chosen to allow support for adjustable LLR widths in the range of 4 bits to 8 bits. However, due to time constraints, this feature did not get implemented. A 6-bit LLR width would be preferable because it would reduce decoder area while attaining similar BER performance.

3.4.2.2 Decoder Control Path

As shown in Figure 3.5, the decoder controller consists of memory controllers and a switch matrix controller. Each processor registers the control signals before passing them to the next processor. This phase delay, allows two clock cycles for the evaluation of the parity-check and variable-node operations. The switch matrix routes LLRs between the memory banks, two check-nodes, one variable-node and the processor's inputs and outputs.

3.4.2.3 Decoder-Memory Interface

LP3's decoder design sought to minimize the number of memory banks while preserving an information throughput of 1 bit per cycle. It was believed that the memory peripheral overhead was significant and that by minimizing the number of memory banks the silicon area could be reduced. Only single-port memories were used since dual-port memories have approximately 1.5 times greater area as well as higher power consumption. The minimum number of memory banks depends on the degree of the variable and check-node operations and the desired throughput. In our case, twelve LLRs are read and written from the memory banks per clock cycle.

The code itself specifies the LLRs required for each node operation. The memory needs to be able to supply the LLRs when they are required. The presence of seven internal pipeline stages in the data path increases the complexity of this problem. A graph-coloring algorithm was developed to find the minimum number of memory banks given the constraints of the code and pipelining timing. As shown in Figure 3.5, the algorithm indicated that 28 memory banks are required even though only 24 reads and writes occur in each cycle. The algorithm was also able to sort the LLRs so as to balance the number of LLRs in each memory bank.

3.4.2.4 Power Consumption Issues

A memory-based decoder design was chosen to reduce power consumption and area. The decoder was originally designed to function with custom SRAMs.

However, since the SRAM blocks were unproven in silicon, registers were instantiated instead to ensure reliability.

In hardware design, it is advantageous when the control path is completely independent of the data path. In the LDPC-CC decoder, all operations are predetermined thus the control of a memory bank is independent of the other memory banks and the rest of the system. The reset signal synchronizes the individual memory controller sub-circuits, thus allowing each sub-circuit to operate without communicating with any other sub-circuit. This reduces the amount of long-distance routing and the associated power consumption. The same principles apply to the design of the switch matrix controller.

3.5 Testing

Our test system consisted of an HP81200 tester along with an Agilent E3647A power supply. The PCB board used, was previously used for two LDPC designs. The controlling PC system was installed with Cygwin [27] and ssh to enabled remote access and scripting of the tests.

The HP81200 has limited IO frequency and is further limited by PCB design and chip IO driving cells. In the case of LP3, our IO are rated around 100 MHz. However, internal components were expected to run at up to 1 GHz. As previous discussed, the on-chip PLL would allow us to reach frequencies exceeding 1-GHz.

Initial tests focused on the verifying that the IO could transfer data onto and off of the chip successfully. We then proceeded to verify that the PLL was generating a clock signal and that it could achieved higher frequencies. Next, individual components were verified by capturing all output at low-clock frequencies. These results were compared with the gate-level simulations of the RTL. Once the individual components passed the functional verification tests, we proceeded to the high-frequency and power tests. Input data was generated on-chip by the random-number generator and then switched to the component under test. The output of the component under test was then directed to the CRC to generate a signature of the output data. The chip generated signatures were compared to the signatures generated in simulation. If the signatures did not match, this represent a failed test. Each test was run over 3.2 billion vectors. The 3.2 billion vector size was chosen to provide sufficient time to allow sufficiently accurate current measurements to be taken. During the test, current measurements were taken at 150 ms intervals. The maximum operating frequency of an individual module is determined by increasing the clock frequency until the on-chip CRC signature value fails to match the simulated CRC signature value.

To run the tester over 3.2 billion vectors, the Helium program [28] was modified to detect and enable loop segments from the generator file. The

HP81200 pattern memory is limited to 65k vectors, so the HP81200 generator files were constructed to have three segments: an initialization segment, a loop segment and a capture segment. The initialization segment configures the chip for the test and runs the test. The loop segment repeats one pattern to allow the chip to run its internal test for the desired length of time. The capture segment extracts and records the output of the chip after the internal-test has finished. A custom waveform file format is used to specify both simulation input vectors as well as tester input vectors. Below is a sample of our custom waveform file:

```
loop 5000 1 0 0 0   // reset for 5000 cycles
loop 5 0 0 0 0      // wait 5 cycles after turning off the reset
0 2c 1 1            // Set the clock to the PLL x8
loop 5000 0 0 0 0   // wait 5000 cycles for the PLL to stabilize
0 2c 1 1            // Set the clock to the PLL x8 (this op is not really needed)
0 1 13 1            // Set the databus-out to all zeros
0 1b 1 1            // Attach the decoder info-code0 to the CRC
0 1c 2000 1         // Set the CRC (1f,1e,1d,1c) to 0xBEBC2000
0 1d bebc 1         //    (internal cycles = 8 x 400,000,000 cycles)
0 1f 0 1
0 2e f 1            // Clock gate everything except the bist and encoder
0 0 0 1             // Start the test
loop 400000000 0 0 0 0  // loop 400,000,000 cycles
0 1 d 1             // Attach the CRC to the output databus
loop 20 0 0 0 0     // Wait 20 cycles
```

For those lines with a "loop" command, the value following "loop" is the number of times to repeat the following command. For non-loop statements, each line represents a four-argument command comprising the off-chip reset signal, the control register address, the input databus value and the write-enable signal (in that order). The custom waveform file is processed to create functional simulation stimulus file as well as a vector input file for the HP81200 tester. The results of the functional simulator are compared to the outputs of the tester to validate the functionality of the on-chip module under test.

Furthering the automation process, test-runs are scripted. Although the HP81200 software runs on a Windows PC, we are able to access the tester via a custom command line program [28] and the Linux-like shell, Cygwin. These programs combined together allow files to be securely copied to/from the tester system and commands remotely executed on the tester system. With scripted tests, we were able to queue frequency versus power sweeps over various voltages and on-chip modules thus allowing us to make efficient use of tester time.

The modularity of LP3 allowed us to approach testing the same way we approached the design, one module at a time. The first thing we did was by-pass the PLL and output the LFSR values on to the output databus. Once we saw that the LFSR was working, we routed the LFSR values into the CRC and configured the CRC to stop after 100 cycles. We compared the chip CRC value to the simulated CRC value. Seeing that they matched, we enabled the PLL. Now we ran the same test, directing the LFSR output to the CRC at

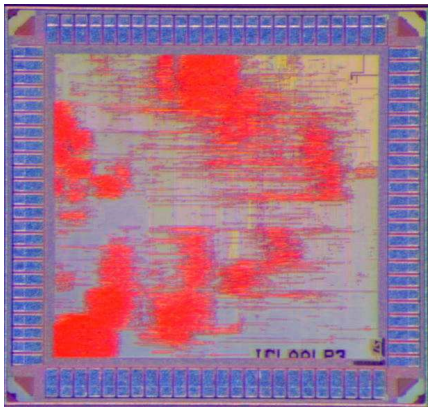

Figure 3.6: LP3 LDPC-CC packaged die photo with a 2-mm^2 core [2].

System Module	Area (μm^2)	Percentage
System Configuration	59,185	3.0
BIST	52,233	2.6
PLL	26,500	1.3
AWGN Generator	292,362	14.6
Encoder	62,304	3.1
Decoder	1,507,416	75.4

Table 3.1: System Module Silicon Area.

higher speeds. Having established that our data generation and results collection work a high-frequency, we started verifying the other components by disabling the PLL and capturing the output data to be compared with simulated data. Once all modules were deemed functional at low clock frequencies, we connected them between the LFSR and the CRC and dialed up the clock frequency until they failed. Power measurements were collected at various voltages and clock frequencies. We were able to quickly work through any issues that arose during testing and completed our testing goals within one month.

3.6 Test Results

LP3 is shown in Figure 3.6. The LP3 component silicon areas are shown in Table 3.1. The encoder and decoder occupy 3.1% and 75.4% of the 2-mm^2 core chip area. If more decoder processors were included in the design the area of the testing modules would be amortized. The termination circuitry increases the area of the encoder by almost four times (not shown).

Chip Module	Measured[1] Power (mW)	Allocated[2] Power (mW)	Net[3] Power (mW)
BIST and Leakage	33.2	–	33.2
Encoder	45.3	33.2	12.1
Noise Generator	163.1	33.2	129.9
Decoder	543.9	175.2	368.7

Table 3.2: LP3i CMOS 90 nm power measurement results with a 1 V supply. [1]The measured power consumption of one or more active modules. [2]Power already allocated for one or more other active modules. [3]The calculated power consumption of a system module. Note that the IO cells are included in the power measurements but remain inactive at that time, except for the clock input pad.

3.6.1 Test Setup

Our test system consists of an HP81200 digital integrated circuit tester and a programmable Agilent E3647A power supply. While the tester, design-under-test (DUT) board and IO drivers all limited the IO performance to 100-MHz, our LDPC-CC modules were designed to run at significantly higher speeds. For example, the encoder module can operate independently at 1.1 GHz and the maximum core frequency of the entire system is 600 MHz. As discussed in Section 3.3.3, the on-chip PLL is available to produce clock frequencies that exceed the capabilities of the IO drivers.

3.6.2 General Test Chip Measurements

The power consumption of individual test chip modules is achieved by measuring the current drawn by the test chip when specific modules are active. These current measurements are made by running the test chip for 3.2 billion clock cycles at various operating frequencies. Billions of vectors are necessary to provide sufficient time to make current measurements at 150-ms intervals. Unless otherwise indicated, the power measurements were made for an energy-per-bit divided by the spectral noise density (E_b/N_0) of 2 dB and a core clock frequency of 600 MHz and is operating with a supply voltage of 1 V.

Individual module power consumption can be measured independently. Measuring the test circuitry power and then the other modules, makes it possible to subtract the test circuitry power from the measured module-under-test power to obtain the individual module power. Unless otherwise specified the operating voltage is 1 V. The measured power consumption of the BIST circuitry, including 9.3 mW of chip leakage, is 33.2 mW. The encoder power is equal to the measured power of encoder and BIST, minus the previously measured power of BIST. Table 3.2 lists the measured and calculated power for

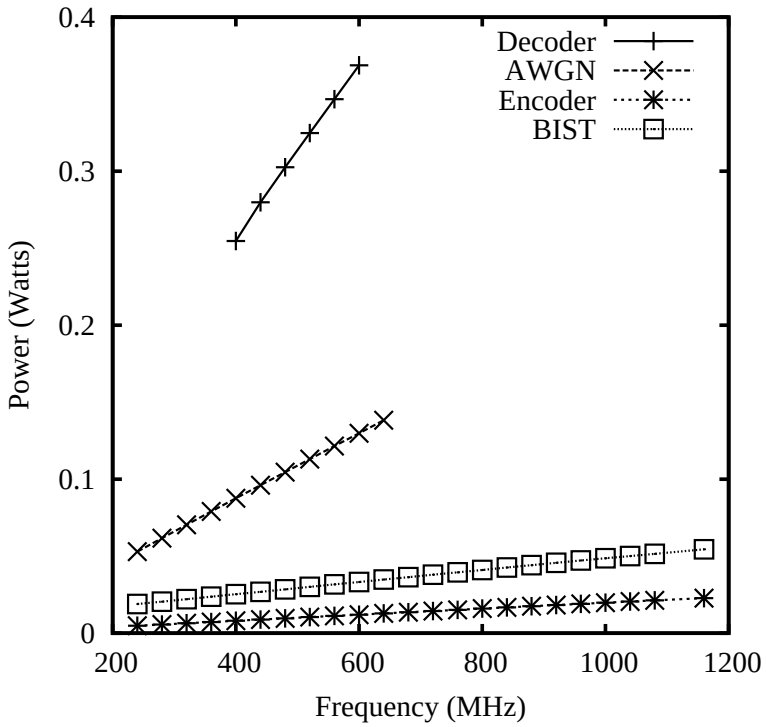

Figure 3.7: Module power consumption versus frequency with a 1-V supply at an E_b/N_0 of 2.0 dB [2].

the BIST, the encoder, the AWGN generator and the decoder, operating at 600 MHz. Assuming that the phase control circuitry's power consumption is negligible, each decoder processor consumes at most 123 mW. When operating at 1.1 GHz, the encoder module consumes 22 mW.

The module re-configurability of the LP3 design allows both the power and the speed of individual modules to be measured independently. The power consumption versus the operating frequency for individual modules is shown in Figure 3.7. The AWGN generator and the decoder, operate over a relatively narrow range of frequencies (from 400 MHz to 600 MHz), whereas, the encoder and BIST can operate at a wide range of frequencies, from 250 MHz to as high as 1.1 GHz.

3.6.3 Encoder Measurements

Reducing the operating voltage results in a significant reduction in the encoder power consumption. Figure 3.8 plots the measured encoder power consumption versus the clock frequency for various core voltages. For a core voltage of 1.0 V the maximum operating frequency of the encoder is 1.1 GHz. The encoder, when operated at 1.0 V and 600 MHz, consumes approximately 12 mW of power. Reducing the operating voltage from 1.0 V to 0.7 V reduces the encoder power consumption from 12 mW to 5 mW, while limiting the

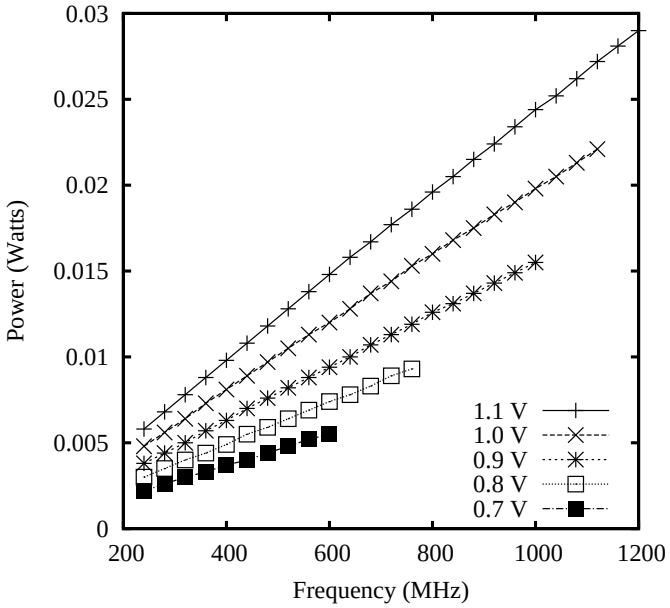

Figure 3.8: Encoder module power consumption versus frequency with a 1-V supply at an E_b/N_0 of 2.0 dB [2].

maximum operation frequency to 600 MHz.

The shmoo plot in Figure 3.9 shows the ranges of voltages and frequencies for which the encoder correctly operates (with only leakage from the termination circuitry) with a 1-V supply at an E_b/N_0 of 2.0 dB.

Comparing LP3's encoder to a previous LDPC-CC encoder design [3] reveals that we have significantly reduced the energy-per-encoded-bit. Figure 3.10 shows the energy-per-encoded-bit and the maximum operating frequency. These results are based on synthesis reports. The encoders were synthesized without termination circuitry, and the LP3 encoder design was synthesized with both one and three stages of pipelining. Compared to the reference encoder in [3] the LP3 encoder reduces the energy-per-encoded-bit by 40%.

3.6.4 Decoder Measurements

Figure 3.11 shows the measured energy-per-decoded-information-bit versus frequency for various values of E_b/N_0. E_b/N_0 changes the distribution of probabilities of receiving a '1' or a '0'. An increase in E_b/N_0 results in the channel samples being more tightly grouped around '1' and '0'. Conversely, decreasing E_b/N_0 results in the channel samples having a larger standard deviation out around '1' and '0'. The decoder has a lower switching activity factor when E_b/N_0 is high as fewer channel samples are in error. Conversely, the decoder works harder to correct errors when E_b/N_0 is low because the distribution of channel samples widens and more of the channel sample magnitudes deviate further from the nominal values of '1' and '0'.

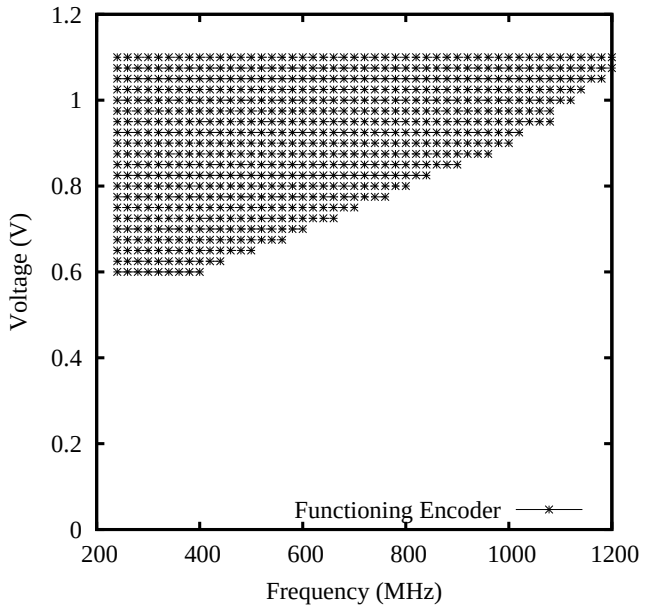

Figure 3.9: Encoder module shmoo plot [2].

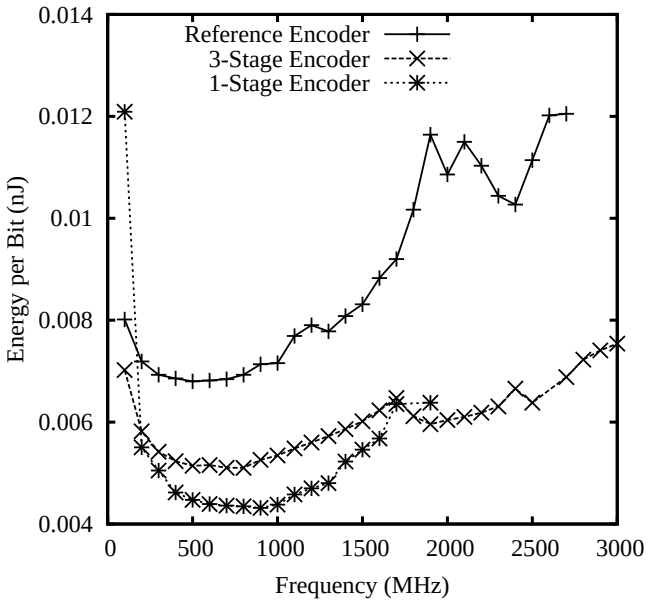

Figure 3.10: Encoder energy per bit from synthesis data [2] compared with [3].

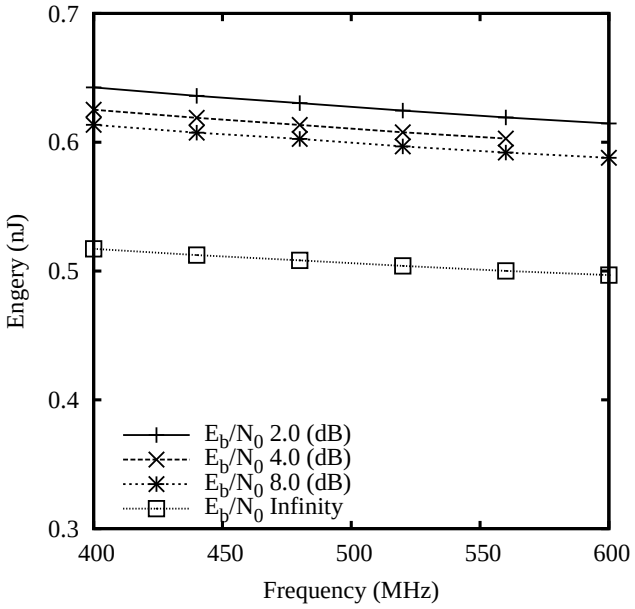

Figure 3.11: Measured energy per decoded bit using 3 processors [2].

In Figure 3.11, the lowest energy-per-decoded-bit occurs when E_b/N_0 is infinite. In this case, the channel samples are either '1' or '0' with no exceptions nor errors. The decoder does not alter the original channel samples in any way. The LLRs are simply passed through the decoder. Power is consumed by the clock-tree network, the control signals and the sign-bits flowing through the decoder data path. The decoder energy, when E_b/N_0 is decreased from infinite to 2 dB, increases by 25 percent. This can be viewed as the energy required to correct errors within the channel samples.

Dynamic power is typically linearly related to frequency and the energy-per-decoded-bit is an indirect measure of the amount of computation done to decode an information bit. However, in Figure 3.11, it is interesting to observe that the energy-per-decoded-bit decreases slightly with frequency. Considering that the power associated with leakage should not change with operating frequency, we suspect that as the operating frequency is increased, the voltage across some standard cell supply rails droops slightly due to IR drops. This drop in voltage results in a reduction in the energy-per-decoded-bit. Alternatively, the reduction in the energy-per-decoded-bit could be attributed to the reduced voltage swing of internal nets. At high frequencies, the internal signals may not have time to swing rail to rail, thus consuming less power. We do not believe this is the case, as the drop in the energy would be more pronounced at higher clock frequencies, just before the operational failure of the circuit. In our case, we see a steady, constant, decline in the energy with increasing clock frequency. While Figure 3.11 shows a decrease in energy with an increase in E_b/N_0, this would not typically be the case.

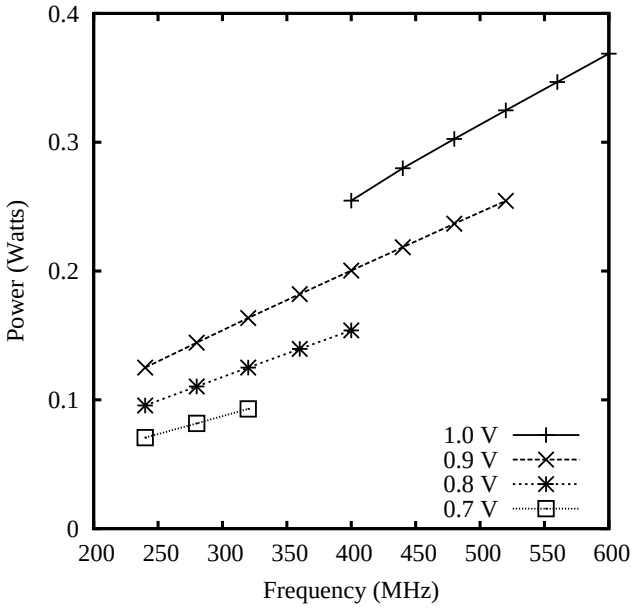

Figure 3.12: Decoder module power consumption versus frequency [2].

The energy for three processors operating at 600 MHz with an E_b/N_0 of 2 dB is 0.61 nJ per decoded information bit. The energy-per-decoded-bit for 30 processors would actually be less than ten times this value (6.1 nJ) due to an effective decrease in the average E_b/N_0 as LLRs pass through the 30-processor decoder.

As the operating voltage is reduced, the power consumption is reduced. Figure 3.12 shows the decoder power versus frequency for various core voltages. The minimum operating frequency of the PLL at a frequency multiple of 8 is 240 MHz. As the power supply is decreased the maximum decoder operating frequency decreases along with the power consumption. At a supply voltage of 1 V, it is interesting to note that the decoder has a minimum operating frequency of 400 MHz. For a 1 V supply, we hypothesize that hold time violations limit the decoder operation at frequencies below 400 MHz. For frequencies above 400 MHz, we believe that the internal core voltage drops slightly below 1 V due to IR drops, which causes the hold time violations to disappear. This hypothesis is supported by the correct operation of the decoder at clock frequencies below 400 MHz with a supply voltage of 0.9 V or less.

Although the LP3 decoder implementation consisted of only three processors, the LP3 LDPC-CC decoder results can be scaled for comparison purposes. Given 50 mm^2 of area, 30 decoder processors could be implemented. Figure 3.13 compares the BER versus E_b/N_0 for an LP3 decoder 10 and 30 processors using 8-bit LLRs, a conventional 10-iteration LDPC-BC decoder [7] and a 10-processor LDPC-CC decoder [3]. The 8-bit (128,3,6) LDPC-CC decoders

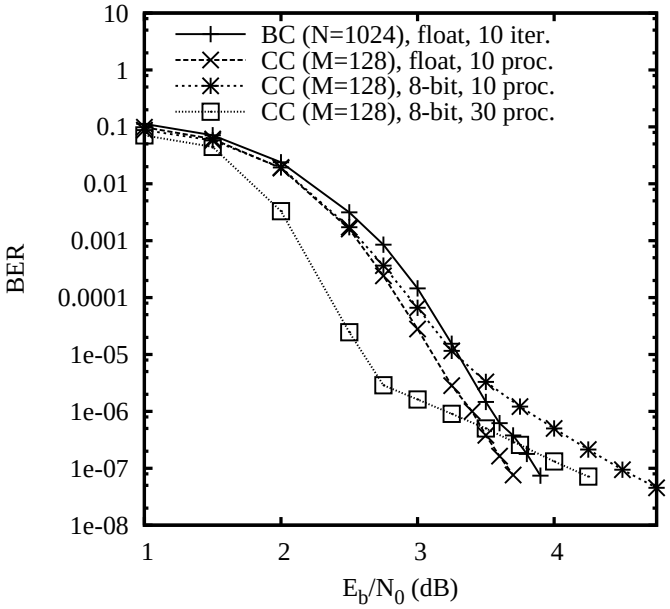

Figure 3.13: Decoder bit-error rate versus E_b/N_0 [2].

start to show error floors around a BER of 1e-06.

3.7 Conclusion

We have presented a rate-1/2 (128,3,6) LDPC-CC encoder and decoder architecture that targets high-throughput operation. LP3, implemented in a 90-nm CMOS process, features an architecture that enables power and performance measurements to be made at operating speeds. On-chip test circuitry permits accurate power measurements to be made at selectable SNR settings.

LP3, a high-throughput encoder and decoder, improves upon previous work [3]. The LP3 encoder uses techniques, such as one-hot encoding logic, to reduce the power consumption. Operating at 1.1-Gb/s, the encoder is compact and consumes 22 mW of power at 1 V while in normal operation and includes built-in termination. The LP3 encoder reduces the energy-per-encoded-bit by 40% compared to a previous convolutional encoder design [3].

The LP3 decoder adds novelty with respect to previous LDPC-CC decoders by removing the control circuitry from the decoder processor by phase-aligning the decoder processors, as well as introducing a new memory organization directed by graph coloring. A single decoder controller can be shared among an arbitrary number of processors. The decoder design uses a memory-based architecture with a minimum number of memory banks to deliver an information throughput of 1 bit per clock cycle. The decoder has energy-per-decoded-information-bit of 0.61 nJ per 3 decoder processors at an E_b/N_0 of 2 dB and

produces a decoded information throughput of 600 Mb/s. When operating at 600 MHz with an E_b/N_0 of 2 dB, and a 1 V supply voltage, the three-processor decoder consumes 369 mW of power. A 30-processor LP3 decoder would have an energy-per-decoded-information-bit of less than 6.1 nJ.

Chapter 4

Parallel-Node LDPC-CC Encoder Architecture Exploration

We present new encoder and decoder architectures that exploit code parallelism to increase both the throughput and decrease the energy-per-bit. These new architectures, compared to non-parallelized architectures, improve the throughput/area. The new encoder and decoder architectures are specific to the architecture-aware Parallel-Node Low-Density Parity-Check Convolutional-Codes (PN-LDPC-CCs) developed by Zhengang Chen and presented in [1]. PN-LDPC-CCs are constrained to allow parallel processing of the multiple information and parity bits. The maximum number of parallel information bits that can be processed at a time will be called the node-parallelization factor, ρ [1].

We present results for 26 1/2 rate (3,6) PN-LDPC-CCs. These particular PN-LDPC-CCs are a result of initial work by Zhengang Chen and later, as a result of the evolution of the encoder and decoder designs and resulting hardware metrics. As the BER performance and hardware metrics showed the benefits, and lack of consequences, of increased parallelism, we obtained codes with an increase in code parallelism to the extent which the code design process was capable of achieving. From the comparison against the published results in the literature, it became apparent that smaller codes would have sufficient BER performance for the comparisons. On account of this need, Zhengang Chen designed codes with shorter lengths and increased parallelism. In addition, to allow us to achieve a comprehensive set of results, Zhengang Chen designed PN-LDPC-CCs with varying node-parallelism and fixed code lengths. As a result, we ended up with 26 PN-LDPC-CCs. The results presented herein, are applicable to higher-rate codes as well, for more discussion on the higher-rate PN-LDPC-CCs see [1].

LDPC-CC decoders are significantly more complex than their corresponding encoders [2]. However, for those applications where one terminal will transmit far more information than they will receive, the performance of the

encoder becomes critical. In Section 4.1, PN-LDPC-CC encoder architectures are presented that set new low-power and throughput benchmarks for all known LDPC-CC and BC encoders.

Compared to previous LDPC-CC implementations, the new PN-LDPC-CC decoder architectures (described in Chapter 5) dramatically reduce the energy-per-decoded-bit while at the same time increasing the data throughput.

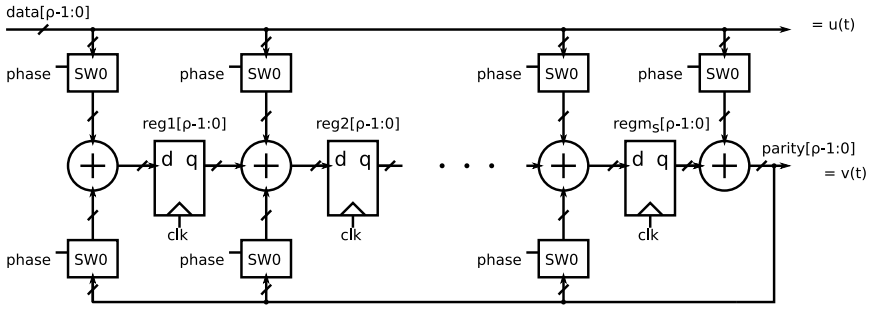

Figure 4.1: Partial syndrome encoder for the architecture-aware codes with ρ=8 [1]. Note that the "SW0" switches have an implementation complexity of $O(\rho^2)$ and that all registers are updated every clock cycle.

4.1 Parallel-Node LDPC-CC Encoders

First we will describe the initial work done in the area of parallel-node LDPC-CC (PN-LDPC-CC) encoder architectures and then proceed to describe the base version of the novel encoder architecture. Then we present a series of improvements to the novel architecture. Each improvement is qualified in terms of energy-per-encoded-bit versus throughput and area versus throughput. The results are obtained from synthesis runs. Area, speed and power estimates are obtained from synthesis to a standard cell gate-level netlist without final layout. In the case of the encoder in Chapter 3, the silicon area was 13 percent larger than the standard cell area. Note that synthesis results can vary depending on a number of factors some of which include: effort level, synthesis algorithms (i.e. different synthesis tool versions) and HDL coding. For more details on the methods used to generate the results and the variability in synthesis results, please refer to Appendix A.

Initial work on the PN-LDPC-CC encoder was presented in [1]. Figure 4.1 depicts the initial architecture. This encoder is a partial syndrome encoder [18], with ρ=8. For a given 1/2-rate PN-LDPC-CC, 8 bits of user data (v) and 8 bits of parity data (u) are output per phase. Note that the input databus data[ρ-1:0] is connected to each register bank regX[ρ-1:0] via a rotation switch matrix (SW0) and a summation circuit. Depending on the phase signal, the SW0 can produce either a circulant of the input vector or a zeroed output. The circuitry required to implement SW0 grows greater than linearly with ρ because SW0 has ρ inputs and outputs and ρ^2 paths between those inputs and outputs.

To facilitate efficient hardware implementations, it is advantageous to link the phase with memory addressing. This allows the same memory addresses to be used for the same phase in every period. For this to hold true: $m_s <= T_s$. In our set of PN-LDPC-CCs this is always the case.

All encoders discussed in this chapter process one phase per clock cycle. While it is possible, in both the encoder and decoder, to use multiple clock

cycles per phase, the total switched gate capacitance increases with this approach. Thus the total dynamic power would normally increase with a multi-cycle approach. However, in the case of poorly implemented high-fanout nets, that have excessive internal node slews, a multi-cycle approach may actually reduce the dynamic power. Dynamic-power and switched gate-capacitance are highly correlated and the power associated with short slew times, is usually small. Thus in a multi-cycle approach, for a non-clock-gated implementation, such as the encoder discussed in Section 4.2, the repeated switching of the clock network in addition to the extra storage required to store intermediate signals, would be undesirable. As a general rule, when high fanout nets are not present, introducing extra cycles can indeed improve throughput, but at the cost of higher power consumption. In rare cases with nets with long slews, then introducing pipelining can potentially save power. However, in order for power savings to occur the power associated with the extra load on the clock and data path in the form of latches or flip-flops must be less than the power consumed via prolonged crowbar current due to longer slews. Typically, buffering shortens the slew, but at the cost of greater latency. Should the latency increase unacceptably, then pipelining may be required, regardless of the increase in power consumption. Given our emphasis on low-power, multi-cycle datapaths will be be avoided in favor of doing as much as possible in a single cycle.

The throughput of an encoder is determined by both the code and implementation architecture. When one phase is processed per cycle, the throughput is equal to the clock frequency multiplied by the ρ:

$$N_{encThroughput} = f_{clk} \cdot \rho \tag{4.1}$$

where $N_{encThroughput}$ is the encoder throughput, f_{clk} is the clock frequency and ρ is the node-parallelization factor.

In the following sections, we present a series of improvements to the initial encoder architecture shown in Figure 2.7. Section 4.2 describes a design that eliminates the hardware dependence on ρ. Section 4.3 describes a design that swaps XOR gates for OR gates to reduce the power consumption. Section 4.4 describes a design that uses clock-gating to reduce power consumption. Section 4.5 describes a design that increases the databus size to further reduce the switching activity and thus the power consumption. Section 4.6 shows how de-multiplexing the input-databus reduces the power consumption. Section 4.7 shows how to merge multiple PN-LDPC-CC encoders together to reduce the energy per encoded bit as well as the throughput/area. Section 4.8 compares the best of the new PN-LDPC-CC encoders to the state-of-the-art for LDPC encoders. Section 4.9 summarizes the main conclusions about the new PN-LDPC-CC encoders. Our encoder version numbering starts at v2. Encoder v1 was a non-functional design and will not be be discussed.

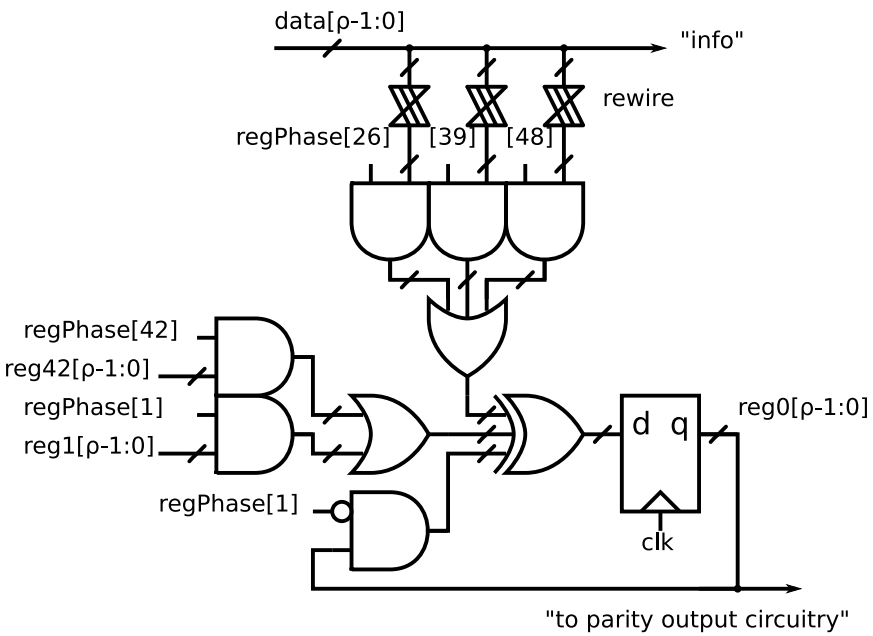

Figure 4.2: Encv2 encoder node. The register is updated only 5 times per T_s'. This architecture is free from the $O(\rho^2)$ hardware complexity.

4.2 Encoder v2 - Circular Buffer (Encv2)

In this section we present the novel encv2 architecture that removes the hardware dependence on the value of ρ. In other words, increasing ρ has no impact on the circuit area. In addition, we show that using the encv2 architecture decreases the energy-per-encoded-bit when the ρ is increased. Recall that in the previous section, we established that the throughput is linearly related to ρ.

The development of the encv2 architecture is best described using a few high-level concepts. The encv2 architecture is created by constraining the register banks so they do not need to shift; instead they become, in the next phase, the next register bank in the shift-chain. Also, the registers are constrained to accept input data unique to each phase. In this way the shift chain in the reference design has been transformed into a circular buffer. This requires the control logic for each encoder node to be updated to compensate for such changes. As a result, the phase-dependent switch matrices (SW0) from Figure 2.7 are eliminated and are replaced with fixed re-wiring and phase-gated inputs. Figure 4.2 shows the new encoder node architecture. The datapath for encv2 is formed by replicating the encoder node for each phase in the group period T_s'. The hardware complexity has been simplified.

The flow of data through encv2 (Figure 4.2) is altered compared to the reference design (Figure 2.7). In the reference encoder architecture, data flows through the shift-chain of registers to the end register where it becomes the parity-output. In encv2, register data is updated according to the PC-LDPC-

CC, but stays with a flip-flop until it is replaced in the next period. In encv2, the parity-output is actively chosen from one set of the T_s' encoder node registers. Similar to the reference design, the input-data vector is broadcast to all register banks. This allows the register to update itself with the appropriate phase selected input data. Listing 4.1 contains pseudo code for the encoder operation.

0.05

```
pcm = parity_check_matrix
current_phase = 0
while( not end of frame )
    data_input = user_data()
    foreach encoder_node in parallel with all_encoder_nodes
        if current_phase is used in encoder_node
            if current_phase is used for data_input to encoder_node
                xorInA <= rewire( data_input, pcm[current_phase] )
            else
                xorInA <= 0

            if current_phase is used for reg_input to encoder_node
                xorInB <= reg[pcm[current_phase]]
            else
                xorInB <= 0

            if current_phase is the update_phase for this encoder_node
                reg <= regs( pcm(e, current_phase) )
            else
                reg <= xor( xorInA, xorInB, reg )
        else
            reg <= reg

        if current_phase == encoder_node
            parity_output <= reg

    end foreach encoder_node
    current_phase = (current_phase + 1) MOD T's
end while
```

Listing 4.1: Encoder pseudo code.

The phase is one-hot encoded to reduce circuit complexity. One-hot encoding, also referred to as "a one in a field of zeros", also reduces the switching activity of the control circuitry. Figure 4.3 shows how the phase is one-hot encoded. When the encoder reset signal is asserted (not shown), all registers are zeroed, except the one register that is being set. Using this signal the one-hot encoded phase is initialized.

A consequence of the parity data not shifting through the registers is that all of the registers need to be connected to the parity output. This differs from the reference design, where only one register bank is connected to the parity-output. As shown in Figure 4.4, depending on the phase, the output parity bits are multiplexed out from one of the registers. Here the one-hot encoded phase is used to conditionally connect T_s' register banks, of width ρ, to the parity output vector $v(t)$. The hardware complexity of the OR tree is given by Equation (4.2).

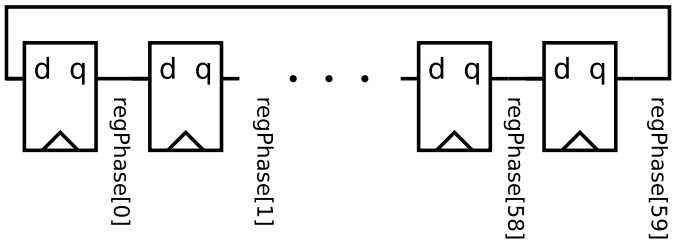

Figure 4.3: One-hot encoding of the phase. In this case for a T_s=480, ρ=8 code. Upon reset, one register is set and all other register are unset.

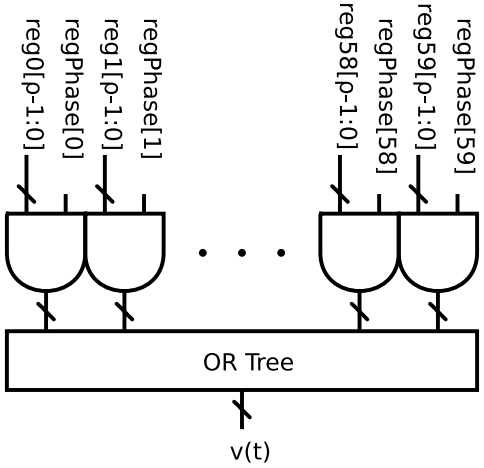

Figure 4.4: Encv2 parity output circuitry. Only one "regPhase" signal is asserted at any given time.

$$C_{ORtreeComplexity} = O(T_s' \cdot log(T_s')) \tag{4.2}$$

The number of OR trees is equal to ρ. From the perspective of the info-input and parity-output databusses, the shift-register chain has been transformed into a circular buffer.

Given insight as to how the PN-LDPC-CCs work, area and power can be saved by replacing XOR gates with OR gates. Knowing that multiple info or multiple parity updates cannot occur in the same phase, encv2 uses OR gates to combine info and code updates before XORing the current register's contents with that of the info and parity updates. This is significant in that XOR gates consume more area, more power and are slower than OR gates. In this case, we know that only one info update and only one parity update are performed per phase. A further optimization would be to detect when both an info and parity update occur in a single phase and then use an XOR gate; otherwise, use an OR gate (this is further explored in Section 4.3).

Synthesis results for cell area, power and timing were obtained using Synopsys Design Compiler version 2007.03. To account for the clock tree network, the timing constraints include a skew of 50 ps. Unless otherwise specified,

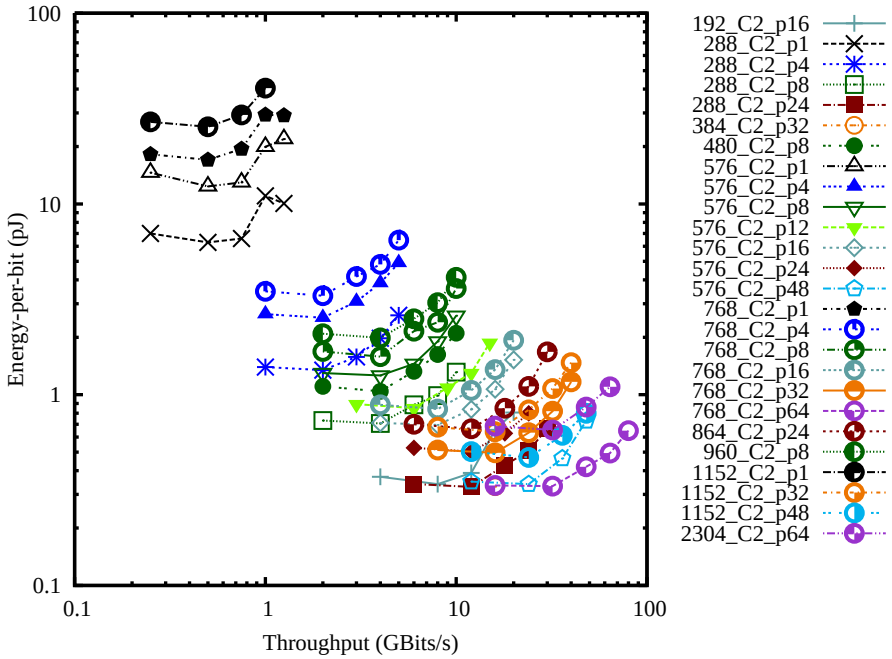

Figure 4.5: Encv2 energy-per-bit versus throughput. Throughput and energy-per-bit are not mutually exclusive. Those codes with a higher ρ also have lower energy-per-encoded-bit values. Those codes with a lower ρ have higher energy-per-encoded-bit values. Consecutive points on the same line are a result of synthesis runs targeting higher clock frequencies. Parts of this graph are examined in detail in Figures 4.7 and 4.8.

the results are based on a ST Microelectonic's CMOS 90-nm standard cell library (for more details see Appendix A). Timing was analyzed using both the best and worst case timing libraries, with the included wireload models. Power estimates were made via Design Compiler, using the worst-case timing libraries, after back-annotating internal node switching activities derived from a 10,000-clock-cycle gate-level functional simulation. Each encoder circuit was independently synthesized for a target clock frequency of 250, 500, 750, 1000 and 1250 MHz. Results are reported if the synthesized encoder passes Design Compiler's static-timing analysis (i.e. has zero or positive slack).

As was noted earlier, as ρ increases, the energy-per-encoded-bit decreases while the throughput increases. Figure 4.5 shows the energy-per-encoded-bit versus the throughput for our PN-LDPC-CCs. These results are based on the encv2 architecture. This graph provides a helpful visual reference showing extremes and trends. The maximum encv2 throughput is 64 Gbits/s (produced by the T_s=2304 and T_s=768 codes with ρ=64). The lowest energy-per-encoded-bit is achieved by those codes with the lowest T_s' and is on the order of 4×10^{-13} Joules. It is interesting to note that the codes with the highest ρ simultaneously achieve the maximum throughput and nearly the lowest energy-per-encoded-bit. Figure 4.6 show the energy-per-encoded-bit versus the

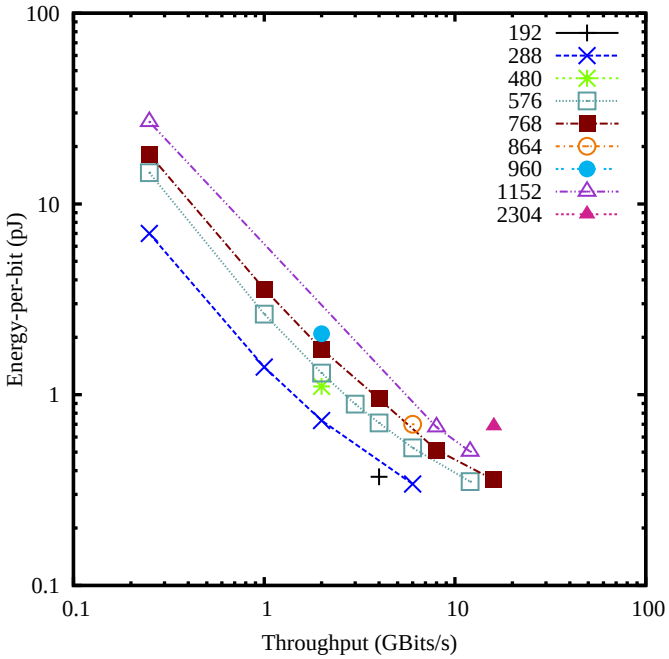

Figure 4.6: Encv2 energy-per-bit versus the information throughput for the PN-LDPC-CCs. The higher the ρ, the higher the information throughput and the lower the energy-per-encoded-bit. Code lengths (T_s) and parallel-node (ρ) values are as specified: for T_s=192 $\rho \in \{16\}$, for T_s=288 $\rho \in \{1,4,8,24\}$, for T_s=384 $\rho \in \{32\}$, for T_s=480 $\rho \in \{8\}$, for T_s=576 $\rho \in \{1,4,8,12,16,24,48\}$, for T_s=768 $\rho \in \{1,4,8,16,32,64\}$, for T_s=864 $\rho \in \{24\}$, for T_s=960 $\rho \in \{8\}$, for T_s=1152 $\rho \in \{32,48\}$, and for T_s=2304 $\rho \in \{64\}$. A clock frequency of 250 MHz is used.

throughput versus ρ for a 250-MHz clock. Each curve in the figure corresponds to a fixed code length. Points along a curve, moving from left-to-right, represent an alternative code with the same T_s but an increased ρ. As ρ increases, energy-per-encoded-bit decreases and throughput increases.

As ρ goes up, the energy-per-bit goes down. Comparing the T_s=768 codes (Figure 4.7) and T_s=576 codes (Figure 4.8) for various values of ρ, we see a nearly inverse linear relationship between energy-per-encoded-bit and ρ. Both sets of codes, T_s=768 and T_s=576, clearly display the trend that as ρ goes up the energy-per-bit goes down.

However, there is no systematic relationship between ρ and the maximum clock frequency. Figure 4.9 shows, for the T_s=768 codes, that the maximum attainable clock frequency is unaffected by increasing ρ. All T_s=768 codes are able to meet timing at a clock frequency of 1250 MHz. None of the T_s=768 codes are able to meet timing at a clock frequency of 1500 MHz. This lack of correlation between ρ and the maximum attainable clock frequency also holds true for the other T_s codes.

The standard cell area required to implement the encv2 architecture is in-

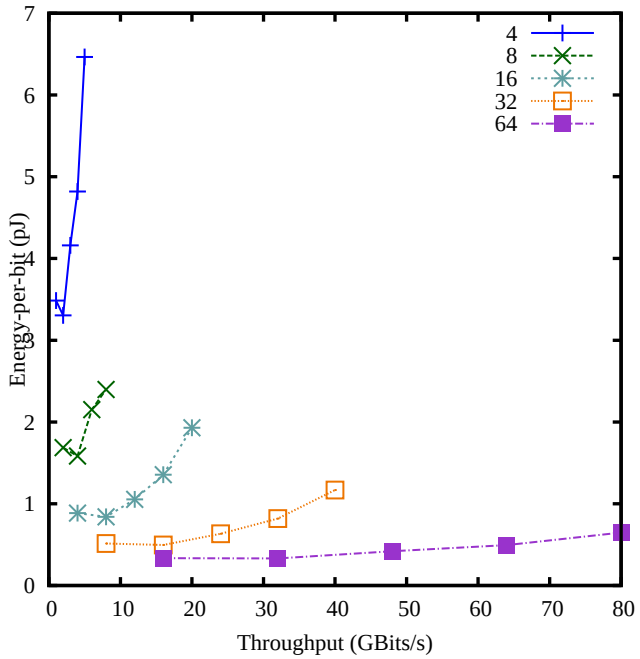

Figure 4.7: Encv2 energy-per-encoded-bit versus throughput for T_s=768 codes with a ρ of 4, 8, 16, 32 and 64. For codes with the same T_s, note how increasing ρ lowers the energy-per-bit and raises the throughput.

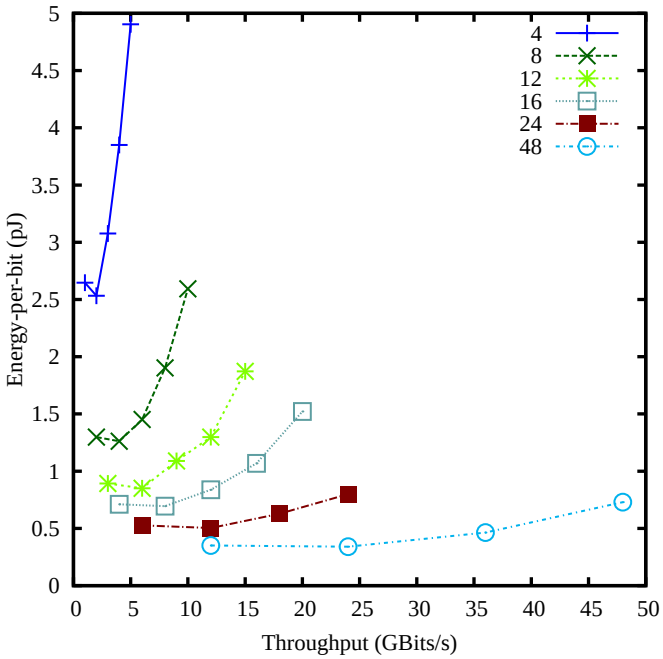

Figure 4.8: Encv2 energy-per-encoded-bit versus throughput for T_s=576 codes with a ρ of 4, 8, 12, 16 and 24. Again, for codes with the same T_s, note how increasing ρ lowers the energy-per-bit and raises the throughput.

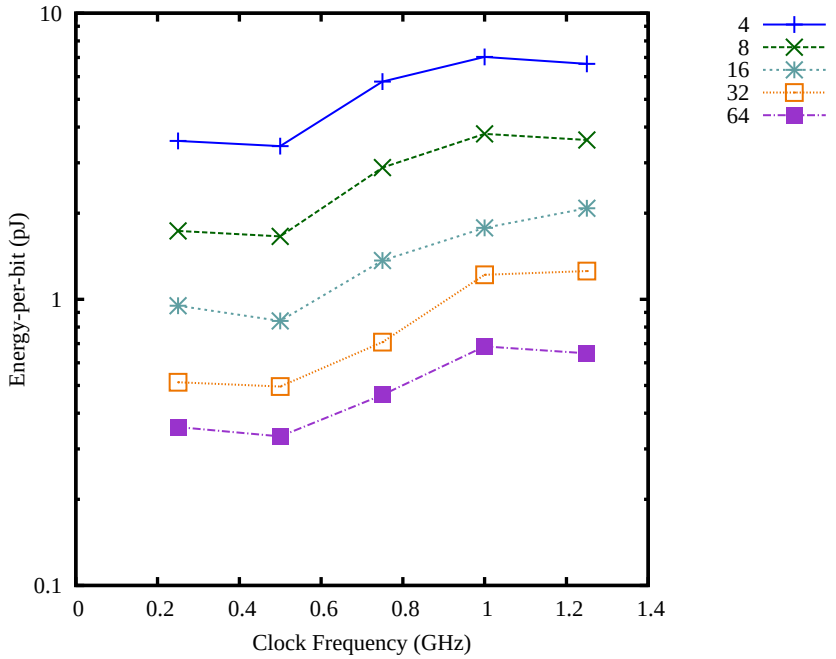

Figure 4.9: Encv2 energy-per-encoded-bit versus clock frequency for T_s=768 codes with a ρ of 4, 8, 16, 32 and 64. The maximum attainable clock frequency is the same from all five values of ρ.

dependent of ρ. Figure 4.10 shows, for the T_s=768 codes, the impact of ρ on the standard cell area over a range of clock frequencies. While increasing the clock frequency does indeed increase the standard cell area, increasing ρ has little impact on standard cell area. This can be explained intuitively by noting that the number of registers in the encoder remains constant regardless of changes in ρ. As ρ increases, the number of encoder nodes decreases proportionally to the increase in ρ while the number of registers in each encoder node increases proportionally to the increase in ρ. By eliminating the rotation switch-matrix found in the reference design, changing ρ has almost no net impact on hardware complexity.

To confirm encv2's area-independence from ρ, Figure 4.11 plots area versus ρ for the T_s=768 codes for clock frequencies of 250, 500, 750, 1000 and 1250 MHz. Note that the area changes very little as ρ increases.

The standard cell area is affected by the clock frequency. Figure 4.12, shows the area versus the throughput for various codes with ρ=8. As expected, the area slightly increases with throughput and nearly linearly increases with T_s. Throughput increases linearly with the clock frequency. As the clock frequency increases the synthesis tool will use more gates or larger gates in order to meet timing, hence the slight increase in area as the throughput goes up.

The required area is primarily determined by T_s. Figure 4.13 shows the standard cell area versus throughput versus ρ. Each curve in the figure corre-

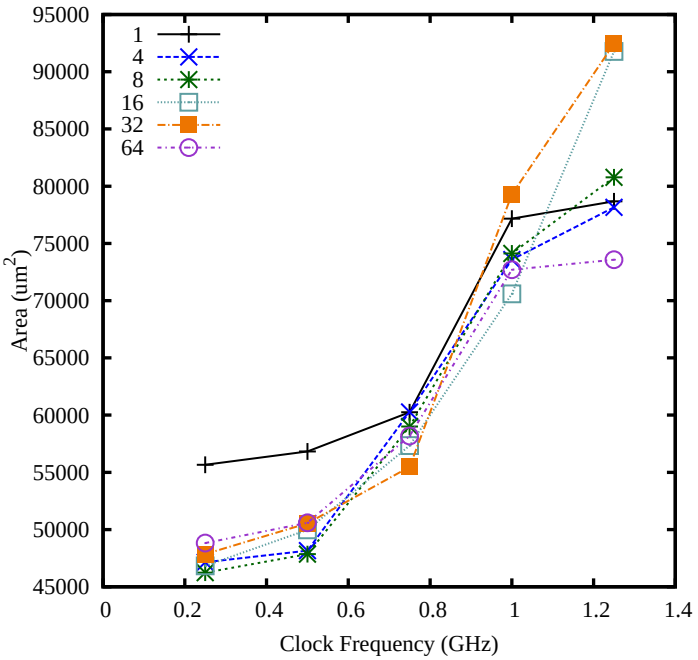

Figure 4.10: Encv2 area versus clock frequency for T_s=768 codes with a ρ of 4, 8, 16, 32 and 64. The encoder area appears to be independent of ρ, but is correlated to the clock frequency.

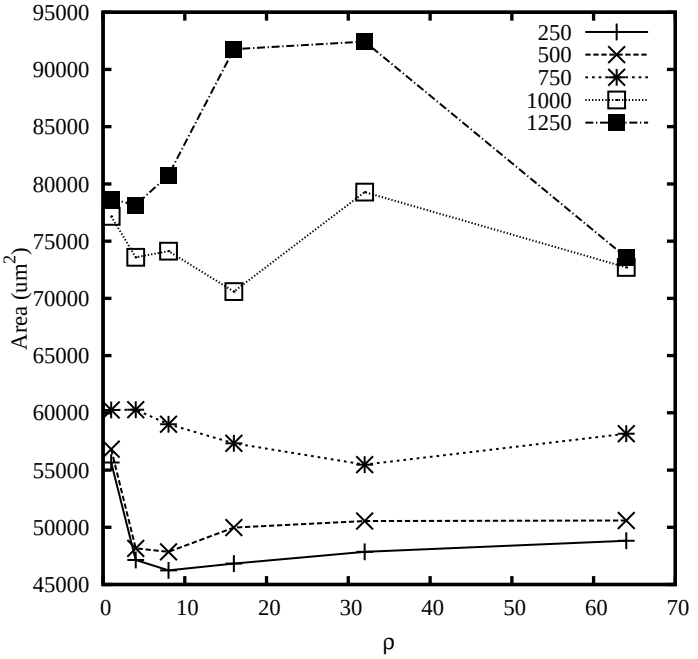

Figure 4.11: Encv2 area versus ρ for T_s=768 codes for clock frequencies of 250, 500, 750, 1000 and 1250 MHz. The encoder area is uncorrelated to ρ.

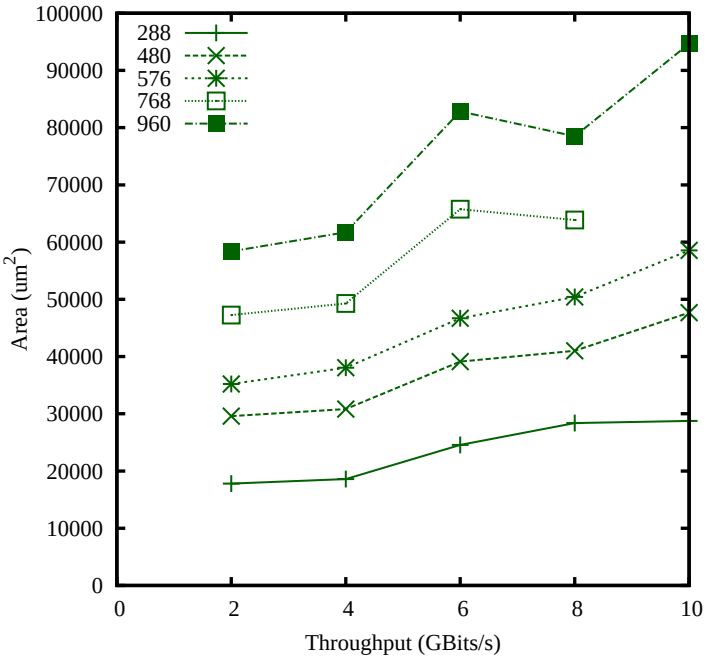

Figure 4.12: Encv2 area versus throughput for ρ=8 codes of lengths T_s equal to 288, 480, 576, 768 and 960. Note that area increases with T_s. The area also increases slightly as the throughput increases.

sponds to a fixed code length. Points along a curve, moving from left-to-right, represent an alternative code with the same T_s but an increased ρ.

As T_s increases, the energy-per-encoded-bit increases. Given ρ=8, Figure 4.14 shows the relationship between T_s and the energy-per-encoded-bit. Although the number of computations done per cycle remains the same for the ρ=8 codes, increasing T_s increases the number of registers, the size of the clock network and the fanout and fan-in of the input databus and output databus, respectively. Thus, for the encv2 architecture, the relationship between T_s and energy-per-encoded-bit is nearly linear.

The energy-per-encoded-bit appears to be a function of T_s', which is defined to be ratio of T_s to ρ. Figure 4.15 compares the energy-per-encoded-bit versus clock frequency for codes with a T_s' of 36. For the various codes that have a T_s' of 36, the energy-per-encoded-bit is roughly independent of T_s, over a range of clock frequencies. At the highest operating frequency we see a scatter in the energy-per-bit that appears to be uncorrelated to either ρ or T_s. Figure 4.9 shows the energy-per-encoded-bit for a constant code length for various values of ρ, while Figure 4.14 shows the energy-per-encoded-bit for a constant ρ for various values of T_s. In both figures, the energy-per-bit changes with T_s and ρ and confirms the observation that T_s' has the largest impact on the energy-per-bit. This belief is confirmed in Figure 4.16, which plots the energy-per-encoded-bit versus T_s' for all codes at a clock frequency of 500 MHz.

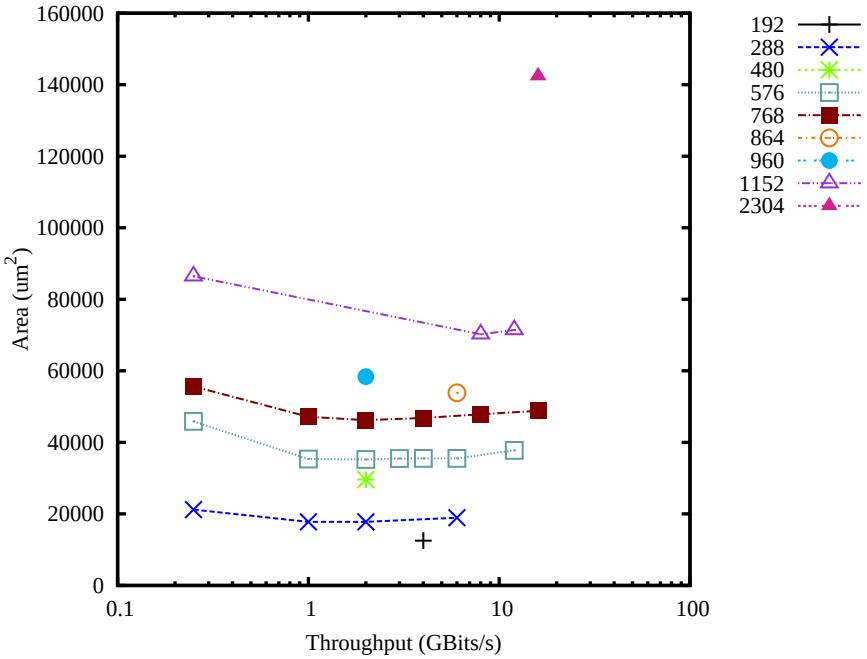

Figure 4.13: Encv2 area versus the information throughput, for the PN-LDPC-CCs. Code lengths (T_s) and parallel-node (ρ) values are as specified: for T_s=192 $\rho \in \{16\}$, for T_s=288 $\rho \in \{1,4,8,24\}$, for T_s=384 $\rho \in \{32\}$, for T_s=480 $\rho \in \{8\}$, for T_s=576 $\rho \in \{1,4,8,12,16,24,48\}$, for T_s=768 $\rho \in \{1,4,8,16,32,64\}$, for T_s=864 $\rho \in \{24\}$, for T_s=960 $\rho \in \{8\}$, for T_s=1152 $\rho \in \{32,48\}$, and for T_s=2304 $\rho \in \{64\}$. A clock frequency of 250 MHz is used.

Note that the energy-per-encoded-bit is nearly linearly dependent on T_s'. The plotted line, generated using least-squares fitting, has an intercept of 0.0834 pJ, a slope of 1.68×10^{-14}, and matches the plotted data with a coefficient of determination of 0.996. In summary, neither the code length T_s nor the ρ alone predict the energy-per-encoded-bit, but their ratio, T_s', does.

Examining the internal circuit switching activity with knowledge of the factors effecting the loading allows us to focus on design modifications with the largest potential for power savings. Table 4.1 shows the typical switching activity per clock cycle for the encv2 architecture. The clock, "clk", switches once per cycle. The switching activity of "regPhase[0]" represents the processing of a databus input vector of size ρ. Multiplying the switching activity of "regPhase[0]" by the T_s' will always produce a value approximately equal to one. Another way of looking at this is that "regPhase[0]" will have two transitions per T_s'. Register "regs[0]" will statistically toggle 2.5 times per T_s'. This is consistent with our (3, 6) half rate codes. The registers are potentially updated five times per T_s'. Half of the time, a register will be updated with the value already present in the register, resulting in no output node switching. Thus the average toggle rate is 2.5 times per T_s'. Given the random values

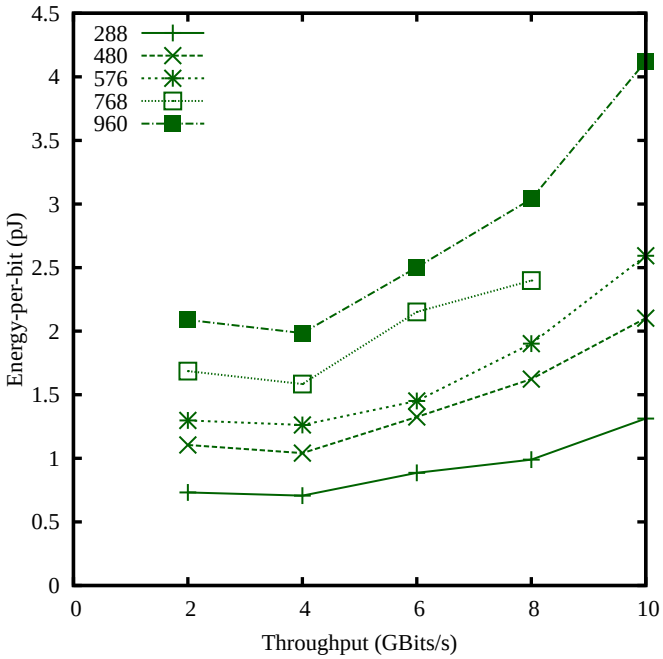

Figure 4.14: Encv2 energy-per-encoded-bit versus throughput for ρ=8 codes of various lengths T_s=288, 480, 576, 768, 960. Note that the energy-per-bit grows with T_s.

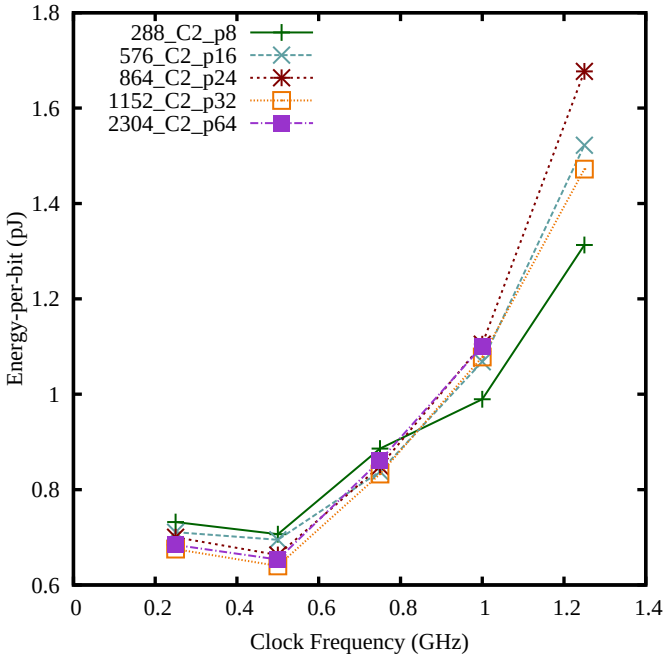

Figure 4.15: Encv2 energy-per-encoded-bit versus the clock frequency for various codes with a common T_s' of 36. The bunching of the curves supports our claim that T_s' is a predictor of energy-per-encoded-bit.

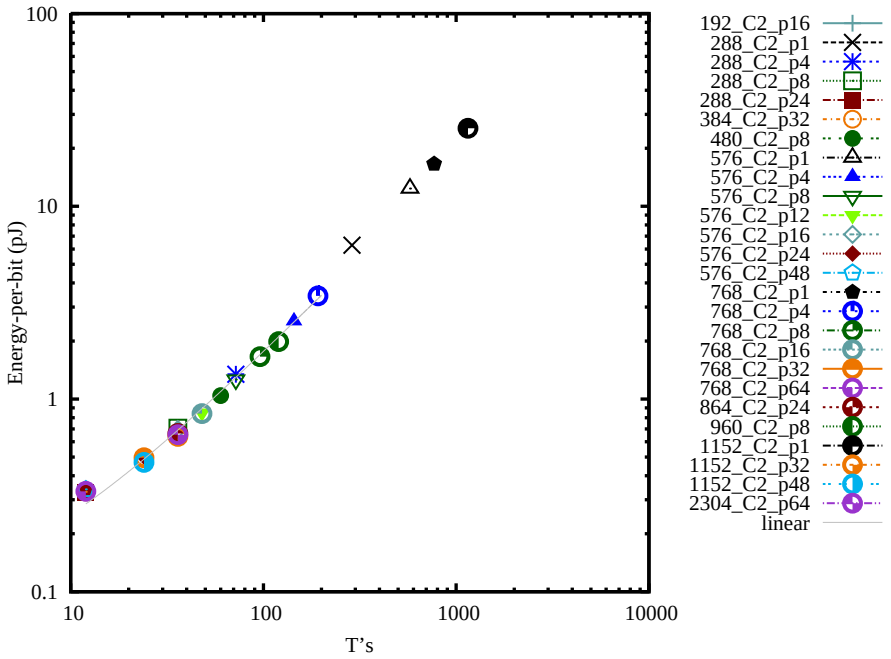

Figure 4.16: Encv2 energy-per-encoded-bit versus T_s' for various codes at a clock frequency of 500 MHz. A best-fit line, with an intercept of 0.0834 pJ and a slope of 1.68×10^{-14}, is plotted to show the correlation of the data. The coefficient of determination is 0.996. These results confirm T_s' as a predictor of energy-per-encoded-bit.

generated for the input databus, databus inputs will, on average, toggle at a rate of 0.5 times per clock cycle. Given the nature of the code, the "code[0]" output is expected to toggle at a rate of 0.5 times per clock cycle. Variations from the predicted values are statistical deviations due to the simulation over only 10,000 clock cycles. To reduce the power consumption, the important factor to consider is the capacitance of a net multiplied by its switching activity. In this case, the clock network and the input databus both have relatively large networks and high switching rates. Section 4.4 explores the possibility of clock-gating and Section 4.6 explores how power can be reduced by de-multiplexing the databus.

Considering the encoder switching activity helps us to explain the nearly linear relationship between T_s' and the energy-per-encoded-bit. In the encv2 architecture the switching power is dominated by the clock network, with relatively little combinational circuitry active in each clock cycle. As ρ increases, the active combinational circuitry activity does indeed increase roughly linearly; however, the resulting contribution of dynamic power remains small compared to that of the clock network. The number of registers in the encoder datapath and the size of the clock network is proportional to the code length (T_s). The larger the value of ρ, the fewer registers in the one-hot control circuitry, and the more bits encoded per clock-cycle. The larger the value of

code	T_s'	clk	data[0]	regPhase[0]	regs[0]	code[0]
T_s 1152 ρ 32	36	1.000	0.255	0.028	0.035	0.253
T_s 1152 ρ 48	24	1.000	0.251	0.042	0.051	0.250
T_s 2304 ρ 64	36	1.000	0.248	0.028	0.034	0.246
T_s 288 ρ 4	72	1.000	0.244	0.014	0.018	0.248
T_s 288 ρ 8	36	1.000	0.247	0.028	0.036	0.250
T_s 480 ρ 8	60	1.000	0.248	0.017	0.021	0.245
T_s 576 ρ 12	48	1.000	0.249	0.021	0.027	0.251
T_s 576 ρ 16	36	1.000	0.249	0.028	0.035	0.249
T_s 576 ρ 24	24	1.000	0.252	0.042	0.052	0.248
T_s 576 ρ 4	144	1.000	0.244	0.007	0.009	0.249
T_s 576 ρ 8	72	1.000	0.248	0.014	0.018	0.248
T_s 768 ρ 16	48	1.000	0.248	0.021	0.021	0.249
T_s 768 ρ 32	24	1.000	0.255	0.042	0.052	0.249
T_s 768 ρ 4	192	1.000	0.244	0.006	0.007	0.250
T_s 768 ρ 8	96	1.000	0.248	0.011	0.014	0.252
T_s 864 ρ 24	36	1.000	0.252	0.028	0.036	0.248
T_s 960 ρ 8	120	1.000	0.248	0.009	0.010	0.247

Table 4.1: Encv2 switching activity per clock-cycle.

ρ, the more the power of the clock network is amortized over a larger number of encoded bits. As was established earlier, the throughput is proportional to ρ. Thus, in a simplified view of encv2, the energy-per-bit can be viewed as the energy consumed by the clock network divided by the ρ multiplied by the clock frequency. If the clock network is a major source of power consumption in the encv2 architecture, then it stands to reason that replacing the clock network with the clock-gated phase signals should significantly reduce the energy-per-encoded-bit. This strategy is explored in Section 4.4.

At low clock frequencies, the leakage power dominates the total power consumption and increases the energy-per-encoded-bit. In Figures 4.9 and 4.15, which plot the energy-per-encoded-bit versus the clock frequency, we see a slight increase in the energy-per-encoded-bit at the lowest clock frequency (250 MHz). This is due to the static power consumption becoming a more significant source of power consumption. At 250 MHz and 500 MHz, the static power consumption represents approximately one-quarter and one-eighth, respectively, of the total power consumption. As the clock frequency increases, the dynamic power dominates the power consumption.

In summary, increasing the node-parallelization factor ρ in the PN-LDPC-CC encv2 architecture increases the throughput, decreases the energy-per-encoded-bit and does not impact the required silicon area. For the encv2 architecture, we have shown that T_s' is a predictor of the energy-per-encoded-bit.

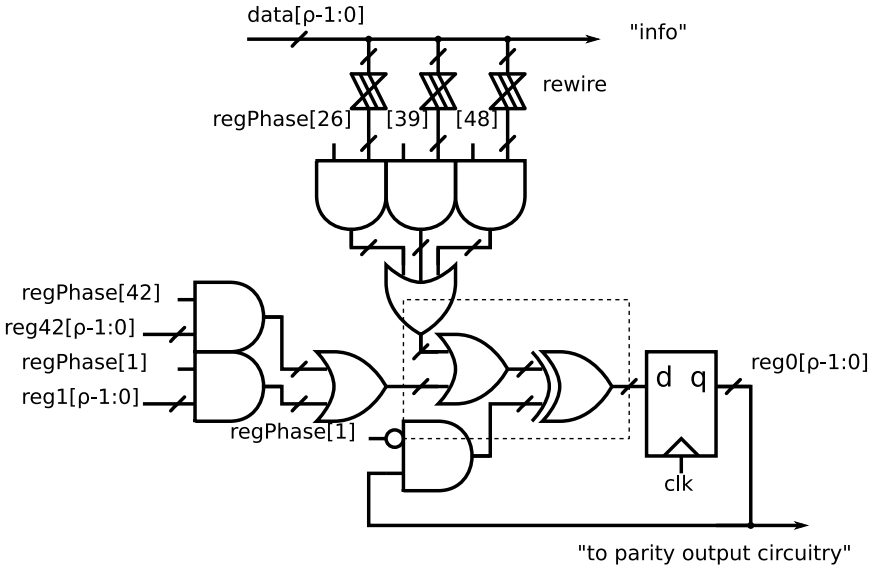

Figure 4.17: Encv3 encoder node. We can conditionally replace the 3-input XOR from Figure 4.2 with a 2-input XOR and a 2-input OR gate.

4.3 Encoder v3 - Replacing a 3-Input XOR-Gate with a 2-Input XOR-Gate and 2-Input OR-Gate (Encv3)

In this section to reduce the power consumption, we replace most of the 3-input XOR gates with one simpler 2-input XOR gate and one OR-gate. We refer to this operation as gate-swapping. In encv2, Figure 4.2, we can see that the phases controlling the data ("info") and register ("parity") updates are not the same. Thus we can OR the phase-gated "info" and "parity" updates together and input the result into a 2-input XOR. When the "info" and "parity" updates occur in the same phase, the 3-input XOR is still required. In the T_s=480, ρ=8 code, which has a T_s' of 60, the "info" and "parity" updates only occurred simultaneously, twice out of 60 phases. In the case of the T_s=768, ρ=16 code (T_s' of 48), simultaneous "info" and "parity" updates occur in 5 out of 48 phases. In the case of the T_s=768, ρ=64 code (T_s' of 12), simultaneous "info" and "parity" updates occur in 4 out of 12 phases. This pattern is representative of the other PN-LDPC-CCs.

Gate swapping results in little change in the energy-per-encoded-bit. Figure 4.18 is shown as a reference for the energy-per-bit versus throughput for the PN-LDPC-CCs. Compared to Figure 4.5, we see that there is actually little change in the minimum energy-per-encoded-bit. Figure 4.19 show the energy-per-encoded-bit versus the throughput versus ρ for a 250-MHz clock. Each curve in the figure corresponds to a fixed code length. Points along a curve, moving from left-to-right, represent an alternative code with the same T_s but an increased ρ. As ρ increases, energy-per-encoded-bit decreases and throughput increases.

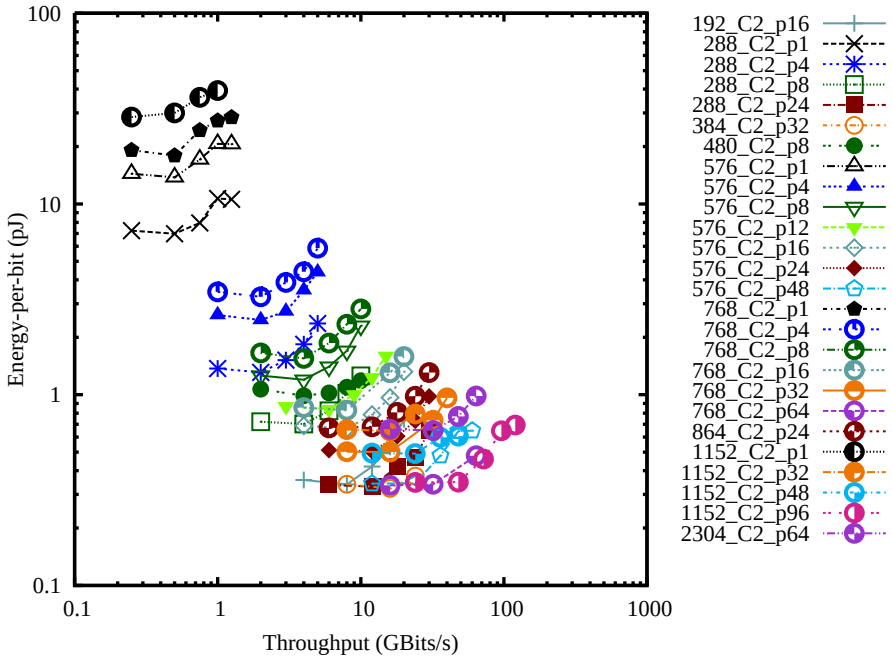

Figure 4.18: Encv3 energy-per-bit versus throughput for all the PN-LDPC-CCs. Compared to encv2, there is an overall slight reduction in the ratio of energy-per-encoded-bit to throughput.

In most cases, gate swapping slightly reduces the energy-per-encoded-bit for those codes with larger T_s'. Figure 4.20 compares encv2 and encv3 for a subset of codes in terms of energy-per-encoded-bit versus throughput. The conditional replacement of the 3-input XOR gates slightly reduces the energy-per-encoded-bit for those codes with larger T_s'. The reduction in energy-per-encoded-bit is typically greater at higher clock frequencies, with the exception of the ρ=64 codes where decv3's energy-per-decoded-bit converges with decv2's. Swapping 3-input XORs for 2-input XORs produces a small, but systematic reduction in power.

On average, gate swapping reduces the area by 4 percent. For the same set of codes, as in Figure 4.20, Figure 4.21, compares the area versus throughput of encv3 to that of encv2. For all but the highest throughputs, area is reduced for every code by using encv3 instead of encv2. At the highest clock frequencies the area savings are the smallest. Overall, gate swapping saves area.

The required area is primarily determined by T_s. Figure 4.22 shows the standard cell area versus throughput versus ρ. Each curve in the figure corresponds to a fixed code length. Points along a curve, moving from left-to-right, represent an alternative code with the same T_s but an increased ρ.

In summary, conditionally swapping the 3-input XOR with a 2-input XOR and a 2-input OR gate is advantageous in terms of the energy-per-encoded-bit and area. The gate swapping is most effective in those codes with larger T_s'.

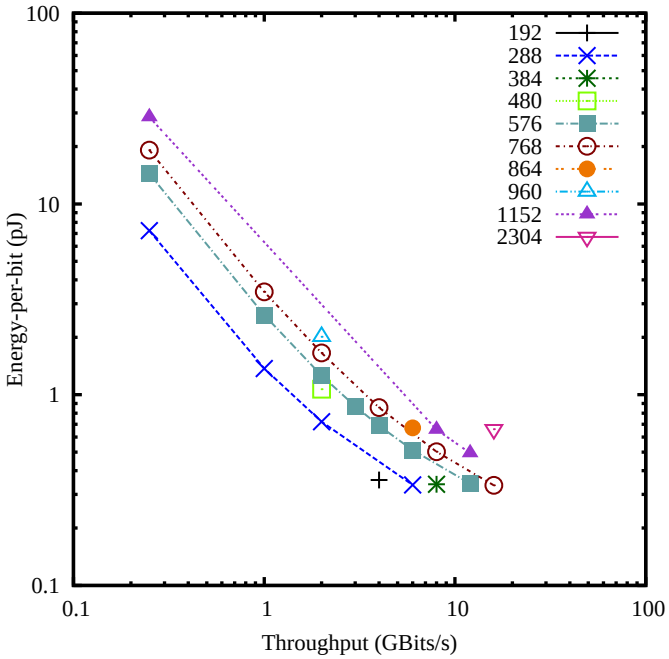

Figure 4.19: Encv3 energy-per-bit versus the information throughput for the PN-LDPC-CCs. The higher the ρ, the higher the information throughput and the lower the energy-per-encoded-bit. Code lengths (T_s) and parallel-node (ρ) values are as specified: for T_s=192 ρ ∈{16}, for T_s=288 ρ ∈{1,4,8,24}, for T_s=384 ρ ∈{32}, for T_s=480 ρ ∈{8}, for T_s=576 ρ ∈{1,4,8,12,16,24,48}, for T_s=768 ρ ∈{1,4,8,16,32,64}, for T_s=864 ρ ∈{24}, for T_s=960 ρ ∈{8}, for T_s=1152 ρ ∈{32,48}, and for T_s=2304 ρ ∈{64}. A clock frequency of 250 MHz is used.

As a general practice, gate swapping is recommended.

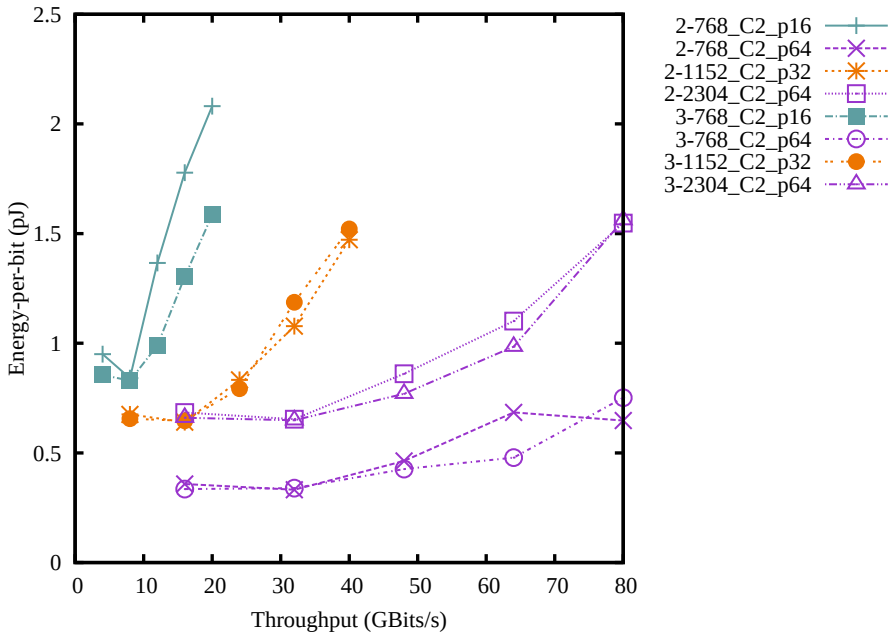

Figure 4.20: Comparison of encv3 and encv2 for various codes in terms of energy-per-encoded-bit versus throughput. The conditional replacement of the 3-input XOR gates slightly reduces the energy-per-encoded-bit. $f_{clk} \in \{250, 500, 750, 1000, 1250\}$MHz for all encoders.

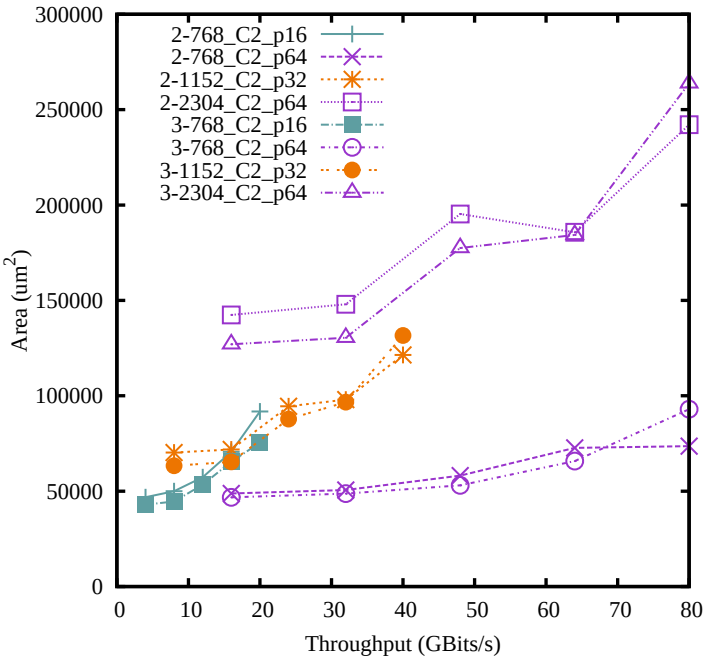

Figure 4.21: Comparison of encv3 and encv2 for various codes with respect to the area versus throughput. Swapping XOR gates for OR gates reduces the required area.

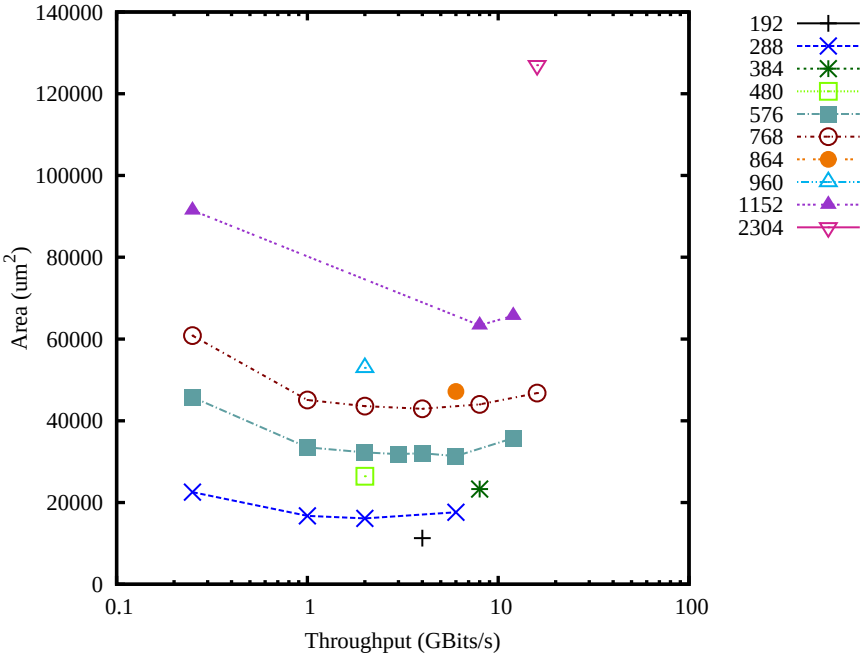

Figure 4.22: Encv3 area versus the information throughput, for the PN-LDPC-CCs. Code lengths (T_s) and parallel-node (ρ) values are as specified: for T_s=192 $\rho \in \{16\}$, for T_s=288 $\rho \in \{1,4,8,24\}$, for T_s=384 $\rho \in \{32\}$, for T_s=480 $\rho \in \{8\}$, for T_s=576 $\rho \in \{1,4,8,12,16,24,48\}$, for T_s=768 $\rho \in \{1,4,8,16,32,64\}$, for T_s=864 $\rho \in \{24\}$, for T_s=960 $\rho \in \{8\}$, for T_s=1152 $\rho \in \{32,48\}$, and for T_s=2304 $\rho \in \{64\}$. A clock frequency of 250 MHz is used.

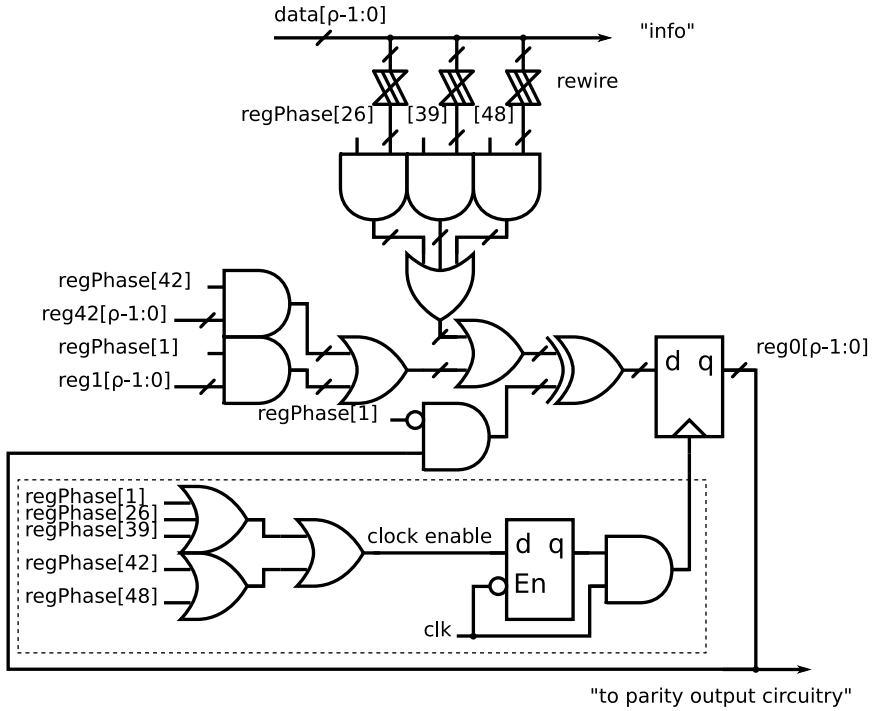

Figure 4.23: Encv4 encoder node. The clock is gated with the "regPhase" signals to reduce the power consumption. One instance of the clock gating circuitry is shared among ρ registers.

4.4 Encoder v4 - Clock-Gating (Encv4)

Clock-gating is a standard strategy for reducing dynamic power consumption [29]. Encv4 uses the phase signal to gate the clock to reduce power consumption. For all PN-LDPC-CCs, over all explored clock frequencies, this strategy produces a 23 to 45 percent reduction in power consumption.

In encv4, latch-based clock-gating [30] is applied to the encoder node registers. For each encoder node, the one-hot phase signals associated with the encoder node are ORed together and the result is latched. The output of the latch is ANDed with the clock to form a new clock signal that is used to control the encoder-node registers. Figure 4.23 shows the latch-based clock-gated encv4 encoder node architecture. In essence, the one-hot phase signals are turned into clocks. Referring back to Table 4.1, we see the switching activity of the phase control signals, represented by "regPhase[0]", have switching rates of $1/2 \cdot T_s'$ with respect to that of the clock. These slowly switching control signals make ideal candidates for gating the clock.

Glitches on the gated-clock signal are avoided by using an active-low clock-sensitive latch. The active low clock-gating latch allows the gating control signal (clock enable) to track the latch input until the clock goes high. When the clock goes high, the value of the gating control signal is held until the clock falls. The gating control signal is ANDed with the clock to produce

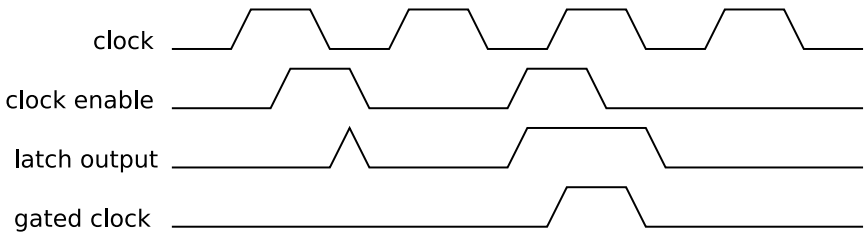

Figure 4.24: Clock-gating timing diagram.

the gated-clock. Only when the clock and the gating control signal are high can the gated-clock be high. Figure 4.24 illustrates the clock-gating timing. Compared to the original clock, the AND gate, after the latch, delays the arrival of the gated-clock signal. This delay allows a slightly longer evaluation time, on the order of an AND gate's propagation delay, for the logic between a non-clock-gated register and the clock-gated register. Conversely, the AND gate delay results in slightly less evaluation time for the logic between the clock-gated register and a non-clock-gated register. Glitches on the gated-clock are prevented in the first half of the clock cycle by the active-low latch continuing to hold its value when the clock is high. Glitches on the gated-clock are prevented in the last half of the clock cycle as the clock is low and thus the output of the AND gate is low. During the falling edge of the clock, glitches are prevented by the path delay of the latch, which allows the falling edge of the clock signal to reach the AND gate before the output of the latch. Conversely, on the rising edge of the clock, glitches are prevented via the synthesis tool meeting the setup-timing constraint that the gating control signal (clock enable) must stabilize at the input to the latch before the clock arrives at the latch and must remain stable after the clock arrives in accordance with the hold-time of the latch. The encoder clocks are now delayed relative to the original clock. The output circuitry of the encoder re-synchronizes the data with the original clock.

Clock-gated flip-flops require the use of asynchronous resets and careful attention to reset design [31]. The clock-gated flip-flops were coded in Verilog as conditionally enabled flip-flops with asynchronous resets. Listing 4.2 shows this using Verilog HDL. The synthesis tool can convert this HDL syntax structure into a clock-gated flip-flop with asynchronous reset.

0.05

```
always @(posedge clk or posedge rstClkPosFF) begin
        if( rstClkPosFF ) begin
                regs0 <= 0;
        end else if(enReg[0]) begin
                regs0 <= ...
    end
end
```

Listing 4.2: Encv4 Verilog code representing a clock-gated flip-flop with an asynchronous reset.

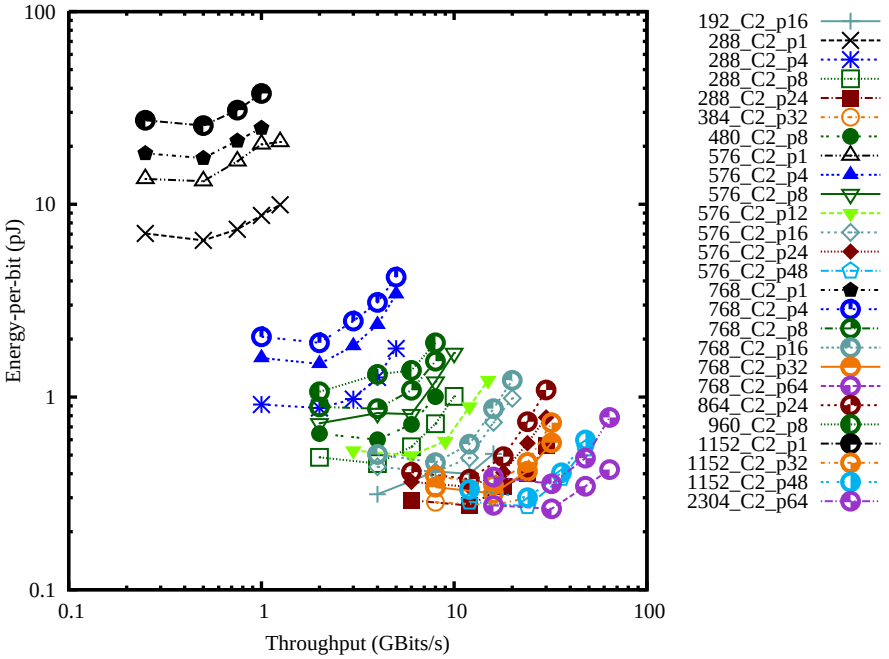

Figure 4.25: Encv4 energy-per-bit versus throughput. An overall reduction in energy-per-encoded-bit is achieved with respect to encv3.

As ρ increases, the energy-per-encoded-bit decreases and the throughput increases. Figure 4.25 shows the energy-per-encoded-bit versus the throughput for our set of PN-LDPC-CCs. We provide this graph as a reference for extremes and trends. The maximum encv4 throughput, produced by the ρ=64 codes, is 64 Gbits/s. The lowest energy-per-encoded-bit, achieved by one of the T_s'=12 codes, is 0.263 pJ/bit. Figure 4.26 show the energy-per-encoded-bit versus the throughput versus ρ for a 250-MHz clock. Each curve in the figure corresponds to a fixed code length. Points along a curve, moving from left-to-right, represent an alternative code with the same T_s but an increased ρ. As ρ increases, energy-per-encoded-bit decreases and throughput increases.

Clock-gating clearly reduces the energy-per-encoded-bit. Figure 4.27 compares encv4 and encv3, for various codes, in terms of their energy-per-bit versus throughput. For a 500-MHz clock, the reduction in energy-per-bit ranges from 17 to 45 percent (see Table 4.2). Clock gating appears to be somewhat more effective at reducing the energy-per-encoded-bit at the lower frequencies than at the higher frequencies (see Appendix C.1). There appears to be no simple correlation between T_s, ρ, T_s' and the reduction in energy-per-bit. However, some rough generalizations can be made at various clock frequencies. For example, at 250 MHz, two factors appear to be at play, T_s and T_s'. The T_s=288 codes, with a T_s' of 36 and 72, have a reduction in energy-per-encoded-bit of 30 percent. The T_s'=24 codes, with a T_s of 576 and 1152, have a reduction in energy-per-encoded-bit of 30 percent. As T_s increases beyond 288, or the

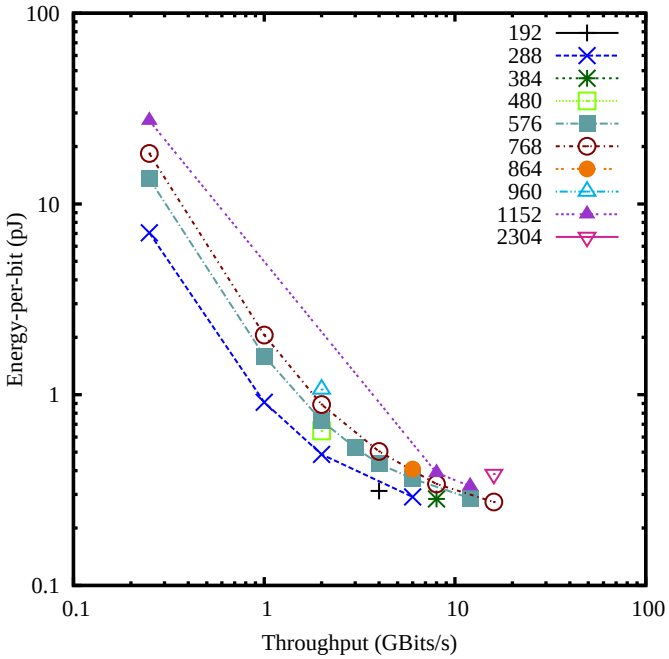

Figure 4.26: Encv4 energy-per-bit versus the information throughput for the PN-LDPC-CCs. The higher the ρ, the higher the information throughput and the lower the energy-per-encoded-bit. Code lengths (T_s) and parallel-node (ρ) values are as specified: for T_s=192 ρ ∈{16}, for T_s=288 ρ ∈{1,4,8,24}, for T_s=384 ρ ∈{32}, for T_s=480 ρ ∈{8}, for T_s=576 ρ ∈{1,4,8,12,16,24,48}, for T_s=768 ρ ∈{1,4,8,16,32,64}, for T_s=864 ρ ∈{24}, for T_s=960 ρ ∈{8}, for T_s=1152 ρ ∈{32,48}, and for T_s=2304 ρ ∈{64}. A clock frequency of 250 MHz is used.

ρ increases above 24, we see a further reduction in energy-per-encoded-bit. Overall, it is clear that clock gating reduces the power consumption.

Clock gating has the unexpected and fortuitous side-effect of typically reducing the required area. Figure 4.28 compares the areas of encv4 and encv3 encoder for various T_s'=36 codes. For this method of clock-gating, less circuit area is required. Given that clock gating requires more logic, the reduction in area is unexpected and may be attributed to a slackening of timing along the critical paths. As the gated clock arrives after the original clock, there is a slightly longer clock cycle in the transition from the non-clock gated circuitry to the clock gated circuitry. Thus there is slightly more time available for logic evaluation. Conversely, the transition from the clock-gated circuitry to the non-clock gated circuitry has a slightly shorter clock period. In other words, although the clock-gated encoder (encv4) uses more gates than the encv3, encv4 consumes less silicon area. We assume the extra gate delay in the gated clock extends the evaluation time from data to gated-clock making equivalent size design faster or equivalent speed designs smaller.

The required area is primarily determined by T_s. Figure 4.29 shows the

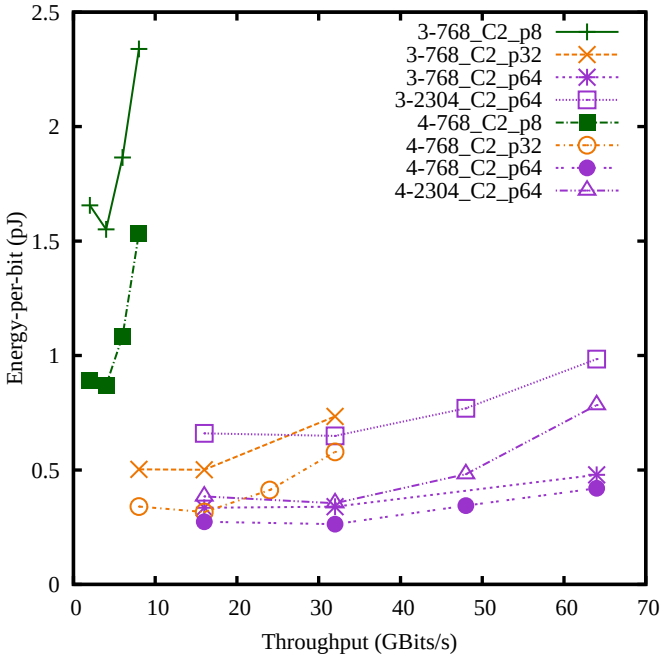

Figure 4.27: Encv4, Encv3, energy-per-bit versus throughput for various codes. Clock gating significantly reduces the power consumption.

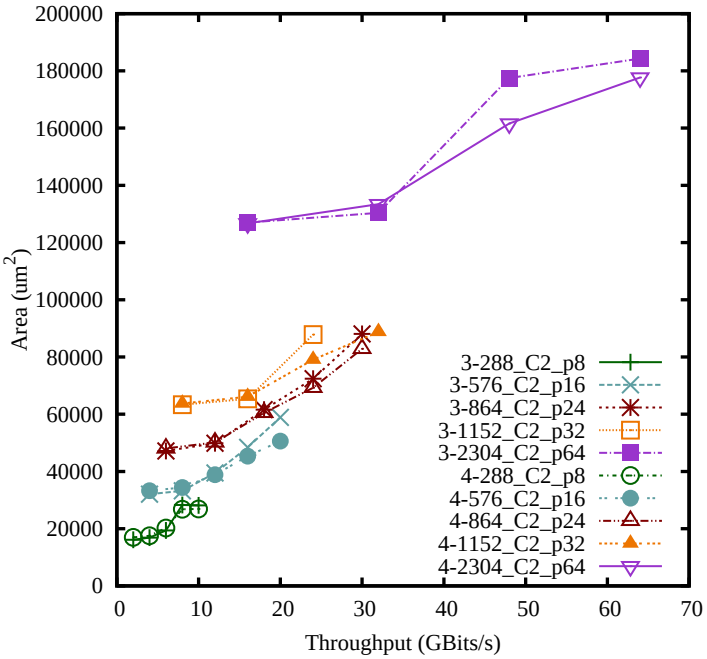

Figure 4.28: Encv4 (4- prefix), Encv3 (3- prefix), area versus throughput for various T_s'=36 codes. Unexpectedly, clock gating reduces the area.

Code	T_s'	Encv3 energy-per-bit (pJ)	Encv4 energy-per-bit (pJ)	Ratio
T_s 1152 ρ 32	36	0.6463	0.3492	1.851
T_s 1152 ρ 48	24	0.4911	0.3004	1.635
T_s 2304 ρ 64	36	0.6490	0.3548	1.829
T_s 288 ρ 24	12	0.3294	0.2731	1.206
T_s 288 ρ 4	72	1.309	0.8776	1.492
T_s 288 ρ 8	36	0.7026	0.4515	1.556
T_s 480 ρ 8	60	0.9882	0.6006	1.645
T_s 576 ρ 12	48	0.8391	0.4971	1.688
T_s 576 ρ 16	36	0.6645	0.4092	1.624
T_s 576 ρ 24	24	0.5046	0.3414	1.478
T_s 576 ρ 4	144	2.466	1.485	1.661
T_s 576 ρ 8	72	1.199	0.8260	1.452
T_s 768 ρ 16	48	0.8294	0.4561	1.818
T_s 768 ρ 32	24	0.5008	0.3162	1.584
T_s 768 ρ 4	192	3.258	1.910	1.706
T_s 768 ρ 64	12	0.3394	0.2630	1.290
T_s 768 ρ 8	96	1.551	0.8705	1.782
T_s 864 ρ 24	36	0.6794	0.3747	1.813
T_s 960 ρ 8	120	1.965	1.310	1.500

Table 4.2: Encv4, Encv3, comparison of energy-per-encoded-bit for a 500-MHz clock. A larger ratio represents a greater reduction in energy-per-bit.

standard cell area versus throughput versus ρ. Each curve in the figure corresponds to a fixed code length. Points along a curve, moving from left-to-right, represent an alternative code with the same T_s but an increased ρ.

In summary, clock-gating slightly reduces the throughput for some codes, slightly reduces the silicon area and significantly reduces the power consumption for all codes. For our PN-LDPC-CC encoder architecture, clock-gating is recommended as a very effective method of reducing power consumption.

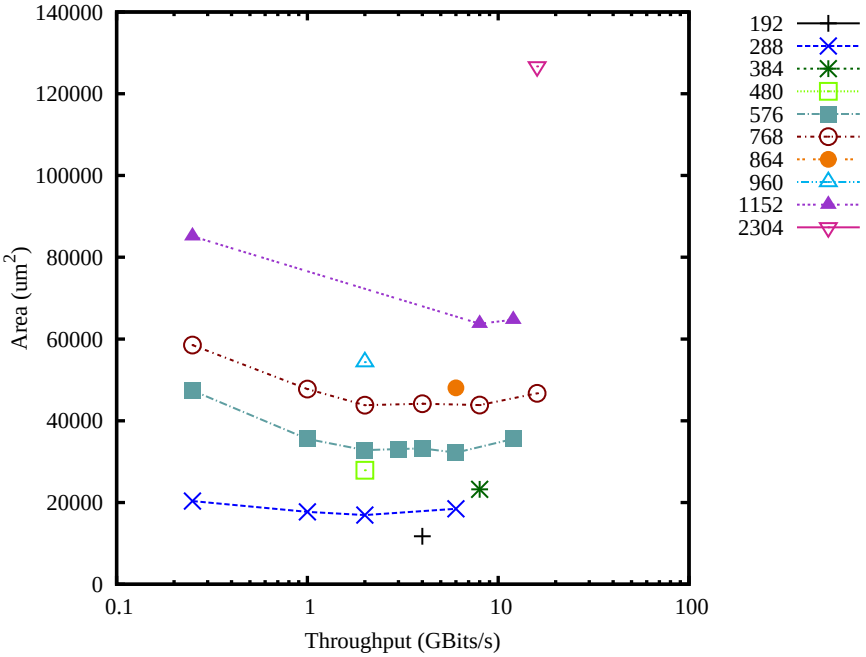

Figure 4.29: Encv4 area versus the information throughput, for the PN-LDPC-CCs. Code lengths (T_s) and parallel-node (ρ) values are as specified: for T_s=192 $\rho \in \{16\}$, for T_s=288 $\rho \in \{1,4,8,24\}$, for T_s=384 $\rho \in \{32\}$, for T_s=480 $\rho \in \{8\}$, for T_s=576 $\rho \in \{1,4,8,12,16,24,48\}$, for T_s=768 $\rho \in \{1,4,8,16,32,64\}$, for T_s=864 $\rho \in \{24\}$, for T_s=960 $\rho \in \{8\}$, for T_s=1152 $\rho \in \{32,48\}$, and for T_s=2304 $\rho \in \{64\}$. A clock frequency of 250 MHz is used.

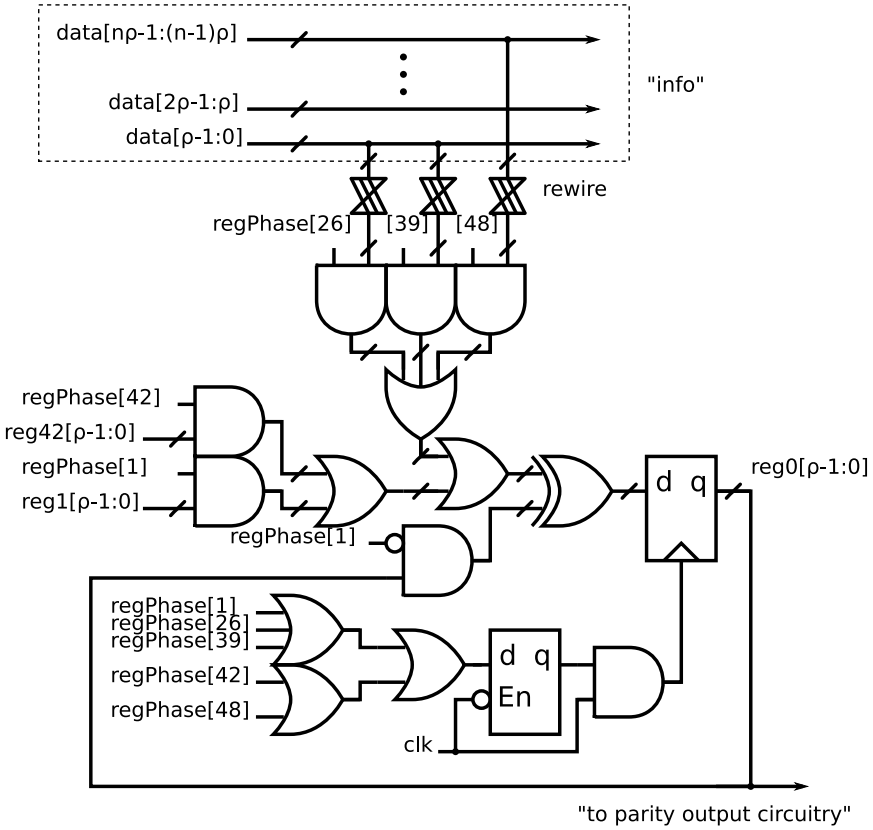

Figure 4.30: Encv5 encoder node. The input databus is expanded to reduce the switching activity on a high fanout net, which in turn, lowers the overall power consumption.

4.5 Encoder v5 - Variable Input Databus Size (Encv5)

In encv5 we increase the input databus width to reduce the power consumption. In previous versions of the encoder, the data bus was attached to all encoder registers and was scaled according to the ρ. As a consequence, codes with a larger ρ would have smaller fanouts on the individual lines in the input databus. Encv5 will allow us to vary the input databus at multiples of the ρ. As a result we expect to see a decrease in power consumption for the codes with smaller ρ. In addition, we feel that having the same size of input databus, regardless of ρ, allows the codes to be compared more fairly. Figure 4.30 shows the encv5 encoder node. Note that the number of input databusses has been increased. This decreases the fanout on each databus.

In previous versions of the encoder, each bit in the input databus would have a 50 percent chance of toggling per cycle and is capacitively loaded by T_s' encoder nodes. Doubling the number of databus inputs allows us to update the input databus every other clock cycle. This reduces the overall power consumption. Note that each databus input is now connected to half the number of encoder nodes. While there are now twice as many inputs, those

inputs switch at half their normal rate. This results in fewer gates switching per clock cycle. Equations 4.3–4.5, show the relationship between the size of the input databus, the rate at which those inputs switch, and the number of associated gates affected per cycle.

$$W_{u(t)} = \rho \cdot M_{inputDB} \tag{4.3}$$

$$R_{u(t)} = 1/M_{inputDB} \tag{4.4}$$

$$\begin{aligned} N_{gatesEffectedPerCycle} &= \frac{W_{u(t)} \cdot T_s' \cdot N_{gatesPerEncNode} \cdot R_{u(t)}}{\rho \cdot M_{inputDB}} \\ &= \frac{(\rho \cdot M_{inputDB}) \cdot T_s' \cdot N_{gatesPerEncNode}}{\rho \cdot M_{inputDB}} \times \frac{1}{M_{inputDB}} \\ &= \frac{T_s' \cdot N_{gatesPerEncNode}}{M_{inputDB}} \end{aligned} \tag{4.5}$$

where

$W_{u(t)}$ is the width of the input databus,

$M_{inputDB}$ is the input databus multiplier,

T_s' is the number of encoder nodes,

$R_{u(t)}$ is the rate at which the input databus switches,

$N_{gatesPerEncNode}$ is the number of gates in an encoder node affected by changes in the input databus, and

$N_{gatesEffectedPerCycle}$ is the number of gates affected by the input databus per clock-cycle.

As can be seen, increasing the input databus size $W_{u(t)}$ is expected to decrease the power associated with the input databus by the multiple $M_{inputDB}$ of the databus size increase. Note that the dynamic power of the wires does not change if the wire length stays the same, as the wire switching remains the same before and after increasing the databus width.

For ease of implementation, the databus size should be a multiple of the ρ. If the input databus size is not a multiple of ρ then more complex circuitry is needed to shuffle data around so that it gets to the correct encoder node. This defeats the goal of reducing the power consumption. In addition, T_s' should be wholly divisible by the multiple of the ρ; otherwise, the input vector must be padded with known values to compensate. Table 4.3 shows the codes that can be implemented with a 64-bit input databus with only minor modifications.

Increasing the input databus to 64 bits slightly reduces the power consumption for those codes with a lower ρ. As expected, when ρ increases to 64, the reduction in power consumption diminishes to zero. Figure 4.31 compares encv5 with encv4 in terms of energy-per-encoded-bit versus throughput for

Code	T_s'	64-bit Input DB ρ Multiple	T_s' / ρ Multiple	Simple Implementation
T_s=1152 ρ=32	36	2	16	yes
T_s=1152 ρ=48	24	1.33	18	no
T_s=2304 ρ=64	36	1	36	yes
T_s=288 ρ=24	12	2.67	4.5	no
T_s=288 ρ=4	72	16	4.5	no
T_s=288 ρ=8	36	8	4.5	no
T_s=480 ρ=8	60	8	7.5	no
T_s=576 ρ=12	48	5.33	9	no
T_s=576 ρ=16	36	4	9	yes
T_s=576 ρ=24	24	2.67	9	no
T_s=576 ρ=4	144	16	9	yes
T_s=576 ρ=8	72	8	9	yes
T_s=768 ρ=16	48	4	12	yes
T_s=768 ρ=32	24	2	12	yes
T_s=768 ρ=4	192	16	12	yes
T_s=768 ρ=64	12	1	12	yes
T_s=768 ρ=8	96	8	12	yes
T_s=864 ρ=24	36	2.67	13.5	no
T_s=960 ρ=8	120	8	15	yes

Table 4.3: Encv5 check for a simple implementation of a 64-bit input databus. A simple implementation is possible when T_s' is wholly divisible by the multiple of ρ.

various codes for an input databus width of 64. Those codes with a ρ significantly lower than 64 benefit the most from the increased databus width. For the T_s=768 codes with a ρ of 4, 8, 16, and 32, the reduction in energy-per-encoded-bit is 23%, 23%, 5% and -1%, respectively. As the T_s=2304, ρ=64 code already has a 64-bit input bus, the energy-per-encoded-bit remains unchanged.

Increasing the input databus has no effect on the area. Figure 4.32 shows that the area remains relatively unchanged.

In summary, increasing the input bus width may not always be an option; however, should the option present itself, a reduction in the energy-per-encoded-bit is possible. Increasing the databus size has little to no impact on the area; however, there were certain cases where the maximum throughput was reduced. In general, if it is possible to increase the input databus, this option should be evaluated for a possible reduction in the power consumption.

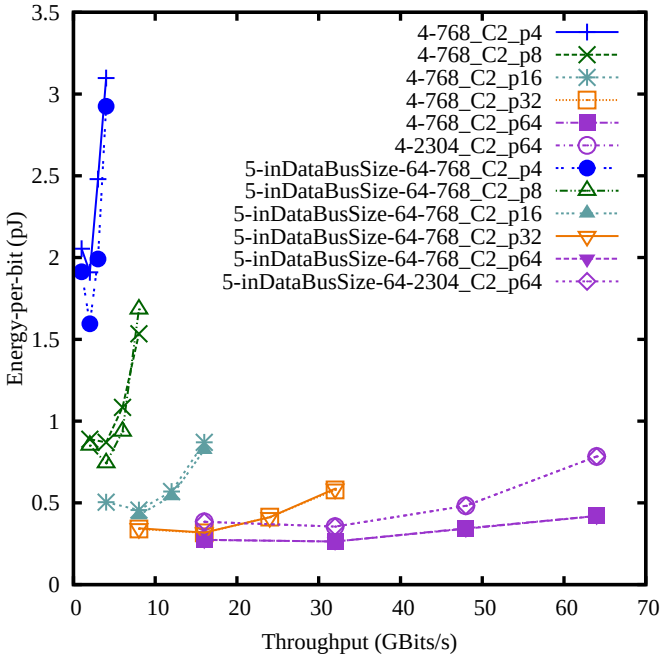

Figure 4.31: Comparing encv5 and encv4 with respect to the energy-per-encoded-bit versus throughput for various codes. As expected, increasing the databus width to 64 bits has the largest impact in terms of energy-per-encoded-bit on those designs/codes with the smallest initial databus widths (ρ).

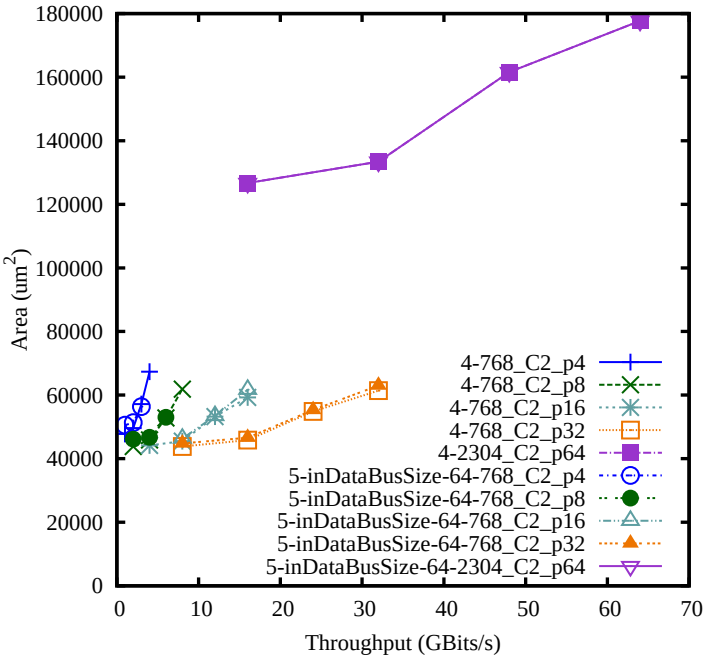

Figure 4.32: Comparing encv5 and encv4 with respect to the area versus the throughput for various codes. Increasing the databus width to 64 bits results in a negligible change in area.

4.6 Encoder v6 - De-multiplexing the Input Databus (Encv6)

In encv5 we explored increasing the databus width to reduce the power. In encv6 we will add a de-multiplexor in between the input databus and the encoder nodes (please see Appendix D.1).

4.7 Encoder v7 - Merged PN-LDPC-CC Encoders (Encv7)

Merging multiple PN-LDPC-CC encoders and sharing control circuitry does not further reduce the energy-per-encoded-bit (see Appendix C.2).

Code	E_b/N_0 for BER 10^{-6}	Tech (nm)	Info. throughput (Gbits/s)	Energy per info-bit (pJ)	Cell area (μm^2)	Incl. termination
T_s=2304 ρ=64 (this work)	1.21	90	64	0.783	177,701	no
T_s=2304 ρ=64 (this work)	1.21	90	48	0.482	161,557	no
T_s=2304 ρ=64 (this work)	1.21	90	32	0.355	133,424	no
T_s=1152 ρ=96 (this work)	1.24	90	96	0.568	102,205	no
T_s=1152 ρ=96 (this work)	1.24	90	48	0.274	71,841	no
T_s=2048 P8 [17]	1.25	90	3.60	na	761,900	no
T_s=2048 P4 [17]	1.25	90	1.95	na	426,300	no
T_s=2048 P2 [17]	1.25	90	1.09	na	279,300	no
T_s=2048 P1 [17]	1.25	90	0.57	na	189,800	no
T_s=768 ρ=64 (this work)	1.3	90	64	0.420	62,586	no
T_s=768 ρ=64 (this work)	1.3	90	48	0.344	58,328	no
T_s=768 ρ=64 (this work)	1.3	90	32	0.263	47,920	no
T_s=288 ρ=24 (this work)	1.7	90	30	0.562	27,776	no
T_s=128 P8 [17]	2.5	90	5.41	na	80,400	no
T_s=128 P4 [17]	2.5	90	2.70	na	48,400	no
T_s=128 P2 [17]	2.5	90	1.35	na	28,200	no
T_s=128 P1 [17]	2.5	90	0.71	na	28,200	no
(8158,7136) QC [32]		250	0.86	628	10,562,500	yes

Table 4.4: Comparison of LDPC encoders.

4.8 Comparison to other LDPC Encoders

In this section the best of our results will be compared with the state-of-the-art. Table 4.4 compares our results with those reported by others, in terms of E_b/N_0 for a BER of 10^{-6}, process technology, information throughput, energy-per-encoded information-bit and standard cell area. The "Incl. termination" column indicates if the encoder has built-in termination circuitry. Termination is the method by which streams of data are ended in LDPC-CCs and the encoder returned to a known state. For a discussion on termination and its application to PN-LDPC-CCs see [1].

The primary metric of comparison used in [17] is the area divided by the information throughput. Our encoder, with a BER of 10^{-6} for an E_b/N_0 of 1.24, has an area over throughput ratio of 1.06 μm^2/Mbps. For the 2048 code, the best area over throughput result presented in [17] is 211.6 μm^2/Mbps. Our result is 205 times better than the best result presented in [17]. For the low coding performance codes, with a E_b/N_0 of 2.5 for a BER of 10^{-6}, our encoder has an area over throughput ratio 16 times better than the best result presented in [17]. Note that our code also achieves a BER of 10^{-6} at an E_b/N_0 of 1.7 dB compared to 2.5 dB in [17].

4.9 PN-LDPC-CC Encoder Summary

In this chapter we have presented a series of novel encoder architectures for the PN-LDPC-CCs. The novel architectures have removed the hardware complexity relationship to the node parallelization factor ρ, while retaining the benefits of ρ: increased throughput and reduced energy-per-encoded-bit. The result of applying the new ideas is a new benchmark for LDPC-CC encoders in terms of energy-per-encoded-bit and throughput. For a PN-LDPC-CC that achieves a BER of 10^{-6} at an E_b/N_0 of 1.21 dB, we presented an encoder architecture that has an area of 177,701 μm^2, an energy-per-encoded-bit of 0.783 pJ and a throughput of 64 Gbits/s. For a PN-LDPC-CC that achieves a BER of 10^{-6} at an E_b/N_0 of 1.3 dB, we presented an encoder architecture that has an area of 62,586 μm^2, an energy-per-encoded-bit of 0.42 pJ and a throughput of 64 Gbits/s.

The encv2 architecture for the T_s=2304, ρ=64 code has an energy-per-encoded-bit of 1.101 pJ and the encv5 architecture has an energy-per-encoded-bit of 0.7834 pJ. This represents a 29 percent reduction in energy-per-encoded-bit.

Given two codes with the same T_s, it is always advantageous to choose the code with the larger ρ.

The keys to the development of the novel architectures is the realization that all LDPC-CCs have a control-path that is independent of the data-path, and that this allows:

- Pipelining within the datapath without consequence to the complexity of the control-path.
- Datapath operations to be re-ordered (provided data dependences are retained).
- One control circuit to control multiple copies of the datapath.
- The removal of all perceived conditional operations within the data path and control path. Note there are no actual conditions, there are only data dependencies.

Applying the preceding keys, the presented PN-LDPC-CC architectures have established the following relationships:

$$Throughput = clock\,frequency \cdot \rho \tag{4.6}$$

$$energy\text{-}per\text{-}encoded\ bit \propto T_s' \tag{4.7}$$

$$area \propto T_s \tag{4.8}$$

This ends our discussion of the PN-LDPC-CC encoder. In Chapter 5, we discuss the new PN-LDPC-CC decoder.

Chapter 5

Parallel-Node LDPC-CC Decoder Architecture Exploration

In this chapter novel Parallel-Node LDPC-CC (PN-LDPC-CC) decoder architectures are presented that have the lowest reported energy-per-decoded-bit and the greatest throughput performance. First we will describe the pre-existing decoder design and then proceed to describe a series of improvements. Each improvement is quantified in terms of the energy-per-decoded-bit versus the throughput, the area versus the throughput, and the energy-per-decoded-bit versus the group period T_s'. Results presented herein are based on synthesis results and adhere to the following conditions. Unless otherwise specified, all results are expressed on a per-processor basis, the presented results have been obtained using an E_b/N_0 of 1.8 dB, and the LLRs have a magnitude precision of 4 bits and a sign bit. An E_b/N_0 of 1.8 dB is chosen because it represents an extreme worst-case condition under which practical implementations of these codes could be expected to operate. To fairly assess timing information, three decoder processors are synthesized together. The middle processor receives its inputs from the first decoder processor and its outputs go to the third decoder processor. Worst-case timing cell libraries are used. Bit-accurate simulations with 32,000 input vectors are used to determine internal net switching activity. From the switching activity, dynamic power can be estimated. For more details on the methods used to gather the decoder results, please see Appendix A.

To facilitate efficient LDPC decoder implementations we found it to be advantageous to link the phase with memory addressing. Specifically, the same memory address is used at the same phase in every period. For this to hold true: $m_s <= T_s$. Our PN-LDPC-CC codes have all been designed to ensure that this is always the case.

All decoders discussed in this section process one phase per clock cycle. While this constraint is possible in both the encoder and decoder, using multiple clock cycles per phase increases the overall amount of switched gate-

capacitance. Since our decoders process one phase per cycle, the information throughput of the decoder is equal to the clock frequency multiplied by ρ.

In the following sections, a series of PN-LDPC-CC decoder architectures are presented. Section 5.1 reviews the first PN-LDPC-CC decoder. Section 5.2 describes a design that removes the saturation bit to reduce the power consumption and the area. Section 5.3 describes a design that removes the rotation switch matrix to remove the greater than linear hardware area dependence on ρ. Section 5.4 describes a design that uses clock-gating to reduce the power consumption. Section 5.5 describes a design that removes the reset circuitry to reduce the area. Section 5.6 describes a design that uses an alternative check-node operation to improve the BER performance while reducing the power consumption and the area. Section 5.7 analyzes hardware metrics and BER performance trade-offs. Section 5.8 compares the best of the new PN-LDPC-CC decoders to the state-of-the-art. Section 5.9 summarizes the main conclusions that arise from the new PN-LDPC-CC decoders.

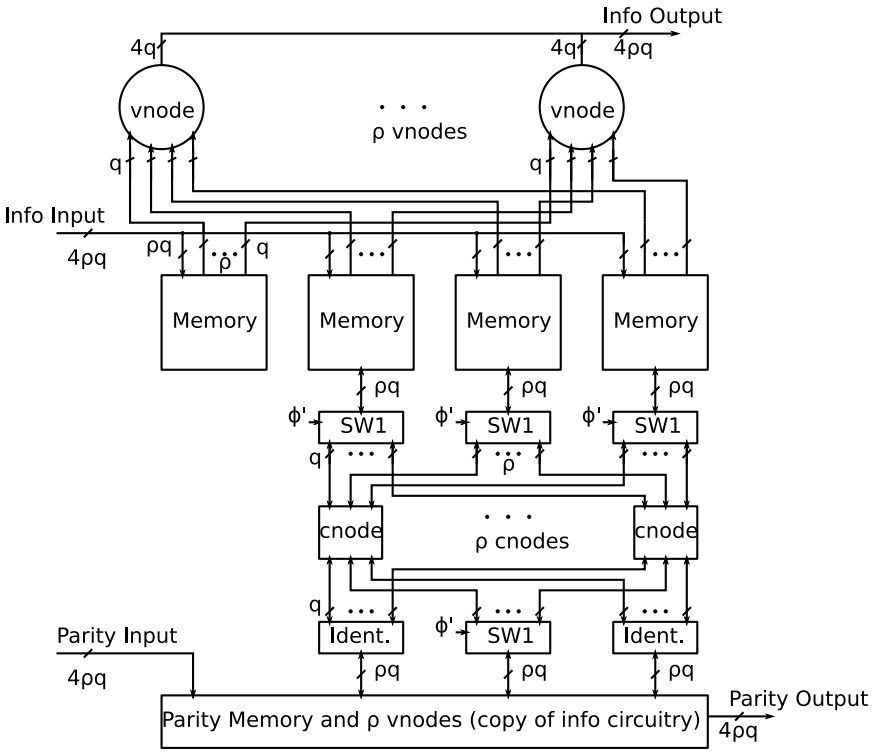

Figure 5.1: Decv1 processor. Consists of eight memory banks: four banks for the information LLRs and four banks for the parity LLRs. The number of variable-nodes and check-nodes is equal to the ρ. The rotation switch matrices, "SW1", have a hardware implementation complexity of order(ρ^2).

5.1 Decoder v1 - Parallel-Node LDPC-CC Decoder (Decv1)

Decv1, like our PN-LDPC-CC encoders, displays the same relationship between ρ, throughput and the energy-per-decoded-bit. As ρ increases, the throughput increases and the energy-per-decoded-bit decreases. Initial work on our baseline PN-LDPC-CC decoder, decv1, was presented in [1]. Figure 5.1 shows the decv1 processor. Similar to previous LDPC-CC decoder implementations [2], this decoder uses cascaded decoder processors to form the decoder. A decoder processor consists of variable-node units, check-node units and multiple memories. Unlike previous LDPC-CC decoder processors the PN-LDPC-CC decoder includes rotation switch-matrices (SW1) and multiple copies, for $\rho > 1$, of the variable-node and check-node units per processor. Based on the phase ϕ, the rotation switch-matrices perform the rotation and reverse rotation operations on LLR data routed between the memories and the ρ check-nodes.

As is shown in Figure 5.1, LLR data from the channel or previous decoder

processor enters on the info and parity input lines and is stored in memory, with the exception of one of the parity input LLRs, that is routed directly through the memory and into a switch matrix SW1. The bit-width of the information input databus is equal to $4\rho q$, where 4 represents the degree of the variable-node, ρ is the node parallelization factor and q is the LLR bit-width. In each group phase ϕ', LLRs from the info and parity memories are read and routed via SW1 to the appropriate check-node units. After the check-node operations, the processed LLRs are routed back to the memory locations they were read from, with exception of one of the parity LLRs which is routed directly to a variable-node unit. Concurrent with the check-node operation, the memories supply processed LLRs to the variable-node units. LLRs leaving variable-node units make up the decoder processor info and parity outputs. The info and parity LLR outputs can be routed to the next decoder processor for another decoding iteration, or they can be routed to the hard decision circuitry where the decoded data stream is formed. Listing 5.1 contains pseudo code for the decoder operation.

0.05

```
pcm = parity_check_matrix
current_phase = 0
while( not end of data )
    rho_llr_data[0] = channel_data()
    // start the processing at the last processor
    foreach decoder_processor in parallel reverse all_decoder_processors
        memoryStoreInputLLRs( pcm, current_phase ) = \
                    rho_llr_data[decoder_processor]

        rho_llr_cnode_inputs = memoryCnodeAccess( pcm, current_phase)
        rho_llr_cnode_inputs = switchMatrixRotate( rho_llr_cnode_inputs, pcm,\
                    current_phase )

        rho_llr_cnode_outputs = cnode( rho_llr_cnode_inputs )

        rho_llr_cnode_outputs = switchMatrixRotateReverse( \
                    rho_llr_cnode_outputs, pcm, current_phase )
        memoryStoreCnodeOutputs( rho_llr_cnode_outputs, pcm, current_phase )

        rho_llr_vnode_inputs = memoryVnodeAccess( pcm, current_phase )
        rho_llr_data[decoder_processor+1] = vnode( rho_llr_vnode_inputs )
    end foreach

    current_phase = (current_phase + 1) MOD T's
end while
```

Listing 5.1: Decoder pseudo code.

When the decoder is reset, all LLR sign and magnitude bits are set to "0" while the LLR saturation bit is set to "1". The LLR saturation bit protects invalid LLR data from corrupting valid data in the check-node operation. With the saturation bit set, the LLR will pass through the check-node operation unaltered nor will it in any way alter the check-node operation. The saturation bit can be de-asserted in the variable-node operation. If the saturation bit

is not de-asserted in the variable-node operation, then it is passed into the next processor. If the memory is constructed out of flip-flops then the reset operation can take place in just one or two clock cycles; however, if the memory is SRAM based, then multiple cycles are required as each memory bank address location needs to be sequentially written.

Decv1 uses a simple memory structure, with functionally complex memories, such as an asynchronous dual-port memory. The result is that the memory-to-check-node/variable-node (MCV) switch-matrix, found in [2], is not required. In [2], the focus was on minimizing the number of required memory banks when using synchronous single-port memories. As a result of this approach a MCV switch-matrix is required to interface each memory bank to each input and output of the check-node and variable-node units. The absence of the MCV switch matrix in decv1 saves significant area and power. The MCV's per processor area and power consumption for a (128,3,6) conventional LDPC-CC, in a CMOS 90-nm process, with no input delays or output loads, synthesized with a target clock frequency of 1.5 GHz, is 62,000 μm^2 and 52 mW, respectively.

Decv1 uses 8 register-based memory banks. By "register-based" we mean that the memory banks are constructed as arrays of conventional flip-flops. Four banks are associated with the "info" LLRs and four with the "code" LLRs. Of the four "info" LLR memory banks, three are dual-port memories and one is a signal-port memory. Of the four "code" LLR memory banks, two are dual-port memories, one is a dual-read/single-write memory and one is a single-port memory. The dual-port memory banks have 4 databus ports of size pq and two address ports. Two databus ports and one address port is used for the input and output of variable-node LLRs and two databus ports and one address port is used for the input and output of the check-node LLRs. The dual-read/single-write memory bank is identical to the dual-port memory bank with the except it lacks a check-node input databus port. The single-port memory has 2 databus ports, one for input data and one for output data, and a single address port.

All of the register-based memory banks are capable of one read operation followed immediately by one write operation in the same clock-cycle (hence the need for memory bank input and output databus ports). This allows LLRs to be read from the desired memory location, processed and then written back to that location within one clock-cycle. The register-memories are designed for asynchronous reads and synchronous writes. In other words, after a rising edge of the clock has occurred, the address to the memories changes, which causes the data associated with the updated address to be read out of the memories. Before the end of the clock cycle, the data input to the memories needs to stabilize. Thus when the next rising-edge of the clock arrives, the data input to the memories is captured in the same address location that was just read. If standard synchronous memories were to be used instead of register-based

memories or asynchronous memories, this single clock-cycle read-modify-write operation would need to be split over multiple clock-cycles or, alternatively, more memory locations would need to be used.

The amount of memory required to implement a decoder processor is proportional to the code period T_s, the variable-node degree and the LLR bit width, as expressed in Equation (5.1). In the case of the T_s=480, ρ=8, (3,6) code, with a 6-bit LLR, the total memory size would be 23,040 bits.

$$N_{memoryBits} = T_s \cdot 2 \cdot (N_{vnodeDegree} + 1) \cdot N_{llrBitWidth} \tag{5.1}$$

The input and output databus widths of each register-memory bank is given by Equation (5.2). In the case of the T_s=480, ρ=8, (3,6) code, the input and output databus width of the register-memory banks would be 40 bits.

$$N_{memoryBankIOBusWidth} = N_{llrBitWidth} \cdot \rho \tag{5.2}$$

The number of rows in each memory bank is equal to $T_s' = T_s/\rho$. For example, the T_s=480, ρ=8 code has 60 rows in each memory bank.

The variable-node sums up three of the 4 LLR inputs to calculate each of the 4 outputs. The check-node performs the min-sum operation.

The PN-LDPC-CC decoders exhibit similar characteristics to the PN-LDPC-CC encoders. As ρ increases, the throughput increases and the energy-per-decoded-bit decreases. Figure 5.2 shows the energy-per-decoded-bit versus the throughput for all of our PN-LDPC-CCs. The energy-per-decoded-bit is expressed on a per-processor basis. The 6-bit LLRs have 4 bits of magnitude, one sign bit and one saturation bit. The results are based on worst-case timing libraries from a CMOS 90-nm process technology (for more details on the method used to generate the results, please see Appendix A). For each code, results at clock frequencies of 100 and 200 MHz were generated. The lowest throughput and highest energy-per-decoded-bit is displayed by those codes with ρ=1. As the node-parallelization factor ρ is increased, the throughput increases and the energy-per-decoded-bit decreases. Figure 5.3 illustrates this point. Each curve in the figure corresponds to a fixed code length. Points along a curve, moving from left-to-right, represent an increasing ρ. As ρ increases the energy-per-decoded bit decreases and the throughput increases.

The total area of the decv1 processor correlated to T_s and is also impacted by ρ and the clock frequency. Figure 5.4 shows the area for a processor versus the throughput for the PN-LDPC-CCs. Note that the area increases as the clock frequency is increased from 100 MHz to 200 MHz. Also, the area increases as ρ is slightly increased. Figure 5.5 shows that as ρ increases beyond 8, the area increases. Each curve in the figure corresponds to a fixed code length. Points along a curve, moving from left-to-right, represent an increasing ρ. Finally, the area increases as T_s is increased.

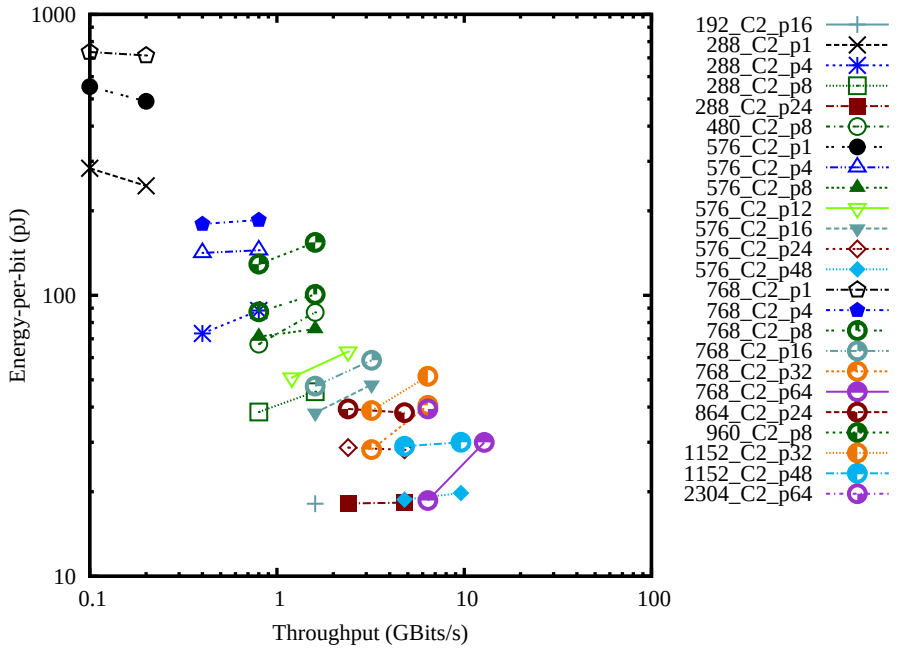

Figure 5.2: Decv1 energy-per-bit versus the throughput for the PN-LDPC-CCs. Similar to the encoders, the higher the ρ, the higher the throughput and the lower the energy-per-decoded-bit. The energy-per-decoded-bit is expressed on a per-processor basis. Clock frequencies of 100 and 200 MHz are used.

There is clearly a strong linear correlation between T_s and the area. Figure 5.6 compares the area for a processor versus T_s for all of the PN-LDPC-CCs at a clock frequency of 100 MHz. Note the strong linear correlation between T_s and the area. For a given T_s, variations in area correspond to variations in ρ. For a 100-MHz clock, in the case of the T_s=768 codes, increasing the ρ from 1 to 64 increases the total processor area by 25%. The combinational circuitry area associated with the variable-nodes and check-nodes increases proportionally to ρ. The ratio of the combinational to non-combinational (flip-flop) area for a ρ of 1 and 64 is 0.832 and 1.425, respectively.

The rotation switch-matrix (SW1) (see Figure 5.1) will produce a phase dependent circulant of the input vector. The number of SW1 units is equal to $2 \cdot N_{codeDegree}$, or in the case of (3,6) codes used here, 6 units. Note that the SW1 components are bi-directional, rotating the LLR values from the memories to the check-nodes and then performing the reverse rotation for the LLR values coming from the check-nodes to the memories. The number of paths in the SW1 component is given by Equation (5.3).

$$N_{SW1Paths} = 2 \cdot N_{llrBitWidth} \cdot \rho^2 \tag{5.3}$$

The complexity of SW1 grows greater than linearly with ρ.

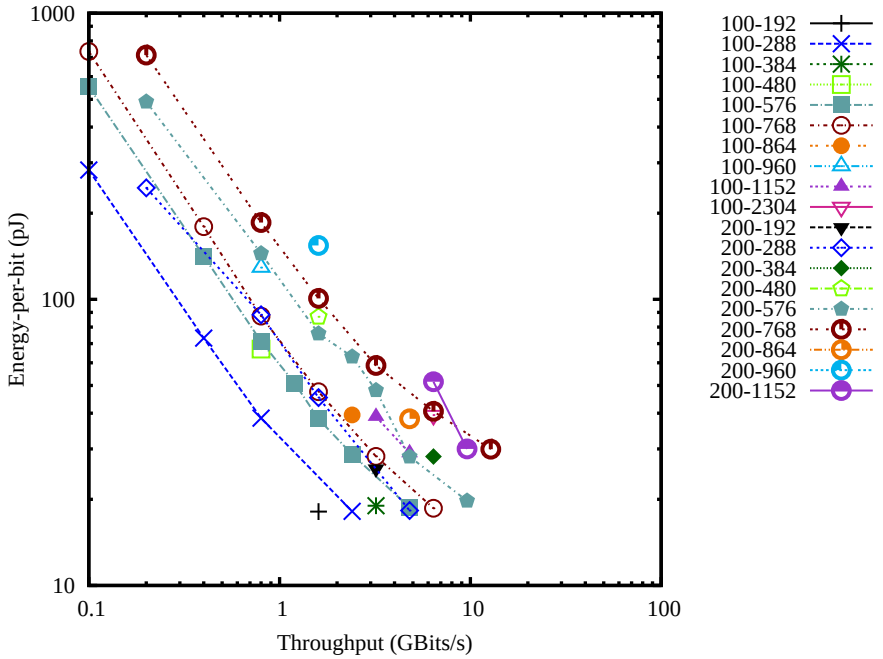

Figure 5.3: Decv1 energy-per-bit versus the information throughput for the PN-LDPC-CCs. Similar to the encoders, the higher the ρ, the higher the information throughput and the lower the energy-per-decoded-bit. The energy-per-decoded-bit is expressed on a per-processor basis. Code lengths (T_s) and parallel-node (ρ) values are as specified: for T_s=192 $\rho \in \{16\}$, for T_s=288 $\rho \in \{1,4,8,24\}$, for T_s=384 $\rho \in \{32\}$, for T_s=480 $\rho \in \{8\}$, for T_s=576 $\rho \in \{1,4,8,12,16,24,48\}$, for T_s=768 $\rho \in \{1,4,8,16,32,64\}$, for T_s=864 $\rho \in \{24\}$, for T_s=960 $\rho \in \{8\}$, for T_s=1152 $\rho \in \{32,48\}$, and for T_s=2304 $\rho \in \{64\}$. Clock frequencies of 100 (prefix 100) and 200 (prefix 200) MHz are used.

The area of the rotation switch matrix (SW1) increases greater than linearly with increasing ρ. Table 5.1 lists the component areas, for a clock frequency of 100 MHz, for T_s=768 codes for ρ equal to 4, 8, 16, 32 and 64. The switch matrix, with ρ=4, starts off very small, representing less that a percent of the total processor area. However, when ρ=64, the switch matrix area represents 15 percent of the total processor area.

The relationships between the decv1 component areas, from Table 5.1, and the characteristics of the PN-LDPC-CCs are as follows. There is a linear relationship between each of the check-node and variable-node areas and the ρ. As was described above, the switch matrix area is greater than linearly dependent on ρ. As was seen in Figure 5.4, the register memory area grows nearly linearly with T_s.

There is a nearly linear relationship between T_s' and the energy-per-decoded-bit. Figure 5.7 compares the energy-per-bit versus T_s' for a 100-MHz clock. The smallest value of T_s' in our set of PN-LDPC-CCs is 12. These codes, with a T_s' of 12, have the lowest energy-per-decoded-bit.

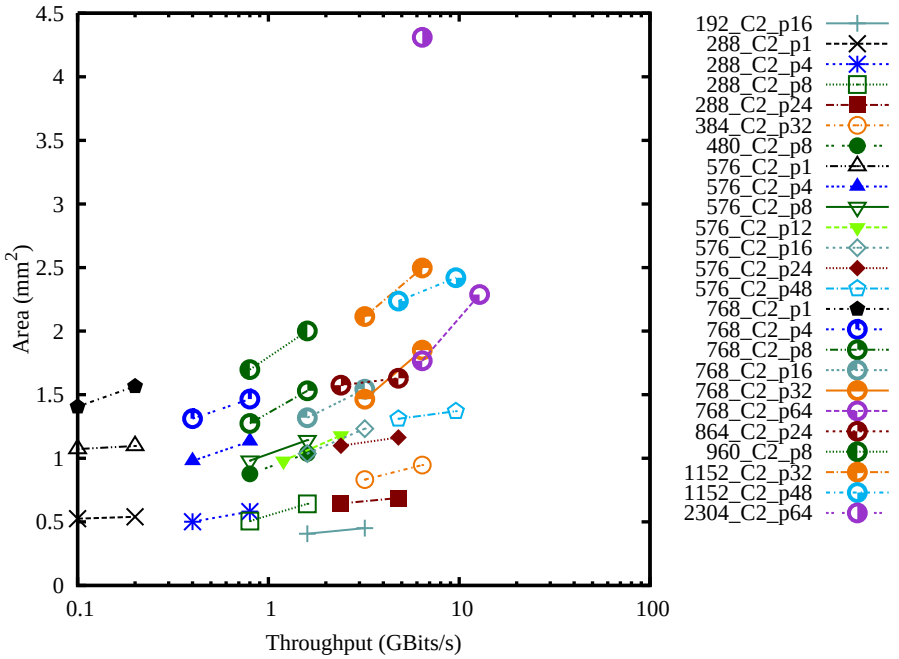

Figure 5.4: Decv1 area for one processor versus the throughput, for all of the PN-LDPC-CCs. Clock frequencies of 100 and 200 MHz are used.

Code decv1 T_s=768	Variable Node (μm^2)	Check Node (μm^2)	Switch Matrix (SW1) (μm^2)	Register Memory (μm^2)	Processor (mm^2)
ρ=4	11264 (1%)	5312 (0%)	5520 (0%)	1277989 (98%)	1.300 (100%)
ρ=8	20288 (2%)	11096 (1%)	15168 (1%)	1190341 (96%)	1.237 (100%)
ρ=16	40576 (3%)	22192 (2%)	40452 (3%)	1181698 (92%)	1.285 (100%)
ρ=32	81152 (6%)	44320 (3%)	103788 (7%)	1184175 (84%)	1.414 (100%)
ρ=64	162304 (10%)	88768 (5%)	249084 (15%)	1170049 (70%)	1.669 (100%)

Table 5.1: Decv1 component standard cell areas for T_s=768 codes, at 100 MHz with a 5-bit LLR. Note the greater than linear increase in the switch matrix area as ρ increases. Also note that the register-memory represents the majority of the decoder processor area. The variable-node and check-node areas increase nearly linearly with ρ.

The critical path for the T_s=768, ρ=64 code running at 200 MHz, starts at a dual-port "memory" flip-flop, flows through a variable-node, out of the processor into a dual-port memory in the next processor. The propagation delay from the rising edge of the clock at the flip-flop to the variable-node is 1.64 ns. From the input of the variable-node to its output the delay is 2.65 ns. From the output of the variable-node to the flip-flop, including a 0.4 ns setup time, the delay is 0.66 ns.

In this section, we have introduced the first parallel-node LDPC-CC decoder (decv1) and have established that:

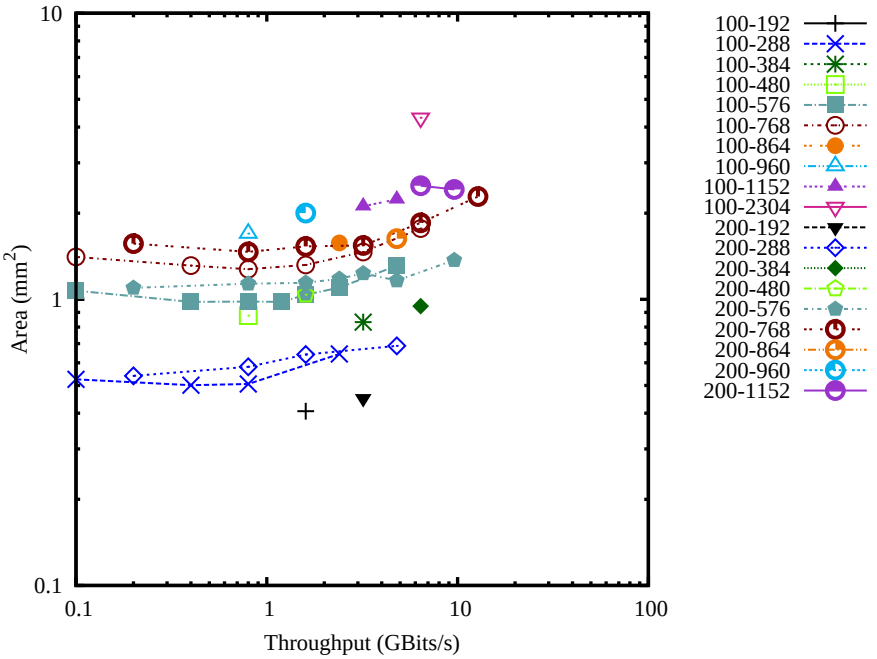

Figure 5.5: Decv1 area for one processor versus the information throughput, for all of the PN-LDPC-CCs. The area is expressed on a per-processor basis. Code lengths (T_s) and parallel-node (ρ) values are as specified: for T_s=192 $\rho \in \{16\}$, for T_s=288 $\rho \in \{1,4,8,24\}$, for T_s=384 $\rho \in \{32\}$, for T_s=480 $\rho \in \{8\}$, for T_s=576 $\rho \in \{1,4,8,12,16,24,48\}$, for T_s=768 $\rho \in \{1,4,8,16,32,64\}$, for T_s=864 $\rho \in \{24\}$, for T_s=960 $\rho \in \{8\}$, for T_s=1152 $\rho \in \{32,48\}$, and for T_s=2304 $\rho \in \{64\}$. Clock frequencies of 100 (prefix 100) and 200 (prefix 200) MHz are used.

- As the node-parallelization factor ρ increases, the energy-per-decoded-bit decreases and the throughput increases.
- The rotation switch-matrix (SW1) grows greater than linearly with ρ.
- There is a strong linear correlation between the decoder processor area and the code period T_s.

Decv1 has been characterized in simulation and synthesis, with respect to energy-per-bit, area and throughput.

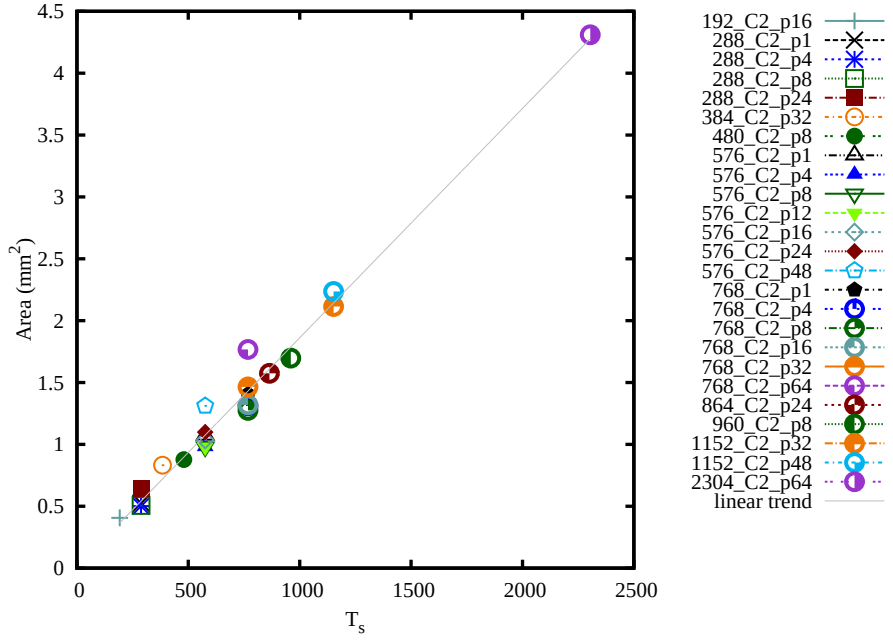

Figure 5.6: Decv1 processor area versus T_s for a 100-MHz clock. Note the strong linear correlation between T_s and the area. For a given T_s, variations in area, correspond to variations in ρ. In the case of the T_s=768 codes, increasing ρ from 1 to 64 increases the total processor area by 25%.

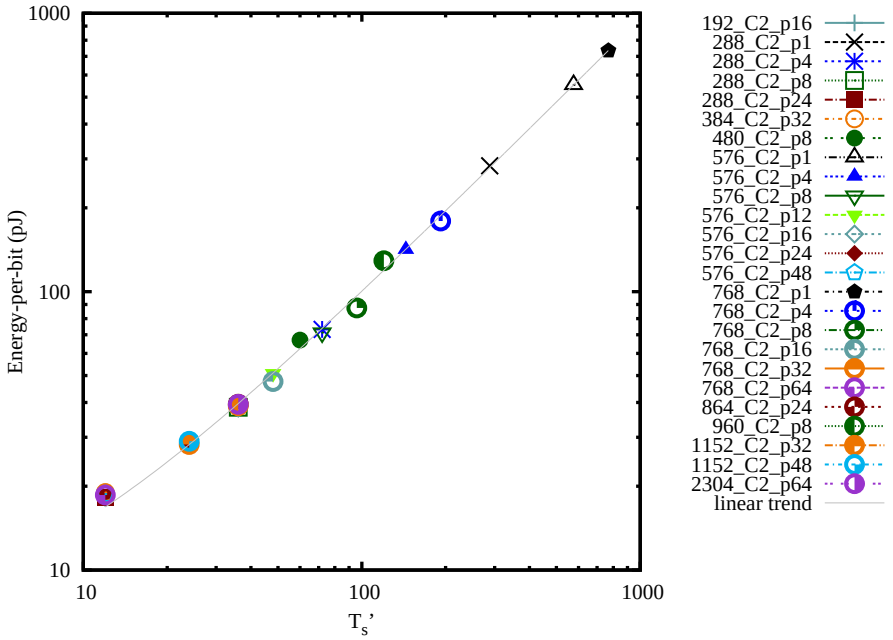

Figure 5.7: Decv1 energy-per-bit versus T_s' for a 100-MHz clock. There is a nearly linear relationship between T_s' and the energy-per-decoded-bit.

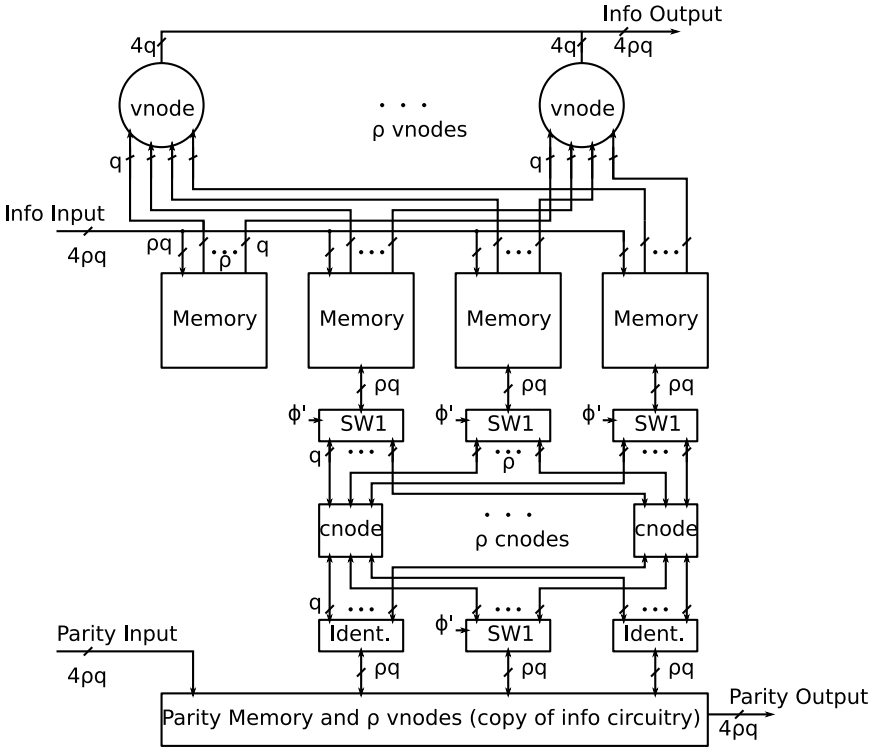

Figure 5.8: Decv2 processor. Decv2 has the same structure as decv1, however, the saturation bit has been removed and its function is emulated by the maximum LLR magnitude. The result is a minor loss of BER performance in short frames, but an area and power savings proportional to $1/N_{llrBitWidth}$.

5.2 Decoder v2 - Removing the Saturation Bit (Decv2)

In this section, a decoder architecture is presented that operates without an LLR saturation-bit. Removing the saturation bit reduces the area by approximately $1/N_{llrBitWidth}$ and reduces the power consumption slightly, with only a small loss in BER performance. Upon reset, the memories are initialized to the maximum LLR magnitude with the sign-bit set to zero. Essentially the function of the saturation bit will be approximated using the maximum LLR magnitude. As can be seen in Figure 5.8, the overall resulting decoder architecture remains unchanged. For shorter data streams, there is a minor decrease in BER performance (according to unpublished work by Zhengang Chen) due to the lack of the saturation-bit protection at the beginning of the data stream. Initializing the LLRs using the maximum magnitude does provide some protection at the beginning of the stream, but it is not as strong as the protection offered by a true saturation bit. In normal operation with a saturation bit, the saturation bit can only be replaced with a signal magnitude value in the check-node, whereas an LLR value at the maximum LLR

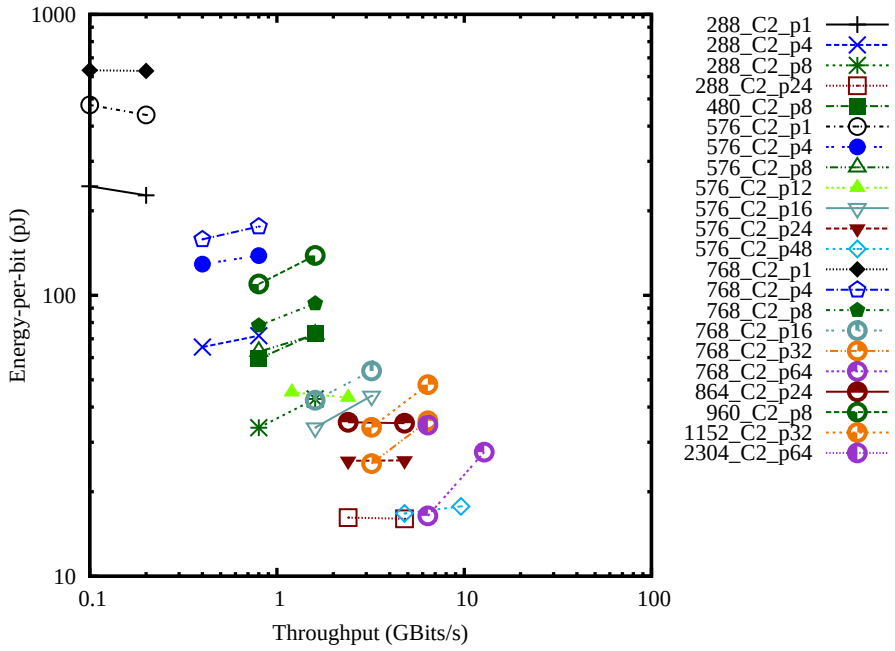

Figure 5.9: Decv2 energy-per-decoded-bit per processor versus throughput for all the PN-LDPC-CCs. Clock frequencies of 100 and 200 MHz are used.

magnitude can be altered in both the check-node as well as the variable-node.

Removing the saturation bit does indeed reduce the energy-per-decoded-bit. Figure 5.9 shows the energy-per-bit versus throughput for all the PN-LDPC-CCs. Figure 5.10 shows the energy-per-decoded bit versus throughput versus ρ. Each curve in the figure corresponds to a fixed code length. Points along a curve, moving from left-to-right, represent an increasing ρ. The general trend of increasing parallelism resulting in lower energy-per-decoded bit, holds true.

For small values of ρ the area of the decoder is primarily determined by T_s. Figure 5.11 shows the required standard cell area for a decoder processor versus the throughput for our set of PN-LDPC-CCs. Figure 5.12 shows the standard cell area versus throughput versus ρ. Each curve in the figure corresponds to a fixed code length. Points along a curve, moving from left-to-right, represent an alternative code with the same T_s but an increased ρ. In addition to the area being correlated with T_s, note that as ρ increases to larger values, the required area begins to grow greater than linearly due to the rotation switch-matrix.

Similar to decv1, decv2 displays a strong linear correlation between T_s and the area. Figure 5.13, for a clock frequency of 100 MHz, compares the area of a processor versus T_s for all of the PN-LDPC-CCs. For a given T_s, variations in area correspond to variations in ρ. For a 100-MHz clock, in the case of the T_s=768 codes, increasing ρ from 1 to 64 increases the total processor area by 24%. With these same changes, Decv1 had a 25% increase in total processor

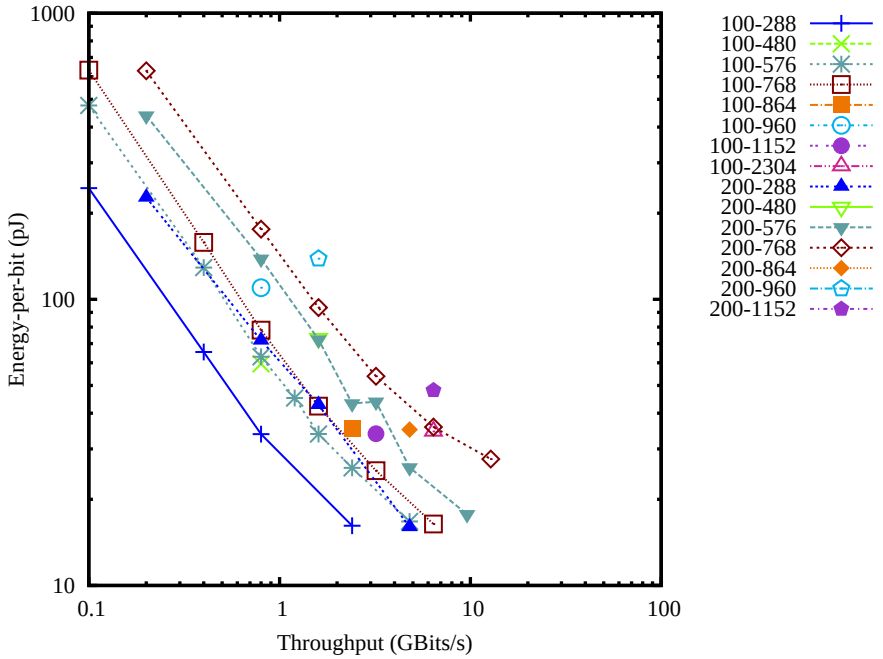

Figure 5.10: Decv2 energy-per-bit versus the information throughput for the PN-LDPC-CCs. The higher the ρ, the higher the information throughput and the lower the energy-per-decoded-bit. The energy-per-decoded-bit is expressed on a per-processor basis. Code lengths (T_s) and parallel-node (ρ) values are as specified: for T_s=192 $\rho \in \{16\}$, for T_s=288 $\rho \in \{1,4,8,24\}$, for T_s=384 $\rho \in \{32\}$, for T_s=480 $\rho \in \{8\}$, for T_s=576 $\rho \in \{1,4,8,12,16,24,48\}$, for T_s=768 $\rho \in \{1,4,8,16,32,64\}$, for T_s=864 $\rho \in \{24\}$, for T_s=960 $\rho \in \{8\}$, for T_s=1152 $\rho \in \{32,48\}$, and for T_s=2304 $\rho \in \{64\}$. Clock frequencies of 100 (prefix 100) and 200 (prefix 200) MHz are used.

area.

Compared to decv1, removing the saturation bit reduces the power consumption by 11.1% on average and the area by 13.9% on average. Figure 5.14 compares decv2 and decv1 in terms of energy-per-decoded-bit versus throughput for various codes. Figure 5.15 compares decv2 and decv1 in terms of area versus throughput for various codes. Again, there is a strong linear correlation between T_s' and the energy-per-decoded-bit. Figure 5.16 compares the energy-per-bit versus T_s' for a 100-MHz clock. The group period T_s' is inversely proportional to ρ. As ρ increases more decoded bits are produced per clock-cycle. The power consumption of a decoder increases with T_s as T_s determines the number of required registers, which in turn determines the size of the clock network. The energy-per-decoded-bit is the power consumption of the decoder, which is nearly linearly related to T_s, amortized over the number of decoded bits produced per cycle, which is directly proportional to ρ. Recall that T_s' is defined to be T_s/ρ. This ensures the energy-per-bit is almost directly proportional to T_s'.

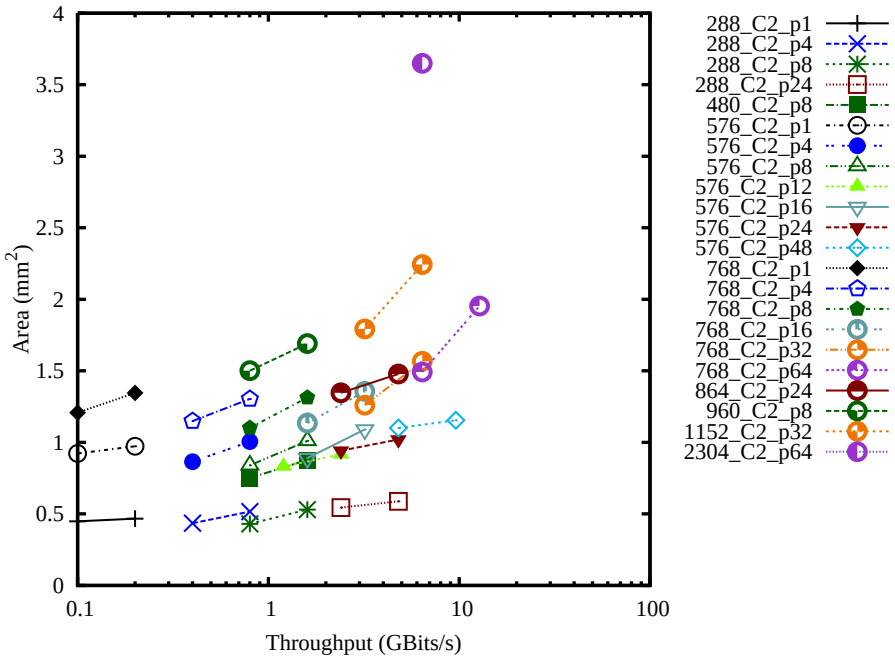

Figure 5.11: Decv2 area for a processor versus throughput for all the PN-LDPC-CCs. The area of the decoder is primarily determined by T_s with ρ having a secondary effect. Clock frequencies of 100 and 200 MHz are used.

The critical path for the T_s=768, ρ=64 code running at 200 MHz, starts at the memory address control and flows through a variable-node output. The delay from the activation of the memory address control to the variable-node input is 1.60 ns, and the delay from the input of the variable-node to its output is 2.85 ns. Finally, the processor output delay is 0.5 ns, as we specified.

While removing the saturation bit results in a minor loss in BER performance for short frames, removing the saturation bit also reduces in energy-per-decoded-bit by 11.1% on average (min. 6.5%, max 22.4%, σ=3.3%) and reduces the area by 13.9% on average (min 11.5%, max 19.1%, σ=1.8%). Thus we recommend removing the saturation bit. Later, in Section 5.5 we show how to compensate for the loss in coding performance caused by dropping the saturation bit.

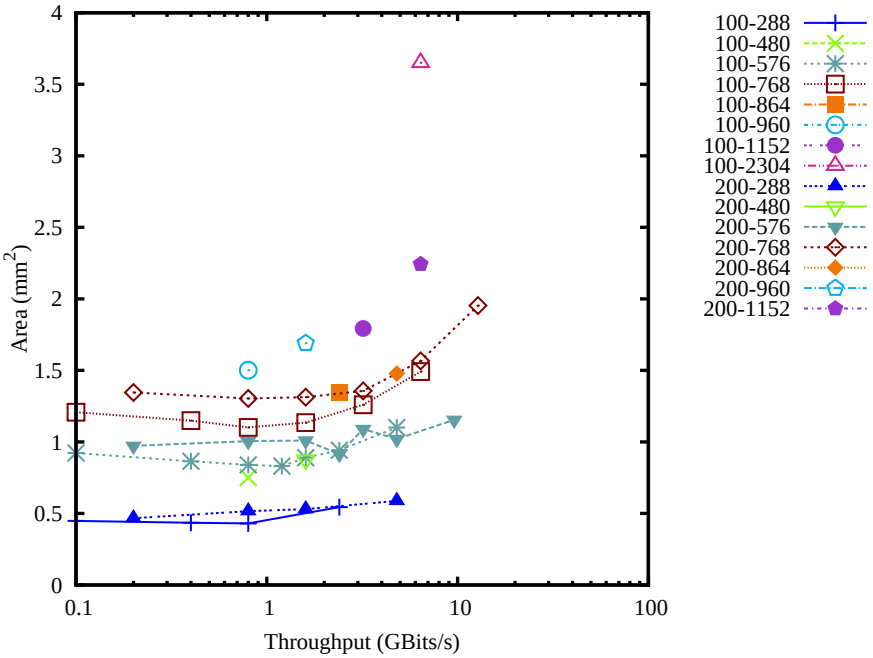

Figure 5.12: Decv2 area for one processor versus the information throughput, for all of the PN-LDPC-CCs. The area is expressed on a per-processor basis. Code lengths (T_s) and parallel-node (ρ) values are as specified: for T_s=192 $\rho \in \{16\}$, for T_s=288 $\rho \in \{1,4,8,24\}$, for T_s=384 $\rho \in \{32\}$, for T_s=480 $\rho \in \{8\}$, for T_s=576 $\rho \in \{1,4,8,12,16,24,48\}$, for T_s=768 $\rho \in \{1,4,8,16,32,64\}$, for T_s=864 $\rho \in \{24\}$, for T_s=960 $\rho \in \{8\}$, for T_s=1152 $\rho \in \{32,48\}$, and for T_s=2304 $\rho \in \{64\}$. Clock frequencies of 100 (prefix 100) and 200 (prefix 200) MHz are used.

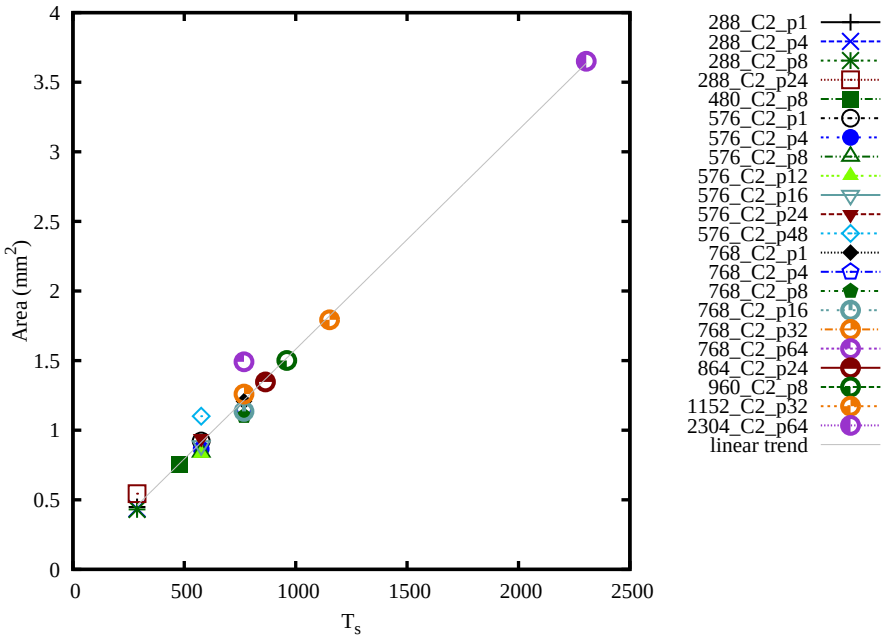

Figure 5.13: Decv2 processor area versus T_s at 100 MHz. Note the strong linear correlation between T_s and the area. For a given T_s, variations in area correspond to variations in ρ. In the case of the T_s=768 codes, increasing ρ from 1 to 64 increases the total processor area by 24%.

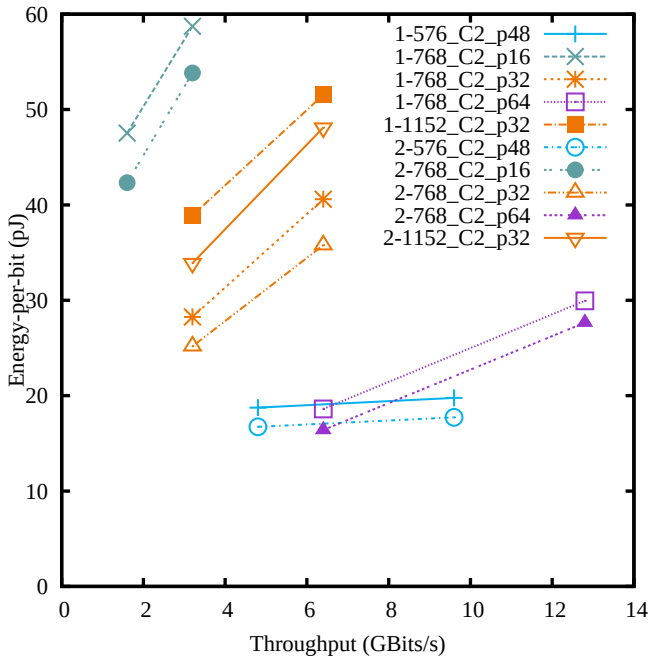

Figure 5.14: Decv2 (prefix 2) and decv1 (prefix 1) compared in terms of the energy-per-decoded-bit versus the throughput for various codes. Removing the saturation bit reduces the LLR bit-width from 6 bits to 5 bits and reduces the energy-per-decoded-bit by 11.1% on average. Clock frequencies of 100 and 200 MHz are used.

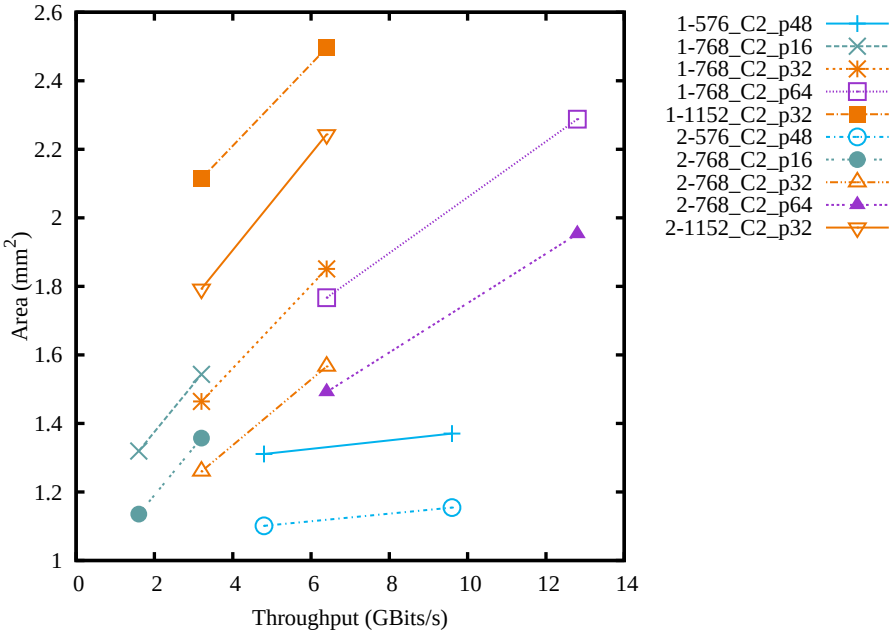

Figure 5.15: Decv2 and decv1 compared in terms of area versus throughput for various codes. Removing the saturation bit reduces the LLR bit-width from 6 bits to 5 bits and reduces the area by 13.9% on average. Clock frequencies of 100 and 200 MHz are used.

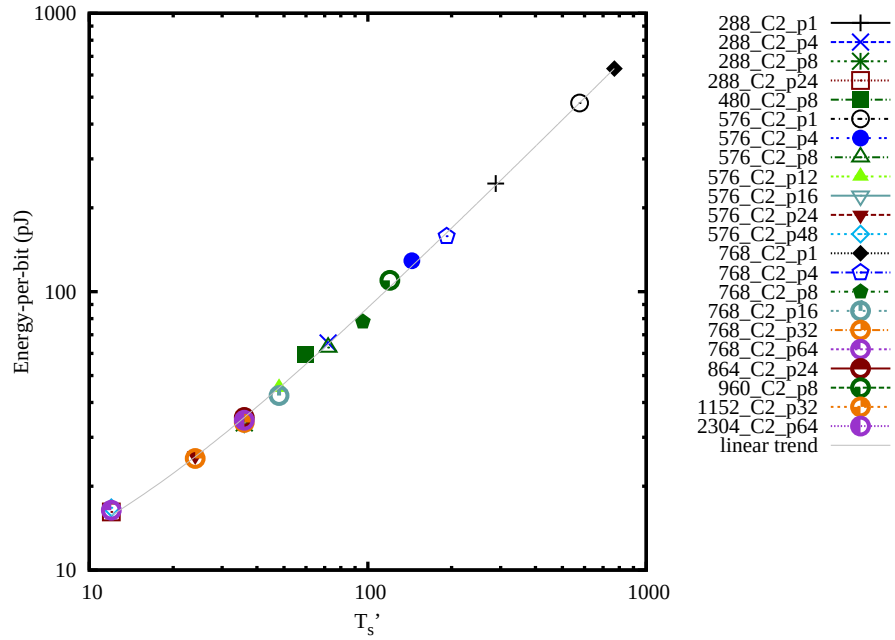

Figure 5.16: Decv2 energy-per-bit versus T_s' for a 100-MHz clock. There is a strong linear correlation between T_s' and the energy-per-decoded-bit.

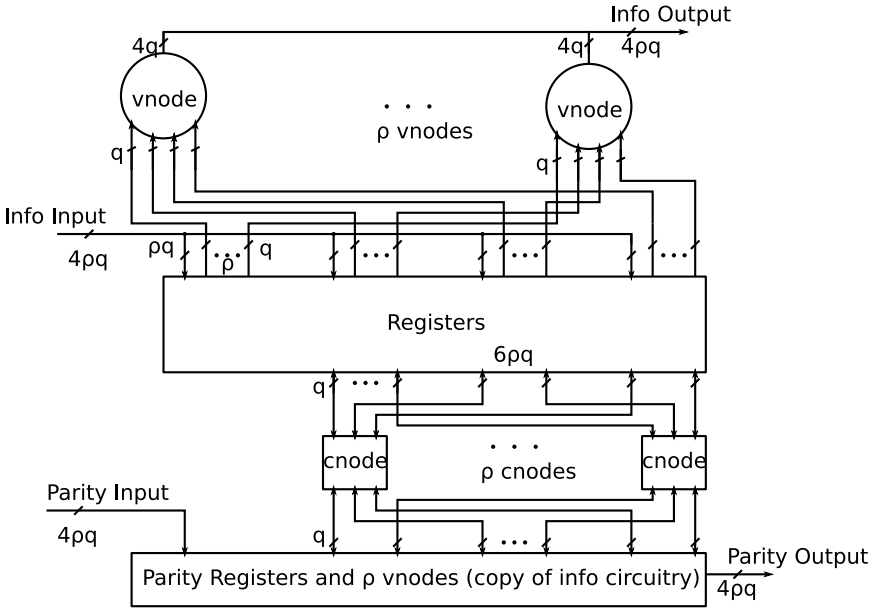

Figure 5.17: Decv3 processor. Decv3 removes the rotation switch-matrix transforming a greater than linear hardware cost relationship to ρ to a less than linear hardware dependence. This design improvement further reduces the area and power consumption.

5.3 Decoder v3 - Removing the Rotation Switch-Matrix (Decv3)

The decv3 architecture removes the rotation switch-matrices present in the decv1 and decv2 architectures. As a result the energy-per-decoded-bit per processor is reduced further by 13.8%. In the two previous architectures decv1 and decv2, as ρ increases linearly, the area of the rotation switch matrix increases greater than linearly. By re-arranging the memory to eliminate the rotation switch matrix, decv3 removes a greater than linear hardware cost dependence on ρ. In the case of the decv3 architecture applied to the T_s=768, ρ=64 code, synthesized for a 100-MHz clock frequency, the decoder processor area is reduced by 10%.

The higher the ρ the higher the throughput and the lower the energy-per-decoded-bit. Figure 5.18 shows the energy-per-bit per processor versus the throughput for all of the PN-LDPC-CCs. Figure 5.19 shows the energy-per-decoded bit versus throughput versus ρ. Each curve in the figure corresponds to a fixed code length. Points along a curve, moving from left-to-right, represent an increasing ρ.

Compared to decv2, removing the rotation switch-matrix in decv3 further reduces the energy-per-decoded-bit. Figure 5.20 compares decv3 to decv2 in terms of energy-per-decoded-bit per processor versus throughput. The energy

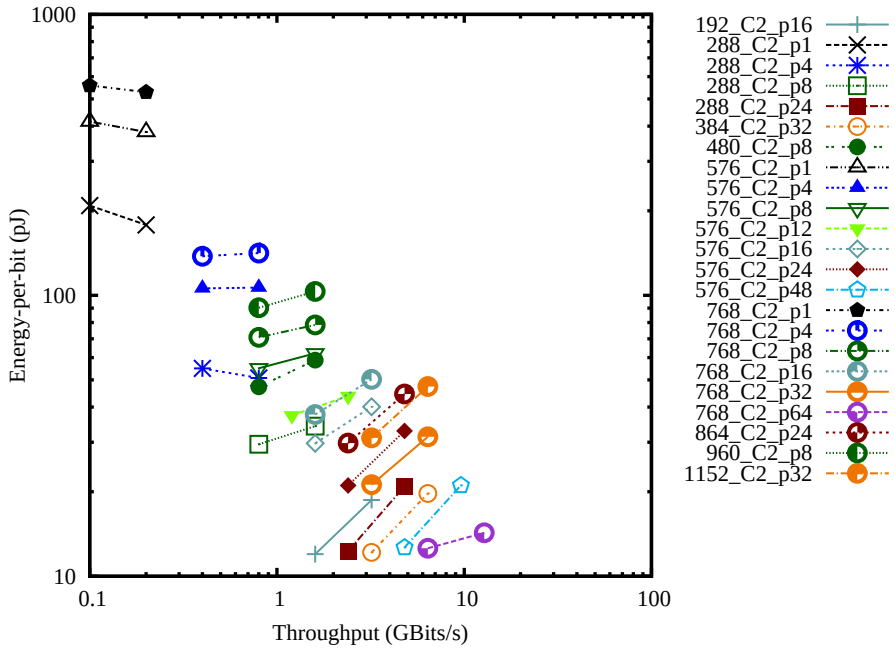

Figure 5.18: Decv3 energy-per-decoded-bit per processor versus throughput for all of the PN-LDPC-CCs. The higher the ρ the higher the throughput and the lower the energy-per-decoded-bit. Clock frequencies of 100 and 200 MHz are used.

saving ranges from a few percent to roughly 50% (T_s=768, ρ=64, 12.8 Gbits/s).

Removing the rotation switch-matrix only reduces the decoder processor area for codes with large values of ρ (i.e., 64). Figure 5.21 compares decv3 to decv2 in terms of area per processor versus throughput. For smaller values of ρ the area may slightly increase (less than 5%). Without the rotation switch-matrix the design can no longer be described hierarchically with memory modules, switch-matrices, check-nodes and variable-nodes. Instead the design must be described in a flattened form with individual registers connecting directly to check-nodes and variable-nodes. These direct connections replace the rotation switch-matrix. As a result, in decv3 the register memories cannot be synthesized independently. In reality the switch matrix structure was purely a figment of our imagination (or perhaps more accurately the lack of our imagination) caused by our desire to impose structure in what is actually unstructured. Unfortunately our synthesis tool has a much harder time optimizing large flat designs that lack hierarchy. The disassembly of the memories and removal of the switch matrices is an architecture change and not a logic optimization. Thus the synthesis tool is unable to make such a change to the design, as the design change crosses decoder control register boundaries.

Figure 5.22 shows the standard cell area versus throughput versus ρ. Each curve in the figure corresponds to a fixed code length. Points along a curve, moving from left-to-right, represent an alternative code with the same T_s but

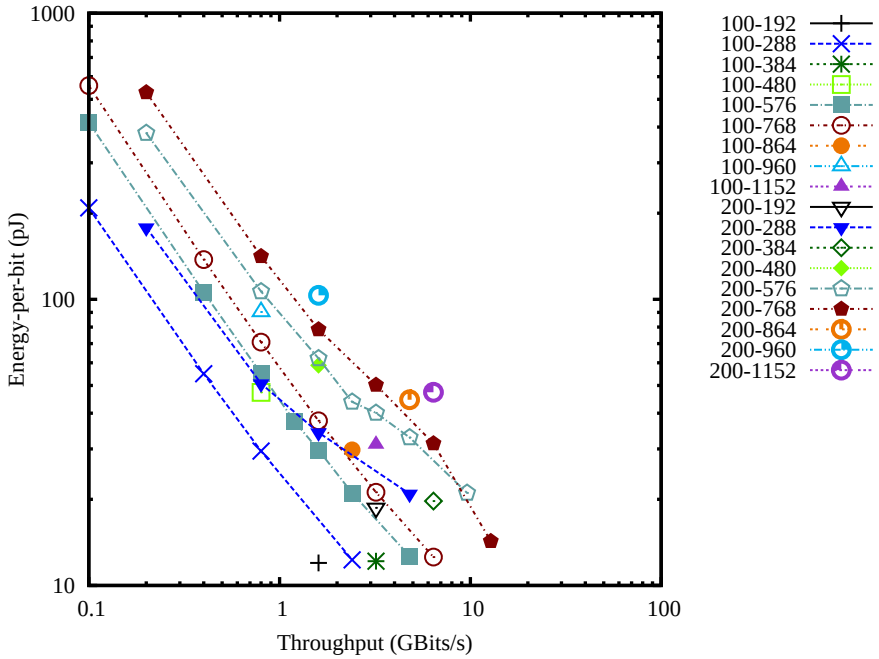

Figure 5.19: Decv3 energy-per-bit versus the information throughput for the PN-LDPC-CCs. The higher the ρ, the higher the information throughput and the lower the energy-per-decoded-bit. The energy-per-decoded-bit is expressed on a per-processor basis. Code lengths (T_s) and parallel-node (ρ) values are as specified: for T_s=192 ρ ∈{16}, for T_s=288 ρ ∈{1,4,8,24}, for T_s=384 ρ ∈{32}, for T_s=480 ρ ∈{8}, for T_s=576 ρ ∈{1,4,8,12,16,24,48}, for T_s=768 ρ ∈{1,4,8,16,32,64}, for T_s=864 ρ ∈{24}, for T_s=960 ρ ∈{8}, for T_s=1152 ρ ∈{32,48}, and for T_s=2304 ρ ∈{64}. Clock frequencies of 100 (prefix 100) and 200 (prefix 200) MHz are used.

an increased ρ.

Area is linearly correlated to T_s. Figure 5.23 compares the area for a decoder processor versus T_s for all of the PN-LDPC-CCs at a clock frequency of 100 MHz. For a given T_s, variations in area correlate with variations in ρ. In the case of the T_s=768 codes, increasing ρ from 1 to 64 increases the area by 9.3%.

In the two previous decoder versions, decv1 and decv2, the rotation switch matrices grow greater than linearly with a linearly increasing ρ. Decv3 removes this dependence. Table 5.2 shows the component areas for decv2 and decv3 for the T_s=768 code with a ρ of 4, 8, 16, 32 and 64, at 100 MHz assuming 5-bit LLRs. One can verify that the decv1 and decv2 switch-matrix areas increase greater than linearly with ρ. In the first decv3 component area analysis, the register memory area is taken to be the same as the decv2 register memory area, as the individual modules have been combined in decv3 and are no longer independently observable. In the second decv3 component area analysis, the register memory area is estimated by taking the average area of the decv2

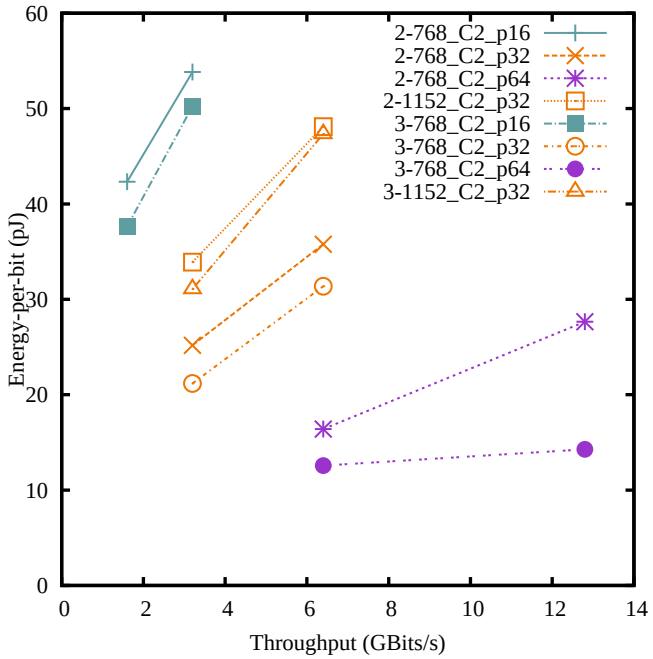

Figure 5.20: Decv3 (prefix 3) and decv2 (prefix 2) energy-per-decoded-bit per processor versus throughput. Removing the switch-matrix results leads to significant power savings. Note the large savings for the T_s=768, ρ=64 code at the higher throughput. This can be attributed to the synthesis tool no longer needing to use larger more power intensive gates in the rotation switch-matrix to meet timing. As will be shown later, the rotation switch-matrix consumes a large amount of area in decv2. Clock frequencies of 100 and 200 MHz are used.

register memory. In both decv3 component analysis, the switch-matrix area is approximated by subtracting the register memory, variable-node and check-node areas from the processor area. Unlike decv1 and decv2, the decv3 area associated with the "rotation switch-matrix" no longer grows greater than linearly with ρ. The increase in decv3 processor area with increasing ρ can now be attributed solely to the increasing variable-node and check-node areas. The rotation switch matrix component has been eliminated, and with it, the greater than linear hardware area dependence on ρ.

There is a nearly linear relationship between T_s' and the energy-per-decoded-bit. Figure 5.24 compares the energy-per-bit versus T_s' for a 100-MHz clock.

The critical path for the T_s=768, ρ=64 code running at 200 MHz starts at a phase control flip-flop, flows through a variable-node, and exits at the output of the processor. The delay from the activation of the phase control flip-flop to the variable-node is 1.59 ns. From the input of the variable-node to its output the delay is 2.86 ns. As we specified, the output delay from the processor is 0.5 ns.

In summary, eliminating the rotation switch-matrix component reduces the

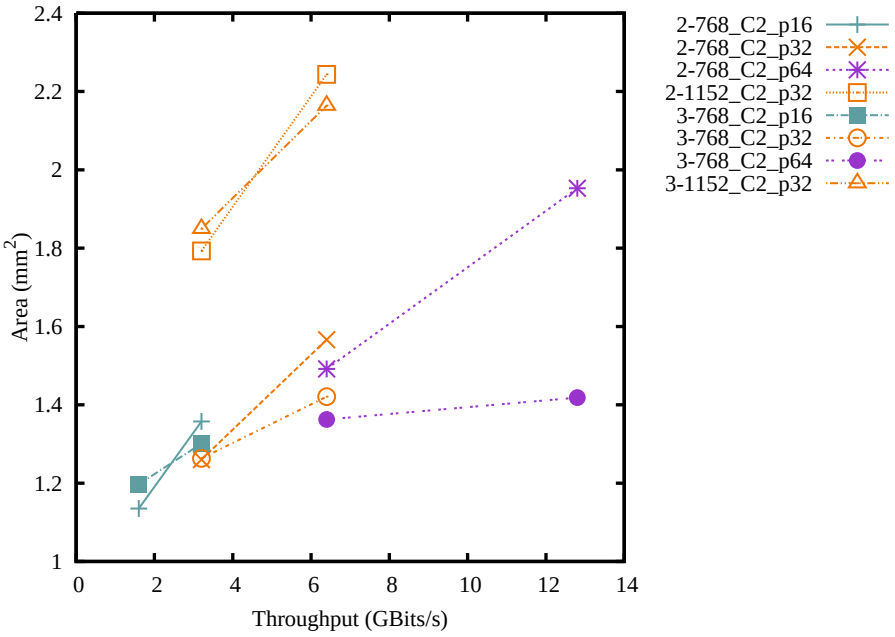

Figure 5.21: Decv3, decv2 area per processor versus throughput. Removing the rotation switch-matrix saves area for those codes with a large value of ρ (i.e. 64). Clock frequencies of 100 and 200 MHz are used.

power consumption by 13.8% on average (min -5.6%, max 39.1%, σ=10.4%) and reduces the area by 1.9% on average (min -5.9%, max 19.3%, σ=4.9%).

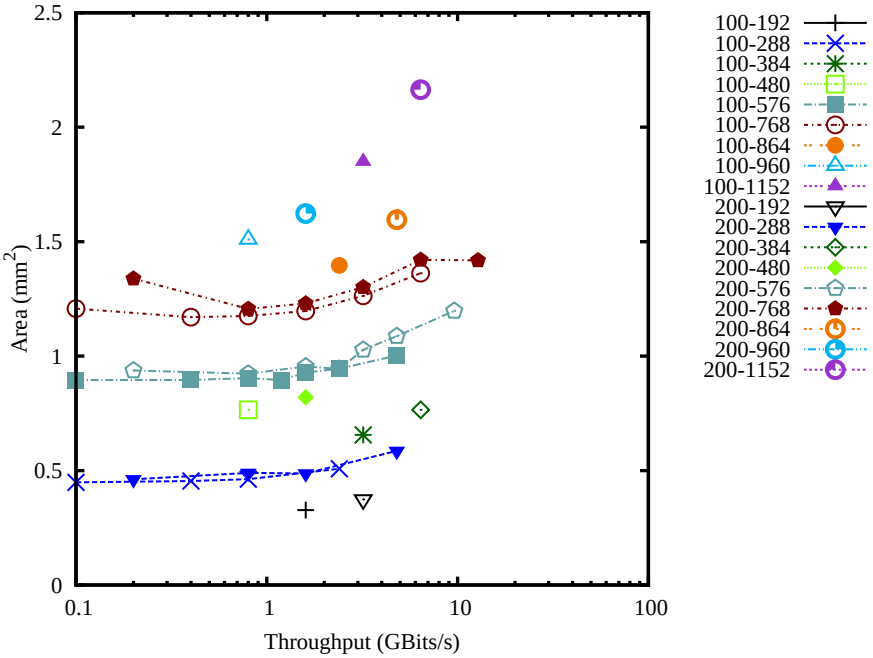

Figure 5.22: Decv3 area for one processor versus the information throughput, for all of the PN-LDPC-CCs. The area is expressed on a per-processor basis. Code lengths (T_s) and parallel-node (ρ) values are as specified: for T_s=192 $\rho \in \{16\}$, for T_s=288 $\rho \in \{1,4,8,24\}$, for T_s=384 $\rho \in \{32\}$, for T_s=480 $\rho \in \{8\}$, for T_s=576 $\rho \in \{1,4,8,12,16,24,48\}$, for T_s=768 $\rho \in \{1,4,8,16,32,64\}$, for T_s=864 $\rho \in \{24\}$, for T_s=960 $\rho \in \{8\}$, for T_s=1152 $\rho \in \{32,48\}$, and for T_s=2304 $\rho \in \{64\}$. Clock frequencies of 100 (prefix 100) and 200 (prefix 200) MHz are used.

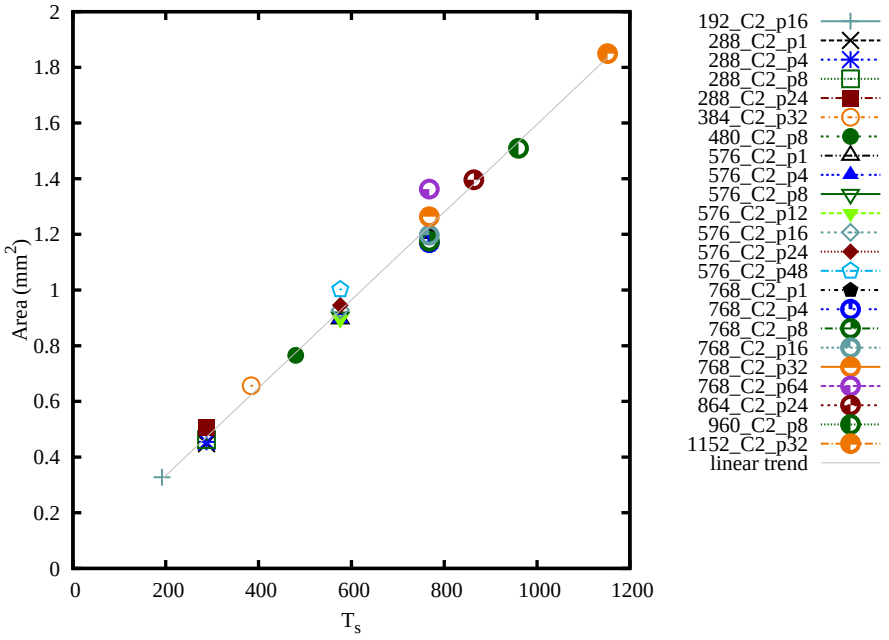

Figure 5.23: Decv3 processor area versus T_s at 100 MHz. Note the strong linear correlation between T_s and the area. For a given T_s, variations in area correspond to variations in ρ. In the case of the T_s=768 codes, increasing ρ from 1 to 64 increases the area by 9.3%.

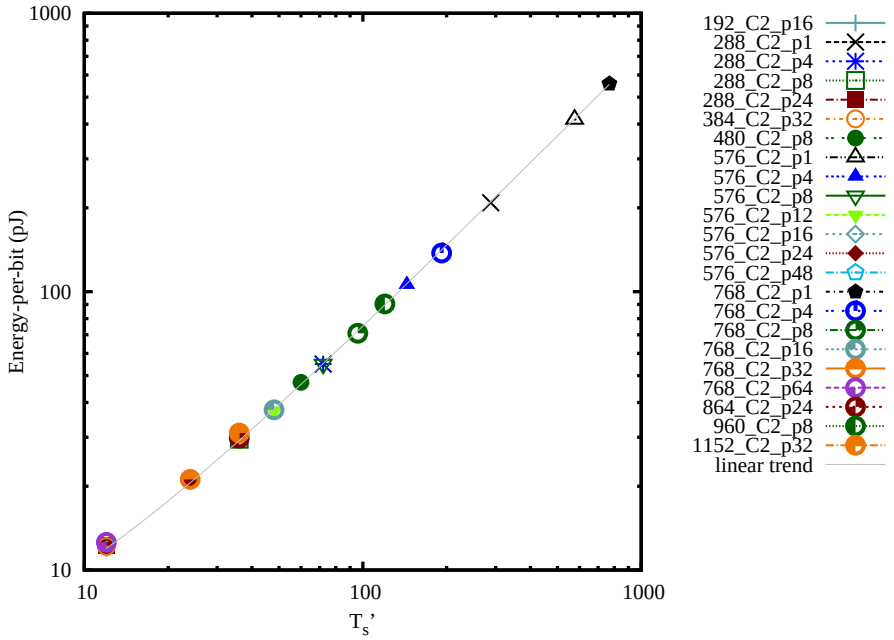

Figure 5.24: Decv3 energy-per-bit versus T_s' for a 100-MHz clock. The energy-per-decoded-bit appears linearly proportional to T_s'.

Code T_s=768	Variable nodes (mm^2)	Check nodes (mm^2)	Switch matrix (mm^2)	Register memory (mm^2)	Processor (mm^2)
decv1 ρ=1	0.003 (0%)	0.001 (0%)	0.000 (0%)	1.373 (98%)	1.402 (100%)
decv1 ρ=4	0.011 (1%)	0.005 (0%)	0.006 (0%)	1.278 (98%)	1.301 (100%)
decv1 ρ=8	0.020 (2%)	0.011 (1%)	0.015 (1%)	1.190 (96%)	1.237 (100%)
decv1 ρ=16	0.041 (3%)	0.022 (2%)	0.040 (3%)	1.182 (92%)	1.285 (100%)
decv1 ρ=32	0.081 (6%)	0.044 (3%)	0.104 (7%)	1.184 (84%)	1.414 (100%)
decv1 ρ=64	0.162(10%)	0.089 (5%)	0.249(15%)	1.170 (70%)	1.669 (100%)
decv2 ρ=1	0.002 (0%)	0.001 (0%)	0.000 (0%)	1.169 (98%)	1.197 (100%)
decv2 ρ=4	0.009 (1%)	0.004 (0%)	0.005 (0%)	1.103 (98%)	1.121 (100%)
decv2 ρ=8	0.017 (2%)	0.008 (1%)	0.013 (1%)	1.034 (97%)	1.072 (100%)
decv2 ρ=16	0.035 (3%)	0.015 (1%)	0.034 (3%)	1.018 (92%)	1.102 (100%)
decv2 ρ=32	0.070 (6%)	0.031 (3%)	0.084 (7%)	1.022 (85%)	1.207 (100%)
decv2 ρ=64	0.140(10%)	0.062 (4%)	0.207(15%)	0.994 (71%)	1.403 (100%)
decv3 ρ=1	0.002 (0%)	0.001 (0%)	0.006[2] (1%)	1.169[1]	1.175 (100%)
decv3 ρ=4	0.009 (1%)	0.004 (0%)	0.036[2] (3%)	1.103[1]	1.145 (100%)
decv3 ρ=8	0.017 (2%)	0.008 (1%)	0.103[2] (9%)	1.034[1]	1.155 (100%)
decv3 ρ=16	0.035 (3%)	0.016 (1%)	0.108[2] (9%)	1.018[1]	1.176 (100%)
decv3 ρ=32	0.070 (6%)	0.031 (3%)	0.095[2] (8%)	1.022[1]	1.218 (100%)
decv3 ρ=64	0.140(11%)	0.062 (5%)	0.122[2] (9%)	0.994[1]	1.318 (100%)
decv3 ρ=1	0.002 (0%)	0.001 (0%)	0.134[2] (12%)	1.034[3]	1.175 (100%)
decv3 ρ=4	0.009 (1%)	0.004 (0%)	0.099[2] (9%)	1.034[3]	1.145 (100%)
decv3 ρ=8	0.017 (2%)	0.008 (1%)	0.095[2] (8%)	1.034[3]	1.155 (100%)
decv3 ρ=16	0.035 (3%)	0.016 (1%)	0.092[2] (8%)	1.034[3]	1.176 (100%)
decv3 ρ=32	0.070 (6%)	0.031 (3%)	0.083[2] (7%)	1.034[3]	1.218 (100%)
decv3 ρ=64	0.140(11%)	0.062 (5%)	0.082[2] (6%)	1.034[3]	1.318 (100%)

Table 5.2: Decv2 and decv3 comparison of component standard cell areas at 100 MHz with 5-bit LLRs. For decv1 and decv2 the area of the switch matrix increases greater than linearly with ρ.
[1]In decv3, the register memory is estimated by taking the area of the decv2 register memory.
[2]In decv3, the switch matrix area is approximated by subtracting the register memory, variable-node and check-node area from the processor area.
[3]In this case the memory area is fixed based on an average of the decv2 memory area. The decv3 switch matrix area no longer grows greater than linearly with ρ. The increase in decv3 processor area with increasing ρ can be attributed to the increasing variable-node and check-node areas.

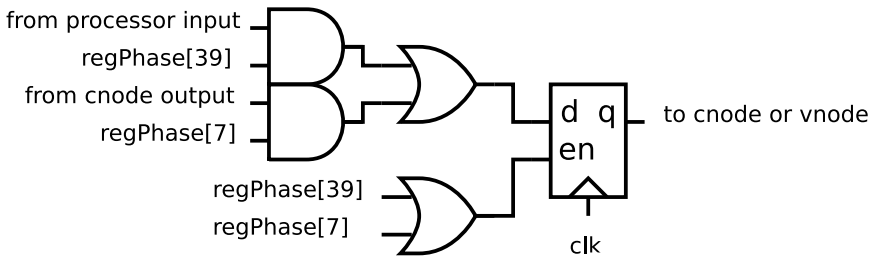

Figure 5.25: Decv4 with clock-gating. Decv4 uses clock-gating to significantly reduce the power consumption.

5.4 Decoder v4 - Clock Gating (Decv4)

Decv4 uses clock-gating to reduce the power consumption and processor area. Clock-gating is more effective at reducing the power consumption for those codes with a high T_s', The drop in power consumption is proportional to the reduction in the switching activity of the registers, which is inversely proportional to T_s'. Each register is written and read at most twice per T_s'. Figure 5.25 shows an example clock-gated register and how it is clock-gated by the phase. When the synthesis tool is looking to gate the clock and it meets the structure shown in Figure 5.25, it converts the structure into a clock-gated structure as was described in Section 4.4.

The energy-per-decoded-bit versus throughput trend remains the same as in the previous versions of the decoder. Figure 5.26 shows the energy-per-bit per processor versus the throughput for all the PN-LDPC-CCs. Figure 5.27 shows the energy-per-decoded bit versus throughput versus ρ. Each curve in the figure corresponds to a fixed code length. Points along a curve, moving from left-to-right, represent an increasing ρ. The higher the ρ the higher the throughput and the lower the energy-per-decoded-bit.

Clock-gating can reduce the energy-per-decoded-bit by 39.8%. Figure 5.28 compares decv4 and decv3 in terms of the energy-per-decoded-bit per processor versus the throughput. In the case of the T_s=768, ρ=64 code operating at 200 MHz, the power consumption is slightly higher than the non-clock gated version. However, in all other examined cases, clock-gating significantly reduces the power consumption.

Clock gating has the unexpected and fortuitous side-effect of reducing the area. Figure 5.29 compares decv4 and decv3 in terms of area per processor versus throughput. Given that clock gating requires more logic, the reduction in area is unexpected and may be attributed to a slackening of timing along the critical paths. As the gated clock arrives after the original clock, there is a slightly longer clock cycle in the transition from the non-clock gated circuitry to the clock gated circuitry. Thus there is slightly more time available for logic evaluation. Conversely, the transition from the clock-gated circuitry to

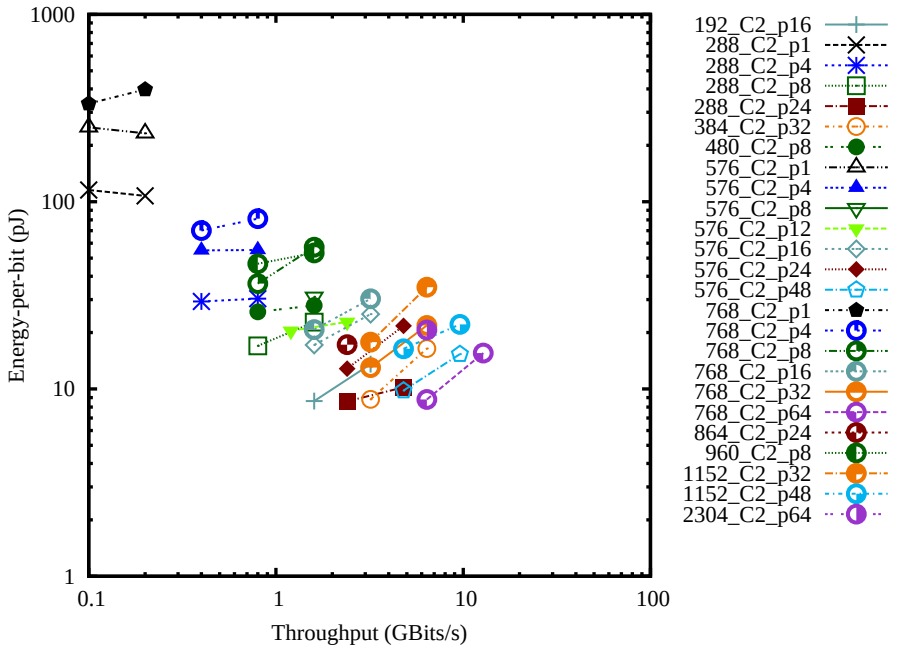

Figure 5.26: Decv4 energy-per-decoded-bit per processor versus the throughput for all of the PN-LDPC-CCs. The higher the ρ the higher the throughput and the lower the energy-per-decoded-bit. Clock-gating further reduces the energy-per-decoded-bit. Clock frequencies of 100 and 200 MHz are used.

the non-clock gated circuitry has a slightly shorter clock period. However, the implemented logic transitioning from the clock-gate circuitry is simpler than that of the logic entering the clock-gated circuitry. As a result, the timing on the critical paths is eased, resulting in smaller and/or fewer gates and thus less area. For those cases examined, the reduction in area is significant, 13.5% on average.

Required area is primarily determined by T_s and secondary factors include: clock frequency and ρ. Figure 5.30 shows the standard cell area versus throughput versus ρ. Each curve in the figure corresponds to a fixed code length. Points along a curve, moving from left-to-right, represent an alternative code with the same T_s but an increased ρ.

Similar to the previous decoders, decv4 has a strong linear correlation between T_s and the area. For a given T_s, variations in area correspond to variations in ρ. Figure 5.31 compares the area for a processor versus T_s for all of the PN-LDPC-CCs at a clock frequency of 100 MHz. In the case of the T_s=768 codes, increasing ρ from 1 to 64 increases the area by 17.9%.

Again, similar to previous decoders, there is a nearly linear relationship between T_s' and the energy-per-decoded-bit. Figure 5.32 compares the energy-per-bit versus T_s' for a 100-MHz clock. The relationship between the energy-per-decoded-bit and T_s' is no longer purely linear as those designs with larger

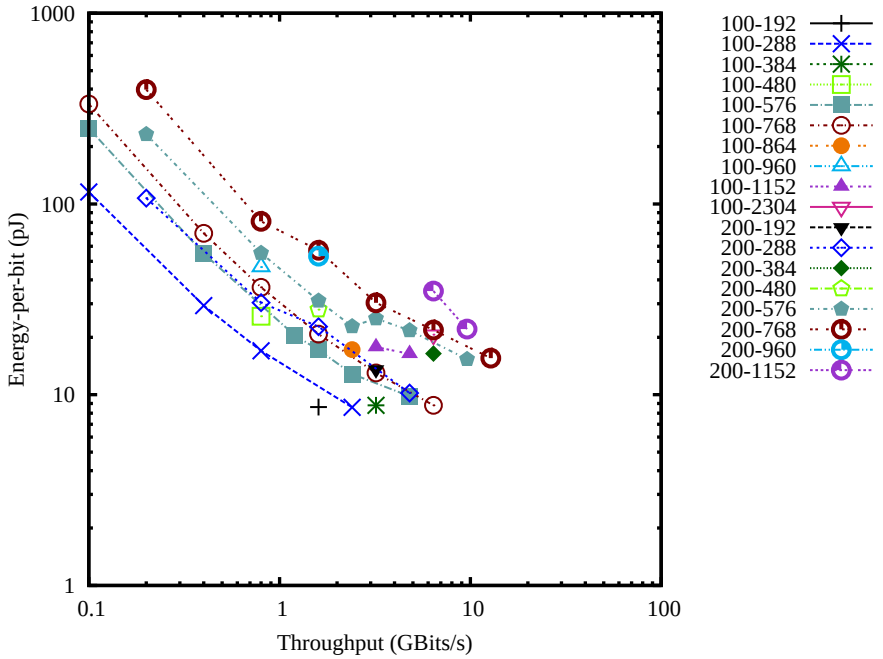

Figure 5.27: Decv4 energy-per-bit versus the information throughput for the PN-LDPC-CCs. The higher the ρ, the higher the information throughput and the lower the energy-per-decoded-bit. The energy-per-decoded-bit is expressed on a per-processor basis. Code lengths (T_s) and parallel-node (ρ) values are as specified: for T_s=192 $\rho \in \{16\}$, for T_s=288 $\rho \in \{1,4,8,24\}$, for T_s=384 $\rho \in \{32\}$, for T_s=480 $\rho \in \{8\}$, for T_s=576 $\rho \in \{1,4,8,12,16,24,48\}$, for T_s=768 $\rho \in \{1,4,8,16,32,64\}$, for T_s=864 $\rho \in \{24\}$, for T_s=960 $\rho \in \{8\}$, for T_s=1152 $\rho \in \{32,48\}$, and for T_s=2304 $\rho \in \{64\}$. Clock frequencies of 100 (prefix 100) and 200 (prefix 200) MHz are used.

values of T_s' experience a greater reduction in the energy-per-decoded-bit due to clock-gating.

The critical path for the T_s=768, ρ=64 code running at 200 MHz, starts at a phase control flip-flop and then flows through a variable-node before exiting out of the processor. From the activation of the phase control flip-flop to the variable-node the delay is 1.60 ns. From the input of the variable-node to its output the delay is 2.85 ns. Finally, from the output delay is specified to be 0.5 ns.

In summary, clock-gating reduces the decoder area by 13.5% on average (min 7.6%, max 20.1%, σ=2.7%) and the energy-per-decoded-bit by 39.8% on average (min 9.3%, max 53.6%, σ=10.4%). Thus we recommend using clock-gating in the decoder.

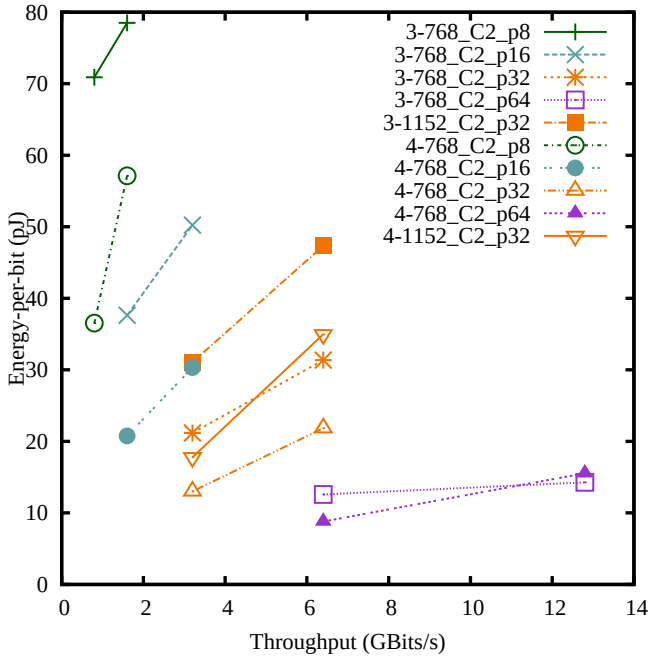

Figure 5.28: Decv4 (prefix 4) and decv3 (prefix 3) energy-per-decoded-bit per processor versus the throughput for various codes. Note for the T_s=768, ρ=8 code operating at 100 MHz, clock-gating reduces the energy-per-bit by roughly 50%. Clock frequencies of 100 and 200 MHz are used.

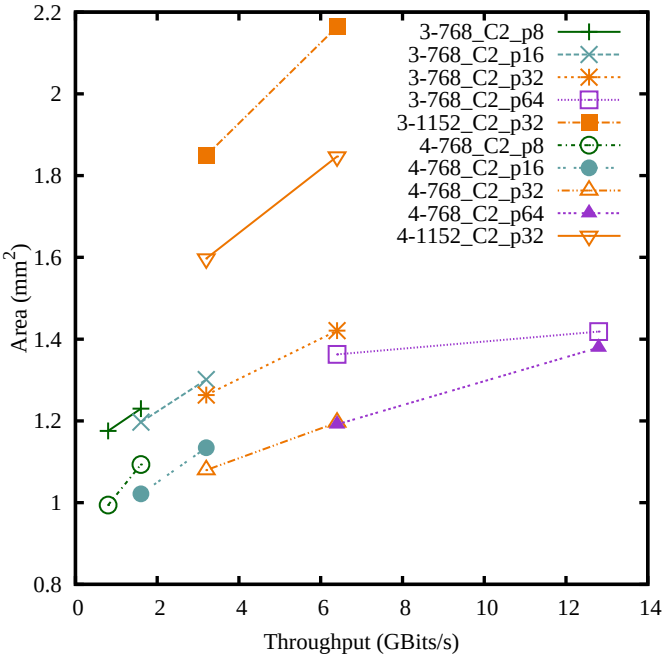

Figure 5.29: Decv4, Decv3, area per processor versus throughput for various codes. Note that the typical reduction in area is 13.5%. Clock frequencies of 100 and 200 MHz are used.

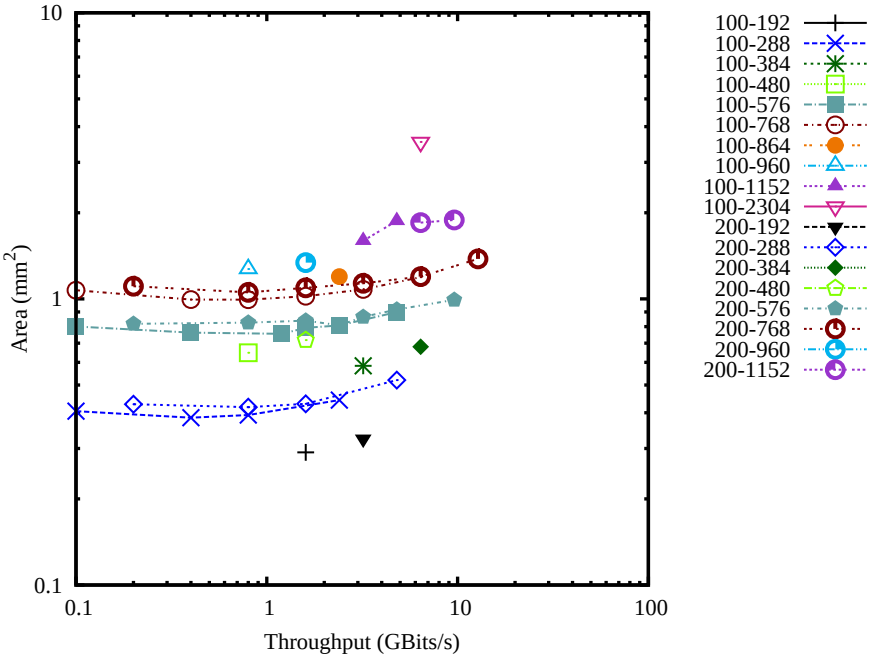

Figure 5.30: Decv4 area for one processor versus the information throughput, for all of the PN-LDPC-CCs. The area is expressed on a per-processor basis. Code lengths (T_s) and parallel-node (ρ) values are as specified: for T_s=192 $\rho \in \{16\}$, for T_s=288 $\rho \in \{1,4,8,24\}$, for T_s=384 $\rho \in \{32\}$, for T_s=480 $\rho \in \{8\}$, for T_s=576 $\rho \in \{1,4,8,12,16,24,48\}$, for T_s=768 $\rho \in \{1,4,8,16,32,64\}$, for T_s=864 $\rho \in \{24\}$, for T_s=960 $\rho \in \{8\}$, for T_s=1152 $\rho \in \{32,48\}$, and for T_s=2304 $\rho \in \{64\}$. Clock frequencies of 100 (prefix 100) and 200 (prefix 200) MHz are used.

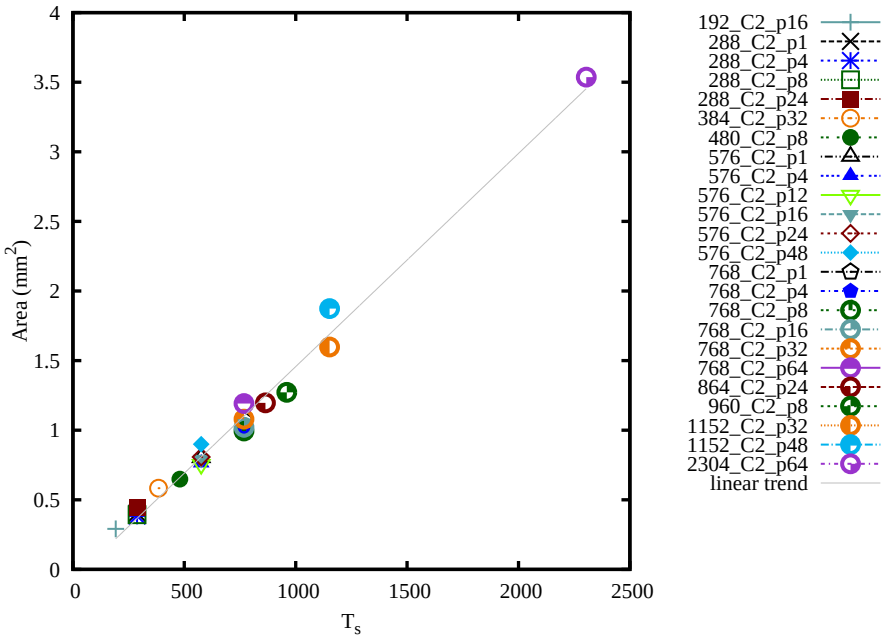

Figure 5.31: Decv4 processor area versus T_s at 100 MHz. Note the strong linear correlation between T_s and the area. For a given T_s, variations in area correspond to variations in ρ. In the case of the T_s=768 codes, increasing ρ from 4 to 64 increases the area by 17.9%.

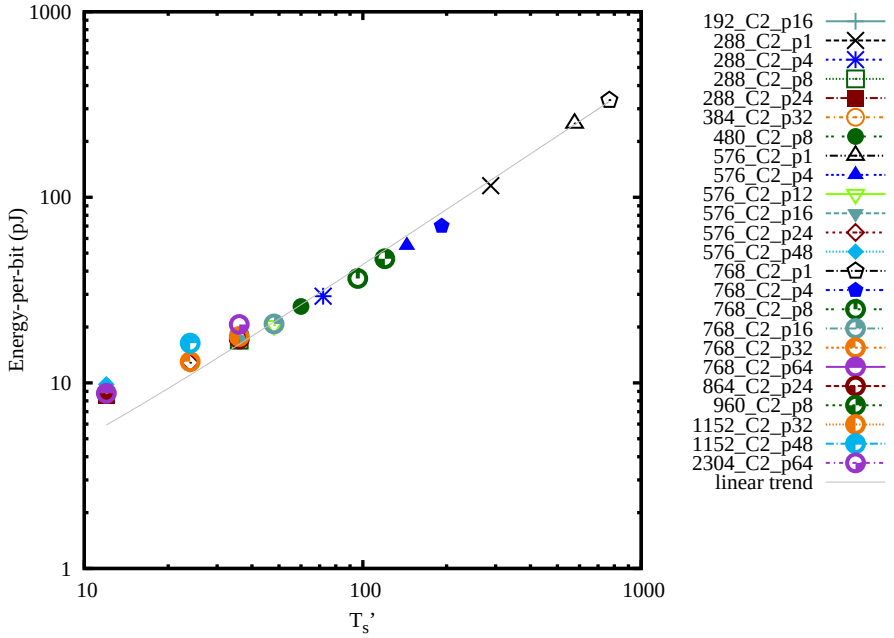

Figure 5.32: Decv4 energy-per-bit versus T_s' for a 100-MHz clock. There is a strong positive correlation between the energy-per-decoded-bit and T_s'. However, the relationship to T_s' is no longer purely linear as those designs with larger values of T_s' experience a greater reduction in the energy-per-decoded-bit due to clock-gating.

5.5 Decoder v5 - Removing Reset Circuitry (Decv5)

Decv5 removes the decoder processor reset circuitry to further reduce the area. Typically, the reset circuitry runs over the entire decoder processor. In the case of short data stream segments, the reset circuitry would be activated often. In previous decoders, the decoder must finish processing the present data stream plus termination bits or padding bits before it can begin processing the next data stream. Reset circuitry also adds to the size of the decoder. Flip-flops with resets are larger than those without resets. Memories do not have the ability to reset their entire contents in a single clock-cycle. In fact, memories need to be addressed at every row and written with the reset values.

LDPC-CC decoders are deterministic in that the order of operations depends on the code and not the data. With this knowledge it is possible to pre-determine which registers hold uninitialized values. Knowing which registers are uninitialized, at any given point in time, allows us to multiplex in the maximum LLR magnitude into the check-node rather than allowing uninitialized LLR data, from uninitialized memory locations, to enter the check-node. Using this method, we no longer need to explicitly initialize or "reset" the decoder memory. The controller circuitry required to multiplex in the maximum magnitude into the check-node is simple. Each input into the check-node is held at the maximum magnitude until a specific phase, which is determined in advance by analyzing the code, is reached. Once this predetermined phase is reached, then the values from memory are used. All inputs to the check-nodes begin to use computed LLR values from memory within T_s' phases.

In the case of decv5, multiplexors are only placed at the check-node inputs; the datapaths from the check-nodes and to/from the variable-nodes remain unchanged. The result is that the BER performance is expected to fall between that of the "maximum magnitude" decoder implementations and the "saturation-bit" implementations. If multiplexors are used at the inputs to the check-nodes and the outputs of the variable-nodes, then the same BER performance as the "saturation-bit" implementations can be achieved (see Section F.1 for more details).

The use of a small amount of extra control circuitry in decv5 eliminates the need for the memory initialization/reset circuitry and thus reduces the energy-per-decoded-bit. Figure 5.33 shows the energy-per-bit per processor versus throughput for all of the PN-LDPC-CCs. Figure 5.34 shows the energy-per-decoded bit versus throughput versus ρ. Each curve in the figure corresponds to a fixed code length. Points along a curve, moving from left-to-right, represent an increasing ρ. The higher the ρ the higher the throughput and the lower the energy-per-decoded-bit.

Eliminating the reset circuitry has little impact on the energy-per-decoded-bit. Figure 5.35 compares decv5 and decv4 in terms of the energy-per-decoded-bit per processor versus the throughput for various PN-LDPC-CCs. The dy-

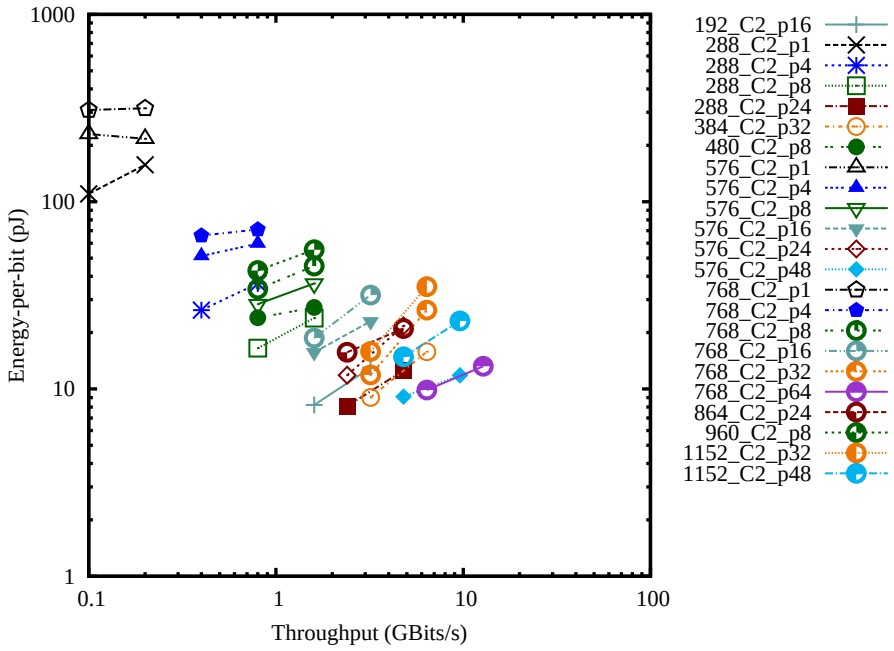

Figure 5.33: Decv5 energy-per-decoded-bit per processor versus throughput for all of the PN-LDPC-CCs. The higher the ρ the higher the throughput and the lower the energy-per-decoded-bit. Clock frequencies of 100 and 200 MHz are used.

namic power consumption associated with the reset circuitry is amortized over the data stream length. With a stream length of 32,000 bits, the reduction in the dynamic power consumption due to the removal of the reset circuitry is not noticeable. Any observable reduction in the power consumption can be attributed to a reduction in the static power consumption.

Eliminating the reset circuitry reduces the decoder processor area. Figure 5.36 compares encv5 and encv4, for various codes, with respect to their area per processor versus throughput. Note that the reduction in area is 11.6% on average.

Required area is primarily determined by T_s and secondary factors include: clock frequency and ρ. Figure 5.37 shows the standard cell area versus throughput versus ρ. Each curve in the figure corresponds to a fixed code length. Points along a curve, moving from left-to-right, represent an alternative code with the same T_s but an increased ρ.

Again, similar to previous decoders, there is a nearly linear relationship between T_s' and the energy-per-decoded-bit. Figure 5.38 compares the decv5 processor energy-per-bit versus T_s' for a 100-MHz clock. Due to the clock-gating introduced in decv4, the relationship between the energy-per-decoded-bit and T_s' is non-linear. Removing the reset circuitry has no perceivable impact on this relationship.

Removing the reset circuitry has little impact on the energy-per-decoded-

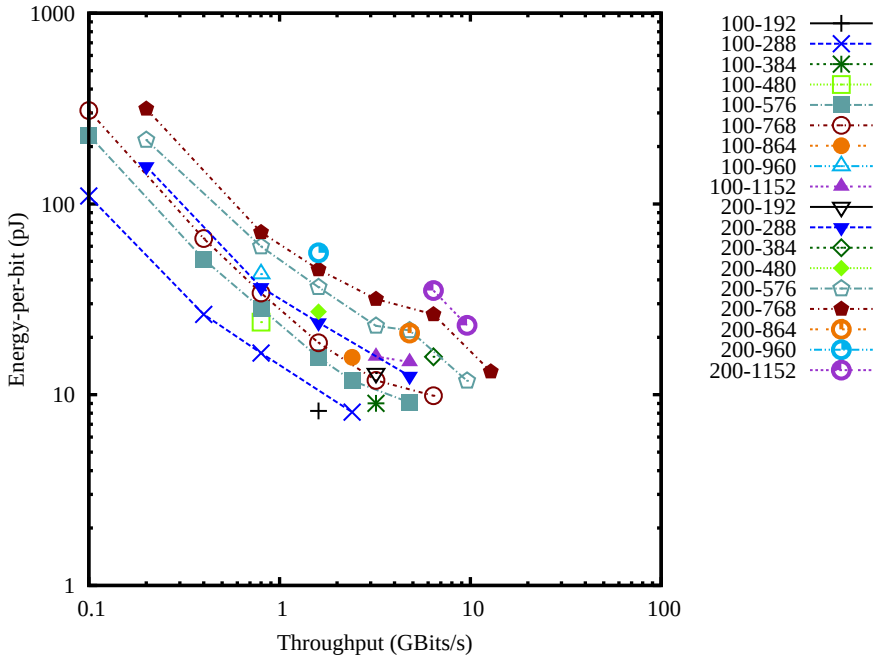

Figure 5.34: Decv5 energy-per-bit versus the information throughput for the PN-LDPC-CCs. The higher the ρ, the higher the information throughput and the lower the energy-per-decoded-bit. The energy-per-decoded-bit is expressed on a per-processor basis. Code lengths (T_s) and parallel-node (ρ) values are as specified: for T_s=192 $\rho \in \{16\}$, for T_s=288 $\rho \in \{1,4,8,24\}$, for T_s=384 $\rho \in \{32\}$, for T_s=480 $\rho \in \{8\}$, for T_s=576 $\rho \in \{1,4,8,12,16,24,48\}$, for T_s=768 $\rho \in \{1,4,8,16,32,64\}$, for T_s=864 $\rho \in \{24\}$, for T_s=960 $\rho \in \{8\}$, for T_s=1152 $\rho \in \{32,48\}$, and for T_s=2304 $\rho \in \{64\}$. Clock frequencies of 100 (prefix 100) and 200 (prefix 200) MHz are used.

bit (reduction of 2.4% on average; min -20.0%; max 16.8%, σ=8.7%) and reduces the area by roughly 11.5% on average (min 0.7%, max 17.6%, σ=3.6%). Thus we recommend removing the reset circuitry on a case by case basis.

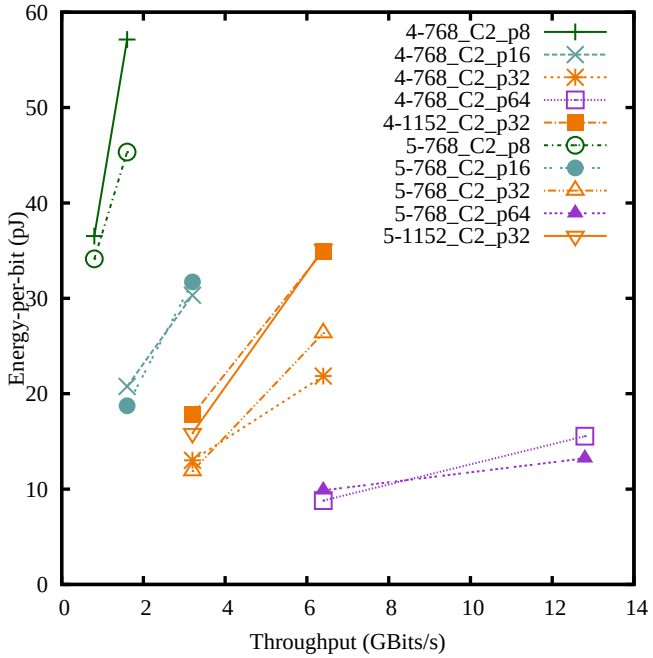

Figure 5.35: Decv5 (prefix 5) and decv4 (prefix 4) energy-per-decoded-bit per processor versus throughput for various PN-LDPC-CCs. Removing the reset circuitry has little impact on the power consumption. Clock frequencies of 100 and 200 MHz are used.

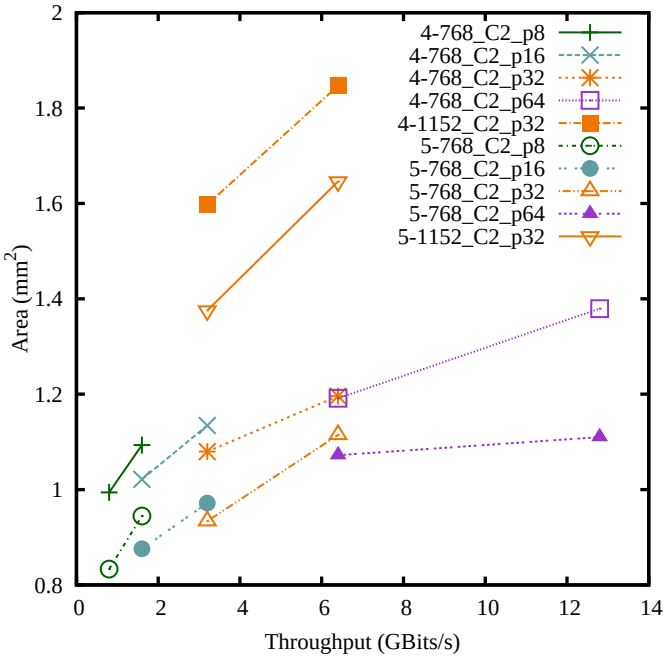

Figure 5.36: Decv5 (prefix 5) and decv4 (prefix 4) area per processor versus throughput for various PN-LDPC-CCs. For every PN-LDPC-CC compared, eliminating the reset circuitry reduces the area.

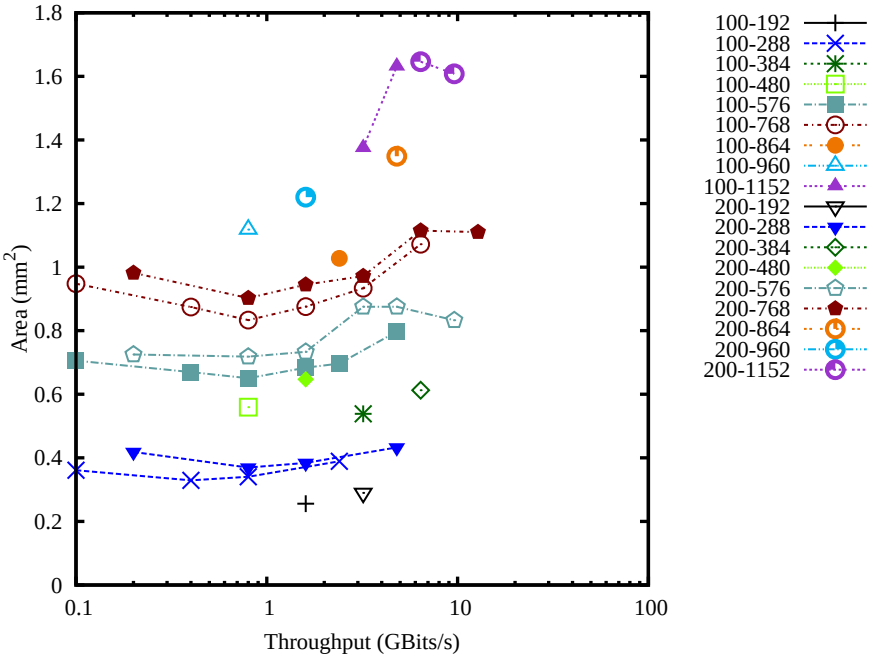

Figure 5.37: Decv5 area for one processor versus the information throughput, for all of the PN-LDPC-CCs. The area is expressed on a per-processor basis. Code lengths (T_s) and parallel-node (ρ) values are as specified: for T_s=192 $\rho \in \{16\}$, for T_s=288 $\rho \in \{1,4,8,24\}$, for T_s=384 $\rho \in \{32\}$, for T_s=480 $\rho \in \{8\}$, for T_s=576 $\rho \in \{1,4,8,12,16,24,48\}$, for T_s=768 $\rho \in \{1,4,8,16,32,64\}$, for T_s=864 $\rho \in \{24\}$, for T_s=960 $\rho \in \{8\}$, for T_s=1152 $\rho \in \{32,48\}$, and for T_s=2304 $\rho \in \{64\}$. Clock frequencies of 100 (prefix 100) and 200 (prefix 200) MHz are used.

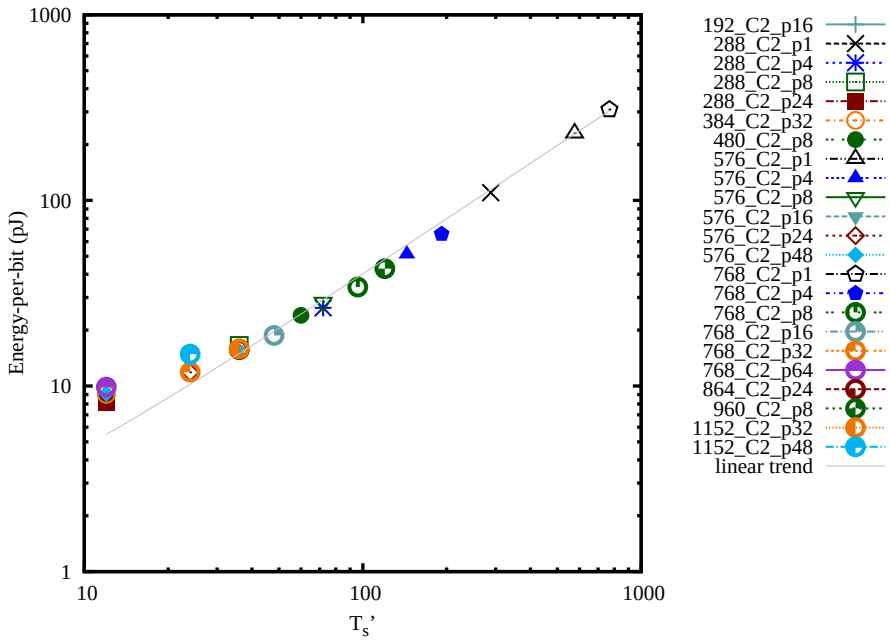

Figure 5.38: Decv5 energy-per-bit versus T_s' for a 100-MHz clock. There is a strong positive correlation between T_s' and the energy-per-decoded-bit.

5.6 Decoder v6 - Truncated Min-Sum Check-Node (Decv6)

Removing the LSB in the check-node operation improves the BER performance[1], reduces the area and reduces the power consumption. We will call this technique the truncated min-sum (TMS) check sum operation. The effect of removing the LSB in the check-node operation ripples though the hardware requirements for the rest of the decoder processor. The LLR bit-width of the storage elements for all the LLRs, with the exception of the channel samples, can be reduced by 1 bit. The variable-node implementation is simplified as three of the four input LLRs have their bit-width reduced by 1 bit. The check-node implementation is simplified as all of the LLR inputs have their bit-width reduced by 1 bit. The result of these simplifications is a significant reduction in the power consumption and the area, and an improvement in the BER performance.

Using a simpler check-node, as in decv6, reduces the power consumption. Figure 5.39 shows the energy-per-bit per processor versus the information throughput for all of the PN-LDPC-CCs. Figure 5.40 shows the energy-per-decoded bit versus throughput versus ρ. Each curve in the figure corresponds to a fixed code length. Points along a curve, moving from left-to-right, represent an increasing ρ. As ρ increases, the throughput increases and the energy-per-decoded-bit decreases.

The benefit of the truncated min-sum operation can be viewed from the point of view of the check-node or the variable-node. From the point of view of the check-node, the truncated min-sum can be viewed as equivalent to sometimes subtracting a constant from the check-node LLR magnitudes. Similar to the offset min-sum operation described in [15], the truncated min-sum operation also subtracts a constant. However, unlike the offset min-sum operation, the truncated min-sum conditionally subtracts a constant value, which is equal to that of the maximum value that the LSB can represent. The truncated min-sum only subtracts the constant value when the LLR LSB is set. In other words, if the LSB is set, it is subtracted. In effect, the LSBs of the LLRs entering the check-node are truncated and do not participate in the check-node operation. As a result, from the point of view of the variable-node, the truncated min-sum operation can be viewed as a normal check-node min-sum operation with LLRs missing their LSB bits. Only the un-altered channel LLRs have full LLR bit precision.

Interestingly, the truncated min-sum technique can actually improve the variable-node operation. In the variable-node operation the higher precision of the un-altered-channel-LLRs helps to preserve, in certain cases, the resulting

[1]The author has been led to believe that truncating the LSB in the check-node operation has been published before; however, the author has not been able to find the work, yet.

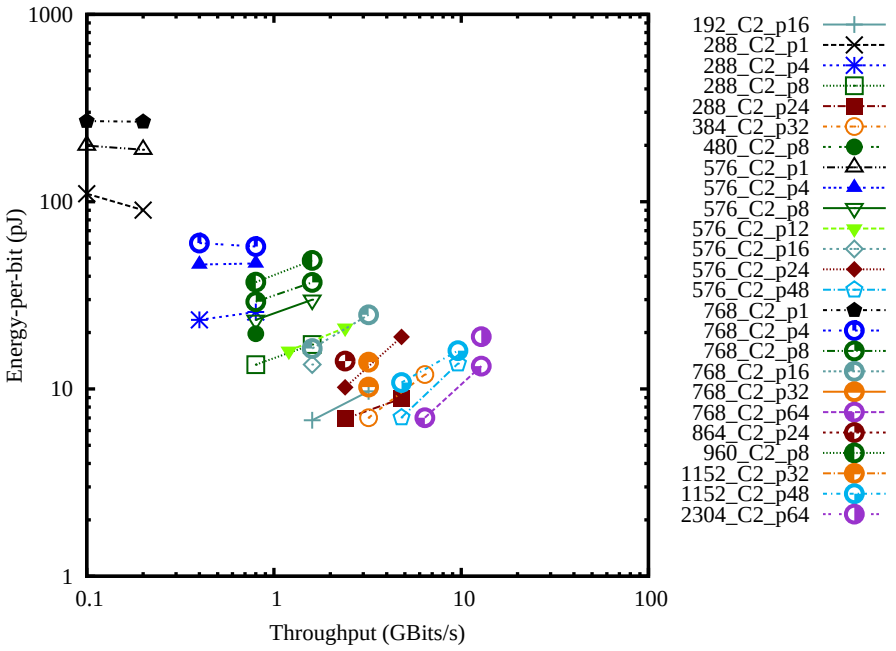

Figure 5.39: Decv6 energy-per-decoded-bit per processor versus throughput for all the PN-LDPC-CCs. The higher the ρ the higher the throughput and the lower the energy-per-decoded-bit. Clock frequencies of 100 and 200 MHz are used.

sign bit. In the cases where the addition of the LLRs in the variable-node would have resulted in a value of zero, a higher precision un-altered-channel-LLR allows the variable-node operation to potentially produce a result of negative zero (from the perspective of the next check-node).

For example, given an LLR with a value of -0.375, represented in binary sign-magnitude as 1011, where the MSB is the sign and there are 3-bits of magnitude. In the TMS operation, the value entering the cnode would be truncated to 101. In the variable-node operation, if we where to add 0.25 to -0.375, the result would be -0.125, or 1001 in binary sign-magnitude. When the resulting value of the variable-node operation enters the TMS check-node operation, its LSB is dropped (truncated) and the binary sign-magnitude value is transformed from 1001 (-0.125) to 100 (-0.000). Note how the sign is preserved. Even though the magnitude is now 0, the sign bit contributes to the outcome of the min-sum check-node operation. Thus we have, in this particular case, preserved the effect of the truncated LSB through the preservation of the sign bit.

The truncated min-sum improves the BER performance by as much as 0.2 dB. For the T_s=288, ρ=24 code and a 4-bit LLR bit-width, Figure 5.41 compares the effect on BER performance of the truncated min-sum operation. For 24 processors the truncated min-sum operation improves the BER performance by 0.2 dB. For 10 processors, the BER performance is improved by

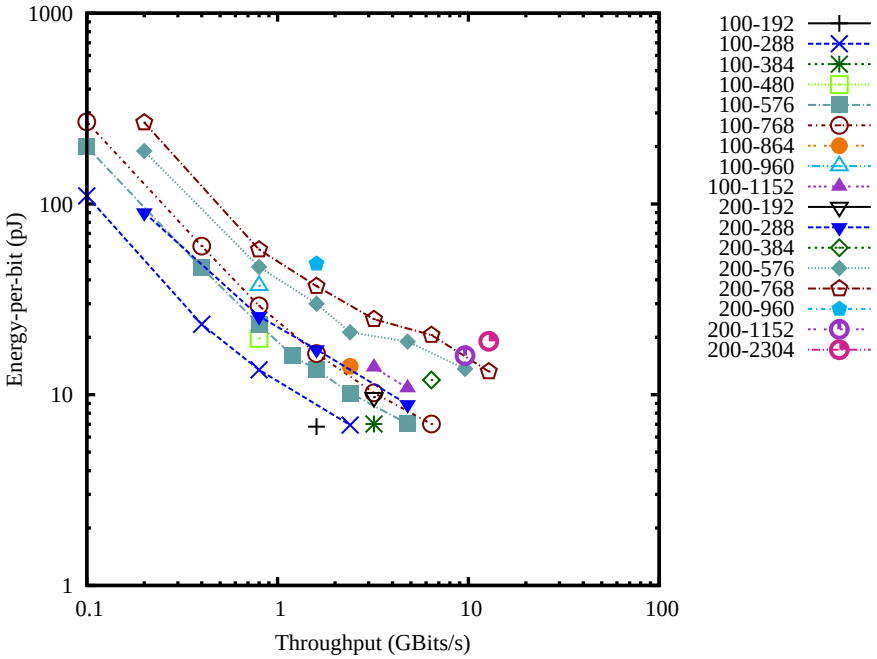

Figure 5.40: Decv6 energy-per-bit versus the information throughput for the PN-LDPC-CCs. The higher the ρ, the higher the information throughput and the lower the energy-per-decoded-bit. The energy-per-decoded-bit is expressed on a per-processor basis. Code lengths (T_s) and parallel-node (ρ) values are as specified: for T_s=192 ρ ∈{16}, for T_s=288 ρ ∈{1,4,8,24}, for T_s=384 ρ ∈{32}, for T_s=480 ρ ∈{8}, for T_s=576 ρ ∈{1,4,8,12,16,24,48}, for T_s=768 ρ ∈{1,4,8,16,32,64}, for T_s=864 ρ ∈{24}, for T_s=960 ρ ∈{8}, for T_s=1152 ρ ∈{32,48}, and for T_s=2304 ρ ∈{64}. Clock frequencies of 100 (prefix 100) and 200 (prefix 200) MHz are used.

0.1 dB. For 5 processors, the BER performance is indistinguishable. In cases where there are fewer than 5 processors, the min-sum operation outperforms the truncated min-sum.

Removing the LSB from the check-node operation reduces the power consumption and the area. Table 5.3 compares decv6 and decv5 in terms of area, power and throughput. The clock frequency is set to 200 MHz. In the case of the T_s=288, ρ=24 code with a 5-bit LLR, the power consumption is reduced by 29% and the decoder processor area is reduced by 12% percent.

Using the truncated min-sum operation significantly reduces the power consumption. Figure 5.42 compares decv6 (prefix 6) and decv5 (prefix 5) in terms of the energy-per-decoded-bit per processor versus throughput for various PN-LDPC-CCs.

Compared to the min-sum operation, the truncated min-sum operation significantly reduces the area consumption. Figure 5.43 compares decv6 and decv5 in terms of area per processor versus throughput.

Required area is primarily determined by T_s and secondary factors include:

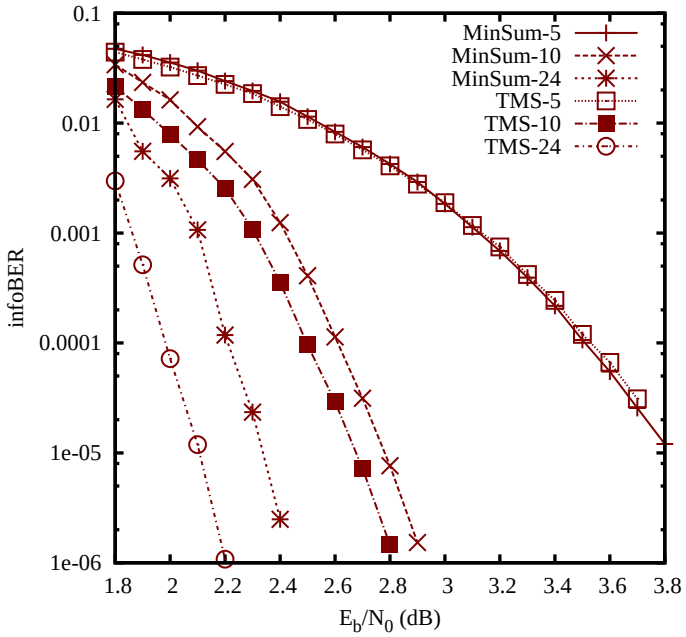

Figure 5.41: The T_s=288, ρ=24 code with 5, 10, and 24 processors is used to compare the BER performance of decoders using the min-sum (Min-Sum) and the truncated min-sum (TMS) operations. In all cases a 5-bit LLR is used. For 24 processors, the BER performance is improved by 0.2 dB. For 10 processors, the BER performance is improved by 0.1 dB. For 5 processors, the BER performance is indistinguishable.

Code	Version	LLR bit width	Area (um^2)	Energy per bit (pJ)	Through-put ($GBits/s$)
T_s 288 ρ 24	decv5	5	432666	12.54	4.800
T_s 288 ρ 24	decv6	5	379666	8.890	4.800

Table 5.3: Compares decv6 and decv5 in terms of area, power and through-put. The clock frequency is set to 200 MHz. A CMOS 90-nm process is used. We see that the truncated min-sum approach, used in decv6, significantly reduces both the area and the power consumption.

clock frequency and ρ. Figure 5.44 shows the standard cell area versus through-put versus ρ. Each curve in the figure corresponds to a fixed code length. Points along a curve, moving from left-to-right, represent an alternative code with the same T_s but an increased ρ.

Decv6 has a strong positive correlation between the energy-per-decoded-bit and T_s'. Figure 5.45 compares the processor energy-per-bit versus T_s' for a 100-MHz clock. Due to the clock-gating introduced in decv4, the relationship between the energy-per-decoded-bit and T_s' is non-linear. Other than reducing the energy-per-bit, using the TMS has no impact on the relationship between

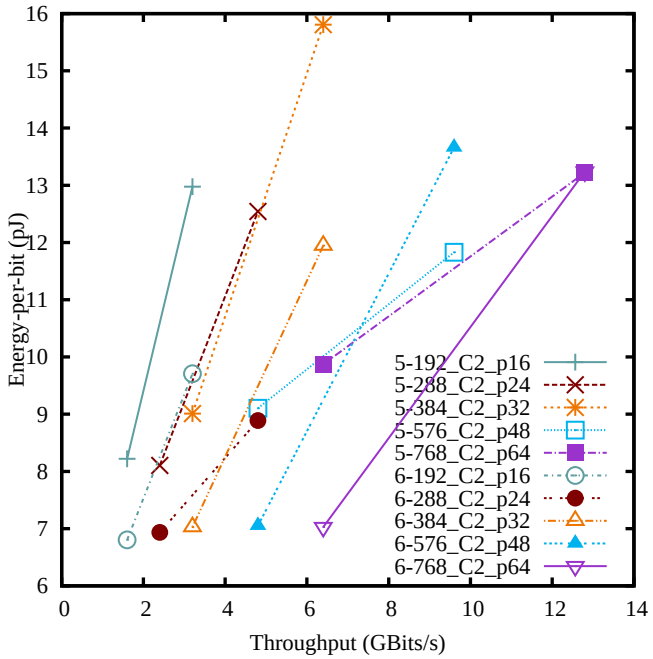

Figure 5.42: Decv6 (prefix 6) and decv5 (prefix 5) energy-per-decoded-bit per processor versus throughput for various PN-LDPC-CCs. Using the truncated min-sum operation reduces the power consumption. Clock frequencies of 100 and 200 MHz are used.

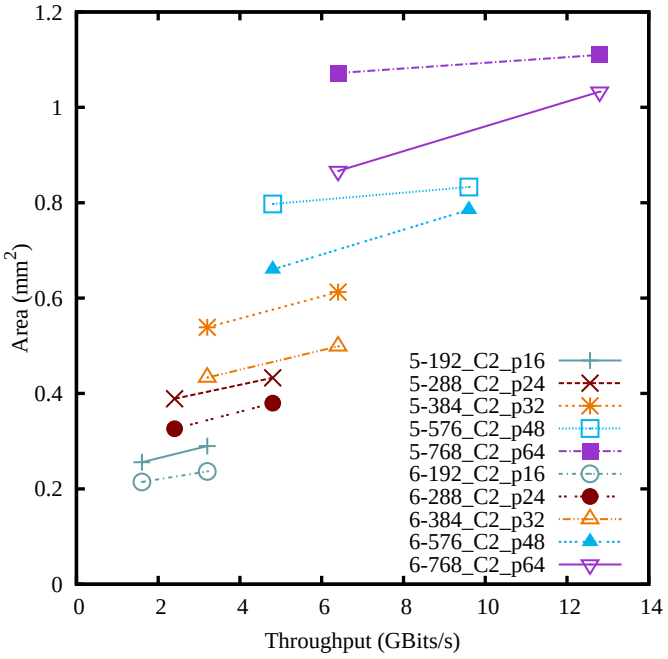

Figure 5.43: Decv6 and decv5 area per processor versus throughput for various PN-LDPC-CCs. Using the truncated min-sum check-node operation reduces the area. Clock frequencies of 100 and 200 MHz are used.

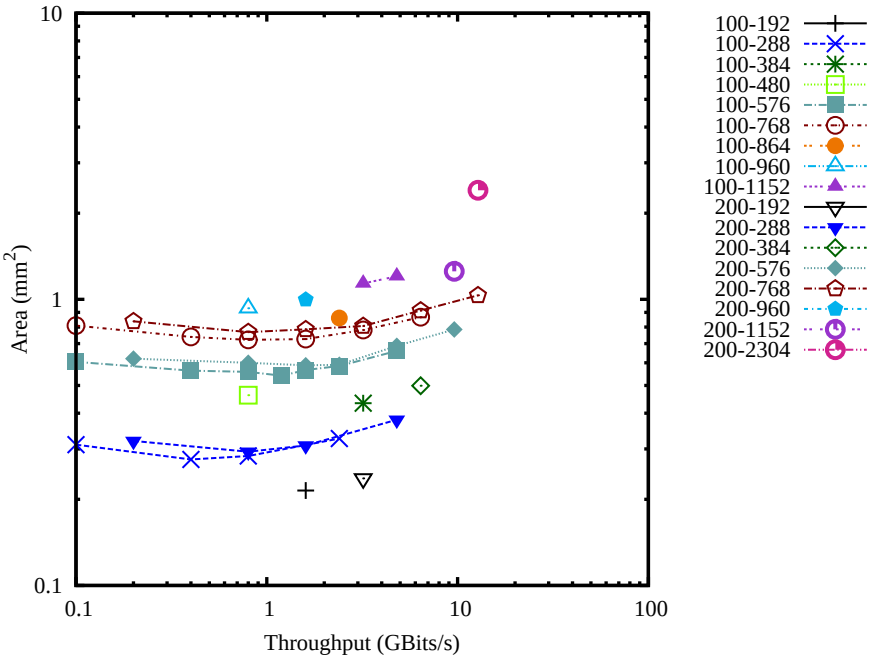

Figure 5.44: Decv6 area for one processor versus the information throughput, for all of the PN-LDPC-CCs. The area is expressed on a per-processor basis. Code lengths (T_s) and parallel-node (ρ) values are as specified: for T_s=192 $\rho \in \{16\}$, for T_s=288 $\rho \in \{1,4,8,24\}$, for T_s=384 $\rho \in \{32\}$, for T_s=480 $\rho \in \{8\}$, for T_s=576 $\rho \in \{1,4,8,12,16,24,48\}$, for T_s=768 $\rho \in \{1,4,8,16,32,64\}$, for T_s=864 $\rho \in \{24\}$, for T_s=960 $\rho \in \{8\}$, for T_s=1152 $\rho \in \{32,48\}$, and for T_s=2304 $\rho \in \{64\}$. Clock frequencies of 100 (prefix 100) and 200 (prefix 200) MHz are used.

the energy-per-decoded-bit and T_s'.

Decv6, displays the similar correlation between area and T_s. Figure 5.46 shows decv6 area versus T_s for all our PN-LDPC-CCs. A clock frequency of 100 MHz is used.

In Figure 5.41 we showed that the TMS operation outperformed the min-sum operation for an equal number of processors. We then showed that the TMS version of the decoder, decv6, generally consumes less power and area. Figure 5.47 compares the BER performance of decv5 (Min-Sum) and decv6 (TMS) for a fixed amount of area. In this case both decoder types occupy 4 mm^2 of area. The number of LLR magnitude bits is allowed to vary from 3 to 4. The number of processors is increased until the 4 mm^2 area limit is reached. As can be seen in the figure, decv6 (TMS) with 3 magnitude bits has the best BER performance. This trend holds true for the other codes and for the various amounts of area (processors). The trend is broken only for extremely few processors (i.e. 5 or less).

In summary, we have shown that decv6 (TMS) compared to decv5 (min-sum) improves the BER performance for the same number of processors. On a

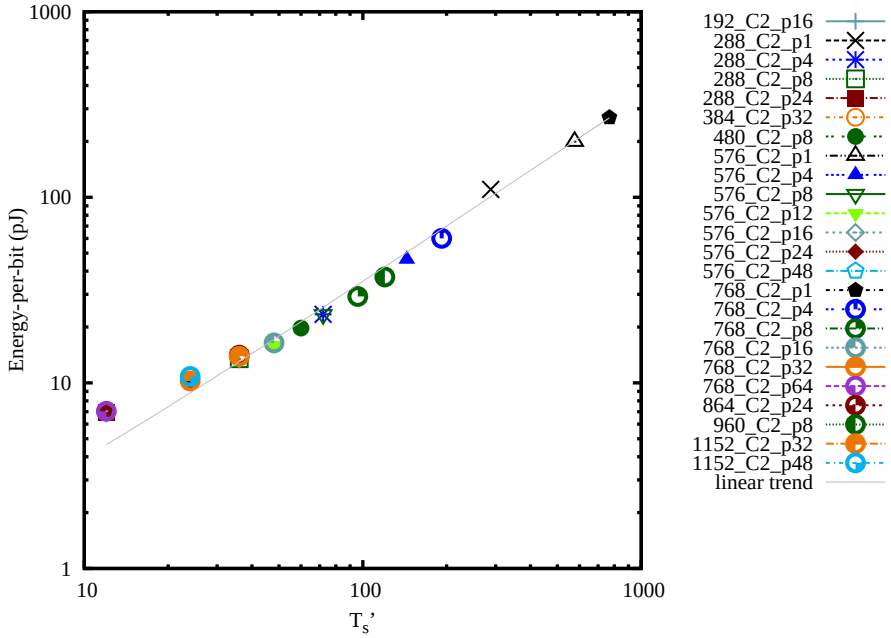

Figure 5.45: Decv6 energy-per-bit versus T_s' for a 100-MHz clock. There is a strong positive correlation between T_s' and the energy-per-decoded-bit.

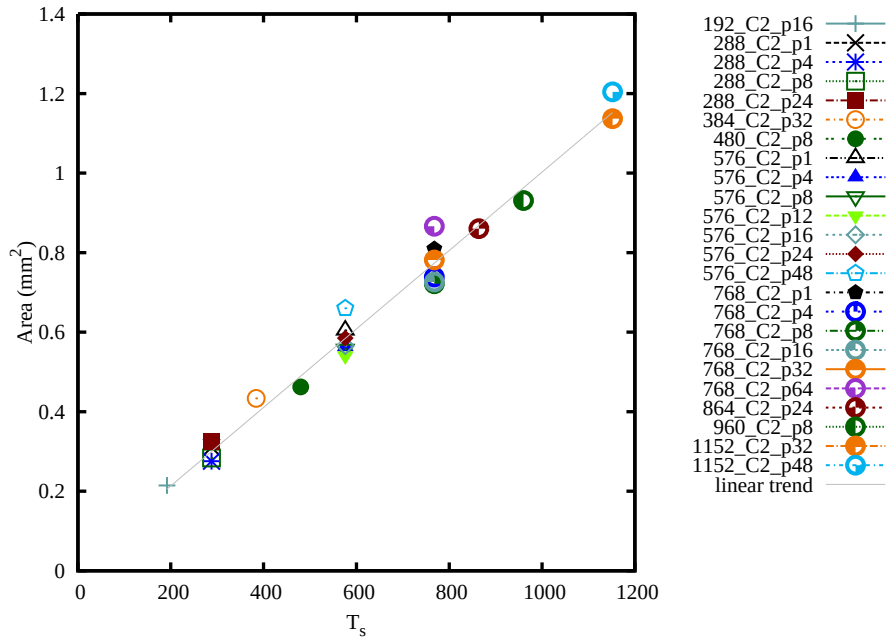

Figure 5.46: Decv6 area versus T_s for a 100-MHz clock.

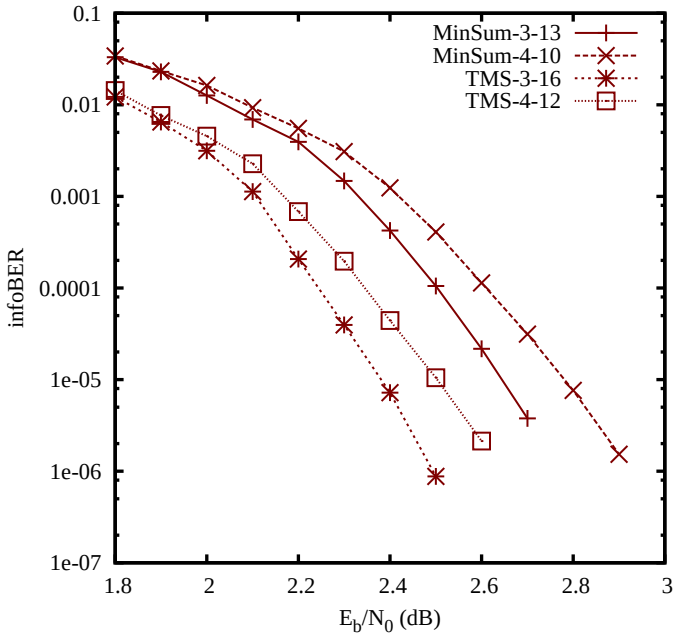

Figure 5.47: Decv6 (TMS) and decv5 (Min-Sum) BER performance for the T_s=288, ρ=24 code, constrained to 4 mm^2 of area. The number of LLR magnitude bits (3 or 4) and number of processors are varied to fit into the 4 mm^2 of area. Note how using the truncated min-sum check-node operation improves the BER performance.

per-processor basis, decv6 (TMS) compared to decv5 (min-sum) decreases the energy-per-decoded-bit by 21.0% on average (min 1.1%, max 45.5%, σ=8.5%) and reduces the processor area by 18.6% on average (min 11.3%, max 27.6%, σ=4.4%). In nearly all circumstances the min-sum operation/hardware should be replaced with the TMS operation/hardware.

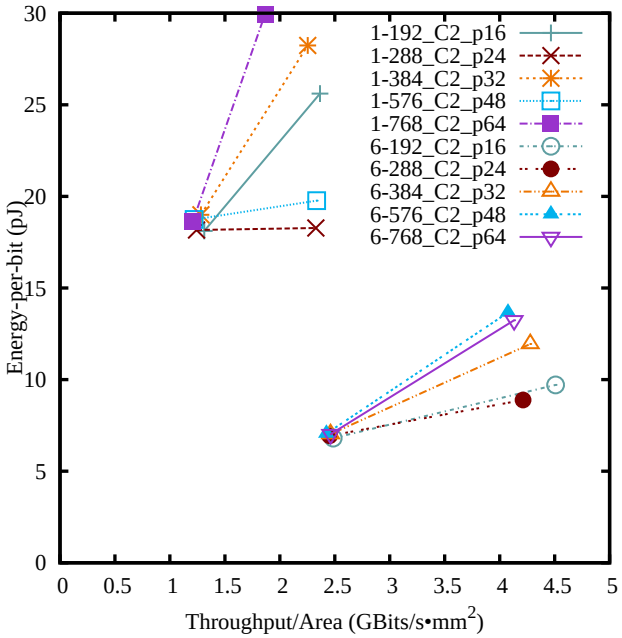

Figure 5.48: Decv6 (prefix 6) and decv1 (prefix 1) compared for T_s'=12 codes, in terms of the energy-per-bit versus the throughput/area. The results are based on a CMOS 90-nm process, 5-bit LLRs and 100 and 200-MHz clocks.

5.7 PN-LDPC-CC Decoder Hardware and BER Performance Analysis

In the previous sections we introduced a number of progressively improved PN-LDPC-CC decoder processor designs. The decoder designs were compared in terms of their energy-per-decoded-bit, area and information throughput. Codes with a group period T_s' of 12 have the lowest energy-per-decoded bit. Codes with the largest node-parallelization factor ρ have the highest throughput. Area is proportional to the code period T_s. The decv6 architecture has the lowest energy-per-decoded-bit and the smallest area per processor.

For the T_s'=12 codes, the decv6 processors, compared to the decv1 processors, reduce the energy-per-decoded-bit by approximately 2.5 times and they double the throughput-to-area ratio. Figure 5.48 compares decv6 and decv1 in terms of the energy-per-decoded-bit versus the throughput over the area for the T_s'=12 codes. The results are based on a CMOS 90-nm process and a 100 and 200 MHz clock. Note that, similar to the energy-per-decoded-bit, the throughput-to-area ratio is strongly correlated to T_s'.

The previous sections compare decoder processor designs using 4 bits of LLR magnitude precision. In this section we will take a closer look at the relationship of the hardware to the BER performance. First we will establish

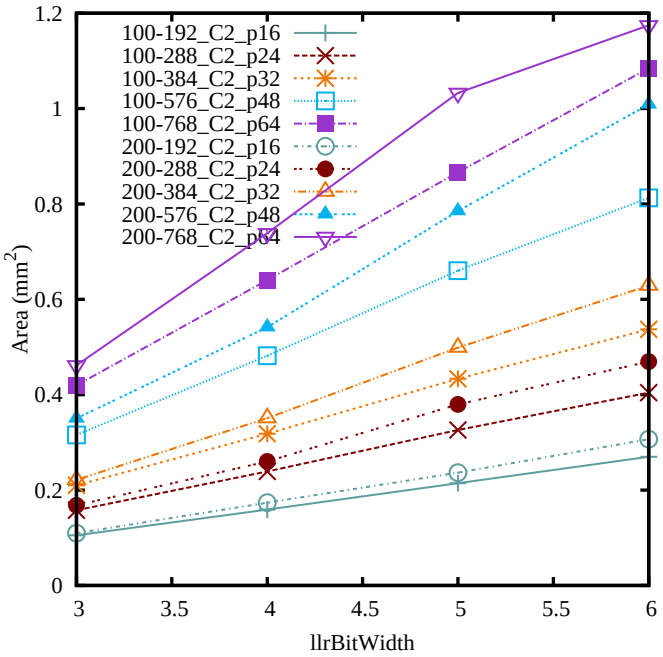

Figure 5.49: Decv6 processor area versus the LLR bit-width. The results are based on a CMOS 90-nm process and 100 and 200-MHz clocks (prefixes 100 and 200, respectively). Note that there is a nearly linear relationship between the decoder processor area, the LLR bit-width and clock frequency.

a relationship between the number of LLR bits and the energy-per-decoded-bit and the decoder processor area. The LLR-bits-to-energy-per-decoded-bit and LLR-bits-to-processor-area relationships will be used to determine our best hardware solution for a given amount of silicon area or a given power budget.

As one might expect, the decoder processor area is nearly linearly related to the LLR bit-width. Figure 5.49 plots the decoder processor area relative to the LLR bit-width. As can be seen, area grows nearly linearly with the number of LLR bits.

For a 100-MHz clock, the decv6 energy-per-decoded-bit is nearly linearly related to the number of LLR bits. However, for a 200-MHz clock, the energy-per-decoded-bit is no longer linearly related to the number of LLR bits. The increase in clock frequency makes it harder for the synthesis tool to meet timing. As a result the designs that the synthesis tool finds have more variation between them. Figure 5.50 shows the relationship between the number of LLR bits and the energy-per-decoded-bit. For the codes examined, as the LLR bit-width increases the power consumed increases.

Tables 5.4 and 5.5 summarize, for a single decv6 processor, the area, power and information throughput for 100 and 200-MHz clocks. PN-LDPC-CC Decoders consist of a cascade of multiple decoder processors. The decoder area, power and energy-per-decoded-bit for a decoder is thus equal to the value for a

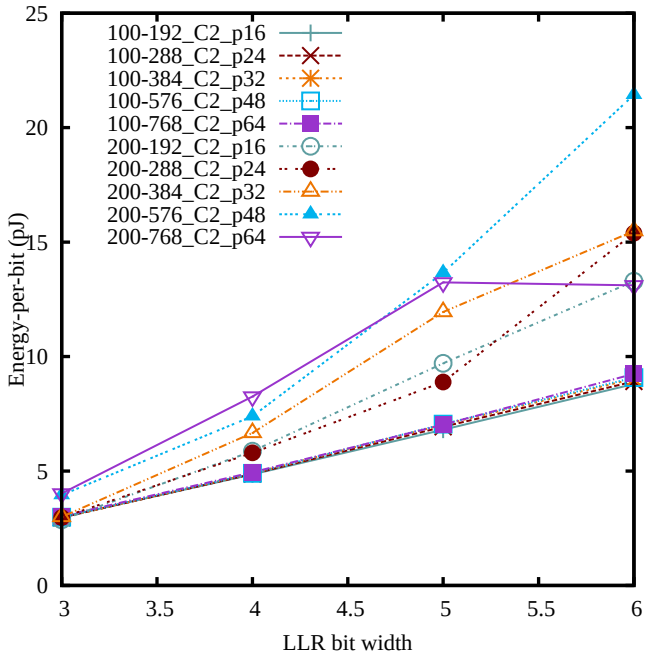

Figure 5.50: Decv6 energy-per-decoded-bit per processor versus the LLR bit-width for various PN-LDPC-CC codes. The results are based on CMOS 90-nm process and 100 and 200-MHz clocks. For the 100-MHz clock (prefix 100), note the linear relationship between the energy-per-decoded-bit and the bit-width of the LLRs. For the 200 MHz clock (prefix 200) we begin to see non-linear increases in the energy-per-decoded-bit with increasing LLR bit-widths.

single decoder processor multiplied by the number of processors. The decoder throughput remains unchanged as the number of processors increases, however, the overall error-correcting-performance should increase. The decoder throughput/area for a decoder is equal to the single decoder processor value divided by the number of processors. The decoder processor running at 200 MHz has greater throughput/area and higher energy-per-decoded-bit than the same decoder processor running at 100 MHz. Given the metrics of a single PN-LDPC-CC decoder processor, it is possible to calculate the metrics of the PN-LDPC-CC decoder provided the number of decoder processors is known.

Now that we know the hardware cost for each decoder processor, two key questions arise: "What is the best BER performance that we can achieve given a fixed amount of area?" and "What is the best BER performance that we can achieve given a fixed amount of power?" The BER performance of the T_s'=12 codes will now be examined while constraining first the area and then the power. Specifically, the area will be constrained to 1, 2, 3, 4, 5, 6, 8, 10 and 12 mm^2. Then the power will be constrained to 100, 200, 300 and 400 mW.

For $1mm^2$ of decoder area, choosing T_s=192, ρ=16 with 3-bit magnitude

Code	LLR bit width ($bits$)	Area (mm^2)	Power (mW)	Through-put ($GBits/s$)	Energy per bit (pJ)	Throughput /Area ($GBits/s \cdot mm^2$)
T_s 192 ρ 16	3	0.105	4.773	1.600	2.983	5.069
T_s 192 ρ 16	4	0.159	7.767	1.600	4.853	3.350
T_s 192 ρ 16	5	0.215	10.88	1.600	6.803	2.486
T_s 192 ρ 16	6	0.270	14.08	1.600	8.800	1.977
T_s 288 ρ 24	3	0.158	7.117	2.400	2.966	5.071
T_s 288 ρ 24	4	0.239	11.67	2.400	4.867	3.348
T_s 288 ρ 24	5	0.326	16.64	2.400	6.933	2.453
T_s 288 ρ 24	6	0.404	21.38	2.400	8.910	1.979
T_s 384 ρ 32	3	0.210	9.467	3.200	2.958	5.080
T_s 384 ρ 32	4	0.318	15.67	3.200	4.897	3.352
T_s 384 ρ 32	5	0.433	22.48	3.200	7.027	2.461
T_s 384 ρ 32	6	0.537	28.89	3.200	9.030	1.987
T_s 576 ρ 48	3	0.316	14.32	4.800	2.985	5.071
T_s 576 ρ 48	4	0.482	23.51	4.800	4.900	3.322
T_s 576 ρ 48	5	0.660	33.83	4.800	7.050	2.424
T_s 576 ρ 48	6	0.813	43.60	4.800	9.083	1.969
T_s 768 ρ 64	3	0.420	19.33	6.400	3.021	5.082
T_s 768 ρ 64	4	0.640	31.64	6.400	4.943	3.331
T_s 768 ρ 64	5	0.866	44.96	6.400	7.027	2.462
T_s 768 ρ 64	6	1.085	59.16	6.400	9.243	1.967

Table 5.4: A single decv6 decoder processor in a CMOS 90-nm process, running at 100 MHz, comparing area, power and throughput versus LLR bit-width.

LLRs and 12 processors results in the best BER performance. Our results are based on a CMOS 90-nm process and a 100-MHz clock. Note that all of the decoders use the same area; however, the required E_b/N_0 to achieve the target BER of 10^{-6} can vary by as much as 1.0 dB. The following figures show the BER performance achieved given increasing area budgets ranging from 1 mm^2 to 12 mm^2. Figures 5.51 to 5.59 show the BER characteristics for areas of 1, 2, 3, 4, 5, 6, 8, 10 and 12 mm^2, respectively.

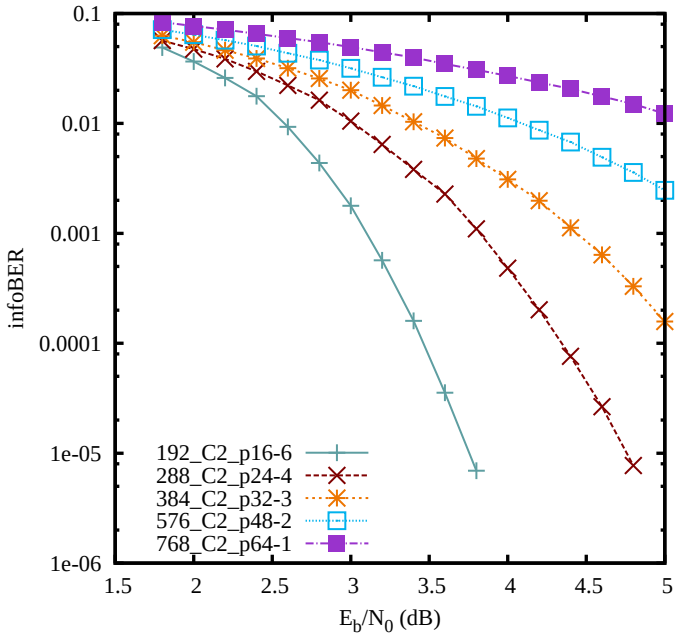

Figure 5.51: For 1 mm^2 of area, the T_s=192, ρ=16 code with 4-bit LLRs and 6 processors (192_C2_p16-6) has the best BER performance of 10^{-6} at an E_b/N_0 of 4.0 dB. The results are based on a CMOS 90-nm process and a 100-MHz clock.

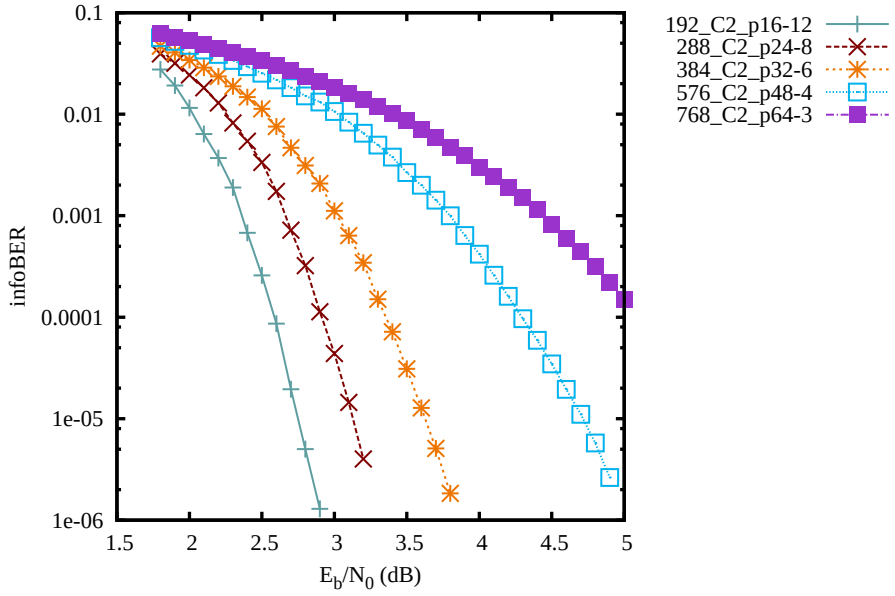

Figure 5.52: For 2 mm^2 of area, the T_s=192, ρ=16 code with 4-bit LLRs and 12 processors (192_C2_p16-12) has the best BER performance of 10^{-6} at an E_b/N_0 of 2.9 dB. The results are based on a CMOS 90-nm process and a 100-MHz clock.

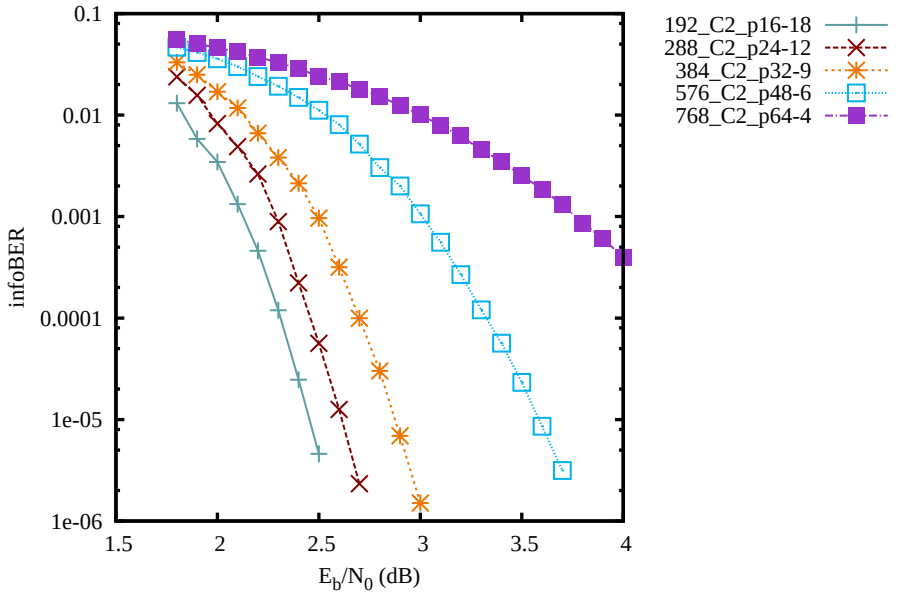

Figure 5.53: For 3 mm^2 of area, the T_s=192, ρ=16 code with 4-bit LLRs and 18 processors (192_C2_p16-18) has the best BER performance of 10^{-6} at an E_b/N_0 of 2.6 dB. The results are based on a CMOS 90-nm process and a 100-MHz clock.

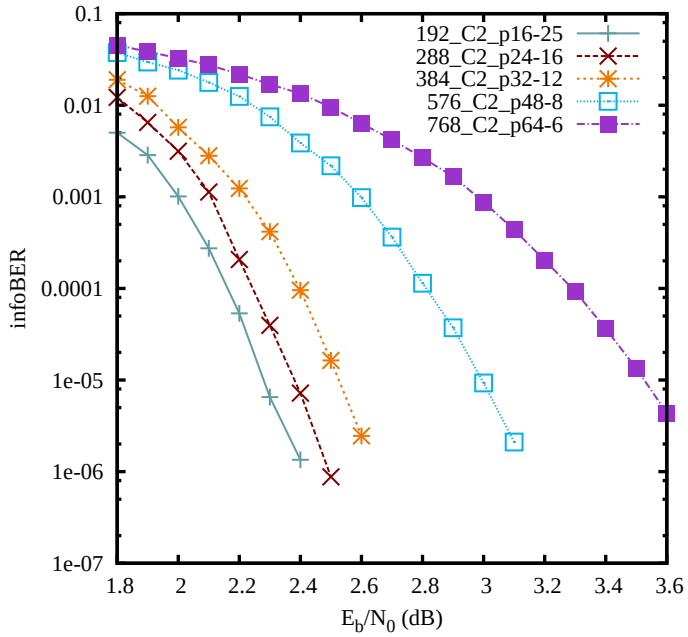

Figure 5.54: For 4 mm^2 of area, the T_s=192, ρ=16 code with 4-bit LLRs and 25 processors (192_C2_p16-25) has the best BER performance of 10^{-6} at an E_b/N_0 of 2.4 dB. The results are based on a CMOS 90-nm process and a 100-MHz clock.

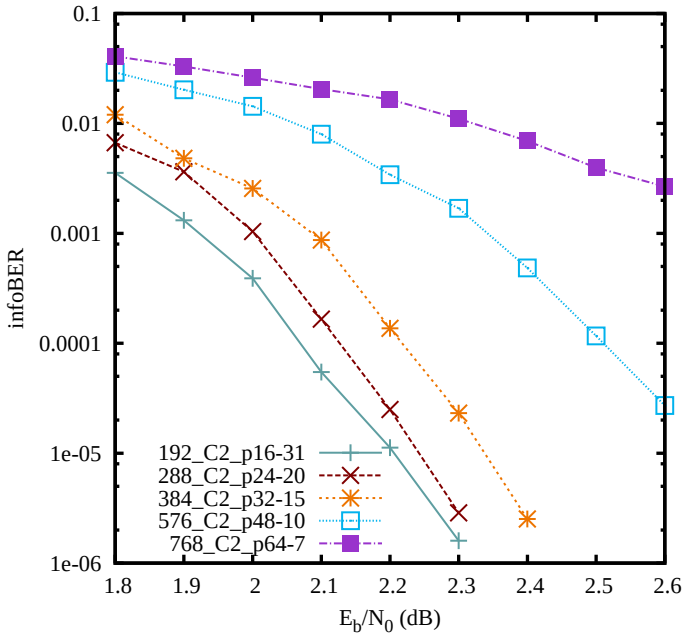

Figure 5.55: For 5 mm^2 of area, the T_s=192, ρ=16 code with 4-bit LLRs and 31 processors (192_C2_p16-31) has the best BER performance of 10^{-6} at an E_b/N_0 of 2.35 dB. The results are based on a CMOS 90-nm process and a 100-MHz clock.

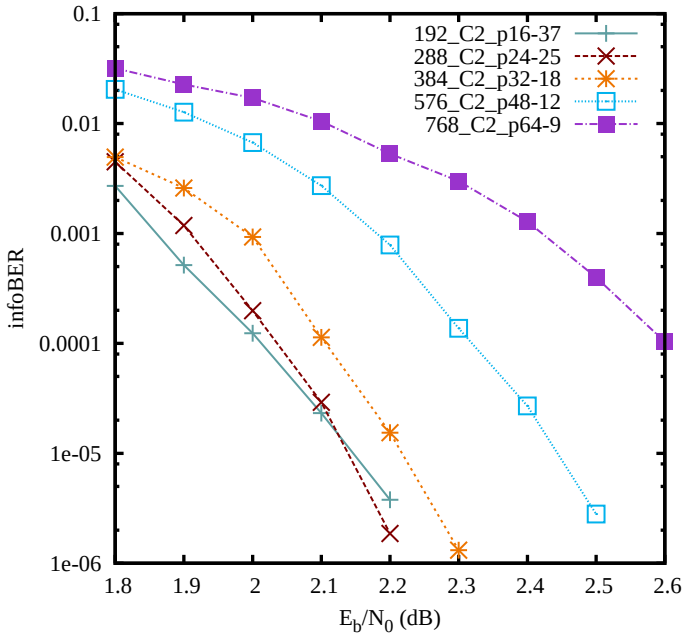

Figure 5.56: For 6 mm^2 of area, the T_s=288, ρ=24 code with 4-bit LLRs and 25 processors (288_C2_p24-25) has the best BER performance of 10^{-6} at an E_b/N_0 of 2.2 dB. The results are based on a CMOS 90-nm process and a 100-MHz clock.

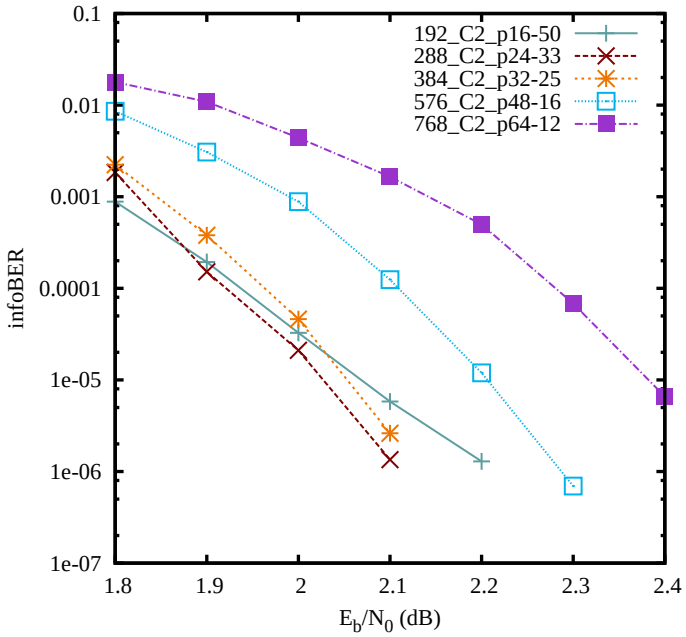

Figure 5.57: For 8 mm^2 of area, the T_s=288, ρ=24 code with 4-bit LLRs and 33 processors (288_C2_p24-33) has the best BER performance of 10^{-6} at an E_b/N_0 of 2.1 dB. The results are based on a CMOS 90-nm process and a 100-MHz clock.

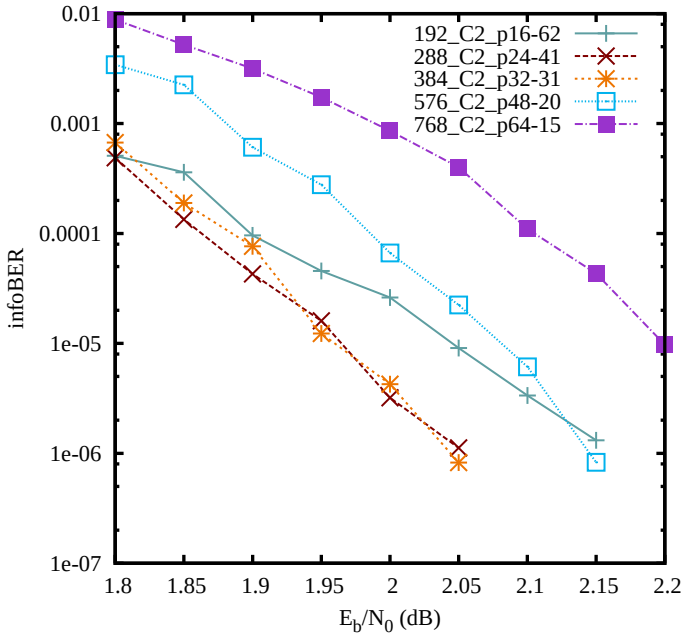

Figure 5.58: For 10 mm^2 of area, the T_s=384, ρ=32 code with 4-bit LLRs and 31 processors (384_C2_p32-31) has the best BER performance of 10^{-6} at an E_b/N_0 of 2.05 dB. The results are based on a CMOS 90-nm process and a 100-MHz clock.

Code	LLR bit width ($bits$)	Area (mm^2)	Power (mW)	Through-put ($GBits/s$)	Energy per bit (pJ)	Throughput /Area ($GBits/s \cdot mm^2$)
T_s 192 ρ 16	3	0.110	9.180	3.200	2.868	9.741
T_s 192 ρ 16	4	0.174	18.76	3.200	5.863	6.146
T_s 192 ρ 16	5	0.237	31.06	3.200	9.707	4.509
T_s 192 ρ 16	6	0.307	42.53	3.200	13.29	3.479
T_s 288 ρ 24	3	0.168	14.23	4.800	2.965	9.525
T_s 288 ρ 24	4	0.260	27.82	4.800	5.797	6.151
T_s 288 ρ 24	5	0.380	42.66	4.800	8.890	4.213
T_s 288 ρ 24	6	0.469	73.86	4.800	15.39	3.410
T_s 384 ρ 32	3	0.221	19.15	6.400	2.993	9.637
T_s 384 ρ 32	4	0.351	42.66	6.400	6.663	6.078
T_s 384 ρ 32	5	0.499	76.46	6.400	11.94	4.278
T_s 384 ρ 32	6	0.629	99.06	6.400	15.48	3.393
T_s 576 ρ 48	3	0.350	37.90	9.600	3.947	9.133
T_s 576 ρ 48	4	0.542	70.93	9.600	7.390	5.909
T_s 576 ρ 48	5	0.785	131.1	9.600	13.66	4.076
T_s 576 ρ 48	6	1.009	205.8	9.600	21.43	3.172
T_s 768 ρ 64	3	0.462	51.43	12.80	4.020	9.242
T_s 768 ρ 64	4	0.739	105.6	12.80	8.253	5.776
T_s 768 ρ 64	5	1.033	169.4	12.80	13.24	4.132
T_s 768 ρ 64	6	1.175	167.8	12.80	13.11	3.633

Table 5.5: A single decv6 decoder processor in a CMOS 90-nm process, running at 200 MHz, comparing area, power and throughput versus LLR bit-width.

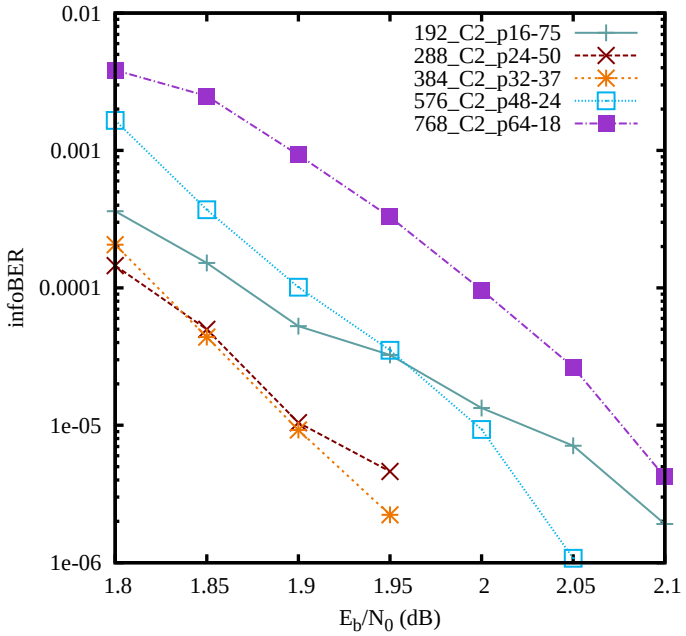

Figure 5.59: For 12 mm^2 of area, the T_s=384, ρ=32 code with 4-bit LLRs and 37 processors (384_C2_p32-37) has the best BER performance of 10^{-6} at an E_b/N_0 of 1.97 dB. The results are based on a CMOS 90-nm process and a 100-MHz clock.

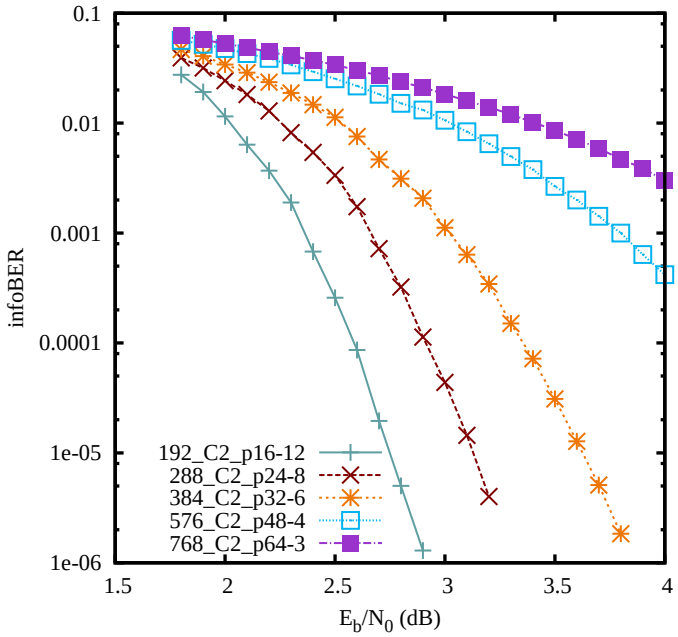

Figure 5.60: For 100 mW of power, the T_s=192, ρ=16 with 4-bit LLRs and 12 processors (192_C2_p16-12) has the best BER performance of 10^{-6} at an E_b/N_0 of 2.9 dB. The results are based on a CMOS 90-nm process and a 100-MHz clock. Note that all the decoders use the same amount of power, but vary the codes, number of processors and LLR bit-width. The T_s=192, 288, 384, 576 and 768 codes have an energy-per-decoded-bit of 62.5 nJ, 41.7 nJ, 31.3 nJ, 20.8 nJ and 15.6 nJ, respectively.

For our decoders and a given power budget, there is an optimal combination of codes, LLR bit-widths and processors that will minimize the BER. The following BER performance figures show our best hardware solutions, given a fixed amount of power. Figures 5.60, 5.61, 5.62 and 5.63 shows the achievable BER given a power limit of 100, 200, 300 and 400 mW, respectively. The energy-per-bit in the caption of Figure 5.60 scales linearly with the power.

From the BER performance given area or power constraints, we see that the BER differs dramatically. From this we can conclude that when comparing LDPC decoders, it is extremely important that the BER performance and the E_b/N_0 be matched before comparing power, area or other hardware metrics. Failure to align the BER performance and the E_b/N_0 will result in an inaccurate comparison and misleading conclusions.

While area and power are important design metrics they do not capture a full and satisfactory notion of decoder implementation "efficiency". It can be argued that more useful metrics for LDPC decoders are the energy-per-decoded-bit and the throughput/area. Figure 5.64 compares the energy-per-decoded-bit versus the throughput/area for various T_s'=12 codes with 4-bit LLRs in 4 different process technologies. The process technologies include

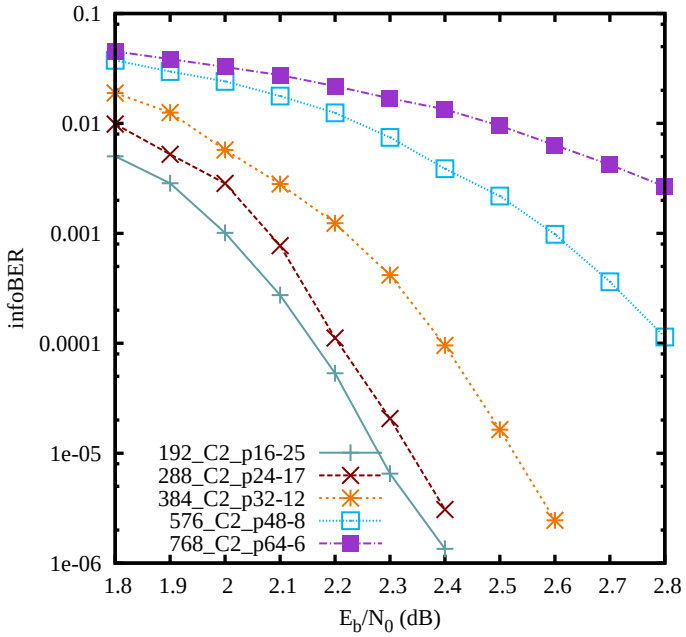

Figure 5.61: For 200 mW of power, the T_s=192, ρ=16 with 4-bit LLRs and 25 processors (192_C2_p16-25) has the best BER performance of 10^{-6} at an E_b/N_0 of 2.4 dB. The results are based on a CMOS 90-nm process and a 100-MHz clock.

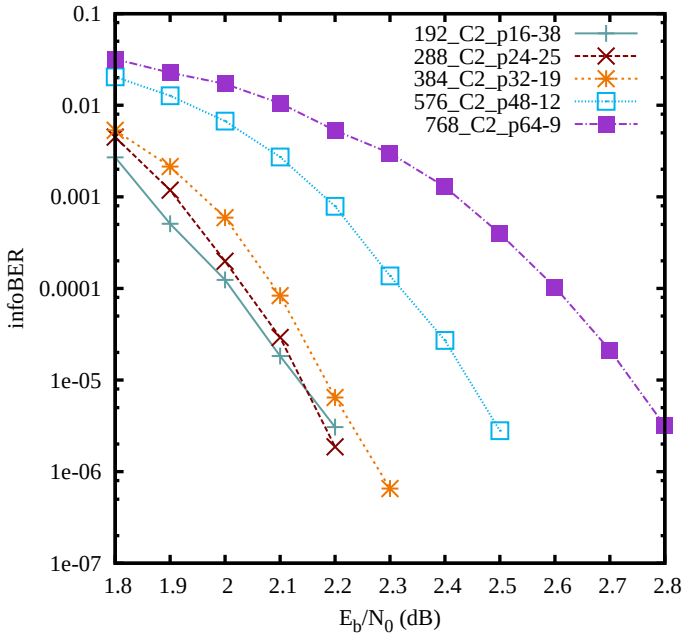

Figure 5.62: For 300 mW of power, the T_s=288, ρ=24 with 4-bit LLRs and 25 processors (288_C2_p24-25) has the best BER performance of 10^{-6} at an E_b/N_0 of 2.2 dB. The results are based on a CMOS 90-nm process and a 100-MHz clock.

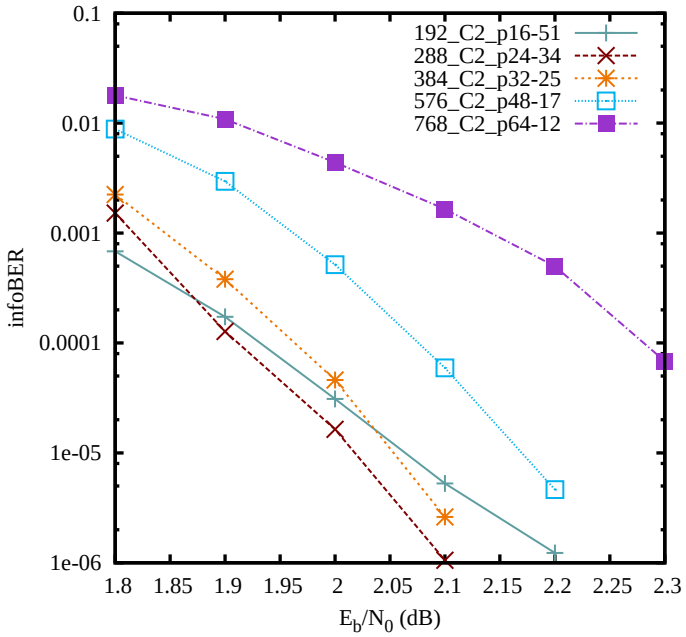

Figure 5.63: For 400 mW of power, the T_s=288, ρ=24 with 4-bit LLRs and 34 processors (288_C2_p24-34) has the best BER performance of 10^{-6} at an E_b/N_0 of 2.1 dB. The results are based on a CMOS 90-nm process and a 100-MHz clock.

TSMC 180-nm CMOS (art18.3.0), IBM 130-nm CMOS (ibm130.1.5), STM 90-nm CMOS (cmos90nm.3.0) and STM 65-nm CMOS (cmos65nm). The results are based upon a E_b/N_0 of 1.8 dB and the clock frequencies 50 MHz, 100 MHz, 150 MHz and 200 MHz.

Each process technology has different energy-per-decoded-information-bit versus throughput/area curves. Figure 5.64 is split by process into four Figures: CMOS 180-nm (Figure 5.65), CMOS 130-nm (Figure 5.66), CMOS 90-nm (Figure 5.67) and CMOS 65-nm (Figure 5.68). As can be seen from the previous figures, the process technology significantly impacts both the energy-per-decoded-information-bit and the throughput/area metrics. In general, the more recent technology processes achieve a lower energy-per-decoded-information-bit and a greater throughput-to-area ratio. The ratio of static to dynamic power consumption is higher in the CMOS 90-nm process technology compared to other process technologies. This creates the bath tub curve seen in Figure 5.67. In all process technologies, increased throughput/area can be traded off for an increase in the energy-per-decoded-information-bit. In the specific process technologies, the decv6 architecture sees only moderate differentiation between the various $T_s's$=12 codes in terms of energy-per-decoded-information-bit and the throughput/area. In general for the examined process technologies, a sweet spot for the energy-per-decoded-information bit exists at around 100 MHz. The exception to this observation is found in the CMOS

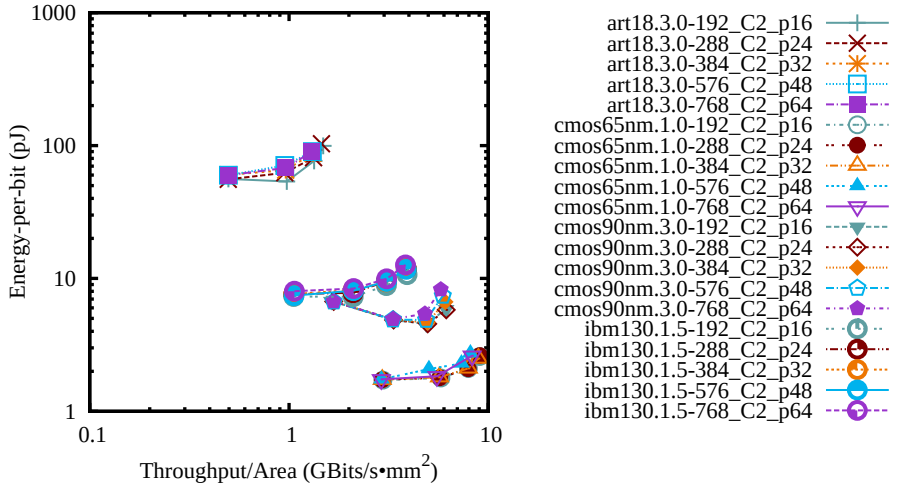

Figure 5.64: Decv6 energy-per-decoded-bit versus the throughput/area for various T_s'=12 codes with 4-bit LLRs in 4 different process technologies. Process technologies include TSMC 180-nm CMOS (art18.3.0), IBM 130-nm CMOS (ibm130.1.5), STM 90-nm CMOS (cmos90nm.3.0) and STM 65-nm CMOS (cmos65nm). The results are based upon a E_b/N_0 of 1.8 dB. The clock frequency was set to 50, 100, 150 and 200 MHz. Based on process technology, the curves in this Figure are split out in to individual Figures 5.65, 5.66, 5.67 and 5.68.

90-nm process, where the sweet spot is located at around 150 MHz. In all the process technologies examined, an increase in the clock frequency up to 150-200 MHz increases the throughput/area at a greater rate than the energy-per-decoded-information-bit.

In previous sections, we established for the CMOS 90-nm process, with designs running at 100 MHz, that the energy-per-decoded-bit varied nearly linearly with T_s'. While this generally remains true, the correlation is weaker for the other examined process technologies. We have seen that the energy-per-decoded-bit can vary by as much as 25% at 100 MHz (see Figure 5.65 CMOS 180-nm energy-per-decoded-bit versus throughput/area).

Given a target BER and E_b/N_0, there is a combination of codes, processors and LLR bit-widths that minimizes the energy-per-decoded-information-bit or that maximizes the throughput/area. Tables 5.6, 5.7, 5.8 and 5.9 list the hardware metrics for the decv6 decoder processor with 4-bit LLRs in 65-nm, 90-nm, 130-nm and 180-nm CMOS processes, respectively. Combined with the number of processors required to achieve the target BER performance, given an E_b/N_0, it is possible to determine the optimum solution that minimizes the

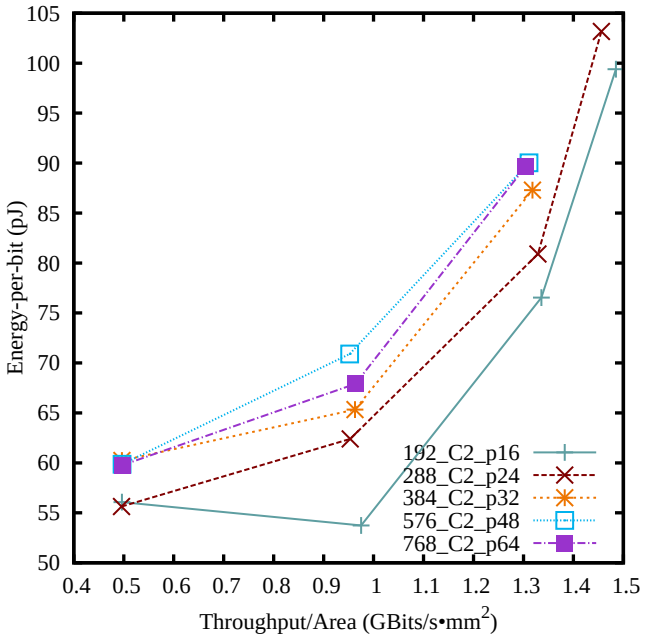

Figure 5.65: Decoder processor energy-per-decoded-bit versus the throughput/area in a TSMC 180-nm CMOS process with 4-bit LLRs. The results are based upon a E_b/N_0 of 1.8 dB and an supply voltage of The clock frequency was set to 50, 100, 150 and 200 MHz. Note that two codes meet timing at a clock frequency of 200 MHz.

energy-per-decoded-information-bit or that maximizes the throughput/area. Table 5.10 shows, for each T_s'=12 code, the number of processors required to a BER of 10^{-6} at the target E_b/N_0. The results in Table 5.10 are based on fixed-point simulations.

Our presented decoder architectures have latencies similar to LDPC-BC decoders. The latency for our decoders is given by Equation (5.4).

$$T_{latency} = T_s' \cdot N_{processors} \cdot T_{clk} \tag{5.4}$$

where T_s' is the group period, $N_{processors}$ is the number of decoder processors and T_{clk} is the period of the clock. In the case of a T_s'=12 code, with 15 decoder processors and a 100-MHz clock, the latency would be 1.8 μs. This is similar to what is reported for LDPC-BCs [33].

This concludes the hardware and BER performance analysis. In the next section we use the results presented in this section to compare against the state-of-the-art in LDPC decoders.

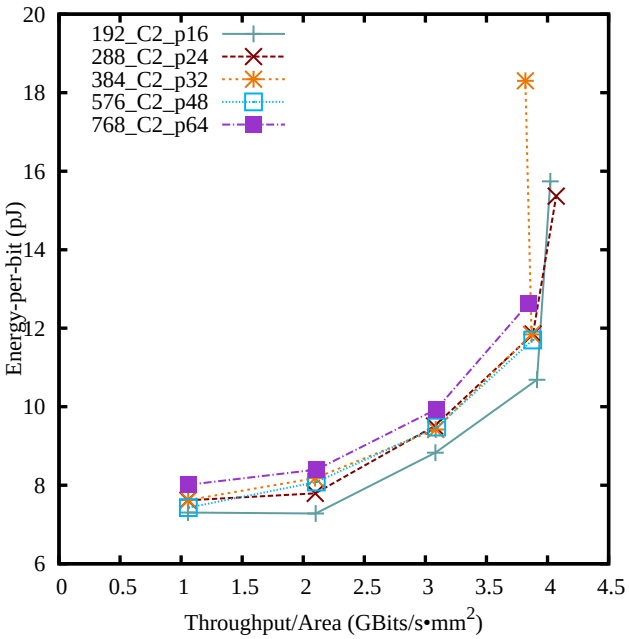

Figure 5.66: Decoder processor energy-per-decoded-bit versus the throughput/area in a IBM 130-nm CMOS process with 4-bit LLRs. The results are based upon a E_b/N_0 of 1.8 dB. The clock frequency was set to 50, 100, 150 and 200 MHz. Note that three codes meet timing at a clock frequency of 200 MHz.

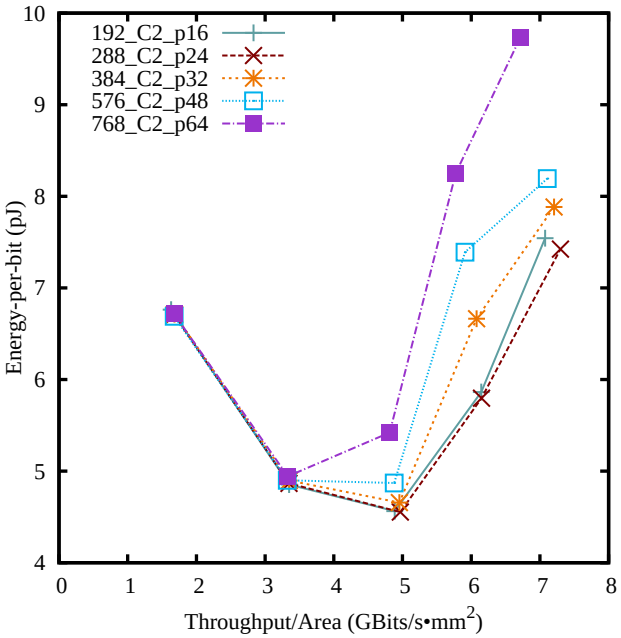

Figure 5.67: Decoder processor energy-per-decoded-bit versus the throughput/area in a STM 90-nm CMOS process with 4-bit LLRs. The results are based upon a E_b/N_0 of 1.8 dB. The clock frequency was set to 50, 100, 150 and 200 MHz. Note that the bathtub curve is caused by leakage: not until 150 MHz does the leakage power drop down to represent 25% of the total power consumption. In none of the other process technologies does the leakage power ever reach these levels.

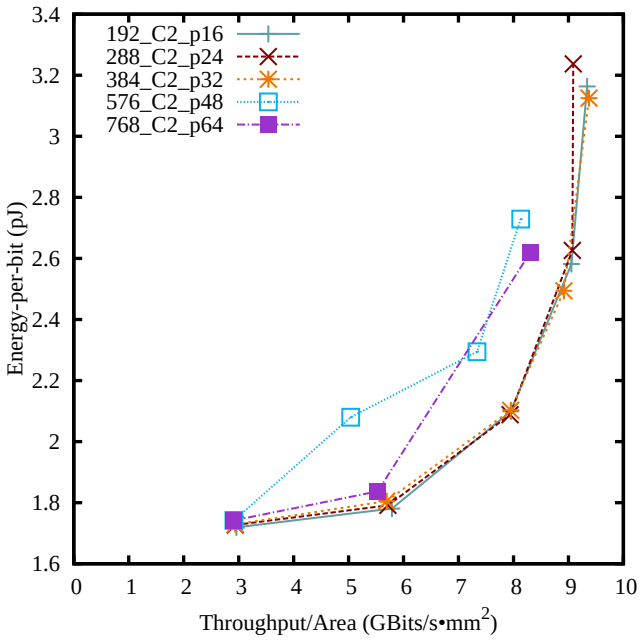

Figure 5.68: Decoder processor energy-per-decoded-bit versus the throughput/area in a STM 65-nm CMOS process with 4-bit LLRs. The results are based upon a E_b/N_0 of 1.8 dB. The clock frequency was set to 50, 100, 150 and 200 MHz. Note that three codes meet timing at a clock frequency of 200 MHz.

Code	Clock Freq. (MHz)	Area (um^2)	Power (mW)	Through-put ($GBits/s$)	Energy per bit (pJ)	Throughput /Area ($GBits/s \cdot mm^2$)
T_s 192 ρ 16	50	90066	1.375	0.800	1.719	8.880
T_s 192 ρ 16	100	92133	2.849	1.600	1.781	17.36
T_s 192 ρ 16	150	100600	5.040	2.400	2.099	23.85
T_s 192 ρ 16	200	117800	8.260	3.200	2.582	27.16
T_s 288 ρ 24	50	136100	2.072	1.200	1.727	8.817
T_s 288 ρ 24	100	139966	4.300	2.400	1.791	17.14
T_s 288 ρ 24	150	151200	7.517	3.600	2.088	23.81
T_s 288 ρ 24	200	176366	12.60	4.800	2.626	27.21
T_s 384 ρ 32	50	181533	2.767	1.600	1.729	8.814
T_s 384 ρ 32	100	187366	5.777	3.200	1.805	17.07
T_s 384 ρ 32	150	201266	10.09	4.800	2.103	23.84
T_s 384 ρ 32	200	239233	15.96	6.400	2.494	26.75
T_s 576 ρ 48	50	274633	4.180	2.400	1.742	8.739
T_s 576 ρ 48	100	317600	9.983	4.800	2.080	15.11
T_s 576 ρ 48	150	327166	16.52	7.200	2.294	22.00
T_s 576 ρ 48	200	393333	26.20	9.600	2.729	24.39
T_s 768 ρ 64	50	367000	5.573	3.200	1.742	8.724
T_s 768 ρ 64	100	385666	11.75	6.400	1.837	16.58
T_s 768 ρ 64	150	na	na	9.600	na	na
T_s 768 ρ 64	200	513999	33.50	12.80	2.618	24.90

Table 5.6: CMOS 65-nm 4-bit LLR decv6 decoder processor area, power, throughput, energy-per-bit and throughput/area. To determine a decoder's hardware metrics, multiply the area, power and energy-per-bit by the required number of processors and divide the throughput/area by the required number of processors.

Code	Clock Freq. (MHz)	Area (um^2)	Power (mW)	Through-put ($GBits/s$)	Energy per bit (pJ)	Throughput /Area ($GBits/s \cdot mm^2$)
T_s 192 ρ 16	50	163433	5.410	0.800	6.763	4.896
T_s 192 ρ 16	100	159200	7.767	1.600	4.853	10.05
T_s 192 ρ 16	150	163700	10.95	2.400	4.563	14.66
T_s 192 ρ 16	200	173533	18.76	3.200	5.863	18.43
T_s 288 ρ 24	50	238400	8.050	1.200	6.707	5.034
T_s 288 ρ 24	100	238933	11.67	2.400	4.867	10.04
T_s 288 ρ 24	150	241533	16.39	3.600	4.553	14.90
T_s 288 ρ 24	200	260100	27.82	4.800	5.797	18.45
T_s 384 ρ 32	50	317366	10.74	1.600	6.713	5.043
T_s 384 ρ 32	100	318200	15.67	3.200	4.897	10.05
T_s 384 ρ 32	150	322900	22.35	4.800	4.657	14.86
T_s 384 ρ 32	200	351000	42.66	6.400	6.663	18.23
T_s 576 ρ 48	50	477333	16.05	2.400	6.690	5.028
T_s 576 ρ 48	100	481666	23.51	4.800	4.900	9.966
T_s 576 ρ 48	150	492000	35.06	7.200	4.870	14.63
T_s 576 ρ 48	200	541666	70.93	9.600	7.390	17.72
T_s 768 ρ 64	50	637000	21.49	3.200	6.717	5.022
T_s 768 ρ 64	100	640333	31.64	6.400	4.943	9.993
T_s 768 ρ 64	150	665000	52.03	9.600	5.420	14.43
T_s 768 ρ 64	200	738666	105.6	12.80	8.253	17.32

Table 5.7: CMOS 90-nm 4-bit LLR decv6 decoder processor area, power, throughput, energy-per-bit and throughput/area. To determine a decoder's hardware metrics, multiply the area, power and energy-per-bit by the required number of processors and divide the throughput/area by the required number of processors.

Code	Clock Freq. (MHz)	Area (mm^2)	Power (mW)	Through-put ($GBits/s$)	Energy per bit (pJ)	Throughput /Area ($GBits/s \cdot mm^2$)
T_s 192 ρ 16	50	0.253	5.843	0.800	7.303	3.168
T_s 192 ρ 16	100	0.254	11.65	1.600	7.280	6.303
T_s 192 ρ 16	150	0.260	21.19	2.400	8.830	9.246
T_s 192 ρ 16	200	0.272	34.20	3.200	10.68	11.74
T_s 288 ρ 24	50	0.379	9.137	1.200	7.617	3.168
T_s 288 ρ 24	100	0.381	18.70	2.400	7.793	6.294
T_s 288 ρ 24	150	0.390	34.23	3.600	9.510	9.225
T_s 288 ρ 24	200	0.412	56.90	4.800	11.85	11.64
T_s 384 ρ 32	50	0.504	12.19	1.600	7.623	3.174
T_s 384 ρ 32	100	0.509	26.17	3.200	8.180	6.291
T_s 384 ρ 32	150	0.519	45.20	4.800	9.420	9.252
T_s 384 ρ 32	200	0.551	75.80	6.400	11.84	11.60
T_s 576 ρ 48	50	0.755	17.83	2.400	7.430	3.177
T_s 576 ρ 48	100	0.759	38.80	4.800	8.083	6.321
T_s 576 ρ 48	150	0.776	68.23	7.200	9.480	9.279
T_s 576 ρ 48	200	0.825	112.3	9.600	11.70	11.63
T_s 768 ρ 64	50	1.003	25.64	3.200	8.013	3.192
T_s 768 ρ 64	100	1.012	53.73	6.400	8.397	6.327
T_s 768 ρ 64	150	1.035	95.40	9.600	9.937	9.270
T_s 768 ρ 64	200	1.111	161.6	12.80	12.62	11.52

Table 5.8: CMOS 130-nm 4-bit LLR decv6 decoder processor area, power, throughput, energy-per-bit and throughput/area. To determine a decoder's hardware metrics, multiply the area, power and energy-per-bit by the required number of processors and divide the throughput/area by the required number of processors.

Code	Clock Freq. (MHz)	Area (mm^2)	Power (mW)	Through-put ($GBits/s$)	Energy per bit (pJ)	Throughput /Area ($GBits/s \cdot mm^2$)
T_s 192 ρ 16	50	0.537	44.86	0.800	56.06	1.489
T_s 192 ρ 16	100	0.547	86	1.600	53.73	2.925
T_s 192 ρ 16	150	0.599	183.7	2.400	76.53	4.008
T_s 192 ρ 16	200	0.718	318.1	3.200	99.40	4.455
T_s 288 ρ 24	50	0.807	66.76	1.200	55.63	1.487
T_s 288 ρ 24	100	0.839	149.8	2.400	62.40	2.860
T_s 288 ρ 24	150	0.903	291.2	3.600	80.90	3.987
T_s 288 ρ 24	200	1.099	495	4.800	103.1	4.368
T_s 384 ρ 32	50	1.075	96.36	1.600	60.23	1.489
T_s 384 ρ 32	100	1.108	209.1	3.200	65.33	2.888
T_s 384 ρ 32	150	1.214	419	4.800	87.30	3.954
T_s 384 ρ 32	200	na	na	na	na	na
T_s 576 ρ 48	50	1.611	143.7	2.400	59.86	1.490
T_s 576 ρ 48	100	1.680	340.3	4.800	70.90	2.857
T_s 576 ρ 48	150	1.830	648.3	7.200	90.03	3.933
T_s 576 ρ 48	200	na	na	na	na	na
T_s 768 ρ 64	50	2.145	191.2	3.200	59.76	1.492
T_s 768 ρ 64	100	2.214	434.6	6.400	67.90	2.891
T_s 768 ρ 64	150	2.453	861	9.600	89.70	3.915
T_s 768 ρ 64	200	na	na	na	na	na

Table 5.9: CMOS 180-nm 4-bit LLR decv6 decoder processor area, power, throughput, energy-per-bit and throughput/area. To determine a decoder's hardware metrics, multiply the area, power and energy-per-bit by the required number of processors and divide the throughput/area by the required number of processors.

E_b/N_0	2.0	2.1	2.2	2.3	2.4	2.5	2.6	2.7	2.8	2.9	3.0	3.5	4.0
T_s 192 ρ 16			50	32	25	21	18	16	14	12	11	8	6
T_s 288 ρ 24	50	35	26	22	18	16	14	13	11	11	10	8	6
T_s 384 ρ 32	34	30	22	18	16	14	13	12	11	10	9	7	6
T_s 576 ρ 48	28	22	18	16	14	13	12	11	10	10	9	7	6
T_s 768 ρ 64	23	19	17	14	13	11	11	10	10	9	8	7	6

Table 5.10: Number of T_s'=12 code decv6 processors required to achieve a BER of 10^{-6} for a given E_b/N_0. The results are based on bit-accurate simulations using 4-bit LLRs.

5.8 Comparison to Other LDPC Decoders

Our PN-LDPC-CC decoder architectures outperform the LDPC decoders found in the literature in terms of throughput/area and energy-per-decoded-bit. To facilitate a fair comparison between our decoder and those found in the literature, we match our competitor's BER performance, E_b/N_0 and process technology and then compare the decoders in terms of the throughput/area and the energy-per-decoded-bit. As was demonstrated in the previous section, changes in the target BER performance can result in dramatic changes in hardware requirements. Thus it is very important that the BER performance be matched for a fair comparison. In certain cases we do compare against competitors that have a higher E_b/N_0 for a give BER. In these cases we are demonstrating that our design is significantly better, given that we outperform them even when we are using a lower E_b/N_0. A number of LDPC decoder designs in the literature lack BER performance and E_b/N_0 information, making a comparison to these decoders impractical. Published decoders which lack the BER performance and E_b/N_0 values, needed for our comparisons, are thus not included in our tables. Decoders used in the first comparison table include: a 3.6 Gb/s flexible LDPC decoder [34], a 5 Gb/s shift-LDPC decoder [35] and 1.45 Gb/s LDPC decoder for the IEEE 802.16e standard [36].

The comparison results are tabulated based on the BER and process technology.

- Table 5.11 deals with CMOS 180-nm designs with E_b/N_0 ratios in the range of 3.9 to 7.
- Table 5.12 deals with CMOS 180-nm designs with E_b/N_0 ratios in the range of 2.5 to 3.0.
- Table 5.13 deals with CMOS 130-nm designs with E_b/N_0 ratios in the range of 4.5 to 5.
- Table 5.14 deals with CMOS 130-nm designs with E_b/N_0 ratios in the range of 2.5 to 3.8.
- Table 5.15 deals with CMOS 90-nm designs with E_b/N_0 ratios in the range of 2.2 to 2.4.
- Table 5.16 deals with CMOS 90-nm designs with E_b/N_0 ratios in the range of 3.2 to 3.2.

For competing LDPC-BC or LDPC-CC decoders known to us, with a BER of 10^{-6} and an E_b/N_0 in the range of 2.0 dB to 3.0 dB, we have a PN-LDPC-CC decoder that matches the E_b/N_0 and BER performance and simultaneously outperforms the competing decoder in terms of both the energy-per-decoded-information-bit and the throughput/area.

	[34]	[35]	[36]	This work (decv5)
E_b/N_0 for a BER of 10^{-6} (dB)	4.06	3.9	∼7	3.95
tech (nm)	180	180	180	180
info. throughput (Gbits/s)	3.15	4.5	1.45	3.6
power consumption (mW)			553	1120
die area (mm^2)	13.1	16.8		
core area (mm^2)	∼8.8	11.3		
synthesis cell area (mm^2)	∼6.6	7.9	∼8.6	5.8
gates (KGates)			881	593
latency (μs)				0.96
energy-per-info-bit (nJ)			0.38	**0.31**
throughput/Area (Gb/$mm^2 \cdot s$)	0.48	0.57	0.17	**0.62**
clock frequency (MHz)	290	317	100	150
operating voltage (V)	1.8			1.8
code type	BC	BC	BC	CC
block size or period	8192	8192	576	288
info bits in the block	7168	7168	288	
degrees	(4, 32)	(4, 32)		(3, 6)
LLR bit width	4		7	2
BC iterations	16	15	10	
CC processors				12

Table 5.11: Comparison of CMOS 180-nm LDPC decoders where a BER of 10^{-6} is achieved at an E_b/N_0 ratio of around 4.0. This work's power estimates are based on an E_b/N_0 ratio of 1.8 dB.

	[11]	This work (decv6)	[37]	This work (decv6)	This work (decv6)	This work (decv6)
E_b/N_0 for a BER of 10^{-6} (dB)	∼3.1	3.0	2.5	2.5	2.5	2.5
tech (nm)	160	180	180	180	180	180
info. throughput (Gbits/s)	0.5	1.6	0.75	1.6	2.4	3.2
power consumption (mW)	630	946	787	1806	2384	2926
die area (mm^2)	52.5		14.3			
core area (mm^2)	∼37		∼10			
synthesis cell area (mm^2)	∼18	6.0	∼7.5	11.5	13.4	15.5
gates (KGates)	1750	606				
latency (μs)		1.32		2.52	1.92	1.68
energy-per-info-bit (nJ)	1.26	**0.59**	1.05	1.13	0.99	**0.91**
throughput/Area (Gb/$mm^2 \cdot s$)	0.03	**0.27**	0.10	0.14	0.18	**0.21**
clock frequency (MHz)	64	100	125	100	100	100
operating voltage (V)	1.5	1.8	1.8	1.8	1.8	1.8
code type	BC	CC	BC	CC	CC	CC
block size or period	1024	192	2048	192	288	384
info bits in the block	512		1024			
degrees		(3, 6)	(3,6)	(3,6)	(3,6)	(3,6)
LLR bit width	4	4		4	4	4
BC iterations	64		10			
CC processors		11		21	16	14

Table 5.12: Comparison of LDPC decoders where a BER of 10^{-6} is achieved at an E_b/N_0 ratio of 3.0 and 2.5 in a CMOS 180-nm process. This work's power estimates are based on an E_b/N_0 ratio of 1.8 dB.

	[38]	[38]	[39]	This work (decv5)
E_b/N_0 for a BER of 10^{-6} (dB)	4.7	4.7	5+	4.47
tech (nm)	130	130	130	130
info. throughput (Gbits/s)	2.4	0.48	3.6	4.8
power consumption (mW)	1290	71	268	115
die area (mm^2)			10.24	
core area (mm^2)	7.3	7.3		
synthesis cell area (mm^2)	5.5	5.5	5.3	1.55
gates (KGates)			1150	332
latency (μs)				0.42
energy-per-info-bit (nJ)	0.54	0.15	0.074	**0.024**
throughput/Area (Gb/$mm^2 \cdot s$)	0.44	0.087	0.68	**3.1**
clock frequency (MHz)	300	59		200
operating voltage (V)	1.2	0.6	1.02	1.08
code type	BC	BC	BC	CC
block size or period	660	660	1200	288
info bits in the block	480	480	720	
degrees	(4, 15)	(4, 15)		(3, 6)
LLR bit width	4	4	6	2
iterations	15	15	8	
CC processors				7

Table 5.13: Comparison of known CMOS 130-nm LDPC decoders where a BER of 10^{-6} is achieved at an E_b/N_0 ratio around 4.5. This work's power estimates are based on an E_b/N_0 ratio of 1.8 dB.

	[40]	[41]	This work decv5	[42]	This work decv6	[10]	This work decv6
E_b/N_0 for a BER of 10^{-6} (dB)	3.6	3.8	3.63	2.5	2.5	3.0	3.0
tech (nm)	130	130	130	130	130	130	130
info. throughput (Gbits/s)	0.06	0.125	2.4	2.3	4.8	0.1	1.6
power consumption (mW)	52	76	114		910	370	128
die area (mm^2)	8.29	7.39					
core area (mm^2)	4.45	3.88		5.3			
synthesis cell area (mm^2)	∼3.3	∼2.5	2.4	∼4	6.6	∼5.3	2.8
gates (KGates)	420	444	504			1152	600
latency (μs)			0.84		0.96		1.32
energy-per-info-bit (nJ)	0.87	0.61	**0.048**		**0.19**	3.7	**0.08**
throughput/Area (Gb/$mm^2 \cdot s$)	0.018	0.050	**1.0**	0.58	**0.73**	0.02	**0.57**
clock frequency (MHz)	83	111	100	100	200	200	100
operating voltage (V)	1.2	1.2	1.08		1.08		1.08
code type	BC	BC	CC	BC	CC	CC	CC
block size or period	2304	1944	288	2082	288	127	192
info bits in the block	1152	972		1024			
degrees			(3,6)	(3,6)	(3,6)	(3,5)	(3,6)
LLR bit width	7	3-4	3	5-6	4		4
BC iterations	8	8		15		10	
CC processors			7		16		11
multi-mode	yes	yes	no		no	yes	no

Table 5.14: Comparison of known CMOS 130-nm LDPC decoders with BERs of 10^{-6} and target E_b/N_0 ratios of 3.6, 3.0 and 2.5. This work's power estimates are based on an E_b/N_0 ratio of 1.8 dB.

	[43]	This work (decv6)	This work (decv6)	This work (decv6)	[44]	This work (decv6)
$log_{10}(BER)$	-6	-6	-6	-6	-3	-3
E_b/N_0 (dB)	2.2	2.2	2.2	2.2	2.4	2.4
tech (nm)	90	90	90	90	90	90
info. throughput (Gbits/s)	0.67	2.4	3.2	4.8	1.6	4.8
power consumption (mW)	248	326	345	492		224
die area (mm^2)	3.5				5.01	
core area (mm^2)	2.98				∼3.8	
synthesis cell area (mm^2)	∼2.2	6.7	7.0	7.1	∼2.6	3.2
gates (KGates)			2417			
latency (μs)		3.36	2.64	1.76		0.80
energy-per-info-bit (nJ)	0.37	0.14	0.11	**0.10**		**0.05**
throughput/Area (Gb/$mm^2 \cdot s$)	0.30	0.36	0.46	**0.68**	0.61	**1.50**
clock frequency (MHz)	450	100	100	150	400	150
operating voltage (V)		0.9	0.9	0.9		0.9
code type	BC	CC	CC	CC		CC
block size or period	2304	288	384	384	1024	384
info bits in the block	1152				512	
degrees		(3,6)	(3,6)	(3,6)		(3,6)
LLR bit width		4	4	4	4	4
BC iterations	15					
CC processors		28	22	22		10
multi-mode		no	no	no		no

Table 5.15: Comparison of LDPC decoders in a CMOS 90-nm process, with various BERs and target E_b/N_0. This work's power estimates are based on an E_b/N_0 ratio of 1.8 dB.

	[45]	This work (decv6)
$log_{10}(BER)$	-6	-6
E_b/N_0 (dB)	3.2	3.2
tech (nm)	90	90
info. throughput (Gbits/s)	13.21	3.2
power consumption (mW)	1513	187
die area (mm^2)	3.5	
core area (mm^2)	4.97	
synthesis cell area (mm^2)	∼3.7	1.74
gates (KGates)		
latency (μs)		0.6
energy-per-info-bit (nJ)	0.115	**0.059**
throughput/Area (Gb/$mm^2 \cdot s$)	**2.66**	1.84
clock frequency (MHz)	400	200
operating voltage (V)	1.2	0.9
code type	BC	CC
block size or period	1024	192
info bits in the block	512	
degrees	(3,6)	(3,6)
LLR bit width	4	4
BC iterations		
CC processors		10
multi-mode		no

Table 5.16: Comparison of LDPC decoders in a CMOS 90-nm process, with various BERs and target E_b/N_0. This work's power estimates are based on an E_b/N_0 ratio of 1.8 dB.

5.9 PN-LDPC-CC Decoder Summary

We have presented a series of novel PN-LDPC-CC decoder designs. Compared to the state-of-the-art, these designs have excellent energy-per-decoded-bit and throughput/area metrics for target combinations BER and E_b/N_0. For a target combinations of BER and E_b/N_0, we have explored code selection, LLR bit-width and number of decoder processors, in conjunction with power and area trade-offs to determine the best selection of these variables to achieve a desired hardware metric. We introduced a number of PN-LDPC-CC designs. From the introduction of the PN-LDPC-CC decoder, decv1 to the last improved design, decv6, we have reduced energy-per-decoded-bit by 2.5 times and doubled the throughput/area.

For 10^{-6} BER performance for a given area, in a CMOS 90-nm process, it is advantageous to use the smallest of our codes, the T_s=192, ρ=16 code with a 4-bit LLR until the E_b/N_0 drops below 2.3 dB. At that point the T_s=288, ρ=24 code has better BER performance for a constrained area. The T_s=288, ρ=24 code is itself supplanted by the T_s=384, ρ=32 code at an E_b/N_0 of 2.05 dB.

For 10^{-6} BER performance for a given amount of power, in a CMOS 90-nm process, it is advantageous to use the smallest of our codes, the T_s=192, ρ=16 code with a 4-bit LLR until the E_b/N_0 drops below 2.3 dB.

Given, our set of codes, to reduce the energy-per-decoded-information-bit at a target BER of 10^{-6} and E_b/N_0 it is better to choose the largest T_s' code available.

Given our set of codes, to increase throughput/area at a target BER of 10^{-6} and E_b/N_0 it is better to choose the largest T_s' code available.

Choosing a PN-LDPC-CC, given a power or area constraint, to achieve the lowest possible BER performance and E_b/N_0 does not result in the most efficient decoder, with respect to either the energy-per-decoded-bit nor the throughput/area. Provided T_s' is not increased, it is generally advantageous, in terms of the energy-per-decoded-information-bit and throughput/area, to choose a larger code.

We have compared our work to the state-of-the-art and have shown in each case, that our PN-LDPC-CC decoders outperform the competition in terms of energy-per-decoded-information-bit and throughput/area.

For a target BER performance of 10^{-6} with an associated E_b/N_0 in the range of 2.0 to 3.0 dB, we can recommend our PN-LDPC-CC decoders over any known LDPC-BC or LDPC-CC decoders.

Chapter 6

Conclusions

We have presented a series of progressively improved parallel-node LDPC-CC encoders and decoders. For each encoder and decoder we used synthesis results for a 90-nm CMOS process to quantify the improvements with respect to the energy-per-bit, area and throughput. We have shown that the energy-per-bit, for both the encoder and decoder, has an almost linear relationship with the group period T_s'. Increasing the node-parallelization factor, for both the presented encoders and decoders, decreases the energy-per-bit and increases the throughput. The area of the encoder and decoder is strongly correlated to the code period T_s and weakly correlated to the node-parallelization factor, ρ. Predictably, the coding gain of PN-LDPC-CCs increases with T_s [1].

The improvements to the PN-LDPC-CC encoder that reduce the power consumption and/or reduce the area include:

- Encv2 4.2 - Removes the rotation switch-matrix and replaces the shift register chain with a circular buffer structure.
- Encv3 4.3 - Simplifies the encoder logic based on knowledge of how the code operates.
- Encv4 4.4 - Applies clock-gating to reduce the power consumption.
- Encv5 4.5 - Optimizes the input databus size.

The improvements to the PN-LDPC-CC decoder that reduce the power consumption and/or reduce the area include:

- Decv2 5.2 - Simplifies the hardware by removing the saturation bit.
- Decv3 5.3 - Removes the rotation switch-matrix.
- Decv4 5.4 - Applies clock-gating.
- Decv5 5.5 - Removes the reset circuitry.

Decoder Version	Incremental Power Reduction			Cumulative Power Reduction		
	Average	Minimum	Maximum	Average	Minimum	Maximum
decv2	11.1%	6.5%	22.4%	11.1%	6.5%	22.4%
decv3	13.8%	-5.6%	39.1%	24.1%	4.3%	44.7%
decv4	39.8%	9.3%	53.6%	53.2%	34.7%	65.1%
decv5	2.4%	-20.0%	16.8%	54.0%	35.7%	66.7%
decv6	21.0%	1.1%	45.5%	63.2%	46.2%	74.4%

Table 6.1: Decoder version contributions to reductions in the power consumption for our set of PN-LDPC-CC codes.

Decoder Version	Incremental Area Reduction			Cumulative Area Reduction		
	Average	Minimum	Maximum	Average	Minimum	Maximum
decv2	13.9%	11.5%	19.1%	13.9%	11.5%	19.1%
decv3	1.9%	-5.9%	19.3%	15.8%	6.6%	31.4%
decv4	13.5%	7.6%	20.1%	26.5%	18.0%	36.6%
decv5	11.5%	0.7%	17.6%	35.3%	25.8%	46.2%
decv6	18.6%	11.3%	27.6%	47.1%	40.8%	53.2%

Table 6.2: Decoder version contributions to reductions in the decoder processor area for our set of PN-LDPC-CC codes.

- Decv6 5.6 - Replaces the min-sum check-node operation with the simpler truncated min-sum check-node operation.

Each version of the decoder improves upon the previous version. Tables 6.1 and 6.2 show the contribution of each decoder version to the reduction of the power consumption and the reduction in the decoder processor area, respectively. The largest incremental reductions in the power consumption come from the use of clock-gating (decv4) and the truncated-min-sum check-node operation (decv6). Significant reductions in the power consumption are achieved by removing the saturation bit (decv2) and removing the rotation switch-matrix (decv3). Significant reductions in the decoder area are made by removing the saturation bit (decv2), clock-gating (decv4), removing the reset circuitry (decv5) and using the truncated min-sum check-node operation (decv6).

Turbo codes have low encoder complexity and LDPC-BCs have relatively low decoder complexity [46]. The PN-LDPC-CC encoders and decoders, presented herein, have both low encoder and decoder complexity. We have shown, in terms of the energy-per-bit and the throughput/area, that our PN-LDPC-CC encoders and decoders are capable of outperforming the best LDPC encoders and decoders presented in the literature (see Tables 4.4, 5.11, 5.12, 5.13, 5.13, 5.14 and 5.15). The best of our PN-LDPC-CC encoders, in a 90-nm CMOS process, has an energy-per-encoded-information-bit on the order of 0.5 pJ and a throughput up to 64 Gbits/s. For a BER of 10^{-6} at an E_b/N_0 of 2.5 dB, in a 90-nm CMOS process, the best of our PN-LDPC-CC decoders

has an energy-per-decoded-information-bit of 65 pJ and a decoded information throughput of 4.8 Gbits/s.

For future comparison purposes, we provide the ability to calculate the hardware metrics for over 1000 variations of our PN-LDPC-CC decoders. Table 5.10 specifies the number of processors required, for each code, to achieve a target BER of 10^{-6} for various E_b/N_0 ratios. Four process technologies, shown in Tables 5.6, 5.7, 5.8 and 5.9, provide the key hardware metrics: area, power, throughput, energy-per-bit and throughput/area, for our family of PN-LDPC-CCs on a per processor basis. Once the required number of processors is ascertained, the hardware metrics in the process tables are multiplied or divided by the number to processors to calculate the decoder hardware metrics. By providing these tables we hope to provide an extensive cross-process reference so that any future work on LDPC decoders can be more readily compared to our work and, in turn, to other published works.

There are a number of proposed improvements to the decoder architecture that have yet to be explored (see Appendix F). Some of the improvements to be explored include: using SRAM memory to reduce area and power, the analysis of a single highly parallelized PN-LDPC-CC with reduced parallelism implementations, reducing the amount of required memory by storing only unique values and introducing pipelining into the architectures to improve throughput. As well, this work would greatly benefit from the inclusion of termination circuitry for the encoder. PN-LDPC-CCs are still relatively new, there are many improvements that have been made to LDPC-BCs that could also be applied to PN-LDPC-CC architectures.

Bibliography

[1] Z. Chen, T. Brandon, D. Elliott, S. Bates, W. Krzymien, and B. Cockburn, "Jointly designed architecture-aware LDPC convolutional codes and high-throughput parallel encoders/decoders," IEEE Transactions on Circuits and Systems I: Regular Papers, accepted.

[2] T. Brandon, J. C. Koob, L. van den Berg, Z. Chen, A. Alimohammad, R. Swamy, J. Klaus, S. Bates, V. C. Gaudet, B. F. Cockburn, and D. G. Elliott, "A 600-Mb/s encoder and decoder for low-density parity-check convolutional codes," in Proc. of the IEEE International Symp. on Circuits and Systems (ISCAS), Seattle, WA, USA, May 2008, pp. 3090–3093.

[3] R. Swamy, S. Bates, T. L. Brandon, B. F. Cockburn, D. G. Elliott, J. C. Koob, and Z. Chen, "Design and test of a 175-Mb/s, rate-1/2 (128,3,6) low-density parity-check convolutional code encoder and decoder," IEEE Journal of Solid-State Circuits, vol. 42, no. 10, pp. 2245–2256, 2007.

[4] C. E. Shannon, "A mathematical theory of communication," Bell System Technical Journal, vol. 27, pp. 379–423 and 623–656, 1948.

[5] S. Lin and J. Daniel J. Costello, Error Control Coding. Pearson Prentice Hall, 2004.

[6] R. Gallager, "Low-density parity-check codes," IRE Transactions on Information Theory, vol. 8, no. 1, pp. 21–28, January 1962.

[7] A. Jimenez Felstrom and K. Zigangirov, "Time-varying periodic convolutional codes with low-density parity-check matrix," IEEE Transactions on Information Theory, vol. 45, no. 6, pp. 2181–2191, 1999.

[8] R. Swamy, S. Bates, and T. Brandon, "Architectures for ASIC implementations of low-density parity-check convolutional encoders and decoders,"

in Proc. of the IEEE International Symp. on Circuits and Systems (ISCAS), vol. 5, 2005, pp. 4513–4516.

[9] Z. Chen, T. Brandon, S. Bates, D. Elliott, and B. Cockburn, "Efficient implementation of low-density parity-check convolutional code encoders with built-in termination," IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 55, no. 11, pp. 3628–3640, Dec. 2008.

[10] M. Tavares, S. Kunze, E. Matus, and G. Fettweis, "Architecture and VLSI realization of a high-speed programmable decoder for LDPC convolutional codes," in International Conference on Application-Specific Systems, Architectures and Processors, 2008. ASAP 2008, July 2008, pp. 215–220.

[11] A. Blanksby and C. 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.

[12] D. MacKay and R. Neal, "Near Shannon limit performance of low density parity check codes," IET Electronics Letters, vol. 32, no. 18, pp. 1645–, Aug 1996.

[13] D. J. Costello, A. E. Pusane, S. Bates, and K. S. Zigangirov, "A comparison between LDPC block and convolutional codes," in Proc. of the IEEE Information Theory and Applications Workshop, San Diego, CA, USA, Feb. 2006.

[14] C. Schlegel and L. Perez, Trellis and Turbo Coding, 1st ed. New York: IEEE Press, 2004.

[15] J. Chen, A. Dholakia, E. Eleftheriou, M. Fossorier, and X.-Y. Hu, "Reduced-complexity decoding of LDPC codes," IEEE Transactions on Communications, vol. 53, no. 8, pp. 1288–1299, Aug. 2005.

[16] Z. Chen, R. Swamy, and S. Bates, "A new encoder implementation for low-density parity-check convolutional codes," in IEEE Northeast Workshop on Circuits and Systems, 2007. NEWCAS 2007, Aug. 2007, pp. 883–886.

[17] S. Bates and R. Swamy, "Parallel encoders for low-density parity-check convolutional codes," in Proc. of the IEEE International Symp. on Circuits and Systems (ISCAS), 2006, pp. 4–8.

[18] A. Pusane, A. Feltstrom, A. Sridharan, M. Lentmaier, K. Zigangirov, and D. Costello, "Implementation aspects of LDPC convolutional codes," IEEE Transactions on Communications, vol. 56, no. 7, pp. 1060–1069, July 2008.

[19] S. Bates, Z. Chen, L. Gunthorpe, A. Pusane, K. Zigangirov, and D. Costello, "A low-cost serial decoder architecture for low-density parity-check convolutional codes," IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 55, no. 7, pp. 1967–1976, Aug. 2008.

[20] H. Zhong and T. Zhang, "Block-LDPC: A practical LDPC coding system," IEEE Journal on Circuits and Systems I, vol. 52, no. 4, pp. 766–775, Apr. 2005, no synthesis, they present logic circuits and count gates.

[21] L. Yang, H. Liu, and C.-J. R. Shi, "Code construction and FPGA implementation of a low-error-floor multi-rate low-density parity-check code decoder," IEEE Journal on Circuits and Systems I, vol. 53, no. 4, pp. 892–904, Apr. 2006.

[22] M. Mansour and N. Shanbhag, "High-throughput LDPC decoders," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 11, no. 6, pp. 976–996, 2003.

[23] A. Alimohammad, S. F. Fard, B. F. Cockburn, and C. Schlegel, "A compact and accurate Gaussian variate generator," IEEE Journal of VLSI, vol. 16, no. 5, pp. 517–527, 2008.

[24] P. H. Bardell and W. H. McAnney, "Pseudorandom arrays for built-in tests," IEEE Transcations on Computers, vol. c-35, no. 7, pp. 653–658, 1986.

[25] A. Alimohammad, S. F. Fard, B. F. Cockburn, and C. Schlegel, "On the efficiency and accuracy of hybrid pseudo-random number generators for FPGA-based simulations," in Proc. of the IEEE International Symp. on Parallel and Distributed Processing, Apr. 2008, pp. 1–8.

[26] Z. Chen, S. Bates, D. Elliott, and T. Brandon, "Efficient encoding and termination of low-density parity-check convolutional codes," in Proc. of the IEEE Global Telecommunications Conf. (GlobeCom), San Francisco, CA, USA, 2006, pp. 1–5.

[27] "Cygwin." [Online]. Available: www.cygwin.com

[28] C. Giason, "Helium," 2005, technical document in the VLSI lab at the Univeristy of Alberta.

[29] T. Lang, E. Musoll, and J. Cortadella, "Individual flip-flops with gated clocks for low power datapaths," Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on, vol. 44, no. 6, pp. 507–516, Jun 1997.

[30] Power Compiler User Guide, Version z-2007.03 ed., Synopsys, March 2007.

[31] S. G. Clifford E. Cummings, Don Mills, "Asynchronous & synchronous reset design techniques - part deux," SNUG, 2003.

[32] L. Miles, J. Gambles, G. Maki, W. Ryan, and S. Whitaker, "An 860-Mb/s (8158,7136) low-density parity-check encoder," IEEE Journal of Solid-State Circuits, vol. 41, no. 8, pp. 1686–1691, Aug. 2006.

[33] E. Amador, R. Pacalet, and V. Rezard, "Optimum LDPC decoder: A memory architecture problem," in Design Automation Conference, 2009. DAC '09. 46th ACM/IEEE, July 2009, pp. 891–896.

[34] C. Zhang, Z. Wang, J. Sha, L. Li, and J. Lin, "Flexible LDPC decoder design for multi-Gb/s applications," IEEE Transactions on Circuits and Systems I, 2009.

[35] J. Sha, Z. Wang, M. Gao, and L. Li, "Multi-Gb/s LDPC code design and implementation," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 17, no. 2, pp. 262–268, Feb. 2009.

[36] J.-H. Hung and S.-G. Chen, "A 1.45Gb/s (576,288) LDPC decoder for 802.16e standard," in IEEE International Symposium on Signal Processing and Information Technology, 2007, Dec. 2007, pp. 916–921.

[37] M. Mansour and N. Shanbhag, "A 640-Mb/s 2048-bit programmable LDPC decoder chip," IEEE Journal of Solid-State Circuits, vol. 41, no. 3, pp. 684–698, 2006.

[38] A. Darabiha, A. Carusone, and F. Kschischang, "A 3.3-Gbps bit-serial

block-interlaced min-sum LDPC decoder in 0.13-um CMOS," in IEEE Custom Integrated Circuits Conference, 2007. CICC '07., Sept. 2007, pp. 459–462.

[39] C.-C. Lin, K.-L. Lin, H.-C. Chang, and C.-Y. Lee, "A 3.33Gb/s (1200,720) low-density parity check code decoder," in Proceedings of the 31st European Solid-State Circuits Conference, 2005. ESSCIRC 2005, Sept. 2005, pp. 211–214.

[40] 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 0.13 um CMOS process," IEEE Journal of Solid-State Circuits, vol. 43, no. 3, pp. 672–683, March 2008.

[41] X.-Y. Shih, C.-Z. Zhan, and A.-Y. Wu, "A 7.39mm^2 76mW (1944, 972) LDPC decoder chip for IEEE 802.11n applications," in Solid-State Circuits Conference, 2008. A-SSCC '08. IEEE Asian, Nov. 2008, pp. 301–304.

[42] K. Gunnam, G. Choi, and M. Yeary, "A parallel VLSI architecture for layered decoding for array LDPC codes," in 20th International Conference on VLSI Design, 2007. Held jointly with 6th International Conference on Embedded Systems, Jan. 2007, pp. 738–743.

[43] Y. Sun and J. Cavallaro, "A low-power 1-Gbps reconfigurable LDPC decoder design for multiple 4G wireless standards," in 2008 IEEE International SOC Conference, Sept. 2008, pp. 367–370.

[44] N. Onizawa, T. Ikeda, T. Hanyu, and V. Gaudet, "3.2-Gb/s 1024-b rate-1/2 LDPC decoder chip using a flooding-type update-schedule algorithm," in 50th Midwest Symposium on Circuits and Systems, 2007. MWSCAS 2007, Aug. 2007, pp. 217–220.

[45] N. Onizawa, T. Hanyu, and V. C. Gaudet, "Design of high-throughput fully parallel LDPC decoders based on wire partitioning," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Accepted for future publication, 2009.

[46] A. Goldsmith, Wireless Communications. Cambridge University Press, 2005.

[47] STMicroelectronics, CORE90GPHVT 2.2 1.00 V Standard Cell Library User Manual and Data Book May 2006.

Appendix A

Appendix - Methods

A.1 Overview

In this section we discuss the methods we use to gather our results and make comparisons.

A.2 Design Description

A challenge in designing an LDPC decoders is the implementation of the interleaver. The interleaver can be implemented in hardware in two ways. The first is with variable connections. The second is with fixed connections. Coding a fixed-connection interleaver by hand is inefficient. A more efficient technique is to use a programming language to transform the parity-check-matrix (PCM) into an HDL description of the interleaver.

Our encoders and decoders are described with a custom Perl netlisting API. Engineering effort to description of the hardware is reduced. The T_s 384 ρ 32 decv6 decoder is described in 794 lines, whereas the generated Verilog HDL takes 20473 lines. This is a line ratio of 1 to 25. The custom netlisting API generates Verilog HDL for the testbenches and the encoder/decoder designs.

A.3 Bit-Error-Rate Simulations - Matlab

Zhengang's Matlab program was modified to support parallel operation, allowing us to run bit-accurate simulations for each of the PN-LDPC-CC codes and a variety of LLR bit-widths. We collect 1000 bit-errors before terminating the simulation and calculating the BER. We modified the check-node operation to add the truncated min-sum operation (as is described in Section 5.6).

A.4 Behavioral Design Verification - VCS 2006.03

Every encoder and decoder design is verified to be bit-accurate. Zhengang's Matlab program can output fixed-point outputs for any PN-LDPC-CC PCM for both the encoding operation and decoding operation. The same input that is used for the Matlab simulations is used as stimulus for the behavioral functional simulations of the encoder and decoder designs. The output of the functional simulations are compared to the output of the Matlab simulations. If the outputs match, the designs are verified at the behavioral level.

A.5 Synthesis - Design Compiler 2007.03

Synthesis transforms behavioral HDL to a gate-level netlist. The gate-level netlist is technology dependent. We synthesize designs for four technology processes: CMOS 180 nm (TSMC), CMOS 130 nm (IBM), CMOS 90nm (ST Microelectronics) and CMOS 65 nm (ST Microelectronics). Each technology has differing timing libraries and wire-load models, but the method of synthesis is the same for all of them.

Synthesis includes the following steps/setup:

- worst case and best case libraries for timing.
- worst case libraries for power and timing estimates.
- simulation of rtl to generate switching activity for synthesis.
- compile using low effort for area optimization.
- simulation of gate-level netlist to generate switching activity for accurate power estimation.
- incremental compile with low effort for power optimization.
- incremental compile with no effort for power or area optimization to fix any remaining timing issues.
- simulation of gate-level netlist to generate switching activity for accurate power estimation.

As is mentioned above, the intermediate gate-level netlist is re-simulated to generate more accurate switching activity which can be used to generate power estimates.

The standard cell libraries used are as follows:

- TSMC 180 nm - Artisan tpz973g_230a

- IBM 130 nm - scx3 cmos8rf rvt and lvt cell libraries
- ST Micro 90 nm - CORE90GPSVT_SNPS-AVT_2.1,
  CORE90GPHVT_SNPS-AVT_2.1.a, CORX90GPSVT_SNPS-AVT_4.2, CORX90GPHVT_SNPS-AVT_4.2, CLOCK90GPSVT_SNPS-AVT_2.1, CLOCK90GPHVT_SNPS-AVT_2.1.a, DP90GPSVT_SNPS-AVT_1.2, DP90GPHVT_SNPS-AVT_1.2 and PR90M7_SNPS-AVT_3.0
- ST Micro 65 nm - CORE65LPSVT_SNPS-AVT-CDS_4.1,
  CORE65LPHVT_SNPS-AVT-CDS_4.1,
  CORX65LPSVT_SNPS-AVT-CDS_6.0,
  CORX65LPHVT_SNPS-AVT-CDS_6.0,
  CLOCK65LPSVT_SNPS-AVT-CDS_2.1,
  CLOCK65LPHVT_SNPS-AVT-CDS_2.1,
  CORL65LPSVT_SNPS-AVT-CDS_4.0,
  CORL65LPHVT_SNPS-AVT-CDS_3.0
  and PRHS65_SNPS-AVT-CDS_5.0

There are a number of synthesis options that effect the output gate-level netlist. On of the trade-off presented to the designer is that of time and the quality of the result. The designer can choose to have the synthesis tool try longer to come up with a "better" gate-level netlist. Generally a balance between synthesis time and the "quality" of the gate-level netlist is desirable. In our case, with hundreds of synthesis runs, we choose to use "low effort" settings to limit the amount of time each run would take. Even with this limitation, some of the larger codes in the decoder designs took 20 hours to complete. In some cases, the synthesis tool failed to generate a gate-level netlist that met the target timing (these data points are missing in the figures or show up as "na" in the tables).

To illustrate the variation in results that can be achieved, we took the $T_s' = 768$, $\rho = 64$ code and ran the encoder v2 (encv2) design at different "area effort" settings. On "low effort", the encoder standard cell are was 7.270e-08 m^2, whereas with "high effort" the area was 6.392e-08 m^2, a difference of 13 percent more in the case of the "low effort" option. In the case of power, the "low effort" option has a power consumption of 4.382e-02 W and the "high effort" option a power consumption of 3.580e-02 W, a difference of 22 percent more in the case of the "low effort" option. These results are specific to the encv2 encoder, no generalized statement about the magnitude of the improvement can be made, only that the synthesis tool will try for longer or use more time consuming methods to try to attain better results.

In our case for the three decoder processor synthesis, early synthesis runs would often fail, after a number of hours, to produce a netlist that met timing. As a result, we were missing many data points of interest. Changing our approach to synthesis, by first doing a "low effort" run, followed by a power optimization stage, then followed by an optimization stage with no constraints (to guarantee that the synthesis tool would meet timing for the design), yeilded a larger number of synthesis runs that met timing. The result of this approach was the area ended up being larger than the original synthesis approach. Given that it was important to us to compare decoders, we opted to use the more consistent synthesis approach even though the reported area, and often the power number, would be larger than the original approach.

In many cases we have set conservative constrains and condition for our synthesis runs. These constrains include, "low effort" options, synthesizing three processors together in one large design and using switching activity for the decoders based on an E_b/N_0 of 1.8 dB.

A.6 Power Estimation and Analysis

Three primary methods are used to generate power consumption of various circuits/architectures/designs. The first of which is the synthesis power estimation methods that consists of generating switching activity for a gate-level netlist followed by using DesignCompiler's built in power estimation. The second method is similar to the first with the back annotation of parasitics on to a P&R gate-level netlist. The third method is an actual power measurement made using test equipment.

We employ the first method.

A.6.1 Power Estimation using the Switching Activity of a Synthesized Gate-level Netlist

To generate an estimation of power using the switching activity of a synthesized gate-level netlist, we provide a clock at a target frequency, toggle the reset at the beginning of the simulation and apply Matlab generated stimulus to the data inputs. Using a functional simulator we simulate the synthesized gate-level netlist. The functional simulator saves a voltage change dump file (VCD) that contains the transitions of every node in the design for the entire simulation. Post-processing the VCD, the transitions on each node are transformed into a switch activity factor for each internal node in the gate-level netlist and is saved in a switching-activity-interface file (SAIF). The SAIF is back-annotated into the DesignCompiler synthesis tool to set the switching activity on every node in the design. Power estimation is then run using the worst case timing library for the target technology. The resulting power

estimation is used for comparison purposes.

The switching-activity is always generated using a E_b/N_0 of 1.8 dB.

Appendix B

Asynchronous Resets and Their Gate-Level Mapping

This section is for those curious about asynchronous resets in Verilog and their mapping to gate-level netlists. The first couple of Verilog code listings B.1 and B.2 fail to synthesize in Design Compiler. Listing B.1 fails to map due to the presence of the "rst" signal in the sensitivity list and the absence of an associated 'if' cause inside the associated procedural block. Listing B.2 appears to have simply confounded Design Compiler, in that Design Compiler could not find a logically-equivalent gate to map to. Verilog code listings B.3 and B.5 synthesize to the same gate-level structure, as shown in listings B.4 and B.6. Verilog code listing B.3 is a standard way to represent the desire for an asynchronous reset. Verilog code listing B.5 tries to confuse Design compiler by moving the conditional statement outside the procedural block, but Design Compiler is not fooled. The FD2QHVT cell is described as, "D Flip-Flop with 1 Phase Positive Edge Triggered Clock, Clear Active Low, Q Output Only, 4x Drive" [47].

0.05

```
assign tmpData = (rst)? 0 : dataIn[2];
// logic equivalent to asynchronous reset
always @(posedge clk or posedge rst) begin
        dataOut[2] <= tmpData;
end
```

Listing B.1: Fails to synthesize. Logical equivalent to an asynchronous reset with no internal 'if' statement.

0.05

```
// asynchronous reset with variable reset value
always @(posedge clk or posedge rst) begin
        if( rst )
                dataOut[1] <= dataIn[0];
        else
                dataOut[1] <= dataIn[1];
end
```

Listing B.2: Fails to synthesize. Asynchronous reset with variable reset value.

0.05

```
// asynchronous reset
always @(posedge clk or posedge rst) begin
        if( rst )
                dataOut[1] <= 0;
        else
                dataOut[1] <= dataIn[1];
end
```

Listing B.3: Verilog code for fflop with an asynchronous reset.

0.05

```
IVSVTX0H U3 ( .A(rst), .Z(n2) );
FD2QHVTX1 dataOut_reg_1_ ( .D(dataIn[1]), .CP(clk), .CD(n2), .Q(dataOut[1])
```

Listing B.4: Synthesized gate-level netlist for a fflop with an asynchronous reset.

0.05

```
assign tmpData = (rst)? 0 : dataIn[2];
always @(posedge clk or posedge rst) begin
        if( rst )
                dataOut[2] <= tmpData;
        else
                dataOut[2] <= tmpData;
end
```

Listing B.5: Logically equivalent to a fflop with an asynchronous reset.

0.05

```
IVSVTX0H U3 ( .A(rst), .Z(n2) );
FD2QHVTX1 dataOut_reg_2_ ( .D(dataIn[2]), .CP(clk), .CD(n2), .Q(dataOut[2])
```

Listing B.6: Synthesized gate-level netlist for a logically equivalent fflop with an asynchronous reset (listing B.5).

Appendix C

Encoder - Extras

C.1 Encv4 versus Encv3 - Correlations to Energy-Per-Bit

The following tables show the effect of clock-gating and the XOR-gate swapping, present in the encv4 encoder, on the energy-per-bit for a range of clock frequencies.

code	encv3 dynamic power	encv4 dynamic power	ratio
1152_C2_p1	5.310e-03	5.343e-03	0.994
1152_C2_p32	4.333e-03	2.155e-03	2.011
1152_C2_p48	4.947e-03	2.992e-03	1.653
1152_C2_p96	7.156e-03	5.556e-03	1.288
192_C2_p16	1.207e-03	1.007e-03	1.199
2304_C2_p64	8.664e-03	4.244e-03	2.041
288_C2_p1	1.376e-03	1.429e-03	0.963
288_C2_p24	1.748e-03	1.444e-03	1.211
288_C2_p4	1.117e-03	6.363e-04	1.755
288_C2_p8	1.205e-03	7.186e-04	1.677
384_C2_p32	2.289e-03	1.822e-03	1.256
480_C2_p8	1.738e-03	8.586e-04	2.024
576_C2_p1	2.688e-03	2.462e-03	1.092
576_C2_p12	2.136e-03	1.071e-03	1.994
576_C2_p16	2.278e-03	1.226e-03	1.858
576_C2_p24	2.623e-03	1.683e-03	1.559
576_C2_p4	2.052e-03	1.022e-03	2.008
576_C2_p48	3.462e-03	2.770e-03	1.250
576_C2_p8	2.041e-03	9.592e-04	2.128
768_C2_p1	3.565e-03	3.556e-03	1.003
768_C2_p16	2.779e-03	1.328e-03	2.093
768_C2_p32	3.353e-03	2.057e-03	1.630
768_C2_p4	2.685e-03	1.277e-03	2.103
768_C2_p64	4.629e-03	3.662e-03	1.264
768_C2_p8	2.595e-03	1.114e-03	2.329
864_C2_p24	3.341e-03	1.737e-03	1.923
960_C2_p8	3.188e-03	1.297e-03	2.458

Table C.1: encv4, encv3, comparison of dynamic power for a 250 MHz clock.

code	encv3 dynamic power	encv4 dynamic power	ratio
1152_C2_p1	1.096e-02	1.100e-02	0.996
1152_C2_p32	8.829e-03	4.450e-03	1.984
1152_C2_p48	9.994e-03	6.060e-03	1.649
1152_C2_p96	1.457e-02	1.132e-02	1.287
192_C2_p16	2.440e-03	2.940e-03	0.830
2304_C2_p64	1.763e-02	8.864e-03	1.989
288_C2_p1	2.908e-03	2.884e-03	1.008
288_C2_p24	3.580e-03	2.926e-03	1.224
288_C2_p4	2.299e-03	1.301e-03	1.767
288_C2_p8	2.496e-03	1.475e-03	1.692
384_C2_p32	4.606e-03	3.874e-03	1.189
480_C2_p8	3.548e-03	1.806e-03	1.965
576_C2_p1	5.494e-03	5.136e-03	1.070
576_C2_p12	4.368e-03	2.214e-03	1.973
576_C2_p16	4.679e-03	2.548e-03	1.836
576_C2_p24	5.332e-03	3.479e-03	1.533
576_C2_p4	4.256e-03	2.101e-03	2.026
576_C2_p48	7.278e-03	5.637e-03	1.291
576_C2_p8	4.214e-03	1.995e-03	2.112
768_C2_p1	7.295e-03	7.247e-03	1.007
768_C2_p16	5.658e-03	2.736e-03	2.068
768_C2_p32	6.842e-03	4.260e-03	1.606
768_C2_p4	5.566e-03	2.633e-03	2.114
768_C2_p64	9.458e-03	7.464e-03	1.267
768_C2_p8	5.293e-03	2.412e-03	2.194
864_C2_p24	6.851e-03	3.596e-03	1.905
960_C2_p8	6.541e-03	2.797e-03	2.339

Table C.2: encv4, encv3, comparison of dynamic power for a 500 MHz clock.

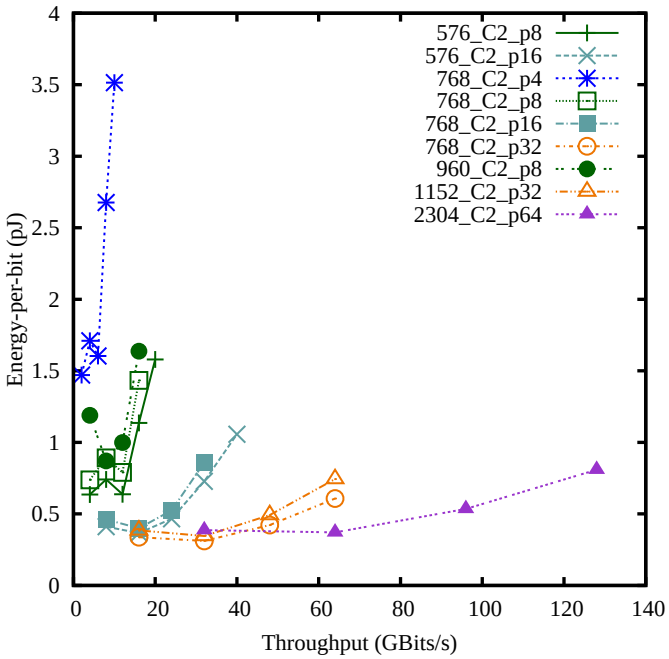

Figure C.1: Encv7, comparing the energy-per-encoded-bit versus the throughput for various codes. The datapath has been doubled, to process two independent data streams, resulting in twice the throughput.

C.2 Encoder v7 - Merged PN-LDPC-CC Encoders (Encv7)

Merging multiple PN-LDPC-CC encoders with synchronized but independent data streams and sharing control circuitry does not further reduce the energy-per-encoded-bit. We initially thought that sharing the control circuitry would amortize the power consumed in the creation of the control signals over the multiple data-paths. Figure C.1 shows the energy-per-encoded-bit versus throughput for various codes. The datapath has been doubled, resulting in twice the throughput.

Unfortunately, merging multiple encoder datapaths together does not result in reduced energy-per-encoded-bit. Figure C.2 compares the power of a design that shares the control circuitry, encv7, with a design that does not, encv5, in terms of the energy-per-bit versus the throughput for various codes. The energy-per-encoded-bit shows no improvement.

Doubling the datapath doubles the area. Figure C.3 compares the area versus the throughput for various codes.

In summary, the multiple datapath approach offers no perceivable benefit over replicating the original encoder circuitry. While it might appear that the multiple datapath approach allows the designer, for an equivalent throughput, to trade-off an increase in area for a decrease in the energy-per-encoded-bit, the original circuit can be replicated with the same result. For the given the bit-widths of the examined datapaths, sharing the control circuitry for multiple

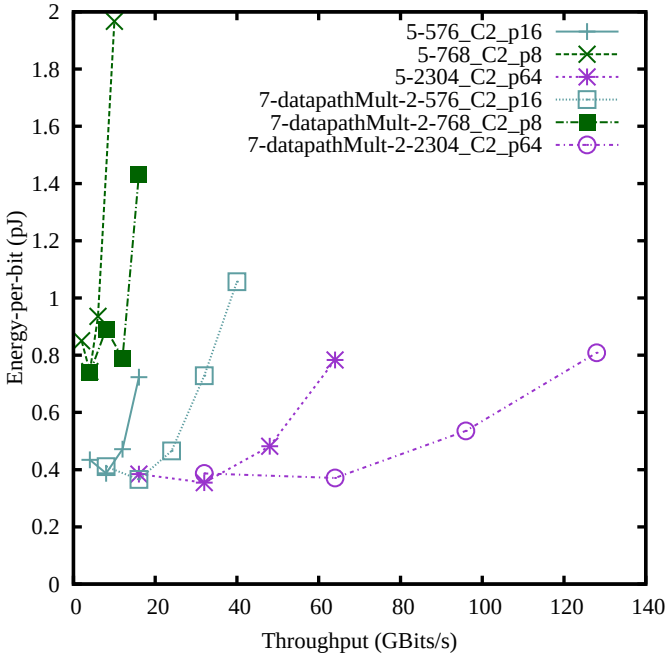

Figure C.2: Encv7 versus Encv5: comparing the energy-per-encoded-bit versus the throughput for various codes. The energy-per-encoded-bit shows no improvement.

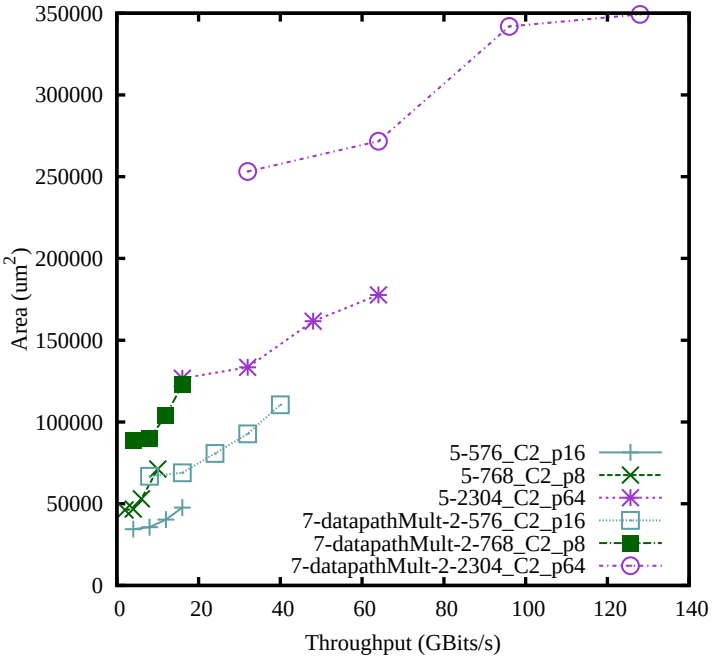

Figure C.3: Encv7 versus Encv5: comparing area versus throughput for various codes. Doubling the datapath doubles the area.

datapaths is not recommended.

Appendix D

Encoder - Future Work

D.1 De-multiplexing the Input Databus (Encv6)

In encv5 we explored increasing the databus width to reduce the power. In encv6 we will add a de-multiplexor in between the input databus and the encoder nodes. Figure D.1 shows a de-multiplexor built from simple gates. Each bit of the data input is attached to two gate inputs and a select signal is connected to two gate inputs. The output of each AND gate is attached to half the encoder nodes. The select signal, derived from the "regPhase" signals, toggles twice per T_s'. The result is that the input now can only potentially toggle inputs of a maximum of $T_s'/2$ encoder nodes per cycle instead of T_s' encoder nodes.

This concept can be expanded from 1:2 de-multiplexors to 1:N de-multiplexors. The limiting factor becomes the ratio of the consumed power of the de-multiplexors and their associated control circuitry versus the power saved by not switching the associated encoder node gates.

There is a basic relationship, that if not met, limits the practicality of this approach. Specifically, the number of gate inputs to which the input databus is attached must be greater than two. The de-multiplexor has two gate inputs for the input databus that will be switching at the same rate as the input databus. In our case, the encoder nodes have three gate inputs. This means

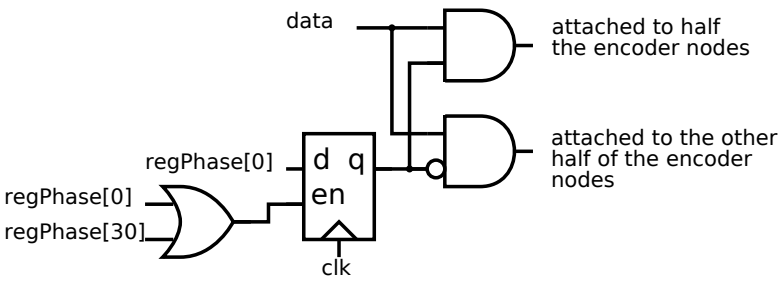

Figure D.1: A 1:2 de-multiplexor built from simple gates for a T_s'=60 code.

we can reduce the power associated with the gate input capacitance of the dynamic power of the input databus by at most 1/3. Once we add the control circuitry, the reduction in power will be even less.

In the unlikely case where the input data is all ones, the switching activity is higher than in the previously presented encoders. The return to zero nature of the phase gating results in the inputs toggling twice per code period.

While the dynamic power associated gate capacitance is reduced using this technique, the dynamic power associated with the wiring capacitance remains almost unchanged. Doubling the number of wires results in those wires switching at half the original rate.

Appendix E

Encoder - Extra Graphs with Grouping by Node-Parallelization

Here we present the encoder energy-per-bit versus throughput and area versus throughput graphs while grouping different codes in terms of the node-parallelization factor. These graphs were not included in the main document. By grouping different codes in terms of the node-parallelization factor we can more clearly see the impact of increasing the node-parallelization.

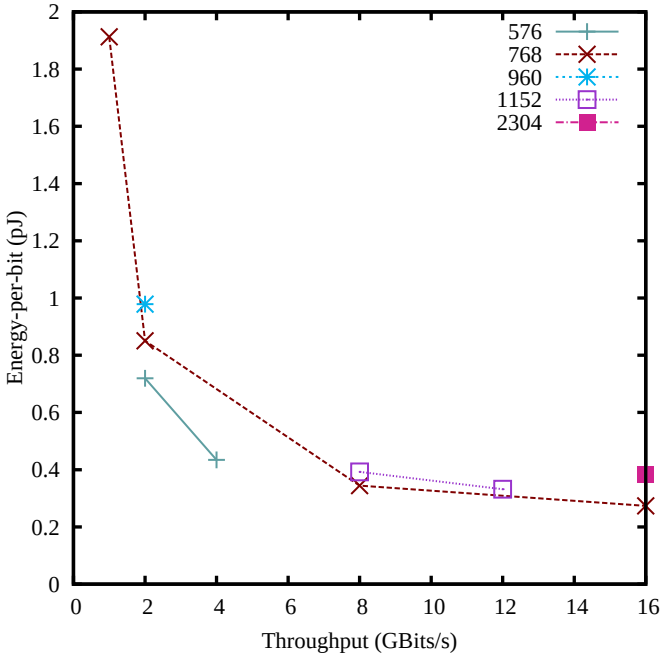

Figure E.1: Encv5 energy-per-bit versus the information throughput for the PN-LDPC-CCs. The higher the ρ, the higher the information throughput and the lower the energy-per-encoded-bit. Code lengths (T_s) and parallel-node (ρ) values are as specified: for T_s=192 $\rho \in \{16\}$, for T_s=288 $\rho \in \{1,4,8,24\}$, for T_s=384 $\rho \in \{32\}$, for T_s=480 $\rho \in \{8\}$, for T_s=576 $\rho \in \{1,4,8,12,16,24,48\}$, for T_s=768 $\rho \in \{1,4,8,16,32,64\}$, for T_s=864 $\rho \in \{24\}$, for T_s=960 $\rho \in \{8\}$, for T_s=1152 $\rho \in \{32,48\}$, and for T_s=2304 $\rho \in \{64\}$. A clock frequency of 250 MHz is used.

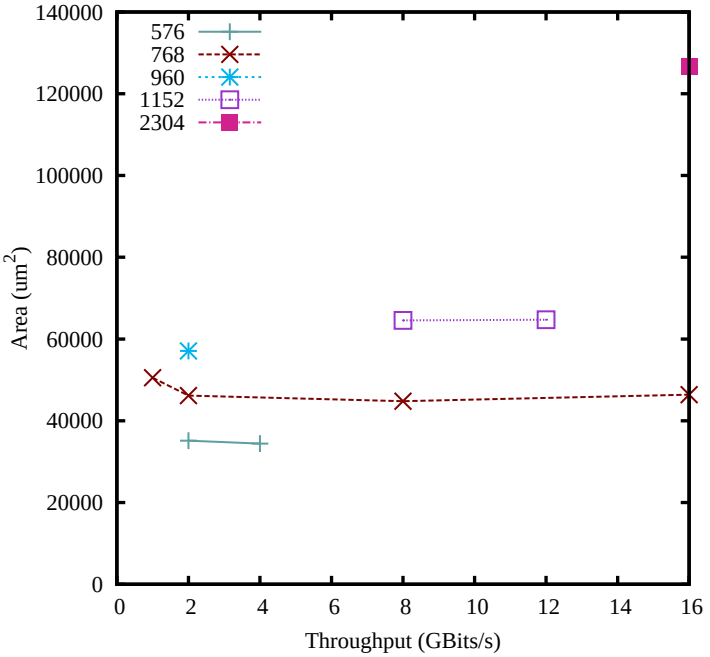

Figure E.2: Encv5 area versus the information throughput, for the PN-LDPC-CCs. Code lengths (T_s) and parallel-node (ρ) values are as specified: for T_s=192 $\rho \in \{16\}$, for T_s=288 $\rho \in \{1,4,8,24\}$, for T_s=384 $\rho \in \{32\}$, for T_s=480 $\rho \in \{8\}$, for T_s=576 $\rho \in \{1,4,8,12,16,24,48\}$, for T_s=768 $\rho \in \{1,4,8,16,32,64\}$, for T_s=864 $\rho \in \{24\}$, for T_s=960 $\rho \in \{8\}$, for T_s=1152 $\rho \in \{32,48\}$, and for T_s=2304 $\rho \in \{64\}$. A clock frequency of 250 MHz is used.

Appendix F

PN-LDPC-CC Decoder Future Work

The work that we presented in this thesis is the first generation of PN-LDPC-CC architectures. However, there are a number of ideas generated during the duration of this work that warrant further exploration. For example there are a number of improvements that have been made to LDPC-BC architectures that could be applied to our PN-LDPC-CC architectures. The following sections give a quick summary of some of the possible improvements and the expected benefits.

F.1 Removing Reset Circuitry While Maintaining BER Performance for Short Streams

In Section 5.5 we eliminated the reset circuitry by replacing the function of the saturation bit with maximum magnitude and additional control circuitry. We propose to add multiplexors to the variable-node outputs to achieve the same BER performance as the "saturation-bit" decv1 implementation without the use of a saturation bit. As was mentioned previously, LDPC-CC are deterministic. Thus we can predetermine when the outputs of the variable-node are "saturated" and when they carry actual data.

The addition of multiplexors at the outputs of the variable-node is expected to result in more slightly higher power consumption. The addition of the variable-node output multiplexors is not expected to impact the overall trend that the higher the ρ, the higher the throughput and the lower the energy-per-decoded-bit.

Compared to decv6, the addition of variable-node output multiplexors is expected to result in slightly higher area consumption.

F.2 Using Synchronous Memories

The register-memory represents 87% of the total decoder processor area. Reducing the area of the memory would thus significantly impact the total area of the decoder processor.

Replacing register-memories with synchronous memories would require the number of memory banks to double to avoid memory access conflicts. In the current register-memory architecture, a read and write operation is performed in a single clock cycle. With synchronous memories only a single read or a single write can be done in one clock cycle. To preserve single-cycle throughput, we can double the number of memory banks on the condition that we do not need to both read and write to any given memory bank in any specific cycle.

If all memories were implemented using purely synchronous memories, we would see an increase in the number of memory banks. As a result, we would need to approximately double the number of memory banks. Likewise, changing from dual-port to single-port port memories would require each dual-port memory be replaced with two single-port memories and thus approximately double the required number of memory banks. Using only single-port memories would require that we again double the number of memory banks to avoid collisions between check-node and variable-node accesses.

Given that clock-gating was effective at reducing the area and power consumption, it would be a very interesting to see if using actual memories reduces the area, power and/or throughput.

F.3 Pipelined Processor

The maximum clock frequency of previous decoder is limited by the signal path that runs from the memory to the check-node, through the variable-node and back to the memory. Examining the previous decoder timing reports reveals that significant time is spent in both the check-node and variable-node. By adding a pipeline stage between the check-node and variable-node, this bottleneck could be alleviated. The addition of a pipeline stage would require that the variable-node operation occur 1 cycle later. As a result the next processor would need to start one cycle later as well. As values in "memory" will need to be stored an extra cycle, it is very likely that a collision would occur. By extending the group period T_s' by one cycle and expanding the "memory" to compensate, this should allow the delayed variable-node operation. The trade-offs between area, power and throughput would need to be examined carefully.

F.4 Memory Reduction

As pointed out in [42], the output of the check-node produces K outputs of which only two are unique and only these two check-node values need to be stored. By storing the check-node input that had the minimum value and also storing the two unique "min" values, then the correct "min" value could be selected for the variable-node operations. Implementation of this idea may reduce area as well as power consumption.

F.5 Re-analyzed using a Single Code with High Parallelism

It is possible to analyze PN-LDPC-CCs with different degrees of parallelism. It would be interesting to take a single code with large parallelism and re-analyze it with lower degrees of parallelism. As increased parallelism simply avoids collisions in read/write accesses, there should be no issue in treating a code with higher parallelism as a code with lower parallelism.

F.6 DRAM

The use of DRAM may possibly reduce power consumption and required area. DRAM consumes very little space, however, to achieve the highest density, a specialized DRAM IC process is required. A DRAM cell implemented in a logic process, may be an acceptable alternative. Given the periodic access to the memory, the DRAM would not need to be refreshed. Additionally, give that we eliminated the need for reset circuitry, the DRAM would not need to be initialized nor reset between streams of data.

F.7 Latches

Latches typically consume less area and power than flip-flops. By replacing the flip-flops in the design a reduction in power and area may be achieved. Further following this concept, the use of dynamic latches may yield ever better results.

F.8 SIMD

With the increased node-parallelism, the PN-LDPC-CC may take advantage of vector processors. As the ρ increases, more operations are capable of being processed in parallel. This will allow SIMD type architectures to efficiently execute the operations required to encode and decoder the PN-LDPC-CCs.

F.9 Higher Rate PN-LDPC-CCs

As E_b/N_0 is increased beyond a certain point, higher-rate codes of the same length will begin to achieve better hardware performance compared to the lower-rate codes. We believe it is possible to design high-rate PN-LDPC-CCs [1] while retaining a number of the benefits of the PN-LDPC-CC encoder and decoder architectures.