View metadata, citation and similar papers at core.ac.uk brought to you by CORE
provided by Texas A&amp;M Repository

DATA INTEGRITY FOR ON-CHIP INTERCONNECTS

A Dissertation

by

ROHIT SINGHAL

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

May 2007

Major Subject: Computer Science

DATA INTEGRITY FOR ON-CHIP INTERCONNECTS

A Dissertation

by

ROHIT SINGHAL

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Approved by:

Co-Chairs of Committee,	Rabi Mahapatra
	Gwan Choi
Committee Members,	Duncan M. Walker
	Eun Jung Kim
	Jiang Hu
	Madhav Pappu
Head of Department,	Valerie Taylor

May 2007

Major Subject: Computer Science

ABSTRACT

Data Integrity for On-Chip Interconnects. (May 2007)

Rohit Singhal, B. Tech., Indian Institute of Technology;
M.S., Texas A&M University
Co–Chairs of Advisory Committee: Dr. Rabi Mahapatra
Dr. Gwan Choi

With shrinking feature size and growing integration density in the Deep Sub-Micron (DSM) technologies, the global buses are fast becoming the “weakest-links” in VLSI design. They have large delays and are error-prone. Especially, in system-on-chip (SoC) designs, where parallel interconnects run over large distances, they pose difficult research and design problems. This work presents an approach for evaluating the data carrying capacity of such wires. The method treats the delay and reliability in interconnects from an information theoretic perspective. The results point to an optimal frequency of operation for a given bus dimension for maximum *data transfer rate*. Moreover, this optimal frequency is higher than that achieved by present day designs which accommodate the worst case delays.

This work also proposes several novel ways to approach this optimal data transfer rate in practical designs.From the analysis of signal propagation delay in long wires, it is seen that the signal delay distribution has a long tail, meaning that most signals arrive at the output much faster than the worst case delay. Using communication-theory, these “good” signals arriving early can be used to predict/correct the “few” signals that arrive late. In addition to this correction based on prediction, the approaches use coding techniques to eliminate high delay cases to generate a higher transmission rate.

The work also extends communication theoretic approaches to other areas of VLSI design. Parity groups are generated based on low output delay correlation to add redundancy in combinatorial circuits. This redundancy is used to increase the frequency of operation and/or reduce the energy consumption while improving the overall reliability of the circuit.

To My Parents, Anuja, Richa, Anil, Ravi and Shubha.

ACKNOWLEDGMENTS

This research work would not have been possible without the support and guidance of my advisors Dr. Gwan Choi and Dr. Rabi Mahapatra. Almost every day, they went out of their way to help me in all aspects of my life and work. There is so much that I've learnt from them and their experiences.

The valuable inputs and advice that I got from my committee members, Dr. Walker, Dr. Kim, Dr. Hu, and Dr. Pappu, throughout the duration of my research is very much appreciated. The students of the computer engineering group were equally helpful to my research effort. In particular I'd like to mention Rupak Samanta, Pankaj Bhagawat, Euncheol Kim, Sanghoan Chang, Kiran Gunnam, and Frank Wang. I really enjoyed working with them.

I would like to thank all my friends who made my stay at Texas A&M very enjoyable. In particular I'd like to thank Waqar, Dali, Veera, Upali, Bhoj, Percy, Gaurav, Pingla, Sudha, Rahul for their constant support and affection. Outside the university, Tiwary, Amina, Michael, and Gullu gave me a lot of pleasant memories.

I'd like to thank my family for giving me unconditional love during the course of this work. My parents, my sister Richa, and brother-in-law Anil were always with me. Ravi and Shubha were like family to me in times of joy and sorrow, never leaving me alone. I can never forget the sacrifices that my lovely wife Anuja made to help make this happen. I couldn't have done this without her. She gave me the strength that I needed to walk, until the last mile . . . and more.

And last, but not the least, I'd like to thank Dr. Ben Zoghi. Without his help and guidance, I'd have quit before even hitting the half-way mark!

Thank you all!

TABLE OF CONTENTS

CHAPTER Page

I INTRODUCTION 1

- A. Motivation and Objectives 2
- B. Contributions 5

II A LINEAR TIME INVARIANT MODEL FOR A WIDE BUS . 7

- A. Resistance, Capacitance and Inductance 7
- B. The Mathematical Model 8
- C. Impulse, Step and Frequency Response 10
- D. Delay Distribution 15

III INFORMATION THEORETIC CAPACITY OF ON-CHIP BUSES 19

- A. Channel Capacity of Binary Symmetric Channels 19
  - 1. Self-Information 19
  - 2. Entropy 20
  - 3. Conditional Entropy 20
  - 4. Capacity of Memory-less BSC Channels 21
  - 5. Capacity of BSC with Memory 22
- B. Bus Capacity in the Presence of Capacitive Coupling . . . 22
  - 1. Interconnect States and Capacity 27
  - 2. The Mathematical Model and Calculations 29
- C. Bus Capacity in the Presence of Capacitive and Inductive Coupling 36

IV CODES FOR ON-CHIP BUSES 41

- A. Previous Work 41
- B. Bus Properties 45
  - 1. Delay 45
  - 2. Energy 46
  - 3. Reliability 50
- C. Code Construction 52
  - 1. Comparison with Existing Codes and Capacity 58

V LDPC CODES 63

CHAPTER Page

- A. LDPC Decoding Process 65
  - 1. The Variable Node 66
  - 2. The Check Node 66
- B. LDPC Decoder Design Issues 67
  - 1. Routing Complexity 67
  - 2. Quantization 67
    - a. Performance of LDPC Codes 67
- C. Decoder Architecture for On-chip Interconnects 72
  - 1. Bubble Sort Algorithm 73
  - 2. The Switching Cell and Computing Nodes 73
  - 3. The Decoder Architecture 76

VI OTHER APPLICATIONS OF CODING - COMBINATORIAL CIRCUITS 79

- A. Error Correction Coding 84
  - 1. Generating Parity Groups 84
  - 2. Proposed Architecture 85
- B. Simulations and Benchmark Results 86
- C. Comparison Against Existing Methods 97

VII CONCLUSION 99

- A. Future Work 101

REFERENCES 102

VITA 110

LIST OF TABLES

TABLE Page

I The Interconnect States 37

II The Code Construction [15] for a 4-Bit Wide Bus 42

III The Data Per Wire Rates for Different Codes 43

IV Wire Requirement for Various Codes as a Function of Information Width 54

V An Example 4B5W Code for 3C Elimination 55

VI An Example 4B11W Code for 2C Elimination and Single Error Correction 56

VII Comparison of Speedups and Overheads of Different Codes for a 4-Bit Bus 58

VIII The ISCAS'85 Circuits 86

IX The Delay of a NAND Gate for Different Input Combinations 87

X The Clock Frequencies for Different Error Probabilities. 95

XI The Prohibited Output Groups Based on Delay Correlations. 95

LIST OF FIGURES

FIGURE Page

1 Schematics of buffer insertion and coding schemes. 3

2 The impulse response of a 1 mm long wire in a 8-bit wide bus. 11

3 The step response of a 1 mm long wire in a 8-bit wide bus. 12

4 The frequency response of a 1 mm long wire in a 8-bit wide bus. . . . 13

5 The frequency response of a wire's input compared to the coupling response from the neighbors. 14

6 The waveform for a 8 bit wide 1 mm long bus 16

7 The step response of one wire in a 8-bit wide bus when all inputs are active. 17

8 The signal delay distribution of an 8-bit wide 1 mm long bus. 18

9 The worst case signal delay of an 8-bit wide bus 18

10 The RC model for a bus 23

11 The waveform for a 4 bit wide bus 24

12 Histogram of the output received from a 0.5mm long, 16-bit wide bus running at 10GHz. 25

13 Histogram of the signal propagation delay of a 1mm long, 16-bit wide bus. 26

14 The channel capacity as a function of wire length and clock speed. . 30

15 The channel capacity in bits per period as a function of wire length. . 31

16 The channel capacity in bits per second as a function of wire length. 32

17 The channel capacity in bits per period as a function of clock speed. 33

FIGURE Page

18 The channel capacity in bits per second as a function of clock speed. 34

19 The capacity of 8 bit wide bus as a function of clock frequency and length 38

20 The data rate limits of 8-bit wide bus as a function of clock frequency and length 40

21 The comparison of various codes against capacity 44

22 The signal delay distribution of a 1 mm long “Shielded” bus. 48

23 The signal delay distribution of a 1 mm long bus with “AND” code. . 49

24 The output voltage distribution of an 8 bit wide 1 mm long bus. . . . 51

25 The signal delay distribution of a 1 mm long bus with 4B11B code. . 57

26 The comparison of speedups/overhead of the various codes 59

27 The comparison of energy consumption of the various codes 60

28 Schematic of the LDPC decoder. 64

29 The comparison of various implementations of the LDPC decoder with varying ϵ_{vn} at 2.3dB SNR AWGN channel. 68

30 The comparison of various implementations of the LDPC decoder with varying ϵ_{cn} at 2.3dB SNR AWGN channel. 69

31 The comparison of various implementations of the LDPC decoder with varying ϵ_{vn} at 10^{-2} BER BSC channel. 70

32 The comparison of various implementations of the LDPC decoder with varying ϵ_{cn} at 10^{-2} BER BSC channel. 71

33 A schematic for the Bubble Sort Algorithm. 74

34 The schematic for the switching cell. 75

35 The schematic for the VN. 77

FIGURE Page

36 The schematic for a single pipeline stage of the decoder. 78

37 TTF v/s clock for a 800 MHz Duron Processor. 82

38 The schematic of a error detection system. 85

39 Delay distribution of a 2-input NAND gate loaded by another 2-input NAND gate. 87

40 Delay distribution of a 2-input NOR gate loaded by another 2-input NOR gate. 88

41 Delay distribution of an inverter loaded by another inverter. 89

42 Delay distribution of output 4 of c432 circuit. 91

43 The frequencies of operation of the ISCAS'85 circuits. 92

44 The gain in frequency due to error correction coding in the ISCAS'85 circuits. 94

45 A layout for error detection in c499. 96

CHAPTER I

INTRODUCTION

Recent advances in the System-on-Chip (SoC) technology have given rise to complex communication systems between different modules on a chip. Even though the problems arising due to the size and magnitude of the logic have been alleviated by Moore's law, the high speed data transfer over long distances poses difficult research problems. Moreover, the emergence of networks-on-chips (NoC) as the communication infrastructure alternative to bus-based communication in SoC has presented the SoC design community with numerous challenges. Designing energy efficient, high performance, reliable on-chip communication systems requires the formulation of strategies to rectify operational glitches.

The long interconnecting wires in such NoC are traditionally modeled as *RC* networks [1, 2]. More recently, with the high speed of switching, the inductive effects have become increasingly significant [3–6]. The most important problem that has been identified in long interconnects is of *Cross-talk*, where adjacent signals interfere with each other resulting in not only errors and large delays, but also high energy consumption. In addition to cross-talk, the signal flowing through a wire is also affected by *Power Noise* and *Process Variations* [7]. Process variations result in imperfect wire dimensions, while power noise results in imperfect V_{dd} and V_{ss} values. Both result in errors at the receiver. Other external noises like thermal noise, electromagnetic noise, slot noise and alpha-particle induced noise also adversely affect the signals.

The major factors that influence the NoC designer's decisions are

---

This thesis follows the style of *IEEE Transactions on VLSI Systems.*

1. Data and signal reliability
2. Signal propagation delays and delay variations.
3. Energy consumption

A. Motivation and Objectives

The design of low power systems has highlighted the contribution of interconnect power, which is upto 50% of total system power [8]. To reduce the interconnect energy consumption, voltage scaling schemes are being used, which in turn reduce the circuit's noise margin. The decrease in noise margin makes the interconnect less immune to errors during transmission. On the other hand, the traditional techniques for noise mitigation and speedup, like buffer insertion [9–11] shown in fig 1 (a), have a very high energy consumption. Other techniques like variable cycle design [12] and wave-piepelining [13] have been explored without significant success. Combining low-power strategies with data reliability in on-chip interconnects has become a daunting task for the designers.

As identified in [14], there are two approaches to address these reliability concerns

1. Noise mitigation, and
2. Tolerance

In [15] it is shown, using simulations, that coding is a better alternative to buffer insertion as it not only reduces wire-delays but also reduces the power consumption for data transmission and improves reliability. Another important advantage of coding over repeater/buffer insertion that is missing in [15], is that buffers usually limit the data-flow in one direction, while coding allows the buses to operate in both directions as shown in fig. 1.

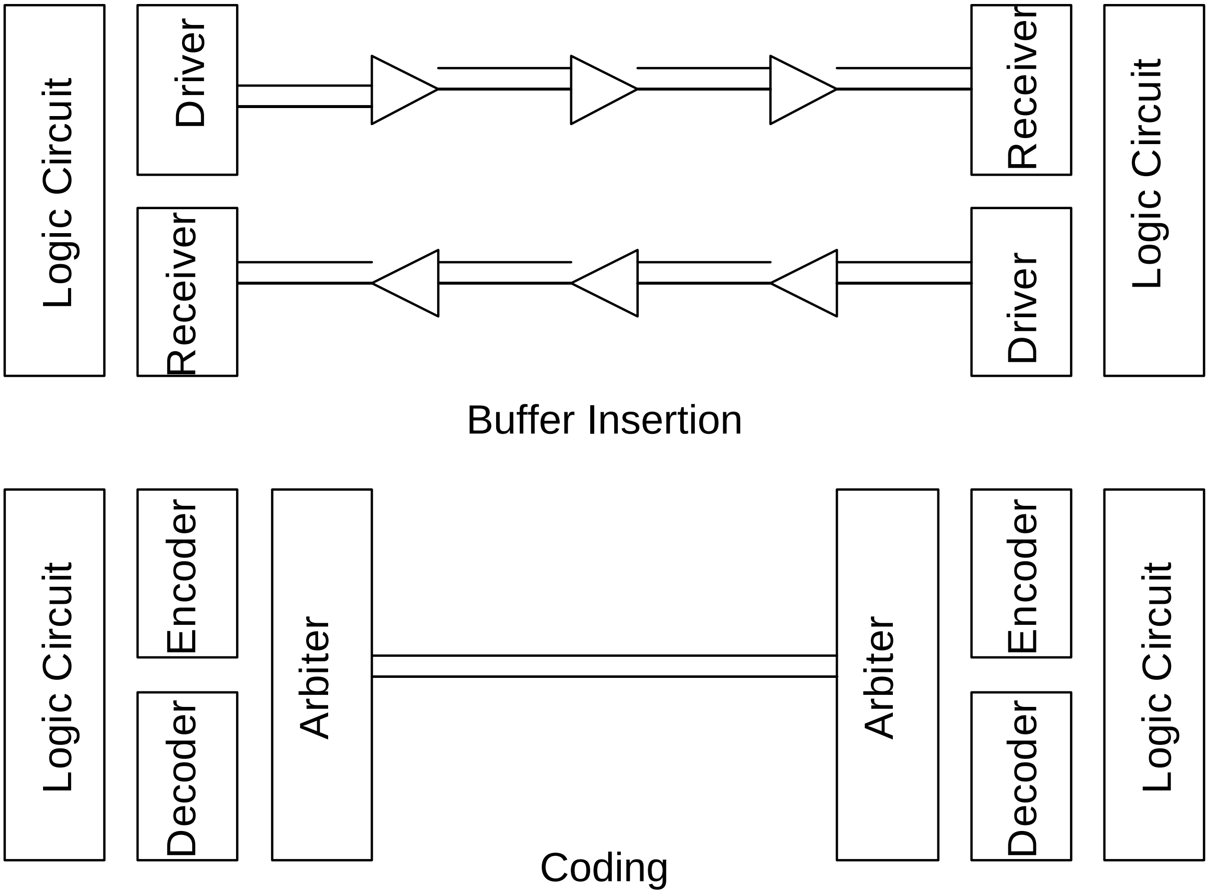
Logic Circuit
Driver
Receiver
Receiver
Driver
Logic Circuit
Buffer Insertion
Logic Circuit
Encoder
Decoder
Arbiter
Arbiter
Encoder
Decoder
Logic Circuit
Coding

Fig. 1. Schematics of buffer insertion and coding schemes.

A lot of research [15–24], has been done to make use of coding for on-chip interconnects. A coding technique to minimize the inductive crosstalk in *off-chip interconnects* has been presented in [25]. The coding techniques however, improve just one of the factors out of reliability, energy and speed. The problem that remains is to design a coding scheme that accomplishes all three tasks of low energy, low delay and low error rates.

The codes that aim to aggressively achieve speedup [16,17,20], ignore energy and reliability considerations. Some others improve the energy consumption [22–24, 26] while leaving out the other two factors. Still others, just improve the reliability by making use of either error detecting codes (EDC) or error correcting codes (ECC).

VLSI self-checking circuits use error detection codes such as parity, two-rail and other unidirectional EDCs (m-out-of-n and Berger codes) [27]. Since crosstalk is bidirectional [28], these codes would not be sufficient. The authors in [29] make a case for the use of Hamming code [30] on on-chip data buses, highlighting its capability to handle single, double errors, its low complexity and flexibility as a purely detecting code or a purely correcting code.

For on-chip networks, authors in [31] suggest using Hamming code for error detection and in [32] cyclic redundancy check (CRC) [33] is used to detect errors over every hop. Retransmissions are then used to correct the detected errors. When it comes to using ECCs in a design, [29] compares the energy efficiency of forward error correction (FEC) versus error detection and retransmission for on-chip data buses. The reported results indicate that FEC is energy inefficient in described applications.

However, the overhead for FEC is expected to subside in emerging NoCs that span large devices using increasing number of hops and complex buffering/ signaling structures. Use of FEC may be cost inefficient when the size of the network is small and the cost of the FEC codecs is high. But as the network size increases and error rates increase, error detection and retransmission schemes become unacceptable with respect to energy use and latency.

The aim of this research effort is three fold,

- Estimate the data handling limits of long on-chip interconnects and buses.
- Design communication theoretic systems and schemes that achieve this limit on interconnects.
- Extend communication theoretic schemes to improve speed and reliability in other aspects of VLSI design.

B. Contributions

Two distinct approaches are employed to calculate the capacity of interconnects. The first approach models a single interconnect as a linear time invariant (LTI) system. This is discussed more in Ch. II. The impulse response, step response and frequency response of this system are studied to determine the optimal data transfer frequency given appropriate pulse shapes for signal transmission.

The second approach makes use of the Shannon's capacity theorem like [34, 35], also discussed in Ch. III. The data limits derived here are for the wire component of the bus only. The approaches discussed can be used in conjunction with the existing techniques, i.e., buffer insertion, wire tapering, etc. to provide even better results.

Results for the data limits of a 1 mm long, 8-bit wide bus in $0.1\mu m$ technology are presented. The worst case delay for this bus configuration from end-to-end is 57 ps. While the traditional design methods, that rely on acoomodating the worst case delay, would have resulted in a maximum data transfer rate of 17Gbps (1/57ps), a capacity of about $\approx$ 30Gbps is established using both techniques.

Codes that achieve this limit even with significant gains in energy consumption and reliability are also presented herein. The example codes “4B5W” and “4B11W” are discussed in Ch. IV. The names 4B5W and 4B11W mean that the 4 bit information is spread on-to 5 and 11 wires respectively to achieve reliability, speedup and energy savings. The codes achieve up-to 2.7x speedup and 14% energy savings over a no-code system. A more elaborate coding scheme, the Low Density Parity Check (LDPC) Code, is discussed in Ch. V. A decoder design is also presented that can be applied to on-chip interconnects.

This work also aims to open up the communication theoretic solutions to other aspects of chip design. Ch. VI discusses the use of coding theory on combinatorial circuits to have upto 4x speedup in the clock frequencies.

CHAPTER II

A LINEAR TIME INVARIANT MODEL FOR A WIDE BUS

In the modern VLSI processes, there are multiple metal layers that can be utilized for laying out the interconnects. According to [7], the dimensions of a global bus in a $0.1\mu m$ technology are as follows

$$\begin{aligned} \rho &= 2.2 \times 10^{-8} \pm 30\% \text{ Ohm-m}; \\ d, w &= 237 \times 10^{-9} \pm 20\% \text{ m}; \\ t, h &= 498 \times 10^{-9} \pm 15\% \text{ m}; \end{aligned} \tag{2.1}$$

where ρ, w, d, t, h are the resistivity of copper, wire width, separation between two wires, the wire height, and the height of the wire above the substrate respectively.

A. Resistance, Capacitance and Inductance

Each global wire possesses a resistance R, a capacitance C and an inductance L. These parameters determine the time and energy required to change the signal on the wire from one voltage level to another. The resistance R can be calculated simply by the equation,

$$R = \frac{\rho L}{wt}; \tag{2.2}$$

where L is the length of the interconnect.

The capacitance C in a long wire is two-fold. There is a wire-to-substrate capacitance C_g, and then there is an inter-wire capacitance called the coupling capacitance C_c. Traditionally, the C_c has been negligible compared to the C_g, but as the fabrication technologies advance into deep sub-micron (DSM) region, the inter-wire capacitance C_c, not only becomes significant, it becomes dominant compared to the wire-to-substrate capacitance C_g.

$$\begin{aligned} C &= C_g + C_c; \\ C_g &= \frac{\epsilon L w}{h}; \\ C_c &= \frac{2\epsilon L t}{d}. \end{aligned} \tag{2.3}$$

where ϵ is the permittivity of the oxide between wires.

As more advances are made into the DSM region, and as the speed of signaling approaches multi-GigaHertz, the inductive effects become increasingly significant [3–6]. Each interconnecting wire has a Self-Inductance L_s and a Mutual-Inductance L_m, with its neighbors given by

$$L_s = 2 \times 10^{-7} L \left(0.5 + \log\left(\frac{2L}{w+t} \right) + 0.11 \frac{w+t}{L} \right), \tag{2.4}$$

$$L_m = 2 \times 10^{-7} L \left(\log\left(\frac{d+w}{d} \right) \right). \tag{2.5}$$

All the equations above assume that the wires are of constant width, height and thickness throughout their length. This assumption is reasonable in light of the fact that the wires being considered are expected to carry bidirectional data. Techniques like wire-tapering etc. will result in uneven characteristics in the two directions.

B. The Mathematical Model

The small section of an n-bit wide bus can be described as a multi-input-multi-output (MIMO) linear time invariant (LTI) system [16, 34, 35]. The term MIMO comes from the fact that each wire has a separate driver (input) and a distinct receiver (output). The transfer function of this MIMO system can be described in Laplace domain as an $n \times n$ matrix $F(s)$ given as

$$F(s) = \left(I + L(s)C(s)\right)^{-1}, \tag{2.6}$$

where, the size of I, $L(s)$ and $C(s)$ is also $n \times n$. $L(s)$ defines the inductance and resistance while $C(s)$ defines the capacitance of the wires as follows,

$$\begin{aligned}
L(s) &= \begin{bmatrix} A & B & 0 & \dots & 0 & 0 \\ D & A & B & \dots & 0 & 0 \\ 0 & D & A & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & A & B \\ 0 & 0 & 0 & \dots & D & A \end{bmatrix}, && (2.7) \\
A &= R + sL_s, && (2.8) \\
B &= sL_m, && (2.9) \\
D &= -sL_m. && (2.10)
\end{aligned}$$

$$C(s) = \begin{bmatrix} Y & Z & 0 & \dots & 0 & 0 \\ Z & X & Z & \dots & 0 & 0 \\ 0 & Z & X & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & X & Z \\ 0 & 0 & 0 & \dots & Z & Y \end{bmatrix}, \tag{2.11}$$

$$X = s(2C_c + C_g), \tag{2.12}$$

$$Y = s(C_g + C_c), \tag{2.13}$$

$$Z = -sC_c. \tag{2.14}$$

C. Impulse, Step and Frequency Response

In order to study the input response of a wire, based on the transfer function $F(s)$ described above, the input of a single wire is excited, while the inputs of the other wires are kept constant. The characteristics that are studied include an impluse response, a step response and a frequency response shown in figs. 2, 3 and 4 respectively.

- The *Impulse Response* in fig. 2 describes the voltage at the receiver of a single wire when the input of that wire rises sharply to infinity and falls back to zero. It is seen that an emf of 3.5×10^{10} V develops at the output. This is enough to burn the chip, therefore it is recommended that the signals transmitted on the wires should not contain sharp edges. In practical systems, it is usually impossible to generate these sharp edges, due to the limitations in the driver capacities.

- The *Step Response* in fig. 3 describes the voltage at the receiver of a single wire when the input of that wire rises sharply to voltage level 1. It is seen that it takes

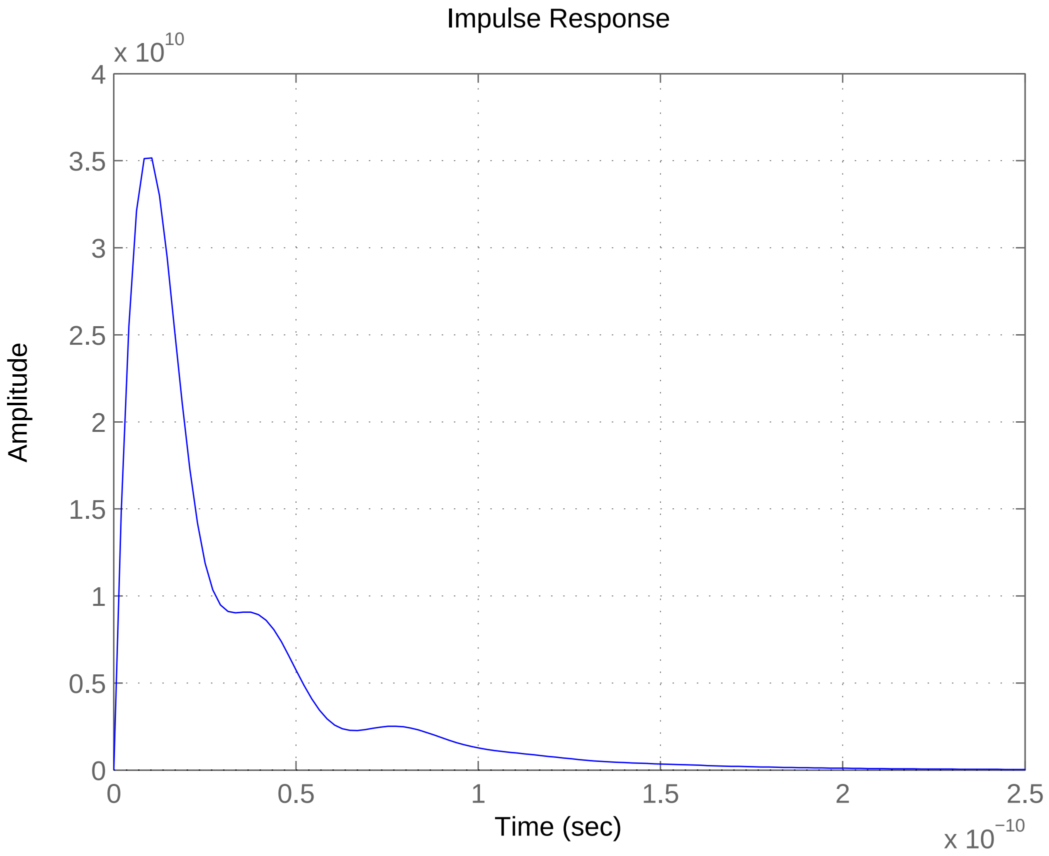

Fig. 2. The impulse response of a 1 mm long wire in a 8-bit wide bus.

the wire at least 18ps to crossover the half voltage point when transiting from one level to another, even without any disturbances from neighboring wires.

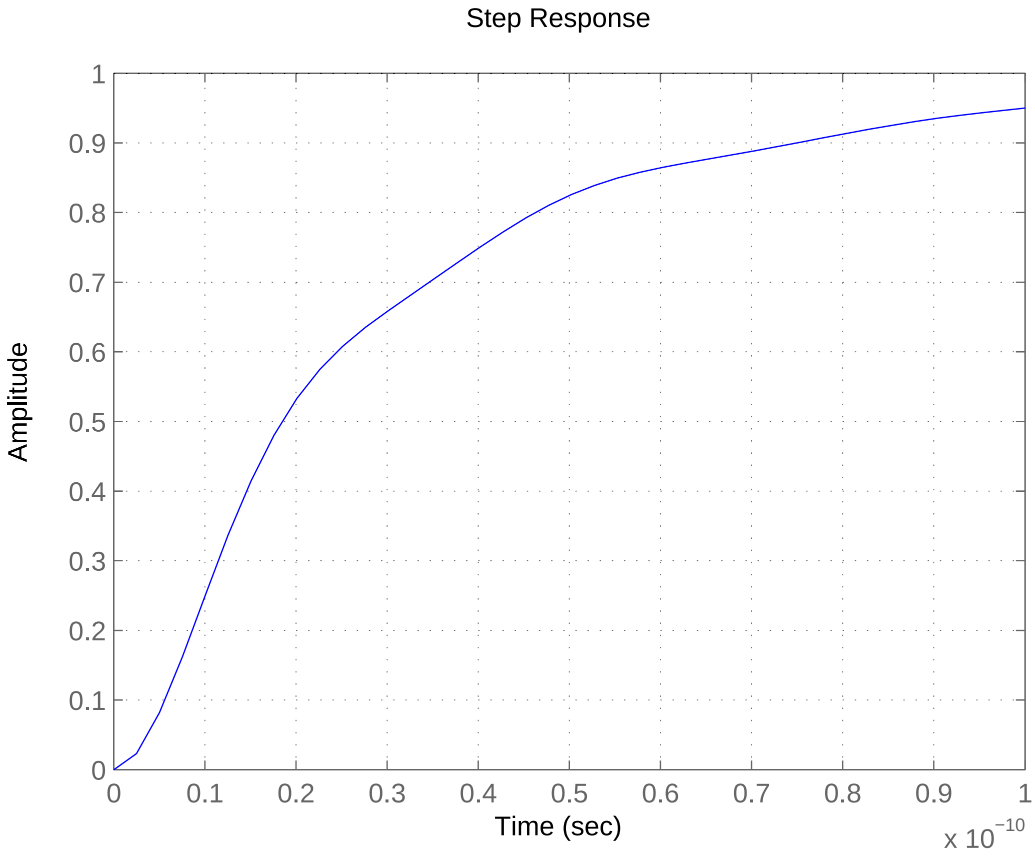

Fig. 3. The step response of a 1 mm long wire in a 8-bit wide bus.

- The *Frequency Response* in fig. 4 describes the voltage at the receiver of a single wire when the input of that wire is a sine wave of a variety of frequencies. The plot measures the ratio of the magnitude of the voltages, and the phase differences at the input to the output at different frequencies. The frequency response plot suggests that the wire acts as a Low Pass Filter (LPF) with less

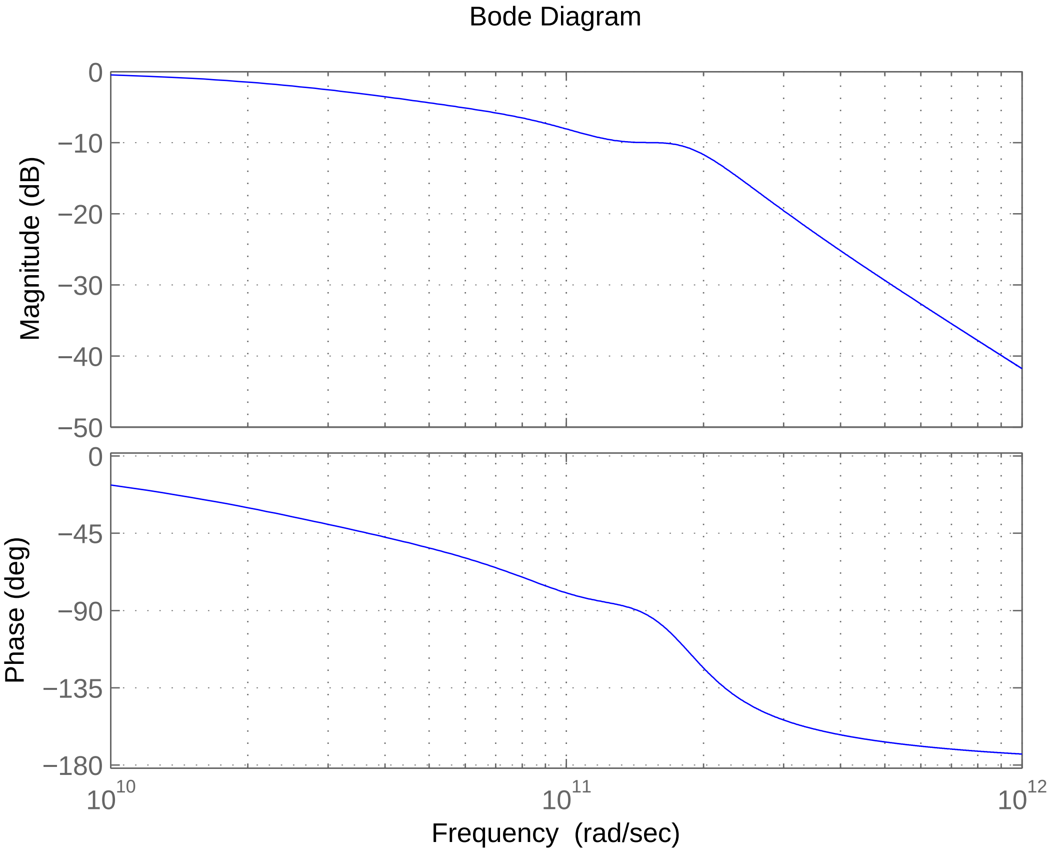

Fig. 4. The frequency response of a 1 mm long wire in a 8-bit wide bus.

than 10dB[1] attenuation at frequencies below 2×10^{11} radian per second, or ≈ 30 GHz.

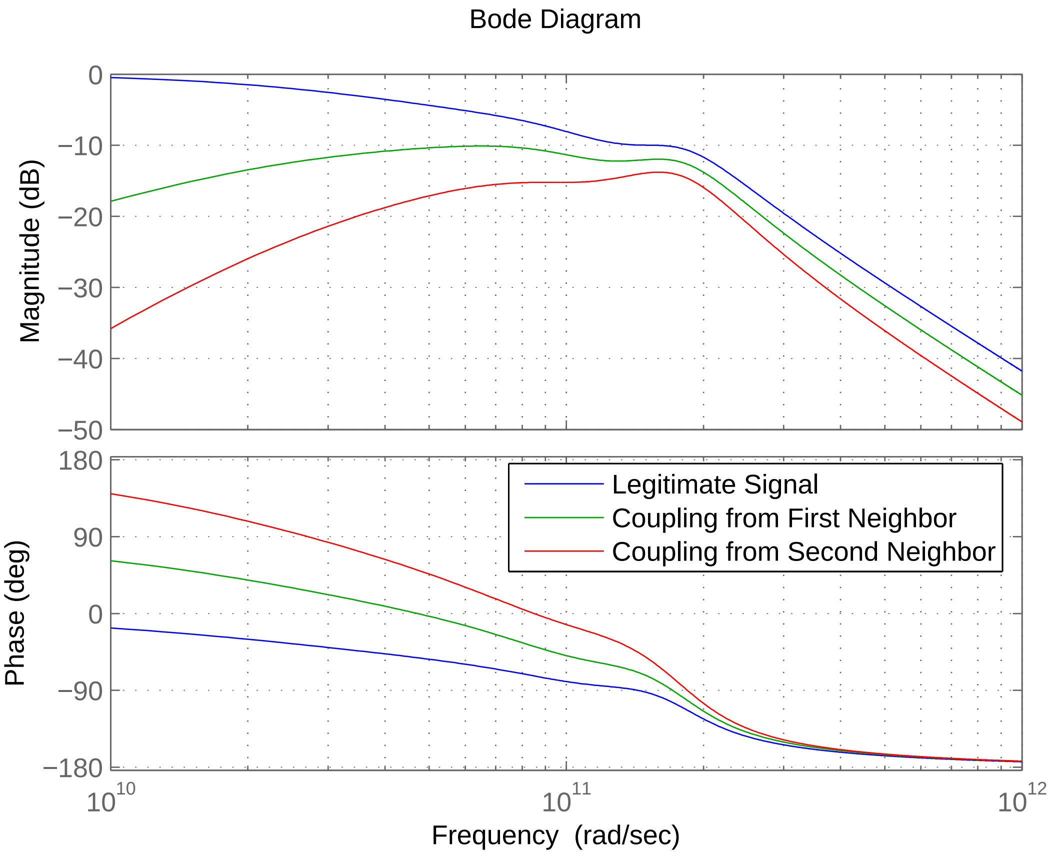

Fig. 5. The frequency response of a wire's input compared to the coupling response from the neighbors.

Fig. 5 shows the frequency dependence of the output of a particular wire to the input in the same wire as well as the first and second neighbors. A single wire acts as a low pass filter with a cut-off of about ≈ 30 GHz, as seen in both figs. 4 and 5. The coupling is negligible at low frequencies, and also at very high frequencies. However, the coupling interference is significant compared to the reference signal above 2×10^{11} radian per second. In this light, the ideal transmission frequency should be ≈ 30 GHz.

---

[1] The 10dB cut-off is a reasonable value, as opposed to the 3dB mark usually used in communication theory, because the threshold voltage V_{th} and the noise margins in modern technologies are both about 10dB lower than the V_{dd} values.

In addition to the input of the wire under study, when all other inputs are excited with a randomly generated data set, the output waveforms are illustrated in fig. 6. The input pulses are shaped as square waves as is customary in current technologies. It is seen that the output of a wire is not only affected by the input on that wire, but the signals on the neighboring wires as well. Some interferences are constructive while others are destructive. Also seen in fig. 6 are temporary glitches at the output, even when the input of the wire is not switching. This happens due to *coupling* with the adjacent wires. This phenomenon is better known as cross-talk.

For the same set-up, fig. 7 shows the receiver voltage corresponding to an input switching from a 0 to 1, when the neighboring signals switch arbitrarily. Note the large variations in this step response compared to fig. 3. These variations are due to cross-talk. Some interference speed-up the signal, while others slow it down. Certain bad patterns not only slow the signal, but also result in a high power consumption.

D. Delay Distribution

The variation in the step response of fig. 7 can be seen as a propagation delay variation as shown in fig. 8. Propagation delay is defined as the time taken for a signal to reach the half-way point of the desired value. In other words, it is the time required before the receiver can make a decision whether the transmitted signal is a 0 or a 1. The long tail of the distribution is clearly visible. This means that even without any data preprocessing, most signals arrive much faster than the worst case. Fig. 9 plots the worst case delays for different bus lengths. It is seen that the worst case delay increases as the square of length.

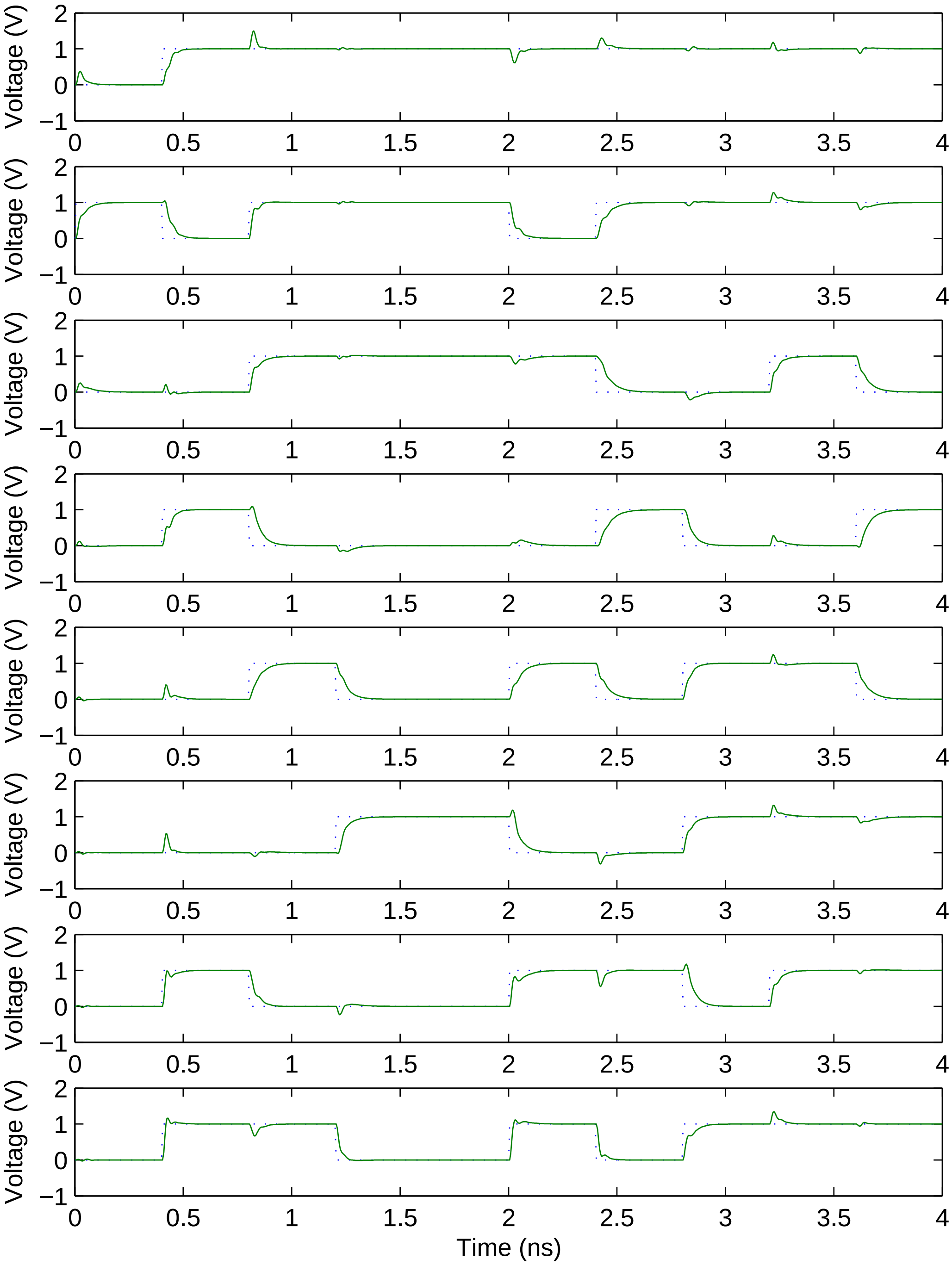

Fig. 6. The waveform for a 8 bit wide 1 mm long bus

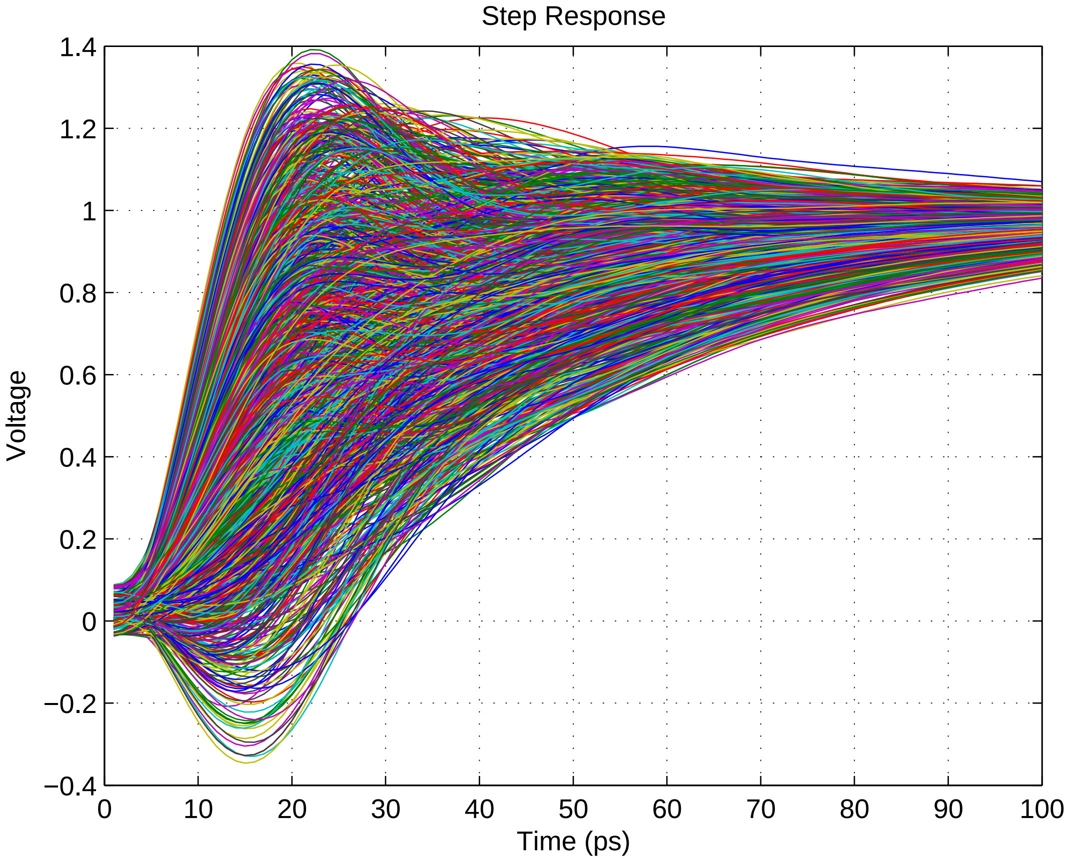

Fig. 7. The step response of one wire in a 8-bit wide bus when all inputs are active.

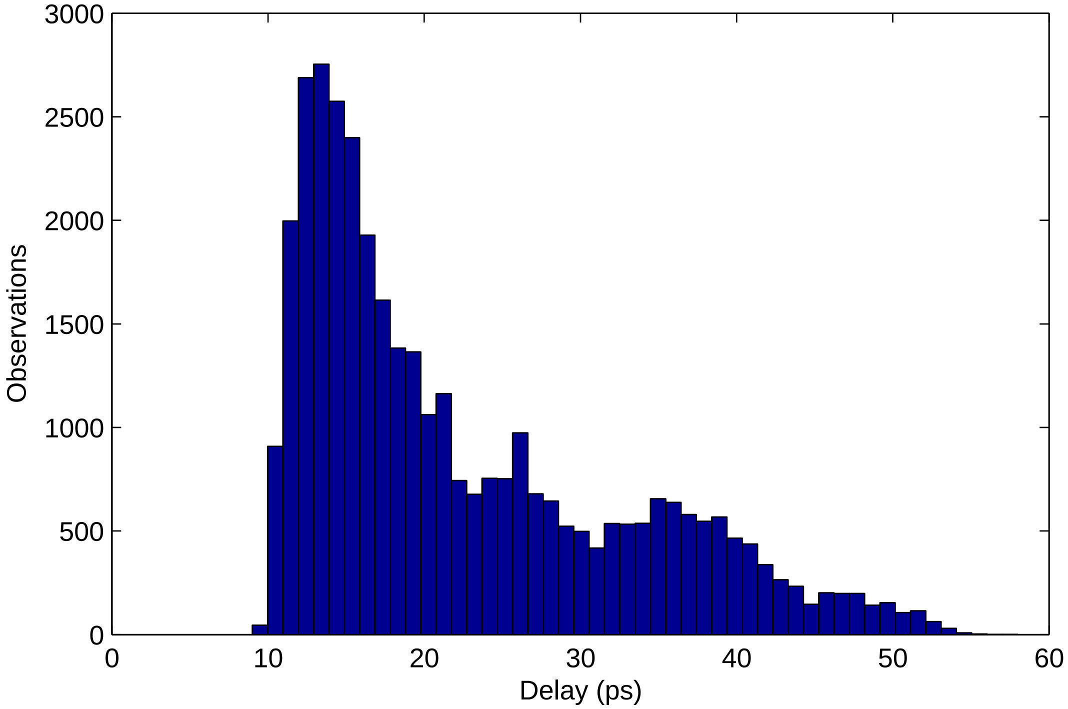

Fig. 8. The signal delay distribution of an 8-bit wide 1 mm long bus.

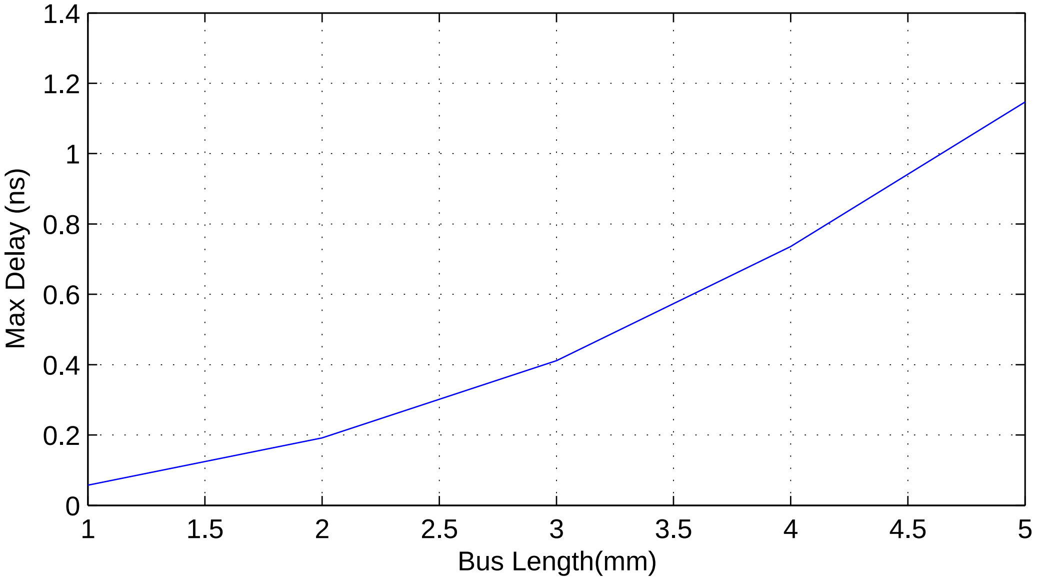

Fig. 9. The worst case signal delay of an 8-bit wide bus

CHAPTER III

INFORMATION THEORETIC CAPACITY OF ON-CHIP BUSES

A. Channel Capacity of Binary Symmetric Channels

A Binary Symmetric Channel (BSC) is a communication medium, the inputs to which are from the binary set $\{0, 1\}$ at the transmitter. The channel transmits this signal from the transmitter to a geograhically separated receiver. During this transmission of signals, errors happen resulting in flipping of some bits from 0 to 1 and 1 to 0. Let the probability with which this happens be p_e. So, if a 0 was transmitted, it will be received as a 0 or a 1 with probability $1 - p_e$ and p_e respectively.

Before proceeding to the definition of a BSC channel capacity, consider the following definitions.

1. Self-Information

For any random event E, which occurs with a probability $p(E)$, the self-information $I(E) = -\log_2 p(E)$ defines the information conveyed in bits by the occurrence of that event. For example, the information conveyed by the occurrence of heads in an unbiased coin is $\log_2 1/2 = 1$ bit. In the BSC context, where the bits 1 and 0 occur with probabilities p_1 and $p_0 = 1 - p_1$ respectively, the information conveyed by the occurence of 1 or 0 is defined as,

$$I(1) = -\log_2 p_1 \tag{3.1}$$

$$I(0) = -\log_2 p_0 = -\log_2(1 - p_1) \tag{3.2}$$

Now, if $p_1 = p_0 = 1/2$, the self-information $I(1) = I(0) = -\log_2(1/2) = 1$ bit. This means that whenever a 1 or a 0 occurs in a equiprobable distribution, 1 bit of information is conveyed by their occurrence.

2. Entropy

The entropy $H(s)$ of the system S of n random events $E_1, E_2, \ldots, E_n$, with probabilities of occurrence $p_1, p_2, \ldots, p_n$ is the measure of the system's uncertainty and is given as

$$H(S) = \sum_{k=1}^{n} p_k I(E_k). \tag{3.3}$$

The uncertainty in a system consisting of an unbiased coin with heads and tails as the events is 1 bit. The uncertainty in a system with a six faced dice is 2.58 bits. Again, in the BSC context, where the system S has only two possible outcomes 0 and 1,

$$\begin{aligned} H(S) &= p_0 I(0) + p_1 I(1), \\ H(S) &= -p_0 \log_2 p_0 - p_1 \log_2 p_1. \end{aligned} \tag{3.4}$$

Now, if $p_0 = p_1 = 1/2$, $H(S) = 1$ meaning that there is an uncertainty of 1 bit in the system. Also, if $p_0 = 0, p_1 = 1$, $H(S) = 0$ meaning that there is no uncertainty in the system, because obviously the outcome will always be 1.

3. Conditional Entropy

The conditional entropy of the system S_1 given the outcome of another random system S_2, is defined as

$$H(S_1|S_2) = -\sum_{j=1}^{n} \sum_{k=1}^{m} p_{jk} \log_2 p_{jk}/p_k \tag{3.5}$$

where $p_{jk} = p(E_j \cap F_k)$, and $p_k = p(F_k)$. n and m are the number of possible outcomes E and F of system S_1 and S_2 respectively. In simple words, conditional entropy is the uncertainty remaining in system S_1 given the outcome of S_2. If system S_1 is defined as a system with two coins, and system S_2 is defined as one of the coins, then before any coin toss, the uncertainty in the system S_1 is $H(S_1) = 2$ bits. Once the outcome of the system S_2 is known, the remaining uncertainty in the system S_1 is $H(S_1|S_2) = 1$ bit.

4. Capacity of Memory-less BSC Channels

For a communication channel, let the input and output be two different random systems S_1 and S_2 respectively. The capacity is defined as the *reduction in uncertainty* about the input given the channel output [36].

$$C = H(S_1) - H(S_1|S_2) \tag{3.6}$$

For a BSC $H(S_1) = 1$ when $p_0 = p_1 = 1/2$

$$\begin{aligned} C &= 1 - H(S_1|S_2) && (3.7) \\ C &= 1 + \sum_{j=1}^{2}\sum_{k=1}^{2} p_{jk} \log_2 p_{jk}/p_k && (3.8) \\ C &= 1 + p_e \log_2(p_e) + (1 - p_e)\log_2(1 - p_e) && (3.9) \end{aligned}$$

where, p_e is the bit-error-rate (BER) of the channel.

The units of capacity can be either bits per second or bits per transmitted symbol. The latter is a more popular notation and will be used in this work (like above), unless otherwise stated.

Another interpretation of capacity is the maximum rate of the code that is required to produce a *zero* probability of error. The rate of a code is the ratio of the total useful information bits to the total transmitted symbols. As an example, if a code takes 7 bits and converts them into a code of 10 bits, the rate is 0.7. Not all codes at this maximum code rate will exhibit a low probability of error, but theoretically, it is possible to construct at least one error correction code that can achieve this.

5. Capacity of BSC with Memory

The bit error rate in some channels is altered significantly, based on the past transmissions. Such channels, called the "Gilbert-Elliot Channels" [37] seem to have a multiple set of operating conditions or states. Consider a channel with n states $A_1, A_2, \ldots, A_n$, with each state having a bit-error-rate p_{ei} and a probability of the occurrence p_i. The capacity of each state $C_i = 1 + p_{ei}\log_2(p_{ei}) + (1 - p_{ei})\log_2(1 - p_{ei})$ and the overall [37] channel capacity is

$$C = \sum_{i=1}^{n} p_i C_i. \tag{3.10}$$

B. Bus Capacity in the Presence of Capacitive Coupling

In order to better understand the calculation of the capacity of a wide on-chip bus, lets take a step back and assume that there are no inductive efffects on the wires. The wires are modeled as low-pass RC circuits [1, 2, 16, 19, 22], with each wire having wire-to-substrate capacitance C_g and inter-wire capacitance C_c as shown in fig. 10.

Fig. 11 shows the signal waveforms in a 0.5 mm long, 4 bit wide bus in a 0.1 μm fabrication process. The effect of crosstalk is seen when the inputs are switching. For example, when the adjacent signals are switching in the same direction, the crosstalk results in a speeding up of the output signal. On the other hand, when the adjacent signals are switching in the opposite direction, crosstalk results in the undesirable slow-down of the output signal. This may lead to increased errors in detection. Fig. 12 shows the histogram of the output received on the same *crosstalk infested* bus. The nature of errors is symmetric on both electrical levels. Due to this symmetric

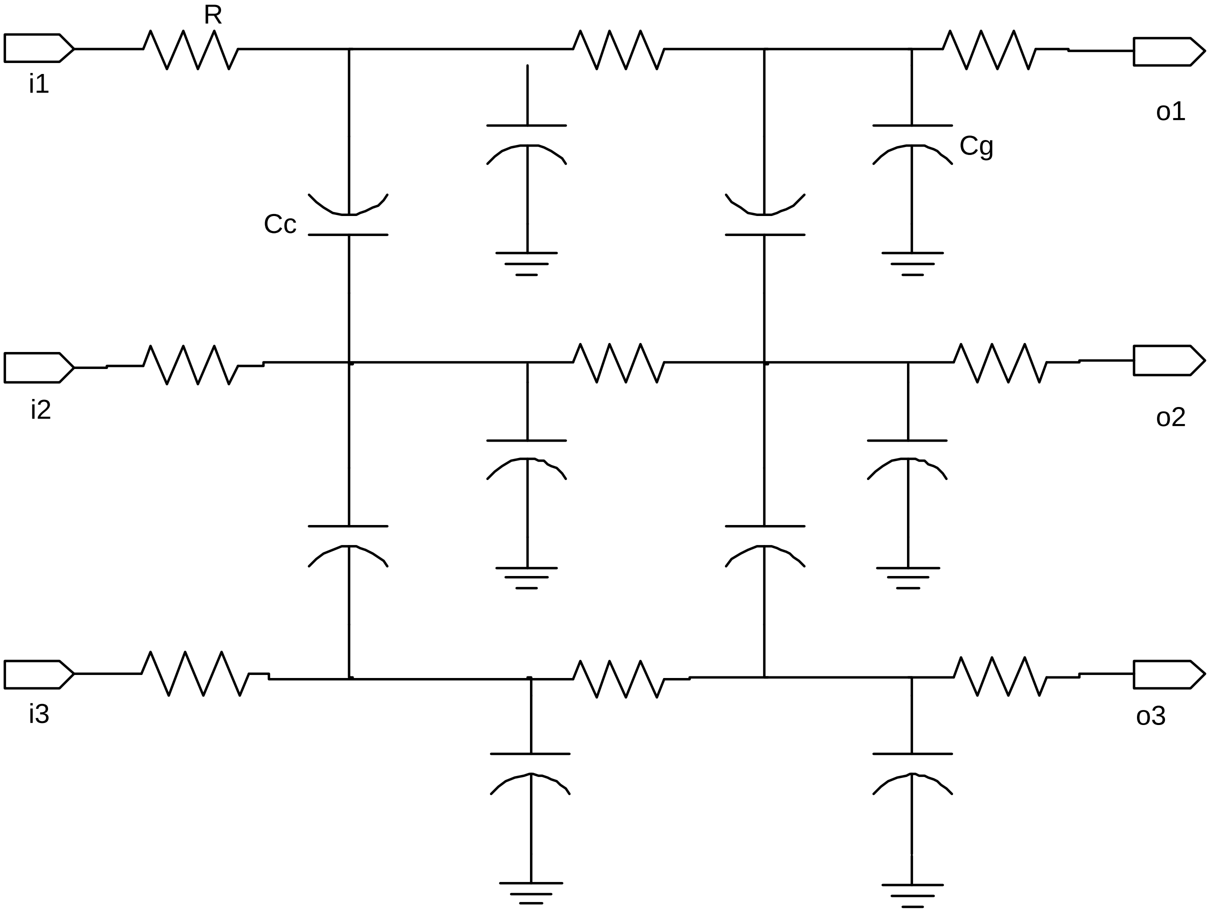
R
i1
o1
Cg
Cc
i2
o2
i3
o3

Fig. 10. The RC model for a bus

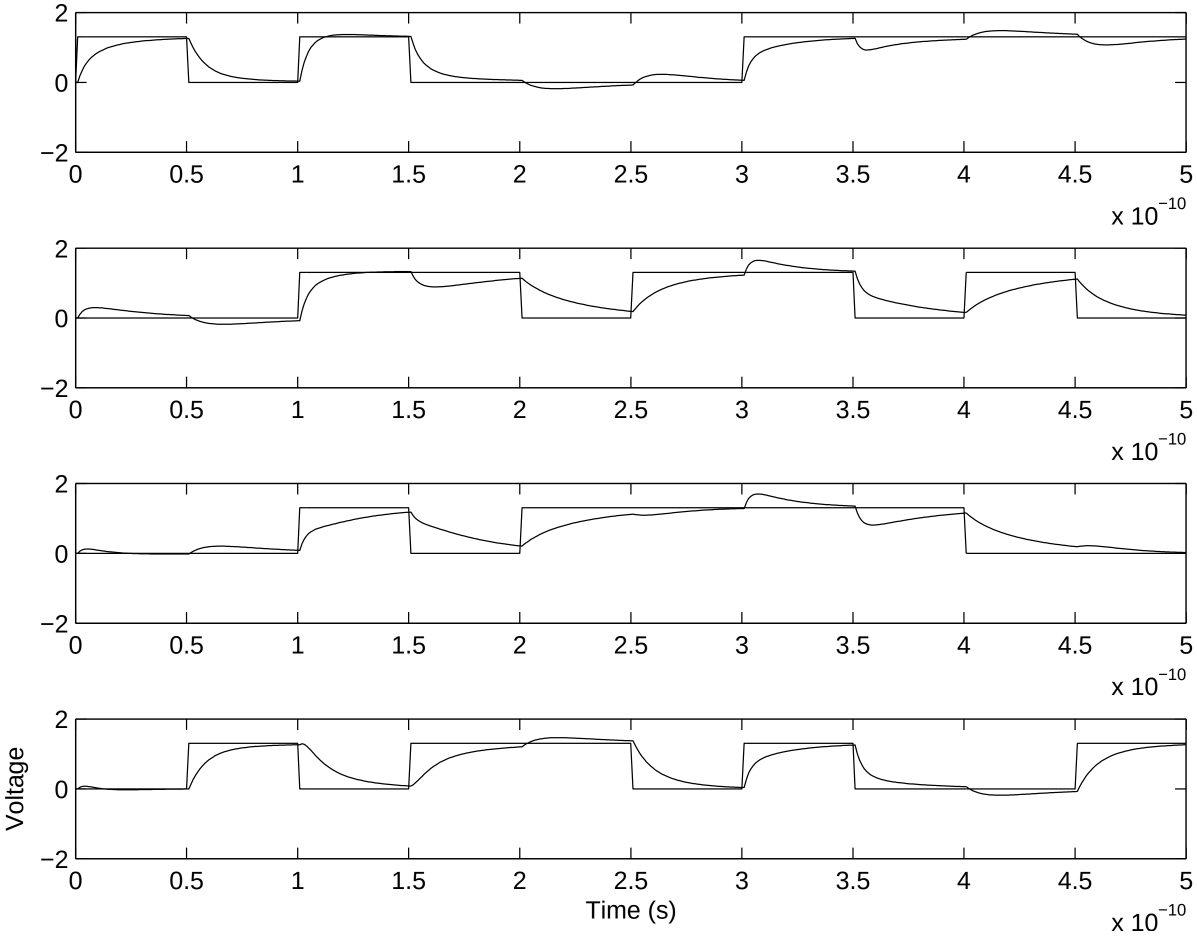

Fig. 11. The waveform for a 4 bit wide bus

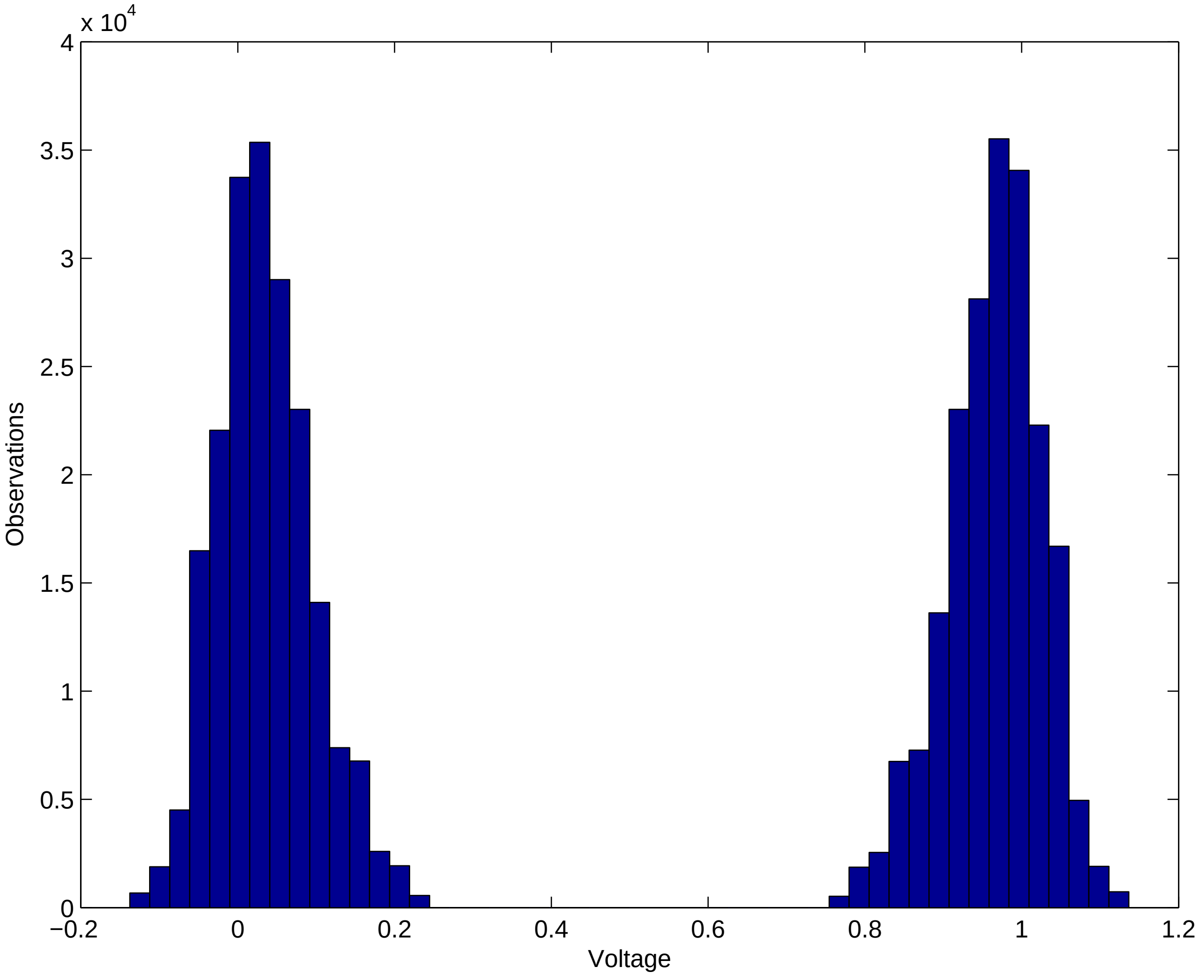

Fig. 12. Histogram of the output received from a 0.5mm long, 16-bit wide bus running at 10GHz.

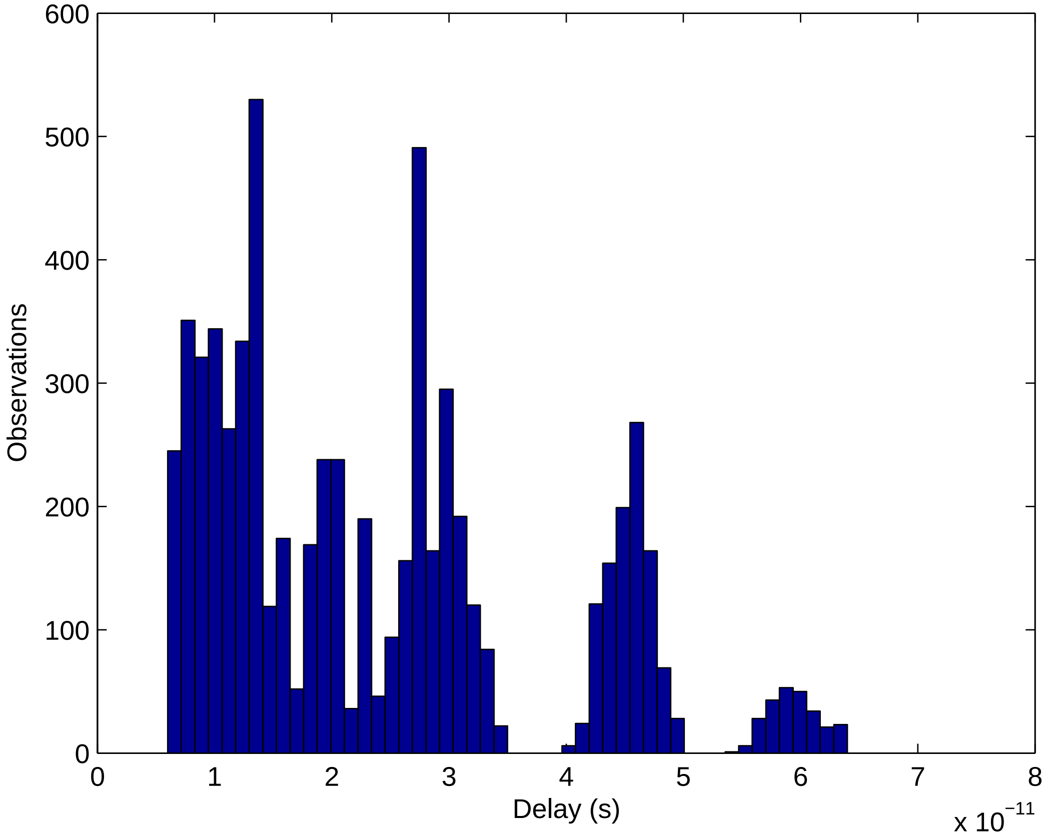

Fig. 13. Histogram of the signal propagation delay of a 1mm long, 16-bit wide bus.

nature of the output distribution, the crossover probabilities and hence the number of errors are minimum if $V_{swing} = 2 \times V_{trip}$, i.e., the swing voltage in the bus is twice of the receiver's decision making switching voltage. Fig. 13 shows the delay distribution of a 16-bit wide, 1 mm long bus. The signal propagation delay is defined as time required to charge the output voltage level to 50% of the steady-state value. Note the difference compared with the delay histogram in fig. 8 where inductive effects were also present.

1. Interconnect States and Capacity

Based on this understanding of the interconnects, the operating conditions can be broadly classified into the following states [19] that differ from each other primarily in terms of the capacitance that the input signal experiences.

1. State A_1 $(\mathbf{0C_c + C_g})$ occurs when the input and both neighbor signals are switching in the same direction. Probability of Occurrence is 1/32.
2. State A_2 $(\mathbf{1C_c + C_g})$ occurs when the input and one neighbor signal switches in the same direction. The other neighbor signal stays constant. Probability of Occurrence is 1/8.
3. State A_3 $(\mathbf{2C_c + C_g})$ occurs when the input is switching and the neighboring signals are either both constant or switching in mutually opposite directions. Probability of Occurrence is 3/16.
4. State A_4 $(\mathbf{3C_c + C_g})$ occurs when the input is switching and one neighboring signal switches in the opposite direction, while the other stays constant. Probability of Occurrence is 1/8.

5. State A_5 ($\mathbf{4C_c + C_g}$) occurs when the input is switching and the neighboring signals are switching in the opposite direction. Probability of Occurrence is 1/32.

6. State A_6 ($\mathbf{0C_c}$) occurs when the input is constant and the neighboring signals are either constant or switching in mutually opposite directions. Probability of Occurrence is 3/16.

7. State A_7 ($\mathbf{1C_c}$) occurs when the input is constant and one neighboring signal is switching. Probability of Occurrence is 1/4.

8. State A_8 ($\mathbf{2C_c}$) occurs when the input is constant and both neighboring signals are switching in the same direction. Probability of Occurrence is 1/16.

Apart from these broad states, there are other factors like process variations, interference from the non-adjacent wires, leakage, power noise, etc. These can be considered as noise and do not change the broad classifications of the interconnect states. Based on these states, the interconnect capacity can be calculated according to equation 3.10 as,

$$C = \sum_{i=1}^{8} p_i C_i \tag{3.11}$$

where, p_i is the probability of the occurrence of state A_i, and

$$C_i = 1 + p_{ei} \log_2(p_{ei}) + (1 - p_{ei}) \log_2(1 - p_{ei}), \tag{3.12}$$

is the capacity of information transfer, while p_{ei} is the crossover probability in each of these states.

2. The Mathematical Model and Calculations

In the absence of inductive effects, equation 2.6 changes to

$$F(s) = T^{-1}(s), \tag{3.13}$$

where

$$T(s) = \begin{bmatrix} Y & Z & 0 & \dots & 0 & 0 \\ Z & X & Z & \dots & 0 & 0 \\ 0 & Z & X & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & X & Z \\ 0 & 0 & 0 & \dots & Z & Y \end{bmatrix}, \tag{3.14}$$

and

$$X = 1 + sR(2C_c + C_g), \tag{3.15}$$

$$Y = 1 + sR(C_g + C_c), \tag{3.16}$$

$$Z = -sRC_c. \tag{3.17}$$

The bus parameters i.e. R, C_g and C_c are calculated as before.

To calculate the capacity, a large number of input signal patterns are passed through this transfer function, and errors are recorded at the receiver. The errors in this "monte-carlo" experiment are then classified into the 8 states mentioned above, depending on the input sequence. To attain a statistical closure on the results, a large number of frames having a random sequence of 16 bits were generated. These frames were passed through the transfer function $F(s)$ for various lengths L of the bus. Also, the input sequences should be switched at an arbitrary frequency f. Based on these errors and their probabilities, the interconnect capacity is calculated according to equation 3.11 and the results are plotted in figs. 14, 15, 16, 17, and 18.

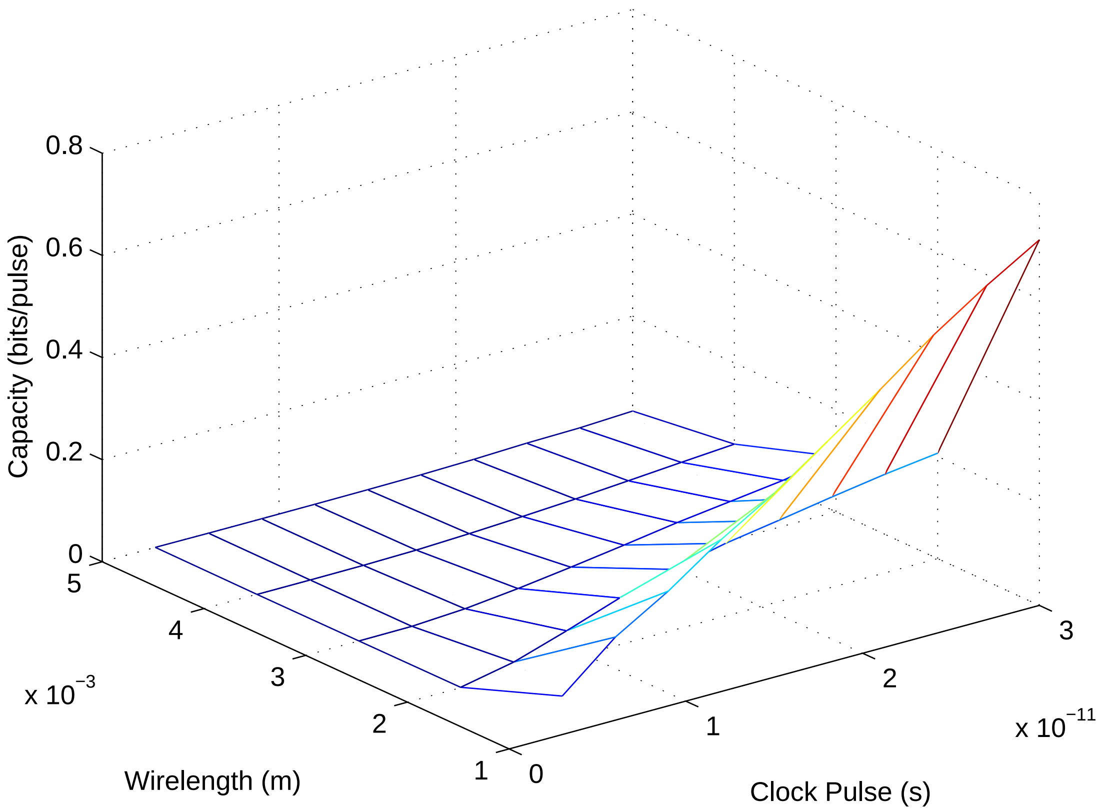

Fig. 14. The channel capacity as a function of wire length and clock speed.

Fig. 14 shows the channel capacity as a function of wire length and the clock speed. Figs. 15 and 16 show the channel capacity as a function of wire length in both bits per pulse and bits per second respectively. Figs. 17 and 18 show the channel capacity as a function of clock pulse time in both bits per clock pulse and bits per second respectively.

From figs. 14, 17, and 18, it is observed that the capacity of the channel in bits per second does not vary much as a function of clock speed. This means that the end result in terms of throughput will remain the same no matter how fast the clock is

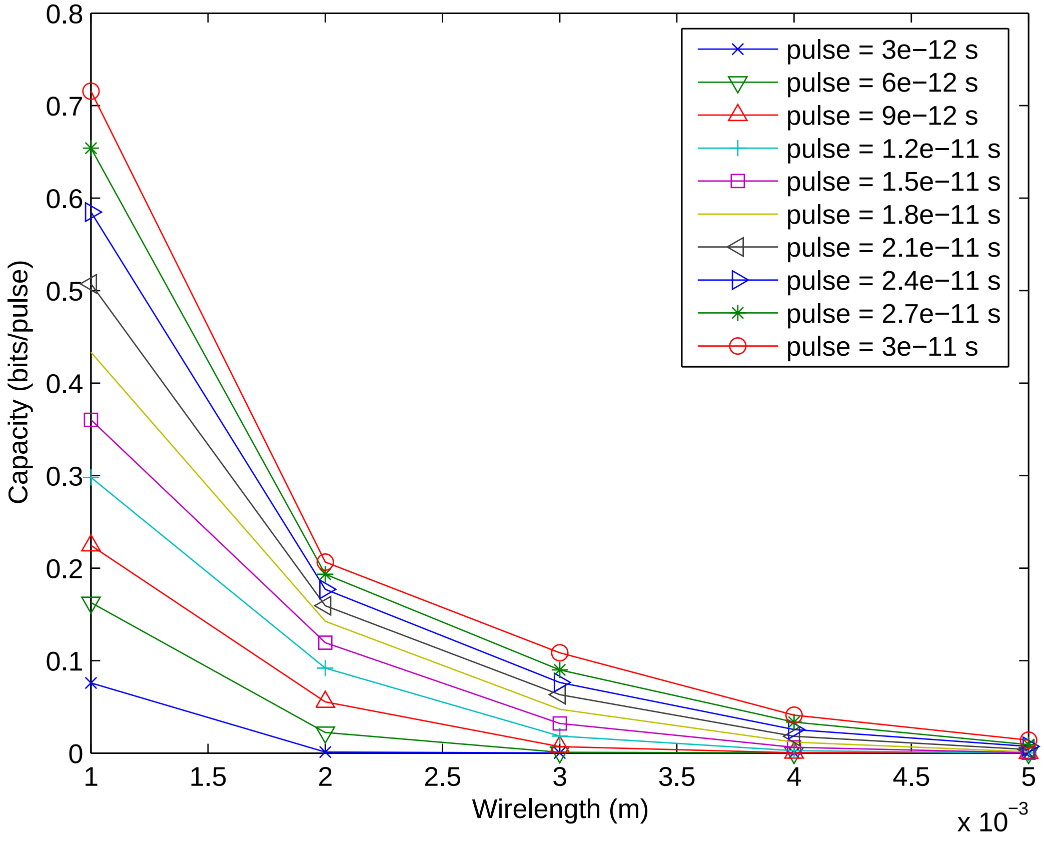

Fig. 15. The channel capacity in bits per period as a function of wire length.

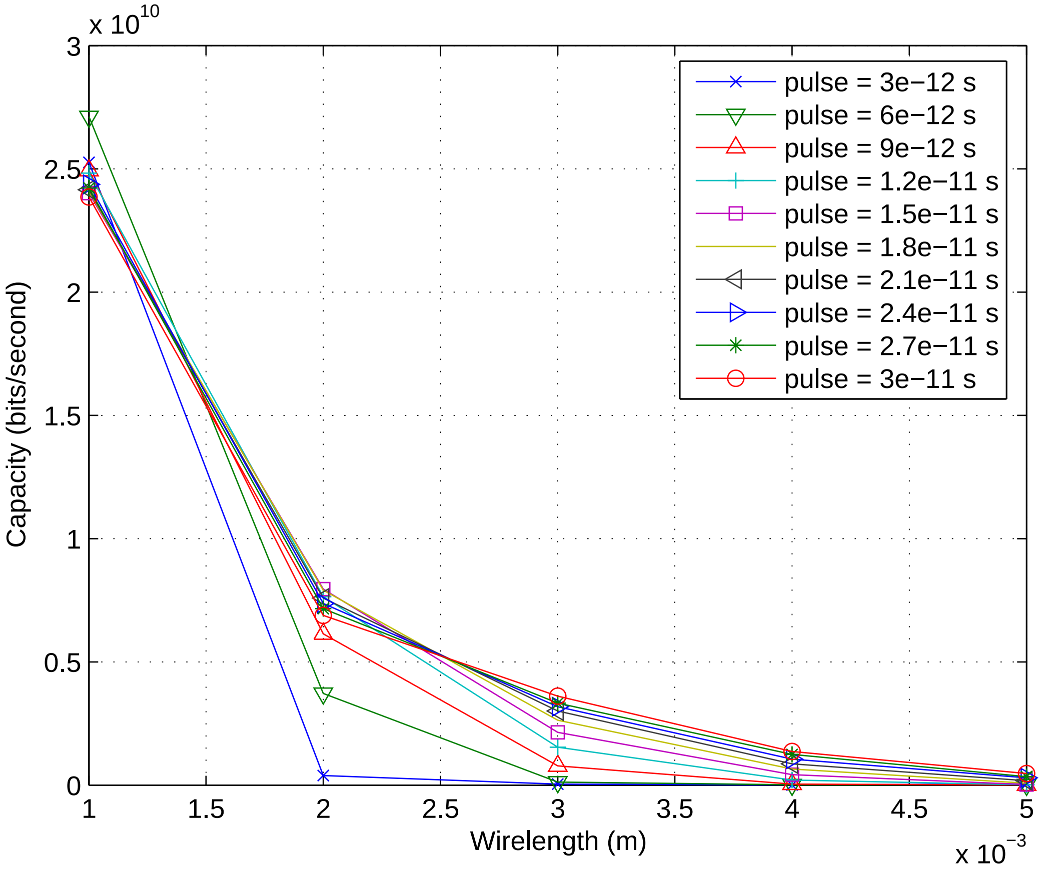

Fig. 16. The channel capacity in bits per second as a function of wire length.

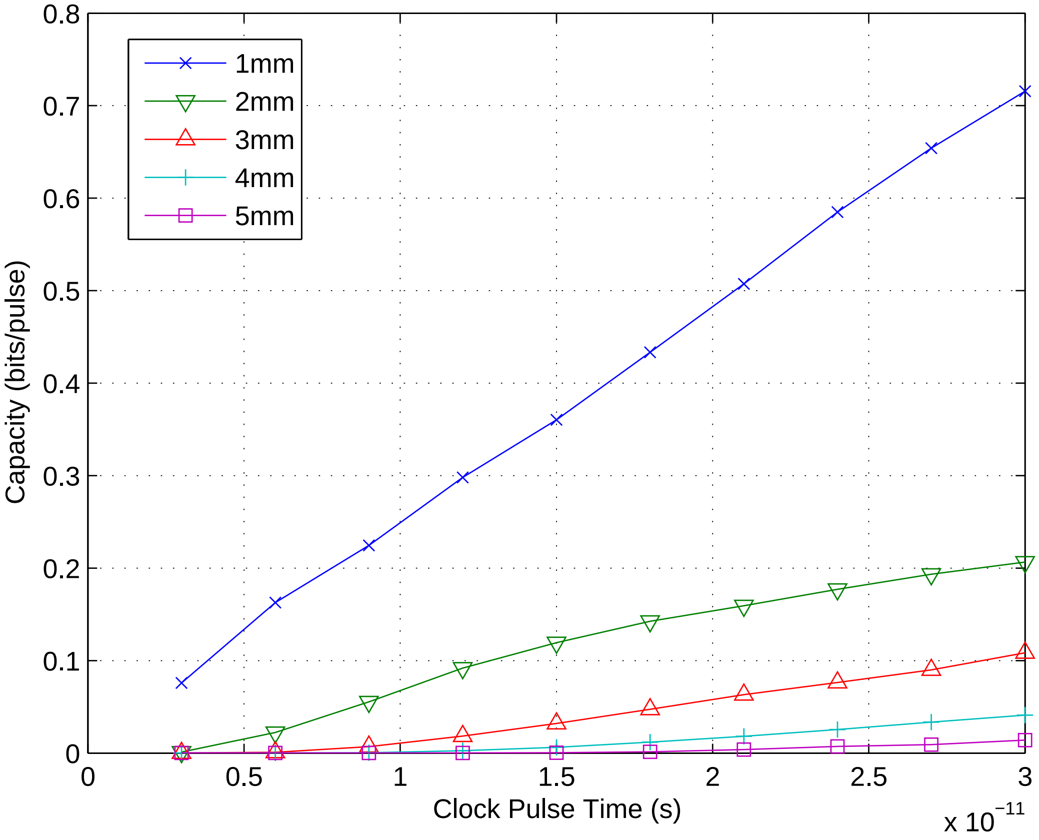

Fig. 17. The channel capacity in bits per period as a function of clock speed.

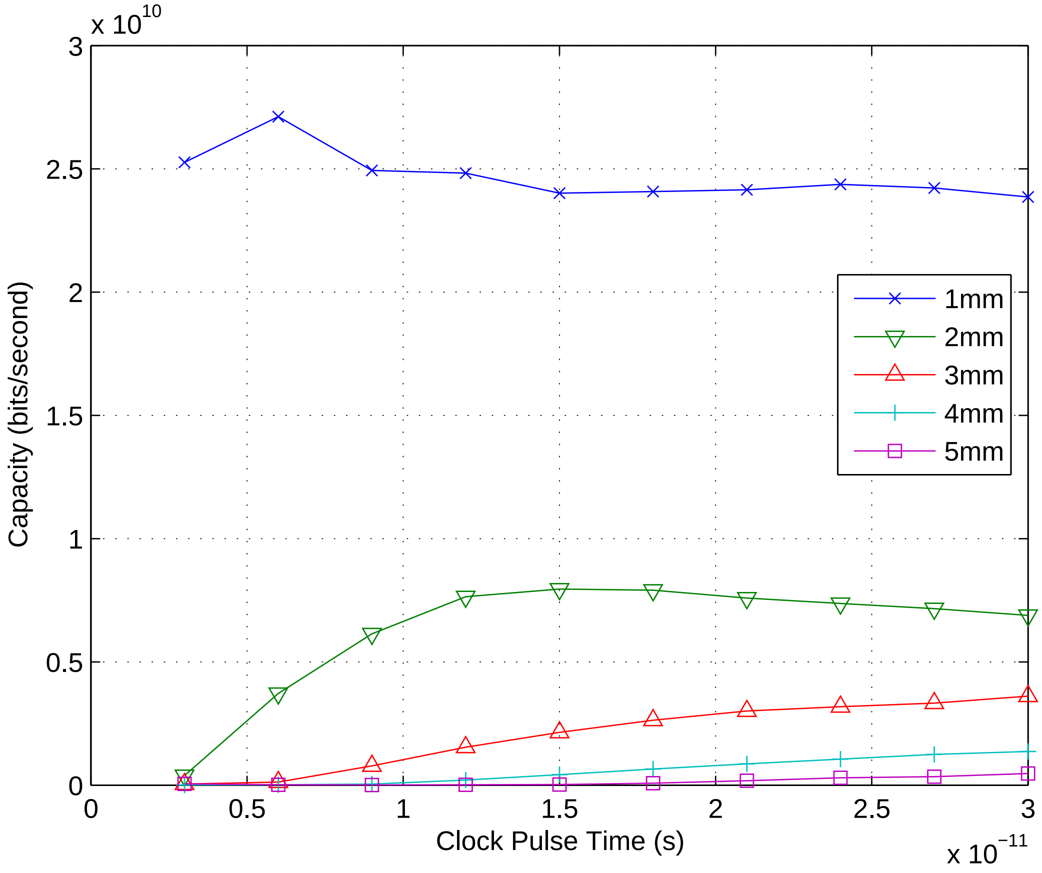

Fig. 18. The channel capacity in bits per second as a function of clock speed.

run. If the clock is very fast, there will be significant number of errors that have to be corrected with appropriate redundancy. While, on the other hand, a slower clock rate will require a lesser redundancy and will therefore achieve similar results.

It is also observed that the channel capacity decreases monotonically for all clock speeds with wire length, implying that the longer the wire, the lesser is its data handling capacity.

This capacity can be used to design better interconnects capable of handling "near capacity" data rates. For example, from fig. 18, for a wire length of 1mm, it is possible to transmit upto 25 Gbps on a single wire, using a faster clock and some error correction coding. On the other hand from fig. 13, for the same wire and same operating conditions, from the worst case timing analysis, a design engineer would design a clock rate of 14 GHz, resulting in a maximum data-rate of 14 Gbps. This design is too pessimistic and tries to accommodate very rare worst case delays. Error correction coding should be employed to compensate for any errors in the tail-end of the delay curve, and thereby significantly increasing the operation frequency and the data rates on interconnects.

It is also worth noting that the results for capacity do not make use of any buffers in the wire length. This makes the findings even more remarkable. Error correction coding can not only speed up the data-rates significantly but also save the power consumed in the buffers. This is an important find in light of the knowledge that the majority of the power in the communication systems is not consumed by the interconnects themselves but by supporting circuitry like buffers etc. [38].

C. Bus Capacity in the Presence of Capacitive and Inductive Coupling

It is apparent from the “multi-modal” distribution in fig. 8 that an interconnect operates in several different distinct conditions. These conditions can be broadly classified into the states [34, 35] listed in table I[1]. There are more states compared to [34] and section B, because unlike capacitive coupling, which is symmetric from the left and the right neighbor, inductive coupling is asymmetric. The probability of occurrence p_i of each state A_i is given in table I. Let C_i be the capacity of the channel when in state A_i, then

$$C = \sum_{i=1}^{13} p_i C_i. \tag{3.18}$$

Simulations are conducted to calculate the bit-error-rates observed in the bus in each of the operating states mentioned in table I. From figs. 4 and 5, it is seen that the output signal power is a function of the clock frequency and the length of the wire. Fig. 19 plots the capacity (bits per clock period per wire) as a function of bus length and the clock frequency. It is seen that the capacity falls down as the clock rate or the length increases. Note the similarity of this figure to figs. 4 and 5.

Fig. 20 plots the capacity as bits per second per wire as a function of the bus length and the clock frequency. It is observed that a given bus has an optimal operation frequency at which the data rate is maximum. This optimal frequency of operation is much higher than the worst case pessimistic design. As an example, the worst case clock design for a 1mm long bus would be 17.5 GHz resulting in 17.5 Gbps data through all the wires. It is possible to drive the bus at 40 GHz to achieve up to 28 Gbps with a rate 0.7 code. Again, a rate 0.7 code means that for every 7 bits of

---

[1] $\uparrow$ means a 0 to 1 transition, $\downarrow$ means a 1 to 0 transition, and $-$ means no transition

Table I. The Interconnect States

State	Left Wire	Relevant Wire	Right Wire	Prob
A_1	↑ ↓	↑ ↓	↑ ↓	1/32
A_2	↑ ↓	↑ ↓	- -	1/16
A_3	↑ ↓	↑ ↓	↓ ↑	1/32
A_4	- -	↑ ↓	↑ ↓	1/16
A_5	- -	↑ ↓	- -	1/8
A_6	- -	↑ ↓	↓ ↑	1/16
A_7	↓ ↑	↑ ↓	↑ ↓	1/32
A_8	↓ ↑	↑ ↓	- -	1/16
A_9	↓ ↑	↑ ↓	↓ ↑	1/32
A_{10}	↑ ↓	- -	↑ ↓	1/16
A_{11}	↑ ↓ - -	- - - -	- - ↑ ↓	1/4
A_{12}	↑ ↓	- -	↓ ↑	1/16
A_{13}	-	-	-	1/8

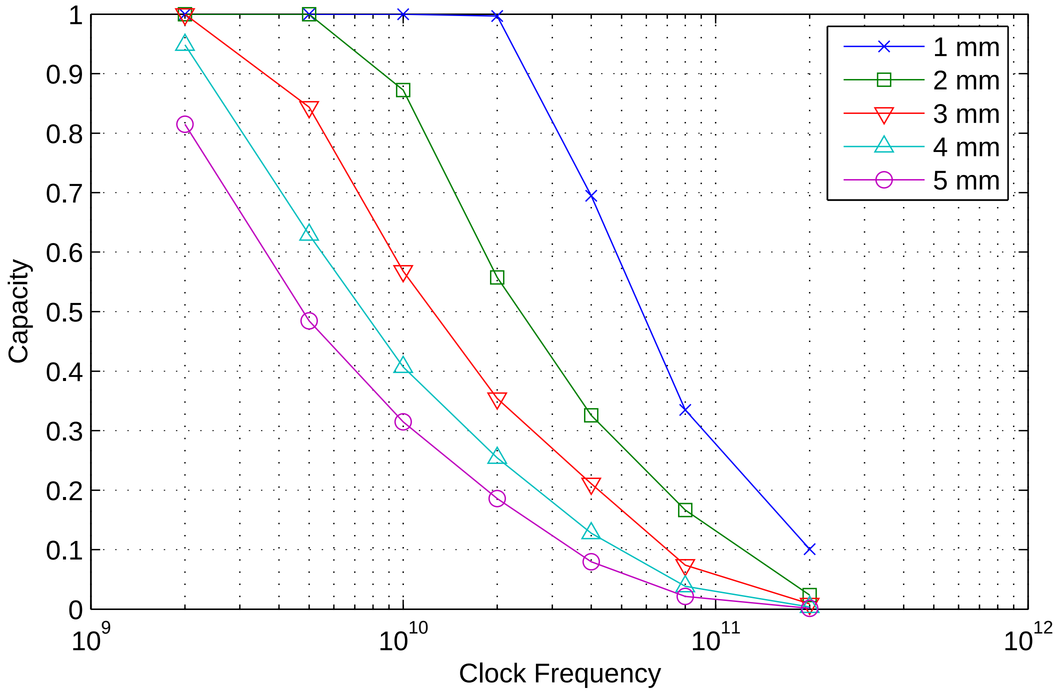

Fig. 19. The capacity of 8 bit wide bus as a function of clock frequency and length

information, there are 3 parity bits to recover errors, resulting in a rate $7/10 = 0.7$

Another important observation is that the optimal frequencies for different wire-lengths require different code rates. For example, running a 1mm long wire at the optimal frequency requires only a rate 0.7 Code while running a 5 mm wire at its own optimal frequency requires a rate 0.18 code. While the coding scheme for the 1 mm wire results in only a 50% overhead, the coding scheme for the 5 mm wires results in more than 500% overhead. Though constructing rate 0.7 codes is fairly simple, constructing rate 0.2 codes to provide error correction in a 5 mm wire may prove tricky if not impossible. This may be a trade-off requiring operation at non-optimal frequencies at a higher rate code. Also, at a very high wire-length buffer insertion may be coupled with coding schemes to become an ideal option.

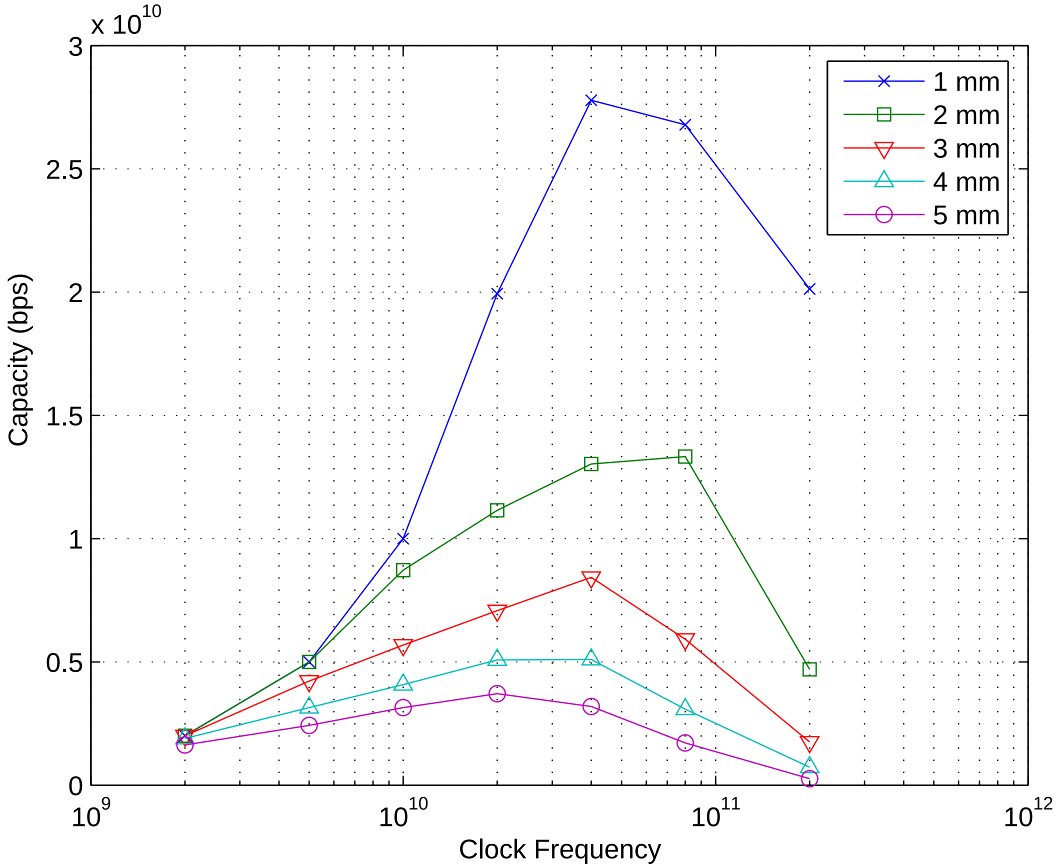

Fig. 20. The data rate limits of 8-bit wide bus as a function of clock frequency and length

CHAPTER IV

CODES FOR ON-CHIP BUSES

A. Previous Work

The problem associated with crosstalk and other error inducing factors is 3 fold.

- High bit error rates
- Large delays and delay variations.
- High energy consumption

While buffers reduce the delays and improve the reliability, they lead to high energy consumption. The coding techniques discussed in [16, 17, 20, 22–24] improve just one of the aspects mentioned above. The codes that aim to aggressively achieve speedup [16,17,20], ignore energy and reliability considerations. Some others improve the energy consumption [22–24] while leaving out the other two factors.

The problem that remains is to design a coding scheme that accomplishes all three tasks of low energy, low delay and low error rates. In [15] an attempt has been made to achieve all of the three improvements by employing a series of coders at the driver side. There is a "low delay" code, followed by a "low power" code(LPC), followed by an "error correcting code" (ECC). While the objective of improving delay, energy and reliability is achieved, this happens at a huge overhead. Each code adds its own redundancy to achieve its goal. The total redundancy overhead in the end is prohibitive for any real implementation. These codes are listed in table II.

All codes are consructed using a cascade of crosstalk avoidance codes (CAC), and LPC followed by ECC. The various techniques used in the construction of these codes are BI [1], Shielding [2], and Duplication [3]. Speedups are only offered where CAC is present. The CAC reduces the worst case capacitive coupling from $4C_c$ to $2C_c$ resulting in a speedup of $\frac{C_g+4C_c}{C_g+2C_c}$. Table III list the properties of these codes. In the table, the data rates per wire is calculated by multiplying the reference frequency (of the no code system) by the speed-up and the code rate. To calculate the speed-up the assumption is $C_c = 4C_g$. Fig. 21 shows a comparison of all codes against the information capacity. It is seen that BSC and DAP are the best codes in terms of data rate v/s overhead comparison.

Table II. The Code Construction [15] for a 4-Bit Wide Bus

Code	CAC	LPC	ECC	Total Extra Wires
Hamming	-	-	Hamming	3
HammingX	-	-	Hamming & parity	4
BIH	-	BI	Hamming	5
FTC+HC	CAC [20]		Hamming	10
BSC	Shielding	-	-	5
DAP	Duplication	-	Parity	5
DAPX	Duplication	-	Parity with shielding	6
DAPBI	Duplication	BI	Parity	7

---

[1] Bus Invert Codes [24] as LPC

[2] Insert a Vdd or Gnd Wire between data lines as a CAC

[3] Duplication as a CAC

Table III. The Data Per Wire Rates for Different Codes

Code	Rate [15]	Speed Up	1mm d/w Gbps	2mm d/w Gbps	3mm d/w Gbps	4mm d/w Gbps
No Code	1	1	17.5	5.21	2.43	1.36
Hamming	0.57	1	10.0	2.98	1.39	0.77
HammX	0.5	1	8.77	2.60	1.22	0.68
BIH	0.44	1	7.80	2.31	1.08	0.60
FTC+HC	0.28	1.88	9.47	2.81	1.31	0.73
BSC	0.44	1.88	14.7	4.37	2.04	1.14
DAP	0.44	1.88	14.7	4.37	2.04	1.14
DAPX	0.4	1.88	13.3	3.94	1.84	1.03
DAPBI	0.36	1.88	12.1	3.58	1.67	0.93
Capacity			28	14	8	5

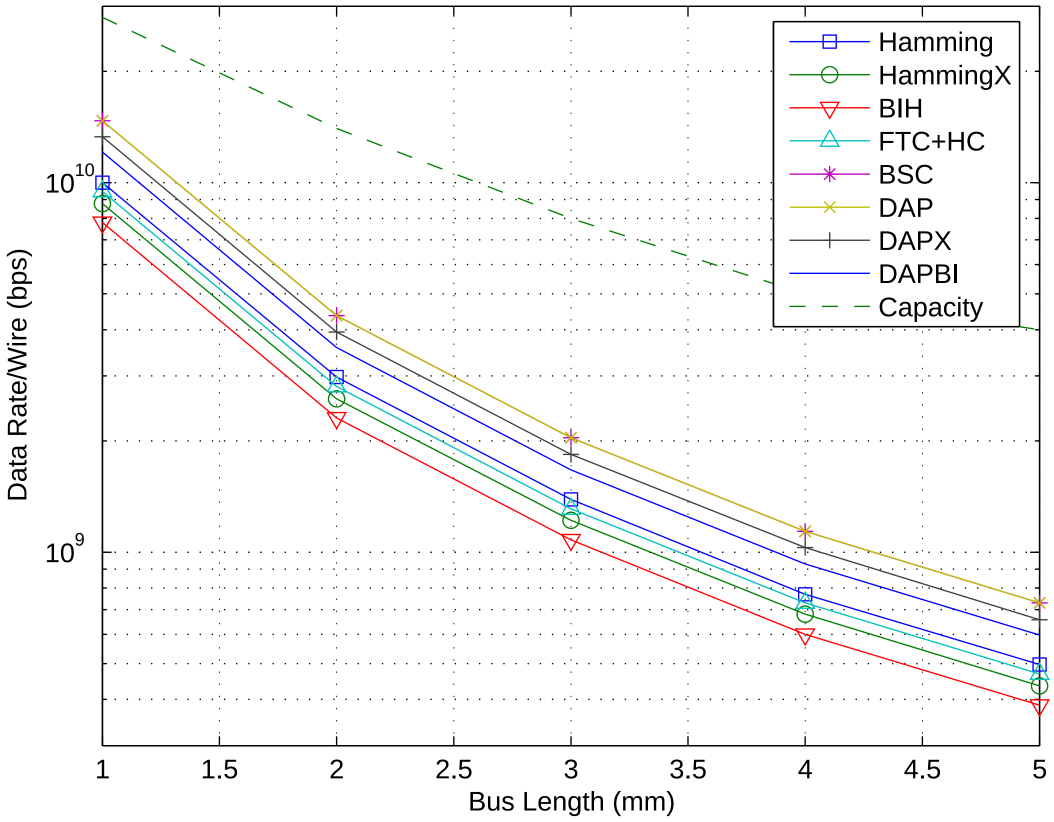

Fig. 21. The comparison of various codes against capacity

B. Bus Properties

As in Ch. II, the small section of an n-bit wide bus can be described as a multi-input-multi-output (MIMO) system, the transfer function of which can be described in Laplace domain as an $n \times n$ matrix $F(s) = (I + L(s)C(s))^{-1}$, where, the size of I, $L(s)$ and $C(s)$ is $n \times n$. $L(s)$ defines the inductance and resistance while $C(s)$ defines the capacitance of the wires.

1. Delay

The signal delay of the lth wire, $(1 < l < n)$ in an n-bit wide bus is given in [17] as

$$T_d \propto C_g \left\{(1 + 2\lambda)|\Delta_l| - \lambda\Delta_l(\Delta_{l-1} + \Delta_{l+1})\right\} \tag{4.1}$$

where, Δ_l is the transition in the lth wire of a bus and $\lambda = C_c/C_g$. This delay equation is not accurate in the presence of inductive effects. However, $R >> \omega L_s > \omega L_m$ for $\omega < 2\pi$ 100 GHz, especially for small wire lengths. This equation is therefore a good approximation for low frequencies. The various various delay patterns [20] are summarized below. In all the patterns $\Delta_l = \pm 1$.

$$\begin{bmatrix} \Delta_{l-1} \\ \Delta_{l+1} \end{bmatrix} = \left\{ \begin{array}{ll} \begin{bmatrix} -\Delta_l \\ -\Delta_l \end{bmatrix} & , 4C \\ \begin{bmatrix} -\Delta_l \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ -\Delta_l \end{bmatrix} & , 3C \\ \begin{bmatrix} -\Delta_l \\ \Delta_l \end{bmatrix}, \begin{bmatrix} \Delta_l \\ -\Delta_l \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \end{bmatrix} & , 2C \\ \begin{bmatrix} \Delta_l \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ \Delta_l \end{bmatrix} & , 1C \\ \begin{bmatrix} \Delta_l \\ \Delta_l \end{bmatrix} & , 0C \end{array} \right\} \tag{4.2}$$

- 4C Patterns - $T_d \propto C_g(1 + 4\lambda)$.

- 3C Patterns - $T_d \propto C_g(1+3\lambda)$.
- 2C Patterns - $T_d \propto C_g(1+2\lambda)$.
- 1C Patterns - $T_d \propto C_g(1+\lambda)$.
- 0C Patterns - $T_d \propto C_g$.

As seen in the patterns above, neighboring signals switching in opposite directions lead to slower signals while neighboring signals switching in the same direction have a "speed-up" effect on each other. The delay of a bus is the worst case coupling pattern. There may be a large number of 1C patterns, but if there is a single 4C pattern, the delay of the bus is 4C. Any coding scheme that aims to speed-up the signal propagation should therefore avoid the worse patterns.

2. Energy

The total energy consumed by the drivers of a bus [18] when transmitting the kth frame is

$$E_k = \frac{V_{dd}^2}{2} S_k^T \frac{C(s)}{s} \left[S_k - S_{k-1} \right]. \tag{4.3}$$

Here s is the laplace variable, V_{dd} is the supply voltage, S_k is a column vector of n binary values in the kth frame. S_k^T is the transpose. The energy consumed by the driver of a single wire l for transmitting the kth bit is

$$E_{l,k} = \frac{C_g V_{dd}^2}{2} s_{l,k} \left[\begin{array}{ccc} -\lambda & 1+2\lambda & -\lambda \end{array} \right] \left[\begin{array}{c} \Delta_{l-1} \\ \Delta_l \\ \Delta_{l+1} \end{array} \right], \tag{4.4}$$

where, $s_{l,k} \in \{0,1\}$ is the final binary value of the lth wire. The energy consumed for the various patterns discussed above are as follows.

- 4C Patterns - $E_{l,k} \propto C_g(1 + 4\lambda)$.
- 3C Patterns - $E_{l,k} \propto C_g(1 + 3\lambda)$.
- 2C Patterns - $E_{l,k} \propto C_g(1 + 2\lambda)$.
- 1C Patterns - $E_{l,k} \propto C_g(1 + \lambda)$.
- 0C Patterns - $E_{l,k} \propto C_g$.

Based on the equations above, it might seem that any coding scheme that aims to eliminate the bad delay patterns will have a similar effect on energy consumed and that the energy problem need not be independently considered while designing codes. This is however not true. While the delay of a bus was the delay of the *worst case* pattern, the energy required for transmission is the *average* of all patterns.

As an example, consider bus shielding versus a simple *illustrative* code called the "AND code" described below. In bus shielding, wires carrying the V_{dd} or ground are inserted between data wires. Since there is no signal transition in the shield wires, there are no 4C, 3C, 1C and 0C patterns. The only pattern available is the 2C pattern. The AND code is implemented by replacing the V_{dd} or ground signal in the shield wires by a logical AND of the two surrounding data wires. This scheme eliminates the 4C and 3C sequences, while keeping 2C, 1C and 0C sequences.

The schemes are implemented on 1 mm long bus in 0.1 μm technology, where $\lambda = 4$ based on a parallel plate model. The delay distributions of both schemes are given in figures 22 and 23 respectively. It is seen that the worst case delay in both cases is 31 ps corresponding to a 2C pattern. The speedup compared to fig 8 is roughly $\frac{1+4\lambda}{1+2\lambda} = \frac{17}{9} = 1.88$ or $\frac{57ps}{31ps} = 1.83$. The small discrepancy is attributed to the inductive effects.

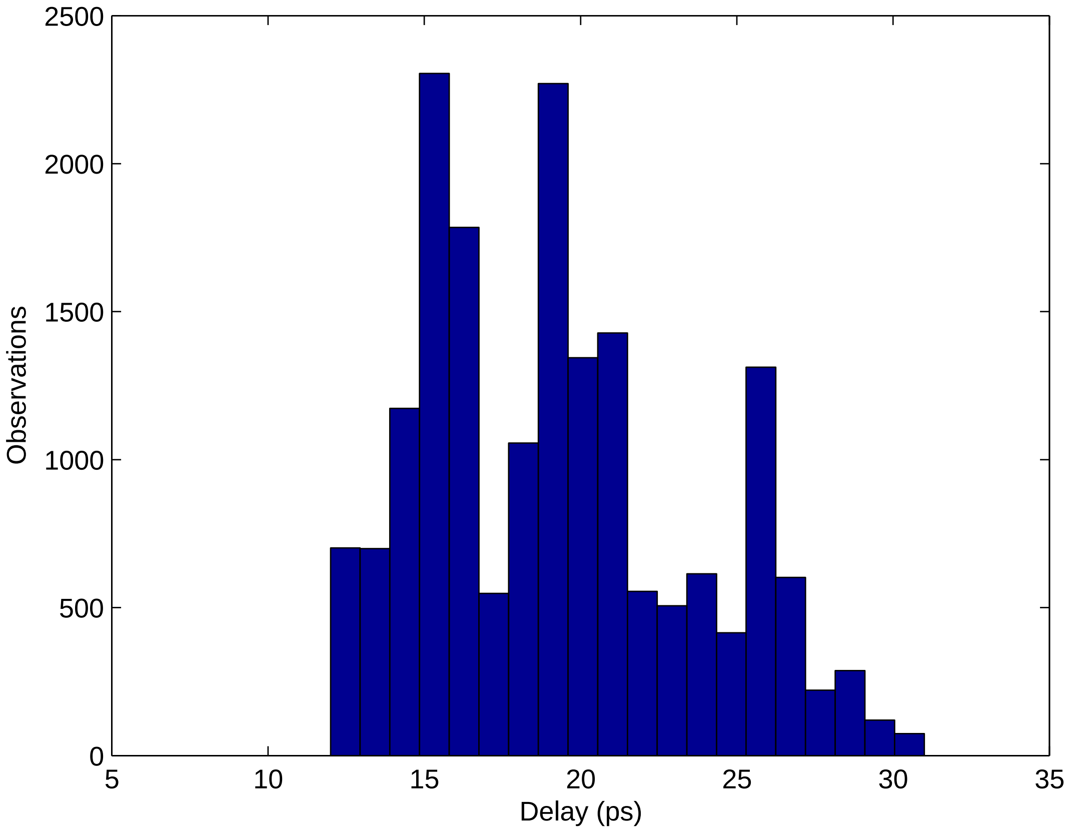

Fig. 22. The signal delay distribution of a 1 mm long “Shielded” bus.

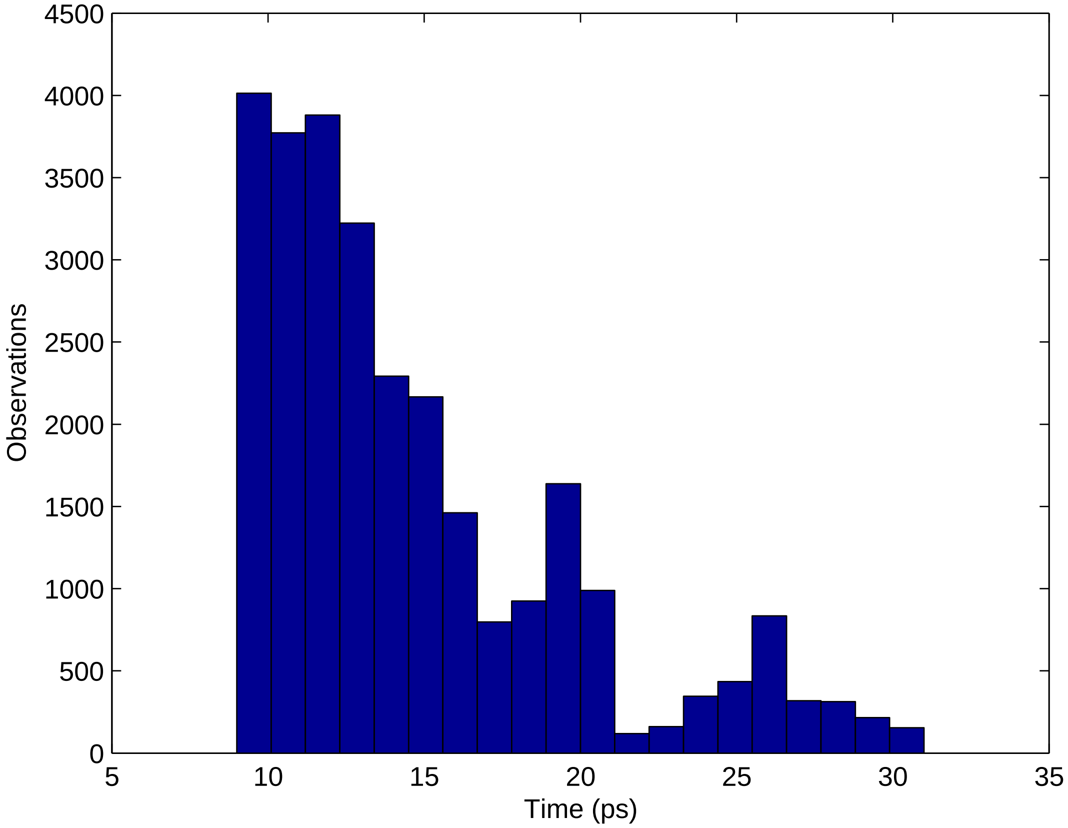

Fig. 23. The signal delay distribution of a 1 mm long bus with “AND” code.

Even though the speedup is equal in both cases, the energy gains are quite different. For 4 bit data, the average energy consumed in an un-coded bus is $\propto 1 + 1.5\lambda$, the average energy in the bus with shielding is also $\propto 1 + 1.5\lambda$, while the average energy in the AND coded bus is $\propto 1.5625 + 1.125\lambda$, meaning a total saving of $1 - \frac{1.5625+1.125\lambda}{1+1.5\lambda} = 1 - \frac{6.0625}{7} = 13.4\%$ over both the un-coded bus and the bus with shielding. Note that this saving is despite the fact that a AND coded bus uses 7 wires versus the 4 wires used by an un-coded bus.

3. Reliability

The reliability of signals in long buses is affected by the DSM noise, which is also known as the power noise. Power noise is a result of a rapid draining of the power capacitors on a chip when a large number of devices switch simultaneously. As a consequence, the supply voltages deviate from their "rail" values. At times, this deviation results in detection errors at the receiver.

Another factor that leads to errors in detection is the clock speed (delay). Fig. 24 shows the dependence of the received output signal on the clock frequency. It is seen that at faster clocks, there is more chance of an error, while at slower clocks, the signals are more "settled-down". Typically the clock speeds of a system are designed such that signals settle down completely, and the only errors are due to the DSM noise. An ECC can not only correct the errors due to DSM noise, but also the errors due to delay, therefore a two-fold gain in speedup can be had with a "low-delay" ECC.

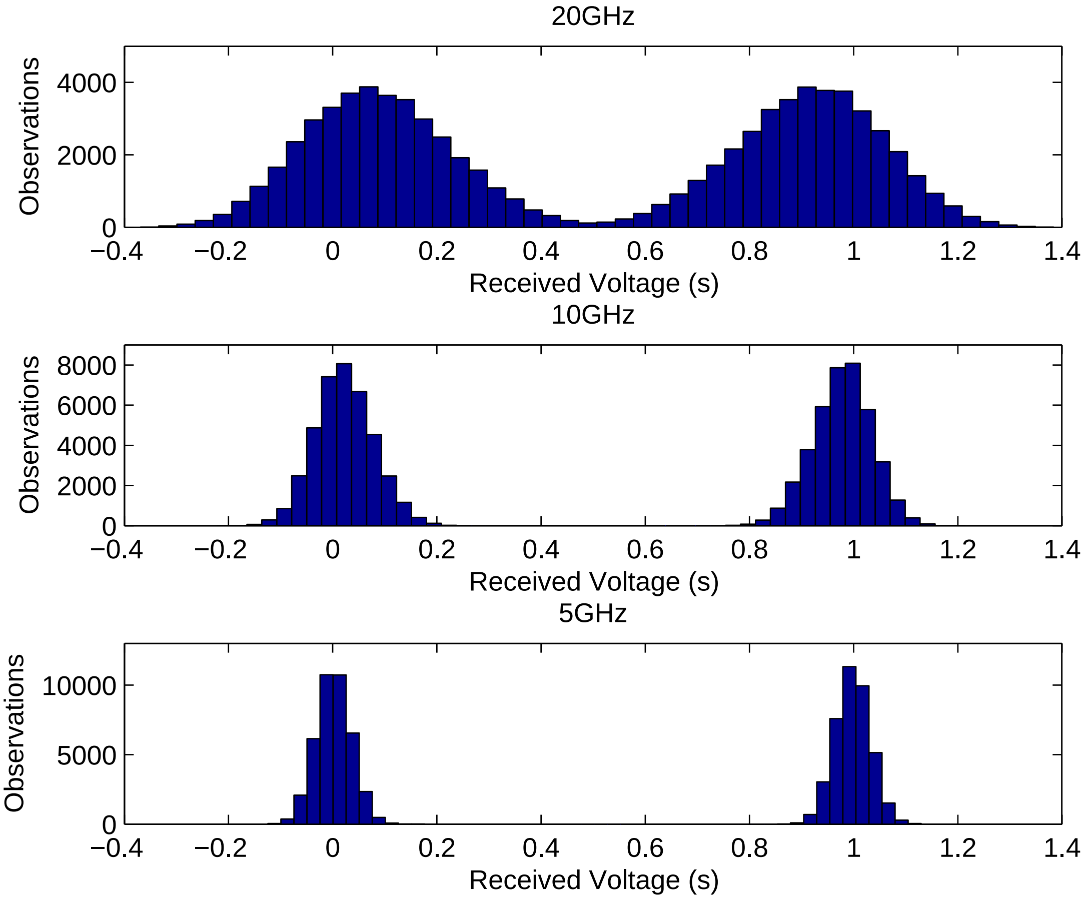

Fig. 24. The output voltage distribution of an 8 bit wide 1 mm long bus.

C. Code Construction

Before describing the coding scheme, here is a review list of the traits desired of a practical code.

- High data reliability - The codec should not leave any errors in the data-path.
- Low signal propagation delay - The signal propagation delay through the encoder, the wires and the decoder combined should be less than un-coded wires.
- Low energy consumption - The total energy consumed in the encoder, drivers, wires, sensors and the decoder combined should be less than the energy used in the un-coded system.
- Low coding overhead - The overhead of coding, i.e., the extra area used up by the encoding/decoding logic should be minimal. The number of extra wires added for redundancy should be low.

The two codes mentioned before, i.e., shielding and the AND code, are very easy to implement and result in a 1.83x speedup. However, they result in a 100% overhead in terms of the number of wires. This is prohibitive for implementation. A more complex coding scheme that possesses the same properties with a low overhead is desired.

If an n bit information is to be coded using a k bit code (k wires), and if S_i is a unique set of 2^n k-bit words, out of a total of 2^k words, then there are a total of $\binom{2^k}{2^n}$ possible sets. Out of these sets, a code (or, a set S_i) is chosen by *elimination* as described below.

- First eliminate the codes that have a high worst case delay pattern. Eliminate the codes $S_i, \forall 1 < i < \binom{2^k}{2^n}$ that fail the test,

$$C_c\left(S_i(x), S_i(y)\right) \leq C_{c,max}, \forall 1 < (x, y) < 2^n \tag{4.5}$$

where, $C_{c,max}$ is the maximum coupling capacitance (delay) pattern allowed, and $C_c(S_i(x), S_i(y))$ is the worst coupling capacitance (delay) when transiting from code word $S_i(x)$ to $S_i(y)$.

- Next eliminate the codes based on their error correction capability. The code correction capability is determined by the minimum hamming distance D_{min} between any two code words. If e error correction is required, then $D_{min} \geq 2e + 1$. Based on this, eliminate the codes that fail the following test.

$$D\left(S_i(x), S_i(y)\right) < D_{min}, \forall 1 < (x, y) < 2^n \tag{4.6}$$

where, $D(S_i(x), S_i(y))$ is the hamming distance between $S_i(x)$ and $S_i(y)$.

Let **S** be the set of all S_i that are not eliminated, out of $\binom{2^k}{2^n}$ initial codes. The choice of the final code C is made based on the average energy consumption.

$$C = \arg\min_{S_i} \sum_{x=1}^{2^n} \sum_{y=1}^{2^n} E(S_i(x), S_i(y)) \tag{4.7}$$

As mentioned before and seen from the algorithm above, this code is constructed using an exhaustive search. A formal technique is difficult to have due to the non-linearity of the low-delay (Crosstalk avoidance) codes. The codes presented are still practical because this search has to be performed only once for a given bus width and a given technology. These codes can be incorporated into the libraries of the design tools and can be transparent to the designer.

Based on the above code search method, table IV presents the values of the number of wires k as a function of the information width n and the code constraint.

The rows of the tables represent the constraint, while the columns represent the information width n. It is seen that for a 4-bits data bus, 5, 5 or 8 wires are required for 3C, 2C or 1C codes respectively. Compared to the Shielding and the AND codes, this is a huge improvement in the overhead requirement.

Table IV. Wire Requirement for Various Codes as a Function of Information Width

	n=2	n=3	n=4	n=5	n=6	n=7
Max. 3C Allowed	2	4	5	6	7	8
Max. 2C Allowed	2	4	5	7	8	10
Max. 1C Allowed	3	6	8	11	13	16
SEC	5	6	7	9	10	11
SEC, Max. 3C	5	7	8	10	11	13
SEC, Max. 2C	5	7	9	11	12	14
SEC, Max. 1C	6	9	11	14	16	19

An example code that eliminates 3C patterns in 4-bit wide data is presented in table V. It is named "4B5W" because 4 bits are transmitted over 5 wires. No two codewords exhibit a 4C or 3C pattern. A 1.88x speedup is achieved with just 25% wire overhead and a 2-level gate delay at the encoder. The energy consumption calculated based on equation 4.3 is proportional to $1.25 + 1.25\lambda$

Another example code that eliminates 2C patterns and correct single errors is presented in table VI. No two codewords exhibit a 4C, 3C or even 2C pattern. Also, each pair of codewords has a hamming distance of at least 3 units. A speedup of $(1 + 4\lambda)/(1 + \lambda) = 17/5 = 3.4$ is expected, while a speedup of 2.7 is seen in fig. 25. The discrepancy is attributed to the inductive effects which become prominent at speeds around 100 GHz (10ps). The energy consumption based on equation 4.3 is proportional to $2.6523 + 1.3789\lambda$.

Table V. An Example 4B5W Code for 3C Elimination

Data Bits	Code
0000	00000
0001	00001
0010	00110
0011	00011
0100	01100
0101	00111
0110	01110
0111	01111
1000	10000
1001	10001
1010	11000
1011	10011
1100	11100
1101	11001
1110	11110
1111	11111

Table VI. An Example 4B11W Code for 2C Elimination and Single Error Correction

Data Bits	Code
0000	00000000000
0001	00000000111
0010	00000011100
0011	00001111111
0100	00111000000
0101	00111000111
0110	00111111100
0111	00111110001
1000	11100000000
1001	10000011111
1010	10001111100
1011	11100000111
1100	11100011100
1101	11111000001
1110	11111110000
1111	11111111111

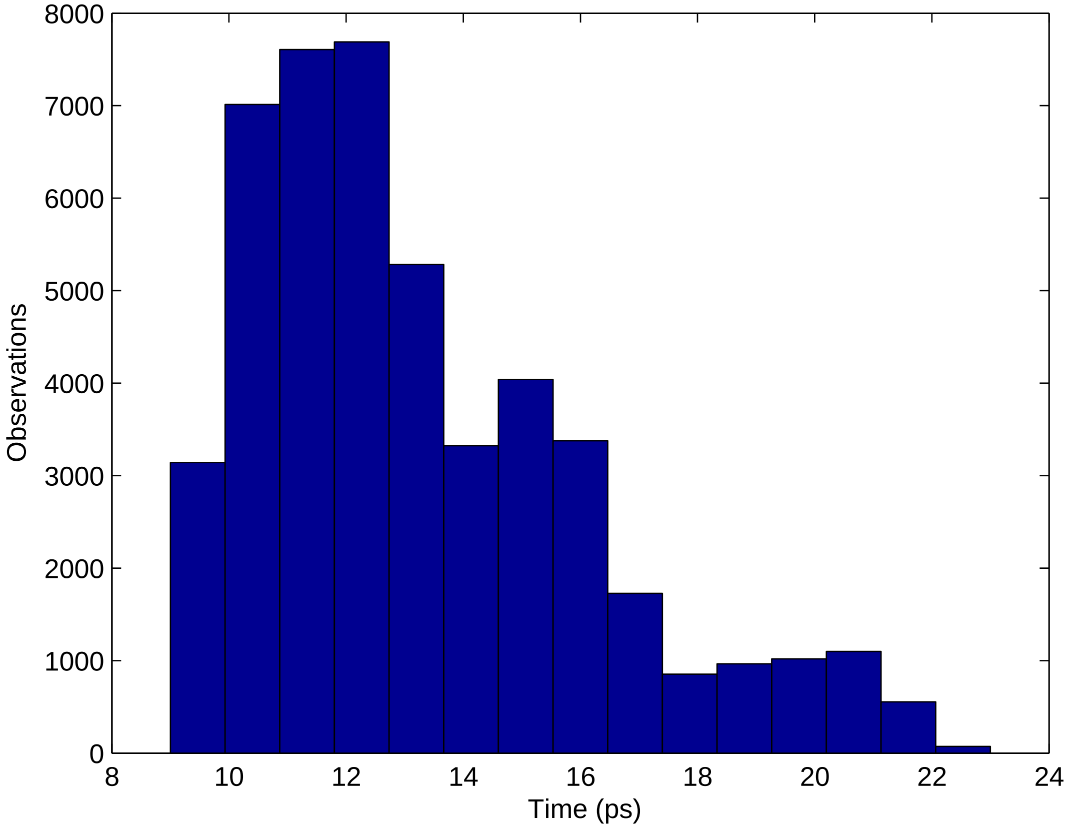

Fig. 25. The signal delay distribution of a 1 mm long bus with 4B11B code.

1. Comparison with Existing Codes and Capacity

As mentioned before, a variety of codes were introduced in [15], with a variety of features. These codes are a cascade of CAC, LPC and single error detection/correction (SED/SEC) codes. Table VII compares the codes in [15] to the codes developed here (Shielding, AND, 4B5W, 4B11W) based on delay, energy, overhead and error correction capability. The rate n/k of a code is the number of information bits per transmitted symbol. The energy is calculated based on equation 4.3.

Table VII. Comparison of Speedups and Overheads of Different Codes for a 4-Bit Bus

Code	Delay $\propto$	Code Rate	Energy	ECC
Hamming	$1 + 4\lambda$	0.57	$1.75 + 3.00\lambda$	SEC
HammingX	$1 + 4\lambda$	0.5	$1.75 + 3.00\lambda$	SEC
BIH	$1 + 4\lambda$	0.44	$1.78 + 3.25\lambda$	SEC
FTC+HC	$1 + 2\lambda$	0.28	$2.09 + 3.20\lambda$	SEC
BSC	$1 + 2\lambda$	0.44	$2.25 + 2.00\lambda$	None
DAP	$1 + 2\lambda$	0.44	$2.25 + 2.00\lambda$	SED
DAPX	$1 + 2\lambda$	0.4	$2.50 + 2.00\lambda$	SED
DAPBI	$1 + 2\lambda$	0.36	$1.81 + 1.75\lambda$	SED
Shielding	$1 + 2\lambda$	0.57	$1.00 + 1.50\lambda$	None
AND Code	$1 + 2\lambda$	0.57	$1.56 + 1.12\lambda$	None
4B5W	$1 + 2\lambda$	0.8	$1.25 + 1.25\lambda$	None
4B11W	$1 + 1\lambda$	0.3636	$2.65 + 1.37\lambda$	SEC

Figure 26 plots the relative speeds of information transfer per wire as a function of $\lambda = C_c/C_g$ for the various codes. The comparison is done based on the ratio of speedup / number of wires. This is a better metric compared to just speedup because

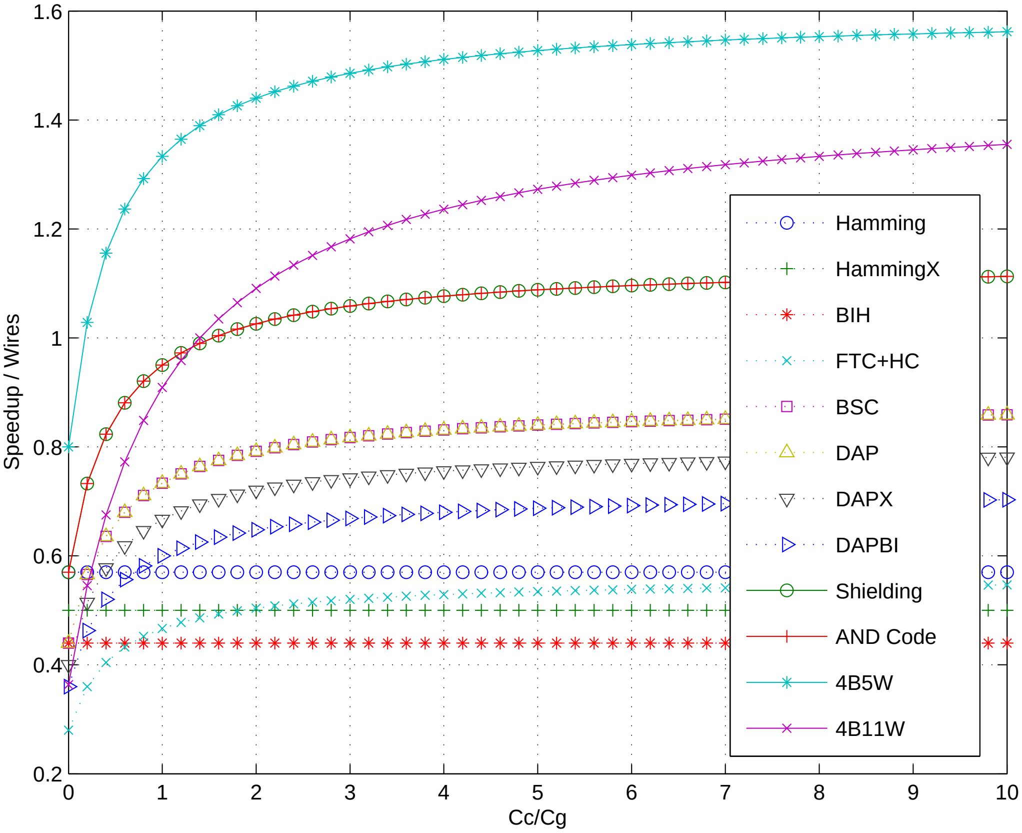

Fig. 26. The comparison of speedups/overhead of the various codes

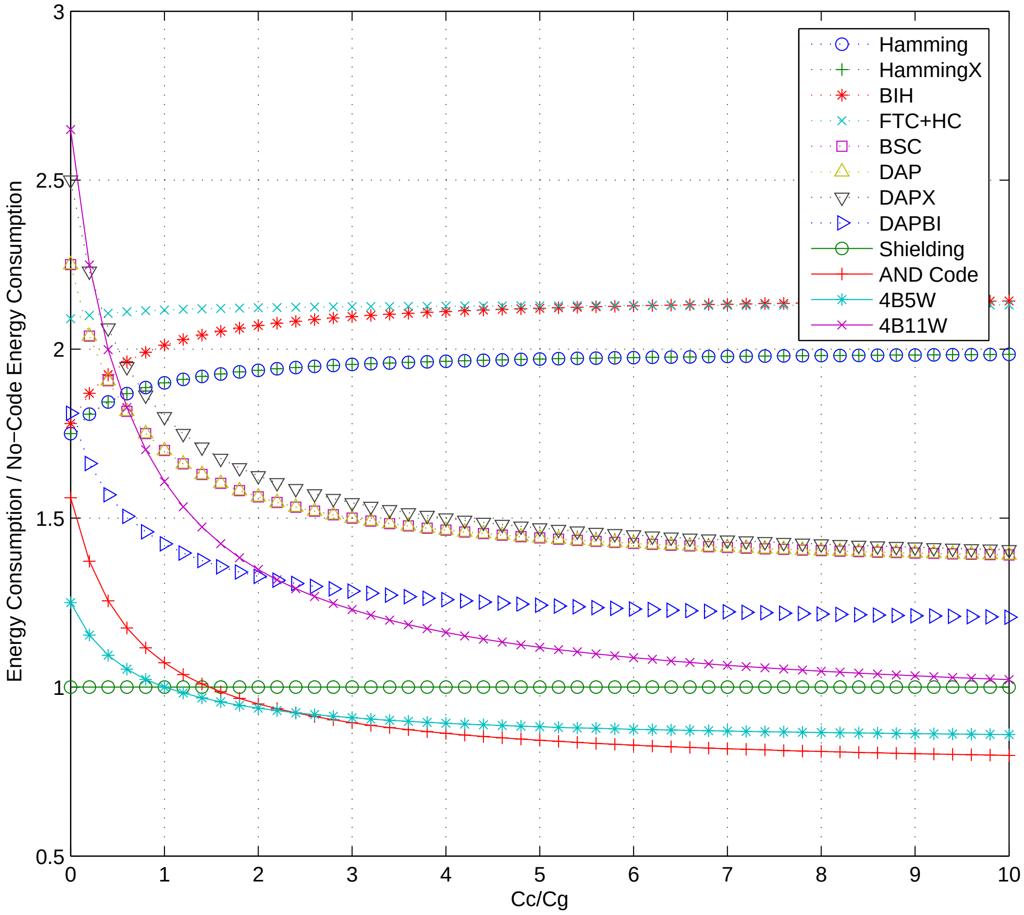

Fig. 27. The comparison of energy consumption of the various codes

if overhead cost was not included, then an infinite speedup can be had by placing a large number of independent un-coded 4-bit buses in parallel to each other. Among the SEC codes, the 4B11W code out performs all other codes, while among the non error correcting codes, the 4B5W code is the best.

Figure 27 plots the energy consumption relative to the un-coded system as a function of λ for the various codes. Among the SEC codes, the 4B11W code out performs all other codes for $\lambda > 2$, while among the non error correcting codes, the 4B5W code and the AND code are the best depending on the λ.

This section presents a unified coding technique for low energy, low delay, reliable transmission on on-chip buses. The codes developed here speed-up the transmission by avoiding and exploiting the effects of crosstalk, reduce the total energy consumed by minimizing signal switches, and provide reliability using communication-theoretic ECC. The codes presented herein show gain for even small wire-lengths, and therefore can be used in conjunction with the existing buffer insertion techniques to achieve even higher speedups. The codes implemented for a 4-bit wide bus in the 0.1μm process show a speedup of 2.7x with a 4 Wire overhead, and 1.83x for a 1 wire overhead. The latter code is called the 4B5W code herein, because it translates 4 bits into 5 wires. If error correction is a requirement then a 2.7x and 1.83 x speedup can be had with 7 and 5 extra wires respectively. The former code is called the 4B11W code. The total energy consumed in the 4B5W code and the 4B11W code are respectively 86% and 120% of the energy consumed in the no-code system. The 20% extra energy consumed in the 4B11W should be seen in light of the fact that it even corrects single errors. Also, as the feature size reduces, i.e., the ratio C_c/C_g increases, this energy consumption is even better.

Even though the codes presented here are based on exhaustive search of the entire code-space and not a formal closed form technique, they are still practical.

This is because the search does not need to be done at the design time. For every given bus width and technology, there is a unique code. Based on the bus parameters, a library can be constructed and incorporated into the CAD tools, making the bus coding scheme transparent to the designer. For very wide buses, when even a one-time solution is difficult to have, a “divide-and-conquer” technique can be deployed breaking the bus into smaller manageable segments.

CHAPTER V

LDPC CODES

Low density parity check (LDPC) codes, introduced by Gallager [39], are a class of linear block codes that perform very close to the turbo codes [40]. In [41], a use of LDPC codes has been proposed in Core-Network Interfaces (CNI) in the modern NoC architectures. Results claiming energy and latency savings have been provided.

Recent research [42–56] have pointed out their potential for a low cost, low latency hardware implementation. Due to this property, a lot of research has been done to find their suitability in different communication media. These codes have been shown to achieve near Shannon-limit performance in Wireless AWGN channels. At the same time these codes can result in significant power reduction in on-chip global and semi-global interconnects. These different applications demand a variety of LDPC coder decoder structures. This work aims to develop a decoder architecture which is "programmable" and is effective even in the high speeds required in on-chip networks.

As described in detail later, and as shown in fig. 28, the LDPC decoding process consists of computing several updates called the check node updates and the variable node updates. The decoder architectures range from those that handle one update at a time (serial) [42] to those that compute all the updates at the same time (parallel) [43, 44, 48]. While the serial architecture has a low hardware cost and are very easy to implement, they suffer from low throughput. On the other hand, while the parallel implementations have a very high throughput, they have a very high hardware resource requirement and have a very complex design. Several other designs that are a combination of the above are presented in [42–56].

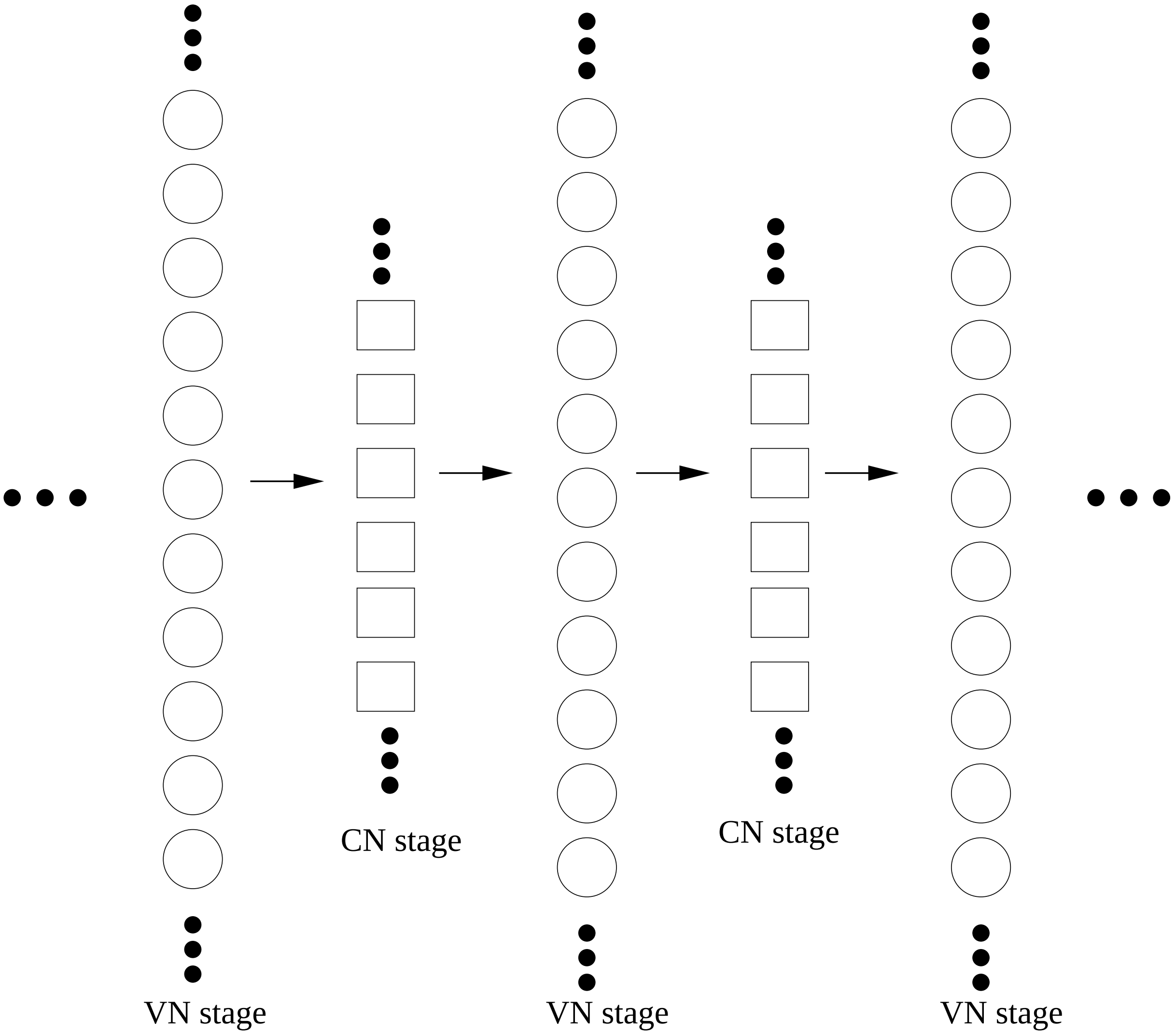
CN stage
CN stage
VN stage
VN stage
VN stage

Fig. 28. Schematic of the LDPC decoder.

A. LDPC Decoding Process

At the transmitter, redundancy bits are added to a data block of length k to derive a code word of length n. These redundancy bits are parity bits formed by linear combination of the k message bits. The rate of the code is $R = k/n$.

The receiver has exactly $n-k$ parity check equations to determine the correctness of a received word. These $n-k$ linear equations can be uniquely represented by a parity check matrix H of size $(n-k) \times n$, where the columns represent the received symbols and the rows represent the parity check equations. All elements of this matrix are either 0 or 1. A 1 in the (x, y) position means that the yth received symbol participates in the xth parity check equation.

The H of a regular (n, j, ρ) LDPC code has exactly $j(\geq 3)$ ones in each column and exactly $\rho(> j)$ ones in each row. Let C be the set of n bit long code words, then all vectors $x \in C$ satisfy the relation $H.x^T = O$, where O is a vector having $n-k$ zeros.

An H matrix can also be seen as a bipartite graph, where the rows form the $n-k$ check nodes (CN) on one side and the columns represent the n variable nodes (VN) on the other. A 1 in the (x, y) position means a connection between the yth VN and the xth CN.

The decoding process is an *iterative message passing* algorithm, where updated messages run sequentially between the two sides of the bipartite graph. At each stage, updates are calculated and are called the VN update (VNU) or the CN update (CNU) respectively. The updates from the VN stage are passed on to the CN stage. The CN stage performs its own updates and passes the signal back to the VN stage.

These updated messages are incrementally improving estimates of the transmitted bits. Each node takes the *a priori* probabilities of the bits as inputs and calculates the *a posteriori* probabilities [40] based on them. These probabilities, also known as extrinsics, are expressed as a Log Likelihood ratio (LLR) defined as

$$L(m) = L(c_m|r_m) = \log\left[\frac{p(c_m = 1|r_m)}{p(c_m = 0|r_m)}\right], \tag{5.1}$$

where, r_m is the value received from the communication medium and c_m is the binary estimate of it.

1. The Variable Node

Every VN has j LLR inputs, called $L_{cb,1}, L_{cb,2}, \ldots, L_{cb,j}$ from the preceding CN stage. It also has one LLR input L_{ch}, which is the information of the bit received from the communication medium. Based on these inputs, it calculates j outputs, $L_{bc,1}, L_{bc,2}, \ldots, L_{bc,j}$ according to the equation (5.2).

$$L_{bc,i} = L_{ch} + \sum_{k \neq i} L_{cb,k}. \tag{5.2}$$

2. The Check Node

The CN has ρ LLR inputs from the preceding VN stage called $L_{bc,1}, L_{bc,2}, \ldots, L_{bc,\rho}$. It calculates the ρ outputs $L_{bc,1}, L_{bc,2}, \ldots, L_{bc,\rho}$ according to the equation (5.3).

$$\begin{aligned}
L_{cb,i} &= -\prod_{k \neq i} \mathrm{Sign}(L_{bc,k}) \times \psi^{-1}\left(\sum_{k \neq i} \psi(L_{bc,k})\right). && (5.3)\\
\psi(x) &= log\left(\tanh\left(\left|\frac{x}{2}\right|\right)\right). && (5.4)\\
\psi(x) &= -\psi^{-1}(x). && (5.5)
\end{aligned}$$

B. LDPC Decoder Design Issues

Several papers showing the implementation of the LDPC decoder have been presented recently. While all these approaches emphasize the potential of LDPC codes as a powerful, low cost, low latency solution for error control, they also bring out serious concerns that need to be addressed for better decoder design. The most important issues while designing a parallel decoder architecture are

1. Routing Complexity

The parallel design is nothing but the implementation of the bipartite graph, including all the nodes and edges. The two parts of the bipartite graph are connected by exactly $n \times j$ edges. Implementing this many edges even for a reasonable length code poses difficult routing problems.

2. Quantization

The performance of the LDPC code improves with the precision of the LLRs used at the decoder. Even though increasing precision results in better decoder performance, it also means higher hardware requirement and therefore higher cost. Fixing the precision of the LLRs is therefore an optimization problem and is an active research area.

a. Performance of LDPC Codes

The decoding of the LDPC codes is an iterative process. LLR updates are calculated at each stage and are passed on to the next stage. These updates are called the VNU and the CNU respectively. The performance of the LPDC codes depends on the precision of the messages that are passed through. Let ϵ_{vn} and ϵ_{cn} be the precision of the

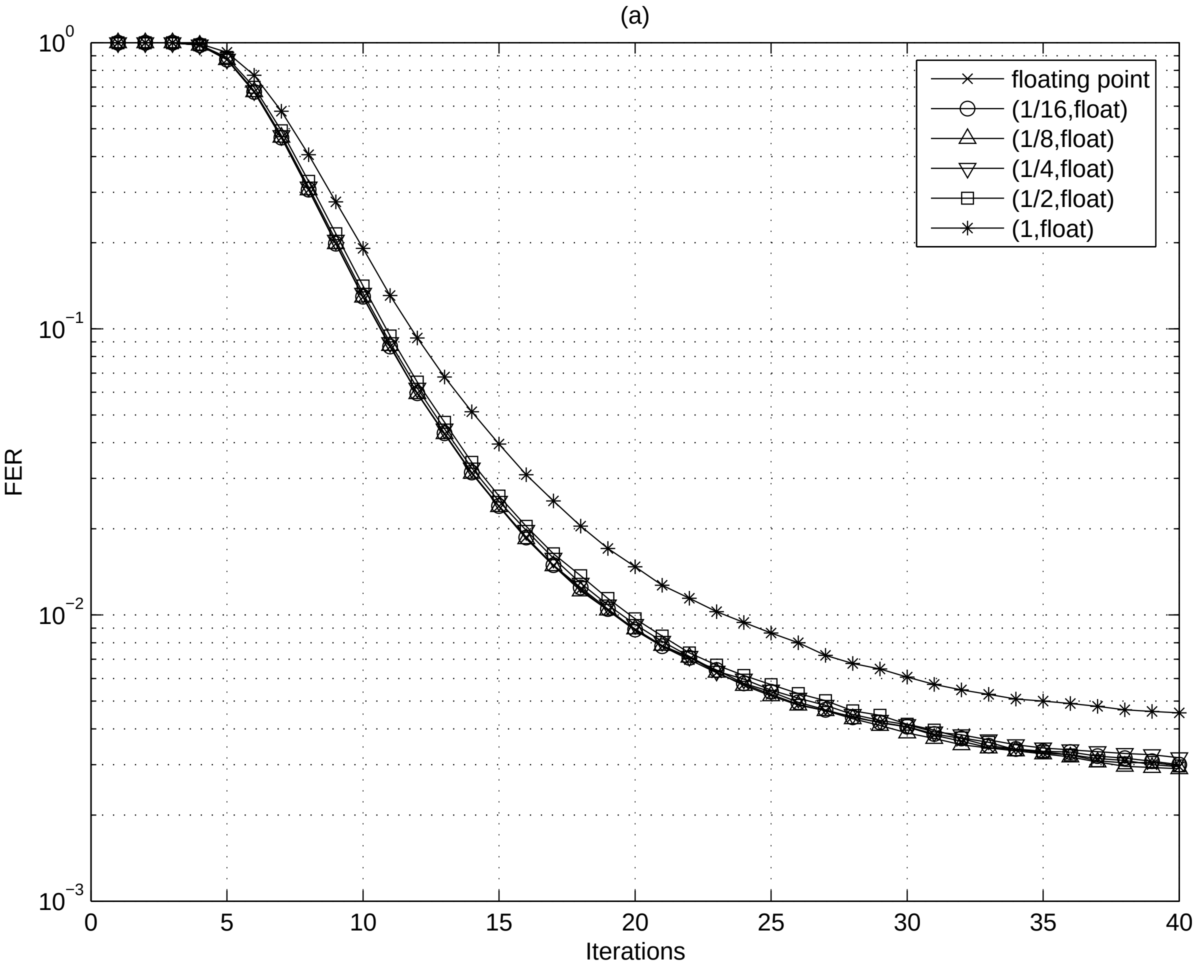

Fig. 29. The comparison of various implementations of the LDPC decoder with varying ϵ_{vn} at 2.3dB SNR AWGN channel.

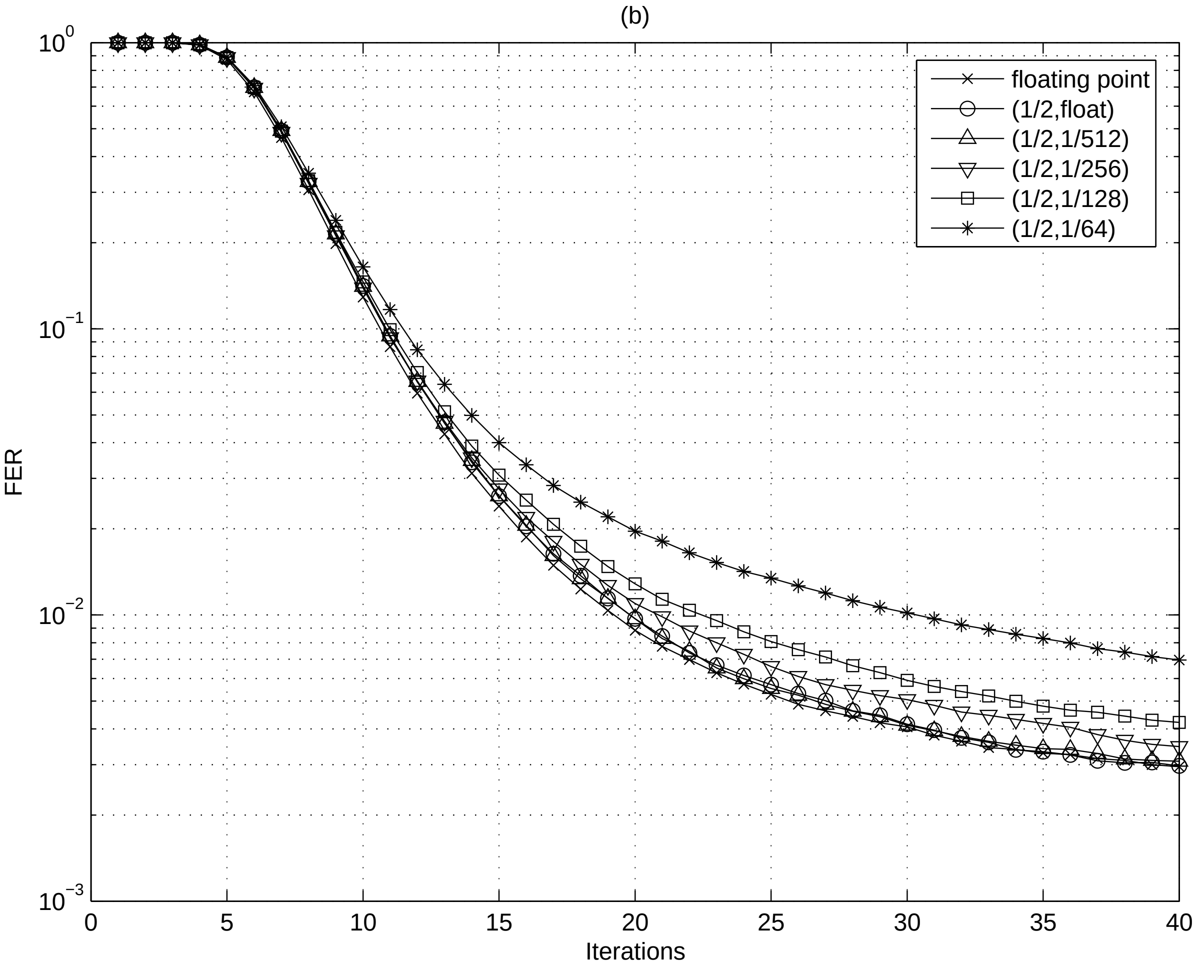

Fig. 30. The comparison of various implementations of the LDPC decoder with varying ϵ_{cn} at 2.3dB SNR AWGN channel.

LLR values at the VN and CN respectively, in the $(\epsilon_{vn}, \epsilon_{cn})$ decoder implementation. Fig. 29 and 30 describes the LDPC code performance as a function of ϵ_{vn}, ϵ_{cn} and the number of iterations. The experiments are conducted for an Additive White Gaussian Noise (AWGN) channel for a 2.3 dB SNR. It is seen that the performance does not suffer until $\epsilon_{vn} = 1/2$ and $\epsilon_{cn} = 1/128$. The optimal operating point is therefore selected as $(1/2, 1/128)$. For a maximum LLR value of ± 4, this corresponds to 5 bits and 11 bits precision at the VN and CN respectively.

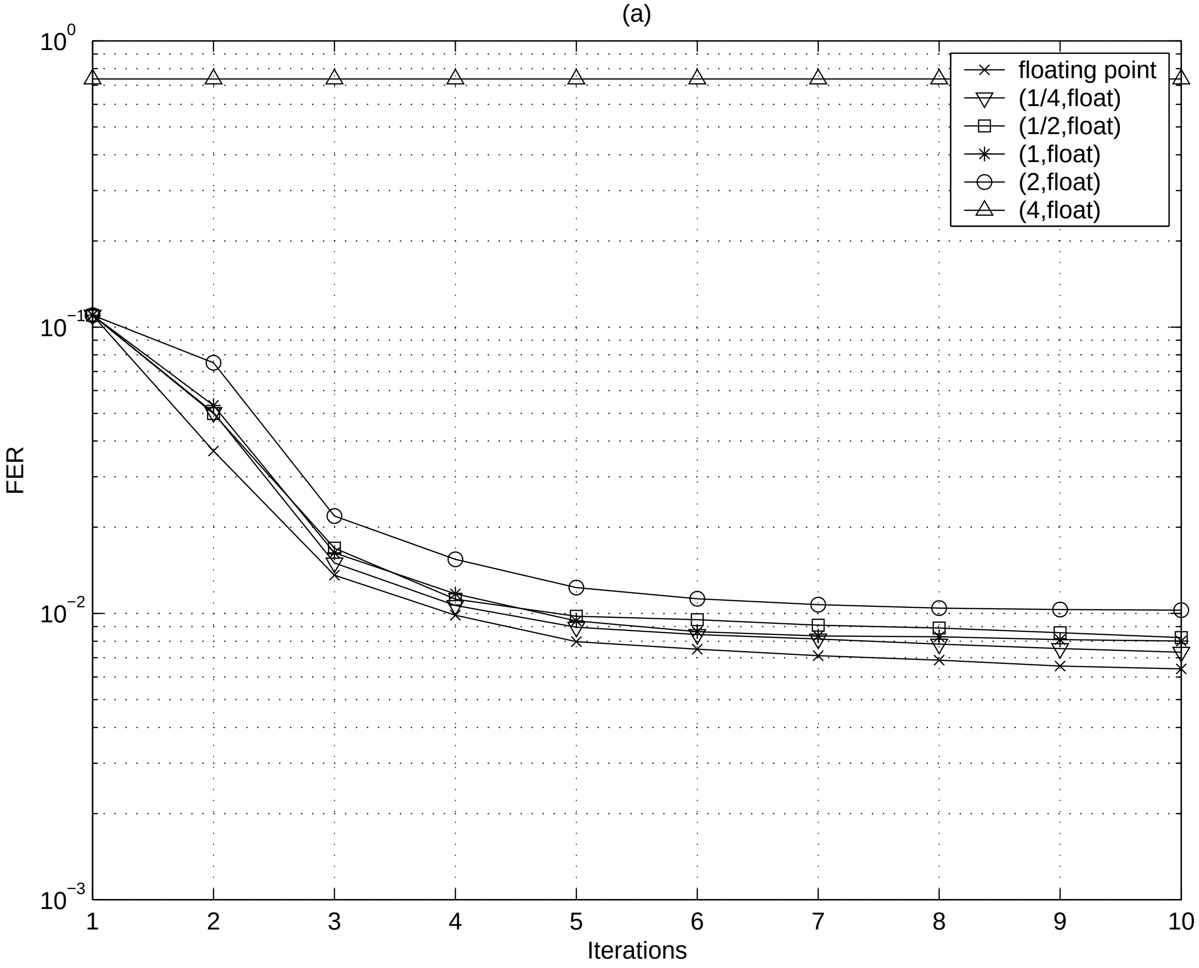

Fig. 31. The comparison of various implementations of the LDPC decoder with varying ϵ_{vn} at 10^{-2} BER BSC channel.

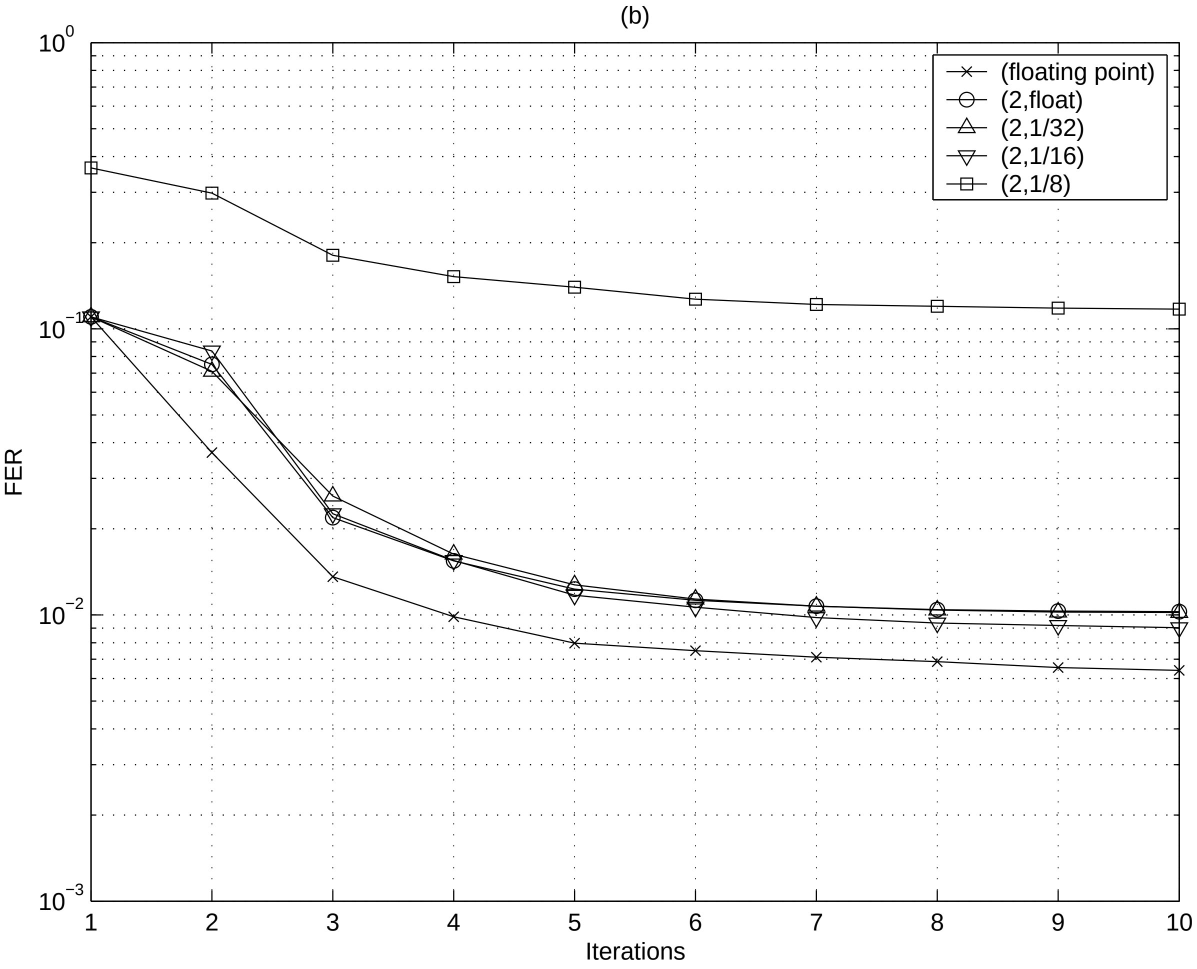

Fig. 32. The comparison of various implementations of the LDPC decoder with varying ϵ_{cn} at 10^{-2} BER BSC channel.

The same experiment is repeated for a 10^{-1} BER BSC Channel. The results are shown in fig. 31 and 32. The optimal operating point selected is $(2, 1/16)$. Again, for a LLR value between ±4, this implies a precision of 3 bits and 8 bits at the VN and CN respectively.

C. Decoder Architecture for On-chip Interconnects

Of the several types of architectures, only the fully-parallel architectures are capable of providing a meaningful throughput for on-chip buses. The LDPC codes are merely an implementation of a linear parity check matrix H. Also, as different communication systems have different error control requirements, and hence have completely different structures of this matrix H, it is a desired trait that the LDPC decoder hardware design be versatile and reusable in a variety of applications. For example, some parts of a chip may be more error-prone than others, and therefore a different type of code may be required for each component.

The adaptation of the hardware to a variety of H matrices can be used for

1. Same hardware for different wire lengths, requiring different error protection.
2. Field programmable data encryption for saving data into DRAM's.

A parallel implementation of the decoder has exactly $n \times j$ messages going from one stage to another. An arbitrary structure of H means that these messages can start from any arbitrary location on one side and end up going to any arbitrary position on the other side. Any parallel decoder implementation will involve carrying these $n \times j$ messages from one side to the other precisely reordering them to deliver the correct message to the concerned node. This operation is similar to a sorting operation, where the output is arrived at by reordering the input values. Based on this amazing similarity, this work describes the implementation of the channel router in a parallel LDPC decoder using a bubble sort algorithm.

1. Bubble Sort Algorithm

The bubble sort algorithm is based on comparison of two consecutive elements of a list. If they are found to be in the right order, they are left untouched, otherwise the values are swapped. Sorting the entire list of length x involves $\mathcal{O}(x^2)$ such comparisons.

The reordering of $n \times j$ messages can be done in a similar fashion by doing $\mathcal{O}(nj^2)$ one-to-one comparisons. As an illustration, consider the fig. 33, where 6 messages are to be routed from one end to the other. The small cells are switching devices with two inputs and two outputs. The switch has 2 states, it either passes the messages through without swapping them, or alternately, it swaps them. The states of these 13 switches can be altered to route any of the 6 messages from any arbitrary starting position to any arbitrary ending position. Moreover, there is no critical path as all path lengths are equal.

The routing of $n \times j$ messages in a typical LDPC decoder can be done in a similar fashion by laying out an array of the switching cells. Exactly $n^2j^2/2 - nj + 1$ switching cells are required and the longest path passes through $nj - 1$ cells.

2. The Switching Cell and Computing Nodes

A switching cell has two inputs, two outputs and a single bit flip flop, and 4 transmission gates. Based on the state of the memory element, the values from the inputs are passed straight through or crossed over. A schematic of such a cell is shown in fig. 34.

The variable nodes and the check nodes are implemented using several adders. According to equation (5.2), the VN has $j+1$ inputs and j outputs. In most practical

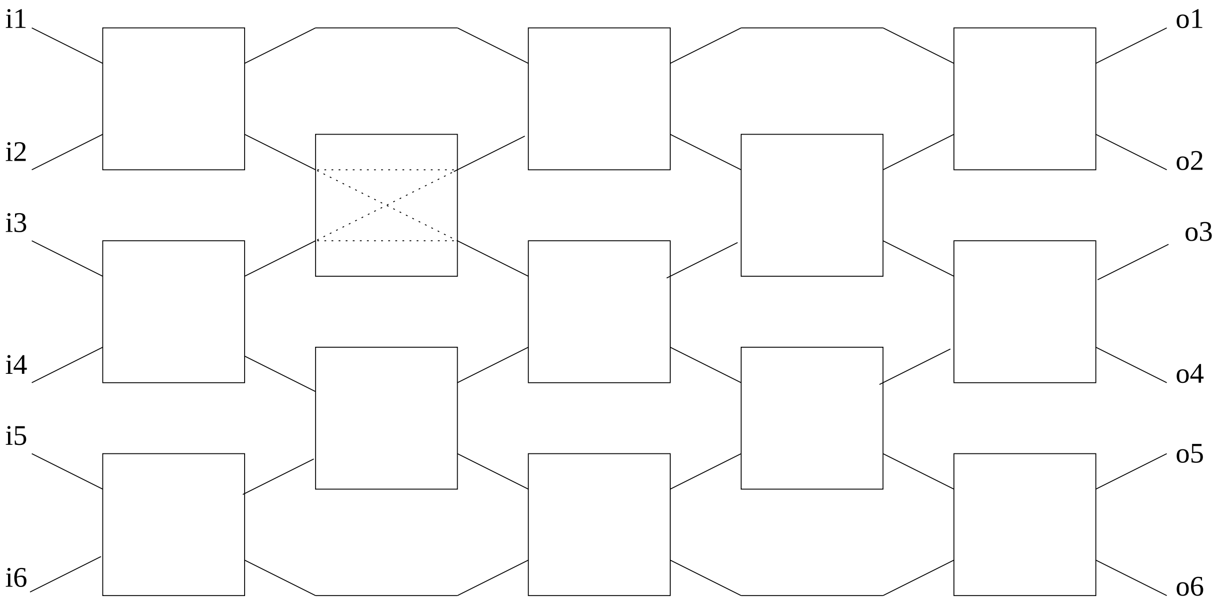

Fig. 33. A schematic for the Bubble Sort Algorithm.

situations, the j of an (n, j, ρ) code is kept at 3. This means that there are exactly 4 inputs into the VN and it produces 3 outputs for the CN. A schematic implementing the euqation (5.2) is shown in fig. 35.

The ρ however, may depend on rate of the code chosen, which is further a function of the channel conditions. For small wires (good channels) the ρ is very high, while for long wires (bad channels) ρ may be very low. For example, for a rate $R = 1/2$ code, if $j = 3$, then $\rho = 6$. Also, for a rate $R = 0.75$ code, if $j = 3$, then $\rho = 12$. The structure of the CN is therefore left programmable, depending on the value of ρ, the inputs to the check node are changed. The basic structure of the cell remains the same. It is an add-accumulate circuit, with all the ρ inputs added together after ψ transformations and then for each output a specific value is subtracted from the accumulated sum.

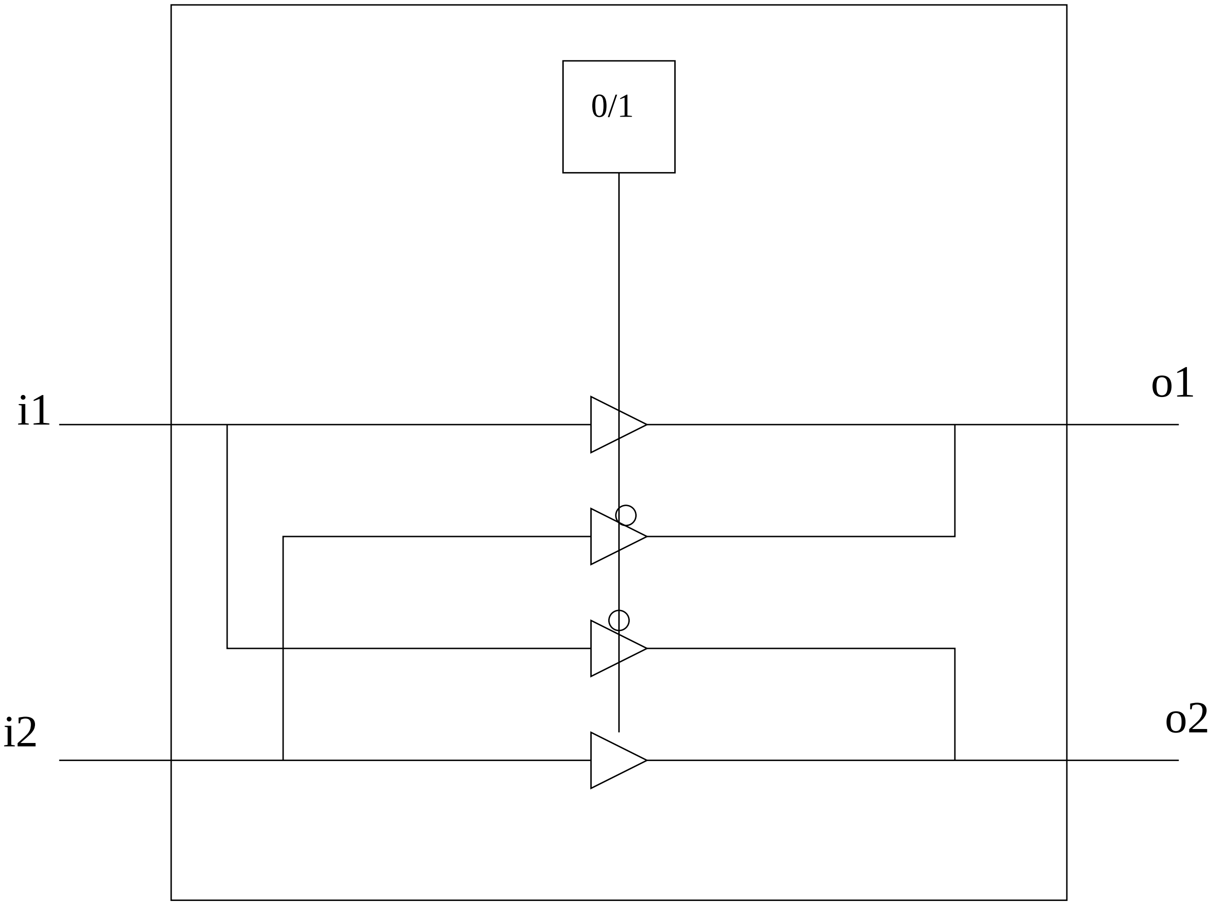
0/1
i1
i2
o1
o2

Fig. 34. The schematic for the switching cell.

3. The Decoder Architecture

A complete decoder can be constructed in a fashion similar to fig. 33. As an example, consider a $(255, 3, 6)$ code over a 255 bit wide bus. There are exactly 255×3 =765 messages going from one stage to another. The longest path will run through 764 stages, but is broken down into multiple cycles using pipeline stages. The complete structure is shown in fig. 36

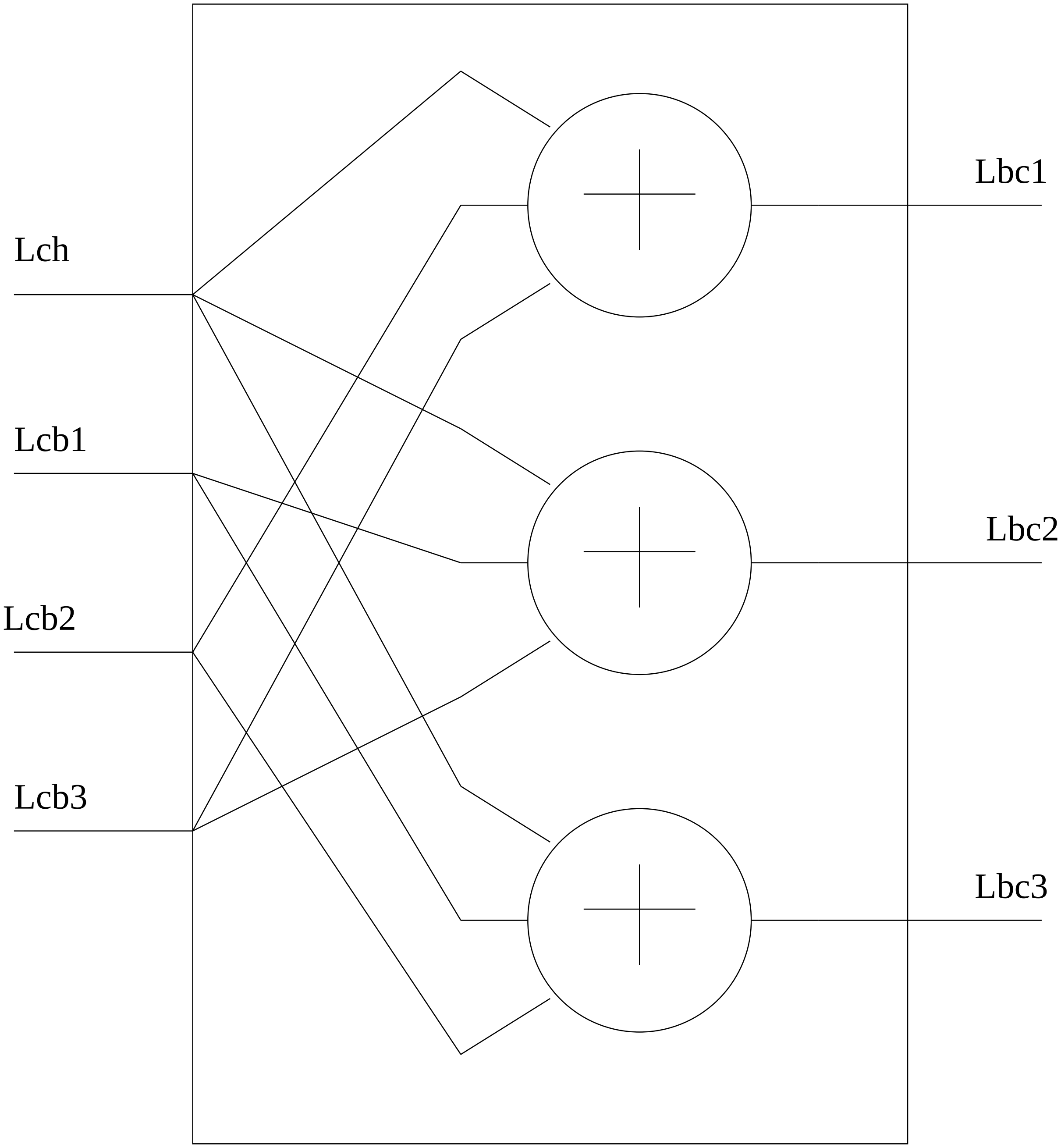
Lch
Lcb1
Lcb2
Lcb3
Lbc1
Lbc2
Lbc3

Fig. 35. The schematic for the VN.

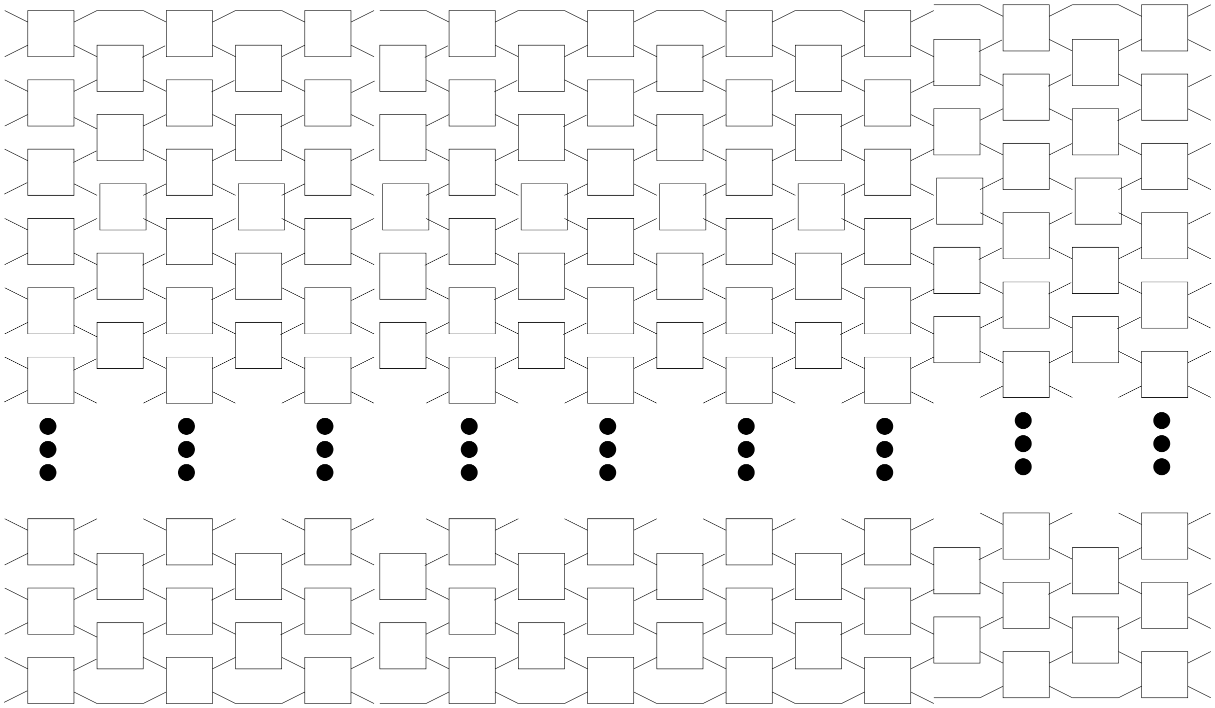

Fig. 36. The schematic for a single pipeline stage of the decoder.

CHAPTER VI

OTHER APPLICATIONS OF CODING - COMBINATORIAL CIRCUITS

With increasingly difficult scaling, it is questionable whether the future VLSI circuit designs will yield acceptable stability and performance necessary to accommodate the exploding computation speeds. Device improvements are being made, and there exist multitude of proposals/efforts for clever design/fabrication solutions including the 3D integration, stacking, and unconventional ways of performing computations. However the question remains: Can we continue to increase the density/size and yet, keep (or improve) the stability of data?

Conventional wisdom is that the density and reliability functions are generally inversely related. Yet, Moore's history evidently shows us that the only direction is up. With this optimism, we face a current pressing problem: Clocking a processor is limited to the flight time of data through the combinatorial circuit, often a dynamic in configuration, and at the latch, ready to stably drive the next stage as given in the following equation,

$$T_{ck} > T_d + T_s + T_h + \Delta T_{ck}, \tag{6.1}$$

where T_d is the worst case signal flight time, T_s is the set-up time, T_h is the hold time and ΔT_{ck} is the clock jitter.

The problem then is; as the processor complexity and fabrication technology elevate, this timing is becoming less and less well-defined and thus less predictable. With deep-submicron (DSM) technology, process variation adds to this problem. The reality shows us the fact that, no matter how conservatively (slowly) a processor chip is clocked, there is always a possibility that some bits will be corrupted as a result of timing violation. At times we blame this phenomenon on an uneventful cosmic ray particle that puzzles us with occasional soft errors (single-event upsets). Regardless of the source, we claim to solve this problem using Error Correction Coding (ECC) and other patch-up remedies at system/board level. This is akin to finger-plugging a hole in a dam. There will be multi-bit and burst-in-time errors to come. Worse, originating from mis-timing from a variety of causes. Without mitigating this issue at a chip-design stage, it is rather difficult to imagine that larger and faster devices can be had in the future.

The first step is to understand the delays that are born in a circuit. It is extremely difficult to have a well behaved signal delay that adheres to strict schedules. The delay, which is a summation of all the path delays has a distribution because of variations in operating conditions. The operating conditions that affect the time that the signal takes for traveling from the input to a particular node are

1. Data-Dependence - The signal or a logic state can arrive at a node via several different paths. These paths may have different lengths. The signal arrival time depends on the path that the signal takes to arrive at a node. This choice of path in turn is made by the combination of the input bits.

2. Coupling (Capacitive and Inductive) - When signals flow on nearby paths, they are influenced by each other. This interference may result in a deviation from the nominal signal delay.

3. Power Noise - When a lot of gates switch simultaneously, they drain a large current from the power supplies and they in turn deviate from their “rail” values. When this happens, the time that a gate takes to switch from one value to another also deviates from the normal.

4. Process Variations - The manufacturing technologies are non-ideal. The device

dimensions may vary from the design values. This results in a “non ideal” delay in switching, leading to non-deterministic delay patterns.

5. Temperature-Induced Delay - Depending on the surrounding temperatures, the devices in circuits may switch faster or slower, deviating the delay from the designed value.

As an illustration, delay failure rates are calculated using experiments on an AMD Duron 800 MHz CPU. Failure conditions are accelerated by worsening the operating frequency. The observations at accelerated operating frequencies are extrapolated to formulate a delay distribution. Based on this delay distribution a failure rate is calculated at normal operating conditions. For the 800 MHz processor, the accelerated frequencies chosen are within the 828 MHz to 834 MHz range. At these frequencies a time-to-failure of between 10s and 10000s is observed. Based on this, and the assumption that the delays are normally distributed, the delay distribution is calculated. The mean of the delay from the input to the output is 1000 ps and the standard deviation is 26.2 ps. It is seen that the normal operating clock period 1250 ps (1/800MHz) is about 9σ away from the mean delay. Another experiment on a 750 MHz Duron shows similar result, i.e., the normal operating clock period is about 9σ away from the mean value. A clock period 9σ away from the mean delay corresponds to a delay error rate of 10^{-37}. These experiments show that even the real-world processors possess a long tail in their delay distributions and sooner or later, these tails will lead to timing violations. Sooner, if the clock rate is pushed too hard.

Figure 37 shows the mean-time-to-failure for different operating clock frequencies for 800 MHz Duron. These errors are due to delay faults only. The values between 828 MHz and 834 MHz are from experimental observations while the others are extrapolation.

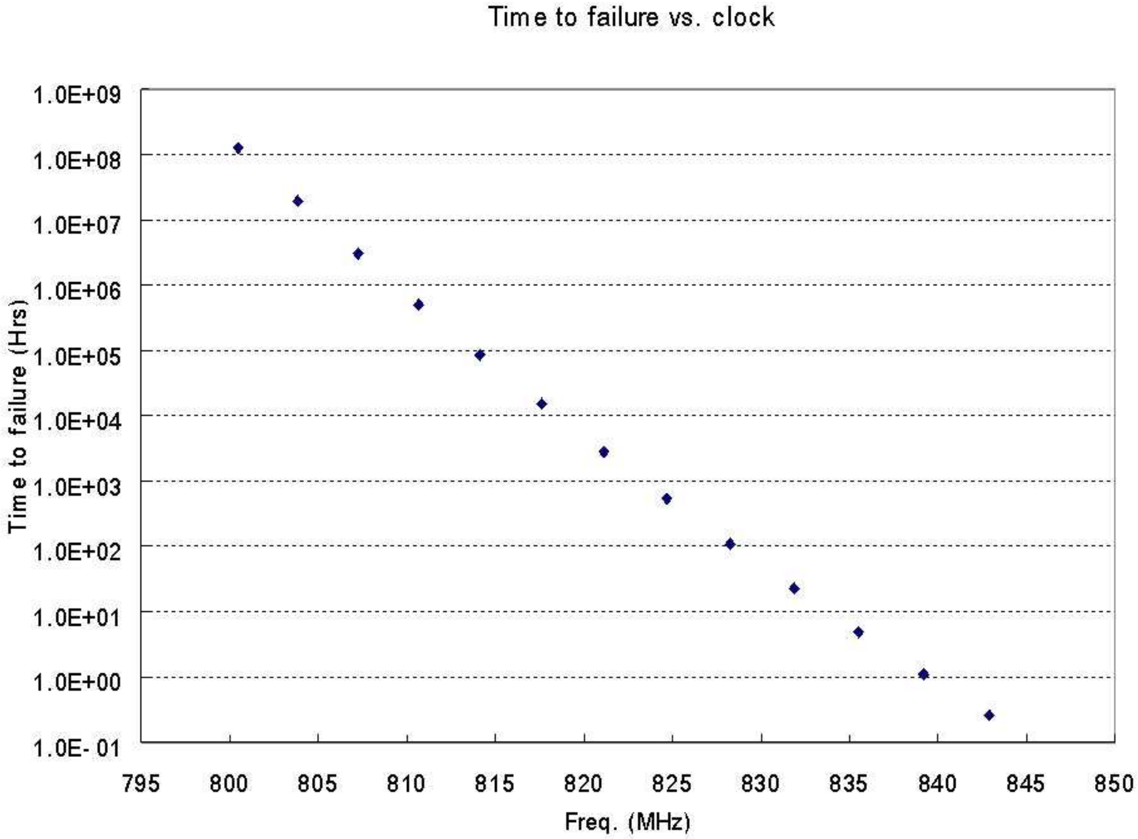

Fig. 37. TTF v/s clock for a 800 MHz Duron Processor.

The objective of this work is to develop a design methodology that leads to a denser integration and faster clock rate, assuming the properties of emerging fabrication technologies. The design must guarantee the signal integrity and timing property for a pipeline circuit structure. To achieve this, mis-timing or soft-errors must be prevented or managed if necessary. The alternatives [57–64] are:

1. conservatively-timed control and data structures,
2. redundancy or

3. coding.

Traditional coding or redundancy, however, do not suffice when cost/benefit is considered. Simple parity coding the I/O or memory will not achieve error correction necessary to contain errors in complex systems and, on the other hand, within a realistic timing budget of a pipeline stage, a more complex decoder such as Turbo or LDPC codes can not possibly be considered. Fault-tolerant techniques and yield-enhancing methods that involve redundancy have been considered excessive and are in no way considered cost-effective with the exception of very few “dependability-critical” applications.

A coding scheme that makes optimal balance between overhead and gain (suppression of bit-error-rate(BER)) is developed. The complete pipeline structure and the error control circuit must all be cohesively designed to optimize the gain and cost associated with the coding. In addition, control flow of the error management must not intrude into the normal operation of a pipeline and recovery must be within reasonable confidence bounds.

ECC is based on redundancy introduced in the circuit, which can be later used to detect/correct the errors that may occur in the “flight” of data. Based on this understanding we propose a method to produce redundancy or parity. These parity bits are used to detect any timing errors that may occur in a cycle. Our approach is to flush the pipeline completely, and re-initiate the pipeline at a slower clock, whenever an error is detected.

This section describes a novel way of constructing error control scheme for arbitrary combinatorial circuits. The extra parity bits required for error correction are generated by grouping together several primary outputs (POs). The POs in a single group should have low “delay correlation” between them. ISCAS benchmark circuits are used to study the feasibility of this approach. A coding hardware design and a novel recovery technique is also presented herein. A timing gain of upto 30% is achieved as a result of this approach

A. Error Correction Coding

Any parity based error detection/correction scheme works best if the error events that it tries to detect/correct are un-correlated. It is therefore necessary that the delay distributions of the participant bits in any parity group are independent of each other for efficient functioning. This, however, is rarely true in any realistic combinatorial circuits due to large number of re-convergent fan-outs. It is sought therefore, to partition the outputs into groups of bits with the lowest delay correlation possible between them.

1. Generating Parity Groups

It can be easily seen that some outputs are faster than the others. For calculating the error-event (long delay) independence, only the slower outputs should be considered. The outputs whose delay distribution 6σ point lies to the "left" of the clock period are "trimmed" out of consideration. In the remaining outputs, the correlation in the delay distributions is calculated. The outputs whose delay correlations are high should never be grouped together.

This new "correlation-matrix" can be treated as an adjacency matrix for undirected graph where an edge between two nodes indicates low correlation between them. Any two nodes that are connected, and are not separated by large number of hops, have a low correlation and can be used in a parity group. If one output has an error event, it has little bearing on the other output being in error. Based in this correlation matrix, all possible combinations of groups are achieved. The outputs in a group can be used to generate parity bits.

2. Proposed Architecture

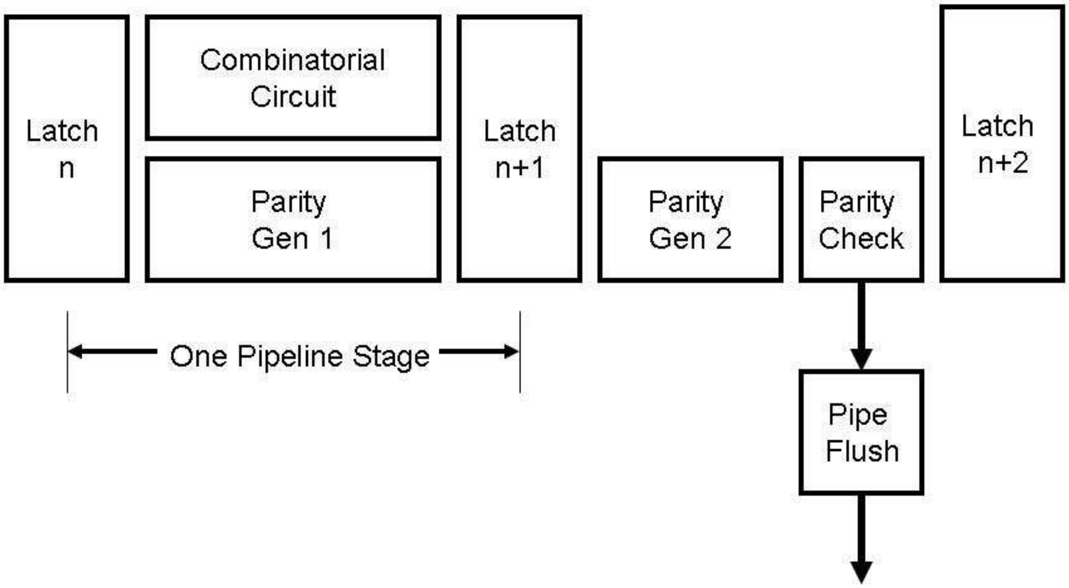
Latch
n
Combinatorial
Circuit
Parity
Gen 1
Latch
n+1
Parity
Gen 2
Parity
Check
Latch
n+2
One Pipeline Stage
Pipe
Flush

Fig. 38. The schematic of a error detection system.

Once the parity groups are decided, the logic circuit of the parity bits is synthesized. This is implemented independent of the original combinatorial circuit such that it has minimal effect on the original circuits as shown in the fig. 38. This is akin to a systematic code in communication theory. The only effect on the original combinatorial circuit sandwiched between two latches in a ordinary pipeline, is that it can operate at a faster than ordinary clock. The increased errors due to a faster aggressive clock will be corrected by the error correcting circuit, thereby resulting in a better timing yield. A parity generator circuit is broken into two parts and is inserted into this and the next pipeline stage. This allows a relaxed timing constraint for the parity generation making it less error prone. The parity checker follows the parity generator in the next pipeline stage to validate the outputs received from the combinatorial circuit. Whenever this circuit detects an error, the entire pipeline is flushed and re-initiated at a slower clock for a brief time interval.

B. Simulations and Benchmark Results

Table VIII. The ISCAS'85 Circuits

	Inputs	Outputs
c432	36	7
c499	41	32
c880	60	26
c1355	41	32
c1908	33	25
c2670	233	140
c3540	50	22
c5315	178	123
c6288	32	32
c7552	207	108

To show the gains achieved by error correction coding, delay patterns are studied on the ISCAS'85 benchmark circuits (table. VIII). The building blocks of all the circuits are logic gates and wires. For a delay simulation, the first step is to

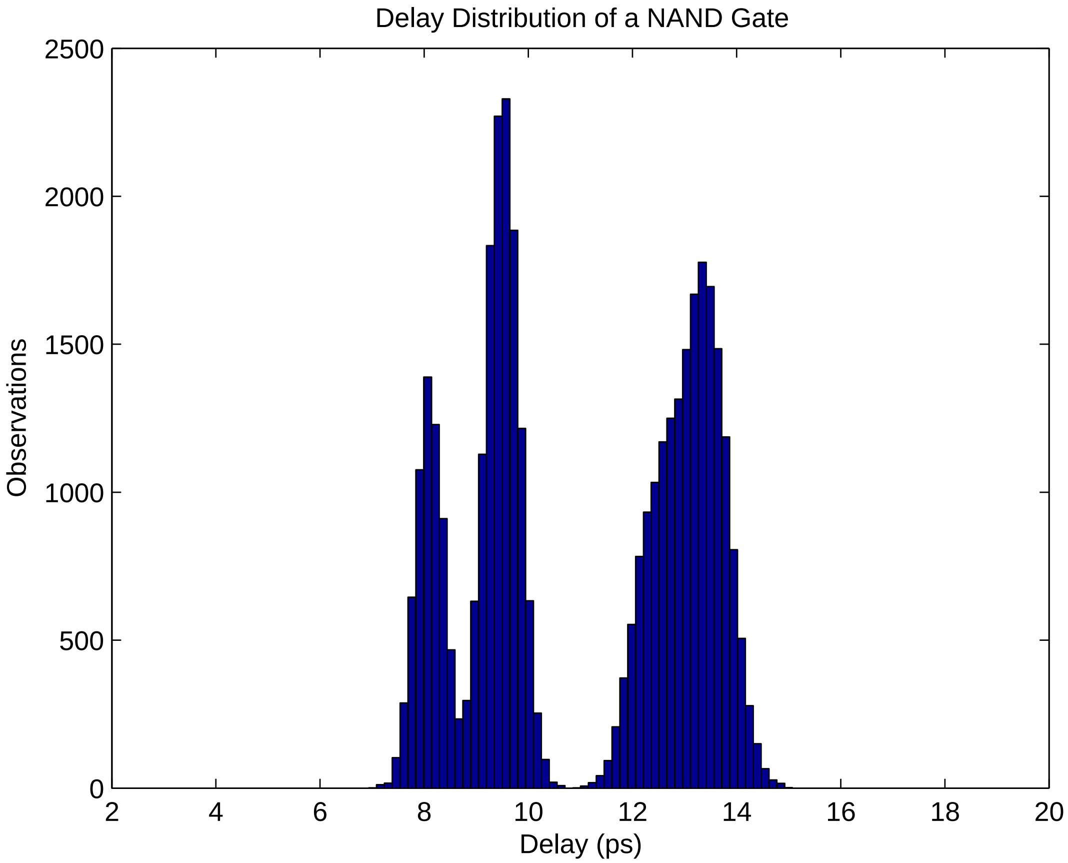

Fig. 39. Delay distribution of a 2-input NAND gate loaded by another 2-input NAND gate.

Table IX. The Delay of a NAND Gate for Different Input Combinations

Input Transitions for Output Switch	Output Delay
0,0 → 1,1	12.4 ps
0,1 → 1,1	9.5 ps
1,1 → 0,0	8.1 ps
1,1 → 1,0	13.4 ps

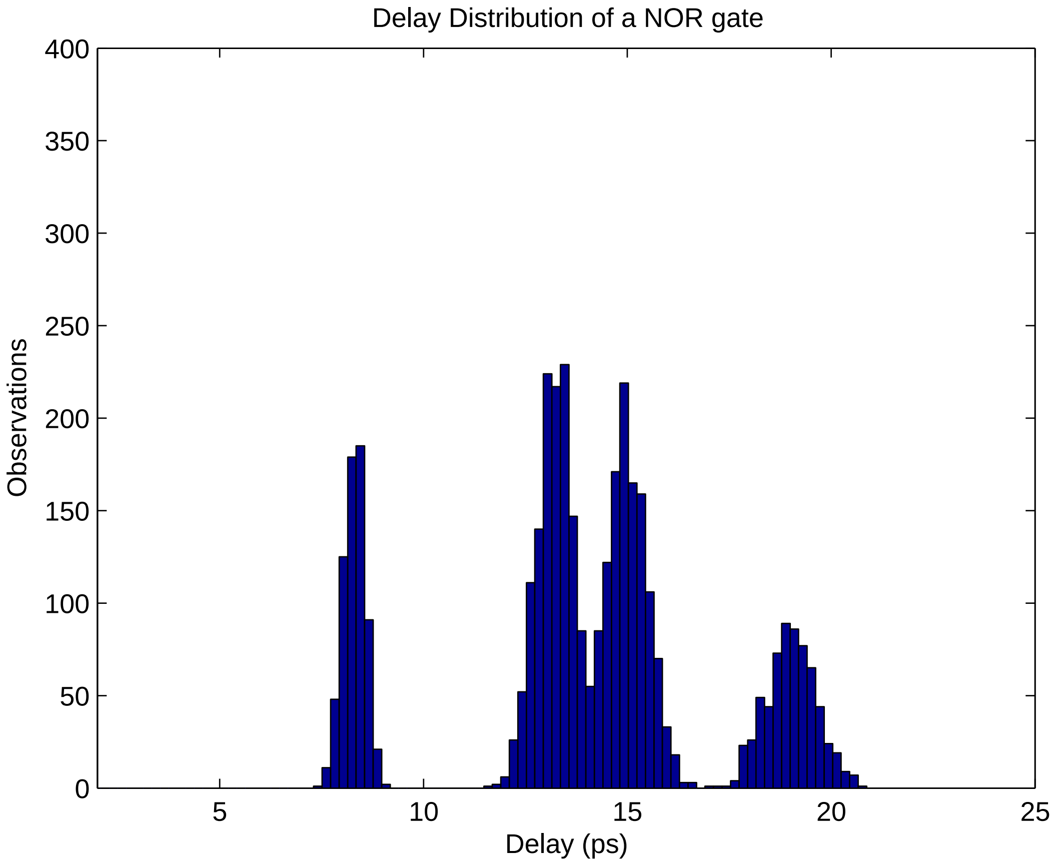

Fig. 40. Delay distribution of a 2-input NOR gate loaded by another 2-input NOR gate.

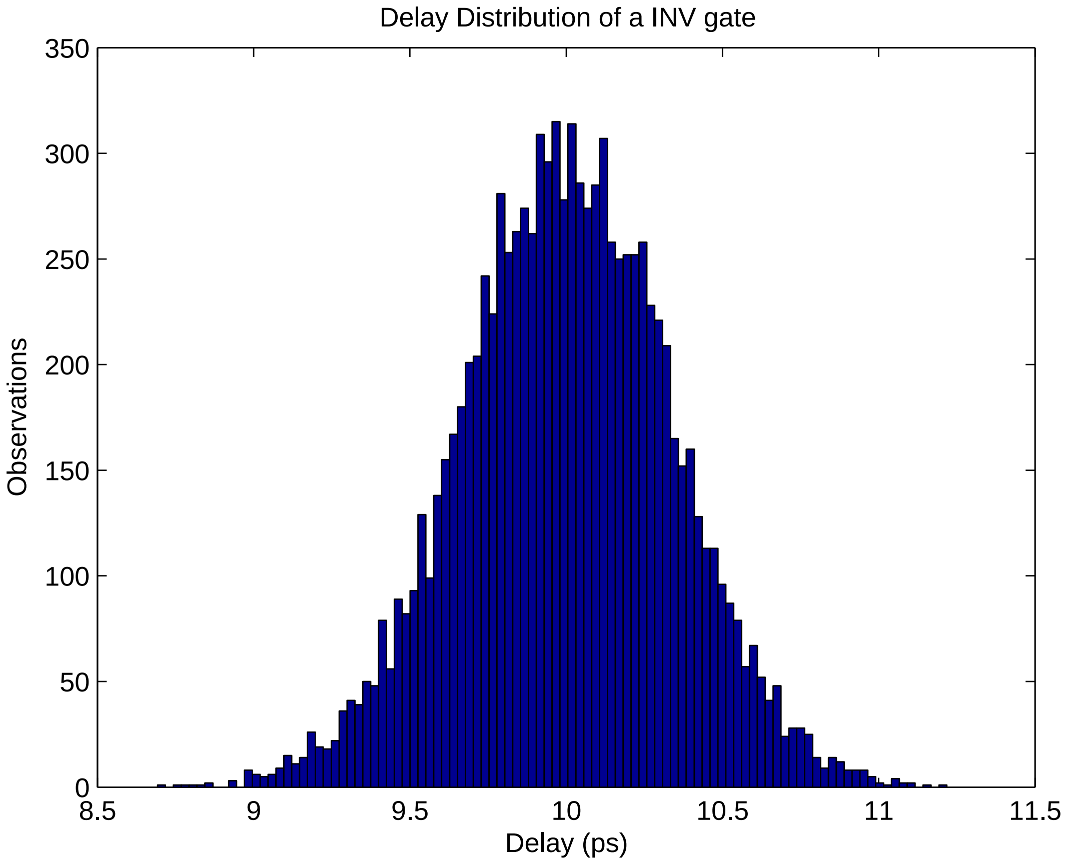

Fig. 41. Delay distribution of an inverter loaded by another inverter.

mathematically model the delays of small gates. The basic gates are a NAND gate, a NOR gate, and an Inverter. A single standard CMOS, a minimum sized 2 input NAND gate (0.1 μm technology) was simulated in spice with an inverter as its load. The delay was calculated for each different input combination that results in a output switch. Also it is assumed that the inputs are ordered for best delay performance of the gate. The results found are summarized in table IX.

Also, it is assumed that the 3σ delay variations due to the other factors are less than 10% of the median values. Based on this mathematical description, the distribution of the delay of a NAND gate is shown in fig. 39. The delay distributions of a NOR gate and an inverter are obtained in a similar fashion and are shown in figs. 40 and 41 respectively.

A bigger circuit, like the ISCAS'85 benchmarks, is constructed using the above smaller gates as basic building blocks. The gate level description of the circuits is obtained from [65]. This description was modified such that there are only the above 3 types of gates present in the circuit, namely a 2 input NAND, a 2 input NOR and an inverter. A large number of "monte carlo" experiments with different combinations of the primary inputs are conducted. For each experiment, the signal delay of each gate is propagated to its dependent gates sequentially all the way to the primary outputs. The effect of wire delays is ignored for this experiment. The statistics of the delays at each primary output are recorded. If the delay is more than the clock period then an error is recorded. As an example, the delays recorded at the output 4 of the c432 circuit are presented in fig. 42. Clearly the output delay distribution has a long "never-ending" tail.

Clock speeds resulting in an error rate of less than 10^{-6} and a 10^{-4} (in any one input) are recorded, they correspond to clock periods 3.45σ and 2.7σ away from the mean signal delay. Based on these clock periods, an *industry standard* value of 9σ is

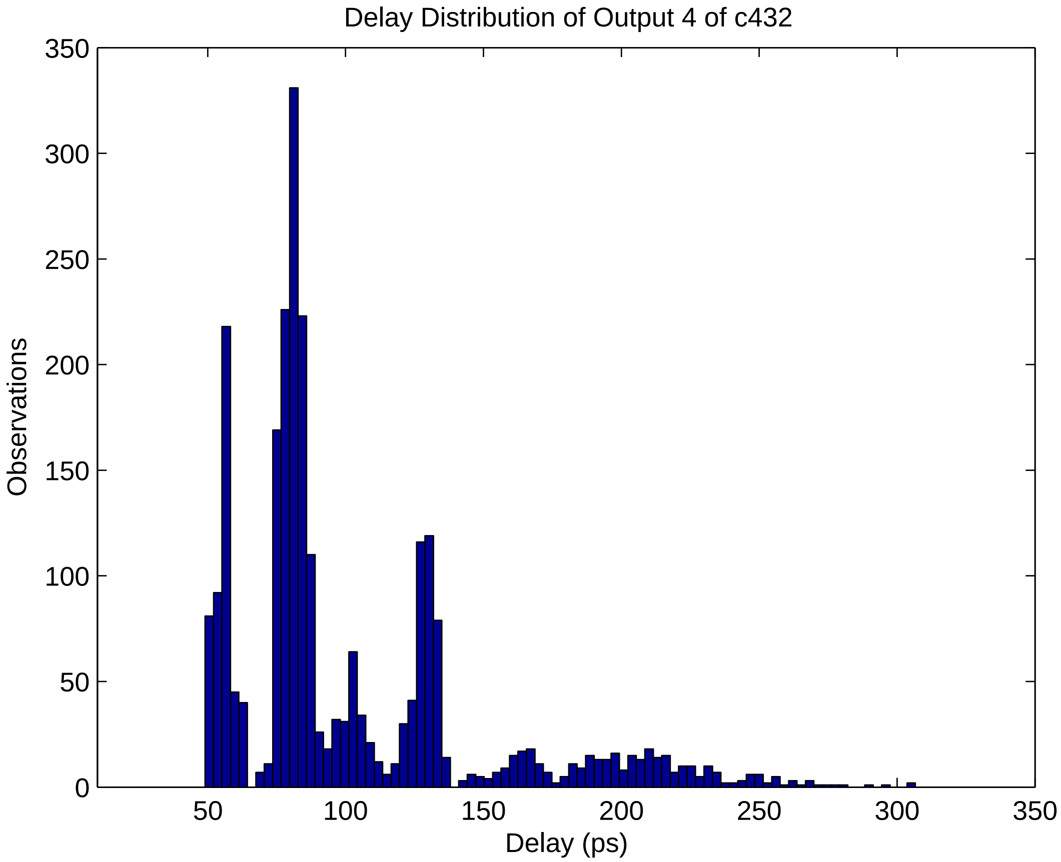

Fig. 42. Delay distribution of output 4 of c432 circuit.

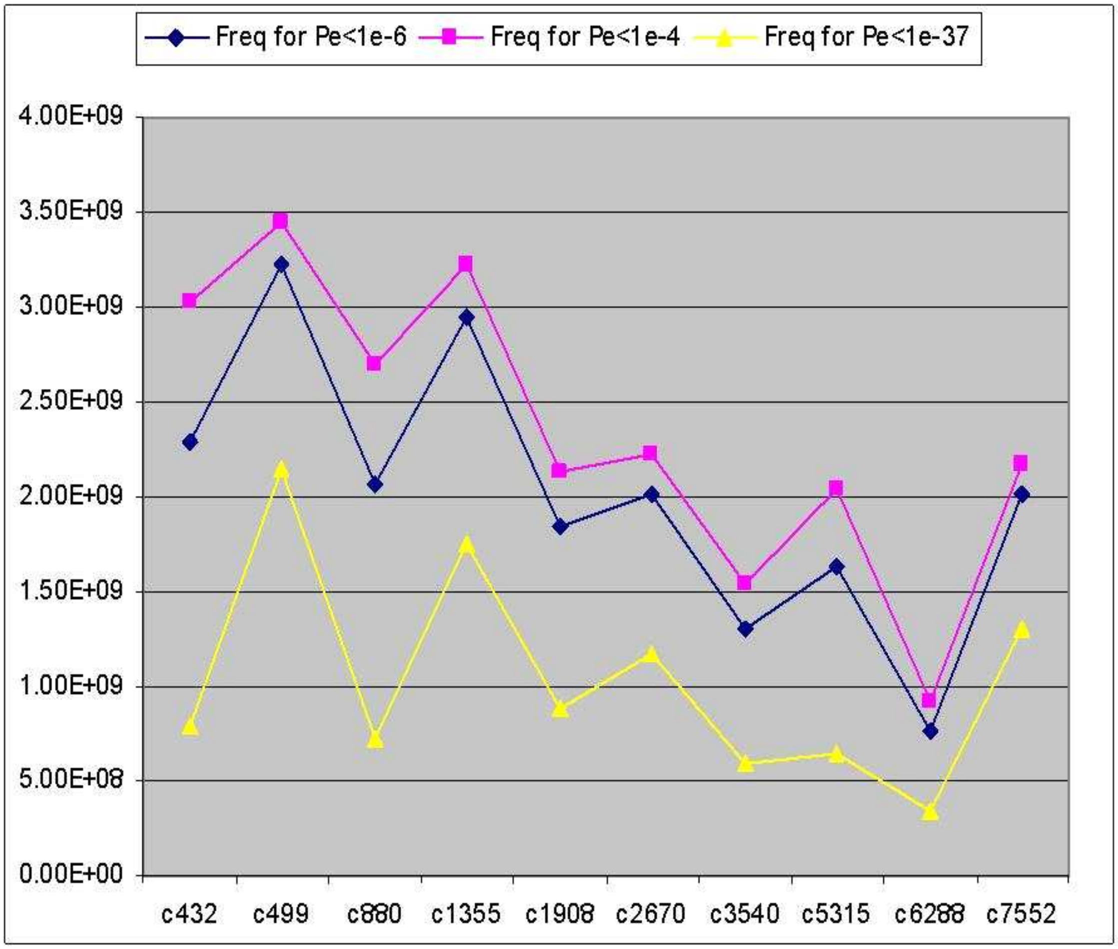

Fig. 43. The frequencies of operation of the ISCAS'85 circuits.

calculated. These values are shown in fig. 43.

In the architecture described in the last section, the parity generation checking circuit is given twice as much time as the original circuit to settle down to a value. Arguably, the delay error rate is so small that it can be concluded that the parity circuit will never have a delay error. But whenever the parity detection detects an error, a extra time is spent in

1. Flushing the pipeline and stalling,
2. Tuning to the new clock frequency, and
3. Retuning to the old clock frequency after a few cycles.

The number of cycles spent in flushing and stalling are equal to the pipeline depth. An estimate of the time required for tuning the clock is given in [66]. The speedup of the new architecture is given as

$$\text{Speedup} = \text{Clock Speedup} \times (1 + P_e N) \tag{6.2}$$

where, P_e is the new error rate and N is the cycles spent when an error occurs. Assuming a total time equivalent to 1000 regular clock cycles is spent in the process, the proposed architecture will show gain only for error rates smaller than 10^{-4} in the combinatorial circuit where the factor $P_e N$ becomes negligible compared to 1.

The Speedup (Gain) of all the ISCAS'85 benchmark circuits, when running at $P_e = 10^{-4}$ and 10^{-6} are listed in table X and shown in fig. 44. As shown in fig. 44, the overall speedups achieved by the new circuit are as high as 4 for some circuits.

As mentioned earlier, the experiments record the delay statistics of all the outputs in all the circuits and aims to find the output pairs with high delay error correlation. Table XI shows a list of output groups that have a high delay error correlation and

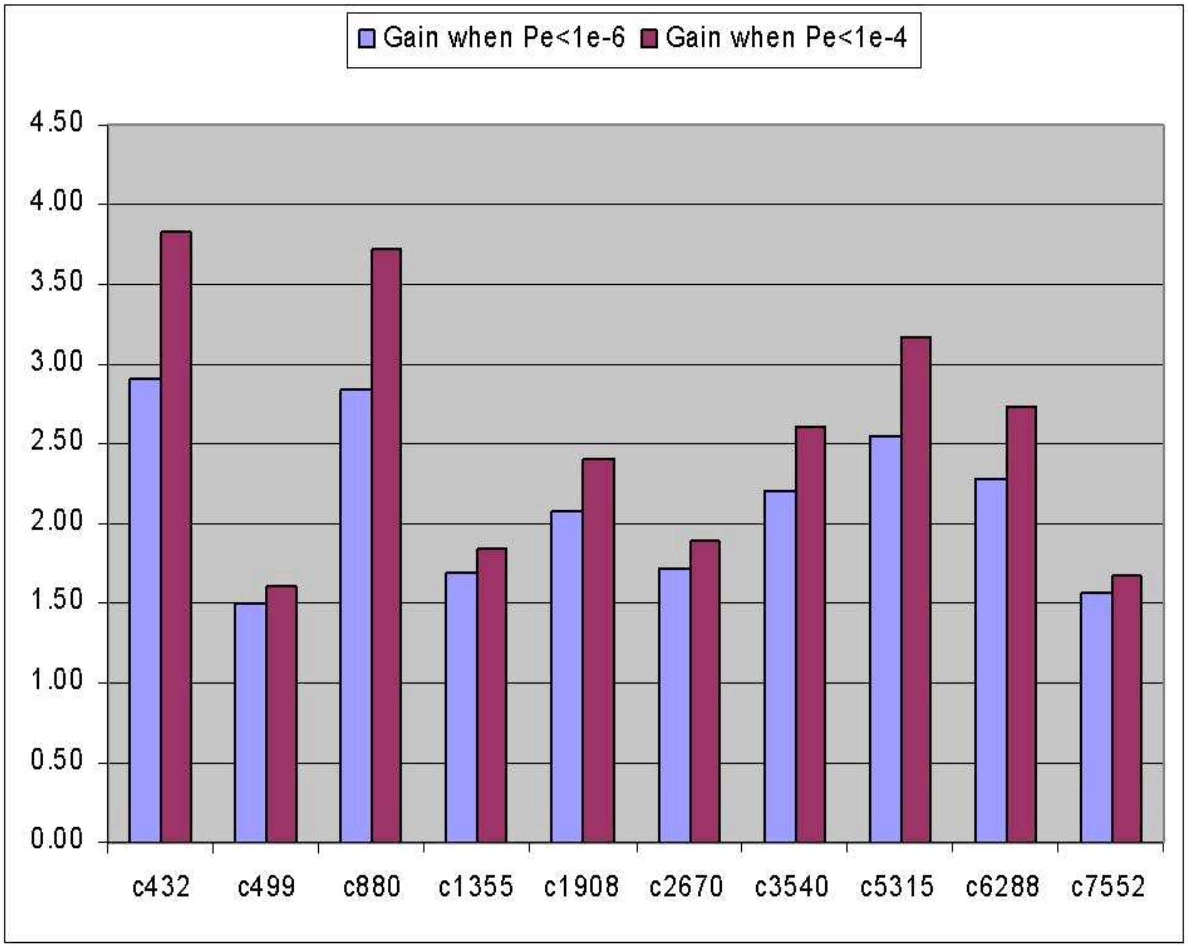

Fig. 44. The gain in frequency due to error correction coding in the ISCAS'85 circuits.

Table X. The Clock Frequencies for Different Error Probabilities.

Circuit	9σ Clock (GHz) for $P_e < 10^{-37}$	3.5σ Clock (GHz) for $P_e < 10^{-6}$	Gain
c432	0.79	2.29	2.90
c499	2.14	3.23	1.51
c880	0.72	2.07	2.84
c1355	1.74	2.94	1.69
c1908	0.88	1.84	2.07
c2670	1.17	2.02	1.72
c3540	0.59	1.30	2.20
c5315	0.64	1.64	2.55
c6288	0.33	0.76	2.28
c7552	1.30	2.02	1.56

Table XI. The Prohibited Output Groups Based on Delay Correlations.

Circuit	Outputs with High Delay Correlation
c432	(4,5,6,7)
c499	none
c880	(23,26)(24,25)
c1355	none
c1908	(17,23) (17,24)
c2670	none
c3540	(21,22)
c5315	(120,121,122,123)
c6288	(15,16,17,18,19,20,21,22,23,24,25,26)
c7552	none

should be avoided in the same parity group for a code construction. As an example in the circiut c880, the outputs 23 and 26 have a very high delay correlation. If output 23 has a delay violation, then there is a high probability that output 26 will also have a delay violation. If an error detection scheme is based on the XOR of these two outputs, then most errors will go undetected due to their simultaneous occurrences.

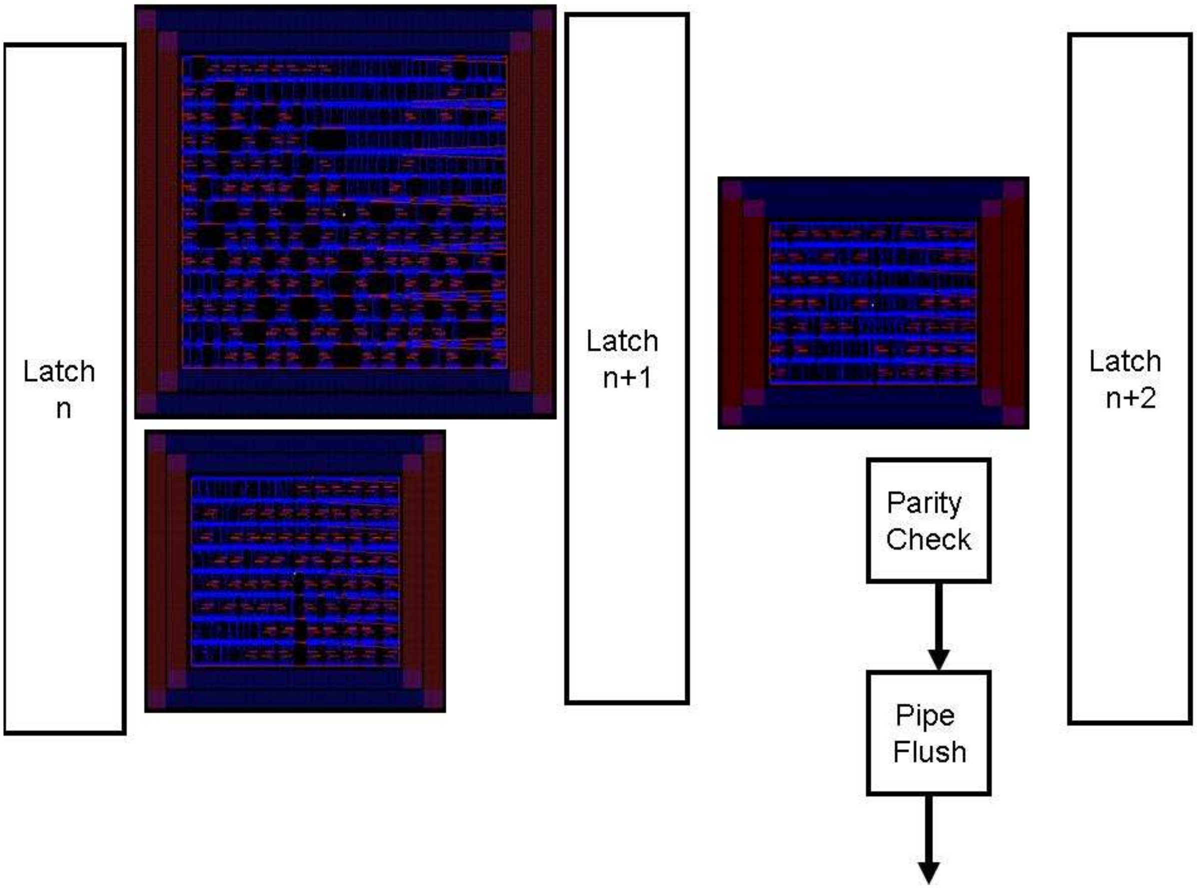

Fig. 45. A layout for error detection in c499.

A pseuso-layout for error detection for the circuit c499 based on the above architecture and the parity groups is given in fig. 45. There are a total of 16 parity groups, generated by pairing the 32 outputs randomly. The parity output logic is a binary XOR of the 2 primary outputs. The area of the parity generator 1 and 2 combined is roughly half that of the original circuit area.

C. Comparison Against Existing Methods

The coding scheme proposed herein has the following advantages over the other conventional coding schemes like Tri-Modulo Redundancy (TMR).

1. The Area overhead required for the proposed architecture is much less than TMR.
2. TMR does not consider the dependence of the delay on the data itself. The signal delay depends largely on the path that is sensitized depending on the data. In TMR if the delay error is due to the data, the error will show in all the 3 components and will go unnoticed into the data-path.
3. The power overhead in the proposed architecture is much less than TMR, which consumes exactly 3 times the power of the original circuit.

In comparison to the newer schemes like the *Razor* [67], the proposed scheme based on output delay correlation has the following advantages.

1. Unlike the Razor, the main circuit remains untouched, no extra delays or loads are added to any paths in the original circuit.
2. The proposed scheme covers all the outputs for delay errors compared to only about 7.4% outputs covered in Razor.
3. In an identical fashion to the razor, the proposed scheme achieves reduction in power due to voltage scaling. While Razor only recovers the errors on outputs as long as the signals arrive within 1.5 clock periods, there is no such constraint

in the proposed architecture. At a minimum 2 clock periods are allowed to recover the error. As a result, the potential power savings are a lot more.

4. While the Razor can achieve only power reduction, the proposed scheme can achieve both power reduction and speedup depending on the requirement of the application.

CHAPTER VII

CONCLUSION

This work has successfully demonstrated the following

- An estimate of the data handling limits of long on-chip interconnects and buses.
- A design methodology for coding/decoding systems for on-chip interconnects to improve reliability, reduce delays and energy consumption.
- A design methodology for coding systems for combinatorial circuits to improve reliability, increase speed (or) reduce power consumption.

Two distinct approaches are employed to calculate the capacity of interconnects. The first approach models a single interconnect as a linear time invariant (LTI) system. This is discussed more in Ch. II. The impulse response, step response and frequency response of this system are studied to determine the optimal data transfer frequency given appropriate pulse shapes for signal transmission. Ch. II models a 1mm long, 8-bit wide bus in a $0.1\mu m$ technology as a LTI system. The worst case delay (wire only) of such a setup is 57 ps resulting in a maximum clock frequency of 17 GHz (1/57 ps). The impulse response, step response and frequency response of this system are studied to determine the optimal data transfer frequency. From this analysis, it is found that the signal fidelity can be preserved even at speeds up to ≈ 30 GHz, for a 1 mm long, 8-bit wide bus in $0.1\mu m$ technology.

The second approach, discussed in Ch. III, makes use of the Shannon's capacity theorem like [34,35]. Like Ch. II, the data limits derived are for the wire component of the bus only. Also, the bus model parameters chosen are exacly the same, i.e., 1 mm long, 8-bit wide, $0.1\mu m$ technology.

Monte-Carlo experiments are performed on a bus with a wide variety of input combinations. Errors at the receiver are counted and the capacity is calculated. The results show that it is possible to achieve up to 28 Gbps, while transmitting 40 Giga *symbols* per second on every wire, and employing a rate 0.7 channel code. This is close to the ≈ 30 Gbps achieved with the LTI analysis. Recall that the maximum allowable bandwidth based on a design that accomodates the worst case delay is 17Gbps per wire.

Ch. IV has presented the 4B5W and the 4B11W codes that come close to the above limits even with significant gains in energy consumption and reliability. The names 4B5W and 4B11W mean that the 4 bit information is spread on-to 5 and 11 wires respectively to achieve reliability, speedup and energy savings. The codes achieve up-to 2.7x speedup in the throughput, while a 1.4x speedup in the information carrying rate of the individual wires. At the same time there are upto 14% energy savings over a no-code system. Even with these savings, more can be achieved by using these codes in conjunction with the existing techniques, i.e., buffer insertion, wire tapering, etc.

A more elaborate coding scheme, the Low Density Parity Check (LDPC) Code, is discussed in Ch. V. A decoder design is also presented that can be applied to on-chip interconnects. This decoder architectures provides complete programmability in addition to a fully-parallel architecture.

Ch. VI opens up the discussion on communication theoretic solutions to other aspects of chip design, i.e., combinatorial circuits to have upto 4x speedup in the clock frequencies. This work successfully demonstrates that the primary outputs can be partitioned into groups based on correlation and/or joint CDF. This approach based on delay characterization of each primary output can be exploited to construct error correcting schemes that have the potential to improve the timing yield of circuits.

The proposed scheme not only improves the reliability of combinatorial circuits but also results in increased speed (and/or) reduced power consumption.

A further exploration of how re-convergent fan-outs contribute to the correlation coefficients of the output bits is required. Intuitively, the further away the divergent fan-outs are from the primary outputs, the lesser correlation they induce. The understanding of this relationship between fan-outs and delay correlation is important to avoid extensive simulations for collecting delay statistics and to improve the synthesis.

A. Future Work

This research effort has introduced the use of coding theory to achieve speed-up, energy saving and reliability in different areas of VLSI design. The work on interconnect networks shows the huge potential for improvement both in terms of speed and energy consumed compared with the present day techniques. The codes introduced herein (4B5W and 4B11W) are based on an exhaustive search of the code-space. Even though this works for smaller bus widths, the complexity of search explodes for higher bus widths. A more formal mathematical technique is required to search for codes that are similar in nature to the 4B5W and the 4B11W code.

The research only briefly proposes the framework for building parity circuits for combinatorial circuits, by taking the example of the ISCAS 85 benchmark circuits. A more detailed exploration is required to form parity groups in more contemporary circuits with lesser variance in the path lengths. Once the parity groups are formed, the hypothesis needs to be proved on silicon.

REFERENCES

[1] H. Bakoglu, *Circuits, Interconnections, and Packaging for VLSI,* Reading, MA: Addison Wesley, 1990.

[2] M. Takahashi, M. Hashimoto, H. Onodera, “Crosstalk noise estimation for generic RC trees,” in *Proc. Int. Conf. Computer Design,* Sep. 2001, pp. 110-116.

[3] X. Qi, B. Kleveland, Y. Zhiping, S. Wong, R. Dutton, T. Young, “On-chip inductance modeling of VLSI interconnects,” in *Proc. Int. Conf. Solid-State Circuits*, Feb. 2000, pp. 172-173.

[4] S.H. Choi, K. Roy, “Noise analysis under capacitive and inductive coupling for high speed circuits,” in *Proc. Int. Workshop Electronic Design, Test and Applications*, Jan. 2002, pp. 365-369.

[5] Y. Tanji, H. Asai, “Closed-form expressions of distributed RLC interconnects for analysis of on-chip inductance effects,” in *Proc. Design Automation Conf.,* 2004, pp. 810 - 813.

[6] F.W. Grover, *Inductance Calculations, Working Formulas and Tables,* New York, NY: Dover, 1962.

[7] Y. Cao, P. Gupta, A.B. Kahng, D. Sylvester, J. Yang, “Design sensitivities to variability: extrapolations and assessments in nanometer VLSI,” in *Proc. Int. Conf. ASIC/SOC,* Sep. 2002, pp. 411-415.

[8] D. Liu, C. Svensson, “Power consumption estimation in CMOS VLSI chips,” *IEEE Journal of Solid-State Circuits,* vol. 29-6, pp. 663-670, June 1994.

[9] D. Liang, M.D.F. Wong, “Buffer insertion under process variations for delay minimization,” in *Proc. Int. Conf. CAD,* Nov. 2005, pp. 317 - 321.

[10] Z. Li, C.N. Sze, C.J. Alpert, J. Hu, W. Shi, “Making fast buffer insertion even faster via approximation techniques,” in *Proc. Asia South Pac. Design Automation Conf.,* Jan. 2005, vol. 1, pp. 13-18.

[11] Z. Li, W. Shi, “An $O(bn^2)$ time algorithm for optimal buffer insertion with b buffer types,” in *Proc. Design, Automation and Test in Europe,* Feb. 2005, vol. 2, pp. 1324-1329.

[12] L. Li, N. Vijaykrishnan, M. Kandemir, M.J. Irwin, “A crosstalk aware interconnect with variable cycle transmission,” in *Proc. Design, Automation and Test in Europe,* Feb. 2004, vol. 1, pp. 102-107.

[13] L. Zhang, Y. Hu, C.P. Chen, “Wave-pipelined on-chip global interconnect,” in *Proc. Asia South Pac. Design Automation Conf.,* Jan. 2005, vol. 1, pp. 127-132.

[14] N. R. Shanbhag, “Reliable and efficient system-on-chip design,” *IEEE Computer,* vol. 37-3, pp. 42-50, Mar. 2004.

[15] S.R. Sridhara, N.R. Shanbagh, “Coding for system on chip networks: A unified framework,” *IEEE Trans. on VLSI,* vol. 13-6, pp. 655-667, June 2005.

[16] S.R. Sridhara, A. Ahmed, N.R. Shanbhag, “Area and energy-efficient crosstalk avoidance codes for on-chip buses,” in *Proc. Int. Conf. Comp. Design* Oct. 2004, pp. 12-17.

[17] P. P. Sotiriadis and A. Chandrakasan, “Reducing bus delay in submicron technology using coding,” in *Proc. Asia South Pac. Design Automation Conf.,* Jan. 2001, pp. 109-114.

[18] P. P. Sotiriadis, A. Chandrakasan, “A bus energy model for deep submicron technology,” *IEEE Trans. on VLSI,* vol. 10-3, pp. 341-350, June 2002.

[19] C. Duan, A. Tirumala, S. P. Khatri, “Analysis and avoidance of cross-talk in on-chip buses,” in *Proc. Hot Interconnects* Aug. 2001, pp. 133 - 138.

[20] C. Duan, S.P. Khatri, “Exploiting crosstalk to speed up on-chip buses,” in *Proc. Design, Automation and Test in Europe,* Feb. 2004, vol. 2, pp. 778-783.

[21] K.H. Baek, Ki-Wook Kim, Sung-Mo Kang, “A low energy encoding technique for reduction of coupling effects in SoC interconnects,” in *Proc. Midwest Symp. on Circuits and Systems,* Aug. 2000, vol. 1, pp. 80-83.

[22] M. Ghoneima, Y. Ismail, “Low power coupling-based encoding for on-chip buses,” in *Proc. Int. Symp. Circuits and Systems*, May 2004, vol. 2, pp. 325-328.

[23] D. Bertozzi, L. Benini, B. Ricco, “Energy-efficient and reliable low-swing signaling for on-chip buses based on redundant coding,” in *Proc. Int. Symp. Circuits and Systems,* May 2002, vol. 1, pp. 93-96.

[24] M.R. Stan, W.P. Burleson, “Bus-invert coding for low power I/O,” *IEEE Trans. on VLSI*, vol. 3-1, pp. 49-58, Mar. 1995.

[25] B.J. LaMeres, S.P. Khatri, “Encoding-based minimization of inductive crosstalk for off-chip data transmission,” in *Proc. Design, Automation and Test in Europe,* Feb. 2005, pp. 1318-1323.

[26] V. Raghunathan, M. B. Srivastava and R. K. Gupta, “A survey of techniques for energy efficient on-chip communication,” in *Proc. Design Automation Conf.,* June 2003, pp. 900-905.

[27] R. L. Pickholtz, *Digital Systems Testing and Testable Design,* New York, NY: Computer Science Press, 1990.

[28] M. Favalli, and C. Metra, “Optimization of error detecting codes for the detection of crosstalk originated errors,” in *Proc. Design, Automation and Test in Europe,* Mar. 2001, pp. 290-296.

[29] D. Bertozzi, L. Benini, G. De Micheli, “Low power error resilient encoding for on-chip data buses,” in *Proc. Design, Automation and Test in Europe,* Mar. 2002, pp. 102-109.

[30] S. Lin and D. J. Costello, *Error Control Coding,* Englewood Cliffs, NJ: Prentice-Hall, 2005.

[31] F. Worm, P. Ienne, P. Thiran and G. De Micheli, “An adaptive low-power transmission scheme for on-chip networks,” in *Proc. Int. Symp. System Synthesis,* Oct. 2002, pp. 92-100.

[32] T. Dumitras, S. Kerner and R. Marculescu, “Towards on-chip fault-tolerant communication,” in *Proc. Asia South Pac. Design Automation Conf.,* Jan. 2003, pp. 225-232.

[33] A. Leon-Garcia and I. Widjaja, *Communication Networks,* New York, NY: McGraw-Hill, 2000.

[34] R. Singhal, G.S. Choi, R.N. Mahapatra, “Information theoretic capacity of long on-chip interconnects in the presence of crosstalk,” in *Proc. Int. Symp. Quality Electronic Design* Mar. 2006, pp. 407-412.

[35] R. Singhal, G.S. Choi, R.N. Mahapatra, “Information theoretic approach to address delay and reliability in long on-chip interconnects,” in *Proc. Int. Conf.*

CAD, Nov. 2006, pp. 310-314.

[36] R. Gallager, *Information Theory and Reliable Communication,* New York, NY: Wiley, 1968.

[37] M. Mushkin, I. Bar-David, "Capacity and coding for the Gilbert-Elliot channels," *IEEE Trans. on Information Theory,* vol. 35-6, pp. 1277-1290, Nov. 1989.

[38] K. Lahiri, A. Raghunathan, "Power analysis of system-level on-chip communication architectures," in *Proc. Int. Conf. Hardware/Software Codesign and System Synthesis* 2004, pp. 236-241.

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

[40] T. J. Richarson, M. A. Shokrollahi, R. L. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Trans. on Information Theory,* vol. 47-2, pp. 619-637, Feb. 2001.

[41] P. Bhojwani, R. Singhal, G. Choi, R. Mahapatra, "Forward error correction for on-chip interconnection networks," in *Proc. Workshop for Unique Chips and Systems* Mar. 2006, pp. 41-47.

[42] E. Yeo, B. Nikolic, V. Anantharam, "Architectures and implementations of low-density parity check decoding algorithms," in *Proc. Midwest symp. Circuits and Systems,* Aug. 2002, vol. 3, pp. 437-440.

[43] A.J. Blanksby, C.J. Howland, "A 690-mW 1-Gb/s 1024-b, rate-1/2 low-density parity-check code decoder," *IEEE Journal of Solid State Circuits,* vol. 37-3, pp. 404-412, Mar. 2002.

[44] C.J. Howland, A.J.Blanksby, “Parallel Decoding Architectures for Low Density Parity Check Codes,” in *Proc. Int. Symp. Circuits and Systems,* May 2001, vol. 4, pp. 742-745.

[45] C. Berrou, A. Glavieux, and P. Thitimajshima, “Near Shannon limit error-correcting codes and decoding,” in *Proc. Int. Conf. Communications,* May 1993, vol. 2, pp. 1064-1070.

[46] R. Singhal, G.S. Choi, N. Mickler, P. Koteeswaran, “Scaleable check node centric architecture for LDPC decoder,” in *Proc. Int. Symp. on Circuits and Systems,* May 2004, vol. 4, pp. 189-192.

[47] R. Singhal, G.S. Choi, R.N. Mahapatra, “Quantized LDPC decoder design for binary symmetric channels,” in *Proc. Int. Symp. Circuits and Systems,* May 2005, vol. 6, pp. 5782-5785.

[48] R. Singhal, G.S. Choi, R.N. Mahapatra, “Programmable LDPC decoder based on the bubble sort algorithm,” in *Proc. Int. Conf. VLSI,* Jan. 2006.

[49] M. Karkooti, J.R. Cavallaro, “Semi-parallel reconfigurable architectures for real-time LDPC decoding,” in *Proc. Int. Conf. on Information Technology: Coding and Computing,* Apr. 2004, vol. 1, pp. 579-585.

[50] S. Sivakumar, “VLSI implementation of encoder and decoder for low density parity check codes,” M.S. Thesis, Texas A&M University, Dec. 2001.

[51] A. Selvarathiam, G. Choi, K. Narayanan, A. Prabakhar, E. Kim, “A massively scaleable decoder architecture for low-density parity-check codes,” in *Proc. Int. Symp. Circuits and Systems,* May 2003, vol. 3, pp. 93-96.

[52] T. Zhang, K.K. Parhi, “Joint code and decoder design for implementation-oriented (3, k)-regular LDPC codes,” in *Proc. Asilomar Conf. Signals, Systems and Computers,* Nov. 2001, vol. 2, pp. 1232-1236.

[53] T. Zhang, Z. Wang and K. K. Parhi, “On finite precision implementation of low-density parity-check codes decoder,” in *Proc. Int. Symp. Circuits and Systems,* May 2001, vol. 4, pp. 202-205.

[54] G. Al-Rawi, J. Cioffi, M. Horowitz, “Optimizing the mapping of low-density parity check codes on parallel decoding architectures,” in *Proc. Int. Conf. Information Technology, Coding and Computing,* Apr. 2001, pp. 578-586.

[55] M. M. Mansour and N. R. Shanbag, “Low-Power VLSI Decoder Architectures for LDPC Codes,” in *Proc. Int. Symp. Low Power Electronics and Design,* 2002, pp. 284-289.

[56] Flarion Technology, Vector-LDPC Coding Solution Data Sheet. [On-line] Available: http://www.flarion.com/products/vector.asp. Accessed Apr. 2004.

[57] A. Saldanha, R.K. Brayton, A.L. Sangiovanni-Vincentelli, “Circuit structure relations to redundancy and delay,” *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems,* vol. 13-7, pp. 875-883, July 1994.

[58] I.D. Elliott, I.L. Sayers, “Implementation of 32-bit RISC processor incorporating hardware concurrent error detection and correction,” in *Proc. Computers and Digital Techniques*, Jan. 1990, vol. 137-1, pp. 88-102.

[59] S.H. Li, C.A. Zukowski, “A self-timed cyclic redundancy check (CRC) in VLSI,” in *Proc. Midwest Symp. Circuits and Systems*, Aug. 1997, vol. 2, pp. 1021-1025.

[60] S. Subramanian, P.K. Lala, "On self-checking design of CMOS circuits," in *Proc. European Conf. Design Automation,* Feb. 1993, pp. 139-143.

[61] P. Kukkal, J. Bowles, "Reliability in CMOS VLSI circuits," in *Proc. Southeastern Symp. System Theory*, Mar. 1991, pp. 246-253.

[62] B. Kapoor, V.S.S. Nair, "Area, performance, and sensitizable paths," in *Proc. Great Lakes Symp. on VLSI,* Mar. 1994, pp. 222-227.

[63] S. Dhawan, R.C. De Vries, "Design of self-checking iterative networks," *IEEE Trans. on Computers,* vol. 37-9, pp. 1121-1125, Sept. 1988.

[64] Z Wang; M.M. Marek-Sadowska, K.H. Tsai, J. Rajski, "Delay-fault diagnosis using timing information," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems,* vol. 24-9, pp. 1315-1325, Sep. 2005.

[65] Layout and Parasitic Information for ISCAS Circuits. [On-line] Available: http://dropzone.tamu.edu/ xiang/iscas.html. Accessed Aug. 2005.

[66] M. Olivieri, A. Trifiletti, and A. De Gloria, "A low-power microcontroller with on-chip self-tuning digital clock generator or variable-load applications," in *Proc. Int. Conf. Computer Design,* Oct. 1999, pp. 476-481.

[67] S. Das, D. Roberts, S. Lee, S. Pant, D. Blaauw, *et al* "A self-tuning DVS processor using delay-error detection and correction," *IEEE Journal of Solid-State Circuits*, vol. 41-4, pp. 792-804, 2006.

VITA

Rohit Singhal has worked as an Assistant Research Engineer, in Electrical Engineering Technolgy at Texas A&M University since Nov. 2006. Previously he worked as a Lecturer in Electrical Engineering Technology at Texas A&M University teaching senior and junior level courses.

He received his PhD degree in Computer Science at Texas A&M University in May 2007. Rohit completed his M.S. in Electrical Engineering at Texas A&M University in 2003 and received a B.Tech. (Honors) in Electronics and Electrical Communication Engineering from the Indian Institute of Technology Kharagpur, India in May 2000.

Rohit can be reached at The Department of Engineering Technology and Industrial Distribution at Texas A&M University, Mail Stop 3367, College Station, Texas 77843.

The typist for this thesis was Rohit Singhal.