STATISTICAL ERROR COMPENSATION FOR ROBUST DIGITAL SIGNAL PROCESSING
AND MACHINE LEARNING

BY
ERIC PARK KIM

DISSERTATION
Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2014

Urbana, Illinois

Doctoral Committee:
Professor Naresh R. Shanbhag, Chair
Professor Rob A. Rutenbar
Professor Andrew C. Singer
Associate Professor Deming Chen
Abstract

Machine learning (ML) based inference has recently gained importance as a key kernel in processing massive data in digital signal processing (DSP) systems. Due to the ever increasing complexity of DSP systems, energy-efficient ML accelerators are critical. Traditionally, energy efficiency was obtained through technology scaling. However, modern nanoscale complementary metal–oxide semiconductor (CMOS) process technologies suffer in reliability caused by process, temperature, and voltage variations. As ML applications are inherently probabilistic and robust to errors, statistical error compensation (SEC) techniques can play a significant role in achieving robust and energy-efficient implementation of these important kernels. SEC embraces the statistical nature of errors and utilizes statistical and probabilistic techniques to build robust systems. Energy efficiency is obtained by trading off the enhanced robustness with energy. This dissertation focuses on utilizing statistical approaches via SEC in implementing energy-efficient digital signal processing (DSP) systems with an emphasis on machine learning kernels.

We first demonstrate the potential of SEC techniques to a detection based application. A 180 nm CMOS pseudonoise (PN) code acquisition integrated circuit (IC) has been implemented and measured. Measurements show that while maintaining a detection probability $P_{\text{det}} \geq 90\%$, an error rate $p_{\eta} \geq 85.83\%$ with energy savings of $2.52 \times$ could be achieved.

SEC is then applied to a communication centric machine learning kernel, a low-density parity check (LDPC) decoder. As iterative message-passing based architectures are inherently robust to small-magnitude errors, the SEC based LDPC decoder shows significant improvement in robustness and energy efficiency. Three different size LDPC codes, (50, 25), (800, 400), and (1800, 900), were implemented with five iterations per block. Circuit simulations in a commercial 45 nm process show that the SEC based LDPC decoder can operate...
at a supply voltage up to 38% less than the nominal voltage and tolerate up to $30 \times$ more errors over an SNR range of 3 dB to 8 dB, while maintaining less than $3 \times$ degradation in bit error rate (BER). This is equivalent to energy savings of 45.7% compared to conventional LDPC decoders, and 33.2% compared to a sign bit protected LDPC decoder.

Motivated by the success of SEC based LDPC decoders, SEC has been applied to a more complex message-passing application: Markov random field (MRF) based stereo image matching. Analysis and simulations show that for a 20-bit architecture, small errors ($\eta \leq 1024$) are tolerable, while large errors ($\eta \geq 4096$) degrade the performance significantly. By applying algorithmic noise tolerance (ANT), experimental results show that the proposed ANT based hardware can tolerate an error rate of 20%, with performance degradation of only 3.5% at an overhead of 97.4%, compared to an error-free full precision hardware with an energy savings of 39.6%. To reduce the compensation complexity, higher level error compensation is explored as well.

Recent studies on approximate computing (AC) follow a principle similar to SEC, but with one critical exception. AC based design still carries the requirement of creating a deterministic design, and thus the improvement in energy efficiency is marginal. We successfully apply SEC to AC based designs and show that by embracing the statistical nature of the underlying process, an additional 44.9% energy savings can be obtained.

Finally, SEC techniques are analyzed to provide insight into the trade-offs in the design of SEC based systems. Algorithmic noise tolerance is analyzed under a unifying framework based on detection and estimation theory. ANT is shown to approximate the Bayes optimal detector and estimator.
To my family, friends, and the curious minds
Acknowledgments

My deepest thanks go to my wife, my children, whom I have not yet had the pleasure to meet, my parents, and my adviser. My wife has always given me motivation and strength to continue my pursuit of this doctorate degree. She herself has put up a goal far challenging than a doctorate degree, and showed to me, by example, how to live everyday with meaning. My parents have always put me and my sister before them, and cared for us with the greatest love any parent can give. Their constant support and advice has inspired me and helped me in numerous situations, and their attitude towards life has set an example that I have always tried to follow. My adviser, Professor Naresh Shanbhag, has been the greatest teacher and mentor one could hope for. He has given me great challenges and showed me what it means to accomplish them. His patience during this journey along with his faith in my abilities is the one reason this dissertation was able to be completed. I am greatly indebted to him for all the advice, knowledge, responsibilities, and opportunities that he gave me in school and life. His passion in life and work, his knowledge and experience, along with his high standards will be a source of inspiration and an example to follow during my entire career. I would also like to thank Professors Rob Rutenbar, Andrew Singer, and Deming Chen for the many insightful discussions and input and for agreeing to be on my committee. Their comments and suggestions have greatly helped improve this dissertation. Additionally, I would like to thank Jungwook Choi for his help on implementing the TRWS HW architecture and running various simulations for the results in Chapter 4 and my research group colleagues for their feedback and input during group meetings. I greatly acknowledge past and present support from the Gigascale Systems Research Center (GSRC), one of six research centers funded under the Focus Center Research Program (FCRP), a Semiconductor Research Corporation (SRC) entity; and Systems on Nanoscale Information fabriCs (SONIC), one of the six SRC STARnet Centers, sponsored by MARCO and DARPA. Finally, I would like to thank my friends at the Korean Tennis Club (KTC), especially Mr. Chulkee Chang, for making me feel at home, and taking me in as a family member during the long and lonely holidays. Because of the KTC, I was able to relieve my stress, and start fresh on my research.
# TABLE OF CONTENTS

List of Tables .................................................................................................................. viii
List of Figures .................................................................................................................... ix
List of Abbreviations ......................................................................................................... xiii

Chapter 1 Introduction ..................................................................................................... 1
  1.1 Research Direction .................................................................................................. 6
  1.2 Robust System Design ............................................................................................. 7
  1.3 Statistical Error Compensation ............................................................................. 17
  1.4 Contributions of This Dissertation ....................................................................... 23
  1.5 Dissertation Organization ..................................................................................... 24

Chapter 2 The Design of an Error-Resilient PN Code Acquisition Prototype Chip .... 26
  2.1 PN Code Acquisition Filter .................................................................................... 27
  2.2 SEC Based PN Code Acquisition Filter ................................................................ 29
  2.3 Chip Architecture .................................................................................................. 31
  2.4 Measurement Results ............................................................................................. 40
  2.5 Summary ................................................................................................................ 47

Chapter 3 Statistical Error Compensation for Low-Density Parity Check Codes ........ 49
  3.1 Low-Density Parity Check Codes .......................................................................... 51
  3.2 LDPC Decoder Architecture .................................................................................. 55
  3.3 Energy, Delay, and Error Modeling ...................................................................... 56
  3.4 Simulation Results .................................................................................................. 59
  3.5 Summary ................................................................................................................ 62

Chapter 4 Statistical Error Compensation for Stereo Image Matching .................... 64
  4.1 TRW-S Message-Passing Based Stereo Matching .................................................. 66
  4.2 Error-Resilient TRW-S via ANT .......................................................................... 67
  4.3 Error Compensation at Various Levels .................................................................. 74
  4.4 System Evaluation Setup ........................................................................................ 77
  4.5 Results ...................................................................................................................... 82
  4.6 Summary .................................................................................................................. 91
<table>
<thead>
<tr>
<th>Chapter 5</th>
<th>Approximate Computing Based Statistical Error Compensation</th>
<th>92</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>Approximate Computing</td>
<td>93</td>
</tr>
<tr>
<td>5.2</td>
<td>TRW-S Accelerator Architecture Using AC and SEC</td>
<td>96</td>
</tr>
<tr>
<td>5.3</td>
<td>Simulation and Results</td>
<td>99</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 6</th>
<th>Analysis of Statistical Error Compensation</th>
<th>106</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1</td>
<td>Analysis of Algorithmic Noise Tolerance</td>
<td>106</td>
</tr>
<tr>
<td>6.2</td>
<td>Comparison of Simulation and Analysis</td>
<td>111</td>
</tr>
<tr>
<td>6.3</td>
<td>Summary</td>
<td>114</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 7</th>
<th>Conclusion and Future Work</th>
<th>116</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.1</td>
<td>Dissertation Contributions</td>
<td>116</td>
</tr>
<tr>
<td>7.2</td>
<td>Future Work</td>
<td>117</td>
</tr>
</tbody>
</table>

| References |                                                            | 121 |
List of Tables

1.1 Various robust system design techniques applied at different levels. . . . . . . 10
1.2 Characteristic of robust system design techniques. . . . . . . . . . . . . . . . 12

2.1 Complexity of circuit blocks in SSNOC prototype IC. . . . . . . . . . . . . . 39
2.2 Comparison of SSNOC prototype IC with other work. . . . . . . . . . . . . . 45

3.1 HW error rate that can be tolerated by SEC based LDPC decoder at a
given BER threshold. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 61

4.1 Depth map and BPR comparison for error-free, conventional, and ANT at
various $V_{dd}$ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 85
4.2 Estimated compensation overhead and power consumption of arithmetic-
level ANT obtained via synthesis in a commercial 45 nm CMOS process. . . 86
4.3 Estimated compensation overhead and power consumption of iteration-
level ANT obtained via synthesis in a commercial 45 nm CMOS process. . . 90

5.1 Truth table for the mirror based approximate adder 1 through 4 [21]. . . . . 96
5.2 Estimated cell count and power consumption obtained via synthesis in a
45 nm CMOS process with AC using approximate adder 1 [21] . . . . . . . . 103
5.3 Estimated cell count and power consumption obtained via synthesis in a
45 nm CMOS process with AC using approximate adder 2 [21] . . . . . . . . 103
5.4 Estimated cell count and power consumption obtained via synthesis in a
45 nm CMOS process with AC using approximate adder 3 [21] . . . . . . . . 104
5.5 Estimated cell count and power consumption obtained via synthesis in a
45 nm CMOS process with AC using approximate adder 4 [21] . . . . . . . . 104
### List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Machine learning applications: (a) computer vision (stereo image matching and background segmentation), and (b) machine listening (sound source separation).</td>
<td>2</td>
</tr>
<tr>
<td>1.2</td>
<td>Ever increasing VLSI power [3]: (a) increase in leakage power due to technology scaling, and (b) expected power per chip. The increase in leakage power may create a large discrepancy from the ITRS requirements.</td>
<td>3</td>
</tr>
<tr>
<td>1.3</td>
<td>Energy challenge due to thermal limits [5]: (a) leakage power of nominal dice over temperature, and (b) silicon junction temperature of different representative cellular baseband processors under heaviest usage scenario.</td>
<td>4</td>
</tr>
<tr>
<td>1.4</td>
<td>Variation in deep scaled CMOS devices [6]: (a) threshold voltage ($V_{th}$) at different process technology, and (b) delay of memory cells, represented by color, of a 32 nm 64 Kbit SRAM cell obtained via simulations.</td>
<td>5</td>
</tr>
<tr>
<td>1.5</td>
<td>$I_{DS}$ vs. $V_{DS}$ curve for 50 different CNFETs [7].</td>
<td>5</td>
</tr>
<tr>
<td>1.6</td>
<td>Simulation results of voltage overscaling (VOS) for a 4-tap correlation filter (a sensor) at 50 MHz in a 180 nm CMOS process.</td>
<td>7</td>
</tr>
<tr>
<td>1.7</td>
<td>Error model used within this dissertation. The actual output $y_i$ consists of the ideal output $y_o$, hardware errors $\eta$, and estimation errors $e$.</td>
<td>12</td>
</tr>
<tr>
<td>1.8</td>
<td>An NMR system. The processing element (PE) is replicated $N$ times, and a majority voter is used to combine the outputs.</td>
<td>13</td>
</tr>
<tr>
<td>1.9</td>
<td>An NMR system with a triplicated voter.</td>
<td>14</td>
</tr>
<tr>
<td>1.10</td>
<td>Statistical error compensation: (a) general form, (b) algorithmic noise tolerance, and (c) error distributions.</td>
<td>18</td>
</tr>
<tr>
<td>1.11</td>
<td>Block diagram of a stochastic sensor network-on-a-chip (SSNOC). By decomposing the main computation into smaller blocks or “sensors,” the stochastic sensor network-on-chip creates an opportunity for efficient system-level error-tolerance techniques.</td>
<td>20</td>
</tr>
<tr>
<td>1.12</td>
<td>Block diagram of (a) NMR, and (b) soft NMR.</td>
<td>22</td>
</tr>
<tr>
<td>2.1</td>
<td>Architecture of a linear feedback shift register (LFSR) used for PN code generation.</td>
<td>28</td>
</tr>
<tr>
<td>2.2</td>
<td>PN code acquisition filter: (a) conventional, and (b) SEC based.</td>
<td>29</td>
</tr>
<tr>
<td>2.3</td>
<td>Measured sensor outputs over time. Outliers in the sensor output are due to hardware errors $\eta_i$ and can be seen to shift the mean significantly.</td>
<td>31</td>
</tr>
<tr>
<td>2.4</td>
<td>Prototype IC architecture.</td>
<td>32</td>
</tr>
</tbody>
</table>
2.5 Sensor architecture. .......................................................... 33
2.6 Distribution of outputs selected by hierarchical median algorithms, selecting the lower of the two central values at each stage. ...................... 35
2.7 Simulated ROCs for hierarchical median: (a) simple two-level, (b) simple three-level, (c) overlapping two-level and (d) overlapping three-level. ...... 36
2.8 Bit slice of the median block. .............................................. 37
2.9 Fusion block architecture. .................................................. 37
2.10 Thresholding block architecture. ......................................... 38
2.11 The 256-tap PN code acquisition filter chip in a 180 nm CMOS process: (a) layout, and (b) microphotograph. ....................................... 39
2.12 Error PMF of region R1: (a) $V_{dd} = 0.88$ V, and (b) $V_{dd} = 0.85$ V. All errors are one bit errors with powers-of-two magnitude. ........... 41
2.13 Error PMF of region R2: (a) $V_{dd} = 0.78$ V, and (b) $V_{dd} = 0.76$ V. Multi-bit errors are observed, but most errors are still small in magnitude. .... 41
2.14 Error PMF of region R3: (a) $V_{dd} = 0.68$ V, and (b) $V_{dd} = 0.66$ V. A dense Gaussian PMF and a sparse Gaussian PMF are mixed. ............ 42
2.15 Error PMF of region R4: (a) $V_{dd} = 0.63$ V, and (b) $V_{dd} = 0.60$ V. ....... 43
2.16 Detection probability $P_{det}$ and sensor probability of error $p_\eta$ vs. supply voltage $V_{dd}$. ....................................................... 44
2.17 Energy consumption and sensor probability of error $p_\eta$ vs. supply voltage $V_{dd}$. .............................................................. 45
2.18 HSPICE simulation of a sensor to measure $\alpha_{0\rightarrow1}$: (a) number of glitches vs. $V_{dd}$, and (b) average $\alpha_{0\rightarrow1}$ vs. $V_{dd}$. ....................... 47

3.1 Supply voltage vs. energy and delay of an LDPC variable node (Fig. 3.4(a)) in a commercial 45 nm process. By voltage overscaling up to an error rate of 70%, the same performance can be achieved with 70% less energy. 51
3.2 Example LDPC code: (a) parity check matrix, and (b) its bipartite graph. 52
3.3 High-level block diagram of the LDPC decoder. .......................... 55
3.4 Architecture of nodes: (a) variable node, and (b) check node. ......... 56
3.5 Energy consumption and delay curves obtained through circuit simulation of the variable and check node architectures in Fig. 3.4, synthesized in a commercial 45 nm process. ........................................ 57
3.6 Interconnect energy for a 200 $\mu$m wire: (a) distributed RC model, and (b) average energy vs. supply voltage curve. ............................... 58
3.7 Methodology for simulating the LDPC decoder architecture under VOS. Energy models can be obtained as well. ............................... 59
3.8 BER vs. SNR plot of a (800, 400) LDPC code decoded with five iterations at $p_\eta = 0.2$. ....................................................... 60
3.9 BER vs. $p_\eta$ at $SNR = 5$ dB. ................................................ 61
3.10 Energy vs. BER plot of a (800, 400) LDPC code decoded with five iterations at $SNR = 5$ dB. ................................................... 62

4.1 Architecture of streaming TRW-S hardware (TRWS_HW): (a) block diagram, (b) reparameterize unit, and (c) message update unit. .................. 68
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.2</td>
<td>Effect of AS error on updated message when (a) $\Delta H &gt; 0$, and (b) $\Delta H &lt; 0$.</td>
<td>70</td>
</tr>
<tr>
<td>4.3</td>
<td>Effect of error after normalization: (a) large effective error when error values differ significantly, and (b) small effective error when error values are similar.</td>
<td>71</td>
</tr>
<tr>
<td>4.4</td>
<td>Verification of error analysis: (a) effect of error on message updated via a box plot of $\eta_{AS}$ vs. $\tilde{\eta}_{AS}$, and (b) effect of error on energy minimization performance of message passing via a plot of energy $E$ vs. $\tilde{\eta}$.</td>
<td>73</td>
</tr>
<tr>
<td>4.5</td>
<td>Performance of ANT with injection of uniform errors of different magnitude using 4-bit precision estimator.</td>
<td>74</td>
</tr>
<tr>
<td>4.6</td>
<td>Block diagrams: (a) error compensation at different levels, and (b) detailed comparison of arithmetic-level and iteration-level compensation in the reparameterize unit.</td>
<td>75</td>
</tr>
<tr>
<td>4.7</td>
<td>Simulation methodology: (a) flow diagram, and (b) error statistics for AS at $V_{dd} = 0.75$ V with $p_\eta = 0.21$.</td>
<td>79</td>
</tr>
<tr>
<td>4.8</td>
<td>Energy minimization performance against various precisions in computation of TRWS_HW.</td>
<td>80</td>
</tr>
<tr>
<td>4.9</td>
<td>Block diagram depicting the truncation and saturation for optimal fixed-point implementation of REPARM and MSG_UPD.</td>
<td>80</td>
</tr>
<tr>
<td>4.10</td>
<td>Architecture of streaming TRW-S stereo matching CPU+FPGA system.</td>
<td>81</td>
</tr>
<tr>
<td>4.11</td>
<td>Timing errors in FPGA: (a) block diagram for error verification and statistics collection, (b) measured error statistics (20-bit) in the FPGA, and (c) error statistics (8-bit) obtained via circuit simulations in a 45 nm CMOS process [91].</td>
<td>83</td>
</tr>
<tr>
<td>4.12</td>
<td>Simulation results of error injection rate vs. energy minimization for different ANT estimators with $M[8,8,4]$.</td>
<td>84</td>
</tr>
<tr>
<td>4.13</td>
<td>Correction performance at (a) system level with SAD, (b) system level with HBP ($S = 8$), and (c) hybrid with HBP ($S = 8$) and HWRPR(3-bit).</td>
<td>88</td>
</tr>
<tr>
<td>4.14</td>
<td>Bad-pixel ratio vs. $V_{dd}$. With no error compensation, TRW-S alone cannot tolerate 10% voltage scaling.</td>
<td>89</td>
</tr>
<tr>
<td>4.15</td>
<td>Results for RPR-ANT applied at the arithmetic level: (a) FPGA emulation, and (b) error injection based simulation.</td>
<td>89</td>
</tr>
<tr>
<td>5.1</td>
<td>Approximate mirror adders [21]: (a) conventional mirror adder, (b) approximate mirror adder 1, (c) approximate mirror adder 2, (d) approximate mirror adder 3, and (e) approximate mirror adder 4.</td>
<td>97</td>
</tr>
<tr>
<td>5.2</td>
<td>Flow chart of the simulation methodology.</td>
<td>100</td>
</tr>
<tr>
<td>5.3</td>
<td>Energy $E$ vs. supply voltage $V_{dd}$ for TRW-S implemented with AC using: (a) approximate adder 1, (b) approximate adder 2, (c) approximate adder 3, and (d) approximate adder 4 [21]. It can be seen that AC can tolerate at most 10% scaling in $V_{dd}$, whereas when combined with ANT, up to 34% scaling in $V_{dd}$ can be tolerated.</td>
<td>101</td>
</tr>
<tr>
<td>6.1</td>
<td>The Bayesian framework for ANT.</td>
<td>108</td>
</tr>
</tbody>
</table>
6.2 Example error PMFs: (a) depiction of $P_\eta$ and $P_e$ that increase or decrease in distance from the mean, and (b) voltage overscaling (VOS) induced timing errors. ............................................. 108

6.3 Simulation results for a simple example: (a) probability of correct detection and MSE vs. $\tau$, (b) error PMF, and (c) $\tau_{s,p}$ and $\tau_{a,p}$ for different PMFs. ................................................................. 112

6.4 Block diagram of the simulation setup for the DCT application. ............ 114

6.5 DCT example: (a) error statistics of a voltage overscaled DCT block at $V_{dd} = 1.1V$ ($p_\eta = 0.0043$), and $V_{dd} = 1.0V$ ($p_\eta = 0.0374$), and (b) the resulting probability of correct detection and MSE vs. $\tau$. ....................... 115
# List of Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AC</td>
<td>approximate computing</td>
</tr>
<tr>
<td>ANT</td>
<td>algorithmic noise tolerance</td>
</tr>
<tr>
<td>AS</td>
<td>add substract</td>
</tr>
<tr>
<td>BER</td>
<td>bit error rate</td>
</tr>
<tr>
<td>BP</td>
<td>belief propagation</td>
</tr>
<tr>
<td>CDMA</td>
<td>code division multiple access</td>
</tr>
<tr>
<td>CMOS</td>
<td>complementary metal-oxide semiconductor</td>
</tr>
<tr>
<td>CNFET</td>
<td>carbon nanotube field effect transistor</td>
</tr>
<tr>
<td>CS</td>
<td>compare select</td>
</tr>
<tr>
<td>DCT</td>
<td>discrete cosine transform</td>
</tr>
<tr>
<td>DMR</td>
<td>double modular redundancy</td>
</tr>
<tr>
<td>DNA</td>
<td>deoxyribonucleic acid</td>
</tr>
<tr>
<td>DSP</td>
<td>digital signal processing</td>
</tr>
<tr>
<td>EDS</td>
<td>error detection sequential</td>
</tr>
<tr>
<td>ERSA</td>
<td>error-resilient system architecture</td>
</tr>
<tr>
<td>FF</td>
<td>flip-flop</td>
</tr>
<tr>
<td>FIR</td>
<td>finite impulse response</td>
</tr>
<tr>
<td>FOM</td>
<td>figure of merit</td>
</tr>
<tr>
<td>FPGA</td>
<td>field-programmable gate array</td>
</tr>
<tr>
<td>GMM</td>
<td>Gaussian mixture model</td>
</tr>
<tr>
<td>Acronym</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>HBP</td>
<td>hierarchical belief propagation</td>
</tr>
<tr>
<td>HMM</td>
<td>hidden Markov model</td>
</tr>
<tr>
<td>HW</td>
<td>hardware</td>
</tr>
<tr>
<td>IC</td>
<td>integrated circuit</td>
</tr>
<tr>
<td>ITRS</td>
<td>International Technology Roadmap for Semiconductors</td>
</tr>
<tr>
<td>LAN</td>
<td>local area network</td>
</tr>
<tr>
<td>LDPC</td>
<td>low-density parity check</td>
</tr>
<tr>
<td>LFSR</td>
<td>linear feedback shift register</td>
</tr>
<tr>
<td>LLR</td>
<td>log likelihood ratio</td>
</tr>
<tr>
<td>LOA</td>
<td>lower-part-OR adder</td>
</tr>
<tr>
<td>LSB</td>
<td>least significant bit</td>
</tr>
<tr>
<td>MAE</td>
<td>minimum absolute error</td>
</tr>
<tr>
<td>MAP</td>
<td>maximum a posteriori</td>
</tr>
<tr>
<td>ML</td>
<td>machine learning</td>
</tr>
<tr>
<td>MMAE</td>
<td>minimum mean absolute error</td>
</tr>
<tr>
<td>MMSE</td>
<td>minimum mean square error</td>
</tr>
<tr>
<td>MP</td>
<td>message passing</td>
</tr>
<tr>
<td>MRF</td>
<td>Markov random field</td>
</tr>
<tr>
<td>MSB</td>
<td>most significant bit</td>
</tr>
<tr>
<td>MSE</td>
<td>mean square error</td>
</tr>
<tr>
<td>NBTI</td>
<td>negative bias temperature instability</td>
</tr>
<tr>
<td>NMR</td>
<td>$N$-modular redundancy</td>
</tr>
<tr>
<td>PCA</td>
<td>principal component analysis</td>
</tr>
<tr>
<td>PDF</td>
<td>probability density function</td>
</tr>
<tr>
<td>PGM</td>
<td>probabilistic graphical model</td>
</tr>
<tr>
<td>PMF</td>
<td>probability mass function</td>
</tr>
<tr>
<td>PN</td>
<td>pseudorandom noise</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Definition</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------------------------------------</td>
</tr>
<tr>
<td>PoFF</td>
<td>point of first failure</td>
</tr>
<tr>
<td>PSNR</td>
<td>peak signal-to-noise ratio</td>
</tr>
<tr>
<td>PVT</td>
<td>process, voltage, and temperature</td>
</tr>
<tr>
<td>RCA</td>
<td>ripple carry adder</td>
</tr>
<tr>
<td>RMS</td>
<td>recognition, mining, and synthesis</td>
</tr>
<tr>
<td>RPR</td>
<td>reduced precision redundancy</td>
</tr>
<tr>
<td>RTL</td>
<td>register-transfer level</td>
</tr>
<tr>
<td>SAD</td>
<td>sum of absolute difference</td>
</tr>
<tr>
<td>SBP</td>
<td>sign bit protection</td>
</tr>
<tr>
<td>SEC</td>
<td>statistical error compensation</td>
</tr>
<tr>
<td>SNR</td>
<td>signal-to-noise ratio</td>
</tr>
<tr>
<td>SOC</td>
<td>system-on-chip</td>
</tr>
<tr>
<td>SRAM</td>
<td>static random access memory</td>
</tr>
<tr>
<td>SSNOC</td>
<td>stochastic sensor network-on-a-chip</td>
</tr>
<tr>
<td>SW</td>
<td>software</td>
</tr>
<tr>
<td>TMR</td>
<td>triple modular redundancy</td>
</tr>
<tr>
<td>TRC</td>
<td>tunable replica circuit</td>
</tr>
<tr>
<td>TRW-S</td>
<td>sequential tree-rewighted message passing</td>
</tr>
<tr>
<td>VLSI</td>
<td>very large scale integration</td>
</tr>
<tr>
<td>VOS</td>
<td>voltage overscaling</td>
</tr>
<tr>
<td>WLAN</td>
<td>wireless LAN</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

New emerging applications such as recognition, mining, and synthesis (RMS) [1] process massive amounts of data in order to derive inferences and enable decision-making tasks. Machine learning (ML), in a broad sense, is being successfully applied to these problems. These applications have the characteristic where the system-level performance is determined by a metric that is statistical in nature. Applications such as pseudo noise (PN) code acquisition are one such class where the performance is measured via a statistical metric known as probability of detection, $P_{\text{det}}$. ML applications, which span broad areas including computer vision, speech recognition, search engines, and deoxyribonucleic acid (DNA) sequencing, also fall within this category. Figure 1.1 depicts examples of such applications that include stereo image matching, background segmentation, and sound source separation, among others.

Machine learning has found its way into commercial products such as Apple’s Siri, Samsung’s S Voice, and Google Now, which are intelligent personal assistants and knowledge navigators that provide answers to queries, make recommendations, and processes many other requests through a natural language user interface. In the future, such machine learning applications will be deeply embedded in our daily lives. It is predicted that the number of embedded processors per person may exceed 1,000 by 2015 [2]. Massive amount of data will be collected by these ubiquitous sensors surrounding our environment, and machine learning will play a critical role in processing data collected by these sensors. Moreover, biomedical applications will be an important application, whereby one’s health can be monitored 24/7. To enable such rich computation in a small form factor, the availability of low-power hardware (HW) machine learning kernel accelerators is critical.

Traditionally, energy efficiency has been achieved by scaling of feature sizes in CMOS process technologies. As the load capacitance decreases along with the supply voltage, dy-
Figure 1.1: Machine learning applications: (a) computer vision (stereo image matching and background segmentation), and (b) machine listening (sound source separation).
Subthreshold Leakage As an Extra Complication

Figure 1.2: Ever increasing VLSI power [3]: (a) increase in leakage power due to technology scaling, and (b) expected power per chip. The increase in leakage power may create a large discrepancy from the ITRS requirements.

Dynamic power has shown a steady decrease with technology scaling (Fig. 1.2(a)). However, as increasingly complex and rich functionality are being integrated on-chip, the total power requirements are increasing exponentially [3]. Furthermore, with deep scaling, the gap between the supply and threshold voltage is decreasing, resulting in higher leakage power, as shown in Fig. 1.2(a). This increase in leakage power is expected to be much larger than the dynamic power savings obtained through scaling, creating a large discrepancy from the International Technology Road Map for Semiconductors (ITRS) [4] power target (Fig. 1.2(b)). This increase in power dissipation will inevitably lead to an increase in temperature. As shown in Fig. 1.3(a), high temperatures also result in increased leakage and exacerbate the energy challenge. The relatively constant thermal constraint across process technology generations suggests that the thermal limit may become the key bottleneck instead of power (Fig. 1.3(b)).

To make matters worse, deep scaling creates reliability problems that have been previously ignored. This is due to the fact that with aggressive scaling, artifacts such as process, voltage, and temperature (PVT) variations significantly affect the operation of modern nanoscale complementary metal–oxide semiconductor (CMOS) devices. The extent of variation in
CMOS devices [6] increases with scaling. Figure 1.4(a) shows that the variance of device threshold voltages increases as feature sizes shrink from 28 nm through 20 nm to 14 nm. Note that an increasing fraction of transistors are fabricated with threshold voltages at or below zero, which means they no longer operate as switches, resulting in faulty operation that cannot be mitigated even through conservative worst-case design methodologies. Figure 1.4(b) shows the delay (represented by colors) of memory cells in a 32 nm 64 kilobit static random access memory (SRAM) array. The color differences represent a 70% difference in delay. Unfortunately, this trend of increased variation and defects resulting in low reliability is true for post-silicon devices too. Figure 1.5 shows the $I_{DS}$ vs. $V_{DS}$ curve for 50 different carbon nanotube field effect transistors (CNFETs) [7]. As the figure shows, the saturation current has almost an order of magnitude difference ranging from 22 $\mu$A to 98 $\mu$A.

This large delay variation in deeply scaled CMOS and post-silicon devices can severely impact the reliability of modern integrated circuits and information processing systems. However, conventional worst-case design techniques incur a heavy energy penalty. On the other hand, nominal-case designs, though energy efficient, fail to meet the robustness specifications of the applications. In fact, in its 2007 report [4], ITRS forecasts that “Relaxing the
Figure 1.4: Variation in deep scaled CMOS devices [6]: (a) threshold voltage ($V_{th}$) at different process technology, and (b) delay of memory cells, represented by color, of a 32 nm 64 Kbit SRAM cell obtained via simulations.

Figure 1.5: $I_{DS}$ vs. $V_{DS}$ curve for 50 different CNFETs [7].
requirement of 100% correctness for devices and interconnects may dramatically reduce costs of manufacturing, verification, and test. Such a paradigm shift is likely forced in any case by technology scaling, which leads to more transient and permanent failures of signals, logic values, devices, and interconnects.” In short, enforcing an ideal deterministic abstraction on the underlying CMOS circuit will incur significant overhead, and error aware system design is essential in order to continue to reap the benefits of technology scaling.

1.1 Research Direction

In this dissertation, we describe a class of solutions that can simultaneously achieve robustness and energy efficiency in the implementation of deeply scaled ML kernels. Robust behavior of communication systems based on Shannon’s information theory [8], inspire the use of similar techniques for robust computation. Statistical system-level performance metrics are essential in the use of communication-inspired techniques, and provide a means to trade off reliability with energy efficiency. Figure 1.6 shows HSPICE simulation results of a four-tap filter in 180nm CMOS subject to voltage overscaling (VOS). If the errors are fully compensated for without additional overhead, energy reductions of up to 9× can be achieved by operating at a significantly lower supply voltage over a system operating at the point of first failure (PoFF). As we show throughout this dissertation, statistical error compensation (SEC) is a communication-inspired/Shannon-inspired technique that provides this low overhead error compensation. SEC is a novel technique that can be applied to DSP, communications, and ML systems to achieve simultaneous robustness and energy efficiency.

In the remainder of this chapter, we present the background information related to this work. We begin by discussing conventional techniques used for robust system design. SEC is then presented with a summary of the techniques that have been developed.
Figure 1.6: Simulation results of voltage overscaling (VOS) for a 4-tap correlation filter (a sensor) at 50 MHz in a 180 nm CMOS process.

1.2 Robust System Design

Robust system design has a long history. As early as 1821, Charles Babbage and John Herschel discussed methods on achieving infallible machines [9]. They viewed failure as a matter related to *purpose*, not to whether an item is physically intact or not. This view highlights the importance of measuring the reliability of a system based on its performance at the application level. The SEC techniques discussed in this dissertation leverage this viewpoint.

Robustness can be discussed at various levels of design abstraction: physical, logical, algorithmic, and system/application. The following summarizes the characteristic of the impairments and available robust design techniques at each level:

- **physical level**: this level deals with the physical aspects of the circuit and device such as geometry, layout, and material properties. Impairments at this level are known as *defects*. A short/open circuit in the manufacturing process is one example of a defect. Negative bias temperature instability (NBTI) [10] or an open circuit caused by electromigration [11] are examples of defects that develop over time due to aging. Defects
can remain dormant, until a particular input excites the defect, which then manifests itself as a *fault* at the logic level. Defects are usually circumvented through the use of worst-case design (over designing), or detected and corrected (eliminated) through testing procedures such as burn-in [12] and periodic maintenance where sensitive parts are replaced (preventive measure against aging).

- **logic level**: Impairments at this level are known as *faults*. A short/open circuit defect can be modeled as stuck-at-faults, where the output of a gate is always a 0 or 1. Even though the gate outputs may be in fault, it does not affect the system operation unless the incorrect values are latched as *errors* at the algorithmic level. Fault avoidance is usually achieved through defect avoidance, and fault tolerant techniques [13,14] require the use of some form of redundancy to detect and circumvent faults before they develop into errors.

- **algorithmic level**: Once incorrect values are latched, they become errors, i.e., incorrect data or internal states. Errors in digital systems manifest as bit flips ($0 \rightarrow 1$ or $1 \rightarrow 0$) in the storage element (latch). Though faults are one source of errors, in modern information systems, complexity makes verification of the design a very difficult task, and design failures, such as bugs in hardware and software, become another source of errors. Thus, in addition to fault avoidance, rigorous verification and testing is used to identify bugs and design errors. To tolerate errors, like faults, redundancy is employed, but at the information level. Error correcting codes have been successfully used on memory to combat errors, while techniques such as checkpointing [15] and Razor [16,17] have been successfully applied to achieve error resilient computation. SEC also performs error compensation at this level.

- **system/application level**: this level is where the actual functionality (purpose) of the system is visible. Impairments at this level can be attributed to system failure and result in catastrophic events. Software-oriented failure tolerance techniques include atomic broadcasting, agreement and commit protocols, synchronization, global state determination, and various other distributed algorithms [18]. Many applications
that employ statistical metrics possess inherent robustness. Especially emerging RMS
applications and machine learning based applications have large inherent application
level robustness, which can or should be exploited by robustness techniques designed
at lower levels. SEC exploits this application-level information at the algorithmic-level
to achieve significant reduction in compensation complexity.

Robust system design enables a system to continuously operate even in the presence of
defects, faults, or errors. Robust design dates back to John von Neumann’s work [19],
where logic networks composed of noisy or probabilistic gates were considered and signal
replication with majority voting was proposed to increase resiliency. This is known as N-
modular redundancy (NMR). Due to its large overhead, in practice, such majority voting
techniques are employed today at the architecture level in mission critical applications and
servers. Thus, fault tolerance is usually obtained by adding some degree of redundancy.
The most widely used redundancy techniques require additional silicon area (by replicating
blocks) and/or time (by performing re-computation). A summary of current robust design
techniques is provided in Table 1.1.

Robust design techniques can be classified depending on whether the defects, faults, or
errors are being avoided or corrected. System aware design adds another dimension where
certain errors can be allowed to propagate to the system level, if the application has sufficient
tolerance for errors. Inference based applications including ML applications, communication
systems, and many DSP systems possess inherent application- or system-level error resiliency.
This is due to the following:

1. The performance metrics for these applications are statistical. Inference applications
   including ML are measured by the probability of correct operation, such as probability
   of detection, or probability of correct classification. Communication systems are based
   on bit error rate (BER), while many filters are measured by SNR based metrics. Such
   metrics show that the application is designed with the possibility of failure in mind.
   Thus, even in the presence of HW errors, the application may provide acceptable
   performance.

2. The modeling of the application and the input data is imperfect. In speech recognition,
Table 1.1: Various robust system design techniques applied at different levels.

<table>
<thead>
<tr>
<th>circuit/process</th>
<th>latch/gate</th>
<th>algorithm micro-architectural</th>
<th>system</th>
</tr>
</thead>
<tbody>
<tr>
<td>burn-in testing [12]</td>
<td></td>
<td>coding [23]</td>
<td>best effort computing</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Razor [16, 17]</td>
<td></td>
</tr>
</tbody>
</table>
Gaussian mixture models (GMM) and hidden Markov models (HMM) for extracting phenoms have been used with great success [25]. However, this is not a perfect model of a real speech system, and many approximations are made to reduce the complexity of the algorithm. In weather prediction, supercomputers are used to process and analyze the data based on a sophisticated model. Due to practical reasons, many factors that influence the weather are not present in the modeling, and the input data to the model, i.e., temperature, atmospheric pressure, wind speed, etc., cannot be measured at every location. Thus, such applications are designed to be robust to modeling and data errors. In addition, if computation errors are introduced, but controlled to be within some bounds, severe degradation in system performance can be prevented.

3. In many inference applications, it is difficult to define a golden output. Recommendation systems such as those used in Netflix and Amazon are good examples. Based on a user’s history and preference, recommendations are made. Sometimes several recommendations are provided to the user in a ranked order. However, it is hard to know whether the recommended results are actually the true correct values. The user may not be sure of his or her preference, and it may change depending on the situation. Web search and data analytics are other examples. Thus, some errors in the computation process that alter the results may not necessarily be regarded as a system error.

4. Human interaction is another important factor. Limited perceptual capability of humans, and the user willingness to accept less-than-perfect results, all give rise to the acceptance of erroneous results.

To achieve optimal robustness in a design, a combination of such techniques should be employed. Table 1.2 summarizes the characteristic of each robustness technique. SEC is the only technique that tries to actively compensate errors in a non-deterministic fashion, while utilizing system-level information.

A uniform error model is used to facilitate the analysis and comparison between robust design techniques. Two different types of errors exist: hardware errors and estimation errors. Hardware errors, denoted as \( \eta \), are errors that occur within the computational block. Timing
Table 1.2: Characteristic of robust system design techniques.

<table>
<thead>
<tr>
<th>technique</th>
<th>avoidance</th>
<th>compensation/correction</th>
<th>system/application aware</th>
</tr>
</thead>
<tbody>
<tr>
<td>worst-case design</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>burn-in testing</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>N-modular redundancy</td>
<td>✓</td>
<td>(deterministic)</td>
<td></td>
</tr>
<tr>
<td>checkpointing</td>
<td>✓</td>
<td>(deterministic)</td>
<td></td>
</tr>
<tr>
<td>coding</td>
<td>✓</td>
<td>(deterministic)</td>
<td></td>
</tr>
<tr>
<td>Razor</td>
<td>✓</td>
<td>(deterministic)</td>
<td></td>
</tr>
<tr>
<td>approximate computing</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>best effort computing</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>SEC</td>
<td></td>
<td>✓ (stochastic)</td>
<td>✓</td>
</tr>
</tbody>
</table>

Figure 1.7: Error model used within this dissertation. The actual output $y_i$ consists of the ideal output $y_o$, hardware errors $\eta$, and estimation errors $e$.

Errors due to PVT variation, soft errors due to alpha particle hits, and errors induced by circuit defects are all different forms of hardware errors. These errors are dynamic errors and give rise to stochastic behavior. Estimation errors, denoted as $e$, are errors that are induced by design. Algorithmic simplifications such as subsampling, approximation, using fewer iterations, or reducing hardware complexity via reduced precision all give rise to estimation errors at the output. The additive error model (see Fig. 1.7) captures both types of errors, and is given by:

$$y_i = y_o + \eta_i + e_i$$  \hspace{1cm} (1.1)

where $y_o$ is the ideal error free output, $\eta_i$ is the hardware errors, and $e_i$ is the estimation error of the $i^{th}$ output $y_i$.

Next, we describe some of the well-known robust system design techniques.
1.2.1 N-modular redundancy (NMR)

The basic concept of NMR (see Fig. 1.8) is to replicate the processing element (PE) $N$ times, then vote (majority, plurality) to determine the final output of the system. NMR has an area overhead, as additional silicon area is required to replicate the PEs. In NMR, the output of each block is assumed to be corrupted by hardware errors only, i.e., the following error model is employed:

$$y_i = y_o + \eta$$  \hspace{1cm} (1.2)  

which is obtained from (1.1) by substituting $e_i = 0$.

Triple-MR (TMR) [20] is most widely used because the overhead of replicating for $N > 3$ is very large. In TMR, if a fault occurs in one PE, the majority voter can generate a correct output and keep the system in operation. In NMR (see Fig. 1.8), the voter becomes a single-point-of-failure, i.e., voter failure leads to system failure, and should be designed to operate error free. One possible solution is to triplicate the voter and have three independent outputs. If these outputs are inputs to another TMR configuration, then each output could be connected to each replicated block as shown in Fig. 1.9. An example of such a system is the Tandem Integrity S2, which was designed to meet the need for a reliable UNIX-based computing system [26].

In cases where all three outputs do not agree, a TMR system will detect but will not be able to correct errors. Further, an undetected error will occur if a majority of the outputs exhibit the same error.
1.2.2 Checkpointing

A checkpoint is a snapshot of the entire state of the system at a specific time. During computation, when a fault has been detected, the computation falls back to the most recent checkpoint and resumes execution [15]. Thus, checkpointing is a fault-tolerant technique that exploits time redundancy. These checkpoints need to be stored in a stable and reliable memory, which is now the single-point-of-failure. The error model for checkpointing is the same as in NMR (1.2). Error detection mechanisms must be incorporated in the design to minimize the re-computation in case of a fault. Checkpointing generally provides more reliable operation than does an NMR system, as in checkpointing, errors are corrected by re-computation, while in NMR, forward error correction is performed via a majority vote. However, as there are certain operations that cannot be undone, such as printing a document or launching a missile, the system must be designed to deliver these outputs only when it is certain there is no need to undo them.

1.2.3 Error resilient processors

Error resilient processors [27–30] employ logic-level techniques to detect errors and architectural-level techniques to correct them. For example, in Razor [16, 17], a shadow latch is employed that latches at a slightly delayed time from the main latch. If the critical path was violated and the main latch was in error, the shadow latch will latch on to the correct value, and an error can be detected by comparing the two. This type of error resiliency to timing errors has been shown to be an effective approach for combating variations while achieving
energy efficiency. Voltage overscaling (VOS) [31] was employed in [27–30] to induce timing errors by reducing the supply voltage ($V_{dd}$) below the critical voltage ($V_{dd,crit}$), which is the point of first failure (PoFF). The error rate, $p_\eta$ (percentage of clock cycles in which the output is in error), increases as $V_{dd}$ is reduced. Razor I [27] employs VOS along with in situ local (FF level) timing error detection and local correction in order to reduce energy while combating variations. Razor I demonstrated that at an error rate of $10^{-7}$, which is near the PoFF, the error-correction overhead is minimal, and energy-efficiency gains of 14% to 17% can be obtained when compared to an error-free architecture operating just above the PoFF. Razor II [28,29] employs local error detection and architectural replay to operate at an error rate of $p_\eta = 4 \times 10^{-4}$, which is also near PoFF, while achieving an energy savings of 33% to 35%. An error-resilient microprocessor core in a 45 nm process [30] employs an error-detection sequential (EDS) and a tunable replica circuit (TRC) to achieve 41% throughput gain or 22% energy reduction, with a 10% $V_{dd}$ droop. However, these techniques strive to deliver exact correctness (deterministic output), which may not be needed by the application, and are severely limited in operable error rates ($p_\eta < 10^{-3}$) and thus achievable energy efficiency. The error detection in these processors is at the logic level. Hence the error model in (1.1) does not apply directly.

Error resilient system architecture (ERSA) [32] has been proposed as an alternative to designing robust processors. ERSA relaxes the requirements of exact correctness. This architecture has one reliable processor core, the super-reliable core (SRC), along with several unreliable cores, known as relaxed reliability cores (RRCs). ERSA is shown to achieve resilience to higher-order bit errors and maintain sufficient accuracy even at very high error rates with minimum impact on execution time.

1.2.4 Approximate computing

Emerging new applications, such as recognition, mining, and synthesis [1], process large amounts of noisy data via statistical and probabilistic computation and are inherently error resilient. Conventional digital signal processing (DSP) applications including image, audio, and communications also possess such inherent reliability. This motivates the use of approx-
imate computing, in which the requirement of numerical exactness on the outputs is relaxed. Several factors such as limited perceptual capability of humans, difficulty to define a golden output (e.g., web search, data analytics), and the user willingness to accept less-than-perfect results all contribute to justification of approximate computing. The goal of approximate computing is to simplify the implementation of computation to gain energy efficiency or to obtain higher performance.

The distinctive feature of AC is that it does not involve assumptions about the stochastic nature of any underlying process implementing the system. It does, however, often utilize statistical properties of data and algorithms to trade quality for energy reduction. AC, hence, employs deterministic designs that produce imprecise results [22], i.e., AC incurs estimation errors in the output and employs the following error model:

\[ y = y_o + e \]  

which is obtained from (1.1) by substituting \( \eta = 0 \). SEC, detailed in the following section, embraces the stochastic nature of the underlying implementation to provide compensation such that the output may be in error but guarantee that the application-level specifications are met.

Approximate computing has been applied at the circuit and algorithmic level. At the circuit level, several arithmetic circuits have been proposed. Adders are partitioned into two parts [21,33], where the more-significant bits are computed in an exact manner, while the less-significant bits are computed in an inexact manner via a simplified circuit. Approximate mirror adders [21,33], approximate XOR/XNOR-based adders [34], and lower-part-OR adders [35] are some examples. Approximate multipliers are constructed via the use of speculative adders for obtaining partial products [35–38]. Approximate logic-level synthesis has been explored as well [39–42]. At the algorithmic level, incremental refinement, a characteristic of iterative algorithms, has been exploited to achieve results that gradually increase in quality [43–47]. To increase energy efficiency, dynamic bit width adaptation has been proposed [48]. A more detailed survey of AC is presented in Chapter 5.
1.3 Statistical Error Compensation

Conventional robust design techniques are usually high in overhead (when applied at circuit/logic levels), or are unable to provide sufficient protection (when applied at the system level). In this dissertation we propose the use of statistical error compensation (SEC). Statistical error compensation techniques, such as algorithmic noise tolerance (ANT) [31], employ statistical estimation and detection techniques to compensate for errors approximately. These methods are applied at the algorithm level and compensate for erroneous behavior, while still utilizing the system-level information. In short, catastrophic algorithm-level errors that lead to system failure are converted to benign errors that the system can absorb. Thus, SEC techniques use a combination of system aware design (utilizing system-level inherent robustness for failure avoidance) and error compensation (approximate error compensation at the algorithm level) as shown in Table 1.2.

SEC techniques are best suited for applications where the performance metrics themselves are statistical. When SEC techniques are applied to these applications, significant power savings, e.g., up to 67% for a finite impulse response (FIR) filter [31], and robustness enhancement can be achieved, compared to logic- or architectural-level techniques such as Razor. A high-level depiction of SEC is given in Fig. 1.10(a). SEC utilizes the statistics of errors to perform detection and estimation to compensate for errors. It also incorporates system-level statistical metrics, such as signal-to-noise ratio (SNR), or bit error rate (BER). SEC operates on multiple observations, where each observation is generated by erroneous hardware, an error-free estimator, or an erroneous estimator. Each observation $y_i$ is a corrupted version of the correct output $y_o$, i.e., $y_i = y_o + \eta_i + e_i$. Based on these observations, detection and estimation techniques are employed in conjunction with the statistical information of $\eta_i$ and $e_i$ to obtain the output most likely to be correct. Errors that have a large effect on the system-level performance are detected and compensated, while errors with minimal effect on performance are considered benign and permitted.
1.3.1 Algorithmic noise tolerance

Statistical error compensation (SEC) in the form of algorithmic noise tolerance (ANT) [31, 49] in Fig. 1.10(b) incorporates a main block and an estimator. The main block, unlike the estimator, is permitted to make hardware or timing errors. The estimator is a low-complexity block (typically 5% to 20% of the main block complexity) generating a statistical estimate of the correct main block output, i.e.,

\[ y_a = y_o + \eta \quad \text{(1.4)} \]
\[ y_e = y_o + e \quad \text{(1.5)} \]

where \( y_a \) is the actual main block output, \( y_o \) is the error-free main block output, \( \eta \) is the hardware error, \( y_e \) is the estimator output, and \( e \) is the estimation error. Equations (1.4) and (1.5) can be obtained by substituting \( e_i = 0 \) and \( \eta_i = 0 \) in (1.1), respectively. Note that
the estimator exhibits estimation error $e$ because it is simpler than the main block. ANT exploits the difference in the statistics of $\eta$ and $e$ (see Fig. 1.10(c)). To enhance robustness, it is necessary that when $\eta \neq 0$, $\eta$ be large compared to $e$. In addition, the probability of the event $\eta \neq 0$ must be small. The final or corrected output of an ANT system $\hat{y}$ is obtained via the following decision rule:

$$
\hat{y} = \begin{cases} 
y_a, & \text{if } |y_a - y_e| < \tau \\
y_e, & \text{otherwise}
\end{cases}
$$

(1.6)

where $\tau$ is an application-dependent parameter chosen to maximize the performance of ANT. Under the conditions outlined above, it is possible to show that

$$
SNR_{uc} \ll SNR_e \ll SNR_{ANT} \approx SNR_o
$$

(1.7)

where $SNR_{uc}$, $SNR_e$, $SNR_{ANT}$, and $SNR_o$ are the signal-to-noise ratios of the uncorrected main block ($\eta$ dominates), the estimator ($e$ dominates), the ANT system, and the error-free main block (ideal), respectively. Thus, ANT detects and corrects errors approximately, but does so in a manner that satisfies an application-level performance specification (SNR). Several low-overhead estimation techniques have been proposed by exploiting data correlation, system architecture, and statistical signal processing [24].

Conventional fault-tolerant techniques focus on providing complete correctness while in operation. However, communication-inspired algorithmic noise tolerance (ANT) techniques [50] utilize the fact that some applications are tolerant to small errors and show significant improvement in robustness while providing energy efficiency. This is done not by concentrating on error-free output, but rather by trying to meet the signal-to-noise ratio (SNR) or bit error rate (BER) specification of the application. The communication-inspired approach treats nanometer circuit fabrics as a noisy channel, and faults and errors resulting from this channel are addressed through statistical signal processing techniques that were primarily used in communication systems for several decades.
1.3.2 Stochastic sensor network-on-a-chip (SSNOC)

Stochastic sensor network-on-chip (SSNOC) \cite{51,52} extends ANT and provides a novel alternative that employs continuous error compensation as opposed to sequential error detection and error correction as in \cite{27–30}. The main computation is divided into a number of smaller sub-computations. These sub-computations are said to be performed by “sensors” because of the resemblance of the overall system to a sensor network. Figure 1.11 illustrates an SSNOC system with four sensors. Traditional sensor networks consist of a large number of measurement nodes that communicate with each other, usually over wireless links, and estimate underlying physical phenomena (e.g., temperature or pollution). Their distributed nature makes sensor networks robust to local variations, node failures, and communication noise. The computational errors made by the sensors of the SSNOC may be viewed as analogous to measurement noise in sensor networks. This view of computation enables us to borrow concepts from estimation theory to build robust IC systems.

In a SSNOC, the outputs of the sensors are combined using a fusion block that employs
techniques from robust statistics [53] to provide system-level robustness to hardware errors in the sensors. The output of sensors can be corrupted by both hardware errors and estimation errors, i.e., (1.1) applies as is

\[ y_i = y_o + \eta_i + e_i \]  

(1.8)

which is in contrast to ANT where the outputs were either corrupted by hardware errors (1.4) or estimation errors (1.5) only. The fusion center needs to be designed to accommodate hardware errors in the sensors that may be difficult to characterize but are potentially severe. Computational errors can be viewed as probabilistic sources of noise that contaminate the measurement noise in the input, commonly modeled as Gaussian. The aggregate noise in the system (comprising computational errors and input noise) may be modeled as drawn from the following set of mixture distributions:

\[ P_\epsilon = \{ F | F = (1 - \epsilon)\Phi + \epsilon H \} \]  

(1.9)

where \( \Phi \) is the class of normal distributions, \( H \) is the class of arbitrary densities with zero mean and finite but unbounded variance, and \( 0 < \epsilon < 1 \) is a mixing parameter. The robust estimation problem is to find the maximum-likelihood estimate for the least informative distribution in the above mixture class. This estimate will be robust to the worst-case probability distribution of the errors caused due to variations. The theory of robust statistics [53] states that the robust estimate will be the solution to the following equation:

\[ \sum_{k=1}^{m} \psi[Y_k - \theta] = 0 \]  

(1.10)

where \( \psi \) is a general odd-symmetric function known as the influence function, and \( Y_k \) are the measurements. For the case of \( \epsilon \)-contaminated \( \mathcal{N}(0, 1) \) distributions, the influence function, \( \psi \), is given by [53],

\[ \psi(x) = \begin{cases} 
  x, & \text{if } |x| \leq \gamma \\
  \gamma \text{sgn}(x), & \text{else} 
\end{cases} \]  

(1.11)

where \( \text{sgn}(\cdot) \) is the sign function, and \( \gamma \) is a constant that depends only on \( \epsilon \) and the nominal
distribution, $\mathcal{N}(0,1)$.

The abstraction provided by the SSNOC can be directly applied to many applications in signal processing that can be parallelized. For instance, polyphase decomposition is often used to obtain a parallel implementation of FIR filters. Such filters are used in the matched filtering operation common in code division multiple access (CDMA) wireless receivers.

1.3.3 Soft NMR

ANT and SSNOC are heavily application dependent. NMR is a general computation technique that requires no knowledge of the application, but has limited performance gain. Soft NMR [54] combines the general applicability of NMR and enhancement in robustness of ANT in a very general technique that can be applied to nearly any computation. Like ANT, soft NMR also incorporates application-specific knowledge to dramatically improve the performance over conventional NMR. As shown in Fig. 1.12, soft NMR differs from NMR in that it incorporates a soft voter, which is composed of an estimator and a detector. Soft NMR views computation in the PEs as a noisy communication channel, i.e., the following error model is used:

$$y_i = y_o + \eta_i$$

which is obtained from (1.1) by substituting $e_i = 0$, and employs the estimator as an equalizer and the detector as the slicer.

The soft voter explicitly uses statistics including data and error statistics. Data statistics
is the distribution of the error-free output $y_o$, which is also known as the prior, while error statistics are the distribution of the hardware errors $\eta$. The soft voter then finds the most likely output, $\hat{y}$, among the hypotheses defined by a hypothesis set $\mathcal{H}$:

$$
\hat{y} = \arg \max_{\mathcal{H}_i \in \mathcal{H}} P(y_1, y_2, \ldots, y_N | \mathcal{H}_i)
$$

(1.13)

Optimally, $\mathcal{H}$ will contain all possible outputs $y_o$ with nonzero probability as given by the prior. However, in practical implementations, the size of $\mathcal{H}$ limits the performance of soft NMR, and thus the size of $\mathcal{H}$ needs to be limited. One good approximation is the set of all observations, $\mathcal{H} = \{y_i\}_{i=1}^N$.

1.4 Contributions of This Dissertation

In the past, SEC techniques have been applied to a wide range of applications in communication and DSP systems. In this dissertation, we extend the application of SEC to include machine learning applications. Specifically, we show that when SEC is applied to applications that possess inherent robustness, its effectiveness is significantly enhanced. The major contributions of the dissertation are summarized as follows:

1. An SSNOC based PN acquisition filter has been implemented in silicon and measured. Results show operation at error rates larger than 85.83%, with energy savings of 2.52×.

2. SEC has been applied to a belief propagation based communication system, low-density parity check (LDPC) codes. Combined with the inherent resiliency of LDPC codes, an SEC based LDPC decoder can operate at a supply voltage up to 38% less than the nominal voltage and tolerate up to 30× more errors over an SNR range of 3 dB to 8 dB, while maintaining less than 3× degradation in BER. This is equivalent to energy savings of 45.7% compared to conventional LDPC decoders, and 33.2% compared to a sign bit protected LDPC decoder.

3. SEC has been applied to a Markov random field based iterative stereo matching architecture. Correction is performed at several levels including arithmetic, iteration,
and system levels. The compensation overhead, robustness, and energy savings are characterized and compared among the different levels of compensation. Arithmetic compensation achieves power savings of 39.6% at an overhead of 97.4%. A hybrid approach can successfully trade off the compensation complexity and energy savings by achieving 16.1% additional power savings compared to arithmetic level while reducing the overhead to 57.9%.

4. A study on combining SEC and approximate computing has been performed to show that SEC can further extend the AC based design to achieve additional robustness and energy efficiency. Results show that ANT combined with AC achieves energy savings of 44.9% compared to a conventional system, while achieving at most 4% degradation in performance. This supports our view that embracing the stochasticity of the underlying process is essential to achieve significant energy savings.

5. Attempts to analyze SEC techniques have been made. An analysis framework has been proposed and under this framework, ANT is shown to be a Bayes optimal detector and estimator.

1.5 Dissertation Organization

This dissertation is organized as follows:

- **Chapter 1** has provided the necessary background information on fault-tolerant design methods and statistical error compensation design techniques.

- **Chapter 2** shows the benefits of SEC applied to detection and correlation based applications. A PN-code acquisition filter was chosen as an example. Through measurements on a 180 nm silicon implementation, it is shown that the SEC based PN code acquisition can achieve detection probability $P_{\text{det}} \geq 90\%$ at an error rate $p_\eta \geq 85.83$ with energy savings of $2.52 \times$.

- **Chapter 3** delves into machine learning based communication applications. SEC is applied to a low-density parity check (LDPC) decoder. As iterative message passing
based architectures are inherently robust to small magnitude errors, the SEC based LDPC decoder shows significant improvement in robustness and energy efficiency.

- **Chapter 4** applies SEC to a more generic machine learning application. With the success in LDPC decoders, SEC has been applied to a complex message-passing application, Markov random field (MRF) based stereo image matching. Measurements and analysis on the inherent robustness of message-passing architectures are explored. Then, the synergy between SEC and ML applications is shown through simulations and FPGA emulations.

- **Chapter 5** combines AC and SEC. AC is applied to the same stereo image-matching architecture of Chapter 4, and it is shown that SEC is essential in achieving significant energy savings.

- **Chapter 6** analyzes SEC techniques to provide insight into the design of optimal SEC based systems. In particular, ANT is shown to be a Bayes optimal detector and estimator.

- **Chapter 7** concludes this dissertation and provides directions for future research activities.
Chapter 2

The Design of an Error-Resilient PN Code Acquisition Prototype Chip

In this chapter, we describe the architecture and measurement results of an SEC based pseudorandom noise (PN) code acquisition filter. As discussed in Chapter 1, the performance of PN code acquisition is determined by its detection probability $P_{\text{det}}$, and is a good application for SEC.

Code division multiple access (CDMA) based communication systems employ PN code sequences to spread the bandwidth of the transmitted signal over a wide frequency band. This technique, employed in communication standards such as IEEE 802.11 WLAN [55], CDMA2000 [56], and WCDMA [57], is useful for multiple access communication, as the resulting transmit signal has a very low signal-to-noise ratio, and causes minimal interference. Conventionally, the receiver correlates the noisy received signal with a local copy of the PN code, and the output is then processed by a detector [58]. Matched filters with locally generated code sequence as tap weights are commonly employed for this purpose. Peaks in the output of the matched filter are employed for detection and synchronization of PN sequences. Code acquisition is a power hungry, computationally intensive operation in a spread-spectrum communication receiver [58]. For example, [59] shows that the matched filter consumes 20% to 25% of total receiver power. Reducing the power required to perform PN code acquisition is therefore essential for mobile wireless communication [60].

The matched filter based PN code acquisition is performed by searching over all possible code phases. At one extreme, this can be achieved by serially processing each code phase [61,62]; while at the other extreme, a full parallel search can be performed. A serial search has low area complexity while incurring a large acquisition and detection latency. On the other hand, a full parallel search can obtain a phase lock in a single search period, but results in large area overhead. Hybrid approaches have been utilized to trade off detection
latency and implementation complexity [63] to achieve low power operation. However, the complexity or latency trade-off is only linear, which results in a constant area delay product [64]. Shibano et al. [65,66] have employed analog techniques to achieve energy efficiency. Yet, such techniques lack programmability. Recent developments in iterative message-passing algorithms have inspired iterative PN acquisition techniques [67]. But, this technique incurs a large interconnect complexity as well as latency due to its iterative nature. These low-power PN code acquisition architectures sacrifice the simplicity of a digital matched filter.

In this chapter, we show that when SSNOC, also known as stochastic networked computation [52], is applied to a 256-tap PN code acquisition filter implemented in a 180 nm CMOS process [68], it can achieve $2.4 \times$ to $5.8 \times$ (ave 3.86×) energy savings over conventional designs, and $1.55 \times$ to $3.79 \times$ (ave 2.52×) energy efficiency and $2146 \times$ to $2281 \times$ (ave 2225×) in error tolerance over existing error resilient designs [27–30]. In terms of a figure of merit (FOM), defined as $(\text{power}/(\text{precision} \times \text{sample rate}))$, this translates to a 19.27× improvement compared to conventional PN code acquisition filters [69–71]. By enabling operation at significantly lower voltage, and being able to effectively compensate for VOS errors with a low error correction overhead, stochastic networked computation is able to achieve a significant fraction of the promised 9× energy savings of Fig. 1.6.

The remainder of the chapter is organized as follows. The PN code acquisition problem and the conventional matched filter based architectures are described in Section 2.1. The application of statistical error compensation to the PN acquisition filter is described in Section 2.2. The architecture of a prototype test chip is shown in Section 2.3. Section 2.4 presents measurement results, and Section 2.5 summarizes the chapter.

2.1 PN Code Acquisition Filter

Pseudorandom noise (PN) codes are a set of sequences whose cross correlation is zero, while its autocorrelation is nonzero only at lag zero. PN codes are generated via a linear feedback shift register (LFSR) as shown in Fig. 2.1. The LFSR can be described by a polynomial $x_{k+r} = g_1 x_{k+r-1} \oplus g_2 x_{k+r-2} \oplus \cdots \oplus g_r x_k$, where $\oplus$ is modulo 2 addition. An LFSR polynomial with degree $r$ that results in a sequence with the maximum period of $2^r - 1$, is referred to
Figure 2.1: Architecture of a linear feedback shift register (LFSR) used for PN code generation.

as a maximal length sequence (MLS) or an \( m \)-sequence. Such \( m \)-sequences are widely used as spreading sequences for spread spectrum systems such as CDMA.

In spread-spectrum systems, communication is possible only when the transmitted sequence matches the sequence stored in the receiver and the offsets are synchronized. This synchronization task is known as PN acquisition, and is performed by utilizing the correlation property of PN codes. The traditional architecture for a PN code acquisition system is a simple matched filter such as the one in Fig. 2.2(a) [69]. An \( N \)-tap PN code acquisition filter correlates the received signal \( x[n] \) with locally generated PN-code \( h_j \) as follows:

\[
y_o[n] = \sum_{j=0}^{N-1} h_j x[n - j]
\]  \hspace{1cm} (2.1)

where \( h_j \) represents the 1 b (1 bit) PN-code, and \( x[n] \) is the received signal. In our implementation, \( x[n] \) is chosen to be 8 b. A peak detector or a thresholding block is used to detect if there is a match via the equation:

\[
m = \text{sgn}(y_o[n] - \tau)
\]  \hspace{1cm} (2.2)

\[
\text{sgn}(a) = \begin{cases} 
1 & a > 0 \\
0 & \text{otherwise}
\end{cases}
\]  \hspace{1cm} (2.3)

Here, \( m \) is a binary variable indicating a match when \( m = 1 \) and no match when \( m = 0 \), and \( \tau \) is a user-defined threshold chosen to maximize the probability of detection at a specific false alarm rate.
2.2 SEC Based PN Code Acquisition Filter

The SEC implementation of a 256-tap PN code acquisition system is shown in Fig. 2.2(b) using stochastic networked computation (SNC) [52]. SNC decomposes the main computation into sub-blocks (sensors) whose outputs have identical mean and variance (statistically similar). For the PN acquisition application, a symmetric structure composed of a bank of identical sub-correlators is used and online error compensation is achieved using a fusion block. This is in contrast to sequential error detection and correction as in [27–30]. With a decomposition factor of 64, the 256-tap correlator in Fig. 2.2(b) is decomposed into 64 4-tap sub-correlators whose output $y_i[n]$ ($i = 0, \ldots, 63$) is given by

$$y_i[n] = \sum_{j=0}^{3} h_{4i+j} x[n - 4i - j]$$  \hspace{1cm} (2.4)

This sub-blocking approach is different from the polyphase decomposition used in [52]. Both perform statistically similar decomposition of the main block and thus will have minimal

Figure 2.2: PN code acquisition filter: (a) conventional, and (b) SEC based.
impact on performance, which was verified through simulations. The sub-blocking approach used in this chapter can achieve additional power savings (see Section 2.3.1).

It should be noted that the sum of all sub-correlators $\sum_{i=1}^{64} y_i[n]$ is equal to $y_o[n]$, the output of the conventional matched filter in (2.1). Each sub-correlator is referred to as a sensor and is subject to estimation errors $e_i[n]$, i.e., $y_i[n] = y_o[n] + e_i[n]$, which are uncorrelated (between different sensors). If additionally, the sensors are subject to VOS, timing errors $\eta_i[n]$ are induced as well, resulting in the following output:

$$y_i[n] = y_o[n] + e_i[n] + \eta_i[n] \quad (2.5)$$

The fusion block in Fig. 2.2(b) combines the 64 outputs of the sub-correlators employing Huber’s robust estimation theory [53] for error compensation. Huber’s robust estimation theory provides the optimal fusion algorithm for $\epsilon-$contaminated errors, i.e., the noise in the system (comprising $\eta_i[n]$ and $e_i[n]$) is modeled as drawn from the following set of mixture distributions:

$$P_\epsilon = \{F | F = (1 - \epsilon)\Phi + \epsilon H\} \quad (2.6)$$

where $\Phi$ is the class of normal distributions, $H$ is the class of arbitrary densities with zero mean and finite but unbounded variance, and $0 < \epsilon < 1$. The error model for the sub-correlator output in (2.5) has a distribution that approximates (2.6) because the estimation error $e_i[n]$ is small magnitude and Gaussian, and timing violations $\eta_i[n]$ are large and intermittent. Thus, the sum $e_i[n] + \eta_i[n] \approx \eta_i[n]$ when $\eta_i[n] \neq 0$, and hence $e_i[n] + \eta_i[n]$ is approximately $\epsilon$-contaminated.

Figure 2.3 plots the correct output $y_o[n]$ and measured outputs of four sensors, along with the mean and median of all 64 sensors, from the prototype test chip. It can be seen that most times the sensor outputs are close to $y_o[n]$, indicating that $\eta_i[n] = 0$, and $y_i[n] = y_o[n] + e_i[n]$, which is approximately Gaussian distributed. Once in a while, the sensor outputs deviate significantly from $y_o[n]$, indicating that $\eta_i[n] \neq 0$ and is large in magnitude. This is to be expected as MSB errors will occur in LSB-first computation.

The robust estimation problem is to find the maximum-likelihood estimate for the least informative distribution in the set $P_\epsilon \ (2.6)$ for a given $\epsilon$. This estimate will be robust to the
Figure 2.3: Measured sensor outputs over time. Outliers in the sensor output are due to hardware errors $\eta_i$ and can be seen to shift the mean significantly.

The worst-case probability distribution of the errors caused due to variations. Huber’s theory [53] enables one to determine a fusion algorithm that finds the solution to this robust estimation problem. This robustness property of the Huber estimator is useful as the distribution of $\eta_i[n]$ may not be known accurately because timing violations are a function of a number of parameters including process, temperature, and voltage. However, this (worst-case) optimal fusion block was shown to be very complex [52], and a simplification of this algorithm leads to the median block, which performs equally well with significant complexity reduction. The median and mean fusion is shown to compensate for errors effectively. Hence, in this dissertation, we employ the median and mean as algorithms to fuse the sensor outputs.

2.3 Chip Architecture

Figure 2.4 shows the architecture of the PN code acquisition test chip consisting of a total of 64 sensors, and a fusion block implementing two algorithms: (a) mean and (b) hierarchical median. An adaptive thresholding block at the output determines whether a detection has
Figure 2.4: Prototype IC architecture.
occurred. Additional logic to control the threshold block and select the correct tap to load, is placed in a control block (not shown).

2.3.1 Sensor architecture

Each sensor is a four-tap FIR filter (Fig. 2.5) with 8 b data \((x[n])\) and 1 b coefficient \((h_j)\), and a 10 b output whose two LSBs are dropped to generate a final 8 b output \(\hat{y}_0[n]\). To reduce complexity, the PN code bits are shifted rather than shifting the 8-bit data. This reduces the number of registers that need to be updated from 2048 (256 8 b data registers) to 264 (a single 8 b data register and the 256 coefficient registers). Each sensor, except for Sensor-1, takes the next coefficient from the previous sensor as input. An external \texttt{pnload} signal switches the Sensor-1 coefficient input between the final coefficient from Sensor-64 and the external signal \texttt{pnnin}. In addition to the FIR filter, a mod-24 state counter determines when to advance the coefficients and latch new inputs and outputs, and the final two bits of the data load counter are decoded to determine which data value, if any, to overwrite. The latches that advance the coefficient and data operate with a different clock as shown in Fig. 2.5 (shaded vs. unshaded latch) with the clock for PN code bits being 24× faster. Three configuration bits, set to constants based on the sensor number during instantiation and thus optimized out during synthesis, modify the operation of the data load decoder. Finally, the sensor value is output serially.
2.3.2 Fusion block architecture

The fusion block implements a three-stage hierarchical mean or median function to avoid global interconnect and simplify the implementation of a 64-median block. The mean is a simple addition of four values at each level, with a depth of three. The hierarchical median operates by grouping the input data, taking the median of each group, and repeating the procedure until only one group remains. Two- and three-level structures with groups of eight and four, respectively, were tested. In addition, to attempt to mitigate the loss of accuracy resulting from grouping the data, overlapping structures in which each input is put in multiple groups were also considered. These use group sizes of 16 and 8, respectively, except for the final level. The accuracy of these structures in estimating the median is dependent on how the inputs are grouped; to estimate the effect, the integers from 0 to 63 are randomly shuffled 100,000 times and used as inputs to the algorithm. A histogram of the final outputs for each hierarchical median structure when selecting the lower of the central values at each stage is shown in Fig. 2.6; selecting the higher instead would produce a mirror image of the plot. The most frequently selected values are 26 for the simple two-level algorithm, 29 for the overlapping two-level algorithm, 18 for the simple three-level algorithm, and 25 for the overlapping three-level algorithm. The performance of each of these algorithms is shown in Fig. 2.7. Interestingly, each algorithm performs noticeably better when selecting the lower of the central values rather than the higher. When the code sequence is present, nearly all sensor outputs should be high, but the distribution of sensor outputs is approximately Gaussian when the sequence is not present. Thus, selecting the lower value is more likely to produce a small output when the sequence is not present while still producing a large output when the sequence is present. In general, the overlapping structures perform best and the two-level structures are more accurate than the respective three-level structures. Using the lower of the central values, the overlapping three-level structure has a higher probability of detection than the simple two-level structure. The resulting hierarchical median algorithm employs a three-level hierarchy, where the two lower levels of the median hierarchy have eight inputs and computes the fifth largest input, while the final fusion block computes the third largest of four inputs.
The median block is based on [72] (Fig. 2.8). The algorithm requires one mask bit \( m_i \) and one set bit \( s_i \) per input to be passed from stage to stage. For the most significant bit, the mask bits are all initialized to 1 and the set bits are not used. At each stage, if a mask bit is 0, the corresponding input bit is replaced with its set bit. Then the majority value \( z \) of the masked inputs is determined. This value \( z \) is the output bit for this stage of the median filter. For the next stage, any inputs that do not already have a mask bit of 0 and do not match \( z \) have their mask bits set to 0 and their set bits to \( z \). Effectively, an input that is known to be below the median is treated as 0, and an input known to be above the median is treated as 1 in all remaining stages. The median bit slice structure can be seen in Fig. 2.8.

In addition to the fusion structures themselves, a decoder (see Fig. 2.9) for two bits of the data load counter is included in the fusion block. The outputs of this decoder are passed to the lower-level fusion blocks or the sensors, as appropriate, as load enable signals. The input and output registers for the adder and median bit slice are disabled when a different algorithm is selected to reduce the number of register switches.
Figure 2.7: Simulated ROCs for hierarchical median: (a) simple two-level, (b) simple three-level, (c) overlapping two-level and (d) overlapping three-level.
Figure 2.8: Bit slice of the median block.

Figure 2.9: Fusion block architecture.
2.3.3 Threshold block architecture

The threshold block is illustrated in Fig. 2.10. The value in the threshold register is adjusted based on the user-selected false alarm rate and selected fusion algorithm. Because the hierarchical median algorithm produces a value lower than the mean, the initial threshold upon reset must be lower than that of the mean. The appropriate initial threshold values were chosen through simulations, which were 8 for the mean and 0 for the median. The output of the final fusion is compared to the threshold, and the comparison result is sent to the external output cmpout.

It is assumed that a PN code match is rare, and thus the number of positive responses is a reasonable estimate of the false alarm rate. In order to maintain the selected rate, the number of positive responses is counted for 1024 cycles. At this time, the count is compared with predetermined values according to the user-selected false alarm rate. If this count is outside the prescribed range, the threshold is incremented or decremented as necessary to bring the count closer to the desired range. This process is then repeated so the threshold will adapt to changing conditions if needed.
Table 2.1: Complexity of circuit blocks in SSNOC prototype IC.

<table>
<thead>
<tr>
<th>block</th>
<th>cells</th>
<th>area (mm(^2))</th>
</tr>
</thead>
<tbody>
<tr>
<td>sensors</td>
<td>25472</td>
<td>0.844</td>
</tr>
<tr>
<td>fusion block</td>
<td>3201</td>
<td>0.109</td>
</tr>
<tr>
<td>control</td>
<td>169</td>
<td>0.0055</td>
</tr>
<tr>
<td>thresholding block</td>
<td>249</td>
<td>0.0098</td>
</tr>
</tbody>
</table>

Figure 2.11: The 256-tap PN code acquisition filter chip in a 180 nm CMOS process: (a) layout, and (b) microphotograph.

2.3.4 Chip layout

The prototype chip layout in a 180 nm CMOS process contains a total of 48440 cells and a total cell area of 1.871 mm\(^2\). A detailed breakdown of the complexity of each block is given in Table 2.1. The core area is approximately 2 mm \(\times\) 2 mm, and 10 IO pins are placed on each side. Including the pad frame, the final layout size is 2.7 mm \(\times\) 2.7 mm. Spacing from the power rings to the pad frame is approximately 15 \(\mu\)m. The chip layout and microphotograph are shown in Fig. 2.11. The chip is packaged in a 68-pin leadless chip carrier package.
2.4 Measurement Results

The chip was tested with the Agilent 16900A logic analysis system, at a frequency of $f_{clk} = 50$ MHz. At this frequency, the point of first failure (PoFF) voltage is $0.95$ V. The test vector was generated by embedding a length 256 PN code in a sequence of Gaussian noise with a SNR of $-12$ dB. Test vectors with length $10^6$ containing $10^3$ detections were employed. A total of five chips were tested at supply voltages from $0.95$ V down to $0.6$ V.

2.4.1 Error characterization

First, the statistics of VOS errors is collected and analyzed. Errors can be observed at operation below the critical voltage of $V_{dd-crit} = 0.95$ V and the error rate increases with further scaling. Error characterization is performed by first subtracting the output at nominal voltage $y_0[n]$ from the output at overscaled voltages. Next, the errors are histogrammed and normalized to generate the error probability mass function (PMF). The error PMFs are grouped into four operating regions based on the supply voltage.

2.4.1.1 Region R1

Region R1 is at $V_{dd} = 0.85$ V, near the PoFF but well below it by 10%, and hence the error rate is 0.074 (approximately 100× higher than in [28]) and consists of only small (single bit) valued errors. Figure 2.12 (a), (b) shows the errors in this region.

2.4.1.2 Region R2

Region R2 is at $V_{dd} = 0.76$ V (20% below the PoFF) where multi-bit errors begin to appear and the error rate is 0.54. This region has mild VOS applied to the circuit and the majority of the errors are still small in magnitude as shown in Fig. 2.13 (a), (b).

40
Figure 2.12: Error PMF of region R1: (a) $V_{dd} = 0.88$ V, and (b) $V_{dd} = 0.85$ V. All errors are one bit errors with powers-of-two magnitude.

Figure 2.13: Error PMF of region R2: (a) $V_{dd} = 0.78$ V, and (b) $V_{dd} = 0.76$ V. Multi-bit errors are observed, but most errors are still small in magnitude.
2.4.1.3 Region R3

Region R3 is at $V_{dd} = 0.66$ V (30% below PoFF) with an error rate of 0.95. As seen in Fig. 2.14 (a), (b), this region shows Gaussian-like error statistics ($\epsilon_i$) overlaid with large magnitude errors ($\eta_i$), which are still correctable with median or mean fusion. Region R3 represents a high VOS region with an exceptionally high error rate. The spiky characteristic of the PMF results in the good performance of median fusion. Median is a function that eliminates outliers, and the spikes that are observed can be interpreted as outliers.

2.4.1.4 Region R4

Region R4 ($V_{dd} = 0.6$ V or 37% below PoFF) is where the error rate is 0.97 and the system breaks down. In this region, the PN acquisition system fails to produce any meaningful computation. The errors have no more spiky characteristics, and look like a narrow Gaussian distribution, thus violating the $\epsilon$-contaminated model. Also, large magnitude errors, up to the MSB, are present as shown in Fig. 2.15 (a), (b).
Figure 2.15: Error PMF of region R4: (a) $V_{dd} = 0.63$ V, and (b) $V_{dd} = 0.60$ V.

2.4.2 Test results

Figure 2.16 plots $P_{det}$ and $p_\eta$ vs. $V_{dd}$ for a fixed false alarm rate of 10%. It can be seen that a near constant $P_{det} \geq 90\%$ is achieved for $V_{dd} \geq 0.69$ V with $p_\eta \leq 0.912$. This voltage is 27% below the PoFF voltage of $V_{dd} = 0.95$ V, indicating significant robustness to voltage variations. These results are consistent with the simulation results in [52] that show that a 25% reduction in $V_{dd}$ has minimal performance degradation.

Figure 2.17 shows the relation between energy and $V_{dd}$ along with $p_\eta$. The measurements include the sensor and fusion block overhead. Compared to $V_{dd} = 0.95$ V (PoFF), energy savings between $2.4 \times$ and $5.8 \times$, for an average of $3.86 \times$, and error tolerance between $2146 \times$ and $2281 \times$, for an average of $2225 \times$, can be achieved at $V_{dd}$ ranging between $0.69$ V and $0.70$ V without any loss in system-level performance (probability of detection $P_{det}$) in the presence of very high probability of error $p_\eta \leq 0.912$. Compared to simulation results in Fig. 1.6, measurements in Fig. 2.17 indicate that expected error tolerance and two-thirds of the potential energy savings have been realized. This represents a $2.52 \times$ greater energy savings over [28] with a $2225 \times$ higher error rate tolerance.

Table 2.2 compares the results of our work with previous published work. The first three entries are compared against generic error resilient designs while the last three are compared against conventional PN code acquisition chips. It can be seen that the SEC based design achieves significantly better performance by utilizing statistical information at the system
Figure 2.16: Detection probability $P_{\text{det}}$ and sensor probability of error $p_{\eta}$ vs. supply voltage $V_{dd}$. 
level. The high error resiliency has been successfully traded off with energy to achieve significant savings compared to other error resilient designs. To compare our PN code filter with existing designs, a figure of merit (FOM) that normalizes performance based on technology, resolution, and sample rate has been employed. The FOM is defined as (normalized power) / (taps × precision × sample rate) with units of (pJ/bit). Normalized power for analog circuits is obtained by normalizing with $V_{dd}$ and the feature size [73], while for digital, it is normalized by $V_{dd}^2$ and the feature size [74]. Resolution is the average of

<table>
<thead>
<tr>
<th>tech.</th>
<th>$V_{dd}$</th>
<th>$p_\eta$</th>
<th>energy savings</th>
<th>figure of merit $^\dagger$</th>
</tr>
</thead>
<tbody>
<tr>
<td>180 nm</td>
<td>1.2 – 1.8 V</td>
<td>0.1%</td>
<td>14-17%</td>
<td>N/A</td>
</tr>
<tr>
<td>130 nm</td>
<td>0.8 – 1.2 V</td>
<td>0.04%</td>
<td>33-35%</td>
<td>N/A</td>
</tr>
<tr>
<td>45 nm</td>
<td>0.9 – 1.0 V</td>
<td>N/A</td>
<td>22%</td>
<td>N/A</td>
</tr>
<tr>
<td>350 nm</td>
<td>1.8 V</td>
<td>0%</td>
<td>N/A</td>
<td>0.319</td>
</tr>
<tr>
<td>350 nm</td>
<td>2 V</td>
<td>0%</td>
<td>N/A</td>
<td>0.09699</td>
</tr>
<tr>
<td>1.2 μm</td>
<td>5 V</td>
<td>0%</td>
<td>N/A</td>
<td>0.713</td>
</tr>
<tr>
<td>180 nm</td>
<td>0.69 – 0.95 V</td>
<td>88.99%</td>
<td>71.87%</td>
<td>0.037</td>
</tr>
</tbody>
</table>

$^\dagger$ Figure of merit is defined as power/(resolution × sample rate) (nJ/bit), where resolution is the average of input (max of data and coefficient bits) and output bit precision.
input (max of data and coefficient bits) and output bit precision. As [69, 71] is an analog implementation, it is not normalized by precision, and IO power has been included for [70]. The SEC based PN code acquisition filter, when compared against conventional PN code acquisition filters, shows a minimum of $2.5 \times$ improvement in the FOM.

### 2.4.3 Impact of VOS on activity factor

The energy savings measured in Section 2.4.2 are larger than the values predicted by voltage scaling via $E = C_L V_{dd}^2$, where $C_L$ is the effective load capacitance. In this section, we analyze a circuit under VOS and provide simulation results to show the source of the additional energy savings.

In VOS design, the operation frequency is kept constant at $f_{CLK}$, while the supply voltage $V_{dd}$ is scaled down below $V_{dd,crit}$. The relationship between energy $E$ and power $P$ is

$$E = Pt = \alpha_{0 \rightarrow 1}(V_{dd}) f_{CLK} C_L V_{dd}^2 t$$

where $\alpha_{0 \rightarrow 1}$ is the activity factor, which is the average probability of $0 \rightarrow 1$ transitions, and $t$ is the total time required to complete a given computation. In VOS design, $\alpha_{0 \rightarrow 1}$ depends on $V_{dd}$, while $f_{CLK}$, $C_L$, and $t$ are all independent of $V_{dd}$. Thus the energy is proportional to the activity factor $\alpha_{0 \rightarrow 1}$ and quadratic to the supply voltage $V_{dd}^2$. $V_{dd}$ is easily controllable and measurable. For the PN acquisition prototype chip, $V_{dd}$ is scaled from 0.95 V down to 0.69 V, providing up to 47.2% energy savings, a little more than half of the total 82.8% energy savings achieved. The remaining portion can be attributed to the reduction in activity factor $\alpha_{0 \rightarrow 1}$ due to VOS. However, the effect of VOS on $\alpha_{0 \rightarrow 1}$ is not easy to measure. To obtain some insight on how VOS affects $\alpha_{0 \rightarrow 1}$, we have performed HSPICE simulations of a sensor (Fig. 2.5) under various voltages and measured the average activity factor. Figure 2.18(a) plots the number of glitches observed for various bit position vs. $V_{dd}$. It can be seen that the number of glitches reduces as the voltage is lowered. This can be explained by noting that under severe VOS, the propagation delay will most certainly be longer than the clock period, and the voltage at the final output gate will see less transitions. Furthermore,
glitches that are generated by input timing mismatches will not be able to propagate to the end of the logic chain, further reducing the activity factor. The resulting $\alpha_{0\rightarrow1}$ is plotted in Fig. 2.18(b). In our simulations, the activity factor has shown a reduction of at most 39.2%. The combination of activity reduction by 40% and VOS from 0.95 V to 0.69 V results in a $(0.95/0.69)^2/0.6 = 3.16 \times$ savings, which is close to the measured average energy savings of $3.86 \times$.

In conclusion, VOS has been shown to also affect the activity factor significantly, and result in greater energy benefits than predicted by voltage scaling alone. However, for further generalization of this relationship between $\alpha_{0\rightarrow1}$ and $V_{dd}$, a rigorous analysis and simulation will need to be performed on various circuit architectures.

### 2.5 Summary

This chapter has shown that SEC is an effective technique for achieving energy efficiency via error resiliency. SEC is an application of statistical estimation and detection techniques, which form the foundation of communication systems, to the problem of robust and energy-efficient information processing. In particular, stochastic networked computation, which
is based on Huber’s robust estimation theory, is particularly well suited for the PN code acquisition filter. In this dissertation, we have approximated the one-step Huber algorithm with the median block. However, there exists further possibility of reducing the fusion block complexity with no loss in performance. Given the PN code acquisition application, the thresholding stage and the fusion stage are interchangeable. By doing so, the median fusion block reduces to $N$ single bit adders followed by a thresholding operation with a threshold at $\frac{N}{2}$, where $N$ is the number of sensors.

The application of SEC requires a good understanding of system-level requirements, which need to be integrated with architecture and circuit design issues. A number of energy-efficient kernels (stochastic kernels) for domain- or application-specific algorithms can be developed based on SEC. Such accelerator cores can comprise a future system-on-a-chip platform on nanoscale CMOSs, where energy efficiency and robustness are jointly achieved through a pervasive application of SEC.
Chapter 3

Statistical Error Compensation for Low-Density Parity Check Codes

Probabilistic graphical models (PGMs) are commonly used to perform inference in machine learning algorithms used in applications such as classification and parameter estimation. PGMs are usually learned using empirical (and potentially incomplete and noisy) sets of data. Since these models are learned from noisy or incomplete data sets, they already have slight perturbations from the exact underlying model. Thus, PGMs are inherently resilient to some degree of computational errors, and tend to have superior and highly accurate classification performance. Belief propagation (BP) is a message-passing based algorithm widely used for inference on these probabilistic graphical models.

Low-density parity check (LDPC) codes are primarily decoded using BP. It is widely adopted in communication standards, such as 802.11 Wi-Fi, DVB-S2 satellite transmission of digital television, and is considered for many 4G systems including WiMax (IEEE Std 802.16e) [75–77] due to its excellent error correction performance. However, the decoding complexity of LDPC codes is quite large and low-power LDPC decoders are required to satisfy the power constraints of wireless handsets. Analog decoder architectures have been proposed for short-length codes [78]; yet, scaling the code length to more than 250 will be challenging due to device mismatch and buffering requirements. Digital low-power LDPC decoder architectures mostly focus on reducing the decoding complexity through early termination or approximation [79].

Previous work on error-resilient LDPC decoders has protected the sign bit of decoding messages (see Section 3.2.1) or employed triple modular redundancy (TMR) [80]. However, the error resiliency provided by sign bit protection (SBP) alone has limitations at high error rates (more than 3% errors per clock cycle), and TMR has very high overhead.

Recently, studies on erroneous LDPC decoders have been performed. Varshney [81] shows
that the performance of a Gallager A decoder subject to random bit flips cannot achieve arbitrarily small probability of error, contrary to the results of Shannon’s channel capacity theorem [8]. Yazdi et. al. also show similar results for the Gallager B decoder [82]. However, these decoders are hard decision based decoders that do not possess the inherent resiliency of BP. For a BP based decoder, Varshney [81] has shown that for an additive white Gaussian noise (AWGN) channel, if the decoding errors are bounded, arbitrarily small probability of error is achievable. Similarly, according to this principle, quantized versions of BP show little degradation in performance [83–85]. Such works show that BP based LDPC decoders possess inherent resiliency, and are a good candidate for SEC. A general error analysis on BP based message passing algorithms is performed in [86] for problems on distributed inference.

In this chapter, we apply SEC to present an energy-efficient BP based binary LDPC decoder. Our focus is on timing errors, as the source of reliability for deep-scaled CMOSs, including PVT variation, all manifest as output delay variation. One characteristic of timing errors in DSP and communication systems is that the error magnitude is large, as most systems are designed to compute LSB first. SEC exploits the system-level inherent resiliency by converting large-magnitude timing errors into small tolerable errors. We evaluate the energy efficiency vs. robustness trade-offs of the SEC based LDPC decoder in a 45 nm process technology. Once again, energy efficiency is obtained by trading reliability for energy savings via voltage overscaling (VOS), which deliberately induces timing errors. Figure 3.1 shows the energy vs. $V_{dd}$ for a variable node, a block used in decoding LDPC codes (see Fig. 3.4(a) on page 56), for a conventional error-free design, and a VOS design that targets an error rate of 70%. If the VOS-induced errors are fully compensated for, significant energy savings, up to 70%, can be achieved.

The chapter is organized as follows. Section 3.1 provides background information on LDPC codes. Section 3.2 describes the LDPC decoder architecture in detail. The simulation setup including energy and error modeling is presented in Section 3.3. Simulation results are given in Section 3.4, with Section 3.5 summarizing the chapter.
Figure 3.1: Supply voltage vs. energy and delay of an LDPC variable node (Fig. 3.4(a)) in a commercial 45 nm process. By voltage overscaling up to an error rate of 70%, the same performance can be achieved with 70% less energy.

3.1 Low-Density Parity Check Codes

Low-density parity check (LDPC) codes, first introduced by Gallager [87], are error-correcting codes used in various communication systems. The code is very powerful in the sense that it can achieve the Shannon capacity for additive white Gaussian noise (AWGN) channels. More interestingly, the decoding of LDPC codes is based on a message-passing algorithm implemented over a factor graph. LDPC codes are linear block codes based on a sparse parity check matrix $H$. Let $H$ be a binary $r \times n$ matrix. In coding theory, any vector $c$ of length $n$ is a valid codeword if $Hc^T = 0$. This parity check matrix gives rise to the bipartite factor graph, where there are $r$ check nodes and $n$ variable nodes. The graph is connected in a way such that if the entry $(i, j)$ of $H$ is 1 then the $i^{th}$ check node is connected to the $j^{th}$ variable node. An example parity check matrix and its factor graph are depicted in Fig. 3.2. Each column of $H$ represent the code bits (variables nodes), while each row represent the parity check constraint (check nodes).

Though decoding of an LDPC code is NP-complete, BP based iterative decoding algo-
Figure 3.2: Example LDPC code: (a) parity check matrix, and (b) its bipartite graph.

The following notation will be used: $P_i$ is the probability of bit $c_i = 1$, given the channel output $y_i$, i.e., $Pr(c_i = 1|y_i)$. Denote $q_{ij}(\cdot)$ as the message sent from variable node $v_i$ to check node $c_j$, and $r_{ji}(\cdot)$ as the message sent from check node $c_j$ to variable node $v_i$. The variable-to-check messages, $q_{ij}(0)$ and $q_{ij}(1)$, are proportional to the probabilities $Pr(v_i = 0)$ and $Pr(v_i = 1)$, respectively. The check-to-variable messages, $r_{ji}(0)$ and $r_{ji}(1)$, are the probabilities of the $j^{th}$ check being satisfied when $v_i = 0$ and $v_i = 1$, respectively. Since $v_i \in \{0, 1\}$, $q_{ij}(0) = 1 - q_{ij}(1)$. The check nodes compute, on average, the probability that
an even number of 1’s are observed through the following equations:

\[
\begin{align*}
    r_{ji}(0) &= \frac{1}{2} + \frac{1}{2} \prod_{i' \in V_j \setminus i} (1 - 2q_{i'j}(1)) \\
    r_{ji}(1) &= 1 - r_{ji}(0)
\end{align*}
\]  

(3.1)

where \(V_j \setminus i\) denotes all the nodes connected to variable node \(v_j\) excluding check node \(c_i\).

This follows directly from Gallager [87], where the probability of a sequence having an even number of 1’s is computed as described in Theorem 3.1.

**Theorem 3.1.** Given a sequence of \(M\) independent binary digits \(\{a_i\}_{i=1}^M\), with the probability of each digit being 1 denoted as \(p_i = Pr(a_i = 1)\), the probability that the sequence contains an even number of 1’s is \(P_M = \frac{1}{2} + \frac{1}{2} \prod_{i=1}^M (1 - 2p_i)\).

**Proof.** The proof follows by induction. Assume this is true for \(M = k\), i.e., the probability that a sequence with \(M = k\) independent binary digits has an even number of 1’s is \(P_k = \frac{1}{2} + \frac{1}{2} \prod_{i=1}^k (1 - 2p_i)\). Then, for \(M = k + 1\),

\[
\begin{align*}
P_{k+1} &= P_k(1 - p_{k+1}) + (1 - P_k)p_{k+1} \\
      &= \left\{ \frac{1}{2} + \frac{1}{2} \prod_{i=1}^k (1 - 2p_i) \right\} (1 - p_{k+1}) + \left\{ \frac{1}{2} - \frac{1}{2} \prod_{i=1}^k (1 - 2p_i) \right\} p_{k+1} \\
      &= \frac{1}{2} + \frac{1}{2} \prod_{i=1}^k (1 - 2p_i) - \frac{1}{2} \prod_{i=1}^k (1 - 2p_i)p_{k+1} - \frac{1}{2} \prod_{i=1}^k (1 - 2p_i)p_{k+1} \\
      &= \frac{1}{2} + \frac{1}{2} \prod_{i=1}^{k+1} (1 - 2p_i)
\end{align*}
\]

Thus, this is also true for \(M = k + 1\).

For \(M = 1\),

\[
P_1 = Pr(a_1 = 0) = 1 - p_0 = \frac{1}{2} + \frac{1}{2} \prod_{i=1}^1 (1 - 2p_i)
\]

Thus, this is also true for \(M = 1\). By induction, this is true for all \(M \geq 1\), completing the
proof.

The variable node then computes its message by

$$q_{ij}(0) = (1 - P_i) \prod_{j' \in C_i \setminus j} r_{j'i}(0) \quad (3.3)$$

$$q_{ij}(1) = P_i \prod_{j' \in C_i \setminus j} r_{j'i}(1) \quad (3.4)$$

This is just multiplying all the beliefs from the check node of being 0 to obtain the final belief of the variable being 0 and likewise for 1. In a more practical implementation of LDPC, instead of tracking two beliefs, only one message is passed, the log likelihood ratio (LLR), which is the log of the ratio of the two messages, $\log \frac{q_{ij}(1)}{q_{ij}(0)}$ and $\log \frac{r_{ji}(1)}{r_{ji}(0)}$. Taking the log ratio of (3.2) to (3.1) and (3.4) to (3.3) we get

$$m_{ij} = \log \frac{q_{ij}(1)}{q_{ij}(0)} = \log \left( \frac{P_i}{1 - P_i} \right) + \sum_{j' \in C_i \setminus j} n_{j'i} \quad (3.5)$$

$$n_{ji} = \log \frac{r_{ji}(1)}{r_{ji}(0)} = \log \frac{1 + \prod_{i' \in V_j \setminus i} \tanh(m_{i'j}/2)}{1 - \prod_{i' \in V_j \setminus i} \tanh(m_{i'j}/2)} \quad (3.6)$$

$$= \psi^{-1} \left( \sum_{i' \in V_j \setminus i} \psi(m_{i'j}) \right) \quad (3.7)$$

where $m_{ij}$ and $n_{ij}$ are the LLR of the variable to check node and check node to variable node messages, respectively, and $\psi(x) = -\log \tanh(|x|/2)$. Using the max-log approximation [88], i.e., $\log \sum_k z_k \approx \max_k \log z_k$, the check node to variable node message (3.7) can be further approximated as

$$n_{ji} \approx ( \min_{i' \in V_j \setminus i} |m_{i'j}| \prod_{i' \in V_j \setminus i} \text{sgn}(m_{i'j}) \quad (3.8)$$
3.2 LDPC Decoder Architecture

High-throughput LDPC decoders require parallel computation of messages. We implement a fully parallel LDPC decoder, with the computation of variable and check nodes multiplexed in time, i.e., at one clock cycle variable nodes operate in parallel, and at the next clock cycle check nodes operate in parallel. The high-level architecture along with node architectures is shown in Figs. 3.3 and 3.4.

The variable node implements (3.5). It takes the prior information given by the received signal $y$, and the messages incoming from the check nodes, and sums them up. A hard decision block extracts the sign bit of the computed messages and outputs them as the decoded bits $\hat{b}$. The check node implements (3.7). The $\psi(\cdot)$ and $\psi^{-1}(\cdot)$ functions are implemented as a lookup table via piecewise linear approximation.

3.2.1 Error resiliency

The original LDPC decoder is inherently robust to errors due to its iterative message-passing decoding algorithm. However, errors in the sign bit have been shown to be detrimental. In our work, we employ SBP as our baseline LDPC decoder. The sign bit is computed separately via the max-log approximation in (3.8). The critical path for the sign bit computation is significantly shorter than that of the magnitude computation, and the sign bit will experience errors only at significant VOS. In the proposed error-resilient LDPC decoder, SEC is added to the variable node and check node via ANT. The estimator is a reduced precision replica of the original variable and check node. For the 8-bit precision main block, three estimators...
are employed with 2, 3, and 4 bits of precision, respectively.

### 3.3 Energy, Delay, and Error Modeling

#### 3.3.1 Energy model

We estimate the energy consumption of the LDPC decoder by its constituent blocks. The total energy of the LDPC decoder $E_{LDPC}$ is given by

$$
E_{LDPC} = N_{var}E_{var} + N_{check}E_{check} + N_{wire}E_{wire}
$$

(3.9)

where $N_{var}$, $N_{check}$, and $N_{wire}$ are the total number of variable nodes, check nodes, and interconnect wires, respectively, and $E_{var}$, $E_{check}$, and $E_{wire}$ are the energy consumptions of a single variable node, check node, and interconnect wire, respectively. To obtain $E_{var}$ and $E_{check}$, a single variable node and check node, as shown in Fig. 3.4, is implemented in Verilog...
Figure 3.5: Energy consumption and delay curves obtained through circuit simulation of the variable and check node architectures in Fig. 3.4, synthesized in a commercial 45 nm process.

and synthesized using a commercial 45 nm standard cell library. The synthesized netlist is then used to extract a SPICE netlist, which is used for circuit simulations. The energy vs. $V_{dd}$ plot and the delay vs. $V_{dd}$ are shown in Fig. 3.5 for both variable node and check node.

A three-wire distributed RC network is used for the interconnect model as shown in Fig. 3.6(a). Only coupling between adjacent wires is considered, and the energy consumed by the middle wire is averaged over all possible transitions. The wire was assumed to be routed on metal 4 with a length of 200 $\mu$m. The values for $R$, $C_C$, and $C_G$ were obtained from the design manual of a commercial 45 nm process. Figure 3.6(b) shows the energy consumption obtained through circuit simulations. As the wire delays were significantly shorter than the bit node and check node delays, the interconnect is assumed to be error-free. The energy values obtained are comparable to those obtained from the bus energy consumption model in [89].
Figure 3.6: Interconnect energy for a 200 µm wire: (a) distributed RC model, and (b) average energy vs. supply voltage curve.
Figure 3.7: Methodology for simulating the LDPC decoder architecture under VOS. Energy models can be obtained as well.

3.3.2 Error modeling

To simulate input-dependent timing errors, gate-level simulations using an HDL simulator are performed. First the gate delay is characterized with respect to supply voltage for basic gates such as a full adder and XOR, using a circuit-level simulator as in Section 3.3.1. Then a structural HDL implementation of the LDPC decoder is simulated via an HDL simulator using the pre-characterized delay values. By choosing the delay values that correspond to various supply voltages, HDL simulation is effectively run at different voltages, and for $V_{dd} < V_{dd,crit}$, errors can be observed in the outputs. The complete simulation methodology is summarized in Fig. 3.7.

3.4 Simulation Results

Simulations for a (800, 400) LDPC code was performed at various $V_{dd}$ and SNRs.
Figure 3.8: BER vs. SNR plot of a (800, 400) LDPC code decoded with five iterations at $p_\eta = 0.2$.

### 3.4.1 BER performance

The BER performance of the LDPC code for an error rate of $p_\eta = 0.2$ is shown in Fig. 3.8. It can be seen that SBP schemes only work until an SNR of 1.8 dB and fails completely at SNR > 3 dB. ANT schemes are significantly more robust with with ANT-2b (ANT with a 2 b estimator precision) breaking down at SNR > 7 dB, ANT-3b breaking down at SNR > 8 dB (not shown in figure), and ANT-4b retaining performance close to error-free LDPC for SNR as high as 10 dB. The performance breakdown at high SNR can be attributed to the error floor. The erroneous outputs created by ANT contribute to raising the error floor, and the decoder hits this wall at a lower SNR than an error-free decoder.

### 3.4.2 Robustness

Figure 3.9 shows the change in BER as $V_{dd}$ is reduced. It can be seen that even with no error correction, the LDPC decoder maintains performance up to $p_\eta = 2 \times 10^{-3}$. However, its performance degrades rapidly when subject to higher error rates. SBP is able to tolerate...
Figure 3.9: BER vs. $p_{\eta}$ at $SNR = 5$ dB.

Table 3.1: HW error rate that can be tolerated by SEC based LDPC decoder at a given BER threshold.

<table>
<thead>
<tr>
<th>BER threshold</th>
<th>2×</th>
<th>3×</th>
<th>4×</th>
<th>5×</th>
</tr>
</thead>
<tbody>
<tr>
<td>no correction</td>
<td>0.0060</td>
<td>0.0097</td>
<td>0.0134</td>
<td>0.0171</td>
</tr>
<tr>
<td>SBP</td>
<td>0.0087</td>
<td>0.0114</td>
<td>0.0140</td>
<td>0.0167</td>
</tr>
<tr>
<td>ANT-4b</td>
<td>0.2873</td>
<td>0.3503</td>
<td>0.4134</td>
<td>0.4839</td>
</tr>
<tr>
<td>ANT-2b</td>
<td>0.0355</td>
<td>0.0513</td>
<td>0.0672</td>
<td>0.0830</td>
</tr>
<tr>
<td>(ANT-4b/SBP)</td>
<td>4.08</td>
<td>30.73</td>
<td>29.53</td>
<td>28.98</td>
</tr>
</tbody>
</table>

higher error rates of up to $p_{\eta} = 7 \times 10^{-3}$, but ANT is still much more powerful, achieving acceptable performance for up to $p_{\eta} = 0.2$. As a slight degradation in BER is tolerable in most systems (as compared to a magnitude change), we have chosen several BER thresholds, 2 to 5 times the BER of the error-free case, and found the tolerable error rate for each scheme. The results are summarized in Table 3.1. It can be seen that ANT shows up to $30.7\times$ more robustness than SBP.
Figure 3.10: Energy vs. BER plot of a (800, 400) LDPC code decoded with five iterations at $SNR = 5$ dB.

3.4.3 Energy savings

Energy savings of various schemes are compared at the same BER performance. ANT schemes are able to achieve the same BER performance at a significantly lower $V_{dd}$ and thus result in energy savings. Figure 3.10 shows that ANT-4b can achieve up to 45.7% energy savings compared to the error-free conventional LDPC decoder, and up to 33.2% energy savings compared to the erroneous conventional LDPC decoder. This is in addition to the $30\times$ enhanced robustness.

3.5 Summary

In this chapter, we applied SEC to LDPC decoders and achieved up to $30\times$ enhancement in robustness and 45.7% energy savings compared to conventional LDPC decoders. Compared to conventional SBP based design, significant enhancement in reliability is achieved. Future work includes analysis of the effect of HW errors on decoding performance and the propaga-
tion of errors along the decoding graph. Based on this understanding, the overhead of SEC can be reduced to achieve greater energy savings. Possible methods are to apply SEC to a partial set of nodes that are critical, while simpler error resiliency techniques, such as SBP, are applied to the remaining nodes.
Chapter 4

Statistical Error Compensation for Stereo Image Matching

Machine learning based inference has recently gained importance as a key kernel in processing massive data in signal processing systems, including computer vision and speech recognition. Such applications contain an inference kernel that is inherently resilient to small magnitude errors [90,91]. By combining the statistical performance metric of machine learning applications and statistical nature of circuit non-idealities, statistical error compensation (SEC) techniques [24] can achieve significant enhancement in error resiliency [90,91]. By trading off the increased robustness with energy reduction, significant energy savings can be achieved as well. In Chapter 3, algorithmic noise tolerance (ANT) has been applied to a message-passing based low-density parity check (LDPC) decoder and shown to achieve 45.7% energy savings while maintaining less than 4.7 dB degradation in bit error rate (BER) at a HW error rate, $p_\eta$ (percentage of clock cycles in which an erroneous output exists), of 30%. In this chapter, an architecture implementing a sequential tree-reweighted (TRW-S) Markov random field (MRF) message-passing based stereo matching [92] will be shown to achieve similar results, with an energy savings of 41%, a bad-pixel ratio (BPR) degradation of less than 3.5%, at a HW error rate of 21.3%. This indicates that message-passing or belief propagation based algorithms are inherently tolerant to estimation errors, $p_c$, within the ANT estimator, and by properly exploiting this tolerance, ANT can provide significant benefits.

We first study the performance of the TRW-S architecture when subject to nanometer imperfections to design an energy-efficient hardware implementation. Our goal is to analyze the inherent error resiliency of message-passing based inference algorithms and to enhance its error resiliency via application of SEC. The enhanced error resiliency can then be traded-off with energy efficiency by exploiting it to tolerate hardware errors generated by relaxed hardware implementation or relaxed operating conditions, resulting in energy savings. In
this chapter, the work of Chapter 3 is extended to a more general case of MRF inference, i.e., TRW-S based stereo matching, with a more complex graph and a larger hypothesis space. The TRW-S based stereo matching hardware architecture (TRWS_HW) from [92] is taken as a test case, and its error resilience, as well as the impact of ANT techniques, is explored. Results show that the ANT based hardware can tolerate an error rate of 21.3%, with performance degradation of only 3.5% at an overhead of 44.7%, compared to an error-free hardware with an energy savings of 41%, but such arithmetic-level error compensation incurred significant overhead.

To reduce the overhead of arithmetic-level error compensation, ANT is further applied at the iteration and system level for the TRW-S stereo matching system [93]. Iteration-level compensation removes all correction overhead except at the final latch of each functional block within the message-passing computation. At the system level, a CPU was used to aid in the estimation and correction of the final stereo matching output (i.e., depth map). This approach may seem similar to the existing error-resilient system architecture (ERSA) [32] paradigm, but is significantly different because an application-specific accelerator is used and permitted to make hardware errors, which are compensated via statistical estimation and detection techniques. A hybrid approach combining both iteration-level and system-level compensation shows promising enhancement in error resiliency with reduced overhead. Results show that, compared to arithmetic-level, system-level compensation reduces overhead by more than 50%, while maintaining stereo matching performance with only 2.5% degradation. These results are verified via FPGA emulation with timing errors induced within the message-passing unit via relaxed synthesis.

The remainder of the chapter is organized as follows. Section 4.1 provides background information on the TRW-S stereo matching algorithm. Section 4.2 describes the error-resilience characteristic of the message-passing hardware in detail. The error compensation methodology is discussed in Section 4.3. The simulation methodology and emulation setup including the generation HW errors of TRW-S is discussed in Section 4.4. Section 4.5 shows the results, while Section 4.6 summarizes the chapter.
4.1 TRW-S Message-Passing Based Stereo Matching

Stereo matching infers depth information from two horizontally shifted images. It can be restated as a maximum a posteriori (MAP) problem, where each pixel is given a label, \( l_s \in \{0, \ldots, D_{\text{max}}\} \), that corresponds to a discrete depth level and the goal is to find the most probable label assignments for all the pixels of an image. The depth level represents the proximity of the pixel to the observation point with 0 being the closest and \( D_{\text{max}} \) the farthest. Typical values of \( D_{\text{max}} \) are 15 and 63. This MAP problem can be formulated in terms of the cost functions defined on an undirected grid graph (i.e., a grid MRF) with nodes \((V)\) and edges \((E)\) as follows [94,95]:

\[
\min_l E(l) = \min_l \left\{ \sum_{s \in V} d_s(l_s) + \sum_{(s,t) \in E} V_{st}(l_s, l_t) \right\} \quad (4.1)
\]

Here, a unary cost function \( d_s(l_s) \) represents the likelihood of a depth label \( l_s \) being assigned to a node \( s \), and a pairwise (smoothness) cost function \( V_{st}(l_s, l_t) \) models, as prior preference, the closeness (smoothness) among neighboring nodes. Note that \( V_{st}(l_s, l_t) \) is a truncated function to allow large difference in labels between adjacent nodes. \( E(l) \) is referred to as energy, and thus (4.1) is called an energy minimization problem; the quality of a label assignment is higher if the obtained energy is lower, and thus \( E(l) \) is used as a metric to gauge the quality of the results obtained by inference. In general, this energy minimization problem is intractable since the complexity of the problem grows exponentially as the size of the graph \( G(V, E) \) is increased.

Message-passing algorithms have been shown to efficiently solve (4.1) by selecting the best label for each node based on local information from its neighbors (i.e., messages). Messages between nodes of an MRF are updated repeatedly (iterations) until they converge to a fixed value. The final label is then assigned based on the converged messages. In general, message-passing algorithms such as belief propagation (BP) [94,95] do not guarantee convergence if a loop exists within the graph. Tree-reweighted message passing (TRW-S) performs tree based sequential message passing which guarantees subsequent convergence to the lower local minimum energy, and shows superior energy minimization performance compared to
BP in practice [96–98].

In case of a grid-MRF, message passing of TRW-S from node $s$ to $t$ can be performed in two steps, a unary and pairwise computation as follows [97]:

\[
H_{st}(l_s) = \frac{1}{2} \left\{ d_s(l_s) + \sum_{u \in Nb(s)} M_{us}(l_s) \right\} - M_{ts}(l_s)
\]  

(4.2)

\[
M_{st}(l_t) = \min_{l_s} \{ H_{st}(l_s) + V_{st}(l_s, l_t) \}
\]  

(4.3)

where $H_{st}(l_s)$ represents the updated belief of node $s$, $M_{st}(l_s)$ denotes the message from node $s$ to $t$, $Nb(s)$ is the set of all neighbors of node $s$, and $\sum_{u \in Nb(s)} M_{us}(l_s)$ indicates accumulation of messages from neighbors of node $s$. The unary computation (4.2) and the pairwise computation (4.3) are called reparameterization (REPARAM) and message update (MSG_UPD), respectively. In this chapter, we use the hardware architecture of [92], where this two-step message passing of TRW-S is pipelined and executed in a streaming manner (see Fig. 4.1). FIFOs are used to handle streaming data access to the main memory in order to feed the pipelined message-passing units without many stalls. The REPARAM unit is a simple hardware realization of (4.2), while for MSG_UPD, a parallel message construction technique [99] is exploited to reduce complexity of the pairwise computation in (4.3), as shown in Fig. 4.1(c). In addition, to avoid overflow, MSG_UPD includes a normalization step, which rescales the value of messages so that the minimum message is always zero.

4.2 Error-Resilient TRW-S via ANT

It is well known in the literature (e.g., [90]) that iterative message-passing algorithms such as TRW-S are intrinsically robust to errors in computation. In this section, we analyze the error propagation of TRW-S as a test case to understand the basis of the inherent error robustness. Based on our findings, ANT is applied to further enhance the error robustness.
Figure 4.1: Architecture of streaming TRW-S hardware (TRWS_HW): (a) block diagram, (b) reparameterize unit, and (c) message update unit.
4.2.1 Error analysis of TRW-S

The error propagation of TRW-S is analyzed starting from (4.2) and (4.3). For simplicity, we assume binary labels \{0, 1\} for all the nodes and the Potts model [96] (with penalty of \(C\)) for the pairwise cost, i.e., the cost is 0 if neighboring labels are equal, while the cost is \(C\), if they differ. The message update (4.3) for \(l_s \in \{0, 1\}\) can then be represented as follows:

\[
\begin{align*}
M_{st}(l_s = 0) &= \min \{H_{st}(0), H_{st}(1) + C\} \\
M_{st}(l_s = 1) &= \min \{H_{st}(0) + C, H_{st}(1)\}
\end{align*}
\]

(4.4) \hspace{1cm} (4.5)

Note that other message-passing algorithms with a truncated smoothness cost (e.g., BP in [94,95]) can also be represented this way.

To see the effect of arithmetic error on the result of the final message, we assume that arithmetic errors occur in add or subtract (AS) and compare and select (CS) operations. An erroneous message for label 0, \(\tilde{M}_{st}(0)\), can be represented as follows:

\[
\tilde{M}_{st}(0) = \min \{H_{st}(0) + \eta_{AS}(0), H_{st}(1) + \eta_{AS}(1) + C\} + \eta_{CS}
\]

(4.6)

where \(\eta_{AS}(0)\) and \(\eta_{AS}(1)\) represent the error propagated from neighbors as well as any arithmetic errors that occurred during computation of \(H_{st}\), and \(\eta_{CS}\) indicates errors that occur in CS. By setting \(\eta_{AS} = \eta_{AS}(0) - \eta_{AS}(1)\), and \(\tilde{\eta}_{CS} = \eta_{CS} + \eta_{AS}(1)\), \(\tilde{M}_{st}(0)\) can be rewritten as

\[
\tilde{M}_{st}(0) = \min \{\underbrace{H_{st}(0) + \eta_{AS}}_{\text{arg1}}, \underbrace{H_{st}(1) + C}_{\text{arg2}}\} + \tilde{\eta}_{CS}
\]

\[
= M_{st}(0) + \tilde{\eta}_{AS} + \tilde{\eta}_{CS}
\]

(4.7)

where \(\tilde{\eta}_{AS}\) and \(\tilde{\eta}_{CS}\) are the effective error in the message \(M_{st}\) generated by AS and CS, respectively. We will refer to \(H_{st}(0) + \eta_{AS}\) and \(H_{st}(1) + C\) as \(\text{arg1}\) and \(\text{arg2}\), respectively, and define \(\Delta H \triangleq H_{st}(1) + C - H_{st}(0)\).

First, assume that \(H_{st}(0) < H_{st}(1) + C\), i.e., \(\Delta H > 0\). Then, when \(\eta_{AS}\) is small, such
that $\eta_{AS} < \Delta H$, $arg1$ will be chosen as the minimum and $\tilde{\eta}_{AS} = \eta_{AS}$. Once $\eta_{AS} \geq \Delta H$, min operation will now choose $arg2$ and $\tilde{\eta}_{AS}$ will be fixed to $\Delta H$ (see Fig. 4.2(a)). Next assume that $H_{st}(0) > H_{st}(1) + C$, i.e., $\Delta H < 0$. In this case, if $\eta_{AS} \geq \Delta H$, $arg2$ will be selected as the minimum, which is the correct minimum, and $\tilde{\eta}_{AS} = 0$. When $\eta_{AS} < \Delta H$, min operation will select $arg1$, and $\tilde{\eta}_{AS} = \eta_{AS} - \Delta H$ (see Fig. 4.2(b)). Note that the effect of $\eta_{AS}$ depends only on $\Delta H$, which, in turn, only depends on the error-free computation of $H_{st}$. Based on Fig. 4.2, we can deduce two error characteristics of TRW-S:

1. TRW-S is affected more by negative errors than positive errors, since the effect of positive errors is either bounded (Fig. 4.2(a)) or removed (Fig. 4.2(b)).

2. If the magnitude of an error is small ($|\eta_{AS}| < |\Delta H|$), it can either be preserved (Fig. 4.2(a)) or removed (Fig. 4.2(b)).

Similar conclusions can be derived for $\tilde{M}_{st}(1)$.

The error generated by CS, $\tilde{\eta}_{CS}$, can be viewed as part of the overall error in a message, $\tilde{\eta} = \tilde{\eta}_{AS} + \tilde{\eta}_{CS}$. The error $\tilde{\eta}$ will affect the message computation of the adjacent node, and thus, can be regarded as a propagated error for the next message computation. Normalization that occurs at the final step of MSG_UPD can also be shown to significantly reduce the magnitude of the propagated errors. Normalization also reduces the effect of $\tilde{\eta}$ significantly,
4.2.2 Verification of error analysis

Two experiments were performed on TRW-S stereo matching hardware to verify our analysis. We first run Tsukuba stereo matching [96] on our software simulator of TRWS_HW with error injection on AS in (4.7) to see the effect of error on the message update. The Potts model with $C = 20$ is used as the smoothness cost, and the computation in hardware has 20-bit fixed point precision. The injected errors, $\eta_{AS}$, are drawn from $U(-128, 128)$, where $U(a,b)$ is a uniform distribution between $a$ and $b$. Figure 4.4(a) shows the resulting relationship between $\eta_{AS}$ and $\tilde{\eta}_{AS}$ summarized as a box plot. Note that $|\tilde{\eta}_{AS}|$ cannot be larger than $C$ due to truncation and normalization in message update. It can be seen that $\tilde{\eta}_{AS}$ is more likely to be close to zero for $\eta_{AS} > 0$ than for $\eta_{AS} < 0$, which agrees with our analysis.

For a macroscopic view of the effect of error on the message-passing performance, we further apply errors with different magnitude on both $\eta_{AS}$ and $\eta_{CS}$ and run the same Tsukuba
stereo matching. We apply uniform errors as follows: \( \eta_{AS}, \eta_{CS} \sim U(\min[0, \eta_{\max}], \max[0, \eta_{\max}]) \), and \(-2^{15} \leq \eta_{\text{max}} \leq 2^{15}\). Note that all injected errors are of the same sign. Fig. 4.4(b) shows the effect of errors on the energy minimization performance for different error magnitudes and error injection rates. If the error magnitude is small (e.g., \(|\eta_{\text{max}}| \leq 512\)), the minimum \( E \), given by (4.1), approaches the error-free performance after 10 iterations. However, if the error magnitude is large (e.g., \(|\eta_{\text{max}}| \geq 1024\)), the minimum \( E \) does not decrease with iterations, which indicates that message passing fails to find the best label assignment. Furthermore, the slope of the minimum \( E \) is steeper in case of the negative errors, as expected by analysis. Therefore, we can conclude that TRW-S is tolerant to small-magnitude errors but suffers from large-magnitude errors.

4.2.3 Enhanced error robustness of TRW-S via ANT

As discussed in Section 4.2.1, message-passing inference, such as TRW-S, has intrinsic robustness to small-magnitude errors but is vulnerable to large-magnitude errors. Thus, the approximate error correction capability of ANT, where large errors are converted to small errors, can provide significant enhancement in error resiliency. In this section, we apply ANT to TRWS_HW and demonstrate its effectiveness. ANT is applied to arithmetic computation of REPARAM and MSG_UPD units of TRWS_HW (Fig. 4.1 (b) and (c)), to compensate arithmetic errors injected on AS and CS. For the estimator, a reduced precision replica (RPR) of the main block (i.e., AS and CS) has been used. This particular estimator has a shorter critical path than does the original block, and thus, is not subject to timing errors but suffers from estimation errors. Figure 4.5 shows the performance of ANT with 4-bit precision estimators when uniform errors with different magnitude, as in Section 4.2.2, were injected at various error rates, \( p_\eta \) (percentage of clock cycles in which an erroneous output exists). Only the tail (\( \eta_{\text{max}} > 32 \)) is shown. For small-magnitude errors (\( \eta < 1024 \)), ANT’s performance is mostly dictated by the main block. When large-magnitude errors (\( \eta > 4096 \)) are injected, the estimator becomes active and successfully compensates for the errors. The slight degradation in performance at medium-sized errors (\( 1024 \leq \eta \leq 8192 \)) is because for such errors, HW errors and estimator errors are similar in magnitude and thus, ANT is not
Figure 4.4: Verification of error analysis: (a) effect of error on message updated via a box plot of $\eta_{AS}$ vs. $\tilde{\eta}_{AS}$, and (b) effect of error on energy minimization performance of message passing via a plot of energy $E$ vs. $\tilde{\eta}$. 

(a) 

(b) 

Figure 4.4: Verification of error analysis: (a) effect of error on message updated via a box plot of $\eta_{AS}$ vs. $\tilde{\eta}_{AS}$, and (b) effect of error on energy minimization performance of message passing via a plot of energy $E$ vs. $\tilde{\eta}$. 

(b)
Figure 4.5: Performance of ANT with injection of uniform errors of different magnitude using 4-bit precision estimator.

able to compensate for the errors. To avoid this ambiguity, HW errors and estimator errors should have very distinct characteristics, which is true for timing based HW errors.

4.3 Error Compensation at Various Levels

In the previous section, we observed that the error resiliency of TRW-S can be enhanced if the arithmetic units are protected by RPR based ANT [91]. However, applying ANT to all the arithmetic units causes large hardware resource overhead. In this section, therefore, we further study the trade-off between granularity of ANT protection and its corresponding overhead. The goal is to find the proper level of error compensation that causes low overhead but maintains high error resilience of the TRW-S hardware.

To this end, errors are compensated via ANT at three levels as illustrated in Fig. 4.6. At the system level, errors are compensated directly on the resulting depth map. At the iteration level, errors are compensated after the computation of each functional block in the message passing unit (Fig. 4.1(a)), i.e., at the end of the reparameterize and message
Figure 4.6: Block diagrams: (a) error compensation at different levels, and (b) detailed comparison of arithmetic-level and iteration-level compensation in the reparameterize unit.

update block. Iteration-level compensation is also performed in an online fashion (compensated results are directly used as updates for the adjacent nodes). A hybrid scheme that employs compensation at both system and iteration levels can enable further reduction in complexity of the estimator. At the arithmetic level, errors are compensated after each arithmetic operation, equivalent to each pipelining stage as shown in Fig. 4.1(b,c). The original computation units of the TRW-S architecture are used as the main block, while
different estimators are used for each level of compensation.

4.3.1 Arithmetic-level compensation

At the lowest level, errors are compensated after each arithmetic operation that includes every AS and CS. Every latch in REPARAM and MSG_UPD (Fig. 4.1(b,c)) is subject to ANT compensation. As the operations in need for error compensation are primitive in nature, our choice of estimators is limited to circuit or structural techniques. In our work, a reduced precision replica (RPR) has been employed at this compensation level. With a suitably designed estimator (i.e., an RPR with proper precision), the resulting ANT architecture is highly reliable, but suffers from high compensation overhead since additional logic is added at each point of protection.

4.3.2 Iteration-level compensation

At the iteration level, compensation occurs at the end of each functional block of the TRW-S hardware architecture (see Fig. 4.6(a)). For example, error compensation is performed only at the output of REPARAM. This is due to the fact that the CS operation within MSG_UPD has a significantly shorter critical path and is not subject to errors. As any approximate computation of REPARAM can be used for the estimator, we simply use RPR as before. With sufficient estimator precision, significant error resiliency is obtained. Also, the overall overhead is reduced compared to the arithmetic-level compensation since fewer detection and correction blocks, which are required for ANT, are used. A detailed compensation comparison of arithmetic level vs. iteration level is depicted in Fig. 4.6(b).

4.3.3 System-level compensation

System-level compensation is performed on the computed erroneous depth map (Fig. 4.6(a)) occurring outside of the TRW-S message-passing loop, while iteration-level compensation occurs at the end of each iteration within the message-passing loop. A separate estimated depth map is used for compensation, which is obtained independently of the TRW-S based
main block. Thus, simpler algorithms such as sum of difference (SAD) based stereo matching algorithm [100] and a scaled version (scaled by a factor of $S$) of TRW-S motivated by hierarchical belief propagation (HBP) [101] can be used as estimators. The idea of HBP is to obtain an overview of inference by running BP on the coarser graph. In HBP, this overview is used as the initial guess of the original inference, but in the system-level compensation, it is used for estimation to compensate erroneous inference. There is a trade-off between accuracy of this estimation and the overhead in execution time of this estimation, since smaller $S$ results in more accurate inference but requires more computation.

### 4.3.4 Combination of iteration-level and system-level compensation

In this dissertation, we show that the system-level compensation can achieve global stereo matching performance while being robust to errors (see Section 4.5). However, its performance depends highly on the output quality of the main block. Thus, additional compensation is needed at the iteration level. A hybrid approach using ANT with a lower precision RPR at the iteration level for reduced compensation overhead, and using system-level compensation to maintain performance provides a good complexity-performance trade-off. This compensation method can be easily extended to system-on-chips (SoC) as such systems make extensive use of HW accelerators combined with a general purpose processor.

### 4.4 System Evaluation Setup

We employ a combination of simulation and emulation to evaluate the reliability of TRWS_HW. A cycle accurate software simulator has been implemented for this purpose, while a CPU+FPGA hybrid system was employed to emulate the Verilog implementation of TRWS_HW, mainly for the purpose of verifying the software simulator results.
4.4.1 Simulation

A cycle accurate software simulator is used to simulate impact of hardware errors on the TRW-S message passing architecture. Three stereo matching benchmarks (Tsukuba, Venus, and Teddy [96]) are used as input to the simulator.

4.4.1.1 Simulation methodology

We evaluate the performance of TRWS_HW in a software simulator via error injection. Input dependent timing based error statistics are obtained for accurate simulation via gate-level simulations using an HDL simulator. First, the gate delay is characterized with respect to supply voltage for basic gates such as a full adder and XOR using circuit-level simulators with a commercial 45 nm process library. Then a structural HDL implementation of the REPARAM and MSG_UPD block is simulated via an HDL simulator using the pre-characterized delay values. By choosing the delay values that correspond to various supply voltages, HDL simulation is effectively run at different voltages. For a supply voltage ($V_{dd}$) less than the critical voltage, errors can be observed in the output. Through characterization of these errors, error statistics are obtained and used to inject errors in the AS and CS block of the software simulator. The simulation methodology is summarized in Fig. 4.7(a). The message-passing unit was designed for $V_{dd,crit} = 1.2 V$ @ $f_{CLK,crit} = 270$ MHz. Figure 4.7(b) shows error statistics obtained for the AS block at $V_{dd} = 0.75 V$. Large-magnitude errors can be seen to occur with high probability, which is expected due to the nature of LSB-first computation. The CS block did not exhibit any errors under these operating conditions, due to its short critical path. Thus, at the arithmetic level, ANT was only applied to AS operations. Each basic computation is protected by reduced precision replica (RPR) based ANT to tolerate timing-induced arithmetic errors occurring in the message-passing hardware at low supply voltage ($V_{dd}$). However, this fine-grain protection causes high overhead, reducing benefits of voltage over-scaling.
Figure 4.7: Simulation methodology: (a) flow diagram, and (b) error statistics for AS at $V_{dd} = 0.75\, V$ with $p_{\eta} = 0.21$.

4.4.1.2 Optimization of precision

We can exploit the intrinsic error robustness of TRW-S to optimize the fixed point precision of arithmetic computations in (4.2) and (4.3). Figure 4.8 shows $E$ vs. precision used in the computation of TRWS_HW when running the Tsukuba stereo matching. The notation $M[a, b, c]$ represents different precision options applied to the main computation of TRWS_HW: a precision of $a$-bits is used to perform the computation of (4.2) and (4.3)
after \( b \)-bit LSB truncation and \( c \)-bit MSB saturation. Truncation and saturation are performed at the output of each function block as depicted in Fig. 4.9. The baseline precision is \( M[20, 0, 0] \) in [92], which outputs the same energy minimization results as the floating point implementation [96]. Compared to this baseline, 8-bit LSB truncation \( (M[12, 8, 0]) \) produces a similarly low energy minimization curve, whereas 12-bit truncation \( (M[8, 12, 0]) \) results in a higher minimum energy curve. According to our experiments, 8-bit precision with less than 11-bit LSB truncation \( (i.e., M[8, 8, 4], M[8, 9, 3], \text{and } M[8, 10, 2]) \) achieves comparable energy minimization performance to the baseline precision \( (M[20, 0, 0]) \). However, other precisions, such as \( M[8, 11, 1] \) or \( M[7, 8, 5] \), perform worse due to large-magnitude quantization or saturation errors. Similarly, the optimal precision of the main block for the other benchmarks, Teddy and Venus, is obtained as \( M[8,8,4] \) and \( M[8,10,2] \), respectively.
4.4.2 Emulation

An emulation platform that generates timing errors was employed to verify the simulation results. Though our entire system was implemented on the TRWS_HW FPGA platform, the previous results were obtained via simulations, as this enabled us to explore injection of the same error statistics, but at different error rate $p_\eta$.

4.4.2.1 Emulation architecture

A TRW-S based stereo matching system has been implemented in a hybrid CPU+FPGA platform to perform high-quality stereo matching in video-rate speed [92]. Figure 4.10 shows the overall architecture of the TRW-S stereo matching system. The platform (Convey HC-1 [102]) contains an Intel Xeon dual core processor and four Virtex 5 (V5LX330) Xilinx FPGAs, with a cache-coherent virtual memory system across both multicore and FPGA fabrics. The two-step (reparameterize, REPARAM, and update message, UPDMSG) message-passing algorithm [97] is accelerated in an FPGA, where its data path is fully pipelined and MRF data is streamed via FIFOs to achieve high throughput. The CPU not only controls FPGA operations but also processes image input and stereo-matching output (i.e., depth map). Our iteration-level error compensation is emulated using this CPU+FPGA, where the precisions of the main block and the estimator are set to 20-bit and 8-bit, respectively.

Figure 4.10: Architecture of streaming TRW-S stereo matching CPU+FPGA system.
4.4.2.2 Error generation

To evaluate the results under real HW errors, the message-passing unit is implemented as an FPGA accelerator exhibiting timing violations via relaxed synthesis. At a target frequency of 150 MHz, all the paths through the REPARAM block were set as false paths. Xilinx’s ChipScope Pro was used to verify that the errors were generated within REPARAM, and compared against error-free results to obtain the error statistics (Fig. 4.11(a)). Figure 4.11(b) shows similar error statistics to the ones obtained through HSPICE simulations (Fig. 4.11(c)) based on the modeling methodology in [91], indicating that the error generation in FPGA emulates the HW timing errors well.

However, controlling the error rate $p_\eta$ in an FPGA system is known to be difficult due to various synthesis and mapping constraints and unpredictable routing within the FPGA [103]. The buffer chain in Fig. 4.11(a) is such an example. Contrary to our expectations, $p_\eta$ and the resulting matching performance did not have strong dependency with the length of the buffer chain. To enable flexible control of $p_\eta$ we extracted the error statistics of the FPGA (Fig. 4.11(b)) and once again used error injection in the simulator. The results of the simulator are comparable with the FPGA emulation results, justifying this approach.

4.5 Results

4.5.1 Simulation results for arithmetic-level compensation

The Tsukuba stereo matching task is run at various $V_{dd}$ for erroneous message-passing computations. A precision of M[8, 8, 4] has been used for the precision optimized main block, and an estimator of precision E[a, b, c] is used for ANT. Figure 4.12(a) shows the energy minimization performance of ANT with different estimators at various $V_{dd}$. It is evident that energy minimization performance is drastically degraded in the conventional case (no error protection), while ANT introduces tolerance to a significant amount of errors. Compared to 2-bit and 3-bit precision estimators, estimators with precision higher than 4-bit show much lower minimum energy. At $V_{dd} = 1$ V, where the error rate of the AS block is $p_{\eta,AS} = 0.8\%$,
Figure 4.11: Timing errors in FPGA: (a) block diagram for error verification and statistics collection, (b) measured error statistics (20-bit) in the FPGA, and (c) error statistics (8-bit) obtained via circuit simulations in a 45 nm CMOS process [91].

the minimum energy of $E[4, 12, 4]$ is almost the same as the minimum energy of the error-free case. Similarly, the best estimator precision for the other benchmarks, Teddy and Venus, is
Figure 4.12: Simulation results of error injection rate vs. energy minimization for different ANT estimators with $M[8, 8, 4]$.

obtained as $E[4,12,4]$ and $E[4,14,2]$, respectively.

To further establish the effectiveness of ANT for TRWS_HW, the depth maps of three cases – error free, conventional, and ANT – are compared. The estimator precision is set to $E[4,12,4]$. Table 4.1 shows the depth map of each case at different $V_{dd}$. The depth map for the conventional case becomes corrupted even at a low error rate of 1%. In contrast, the depth map when ANT is applied is comparable to the error-free case when the same 1% error rate is applied. Furthermore, at low $V_{dd}$, where the error rate is 10% to 25%, the depth map of ANT is still close to the depth map of the error-free case, which demonstrates the outstanding error robustness of ANT.

To evaluate the accuracy of the depth maps, bad-pixel ratio (BPR) is employed [104] as the system-level metric. BPR is calculated by comparing the depth label of each pixel in the non-occlusion region to the true depth map (ground truth) and counting a pixel to be bad if the label differs by more than a threshold $\kappa$ ($\kappa = 1$ in our case). As shown in Table 4.1, BPR of the conventional case rises drastically from 10.4% to 70.4% as $V_{dd}$ is scaled down. In contrast, BPR of ANT is robust to errors; BPR is at most 6.27%, at a high error rate of
Table 4.1: Depth map and BPR comparison for error-free, conventional, and ANT at various $V_{dd}$.

<table>
<thead>
<tr>
<th>Task</th>
<th>Tsukuba</th>
<th>Teddy</th>
<th>Venus</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_{dd}, p_{h,AS}$</td>
<td>Error-Free</td>
<td>Conv.</td>
<td>ANT</td>
</tr>
<tr>
<td>1.0 V</td>
<td><img src="image1" alt="Image" /></td>
<td><img src="image2" alt="Image" /></td>
<td><img src="image3" alt="Image" /></td>
</tr>
<tr>
<td>0.8%</td>
<td>2.7%</td>
<td>10.4%</td>
<td>2.51%</td>
</tr>
<tr>
<td>0.9 V</td>
<td><img src="image11" alt="Image" /></td>
<td><img src="image12" alt="Image" /></td>
<td><img src="image13" alt="Image" /></td>
</tr>
<tr>
<td>2.9%</td>
<td>2.7%</td>
<td>60.7%</td>
<td>2.31%</td>
</tr>
<tr>
<td>0.84 V</td>
<td><img src="image21" alt="Image" /></td>
<td><img src="image22" alt="Image" /></td>
<td><img src="image23" alt="Image" /></td>
</tr>
<tr>
<td>5.1%</td>
<td>2.7%</td>
<td>70.3%</td>
<td>3.93%</td>
</tr>
<tr>
<td>0.75 V</td>
<td><img src="image31" alt="Image" /></td>
<td><img src="image32" alt="Image" /></td>
<td><img src="image33" alt="Image" /></td>
</tr>
<tr>
<td>21.3%</td>
<td>2.7%</td>
<td>70.4%</td>
<td>6.27%</td>
</tr>
</tbody>
</table>
Table 4.2: Estimated compensation overhead and power consumption of arithmetic-level ANT obtained via synthesis in a commercial 45 nm CMOS process.

<table>
<thead>
<tr>
<th></th>
<th>M [8,8,4] only</th>
<th>ANT (M[8,8,4] + E[4,8,4])</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cell</td>
<td>40,790</td>
<td>80,534</td>
</tr>
<tr>
<td>$V_{dd}$ (V)</td>
<td>1.10</td>
<td>1.10</td>
</tr>
<tr>
<td>Power (mW)</td>
<td>11.82</td>
<td>17.81</td>
</tr>
<tr>
<td>(leak, dyn)</td>
<td>(2.03; 9.79)</td>
<td>(3.39;14.42)</td>
</tr>
</tbody>
</table>

21.3% for the Tsukuba task.

Such enhancement in error robustness of ANT can be exploited to achieve power savings in circuit implementation. The power consumption of the message-passing unit is estimated via switching activity based RTL synthesis in a commercial 45 nm synthesis library. Switching activity is obtained from Verilog simulations running stereo matching of the Tsukuba task. This switching activity is then annotated in RTL synthesis to estimate accurate power consumption. Table 4.2 summarizes the synthesis results with cell utilization and estimated power consumption including leakage and dynamic power. As can be seen, the cell overhead of ANT is approximately 97.4%, which is significant. However, at an extreme supply voltage of $V_{dd} = 0.72$ V, ANT is able to achieve 39.7% power savings while maintaining the system performance. This result shows that significant energy savings is possible via voltage scaling, even when additional complexity has to be added for the system to perform correctly. To overcome the significant overhead in complexity, compensation at higher levels, iteration and system, is considered next.

4.5.2 Simulation results for system-level and iteration-level compensation

Once again, a cycle accurate simulator was used to perform simulation of system- and iteration-level compensation. BPR was used as our performance metric. An RPR based HW estimator (HWRPR) with 2- to 4-bit precision was used for the iteration-level compensation, while SAD of window size 7 ($W = 7$), or HBP with a scale factor of $S = 8$ and $S = 16$, were used for the estimator at the system level.

SAD based correction at the system level with $V_{dd} = 0.75$ V is shown in Fig. 4.13(a).
At this error rate, system-level correction gives marginal improvement and is insufficient for satisfactory operation (main block BPR is 68.5%). Using HBP (Fig. 4.13(b)) alone, the final output is significantly better at a BPR of 6.95%. Further combining with HWRPR with 3-bit precision, BPR of 5.21% is achieved (Fig. 4.13(c)), which is a 2.5% degradation compared to the error-free BPR of 2.7%. Note that this error compensation result (BPR of 5.21%) is even better than the result of arithmetic error compensation (BPR of 6.27%).

Figure 4.14 summarizes the error compensation performance of the system- and iteration-level compensation. As expected, HBP with low-precision HWRPR shows comparable error resiliency to high-precision (and large overhead) HWRPR. One interesting observation is that the hybrid scheme combining HBP and HWRPR performs exceptionally well under aggressive voltage scaling. However, the same cannot be said for moderate voltage scaling. This is due to the limitation in quality of the HBP estimator used at the system level.

The overhead of HWRPR for the iteration-level compensation is derived through the same activity factor based RTL synthesis. A summary of the power estimation and the cell utilization for a 4-bit, 3-bit, and 2-bit estimator is given in Table 4.3. Compared to arithmetic-level compensation, the cell utilization has been reduced by 11.8% to 20.3%. Using the same synthesis based power estimation as for the arithmetic compensation, the power consumption of the iteration-level ANT was estimated to be 5.227 mW at the lowest point, which is an additional 16.1% power savings.

### 4.5.3 Emulation results

For the estimator in emulation, the Intel Xeon processor available within the convey HC-1 platform was used to compute estimates of the depth map.

The correctness of our simulator is verified by comparing the results against FPGA emulation. FPGA emulation results of ANT applied at the iteration level and system level for Tsukuba are shown in Fig. 4.15. For iteration-level compensation, the resulting performance is very similar, and we conclude that the simulator faithfully represents the FPGA emulation results.

Many system-on-chip (SoC) systems employ a platform consisting of a general purpose
Figure 4.13: Correction performance at (a) system level with SAD, (b) system level with HBP ($S = 8$), and (c) hybrid with HBP ($S = 8$) and HWRPR(3-bit).
Figure 4.14: Bad-pixel ratio vs. $V_{dd}$. With no error compensation, TRW-S alone cannot tolerate 10% voltage scaling.

Figure 4.15: Results for RPR-ANT applied at the arithmetic level: (a) FPGA emulation, and (b) error injection based simulation.
Table 4.3: Estimated compensation overhead and power consumption of iteration-level ANT obtained via synthesis in a commercial 45 nm CMOS process.

<table>
<thead>
<tr>
<th></th>
<th>ANT (M[8,8,4] + E[4,12,4])</th>
<th>ANT (M[8,8,4] + E[3,13,4])</th>
<th>ANT (M[8,8,4] + E[2,14,4])</th>
</tr>
</thead>
<tbody>
<tr>
<td>( V_{dd} ) (V)</td>
<td>1.10</td>
<td>1.10</td>
<td>1.10</td>
</tr>
<tr>
<td>Cell</td>
<td>71,096</td>
<td>67,419</td>
<td>68,081</td>
</tr>
<tr>
<td>Power (mW)</td>
<td>15.234</td>
<td>13.814</td>
<td>13.432</td>
</tr>
<tr>
<td>(leak, dyn)</td>
<td>(3.131; 12.102)</td>
<td>(2.866; 10.948)</td>
<td>(2.755; 10.678)</td>
</tr>
</tbody>
</table>
processor and accelerators. Under this environment, when the accelerator is under load, the processor can be idle and available for computation of simple estimations. Thus, we employed a general purpose processor to perform computation of the HBP based depth map. However, the error rate on the FPGA emulation platform was not high enough for a degraded iteration level output, and thus system-level compensation was not able to produce any more performance benefits. Nonetheless, the concept of using a general purpose processor for computation of the estimator has been shown to be valid. One important point of consideration is that the execution time must be less than that of the accelerator, and the complexity of HBP depends on the number of nodes in the graph, which is $\frac{1}{64}$ for $S = 8$, and $\frac{1}{256}$ for $S = 16$. The execution time for the HBP estimator on a Intel Xeon @ 2.13 GHz with 10 iterations was approximately 250 ms and 60 ms for $S = 8$ and 16, respectively, while the main block took approximately 130 ms for 40 iterations. Thus, $S = 16$ is a viable choice for performing estimation on a processor.

4.6 Summary

Statistical error compensation (ANT) in the form of algorithmic noise tolerance (ANT) has been applied to a Markov random field (MRF) based stereo image matching architecture. The error propagation characteristics of an iterative message-passing based stereo image matching application has been studied in depth. Analysis and simulations show that for a 20-bit architecture, small errors ($\eta \leq 1024$) are tolerable, while large errors ($\eta \geq 4096$) degrade the performance significantly. Low overhead compensation methods were proposed and applied at various levels (system, iteration, and arithmetic) and show significant energy savings can be achieved. For best results, system-level correction reduces correction complexity by a significant amount, but is inadequate. When combined with iteration-level compensation, performance gains are significant with little increase in complexity.
Chapter 5

Approximate Computing Based Statistical Error Compensation

Statistical error compensation (SEC) has shown significant benefits in achieving energy efficiency and error resiliency by embracing the stochastic nature of the underlying process (see Chapters 2, 3, and 4). Approximate computing (AC) [22], on the other hand, employs deterministic designs that produce imprecise results to achieve energy efficiency. In this chapter, we bridge the two design paradigms by utilizing SEC and AC in the design of a machine learning accelerator core.

Approximate computation (AC) [22], as briefly discussed in Section 1.2.4, relaxes the required numerical exactness on the output. This relaxation is based on application-level information just as SEC techniques have used application-level information to achieve enhancement in error resiliency. Emerging new applications, including recognition, mining, and synthesis (RMS), process large amounts of noisy data via statistical and probabilistic computation and are inherently error resilient. Furthermore, many machine learning applications have application metrics that are probabilistic, and are inherently robust to errors as discussed in Section 1.2. Similar to SEC, these characteristics motivate the use of AC. The goal of approximate computing is to prune the computation such that the application is oblivious to the approximation and thereby gains energy efficiency or obtains higher performance.

AC requires the computation to be free of hardware errors (i.e., $y = y_o + e$). AC is based on intentional simplification of the design that produces imprecise results by utilizing statistical properties of data and algorithms [22]. Careful study of AC based algorithms needs to be performed to ensure system-level performance. Through this process, the quality of the output is traded off with area and energy. On the other hand, SEC is a more aggressive technique that relaxes the correctness requirement of the HW implementation. SEC embraces
the stochastic nature of the underlying implementation then provides approximate error compensation such that the output may be in error but the application-level performance specifications are met.

In this chapter, we combine SEC and AC in the implementation of an accelerator architecture for a tree-reweighted message-passing (TRW-S) based stereo image matching application (see Chapter 4). Algorithmic noise tolerance (ANT) was applied at the arithmetic level while a mirror adder based approximate adder [21] was used in the application of AC. Hardware errors were induced in the message update and reparameterize unit of the TRW-S HW architecture. A study on the trade-off between the amount of approximation and the energy efficiency was performed by implementing AC at different granularity. To jointly optimize the performance, AC was not necessarily applied at the lowest significant bits (LSBs). Results show that ANT combined with AC achieves energy savings of 44.9% compared to a conventional system while achieving at most 4% degradation in bad-pixel ratio (BPR). This result further strengthens the belief that energy efficiency is improved by embracing the underlying stochasticity, i.e., circuit design should no longer follow worst-case design principles, but allow errors to propagate to the next level via nominal case design.

The remainder of the chapter is organized as follows. Section 5.1 provides background information on approximate computing. The TRW-S MRF message-passing based stereo matching algorithm and its architecture are discussed in Section 5.2. Section 5.3 shows the simulation setup and results, while Section 5.3.4 concludes the chapter.

5.1 Approximate Computing

In this section, we present a comprehensive survey on AC techniques. The actual AC technique used in this dissertation, a mirror adder based approximate adder, is then discussed in further detail.
5.1.1 Circuit-level AC techniques

5.1.1.1 Approximate adders

Approximate adders are implemented in two parts, an accurate MSB part and an approximate LSB part. For the approximate LSB part, a simplified full adder is used. The truth table of the full adder is altered to provide significant savings in the number of transistors required. Many design variations exist. Approximate mirror adders [21] are based on mirror adders. Five approximate adders are designed, each based on a different alteration of the truth table. More details on approximate mirror adders are given in Section 5.1.3. Approximate XOR/XNOR based adders are based on the 10-transistor adder that uses multiplexers implemented using pass transistors. Three different approximate adders are presented in [34] and show good error distance [105] properties with low power-delay product compared to other approximate adders. A third type of approximate adder, lower-part-OR adder (LOA) [35], uses OR gates to approximately compute the LSBs. The carry in for the MSB part is computed through an AND gate that takes inputs to the MSB of the LSB part. Most carries are ignored in the lower-part module of the LOA resulting in a loss of precision.

5.1.1.2 Approximate multipliers

In [35, 36], approximate multipliers are constructed via the use of speculative adders on obtaining the sum of partial products. However, directly applying approximate adders to the design of multipliers may not be suitable for approximate multiplier designs. In [37, 38], it is noted that reducing the critical path of partial product adders is essential. In [38], instead of directly replacing adders with approximate adders, partial sums that are part of the LSBs are simply omitted. In [37], a hierarchical multiplier is used, where an approximate $2 \times 2$ multiplier is implemented, then used to build a larger multiplier.
5.1.1.3 Approximate logic synthesis

Approximate logic-level synthesis has been explored as well. In [40], logic synthesis is used to synthesize the minimum area circuit, at a given error rate. This is performed in a two-step synthesis process. In [41], a multi-level logic minimization algorithm is developed to simplify the design and minimize the area of approximate circuits. A systematic methodology for automatic logic synthesis of approximate circuits (SALSA) has been proposed in [42]. SALSA encodes quality constraints in logic functions known as Q-functions. The increase in flexibility denoted by Q-functions is then represented as approximate don’t cares, and traditional circuit simplification and don’t care optimization techniques are utilized for synthesis.

5.1.2 Algorithmic-level AC techniques

Increased area and energy savings can be achieved when AC is applied at the algorithm level. The quality of the output of iterative algorithms enhance incrementally with each iteration. This characteristic, known as incremental refinement has been exploited to achieve results that gradually increase in quality. In [43,44], this idea was demonstrated on a FFT-based maximum-likelihood detection algorithm. This was expanded in [45,46], where several DSP algorithms were transformed to exhibit the incremental refinement property. In machine learning, an approximate support vector machine (SVM) was shown [47], where the number of dimensions (features) used in the SVM were reduced. Algorithmic AC techniques, in essence, can be employed to design the estimator in ANT.

Inexact computing is based on a similar principle. In this work, probabilistic pruning [106,107] and probabilistic logic minimization [108] is applied at the architecture and logic level. In probabilistic pruning, components that have lower significance or probability of activation are pruned in a systematic way, while probabilistic logic minimization transforms logic functions to a lower cost variant.
Table 5.1: Truth table for the mirror based approximate adder 1 through 4 [21].

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Accurate Outputs</th>
<th>Approximate Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>B</td>
<td>$C_{in}$</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

5.1.3 Mirror adder based approximate adder

In this dissertation, we have used the mirror adder based approximate adder 1 through 4 in [21], where the truth table is shown in Table 5.1. By simplifying the truth table of the adder, fewer transistors can be used to implement the approximate version of the adder, resulting in area and energy reduction. The accurate mirror adder and the four different approximate full adders corresponding to the truth table are depicted in Fig. 5.1. As can be seen, the approximate versions require fewer transistors, and result in area savings ranging from 28% to 55%. Further energy savings can be achieved if the reduction in the critical path is traded off with energy via voltage scaling. A ripple carry adder that utilizes these approximate adders was applied in the message passing unit of the TRWS_HW.

5.2 TRW-S Accelerator Architecture Using AC and SEC

5.2.1 TRW-S accelerator architecture

In this chapter, we employ the same TRW-S hardware architecture from Chapter 4, where the two-step message passing of TRW-S is pipelined and executed in a streaming architecture (Fig. 4.1). FIFOs are used to handle streaming data access to the main memory in order
Conventional MA:

Approximation 1:

Approximation 2:

Approximation 3:

Approximation 4:

Figure 5.1: Approximate mirror adders [21]: (a) conventional mirror adder, (b) approximate mirror adder 1, (c) approximate mirror adder 2, (d) approximate mirror adder 3, and (e) approximate mirror adder 4.
to feed the pipelined message-passing units without many stalls. The REPARAM unit is a simple hardware realization of (4.2), while for MSG_UPD, a parallel message construction technique [99] is exploited to reduce complexity of the pairwise computation in (4.3), as shown in Fig. 4.1(c). In addition, to avoid overflow, MSG_UPD includes a normalization step, which rescales the value of messages so that the minimum message is always zero.

5.2.2 Design of ANT and AC

ANT and AC can be combined in a number of ways. The available design options are described using the following notation:

- \(C[a]\) represents a conventional system (no robust design applied, i.e., neither AC nor ANT applied) designed with a bit precision of \(a\)-bits.

- \(AC[a, b, c]\) represents an AC only design, with total bit precision \(a\)-bits, but using AC on \(b\)-bits from the MSB and \(c\)-bits from the LSB.

- \(ANT[a, b, c; d, e, f]\) represents an ANT design combined with AC. The three values, \(a, b,\) and \(c\), represent the main block designed with \(AC[a, b, c]\), while the latter three values, \(d, e,\) and \(f\), represent that the estimator is designed using \(AC[d, e, f]\). Thus, an ANT only design (with no AC), is denoted as \(ANT[a, 0, 0; d, 0, 0]\) or \(ANT[a; d]\), for simplicity.

ANT was applied at the arithmetic level, i.e., errors are compensated after each arithmetic operation that includes every add or subtract (AS) and compare and select (CS). Every latch in the reparameterize unit and message update unit (Fig. 4.1(b,c)) is subject to ANT compensation. As the operations to correct are primitive in nature, our choice of estimators is limited to circuit or structural techniques. In this dissertation, a reduced precision replica (RPR) has been employed. An 8-bit main block \((a = 8)\) and a 4-bit estimator \((d = 4)\) were used based on previous work [91].

AC has been applied exclusively to the main block. This is because ANT gains its benefits via VOS, and the critical path of the estimator is an important factor in determining the
amount of VOS that can be applied. Thus, instead of applying AC to the estimator, only truncation was used. The bit location where AC has been applied was varied and was not limited to the least significant bits (LSBs), as ANT can provide compensation for large-magnitude errors. In this chapter, we compare the following:

- Conventional: \( C[8] \).
- AC only design: \( AC[8, 0, 1], AC[8, 0, 2], \) and \( AC[8, 1, 2] \).
- ANT only design: \( ANT[8; 4] \).
- ANT combined with AC: \( ANT[8, 0, 1; 4], ANT[8, 0, 2; 4], \) and \( ANT[8, 1, 1; 4] \).

Four different AC based adder structures as shown in Fig. 5.1 [21] were used in the design of the AC based main block.

5.3 Simulation and Results

5.3.1 Simulation setup

A cycle accurate software simulator is used to simulate the impact of hardware errors on the TRW-S message-passing architecture. The Tsukuba task from the Middlebury benchmark [96,100] is used as input to the simulator.

HW errors are modeled as voltage scaling induced timing errors, and error injection was used to evaluate performance. Input dependent timing based error statistics are obtained for accurate simulation via gate-level simulations using an HDL simulator. First, the gate delay is characterized with respect to supply voltage for basic gates such as a full adder and XOR using circuit-level simulators with a commercial 45 nm process library. Then a structural HDL implementation of the REPARAM and MSG_UPD block is simulated via an HDL simulator using the pre-characterized delay values. By choosing the delay values that correspond to various supply voltages, HDL simulation is effectively run at different voltages. For a supply voltage \( (V_{dd}) \) less than the critical voltage, errors can be observed in the output. Through characterization of these errors, error statistics are obtained and
used to inject errors in the AS and CS block of the software simulator. The simulation methodology is summarized in Fig. 5.2.

5.3.2 Error resiliency

The performance of the TRW-S HW is measured in terms of the energy $E$ (4.1) and BPR. BPR is calculated by comparing the depth label of each pixel in the non-occlusion region to the true depth map (ground truth) and counting a pixel to be bad if the label differs by more than a threshold $\kappa$ ($\kappa = 1$ in our case). Four different types of approximate adders were simulated using approximate mirror adder 1 through 4 in [21].

Figure 5.3 shows the energy $E$ of various design choices vs. the supply voltage $V_{dd}$. From Fig. 5.3, it can be seen that ANT based systems are significantly robust to VOS as compared to AC or conventional systems. This is not surprising because in Chapter 4, we showed that a reduction up to 34% in $V_{dd}$ is possible with a BPR degradation of at most 4%. In particular, as $V_{dd}$ is scaled, $ANT[8,1,1;4]$ has significantly lower $E$ than $AC[8,1,1]$ and $C[8]$. This is because ANT is able to correct MSB errors induced by the use of AC and VOS.

Furthermore, $ANT[8;4]$, which is an ANT only design, has minimum energy $E$ at any voltage as compared to all other options. This is because introducing AC incurs additional estimation errors when the main block output is chosen as the final output. This is a very
Figure 5.3: Energy $E$ vs. supply voltage $V_{dd}$ for TRW-S implemented with AC using: (a) approximate adder 1, (b) approximate adder 2, (c) approximate adder 3, and (d) approximate adder 4 [21]. It can be seen that AC can tolerate at most 10% scaling in $V_{dd}$, whereas when combined with ANT, up to 34% scaling in $V_{dd}$ can be tolerated.
frequent event as $30\% \leq p_\eta \leq 40\%$.

Also, as shown in Fig. 5.3, AC based designs result in less degradation in performance at moderately scaled voltages, as it is subject to fewer hardware errors due to its shorter critical path. The energy $\mathcal{E}$ for $AC[8, 0, 2]$ is lower by 58.1% to 33.8% at $V_{dd} = 1V$, depending upon the type of approximate adder being used. This is because the TRW-S architecture is resilient to small-magnitude errors as shown in Chapter 4, and due to its shorter critical path, $AC[8, 0, 2]$ has fewer hardware errors than both $AC[8, 0, 1]$ and $C[8]$. Further, the gap in $\mathcal{E}$ between AC based design and conventional design reduces with $V_{dd}$. This is because as $V_{dd}$ is scaled, hardware errors begin to dominate over the estimation errors induced by AC.

5.3.3 Overhead and energy savings

The overhead of AC and ANT was measured through the cell count as synthesized in a commercial 45 nm process. Tables 5.2 to 5.5 summarizes the result. Two supply voltages are listed in the table: first is the minimum supply voltage with error-free operation (critical voltage, $V_{dd,crit}$), and the second is the minimum voltage that achieves a bad-pixel ratio (BPR) lower than 6.7% ($V_{dd,BPR}$), which is 4% more than the error-free BPR of 2.7%. The results for $ANT[8; 4]$ show an overhead of 2.21, which is consistent with the results obtained in Chapter 4. The difference is due to the synthesis library used; in order to reduce leakage at low $V_{dd}$, a high threshold voltage library was used. It can be seen that the overhead and $V_{dd,crit}$ greatly depends on the number of bits AC was applied to regardless of whether it was applied at the LSB or MSB. However, $V_{dd,BPR}$ has a higher dependency on LSBs than on MSBs. The largest energy savings of 44.9% is achieved for $ANT[8, 0, 2; 4]$ using approximate adder 3 compared to $C[8]$. This is in spite of the greater than $2 \times$ increase in cell count overhead.

Also, note that $ANT[8, 0, 2; 4]$ is not the design that operates at the lowest $V_{dd}$; however, when combined with the savings that AC provides, the total energy savings is the largest. In general, an energy savings peak is observed at a $V_{dd}$ slightly higher than $V_{dd,BPR}$. This peak is due to the fact that, as $V_{dd}$ is scaled, it gives diminishing returns (see Fig 3.1), whereas, AC provides a constant proportion of energy savings regardless of $V_{dd}$. Thus, at
Table 5.2: Estimated cell count and power consumption obtained via synthesis in a 45 nm CMOS process with AC using approximate adder 1 [21].

<table>
<thead>
<tr>
<th></th>
<th>C[8]</th>
<th>AC[8,0,1]</th>
<th>AC[8,0,2]</th>
<th>AC[8,1,1]</th>
<th>ANT[8;4]</th>
<th>ANT[8,0,1;4]</th>
<th>ANT[8,0,2;4]</th>
<th>ANT[8,1,1;4]</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_{dd,crit}$ (V)</td>
<td>1.10</td>
<td>1.09</td>
<td>1.08</td>
<td>1.08</td>
<td>1.10</td>
<td>1.09</td>
<td>1.08</td>
<td>1.08</td>
</tr>
<tr>
<td>power (mW)</td>
<td>11.82</td>
<td>11.62</td>
<td>11.49</td>
<td>11.51</td>
<td>17.81</td>
<td>17.62</td>
<td>17.38</td>
<td>17.39</td>
</tr>
<tr>
<td>$V_{dd,BPR}$ (V)</td>
<td>N/A</td>
<td>1.08</td>
<td>1.06</td>
<td>1.08</td>
<td>0.72</td>
<td>0.76</td>
<td>0.76</td>
<td>0.84</td>
</tr>
<tr>
<td>power (mW)</td>
<td>N/A</td>
<td>11.41</td>
<td>11.13</td>
<td>11.61</td>
<td>7.14</td>
<td>8.03</td>
<td>7.88</td>
<td>8.62</td>
</tr>
<tr>
<td>energy savings</td>
<td>N/A</td>
<td>3.5%</td>
<td>5.8%</td>
<td>2.6%</td>
<td>39.7%</td>
<td>32.1%</td>
<td>33.3%</td>
<td>27.1%</td>
</tr>
<tr>
<td>cell count</td>
<td>40,790</td>
<td>39,215</td>
<td>37,643</td>
<td>37,589</td>
<td>90,524</td>
<td>88,215</td>
<td>86,733</td>
<td>86,850</td>
</tr>
<tr>
<td>overhead</td>
<td>1.0</td>
<td>0.96</td>
<td>0.92</td>
<td>0.92</td>
<td>2.21</td>
<td>2.16</td>
<td>2.13</td>
<td>2.13</td>
</tr>
</tbody>
</table>

Table 5.3: Estimated cell count and power consumption obtained via synthesis in a 45 nm CMOS process with AC using approximate adder 2 [21].

<table>
<thead>
<tr>
<th></th>
<th>C[8]</th>
<th>AC[8,0,1]</th>
<th>AC[8,0,2]</th>
<th>AC[8,1,1]</th>
<th>ANT[8;4]</th>
<th>ANT[8,0,1;4]</th>
<th>ANT[8,0,2;4]</th>
<th>ANT[8,1,1;4]</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_{dd,crit}$ (V)</td>
<td>1.10</td>
<td>1.09</td>
<td>1.07</td>
<td>1.07</td>
<td>1.10</td>
<td>1.09</td>
<td>1.07</td>
<td>1.07</td>
</tr>
<tr>
<td>power (mW)</td>
<td>11.82</td>
<td>11.58</td>
<td>11.03</td>
<td>11.13</td>
<td>17.81</td>
<td>16.92</td>
<td>16.80</td>
<td>16.82</td>
</tr>
<tr>
<td>$V_{dd,BPR}$ (V)</td>
<td>N/A</td>
<td>1.05</td>
<td>1.02</td>
<td>1.07</td>
<td>0.72</td>
<td>0.74</td>
<td>0.76</td>
<td>0.82</td>
</tr>
<tr>
<td>power (mW)</td>
<td>N/A</td>
<td>10.38</td>
<td>9.83</td>
<td>11.13</td>
<td>7.14</td>
<td>7.03</td>
<td>6.62</td>
<td>8.53</td>
</tr>
<tr>
<td>energy savings</td>
<td>N/A</td>
<td>8.0%</td>
<td>16.8%</td>
<td>5.89%</td>
<td>39.7%</td>
<td>40.5%</td>
<td>44.0%</td>
<td>27.8%</td>
</tr>
<tr>
<td>cell count</td>
<td>40,790</td>
<td>38,218</td>
<td>35,370</td>
<td>35,152</td>
<td>90,524</td>
<td>87,512</td>
<td>85,370</td>
<td>85,414</td>
</tr>
<tr>
<td>overhead</td>
<td>1.0</td>
<td>0.94</td>
<td>0.87</td>
<td>0.86</td>
<td>2.21</td>
<td>2.15</td>
<td>2.09</td>
<td>2.09</td>
</tr>
</tbody>
</table>
Table 5.4: Estimated cell count and power consumption obtained via synthesis in a 45 nm CMOS process with AC using approximate adder 3 [21].

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_{dd,crit}$ (V)</td>
<td>1.10</td>
<td>1.09</td>
<td>1.07</td>
<td>1.07</td>
<td>1.10</td>
<td>1.09</td>
<td>1.07</td>
<td>1.07</td>
</tr>
<tr>
<td>power (mW)</td>
<td>11.82</td>
<td>11.63</td>
<td>10.92</td>
<td>11.02</td>
<td>17.81</td>
<td>16.95</td>
<td>16.72</td>
<td>16.73</td>
</tr>
<tr>
<td>$V_{dd,BPR}$ (V)</td>
<td>N/A</td>
<td>1.06</td>
<td>1.02</td>
<td>1.07</td>
<td>0.72</td>
<td>0.73</td>
<td>0.75</td>
<td>0.81</td>
</tr>
<tr>
<td>power (mW)</td>
<td>N/A</td>
<td>11.21</td>
<td>9.62</td>
<td>11.02</td>
<td>7.14</td>
<td>6.99</td>
<td>6.51</td>
<td>8.48</td>
</tr>
<tr>
<td>energy savings</td>
<td>N/A</td>
<td>5.2%</td>
<td>18.6%</td>
<td>6.8%</td>
<td>39.7%</td>
<td>40.8%</td>
<td>44.9%</td>
<td>28.3%</td>
</tr>
<tr>
<td>cell count</td>
<td>40,790</td>
<td>37,972</td>
<td>34,518</td>
<td>34,499</td>
<td>90,524</td>
<td>86,825</td>
<td>84,821</td>
<td>84,905</td>
</tr>
<tr>
<td>overhead</td>
<td>1.0</td>
<td>0.93</td>
<td>0.85</td>
<td>0.85</td>
<td>2.21</td>
<td>2.13</td>
<td>2.08</td>
<td>2.08</td>
</tr>
</tbody>
</table>

Table 5.5: Estimated cell count and power consumption obtained via synthesis in a 45 nm CMOS process with AC using approximate adder 4 [21].

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_{dd,crit}$ (V)</td>
<td>1.10</td>
<td>1.09</td>
<td>1.07</td>
<td>1.07</td>
<td>1.10</td>
<td>1.09</td>
<td>1.07</td>
<td>1.07</td>
</tr>
<tr>
<td>power (mW)</td>
<td>11.82</td>
<td>11.59</td>
<td>10.90</td>
<td>10.98</td>
<td>17.81</td>
<td>16.95</td>
<td>16.72</td>
<td>16.73</td>
</tr>
<tr>
<td>$V_{dd,BPR}$ (V)</td>
<td>N/A</td>
<td>1.06</td>
<td>1.04</td>
<td>1.07</td>
<td>0.72</td>
<td>0.74</td>
<td>0.79</td>
<td>0.88</td>
</tr>
<tr>
<td>power (mW)</td>
<td>N/A</td>
<td>11.18</td>
<td>9.68</td>
<td>10.98</td>
<td>7.14</td>
<td>7.08</td>
<td>8.31</td>
<td>10.21</td>
</tr>
<tr>
<td>energy savings</td>
<td>N/A</td>
<td>5.4%</td>
<td>19.0%</td>
<td>7.1%</td>
<td>39.6%</td>
<td>40.1%</td>
<td>29.7%</td>
<td>13.6%</td>
</tr>
<tr>
<td>cell count</td>
<td>40,790</td>
<td>38,026</td>
<td>34,873</td>
<td>34,780</td>
<td>90,524</td>
<td>87,014</td>
<td>85,093</td>
<td>85,111</td>
</tr>
<tr>
<td>overhead</td>
<td>1.0</td>
<td>0.93</td>
<td>0.85</td>
<td>0.85</td>
<td>2.21</td>
<td>2.13</td>
<td>2.09</td>
<td>2.09</td>
</tr>
</tbody>
</table>
high $V_{dd}$ the energy savings due to VOS dominates, while at low $V_{dd}$ energy savings due to AC becomes an important factor, and a trade-off point exists. The savings for the AC only designs (AC[8,b,c]) is less than the amount reported in [21], as the number of bits where AC was applied was limited to at most two bits to ensure system-level performance.

We also show that applying AC at the MSB is a viable design choice when combined with ANT, as is the case with ANT[8,1,1;4]. This shows that ANT can indeed tolerate large-magnitude errors, and jointly optimizing the design with AC and ANT will result in greater energy efficiency.

5.3.4 Conclusions

In this chapter, we have studied the use of both AC and ANT for robust and energy-efficient design. First, AC-only designs were shown to achieve up to 10% energy savings by incorporating application-level information. When combined with ANT, significant energy savings, in the order of 45% can be achieved. This shows that embracing the stochasticity of the underlying process is crucial in achieving high energy efficiency. This dissertation has mainly focused on the application of AC to the main block. However, the design space of AC combined with ANT is large. Additional cases where AC is applied only to the estimator, and where AC is applied to the main block and estimator together should be studied. After full exploration of the design space, a design methodology to obtain the optimal design in combining AC and ANT can be obtained. To reduce the ANT overhead, AC combined with ANT at the iteration and system level (Chapter 4) is another direction for future work.
Chapter 6

Analysis of Statistical Error Compensation Techniques

In the previous chapters, SEC techniques have been successfully applied to tolerate errors in hardware. In this chapter, we provide a theoretical analysis for algorithmic noise tolerance (ANT).

In the design of an ANT-based system, the threshold $\tau$ is an important design parameter that was previously chosen empirically. In this analysis, we establish a link between ANT and the Bayesian detection and estimation framework. We show that ANT can be viewed as a two-step process, where detection of an error event is performed, followed by estimation of the correct value. The detection stage is a threshold detector, while the estimation stage is an approximation to the minimum mean square error (MMSE) estimator. We further derive expressions for determining the Bayesian optimal threshold $\tau^*$, and verify it with Monte Carlo simulations. Thus, this analysis proves that ANT has a strong Bayesian foundation. Furthermore, this dissertation opens up the possibility of establishing a similar basis for other SEC techniques.

The remainder of the chapter is organized as follows. In Section 6.1 we formulate the detection and estimation problem, then derive the optimal decision rule under the Bayesian estimation framework. Section 6.2 compares the performance of ANT and the optimal decision rule via a simple example and a 2D-DCT application. Section 6.3 summarizes the chapter.

6.1 Analysis of Algorithmic Noise Tolerance

In this section, we formulate a general detection and estimation problem with two observations. The optimal detection and estimation rule is derived and shows that ANT is a
low-complexity approximation of this optimal solution. A description of ANT is provided in Section 1.2.4, where it was shown that the final or corrected output $\hat{y}$ is obtained via the following decision rule:

$$\hat{y} = \theta_{ANT}(y) = \begin{cases} y_a, & \text{if } |y_a - y_e| < \tau \\ y_e, & \text{otherwise} \end{cases}$$  \hspace{1cm} (6.1)$$

where $y = (y_a, y_e)$ is the observation vector (see Fig. 6.1), and $\tau$ is an application-dependent parameter chosen empirically to maximize the performance (MMSE, probability of correct operation $P_{\text{correct}}$) of ANT.

6.1.1 A Bayesian formulation of ANT

The detector assumes binary hypotheses: (1) the error-free event $H_0$ (i.e., $\eta = 0$), and (2) the error-event $H_1$ (i.e., $\eta \neq 0$). The goal is to determine the optimal decision rule $\delta(y|P_{\eta}, P_e)$, which chooses one hypothesis based on the observation $y$, and the error statistics $P_{\eta}$ and $P_e$. Then estimation is performed via an estimation rule $\hat{y} = \theta(y)$, which finds the best estimate of $y_o$ based on a specific optimization criteria. This is different from a classical detection and estimation problem, as the hypotheses are not based on $y_o$, but instead on $\eta$, and that the estimation step utilizes the detection stage information to simplify the estimation process. This distinction turns out to be crucial in enabling a simplification of the Bayesian error compensation scheme leading to ANT.

The following assumptions will be made. The error-free main block output $y_o$ and the errors $\eta$ and $\epsilon$ are a fixed-point number with bit width $N$ and defined as a random variable over the set $S = \{-2^{N-1}, \ldots, 2^{N-1} - 1\}$. Setting the bit width of the output as $N$, the cardinality of $S$ is $2^N$. Furthermore, $y_o$ is assumed to be uniformly distributed and thus $p(y_o) = \frac{1}{2^N}$. The errors $\eta$ and $\epsilon$ are assumed to be independent. We will further assume that $\eta$ is zero mean and $P_{\eta}$ has a peak at zero that represents the probability of no errors. Further, $P_{\eta}$ is symmetric about the mean, and $P_{\eta}(\alpha) \geq P_{\eta}(\beta)$ for $|\alpha| \geq |\beta|, \beta \neq 0$. On the other hand, $\epsilon$ is assumed to be zero mean, and $P_e$ is symmetric about the mean, and $P_{\epsilon}(\alpha) \leq P_{\epsilon}(\beta)$.
Figure 6.1: The Bayesian framework for ANT.

Figure 6.2: Example error PMFs: (a) depiction of $P_\eta$ and $P_e$ that increase or decrease in distance from the mean, and (b) voltage overscaling (VOS) induced timing errors.

for $|\alpha| \geq |\beta|$, as depicted in Fig. 6.2(a). Large magnitude values of $\epsilon$ are assumed to be extremely unlikely. These assumptions are motivated by noting that timing-induced hardware errors are large in magnitude due to LSB-first computation, and estimation errors are Gaussian (subsampling estimators) or uniform distributed (reduced precision estimators).

An example conditional PMF of timing errors, conditioned on the error event induced by voltage overscaling (VOS), is given in Fig. 6.2(b). This error statistic was obtained through Verilog simulations of a 16-bit ripple carry adder. It can be seen that large-magnitude errors occur with large probability, which justifies our assumption for $P_\eta$. 
6.1.2 Detection

We perform Bayesian hypothesis testing to detect the error event. For equally likely priors, the decision rule is given by comparing the likelihoods [109]:

\[ p(H_1|y) \gtrsim p(H_0|y) \quad (6.2) \]

The likelihoods are derived as follows:

\[
p(H_0|y) = \frac{p(H_0 \cap y)}{p(y)} = \frac{\frac{1}{2^N} P_\eta(0) P_e(y_e - y_a)}{p(y)} \quad (6.3)
\]

where the second equality comes from the fact that all possible values of \( y_a \) are equally likely, and \( P_\eta \) and \( P_e \) are independent. Also, to have the no error event, i.e., \( \eta = 0 \), when \( y = (y_a, y_e) \) is observed, we need \( \eta = 0 \) and \( e = y_e - y_a \). Likewise,

\[
p(H_1|y) = \frac{p(y) - \frac{1}{2^N} P_\eta(0) P_e(y_e - y_a)}{p(y)} \quad (6.4)
\]

where once again the independence assumption between \( P_\eta \) and \( P_e \) has been used.

The posterior probability \( p(y) \) is the sum of the probability of all possibilities to observe \( y \). Thus, it is given by

\[
p(y) = \sum_{y_o} p(y_o) P_\eta(y_a - y_o) P_e(y_e - y_o) = \frac{1}{2^N} \sum_{\eta} P_\eta(\eta) P_e(\eta + (y_e - y_a)) \quad (6.5)
\]

Defining the likelihood ratio \( \mathcal{L} \) by taking the ratio of (6.4) to (6.3), \( \mathcal{L} = \frac{p(H_1|y)}{p(H_0|y)} \), the optimal
Bayesian detection rule $\delta(y)$ is now denoted as

$$\delta(y) = \begin{cases} H_1 & \text{if } L \geq 1 \\ H_0 & \text{if } L < 1 \end{cases} \quad (6.6)$$

From (6.3) and (6.5), it can be seen that $L$ only depends on the difference of the observed outputs $d = y_a - y_e$. Furthermore, given that $P_e$ is decreasing in distance from the mean and symmetric, $P_\eta(0)P_e(y_e - y_a)$ is a decreasing function in $|d|$. Also, $p(y)$ is symmetric, and convex in the half-plane, and given that $P_e$ is heavily concentrated around the mean, the minimum of $p(y)$ is close to 0. Thus we can approximate $p(y)$ as an increasing function in $|d|$. Now we can rewrite the decision rule as

$$\delta(y) = \begin{cases} H_1 & \text{if } |d| \geq \tau \\ H_0 & \text{if } |d| < \tau \end{cases} \quad (6.7)$$

where the theoretically optimal threshold $\tau_{a,p}^*$ is the value of $|d|$ when $L=1$, and satisfies

$$\sum_\eta P_\eta(\eta)P_e(\eta + \tau_{a,p}^*) = 2P_\eta(0)P_e(\tau_{a,p}^*) \quad (6.8)$$

We can see that the resulting optimal decision rule (6.7) is equivalent to the detection rule (6.1) used in ANT. Furthermore, we have shown that the optimal threshold $\tau_{a,p}^*$ to be used in ANT is given by (6.8).

6.1.3 Estimation

When $H_0$ is detected ($\eta = 0$), the main block output is used as the corrected output i.e., $\hat{y} = y_a$. When $H_1$ is detected, a more complex estimation needs to take place. There are many different optimality conditions available for estimation such as minimum mean squared error (MMSE), minimum mean absolute error (MMAE), and maximum a posteriori probability (MAP). In this analysis, we will focus on MMSE.

The optimization criterion will be to minimize $E(\hat{y} - y_o)^2$ given $y$ and that $H_1$ has been
detected where the expectation is over \( y_o \). It is well known that the MMSE estimator is the conditional mean, i.e., the estimate \( \hat{y}(y) = E(y_o|y, H_1) \) [109]. As the posterior distribution of \( y_o \) is

\[
p(y_o|y, H_1) = \frac{1}{1 - p_\eta} p_\eta(y_a - y_o) P_e(y_e - y_o)
\]  

the MMSE optimal estimator will be

\[
\hat{y} = \theta(y) = \begin{cases} 
y_a & \text{if } \delta(y) = H_0 \\
\sum_{y_o \neq y_a} p(y_o|y, H_1) y_o & \text{if } \delta(y) = H_1
\end{cases}
\]

where \( y_a \) is excluded in the summation as it corresponds to \( H_o \). Determining \( \hat{y} \) is a complex and power hungry task. In ANT, an approximation to the optimal estimator has been made, such that \( \hat{y} = y_e \). This estimator can be implemented by a simple mux with the output of the detection stage used as the control. In Section 6.2, we will show that such approximation results in minimal performance degradation.

### 6.2 Comparison of Simulation and Analysis

In this section, we will first apply ANT on a simple example and compare the results with analysis. For this example, all signals are considered to be eight bits \((N = 8)\). The error PMFs \( P_\eta \) and \( P_e \) are constructed as follows. \( P_e \) is a truncated discrete Laplace distribution with parameter \( a = 0.9 \) normalized by a constant \( A \). \( P_\eta \) is a circularly shifted version of \( P_e \) scaled with the probability of error \( p_\eta = 0.1 \), and a peak of \( 1 - p_\eta \) at 0. The exact forms of the error PMFs \( P_\eta \) and \( P_e \) are given below:

\[
P_e(k) = A \frac{1 - a}{1 + a} a^{|k|}, -128 \leq k \leq 127
\]

\[
P_\eta(k) = \begin{cases} 
1 - p_\eta & \text{if } k = 0 \\
p_\eta P_e \left( (k + 2^{N-1})_{2^N} \right) & \text{if } k \neq 0
\end{cases}
\]

where \((\cdot)_{2^N}\) represents the truncation operation where the \( N \) LSBs are obtained. The PMF is depicted in Fig. 6.3(b), where the peak at \( P_\eta(0) \) has been clipped for better visibility.
Figure 6.3: Simulation results for a simple example: (a) probability of correct detection and MSE vs. $\tau$, (b) error PMF, and (c) $\tau_{s,p}^*$ and $\tau_{a,p}^*$ for different PMFs.
The performance of ANT with varying thresholds is shown in Fig. 6.3(a). The design of an ANT system would go through such simulations to obtain the optimal threshold. For this example, Monte Carlo simulations show that the probability of correct detection is maximum at \( \tau_{s,p}^* = 70 \), with \( P_{\text{det}} = 0.9986 \), which is very close to \( \tau_{a,p}^* = 68 \), the theoretical optimum obtained from (6.8). Similarly, simulations indicate that the MSE is minimized at \( \tau_{s,m}^* = 65 \), with MSE = 27.21, which is also very close to the MSE = 25.19 obtained via the optimal estimator in (6.10). Furthermore, the utility of our analysis in design can be seen by the fact that the MSE achieved via simulations with \( \tau = \tau_{a,p}^* \) is 33.55, which corresponds to a difference of 0.4% when normalized with respect to the maximum MSE obtained when \( \tau = 128 \). Note that the detection probability and MSE curves are flat around the optimal point. This shows that the exact value of the optimal threshold is unnecessary to achieve near-optimal performance, and that both \( \tau_{a,p}^* \) and \( \tau_{s,p}^* \) are good approximations for \( \tau_{s,m}^* \).

Similar results (see Fig. 6.3(c)) were obtained with different forms for \( P_\eta \) and \( P_e \), as long as they met the assumptions outlined in Section 3. For the total of 20 cases shown in Fig. 6.3(c), the maximum difference in \( \tau_{s,p}^* \) and \( \tau_{a,p}^* \) was 11, while on average the difference was 6.6. In terms of MSE, this corresponds to a maximum degradation of 2.1%, with an average degradation of 0.6% when normalized to the maximum MSE.

We have also applied ANT to a 2D-DCT image compression application. Figure (6.4) depicts the simulation setup. For ANT, the main block is an 8-bit input, 8-bit output, 8 \( \times \) 8 2-D DCT block followed by a quantizer using Chen’s algorithm [110], with mirror adders and array multipliers [111] as fundamental building blocks, implemented in a commercial 45 nm, 1.2 V CMOS process. \( P_\eta \) is characterized at various different voltages through delay based Verilog simulations, with two examples shown in Fig. 6.5(a). \( P_\eta \) has a few large amplitude errors that have a high probability of occurrence, which follows our assumptions for \( \eta \) but not fully. Only the main 2D-DCT block is subject to voltage overscaling (VOS), and hence is the only block that exhibits errors. The estimator is a reduced precision version of the main block. Figure 6.5(b) shows the simulation result at \( V_{dd} = 1V \). Simulations show \( \tau_{s,p}^* = 55 \), while \( \tau_{s,m}^* = 57 \). Using (6.8), \( \tau_{a,p}^* = 65 \). We can see that even though \( P_\eta \) does not satisfy the assumptions in Section 6.1, the values are similar. In this case as well, the detection probability exhibits a very flat behavior around \( \tau_p^* \) and shows that ANT is robust to the
value of the detection threshold.

6.3 Summary

In this chapter we have provided a statistical analysis of ANT to aid the design of error-resilient DSP systems. This is a first attempt in providing an analytical basis for designing and verifying the performance of such systems. Analysis gives insight to the significant robustness enhancement that ANT provides. The simple thresholding based detection is a good approximation to the optimal Bayesian rule for detecting HW errors. Furthermore, the performance of ANT is not sensitive to the threshold value, which suggests that a single threshold value can be used without calibration for multiple instances of a design. This analysis methodology can be generalized to include other statistical error compensation techniques.
Figure 6.5: DCT example: (a) error statistics of a voltage overscaled DCT block at $V_{dd} = 1.1V$ ($p_\eta = 0.0043$), and $V_{dd} = 1.0V$ ($p_\eta = 0.0374$), and (b) the resulting probability of correct detection and MSE vs. $\tau$. 
Machine learning applications have become prevalent. Energy-efficient ML accelerators will be a critical enabler for deploying ML applications not only in the mobile space but also in the high-performance computing space due to the thermal bottleneck. In this dissertation the statistical nature of DSP and ML applications is combined with the stochasticity of deeply scaled process technology through statistical error compensation to enhance error resiliency and energy efficiency.

7.1 Dissertation Contributions

In the past, SEC techniques have been applied to a wide range of applications in communication and DSP systems. In this dissertation, we extended the application of SEC to include machine learning applications. Specifically, we showed that when SEC is applied to applications that possess inherent robustness, its effectiveness is significantly enhanced.

First PN code acquisition, a detection based application widely used in communication systems, was chosen. An SSNOC based PN acquisition filter was implemented in 180nm CMOS and its performance and energy consumption was measured. Results show a detection rate higher than 90% at a false alarm rate of 10% is maintained at aggressive VOS of a supply voltage 28.4% below the critical voltage, resulting in error rates larger than 85.83%, with a corresponding energy savings of 2.52×.

Then SEC was applied to a belief propagation based communication system, a decoder for low-density parity check (LDPC) codes. Combined with the inherent resilience of LDPC codes, the SEC based LDPC decoder can operate at a supply voltage up to 38% lower than the nominal voltage and tolerate up to 30× more errors over an SNR range of 3dB to
8 dB, while maintaining less than 3\times degradation in BER. This is equivalent with energy savings of 45.7\% compared to conventional LDPC decoders, and 33.2\% compared to a sign bit protected LDPC decoder.

SEC was applied to a Markov random field based iterative stereo matching architecture. Building upon the SEC based LDPC decoder, SEC was applied to a more complex ML application with a larger graph and larger dimension. Error compensation is performed at several levels including the arithmetic, iteration, and system level. The compensation overhead, robustness, and energy savings are characterized and compared among the different levels of compensation. Arithmetic compensation achieves power savings of 39.6\% at an overhead of 97.4\%. A hybrid approach can successfully trade off the compensation complexity and energy savings by achieving 16.1\% additional power savings compared to arithmetic level while reducing the overhead to 57.9\%.

A study on combining SEC and approximate computing has been performed to show that SEC can further extend the AC based design to achieve additional robustness and energy efficiency. Results show that ANT combined with AC achieves energy savings of 44.7\% compared to a conventional system, while achieving at most 4\% degradation in performance. This supports our view that embracing the stochasticity of the underlying process is essential to achieve significant energy savings.

Despite the successful design of SEC based communication and ML accelerators, SEC design is mostly done in a non-systematic ad hoc manner. To develop theory and design guides, attempts to analyze SEC techniques have been made. An analysis framework has been proposed and under this framework, ANT was shown to be an approximation to the Bayesian optimal detector and estimator. Furthermore, the performance of ANT is not sensitive to the threshold value, which suggests that a single threshold value can be used without calibration for multiple instances of a design.

7.2 Future Work

Design of DSP and ML systems based on SEC creates a big paradigm shift along with significant increase in error resiliency and energy efficiency. To fully embrace this paradigm
shift and appreciate the potential benefits SEC can provide, several interesting challenges exist that should be explored. To solve these problems, an interdisciplinary research effort is needed, with collaboration in various research areas including all levels of the design hierarchy.

7.2.1 SEC for ML kernels

Machine learning is a very broad field. In this dissertation, we have applied SEC to one specific class of ML applications, probabilistic graphical models (PGMs). LDPC decoders are based on message passing, while the TRW-S stereo image matching architecture is based on belief propagation. As PGMs perform inference in an iterative manner, it possesses significant inherent resiliency, which we have successfully exploited. However, the benefits of SEC are not limited to PGMs. Exploration of SEC applied to other class of ML applications should be carefully done. As ML applications, in general, possess significant inherent error resiliency, one can expect when SEC is applied, similar results as we have seen for PGMs will be observed. Many classes such as traditional MAP estimation, expectation maximization (EM), linear discriminants, and principal component analysis (PCA) can all benefit from SEC to varying degrees.

One other class of ML that is of particular interest is applications that are implemented in two steps: training and classification. The classification step gives rise to inherent resiliency. As long as the computed value falls within the same classification boundary, hardware errors will not affect system performance. Support vector machines (SVMs) [112] are one example successfully used in many classification systems. SVMs extract features from the input data, then map the features to a high-dimensional space, which is divided via decision boundaries. There already exists work on energy-efficient and error-resilient SVM kernels using error-aware decision boundaries at the classification stage to combat against hardware errors occurring at the feature extraction stage [113]. Many structures based on neural networks and deep belief networks [114] including convolutional nets [115] are also based on the training-classification principle, and possess the potential for significant energy efficiency and robustness via SEC.
7.2.2 SEC techniques augmented with circuit- or logic-level techniques

In this dissertation, SEC was applied at the algorithm or microarchitecture level, and utilized system- or application-level tolerance to achieve its benefits. Studies on combining SEC with circuit- or logic-level techniques can be one research direction. The manufacturing processes of post-silicon devices are still immature. Implementations that rely on post-silicon devices, and even deeply scaled CMOS, have a very low yield due to the defects introduced within the manufacturing process. For such defective processes, compensation closer to the physical layer may be required, and circuit- or logic-level compensation techniques combined with algorithmic compensation techniques can provide effective solutions to the manufacturing challenge and enable the design of such systems.

7.2.3 Error-aware design methodology

In all our work, SEC design was performed in a custom manner that required deep knowledge of systems and architecture as well as SEC techniques. To enable the general designer to implement SEC based designs, an error-aware design methodology needs to be developed. It is essential that this methodology incorporates system-level information such as a target error rate or error PDF shape that the application can tolerate. Based on this methodology, computer aided design (CAD) tools can be developed that truly automate the design of SEC based systems. Similar design flows have been proposed in the work of approximate logic level synthesis [40–42]. Synthesis methodology targeted for processors that allow voltage or reliability trade-offs [116] or graceful degradation [117] is another effort in this direction.

7.2.4 Verification and testing

Verification and testing is one big challenge in designing stochastic systems. As SEC inherently causes the outputs to be stochastic, new methods to verify the correctness of the design and test its functional correctness need to be developed. The goal for any component in a system is to enable the system to perform useful tasks, and thus the testing procedure should also be aware of system-level information. Defining new system-level metrics that
faithfully represent the validity of the HW implementation will be an interesting problem to explore. In the context of AC, Liang has proposed the use of new metrics in [105], but it is essential that application-level information is incorporated as well.

7.2.5 Theoretical foundations for SEC

Last but not least, theoretical foundations for stochastic design principles need to be established. Though SEC is a communication- and Shannon-inspired technique, there is no analogue to information theory in the computing domain. An analogue to the channel capacity theorem is essential in the computing domain to find fundamental limits on the amount of error resiliency or energy efficiency one can achieve. This limit will enable us to gauge how well our current systems are designed, and provide us a means to keep exploring better designs. To this end, an understanding of existing SEC techniques is important to enable intelligent and efficient design choices.
References


121


[34] Z. Yang, A. Jain, J. Liang, J. Han, and F. Lombardi, “Approximate XOR/XNOR-based adders for inexact computing,” in *IEEE International Conference on Nanotechnology (NANO)*, 2013, pp. 690–693.


