Design of Low Power VLSI Systems
Powered by Ambient Mechanical Vibration

by

Rajeevan Amirtharajah

Bachelor of Science in Electrical Engineering
Massachusetts Institute of Technology, May 1994

Master of Engineering in Electrical Engineering and Computer Science
Massachusetts Institute of Technology, May 1994

Submitted to the Department of Electrical Engineering and Computer Science in Partial
Fulfillment of the Requirements for the Degree of

Doctor of Philosophy in Electrical Engineering and Computer Science

at the

Massachusetts Institute of Technology

June 1999

© 1999 Massachusetts Institute of Technology. All rights reserved.

Signature of Author

______________________________

Department of Electrical Engineering and Computer Science

May 21, 1999

Certified by

______________________________

Anantha Chandrakasan, Ph.D.
Associate Professor of Electrical Engineering
Thesis Supervisor

Accepted by

______________________________

Arthur Clarke Smith, Ph.D.
Professor of Electrical Engineering
Graduate Officer
Design of Low Power VLSI Systems
Powered by Ambient Mechanical Vibration

by

Rajeevan Amirtharajah

Submitted to the Department of Electrical Engineering and Computer Science on May 21, 1999 in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Electrical Engineering and Computer Science

Abstract

Low power design trends raise the possibility of using ambient energy to power future digital systems. This thesis explores the design of such systems for collecting and processing data from sensors. The low throughput requirements of this type of computation allows aggressive scaling of supply voltages and enables very low power solutions. We discuss implementations of a generator for transducing mechanical vibration to electrical energy using macroscopic and MEMS technology. A DC/DC converter chip has been designed and tested to demonstrate the feasibility of operating a digital system from power generated by vibrations in its environment. A moving coil electromagnetic transducer was used as a power generator. A single generator excitation produced 23 ms of valid DSP operation at a 500 kHz clock frequency, corresponding to 11,700 cycles. An ultra low power DSP chip has also been designed that implements a power scalable detection and classification algorithm for a biomedical sensor. This chip demonstrates appropriate architectural and circuit techniques for low to medium throughput sensor applications. It consumes 550 nW at 1.5 V with a 1 kHz clock frequency.

Thesis Supervisor: Anantha Chandrakasan
Title: Associate Professor of Electrical Engineering
Acknowledgments

"The production of ideas, contrary to the mere selection of the appropriate ideas from an available pool, cannot be achieved with great economy of energy and elegance, because it requires investigation and thought, and thinking is a wasteful process. It was natural then to concentrate on method ... and to leave the ideas to others." Elting E. Morison, Men, Machines, and Modern Times

"Remember, you’ve got a whole lifetime to squander." Sanjay Aggarwal

After several years, you come to the disheartening realization that the simple research project you embarked on to explore a few small ideas has taken on a monstrous life of its own. It has led you down blind alleys, trapped you in quagmires, forced you to climb difficult obstacles, and challenged you to solve abstruse puzzles. As Mike Bolotski once told me, you have to keep in mind that the degree is the means to an end, not an end in itself. But, the more time that goes by, the more difficult it is to hold on to that attitude. One starts to believe graduation is the end of the line when in fact it is merely a way station.

Graduate school is like a Zen monastery in some respects, and not just because the masters like to be inordinately demanding of their disciples. The idea behind the thesis is often seductively simple and elegant. It seems so easy to flesh out and verify an idea like that. Like Zen, it is like a beautiful mountain shrouded in mist. From afar, all you see is the glorious, lofty peak nestled in the ephemeral clouds. When you get up close, you see the true essence of the mountain: hard rock. Now I can finally thank all those who helped me chip away at that rock. It was much harder than I expected.

Financial support for my work was provided through collaborative participation in the Advanced Sensors Consortium sponsored by the U.S. Army Research Laboratory under Cooperative Agreement DAAL01-96-2-0001. Further support was provided by Draper Laboratories under award numbers DL-H-505348 and DL-H-513218. I also received the generous support of a National Science Foundation graduate student fellowship.

I would like to thank my advisor, Prof. Anantha Chandrakasan, for much patience, encouragement, and advice on writing and presentations. I have benefited greatly from the example of his success. Prof. Jeff Lang provided insight into the workings of electromechanical devices, power electronics, MEMS, and constructive comments for this dissertation. Dr. Tom Knight is the person most responsible for my education at MIT. His advice, support, creativity, and vast knowledge are invaluable to me. I would like to acknowledge
the efforts of Prof. Charlie Sodini and Prof. Harry Lee in creating the research environment at the Center for Integrated Circuits and Systems at MTL. Prof. Marty Schlecht answered many questions on power electronics. The characterization and modeling of mechanical vibration were helped by Prof. Richard Lyon.

My summer at Lockheed Sanders taught me a great deal about signal processing in general and detection and classification in particular. I thank Jim Ortolf, Dr. T. S. Sun, Dr. Steve Lang, and Dr. Kathy Brown for their help and guidance.

The development of the biomedical sensor was driven by Mike Scanlon at the Army Research Labs and his efforts are greatly appreciated. Neel Karkhanis assisted in the taking of physiological data. Steve Ambrose of Tritek, Inc. provided invaluable assistance in transferring that data. A moving-coil generator prototype was provided by Mark Probst at Army Research Labs and I thank him for his help.

The MEMS vibration to energy project would not exist without the aid of Tony Kourepennis and Kevin Toomey at Draper Lab. Paul Rosenstrach provided the signal processing requirements and vibration spectrum for the gas turbine generator.

Former lab mate Tom Barber went above and beyond the call of duty in the tapeout of the Sensor DSP chip, which would not have worked at all without the suggestions and support of my fellow graduate students. I am honored to have been associated with such a talented group of people and their contributions to this thesis, my education, and my sanity are immeasurable. Duke Xanthopoulos helped greatly in many ways on my quest to graduate as well as being my constant dinner companion during late-night chip design sessions. Thankfully we can now both spend more time with our girlfriends than with each other. Tom Simon, our resident genius, provided more circuit insight, sanity checks, sardonic wit, and moral support than I can say. Every interesting circuit in this dissertation is in some way his. Jim Goodman, always cheerful, was a bright spot in the lab during the darkest days, as well as having the patience to listen to and help me evaluate my crackpot ideas. I appreciate his good humor, patience, and sharp intellect. Scott Meninger contributed all of the detailed MEMS transducer power electronics and was very supportive, as well as being a great guy to hang out with. Abram Dancy spent many hours working on control theory and power electronics with me, as well as discussing the feasibility of energy harvesting. His lively family provided many welcome distractions. Vadim Gutnik contributed to many intellectual discussions as well as pioneering work on performance feedback. His system administration efforts were always appreciated. I am grateful to Wendi Rabiner for discussions on signal processing and many other topics. Numerous others involved in our group help make it a fun place to work: Rex Min, Debashis Saha, Oscar Mur-Miranda, James Kao, Seong-Hwan Cho, Josh Bretz, Amit Sinha, Alice Wang, Gangadhar Konduri, and Siva Narendra.

My early training as a UROP at the AI Lab was essential to my learning the nuts and bolts of chip design. I would like to thank Mike Bolotski from the bottom of my heart for the outstanding effort he made in creating a usable CAD environment. His sage advice has helped me make sense of the graduate school experience. Carlin Vieri, Matt Becker, and many others contributed to the fun and stimulating environment I enjoyed at AI. At MTL I would like to acknowledge Don Hitko, Dan McMahill, Mike Perrott, Jim MacArthur, Nate
Shnidman and other denizens of 38-107 for their friendship and help. Jeff Gealow, Kush Gulati, and Iliana Fujimori helped with my first MOSIS fab. Nothing would get done around here without our administrative assistants Beth Chung and Margaret Flaherty. In the EECS department, Lourenco Pires assisted me with some magnet measurements.

Oscar Yeh provided help with the construction of the first moving-coil generator prototype. I thank him for his friendship and support over these many years. Sekhar Tatikonda shared many lunches and research discussions with me and commiserated constantly. Since our undergrad days, he has been a constant source of intellectual stimulation and I appreciate his advice, perspective, and friendship. Raj Surati also slugged it out in the trenches with me and I thank him for his companionship.

Although some of them didn’t see me very much, I appreciate the warmth and support of my various roommates over the years: Aaron Woolsey, Darius Sadri, Gene Schuster, Robin Kullberg, Sean Scarlett, Lloyd D’Souza, and Mon-Fen Hong. Without the conversation, barbecues, and beer shared with my many friends outside the lab I would not have escaped with my sanity mostly intact. Thanks to Chris Adler, Pete Mao, Will Gump, Jimmy Ko, Denise Nickerson, Sanjay Aggarwal, John Baur, Brian Rider, Dan Grodner, Darrin Jewell, Srikar Srinath, Molly Hennen, Dave Mundy, Abe Stroock, Darcy Duke, and Ryan Evans. If I forgot you, I apologize profusely and I promise to buy you a beer.

My parents Uma and Appiah Amirtharajah were always there for me, with encouragement, support, friendship, and love. I couldn’t have done it without them and I can’t repay what I owe them. In many ways, this thesis is theirs as much as mine; I hope they don’t hold that against me. My sister Mohana Amirtharajah gave me love and a strong friendship; her humor brought me much joy and I hope I can be as much help to her as she becomes the third Dr. Amirtharajah.

My love, Amy Chen, brought me out of my work and into an appreciation of what is really important in life. Her love and support are an invaluable source of strength and comfort for me. I hope I can make up for the lost time someday. I am truly blessed to have her share my life. Thanks Amy. I love you!
Contents

1 Introduction ........................................... 23
  1.1 Previous Work ........................................ 24
  1.1.1 Compact Power Sources .......................... 25
  1.1.2 Low Power VLSI Design ......................... 26
  1.2 Research Issues ....................................... 26
  1.3 Architecture Level Power Estimate for Specialized FFT Processor ............ 28
  1.3.1 CMOS Dynamic Power Dissipation and Modeling ....................... 28
  1.3.2 Low Power FFT Processor .......................... 29
  1.3.3 Summary ............................................ 32
  1.4 Overview .............................................. 34

2 Stored vs. Ambient Energy Technologies ...................... 37
  2.1 Performance Metrics for Energy Use in Portable Electronics ............... 38
  2.2 Stored Energy Sources .................................. 39
  2.2.1 Batteries ........................................... 39
  2.2.2 Combustion of Fuels ............................... 40
  2.2.3 Electrical Storage Mechanisms ...................... 40
  2.3 Ambient Energy Sources ................................ 41
  2.3.1 Solar Power ........................................ 41
  2.3.2 Thermal Power ...................................... 41
  2.3.3 Electromagnetic Fields ........................... 42
  2.3.4 Fluid Flow .......................................... 42
  2.3.5 Mechanical Vibration .............................. 43
  2.3.6 Other Mechanical Energy Sources .................... 43
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.2.7</td>
<td>Continuous Time Analysis</td>
<td>84</td>
</tr>
<tr>
<td>4.2.8</td>
<td>Digital Implementation</td>
<td>85</td>
</tr>
<tr>
<td>4.2.9</td>
<td>Low Resolution Design Considerations</td>
<td>86</td>
</tr>
<tr>
<td>4.2.10</td>
<td>High Resolution Digital Feedback</td>
<td>89</td>
</tr>
<tr>
<td>4.3</td>
<td>Controller Design</td>
<td>89</td>
</tr>
<tr>
<td>4.3.1</td>
<td>Frequency Comparator and Integrator</td>
<td>90</td>
</tr>
<tr>
<td>4.3.2</td>
<td>Duty Cycle Generator</td>
<td>93</td>
</tr>
<tr>
<td>4.3.3</td>
<td>Sample Rate</td>
<td>94</td>
</tr>
<tr>
<td>4.4</td>
<td>Datapath Load Circuit</td>
<td>95</td>
</tr>
<tr>
<td>4.4.1</td>
<td>Subband Filter</td>
<td>95</td>
</tr>
<tr>
<td>4.4.2</td>
<td>Voltage Controlled Oscillator</td>
<td>96</td>
</tr>
<tr>
<td>4.5</td>
<td>Power FET Switches and Drivers</td>
<td>97</td>
</tr>
<tr>
<td>4.6</td>
<td>Bootstrapping and Other Issues</td>
<td>97</td>
</tr>
<tr>
<td>4.7</td>
<td>VLSI Implementation</td>
<td>99</td>
</tr>
<tr>
<td>4.8</td>
<td>Self-Powered DSP System: Design and Test</td>
<td>99</td>
</tr>
<tr>
<td>4.8.1</td>
<td>Control Loop Performance</td>
<td>99</td>
</tr>
<tr>
<td>4.8.2</td>
<td>Acoustic Power Generation</td>
<td>101</td>
</tr>
<tr>
<td>4.8.3</td>
<td>Test Result Summary</td>
<td>103</td>
</tr>
<tr>
<td>5</td>
<td>Energy Scalable Classifier</td>
<td>105</td>
</tr>
<tr>
<td>5.1</td>
<td>Introduction</td>
<td>105</td>
</tr>
<tr>
<td>5.2</td>
<td>Heartbeat Detection</td>
<td>105</td>
</tr>
<tr>
<td>5.2.1</td>
<td>Preprocessing</td>
<td>106</td>
</tr>
<tr>
<td>5.2.2</td>
<td>Feature Extraction</td>
<td>108</td>
</tr>
<tr>
<td>5.2.3</td>
<td>Detection Using Multivariate Gaussian Classification</td>
<td>109</td>
</tr>
<tr>
<td>5.2.4</td>
<td>Influence of Sample Rate and Quantization on Classifier Performance</td>
<td>113</td>
</tr>
<tr>
<td>5.3</td>
<td>Power Scalable Classification Using Distributed Arithmetic</td>
<td>116</td>
</tr>
<tr>
<td>5.3.1</td>
<td>Distributed Arithmetic</td>
<td>117</td>
</tr>
<tr>
<td>5.3.2</td>
<td>Successive Approximation Using Distributed Arithmetic</td>
<td>119</td>
</tr>
<tr>
<td>5.3.3</td>
<td>Implementation of Heartbeat Matched Filter</td>
<td>121</td>
</tr>
<tr>
<td>5.4</td>
<td>Breath Detection</td>
<td>122</td>
</tr>
<tr>
<td>5.4.1</td>
<td>Feature Extraction</td>
<td>122</td>
</tr>
<tr>
<td>5.4.2</td>
<td>Classification</td>
<td>124</td>
</tr>
<tr>
<td>5.5</td>
<td>Voice Stress Analysis</td>
<td>124</td>
</tr>
<tr>
<td>5.6</td>
<td>Future Work: Extrapolation to Exertion State</td>
<td>125</td>
</tr>
</tbody>
</table>
6 Sensor DSP Chip

6.1 Low Power Classifier Chip Architecture .................................................. 127
6.1.1 Linear Filter ......................................................................................... 129
6.1.2 Nonlinear/Short Linear Filter ................................................................. 130
6.1.3 Feature Extraction and Classification Microcontroller ...................... 132
6.1.4 Algorithm Workload ........................................................................... 132
6.1.5 Future Work: Integrated Power Supply Controller ......................... 133

6.2 Low Voltage Logic Circuit Design .............................................................. 134
6.2.1 Ripple Carry Adders ............................................................................. 134
6.2.2 Squarers and Multipliers ..................................................................... 135
6.2.3 Latches and Registers ......................................................................... 136
6.2.4 Read Only Memories ........................................................................... 137

6.3 Low Voltage SRAM Circuit Design ............................................................ 137
6.3.1 SRAM Sensing Scheme ....................................................................... 137
6.3.2 SRAM Timing Scheme ......................................................................... 138

6.4 SRAM Bank Partitioning .......................................................................... 140

6.5 Multiport Register File Design .................................................................. 145

6.6 Clock Design ............................................................................................. 146

6.7 JTAG Programming Interface .................................................................... 147

6.8 Chip Test Board .......................................................................................... 147

6.9 Sensor DSP Chip Test Result Summary .................................................... 148

7 Conclusions and Future Work ..................................................................... 155

7.1 Ambient Energy Sources .......................................................................... 155

7.2 Vibration to Electrical Energy Transducers .............................................. 156

7.3 Low Power DC/DC Conversion ................................................................ 157

7.4 Energy Scalable DSP Algorithms ............................................................. 157

7.5 Low Power Sensor DSP ........................................................................... 158

7.6 Energy Harvesting and Future Work ......................................................... 158

References ....................................................................................................... 161
A  Sensor DSP Microcontroller Assembly Language  167
   A.1  Instruction Set Overview ........................................ 167
   A.2  Registers and Other State ........................................ 169
   A.3  Instruction Descriptions ......................................... 170
      A.3.1  RTL Description ............................................. 170
      A.3.2  Miscellaneous Instructions ................................. 172
      A.3.3  Arithmetic Instructions ................................. 172
      A.3.4  Arithmetic Macros ........................................... 174
      A.3.5  Relational Macros ........................................... 175
      A.3.6  Logical and Shift Instructions ......................... 177
      A.3.7  Control Flow Instructions ................................ 179
      A.3.8  Memory Instructions ...................................... 180
      A.3.9  Filter Interface Instructions ............................ 182
      A.3.10  Unused Opcodes ............................................ 183
      A.3.11  Miscellaneous Reserved Words ......................... 183
   A.4  Example Program: Fibonacci Numbers ......................... 184

B  Sensor DSP Nonlinear/Short Linear Filter Assembly Language 185
   B.1  Registers .......................................................... 186
   B.2  Instruction Descriptions ....................................... 186
      B.2.1  SAC Instructions ........................................... 186
      B.2.2  MAC Instructions ........................................... 188
      B.2.3  LSU Instructions ........................................... 189

C  Chip Pinouts .......................................................... 193

D  Moving-Coil Generator Design Notes  197
   D.1  Basic Mechanical Design ....................................... 197
   D.2  Desired Generator Dynamics ................................... 198
   D.3  Other Design Notes ............................................. 199
   D.4  Testability ...................................................... 200
# List of Figures

1.1 Trends in power consumption for low to medium throughput DSP.
1.2 System block diagram.
1.3 Basic FFT butterfly.
1.4 Simple FFT processor architecture.
1.5 FFT power consumption versus time.
1.6 Storage capacitance versus voltage for the energy consumed by the FFT.

2.1 Schematic of power transfer mechanism. As the housing surrounding the mass-spring-dashpot system is vibrated, kinetic energy is imparted to the mass. The electromechanical transducer and the mechanical losses in the system are modeled by the dashpot.

2.2 Displacement shaping filter and representative time domain displacement simulation of vibration of the generator housing due to human walking.

2.3 Velocity shaping filter and representative time domain velocity simulation of vibration of the generator housing due to human walking.

2.4 Acceleration shaping filter and representative time domain acceleration simulation of vibration of the generator housing due to human walking.

2.5 Stochastic output power simulation: best case estimate of ambient power generation from vibration due to human walking. Mean output power \( \approx 400\mu W \).

2.6 Transfer functions from housing position \( y(t) \) to mass velocity \( \dot{z}(t) \) and position \( z(t) \). The damping is kept constant while the first corner frequency of the highpass characteristic varies. The high order single pole rolloff at 1000 rad/s represents unmodeled high frequency dynamics.

2.7 Transfer functions from housing position \( y(t) \) to mass velocity \( \dot{z}(t) \) and position \( z(t) \). The corner frequency is kept constant while the damping factor varies. The high order single pole rolloff at 1000 rad/s represents unmodeled high frequency dynamics.

2.8 Spectrum of Power Signal.
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Generator mechanical schematic.</td>
<td>60</td>
</tr>
<tr>
<td>3.2</td>
<td>Block diagram of traditional low frequency moving-coil transducer model.</td>
<td>61</td>
</tr>
<tr>
<td>3.3</td>
<td>Model fits of two generator configurations using the simplified linear transducer model.</td>
<td>64</td>
</tr>
<tr>
<td>3.4</td>
<td>Generator and Rectifier Circuit Schematic.</td>
<td>65</td>
</tr>
<tr>
<td>3.5</td>
<td>Measured generator output waveforms for a mechanical impulse.</td>
<td>66</td>
</tr>
<tr>
<td>3.6</td>
<td>Two types of variable capacitor for electric power transducers.</td>
<td>67</td>
</tr>
<tr>
<td>3.7</td>
<td>MEMS transducer plan view.</td>
<td>68</td>
</tr>
<tr>
<td>3.8</td>
<td>MEMS transducer cross-sectional view.</td>
<td>68</td>
</tr>
<tr>
<td>3.9</td>
<td>Energy conversion cycles.</td>
<td>69</td>
</tr>
<tr>
<td>3.10</td>
<td>Modified energy conversion with $C_{par}$.</td>
<td>72</td>
</tr>
<tr>
<td>3.11</td>
<td>Power electronics for charge-constrained conversion cycle.</td>
<td>73</td>
</tr>
<tr>
<td>3.12</td>
<td>Circuit topology in various states during energy conversion.</td>
<td>73</td>
</tr>
<tr>
<td>4.1</td>
<td>System block diagram.</td>
<td>76</td>
</tr>
<tr>
<td>4.2</td>
<td>Detailed block diagram of self-powered DSP system.</td>
<td>76</td>
</tr>
<tr>
<td>4.3</td>
<td>Block diagram of buck converter system.</td>
<td>78</td>
</tr>
<tr>
<td>4.4</td>
<td>Integrator compensation system root locus.</td>
<td>79</td>
</tr>
<tr>
<td>4.5</td>
<td>One bit error constitutive relation.</td>
<td>80</td>
</tr>
<tr>
<td>4.6</td>
<td>Nyquist plot with describing function for 1 bit error feedback.</td>
<td>82</td>
</tr>
<tr>
<td>4.7</td>
<td>CT limit cycle for 1 bit feedback.</td>
<td>83</td>
</tr>
<tr>
<td>4.8</td>
<td>Digital limit cycle for 1 bit feedback (10 bit quantization).</td>
<td>83</td>
</tr>
<tr>
<td>4.9</td>
<td>Digital limit cycle for 1 bit feedback (8 bit quantization).</td>
<td>84</td>
</tr>
<tr>
<td>4.10</td>
<td>One bit error with dead zone constitutive relation.</td>
<td>85</td>
</tr>
<tr>
<td>4.11</td>
<td>One bit error with dead zone describing function.</td>
<td>86</td>
</tr>
<tr>
<td>4.12</td>
<td>Nyquist plot with describing function for 1 bit error with dead zone feedback.</td>
<td>87</td>
</tr>
<tr>
<td>4.13</td>
<td>CT output with 1 bit error feedback with large dead zone.</td>
<td>87</td>
</tr>
<tr>
<td>4.14</td>
<td>CT output with 1 bit error feedback with small dead zone.</td>
<td>88</td>
</tr>
<tr>
<td>4.15</td>
<td>Digital output with 1 bit error feedback with small dead zone.</td>
<td>88</td>
</tr>
<tr>
<td>4.16</td>
<td>Digital output with 1 bit error feedback with large dead zone.</td>
<td>89</td>
</tr>
<tr>
<td>4.17</td>
<td>Digital controller block diagram.</td>
<td>90</td>
</tr>
<tr>
<td>4.18</td>
<td>Frequency comparator block diagram. The high order bit of the 2 bit counter clocked by $f_{clk}$.</td>
<td>90</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

4.19 Frequency Comparator Timing Diagram ........................................ 91
4.20 Low power digital PWM controller architecture. .......................... 94
4.21 Fast clocked counter PWM waveform generator. The duty cycle d[k] is digitally compared to the counter output x[k] and the comparator output is used as the PWM waveform. ........................................ 95
4.22 Discrete-time FIR subband filter, impulse and frequency response. .... 96
4.23 Discrete-time subband filter block diagram. ................................. 97
4.24 Subband filter and VCO layout. The VCO layout is placed near the DSP for good delay matching. Its shape is also copied from the DSP, and so interconnect delays are incorporated into the delay of the oscillator. .......... 98
4.25 Bootstrap Circuit Schematic. ..................................................... 99
4.26 Die Photo. ................................................................................. 100
4.27 Step Response. Note steady-state limit cycle due to low resolution digital control. This can be ameliorated with better deadzone width and error sample rate. ......................................................... 101
4.28 Bootstrapping. ........................................................................... 102
4.29 Acoustic power generation. Speaker S0 and microphone M0 are actually identical components. Transformer X0 is necessary to allow the signal generator to drive S0. ......................................................... 102
4.30 Voltage Regulation. ..................................................................... 104

5.1 Matched filter template. ............................................................... 106
5.2 Matched filter output without speech. ........................................... 107
5.3 Matched filter output with speech. ............................................... 108
5.4 Typical correlation peak with labeled features. ............................... 109
5.5 Probability density functions for the normalized candidate heartbeat features. 110
5.6 Recognition performance for the FDR1 feature ranking. ................. 112
5.7 Recognition performance for the MOM feature ranking. .................. 113
5.8 Recognition performance for the TER feature ranking. .................... 114
5.9 Classifier recognition performance for different sample rates, both floating point and fixed point implementations. .............................. 115
5.10 Classifier recognition performance for different quantization levels, 160 Hz sample rate. ................................................................. 116
5.11 Classifier recognition performance for different feature sets. ............. 117
5.12 Distributed Arithmetic ROM and Accumulator (RAC) Structure ....... 119
5.13 Power reduction versus input quantization (simulated using PowerMill) 121
5.14 Recognition performance tradeoffs with DA Unit power 122
5.15 Moving average energy in various high frequency bands indicating the presence of breath sounds 123
5.16 Recognition performance for breath sounds using the FDR1 feature ranking 125

6.1 Classification signal processing block diagram 128
6.2 Signal processing chip architecture 128
6.3 Linear filter implementation architecture 130
6.4 DA Unit shift register implementation 131
6.5 Short linear and nonlinear filter implementation architecture 131
6.6 Feature extraction and classification microcontroller architecture 132
6.7 Period versus power supply voltage for a 17 stage ring oscillator in 0.6 µm CMOS 134
6.8 Full adder cell schematic 136
6.9 Positive edge-triggered flip-flop with asynchronous clear schematic 137
6.10 SRAM sensing circuit 139
6.11 SRAM timing generator delay element 139
6.12 Simulated timing waveforms of an SRAM read cycle 141
6.13 Simulated SRAM read waveforms 141
6.14 Simulated timing waveforms of an SRAM write cycle 142
6.15 Simulated SRAM write waveforms 142
6.16 Simulated SRAM storage node waveforms during write 143
6.17 SRAM bank schematic for high-level power estimation 144
6.18 Simple memory floorplan for high-level power estimation 144
6.19 Optimal size of SRAM banks for various levels of predecoding 145
6.20 Register file cell for 2-read port, 1-write port register file 146
6.21 Sensor DSP chip test board block diagram 148
6.22 Measured Sensor DSP Chip Power, DA Unit and microcontroller active 149
6.23 Measured Sensor DSP Chip Power, all units active 150
6.24 Simulated power breakdown between frontend and backend processing units of the Sensor DSP chip 150
6.25 Simulated power breakdown between various functional units of the Sensor DSP chip 151
6.26  Measured SRAM timing waveforms. .................................. 152
6.27  SensorDSP chip photograph. .......................................... 153

A.1   Sensor DSP microcontroller architecture. ............................ 167

B.1   Short linear and nonlinear filter implementation architecture. .... 185

C.1   DC/DC Converter Chip Pinout ........................................ 194
C.2   Sensor DSP Chip Package Footprint ................................ 196
# List of Tables

1.1 Process characterization for energy estimation. ........................................ 29  
1.2 FFT throughput specification. ................................................................. 31  
1.3 FFT power for both types of FFT computation. ........................................ 31  
1.4 FFT power for total computation (10 FFTs). ........................................... 32  

2.1 Summary of ambient energy sources. ....................................................... 43  
2.2 Assumed parameters for a discrete transducer implementation. ............... 54  
2.3 Maximum output power for different machinery classes for a discrete transducer. ................................................................. 55  
2.4 Assumed parameters for a micromachined transducer implementation....... 55  
2.5 Maximum output power for different machinery classes for a micromachined transducer. ................................................................. 55  
2.6 Summary of stored and ambient energy sources. ..................................... 57  

4.1 Frequency comparator values and bounds on $f_v$. .................................. 93  
4.2 Test Chip Specifications. ............................................................... 104  

6.1 Sensor DSP Chip workload on both phases of heartbeat algorithm............. 133  
6.2 Sensor DSP Chip Specifications. ................................................................. 151  
6.3 Sensor DSP Chip Power versus a StrongARM implementation.................. 152  

A.1 Microcontroller instruction types. ......................................................... 168  
A.2 Microcontroller registers. ................................................................. 169  
A.3 Condition code register field specifiers for conditional jumps................. 170  
A.4 Configuration state bit specifiers for configuration instructions.............. 170  
A.5 Microcontroller instruction syntax and RTL level description.................. 171  
A.6 Arithmetic instructions and opcodes. .................................................. 172
A.7  Arithmetic instructions and opcodes. .............................. 172
A.8  Arithmetic macros and implementations. .......................... 174
A.9  Relational macros and implementations. ........................... 175
A.10 Logical and shift instructions and opcodes. ....................... 177
A.11 Control flow instructions and opcodes. ............................ 179
A.12 Memory instructions and opcodes. .................................. 181
A.13 Filter interface instructions and opcodes. ......................... 182
A.14 Configuration state specifiers and codes. ......................... 182
A.15 Unused opcodes. ...................................................... 183
A.16 Microcontroller instruction types. .................................. 184

B.1  NLSL Unit registers. .................................................. 186
B.2  SAC Unit instructions and opcodes. ................................ 187
B.3  MAC Unit instructions and opcodes. ................................ 188
B.4  LSU Unit instructions and opcodes. ................................ 190

C.1  Sensor DSP Chip Pinout .............................................. 195

D.1  Generator specifications. ............................................. 199
Chapter 1

Introduction

There has been much interest in recent years in low power VLSI design stemming from the demands of long battery life in portable systems and heat removal in larger, nonportable ones. Voltage scaling coupled with other algorithmic and architectural optimizations have allowed dramatic scaling of power consumption for a wide variety of low to medium throughput DSP applications [1]. Figure 1.1 shows the current power consumption for a variety of such DSPs, including programmable [2], custom [3], and sensor related systems [4, 5]. Projecting current power scaling trends into the future (based on deep voltage scaling and other power management techniques), we expect the power consumption to be reduced to 10’s of $\mu$W to 100’s of $\mu$V. At these low power levels, an interesting question arises: can we use ambient energy sources to power electronic systems? Ambient energy is energy that is in the environment of the system and is not stored explicitly, for example in a battery. Portable systems that depend on batteries have a limited operating life and can fail at inconvenient times, while a circuit powered by ambient sources has a potentially infinite lifetime. In long-lived systems where battery replacement is difficult, generating power from ambient sources becomes imperative. For example, in a smart structure where sensors and actuators are embedded in a bulk material (like a steel structural member) to modify its properties, access to the electronics is greatly reduced.

In this thesis, we explore the feasibility of operating a DSP system on power generated by external means. Since ambient energy sources are by definition uncontrolled, we require a mechanism for converting the energy to a form usable by digital logic. We propose a system as in Figure 1.2, consisting of a generator to create a voltage $V_{in}$, which can vary rapidly depending on the energy environment of the system, a voltage regulator to set the voltage to a desired level, $V_{dc}$, and a DSP load circuit which performs some computation. The DSP receives data $x[k]$ from a sensor and produces a result $y[k]$. The desired voltage is set using delay feedback instead of voltage feedback [6], [7], [8]. Some measure of the performance of the DSP, $f_{dp}$, based on its critical path delay is compared to a desired performance $f_{clk}$, and $V_{dc}$ is adjusted until that performance constraint is met. In a conventional fixed supply voltage scheme, the supply is at a level high enough to meet the most demanding performance required from the load circuit under worst case process and temperature conditions. However, the supply voltage is often higher than necessary.
under nominal operating conditions, and the circuit is then idle for some portion of the cycle. The delay (or performance) feedback scheme, on the other hand, compensates for temperature, process, and computational workload variations. It also allows a simple all digital implementation of the control loop.

In addition to a generator, the self-powered system requires a backup power source providing voltage $V_{bk}$. This is necessary since at startup the voltage regulator must derive its power from some source and the generator output is too uncontrolled to be used. The source could be a very small battery or a previously charged large capacitor, but it need not provide much energy since it is only used during the startup transient of the system. Another key difference is that the generator output voltage $V_{in}$ varies rapidly with time, in contrast to the slowly drooping battery voltages of conventional systems.

### 1.1 Previous Work

The research draws on two major bodies of previous research: work on compact (e.g. portable) electrical power sources and low power digital design.
1.1.1 Compact Power Sources

There are two approaches to producing power for a portable or embedded electronic system. The dominant approach today involves storing energy in a compact space and packaging the storage element with the system. Chemical batteries are the dominant technology for this. An alternative approach is to transduce ambient energy in the environment of the system and to package the transducer with the system. Solar-powered calculators are the most common example of this technique.

Stored Energy

The dry cell is the most commonly used form of chemical battery, widely used in flashlights, portable radios, etc. Since any spontaneous oxidation-reduction reaction can form the basis of a voltaic cell [9], there are a large number of battery options with different chemistry, each displaying its own tradeoffs between energy density, internal resistance, discharge characteristics, and so on [10]. Embedded or wearable sensor electronics require relatively small battery form factors and long lifetime, so lithium button cells are the most desirable solution. Work has also been done on integrating solid-state batteries with electronics using standard microfabrication techniques.

An alternative stored energy solution is to produce electricity from a stored fuel, either by combustion as in the MEMS gas turbine generator or from fuel cells [9]. Fuel cells produce electricity from a fuel through a direct chemical reaction facilitated by a catalyst, but miniature fuel cells are as yet impractical.
Power from Ambient Energy Sources

Various schemes have been proposed to eliminate the need for batteries in a portable digital system [11]. The most familiar ambient energy source is solar power, but other examples include electromagnetic fields (used in RF powered ID tags [12], inductively powered smart cards [13], or noninvasive pacemaker battery recharging [14]), thermal gradients, fluid flow, energy produced by the human body [15], and the action of gravitational fields [16]. A generator based on transducing mechanical vibrations can be enclosed to protect it from a harsh environment, it functions in a constant temperature field, and a person can activate it by shaking it. However, its moving parts imply less long-term reliability and more complex mechanical design. Applications include sensors mounted on vibrating machinery or worn on the body. Ambient acoustic energy can also be used, but as experiments show, the high field intensities required make this approach very difficult.

1.1.2 Low Power VLSI Design

There are four sources of power dissipation in digital CMOS circuits: switching (or dynamic) power, short-circuit power, leakage power, and static power [1]. These are summarized in the following equation:

\[
P_{\text{avg}} = P_{\text{switching}} + P_{\text{short-circuit}} + P_{\text{leakage}} + P_{\text{static}}
\]

\[
= \alpha C_L V_{dd} f + I_{sc} V_{dd} + I_{\text{leak}} V_{dd} + I_{\text{static}} V_{dd}
\]  

(1.1)

The switching power \( P_{\text{switching}} \) consists of the activity factor, \( \alpha \), or the average number of transitions for a circuit node per clock cycle, the load capacitance \( C_L \), switched, the voltage swing \( V \) through which the capacitance moves, and the power supply voltage \( V_{dd} \). In most cases, \( V = V_{dd} \), although there are common reduced-swing circuits like SRAM bitlines. The short-circuit power is controlled by the current that flows between \( V_{dd} \) and ground, \( I_{sc} \). This current flows in static CMOS logic when both the P pullup and N pulldown networks are on simultaneously during an output transition. \( I_{\text{leak}} \), the main component of the leakage power, is current due to leakage through reverse-biased PN junctions and subthreshold conduction of MOSFETs. Finally, static power is due to designed current sources, like biasing networks for linear amplifiers, whose current sums to \( I_{\text{static}} \). In a fully digital design with fast transients, the switching power \( P_{\text{switching}} \) dominates in Equation 1.1. Consequently, low power digital design focuses on reducing \( \alpha, C_L, V_{dd} \) (and \( V \)), and \( f \). These parameters can be reduced through algorithmic, architectural, and circuit techniques, all of which have received a great deal of attention in recent years.

1.2 Research Issues

A large amount of research has been done on low power VLSI design, less on the recovery of usable power from ambient energy. This thesis will attempt to synthesize these areas in
1.2. RESEARCH ISSUES

developing an integrated prototype self-powered system and explore application regimes where self-powered systems can be made practical. The goal of this research is to attempt a narrow, focused study of a few key concepts at all levels of system design, from the power supply to the final algorithms, and explore how decisions at one level affect issues at the other levels. It is not meant to be a definitive exploration at all levels.

Toward this goal, we will address the following system design issues:

• **Energy Storage vs. Ambient Energy Technologies**: A detailed analysis must be done to compare trends in energy and power density for battery, fuel cell, and other stored energy techniques such as power MEMS with recoverable ambient energy. The energy analysis must be compared to the future power requirements of electronic systems. This analysis will discover the tradeoffs between using stored vs. ambient energy and establish broad application ranges where it makes sense to use one, the other, or both. It should be noted that these are not mutually exclusive technologies, since self-powered techniques can extend battery life in applications where electronics power cannot be reduced due to other constraints. Energy harvesting techniques can result in infinite lifetime but the output power available is in general less than stored energy solutions. For a given volume, stored energy techniques usually result in greater power density, although in some cases (extremely fast fluid flow, for example) ambient sources may deliver more power.

• **Power Generation from Ambient Mechanical Vibration**: The thesis will focus on extracting energy from ambient mechanical vibration. We will explore techniques for designing electric power transducers for mechanical inputs and analyze their performance through simulation and experiment. This work will provide insight into what regimes of operation this particular power source will be practical.

• **Low Power Power Electronics Design**: Voltage regulation is a key feature of the self-powered system. Since the power requirements of the load and the amount of available power are substantially lower than previous work on low power power supplies, techniques for very low power consumption regulation circuits must be developed. As will be shown below, some approaches to the electromechanical generator require control circuitry for efficient power generation. Some low power approaches for these controllers will also be explored.

• **Low Power DSP Design for Sensor Applications**: Low power design involves optimization at all levels of the system. For this thesis, we intend to develop a signal processor for heartbeat detection and classification using an acoustic sensor. Low power algorithms have been developed to run on a custom low power programmable DSP chip. The architecture and circuit design of the chip will be optimized to run this algorithm, but will be flexible enough to efficiently implement other sensor-related algorithms. These share the feature of a low throughput requirement compared to multimedia DSPs and provide an opportunity to explore new low power architectures and circuit techniques.
To summarize, the research scope covers all the systems shown in Figure 1.2 except there will be no explicit design of the sensor or the backup power source. It presents an opportunity to optimize a system at every level from the power supply to the high level DSP algorithm.

1.3 Architecture Level Power Estimate for Specialized FFT Processor

As an example of the types of low and medium throughput sensor signal processing described in Figure 1.1, this section discusses a detailed high-level power estimate of an FFT processor for vibration analysis of a gas turbine generator on a ship.

The power estimation methodology is very simple: estimates of the energy for different basic logic operations and memory accesses at the bit level are summed for all the operations required to perform the FFT at the requested input data width of 12 bits, and this energy was divided by the total time required to perform them.

The low power implementation uses basically two very simple techniques: aggressive voltage scaling can be performed due to the relatively low sample rates, and clock gating can be used because the duty cycle for this operation is so low. Other tricks, like alternate circuit styles and architectural tradeoffs for low power [1] are ignored for this initial study. They can only reduce power beyond the estimates provided below.

1.3.1 CMOS Dynamic Power Dissipation and Modeling

Assuming fully static CMOS logic design for the FFT processor, dynamic power dissipation due to switched capacitance should dominate the chip power:

\[ P = \alpha CV_{dd}(\Delta V)f \]
\[ = \frac{E_{diss}}{\Delta t} \]  
\[ (1.2) \]
\[ (1.3) \]

where \( \alpha \) is an activity factor that represents the probability of a particular node switching in any given clock cycle, \( C \) is the node capacitance, \( V_{dd} \) is the power supply voltage, \( \Delta V \) is the voltage swing of the node, and \( f \) is the clock frequency. Equation 1.3 expresses the power in terms of the energy dissipated, \( E_{diss} \), over the time duration that energy is lost, \( \Delta t \). We will use Equation 1.3 rather than Equation 1.2 as the basis for our power estimate.

The power estimation algorithm uses a very simple approach. It has estimates for a number of rudimentary logic operations and values of \( E_{diss} \) for each of these. The energy values take into account the activity factors, node capacitances, and voltage swings. Then, given the number of points for the FFT, it determines the number of these rudimentary operations required to compute the FFT. The total energy is the sum of the energies for the
Table 1.1
Process characterization for energy estimation.

<table>
<thead>
<tr>
<th>Cell</th>
<th>Capacitance</th>
<th>$\Delta V$</th>
</tr>
</thead>
<tbody>
<tr>
<td>AND</td>
<td>$37 \times$ CUNIT</td>
<td>$V_{dd}$</td>
</tr>
<tr>
<td>Full Adder</td>
<td>$124 \times$ CUNIT</td>
<td>$V_{dd}$</td>
</tr>
<tr>
<td>2 Input Multiplexer</td>
<td>$46 \times$ CUNIT</td>
<td>$V_{dd}$</td>
</tr>
<tr>
<td>SRAM Bit Line Cell</td>
<td>$1 \times$ CBLC</td>
<td>$100 \text{ mV}$</td>
</tr>
<tr>
<td>SRAM Sense Amplifier</td>
<td>$144 \times$ CUNIT</td>
<td>$V_{dd}$</td>
</tr>
<tr>
<td>SRAM Word Line Cell</td>
<td>$2.27 \text{ fF} + 4 \times$ CUNIT</td>
<td>$V_{dd}$</td>
</tr>
</tbody>
</table>

Figure 1.3: Basic FFT butterfly.

rudimentary operations. Power is determined by dividing the total energy by the amount of time required.

Energy estimates for various logic operations are based on a capacitance characterization of two standard cell libraries for a 1 $\mu$m CMOS process. The unit capacitance for a minimum-sized gate or junction, CUNIT, is 1.73 fF. The unit capacitance for a bitline, CBLC, is 6.92 fF, accounting for junction and interconnect capacitance. Table 1.1 shows the energy parameters for the various cells per bit. These parameters are derived from standard cell libraries used in the design of a low power DC/DC converter chip [17] and an IDCT processor [18]. Since the cells in Table 1.1 have single bit outputs, the energy scales linearly with the output bitwidth.

All major datapath blocks for the FFT processor, including array multipliers, memory banks, etc., can be constructed from these basic cells.

1.3.2 Low Power FFT Processor

Figure 1.3 shows a flowgraph of the basic FFT computation. Two inputs (real or complex, depending on which stage of the FFT is being computed) are multiplied by two complex twiddle factors (again, depending on which stage) and summed at the end. The entire
N-point FFT can be built from this basic unit. The computation requires 2 complex multiplies and 2 complex adds. Each complex multiply requires 4 real multiplies and 2 real adds. Each complex add requires 2 real adds. Thus, the FFT butterfly computation needs 8 multiplies and 8 adds.

The great advantage of the FFT is the great reduction in the number of operations versus other schemes to compute the Discrete Fourier Transform. By slightly changing the butterfly computation, the total number of complex multiplications for an N-point FFT is [19]:

\[
\frac{N}{2} \log_2 N
\]

The number of complex adds is \(N \log_2 N\).

Figure 1.4 is a very general block diagram of an FFT processor. Four operands must be read for each butterfly operation: the two (possibly complex) inputs and the two complex twiddle factors. Two complex results must then be written back. These accesses must occur for each of the \(N \log_2 N\) butterfly operations in the entire computation.

Alternative FFT architectures will not affect the number of each type of operation, as long as array multipliers are used in the datapath. What can affected through parallelism or pipelining is the value of the supply voltage \(V_{dd}\) and the time \(\Delta t\) required to perform the FFT. The very simplest architecture is to assume a flat memory hierarchy (i.e. no caching) and a fully serial computation: every complex operation requires one read-evaluate-write memory cycle. What this means is that the total time required to perform the computation in cycles is large. This is feasible in this case because the duty cycle is low and the computation can be spread out in time. This reduces the power by having the processor on
1.3. ARCHITECTURE LEVEL POWER ESTIMATE FOR SPECIALIZED FFT PROCESSOR

<table>
<thead>
<tr>
<th>Computation</th>
<th>FFT Size (N)</th>
<th>Sample Rate</th>
<th>Averaging Duration</th>
<th>Overlap</th>
<th>Number of N-point FFTs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low BW FFT</td>
<td>512</td>
<td>1.8 kHz</td>
<td>2 s</td>
<td>256</td>
<td>13</td>
</tr>
<tr>
<td>High BW FFT</td>
<td>512</td>
<td>18 kHz</td>
<td>2 s</td>
<td>256</td>
<td>139</td>
</tr>
</tbody>
</table>

Table 1.2
FFT throughput specification.

<table>
<thead>
<tr>
<th>Computation</th>
<th>Operations</th>
<th>(E_{\text{dis}})</th>
<th>Delay</th>
<th>Power</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low BW FFT</td>
<td>542464</td>
<td>2.5 (\mu)J</td>
<td>76 s</td>
<td>340 nW</td>
</tr>
<tr>
<td>High BW FFT</td>
<td>5800192</td>
<td>270 (\mu)J</td>
<td>81 s</td>
<td>3.4 (\mu)W</td>
</tr>
</tbody>
</table>

Table 1.3
FFT power for both types of FFT computation.

longer. However, by parallelizing it one can reduce the voltage, and allow the computation to fill up the same time and reduce the power even further. This very serial approach is thus somewhat of a worst-case estimate for the FFT processor architecture.

The final constraint necessary to perform the power estimation is the throughput specification. One FFT computation is to be performed every 5 minutes. Of every 10 FFTs, 9 are low bandwidth computations while 1 is a high bandwidth computation. The specifications for both, including the spectral averaging, are shown in Table 1.2.

The throughput also sets the clock rate. Five minutes is a very long time to perform the computations required, even if the serial approach described above is used. To be conservative on the timing, we assume a clock rate for each computation equal to the sample rate. The clock is gated during the time that the processor is not required to compute, reducing the power linearly with the duty cycle. Thus, even though the number of cycles has been maximized, there is still plenty of “downtime” for the FFT processor since the throughput requirement is so low. The low duty cycle implies that leakage power would become significant, especially for a low threshold voltage process similar to the one used for the StrongARM microprocessor [20]. This can be translated into greater power reduction through parallelism, clock rate reduction, and further voltage scaling. Since the cycle time is fairly long (55.6 \(\mu\)s) to perform a memory read, a complex multiply consisting of two 12 bit real multiplies in parallel, and a memory write, \(V_{dd}\) can easily be scaled to 1 Volt even in an old process technology.

Table 1.3 summarizes the power consumption for each type of FFT computation. The power corresponds to the peak power dissipation during the computations. This sets the current specification that the power electronics must be able to deliver.

Table 1.4 summarizes the power consumption for the entire cycle of 10 FFT computations. The average power is very low due to the extremely low duty cycle of the throughput specification.

Figure 1.5 shows a plot of the processor power consumption versus time. The impact
Table 1.4
FFT power for total computation (10 FFTs).

<table>
<thead>
<tr>
<th>Total Operations</th>
<th>Percentage Arithmetic</th>
<th>Percentage Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>10682368</td>
<td>75 %</td>
<td>25 %</td>
</tr>
<tr>
<td>Total Energy</td>
<td>500 µJ</td>
<td>28 %</td>
</tr>
<tr>
<td>Total Delay</td>
<td>3000 s</td>
<td>66 %</td>
</tr>
<tr>
<td>Average Power</td>
<td>170 nW</td>
<td></td>
</tr>
</tbody>
</table>

Figure 1.5: FFT power consumption versus time.

of the low duty cycle requirements of the computation can clearly be seen: the average power is much lower than even the power for the slow data rate computation.

1.3.3 Summary

At the bit level, note the following concerning the energy per operation models:

1. The technology we assumed is old. This implies a certain relatively large minimum feature size (1 µm) and high power supply (V_{dd} = 1 V). High device thresholds drive the power supply higher but give the benefit of negligible leakage power. Advanced processes with multiple threshold voltages may allow greater supply voltage reduction while still keeping leakage to a minimum.
2. The switched capacitance estimates are very conservative, since we assumed that the *entire* capacitance would be switched for each operation. Statistically, that is very unlikely but this is a worst case number.

3. Interconnect was modeled where it could be, based on standard cell layouts in libraries used in a low power DC/DC converter chip [17] and an IDCT processor [18]. These are probably also conservative.

At the architectural level, the right architecture to follow is probably a cached one [21]. Structuring the FFT butterfly in a certain way [22] lets one take advantage of the small cache memory for speed and power (reduced switched capacitance). I ignored this approach and used the simple, flat memory hierarchy that basically maximizes the amount of switched capacitance for memory accesses. Caching will not affect the number of accesses, but will reduce the energy for most of them. Again, this contributes to a worst case estimate.

Other architectural level notes:

1. The multiplier is assumed to be an array. Other architectures (like Wallace trees) might have lower switched capacitance.

2. The timing is computed based on using the same clock as the sample rate (1.8 kHz at low frequency and 18 kHz at fast frequency). The arithmetic operations are assumed to occur serially at the word level and a read-evaluate-write operation is assumed to occur in each clock cycle. The total delay is thus the clock period times the total number of arithmetic operations to compute the FFT (including the averaging at the end) for each FFT to be transmitted. Dividing the total energy by this delay for the FFT computation gives an average power during the time that the chip is working, which is less than the 5 minutes between FFT transmissions specified. This power is sort of the "short-term power" while a computation is proceeding, the rest of the time the chip is off.

3. Pad power (driving off-chip) is not included in these estimates.

To summarize, the total average power for the entire 50 minute computation period is around 170 nW, while the highest short-term power (for the fast sample rate FFT) is about 3.4 µW for about 1.5 minutes. The slow clock rate is what really drives down the power. The caveat is that the peak power (when all those multiplier nodes are switching, for example) might be quite a bit higher. The power supply has to be able to supply these high peak current loads, although this should not be a problem for a big enough capacitor at the output of the DC/DC converter.

A ballpark estimate for the peak power (high sample rate, reasonably fast clock frequency) is

\[ 1.68 \mu W \leq P_{\text{peak}} \leq 5.04 \mu W \]  \hspace{1cm} (1.5)
where $P_{peak}$ is the peak power running for about 81 seconds at 18 kHz. For a comparison, we can compute the power from the chip described in [21] rescaled to the specified FFT parameters. The average power for this chip under our specifications is 3.5 $\mu W$.

For a self-powered system it may be necessary to store energy harvested by the power generator so that it can be used when the load circuitry is powered on. This storage mechanism is a way of conserving the average power scavenged from the environment by the generator so that peak power outputs can be sustained when the load is computing. A convenient mechanism would be to store the accumulated energy as voltage on a large capacitor. The total energy for an entire cycle of FFT computation is 5.015 mJ. The capacitance required to store the total energy consumed by the FFT computer is plotted versus voltage in Figure 1.6. The size of the capacitor influences system design because a large one requires volume that may be impractical in an embedded environment. Size also affects cost, so it is preferable to have as small a volume as possible. This in turn affects the output voltage of the power electronics which are responsible for charging this energy reservoir. High output voltages may be achievable using MEMS energy transducers, but are unlikely to be tolerated by the integrated power FETS of the power electronics. For the FFT processor at 2 volts, the required capacitance is 251 $\mu F$.

1.4 Overview

This dissertation is an exploration of the key concepts in designing systems to be powered from ambient energy sources. Many of the issues discussed hereafter are relevant to a
variety of energy sources, transduction mechanisms, and system applications. The FFT example of Section 1.3 shows that useful computation can be performed with very little power dissipation. The rest of this thesis discusses how to reduce Figure 1.2 to practice.

In Chapter 2, we discuss various opportunities to extract power from the environment of a computing system and how these compare to stored energy alternatives. Chapter 3 reviews the design of transducers for converting mechanical vibration to electrical energy. Chapter 4 describes the design, implementation, and test of a power electronics system for converting mechanical energy to power a simple load DSP. Energy scalable algorithms for typical sensor applications like detection and classification are described in Chapter 5. This is in the context of an acoustic medical sensor for heartbeat detection. This driver application is then implemented in a low power DSP chip discussed in Chapter 6. Chapter 7 reviews the contributions of this work and makes suggestions for continued work in this relatively unexplored area of electronic systems design.

Appendix A is a user guide to the assembly language of the microcontroller implemented as part of the Sensor DSP chip (see Chapter 6). Appendix B is a user guide to the much smaller instruction set of the programmable filter unit of the DSP. Chip pinouts for both implemented chips are shown in Appendix C. Finally, Appendix D presents design notes and suggestions for the practical construction of moving-coil generators.
Chapter 2

Stored vs. Ambient Energy Technologies

Chapter 1 opened the possibility of using ambient energy as a power source for low power VLSI systems, in particular sensor applications with low to medium throughput requirements. Traditionally, power for portable electronics or embedded systems has come from stored energy mechanisms like batteries. Much research has has been focused on improving energy density for chemical batteries, reducing volume and cost or increasing lifetime for the same battery volume [10]. Other research has focused on increasing system integration by finding ways to use micro-electromechanical systems (MEMS) fabrication technology to incorporate batteries on the same semiconductor substrate as the load circuit.

Other recent research has involved developing technologies that use fuels to generate electricity [23]. This research also involves using MEMS to integrate jet turbine technology onto a silicon substrate. The advantage of fuels is the energy density is much higher than for electrochemical batteries.

There are many electronic applications where deriving power from a fixed source, such as a wall outlet, is inconvenient or impossible. Portable electronics are the dominant systems that have this requirement at present. Cellular telephones, personal digital assistants (PDAs) like the PalmPilot, and laptop computers are designed to be small and light enough to be conveniently transported from location to location. This places constraints on the power sources for these devices in terms of lifetime, size, and weight. These systems have the advantage that it is convenient to replace their stored energy systems and it is acceptable to have system downtime for this replacement.

In the future, the dominant systems that require their own power source may be embedded systems. These systems are sensors, actuators, or other appliances that are embedded in everyday objects [24]. For example, actuators may be embedded in the structure of a bridge to increase the load bearing capacity of the members [25]. Temperature sensors may be distributed around a house to more efficiently control a home heating and air conditioning system. Vibration spectra of large machines may need to be monitored by a network of sensors as in Chapter 1. These applications differ from current portable appli-
cations in that they require lower performance and much longer lifetimes. It is also much more difficult or impossible to replace power sources for these embedded systems.

This chapter explores alternative means of powering VLSI systems and the advantages and disadvantages of stored energy power sources and energy harvesting techniques. Both contexts of current portable electronics and future embedded systems and the different demands they place on power sources will be discussed.

2.1 Performance Metrics for Energy Use in Portable Electronics

Low power electronics design has been a major area of recent research. Traditionally there have been two main drivers for low power design: fixed throughput applications and general purpose applications. Fixed throughput applications include video compression and decompression for wireless terminals, low data rate sensor applications, etc. In this domain the performance is constrained to a minimum level set by the desired quality of service to the user. Low power design in this case means reducing the power consumption as much as possible and thus extending battery lifetime as much as possible. General purpose applications, like desktop computers, drive low power design because of the increasing strain of removing the heat generated by these systems. Packaging, airflow limits, etc. can force a maximum power dissipation on these designs. In this domain, it is up to the designer to deliver as much performance as possible while still meeting the maximum power dissipation limit.

Some portable applications, like wireless video terminals, explicitly have fixed throughput requirements because of the user’s desire for real-time video. Fixed throughput is characteristic of signal processing applications. We are primarily concerned with sensor applications like the machine vibration FFT described in Chapter 1; these are also fixed throughput systems although they do not necessarily depend on real-time user interactivity. Their real-time constraints come from system requirements.

Many devices can implicitly have a fixed throughput by imposing a performance requirement on a suite of benchmark applications. For example, a PDA has to be able to store and display addresses and appointments and do some handwriting recognition in real-time. Any extra performance beyond the ability to do this comes at the expense of decreased battery lifetime. For systems with fixed throughput, where by throughput we mean a minimum acceptable level of performance for a suite of typical applications, the design goal should be to maximize battery lifetime.

Other portable systems are more like general purpose computer applications, laptops being the prime example. In this case, power dissipation limits are set not only by the thermal considerations but also by a minimum acceptable battery lifetime. It may be necessary that a laptop battery should be able to last the duration of a transoceanic flight. This duration of service constraint sets an upper bound on the power dissipation of the system. Once this power dissipation maximum is set, then the design should be optimized for performance.
2.2. STORED ENERGY SOURCES

There has been some emphasis in the literature on other, unconstrained low power system optimizations [26] that rely on other metrics like work [27] and energy-delay product [28] [29]. These metrics can result in optimizations that do not accurately reflect user requirements so they will not be considered. The analysis below will show that different sources (stored or ambient) will be more effective in certain application domains. In particular, energy scavenging techniques are very useful for long-life, fixed throughput embedded applications.

2.2 Stored Energy Sources

Since the Nineteenth Century, the dominant mechanism for energy storage for electronic systems has been chemical batteries. The new alternative of combusting fuels is also a chemical energy storage mechanism. Advances in capacitor technology allow the possibility to store energy electrically as well. These alternatives are discussed in this section.

2.2.1 Batteries

All battery technology is based on oxidation-reduction chemical reactions. A typical electrochemical cell consists of an anode, a separator material, and a cathode. The anode is the electrode where the oxidation reaction takes place. Oxidation is the giving up of electrons. Reduction, the absorption of electrons, is the reverse process and occurs at the cathode. For example, a typical lithium battery consists of a solid lithium metal anode. This oxidizes into free electrons and positive lithium ions. The separator divides the anode from the cathode and presents a barrier to the flow of electrons but not to ions. The electrons must flow through the external circuit of the cell to the cathode to be absorbed in the reduction. The lithium ions travel through the separator to a porous electrode cathode. The cathode consists of a solid material that absorbs the lithium ions from solution into the crystal lattice where the lithium combines with the electrons from the external circuit and the lattice material [30]. During recharging, an external power source drives this reaction in reverse.

A wide variety of battery chemistries are currently commercially available and much research is focused on commercializing new materials [10]. In general the movement is toward ever increasing energy density. Both primary (disposable) and secondary (rechargeable) batteries are used in portable electronics. The energy density for primary cells is currently higher, especially for the zinc-air batteries popular in pagers and hearing aids. These offer energy densities on the order of 4.3 MJ/L. A 1 cm³ cell with this energy density can power a 10 µW load for about 14 years! These cells have drawbacks that make them unsuitable for high current draw applications [10], but they are sufficient for low to medium throughput DSPs. Lithium and other chemistry based cells cannot match these energy densities yet, since the cathode material for the zinc-air cell is air. Air does not have to be stored in the cell and so much more volume can be occupied by the anode. Zinc-air cells are still in the early stages of commercialization [31], however, this density can be used as an upper bound for electrochemical energy storage.
An alternative to the construction of traditional cells for energy storage is the use of microfabrication techniques to integrate batteries onto VLSI substrates. This research is in its very early stages and does not offer very high energy densities, 1 J/cm² of area being typically reported [10]. Charge densities of 0.32 Ah/cm³ have been achieved for voltages between 4.5 V and 3.8 V [32], which implies that a 10 μW power load can run for 16 years from a 1 cm³ source. However, it is not likely that devices of this size can be fabricated with this thin-film technology. However, the devices have the advantage of being able to sustain a high charging current and numerous charge-discharge cycles without a reduction in capacity.

The explosive growth of portable devices that use batteries has led to concern regarding their disposal. Large lead-acid batteries such as those used in automobiles are recycled at a very high rate. However, small primary and secondary cells often end up in landfills where their often hazardous materials are leached into the environment. As the use of batteries continues to grow into the next century, their environmental impact will likely eventually require recycling of all battery types [10, 31].

2.2.2 Combustion of Fuels

Fuels are probably the oldest energy source used by mankind, starting with the burning of wood. Fossil fuels have been used since the last century in steam engines, internal combustion engines, and jet or rocket engines. Combustion is the reaction of a fuel with oxygen to produce oxides of the fuel elements. These reactions are exothermic and the energy they release can be used to heat water to drive an electrical generator or can be converted directly to rotational movement of a turbine or linear movement of a piston. This mechanical movement can then be converted into electricity. All of these devices have been built on the macroscopic scale and generate kilowatts to megawatts of power. The advent of MEMS technology now allows the possibility to create microscopic versions of these engines. One application is to convert the fuel to electricity to power electronics [23].

The advantage of this approach is the energy density is higher for fuels than for electrochemical cells. For the same volume, a microscopic turbine generator can produce much more energy. The downside is that these devices produce power on the order of 10s of Watts, which is much higher than is required for sensor applications although it is sensible for other applications like laptops. There is no way to reduce this power output currently, but it may be possible to store the electrical energy (see Section 2.2.3). It is also difficult to switch this energy source on and off so it is not optimal for low duty cycle applications like the vibration FFT of Chapter 1.

2.2.3 Electrical Storage Mechanisms

Large capacitors are currently being used as battery backup devices in computer memories. Several devices are commercially available that have a 1 F capacity in volumes slightly more than 1 cm³ [33]. Although these capacitors offer large charge storage in a small volume, they do have a large parasitic series resistance which can cause significant voltage
drop for high current loads. This is not an issue for sensor applications as long as the peak current draws are low, in the 10s of microwatts. The addition of a DC/DC converter can reduce the peak current requirements to an acceptable level. A 1 F capacitor charged to 5 V can provide 10 μA at 1 V for about 140 hours. For the FFT example of Chapter 1, the capacitor can provide 8200 hours of operation.

2.3 Ambient Energy Sources

Various schemes have been proposed to eliminate the need for batteries in a portable digital system [11]. Basically, energy that exists in the environment of the device is transduced into electrical form which can be used by a circuit to perform useful work. The sources of ambient energy available to the system depend on the application. Examples include light or electromagnetic fields, thermal gradients, fluid flow, and mechanical vibration. Other proposals include powering electronic devices through harnessing energy produced by the human body [15] or the action of gravitational fields [16].

A generator based on transducing mechanical vibrations has some distinct advantages: it can be enclosed and protected from the outside environment, it functions in a constant temperature field, and it can be activated by a person (by shaking it, for example). However, it has moving parts and hence less long-term reliability, as well as a more complex mechanical design.

2.3.1 Solar Power

Solar power has been the most extensively used ambient energy source. Large scale solar power arrays have been used for home electricity needs and in satellites. Smaller arrays are used in handheld calculators and more recently in wristwatches. The upper bound for efficiency was calculated to be about 30 % [34]. Current research has increased efficiency to about 20 % which yields about 20 mW for a 1 cm² solar cell [35]. This power level is certainly enough to support sensor applications, but it is not enough for higher power loads like laptop computers. An additional drawback is the requirement of a light source. This may not be available for sensors embedded in a structure or attached to an item in a shipping container. In addition, solar power scales with the surface area of the system instead of the volume, so the size requirements of the application may also preclude the use of this energy source.

2.3.2 Thermal Power

Electrical current can also be generated by a difference in temperature. A thermocouple is a device that creates a current proportional to the temperature difference across it. This current can then be used to charge a capacitor. The total power available from this source is fairly high: about 30 mW for a 1 cm² device with a temperature difference of 25 C [36]. Like the solar cell, this technique requires certain environmental conditions to operate.
The output power decreases quadratically with the temperature difference. Sensors that are embedded in constant temperature environments would not be able to generate their own power.

2.3.3 Electromagnetic Fields

Many recent efforts have focused on using electromagnetic radiation as a power source. Some examples include using RF. Identification tags have been created that receive and store energy from an incident "interrogating" RF field [12]. This energy is then used to transmit a serial number back to the interrogator. Power levels for this technique can be quite high, but it requires the existence of a high energy incident field. Ambient RF energy from solar radiation and man-made sources is too small to be effectively used for electronics.

An alternative to using RF energy is to use coupling of magnetic fields, much like the coils of a transformer. This technique has been used to recharge pacemaker batteries without performing invasive procedures. It has also been used in telephone smart cards, where data from the cards is read using magnetic fields created in the slot where the card is inserted [13]. This field is used to power the readout circuitry on the cards in addition to forming the communication channel between card and telephone. Again, this requires an incident field since the levels of ambient field are too low.

2.3.4 Fluid Flow

Windmills are another ancient energy source. A miniature windmill (or propeller for a submerged application) can spin a turbine generator to create power. High wind speeds can result in large amounts of power, on the order of 100 W for a generator with even a few square centimeters of flow area. Using Bernoulli's equation, it is possible to write the power of a flowing liquid as the product of the specific energy and the volume flow [37]:

\[
P_{\text{liq}} = e_v q = p + \rho gz + \rho \frac{v^2}{2}
\]  

(2.1)

where \( P_{\text{liq}} \) is the power of the flow, \( e_v \) is the specific energy per unit volume, \( q \) is the volume flow, \( p \) is the pressure, \( \rho \) the liquid density, \( g \) the gravitational constant, \( z \) the difference in inlet and outlet height, and \( v \) the fluid velocity. Bernoulli's equation is a statement of conservation of energy for a fluid under all circumstances so not all of the terms in Equation 2.1 are nonzero in some applications. For the example of pure water flowing at a rate of 1 cm/s at a volume flow of 10 cm³/s, the power is 50 μW. The wide range of circumstances where this source could be used means that almost any level of power could be harvested if the right situation is found. However, this source requires fluid movement to be effective and so it is unsuitable for many applications. Sensors installed in exhaust
2.3. AMBIENT ENERGY SOURCES

<table>
<thead>
<tr>
<th>Energy Source (Direct Conversion)</th>
<th>Transducer</th>
<th>Output Power</th>
</tr>
</thead>
<tbody>
<tr>
<td>Walking</td>
<td>Piezoelectric</td>
<td>5 W [15]</td>
</tr>
<tr>
<td>Thermal</td>
<td>Thermoelectric</td>
<td>30 mW [36]</td>
</tr>
<tr>
<td>Solar</td>
<td>Photovoltaic Cell</td>
<td>20 mW [35]</td>
</tr>
<tr>
<td>Magnetic Field</td>
<td>Coil</td>
<td>1.5 mW [13]</td>
</tr>
<tr>
<td>Walking (Vibration)</td>
<td>Discrete Moving Coil</td>
<td>400 μW</td>
</tr>
<tr>
<td>High Frequency Vibration</td>
<td>MEMS Moving Coil</td>
<td>100 μW [11]</td>
</tr>
<tr>
<td>Small Fluid Flow</td>
<td>Turbine</td>
<td>50 μW</td>
</tr>
<tr>
<td>RF Field</td>
<td>Antenna</td>
<td>5 μW [12]</td>
</tr>
</tbody>
</table>

Table 2.1
Summary of ambient energy sources.

pipes, air conditioning and heating systems, or other plumbing could all take advantage of this available energy.

2.3.5 Mechanical Vibration

The last ambient energy source we will consider is mechanical vibration, which will be explored in detail throughout this dissertation. There are many sources of vibration throughout the environment which will be discussed in the next section. The advantages of mechanical vibration, in addition to wide variety of sources, is the wide variety of transduction mechanisms available. These will also be discussed in the next section. In general, mechanical vibration offers the most flexible source of ambient energy since it is available to embedded sensors placed in constant temperature fields, away from light or shielded from other electromagnetic radiation, and isolated from fluid motion. As will be shown below, significant quantities of power can be generated from a number of vibration sources.

Table 2.1 summarizes the available power harvested from a subset of the sources described above, in descending order of their typical output power. The data is taken from various sources in the literature. It is clear that there is a substantial amount of useful energy in the environment available in a variety of forms. As the power requirements of VLSI systems continues to decrease, more opportunities arise to use these sources.

2.3.6 Other Mechanical Energy Sources

Sound is a form of mechanical vibration and solids and liquids conduct sound waves very well. For our purposes, we will consider only sound waves that travel through the air and are within the hearing range of humans. As will be shown in Section 4.8.2, the sensitivity of the human ear is so high that significant levels of acoustic power are only available beyond the threshold of pain. However, there are other regions where this requirement
can be met, notably airports with their attendant noise pollution. The possibility of using acoustic energy needs to be explored in future work.

Some commercial wristwatches use gravity to generate electricity [16], which is stored on a capacitor. The transducer consists of a gear whose center of gravity is offset radially from the center of rotation. As the watch changes orientation with respect to gravity, the gear turns through a fraction of a rotation. This movement is then geared up through a geartrain into multiple rotations of a permanent magnet. The AC voltage is then rectified onto a storage capacitor and is then used to power a quartz crystal and other timekeeping circuitry as well as the motor that moves the watch hands. This commercial application is a proof of concept that ambient energy techniques have practical uses.

2.4 Power from Mechanical Vibration

Mechanical vibration is a promising source of ambient energy, especially for embedded sensor applications. In this section, we describe various sources of vibration, discuss the implications of using discrete versus MEMS generators, and show a quantitative analysis of the amount of energy available from different vibration sources.

We describe our particular approach to transduction of mechanical vibration to electrical energy and evaluate the potential of this approach for batteryless operation of portable electronics. First, we present a simple model for the power transfer from mechanical vibration to the transducer and then follow that with a stochastic evaluation of the power generated by a typical application.

2.4.1 Power Transfer

A reasonable model of the power transfer from ambient vibration to the power generated is necessary to study the feasibility of using a generator based on mechanical vibration. Figure 2.1 shows a schematic representation of the mechanical power transfer mechanism of the generator. The position of the generator housing relative to an absolute coordinate system is $y(t)$ and the position of the mass relative to its rest position fixed with respect to the generator housing is $z(t)$. The forces produced by the spring and the dashpot are related to the displacement relative to the housing since the entire system is translated by $y(t)$. The absolute position of the mass is therefore $y(t) + z(t)$. For the preliminary analysis, we ignore the higher order modes of the system and concentrate on the vertical dynamics only.

Assuming a linear spring and dashpot and applying Newton's second law by taking second derivatives of the absolute mass position, we can derive the equation of motion for the mass [11]:

$$m\ddot{z} + B_m\dot{z} + kz = -m\dot{y}$$ (2.2)
Figure 2.1: Schematic of power transfer mechanism. As the housing surrounding the mass-spring-dashpot system is vibrated, kinetic energy is imparted to the mass. The electromechanical transducer and the mechanical losses in the system are modeled by the dashpot.

where \( m \) is the generator mass, \( B_m \) is the dashpot damping coefficient, and \( k \) is the Hooke's Law spring constant. The applied mechanical excitation force, \( f_m \), is

\[
f_m = -m\ddot{y}
\]

which, combined with Equation 2.2, conveniently relates displacements of the housing to displacements of the proof mass. In the frequency domain, the transfer function is

\[
H(s) = \frac{Z(s)}{Y(s)} = \frac{-s^2}{s^2 + \frac{B_m}{m}s + \frac{k}{m}} \quad \frac{-s^2}{s^2 + 2\zeta\omega_0 s + \omega_0^2} \tag{2.3}
\]

where Equation 2.3 has been rewritten in terms of the standard notation for natural frequency \( \omega_0 \) and dimensionless damping factor \( \zeta \). However, the quantities of most interest are the relative velocity \( \dot{z}(t) \) and absolute velocity \( v(t) = \dot{y}(t) + \dot{z}(t) \) of the proof mass. The output voltage of the coil turns out to be roughly proportional to the relative velocity (see Section 3.1) and the transfer function from the housing displacement to this velocity is merely \( sH(s) \), where \( H(s) \) is defined in Equation 2.3. The power transferred from the mechanical excitation force \( f_m \) to the proof mass is the product of the force and the absolute velocity of the mass:
\[ p(t) = f_m(t)v(t) = -m\ddot{y}(t)[\ddot{y}(t) + \dot{z}(t)] \] (2.4)

An analytic formula for the power transferred by a sinusoidal excitation, \( y(t) = Y_0 \cos(\omega t) \), of the generator housing is presented in [11]. However, to estimate the amount of power that can be obtained and to gain insight into generator design it is useful to look at the frequency domain representation. The power \( p(t) \) is the product of the applied force and the velocity of the mass. From this we can derive the Fourier transform of the power:

\[ P(j\omega) = -m\omega^2Y(j\omega) \ast \left( \frac{-j\omega^3}{\omega_0^2 - \omega^2 + 2j\omega_0\zeta\omega} \right) Y(j\omega) \] (2.5)

where \( \zeta \) and \( \omega_0 \) are the damping factor and natural frequency (radians/s) for a second order system respectively.

### 2.4.2 Vibration Sources

There are numerous sources of mechanical vibration. Low frequency vibrations are transmitted through the earth at all times. These movements are due to microtremors caused by geologic activity or the movement of vehicles or people. A device worn on the body can be powered by movements of the user. These sources are not necessarily steady state, but the occasional impulse can generate electricity that can be stored, like the self-winding watches of the past. As long as the impulses occur frequently enough to support the average power requirements of the device, these intermittent sources are viable sources of energy.

Many steady state sources of vibration also exist and these are the most promising for powering electronics. Like the gas turbine generator example of Chapter 1, all large machines support constant vibrations created by their moving parts. These machines present the most natural application of energy harvesting from mechanical vibration. It is desirable to instrument these machines with sensors to monitor their performance and predict failures so that preventive maintenance can be performed. These applications also have a long lifetime requirement that may be met more easily by harvesting energy than by using batteries. In Section 2.4.6, we discuss estimates of the maximum extractable power from machine vibration.

### 2.4.3 Macroscopic vs. MEMS Generator Implementations

Many alternative transducers exist for converting vibrations to electrical signals. Accelerometers and microphones are common examples of sensors that perform this function. For power conversion, the requirements for linearity can be relaxed in exchange for maximum power output. In Chapter 3, we will describe in detail two different transducer designs:
2.4. POWER FROM MECHANICAL VIBRATION

![Stochastic Vibration Bandpass Model Displacement](image1)

![Stochastic Simulation: Displacement](image2)

Figure 2.2: Displacement shaping filter and representative time domain displacement simulation of vibration of the generator housing due to human walking.

one macroscopic implementation based on a moving-coil electromagnetic transducer and another MEMS implementation based on a variable capacitor. Both of these mechanisms can produce enough current to supply sensor DSP loads. The next section describes how the amount of energy harvested depends on the scale of the device and the frequency of the vibration. In general, a macroscopic device is able to produce much greater amounts of energy for the same frequency of vibration. The chief drawback of a macroscopic implementation is cost. MEMS devices have the advantage of massively parallel microfabrication and so the cost of each device is potentially much lower. Small volume requirements may even preclude the use of the discrete devices.

2.4.4 Models of Vibration

In order to assess the amount of power available from ambient vibration, it is necessary to develop reasonable models of vibration sources. The following sections describe models for vibrations due to human walking and machine vibration.

Walking

A typical person walking at about 3.5 mph takes 2 steps per second [15]. The typical person's heel travels through approximately 5 cm in the vertical plane. The model for power transfer derived in Section 2.4.1 relates the power into the proof mass to the vertical displacement of the generator housing.
Figure 2.3: Velocity shaping filter and representative time domain velocity simulation of vibration of the generator housing due to human walking.

Figure 2.4: Acceleration shaping filter and representative time domain acceleration simulation of vibration of the generator housing due to human walking.
We can construct a stochastic model of the vibrational displacement due to human walking by passing white noise through a narrow band filter [38] centered at about 2 Hz [15, 39]. Assuming that an object carried in a pocket or worn will have less vertical displacement than the heel, we limit the peak displacements to ±2 cm. Figure 2.2(a) shows the shaping filter for the stochastic vibration displacement. The center frequency of the passband is at 2 Hz with a bandwidth of 2 Hz. The first transition band has a 20 dB/decade rollup while the second transition band rolls off at -40 dB/decade, so we expect more low frequency content in the displacement signal.

For completeness, Figure 2.3(a) and Figure 2.4(a) shows the spectra for the housing velocity and acceleration due to walking. As the signal is differentiated, the velocity and acceleration should look increasingly like white noise, especially acceleration which is basically a simple highpass filter.

The stochastic model system for walking was simulated using MATLAB. Figure 2.2(b) shows typical steady-state outputs for a 2 second interval. We see that the displacement has mostly low frequency content while the velocity and acceleration look increasingly white. The peak values of displacement are bounded by 2 cm. The RMS values for the quantities are 3.97 mm, 5.63 cm/s, and 1.44 m/s² for the displacement, velocity, and acceleration respectively. The RMS value for displacement seems quite reasonable for a device that is worn or carried in a pocket.

The stochastic model described above for y(t) was simulated to generate an output power time series using Equation 2.6. This is plotted in Figure 2.5 for about 2 seconds.
The average output power dissipated in the dashpot was 400 $\mu W$, but this is really a best case estimate of output power. The mechanical damping was ignored in this case and the generator was optimized by tuning the mechanical resonance of the spring-mass system to the 2 Hz vibration center frequency and minimizing the damping within the constraints of the relative mass displacement. A self-powered system using human walking as a vibrational power source must therefore consume less than 400 $\mu W$, a very reasonable goal considering the current state of the art in low power VLSI design.

**Machine Vibration**

As discussed in Chapter 2, machine vibration is the most reasonable steady-state source of mechanical power. Both reciprocating engines, like multicylinder automobile engines, and rotating machinery produce oscillations throughout their structure. Cylinder engines generate vibration in their engine blocks as the cylinders fire. Torsional vibrations occur in the crankshafts as well. Rotating machines also have torsional vibrations and unbalanced ones may vibrate in the plane normal to the axis of rotation. In general, mechanical design of these devices involves a careful analysis of vibrational modes and the possible addition of damping to eliminate dangerous movement that may cause failure [40]. However, as in the case of the gas turbine generator mentioned in Chapter 1, there is often sufficient vibration for energy harvesting after the dangerous oscillations have been minimized.

Machine vibration is a much simpler to model than human movement. Machines move at a fixed frequency or set of frequencies, determined by the cylinder firing timing of an internal combustion engine for example. These fundamental frequencies then couple into other modes of the system whose frequencies are set by geometry and material factors. In general, the vibration spectrum seen by a sensor mounted on a machine will have several narrowband peaks at harmonics of the fundamental vibration mode. Thus, for optimal power transfer, the generator resonant frequency should be tuned to one of these peaks.

### 2.4.5 Power from Ambient Vibration

Figure 2.1 shows a simplified model of the generator. The housing vibration displacement $y(t)$ couples into a relative mass displacement $z(t)$. The electrical and mechanical damping are lumped into one dashpot element. The output power is therefore

$$P_o(t) = B_m \dot{z}^2$$  \hspace{1cm} (2.6)

where $B_m$ is the lumped dashpot coefficient and $\dot{z}$ is the mass velocity relative to the housing. Suppose a sinusoidal excitation $y(t) = Y_0 \cos(\omega t)$ drives the system. From the well known theory of linear resonant systems, the output DC power in sinusoidal steady state is:

$$P_{out} = \frac{F_0^2 \zeta \omega^2}{m \omega_0^3 \left[ \left(1 - \frac{\omega}{\omega_0} \right)^2 + \left(2 \zeta \frac{\omega}{\omega_0} \right)^2 \right]^3}$$  \hspace{1cm} (2.7)
where $\zeta$ is the normalized damping coefficient, $m$ is the mass, and $\omega_0$ is the natural frequency in rad/s of the resonator.

From Equation 2.7, we see that the output power is maximized when the system is driven at the resonant frequency, i.e. $\omega = \omega_0$. Equation 2.7 simplifies to:

$$P_{out} = \frac{F_0^2}{4m\omega_0\zeta}$$

(2.8)

To determine the maximum extractable power, we must determine the input force $F_0$ and the damping coefficient $\zeta$. We can write the force in two ways,

$$F_0 = m\omega_0^2 Y_0$$

(2.9)

$$= m\omega_0 V_y$$

(2.10)

where $Y_0$ is the amplitude of the excitation vibration and $V_y$ is its velocity.

Maximizing the output power would require minimizing the damping coefficient $\zeta$, which results in an unacceptably large displacement of the proof mass. A more realistic formulation would be to set a limit on proof mass movement and choose the damping coefficient that results in that displacement. For a resonant system,

$$Z = \frac{F_0}{m \left[ (\omega_0^2 - \omega^2)^2 + (2\zeta\omega_0\omega) \right]^{\frac{1}{2}}}$$

(2.11)

where $Z$ is the mass displacement. At resonance and setting $Z = Z_{\text{max}}$, where $Z_{\text{max}}$ is the maximum allowable mass displacement yields

$$Z_{\text{max}} = \frac{F_0}{2m\zeta\omega_0^2}$$

Solving for $\zeta$,

$$\zeta = \frac{F_0}{2m\omega_0^2Z_{\text{max}}}$$

(2.12)

Finally, substituting Equations 2.9, 2.10, and 2.12 into Equation 2.8 yields

$$P_{out} = \frac{m\omega_0^3Y_0Z_{\text{max}}}{2}$$

(2.13)

$$= \frac{m\omega_0^3V_yZ_{\text{max}}}{2}$$

(2.14)
Figure 2.6: Transfer functions from housing position \( y(t) \) to mass velocity \( \dot{z}(t) \) and position \( z(t) \). The damping is kept constant while the first corner frequency of the highpass characteristic varies. The high order single pole rolloff at 1000 rad/s represents unmodeled high frequency dynamics.

Figure 2.7: Transfer functions from housing position \( y(t) \) to mass velocity \( \dot{z}(t) \) and position \( z(t) \). The corner frequency is kept constant while the damping factor varies. The high order single pole rolloff at 1000 rad/s represents unmodeled high frequency dynamics.
Equations 2.13 and 2.14 are used given the appropriate data to estimate the possible extractable power.

Figures 2.6 and 2.7 represent different transfer functions from the housing position $y(t)$ to the proof mass velocity $\dot{z}(t)$ and position $z(t)$. These plots are based on the transfer function described in Equation 2.3. In Figure 2.6, the damping coefficient is kept constant while the corner frequency of the highpass filter is varied. Figure 2.6(a) shows that as the velocity of the housing displacement increases, it couples more strongly to the mass velocity (increasing the kinetic energy stored in the mass) until the coupling is maximized above the corner. Figure 2.6(b) shows the mass displacement $z(t)$ as the housing displacement frequency varies. The displacement peaks at the corner frequency, or resonant frequency, of the system, which also conforms to the maximum power transfer as will be shown in Section 2.4.5.

Figure 2.7 shows what happens as the damping coefficient of the system varies. Decreasing the damping creates a more sharply defined and higher resonant peak, which again results in maximum power transfer. The highpass nature of both transfer functions show that it is better to be at or higher than the resonant frequency of the system to maximize the coupling from the housing vibration to the internal mass vibration. Although maximum power transfer occurs when the system’s resonant frequency is tuned to the input vibration, this is not always possible if the vibration source is stochastic, time-varying, or broadband in nature. In that case it is better to set the corner frequency lower than the lower end of the broadband vibration spectrum.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Assumed Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proof Mass</td>
<td>1 g</td>
</tr>
<tr>
<td>Resonant Frequency (Vibration Frequency) (f_0)</td>
<td>1 kHz</td>
</tr>
<tr>
<td>Maximum Mass Displacement</td>
<td>(\pm 1) cm</td>
</tr>
</tbody>
</table>

Table 2.2
Assumed parameters for a discrete transducer implementation.

Equation 2.5 gives the frequency domain representation of the power transfer function. Figure 2.8 shows an example of two output power spectra for a bandpass input vibration centered on 5 Hz. The top plot shows the power for a natural frequency \(f_0\) of 1 kHz, much greater than the excitation frequency. The DC component of the power is almost zero in this case. The bottom plot shows the power for a natural frequency of 5 Hz and the DC power is much greater. For maximum power, one would like \(\omega_0\) to be close to the expected input frequencies. This is often not practical, especially if the generator must have a small mass and displacement and the vibration frequencies are very low (corresponding to human movement). Even in the first case in the figure, the DC power is approximately 400 \(\mu\)W - enough to power ultra low power DSP systems.

2.4.6 Estimates of Maximum Extractable Power

These back of the envelope estimates were made using tables of maximum allowable machine vibration from different vibration standards [41]. The standards are specified as maximum peak-to-peak displacements or maximum RMS velocity of different measurement points. To compute the maximum output power, we assume a transducer of the proof mass-spring type fixed to a rigid housing. We further assume that the frequency of vibration is 1 kHz, about the midrange for vibration frequencies for most machinery, and that the transducer resonant frequency is tuned to this excitation frequency. We also assume that the damping coefficient \(\zeta\) is chosen to limit the proof mass vibration amplitude to \(\pm 1\) cm. The mass of the moving part of the transducer was chosen to be 1 g, potentially much larger than any micromachined proof mass. Table 2.2 lists the mechanical assumptions for maximizing output power for a discrete implementation.

Table 2.3 shows the maximum output power calculated using Equation 2.7 for different types of machinery. Tables 2.4 and 2.5 repeat the calculations for assumptions based on a microfabricated transducer. In general, the output powers listed above are scaled down by several orders of magnitude thanks to the greatly decreased maximum deflections and sizes of the proof mass. The frequency remains the same for the calculation, however. Equation 2.13 shows that the output power scales as the cube of the vibration frequency. Microfabricated generators can be tuned more easily to high frequencies because of their small mass. The capacitive transducer described in Chapter 3 takes advantage of this frequency dependence to produce reasonable output power for the small size.

These estimates rely on a perfect energy conversion from vibration to electricity. In reality, numerous loss mechanisms drive the efficiency down. First, it is difficult to tune
### Table 2.3
Maximum output power for different machinery classes for a discrete transducer.

<table>
<thead>
<tr>
<th>Machine</th>
<th>Max. Output Power</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class I Rotating Machinery</td>
<td>0.5527 W</td>
</tr>
<tr>
<td>(e.g. electric motors up to 15 kW)</td>
<td></td>
</tr>
<tr>
<td>Class II Rotating Machinery</td>
<td>0.8883 W</td>
</tr>
<tr>
<td>(e.g. 15-75 kW electric motors, 300 kW engines)</td>
<td></td>
</tr>
<tr>
<td>Class III Rotating Machinery</td>
<td>1.4015 W</td>
</tr>
<tr>
<td>(Large prime movers on rigid foundations.)</td>
<td></td>
</tr>
<tr>
<td>Class IV Rotating Machinery</td>
<td>2.2108 W</td>
</tr>
<tr>
<td>(Large prime movers on soft foundations.)</td>
<td></td>
</tr>
<tr>
<td>Electric Motor, 1 m Shaft Height</td>
<td>0.3553 W</td>
</tr>
<tr>
<td>Horizontal Clear-Liquid Pumps</td>
<td>19.5315 W</td>
</tr>
<tr>
<td>Horizontal Nonclog Pumps</td>
<td>34.1801 W</td>
</tr>
<tr>
<td>Reciprocating Piston Compressors</td>
<td>13.8174 W</td>
</tr>
<tr>
<td>Gear Units</td>
<td>0.6218 W</td>
</tr>
<tr>
<td>Shipboard Equipment</td>
<td>4.8829 W</td>
</tr>
</tbody>
</table>

### Table 2.4
Assumed parameters for a micromachined transducer implementation.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Assumed Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proof Mass</td>
<td>60 mg</td>
</tr>
<tr>
<td>Resonant Frequency (Vibration Frequency) $f_0$</td>
<td>1 kHz</td>
</tr>
<tr>
<td>Maximum Mass Displacement</td>
<td>±400 μm</td>
</tr>
</tbody>
</table>

### Table 2.5
Maximum output power for different machinery classes for a micromachined transducer.

<table>
<thead>
<tr>
<th>Machine</th>
<th>Max. Output Power</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class I Rotating Machinery</td>
<td>1.3265 mW</td>
</tr>
<tr>
<td>(e.g. electric motors up to 15 kW)</td>
<td></td>
</tr>
<tr>
<td>Class II Rotating Machinery</td>
<td>2.1318 mW</td>
</tr>
<tr>
<td>(e.g. 15-75 kW electric motors, 300 kW engines)</td>
<td></td>
</tr>
<tr>
<td>Class III Rotating Machinery</td>
<td>3.3636 mW</td>
</tr>
<tr>
<td>(Large prime movers on rigid foundations.)</td>
<td></td>
</tr>
<tr>
<td>Class IV Rotating Machinery</td>
<td>5.3059 mW</td>
</tr>
<tr>
<td>(Large prime movers on soft foundations.)</td>
<td></td>
</tr>
<tr>
<td>Electric Motor, 1 m Shaft Height</td>
<td>852.73 μW</td>
</tr>
<tr>
<td>Horizontal Clear-Liquid Pumps</td>
<td>46.876 mW</td>
</tr>
<tr>
<td>Horizontal Nonclog Pumps</td>
<td>82.032 mW</td>
</tr>
<tr>
<td>Reciprocating Piston Compressors</td>
<td>33.162 mW</td>
</tr>
<tr>
<td>Gear Units</td>
<td>1.4923 mW</td>
</tr>
<tr>
<td>Shipboard Equipment</td>
<td>11.719 mW</td>
</tr>
</tbody>
</table>
the mechanical filter resonance exactly to the input frequency and may not even be theoretically possible if the vibration source is broad and. Therefore not as much of the energy couples into the generator's moving mass. The transduction to electrical energy is not perfect either as energy is dissipated in heat through electrical resistance as well as mechanical damping. However, for sensor applications the load power requirements are substantially lower than the output power available even in the microfabricated generator case. Efficiency is not much of an issue since even harvesting 1% of the available energy is often sufficient for the applications of interest.

2.5 The Future of Portable Electronic Systems

As mentioned previously in this chapter, the dominant use of portable electronics has been in explicit information appliances and communications devices. These include pagers, cellular phones, PDAs, laptops, handheld calculators, portable radios and CD players, etc. This market continues to grow as prices fall and consumers are attracted to the convenience of wireless connectivity and portability. In addition to this existing application domain, a new domain is currently being explored through a number of research efforts. This domain involves moving information processing technology to other, everyday objects and not just information devices.

The applications in this new domain are tremendous. Networking and programming standards like Bluetooth and Jini are making wireless connectivity possible for a wide range of devices and making this network transparent to the end user. This network addresses the issue of how to communicate with sensors. Adding sensors to everyday devices adds a new dimension to the standard networked computing model because it allows more flexibility for the user to interact with the environment. For disposable items or items with short user life, it is not necessary to provide long term energy. For other things, like capital investments in machinery, buildings, or transportation, long life is a very desirable feature of the embedded sensors. Section 2.4.6 has shown that useful levels of power are available from mechanical vibration of large equipment. Future technologies will present even more opportunities for energy harvesting.

2.6 Tradeoffs Between Stored and Ambient Energy Technology

Table 2.6 summarizes the performance of various stored and ambient energy sources. Energy density is not a particularly useful metric for ambient sources since it is infinite. Lifetime is also potentially infinite for energy harvesting systems. However, some stored energy techniques can provide enough energy to support an extended life application. The table also shows that some sources, fuels in particular, cannot support the low power requirements of sensor applications. The lifetime column in the table was computed for a 1 cm³ volume of stored energy supplying a 10 µW load, if possible. Otherwise the lowest output power available from the technology is used.
2.6. TRADEOFFS BETWEEN STORED AND AMBIENT ENERGY TECHNOLOGY

<table>
<thead>
<tr>
<th>Energy Source</th>
<th>Energy Density</th>
<th>Output Power Range</th>
<th>Lifetime</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Stored Energy Sources</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Battery</td>
<td>4.3 kJ/cm³</td>
<td>10 W - 1 μW</td>
<td>14 years</td>
</tr>
<tr>
<td>Fuel Combustion</td>
<td>125 kJ/cm³</td>
<td>100 - 10 W</td>
<td>3.5 hours</td>
</tr>
<tr>
<td>Capacitor</td>
<td>5 J/cm³</td>
<td>100 - 10 μW</td>
<td>8200 hours</td>
</tr>
<tr>
<td><strong>Ambient Energy Sources</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Thermal</td>
<td>-</td>
<td>30 mW</td>
<td>-</td>
</tr>
<tr>
<td>Solar</td>
<td>-</td>
<td>20 mW</td>
<td>-</td>
</tr>
<tr>
<td>Electromagnetic Fields</td>
<td>-</td>
<td>1.5 mW - 5 μW</td>
<td>-</td>
</tr>
<tr>
<td>Fluid Flow</td>
<td>-</td>
<td>10 W - 50 μW</td>
<td>-</td>
</tr>
<tr>
<td>Mechanical Vibration</td>
<td>-</td>
<td>100 μW</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 2.6
Summary of stored and ambient energy sources.

The table supports the notion that ambient energy technology is most useful for long life embedded applications. Advances in low power electronics indicate that traditional stored energy sources or novel sources like fuels are good for short lifetime, higher power applications. However, ambient and stored energy technologies are really complementary and not mutually exclusive. For example, a cellular phone may have a battery that can be recharged through an electrical socket, but it may also have a generator that can charge the battery as a person shakes the phone. This redundancy is very useful for emergencies. Some commercial applications have used wind-up mechanisms to provide power for flashlights and radios [42]. Energy harvesting can be used to increase the lifetime of battery powered systems or to reduce battery weight for portable systems. Reducing the use of batteries may also be beneficial for the environment. This chapter has merely outlined some of the potential benefits of energy harvesting; much work remains to be done in exploiting ambient sources of energy for portable electronics.
Chapter 3

Vibration to Electrical Energy Transducer Design

In Chapter 2, several different ambient energy sources were discussed and compared to options for using stored energy. For long lifetime, low power embedded sensor applications, ambient energy sources have many advantages. It is in this context that we choose to explore mechanical vibration as a power source for VLSI systems. This chapter discusses several transducer options and design optimizations for converting mechanical energy to electrical energy.

3.1 Moving Coil Generator

Williams and Yates propose an inertial electromechanical generator [11]. This device consists of a mass $m$ connected to a spring $k$, the whole mounted within a rigid housing. As the housing is vibrated, the mass moves relative to the housing and energy is stored in the mass-spring system (see Figure 3.1). A coil of wire $l$ is attached to the mass and moves up and down into a field generated by the permanent $B$. The changing position of the coil varies the amount of flux linked by the coil wires and thus induces a voltage on the coil. The basic principle of operation of this transducer is simple, however the moving parts required result in decreased reliability of the system. Detailed models will be presented in the following sections.

3.1.1 Linear Model

Making the simplified field assumptions corresponding to the geometry above, it is relatively simple to write the equations of motion for the system using magnetostatics [43] [44]. There are two components to the magnetic flux $\lambda$ linked by the coil. First, the space between the coils links the radially directed field produced by the permanent magnet. Applying Gauss' Law for this surface shows that this term is proportional to the
product of the permanent magnet field, \( B \), the length of the coil, \( l \), and the time-varying position of the coil, \( z(t) \). Second, assume a current \( i(t) \) runs through the coil. This current also sets up a radially directed field whose flux is linked by the inductance of the coil, \( L(z) \), where we have made explicit the position dependence of the inductance. The magnetic flux is therefore

\[
\lambda = L(z)i(t) + Blz(t).
\] (3.1)

Since we are concerned with the electrical power delivered by the coil, assume that a resistor \( R \) is placed across the coil terminals. Then the voltage \( v(t) \) on the coil is

\[
v(t) = -i(t)R
\] (3.2)

where the current direction has been chosen to reflect the negative power flow from the generator coil. Suppose the coil has a parasitic series resistance \( R_c \). Then writing KVL around the loop shows

\[
v(t) = R_c i + \frac{d\lambda}{dt} = \frac{-R_c v}{R} + \frac{-L(z) d\lambda}{dt} \left( -\frac{v}{R} \frac{dL}{dz} \frac{dz}{dt} + B \frac{dz}{dt} \right)
\]

\[
= \frac{-R_c v}{R} + \frac{-L d\lambda}{R dt} + B l \frac{dz}{dt}
\] (3.3)

where Equations 3.1 and 3.2 have been used to solve for the voltage. The deflections of the coil are assumed to be small enough that the inductance is approximately constant.
Now we must relate the voltage to the dynamics of the mass motion. There are two forces acting on the mass, the mechanical forcing function and the electromagnetic force caused by the induced currents in coil. Writing this second force using the Lorentz force law, we have and Newton's Second Law of Motion [45]

\[ f_m + f_e = f_m + Bli \]
\[ = m \frac{d^2 z}{dt^2} + B_m \frac{dz}{dt} + kz \]

where the mechanical parameters are as defined in Section 2.4.5. Rearranging the above equation and combining with Equation 3.3 yields the coupled equations of motion for the system:

\[ f_m = m \frac{d^2 z}{dt^2} + B_m \frac{dz}{dt} + kz + Bl \frac{v}{R} \tag{3.4} \]
\[ Bl \frac{dz}{dt} = L \frac{dv}{dt} + \left( \frac{R_c}{R} + 1 \right) v. \tag{3.5} \]

Using Equations 3.4 and 3.5 we can construct a block diagram for the transfer function from the applied force to the output voltage. Figure 3.2 shows the traditional linearized model for a moving coil electrodynamic transducer adapted from [46, 47]. Applying Black's law to the feedback system of the figure, the transfer function from input force to output voltage is

\[ G(s) = \frac{V(s)}{F_m(s)} = \frac{BlRs}{(ms^2 + B_ms + k)(Ls + R + R_c) + (Bl)^2 s} \]
\[ = \frac{BlRs}{Lms^3 + B_m L s^2 + (kL + B_m (R + R_c) + (Bl)^2 s + k(R + R_c)} \tag{3.6} \]

where \( F_m(s) \) is the Laplace transform of the forcing function.
Figure 3.2 is a third-order system in feedback, with one pole set by the electrical parameters and the other two by the second-order mechanical system. For applications where the mechanical frequency is very low, like human movement vibrations, the electrical pole can be ignored to first order. The system then reduces to a second-order mechanical system with a damped sinusoidal impulse response:

\[ G(s) = \frac{BIRs}{(R + R_c)(ms^2 + \left[B_m + \frac{(Bl)^2}{R + R_c}\right]s + k)} \]  \hspace{1cm} (3.7)

The proof-of-concept generator tested in Section 3.2 exhibits this primarily second-order behavior.

### 3.1.2 Nonlinear Model

The linear model described above is a great simplification of the actual dynamics for the moving-coil transducer system, which is in reality nonlinear. The primary nonlinearities are can be lumped into the stiffness parameter \( k \) of the spring and the nonlinear electromagnetic coupling \( Bl \) \[48\]. In general, these parameters are functions of the coil displacement \( z \) and are represented as second-order polynomials,

\[
\begin{align*}
  k(z) &= k_0 + k_1z + k_2z^2 \quad \text{(3.8)} \\
  Bl(z) &= Bl_0 + Bl_1z + Bl_2z^2 \quad \text{(3.9)}
\end{align*}
\]

where \( k_0 \) and \( Bl_0 \) are the nominal values from the linear model described above. It is more convenient to represent these in the following form:

\[
\begin{align*}
  k(z) &= k_0[1 + \mu_2(z - z_{02})^2] \quad \text{(3.10)} \\
  Bl(z) &= Bl_0[1 - \mu_1(z - z_{01})^2] \quad \text{(3.11)}
\end{align*}
\]

where \( \mu_1 \) and \( \mu_2 \) are nonlinearity coefficients that determine the width of the parabolas. The sign difference accounts for the fact that stiffness increases nonlinearly with displacement while the magnetic coupling decreases. \( z_{01} \) and \( z_{02} \) characterize the asymmetry of the parabolas relative to the resting position of the mass.

Introducing the nonlinearities described above into Equations 3.4 and 3.5 yields two coupled nonlinear differential equations,

\[
\begin{align*}
  f_m &= m\frac{d^2z}{dt^2} + B_m\frac{dz}{dt} + k_0[1 + \mu_2(z - z_{02})^2]z \\
  &\quad \quad \quad + Bl_0[1 - \mu_1(z - z_{01})^2] \frac{v}{R} \quad \text{(3.12)} \\
  Bl_0[1 - \mu_1(z - z_{01})^2] \frac{dz}{dt} &= \frac{L}{R} \frac{dv}{dt} + \left(\frac{R_c}{R} + 1\right)v. \quad \text{(3.13)}
\end{align*}
\]
where all parameters are the same as above. For certain choices of parameters, the system of equations 3.12 and 3.13 can be solved analytically, but in general numerical methods are required.

3.1.3 Optimal Power Generation

Given the models described above in Sections 3.1.1 and 3.1.2, it is possible to determine the output power as a function of the moving-coil transducer parameters. By assuming a second-order system response and matching the resulting parameters to those in Equation 2.13, it is possible to write the output power $P_{out}$ in terms of the transducer parameters:

$$P_{out} = \frac{k^2 Y_0^2}{2 \left[ B_m + \frac{(Bl)^2}{R} \right]}$$ \hspace{1cm} (3.14)

where $Y_0$ is the input displacement amplitude, $m$ is the system mass, $k$ is the spring constant, $B_m$ is the coefficient of mechanical damping, $R$ is the load resistance, and $Bl$ is the electromechanical coupling.

Equation 3.14 shows a clear tradeoff. To maximize the output power, it is necessary to minimize the damping by reducing the mechanical damping $B_m$ or the electrical damping $Bl$. This can be achieved up to the maximum tolerable displacement as discussed in Section 2.4.5. However, reducing the electrical coupling coefficient also reduces the output voltage directly as seen in Equation 3.6. Therefore, to maximize output power two design constraints must be met:

- The transducer resonant frequency should be tuned to the input vibration frequency.
- The damping coefficient must be minimized subject to the minimum of two constraints: a maximum allowable displacement $Z_{max}$ or a minimum output voltage which sets a minimum value of $Bl$.

When the vibration frequency is unknown or the signal is broadband in nature, the transfer function must also be broadened by increasing the damping coefficient. This must be balanced against the requirement for a minimum output voltage. This voltage is set by the rectifier diodes or the DC/DC converter that creates the load circuit power supply.

3.2 Model Verification

A prototype generator following the design of Figure 3.1 was built using discrete components. The generator was tested by giving the mass an initial displacement and then releasing it. The coil load resistance was 10 $\Omega$. The resulting voltage output waveform is shown in Figure 3.3 and compared to the predicted results from the fitted linearized model of Equation 3.6. The model captures the basic behavior of the system. Figure 3.3 shows the
Figure 3.3: Model fits of two generator configurations using the simplified linear transducer model.
fit of the linear model to measured data for two configurations. The model was assumed to be reduced to second-order. The data was fit using nonlinear least squares Gauss-Newton recursive estimation [49] for the spring constant. In particular, for a mass \( m = 0.5 \) g and a spring constant \( k = 174 \) N/M, the resulting natural frequency \( f_0 = 94 \) Hz, which matches the frequency of the measured output well. Note that this is much less than the electrical pole which was found to be at 1.2 kHz, validating the second order assumption. The model also accounts for the effect of mass. A larger mass should result in a lower resonant frequency and consequently a lower output voltage, as shown in Figure 3.3(a) for \( m = 20 \) g.

There are several things to note in Figure 3.3(b). First, the peak output voltage of the generator is only 180 mV, too small to be rectified by a diode. A transformer is thus necessary to create a large enough voltage from the generator output to be converted by the regulator. Although it is possible to use a diode with a lower turn-on voltage, leakage currents become a problem. Synchronous rectification is also possible but requires sensing of the generator output and rapid switching of the rectifying transistors since the generator voltage varies quickly. A transformer is the simplest electrical solution to this problem.

There are also significant differences in the linear model and the measured output. The RMS error between simulated and measured responses is 15 mV. The errors are due to unmodeled nonlinearities in the spring and dashpot, magnetic field variation vertically and in the air gap, and higher order vibrational modes of the system including twisting and flexing of the mass. However, this simple model provides a reasonable baseline for estimating power generation capability and system design. More sophisticated modeling using the nonlinear differential equations of Section 3.1.2 should be done as future work to learn more about this system.

### 3.3 Generator System

Figure 3.4 is a diagram of the generator and rectifier subsystem. The moving coil generator is constructed using commercially available components. The mass and spring can
Figure 3.5: Measured generator output waveforms for a mechanical impulse.

be changed easily to allow experimentation with the mechanical parameters of the system. As can be seen from Figure 3.3(b), the output voltage of the generator is only 180 mV, which is not sufficient to turn on a diode in a rectifier. Transformer \( X_1 \) (with a 1:10 turns ratio) converts the output voltage of the generator to a higher voltage that can be rectified by the half-wave rectifier formed by diode \( D_1 \) and capacitor \( C_1 \). Note that with proper electromechanical design, the transformer can be eliminated (see Section 3.1.3). In Equation 3.7, increasing the strength of the permanent magnet field \( B \) or the length of the coil \( l \) results in increased voltage. This also results in increased damping and potentially less power out of the generator. Voltage \( V_{in} \) is the input voltage to the regulator.

Measured output of the generator is shown in Figure 3.5. A mechanical impulse is given to the transducer which results in an oscillation at the natural frequency, about 1 kHz in this example. As the diode \( D_1 \) is forward biased, energy is drawn from the vibrating mass and dumped as a charge packet onto the rectifying capacitor. This accounts for the steps in voltage \( V_{in} \) at the zero crossings of the \( V_{gen} \) waveform. \( V_{out} \) is output of the DC/DC converter described in Chapter 4.

### 3.4 Variable Capacitor Generator Design

An alternative transducer to the moving coil generator is a variable capacitor. Electric charge is stored on the capacitor at some initial time. The capacitor plates are then moved, which requires mechanical work to be done. This increases the electrical energy stored in the system. Since this approach requires no permanent magnetic materials and variable capacitors are a well understood MEMS technology, this approach makes it attractive for
implementing a micromachined power generator. The MEMS device design and analysis is carried out by Oscar Mur-Miranda under the supervision of Prof. Jeff Lang [50].

3.4.1 MEMS Transducer Design

There are two different variable capacitor structures which can be implemented in a MEMS technology. Figure 3.6(a) shows a variable area capacitor, where the mechanical input slides one capacitor plate by the other, changing the area. Since the capacitor gap is fixed, the charge varies linearly with the capacitor area to keep the voltage fixed as the plates move. This geometry lends itself naturally to the voltage-constrained conversion cycle.

Figure 3.6(b) shows a variable gap capacitor, where the mechanical input changes the separation between the capacitor plates. Since the charge does not vary in this configuration, it lends itself to the charge-constrained energy conversion cycle. Either conversion cycle can be performed with either capacitor geometry, but the power electronics for the voltage-constrained cycle are more complex than for the charge-constrained cycle. However, the former produces much more energy. The control electronics tradeoffs between these approaches is evaluated in [51].

The actual MEMS transducer being constructed is shown in plan view in Figure 3.7. It is constructed using deep reactive ion etching and wafer bonding technology as can be seen in the cross-sectional view shown in Figure 3.8. The device itself is fabricated in the upper device wafer while the lower handle wafer provides mechanical support. A thin layer of silicon dioxide serves to bond the wafers. The parasitic capacitance created by this layer can be used to improve the energy conversion cycle as will be shown in Section 3.4.4.

The central mass of the transducer is connected to a two bar linkage mechanism anchored in turn to the handle wafer at the anchor points shown in Figure 3.7. Etching frees the mass to move, but the linkage mechanism restricts this movement in the lateral direction only. Fingers are etched into the mass and these are interdigitated with fingers etched onto a fixed comb. As the system is vibrated, the motion couples into the mass and the fingers of the mass slide past the fingers of the comb, creating a variable area capacitance
as described in Figure 3.6(a). The maximum and minimum capacitances are determined by the number of fingers, the surface area overlap of the fingers, and the maximum and minimum spacing in the gap at the extremes of the proof mass deflection. A Teflon-like insulating coating is added to the fingers during fabrication which prevents short circuits should the fingers accidentally touch.

The symmetrical layout of the MEMS device implies that two variable capacitors are formed using two edges of the proof mass and two stationary combs. As one edge moves toward its corresponding comb, the other moves away, and so the capacitances vary exactly out of phase with each other. By having two sets of power electronics, both capacitors can be used in conversion cycles a half-cycle out of phase with each other. This creates twice the output energy for a steady-state vibration.

3.4.2 Charge Constrained Energy Conversion Cycle

Figure 3.9 shows the two energy conversion cycles for the variable capacitor: charge-constrained and voltage-constrained. In the charge-constrained cycle, the amount of charge
placed on the capacitor is fixed, while the voltage-constrained approach fixes the capacitor voltage while the plates are moved. The maximum voltage on the capacitor, $V_{\text{max}}$, is a hard constraint set by the breakdown field of the gap dielectric. Consider the charge-constrained cycle first, triangle ABD in the figure. Initially, the capacitor is at its maximum value and charged to an initial charge $Q_0$. This is line AB in the Q-V plot. The capacitance is then reduced as the plates are moved, but the charge is fixed so the voltage increases to its maximum value. This transition corresponds to line BD. Then, the charge is removed from the capacitor at its minimum value and voltage $V_{\text{max}}$. The total energy put into the system is the area of triangle ABD.

This result can be shown mathematically by examining the energy stored in the capacitor [44]. The energy as a function of position and charge is:

$$E_C(Q) = \frac{Q^2}{2C(x)}$$  \hspace{1cm} (3.15)

where $E_C(Q)$ is the capacitor energy, $Q$ is the capacitor charge, and $C(x)$ is the capacitance as a function of the position. Considering only one finger pair of the transducer, the expression for the capacitance is:

$$C(x) = \frac{\varepsilon w (l - x)}{h}$$  \hspace{1cm} (3.16)

where $\varepsilon$ is the permittivity of the capacitor dielectric material, $w$ is the width of the finger, $l$ is the length of the finger at maximum overlap, $x$ is the current position of the
finger with 0 corresponding to maximum overlap, and \( h \) is the gap length between the fingers. As the expression shows, the area of the capacitor varies with position.

The amount of work done by the mechanical vibration must equal the change in energy stored in the capacitor if we assume that there are no loss mechanisms like electrical or mechanical damping. This \( \Delta E_Q \) corresponds to the difference in energy between points D and B in Figure 3.9. The energy difference is

\[
\Delta E(Q) = \frac{1}{2} Q_0 (V_{\text{max}} - V_{\text{start}})
\]

\[
= \frac{Q_0^2 h}{2 \varepsilon \omega} \left( \frac{1}{l - x_f} - \frac{1}{l} \right)
\]

where \( Q_0 \) is the initial charge, \( V_{\text{start}} \) the initial voltage, and \( x_f \) is the final position determined by the maximum tolerable voltage. All other parameters are as defined above. Equation 3.17 clearly corresponds to the area of the triangle ABD.

### 3.4.3 Voltage Constrained Energy Conversion Cycle

Since \( V_{\text{max}} \) is the hard constraint, we see that the voltage-constrained cycle ACD generates substantially more energy. The capacitor starts off with the maximum amount of charge allowable by the breakdown voltage. During the initial charging phase, AC each charge packet was placed at a different, increasing voltage. During the mechanical work phase CD, the charges are removed at the same, higher voltage \( V_{\text{max}} \), which accounts for the increased energy in the system. Any residual energy is then removed along line DA. The energy for this conversion cycle corresponds to area ACD.

This result is somewhat more difficult to show mathematically. Since the voltage is fixed in this cycle, it is convenient to examine the coenergy:

\[
E^*_C(V) = \frac{C(x)V^2}{2}
\]

\[
= \frac{\varepsilon \omega (l - x)V^2}{2h}
\]

where \( E^*_C(V) \) is the capacitor coenergy, \( V \) is the capacitor voltage, and \( C(x) \) is the same capacitance as a function of the position as described above. To determine the work done on the capacitor, it is necessary to find the force exerted by the capacitor electric field. This can be found from the differential coenergy,

\[
dE^*_C = \frac{\partial E^*_C}{\partial x} \bigg|_V dx + \frac{\partial E^*_C}{\partial V} \bigg|_x dV
\]

\[
= -\frac{\varepsilon \omega V^2}{2h} dx + \frac{\varepsilon \omega (l - x)V}{h} dV
\]

(3.19)
where the variables are as defined above. The coefficient of the $dx$ term in Equation 3.19 is the force $F_e$ exerted by the electric field. The work done by the field as the system moves from point C to point D in Figure 3.9 is therefore

$$W_e = \int_{0}^{x_f} F_e dx$$

$$= -\frac{\varepsilon w x_f V_{max}^2}{2h}$$

(3.20)

where the parameters are as defined earlier. Notice that the electric field does negative work during this system transition, indicating that energy has come out of the capacitor as the plates have moved apart. By conservation of energy, the vibration source must have done the identical amount of work to move the charge from the MEMS capacitor back to the voltage source. It is easy to show that this work is equal to the area of triangle ACD.

### 3.4.4 Modified Charge Constrained Energy Conversion Cycle

Figure 3.10 shows an alternative approach to energy conversion. In this system, another capacitor of constant value $C_{par}$ has been added in parallel to the MEMS variable capacitor. Path abda represents a charge-constrained conversion cycle without $C_{par}$ while path ab'd'a includes $C_{par}$. The impact of this additional capacitor can be seen by rewriting Equation 3.17 in terms of the initial voltage $V_{start}$:

$$\Delta E(Q) = \frac{\varepsilon w V_{start}^2 x_f}{2h} \frac{l}{l - x_f}.$$  

(3.21)

The addition of the extra constant capacitance is effectively to make the variable term $x_f$ small compared to the initial capacitance term $l$. In the limit as $l \gg x_f$ and $V_{start}$ approaches $V_{max}$ (as shown in Figure 3.10 as $V_{start}$ goes from $V_1$ to $V_2$), Equation 3.21 approaches in magnitude Equation 3.20 and more energy is obtained from the system. $C_{par}$ helps hold the voltage constant in the system and converts the charge-constrained cycle to an approximation of the voltage-constrained loop.

Alternative formulations of the energy conversion cycles are discussed in detail in [51].

### 3.4.5 Power Electronics

Figure 3.11 shows the power electronics for the charge-constrained energy conversion cycle. The voltage-constrained version is more complicated, but Section 3.4.4 showed how one can approach the voltage-constrained energy limit using a large parasitic capacitor. Capacitor C0 is a large charge reservoir (it could also be a rechargeable battery). Charge must be placed on the variable capacitor C1 before its plates can be moved. This is accomplished by closing transistor N0 and ramping up the current in inductor L0. Then,
Figure 3.10: Modified energy conversion with $C_{par}$.

N0 is opened and P0 is closed, so the inductor energy is moved onto the variable capacitor C1 as charge. C1 is in its maximum capacitance geometry. After it is charged to $Q_0$, switch P0 is opened and the plates are allowed to move. Work is done until the capacitance reaches its minimum value (or its voltage reaches $V_{max}$). Then, switch P0 is closed again and the energy moves from C1 to L0. After the C1 discharges, P0 opens and N0 closes. The inductor returns its energy to the reservoir C0 and the cycle is ready to begin again.

A MEMS based implementation using this approach has the potential to generate 10's of $\mu W$ of power. The circuit challenge is to design the ultra-low power control electronics for timing the switches correctly, so that power can be delivered efficiently to the load.

Figure 3.12 shows the time-varying circuit topology as the conversion cycle is performed. A detailed discussion of the circuit design, including timing waveform generation using open loop and feedback techniques as well as the sizing of the power transistors is discussed in [51].
3.4. VARIABLE CAPACITOR GENERATOR DESIGN

Figure 3.11: Power electronics for charge-constrained conversion cycle.

Figure 3.12: Circuit topology in various states during energy conversion.
Chapter 4

DC/DC Converter for Self-Powered Systems

Portable systems that depend on batteries have a limited operating life and are prone to failure at inconvenient times. We propose a system as in Figure 4.1, consisting of a generator to create a DC voltage, a DC/DC converter to set the voltage to meet the desired performance, and a DSP which performs some computation. The desired voltage is set by the rate at which the DSP is to produce results.

Since we are interested in generating power from ambient vibration, \( V_{in} \) is a time varying voltage. Its variations occur over much faster time scales than the depletion of a battery charge. To maintain correct functionality at a specified performance level, this input voltage must be regulated to a desired value before it can be used to power a load circuit. This is done using a very low power DC/DC converter. The generator system is designed to produce a high enough voltage that only a down (Buck) converter [52] is required. These converters have successfully been used in low power battery-based applications [53]. This chapter describes the design and implementation of a very low power DC/DC switching converter appropriate for use with a generator system.

4.1 Chip Architecture

The converter consists of five main subsystems: a VCO, frequency comparator, pulse-width modulated (PWM) waveform generator, bootstrap detection circuit, and a Buck converter. Figure 4.2 shows the converter architecture, similar to [7], [8]. An external input provides the performance constraint: the rate at which the load circuit must produce results. The load circuit in this case is an 8 bit FIR filter. The DSP rate command is delivered in the form of a clock, \( f_{clk} \), whose period corresponds to the total delay between valid output samples of the DSP. To achieve the lowest possible power consumption, the converter downconverts \( V_{in} \) to the lowest voltage at which the DSP can run and still produce correct results at the rate set by \( f_{clk} \).
Figure 4.1: System block diagram.

Figure 4.2: Detailed block diagram of self-powered DSP system.
The overall control scheme is similar to a phase-locked loop. The rate $f_{clk}$ is compared to the output of a voltage-controlled oscillator, $f_{vco}$. The VCO is a ring oscillator consisting of the DSP adder critical path padded with a few inverters and is supplied by the regulated output voltage $V_{out}$. Thus it is a replica of the circuit whose power supply is being controlled with some delay margin to account for processing mismatches. The controller adjusts $V_{out}$ until the period of $f_{vco}$ is short enough to satisfy the performance demand but not so short as to waste power. The error is represented as a two bit digital signal produced by the frequency compare block in Figure 4.2.

The converter is a Buck converter with very small P and N FETs (1200 µm and 300 µm, respectively) controlled by a pulse-width modulated waveform, $D(t)$. The PWM block generates $D(t)$ with 6 bit resolution, i.e. 64 different duty cycles. A ring oscillator running at voltage $V_{dd}$ sets the frequency of the modulation signal. A new duty cycle is determined from the frequency error at a rate set by a programmable counter. By initializing this counter with different values, the dynamics of the controller can be adjusted. The modulated $V_{in}$ signal is then passed through an LC lowpass filter, external to the chip, to produce $V_{out}$. The FIR filter digital load is a well-known subband filter [54].

Power for the controller, voltage $V_{dd}$ in the figure, initially comes from a backup voltage source. The bootstrap detect block switches the controller to $V_{out}$ when that voltage is deemed stable. Many tradeoffs between performance and power consumption were made in the design of the converter and are discussed in the following sections.

### 4.2 Low Resolution Digital Control of Buck Converters

Implementing an efficient DC/DC converter at the delivered power levels required by low power digital systems implies that the control power for performing closed-loop voltage regulation must be kept at a minimum. For delivered power on the order of 100's of µW to a few mW, controller power must be far below levels which are typical for big power supplies. The low power requirement makes analog feedback compensation, which usually needs steady-state current draw to bias amplifiers, undesirable. Moreover, digital techniques which attempt to mimic the performance of an analog system may also require too much power, for example in the A/D conversion or in the high resolution digital datapath.

An alternative approach is to implement a digital compensator using as little feedback resolution as possible, i.e. one bit. This section analyzes the impact of such a feedback scheme and makes recommendations on designing them to achieve a certain level of performance.

#### 4.2.1 Analog Linear Compensation

Figure 4.3 is a block diagram of the buck converter under feedback control. The transfer function of the output lowpass filter is a typical second order system. The square wave signal $V_{s}(t)$ that is filtered is generated by the PWM generation block. This block takes a
Figure 4.3: Block diagram of buck converter system.

commanded duty cycle \( d(t) \) which in steady state should be the ratio between the unregulated input voltage \( V_{in} \) and the desired output voltage \( V_{ref} \). The compensation network consists of a linear block \( K(s) \) and a nonlinear block \( N(X,s) \).

Consider the fully linear system \( (N(X,s) = 1) \). To eliminate steady-state error, we will use an integral compensator with gain \( K \):

\[
K(s) = \frac{K}{s}.
\]  

(4.1)

This leads to the following open loop transfer function:

\[
L(s) = \frac{K}{s(s^2 + \frac{s}{RC} + \frac{1}{LC})}.
\]  

(4.2)

Equation 4.2 indicates that there are three poles, one at the origin and two more in the left half plane, so the system is open loop stable. Figure 4.4 shows the root locus as the feedback gain \( K \) is varied. As \( K \) increases, two of the poles move toward the \( j\omega \)-axis and the system responds more quickly to step changes at the input at the expense of greater ringer. For some value of \( K \) the poles move into the right half plane and the system is unstable. To compensate the system, simply choose a gain that achieves the desired transient performance, making sure that it isn’t so large that it destabilizes the converter.

4.2.2 Theory of Describing Functions

Nonlinear systems do not have a strictly correct transfer function since a sinusoidal excitation of frequency \( \omega_0 \) at their inputs may produce outputs with frequencies different than \( \omega_0 \). Describing functions are approximations to the true frequency response of a nonlinear element which are obtained by considering the Fourier series of the output to a sinusoidal input. [55] Suppose we excite a nonlinear element \( N \) with an input \( x(t) = X \sin \omega t \). Then, the fundamental term of the output is

\[
y_1(t) = F(X, \omega) \sin[\omega t + \phi(X, \omega)].
\]  

(4.3)
Figure 4.4: Integrator compensation system root locus.

The describing function is defined as follows:

\[ N(X, \omega) = \frac{F(X, \omega)}{X} \phi(X, \omega). \]  \hspace{1cm} (4.4)

The describing function approximation is valid if two assumptions are met:

1. The input to the nonlinear block \( N \) is a pure sine wave.
2. The output contains no DC or subharmonic terms and the higher order harmonics are not of interest.

In feedback situations where the output of \( N \) goes through some linear blocks and then is fed back to its input, these assumptions require the output to go through some lowpass filtering before reaching the summing node.

Finally, stability may be analyzed in terms of Nyquist's criterion. Suppose the open loop transfer function of the system is \( L(j\omega) = N(X, j\omega)G(j\omega) \). The poles of the closed loop system are then given by

\[ 1 + N(X, j\omega)G(j\omega) = 0, \]  \hspace{1cm} (4.5)
Figure 4.5: One bit error constitutive relation.

which can be rewritten as

\[ G(j\omega) = \frac{-1}{N(X, j\omega)} \]  \hspace{1cm} (4.6)

Stability is then analyzed by comparing \( G(j\omega) \) in the complex plane to the loci of \(-1/N(X, j\omega)\).

4.2.3 Single Bit Error Feedback

A single bit feedback scheme can be implemented by using a comparator to sample the error \( V_{err} \). The comparator then outputs a +1 if \( V_{err} \) is positive and a -1 if it is negative. Figure 4.5 shows the nonlinear constitutive relation of the comparator. To analyze the feedback system with this nonlinearity, we apply the theory of describing functions to determine the transfer function \( N(X, j\omega) \) [55]:

\[ N(X, j\omega) = \frac{4}{\pi X} \]  \hspace{1cm} (4.7)

where \( X \geq 0 \) is the amplitude of the exciting sinusoidal function. Note that this describing function does not depend on frequency, which simplifies its use in stability analysis. It also has zero phase, so we can think of it as a variable gain in our open loop transfer function.

4.2.4 Continuous Time Analysis

The locus of points traced out for different values of \( X \) is shown in Figure 4.6, along with the Nyquist plot of the linear transfer function of the integral compensated system (\( L(j\omega) \) from Equation 4.2). This locus is simply the negative real axis. To analyze the stability, consider what happens to a small oscillation at the input of the closed loop system. For
4.2. LOW RESOLUTION DIGITAL CONTROL OF BUCK CONVERTERS

$X$ small, $-1/N$ is also small, and our operating point is inside the encirclements of $L(j\omega)$. This means that the operating point is unstable and small amplitude oscillations will grow, moving the operating point to the left along the $= 1/N$ curve. If $X$ is large, we start outside the encirclements and the operating point is stable, meaning the amplitude of the oscillation decreases with time. This in turn moves us to the right on the $-1/N$ curve. The intersection of $L(j\omega)$ and $-1/N$ is therefore a stable operating point and we expect the closed loop system to have a limit cycle (steady-state oscillation). The frequency of the oscillation is the frequency $\omega_0$ at which $\angle L(j\omega) = 180^\circ$. Its amplitude is obtained by setting $-1/N = L(j\omega)$ and solving for $X$. For our system,

$$X = \frac{-4L(j\omega_0)}{\pi} = \frac{4KRC}{\pi}. \quad (4.8)$$

To verify this continuous time analysis, the buck converter system was simulated using MATLAB. The resonant frequency of the converter was chosen to be 100 kHz and the load resistance $R = 10\Omega$. The compensator gain $K = 1000$ and the switching frequency of the PWM generation was 1 MHz. From these numbers, we expect the limit cycle frequency to be 100 kHz and the amplitude $X = 20.3$ mV. Figure 4.7 shows the $V_{out}$ limit cycle caused by having one bit error feedback and confirms the theoretically predicted frequency and amplitude. $V_{ref} = 1.1$ V for this simulation. Superimposed on the plot is the commanded output voltage $d(t)V_{in}$ which can be seen as a triangle wave. Since the cutoff frequency of the output filter is only a decade above the switching frequency of the PWM generation, there is substantial 1 MHz ripple on the output.

4.2.5 Digital Implementation

Implementing the control using a digital system requires quantizing the duty cycle command $d(t)$ and discretizing the time axis. A typical implementation is to use an up/down counter as the integrator and a comparator to generate the 1 bit error signal. One can think of the "gain" of such a system as

$$K_{DT} = \frac{\Delta d}{T_{cnt}} \quad (4.9)$$

where $\Delta d$ is the duty cycle resolution of the PWM generation and $T_{cnt}$ is the period of the counter clock. Since the input to the counter tells it to add or subtract 1 from its state every cycle and a 1 corresponds to a minimum change in the duty cycle, this equation is intuitively clear. For example, if the error is 1 for $T$ seconds, the counter output value should be $T/T_{cnt}$.

Equation 4.9 therefore yields a notion of how to understand the performance of the digital control system. For a given duty cycle quantization interval $\Delta d$, increasing the clock frequency results in a higher digital "gain" and can destabilize the system as the root locus diagram showed. Equation 4.9 can also be used as a design tool. If we have a continuous time compensator gain $K$ that produces a desired performance, we can simply solve for
the ratio of $\Delta d$ and $T_{cnt}$ using 4.9. However, this does not guarantee that the performance of the digital system approximates the continuous time one.

Figure 4.8 shows the steady-state limit cycle of the digital controller using 1 bit error feedback for a 10 bit quantization of the duty cycle. The sample rate $T_{cnt}$ was set such that the digital gain was equal to the analog gain of Figure 4.7. The performance is similar to the analog compensator although the effect of the quantization can be seen in the commanded output voltage signal (dashed line). There is also a lower frequency, small amplitude oscillation superimposed on the output. As the quantization is increased to 12 bits, the digital controller becomes almost indistinguishable from the analog implementation.

As the quantization level is reduced and the sample rate is decreased to keep the gain constant, the performance of the digital system starts to deviate substantially from the analog. Figure 4.9 shows the output voltage limit cycle using only 8 bits of resolution in the duty cycle. The amplitude of the cycle does not differ by much from the previous plots, but the frequency of the oscillation is not the same as predicted by the theory.

### 4.2.6 Single Bit Error with Dead Zone Feedback

Figure 4.6 shows that limit cycles are inherent for the single bit feedback case since the locus of the describing function spans the entire negative real axis. An alternative to this scheme is to allow some free play (a dead zone) in the error comparator so that it can output a 0 as well as ±1.
Figure 4.7: CT limit cycle for 1 bit feedback.

Figure 4.8: Digital limit cycle for 1 bit feedback (10 bit quantization).
Figure 4.9: Digital limit cycle for 1 bit feedback (8 bit quantization).

Figure 4.10 shows how addition of a dead zone affects the nonlinearity introduced by having one bit error feedback. There is a range of input values $\delta$ for which we decide not to change the duty cycle command to the PWM generation, implying that the output voltage is close enough to $V_{ref}$ for our needs. This approach can eliminate limit cycles at the expense of finite steady state error. This tradeoff may be acceptable, especially if the Buck converter powers analog circuits which cannot tolerate substantial ripple on their power supplies.

### 4.2.7 Continuous Time Analysis

Again, we can use the theory of describing functions to analyze the performance of a linear compensator with this type of nonlinearity. The describing function is

$$
N(X, j\omega) = \frac{4}{\pi X} \sqrt{1 - \left(\frac{\delta}{2X}\right)^2}
$$

(4.10)

where $\delta$ is the width of the dead zone. Again, since there is no frequency dependency or phase modification this nonlinearity may be treated simply as a variable gain. However, the locus of points $-1/N(X, j\omega)$ for Equation 4.10 is slightly more complex than for Equation 4.7. Consider the points of interest for stability analysis:
\[ \frac{-1}{N(X, j\omega)} = \frac{-\pi X}{4\sqrt{1 - \left(\frac{2\pi}{\delta X}\right)^2}}. \] (4.11)

For \( X < \frac{\delta}{2} \), the locus lies along the positive \( j \)-axis. For \( X \geq \frac{\delta}{2} \), \(-1/N\) is negative and real. However, there is a point on the real axis beyond which the locus does not go. Figure 4.11 shows the real part of \(-1/N\) for some values of \( \delta \) and \( X \) and we see that there is a maximum. Using elementary calculus, we can maximize Equation 4.11 with respect to \( X \) to yield the endpoint \( P_N \) of the locus on the negative real axis:

\[ P_N = \frac{-\pi \delta}{4}. \] (4.12)

Figure 4.12 shows the Nyquist plot for the compensated system with the \(-1/N\) locus. The locus does not extend all the way to the origin as in the previous case. If \( \delta \) is chosen such that \( \delta > \frac{-4L(j\omega)}{\pi} \), then \( L(j\omega) \) does not intersect the locus of the describing function and there is no stable limit cycle.

Figure 4.13 shows the simulated output for a system with a sufficiently large dead zone to eliminate the limit cycle. Note that there is a finite steady state error. When the dead zone is made smaller, the limit cycle reappears as the nonlinearity approaches the 1 bit feedback case. Figure 4.14 shows the output for a small dead zone. The limit cycle frequency is the same as the resonant frequency and the amplitude is determined in the same manner as above using the new describing function.

### 4.2.8 Digital Implementation

The digital implementation of this controller is also straightforward. A counter is used as the integrator as above, however the comparator design is more sophisticated as the dead
zone must be designed in. The same considerations regarding duty cycle quantization and sample rate as in Section 4.2.5 apply.

Figures 4.15 and 4.16 show the performance for a digital implementation under different values of $\delta$.

### 4.2.9 Low Resolution Design Considerations

Describing function techniques can be a powerful tool in the design of low resolution digital controllers. The first step is to design an analog compensation scheme, being careful to choose the gain of the system as low as possible to meet the required performance. This is important to save power in the later digital implementation. Once the analog gain is determined, Equation 4.9 can be used to determine the correct duty cycle resolution to sample rate ratio. It is important to keep these numbers as large as possible since making them smaller roughly corresponds to increasing the power consumed by the controller. Since the translation from continuous time to the digital domain is not exact, the digital system should be simulated to verify that it performs as desired.
Figure 4.12: Nyquist plot with describing function for 1 bit error with dead zone feedback.

Figure 4.13: CT output with 1 bit error feedback with large dead zone.
Figure 4.14: CT output with 1 bit error feedback with small dead zone.

Figure 4.15: Digital output with 1 bit error feedback with small dead zone.
4.2.10 High Resolution Digital Feedback

As more bits of resolution are added to the error sample, the digital system starts to approach a linear analog controller. Describing function analysis is unnecessary once the resolution is high enough. For power converters where the overhead of good A/D resolution and wide digital data is insignificant, this approach is best since there are no steady state limit cycles or steady state errors.

4.3 Controller Design

The controller block diagram is shown in Figure 4.17. This all digital feedback controller simplifies the design considerably and reduces power by eliminating constant current bias circuits. The tradeoff, as explained earlier, is the loss of performance. The power supply voltage created by this system has ripple levels that are probably beyond the tolerance of analog circuits. For low to medium throughput digital loads the only requirement is that the voltage remain above a certain level to meet the throughput constraints.


Figure 4.17: Digital controller block diagram.

Figure 4.18: Frequency comparator block diagram. The high order bit of the 2 bit counter clocked by $f_{clk}$

4.3.1 Frequency Comparator and Integrator

To close the controller loop using delay feedback, the reference clock signal $f_{clk}$ must be compared to the locally generated VCO clock, $f_{vco}$, which is powered from the output voltage of the converter. The Frequency Compare block of Figure 4.17 consists of a frequency comparator and some logic that generates the output signals Stay and Down. These two bits control the digital integrator, in this case a 6 bit up/down counter. The polarity of the Down signal determines whether the counter increments or decrements its current state. It does not roll over at the boundary conditions $0x0$ or $0xFF$. The Stay signal is simply a counter enable. This extra signal helps smooth limit cycles in the digital control since the state of the counter can be held steady instead of constantly either incrementing or decrementing.

Figure 4.18 shows a detailed schematic of the frequency error detector. It consists of two small counters. The 3 bit counter is clocked by $f_{vco}$ and the 2 bit counter by $f_{clk}$. Both are enabled by an external signal (not shown). After the counters are enabled, they increment on the rising edges of their respective clocks. After the output of the 2 bit counter, $Q0$
reaches 2, the three bit counter is disabled and the rising edge of \( Q0(2) \) latches the output of the three bit counter \( Q1 \) into a register. The register therefore stores how many positive edges of \( f_{vco} \) occurred between two edges of \( f_{clk} \). This value, \( N \), can be used to determine the relative values of the two frequencies.

![Figure 4.19: Frequency Comparator Timing Diagram](image)

Figure 4.19 shows a timing diagram describing the operation of the frequency comparator. Suppose that the reference signal \( f_{clk} \) has a fixed period \( T_0 \) with a corresponding frequency \( f_0 \). Similarly, \( f_{vco} \) has a period \( T_v \) (frequency \( f_v \)) and is offset from \( f_{clk} \) by some phase lag \( \phi \), where

\[
\phi \in [0, 2\pi)
\]  

(4.13)

The time interval of interest is spanned by two periods of the reference clock, starting from a positive edge at time \( t_0 \):

\[
t \in [t_0, t_0 + 2T_0)
\]

For convenience, set \( t_0 \) equal to 0. Let \( N \) be the number of counts reported by the counter clocked by \( f_{vco} \). Therefore there are \( N \) positive edges of \( f_{vco} \) in the time interval \([0, 2T_0)\). If the first edge occurs at time \( t_1 \), we have

\[
0 \leq t_1, t_1 + T_v, \ldots, t_1 + (N - 1)T_v \leq 2T_0
\]

which implies the following bounds on \( t_1 \):

\[
0 \leq t_1 \leq 2T_0 + (1 - N)T_v
\]

(4.14)

Signals \( f_{clk} \) and \( f_{vco} \) are square waves and can be decomposed into Fourier series. The times of the positive edges must therefore also satisfy a periodicity condition

\[
\frac{2\pi t_1}{T_v} + \phi = 2\pi k
\]

(4.15)

\[
\frac{2\pi (t_1 + T_v)}{T_v} + \phi = 2\pi (k + 1)
\]

\[
\vdots
\]

\[
\frac{2\pi (t_1 + (N - 1)T_v)}{T_v} + \phi = 2\pi (k + N - 1)
\]
where \( k \) is a nonnegative integer, \( k \in [0, 1, \ldots) \). Rearranging Equation 4.15 yields

\[
t_1 = T_v \left( k - \frac{\phi}{2\pi} \right) \tag{4.16}
\]

Equation 4.16 determines the allowable values of the initial period number \( k \). From the bounds on phase lag \( \phi \) (Equation 4.13), \( 0 \leq \frac{\phi}{2\pi} < 1 \). Using this result puts bounds on the value of \( t_1 \) related to \( k \):

\[
T_v (k - 1) \leq t_1 < T_v k \tag{4.17}
\]

Suppose \( k \geq 2 \). The left hand relation of Equation 4.17 implies \( t_1 \geq T_v \). This violates the assumption that the first positive edge occurs at \( t_1 \) since a previous edge must have happened at \( 0 \leq t_1 - T_v \), which is in the allowed time interval. Suppose \( k = 0 \); then the right hand relation of Equation 4.17 requires \( t_1 < 0 \), outside the allowed time range. Therefore, the only allowable value of \( k \) is \( k = 1 \). Substituting this \( k \) in Equation 4.16 and substituting the result into Equation 4.14 yields

\[
T_v \left( 1 - \frac{\phi}{2\pi} \right) \leq 2T_0 + (1 - N)T_v
\]

Solving the above relation puts an upper bound on the period \( T_v \),

\[
T_v \leq \frac{2T_0}{N - \frac{\phi}{2\pi}} \tag{4.18}
\]

There are exactly \( N \) positive edges of \( f_{vco} \) in the interval \( t \in [0, 2T_0) \). Therefore, we must have

\[
t_1 + NT_v > 2T_0
\]

We can construct a lower bound on the VCO period using steps similar to the manipulations above. The bound is then

\[
T_v > \frac{2T_0}{N + 1 - \frac{\phi}{2\pi}} \tag{4.19}
\]

Using Equations 4.19 and 4.18 and the fact that \( f_v = \frac{1}{T_v} \) and \( f_0 = \frac{1}{T_0} \), we can bound the VCO frequency using the count \( N \):

\[
\frac{N - 1}{2} f_0 \leq f_v < \frac{N + 1}{2} f_0 \tag{4.20}
\]

where we have used the bounds on \( \frac{\phi}{2\pi} \) as well.
### Table 4.1
Frequency comparator values and bounds on $f_v$.

<table>
<thead>
<tr>
<th>$N$</th>
<th>Frequency Bounds</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$0 \leq f_v \leq \frac{f_0}{2}$</td>
</tr>
<tr>
<td>1</td>
<td>$0 \leq f_v \leq f_0$</td>
</tr>
<tr>
<td>2</td>
<td>$\frac{f_0}{2} \leq f_v \leq \frac{3f_0}{2}$</td>
</tr>
<tr>
<td>3</td>
<td>$f_0 \leq f_v \leq 2f_0$</td>
</tr>
<tr>
<td>$\geq 4$</td>
<td>$\frac{3f_0}{2} \leq f_v$</td>
</tr>
</tbody>
</table>

Table 4.1 shows the ranges of frequency $f_v$ for different values of $N$. For $N \geq 3$ the system meets the specification guaranteeing $f_v \geq f_0$. However, a frequency error of $N \geq 4$ implies that the VCO is running too fast. The controller asserts Down in this case to reduce the duty cycle. If $N = 3$ the controller disables the counter and the duty cycle is unchanged. The $f_v \geq f_0$ constraint can only be violated if a counter decrement reduces the voltage output by a large enough amount that $f_v$ becomes too slow. Careful selection of the duty cycle resolution can prevent that circumstance. Note that having the controller drive the frequency error to a constant value means that some power is inevitable wasted, i.e. the load circuit runs at a higher voltage than the ideal. Since the design operates at very low voltages, near the edge of subthreshold MOSFET conduction, small supply voltage changes result in large shifts in circuit delay. Thus, even if the VCO runs at a frequency $2f_0$, there is very little extra power dissipated.

Lastly, there is a metastable state the frequency comparator can enter if the outputs of the 3 bit counter $Q1$ are changing exactly during a rising edge of $Q0(2)$. The register outputs pass through some logic to determine the counter control signals. The up/down counter state is not updated until one cycle of the controller FSM later; so since this clock has a relatively long period there is a good chance that the register outputs (or at least the counter control signals) have settled before the next clock. It is therefore not necessary to have multiple latches of the register state to deal with the metastability problem.

#### 4.3.2 Duty Cycle Generator

Figure 4.20 shows a combined view of the Frequency Compare block and the PWM waveform generator. The PWM block generates $D(r)$ with 6 bit resolution, i.e. 64 different duty cycles. A free running ring oscillator running at the controller voltage $V_{dd}$ sets the frequency of the modulation signal. One fixed edge of the square wave is used to set an R/S flip-flop. A 64:1 mux selects an edge delayed by a discrete amount from the reference edge to reset the flip-flop. Using the output of the flip-flop as the modulating waveform $D(i)$, we can control the duty cycle by changing the mux selection bits. This delay-line based duty cycle generation is different from previous digital techniques which use a fast clocked counter [8]. Figure 4.21 shows a block diagram of this type of PWM generator. The digital duty cycle value $d[k]$ is digitally compared to a counter output $x[k]$, with the output of the comparator taken as the modulating waveform $D(r)$. The counter is clocked...
at \( K \) times the PWM waveform frequency, where \( K \) is the duty cycle resolution. When \( x[k] > d[k] \), the comparator output goes low producing the falling edge of \( D(t) \). When the counter overflows, \( x[k] = 0 \) and the comparator goes high, generating the rising edge. This approach is very similar to classic analog PWM, but consumes a lot of power because of the high clock rate for the counter.

The oscillator is free running in this implementation, so as the controller power supply changes (by being switched from the backup voltage \( V_{bh} \) to the regulated output \( V_{out} \), for example), the frequency of the PWM waveform changes. It can also change due to variations in process and environmental conditions, but these changes are slow compared to the voltage switching. As long as the design maintains this frequency much higher than the LC filter cutoff for all potential supply voltages, this frequency variation is not a major issue for the performance of the switching converter. A variation on this for generating the PWM waveform is to fix the oscillation frequency via a delay-locked loop (DLL). The feedback can fix the frequency of \( D(t) \) independent of variations in the controller supply voltage \( V_{dd} \), for example by proper control of current starved inverters in a ring oscillator. Using the lowest possible frequency for the PWM waveform, just high enough above the cutoff frequency of the LC filter to reduce output voltage ripple to an acceptable value, ensures that the lowest power dissipation is achieved by the PWM waveform. Although the implementation in this system may be less power efficient than a fixed frequency PWM generator using a DLL, it does have reduced complexity by eliminating the DLL control loop. When the overhead of implementing the DLL is considered, it is not clear which approach has the lowest power.

### 4.3.3 Sample Rate

As shown in Figure 4.17, the sample rate for the frequency comparator is set by shifting in a configuration word and loading it into a counter. This is analogous to setting a gain in the feedback loop (see Section 4.2). The system under control here is very nonlinear, in contrast to the examples discussed in Section 4.2, because the VCO frequency \( f_{vco} \) is a
Figure 4.21: Fast clocked counter PWM waveform generator. The duty cycle $d[k]$ is digitally compared to the counter output $x[k]$ and the comparator output is used as the PWM waveform.

highly nonlinear function of the converter output voltage $V_{out}$ (see Figure 6.7 for an example). The linear results derived in Section 4.2 are not directly applicable to this situation so the method of choosing the sample rate $T_s$ is ad hoc.

4.4 Datapath Load Circuit

Chapter 1 asserts that techniques at all system levels can be applied to reduce the power consumption of digital CMOS circuits. To determine if these techniques can reduce the power of useful digital systems to the point where they can be powered from ambient energy sources, a test circuit based on a design in the literature was fabricated along with the voltage converter described above. Experiments could then be performed evaluating the number of useful cycles the circuit could operate based on single excitations of the generator. In addition, the performance of the control loop could be evaluated in terms of disturbance rejection and power consumption.

4.4.1 Subband Filter

The recent explosion of interest in portable multimedia systems has led to a lot of research in low power techniques for DSP. In particular, image coding and compression has received a lot of attention [54]. A modified version of a lowpass subband filter used in a 1.2mW video-rate subband decoder [54] is used as the load circuit in our self-powered DSP system. This design incorporates many key low power design techniques: aggressive voltage scaling ($1$ $V_{dd}$), a short length (4-tap) FIR filter, and coefficient selection that allows a multiplierless implementation. The filter performs comparably to a 9-tap quadrature mirror filter (QMF) but consumes one third the power [54]. The impulse and frequency response of the lowpass subband filter implemented on-chip is shown in Figure 4.22. Clearly it has a lowpass characteristic with a fairly wide transition band.

Figure 4.23 shows the block diagram for the filter. The delay line only contains four elements and since the filter has a finite impulse response there is no feedback of the output...
Figure 4.22: Discrete-time FIR subband filter, impulse and frequency response.

sample. The four gain blocks are simple enough that they can be implemented using only adders, shifts, and inversions. The input and output samples are in 8 bit 2's complement notation and overflow is simplistically handled by truncating the high order bits. Since the gains are implemented using adders, the filter contains a total of five 8 bit full adders, three 8 bit static registers for the delay line, and one 8 bit output register. The circuit must function at extremely low voltages, near the edge of subthreshold MOSFET operation, so a static CMOS design style is the best choice for the adders. The particular circuit implementation lends itself to a convenient, symmetric layout as the pullup and pulldown trees are complementary [56]. The transistor sizes are chosen to minimize capacitive loading, since there are few elements in the critical path delay the speed of the adder circuits is not as important as power consumption. The registers consist of resettable edge-triggered flip-flops implemented in a true single-phase clock (TSPC) latch style [57]. The dynamic latches are staticized using weak feedback devices on the output. Careful simulation of the design at the intended supply voltages ensures that the glitching behavior inherent in this circuit does not occur for typical operation. Level converting circuits on the filter input translate the high voltage signals from the input pads to the low voltage DSP inputs. Similar level converters take the filter outputs to the pad voltage.

4.4.2 Voltage Controlled Oscillator

From the discussion above, it is clear that the control loop must be closed around some notion of performance of the load circuit. In this case the rate at which output samples are
produced is chosen as the performance metric. An oscillator whose period is tuned to the delay of the critical path of the subband filter provides the delay measurement since its frequency is compared to the reference clock (see Section 4.3.1). This critical path consists of a 10 bit adder delay from the carry in of the lowest order bit to the sum output of the highest order bit. An odd number of inverters is added after this output bit to allow some delay margin, ensuring that the VCO period is always longer than the actual critical path. The output of the last inverter is fed back to the carry in to form the ring.

The combined layout for the datapath load is shown in Figure 4.24. Simply using the same logic gates as the critical path is not enough to ensure good matching of the oscillator to the delay. This design uses the same layout for the oscillator as for the actual datapath, so interconnect loading and delays are incorporated into the VCO period. The devices are placed near each other to improve the matching of the delay.

4.5 Power FET Switches and Drivers

The power FETs for this particular application are fairly small because the load current requirements are very low. There is a tradeoff in the sizing of the output FETs for Buck converters because power is dissipated in driving the gates and in the on resistances of the transistors [58]. As the width of the transistors is increased, the resistance decreases. The gate capacitance increases and can offset the series resistance power savings. The FETs for this implementation were optimized for the load current of the subband filter DSP. The PFET SW0 is 1.2 mm wide while the NFET SW1 is 0.3 mm; these are small even for low power DC/DC converter applications.

4.6 Bootstrapping and Other Issues

The bootstrapping circuit is shown in the top half of Figure 4.25. At startup, the controller supply $V_{dd}$ receives its power from the backup source $V_{bk}$, either a battery or a capacitor. After the controller has converted $V_{in}$ to the desired level, BOOT goes high and $V_{dd}$ is con-
Figure 4.24: Subband filter and VCO layout. The VCO layout is placed near the DSP for good delay matching. Its shape is also copied from the DSP, and so interconnect delays are incorporated into the delay of the oscillator.
Figure 4.25: Bootstrap Circuit Schematic.

needed to the generated voltage. If $V_{out}$ drifts too far from the desired value the controller will switch itself back to the backup supply.

4.7 VLSI Implementation

A die photo of the implemented DC/DC converter chip is shown in Figure 4.26. It is fabricated in a 0.8 $\mu$m CMOS process. The entire die measures 2.6 $mm$ by 2.6 $mm$. The layout is clearly pad limited, with the core area (controller and DSP load) consuming 885 $\mu m$ by 985 $\mu m$. The DSP in the lower left hand corner of the core is the most dense layout because of its regularity. It measures 420 $\mu m$ by 386 $\mu m$. The component is packaged in a 40-pin DIP. Testing and evaluation of the entire self-powered system, including the DC/DC converter and the moving-coil generator prototype is described in the next chapter.

4.8 Self-Powered DSP System: Design and Test

To demonstrate the feasibility of a self-powered system, a generator, rectifier, and an integrated circuit containing a switching DC/DC converter and an FIR filter were constructed and later assembled into a self-powered system. The full self-powered system was tested using both the moving-coil generator discussed above and an acoustic generator to test steady-state operation of the system. This chapter presents the system-level design and test results.

4.8.1 Control Loop Performance

The output response to a step change in $V_{in}$ is shown in Figure 4.27. After some initial ringing (~40 ms), the output settles except for small low frequency ripples at the far right
Figure 4.26: Die Photo.
Figure 4.27: Step Response. Note steady-state limit cycle due to low resolution digital control. This can be ameliorated with better deadzone width and error sample rate.

of the trace. A limit cycle in the controller due to the low resolution error feedback causes the PWM to oscillate around the correct value. Tuning of the controller parameters can do more to eliminate this limit cycle.

Figure 4.28 shows the bootstrapping circuit in operation. When BOOT goes high, $V_{dd}$ rises to the higher regulated value, and the frequency of the PWM output increases, as expected. The $V_{dd}$ glitch is due to timing mismatch in the switch gate signals.

4.8.2 Acoustic Power Generation

Ambient sound waves can also be transduced to electrical energy and used to power a system without batteries. For test purposes, a generator consisting of a miniature moving coil loudspeaker used in reverse as a microphone was stimulated by sound waves. An identical loudspeaker driven by a signal generator provided the input stimulus. The speaker was placed directly opposite the generator with a separation distance of 1 mm between their diaphragms, as shown in Figure 4.29. Speaker S0 and microphone M0 are both Panasonic EAS2P20 miniature speakers [59]. A function generator $V_{sig}$ drove S0 with a sinusoidal input with amplitude 6 V and frequency 1 kHz. The low input impedance (8 Ω) of the speaker required a 10:1 step down transformer X0 to be placed between the signal generator and the speaker. This stimulus generated a steady-state $V_{in}$ of approximately 1 V peak-to-peak, which was then rectified and regulated by the DC/DC converter of Chapter 4. The power load of the converter and DSP is represented by resistor R0 in the figure. The full system including the DSP was observed to work for $f_{clk}$ from 100 kHz up to 1 MHz with $V_{out}$ varying from about 0.85 V up to 0.97 V.

To determine if it is possible to use such an acoustically driven generator in practice, it
Figure 4.28: Bootstrapping.

Figure 4.29: Acoustic power generation. Speaker S0 and microphone M0 are actually identical components. Transformer X0 is necessary to allow the signal generator to drive S0.
is necessary to compute the input power levels and the intensity of the acoustic radiation. The input stimulus had an amplitude of 6 V while the input impedance of the transformer is 1 kΩ at 1 kHz. The power driven into the speaker $P_{in}$ is thus 36 mW. The following formula relates sound level $\beta$ measured in decibels (dB) to sound intensity $I$:

$$\beta = 10 \log \frac{I}{I_0}$$

(4.21)

where $I$ has units of W/m² and $I_0$ is a reference intensity corresponding to the lower limit of human hearing [60]. The value of $I_0$ is $10^{-12}$ W/m². The datasheet indicates a sound level of 80 dB for a 1 W input power at a distance of 0.5 meters. Assuming the sound intensity $I$ is related to the input power by an efficiency factor $\eta$, we have

$$I = \frac{\eta P_{in}}{4\pi r^2}$$

where $r$ is the distance away from the acoustic source at which the intensity is measured in meters. The factor $\eta$ can be solved for using the information from the datasheet. Plugging the appropriate parameters from the experiment, the sound level at the microphone M0 was 114 dB. This is approximately the sound level at a rock concert, so this particular generator configuration requires very strong acoustic fields to produce sufficient power for the circuit. However, more careful design of the microphone M0 might improve its efficiency enough to make this system practical.

### 4.8.3 Test Result Summary

The test chip parameters are summarized in Table 4.2. For $f_{clk}$ at 500 kHz, the controller consumed 5.71 µW and the FIR filter and load ring oscillator 4.75 µW at 1 Volt power supplies. This does not include switching the PWM output FETs or the chip pads. The switch driver power is 7.50 µW. Figure 4.30 shows a voltage regulation waveform resulting from an impulse stimulus to a speaker used as a vibration generator. The impulse results in several packets of charge being dumped in to the half-wave rectifier capacitor whose voltage is $V_{in}$. $V_{out}$ is the DC/DC converter output voltage which is regulated to a level adequate to satisfy the required system throughput. One impulse generated sufficient energy to produce 23 ms of valid DSP operation.

From an efficiency standpoint, this system does not deliver great performance. This is partly because the subband filter implementation is extremely low power. In addition, this is a characteristic of these self-powered systems. The FFT example of Section 1.3 has even lower average power than the DSP implemented as a test load. The penalty for harvesting energy is often spending more energy to scavenge it than is required by the load. On the other hand, the advantages of possibly infinite lifetime for a small system volume and cost can outweigh this inefficiency.
<table>
<thead>
<tr>
<th>Area</th>
<th>2609 μm x 2609 μm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transistor Count</td>
<td>5 K</td>
</tr>
<tr>
<td>Process</td>
<td>0.8 μm CMOS</td>
</tr>
<tr>
<td>NMOS Threshold Voltage</td>
<td>$V_{iN} = 0.70$ V</td>
</tr>
<tr>
<td>PMOS Threshold Voltage</td>
<td>$V_{iP} = -0.87$ V</td>
</tr>
<tr>
<td>Controller Power</td>
<td>5.71 μW ($f_{ref} = 500$kHz, $V_{dd} = 1$V)</td>
</tr>
<tr>
<td>Subband Filter Power</td>
<td>4.75 μW ($f_{ref} = 500$kHz, $V_{out} = 1$V)</td>
</tr>
<tr>
<td>Switch Drive Power</td>
<td>7.50 μW ($V_{in} = 1.07$V)</td>
</tr>
<tr>
<td>1 Generator Excitation</td>
<td>23 ms of valid DSP operation</td>
</tr>
<tr>
<td></td>
<td>11,700 cycles</td>
</tr>
<tr>
<td></td>
<td>2,340 operations</td>
</tr>
</tbody>
</table>

Table 4.2
Test Chip Specifications.

Figure 4.30: Voltage Regulation.
Chapter 5

Energy Scalable Classifier

5.1 Introduction

An acoustic sensor worn at the neck detects three pieces of information that may be used in classifying the wearer’s state of physical exertion: heartbeats, breaths, and voice stress levels. The simplest of these to detect is the heartbeat, through the pulse sound that occurs in the neck. Voice sounds and extraneous noise make this somewhat more difficult, and below we summarize an algorithm that performs good heartbeat detection in the presence of these noise sources. We also discuss preliminary investigations of methods of detecting breaths and estimating breathing rate. Other acoustic information or data from different sensors may also be relevant for this classification, but that is future work in this area.

5.2 Heartbeat Detection

Visual evaluation of the spectrogram of the acoustic data indicates that most of the energy from heartbeat sounds lies in the low frequency range, below 200 Hz. We developed a classifier based approach to heartbeat detection that takes advantage of this spectral characteristic to improve detection performance in the presence of speech and other high frequency energy.

The basic algorithm is outlined below:

1. Preprocessing:
   - Segmentation: The sensor output is divided into overlapping segments at least long enough to contain a full heartbeat in the time domain, but short enough not to contain more than one.
   - Lowpass Filtering: The segmented data is bandlimited to below 200 Hz to eliminate as much of the voice and breath energy as possible.
- Matched Filtering: The output of the lowpass filter is passed through a matched filter to determine the candidate heartbeat locations in the time domain.

2. Feature Extraction: A set of seven features are computed from the matched filter output.

3. Classification: Each feature vector is classified into a heartbeat or nonheartbeat using a parametric Gaussian multivariate classifier.

5.2.1 Preprocessing

The heartbeat waveform averages about 0.5 seconds in duration and much of the energy is concentrated below 100 Hz. The first preprocessing step is segmenting the sensor output into 0.5 s segments, each overlapping by 0.25 s. This is just long enough to contain a single heartbeat. The overlap is set so that it is unlikely to miss a heartbeat at the resting heart rate. The overlap should be increased if the heartrate is higher, and can be adjusted dynamically to reduce power consumption.

The segmented signal is lowpass filtered at the 200 Hz cutoff to remove as much high frequency energy as possible. This step greatly increases the effectiveness of the heartbeat detection classifier. Since most of the signal energy is actually in the band below 100 Hz, it is possible for the lowpass filter to have a fairly wide transition band - as long as there is enough stopband attenuation above 200 Hz.

Matched filtering is the final preprocessing step. The matched filter impulse response, or filter template, is a cleaned up version of the acoustic signature of the heartbeat. When
Figure 5.2: Matched filter output without speech.

convolved with the input data, the filter output has a large correlation peak at the time location of a heartbeat in the input. The template was constructed through iterations. First, a small set of representative waveforms, uncorrupted by voice, was selected and averaged to generate an initial template. This initial template was then used to locate the positions of all the heartbeat waveforms. Finally, a refined template was obtained by averaging all the available waveforms in the nonspeech data. This template is shown in Figure 5.1.

Figure 5.2 shows the input data free of speech and the matched filter output below it. There are clearly defined correlation peaks corresponding to the locations of acoustic heartbeat signatures.

On the other hand, Figure 5.3 shows the input data with speech and the matched filter output shows that it is difficult to find clear correlation peaks while there is significant voice band energy. Therefore, we require more sophisticated processing than a simple thresholding of the matched filter output to detect heartbeats. Feature extraction and classification of the matched filter output enables accurate heartbeat detection in the presence
of voice signals.

5.2.2 Feature Extraction

The hundreds of samples produced in the segmented output of the matched filter are reduced to a small set of features for classification into heartbeats or nonheartbeats. This information, accumulated over time, can give an estimate of the wearer's heart rate, which in turn can be integrated over time to give an estimate of fatigue. Figure 5.4 shows a typical correlation peak in the matched filter output with 6 of 7 candidate features labeled. The values of three peaks and two valleys are used as the first five candidate features. The algorithm selects the largest peak and finds the other four peak values by taking the value of the correlation output at four fixed time offsets from the main peak. This preserves the time domain characteristics of the correlation in that the features become shift invariant and sensitive to scale. Two other candidate features are the matched filter output energy
5.2. HEARTBEAT DETECTION

![Correlation Output of Matched Filter with Lowpass Throat Data]

**Figure 5.4:** Typical correlation peak with labeled features.

...and the width of the central correlation peak, labeled 7 in the figure.

The extracted features are summarized below:

1. peak correlation output value  
2. first valley after peak correlation  
3. first valley before peak correlation  
4. first peak after peak correlation  
5. first peak before peak correlation  
6. matched filter output energy  
7. peak correlation output width

5.2.3 Detection Using Multivariate Gaussian Classification

To evaluate the effectiveness of the candidate features in detecting heartbeats, we used the Sanders Classification Toolbox [61]. The time series of the lowpass throat acoustic data both with and without speech as divided into several overlapping time segments. Each segment contained hearbeats, speech, or background noise (including breath sounds), or parts of these different signals. The seven features described above were extracted for
Figure 5.5: Probability density functions for the normalized candidate heartbeat features.

these segments. The segments were labeled by hand as containing a heartbeat or not. Using the classifier toolbox, a classifier was designed, trained, and evaluated for recognition performance on these time segments.

Figure 5.5 shows the PDFs for the candidate features, whose magnitudes have been normalized to the extremes. The solid line corresponds to a heartbeat present (class 1) and the dashed to no heartbeat present (class 2). The functions are vaguely unimodal, although the presence of more than one peak may be the result of the rather small sample size (120 labeled segments). Feature 1 (peak correlation output value) appears to have the greatest spread between the two classes, so it should be the most distinguishing feature.

Feature ranking determines which features are most useful for doing a classification. The can be used as a basis for deciding which features are worth keeping and which should be discarded. It can also determine how to trade off features for recognition performance.
Using two quick (i.e. deterministic) methods of feature ranking, the Fisher's Discriminant Ratio 1 (FDR1) and the Multimodal Overlap Measure (MOM), the features were evaluated using the multivariate Gaussian parametric classifier [62]. This classifier simply chooses the class for a particular feature based on minimizing the Mahalanobis distance,

\[ d_n = \sqrt{(f - \mu_n)^T R_n^{-1} (f - \mu_n)} \]  

(5.1)

where \( f \) is the feature vector for the input pattern, \( R_n \) is the feature covariance matrix for the \( n \)th class, and \( \mu_n \) is the mean vector for the \( n \)th class. This function is actually quite computationally cheap since the feature vectors are small.

The linear Fisher's Discriminant Ratio 1 is computed from the basic statistics of the probability density functions for each class for a particular feature:

\[ FDR1(p) = \sum_{i=1}^{K} \sum_{i \neq j}^{K} \frac{(\mu_{ij} - \mu_{ij})^2}{(\sigma_{ij})^2 + (\sigma_{ij})^2} \]  

(5.2)

where \( K \) is the number of classes, \( p \) is the feature index, \( \mu_{ij} \) is the mean of the \( p \)th feature for the \( i \)th class, and \( \sigma_{ij} \) is the standard deviation of the \( p \)th feature for the \( i \)th class. FDR1 compares the distance between the means of the candidate features for different classes and normalizes it to the feature variances. Feature pairs that have a large FDR1 value are good for discriminating between classes. FDR1 is good for comparing unimodally distributed features for two classes, as in the heartbeat detection example.

The Multimodal Overlap Measure is used to compare features with multimodal PDFs. It simply computes the overlap between the PDFs of a feature for two distinct classes. If the overlap is small, the feature is good for distinguishing the two classes:

\[ MOM = \int_{x_1}^{x_2} \min(p_1(x), p_2(x)) \, dx \]  

(5.3)

where \( x_1 \) and \( x_2 \) are the limits of the region of support for the PDFs \( p_1(x) \) and \( p_2(x) \).

The deterministic feature rankings based on Equations 5.2 and 5.3 were:

FDR1: 1, 6, 2, 3, 4, 5, 7

MOM: 1, 3, 2, 4, 6, 5, 7

Figure 5.6 shows the classifier recognition performance for the FDR1 feature ranking. Incorporating features 1, 6, and 2 improved recognition performance while the other features decreased it, especially recognition of the nonheartbeat class, class 2. The training and testing tokens were selected randomly over 100 Monte Carlo tests of the classifier. Since the worst case performance for three features is above 90%, this appears to be a reasonably good system for heartbeat detection.

Figure 5.7 shows the recognition performance for the MOM feature ranking. It is substantially poorer than the FDR1 ranking, indicating that the unimodal assumption on the feature PDFs results in better recognition performance.
Figure 5.6: Recognition performance for the FDR1 feature ranking.

After numerous trial and error experiments, a slightly better performing feature ranking was determined:

TER: 1, 6, 2, 7, 5, 4, 3

Figure 5.8 shows the classifier performance for this ranking. The peak recognition has increased to 94% at the expense of examining an extra feature. Moreover, the recognition performance of each class is very similar at the peak overall recognition (4 features). With a worst case Monte Carlo performance of 92.7% four 4 features, this ranking appears to be the best for reliable heartbeat detection.

All classification schemes were also tested against simulated broadband noise and $1/f$ noise to ensure adequate noise rejection performance. This test ensures that the classifier can accurately detect that there is no signal appearing in the presence of noise. The segments were correctly classified as noise with from 98% to 100% recognition performance,
Figure 5.7: Recognition performance for the MOM feature ranking.

depending on the noise power level.

5.2.4 Influence of Sample Rate and Quantization on Classifier Performance

To implement a detection and classification algorithm in hardware, it is important to determine the impact of input data sample rate and quantization affect the performance of the algorithm. In addition to input quantization, if an integer digital implementation is used for the algorithm, truncation effects also enter the picture. At high levels of quantization, it is possible to treat the nonlinearity mathematically as additive white noise at the input [19]. For this application where the resolution is fairly low, simulation can be used to determine adequate sample rates and quantization levels.

Most of the energy of an acoustic heartbeat signal is in a band below 100 Hz. As the
Figure 5.8: Recognition performance for the TER feature ranking.

sample rate is lowered, antialiasing rejects more and more of the signals of interest, effectively degrading the signal-to-noise ratio at the input to the classifier. Figure 5.9 shows how changes in sample rate affect heartbeat recognition performance for both fixed and floating point arithmetic implementations of the algorithm. As the sample rate is decreased, the recognition percentage also decreases as more signals are classified incorrectly. However, lower sample rates also result in lower power consumption as the signal processing circuitry can be clocked slower for a real-time constraint. This is one example of a power and performance tradeoff which can be exploited in an energy scalable classifier design. Intuitively, one would expect the floating point performance to be better on average than the fixed point performance. Changing the sample rate affects how much signal and noise energy reaches the classifier and can change how signals on the borderline between two classes are resolved. That is the reason why the floating point classifier has worse performance than the fixed point implementation at certain sample rates. A more robust classifier
Figure 5.9: Classifier recognition performance for different sample rates, both floating point and fixed point implementations.

architecture would have a third class for unresolved feature vectors to which difficult input segments (segments on the class boundaries in feature space) would be assigned. Since heartbeat detection does not need to be as robust as, for example, a radar target detection scheme, it is desirable to eliminate the extra computational overhead of introducing this third class. Therefore the classifier is more sensitive to changes in input sample rate and quantization as discussed below.

Figure 5.10 shows how input quantization levels affect recognition performance for both a floating point classifier and a fixed point classifier at a fixed input sample rate. Again, both curves show a downward trend as the input quantization becomes coarser and coarser. The signal-to-noise ratio at the input has been degraded by the quantizer. The quantization level can also be used to trade power for performance as will be discussed in Section 5.3. However, quantization is a nonlinear process and the error introduced by it may be adequate to push the decision boundaries between the classes enough to change the recognition performance for data that is on a class boundary. These changes are unpredictable, and since the classifier is required to resolve each input into one of only two classes, there is nonmonotonicity in the curves of Figure 5.10.

Finally, the choice of features used in the classification algorithm also affects recognition accuracy. As more features are used (the feature space grows in dimension), it is intuitive that recognition percentage should increase since more data is being used to distinguish positive and negative events. Using the feature rank ordering discussed previously, a choice can be made as to which features should be used when only a subset
Figure 5.10: Classifier recognition performance for different quantization levels, 160 Hz sample rate.

of the possible ones are available. Figure 5.11 shows the tradeoff: starting with the one most discriminating feature for this example, as other features are added in the rank order, the recognition performance increases. This is yet a third approach to power-performance scalability since it costs power to compute and discriminate between additional features. The significance of this tradeoff is implementation dependent, as some features may have been precomputed as part of the detection and segmentation of the input.

This section has described three alternatives to trading energy and performance: input data sample rate, input data quantization, and feature selection. The impact of these tradeoffs is implementation dependent, depending on the cost of running at a higher clock frequency, computing with wider bit widths, or computing more features. For the particular example of the heartbeat classification, the input bitwidth is the primary means of achieving energy scalable classification.

5.3 Power Scalable Classification Using Distributed Arithmetic

A recent trend in low power design has been the employment of reduced precision “approximate processing” methods for reducing arithmetic activity and chip average power dissipation. Such designs treat power and arithmetic precision as system parameters that can be traded-off vs. each other on and ad-hoc basis. Ludwig et. al [63] have demonstrated an approximate filtering technique which dynamically reduces the filter order based on the input data characteristics. More specifically, the number of taps of a frequency-selective
Figure 5.11: Classifier recognition performance for different feature sets.

FIR filter is dynamically varied based on the estimated stopband energy of the input signal. The resulting stopband energy of the output signal is always kept under a predefined threshold. This technique results in power savings of a factor of 6 for speech inputs. Larsson and Nicol [64] [65] have demonstrated an adaptive scheme for dynamically reducing the input amplitude of a Booth-encoded multiplier to the lowest acceptable precision level in an adaptive digital equalizer. Their scheme simply involves an arithmetic shift (multiplication/division by a power of 2) of the multiplier input depending on the value of the error at the equalizer output. They report power savings of 20%.

This work explores the potential of Distributed Arithmetic (DA) [66] [67] computation structures for low power precision-on-demand computation. Distributed Arithmetic is a method of computing vector inner products without the use of a multiplier. It has a number of applications in fixed-function DSP VLSI implementations [18] [68] [69]. When used appropriately it features stochastically monotonic successive approximation properties. In this work, we present the theory behind Distributed Arithmetic and its approximate processing properties. We present a proof-of-concept VLSI implementation in Chapter 6, a heartbeat classifier chip whose power dissipation characteristics change on-the-fly according to the precision of the computation performed.

5.3.1 Distributed Arithmetic

Distributed Arithmetic (DA) [66] [67] is a bit-serial operation that computes the inner product of two vectors (one of which is a constant) in parallel. Its main advantage is the effi-
ciency of mechanization and the fact that no multiply operations are necessary. DA has an inherent bit-serial nature, but this disadvantage can be completely hidden if the number of bits in each variable vector coefficient is equal or similar to the number of elements in each vector.

As an example of DA mechanization let us consider the computation of the following inner (dot) product of \( M \)-dimensional vectors \( \mathbf{a} \) and \( \mathbf{x} \), where \( \mathbf{a} \) is a constant vector:

\[
\mathbf{y} = \sum_{k=0}^{M-1} a_k x_k
\]  

(5.4)

Let us further assume that each vector element \( x_k \) is an \( N \)-bit two’s complement binary number and can be represented as

\[
x_k = -b_{k(N-1)} 2^{N-1} + \sum_{n=0}^{N-2} b_{kn} 2^n
\]  

(5.5)

where \( b_{ki} \in \{0, 1\} \) is the \( i \)th bit of vector element \( x_k \). Please note that \( b_{k0} \) is the least significant bit (LSB) of \( x_k \) and \( b_{k(N-1)} \) is the sign bit.

Substituting eq. 5.5 in eq. 5.4 yields:

\[
y = \sum_{k=0}^{M-1} a_k [-b_{k(N-1)} 2^{N-1} + \sum_{n=0}^{N-2} b_{kn} 2^n]
\]  

(5.6)

\[
y = -\sum_{k=0}^{M-1} a_k b_{k(N-1)} 2^{N-1} + \sum_{k=0}^{M-1} a_k \sum_{n=0}^{N-2} b_{kn} 2^n
\]  

(5.7)

\[
y = -\sum_{k=0}^{M-1} a_k b_{k(N-1)} 2^{N-1} + \sum_{n=0}^{N-2} \left[ \sum_{k=0}^{M-1} a_k b_{kn} \right] 2^n
\]  

(5.8)

Let us consider the term in brackets:

\[
q_n = \sum_{k=0}^{M-1} a_k b_{kn}
\]  

(5.9)

Because \( b_{kn} \in \{0, 1\} \), \( q_n \) has only \( 2^M \) possible values. Such values can be precomputed and stored in a ROM of size \( 2^M \). The bit serial input data \( \{b_{0i}, b_{1i}, b_{2i}, \ldots, b_{ki}\} \) for \( i = 0, 1, \ldots, N-1 \) is used to form the ROM address, and the ROM contents can be placed in an accumulator structure to form the outer sum of eq. 5.8. Successive scalings with powers of 2 can be achieved with an arithmetic shifter in the accumulator feedback path. The first term of eq. 5.8 \( \left( \sum_{k=0}^{M-1} a_k b_{k(N-1)} \right) \) is also stored in the ROM at address \( \{b_{0(N-1)}, b_{1(N-1)}, b_{2(N-1)}, \ldots, b_{k(N-1)}\} \). Some extra control circuitry is necessary to ensure that the accumulator subtracts (as opposed to adding) the partial sum to the total result at sign bit time. After \( N \) cycles (as a reminder \( N \) is the bitwidth of the \( x_k \) vector elements) the final result \( \mathbf{y} \) has converged to its final value within the accumulator.
Figure 5.12: Distributed Arithmetic ROM and Accumulator (RAC) Structure

Figure 5.12 shows a detailed example of a Distributed Arithmetic computation. The structure shown computes the dot product of a 4-element vector $X$ and a constant vector $A$. All 16 possible linear combinations of the constant vector elements ($A_i$) are stored in a ROM. The variable vector $X$ is repackaged to form the ROM address most significant bit first. We have assumed that the $X_i$ elements are 4-bits 2's complement (bit 3 is the sign bit.) Every clock cycle the RESULT register adds $2 \times$ its previous value (reset to zero) to the current ROM contents. Moreover, each cycle the 4 registers that hold the four elements of the $X$ vector are shifted to the right. The sign timing pulse $T_s$ is activated when the ROM is addressed by bit 3 of the vector elements (sign). In this case the adder subtracts the current ROM contents from the accumulator state. After four cycles (bitwidth of the $X_i$ elements) the dot product has been produced within the RESULT register.

5.3.2 Successive Approximation Using Distributed Arithmetic

In this section, we show that when the Distributed Arithmetic operation is performed MSB first, it exhibits stochastically monotonic successive approximation properties. In other words, each successive intermediate value is closer to the final value in a stochastic sense. An analytical derivation follows:

The $i$th intermediate result of an MSB-first DA computation ($i > 0$) is:

$$y_i = -q_{(N-1)} + \sum_{n=N-1-i}^{N-2} q_n 2^n$$

(5.10)
where
\[ q_n = \sum_{k=0}^{M-1} a_k b_{kn} \]  \hspace{1cm} (5.11)

Please note that when \( i = N - 1 \), eq. 5.10 yields eq. 5.8.

Let us define an error term \( e_i, \ i = 0, 1, \ldots, N - 1 \) as the difference between each intermediate value \( y_i \) and the final value \( y \):
\[
\begin{align*}
e_i &= y - y_i \hspace{1cm} (5.12) \\
\nonumber \\
e_i &= \sum_{n=0}^{N-2-i} q_n 2^n \hspace{1cm} (5.13)
\end{align*}
\]

We model \( q_n \) as experimental values of a discrete random variable \( q \). The underlying stochastic experiment is random accesses of the DA coefficient ROM in the presence of random inputs. The experimental values of \( q \) are the DA ROM contents. The first and second order statistics of the error term \( e_i \) are:
\[
\begin{align*}
E[e_i] &= E[q] \sum_{n=0}^{N-2-i} 2^n \\
&= E[b_{kn}] \sum_{k=0}^{M-1} a_k \sum_{n=0}^{N-2-i} 2^n \hspace{1cm} (5.15) \\
&= \frac{2^{N-1-i} - 1}{2} \sum_{k=0}^{M-1} a_k \hspace{1cm} (5.16) \\
\sigma_{e_i}^2 &= \sigma_q^2 (1 + 4 + \cdots + 2^{2(N-2-i)}) \hspace{1cm} (5.17) \\
&= \frac{2^{2(N-1-i)} - 1}{3} \sigma_q^2 \hspace{1cm} (5.18) \\
&= \frac{2^{2(N-1-i)} - 1}{3} \text{Var} \left[ \sum_{k=0}^{M-1} a_k b_{kn} \right] \hspace{1cm} (5.19) \\
&= \frac{2^{2(N-1-i)} - 1}{3} \sum_{k=0}^{M-1} a_k^2 \sigma_{b_{kn}}^2 \hspace{1cm} (5.20) \\
&= \frac{2^{2(N-1-i)} - 1}{12} \sum_{k=0}^{M-1} a_k^2 \hspace{1cm} (5.21)
\end{align*}
\]

where equations 5.16 and 5.21 have been computed under the assumption that the least significant bits \( b_{kn} (i \text{ large}) \) are independent identically distributed random variables uniformly distributed between 0 and 1 \( (E[b_{kn}] = 1/2, \ \sigma_{b_{kn}}^2 = 1/4) \). This is a valid assumption for input DSP data \[70\][71]. The fact that equations 5.16 and 5.21 are monotonically decreasing functions of \( i \) (RAC cycles) shows the successive approximation property (in probabilistic terms) of the Distributed Arithmetic mechanization.
Figure 5.13: Power reduction versus input quantization (simulated using PowerMill).

5.3.3 Implementation of Heartbeat Matched Filter

Using Distributed Arithmetic, a complete dot product can be performed in as many cycles as correspond to the bit widths of the input samples. If the bitwidth \( M \) is less than the filter length \( N \), this implementation requires fewer clock cycles than a multiply accumulate. This is beneficial for long filters like the matched filter described above, where \( M \gg N \). The reduced clock results in low total power not just through frequency reduction, but also through increased voltage reduction since the delay constraint of the DA filter critical path is much less stringent than the multiply-accumulate architecture. Figure 5.13 shows the power reduction in the Distributed Arithmetic unit as the input quantization level is decreased (i.e. fewer bits of the input are shifted into the filter). The lower end of the curve saturates since there is a constant power cost associated with the control logic and clocking of the DA unit. The choice of powers of 2 for the available bitwidths was an architectural decision based on the convenience of clock gating and multiplexing in the VLSI implementation and will be discussed in Chapter 6.

As discussed above, the bit-serial nature of the implementation also allows an alternative approach to approximate processing. By clocking the DA units at less than the full bitwidth, we are in effect reducing the input quantization level. This is roughly equivalent to injecting noise at the input of the filter. In a detection scheme like the heartbeat detection algorithm, this reduced signal to noise ratio should result in lower performance, i.e. less reliable detection of heartbeat events. However, the reduced performance has also resulted in reduced power since the switched capacitance per output filter sample decreases linearly with the number of input bits clocked in. Figure 5.14 shows the classifier
Figure 5.14: Recognition performance tradeoffs with DA Unit power.

performance reduction as the DA unit power is decreased.

5.4 Breath Detection

Breath and speech sound energy is concentrated in the high frequency portion of the spectrum (> 200 Hz). Figure 5.15 shows the moving average energy versus time of nonspeech acoustic data gathered from the neck worn sensor. The energy is computed for narrowband filtered versions of the original signal. The peaks indicate fairly well when breaths are occurring and the duration of the breaths. The impulsive "popping" noise indicated by sharp, narrow spikes in the time series contributes the most extra energy in these bands that might lead to a misclassification. A classifier based approach is again used for breath detection.

5.4.1 Feature Extraction

The time series is divided up into short duration nonoverlapping segments. Each segment is labeled according to whether breathing (class 1), "popping" noise or speech (class 3), or background noise is occurring during the segment. The extracted features are basically the signal energy in 3 different high frequency bands, normalized by the energy in the highest band to eliminate misclassification of broadband noise. The normalization factor and the total high frequency energy form the last two candidate features.
Figure 5.15: Moving average energy in various high frequency bands indicating the presence of breath sounds.

The five features are listed below:

1. normalized energy, 200-600 Hz
2. normalized energy, 600 Hz - 1 kHz
3. normalized energy, 1 kHz - 1.4 kHz
4. normalization factor, energy in 1.4-1.8 kHz band
5. total energy, 200 Hz - 1.8 kHz

The upper bound on frequency set by the 4 kHz sampling rate is 2 kHz, but there is not much energy in the signal at the edge of the antialiasing passband.
5.4.2 Classification

Examination of the feature PDFs indicate that they are basically unimodal and Gaussian in distribution. Therefore we again use the multivariate Gaussian parametric classifier as the breath detection engine. For test purposes, the speech and "pop" noise tokens were eliminated since a more sophisticated algorithm will probably be needed to handle these high energy signals.

A modified version of FDR1 is Fisher's Discriminant Ratio 2, which is useful for evaluating features to distinguish more than two classes. In this case, the classifier may separate the acoustic signals into breaths, nonbreaths, and a don't know class. FDR 2 is defined as follows:

$$FDR2(p) = \frac{\sum_{i=1}^{K} \sum_{j=1}^{K} (\mu_i^j - \mu_p^j)^2}{\sum_{i=1}^{K} (\sigma_p^i)^2}$$

(5.22)

where the symbols have the same meaning as in Equation 5.2. The three different feature ranking methods were tried using the classification toolbox, Fisher's Discriminant Ratio 1, 2, and MOM:

FDR1, FDR2: 3, 2, 5, 1, 4 MOM: 5, 4, 1, 3, 2

The MOM ranking is more intuitive since it primarily looks at the total energy in that band, which is much higher when breath sounds are present than not.

Figure 5.16 shows the classifier recognition performance for these candidate features. In general it is poor, particularly because in the transition times when a breath starts and stops, it is difficult to accurately classify the segment based on the energy content. However, if one can classify consecutive breath samples consistently, a binary sequence can be constructed that determines when a breath is occurring. Each breath would consist of several 1s in a row, pauses in breathing by a string of 0s. Using the 0 to 1 and 1 to 0 transitions enables the duration of the breath and the pauses to be estimated. Counting the number of pulses of 1s is an estimate of breathing rate. This is a first step toward estimating the wearer's exertion state using the breath signals.

5.5 Voice Stress Analysis

Voice stress might also be a good indicator of physical exertion state. Further research on low power speech algorithms may show that voice stress processing is feasible for low to medium throughput low power DSPs. However, its analysis requires complex algorithms that we have yet to explore and may require unreasonably high performance from the signal processing engine for the limited available power.
Figure 5.16: Recognition performance for breath sounds using the FDR1 feature ranking.

5.6 Future Work: Extrapolation to Exertion State

The detection algorithms described above can be used as a jumping off point for determining the physical exertion state of the user by computing heart rate, breath rate, etc. Truth data for this classification problem was obtained from ARL and we are currently awaiting a data transfer to improve and expand the performance of the preceding algorithms.
Chapter 6

Sensor DSP Chip

The previous chapter described in detail a particular sensor signal processing application: heartbeat detection and classification for an acoustic biomedical sensor. These algorithms can be used in a number of sensor systems where measured data must be converted into the detection of events of interest to the user. It was also shown how Distributed Arithmetic can be used to achieve power scalability for linear filtering operations. However, an algorithm cannot be low power in itself; it must be computed using an architecture which has low power consumption. This chapter describes the architecture, circuit design, and VLSI implementation of a programmable sensor signal processing chip. Its architecture is tailored to detection and classification algorithms and is applicable to a range of sensor DSP problems.

6.1 Low Power Classifier Chip Architecture

For any particular detection and classification application, we find that there are a number of steps subsumed in the feature extraction and classification blocks. In particular, it is often necessary to perform a substantial amount of preprocessing on measurement vector \( x \) [62]. Figure 6.1 is a detailed block diagram of the typical signal processing that must occur for classification, including the acoustic heartbeat detection that this system performs. The measurement signal \( x[k] \) is a scalar time series. It is passed through a linear filter, for example a matched filter, to increase the signal to noise ratio, bandlimit it for the digital postprocessing, etc. The filter output \( y[k] \) is then divided by a segmentation unit into regions, or subsets of the filter output for all time. These regions contain discrete events that must be analyzed by the classifier. One of these regions at a time, \( s \), is passed to the feature extraction block where the transformation \( A(s) \) is applied to create the feature vector \( f \). The feature vector should contain all the important information contained in the large set of samples \( y[k] \in s \) in each segment. The classifier then assigns the event in \( s \) to an output class \( z \). The purpose of the preprocessing and segmentation is to facilitate the feature extraction and provide manageable chunks to the classifier at the end.
Figure 6.1: Classification signal processing block diagram.

Figure 6.2: Signal processing chip architecture.

Figure 6.2 shows the architecture of the proposed sensor DSP chip. It follows the algorithmic architecture shown in Figure 6.1. The discrete-time linear filter $H(z)$ is implemented using the Distributed Arithmetic Unit. Its output is then passed to a nonlinear filtering unit to calculate quantities used in segmentation. The final segmentation, feature extraction, and classification is performed by the programmable microcontroller at the end to produce the class assignment $z$. The buffer provides a mechanism for synchronization between the front end filtering and the backend processing. This is necessary for power reduction. The filtering front end must be running continuously to process the input samples, which arrive at a fixed rate. However, the back end classification only needs to be performed for every segment $s$, not every input sample $x[k]$. The system operates as follows: first, the front end filters the input and writes important results to the buffer. A small loop is continuously executed in the microcontroller, checking to see if a full segment has been written to the buffer. The filtering units could do this, but it involves adding circuits that already exist in the ALU of the microcontroller, which is idle anyway while it is trying to detect a segment. To conserve area, we use it rather than add complexity to the filter functional units.

When a segment is detected, the microcontroller executes the feature extraction and classification code on the data in the buffer that was just written. In general, many more instructions are executed in the backend processing than the filtering. On the other hand, new segments are produced fairly infrequently, so these large programs are not run very often. Synchronization is handled by muxing between a slow clock, $\Phi_0$ and a fast clock $\Phi_1$ on chip. When the microcontroller enters this classification mode, it disables the front end filters and requests a higher clock rate ($\Phi_1$). The long program is run at this higher rate
so that it can complete before the next input sample appears at the filter inputs. When the long program finishes, the microcontroller switches back to the slow clock Φ0 and enables the filtering units. This clock muxing approach ensures that the system operates in high performance/high power mode when necessary instead of at all times.

The important feature of the algorithm that this architecture addresses is the fact that the workload is dominated by the preprocessing. Events occur fairly rarely in time compared to the number of filter outputs that must be computed and compared against thresholds for segmentation. Therefore the architecture contains several front end functional units that are optimized for the preprocessing operations. These will be discussed in detail in the following sections. A quantitative workload analysis is presented in Section 6.1.4.

6.1.1 Linear Filter

Figure 6.3 shows the architecture of the proposed linear filter implementation based on the well-known technique of Distributed Arithmetic [67]. This technique is primarily a bit-serial method for forming the dot product of two vectors in a single step, rather than performing many multiply-accumulates as is typical in DSP implementations. The various bits of all the input samples are fed serially into a group of tables, where they address words that correspond to partial sums of the dot product. These sums are accumulated into a register along with a shifted version of the previous accumulator value. The shift accounts for the different weightings of the bits addressing the DA tables. Thus, a complete dot product can be performed in as many cycles as correspond to the bit widths of the input samples. If the bitwidth \( M \) is less than the filter length \( N \), this implementation requires fewer clock cycles than a multiply accumulate. The reduced clock results in low power not just through frequency reduction, but also through increased voltage reduction since the delay constraint of the DA filter critical path is much less stringent than the multiply-accumulate architecture.

The bit-serial nature of the implementation also allows an alternative approach to approximate processing. Previous work has explored varying the number of filter taps when there are less stringent requirements on the frequency domain performance, resulting in lower power [63]. Multiply-accumulate architectures have also been used that truncate their filter coefficients adaptively [65, 64] to reduce power. Distributed arithmetic can take another approach by using fewer bits of the input samples in the DA unit addresses to produce an approximate filtering result. The reduced clock requirement and switched capacitance versus the multiply-accumulate structure again results in lower power.

Figure 6.4 shows the implementation of the DA Unit shift register. The input samples are shifted in serially through the input \( X \). Since the maximum bitwidth of an input sample is 8 bits, the shift register stores up to four 8 bit samples to address the DA table of precomputed results. To provide the energy scalability, selection bits control the total length of the register by using the bypass multiplexers to reconfigure the serial path. For example, to handle two bit wide input data the first string of four flip-flops and the second string of two flip-flops are bypassed. The register supports input bitwidths of 8, 4, 2, and
1 bit. Further power savings come from gating the clocks of the bypassed shift register segments.

6.1.2 Nonlinear/Short Linear Filter

For filters with a length shorter than the bitwidth \((N < M)\), a multiply-accumulate approach requires fewer clock cycles than Distributed Arithmetic. A multiply-accumulate architecture can result in decreased power consumption compared to a DA implementation as long as the switched capacitance of the multiplier is less than the overhead associated with the DA tables. The Nonlinear/Short Linear (NLSL) filter unit is used to implement these types of filters. Figure 6.5 shows the architecture of the final filter and time series segmenting block. This block must also operate continuously. To maintain the same clock rate as the DA front-end linear filter, we adopt a VLIW approach to issue multiple instructions in parallel \([72, 73]\). This prevents the DA Unit from idling for precious cycles while waiting for the NLSL unit to finish the processing for the most recent input sample. The two functional units include a square-accumulate and a multiply-accumulate block. If the bitwidth of the input samples is low, the number of available clock cycles for this filter to operate is also low. That is acceptable since we expect any long linear filters to be implemented in the DA unit. The NLSL unit only performs some simple nonlinear filtering, for example energy computation, to help determine when to start and end a valid segment.

Although there are techniques for implementing nonlinear filters using Distributed Arithmetic \([74]\), their memory requirements are rather large. Having a multiply-accumulate allows more flexibility in that short linear filters can also be implemented. Since we can
Figure 6.4: DA Unit shift register implementation.

Figure 6.5: Short linear and nonlinear filter implementation architecture.
approximate a nonlinear filter by a truncated power series, having the squarer unit is a convenience that also reduces power since its implementation is simpler than a full multiplier [75]. The Load/Store Unit is used to implement the delay line for a digital filter. It accesses the contents of the buffer relative to a pointer that points to the location where the current output samples are to be written.

6.1.3 Feature Extraction and Classification Microcontroller

Figure 6.6 shows the architecture of the back end processor required for classification of the sensor signal time segment. It is a standard load/store type of computer architecture [73]. The program and data memory is much larger for this unit than the short linear/nonlinear filter block of Section 6.1.2, so having a register file reduces the power by reducing the number of times the data memory bitlines must be switched. Instruction level parallelism is not required in this case since the time constraint is much more relaxed for this block due to the fast clock rate. ILP can be used to reduce the fast clock frequency, but the increased complexity required is not worth the power reduction for a unit that is activated in fast mode rarely. The clock frequencies for sensor applications are so low that the voltage cannot really be reduced to an extent that would decrease power significantly.

6.1.4 Algorithm Workload

The architecture described in this section was simulated using the Verilog hardware description language, to determine the workload breakdown between the preprocessing
Table 6.1
Sensor DSP Chip workload on both phases of heartbeat algorithm.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Frequency</th>
<th>Time</th>
<th>Cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preprocessing</td>
<td>1 kHz</td>
<td>99.8%</td>
<td>230,000</td>
</tr>
<tr>
<td>Feature Extraction and Classification</td>
<td>250 kHz</td>
<td>0.2%</td>
<td>110,000</td>
</tr>
</tbody>
</table>

phase of the algorithm and the feature extraction and classification. Figure 6.1 shows the results of the workload simulation. Although the implementation of the matched filter using Distributed Arithmetic reduces the cycle requirements of the preprocessing because of parallelism, this phase still dominates the cycle count by factor of 2. A much larger difference occurs in real time, since the backend code runs at a much higher clock rate. This analysis shows that the average power consumption of the chip will be dominated by the preprocessing mode since it spends the vast majority of time doing those computations. Thus it is important to optimize these units for low power.

6.1.5 Future Work: Integrated Power Supply Controller

In the future, a low power power supply controller of the same architecture as presented in Chapter 4 should be integrated on the DSP chip. A replica of the critical path, most likely the adder chain for the accumulation in the DA unit, should be used to close the performance feedback loop. The fast clock $\Phi_1$ will be used to set the voltage. At the low operating speeds required for this system, this does not result in significant power dissipation even though we are running at a higher voltage than necessary when $\Phi_0$ is the system clock. The transistors operate at the edge of subthreshold conduction, so small changes in $V_{dd}$ result in large changes in speed without large changes in power consumption. Fixing the reference and doing clock muxing reduces complexity in the DSP-power supply interface. The power optimal approach would be to vary the supply voltage depending on whether $\Phi_0$ or $\Phi_1$ is the system clock. However, the power supply dynamics will then be in the critical path of the throughput constraint since the DSP must wait for the supply to stabilize before beginning computation at the new clock rate. The DC/DC converter dynamics will be slow compared to a DSP clock period and waiting for the transient to settle will degrade performance unacceptably for what little power reduction can be obtained.

If $\Phi_0$ and $\Phi_1$ are very different in frequency, then having the power supply switch between the required voltages for the different operating modes would result in a very large power savings. The ability to do this expands the system's ability to deal with more sophisticated backend signal processing algorithms than the parametric Gaussian classifier implemented in the test system.
Figure 6.7: Period versus power supply voltage for a 17 stage ring oscillator in 0.6 μm CMOS.

6.2 Low Voltage Logic Circuit Design

Figure 6.7 shows the delay versus power supply voltage curve for the 0.6 μm CMOS process used to fabricate the Sensor DSP chip. Voltage reduction is the most direct way of decreasing power consumption and also yields the largest benefits because of the quadratic dependence of the dynamic power $fCV^2$. Beyond a certain point, however, the delay of the circuit increases exponentially with a supply voltage decrease. This is verified in Figure 6.7 where as the supply nears the sum of the absolute value of the device thresholds ($V_{Th} = 730$ mV and $V_{Th} = -810$ mV), the delay curve rises exponentially. In this region of operation, small changes in voltage result in large changes in delay. These small changes do not affect power consumption dramatically, therefore it is more important in this regime to minimize the other main contribution to dynamic power, switched capacitance. The logic circuit design for this DSP focuses on reducing switched capacitance while maintaining the low throughput real time performance typical of sensor applications.

6.2.1 Ripple Carry Adders

The low throughput requirements of typical sensor applications allow for the use of ripple carry adders in the critical path of arithmetic units. All adders in the Sensor DSP chip use the ripple carry architecture since the longest ones, twenty-four full adder cells wide, are
still fast enough for the relevant applications even at low power supply voltages. Ripple carry adders can have power dissipation due to glitching, but since most adders in the design are much smaller than twenty-four bits wide the extra power is not significant.

The previous section has shown that for typical CMOS processes, it is possible to operate circuits on the edge of subthreshold conduction and still achieve adequate performance for sensor applications. Voltage reduction in this regime does not result in large power savings, so the full adder design of Figure 6.8 is a low area, low switched capacitance implementation. It only uses 16 transistors in its smallest implementation and is based on using pass gate XOR gates [56]. The first XOR gate consists of two inverters and transistors N0-1 and P0-1. This computes the propagate signal and its inverse. These signals then go into another XOR gate and a pass gate multiplexer. Because the power supply voltage is low, it is essential that full pass gates are used to ensure adequate transmission of logic high levels.

Because the outputs of this full adder are not fully driven to the supplies, problems occur as adder cells are cascaded. The delay of the pass gate chain grows quadratically with the number of cells and at low $V_{dd}$ the delay becomes intolerable even at low clock frequencies. The low throughput requirements imply that pipelining is undesirable for this design, as the required clock loads and switched capacitance overhead would not pay off in terms of voltage reduction. Therefore the entire computation cycle of instruction fetch, decode, operand fetch, execution, and write back must be completed in one clock cycle. The addition of these elements in the critical path put significant restrictions on the delay of the adder circuits. To solve this, simply buffer the outputs of the carry path. The number of additional transistors can be small since inverting the output of an XOR gate simply requires inverting one of its inputs. In the adders used in this chip, buffers are placed every four full adder cells.

### 6.2.2 Squarers and Multipliers

In addition to adders, the arithmetic units of the chip also require multipliers and one squarer. PowerMill simulations show that using a custom squaring circuit consumes about half the power of using a multiplier with both inputs tied together. The multipliers in this chip are array multipliers since the input words are only 12 bits wide. The low input bit width means using other architectures, like a Wallace tree, is suboptimal. Buffering is again required every four cells, particularly in the final 24 bit ripple carry adder.

The squarer is implemented using a divide-and-conquer architecture [75]. It is a 12 bit implementation based on several smaller array multipliers. The multiplier products are combined using adders and some other standard cells.

To prevent unnecessary glitching leading to extra power dissipation, the inputs to the array units are latched when the units are unused [76]. This latching is purely for power reduction and therefore its timing is not critical; if data flows through them it only results in increased power dissipation and not incorrect circuit operation. The latches are simply positive level sensitive static latches as described in the next section. The amount of
power savings gained by application of this technique varies according to the number of multiplier instructions in any particular execution trace.

6.2.3 Latches and Registers

Pipelining is not necessary in this design for performance or power reduction. Consequently, there are very few registers in this design. Their power consumption is not a major component of chip power although the rather large physical size of the clock net is significant. There are both positive level sensitive latches and edge-triggered flip-flops in the design.

Figure 6.9 is a schematic of a typical edge-triggered flip-flop. The master stage is a dynamic C²MOS latch. The slave stage is an inverter with a passgate input and a tristate buffer to implement feedback. This stage is static so that the clock can be gated low and the state preserved. When the clock goes high, the slave stage blocks the input while the master is transparent and the feedback is broken. While the clock is low, the master is transparent and the feedback is enabled as the slave blocks the data.

Multiple versions of this flip-flop include ones with synchronous and asynchronous clears, enables, and a fully static version (static master and slave) which is used in very low duty cycle functional units like the JTAG registers. Level sensitive latches are static based on the slave stage shown in the figure.
Figure 6.9: Positive edge-triggered flip-flop with asynchronous clear schematic.

6.2.4 Read Only Memories

Several different ROMs are used to implement instruction decoders. Since the total number of ROM bits is small, a relatively large bit cell can be used. The cell is either a PMOS pullup or an NMOS pulldown and both polarities of the wordline are routed through the cell [18]. The increased area does not significantly impact the chip area. The minimum sized PMOS pullup is very slow at the typical $V_{dd}$ for operation and even though the maximum depth of the ROMs is only 32 words, it is necessary to use inverters at the bitlines to buffer the ROM signals.

6.3 Low Voltage SRAM Circuit Design

As described previously, much of the power reduction in the implementation of this DSP chip is due to aggressive voltage scaling and minimization of switched capacitance. Low voltage operation presents challenges for designing the on-chip SRAM circuitry, particularly in the sense amplifier design. Reducing switched capacitance results in a memory architecture that emphasizes small bank size.

6.3.1 SRAM Sensing Scheme

Conventional SRAM sensing schemes involve biasing a differential sense amplifier to magnify the voltage difference between the complementary bitlines [77, 56]. This approach has several drawbacks at low voltage. First, the transistors are biased in the subthreshold region and the resulting amplifier is too slow due to the lack of current drive. Second, biasing
requires static current flow. This constant power dissipation is unacceptable for microwatt level designs with a significant amount of SRAM.

An alternative is to use a single-ended charge-transfer sensing scheme [78]. A diagram of the circuit is shown in Figure 6.10. The SRAM cell is a simple 6-T cell with all minimum-sized devices. For operation near 1 Volt, the NMOS pass transistors cannot pass a high enough voltage to be read as a logic one. The PMOS devices are also substantially weaker than the NMOS due to their reduced carrier mobility and higher threshold voltage, so a low to high transition on the highly capacitive bitline would be exceedingly slow. To eliminate these problems, the bitline (labeled BL in the figure) is precharged high, but through an NMOS device to \( V_{dd} - V_{Ts} \) to save power. The SENSE node is also precharged, but directly to \( V_{dd} \) and the output OUT is precharged to ground.

If the cell drives a one, the bitline remains at its precharged value as does the sense node and the output. If the cell drives a zero, then the bitline capacitance gets pulled slightly low. This voltage is close to the threshold of the passgate device N2. When N2 turns on, the large bitline capacitance is connected to the relatively small sense node capacitance an charge is shared between them. This results in a dramatic voltage swing on the sense node. Once it crosses the output inverter threshold, the output goes to a valid logic high.

Because the bitline is precharged close to the threshold of the passgate device, the scheme is sensitive to noise events on the bitline: a small downward voltage glitch may inadvertently discharge the sense node. This scenario can be averted by ratioing N2 such that its effective threshold is higher than the precharge NMOS device N1. Consequently, N2 is a long-channel, minimum-width device. This approach results in increased read time, but the low throughput requirement and the load-store architecture of the DSP make such a tradeoff acceptable.

Because of the difficulty of passing a high voltage through the NMOS pass transistors at low \( V_{dd} \), the SRAM cell is written differentially. Some memories also support dual read ports and the sensing scheme described in this section is replicated on both bitlines. The second pass gate of the 6-T cell is not shown in Figure 6.10 for clarity of presentation.

### 6.3.2 SRAM Timing Scheme

The sensing scheme described above requires the generation of multiple timing signals. The large on-chip SRAMs are synchronous with respect to the system clock, with reads and writes being triggered by clock edges instead of using address-transition detection (ATD) logic typical in commercial SRAMs [79]. Timing signals are generated using delay lines. Figure 6.11 shows the unit delay element which simply consists of two misratioced inverters. Because the timing budget due the low throughput requirement is so generous, the total delay is controlled open loop and simply determined by the process variation and ambient temperature. The total delay is such that all signals have a large margin of time before the next phase of operation begins. Static CMOS AND gates are used to generate pulses by tapping various points of the delay line.

Figure 6.12 shows the simulated timing of the read operation control signals. The first pulse is the precharge PRE which controls the charging of the bitlines and the dynamic
Figure 6.10: SRAM sensing circuit.

Figure 6.11: SRAM timing generator delay element.
sense nodes. This pulse must be long enough to account for the bitline precharge since it is important that the bitline be high enough to keep the passgate transistor NZ off. The precharge is followed by the word line enable WL which causes the cell to either discharge the bitline or leave it high. Lastly, the result of the sensing operation is latched using the rising edge of the DLATCH signal. The delay between the word line enable and the latching signal must be long enough to account for the sensing operation as well as driving the data bus between memory banks (see Section 6.4).

The simulation results of the read operation are presented in Figure 6.13. In the first part of the cycle, the bitline and the sense node are precharged to their respective voltages. The output node is driven low. For this example, the cell is storing a zero so when the word line is asserted, the bitline BL drops. Charge sharing between the highly capacitive bitline and the small dynamic sense node causes a large voltage swing on the sense node and the output is driven to a valid high level.

Write operations are performed by driving the bitlines using tristate buffers and then enabling the word lines. Entire rows are written at once. Figure 6.14 shows an HSPICE simulation of the write timing control signals. The WEN signal enables the write drivers. Within the WEN pulse, the word lines are driven high and low. This ensures that there is valid data on the bitlines during the entire interval the cells are written.

Figure 6.15 shows a simulation of the write operation. When WEN goes high, the drivers pull the bitline BL high. The capacitance is large enough that it takes about 20 ns to drive the lines. However, writes have a full half clock cycle to complete so there is plenty of timing margin as long as the WEN pulse is long enough in time. When the word line WL is asserted, the storage nodes A and A are switched. These nodes are shown in detail in Figure 6.16. Because the power supply voltage is so low, the pass gates of the SRAM cells cannot pass a reasonable high voltage so writes must be fully differential. Although using boosted wordline voltages might make differential writes unnecessary, the overhead of adding an extra power supply and voltage upconverters is undesirable.

6.4 SRAM Bank Partitioning

A substantial portion of the Sensor DSP chip area is devoted to memories, but the memory area is not a limiting factor in the total chip area. To avoid having the memory power consumption dominate the chip power, these memories were partitioned into banks optimized for power at the expense of area. High-level power estimation was done using switched capacitance estimates and parameters from the CMOS process [80].

Figure 6.17 shows a schematic representation of a single SRAM bank for purposes of power estimation. The problem is how to partition a total number of required memory words among banks to realize the lowest power for the memory. It is important to account for the possible predecoding of addresses as well as the various drivers, precharges, and other circuit operations required for memory reads and writes. Figure 6.17 shows the main contributions to power within a bank: predecoded address lines, normal address lines, word lines to access the cells, and bitlines. Since true and complement versions of
Figure 6.12: Simulated timing waveforms of an SRAM read cycle.

Figure 6.13: Simulated SRAM read waveforms.
Figure 6.14: Simulated timing waveforms of an SRAM write cycle.

Figure 6.15: Simulated SRAM write waveforms.
the normal address lines must be distributed in the address decoder, the probability that an address line is charged per address bit is $\frac{1}{2}$ assuming the address bits are uniformly distributed. Although this assumption does not strictly hold for all of the bits in highly correlated address traces, this is a worst case assumption since the probability of an energy dissipating transition is lower than one half.

The predecoded address lines have different transition probabilities because the AND gates used in the predecoding combine the input bit transition probabilities. From [1], the output transition probability for an N-input CMOS gate is:

$$P_{0\rightarrow1} = \frac{N_0(2^N - N_0)}{2^{2N}}$$  \hspace{1cm} (6.1)

where $N_0$ is the number of zeroes in the truth table and $N$ is the number of inputs. $N_0 = 3$ for a two-input AND gate.

Figure 6.18 shows a schematic representation of a single SRAM bank for purposes of power estimation. Several SRAM subarrays are multiplexed onto a single read bus using tristate buffers. As more subarrays are added, the bitlines become shorter and have less capacitance, but the bus wires get longer and add capacitance. Thus there is an optimum in the number of subarrays the total amount of memory is partitioned into. The total power for the memory is
Figure 6.17: SRAM bank schematic for high-level power estimation.

Figure 6.18: Simple memory floorplan for high-level power estimation.

\[ P_{TOT} = P_{mux} + P_{BL} + P_{WL} + P_{AL} \]  \hspace{1cm} (6.2)

where \( P_{TOT} \) is the total memory bank power, \( P_{mux} \) is the subarray readout bus multiplexer, \( P_{BL} \) is the bitline power, \( P_{WL} \) is the wordline power, and \( P_{AL} \) is the address line power, including the predecoding overhead. Substituting expressions for the various terms yields a more detailed equation for the SRAM power:

\[ P_{TOT} = \frac{M^{2L-N}}{2} \left( \frac{M}{4} + 6 \right) C_u V_{dd}^2 + \frac{M}{2} \left( \frac{2^N}{2} + 2^N \right) C_u V_{dd} (V_{dd} - V_m) \]
\[ + \frac{N}{2} \left( 2^N + \frac{2^N(2+N)}{2} \right) C_u V_{dd}^2 \]  \hspace{1cm} (6.3)

where \( L \) is the total number of memory address bits, \( M \) is the width of memory word, \( C_u \) is a unit capacitance representing gate capacitance, drain capacitance, or wire capacitance...
all normalized to the same standard, \( N \) is the number of address bits for each subarray, \( V_{dd} \) is the power supply voltage, and \( V_{th} \) is the NMOS transistor threshold voltage. The equation assumes that the bitlines are precharged to \( V_{dd} - V_{th} \). Equation 6.3 assumes no predecoding of the address lines for simplicity, but it is straightforward to incorporate the appropriate version of Equation 6.1 into Equation 6.3.

Figure 6.19 shows a plot of the estimated SRAM power for a fixed memory size and varying levels of bank partitioning and address predecoding. There is a relatively shallow minimum for a subarray size of 32 words. This is small for a high density memory, but since the memory requirements for the Sensor DSP chip are purposefully small, this partitioning can be used without a large area impact. Although many assumptions and simplifications were made in this high level analysis, the shallowness of the minimum indicates that there is not a large penalty for choosing a suboptimal partitioning.

6.5 Multiport Register File Design

There are two multiport register files in the chip architecture: a 4-read port, 1-write port block in the VLIW NLSL filter Unit and a 2-read port, 1-write port register file in the microcontroller. Multiport register files also require careful design because of the greatly decreased current drive available at the low power supply voltages. Rather than use a
Figure 6.20: Register file cell for 2-read port, 1-write port register file.

high-performance, 6-T SRAM cell based design which adds ports by increasing the number of pass gates [81], we use a more conservative scalable circuit [56].

Figure 6.20 is a circuit schematic of a 2-read port, 1-write port register file cell. It is based on a 6-T SRAM cell but the two bitlines \textit{WB}IT and \textit{WB}IT are used solely for writes. The storage nodes are isolated from the read lines using the inverter formed by \textit{P0} and \textit{N0}. This inverter drives the read port pass gates which are fully complementary transmission gates. At low power supply voltages, the high voltage passed by an NMOS transistor is unacceptably low. The advantage of this design is the easy scalability of the read ports. More can be added by simply increasing the number of full pass gates driven by the isolating inverter. A four read port version of this cell is used in the NLSL filter unit. The distributed arithmetic tables also use register file cells, although since there is only one read port the extra inverter is removed and one of the write NMOS pass transistors is replaced by a full pass gate.

6.6 Clock Design

The real-time constraints of the heartbeat detection algorithm and other detection and classification applications, can be met using a clock multiplexing approach as in this system study. Assuming the input samples are coming in at a constant rate, the extra cycles required to compute a classification of an event can occur in the intersample time if the backend processing unit is clocked at a higher clock frequency. The power supply voltage is high enough to support this higher throughput requirement.

Putting multiplexing logic in the clock path is a dangerous design practice. Changing the clock frequency in this chip is performed using a pulse swallowing or clock gating technique. The clock generation unit takes a clock whose frequency is higher than the highest
required frequency for real-time performance. A counter is clocked using this input clock and its outputs are used to enable a toggle flip-flop. This ensures that the output clock is glitch-free since the toggle flip-flop state only changes on a clock edge. A timing signal for the on-chip SRAMs is also generated in this manner such that it maintains a certain phase relationship with the output clock. Skew is not a major issue in this design since the clock period is very long compared to mismatches in the clock distribution net. There is only one large clock driver at the end of buffer up tree, so there is no buffer mismatch either.

6.7 JTAG Programming Interface

To reduce the number of pins required for the Sensor DSP chip, the programmable memories and configurations are written through a serial interface. This interface implements a subset of the IEEE Standard 1149.1 [82] for a Test Access Port. One instruction register is loaded serially with an address that, when decoded, connects the JTAG port serial input to one of several data registers. These registers are then serially loaded with data for instruction memories, the Distributed Arithmetic Unit tables, and various configuration states. The outputs of these registers are also multiplexed to the serial output of the JTAG port. The data registers are also used to implement some SRAM write timing signals for programming the instruction memories.

6.8 Chip Test Board

To test the Sensor DSP chip, a printed circuit board was constructed that interfaces the chip to a personal computer. A block diagram of the PC board is shown in Figure 6.21. All data to the chip being tested goes through the personal computer. Inputs include the JTAG programming signals, the chip clock, the serial data input, resets, enables, and configuration signals for the 12 bit chip output test data bus. This bus feeds into the PC interface as well. The cable from the PC is terminated with resistors and the signals are improved using Schmitt triggers. The output of the chip is buffered before going back to the PC. A commercial A/D converter is also placed on the board and its output can optionally be connected to the PC as well. The control signals for the A/D including configuration and sample clock come from the PC interface. Since all clocks come from the computer, there are no crystals or other clock generation circuitry on the board.

The simplicity of the test board and the PC interface allowed for a very elegant testing flow. The chip was stimulated using vectors created from a Verilog emulation of the chip; the same emulator that was used for the final chip verification. Any future algorithm development can proceed by testing the assembly code, filter coefficients, and JTAG programming sequence directly on the Verilog emulator. These vectors can be transferred to the PC and are output directly by the interface card. The timing of these vectors is determined by the C test program running on the PC. Unfortunately, since this program runs under Linux, its real-time performance is quite poor and it is unable to sustain high clock frequencies. The use of a real-time operating system or porting of the test code to a
Windows environment should alleviate this problem. However, the PC interface is quite capable of determining logical correctness for the chip and performing some real-time experiments if the required throughput is low enough.

6.9 Sensor DSP Chip Test Result Summary

The chip described in this chapter was fabricated in a 0.6 µm standard CMOS process with three levels of metallization. It was tested and verified using eight bit sensor data at a clock rate of 1.2 kHz. At this frequency, the voltage could be scaled down to 1.1 V before logical failure. Since many circuits lie in the critical path of the system, the increased sum of their delays all contribute to violating the cycle time. This voltage results in a power consumption of 300 nW. To satisfy the backend processing requirements in real-time, the voltage would have to be scaled to 1.5 V, even for the low frequency filtering operations. This results in 550 nW of average power consumption, again far below what is achievable for a vibration-based power system as shown in Chapter 4.

The implementation of the matched filter unit using Distributed Arithmetic results in the ability to scale the power consumption for this unit with the required performance. Figure 6.22 shows the measured power consumption of the Sensor DSP chip as it is configured for decreasing input data bitwidth. Only the Distributed Arithmetic unit and microcontroller are enabled for these measurements. The graph shows that the power decreases monotonically as the the bitwidth decreases, as expected from Chapter 5. These functional units have about 100 nW of overhead power at 1.1 V and a 1.2 kHz clock frequency. Figure 6.23 shows the power scalability for the entire frontend signal processing enabled. The
Figure 6.22: Measured Sensor DSP Chip Power, DA Unit and microcontroller active.

NLSL filter unit and the filter buffer add about another 110 nW of power consumption for the chip.

In Section 6.1.4 it was asserted that the preprocessing phase of the signal processing would dominate the chip power. Figure 6.24 shows the power breakdown between the two phases using simulated data from the PowerMill switch-level circuit simulator. The backend processing consumes about 20% of the average chip power. This is much higher than the percentage of time this processing occurs since the clock rate here is 250X higher. The figure shows that even with the extensive measures taken to reduce the preprocessing power consumption, it is still the dominant component of chip power. The total power for the simulated chip is a factor of 2 lower than the measured power because it neglects the interconnect capacitance.

Each functional unit described in Section 6.1 contributes some fraction of the preprocessing power consumption. The simulated results for the preprocessing computation are shown in Figure 6.25. The specialized filter units and the synchronizing filter buffer consumes 57% of the power in this mode while the microcontroller consumes 43%. The overhead of the microcontroller is significant in this mode and it may be profitable to reduce its power by replacing it with a specialized control flow unit. However, the total power is so low that such optimizations do not affect the chip's applicability to all but the very lowest power applications.

Figure 6.26 displays the SRAM timing signals generated from the on-chip delay lines to show verification of the memories. The relative timings are the same as the simulated results shown in Figure 6.12. A chip photo is shown in Figure 6.27. The design is almost
Figure 6.23: Measured Sensor DSP Chip Power, all units active.

Figure 6.24: Simulated power breakdown between frontend and backend processing units of the Sensor DSP chip.
entirely full custom, with the few standard cells used placed and routed by hand. The chip area is dominated by memories: the Distributed Arithmetic tables, filter buffer SRAM, and microcontroller instruction and data memories. The bulk of the arithmetic area is consumed by the array multipliers and the adder tree for the Distributed Arithmetic unit. Empty space is filled with bypass capacitance and dummy metal 3 for process reasons.

Table 6.2 summarizes the chip and process parameters. The 0.6 \( \mu \text{m} \) process used for this implementation is a 5 volt process, accounting for the relatively large threshold voltages.

Custom and hybrid custom-programmable DSP architectures can result in dramatic energy efficiency when compared to fully programmable microprocessor solutions. The same detection and classification algorithm was run on a StrongARM microprocessor [20] based on a C implementation. Table 6.3 shows that the energy per input sample for the hardware implementation is six orders of magnitude lower than the microprocessor implementation. Energy is used as the metric since the StrongARM cannot be clocked slow enough for a real-time sensor application like the heartbeat monitor.

The \( 10^6 \) improvement in energy per sample comes from a variety of factors in the design of the Sensor DSP chip. The StrongARM microprocessor is designed for palmtop computing devices; consequently its general purpose architecture is poorly suited to sen-
Figure 6.26: Measured SRAM timing waveforms.

<table>
<thead>
<tr>
<th>SensorDSP Chip Power</th>
<th>556 nW</th>
</tr>
</thead>
<tbody>
<tr>
<td>SensorDSP Chip Energy</td>
<td>26.5 pJ per sample</td>
</tr>
<tr>
<td>StrongARM SA-1100 Energy</td>
<td>11 μJ per sample</td>
</tr>
</tbody>
</table>

Table 6.3
Sensor DSP Chip Power versus a StrongARM implementation.
Figure 6.27: SensorDSP chip photograph.
sensor applications. The Sensor DSP chip uses a narrow bitwidth throughout its datapath since typically sensor data is not very wide. The hybrid architecture of a power optimized filter unit and a programmable microcontroller ensures that the majority of the algorithmic workload runs on a power optimized computing substrate. The multiply-accumulate implementation that is required on the StrongARM is not efficient for long impulse response filters like the heartbeat matched filter. The much simpler instruction set of the Sensor DSP microcontroller ensures that there is less control overhead for it than the more complicated programming model of the StrongARM. In addition, the StrongARM attempts to be a high performance chip. It therefore requires static power overhead in biasing an on-chip PLL and in the sense amplifiers for its large memory arrays. The large size of the StrongARM die also exacts a cost: large capacitances like the clock net must be switched. The very small size of the Sensor DSP chip, although it limits its computational performance, reduces the power considerably. The StrongARM fabrication process is optimized to deliver speed at low power supply voltages and so the leakage current for it is quite high. In contrast, the Sensor DSP uses a mature process with high threshold voltages and avoids the leakage problem. All of these factors play a part in the large energy performance gains seen in this custom chip.
Chapter 7

Conclusions and Future Work

This final chapter summarizes the contributions of the research presented in this dissertation. The contributions span a wide range of areas, however, there is a large amount of work for future researchers to unlock the potential of using ambient energy sources in electronic applications.

7.1 Ambient Energy Sources

Chapter 2 describes the tradeoffs between using ambient energy sources and stored energy. Traditionally, electrical energy has been stored in batteries. Recent research has opened the possibility ofcombusting fossil fuels to create electrical power by using a microfabricated turbine generator. Advanced capacitor technology may someday enable the use of direct electrical energy storage. Each of these storage techniques has its own set of drawbacks that make them unsuitable for some low power VLSI applications, in particular long lifetime embedded sensors. For example, even though fuel combustion can store enough energy to supply a microwatt load for years, it may not be able to deliver power at such a low level.

Numerous sources of ambient energy have also been considered as power sources. Solar power is the most familiar and it has seen several commercial applications in portable electronics as well as large infrastructure applications. Research has shown that thermal gradients and fluid flow can be used for small power electronics whereas historically these forms of energy have been used for large scale energy needs (geothermal sources and windmills). Electromagnetic radiation other than light has also been used for electronics, although the paradigm for the use of RF fields or magnetic fields is different than energy harvesting. These sources have to be provided by the user as a kind of wake up signal. By contrast, a true ambient source exists in the environment independently of an intentional user stimulus. The energy derived from these stimuli is then used for readout of information. The small field strengths of truly ambient electromagnetic radiation make it unlikely that this source will be used for electronics.
Mechanical vibration is the power source most carefully analyzed in this dissertation. The advantages of this source is that it is widely available at a variety of different frequencies and in a variety of environments, including sealed packages and embedded into structural materials. It is also available in the most likely first use for energy harvesting: embedded sensors for machine monitoring. Large capital equipment creates a lot of vibration as it operates. The tables from Chapter 2 show that machine vibration is a viable power source for low to medium throughput sensor applications like the vibration FFT evaluated in Chapter 1.

7.2 Vibration to Electrical Energy Transducers

Many different transducers exist for converting vibration to electrical energy. Accelerometers and microphones are two examples. Transducers optimized for power transfer need not be as linear as real vibration sensors. The power derived from vibration can be as high as a few watts for a macroscopic generator, but microwatts to hundreds of microwatts is more typical for miniature and microfabricated transducers.

The first transducer described in detail is an electromagnetic one which works like a moving-coil loudspeaker in reverse. A wire coil is attached to a mass which is connected to a spring. The mass is free to move but one end of the spring is connected to a rigid housing. As the housing is vibrated, the energy couples to the mass. The mass moves up and down and the coil moves in and out of an air gap cut into a permanent magnet. By Faraday’s Law, a time varying magnetic flux creates a voltage. This voltage signal is rectified onto a capacitor. From there, a DC/DC converter can regulate the output to a desired level for powering logic. Although the particular example discussed in Chapter 3 required a transformer to boost its output voltage, this can be compensated for by adding more turns to the coil, increasing the magnetic field, or tuning the generator to a higher vibration frequency. Although it is possible to integrate permanent magnets into a microfabrication process, this transduction principle is more applicable to discrete generator designs.

A variable capacitor transducer is easily incorporated into a MEMS process. In this device, a certain amount of charge is placed in the capacitor. As the capacitor plates move apart due to vibration, energy is stored in the electric field of the capacitor. The device can be operated in either a constant charge or constant voltage conversion cycle. Power on the order of 100 μW has been estimated to be available. The chief downside of this transducer is that it requires a source of energy to begin the conversion cycle, unlike the moving-coil approach. Power electronics that control the conversion cycle have been developed that only consume 4 μW of power [51].

Another possible transducer is a piezoelectric disk. When a piezoelectric material is deformed, it creates a voltage, often a very high one. Very low currents are available from this source and so it may not be sufficient even for low throughput sensor applications. Piezoelectrics have been suggested as inserts for shoes for use in higher power applications [15]. In summary, there are a number of transduction techniques applicable to vibration to elec-
tric energy conversion. Future work includes a more detailed and systematic examination of transducer designs.

7.3 Low Power DC/DC Conversion

Power electronics is an essential component of self-powered systems since generators produce a time-varying output voltage. The requirements for DC/DC conversion for self-powered systems are similar to battery-powered systems, but battery discharge curves are much slower in time than the vibration frequencies of interest, for example.

Chapter 4 showed an example of how traditional DC/DC conversion techniques can be applied to self-powered systems. By reducing some requirements on the performance of the power electronics and by employing performance feedback instead of voltage feedback, it is possible to implement very low power digital control. The low power overhead of this control enables operation from ambient vibration. In the example, the converter consumes under 14 \( \mu \text{W} \), which means it can be powered by both the moving-coil design it was tested with and the proposed capacitive MEMS device.

7.4 Energy Scalable DSP Algorithms

Sensors and other electronic systems powered from ambient energy sources require low power algorithms to be viable. The primary advantage of these systems for low power is their low to medium fixed throughput requirement. A large number of algorithmic, architectural, and circuit techniques have been applied to reducing power consumption for these systems [1]. Chapter 1 shows using high level power estimation that a low throughput FFT has power requirements well within what is achievable using ambient techniques, even without using power reduction techniques.

Besides the FFT, detection and classification are also typical applications for sensor signal processing. An example explored in detail in this dissertation is the detection of acoustic heartbeats sensed by a microphone worn on the body. The signal processing flow involves a large amount of preprocessing which consists of different types of filtering. The preprocessing does not require much programmability or control flow. On the other hand, the backend processing of actual classification does require programming flexibility. These complementary requirements drive the DSP chip architecture reviewed in Chapter 6. This architectural specialization enables a low power implementation of the desired algorithm.

To enable a power and performance tradeoff, a novel technique is used to implement the linear filtering which makes up the bulk of the preprocessing. Distributed arithmetic is a bit serial technique for computing dot products. By performing the computation using the most significant bit first, the power consumption can be reduced simply by shifting in fewer bits of the filter input data. This also reduces the filter’s performance by effectively reducing the signal to noise ratio at the filter input. This energy scalability is a useful feature, especially in situations where the amount of available ambient energy varies.
7.5 Low Power Sensor DSP

Low power and power scalable algorithms must be implemented on an architecture that is optimized for low power. As described in the preceding section, a low power sensor DSP chip has been implemented that is a hybrid custom and programmable architecture. The front end signal processing is performed on a distributed arithmetic filter unit while some auxiliary filtering is implemented on a small programmable filter unit that does not implement any control flow. The backend signal processing (time series segmentation and classification for the heartbeat detection algorithm) is computed on a programmable microcontroller. The chip has orders of magnitude lower energy per operation cost than a similar implementation on a StrongARM microprocessor.

The low throughput requirement of the algorithm allows aggressive voltage scaling for power reduction. As shown in Chapter 6, the regime of operation for the circuits is such that small changes in the power supply voltage results in large changes in circuit delay. For further power reduction, it is more important to reduce the switched capacitance of the circuit rather than make further changes to reduce voltage. Consequently, the bulk of the arithmetic units on the chip are based on a very small full adder cell. Fewer transistors and small area both reduce switched capacitance. In addition, the large memory arrays are partitioned in an optimal way to reduce power by reducing switched capacitance.

A proof-of-concept demonstration chip has been implemented in 0.8 \( \mu \)m standard CMOS technology. It consumed 220 nW of power when clocked at 1.2 kHz with a power supply of 1 Volt. This level of power consumption is much lower than that available from vibration sources, both for the macroscopic and micromachined transducers. It is also smaller than the energy required for the power electronics, which shows that energy scavenging systems have the feature that the load electronics may be negligible compared to the cost of extracting the ambient energy.

7.6 Energy Harvesting and Future Work

This dissertation has barely scratched the surface of the potential of using ambient energy to power VLSI systems. To limit the scope of the research, we have focused purely on using mechanical vibration as a power source and a heartbeat detection algorithm as the user application. Future work involves expanding the number of different energy sources that can be used to include thermal sources, fluid flow, and possibly electromagnetic radiation. Solar power is a fairly well understood ambient energy source. Much work remains to be done regarding exploitation of vibration as an energy source, including transducer optimization and migration to various MEMS technologies. MEMS holds the exciting possibility of integrating the load electronics with the power generation device and possibly the sensor transducer as well. Integration, in addition to reducing system volume, also reduces power by eliminating interconnect capacitance and other parasitics between the system components.

Low power design of electronics, in particular DSP chips, is currently a well understood field. From a systems perspective, the majority of power is dissipated in analog
and communication circuits rather than in digital processing units. Communication and sensing are explicitly not researched in this dissertation. Future work should concentrate on reducing the power requirements of these circuits to fold them into the self-powered paradigm.

This thesis has shown a top-to-bottom exploration of the feasibility of using ambient energy as a power source for electronic systems. There are a wide variety of sources available, but the choice of any particular one depends on the application at hand. Environmental factors determine which sources are present and in what quantity and the load application decides how much energy is necessary. The alternative technologies of energy storage may be more attractive, especially for limited lifetime solutions primarily because of their simplicity and reliability. On the other hand, the lifetime requirement may dictate a volume for stored energy solutions that is not feasible due to space constraints. Yet another alternative is to combine stored energy and ambient energy to provide reliability and long lifetime within a fixed volume. All of these factors must be considered when designing a self-powered system. Energy harvesting is a practical technique, but its uses are restricted to low power applications in general because of the ambient energy levels available. For the systems discussed in this dissertation, the power requirements of the load are so low that it is the overhead of converting ambient energy that dominates the system power consumption. The main reason for this is the use of power electronics that must have a 100% duty cycle and a switching frequency comparable to or higher than the load. This cost is not unique to energy harvesting systems, but to any aggressively designed low power system regardless of whether the energy is stored or ambient. Therefore low energy efficiency is not necessarily a problem for ambient energy systems. Future advances in low power design and integrated circuit technology will expand the functionality and power of loads that can be run from ambient energy and yet more applications will become practical. For these reasons, energy harvesting is a promising technique that will likely bring many technological benefits to the design of future electronic systems.
References


[61] Lockheed Sanders, Inc., Hudson, NH, *Classification Toolbox Documentation*.


Appendix A

Sensor DSP Microcontroller Assembly Language

The backend processing of the Sensor DSP chip is done on a microcontroller with a custom instruction set architecture. The microcontroller is a load/store, single pipeline stage architecture. This appendix is a user guide and documentation for the microcontroller ISA.

A.1 Instruction Set Overview

Figure A.1: Sensor DSP microcontroller architecture.
The programming model for the Sensor DSP microcontroller is shown in Figure 6.6 and repeated in Figure A.1 for convenience. It is a straightforward single instruction word RISC architecture. Both the filter buffer and the microcontroller data memory share the same address space, as will be discussed below.

<table>
<thead>
<tr>
<th>Instruction Types</th>
<th>(OPCODE) 0000 0000 00000000</th>
<th>no operand</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(OPCODE) 0000 (ADDRESS)</td>
<td>unconditional jump</td>
</tr>
<tr>
<td></td>
<td>(OPCODE) (DEST) (ADDRESS)</td>
<td>direct memory instruction</td>
</tr>
<tr>
<td></td>
<td>(OPCODE) (CC BIT) (ADDRESS)</td>
<td>conditional jump</td>
</tr>
<tr>
<td></td>
<td>(OPCODE) (DEST) (CONSTANT)</td>
<td>load constant</td>
</tr>
<tr>
<td></td>
<td>(OPCODE) (DEST) (SRC 0) 00000000</td>
<td>2 operand w/o constant</td>
</tr>
<tr>
<td></td>
<td>(OPCODE) (DEST) (SRC 0) (CONSTANT)</td>
<td>indirect memory instruction</td>
</tr>
<tr>
<td></td>
<td>(OPCODE) (DEST) (SRC 0) (SRC 1) 0000</td>
<td>2 operand with implicit destination</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 operand w/ arithmetic constant</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 operand w/ constant field padding</td>
</tr>
</tbody>
</table>

Table A.1
Microcontroller instruction types.

The Sensor DSP microcontroller is a twelve bit data load/store architecture with only one pipeline stage. Instructions are fetched from the instruction memory, decoded, operands are read from the register file, an operation is performed, and the result written back all in one cycle. Consequently, there are no concerns about pipeline bubbles. All instructions take exactly one clock cycle to complete so there is no possibility of out-of-order execution or issuing instructions while previous instructions are still “live”. All of this makes programming the machine fairly simple.

The instruction types that the microcontroller opcode decoder recognizes are shown in Figure A.1. The actual instruction memory locations must be padded with extra 0’s when a field will not be interpreted. There are 32 instructions for a five bit opcode. There are also 16 registers so register specifiers require 4 bits each. Loads and stores to the register file are done explicitly and must be performed before the operands are used in any subsequent arithmetic operations. Writes of arithmetic operation results only go back to the register file. Arithmetic operations have either two sources and one destination register or one source and one destination register. Conditional jumps require a field specifier which accesses one bit of the Condition Code (CC) register. Jump targets are twelve bits wide as well for compatibility reasons: the instruction memory on-chip is only 256 instructions deep. There are twelve address bits for data memory accesses as well. There are two kinds of constant operations: explicit loads take full twelve bit constants while some arithmetic operations have implicit four bit constants.
A.2 Registers and Other State

There are two sets of state that may be written by the microcontroller program. The first set is the set of registers that store the temporary values for the computation. The second is a group of bits that control the configuration of the front-end processing units and chip clocks, thus making the microcontroller the master functional unit on the chip.

<table>
<thead>
<tr>
<th>Register Specifiers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Special Registers</td>
</tr>
<tr>
<td>Mnemonic</td>
</tr>
<tr>
<td>----------</td>
</tr>
<tr>
<td>DA_ACC</td>
</tr>
<tr>
<td>ACC0</td>
</tr>
<tr>
<td>ACC1</td>
</tr>
<tr>
<td>BUFPTR</td>
</tr>
<tr>
<td>CC</td>
</tr>
<tr>
<td>MUACCL</td>
</tr>
<tr>
<td>MUACCH</td>
</tr>
<tr>
<td>Other Registers</td>
</tr>
<tr>
<td>R7-R15</td>
</tr>
</tbody>
</table>

Table A.2

Microcontroller registers.

Table A.2 shows the register specifiers for the various registers in the architecture. There are numerous special registers that interface with the preprocessing units (the Distributed Arithmetic Unit and the Nonlinear/Short Linear Filter Unit). The DA_ACC register is implicitly loaded with the results of the DA operation from the filter buffer. Similarly, the ACC0 and ACC1 registers store the results of the two filter ALUs from the NLSL Unit, also via the filter buffer. BUFPTR points to the location of the current sample in the filter buffer. The condition code register CC implicitly stores several bits from the datapath that are useful for conditional operations. Its use will be described in detail below. Lastly, MUACCH and MUACCL are the high low twelve bit words of the twenty-four bit wide microcontroller accumulator. The results of certain arithmetic operations have twice as many bits as the datapath operands so it is necessary to have this register store the full result. The user is then free to implement any roundings or truncations he or she sees fit. Registers R7 through R15 are general purpose registers and are never implicitly written.

Table A.3 shows the various specifiers for the bits of the condition code register. Conditional instructions branch based on the value of the one bit specified using these mnemonics. These specifiers and their use will be described in more detail below in the discussion of control flow instructions in Section A.3.7. Note that in addition to the mnemonics, the strings CC[1] may also be used as register field specifiers. CC[0] is equivalent to SIGN12 to the instruction decoder.

The Sensor DSP microcontroller is also responsible for software enables of the preprocessing filter units and selection of the appropriate clock frequency for real-time operation. This is accomplished using the configuration bit specifiers shown in Table A.4. See
### CC Register Field Specifiers

<table>
<thead>
<tr>
<th>CC Field</th>
<th>Mnemonic</th>
<th>Opcode</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>CC[0]</td>
<td>SIGN12</td>
<td>0000</td>
<td>ALU 12 bit result sign bit</td>
</tr>
<tr>
<td>CC[1]</td>
<td>OFLW12</td>
<td>0001</td>
<td>ALU 12 bit result overflow bit</td>
</tr>
<tr>
<td>CC[2]</td>
<td>WNOR12</td>
<td>0010</td>
<td>wired NOR of ALU 12 bit result</td>
</tr>
<tr>
<td>CC[5]</td>
<td>LOW12</td>
<td>0101</td>
<td>ALU 12 bit result low bit</td>
</tr>
<tr>
<td>CC[6]</td>
<td>WOR12</td>
<td>0110</td>
<td>wired OR of ALU 12 bit result</td>
</tr>
<tr>
<td>CC[7]</td>
<td>PCSEL</td>
<td>0111</td>
<td>jump select bit result</td>
</tr>
<tr>
<td>CC[8]</td>
<td>WHIGH</td>
<td>1000</td>
<td>wired high</td>
</tr>
<tr>
<td>CC[9]</td>
<td>WLOW</td>
<td>1001</td>
<td>wired low</td>
</tr>
<tr>
<td>CC[10]</td>
<td>DATA</td>
<td>1010</td>
<td>new buffer data valid bit</td>
</tr>
</tbody>
</table>

**Table A.3**

Condition code register field specifiers for conditional jumps.

### Configuration Bit Specifiers

<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Opcode</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>DA_ENABLE</td>
<td>0000</td>
<td>distributed arithmetic unit enable</td>
</tr>
<tr>
<td>NLSL_ENABLE</td>
<td>0001</td>
<td>nonlinear/short linear filter unit enable</td>
</tr>
<tr>
<td>DATA_VAL</td>
<td>0010</td>
<td>filter buffer valid data bit</td>
</tr>
<tr>
<td>BUF_ENABLE</td>
<td>0011</td>
<td>filter buffer enable</td>
</tr>
<tr>
<td>ALL_ENABLE</td>
<td>0100</td>
<td>enable all units</td>
</tr>
<tr>
<td>FAST_MODE</td>
<td>0101</td>
<td>fast mode select</td>
</tr>
</tbody>
</table>

**Table A.4**

Configuration state bit specifiers for configuration instructions.

Section A.3.9 for instructions and examples of their use.

### A.3 Instruction Descriptions

In this section, we describe in detail the commands that the assembler understands. In addition to explicit instructions implemented in the hardware, there are several macros which the assembler expands into one line assembly instructions. All macros are also implemented in one cycle. Some additional commands for naming branch targets and aliasing constants and memory locations are also described.

Note that the assembler program mustlasm is case-insensitive, even though the majority of the example code is uppercase.

#### A.3.1 RTL Description
### Instruction Syntax and RTL Description

<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Oprnd 1</th>
<th>Oprnd 2</th>
<th>Oprnd 3</th>
<th>Oprnd 4</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOP</td>
<td>[R1]</td>
<td>[R2]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MACC</td>
<td>[R1]</td>
<td>[R2]</td>
<td>[R3]</td>
<td></td>
<td>((\text{MUACC}) \leftarrow \text{MUACC} + (R1) \times (R2))</td>
</tr>
<tr>
<td>MDEC</td>
<td>[R1]</td>
<td>[R2]</td>
<td></td>
<td></td>
<td>((\text{MUACC}) \leftarrow \text{MUACC} - (R1) \times (R2))</td>
</tr>
<tr>
<td>MULT</td>
<td>[R1]</td>
<td>[R2]</td>
<td></td>
<td></td>
<td>((\text{MUACC}) \leftarrow (R1) \times (R2))</td>
</tr>
<tr>
<td>ADD</td>
<td>[R1]</td>
<td>[R2]</td>
<td>[R3]</td>
<td></td>
<td>((R1) \leftarrow (R2) + (R3))</td>
</tr>
<tr>
<td>SUB</td>
<td>[R1]</td>
<td>[R2]</td>
<td>[R3]</td>
<td></td>
<td>((R1) \leftarrow (R2) - (R3))</td>
</tr>
<tr>
<td>ADDC</td>
<td>[R1]</td>
<td>[R2]</td>
<td></td>
<td>[const]</td>
<td>((R1) \leftarrow (R2) + [\text{const}])</td>
</tr>
<tr>
<td>SUBFC</td>
<td>[R1]</td>
<td>[R2]</td>
<td></td>
<td>[const]</td>
<td>((R1) \leftarrow [\text{const}] - (R2))</td>
</tr>
<tr>
<td>NOT</td>
<td>[R1]</td>
<td>[R2]</td>
<td></td>
<td></td>
<td>((R1) \leftarrow (R2)) (bitwise NOT)</td>
</tr>
<tr>
<td>NAND</td>
<td>[R1]</td>
<td>[R2]</td>
<td>[R3]</td>
<td></td>
<td>((R1) \leftarrow (R2) \times (R3)) (bitwise NAND)</td>
</tr>
<tr>
<td>AND</td>
<td>[R1]</td>
<td>[R2]</td>
<td>[R3]</td>
<td></td>
<td>((R1) \leftarrow (R2) \times (R3)) (bitwise AND)</td>
</tr>
<tr>
<td>NOR</td>
<td>[R1]</td>
<td>[R2]</td>
<td>[R3]</td>
<td></td>
<td>((R1) \leftarrow (R2) + (R3)) (bitwise NOR)</td>
</tr>
<tr>
<td>OR</td>
<td>[R1]</td>
<td>[R2]</td>
<td>[R3]</td>
<td></td>
<td>((R1) \leftarrow (R2) + (R3)) (bitwise OR)</td>
</tr>
<tr>
<td>XOR</td>
<td>[R1]</td>
<td>[R2]</td>
<td>[R3]</td>
<td></td>
<td>((R1) \leftarrow (R2) \times (R3)) (bitwise XOR)</td>
</tr>
<tr>
<td>EQ</td>
<td>[R1]</td>
<td>[R2]</td>
<td>[R3]</td>
<td></td>
<td>((R1) \leftarrow (R2) \times (R3)) (bitwise EQ)</td>
</tr>
<tr>
<td>LRSHFT</td>
<td>[R1]</td>
<td>[R2]</td>
<td></td>
<td></td>
<td>((R1) \leftarrow {0, (R2)[11:1]})</td>
</tr>
<tr>
<td>LLSHFT</td>
<td>[R1]</td>
<td>[R2]</td>
<td></td>
<td></td>
<td>((R1) \leftarrow {(R2)[10:0], 0})</td>
</tr>
<tr>
<td>ARSHFT</td>
<td>[R1]</td>
<td>[R2]</td>
<td></td>
<td></td>
<td>((R1) \leftarrow {(R2)[11], (R2)[11:1]})</td>
</tr>
<tr>
<td>LCIRC</td>
<td>[R1]</td>
<td>[R2]</td>
<td></td>
<td></td>
<td>((R1) \leftarrow {(R2)[0], (R2)[11:1]})</td>
</tr>
<tr>
<td>RCIRC</td>
<td>[R1]</td>
<td>[R2]</td>
<td></td>
<td></td>
<td>((R1) \leftarrow {(R2)[10:0], (R2)[11]})</td>
</tr>
<tr>
<td>JUMP</td>
<td>[addr]</td>
<td></td>
<td></td>
<td></td>
<td>((\text{MUCTRLPC}) \leftarrow [\text{addr}])</td>
</tr>
<tr>
<td>CTJUMP</td>
<td>[CC][i]</td>
<td>[addr]</td>
<td></td>
<td></td>
<td>if ((\text{CC}[i])) then ((\text{MUCTRLPC}) \leftarrow [\text{addr}]) (\text{else} (\text{MUCTRLPC}) \leftarrow (\text{MUCTRLPC}) + 1)</td>
</tr>
<tr>
<td>CFJUMP</td>
<td>[CC][i]</td>
<td>[addr]</td>
<td></td>
<td></td>
<td>if ((\text{CC}[i])) then ((\text{MUCTRLPC}) \leftarrow [\text{addr}]) (\text{else} (\text{MUCTRLPC}) \leftarrow (\text{MUCTRLPC}) + 1)</td>
</tr>
<tr>
<td>LOAD</td>
<td>[R1]</td>
<td>[addr]</td>
<td></td>
<td></td>
<td>((R1) \leftarrow (\text{DMEM}[\text{addr}]))</td>
</tr>
<tr>
<td>STOR</td>
<td>[R1]</td>
<td>[addr]</td>
<td></td>
<td></td>
<td>((\text{DMEM}[\text{addr}]) \leftarrow (R1))</td>
</tr>
<tr>
<td>LDI</td>
<td>[R1]</td>
<td>[R2]</td>
<td></td>
<td></td>
<td>((R1) \leftarrow (\text{DMEM}[\text{addr}]))</td>
</tr>
<tr>
<td>STI</td>
<td>[R1]</td>
<td>[R2]</td>
<td></td>
<td></td>
<td>((\text{DMEM}[\text{addr}]) \leftarrow (R1))</td>
</tr>
<tr>
<td>ldc</td>
<td>[R1]</td>
<td>[const]</td>
<td></td>
<td></td>
<td>((R1) \leftarrow [\text{const}])</td>
</tr>
<tr>
<td>CONFIG</td>
<td>[CNF][i]</td>
<td>[state]</td>
<td></td>
<td></td>
<td>((\text{CNF}[i]) \leftarrow [\text{state}])</td>
</tr>
</tbody>
</table>

**Table A.5**  
Microcontroller instruction syntax and RTL level description.
The instruction syntax and RTL description for the explicitly implemented instructions are shown in Table A.5. Macros and other abstracted instructions get their operands from the same fields as their underlying implementations, which are described in various tables throughout this manual. Refer to Table A.5 to determine which fields operands are coming from and results are going to within a program.

### A.3.2 Miscellaneous Instructions

<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Opcode</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOP</td>
<td>00000</td>
<td>separate instruction to disable accesses and save power</td>
</tr>
</tbody>
</table>

**Table A.6**
Arithmetic instructions and opcodes.

Table A.6 summarizes the miscellaneous and utility instructions for the architecture. Each of these will be described in turn below.

**NOP**

This is the null operation instruction. It is implemented by simply turning off the write enables to the microcontroller register file.

Example: **NOP**.

### A.3.3 Arithmetic Instructions

<table>
<thead>
<tr>
<th>Arithmetic Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mnemonic</td>
</tr>
<tr>
<td>----------</td>
</tr>
<tr>
<td>MACC</td>
</tr>
<tr>
<td>MDEC</td>
</tr>
<tr>
<td>MULT</td>
</tr>
<tr>
<td>ADD</td>
</tr>
<tr>
<td>SUB</td>
</tr>
<tr>
<td>ADDC</td>
</tr>
<tr>
<td>SUBFC</td>
</tr>
</tbody>
</table>

**Table A.7**
Arithmetic instructions and opcodes.

Table A.7 summarizes the arithmetic instructions for the architecture. Each of these will be described in turn below.
MACC

MACC is the multiply-accumulate instruction. All multiplication related instructions implicitly send their results to the multiplier accumulator register MUACC and so require only two source operands to be specified instead of two sources and a destination. MACC takes the product of its operands and adds them to the contents of the MUACC register.

Example: MACC ACC0 ACC1

MDEC

The multiply-decumulate instruction is MDEC. It works similarly to the multiply and accumulate instruction, except it subtracts the product of its operands from the MUACC register instead of adding them.

Example: MDEC CC MUACCH

MULT

MULT multiplies its operand together and overwrites the value in the MUACC register with the resultant product. It also has an implicit destination like the previous instructions.

Example: MULT R7 R8

ADD

ADD performs arithmetic addition. It computes the sum of its operands and stores the result in the destination register.

Example: ADD R9 R10 R11

SUB

Arithmetic difference is computed by SUB. It writes the value of subtracting its second operand from its first into the destination register. Integers are represented in two's complement notation so negative values are allowed.

Example: SUB R12 R13 R14

ADDC

ADDC is similar to ADD above, except that one of its arguments is an eight bit two's complement integer constant. This constant is sign extended to twelve bits when the actual addition is performed.

Example: ADDC R15 R15 17
SUBFC

SUBFC (SUBtract From Constant) computes a difference like SUB above except that one of its arguments is an eight bit constant. Also, the order of the operands in the computation is reversed (see Table A.5) in that the value from the register is subtracted from the sign extended version of the eight bit constant.

Example: SUBFC R7 R9 -1

In the example above, the contents of register R9 would be subtracted from -1 and stored into register R7.

A.3.4 Arithmetic Macros

<table>
<thead>
<tr>
<th>Arithmetic Macros</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>NEG</td>
<td>SUBFC</td>
<td>[R1]</td>
<td>[R2]</td>
<td>0</td>
<td></td>
<td>arithmetic negation</td>
</tr>
<tr>
<td>COPY</td>
<td>ADDC</td>
<td>[R1]</td>
<td>[R2]</td>
<td>0</td>
<td></td>
<td>register to register copy</td>
</tr>
<tr>
<td>INC</td>
<td>ADDC</td>
<td>[R1]</td>
<td>[R1]</td>
<td>1</td>
<td></td>
<td>increment</td>
</tr>
<tr>
<td>DEC</td>
<td>ADDC</td>
<td>[R1]</td>
<td>[R1]</td>
<td>-1</td>
<td></td>
<td>decrement</td>
</tr>
<tr>
<td>SUBC</td>
<td>ADDC</td>
<td>[R1]</td>
<td>[R2]</td>
<td>-[const]</td>
<td></td>
<td>subtract constant, negation by assembler</td>
</tr>
<tr>
<td>ZERO</td>
<td>SUB</td>
<td>[R1]</td>
<td>[R1]</td>
<td>[R1]</td>
<td></td>
<td>zero a register</td>
</tr>
</tbody>
</table>

Table A.8
Arithmetic macros and implementations.

Table A.8 shows the definitions of various arithmetic macros that may be used exactly like real instructions in an assembly program. The assembler handles the expansion and any other preprocessing before the final translation into binary.

NEG

NEG computes the additive inverse in two's complement notation of the contents of register R2 and stores the results in register R1.

Example: NEG R7 R8

COPY

COPY copies the contents of register R2 and stores them into register R1.

Example: COPY R9 R10
INC

Incrementing macro INC adds 1 to the contents of register R1 and stores the result in the same place, overwriting its previous value.

Example: INC R11

DEC

Decrementing instruction macro DEC works just like INC above, except that the contents of R1 are replaced with the sum of R1 and -1.

Example: DEC R12

SUBC

SUBC subtracts an explicit constant from the contents of R2 and stores the value into destination register R1. Thus, the order of its operation is the same as the SUB instruction above and the reverse of the SUBFC instruction.

Example: SUBC R9 R10 16

ZERO

ZERO zeroes out a register by subtracting its contents from itself.

Example: ZERO R6

A.3.5 Relational Macros

<table>
<thead>
<tr>
<th>Relational Macros</th>
<th>GT</th>
<th>LT</th>
<th>EQUAL</th>
<th>LTC</th>
<th>GTC</th>
<th>EQC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SUB CC [R2] [R1]</td>
<td>SUB CC [R1] [R2]</td>
<td>SUB CC [R1] [R2]</td>
<td>ADDC CC [R1] [-const]</td>
<td>SUBFC CC [R1] [const]</td>
<td>SUBFC CC [R1] [const]</td>
</tr>
<tr>
<td></td>
<td>greater than</td>
<td>less than</td>
<td>arithmetic equivalence</td>
<td>less than constant, constant negated by assembler</td>
<td>greater than constant</td>
<td>equal to constant</td>
</tr>
</tbody>
</table>

Table A.9

Relational macros and implementations.

Table A.9 shows the definitions of various relational operator macros that may be also used exactly like real instructions in an assembly program. These instructions are implemented as macros to conserve opcodes in the instruction set. All relational instructions
implicitly write the condition code register using the same data from implicit writes in
typical arithmetic and logical operations even though the macro expansion explicitly uses
the CC register specifier as the destination. This implicit write is implemented within the
instruction decoding ROM of the microcontroller.

GT

GT (Greater Than) determines whether the value in register R1 is greater than in register
R2 and stores the result bit implicitly in the CC register field SIGN12.

Example: GT R7 R8

In the example above, the GT instruction sets the SIGN12 bit in the CC register true if
the value in R7 is greater than the value in R8. This is done by subtracting the value in R7
from the value in R8. If this difference is negative, the sign bit of the result will be 1 and
the relation will be true.

LT

LT (Less Than) sets the SIGN12 field of the CC register true if the value in register R1 is
less than the value in register R2.

Example: LT R9 R10

EQUAL

EQUAL sets the WNOR12 field of the CC register true if the value in register R1 is equal
to the value in register R2.

Example: EQUAL R11 R8

LTC

LTC (Less Than Constant) compares the value in register R1 to an explicit constant. It sets
the result bit in the SIGN12 field of the CC register.

Example: LTC R12 9

GTC

GTC (Greater Than Constant) is the complement of LTC above: it compares the values of
register R1 and an explicit constant and stores the result bit into the SIGN12 field of the CC
register.

Example: GTC R9 25
EQLC

Comparing equality with a constant is the function of the EQLC macro. If register R1's value is equal to the explicit constant, then the WNOR12 field of the CC register is set to true.

Example: EQLC R6 -33

A.3.6 Logical and Shift Instructions

<table>
<thead>
<tr>
<th>Logic and Shift Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mnemonic</td>
</tr>
<tr>
<td>----------</td>
</tr>
<tr>
<td>NOT</td>
</tr>
<tr>
<td>NAND</td>
</tr>
<tr>
<td>AND</td>
</tr>
<tr>
<td>NOR</td>
</tr>
<tr>
<td>OR</td>
</tr>
<tr>
<td>XOR</td>
</tr>
<tr>
<td>EQ</td>
</tr>
<tr>
<td>LRSHFT</td>
</tr>
<tr>
<td>LLUSHFT</td>
</tr>
<tr>
<td>ARSHIFT</td>
</tr>
<tr>
<td>LCIRC</td>
</tr>
<tr>
<td>RCIRC</td>
</tr>
</tbody>
</table>

Table A.10
Logical and shift instructions and opcodes.

In addition to the arithmetic instructions and macros supported above, the microcontroller architecture also implements a set of logical and shift operations. These are summarized in Table A.10 and described in detail in the following section.

NOT

The bitwise negation instruction NOT inverts the bits of the contents of its argument register and stores the result into the destination register.

Example: NOT R7 R8

NAND

NAND performs a bitwise NAND operation between the values stored in its operand registers and writes the result into the destination.

Example: NAND R6 R10 R11
AND

AND computes the bitwise AND of the values in its argument registers. The results are written to the destination. AND is the bitwise inverse of NAND above.

Example: AND R6 R10 R11

NOR

NOR computes the bitwise NOR of the arguments and writes the result to the destination.

Example: NOR da_acc r8 r9

OR

The bitwise OR of the operands is performed by the OR instruction, which is the bitwise inverse of NOR above. Results are written to the destination register.

Example: OR da_acc acc0 muacc1

XOR

XOR computes the bitwise XOR of the two operands and writes the result to the third register.

Example: XOR buptr cc accl

EQ

EQ is the bitwise inverse of XOR above and computes the bitwise equivalence of the two operands and writes the result to the destination.

Example: EQ r6 r7 r10

LRSHFT

LRSHFT (Logical Right SHIFT) takes a single operand and bit shifts it to the right, shifting in a 0 into the most significant bit. The result is written to the destination register.

Example: LRSHFT r11 r12

LLSHFT

LRSHFT (Logical Left SHIFT) takes a single operand and bit shifts it to the left, shifting in a 0 into the least significant bit. The result is written to the destination register.

Example: LLSHFT r13 r14
A.3. INSTRUCTION DESCRIPTIONS

ARSHFT

ARSHFT (Arithmetic Right SHIFT) takes a single operand and bit shifts it to the right, shifting in a 0 into the most significant bit if the previous MSB is 0, and a 1 if the MSB was previously 1. This preserves the sign of the binary value if it is interpreted as a two’s complement number. The result is written to the destination register. Note that there is no complementary instruction since the arithmetic left shift is equivalent to the logical left shift.

Example: ARSHFT r15 r15

LCIRC

LCIRC (Left CIRCular shift) takes the value stored in its operand register, shifts it to the left one bit, and copies the previous most significant bit into the new least significant bit position. The result is written to the destination register.

Example: LCIRC r7 da.acc

RCIRC

RCIRC (Right CIRCular shift) is the complement of the LCIRC instruction above. It takes the value stored in its operand register, shifts it to the right one bit, and copies the previous least significant bit into the new most significant bit position. The result is written to the destination register.

Example: RCIRC r10 muaccch

A.3.7 Control Flow Instructions

<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Opcode</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>JUMP</td>
<td>10100</td>
<td>jump</td>
</tr>
<tr>
<td>CTJUMP</td>
<td>10101</td>
<td>conditional jump true</td>
</tr>
<tr>
<td>CFJUMP</td>
<td>10110</td>
<td>conditional jump false</td>
</tr>
</tbody>
</table>

Table A.11
Control flow instructions and opcodes.

Table A.11 lists the control flow instructions available to the microcontroller programmer. These consist of one unconditional and two conditional jumps. Because of the limited instruction set size and instruction memory on-chip, there are no provisions for subroutines.
JUMP

JUMP is the unconditional jump instruction. It loads the program counter (PC) register with its argument, an explicit address or an address label.

First Example: JUMP 0x1AF

In the first example, the target for the jump is the hexadecimal address 0x1AF.

Second Example: JUMP LOOP1: ; jump forward

The second example shows the use of an address label. The assembler makes two passes over the source code and resolves labels like LOOP1: into absolute addresses like in the first example. The program example in Section A.4 shows an example of the use of address labels in a loop structure.

CTJUMP

CTJUMP (Conditional True JUMP) replaces the program counter with its address argument if the boolean argument is true. The boolean argument is the bit of the condition code register specified with one of the condition code field specifiers listed in Table A.3.

Example: CTJUMP CC[0] LOOP0:

The example will write the program counter with the address referred to by the address label LOOP0: if the value of the CC[0] field of the condition code register is true.

CFJUMP

CFJUMP (Conditional False JUMP) is the complement of CTJUMP above. It replaces the program counter with its address argument if the boolean argument is false. The boolean argument is the bit of the condition code register specified with one of the condition code field specifiers listed in Table A.3.

Example: CFJUMP sign24 0x232

The example will write the program counter with the address referred to by the address label 0x232 if the value of the sign24 field of the condition code register is 0 (false).

A.3.8 Memory Instructions

Table A.12 shows the memory interface instructions of the microcontroller. The Sensor DSP microcontroller is a load/store architecture, so memory accesses use only the register file as a destination. It can use either the register file (for storing variables) or the instruction stream (for storing constants) as a source. Constants can only be written to the register file. The instruction set supports both direct and indirect operations. Only the data memory is visible to the processor; the instruction memory can only be loaded through the JTAG programming interface. Consequently, self-modifying code is not allowed.
### Memory Instructions

<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Opcode</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOAD</td>
<td>11000</td>
<td>load register from memory</td>
</tr>
<tr>
<td>STOR</td>
<td>11001</td>
<td>store register to memory</td>
</tr>
<tr>
<td>LDI</td>
<td>11010</td>
<td>indirect load register from memory</td>
</tr>
<tr>
<td>STI</td>
<td>11011</td>
<td>indirect store register to memory</td>
</tr>
<tr>
<td>LDC</td>
<td>11100</td>
<td>load 12 bit constant to register</td>
</tr>
</tbody>
</table>

**Table A.12**

Memory instructions and opcodes.

**LOAD**

**LOAD** is the direct load from memory. It retrieves the data stored in the data memory (or filter buffer, since the processor sees a flat address space) and stores it in the destination register.

Example: LOAD R7 0x1FF

The example loads the value at memory location 0x1FF and stores it in the register R7.

**STOR**

The direct memory write instruction is **STOR**. This operation takes the data stored in the source register and writes it to the memory location specified as a target.

Example: STOR R11 0x9AC

The data stored in register R11 will be written to data memory location 0x9AC in the above example.

**LDI**

**LDI** is the indirect load instruction. It takes two register specifiers as arguments and reads the contents of the memory location addressed by the operand register value and writes this value into the destination register.

Example: LDI R12 R13

In the example, the data memory location pointed to by the value of register R13 is written to the destination register R12.

**STI**

**STI** is the indirect store instruction; it is the complement to **LDI** above. It takes the contents of the operand register and writes the value to the memory location addressed by the contents of the destination register.
Example: `STI R6 R15`

The above example reads the contents of register R6 and writes the value to the data memory location pointed to by the value stored in register R15. Note that this convention is the reverse of the argument convention for `LDI`.

**LDC**

**LDC (LOAD Constant)** is the register constant load instruction. It takes the explicit 12 bit integer constant specified as an argument and writes it to the register location also specified in its arguments.

Example: `LDC R9 -3`

The example writes the constant -3 (in two's complement representation) into register R9.

**A.3.9 Filter Interface Instructions**

<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Opcode</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG</td>
<td>11110</td>
<td>set configuration state bits</td>
</tr>
</tbody>
</table>

Table A.13

Filter interface instructions and opcodes.

<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Code</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR</td>
<td>00</td>
<td>set bit to 0</td>
</tr>
<tr>
<td>SET</td>
<td>01</td>
<td>set bit to 1</td>
</tr>
<tr>
<td>FLIP</td>
<td>10</td>
<td>invert bit</td>
</tr>
<tr>
<td>HOLD</td>
<td>11</td>
<td>maintain state (basically a NOP)</td>
</tr>
</tbody>
</table>

Table A.14

Configuration state specifiers and codes.

The front-end filtering units of the Sensor DSP chip are under software control of the microcontroller using the configuration instructions and state bit specifiers shown in Tables A.13 and A.14 respectively.

**CONFIG**

`CONFIG` is the sole configuration instruction available to the microcontroller. In conjunction with the state specifiers shown in Table A.14 and the configuration bit specifiers shown
in Table A.4 it configures the operation of the entire processor chip. The state of each configuration bit can be modified using the state specifiers shown in Table A.14. An example will clarify the operation:

Example: CONFIG DA_ENABLE CLR

The example shows that the configuration bit DA_ENABLE will be cleared (set to 0) by the instruction. The bit could also be set to 1, inverted, or held in the same state (a null operation).

### A.3.10 Unused Opcodes

<table>
<thead>
<tr>
<th>Unused Opcodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opcode</td>
</tr>
<tr>
<td>1011</td>
</tr>
<tr>
<td>11101</td>
</tr>
<tr>
<td>11111</td>
</tr>
</tbody>
</table>

**Table A.15**

Unused opcodes.

Table A.15 lists the opcodes that are unused in the ISA specification. These codes are not output by the assembler and should not be used if the processor is programmed directly in binary. They actually map to the **NOP** instruction in the opcode decoder.

### A.3.11 Miscellaneous Reserved Words

There are three other strings that may be used in the assembly language program and which are similar to preprocessor commands in C. The first is the **.DEF** construct which is used to alias constants. Any constant in decimal or hexadecimal notation can be referred to by an alphanumeric string. The mapping is done using the **.DEF** construct.

Example: **.DEF FOO 0x03E**

In the example, **FOO** will be replaced by the binary equivalent of **0x03E** in the assembler output. This works for positive and negative decimal constants, where the binary representation is 12 bit two's complement.

Labels are specified with colons and are mapped to instruction addresses by the assembler during its first pass. During the second pass, the labels are resolved to the instruction addresses they refer to.

Example: **LOOP1: ZERO R8**

The example labels the address of the instruction **ZERO R8** with the label **LOOP1: ..**. Control flow instructions upstream or downstream in the code can then refer to the label as a target for branches.

Finally, comments are specified using the semicolon **;**. Any characters appearing to the right of a semicolon are ignored by the assembler until a newline is reached.
A.4 Example Program: Fibonacci Numbers

Table A.16 is a summary listing of an example program which computes the first few numbers of the Fibonacci sequence. It begins by using the .DEF construct to alias the address of an array in data memory and the number of Fibonacci numbers to compute. Several constants are loaded to initialize the first few numbers of the series. A loop then computes the numbers in the series. A second loop reads these back in reverse order. Then the program terminates. Note the end of program character $ to force the assembler to stop. This character is necessary for all programs.
Appendix B

Sensor DSP Nonlinear/Short Linear Filter Assembly Language

Nonlinear and short linear filters are implemented using the VLIW architecture shown in Figure 6.5 and repeated in Figure B.1. Each functional unit, the Square-Accumulate Unit (SAC), the Multiply-Accumulate Unit (MAC), and the Load/Store Unit (LSU) implements its own small instruction set. To synchronize with the operation of the Distributed Arithmetic Unit, the total number of instruction slots for each of these units is only 8. This processor is a load/store architecture like the microcontroller unit, so the arithmetic operations use only registers as sources and destinations. The only memory interaction is with the filter buffer which acts as a delay line and implements relative addressing. This means that any address specified to the LSU is really an offset from the current buffer pointer backward in time to an earlier output sample.

![Diagram of VLIW architecture](image)

**Figure B.1:** Short linear and nonlinear filter implementation architecture.

Since the NLSL Unit simply does filtering operations, there is no notion of control flow.
in the instruction set architecture. All instructions are executed through every iteration of
the program. The program is repeated every eight clock cycles.

B.1 Registers

A total of 16 registers are available to the NLSL program, of which three are special regis-
ters which access the current results in the accumulators of the different functional units.
All values are 12 bit two's complement integers. Intermediate values are 24 bits, but these
are truncated to 12 bits by specifying some configuration bits through the JTAG interface.

<table>
<thead>
<tr>
<th>Register Specifiers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Special Registers</td>
</tr>
<tr>
<td>Mnemonic</td>
</tr>
<tr>
<td>DA_ACC</td>
</tr>
<tr>
<td>ACC0</td>
</tr>
<tr>
<td>ACC1</td>
</tr>
<tr>
<td>Other Registers</td>
</tr>
<tr>
<td>R3-R15</td>
</tr>
</tbody>
</table>

Table B.1
NLSL Unit registers.

Table B.1 lists the register specifiers and codes for the registers available to the NLSL
program. The DA_ACC register is loaded every clock cycle from the output of the dis-
tributed arithmetic unit. ACC0 and ACC1 are also loaded every clock cycle with the new
truncated values from the SAC and MAC functional unit accumulators, respectively. The
other registers are general purpose.

B.2 Instruction Descriptions

This section describes in detail the instruction sets available to each of the functional units.
Each instruction takes one clock cycle to execute. None of the units are pipelined so there
are no pipeline hazards or bubbles. All writes are implicitly to the functional unit’s accum-
ulator except for loads and stores which require a destination register specifier.

B.2.1 SAC Instructions

Table B.2 summarizes the SAC Unit instructions, each of which will be described in detail
below. All SAC unit arithmetic instructions take only one register specifier, the source
register for the operand.

Example: SAC SQAC DA_ACC

In the example, the register contents of DA_ACC are squared and added to the accu-
mulator value. All arithmetic operations have the same syntax as the example.
### SAC Unit Instructions

<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Opcode</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAC_NOP</td>
<td>000</td>
<td>null operation</td>
</tr>
<tr>
<td>SAC_SQAC</td>
<td>001</td>
<td>square-accumulate</td>
</tr>
<tr>
<td>SAC_SQRE</td>
<td>010</td>
<td>square</td>
</tr>
<tr>
<td>SAC_SQSU</td>
<td>011</td>
<td>square-subtract from accumulator</td>
</tr>
<tr>
<td>SAC_ADD</td>
<td>100</td>
<td>add to accumulator</td>
</tr>
<tr>
<td>SAC_SUB</td>
<td>101</td>
<td>subtract from accumulator</td>
</tr>
<tr>
<td></td>
<td>110</td>
<td>unused</td>
</tr>
<tr>
<td></td>
<td>111</td>
<td>unused</td>
</tr>
</tbody>
</table>

**Table B.2**

SAC Unit instructions and opcodes.

**SAC_NOP**

SAC_NOP is the null operation. No writes occur to the SAC unit accumulator for this instruction.

**SAC_SQAC**

SAC_SQAC is the square-accumulate instruction. It adds the square of the contents of its argument to the SAC unit accumulator, leaving the result in the accumulator.

**SAC_SQRE**

SAC_SQRE is the square instruction. It reads the value of its argument register, squares it, and overwrites the contents of the accumulator with the result.

**SAC_SQSU**

The square-subtract instruction is SAC_SQSU. This operation subtracts the square of its operand from the current accumulator value and writes the result to the accumulator.

**SAC_ADD**

SAC_ADD simply adds the value of its operand to the contents of the accumulator. The result is written to the accumulator.

**SAC_SUB**

SAC_SUB is the complement of the SAC_ADD instruction above. It subtracts the value of its operand from the accumulator value, leaving the new result in the accumulator.
B.2.2 MAC Instructions

<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Opcode</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAC.NOP</td>
<td>000</td>
<td>null operation</td>
</tr>
<tr>
<td>MAC.MACCC</td>
<td>001</td>
<td>multiply-accumulate</td>
</tr>
<tr>
<td>MAC.MDEC</td>
<td>010</td>
<td>multiply-subtract from accumulator</td>
</tr>
<tr>
<td>MAC.MULT</td>
<td>011</td>
<td>multiply</td>
</tr>
<tr>
<td>MAC.ADD</td>
<td>100</td>
<td>add</td>
</tr>
<tr>
<td>MAC.SUB</td>
<td>101</td>
<td>subtract</td>
</tr>
<tr>
<td>MAC.ACC</td>
<td>110</td>
<td>add to accumulator</td>
</tr>
<tr>
<td>MAC.DEC</td>
<td>111</td>
<td>subtract from accumulator</td>
</tr>
</tbody>
</table>

Table B.3
MAC Unit instructions and opcodes.

Table B.3 lists the instructions available to the MAC unit program. These instructions require either one or two register specifiers for operands and implicitly write their results to the MAC unit accumulator.

Example: MAC.MULT ACC1 R3

In the example, the contents of registers ACC1 and R3 are multiplied together and their product is written to the MAC unit accumulator. All MAC arithmetic instructions share the same syntax.

MAC.NOP

MAC.NOP is the null instruction. No writes are performed to the MAC accumulator when this instruction is executed.

MAC.MACC

MAC.MACC is the multiply-accumulate instruction. It computes the product of the values stored in its operand registers and adds the result to the accumulator value. The new result is written to the accumulator.

MAC.MDEC

MAC.MDEC is the complement of the MAC.MDEC instruction above. It subtracts the product of the values stored in its operands from the accumulator value, leaving the result of the subtraction in the accumulator.
MAC.MULT

The multiplication instruction is MAC.MULT. It simply computes the product of its argument values and overwrites the accumulator result with the product.

MAC.ADD

MAC.ADD is the addition instruction. It sums the values of its operand registers and replaces the accumulator value with the value of the sum.

MAC.SUB

MAC.SUB is the complementary subtraction instruction to the MAC.ADD instruction above. It computes the difference between its first and second operands. The result is written to the accumulator.

MAC.ACC

MAC.ACC is the accumulate instruction. It requires only one argument and adds the value in its operand register to the accumulator contents, leaving the result in the accumulator.

Example: MAC.ACC R7

In the example above, the contents of register R7 are added to the current value of the MAC unit accumulator.

MAC.DEC

MAC.DEC subtracts the value stored in its lone argument register from the value stored in the accumulator. The result is written to the accumulator. The instruction syntax for this operation is the same as in the example above for MAC.ACC.

B.2.3 LSU Instructions

Table B.4 lists the instructions available to the LSU Unit. These instructions are responsible for loading data to the registers from the instruction stream (loading constants) or previous filter outputs stored in the filter buffer. Note that there is no store instruction as stores are done implicitly to the filter buffer every eight cycles. Each LSU instruction requires two arguments: a destination register and a source register, constant, or memory address.

Example: LSU_LOAD R3 0x0100000010 ; load R3 with ACC0[0:2]
LSU Unit Instructions

<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Opcode</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSU_NOP</td>
<td>000</td>
<td>null operation</td>
</tr>
<tr>
<td>LSU_LDC</td>
<td>001</td>
<td>load constant to register</td>
</tr>
<tr>
<td>LSU_LOAD</td>
<td>010</td>
<td>load direct from memory to register</td>
</tr>
<tr>
<td>LSU_LDI</td>
<td>011</td>
<td>load indirect from memory to register</td>
</tr>
<tr>
<td>LSU_COPY</td>
<td>100</td>
<td>copy from one register to another</td>
</tr>
<tr>
<td></td>
<td>101</td>
<td>unused</td>
</tr>
<tr>
<td></td>
<td>110</td>
<td>unused</td>
</tr>
<tr>
<td></td>
<td>111</td>
<td>unused</td>
</tr>
</tbody>
</table>

Table B.4
LSU Unit instructions and opcodes.

In the example, register R3 is loaded with the data value from the filter buffer corresponding to the output of the ACC0 accumulator at time $k - 2$, where $k$ (the current time) corresponds to the current value of the filter buffer pointer. There are three specifiers corresponding to the buffers after each of the filtering units. These specifiers precede the 7 bit address corresponding to the delay from the current sample: $0 \times 00$ corresponds to results from the distributed arithmetic unit, the DA_ACC register; $0 \times 01$ corresponds to the results from the SAC unit accumulator ACC0; finally, $0 \times 10$ corresponds to results from the MAC unit accumulator ACC1.

LSU_NOP

**LSU_NOP** is the null operation for the LSU unit. No results are written to any of the registers for this instruction.

LSU_LDC

**LSU_LDC** is the load constant instruction. It takes a constant specified in the instruction stream and writes it to the register specified as its destination.

LSU_LOAD

**LSU_LOAD** is the direct memory load instruction. It accesses the filter buffer value corresponding to the appropriate filter unit result and time offset into the previous samples and writes this value into the destination register.

LSU_LDI

**LSU_LDI** is the indirect memory load instruction. Instead of using an explicit offset into the filter buffer, it uses the value stored in its second register argument as the filter buffer
address. It takes the value stored in the filter buffer and writes it into the destination register.

**LSU.COPY**

**LSU.COPY** is the register to register copy instruction. It takes two register specifiers as arguments and copies the value from the source register to the destination register.
Appendix C

Chip Pinouts
Figure C.1: DC/DC Converter Chip Pinout
<table>
<thead>
<tr>
<th>Pin</th>
<th>Type</th>
<th>Signal Name</th>
<th>Pin</th>
<th>Type</th>
<th>Signal Name</th>
<th>Pin</th>
<th>Type</th>
<th>Signal Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Output</td>
<td>PRE</td>
<td>26</td>
<td>Supply</td>
<td>GND</td>
<td>51</td>
<td>Supply</td>
<td>VHH</td>
</tr>
<tr>
<td>2</td>
<td>Output</td>
<td>WORD.EN</td>
<td>27</td>
<td>Supply</td>
<td>VHH</td>
<td>52</td>
<td>Supply</td>
<td>GND</td>
</tr>
<tr>
<td>3</td>
<td>Output</td>
<td>DLATCH</td>
<td>28</td>
<td>Input</td>
<td>DATA.VAL.SET</td>
<td>53</td>
<td>Input</td>
<td>YTEST_SEL(0)</td>
</tr>
<tr>
<td>4</td>
<td>Output</td>
<td>BUF.WEN</td>
<td>29</td>
<td>Input</td>
<td>MUCTRL.EN</td>
<td>54</td>
<td>Input</td>
<td>YTEST_SEL(1)</td>
</tr>
<tr>
<td>5</td>
<td>Supply</td>
<td>VHH</td>
<td>30</td>
<td>Supply</td>
<td>GND</td>
<td>55</td>
<td>Input</td>
<td>YTEST_SEL(2)</td>
</tr>
<tr>
<td>6</td>
<td>Input</td>
<td>Xin</td>
<td>31</td>
<td>Supply</td>
<td>VDD</td>
<td>56</td>
<td>Output</td>
<td>YTEST(0)</td>
</tr>
<tr>
<td>7</td>
<td>Supply</td>
<td>GND</td>
<td>32</td>
<td>Supply</td>
<td>GND</td>
<td>57</td>
<td>Output</td>
<td>YTEST(1)</td>
</tr>
<tr>
<td>8</td>
<td>Supply</td>
<td>VDD</td>
<td>33</td>
<td>Supply</td>
<td>VDD</td>
<td>58</td>
<td>Output</td>
<td>YTEST(2)</td>
</tr>
<tr>
<td>9</td>
<td>Supply</td>
<td>GND</td>
<td>34</td>
<td>Input</td>
<td>RST</td>
<td>59</td>
<td>Output</td>
<td>YTEST(3)</td>
</tr>
<tr>
<td>10</td>
<td>Input</td>
<td>TDI</td>
<td>35</td>
<td>Supply</td>
<td>VHH</td>
<td>60</td>
<td>Output</td>
<td>YTEST(4)</td>
</tr>
<tr>
<td>11</td>
<td>Input</td>
<td>TMS</td>
<td>36</td>
<td>Supply</td>
<td>GND</td>
<td>61</td>
<td>Output</td>
<td>YTEST(5)</td>
</tr>
<tr>
<td>12</td>
<td>Input</td>
<td>TCK</td>
<td>37</td>
<td>Output</td>
<td>FAST.MODE</td>
<td>62</td>
<td>Output</td>
<td>YTEST(6)</td>
</tr>
<tr>
<td>13</td>
<td>Input</td>
<td>TRST</td>
<td>38</td>
<td>Output</td>
<td>RD.TRIG.OUT</td>
<td>63</td>
<td>Output</td>
<td>YTEST(7)</td>
</tr>
<tr>
<td>14</td>
<td>Output</td>
<td>TDO</td>
<td>39</td>
<td>Output</td>
<td>CLK.OUT</td>
<td>64</td>
<td>Output</td>
<td>YTEST(8)</td>
</tr>
<tr>
<td>15</td>
<td>Output</td>
<td>Xout</td>
<td>40</td>
<td>Input</td>
<td>RD.TRIG</td>
<td>65</td>
<td>Output</td>
<td>YTEST(9)</td>
</tr>
<tr>
<td>16</td>
<td>Supply</td>
<td>GND</td>
<td>41</td>
<td>Input</td>
<td>CLKIN</td>
<td>66</td>
<td>Output</td>
<td>YTEST(10)</td>
</tr>
<tr>
<td>17</td>
<td>Supply</td>
<td>VHH</td>
<td>42</td>
<td>Input</td>
<td>CLK_RST</td>
<td>67</td>
<td>Output</td>
<td>YTEST(11)</td>
</tr>
<tr>
<td>18</td>
<td>Supply</td>
<td>GND</td>
<td>43</td>
<td>Input</td>
<td>FHRef</td>
<td>68</td>
<td>Supply</td>
<td>VHH</td>
</tr>
<tr>
<td>19</td>
<td>Supply</td>
<td>VHH</td>
<td>44</td>
<td>Supply</td>
<td>GND</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>Supply</td>
<td>GND</td>
<td>45</td>
<td>Supply</td>
<td>VDD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>Supply</td>
<td>VDD</td>
<td>46</td>
<td>Input</td>
<td>CLK.CONF(0)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>Input</td>
<td>GLOBAL.EN</td>
<td>47</td>
<td>Input</td>
<td>CLK.CONF(1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>Supply</td>
<td>GND</td>
<td>48</td>
<td>Input</td>
<td>CLK.CONF(2)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>Supply</td>
<td>VDD</td>
<td>49</td>
<td>Input</td>
<td>CLK.CONF(3)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>Supply</td>
<td>GND</td>
<td>50</td>
<td>Supply</td>
<td>GND</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table C.1**
Sensor DSP Chip Pinout
Figure C.2: Sensor DSP Chip Package Footprint
Appendix D

Moving-Coil Generator Design Notes

This Appendix is a set of design guidelines for constructing electromechanical generators using the moving coil principle discussed in Chapter 3.

D.1 Basic Mechanical Design

The basic principle of any moving-coil transducer is that mechanical motion of a coil of wire through a magnetic field induces a voltage or current on the coil. Typical examples are loudspeakers and some older types of microphones.

Figure 3.1 shows a sketch of the generator. It consists of a mass-spring system with a wire coil attached to the mass. The spring is an helical extension spring anchored at one end to the generator housing, leaving the mass and coil free to move vertically. At the bottom of the housing is fixed a permanent magnet which is a thick disk with a circular gap machined in it. The magnetic field extends radially from the center pole piece to the ring surrounding it. At rest, the coil extends into this gap and a certain amount of magnetic flux is cut by the spaces between the coil windings. Note that this is different than a typical coil inductor where the magnetic field extends through the center of the coil (the \( \hat{z} \) direction in a cylindrical coordinate system). In this case, the field extends from inside the coil to outside the coil along the \( \hat{r} \) direction. As the mass moves up and down, the coil moves up and down into and out of the gap, changing the flux linked by the coil and thus producing a time-varying electrical signal on the coil.

The mechanical design is fairly unconstrained, except for the permanent magnet. Since magnet materials tend to be brittle and difficult to machine and the appropriate radially directed field has to be created in this case, I used the permanent magnet from a commercially available Panasonic miniature speaker. This component has a diameter of 20 mm, which sets the minimum size of the generator housing. Figure 1 in the attachment shows the magnet dimensions. The prototype generator I constructed is about 7.5 cm in total height and that general size is convenient to work with.
The attached Figure 2 shows the cross-section of the actual generator design. Mounted on top of the magnet's center pole is a long column used as a guide post. Large displacements of the coil occur in typical generator operation so it is important that the coil motion be constrained to be linear. The coil is attached to a flange at the end of a sleeve that fits around the guide post but with a small space between them so that it is free to move with little friction. A hook at the top of the sleeve can be connected to a hook at one end of the extension spring. The other spring hook is connected to the fixed anchor. Not shown is the housing that fixes the distance between the anchor and the permanent magnet. A mass that can slide over the sleeve and is supported by the sleeve flange determines the natural frequency of the generator.

Ideally, one would like to have maximum flexibility in the generator design to perform a number of experiments and determine the optimal mass, spring rate, travel distance for the mass and coil, and coil parameters for power generation. By using an eyelet in the anchor and a hook on the sleeve it is easy to change springs. Having a set of masses that can fit onto the sleeve gives flexibility in choosing the mass. Set screws can be used to affix the anchor to the housing, perhaps fitting into long slots cut into the housing itself, and thus the anchor-magnet distance can be modified. Changing the coil itself is tricky - the prototype merely epoxied the coil onto the sleeve flange. However, since it is well known how changing the magnet and coil parameters will affect the power generation it is really flexibility in the mechanical parameters that is of interest.

The following wish list summarizes the desired adjustability of the generator:

- Adjustable anchor position. It would be nice to have full control over the anchor position, although a series of discrete positions say at 2-3 mm intervals is acceptable.

- Interchangeable springs. This should not be a problem since if the anchor height can be modified the generator can accommodate springs of various lengths.

- Adjustable mass or a set of interchangeable masses. For example, a set of small 0.5 g masses that can be added or taken off the sleeve (like barbell weights).

D.2 Desired Generator Dynamics

The generator tries to couple mechanical energy in the low frequency vibration range from about 1 Hz to 1 kHz. For optimal energy transfer between vibrations of the housing to vibrations of the mass, the resonant frequency of the generator spring-mass system should also lie in this frequency range. Since this frequency varies as the square root of the spring rate and inversely with the square root of the mass, this is a wide range to cover with a set of springs and masses. However, the range of most interest is the 10 - 100 Hz range and for masses between 1 - 20 g, it is easy to find commercially available stock springs with rates between 0.1 - 100 lbs./in. to yield the right frequencies. The key issue with these springs is that their length varies, so having an adjustable position for the anchor is crucial.
Table D.1
Generator specifications.

<table>
<thead>
<tr>
<th>Approximate dimensions:</th>
<th>base height</th>
<th>4 cm x 4 cm&lt;br&gt;4 - 10 cm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spring:</td>
<td>rate</td>
<td>0.02 - 50 lbs./in.</td>
</tr>
<tr>
<td></td>
<td>initial tension</td>
<td>0.002 - 0.05 lbs.</td>
</tr>
<tr>
<td></td>
<td>free length</td>
<td>1/4 - 3 in.</td>
</tr>
<tr>
<td>Mass:</td>
<td>minimum(^1)</td>
<td>~ 0.50 g</td>
</tr>
<tr>
<td></td>
<td>adjustable range</td>
<td>0.50 - 20 g</td>
</tr>
<tr>
<td></td>
<td>total generator mass(^2)</td>
<td>&lt; 75 g</td>
</tr>
<tr>
<td>Travel:</td>
<td>from equilibrium</td>
<td>up to ±1 cm</td>
</tr>
<tr>
<td>Magnet:</td>
<td>total dimensions</td>
<td>20 mm x 8.8 mm</td>
</tr>
<tr>
<td></td>
<td>gap width</td>
<td>0.76 mm</td>
</tr>
<tr>
<td>Coil:</td>
<td>dimensions</td>
<td>5.46 mm x 2.84 mm</td>
</tr>
<tr>
<td></td>
<td>turns</td>
<td>~ 200</td>
</tr>
<tr>
<td></td>
<td>wire gauge</td>
<td>40 or higher</td>
</tr>
</tbody>
</table>

For optimal power transfer it is also important to have low mechanical damping. It is not possible to do anything about the air resistance, however in the prototype I constructed I noticed that occasionally the sleeve tended to rub against the guide post for more severe losses. More importantly, since there was (relatively) a lot of space between the sleeve and the post some of the initial energy of the mass went into higher order modes of the system (flexural modes of the sleeve/flange and mass motion in the x-y plane instead of strictly perpendicular to it). Without careful control of the initial condition, the sleeve and coil tended to rattle around the post and the magnet air gap, producing all kinds of non-reproducible high-frequency voltage waveforms. For a first cut, it is reproducibility of the generator operation that is most important, and should be traded off for generator efficiency if necessary. I would rather have more damping and a more reproducible waveform than the highest power out for right now. Having a good test bed that allows performance of careful, repeatable experiments will allow us to construct a good model of the system. From the model we can exploit ways to improve the efficiency of the power transfer.

The overall design specifications are shown in Table D.1. If necessary, we can obtain the springs and masses from other sources as well as provide the magnet and coil. What I would most like is a well-constructed housing, adjustable anchor, and a good guide post and sleeve that really restricts the coil motion to the z direction only - even for large (~ 1 cm) deflections in the spring.

D.3 Other Design Notes

There are two important details about the generator that should be kept in mind. First, it is important that the anchor, post, symmetry axis of the spring, and center of mass of

\(^1\)Sleeve, hook, and coil mass only.

\(^2\)Housing, spring, sleeve, and coil assembly.
the sleeve and coil assembly all line up closely. This was a problem in the prototype and contributed to the unreproducibility of the voltage. Second, the coil wire is typically of very narrow gauge and consequently is very mechanically fragile. To make measurements of the coil voltage with an oscilloscope, thicker gauge wires must be connected to the coil. Direct attachment of these wires tends to break the ends of the coil wire. Also, the added mass affects the dynamics somewhat. The thick wires are somewhat stiff and as they extend away from the sleeve, they significantly change the moment of inertia of the sleeve assembly (especially for low masses). This contributes to unpredictable motion of the coil.

One solution for the fragile wire problem is to connect the narrow coil wire to conductive contacts on the flange. Then, thicker wire is also soldered to the contacts. However, this does tend to change the moment of inertia in undesirable ways. Another approach would be to connect narrow wire from the flange contacts to a fixed point on the housing. This wire would be long enough that no matter what the travel of the coil it would not be stressed to the point of breaking. Since these wires are of such low mass, they probably won’t affect the linear motion of the coil much. Thicker wires can then be attached to contacts on the housing.

D.4 Testability

For testability and accurate modeling of the generator, it is necessary to have a convenient means of putting initial conditions on the system and then monitoring the voltage output. For example, a good test is to displace the sleeve assembly from equilibrium and then let it go. As it oscillates the voltage can be captured on the scope. To do this accurately it is necessary to synchronize the scope trigger with the initial motion of the mass. One approach to solving this problem is to use a catch on the sleeve that can hold the spring under tension as the sleeve is displaced. A relay can then release this catch on an electrical signal that also triggers the scope. As long as the delay of the relay is well-characterized, we can obtain accurate responses this way. A purely electrical approach would be to source a current through the loop strong enough to displace the sleeve, and then turn this current off. However, I do not believe the coil wire can supports a large enough current to overcome the spring rates of interest. I would definitely like some mechanism which can set initial conditions on the spring-mass system and that can help synchronize the oscilloscope to the generator vibration.
THESIS PROCESSING SLIP

FIXED FIELD: ill __________ name __________ index __________ biblio __________

COPIES: Archives Aero Dewey Eng Hum
Lindgren Music Rotch Science

TITLE VARIES: □ __________

NAME VARIES: □ __________

IMPRINT: (COPYRIGHT) __________

COLLATION: __________

ADD: DEGREE: ________ DEPT.: ________

SUPERVISORS: ______________________
__________________________________
__________________________________
__________________________________
__________________________________

NOTES: ________________________________

__________________________________ cat'r: __________ date: __________

DEPT: E.E. □ J 174

YEAR: 1999 □ DEGREE: Ph.D.

NAME: AMIRTHARATAN, Rajeevan