VLSI Architectures for Digital Signal Processing on Energy-Constrained Systems-on-Chip

---

A Dissertation

Presented to

the Faculty of the School of Engineering and Applied Science

University of Virginia

In Partial Fulfillment

of the requirements for the Degree

Doctor of Philosophy (Electrical and Computer Engineering)

by

Alicia Klinefelter

August 2015

Abstract

The design of ultra-low power (ULP) integrated circuits for Systems-On-Chip (SoCs) requires consideration of both flexibility and robustness during low-energy operation. Due to their quadratic relationship, the strongest design knob for reducing energy on-chip is the supply voltage. By operating digital circuits in the subthreshold region, using a supply voltage that falls below the threshold of the device, designers can minimize energy per operation at the cost of a performance penalty. However, there are many applications with low throughput requirements that can benefit from these energy savings. For example, systems that process biomedical data such as ECG, EEG, and EMG require sampling and processing rates on the order of kHz, making subthreshold operation feasible. The result of these substantial energy improvements is an emerging application space for these ULP SoCs in batteryless sensor nodes capable of running on energy harvested from the surrounding environment alone.

This work presents two versions of a highly integrated, flexible SoC platform targeted for Internet-of-Things (IoT) applications. These SoCs support multiple sensing modalities, extract information from data flexibly across applications, harvest and deliver power efficiently, and communicate wirelessly. The first version of the chip acquired ECG data, extracted the heart-rate, and transmitted the raw signal operating off of harvested energy while consuming 19μW. The first revision of the chip was tested end-to-end for motion capture with an ADXL362Z accelerometer (over SPI) while powered from indoor solar by an integrated power management unit. The chip consumed 6.45μW while streaming raw motion data wirelessly over the UWB radio.

To enable ULP operation, low power techniques need to be utilized across the hierarchy from the system-level down to the low-level circuit design. When focusing on digital design and processing, energy reductions can be achieved by modifying algorithm complexity, system or block-level architecture, and/or using leakage reduction techniques such as fine-grained clock and power gating. This dissertation introduces and explores these techniques for digital signal processing circuits and evaluates their system-level impacts. Multiple examples are discussed including: reducing memory overheads using approximate coefficients for an FIR filter, block-level error analysis of a logarithm unit and its impacts on system-level accuracy in voice activity detection, and the development of a ULP, scalable DSP fabric for flexible on-chip processing. The proposed methods evaluate the accuracy, area, power, and latency metrics for these block-level modifications and discuss the system-level consequences. The impact of these design decisions reduces overall power consumption without compromising system flexibility. The solutions presented in this work are by no means comprehensive; however, a general framework is built in this dissertation that can become a basis for further innovation and expansion.

Approval Sheet

This dissertation is submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy (Electrical and Computer Engineering)

Alicia Klinefelter

Alicia Klinefelter

This dissertation has been read and approved by the Examining Committee:

Benton H. Calhoun

Benton H. Calhoun, Adviser

John Lach

John Lach, Committee Chair

Joanne Bechta Dugan

Joanne Bechta Dugan

Scott Acton

Scott Acton

Kevin Skadron

Kevin Skadron

Accepted for the School of Engineering and Applied Science:

James H. Aylor

James H. Aylor, Dean, School of Engineering and Applied Science

August 2015

Every day you may make progress. Every step may be fruitful. Yet there will stretch out before you an ever-lengthening, ever-ascending, ever-improving path. You know you will never get to the end of the journey. But this, so far from discouraging, only adds to the joy and the glory of the climb.
- Sir Winston Churchill

Acknowledgments

I've met so many great people during the last five years of graduate school. If it weren't for my family, friends, and colleagues, this would have been a much less enjoyable experience. I hope I'm able to remember everyone that has helped me along the way as I'm incredibly grateful for the people that took the time out of their days to work with me.

My adviser, Ben Calhoun, was the motivation for me attending the University of Virginia many years ago. It was his attention to detail and pursuit of excellence in all professional areas that made me appreciate having the opportunity to work with him. His support has been endless and his professional advice invaluable. I am grateful to him for taking me on as his student and for guiding me through the last five years of graduate school. My committee members: John Lach, Joanne Dugan, Scott Acton, and Kevin Skadron have also been a great source of advice during the proposal process and towards the end. At times my work can span a few disciplines and all members have provided thoughtful evaluation of all aspects of my research.

I would like to thank many of the students that are part of the Robust Low Power VLSI group as I've had the opportunity to work with so many great people over the years. In particular, I worked closely with Yousef Shakhsheer, Kyle Craig, and Yanqing Zhang. I was so lucky to come into graduate school to work with the people that would not only teach me so much technically, but also be great friends long after the projects and deadlines were over. When I first started, Yousef seemed to have an endless amount of patience for my lack of knowledge and always took the time to explain new concepts to me. In the moments where

I doubted myself most, Kyle was always there for a pep talk, but also my technical match when it came to on-the-fly brainstorm sessions (when we weren't arguing, of course). Yanqing was one of the most curious people I met in graduate school that taught me the research mindset doesn't end when you leave work. He was a great model of how to start and conduct research and I'm grateful that I had him as an example. Jim Boley and Aatmesh Shrivastava were also great collaborators during the BSN project and always had interesting topics to discuss or things to joke about.

In my later years of graduate school, I was particularly grateful for having Seyi Ayorinde, Chris Lukas, and Farah Yahya in our lab. Seyi was always a level-headed source of advice as well as a great person to flesh ideas out with. He has showed me by example that it's always best to be honest about what you do and do not know as that provides the best opportunities for growth. Chris was always up for interesting discussions whether they were technical or not and always echoed my exaggerated sentiments about life in general, which was much appreciated. I learned so much about professional and technical communication from Farah, who always had great advice about dealing with conflict while also being refreshingly honest and objective. Many of the other bengroup students of past and present have also played vital roles in my time at UVA, including Randy Mann, Satya Nalam, Joe Ryan, Sudhanshu Khanna, Patricia Gonzalez, Divya Akella, Arijit Banerjee, He Qi, Yu Huang, Harsh Patel, Abhishek Roy, Ningxi Liu, Manula Pathirana, Dilip Vasudevan, and Terry Tigner. Students from John Lach's lab including Saad Arrabi and Jeff Brantley became good friends during the CPS project. I always appreciated Saad's upbeat attitude in any situation as well as Jeff's intense attention to detail and high level of care he had for his work.

Our collaborators at the University of Washington and the University of Michigan were integral to the BSN project. Helen Zhang and Jason Silver taught me much about analog design during my first year, and Luis Perez was great to collaborate with during the NEMO project. Nate Roberts and Moh Faisal, from the University of Michigan, both put in so much time to complete the November 2012 tapeout. I'm glad that I had the opportunity to talk to

them about RF design, PLLs, and learn about best practices at other universities.

I appreciate everyone at PsiKick for allowing me to use their resources for the last chapter of my dissertation and have some precious die area to tapeout a design in the fall.

My colleagues from Intel where I interned during the summer of 2013, including Joe Ryan and Jim Tschanz, were great sources of advice from the start of internship to the end of graduate school. I'm indebted to them for the time they both took to mentor me during my time there.

My mom, Lourdes, and sister, Laura, have always been understanding during my busy times in graduate school. They never doubted that I had the capabilities to finish and would always be around to openly listen to my complaints and stories. They were both incredibly supportive and I could always rely on them for honest advice when I needed it.

I am thankful for my colleague and boyfriend, Ben Boudaoud. I have yet to meet someone that matches his breadth and depth of knowledge in electrical engineering, and he was truly a source of both advice, information, and support over the last few years. He always reviewed my papers and engaged me on every doubt in my work with a passion and rigor as if it were his own. His selflessness throughout this process has been extraordinary and I'm glad that I had the opportunity to meet him during my time here at UVA.

Finally, I would like to thank my funding sources - without them, this work would not be possible. This work was funded in part by the NSF (1035771), the NSF NERC ASSIST Center (EEC-1160483), and by the Office for Naval Research (ONR).

Contents

List of Tables

List of Figures

Acronyms

ADC Analog-to-Digital Converter

AFE Analog Front-End

AFib Atrial Fibrillation

APK Access Pattern Kernels

ASIC Application-Specific Integrated Circuit

BSN Body Sensor Network

CMOS Complementary Metal Oxide Semiconductor

COTS Commercial Off-the-Shelf

DCT Discrete Cosine Transform

DMA Direct Memory Access

DSP Digital Signal Processing

DVS Dynamic Voltage Scaling

DMEM Data Memory

ECG Electrocardiography

EDA Electronic Design Automation

EEG Electroencephalography

EMG Electromyography

FIR Finite-Impulse Response

FFT Fast-Fourier Transform

FOM Figure of Merit

FPGA Field Programmable Gate Array

GDSII Graphic Database System (layout database format)

GPP General-Purpose Processor

HDL Hardware Description Language

IC Integrated Circuit

IIR Infinite-Impulse Response

IMEM Instruction Memory

IoT Internet of Things

LEF Library Exchange Format

MCU Microcontroller

P&R Place and Route

RC RTL Compiler

RF Radio Frequency

ROM Read-Only Memory

SCM Standard Cell Memory

SoC System-on-Chip

SPC Signal Power Circuit

TEG Thermoelectric Generator

ULP Ultra-Low Power

UWB Ultra-Wideband

VAD Voice Activity Detection

VDD Supply Voltage

V_{GS} Gate-to-Source Voltage

VSS Negative supply voltage, ground

V_T Threshold Voltage

WT Wavelet Transform

Chapter 1

Introduction

1.1 Motivation

Technology is becoming embedded into the background of everyday life through continuous monitoring of people, infrastructures, and the environment. Many people are running diagnostics, receiving alerts, and actuating decisions about their homes, health, and businesses from devices that seamlessly interact with mobile platforms such as cell phones (Figure 1.1). Within this Internet of Things (IoT) space, body sensor nodes are one application seeing a surge in popularity and relevance. With an increase in life-expectancy, a large aging population, and a desire for autonomy amongst the elderly, research in the area of body sensor nodes is increasing to provide devices for unobtrusive and precise monitoring [1]. To increase adopt-ability these ultra-low-power (ULP) platforms should be non-invasive, reliable, safe, secure, and have long lifetimes to avoid frequent battery replacement [2]. Due to this lifetime constraint, body sensor nodes and other ULP systems demand energy-efficient sensing, signal processing, and transmission to meet their stringent power requirements. Consider a BSN SoC with four input channels for electrocardiography (ECG)/ electroencephalography (EEG)/electromyography (EMG), analog-to-digital converter (ADC), microcontroller (MCU), radio, and power management runs entirely on power harvested from body heat with no

battery [3]. To enable this, the chip must consume an average power less than the harvested power, which is typically in the 30-50μW range. When extracting heart rate from ECG and sending regular RF updates, the entire chip consumes just 19μW [3]. Since the chip is powered from harvested energy, the processing blocks are power-limited to keep the total average chip power below the harvested amount and avoid depleting energy on the storage capacitor. The application space for BSNs spans data transfer rates from 1-10,000bps with lifetime requirements from seconds (RFID) to years (temperature, light, and pressure) [2]. To achieve the broadest application scope for the platform, high levels of programmability are needed. The presented BSN node aims to be flexible and low power while meeting its throughput requirements for a variety of applications from ECG to speech processing.

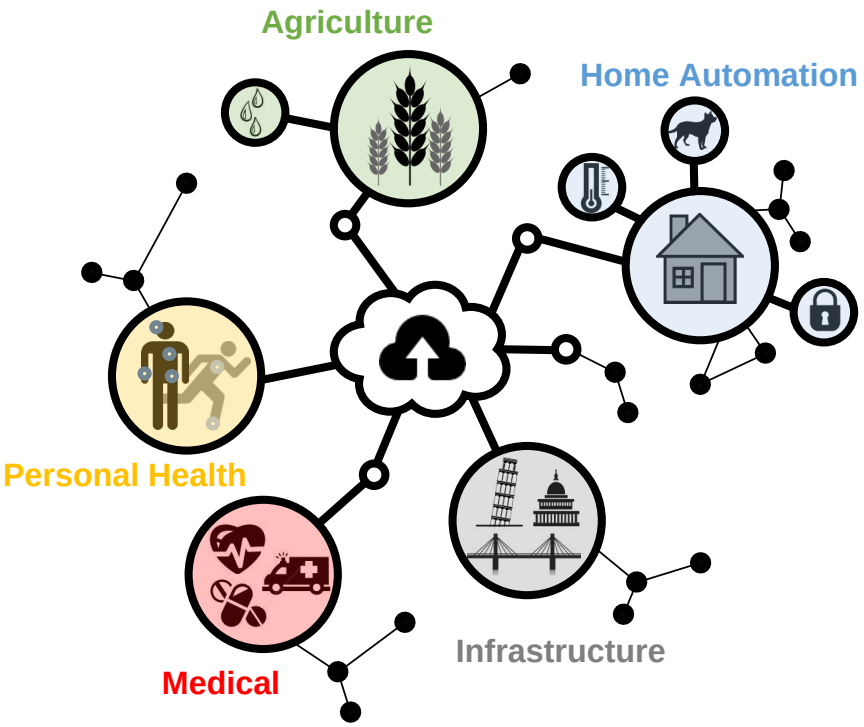


Figure 1.1: The Internet-of-Things (IoT) application space including personal health, medical, infrastructure, home automation, and agriculture.

In a system such as the BSN SoC, the circuits that consume the most instantaneous power are typically in the analog domain. In particular, the AFE and RF transceiver dominate the

power budget when active. Within the digital domain, memories often consume the largest percentage of both active and leakage powers due to demands for high capacities as a result of their relevance. Figure 1.2 shows the percentage of power consumed by system components for a variety of recent SoC platforms. Based on the data presented in this figure, it can be concluded that DSP power can be a minor or significant percentage of the overall power budget depending on the application. As an additional point of comparison, the current budget of the BSN chip in [3] showed that a third of the total current is consumed by the digital circuitry. Since this is a significant portion of the total current consumed by the chip, energy optimizations in the digital domain will have a large impact on the overall system power. Although digital logic is not always the primary power consumer, it can be leveraged to reduce the reliance on higher-power components of the system. For example, by using energy efficient on-chip compression methods, the number of bits that need to be transmitted over the radio can be reduced, leading to dramatic savings in the system power. When designing flexible platforms for a wide variety of target applications, it is important for designers to consider both direct reduction in logic power, while also creating avenues for more application specific reductions.

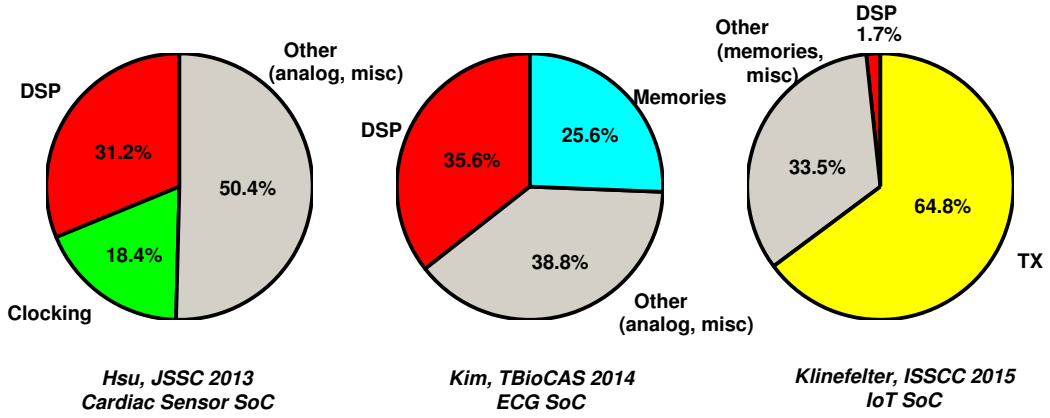


Figure 1.2: Breakdown of SoC power by classes of circuit components: (a) [4], (b) [5], (c) [6].

This work focuses on improving power consumption through novel DSP techniques at the algorithm, architecture, and circuit level. The digital processing schemes on SoCs that

have been proposed for the low power space are typically a mix of microcontroller-based processing and a small set of accelerators (e.g. FIR filter, FFT). Taking these frequently used algorithms and mapping them directly into hardware provides energy improvement, but when DSP power is not a large percentage of the system budget, these gains are low-impact. Additionally, for these systems to be flexible across a wide-range of IoT applications, they should not only accelerate a small and fixed set of operations. For example, in the event the user wants to implement a Wavelet Transform and magnitude calculation on a platform with an FIR and FFT accelerator, they must resort to using the microcontroller to implement a majority of the logic. This leads to diminishing returns from focusing solely on optimizing the DSP logic as its impact on system-level power improvements is heavily dependent on the user's application.

In addition to performing power optimizations at the processing element level, designers can also leverage DSP to reduce power in other areas of the system. To help structure the discussion of system-level power improvements, two classes of power reductions are introduced: direct and indirect. For the sake of this discussion, consider a system undergoing power optimization. Let the un-optimized system power be represented by, P_0, and the post-DSP optimized power be represented by P_{OPT}. The system-level power improvement, ΔP_{SYS}, can now be defined as the difference of P_0 and P_{OPT} as seen in Equation 1.1.

$$\Delta P_{SYS} \equiv P_0 - P_{OPT} \tag{1.1}$$

A direct power improvement technique reduces system power exclusively through DSP logic optimization. An example of direct improvement is using power gating or an optimized multiplier within a heavily used DSP module to reduce overall system power. Let DSP power improvement be defined as ΔP_{DSP}. When this DSP power improvement closely approximates ΔP_{SYS}, the power improvement is considered direct as shown in Equation 1.2.

$$\Delta P_{SYS} \approx \Delta P_{DSP} \Rightarrow \text{Direct Reduction} \tag{1.2}$$

An indirect DSP power reduction technique reduces system power by using digital logic to impact the power consumption of more energy critical system components. An example of this is using DCT-based compression to reduce the amount of data that's stored or transmitted wirelessly, typically a much higher power operation. In these cases, ΔP_{DSP} does not approximate ΔP_{SYS} as the primary power reduction is not within the DSP circuit itself. This relationship is described in Equation 1.3.

$$\Delta P_{SYS} \gg \Delta P_{DSP} \Rightarrow \text{Indirect Reduction} \tag{1.3}$$

For the purpose of this work, two types of indirect DSP power reduction techniques are considered: data reduction[1] and platform optimization. Data reduction-based methods rely on approximation and coding of data to reduce the representation required for stored and transmitted information. Platform optimization methods leverage application specific opportunities into system-level power reductions. This includes all-digital low-power wake-up and digitally assisted analog such as improved ADC techniques or all-digital phase-locked loops. Additionally, as digital techniques continue to improve, there is a growing trend towards digitization of analog leading to increased robustness, area, configuration, and reduced variability. Due to this, recent proposals recommend that many analog signal processing tasks be relegated to the digital domain [7].

This work aims to reduce system power through direct and indirect methods and to define a methodology for applying these techniques to any VLSI system with energy-efficiency as the driving metric. To accomplish this, the challenges and tradeoffs of this approach must be evaluated including precision/accuracy versus system robustness and specialization versus energy-efficiency. The solutions presented here are by no means comprehensive. However, a general framework is built in this work that can become a basis for further innovation and expansion.

[1]The term "data reduction" is used here in place of "compression" to not confuse classic data compression methods with the sub-category of indirect power reduction through DSP.

1.2 Thesis

Energy-efficiency has displaced performance as the primary design constraint in the optimization of digital integrated circuits particularly in the emerging area of IoT applications. The demand for these systems to have long lifetimes while also reporting useful data to the user places an emphasis on system processing flexibility under rigorous power constraints. To reduce total system power and increase processing capabilities on energy-constrained SoCs, novel architectures utilizing both direct and indirect DSP optimization techniques should be considered. The proposed methods evaluate the accuracy, area, power, and latency metrics for these block-level modifications and discuss the system-level consequences. The impact of these design decisions will reduce overall power consumption without compromising system flexibility.

1.3 Approach

The need for DSP-based improvements is motivated by the batteryless SoC case study presented in Chapter 3. For this system, it's clear that there is the need for aggressive power reductions from all domains of the system as they each significantly contribute to the system power budget. We separate the problem of lowering system-level power into two categories: direct and indirect improvements.

1.3.1 Directly Reducing System Power Through DSP

This is the more conventional method of reducing power by making architecture and circuit-level optimizations to individual DSP components but not impacting the operating modes or building blocks of other system components. In this work, we propose a multi-channel FIR filter with a power-of-two based signal power extractor to compute the power spectral density (PSD), a frequently used function in feature detection. This is traditionally computed in the frequency domain using the FFT, but this work seeks to lower the power through algorithmic

changes and approximation. The analysis of logarithmic approximations presented in Chapter 6 have both an indirect and direct impact on system power. A variety of approximations are proposed to minimize the power consumption of the standalone logarithm using non-linear optimization techniques in MATLAB. The digital signal processor presented in Chapter 7 also directly reduces the power of frequently used algorithms such as the DCT and CWT by using a highly energy-efficient FFT kernel as the core of the inner-product processor. This hardware re-purposing reduces system area and leakage as opposed to using dedicated accelerators for each task.

1.3.2 Indirectly Reducing System Power Through DSP

Digital capabilities are rapidly improving fueled by Moore's Law, but technology scaling trends are only conditionally beneficial for analog circuits such as ADCs, amplifiers, and even arguably mixed-signal circuits such as SRAMs. Since these single-chip solution systems include many analog and mixed-signal components, finding a way to slowly replace their functionality using DSP will reduce power and increase robustness as technology continues to scale. This work looks at a few ways to indirectly improve system power including reducing the reliance on on-chip SRAMs for an FIR filter by generating coefficients on-the-fly instead of storing them on-chip. Another piece of work looks at a voice activity detection pipeline for a speech processor and attempts to relax accuracy constraints on the arithmetic components to test the impacts on system detection accuracy. Additionally, this work aims to indirectly reduce system power by intelligently consolidating hardware accelerator blocks into an efficient but minimum area digital signal processor. This processor can be used to compress data before storage or transmission to similarly reduce the reliance of the analog circuitry in an SoC.

1.4 Dissertation Contributions and Organization

1.4.1 Background

Chapter 2 gives background information on the topics covered in this work. This section describes the components of ULP SoCs and the typical low power design methodologies used to achieve batteryless operation. Techniques for continuing to reduce system power from the architecture down to the circuit level are also discussed.

1.4.2 Body Sensor Node SoC

Chapter 3 focuses on the design of a batteryless SoC targeted for the biomedical and personal health monitoring space. There were two full versions of this platform with each targeting a different set of sensing modalities and processing capabilities. Version 1 of the platform acquired ECG data through a low power AFE, extracted the heart-rate, and transmitted raw data for 19μW while running off of harvested energy from a TEG. The first revision acquired x, y, and z-axis accelerometer data from an ADXL362Z tri-axis accelerometer, concatenated and filtered the data, and transmitted over UWB for 6.45μW. Both versions showed high levels of system integration compared to the state of the art. These platforms serve as a motivation for why direct and indirect DSP methodologies for reducing system power can have system-level impact on the power consumption and lifetime of these devices.

1.4.3 ULP Signal Power Extractor

Chapter 4 discusses the design of a ULP signal power extractor (SPE) designed for the body sensor node SoC described in Chapter 3. The SPE is comprised of an FIR filter and digital envelope detection circuit used to measure signal power within a specific frequency band. This design replaced an analog circuit implemented to achieve the same functionality for orders of magnitude less power.

1.4.4 Approximate Coefficients for FIR Filters

Chapter 5 describes a method for system-wide power reductions through indirect DSP using FIR filtering. Since on-chip memories are one of the most high-power circuits in the digital domain, this method attempts to bypass them by adding additional digital logic to generate filtering coefficients on-the-fly. For systems such as BSNs, filtering is a very frequent operation, and to filter data across a variety of bands these coefficients must be stored in the data memory at program time. This leads to the DMEM leaking unnecessarily just for storing this data. By generating the coefficients as needed, the system can reduce the reliance on the on-chip SRAMs and power gate larger portions of the memory to reduce leakage.

1.4.5 Arithmetic Approximation Methods for a Hardware VAD

Chapter 6 looks at the system level impacts of block-level error in a voice activity detection speech pipeline. This work primarily focuses on the design of a base-2 logarithm unit typically used for speech applications. By completing early, system-level modeling of the speech pipeline to determine the system-level impact of injected error, the accuracy constraints of system components can be relaxed in the event they contribute little to the overall system accuracy. If so, the power can be greatly reduced through methods such as approximate computing. Other components such as the DCT and triangular filters of the voice activity detection pipeline are evaluated for the impact of block-level error on the system. This chapters shows that indirect and direct methods of reducing system power through DSP work best when used in conjunction.

1.4.6 A Flexible, ULP Digital Signal Processor

Chapter 7 takes the results of design decisions and results from previous chapters to describe the design of a ULP digital signal processor targeting applications in the IoT space. Much work has been completed looking at the flexibility-efficiency trade-off between using GPPs,

FPGAs, and ASICs in the ULP space, but digital signal processors are rarely mentioned despite their suitability for IoT applications. This chapter examines the algorithms frequently run on ULP systems to date and discusses an architecture that minimizes hardware resources but achieves high-levels of flexibility to target a range of DSP tasks.

1.4.7 Conclusion

Chapter 8 concludes the dissertation with a summary of the contributions, descriptions of future work and research goals, and discusses the broad impacts of this work.

Chapter 2

Background

This section is an introduction to topics that are lightly covered throughout the dissertation, but integral to the motivation and work.

2.1 Energy-Efficient SoCs for the IoT

Targeting a wide variety of long-term monitoring applications in the areas of health, agriculture, and home automation requires systems with flexible processing capabilities and a variety of analog and digital interfaces. While these systems can be implemented using COTS devices such as standalone microcontrollers, radios, and sensors, they become smaller and more energy efficient when tightly integrated onto a single-chip solution such as an SoC [2].

The desired operating model for these SoCs for the IoT is to sense, process/compress, and store data in a sink device such as a memory or transmitter for μW of power and potentially running off of harvested energy. In order to achieve these goals, these systems must integrate reliable and low power components such as regulation, clocking, sensing, and processing.

2.1.1 Components

To reduce the overall system power, each component of the system must be examined for potential energy savings. Circuit components typically seen in a ULP SoC are shown in Figure 2.1 and include the following:

1. **General purpose processor (GPP) or bus controller**: GPPs are used for flexible processing when on-chip accelerators cannot accommodate the desired task. They are also used as bus controllers by dictating the flow of data on-chip from module to module.
2. **Storage (program and data memories)**: The system's bus controller interfaces to an IMEM to receive its instructions, and there is general-purpose DMEM that is used for storing system data. Capacities for each are determined based on the application and use-case scenarios.
3. **Sensing**: Hardware platforms targeting the IoT space can acquire data from a series of sensing sources ranging from ECG, EMG, and EEG electrodes to accelerometer and gyroscope ICs. These SoCs should have the capabilities to interface to both analog and digital sensors. Examples of common digital interfaces include SPI, I^2C, and UART.
4. **Power management**: A digital system requires a stable DC voltage applied to all of the gates. Whether the hardware is operating off of energy from a battery or harvested from an ambient source, the available on-chip voltage rails must be regulated for reliable operation. If the energy source is producing a voltage that is below the expected system rail, it must be stepped up, and if the supplied voltage is too high, it must be stepped down.
5. **Clocking**: Most digital, CMOS circuits use a clock signal to synchronize all of the sequential elements in the system. This clock is typically generated by on on-chip oscillator being supplied by a crystal or a programmable source such as a phase-locked loop with a range of output frequencies.

6. **Accelerators**: There are sets of tasks that are frequently run on application specific platforms such as ASICs. Some of these algorithms may map poorly to a GPP and may hog clock cycles that could be used for system control. In these cases, the algorithms are accelerated by developing dedicated hardware to perform these tasks. This is generally orders of magnitude more energy-efficient than implementing them on a GPP, with the added cost of the accelerator area on die.
7. **Radios**: With mobile devices such as cell phones, tablets, and smart watches becoming ubiquitous, including a wireless interface increases the usability of these platforms. These radios should be energy-efficient to not dominate the system power budget and take advantage of the short range requirements and channel asymmetry typically seen in the IoT application space.

The classic big power consumers for SoCs are analog modules such as the AFE and radios or the subthreshold SRAMs. Due to the relatively low sampling and datarate requirements for IoT applications, these circuits can be heavily duty-cycled to keep power consumption down.

2.1.2 Enabling Batteryless Operation

Since the power consumption of these SoCs is expected to be in the μW range, powering them solely from harvested energy becomes feasible. This energy can be harvested from the surrounding environment from a variety of sources shown in Table 2.1. The choice of harvesting mechanism is highly application dependent, and primarily depends on the form factor, power density, and application appropriateness. For example, to achieve high power densities from inductive coupling, a high quality factor, Q, inductor is required as well as proximity to the source inductor. For a high power system that cannot achieve this close range, it is not appropriate and requires a larger form factor to get the required power density. Similarly, vibrational energy harvesting would not be a good fit for a stationary platform.

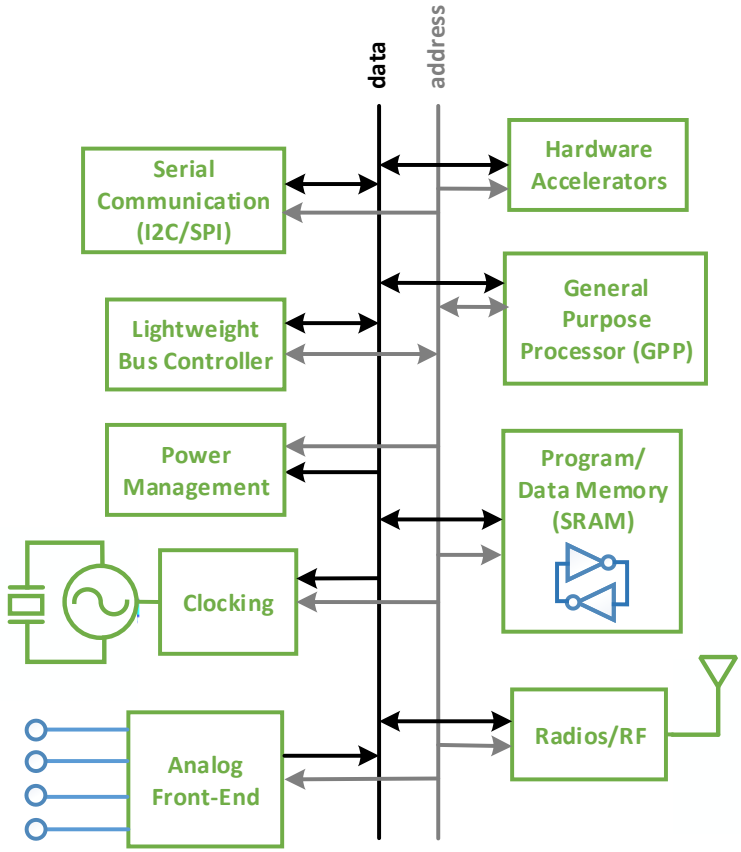


Figure 2.1: Typical components and structure of an SoC

When the energy source is not available (e.g. direct sunlight for outdoor solar) and energy cannot be reliably harvested, the system must conserve the previously stored energy. Using a monitoring circuit to periodically check the available system energy and gracefully shut system blocks down as in [9] can prevent complete system state loss and node death. Since peak power surges coming from the radios or AFE may discharge the nodes storage capacitor below a usable voltage, care must be taken to only enable these circuits when harvesting conditions have been stable and there is ample energy available.

Source	Limitation	Power Density
Inductive Coupling	Short range (cm), inefficient	$\propto$ D, Q, $1/d^3$
Far-field RF	Base station (a few m), safety	$\propto 1/d^2$
Solar (indoor)	Available artificial lighting	10 $\mu W/cm^2$
Solar(outdoor)	Direct sunlight	10,000 $\mu W/cm^2$
Vibration	Relatively constant movement	4 $\mu W/cm^2$
Thermoelectric	Thermal gradient	25 $\mu W/cm^2$

Table 2.1: Energy-harvesting mechanisms [8]

2.1.3 Energy Efficiency versus Flexibility

Conventional hardware platforms range from highly flexible and configurable (e.g. CPUs and FPGAs) to highly application specific in the case of ASICs. The most flexible hardware platforms exhibit poor energy efficiency due to overheads associated with instruction fetch/decode in the case of CPUs or interconnect in the case of FPGAs. For sophisticated operations such as signal processing algorithms (e.g. FFT, FIR filtering), using a simple core without a multiplier/MAC peripheral to compute the result will take many clock cycles. Implementing these algorithms on an ASIC, where the hardware is specifically designed to do one task, leads to highly energy-efficient designs but with poor flexibility. A recently published BSN SoC [10] shows the typical energy gains achieved when implementing common biomedical system tasks using accelerators in Table 2.2. This system used an 8-bit PIC as the GPP comparison point.

Energy Efficiency Comparison Per Sample		
30-tap FIR Filter	GPP	6.3nJ
	Accel.	57pJ
Signal Energy Extractor	GPP	3.6nJ
	Accel.	530fJ
R-R Interval Extractor	GPP	12pJ
	Accel.	3fJ

Table 2.2: Energy reduction achieved through implementing common tasks as accelerators versus in a GPP [10]

Here, gains are seen on the order of 1000x. Although ASICs offer extreme energy efficiency, they can be orders of magnitude more expensive than more general purpose COTS parts. Additionally, when fabricating ASICs, every square micron becomes a costly commodity when anticipated volumes are low. For this reason, the target application set must be determined before design-time to evaluate the energy-area-cost trade-off of implementing the platform using an ASIC and whether certain tasks are well-suited for hardware acceleration. For this work, our goal is to implement a batteryless SoC that requires aggressive power scaling. COTs solutions currently on the market would consume 100s of μW to mW of power to achieve the same functionality. Due to this, an ASIC-based platform with carefully selected accelerators is needed to achieve our goals.

2.2 Low Power Design Techniques

Reductions in the total system power consumption, P_{TOT}, of a digital circuit result from reducing the active/dynamic (P_{ACT}) or the leakage (P_{LEAK}) components of the overall power where $P_{TOT} = P_{ACT} + P_{LEAK}$. The active power component is comprised of charging load capacitances, C_L within the driving or driven devices and from short-circuit current during switching (Figure 2.2). Since rise and fall times are non-ideal, there are periods during switching where both NMOS and PMOS devices are simultaneously conducting and there exists a direct path between VDD and ground causing short-circuit current. This is proportional to the switching activity and is a contributor to the active power. Minimizing the clock slew by constraining the fan-out during the synthesis process can minimize this component.

During transitions, energy is drawn from the power supply; some of it is dissipated in the devices and the rest is stored in C_L. This power supply energy, E_{VDD}, can be derived by integrating the instantaneous power over a window of time shown in equation 2.1 [11]. The power consumption is computed by incorporating how frequently the device is switched

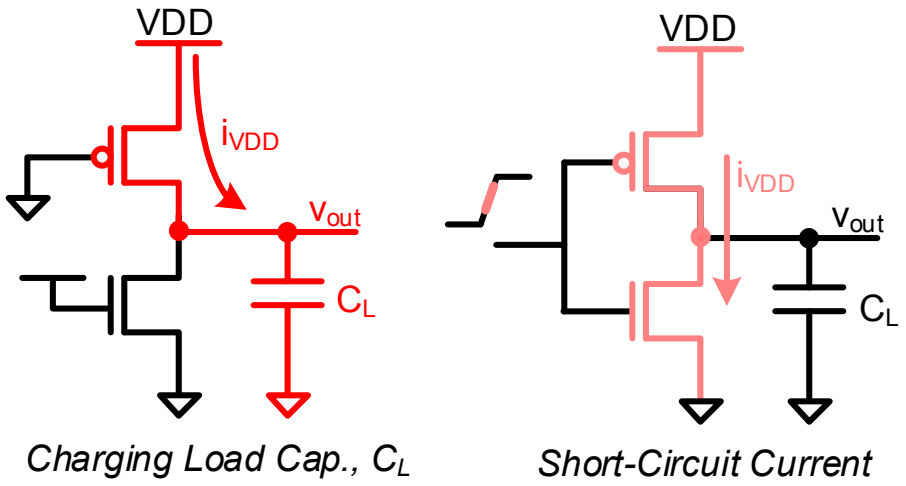


Figure 2.2: Sources of active power consumption in CMOS circuits

(Equation 2.2). Here, f is the frequency of switching and α is the activity factor.

$$E_{VDD} = \int_0^{\infty} i_{VDD}(t) V_{DD}\, \mathrm{d}t = V_{DD} \int_0^{\infty} C_L \frac{dv_{out}}{dt}\, \mathrm{d}t = C_L V_{DD} \int_0^{V_{DD}} \mathrm{d}v_{out} = C_L V_{DD}^2 \quad (2.1)$$

$$P_{ACT} = C_L V_{DD}^2 f \alpha \quad (2.2)$$

The result in Equation 2.2 shows that the active energy and power depends on the square of the supply voltage. This presents a strong design knob for reducing the overall energy of digital circuits.

When no switching occurs and the devices are idle, static/leakage power consumption begins to dominate. One source of this leakage is subthreshold current that occurs when the devices are "off" ($V_{GS} < V_T$) but still experience a small drain-to-source current. This current is dependent on the V_T of the devices such that high threshold voltages lead to reduced

currents. Although these subthreshold currents are small, the on-current in subthreshold remains larger than the off-current by enough to enable proper functionality of the digital gates. This provides an opportunity to drastically reduce the power consumption by lowering VDD below V_T.

2.2.1 Subthreshold Operation

Subthreshold conduction refers to lowering the supply voltage below a devices threshold voltage, V_T, so that the device is partially conducting and operates from leakage currents. Active energy scales quadratically with reductions in the supply voltage, making it a useful knob for designs where energy efficiency is the driving metric and performance is secondary. Equation 2.3 shows the relationship between the drain current in the subthreshold region and the main device parameters: I_{D0}, the current when $V_{GS} = V_T$, V_{GS}, the gate-to-source voltage, V_{th}, the thermal voltage kT/q, V_T, the threshold voltage, and n a term dependent on the ratio of the depletion to oxide device capacitances.

$$I_D \approx I_{D0} e^{-\frac{V_{GS}-V_T}{nV_{th}}} \tag{2.3}$$

Similar to the superthreshold device model where the drain current has a linear relationship to the device widths and lengths, the subthreshold model also has exponential dependencies on other parameters making device sizing a weak knob in this region.

Besides the performance degradation that comes with subthreshold operation, there are also non-idealities such as increased sensitivity to variation due to an exponential dependence of the drain current of a device on V_T. The threshold voltage can vary on chip due to non-uniform doping or lithography such that the impact on performance variability is pronounced in this region.

2.2.2 Clock and Power Gating

When a circuit element is not active but requires state retention, switching power can be reduced by clock gating to reduce active power on the clock tree. There are multiple types of clock gating cells. A simple combinational gate can be used (e.g. AND), but this is susceptible to glitching. Sequential clock gating methods involve latching the enable signal and using the output of the latch as the input to the combinational gate shown in Figure 2.3.

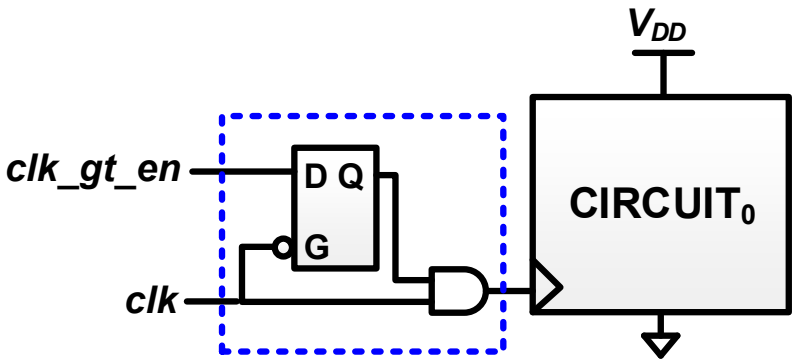


Figure 2.3: Sequential clock gating circuit

When the circuit element is not active and data retention within the block is not required, the block can be power gated to reduce leakage current. This method typically uses a PMOS header or NMOS footer to sever the connection between the supply voltage and the load during inactive periods where leakage is the primary concern (Figure 2.4).

The sizing of these power gate devices is paramount to the block's operation. Headers that are sized too small will starve the block of current during active periods, causing a potential droop in the VDD. In subthreshold operation, small decreases in the supply voltage lead to exponential impacts in the performance, which could lead to the block being non-functional. Headers that are sized too large will lead to excess leakage during idle periods. For the following work, PMOS headers were used throughout due to their reduced leakage.

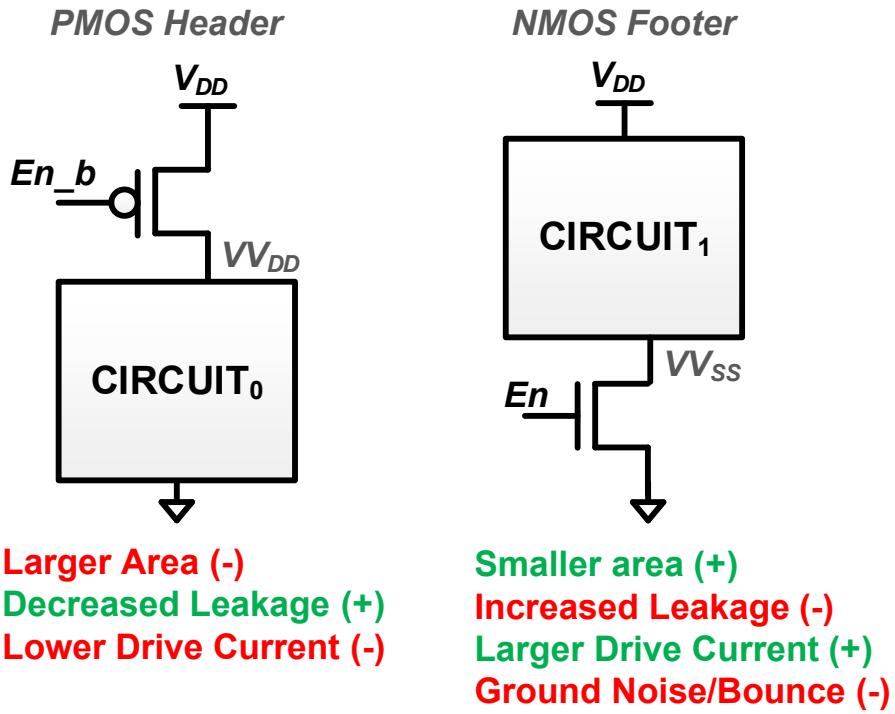


Figure 2.4: Advantages and disadvantages of each type of power gate scheme

2.3 Digital Design Flow

All of the digital circuity designed and presented in this work was generated using a Cadence EDA tool flow. All designs are first described using Verilog HDL, translated to a gate-level netlist using RTL compiler, and placed and routed to create a GDSII using Cadence Encounter (Figure 2.5).

The digital designs are composed from a standard cell library that contains gate-level primitives. Cadence Encounter enables a hierarchical approach to digital design by constructing small digital macros from standard cells and then combining these macros to create an IC in a scripted and automated fashion. For example, an adder module can be defined to have a specific propagation and input/output delay during the synthesis process and its performance can be captured in a model with few parameters. When hierarchically generating an IC that

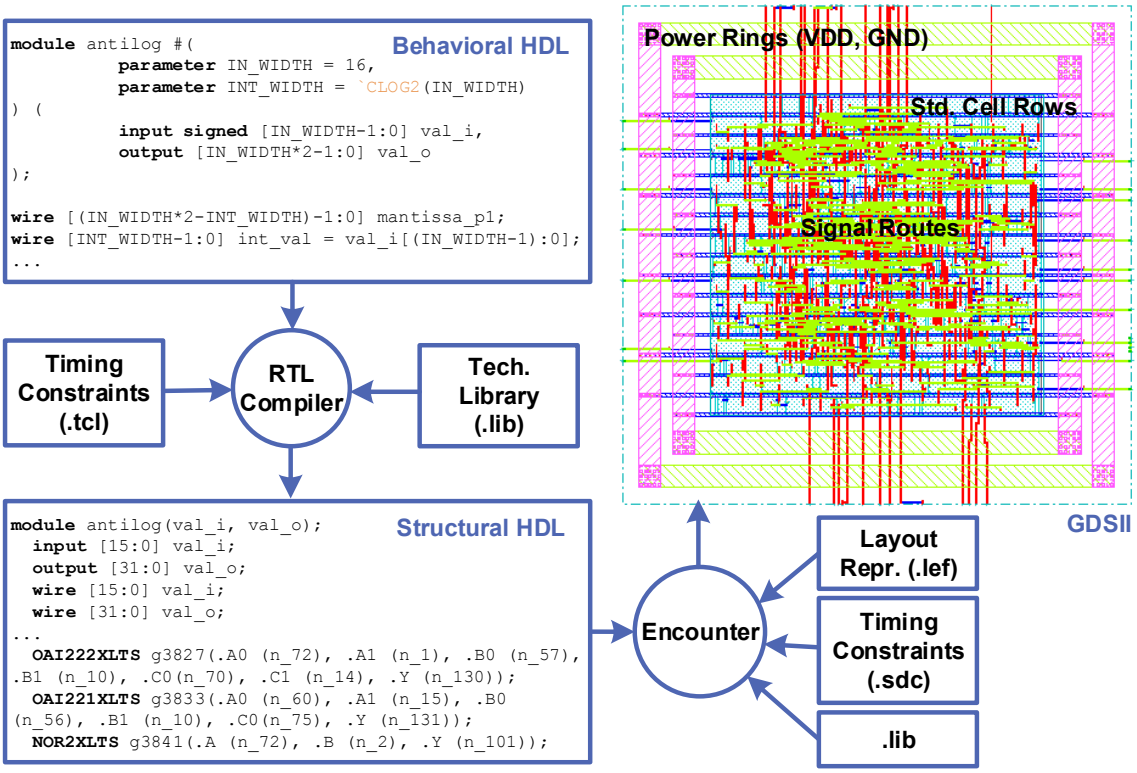


Figure 2.5: Typical VLSI CAD flow from behavioral RTL to GDSII

uses this adder module, it can be treated as a black box with known characteristics used by the place and route tool. This modular design flow greatly reduces the complexity of integration that allows complex ICs to be developed.

2.3.1 Synthesis for Subthreshold Logic

Since most digital design is automated using HDL and EDA tools, there are various techniques used for subthreshold synthesis. One technique is to do robustness analysis of the available standard cells using Monte Carlo analysis as in [12] to prune gates that are unreliable at low voltages. Gates that are known to be unreliable in subthreshold include ratioed logic (e.g. SRAM bitcells) as sizing is a weak knob in subthreshold and large device stacks (NOR gates). Standard cells with these characteristics are avoided since they degrade the reliability of the

design to the point that the circuit may not work deep into the subthreshold region.

Since subthreshold circuits are more susceptible to variations, synthesis constraints must be set with this in mind. To verify that a digital design has met the timing specifications including setup, hold, frequency, slew, and skew constraints, the user must check timing reports provided by the tools to achieve timing closure. EDA tools can determine the amount of slack in each timing parameter through each logical path in the design and report this information to the user. Since the impacts of process, temperature, and voltage variations are exacerbated in subthreshold, achieving timing closure without added timing constraint guardbands may not lead to a robust design. The designer should particularly consider guardbanding for hold time, which cannot be fixed post-fabrication and lead to design failures.

2.4 General Strategies for Reducing DSP Power

When mapping DSP algorithms to hardware, there are a few common design steps, considerations, and strategies when designing for low power. The design typically starts in MATLAB where functionality can be verified against a floating-point golden standard. From there, the algorithm can be partitioned into functional units and translated to an HDL such as Verilog. Once the design is coded, verification and validation must occur between the golden standard and the hardware implementation. During this step, the impacts of quantization and data representation on the accuracy of the results are seen, which must be considered when going from a floating-point MATLAB model to the actual design in hardware. Many COTS digital signal processors that are designed for low power use a fixed-point data representation to avoid the circuitry required for floating point conversion. There are many architectural and circuit-level design decisions throughout this process with a few described here.

2.4.1 Arithmetic Optimizations

Much work has been done in the area of designing arithmetic units typically used in DSPs including low power adders and multipliers ranging from imprecise implementations [13] to logarithmic-based processing [14]. In [15], the adder and multiplier topologies are modified to easily switch between an 8 and 16-bit datapath to reduce the switching power when 16-bit accuracy is not needed. The impacts of the error introduced by imprecise arithmetic needs to be evaluated at the application level to determine the resilience of algorithm to these inaccuracies. As these are the arithmetic building blocks of DSPs, the right topologies and bit-widths for the application should be determined before mapping the algorithm to hardware.

2.4.2 Hardware Reuse

For many DSP algorithms, the implementations can range from bit serialized to fully parallel. When algorithms are directly translated to hardware by mapping the operations of a dataflow graph directly into functional units and hard-wiring the connections between them, it results in the maximum parallelism being obtained. This allows for the minimum clock rate and supply voltage to be used, resulting in reduced energy per operation [16]. For a serial implementation that needs to achieve the same throughput as a parallel implementation, it must be clocked faster, which requires a higher supply voltage thereby increasing the power. Direct-mapped architectures are unattractive for platforms that require more flexibility primarily because the tremendous design effort involved and the added area/leakage caused by parallelizing the algorithm. Since platforms that target the IoT application space have more relaxed throughput requirements, serializing the logic for DSP implementations becomes feasible, low-area, and energy-efficient [17].

2.4.3 Memory

There is a heavy reliance on memory for a variety of classical DSP algorithms. In the subthreshold region, this can be problematic as the typical 6T SRAM bitcell is unreliable at lower voltages [18] requiring larger 8 or 10T bitcells with additional circuitry to ensure robust read and writes. Using standard-cell-based memories is area-efficient up to a certain size compared to SRAM with its peripheral circuitry, but SCMs eventually become prohibitively high-area for larger capacity memories. Some platforms use a combination of local SCM storage and then utilize an SRAM-based data memory once input data size exceeds the SCM capacity [19]. This is useful when the average size of the input data is know a priori and is small enough to justify the use of an SCM.

Determining Memory Capacity Target applications in the biomedical space include a wide range of sensing modalities, sampling rates, processing requirements, and storage considerations. As a majority of biomedical applications have sampling rates <1kHz, meaning new data samples are infrequent relative to common system clock rates used in systems operating in the near- or subthreshold regime (<1MHz). Minimum required memory capacities can be determined based on the data rate requirements between on-chip components. Along any processing path in the SoC, there exist data sources (e.g. sensor data from the ADC, data received over RX or SPI), data processing (e.g. FIR filters, CORDIC), and data sinks (e.g. TX, data memory). If the data rate of the source/processing unit, R_{SRC}, (in bits/s) is greater than the data rate of the processing/sink unit, R_{DEST}, then intermediate buffering is required. The minimum amount of memory required, $N_{MinBuff}$, (in bits) to meet application constraints is dependent on the maximum continuous runtime of the program, t_{prog}, (in seconds). Compression that occurs during data processing eases the requirements on the intermediate buffer between the processing and sink units and reduces R_{SRC}. The final relationship for determining the minimum buffer size is shown in 2.4.

$$N_{MinBuff} = (R_{SRC} - R_{DEST})t_{prog}, R_{SRC} > R_{DEST} \tag{2.4}$$

Since wireless communication consumes the most power in BSN SoCs, minimizing the time that the transmitter or receiver is on is critical in energy-constrained systems. This can be accomplished using data encoding or compression methods to reduce packet sizes, but the maximum packet size (i.e. TX/RX buffer size) is determined by the available energy for processing. The maximum radio transmit and receive buffer sizes, $N_{RX/TX}$, (in bits) can be computed using an estimate for available system energy for communication, $Eavail$, radio startup energy, $E_{startup}$, and energy/bit of the radios, $Eb_{RX/TX}$, shown in 2.5.

$$N_{RX/TX} = \frac{(E_{avail} - E_{startup})}{Eb_{RX/TX}} \tag{2.5}$$

This can reduce the leakage overheads due to unnecessary memory resources.

Commercial DSP Memory Features Multi-port memories that can perform simultaneous reads and writes to multiple addresses are beneficial for DSP algorithms such as FFTs that rely on in-place addressing algorithms to minimize memory resources. The downside of these memories is the overhead of additional read/write logic as well as conflict management for memory access hazards. The tradeoff analysis between adding additional ports to the memory versus the added performance gains must be performed on a per-application basis.

Another aspect of memory architectures commonly seen on commercial DSPs is circular buffering. Many DSP algorithms such as digital filtering rely on knowing the order of the input samples and typically shift all stored data by one (getting rid of the oldest sample) when new data is received. Due to this, a buffer of previous values (also called a delay line) needs to be maintained along with the current sample. Implementing the pointer logic required for this circular buffer structure in hardware can lead to faster and more energy efficient designs.

Chapter 3

Body Sensor Node SoC

3.1 Version 1 [1]

3.1.1 Introduction

Body sensor nodes promise to provide significant benefits to the health care domain by enabling continuous monitoring, actuation, and logging of patient bio-signal data, which can help medical personnel to diagnose, prevent, and respond to various illnesses such as diabetes, asthma, and heart attacks [20]. Though they show great potential, body sensor nodes have many design challenges that impede their widespread adoption including node operating lifetime, small form factor for wearability, and affordable cost. One of the most critical issues is node lifetime. In many applications, such as long-term monitoring of chronic illnesses, limited battery lifetimes severely undermine the deployment of body sensor nodes, since the required node operating lifetime is effectively indefinite. Supplying the node with sufficient energy over a long lifetime poses a challenge. A large battery increases the form factor of the node, making the node unwearable or uncomfortable, while a small battery requires frequent changing and reduces wearer compliance. Energy harvesting from ambient energy sources, such as thermal gradients or mechanical vibrations, potentially provides

[1] This section on version 1 is based off of the work in [10].

indefinite lifetime. Examples such as [21, 22] have shown that commercial thermopiles can be applied to body sensor nodes. To eliminate battery changing, nodes can operate solely from energy harvesting instead of using a battery, although this introduces new challenges. The full system must consume less energy than the amount harvested, high power components such as the transmitter must be heavily duty-cycled, and the node must cope with time varying harvested energy profiles [23].

To ensure sustained operation of the node using harvested energy, on-node processing to reduce the amount of data transmitted, power management, and ULP circuits are key. Since COTS based body sensor nodes (e.g. Zigbee-type radios) can consume 10s of mA during operation and consume additional board area, it is infeasible to use COTS components to build a batteryless system. Instead, we pursue an integrated ULP SoC approach. Recent advances in ULP chip design techniques, with many targeting wireless sensor networks, have enabled a push toward long lifetime body sensor node devices performing complex applications. For example, [24] presents an ECG acquisition and processing SoC with a 3-channel AFE and flexible, generic DSP components. Clock-gating and duty-cycling are used to reduce power, but without voltage reduction techniques (VDD=1.2V), the minimum power is 31.1μW. The SoC does not include a transmitter or voltage regulators. An SoC for EEG seizure detection integrates a COTS radio and does not include voltage regulation or power management [25]. The system in [26] integrates several chips with solar cells and a battery to accomplish near-perpetual operation for measuring intraocular pressure. The system consumes 3.3fW/bit at 400mV when taking one measurement every hour. These works show that energy harvesting could provide a viable power supply for ULP BSN circuits. However, integration of a complete wireless, flexible, easily deployable BSN node on an SoC that supports closed-loop power management and energy harvesting has yet to be demonstrated.

We utilize recent advances in energy harvesting, low voltage boost circuits, dynamic power management, subthreshold processing, bio-signal front-ends, and low power RF transmitters to realize an integrated reconfigurable wireless BSN SoC for ECG, EMG, and EEG applications

with autonomous power management for completely battery-free operation [3]. This SoC can run indefinitely from energy harvested from body heat while worn, and potentially decreases cost by having high integration and targeting a wide range of bio-electric sensing applications.

3.1.2 System Overview

Conventional wireless sensors use batteries (Figure 3.1), limiting node lifetime and reducing user compliance due to the requirement for charging or replacing batteries.

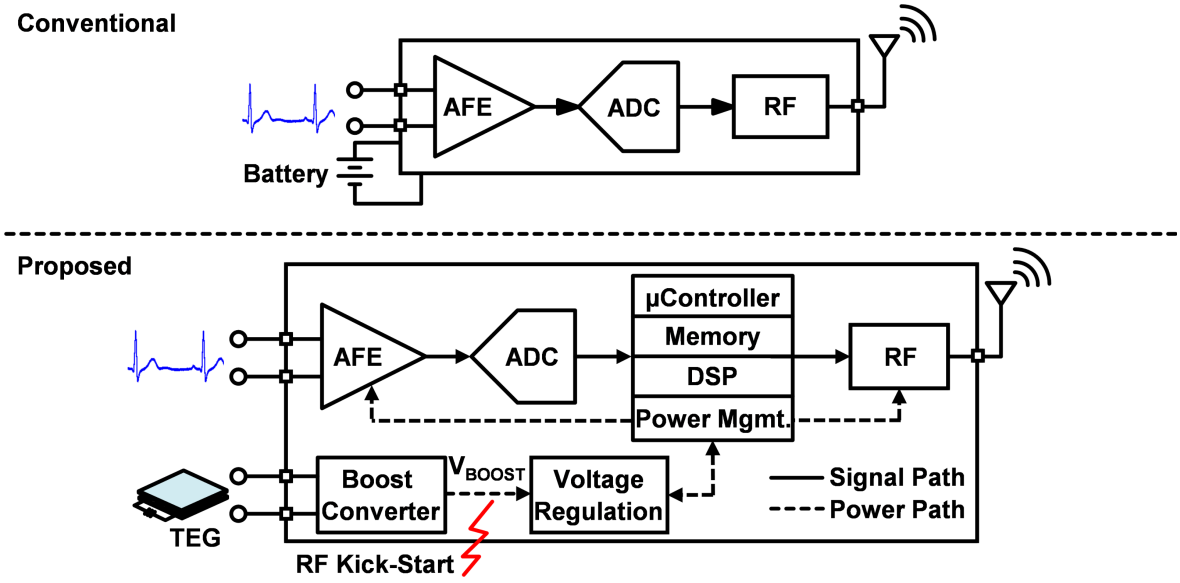


Figure 3.1: High-level diagram of conventional body sensor node solution (top) and the proposed (bottom) solution with energy harvesting, integrated power management, and ULP flexible DSP architecture

These sensors often require high data transmission rates that quickly drain battery energy. More sophisticated approaches that include on-node processing and duty-cycling of the power-hungry radio are available [27], but the lifetime of such COTS nodes is usually limited to a few days, and custom BSN SoC chips have not yet integrated radios, processing, and power management with energy harvesting. In contrast, we propose a wireless BSN SoC chip powered by energy harvested from human body heat using a TEG. This, in conjunction with ULP circuits, intelligent duty cycling of power-hungry blocks (e.g. the transmitter), and

a programmable power management system allows for indefinite operation of the chip. To demonstrate, we present a chip targeting ExG (ECG, EMG, and EEG) applications.

To achieve flexible data acquisition and processing while operating the node solely from harvested energy, we propose a system architecture, illustrated in Figure 3.2, which comprises four subsystems.

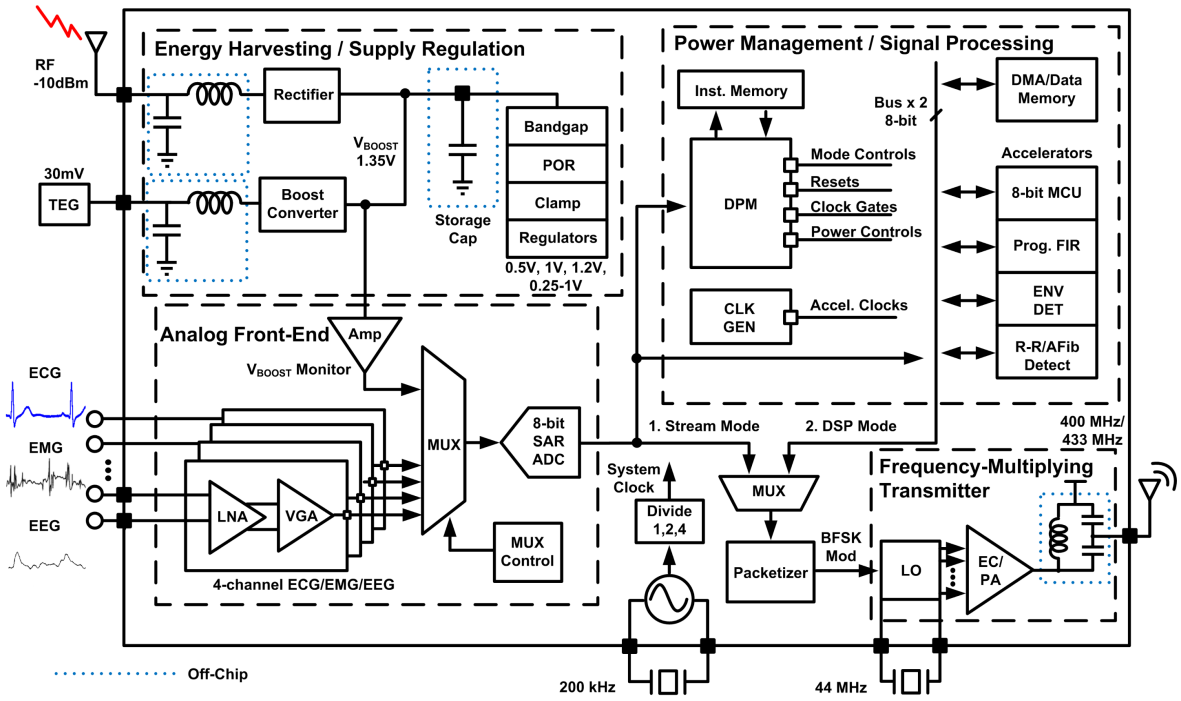


Figure 3.2: System block diagram for the proposed chip comprising the energy harvesting/supply regulation, analog front-end (AFE), subthreshold digital signal processing, and transmitter subsystems

First, the energy harvesting/supply regulation section boosts a harvested supply input as low as 30mV up to a regulated 1.35V using an off-chip storage capacitor. It provides five regulated voltage supplies to the rest of the chip, and generates a bandgap reference. Second, the four-channel AFE subsystem provides bio-signal acquisition with programmable gain and sampling rate, amplifying bio-signals as low as a few Vs while consuming <4 μ W/channel. A variable gain amplifier (VGA) maximizes the signal at the input to an 8-bit successive-approximation (SAR) analog to digital converter (ADC), reducing the ADC resolution requirement. Third, the acquired data is sent to a subthreshold digital processing

subsystem that also performs mode control and power management (including power/clock-gating of blocks and dynamic voltage scaling (DVS)) based on the available energy on the storage capacitor. The digital section includes a custom digital power management (DPM) processor, general purpose microprocessor (MCU), programmable FIR, 1.5kB instruction SRAM/ROM, 4kB data memory FIFO, and dedicated accelerators for ECG heart rate (R-R) extraction, atrial fibrillation (AFib) detection, and EEG band energy calculation. The DPM is responsible for power management, node control, data flow management, and overseeing all processing on-node. Finally, a sub-mW 400/433 MHz MICS/ISM band frequency-multiplying transmitter (TX) performs BFSK transmission up to 200 kbps. The TX has low instantaneous power consumption to avoid the need of large filtering capacitors on the supplies and is intelligently duty-cycled to achieve low average power consumption.

This was a chip with many collaborators that each worked on specific sub-systems. Since the focus of this work is on the digital processing subsystem, details regarding the other three sub-systems can be found in [10].

3.1.3 Subthreshold Digital Signal Processing Subsystem

Figure 3.3 shows the subthreshold DSP subsystem. To achieve ultra-low power consumption, we implement ASIC versions of the heart rate extractor (R-R), atrial fibrillation detector (AFib), and energy band extractor/envelope detector (ENV DET). An 8-bit RISC ISA general purpose processor (GPP) MCU executes generic computations, and a re-programmable FIR performs digital filtering. A digital packetizer streams serial data to the transmitter. Two memory arrays store the program (Instruction Memory, IMEM) and bio-signal data (Data Memory, DMEM). A DMA achieves easy FIFO control and low memory latency for the DMEM. Two 8-bit switch-box busses, controlled by the DPM, configure the connections of all the processing accelerators, MCU, DMA, and packetizer. Each input/output bus port has a 4-bit address. Having two busses eases data steering and simplifies the control instructions. To support the stoplight scheme in the DPM, each processing element has a clock-gate and

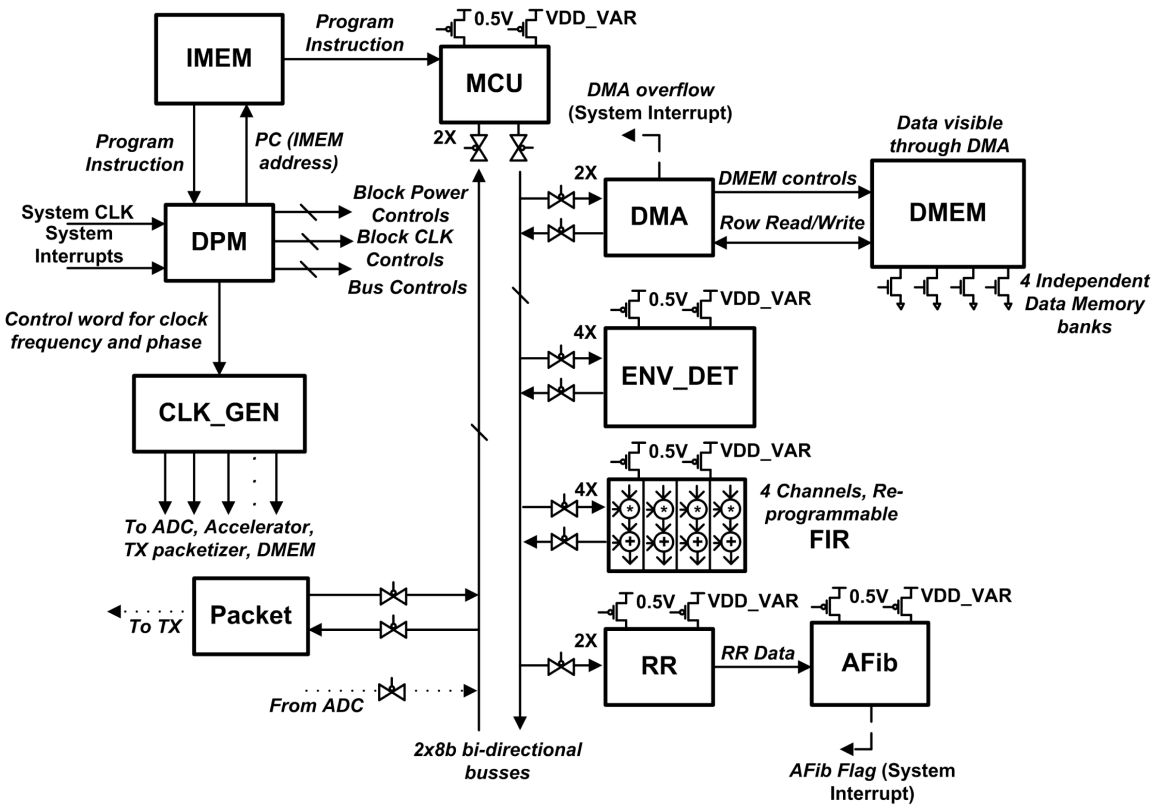


Figure 3.3: Block diagram for the proposed subthreshold data processing subsystem.

two PMOS headers [28], one connected to 0.5 V, the other to the variable voltage for DVS. The clock generator block (CLK GEN) distributes a programmable clock signal (frequency and phase) to each of the processing units.

GPP MCU The chip can process data flexibly with the MCU, use the highly efficient hardware accelerators, or cascade accelerators with MCU processing. It also can stream data on the transmitter, store and burst data, or do event based transmission. The 8-bit GPP MCU is a subthreshold RISC that is compatible with the PIC16C57 from Microchip. The MCU is designed to run arbitrary programs and functions down to 0.26V, 1.2kHz. Figure 3.4 shows the measured energy-delay (E-D) curve for the MCU and the on-chip hardware accelerators.

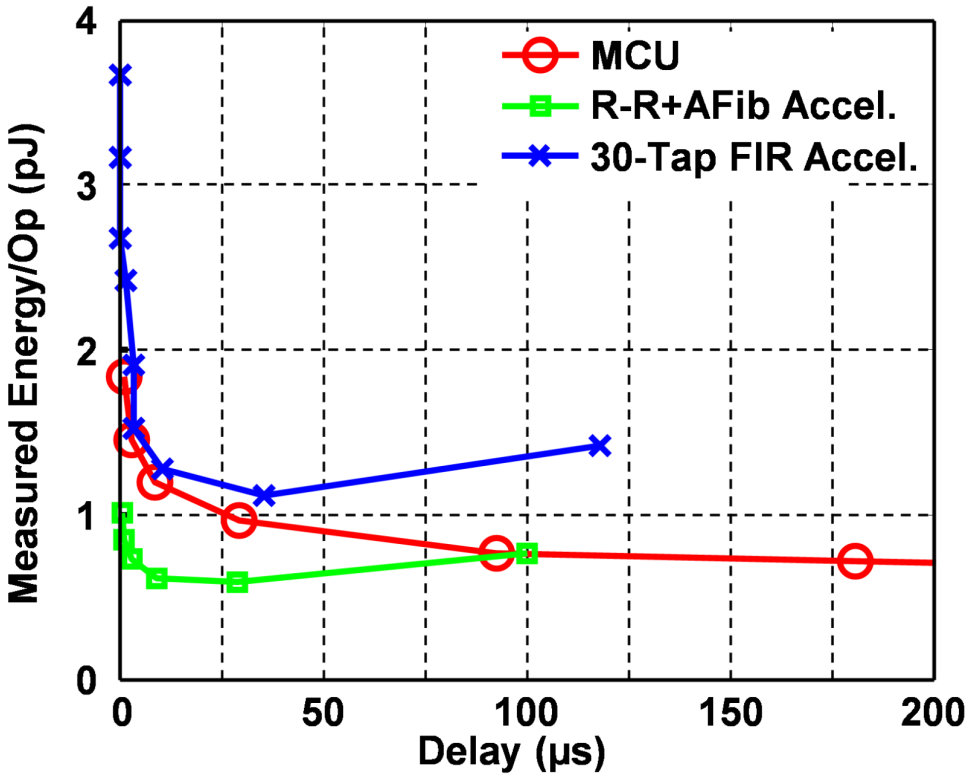


Figure 3.4: Energy-delay curves for MCU, RR+AFib accelerator, and 30-Tap, 1-Channel FIR.

The MCU consumes 0.7nW to 1.4 μ W measured power (0.26-0.55V) and 1.5pJ/op at the default 0.5V, 200kHz setting. The MCU shares the IMEM with the DPM. A multiplexer steers each instruction to either the MCU or DPM (INST_STEER) based on a special code word. When the MCU is executing instructions, the DPM automatically goes into a low power sleep mode. When the DPM is executing instructions, the MCU is either turned off or clock gated to save state. In this way, we retain the energy efficiency of the DPM as a chip controller and the generic flexibility of the MCU without requiring extra instruction memory space. The MCUs instructions are programmed at the same time as the DPM during the chips pre-deployment.

DMA The DMA is an efficient subthreshold accelerator to interface between the DMEM and the rest of the SoC. It is programmed by one instruction of the DPM and effectively

treats the DMEM as a FIFO to support efficient streaming. A clock multiplexor synchronizes the DMA clock rate to the component it interfaces to by choosing between several clock rates. A memory controller uses separate DMEM banks during green and yellow modes for easier data management. To solve the half-select stability issue during writes, we use a row buffer and only write a row when all words are ready. When the difference between the write pointer address and read pointer address is greater than or equal to 4 bytes, a DMA_flag is raised, which signifies to the DPM that there is a full packet of data. This simple and efficient mechanism of interrupting for transmission limits overflows.

AFib Accelerator The AFib detector is an ASIC accelerator that detects the arrhythmia using an implementation of the clinically validated algorithm described in [29]. It receives its inputs from the R-R accelerator and outputs an $AFib\_flag$ signal to the DPM signifying the detection of atrial fibrillation. The algorithm uses only 12 R-R intervals [29]. Many variables in the algorithm, such as the margin of error, are programmable. The algorithm uses a pattern recognition scheme that quantifies the entropy in these 12 R-R intervals. If the entropy is more than a programmable threshold, then an AFib event is reported.

Signal Power Extractor The signal power extractor (SPE) is comprised of an FIR filter and digital envelope detection circuit used to measure signal power within a specific frequency band. The design and discussion of this accelerator is discussed in Chapter 4.

3.1.4 System Measurements

An ECG experiment was performed on a healthy human subject. In our in-vivo experiments, we used self-adhesive surface electrodes (Kendal Meditrace 535) to acquire ECG signals on the order of a few mVs. One electrode is attached to the chest, while the second (reference) electrode is attached to the abdomen. The two electrode leads are twisted before interfacing with the SoC to minimize interference. First, the chip was set to ECG raw data mode (consuming 397μW from the 1.35V V_{BOOST} node) (Figure 3.5).

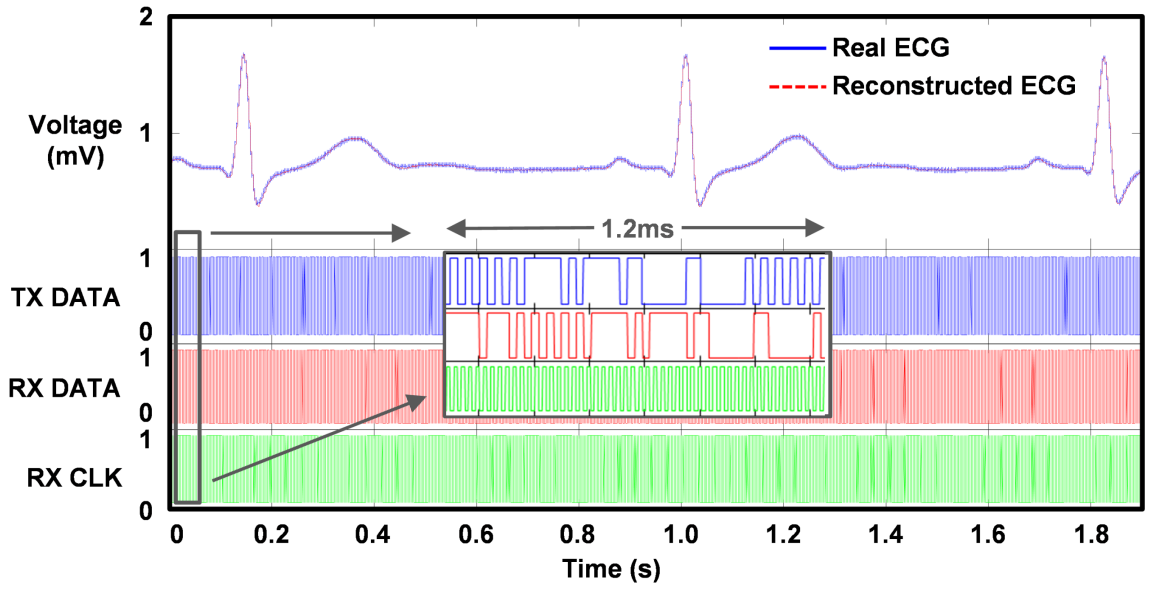


Figure 3.5: Measured system experiment showing correct data acquisition and streaming from the transmitter. Total power of 397μW prevents long use of this mode from harvested power

Our chip was paired to an unmodified TI CC1101 receiver and a wireless link was successfully established in the 433 MHz ISM band. The reconstructed ECG (dashed) closely matched the actual ECG. Also shown is the transmitted data, received data and clock waveforms. The zoomed-in section shows one 44b packet of data, including 9b header, 32b data, and 3b CRC.

Next, the chip ran an R-R interval extraction algorithm on the MCU and transmitted measured heart-rate every 5s operating from a 30mV supply voltage (Figure 3.6).

Every 5 seconds, V_{BOOST} is sampled to check for sufficient available energy, in which case the crystal oscillator is enabled for 20ms before the TX transmission, which takes 650μs including turn-on time and transmission of a 24-bit packet. The heart-rate extractor algorithm measures the R-R interval with a time resolution of (1/128) seconds (Figure 3.6). The extremely low duty-cycle of TX and crystal oscillator dramatically reduces the power consumption of the TX and crystal oscillator to negligible amounts.

In AFib detection mode, the R-R and AFib accelerators enable the TX and transmit

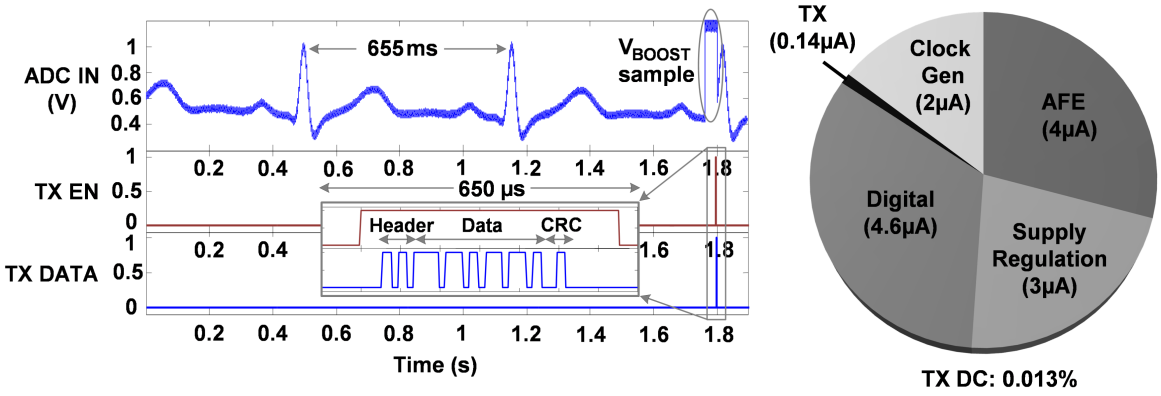


Figure 3.6: Measured system results for an R-R extraction algorithm. Measured system results are for acquiring ECG,extracting R-R intervals, and sending RF updates for R-R every 5 seconds. Total power in this mode is 19μW drawn from a 30mV input.

the last 8 beats of raw ECG (buffered in the DMEM) only when a rare AFib event occurs. Measurement results for the AFib demo are presented in Figure 3.7.

A pre-recorded set of AFib data from MIT-BIH database is used for this demo [30]. Detection occurs 12 R-R intervals after the inception of an AFib event. A pattern recognition algorithm determines if an AFib has occurred [29]. The total chip power in both the R-R and AFib modes is 19W, and the chip is powered exclusively from a 30mV harvested input.

Figure 3.6 presents a current breakdown of the R-R extraction demo. The current is nearly evenly distributed among different components, and selective transmission significantly reduces the average power consumption of the transmitter. Figure 3.8 shows the micrograph of the 2.5mm x 3.3mm batteryless BSN SoC (130nm CMOS), and Figure 3.9 gives a performance summary.

Table 3.1 shows a performance comparison table with recent BSN SoCs. This work is the first wireless bio-signal processing chip enabling battery-free operation.

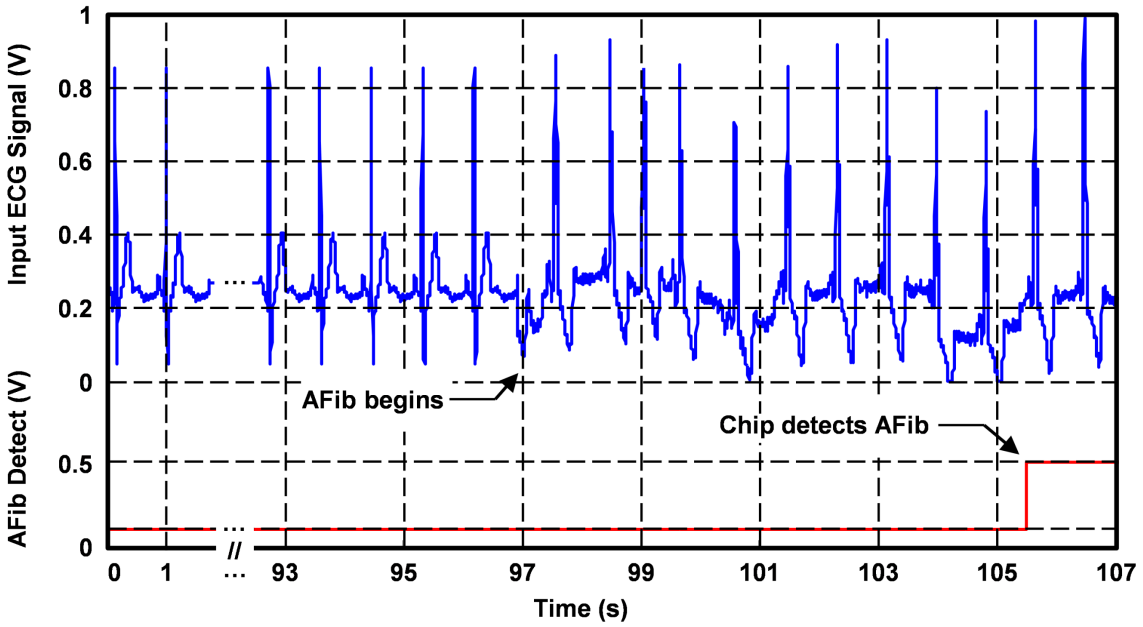


Figure 3.7: Measured system AFib demo experiment using R-R extractor and AFib accelerator. Normal and atrial fibrillation heart waveforms from MIT-BIH database. Last 8 beats of raw ECG are stored in DMEM and streamed over TX if AFib is detected. Total chip power in this mode is 19μW from a 30mV input.

3.2 Revision 1 [2]

The previous version of the BSN SoC was the first batteryless biomedical platform, but lacked architecture scalability, processing flexibility, digital interfaces, and a full transceiver. Revision 1 sought to dramatically increase the platform's flexibility without compromising ULP system operation.

3.2.1 Introduction

A 1 trillion node IoT will require sensing platforms that support numerous applications using power harvesting to avoid the cost and scalability challenge of battery replacement in such large numbers. Previous SoCs achieve good integration and even energy harvesting [34, 3, 35], but they limit supported applications, need higher end-to-end harvesting efficiency, and

[2]This section on revision 1 is based off of the work in [6]

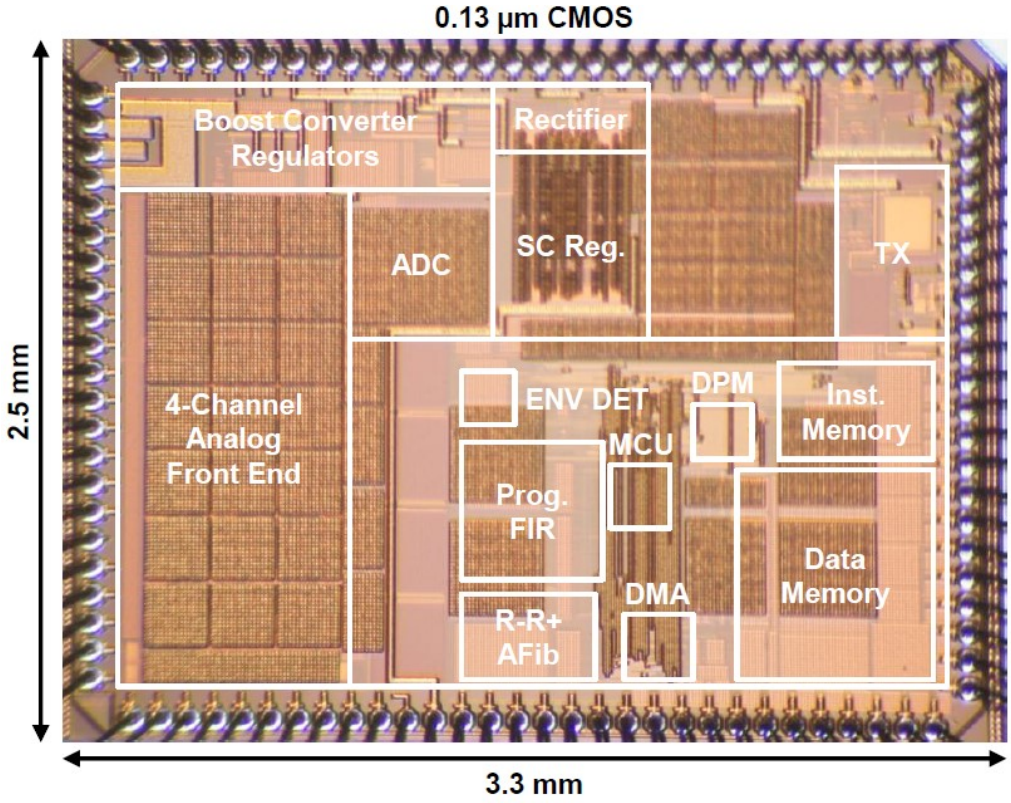


Figure 3.8: Version 1 chip micrograph

require duty-cycling for RF communication. This paper demonstrates a highly integrated, flexible SoC platform that supports multiple sensing modalities, extracts information from data flexibly across applications, harvests and delivers power efficiently, and communicates wirelessly.

The SoC shown in Figure 3.10 integrates a power management unit (PMU) with a boost converter for solar and TEG energy harvesting and a single-inductor, multiple-output (SIMO) DC-DC converter for high end-to-end self-powered efficiency. An asymmetric radio leverages ULP ultra-wideband (UWB) transmission and an always-on ULP receiver to reduce RF power significantly relative to prior SoCs for communication at higher data rates in an energy harvesting platform. The sensing interface includes a 4-channel (2W/channel) AFE [3] and SPI with variable voltage output pads (0.4-3.3V) for commercial sensor compatibility. The OpenMSP430 (OMSP) processor [36] and a suite of accelerators can execute numerous

Energy Harvesting		Supply Regulation		AFE (1 CH, ADC)		DSP		TX	
V_{in}	30mV	V_{unreg}	>1.25V	Current	4µA	Op. range	0.3-1.2V	Current	280µA
Kick-Start	RF@ -10dBm	$I_{quiescent}$	3µA	Supply	1.2V	MCU E/op @ 0.5V	1.5pJ	Supply	1V (LO) 0.5V (PA)
V_{out}	1.35V	$V_{analog,digital}$	0.5V, 1V	Gain	40-78dB	IMEM E/rd/Inst.	1.0pJ	Data-rate	200kbps
Efficiency	38%	V_{PA} V_{DVS}	1.2V 0.3-0.5V	$V_{ni,rms}$	<2µV_{rms}	FIR FOM*	0.27	E/b	0.8nJ/b
System Power				Bandwidth	0-320 Hz	AFib E/sample @ 0.5V	6pJ	Output Power	-18.5dBm
Current	14µA(R-R) (0.013% TX duty cycle)		294µA (Stream)	CMRR	>70dB	ENV DET E/sample	0.53pJ	Band	400 MHz MICS 433 MHz ISM
Supply	1.35V					Sub-V_T DVS: 0.3V to 0.6V	2 kHz-1.7 MHz	Modulation	BFSK
FIR FOM*: Power (nW) / frequency (MHz) / # of taps / input bit length / coefficient bit length									

Figure 3.9: Measured performance summary

biomedical and environmental signal processing algorithms (e.g. filtering, peak detection, histograms) combining ASIC energy efficiency and flexibility. A lightweight control unit (LCU) can manage chip data and node control while the OMSP is off, and uses a custom ISA and interrupt-driven programs to reduce the program size. The chips flexible clocking unit, containing a programmable ADPLL and configurable system clock, can drive the system clock from a low-power crystal. The digital blocks run in sub-threshold on a 0.5V supply from the PMU, while the radios use both the 1.2V and 0.5V rails.

3.2.2 Architecture

The chip uses two independent buses controlled by either the OMSP or LCU (for bus 1) or by the two-channel DMA (bus 2). The LCU can configure the OMSP as the main controller or as a bus peripheral that is used only for ALU or background operations. Since most data transfer occurs between the peripherals and the on-chip memories, the two-channel DMA (configured by OMSP or LCU) allows data movement on bus 2 in parallel with chip control on bus 1. Peripheral block wrappers decode and route bus data and manage independent block reset, clock gating, power-gating, and power mode settings. Most peripherals contain three power domains: always-on configuration registers, a bus decoder domain, and a block core logic domain (Figure 3.11). Programmable peripherals include a 4-channel, programmable FIR

	This work	[5]	[31]	[32]	[33]	[26]
Sensors	ECG, EMG, EEG	ECG	Neural, ECG, EMG, EEG	EEG	EEG, TIV	Temp., Pressure
Supply Voltage	30mV, -10dBm	1.2V	1V	1V	1.2V	0.4V/0.5V
E. Harvesting	Thermal, RF	✗	✗	✗	✗	Solar
Supply Reg.	✓	✗	✗	✗	✗	✓
AFE	4-channel	3-channel	1-channel	18-channel	4-channel	✗
Power Mgmt.	DPM, clock and power gating	clock gating	✗	✗	✗	power gating
GPP MCU	1.5pJ/instr @ 200kHz (8-bit RISC ISA)	✗	✗	✗	✗	28.9pJ/instr @ 73kHz (32-bit CORTEX-M3)
Accelerators	Programmable FIR, AFib, MCU, sig. energy extr., DMA, packetizer	ASIC DSP (4x SIMD), FIR, encryption, DMA	✗	ASIC DSP	FIR, packetizer, compression	✗
Memory	5.5kB (0.3-0.7V)	42kB (1.2V)	✗	✗	20kB (1.2V)	5kB (0.4V)
DVS	✓	✗	✗	✗	✗	✗
Digital Power	2.1μW	12μW	N/A	2.1μW	500μW	2.1μW (MCU)
TX (datarate)	200 kb/s	✗	100 kb/s	✗	1 Mbps (on-body link)	✗
TX P_{DC} (100% on)	160μW	✗	400μW	✗	2.8mW	✗
TX P_{OUT}	-18.5dBm	✗	-16dBm	✗	-6dBm	✗
TX band	402/433MHz	✗	402/433MHz	✗	20-40MHz	✗
Total Chip Power	19μW	31.1μW	500μW	77.1μW	2.4mW	7.7μW
Note on Total Power (includes)	1-channel AFE, 8-bit ADC, DSP (R-R extraction), and TX duty-cycled at 0.013%	AFE, 12-bit ADC, DSP (heart-beat detection)	1-channel AFE, 8-bit ADC, and streaming TX with 100% duty-cycle	18-channel AFE, 12-bit ADC, and DSP (EEG feature extraction)	4-channel AFE, 10-bit ADC, DSP (data compression, FIR), SRAM, TX at 5% duty cycle	Data acquisition, DSP (DFT), storage in SRAM
Technology	130nm	180nm	130nm	180nm	180nm	180nm

Table 3.1: Version 1 performance comparison with state-of-the-art BSN nodes.

filter, a CORDIC, 16-point complex FFT/IFFT, two timer modules with capture/compare, a multiplier, and heart rate (R-R) AFIB detection. This suite of hardware accelerators supports flexible processing for a wide range of applications. The SoC has a 4 kB data memory and three, 2kB memories: Tx buffer, LCU instruction memory, and OMSP instruction memory. All SRAMs use an 8T sub-threshold bitcell and a read before write scheme. The memories

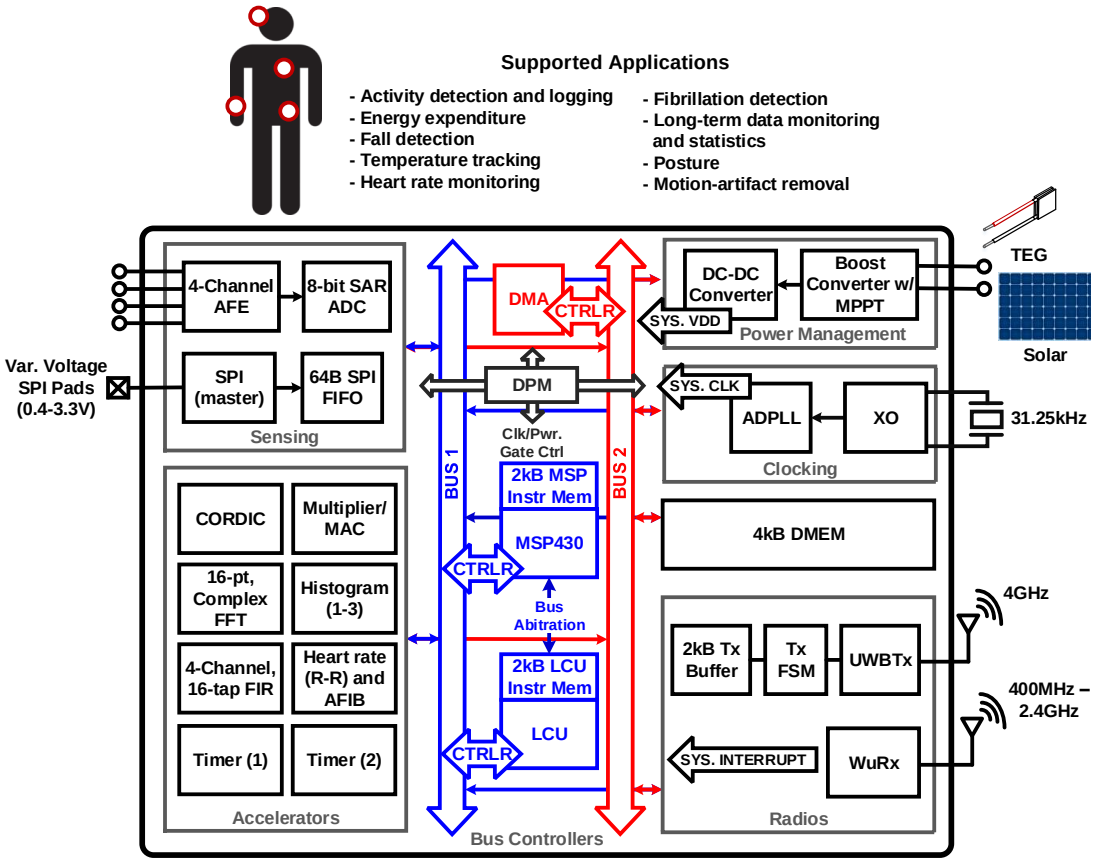


Figure 3.10: System block diagram for the SoC, highlighting supported applications and interfaces

are partitioned into 1 KB, independently power gate-able banks and operate down to 0.35V.

3.2.3 System Clocking

An on-chip 187.5kHz to 500kHz ADPLL uses a dual-loop architecture that eliminates the divider to consume 300-600nW from 0.5V with jitter <0.1% in 0.07mm2. The entire ADPLL was implemented using standard digital design flows and automatic place and route (APR). An integrated crystal oscillator (31.25 kHz) gives the reference to the ADPLL, and a digital clock module lets the LCU or the OMSP control the clock frequency and reset state for the SoC.

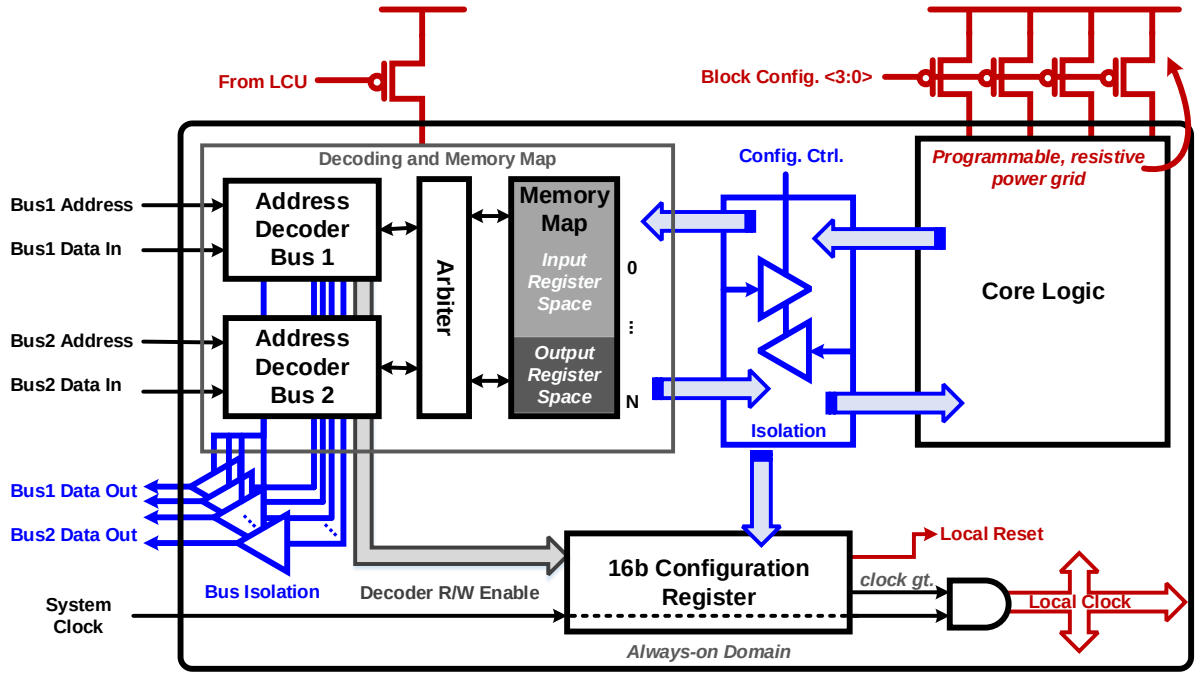


Figure 3.11: Block-level architecture of the peripheral units showing fine-grained clock and power gating.

3.2.4 Energy Harvesting

The integrated energy harvesting and PMU (Figure 3.12) couples a boost converter and SIMO DC-DC converter to achieve a measured 74.9% end-to-end (boost and DC-DC) peak efficiency for the 1.2V output (65.7% end-to-end efficiency for the 0.5V output) for a 100A load while harvesting from indoor solar at the maximum power point (MPP) input voltage of 1.3V. The system can harvest energy from a photovoltaic cell (PV) (e.g., with an open circuit voltage of 1.7V) or from a TEG. The boost converter [37] can harvest energy from a VIN as low as 10mV and charges V_{CAP} to up to 1.4V. An integrated MPP tracking circuit tunes the input impedance of the boost converter to extract the maximum energy from the ambient source (either the TEG or PV). A SIMO DC-DC converter regulates V_{CAP} to a 1.2V and 0.5V supply to improve end-to-end efficiency over [3], which used on-chip LDOs. The system achieves peak end-to-end efficiency at the MPP of the solar cell, which is measured to be 1.3V. An integrated digital power-management unit (DPM) checks the available energy

on the capacitor before allowing system operations, and performs system mode changes or shutdown if available energy is low by overriding the block level power mode configuration.

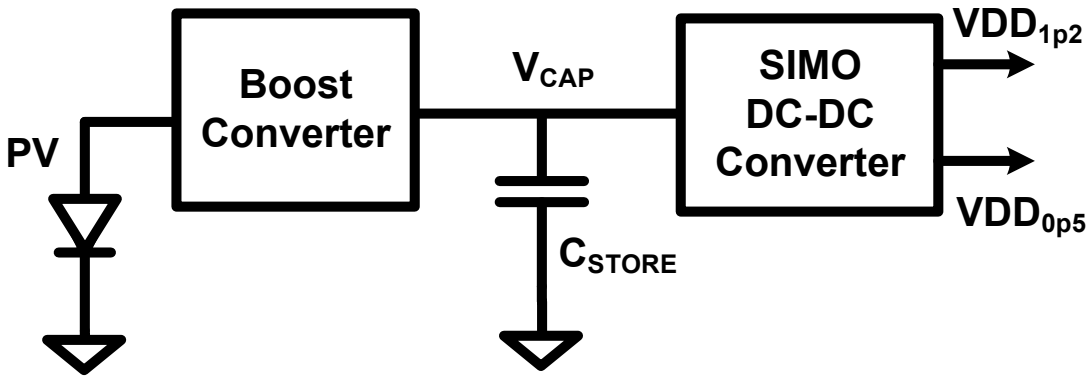


Figure 3.12: Architecture of the integrated PMU for high end-to-end self-powered efficiency

3.2.5 Wireless Transmission

Despite the frequent use of heavily duty-cycled radios in low-power applications, maintaining high data-rate RF communication is needed in many IoT applications. In a BSN scenario using a star topology network (i.e. all nodes communicating to one aggregator), data is more frequently transmitted to the base station than it is received from the base station. To support this, we use an asymmetric RF architecture to minimize power on node (Figure 3.13).

The UWB transmitter uses OOK modulation with a data rate of 187.5kbps and a center frequency of 3.99GHz with a 500MHz bandwidth. It consists of a pulse-width generation circuit that uses two separate signal paths with different variable delays to create a short pulse, which is then used to enable/disable a ring oscillator, creating the UWB pulses. A Class AB power amplifier buffers the signal onto the antenna. The Tx FSM has a 2kB sub-threshold memory buffer that stores the raw data as well as any synchronization and preamble headers the base station receiver might need. The Tx FSM clocks the data out of the memory and serializes it before transmission, and the FSM can append a timestamp to the serial data

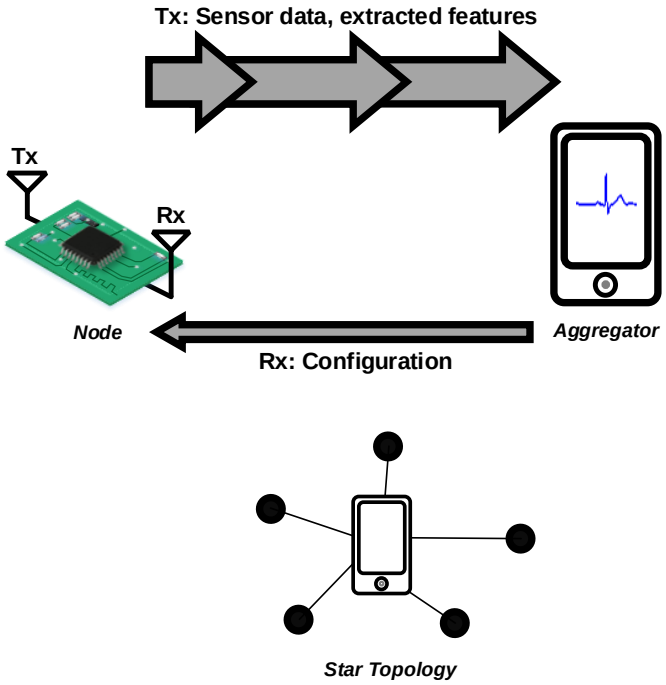


Figure 3.13: Motivation for asymmetric transceiver design.

as needed. The CDMA wakeup receiver (WuRx) [38] has four separate interrupt outputs, one hardcoded to each of the OMSP and LCU, and two other programmable outputs. Each interrupt is triggered by a unique 15-bit OOK modulated Kasami code. The WuRx can also be used to receive data packets at 8kbps. The architecture and results for the radios is shown in Figure 3.14.

3.2.6 Measured Results

The chip was tested end-to-end for motion capture with an ADXL362Z accelerometer (over SPI) while powered from indoor solar by the PMU. The chip consumes 6.45μW while streaming raw motion data wirelessly over UWB. Figure 3.15 shows the flexible data path capabilities of the SoC for motion capture applications and transmission of accelerometer data using the on-chip SPI with variable voltage pads.

Table 3.2 compares the SoC to prior work. To date, this work has the highest level of

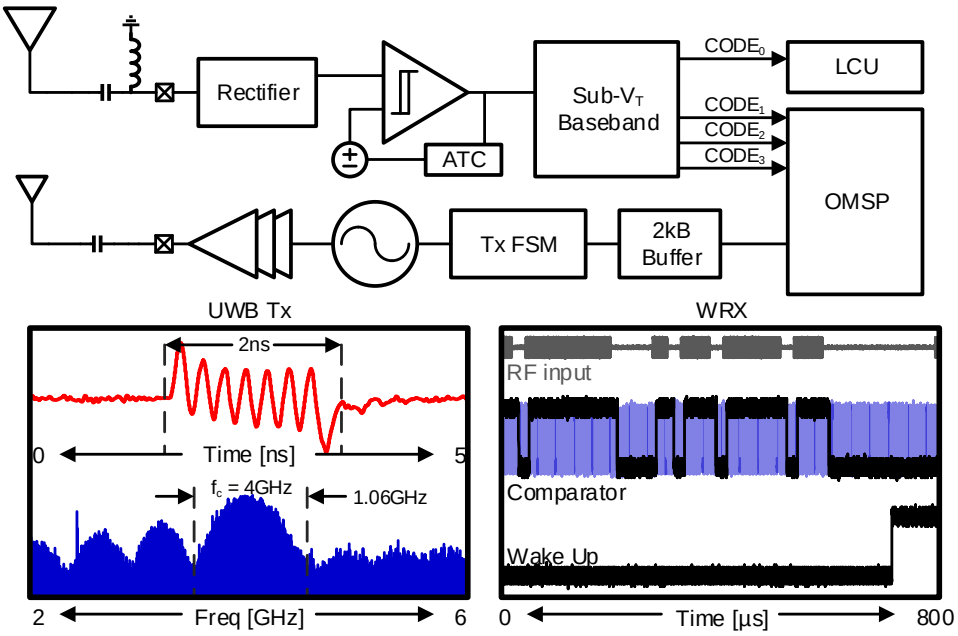


Figure 3.14: Transceiver architecture and performance

integration, including energy harvesting and a full transceiver, for the lowest power. This work also has the highest energy harvesting / regulation efficiency, and achieves lower RF connectivity power by 38X. The carefully integrated ULP components on this SoC support numerous IoT applications on a self-powered platform. The chip micrograph is shown in Figure 3.16, indicating the high level of integration between system modules.

e comparison with state-of-the-art BSN nodes]Performance comparison with state-of-the-art BSN nodes.

3.2.7 Individual Contributions

My contributions to this platform are listed here. As there were two sub-revisions of this platform before the final, working chip, the first iteration will be denoted as 1.1 and the final revision as 1.2.

1. **Interfacing, Organization, and Integration**: To create an SoC with such a high-level of integration and many collaborators, large amounts of organization and interfacing

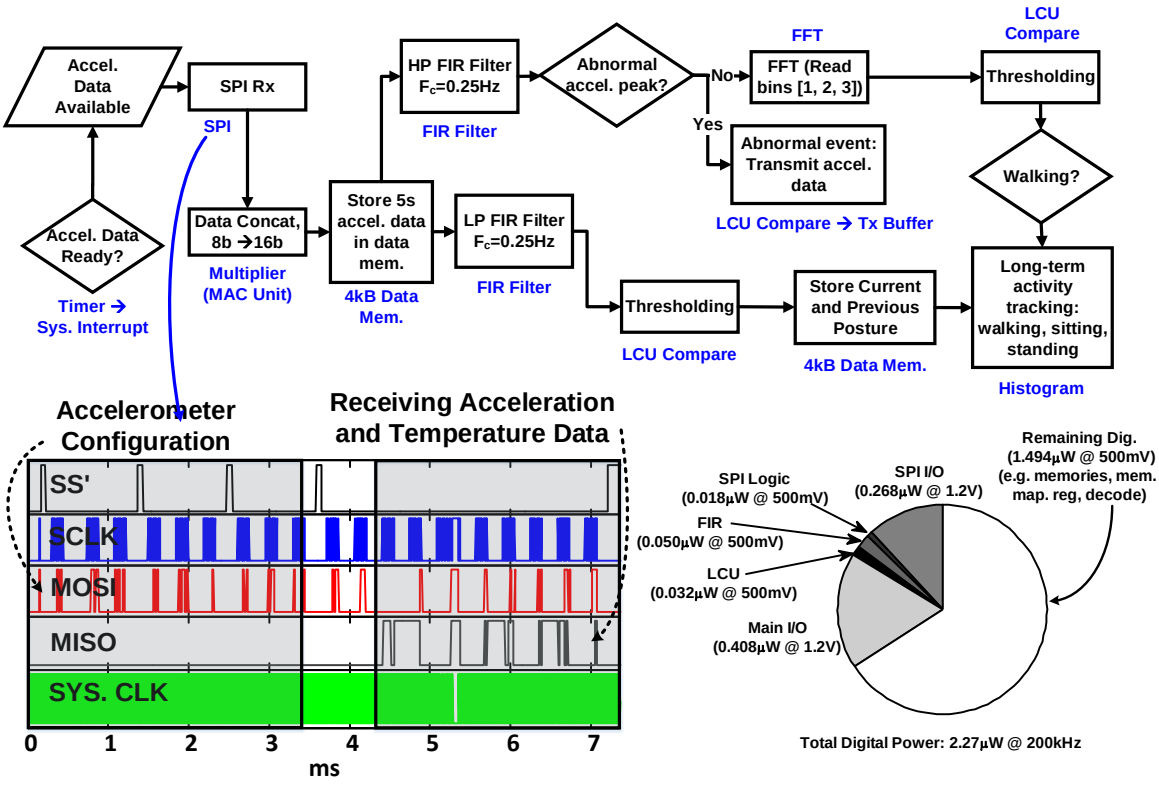


Figure 3.15: Supported system demo block diagram and power breakdown using the digital SPI interface and ADXL362Z accelerometer.

had to occur to ensure a working node. This included creating digital interfaces for mixed-signal and analog blocks to be compatible with our bus architecture and keeping track of memory-mapped addresses taken by each peripheral to prevent overlap and overflow.

2. **System-Level Architecture**: For revision 1, we made the decision to switch from a global decoding scheme implemented using transmission gates to a local, memory-mapped decoding scheme. For revision 1.1, I implemented the bus decoders and memory-map structure.

3. **Tx FSM**: The Tx FSM has a 2kB memory array that stores the raw data to be transmitted and is written by the processor. Raw data includes any synchronization and preamble headers the receiving base station receiver might need. The Tx FSM

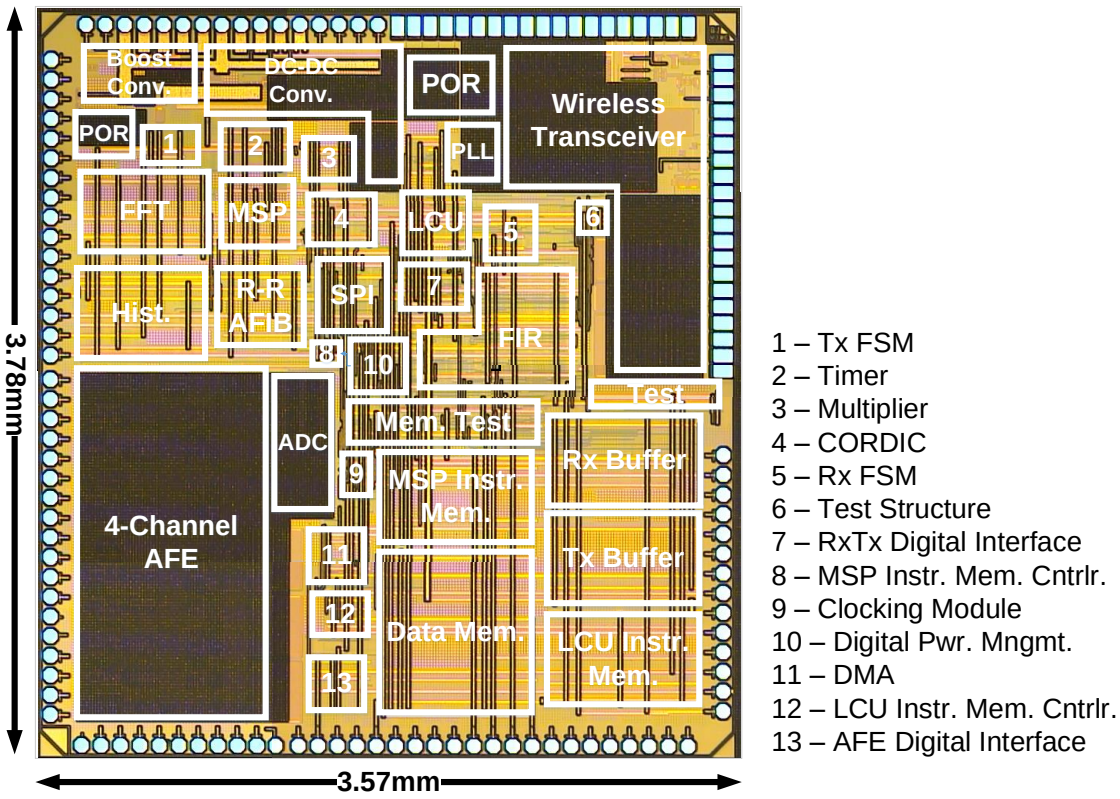


Figure 3.16: Chip micrograph

is responsible for clocking the data out of the memory array and serializing it before sending it to the UWB transmitter. The Tx FSM also has the option to append a timestamp to the serial data before it is sent to the transmitter.

4. **Rx FSM**: The Rx FSM processes the incoming bitstream from the narrowband receiver (NBRx) and writes it into a receive memory buffer where it can be accessed by the main processor. At the beginning of a packet the NBRx synchronization header is received and the FSM controls the demodulation PLLs 17-bit mux select value to synchronize with the data. Following the sync header is the 802.15.6 PLC preamble. According to 802.15.6, two different preambles can be sent, but this FSM only looks for one specific preamble. After the preamble the Physical Layer Convergence Protocol (PLCP) header is sent which is typically used to convey physical layer information. However, since the

NBRx does not have all the configuration options available, this information acts as another preamble. The data information that follows the PLCP header is then stored in its raw form in a 2kB memory array. Any post-processing or Media Access Control (MAC) level functionality is performed by the microcontroller.

5. **Timers**: This includes two, independently controlled timers for general purpose use. The block can be programmed to increment or decrement, rollover, and have a divided-down clock input for increased range. Each timer has an interrupt tied to the LCU and MSP when the timer has finished.

6. **SPI master**: This block is a modified version of the OpenCores Simple SPI [39] with additions for bus interfacing, added slave select signal, and increased FIFO size. It includes a 128 entries deep read and write FIFO, options for interrupts, enables, clock phase, and clock polarity. Data can be moved from a memory-mapped register to write to the FIFO before transmission, and data can be read through a memory-mapped register from the read FIFO after transmission is complete. SPI logic was provided by OpenCores, but used a Wishbone Interface. A decoder wrapper and FSM was added to these blocks for easy data transmission/receives using minimal bus instructions.

7. **DMA**: For this platform, the bulk of data needs to be transferred between the various peripheral blocks and the on-chip memories. This data transfer is more efficient when it can be done without constantly engaging the bus 1 controller. I modified the DMA HDL for synthesis on revision 1.1 and implemented simultaneous, dual-channel operation for revision 1.2.

8. OpenMSP430 HDL modifications for bus arbitration between the LCU and MSP430.

9. **Accelerators**

   (a) **FIR Filter**: A four-channel, 16-bit FIR filter, with each channel having up to 16-taps. The number of coefficients, number of active channels, and number of

parallel filters are programmable. Each channel can be independently clock gated when not in use.

(b) **16-pt FFT/IFFT**: This is a 16-pt, complex FFT and IFFT. This block re-purposes a single radix-2 butterfly per clock cycle and uses in-place memory addressing to reduce the memory requirements.

(c) **Clock Arbiter for rev 2.1**: Clock arbiter for dividing down the system clock for blocks with lower frequency requirements

(d) **Histogram**: This block can construct up to three histograms of data over time. The bin thresholds and number of active bins are programmable. Each histogram can contain up to 32 bins. One histogram has the option of only recording data when its above a certain threshold or stopping data acquisition once a certain number of events have been recorded.

The system architecture and composite blocks of this chip are a motivator for the design methods discussed in this dissertation, and the proposed designs and ideas work primarily within this scope.

3.2.8 Conclusions and Discussion

Both versions of the BSN SoC are examples of systems that have strict power constraints to enable batteryless operation and long lifetimes. These platforms illustrate the point that every nW of power is significant when running an application from harvested energy. Version 1 showed that for an R-R Afib demo, that's heavily reliant on DSP for extraction, 1/3 of the current consumption is consumed by the digital. Although this number includes regulation and leakage from the memories, the digital processing is still a significant portion. Power gains are achieved through operating all digital logic in the subthreshold region leading to quadratic reductions in the digital power due to direct DSP methods. Revision 1 showed that very little of the power budget is due to digital processing, but is in fact dominated by the

transmission power. In both scenarios, indirect power reductions using DSP can be leveraged to reduce the power contributed by the radios and memories. Aggressive duty-cycling is already used to keep the radios on for short windows of time leading to low average powers, but Chapter 5 will discuss another technique for indirectly reducing the system power by bypassing the memories to reduce overall leakage. Although the revision 1 node aimed to target a wider set of applications for the IoT, key interfaces and processing capabilities are missing that will be discussed in Chapter 7.

		This work	[34]	[10]	[35]
	Sensor Interfaces	Analog (ECG, EEG, EMG), Digital (SPI)	Analog (ECG, bio-impedence), Digital (SPI, I2C, UART)	Analog (ECG, EMG, EEG)	Analog (ECG)
Digital	**MCU**	MSP430	ARM Cortex M0	8-bit PIC	ARM Cortex M0
	Peripherals	DMA, FIR, FFT CORDIC, Timer, Histogram, Multiplier, R-R AFIB	DMA, Matrix-Multiply-Accumulate	DMA FIFO, FIR, Envelope Detector, R-R AFIB	FIR, FFT
	On-Chip Memory	12 kB (0.35 V - 0.7 V)	128 KB	5.5 kB	3.7 kB
	DVS	✓	✗	✓	✗
	Digital Power Management	✓	✗	✓	✗
	Flexible Clocking	ADPLL (187.5 to 500 kHz with /1/2/4/8)	MCU Clock (1-20 MHz)	200 kHz (with /1/2/4)	10 kHz
	Digital Power (Active)	2.27 μW	120 μW	2.1 μW	45 nW
PMU	**Integrated PMU**	✓	✗	✓	✗
	Energy Harvesting	Solar, TEG	✗	TEG	✗
	End-to-End PMU Efficiency	74.9%	✗	38%	✗
	Regulated Output Voltages	1.2 V, 0.5 V, Variable	✗	1.2 V, 1.0 V, 0.5 V (2)	✗
	SIMO Regulation	✓	✗	✗	✗
	Boost Voltage	10 mV	✗	30 mV	✗
AFE	**ADC Resolution**	8	10	8	8
	ADC Sampling Frequency	128/256 Hz	500 Hz	128/256 Hz	500 Hz
	AFE Channels	4	3	4	1
Radios	**Integrated Transceiver**	✓	✗	✗	✗
	Tx Band	2400 MHz - 2480 MHz (3.99 GHz center)	✗	402 / 433 MHz	✗
	Tx Data Rate	187.5 kbps	✗	200 kbps	✗
	Tx Output Power	-28.9 dBm	✗	-18.5 dBm	✗
	Tx Power	4.18 μW	✗	160 μW	✗
	Rx Center Frequency	400-2400 MHz	✗	✗	✗
	Rx Data Rate	7.8125 kbps	✗	✗	✗
	Rx Power	112 nW	✗	✗	✗
	Rx Energy/15-bit Wakeup	161 pJ/bit	✗	✗	✗
	Technology	130 nm	180 nm	130 nm	65 nm
	Die Area	13.49 mm^2	49.00 mm^2	8.25 mm^2	3.32 mm^2
	Max. Voltage	1.4 V	1.2 V	1.2 V	0.6 V

Table 3.2: Revision 1 performance comparison with state-of-the-art BSN nodes.

Chapter 4

ULP Signal Power Extractor [1]

This chapter presents a synthesizable, subthreshold, four-channel signal band power extractor for the version 1 batteryless body sensor node SoC presented in Chapter 3. The power extractor consists of a programmable 30-tap finite impulse response (FIR) filter and signal power circuit (SPC). The filter uses a serial, resource-shared architecture to reduce area, leakage, and power. The FIR supports a programmable number of taps, number of active channels, and coefficient register data. The SPC uses power-of-two arithmetic for reduced complexity, area and power. The design was synthesized in 130nm CMOS, and consumes 34nW (32nW for FIR, 2nW for SPC) per channel at 350mV and 29kHz.

4.1 Introduction

Wireless sensor nodes and other ULP systems demand energy efficient signal processing to meet their stringent power requirements. For example, a BSN SoC with four input channels for ExG data, ADC, MCU, radio, power management, and energy harvesting runs entirely on power harvested from body heat with no battery [3]. To enable this, the chip must consume an average power less than the power harvested, which is typically in the 30-50μW range. When extracting heart rate from ECG and sending regular RF updates, the entire chip

[1] This section is based off of the work in [40]

consumes just 19μW [3]. Since the chip harvests power, the processing blocks are power limited to keep the total average chip power below the harvested power and avoid depleting energy on the storage capacitor.

Filtering is frequently required for processing ExG data, and signal spectra power analysis is used for various neural applications, so we present a custom ULP FIR accelerator block and SPC integrated on the SoC in [3]. A primary use of the power extractor on the SoC is processing EEG signals to determine the amount of cortical neuronal activity in the brain. From individual electrodes, the band power can be determined by filtering the data and averaging the square of the data over a window of time [41]. The majority of the signal power of EEG signals is concentrated at frequencies <200Hz and can determine neuronal activity such as processing or movement, sleeping events, and seizure prediction as seen in Table 4.1 [42].

Neurological State	**Frequency Band**
Visual processing/motor planning	8-12 Hz (α)
Awake and Alert	18-26 Hz (β)
Consciousness/Awareness (Present)	70-100 Hz (γ)
Consciousness/Memories (Past)	30-50 Hz (low-γ)
Light Sleep	4-7 Hz (θ)
Deep Sleep	0.5-3 Hz (δ)

Table 4.1: EEG frequency bands of interest

By filtering the EEG data prior to processing using the SPC, the amount of signal energy within a single frequency band can be determined and used to classify specific brain activities if the energy passes a known threshold. Normal neuronal activity from an awake subject can be extracted from four key EEG frequency bands: α (8-12 Hz), β (18-26 Hz), low-γ (30-50 Hz), and γ (70-100 Hz). To begin to classify sleep, the δ (0.5-25 Hz) and θ (4-7 Hz) bands are also used. The filter is programmable to allow for subject-specific frequency bands or to handle general purpose filtering on-node. The presented FIR was designed considering these bands of interest, but is not limited to this application.

Other power extractor circuits have also been presented for EEG/ECoG systems that consume more power and area due to an analog or mixed signal implementation as in [43] and [44] or the use of more versatile, but costly spectral density algorithms as in [45]. As the filter consumes >90% of the power in this implementation of the power extractor, reducing power within the FIR was of high importance and was completed through the use of a serial architecture and subthreshold operation seen in Figure 4.1 with corresponding timing diagram.

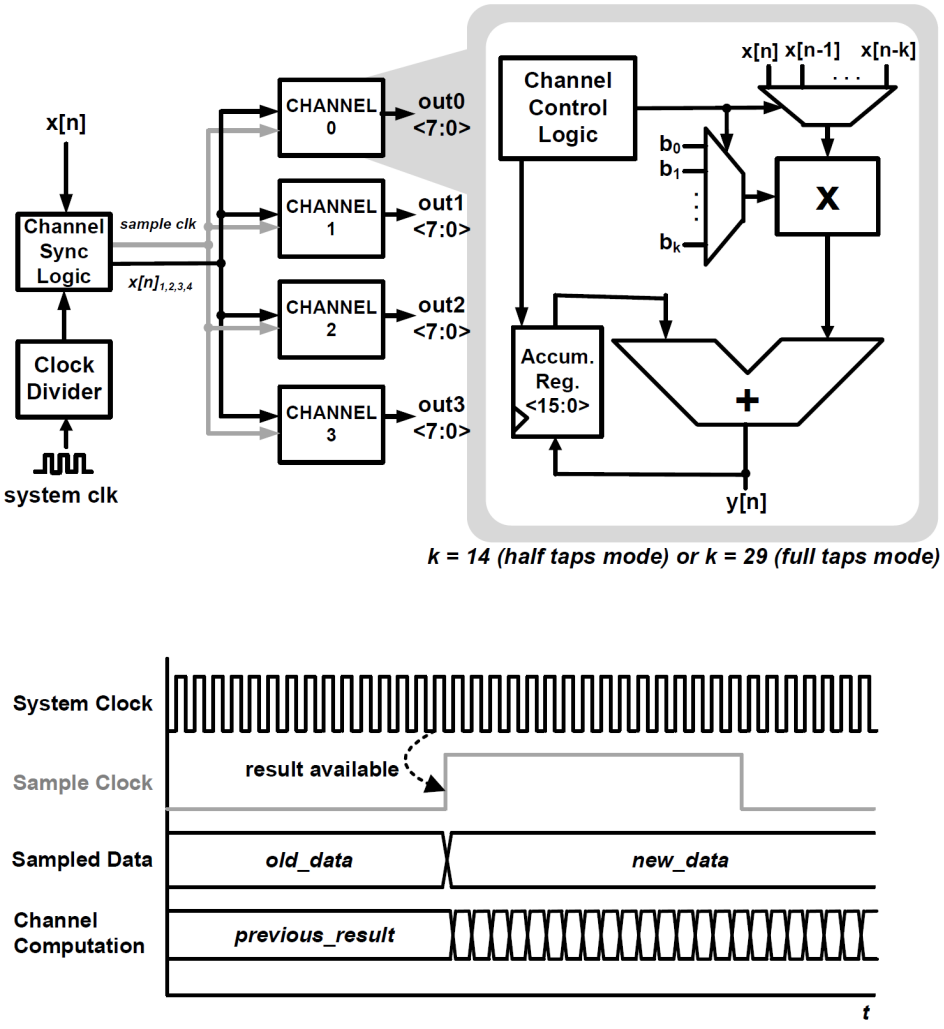


Figure 4.1: The filter resource-shared architecture of the 4-channel FIR including detailed diagram of the channel including the timing diagram of FIR filter showing serial operation

Other subthreshold filters have been proposed that use body-biasing techniques to reduce

the effects of variation as in [46] or boost a subthreshold supply voltage to the super-threshold region so that the circuit no longer operates in subthreshold as in [47]. We avoided body biasing to make our synthesizable design process portable and kept a subthreshold supply voltage since it provided ample performance for our set of BSN applications. The power extractor circuit used power-of-two arithmetic to eliminate the need for multipliers and divider circuits for reduced power as seen in Figure 4.2.

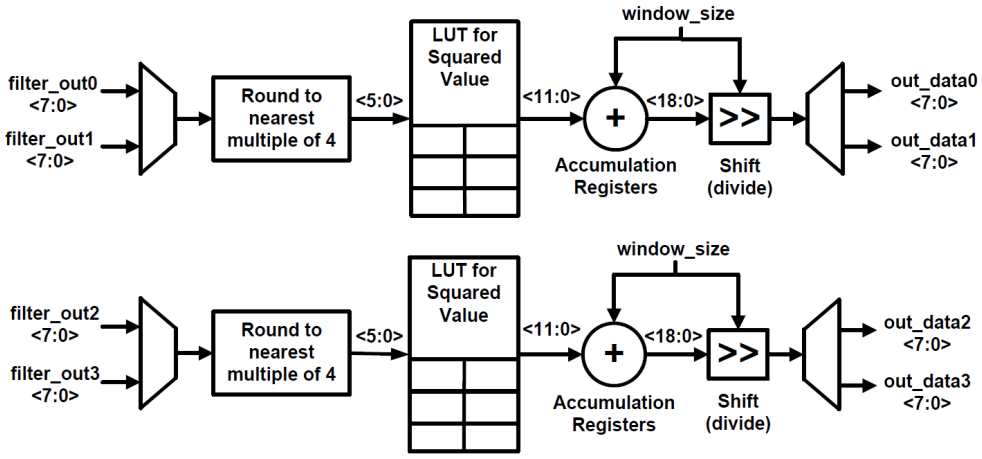
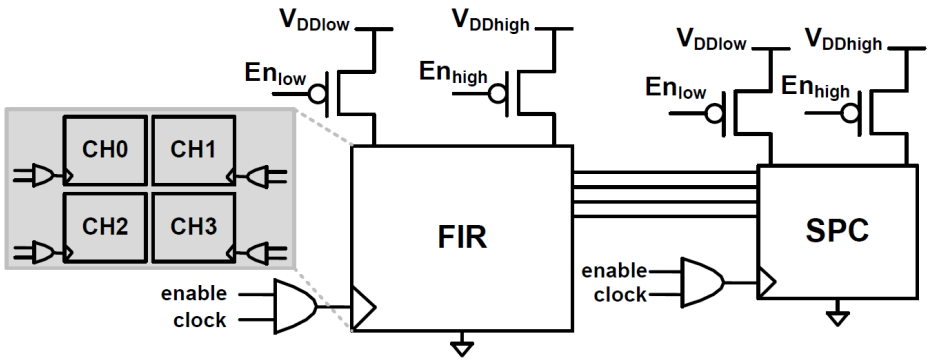


Figure 4.2: Block diagram of the SPC

4.2 Design Considerations

The effects of variation on the threshold voltage, V_T, are pronounced when operating in the subthreshold region due to an exponential dependence of the drain current of the transistor on V_T. To mitigate the effects of this on the performance of the power extractor, static CMOS gates with short stacks were used throughout the design to provide robustness. To avoid ratioed circuits, SRAM cells for storing data and coefficients were avoided in favor of standard cell registers using high-V_T devices for reduced leakage and to maintain a synthesizable design [48].

As the filter and power extractor are not always in use by the reconfigurable datapath on chip, they required extremely low-leakage during idle periods. To reduce leakage energy, PMOS headers were used to cut off the voltage supply to the gates during idle mode. Appropriate header sizes were chosen based on the current draw of each circuit. Based on this, a header width was chosen that prevented <10% supply voltage droop in simulation, but not any larger to reduce leakage when off. The filter and extractor can both be clock gated at the block or individual channel level. This prevents excess switching energy due to an active system clock when these blocks are not in use as seen in Figure 4.3.



	Savings
Headers	10x-15x
Clock Gating	2x-4x

	Clock Cycles	Energy
FIR	30	33pJ
MCU	880	6.3nJ
Improvement	29x	191x

Figure 4.3: PMOS headers and clock gating for power savings. Table of measured energy savings for the FIR (energy bottleneck of the system) compared to using the on-chip MCU

4.3 FIR Filter Design

Filtering topologies are generally divided into two categories: Finite Impulse Response (FIR) and Infinite Impulse Response (IIR). The latter is often defined based on the outputs

dependence on past and present inputs (i.e. recursive), while FIR is dependent only on the present input as seen in Figure 4.4.

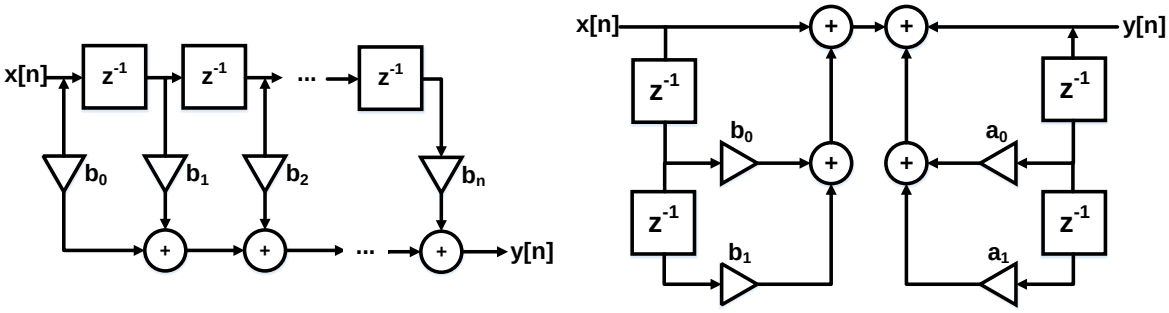


Figure 4.4: (left) Direct-form FIR filter topology (right) Direct-form IIR filter topology

To compute each output, $y[n]$, the algorithm must complete one iteration where the delay between the system receiving the input and producing a corresponding output is the block latency [49]. These systems require one multiplication and addition operation per tap, and the number of taps used determines the accuracy of the system. FIR filtering is most commonly seen in the literature due to a guaranteed system stability and linear phase characteristic. IIR filtering has the advantage of needing fewer computing resources to achieve the same performance as an FIR, but can become unstable due the feedback structure. IIRs also have a nonlinear phase characteristic that is not ideal for certain applications such as speech/audio, although methods to linearize the phase have been presented in the literature as in [50].

To support concurrent ExG processing on data from the four analog input channels on chip, the architecture of the FIR has four independent channels as seen in Figure 4.1. The SoC uses a 200kHz XTAL for the system clock to modulate data in the Medical Implant Communication Service (MICS) band radio and for a digital clock. Since the typical data sampling rate for ExG is much lower than 200kHz (<1024 Hz for our chip), we use a serial architecture [17] versus the traditional direct-form architecture for a FIR filter to reduce power, area, and leakage. The direct-form FIR architecture computes the result in parallel using as many adders and multipliers as there are taps. This method results in a much higher throughput than required for this application space as well as a large area penalty

for up to 30 taps replicated over four channels. Resource reuse of the arithmetic units was achieved through using the faster 200 kHz clock for serially processing data between receiving input samples. This allowed for a higher utilization of arithmetic units per sample, resulting in lower leakage from inactive or excess circuitry when the design is operating or not on the datapath. At every rising edge of the sampling clock, a new input sample is received, processed at the faster clock rate, and the result is computed in a fraction of the sample period as seen in Figure 4.1.

Each channel contains one 8-bit Baugh-Wooley multiplier, one 16-bit ripple-carry adder, coefficient registers, and a small filter controller for channel synchronization and state retention of the channels. During active operation, each channel is individually clock-gated for any remaining cycles after the result is computed to further reduce energy. Sharing arithmetic units within each channel reduced the overall area by 6-12X/filter compared to prior work [46, 47] and by 12X/channel compared to a traditional parallel architecture. The smaller area also reduces leakage, which helps reduce energy drawn from the off-chip storage capacitor especially during idle periods.

The filter supports programmability by the SoCs MCU for several modes of operation. A programmable filter is important due to the volatility of energy on the storage capacitor during power harvesting. The on-chip power management controller can reduce the number of taps used for filtering or the number of active channels to reduce power in the circuits based on available energy. A 30-tap filter met the accuracy specifications for this set of applications as this was the point that the magnitude responses for bandpass filters in the targeted frequency ranges (for non-sleep bands) were attenuating frequencies below the lower cutoff frequency as seen in Figure 4.5.

When high-throughput and energy are more critical than accuracy, a 15-tap mode is available. Each stream of input data can be filtered using one channel or two simultaneously with different filtering coefficients (e.g. EEG bands). The frequency spectra of a filtered EEG signal for an awake subject are show in Figure 4.6.

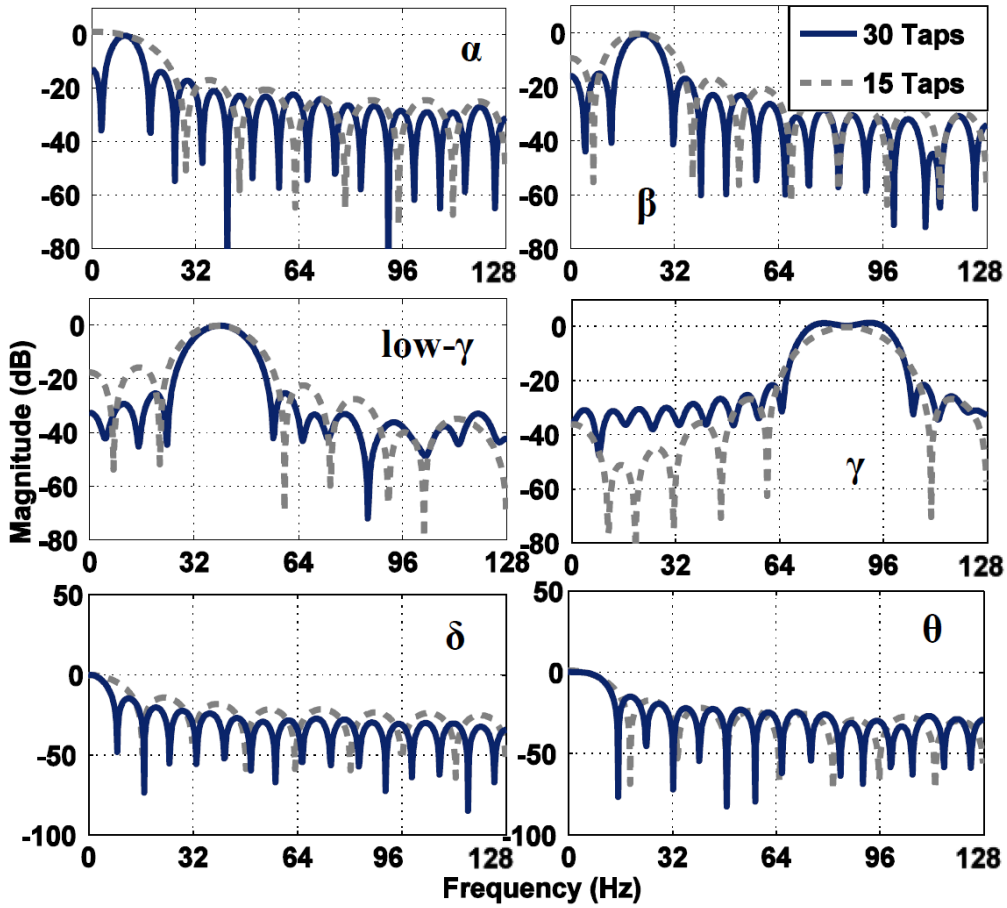


Figure 4.5: FIR frequency response for the α, β, low-γ, γ, δ, and θ bands of EEG for tap lengths of 15 and 30 and a sampling rate of 256 Hz.

The FIR filter coefficients used for testing were found through MATLABs filter toolbox using a Kaiser window with β=0.5. The relationship between energy/sample and number of taps in the FIR is linear in our serial design as each tap requires one additional multiply-accumulate operation than the previous, so varying the number of taps trades off energy with fidelity using this architecture. This also allows the processing of different types of data with different fidelity requirements.

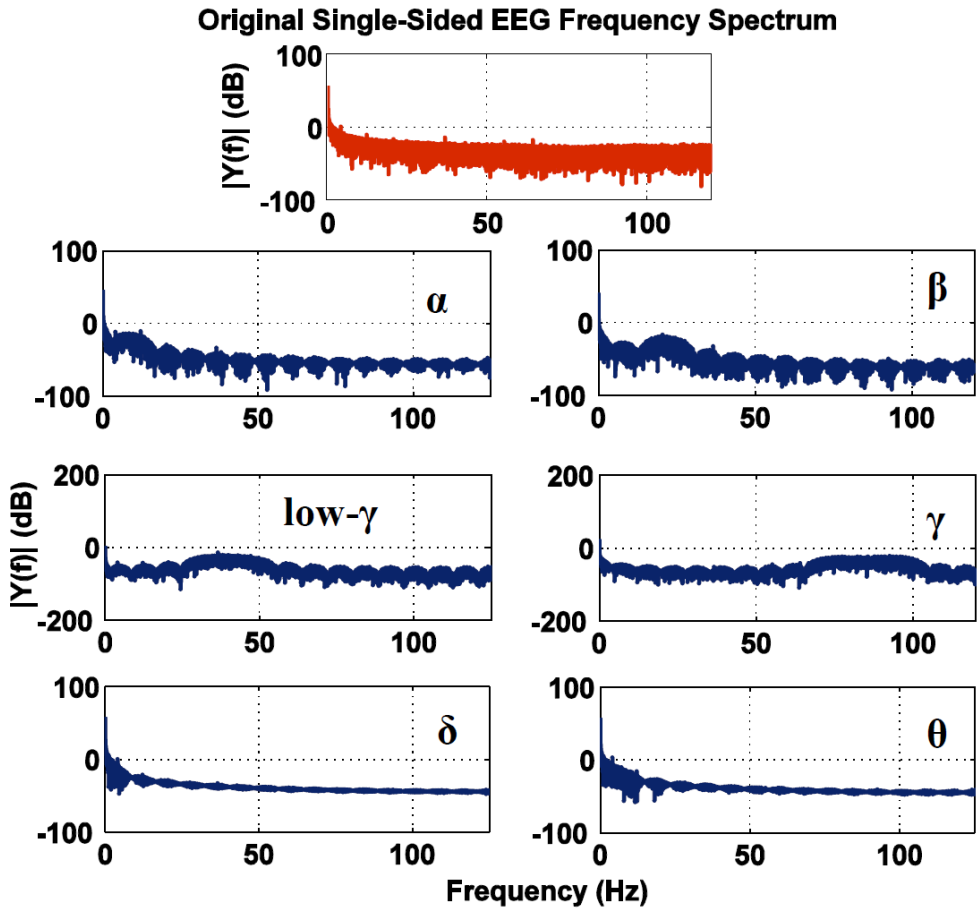


Figure 4.6: Original EEG motor activity signal spectrum [51] and measured filtered waveforms using 30 taps.

4.4 Signal Power Extractor Design

The SPC receives output data from the FIR filter and computes the average signal power within a specific frequency band. This block has four input channels corresponding to the channel outputs of the filter and can save state for each channel during a channel switch from the ADC. The extractor also has a programmable summing window size and number of active channels. The circuit directly implements the signal power equation shown in 4.1,

$$p_x = \frac{1}{N} \sum_{n=0}^{N-1} |x[n]|^2 \tag{4.1}$$

where p_x is the average signal power for a signal, x, and N is the summing window size. This is an alternative to directly computing the power spectral density of the signal that gives energy per hertz as seen in 4.2, where ω is the angular frequency. Power spectral density is a robust way for determining EEG signal characteristics as a function of frequency [41].

$$\varphi(\omega) = \left| \frac{1}{\sqrt{2\pi}} \sum_{n=\infty}^{-\infty} x[n] e^{-j\omega n} \right|^2 = \frac{F(\omega) F*(\omega)}{2\pi} \tag{4.2}$$

The equation in 4.2 is equivalent to finding the Fast Fourier Transform (FFT) of the signal, x, and multiplying by the complex conjugate of the result. The FFT is a computationally expensive operation that would also require post-processing for determining the signal power.

Implementing the signal power equation in 4.1 directly requires a squaring circuit, accumulator (adder), and a division circuit. Since these are costly operations to replicate over 4 channels, computation complexity was reduced by working with the data in powers-of-two, which reduces the division operation to a series of shifts (Figure 4.2). The window size is a programmable input and can be set to any power-of-two in the range of 4 to 128. To replace the multiplier required for the squaring operation, data were rounded to the nearest power of 4, and the squared results came from a lookup table using high-V_T standard cells to reduce leakage. The rounding reduced the number of bits required during data transformation as the lower two bits were always 0. The error due to this implementation can be seen in Figure 4.7 with an example EEG signal for an awake subject performing random 1D movements of the left and right hand [51].

The SPC used a window size of 128 to process this data and shows higher relative error in frequency bands that dont contain a large amount of the overall signal power and low error for the frequency bands describing motor planning and awareness. The absolute error between the ideal and the power-of-two methods is small enough to still allow for accurate detection.

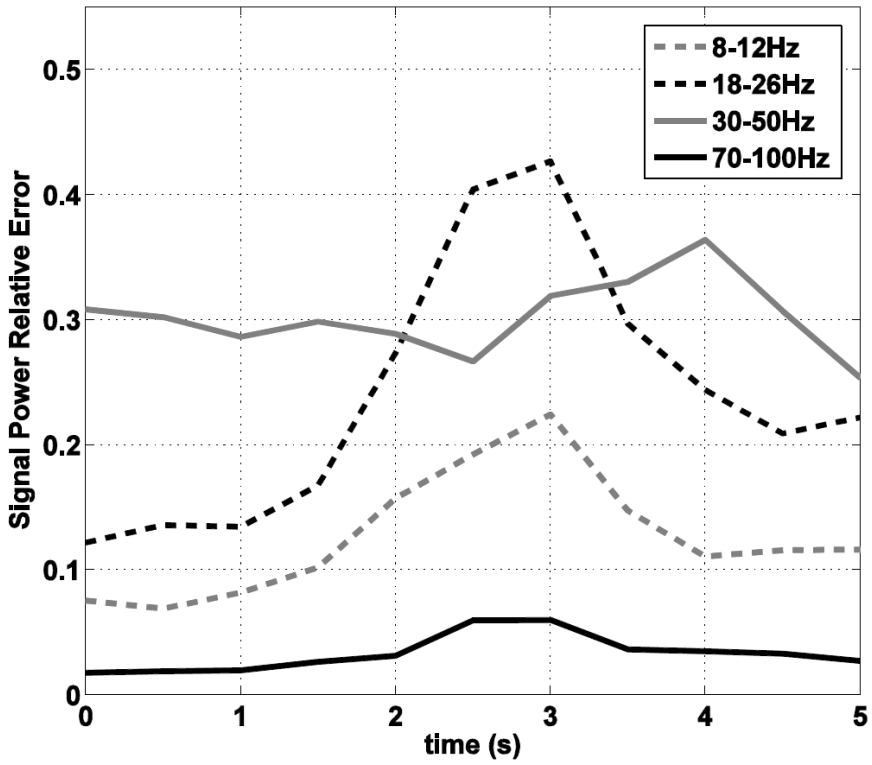


Figure 4.7: Relative error due to a power-of-two implementation of the SPC for an awake subject completing 1D random motion of both hands.

4.5 Experimental Results

The filter and power extractor were synthesized using only standard cells and fabricated in 130nm CMOS as part of the BSN SoC (Figure 4.8).

They operate correctly across the target range of 0.3V-0.7V with a corresponding frequency range of 8 kHz-6 MHz. Figure 4.3 shows the mechanisms for additional power reduction in both blocks. Clock gating channels after the result is ready reduces switching energy by 4X, and the MCU power gates the filter and extractor using PMOS headers when the blocks are unused, reducing leakage by up to 15X. It was shown that the frequency response of the FIR in the EEG energy extraction bands shows that coefficient quantization had little effect on cutoff steepness (Figure 4.5). Figure 4.6 also shows the accuracy benefits of going from the 15 to 30 tap modes in the α, β, low-γ, and γ, δ, and θ frequency bands. A measured

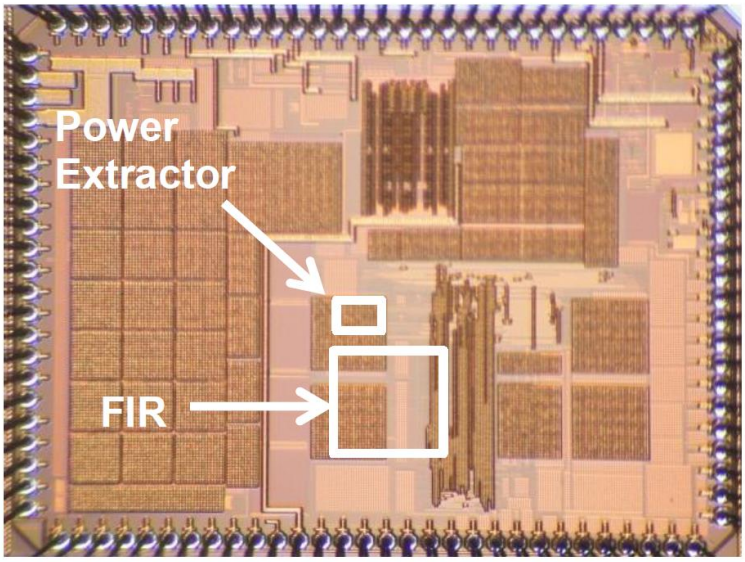


Figure 4.8: Body sensor node chip micrograph with power extractor and FIR.

energy-delay plot is shown in Figure 4.9 with a minimum energy-delay product occurring at 350mV and 29kHz for the one channel, 30-tap, and 128 length SPC window case.

Energy with respect to the supply voltage and delay with respect to the supply voltage are shown in Figure 4.10.

Recent subthreshold FIR filters have included fewer than 15 taps, consumed more power, more area, and have a much larger FOM without having the flexibility demonstrated in this design as seen in Table 4.2[2]. To date, our filter has the smallest FOM compared to the state of the art. The design implemented in [43] implements an analog multi-channel extractor with 4th order bandpass filters followed by an integrator for an area 12x larger and a power consumption of 24x larger than our design.

To measure the signal power within different frequency bands, filtered EEG data was input to the SPC to determine power per band. Compared to a direct implementation of the power extraction equation in 4.1, the power-of-two format SPC has <7% error in the average case for random input values with a mean of 128 such as the data provided by the

[2]FIR Figure of Merit: power(nW)/frequency(MHz)/# of taps/input bit length/coefficient bit length.

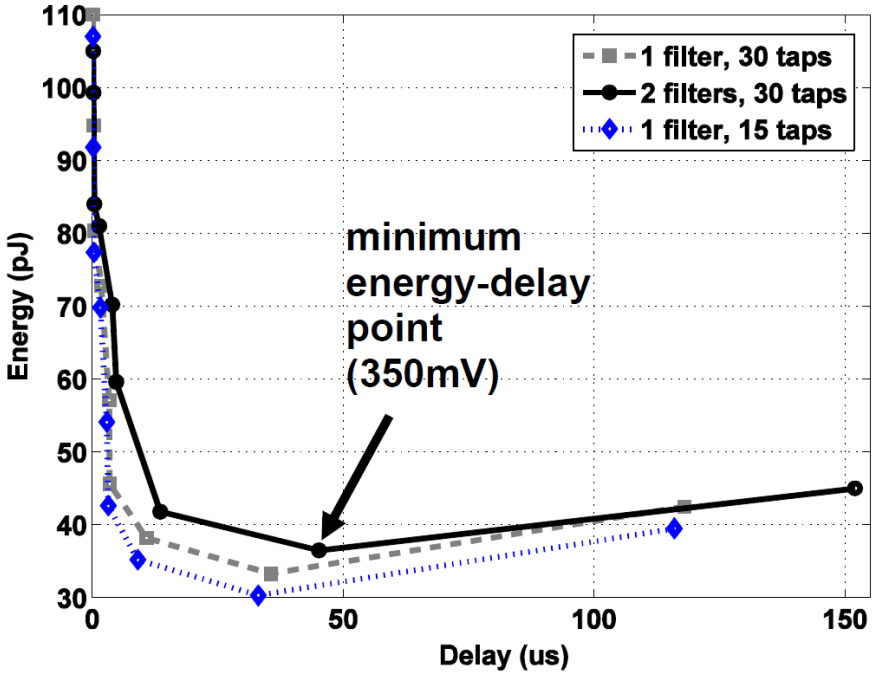


Figure 4.9: Measured energy-delay curves for 1 active channel with 30 taps, 2 active filters using 30 taps, and 1 active channel using 15 taps including the SPC with a window size of 128.

8-bit on-chip ADC. The total area of the signal power circuit was 180x180μm and always consumed <10% of the energy of the total power extractor including the FIR. This FIR filter and SPC blocks help the BSN SoC maintain its robust, ULP energy harvesting operation.

4.5.1 Conclusions

A programmable, subthreshold signal band power extractor was designed for use in a low-throughput and ULP BSN SoC. The filter and full extractor circuit are the lowest power designs compared to the state-of-the-art. Although presented here for a single biomedical application, the filter is flexible for use in general purpose DSP applications. By serializing the traditional FIR architecture, the number of adders and multipliers required for the design was reduced, thereby reducing active and leakage energy for the overall design. Individual channel clock gating was used to reduce leakage for on-chip modes using fewer than the four

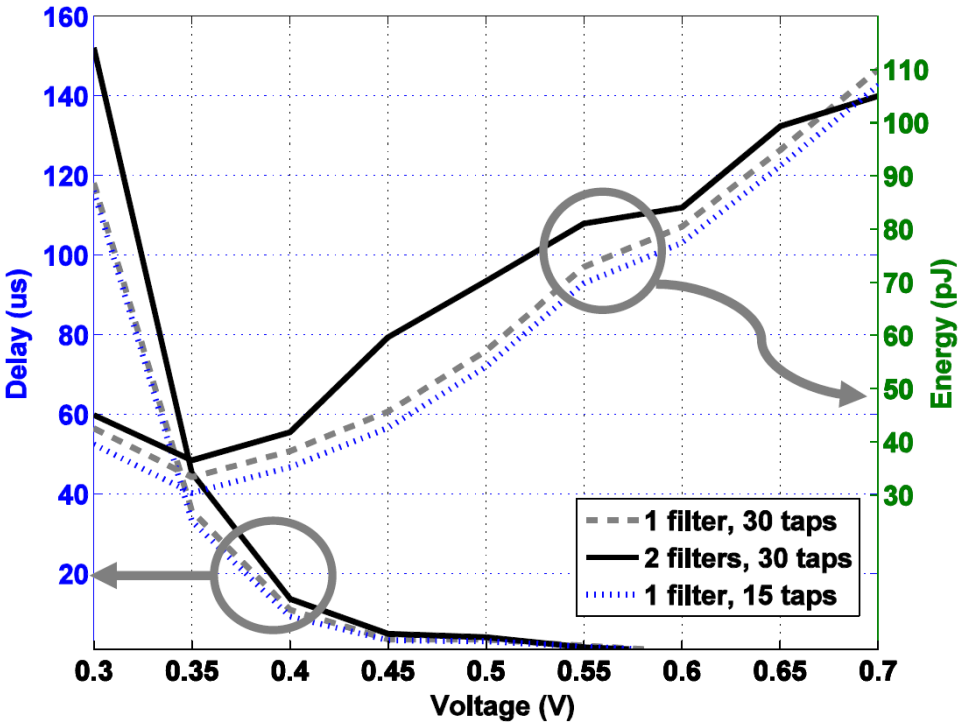


Figure 4.10: Relationships between the supply voltage and the energy and delay of the power extractor.

maximum electrodes. Similarly, the SPC successfully uses power-of-two format to reduce computational complexity, area, and power for little loss in data fidelity. As the extractor was synthesized using only standard cells, the design is highly portable for new technologies.

This design is an example of both direct and indirect DSP power reductions. By modifying the SPC architecture to be serialized to reduce area and leakage and use power-of-two arithmetic, the system power was directly lowered. By replacing an analog implementation in [43] that was orders of magnitude larger in area and power with a simpler, more energy-efficient circuit, system power was indirectly impacted. Both techniques were used to dramatically lower the system power of the BSN SoC presented in the previous chapter.

	This Work	[46]	[47]	[43]
Type	30-tap, 8-bit	8-tap, 8-bit	14-tap, 8-bit	4^{th}-order analog
Channels	4	1	1	4
Programmable	✓	✗	✗	✓
Technology	130 nm	130 nm	130 nm	130 nm
Supply Voltage	350 mV	200mV	270 mV	1.2 V
Frequency	29 kHz	12 kHz	20 MHz	20 kHz
Energy/Tap	1.10 pJ	1.19 pJ	1.11 pJ	(total) 39 pJ
Power	32 nW	114 nW	310 μW	780 nW
FOM	0.57	18.55	17.37	N/A
Area/Channel	0.058 mm^2	1.54 mm^2	0.38 mm^2	0.7 mm^2

Table 4.2: FIR comparison table

Chapter 5

A Reduced-Memory FIR Filter Using Approximate Coefficients

This chapter presents a ULP FIR filter using a method that approximates filter coefficients on-chip without reliance on dedicated memory such as SRAM. In an SoC context, this method allows for full power gating of the coefficient unit without coefficient state-loss, and runtime modifications of filtering specifications, such as the filter order, N, and cutoff frequency, f_c. Using trigonometric approximation methods for the sinc and resource sharing of computational units, a single coefficient is generated in five clock cycles. The approximation unit is compared against standard-cell-based memories, such as register and latch files, for energy and area, and the design is synthesized in 130 nm CMOS consuming 6.9 nW at 300 mV and 6.5 kHz.

5.1 Introduction

Applications with low-speed requirements, such as environmental and physiological monitoring, enable system clock frequencies to be scaled into the 10-100kHz range allowing for subthreshold operation in the digital circuitry. Since the sampling rates of these acquired signals, such as ECGs, typically have sampling rates in the hundreds of Hz, it leaves hundreds to thousands

of clock cycles between samples for processing between receiving the next sample of data. In the subthreshold region, where leakage energy begins to dominate, serializing logic at the cost of more clock cycles has little impact on energy consumption [52] so that the timing slack between receiving a new sample can be utilized for more serialized processing and reduce the area footprint of signal processing blocks.

Flexible and programmable biomedical SoCs often require large sets of filtering coefficients during deployment for a variety of scenarios (e.g. removing 60Hz power line noise or band selection). This can lead to large amounts of coefficient storage, and for batteryless systems such as in [10], an intermittent supply can lead to complete coefficient state loss and the need for reprogramming the on-chip storage. Generally, an FIR filters coefficients are stored locally in a register file when there are few or in an on-chip SRAM or register file that must also operate in subthreshold. The SoCs data memory is rarely power-gated as it also stores chip data that may be unrelated to the filtering process. This leads to excess leakage in memory dedicated to stored coefficients when the filter is not being used. For ULP SoCs, on-chip SRAMs can consume 60% of the total digital power, and if not power gated can dominate the digital power budget [53]. They are often the primary barrier to low-voltage operation (Figure 5.1) that is being exacerbated with technology scaling and pose the first failure point in low-voltage designs [52].

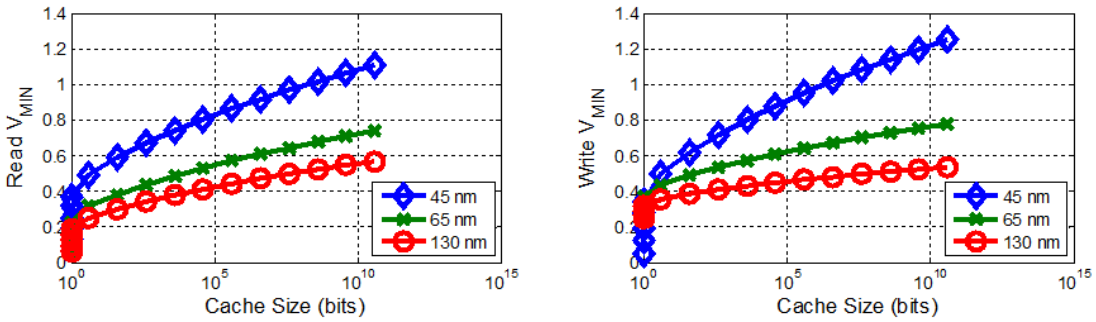


Figure 5.1: Read (left) and write (left) static V_{MIN} versus cache size across technology nodes [54]

Additionally, in the deep subthreshold region, decreased I_{on}/I_{off} ratios reduce the reliability of SRAM. Standard-cell-based coefficient memories that are local to the accelerator to hold data and coefficients are an alternative, but this puts an upper bound on the order of the filter and becomes energy and area inefficient for sizes >1 kb [55]. By generating the coefficients in real-time using digital, synthesized logic, design robustness is improved and the minimum supply voltage and leakage is decreased without the need for SRAM-based storage.

5.2 Methods of Approximation

Assuming a symmetric FIR filter, the filter can be implemented using a folded delay line shown in 5.1.

$$y(n) = \sum_{k=0}^{N/2} h(k)[x(n-k) + x(k)] \tag{5.1}$$

Here, x is the sampled and delayed input data, h is the array of coefficients, M is the filter order, and y is the filtered output. This form reduces the number of multiplications by a factor of two relative to the classical form. FIR filtering coefficients are sampled points along the sinc function, $sin(x)/x$, where windowing functions are applied to smooth the effects of truncating this infinite function. Although there are other methods of creating filter coefficients such as frequency sampling and least-squared, many of these are computationally complex requiring FFTs or optimization methods.

To generate the sinc, the sine function must be approximated in a way that isnt too complex to implement in hardware, but is also accurate enough. Four methods were evaluated and the results are shown in 5.2. Before the sine and cosine can be computed, the input values must first be normalized to the the range $[0, \pi]$.

A common approximation used for trigonometric functions is the Taylor series seen in Equation 5.2 for sine and 5.3 for cosine.

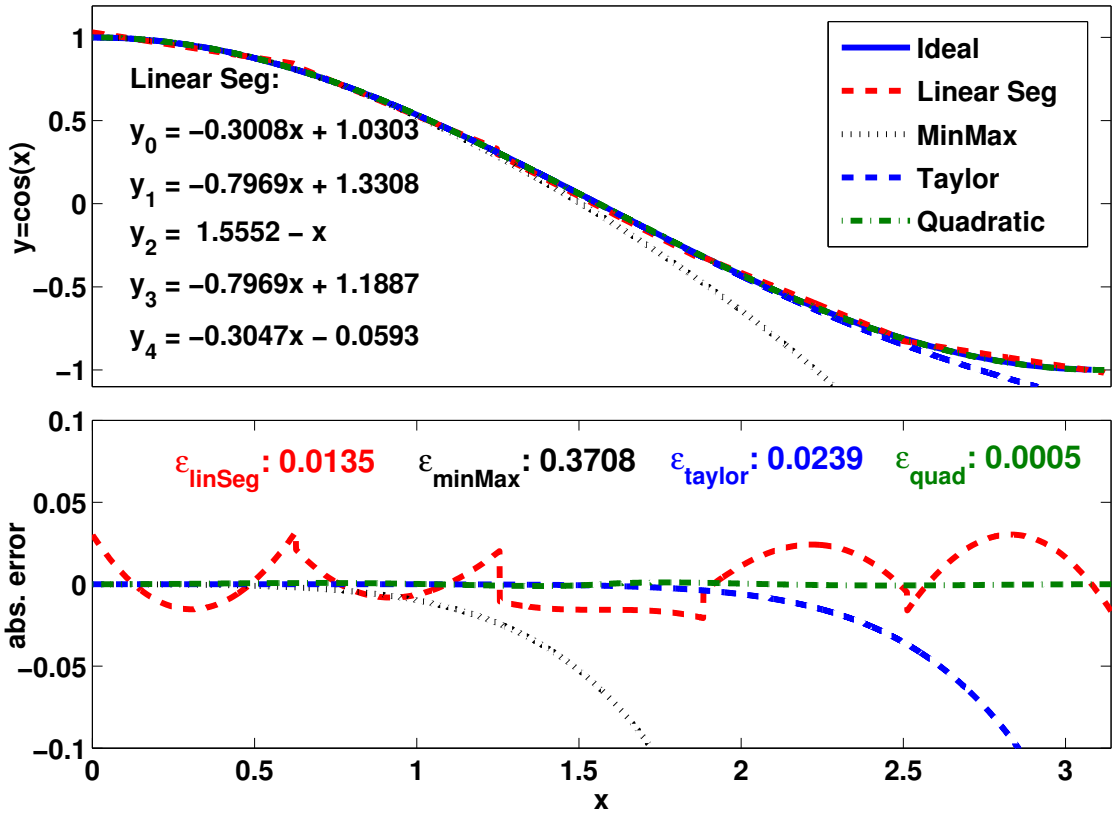


Figure 5.2: Approximations (top) and absolute error (bottom) for cosine $[0, \pi]$. Average error, ε, is shown for each method.

$$\sin(\theta) \approx \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!} x^{2n+1} = \theta - \frac{\theta^3}{3!} + \frac{\theta^5}{5!} - \frac{\theta^7}{7!} + \ldots \tag{5.2}$$

$$\cos(\theta) \approx \sum_{n=0}^{\infty} \frac{(-1)^n}{2n!} x^{2n} = 1 - \frac{\theta^2}{2!} + \frac{\theta^4}{4!} - \frac{\theta^6}{6!} + \ldots \tag{5.3}$$

This series is expensive to compute for accurate results (i.e. more terms) and the resulting error is only acceptably low for small values of the input argument [56]. A 7th order approximation was used for comparison here, but it required seven multiplications, three divisions, and three additions to compute the result. The Taylor expansion has a poor maximal error, and it's desirable to find a way to take some of this error and spread it out across the entire range. Chebychev showed that every approximation has a unique

polynomial that has an equal amount of error across the range where the maximal error has been minimized and its called the minimax polynomial [57]. The derivation of the sine and cosine functions is outside of the scope of this work, but is shown in Equations 5.4 and 5.5.

$$\sin(\theta) \approx \theta - 0.16666x^3 + 0.00833216x^5 - 0.00019515x^7 \tag{5.4}$$

$$\cos(\theta) \approx \sin(x + \pi/2) \tag{5.5}$$

This method is often preferred over the Taylor polynomial method due to its distributed error over the range, which also decreases with additional terms, but suffers from the same computational complexity issue as Taylors approximation [56]. The third method, a quadratic approximation, uses a parabola to approximate the sine/cosine function across the [0, π] range (Figure 5.3), but requires three total multiplications per approximation.

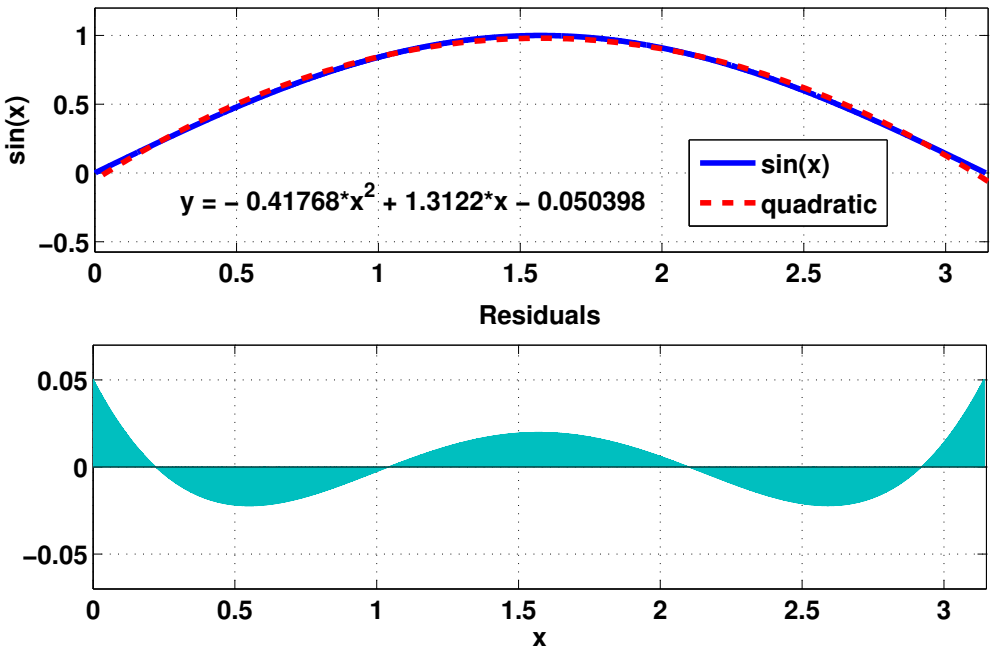


Figure 5.3: Coefficients for quadratic approximation derived from MATLAB best-fit

The final method uses linear segments to approximate the cosine and thus requires only one multiplication per computation. An unconstrained nonlinear optimization method was

used to choose slope and intercept values that reduced the residual sum of squares of the fitted lines, and the segments are shown in 5.2. Five linear segments were used to achieve comparable error with the other, higher order methods, while requiring fewer hardware resources, and this was used for the final design.

One common shift and add based method of computing elementary functions that was not evaluated was the CORDIC algorithm. CORDIC belongs to a class of digit-by-digit algorithms with linear convergence and sequential behavior. This means for an n-bit precision result, n iterations are needed, with the constraint that the $(k+1)^{th}$ iteration may begin only after the k^{th} has been completed [58]. Due to the excessive flexibility of the CORDIC algorithm and the large area due to the use of multiple barrel shifters and adders to achieve a result in few clock cycles, this method was not considered in the comparison.

5.3 Coefficient Generation

Using a low-pass filter as an example, the methodology for coefficient generation is described here and shown in Figure 5.4. First, the filter specification is defined by providing the desired cutoff frequency, f_c, and the filter order. From this, an ideal impulse response function (sinc) can be generated shown in equation 5.6.

$$h(i) = \frac{\sin(2\pi f_c(i - M/2))}{i - M/2} \tag{5.6}$$

This impulse response is delayed to keep the system causal, and the sinc function is multiplied by a specified window function and sampled to obtain coefficients. Many of the most common windows such as Hanning, Hamming, or Blackman contain cosine functions that require an additional trigonometric computation shown in Table 5.1.

For this purpose, the approximation unit is dual-purposed for both sine and cosine approximations, using a shift of $\pi/2$ shown in equation 5.7.

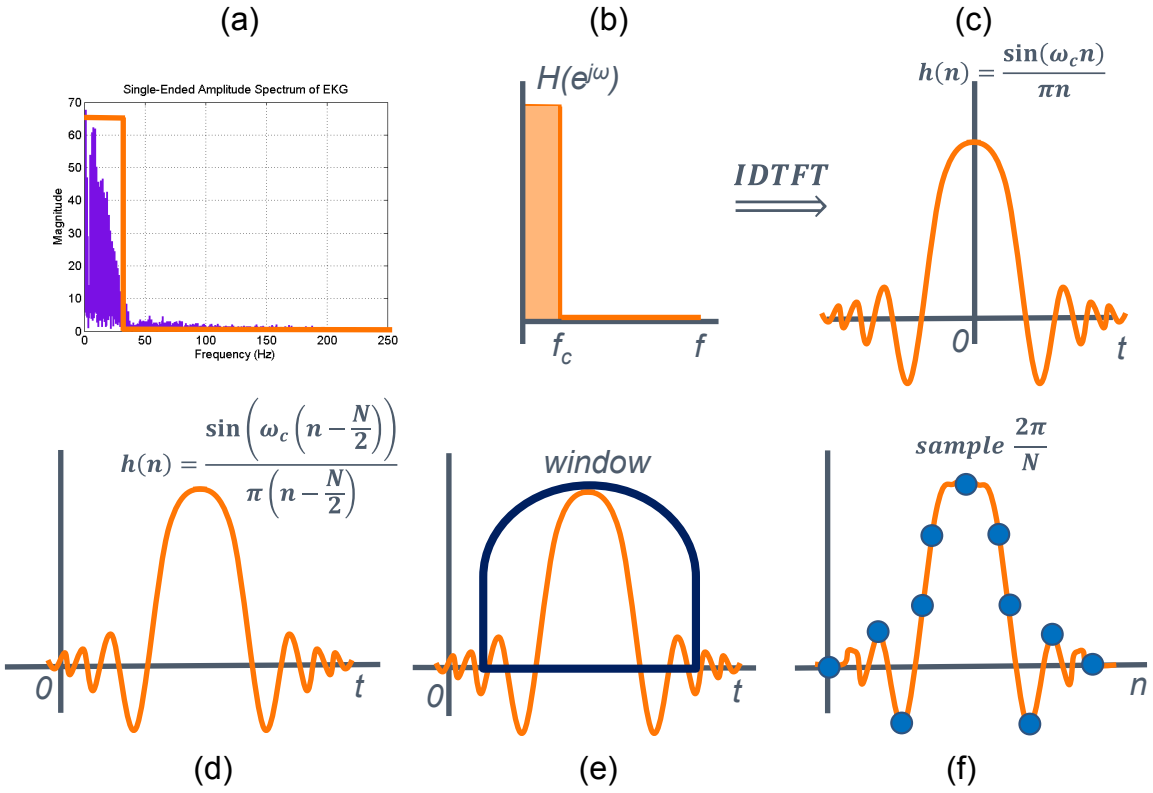


Figure 5.4: Generating FIR filtering coefficients from a specification, f_c. (a)(b) Define a cutoff frequency in the frequency domain (c) convert back to the time domain using an IDTFT (d) shift sinc function to make causal (e) window to reduce effects of truncation (f) sample to generate coefficients

$$w(i) = 0.5 - 0.5\cos(2\pi i/M) \tag{5.7}$$

The approximated frequency response functions were compared with the ideal response in 5.5 for both rectangular ($w(i) = 1, \forall i$) and Hanning windows. Although the passband regions appear unaffected by the approximation, the stopband attenuation is slightly degraded for the approximated cases.

Since this algorithm can be mapped to hardware in a variety of ways, the specific, serialized method is shown in 5.6. The arithmetic resources were constrained to a single multiplier, divider, and approximation unit. Due to the resource-shared hardware, state order had to be carefully chosen to prevent combinational loops.

Window Type	Functional Description
Rectangular	$w[n] = 1$
Bartlett	$w[n] = 1 - \frac{\lvert n \rvert}{N+1}$
Hamming	$w[n] = 0.54 + 0.46\cos(\frac{2\pi n}{2N+1})$
Hanning	$w[n] = 0.5 - 0.5\cos(\frac{2\pi n}{N})$
Blackman	$w[n] = 0.43 - 0.5\cos(\frac{2\pi n}{N-1}) + 0.07\cos(\frac{4\pi n}{N-1})$

Table 5.1: Common filtering windows

5.4 Measurement Results

To determine the suitability of this method in a real system, an approximation and multiply-accumulate (MAC) unit were taped out in a 130 nm commercial process with chip micrograph and logical area distribution shown in 5.7.

Figure 5.8 shows the measured energy results of the approximation unit against the simulated results of 16-bit register/latch files of different depths/rows, corresponding to number of taps.

From this, the point at which this method becomes more advantageous than standard-cell-based coefficient storage is seen with respect to supply voltage. Since the area of the approximation unit remains constant for any number of tap values, it becomes more area efficient than a register file at 30 rows/taps and a latch file at 55 rows/taps. The design was synthesized in 130 nm CMOS and was tested to operate down to 300 mV where it consumed an average energy of 1 pJ/state.

5.5 Conclusions

This work presents a methodology for generating coefficients on-chip in lieu of storing them in high-leakage SRAMs operating below the threshold voltage. This design can enable reliable, ULP operation for applications requiring large banks of coefficient data and diverse

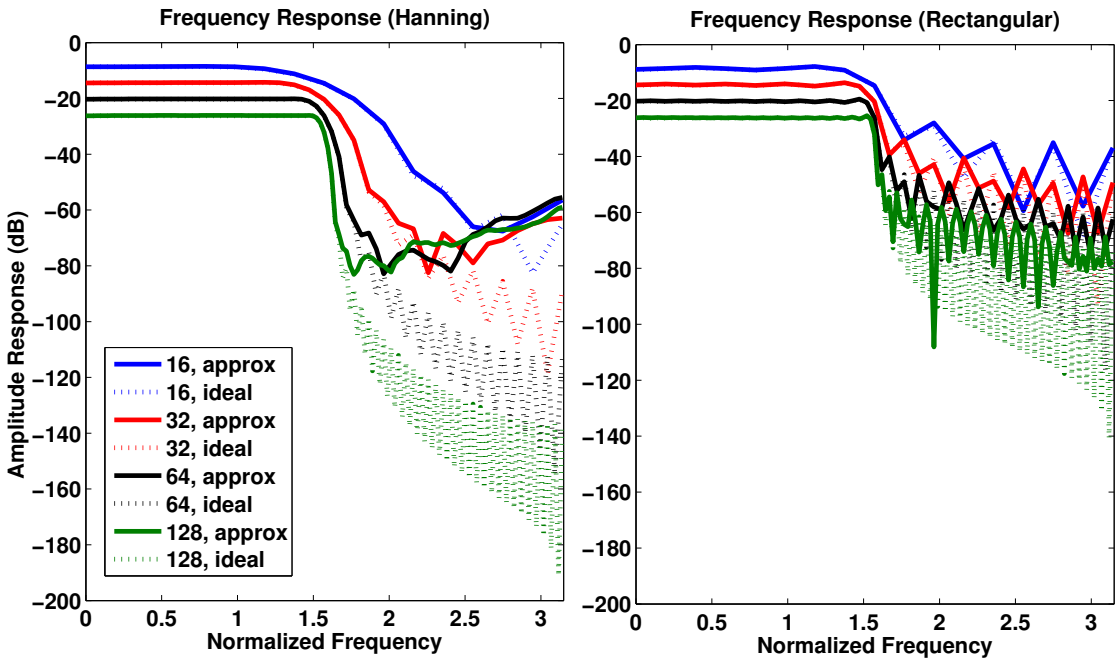


Figure 5.5: Ideal and approximated frequency response functions for $f_c = 0.5\pi$ using a rectangular and Hanning window.

filtering operations. By evaluating a variety of trigonometric approximation methods for the complexity-accuracy tradeoff, a linear segment-based approximation was selected to generate coefficients. Although this coefficient generation adds latency to the filtering operation, there is often available slack for serial processing in low-throughput, subthreshold systems.

This scheme shows an indirect way to improve system power by using novel DSP techniques to keep more of the on-chip memories power gated, thereby reducing leakage. As large memories are typically banked to allow for finer-grained power gating of sections of the memory at a time, this methodology could exploit this to reduce system-level leakage.

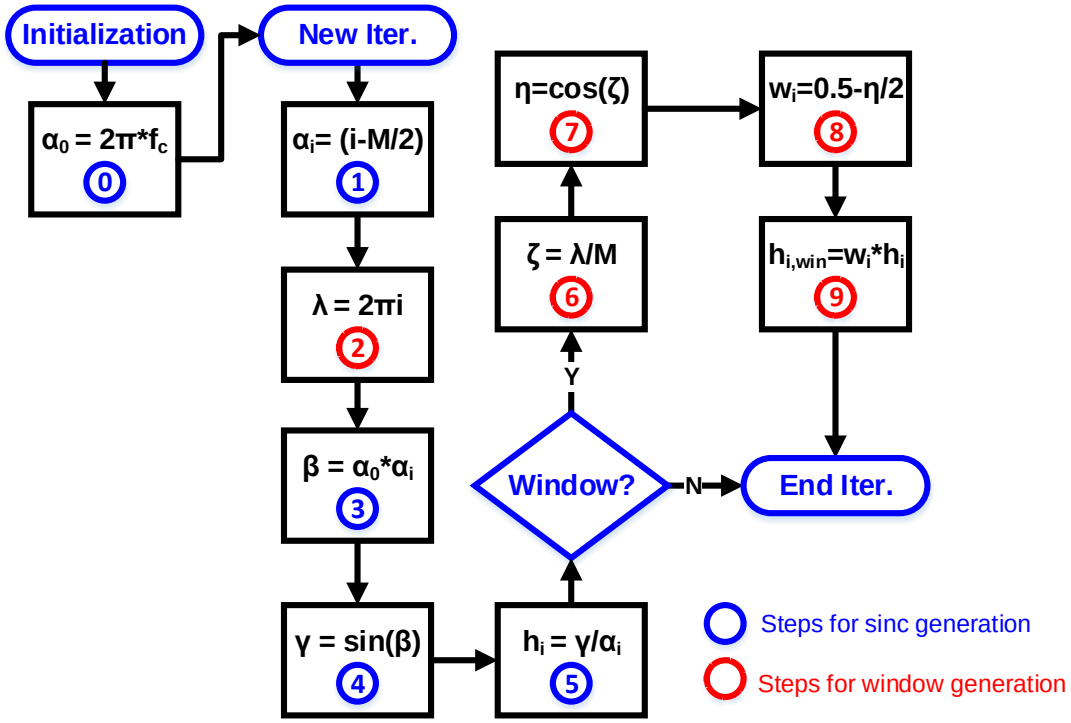


Figure 5.6: A reduced version of the sample figure.

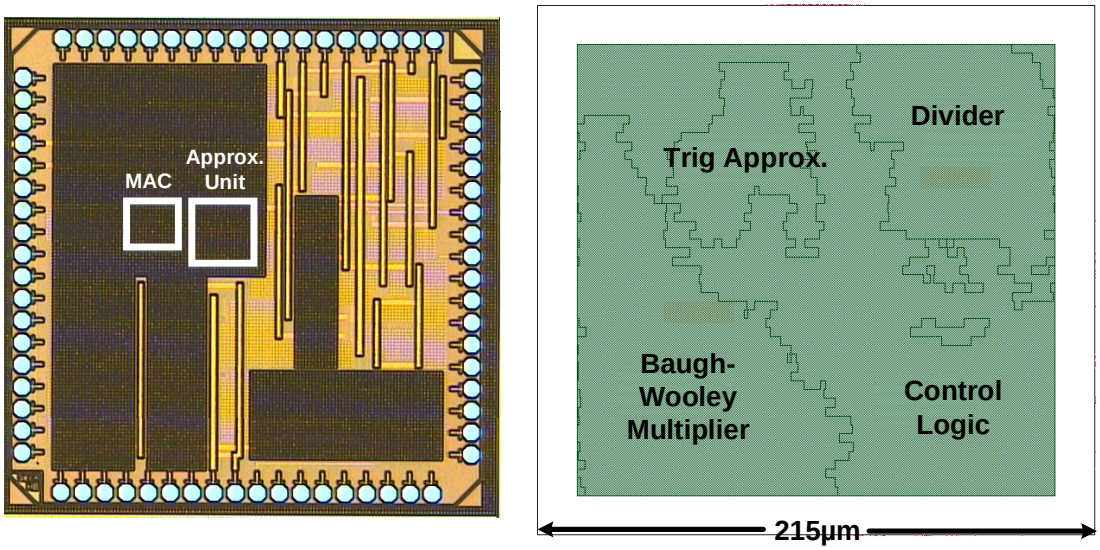


Figure 5.7: Chip micrograph (left) and approximation unit division of area (right).

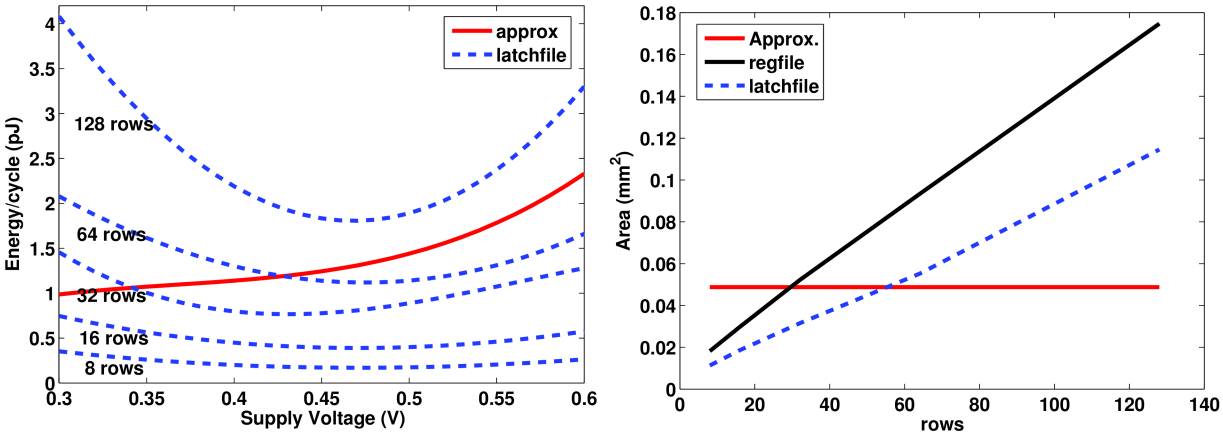


Figure 5.8: Energy and area comparison of the approximation unit with 16-bit register and latch files of various depths.

Chapter 6

Arithmetic Approximation Methods for a Hardware VAD Unit

This chapter presents low power design techniques for a voice activity detection (VAD) pipeline focusing on an overview of methods for combinational, base-two logarithmic approximation using Mitchells algorithm, piecewise-linear/quadratic error compensation schemes, and direct approximation. For the logarithm, optimization methods are used for computing linear segments for each compensation scheme including Hamming weight minimization and pattern recognition of segment slopes for multiplier-less error compensation. A novel, near-zero-average error quadratic compensation scheme is also presented. A test chip was fabricated in a commercial 130nm technology including fifteen base-two logarithm approximations and each was evaluated for its standalone accuracy, measured energy, delay, and area in order to determine the best classes of approximation for low-energy and high-accuracy operation.

A system-level evaluation of the DCT and logarithm blocks is performed using a software model of a keyword detection speech pipeline to predict the impact of arithmetic inaccuracies on the detection accuracy of keywords across various noise levels. Recommendations for relaxing accuracy constraints are provided for arithmetic modules where block-level accuracy has little impact on system-level detection rates. This allows for more imprecise designs that

will reduce the overall VAD power while maintaining the keyword detection accuracy. As this work focuses on the design of a ULP logarithm, the standalone power improvements of this block will also be discussed.

6.1 Introduction

Voice-based control is emerging as a promising interface for mobile devices including applications such as voice command and speech recognition. Voice interfaces are often active and continuous listening is a high power operation, thus there is demand to minimize power consumption of all blocks involved in passive monitoring of keywords. Most importantly, using computational approximations to reduce complexity can often save power without compromising the overall speech detection accuracy. Speech processing applications typically use Mel-Frequency Cepstral Coefficients (MFCCs) to evaluate the content of audio data. MFCCs are frequently used in speaker identification, which is beneficial to differentiate the voice signals of a legitimate user to prevent frequent wakeups in the example of a battery-constrained mobile device. The Mel scale is commonly used due to its ability to approximate the human auditory system, and they have proved to be more compact and efficient for representing important speech characteristics. To extract features such as keywords or emotion, the MFCC parameters must be evaluated to determine various characteristics of utterances. Some speech processing schemes compress speech data and extract key distinguishing features of the signal content that can be used for multiple types of feature extraction. MFCCs are an example of this, and are a parametric representation of acoustic signal data that can be used to detect the presence of voice or further processed to do keyword or emotion detection.

The datapath to compute MFCC coefficients is shown in Figure 6.1 and typically consists of the following stages: sampling/windowing, power spectral density extraction using an FFT, triangular filtering for converting to the Mel Scale, data scaling using the logarithm, and a DCT for data decorrelation [59].

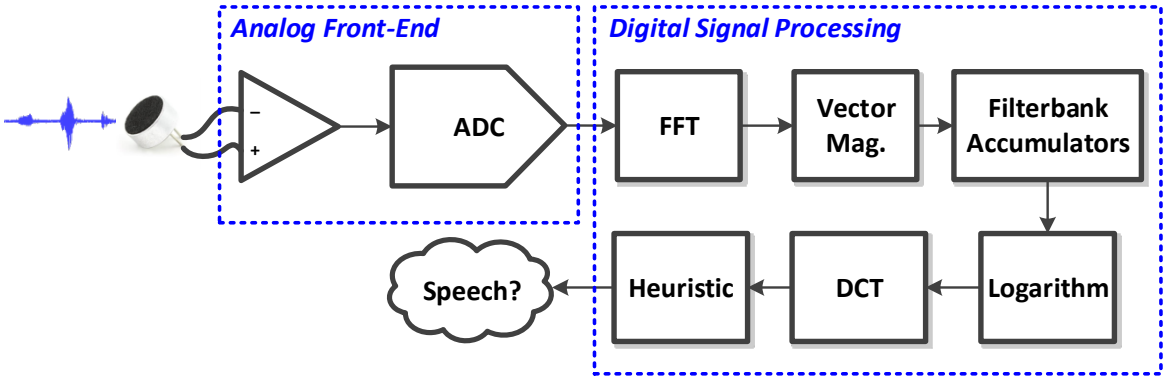


Figure 6.1: Typical voice activity detection pipeline used for computing MFCCs

6.1.1 Sampling and Framing

Before the features can be extracted from the analog speech signal, they must be amplified and digitized in the AFE. Digitizing the signal requires an ADC that samples the analog signal at a rate that captures the full frequency range of the data. Many systems use 16 kHz sampling with frame lengths in the 10s of ms, and these are design knobs that can be used to reduce power.

6.1.2 FFT

Determining the signal power of a speech segment is needed for speech applications that rely on the ratio between the signal and noise and MFCCs that operate on the signal spectrum. An FFT is commonly used for this purpose, but it tends to dominate the energy in a typical VAD scheme consuming up to 50% of the power [59]. Looking at techniques to reduce the power of this commonly used block should greatly reduce the overall system power. The speech applications that require the FFT for frequency domain analysis have a wide range of sampling rates from <8kHz (speech detection, gender identification)-16kHz (keyword detection). A 64-point hardware FFT takes hundreds of clock cycles to compute the result with one available butterfly unit, such that large buffers are needed to store incoming samples for high sampling rate applications. Increasing the radix of the FFT increases the throughput,

but doubles the amount of processing resources required per clock cycle. An all radix-2 FFT can be used for lower throughput applications to reduce energy per operation, while the radix-4 butterfly can be used in some FFT stages for a performance improvement. This introduced flexibility will increase the amount of control logic for addressing and butterfly input data selection, but this will provide finer-grained control of the FFT power for varying detection modes. This throughput-power tradeoff motivates the need for a flexible FFT module with multiple power modes dependent on the current feature being extracted.

6.1.3 Filterbank Accumulators

The filter banks perform a mapping between the linear frequency scale and the Mel scale. These filters are defined by their center frequencies, slope values, and the number of filters. Triangular-shaped filters are typically used, but other shapes have been investigated for improved noise robustness and lower power. Using rectangular-shaped filters with unity height only requires summation of the input values versus the multiply-accumulate operations needed for the triangular filtering. The rectangular method is known to reduce the robustness, but also greatly reduce the power. Hamming-weight filter shapes have also been used to improve the robustness to noise [60].

6.1.4 Logarithm

The logarithm operation is used to generate the cepstrum of the data. In order to evaluate low-power base-two logarithm implementations, we need to evaluate the power-accuracy tradeoff of each method, develop software models for each, test the implementations in a speech pipeline model, and then synthesize the designs into hardware to determine complexity (e.g. power, performance). Logarithms are also used in scientific computing, computer graphics, and artificial neural network applications [59, 61, 14]. Logarithms can be used to simplify multiply/divide or power/root operations to addition/subtraction or multiplication/division in the log domain. If the logarithm is an intermediate step in computation, such as in the case

of data-range compression, high accuracy is often less of an issue than in applications where the logarithms result is the end goal. For applications that tolerate error at the arithmetic block level without heavily impacting system-level results, lower accuracy and lower energy implementations are preferable.

For low complexity designs, three categories of methods are typically discussed: (1) lookup table (LUT) based with/without interpolation; (2) Mitchells approximation with/without interpolation; and (3) direct approximation using linear segments. Recent implementations have relied on Mitchells algorithm, a method based on linear interpolation between powers of two, to compute the base-two logarithm [62]. Mitchells algorithm allows for an extendable implementation of the logarithm using only a leading-one detector circuit and a barrel shifter. The downside of using this method is its peak error that occurs at every half-interval between powers of two as shown in Figure 6.2.

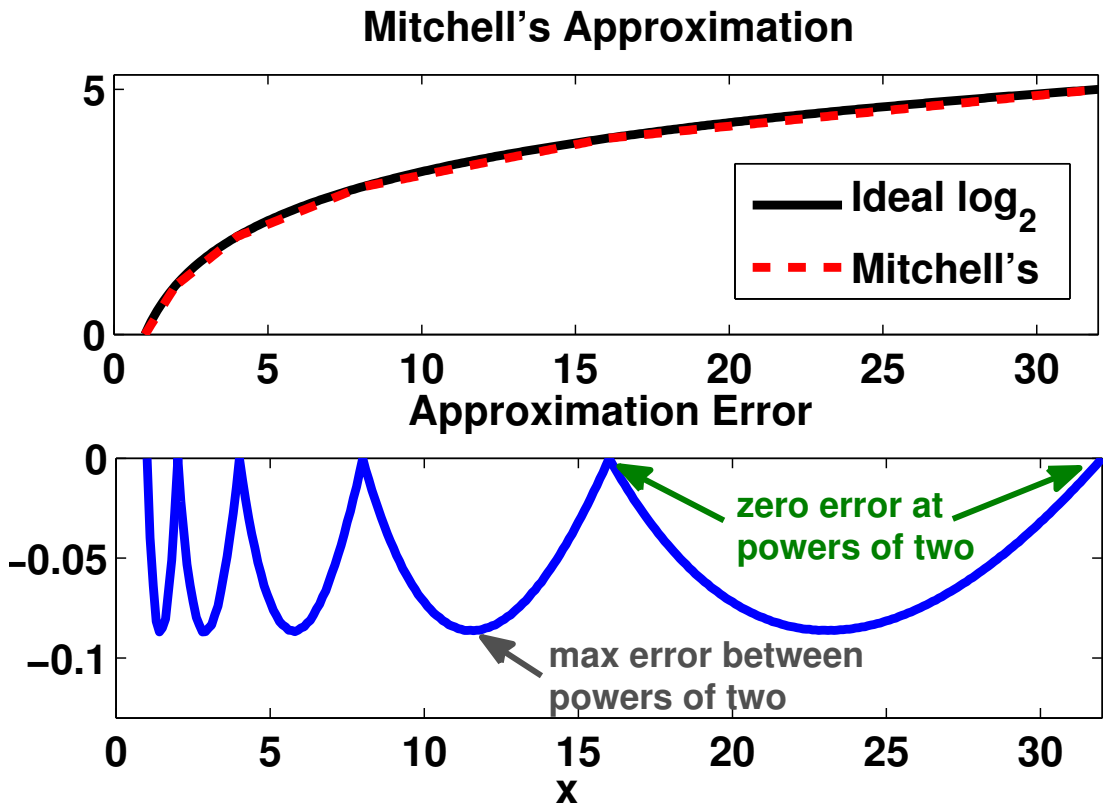


Figure 6.2: Mitchells approximation for the base-two logarithm with corresponding absolute error in the closed range 1-32.

Compensation methods have been developed [61, 14, 63, 64, 65, 66, 67, 68] that model this error to correct the result of Mitchells approximation. In the area of linear compensation methods, few rigorous methodologies are presented for determining slope and intercept values of segments, and it is unclear whether the selected linear segment parameters fully minimize the error. In [64, 65, 66] the most significant bits of the mantissa from Mitchells approximation are used to generate a compensation factor using only addition and simple logic, leading to a multiplier-less method. Other methods such as [67] make use of an LUT in conjunction with interpolators and are seen in applications where performance is the driving metric, but preclude large input ranges as the table will dominate the system power budget.

Although using the approximation with the highest standalone accuracy is often desirable, reduced error implementations tends to be coupled with complex logic and higher power consumption. Certain applications are inherently resistant to inaccuracies in computational sub-blocks, leading to a more relaxed accuracy constraint. For this work, a set of approximation methods with varying accuracy, area, and power were designed using both pre-existing and novel methods. This work aims to evaluate a wide range of base-two logarithmic approximations and demonstrate a speech-based use case where system power can be reduced by evaluating the impact of approximation error at the system level.

6.2 Logarithm Metrics

6.2.1 Accuracy

Approximate computing relies on the ability of a system or application to tolerate some loss of quality or optimality in the computed result [69]. Methods such as dynamic bit-width adaptation can be used to scale the system power in response to available energy or types of computation. This is a powerful and easily available knob for controlling the energy-quality trade-off.

For the accuracy metric, the approximations will be compared to the ideal logarithm of base-two. A goodness of fit metric will be used for describing the accuracy. The sum of squared residuals (SSE) is used for this purpose and is described in equation 6.1. Here, d is the residual value or the difference between the ideal value and the approximated value.

$$norm(d, 2) \equiv \sqrt{\sum_{i=1}^{N} d_i^2} \tag{6.1}$$

The percent error will also be evaluated. For the accuracy at the speech pipeline level, two tests are used. The first test provides the number of correctly identified utterances for different noise levels as a percent. The second test is an edge alignment test to determine how well the approximation works if the utterance doesnt occur in the middle of the frame (i.e. some of it is cut off). The output of the edge-alignment test is a distribution for each sound file. The mean and standard deviation of each was extracted, where having both statistics close to 0 was the ideal.

6.2.2 Complexity

All approximations will be compared based on their circuit complexity. The metrics of focus will be power, area, and delay. Verilog is completed for each approximation and the output of Synopsys DC Compiler is used to estimate these values. Each method was taped out in 130nm CMOS for a full comparison of these key metrics.

6.3 Algorithms for Logarithmic Approximation

6.3.1 Mitchell's Approximation

Mitchell derived an approximation for the base-two logarithm that results in connecting linear segments between power-of-two points on the logarithm curve [62]. For a binary input

value, N, he concluded that the base-two logarithm could be computed using equation 6.2 and the affiliated approximation in equation 6.3.

$$log_2 N = k + log_2(1 + m) \tag{6.2}$$

$$log_2(1 + m) \approx m \tag{6.3}$$

Here, k is the non-negative, integer characteristic given by the location of the leading one in N, and m is the mantissa composed of the bits to the right of the leading one. This implies that the logarithm can be computed using a combination of shifts and adds, making it more amenable to hardware implementation. To compute Mitchells approximation, the position of the leading one in the binary representation of the input value is determined, this position then determines a shift value for a barrel shifter that shifts the bits to the right of the leading one past the decimal point before concatenating k and m to achieve the final approximation.

6.3.2 Error Compensation for Mitchell's Approximation

As demonstrated in Figure 6.2, Mitchells approximation has large error at the midpoint between powers of two and when used alone, provides only 3.53 bits of accuracy. A commonly-used technique to improve this accuracy is modelling the error of Mitchells using a number of linear segments between each power of two. As per the Mitchells error curve shown in Figure 6.2, this error magnitude increases from any power of two to the midpoint between it and the next power of two, at which point it begins decreasing. If compensation schemes are to be used for every inter-power interval, the regions need to be divided to account for at least two separate slopes. The most significant mantissa bit of Mitchells approximation can be used to determine the current segment to reduce the number of values stored in a LUT [64]. For this work, linear segments and their parameters were determined by minimizing

the residual sum of squares (RSS) between the ideal logarithm and the segments using an unconstrained nonlinear optimization method in MATLAB.

Since the shape of Mitchells error is less linear than quadratic, an additional compensation scheme was developed. The best fit quadratic parameters in equation 6.4 were determined for each power of two interval using MATLABs polynomial curve fitting tools. The quadratic scaling factor, j, was recorded for each power of two interval in Table 6.1, and a pattern emerged indicating the value of j is dependent on the smaller power of two endpoint, a. Since Mitchells approximation has zero error at the powers of two, a quadratic function can be derived using the factored form for a quadratic in equation 6.4, where x_1 and x_2 are always powers of two.

$y = a(x - 2^a)(x - 2^b)$				
Range (2^a - 2^b)	a	$b(= a + 1)$	$k(= -2(a + 1))$	j
$2^1 - 2^2$	1	2	-4	$2^k + 2^{k-2} + 2^{k-3}$
$2^2 - 2^3$	2	3	-6	
$2^3 - 2^4$	3	4	-8	
...	...	...	...	

Table 6.1: Derivation of Quadratic Compensation Scheme.

Using Table 6.1, the derived value for j is substituted into the function in equation 6.5. Substituting in the parametric representation $b = a + 1$ and $k = -2b = -2(a + 1)$ yields the final form provided in equation 6.6.

$$y = j(x - x_1)(x - x_2) \tag{6.4}$$

$$y = (2^k + 2^{k-2} + 2^{k-3})(x - 2^a)(x - 2^b) \tag{6.5}$$

$$y_a = 2^{-2a-2}x^2 + 2^{-2a-4}x^2 + 2^{-2a-5}x^2 - 2^{-a}x - 2^{-a-5}x + (2^{-1} + 2^{-3} + 2^{-4}) \tag{6.6}$$

Although this quadratic compensation scheme requires a multiplication for computation of x^2, shifts and adds can be used for the remaining calculation, reducing the hardware complexity compared to using additional multiplications. The quadratic method has a maximum error of 0.008 and a near-zero average error of 0.000013. Figure 6.3 shows the approximation including the quadratic compensation and the resulting absolute error.

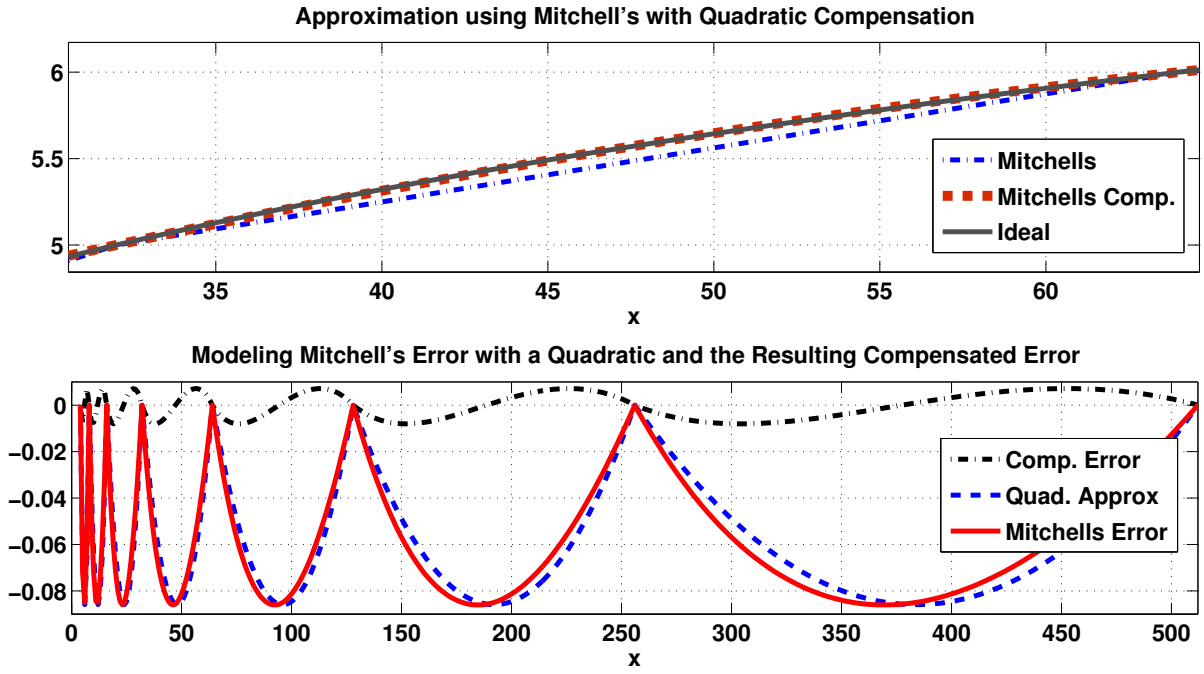


Figure 6.3: Quadratic compensation scheme results showing final approximation and absolute error

6.3.3 Direct Approximation

The logarithm can also be approximated directly using linear segments without the added delay or power of using Mitchells approximation. This requires mapping a set of linear segments to the logarithm function and storing the slope and intercept values shown in Figure 6.4. An RSS-based minimization routine was used to determine the optimal x-axis partition for the linear segments that reduced the percent error for a specified number of segments. The MATLAB routines can be used to approximate any (integer) range of the logarithm for

any given number of segments. For this evaluation, two and four segments were used for each power-of-two region.

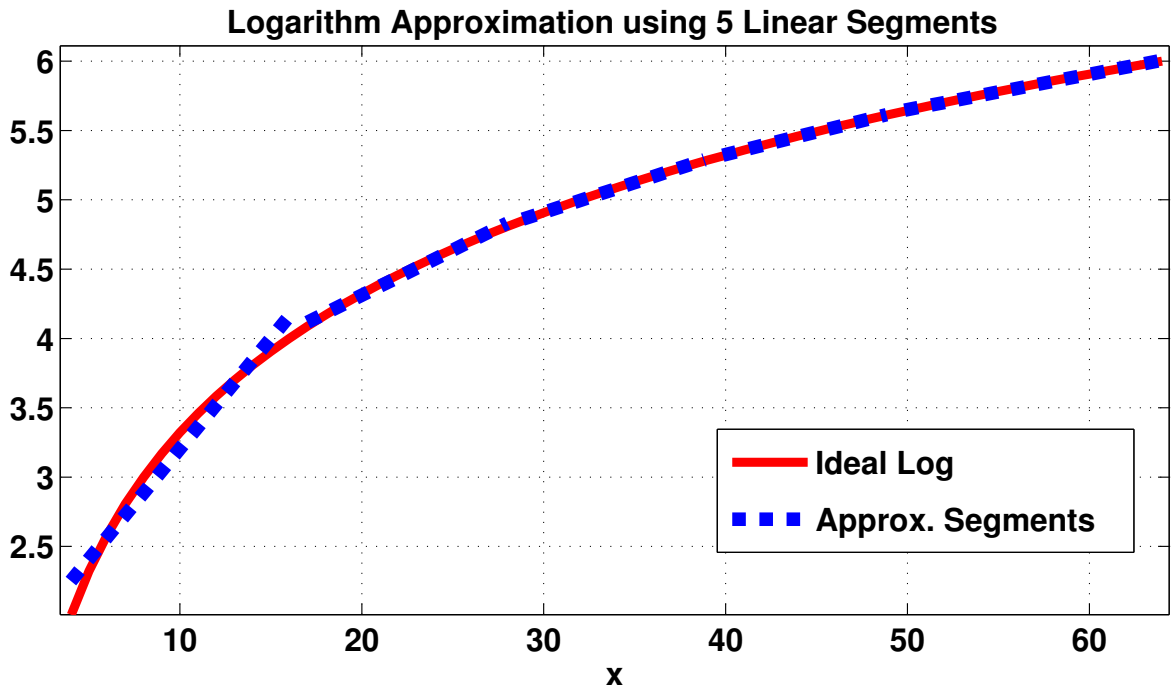


Figure 6.4: Direct approximation of a logarithm from [1, 64] using linear segments.

6.4 Hardware Implementation

Implementing Mitchells algorithm in hardware requires a leading-one detector (LOD) circuit and a barrel shifter. Leading one detection is used in floating point arithmetic, carry-lookahead adder design, and the calculation of the logarithm. LOD circuits are a large factor in the overall delay of the approximation and can be implemented using three primary methods: (1) shift and count routines [63]; (2) ROM-based decoder with a one-hot word; (3) combinational implementations. The shift and count method takes multiple cycles to compute a result, while decoding with a one-hot word is not scalable for large N. For this work, a novel, combinational LOD circuit (Figure 6.5) is developed for all Mitchells-based implementations using 47%

fewer standard cells than common, high-performance state-of-the-art methods [70]. The comparison is shown in Table 6.2.

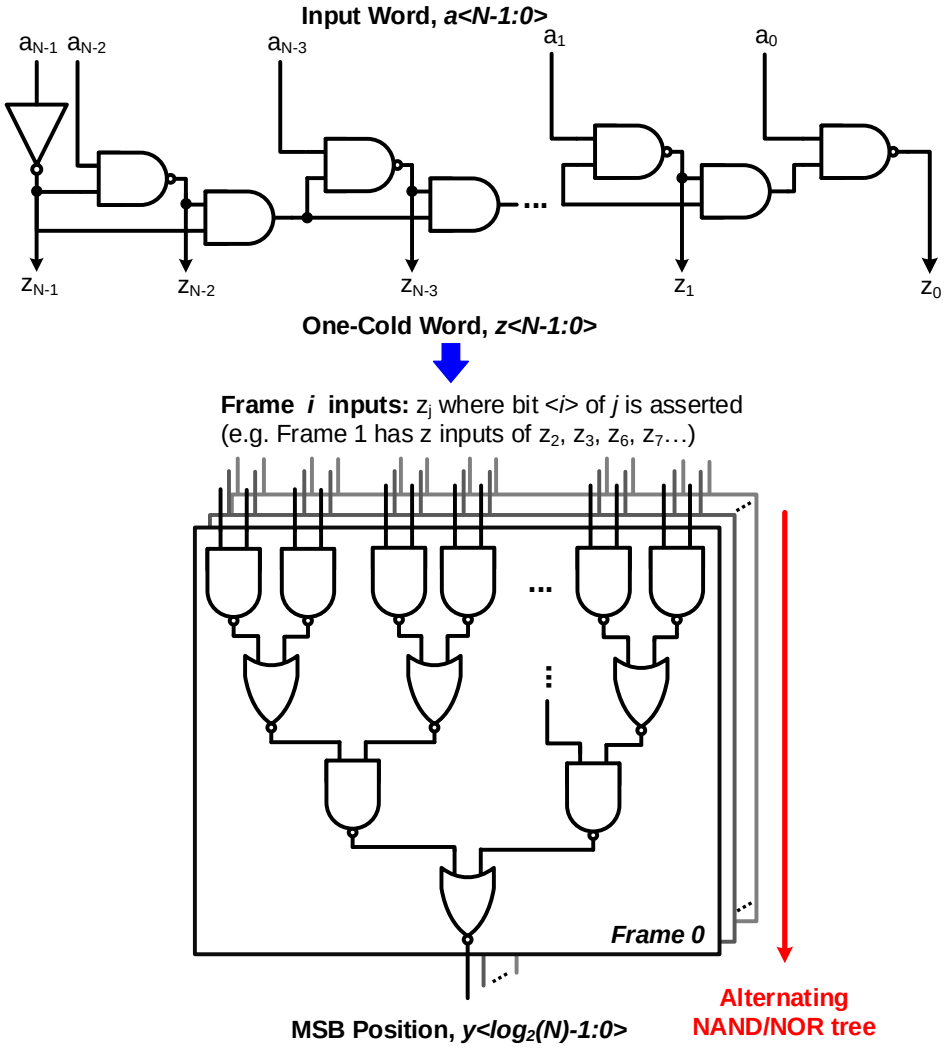


Figure 6.5: Reduced gate count leading-one detector circuit for Mitchells algorithm.

Due to the bit-by-bit serial evaluation of this design, gate delay is a concern for large values of N, so this design targets low power applications where delay is not the constraining metric. To remedy this for faster topologies, combinational implementations that use tree structures to parallelize processing or anticipators/detectors, such as in [71], are typically used.

	Complexity (Synopsys DC)		
	Total Std. Cells	Slack (ns)	Total Cell Area (μm^2)
This Work	42	607	15.19
Abed 2006 [71]	101	542	37.87

Table 6.2: Comparison to state-of-the-art LOD circuit

6.4.1 Hardware Complexity Minimization

For each of the aforementioned approximation methods, hardware complexity can be reduced to improve gate count and power at the expense of the accuracy. The first method investigated in this work aims to trade off the accuracy of the slope values used for Mitchells error compensation with the Hamming weight of the binary representation. The Hamming weight of a number represents the number of 1s in its binary representation and constraining this value can allow for multiplier-less compensation using shifts and adds. Figure 6.6 shows the break-even point for power consumption between using a shift-add based multiplier and a tradition multiplier. This plot helps to determine an upper bound to constrain the Hamming-weight of the coefficients used for approximations.

After minimizing the RSS for linear segment compensation, patterns emerged between the slopes and intercepts for each region of the inter-power of two interval. The slope values can be determined using shifts and adds based on the current power of two, a, thus eliminating the need for a LUT containing slopes for all intervals in the range. The intercept values exhibited similar patterns and remained constant for all positive/negative slope segments. The slope and intercept values used for all multiplier-less implementations are discussed in Appendix B.

6.5 Implementation and Results

Using a combination of methods from the previous sections, fifteen logarithm units were fabricated in a commercial 130 nm technology (Figure 6.7). The mapping between the

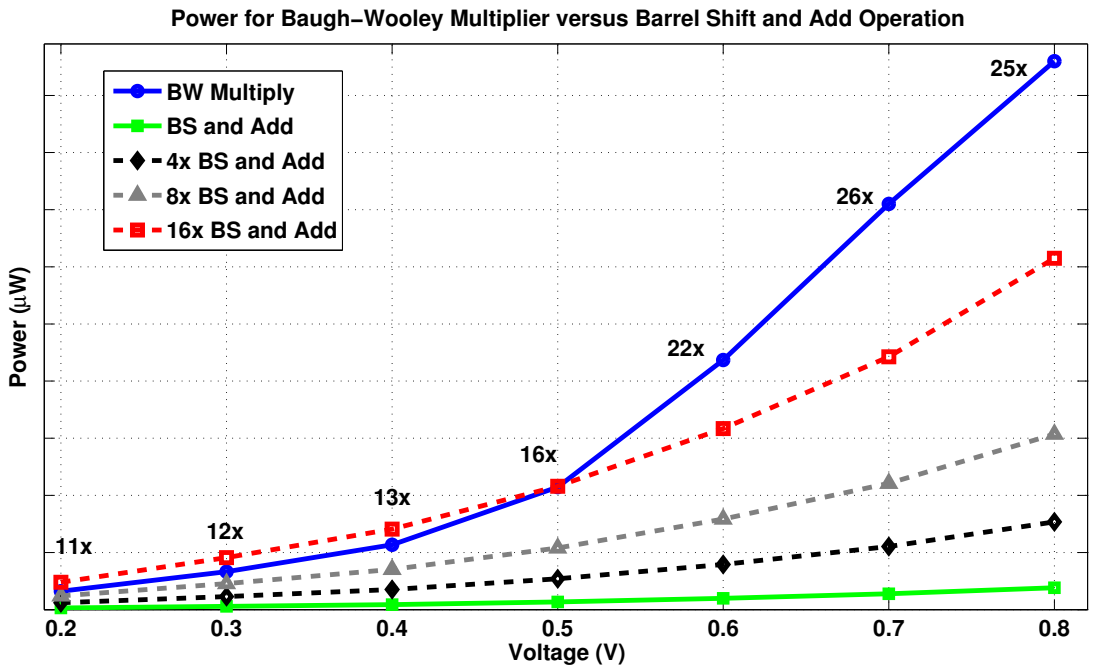


Figure 6.6: Comparison of the power consumption of barrel shift and add-based multipliers versus a standard multiplier across voltages.

micrograph and logarithm units is shown in Table 6.3.

Each approximation block takes 16-bit integer data as input. The key for naming conventions used in this work is: (PWL) piecewise-linear approximation without Mitchells, (M) Mitchells approximation, (MH) minimized Hamming weight, (MULT) multiplier, (SO) optimized x-axis partitioning, and (4S) four segment method (versus two for all others). The full names also use +/- to indicate the addition or subtraction of a feature. The energy and delay of each approximation block was measured across a range of voltages (500 mV - 1.2 V). Figure 6.8 shows the E-D curves of these blocks grouped by their energy for readability.

Figure 6.9 provides the area of each design and Table 6.4 demonstrates the standalone error (evaluated using RSS) for each method.

Based on these results, some approximation methods do not yield significant enough accuracy improvement to warrant their additional circuit complexity. A comparison of all methods is shown in Figure 6.10 to show the accuracy-energy-delay trade-off.

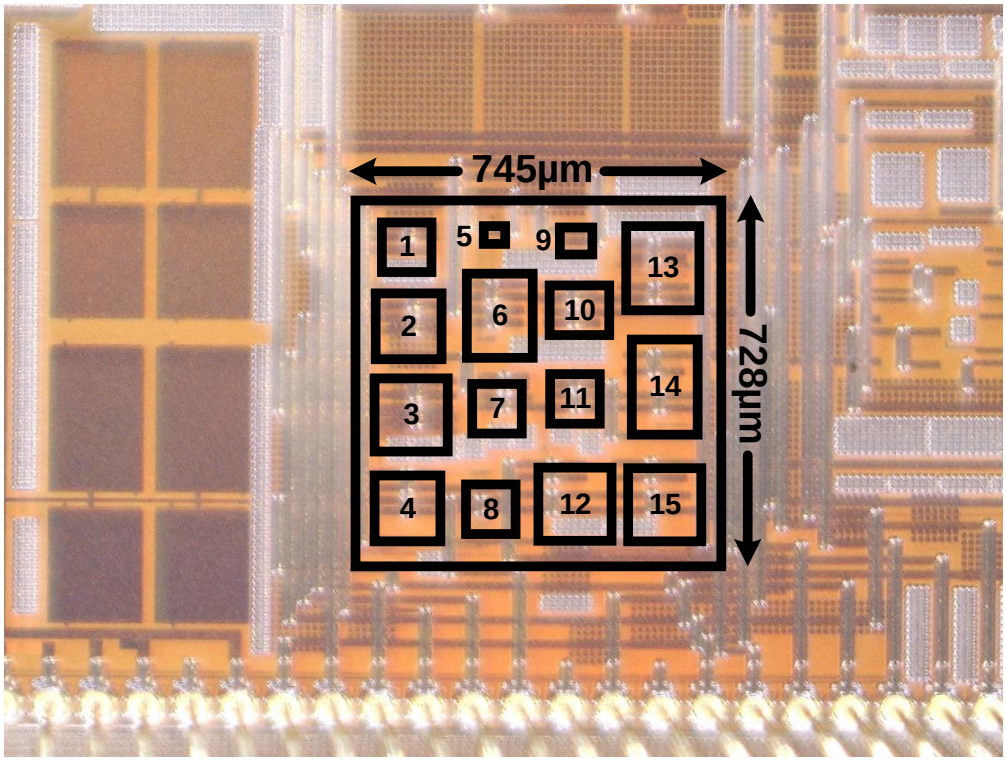


Figure 6.7: Chip micrograph highlighting the logarithmic approximation units.

From this, the best methods that optimize this tradeoff are the direct piecewise-linear approximations, particularly those using pattern-based intercepts and slopes (i.e. -MULT).

6.6 System-Level Case Study

A software model of the voice activity detection pipeline with a keyword detection back-end was used with the TIMIT speech corpus [72] with varying levels of superimposed noise to test the system-level accuracy of all logarithm approximations. The keyword detection accuracy for each approximation is shown in Table 6.4.

The important observation drawn from this system-level evaluation is that the standalone accuracy of the logarithm had little to no impact on keyword detection accuracy. This implies the designer should simply choose the lowest power approximation to implement

1	M+SO+MH-MULT
2	PWL+MH
3	M+4S
4	M+4S-MULT
5	LOD
6	PWL+SO+MH
7	M+MH-MULT
8	PWL+MH-MULT
9	Mitchells
10	M+SO-MULT
11	PWL+SO+MH-MULT
12	M+SO+MH
13	M+SO
14	M+MH
15	M+QUAD

Table 6.3: Mapping between numbered blocks in chip micrograph and logarithm units

the logarithm, leading to lower overall system power and challenging the idea that the most accurate block-level implementation is always the best choice.

6.7 Hardware Reuse for the DCT

Following the logarithm computation in the keyword detection pipeline is the multiplication with a cosine term shown in Equation 6.7.

$$\cos\left(\frac{j(i+0.5)\pi}{N}\right) = \cos\left(\frac{ij\pi}{N}\right)\cos\left(\frac{j\pi}{2N}\right) - \sin\left(\frac{ij\pi}{N}\right)\sin\left(\frac{j\pi}{2N}\right) \tag{6.7}$$

This DCT is used to de-correlate the data and return real-valued, time-domain representations of the signal power cepstrum. The most direct method for approximating the cosine is to use linear segments that closely approximating the curve from the range of [0, $\pi/2$],

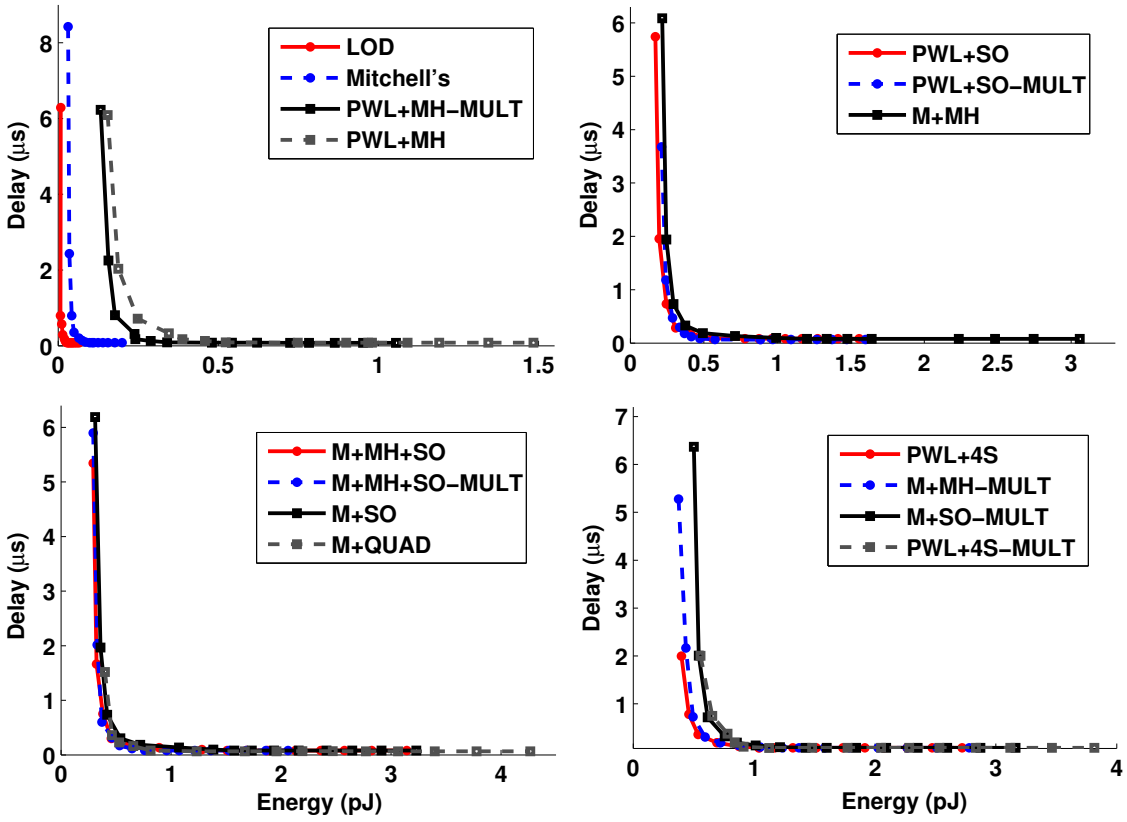


Figure 6.8: Measured energy-delay curves for all fabricated logarithm approximation circuits, grouped by their energy.

normalizing the input data to this range, and then computing the result using a multiply-add. In this section, another method is explored to reduce the complexity and power.

For a DCT of size N, the ranges for the parameters are $i \in [0, VAD\_BINS)$ and $j \in [0, MFCC\_BINS)$ where VAD_BINS is the number of triangular filters used for the processing. As there is an FFT the precedes the logarithm in the pipeline, there is a lookup table for the FFT twiddle factors that can be reused to determine the trigonometric values in this function. The only problem is that these twiddle factors are often indexed by an integer value multiplied by π/N. Due to the decimal term in the numerator of the argument in Equation 6.7, the identity on the right hand side is used to index through the twiddle factor LUT.

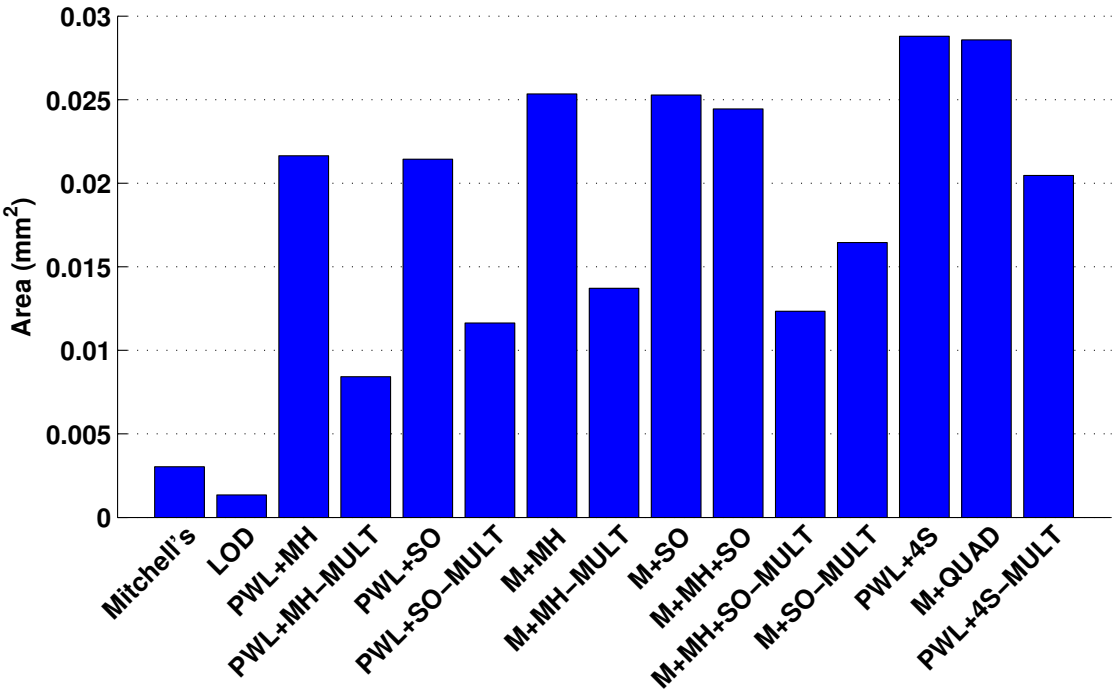


Figure 6.9: Area comparison for all logarithmic approximation methods.

6.7.1 Small Angle Approximation and Twiddle Factor Recycling

One of the arguments for the trigonometric functions in Equation 6.7, $j\pi/2N$, is always <1 for common values of j, i, and N. A very reliable approximation for small values of trigonometric function arguments is the small angle approximation. Equation 6.8 shows the result of the two terms containing arguments <1 being replaced by their small angle approximation equivalents.

$$\cos\left(\frac{j(i+0.5)\pi}{N}\right) = \cos\left(\frac{ij\pi}{N}\right)\left(1-\frac{\frac{j\pi}{2N}^2}{2}\right) - \sin\left(\frac{ij\pi}{N}\right)\left(\frac{j\pi}{2N}\right) \tag{6.8}$$

Since equation 6.8 contains an added multiplication due to the squaring, its of interest to approximate this term using a linear function. For small values of x^2, this is reasonable as the function appears linear in this range. This results in the modification to the full approximation shown in equations 6.9 and 6.10.

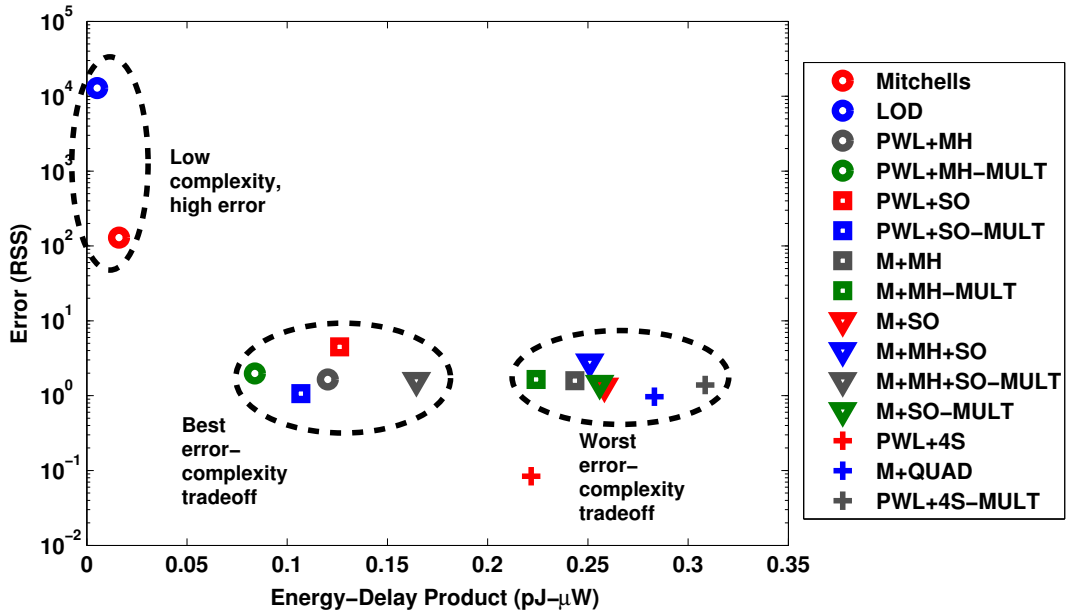


Figure 6.10: Comparison of approximation units to determine the method with the best measured error-complexity tradeoff at the nominal voltage (1.2V).

$$\cos\left(\frac{j(i+0.5)\pi}{N}\right) = \cos\left(\frac{ij\pi}{N}\right)\left(1 - \frac{\frac{j\pi 0.54}{2N} - 0.0477}{2}\right) - \sin\left(\frac{ij\pi}{N}\right)\left(\frac{j\pi}{2N}\right) \tag{6.9}$$

$$\cos\left(\frac{j(i+0.5)\pi}{N}\right) = \cos\left(\frac{ij\pi}{N}\right)\left(1.02385 - \frac{j\pi 0.27}{2N}\right) - \sin\left(\frac{ij\pi}{N}\right)\left(\frac{j\pi}{2N}\right) \tag{6.10}$$

To reduce the Hamming weight of the multiplication, 0.25 is used to reduce the number of multiplications. The final result is shown in equation 6.11.

$$\cos\left(\frac{j(i+0.5)\pi}{N}\right) = \cos\left(\frac{ij\pi}{N}\right)\left(1.02385 - \frac{j\pi 0.25}{2N}\right) - \sin\left(\frac{ij\pi}{N}\right)\left(\frac{j\pi}{2N}\right) \tag{6.11}$$

The comparison of the ideal, the approximation using a multiplier for the squared term, and the linear approximation are shown in Figure 6.11. The original approximation required

Approximation Method	Residual Sum of Squares (RSS) (Standalone Log)	Median % Correctly Classified Speech Frames	Std. Dev. % of Correctly Classified Speech Frames
Ideal Logarithm	-	84.33%	11.42%
Mitchells	129.208	84.19%	11.62%
LOD	12,835.660	84.31%	11.72%
PWL+MH	1.643	83.97%	11.48%
PWL+MH-MULT	1.983	84.30%	11.61%
PWL+SO	4.494	83.98%	11.40%
PWL+SO-MULT	1.067	84.23%	11.66%
M+MH	1.588	83.99%	11.55%
M+MH-MULT	1.649	83.97%	11.56%
M+SO	1.337	84.01%	11.50%
M+MH+SO	2.809	84.02%	11.51%
M+MH+SO-MULT	1.596	84.04%	11.51%
M+SO-MULT	1.453	84.04%	11.51%
PWL+4S	0.084	84.01%	11.49%
M+QUAD	0.967	84.05%	11.52%
PWL+4S-MULT	1.386	84.14%	11.51%
LUT (2000 Rows)	-	84.63%	11.61%

Table 6.4: Logarithm and Speech Pipeline Detection Accuracy.

four table lookups, two multiplications and a subtraction. The new approximation needs two table lookups, two multiplications, a shift, and two subtractions. The average error for the original implementation is 0.00029 while its 0.0015 for the approximated.

Using the small angle approximation for the DCT resulted in an average keyword detection accuracy degradation of 0.13% across noise levels ranging from 0-15 dB.

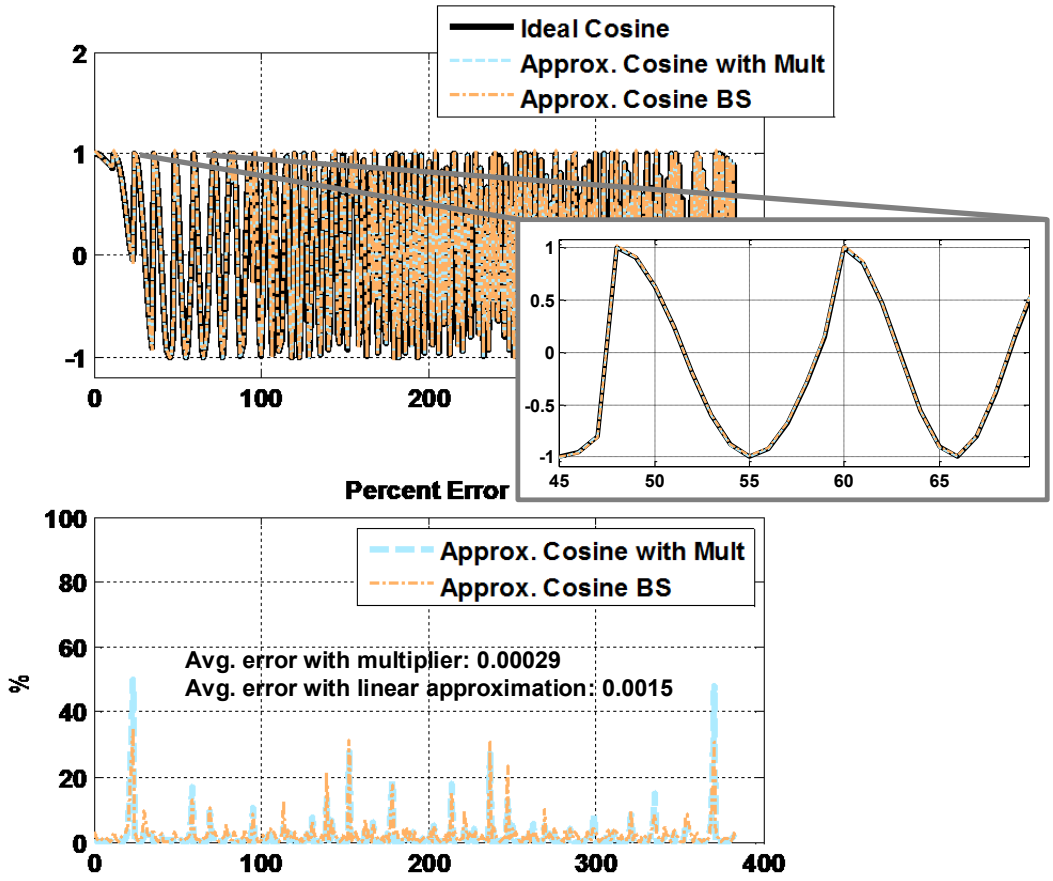


Figure 6.11: Resulting area due to the approximation of x^2 in the DCT

6.8 Mapping to the Slice Architecture

The aforementioned circuit design is for the ASIC implementation of the logarithm and DCT. The algorithm for voice activity detection features many repeated, parallelizable operations on vectors and matrices. Implementing these steps on a common reconfigurable DSP fabric will not only be more efficient in terms of energy, but also makes the system more robust to algorithm changes. If these design can be mapped onto a slice architecture containing various logic units shown in Figure 6.12, then this will reduce the hardware complexity and area.

Some of the proposed logarithm approximations are more suited for mapping to the slice than others. Mitchell's approximation is difficult to map due to the lack of barrel shifter and

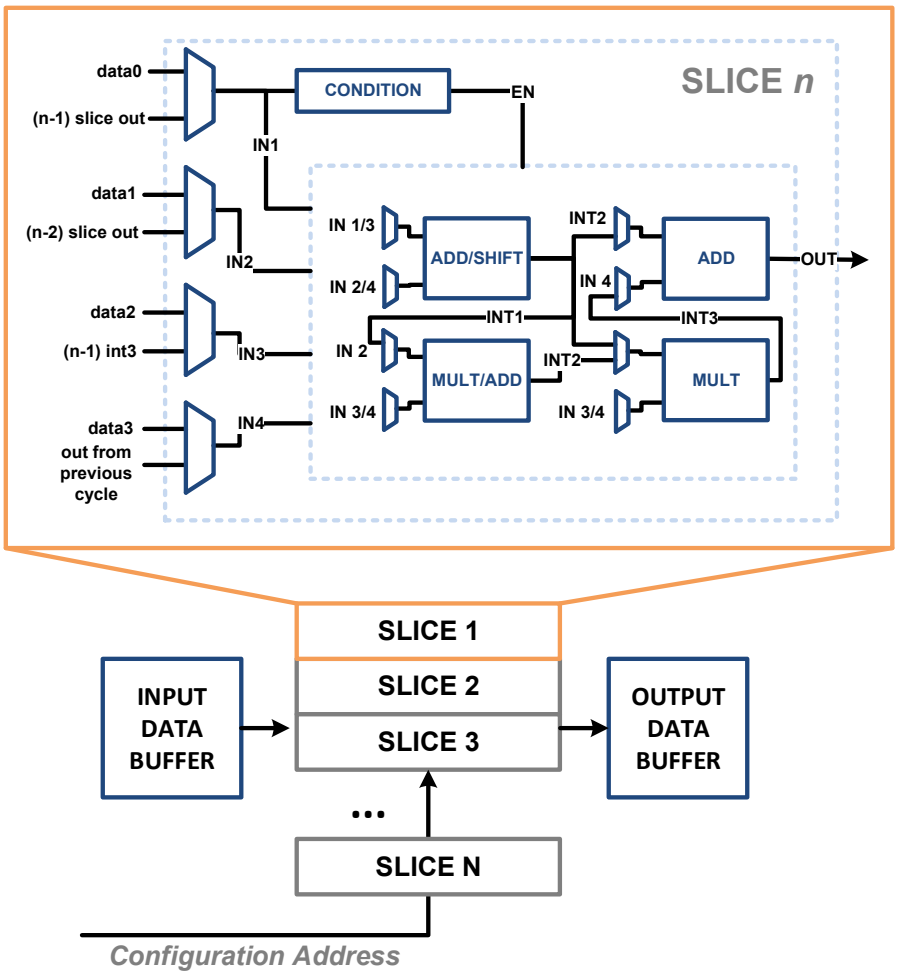


Figure 6.12: Slice architecture for implementing a flexible speech processing pipeline

bit selecting and checking within the proposed slice. Compensation methods work well for the slice as they require a multiplication and addition for each linear segment, but they also require heavy conditional checking to determine the correct slopes and intercepts based on the input data. The recommendation for the logarithm based on the system-level accuracy data is to create a LOD ASIC block and have the option to configure the input to the slices be the output of the LOD. Mapping both the DCT and logarithm to the slice architecture requires three total slices and is shown in Figure 6.13.

6.9 Conclusion

This work presents a methodology for designing low power blocks for a continuous listening VAD pipeline. This section focuses on the design of logarithmic approximation blocks using existing methods such as Mitchells and piecewise-linear approximation. They were implemented using methods to help reduce complexity and power such as Hamming weight and x-axis segment optimization. A novel, high-accuracy, and near-zero-average error quadratic compensation scheme is also presented. The results provide designers with avenues to modify new and existing approximation architectures in order to satisfy accuracy constraints while managing energy trade-offs.

This work also demonstrates that the most accurate block-level implementation does not necessarily lead to the most advantageous system-level accuracy results. Using software modeling of system components early in the design process can help determine error tolerances for feature extraction accuracy and lead to system-level power reductions without significant loss of accuracy. The need to reduce area by implementing the digital logic in a flexible slice architecture is seen here as there is much arithmetic overlap in the processing of MFCCs. This aims to motivate the next chapter where a ULP DSP fabric is designed to reduce redundant hardware.

This chapter discussed both direct and indirect power reduction techniques using DSP. The tradeoff between standalone logarithm accuracy and complexity showed the full range of implementations for this arithmetic block allowing for direct reductions in the system power. Considering the VAD pipeline, indirect DSP methods were applied to choose to lowest complexity implementation as standalone logarithm accuracy had little impact on system-level detection rates.

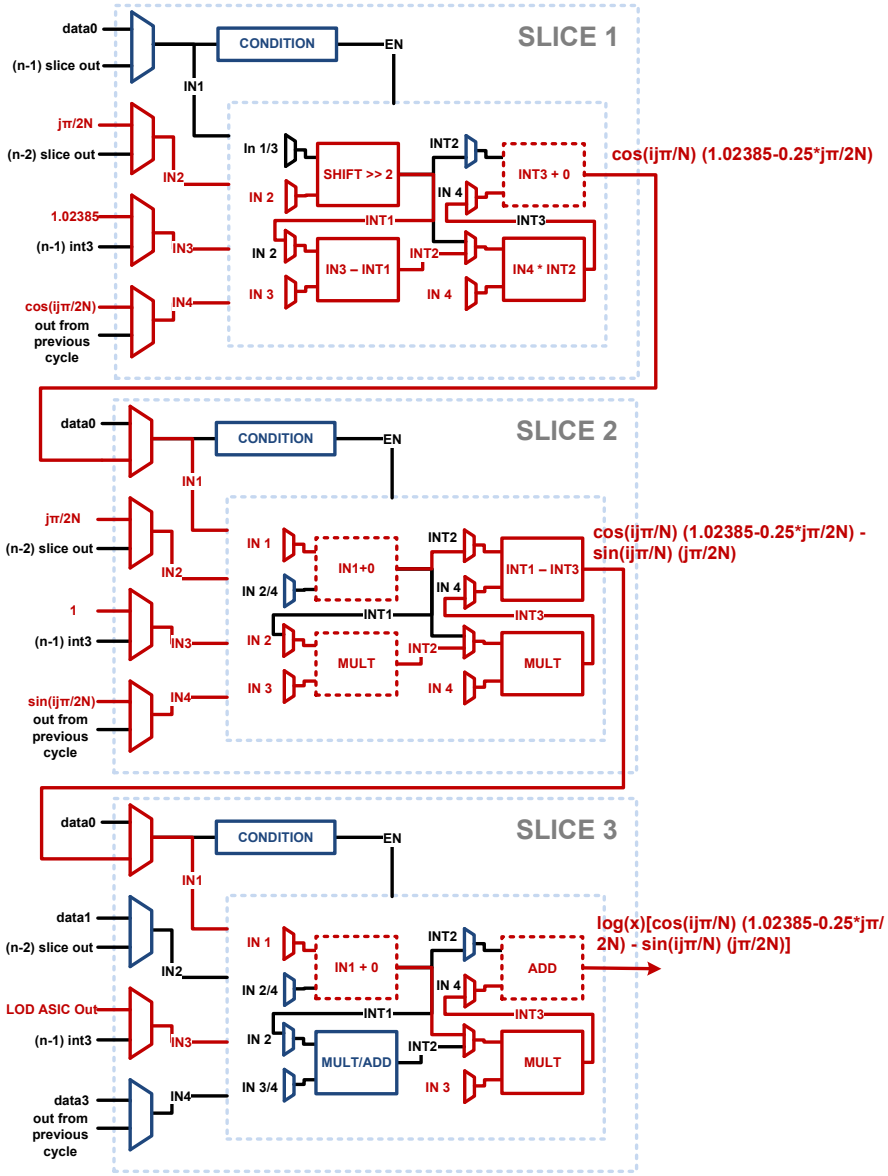


Figure 6.13: Mapping the logarithm operation followed by the modified DCT on the slice architecture

Chapter 7

A Flexible, Energy-Efficient DSP Accelerator with Minimized Hardware

In a typical low power SoC, such as those targeting biomedical applications, the range of sampling rates for sensing modalities such as glucose or speech spans hundreds of kHz as shown in Figure 7.1. SoC platforms that aim to target the biomedical or IoT applications space must have hardware that can accommodate applications receiving data at this full range of rates. Many energy-efficient SoCs rely on a variety of DSP accelerators including filters, FFTs, Discrete-Cosine Transforms (DCT), CORDICs, or pattern matching for processing data. Designing these blocks so that they're both energy-efficient and flexible across multiple data rates is a challenge as specificity and power reductions tend to go hand in hand. When considering the logical makeup of these accelerators, it can be observed that many contain redundant hardware such as addition, multiplication, and multiply-accumulate (MAC) blocks. Similarly, many real-valued transforms such as the DCT can be easily derived from the FFT, leading to the possibility of redundant control logic in an SoC. The combined area of the accelerators on the revision 1 node presented in Chapter 3 is shown in Figure 7.2 and they consume roughly 10% of the total chip area. From this figure it can also be seen that some of the individual accelerators are larger than a 2kB SRAM. For the operating modes and

applications presented in the case study in Chapter 3, it was rare to use multiple accelerators in parallel, and this is a strong motivation for consolidating the control and arithmetic logic into one, flexible low power DSP accelerator (lp-DSP). Little work has been completed on creating ULP DSPs, but work such as [73] has begun to consolidate classical DSP algorithms into a single accelerator for low power applications.

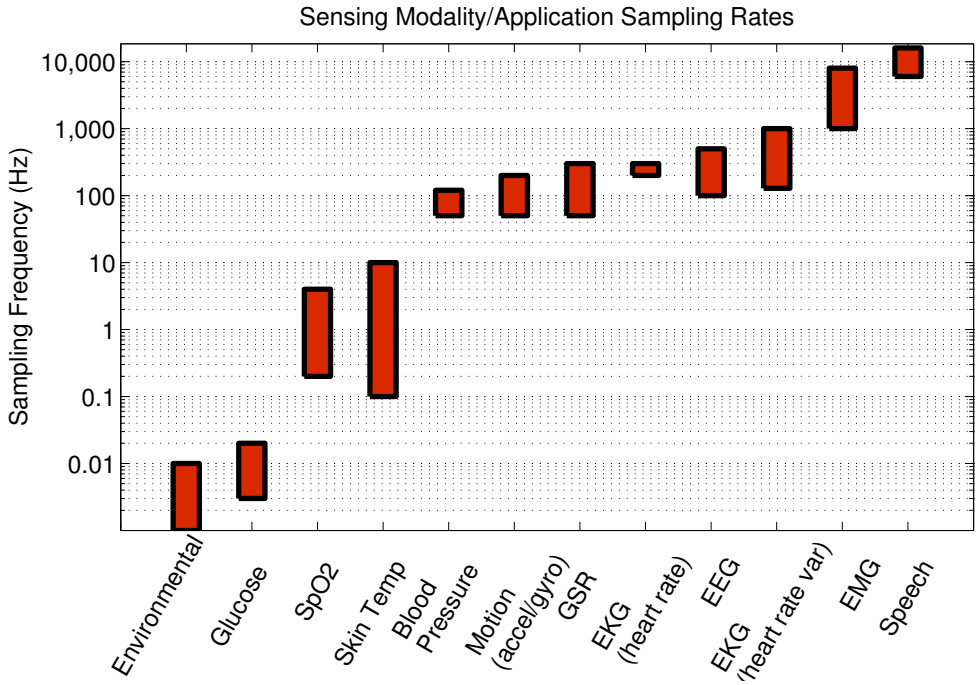


Figure 7.1: Typical sampling rate ranges for sensing modalities used in biomedical applications.

7.1 Introduction

A survey of recently published works targeting the IoT space was completed [19, 4, 74, 75, 76, 15, 77, 78, 79, 80, 81] to determine frequently used algorithms and arithmetic units used for on-node processing. The following list presents all processing units found in the literature and is categorized by computational similarity.

1. Inner-product based Algorithms: $y[n] = x[n] * h[n] = \sum_{n=0}^{N-1} x[i]h[n-i]$
   (a) Cross/Auto-Correlation

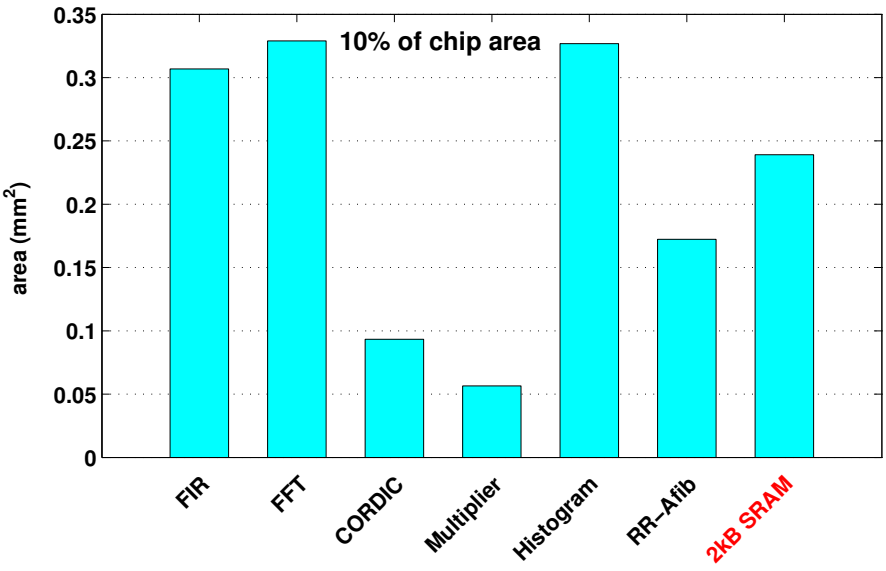


Figure 7.2: Comparison of total area of on-chip accelerators to a 2kB SRAM.

(b) Filtering (moving average, median, FIR, IIR)

(c) Fast-Fourier Transform

(d) Discrete Cosine Transform

(e) Wavelet Transform

2. Decimation Filtering
3. Summation/Integration
4. Derivative
5. Elementary function calculations (e.g. CORDIC)
6. Adaptive Filtering
7. Statistical

(a) Mean, Min, Max, Range

(b) Standard deviation, variance

(c) Spectral centroid: compute the FFT, X[n], with the center frequency of each bin being f[n]. $centroid = \frac{\sum_{n=0}^{N-1} f[n]X[n]}{\sum_{n=0}^{N-1} X[n]}$

8. Peak detection
9. Thresholding
10. Arithmetic: These are the blocks that would make up an ALU of primitive operators.

    (a) Multiplication
    (b) Division
    (c) Addition and Subtraction
    (d) Barrel Shift
    (e) Log and antilog
    (f) Two's Complement
    (g) Complex conjugate: This is a derived ALU calculation combining the two's complement converter.
    (h) Magnitude: This is a derived ALU calculation combining the logarithm, antilogarithm, multiplier, and barrel shifter: $\sqrt{x^2 + y^2} = \text{antilog}(\log(x^2 + y^2) >> 1)$.

Many of the algorithms in this list have overlapping control or computation schemes. For example, range is the difference between the min and max. The variance can be calculated from the mean, and the standard deviation can be calculated from the variance. Similarly, many of the algorithms repeatedly use addition/subtraction, multiplication, addition, and negation. The rest of this chapter seeks to determine how to utilize this overlap to reduce logical complexity and determine a set of shared arithmetic resources needed to compute these algorithms.

7.2 Algorithms

7.2.1 Inner-Product Based Algorithms

Fast-Fourier Transform The forward FFT is a method of computing the Discrete Fourier Transform (DFT). The DFT takes a signal and decomposes it into a set of scaled and shifted basis functions, which for the DFT are cosine and sine functions shown in Equation 7.1. Computing the DFT of N points in the naive way takes $O(N^2)$ arithmetical operations, while an FFT can compute the same DFT in only $O(NlogN)$ operations. An FFT rapidly computes the result by recursively factoring the DFT summation into DFTs of smaller sizes. The most well-known example of this is the Cooley-Tukey algorithm. This radix-2 algorithm can compute the same result with only (N/2)log(N) complex multiplications, assuming that N is a power of 2.

Many other algorithms have been introduced over the last few decades that attempt to minimize the computational complexity of the algorithm or minimize the number of multiplications and divisions. One example of this is the Winograd algorithm that requires few (if any for small N) multiplications and was proven to be the lower bound of multiplication operations for the FFT [82].

$$X[n] = \sum_{i=0}^{N-1} x[i] e^{\frac{-j2\pi in}{N}} \tag{7.1}$$

The inverse FFT can be easily computed using the forward FFT with additional pre and post-processing steps shown in Figure 7.3. This makes it easy to adapt an existing FFT algorithm to include this feature.

Algorithms to perform FFTs typically assume complex input and output data. However, many applications use only real-valued data in the time domain. A simple but inefficient solution to this problem is to zero-pad N-length real input signals with N-length zero-valued complex input components. Alternatively, it is possible to express a real-input DFT as a

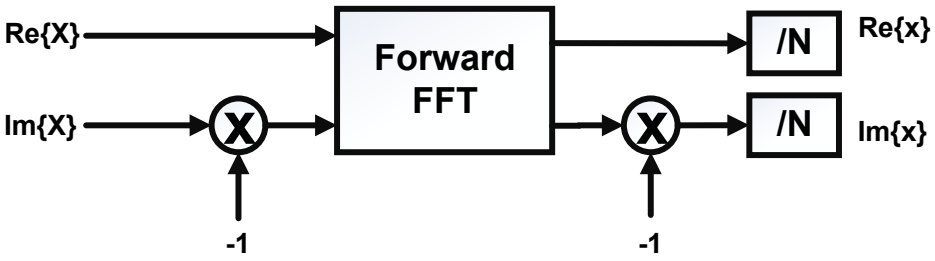


Figure 7.3: Computing the inverse FFT from the forward FFT

complex DFT of half the length followed by $O(N)$ post-processing operations. To compute the real-valued FFT, the following steps are taken (based on results from [83]):

1. Form the the N/2-point complex valued sequence, $x(n) = x_1(n) + jx_2(n)$, where $x_1(n)$ are the even indexes and $x_2(n)$ are the odd indexes.
2. Compute the N/2-point complex FFT on the "complex" valued sequence x(n) to obtain $X_0(k)$.
3. Perform an additional computation to get the final result, $X_1(k)$, from $X_0(k)$ shown below starting at Equation 7.2.

$$\begin{aligned} Re\{X_1(k)\} = Re\{X_0(k)W_0(k)\} - Im\{X_0(k)W_0(k)\} + \\ Re\{X_0(N/2-k)W_1(k)\} + Im\{X_0(N/2-k)W_1(k)\} \end{aligned} \tag{7.2}$$

$$\begin{aligned} Im\{X_1(k)\} = Im\{X_0(k)\}Re\{W_0(k)\} + Re\{X_0(k)\}Im\{W_0(k)\} + \\ Re\{X_0(N/2-k\}Im\{W_1(k)\} - Im\{X_0(N/2-k\}Re\{W_1(k)\} \end{aligned} \tag{7.3}$$

Both the above equations are for $k = 0, 1, ..., (N/2-1)$. Where the modified twiddle factors W_0 and W_1 are shown in Equations 7.4 and 7.5.

$$W_0(k) = \begin{cases} \frac{1}{2}\left(1 - \sin(\frac{2\pi k}{2N})\right) & \text{if } k \text{ is even} \\ \frac{1}{2}\left(\cos(\frac{2\pi k}{2N}) - 1\right) & \text{if } k \text{ is odd} \end{cases} \tag{7.4}$$

$$W_1(k) = \begin{cases} \frac{1}{2}\left(1 + \sin(\frac{2\pi k}{2N})\right) & \text{if } k \text{ is even} \\ \frac{1}{2}\cos(\frac{2\pi k}{2N}) & \text{if } k \text{ is odd} \end{cases} \tag{7.5}$$

Since this is just for the first $N/2 - 1$ points of the result, the remaining points are computed using the following relationships starting in Equation 7.6. These relationships exist due to the symmetric properties of the FFTs of real-valued signals.

$$Re\{X_1(N/2)\} = Re\{X_0(0)\}Im\{X_0(0)\} \tag{7.6}$$

$$Im\{X_1(N/2)\} = 0 \tag{7.7}$$

$$Re\{X_1(N-k)\} = Re\{X_0(k)\} \quad \text{for k = 1, 2, ..., N/2-1} \tag{7.8}$$

$$Im\{X_1(N-k)\} = -Im\{X_0(k)\} \quad \text{for k = 1, 2, ..., N/2-1} \tag{7.9}$$

Using the elementary function generator that's part of the proposed ALU, twiddle factors used for this post-processing can be generated to compute the RV-FFT of an input sequence. This reduces the overall complexity for real-valued FFTs on low-power platforms where the input data is rarely complex.

Short-Time Fourier Transform The Short-Time Fourier Transform (STFT) is useful in applications where a time-frequency representation of the data is needed. Typically, this requires the computation of multiple FFTs on overlapping frames of the input data, which can be costly and prohibitive due to the associated latency to compute the result. A method that computes a new array of FFT data per sample can be derived with N additional multiplications per new sample.

To derive a sliding Fourier Transform, we want to prove that given the k^{th} FFT of length N, the $(k+1)^{th}$ FFT of length N can be derived from its result with less computational

complexity than recomputing the full transform [84]. The transform is shown in equation 7.10.

$$X[n]_{k+1} = \sum_{i=0}^{N-1} x[i]_{k+1} e^{\frac{-j2\pi in}{N}} \tag{7.10}$$

Completing a variable substitution on i, for $p = i + 1$, results in equation 7.11.

$$X[n]_{k+1} = \sum_{p=1}^{N} x[p]_k e^{\frac{-j2\pi n(p-1)}{N}} \tag{7.11}$$

To extract the k^{th} FFT, the result needs to be embedded in equation 7.11. The N^{th} term of the summation can be factored out, and a $0th$ term can be added to the summation while also being subtracted from the total. This result is shown in equations 7.12 and 7.13.

$$X[n]_{k+1} = \left[\sum_{p=0}^{N-1} x[p]_k e^{\frac{-j2\pi n(p-1)}{N}}\right] + x[N]_k e^{\frac{-j2\pi n(N-1)}{N}} - x[0]_k e^{\frac{j2\pi n}{N}} \tag{7.12}$$

$$X[n]_{k+1} = e^{\frac{j2\pi n}{N}} \left(\left[\sum_{p=0}^{N-1} x[p]_k e^{\frac{-j2\pi np}{N}}\right] + x[N]_k e^{\frac{-j2\pi nN}{N}} - x[0]_k\right) \tag{7.13}$$

This results in the simplified form shown in equation 7.14. Within the parentheses is the k^{th} FFT added to the difference between the newest and oldest samples. For each transform, the update requires one complex addition that is common to all bins. It also requires a complex multiplication by a twiddle factor.

$$X[n]_{k+1} = e^{\frac{j2\pi n}{N}} \left(X[n]_k + x[N]_k - x[0]_k\right) \tag{7.14}$$

Re-computing the FFT for each new sample would require log(N) stages, so this provides a methodology for efficiently computing subsequent transforms.

Discrete Cosine Transform The DCT is similar to the FFT in that it decomposes the input signal into a set of scaled and shifted basis functions, but the basis function is strictly

a cosine. Due to this, it's a real-valued transform and is typically used for compression applications due to its energy compaction properties (much information for few coefficients). Dedicated and fast algorithms to compute the DCT have been proposed [85, 86], but their addressing schemes and control structures are specialized, preventing hardware reuse. The DCT of length N, X[n], can also be derived from the FFT of length N [87], Y[n], by reconstructing the DCT input sequence, performing an FFT, and completing post-processing of the output sequence shown in 7.15. This includes taking the real values of the FFT output and multiplying them by the real portion of the twiddle factors.

$$X[n] = Re[e^{\frac{-j\pi n}{2N}} Y[n]] \tag{7.15}$$

Where $Y[n]$ is the FFT of the reconstructed sequence, $y[m]$, derived from the original input sequence, $x[m]$ shown in 7.16.

$$y(m) = \begin{cases} x[2m], & m = 0, 1, 2, ..., N/2 - 1 \\ x[N - 1 - m], & m = 0, 1, 2, ..., N/2 - 1 \end{cases} \tag{7.16}$$

The additional computation required for this FFT-based DCT is N multiplications.

Wavelet Transform (WT) The wavelet transform has applications in compression, acoustics processing, ECG, and gait analysis. Similar to the FFT in that it's an inner-product-based algorithm, the WT decomposes a time series into time-frequency phase space by convolving the input signal ,$x(t)$, with the mother wavelet function, $\Psi(t)$. From an applications perspective, the mother wavelet function is chosen based on the feature to be extracted from the input signal. Common examples of mother wavelet functions include Haar and Morlet.

Similar to the DCT, work has gone into dedicated CWT implementations in hardware [88, 89], but finding an FFT-based implementation would allow for hardware re-use at the expense of the energy-efficiency of this algorithm. Implementations such as [90] re-purpose

the FFT for implementing the CWT, and the analysis discussed here is based on this implementation.

7.2.2 Statistical Processing

For this work, basic statistical processing methods are used for computing first-order statistics such as the mean, min, max, and range. Similarly, computing the standard deviation and variance are based on textbook definitions.

7.2.3 Arithmetic

Many of the derived algorithms such as square root or division are implemented in the log domain to increase hardware reuse and reduce power. The logarithms developed in Chapter 6 were used in the lp-DSP ALU, particularly the logarithm that contained the best energy-delay-error tradeoff, which was the piece-wise linear multiplierless method. Using logarithms to compute roots and division also requires an antilog circuit to convert back to the linear domain. An antilog circuit was developed based on Mitchell's algorithm for the antilog that also included linear interpolation of the error to match the accuracy (average absolute error) of the logarithm.

7.3 Architecture

At the architecture level, the design knobs include the amount of shared memory, number of arithmetic units, interconnect, and the access pattern kernels used to implement the logic of the DSP algorithms. At the micro-architecture level, pipelining and parallelism were used as design tuning variables. At the logic and arithmetic level, the choice and number of arithmetic units was determined based on application constraints and the adder/multiplier topologies were left to the discretion of the tool based on a target voltage and frequency point.

7.3.1 Arithmetic Resources

The arithmetic units used at the core of every DSP unit impact the power, area, and latency of each algorithm. As all of these units are combinational, the amount of available timing slack must be known for implementation of the access pattern kernel state machines discussed in the next section.

Considering the list of typical DSP tasks completed in ULP SoCs, the following arithmetic building blocks are frequently needed for processing: adder/subtractor, multiplier, divider, barrel shifter, log/antilog, and two's complement. There are a variety of topologies for implementing each of these arithmetic blocks that depend on the throughput demands, bit-width constraints, and power. Synthesized multiplication, addition, and division blocks (i.e. Verilog: x = a * b) were compared against barrel shifter and logarithmic based implementations for the multiplication and division cases. Cadence RTL Compiler Tcl scripts were created to synthesize these building blocks at TT, 25°C assuming a target voltage of 500mV using a high-V_T standard cell library and constrained to have a propagation delay of 2μs (500kHz). Power numbers in this analysis were gained from simulation results in Spectre, while area and slack time were taken from Encounter post-P&R (Table 7.1). The leakage numbers reported in the table are when the block is being power gated using a PMOS header.

Both the 16 and 32-bit synthesized divider circuits (i.e. Verilog: x = a / b) were not able to meet timing at 500 kHz even after multiple attempts at timing optimization. The 16-bit divider circuit had a slack time of -2806 ns, leading to the development of an alternative. The logarithm-based multiplier/divider circuit is the largest and most power hungry of the arithmetic units, but meets the timing specifications for the design. These results impact the design of the access pattern kernels that are used to implement the control logic. These kernels implement the state machines for the control logic and must know the slack time of all arithmetic units to determine the number that can be used per clock cycle.

Block	**Area (μm^2)**	**Worst Slack (ns)**	**Active Power (nW)**	**Leakage Power (pW)**
Adder (16-bit)	182.5	575.8	4.4	25.7
Adder (32-bit)	510.7	11.5	16.8	131.0
Barrel Shifter (16-bit)	509.0	1120.0	10.4	57.3
Barrel Shifter (32-bit)	1230.2	1025.9	38.6	101.7
Two's Comp. (16-bit)	136.6	1448.6	3.6	37.6
Two's Comp.t (32-bit)	390.8	266.8	12.3	67.8
Log_2 (16-bit)	5462.0	1.9	282.5	563.0
$Antilog_2$ (16-bit)	2104.2	20.7	41.1	157.4
Synth. Multiplier (16-bit)	3656.9	14.0	152.8	399.4
BS Multiplier (16-bit)	3545.9	27.0	146.4	327.4
Log Mult/Div (16-bit)	15104.0	16.7	1820.5	108,900.0
Log Mult/Div (32-bit)	23035.5	9.4	2391.5	140,100.0

Table 7.1: Summary of key metrics for lp-DSP arithmetic units

7.3.2 Fast-Fourier Transform Architectures

Much work has gone into implementing the FFT in an efficient manner; it's symmetric nature lends well to implementations of order $NlogN$ as was introduced in [91]. As discussed above, most inner-product-based algorithms can be implemented using the FFT with post-processing. To increase hardware re-use, this class of algorithms will be implemented using an efficient FFT core.

Winograd mathematically proved a way to minimize the number of multiplications required to compute an FFT in 1976 [82] and this was later translated into a a set of compute-friendly steps by Silverman in 1977 [92].

This method is only efficient up to small sizes (i.e. 16), such that making FFTs of larger sizes requires repeatedly using smaller FFT building blocks. For this work, we used three small Winograd FFT kernels of size 4, 8, and 16. It has been shown that an FFT of size N can be computed using two FFTs of sizes N_1 and N_2 where $N = N_1 * N_2$ in equation 7.17

[93]. This is equivalent to the FFT of size N for $0 \leq k_1 \leq (N_1 - 1)$ and $0 \leq k_2 \leq (N_2 - 1)$.

$$X[k_1 N_2 + k_2] = \sum_{n_1=0}^{N_1-1} \left[e^{\frac{-j2\pi n_1 k_2}{N}} \left[\sum_{n_2=0}^{N_2-1} x[n_2 N_1 + n_1] e^{\frac{-j2\pi n_2 k_2}{N_2}} \right] \right] e^{\frac{-j2\pi n_1 k_1}{N_1}} \tag{7.17}$$

This is equivalent to "rearranging" the input data, x, into a matrix of size N_1 x N_2 shown in equations 7.18. The FFT of each row of length N_2 is taken in place. Each element of A is then multiplied by the twiddle factors $T(n_1, n_2) = e^{\frac{-j2\pi n_1 n_2}{N}}$.

$$A_{n_1,n_2} = \begin{pmatrix} x(0) & x(N_1) & x(2N_1) & \cdots & x(N - N_1) \\ x(1) & x(N_1 + 1) & x(2N_1 + 1) & \cdots & x(N - N_1 + 1) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x(N_1 - 1) & x(2N_1 - 1) & x(3N_1 - 1) & \cdots & x(N - 1) \end{pmatrix} \tag{7.18}$$

Take the FFT of each of the column of length N_1 in place, and the output data, X, will be arranged in A in the order shown in 7.19.

$$A_{n_1,n_2} = \begin{pmatrix} X(0) & X(1) & X(2) & \cdots & X(N_2 - 1) \\ X(N_2) & X(N_2 + 1) & X(N_2 + 2) & \cdots & X(2N_2 - 1) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ X(N - N_2) & X(N - N_2 - 1) & X(N - N_2 - 2) & \cdots & X(N - 1) \end{pmatrix} \tag{7.19}$$

Compared to the direct computation of the FFT, there is the added overhead of N twiddle factor multiplications and control logic required for the memory management. Using this methodology, three highly efficient FFT kernels can be used to directly compute FFT sizes of 32, 64, 128, and 256. FFTs of greater size (e.g. 512, 1024) can be computed by factoring into multiple FFTs of sizes 4, 8, and 16 with the added penalty of control logic. Despite the addition of the twiddle multiplications, the overall number of multiplications is reduced.

The Winograd FFTs of sizes 4, 8, and 16 were implemented in Verilog HDL and synthesized at 500mV and 500kHz. Two architectures were compared: one with local, register-based storage (high-throughput), and a serialized implementation using a two-port memory structure where up to 4 words of data could be retrieved in each clock cycle. The serialized algorithms were completed in-place such that only N memory spaces were required for an FFT of size N. [1]

Size	**Clock Cycles**	**# Adders**	**# Mult.**	**# 2's Comp.**
4	10	4	0	2
8	38	4	0**	2
16	94	6	6	4

Table 7.2: Comparison of Winograd FFT implementations of sizes 4, 8, and 16 (arithmetic resources)

Size	**Area (par.)** (μm^2)	**Area (ser.)** (μm^2)	**Area Reduction**	**Active Power (nW)**	**Leakage Power (pW)**
4	5624	2468	2.3x	94.8	661
8	17688	4696	3.8x	103.0	779
16	60508	7171	8.4x	131.0	885

Table 7.3: Area and power comparison of Winograd FFT implementations of sizes 4, 8, and 16

7.3.3 Memory

Many recent ULP SoCs have relied on SRAMs to implement memory functionality on chip. SRAM macros are a common choice due to their density and energy efficiency for larger memory sizes typically used for data and instruction memories. To allow for ULP chip operation, most BSN SRAMs operate in the subthreshold regime, which reduces overall power but introduces challenges related to robustness and leakage. During power outage

[1]** Winograd 8 includes a dedicated multiplier circuit that can only multiply by 0.707

events, SRAM-based memories lose state, which can mean the loss of important medical data, chip state, and instructions. In this case, the chip must also be reprogrammed, which is inconvenient for longitudinally deployed systems.

An alternative to SRAM-based memories is using commercial, non-volatile memory (NVM) options such as Flash or EEPROM, but these require high read/write voltages and have large peak current demands. Alternative NVM solutions exist that are promising for ULP designs but are either not yet commercially available or still require high read/write voltages.

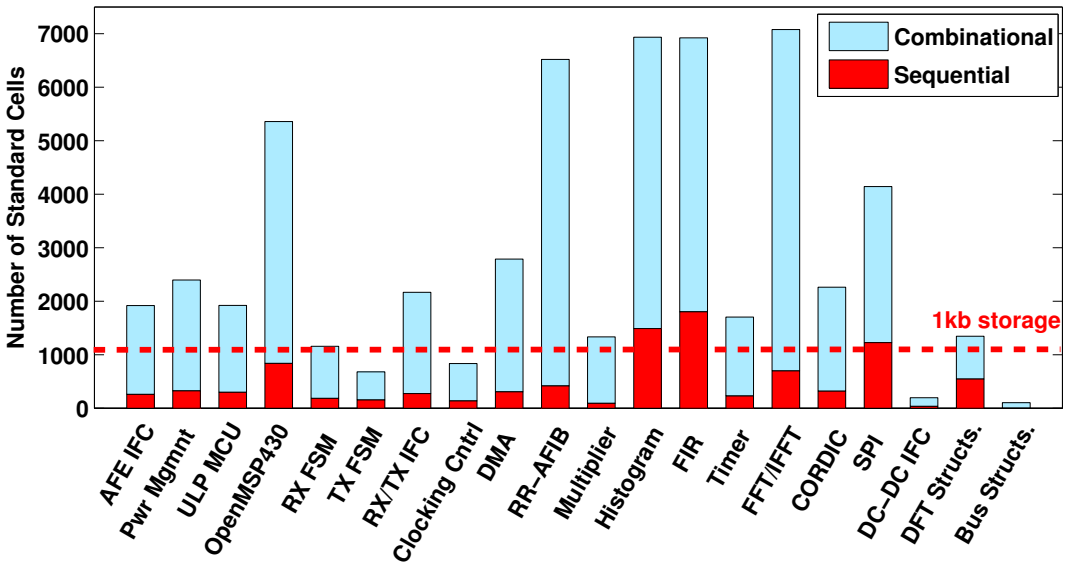


Figure 7.4: Comparison of total standard cells in each BSN sub-block.

Since all memories are not large (>1kb) on an SoC, standard cell-based memories (SCMs) synthesized using registers and latches must be considered for optimal energy and area efficiency for small-capacity memories. SCMs are easily integrated into digital blocks during synthesis without the need for extra power rings, reducing the overall area. Figure 7.4 shows the number of combinational and sequential standard cells on various blocks on the revision 1 SoC presented in Chapter 3. In this case, sequential elements account for 17% of the total standard cell count and more than 40% of the digital chip area (not including SRAMs). Based on [94], blocks containing >1kb memory can benefit from SRAM-based storage, while blocks <1kb see power benefits using latch-based storage. The example in 7.4 shows that

some BSN accelerators with >1k sequential standard-cells can benefit from the integration of SRAM-based memories. This makes early design space exploration based on application requirements crucial to avoid excessively high power and area implementations.

In a 65nm CMOS technology, the target for the lp-DSP design exploration, 8T and 10T SRAM bitcells have areas in the range of 0.9-1.7μm^2 [95, 96]. A minimum sized standard cell latch has an area range of 4.32-4.68μm^2 while a minimum sized standard cell register has an area range of 6.84-7.94μm^2 (depending on the polarity of the clock). This is an area overhead of 2.5-2.7x for latch-based memories. Based on area estimates from Cadence's RTL Compiler after synthesizing latch-based register files of 32x16, 64x16, 128x16, and 256x16, the logic overhead is consistently 1/3 of the total area of the memory. For two port memories, the logic overhead is roughly 2/3 of the total area. Results of this exploration can be seen in Figure 7.5. Each memory kernel was synthesized including two power gate (PG) control signals for sleep mode. One PG signal controls an array of PMOS headers that are 64x the minimum transistor width, while the other PG signal controls an array of PMOS headers that are 4x the minimum transistor width. For active operation of the memory kernels, both headers are "on" allowing the maximum amount of current to be delivered to the block. For a low power data retention mode, the 64x headers can be turned "off" to minimize current going to the block and reduce leakage. This leads to an average of 1.7x reduction in current consumption compared to clock gating.

A global memory controller is used to aggregate the memory kernels into a larger memory bank. The controller has the capabilities to clock and power gate individual memory kernels when they're not needed for the current or future operations. The controller also has multi-port capabilities to complete simultaneous read/writes from all memory kernels. The block diagram of the global memory controller is shown in 7.6.

The choice of memory kernel size is heavily dependent on the application. For a given application, there will be a variety of data "frames" used for computation. For example, a 16-tap FIR filter requires 16 input data for processing to occur. Similarly, a 64-pt, complex

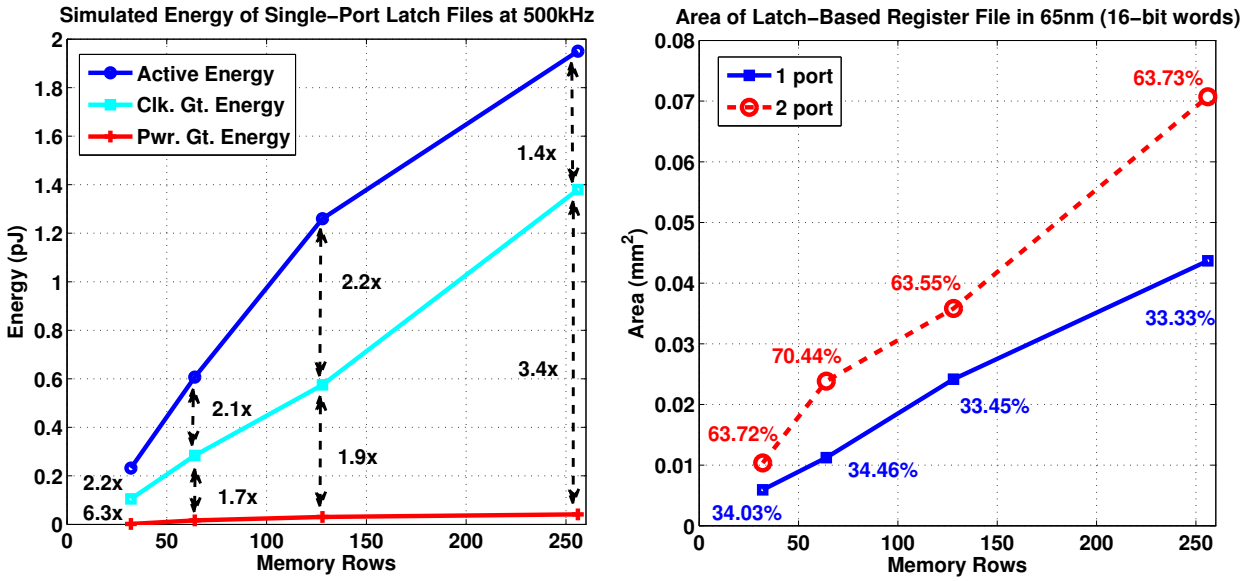


Figure 7.5: Energy (left) and area (right) comparison of latch-based register files in 65nm CMOS. Figure on right shows percent area due to addressing logic. Results taken at 500mV, 500kHz

FFT requires 128 input data. The frame sizes in these examples are 16 and 128, respectively. Determining the most common frame size for the application will reduce leakage energy due to the memories as the global memory controller can power gate the unused kernels.

7.3.4 Throughput

With the memory architecture explored, it was clear that increasing the number of memory ports leads to a large percentage increase in memory kernel area. The fewer ports for each memory, the fewer data samples that can be accessed at once from one of the access pattern kernels. This leads to a system throughput that's dependent on the access to memory and is the bottleneck for the performance of the lp-DSP.

To determine the required throughput for a give application set, the following design parameters should be considered: sampling frequency, number of processing elements per sample and the number of clock cycles per processing element (e.g. an FIR followed by an FFT need to be computed on data), and the serial/parallel nature of the processing

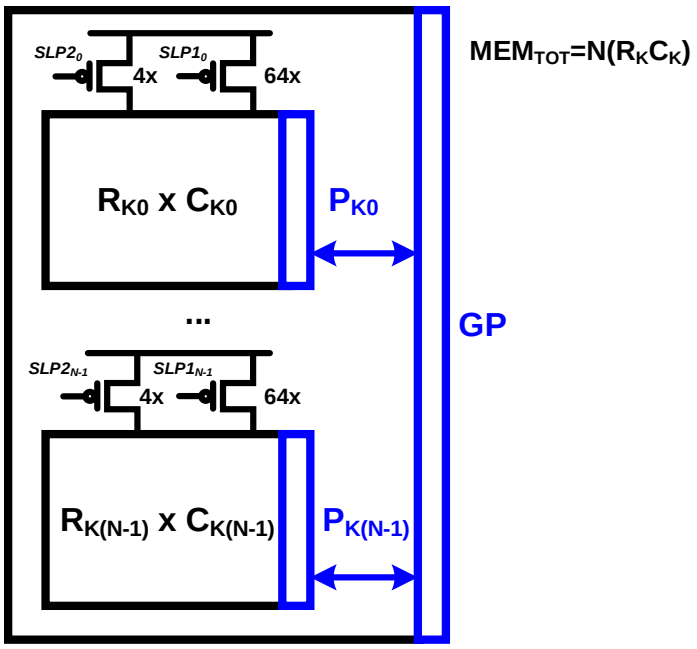


Figure 7.6: LP-DSP memory architecture using memory kernels with P_K read/write ports connecting to a global memory controller with GP read/write ports

units. Knowing the longest critical path that's required for the applications can inform the architecture of the lp-DSP.

7.3.5 Access Pattern Kernels (APKs)

To implement each algorithm described in this chapter, the control logic had to be described in Verilog omitting the memory and arithmetic units that are typically internal to the block. These control logic blocks determine the access patterns to the memory and arithmetic units in order to implement a DSP algorithm. The proposed architecture for the lp-DSP is shown in Figure 7.7, and here the APKs sit between the memory and arithmetic.

The designs were synthesized and placed and routed and then simulated using Spectre. The area and power results for the APKs not including the area/power of the memories or arithmetic units is shown in Table 7.4. This table is not an exhaustive list of APKs that were described at the beginning of this chapter for the IoT space, but rather a small set to

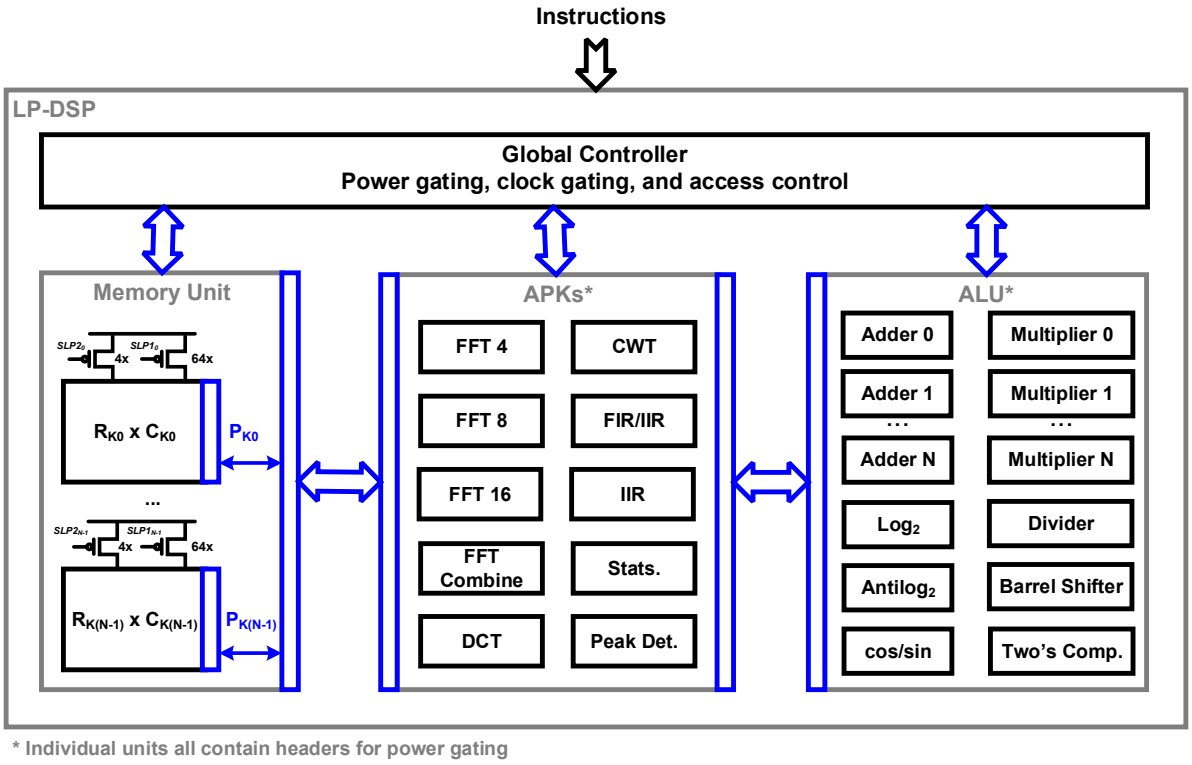


Figure 7.7: Proposed architecture for the lp-DSP

show the power and area savings that can be achieved through exploiting logic overlap and resource sharing between algorithms.

From this, it's seen that the control logic consumes very little power relative to the full implementation of each independent block at the target operating point of 0.5V and 500kHz. These APKs are orders of magnitude smaller than the accelerators described in Figure 7.2 that were on revision 1 of the BSN node discussed in Chapter 3.

7.4 Results and Conclusions

All designs were implemented using Verilog HDL and synthesized using Cadence's RTL compiler. A high-V_T standard cell library was characterized at 500mV (subthreshold) in a commercial 65nm technology and used for the synthesis process. To improve design robustness and minimize V_{MIN}, specific standard cells were pruned from the synthesis process. This

Name	**Clock Cycles**	**Area (μm^2)**	**Power (nW)**
FFT Combine	(var) 43-263	15330	95.9
RV-FFT Combine	FFT/2 + N	6211	42.7
DCT	FFT + N	5003	39.5
FIR	num. taps	6628	58.0
IIR	num. taps	7490	69.3
Running Avg.	window size	2924	26.6
Min/Max	N/A	1744	18.4
Variance	window size	3371	34.1

Table 7.4: Metric comparison of APKs

included gates with large stacks (e.g. NOR3) or transmission gate-based logic cells (e.g. XOR3, MUX3) with more the two inputs, as the stacked transmission gates cannot produce enough gain at low voltages, potentially resulting in ambiguous voltage levels in the face of variation [12]. Early analysis of memory, control, and arithmetic units were completed using the parameter estimation techniques discussed in Appendix C.

There are many variables in this scheme that are dependent on the target application. The number of memory ports, serial nature of the APKs, and the number of each arithmetic unit all depend on the desired throughput of the system. This exploration is intended to be a proof of concept of the idea that there is a middle ground between the specificity of hardwired accelerators and the flexbility of GPPs in the emerging area of flexible SoCs for the IoT. The goal of the work is to provide the framework for an architecture where the elements can be mixed and matched for the intended application specifications. As it's rare to use more than one DSP accelerator in parallel for a typical BSN application, distributing the memory resources and arithmetic units reduces the overall area (and leakage) required for accelerated hardware, while exploiting the mathematical overlap between algorithms increases the processing flexibility.

Chapter 8

Conclusions

Wireless sensor nodes and other ULP systems demand energy efficient signal processing to meet their stringent power requirements. To enable batteryless operation on these platforms, they must consume an average power less than the power harvested, which is typically in the μW range. The processing blocks are power limited to keep the total average chip power below the harvested power and avoid depleting energy on the storage capacitor. This dissertation focuses on improving power consumption on SoCs where energy-efficiency is the primary design constraint through novel DSP techniques at the algorithm, architecture, and circuit levels. By completing a top-down evaluation of the system specifications from the application down to the circuits, opportunities for power optimization are revealed. In this work, system power is reduced through each step of the hierarchy while still meeting system-level specifications and accuracy constraints.

This work looked at a case study of a ULP SoC targeting the biomedical and IoT spaces that consumed μW of power while harvesting energy from the environment. This design has strict power demands that motivate the need for aggressively scaled power of individual circuit components. A ULP signal power extractor was designed for this platform containing a four-channel FIR filter and digital envelope detector having the lowest FOM compared to the state of the art. This work also aimed to indirectly reduce system power by reducing the reliance on

high-leakage system elements such as the subthreshold SRAMs by designing an approximate coefficient block for nW of power. Considering an application that is more dominated by the DSP power, speech processing and VAD were evaluated component-by-component for power reduction opportunities within each functional unit. The analysis yielded low power design techniques for logarithms and DCTs when computing MFCCs for speech. Finally, this thesis presented a methodology for developing a flexible, low-area DSP fabric that is between GPPs and dedicated accelerators on the flexibility versus power spectrum.

Each chapter in this work showed many examples of both direct and indirect power improvements. Generalizing many of the techniques from the chapters in this work is beneficial for system designers looking to reduce overall system power but wanting to have a large system-level impact. Figure 8.1 shows a flowchart for determining system suitability for either direct or indirect reductions. From this chart, the designer can iterate between making direct and indirect power reductions using DSP until the specification is met.

Using this chart, the techniques from this dissertation can be broadly applied to other systems targeting ULP operation. This work has also shown that the potential system-level power impacts coming from DSP are not limited to direct optimization.

8.1 Summary of Contributions

8.1.1 ULP SoC

- Created a ULP wireless SoC with integrated power management, transceiver, sensing interfaces, DSP, and flexible clocking.
- The platform achieved the highest level of integration, including energy harvesting and a full transceiver, for the lowest power on both versions of the node.
- Included flexible sensing interfaces and on-chip processing for a diverse set of target applications.

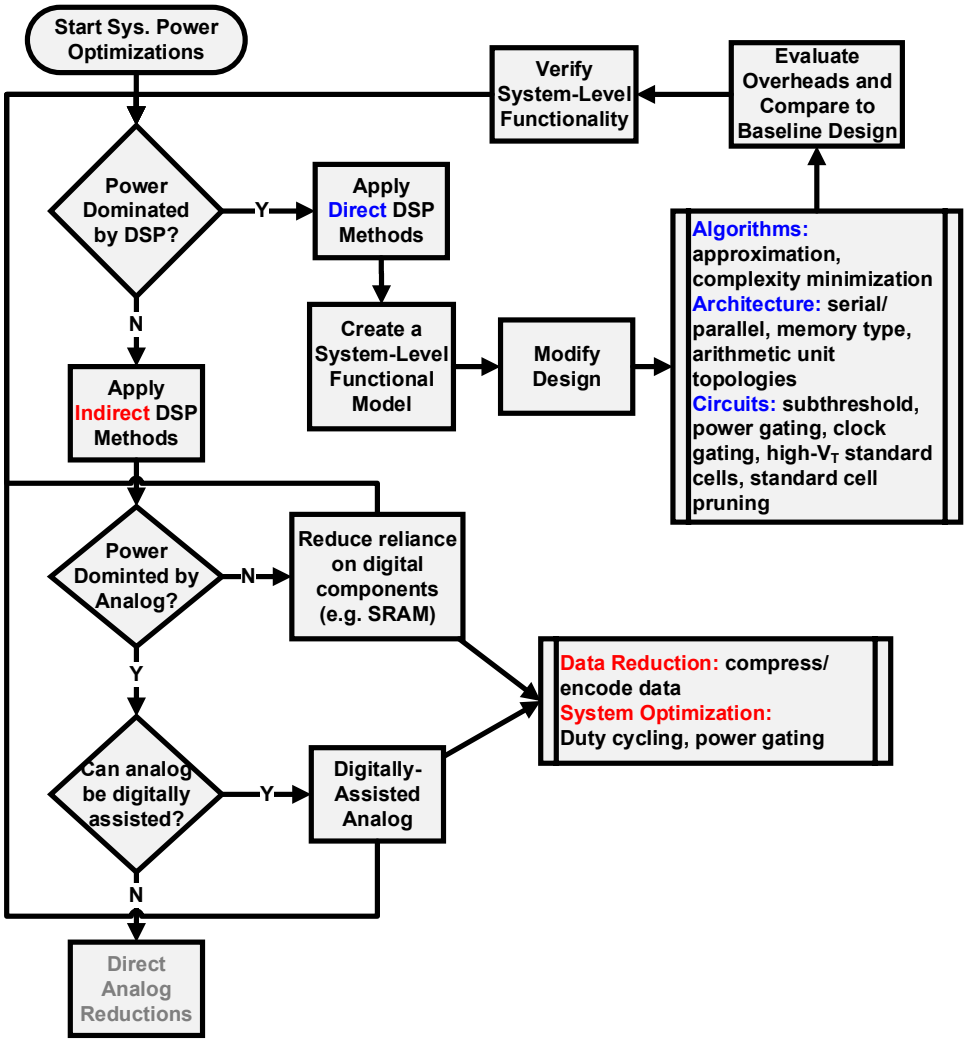


Figure 8.1: Methodology for directly and indirectly reducing system power using DSP

- Incorporated autonomous power management for battery-free operation on both versions of the node.

- The system achieved the highest energy harvesting / regulation efficiency on the revision 1 node.

8.1.2 FIR Filter for BSN SoC

- Designed a synthesizable, sub-threshold, four-channel signal band power extractor for a batteryless body sensor node SoC.
- The filter and full extractor circuit are the lowest power designs compared to the state of the art.
- The filter is flexible for use in general purpose DSP applications.
- By serializing the traditional FIR architecture, the number of adders and multipliers required for the design was reduced, thereby reducing active and leakage energy for the overall design.
- The SPC successfully uses power-of-two format to reduce computational complexity, area, and power for little loss in data fidelity.
- Indirectly reduced system power by replacing analog implementation of signal power extractor.
- Lowest FOM to date for ULP FIR filters.

8.1.3 Approximate Coefficients

- Provides flexible and scalable, on-chip filtering for ULP applications.
- Reduces the dependence on on-chip SRAM for storing coefficients.
- Provides runtime modifications of filtering specifications.
- Reduces on-chip area for large number of filtering coefficients.
- Eliminates reprogramming of filtering coefficients after system power-down event.
- Completed break-even analysis to determine the point at which this method is advantageous over using traditional storage methods.

8.1.4 Logarithmic Approximation for a VAD Pipeline

- Determined that logarithm accuracy in a VAD has low impact on overall accuracy of keyword detection, allowing for the use of the lowest power implementation.
- Evaluated existing logarithmic approximations and developed novel methods for approximation.
- Developed the first zero average error logarithmic approximation using quadratic interpolation.
- Reduced the complexity of the DCT by re-purposing the FFT twiddle-factor ROM.
- Completed system-level analysis of the impacts of block-level error (log, DCT, FFT, triangular filters) on keyword detection accuracy.

8.1.5 lp-DSP Fabric

- Surveyed the low power SoC literature for common DSP algorithms and completed classification of their processing tasks to minimize hardware.
- Determined hardware overlap between DSP tasks to maximize hardware reuse.
- Developed a novel architecture for digital signal processing on ULP SoCs.
- Evaluated arithmetic units for their power, area, and latency in the subthrehsold region.
- Developed a methodology for incorporating low power synthesized memories.
- Designed APKs for interfacing between the arithmetic and memory units.

8.2 Team and Individual contributions

The BSN project presented in Chapter 3 was a massive collaboration between three universities over the span of five years. For version 1, we worked with the University of Washington to

complete the node. Their contributions included the AFE, PMU, transmitter, and clocking. Five students from The University of Virginia completed the ULP digital processor (Yanqing Zhang, Yousef Shakhsheer, Jim Boley, Aatmesh Shrivastava) led by Yanqing Zhang and Yousef Shakhsheer. For this version, I was less involved in the system-level planning and architecture and focused on the design of the signal energy extractor presented in Chapter 4.

Revision 1 of the BSN node was a collaboration between the University of Michigan and the University of Virginia. Students at Michigan completed the design of the transceiver, crystal oscillator, and ADPLL circuits. They also developed the specifications for the subthreshold baseband processors (i.e. Tx/Rx FSMs). Many students at the University of Virginia were involved with other components of the design including Yousef Shakhsheer (LCU, DPM, and system organization and architecture), Aatmesh Shrivastava (PMU), Jim Boley (SRAM), Kyle Craig (synthesis and top-level integration), Patricia Gonzalez (CORDIC, system organization), Divya Akella (DMA), and Yanqing Zhang (AFE pin pruning and organization). As mentioned in Chapter 3, I was involved in the system organization and architecture as well as the design of many DSP components including the Tx/Rx baseband, FIR, FFT, and histogram.

The voice activity detection work in Chapter 6 was completed during an internship at Intel Corporation during the summer of 2013. The context of the problem was provided by my manager, Jim Tschanz, and colleague, Joe Ryan. The speech pipeline software model, including the keyword detection backend, was developed by researchers within Intel, but provided to me for modification based on new arithmetic approximations.

8.3 Open Problems

8.3.1 ULP SoC

The BSN node is a very large system involving many components that each have individual research questions. Considering the system, there is much work to be done in evaluating

tradeoffs from the application level down to the architecture and circuits. Completing more system-level analysis prior to tapeout of the nodes is what the project needs moving forward.

From a circuit-level perspective, developing a methodology to determine the system-wide minimum energy point (MEP) before design time would be beneficial. The 500mV, 200kHz operating point that was chosen for the BSN chip was arbitrary and was not based on energy minimization. It has been shown that the MEP is dependent on the workload and duty cycle of a circuit [97] such that the operating conditions of the block have an impact. Similarly, many complex digital circuits have different proportions of path lengths (critical to non-critical) leading to more/less leakage contributions at the maximum frequency. This causes the MEP to shift for each block, and means the the system-level MEP has the potential to change on a per-application basis as some blocks are used more frequently in certain applications than others. Choosing an operating voltage the minimizes the energy for the system across a set of applications and not just the individual block would improve overall lifetime.

Another area that could be improved is block-level floorplanning for area minimization given a number of power domains. For revision 1 of the node, most of the digital peripherals had three power domains and the power domain floorplanning was always 1 row by 3 columns for all blocks. In many instances, one of the block's power domains was very large compared to the area of the other two domains combined. This led to large amounts of wasted space (filler cells) being inserted within the under-utilized power domains resulting in wasted area. Finding a way to determine the best power domain arrangement on a per-block basis based on the results of RTL Compiler's estimates on area would be useful.

There are also many block-level modifications that could be added to the SoC. The revision 1 node did not include GPIO or more flexible serial interfaces such as UART or I2C. Including these would improve the flexibility of the node and allow for more standard programming options (e.g. JTAG).

The system memory (SRAM) caused both power and capacity issues when running system

tests. The leakage dominated the digital power budget and the 4kB capacity was not sufficient for storing the 12 samples of 256 Hz ECG data needed for AFib detection. Additionally, the SRAM is volatile leading to a complete loss of state in the event of node death. Looking into incorporating off-chip non-volatile memories for additional storage would alleviate this issue. Since the SRAM is typically divided into banks that can be individually power gated, looking into the ideal partitioning of the banks within each memory to minimize leakage would improve system power consumption. This would require analysis of the target applications and their instruction and data memory demands.

The lp-DSP work in Chapter 7 was completed after revision 1 of the BSN node was finished and was never taped out. While programming the BSN, we realized that completing simple arithmetic functions such as compare, shift, an two's complement was a challenge in LCU mode as there wasn't a dedicated ALU. The MSP430 had this functionality within its core, but it could not be used simultaneously. Implementing the lp-DSP to take advantage of both its high-complexity DSP operations as well as its low-level ALU operations would improve this programming bottleneck. Overall, using the lessons from the lp-DSP chapter to consolidate the digital processing blocks into a smaller, more energy efficient processor would lead to a more application flexible node.

8.3.2 FIR Filter for BSN SoC

The original architecture for this filter included in version 1 of the BSN node was implemented specifically for ECG processing within specific bands. For revision 1, the architecture was made more generic and included additional features such as channel daisy-chaining. Originally, IIR topologies were discounted due to the narrowband of data being filtered in ECG that led to a high probability of filter instability (poles close to the unit circle). Looking into architectures that can transition between FIR and IIR topologies would make the design more flexible. If IIR topologies are explored, looking into methods that can reduce the phase

non-linearity for applications where it has more of an impact (e.g. speech) would be an interesting research topic.

8.3.3 Approximate Coefficients

This design was a proof-of-concept for the idea of dynamic coefficients but there are many more areas to explore with this idea. Increasing the number of window types available in the approximation unit would make the designs more flexible. Certain windows could be more robust to error than others and determining this set would make for a more accurate approximation unit.

The hardware used for the multiplier was a Baugh-Wooley topology and the divider was synthesized directly from Verilog code (i.e. x = a / b). These topologies were not chosen based on analysis of alternative arithmetic units, and further investigation is needed to determine the best units that minimize error and power.

For the cosine/sine approximation methods, symmetric bipartite tables were not considered for the comparison as I was not aware of the method during the project. This method performs two parallel table lookups to obtain a carry-save function approximation that is either converted to a two's complement number or is Booth encoded. This method uses less memory by taking advantage of symmetry and leading zeros in one of the two bipartite tables [98].

8.3.4 Logarithmic Approximation for a VAD Pipeline

The work in creating methods for computing a low power logarithm through approximation was explored very thoroughly, but there was still much work to be completed at the system level. This work focused on the logarithm and DCT, but the triangular filter accuracy had a high impact on the system accuracy and alternative implementations were not thoroughly explored. Generating a similar speech pipeline implemented in software such as MATLAB at

the university would allow for this approximate computing research to continue and expand not only further into the frontend processing units, but also backend optimizations.

Experimenting with multiple types of VAD schemes including MFCC generation and signal-to-noise power to identify speech content would be an interesting extension of this work. Both schemes require an FFT, the highest power system component, and this block should be the focus for reducing power. Since typical speech applications require sampling rates of 16kHz and have frame sizes 32ms, the FFT must still be able to compute the result before a new frame arrives, placing a strict throughput constraint on the circuit. Other commonly used blocks such as filters, logarithms, DCTs and Euclidean distance computations can benefit from approximate computing techniques to reduce power while maintaining the system-level accuracy and throughout requirements for the system. Since there are many methods for keyword detection, a more in-depth survey of methods should be computed and modeled to determine the optimal accuracy-power tradeoff for backend processing. Vector quantization and Hidden Markov Model-based methods should be the initial approaches due to their lower complexity implementations.

8.3.5 LP-DSP Fabric

The work in Chapter 7 was some of the last work that I completed while in graduate school and has the most potential for extension and many unanswered research questions.

The choice to use a highly efficient Winograd FFT kernel opposed to a Cooley-Tukey kernel was a decision based on intuition that repeated use of multipliers tends to be high power, area, and latency. Completing the same work with 4, 8, and 16-point FFT kernels using the traditional, Cooley-Tukey algorithm for comparison would create a stronger argument for the use of the Winograd kernels.

For the memories, looking at multi-port SRAM blocks (opposed to latch files) and determining a way to limit the read/write power and maintain reliability in subthreshold would be valuable to this research effort. For large frame sizes (e.g. FFT ¿ 1024), the area

overhead due to the SCMs would be too costly, so having SRAM kernels for this purpose would make the architecture more flexible.

Considering alternative architectures for connecting the arithmetic, control, and memory units would be useful for when the size of these units gets very large. At that point, the area of the interconnect will begin to dominate and the focus would no longer need to be on the arithmetic or control hardware. Similarly, if throughput constraints are increased, this will lead to more arithmetic units and block I/Os to access the memories and exacerbate the interconnect problem. Determining if high-throughput or large sized FFTs, filters, and other DSP components are even needed in this ULP application space would be an interesting study to complete so that the impact of the interconnect is determined early in the design process.

Finally, there is a software element to this project that was not addressed and it's developing an ISA for the DSP unit and a way to run operations in parallel (with shared resources). For example, a system might need to run an FFT and an FIR on two independent sets of data stored in the memory. The global controller could allow these two operations to run simultaneously while resource sharing the arithmetic units. This would require further research into scheduling algorithms fit for hardware implementation and the development of an efficient ISA/decoder scheme for coordinating all of the DSP/ALU operations as well as managing the resource sharing. Using the decoded instructions to control power gating of APK, memory, and arithmetic units would also be a way to minimize energy by powering only what is needed.

Appendices

Appendix A

Publications

[1] **Klinefelter, A.**, Ryan, J., Tschanz, J., Calhoun B. H., *Error-Energy Analysis of Hardware Logarithmic Approximation Methods for Low Power Applications*, International Symposium on Circuits and Systems (ISCAS), May 2015.

[2] **Klinefelter, A.**, Roberts, N., et. al, A 6.45 μ W Self-Powered IoT SoC with Integrated Energy-Harvesting Power Management and ULP Asymmetric Radios, International Solid-State Circuits Conference (ISSCC), February 2015.

[3] **Klinefelter, A.**, Calhoun B. H., A Reduced Memory FIR Filter using Approximate Coefficients, Subthreshold Microelectronics Conference (S3S), October 2014.

[4] **Klinefelter, A.**, Zhang Y., Otis B., Calhoun B. H., A Programmable 34nW/channel Sub-threshold Signal Band Power Extractor on a Body Sensor Node SoC, IEEE Transactions on Circuits and Systems (TCAS) II, December 2012.

[5] Zhang Y., Zhang F., Shakhsheer Y., Silver J., **Klinefelter A.**, et al., A Batteryless 19 W MICS/ISM-Band Energy Harvesting Body Sensor Node SoC for ExG Applications, IEEE Journal of Solid-State Circuits (JSSC), December 2012.

[6] Zhang, F., Zhang, Y., Silver J., Shakhsheer Y., Nagaraju M., **Klinefelter, A.**, et al., "A Batteryless 19uW MICS/ISM-Band Energy Harvesting Body Area Sensor Node

SoC", International Solid-State Circuits Conference (ISSCC), San Francisco, February 2012.

A.1 Patents

[1] Calhoun, B.H., Shakhsheer, Y., Zhang, Y., **Klinefelter, A.**, et al. 2013. *Ultra Low Power Sensing Platform With Multimodal Radios*. U.S. Patent, Publication Number: WO 2014036451 A2, filed August 30, 2013, and issued March 6, 2014.

Appendix B

Logarithm Approximation Coefficients

For the logarithmic approximations, patterns emerged between slopes and intercepts across the range when directly approximating the function or modeling Mitchell's error. From these patterns, slope and intercept functions were derived to reduce the size of the LUT used for linear modeling of the function or error. For non-LUT methods discussed in Chapter 5, the following section describes the values used.

B.1 Direct Approximation (No Mitchell's)

PWL+MH-MULT

One case that will be used for the pipeline is two segments per power-of-two region where the Hamming weight of the number is reduced using a minimization routine. The extracted parameters for the lookup table-less implementation are shown in equations B.1, B.2, B.3, and B.4.

$$slope_{seg0} = 2^{-a} + 2^{-a-3} + 2^{-a-5} \tag{B.1}$$

$$intercept_{seg0} = a - 2 + 0.8577 \tag{B.2}$$

$$slope_{seg1} = 2^{-a-1} + 2^{-a-2} + 2^{-a-6} \tag{B.3}$$

$$intercept_{seg1} = (a-1) + 0.3537 \tag{B.4}$$

Patterns were extracted from the values based on which power-of-two value the input fell between. Those relationships are shown in the equations above and can be implemented using barrel shifters and adders. Here, a is the lower bound for the power of two region (e.g. if the input is 23, then a is 16).

PWL+SO-MULT

Another case that will be used for the pipeline is two segments per power of two region where the Hamming weight of the number is reduced using a minimization routine and the optimal x-axis partition is used. The extracted parameters for the lookup table-less implementation are shown in equations B.5, B.6, B.7, and B.8.

$$slope_{seg0} = 2^{-a} + 2^{-a-2} + 2^{-a-10} \tag{B.5}$$

$$intercept_{seg0} = a - 2 + 0.74961 \tag{B.6}$$

$$slope_{seg1} = 2^{-a-1} + 2^{-a-2} + 2^{-a-4} + 2^{-a-5} + 2^{-a-11} \tag{B.7}$$

$$intercept_{seg1} = (a-1) + 0.2488 \tag{B.8}$$

Patterns were extracted from the values based on which power-of-two value the input fell between. Those relationships are shown in the equations above and can be implemented

using barrel shifters and adders.

PWL+4S-MULT

The extracted parameters for the lookup table-less implementation are shown in equations B.9, B.10, B.11, B.12, B.13, B.14, B.15, and B.16.

$$slope_{seg0} = -\left(2^{-a-2} + 2^{-a-5} + 2^{-a-6} + 2^{-a-8} + 2^{-a-9} + 2^{-a-11}\right) \tag{B.9}$$

$$intercept_{seg0} = 0.309461863049610_{10} = 0.01001111001110_2 \tag{B.10}$$

$$slope_{seg1} = -\left(2^{-a-4} + 2^{-a-5}\right) \tag{B.11}$$

$$intercept_{seg1} = 0.050921425244932_{10} = 0.0000110100001001_2 \tag{B.12}$$

$$slope_{seg2} = 2^{-a-4} + 2^{-a-6} + 2^{-a-10} \tag{B.13}$$

$$intercept_{seg2} = -0.198691562819398_{10} = 1.111001101001000_2 \tag{B.14}$$

$$slope_{seg3} = 2^{-a-3} + 2^{-a-4} + 2^{-a-5} \tag{B.15}$$

$$intercept_{seg3} = -0.439897981923789_{10} = 1.1100011110110000_2 \tag{B.16}$$

Since the value of a changes for every input, the first term of the slope values will require a barrel shift operation. The other terms will be generated from the barrel shifted output.

B.2 Mitchell's with Compensation

M+SO-MULT

The extracted parameters for the lookup table-less implementation are shown in equations B.17, B.18, B.19, and B.20.

$$slope_{seg0} = -\left(2^{-a-3} + 2^{-a-4} + 2^{-a-8} + 2^{-a-10}\right) \tag{B.17}$$

$$intercept_{seg0} = 0.177769647051489_{10} = 0.0010110110000010_2 \tag{B.18}$$

$$slope_{seg1} = 2^{-a-3} + 2^{-a-6} + 2^{-a-7} + 2^{-a-8} + 2^{-a-9} + 2^{-a-10} \tag{B.19}$$

$$intercept_{seg1} = -0.321419395367955_{10} = 1.11010110110111000_2 \tag{B.20}$$

Since the value of a changes for every input, the first term of the slope values will require a barrel shift operation. The other terms will be generated from the barrel shifted output.

M+MH-MULT

The extracted parameters for the lookup table-less implementation are shown in equations B.21, B.22, B.23, and B.24.

$$slope_{seg0} = -\left(2^{-a-3} + 2^{-a-5} + 2^{-a-8} + 2^{-a-9}\right) \tag{B.21}$$

$$intercept_{seg0} = 0.142279004317373_{10} = 0.00100100011011000_2 \tag{B.22}$$

$$slope_{seg1} = 2^{-a-3} + 2^{-a-5} + 2^{-a-6} \tag{B.23}$$

$$intercept_{seg1} = -0.353787570264385_{10} = 1.11010010101101111_2 \tag{B.24}$$

Since the value of a changes for every input, the first term of the slope values will require a barrel shift operation. The other terms will be generated from the barrel shifted output.

M+MH+SO-MULT

The extracted parameters for the lookup table-less implementation are shown in equations B.25, B.26, B.27, and B.28.

$$slope_{seg0} = -\left(2^{-a-3} + 2^{-a-4} + 2^{-a-7}\right) \tag{B.25}$$

$$intercept_{seg0} = 0.177800_{10} = 0.0010110110000100_2 \tag{B.26}$$

$$slope_{seg1} = 2^{-a-3} + 2^{-a-5} + 2^{-a-7} \tag{B.27}$$

$$intercept_{seg1} = -0.334723_{10} = 1.11010101001010000_2 \tag{B.28}$$

Since the value of a changes for every input, the first term of the slope values will require a barrel shift operation. The other terms will be generated from the barrel shifted output.

Appendix C

Parameter Estimation from Synthesis

It's advantageous to gain information about the power, area, and delay of designs early in the design process to complete design space exploration and comparison.

The estimates given by front-end tools such as Cadence's RTL Compiler or Synopsys's DC are dependent on the accuracy of the Liberty (.lib) file and the constraints provided. This section explores methods for extracting early estimates of key design parameters using synthesis tools such as RTL Compiler and Encounter.

Area Estimation

Estimating the area of a synthesized block from the frontend estimate heavily depends on the sequential nature of the circuit. Designs with a large percentage of sequential elements (e.g. registers, latches) will have many additional buffers and routing added to reduce skew and slew in the clock tree. To give an example of the estimated area given by RTL compiler and the actual area within the P&R tool, Table C.1 shows the comparison.

These designs were synthesized with an initial density of 75% and a frequency of 500 kHz. Regardless of the % sequential cells, RTL compiler tends to drastically underestimate the area of the designs by up to 70%. Although there is some wasted area in each design due

Design	% Seq. Cell Area	RC Estimate (μm^2)	Actual (μm^2)	% Diff.
Winograd 4pt	73.8	1,460.2	2,468	59.2
Winograd 8pt	45.2	2,762.3	4,696	58.8
Winograd 16pt	53.5	4,232.2	7,171	59.0
Synth. Multiplier	0.0	2677.6	3,656	73.2
Log Divider	0.0	10,752.1	15,104	71.2
Latchfile 32r	67.3	3682.4	5,920	62.2

Table C.1: Area estimation between RTL compiler and Encounter (post-CTS and post-route)

to the final design density being <100%, the underestimate is consistently large. This is something to consider when using these early estimates from the tool.

To quickly extract this from backend tools such as Cadence's Encounter, the design area can be found by providing an estimated initial density.

```
# the initial density for the block (based on % seq. elements)
set fpDensity 0.65
# set a generic floorplan and save it to get the coordinates
floorPlan -site sc9_cln65lp -r 1 $fpDensity 0.0 0.0 0.0 0.0
# Get dimensions of this floorplan
set boundingBox [getObjFPlanBoxList Cell ${TOPMODULE}]
set bBoxWidth [lindex $boundingBox 2]
set bBoxHeight [lindex $boundingBox 3]
# redirect dimensions to a file called dim.txt
echo $bBoxWidth > dim.txt
echo $bBoxHeight >> dim.txt
exit
```

This places the dimensions of the logic bounding box (not including power rings or power ring boundary) in the text file dim.txt. These results represent that actual dimensions of the final design and are not an estimate.

Delay Estimation

Similarly to the area estimation from RTL Compiler, delay is most closely estimated after P&R when the routing and CTS has completed. The comparison between the two estimates

is shown in Table C.2 for the designs discussed in Chapter 7 for the lp-DSP. The delays extracted from RTL Compiler and Encounter are the worst negative slack reported for the entire design. From this table, it's clear that RTL compiler tends to give an optimistic estimate of the setup time slack with the exception being the multiplier and divider. For these designs, additional timing optimization steps were used to create positive timing slack in Encounter, so the slack is effectively 0 in both cases.

Design	**% Seq. Cell Area**	**RC (ns)**	**Encounter (ns)**	**% Diff.**
Winograd 4pt	73.8	4,493.3	4,109.6	8.5
Winograd 8pt	45.2	4,228.5	4,009.8	5.2
Winograd 16pt	53.5	3,945.4	3,383.9	14.2
Synth. Multiplier	0.0	0.386	14.1	~ 0
Log Divider	0.0	0.018	9.4	~ 0
Register File	67.8	3,541.8	2,403.3	32.1

Table C.2: Delay estimation between RTL compiler and Encounter (post-CTS and post-route)

It appear that regardless of the % sequential cells in the design, the overestimate of the timing slack from RTL Compiler that uses an ideal clock is not consistent or predictable based on this information alone. Since delay estimation is very accurate after Clock Tree Synthesis (CTS) in Cadence Encounter, the timing reports can be parsed to extract the worst timing slack in the design and use that as the total block delay to increase the estimate accuracy.

Power Estimation

Power estimation is the most difficult parameter to capture as it's heavily dependent on the switching behavior and duty cycle of the digital circuit. This means that both the active and leakage powers must be approximated. Estimates from RTL Compiler are rarely accurate as they do not include clock tree power or information about the activity of the design. Using Encounter to generate an Standard Delay Format (SDF) file that represents the timing

information within the design will improve the accuracy of the estimate. This SDF file is then used to generate a VCD file that represents the switching activity within the design based on an accurate design testbench, and this is used within Encounter to generate better power estimates.

The following code writes delays to a SDF file.

```
# Write the SDF file
write_sdf ${TOPMODULE}.sdf
```

When using SDF files with conditionals (keyword: COND), an error can show up that is similar to: "Failed to find matching specify module path.". This is typically due the conditional ordering between the SDF and the gate-level Verilog file provided by the standard cell vendor. The ordering between conditionals must be exact or some simulators will throw an error.

```
‘timescale 1ns/1ps
‘celldefine
module ADDF_X4 (CO, S, VDD, VSS, A, B, CI);
inout VDD, VSS;
output S, CO;
input A, B, CI;
  xor I0(sum_temp, A, B, CI);
  assign S = ((VDD === 1’b1) && (VSS === 1’b0))? sum_temp : 1’bx;
  and I1(a_and_b, A, B);
  and I2(a_and_ci, A, CI);
  and I3(b_and_ci, B, CI);
  or I4(cout_temp, a_and_b, a_and_ci, b_and_ci);
  assign CO = ((VDD === 1’b1) && (VSS === 1’b0))? cout_temp : 1’bx;

specify
if (B==1’b0 && CI==1’b1)
   (A => CO) = (‘PROP_DELAY,‘PROP_DELAY);
if (B==1’b1 && CI==1’b0)
   (A => CO) = (‘PROP_DELAY,‘PROP_DELAY);
if (A==1’b0 && CI==1’b1)
   (B => CO) = (‘PROP_DELAY,‘PROP_DELAY);
if (A==1’b1 && CI==1’b0)
   (B => CO) = (‘PROP_DELAY,‘PROP_DELAY);
//...
```

```
if (A==1'b0 && B==1'b0)
   (CI => S) = (`PROP_DELAY,`PROP_DELAY);
if (A==1'b1 && B==1'b1)
   (CI => S) = (`PROP_DELAY,`PROP_DELAY);
endspecify
endmodule // ADDF_X4
`endcelldefine
```

```
  (CELL
    (CELLTYPE "ADDF_X4")
    (INSTANCE add_16_23/g697)
      (DELAY
   (ABSOLUTE
   (CONDELSE (IOPATH CI CO (48.877::48.877) (89.017::89.017)))
   (CONDELSE (IOPATH (posedge CI) S (133.900::133.900) (89.938::89.938)))
   (CONDELSE (IOPATH (negedge CI) S (133.349::133.349) (114.859::114.859)))
   (CONDELSE (IOPATH B CO (40.195::40.195) (83.503::83.503)))
   (CONDELSE (IOPATH (posedge B) S (129.413::129.413) (119.823::119.823)))
   (CONDELSE (IOPATH (negedge B) S (126.603::126.603) (117.369::117.369)))
   (CONDELSE (IOPATH A CO (41.212::41.212) (86.543::86.543)))
   (CONDELSE (IOPATH A CO (41.212::41.212) (86.543::86.543)))
   (CONDELSE (IOPATH (posedge A) S (132.935::132.935) (111.236::111.236)))
   (CONDELSE (IOPATH (negedge A) S (130.404::130.404) (108.859::108.859)))
   (COND (~(B)&CI) (IOPATH A CO (41.211::41.211) (86.540::86.540)))
   (COND (B&~(CI)) (IOPATH A CO (38.599::38.599) (86.540::86.540)))
   (COND (~(A)&CI) (IOPATH B CO (40.194::40.194) (83.500::83.500)))
   (COND (A&~(CI)) (IOPATH B CO (40.194::40.194) (83.500::83.500)))
   ...
   (COND (~(A)&~(B)) (IOPATH CI S (133.896::133.896) (114.863::114.863)))
   (COND (A&B) (IOPATH CI S (71.850::71.850) (114.859::114.859)))
   )
      )
  )
```

Note how the ordering of the conditionals in the Verilog "specify" statement match those in the SDF file: (1) B' & CI (2) B & CI' (3) A' & CI, etc. If this error appears, one way around this is using the no_condition option that completely removes the conditionals. This will make timing less accurate as it is no longer dependent on the input values.

```
# the initial density for the block (based on % seq. elements)
write_sdf -no_condition ${TOPMODULE}.sdf
```

The resulting SDF file from Encounter can be used to simulate the structural design from P&R with accurate timing. Additional lines of code must be added the testbench to generate the VCD file used for power estimation within Encounter. A VCD file represents the switching behavior of the design and can be extracted from the behavioral simulator (e.g. Modelsim, VCS, Incisive) by including additional code within the testbench. This new simulation must occur using the structural netlist generated by P&R as instances in the SDF must exactly match the instances in the netlist.

```
// Must add timescale and celldefine statements
`timescale 1ns/1ps
`celldefine

module tb_adder_16b ();

reg[15:0] val0_i;
reg [15:0] val1_i;
wire [15:0] val_o;
reg pwrgate;
// module instantiation
adder_16b adder_16b (
   .val0_i(val0_i),
   .val1_i(val1_i),
   .val_o(val_o),
   .pwrgate(pwrgate)
);
// Stimulus
initial begin
   pwrgate = 0;
   val0_i = 0;
   val1_i = 0;
   #2000
   val0_i = 56;
   val1_i = 123;
//...
end
// Add the following statements to generate the VCD
initial begin
   $dumpfile("adder_16b.vcd");
   $dumpvars(1,adder_16b);
   $sdf_annotate("adder_16b.sdf",adder_16b,,,"MAXIMUM",,);
   // Amount of time in timescale units until simulation is completed
   #30000 $finish;
```

```
end
endmodule
`endcelldefine
```

An $sdf_annotate command incorporates the timing from the SDF file. These timing simulations are used to detect bugs such as flip-flop setup and hold violations. Once the VCD is generated by the simulator, Encounter uses it to estimate a more accurate power number based on expected activity. The VCD profile can be loaded into the design and power estimated using the following Encounter commands. The start and end times allow for power analysis within a specified window of the VCD file.

```
read_activity_file -format VCD -start 10ns -end 20ns <design name>.vcd
report_power -outfile all_power.rpt
```

The report_power command within Encounter has many useful options including -leakage, -clock_network, and -pg_domain. These allow for more specific reports on the power consumed by the clock network or individual power domains in the design.

Summary

This section provided an overview of methods for accurately extracting design metrics early in the digital design process. These flows can be further scripted to provide early, representative models of block power, area, and delay within a system.

Bibliography

[1] I. Korhonen, J. Parkka, and M. van Gils. Health monitoring in the home of the future. *Engineering in Medicine and Biology Magazine, IEEE*, 22(3):66–73, May 2003.

[2] Yanqing Zhang, Yousef Shakhsheer, Adam T. Barth, Harry C. Powell Jr., Samuel A. Ridenour, Mark A. Hanson, John Lach, and Benton H. Calhoun. Energy efficient design for body sensor nodes. *Journal of Low Power Electronics and Applications*, 1(1):109–130, 2011.

[3] Fan Zhang, Yanqing Zhang, J. Silver, Y. Shakhsheer, M. Nagaraju, A. Klinefelter, J. Pandey, J. Boley, E. Carlson, A. Shrivastava, B. Otis, and B. Calhoun. A batteryless 19 μ w mics/ism-band energy harvesting body area sensor node soc. In *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2012 IEEE International*, pages 298–300, Feb 2012.

[4] Shu-Yu Hsu, Yingchieh Ho, Po-Yao Chang, ChauChin Su, and Chen-Yi Lee. A 48.6-to-105.2μw machine learning assisted cardiac sensor soc for mobile healthcare applications. *Solid-State Circuits, IEEE Journal of*, 49(4):801–811, April 2014.

[5] Hyejung Kim, Sunyoung Kim, Nick Van Helleputte, Antonio Artes, Mario Konijnenburg, Jos Huisken, Chris Van Hoof, and Refet Firat Yazicioglu. A configurable and low-power mixed signal soc for portable ecg monitoring applications. *Biomedical Circuits and Systems, IEEE Transactions on*, 8(2):257–267, 2014.

[6] A. Klinefelter, N.E. Roberts, Y. Shakhsheer, P. Gonzalez, A. Shrivastava, A. Roy, K. Craig, M. Faisal, J. Boley, Seunghyun Oh, Yanqing Zhang, D. Akella, D.D. Wentzloff, and B.H. Calhoun. 21.3 a 6.45μw self-powered iot soc with integrated energy-harvesting power management and ulp asymmetric radios. In *Solid- State Circuits Conference - (ISSCC), 2015 IEEE International*, pages 1–3, Feb 2015.

[7] B. Murmann. Digitally assisted analog circuits. *Micro, IEEE*, 26(2):38–47, March 2006.

[8] Murugavel Raju and Mark Grazier. Energy harvesting. ulp meets energy harvesting: a game-changing combination for design engineers. *TI, http://focus. ti. com/lit/wp/slyy018/slyy018. pdf*, 2008.

[9] *A Custom Processor for Node and Power Management of a Battery-less Body Sensor Node in 130nm CMOS*, San Jose, 09/2012 2012.

[10] Yanqing Zhang, Fan Zhang, Y. Shakhsheer, J.D. Silver, A. Klinefelter, M. Nagaraju, J. Boley, J. Pandey, A. Shrivastava, E.J. Carlson, A. Wood, B.H. Calhoun, and B.P. Otis. A batteryless 19 μ w mics/ism-band energy harvesting body sensor node soc for exg applications. *Solid-State Circuits, IEEE Journal of*, 48(1):199–213, Jan 2013.

[11] Jan M. Rabaey. *Digital Integrated Circuits: A Design Perspective*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1996.

[12] Yanqing Zhang. *Synthesis Based Design Techniques for Robust, Energy Efficient Subthreshold Circuits*. PhD thesis, ECE Department, University of Virginia, Charlottesville, Dec 2013.

[13] V. Gupta, D. Mohapatra, Sang Phill Park, A. Raghunathan, and K. Roy. Impact: Imprecise adders for low-power approximate computing. In *Low Power Electronics and Design (ISLPED) 2011 International Symposium on*, pages 409–414, Aug 2011.

[14] Ernest L. Hall, D.D. Lynch, and III Dwyer, S.J. Generation of products and quotients using approximate binary logarithms for digital filtering applications. *Computers, IEEE Transactions on*, C-19(2):97–105, Feb 1970.

[15] A. Wang and A. Chandrakasan. A 180-mv subthreshold fft processor using a minimum energy design methodology. *Solid-State Circuits, IEEE Journal of*, 40(1):310–319, Jan 2005.

[16] A.P. Chandrakasan and R.W. Brodersen. Minimizing power consumption in digital cmos circuits. *Proceedings of the IEEE*, 83(4):498–523, Apr 1995.

[17] W.R. Davis, Ning Zhang, K. Camera, F. Chen, D. Markovic, N. Chan, B. Nikolic, and R.W. Brodersen. A design environment for high throughput, low power dedicated signal processing systems. In *Custom Integrated Circuits, 2001, IEEE Conference on.*, pages 545–548, 2001.

[18] Benton H. Calhoun and Anantha Chandrakasan. A 256kb 65nm sub-threshold sram design for ultra-low voltage operation. *IEEE Journal of Solid-State Circuits (JSSC)*, 42:680–688, 03/2007 2007.

[19] J. Kwong and A.P. Chandrakasan. An energy-efficient biomedical signal processing platform. In *ESSCIRC, 2010 Proceedings of the*, pages 526–529, Sept 2010.

[20] Guang-Zhong Yang. *Body Sensor Networks*. Springer, London, UK, first edition, 2006.

[21] B. Gyselinckx, C. Van Hoof, J. Ryckaert, R.F. Yazicioglu, P. Fiorini, and V. Leonov. Human++: autonomous wireless sensors for body area networks. In *Custom Integrated Circuits Conference, 2005. Proceedings of the IEEE 2005*, pages 13–19, Sept 2005.

[22] V. Leonov, T. Torfs, P. Fiorini, and C. Van Hoof. Thermoelectric converters of human warmth for self-powered wireless sensor nodes. *Sensors Journal, IEEE*, 7(5):650–657, May 2007.

[23] B.H. Calhoun, S. Khanna, Yanqing Zhang, J. Ryan, and B. Otis. System design principles combining sub-threshold circuit and architectures with energy scavenging mechanisms. In *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, pages 269–272, May 2010.

[24] Hyejung Kim, Sunyoung Kim, N. Van Helleputte, A. Artes, M. Konijnenburg, J. Huisken, C. Van Hoof, and R.F. Yazicioglu. A configurable and low-power mixed signal soc for portable ecg monitoring applications. *Biomedical Circuits and Systems, IEEE Transactions on*, 8(2):257–267, April 2014.

[25] N. Verma, A. Shoeb, J. Bohorquez, J. Dawson, J. Guttag, and A.P. Chandrakasan. A micro-power eeg acquisition soc with integrated feature extraction processor for a chronic seizure detection system. *Solid-State Circuits, IEEE Journal of*, 45(4):804–816, April 2010.

[26] G. Chen, M. Fojtik, Daeyeon Kim, D. Fick, Junsun Park, Mingoo Seok, Mao-Ter Chen, Zhiyoong Foo, D. Sylvester, and D. Blaauw. Millimeter-scale nearly perpetual sensor system with stacked battery and solar cells. In *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2010 IEEE International*, pages 288–289, Feb 2010.

[27] A.T. Barth, M.A. Hanson, H.C. Powell, and J. Lach. Tempo 3.1: A body area sensor network platform for continuous movement assessment. In *Wearable and Implantable Body Sensor Networks, 2009. BSN 2009. Sixth International Workshop on*, pages 71–76, June 2009.

[28] Liang Di, M. Putic, J. Lach, and B.H. Calhoun. Power switch characterization for fine-grained dynamic voltage scaling. In *Computer Design, 2008. ICCD 2008. IEEE International Conference on*, pages 605–611, Oct 2008.

[29] Douglas E Lake and J Randall Moorman. Accurate estimation of entropy in very short physiological time series: the problem of atrial fibrillation detection in implanted ventricular devices. *American Journal of Physiology-Heart and Circulatory Physiology*, 300(1):H319–H325, 2011.

[30] Ary L Goldberger, Luis AN Amaral, Leon Glass, Jeffrey M Hausdorff, Plamen Ch Ivanov, Roger G Mark, Joseph E Mietus, George B Moody, Chung-Kang Peng, and H Eugene Stanley. Physiobank, physiotoolkit, and physionet components of a new research resource for complex physiologic signals. *Circulation*, 101(23):e215–e220, 2000.

[31] Shailesh Rai, Jeremy Holleman, Jagdish Nayayan Pandey, Fan Zhang, and B Otis. A 500μw neural tag with 2μv rms afe and frequency-multiplying mics/ism fsk transmitter. In *Solid-State Circuits Conference-Digest of Technical Papers, 2009. ISSCC 2009. IEEE International*, pages 212–213. IEEE, 2009.

[32] Naveen Verma, Ali Shoeb, Jose Bohorquez, Joel Dawson, John Guttag, and Anantha P Chandrakasan. A micro-power eeg acquisition soc with integrated feature

extraction processor for a chronic seizure detection system. *Solid-State Circuits, IEEE Journal of*, 45(4):804–816, 2010.

[33] Long Yan, Joonsung Bae, Seulki Lee, Taehwan Roh, Kiseok Song, and Hoi-Jun Yoo. A 3.9 mw 25-electrode reconfigured sensor for wearable cardiac monitoring system. *Solid-State Circuits, IEEE Journal of*, 46(1):353–364, 2011.

[34] Nick Van Helleputte, Mario Konijnenburg, Hyejung Kim, Julia Pettine, Dong-Woo Jee, Arjan Breeschoten, Alonso Morgado, Tom Torfs, Harmke de Groot, Chris Van Hoof, et al. 18.3 a multi-parameter signal-acquisition soc for connected personal health applications. In *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2014 IEEE International*, pages 314–315. IEEE, 2014.

[35] Dongsuk Jeon, Yen-Po Chen, Yoonmyung Lee, Yejoong Kim, Zhiyoong Foo, Grant Kruger, Hakan Oral, Omer Berenfeld, Zhengya Zhang, David Blaauw, et al. 24.3 an implantable 64nw ecg-monitoring mixed-signal soc for arrhythmia diagnosis. In *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2014 IEEE International*, pages 416–417. IEEE, 2014.

[36] Olivier Girard. openmsp430, 2002.

[37] Aatmesh Shrivastava, David Wentzloff, and Benton H Calhoun. A 10mv-input boost converter with inductor peak current control and zero detection for thermoelectric energy harvesting. In *Custom Integrated Circuits Conference (CICC), 2014 IEEE Proceedings of the*, pages 1–4. IEEE, 2014.

[38] Seunghyun Oh, Nathan E Roberts, and David D Wentzloff. A 116nw multi-band wake-up receiver with 31-bit correlator and interference rejection. In *Custom Integrated Circuits Conference (CICC), 2013 IEEE*, pages 1–4. IEEE, 2013.

[39] Richard Herveille. Simple spi, 2009.

[40] A. Klinefelter, Yanqing Zhang, B. Otis, and B.H. Calhoun. A programmable 34 nw/channel sub-threshold signal band power extractor on a body sensor node soc. *Circuits and Systems II: Express Briefs, IEEE Transactions on*, 59(12):937–941, Dec 2012.

[41] G. Pfurtscheller, C. Neuper, C. Guger, W. Harkam, H. Ramoser, A. Schlogl, B. Obermaier, and M. Pregenzer. Current trends in graz brain-computer interface (bci) research. *Rehabilitation Engineering, IEEE Transactions on*, 8(2):216–219, Jun 2000.

[42] A.-T. Avestruz, W. Santa, D. Carlson, R. Jensen, Scott Stanslaski, A. Helfenstine, and T. Denison. A 5 μ w/channel spectral analysis ic for chronic bidirectional brain-machine interfaces. *Solid-State Circuits, IEEE Journal of*, 43(12):3006–3024, Dec 2008.

[43] Fan Zhang, A. Mishra, A.G. Richardson, S. Zanos, and B.P. Otis. A low-power multi-band ecog/eeg interface ic. In *Custom Integrated Circuits Conference (CICC), 2010 IEEE*, pages 1–4, Sept 2010.

[44] Karim Abdelhalim and R. Genov. 915-mhz wireless 64-channel neural recording soc with programmable mixed-signal fir filters. In *ESSCIRC (ESSCIRC), 2011 Proceedings of the*, pages 223–226, Sept 2011.

[45] N. Verma, A. Shoeb, J. Bohorquez, J. Dawson, J. Guttag, and A.P. Chandrakasan. A micro-power eeg acquisition soc with integrated feature extraction processor for a chronic seizure detection system. *Solid-State Circuits, IEEE Journal of*, 45(4):804–816, April 2010.

[46] Myeong-Eun Hwang, A. Raychowdhury, Keejong Kim, and K. Roy. A 85mv 40nw process-tolerant subthreshold 8x8 fir filter in 130nm technology. In *VLSI Circuits, 2007 IEEE Symposium on*, pages 154–155, June 2007.

[47] Wei-Hsiang Ma, J.C. Kao, V.S. Sathe, and M.C. Papaefthymiou. 187 mhz subthreshold-supply charge-recovery fir. *Solid-State Circuits, IEEE Journal of*, 45(4):793–803, April 2010.

[48] B.H. Calhoun, J.F. Ryan, S. Khanna, M. Putic, and J. Lach. Flexible circuits and architectures for ultralow power. *Proceedings of the IEEE*, 98(2):267–282, Feb 2010.

[49] Alan V Oppenheim, Ronald W Schafer, John R Buck, et al. *Discrete-time signal processing*, volume 2. Prentice-hall Englewood Cliffs, 1989.

[50] Jagdish Pandey and Brian P Otis. A sub-100 w mics/ism band transmitter based on injection-locking and frequency multiplication. *Solid-State Circuits, IEEE Journal of*, 46(5):1049–1058, 2011.

[51] Brain Computer Interface research at NUST Pakistan. Eeg motor activity data set. *https://sites.google.com/site/projectbci/*.

[52] N. Verma, J. Kwong, and A.P. Chandrakasan. Nanometer mosfet variation in minimum energy subthreshold circuits. *Electron Devices, IEEE Transactions on*, 55(1):163–174, Jan 2008.

[53] M. Khayatzadeh, Xiaoyang Zhang, Jun Tan, Wen-Sin Liew, and Yong Lian. A 0.7-v 17.4- μ w 3-lead wireless ecg soc. In *Biomedical Circuits and Systems Conference (BioCAS), 2012 IEEE*, pages 344–347, Nov 2012.

[54] J. Boley, A. Klinefelter, and B.H. Calhoun. Memory challenges and opportunities for body sensor node socs. April 2014.

[55] P. Meinerzhagen, C. Roth, and A. Burg. Towards generic low-power area-efficient standard cell based memory architectures. In *Circuits and Systems (MWSCAS), 2010 53rd IEEE International Midwest Symposium on*, pages 129–132, Aug 2010.

[56] Jean-Michel Muller. *Elementary Functions: Algorithms and Implementation*. Birkhauser Boston, Inc., Secaucus, NJ, USA, 1997.

[57] Robin Green. Faster math functions. In *Proc. of Game Developers Conference*, 2003.

[58] D. Timmermann, H. Hahn, and Bedrich J. Hosticka. Low latency time cordic algorithms. *Computers, IEEE Transactions on*, 41(8):1010–1015, Aug 1992.

[59] A. Raychowdhury, C. Tokunaga, W. Beltman, M. Deisher, J. Tschanz, and V. De. A 2.3nj/frame voice activity detector based audio front-end for context-aware system-on-chip applications in 32nm cmos. In *Custom Integrated Circuits Conference (CICC), 2012 IEEE*, pages 1–4, Sept 2012.

[60] M. Murugappan, N.Q.I. Baharuddin, and S. Jerritta. Dwt and mfcc based human emotional speech classification using lda. In *Biomedical Engineering (ICoBE), 2012 International Conference on*, pages 203–206, Feb 2012.

[61] F. Sheikh, S.K. Mathew, M.A. Anders, H. Kaul, S.K. Hsu, A. Agarwal, R.K. Krishnamurthy, and S. Borkar. A 2.05 gvertices/s 151 mw lighting accelerator for 3d graphics vertex and pixel shading in 32 nm cmos. *Solid-State Circuits, IEEE Journal of*, 48(1):128–139, Jan 2013.

[62] John N. Mitchell. Computer multiplication and division using binary logarithms. *Electronic Computers, IRE Transactions on*, EC-11(4):512–517, Aug 1962.

[63] M. Combet, H. Van Zonneveld, and L. Verbeek. Computation of the base two logarithm of binary numbers. *Electronic Computers, IEEE Transactions on*, EC-14(6):863–867, Dec 1965.

[64] S.L. SanGregory, C. Brothers, D. Gallagher, and R. Siferd. A fast, low-power logarithm approximation with cmos vlsi implementation. In *Circuits and Systems, 1999. 42nd Midwest Symposium on*, volume 1, pages 388–391 vol. 1, 1999.

[65] Tso-Bing Juang, Sheng-Hung Chen, and Huang-Jia Cheng. A lower error and rom-free logarithmic converter for digital signal processing applications. *Circuits and Systems II: Express Briefs, IEEE Transactions on*, 56(12):931–935, Dec 2009.

[66] K.H. Abed and R.E. Siferd. Cmos vlsi implementation of a low-power logarithmic converter. *Computers, IEEE Transactions on*, 52(11):1421–1433, Nov 2003.

[67] S. Paul, N. Jayakumar, and S.P. Khatri. A fast hardware approach for approximate, efficient logarithm and antilogarithm computations. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 17(2):269–277, Feb 2009.

[68] R. Gutierrez and J. Valls. Low cost hardware implementation of logarithm approximation. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 19(12):2326–2330, Dec 2011.

[69] R. Venkatesan, A. Agarwal, K. Roy, and A. Raghunathan. Macaco: Modeling and analysis of circuits for approximate computing. In *Computer-Aided Design (ICCAD), 2011 IEEE/ACM International Conference on*, pages 667–673, Nov 2011.

[70] Martin S. Schmookler and K.J. Nowka. Leading zero anticipation and detection-a comparison of methods. In *Computer Arithmetic, 2001. Proceedings. 15th IEEE Symposium on*, pages 7–12, 2001.

[71] K.H. Abed and R.E. Siferd. Vlsi implementations of low-power leading-one detector circuits. In *SoutheastCon, 2006. Proceedings of the IEEE*, pages 279–284, March 2006.

[72] John S Garofolo, Lori F Lamel, William M Fisher, Jonathon G Fiscus, and David S Pallett. Darpa timit acoustic-phonetic continous speech corpus cd-rom. nist speech disc 1-1.1. *NASA STI/Recon Technical Report N*, 93:27403, 1993.

[73] Chin-Teng Lin, Yuan-Chu Yu, and Lan-Da Van. Cost-effective triple-mode reconfigurable pipeline fft/ifft/2-d dct processor. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 16(8):1058–1071, Aug 2008.

[74] A.Y. Dogan, J. Constantin, M. Ruggiero, A. Burg, and D. Atienza. Multi-core architecture design for ultra-low-power wearable health monitoring systems. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2012*, pages 988–993, March 2012.

[75] N. Verma, A. Shoeb, J. Bohorquez, J. Dawson, J. Guttag, and A.P. Chandrakasan. A micro-power eeg acquisition soc with integrated feature extraction processor for a chronic seizure detection system. *Solid-State Circuits, IEEE Journal of*, 45(4):804–816, April 2010.

[76] Yifan He, Yu Pu, Zhenyu Ye, S.M. Londono, R. Kleihorst, A.A. Abbo, and H. Corporaal. Xetal-pro: An ultra-low energy and high throughput simd processor. In *Design Automation Conference (DAC), 2010 47th ACM/IEEE*, pages 543–548, June 2010.

[77] J. Hulzink, M. Konijnenburg, M. Ashouei, A. Breeschoten, T. Berset, J. Huisken, J. Stuyt, H. de Groot, F. Barat, J. David, and J. Van Ginderdeuren. An ultra low energy biomedical signal processing system operating at near-threshold. *Biomedical Circuits and Systems, IEEE Transactions on*, 5(6):546–554, Dec 2011.

[78] Mingoo Seok, Dongsuk Jeon, C. Chakrabati, D. Blaauw, and D. Sylvester. Extending energy-saving voltage scaling in ultra low voltage integrated circuit designs. In *IC Design Technology (ICICDT), 2012 IEEE International Conference on*, pages 1–4, May 2012.

[79] F. Philipp and M. Glesner. A reconfigurable wireless platform for biomedical signal processing. In *Biomedical Engineering International Conference (BMEiCON), 2013 6th*, pages 1–5, Oct 2013.

[80] R.A. Abdallah and N.R. Shanbhag. A 14.5 fj/cycle/k-gate, 0.33 v ecg processor in 45nm cmos using statistical error compensation. In *Custom Integrated Circuits Conference (CICC), 2012 IEEE*, pages 1–4, Sept 2012.

[81] M.A. Bin Altaf and J. Yoo. A 1.52 μj/classification patient-specific seizure classification processor using linear svm. In *Circuits and Systems (ISCAS), 2013 IEEE International Symposium on*, pages 849–852, May 2013.

[82] Shmuel Winograd. On computing the discrete fourier transform. *Mathematics of computation*, 32(141):175–199, 1978.

[83] Eleanor Chu and Alan George. *Inside the FFT black box: serial and parallel fast Fourier transform algorithms*. CRC Press, 1999.

[84] E. Jacobsen and R. Lyons. The sliding dft. *Signal Processing Magazine, IEEE*, 20(2):74–80, Mar 2003.

[85] Wen-Hsiung Chen, C. Smith, and S. Fralick. A fast computational algorithm for the discrete cosine transform. *Communications, IEEE Transactions on*, 25(9):1004–1009, Sep 1977.

[86] A.M. Shams, A. Chidanandan, W. Pan, and M.A. Bayoumi. Neda: a low-power high-performance dct architecture. *Signal Processing, IEEE Transactions on*, 54(3):955–964, March 2006.

[87] Kamisetty Ramamohan Rao and Pat Yip, editors. *The Transform and Data Compression Handbook*. CRC Press, Inc., Boca Raton, FL, USA, 2000.

[88] D.L. Jones and R.G. Baraniuk. Efficient approximation of continuous wavelet transforms. *Electronics Letters*, 27(9):748–750, April 1991.

[89] O. Rioul and P. Duhamel. Fast algorithms for discrete and continuous wavelet transforms. *Information Theory, IEEE Transactions on*, 38(2):569–586, March 1992.

[90] Liang chuan Li. A new method of wavelet transform based on fft for signal processing. In *Intelligent Systems (GCIS), 2010 Second WRI Global Congress on*, volume 3, pages 203–206, Dec 2010.

[91] Marshall C. Pease. Organization of large scale fourier processors. *J. ACM*, 16(3):474–482, July 1969.

[92] H.F. Silverman. An introduction to programming the winograd fourier transform algorithm (wfta). *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 25(2):152–165, Apr 1977.

[93] Dillon Engineering. An efficient architecture for ultra long ffts in fpgas and asics. *IEEE High Performance Extreme Computing*, 2004.

[94] P. Meinerzhagen, S.M.Y. Sherazi, A. Burg, and J.N. Rodrigues. Benchmarking of standard-cell based memories in the sub- v_{t} domain in 65-nm cmos technology. *Emerging and Selected Topics in Circuits and Systems, IEEE Journal on*, 1(2):173–182, June 2011.

[95] L. Chang, Y. Nakamura, R.K. Montoye, J. Sawada, A.K. Martin, K. Kinoshita, F.H. Gebara, K.B. Agarwal, D.J. Acharyya, W. Haensch, K. Hosokawa, and D. Jamsek. A 5.3ghz 8t-sram with operation down to 0.41v in 65nm cmos. In *VLSI Circuits, 2007 IEEE Symposium on*, pages 252–253, June 2007.

[96] Vibhu Sharma, Francky Catthoor, and Wim Dehaene. *SRAM Design for Wireless Sensor Networks: Energy Efficient and Variability Resilient Techniques*. Springer Publishing Company, Incorporated, 2012.

[97] Benton H Calhoun, Alice Wang, and Anantha Chandrakasan. Modeling and sizing for minimum energy operation in subthreshold circuits. *Solid-State Circuits, IEEE Journal of*, 40(9):1778–1786, 2005.

[98] Michael J Schulte and James E Stine. Approximating elementary functions with symmetric bipartite tables. *Computers, IEEE Transactions on*, 48(8):842–847, 1999.