Energy-Efficient Neural Network Architectures

by

Hsi-Shou Wu

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Electrical and Computer Engineering)
in The University of Michigan
2018

Doctoral Committee:

Professor Marios C. Papaefthymiou, Co-Chair
Associate Professor Zhengya Zhang, Co-Chair
Professor Trevor N. Mudge
Professor Dennis M. Sylvester
Associate Professor Thomas F. Wenisch

Hsi-Shou Wu

hsiwu@umich.edu

ORCID iD: 0000-0003-3437-7706

To my family and friends for their love and support

ACKNOWLEDGMENTS

First and foremost, my sincere gratitude goes to my co-advisors Professor Marios Papaefthymiou and Professor Zhengya Zhang for their support and guidance. Co-advising has provided me with a diversity of perspectives, allowing me to approach problems from different angles during the course of this research.

I am grateful to Marios for sharing his vision and valuable advice throughout my research journey. His willingness to share the process of conducting good research as well as developing generous collaborative relationships in the field has been instrumental to my growth as a colleague, teacher, and researcher.

I also would like to express my gratitude to Zhengya for making beneficial suggestions as well as sharing his expertise with me. Our insightful discussions enriched my research. His encouragement and support during my years as Ph.D. student are deeply appreciated.

In addition to my co-advisors, I would like to thank my thesis committee members, Professors Trevor Mudge, Thomas Wenisch, and Dennis Sylvester, for providing feedback and advice during the course of this work. Their technical comments on my research from the very beginning helped me to re-examine and reconsider the proposed approaches.

Many thanks to my colleagues and friends in the Electrical Engineering and Computer Science (EECS) Department for sharing their expertise: Tai-Chuan Ou, Yejoong Kim, Li-Xuan Chuo, Chester Liu and Wei Tang. I also wish to express my appreciation to the EECS Department staff for their generosity and assistance: Denise

DuPrie, Lisa Wonfor, Joel VanLaven, Don Winsor, Elizabeth Zaenger, and Stephen Reger.

Finally and most importantly, I want to thank my family for their love and never-ending support throughout my six years in Michigan. I dedicate my Ph.D. to them.

TABLE OF CONTENTS

LIST OF FIGURES

Figure

LIST OF TABLES

Table

ABSTRACT

Emerging systems for artificial intelligence (AI) are expected to rely on deep neural networks (DNNs) to achieve high accuracy for a broad variety of applications, including computer vision, robotics, and speech recognition. Due to the rapid growth of network size and depth, however, DNNs typically result in high computational costs and introduce considerable power and performance overheads. Dedicated chip architectures that implement DNNs with high energy efficiency are essential for adding intelligence to interactive edge devices, enabling them to complete increasingly sophisticated tasks by extending battery lie. They are also vital for improving performance in cloud servers that support demanding AI computations.

This dissertation focuses on architectures and circuit technologies for designing energy-efficient neural network accelerators. First, a deep-learning processor is presented for achieving ultra-low power operation. Using a heterogeneous architecture that includes a low-power always-on front-end and a selectively-enabled high-performance back-end, the processor dynamically adjusts computational resources at runtime to support conditional execution in neural networks and meet performance targets with increased energy efficiency. Featuring a reconfigurable datapath and a memory architecture optimized for energy efficiency, the processor supports multilevel dynamic activation of neural network segments, performing object detection tasks with 5.3× lower energy consumption in comparison with a static execution baseline. Fabricated in 40nm CMOS, the processor test-chip dissipates 0.23mW at 5.3 fps. It demonstrates energy scalability up to 28.6 TOPS/W and can be configured to run

a variety of workloads, including severely power-constrained ones such as always-on monitoring in mobile applications.

To further improve the energy efficiency of the proposed heterogeneous architecture, a new charge-recovery logic family, called zero-short-circuit current (ZSCC) logic, is proposed to decrease the power consumption of the always-on front-end. By relying on dedicated circuit topologies and a four-phase clocking scheme, ZSCC operates with significantly reduced short-circuit currents, realizing order-of-magnitude power savings at relatively low clock frequencies (in the order of a few MHz). The efficiency and applicability of ZSCC is demonstrated through an ANSI S1.11 1/3 octave filter bank chip for binaural hearing aids with two microphones per ear. Fabricated in a 65nm CMOS process, this charge-recovery chip consumes 13.8μW with a 1.75MHz clock frequency, achieving 9.7$\times$ power reduction per input in comparison with a 40nm monophonic single-input chip that represents the published state of the art. The ability of ZSCC to further increase the energy efficiency of the heterogeneous neural network architecture is demonstrated through the design and evaluation of a ZSCC-based front-end. Simulation results show 17$\times$ power reduction compared with a conventional static CMOS implementation of the same architecture.

CHAPTER I

Introduction

Modern artificial intelligence (AI) systems are transforming the world around us. AI systems have revolutionized applications such as visual/speech recognition and natural language processing by providing a convenient way to interact through an intelligent application on a mobile device. In these applications, computations can be performed purely on the edge, purely on the cloud, or through a mix of both, depending on workload. Custom application-specific integrated circuits (ASICs) have been introduced in datacenters to address the rigid response-time issue in voice search and achieve lower power than conventional general-purpose processors (CPUs) and graphics processor units (GPUs) [8]. As indicated in [9], however, low-power chip designs for AI systems on the edge provide an effective solution for running the entire system in mixed mode, reducing the queries sent to the Cloud, relieving pressure on the server side, furhter enhancing performance, and enabling more services.

Deep neural networks (DNNs) have attracted significant attention in recent years, as processing speech and image inputs require complicated computational models trained through machine learning. Compared with conventional machine-learning techniques, which are limited in processing natural data in raw data form and usually require careful engineering and domain expertise to design a dedicated feature extractor, deep-learning methods use multiple simple but non-linear modules to com-

pose transformations for complex models needed to target applications without being carefully designed by human engineers. Moreover, DNNs have been shown to provide better accuracy in numerous applications and have now become one of the core techniques in modern AI systems. However, the advantages of superior accuracy come at an increased computation cost and raise significant power and performance issues. Therefore, the efficient execution of DNNs is met with significant challenges in practice.

A variety of techniques has been investigated for increasing the efficiency of DNN execution in computing systems. Custom datapaths tailored to different types of neural network topologies have been proposed. For example, energy-efficient dataflow for convolutional neural networks (CNNs) and recurrent neural networks (RNNs) have been presented in [7] and [10], respectively. Numerical quantization is another common approach used in digital signal processing (DSP) systems and has been also applied to neural network processors [5,11]. Dedicated mixed-signal circuit design for neural networks has been proposed to achieve low-power neuromorphic computing [12]. These approaches are shown to be effective in reducing power consumption, but they all assume network execution in an ordered sequence and perform static network optimization. Therefore, they are not applicable in the context of highly-optimized neural networks whose execution is determined dynamically at runtime.

1.1 Contributions

In this dissertation, we explore two complementary design approaches for designing energy-efficient neural network processors, ranging from architecture to circuit-level innovation. We first present a new heterogeneous architecture for conditionally-executing neural networks. This architecture is composed of a dedicated, relatively slow, always-on front-end processor that continuously monitors the input, and a high-performance back-end processor that runs in duty-cycled mode to reduce power con-

sumption while maintaining accuracy and latency requirement. The dynamic execution units (DEU) embedded in the proposed design enable the neural network processor to bypass unnecessary inference paths in DNNs and therefore save power dynamically.

A prototype test-chip with the proposed heterogeneous architecture has been designed and fabricated in a 40nm CMOS process. Micro-architecture and circuit optimization techniques applied to the test-chip include low-power memory architecture, dataflow reuse, and fine-grain circuit gating to achieve an energy-efficient operation. The chip achieves 5.3× lower energy and 0.23mW average power and 1.4% accuracy loss on the LFW dataset. Using dynamic execution, it demonstrates a competitive energy scalability of 4.7-28.6 TOPS/W and can therefore support edge devices under a broad variety of operating conditions.

We further explore the opportunity to reduce power consumption of the always-on front-end processor in the heterogeneous architecture through innovative circuit topologies that achieve high energy efficiency at relatively low operating frequencies. For circuitry operating at medium to low clock frequencies, low-voltage design is a popular design approach [2, 13]. By reducing the supply voltage to the near or sub-threshold region, energy consumption can be reduced significantly. However, aggressive voltage scaling design is vulnerable to process, voltage, and temperature (PVT) variation and causes robustness issues. Moreover, voltage scaling design usually needs to insert additional buffers to increase driving strength and slew rate for reducing time conducting short-circuit currents. This becomes a more severe issue for charge-recovery logic because it relies on smooth transition during charging and discharging to increase energy efficiency. In this thesis, we investigate a new logic family called zero-short-circuit current (ZSCC) logic to address this issue and boost energy efficiency. Using a four-phase clocking scheme, ZSCC divides gate operation into fine-grain stages, providing correct timing to reset outputs signals, while significantly

reducing short-circuit power.

We have assessed the energy efficiency of ZSCC through the design of a high performance hearing-aid test-chip. Specifically, we have used ZSCC to design a binaural dual-microphone hearing-aid front-end chip that processes four audio input streams and complies with the ANSI S1.11 standard. To that end, we have designed a library consisting of ZSCC library cells in a 65nm CMOS process. We have also developed a semi-automated design flow for place-and-route with ZSSC. The chip dissipates 13.8μW at 1.75MHz and achieves 9.7$\times$ lower power at 1.75MHz compared with a 40nm monophonic chip that represents the published state of the art [14]. It also demonstrates ZSCC as a digital low-power technique competitive with analog computing [15], while at the same time offering more programmability and lower area overhead.

To assess the promise of ZSCC to increase the energy efficiency of the proposed heterogeneous neural network processor architecture, we have used it to design the front-end of that processor. In circuit-level netlist simulations, the energy consumption of the ZSCC design scales with clock frequency, reducing power down to 81.4μW at 10MHz. Compared with the static CMOS implementation of the same front-end architecture, the ZSCC-based design achieves 17$\times$ lower power, showing superior power benefits for neural network computations.

1.2 Thesis Outline

The remainder of this dissertation is organized as follows: In Chapter II, we survey and give a summary of previous work in the area of energy-efficient neural network processor design. First, we provide a brief introduction to neural networks. We then discuss energy-efficient computing techniques, including low-voltage and charge-recovery design. The chapter concludes with the description of recently-published neural network architecture optimization techniques.

In Chapter III, we present a heterogeneous neural network processor for dynamic neural networks. We first introduce the concept of conditionally-executing DNNs. We then show how our heterogeneous front-end and back-end processors can efficiently support DNNs. The processor includes a dynamic execution unit, that dispatches workload in runtime and enables the dynamic execution of the neural networks. It also includes a low-power memory architecture for further reducing data movement costs during neural network computation. The chapter concludes with chip measurement and evaluation results.

Chapter IV presents the proposed ZSCC logic family. It first describes the operation of ZSCC logic, focusing on the reduction of short-circuit current through dedicated four-phase operation and circuit topologies. It then describes the standard-cell-like semi-automated design flow for supporting large-scale digital design with ZSCC, including automatic place-and-route with a four-phase clock distribution. It also includes a description of ZSCC library cells and a comparison with the traditional static CMOS standard library cells. A prototype chip designed in ZSCC to support a high-performance binaural hearing-aid algorithm is also discussed in this chapter, including the implementation of the ZSCC datapath, silicon measurement results, and comparisons with state-of-the-art hearing-aid and acoustic-sensing chips. The chapter concludes by presenting a ZSCC-based implementation for the front-end of the heterogeneous processor presented in Chapter III, and a comparison with conventional static CMOS implementation.

Chapter V summarizes the contributions in this dissertation and presents directions for future research.

CHAPTER II

Background

In this chapter, we survey prior works in the area of neural network architectures and energy-efficient design. Specifically, Section 2.1 briefly describes the fundamentals of neural networks. Section 2.2 discusses techniques for energy-efficient digital design, with a particular focus on low-voltage and charge-recovery design. Section 2.3 presents early work on energy-efficient neural network architectures. Various optimization techniques and challenges related to neural network computations are discussed, motivating this dissertation research.

2.1 Neural Networks

Machine learning has entered the mainstream, playing a key role behind technologies such as recommendations systems on e-commerce websites, personal assistant systems, and internet search engines. These systems identify objects in images, transcribe speech into text, match new products with user interests, and select relevant search results. Behind these systems, a class of machine learning techniques called deep neural networks has emerged as a popular approach.

Neural networks attempt to mimic the intelligence of biological brains by combining a multitude of simple artificial neurons, each performing a nonlinear function of a weighted sum of its inputs. These neurons are collected into layers, and cascades

of such layers form a neural network. In so-called deep neural networks (DNNs), the number of layers ranges from tens to hundreds. Through these deep cascades, representations are transformed to a higher and more abstract level, therefore providing better model-fitting in numerous applications. Deep neural networks operate in two modes: training (or learning) and inference (or prediction). In training, the developers choose the number of layers and the type of neurons, and then determine the weights for the target application. In inference, the trained networks are used to perform predictions based on the given model.

Among various network topologies, two particular types of DNNs have brought about breakthroughs in image, video, and text and speech processing. The first type is convolutional neural networks (CNNs), whose effectiveness has been experimentally demonstrated in the area of computer vision, and which are now widely used for image and video applications. The other type is recurrent neural networks (RNNs), a powerful dynamic network architecture that processes sequential data such as speech or text. In this thesis, we focus primarily on hardware architecture design for convolutional neural networks. However, the circuit technique discussed in Chapter IV can be also applied to RNN accelerator design.

Convolutional Neural Networks

Convolutional neural networks are designed especially for processing data in the form of multiple arrays. A common application of CNNs is the processing of RGB three-channel color images containing three 2D arrays for storing pixel intensity information. CNNs have also been used to process audio spectrograms and model spectral correlations for large vocabulary speech tasks [16]. One key property of CNNs is their ability to extract local conjunctional information of input signals. CNNs use shared weights to perform neuromorphic computation through local connections of neighboring neurons. This computation is repeated through deep layers, cascaded to transform representations to higher and more abstract levels.

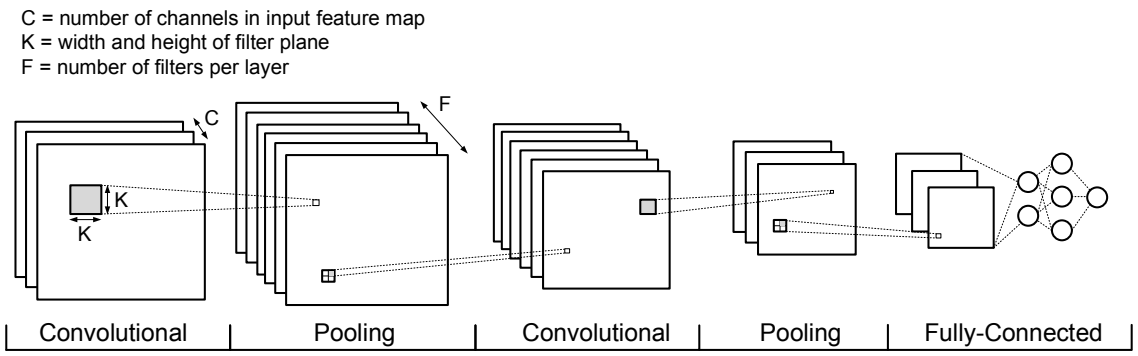


Figure 2.1: Deep convolutional neural network.

Figure 2.1 shows a typical CNN architecture structured as a cascade of stages. Most of these stages are composed of convolutional layers and pooling layers. Convolutional layers are composed of feature maps that store the collection of features generated by previous layers. Each feature in the map is generated by performing a convolution operation on local patches in the previous layer using a set of weights called a *filter bank*. After the features in the patch are combined with the weights through an inner product operation and the sums are accumulated, the result is passed to a non-linear function such as a rectified linear unit (ReLU) and stored as the output of the current convolutional layer.

Figure 2.2 shows an example of a convolution operation in a CNN layer. A three-channel 7×7×3 input feature map is convolved with the weights of two three-channel 3×3×3 filters $F0$ and $F1$. Each 7×7 channel is segmented into nine 3×3 subarrays that are formed by sweeping the 7×7 array left-to-right top-to-bottom with a stride of 2, starting from the top-left corner of the array. The convolution $Conv[i,j,k]$ of a subarray $X_{i,j}[*,*,k]$ whose top-left corner is at $X[i,j,k]$ with filter $F0[*,*,k]$ is defined as follows:

$$Conv[i,j,k] = \sum_{\substack{u=0,1,2 \\ v=0,1,2}} X_{i,j}[i+u, j+v, k] \cdot F0[u,v,k] \ , \ \text{for } k = 0,1,2 \ .$$

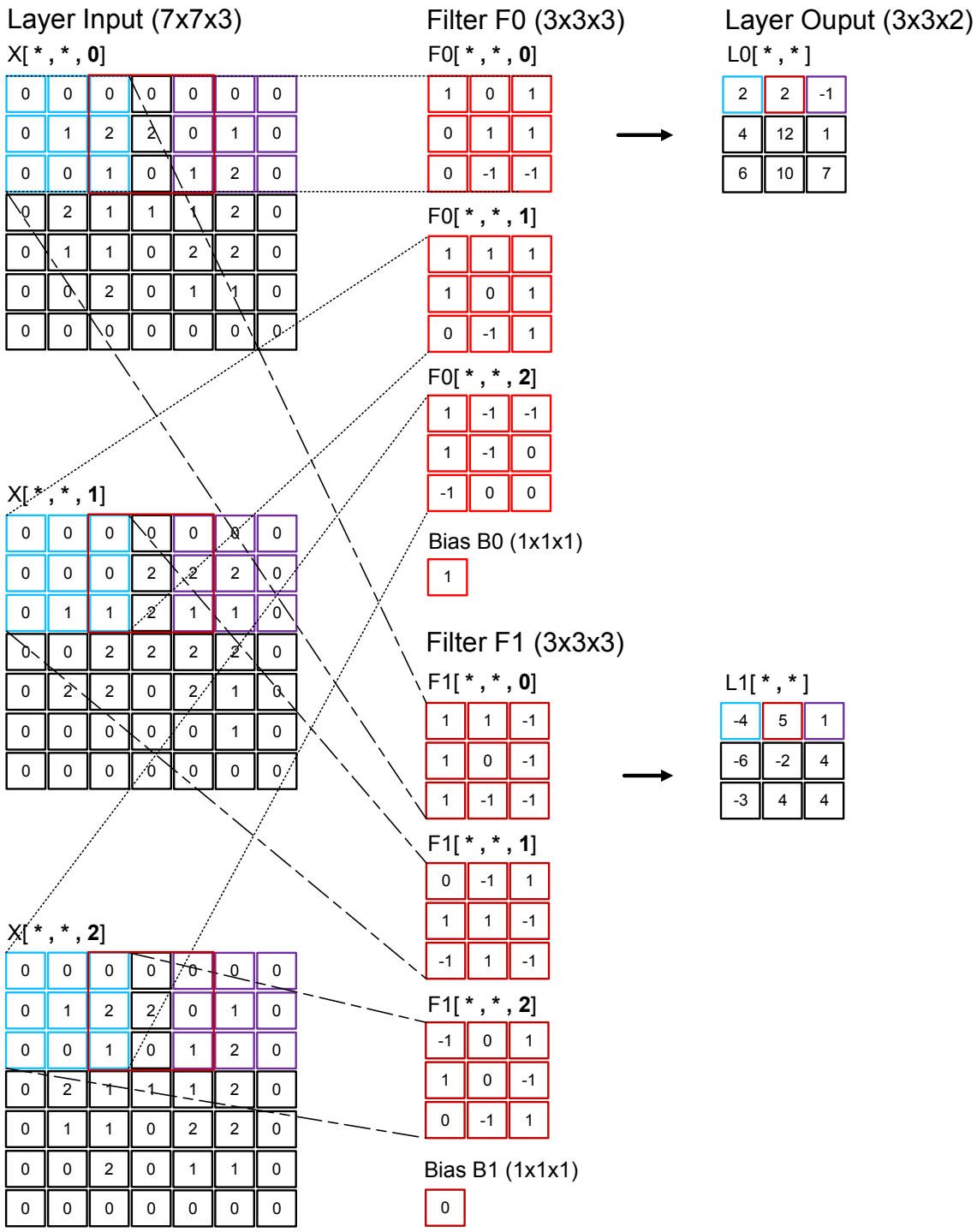


Figure 2.2: Convolution operation with a stride of two.

The element $L0[i, j]$ in the output layer corresponding to $F0$ is obtained by adding a bias $B0$ to the sum of the three convolutions $Conv[i, j, k]$, for $k = 0, 1, 2$. For example, in Figure 2.2, the 3×3 subarrays in the top-left corner $[0, 0]$ of each channel

are indicated in blue. The convolutions of these subarrays with the corresponding channels of $F0$ are added to the bias $B0 = 1$ to obtain the value 2 in the top-left corner of the layer output $L0$ corresponding to $F0$. The remaining eight elements in $L0$ are generated by repeating the above procedure for the remaining subarrays starting at $[i, j]$, where $i, j = 0, 2, 4$, and $[i, j] \neq [0, 0]$.

The convolutional layers are designed in response to the high correlation of local feature values. These correlated data can form distinctive motifs for the targeted object. Moreover, the location of the motif of interest can vary in input data. For instance, a targeted object could appear in the middle of the input image or at the corner. So, the same procedure must be applied to find the motif we search for. Shared filter weights are therefore used during convolution operation for detecting the features we search for.

Figure 2.1 also shows a pooling layer between consecutive convolutional layers. A common implementation of pooling layers is to perform down-sampling by finding the maximum value of a local patch of units in one feature map. Pooling layers merge similar features into one, thus reducing the dimensions of the representation. Therefore, some distortion or relative position variation of features would be ignored, and the same motif is easily detected. The last few layers of a CNN usually consist of fully-connected (FC) layers. The FC layers perform final classification from assembled high-level features generated by previous layers.

Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are designed for processing sequential inputs containing context relationships such as speech or language. Due to the strong sequential relationship in these data, RNNs process the current input data with an embedded "memory" that records what has been seen or heard so far. Figure 2.3 shows a typical recurrent neural network topology and the unfolding of the computation in time. When new input data is received by a RNN, the network generates

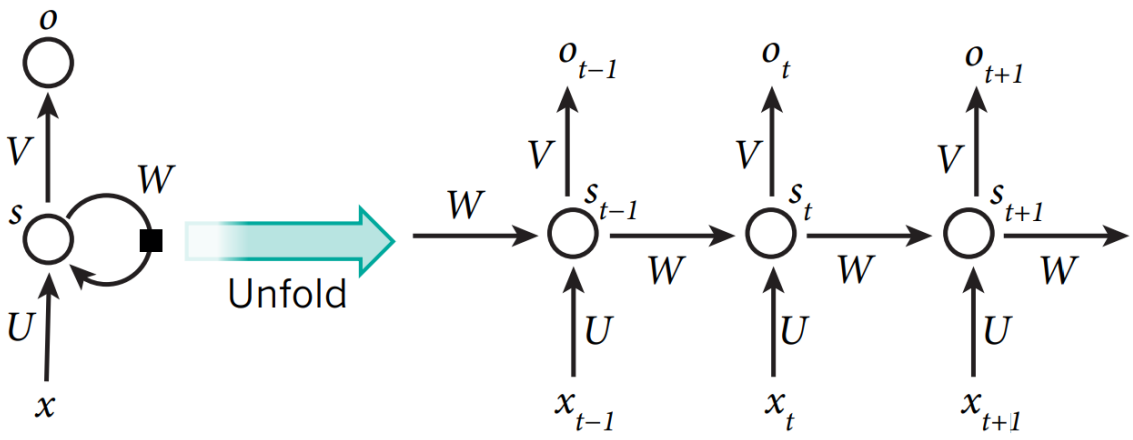


Figure 2.3: A recurrent neural network and the unfolding in time of the computation involved in its forward computation [1]. Node s comprises groups of hidden neural networks with state value s_t at time t. Matrices U, V and W are parameters of the network. Input and output are represented using symbol x and o, with the subscripts representing the time step.

corresponding activation outputs depending on its current value x_t and the hidden state value s_t. The hidden state s_t is updated based on the previous hidden state value s_{t-1} and the current input x_t to mimic the memory functionality. Matrices U,V and W are parameters used to calculate internal hidden states and output, which are adjusted through the training phase for fitting the current input task. As a result, the output s_t has a relationship between all previous x'_t for $t' \leq t$, properly modeling the sequential information inside data.

2.2 Energy-Efficient Computing

The dynamic power consumption P_{dyn} of digital circuitry is given by the following formula.

$$P_{dyn} = C_{eff} V_{DD}^2 f_{clk} \ , \tag{2.1}$$

where C_{eff} is the effective switching capacitance, V_{DD} is the supply voltage, and f_{clk} is the operating clock frequency. Energy consumption can be calculated as follows:

$$E_{dyn} = C_{eff} V_{DD}^2 \ . \tag{2.2}$$

For several decades now, designers have relied on technology scaling to build smaller devices and thus reduce effective switching capacitance. In the following two sections, we discuss two key design techniques for reducing energy consumption further.

2.2.1 Voltage Scaling

One of the most popular techniques for reducing energy (and consequently power) consumption is to reduce (scale) the supply voltage. From Equation (2.1) and Equation (2.2), we notice that dynamic power and energy consumption have a quadratic relation with supply voltage. Therefore, designing a digital system to operate with low voltage can be an effective solution for reducing its power consumption. For instance, the work in [17] presents an 8-bit processor that operates at a low voltage and achieves 11nW at $V_{DD} = 160$mV. The fast Fourier transform (FFT) processor in [2] reduces power down to 90nW when lowering supply V_{DD} to 180mV.

Figure 2.4 shows energy dissipation at different voltage supply levels for the FFT processor in [2]. As expected from Equation (2.1), energy dissipation decreases with supply voltage. However, the decrease stops at 350mV, and energy consumption starts to increase when supply voltage becomes lower. The unexpected energy increase is due to the leakage current, which is not considered in Equation (2.2). As supply voltage decreases, the device turn-on current (I_{on}) decreases, resulting in reduced maximum clock frequency and longer cycle time. As cycle time extends, it introduces considerable leakage energy, which becomes dominant when voltage is scaled down to very low values.

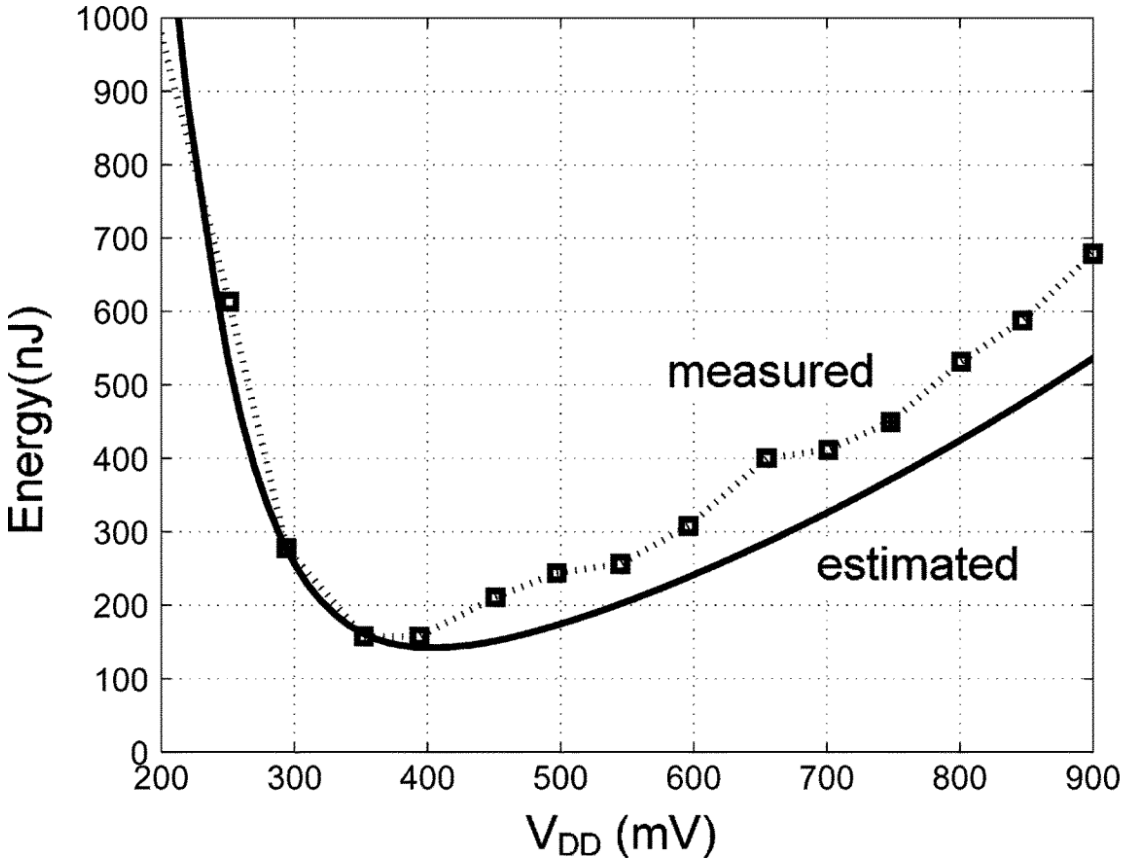


Figure 2.4: Energy dissipation as a function of V_{DD} for the 16-b 1024-pt FFT. The optimal operating point for minimal energy dissipation is at $V_{DD} = 350$mV [2].

Figure 2.5 shows the degradation of clock frequency of the FFT processor in [2] when supply V_{DD} is reduced. Voltage scaling has a significant effect on performance. The degradation becomes more severe when supply voltage is lower than the threshold voltage, and the device starts driving loads without operating in the saturation region.

2.2.1.1 Operating Region

In the case of mobile or edge devices, electronic systems are usually supplied by a limited-size battery that can only provide a fixed amount of pre-stored energy. Thus, designers build the system from the energy consumption point of view instead of that of power. A minimum-energy operating point should be pursued only after the performance requirement is met. Typically, for modern static CMOS devices, the

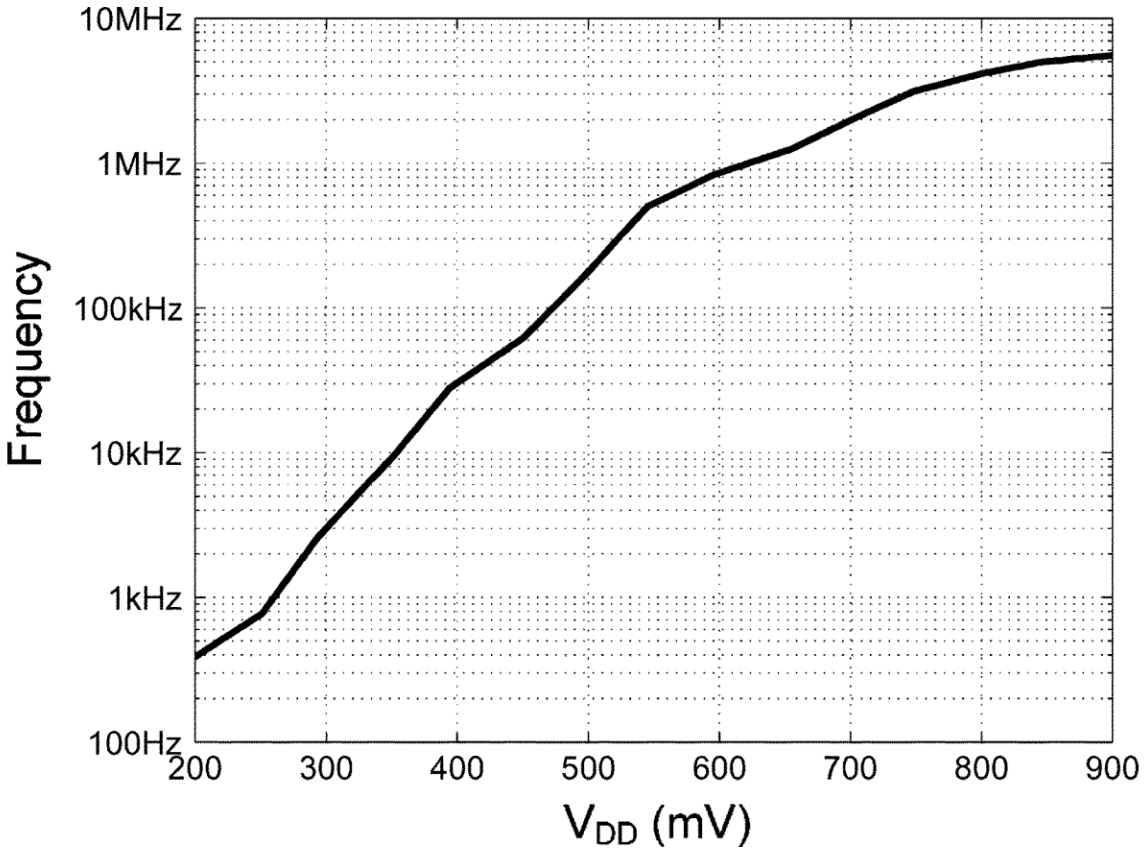


Figure 2.5: Clock frequency versus V_{DD} for a 16-b 1024-pt FFT [2].

minimum-energy point occurs at a voltage slightly lower than the device threshold voltage (V_{th}). When V_{DD} is smaller than V_{th}, the circuitry is said to operate in the subthreshold region.

Although subthreshold design can provide significant energy savings, the resulting performance degradation limits its applicability. For example, image processors in modern mobile devices demand tens to hundreds of MHz to meet performance requirements. Audio processing systems, especially speech recognizers, also need few to tens of MHz for real-time operation. One solution is to adjust the voltage to around threshold regime, called near-threshold operation. Researchers have found that when V_{DD} is in near-threshold regime, circuit delay can decrease by 50-100× while energy increases by 2×, compared with subthreshold design [18]. Therefore, near-threshold operation could be a good compromise for energy and performance

trade-offs as it sacrifices relatively small energy benefits but saves significantly on performance degradation.

2.2.1.2 Circuit Design

The performance of subthreshold and near-threshold designs is susceptible to process, voltage, and temperature (PVT) variations. In low voltage operation, MOSFET drive current has an exponential dependency on supply voltage V_{DD}, threshold voltage V_{th}, and temperature. Any small variation could deviate circuit operation from design conditions, thus resulting in additional design challenges.

Researchers have looked into the variation-introduced overhead and have proposed circuit techniques to overcome it. For example, a Pentium^{TM} class IA-32 processor design in [19] presents several design techniques to support 280mV to 1.2V operating range. The processor performs circuit optimization, including variation-aware pruning on the standard cell library, custom clocked-CMOS flip-flop circuitry, and contention-free caches using a 10-transistor bitcell. Dedicated clock distribution network incorporating programmable delay buffers in the clock paths has also been proposed for compensating skew variations in low-voltage operation.

Another circuit technique for addressing the variation issue in near-threshold design has been presented in [20]. Based on the strong sensitivity of sequential elements to voltage scaling, this work proposes a new flip-flop design that removes all dynamic nodes which are susceptible to PVT variations and features contention-free operation during signal transitions. Single-phase clocking is also used to avoid toggling of internal inverters, reducing power consumption.

2.2.1.3 Architecture

To compensate for the reduced circuit speed from voltage scaling, researchers have proposed architectural techniques to reduce performance degradation while at

the same time achieving low-power operation. Among various proposed techniques, parallelizing functional units and datapaths is one of the most popular approaches. By trading off area to add hardware units, clock requirements are reduced, and lower supply voltage can be used.

In general, parallelism is applied as follows: A central control unit broadcasts the data to be processed to multiple parallel functional units. The parallel function units receiving the data can execute tasks with lower supply voltage due to increased timing slack. The output results of the function units are collected back by the central control unit after the job assigned is finished. A common implementation of this scheme is the single-instruction multiple-data (SIMD) architecture. SIMD architecture is widely used in image/video multimedia applications which often apply the same signal processing procedure on different locations of pixel values. The architecture contains multiple processing elements (PEs) that use one single instruction to operate on multiple data at the same time. Due to its high degree of data parallelism, SIMD architecture can meet throughput targets using lower voltage, reducing energy consumption while still meeting performance requirements. A similar concept in applied in recent neural network processor designs, as discussed in Section 2.3.

Parallelism can also be applied to increase the energy efficiency of systems consisting of multiple cores. A cluster-based architecture enabling near-threshold-computing-based parallelism has been proposed in [3]. As shown in Figure 2.6, this architecture replaces the one cache per core architecture by a cluster design that contains a shared L1-cache running at a higher speed, serving a number of slow cores (k cores in the example diagram) and greatly benefiting from parallelizable workloads. After properly configuring the multicore system, the parallel architecture compensates performance loss due to near-threshold operation.

In addition to parallelism, pipelining is another technique that can be used for low-voltage design. As circuit delay increases with lower supply voltage, dividing a

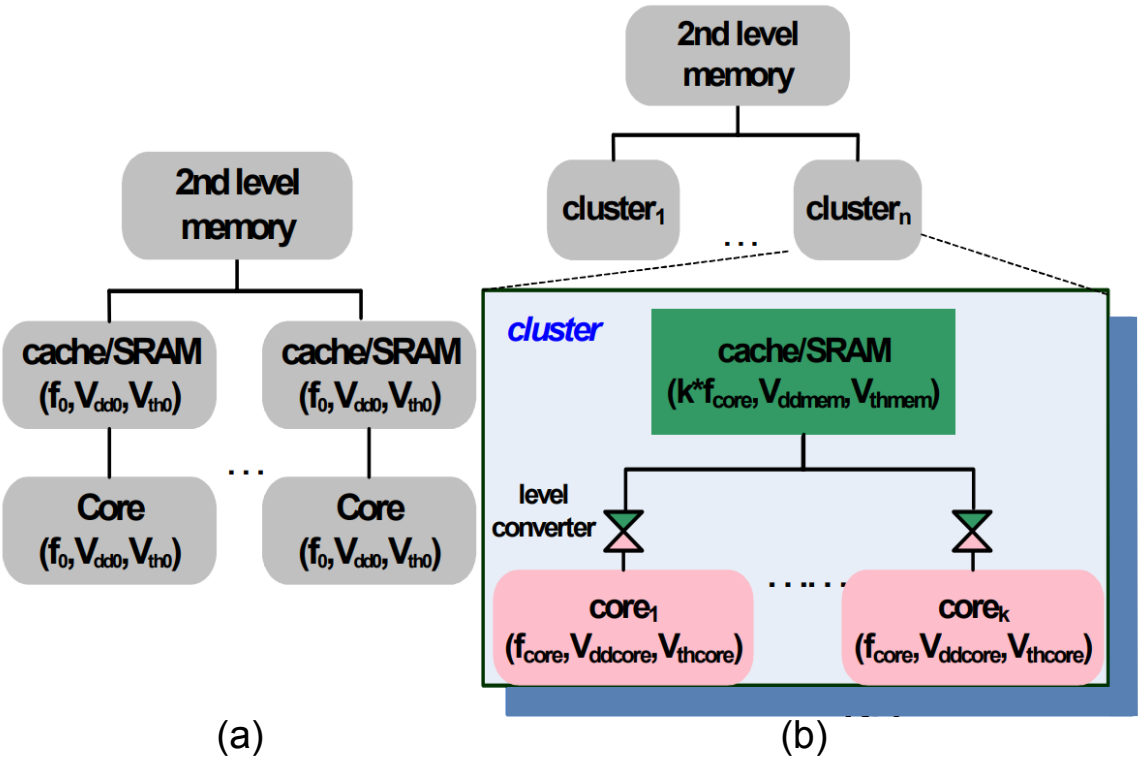


Figure 2.6: (a) Conventional multiprocessor architecture; (b) cluster-based multiprocessor architecture [3].

processing module into multiple stages can yield a pipeline design with the original stage delay, therefore operating at the original clock rate. The authors in [21] illustrate this idea using an 8-b datapath as an example. They show that after adding an additional pipeline latch between the adder and the comparator to reduce critical path delay, power consumption is reduced by 2.56× in comparison with a non-pipelined version. Aggressive pipelining can further improve energy efficiency compared with conventional ultra-low voltage pipelining by employing more stages to reduce the ratio of leakage to dynamic energy and improve performance at the same time [13]. Although pipelining can yield energy and power savings, it also increases overall latency. Careful evaluation of system aspects is therefore required before deploying this technique in target applications.

Heterogeneous architectures have the potential to achieve high performance and high energy efficiency by combining dissimilar co-processors, each running at a volt-

age and frequency optimized for maximal energy efficiency and performance. The concept of heterogeneous computing has been around for decades. In [22], authors proposed the use of two different types of processors in supercomputers to speed up applications with different characteristics, one for parallelizable tasks and another one for more sequential tasks. The work in [23] further proposed and analyzed a heterogeneous multicore design sharing a single ISA that increases energy efficiency by 2 to 3× without significant performance degradation. ARM's proposed big.LITTLE architecture [24] is another representative design, which achieves high performance and energy efficiency through dynamic voltage and frequency scaling (DVFS) on a heterogeneous architecture consisting of a high performance out-of-order processor for compute-intensive tasks and an energy-efficient in-order processor for less intensive, background tasks. As device technology advances, integrating distinct processing units into the same die to build so-called systems-on-chip (SoC) has become a popular and common approach for modern low-power mobile operation. Apple's A series and Samsung's Exynos are examples of SoC design integrating a central processing unit, a graphics processing unit, a video encoder/decoder, and an image signal processor into a single chip. Through specialized processing units combining power-efficient DVFS operation, these SoCs can achieve high energy-efficiency, while supporting a variety of applications and meeting performance requirements at the same time.

2.2.2 Charge Recovery

Charge recovery is another promising technique for energy efficient computing. A charge-recovery system divides the design into subsystems and performs energy transfer and reuse between these subsystems. In practical demonstrations of the technology, electric energy is typically converted into magnetic energy using inductive elements to recover and reuse charge in subsequent cycles.

Figure 2.7 illustrates and compares the difference between a conventional static

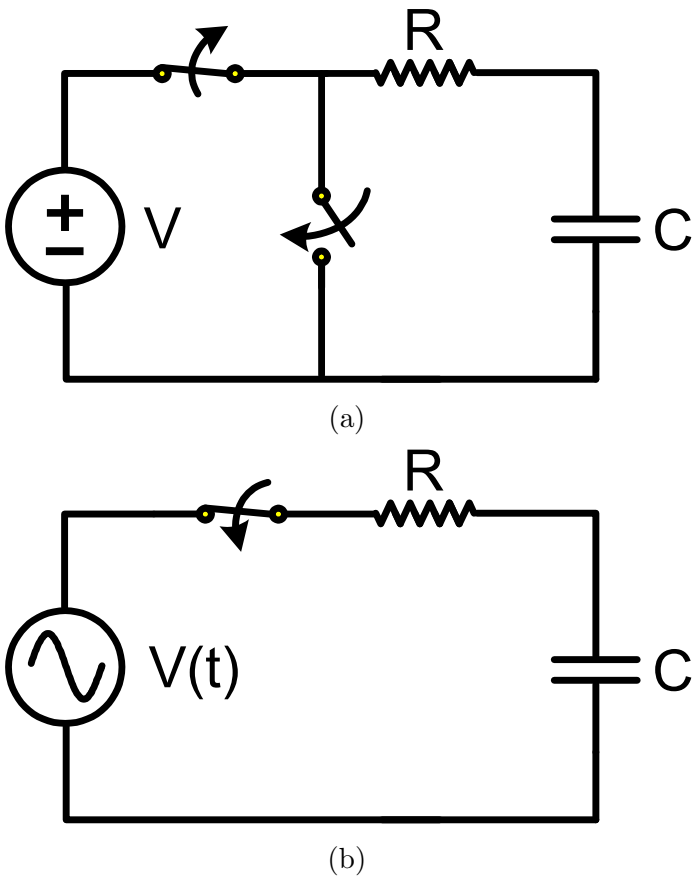
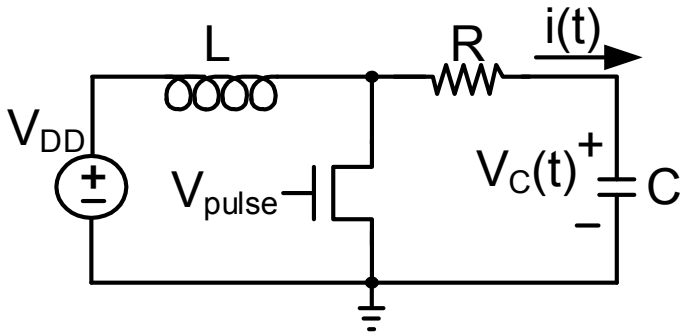


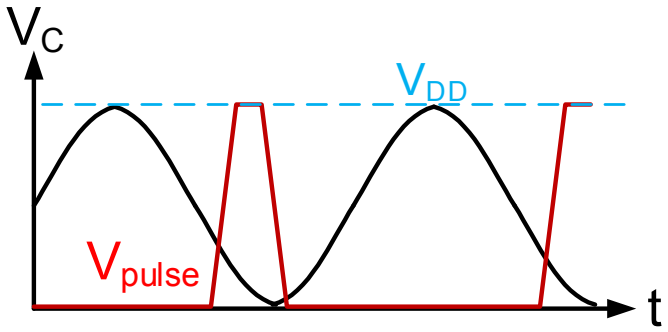
Figure 2.7: Electrical models of (a) conventional static CMOS and (b) charge-recovery switching system.

CMOS and a charge-recovery switching system. As shown in Figure 2.7(a), the conventional static MOS relies on a constant voltage supply to charge capacitive loads. To change logic state from zero to one, the switch implemented by MOS devices is turned on, resulting in current flowing from the supply to charge up the capacitive loads to V_{DD}. To toggle the logic state from one to zero, the capacitance is connected to ground through the MOS switch. The capacitance is then discharged and reset to zero voltage. The stored energy is dissipated through this charging and discharging process, and the supply source needs to replenish energy every time a state transition occurs.

A charge-recovery system is supplied by a time-varying supply source, as shown in Figure 2.7(b), and operates in a different manner compared with static CMOS. Specif-



(a) Schematic



(b) Waveform

Figure 2.8: Practical implementation of power-clock using an inductor.

ically, during operation the system recovers charge from its computing elements and transfers it to a storage medium. It then redirects charge back to the computing elements, timed as appropriate. Therefore, through this operation, a significant portion of the energy is reused. This time-varying supply source is usually referred to as a *power-clock* (*PC*). The name comes from the dual nature of this clock waveform, providing charge to internal circuit nodes, while at the same time synchronizing the computation of the gates.

One possible way for realizing the PC supply is to use an inductive element, as shown in Figure 2.8(a). An inductor L is used to resonate the capacitance and convert the energy of its electric field to the magnetic field of the inductor. This resonance results in a periodic energy transfer between the inductor L and the capacitor C, generating a sinusoidal voltage waveform across the capacitance. To sustain the oscillation, an external energy source is needed to compensate the losses introduced

by resistance R. To that end, a MOS can be inserted to the system, controlled by a pulse voltage V_{pulse}. When the V_{pulse} is high, it turns on the switch and current is being injected into the inductor for a short period of time each cycle. The self-resonance frequency F_r of this ideal LC system is given by the equation

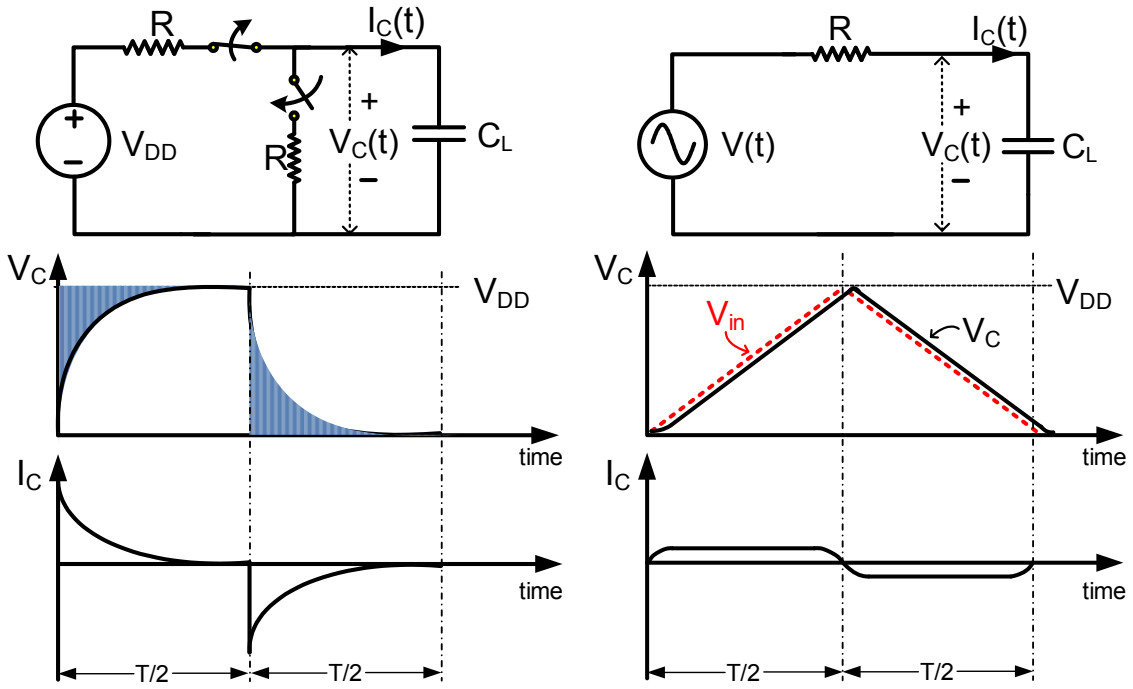
$$F_r = \frac{1}{2\pi}\sqrt{\frac{1}{LC}}\ , \tag{2.3}$$

where C is the capacitance of the system, and L is its inductance. The system achieves its highest energy efficiency when operating at F_r. By controlling the pulse input frequency of V_{pulse}, we can force the system to run at a target frequency near F_r

Figure 2.9 shows the difference of the voltage and current waveforms of a conventional static CMOS and a charge-recovery system during operation. In a conventional switching system, shown in Figure 2.9(a), at the moment its switches toggle, the full voltage value V_{DD} is seen across the charging and discharging resistive elements. This full voltage introduces a current spike that results in high energy consumption. The energy dissipated by the resistor when charging the load from time 0 to $T/2$ can be calculated as follows:

$$\begin{aligned} E_{conv} &= \int_0^{\frac{T}{2}} v_R\ i_R\ dt \\ &= \int_0^{\frac{T}{2}} (V_{DD} - v_C)\ i_c\ dt \\ &= C_L \int_0^{V_{DD}} (V_{DD} - v_c)\ dv_c \\ &= \frac{C_L V_{DD}^2}{2}\ . \end{aligned} \tag{2.4}$$

The energy dissipated during discharging can be calculated in a similar manner and



(a) (b)

Figure 2.9: (a) Conventional static CMOS switching schematic with voltage and current plots. (b) Charge-recovery switching schematic with voltage and current plots.

equals that of charging the load. Note that energy consumption is independent of charging/discharging time $T/2$. Therefore, dynamic energy consumption in a conventional static CMOS system is independent of its operating frequency.

A charge-recovery system has a different voltage and current waveform, as shown in Figure 2.9(b). Due to the gradually changing supply source, the voltage across the output load follows the supply voltage closely. Voltage drop across the resistance is therefore limited, preventing the flow of current spikes toward the source. As the supply oscillates from high to low voltage, current flows from load to source, and charge is recovered. To a first order, the current i_{CR} flowing toward the load during the charging process is

$$i_{CR} = 2\frac{C_L V_{DD}}{T} \ . \tag{2.5}$$

The energy dissipated by the resistor when charging the load between time 0 and $T/2$ can be calculated as follows:

$$\begin{aligned} E_{CR} &= \int_0^{\frac{T}{2}} I_{CR}^2 \; R \; dt \\ &= \int_0^{\frac{T}{2}} (2\frac{C_L V_{DD}}{T})^2 \; R \; dt \\ &= \frac{2RC_L}{T} C_L V_{DD}^2 \; . \end{aligned} \tag{2.6}$$

The energy consumed during discharging can be calculated in a similar manner and equals the energy consumed during charging.

From Equation (2.6) we conclude that the energy consumption of the recovery system is inversely proportional to the charging/discharging time $T/2$. Therefore, unlike conventional static CMOS design, increased energy efficiency can be achieved by operating the system with a longer clock period. Moreover, when clock period approaches infinity, dynamic energy consumption asymptotically approaches zero. Thus, by trading off energy for latency, a charge-recovery system can in principle achieve very high energy efficiency compared with a conventional static CMOS design.

Several issues arise, however, that have prevented charge-recovery design from reaching its theoretical asymptotically-zero energy consumption. The first issue is leakage currents. As operating frequency decreases and transition times become longer, leakage energy begins to increase. Therefore, total energy consumption achieves an optimal value at a certain frequency point, after which leakage energy begins to dominate, canceling the advantages of charge-recovery operation.

The second issue is short circuit currents, which can be more severe than leakage currents. Short circuit currents typically occur when devices in the circuitry are in transition. The internal nodes of the circuitry staying near the half of the nominal

voltage form a conductive path between supply and ground. This path then draws large amounts of current, increasing energy consumption. This issue is especially important when circuitry is operating in low-voltage mode with a low clock rate. As drive strengths degrade due to the lower supply voltage, circuitry remains in transition for a longer time, and short-circuit current energy becomes considerable. In traditional static CMOS design, local buffers are inserted to increase slew rate and exit transition earlier at the cost of adding energy overhead. In recovery design, this approach cannot be adopted, as slow transitions are essential for achieving energy efficient operation. In Chapter IV, we present a new logic family, called zero-short-circuit current (ZSCC) logic, to address these issues. ZSCC uses a dedicated four-phase clocking scheme and a custom circuit topology to reduce short-circuit currents, while still greatly benefiting from charge-recovery design.

2.3 Energy-Efficient Neural Network Design

The superior accuracy of DNNs is accompanied by high computational costs due to the complexity of their training models. For example, *AlexNet* [25], which in 2012 was the first CNN to win the ImageNet Challenge, requires 61 million weights and 724 million multiply-and-accumulate (MAC) operations to process a single 227×227 input image. *VGG-16* [26], which was proposed in 2015 with deeper layers, requires 138 million weights and 15.5G MACs to process a single 224×224 input image. Other topologies such as *GoogLeNet* [27] and *ResNet* [28] also need a large number of MAC operations. As these popular DNNs try to increase the depth of the network to provide better accuracy, the fundamental components of the neural layers still consist of convolution (Conv) and FC layers. These Conv and FC layers contain massive numbers of MAC operations, which can be highly parallelized.

Numerous hardware architectures have been proposed to support high degree of parallelism for optimizing throughput and energy, with SIMD architectures attract-

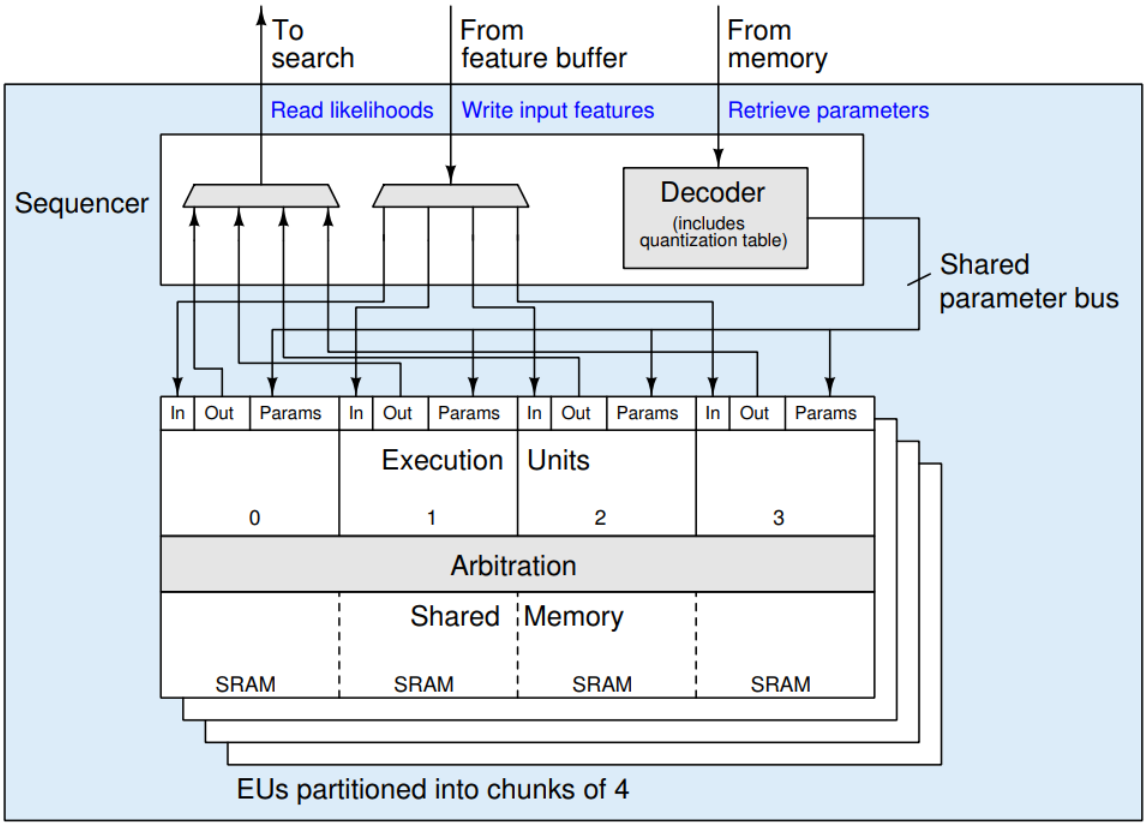


Figure 2.10: Diagram of a SIMD neural network evaluator for speech recognition [4].

ing considerable attention, as mentioned in Section 2.2.1.3. The diagram in Figure 2.10 shows the SIMD architecture of a DNN automatic speech recognizer recently presented in [4]. This recognizer consists of 32 parallel execution units (EUs) and supports 1024 nodes per layer. Each EU has a dedicated function unit that can execute MACs and different activation operations (e.g., sigmoid, rectified linear unit) for neural network computations. Local memories are built for each EU to store intermediate results and features during DNN inference. The parallelized architecture leads to reduced memory bandwidth and clock frequency requirements, and can therefore operate at a lower voltage to enhance energy efficiency.

Beyond DNN accelerators for audio applications, the SIMD architecture is particularly suitable for vision applications. The authors in [5] present an energy-efficient CNN processor that targets low-power and real-time CNN hardware for embedded

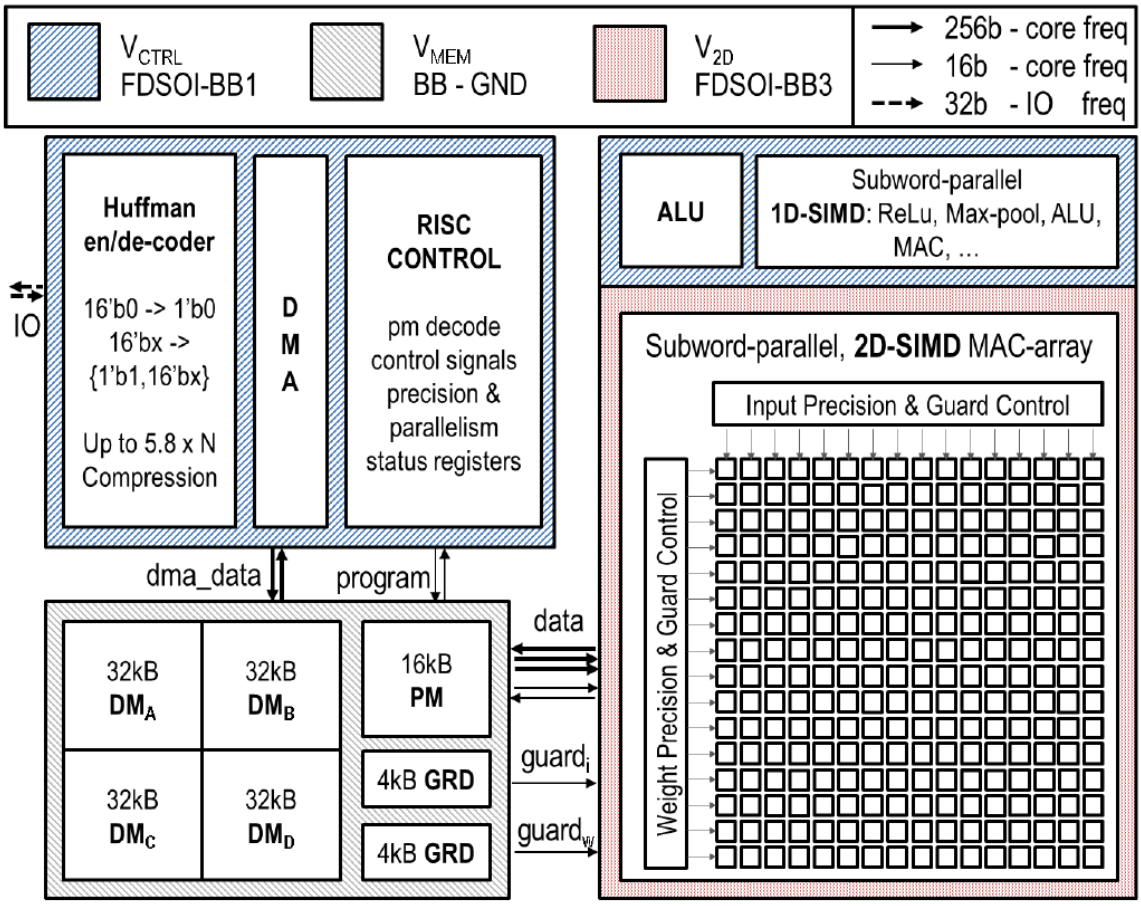


Figure 2.11: High-level architecture of the Envision processor in [5].

applications. As shown in Figure 2.11, a 16x16 2D SIMD MAC array supporting parallelism across feature and channel dimensions is built into the design. To further decrease power consumption, FIFO registers are used at the input of the MAC arrays to realize data reuse in CNN dataflow. The FIFO registers reuse overlapped features between convolutional operations and reduce local communication to the on-chip SRAM to achieve energy-efficient operation.

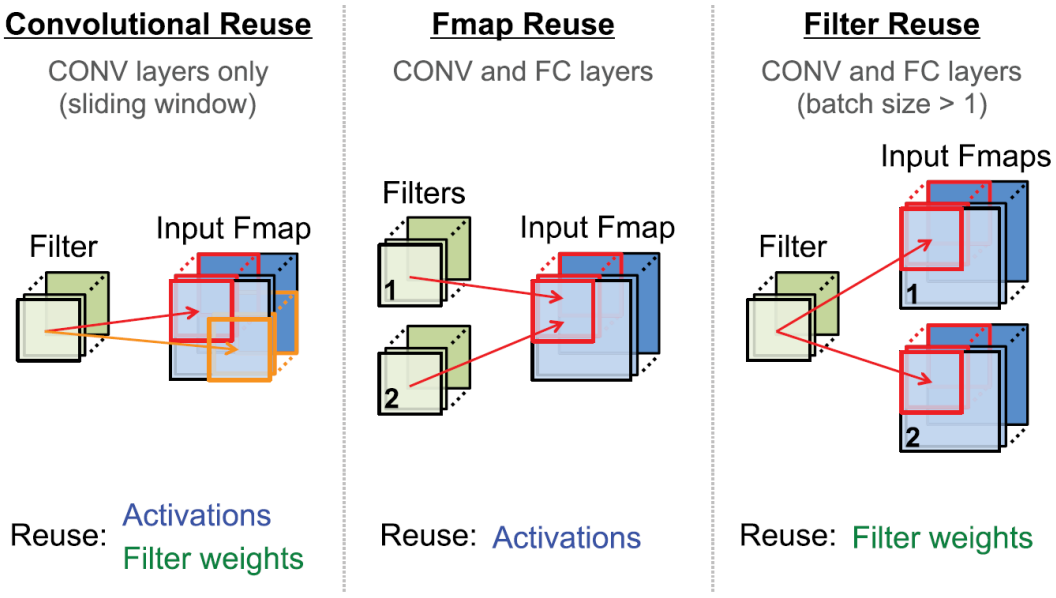
Another approach to optimize DNN computations is to use spatial architectures. These architectures are composed of multiple processing elements (PEs) that allow local communication among them. By controlling the data passing scheme to adapt to the target application, these architectures form a custom data processing chain to support energy-efficient data flow and computation. Each PE embeds local memory

and control logic. A large global buffer connected to DRAM and PE networks passes data between them and forms a memory hierarchy to save power. In the case of optimizing DNN computations, the spatial array can configure the PEs to a processing chain according to the DNN shape and size and form a specialized processing dataflow. The specialized dataflow does not only enable data parallelism but also reduces energy consumption related to data movement, which is very beneficial for neural network systems whose operation is dominated by data processing, transfer, and reuse.

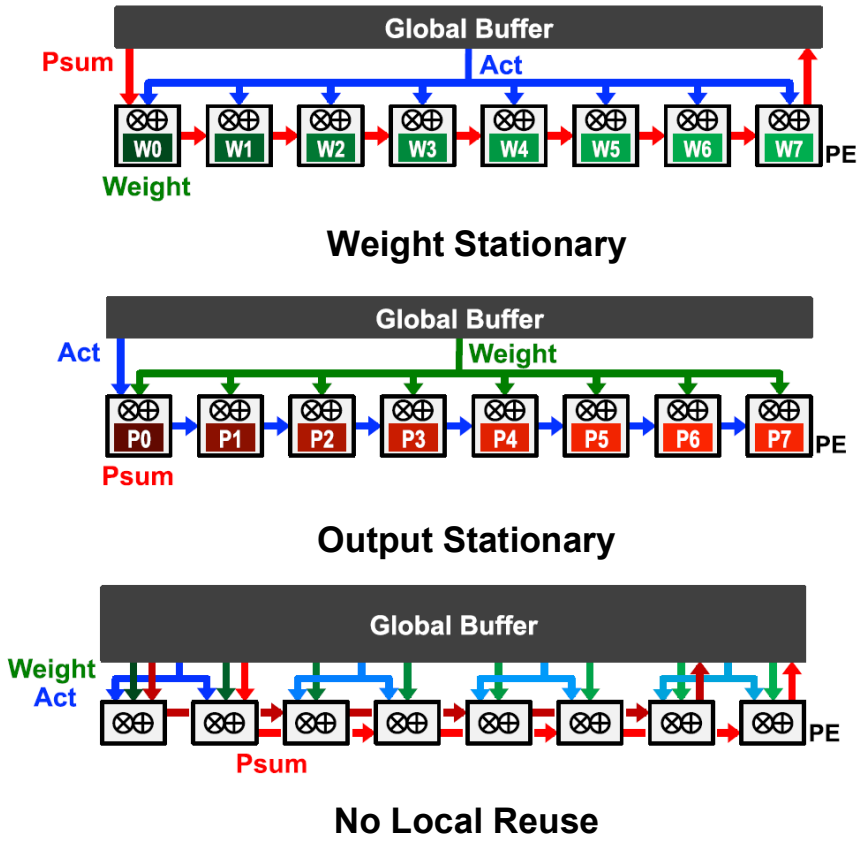
There are mainly three forms of data reuse for DNN computations: convolutional, feature map, and filter reuse, as summarized in [6]. Figure 2.12(a) shows different data reuse opportunities in these three conditions. For convolutional reuse, the overlapped input features between different convolution operations and the filter weights can be reused, as the same filter window slides on the same input feature map when performing Conv layer execution. For feature map reuse, both Conv and FC layers can have independent sets of filter weights that operate on the same input feature map, resulting in the feature map being reused multiple times before completing the updating of layer outputs. For filter reuse, the same filter weights are used to perform operations on different input feature maps which are batched at once in Conv and FC layers, leading to filter reuse opportunities.

In recent work, the spatial architecture has been exploited to optimize data reuse opportunities in DNN dataflows, which can be classified into the following taxonomy, as proposed in [6]:

1. Weight Stationary (WS): This architecture is optimized to reduce the energy cost of transferring filter weights between the local memory in the PEs and a global buffer. As shown in Figure 2.12(b), each weight read from a global buffer is stored into the local memory of PEs for initial setup. Inputs and temporary partial sums are then passed through the processing chain formed by the PEs to complete neural network computation. During this process, the filter weights



(a)



(b)

Figure 2.12: (a) Data reuse opportunities in DNNs; (b) dataflows for DNNs: Weight Stationary, Output Stationary and No Local Reuse [6].

are designed to be kept stationary, which realizes efficient architecture support for the convolutional and feature map reuse shown in Figure 2.12(a). Related research work can be found in [29–31].

2. Output Stationary (OS): This architecture is optimized to reduce the energy cost of transferring temporary partial sums during neural network computation. As shown in Figure 2.12(b), the partial sums are stored in the local memory of the PEs during the process. The central control unit instructs the global buffer to broadcast weights to the PEs and passes features down to the processing chain formed by the PEs. The partial sums remain stationary until the accumulation ends. An example of output stationary dataflow is the work presented in [32]. This architecture instructs the global buffer to send filter weights to the PE array. Input feature maps are also sent out to specific PEs at the same time. The built-in FIFO registers within each PE then pass these input feature maps to neighboring PEs horizontally and vertically to transfer feature maps within the spatial array. The temporary partial sums are kept within the PEs, and final results are transferred back to global buffer after the accumulation process is done.

3. No Local Reuse (NLR): This architecture does not contain any local memory or register file within the PEs. It is used when the design is constrained by area rather than its energy/power budget or needs to support a very large scale network. As multiple small local memories or register files are less area efficient but consume less energy, the NLR design substitutes all small storage elements for a large global buffer to maximize storage capacity. In the NLR architecture described in [33], the global buffer is instructed to send filter weights and input feature maps to the target PE units in need. The temporary partial sums and final outputs are written back to the global buffer during Conv layer execution.

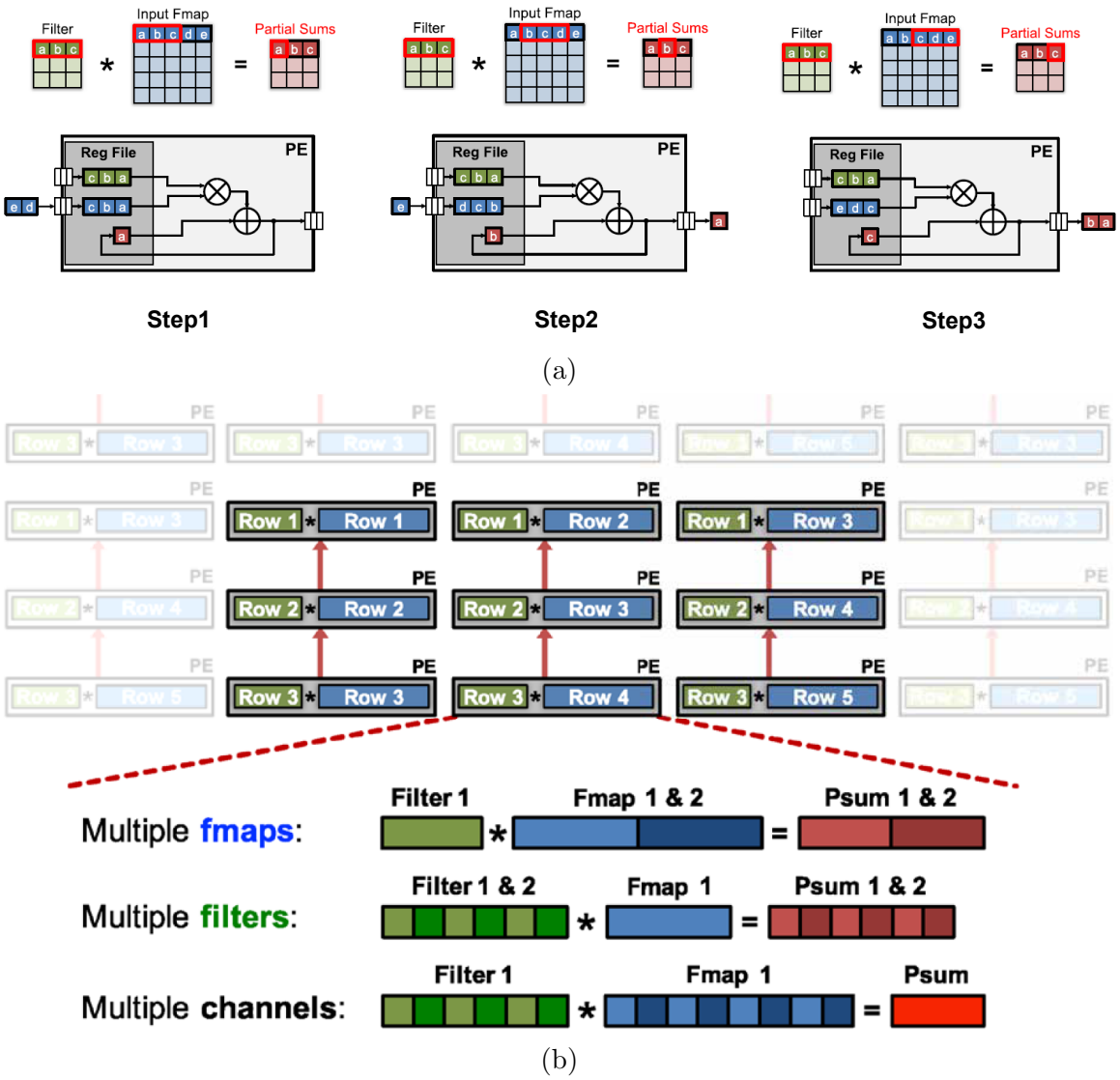


Figure 2.13: (a) One-dimensional convolutional reuse within a PE for row stationary dataflow; (b) grouping PEs into a spatial array for input feature maps, filters, and partial sums to realize local data passing and reuse [7].

Beyond the WS and OS dataflows, which optimize for filter coefficients or partial sum reuse, researchers have proposed a dataflow called Row Stationary (RS) to support the three data reuse opportunities: convolutional reuse, feature map reuse, and filter reuse at the same time [7]. Figure 2.13(a) shows how a single PE in this work process a 1-D row convolution. The input activations slide by over time while keeping the filter stationary inside the PE. The PE then performs the corresponding MAC operations and generates the partial sum for the single row in this example.

To complete a 2-D convolution, the design stacks the PEs into a spatial array and accumulates partial sums vertically to produce the row results. Different row results can be generated by different columns in the spatial array. Moreover, the authors further propose to interleave or concatenate the data sent to the PEs to compute the high-dimensional convolution of Conv layer. As shown in Figure 2.13(b), different filter coefficients are interleaved, and input feature maps are concatenated to be supplied to the same PE in a time-multiplexed manner to support multiple channels, filters, and features in convolution operation. Programmability support and energy optimization of high-dimensional convolution operation are realized at the same time.

The neural network architectures described in the previous paragraphs achieve improvements in energy efficiency, but all of them consider optimization of static neural network graphs, executing all the layers in sequential order during runtime inference. Moreover, they rely on a large number of FIFO registers in the design for local data passing and time synchronization during operation. These FIFO structures have high activity factors and unnecessary transitions, which can be further reduced through architecture and logic design to save power.

In Chapter III, we present a heterogeneous architecture supporting dynamic execution of conditional-execution neural networks, which is a type of deep neural network with dynamic graph topology. In addition to its heterogeneous organization, the proposed architecture relies on a dedicated low-power memory architecture with custom register files to further reduce energy consumption.

CHAPTER III

Heterogeneous Architecture for Conditional Neural Networks

As discussed in the previous two chapters, DNNs are now used in numerous applications, showing improved accuracy over traditional approaches. This increased accuracy comes at the cost of high computational loads, however, with associated power and response time overheads. Such overheads become a critical issue, as AI deployment migrates from data centers to energy-constrained embedded and mobile platforms. The exploration of energy-efficient DNN hardware accelerators that can support always-on operation while meeting stringent performance and power constraints has therefore become a particularly active research area.

Recently, researchers have proposed a new class of neural networks that can be dynamically adjusted during inference to adapt to input data. Specifically, [34] introduces a conditional classifier on the feedforward path to obtain classification early without completing execution of the entire network. The work in [35] proposes a methodology for augmenting neural networks using control edges to determine inference paths dynamically. In principle, such conditional execution can be effective in increasing power efficiency of DNN computations above and beyond what can be achieved by other techniques such as quantization and pruning. In practice, however, power savings may be difficult to realize, or they may come at unacceptable perfor-

mance overheads. Specifically, conditional branches introduce additional latency to the computation. Moreover, conditional classifiers may yield irregular memory access patterns, resulting in energy-inefficient memory operation. Processor architectures for conditionally-executing DNNs have thus remained elusive.

In this chapter, we present a heterogeneous processor that relies on an energy-efficient front-end and a high-performance back-end to support conditionally-executing DNNs with high performance and low power consumption through voltage and frequency scaling. A key feature of this processor is a *dynamic execution unit* that can be reconfigured dynamically at runtime to optimize performance and power efficiency. A real-time controller dispatches jobs to the execution unit during runtime reconfiguration, reducing conditional branch overheads. The data memory architecture supports burst-mode access and fine-grain gating to maximize energy efficiency under irregular access patterns. Fabricated in a 40nm CMOS process, a test-chip implementation of the proposed architecture achieves 5.3× higher energy efficiency than a baseline static execution, with an average power of 0.23mW at 5.3fps for the LFW face recognition data set and maximum achievable efficiency of 28.6 TOPS/W. Parts of the work covered in this chapter appear in ESSCIRC 2018 [36].

The remainder of this chapter is organized as follows: Section 3.1 gives background on conditional deep neural networks. Section 3.2 describes the proposed heterogeneous architecture and corresponding hardware mapping for conditional neural networks. Section 3.3 shows the system architecture. Section 3.4 presents the dynamic execution unit, a key feature of our design. Section 3.5 describes our low-power memory architecture. Section 3.6 describes the implementation of the test-chip, and Section 3.7 gives results from its experimental evaluation. Section 3.8 summarizes this chapter.

3.1 Conditional Deep Neural Networks

A variety of techniques has been proposed for reducing power consumption in DNN processors. One such relatively simple and effective approach is numerical quantization [37]. Early work in this area has focused on configurable hardware that supports variable bit precision to trade off accuracy for lower power consumption [5] [11]. BinaryNet further quantizes weights and activation to +1 or -1, simplifying multiplication to XNOR and leveraging mixed-signal circuits to yield highly efficient designs [12]. The work described in [38] leverages the sparsity of neural networks obtained by network pruning without sacrificing classification accuracy. Zero-valued weights and activations are skipped, reducing workload and data bandwidth to increase energy efficiency.

Another research direction for improving computational efficiency by achieving input-dependent execution during the DNN inference stage is inspired from the fact that not all input needs the same processing procedure. For example, in an autonomous car system, when a road sign is captured by the camera, the system should intelligently trigger the traffic sign detection units for processing. Only when a car appears in front of the camera should the vehicle detection units be awakened. That is, the system adapts to input conditions and reduces the processing time and energy consumption by activating only the corresponding portion of the neural network.

Earlier work related to conditional neural network computations can be found in [34, 39–42]. For example, the work in [39] discussed conditional operation opportunities for FC networks. A learning policy for FC layers is proposed for training the network to activate only the corresponding nodes related to current input. The authors in [41] proposed a type of neural network that could be used for language modeling. It is composed of network layers consisting of a number of feed-forward networks as expert units and gating networks. The gating networks can be trained to select specific expert units at runtime to perform necessary computations for a given

input.

Another approach recently proposed in [35] explores the notion of conditional computation for reducing computation costs and processing time. Unlike the work presented in [39, 41], the authors in [35] propose a methodology to train the control units used for activating specific sub-networks with flexible network topology. The locations of the controllers are also unconstrained, unlike in prior work where particular locations are required to insert them. Moreover, compared with prior works which require separate learning procedures for control units, the work in [35] proposes an end-to-end training method and an ability to optimize accuracy-efficiency trade-offs, showing that the conditional DNNs are applicable to a broad variety of situations.

A high-level view of conditional DNNs is shown in Figure 3.1. Conditional DNNs are directed acyclic graphs (DAGs) consisting of two type of networks, compute nets and decision nets. The compute nets are conventional neural networks performing regular data processing, such as convolution operation, pooling, or FC layer computation. The decision nets are control units that are augmented between the compute net stages to adjust the operation of the original DNN. When a decision net executes, it extracts additional features and calculates scores to determine which compute net is to be activated in the next stage. For example, as shown on the left hand side of Figure 3.1, the decision network controls Compute Network 2 and Compute Network 3. One of the networks will be awakened and will execute the task after the decision network selects the path with the higher score.

The concept can be extended to a more complex topology such as the cascaded or hierarchical topology shown on the right hand side network of Figure 3.1. Multiple stages of decision nets are inserted between deep compute networks for complex multi-object recognition tasks. These decision nets help with the inference task by examining the early features generated in the middle stages, and determining whether to branch to a small compute subnet or terminate in advance.

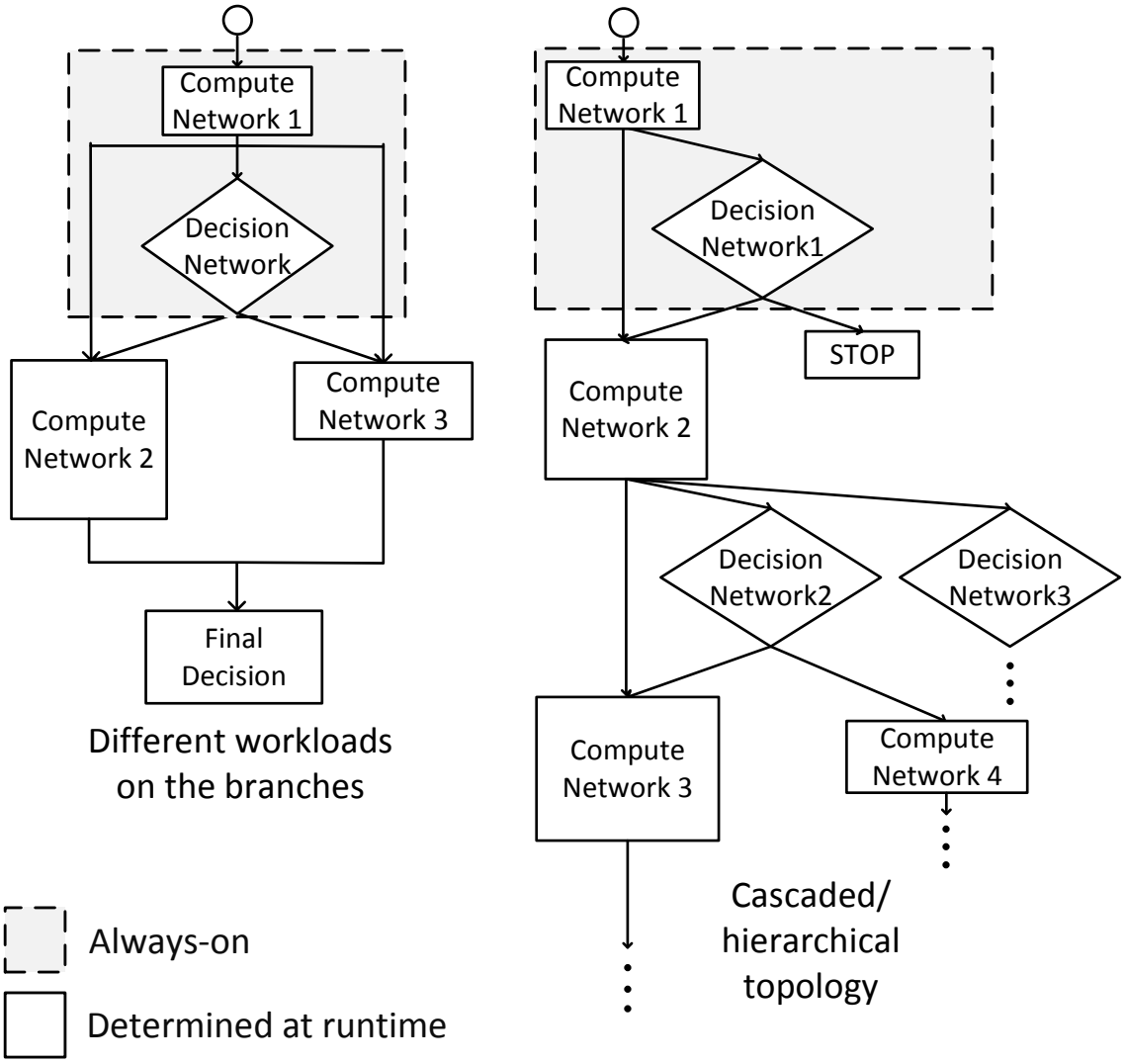


Figure 3.1: Conditional deep neural network.

3.2 Heterogeneous Architecture and Hardware Mapping

The main characteristic of conditional DNNs is that initially a few head compute nets and decision nets perform coarse classification, pruning unnecessary paths in the early stages. The remaining layers then perform detailed fine-grain feature extraction for final classification. The head compute nets and decision nets determine the high-level runtime pattern of the DNN. Although they are relatively light-weight in terms of complexity, they must be implemented with high efficiency, as they are always active.

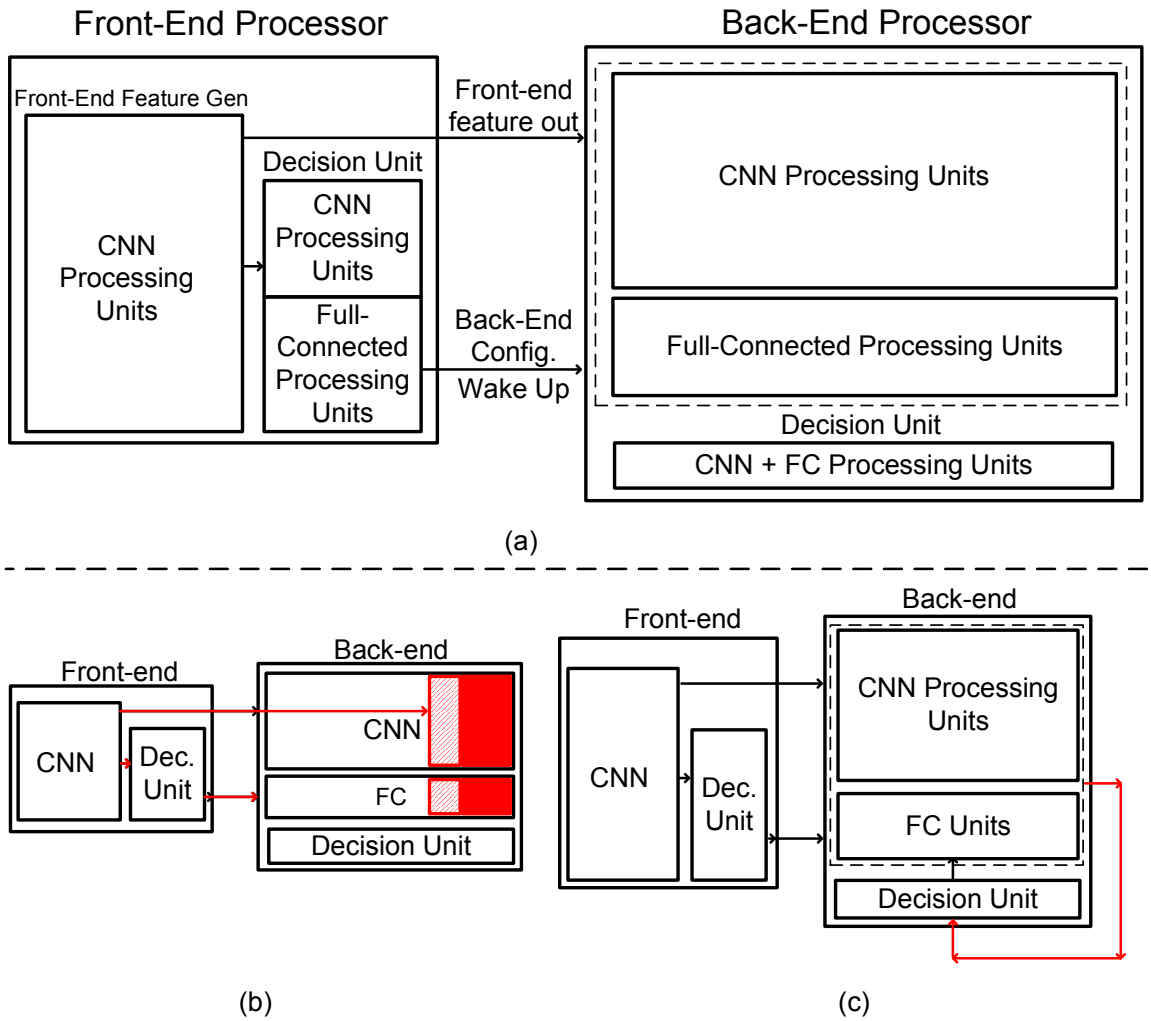


Figure 3.2: (a) Heterogeneous architecture for efficient conditional neural network hardware mapping. (b) Dynamic workload adjustment during runtime. (c) Hierarchical/cascaded inference for complex tasks. Embedded decision unit in back-end controls the inference path after wakeup.

This characteristic makes the conditional DNN suitable to be mapped to a heterogeneous architecture as shown in Figure 3.2. A heterogeneous architecture composed of a relatively low-performance energy-efficient processor and a high-speed power-demanding processor is shown in Figure 3.2(a). The head compute and decision nets, both of which consist of Conv or FC layers, are assigned to a low-power front-end. A high-performance back-end is used for processing fine-grain computationally demanding tasks. By running the front-end at low voltage and heavily duty-cycling the operation of the back-end, the architecture is capable of operating with high energy

efficiency.

We designed the front-end unit as an always-on subsystem, continuously monitoring the processors input and automatically adjusting the workload of the back-end at runtime. For instance, in an object recognition scenario, if the targeted object is absent from view, the front-end executes light workloads and then idles. When an object of interest appears, it wakes up the back-end to perform the more computationally intensive tasks for correct recognition, as shown in Figure 3.2(b). Moreover, after the back-end is awakened, it can execute fine-grain sub-tree inferences for the cascaded or hierarchical network topology shown in Figure 3.1 on the right hand side. By using the proposed datapath in the back-end processor as shown in Figure 3.2(c), the architecture can support a variety of object searching with minimal energy cost.

3.3 System Architecture

Figure 3.3 shows the overall system design for supporting conditional DNNs. The conditional DNNs are first trained through an off-line software framework. After network topologies and parameters are fixed at the end of the training phase, a mapper compiles the DNN structures to proper binary codes given the architecture implementation specification. The generated codes are sent to an on-chip Host-CPU to configure and control the operation of the front-end and back-end processors for processing input data.

Implementation specifics of the proposed heterogeneous processor are shown in Figure 3.4. The front-end always-on processor contains its own memories and input image buffers for storing data locally. The total size of the static memory used for storing/framing continuous input data, weights, and temporary features is 34KB. The front-end core is embedded with a 3×8 array of single-instruction multiple-data (SIMD) multiply-accumulate (MAC) units for RGB raw image input processing and Conv and FC computations. Dedicated ReLU and maxpooling functional units are

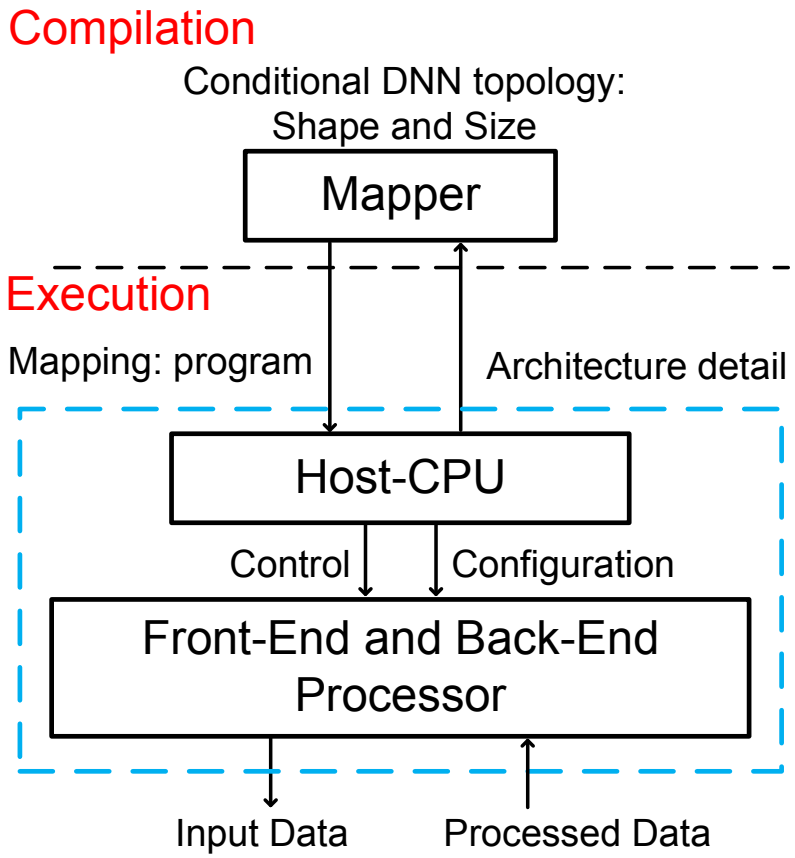


Figure 3.3: System overview for conditional DNN.

also included in the front-end core for thresholding and pooling in DNN operation. A direct memory interface between front-end and back-end is designed for moving the processed data directly during runtime inference.

Figure 3.4 also shows the high-performance back-end processor design for accelerating compute nets and decision nets. Instead of making dedicated compute net accelerators and decision net accelerators, which would result in hardware underutilization due to the dynamic nature of the conditional DNNs, we design the back-end using Conv cores and FC cores that can be configured to implement compute nets, decision nets, or a mix of both. These cores comprise a finer-grain, dynamic execution unit (DEU) that is designed to ensure full hardware utilization while providing the execution flexibility for gaining performance or reducing power by taking advantage of timing slack. The back-end includes two Conv cores, each with a 16×16 SIMD

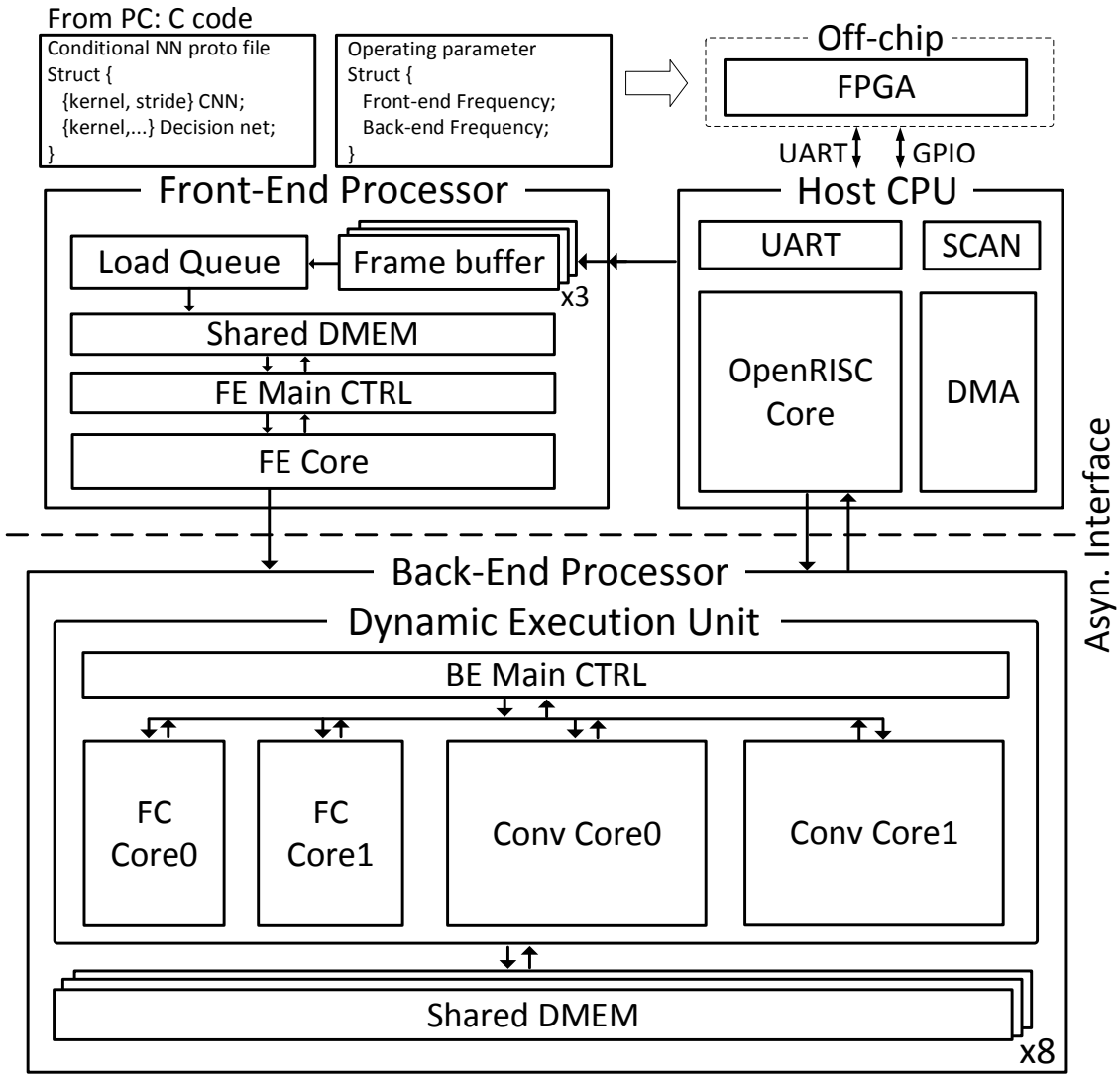


Figure 3.4: Top-level processor architecture.

MAC array, and two FC cores, each with a 4×13 SIMD MAC array. A 240KB static memory is used to store features from the front-end, convolutional kernels, FC layer weights, and temporary features results. Once the back-end wakes up, it executes the fine-grain subtree inference shown in Figure 3.1. The configurable dynamic execution units are therefore designed to efficiently execute compute nets and decision nets at the same time, supporting a variety of object search modes with minimal energy cost.

To meet application latency requirements (1s in surveillance systems, 30ms for real-time applications), the front-end and the back-end run at or above 10MHz and

150 MHz, respectively. An OpenRISC processor serves as the host CPU to control the operation of the two cores during runtime, sending compiled network configurations and instructions to the two cores and receiving data from them. It also controls the direct memory access (DMA) for on-chip data movement. A field-programmable gate array (FPGA) serves as an external host processor to program the OpenRISC processor through a UART interface and move image data from an off-chip DRAM onto the chip.

3.4 Dynamic Execution Unit

A key feature of the proposed processor is the DEU in the back-end, shown in Figure 3.5. The DEU consists of a main controller, two Conv cores, and two FC cores. Each core has its own small sub-controller which receives instructions from the main controller. Dedicated operation codes are designed for these sub-controllers to support the unique low-power dataflow, which is presented in Section 3.5, including data movement, flow control, and operation triggering. The Conv core has post-conv units and pooling engines to execute activation and maxpooling operation after convolution. Local kernel memories are also embedded inside each FC and Conv core for storing the filter coefficients needed by conditional DNNs and allow for fast access during operation.

The DEU is designed to support a variety of workloads to meet dynamic inference requirements. The conditional DNN configurations are stored in the Host-CPU, which maps them on the DEU part-by-part in runtime. The Host-CPU monitors the decision net outputs and dynamically maps the next part on the DEU by selecting an appropriate number of Conv cores and FC cores, configuring the cores, and setting the memory address space needed by the cores. Jobs are dispatched through the main controller and down to the local sub-controllers inside each FC and Conv core for performing scheduled operations.

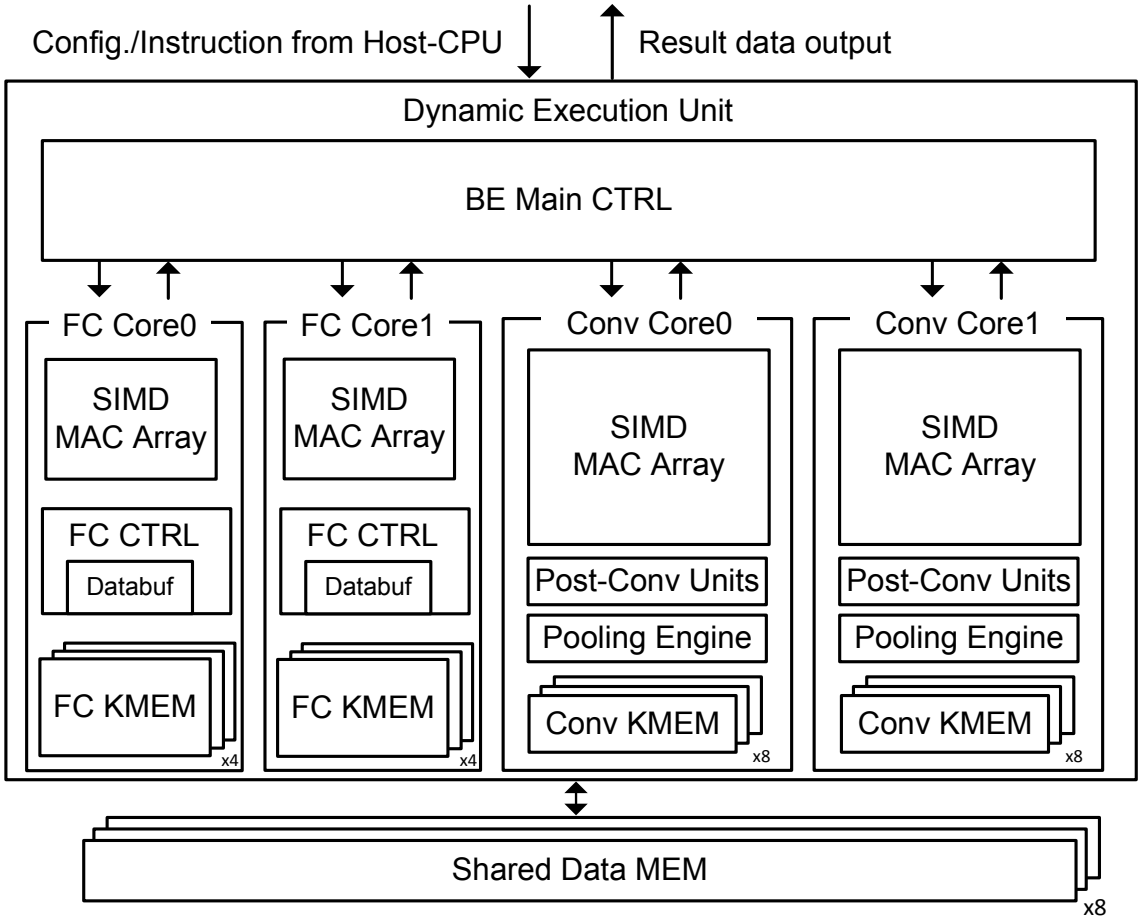


Figure 3.5: Dynamic Execution Unit (DEU) in back-end.

There is flexibility in configuring the DEU. If higher performance is desired, decision nets and the next compute nets can be both mapped onto the DEU. As shown in Figure 3.6, the next compute nets are speculatively executed without waiting for the decision nets' output, allowing the possibility of lower latency and faster processing speed. The lower processing latency can also be exploited to lower the supply voltage for reducing power consumption. When multiple decision nets and/or compute nets need to be executed, they can be mapped on the DEU in a pipelined fashion: when one net reaches the FC core, the Conv core is freed up to accommodate the next net. More importantly, the DEU makes it possible to perform load balancing. While an FC core is busy, the Conv cores can speculatively execute the next compute net or

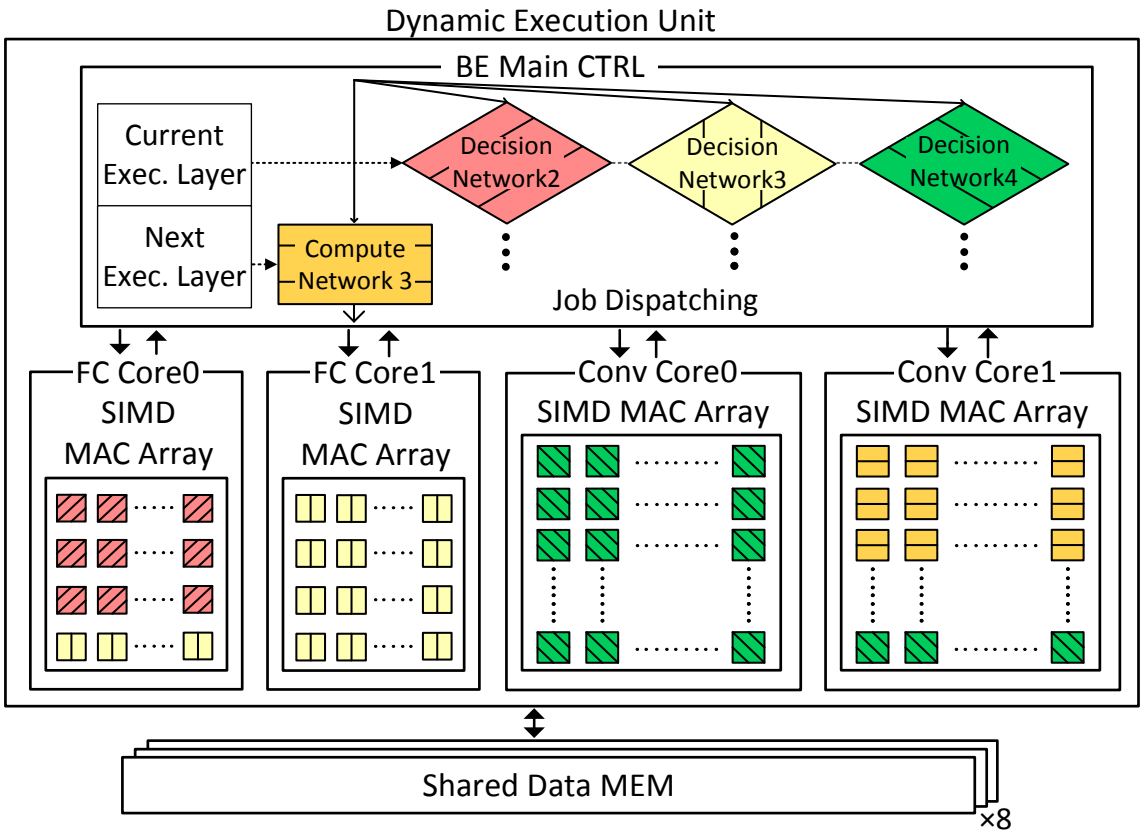


Figure 3.6: DEU workload balancing. The DEU computes multiple decision networks in parallel and speculatively executes compute networks in the next layer.

decision net. Load balancing also ensures the full utilization of the hardware to reach the highest possible efficiency.

3.5 Memory Architecture

To support irregular data access patterns during decision net execution, the back-end relies on a shared local data memory with 8 banks and 240b-wide I/O for fast data sharing. A custom-designed register file, called *frame station*, stores the memory data, enabling efficient data reuse. Due to their spatial locality, Conv/FC computations can be performed with notably lower memory bandwidth by reusing data in the frame station. Each Conv/FC core has a dedicated 1.5KB frame station tailored to accommodate the data transfer pattern of that core.

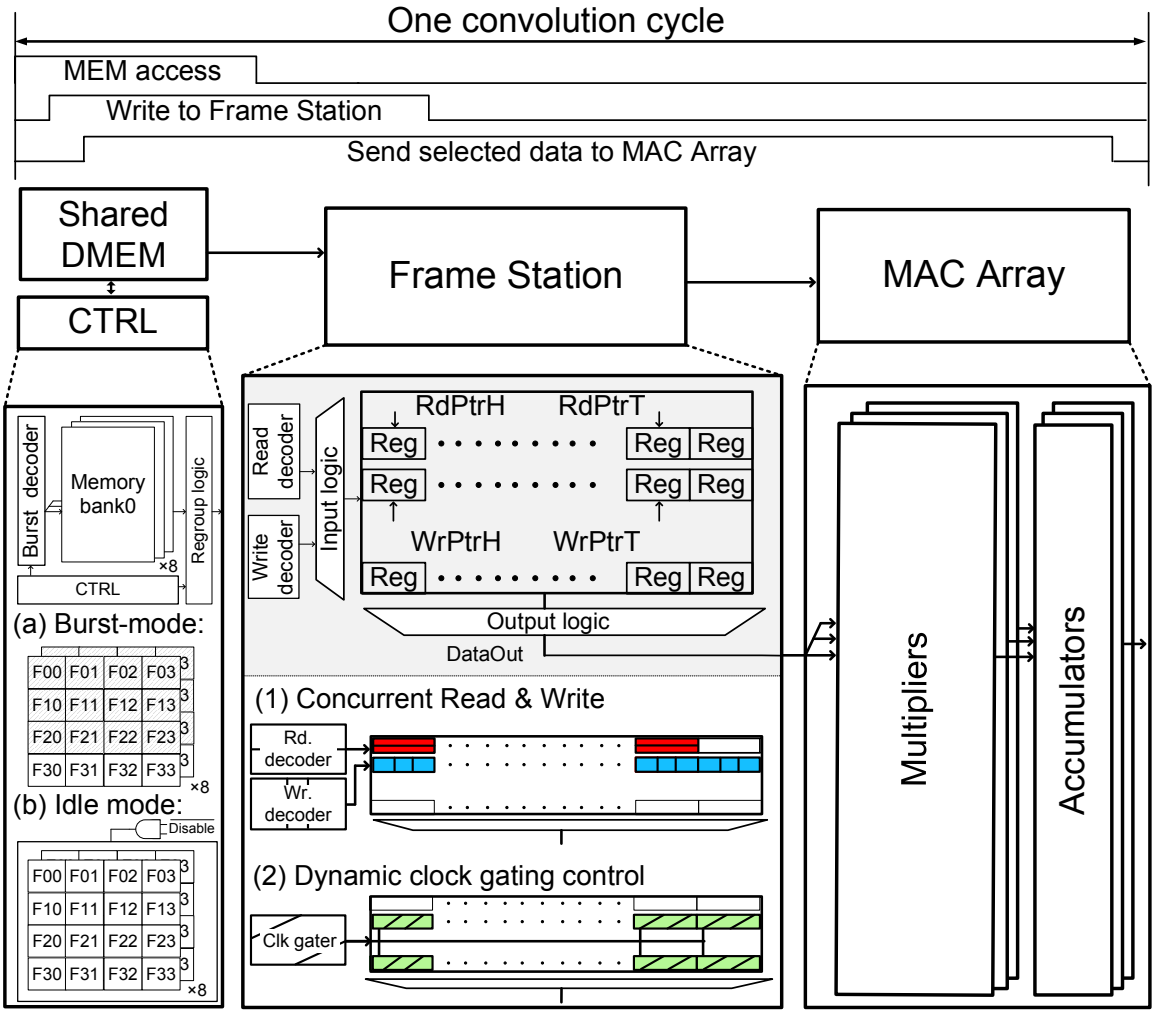


Figure 3.7: Burst-mode memory access.

The data memory and the frame station support burst-mode operation, as shown in Figure 3.7 for a Conv core. The data used in several consecutive cycles is fetched from the data memory in a single access, and the data memory is duty-cycled to save power. Convolution computations are coordinated with frame station operation through a 6-phase protocol. Each phase depicts a different stage of data memory access and frame station read/write operations for data caching and transfer, and MAC array operation. A detailed illustration is given in the following paragraphs, and corresponding graphs can be found from Figure 3.8 to Figure 3.10.

At the beginning of Phase 1 of a convolution cycle, as shown in Figure 3.8(a), burst-mode memory access initiates. The local controller inside the convolution core

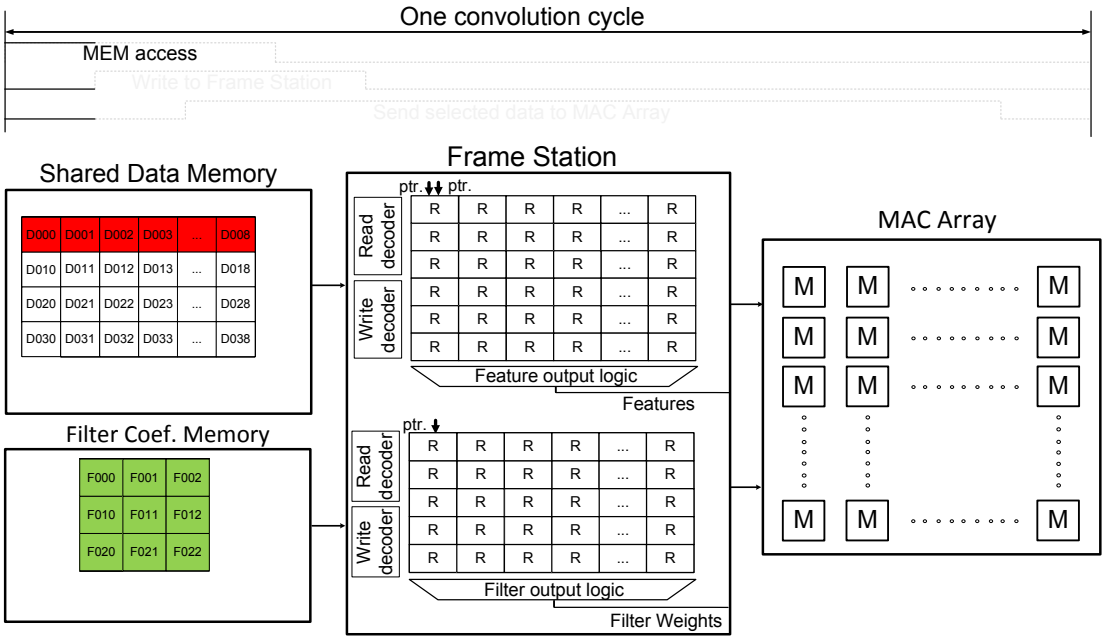
begins to send read requests. Features and filter weights are then read from target banks of shared data memory and filter coefficient memory. In Phase 2, shown in Figure 3.8(b), the data read from memories are stored into dedicated registers inside the frame station. The local controller performs proper regrouping upon receiving the data and updates write-head and write-tail pointers to indicate their storage locations when writing data into the frame station. At the same time, new read requests are sent to memories to retrieve data needed in later cycles.

In Phase 3, shown in Figure 3.9(a), after the first group of data is written to the frame station, the controller begins to select portions of stored data that are ready to be sent to the MAC array by updating the read-head and read-tail pointers to specify the range of the output selection. The MAC array then begins to perform computations needed in convolution operations starting in this phase. In the meantime, the controller continues to send new read requests and receives new data from a shared memory.

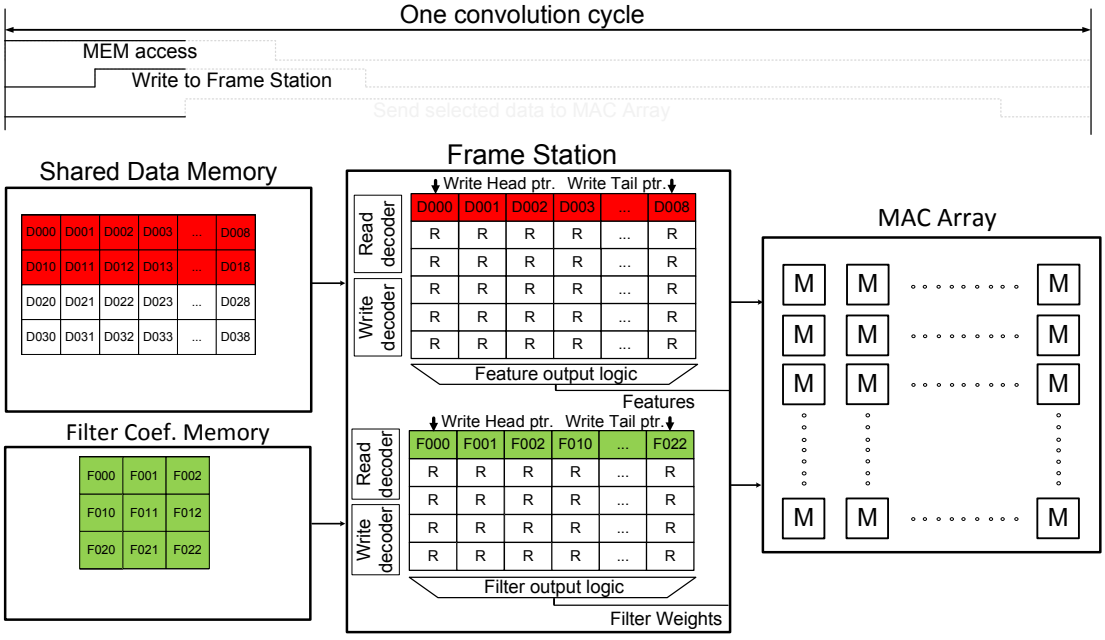
The data transfer between the memory and the frame station ends when the last group of data is fetched. As the controller puts memory peripheral circuitry into idle mode, it continues to adjust read pointer positions to send corresponding operands to the MAC array every cycle until all data are sent out for processing, as shown in Figure 3.9(b) (Phase 4) and Figure 3.10(a) (Phase 5).

Phase 6 begins when the last batch of data is received from the frame station, as shown in Figure 3.10(b). During this phase, the MAC array completes operating on the remaining data. After all results have been obtained and sent to the designated location for storage, the convolution cycle ends.

To achieve data parallelism when processing a high-dimensional convolution, the frame station opens multiple I/O ports that allow high-bandwidth data transfer between memories and the MAC array, as shown in Figure 3.11. Features and filter weights read from multiple banks of memory are stored in separate parallel locations

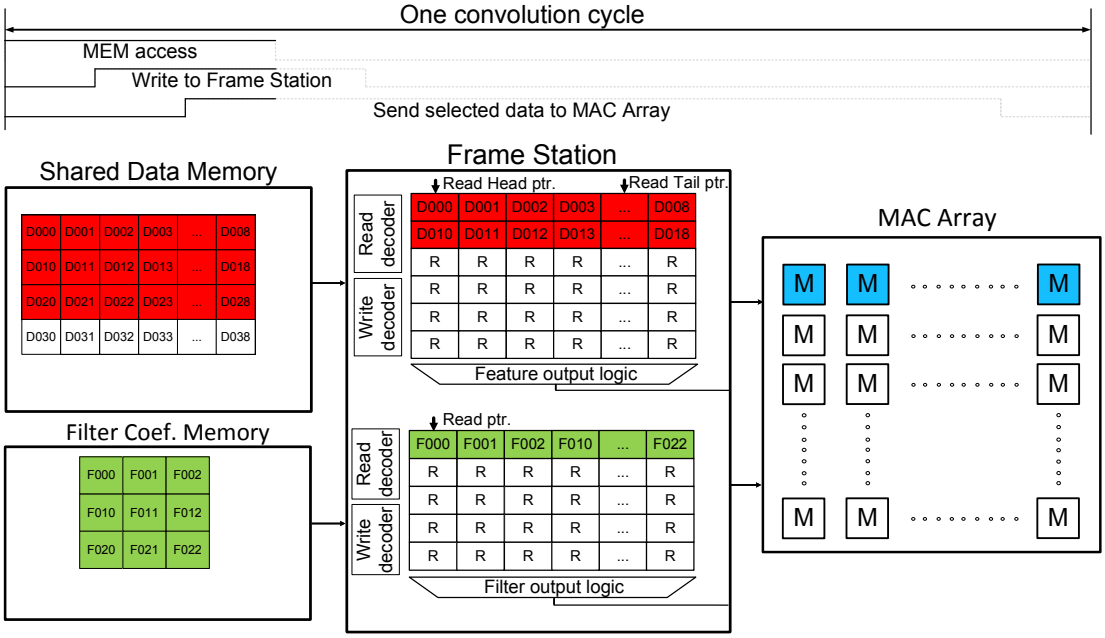


(a)

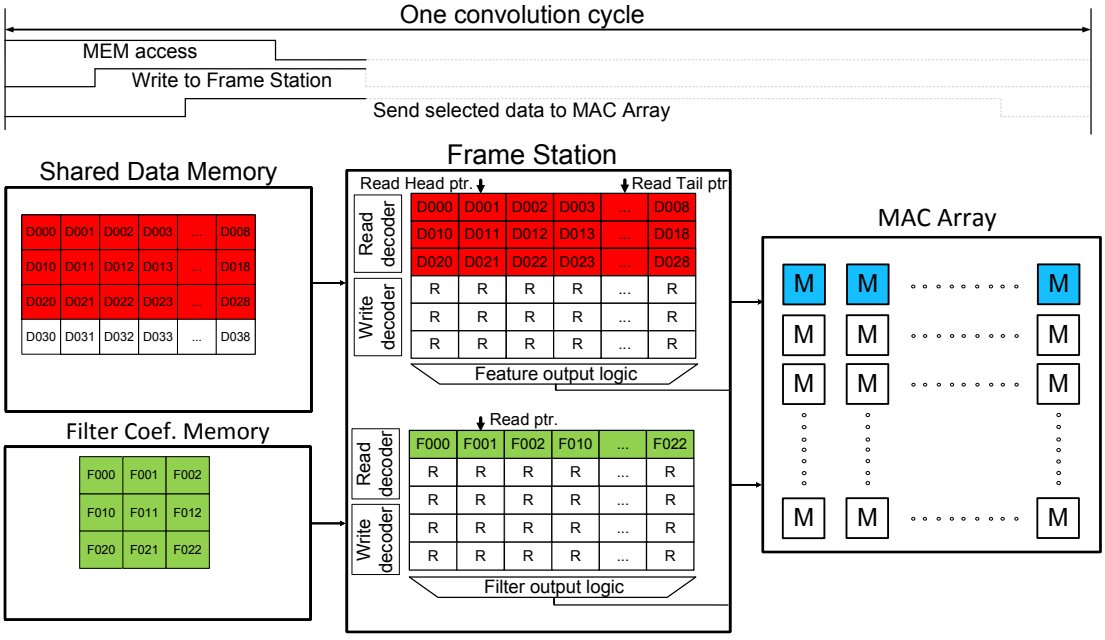


(b)

Figure 3.8: Low-power memory architecture operation. (a) Phase 1: Initial burst-mode memory access. (b) Phase 2: Continuous memory reading; initial data writing to the frame station.

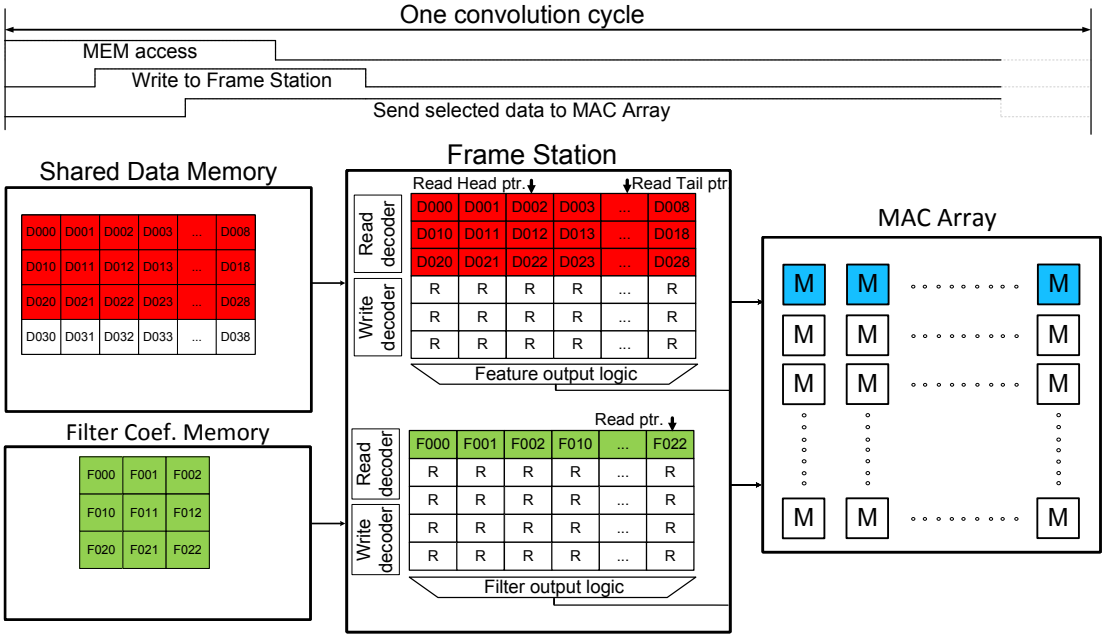


(a)

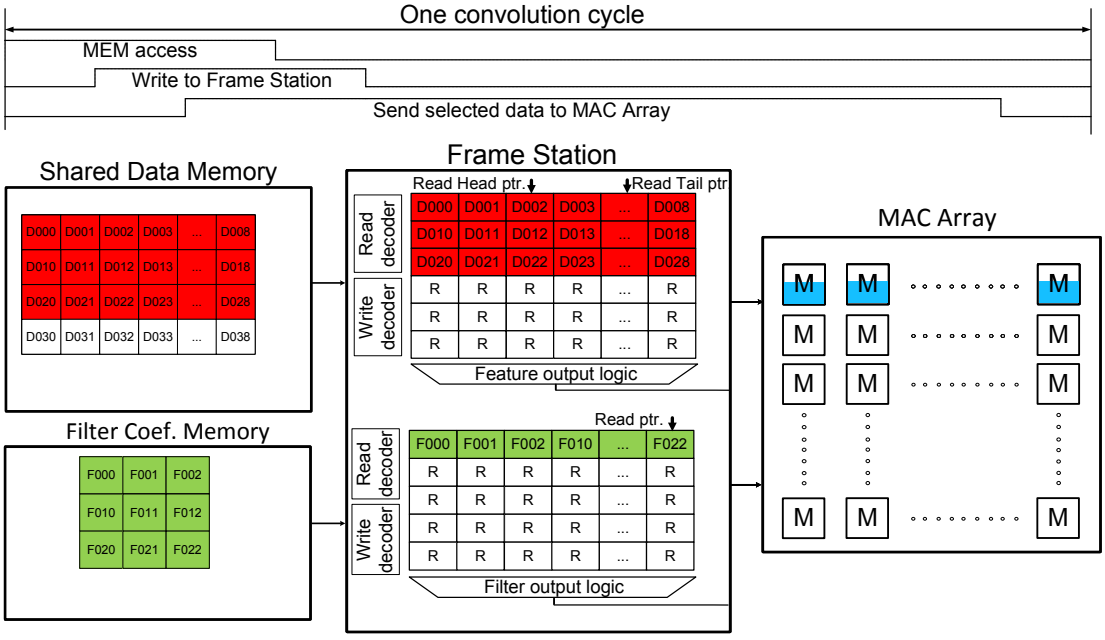


(b)

Figure 3.9: Low-power memory architecture operation. (a) Phase 3: Continuous memory reading; concurrent frame station reading and writing. (b) Phase 4: End of burst-mode memory access; updating selection range again for sending new operands to the MAC-array.



(a)



(b)

Figure 3.10: Low-power memory architecture operation. (a) Phase 5: Continuous data outputting from the frame station until the end of the convolution cycle. (b) Phase 6: Operating on last batch of data received from the frame station and finishing current convolution cycle.

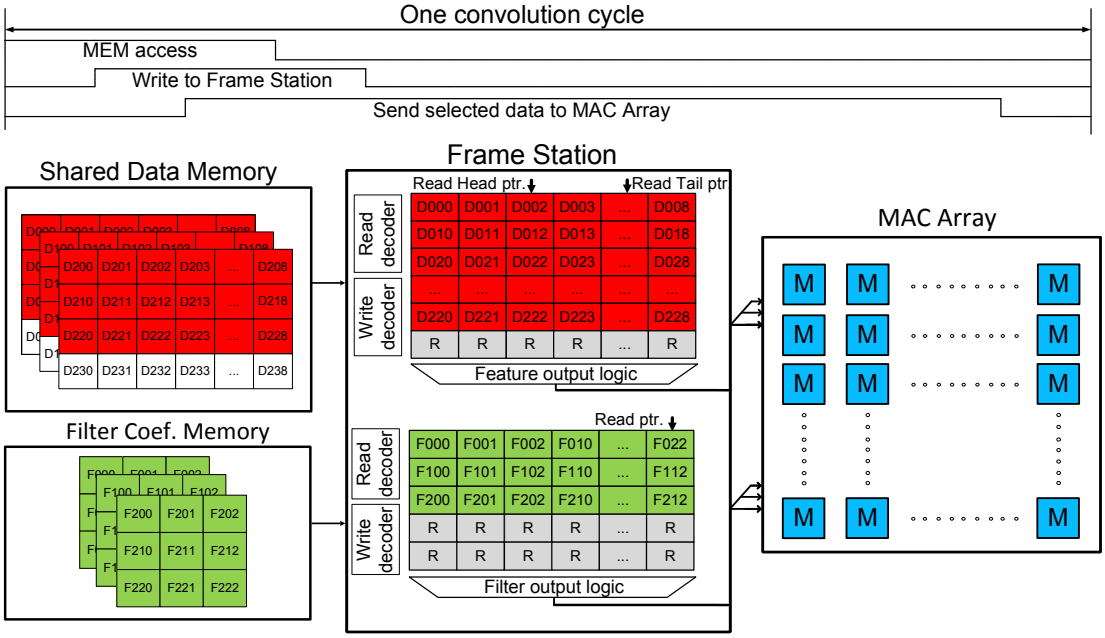


Figure 3.11: Data parallelism of each phase. The clock-gated registers are shown in grey.

in the frame station and are then fetched to selected MAC units inside the MAC array, as discussed in Section 3.4. The registers and logic that are not involved in the current operation are further clock-gated to save power.

The proposed memory architecture provides power efficiency benefits through burst-mode memory access, minimizing the energy cost of data movement during operation. Compared with a 64b-wide I/O memory with no burst mode, the burst-mode access reduces power by 29%, reducing the overall power consumption of the memory cell, word line, and sense amplifier during memory access, and saving power through duty-cycled operation. Unlike a typical FIFO, which keeps data moving to feed the compute pipelines, the frame station keeps data stationary to eliminate data movement costs through read/write head/tail pointers to keep track of the range of data currently being referenced and processed. In simulation, the frame station saves 54% of dynamic power compared with a FIFO implementation by keeping data stationary and activating only a portion of its access logic. Through data reuse, the

frame station reduces memory access by 67% with the smallest convolutional window (3x3, stride=1). Reductions are even higher when kernel size and overlap increase.

3.6 Chip Implementation

A test-chip implementing the proposed architecture was fabricated in a 40nm CMOS 1P10M technology. The core occupies 4.84mm^2. The chip microphotograph is shown in Figure 3.12. For the front-end, high V_{th} standard cells were used to reduce leakage current. We synthesized the design with 10ns clock period and performed functionality check in post automated place-and-route (APR) simulation to verify timing. For the back-end, we synthesized the design at 1.5ns and performed fine-grain pipelining to meet a more stringent timing constraint. Hybrid V_{th} cells including HVT and SVT devices are used during synthesis and APR. To further reduce power, we used commercial EDA tools to detect logic paths with extra timing slack and to replace SVT cells along the path with HVT ones.

During physical design, we used dedicated power supplies for front-end, back-end, and Host CPU. Custom level converter cells were designed and inserted across different voltage domains. To reduce the extra timing overhead in the final APR, we manually placed memory macros and performed hierarchical place-and-route. Memory locations were chosen to reduce signal routing length and placement-and-routing congestion. We first placed and routed the front-end processors individually. We then instantiated the front-end macro with unplaced logic and memories of the back-end design to perform second APR level. A third level of APR was performed with the Host-CPU and routed front-end and back-end macros. With this hierarchical flow, we resolved hold time issues introduced during the flattened APR process.

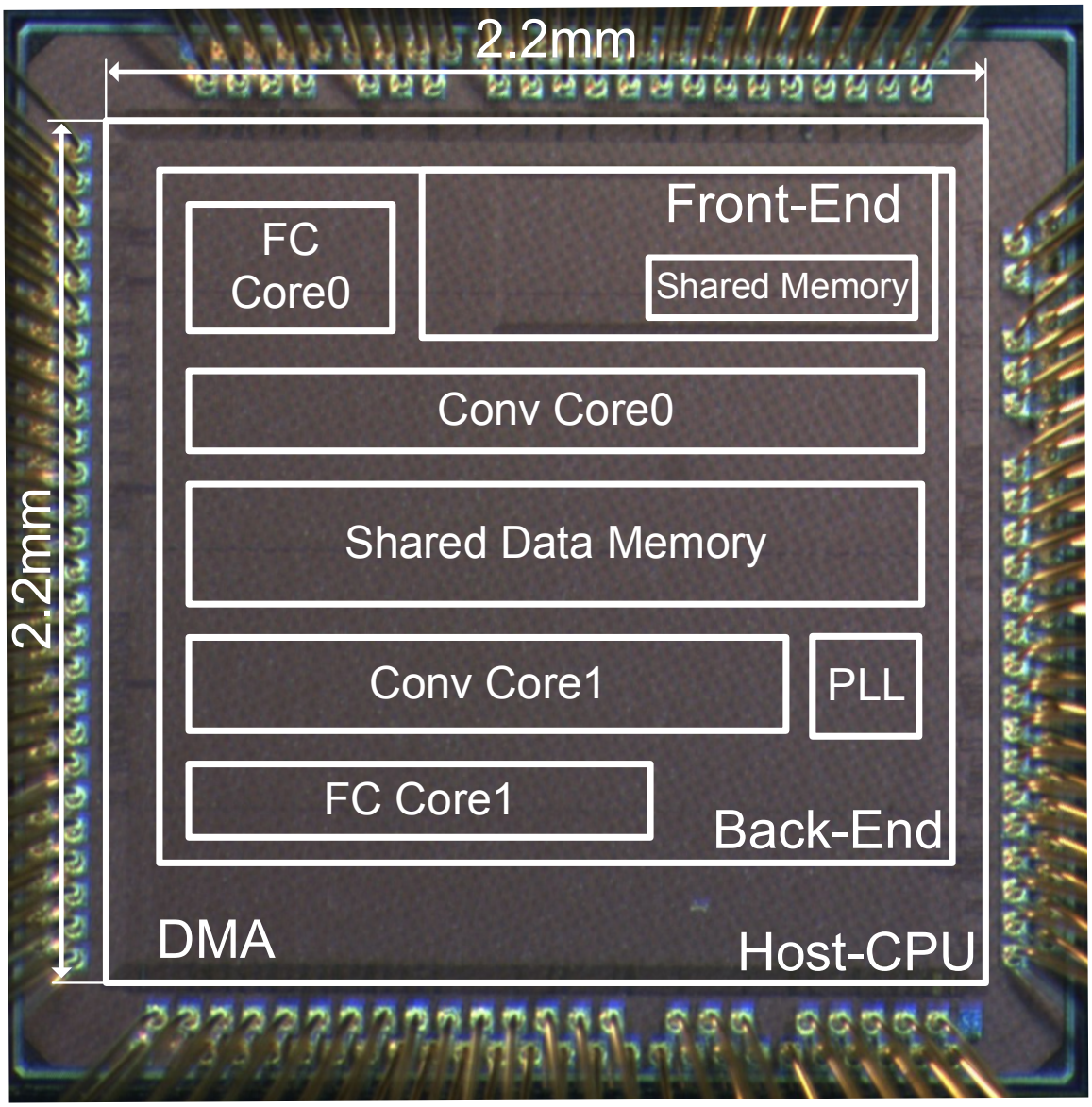
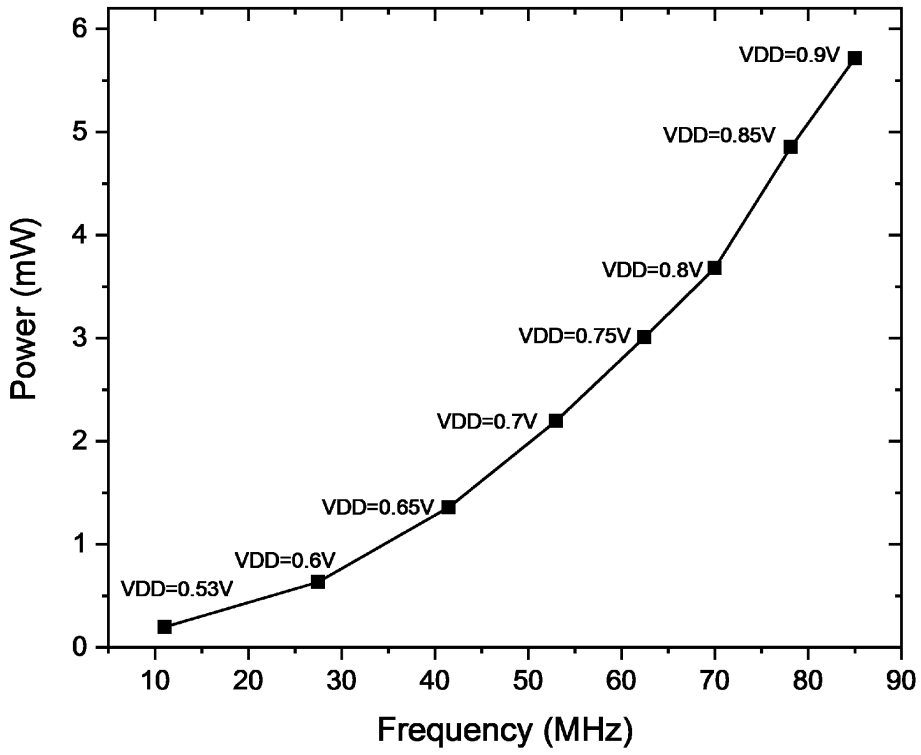


Figure 3.12: Chip micrograph of proposed design.

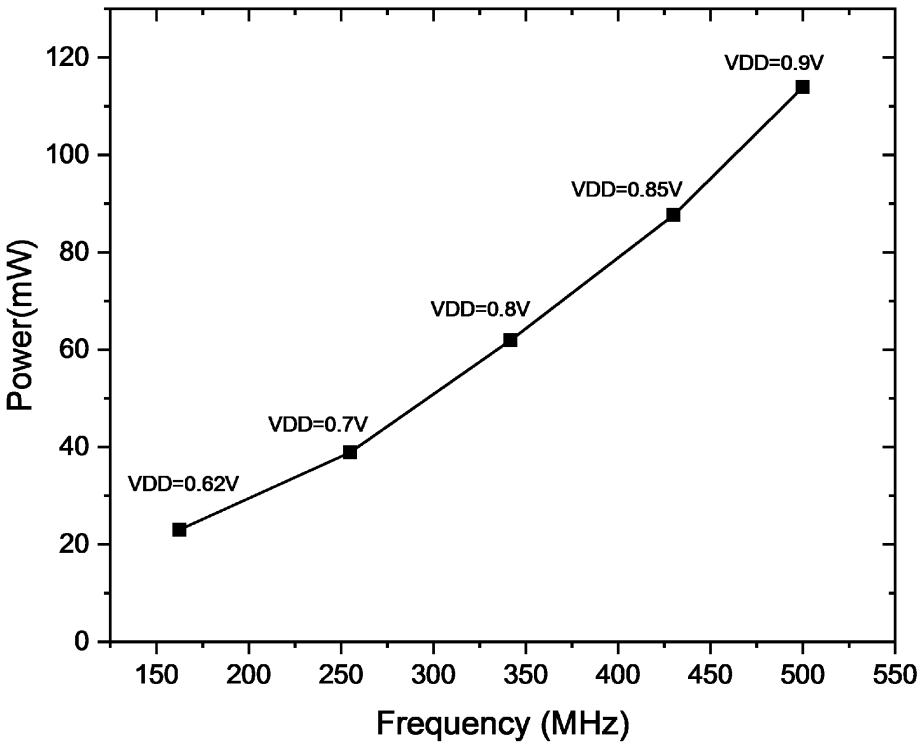
3.7 Experimental Results

Figure 3.13 shows measured power across a range of operating clock frequencies. Front-end power is as low as 0.19mW at 0.53V, 11MHz. The back-end achieves a peak clock rate of 500MHz at 0.9V, consuming 114mW.

To assess energy savings achieved by dynamic execution, we obtained the baseline power of the test-chip's back-end with the DEU disabled. We also trained conditional



(a)



(b)

Figure 3.13: Measured power consumption versus operating frequency for (a) front-end and (b) back-end processors.

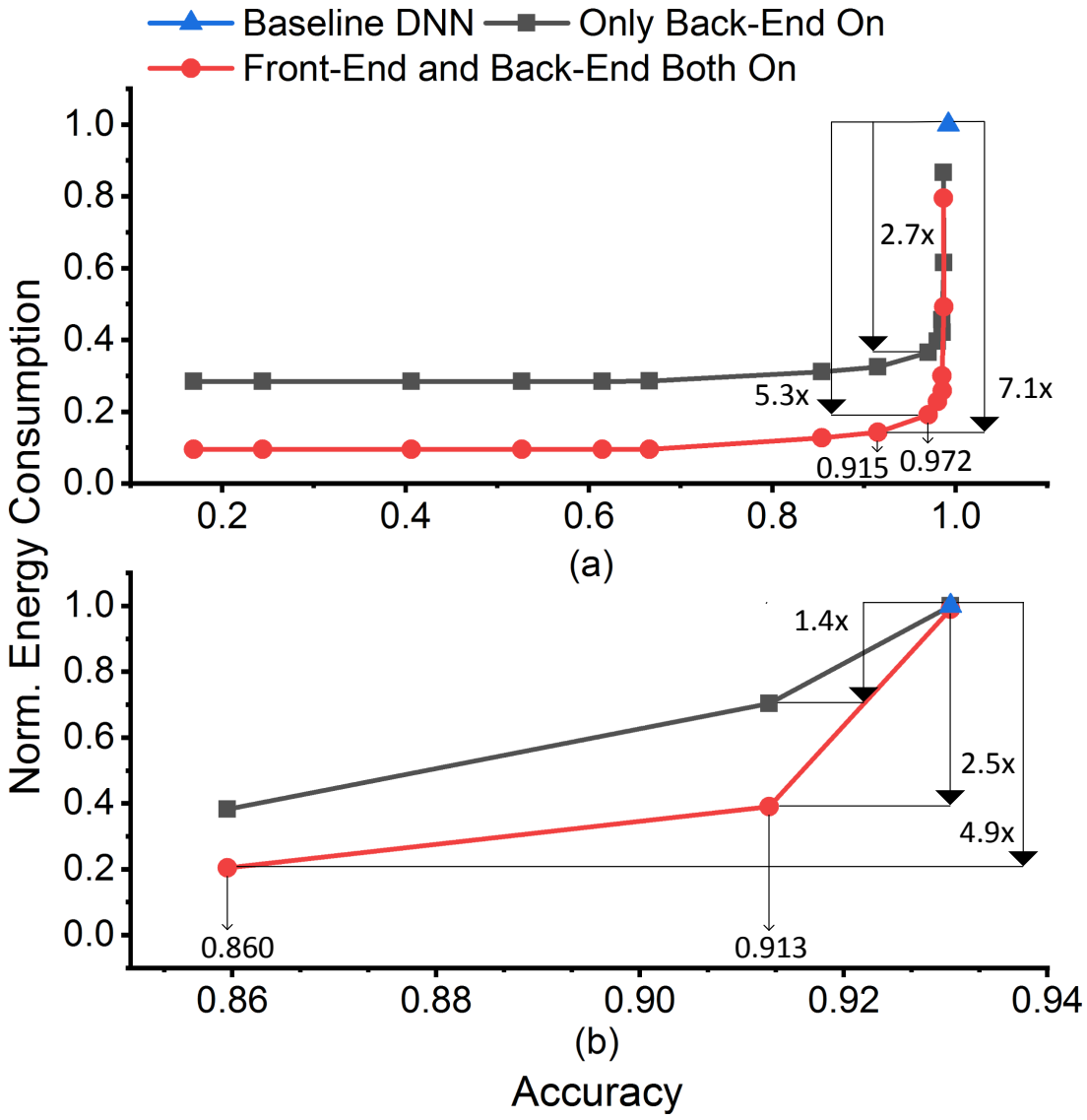


Figure 3.14: Normalized energy consumption versus accuracy for (a) LFW and (b) CIFAR-10 data sets.

DNNs using reinforcement learning with different rewards, thus obtaining different sets of weights that yield different decision net outcomes and trade off energy consumption for recognition accuracy. The graphs in Figure 3.14 show efficiency results of the test-chip on the LFW face dataset and the CIFAR-10 dataset for face and object recognition. For the LFW data in Figure 3.14(a), energy savings scale with relatively low loss in accuracy. Energy consumption decreases by 2.7× with 97.2%

accuracy rate (1.4% loss) when mapping all DNNs to the back-end with the DEU enabled for dynamic execution. Energy consumption decreases by 5.3× with the front- and back-end both activated. Using different sets of weights, the chip operates at various power/accuracy points. Energy consumption is reduced by 7.1× when accuracy rate decreases to 91.5%.

Figure 3.14(b) shows the results obtained with the CIFAR-10 dataset. In this experiment, we trained and tested a conditional DNN using ResNet as the baseline to show applicability to advanced benchmarks. Energy savings are 2.5× with 91.3% accuracy, and 4.9× with 86% accuracy using dynamic execution.

Table 3.1 compares our test-chip with previously published designs. Our processor consumes an average of 0.23mW at 5.3fps on the LFW dataset. Using dynamic execution, it achieves 2.49 TOPS and demonstrates a competitive energy scalability of 4.7-28.6 TOPS/W. By comparison, with the same dataset and accuracy, the processor with analog-digital front-end face detection and digital back-end face recognition in [43] consumes 0.62mW on average at 1fps. Moreover, the digital front-end in our chip provides support for different applications and energy scalability under various conditions. Compared with [5, 11], which combine voltage-frequency scaling and quantization using optimized configurable hardware, our test-chip achieves higher energy efficiency without sacrificing bit precision.

	This Work		ISSCC'17 [5]	ISSCC'17 [43]	ISSCC'18 [11]
	Front-End	Back-End			
Technique	HPDE[1]		DVAFS	HHFD(FD) + DM-CNNP/ SF-CONV (FR)	UNPU
Technology	40nm		28nm	65nm	65nm
Die Area	4.84mm^2		1.87mm^2	27mm^2	16mm^2
SRAM Size	34KB	240KB	144KB	N/A	256KB
Max Frequency	85MHz	500MHz	200MHz	100MHz	200MHz
Power	0.23mW		7.5mW	0.62mW	3.2mW
Peak Performance (TOPS)	2.49 (8b)		0.408 (4b)	0.072 (16b)	7.372 (1b)
Efficiency (TOPS/W)	4.7-28.6 (8b)		0.53 (16b) 10 (4b)	2.1 (16b)	3.08 (16b) 11.6 (4b) 50.6 (1b)

[1]Heterogeneous Processor with Dynamic Execution

Table 3.1: Performance summary and comparison.

3.8 Conclusion

In this chapter, we presented a heterogeneous low-power processor architecture that combines a low-power front-end with a high-performance back-end to support conditionally-executing DNNs. The proposed architecture is compatible with previous approaches to DNN processor design, such as bit-precision and sparsity optimization, enabling additional energy savings above and beyond what is achievable by those approaches. The back-end can be configured in runtime to support the most efficient execution of dynamic DNNs through speculative execution of compute nets and overlapped execution of compute nets and decision nets. To assess the efficiency of the proposed architecture, we designed and evaluated a test-chip fabricated in a 40nm process. On the LFW dataset, the test-chip achieves 5.3× lower energy with 0.23mW average power in comparison with a static execution baseline. With energy efficiency in the 4.7-28.6 TOPS/W range, it can adjust its operation to support edge devices under various operating conditions.

CHAPTER IV

Zero-Short-Circuit-Current Logic for Heterogeneous Architecture

In the heterogeneous architecture proposed in the previous chapter, the energy efficiency of the front-end is critical for the energy efficiency of the overall system. Most of the time the system only turns on the front-end processing block, executing the back-end in a heavily duty-cycled mode. Therefore, an ultra-low power technique for front-end implementation can significantly increase the energy efficiency of the overall system.

Low-power digital design has been a particularly active research topic for several decades now. While lots of research efforts target power optimization for the high-performance end, low-power digital design for moderate to low-performance applications still presents several challenges. Reducing the supply voltage is one of the common approaches for low-frequency design. Numerous efforts have explored the application of low-voltage design techniques to reduce power consumption in digital signal processors or general-purpose processors. However, aggressive voltage scaling causes robustness issues. As operating frequency is tightly coupled with supply voltage, voltage scaling design also reduces design choices, often requiring particular architecture implementations for meeting performance or throughput requirements.

To achieve higher energy efficiency and more reliable operation than what is

achievable by voltage scaling, we propose an energy-efficient computing technology, called zero-short-circuit current (ZSCC) logic. The energy efficiency of the proposed circuit technology is successfully demonstrated in a high performance hearing-aid device: a binaural dual-microphone hearing-aid chip processing 4 audio input streams and fabricated in a 65nm CMOS process achieves 9.7× lower power at 1.75MHz compared with the 40nm monophonic chip that represents the published state of the art [14]. Parts of the work described in this chapter have appeared in ISSCC 2017 [44].

To demonstrate the effectiveness of ZSCC in further reducing the power consumption of the heterogeneous neural network processor proposed in Chapter III, we have used ZSCC to implement the front-end of that processor. In netlist simulations, the ZSCC-based front-end achieves 17× lower power consumption compared with the static CMOS implementation of the same architecture, showing the superior power benefits it brings over conventional design for neural network computations.

The remainder of this chapter is organized as follows: Section 4.1 gives background on charge-recovery logic. Section 4.2 presents the proposed zero-short-circuit-current logic. Section 4.3 describes the operation of zero-short-circuit-current logic. Section 4.4 describes synchronization and four-phase clock generation with ZSCC gates. Section 4.5 explains the proposed semi-automated design methodology for ZSCC logic. Section 4.6 presents a hearing-aid design with ZSCC logic. Section 4.7 shows the custom chip-on-board design and experimental setup. Section 4.8 describes the experimental results of the hearing-aid chip. Section 4.9 highlights the ZSCC-based design for the front-end of the heterogeneous processor described in Chapter III and compares it with static CMOS implementation. Section 4.10 summarizes this chapter.

4.1 Charge-Recovery Logic

As discussed in Section 2.2.2, charge-recovery design is a compelling alternative to static CMOS design for low-power VLSI implementation. A variety of charge-recovery techniques has been proposed and applied to the design of microprocessors or DSP processors [45–53]. Among these techniques, charge-recovery logic has the potential to achieve significant energy savings. Unlike resonant-clocked designs which only recover energy from the clock distribution network and clock-related sequential elements [51, 54–57], charge-recovery logic relies on a charge-recovery clock signal to perform logic evaluation and recover charge from the fanouts of logic gates. For instance, Boost logic in [53] is capable of operating at GHz-level frequencies while achieving energy recovery rate of 60%.

A variety of circuit topologies has been proposed for charge-recovery logic design, addressing energy optimization at high and low performance levels. Equation (2.6) implies that charge-recovery logic has the potential to achieve dramatic savings in energy efficiency when charging and discharging times are prolonged. In practice, however, realizing this potential is challenging, as short-circuit currents result in high power consumption at low frequencies. Figure 4.1 shows the measured energy consumption versus operating frequency of a recently published state-of-the-art charge-recovery logic, called subthreshold boost logic (SBL), presented in [58,59]. Test results of SBL chips confirm that energy decreases with operating frequency, as expected from Equation (2.6). However, at very low clock frequencies, energy consumption increases sharply due to multiple short-circuit paths during operation. Generally, short-circuit current is present along three paths: DC supply to ground, power-clock to ground, and DC supply to power-clock. While different designs have different ratios of energy loss from these three paths, any one of these paths could easily dominate total energy consumption at low frequency operation and cause a deviation from theoretical values.

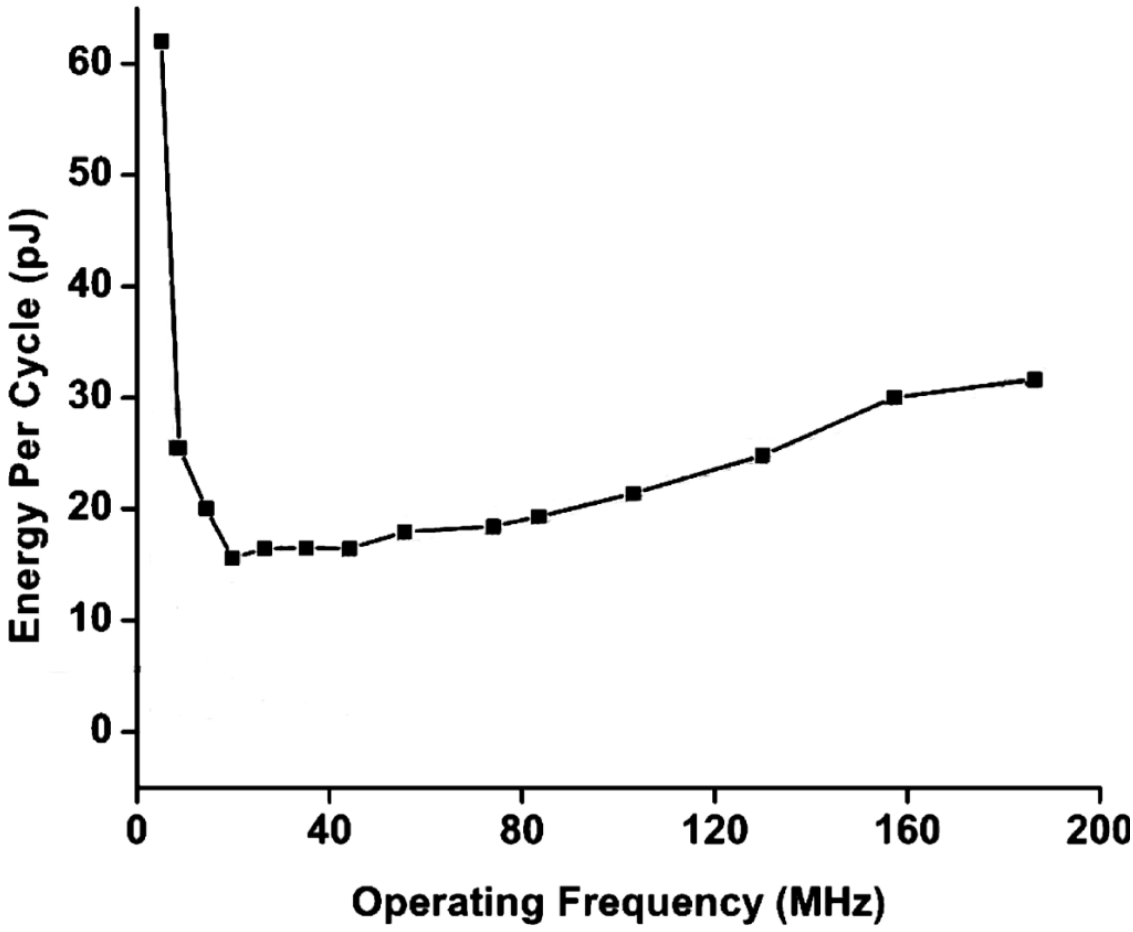


Figure 4.1: Measured energy consumption versus operating frequency for SBL FIR filter.

4.2 Zero-Short-Circuit-Current Logic

To address the issue illustrated in Section 4.1, we propose a new charge-recovery logic family, called zero-short-circuit current (ZSCC) logic, that eliminates possible short-circuit current paths and leverages charge recovery to operate with high energy efficiency at relatively low clock frequencies. The schematic of a ZSCC gate is shown in Figure 4.2. ZSCC is a dynamic dual-rail logic consisting of a logic stage and a state holder. The logic stage has differential outputs out and out_b. Each output is connected to a pull-up networks (PUN) and a pull-down network (PDN) with high-V_{th} NMOS devices for function evaluation. The state holder is composed of

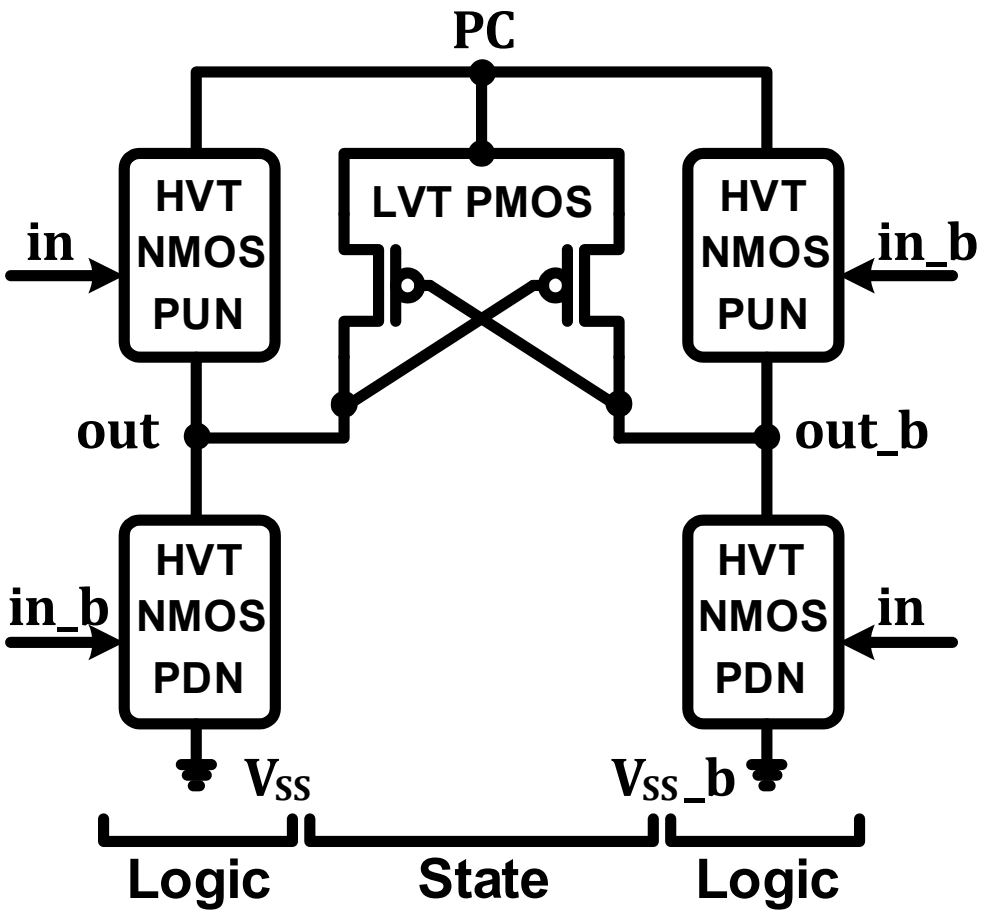


Figure 4.2: Schematic of a ZSCC gate.

a pair of cross-coupled low-V_{th} PMOS. Both the logic stage and the state holder are connected to a charge-recovery supply power-clock (PC), which is generated by inductive elements. Each ZSCC gate works as a small pipeline stage which uses PUN and PDN as combinational logic to drive the cross-coupled PMOS as the transparent latch for holding state. To achieve correct operation, the ZSCC datapath is formed by clocking cascades of ZSCC gates with a four-phase clocking scheme. The operation of ZSCC gates is explained in detail in Section 4.3.

As ZSCC is a cell design aimed at medium to low-speed applications, the timing requirements for the ZSCC micro-pipelines can be met relatively easily. We therefore use high V_{th} devices for the PUN and PDN function evaluation stages to reduce leakage current. Unlike static CMOS logic that uses PMOS devices in pull-up networks, the

PUN of ZSCC is implemented by NMOS devices for providing smooth transition at the beginning of evaluation, thus obeying adiabatic switching principles. Moreover, ZSCC is designed to use a single power supply PC for operation. Compared with static CMOS design, ZSCC does not have a DC power supply, thus avoiding possible short-circuit current paths between the DC supply and ground or the DC supply and PC to increase energy efficiency.

4.3 ZSCC Operation

With every cycle, ZSCC operation is divided into four stages: evaluate, hold, recover, and wait. Figure 4.3 and Figure 4.4 show details of its operation. The left hand side highlights the parts of the gate that conduct electrical current during each stage. The right hand side shows voltage waveforms for the two outputs out and out_b with the power-clock overlaid, and current waveforms for vss and vss_b. During evaluation, the first of the four stages, inputs are held stable, as shown in Figure 4.3(a). One of the output nodes is charged up by PC, initially through the corresponding pull-up network and eventually to full-rail through the cross-coupled PMOS devices. The pull-up network limits the voltage drop between the PC node and the charging output node, thus limiting switching energy by preventing current spikes during operation, and maintaining a relatively low voltage drop across the charging NMOS devices.

As PC attains full voltage, ZSCC enters the hold stage, as shown in Figure 4.3(b). The output nodes are held stable through the PMOS pair, and fanout gates evaluate. At the same time, input signals gradually decrease, and pull-up and pull-down networks gradually turn off.

When power-clock PC begins to fall, as shown in Figure 4.4(a), the charge at the output nodes is recovered through PC. The output voltage follows the power-clock till approximately V_{th} levels, where the PMOS cannot conduct any longer. To

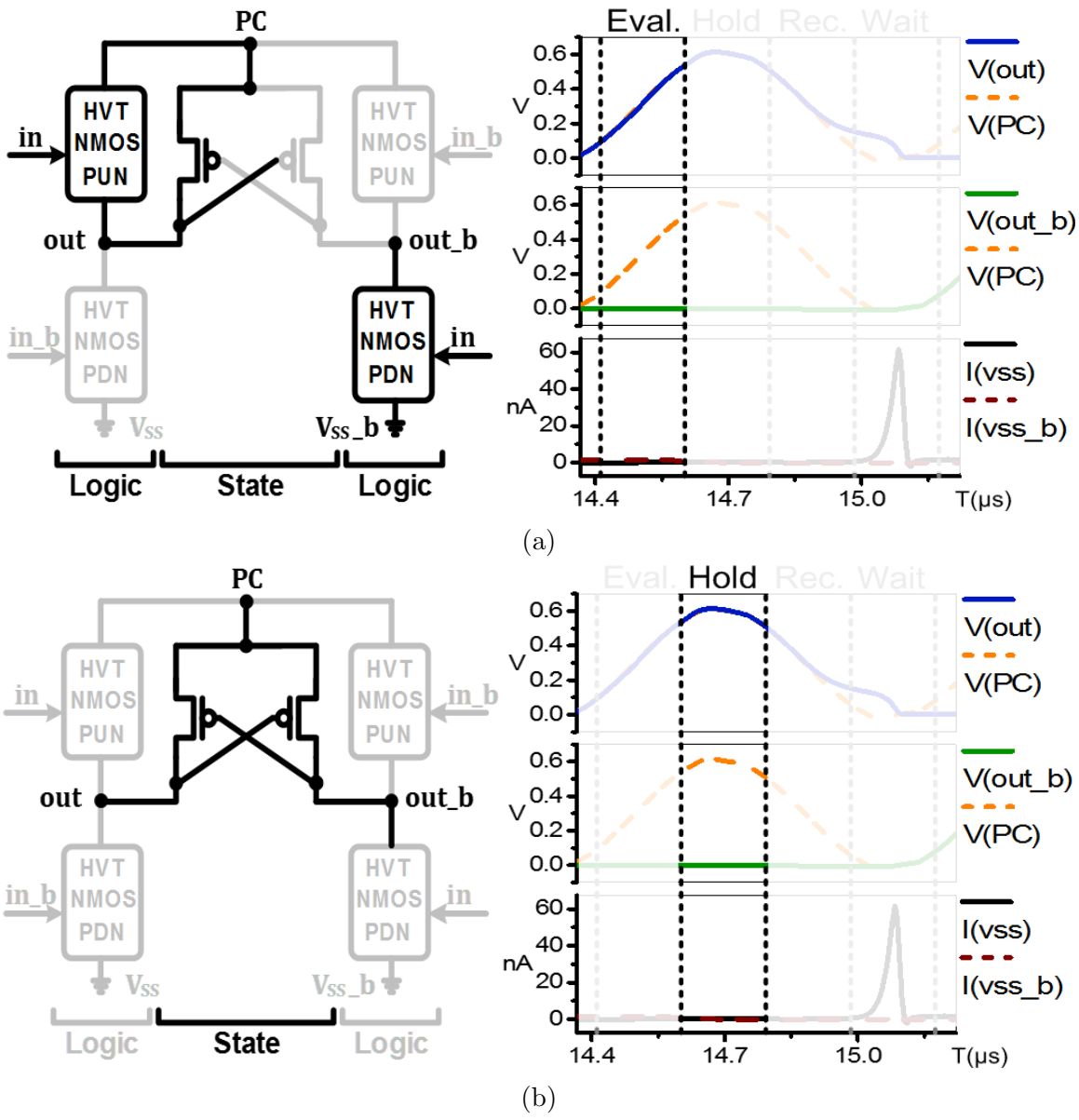


Figure 4.3: ZSCC operation: (a) Evaluate Stage. (b) Hold Stage.

further increase recovery rates, we use low-V_{th} PMOS devices for state holder in implementation.

During the wait stage, the last of the four stages, the PC remains at zero voltage, as shown in Figure 4.4(b). The inputs to the pull-up and pull-down network gradually build up, as fanin gates evaluate. Thus, any residual charge at the outputs that has not been recovered is discharged to ground or PC before the next cycle begins. No current flows into ground except during the wait stage, as the ZSCC gate is designed

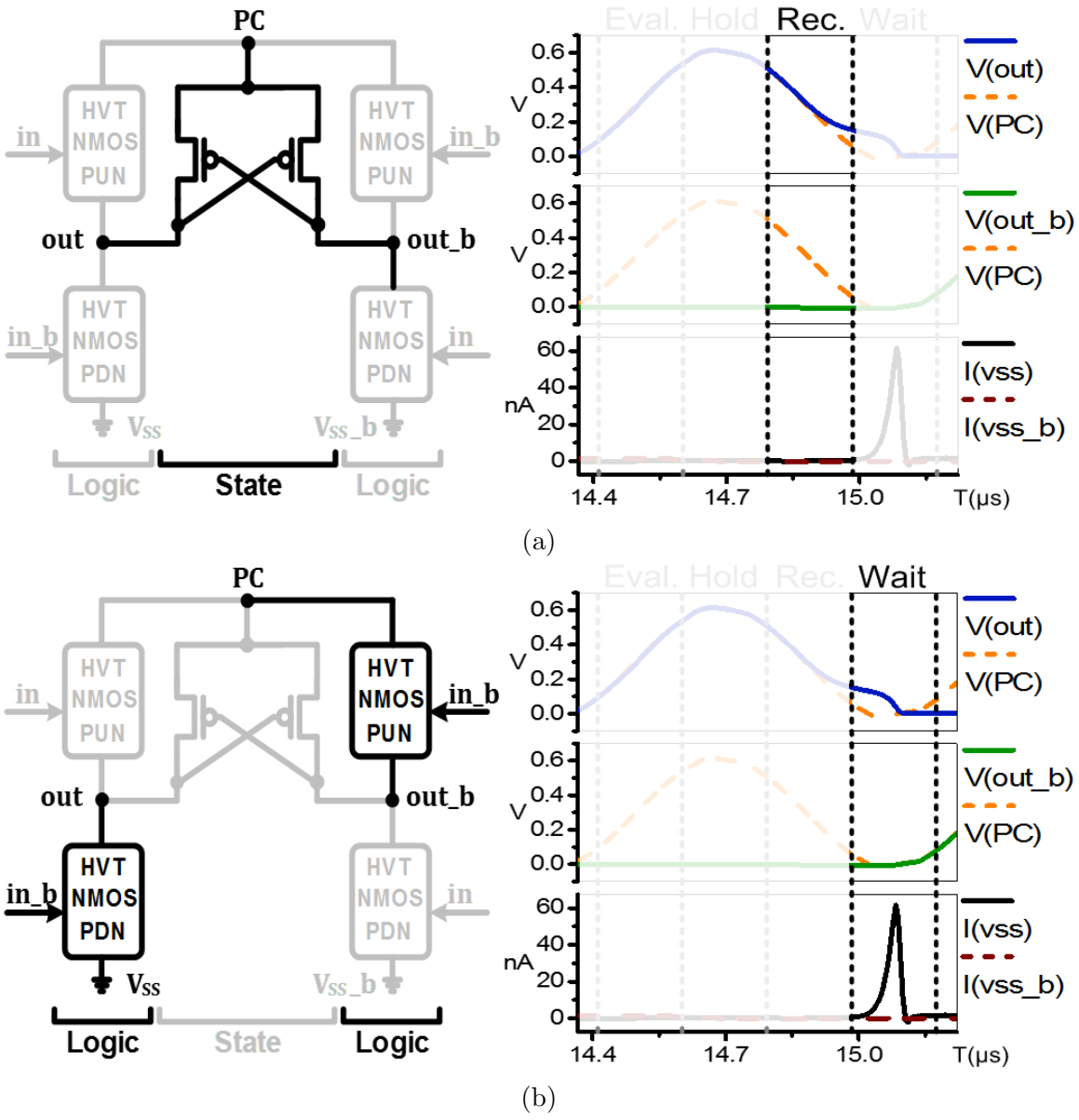


Figure 4.4: ZSCC operation: (a) Recover Stage. (b) Wait Stage.

to reset its output nodes to zero voltage only at this stage and allow current flows into ground only when PC is low.

As evidenced by the current simulation results shown in Figure 4.3 and Figure 4.4, no short-circuit current is observed during the four stages of ZSCC operation. Unlike previous charge-recovery logic families such as SBL [59], that introduce multiple short-circuit current paths due to interleaving of stages, the four-stage ZSCC operation provides time to reset outputs and prevents complementary output signals

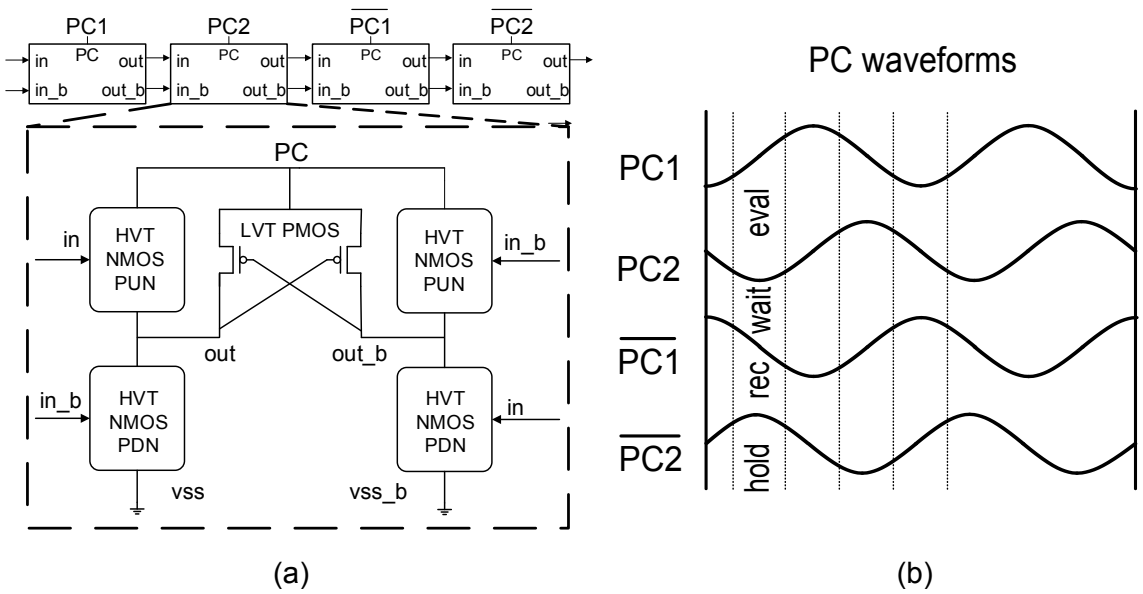


Figure 4.5: (a) Cascade of ZSCC gates. (b) Four-phase power-clock waveform.

from overlapping. Moreover, ZSCC logic is designed to operate without a DC supply, thus preventing the possibility of connecting power and ground during operation. Compared with the charge-recovery logic in [60], ZSCC limits voltage drop between PC and output nodes through the introduction of the two PUNs, preventing current spikes during operation and forcing the charging NMOS devices in deep triode region to function as ideal resistors. As shown in Figure 4.3(a), the output voltage closely tracks PC during the evaluation stage, consistent with adiabatic design principles that require only small voltage drops across conducting resistive paths.

4.4 Gate Synchronization and Four-Phase Clock Generation

A four-phase power-clock is used to synchronize ZSCC gates and ensure correct operation of ZSCC cascades. As shown in Figure 4.5, cascades of ZSCC gates are formed by clocking the gates using four phases, PC1, PC2, $\overline{\text{PC1}}$ and $\overline{\text{PC2}}$, in quadrature. Adjacent gates are supplied by power-clocks with 90 degrees phase difference to ensure correct timing.

The four-phase power-clock waveforms required by ZSCC can be generated by two

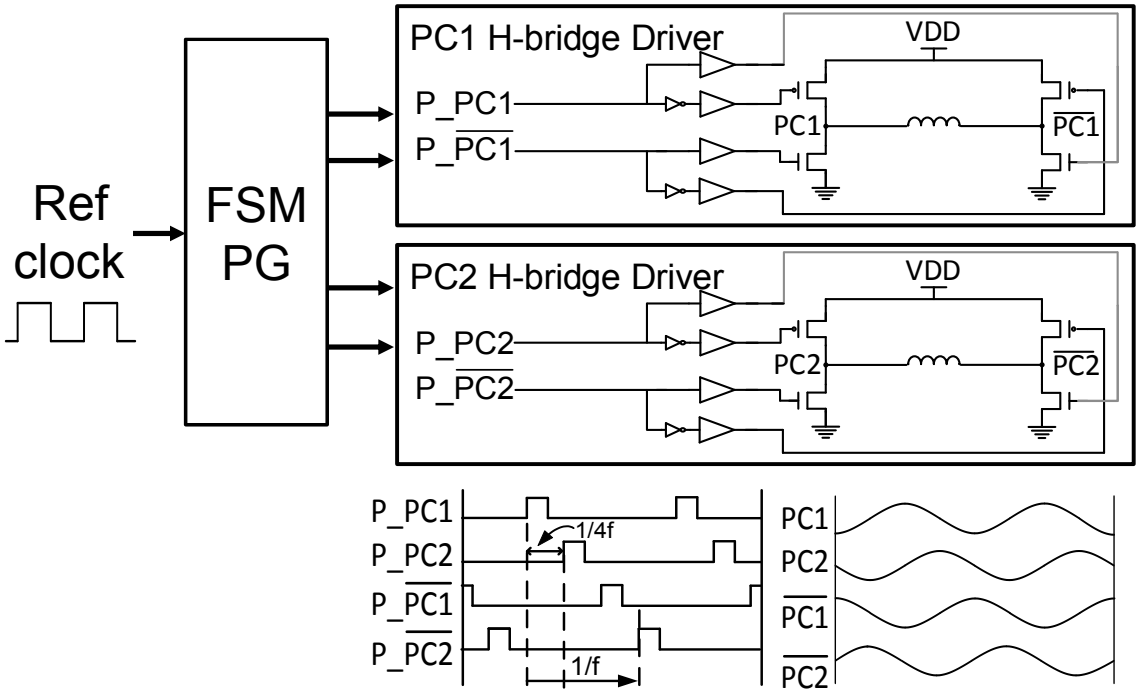


Figure 4.6: Four-phase power-clock generator.

H-bridge drivers, as shown in Figure 4.6. Each H-bridge generates two 180 degree out-of-phase power clock signals PC and $\overline{\text{PC}}$. A reference clock signal is supplied to a finite-state machine PG that generates four pulses with programmable duty cycle. These pulses are 90 degrees out-of-phase and are used to drive the two H-bridge drivers. Each H-bridge driver then produces control signals to pull-up and pull-down MOS devices that are connected to an inductor to replenish energy losses when resonating the parasitic capacitance of the clock distribution network and ZSCC gates. To allow for efficiency optimization, the pull-up and pull-down devices are implemented through an array of MOS devices that can be selectively turned on for adjustment at different operating frequencies.

A key benefit of an H-bridge driver is that it reduces the crowbar current of the LC oscillator. In prior work, a blip clock generator has been used for generating 180 degree out-of-phase clock signals [61]. The work in [62] proposes a quadrature LC oscillator through coupling of two blip-like LC oscillators. However, these schemes are

optimized for high speed design such as the resonant-clock latch-based filter design in [63] or the SBL-based low-density parity-check (LDPC) decoder in [64]. In low-end applications, however, the cross-coupled MOS devices used to replenish dissipated energy suffer greatly from short-circuit currents, due to the slow transition of the signals. Therefore, the external driver approach in H-bridge is more suitable for generating the four-phase power clock for ZSCC logic. Moreover, since the speed requirement is not as high as in [63] [64], the self-resonant mode is not necessary if we can design the clock distribution network properly to reduce clock skew to an acceptable range.

4.5 Semi-Automated Design Methodology for ZSCC Logic

In modern digital VLSI design, designers typically use a library of static CMOS cells, called *standard-cell library*, to map register-transfer level (RTL) design to physical design. Foundries provide characterized standard cells with timing information, which is used by commercial electronic design automation (EDA) tools to convert high-level system specifications to physical layout. This standard-cell design flow shields the front-end RTL design from back-end physical design considerations and saves design time by abstracting away lower-level stages of design.

To enable the use of ZSCC logic in large-scale VLSI design, we propose a standard-cell-like semi-automated design flow. The proposed design flow consists of three key stages, as in standard-cell design flow: RTL design and netlist synthesis, cell library physical design, and back-end design. We have successfully applied the proposed flow for the silicon prototyping of a hearing-aid application in Section 4.6.

4.5.1 RTL Design and Netlist Synthesis

With a target application in mind, an RTL description of the hardware architecture is first generated in the same manner as in a conventional CMOS design flow.

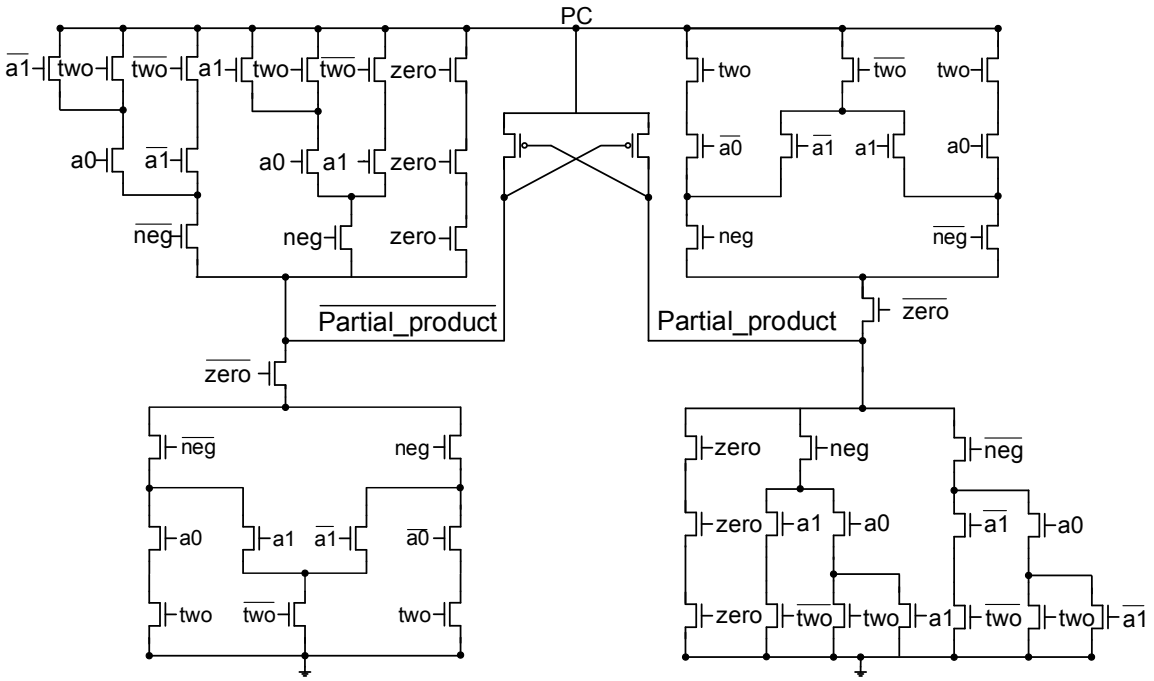


Figure 4.7: Schematic of a ZSCC Booth selector for generating a partial product.

When the high-level RTL description of the system is finished, a commercial synthesis tool is used to convert a sub-block of the design to ZSCC pipeline stages. Similar to the flow in [64], basic standard cells with low logic fanins are used during synthesis stage. We annotate each basic cell with a unit delay in the synthesis setting. After specifying the clock period in unit delays, the tool synthesizes the design to meet the constraint. The logic depth between pipeline stages is therefore equal to maximum stack height for ZSCC gates. The designer can tune the timing constraints provided to the synthesis tool to adjust the stack height of ZSCC gates. The stack height is determined by considering overall latency in the system and leakage power at operating frequency. In the prototype implementation described in Sections 4.6 through 4.8, we constrained the stack height to be 6, meeting the real-time performance requirements of the application with minimal leakage current when the design operates at 2MHz.

After synthesis is completed, the pipelined stages are converted to ZSCC logic cells using the procedure in [65]. To give an example of the converted ZSCC design, a ZSCC Booth selector used for Booth's multiplication algorithm is shown in Figure 4.7. The

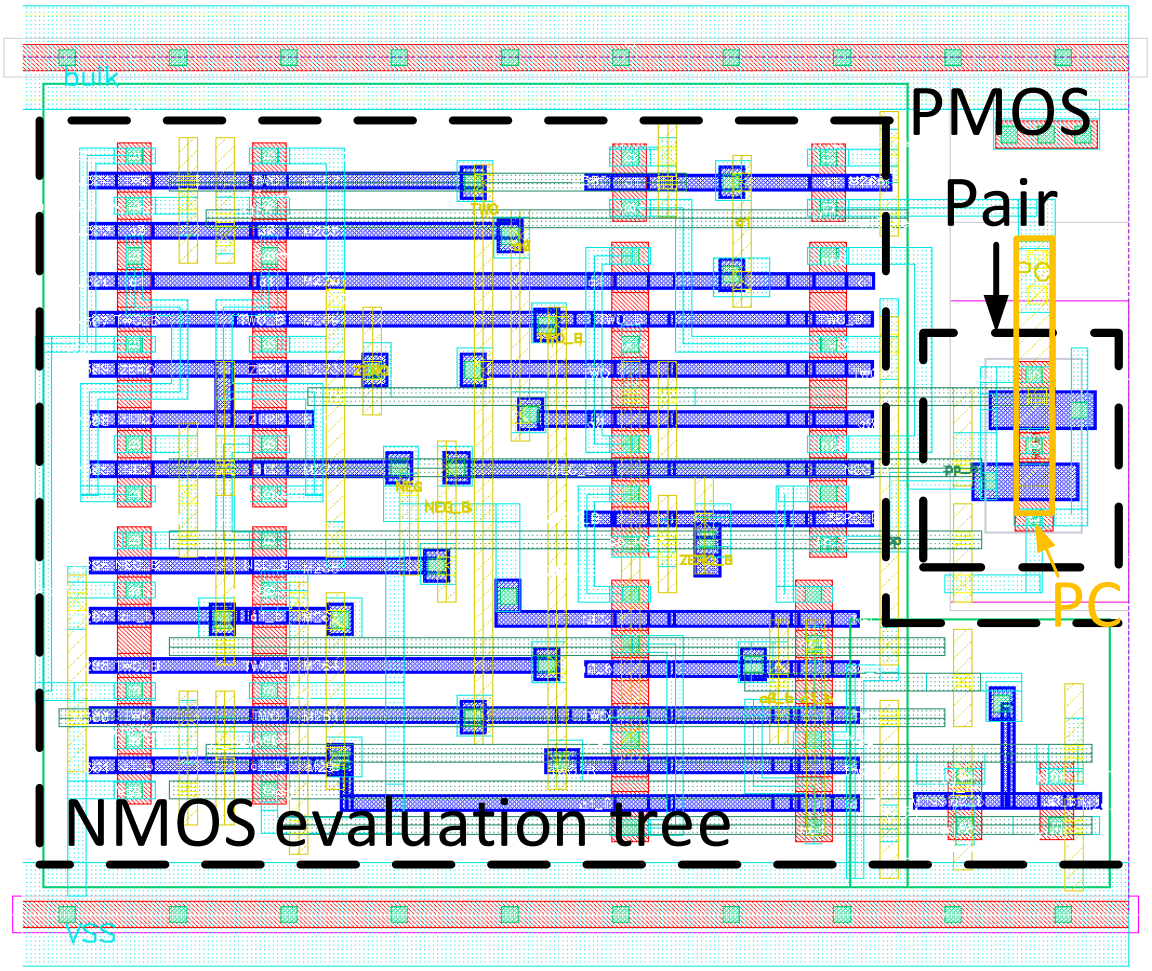


Figure 4.8: Layout of ZSCC Booth selector gate.

Booth selector takes zero, two, and negative flags to generate corresponding partial products during the Booth encoding process.

4.5.2 Cell Library Design

To support an automated place-and-route (APR) flow, a library of ZSCC cells that are compatible with the commercial APR tool needs to be designed. During the APR process, the APR tool automatically chooses the location for cell placement and performs detailed signal routing. In this process, various loading conditions arise for ZSCC gates with the same logic function. A number of different physical implementations of ZSCC gates are thus required to support the APR flow. To realize

ZSCC Implementation	Standard Cell Implementation
$28.80\mu m^2$	Combinational Logic: $26.64\mu m^2$
	Flip-Flop: $9.00\mu m^2$

Table 4.1: Area comparison for Booth selector.

these implementations, a SPICE simulator is used to choose the transistor width and length for each ZSCC gate and to ensure correct function at the target frequency and under the given output loading. For the hearing-aid test-chip described in Sections 4.6 through 4.8, we implemented a ZSCC library consisting of 64 cells with drive strengths ranging from 10fF to 150fF. For each cell, we performed SPICE simulations to verify correct operations at different process corners at the 1MHz to 2MHz frequency range.

After circuit topology and transistor width and length are determined, a commercial back-end EDA tool is used for cell layout design. A standard-cell like layout design is adopted for supporting APR flow. An example cell layout of a ZSCC Booth selector is shown in Figure 4.8. A metal-1 V_{SS} strip connecting to ground voltage is distributed on the bottom of the cell layout. As ZSCC has no DC power supply, bulk supply for body bias of the PMOS pair is routed on the top of the cell. The cross-coupled PMOS pair could be placed on the left, middle, or right of the cell, as shown in Figure 4.8, depending on the influence of location on cell area utilization.

To compare with static CMOS design, we synthesize the Booth selector logic with a standard cell library operating at the same frequency, as shown in Table 4.1. The ZSCC implementation occupies $28.80\mu m^2$, which is only 8% larger than using static CMOS gates due to the use of NMOS in PUN and the removal of input inversion buffers between cells. Moreover, ZSCC embeds a PMOS cross-coupled pair inside

each cell as a latch element. Flip-flops are therefore eliminated in ZSCC pipeline stages, introducing further reduction of area and power compared with static CMOS design.

4.5.3 Back-End Design

To distribute the four clock phases with minimal skew and enable APR flow, we developed a clock distribution scheme using four interleaved clock meshes. Top-level metal 9 and 8 are used to implement the meshes and distribute power-clock signals within the core area. An example of the clock mesh structure can be found in Figure 4.9, which is the floorplan used to implement the design described in Sections 4.6 through 4.8. Each top-level mesh is connected directly to metal-3 strips to deliver power-clock signals to local cell area, as shown in Figure 4.10. An automatic script is developed to run metal-3 power-clock strips along the cell rows, so the power-clock signals alternate. During placement, each cell is automatically placed in the row immediately above or below the metal-3 stripe of the corresponding power-clock phase. The proposed scheme not only allows large-scale APR with ZSCC gates but also minimizes the local clock interconnect and yield a competitive placement density.

The hearing-aid chip described in Sections 4.6 through 4.8 has been designed using the proposed back-end implementation methodology, achieving a placement density of 81.4%. The pitch of the top-level clock distribution network is 14.4μm for metal-8 vertical routing and 28μm for metal-9 horizontal routing. Wire width of the top-level clock distribution network is 2μm for metal-8 vertical routing and 4μm for metal-9 horizontal routing. A width of 0.1μm is used for local metal-3 PC stripes distributed on the cell rows.

As illustrated in Section 4.4, to generate the four power-clock signals, two off-chip inductors and corresponding on-chip H-bridge drivers are used to resonate the parasitic capacitance of the four-phase clock distribution network and the ZSCC gates.

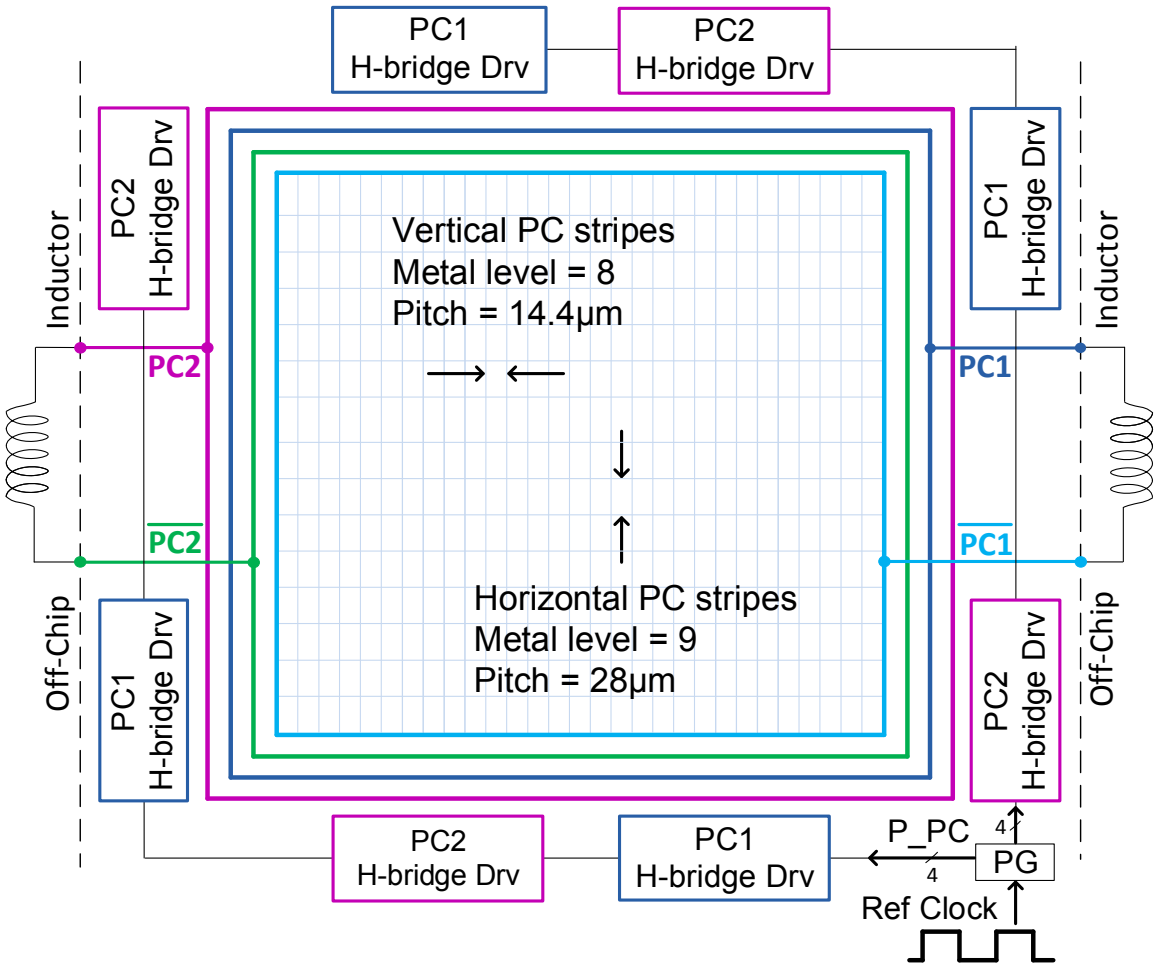


Figure 4.9: Top-level floorplan showing four-phase clock mesh.

The H-bridge drivers are placed around the periphery of the clock mesh network. As shown in Figure 4.9, 4 pairs of on-chip H-bridge drivers are used, placed on the four sides of the mesh network. Each driver is designed with programmable widths to support different clock speeds and enable tuning for maximum energy efficiency. The H-bridges are driven by four pulses in quadrature generated by a finite-state machine PG running off a reference clock. To allow for energy efficiency tuning, PG can be programmed to generate pulses with different duty cycles. Symmetric distribution of the pulses is intended to reduce skew at the H-bridges. For the prototype test-chip described in Sections 4.6 through 4.8, we designed the H-bridge drivers with programmable NMOS width ranging from 5.4μm to 37.8μm. A finite-state machine

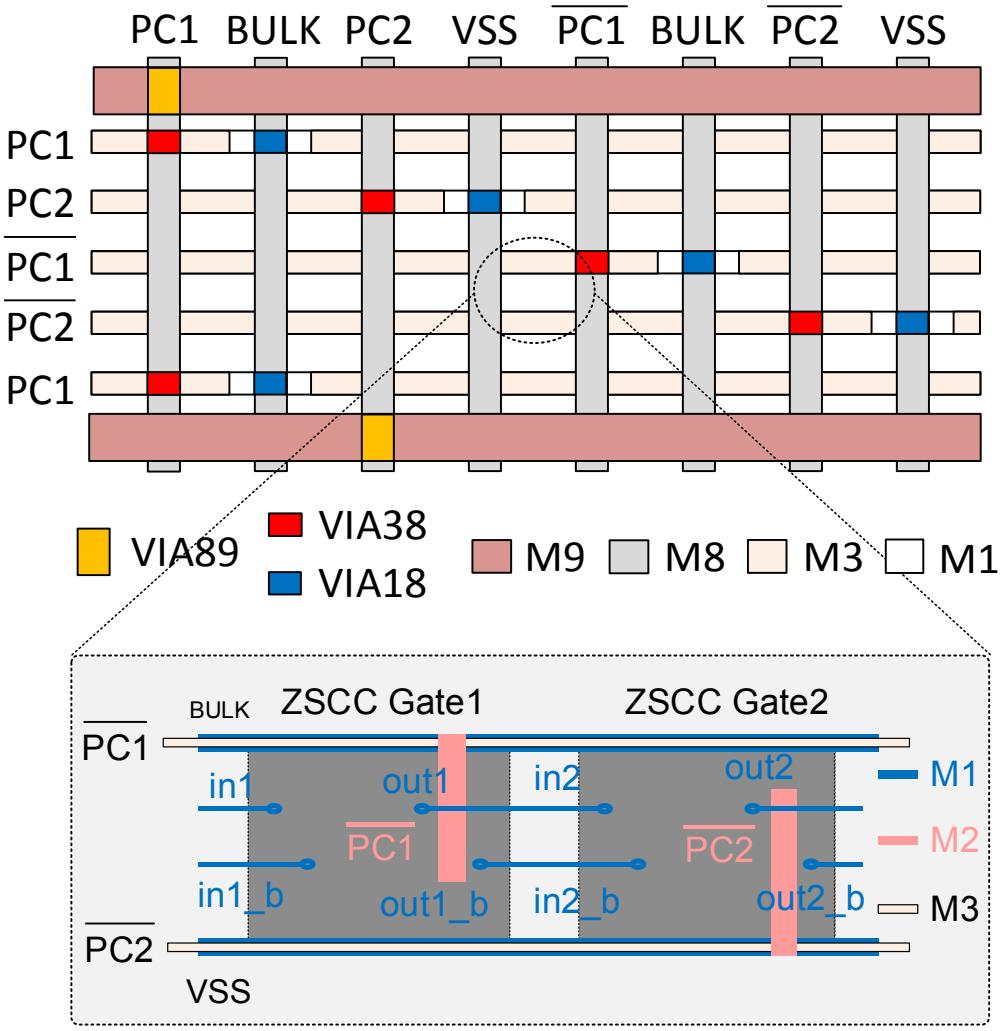


Figure 4.10: ZSCC cell placement.

inside the PG was implemented to generate pulses with programmable duty cycle ranging from 2.5% to 25%. Post-layout SPICE simulation was performed to verify the pulse skew at different H-bridge drivers. Our simulation results show that the symmetric distribution of the pulses at the H-bridges reduces worst-case skew to 135.6ps, with the drivers and peripheral circuitry operating at 0.6V supply voltage.

4.6 Hearing-Aid Design

To demonstrate the energy saving potential of the ZSCC logic family, we designed a hearing-aid test-chip using ZSCC logic. Numerous studies for binaural multimicro-

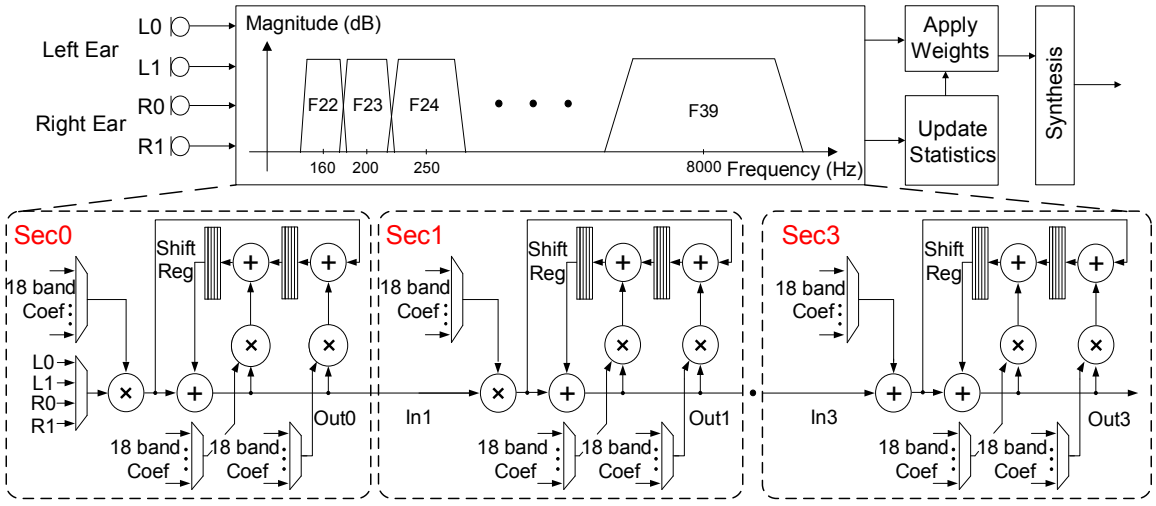


Figure 4.11: ANSI S1.11 1/3-octave binaural hearing aid: Bands F22 to F39, datapath block diagram.

phone hearing-aids systems have been proposed recently [66–69]. Binaural hearing aids assume a wireless link between the two ears to exchange information and significantly suppress noise interference. Due to the binaural setup, the hearing aids also preserve interaural time cues used for localizing sounds. However, these features come at the cost of significantly higher computational and power requirements than monophonic single-microphone systems. With the audio sampling rate at 24KHz range, the chip clock rate can be around the 1MHz mark and still meet real-time requirements. Therefore, these high performance hearing-aid systems are ideal candidates for low-power implementation through ZSCC design.

The proposed hearing-aid system and datapath design are shown in Figure 4.11. The binaural chip time-multiplexes 4 inputs (two inputs per ear) onto a datapath consisting of 4 second-order biquad sections that are cascaded to implement 18 ANSI S1.11 1/3-octave frequency bands F22 to F39. Inside the datapath, each section uses local shift registers to record the most recent audio cycle states to avoid pipeline stalls and run at low-speed.

To illustrate how the filter bank is designed with ZSCC logic, Figure 4.12 shows the gate-level block diagram of a MAC unit inside each biquad section. The MAC unit is

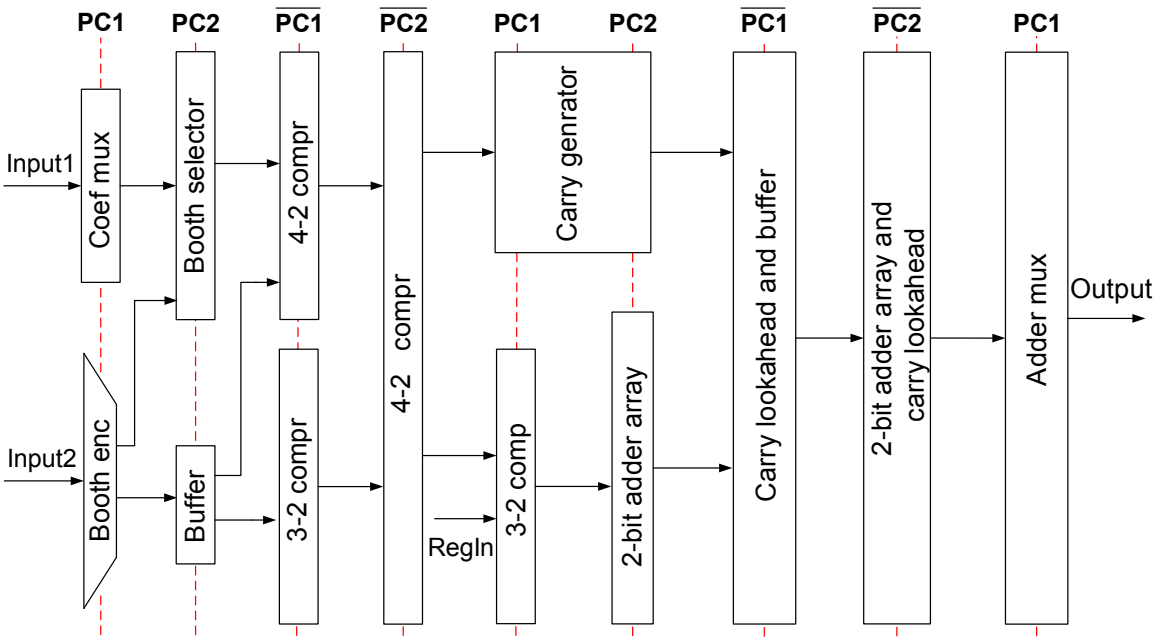


Figure 4.12: Detail of multiply-accumulate unit four-phase pipeline implementation.

implemented by merging a hybrid carry-lookahead/carry-select adder with a Booth-encoded multiplier and takes 9 phases or 2.25 cycles to complete one computation. The benefit of using four-phase clocking is that it not only helps to achieve zero-short-circuit-current operation but also reduces system latency, enabling operation at a lower clock frequency and higher energy-efficiency through charge recovery.

4.7 Chip-On-Board Design and Experimental Setup

To minimize the parasitic resistance and capacitance between the core and off-chip inductive elements, we have created a chip-on-board (COB) design for our test-chip. The COB serves as a custom package for the chip and also a daughter board in the system. A separate mother board has been designed for delivering power and scan signals to the chip.

Figure 4.13 shows our custom COB design. The bare-die hearing aid is mounted directly on the center of the board. The two $4.95 \times 3.81 mm^2$ surface-mount inductors used to resonate the chip are located on the left and right hand side of the chip.

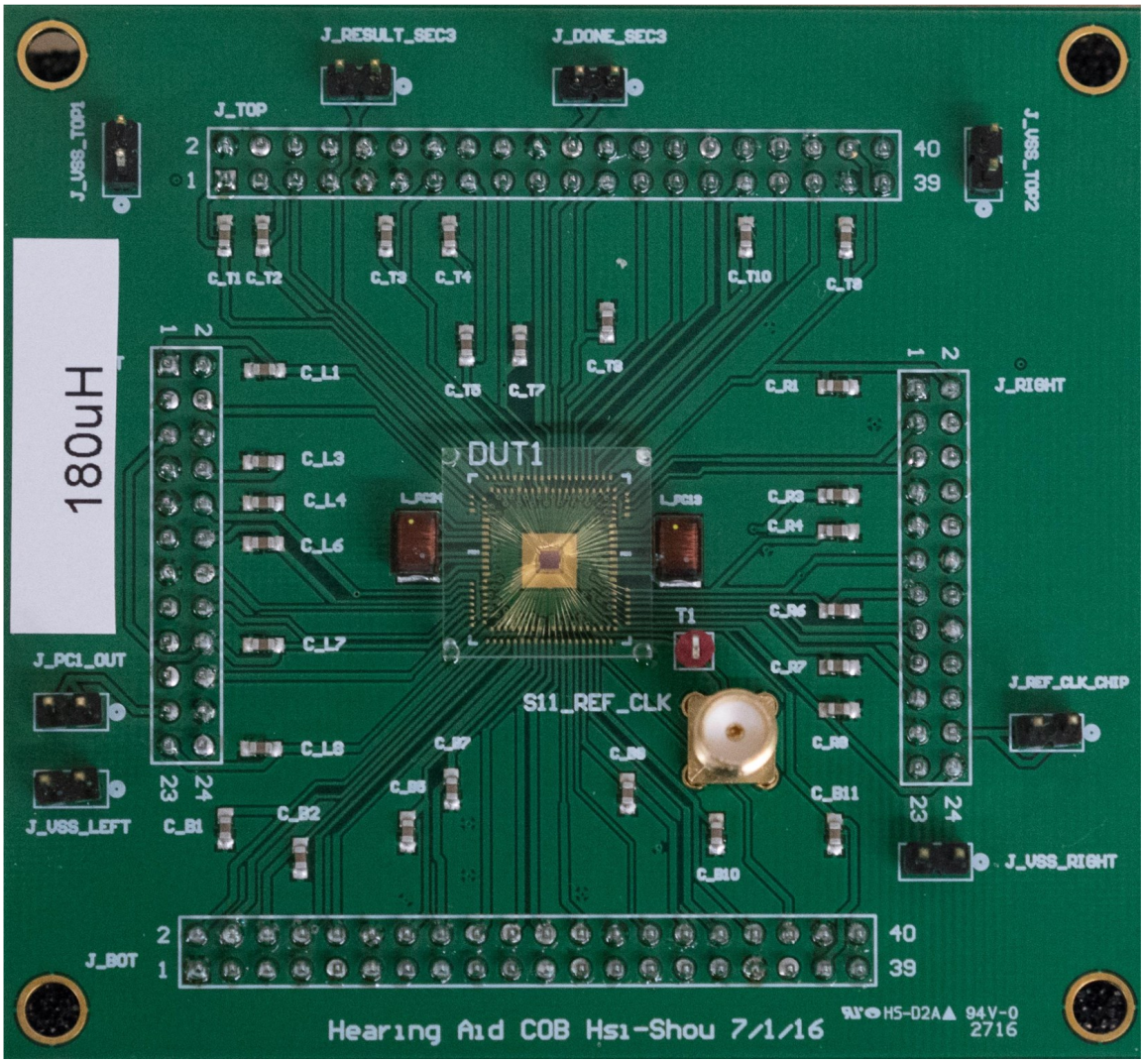


Figure 4.13: Custom chip-on-board design. The bare die is mounted directly on the center of the daughter board.

A SubMiniature version A (SMA) connector is installed on the board for feeding a reference clock to the chip. The routing between power-clock on-chip pins and off-chip inductors is kept as short as possible to reduce parasitics. Any extra parasitic resistance or capacitance introduced by conventional packaging is also reduced as parasitic-sensitive pins are connected directly to the components on the board.

Figure 4.14 shows the experimental evaluation setup. The hearing-aid COB is connected to the testing motherboard and is controlled and configured by a 96-channel USB digital I/O device connected to the motherboard. Level converters using Schmitt

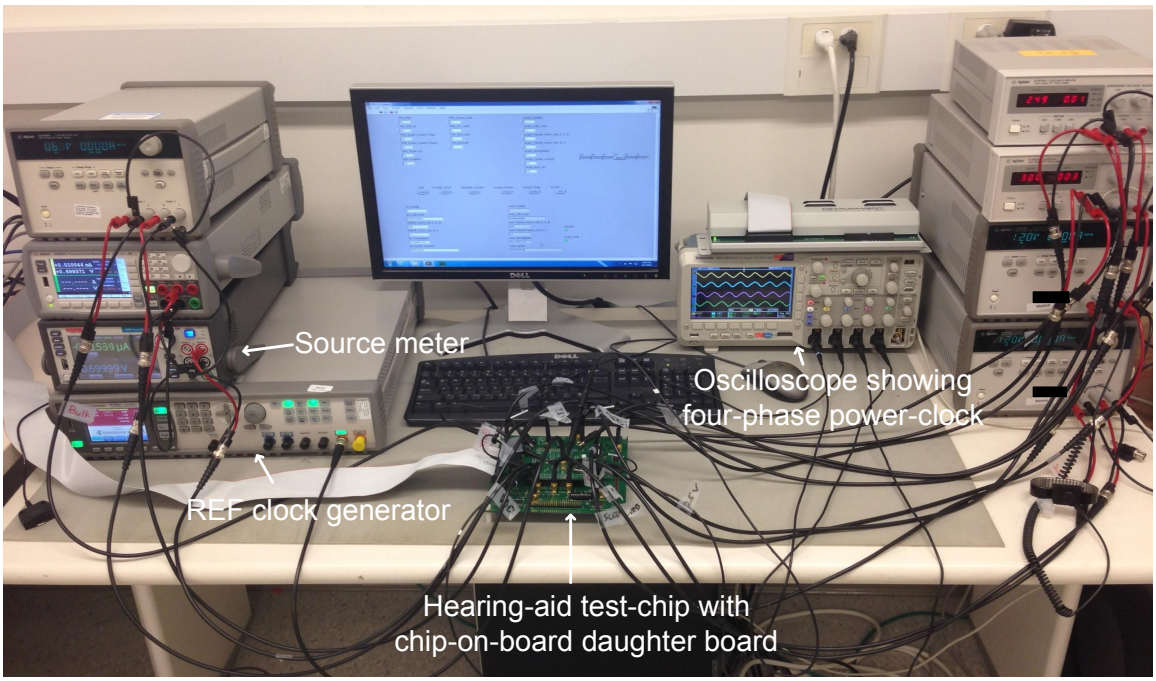


Figure 4.14: Experimental evaluation setup. Reference clock is generated with either an external pulse generator or an on-chip ring oscillator. Signals are transmitted to and received from the test-chip via the USB digital I/O device.

triggers are located on the motherboard to convert signals between USB digital I/O device voltage and chip I/O voltage and to prevent glitches during signal transfer. A reference clock for the on-chip four-phase pulse generator is generated using the external pulse generator shown in Figure 4.14 or the on-chip ring oscillator. An oscilloscope is also used during testing for verifying the operation of the chip.

4.8 Experimental Results

The hearing-aid test-chip was fabricated in a 65nm CMOS process. A die microphotograph is shown in Figure 4.15. To reduce the parasitic resistance of wire-bonding and I/O pads, the two off-chip inductors were connected to the die with three pads per clock phase. A built-in-self-test (BIST) circuit was implemented with static CMOS logic to verify functionality.

The test-chip has been tested at various clock frequencies. Figure 4.16 shows mea-

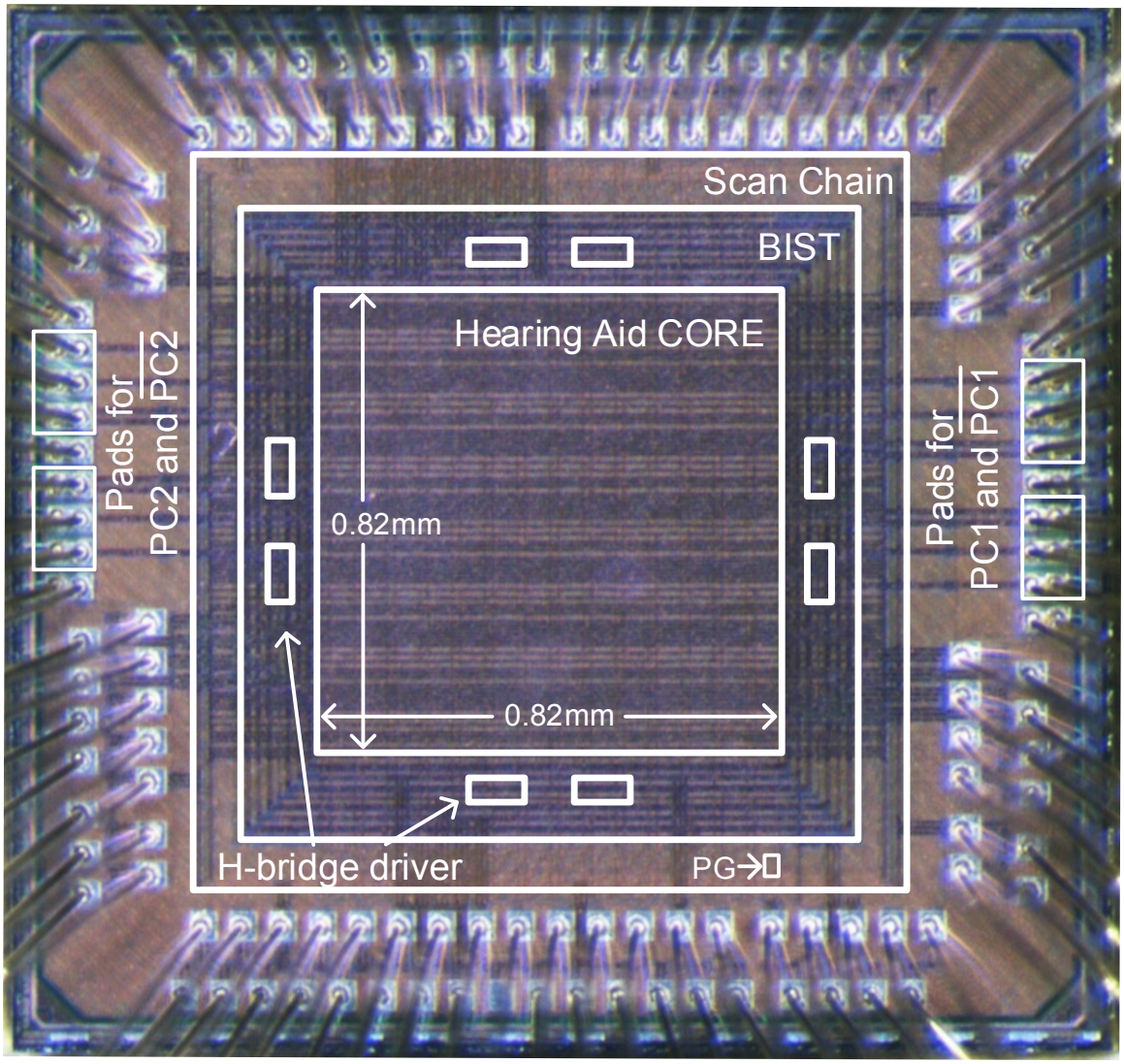


Figure 4.15: Microphotograph of the ZSCC hearing-aid test-chip in 65nm CMOS.

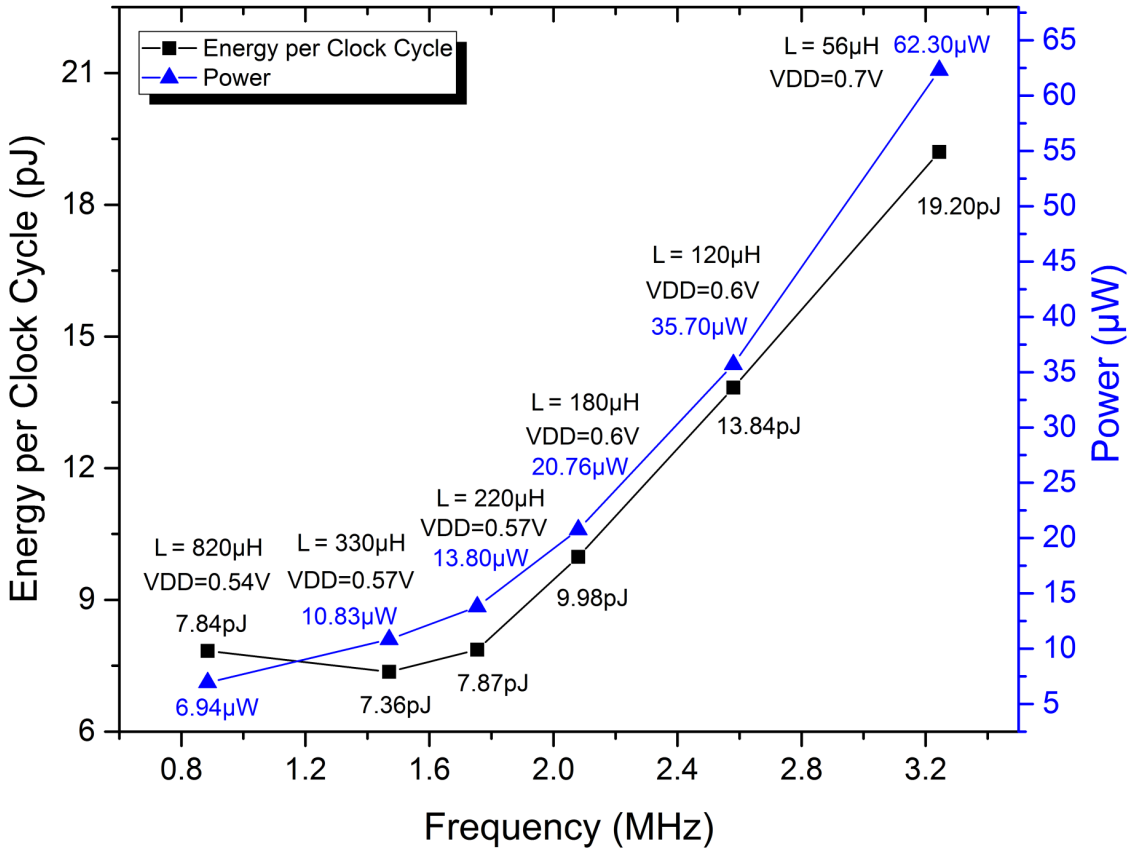


Figure 4.16: Measured energy per clock cycle and power versus frequency.

sured energy per clock cycle and power vs. operating frequency for different inductance values. The graph shows that energy consumption scales with frequency, as expected from charge-recovery design. At each frequency, the supply to the power-clock H-bridge driver VDD and its pulse output duty cycle are tuned to yield minimum total energy. The ANSI S1.11 standard is met at 1.75MHz with 7.87pJ per cycle. Minimum energy consumption is 7.36pJ per cycle at 1.47MHz with supply voltage VDD = 0.57V and 5% pulse duty cycle. Unlike previous published charge-recovery logic, no sharp increase in energy consumption from short-circuit currents is observed at lower clock frequencies. The energy per cycle gradually increases below 1MHz due to leakage current. Figure 4.17 shows the measured four-phase clock waveform when the test-chip operates at 1.68MHz with 0.64V voltage swing.

Table 4.2 compares our ZSCC-based test-chip with the state-of-the art hearing-aid

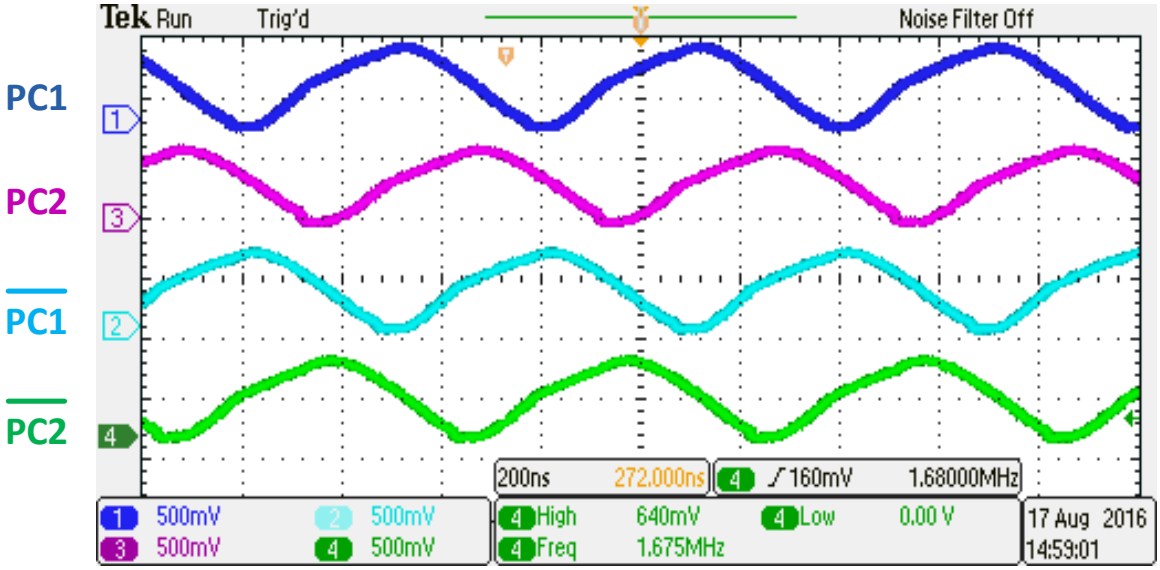


Figure 4.17: Measured four-phase clock waveform.

chip in [14] and a silicon cochlea for Internet-of-Everything (loE) acoustic detection tasks in [15]. Compared with the 40nm single-input chip in [14], our 65nm 4-input chip achieves 9.7× lower power per input/band. Compared with the chip for audio sensing applications in [15], which greatly benefits from efficient analog approaches, this charge-recovery low-power digital filter provides superior energy efficiency and programmable filter coefficients that support the adjustments necessary with hearing aids.

	This Work	VLSI'13 [14]	ISSCC'16 [15]
Technology	65nm	40nm	180nm
Application	Hearing Aid	Hearing Aid	Silicon Cochlea for IoE
Processing	Digital Filter Bank	Digital Filter Bank	Analog Filter Bank + Event coding
Standard	ANSI S1.11 1/3 Octave	ANSI S1.11 1/3 Octave	N/A
Binaural	Yes, 2x2 inputs	No, 1 input	Yes, 2x1 inputs
# Bands	18x4	18	64x2
Technique	ZSCC	TBLB[1]	Analog Filtering + ADM[2]
Programmability	Digital coefficients	Digital coefficients	Bias current
Area per band ($10^{-3}mm^2$)	9.34	5.27	262
Clock Frequency (MHz) / Sampling rate (kHz)	1.75 / 24	12 / 24	N/A
Power Supply (V)	0.57	0.36	0.5
Power (µW)	13.8	33.3	55
Power per input (µW)	3.45	33.30	27.50
Power per band (µW)	0.19	1.85	0.43

Area and power are not normalized for technology scaling.
[1] Time borrow and local boost (TBLB)
[2] Asynchronous delta modulation (ADM)

Table 4.2: Chip summary and comparison with state-of-the-art designs.

4.9 ZSCC Neural Network Implementation

To demonstrate the effectiveness of the proposed ZSCC logic family for improving the energy efficiency of the heterogeneous neural network processor presented in Chapter III, we have implemented the low-power front-end of the proposed heterogeneous processor shown in Section 3.3 with a ZSCC-based 3×8 SIMD MAC array in a 65nm CMOS process. 8×8 Booth multipliers are designed and used in the SIMD computation pipeline. Each multiplier takes 4 phases, or 1 cycle, to generate sum and carry vectors. Hybrid carry-lookahead/carry-select adders accumulate the outputs of the multipliers. Each hybrid adder also takes 4 phases, or 1 cycle, to generate a result.

To compare the efficiency of the ZSCC-based design with a conventional static CMOS implementation, we have performed SPICE simulations of the proposed ZSCC SIMD MAC array. Using the 65nm BSIM model and assuming each gate has 15fF fanout loading, the MAC array is tested at different frequencies with 0.7V power-clock voltage swing. Figure 4.18 shows the energy per cycle and power versus operating frequency of the design. As shown in the graph, the energy per cycle decreases with frequency of operation, as expected by the charge-recovery nature of the logic.

We also synthesized a standard-cell static CMOS version of the same 2-cycle SIMD MAC array architecture in the same 65nm technology. As timing requirements in the 7MHz to 11MHz range can be met in a relatively straightforward manner, we used only high V_{th} standard cells during synthesis to reduce leakage current. Consistent with the performance requirements in Section 3.3, both designs are compared at 10MHz clock rate with a 0.7V supply using SPICE simulation. The conventional CMOS version dissipates 1.4mW, while our ZSCC implementation dissipates 81.4μW, yielding 17× higher power efficiency. A more accurate simulation number could be measured after APR and the extraction of parasitic loading from routing. Still, the order-of-magnitude power improvement observed in pre-layout evaluation suggests that the ZSCC logic family is a highly promising option for further improving the

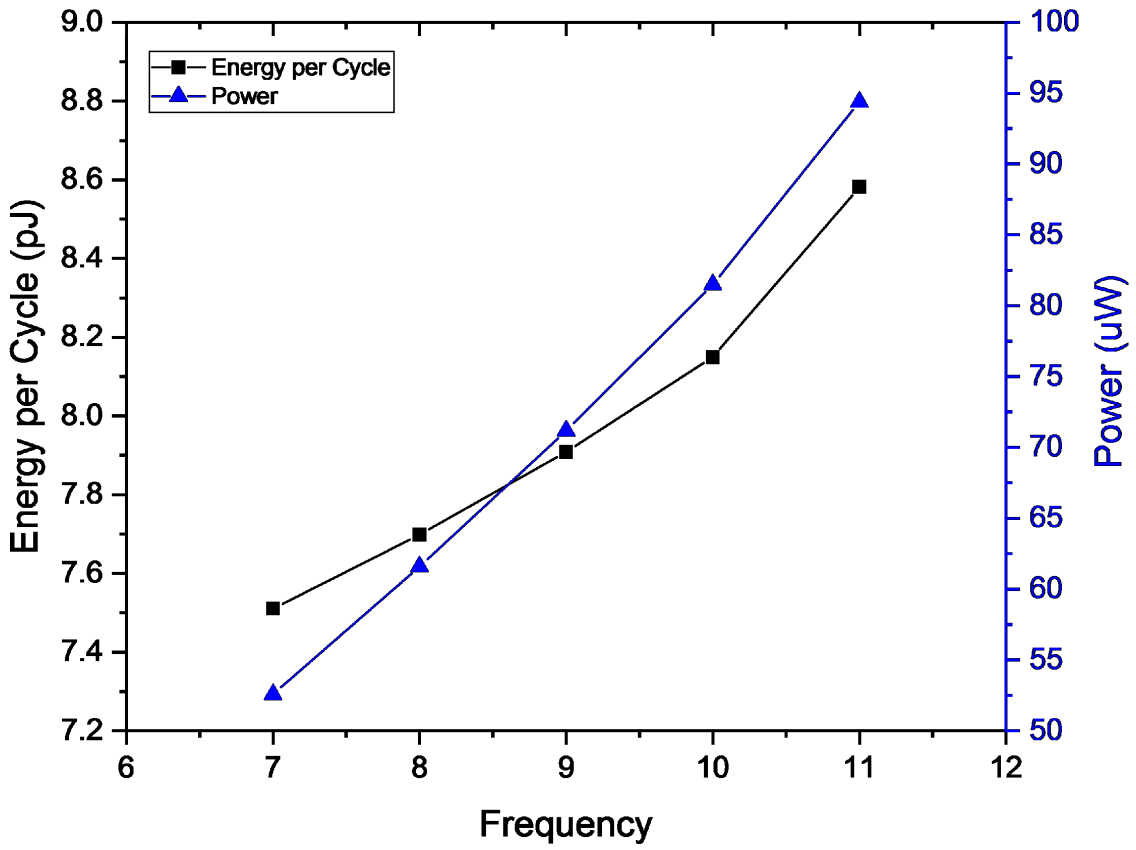


Figure 4.18: Simulated energy per clock cycle and power versus frequency of the ZSCC SIMD MAC array.

energy efficiency of the heterogeneous neural network accelerator architecture.

4.10 Conclusion

In this chapter, we presented a novel charge-recovery logic family, called ZSCC, that is suitable for relatively low frequency low-power design and always-on applications. ZSCC has been designed to dramatically reduce short-circuit currents at low operating frequencies, leveraging the benefits of charge-recovery principle to achieve order-of-magnitude power savings. In addition to ZSCC, we presented an associated standard-cell-like semi-automated design flow for large-scale ZSCC VLSI implementation.

To demonstrate the effectiveness of the proposed ZSCC logic family, we have

designed a binaural hearing-aid test-chip complying with the ANSI S1.11 standard using an automated design flow and a ZSCC cell library of 64 different functions, each with 2 to 4 different drive strengths, implemented in a 65nm CMOS process. A four-phase clock mesh is used for power-clock distribution and supported by the APR flow. Our test results show that the ZSCC-based hearing-aid test-chip outperforms state-of-the-art low-voltage digital designs by 9.7×.

To assess the effectiveness of ZSCC in improving the energy efficiency of the heterogeneous neural network processor described in Chapter III, we have implemented the processor's front-end using ZSCC logic. In SPICE simulations of netlists, the ZSCC design achieves 17× higher energy efficiency than its static CMOS counterpart. Therefore, the proposed ZSCC logic family has the potential to increase the overall energy efficiency of the proposed architecture by a significant factor.

CHAPTER V

Conclusion and Future Research Directions

Machine learning technology is now used in numerous aspects of daily life, and energy-efficient neural network systems supporting ML applications have become crucial for both the server and the device end. This dissertation explores architectures and circuits for designing energy-efficient neural network accelerators. In this chapter, we summarize these complementary approaches, presented in Chapters III and IV. We also include a discussion of promising future research directions.

At the architecture level, this thesis proposes a heterogeneous architecture for energy-efficient neural networks. The proposed architecture relies on a low-power always-on front-end and a high-performance selectively-enabled back-end to support conditionally-executing DNNs with high performance and low power consumption through voltage and frequency scaling. A key feature of this architecture is a dynamic execution unit that can dynamically adjust computational resources at runtime to meet performance targets with increased energy efficiency. To address energy inefficiencies resulting from irregular memory access patterns introduced by conditional classifiers, a low-power data memory architecture is introduced to support fast data sharing and maximize energy efficiency through burst-mode access and fine-grain gating.

Compared with other low-power neural network accelerator designs, the proposed

heterogeneous architecture achieves higher energy efficiency using an orthogonal approach. Specifically, it is compatible with previous approaches for neural network hardware design, such as custom analog and mixed-signal datapath implementation, configurable hardware combining bit-quantization and voltage-frequency scaling, and dedicated datapath for sparsity optimization, yielding additional power savings above and beyond what is achievable by earlier approaches.

To demonstrate the energy-efficiency of the proposed architecture, a test-chip has been fabricated in a commercial 40nm CMOS technology. Various sets of weights that yield different decision outputs are trained to trade off energy consumption for recognition accuracy. In measurement results, the chip achieves 5.3× lower energy with 0.23mW average power and 1.4% accuracy loss on the LFW dataset. By conditional execution in an advanced neural network topology, energy savings of 2.5× at 91.3% accuracy and 4.9× at 86% accuracy can be achieved on the CIFAR-10 dataset. By using dynamic execution, the chip achieves a competitive energy scalability of 4.7-28.6 TOPS/W which allows adjustment to adapt to various operating points under power-constrained conditions.

To further improve the energy efficiency of the proposed heterogeneous architecture, we introduce a charge-recovery logic family that significantly reduces short-circuit current and achieves order of magnitude energy savings at relatively low clock frequencies. Unlike conventional CMOS logic which relies on reducing voltage supply to lower energy consumption, this zero-short-circuit current (ZSCC) logic fully benefits from charge-recovery operation to achieve ultra-energy-efficient computation. In conjunction with the ZSCC circuit topology, an automatic design flow is proposed to enable large-scale ZSCC-based VLSI design.

To demonstrate the energy efficiency and applicability of ZSCC, we have used it to design a hearing-aid test-chip that complies with the ANSI S1.11 standard. Fabricated in a 65nm bulk silicon process, the test-chip shows energy scalability with

operating frequency, consistent with the theoretical model. It achieves a minimum energy consumption of 7.36pJ per cycle at 1.47MHz with supply voltage of 0.57V. Our measurement results show no sharp increase in energy consumption from short-circuit currents at lower clock frequencies. Compared with a 40nm monophonic single-input chip that represents the published state of the art, the 65nm test-chip processes 4 input streams at 1.75MHz, achieving 9.7× lower power per input.

To evaluate the effectiveness of ZSCC for neural network accelerator design, we used it to design the front-end of the proposed heterogeneous deep-learning processor. In SPICE simulations, the ZSCC-based front-end operates with 17× higher energy efficiency at 10MHz compared with a conventional static CMOS implementation of the same architecture.

Our dissertation research points to several interesting questions for further research. In our test-chip implementation of the proposed heterogeneous architecture, supply voltage and clock frequency are fixed at runtime. An interesting topic for further investigation would be the introduction and experimental evaluation of dynamic voltage-frequency operation into the design to further improve energy efficiency. To that end, a workload scheduler and power management unit (PMU) could be designed to dynamically adjust the front-end and back-end voltage supplies depending on the workload or status of an executed path. Whenever the network is about to branch to a larger network, the scheduler would foresee a potential workload increase, instructing the PMU to set supply to a higher voltage, and adjusting processor speed to a higher clock frequency to reduce latency. Conversely, should the network branch to a small compute subnet, the scheduler would instruct the PMU to reduce voltage supply and lower the processor clock frequency to save power. Such a dynamic power management scheme that tunes supply voltage and clock frequency according to conditional execution could lead to significant additional increases in the energy efficiency of the proposed heterogeneous architecture.

At the circuit level, an interesting research problem is the design of power-clock generators for ZSCC that do not require two inductors to generate the four-phase power-clock signals. In a commercial system, discrete inductors introduce additional costs and challenges with area-constrained PCB board implementations. Therefore, reducing the number of required inductors from two to one would have significant practical implications. One possible solution is to design a specialized controller to connect and disconnect the inductor with proper timing. By clamping the voltage to high and low for a complementary pair of power-clocks while resonating the other pair of 90-degrees out-of-phase complementary power-clocks through the inductor in an alternating manner, it may be possible to generate a four-phase power-clock using a single off-chip inductor. However, the additional resistance introduced by the switching MOS and reconnecting operations could reduce overall energy efficiency greatly. Using a switched-capacitor clock driver for four-phase clock generation could be an alternative solution to this problem. A switched-capacitor approach has been explored in [70] for single-phase adiabatic clocking. However, this concept has not been explored in the context of four-phase clock generation.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[2] A. Wang and A. Chandrakasan, "A 180-mV subthreshold FFT processor using a minimum energy design methodology," *IEEE Journal of Solid-State Circuits*, pp. 310–319, Jan 2005.

[3] R. G. Dreslinski and B. Zhai and T. Mudge and D. Blaauw and D. Sylvester, "An energy efficient parallel architecture using near threshold operation," in *Proceedings of the International Conference on Parallel Architecture and Compilation Techniques (PACT)*, Sept 2007, pp. 175–188.

[4] M. Price, J. Glass and A. P. Chandrakasan, "14.4 A scalable speech recognizer with deep-neural-network acoustic models and voice-activated power gating," in *IEEE International Solid-State Circuits Conference (ISSCC)*, Feb 2017, pp. 244–245.

[5] B. Moons, R. Uytterhoeven, W. Dehaene and M. Verhelst, "14.5 Envision: A 0.26-to-10TOPS/W subword-parallel dynamic-voltage-accuracy-frequency-scalable Convolutional Neural Network processor in 28nm FDSOI," in *IEEE International Solid-State Circuits Conference (ISSCC)*, Feb 2017, pp. 246–247.

[6] V. Sze, Y. H. Chen, T. J. Yang and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proceedings of the IEEE*, pp. 2295–2329, Dec 2017.

[7] Y. H. Chen, J. Emer and V. Sze, "Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks," in *Proceedings of the Annual International Symposium on Computer Architecture (ISCA)*, June 2016, pp. 367–379.

[8] N. P. Jouppi *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *Proceedings of the Annual International Symposium on Computer Architecture (ISCA)*, June 2017, pp. 1–12.

[9] Y. Kang, J. Hauswald, C. Gao, A. Rovinski, T. Mudge, J. Mars and L. Tang, "Neurosurgeon: Collaborative intelligence between the cloud and mobile edge,"

in *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, April 2017, pp. 615–629.

[10] C. Chen, H. Ding, H. Peng, H. Zhu, R. Ma, P. Zhang, X. Yan, Y. Wang, M. Wang, H. Min and R. C. -. Shi, "Ocean: An on-chip incremental-learning enhanced processor with gated recurrent neural network accelerators," in *IEEE European Solid-State Circuits Conference (ESSCIRC)*, Sept 2017, pp. 259–262.

[11] J. Lee, C. Kim, S. Kang, D. Shin, S. Kim and H. Yoo, "UNPU: A 50.6TOPS/W unified deep neural network accelerator with 1b-to-16b fully-variable weight bit-precision," in *IEEE International Solid-State Circuits Conference (ISSCC)*, Feb 2018, pp. 218–220.

[12] D. Bankman, L. Yang, B. Moons, M. Verhelst and B. Murmann, "An always-on 3.8μJ/86% CIFAR-10 mixed-signal binary CNN processor with all memory on chip in 28nm CMOS," in *IEEE International Solid-State Circuits Conference (ISSCC)*, Feb 2018, pp. 222–224.

[13] D. Jeon, M. Seok, C. Chakrabarti, D. Blaauw and D. Sylvester, "A Super-Pipelined Energy Efficient Subthreshold 240 MS/s FFT Core in 65 nm CMOS," *IEEE Journal of Solid-State Circuits*, pp. 23–34, Jan 2012.

[14] J. Wang, K. Chang, T. Lin, R. W. Prasojo and C. Yeh, "A 0.36V, 33.3μW 18-band ANSI S1.11 1/3-octave filter bank for digital hearing aids in 40nm CMOS," in *IEEE Symposium on VLSI Circuits*, June 2013, pp. C254–C255.

[15] M. Yang, C. Chien, T. Delbruck and S. Liu, "A 0.5V 55μW 64$\times$2-channel binaural silicon cochlea for event-driven stereo-audio sensing," in *IEEE International Solid-State Circuits Conference (ISSCC)*, Jan 2016, pp. 388–389.

[16] T. N. Sainath, A. Mohamed, B. Kingsbury and B. Ramabhadran, "Deep convolutional neural networks for LVCSR," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2013, pp. 8614–8618.

[17] S. Hanson, B. Zhai, M. Seok, B. Cline, K. Zhou, M. Singhal, M. Minuth, J. Olson, L. Nazhandali, T. Austin, D. Sylvester and D. Blaauw, "Performance and Variability Optimization Strategies in a Sub-200mV, 3.5pJ/inst, 11nW Subthreshold Processor," in *IEEE Symposium on VLSI Circuits*, June 2007, pp. 152–153.

[18] R. G. Dreslinski, M. Wieckowski, D. Blaauw and D. Sylvester and T. Mudge, "Near-threshold computing: Reclaiming moore's law through energy efficient integrated circuits," *Proceedings of the IEEE*, pp. 253–266, Feb 2010.

[19] S. Jain, S. Khare, S. Yada, V. Ambili, P. Salihundam, S. Ramani, S. Muthukumar, M. Srinivasan, A. Kumar, S. K. Gb, R. Ramanarayanan, V. Erraguntla, J. Howard, S. Vangal, S. Dighe, G. Ruhl, P. Aseron, H. Wilson, N. Borkar, V. De and S. Borkar, "A 280mV-to-1.2V wide-operating-range IA-32 processor in

32nm CMOS," in *IEEE International Solid-State Circuits Conference (ISSCC)*, Feb 2012, pp. 66–68.

[20] Y. Kim, W. Jung, I. Lee, Q. Dong, M. Henry, D. Sylvester and D. Blaauw, "27.8 A static contention-free single-phase-clocked 24T flip-flop in 45nm for low-power applications," in *IEEE International Solid-State Circuits Conference (ISSCC)*, Feb 2014, pp. 466–467.

[21] A. P. Chandrakasan, S. Sheng and R. W. Brodersen, "Low-power CMOS digital design," *IEEE Journal of Solid-State Circuits*, pp. 473–484, Apr 1992.

[22] D. Menasce and V. Almeida, "Cost-performance analysis of heterogeneity in supercomputer architectures," in *Proceedings of the ACM/IEEE Conference on Supercomputing*, Nov 1990, pp. 169–177.

[23] R. Kumar, K. I. Farkas, N. P. Jouppi, P. Ranganathan and D. M. Tullsen, "Single-ISA heterogeneous multi-core architectures: the potential for processor power reduction," in *Proceedings of the Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, Dec 2003, pp. 81–92.

[24] Big.LITTLE Processing with ARM Cortex-A15 & Cortex-A7. [Online]. Available: http://goo.gl/7mgbL

[25] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Proceedings of the 25th International Conference on Neural Information Processing Systems*, Dec 2012, pp. 1097–1105.

[26] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations (ICLR)*, 2015.

[27] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke and A. Rabinovich, "Going deeper with convolutions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015, pp. 1–9.

[28] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 770–778.

[29] S. Park, K. Bong, D. Shin, J. Lee, S. Choi and H. Yoo, "4.6 A 1.93TOPS/W scalable deep learning/inference processor with tetra-parallel MIMD architecture for big-data applications," in *IEEE International Solid-State Circuits Conference (ISSCC)*, Feb 2015, pp. 1–3.

[30] L. Cavigelli, D. Gschwend, C. Mayer, S. Willi, B. Muheim, and L. Benini, "Origami: A convolutional network accelerator," in *Proceedings of the 25th Edition on Great Lakes Symposium on VLSI*, 2015, pp. 199–204.

[31] M. Sankaradas, V. Jakkula, S. Cadambi, S. Chakradhar, I. Durdanovic, E. Cosatto and H. P. Graf, "A Massively Parallel Coprocessor for Convolutional Neural Networks," in *2009 20th IEEE International Conference on Application-specific Systems, Architectures and Processors*, July 2009, pp. 53–60.

[32] Z. Du, R. Fasthuber, T. Chen, P. Ienne, L. Li, T. Luo, X. Feng, Y. Chen and O. Temam, "ShiDianNao: Shifting vision processing closer to the sensor," in *Proceedings of the Annual International Symposium on Computer Architecture (ISCA)*, June 2015, pp. 92–104.

[33] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao and J. Cong, "Optimizing FPGA-based Accelerator Design for Deep Convolutional Neural Networks," in *Proceedings of the ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2015, pp. 161–170.

[34] S. Teerapittayanon, B. McDanel and H. T. Kung, "BranchyNet: Fast inference via early exiting from deep neural networks," in *International Conference on Pattern Recognition (ICPR)*, Dec 2016, pp. 2464–2469.

[35] L. Liu and J. Deng, "Dynamic Deep Neural Networks: Optimizing Accuracy-Efficiency Trade-offs by Selective Execution," in *AAAI Conference on Artificial Intelligence*, 2018.

[36] H. S. Wu, Z. Zhang and M. C. Papaefthymiou, "A 0.23mW Heterogeneous Deep-Learning Processor Supporting Dynamic Execution of Conditional Neural Networks," in *IEEE European Solid-State Circuits Conference (ESSCIRC)*, Sept 2018, to be published.

[37] S. Gupta, A. Agrawal, K. Gopalakrishnan and P. Narayanan, "Deep learning with limited numerical precision," in *Proceedings of the International Conference on International Conference on Machine Learning*, 2015, pp. 1737–1746.

[38] A. Parashar, M. Rhu, A. Mukkara, A. Puglielli, R. Venkatesan, B. Khailany, J. Emer, S. W. Keckler and W. J. Dally, "SCNN: An accelerator for compressed-sparse convolutional neural networks," in *Proceedings of the Annual International Symposium on Computer Architecture (ISCA)*, June 2017, pp. 27–40.

[39] E. Bengio, P.-L. Bacon, J. Pineau and D. Precup, "Conditional Computation in Neural Networks for faster models," *Workshop Track, ICLR*, 2016.

[40] Y. Bengio, N. Léonard and A. C. Courville, "Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation," *arXiv preprint arXiv:1308.3432*, 2013.

[41] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. V. Le, G. E. Hinton and J. Dean, "Outrageously large neural networks: The sparsely-gated mixture-of-experts layer," in *International Conference on Learning Representations (ICLR)*, 2017.

[42] L. Denoyer and P. Gallinari, "Deep sequential neural network," *arXiv preprint arXiv:1308.3432*, 2014.

[43] K. Bong, S. Choi, C. Kim, S. Kang, Y. Kim and H. Yoo, "14.6 A 0.62mW ultra-low-power convolutional-neural-network face-recognition processor and a CIS integrated with always-on haar-like face detector," in *IEEE International Solid-State Circuits Conference (ISSCC)*, Feb 2017, pp. 248–249.

[44] H. S. Wu, Z. Zhang and M. C. Papaefthymiou, "20.7 A 13.8μW binaural dual-microphone digital ANSI S1.11 filter bank for hearing aids with zero-short-circuit-current logic in 65nm CMOS," in *IEEE International Solid-State Circuits Conference (ISSCC)*, Feb 2017, pp. 348–349.

[45] W.C. Athas, L.J. Svensson, J.G. Koller, N. Tzartzanis and E. Y.-C. Chou, "Low-power digital systems based on adiabatic-switching principles," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 2, no. 4, pp. 398–407, Dec 1994.

[46] A.G. Dickinson and J.S. Denker, "Adiabatic dynamic logic," *IEEE Journal of Solid-State Circuits*, vol. 30, no. 3, pp. 311–315, Mar 1995.

[47] A. Kramer, J.S. Denker, S.C. Avery, A.G. Dickinson and T.R. Wik, "Adiabatic Computing with the 2n-2n2d Logic Family," in *IEEE Symposium on VLSI Circuits*, Jun 1994, pp. 25–26.

[48] S. Kim and M. C. Papaefthymiou, "Single-phase source-coupled adiabatic logic," in *Proceedings of the International Symposium on Low Power Electronics and Design (ISLPED)*, Aug 1999, pp. 97–99.

[49] S. Kim and M. Papaefthymiou, "True single-phase adiabatic circuitry," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 9, no. 1, pp. 52–63, Feb 2001.

[50] S. Kim, C.H. Ziesler and M.C. Papaefthymiou, "A true single-phase energy-recovery multiplier," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 11, no. 2, pp. 194–207, 2003.

[51] C. H. Ziesler, Joohee Kim, V. S. Sathe and M. C. Papaefthymiou, "A 225 MHz resonant clocked ASIC chip," in *Proceedings of the International Symposium on Low Power Electronics and Design (ISLPED)*, Aug 2003, pp. 48–53.

[52] C.H. Ziesler, J. Kim and M.C. Papaefthymiou, "Energy recovering ASIC design," in *Proceedings of the IEEE Computer Society Annual Symposium on VLSI*, Feb 2003, pp. 133–138.

[53] V. S. Sathe, J. Y. Chueh and M. C. Papaefthymiou, "Energy-Efficient GHz-Class Charge-Recovery Logic," *IEEE Journal of Solid-State Circuits*, vol. 42, no. 1, pp. 38–47, Jan 2007.

[54] W.C. Athas, N. Tzartzanis, L.J. Svensson and L. Peterson, "A low-power microprocessor based on resonant energy," *IEEE Journal of Solid-State Circuits*, vol. 32, no. 11, pp. 1693–1701, Nov 1997.

[55] V. Sathe, S. Arekapudi, C. Ouyang, M. Papaefthymiou, A. Ishii and S. Naffziger, "Resonant clock design for a power-efficient high-volume x86-64 microprocessor," in *IEEE International Solid-State Circuits Conference (ISSCC)*, Feb 2012, pp. 68–70.

[56] D. Shan, P. Restle, D. Malone, R. Groves, E. Lai, M. Koch, J. Hibbeler, Y. Kim, C. Vezyrtzis, J. Feder, D. Hogenmiller and T. Bucelot, "Resonant clock mega-mesh for the IBM $z13^{TM}$," in *IEEE Symposium on VLSI Circuits*, June 2015, pp. C322–C323.

[57] F. U. Rahman and V. S. Sathe, "19.6 voltage-scalable frequency-independent quasi-resonant clocking implementation of a 0.7-to-1.2V DVFS System," in *IEEE International Solid-State Circuits Conference (ISSCC)*, Jan 2016, pp. 334–335.

[58] W.-H. Ma, J. Kao, V. Sathe and M.C. Papaefthymiou, "A 187MHz subthreshold-supply robust FIR filter with charge-recovery logic," in *IEEE Symposium on VLSI Circuits*, June 2009, pp. 202–203.

[59] W.-H. Ma, J. Kao, V. Sathe and M.C. Papaefthymiou, "187MHz subthreshold-supply charge-recovery FIR," *IEEE Journal of Solid-State Circuits*, vol. 45, no. 4, pp. 793–803, Apr 2010.

[60] Y. Moon and D.-K. Jeong, "An efficient charge recovery logic circuit," *IEEE Journal of Solid-State Circuits*, pp. 514–522, Apr 1996.

[61] W. C. Athas, L. J. Svensson and N. Tzartzanis, "A resonant signal driver for two-phase, almost-non-overlapping clocks," in *1996 IEEE International Symposium on Circuits and Systems*, May 1996, pp. 129–132 vol.4.

[62] A. Mirzaei, M. E. Heidari, R. Bagheri, S. Chehrazi and A. A. Abidi, "The Quadrature LC Oscillator: A Complete Portrait Based on Injection Locking," *IEEE Journal of Solid-State Circuits*, pp. 1916–1932, Sept 2007.

[63] V. S. Sathe, J. C. Kao and M. C. Papaefthymiou, "Resonant-clock latch-based design," *IEEE Journal of Solid-State Circuits*, pp. 864–873, April 2008.

[64] T.-C. Ou, Z. Zhang and M.C. Papaefthymiou, "An 821MHz 7.9Gb/s 7.3pJ/b/iteration charge-recovery LDPC decoder," in *IEEE International Solid-State Circuits Conference (ISSCC)*, Feb 2014, pp. 462–463.

[65] K.M. Chu and D.L. Pulfrey, "Design procedures for differential cascode voltage switch circuits," *IEEE Journal of Solid-State Circuits*, vol. 21, no. 6, pp. 1082–1087, 1986.

[66] D. Marquardt, V. Hohmann and S. Doclo, "Perceptually motivated coherence preservation in multi-channel wiener filtering based noise reduction for binaural hearing aids," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2014, pp. 3660–3664.

[67] D. Marquardt, V. Hohmann and S. Doclo, "Binaural cue preservation for hearing aids using multi-channel wiener filter with instantaneous ITF preservation," in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, March 2012, pp. 21–24.

[68] B. Cornelis, S. Doclo, T. Van dan Bogaert, M. Moonen and J. Wouters, "Theoretical analysis of binaural multimicrophone noise reduction techniques," *IEEE Transactions on Audio, Speech, and Language Processing*, pp. 342–355, Feb 2010.

[69] J. I. Marin-Hurtado and D. V. Anderson, "Reduced-bandwidth and low-complexity multichannel wiener filter for binaural hearing aids," in *2011 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, Oct 2011, pp. 85–88.

[70] L. G. Salem and P. P. Mercier, "26.4 A 0.4-to-1V 1MHz-to-2GHz switched-capacitor adiabatic clock driver achieving 55.6% clock power reduction," in *IEEE International Solid-State Circuits Conference (ISSCC)*, Feb 2017, pp. 442–443.