LOW-POWER DISCRETE FOURIER TRANSFORM AND SOFT-DECISION VITERBI DECODER FOR OFDM RECEIVERS

A Dissertation
Presented to
The Academic Faculty

by

Sangwook Suh

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology
December 2011

LOW-POWER DISCRETE FOURIER TRANSFORM AND SOFT-DECISION VITERBI DECODER FOR OFDM RECEIVERS

Approved by:

Prof. John Barry, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Prof. Paul Hasler, Co-Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Prof. Steven McLaughlin
School of Electrical and Computer Engineering
Georgia Institute of Technology

Prof. David Anderson
School of Electrical and Computer Engineering
Georgia Institute of Technology

Prof. James Hamblen
School of Electrical and Computer Engineering
Georgia Institute of Technology

Prof. Michael Lacey
School of Mathematics
Georgia Institute of Technology

Date Approved: August 23, 2011

Any intelligent fool can make things bigger, more complex, and more violent. It takes a touch of genius -- and a lot of courage -- to move in the opposite direction.

-- Albert Einstein

To all my sweet family and my beloved wife
with the most gratitude that I feel stuck that I cannot show it enough

ACKNOWLEDGEMENTS

I would first like to express the most gratitude to my advisor, Prof. John Barry, for his encouragement and support during my time at Georgia Tech. He not only has brightened the way when I was totally lost in finding a lighthouse for my research, but also has continuously inspired me with the guidance for my life. I deeply appreciate every opportunity I had to speak with him and learn from him in person. I learned a lot from his rigorous thought process in defining problems, modeling systems, and solving the problems, which will affect me in one way or another throughout the rest of my research activities.

I am also indebted to my co-advisor, Prof. Paul Hasler, who has guided me with his invaluable insights on circuit implementation aspects. Being able to be involved in actual circuit implementation was a great advantage for me to maintain a close distance from practical issues. His advice and directions meant a lot to me whenever I was struggling with understanding the fundamental nature of problems during my research.

I am also thankful to Prof. Steven McLaughlin, Prof. David Anderson, Prof. James Hamblen, and Prof. Michael Lacey for serving as committee members for my dissertation, and for sharing their expertise from a wide range of areas.

I would also like to thank my colleagues Arindam Basu, Stephen Brink, Craig Schlottmann, Scott Koziol, and Shubha Ramakrishnan. The discussions I had with them helped me in finding pathways whenever I was facing numerous barriers in my research.

While I am doing research, I felt so lucky that I can read others' elaborate works with just a few clicks, meeting legendary scholars via letters and being updated with the latest state-of-the-art works. I hope that I can fill in a small yet firm brick with this work into the ever-growing towel of knowledge.

Finally, I cannot forget those enjoyable time I had with my Korean fellows Hyungsuk Yoo, Seokchul Kwon, Seungbae Lee, Dr. Namsik Kim, Dr. Jongsub Baek, Dr. Inho Lee, Dr. Hyungsuk Jeon, and Dr. Eunseok Ryu. I owe to them a lot in making my Ph.D. life filled with plenty of varieties and many delightful discussions, from academic enthusiasm to pleasant humors.

I am already missing my time at Georgia Tech - the class room, library, 'Tin Drum', fishbowl lab, Centergy 5th floor, and people there. So, I just want to say thank all I met here at Georgia Tech for everything. I hope this first step will lead me to follow the right track of being a good researcher as they all have gone through so far.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

The purpose of this research is to present a low-power wireless communication receiver with an enhanced performance by relieving the system complexity and performance degradation imposed by a quantization process. With an overwhelming demand for more reliable communication systems, the complexity required for modern communication systems has been increased accordingly. A byproduct of this increase in complexity is a commensurate increase in power consumption of the systems. Since the Shannon's era, the main stream of the methodologies for promising the high reliability of communication systems has been based on the principle that the information signals flowing through the system are represented in digits. Consequently, the system itself has been heavily driven to be implemented with digital circuits, which is generally beneficial over analog implementations when digitally stored information is locally accessible, such as in memory systems. However, in communication systems, a receiver does not have a direct access to the originally transmitted information. Since the received signals from a noisy channel are already continuous values with continuous probability distributions, we suggest a mixed-signal system in which the received continuous signals are directly fed into the analog demodulator and the subsequent soft-decision Viterbi decoder without any quantization involved. In this way, we claim that redundant system complexity caused by the quantization process is eliminated, thus gives better power efficiency in wireless communication systems, especially for battery-powered mobile devices. This is also

beneficial from a performance perspective, as it takes full advantage of the soft information flowing through the system.

CHAPTER 1

INTRODUCTION

The purpose of this research is to present a low-power wireless communication receiver with an enhanced performance by relieving the system complexity and performance degradation imposed by a quantization process.

With an overwhelming demand for more reliable communication systems, the complexity required for modern communication systems has increased accordingly to cope with several signal processing strategies that deal with inevitable channel impairments, such as noise, interference, and Doppler shift. Ever since the Shannon's era, researchers have aimed for more reliable communication systems by first converting fuzzy analog signals into digital signals based on the principle that the information flowing through a system can be represented in bits, either 0 or 1, and there is no ambiguity in-between. Consequently, the system itself has been heavily driven to be implemented with digital circuits.

Ironically enough, this digitizing (or quantizing) process imposes fundamentally redundant complexity and performance degradation into communication systems from an information theoretic point of view — any data processing can only reduce the mutual information or at best keep it to be the same. In fact, with the advent of soft-decision decoding schemes, this digitizing process imposes an upper limit on the performance of

communication systems due to the quantization errors experienced in representing soft information.

Digital implementations are generally beneficial over analog implementations when digitally stored information is locally accessible, such as in memory systems. However, in wireless communication systems, a receiver does not have a direct access to the originally transmitted information, and it can only use *continuous* received signals from a noisy channel to reconstruct the original information, preferably with the degree of reliability in each received signal conserved as *soft* information during the decoding process. Since the received signals from a noisy channel are already continuous values with continuous probability distributions, we suggest a mixed-signal system in which the received continuous signals are directly fed into the demodulator and the subsequent soft-decision Viterbi decoder without any quantization involved. In this way, we claim that redundant system complexity caused by a quantization process is removed, thus gives better power efficiency in wireless communication systems, especially for battery-powered mobile devices. This is also potentially beneficial from a performance perspective, as it takes full advantage of the soft information flowing through the system.

Despite the advantages of analog implementations over digital implementations both in performance and power efficiency aspects, they have not been chosen as the main stream of signal processing implementations for modern wireless communication systems, mainly due to the hardness in managing inherent device mismatch and nonlinearity of analog circuits. However, these mismatch and nonlinearity are *deterministic* in the sense that they are the characteristics of analog components determined at the fabrication process. With

the aid of a tunable property in the field-programmable analog array (FPAA) platform using floating-gate components, these impairments can be easily mitigated after the fabrication process.

Hence, as far as these deterministic mismatch and nonlinearity are well characterized and properly mitigated, analog systems potentially provide much better resolution in representing soft information than digital systems with finite bit-precision. So, with an appropriate control, the continuum exhibited in analog signals actually helps, and not degrades, the performance of the wireless communication systems — when a noise level for a particular sample is high enough to make it misunderstood at the receiver, it is preferable to leave the noisy signal *intact* than to produce any bit-flipping, then let the soft-decision decoder take care of the additional information on the degree of reliability conserved in the soft information. Since the final outputs from a decoder block are not the input signals nor the path metric signals themselves, but the binary decisions from the 2-entry comparator at the decoder, the ambiguity exhibited in analog signals is eventually removed during the decoding process.

In the following chapter, we will revisit the notions of analog and digital, and discuss whether the quantization process is worthwhile for different scenarios from a systematic point of view. Chapter 3 presents a continuous-valued discrete-Fourier transform (DFT) block for orthogonal frequency-division multiplexing (OFDM) systems, followed by discussions on the performance aspect along with the power consumption and the processing delay compared to digital implementations. In Chapter 4, a reduced-complexity soft-decision Viterbi decoder is discussed, and the performance comparisons between the

suggested mixed-signal implementation and digital implementations are reported. Chapter 5 addresses the conclusions of the research and the challenges that remain.

CHAPTER 2

DEFINITION AND SCOPE OF RESEARCH

2.1 Notions of Analog and Digital

We first revisit the notions of *analog* and *digital*. Every electric circuit consists of analog components when it comes down to a transistor level. A digital circuit is specially devised in such a way that signals flowing through the circuit are regulated to be represented in a binary manner — by utilizing the extreme nonlinearity of analog components. So, it is obvious that digital circuits need to perform additional tasks than analog circuits to regulate signals, such as managing noise, parasitic effects, and timing margins, and thus require more power. This can limit the use of digital systems especially in battery-powered mobile devices. For instance, a radio frequency (RF) front-end in a mobile terminal typically uses analog power amplifiers and tuners, whereas a base station with grid power can use a digitally implemented RF front-end. The extreme of this trend is a software-defined radio running on a general-purpose processor, which often requires an additional cooling system to prevent being overheated.

All signals observed in a real-life environment are analog, including signals received from a wireless channel. A sampling process brings continuous-time signals to discrete-time signals, but it does not necessarily accompany a quantizing process that brings the sampled discrete-time continuous-value into a discrete-time discrete-value. A sampling

Figure 1: An illustration of a sampling process without a quantization process. A dam blocks and permits water flow in discrete time, while the quantity of water in each drain is a continuous value.

process without a quantization process can be illustrated as Figure 1, in which a dam blocks and permits water flow in discrete time, while the quantity of water in each drain is a continuous value. Throughout this work, we will use the terms analog and digital based on the criterion that signals flowing through the system are continuous values or discrete values, and discuss whether the quantizing process is worthwhile for different scenarios.

2.2 Motivation

In modern communication systems, there are two main keywords that need to be considered — "reliable" communications, and "green" communications. Especially for the mobile terminals that operate on batteries, such as cellular phones, laptops, and tablets, the functionality of the devices lie directly on which signal processing gives more reliable performance with better power efficiency.

In order to meet the demand for higher data rates and more reliability in wireless communication systems, engineers have responded by increasing the level of sophistication in the signal processing strategies that deal with noise, interference, Doppler shift, and other channel impairments. A byproduct of this increase in complexity is a commensurate increase in power consumption of the systems. Since the Shannon's era, the main stream of the methodologies for promising the high reliability of communication systems has been based on the principle that the information signals flowing through the system are represented in digits. Consequently, the system itself has been heavily driven to be implemented with digital circuits. However, for mobile devices with limited battery power, replacing these digital circuits with low-power analog circuits can significantly improve the power efficiency of the devices [1][2]. The cost paid for this reduced power is the long development cycle and lack of flexibility that typifies analog circuit design.

So as to retain the rapid-prototyping capability and flexibility of a field-programmable gate array (FPGA) but with reduced power consumption, an analog counterpart of the FPGA, namely a field-programmable analog array (FPAA) was first proposed in [3], followed by several different FPAA realizations using a switched-capacitor [4][5], a transconductor [6], or an OTA with a capacitor [7]. The early FPAAs, however, contained only a few computational elements and their applications were restricted to analog filters, until floating-gate transistors were used as switches of the FPAA to enable large-scale analog circuit design [2][8]. Recently, a hexagonal arrangement of computational analog blocks (CABs) was reported in [9][10] to reduce the size and path delay of the FPAA chip. Two decades since its advent, the FPAA is finding viability in space applications as well by imposing self-reconfigurable features [11][12].

Another distinction of an analog circuit from a digital circuit is that it naturally deals with continuous signals. For digital information represented in bits, all the information is a series of either 0 or 1, and there is no ambiguity in-between. So, if the information signal is disturbed by a noise, the noisy signal can be recovered back to discrete bits — though may include some bit-flipping — through a hard slicer. However, during this quantizing process the degree of reliability conveyed in the noisy signal is distorted by quantization errors. It is well known that this additional knowledge on the noisy signal at the receiver can contribute to a better chance to recover the originally transmitted signal correctly [13]-[15]. In order to maintain this additional knowledge, several soft-decision decoding schemes have been suggested, in which each noisy bit is represented in multiple bits depending on the bit-precision of the digital system [16]-[20]. This implies that the quantizing process at the analog-to-digital converter (ADC) block needs to allocate multiple bits to represent each received signal. In wireless communications systems, this can be considered as a redundant complexity, since the received signals from a noisy wireless channel are already continuous values that possess the degrees of reliability in themselves, thus can be directly applied to the soft-decision decoder. With this approach, the quantizing process takes place after the decoder, so the decoder block has to be designed in analog. The quantization after the decoder becomes trivial, as it only requires 1-bit precision with a hard slicer. This approach is also beneficial from a performance perspective, as it takes full advantage of the soft information flowing through the decoding system. There have been extensive research solely on reducing the ADC power consumption by deploying several different topologies [21]-[26]. In this work, we take a

different approach to the problem — by fundamentally removing the ADC requirement without compromising performance of the system.

We want to make it clear that quantization error does not always degrade the performance. For signals with a high signal-to-noise ratio (SNR), such as signals in local memory systems mentioned earlier, the quantization process removes the fuzziness in noisy signals. However, a channel coding scheme sacrifices a data rate by appending redundant information to the original information in order to combat noisy channel conditions with a limited SNR. Thus, in the context of a channel coding, we are more interested in signals with a low SNR, and quantizing these received signals that have undergone a noisy channel often yields bit-flipping.

2.3 Problem Statement

In this research, we present a low-power wireless communication receiver by eliminating redundant complexity due to the quantization process and by replacing digital circuits with small-signal analog circuits.

To verify this statement, we used an FPAA platform to implement essential signal processing blocks for modern communication systems, such as a DFT and a Viterbi decoder, with proposed mixed-signal circuit configurations, and compared the measurement results to that from digital implementations.

CHAPTER 3

DISCRETE-FOURIER TRANSFORM FOR CONTINUOUS SIGNALS IN OFDM SYSTEMS

3.1 Backgrounds

The OFDM is widely used in numerous wireless communications systems not only because of its spectral efficiency and robustness to a multipath fading, but also because of its ease of implementation; OFDM modulators and demodulators can be implemented using simple fast-Fourier transform (FFT) blocks, typically in digital circuits. For a mobile terminal which is powered by a battery, replacing these digital circuits — the FFT block for the downlink and the IFFT block for the uplink — with low-power analog circuits can significantly improve the power efficiency of the devices.

There have been several dedicated non-programmable analog implementations of FFT and DFT circuits. A voltage-mode analog FFT block was reported in [27][28] that requires analog multipliers and dedicated input signals representing the FFT coefficients, and analog adders for summing voltage signals. An FFT based on analog current mirrors was proposed in [29], where the FFT coefficients are not reconfigurable but are determined by the W/L ratio of the output transistor of each mirror. More recently, a numerical

simulation for approximating a fast DFT operation with a 2-D lattice of inductors and capacitors was introduced in [30].

To overcome the drawbacks of previous works, this work presents a current-mode analog DFT block implemented as a vector-matrix multiplier (VMM) using the reconfigurable analog signal processor (RASP) 2.9 FPAA chip [8]. Floating-gate transistors inside the FPAA chip are used as partially connected switches to store the DFT coefficients by locking in an appropriate amount of electrical charge in each floating-gate capacitor. So, dedicated input signals representing the DFT coefficients are not required. Furthermore, these coefficients are reconfigurable without changing the circuit structure. The VMM structure using floating-gate transistors as programmable switches also enables tuning the DFT coefficients to compensate for the inherent mismatch between different transistors. Another benefit of the current-mode design over a voltage-mode circuit is the ease with which signals can be added and broadcasted, which is especially beneficial in systems having multiple inputs and multiple outputs. The RASP 2.9 FPAA chip contains more computational elements than previous FPAA chips, including the commercial products of [31][32]. The large number of computational elements and the configurable floating-gate switches make the RASP 2.9 FPAA chip viable in a wide range of applications. Such versatility is an important figure of merit for any programmable circuit.

3.2 System Description and Analysis

3.2.1 Analog OFDM Demodulator

Figure 2(a) illustrates a simplified block diagram of a conventional OFDM receiver, such as for an 802.11 a/g system, where the received signals are converted to digital signals immediately after downconversion. The OFDM demodulation is performed using a digitally implemented DFT. As an alternative, an analog implementation is proposed, as shown in Figure 2(b), where the downconverter outputs are fed directly to the FPAA, without being converted to digital signals. The DFT functionality is implemented in analog using the FPAA. The N outputs of the FPAA — one for each subcarrier — are each converted to digital signals separately.

Besides the reduced power consumption with an analog implementation of the DFT, an important benefit of the proposed receiver structure in Figure 2(b) is that it greatly

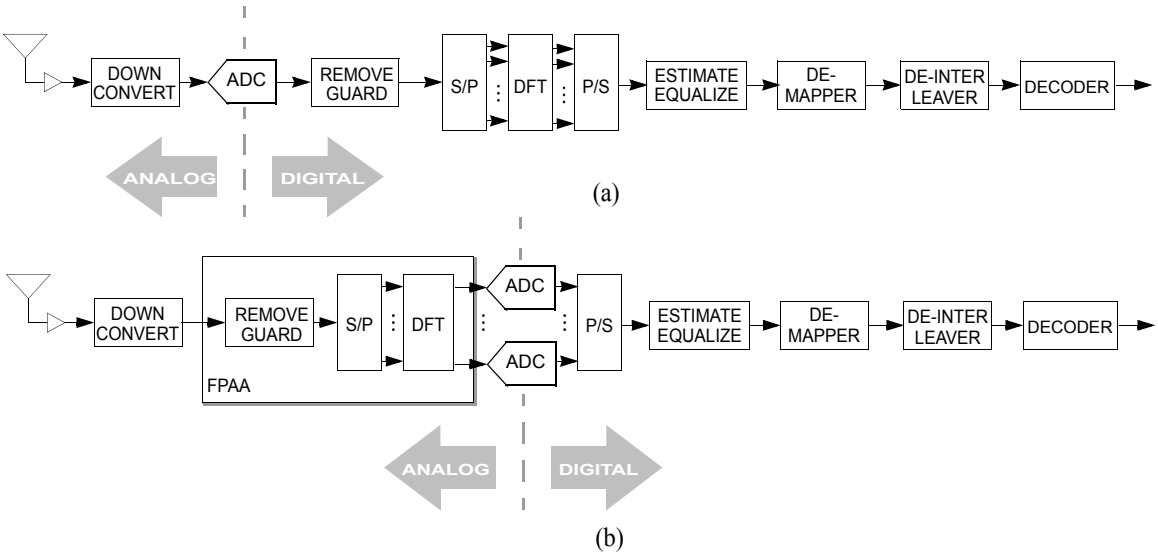


Figure 2: An OFDM receiver: (a) conventional implementation in which ADC occurs before a digital DFT; (b) the proposed implementation in which ADC occurs after an analog DFT.

relieves the speed and precision burdens of the ADC. In particular, the ADC of the conventional receiver shown in Figure 2(a) would need to sample at a rate equal to the full signal bandwidth, and its precision would need to be high (on the order of 10 bits or more) to accommodate the wide dynamic range and Gaussian-like distribution of OFDM signals. In contrast, the proposed receiver of Figure 2(b) has not one but N ADCs, one for each subcarrier, with a sampling rate slower by a factor of N. Moreover, each DFT output is a finite alphabet signal that can be sampled with significantly less bit precision [28]. Compared to the ADC in Figure 2(a), each ADC in Figure 2(b) requires a sampling rate that is lower by a factor of N, and a number of bits of precision is smaller by a factor of three or more, depending on the modulation alphabet size. Both effects yield a reduction in ADC power consumption, although the exact amount depends on the type of the ADC structure; ADC power consumption is between a linear and quadratic function of the sampling rate [33].

Thus, the proposed receiver of Figure 2(b) is beneficial not only because of the reduced power consumption with an analog implementation of the DFT, but also because of the additional power savings resulting from the lower speed and bit-precision requirements of the subsequent ADC. Although the focus here is on the receiver side, it can be briefly pointed out that these same advantages are also valid at the transmitter side, where the outputs from the symbol mapper are modulated with an inverse discrete-Fourier transform (IDFT) block, which has the same structure as the DFT block with different coefficients. Besides the power savings with an analog IDFT implementation, shifting the IDFT block after the digital-to-analog converter (DAC) enables an OFDM transmitter to replace a full-

speed high-precision DAC by N separate DACs, each operating at a $1/N$ times lower clock with lower bit-precision.

3.2.2 Floating-Gate Transistors and the RASP 2.9 FPAA

Floating-gate transistors can store a non-volatile electrical charge, so arrays of floating-gate transistors can be programmed as a signal processing block for specific functionality. One application is to use them as a VMM circuit. Figure 3 shows a current multiplier circuit — the basic element of a VMM circuit — composed of floating-gate transistors and an operational transconductance amplifier (OTA). Two pMOSFETs are connected at the source, and the gate of each transistor is connected to a capacitor C_g to be electrically isolated, so as to form a *floating* gate. The source voltage V_s is common for both transistors, and $V_{g,1}$, $V_{g,2}$ are the voltage potential at the input and output gates. V_d is the drain voltage of the transistor. The voltage on the other side of the capacitors connected to the gates are set to be the same at the fixed potential V_{fg} .

The input current I_{in} and output current I_{out} are defined as drain currents of the transistors operating in the sub-threshold region. Neglecting the Early effects, the input and output currents of the pMOSFET are given by [34]:

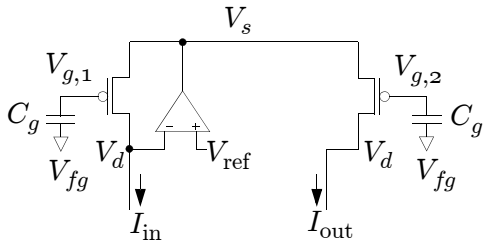


Figure 3: A current multiplier circuit composed of floating-gate transistors and an OTA. The output current is a scaled multiple of the input current.

$$I_{\text{in}} = I_1 \exp\left(\frac{V_s - \kappa_{\text{eff}} V_{g,1}}{U_T}\right)\left\{1 - \exp\left(-\frac{V_s - V_d}{U_T}\right)\right\}, \tag{1}$$

$$I_{\text{out}} = I_2 \exp\left(\frac{V_s - \kappa_{\text{eff}} V_{g,2}}{U_T}\right)\left\{1 - \exp\left(-\frac{V_s - V_d}{U_T}\right)\right\}, \tag{2}$$

where $U_T = kT/q$ (where T is temperature, k Boltzmann constant, and q elementary charge). Let C_t be the total capacitance at each gate including the floating-gate capacitor C_g and the internal capacitance of the MOSFET, then $\kappa_{\text{eff}} = \kappa C_g/C_t$ where κ is the back-gate coefficient. The parameters I_1 and I_2 are the pre-exponential factors of the MOSFETs that can be defined as a drain current flowing in each transistor when $V_s = \kappa_{\text{eff}} V_{g,1}$ and $V_s = \kappa_{\text{eff}} V_{g,2}$, respectively. Let Q_1 and Q_2 be the electrical charges stored in the input and output floating-gate capacitors, then the voltage drop across each capacitor is given by

$$V_{g,1} - V_{fg} = \frac{Q_1}{C_g}, \tag{3}$$

$$V_{g,2} - V_{fg} = \frac{Q_2}{C_g}. \tag{4}$$

Plugging in the gate voltage in (3), (4) into (1), (2) yields

$$I_{\text{in}} = I_1 \exp\left(\frac{V_s - \kappa_{\text{eff}} V_{fg}}{U_T}\right)\exp\left(\frac{-\kappa Q_1}{C_t U_T}\right)\left\{1 - \exp\left(-\frac{V_s - V_d}{U_T}\right)\right\}, \tag{5}$$

$$I_{\text{out}} = I_2 \exp\left(\frac{V_s - \kappa_{\text{eff}} V_{fg}}{U_T}\right)\exp\left(\frac{-\kappa Q_2}{C_t U_T}\right)\left\{1 - \exp\left(-\frac{V_s - V_d}{U_T}\right)\right\}. \tag{6}$$

When there is negligible mismatch between the threshold voltage[1] of the input and output transistors, so that I_1 and I_2 are approximately identical, the ratio of the output current to the input current reduces to

$$W = \frac{I_{\text{out}}}{I_{\text{in}}} = \exp\left(\frac{-\kappa(Q_2 - Q_1)}{C_t U_T} \right). \tag{7}$$

So, the weighting coefficient W is determined by the difference in charge values between the input and output floating-gate transistors, provided that both transistors operate in the sub-threshold region. One can observe in (7) that W is also a function of temperature. Note that from (7) only positive weights can be realized. The weight zero can be realized by not connecting the input and output floating-gate transistors.

In practice, there exists an inherent mismatch between the threshold voltage of different transistors which leads to a mismatch in the pre-exponential factors I_1 and I_2, and this in turn leads to multiplicative distortion in the weighting coefficient. However, this mismatch can be compensated for by adjusting the charge values Q_1 and Q_2while they are programmed into the FPAA chip. Although a floating-gate current mirror can be similarly tuned over a narrow range to compensate for the device mismatch, its weight is fixed by the W/L ratio of the transistors and cannot be changed widely without changing the circuit structure.

Figure 4 shows a plot of the output currents versus input current for a set of programmed weights between $1/4$ and 4. The mismatch effect has been cancelled through

1. The threshold voltage is the gate voltage at which channel formation occurs between oxide and body of a transistor.

the weight programming procedure that will be discussed in Chapter 3.3. The input current is swept from 0.2 μA to 1.0 μA. For $W \leq 1$, the linear relation between I_{in} and I_{out} shows that the programmed circuit can be used as a current multiplier for the range of currents shown. It can be observed that the output current shows transition to the strong-inversion region as W increases.

The basic current multiplier circuit of Figure 3 can be expanded to a multiple-input multiple-output structure to construct a larger size VMM circuit in the FPAA. The RASP 2.9 FPAA [8] consists of 133,744 floating-gate transistors and 84 CABs, each of which contains three OTAs, three capacitors, a transmission gate, and a voltage buffer. The floating-gate transistors can be programmed as a VMM circuit by storing an appropriate

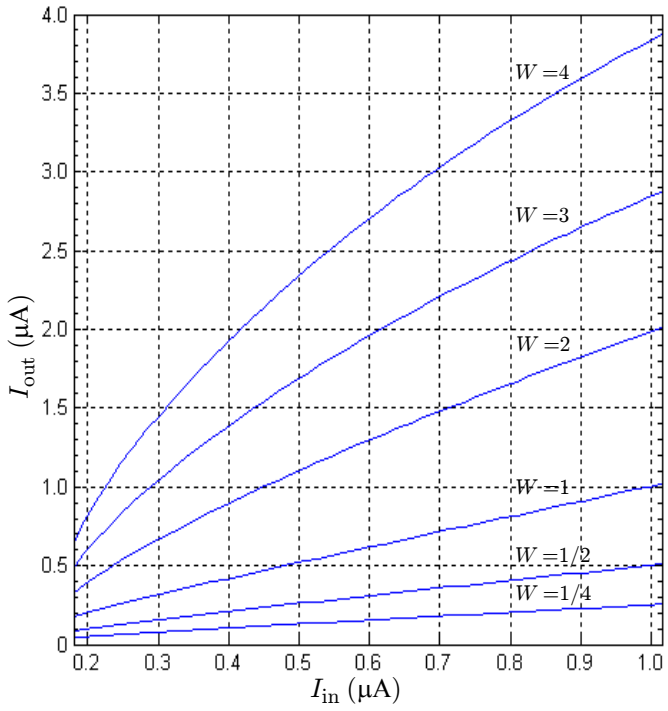


Figure 4: The output currents with the programmed weights of $1/4$ to 4. The input current is swept from 0.2 μA to 1.0 μA. The straight curves for $W \leq 1$ shows the programmed circuit can be used as a current multiplier in the current range. As W increases, the output current shows transition towards the strong-inversion region.

amount of charge in each floating-gate capacitor. Figure 5 depicts the RASP 2.9 FPAA chip mounted on the board. The chip is fabricated with a 0.35μm CMOS process. The size of the transistors is $W = 1.8$ μm, $L = 0.6$ μm. Figure 6 depicts the wide-output-range OTA inside the CABs, that is used to supply the input and output currents of the VMM circuit. The board has 56 I/O pins that can be used to set drain voltages of the floating-gate transistors and to measure output currents. Programming process for the FPAA chip is controlled through a USB interface equipped on the board.

The floating-gate transistors in the FPAA are also used as fully turned-on switches to connect routed nodes. This comes from the same principle given in (3), (4). As the negative charge stored in the floating-gate capacitor increases, the gate voltage decreases for the

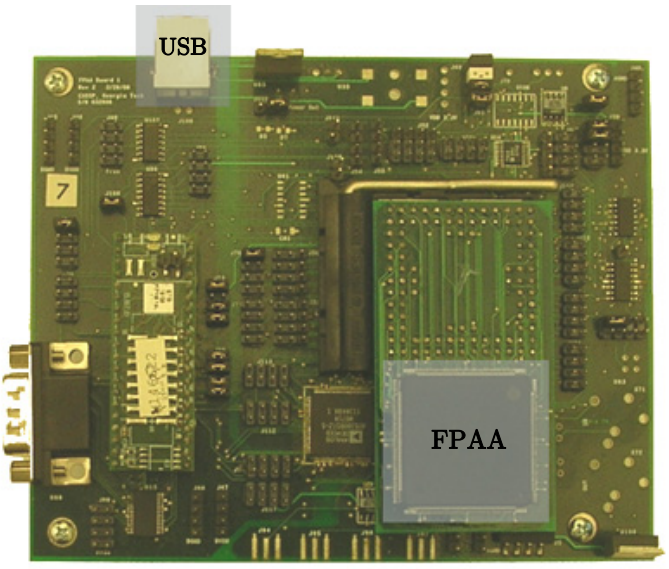


Figure 5: The RASP 2.9 FPAA chip mounted on the board. The chip is implemented in the 0.35 μm CMOS process. The board has 56 I/O pins for setting drain voltages of floating-gate transistors and measuring output currents. It is connected to a PC through a USB interface for controlling the FPAA chip programming. The sizes of the chip and the board are 5 mm × 5 mm and 114 mm × 140 mm, respectively.

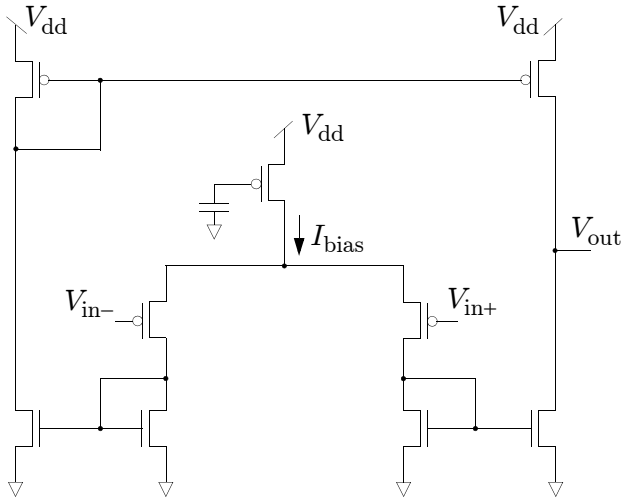


Figure 6: A wide-output-range OTA inside the CABs of the RASP 2.9 FPAA chip. V_{dd} is the bias voltage and I_{bias} is the bias current. The floating gate transistor in the middle is programmed with the charge value corresponding to the bias current. The targeting bias current is determined at the amount sufficient to provide the input and output currents of the connected floating-gate transistors.

same floating-gate voltage, thus turns on a pMOSFET that is connected to the floating-gate capacitor.

3.2.3 VMM Representation of Continuous-Valued DFT

The key to an OFDM receiver is to compute the DFT of a set of complex samples $\{x(n)\}$, defined by:

$$X(k) = \frac{1}{N}\sum_{n=0}^{N-1} x(n)\, e^{-j2\pi kn/N}, \tag{8}$$

where N is the number of subcarriers and k is an integer ranging from 0 to $N-1$. By splitting each complex number into real and imaginary parts, we can rewrite (8) as

$$Re[X(k)] = \frac{1}{N}\sum_{n=0}^{N-1}\left[Re[x(n)]\cos\left(\frac{2\pi kn}{N}\right) + Im[x(n)]\sin\left(\frac{2\pi kn}{N}\right)\right], \quad (9)$$

$$Im[X(k)] = \frac{1}{N}\sum_{n=0}^{N-1}\left[-Re[x(n)]\sin\left(\frac{2\pi kn}{N}\right) + Im[x(n)]\cos\left(\frac{2\pi kn}{N}\right)\right], \quad (10)$$

which is equivalent to a VMM:

$$\mathbf{X} = \mathbf{Hx}, \quad (11)$$

where $\mathbf{X}$ is a $2N \times 1$ vector consisting of the real and imaginary components of $\{X(k)\}$, where $\mathbf{x}$ is a $2N \times 1$ vector consisting of the real and imaginary components of $\{x(n)\}$, and where $\mathbf{H}$ is a real-valued $2N \times 2N$ matrix.

The VMM of (11) cannot be implemented directly, as some of the coefficients in $\mathbf{H}$ are negative. Instead, we represent each signal differentially, and we represent each element of $\mathbf{H}$ by a 2×2 non-negative differential submatrix by mapping a positive gain G to $[G\ 0;\ 0\ G]$, and a negative gain $-G$ to $[0\ G;\ G\ 0]$. We thus transform the $2N \times 2N$ matrix $\mathbf{H}$ into an equivalent $4N \times 4N$ matrix with non-negative weights that can be programmed into the FPAA chip.

For a size-4 DFT, we need a 16×16 VMM circuit with real-valued non-negative weights. The schematic of the 16×16 VMM structure in the FPAA chip is shown in Figure 7. As discussed in Chapter 3.2.2, the weighting coefficients $W_{1,1}$ to $W_{16,16}$ can be programmed into the FPAA chip by assigning appropriate charge values to the input and output floating-gate transistors. Note that the weights in each column of the VMM circuit corresponds to the coefficients in each row of the 16×16 VMM matrix. Even though

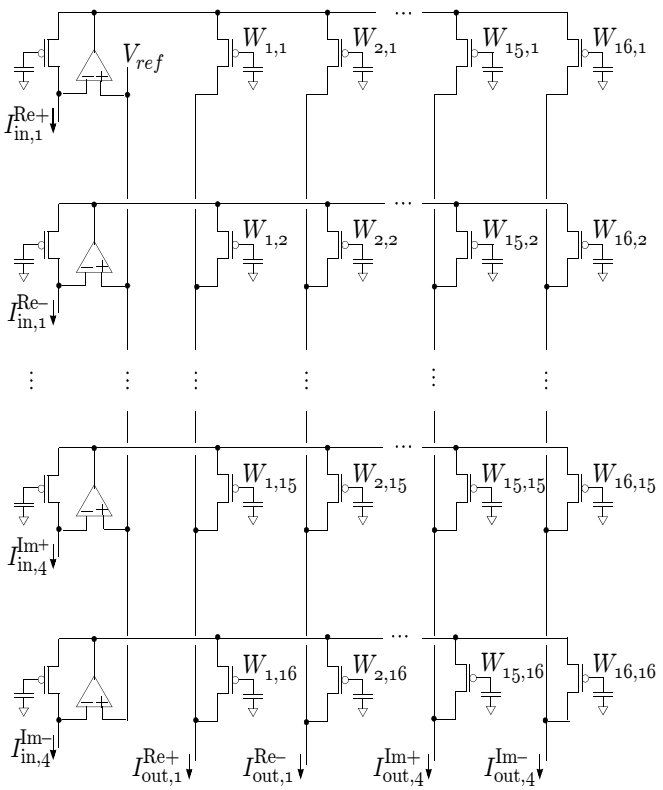


Figure 7: FPAA implementation of a 4-point DFT as a 16 × 16 VMM circuit. Each input current of the floating-gate transistors on the left determines the output voltage of the corresponding OTA, and this output voltage is broadcasted to the source of all the connected floating-gate transistors in each row. At the output floating-gate transistor, this source voltage drives a drain current which is a scaled multiple of the corresponding input current. Then, the scaled currents are added up along each column to give a combined output current per each column.

signals flowing through the VMM circuit are continuous values, it is still a discrete Fourier transform in the sense that each signal travels through a distinct path.

The output port of each OTA is connected to the source of the input floating-gate transistor, and the negative input port is connected to the drain. The positive input port is set to the reference voltage *V*ref. Because of the negative feedback of the OTA, the drain voltage of each input floating-gate transistor is also close to *V*ref. So, the input currents of the VMM circuit, defined as the drain currents of the input floating-gate transistors, can be

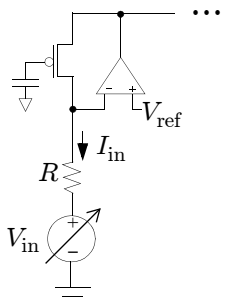


Figure 8: An input voltage supplied to the drain of an input floating-gate transistor through a resistor. The setup enables the received OFDM signals to be linearly mapped to the input current of the VMM circuit.

controlled by connecting a resistor to the drain of each input floating-gate transistor then varying the input voltage V_{in} applied to the resistors. This configuration is shown in Figure 8. Due to the non-ideal characteristic of the negative feedback OTA, the realistic gain of the negative feedback OTA is finite. So, the negative input voltage of the OTA does not stay close to V_{ref}, especially when the input voltage V_{in} gets close to V_{ref}, or equivalently the input current I_{in} gets close to zero, as shown in Figure 9. Hence, the operating range of the input current is chosen at 0.2 μA to 1.0 μA in order to provide a linear relationship between the input voltage and the corresponding input current, so that the received OFDM signals can be linearly mapped to the input currents of the VMM circuit, and to minimize redundant power consumption while holding a reasonable current resolution and path delay. This range also guarantees that each transistor operates in the sub-threshold region for weights less than or equal to 1.

The drain current of each input floating-gate transistor determines the output voltage of the corresponding OTA, and this output voltage is broadcasted to the source of all the connected floating-gate transistors in each row. When the drain voltage of each output

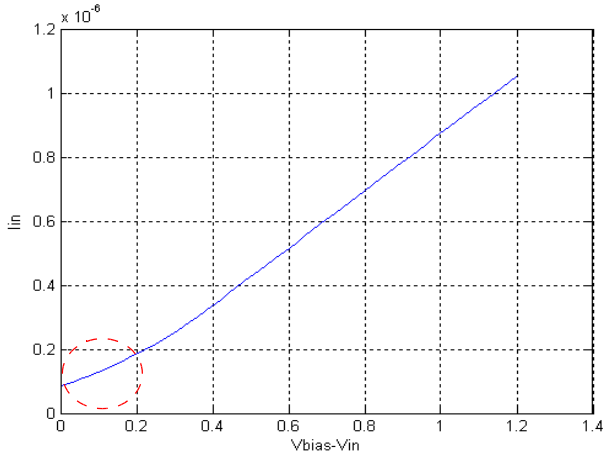


Figure 9: The input current Iin of the OTA exhibits nonlinearity as the input voltage Vin gets close to $Vref$, due to the non-infinite gain of the negative feedback OTA.

floating-gate transistor is set to *V*ref, this source voltage drives a drain current of each output floating-gate transistor, which is a scaled multiple of the corresponding input current. Then, the drain currents from the output floating-gate transistors are added up along each column to give a combined output current per each column.

Note that in Figure 7 each output floating-gate transistor has only one OTA connected to it, so the current level flowing through each output transistor is determined by the input current level and the programmed weights with respect to the connected input. Also, the DFT coefficients involve a factor of $1/N$ as shown in (8), so all the converted non-negative weights span within the range of $0 \le W \le 1/N$. These factors guarantee that each transistor of the VMM circuit operates in the sub-threshold region, as far as the input current level stays less than the threshold current.

The required operations in the VMM given in (11) are scaling and summing operations. Since the information signals are conveyed in current levels, the summing operations do

not require additional circuits. So, the power consumption becomes less than that for the digital circuits where the information signals are conveyed in voltage levels and adding entries requires full adders. The scaling operations do not involve any complex multiplications, as all the signals and weights are real-valued. So, the power consumption in the scaling operations is also limited.

Now, we can take into account a butterfly operation in order to reduce the number of computations for a DFT operation. The butterfly operation basically decomposes a DFT matrix into a series of smaller matrix, and each output from the previous stage is handed over to the next stage. This can be viewed as a cascade of VMMs, where VMM size in each stage gets smaller by a factor of the radix size. So, it is clear that applying butterfly operations in the VMM circuit increases the path delay of the circuit. As the radix size gets lower, the number of stage increases, and consequently, the path delay increases. Also, the butterfly operation substitutes copying operations for summing and scaling operations. This is beneficial in voltage-mode circuits, where a summing operation requires a full adder whereas a copying operation is trivial. However, in a current-mode circuit, a copying operation requires a current mirror or a unity-gain current multiplier, whereas a summing operation is trivial. It is also claimed in [29] that an FFT design with a higher radix becomes less sensitive to the device mismatch. For these reasons, a full-radix DFT is more preferable for a current-mode analog circuit design.

3.3 Continuous-Valued DFT Implementation in FPAA

3.3.1 Programming Procedure

To simplify implementation of an analog DFT in the FPAA, we scale up the DFT matrix by a factor of N, with the understanding that it can be compensated for by scaling down the outputs of the analog DFT by a reciprocal of the scaling factor. This simplification makes the coefficients span within $0 \le W \le 1$ range, so each transistor will still operate in the sub-threshold region. In particular, the 16×16 matrix for a 4-point DFT contains only 1's and 0's with this simplification, as the phases of complex coefficients are multiples of $\frac{\pi}{2}$. This makes the amount of charge to be stored in each transistor relatively close to each other, so as to increase the linearity between input and output current levels for each transistor. The resulting matrix shown in Figure 10 determines the weighting coefficients of the VMM circuit in Figure 7. The netlist of the 16×16 VMM circuit for a 4-point DFT is given in Appendix A. Note that the targeting values with 0 in the netlist are shown only for the sake of integrity and can be omitted out.

$$
\begin{bmatrix}
\mathrm{Re}[X(0)]+ \\ \mathrm{Re}[X(0)]- \\ \mathrm{Re}[X(1)]+ \\ \mathrm{Re}[X(1)]- \\ \mathrm{Re}[X(2)]+ \\ \mathrm{Re}[X(2)]- \\ \mathrm{Re}[X(3)]+ \\ \mathrm{Re}[X(3)]- \\ \mathrm{Im}[X(0)]+ \\ \mathrm{Im}[X(0)]- \\ \mathrm{Im}[X(1)]+ \\ \mathrm{Im}[X(1)]- \\ \mathrm{Im}[X(2)]+ \\ \mathrm{Im}[X(2)]- \\ \mathrm{Im}[X(3)]+ \\ \mathrm{Im}[X(3)]-
\end{bmatrix}
=
\begin{bmatrix}
1&0&1&0 & 1&0&1&0 & 0&0&0&0 & 0&0&0&0 \\
0&1&0&1 & 0&1&0&1 & 0&0&0&0 & 0&0&0&0 \\
1&0&0&0 & 0&1&0&0 & 0&0&1&0 & 0&0&0&1 \\
0&1&0&0 & 1&0&0&0 & 0&0&0&1 & 0&0&1&0 \\
1&0&0&1 & 1&0&0&1 & 0&0&0&0 & 0&0&0&0 \\
0&1&1&0 & 0&1&1&0 & 0&0&0&0 & 0&0&0&0 \\
1&0&0&0 & 0&1&0&0 & 0&0&0&1 & 0&0&1&0 \\
0&1&0&0 & 1&0&0&0 & 0&0&1&0 & 0&0&0&1 \\
0&0&0&0 & 0&0&0&0 & 1&0&1&0 & 1&0&1&0 \\
0&0&0&0 & 0&0&0&0 & 0&1&0&1 & 0&1&0&1 \\
0&0&0&1 & 0&0&1&0 & 1&0&0&0 & 0&1&0&0 \\
0&0&1&0 & 0&0&0&1 & 0&1&0&0 & 1&0&0&0 \\
0&0&0&0 & 0&0&0&0 & 1&0&0&1 & 1&0&0&1 \\
0&0&0&0 & 0&0&0&0 & 0&1&1&0 & 0&1&1&0 \\
0&0&1&0 & 0&0&0&1 & 1&0&0&0 & 0&1&0&0 \\
0&0&0&1 & 0&0&1&0 & 0&1&0&0 & 1&0&0&0
\end{bmatrix}
\cdot
\begin{bmatrix}
\mathrm{Re}[x(0)]+ \\ \mathrm{Re}[x(0)]- \\ \mathrm{Re}[x(1)]+ \\ \mathrm{Re}[x(1)]- \\ \mathrm{Re}[x(2)]+ \\ \mathrm{Re}[x(2)]- \\ \mathrm{Re}[x(3)]+ \\ \mathrm{Re}[x(3)]- \\ \mathrm{Im}[x(0)]+ \\ \mathrm{Im}[x(0)]- \\ \mathrm{Im}[x(1)]+ \\ \mathrm{Im}[x(1)]- \\ \mathrm{Im}[x(2)]+ \\ \mathrm{Im}[x(2)]- \\ \mathrm{Im}[x(3)]+ \\ \mathrm{Im}[x(3)]-
\end{bmatrix}
$$

Figure 10: 16x16 VMM representation with non-negative coefficients for 4-point DFT.

The netlist is taken by the RASPER tool that places and routes the available components in the FPAA chip [35]. The output file of the RASPER is a list of switch addresses and the targeting current value for each switch. This list is loaded by the Matlab script to be programmed into the FPAA chip. The RASP 2.9 FPAA chip contains the necessary circuitry for tunneling and injecting electrical charges of floating-gate transistors and the circuitry for current measurement. All the stored charges are tunneled before getting programmed, and an appropriate amount of charge value is injected to each floating-gate transistors while targeting on the corresponding current level determined by (1) and (2). The targeting current levels are represented with 10-bit floating-point precision, of which 3 bits are assigned for the exponent and 7 bits for the significand [36]. Even though the information signals are conveyed in un-quantized current levels, the accuracy in the programmed weights will impose a limit to the resolution of the analog DFT system.

In order to reduce the circuitry required for measurement and tunneling and injecting charges, the indirect programming method is used to charge the floating-gate transistors [37]. Figure 11 shows the indirect programming structure for a floating-gate pMOSFET.

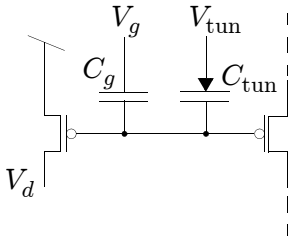


Figure 11: Indirect programming structure of a pMOSFET. The left transistor is part of the on-chip programming circuitry and is actively programmed. The transistor on the right is the transistor that is used for the VMM circuit and is passively programmed.

The floating-gate transistor on the left is connected to the on-chip programming circuitry, and is actively programmed. The one on the right is the floating-gate transistor that is used for the VMM circuit, and is passively programmed.

Due to the inherent mismatch between threshold voltages of different transistors, the indirectly programmed charges in the floating-gate transistors for the VMM circuit can be different from the directly programmed charges in the floating-gate transistors of the programming circuitry. This mismatch also occurs in-between the programmed charges in input and output floating-gate transistors of the VMM circuit. While this mismatch is inherent, we can circumvent this by adjusting the charge value in each input and output floating-gate transistor. We will discuss this process in the following chapter.

3.3.2 Mismatch Compensation

When there is mismatch in threshold voltages of different transistors, the pre-exponential factor of each MOSFET can be different from each other, and thus the programmed weights can suffer from multiplicative distortion. However, the ratio of the output current to the input current is a function of the relative difference in charge values as shown in (7), so the mismatch in weights can be compensated for by adjusting the charge values in the floating-gate transistors. This process can be accomplished by targeting on the desired *ratio* of the input and output currents, rather than targeting on the desired input and output currents themselves.

In the VMM circuit, each output current is the sum of the drain currents of the output floating-gate transistors in each column. Due to the different levels of nonlinearity in the I-V characteristics of the input OTAs, there exist additive offsets between the input and

output current levels. So, targeting on the ratio at a single point will not suffice. Instead, we need two points of measurement, so that a *slope* of the output current versus input current can be targeted. Thus, the FPAA programming procedure is conducted in the following two steps, so as to minimize errors in the programmed weights.

- *Coarse programming*
  - The fully turned-on switches and the input floating-gate transistors are first programmed with the desired charge values by targeting on the corresponding current levels.
  - The floating-gate transistors inside the OTAs are also programmed with the charge values corresponding to the bias current.
  - On the other hand, the output floating-gate transistors are programmed with lower charge values than desired by targeting on a half of the corresponding current levels.
- *Fine programming*
  - Each output floating-gate transistor is then injected with a small amount of electron iteratively to increase the stored charge.
  - In each iteration, the input and output current values are measured at two different input voltages, then the slope of input and output currents is obtained.
  - The iteration stops when the slope of the input and output currents reaches the desired weight.

After the fine programming, the output currents of the VMM circuit still involve additive offsets, but these offsets do not vary as the input current levels change. So, the sum of these offsets per each output node is constant, and it can be easily calculated to be subtracted out from each output current.

3.4 Measurement Results

3.4.1 FPAA Measurement

We now investigate the measured data of the OFDM receiver with an analog DFT demodulator. The transmitted symbols are randomly generated and mapped to 16 QAM

complex symbols with a Gray coding. The generated symbols are then modulated by a size-4 IFFT. The guard interval is allocated for $1/4$ of the FFT size, and the resulting complex samples are serialized, applied to a DAC, and upconverted to the carrier frequency. On the receiver side, after downconversion and removal of the guard interval, the received OFDM signals are split into real-valued differential pairs, and converted to the input currents of the analog DFT within the current range of $0.2 - 1.0$ μA as discussed in Chapter 3.2.3. The converted 16 input currents are then fed into the 16×16 VMM analog circuit implemented in the FPAA to demodulate the OFDM signals. The resulting 16 output currents of the FPAA are sampled, then reverted back to the voltage levels, and reassembled to yield four complex single-ended demodulated OFDM signals. These are then fed into a 16 QAM de-mapper to recover the transmitted symbols.

Figure 12(a)-(b) show the I-Q plots of the demodulated symbols without injecting any channel noise. The color maps are used to illustrate the density of the occurrence. It can be observed in Figure 12(a) that there is some dispersion in the demodulated symbols even without any channel noise, which results in a performance penalty as a price for the reduced power consumption in the analog DFT. The performance degradation arises for multiple reasons, including:

- errors in the programmed weights due to the limit on the bit-precision for targeting current values
- temperature sensitivity of the programmed weights
- nonlinear mapping between input voltage and input current caused by the non-ideal characteristic of the OTA
- thermal noise
- parasitic capacitance between routed paths

Despite the dispersion, however, the 16 QAM de-mapper in the Matlab determined all the demodulated symbols correctly. This implies that the error rate will still converge to zero in a noisy channel as the SNR increases.

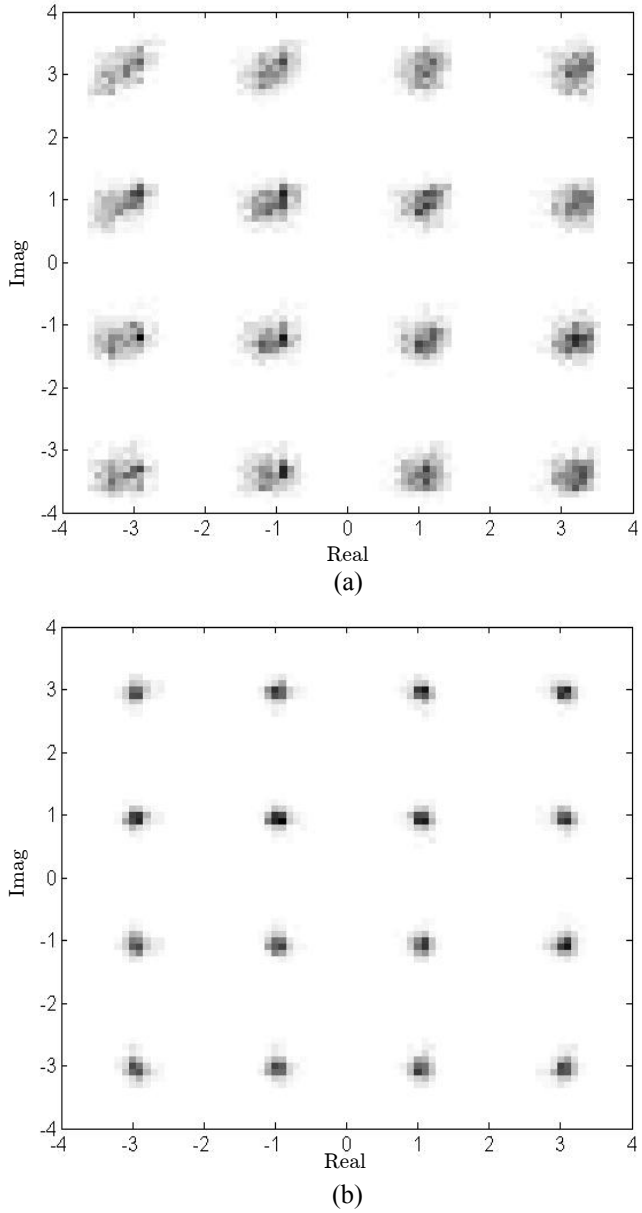


Figure 12: Constellations of the demodulated symbols for 16 QAM without channel noise: (a) before equalization; (b) after MMSE equalization. The gradation depicts the density of the occurrence in each pixel.

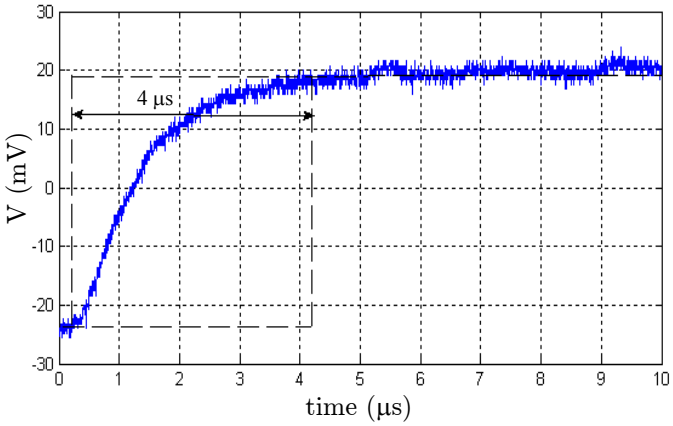


Figure 13: The step response of the VMM circuit implemented in the RASP 2.9 FPAA.

The processing speed of the analog DFT in the FPAA is limited by the settling time of the VMM circuit. Figure 13 shows the step response of the 16 × 16 VMM circuit for a size-4 analog DFT implemented in the FPAA (RASP 2.9) while the step input changes from 0.2 μA to 1.0 μA. To measure the accurate settling time of the VMM circuit, an I-V conversion circuit shown in Figure 14 was included to the programming netlist, so that the output voltage signals can be measured by a high frequency oscilloscope. It can be observed in Figure 13 that the settling time of the VMM circuit is around 4 μs, which is close to a typical OFDM symbol duration for the IEEE 802.11a/g with 64-FFT. Note that

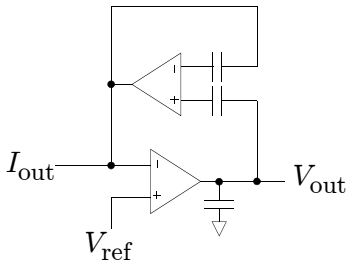


Figure 14: An on-chip I-V conversion circuit. This is attached to the output node of the VMM circuit to eliminate delays caused by the measurement setup.

the measured settling time includes additional delay caused by the auxiliary I-V conversion circuit itself, so the actual settling time of the VMM circuit will be less than the measured value. For a size-4 digital DFT implemented in the FPGA with an 8-bit data width, the data path delay is 5.3 ns for Xilinx Virtex2Pro (0.13μm CMOS process, device: XC2VP30, package: ff896, speed:-7), and 15.9 ns for Xilinx Virtex (0.22μm CMOS process, device: XCV50, package: fg256, speed:-5). However, when the number of subchannels increases, and thus the OFDM symbol duration increases, the path delay in the analog DFT stays almost the same because of the parallelized structure of the real-valued VMM operation. In [10], an analog filter implemented in a FPAA with a 0.13μm CMOS process is reported to achieve a frequency range up to 135 MHz, thus shows a potential increase in the processing speed of an analog DFT implemented in a FPAA with a smaller CMOS process.

Total power consumed in the analog 4-DFT of the FPAA is measured to be 13.4 mW. The digital 4-DFT in the Virtex2Pro FPGA required 307 mW at the maximum speed, and 118 mW at the speed same as the FPAA implementation. For the Virtex FPGA, it required 141 mW at the maximum speed, and 68 mW at the same speed as the FPAA implementation. So, the power consumption required for the 4-point DFT operation is

TABLE 1:
POWER AND DELAY COMPARISONS OF 4-POINT DFT IMPLEMENTED IN FPGA AND FPAA

Chipset	Virtex2Pro FPGA 0.13 μm CMOS process	Virtex FPGA 0.22 μm CMOS process	RASP2.9 FPAA 0.35 μm CMOS process
Power consumption @ Processing delay	307 mW @ 5.3 ns	141 mW @ 15.9 ns	13.4 mW @ 4 μs
	118 mW @ 4 μs	68 mW @ 4 μs	

The upper power consumption values for the FPGAs are measured when operating at its own fastest speed. The lower values are measured when operating at the same speed as the RASP2.9 FPAA.

significantly reduced by 9.4 dB and 7.1 dB respectively for the same speed. Table I shows the comparisons of the power and delay in the FPGA and FPAA implementations. Aside from the power saving in the DFT block itself, implementing a DFT block in an analog circuit allows the ADC to be placed after the DFT block at the receiver, thus effectively reduces the overall power consumption by relieving the speed and bit-precision requirements of the ADC block. The application-specific integrated circuit (ASIC) implementation of an analog DFT in [29] was reported to consume lower power than the FPAA implementation, where the full-radix analog 256-DFT implemented in an ASIC with a 0.18μm CMOS process was claimed to consume 1.6 mW. Despite the higher power consumption compared to the ASIC implementation, the benefit of the FPAA implementation with floating-gate transistors is its ability to tune the DFT coefficients caused by the mismatch in transistors without changing the circuit structure.

3.4.2 Equalization of FPAA Outputs

Any residual errors in the programmed weights of the DFT will prevent it from perfectly separating the symbols for the different subcarriers, leading to a form of ISI. These errors can be mitigated by applying an equalizer to each output of the analog DFT block. The equalizer coefficients can be obtained by injecting training symbols. As the errors in the programmed weights are independent with each other, each output of the equalizer has 16 taps and the 16 parallel outputs are equalized separately. So, the k-th output z_k of the equalizer (for $k = 1,...,16$) is given by the inner product $z_k = \mathbf{c}_k^T \mathbf{X}$ of the k-th equalizer coefficients vector $\mathbf{c}_k = [c_{k,1} \ \dots \ c_{k,16}]^T$ and the output vector $\mathbf{X} = [X_1 \ \dots \ X_{16}]^T$ of the DFT block.

The minimum mean-squared-error (MMSE) coefficients that minimize $MSE = E((\mathbf{c}_k^T\mathbf{X} - a_k)^2)$, where a_k is training symbols, are [52]:

$$\mathbf{c}_k = \mathbf{R}^{-1}\mathbf{p}_k\,, \tag{12}$$

where $\mathbf{R} = E(\mathbf{X}\mathbf{X}^T)$ and $\mathbf{p}_k = E(a_k\mathbf{X})$. So each of the 16 parallel outputs from the DFT block can be equalized with the 16 tap coefficients of (12). Figure 12(b) depicts the equalized symbols while using a 16×16 MMSE equalizer. It can be observed that demodulated symbols are located tighter in the I-Q map by applying equalization to the outputs of the DFT.

For a least-mean squares (LMS) equalizer, the equalizer coefficients are updated along the steepest decent direction using [52]:

$$\mathbf{c}_k[n+1] = \mathbf{c}_k[n] - \mu(z_k - a_k)\mathbf{X}[n], \tag{13}$$

where μ is the step size. Figure 15 exhibits the trace of MSE for an LMS equalizer when the step size is $\mu = 5 \times 10^{-4}$ and the initial coefficients vector for each output is set to each row of the 16×16 identity matrix. It can be seen from the figure that the convergence occurs within 500 iterations with these parameters. This iteration can be also

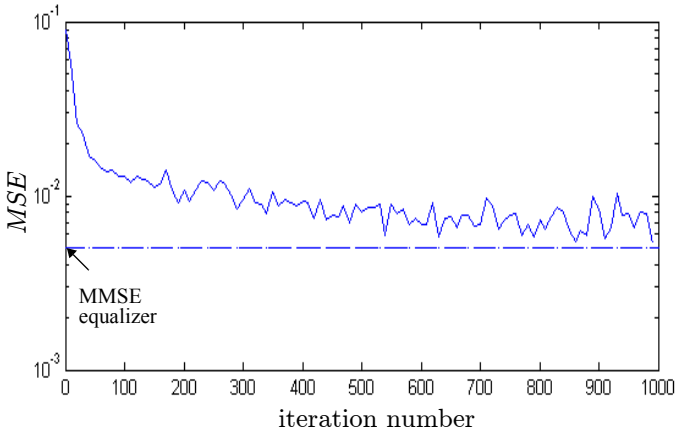


Figure 15: The MSE trace of a 16×16 LMS equalizer.

applied to an adaptive programming scheme by charging floating-gate transistors with an updated amount of injection based on the measured current level.

3.4.3 BER Performance in AWGN Channel

We now consider the case when the modulated OFDM signals are passed through a noisy channel to see how the channel noise affects the performance of the analog DFT demodulator. Note that there is a certain range of an input voltage that is allowed to be fed into the FPAA chip, effectively 0 – *V*ref, but due to the nature of high peak-to-average power ratio (PAPR) in OFDM signals, some received OFDM signals from a noisy channel can be converted to the input voltage levels beyond the allowed range. To avoid this case, the converted input voltage levels that are lower than 0 V are set to 0 V. This results in clipping distortions when the input current value is high, but it happens at rare peak voltages of the OFDM signals.

Figure 16 demonstrates the measured bit error rate (BER) versus E_b/N_0 (SNR per bit) for a 16-QAM OFDM demodulator implemented in an analog DFT, assuming additive white Gaussian noise. Also shown is the performance with MMSE and LMS equalization. These results are compared to the theoretical BER for 16 QAM with Gray mapping [13]:

$$BER \approx \frac{3}{4} Q\left(\sqrt{\tfrac{4}{5} E_b/N_0}\right). \tag{14}$$

The measurement was iterated for 2500 cycles, so the sample size is 10000 symbols or 40000 bits per each E_b/N_0 value. As can be observed from the plots in Figure 16, the demodulated OFDM symbols with an analog DFT suffer a performance penalty of 2 dB compared to the theoretical BER curve. This is because the remaining errors in the

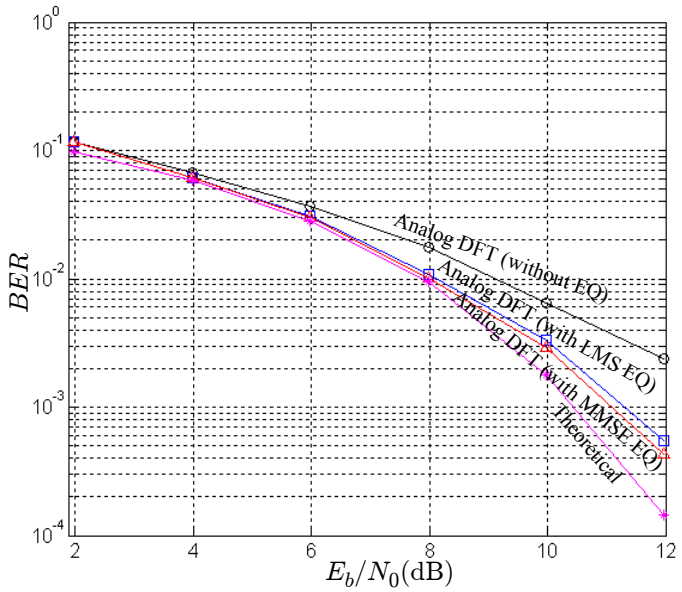


Figure 16: Performance of 16 QAM OFDM demodulator using an analog DFT, with and without equalization. When compared to theory, the penalty after equalization is less than 1 dB.

programmed weights produces an ISI across the parallel outputs of the analog DFT block. However, applying equalization to the outputs of the FPAA significantly relieves the penalty by mitigating the errors in the programmed weights, and the E_b/N_0 gap between the equalized outputs and the theoretical values becomes less than 1 dB.

For a digital DFT demodulator with an 8-bit data width, the measured BER for the same sample size converged to the theoretical values of a 16 QAM regardless of the existence of the IDFT/DFT blocks in-between. So, there is a trade-off between performance and power consumption, but the power saving of the analog circuit outweighs the performance penalty without equalization.

3.5 Summary

A low-power analog DFT was implemented in an FPAA as an alternative to a conventional OFDM demodulator based on a digital DFT. The analog DFT is configured as a VMM using floating-gate transistors. The floating-gate transistors of the FPAA are used not only to configure the VMM circuit connections as fully turned-on switches, but also to store the DFT coefficients by locking in an appropriate amount of charge in each floating-gate capacitor. The analog 4-DFT in the FPAA consumed 9.4 dB less power than a digital implementation using a Virtex2Pro FPGA, and 7.1 dB less power than a digital implementation using a Virtex FPGA. This power reduction, although significant, does not reflect the additional power savings that comes from the fact that an analog DFT reduces the speed and precision requirements of the analog-to-digital converters. The price paid for this power reduction was a 2 dB performance degradation. This performance loss can be mitigated by exploiting an equalizer technique as a top-down approach to tackle the device mismatch problem. Also, the un-quantized output signals from the analog DFT block enable the real *soft* inputs to the subsequent soft-decision decoder block.

CHAPTER 4

REDUCED-COMPLEXITY VITERBI DECODER WITH UN-QUANTIZED SOFT INFORMATION

4.1 Backgrounds

In digital communication systems, one of the most power consuming blocks is the channel decoder. In this chapter, a reduced-complexity soft-decision Viterbi decoder will be proposed. We will also discuss the performance aspect of the proposed Viterbi decoder compared to that of digital implementations.

It is well known that a soft-decision decoding gives better chance to recover originally transmitted symbols than a hard-decision decoding [13]-[15]. In fact, most of the advanced decoding schemes, such as the BCJR algorithm [18] for Turbo code [20], the soft output Viterbi algorithm (SOVA) [19], and the belief propagation for low-density parity-check (LDPC) code [16][17], all exploit the soft information in the decoding process. These channel coding schemes are applied to protect data sent over a noisy channel at the cost of sending redundant information, especially when the SNR is limited. Received signals from the channel for a low SNR contain a noise level that is large enough to make the transmitted alphabet symbols misunderstood with other alphabets. In that case, quantizing the noisy signals entails bit-flipping in the resulting bits, thus making the received noisy symbols

even more different from the originally transmitted symbols. So, it is preferable to leave the noisy continuous signals *intact*, then let the soft-decision decoder take care of the additional information on the degree of reliability conserved in the soft information.

Since an analog circuit naturally deals with continuous signals, the received noisy signals do not need to go through a quantization process, and the performance of the decoder can potentially benefit from the soft information signals flowing through the system. In a digital circuit, however, other than the performance limit mentioned earlier, each noisy symbol is a continuous number so it has to be represented in multiple bits depending on the bit-precision of the system in order to perform the soft decoding. So, it is obvious that the ADC block at the receiver needs to have a sufficiently large bit-precision, which requires higher power consumption. In wireless communication systems, this quantizing process can be considered as a redundant complexity, because the received noisy signals from the wireless channel are already continuous values, in that the degrees of reliability are conserved in themselves, thus can be directly applied to the soft-decision decoder block. With this scheme, the ADC block is placed after the channel decoder, so the demodulator and decoder blocks need to be capable of handling un-quantized analog signals. The analog OFDM demodulator using a VMM circuit was addressed in Chapter 3, and in this chapter we will discuss a soft-decision decoder that takes the continuous output from the analog OFDM demodulator as its soft input. The ADC block after the decoder is trivial, because it only requires 1-bit precision for the binary outputs from the comparator of the decoder.

There have been several analog implementations for soft-decision decoders. Hagenauer *et al*. [38] and Loeliger *et al*. [39] came up with a similar approach in which they exploited the similarity between the characteristic equations of transistors and the equations required for calculating a posteriori probability (APP) from log-likelihood ratio (LLR) values. Hagenauer *et al*. used a pair of bipolar transistors to represent the LLR in the form of a differential voltage level and the corresponding pair of APPs in the form of current levels, whereas Loeliger *et al.* used MOS transistors in a sub-threshold mode, where the equation for I-V characteristics of MOS transistors is similar to that of bipolar transistors, in the sense that they are both expressed in exponential terms. However, both implementations require Gilbert current multipliers for iterative computing of APP which make the system complexity to be high. Another approach by Mondragon-Torres *et al*. [40] used multiple-input floating-gate CMOS transistors in a sub-threshold mode to represent LLR values in voltage levels, which requires a special CMOS fabrication with multiple gates.

One of the most widely used channel decoders in wireless communication systems as well as disk storage systems and speech recognition systems is the Viterbi decoder [41]-[43]. It is the maximum likelihood (ML) decoder, meaning that it aims to determine the sequence that maximizes likelihood of the codeword based on the observed signals. The soft-decision Viterbi decoder is named as is to indicate the observed signal and the information flowing through the system are continuous. For an AWGN channel with a bipolar mapping, the likelihood of a codeword is expressed in a Gaussian distribution form, so the ML decoder becomes equivalent to the minimum Euclidean distance decoder [15]. In this case, the computation required for the updated path metric becomes additive, which

gives less system complexity than a multiplicative path metric. This means the multipliers for iterative computing of APP in [38][39] are not required. Note that we are dealing with a quantity of probability for the path metric, and the AWGN itself has the continuous probability distribution, so it is preferable to detect the received noisy signals in an analog way without involving quantization errors.

Some previous works in implementing analog Viterbi decoders have been reported. Acampora *et al*. [44] suggested an analog Viterbi decoder using sample-and-hold circuits and voltage adders to store and update the path metrics. In their work, several nonlinear effects in the voltage-mode analog circuit have been addressed, such as amplifier voltage offset, loop gains differing from unity, and nonlinear compression within analog devices. Demosthenous *et al*. [45] realized the minimum Euclidean distance decoder in a current-mode analog circuit, where they used a switched-capacitor circuit as a front-end sample-and-hold block to store the current value that represents the previous path metric. However, their circuit requires an additional block to compensate for accumulated path metric errors caused by the signal-dependent charge injection in a switched-current circuit, that is especially problematic for a recursive process [46]. Similarly, He *et al*. [47] implemented the minimum Hamming distance decoder with a current-mode analog circuit based on a switched-capacitor and a winner-take-all circuit. Since they considered a Hamming distance, it is a hard-decision Viterbi decoder, thus it does not take advantage of having continuous signals. Although their work did not address actual measurement data for performance, the current-mode switched-capacitor and the winner-take-all circuits used to store and compare the current values usually suffer from device mismatch and nonlinearity in analog components. More recently, Maunu *et al*. [48] suggested an analog Viterbi

decoder for an OFDM ultra-wideband (UWB) receiver to increase the decoding speed and to reduce power consumption. However, it requires additional DAC between the digital demodulator and the analog Viterbi decoder of the receiver, so as to bring the digitally demodulated OFDM signals back to analog. In our work, however, this additional conversion is not required as the input signals to the Viterbi decoder are provided by a preceding analog DFT described in Chapter 3. The Viterbi decoder can also be applied to a partial-response maximum likelihood (PRML) channel coding scheme for disk storage systems [49], when signals are corrupted by intersymbol interference (ISI). The analog realization of the PRML has also shown power saving mainly due to the elimination of ADCs that are commonly chosen as a power-greedy flash architecture for the high speed requirement [50][51].

In order to circumvent any potential sources of nonlinear effects that previous works suffered, we propose a soft-decision Viterbi decoder that makes use of trivial operations in each signal mode. Also, we will deliver systematic discussions on the benefits of this analog approach from complexity and performance point of view.

4.2 System Description and Analysis

4.2.1 Soft Information

Soft Information contains the degree of *reliability* in each received signal. As a simple example, with a BPSK alphabet 1 and −1, if a received signal from an AWGN channel is 0.9, it is more *likely* that a transmitted symbol is 1 than −1. This is because the probability density function (pdf) of a Gaussian noise gets smaller as a noise level gets large, so it is

statistically less probable to make an original symbol −1 to become 0.9 by adding a noise level of 1.9 than to make a symbol 1 to 0.9 with a noise level of −0.1. However, if a received signal is in the vicinity of a threshold level, say 0.1, the difference between the likelihood gets closer, and it is uncertain whether the transmitted symbol was −1 or 1. In either case, we are sure that the transmitted symbol is disturbed by a substantial amount of noise. So, stochastically speaking, we can claim a received signal of 0.9 is more reliable than a received signal of 0.1.

This degree of reliability in each received signal gets even more crucial when it come to a sequence detector, where symbols in the sequence are correlated with each other. Since a decoding process basically is to determine a *sequence* of symbols that maximizes the *joint* probability of each symbol among valid sequences, it is a better strategy to weight more on reliable symbols in choosing the most likely sequence of symbols from valid codewords.

We can find many intuitive examples of soft information in a real life. For example, if you are solving Sudoku or crossword puzzles, you may want to first start from questions that you are more certain about the answers, then let the blanks for uncertain questions be filled in as the process evolves. Another example can be found at a TV quiz show, 'Who wants to be a millionaire'. When a player is uncertain about his answer, he has a chance to call his friend to ask for the answer. The next thing you often see is that the player ask his friend how confident he is with his answer. This additional information on the reliability actually helps to get the correct answer, because a player can weigh between his own answer with his confidence and the friend's answer with his confidence.

In order to illustrate the concepts of reliability and correlated sequence in the context of communication systems, consider the following two questions:

- Question 1. Who is the 7th President of the United States?

A. Andrew Jackson

B. Martin Buren

- Question 2. Who is the 7th President of the United States?

A. Andrew Jackson

B. Michael Jackson

Both questions asks for the same answer but with different options. In Question 1, you may not be familiar with both of the given options, so it is not too clear which one is the correct answer. In Question 2, you may still not be familiar with the option *A*, but it seems the option *B* is not the answer. So, a reasonable response in this case is to choose option *A* for your answer. However, there might be a President with the same name as the famous popstar among 44 U.S. Presidents, so you may still suspect the answer. If you are to answer for this question only, the best choice is *A*. On the other hand, if you are to answer for several questions, and you somehow know the total number of *A*'s in the answers (similar to a parity bit), a better strategy is to keep this question unanswered until the last question as additional information becomes available. Analogies to these questions in communication systems are shown in Figure 17. In Figure 17(a), the received sample resides in the vicinity of threshold, so it is unclear whether the originally transmitted symbols was the higher alphabet or the lower one, thus it can be claimed as an unreliable

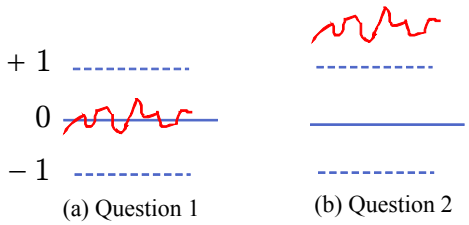


Figure 17: Analogies to Questions 1 and 2 in the context of received signals in communication systems.

sample. In Figure 17(b), the received sample is above the higher alphabet, so it is less likely that the lower alphabet was transmitted than the higher one. However, the received sample holds a substantial *distance* from the higher alphabet as well, which corresponds to the *suspicion* we had in solving Question 2. This distance is quantified as a Euclidian distance that is used as a cost function in a decoding process.

4.2.2 Maximum-Likelihood Sequence Detection

The aim for a decoding process is to determine the sequence of symbols that maximizes the joint a posteriori probability (APP), or the joint conditional probability, of transmitted symbols given the received symbols. A simplified flow of baseband signals with a $1/R$ rate encoder and an M-ary symbol mapper is shown in Figure 18. A transmitted symbol vector $\underline{t}$ and a received symbol vector $\underline{r}$ from a noisy channel are related by

$$\underline{r} = \underline{t} + \underline{n}, \tag{15}$$

where $\underline{n}$ is a noise vector. The maximum a posteriori probability (MAP) criterion for determining the sequence of codewords $\underline{c}$ based on the observation of received symbols is expressed as

$$\arg \underset{\underline{c}}{Max} \left[\prod_k \prod_i P(c_k^{(i)} | \underline{r}) \right], \tag{16}$$

where $c_k^{(i)}$ is the k-th transmitted codeword (i=1,..,R) and $\underline{r}$ is the received symbol that contains the corresponding portion of the k-th codeword. Applying Bayes' rule, the criterion (16) becomes

$$\arg\max_{\underline{c}} \left[\prod_k \prod_i \frac{P(\underline{r}|c_k^{(i)})P(c_k^{(i)})}{P(\underline{r})} \right]. \tag{17}$$

Since $P(\underline{r})$ does not affect the choice of $\underline{c}$ that maximizes the joint APP, (17) can be simplified to

$$\arg\max_{\underline{c}} \left[\prod_k \prod_i P(\underline{r}|c_k^{(i)})P(c_k^{(i)}) \right]. \tag{18}$$

With a common assumption made in communication systems that source codes are equally probable, that is, a priori probability $P(c_k^{(i)}=1)=P(c_k^{(i)}=0)=\frac{1}{2}$, (18) becomes

$$\arg\max_{\underline{c}} \left[\prod_k \prod_i P(\underline{r}|c_k^{(i)}) \right], \tag{19}$$

which is now converted to the maximum likelihood (ML) criterion [15].

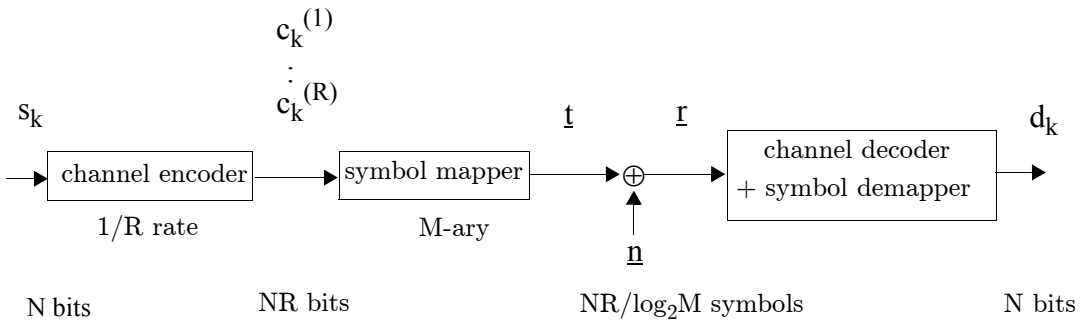


Figure 18: Simplified flow of baseband signals with a 1/R rate encoder and M-ary symbol mapper.

For an AWGN channel with a bipolar mapping, $t_k^{(i)} = 2c_k^{(i)} - 1$ and $\underline{n} \sim N(0,\sigma^2)$. So, the likelihood of the k-th transmitted codeword $c_k^{(i)}$ (i=1,...,R) is given by

$$P(r_k^{(i)} | c_k^{(i)}) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{ - \frac{(r_k^{(i)} - t_k^{(i)})^2}{2\sigma^2} \right\}, \tag{20}$$

and the ML criterion becomes

$$\begin{aligned}
&\underset{\underline{c}}{\arg} \; Max\left[\prod_k \prod_i P(r_k^{(i)} | c_k^{(i)}) \right] \\
&= \underset{\underline{c}}{\arg} \; Max\left[\prod_k \prod_i \exp\left\{ - \frac{(r_k^{(i)} - t_k^{(i)})^2}{2\sigma^2} \right\} \right] \\
&= \underset{\underline{c}}{\arg} \; Max\left[\exp\left\{ - \sum_k \sum_i \frac{(r_k^{(i)} - t_k^{(i)})^2}{2\sigma^2} \right\} \right] \\
&= \underset{\underline{c}}{\arg} \; Min\left[\sum_k \sum_i (r_k^{(i)} - t_k^{(i)})^2 \right] ,
\end{aligned} \tag{21}$$

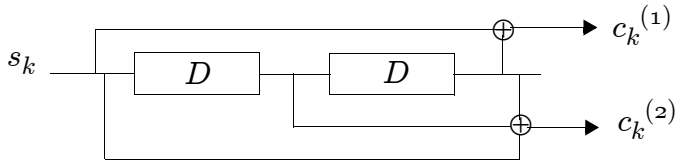
which is equivalent to the minimum Euclidian distance criterion [15]. The Viterbi algorithm is known to perform well in decoding a sequence of transmitted codewords based on the minimum Euclidian distance criterion [13]-[15]. Note that the quantity we are using in determining the most likely sequence is the joint conditional probability of received signals, each of which follows the Gaussian distribution same as the AWGN with a shift in the mean value by $t_k^{(i)}$. As the Gaussian distribution spans over a continuous range for a random variable (input signals) with a continuous range for a probability density (branch metrics), it is natural to represent input signals, branch metrics, and path metrics — sum of branch metrics — all in continuous values.

4.2.3 The Viterbi Algorithm

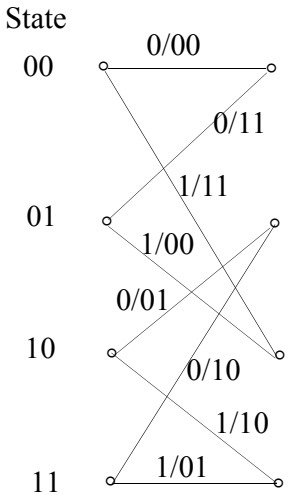
We consider a 1/2 rate convolutional code with 4 states. The corresponding encoder and the trellis diagram are given in Figure 19, where s_k is a source encoded bit and $c_k^{(1)}$, $c_k^{(2)}$ are channel encoded bits. The boxes with the letter *D* (delay) represent shift registers, and $\oplus$ represents an exclusive-OR operation. The generator polynomial of the encoder is given as [7, 5] octal. Each encoded bit is affected by bits in 3 different stages, so the constraint length *K* of the encoder is 3.

The channel encoded bits are mapped to BPSK symbols, and they are transmitted through an AWGN channel. On the receiver side, the Viterbi decoder estimates the sequence of transmitted codewords based on the sequence of observed noisy symbols, so that the decoded bit sequence is as close as possible to the source encoded bit sequence. The Viterbi decoder consists of a branch metric unit (BMU), an add-compare-select unit (ACSU), and a trace-back unit (TBU). The BMU is to compute the sum of the squared Euclidean distance from the received symbol r_k to the bipolarly mapped codeword y_k. Since $\sum_{i=1}^{2}(r_K^{(i)})^2$ and $\sum_{i=1}^{2}(y_K^{(i)})^2$ terms in the squared Euclidean distance are common for all branches [53], (21) becomes $\underset{\underline{c}}{\arg}\ Max\left[\sum_{K}\sum_{i=1}^{2} r_K^{(i)} y_K^{(i)}\right]$. So, the branch metric is defined as

$$b_{y_K^{(1)} y_K^{(2)}}(k) = \sum_{i=1}^{2} r_K^{(i)} y_K^{(i)}, \tag{22}$$



(a)



(b)

Figure 19: (a) A 1/2 rate convolutional encoder with 2 shift registers, and (b) the corresponding trellis diagram with 4 states.

where $y_k^{(i)}$ is either 1 or –1, and it is fixed for a given trellis. So, the only operations required for computing the branch metrics are sign inversions and additions. The ACSU computes the path metric by adding a branch metric to the previous path metric, then compares two path metrics to determine the selected path. That is, the path metric at each state is given by

$$P_{00}(k) = Max\{ b_{11}(k) + P_{01}(k-1), b_{-1-1}(k) + P_{00}(k-1) \}, \quad (23)$$

$$P_{01}(k) = Max\{ b_{1-1}(k) + P_{11}(k-1), b_{-11}(k) + P_{10}(k-1) \}, \quad (24)$$

$$P_{10}(k) = Max\{ b_{11}(k) + P_{00}(k-1), b_{-1-1}(k) + P_{01}(k-1) \}, \quad (25)$$

$$P_{11}(k) = Max\{ b_{1-1}(k) + P_{10}(k-1), b_{-11}(k) + P_{11}(k-1) \}, \quad (26)$$

where b_{-1-1}, b_{-11}, b_{1-1}, and b_{11} represent branch metrics for the bipolarly mapped codewords -1-1, -11, 1-1, and 11, respectively, and P_{oo}, P_{o1}, P_{1o}, and P_{11} represent the path metrics of the state 00, 01, 10, and 11, respectively. The TBU follows the survived path in the opposite direction to determine the decoded bit sequence.

4.2.4 Proposed Soft-Decision Viterbi Decoder

In order to circumvent any potential sources of nonlinear effects that previous analog implementations suffered, we propose a soft-decision Viterbi decoder that makes use of trivial operations in each signal mode. That is, adding operations are performed in a current-mode, whereas copying, comparing, and sample-and-hold operations are performed in a voltage-mode. This also enables a highly parallelized structure for adding signals from multiple sources and propagating signals to multiple destinations, as illustrated in the information flow in Figure 20. The sign-inversions occur during the I-V or V-I conversions, if needed. The circuit configuration of the proposed BMU/ACSU is

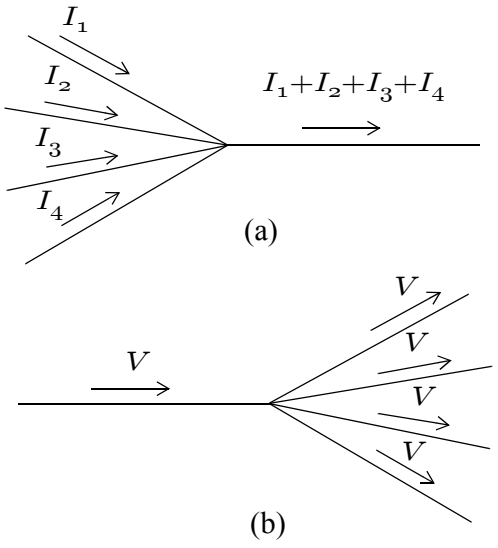


Figure 20: Parallelized flow of information for (a) adding operations in current-mode, and (b) copying operations in voltage-mode.

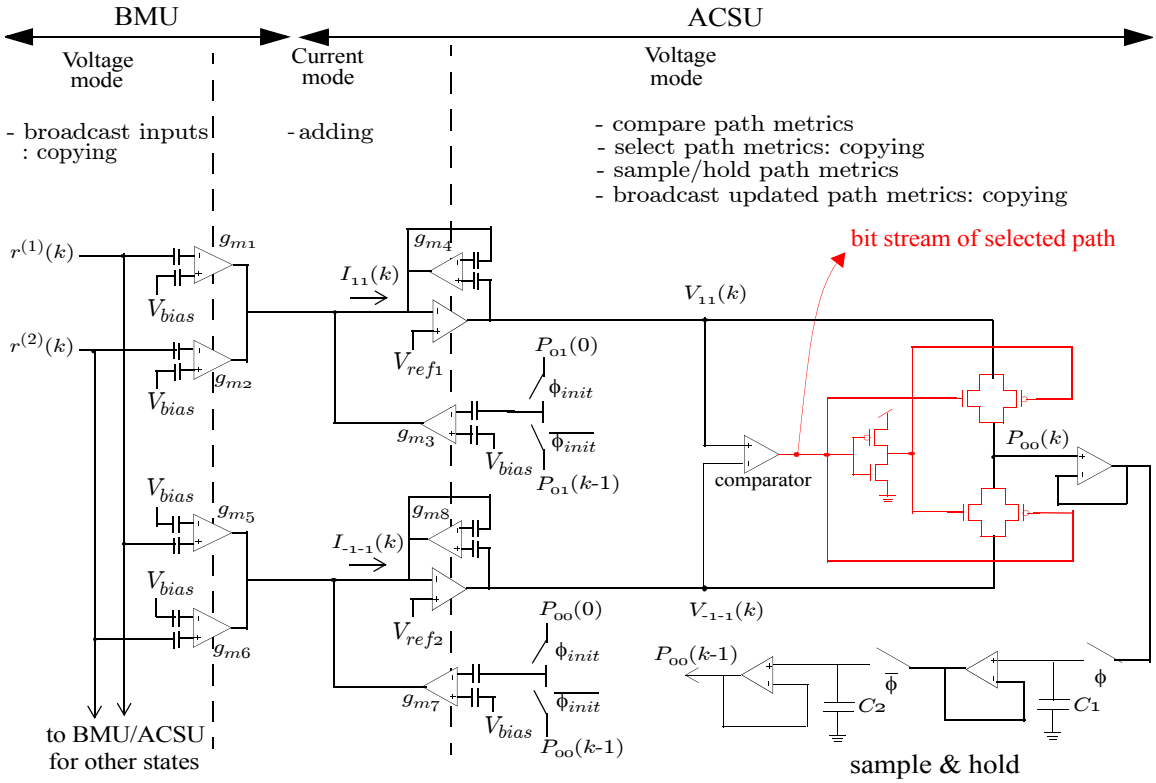


Figure 21: BMU/ACSU circuit configuration for state 00. Input signals and path metrics are represented in voltage levels, whereas branch metrics are represented in current levels. Floating-gate OTAs are used as linear V-I or I-V converters. Initial path metrics are applied to the feedback inputs when ϕ_{init} is high in order to force trellis to start from state 00. The updated path metrics are output-buffered before getting fed into the sample-and-hold circuit. When ϕ is high, the voltage stored in the capacitor C_2 is applied to the feedback input and the capacitor C_1 is charged with the updated path metric. When ϕ is low, the voltage stored in the capacitor C_1 is relayed to the capacitor C_2. The signal flowing through the red-colored part is binary, and the output bit stream from the comparator is stored in a digital memory to be used in the trace-back process.

given in Figure 21. Here, the input signals and the path metrics are represented in voltage levels, and the branch metrics are represented in current levels. The floating-gate OTA (FG-OTA) is a voltage-controlled current-source, so it can be used as a linear V-I or I-V converter, with the relation of

$$I_{out} = g_m (V_{in+} - V_{in-}), \tag{27}$$

where I_{out} is the output current, g_m is the transconductance, and V_{in+}, V_{in-} are differential input voltages applied to positive and negative terminals of the FG-OTA.

Sign-inversions can be realized by swapping the two input signals of the terminals. The FG-OTA has the same schematic as the OTA in Figure 6 but has floating-gate input terminals. Notice that FG-OTAs are chosen over OTAs for the converters, because they have lower transconductance than OTAs due to the lower κ, thus provide wider ranges for the linear V-I or I-V conversion. On the contrary, OTAs are more suitable for comparators due to the high transconductance in order to provide steeper transition as two input signals vary. The summation of the output current from each V-I converter is given by

$$I_{11}(k)=\{V_{bias}-r^{(1)}(k)\}g_{m1}+\{V_{bias}-r^{(2)}(k)\}g_{m2}+\{V_{bias}-P_{01}(k-1)\}g_{m3}. \quad (28)$$

Note that we now have mapped received signals $r_k^{(i)}$ to input voltages $r^{(i)}(k)$ to cope with dynamic ranges of the OTAs. Because of the virtual short at the buffering OTA of the I-V converter, the output voltage from the I-V converter is

$$V_{11}(k)=-\frac{I_{11}(k)}{g_{m4}}+V_{ref1}$$

$$=\{r^{(1)}(k)-V_{bias}\}\frac{g_{m1}}{g_{m4}}+\{r^{(2)}(k)-V_{bias}\}\frac{g_{m2}}{g_{m4}}+\{P_{01}(k-1)-V_{bias}\}\frac{g_{m3}}{g_{m4}}+V_{ref1}. \quad (29)$$

Similarly,

$$I_{-1-1}(k)=\{r^{(1)}(k)-V_{bias}\}g_{m5}+\{r^{(2)}(k)-V_{bias}\}g_{m6}+\{V_{bias}-P_{00}(k-1)\}g_{m7}, \quad (30)$$

so

$$V_{-1-1}(k)=-\frac{I_{-1-1}(k)}{g_{m8}}+V_{ref2}$$

$$=\{V_{bias}-r^{(1)}(k)\}\frac{g_{m5}}{g_{m8}}+\{V_{bias}-r^{(2)}(k)\}\frac{g_{m6}}{g_{m8}}+\{P_{00}(k-1)-V_{bias}\}\frac{g_{m7}}{g_{m8}}+V_{ref2}. \quad (31)$$

From (29) and (31), it is obvious that if there are mismatch among g_m values, the slope of output versus input voltages get skewed and the offset value changes. Besides, there are additional offsets when the two input terminals of the FG-OTAs are not balanced out, thus cause different offsets for different buffering OTAs. However, the deviation in offset values can be mitigated by adjusting the independent reference voltage of the buffering OTA in each I-V converter. Alternatively, we can adjust the charge stored in the floating-gate capacitors at two input terminals of the FG-OTA to combat this imbalance, because the voltage drop across the capacitor is $V_c = Q/C$, which is proportional to the charge stored in the capacitor. Thus, the actual gate voltage applied to the FG-OTA is the summation of the input voltage and the voltage drop V_c. Also, the transconductance $g_m = \kappa I_{bias}/2U_T$ is proportional to the bias current of each FG-OTA, so the skew in slopes can be mitigated by programming the bias currents (determined by the charge stored in the floating-gate capacitor connected to the pMOSFET inside the FG-OTA) of the I-V converting FG-OTAs first, then programming the bias current of the V-I converting FG-OTAs, while targeting on the desired slope of the output versus input voltages. Here, the order of programming is as stated and not the other way around, because each I-V converting FG-OTA has three different V-I converting FG-OTAs connected. Mismatch in g_m values among different I-V converting FG-OTAs should not be an issue as well, as this programming procedure cancels out the g_m term in each BMU-ACSU suit. So, within tolerable skews in the slopes, we can neglect the mismatch in g_m values, and hence (29) and (31) become

$$V_{11}(k) = \{r^{(1)}(k) - V_{bias}\} + \{r^{(2)}(k) - V_{bias}\} + \{P_{01}(k-1) - V_{bias}\} + V_{ref1}, \quad (32)$$

$$V_{-1-1}(k) = \{V_{bias} - r^{(1)}(k)\} + \{V_{bias} - r^{(2)}(k)\} + \{P_{00}(k-1) - V_{bias}\} + V_{ref2}, \quad (33)$$

that correspond to each argument of the right-hand side in (23) for the cases $y_k^{(1)} = y_k^{(2)} = 1$ and $y_k^{(1)} = y_k^{(2)} = -1$, respectively. For input voltages with the dynamic ranges of $V_{bias}-\Delta V \le r^{(1)}(k) \le V_{bias}+\Delta V$ and $V_{bias}-\Delta V \le r^{(2)}(k) \le V_{bias}+\Delta V$, the output voltages reside in the range of $V_{ref1}-2\Delta V \le V_{11}(k) \le V_{ref1}+2\Delta V$ and $V_{ref2}-2\Delta V \le V_{-1-1}(k) \le V_{ref2}+2\Delta V$ when initial path metrics are set to V_{bias}.

The following comparator yields a stream of binary decisions for selected paths. Note that these binary decisions require only 1-bit precision. Thus, we can abstract out *bits* from continuous signals without any quantization involved in the system, while keeping the inputs, branch metrics, and path metrics to be *soft* throughout the entire iterative decoding process. The selector unit shown as the red-colored part in Figure 21 can be viewed as a digital circuit, in the sense that the signal flowing through the circuit is binary, which makes the BMU/ACSU circuit to be a mixed-signal circuit. Depending on the comparator output, the subsequent CMOS inverters drive the selected output voltage to the updated output. That is,

$$P_{00}(k) = Max[\{r^{(1)}(k) - V_{bias}\} + \{r^{(2)}(k) - V_{bias}\} + \{P_{01}(k-1) - V_{bias}\} + V_{ref1}, \\ \{V_{bias} - r^{(1)}(k)\} + \{V_{bias} - r^{(2)}(k)\} + \{P_{00}(k-1) - V_{bias}\} + V_{ref2}]. \quad (34)$$

This selecting process is basically a copying operation in a voltage-mode, so it does not suffer from nonlinear effects as in the current-mode winner-take-all circuit. The path metrics for other states are obtained in the same manner, and can be expressed as

$$P_{01}(k) = Max[\{r^{(1)}(k) - V_{bias}\} + \{V_{bias} - r^{(2)}(k)\} + \{P_{11}(k-1) - V_{bias}\} + V_{ref5}, \\ \{V_{bias} - r^{(1)}(k)\} + \{r^{(2)}(k) - V_{bias}\} + \{P_{10}(k-1) - V_{bias}\} + V_{ref6}], \quad (35)$$

$$P_{10}(k)=Max[\{r^{(1)}(k)-V_{bias}\}+\{r^{(2)}(k)-V_{bias}\}+\{P_{00}(k-1)-V_{bias}\}+V_{ref3},$$
$$\{V_{bias}-r^{(1)}(k)\}+\{V_{bias}-r^{(2)}(k)\}+\{P_{01}(k-1)-V_{bias}\}+V_{ref4}], \quad (36)$$

$$P_{11}(k)=Max[\{r^{(1)}(k)-V_{bias}\}+\{V_{bias}-r^{(2)}(k)\}+\{P_{10}(k-1)-V_{bias}\}+V_{ref7},$$
$$\{V_{bias}-r^{(1)}(k)\}+\{r^{(2)}(k)-V_{bias}\}+\{P_{11}(k-1)-V_{bias}\}+V_{ref8}], \quad (37)$$

that correspond to (23)-(26), respectively.

The updated output at each state is output-buffered and fed-back to the input for the next stage to yield accumulated path metrics. This can be realized by a sample-and-hold circuit shown at the right-bottom in Figure 21. Note that each updated output is copied to two different states as seen in (34)-(37), so the voltage mode is chosen for this operation. When ϕ is high, the voltage stored in the capacitor C_2 is applied to the feedback input and the capacitor C_1 is charged with the updated path metric. When ϕ is low, the voltage stored in the capacitor C_1 is relayed to the capacitor C_2.

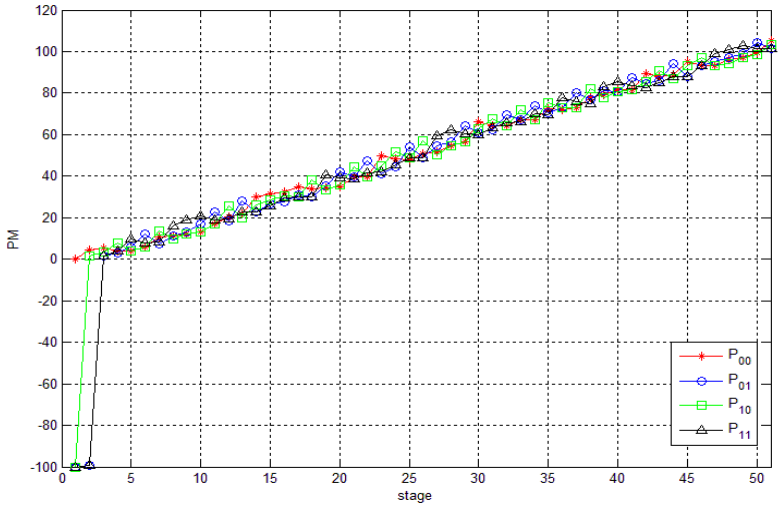


Figure 22: Simulated trace of path metrics in each state. The path metrics tends to increase linearly during the iterative decoding process.

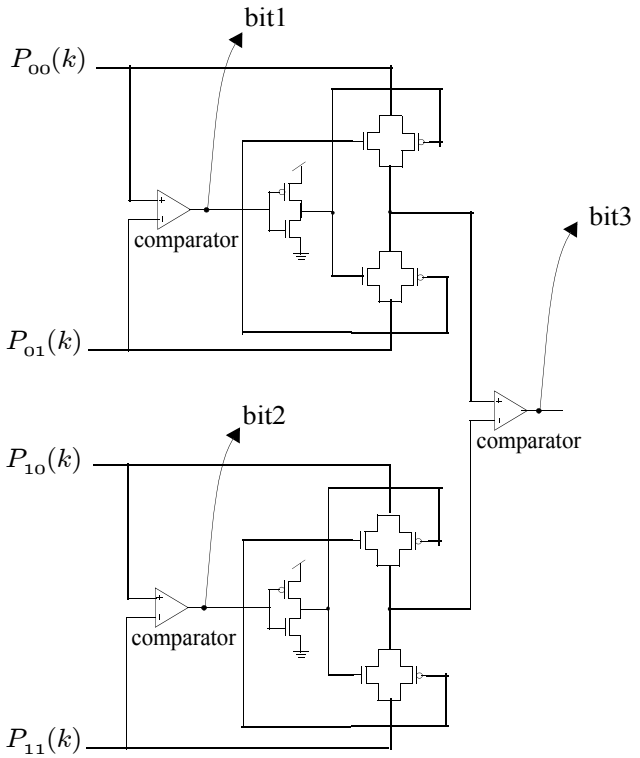


Figure 23: Auxiliary comparators and selectors. The two-bit decisions composed of (bit1,bit3) and (bit2,bit3) are handed over to the trace-back process to indicate the starting point for each sliding window.

During the iterative process, the updated path metrics tend to increase linearly with respect to the number of iterations as simulated in Figure 22. So, in order to keep the signals flowing through the system within the dynamic range of each circuit component, we need to subtract out a constant from the path metric at each state to avoid an overflow. This can be easily accounted for by lowering down the reference voltages of the buffering OTAs in the I-V converters.

In the course of the ACS process, the output bits from the comparator are stored in a digital memory for a decoding window size (typically determined at 5 times of the constraint length), so that they can be used to determine the selected paths during the trace-

back process. However, it is noteworthy that the proposed configuration do not require to store any intermediate information flowing through the system, such as input signals, branch metrics, and path metrics. Noticing that a system composed of a convolutional encoder, a noisy channel, and a Viterbi decoder is a hidden Markov model (HMM), this complies with the general property of the Markov model — the conditional probability distribution of future states of the process depend only upon the present state. In this way, the updated path metric values are kept to be soft until the last stage is reached. This is beneficial from both performance and power consumption perspective, as it enables to take full advantage of the soft information, at the same time it does not require any storing/ fetching process to access a digital memory in each iteration.

To determine the appropriate starting state of the trace-back process for each sliding window, the auxiliary comparators and selectors shown in Figure 23 are appended at the output of the each BMU/ACSU suit. Based on the combination of two-bit decisions, the trace-back unit can determine the starting state out of the four states. Again, these decision bits require only 1-bit precision, and do not need to be stored. Figure 24 depicts the entire block diagram of BMU/ACSU suit for each state, along with auxiliary CSU blocks appended to the updated outputs.

Once the ACS process is finished, the trace-back process is conducted in a digital domain. Despite the fact that we mainly focus on analog signal processing blocks in this work, it is worthwhile to briefly mentioned that this process can be performed with a low-complexity circuit compared to the BMU or ACSU blocks. Since the most recently updated bit of the shift registers for each state indicates the input bit that brought the trellis path to

the state, we can simply take the first bit of the state as the decoded bit while tracing back. So, the decoded bits can be obtained by keeping tack of which state the trace-back pointer falls into in a reverse state machine of the trellis shown in Figure 25.

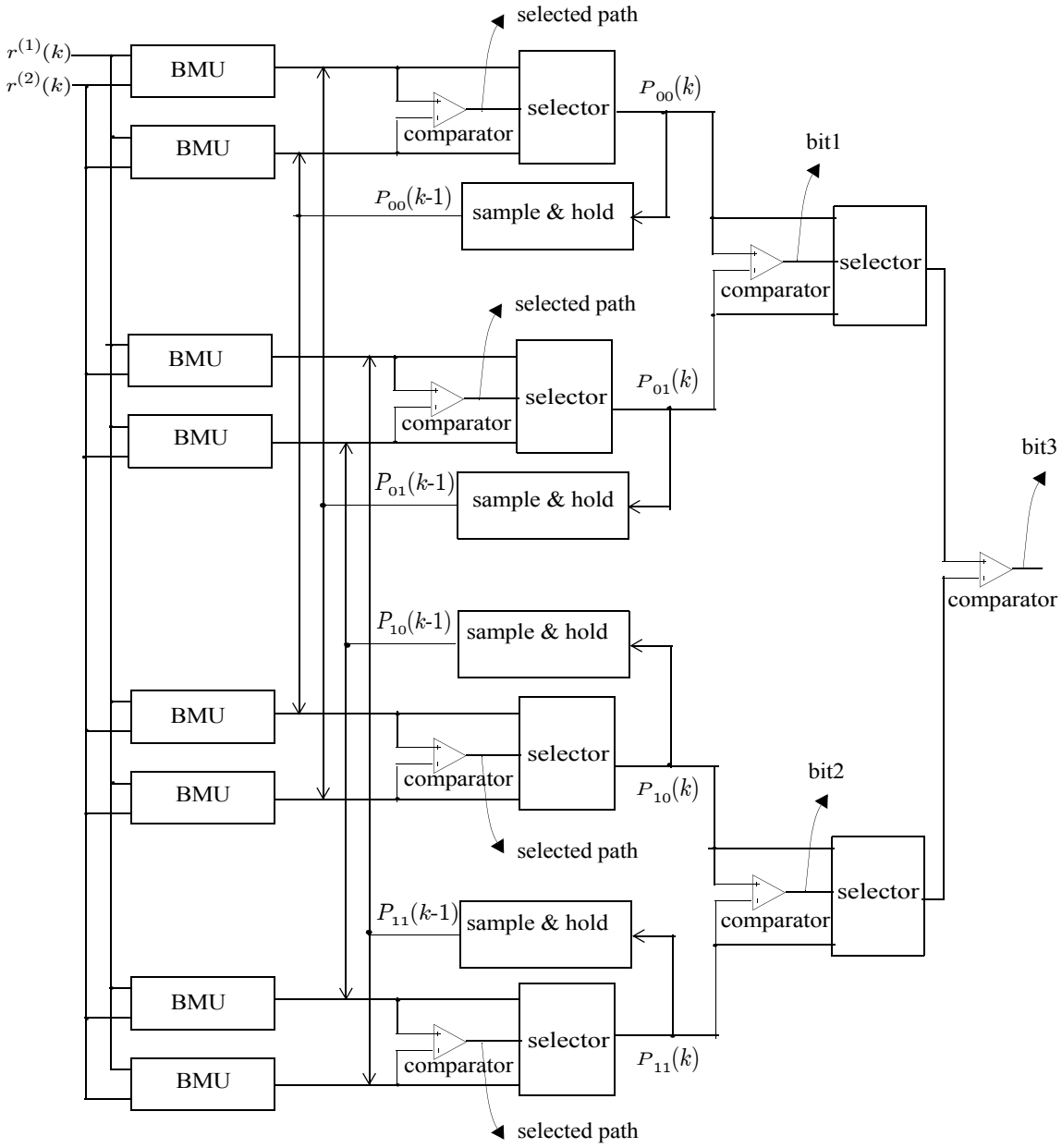


Figure 24: Block diagram of BMU/ACSU suit for each state, along with auxiliary CSU appended to the updated outputs.

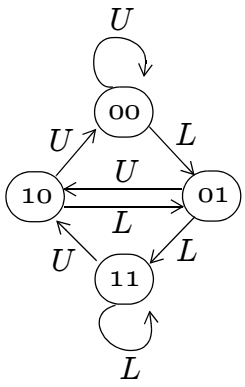


Figure 25: Reverse state machine for trace-back process. U and L indicate upper and lower path, respectively.

4.3 Soft-Decision Viterbi Decoder Implementation in FPAA

4.3.1 Programming Procedure

The netlist given in Appendix B shows the circuit configuration shown in Figure 24 for the states 00, 01, 10, and 11. Note that the updated path metrics from the ACSUs (Vp_00, Vp_01, Vp_10, Vp_11) and the previous path metrics from the sample-and-hold circuits (Vp_00_sh, Vp_01_sh, Vp_10_sh, Vp_11_sh) are pinned-out in the netlist just for the testing purpose, and they are not required to be accessible externally. Using the RASPER tool [35], this netlist is converted to a list of switch addresses and the targeting current value for each switch. This list is loaded by the Matlab script to be programmed into the FPAA chip. In practice, there are device mismatch among different components inside the FPAA, so the programmed charge of OTAs can be different with each other. The match between different devices can yield multiplicative or additive errors in signal behaviors. However, this inherent mismatch is determined during the fabrication process, so it becomes deterministic characteristics of each fabricated component, thus can be mitigated by an appropriate counteraction. Due to the programmable property of the FPAA with

floating-gate switches, this counteraction can be executed after the fabrication process by adjusting the charge stored in each floating-gate capacitor.

4.3.2 Mismatch Compensation

Similar to the programming procedure discussed in Chapter 3.3, the programming procedure for the Viterbi decoder also consists of two stages to account for the mismatch. In the first stage, fully turned-on switches and the bias current of FG-OTAs at the I-V converters are programmed with targeting currents. In the second stage, the bias current of FG-OTAs at the V-I converters are programmed while targeting on the slope of the output voltage versus input voltage. This tuning process is done by iteratively injecting a small amount of charge into the floating-gate capacitor at a time until the slope reaches the desired value. The amount of electron injected into the floating-gate capacitor can be controlled by varying the time interval the capacitor is exposed to an internal voltage supply.

4.4 Measurement Results

4.4.1 Measuring Offsets

Although the multiplicative mismatch was compensated for by adjusting charge value stored in each FG-OTA, there still remain additive mismatch due to different offsets in different FG-OTAs when two input terminals of the FG-OTAs are not balanced out. However, these offsets can be calibrated by adjusting reference voltages of the connected buffering OTAs at the I-V converters. As can be seen in Figure 21, each buffering OTA has three input stage FG-OTAs connected to it. Because any offset in each input stage FG-

OTA can only contribute to shifting a DC offset of the output from the BMU-ACSU block, it is enough to account for the sum of offsets in the three input FG-OTAs, and not the individual offset per each FG-OTA.

The calibration process can be conducted by injecting different set of input vectors through the BMU-ACSU blocks, such that one of the two candidates are selected to contribute to the output from each BMU-ACSU block. For example, by applying input signals $V_{in1} = V_{bias} + V_{test}$ ($0 \le V_{test} \le V_{bias}$), $V_{in2} = V_{bias}$, and applying the previous PM inputs $VP_{00} = VP_{01} = V_{bias}$, while setting all the initial reference voltage at V_{ref}, the calibrated reference voltage at the upper path in Figure 21 can be obtained from (34) by

$$V_{ref1} = 2V_{ref} + V_{test} - V_{out1}, \tag{38}$$

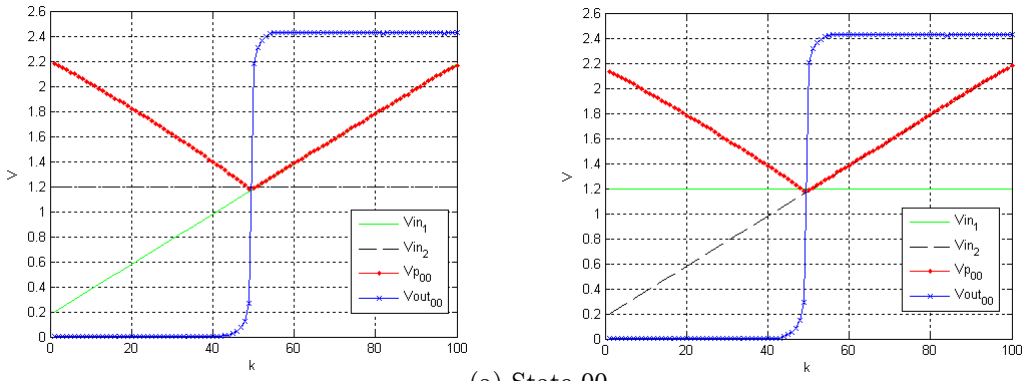
where V_{out1} is the measured output for the set of input vectors. Similarly, with $V_{in1} = V_{bias} - V_{test}$, $V_{in2} = V_{bias}$, and $VP_{00} = VP_{01} = V_{bias}$, while setting all the initial reference voltage at V_{ref}, the calibrated reference voltage at the lower path is obtained as

$$V_{ref2} = 2V_{ref} + V_{test} - V_{out2}, \tag{39}$$

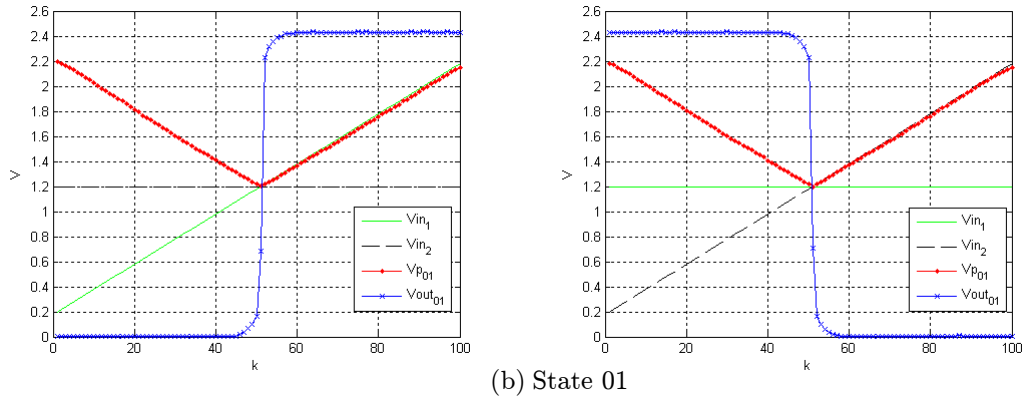
where V_{out2} is the measured output for the set of input vectors. The rest of the calibrated reference voltages for other states can be obtained in the same manner.

4.4.2 BMU-ACSU Computation

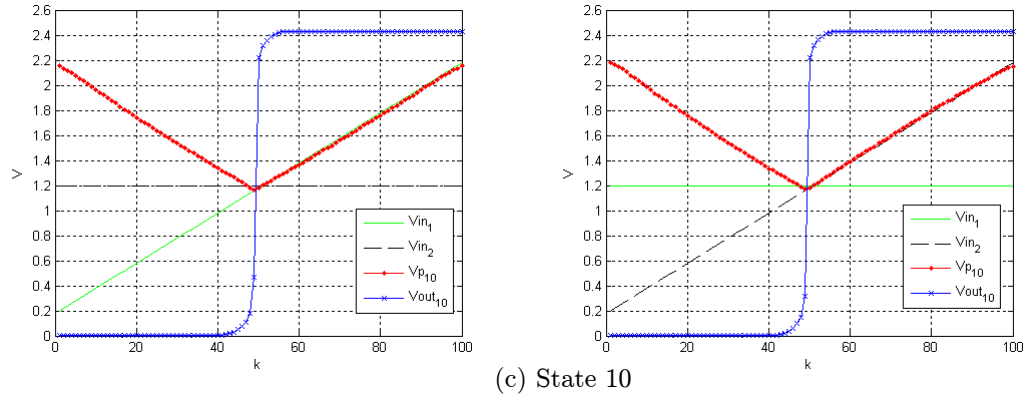
Once the multiplicative and additive mismatch are mitigated, we can now verify the functionality of the BMU-ACSU blocks. Figure 26(a)-(d) show the measured outputs from the BMU-ACSU block at each state implemented in the FPAA, while sweeping one input from 0.2V to 2.2V and keeping the other input at $Vbias$ = 1.2V. The previous path metrics



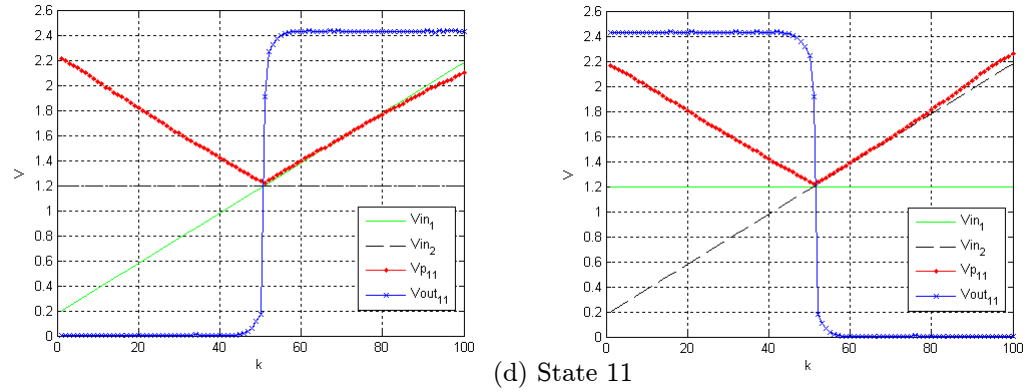
(a) State 00



(b) State 01



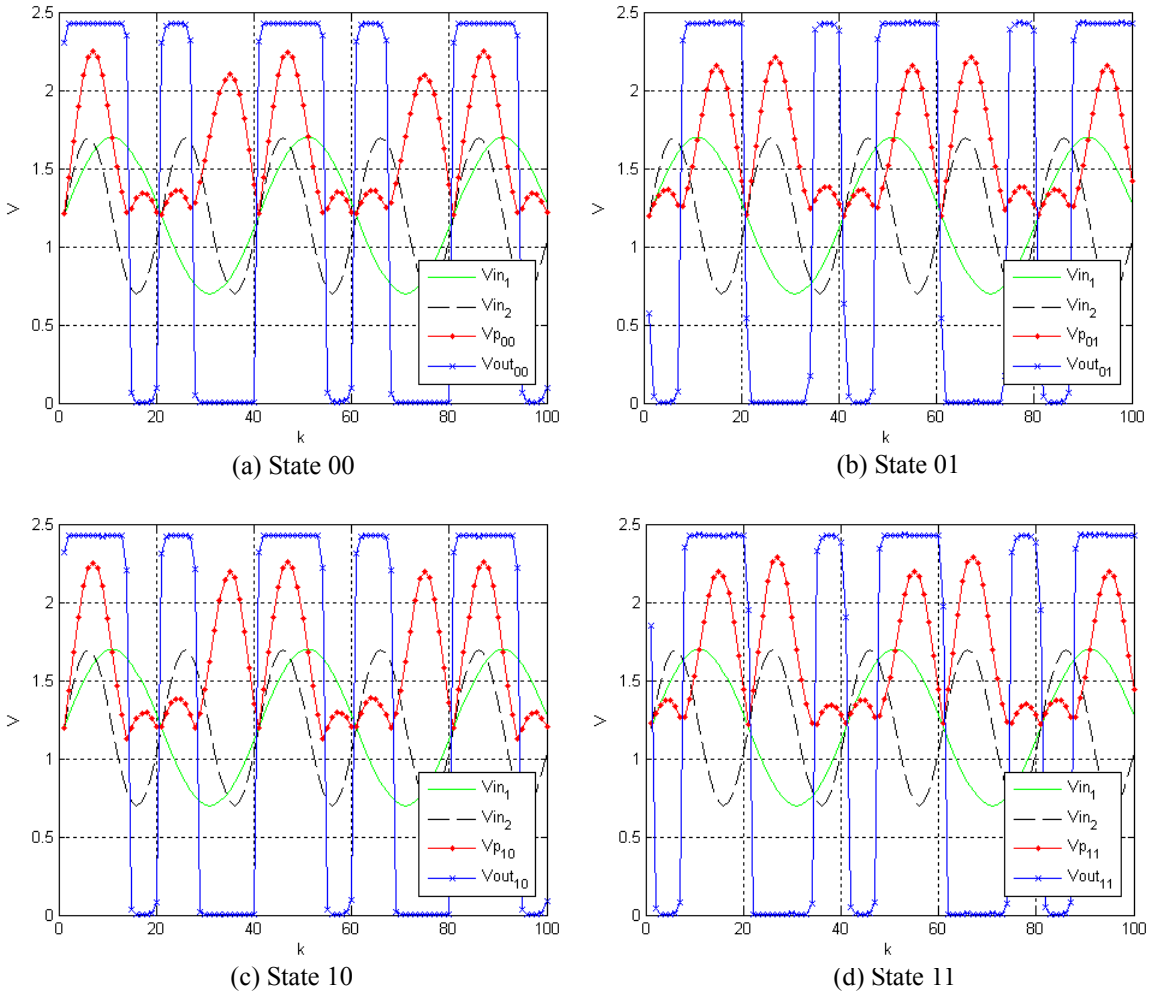
(c) State 10



(d) State 11

Figure 26: Measured outputs from the BMU-ACSU suits implemented in the FPAA for (a) state 00, (b) state 01, (c) state 10, and (d) state 11, while sweeping one input signal from 0.2V to 2.2V and keeping the other input signal at $Vbias$. The previous PM input is also kept to be $Vbias$.

are kept at $Vbias$. In the figure, $Vout_{00}$, $Vout_{01}$, $Vout_{10}$, and $Vout_{11}$ are the binary outputs from the comparators at the state 00, state 01, state 10, and state 11, respectively. Also, Vp_{00}, Vp_{01}, Vp_{10}, Vp_{11} are the selected path metrics at each state. It can be observed that the measured path metrics exhibit fairly linear behaviors within the dynamic range that comply with expected results from (34)-(37). The outputs from the comparators show steep transitions at the point where the majority between two branch metrics changes. This shows that the gain of the OTA inside the FPAA is large enough to be used as a comparator



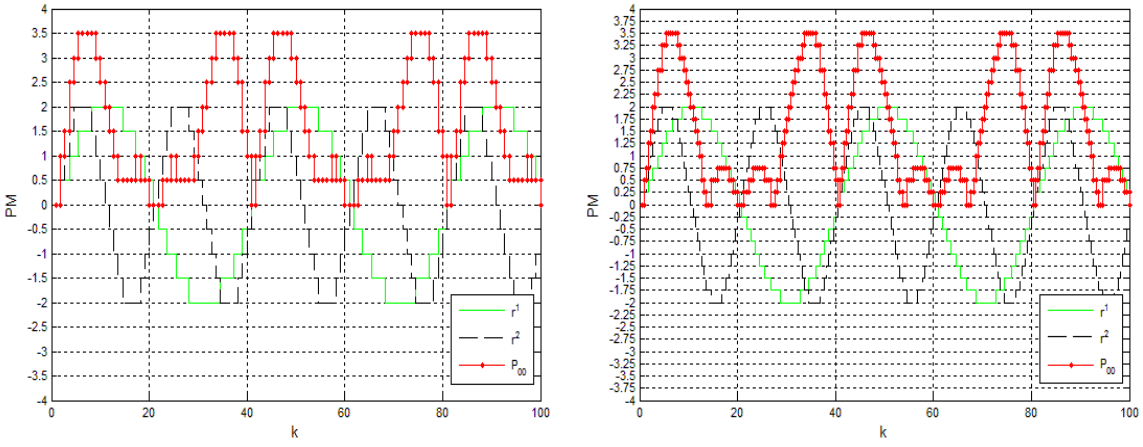
(a) State 00 (b) State 01

(c) State 10 (d) State 11

Figure 27: Measured outputs from the BMU-ACSU suits implemented in the FPAA for (a) state 00, (b) state 01, (c) state 10, and (d) state 11, while varying two input signals with different frequencies and keeping the previous PM input at $Vbias$.

that delivers binary decisions to the TBU. Figure 27(a)-(d) show the measured outputs from the same circuits while varying two input signals with different frequencies and keeping the previous path metrics at *Vbias*. The voltage swing of the input signals is 0.5V centered at *Vbias*. Here, it is more obvious that adding and subtracting (adding sign-inverted signals) operations for the BMU occur properly.

Ideally, the outputs should be identical when the branch metrics have the same absolute value but with the opposite sign. Also, since the previous path metrics are the same, the outputs at state 00 and state 10 should be the same, and the outputs at state 01 and state 11 should be the same. However, one can see in Figures 23-24 that they are not exactly identical due to the remaining mismatch. Moreover, nonlinear effects can be found in the path metrics at state 11 shown in Figure 26(d) when the branch metrics get close to 2.2V. These factors suggest that there still remains a margin for the performance enhancement by tuning each component in the FPAA more accurately.



(a) 4-bit precision (b) 5-bit precision

Figure 28: Quantized inputs and path metrics for state 00 in the digitally implemented BMU-ACSU with (a) 4-bit precision and (b) 5-bit precision.

Figure 28(a)-(b) show the quantized inputs and the path metrics for state 00 in digital implementations that are represented with 4-bit fixed-point precision (1 bit for sign, 2 bits for integer, and 1 bit for fragment) and 5-bit fixed-point precision (1 bit for sign, 2 bits for integer, and 2bit for fragment), respectively. Here, the quantized inputs and path metrics are not in voltage but they are fixed-point values that span over the representable ranges of uniform quantizations, from – 4 to 3.5 spaced by 0.5, and from – 4 to 3.75 spaced by 0.25, respectively. The non-uniform quantization using the Lloyd-Max algorithm was reported to yield similar performance [54], so the uniform quantization is used for simplicity and for the linear mapping of input signals onto path metrics. With these digital implementations, mismatch or nonlinearity is absent, but signals suffer from distortions caused by quantization errors.

Figure 29 shows the measured outputs from the sample-and-hold circuit implemented in the FPAA as given in Figure 21, while sweeping the inputs from 0V to 2.4V. It can be

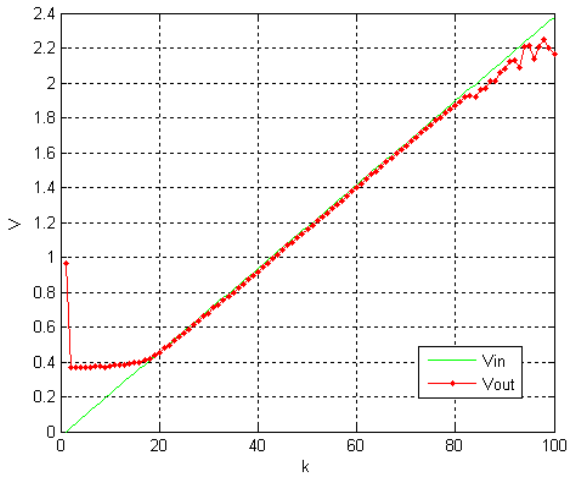


Figure 29: Measured outputs from the sample-and-hold circuit implemented in the FPAA, while sweeping the inputs from 0V to 2.4V.

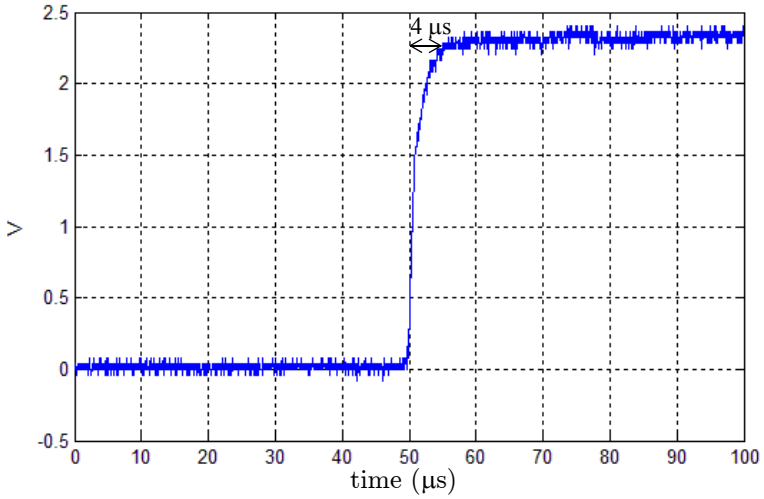


Figure 30: The step response of the BMU-ACSU circuit implemented in the RASP 2.9 FPAA.

observed that the outputs follow the inputs for 0.6V - 1.8V range within fairly tolerable deviations.

The step response of the BMU-ACSU implemented in the FPAA is depicted in Figure 30. As shown in the figure, the processing delay of the BMU-ACSU is 4 µs. The power dissipated in the circuit is measured to be 11.5 mW. A digital implementation using

TABLE 2:
POWER AND DELAY COMPARISONS OF BMU-ACSUS IMPLEMENTED IN FPGA AND FPAA

Chipset	Virtex2Pro FPGA 0.13 µm CMOS process		Virtex FPGA 0.22 µm CMOS process		RASP2.9 FPAA 0.35 µm CMOS process
	5-bit precision	4-bit precision	5-bit precision	4-bit precision	
Power consumption @ Processing delay	129 mW @ 9.4 ns	119 mW @ 8.1 ns	181 mW @ 15.6 ns	129 mW @ 14.9 ns	11.5 mW @ 4 µs
	107 mW @ 4 µs	106 mW @ 4 µs	43 mW @ 4 µs	42 mW @ 4 µs	

The upper power consumption values for the FPGAs are measured when operating at its own fastest speed. The lower values are measured when operating at the same speed as the RASP2.9 FPAA.

the Virtex2P FPGA with 5 bit-precision has the data path delay of 9.4 ns and requires 129 mW power at the maximum speed, and 107 mW at the same speed as the FPAA implementation. For a 4 bit-precision implementation, the data path delay is 8.1 ns and requires 119 mW power at the maximum speed, and 106 mW at the same speed as the FPAA implementation. The Virtex FPGA implementation with 5 bit-precision has 15.6 ns data path delay and requires 181 mW at the maximum speed, and 43 mW at the same speed as the FPAA implementation. A 4 bit-precision implementation has 14.9 ns data path delay and requires 129 mW at the maximum speed, and 42 mW at the same speed as the FPAA implementation. Table II shows the comparisons of the power and delay in the FPGA and FPAA implementations.

So, the power consumption required for the BMU-ACSU operation in the FPAA implementation is significantly reduced by 9.7 dB and 5.7 dB, compared to the Virtex2Pro and Virtex digital implementations, respectively, for the same speed. Aside from the power saving in the BMU-ACSU block itself, it also relieves the bit-precision requirement for the ADC placed after the BMU-ACSU block, which in turn effectively reduces the overall power consumption of the receiver system.

4.4.3 BER Performance in AWGN Channel

Based on the observations from Figure 26 and Figure 29 that suggest the range for linear behaviors, the signal dynamic range is chosen at the range with the voltage swing ΔV = 0.6V centered at $Vbias$ = 1.2V, that is, within 0.6V - 1.8V range. Setting the other input terminal of the FG-OTA at $Vbias$ enables mapping positive values into the range between 0V and V_{bias}+ΔV, and negative values into the range between V_{bias}–ΔV and 0V.

In order to exploit the fine resolution enjoyed in the mixed-signal implementation, the input signals to the decoder block are scaled down by a factor of 1/20 to provide margins for the noisy input signals and the increasing path metrics, then mapped into this dynamic range. Since the increasing rate of the path metrics is roughly 2 as can be seen in Figure 22, each calibrated reference voltage of the I-V converter is further lowered down by 2 (increasing rate) × 1/20 (scaling factor) = 0.1V, so that it is being subtracted out from the updated path metric at each state in each stage. A more robust way to keep the signals within the dynamic range is to subtract out the maximum path metric from the updated path metrics in each stage, at the cost of a larger component count. The maximum path metric can be obtained from the auxiliary comparators and selectors in Figure 23 along with another selector attached at the bit3 stage. With this configuration, the maximum path metric in each stage is buffered with an additional sample-and-hold circuit at the same clock ϕ, and applied to the positive terminal of an additional FG-OTA with V_{bias} on the negative terminal. The output current from the FG-OTA is then added to each output current from the BMU.

To force the trellis to start from state 00, initial PM voltages are applied to the previous PM inputs while ϕ_{init} is high. For state 00, the initial PM voltage is set at V_{bias}, whereas the initial PM for the other three states are set to be 0V. For digital implementations with 4-bit fixed-point precision (1 bit for sign, 2 bits for integer, and 1 bit for fragment) and with 5-bit fixed-point precision (1 bit for sign, 2 bits for integer, and 2 bit for fragment), the initial PM value for state 00 is set at 0, and the other initial PM values are set at the smallest representable value $-2^2 = -4$. As opposed to the mixed-signal implementation, it is not

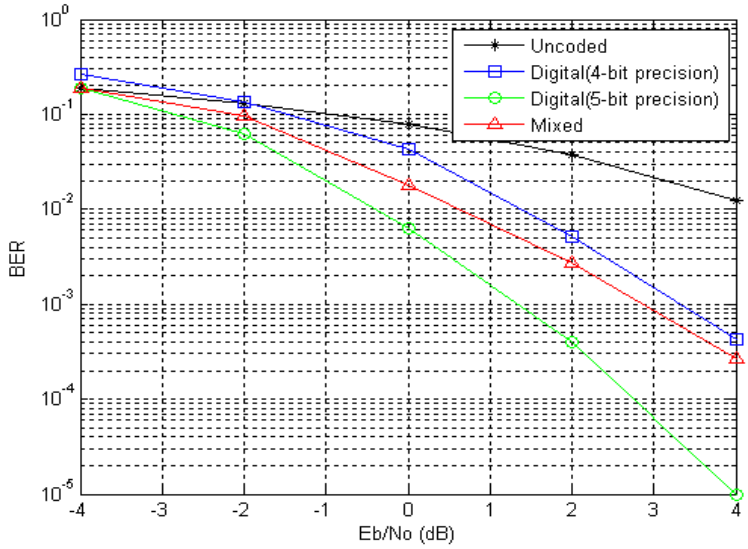


Figure 31: Comparisons of BER performance in AWGN channel. The BER curve of the mixed-signal Viterbi decoder is between that from digital implementations with 4-bit precision and 5-bit precision.

preferable to scale down the received signals in this case to provide the best resolution in representing the path metrics, as the finest quantization step size is limited by the order of the least significant bit (LSB), that is $2^{-1} = 0.5$ and $2^{-2} = 0.25$, respectively. To prevent overflow in the updated path metrics, the maximum path metric is subtracted from all the path metrics in each stage.

Figure 31 shows the BER performance of the proposed mixed-signal Viterbi decoder for - 4dB $\le E_b/N_0 \le$ 4dB, compared to that of digital implementations and the theoretical BER of the uncoded BPSK modulation given by [13]

$$BER = Q(\sqrt{2E_b/N_0}). \tag{40}$$

In the figure, it can be observed that all three implementations provide substantial coding gains compared to the uncoded case, and the BER of the mixed-signal Viterbi decoder resides between that of 4-bit precision and 5-bit precision digital implementations.

4.5 Summary

We have presented a mixed-signal implementation for the soft-decision Viterbi decoder, and achieved the BER performance of the mixed-signal implementation lies between the BER of the 4-bit and 5-bit precision digital decoders without any quantization process involved in the system. The achieved power saving was 9.7 dB and 5.7 dB compared to the Virtex2Pro and Virtex digital implementations, respectively. The mixed-signal implementation exhibits limited distortions due to the inherent mismatch and nonlinearity in the analog domain of the circuits, but as far as these deterministic characteristics of analog components are well characterized and properly mitigated, it provides a better resolution in representing soft information than finite bit-precision digital implementations with less system complexity. The fuzziness in analog signals are eventually removed at the binary decisions from the comparators at the ACSUs during the decoding process.

A convolutional encoder, a noisy channel, and the Viterbi decoder form a hidden Markov model, so the path metric in the next stage depends only on the path metric of the present stage, and not the previous stage. Hence, the BMU and ACSU of the Viterbi decoder do not require a memory to store any intermediate information during the iterative process. However, the TBU needs a memory to back-trace the trellis along the selected paths. Nevertheless, it is not the path metric value itself but the binary decision from the comparator that needs to be stored. So, an ADC with multiple bit-precision to represent soft path metrics is not required, and a slicer with 1-bit precision will suffice for those binary decisions.

CHAPTER 5

CONCLUSIONS

5.1 Summary of Research

We have explored the mixed-signal implementations for reduced-complexity wireless communication systems by eliminating redundant complexity due to the quantization process and by replacing digital circuits with small-signal analog circuits. We also discussed the performance aspect of the system when the quantization process is absent.

Two essential signal processing blocks for modern communication systems, a DFT block for OFDM demodulator and a soft-decision Viterbi decoder, were implemented in the FPAA and the measurement results were compared to that of digital implementations. The DFT block implemented with analog circuits can provide unquantized soft inputs to the subsequent soft-decision Viterbi decoder in OFDM receivers. The analog 4-DFT implemented in the FPAA consumed 9.4 dB less power than a digital implementation using a Virtex2Pro FPGA, and 7.1 dB less power than a digital implementation using a Virtex FPGA for the same speed. The price paid for this power reduction was a 2 dB performance degradation, but the power saving using the analog circuits dominates the signal power required to cope with the performance degradation. The BMU-ACSU of the soft-decision Viterbi decoder implemented in the FPAA achieved the power saving of 9.7 dB and 5.7 dB compared to the Virtex2Pro and Virtex digital implementations with the

same speed, respectively. The BER performance of the FPAA implementation lies between the BER of the 4-bit and 5-bit precision digital decoders.

These achieved power reduction, although significant, does not reflect the additional power savings that comes from the relieved requirement for an ADC in the wireless receiver. A typical ADC first samples received signals to bring continuous-time continuous-valued signals to discrete-time continuous-valued samples, then quantizes them to bring discrete-time continuous-valued samples to discrete-time discrete-valued samples. So, implementing signal processing blocks in an analog domain releases the requirements for the quantization process at the ADC block, which yields further improvement in overall system power efficiency.

To improve the performance of suggested mixed-signal processing blocks, the mismatch and nonlinearity exhibited in analog components are characterized and mitigated properly at the post-fabrication stage. The multiplicative gain mismatch was alleviated by adjusting the charge stored in the floating-gate capacitors connected to the MOSFETs. The additive mismatch in device offsets are compensated for by assigning independent reference voltage in each I-V converter.

The nonlinear behaviors in analog components are circumvented by keeping the signal flowing though the system within the dynamic range in which each analog component behaves linearly. This is especially crucial when iterative operations are involved, as in the ACSU of the Viterbi decoder. In particular, the accumulating nature of path metrics from the ACSU will eventually cause overflow, unless the updated path metrics are reduced in each iteration. In our suggested configuration, this can be accomplished by simply

lowering down the reference voltage of the I-V converter by a constant. This provides a longer sliding window for convolutional codes with a longer constraint length, without any aid of digital blocks.

We have also employed different signal modes for different operations, so that each required operation for the signal processing becomes trivial. That is, adding operations are performed in current-mode, whereas copying (broadcasting and selecting), comparing, and sample-and-hold operations are performed in voltage-mode. This enables to avoid any source of nonlinearity in analog circuit operations, and to have a highly parallelized structure for adding and copying operations.

Auxiliary comparators and selectors are attached to the outputs from the ACSUs, so as to hand over the starting point of the backward trace to the TBU. The binary outputs from the auxiliary comparators are not required to be stored, and can be represented with a trivial 1-bit precision.

5.2 Challenges in the FPAA Roadmap

There are several challenges remained for the FPAA to be applied in variety of signal processing blocks for wireless communication systems. Those can be summarized by

- performance degradation due to the inherent mismatch and different levels of nonlinearity between devices
- current leakage in a sample-and-hold circuit due to the parasitic resistance and capacitance across the routed paths
- long settling time for small signals
- nonlinearity between input voltage and input current caused by the non-ideal characteristic of the OTA
- temperature sensitivity of the drain current in the floating-gate transistors
- current congestion errors during the placement and routing of the netlist

- insufficient I/O pins for larger multiple-input multiple-output blocks

Although some approaches to tackle these obstacles have been addressed in this work, there still remain fundamental challenges, such as the long settling time and the current congestions that need to be overcome by employing a smaller CMOS process and optimizing the placement and routing of the netlist. The current leakage issue in the sample-and-hold circuit can be addressed by fabricating a CAB that includes a hard-wired capacitor connected to an OTA input node.

5.3 Expected Impacts

Despite the remaining challenges in FPAA implementations, mixed-signal processing implementations can significantly reduce the overall power consumption of the system by using a small-signal analog circuits and at the same time by relieving the redundant complexity of the system. It can also be beneficial from a performance point of view, because it enables to take full advantage of the soft decoding scheme without quantizing the signals.

We have demonstrated how the floating-gate capacitors in the FPAA can play a role in tuning inherent mismatch experienced in analog circuit design. However, this tunability can also be applied to ASIC implementations by deploying floating-gate capacitors in the MOSFET circuit design. In this way, the device mismatch determined during the fabrication process can be tuned afterwards by adjusting the charge stored in the floating-gate capacitors.

APPENDIX A: DFT NETLIST

```
.INCLUDE fpaa_tech.sp

*INPORT invector
*>> pin io_up 25 net int1net1p
*>> pin io_rt 12 net int1net1n
*>> pin io_lt 3 net int1net2p
*>> pin io_rt 26 net int1net2n
*>> pin io_lt 9 net int1net3p
*>> pin io_rt 3 net int1net3n
*>> pin io_up 13 net int1net4p
*>> pin io_rt 9 net int1net4n
*>> pin io_lt 15 net int1net5p
*>> pin io_rt 15 net int1net5n
*>> pin io_lt 18 net int1net6p
*>> pin io_rt 18 net int1net6n
*>> pin io_lt 21 net int1net7p
*>> pin io_rt 21 net int1net7n
*>> pin io_lt 24 net int1net8p
*>> pin io_rt 24 net int1net8n

*INPORT vmm_ref
*>> pin io_lt 26 net int3net1

.subckt VMMSUB int1net1pInp int1net1nInn int1net2pInp int1net2nInn int1net3pInp int1net3nInn int1net4pInp int1net4nInn int1net5pInp int1net5nInn int1net6pInp int1net6nInn int1net7pInp int1net7nInn int1net8pInp int1net8nInn int3net1In int2net1pOutp int2net1nOutn int2net2pOutp int2net2nOutn int2net3pOutp int2net3nOutn int2net4pOutp int2net4nOutn int2net5pOutp int2net5nOutn int2net6pOutp int2net6nOutn int2net7pOutp int2net7nOutn int2net8pOutp int2net8nOutn

XfgInp1 nRowp1 int1net1pInp FGE1 PARAMS: val=2e-007
xotap1 int3net1In int1net1pInp nRowp1 OTA PARAMS: Ib=1.8e-004
XfgInn1 nRown1 int1net1nInn FGE1 PARAMS: val=2e-007
xotan1 int3net1In int1net1nInn nRown1 OTA PARAMS: Ib=1.8e-004
XfgTL11 nRowp1 int2net1pOutp FGE1 PARAMS: val=3e-007
XfgTR11 nRowp1 int2net1nOutn FGE1 PARAMS: val=1e-007
XfgBL11 nRown1 int2net1pOutp FGE1 PARAMS: val=1e-007
XfgBR11 nRown1 int2net1nOutn FGE1 PARAMS: val=3e-007
XfgTL21 nRowp1 int2net2pOutp FGE1 PARAMS: val=3e-007
XfgTR21 nRowp1 int2net2nOutn FGE1 PARAMS: val=1e-007
XfgBL21 nRown1 int2net2pOutp FGE1 PARAMS: val=1e-007
```

XfgBR21 nRown1 int2net2nOutn FGE1 PARAMS: val=3e-007

XfgTL31 nRowp1 int2net3pOutp FGE1 PARAMS: val=3e-007

XfgTR31 nRowp1 int2net3nOutn FGE1 PARAMS: val=1e-007

XfgBL31 nRown1 int2net3pOutp FGE1 PARAMS: val=1e-007

XfgBR31 nRown1 int2net3nOutn FGE1 PARAMS: val=3e-007

XfgTL41 nRowp1 int2net4pOutp FGE1 PARAMS: val=3e-007

XfgTR41 nRowp1 int2net4nOutn FGE1 PARAMS: val=1e-007

XfgBL41 nRown1 int2net4pOutp FGE1 PARAMS: val=1e-007

XfgBR41 nRown1 int2net4nOutn FGE1 PARAMS: val=3e-007

XfgTL51 nRowp1 int2net5pOutp FGE1 PARAMS: val=0

XfgTR51 nRowp1 int2net5nOutn FGE1 PARAMS: val=0

XfgBL51 nRown1 int2net5pOutp FGE1 PARAMS: val=0

XfgBR51 nRown1 int2net5nOutn FGE1 PARAMS: val=0

XfgTL61 nRowp1 int2net6pOutp FGE1 PARAMS: val=0

XfgTR61 nRowp1 int2net6nOutn FGE1 PARAMS: val=0

XfgBL61 nRown1 int2net6pOutp FGE1 PARAMS: val=0

XfgBR61 nRown1 int2net6nOutn FGE1 PARAMS: val=0

XfgTL71 nRowp1 int2net7pOutp FGE1 PARAMS: val=0

XfgTR71 nRowp1 int2net7nOutn FGE1 PARAMS: val=0

XfgBL71 nRown1 int2net7pOutp FGE1 PARAMS: val=0

XfgBR71 nRown1 int2net7nOutn FGE1 PARAMS: val=0

XfgTL81 nRowp1 int2net8pOutp FGE1 PARAMS: val=0

XfgTR81 nRowp1 int2net8nOutn FGE1 PARAMS: val=0

XfgBL81 nRown1 int2net8pOutp FGE1 PARAMS: val=0

XfgBR81 nRown1 int2net8nOutn FGE1 PARAMS: val=0

XfgInp2 nRowp2 int1net2pInp FGE1 PARAMS: val=2e-007

xotap2 int3net1In int1net2pInp nRowp2 OTA PARAMS: Ib=1.8e-004

XfgInn2 nRown2 int1net2nInn FGE1 PARAMS: val=2e-007

xotan2 int3net1In int1net2nInn nRown2 OTA PARAMS: Ib=1.8e-004

XfgTL12 nRowp2 int2net1pOutp FGE1 PARAMS: val=3e-007

XfgTR12 nRowp2 int2net1nOutn FGE1 PARAMS: val=1e-007

XfgBL12 nRown2 int2net1pOutp FGE1 PARAMS: val=1e-007

XfgBR12 nRown2 int2net1nOutn FGE1 PARAMS: val=3e-007

XfgTL22 nRowp2 int2net2pOutp FGE1 PARAMS: val=0

XfgTR22 nRowp2 int2net2nOutn FGE1 PARAMS: val=0

XfgBL22 nRown2 int2net2pOutp FGE1 PARAMS: val=0

XfgBR22 nRown2 int2net2nOutn FGE1 PARAMS: val=0

XfgTL32 nRowp2 int2net3pOutp FGE1 PARAMS: val=1e-007

XfgTR32 nRowp2 int2net3nOutn FGE1 PARAMS: val=3e-007

XfgBL32 nRown2 int2net3pOutp FGE1 PARAMS: val=3e-007

XfgBR32 nRown2 int2net3nOutn FGE1 PARAMS: val=1e-007

XfgTL42 nRowp2 int2net4pOutp FGE1 PARAMS: val=0

XfgTR42 nRowp2 int2net4nOutn FGE1 PARAMS: val=0

XfgBL42 nRown2 int2net4pOutp FGE1 PARAMS: val=0

XfgBR42 nRown2 int2net4nOutn FGE1 PARAMS: val=0

XfgTL52 nRowp2 int2net5pOutp FGE1 PARAMS: val=0

XfgTR52 nRowp2 int2net5nOutn FGE1 PARAMS: val=0

XfgBL52 nRown2 int2net5pOutp FGE1 PARAMS: val=0

XfgBR52 nRown2 int2net5nOutn FGE1 PARAMS: val=0

XfgTL62 nRowp2 int2net6pOutp FGE1 PARAMS: val=1e-007

XfgTR62 nRowp2 int2net6nOutn FGE1 PARAMS: val=3e-007

XfgBL62 nRown2 int2net6pOutp FGE1 PARAMS: val=3e-007

XfgBR62 nRown2 int2net6nOutn FGE1 PARAMS: val=1e-007

XfgTL72 nRowp2 int2net7pOutp FGE1 PARAMS: val=0

XfgTR72 nRowp2 int2net7nOutn FGE1 PARAMS: val=0

XfgBL72 nRown2 int2net7pOutp FGE1 PARAMS: val=0

XfgBR72 nRown2 int2net7nOutn FGE1 PARAMS: val=0

XfgTL82 nRowp2 int2net8pOutp FGE1 PARAMS: val=3e-007

XfgTR82 nRowp2 int2net8nOutn FGE1 PARAMS: val=1e-007

XfgBL82 nRown2 int2net8pOutp FGE1 PARAMS: val=1e-007

XfgBR82 nRown2 int2net8nOutn FGE1 PARAMS: val=3e-007

XfgInp3 nRowp3 int1net3pInp FGE1 PARAMS: val=2e-007

xotap3 int3net1In int1net3pInp nRowp3 OTA PARAMS: Ib=1.8e-004

XfgInn3 nRown3 int1net3nInn FGE1 PARAMS: val=2e-007

xotan3 int3net1In int1net3nInn nRown3 OTA PARAMS: Ib=1.8e-004

XfgTL13 nRowp3 int2net1pOutp FGE1 PARAMS: val=3e-007

XfgTR13 nRowp3 int2net1nOutn FGE1 PARAMS: val=1e-007

XfgBL13 nRown3 int2net1pOutp FGE1 PARAMS: val=1e-007

XfgBR13 nRown3 int2net1nOutn FGE1 PARAMS: val=3e-007

XfgTL23 nRowp3 int2net2pOutp FGE1 PARAMS: val=1e-007

XfgTR23 nRowp3 int2net2nOutn FGE1 PARAMS: val=3e-007

XfgBL23 nRown3 int2net2pOutp FGE1 PARAMS: val=3e-007

XfgBR23 nRown3 int2net2nOutn FGE1 PARAMS: val=1e-007

XfgTL33 nRowp3 int2net3pOutp FGE1 PARAMS: val=3e-007

XfgTR33 nRowp3 int2net3nOutn FGE1 PARAMS: val=1e-007

XfgBL33 nRown3 int2net3pOutp FGE1 PARAMS: val=1e-007

XfgBR33 nRown3 int2net3nOutn FGE1 PARAMS: val=3e-007

XfgTL43 nRowp3 int2net4pOutp FGE1 PARAMS: val=1e-007

XfgTR43 nRowp3 int2net4nOutn FGE1 PARAMS: val=3e-007

XfgBL43 nRown3 int2net4pOutp FGE1 PARAMS: val=3e-007

XfgBR43 nRown3 int2net4nOutn FGE1 PARAMS: val=1e-007

XfgTL53 nRowp3 int2net5pOutp FGE1 PARAMS: val=0

XfgTR53 nRowp3 int2net5nOutn FGE1 PARAMS: val=0

XfgBL53 nRown3 int2net5pOutp FGE1 PARAMS: val=0

XfgBR53 nRown3 int2net5nOutn FGE1 PARAMS: val=0

XfgTL63 nRowp3 int2net6pOutp FGE1 PARAMS: val=0

XfgTR63 nRowp3 int2net6nOutn FGE1 PARAMS: val=0

XfgBL63 nRown3 int2net6pOutp FGE1 PARAMS: val=0

XfgBR63 nRown3 int2net6nOutn FGE1 PARAMS: val=0

XfgTL73 nRowp3 int2net7pOutp FGE1 PARAMS: val=0

XfgTR73 nRowp3 int2net7nOutn FGE1 PARAMS: val=0

XfgBL73 nRown3 int2net7pOutp FGE1 PARAMS: val=0

XfgBR73 nRown3 int2net7nOutn FGE1 PARAMS: val=0

XfgTL83 nRowp3 int2net8pOutp FGE1 PARAMS: val=0

XfgTR83 nRowp3 int2net8nOutn FGE1 PARAMS: val=0

XfgBL83 nRown3 int2net8pOutp FGE1 PARAMS: val=0

XfgBR83 nRown3 int2net8nOutn FGE1 PARAMS: val=0

XfgInp4 nRowp4 int1net4pInp FGE1 PARAMS: val=2e-007

xotap4 int3net1In int1net4pInp nRowp4 OTA PARAMS: Ib=1.8e-004

XfgInn4 nRown4 int1net4nInn FGE1 PARAMS: val=2e-007

xotan4 int3net1In int1net4nInn nRown4 OTA PARAMS: Ib=1.8e-004

XfgTL14 nRowp4 int2net1pOutp FGE1 PARAMS: val=3e-007

XfgTR14 nRowp4 int2net1nOutn FGE1 PARAMS: val=1e-007

XfgBL14 nRown4 int2net1pOutp FGE1 PARAMS: val=1e-007

XfgBR14 nRown4 int2net1nOutn FGE1 PARAMS: val=3e-007

XfgTL24 nRowp4 int2net2pOutp FGE1 PARAMS: val=0

XfgTR24 nRowp4 int2net2nOutn FGE1 PARAMS: val=0

XfgBL24 nRown4 int2net2pOutp FGE1 PARAMS: val=0

XfgBR24 nRown4 int2net2nOutn FGE1 PARAMS: val=0

XfgTL34 nRowp4 int2net3pOutp FGE1 PARAMS: val=1e-007

XfgTR34 nRowp4 int2net3nOutn FGE1 PARAMS: val=3e-007

XfgBL34 nRown4 int2net3pOutp FGE1 PARAMS: val=3e-007

XfgBR34 nRown4 int2net3nOutn FGE1 PARAMS: val=1e-007

XfgTL44 nRowp4 int2net4pOutp FGE1 PARAMS: val=0

XfgTR44 nRowp4 int2net4nOutn FGE1 PARAMS: val=0

XfgBL44 nRown4 int2net4pOutp FGE1 PARAMS: val=0

XfgBR44 nRown4 int2net4nOutn FGE1 PARAMS: val=0

XfgTL54 nRowp4 int2net5pOutp FGE1 PARAMS: val=0

XfgTR54 nRowp4 int2net5nOutn FGE1 PARAMS: val=0

XfgBL54 nRown4 int2net5pOutp FGE1 PARAMS: val=0

XfgBR54 nRown4 int2net5nOutn FGE1 PARAMS: val=0

XfgTL64 nRowp4 int2net6pOutp FGE1 PARAMS: val=3e-007

XfgTR64 nRowp4 int2net6nOutn FGE1 PARAMS: val=1e-007

XfgBL64 nRown4 int2net6pOutp FGE1 PARAMS: val=1e-007

XfgBR64 nRown4 int2net6nOutn FGE1 PARAMS: val=3e-007

XfgTL74 nRowp4 int2net7pOutp FGE1 PARAMS: val=0

XfgTR74 nRowp4 int2net7nOutn FGE1 PARAMS: val=0

XfgBL74 nRown4 int2net7pOutp FGE1 PARAMS: val=0

XfgBR74 nRown4 int2net7nOutn FGE1 PARAMS: val=0

XfgTL84 nRowp4 int2net8pOutp FGE1 PARAMS: val=1e-007

XfgTR84 nRowp4 int2net8nOutn FGE1 PARAMS: val=3e-007

XfgBL84 nRown4 int2net8pOutp FGE1 PARAMS: val=3e-007

XfgBR84 nRown4 int2net8nOutn FGE1 PARAMS: val=1e-007

XfgInp5 nRowp5 int1net5pInp FGE1 PARAMS: val=2e-007

xotap5 int3net1In int1net5pInp nRowp5 OTA PARAMS: Ib=1.8e-004

XfgInn5 nRown5 int1net5nInn FGE1 PARAMS: val=2e-007

xotan5 int3net1In int1net5nInn nRown5 OTA PARAMS: Ib=1.8e-004

XfgTL15 nRowp5 int2net1pOutp FGE1 PARAMS: val=0

XfgTR15 nRowp5 int2net1nOutn FGE1 PARAMS: val=0

XfgBL15 nRown5 int2net1pOutp FGE1 PARAMS: val=0

XfgBR15 nRown5 int2net1nOutn FGE1 PARAMS: val=0

XfgTL25 nRowp5 int2net2pOutp FGE1 PARAMS: val=0

XfgTR25 nRowp5 int2net2nOutn FGE1 PARAMS: val=0

XfgBL25 nRown5 int2net2pOutp FGE1 PARAMS: val=0

XfgBR25 nRown5 int2net2nOutn FGE1 PARAMS: val=0

XfgTL35 nRowp5 int2net3pOutp FGE1 PARAMS: val=0

XfgTR35 nRowp5 int2net3nOutn FGE1 PARAMS: val=0

XfgBL35 nRown5 int2net3pOutp FGE1 PARAMS: val=0

XfgBR35 nRown5 int2net3nOutn FGE1 PARAMS: val=0

XfgTL45 nRowp5 int2net4pOutp FGE1 PARAMS: val=0

XfgTR45 nRowp5 int2net4nOutn FGE1 PARAMS: val=0

XfgBL45 nRown5 int2net4pOutp FGE1 PARAMS: val=0

XfgBR45 nRown5 int2net4nOutn FGE1 PARAMS: val=0

XfgTL55 nRowp5 int2net5pOutp FGE1 PARAMS: val=3e-007

XfgTR55 nRowp5 int2net5nOutn FGE1 PARAMS: val=1e-007

XfgBL55 nRown5 int2net5pOutp FGE1 PARAMS: val=1e-007

XfgBR55 nRown5 int2net5nOutn FGE1 PARAMS: val=3e-007

XfgTL65 nRowp5 int2net6pOutp FGE1 PARAMS: val=3e-007

XfgTR65 nRowp5 int2net6nOutn FGE1 PARAMS: val=1e-007

XfgBL65 nRown5 int2net6pOutp FGE1 PARAMS: val=1e-007

XfgBR65 nRown5 int2net6nOutn FGE1 PARAMS: val=3e-007

XfgTL75 nRowp5 int2net7pOutp FGE1 PARAMS: val=3e-007

XfgTR75 nRowp5 int2net7nOutn FGE1 PARAMS: val=1e-007

XfgBL75 nRown5 int2net7pOutp FGE1 PARAMS: val=1e-007

XfgBR75 nRown5 int2net7nOutn FGE1 PARAMS: val=3e-007

XfgTL85 nRowp5 int2net8pOutp FGE1 PARAMS: val=3e-007

XfgTR85 nRowp5 int2net8nOutn FGE1 PARAMS: val=1e-007

XfgBL85 nRown5 int2net8pOutp FGE1 PARAMS: val=1e-007

XfgBR85 nRown5 int2net8nOutn FGE1 PARAMS: val=3e-007

XfgInp6 nRowp6 int1net6pInp FGE1 PARAMS: val=2e-007

xotap6 int3net1In int1net6pInp nRowp6 OTA PARAMS: Ib=1.8e-004

XfgInn6 nRown6 int1net6nInn FGE1 PARAMS: val=2e-007

xotan6 int3net1In int1net6nInn nRown6 OTA PARAMS: Ib=1.8e-004

XfgTL16 nRowp6 int2net1pOutp FGE1 PARAMS: val=0

XfgTR16 nRowp6 int2net1nOutn FGE1 PARAMS: val=0

XfgBL16 nRown6 int2net1pOutp FGE1 PARAMS: val=0

XfgBR16 nRown6 int2net1nOutn FGE1 PARAMS: val=0

XfgTL26 nRowp6 int2net2pOutp FGE1 PARAMS: val=3e-007

XfgTR26 nRowp6 int2net2nOutn FGE1 PARAMS: val=1e-007

XfgBL26 nRown6 int2net2pOutp FGE1 PARAMS: val=1e-007

XfgBR26 nRown6 int2net2nOutn FGE1 PARAMS: val=3e-007

XfgTL36 nRowp6 int2net3pOutp FGE1 PARAMS: val=0

XfgTR36 nRowp6 int2net3nOutn FGE1 PARAMS: val=0

XfgBL36 nRown6 int2net3pOutp FGE1 PARAMS: val=0

XfgBR36 nRown6 int2net3nOutn FGE1 PARAMS: val=0

XfgTL46 nRowp6 int2net4pOutp FGE1 PARAMS: val=1e-007

XfgTR46 nRowp6 int2net4nOutn FGE1 PARAMS: val=3e-007

XfgBL46 nRown6 int2net4pOutp FGE1 PARAMS: val=3e-007

XfgBR46 nRown6 int2net4nOutn FGE1 PARAMS: val=1e-007

XfgTL56 nRowp6 int2net5pOutp FGE1 PARAMS: val=3e-007

XfgTR56 nRowp6 int2net5nOutn FGE1 PARAMS: val=1e-007

XfgBL56 nRown6 int2net5pOutp FGE1 PARAMS: val=1e-007

XfgBR56 nRown6 int2net5nOutn FGE1 PARAMS: val=3e-007

XfgTL66 nRowp6 int2net6pOutp FGE1 PARAMS: val=0

XfgTR66 nRowp6 int2net6nOutn FGE1 PARAMS: val=0

XfgBL66 nRown6 int2net6pOutp FGE1 PARAMS: val=0

XfgBR66 nRown6 int2net6nOutn FGE1 PARAMS: val=0

XfgTL76 nRowp6 int2net7pOutp FGE1 PARAMS: val=1e-007

XfgTR76 nRowp6 int2net7nOutn FGE1 PARAMS: val=3e-007

XfgBL76 nRown6 int2net7pOutp FGE1 PARAMS: val=3e-007

XfgBR76 nRown6 int2net7nOutn FGE1 PARAMS: val=1e-007

XfgTL86 nRowp6 int2net8pOutp FGE1 PARAMS: val=0

XfgTR86 nRowp6 int2net8nOutn FGE1 PARAMS: val=0

XfgBL86 nRown6 int2net8pOutp FGE1 PARAMS: val=0

XfgBR86 nRown6 int2net8nOutn FGE1 PARAMS: val=0

XfgInp7 nRowp7 int1net7pInp FGE1 PARAMS: val=2e-007

xotap7 int3net1In int1net7pInp nRowp7 OTA PARAMS: Ib=1.8e-004

XfgInn7 nRown7 int1net7nInn FGE1 PARAMS: val=2e-007

xotan7 int3net1In int1net7nInn nRown7 OTA PARAMS: Ib=1.8e-004

XfgTL17 nRowp7 int2net1pOutp FGE1 PARAMS: val=0

XfgTR17 nRowp7 int2net1nOutn FGE1 PARAMS: val=0

XfgBL17 nRown7 int2net1pOutp FGE1 PARAMS: val=0

XfgBR17 nRown7 int2net1nOutn FGE1 PARAMS: val=0

XfgTL27 nRowp7 int2net2pOutp FGE1 PARAMS: val=0

XfgTR27 nRowp7 int2net2nOutn FGE1 PARAMS: val=0

XfgBL27 nRown7 int2net2pOutp FGE1 PARAMS: val=0

XfgBR27 nRown7 int2net2nOutn FGE1 PARAMS: val=0

XfgTL37 nRowp7 int2net3pOutp FGE1 PARAMS: val=0

XfgTR37 nRowp7 int2net3nOutn FGE1 PARAMS: val=0

XfgBL37 nRown7 int2net3pOutp FGE1 PARAMS: val=0

XfgBR37 nRown7 int2net3nOutn FGE1 PARAMS: val=0

XfgTL47 nRowp7 int2net4pOutp FGE1 PARAMS: val=0

XfgTR47 nRowp7 int2net4nOutn FGE1 PARAMS: val=0

XfgBL47 nRown7 int2net4pOutp FGE1 PARAMS: val=0

XfgBR47 nRown7 int2net4nOutn FGE1 PARAMS: val=0

XfgTL57 nRowp7 int2net5pOutp FGE1 PARAMS: val=3e-007

XfgTR57 nRowp7 int2net5nOutn FGE1 PARAMS: val=1e-007

XfgBL57 nRown7 int2net5pOutp FGE1 PARAMS: val=1e-007

XfgBR57 nRown7 int2net5nOutn FGE1 PARAMS: val=3e-007

XfgTL67 nRowp7 int2net6pOutp FGE1 PARAMS: val=1e-007

XfgTR67 nRowp7 int2net6nOutn FGE1 PARAMS: val=3e-007

XfgBL67 nRown7 int2net6pOutp FGE1 PARAMS: val=3e-007

XfgBR67 nRown7 int2net6nOutn FGE1 PARAMS: val=1e-007

XfgTL77 nRowp7 int2net7pOutp FGE1 PARAMS: val=3e-007

XfgTR77 nRowp7 int2net7nOutn FGE1 PARAMS: val=1e-007

XfgBL77 nRown7 int2net7pOutp FGE1 PARAMS: val=1e-007

XfgBR77 nRown7 int2net7nOutn FGE1 PARAMS: val=3e-007

XfgTL87 nRowp7 int2net8pOutp FGE1 PARAMS: val=1e-007

XfgTR87 nRowp7 int2net8nOutn FGE1 PARAMS: val=3e-007

XfgBL87 nRown7 int2net8pOutp FGE1 PARAMS: val=3e-007

XfgBR87 nRown7 int2net8nOutn FGE1 PARAMS: val=1e-007

XfgInp8 nRowp8 int1net8pInp FGE1 PARAMS: val=2e-007

xotap8 int3net1In int1net8pInp nRowp8 OTA PARAMS: Ib=1.8e-004

XfgInn8 nRown8 int1net8nInn FGE1 PARAMS: val=2e-007

xotan8 int3net1In int1net8nInn nRown8 OTA PARAMS: Ib=1.8e-004

XfgTL18 nRowp8 int2net1pOutp FGE1 PARAMS: val=0

XfgTR18 nRowp8 int2net1nOutn FGE1 PARAMS: val=0

XfgBL18 nRown8 int2net1pOutp FGE1 PARAMS: val=0

XfgBR18 nRown8 int2net1nOutn FGE1 PARAMS: val=0

XfgTL28 nRowp8 int2net2pOutp FGE1 PARAMS: val=1e-007

XfgTR28 nRowp8 int2net2nOutn FGE1 PARAMS: val=3e-007

XfgBL28 nRown8 int2net2pOutp FGE1 PARAMS: val=3e-007

XfgBR28 nRown8 int2net2nOutn FGE1 PARAMS: val=1e-007

XfgTL38 nRowp8 int2net3pOutp FGE1 PARAMS: val=0

XfgTR38 nRowp8 int2net3nOutn FGE1 PARAMS: val=0

XfgBL38 nRown8 int2net3pOutp FGE1 PARAMS: val=0

XfgBR38 nRown8 int2net3nOutn FGE1 PARAMS: val=0

XfgTL48 nRowp8 int2net4pOutp FGE1 PARAMS: val=3e-007

XfgTR48 nRowp8 int2net4nOutn FGE1 PARAMS: val=1e-007

XfgBL48 nRown8 int2net4pOutp FGE1 PARAMS: val=1e-007

XfgBR48 nRown8 int2net4nOutn FGE1 PARAMS: val=3e-007

XfgTL58 nRowp8 int2net5pOutp FGE1 PARAMS: val=3e-007

XfgTR58 nRowp8 int2net5nOutn FGE1 PARAMS: val=1e-007

XfgBL58 nRown8 int2net5pOutp FGE1 PARAMS: val=1e-007

XfgBR58 nRown8 int2net5nOutn FGE1 PARAMS: val=3e-007

XfgTL68 nRowp8 int2net6pOutp FGE1 PARAMS: val=0

XfgTR68 nRowp8 int2net6nOutn FGE1 PARAMS: val=0

XfgBL68 nRown8 int2net6pOutp FGE1 PARAMS: val=0

XfgBR68 nRown8 int2net6nOutn FGE1 PARAMS: val=0

XfgTL78 nRowp8 int2net7pOutp FGE1 PARAMS: val=1e-007

XfgTR78 nRowp8 int2net7nOutn FGE1 PARAMS: val=3e-007

XfgBL78 nRown8 int2net7pOutp FGE1 PARAMS: val=3e-007

XfgBR78 nRown8 int2net7nOutn FGE1 PARAMS: val=1e-007

XfgTL88 nRowp8 int2net8pOutp FGE1 PARAMS: val=0

XfgTR88 nRowp8 int2net8nOutn FGE1 PARAMS: val=0

XfgBL88 nRown8 int2net8pOutp FGE1 PARAMS: val=0

XfgBR88 nRown8 int2net8nOutn FGE1 PARAMS: val=0

.ends

XVMMSUB int1net1p int1net1n int1net2p int1net2n int1net3p int1net3n int1net4p int1net4n int1net5p int1net5n int1net6p int1net6n int1net7p int1net7n int1net8p int1net8n int3net1 int2net1p int2net1n int2net2p int2net2n int2net3p int2net3n int2net4p int2net4n int2net5p int2net5n int2net6p int2net6n int2net7p int2net7n int2net8p int2net8n VMMSUB

*OUTPORT outvector

*>> pin io_dn 0 net int2net1p

*>> pin io_dn 21 net int2net1n

*>> pin io_dn 2 net int2net2p

*>> pin io_dn 22 net int2net2n

*>> pin io_dn 4 net int2net3p

*>> pin io_dn 24 net int2net3n

*>> pin io_dn 6 net int2net4p

*>> pin io_dn 26 net int2net4n

*>> pin io_dn 8 net int2net5p

*>> pin io_up 11 net int2net5n

*>> pin io_dn 9 net int2net6p

*>> pin io_up 4 net int2net6n

*>> pin io_dn 10 net int2net7p

*>> pin io_up 6 net int2net7n

*>> pin io_dn 12 net int2net8p

*>> pin io_up 22 net int2net8n

*>> devicefile rasp2_9a_rev2.dev

APPENDIX B: VITERBI DECODER NETLIST

```
.INCLUDE fpaa_tech.sp

*INPORT invector
*>> pin io_rt 3  net Vphi_init
*>> pin io_lt 15 net Vinit1
*>> pin io_rt 9  net Vinit2
*>> pin io_up 13 net Vphi
*>> pin io_lt 21 net Vin1
*>> pin io_rt 12 net Vin2
*>> pin io_lt 9  net Vbias
*>> pin io_lt 3  net Vref1
*>> pin io_rt 21 net Vref2
*>> pin io_lt 24 net Vref3
*>> pin io_rt 24 net Vref4
*>> pin io_lt 18 net Vref5
*>> pin io_rt 18 net Vref6
*>> pin io_up 25 net Vref7
*>> pin io_rt 26 net Vref8

.subckt CSU Vcsu_in1 Vcsu_in2 Vcsu_out Vcomp_out
XOTA_cmp Vcsu_in1 Vcsu_in2 Vcomp_out OTA PARAMS: Ib=1.8e-004
Xpmos1 Vdd Vcomp_out Vcomp_sel2 PFET1
Xnmos1 0 Vcomp_out Vcomp_sel2 NFET1
XTGATE1 Vcsu_in1 Vcsu_out Vcomp_out TGATE
XTGATE2 Vcsu_in2 Vcsu_out Vcomp_sel2 TGATE
.ends

.subckt SMPHLD Vsmphld_in Vsmphld_out Vsmphld_phi Vsmphld_phi_inv
XOTA1 Vsmphld_in Vsmphld_in_buf Vsmphld_in_buf OTA PARAMS: Ib=1.8e-004
XTGATE1 Vsmphld_in_buf Vsmphld_c1 Vsmphld_phi TGATE
XOTA2 Vsmphld_c1 Vsmphld_out_c1 Vsmphld_out_c1 OTA PARAMS: Ib=1.8e-004
XC1 Vsmphld_c1 0 C500F
XTGATE2 Vsmphld_out_c1 Vsmphld_c2 Vsmphld_phi_inv TGATE
XOTA3 Vsmphld_c2 Vsmphld_out Vsmphld_out OTA PARAMS: Ib=1.8e-004
XC2 Vsmphld_c2 0 C500F
.ends
```

Xtia_res1 Vbias Vin1 Ib_pp_00 FGOTA PARAMS: Ib=100e-8 valn=800e-9 valp=800e-9

Xtia_res2 Vbias Vin2 Ib_pp_00 FGOTA PARAMS: Ib=100e-8 valn=800e-9 valp=800e-9

Xtia_res3 Vbias Vinp_01 Ib_pp_00 FGOTA PARAMS: Ib=100e-8 valn=800e-9 valp=800e-9

Xtia_res4 Vb_pp_00 Ib_pp_00 Ib_pp_00 FGOTA PARAMS: Ib=100e-8 valn=800e-9 valp=800e-9

XOTA_amp1 Vref1 Ib_pp_00 Vb_pp_00 OTA PARAMS: Ib=1.8e-004

Xtia_res5 Vin1 Vbias Ib_nn_00 FGOTA PARAMS: Ib=100e-8 valn=800e-9 valp=800e-9

Xtia_res6 Vin2 Vbias Ib_nn_00 FGOTA PARAMS: Ib=100e-8 valn=800e-9 valp=800e-9

Xtia_res7 Vbias Vinp_00 Ib_nn_00 FGOTA PARAMS: Ib=100e-8 valn=800e-9 valp=800e-9

Xtia_res8 Vb_nn_00 Ib_nn_00 Ib_nn_00 FGOTA PARAMS: Ib=100e-8 valn=800e-9 valp=800e-9

XOTA_amp2 Vref2 Ib_nn_00 Vb_nn_00 OTA PARAMS: Ib=1.8e-004

Xtia_res9 Vbias Vin1 Ib_pn_01 FGOTA PARAMS: Ib=100e-8 valn=800e-9 valp=800e-9

Xtia_res10 Vin2 Vbias Ib_pn_01 FGOTA PARAMS: Ib=100e-8 valn=800e-9 valp=800e-9

Xtia_res11 Vbias Vinp_11 Ib_pn_01 FGOTA PARAMS: Ib=100e-8 valn=800e-9 valp=800e-9

Xtia_res12 Vb_pn_01 Ib_pn_01 Ib_pn_01 FGOTA PARAMS: Ib=100e-8 valn=800e-9 valp=800e-9

XOTA_amp3 Vref3 Ib_pn_01 Vb_pn_01 OTA PARAMS: Ib=1.8e-004

Xtia_res13 Vin1 Vbias Ib_np_01 FGOTA PARAMS: Ib=100e-8 valn=800e-9 valp=800e-9

Xtia_res14 Vbias Vin2 Ib_np_01 FGOTA PARAMS: Ib=100e-8 valn=800e-9 valp=800e-9

Xtia_res15 Vbias Vinp_10 Ib_np_01 FGOTA PARAMS: Ib=100e-8 valn=800e-9 valp=800e-9

Xtia_res16 Vb_np_01 Ib_np_01 Ib_np_01 FGOTA PARAMS: Ib=100e-8 valn=800e-9 valp=800e-9

XOTA_amp4 Vref4 Ib_np_01 Vb_np_01 OTA PARAMS: Ib=1.8e-004

Xtia_res17 Vbias Vin1 Ib_pp_10 FGOTA PARAMS: Ib=100e-8 valn=800e-9 valp=800e-9

Xtia_res18 Vbias Vin2 Ib_pp_10 FGOTA PARAMS: Ib=100e-8 valn=800e-9 valp=800e-9

Xtia_res19 Vbias Vinp_00 Ib_pp_10 FGOTA PARAMS: Ib=100e-8 valn=800e-9 valp=800e-9

Xtia_res20 Vb_pp_10 Ib_pp_10 Ib_pp_10 FGOTA PARAMS: Ib=100e-8 valn=800e-9 valp=800e-9

XOTA_amp5 Vref5 Ib_pp_10 Vb_pp_10 OTA PARAMS: Ib=1.8e-004

Xtia_res21 Vin1 Vbias Ib_nn_10 FGOTA PARAMS: Ib=100e-8 valn=800e-9 valp=800e-9

Xtia_res22 Vin2 Vbias Ib_nn_10 FGOTA PARAMS: Ib=100e-8 valn=800e-9 valp=800e-9

Xtia_res23 Vbias Vinp_01 Ib_nn_10 FGOTA PARAMS: Ib=100e-8 valn=800e-9 valp=800e-9

Xtia_res24 Vb_nn_10 Ib_nn_10 Ib_nn_10 FGOTA PARAMS: Ib=100e-8 valn=800e-9 valp=800e-9

XOTA_amp6 Vref6 Ib_nn_10 Vb_nn_10 OTA PARAMS: Ib=1.8e-004

Xtia_res25 Vbias Vin1 Ib_pn_11 FGOTA PARAMS: Ib=100e-8 valn=800e-9 valp=800e-9

Xtia_res26 Vin2 Vbias Ib_pn_11 FGOTA PARAMS: Ib=100e-8 valn=800e-9 valp=800e-9

Xtia_res27 Vbias Vinp_10 Ib_pn_11 FGOTA PARAMS: Ib=100e-8 valn=800e-9 valp=800e-9

Xtia_res28 Vb_pn_11 Ib_pn_11 Ib_pn_11 FGOTA PARAMS: Ib=100e-8 valn=800e-9 valp=800e-9

XOTA_amp7 Vref7 Ib_pn_11 Vb_pn_11 OTA PARAMS: Ib=1.8e-004

Xtia_res29 Vin1 Vbias Ib_np_11 FGOTA PARAMS: Ib=100e-8 valn=800e-9 valp=800e-9

Xtia_res30 Vbias Vin2 Ib_np_11 FGOTA PARAMS: Ib=100e-8 valn=800e-9 valp=800e-9

Xtia_res31 Vbias Vinp_11 Ib_np_11 FGOTA PARAMS: Ib=100e-8 valn=800e-9 valp=800e-9

Xtia_res32 Vb_np_11 Ib_np_11 Ib_np_11 FGOTA PARAMS: Ib=100e-8 valn=800e-9 valp=800e-9

XOTA_amp8 Vref8 Ib_np_11 Vb_np_11 OTA PARAMS: Ib=1.8e-004

XCSU1 Vb_pp_00 Vb_nn_00 Vp_00 Vout_00 CSU

XCSU2 Vb_pn_01 Vb_np_01 Vp_01 Vout_01 CSU

XCSU3 Vb_pp_10 Vb_nn_10 Vp_10 Vout_10 CSU

XCSU4 Vb_pn_11 Vb_np_11 Vp_11 Vout_11 CSU

Xpmos_sw1 Vdd Vphi Vphi_inv PFET1

Xnmos_sw2 0 Vphi Vphi_inv NFET1

XSMPHLD1 Vp_00 Vp_00_sh Vphi Vphi_inv SMPHLD

XSMPHLD2 Vp_01 Vp_01_sh Vphi Vphi_inv SMPHLD

XSMPHLD3 Vp_10 Vp_10_sh Vphi Vphi_inv SMPHLD

XSMPHLD4 Vp_11 Vp_11_sh Vphi Vphi_inv SMPHLD

Xpmos_sw3 Vdd Vphi_init Vphi_init_inv PFET1

Xnmos_sw4 0 Vphi_init Vphi_init_inv NFET1

XTGATE_sw1 Vinit1 Vinp_00 Vphi_init TGATE

XTGATE_sw2 Vp_00_sh Vinp_00 Vphi_init_inv TGATE

XTGATE_sw3 Vinit2 Vinp_01 Vphi_init TGATE

XTGATE_sw4 Vp_01_sh Vinp_01 Vphi_init_inv TGATE

XTGATE_sw5 Vinit2 Vinp_10 Vphi_init TGATE

XTGATE_sw6 Vp_10_sh Vinp_10 Vphi_init_inv TGATE

XTGATE_sw7 Vinit2 Vinp_11 Vphi_init TGATE

XTGATE_sw8 Vp_11_sh Vinp_11 Vphi_init_inv TGATE

XCSU_ptr1 Vp_00 Vp_01 Vp_ptr1 Vout_ptr1 CSU

XCSU_ptr2 Vp_10 Vp_11 Vp_ptr2 Vout_ptr2 CSU

XOTA_ptr3 Vp_ptr1 Vp_ptr2 Vout_ptr3 OTA PARAMS: Ib=1.8e-004

*OUTPORT outvector

*>> pin io_dn 4 net Vout_00

*>> pin io_dn 6 net Vout_01

*>> pin io_dn 24 net Vout_10

*>> pin io_dn 26 net Vout_11

*>> pin io_dn 0 net Vp_00

*>> pin io_dn 2 net Vp_01

*>> pin io_dn 21 net Vp_10

*>> pin io_dn 22 net Vp_11

*>> pin io_dn 8 net Vout_ptr1

*>> pin io_up 11 net Vout_ptr2

*>> pin io_dn 9 net Vout_ptr3

*>> pin io_up 4 net Vp_00_sh

*>> pin io_dn 12 net Vp_01_sh

*>> pin io_up 6 net Vp_10_sh

*>> pin io_up 22 net Vp_11_sh

*>> devicefile rasp2_9a_rev2.dev

REFERENCES

[1] R. Chawla, A. Bandyopadhyay, V. Srinivasan, and P. Hasler, "A 531 nW/MHz, 128x32 current-mode programmable analog vector-matrix multiplier with over 2 decades of linearity," in *IEEE Custom Integrated Circuits Conference*, pp. 651-654, 2004.

[2] T. S. Hall, C. M. Twigg, J. D. Gray, P. Hasler, D. V. Anderson, "Large-scale field-programmable analog arrays for analog signal processing," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 52, no. 11, pp. 2298-2307, 2005.

[3] E. K. F. Lee, P. G. Gulak, "A CMOS field-programmable analog array," *IEEE Journal of Solid-State Circuits*, vol. 26, no. 12, pp. 1860-1867, 1991.

[4] H. Kutuk, K. Sung-Mo, "A field-programmable analog array (FPAA) using switched-capacitor techniques," in *IEEE International Symposium on Circuits and Systems*, pp. 41-44, 1996.

[5] E. K. F. Lee, W. L. Hui, "A novel switched-capacitor based field-programmable analog array architecture," *Analog Integrated Circuits and Signal Processing*, vol. 17, no. 1-2, pp. 35-50, 1998.

[6] E. K. F. Lee, P. G. Gulak, "A transconductor-based field-programmable analog array," in *IEEE International Solid-State Circuits Conference*, pp. 198-199, 1995.

[7] B. Pankiewicz, M. Wojcikowski, S. Szczepanski, S. Yichuang, "A field programmable analog array for CMOS continuous-time OTA-C filter applications," *IEEE Journal of Solid-State Circuits*, vol. 37, no. 2, pp. 125-136, 2002.

[8] A. Basu, C.M. Twigg, S. Brink, P. Hasler, C. Petre, S. Ramakrishnan, S. Koziol, and C. Schlottmann, "Rasp 2.8: A new generation of floatinggate based field programmable analog array," in *IEEE Custom Integrated Circuits Conference*, pp. 213-216, 2008.

[9] J. Becker, F. Henrici, S. Trendelenburg, M. Ortmanns, Y. Manoli, "A field-programmable analog array of 55 digitally tunable OTAs in a hexagonal

lattice," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 12, pp. 2759-2768, 2008.

[10] F. Henrici, J. Becker, S. Trendelenburg, D. DeDorigo, M. Ortmanns, Y. Manoli, "A field programmable analog array using floating gates for high resolution tuning," in *IEEE International Symposium on Circuits and Systems*, pp. 265-268, 2009.

[11] A. Stoica, D. Keymeulen, M. Mojarradi, R. Zebulum, T. Daud, "Progress in the development of field programmable analog arrays for space applications," in *IEEE Aerospace Conference*, pp. 1-9, 2008.

[12] D. Keymeulen, A. Stoica, R. Zebulum, S. Katkoori, P. Fernando, H. Sankaran, M. Mojarradi, T. Daud, "Self-reconfigurable analog array integrated circuit architecture for space applications," in *NASA/ESA Conference on Adaptive Hardware and Systems*, pp. 83-90, 2008.

[13] J. Proakis, *Digital Communications*. McGraw-Hill, 2000.

[14] B. Sklar, *Digital Communications: Fundamentals and Applications*. Prentice Hall, 2001.

[15] J. R. Barry, D. G. Messerschmitt, E. A. Lee, *Digital Communication*. Springer, 2003.

[16] R. G. Gallager, *Low-Density Parity-Check Codes*. MIT Press, 1963.

[17] D. J. C. MacKay, R. M. Neal, "Near Shannon limit performance of low density parity check codes," *Electronics Letters*, vol. 32, no. 18, pp. 1645, 1996.

[18] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Transaction on Information Theory*, vol. 20, no. 2, pp. 284–287, 1974.

[19] J. Hagenauer, P. Hoeher, "A Viterbi algorithm with soft-decision outputs and its applications," in *IEEE Global Telecommunications Conference*, pp. 1680-1686, 1989.

[20] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo codes," in *IEEE International Conference on Communications*, pp. 1064–1070, 1993.

[21] D. Chang, S. Lee, "Design techniques for a low-power low-cost CMOS A/D converter," *IEEE Journal of Solid-State Circuits*, vol. 33, no. 8, pp. 1244-1248, 1998.

[22] D. Miyazaki, S. Kawahito, M. Furuta, “A 10-b 30-MS/s low-power pipelined CMOS A/D converter using a pseudodifferential architecture,” *IEEE Journal of Solid-State Circuits*, vol. 38, no. 2, pp. 369-373, 2003.

[23] C. Lin, B. Liu, “A new successive approximation architecture for low-power low-cost CMOS A/D converter,” *IEEE Journal of Solid-State Circuits*, vol. 38, no. 1, pp. 54-62, 2003.

[24] C. Chen, M.Q. Le, K. Kim, “A low power 6-bit Flash ADC with reference voltage and common-mode calibration,” in *IEEE Symposium on VLSI Circuits*, pp. 12-13, 2008.

[25] B. Li, Z. Li, T. Chen, “Research on the low power dissipation of pipeline ADC,” in *International Conference on Innovative Computing, Information and Control*, p. 498, 2007.

[26] R. Dlugosz, K. Iniewski, “Ultra low power current-mode algorithmic analog-to-digital converter implemented in 0.18 μm CMOS technology for wireless sensor network,” in *International Conference on Mixed Design of Integrated Circuits and System*, pp. 401-406, 2006.

[27] M. Lehne, S. Raman, "An analog/mixed-signal FFT processor for wideband OFDM systems," in *IEEE Sarnoff Symposium*, pp. 1-4, 2006.

[28] M. Lehne, S. Raman, "A prototype analog/mixed-signal fast fourier transform processor IC for OFDM receivers," in *IEEE Radio and Wireless Symposium*, pp. 803-806, 2008.

[29] N. Sadeghi, V. C. Gaudet, C. Schlegel, "Analog DFT processors for OFDM Receivers: Circuit mismatch and system performance analysis," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 56, no. 9, pp. 2123-2131, 2009.

[30] E. Afshari, H. S. Bhat, A. Hajimiri, "Ultrafast analog Fourier transform using 2-D LC lattice," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 55, no. 8, pp. 2332-2343, 2008.

[31] Anadigm, AN231E04 Datasheet Rev. 1.0, Dynamically reconfigurable dpASP, 2007.

[32] Lattice Semiconductor, ispPAC81 Datasheet, In-system programmable analog circuit, 2001.

[33] L. Bin, T. W. Rondeau, J. H. Reed, C. W. Bostian, "Analog-to-digital converters," *IEEE Signal Processing Magazine*, vol. 22, no. 6, pp. 69-77, 2005.

[34] M. Kucic, A. Low, P. Hasler, J. Neff, "A programmable continuous-time floating-gate Fourier processor," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 48, no. 1, pp. 90-99, 2001.

[35] F. Baskaya, S. Reddy, Sung Kyu Lim, and D.V. Anderson, "Placement for large-scale floating-gate field-programable analog arrays," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 14, no. 8, pp. 906-910, 2006.

[36] A. Basu, P. E. Hasler, "A fully integrated architecture for fast and accurate programming of floating gates over six decades of current," *IEEE Transactions on Very Large Scale Integration Systems*, vol. PP, no. 99, pp. 1-11, 2010.

[37] D. W. Graham, E. Farquhar, B. Degnan, C. Gordon, P. Hasler, "Indirect programming of floating-gate transistors," *IEEE Transactions on Circuits and Systems I*, vol. 54, no. 5, pp. 951-963, 2007.

[38] J. Hagenauer, M. Moerz, A. Schaefer, "Analog decoders and receivers for high speed applications," in *IEEE International Zurich Seminar on Broadband Communications*, pp. 3-1-3-8, 2002.

[39] H. A. Loeliger, F. Lustenberger, M. Helfenstein, F. Tarkoy, "Probability propagation and decoding in analog VLSI," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 837-843, 2001.

[40] A. F. Mondragon-Torres, E. Sanchez-Sinencio, K. R. Narayanan, "Floating-gate analog implementation of the additive soft-input soft-output decoding algorithm," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 50, no. 10, pp. 1256-1269, 2003.

[41] A. Viterbi, “Error bounds for convolutional codes and an asymptotically optimum decoding algorithm,” *IEEE Transactions on Information Theory*, vol. 13, no. 2, pp. 260-269, 1967.

[42] J. Heller, I. Jacobs, “Viterbi decoding for satellite and space communication,” *IEEE Transactions on Communication Technology*, vol. 19, no. 5, pp. 835-848, 1971.

[43] G. Forney, “The Viterbi algorithm,” *Proceedings of the IEEE*, vol. 61, no. 3, pp. 268-278, 1973.

[44] A. Acampora, R. Gilmore, “Analog Viterbi Decoding for High Speed Digital Satellite Channels,” *IEEE Transactions on Communications*, vol. 26, no. 10, pp. 1463-1470, 1978.

[45] A. Demosthenous, J. Taylor, "A 100-Mb/s 2.8-V CMOS current-mode analog Viterbi decoder," *IEEE Journal of Solid-State Circuits*, vol. 37, no. 7, pp. 904-910, 2002.

[46] A. Demosthenous, J. Taylor, "Effects of signal-dependant errors on the performance of switched-current Viterbi decoders," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 48, no. 10, pp. 1225–1228, 2001.

[47] K. He, G. Cauwenberghs, "An area-efficient analog VLSI architecture for state-parallel Viterbi decoding," in *IEEE International Symposium on Circuits and Systems*, pp. 432-435, 1999.

[48] J. Maunu, *et al*. "Mixed-signal Viterbi decoder for a MB-OFDM receiver," in *IEEE International Conference on Ultra-Wideband*, pp. 121-124, 2008.

[49] H. Kobayashi, D. Tang, "Application of partial-response channel coding to magnetic recording system," IBM Journal of Research and Development, vol. 14, no. 4, pp. 368-375, 1970.

[50] T. Matthews, R. Spencer, "An analog CMOS Viterbi detector for digital magnetic recording," in *IEEE International Solid-State Circuits Conference*, pp. 214-215, 1993.

[51] M. Shakiba, D. Johns, K. Martin, "BiCMOS circuits for analog Viterbi decoders," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 45, no. 12, pp. 1527–1537, 1998.

[52] S. Haykin, *Adaptive Filter Theory.* Prentice Hall, 2001.

[53] H. Lou, "Implementing the Viterbi algorithm," *IEEE Signal Processing Magazine*, vol. 12, no. 5, pp. 42–52, 1995.

[54] Y. Lee, H. Lou, "Normalization, windowing and quantization of soft-decision Viterbi decoder inputs in CDMA systems," in *IEEE Vehicular Technology Conference*, pp. 221-225, 1999.