THEORY, SYNTHESIS AND IMPLEMENTATION OF CURRENT-MODE CMOS PIECEWISE-LINEAR CIRCUITS USING MARGIN PROPAGATION

By

Ming Gu

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

Electrical Engineering

2012

ABSTRACT

THEORY, SYNTHESIS AND IMPLEMENTATION OF CURRENT-MODE CMOS PIECEWISE-LINEAR CIRCUITS USING MARGIN PROPAGATION

By

Ming Gu

Achieving high energy-efficiency is a key requirement for many emerging smart sensors and portable computing systems. While digital signal processing (DSP) has been the de-facto technique for implementing ultra-low power systems, analog signal processing (ASP) provides an attractive and alternate approach that can not only achieve high energy efficiency but also high computational density. Conventional ASP techniques are based on a top-down design approach, where proven mathematical principles and related algorithms are mapped and emulated using computational primitives inherent in the device physics. An example being the translinear principle, which is the state-of-the-art ASP technique, that uses the exponential current-to-voltage characteristics for designing ultra-low-power analog processors. However, elegant formulations could result from a bottom-up approach where device and bias independent computational primitives (e.g. current and charge conservation principles) are used for designing ”approximate” analog signal processors. The hypothesis of this proposal is that many signal processing algorithms exhibit an inherent calibration ability due to which their performance remains unaffected by the use of ”approximate” analog computing techniques.

In this research, we investigate the theory, synthesis and implementation of high performance analog processors using a novel piecewise-linear (PWL) approximation algorithm called margin propagation (MP). MP principle utilizes only basic conservation laws of physical quantities (current, charge, mass, energy) for computing and therefore is scalable across devices (silicon, MEMS, microfluidics). However, there are additional advantages of MP-based processors when implemented using CMOS current-mode circuits, which includes: 1) bias-scalability and robust to variations in environmental conditions (e.g. temperature); and 2) improved dynamic range and faster convergence as compared to the translinear implementations. We verify our hypothesis using two ASP applications: (a) design of high-performance analog low-density parity check (LDPC) decoders for applications in sensor networks; and (b) design of ultra-low-power analog support vector machines (SVM) for smart sensors. Our results demonstrate that an algorithmic framework for designing margin propagation (MP) based LDPC decoders can be used to trade-off its BER performance with its energy efficiency, making the design attractive for applications with adaptive energy-BER constraints. We have verified this trade-off using an analog current-mode implementation of an MP-based (32,8) LDPC decoder. Measured results from prototypes fabricated in a 0.5 μm CMOS process show that the BER performance of the MP-based decoder outperforms a benchmark state-of-the-art min-sum decoder at SNR levels greater than 3.5 dB and can achieve energy efficiencies greater than $100pJ$/bit at a throughput of 12.8 Mbps.

In the second part of this study, MP principle is used for designing an energy-scalable SVM whose power and speed requirements can be configured dynamically without any degradation in performance. We have verified the energy-scaling property using a current-mode implementation of an SVM operating with 8 dimensional feature vectors and 18 support vectors. The prototype fabricated in a 0.5 μm CMOS process has integrated an array of floating gate transistors that serve as storage for up to 740 SVM parameters as well as novel circuits that have been designed for interfacing with an external digital processor. These include a novel current-input current-output logarithmic amplifier circuit that can achieve a dynamic range of 120dB while consuming nanowatts of power and a novel varactor based temperature compensated floating-gate memory that demonstrates a superior programming range than other competitors.

I dedicate this dissertation to my husband, Wei Wang, for his love and support.

ACKNOWLEDGMENT

I am deeply grateful to Dr. Shantanu Chakrabartty for his advice during my Ph.D studies. I owe my successful transition from control theory to analog integrated circuit design to his guidance, encouragement, support, and constant assurance. As an academic advisor, he was always approachable and ready to help in every possible way to make sure I was always at ease while working on my research. I wish him well for all his future endeavors and would be happy to cooperate with him in the future.

Many thanks to Dr. Hayder Radha, Dr. Tongtong Li and Dr. Jonathan I. Hall for being on my Ph.D committee, for their feedbacks, suggestions, and continual interest in my progress. Thanks to Dr. Hayder Radha for helping me in our collaborated project and the instructions on information theory. Thanks to Dr. Tongtong Li and Dr. Jonathan I. Hall in helping me in communication theory and error-correcting coding.

I would like to thank my lab mates Ravi Krishna Shaga, Tao Feng, Pikul Sarkar and Thamira Hindo. I really enjoy the opportunities to work with them and have a good time with them during my graduate studies.

I am grateful to my friends Rui Lin, Xiaoqin Tang, Lei Zhang, Kiran Misra, Hao Wen and Yun Li for their friendship, support, and motivation.

My special thanks go to my husband, Dr. Wei Wang, for his support during my graduate studies. His love, support, and encouragement have carried me to where I am today. We met, got to know each other, fell in love and got married at MSU. The time we spent together at MSU pursuing our Ph.Ds is the best part in our lives. I would dedicate this dissertation to my parents, and grandparents for their support all the time.

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

Digital signal processing (DSP) has been prevalent in all computing systems for the past five decades. Since its beginnings in early 1950's, the progress in the field of DSP has paralleled the growth of digital computers, as signified by the celebrated Moore's law [9]. The key milestones in the area of DSP include the development of the Fast Fourier Transform (FFT) in 1965 [10] followed by the implementation of DSP algorithms on integrated circuit (IC) technology in 1980's. The use of analog signal processing (ASP) can be traced back to an earlier time than its digital counterpart, primarily driven by the need to process naturally occurring signals (image, speech) which are inherently "analog". However, due to rapid progress in digital IC technology and due to the limitations imposed by analog artifacts, in 1970's analog computers were gradually superseded by digital computers. Many experts even predicted the demise of ASP in early 1980s. However, the interest in ASP was renewed a decade ago, primarily due to the need for more energy efficient and high-density signal processing systems. The inspiration for achieving this using ASP techniques came from biological computing systems which are inherently "analog". Biological neural processors excel at solving hard problems in sensory signal processing (video and speech) by sustaining high computational throughput while keeping energy dissipation at a minimal level. At the core of these neural processors and similarly analog signal processors are massively parallel circuits which exploit computational primitives inherent in device physics to achieve high energy-efficiency and high computational density. In this dissertation, we will follow the ASP principle as a guideline to investigate energy efficient computing systems.

From a practical point-of-view there are additional advantages of ASP over DSP techniques which are listed below:

(a) Practical limitations in the analog-to-digital converters (ADC) design.

Since all the naturally occurring signals are analog, ADC is the essential element attached to DSP systems. Many DSP applications are required to recover signals over a wide dynamic range or from a noisy background, which necessitate high speed and high resolution ADC. However, the design of ADC for high resolution/high performance and low power dissipation is one of the challenges nowadays. It seems that the scaling factor is not affecting ADC as much as DSP efficiency. Resolution of ADCs has been increased at 1.5 bits/5 years [11]. As a consequence, the long design time of ADC has become an increasingly severe constraint for the system design as well.

One efficient solution is the cooperative analog/digital signal processing (CADSP), proposed by Hasler in [2, 3]. By utilizing ASP at the front-end, the ADC complexity is reduced significantly, and hence the overall system complexity [2, 3]. Fig. 1.1 shows the tradeoff between CADSP and DSP.

(b) The limited power consumption.

Nowadays, an increasing desire for portability has been posed on electronic devices. A low power dissipation is attractive and crucial in these applications to obtain a long battery

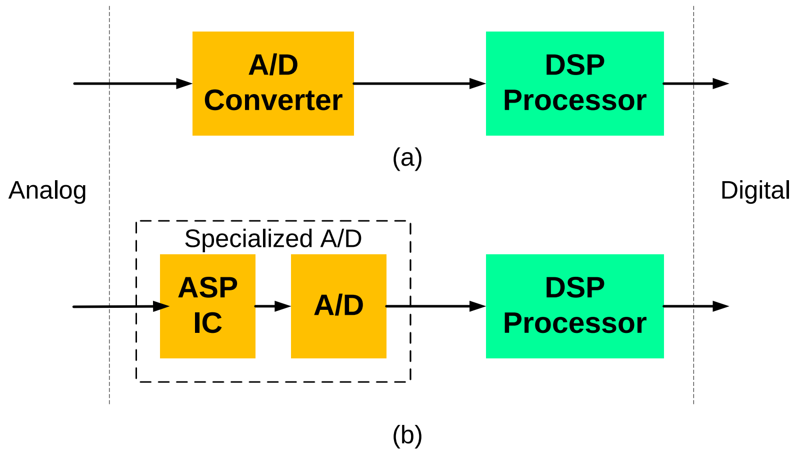


Figure 1.1: Illustration of the tradeoffs in (a)DSP and (b)CADSP systems [2, 3]. For interpretation of the references to color in this and all other figures, the reader is referred to the electronic version of this dissertation.

life. Power consumption of DSP, measured in $mW/MIPS$, reduces by half around every 18 months [4]. This has been keeping pace with Moore's law [9]. Even so, a fixed power budget still constrains the increasing proliferation of portable electronics.

In contrast, ASP tends to be more power efficient. Since ASP block lacks ADC as shown in Fig. 1.1, which has been proved to be a major energy consumer, especially for the scaled-down system. Table 1.1 lists all the key elements power consumption of a state-of-the-art 0.6-μm CMOS image sensor with mixed-signal processor array [1]. It can be seen that the power consumption of ADC is a major factor amongst all and occupies more than one-third of the total amount.

Furthermore, the arithmetic unit of DSP, which consists of digital multipliers and adders, also consumes more energy than ASP. The main cause is less number of transistors used in the case of ASP than DSP. As a consequence, even custom digital solutions are inadequate for ultralow-power applications [12].

In Fig. 1.2 the power consumption of DSP and ASP are compared. The black square rep-

Table 1.1: Power dissipation of major components for a 0.6 μm mixed-signal CMOS image sensor [1]

Supply voltage	5 V
Processor unit	0.25 mW
Sensor, readout and biasing	2.25 mW
ADC	21 mW
FIFO memory	83 mW

resents the power consumption of a state-of-the-art DSP core [5]. It seems the power consumption of the DSP core follows Gene's law. The star represents the power consumption of an analog, floating-gate integrated chip [6]. And the circuit represents the power consumption of a CMOS analog processor that can be used for long-term, self-powered mechanical usage monitoring [7].

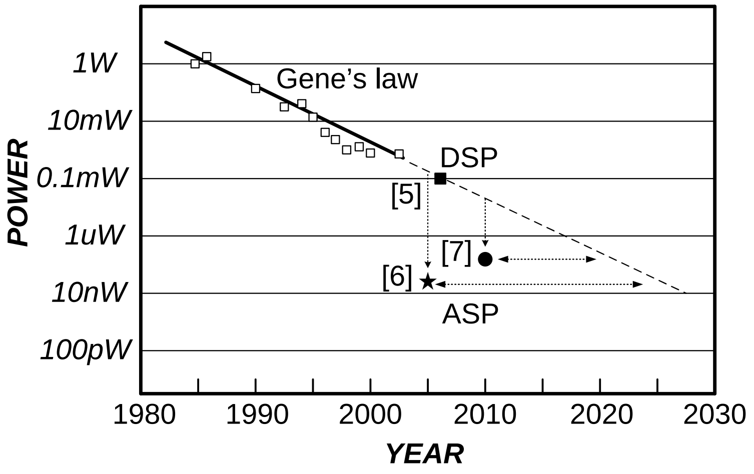


Figure 1.2: Power consumption comparison for DSP [4, 5] and ASP [6, 7].

(c) Size constraints.

Circuit size constraints also favor analog VLSI circuits. It is often possible to perform complex operations on device physics with only a few transistors. Fig. 1.3 shows the comparison of basic circuit schematics for digital/analog adders, and digital/analog multipliers implemented in metal-oxide-semiconductor field-effect transistors (MOSFET's). It is obvious that the number of transistors in digital implementation surpasses that in analog implementation.

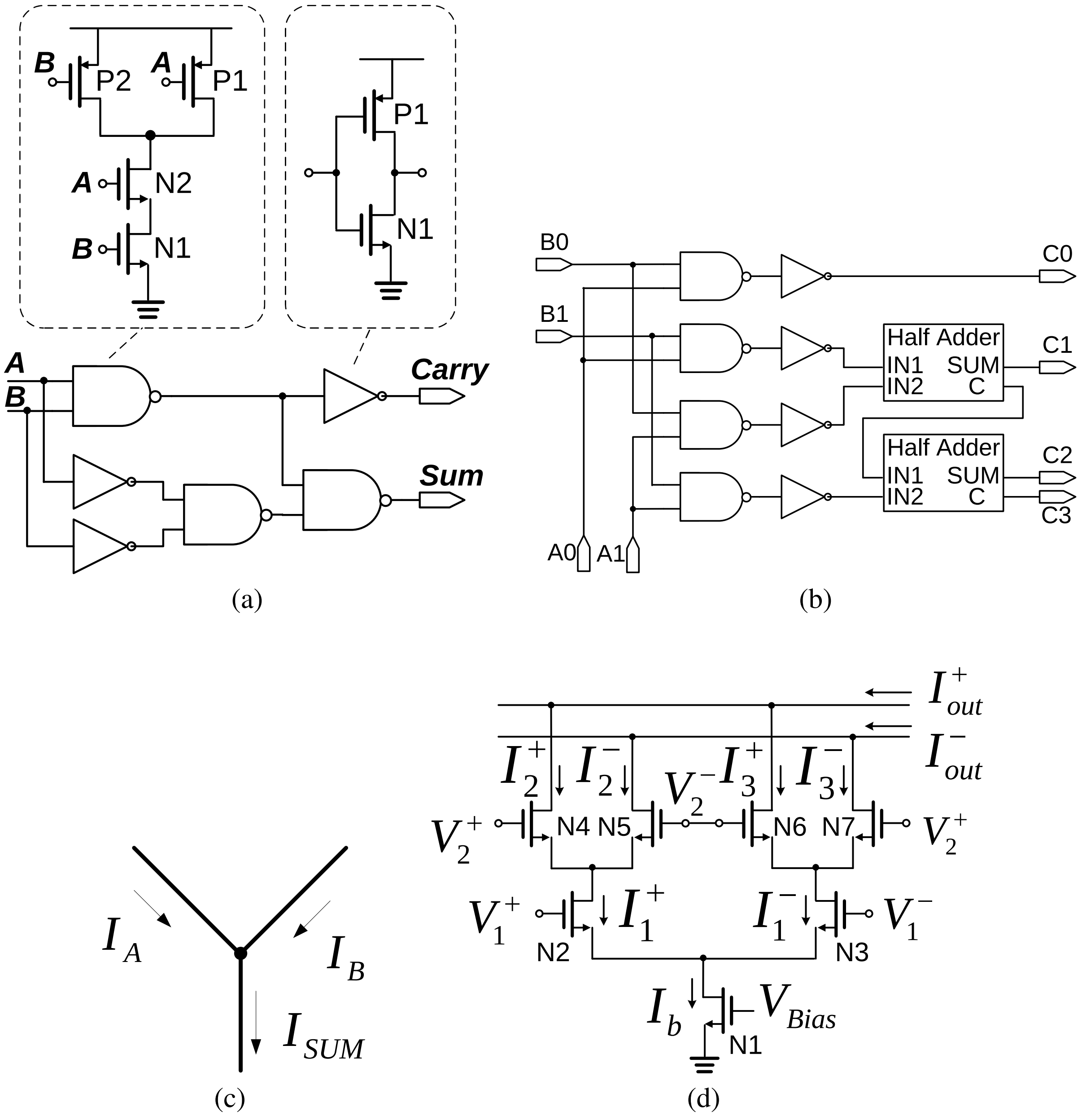


Figure 1.3: Basic CMOS circuits for (a) digital half adder based on NAND and NOT gates; (b) digital multiplier based on (a); (c) analog adder; (d) analog multiplier (Gilbert cell).

(d) Speed requirements.

In recent years great effort has been put in real-time computation, which is applied throughout three-dimensional (3D) graphic systems, high-speed communication system, image sensors, etc.. For these applications, the defect of DSP arise largely from the fact that all the operations in a single arithmetic unit have to be carried out in sequence. In contrast, ASP has a simpler implementation and an inherent parallel architecture for computation. Thus when the signal processing has to be executed under the condition of high speed or high frequency, ASP outperforms DSP on almost all the metrics.

Although there are some crucial reasons for that ASP remains indispensable for scientific computations, there are still some problems existing for ASP. One primary disadvantage is the lack of flexibility. By directly exploiting the computational primitives inherent in the device physics, complicated non-linear functions can be implemented in analog using significantly lower number of transistors compared to its digital counterpart. However, this methodology only works for specific computations. For instance, the subthreshold translinear characteristic of MOS transistor can only be used to solve log-domain computations. Whereas DSP can be used to solve almost any computation by programming.

Another disadvantage lies in the high noise sensitivity for analog implementation. As is known, analog circuits is more vulnerable to noise than digital circuits. It can be explained as that the number in analog computation is represented by continuous voltages or currents, which will be deteriorated by noise easily. Digital system, however, due to the quantization and bi-value logic representation (either one or zero) of discrete signals, is more robust to noise. Only when the noise levels are high enough to flip logical bit, otherwise it works without any effect on precision of the result.

Also the effective resolution of the computing system is a major concern for signal processors. To evaluate the input effort to obtain a designated resolution, in [2, 13], a comprehensive metric "cost" is proposed. It is calculated based on a variety of metrics ranging from the circuit performance metrics like size, power dissipation, etc. to the human effort and other resources involved in the design or fabrication. The comparison of cost versus resolution is described as Fig. 1.4 shows. It can be seen that a "threshold" exists. When the required computation resolution is beyond the threshold, it is less "expensive" to use digital implementation, otherwise, it is more "worthy" to use analog implementation instead of digital.

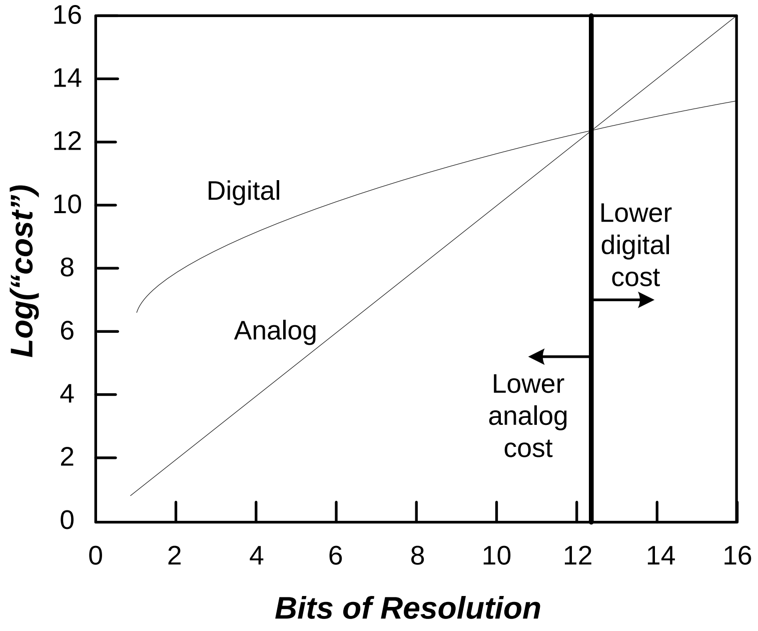


Figure 1.4: Resolution versus "cost" on using ASP or DSP [2].

In summary, ASP is suitable for applications where

- precision/resolution of computation is not required strictly;
- a stringent budget for power, area, or number of transistors is imposed on the system design (e.g. image sensors);

- high-speed or fully-parallel computation is required (e.g. analog decoders);

1.1 Motivation for the proposed research

The underlying principle of analog computation is to represent numbers as physical quantities such as current, voltage, charge, time, etc. and then use the physics of the device for computing (e.g. addition, subtraction, multiplication, etc.). Amongst all the analog computation techniques, translinear analog signal processing (TASP) is the most widely used and the most intensively researched method. Since TASP technique was introduced by Gilbert in 1975 [14], the method has been applied for synthesis of neural, filter and radio-frequency circuits. TASP utilizes the linear relation between the transconductance and the current of the device, and can be readily implemented on MOSFETs and bipolar transistors. In Table 1.2 we summarize different variants and application of the TASP principle that has been reported in the literature:

Table 1.2: Milestones in TASP

Year	Device Characteristic Utilized	Operation Region	Analog Computation Circuits Synthesized
Static TASP			
1975 [14]	$I \propto e^{V}$	weak inversion	multiplier, divider [15], maximum [16]
1991 [17]	$I \propto V^{2}$	strong inversion	square, square-root, polynomial, multiplier-divider [18]
2001 [19]	$I \propto e^{(w_1 V_1 + \cdots + w_N V_N)}$	weak inversion	square, square-root, divider, magnitude of vector
Dynamic TASP			
1979 [20]	$CU_T \dot{I}_{DS} = I_{cap} I_{DS}$	weak inversion	first-order filter
1997 [21]	$CU_T \dot{I}_{DS} = I_{cap} I_{DS}$	weak inversion	RMS-DC converting
1998 [22]	$CU_T \dot{I}_{DS} = I_{cap} I_{DS}$	weak inversion	second-order filter

The advantages of TASP circuits are as follows:

- A high functional density, which explains the extensive application of translinear circuits in neural networks.

- Insensitivity to temperature due to U_T cancelation inside the translinear loop.

- Low-power consumption due to the absence of passive components. In translinear filters implementation, only transistors and capacitors are required.

- A large dynamic range. The trend toward low supply voltages causes a decrease of the dynamic range for filters implemented with conventional circuit techniques, for which companding may be a possible solution. Whereas companding is implicitly performed in translinear filters as a consequence of the exponential behavior of the translinear devices.

- High controllability. Most parameters can be controlled by tuning the current, which is doomed by the translinear principles.

However, there are some disadvantages of TASP circuits when implemented in CMOS. To understand these disadvantages, we first briefly describe the operating regions of a MOS transistor. Fig. 1.5 shows the current-vs-gate voltage response of a typical n-type MOSFET. The characteristic consists of three different operating regions namely: (a) weak inversion: where the current is exponentially dependent on the gate voltage; (b) strong inversion: where the current is quadratically dependent on the gate voltage; and (c) moderate inversion: is the intermediate region between the strong and the weak inversion. Operating in each of these regions have specific advantages which are summarized below:

- In weak inversion region, the minimum power consumption is achieved (low I_D).

- In strong inversion region, the higher operational speed can be obtained (high I_D, high g_m).

- In moderate inversion region, an optimal compromise between power consumption and speed can be achieved.

The power consumption (energy-efficiency), speed trade-off can also be expressed using the "gm/ID" metric [8], where g_m is the input transconductance of the MOS transistor and I_D is the drain current flowing through the transistor. The parameter g_m effectively determines the speed of the transistor and the parameter I_D determines its power consumption. g_m/I_D is chosen as a metric since it reflects the efficiency to translate current (i.e. power) into transconductance. The greater the g_m/I_D value is, the greater the transconductance we obtain at a constant current value [8]. Fig. 1.5 displays the g_m/I_D versus the drain current I_D for MOS transistors under the three operating regions. Fig. 1.5 shows that in weak inversion region, we obtain the highest g_m/I_D, thus the best power efficiency.

As can be observed from table 1.2, TASP circuits are designed to operate in only one specific operating region. Thus it lacks flexibility, or the trade-off capability across the operation regions, which is a severe problem when a high dynamic range is required. This limits the application of TASP circuits in designing more complex and flexible analog signal processors.

To solve this problem, we propose ASP circuits that only depend on the universal conservation principles like the Kirchhoff's current law, as illustrated in Fig. 1.3 (c). By construction, ASP circuits based on universal conservation principles will have several advantages over their TASP counterparts which are:

- Independence on transistor operation regions since its underlying principle follows physical laws (e.g. universal conservation law).

- Wide dynamic range due to the unlimited operation region.

- Diverse ways of implementation (e.g. charge-mode, voltage-mode, time-mode, etc.). For instance, Fig. 1.6 illustrates how reverse water-filling can naturally be implemented using a charge-coupled device. Initially, a fixed amount of charge (shown by shaded area in Fig. 1.6a) is introduced into the charge-coupled device (CCD). As potential barriers (wells) are increased adiabatically charge re-distributes itself (Fig. 1.6b) amongst the wells. At equilibrium (Fig. 1.6c) total charge accumulated in the potential wells (residue) and the normalization level is determined by charge conservation principles. Even though potential wells have been used in the above example the procedure is applicable to other physical quantities (current, mass, energy) implemented by different analog structures (MEMS, microfluidic devices). However, since current-mode circuits is the most easy and mature way of implementation, in this dissertation, we adhere to current-mode implementation.

- Insensitivity to the outside variations (e.g. temperature).

However, the using universal conservation laws restricts the choice of computations that can be precisely implemented. That leads to the main hypothesis of this dissertation: many signal processing algorithms exhibit an inherent calibration ability, and hence their performance remains unaffected by the use of "approximate" analog computing techniques which is proposed in this study. Also, approximations at a circuit level would require a bottom-up and top-down synthesis of signal processing algorithms as is illustrated in Fig. 1.7 and is compared with the conventional top-down design flow for a TASP based synthesis. The chart in Fig. 1.7 shows that many of the approximation techniques have be simulated, verified and calibrated at the algorithmic and system level before prototyping on hardware. However, algorithmic and system levels of the design flow offers more flexibility in implementing a given system and hence can overcome performance limitations due to low-level approximations. In this dissertation, we plan to illustrate this using two specific applications: (a) designing high-performance analog LDPC decoders; and (b) designing ultra energy-efficient analog SVM.

1.2 Scientific contributions

This dissertation is motivated by improving the energy efficiency of computation implementation with ASP while resorting to the physical law to ensure bias-scalability. The major scientific contributions are listed below:

(a) An analog signal processing (ASP) algorithm named margin propagation (MP) as an efficient piece-wise linear (PWL) approximation technique to a "log-sum-exp" function along with its current-mode analog circuit implementation is proposed. Due to its underlying principle of conservative law, MP algorithm can be mapped to various hardware easily. The current-mode MP circuit implementation is designed to be bias-scalable. This characteristic is beneficial for computational circuit since it enables the design adaptive to various applications with different specification requirements.

(b) A wide-range of mathematical functions are synthesized with MP-based bias-scalable analog computational circuits. The synthesized functions include addition, subtraction, multiplication, division, power and polynomial computation, etc.. MP circuits operate in the log-likelihood domain where complex functions such as multiplication, division, square-root operations are mapped into simpler operations like addition, subtraction or scaling, making circuit design less complex. The circuits can operate over a wider dynamic range as the transistors can span different biasing regions.

(c) A 100pJ/bit, (32,8) CMOS analog low-density parity-check (LDPC) decoder based on MP is designed and implemented. One fascinating property is its capability of trading off BER performance with energy efficiency due to the tunable hyper parameter γ in MP algorithm. The prototyped MP-based LDPC decoder can achieve an energy efficiency of 100nJ/bit while an optimal configuration can also deliver up to 3 dB improvement in BER compared to the benchmark min-sum LDPC decoder. This design is helpful for high energy-efficiency high error-correcting performance analog LDPC decoder design.

(d) An analog energy-scalable MP-based support vector machine (SVM) is designed and implemented. The prototype stores 2052 SVM parameters into floating-gate memory array and is fully programmable. Due to the using of MP computation circuit, the prototyped SVM is energy-scalable.

(e) A novel current-input current-output logarithmic amplifier circuit is designed and implemented. This design based on translinear Ohm's law directly generates currents as a logarithmic function of the input current, which exhibits a dynamic range of 120dB and a temperature sensitivity of 230 ppm/K, while consuming less than 100nW of power. This design brings three major benefits for implementing logarithmic amplifier circuit: high dynamic range, low power consumption, and temperature compensation.

(f) A novel varactor driven temperature compensation circuitry of CMOS floating-gate current memory is designed and implemented. By adapting the floating-gate capacitance with voltage controlled capacitor (varactor), this design enables temperature dependent factors be effectively canceled. With this design, a temperature sensitivity of 150 ppm/K is achieved. This design is instructive for implementing high-density, temperature compensated floating-gate current memories.

1.3 Dissertation Organization

The dissertation is organized as follows: Other ASP research work is briefly introduced in chapter 2. The MP algorithm related theories and properties are introduced in detail in chapter 3. The circuit implementation and synthesized analog computation circuits are introduced in chapter 4. The applications of MP-based PWL circuits are described in chapter 5 and 6.

In chapter 2, the device models of MOS transistor are introduced first as background material. Then a thorough survey of state-of-the-art research work in ASP is illustrated, i.e. a survey on translinear analog signal processing circuits.

Chapter 3 presents MP algorithm related theories. Fundamental definitions and mathematical properties of MP algorithm are illustrated first, followed by the proof that MP algorithm is a PWL approximation for "log-sum-exp" function. These definitions and properties serves as the theoretical fundamental for later chapters.

Since MP is a PWL approximation algorithm to "log-sum-exp" function, a wide variety of scientific computation can be synthesized with MP algorithm. In chapter 4, the basic current-mode circuit implementation for MP algorithm is presented. Based on the basic MP circuit block, a variety of analog computation circuits are synthesized, which include addition, subtraction, multiplication, division, power, inner-product, polynomial, and tanh. The PWL approximation effect of MP-based circuits are demonstrated by the simulation results.

Chapter 5 presents an LDPC decoder which is implemented based on the MP circuits. This chapter can be divided into theory sections and implementation sections. The theory sections starts with the metrics to evaluated the performance of LDPC decoders and a significant property of LDPC decoder: performance trade-off. Density evolution is introduced as an analytical tool and utilized to evaluate the message sparsity performance for the MP-based LDPC decoding algorithm and the other two conventional algorithms. The subsequent simulation results verify the density evolution analysis results as well as the BER performance. It is also demonstrated that the trade-off capability of MP-based LDPC decoder can be realized elegantly by just tuning the hyper parameter γ. The implementation sections start with circuit for basic module. Then the systematic architecture is introduced. The test setup and the measurement results are presented.

Chapter 6 presents another application of the MP-based PWL circuits: an analog SVM. In this application, the inner product computation of vector, the addition, multiplication, square of scalar are all synthesized with MP-based PWL circuit. In this chapter, the core computation, the system architecture and primary circuits are introduced in sequence. Finally, the simulation results and the measurement results are demonstrated.

Chapter 7 offers conclusions and outlook.

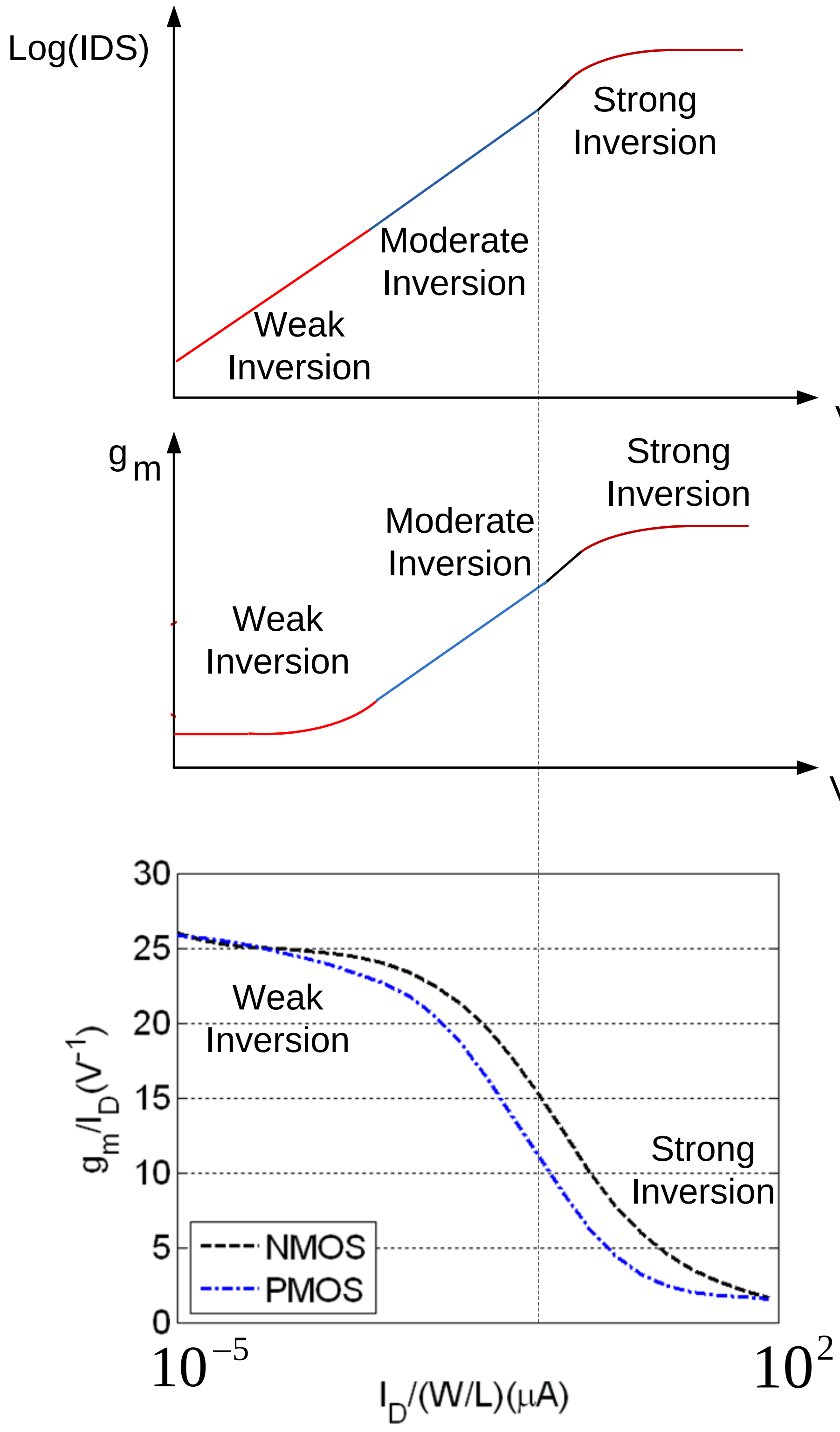
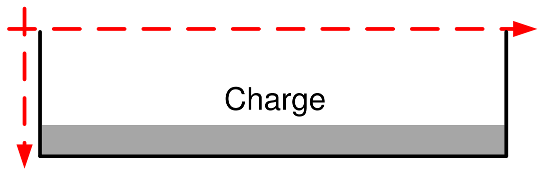
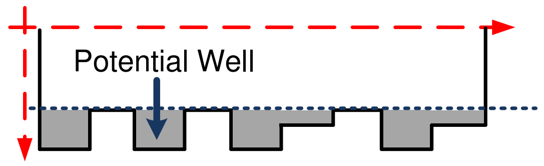


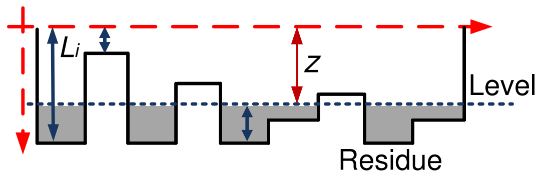
Figure 1.5: $log(I_D)$ versus V_{GS}, g_m versus V_{GS}, and g_m/I_D versus $I_D/(W/L)$ curves for various operation regions [8].



(a)



(b)



(c)

Figure 1.6: Illustration of the proposed concept for designing scalable analog computational systems. The depth of potential wells can be controlled by an external voltage source. At equilibrium the charge re-distributes amongst potential wells to satisfy charge conservation laws.

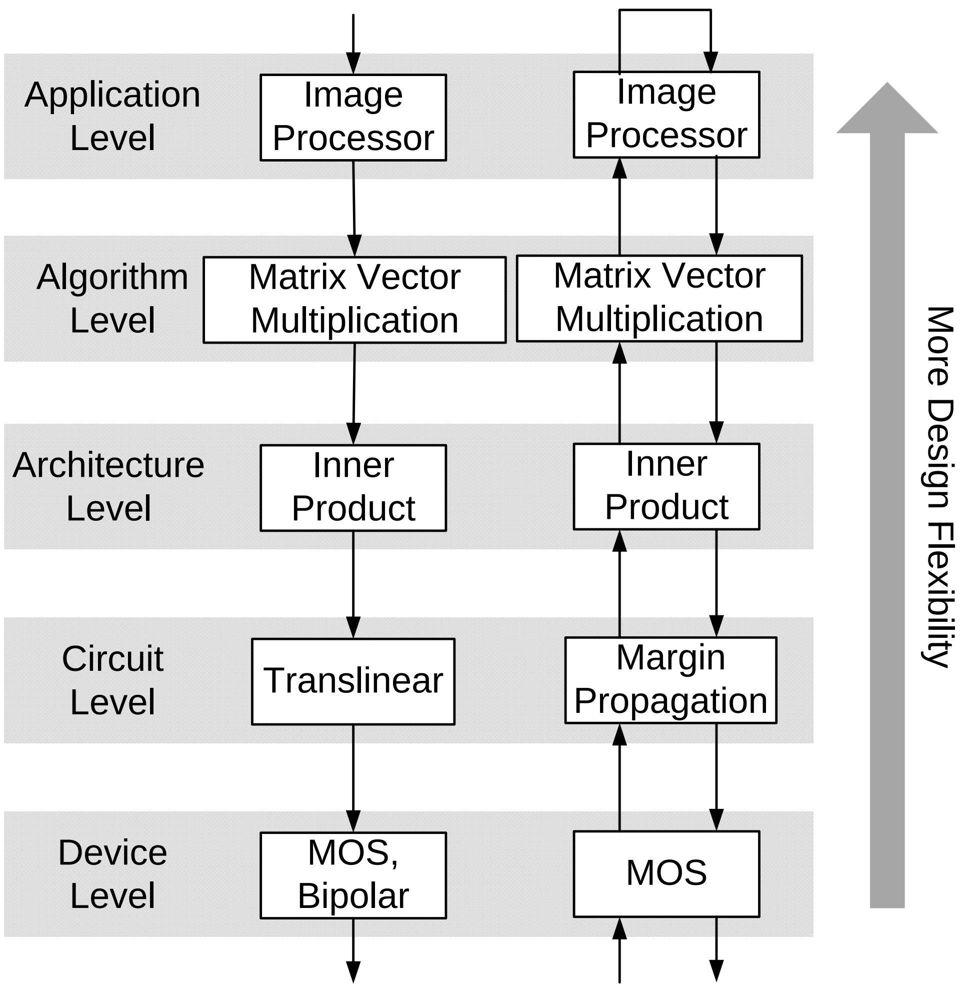


Figure 1.7: Top-down design procedure for conventional TASP and the bottom-up procedure we propose.

Chapter 2

Device Models and Translinear Analog Signal Processing

2.1 Device Models of MOS Transistor

Fig. 2.1 illustrates the terminals of MOS transistors. Since the model of n-type MOS and that of p-type MOS only differ in polarity definition of current flow through the devices, their mathematic models only differs in signs. Thus only NMOS transistors are discussed thereinafter.

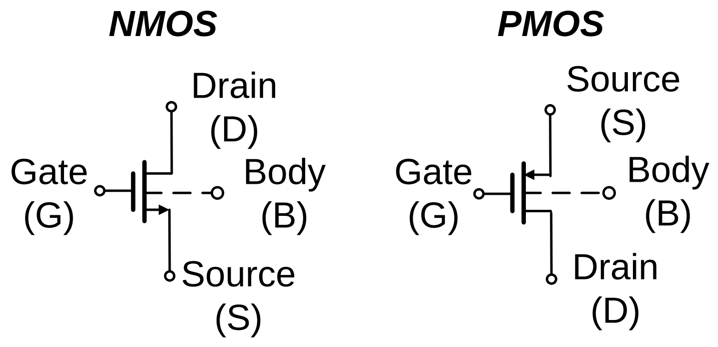


Figure 2.1: MOS symbols

The operation of a MOSFET can be separated into three different modes, depending on the voltages at the terminals. The most frequently used models of MOSFET are for the operation in strong inversion (or called “above threshold operation”) and weak inversion (or called “subthreshold conduction”).

The strong inversion model is described as follows:

$$I_{DS} = \frac{\mu_n C_{ox}}{2\kappa}\frac{W}{L}[\kappa(V_G - V_T) - V_S]^2. \tag{2.1}$$

In equation (2.1), W and L denotes the width and length of transistor respectively. $\frac{W}{L}$ is called the aspect ratio. μ_n represents the mobility of electrons. And C_{ox} denotes the gate oxide capacitance per unit area. V_T represents the threshold voltage. κ is a parameter reflecting the effectiveness of the gate potential in controlling the channel current, which is always approximated as $\kappa \simeq 1$ for strong inversion. According to (2.1), the relationship between the current flow and the voltages can be described as “square”.

The weak inversion model is described as:

$$I_{DS} = I_0 e^{(1-\kappa)V_{BS}/U_T} e^{\kappa V_{GS}/U_T}(1 - e^{-V_{DS}/U_T} + V_{DS}/V_0). \tag{2.2}$$

In this equation, V_{GS} is the gate-to-source potential, V_{DS} is the drain-to-source potential, V_{BS} is the bulk-to-source potential (body effect), I_0 is the zero-bias current for the given device, V_0 is the early voltage, and U_T is the thermodynamic voltage. For devices in subthreshold saturation, which is determined by $V_{DS} \geq 4U_T$, neglecting the early effect and the body effect, equation

(2.2) can be simplified as

$$I_{DS} = I_0 e^{\kappa V_{GS}/U_T}. \tag{2.3}$$

According to (2.3), the current is the exponential of the potential. Also we can deduce the input transconductance in subthreshold saturation region as:

$$g_m = \frac{\partial I_{DS}}{\partial V_{GS}} = \frac{\kappa I_{DS}}{U_T}. \tag{2.4}$$

And the output transconductance in subthreshold saturation region as:

$$g_d = \frac{\partial I_{DS}}{\partial V_{DS}} = \frac{I_{DS}}{V_0}. \tag{2.5}$$

It can be seen that g_m and g_d are linear to the current I_{DS} in this operation region.

There is no uniform model for MOSFET throughout all the operation regions until it was developed by C. C. Enz, F. Krummenacher and E. A. Vittoz. Enz-Krummenacher-Vittoz in 1995, named as EKV model. With EKV model, all the currents, transconductance, the intrinsic capacitance, and thermal noise can be expressed in a continuous way in all operation regions, including weak inversion, moderate inversion, strong inversion, conduction, and saturation. The equation of EKV model is described as

$$I_{DS} = I_S log^2\left[1 + e^{(\kappa(V_G - V_T) - V_S)/2U_T}\right] - I_S log^2\left[1 + e^{(\kappa(V_G - V_T) - V_D)/2U_T}\right]. \tag{2.6}$$

In this equation, I_S is a specific current, which depends essentially on the aspect ratio $\frac{W}{L}$ and the mobility μ_n.

2.2 Translinear Analog Signal Processing

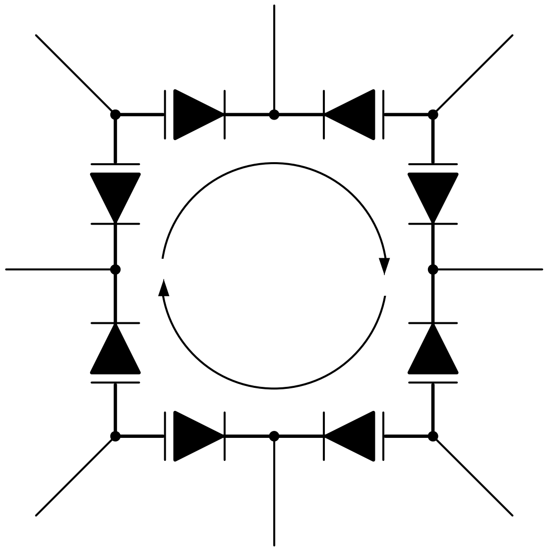


Figure 2.2: An alternating translinear loop of NMOS transistors. Here the number of clockwise facing transistors in junction is the same as the number of counterclockwise facing transistors.

The translinear principle was defined by Gilbert in 1975 [14], originally with bipolar transistors. In translinear circuits, the exponential current-voltage characteristic of bipolar transistor is exploited. With the emergence of MOSFET circuit, translinear principle is extended into MOS circuits, since MOS transistors in subthreshold region also have the exponential current-voltage non-linearity characteristic, as shown in (2.3). The translinear principle states that, in a closed loop of tranlinear devices comprising an equal number of of clockwise facing and counterclockwise facing junction devices, their currents follows:

$$\prod_{j\in CW}\frac{I_j}{S_j}=\prod_{j\in CCW}\frac{I_j}{S_j}, \tag{2.7}$$

where CW and CCW denote the set of junction transistors clockwise facing and counterclockwise facing respectively. I_j denotes the I_{DS} of the transistor. S denotes the aspect ratio, i.e. W/L ratio of the transistor. Fig. 2.2 shows a conceptual alternating translinear loop comprising NMOS transistors.

Note that the translinear principle is derived by Kirchhoff's voltage law for the loop, which is:

$$\sum_{j \in CW} V_{GS(j)} = \sum_{j \in CCW} V_{GS(j)}, \tag{2.8}$$

In 1991, the translinear principle was extended by Seevinck and Wiegerink [17]. They generalized the translinear principle with the statement that the transconductance of a device is linear in the controlling voltage rather than it being linear in the current, as Gilbert intended originally. In this generalized translinear principle, the square-law MOS characteristic, as we show in equation (2.1), is exploited. Correspondingly, the MOS transistors in a translinear loop they presented were biased in strong inversion. And their translinear principle takes the form as

$$\sum_{j \in CW} \sqrt{\frac{I_j}{S_j}} = \sum_{j \in CCW} \sqrt{\frac{I_j}{S_j}}. \tag{2.9}$$

This generalized translinear principle is suggested to be renamed as voltage-translinear circuits to prevent possible confusion [23].

However, there is no translinear principle describing the behavior of translinear loops when any of the transistors enters the moderate inversion region, where the device is in neither weak inversion nor strong inversion, but in between. In this region, the current-voltage characteristic follows neither the exponential form as shown in equation (2.3) nor the square-law form as described in equation (2.1).

The EKV model, which is described in (2.6), provides a model for the current-voltage characteristics of the MOS transistor at all levels of inversion including moderate inversion. In 2008, Minch brought out a generalized translinear principle for alternating loops of saturated MOS transistors that is valid at all levels of inversion based on the EKV model [24]. This generalized translinear principle reduces to the conventional one when all transistors in a translinear loop are biased in weak inversion and reduces to the voltage-translinear principle when all transistors in the loop are biased in strong inversion. They model the current-voltage characteristic of an NMOS transistor with its bulk tied to ground as:

$$I_{sat} = SI_S log^2(1 + e^{(\kappa(V_G - V_{T0}) - V_S)/2U_T}), \tag{2.10}$$

where $S = W/L$ denotes the aspect ratio, $\kappa = C_{ox}/(C_{ox} + C_{dep})$ denotes the subthreshold slope factor, V_{T0} denotes the zero-bias threshold voltage, $U_T = kT/q$ denotes the thermal voltage, and $I_S = 2\mu_n C_{ox} U_T^2/\kappa$ denotes the transistor's specific current. Based on this equation it is explicitly to deduce

$$\kappa(V_G - V_{T0}) - V_S = 2U_T log(e^{\sqrt{I_{sat}/SI_S}} - 1). \tag{2.11}$$

When the transistor is in weak inversion, by expanding $e^{\sqrt{I_{sat}/SI_S}}$ in a Maclaurin series and retaining the constant and linear terms, we obtain the approximation as

$$\kappa(V_G - V_{T0}) - V_S \approx U_T log \frac{I_{sat}}{SI_S}. \tag{2.12}$$

When the transistor is in strong inversion, -1 is negligible compared to $e^{\sqrt{I_{sat}/SI_S}}$, therefore we can obtain

$$\kappa(V_G - V_{T0}) - V_S \approx 2U_T\sqrt{\frac{I_{sat}}{SI_S}}. \tag{2.13}$$

Based on equation (2.12) and (2.13), we can derive translinear principle on weak inversion, shown in (2.9), and voltage-translinear principle on strong inversion, shown in (2.10). And the continuous moderate inversion model is obtained on equation (2.11) as well.

Based on these aforementioned translinear principles, continuous-time translinear analog signal processing (TASP) systems are constructed. TASP circuits can be divided into two major classes: static translinear circuits and dynamic translinear circuits.

Static translinear circuits exploit the exponential relation between voltage and current to realize static linear and nonlinear transfer functions, such as products, quotients, arbitrary fixed power laws, etc. [15, 25, 18, 26]. With such operations, we can implement signal-processing algorithms involving auto-correlation, cross-correlation, signal energy computation, least-mean-square (LMS) error metric computation, and the like [23, 25]. For instance, the Gilbert multiplier [15] shown in Fig. 1.3 (d) is a static translinear circuit. For the circuit of Fig. 1.3 (d):

$$I_1^+ = \frac{I_b}{1+e^{-(V_1^+-V_1^-)/U_T}},\ I_1^- = \frac{I_b}{1+e^{(V_1^+-V_1^-)/U_T}} \tag{2.14}$$

$$I_2^+ = \frac{I_1^+}{1+e^{-(V_2^+-V_2^-)/U_T}},\ I_2^- = \frac{I_1^+}{1+e^{(V_2^+-V_2^-)/U_T}} \tag{2.15}$$

$$I_3^+ = \frac{I_1^-}{1+e^{(V_2^+-V_2^-)/U_T}},\ I_3^- = \frac{I_1^-}{1+e^{-(V_2^+-V_2^-)/U_T}} \tag{2.16}$$

If we denote $(V_1^+ - V_1^-)/U_T = x$, $(V_2^+ - V_2^-)/U_T = y$, we can obtain

$$I_{out}^+ = I_2^+ + I_3^+ = I_b \frac{e^x e^y + 1}{(e^x+1)(e^y+1)} \tag{2.17}$$

$$I_{out}^- = I_2^- + I_3^- = I_b \frac{e^x + e^y}{(e^x+1)(e^y+1)} \tag{2.18}$$

Thus,

$$I_{out}^+ - I_{out}^- = I_b tanh(\frac{x}{2}) tanh(\frac{y}{2}) \tag{2.19}$$

By tailor series expression, here we take $tanh(z) \approx z$. Then

$$I_{out} \approx I_b \cdot \frac{x}{2} \cdot \frac{y}{2} \tag{2.20}$$

$$\propto x \cdot y. \tag{2.21}$$

Another example of static translinear circuit is “winner-take-all” circuit, which can be used to find out the maximum amongst a couple of inputs [16]. A voltage-in-current-out winner-take-all circuit is shown in Fig. 2.3. In this circuit, current flow through each branch follows:

$$I_i = \frac{e^{\kappa V_i/U_T}}{\sum_{j=1}^{N} e^{\kappa V_j/U_T}} \cdot N \cdot I_b \tag{2.22}$$

Due to the fast decay of exponential, at equilibrium, the branch with the maximum voltage takes all the currents and the other branch obtains currents which decays gradually to zero. Consequently, the winner (maximum) is easy to pick out.

There is also an increasing effort to develop new circuit design techniques in order to build

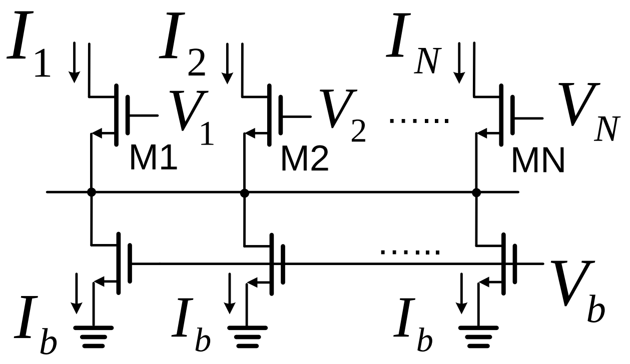


Figure 2.3: A voltage-in-current-out winner-take-all circuit.

analog information-processing systems. In 2001, Minch proposed multiple-input translinear element (MITE) networks [19], which produces an output current that is exponential in a weighted sum of its input voltages. MITEs are simply implemented using multiple-input floating gate MOS (FGMOS) transistors. An MITE is shown in Fig. 2.4. The output current I_{out} follows:

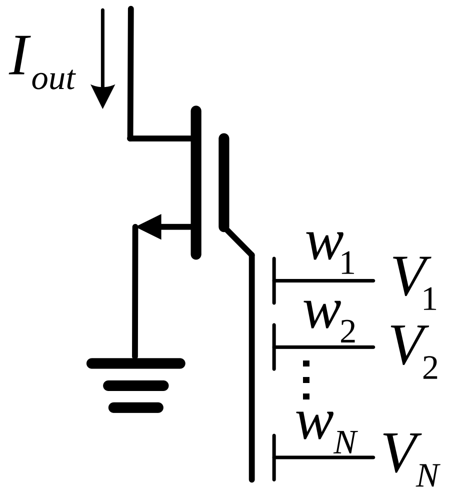


Figure 2.4: A multiple-input translinear element (MITE).

$$I_{out} \propto e^{(w_1 V_1 + w_2 V_2 + \cdots + w_N V_N)}. \tag{2.23}$$

From equation (2.23) it can be seen that if these weights are adjusted according to some learning algorithm, such as the least-mean-square (LMS) algorithm or the recursive least-squares (RLS) algorithm, these built-in weights can be used to implement a variety of linear adaptive signal-processing architectures. In this respect, TASP circuits are also applied into neural systems [27, 28, 29, 30]. Furthermore, these weights can be programmed in a digital fashion to realize reconfigurability in TASP systems. Such reconfigurable analog signal processing systems are called field-programmable analog arrays (FPAAs) [6]. This kind of TASP FPAA is analogous to an E^2PROM-based field programmable gate arrays (FPGA) in digital domain.

Based on MITEs, a variety of static and dynamic translinear circuits can be synthesized (e.g. the vector-magnitude circuit, the vector-normalization circuit, the second-order low-pass filter, and the rms-to-dc converter) [19, 31, 32, 26].

By admitting capacitors in the translinear loops, the dynamic translinear circuits are constructed to deal with frequency dependent transfer functions. The dynamic translinear principle can be explained with reference to the circuit shown in Fig.2.5. The expression for I_{cap} is easily found

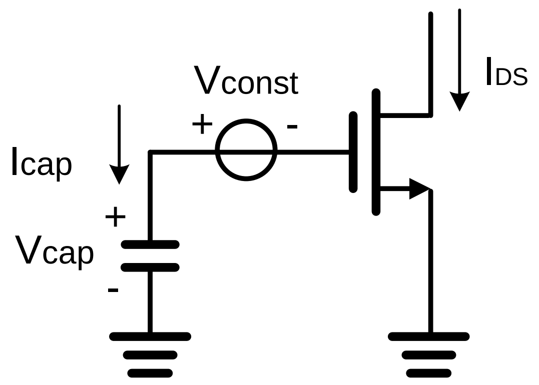


Figure 2.5: Principle of dynamic translinear circuits.

to be

$$I_{cap} = CU_T \frac{\dot{I_{DS}}}{I_{DS}}, \tag{2.24}$$

which shows that I_{cap} is a nonlinear function of I_{DS} and its time derivative $\dot{I_{DS}}$. Equation (2.24) can be rewritten as

$$CU_T \dot{I_{DS}} = I_{cap} I_{DS}, \tag{2.25}$$

which states the dynamic translinear principle: A time derivative of a current can be mapped onto a product of currents [33].

The first dynamic translinear circuit, or log-domain filter, was originally introduced by Adams in 1979 [20]. The first-order filter described in [20] is in fact a translinear circuit. Another milestone is a general class of filters, called exponential state-space filters, which comprise a number of translinear loops, first published by Frey [34]. The exponential state-space filters enables the design of higher order log-domain filters[35, 36]. Since then the interest of dynamic translinear filter increases dramatically. More studies were reported in this area [37, 38, 39, 40, 41, 42].

However, the dynamic translinear principle is not limited to filters, or linear differential equations. It also can be utilized to realize nonlinear differential equations using transistors and capacitors only. In [21], the nonlinear differential equations describing the rms-dc conversion function are implemented on the dynamic translinear principle. Another example is [22], in which a translinear second-order oscillator was implemented. Some other example functions described by nonlinear differential equations and implemented by translinear circuits are phase-locked loops (PLL's) [43], translinear sinusoidal frequency tripler [44], etc..

Chapter 3

Theory of PWL technique based on margin propagation

3.1 Definition of MP algorithm

At the core of MP algorithm is the reverse water-filling procedure, which has been described in detail in [45, 46]. Given a set of scores $L_i \in \mathbb{R}, i = 1, ..., N$, reverse water-filling computes a normalization factor z according to the constraint:

$$\sum_{i=1}^{N} [L_i - z]_+ = \gamma \tag{3.1}$$

where$[\cdot]_+ = max(\cdot, 0)$ denotes a rectification operation and $\gamma \geq 0$ represents a parameter of the algorithm. Computation of z according to constraint (3.1) is summarized in the Algorithm 1. The algorithm recursively computes the factor z such that the net balance of log-likelihood scores L_i in excess of z equals γ.

The solution to equation (3.1) can also be visualized using Figure 3.1, where the cumulative

Algorithm 1 Reverse water-filling procedure to compute the parameter z

```
Require: Set of log-likelihood scores $\{L_i\}, i = 1, .., N$.
Ensure: $z = 0, N = 1, T = 0$
  $a = max\{L_i\}$
  $\{s\} \leftarrow \{L_i\} - \{a\}$
  while $T < \gamma \quad \& \quad m < N$ do
    $b = max\{s\}$
    $T \leftarrow T + M(a - b)$
    $a \leftarrow b$
    $\{s\} \leftarrow \{s\} - \{b\}$
    $m \leftarrow m + 1$
  end while
  $z \leftarrow b + M(\gamma - T)$
```

score beyond the normalization factor z (shown by the shaded area) equals to γ . To be practical, we always limit z in the range of $z \leq \max_i L_i$. The computation of z can be expressed as an equivalent function $M(\mathcal{L}, \gamma)$ whose inputs are a set of log-likelihood scores $\mathcal{L} = \{L_i\}, i = 1, .., N$ of size N and a hyper-parameter γ:

$$z^{mp} = M(\mathcal{L}, \gamma). \tag{3.2}$$

3.2 Properties of MP algorithm

In this section we present some of the key properties of MP function $M(\mathcal{L}, \gamma)$. Since margin approximation uses thresholds $[x]_+ = \max(x, 0)$, we first state two lemmas which will be useful for proving some of the other properties.

Lemma 1 : $\forall a, b \in R$,

$$[a]_+ - [b]_+ \leq [a - b]_+.$$

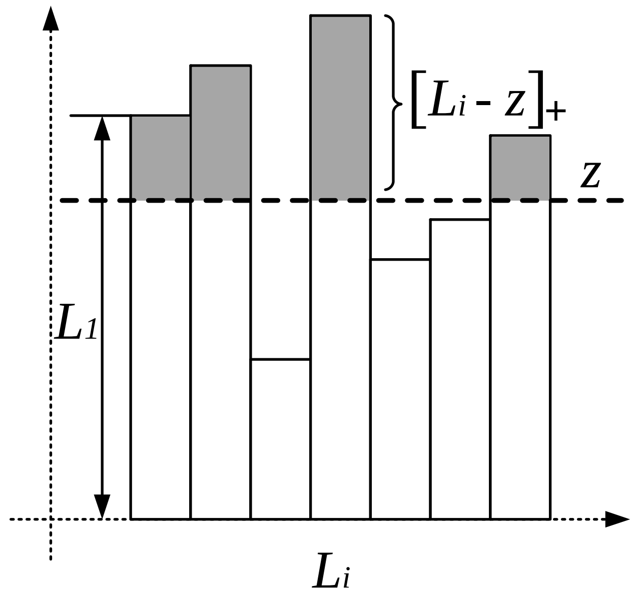


Figure 3.1: Illustration of reverse water-filling procedure

Lemma 2 : $\forall a, b \in R$,

$$[a]_+ + [b]_+ \leq [a+b]_+.$$

The proof of **Lemma 1** and **Lemma 2** is straightforward and can be found in any standard analysis textbook. For the sake of brevity, only the statement and the illustration of some of the properties are presented in this section and all the proofs have been presented in detail in the appendix.

Property 1 (Scaling Property): For any $\alpha \in R$, $\alpha > 0$ and a set of scores $\mathcal{L} = \{L_i\}, i = 1, .., N$

$$M(\alpha\mathcal{L}, \alpha\gamma) = \alpha M(\mathcal{L}, \gamma). \tag{3.3}$$

Here $\alpha\mathcal{L} = \{\alpha L_i\}, i = 1, .., N$. The proof of this property is simple since if the condition

$$\sum_{i=1}^{N} [L_i - z]_+ = \gamma$$

is satisfied, then the following condition

$$\sum_{i=1}^{N} [\alpha L_i - \alpha z]_+ = \alpha\gamma$$

is also satisfied. Figure3.2 illustrates the scaling property of margin approximation function, where the threshold z scales with the scaling of the log-likelihood scores and the hyper-parameter γ.

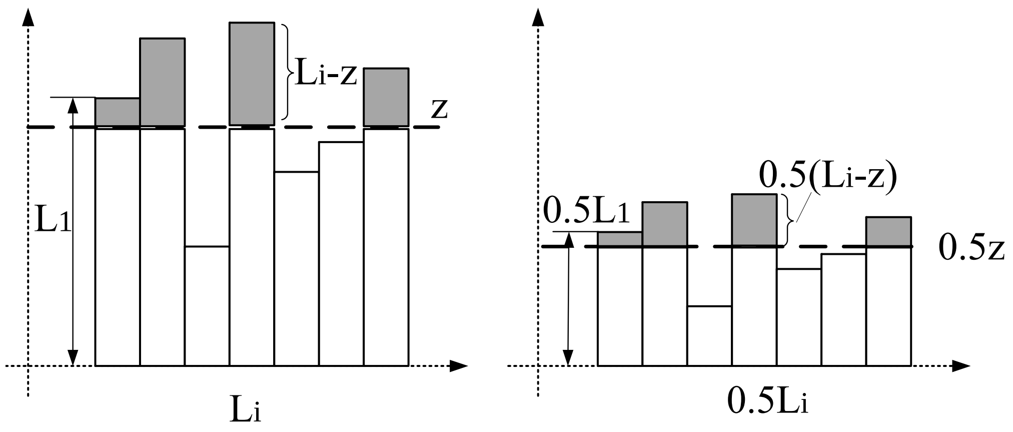


Figure 3.2: Scaling property of margin propagation

Property 2 (Monotonicity): Given a set of scores $\mathcal{L} = \{L_i\}, i = 1, .., N$ and if $\gamma_1 \geq \gamma_2 \geq 0$ then

$$M(\mathcal{L}, \gamma_1) \leq M(\mathcal{L}, \gamma_2)$$

One of the important implications of the monotonicity is the asymptotic property when $\gamma \rightarrow 0$ and is given by

$$\lim_{\gamma \rightarrow 0} M(\mathcal{L}, \gamma) = \max(\mathcal{L}). \tag{3.4}$$

The importance of this asymptotic property is that the min-sum algorithm for LDPC decoding is a special case of the proposed margin propagation algorithm.

Proof. Given

$$\begin{cases} \sum_{i=1}^{N}[L_i - z_1]_+ = \gamma_1, \\ \sum_{i=1}^{N}[L_i - z_2]_+ = \gamma_2, \\ \gamma_1 \geq \gamma_2 \geq 0, \end{cases} \tag{3.5}$$

what we need to prove is

$$z_1 \leq z_2.$$

If we subtract the first equation by the second equation in (3.5), we can get

$$\sum_{i=1}^{N}([L_i - z_1]_+ - [L_i - z_2]_+) = \gamma_1 - \gamma_2 \geq 0. \tag{3.6}$$

Based on Lemma 1,

$$\sum_{i=1}^{N}[z_2 - z_1]_+ \geq \sum_{i=1}^{N}([L_i - z_1]_+ - [L_i - z_2]_+).$$

Combined with (3.6), it can be deduced that

$$\sum_{i=1}^{N}[z_2 - z_1]_+ \geq 0.$$

Consequently,

$$z_2 - z_1 \geq 0.$$

In conclusion,

$$z_1 \le z_2.$$

□

Property 3 (Convexity Property): Given a set of coefficients $\{\alpha_k\}$ satisfying $0 \le \alpha_k \le 1$ and $\sum_k \alpha_k = 1$ and given a set of hyper-parameters $\{\gamma_k\}$

$$M(\mathcal{L}, \sum_k \alpha_k \gamma_k) \ge \sum_k \alpha_k M(\mathcal{L}, \gamma_k).$$

Proof. Given the same group of L_is, and a set of $\gamma_k, k = 1, .., n$

$$\begin{cases} \sum_{i=1}^{N} [L_i - z_1]_+ = \gamma_1, \\ \sum_{i=1}^{N} [L_i - z_2]_+ = \gamma_2, \\ \vdots \\ \sum_{i=1}^{N} [L_i - z_n]_+ = \gamma_n, \end{cases} \tag{3.7}$$

Based on Property 1, we can transform the equations in (3.7) as

$$\begin{cases} \sum_{i=1}^{N} [\alpha_1 L_i - \alpha_1 z_1]_+ = \alpha_1 \gamma_1, \\ \sum_{i=1}^{N} [\alpha_2 L_i - \alpha_2 z_2]_+ = \alpha_2 \gamma_2, \\ \qquad \vdots \\ \sum_{i=1}^{N} [\alpha_n L_i - \alpha_n z_n]_+ = \alpha_n \gamma_n, \end{cases}$$

Add the above equations up, based on Lemma 2, we will get

$$\sum_{i=1}^{N}\left[L_i - \sum_{k=1}^{n}\alpha_k z_k\right]_+ \geq \sum_{k=1}^{n}\alpha_k\gamma_k.$$

Denote z' satisfies

$$\sum_{i=1}^{N}\left[L_i - z'\right]_+ = \sum_{k=1}^{n}\alpha_k\gamma_k.$$

Based on Property 1, then $z' \geq \sum_{k=1}^{n}\alpha_k z_k$.

That is,

$$M(\mathcal{L}, \sum_k \alpha_k\gamma_k) \geq \sum_k \alpha_k M(\mathcal{L}, \gamma_k).$$

□

Property 4 (Superposition Property): Given two sets of scores $\mathcal{L}$ and $\mathcal{G}$ of size N with a well defined ordering and if $\mathcal{L} + \mathcal{G}$ represent an element by element scalar addition then

$$M(\mathcal{L}+\mathcal{G}, \gamma) \leq M(\mathcal{L}, \gamma) + M(\mathcal{G}, \gamma).$$

Proof.

$$\begin{cases} \sum_{i=1}^{N}[L_i - z_1]_+ = \gamma, \\ \sum_{i=1}^{N}[g_i - z_2]_+ = \gamma, \end{cases} \tag{3.8}$$

what we need to prove is if

$$\sum_{i=1}^{N}[L_i + g_i - z_3]_+ = \gamma, \tag{3.9}$$

then

$$z_3 \leq z_1 + z_2.$$

All the L_is are sorted and rearranged in a decreasing order so that $L_1^{'} \geq L_2^{'} \geq L_3^{'} ... \geq L_N^{'}$. We use N_1 to represent the number of $L_i^{'}$s above the threshold z_1. The same operations are done to g_is. All the g_is are sorted and rearranged decreasingly as $g_1^{'} \geq g_2^{'} \geq g_3^{'} ... \geq g_N^{'}$. And the number of $g_i^{'}$s above the threshold z_2 is N_2.

Rewrite (3.8) as

$$\begin{cases} \sum_{i=1}^{N_1} L_i^{'} - N_1 z_1 = \gamma, \\ \sum_{i=1}^{N_2} g_i^{'} - N_2 z_2 = \gamma, \end{cases} \tag{3.10}$$

Assume $N_2 \geq N_1$, then from the bottom equation in (3.10), we get

$$0 \leq \sum_{i=1}^{N_1} g_i^{'} - N_1 z_2 \leq \gamma. \tag{3.11}$$

Add both sides of (3.11) to those of the upper equation in (3.10), we can get

$$\gamma \leq \sum_{i=1}^{N_1} L_i^{'} + \sum_{i=1}^{N_1} g_i^{'} - N_1 z_1 - N_1 z_2 \leq 2\gamma. \tag{3.12}$$

(3.12) can be transformed as

$$\sum_{i=1}^{N_1} [L_i^{'} + g_i^{'} - (z_1 + z_2)] \geq \gamma,$$

from which we can deduce that

$$\sum_{i=1}^{N}[L_i + g_i - (z_1 + z_2)]_+ \geq \gamma, \tag{3.13}$$

Given (3.9) and (3.13), based on Property 2, it can be deduced that

$$z_3 \leq z_1 + z_2.$$

And vice versa if we assume $N_1 \geq N_2$. If written in MP function,

$$M(\mathcal{L} + \mathcal{G}, \gamma) \leq M(\mathcal{L}, \gamma) + M(\mathcal{G}, \gamma).$$

$\square$

Under special condition the above property reduces to an equality and is given by **Property 5 (Offset Invariance):** Given a set of scores $\mathcal{L}$ of size N and a scalar $g \in \mathcal{R}$ then

$$M(\mathcal{L} + g, \gamma) = M(\mathcal{L}, \gamma) + g.$$

Property 5 implies that if a constant offset to all the elements of input set leads to an equivalent offset in the output of the margin approximation function.

Proof. Based on the proof of Property 4, given

$$\sum_{i=1}^{N}[L_i - z]_+ = \gamma,$$

if $\mathcal{G}$ is a scalar set with each $g_i = g, g \in \mathcal{R}$, 3.9 can be rewritten as

$$\sum_{i=1}^{N}[L_i + g - (z + g)]_+ = \gamma,$$

Written in MP function format,

$$M(\mathcal{L} + g, \gamma) = M(\mathcal{L}, \gamma) + g.$$

□

Property 6 (Invariance Property): Given a set of scores $\mathcal{L}$ of size N, and given a set of scores $\mathcal{G}$ whose elements satisfy $g_i \leq M(\mathcal{L}, \gamma), i = 1, ..., N$, then

$$M(\mathcal{L} \bigcup \mathcal{G}, \gamma) = M(\mathcal{L}, \gamma).$$

Proof. Given

$$\sum_{i=1}^{N}[L_i - z]_+ = \gamma,$$

$\forall g_j, j \in [1, n]$, since $g_j \ll z$,

$$\sum_{i=1}^{n}[g_j - z]_+ = 0$$

which means

$$\sum_{i=1}^{N}[L_i - z]_+ + \sum_{i=1}^{n}[g_j - z]_+ = \gamma.$$

Therefore, we can get

$$M(\mathcal{L} \bigcup \mathcal{G}, \gamma) = M(\mathcal{L}, \gamma).$$

□

3.3 PWL approximation based on MP algorithm

The generalized form of a “log-sum-exp” function is described as

$$z_{log} = \log\left(\sum_{i=1}^{N} e^{L_i}\right). \tag{3.14}$$

The Piecewise-linear (PWL) approximation of “log-sum-exp” function can be achieved with MP algorithm.

Proof. Assume that (3.14) and (3.2) share the same group of operands $L_1, \ldots, L_N$, $L_i \in R, i = 1, ..., N$. If the operands are sorted and rearranged in a decreasing order as $L_1^{'} \geq L_2^{'} ... \geq L_N^{'}$, (3.14) can be rewritten as

$$z_{log} = \log\left(\sum_{i=1}^{N} e^{L_i^{'}}\right), \tag{3.15}$$

For MP algorithm, (3.2) is rewritten as

$$z = M(L_1^{'}, \ldots, L_N^{'}, \gamma). \tag{3.16}$$

If the number of $L_i^{'}$s above threshold z is n, $1 \leq n \leq N$, according to the definition of MP algorithm, (3.1) is rewritten as

$$\begin{aligned}\sum_{i=1}^{n}\left(L_i^{'}-z\right)&=\gamma,\\ \sum_{i=1}^{n}L_i^{'}-nz&=\gamma.\end{aligned}\tag{3.17}$$

And z can be deduced as

$$z=\frac{\sum_{i=1}^{n}L_i^{'}}{n}-\frac{\gamma}{n}.\tag{3.18}$$

For (3.18), under the conditions (a) γ and all the other $L_i^{'}$s are fixed except $L_n^{'}$ (b) $L_n^{'}$ is varying within the range not below the threshold z, we have

$$\frac{\partial z}{\partial L_n^{'}}=\frac{1}{n}.\tag{3.19}$$

For z_{log} in (3.15), under the same conditions (a) and (b),

$$\frac{\partial z_{log}}{\partial L_n^{'}}=\frac{e^{L_n^{'}}}{\sum_{i=1}^{N}e^{L_i^{'}}}=\frac{1}{\sum_{i=1}^{N}e^{(L_i^{'}-L_n^{'})}}.\tag{3.20}$$

Define a function $u(L_i^{'},L_n^{'})$ to approximate $e^{(L_i^{'}-L_n^{'})}$ as

$$u(L_i^{'},L_n^{'})=\begin{cases}1, & L_i^{'}\geq L_n^{'},\\ 0, & L_i^{'}<L_n^{'},\end{cases}\quad 1\leq n\leq N.$$

Fig. 3.3 shows us the curve of $u(L_i^{'}, L_n^{'})$ and its PWL approximating effect to that of $e^{(L_i^{'} - L_n^{'})}$.

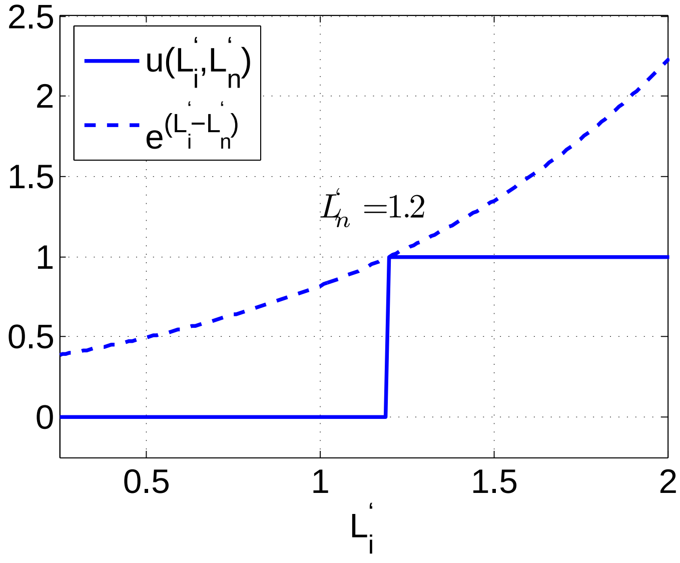


Figure 3.3: The curve of $e^{(L_i^{'} - L_n^{'})}$ and its PWL approximation curve of $u(L_i^{'}, L_n^{'})$

Since the number of $L_i^{'}$s not less than $L_n^{'}$ is n, then

$$\sum_{i=1}^{N} u(L_i^{'}, L_n^{'}) = \sum_{i=1}^{n} u(L_i^{'}, L_n^{'}) = n. \tag{3.21}$$

Therefore,

$$\frac{\partial z_{log}}{\partial L_n^{'}} \approx \frac{1}{\sum\limits_{i=1}^{N} u(L_i^{'}, L_n^{'})} = \frac{1}{n}. \tag{3.22}$$

Comparing (3.19) and (3.22), we can find out that

$$\frac{\partial z_{log}}{\partial L_n^{'}} \approx \frac{1}{n} = \frac{\partial z}{\partial L_n^{'}}. \tag{3.23}$$

Based on the illustrations above, we can draw the conclusion that z in (3.2) approximates z_{log}

in (3.14) under the condition that γ and all the other $L_i^{'}$s are fixed except $L_n^{'}$ varying within a certain range. Since $1 \leq n \leq N$, by adjusting the value of γ to change the number of operands above threshold (reflected by n), this conclusion can be extended to the whole group of operands.

$\square$

Based on the proof, we can write

$$z^{mp} = M(\mathcal{L}, \gamma) \approx \log\left(\sum_{i=1}^{N} e^{L_i}\right). \tag{3.24}$$

To demonstrate the PWL approximation effect of MP algorithm, a "log-sum-exp" based function is utilized for reference. It is a transform function ϕ with two log-likelihood operands L_1 and L_2 [47], which is named "sum-product" in the field of factor graph decoding [48]:

$$\phi_{L_2}(L_1) = \log\left(\frac{1 + e^{L_1 + L_2}}{e^{L_1} + e^{L_2}}\right). \tag{3.25}$$

The two operands are expressed in differential form, i.e., $L_1 = L_1^+ - L_1^-$, $L_2 = L_2^+ - L_2^-$. Now (5.11) is rewritten and decomposed as

$$\begin{aligned} \phi_{L_2}(L_1) &= \log\left(\frac{e^{L_1^- + L_2^-} + e^{L_1^+ + L_2^+}}{e^{L_1^- + L_2^+} + e^{L_1^+ + L_2^-}}\right), \\ &= \log\left(e^{L_1^- + L_2^-} + e^{L_1^+ + L_2^+}\right) - \log\left(e^{L_1^- + L_2^+} + e^{L_1^+ + L_2^-}\right), \end{aligned} \tag{3.26}$$

where $\phi_{L_2}(L_1)$ can be represented as a difference of two "log-sum-exp" function. Based on equation (3.24), we can approximate (3.26) using two MP units with two operands for each:

$$\phi_{L_2}(L_1) \approx M(L_1^- + L_2^-, L_1^+ + L_2^+, \gamma) - M(L_1^+ + L_2^-, L_1^- + L_2^+, \gamma). \tag{3.27}$$

Fig. 3.4 shows us the comparison of PWL approximation effect achieved with a fixed γ and different schemes of operands in either MP unit: (a) two operands for either MP unit as described in equation (3.27); (b) four operands for either MP unit and (c) six operands for either MP unit. The operands for MP units are listed in Table 3.1.

Table 3.1: Operands of MP units in Fig 3.4

Num	Operands for MP unit z^+	Operands for MP unit z^-
2	$L_1^- + L_2^-$, $L_1^+ + L_2^+$	$L_1^- + L_2^+$, $L_1^+ + L_2^-$
4	$L_1^- + L_2^-$, $\frac{1}{2}(L_1^- + L_2^- + L_1^+ + L_2^+)$, $\frac{3}{5}(L_1^- + L_2^- + L_1^+ + L_2^+)$, $L_1^+ + L_2^+$	$L_1^+ + L_2^-$, $\frac{1}{2}(L_1^- + L_2^- + L_1^+ + L_2^+)$, $\frac{3}{5}(L_1^- + L_2^- + L_1^+ + L_2^+)$, $L_1^- + L_2^+$
6	$L_1^- + L_2^-$, $\frac{2}{5}(L_1^- + L_2^- + L_1^+ + L_2^+)$, $\frac{1}{2}(L_1^- + L_2^- + L_1^+ + L_2^+)$, $\frac{3}{5}(L_1^- + L_2^- + L_1^+ + L_2^+)$, $\frac{2}{3}(L_1^- + L_2^- + L_1^+ + L_2^+)$, $L_1^+ + L_2^+$,	$L_1^+ + L_2^-$, $\frac{2}{5}(L_1^- + L_2^- + L_1^+ + L_2^+)$, $\frac{1}{2}(L_1^- + L_2^- + L_1^+ + L_2^+)$, $\frac{3}{5}(L_1^- + L_2^- + L_1^+ + L_2^+)$, $\frac{2}{3}(L_1^- + L_2^- + L_1^+ + L_2^+)$, $L_1^- + L_2^+$

As can be seen from Fig. 3.4, when more intercepted operands are introduced into MP computation, the approximating curves are becoming more fine and similar to the original "sum-product" algorithm. That means, the PWL approximation effect of (3.27) can be improved when more operands are utilized in the approximating algorithm.

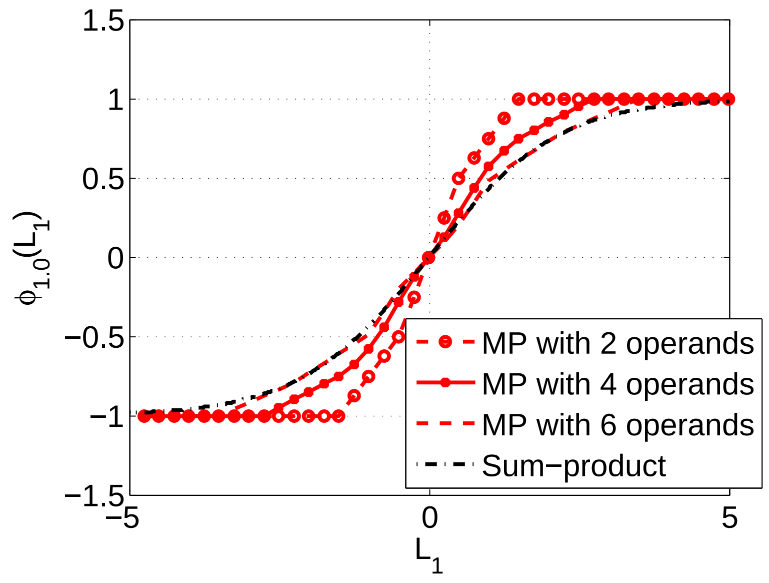


Figure 3.4: PWL approximation effect of MP algorithm

Chapter 4

MP-based current-mode PWL circuit and synthesis

4.1 Log-domain computation

As high-speed computation in signal processing system, graphic systems, communication system, are receiving increasingly concerns, logarithmic computation, as a specific and effective option in signal processing area, becomes the central of research.

Logarithmic number system (LNS) has been adopted in many trials of digital signal processor (DSP) [49, 50, 51, 52, 53]. A list of operations in base-2 LNS in comparison to normal arithmetic system is shown in table 4.1. As can be seen, log-domain computation can dramatically simplify various complex ordinary arithmetic computations, such as multiplication, division, reciprocal, square-root, and so on. Consequently, with LNS, the number of bits needed in the computation decreases and therefore the power consumption drops [52, 53]. At the same time, lower complexity results in high-speed computation as well [49, 52, 53]. Generally speaking, for the scenarios in demand of low dissipated power, high throughput and small gate counts, LNS is more appropriate than conventional arithmetic systems. However, a major obstacle of LNS implemented in digital circuits is the complicated addition and subtraction. Moreover, to simplify the hardware implementation, PWL schemes are frequently adopted by the digital logarithmic converters in real number to logarithmic domain data conversion. As a result of PWL, small coefficient lookup tables are resorted to as the power- and area-efficient implementation [52, 53, 54]. However, the use of read only memory (ROM) is costly and seriously restricts the word length of the LNS data system [49, 55].

Table 4.1: Operations in ordinary and logarithmic arithmetic in digital implemented LNS processor

Operation	Ordinary Arithmetic	Logarithmic Arithmetic
Multiplication	$x \times y$,	$X + Y$
Division	$x \div y$,	$X - Y$
Square Root	$\sqrt{x}$,	$X >> 1$
Square	x^2,	$X << 1$
Addition	$x + y$,	$X + log_2(1 + 2^{Y-X})$
Subtraction	$x - y$,	$X + log_2(1 - 2^{Y-X})$

The proposed MP-based algorithm is capable of approximate a specific log-domain computation (“log-sum-exp” function) and can be implemented in an analog manner, which circumvents the problems aforementioned for digital implementation. In later section of this chapter, a diversity of log-domain computation circuits will be shown synthesized with the a basic MP circuit.

4.2 Architecture of synthesized system

Let us take an arbitrary function as example:

$$Y = a_1 * b_1 + a_2 * b_2. \tag{4.1}$$

The operands can be converted into differential forms as

$$\begin{aligned} a_1 &= a_1^+ - a_1^-, & a_2 &= a_2^+ - a_2^-, \\ b_1 &= b_1^+ - b_1^-, & b_2 &= b_2^+ - b_2^-, \end{aligned}$$

and the corresponding operands in LNS are

$$\begin{aligned} L_{a_1}^+ &= \log a_1^+, & L_{a_1}^- &= \log a_1^-, \\ L_{a_2}^+ &= \log a_2^+, & L_{a_2}^- &= \log a_2^-, \\ L_{b_1}^+ &= \log b_1^+, & L_{b_1}^- &= \log b_1^-, \\ L_{b_2}^+ &= \log b_2^+, & L_{b_2}^- &= \log b_2^-. \end{aligned}$$

Accordingly, (4.1) can be converted into

$$Y = e^{z^+} - e^{z^-}, \tag{4.2}$$

where z^+ and z^- are expressed as

$$z^+ = \log\left(e^{L_{a_1}^+ + L_{b_1}^+} + e^{L_{a_1}^- + L_{b_1}^-} + e^{L_{a_2}^+ + L_{b_2}^+} + e^{L_{a_2}^- + L_{b_2}^-}\right)$$
$$z^- = \log\left(e^{L_{a_1}^+ + L_{b_1}^-} + e^{L_{a_1}^- + L_{b_1}^+} + e^{L_{a_2}^+ + L_{b_2}^-} + e^{L_{a_2}^+ + L_{b_2}^-}\right) \quad (4.3)$$

From (4.3), it can be seen that the original computation in (4.1) can be transformed into a "log-sum-exp" function shown in (3.14) when the computation is converted into LNS. Then it can be approximated with MP algorithm, as equation (3.24) shows.

Based on the illustration above, for any arbitrary function that can be be converted into the form of (4.2) and (4.3), i.e. expressed in terms of "log-sum-exp" function, its PWL approximation can be obtained with MP algorithm. Let us generalize such an arbitrary function with all the operands in real domain as $Y = f(a, ..., b)$. It can be converted into differential form and LNS as

$$Y = e^{z^+} - e^{z^-}.$$
$$z^+ = F^+(L_a^+, L_a^-, ..., L_b^+, L_b^-).$$
$$z^- = F^-(L_a^+, L_a^-, ..., L_b^+, L_b^-). \quad (4.4)$$

In (4.4), F^+ and F^- denote the "log-sum-exp" functions of L_a^+, L_a^-, L_b^+, L_b^-, etc.

Fig. 4.1 shows the system level architecture of arbitrary computations. The input signals are represented by their differential forms $a = a^+ - a^-$, $b = b^+ - b^-$ and are converted into logarithms at input stage. Similarly, the logarithms of the output signals are represented by z^+ and z^-. With all the logarithm/antilogarithm conversion realized by outer units at input/output stage, the core computations (within the dashed block) are carried out in log-likelihood domain and im-

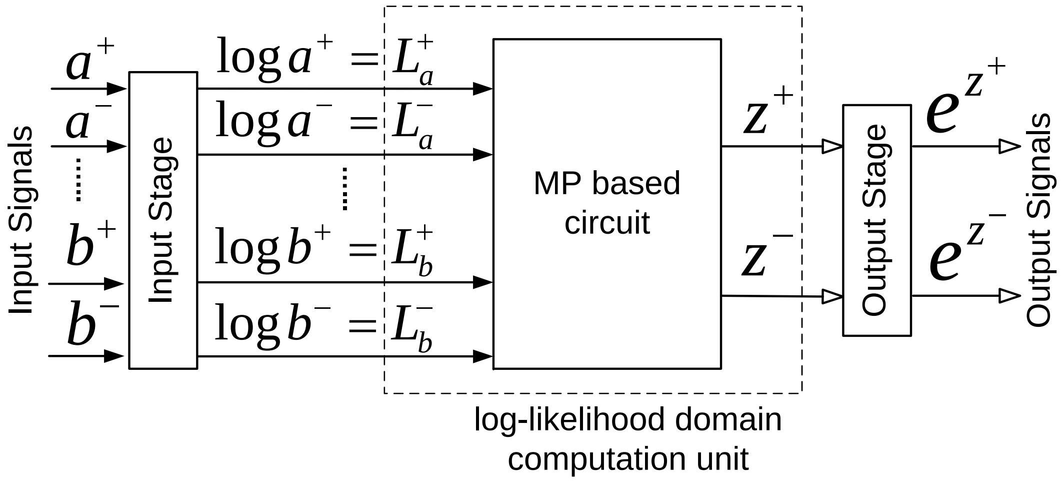


Figure 4.1: Top architecture of the computation unit

plemented by MP based PWL approximation circuit. Here we only focus on the core computation unit.

4.3 Current-mode circuit implementation for MP algorithm

A current-mode circuit implementing the basic reverse water-filling equation (3.1) is shown in Fig. 4.2. The four operands $L_1 \sim L_4$ in (3.2) are represented by currents $I_{L_1} \sim I_{L_4}$. The hyper-parameter γ in (3.2) is implemented by the current I_γ. The circuit in Fig. 4.2 bears similarity to other parallel current-mode circuits like the winner-take-all circuits. However, as mentioned in the previous section, the MP implementation is more general and for $I_\gamma \rightarrow 0$ the MP circuit implements a winner-take-all function. The Kirchoff's current law (KCL) when applied to node A, is equivalent to the reverse water-filling condition $\sum_{i=1}^{4}[I_{L_i} - I_z]_+ = I_\gamma$. I_z represents the MP approximation $M(L_1, .., L_4, \gamma)$, and $[.]_+$ denotes a rectify operation which is implemented by the PMOS diodes $P_1 \sim P_4$. It can be readily verified that the PMOS diodes ensure that $V_{DS,N_i} = V_{SG,P_i} + V_{DS,sat}$, where V_{DS,N_i} is the drain-to-source voltage drop across transistor $N_i, i = 1,..,4$, V_{SG,P_i} is the voltage drop across the PMOS diodes and $V_{DS,sat}$ is the saturation voltage across the current sink I_γ. Thus, transistors $N_1 \sim N_4$ are always maintained in saturation irrespective of the operation region (weak, moderate and strong inversion). This is important because the operation of the MP circuit relies on the accuracy of the current mirrors formed by $N_0 \sim N_4$. Also, to ensure proper operation of the core MP circuit the input currents have to satisfy the constraint $\sum_{i=1}^{4} I_{L_i} \geq I_\gamma$ and the input current range can be expressed as

$$\frac{1}{2}\mu n Cox(\frac{W}{L})(\frac{1}{2}V_{dd})^2 \geq I_{L_i} \geq 0. \tag{4.5}$$

In the derivation of the upper-bound of the input current range we have assumed that the $I_{L_i} >> I_\gamma$ and the PMOS and the NMOS transistors have similar drain-to-source voltage drops (implying $V_B \approx V_{dd}/2$). It should be noted that the PMOS diodes in circuit shown in Fig. 4.2 introduces a threshold voltage drop which implies that the supply voltage has to be greater than $2V_{GS} + V_{DS}$. However, a low-voltage implementation of the circuit in Fig. 4.2 is possible by replacing the PMOS diodes by low-threshold active diodes [56], but at the expense of larger silicon area, higher power dissipation and degraded speed and stability.

We have prototyped a programmable version of the MP circuit shown in Fig. 4.2 in a 0.5μm standard CMOS process. The architecture of the prototype is shown in Fig. 4.4 and its micrograph is shown in Fig. 4.3. A serial shift-register is used for selectively turning on and off each of different branches of currents. In this manner, the same MP circuit in Fig. 4.2 can be reconfigured to implement different mathematical functions. The architecture in Fig.4.4 uses an on-chip first-order $\Sigma\Delta$ modulator as an analog-to-digital converter for measurement and calibration of the output currents. Because the circuits used for implementing the $\Sigma\Delta$ modulator have been reported

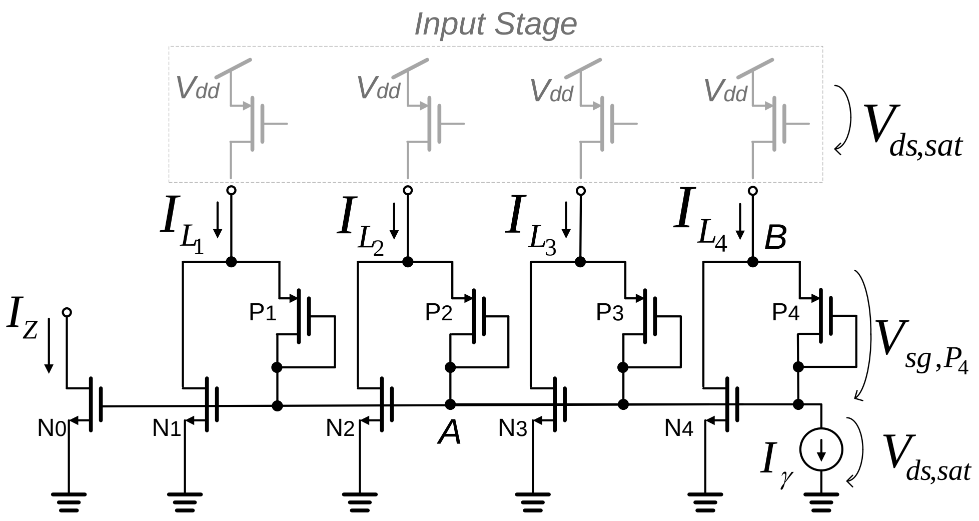


Figure 4.2: CMOS implementation of the core MP circuit.

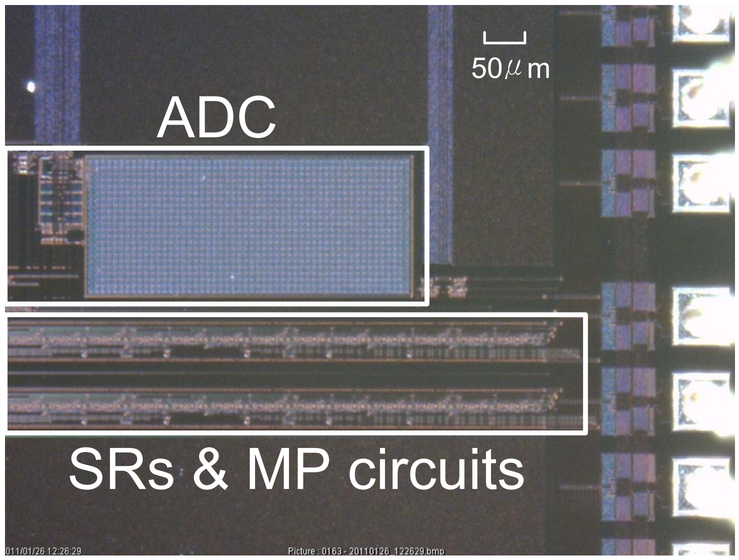


Figure 4.3: Die micrograph of the chip.

elsewhere [57] we have omitted its implementation details in this paper. The input currents and hence the operating region (weak, moderate and strong inversion) of all the transistors are adjusted by controlling the gate voltages of the PMOS transistors. For each of the operating regions, the ADC is re-calibrated to the maximum current and the measured results (presented in the next sections) are normalized with respect to the maximum current.

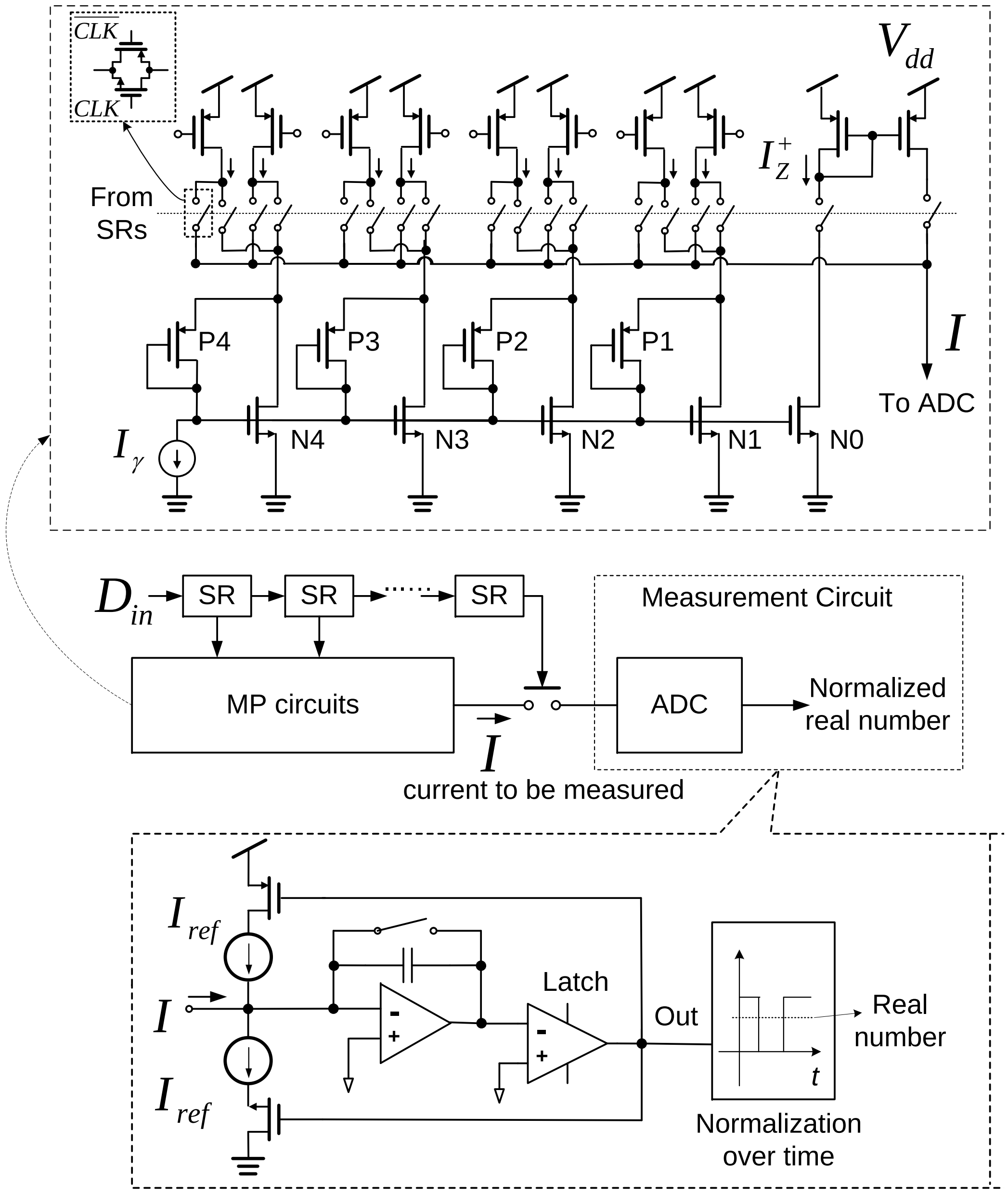


Figure 4.4: System architecture of the chip.

4.4 MP based computational circuits

In previous section, we already illustrate MP is an effective PWL approximation algorithm for a specific log-domain function. The circuit schematic of a four-operand MP based PWL approximation circuit is shown in Fig. 4.2. In this section, we illustrate some log-domain fundamental computations that can be synthesized based on the "log-sum-exp" function and their MP based PWL approximation circuits. Based on these fundamental computation circuits, arbitrary log-domain arithmetic operations and their combinations can be implemented, which can be validated with some classical mathematic functions presented in this section as well. Current-mode signal representation is exploited so that power dissipation can be controlled adaptively under different power consumption requirements.

4.4.1 Addition

To compute $a+b$ with respect to the two input variables a and b, $a \in \mathbb{R}, b \in \mathbb{R}$ the operands are first represented in their differential forms $a = a^+ - a^-$ and $b = b^+ - b^-$ with $a^+, a^-, b^+, b^- \in \mathbb{R}^+$. Then, the operation is rewritten as in Fig. 4.1

$$\begin{aligned} a + b = a^+ - a^- + b^+ - b^- &= \left(a^+ + b^+\right) - \left(a^- + b^-\right) \\ &= e^{\log\left(a^+ + b^+\right)} - e^{\log\left(a^- + b^-\right)}. \end{aligned} \tag{4.6}$$

All the operands are then mapped into a log-likelihood domain according to

$$\begin{aligned} L_a^+ &= \log a^+, \quad & L_a^- &= \log a^-, \\ L_b^+ &= \log b^+, \quad & L_b^- &= \log b^-, \end{aligned}$$

where $L_a^+ \in \mathbb{R}$, $L_a^- \in \mathbb{R}$, $L_b^+ \in \mathbb{R}$, $L_b^- \in \mathbb{R}$. Since, our circuit implementation uses only unipolar current sources, we impose additional constraints $a^+, a^-, b^+, b^- > 1$. Due to the differential representation, this additional constraint will not affect the final output. Let z^+ denote $\log\left(a^+ + b^+\right)$ and z^- denote $\log\left(a^- + b^-\right)$. Then, (4.6) can be written as

$$a + b = e^{z^+} - e^{z^-}. \tag{4.7}$$

and z^+ can be expanded as

$$\begin{aligned} z^+ &= \log\left(a^+ + b^+\right) = \log\left(e^{\log a^+} + e^{\log b^+}\right) \\ &= \log\left(e^{L_a^+} + e^{L_b^+}\right). \end{aligned} \tag{4.8}$$

We now apply the MP approximation to the log-sum-exp functions to obtain

$$z^+ \approx M(L_a^+, L_b^+, \gamma). \tag{4.9}$$

Similarly,

$$\begin{aligned} z^- &= \log\left(a^- + b^-\right) = \log\left(e^{\log a^-} + e^{\log b^-}\right) = \log\left(e^{L_a^-} + e^{L_b^-}\right) \\ &\approx M(L_a^-, L_b^-, \gamma) \end{aligned} \tag{4.10}$$

Since the equation (4.9) and the equation (4.10) uses only two operands, its circuit level implementation is identical to Fig. 4.2 except that it has only two input branches instead of four. The single-ended circuit for addition is shown in Fig. 4.5. I_{z^+}, the current through N_0, represents the value of z^+. $I_{L_a^+}$, $I_{L_b^+}$ and I_γ represents L_a^+, L_b^+ and γ respectively.

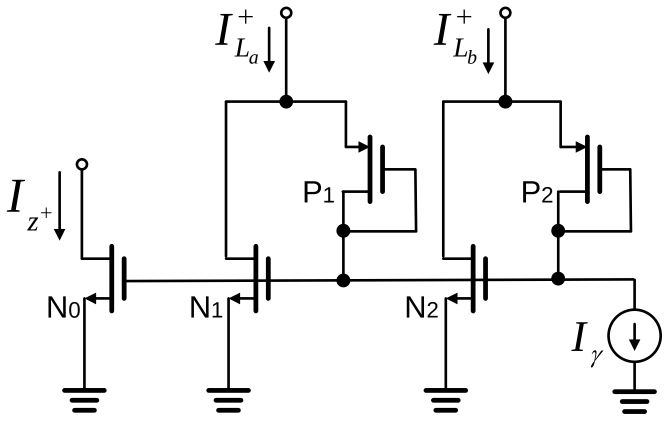


Figure 4.5: MP circuit for implementing two operand addition.

Fig. 4.6a (a)-(c) compares the output $z^+ - z^-$ obtained using floating-point implementation of the “log-sum-exp” functions, the MP approximation and from measurements (normalized currents) using the fabricated prototype.

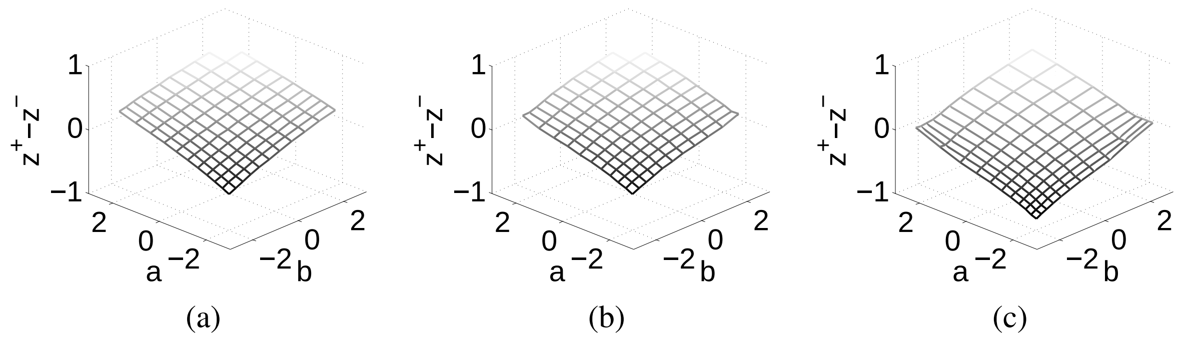


Figure 4.6: Addition: $z^+ - z^-$ computed according to (a) the original log-sum-exp function; (b) MP approximations (simulation); and (c) MP circuits (measurement).

Fig. 4.7 (a)-(c) show the measurement results when the transistors are biased in (a) strong inversion; (b) moderate inversion; and (c) weak inversion. Fig. 4.7 (d)-(f) show the approximation error between the measured response and the software model. The results validate our claim for the addition operation that MP approximation is scalable across different biasing regions and the circuit in Fig. 4.5 approximates the "log-sum-exp" function with a reasonable error. The error increases when the circuit is biased in the weak inversion and as with any sub-threshold analog VLSI circuit, which we attribute to transistor mismatch, flicker-noise, and errors introduced in mapping floating-point operands into analog operands (currents).

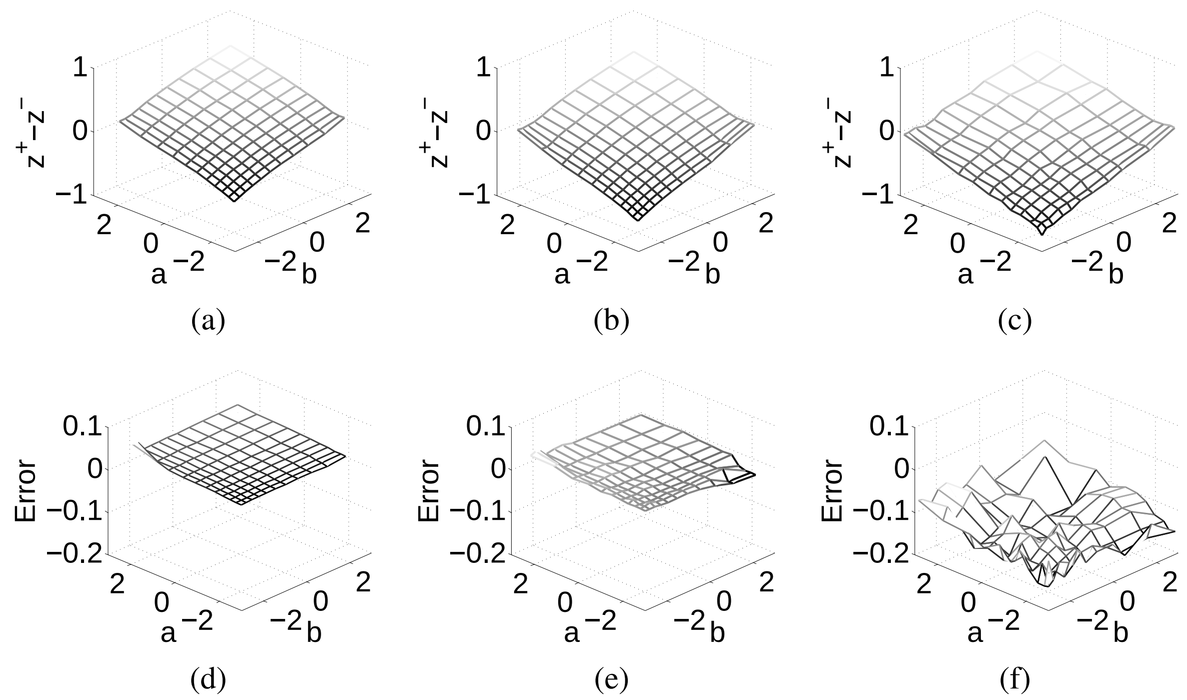


Figure 4.7: Addition: measurement results for $z^+ - z^-$ with transistors biased in (a)strong inversion; (b)moderate inversion; (c)weak inversion; (d)-(f) computation errors for (a)-(c).

4.4.2 Subtraction

Synthesizing an MP circuit for subtraction $a - b$, with a and b being the two operands is similar to the addition operation. The operation can be written in a differential form as $a = a^+ - a^-$ and $b = b^+ - b^-$, with $a^+, a^-, b^+, b^- \in \mathbb{R}^+$ and satisfying $a^+, a^-, b^+, b^- > 1$. Thus, $a - b$ is written as

$$
\begin{aligned}
a - b &= \left(a^{+} - a^{-}\right) - \left(b^{+} - b^{-}\right) = \left(a^{+} + b^{-}\right) - \left(a^{-} + b^{+}\right) \\
&= e^{\log\left(a^{+}+b^{-}\right)} - e^{\log\left(a^{-}+b^{+}\right)} \qquad (4.11)
\end{aligned}
$$

which after conversion into a log-likelihood domain

$$
\begin{aligned}
L_a^{+} &= \log a^{+}, \quad L_a^{-} = \log a^{-}, \\
L_b^{+} &= \log b^{+}, \quad L_b^{-} = \log b^{-},
\end{aligned}
$$

leads to

$$
\begin{aligned}
z^{+} &= \log\left(a^{+} + b^{-}\right) = \log\left(e^{\log a^{+}} + e^{\log b^{-}}\right) = \log\left(e^{L_a^{+}} + e^{L_b^{-}}\right) \\
&\approx M(L_a^{+}, L_b^{-}, \gamma) \qquad (4.12)
\end{aligned}
$$

and

$$
\begin{aligned}
z^{-} &= \log\left(a^{-} + b^{+}\right) = \log\left(e^{\log a^{-}} + e^{\log b^{+}}\right) = \log\left(e^{L_a^{-}} + e^{L_b^{+}}\right) \\
&\approx M(L_a^{-}, L_b^{+}, \gamma) \qquad (4.13)
\end{aligned}
$$

where z^{+} and z^{-} are differential forms of the output log-likelihoods as described in (4.2). The circuit implementation of subtraction is identical to that of addition, except that the input differential currents are now permuted. Fig. 4.8 compares the measured response with software based "log-sum-exp" and MP implementation. Also, Fig. 4.9 (a)-(c) shows the measured results when the transistors are biased in the three different operating regimes. Similar to the results obtained for the addition operation, the measured result show the bias-scalable operation of the MP circuits.

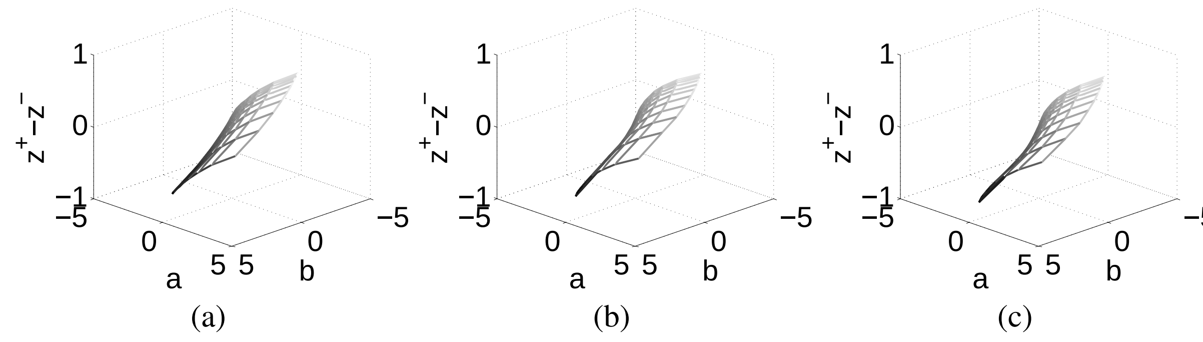


Figure 4.8: Subtraction: $z^+ - z^-$ computed by (a) original functions; (b) MP approximations (simulation); and (c) MP circuits (measurement).

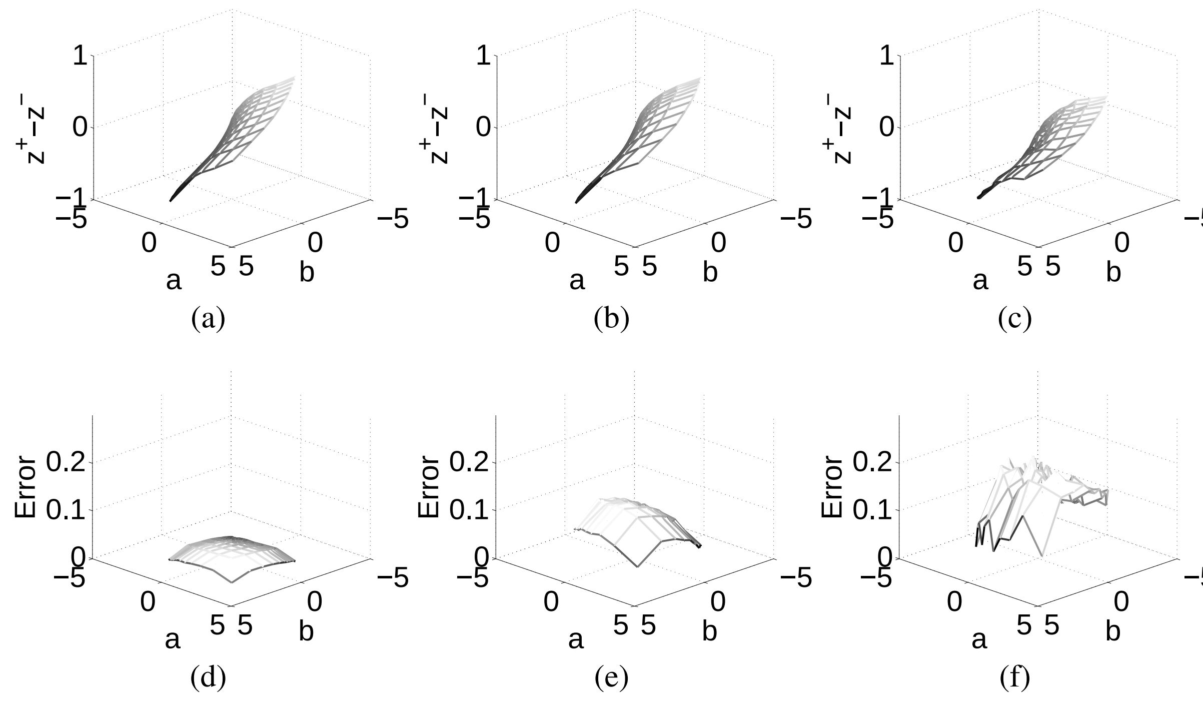


Figure 4.9: Subtraction: measurement results for $z^+ - z^-$ with transistors biased in (a)strong inversion; (b)moderate inversion; (c)weak inversion; and (d)-(f) show respective approximation errors.

4.4.3 Four quadrant multiplication

Implementing a single quadrant multiplier using MP circuit is straight-forward because $z = a \cdot b$ in its logarithmic form can be expressed as a sum of likelihoods $L_z = L_a + L_b$, where L_x denotes the logarithm of the operand x. Thus, a single quadrant multiplier requires only a current summation and can be implemented using KCL. For an MP-based four-quadrant multiplier the operation $a \cdot b$ is again expressed in terms of the differential operands as:

$$\begin{aligned} a \cdot b &= \left(a^+ - a^-\right) \cdot \left(b^+ - b^-\right) = \left(a^+ \cdot b^+ + a^- \cdot b^-\right) - \left(a^- \cdot b^+ + a^+ \cdot b^-\right) \\ &= e^{\log\left(a^+ \cdot b^+ + a^- \cdot b^-\right)} - e^{\log\left(a^- \cdot b^+ + a^+ \cdot b^-\right)}. \end{aligned} \tag{4.14}$$

After log-likelihood mapping, the following differential forms are obtained

$$\begin{aligned} z^+ &= \log\left(a^+ \cdot b^+ + a^- \cdot b^-\right) = \log\left(e^{\log\left(a^+ \cdot b^+\right)} + e^{\log\left(a^- \cdot b^-\right)}\right) \\ &\approx M\left(L_a^+ + L_b^+, L_a^- + L_b^-, \gamma\right) \end{aligned} \tag{4.15}$$

and

$$\begin{aligned} z^- &= \log\left(a^- \cdot b^+ + a^+ \cdot b^-\right) = \log\left(e^{\log\left(a^- \cdot b^+\right)} + e^{\log\left(a^+ \cdot b^-\right)}\right) \\ &\approx M\left(L_a^- + L_b^+, L_a^+ + L_b^-, \gamma\right) \end{aligned} \tag{4.16}$$

The circuit implementing an MP based four-quadrant multiplier is shown in Fig. 4.10. I_{z^+} denotes the value of z^+. I_γ represents γ. $I_{L_a^+}(I_{L_a^-})$, $I_{L_b^+}(I_{L_b^+})$ represent $L_a^+(L_a^-)$, $L_b^+(L_b^-)$ respectively.

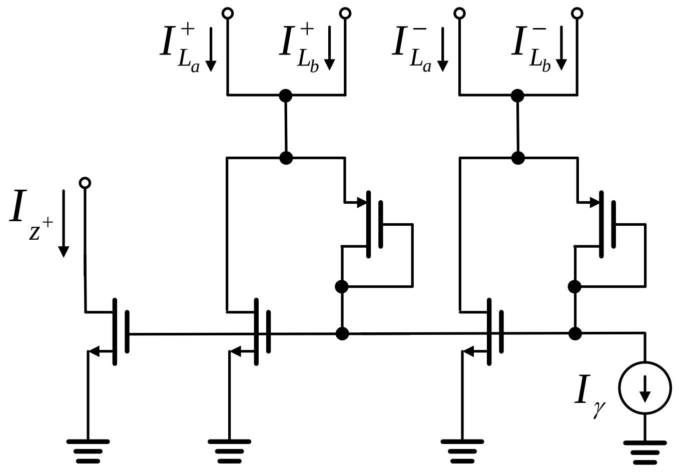


Figure 4.10: MP circuit for implementing four quadrant operand multiplication

Fig. 4.11 compares the measured output $z^{+} - z^{-}$ with the ideal implementation of the model (4.15) and (4.16). Fig. 4.12 (a)-(c) shows the measurement results when the transistors are biased in: (a) strong; (b) moderate and (c) weak inversion along with the respective approximation error. Again, the measured results show the bias-scaling property of MP circuits.

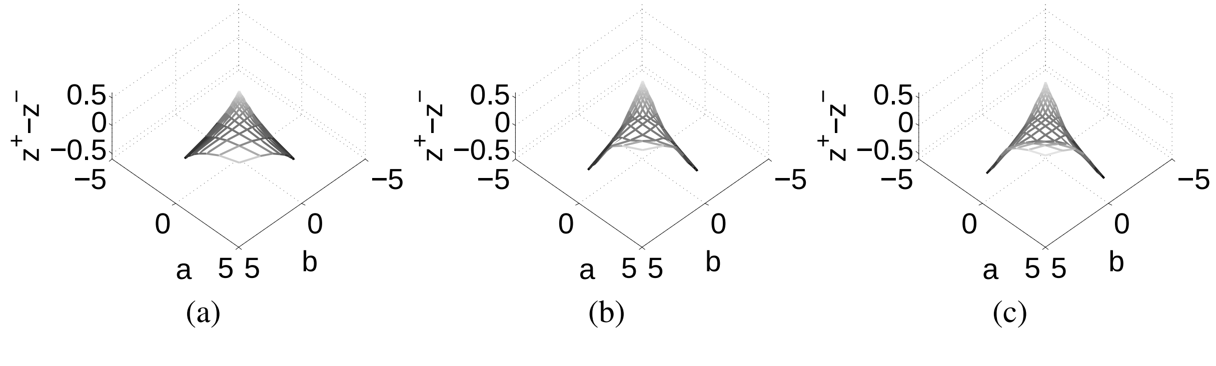
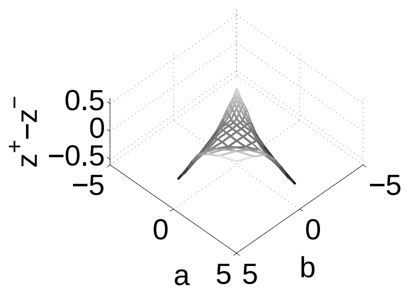
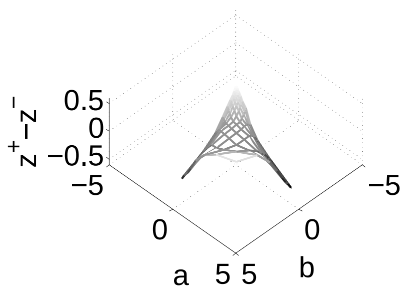


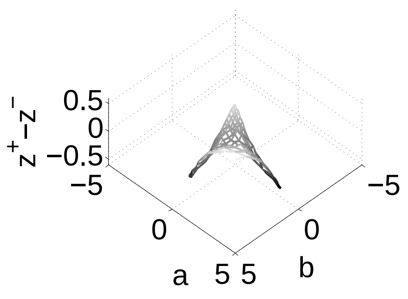
Figure 4.11: Multiplication: $z^{+} - z^{-}$ computed according to the (a) original function; (b) MP approximations (simulation); and (c) MP circuits (measurement).



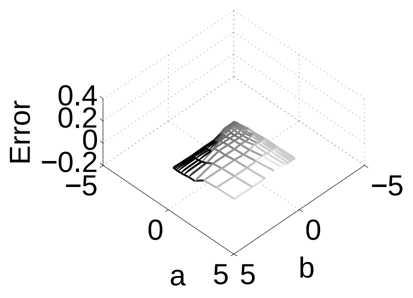
(a)



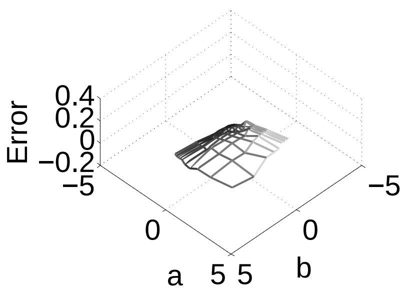
(b)



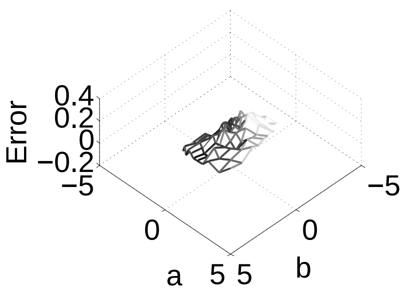
(c)



(d)



(e)



(f)

Figure 4.12: Multiplication: measurement results for $z^{+} - z^{-}$ with transistors biased in (a)strong inversion; (b)moderate inversion; (c)weak inversion; and (d)-(f) the respective approximation errors .

4.4.4 Single Quadrant Division

Like the implementation of a single quadrant multiplication, implementing a single quadrant division in logarithmic domain is straight-forward. For operands $a, b > 0$, division in the logarithmic domain is equivalent to a subtraction operation $L_a - L_b$ which can be implemented using KCL. To implement four quadrant division, the operation first requires extracting the sign of the operand as $x = sgn(x) \cdot |x|$, where $|x|$ is the absolute value of x. A single quadrant division can now be used for the absolute values and the sign of the final output can be computed using only comparators.

4.4.5 Power and Polynomial Computation

For positive operands satisfying $a \in \mathbb{R}^+$, $a > 1$, computing a^n, $n \in \mathbb{Z}^+$ in the logarithmic domain is equivalent to

$$\log\left(a^n\right) = n \cdot \log\left(a\right) = n \cdot L_a. \tag{4.17}$$

which can be implemented using a simple current mirror as shown in Fig. 4.13. Thus, in MP-based synthesis, power functions like square-root or cube-root can be implemented using current mirrors which significantly simplifies the implementation compared to translinear based synthesis [58]. However, for a four-quadrant implementation of power and polynomial function would require evaluating a Taylor expansion using multiplication and addition operations.

For a non-positive operand a, $a \in \mathbb{R}$, its power function is expressed as a^n, $n \in \mathbb{Z}^+$. a is expressed in its differential form $a = a^+ - a^-$, $a^+(a^-)\in \mathbb{R}^+$ and $a^+(a^-)> 1$. For arbitrary n,

a^n can be written as

$$
\begin{aligned}
\left(a^+ - a^-\right)^n &= \sum_i (-1)^i C_n^i \left(a^+\right)^{n-i} \left(a^-\right)^i \\
&= \sum_{i \text{ is even}} C_n^i \left(a^+\right)^{n-i} \left(a^-\right)^i - \sum_{i \text{ is odd}} C_n^i \left(a^+\right)^{n-i} \left(a^-\right)^i \\
&= e^{\log\left(\sum_{i\text{: even}} C_n^i \left(a^+\right)^{n-i} \left(a^-\right)^i\right)} - e^{\log\left(\sum_{i\text{: odd}} C_n^i \left(a^+\right)^{n-i} \left(a^-\right)^i\right)}.
\end{aligned}
\tag{4.18}
$$

Designate z^+ as $\log\left(\sum_{i\text{: even}} C_n^i \left(a^+\right)^{n-i} \left(a^-\right)^i\right)$ and z^- to represent $\log\left(\sum_{i\text{: odd}} C_n^i \left(a^+\right)^{n-i} \left(a^-\right)^i\right)$ in (4.18). Then

$$
\begin{aligned}
z^+ &\approx M(\mathcal{L}^+, \gamma). \\
z^- &\approx M(\mathcal{L}^-, \gamma).
\end{aligned}
\tag{4.19}
$$

$\mathcal{L}^+(\mathcal{L}^-)$ is a group of L_is satisfying

$$
L_i = \log C_n^i + (n-i)\log a^+ + i \log a^-,
\tag{4.20}
$$

$i \in [0, n]$ and i is even(odd).

As an example, a four quadrant square function a^2 can be implemented by expressing it as a product of two differential operands as $(a^+ - a^-) \cdot (a^+ - a^-)$ which has an architecture similar to the four-quadrant multiplier described in above section. The differential output z^+ and z^- corresponding to the square operation can be expressed in terms of differential log-likelihood

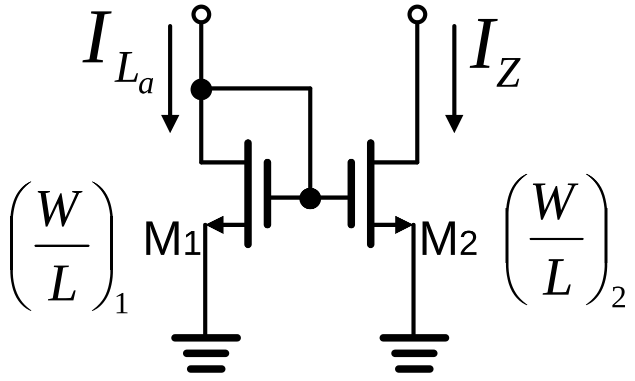


Figure 4.13: A current mirror implementing the power function in logarithmic domain.

inputs L_a^+, L_a^- as

$$z^+ \approx M\left(2L_a^+, 2L_a^-, \gamma\right) \tag{4.21}$$

$$z^- = \log 2 + L_a^+ + L_a^- \tag{4.22}$$

Fig.4.14 shows the measured results when the circuit in Fig. 4.10 has been configured to compute a four-quadrant square function. Again, the circuit is shown to approximate the "log-sum-exp" function for strong-inversion, moderate-inversion and weak-inversion biasing conditions.

4.5 Scalability analysis of MP circuit

In Fig. 3.4, we showed that the accuracy of the PWL approximation improves when the number of operands in the margin function increases. In this section, we analyze the effect of hardware artifacts (noise, mismatch, speed and input/output impedance) on the performance and scalability of MP circuits.

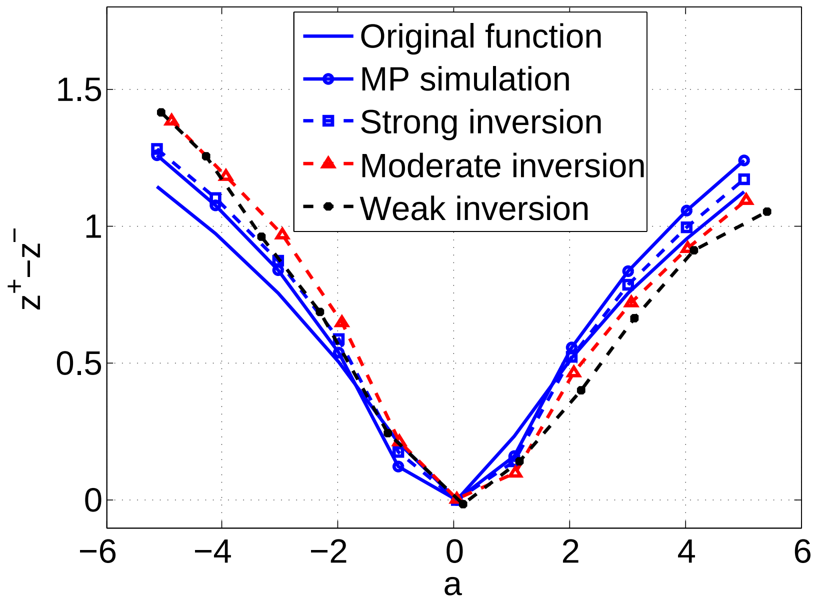


Figure 4.14: Power:$z^{+} - z^{-}$ computed according to the log-sum-exp function, the MP approximation simulation and the MP circuit measurement.

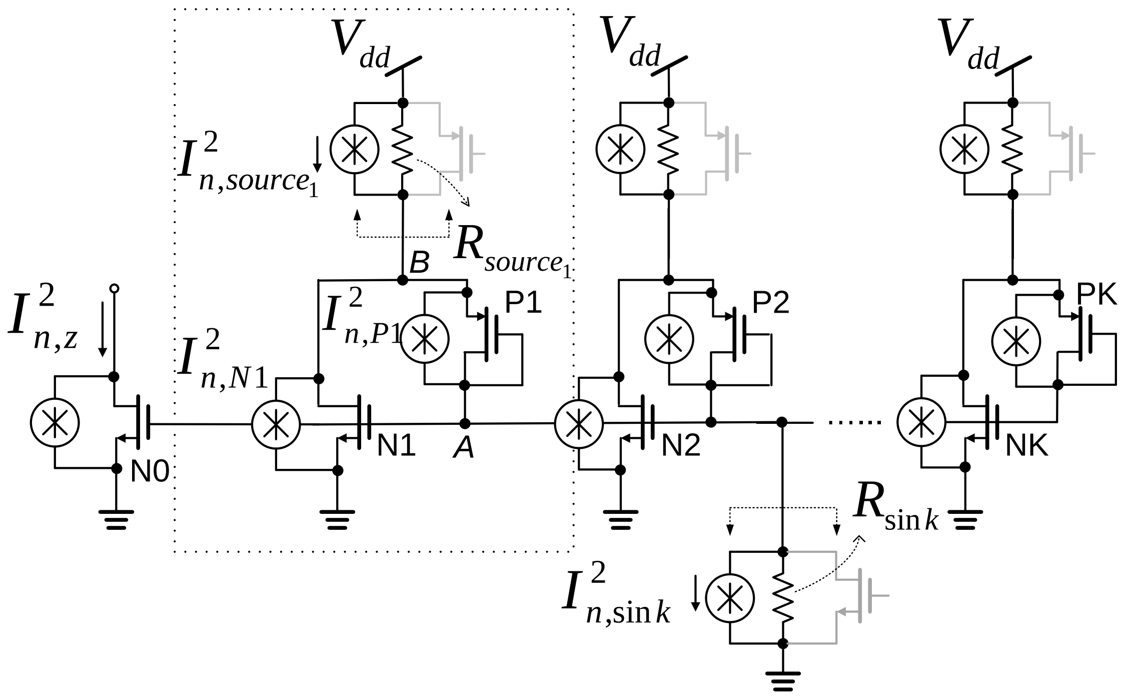


Figure 4.15: Noise model for MP circuit in Fig. 4.2.

4.5.1 Noise and Mismatch Analysis of MP circuit

The noise model for the basic MP circuit in Fig. 4.2 is shown in Fig. 4.15. For the following analysis we will assume that the number of input current branches in the MP circuit equals K. Also, assuming that the noise in $K-1$ (due to KCL) of the parallel branches are mutually independent, the total output noise power $I^2_{n,z}$ can be expressed as:

$$I^2_{n,z} = \sum_{i=1}^{K} \left[\frac{I^2_{n,source_i}}{(K-2)^2} + \frac{I^2_{n,N_i}}{(K-2)^2} + \frac{I^2_{n,P_i}}{g_m^2 R^2_{source}(K-2)^2} \right] + \frac{I^2_{n,sink}}{K^2} + I^2_{n,N_0} \quad (4.23)$$

where $I_{n,source_i}$, I_{n,N_i}, I_{n,P_i}, and $I_{n,sink}$ are noise currents as shown in Fig. 4.15 and R_{source} is the output impedance of the current sources. We have simplified the expression in (4.23) by assuming K to be large and by assuming that the small-signal parameters g_m, R_{source} for all the transistors are approximately equal. Equation (4.23) can be written as

$$I^2_{n,z,thermal} = \left[1 + \frac{2}{K} + \frac{1}{g_m^2 R^2_{source} K} \right] \frac{8}{3} kT g_m, \quad (4.24)$$

where we have considered only the effect of thermal noise as given by $I^2_n = 8/3kTg_m$ [59]. The effect of the flicker-noise has been ignored in equation (4.24), because it exhibits a similar dependency on K as the thermal-noise.

Equation (4.24) shows that as the number of branches (or operands) K increases, the output noise reduces and in the limit $K \to \infty$, the output noise is just equal to the noise due to transistor N_0. The reduction in the output noise with the increase in K can be attributed to the attenuation and averaging effect of the K stage current mirrors formed by $N_1 \sim N_K$. Also, it is important that $g_m R_{source} >> 1$, not only for reducing the effect of noise but also to ensure that the input impedance of the MP circuit is much lower than the output impedance of the current sources (see Fig. 4.15).

The mismatch analysis for the MP circuit follows a similar procedure as the noise analysis. The MOS transistor mismatch models used in the analysis are based on the work reported in [60] which uses two technology dependent constants A_{V_T} and A_β that determine the mismatch variance as:

$$\sigma^2(\Delta V_T) = \frac{A_{V_T}^2}{WL} \tag{4.25}$$

$$\left(\frac{\sigma(\Delta\beta)}{\beta}\right)^2 = \frac{A_\beta^2}{WL} \tag{4.26}$$

V_T denotes the threshold voltage, $\beta = \mu C_{ox} W/L$, W and L represent the width and length of the MOS transistor. It is also claimed in [60] that for MOS transistors the mismatch in V_T dominates the mismatch in β. Therefore, in this paper, our analysis is only be based on the V_T mismatch. The error variance $\sigma^2(\Delta V_T)$ can be converted into an equivalent noise current source in Fig. 4.15 according to

$$\sigma^2(I) \simeq g_m^2 \sigma^2(\Delta V_T) = g_m^2 \frac{A_{V_T}^2}{WL} \tag{4.27}$$

Hence, similar to the noise analysis, the error variance in the output current $\sigma^2(I_z)$ is given by:

$$\sigma^2(I_z) \simeq \left(1 + \frac{1}{K} + \frac{1}{g_m^2 R_{source}^2 K}\right) g_m^2 \frac{A_{V_T}^2}{WL}. \tag{4.28}$$

Equation (4.28) shows that for a large K the accuracy of the MP circuit is limited by the mismatch due to the output current mirror stage N_0. Thus, all layout and impedance transformation methods which are used for improving the accuracy of current mirrors, like centroid layout and cascoding techniques, can be used to improve the accuracy of MP circuits. The use of cascoding at the output stage transistor N_0 increases the output impedance and makes the core MP module scalable to implement large networks.

4.5.2 Speed and Bandwidth Analysis

As with other current-mode circuits, the transient response of the MP circuits is determined by its: (a) slew-rate which determines the large-signal response; and (b) bandwidth which determines the small-signal response. The node A in Fig. 4.2 has the largest capacitance and is discharged by the current I_γ. Therefore, the slew-rate of the MP circuit can be expressed as:

$$SR_A = \frac{I_\gamma}{K(C_{GS,N} + C_{GS,P} + C_{DB,P})} \tag{4.29}$$

where $C_{GS,N}$ represents the gate-to-source capacitance of NMOS transistor N_1 through N_K, and $C_{GS,P}/C_{DB,P}$ denotes the gate-to-source/drain-to-body capacitance of the PMOS transistor P_1 through P_K. Equation (4.29) shows that the slew-rate will reduce when the number of input branches K increase. However, the small-signal bandwidth of the circuit remains invariant with respect K and is determined by the frequency of the pole located at node A and is given by

$$f_{-3dB,A} = \frac{g_{m,P}}{2\pi(C_{GS,N} + C_{GS,P} + C_{DB,P})}. \tag{4.30}$$

$g_{m,P}$ represents the small-signal gate referred transconductance for the PMOS transistors $P_1 \sim P_K$. The slew-rate and bandwidth analysis show that scaling of MP circuit is limited by the application specific speed requirements which is controlled by the current I_γ. However, unlike translinear synthesis this limitation can be overcome in MP circuit by re-biasing the circuit in strong-inversion.

4.5.3 Performance comparison with Translinear synthesis

In this section, we summarize and compare the performance of CMOS circuits designed using MP based synthesis with circuits designed using translinear synthesis. The comparisons made here are qualitative in nature because many of the performance metrics (silicon area and power dissipation) are dependent on the topology and complexity of the translinear circuit.

- **Accuracy:** MP based synthesis relies on PWL approximations at the algorithmic level. However, the accuracy of its circuit level implementation using CMOS current-mode circuit is only limited by mismatch. Our analysis in the previous section showed that the techniques used for designing precision current-mirrors can also be used to improve the accuracy of the MP circuits. TL based synthesis, on the other hand, are precise at the algorithmic level. However, their mapping to CMOS current-mode circuits is approximate due to the effect of sub-threshold slope (unlike bipolar transistors) and finite drain impedance. These artifacts also introduce temperature dependency in TL circuits, whereas MP circuits are theoretically temperature invariant. Also, due to their operation in weak-inversion, CMOS TL circuits are more prone to errors due to mismatch, whereas the MP circuits can be re-biased in strong-inversion if higher precision is desired, but at the expense of higher power dissipation.
- **Dynamic range:** The bias-scalable property of the MP circuit enables it to achieve a larger dynamic range compared to a TL circuit. Also, MP synthesis uses currents to represent log-likelihoods which can also achieve a larger dynamic range.
- **Speed:** For an MP circuit higher speed can be achieved by re-biasing the circuit in strong-inversion, without changing the circuit topology. For CMOS translinear circuits higher speed

can be achieved by the use of large currents which implies using large size transistors to maintain their operation in the weak inversion. This will increase the silicon area when compared to an equivalent MP circuit.

4.6 Application Example: Pattern Classifier

One of the applications where ultra-low power analog computing is attractive is pattern classification [61, 62]. Most pattern classifiers do not require high precision and any analog artifacts (like mismatch and non-linearity) can be calibrated using an offline training procedure. In this example, we use the MP circuits to design a bias-scalable linear classifier. A linear classifier implements an inner-product computation between a weight vector $\mathbf{w} \in \mathbf{R}^n$ and the classifier input $\mathbf{x} \in \mathbf{R}^n$ and is given by

$$f = \mathbf{w}^T \mathbf{x} \tag{4.31}$$

The weight vector $\mathbf{w}$ is determined by an offline training procedure [61] using a labeled training dataset. The two vectors $\overrightarrow{\mathbf{w}}$ and $\overrightarrow{\mathbf{x}}$ are expressed in differential form as

$$\begin{aligned}\overrightarrow{\mathbf{w}} &= \overrightarrow{\mathbf{w}}^+ - \overrightarrow{\mathbf{w}}^- = [w_1^+, w_2^+, ..., w_N^+]^T - [w_1^-, w_2^-, ..., w_N^-]^T \\ \overrightarrow{\mathbf{x}} &= \overrightarrow{\mathbf{x}}^+ - \overrightarrow{\mathbf{x}}^- = [x_1^+, x_2^+, ..., x_N^+]^T - [x_1^-, x_2^-, ..., x_N^-]^T\end{aligned} \tag{4.32}$$

with all the $w_i^+(w_i^-)$ and $x_i^+(x_i^-) \in \mathbb{R}^+$. Then equation (4.31) can be rewritten as

$$\begin{aligned} \vec{\mathbf{w}}^T \cdot \vec{\mathbf{x}} &= \sum_{i=1}^{N} \left(w_i^+ x_i^+ + w_i^- x_i^- \right) - \sum_{i=1}^{N} \left(w_i^+ x_i^- + w_i^- x_i^+ \right) \\ &= e^{\log \sum_{i=1}^{N} \left(w_i^+ x_i^+ + w_i^- x_i^- \right)} - e^{\log \sum_{i=1}^{N} \left(w_i^+ x_i^- + w_i^- x_i^+ \right)} \\ &= e^{z^+} - e^{z^-}. \end{aligned} \tag{4.33}$$

when z^+ and z^- are expressed as

$$\begin{aligned} z^+ &= \log \sum_{i=1}^{N} \left(w_i^+ x_i^+ + w_i^- x_i^- \right) \\ &\approx M(L{w_1}^+ + L{x_1}^+, L{w_1}^- + L{x_1}^-, ..., \\ &\qquad L{w_N}^+ + L{x_N}^+, L{w_N}^- + L{x_N}^-, \gamma) \\ z^- &= \log \sum_{i=1}^{N} \left(w_i^+ x_i^- + w_i^- x_i^+ \right) \\ &\approx M(L{w_1}^+ + L{x_1}^-, L{w_1}^- + L{x_1}^+, ..., \\ &\qquad L{w_N}^+ + L{x_N}^-, L{w_N}^- + L{x_N}^+, \gamma). \end{aligned} \tag{4.34}$$

$L{w_i}^+(L{w_i}^-)$ and $L{x_i}^+(L{x_i}^-)$ denote the logarithms of $w_i^+(w_i^-)$ and $x_i^+(x_i^-)$ in equation (4.33). For a binary classifier, only the sign of $z^+ - z^-$ is important, implying that the output stage in the architecture (Fig. 4.1) could be implemented using a comparator (instead of $e^{z^+} - e^{z^-}$). The system architecture and circuit diagram for the pattern classifier is shown in Fig. 4.16.

In this example, we generated a synthetic linearly separable dataset corresponding to classes: class I and class II. We used a linear support vector machine training algorithm [63] to determine weight $\mathbf{w}$. The parameter vector $\mathbf{w}$ is then programmed as currents on the prototype chip. Test

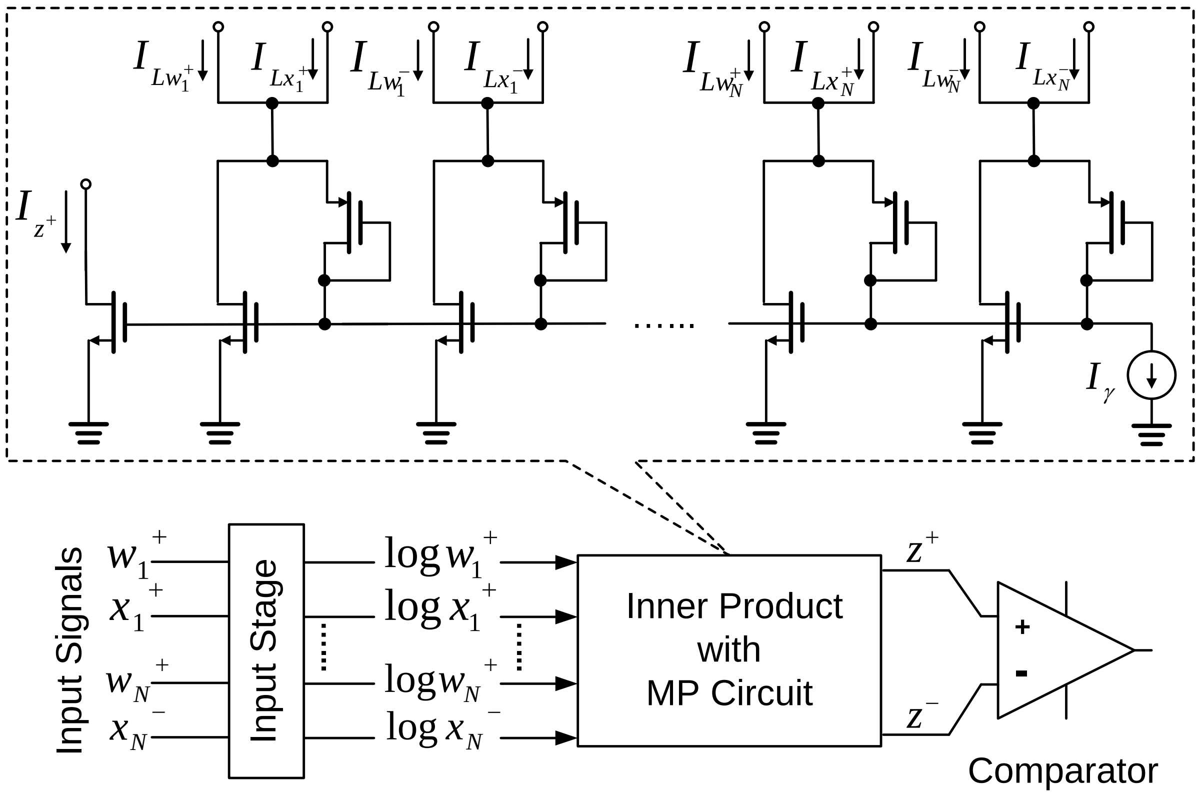


Figure 4.16: System architecture and circuit diagram of the MP-based pattern classifier.

vectors $\mathbf{x}$ were selected randomly and were then applied as input to the chip. For the sake of visualization, we show only results from a two-dimensional dataset. In this special case, the classifier equation (4.31) reduces to a two-dimensional inner product formulation as described in the example in section I. The circuit implementation is therefore identical to that of a full quadrant multiplier in Fig. 4.10. Fig. 4.17 shows the classification boundary and the classifier scores (negative values are represented by darker shades, whereas positive values are represented by lighter shades) obtained from software simulations and from the fabricated prototype under different biasing conditions. Again, the results show that the operation of the prototype is bias-scalable and the classification performance has been verified to be similar to that of the software model.

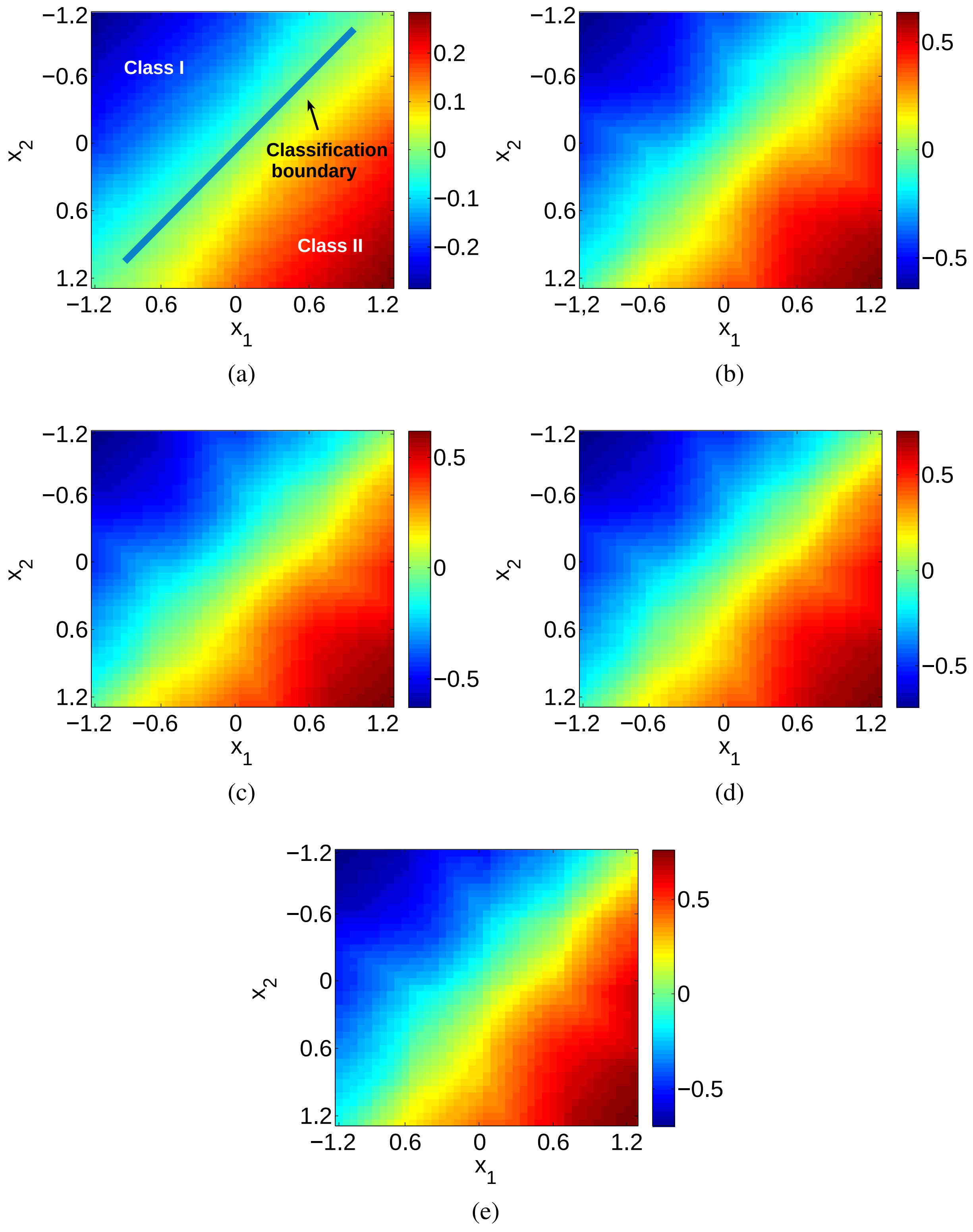


Figure 4.17: Classifier: $z^{+} - z^{-}$ simulated by (a) log-sum-exp function (simulation); (b) MP approximation (simulation); and measured $z^{+} - z^{-}$ when the transistors are biased in (c)strong inversion (measurement); (d)moderate inversion (measurement); (e)weak inversion (measurement).

Chapter 5

An analog LDPC decoder implemented on MP-based PWL circuits

5.1 Performance metrics and trade-off of LDPC decoder

Low-density parity-check codes [64] constitute an important class of capacity approaching error-correcting codes which has seen widespread acceptance in emerging communication standards [65, 66, 67]. One of the key factors behind the success of LDPC codes is its iterative decoding algorithms [68, 69, 70, 71] which are scalable and hence can be easily mapped onto digital [72, 73, 74, 75] and analog [76, 77, 78, 79] hardware.

As applications of LDPC codes expand into new areas like sensor networks [80, 81, 82] it is becoming more important to understand theoretical foundations that govern and determine the energy efficiency of LDPC decoders. Even though numerous studies have been reported in literature that evaluate the performance of LDPC decoding in regards to its bit-error-rate (BER) performance for different signal-to-noise ratios (SNR) [83, 84], there exists no theoretical basis for understanding the BER/SNR trade-off with respect to the decoder's energy efficiency. Understanding this trade-off is important because:

- Different applications impose different constraints on the energy efficiency of the decoding algorithm. For instance a decoder used in sensor networks will be more severely constrained for energy [81, 82] than a decoder which is used in a base-station of a wireless network. Therefore, decoding algorithms that provide an elegant trade-off between BER, SNR and power-dissipation is important as superior energy efficiency can be achieved by sacrificing BER performance.

- Understanding the trade-off could lead to novel class of decoding algorithms that deliver the similar BER performance as existing state-of-the-art algorithms while demonstrating superior energy efficiency.

One of the ways to quantify the energy efficiency of a decoding algorithm is to evaluate the sparsity of the messages being passed between the nodes of the bipartite graph during LDPC decoding. Here we have considered one notion of sparsity: L_0 norm of the messages, which is defined as the percentage of the non-zero messages out of a fixed total. Since zero-message consumes no energy, it is obvious to see that with the same amount of messages, the higher percentage of the zero-messages there is, the lower the L_0 norm is, and also the lower energy consumption. In this regard, the overall energy consumption can be characterized by the L_0 norm of the messages. This method of energy analysis is relevant to analog implementations because decoding of large LDPC codes is communication intensive rather than being computation intensive [77, 78, 79]. Also, it has been shown that in emerging deep sub-micron and nanoscale integrated circuits, energy dissipated during communication (through interconnects) will be significantly larger than the energy cost of local computations [85]. We use density evolution (DE) [86, 87, 84] as a tool to observe and analyze the L_0 norm of the messages generated during iterative LDPC decoding.

5.2 Density evolution and message sparsity

5.2.1 Density evolution analysis for conventional LDPC decoding algorithms

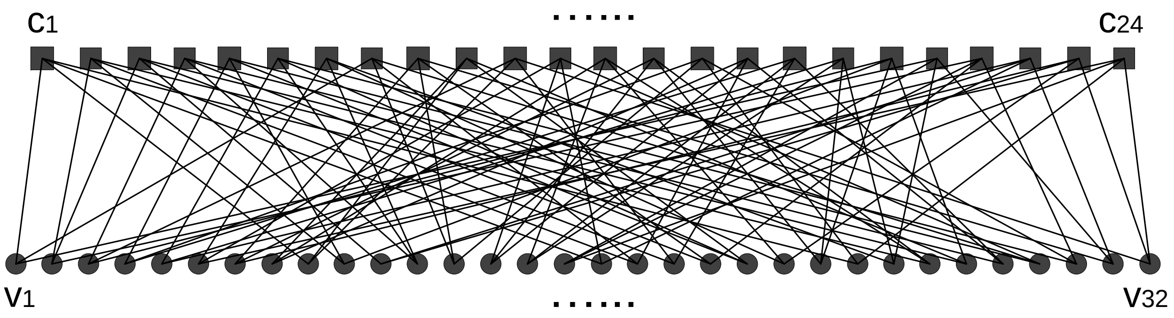


Figure 5.1: Factor graph corresponding to a 32-bit $(3, 4)$ LDPC code

Density evolution is a well-established technique to determine the asymptotic performance of any LDPC code for a given channel condition and a code-rate [87]. It provides a way to study the decoding algorithm by analyzing the evolution of message densities on a hypothetical infinite tree-like graph with similar local topology as the LDPC code of interest [88]. In this section, we first provide a brief introduction of density evolution analysis whose details can be found in [87]. Consider an example of a factor graph shown in Fig. 5.1. It consists of *variable nodes* $v_k, k = 1, .., N$ which are connected to *check nodes* $c_i, i = 1, .., M$ using *edges*. For the description that follows, the number of edges associated with each *check* node and *variable* node (also known as the degree of the node) will be denoted by d_c and d_v.

Let $\mathbf{V}_i$ denote the set of check nodes connected to the variable node $\mathbf{v}_i$ and $\mathbf{V}_{i\sim j}$ represent the set of check nodes other than $\mathbf{c}_j$ that are connected to variable node $\mathbf{v}_i$. Similarly, let $\mathbf{C}_j$ denote the set of variable nodes connected to the check node $\mathbf{c}_j$ and $\mathbf{C}_{j\sim i}$ represent the set of variable nodes other than the node $\mathbf{v}_i$ connected to $\mathbf{c}_j$.

In a sum-product based LDPC decoding algorithm [69], each check node $\mathbf{c}_j$ receives messages from its set of neighbors $\mathbf{C}_j$ (denoted by $L(\mathbf{v}_i \rightarrow \mathbf{c}_j)$) and computes messages to be sent to the variable nodes $\mathbf{v}_i \in \mathbf{C}_j$ (denoted by $L(\mathbf{c}_j \rightarrow \mathbf{v}_i)$) according to

$$L(\mathbf{c}_j \rightarrow \mathbf{v}_i) = 2\tanh^{-1}\left[\prod_{\mathbf{v}_k \in \mathbf{C}_{j\sim i}} \tanh\left(\frac{L(\mathbf{v}_k \rightarrow \mathbf{c}_j)}{2}\right)\right] \tag{5.1}$$

In subsequent iterations, each variable node $\mathbf{v}_i$ receives messages from its neighboring check nodes $\mathbf{c}_j \in \mathbf{V}_i$ and re-computes messages that will be sent to the check node $\mathbf{c}_j$ (denoted by $L(\mathbf{v}_i \rightarrow \mathbf{c}_j)$) according to

$$L(\mathbf{v}_i \rightarrow \mathbf{c}_j) = L(\mathbf{v}_i) + \sum_{\mathbf{c}_k \in \mathbf{V}_{i\sim j}} L(\mathbf{c}_k \rightarrow \mathbf{v}_i) \tag{5.2}$$

where $L(\mathbf{v}_i)$ denotes the initial messages obtained from the communication channel. Messages are then propagated back and forth between the variable and check nodes for a pre-determined number of iterations before a decision on the received bits is made [69]. An alternative implementation of the LDPC decoder is based on the min-sum approximation of the $L(\mathbf{c}_j \rightarrow \mathbf{v}_i)$ in equation (5.1) according to

$$L(\mathbf{c}_j \rightarrow \mathbf{v}_i) = L(\mathbf{v}_k \rightarrow \mathbf{c}_j), \tag{5.3}$$

where $k = \arg\min_{\mathbf{v}_{\hat{k}} \in \mathbf{C}_{j\sim i}} |L(\mathbf{v}_{\hat{k}} \rightarrow \mathbf{c}_j)|$.

Even though min-sum decoders are relatively easy to implement on hardware compared to sum-product decoders, we will show later in this section and in section 5.3 that message distributions that the algorithm produces are less sparse and hence its energy efficiency is inferior compared to that of sum-product decoders. In this regard, density evolution will serve as an important compu-

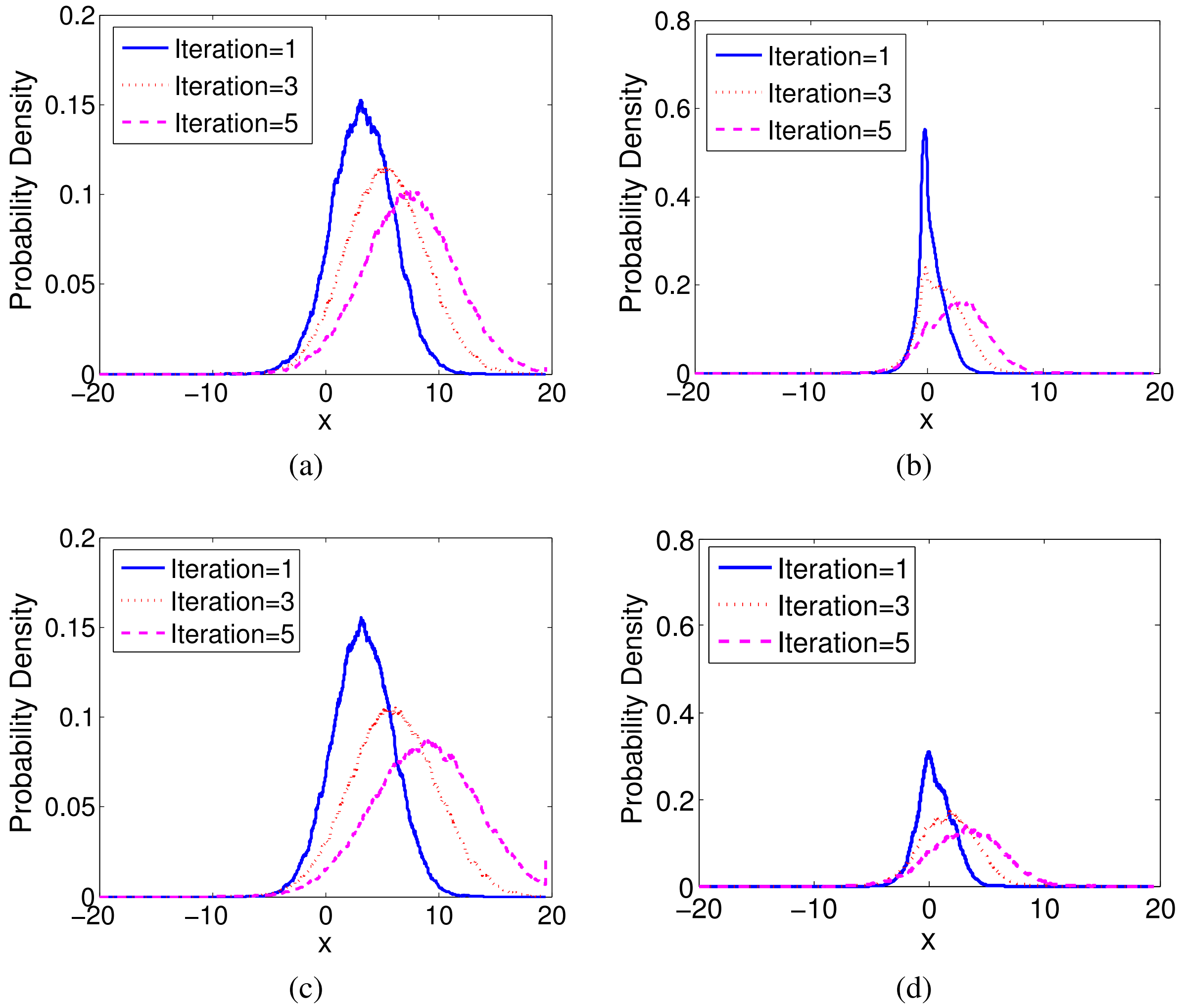


Figure 5.2: A (3,6) LDPC message PDF evolution for: (a) sum-product variable to check messages; (b) sum-product check to variable messages; (c) min-sum variable to check messages; (d) min-sum check to variable messages;

tational tool to track distributions of messages that are propagated between the variable and check nodes and hence visualize its sparsity. The DE procedure computes the probability density of messages during an iterative decoding algorithm based on a recursive estimation procedure that can be evaluated analytically or numerically using Monte-Carlo simulations [87, 86, 84]. By observing the asymptotic message distributions, traditional DE analysis estimates a minimum threshold parameter (e.g. noise-variance of a Gaussian channel) that ensures BER performance less than some specific tolerance. Even though the asymptotic behavior is of importance in DE, we will use the formulation for estimating sparsity of messages which will serve as a theoretical estimator for energy efficiency for a given decoder. The underlying assumption in DE analysis is that the graph rooted at a variable node, and incident by messages originating at the variable node, is cycle-free (for a finite number of iterations) which allows presumption of independence between incoming messages [88]. The independence assumption implies that the summation of message at variable and check nodes translates to convolution of their probability distributions, making it easier to track the evolution of the message densities [86]. We now outline equations which concisely capture the analytical framework for DE analysis. Let the degree distribution function $\lambda(x)(\rho(x))$ for variable (check) nodes be represented as

$$\lambda(x) = \lambda_2 x + \lambda_3 x^2 + \cdots + \lambda_n x^{(n-1)} = \sum_{i=2}^{n} \lambda_i x^{(i-1)} \tag{5.4}$$

$$\rho(x) = \rho_2 x + \rho_3 x^2 + \cdots + \rho_{q+1} x^q = \sum_{i=2}^{q+1} \rho_i x^{(i-1)} \tag{5.5}$$

where,

ρ_i is the fraction of edges connected to *check nodes* with degree i.

λ_i is the fraction of edges connected to *variable nodes* with degree i.

x^i represents convolution of the message density x with itself $(i-1)$ times.

Note: The two tuples (λ, ρ) are often referred to as the "*Code Degree Distribution*"

Also let the message distributions corresponding to *check* and *variable* nodes after ℓ message passing iterations be denoted by R_ℓ and P_ℓ. Then, DE analysis computes R_ℓ and P_ℓ recursively according to:

$$R_\ell = \Gamma^{-1}\rho\left(\Gamma\left(P_0 \otimes \lambda(R_{\ell-1})\right)\right) \tag{5.6}$$

$$P_\ell = P_0 \otimes \lambda(R_{\ell-1}) \tag{5.7}$$

where P_0 represents the probability distribution of messages received from the communication channel, $\otimes$ represents convolution operation and Γ is the Probability Density Function (PDF) transformation due to the non-linear operation at each of the check node. For instance, in sum-product decoding Γ is associated with the variable change operation related to the $\tanh(.)$ function in equation (5.1). Note that for sum-product decoding $\Gamma = \Gamma^{-1}$.

At final step of the DE recursions given by equation (5.7), the fraction of incorrect messages propagated from the variable nodes to the check nodes, can be determined using:

$$P_e^\ell(\lambda, \rho) = \sum_x P_\ell(x) \tag{5.8}$$

where x is a quantized random variable within a bounded range correspondent to the codeword transmitted.

The probability distribution functions computed by DE at each iteration step ℓ can also be used to compute metrics that quantify sparsity of messages. One metric of sparsity which is commonly used in signal processing literature [89] is the L_0 norm. For a vector θ, $||\theta||_0$ is denoted as the number of non-zeros of θ. However, most frequently the size of the vector is taken into account. Thus $||\theta||_0$ is normalized as the percentage of the non-zero coefficients [90]. Let $\mathcal{L}^\ell(c \to v)(\mathcal{L}^\ell(v \to c))$ denote the set of messages propagated from check(variable) nodes to variable(check) nodes during the ℓ-th iteration. The L_0 norm for $\mathcal{L}^\ell(c \to v)(\mathcal{L}^\ell(v \to c))$ can be represented by $\|\mathcal{L}^{\ell}(c \rightarrow v)\|_0(\|\mathcal{L}^{\ell}(v \rightarrow c)\|_0)$. The PDFs computed by DE give out the discrete message density. Thus it is convenient to calculate the normalized L_0 norm with

$$\|\mathcal{L}^{\ell}(c \rightarrow v)\|_0 = \sum_{x \neq 0} R_{\ell}(x) \tag{5.9}$$

$$\|\mathcal{L}^{\ell}(v \rightarrow c)\|_0 = \sum_{x \neq 0} P_{\ell}(x) \tag{5.10}$$

The random variable x in (5.9) and (5.10) is quantized.

In Figure 5.2 we show the mean message PDFs obtained during iterative decoding of a length 2000, (3,6) LDPC code. The decoding algorithms employed are sum-product and min-sum. To obtain the PDFs a zero codeword was transmitted through an additive white gaussian noise (AWGN) channel with a SNR of $2.5dB$, before the decoding was carried out. The figure shows trends similar to those previously reported for DE analysis in [87] [86]. The distribution P_{ℓ} (Fig. 5.2 (a) and (c)) shifts to the right reducing the area under the curve to the left of the origin. As a result, after several iterations the probability of error calculated according to equation (5.8) is small. Additionally, we can infer (by comparing Fig. 5.2 (b) and (d)) that the PDF of the messages propagated from check nodes to variable nodes in ℓ-th iteration, denoted by $\mathcal{L}^{\ell}(c \rightarrow v)$, for the sum-product algorithm is more concentrated around zero than that of the min-sum algorithm. Consequently, the sum-product algorithm has lower L_0 norm for $\mathcal{L}^{\ell}(c \rightarrow v)$ as against the min-sum algorithm. Overall, the L_0 norm of $\mathcal{L}^{\ell}(c \rightarrow v)$ increases with the number of iterations ℓ. And the message distribution approaches a gaussian distribution with increasing number of iterations. This observation has been verified by several previous studies [91, 92] and is also the basis of gaussian approximation based analysis of the sum-product algorithm [91].

5.2.2 Sparse LDPC decoding with MP

To understand how one can achieve sparsity in LDPC decoding, we analyze the core transforms that are related with the operator Γ for DE recursions given in equation (5.6). Since the check node computations in LDPC decoding can be performed pair-wise, we will analyze the properties of two dimensional transforms related to Γ. For the sum-product decoding the check function is given by [47]

$$\phi_y^{sp}(x) = \log\left(\frac{1+e^{x+y}}{e^x+e^y}\right) \tag{5.11}$$

where x and y are the two variables of function ϕ representing the messages in (5.1) and (5.2). (5.11) represents the result of combining two messages x and y. This can be recursively applied to the sum-product checking function computation.

For min-sum decoding the equivalent operation $\phi_y^{ms}(x)$ is given by [47]

$$\phi_y^{ms}(x) = \max(0, x+y) - \max(x, y) \tag{5.12}$$

where we have similar x and y as in (5.11). (5.12) is recursively applicable in checking function computation as well.

Figure 5.3 compares the functions $\phi_y^{sp}(x)$ and $\phi_y^{ms}(x)$ for $y = 1.0$. The plot shows that around the origin, the behavior of both functions are linear. As a result, during the first few iterations of LDPC decoding, when the value of messages propagated from the check to variable nodes are concentrated around the origin (see Fig. 5.2), the resulting distribution obtained using the transform Γ is non-sparse. This indicates that if we choose a transform function containing a "dead-zone" around the origin, and at the same time approximate the sum-product transformation, we could obtain a sparser LDPC decoding procedure.

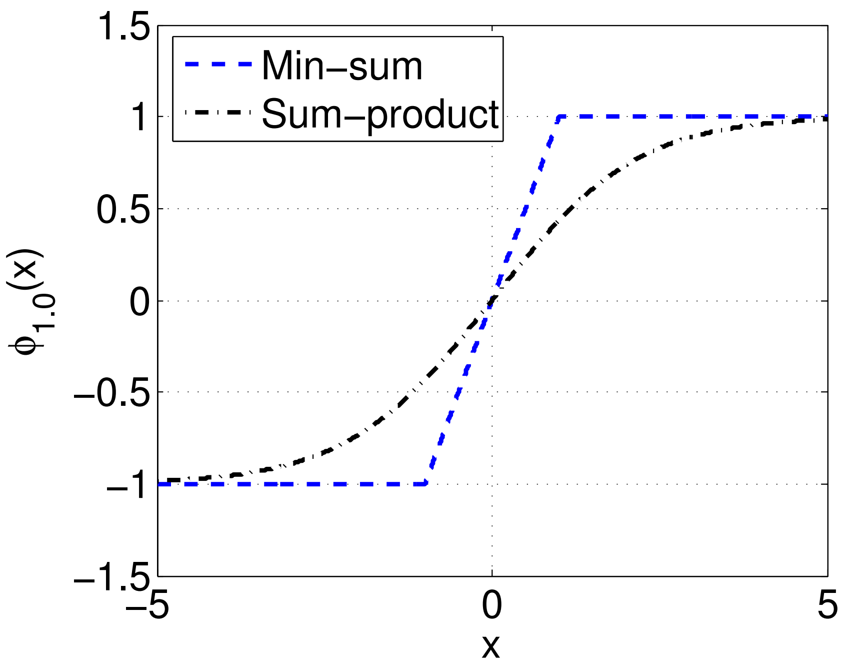


Figure 5.3: Comparison of pair-wise decoding functions for sum-product and min-sum LDPC decoding algorithms

In the first step of our approximation procedure, the sum-product transform is decomposed as

$$\phi_y^{sp}(x) = \log\left(1 + e^{x+y}\right) - \log\left(e^x + e^y\right) \tag{5.13}$$

which is a difference of two "log-sum-exp" function whose general form can be expressed as

$$z_{log} = \log\left(\sum_{i=1}^{N} e^{L_i}\right) \tag{5.14}$$

where $\{L_i\}, i = 1, .., N$ denote a set of log-likelihood ratios which are used in DE analysis and LDPC decoding.

Equation (5.14) can be re-written equivalently as:

$$\sum_{i=1}^{N} e^{\left(L_i - z_{\log}\right)} = 1 \tag{5.15}$$

Next, we lower-bound the equality (5.15) using the inequality $[x]_+ \leq e^{(x)}$ where $[x]_+ = \max(x, 0)$ is a threshold function. Equation (5.15) can then be expressed in its piecewise-linear form as

$$\sum_{i=1}^{N} \left[L_i - z_{\log}\right]_+ \leq 1. \tag{5.16}$$

As is seen from equation (5.16), if $z_{\log}$ in is replaced by z in equation (3.1), and ≤ 1 is changed into γ in equation (3.1), equation (5.16) is turned out to be MP algorithm.

When the MP-based PWL approximation is applied to the "log-sum-exp" functions in equation (5.11) the MP-based check function is obtained as:

$$\phi_y^{mp}(x) = M(\{0, x+y\}, \gamma) - M(\{x, y\}, \gamma) \tag{5.17}$$

Fig. 5.4 compares the margin propagation check functions to its sum-product and min-sum equivalent for different values of the hyper-parameter γ. It can be seen that the margin check function is a PWL approximation to the sum-product check function and the width of the "dead-zone" around the origin is controlled by the hyper-parameter γ. When γ diminishes to zero, the "dead-zone" shrinks to zero as well and the margin propagation algorithm converts to min-sum, as can be seen through figure 5.4. The "dead-zone" can also be clearly seen in the three-dimensional plot of the check functions illustrated in Figure 5.5. Note that for the sum-product check function

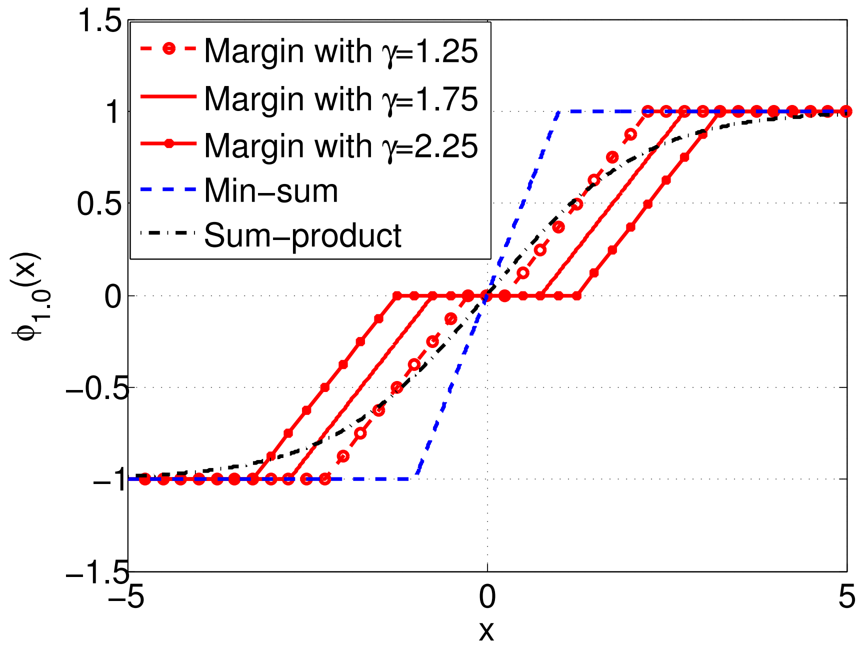


Figure 5.4: Comparison of pair-wise decoding function for sum-product, min-sum and margin propagation decoding algorithms

(Figure 5.5(a)) and min-sum check function (Figure 5.5(b)) the origin is the saddle point and the gradient at the origin is non-zero.

5.2.3 Density evolution analysis of MP

As shown above, at the core of the margin propagation based decoding is the margin approximation algorithm. Consequently we first simplify the margin approximation algorithm. If we consider two likelihood scores with levels ℓ_1 and ℓ_2, the solution to the algorithm is the level z. Figure 5.6 displays the possible solutions for different values of ℓ_1 and ℓ_2. Mathematically the solution for

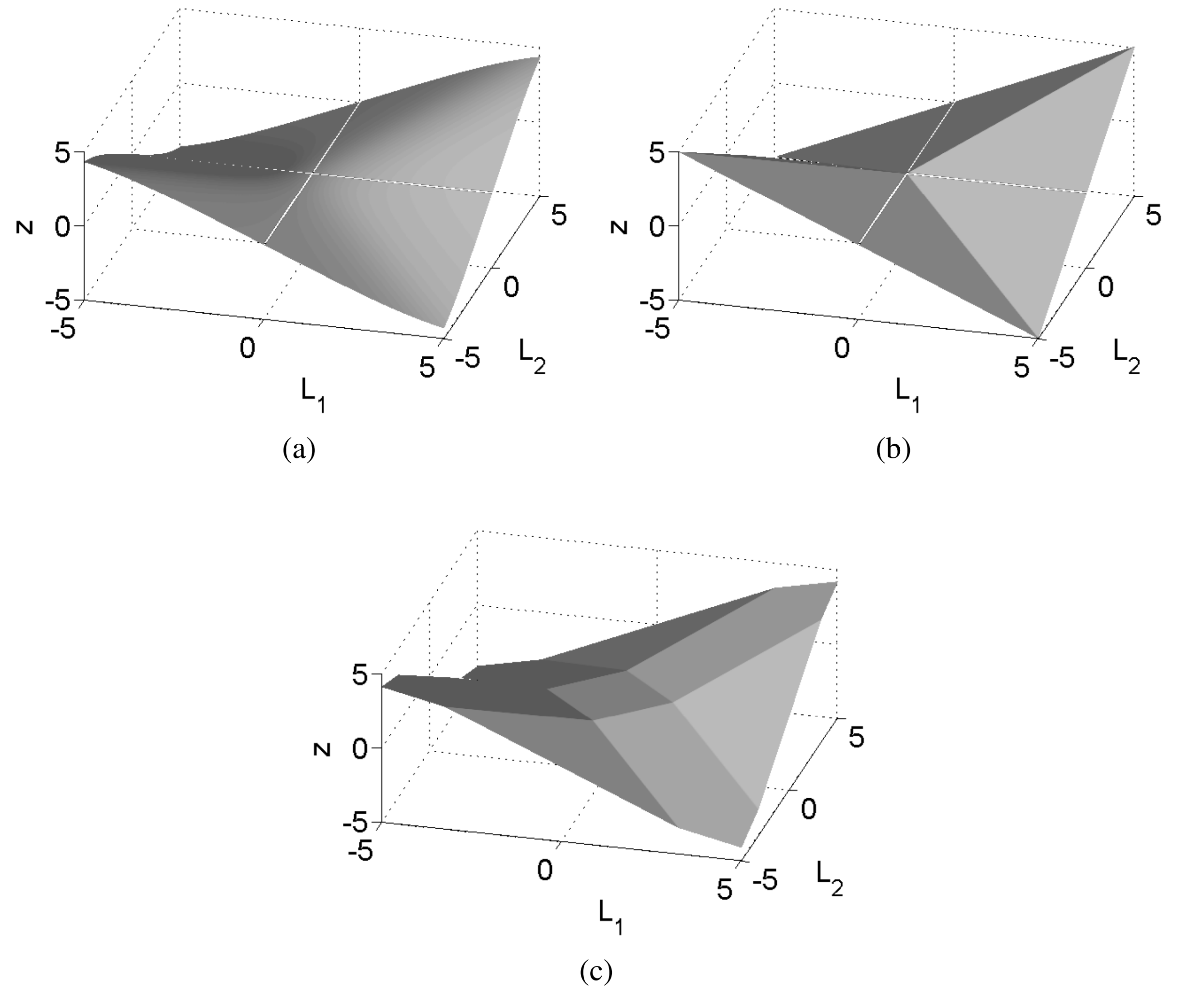


Figure 5.5: Comparison of 3D LDPC check function for (a) sum-product (b) min-sum and (c) margin propagation with γ=1.75

$M(l_1, l_2)$ can be stated as:

$$z = \begin{cases} \max(\ell_1, \ell_2) - \gamma & \text{if } |\ell_1 - \ell_2| > \gamma \\ \dfrac{\ell_1 + \ell_2 - \gamma}{2} & \text{otherwise} \end{cases} \tag{5.18}$$

$$= \max\left\{ \max(\ell_1, \ell_2) - \gamma, \frac{\ell_1 + \ell_2 - \gamma}{2} \right\} \tag{5.19}$$

If the (recursive) operation at the check side of an LDPC graph is represented as $L_1 \boxplus L_2$, then:

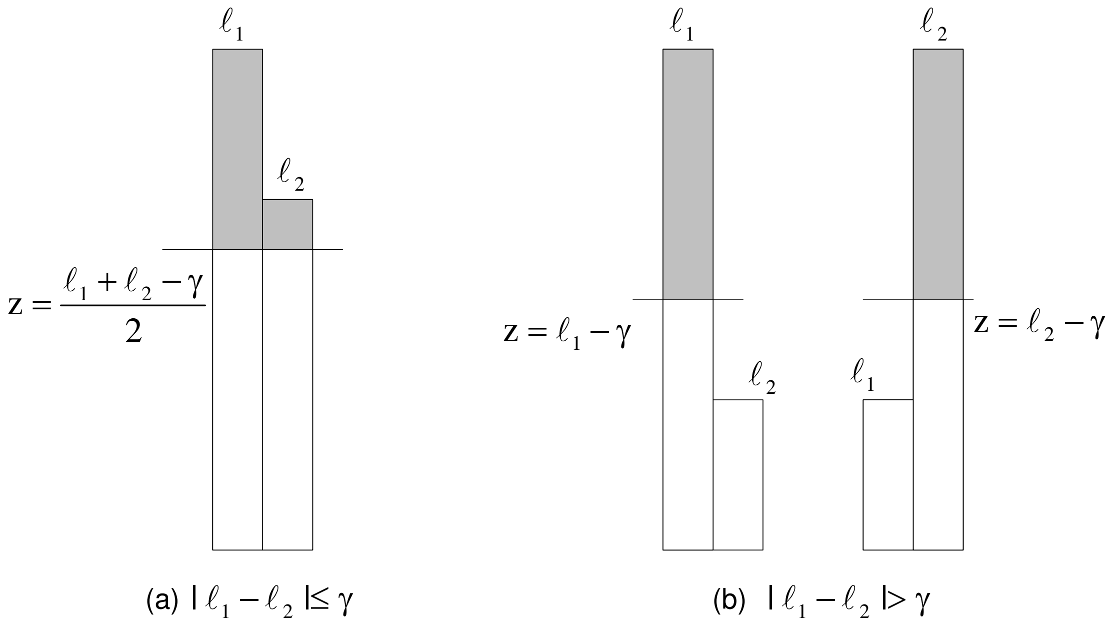

ℓ_1
ℓ_2
$z = \frac{\ell_1 + \ell_2 - \gamma}{2}$
(a) $|\ell_1 - \ell_2| \leq \gamma$
ℓ_1
ℓ_2
$z = \ell_1 - \gamma$
ℓ_2
ℓ_1
$z = \ell_2 - \gamma$
(b) $|\ell_1 - \ell_2| > \gamma$

Figure 5.6: Possible solutions z for $M(\ell_1, \ell_2)$

$$\begin{aligned} L_1 \boxplus L_2 &= M(L_1^+ + L_2^+, L_1^- + L_2^-, \gamma) \\ &\quad - M(L_1^+ + L_2^-, L_1^- + L_2^+, \gamma) \\ &= z_1 - z_2 \end{aligned} \tag{5.20}$$

where, $L_1^+, L_1^-, L_2^+, L_2^-$ are the differential forms of L_1, L_2 given by:

$$\begin{aligned} L_1^+ &= T + L_1/2 \\ L_1^- &= T - L_1/2 \\ L_2^+ &= T + L_2/2 \\ L_2^- &= T - L_2/2 \end{aligned} \tag{5.21}$$

where T denotes the common mode signal.

Using equation (5.18) the terms z_1 and z_2 in equation (5.20) can be rewritten as:

$$z_1 = \begin{cases} \max(L_1^+ + L_2^+, L_1^- + L_2^-) - \gamma & \\ \qquad \text{if } |L_1^+ + L_2^+ - L_1^- - L_2^-| \geq \gamma & \\ \dfrac{L_1^+ + L_2^+ + L_1^- + L_2^- - \gamma}{2} & \text{otherwise} \end{cases} \tag{5.22}$$

$$z_2 = \begin{cases} \max(L_1^+ + L_2^-, L_1^- + L_2^+) - \gamma & \\ \qquad \text{if } |L_1^- + L_2^+ - L_1^+ - L_2^-| \geq \gamma & \\ \dfrac{L_1^+ + L_2^- + L_1^- + L_2^+ - \gamma}{2} & \text{otherwise} \end{cases} \tag{5.23}$$

The computations outlined in equation (5.20), (5.22) and (5.23) transform the PDFs of the incoming messages L_1, L_2 into a new PDF. This PDF transformation can be captured “exactly” by using the surjective function corresponding to the operation $L_1 \boxplus L_2$. Let us represent this PDF transfer function as follows:

$$\mathbb{MP} : P_{L_1}(\ell_1) \times P_{L_2}(\ell_2) \to P_Z(z), \text{ where } \ell_1, \ell_2, z \in \mathbb{R} \tag{5.24}$$

More specifically the resulting PDF is calculated as follows:

$$\begin{aligned} P_Z(z) &= \sum_{\ell_1 \in \mathbb{R}} \sum_{\{\ell_2 \in \mathbb{R} \mid \ell_1 \boxplus \ell_2 = z\}} P_{L_1,L_2}(\ell_1, \ell_2) \bigtriangleup \ell_2 \bigtriangleup \ell_1 \\ &= \sum_{\ell_1 \in \mathbb{R}} \sum_{\{\ell_2 \in \mathbb{R} \mid \ell_1 \boxplus \ell_2 = z\}} P_{L_1}(\ell_1) P_{L_2}(\ell_2) \bigtriangleup \ell_2 \bigtriangleup \ell_1 \end{aligned} \tag{5.25}$$

In the above equation, the cycle-free assumption [86] for the message incident sub-graph (rooted at a variable node) allows us to express the joint distribution $P_{L_1,L_2}(\ell_1, \ell_2)$ as the product of marginals $P_{L_1}(\ell_1)P_{L_2}(\ell_2)$. Equivalent to the parity degree distribution function (5.5) used in density evolution analysis of the sum-product algorithm, we can now define the following parity degree distribution function for margin propagation:

$$\begin{aligned} \rho_{MP}(x) &= \rho_2 x + \rho_3 x^{\odot 2} + \cdots + \rho_{q+1} x^{\odot q} \\ &= \sum_{i=2}^{q+1} \rho_i x^{\odot(i-1)} \end{aligned} \tag{5.26}$$

where,

$x^{\odot i}$ represents $\underbrace{\mathbb{MP} \cdots \mathbb{MP}(\mathbb{MP}}_{i \text{ times}}(x,x),x) \cdots x)$

Consequently, the density evolution analysis for margin propagation at the check side becomes:

$$R_\ell = \rho_{MP}\left(P_0 \otimes \lambda(R_{\ell-1})\right) \tag{5.27}$$

Using equations (5.7), (5.25), (5.26) and (5.27) we determine the AWGN noise threshold for a (3,6) and an irregular LDPC code which we list in table 5.1 (under exact analysis column). Note, the outer integral over l_1 in the PDF equation (5.25) (corresponding to $L_1 \boxplus L_2$) involves calculation over the entire real line $\mathbb{R}$, which we can restrict to large enough values for practical considerations. However, determining the domain for the second integral in equation (5.25) involves a search operation identifying the region where equation $\ell_1 \boxplus \ell_2 = z$ is satisfied. This search operation significantly increases the complexity of density evolution analysis for margin propagation. We therefore look for simplifications which reduce the complexity involved in computing thresholds for the margin propagation algorithm without sacrificing too much accuracy. A similar approach aiming to reduce complexity has been proposed for sum-product threshold calculation and is often referred to as the gaussian approximation analysis for sum-product [91].

To obtain approximations which simplify the analysis of margin propagation decoding let us consider the expressions involved in the first part of equation (5.22) in conjunction with equation (5.21). We get:

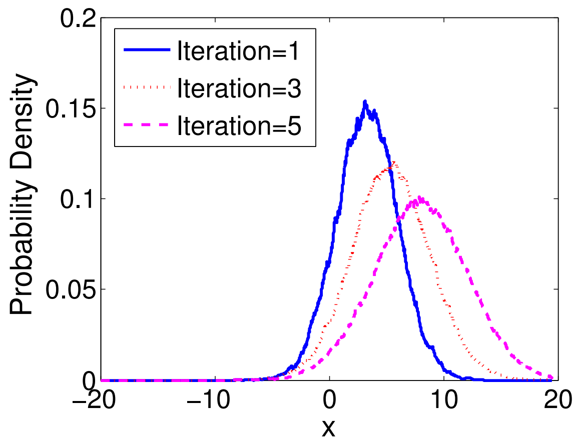
$$\begin{aligned} L_1^+ + L_2^+ &= T + \frac{L_1}{2} + T + \frac{L_2}{2} && = 2T + \frac{1}{2}\left[L_1 + L_2\right] \\ L_1^- + L_2^- &= T - \frac{L_1}{2} + T - \frac{L_2}{2} && = 2T - \frac{1}{2}\left[L_1 + L_2\right] \end{aligned}$$

To simplify our representation we substitute $U = \frac{1}{2}\left[L_1 + L_2\right]$, then:

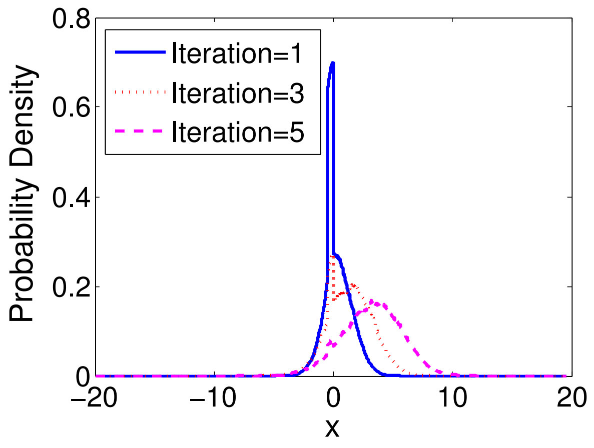
$$\begin{aligned} L_1^+ + L_2^+ &= 2T + U \\ L_1^- + L_2^- &= 2T - U \end{aligned} \tag{5.28}$$

Substituting the representation (5.28) in first part of equation (5.22) gives:

$$\begin{aligned} \max(L_1^+ + L_2^+, L_1^- + L_2^-) - \gamma &= 2T + \max(U, -U) - \gamma \\ &= 2T + |U| - \gamma \end{aligned} \tag{5.29}$$



(a)



(b)

Figure 5.7: A (3,6) LDPC message PDF evolution for: (a) margin propagation variable to check messages; (b) margin propagation check to variable messages;

Table 5.1: Threshold comparison for AWGN channel under margin propagation and sum-product

$\lambda(x)$	$\rho(x)/\rho_{MP}(x)$	Margin Propagation threshold ($\gamma = 1.75$)		Sum-Product threshold
		Exact density evolution analysis	Approximate density evolution analysis	
x^2	$x^5/x^{\odot 5}$	1.14 dB	1.29 dB [1]	1.10 dB [87]
$0.4x + 0.6x^2$	$x^4/x^{\odot 4}$	1.09 dB	0.95 dB	1.07 dB [93]

Substituting the representation (5.28) in second part of equation (5.22) gives:

$$\begin{aligned}\frac{L_1^+ + L_2^+ + L_1^- + L_2^- - \gamma}{2} &= \frac{1}{2}\left[2T + U + 2T - U\right] - \frac{\gamma}{2}\\ &= 2T - \frac{\gamma}{2}\end{aligned} \tag{5.30}$$

Using equations (5.19), (5.29) and (5.30) we can rewrite z_1 as:

$$\begin{aligned}z_1 &= \max\left[2T + |U| - \gamma, 2T - \frac{\gamma}{2}\right]\\ &= 2T + \max\left[|U| - \gamma, -\frac{\gamma}{2}\right]\end{aligned} \tag{5.31}$$

If common mode signal T is set to zero, the term $2T$ disappears from the above equations. Using arguments similar to those used for z_1 we can derive the following expression for z_2:

$$\begin{aligned} z_2 &= \max\left[2T + |V| - \gamma, 2T - \frac{\gamma}{2}\right] \\ &= 2T + \max\left[|V| - \gamma, -\frac{\gamma}{2}\right] \end{aligned} \tag{5.32}$$

Again, if we choose $T = 0$, the term $2T$ disappears from the above equations.

Substituting (5.31) and (5.32) in (5.20) gives:

$$L_1 \boxplus L_2 = \max\left[|U| - \gamma, -\frac{\gamma}{2}\right] - \max\left[|V| - \gamma, -\frac{\gamma}{2}\right] \tag{5.33}$$

Let us now focus on calculating the evolution of message probability densities. The absolute value operation just leads to the folding of the probability distribution. Moreover, since L_1 and L_2 are drawn from independent random variables (due to the cycle-free assumption) the summation or difference of the two used to obtain U and V just leads to the convolution of the probability density functions. The calculations associated with U and V are listed below:

$$U = \frac{1}{2}\left[L_1 + L_2\right] \tag{5.34}$$

$$V = \frac{1}{2}\left[L_1 - L_2\right] \tag{5.35}$$

The $c = \max(a, b)$ operation where a and b are drawn from two independent random variables A

[1]Check side density evolution is calculated by partitioning $x^{\odot 5}$ into $x^{\odot(4)+1}$. Density evolution corresponding to the part within the bracket is calculated using approximate analysis followed by exact analysis for the remaining part.

and B, gives c with probability distribution function C determined as follows [94]:

$$P(C=c) = P(A=c)P(B \leq c) + P(B=c)P(A \leq c) \tag{5.36}$$

Using the above arguments we can determine the PDFs of z_1 and z_2 from the PDFs of L_1 and L_2. However, determining the PDF of $z_1 - z_2$ is not trivial since in general z_1 and z_2 are drawn from dependent random variables. If however the random variables corresponding to L_1 and L_2 are independent identically distributed (i.i.d.) and gaussian in nature, the PDF of $z_1 - z_2$ is just the convolution of PDFs corresponding to z_1 and $-z_2$. Due to the i.i.d. and gaussian restriction, the density evolution computation for the parity side needs to be carried out in a particular order. For example, if we are interested in determining the PDF of $L_1 \boxplus L_2 \boxplus L_3 \boxplus L_4$, we first determine the PDFs corresponding to $(L_1 \boxplus L_2)$ and $(L_3 \boxplus L_4)$. This gives two different PDFs which are i.i.d. however they are not gaussian, so we approximate these PDFs with gaussian distributions before computing the PDF of $(L_1 \boxplus L_2) \boxplus (L_3 \boxplus L_4)$. Using the approximation approach described above in conjunction with equation (5.33) allows us to track the evolution of the message densities at the parity side of an LDPC graph when margin propagation based decoding is employed. However the i.i.d and gaussian requirements for the input, restricts our ability to track message densities, to parity degree distribution functions which have the following representation:

$$\rho_{MP}(x) = \sum_{i \in \{2^0+1, 2^1+1, 2^2+1 \cdots\}} \rho_i x^{\odot(i-1)} \tag{5.37}$$

AWGN noise thresholds are obtained for a (3,6) and an irregular rate one-half LDPC code using the above approximation and listed in table 5.1. Noise thresholds obtained using exact and approximate analysis are found to be comparable. The approximate analysis noise threshold for margin propagation is within 0.16 dB of the actual threshold. The noise threshold for sum-product decoding is included to highlight the minimal performance loss for margin propagation. Thresholds obtained using exact analysis shows that the loss in performance for margin-propagation decoding is less than 0.04 dB.

Using the setup described in section 5.2, we can track the evolution of message densities for margin propagation decoding using simulations for a (3,6) LDPC code of length 1000. Figure 5.7 displays the message density evolution for margin propagation in 5 iterations. The left column corresponds to the messages from variable-side to check-side and right column corresponds to the message from check-side to variable-side. The message PDF behaves similar to that outlined in figure 5.2 (sum-product algorithm). As the iterations progress, the message PDF shifts to the right, reducing the probability of making errors.

5.3 Performance simulation results of MP-based LDPC decoder

This section presents Monte-Carlo simulations for evaluating the performance of LDPC decoding based on margin propagation. The results are presented in the following order: First, we analyze the effect of the hyper-parameter γ on the L_0 norm of the messages. Next we compare the BER performance of an optimized margin propagation with an equivalent sum-product and min-sum decoder. Next (keeping the BER performance the same) we compare the sparsity of LDPC decoding for MP, sum-product and min-sum algorithm by observing the L_0 norm of their messages. For all the simulations described in this section, binary phase shift keying (BPSK) has been assumed for modulation and the channel is assumed to be an AWGN channel. The experimental parameter being varied is the signal-to-noise ratio (SNR) of the AWGN channel which can be expressed as

$$SNR(dB) = -20 \log_{10} \sigma \tag{5.38}$$

where σ denotes the standard deviation of noise.

5.3.1 Effect of γ on L_0 norm

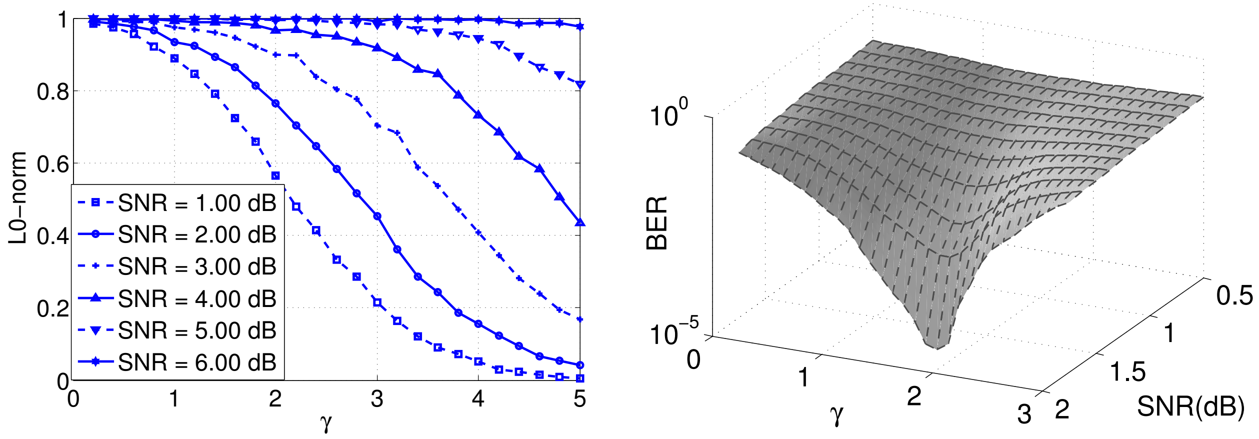


Figure 5.8: : (a)Effect of hyper-parameter γ on the L_0 norm of the check to variable messages(for different AWGN channel conditions) (b)Effect of hyper-tuning parameter γ on BER performance (for different AWGN channel conditions)

Figure 5.8a shows the L_0 norm of messages propagated from check side to variable side for different values of γ. For this experiment, a (2000,1000) regular LDPC code with a (3×6) parity check matrix H was chosen. The maximum number of decoder iterations was set to 20. The average L_0 norm was then computed over all the decoding iterations. The simulation results presented here consider only the average L_0 norm for messages propagating from the check to variable side. As we will show in the later experiment, the L_0 norm for messages propagated from variable to check side does not exhibit sparse behavior and hence does not provide any efficiency improvement over sum-product or min-sum algorithm.

When the parameter γ increases, figure 5.8a shows that the L_0 norm of the messages decreases. This can be attributed to the size of the "dead zone" in the margin transfer function (see Fig. 5.4) which increases with an increase in γ. Thus γ is a consistent metric to modulate the sparsity (L_0 norm) of the LDPC messages. Also for a fixed value of γ the L_0 norm increases with an increase in channel SNR. This is because lower SNR would concentrate the likelihood scores around the "dead-zone" (due to noise) thus leading to a sparser message distribution (lower L_0 norm). An important consequence of this experiment is that the sparsity of the MP decoding algorithm can be controlled using γ and can also be adaptively varied depending on the channel conditions.

To understand the trade-off between γ, SNR and the BER performance of the MP algorithm, we conducted a three parameter experiment for a length 1000 (3,6) LDPC code, whose results are summarized in Fig. 5.8b. It can be clearly seen that for a fixed channel SNR, there exists an optimal value of γ (hence sparsity) that yields the best BER performance. This indicates that a correct choice of γ could provide an optimal performance in terms of sparsity and BER performance. Also, Figure 5.8b could be used as a calibration curve which can be used for designing different LDPC decoders with varying specifications of sparsity (energy efficiency).

Figure 5.8b shows that with SNR ranging from $0.5dB$ to $2dB$, when the value of γ increases from as low as 0 to around 1.75, the BER surface lowers monotonically, however when γ increases from 1.75 to 3, the trend is reversed. And it can be seen that at the SNR of $2dB$, the "waterfall" phenomenon happens to the BER curve with γ equal to 1.75 ahead of others. Hence when SNR is ranging from $0.5dB$ to $2dB$, 1.75 is deemed to be the optimal value of γ to achieve the least BER.

5.3.2 Comparison of BER performance

In the next set of experiments we compare the BER performance of the optimal MP decoding algorithm to the equivalent sum-product and min-sum algorithms. Three different lengths of the (3,6) LDPC codes were chosen: 256, 10^4 and 10^6. For each of the runs, the optimal parameter γ was found to be 1.75. Figure 5.9 summarizes the BER performance for different algorithms. It can be seen from the results that for a 10^6 length code, min-sum decoding incurs a $0.3dB$ penalty in performance compared to its sum-product equivalent, whereas MP decoding incurs less than $0.03dB$ penalty. The result shows that the performance of MP and sum-product decoding is nearly identical and is consistent with observations made in table 5.1.

This has been verified to be true also for a rate 1/2 irregular code. Figure 5.10 compares the BER performance for a $(\lambda(x), \rho(x)) \equiv (0.4x + 0.6x^2, x^4)$ irregular LDPC code. We refer to this LDPC code as a (2.5,5) code. For this experiment, the optimal value of γ was found to be 1.75. It can also be seen from the figure that with a higher degree at both sides, the regular code performs better than the irregular code with penalty not less than $0.7dB$ for all the three algorithms. This is embodied by the BER "water-fall" which happens at a lower SNR and is more pronounced. However, the results presented here differ slightly from the BER metrics reported in the literature [87]. We attribute this difference to the number of decoding iterations used for the reported experiments.

5.3.3 Comparison of message sparsity

The previous results confirmed that the BER performance of an optimized MP decoding algorithm is near identical to the BER performance of the sum-product decoding algorithm. However, the MP decoding algorithm produces more sparse distribution of messages compared to the sum-product

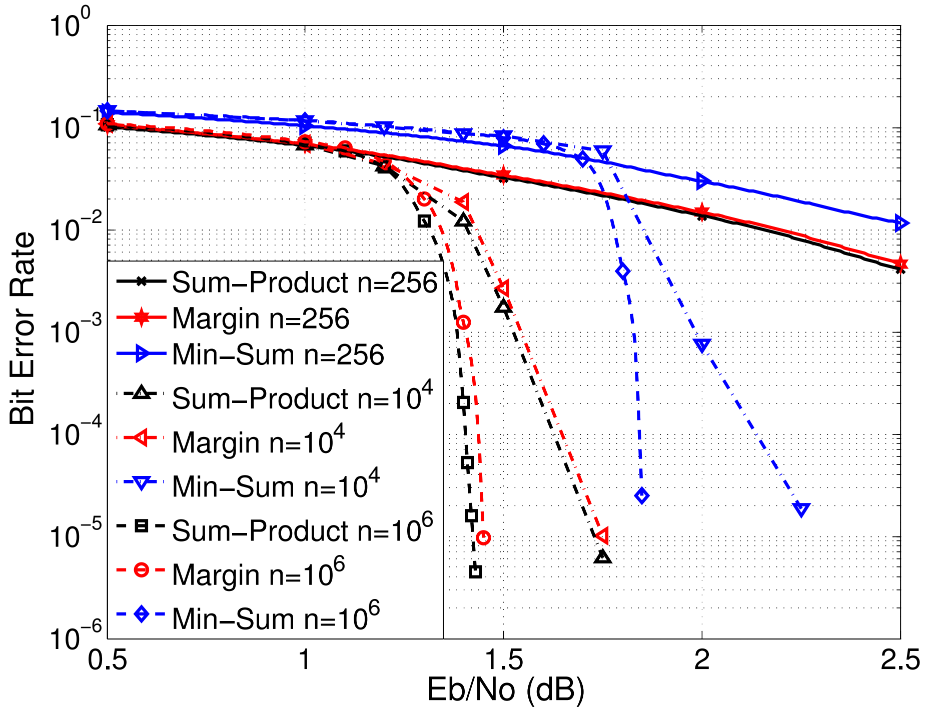


Figure 5.9: The comparison of BER for LDPC decoders based on different algorithms achieved by (3,6)-regular LDPC code of length 256, 10^4, and 10^6; note the maximum iteration number is set to 20

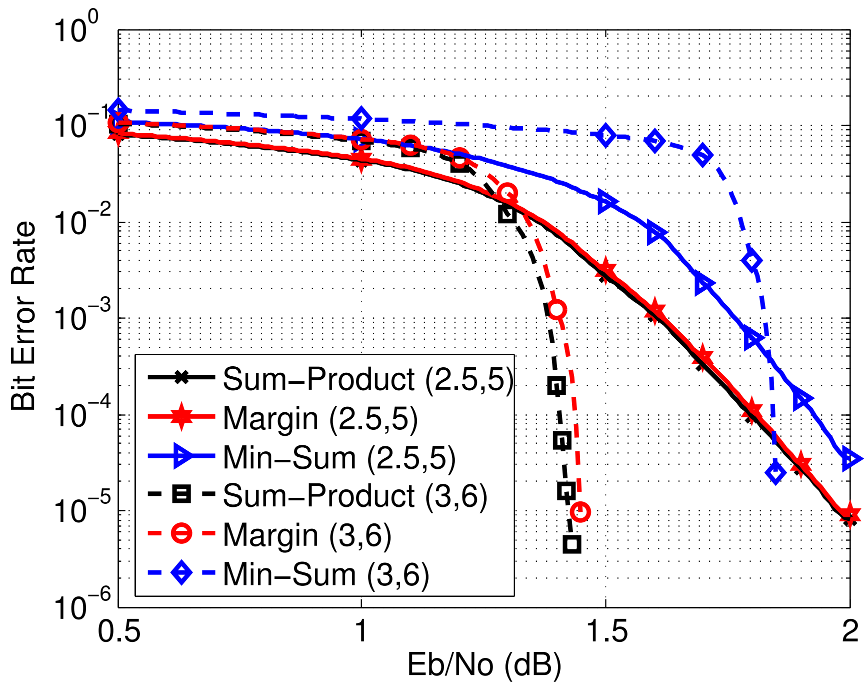


Figure 5.10: The comparison of BER for LDPC decoders based on different algorithms achieved by length 10^6 (3,6)-regular LDPC code and (2.5,5) irregular LDPC code; note the maximum iteration number is set to 20.

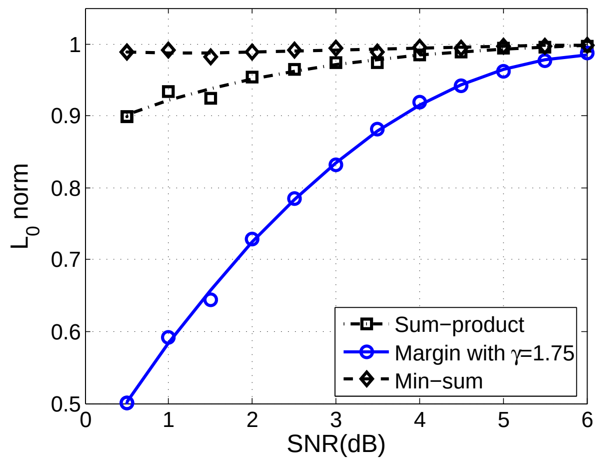
algorithm as confirmed by the following experiment. A $(2000, 1000)$ regular LDPC code with a (3×6) parity check matrix is used. Figure 5.11 shows the comparison of average L_0 norm for messages propagated between check (variable) and the variable (check) nodes. It can be seen that the L_0 norm (sparsity) of the messages propagated between the variable to check are similar. For a high SNR, the message convergence is considerably faster and requires only two iterations. However, the L_0 norm of messages propagated between the check-to-variable nodes exhibit a significant difference between the different algorithms. For the MP algorithm, the achieved sparsity is the best whereas for the min-sum algorithm the achieved sparsity is the worst. This result shows that, MP algorithm provides an elegant approach for achieving sparse decoding (hence superior energy efficiency) without sacrificing the BER performance.

5.4 Basic circuits of MP-based LDPC decoder

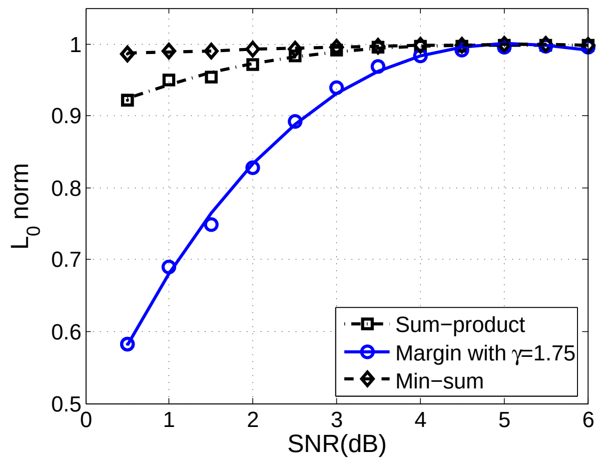
All the basic operations of check and variable nodes for this LDPC decoder have been illustrated in section 5.2. In this section, we present the basic modules of the MP based LDPC decoder. For check node, we take c_1 in Fig.5.1 as an example. Note differential circuits are used for both check node and variable node implementation.

5.4.1 Implementation of check node

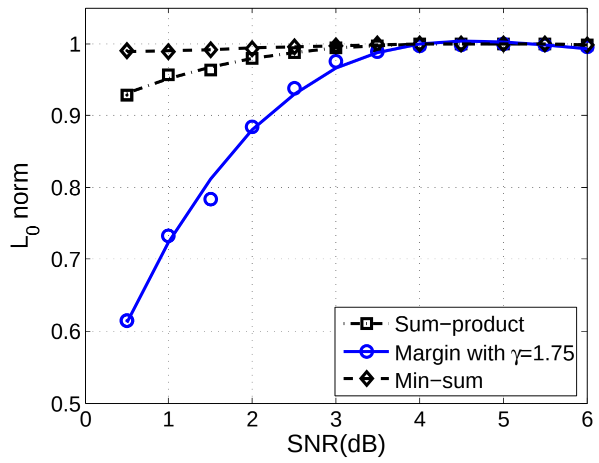
As shown in equation (3.1), MP approximation requires addition, subtraction and rectification. Each of these operations can be implemented in a CMOS process using basic conservation laws (e.g. Kirchhoff's current summation) which scale across transistor operating regimes (weak, moderate and strong inversion). Therefore, the same decoder architecture can be operated at different decoding speeds by scaling the bias current and hence by scaling its power dissipation. In this sec-



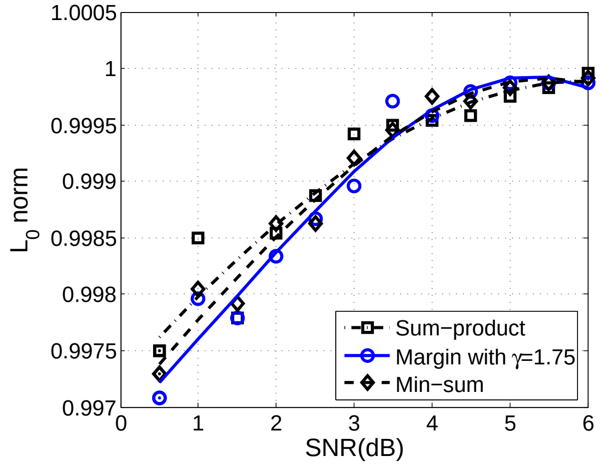
(a) Iteration 1: check to variable



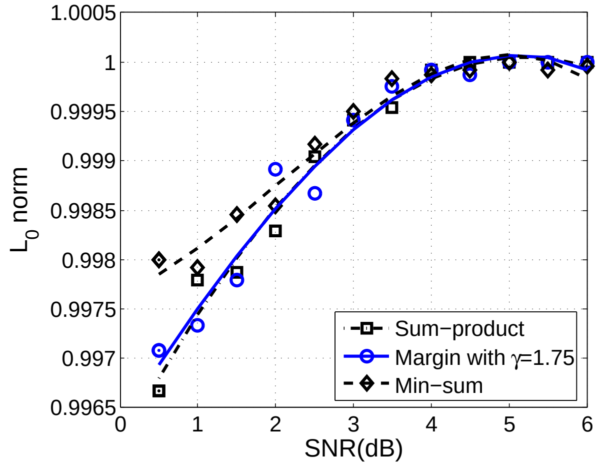
(b) Iteration 2: check to variable



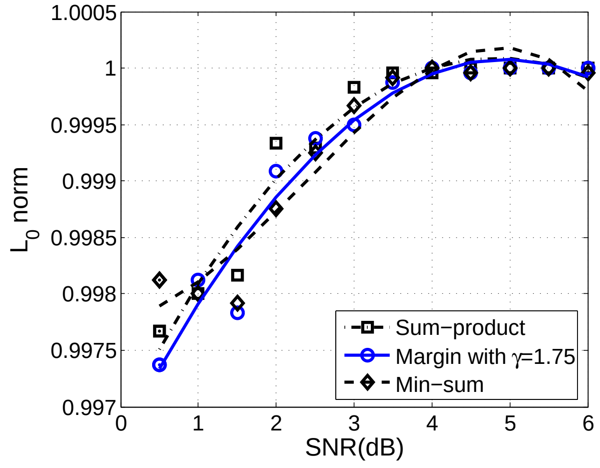
(c) Iteration 3: check to variable



(d) Iteration 1: variable to check



(e) Iteration 2: variable to check



(f) Iteration 3: variable to check

Figure 5.11: : L_0 norm comparison of sum-product, min-sum and margin propagation for iterations 1,2,3 for check to variable messages: (a),(b),(c) and variable to check messages: (d),(e),(f)

tion, we describe the current-mode circuits which has been used for implementing the check node module of the proposed MP-based LDPC decoder. A differential topology has been adopted to represent the positive and negative log-likelihood ratios and for canceling out any common-mode interference.

We used the check node c_1 in Fig.5.1 as an example to explain the implementation details. c_1 is connected to the variable nodes v_1, v_9, v_{17}, and v_{24}. To avoid long mathematical notations we will denote the message from c_1 to v_{24} ($L(\mathbf{c}_1 \rightarrow \mathbf{v}_{24})$) as z^{sp} and the messages $L(\mathbf{v}_1 \rightarrow \mathbf{c}_1)$ as L_1, $L(\mathbf{v}_9 \rightarrow \mathbf{c}_1)$ as L_2 and $L(\mathbf{v}_{17} \rightarrow \mathbf{c}_1)$ as L_3. The check function, according to equation (5.1) can then be written as:

$$z^{sp} = \log\left(\frac{e^{L_1} + e^{L_2} + e^{L_3} + e^{L_1+L_2+L_3}}{1 + e^{L_1+L_2} + e^{L_1+L_3} + e^{L_2+L_3}}\right). \tag{5.39}$$

If the log-likelihood ratios L_is are represented by their differential forms as $L_i = L_i^+ - L_i^-$, with $L_i^+, L_i^- > 0$, and $i = 1, \ldots, 3$, then equation (5.39) can be rewritten as:

$$z_{diff}^{sp} = \log\left(\frac{e^{L_1^+ + L_2^- + L_3^-} + e^{L_1^- + L_2^+ + L_3^-} + e^{L_1^- + L_2^- + L_3^+} + e^{L_1^+ + L_2^+ + L_3^+}}{e^{L_1^+ + L_2^+ + L_3^-} + e^{L_1^+ + L_2^- + L_3^+} + e^{L_1^- + L_2^+ + L_3^+} + e^{L_1^- + L_2^- + L_3^-}}\right). \tag{5.40}$$

Once the check function has been expressed in terms of log-sum-exp functions, we can apply the MP-based approximation according to the procedure described in section 5.2. The MP approximate check function (3.2) is written as

$$z_{diff}^{mp} = z^+ - z^-. \tag{5.41}$$

where z^+ and z^- are given by

$$z^+ = M(L_1^+ + L_2^- + L_3^-, L_1^- + L_2^+ + L_3^-, L_1^- + L_2^- + L_3^+, L_1^+ + L_2^+ + L_3^+, \gamma).$$
$$z^- = M(L_1^+ + L_2^+ + L_3^-, L_1^+ + L_2^- + L_3^+, L_1^- + L_2^+ + L_3^+, L_1^- + L_2^- + L_3^-, \gamma). \quad (5.42)$$

These equations are also applicable for computing messages $L(\mathbf{c}_1 \to \mathbf{v}_1)$, $L(\mathbf{c}_1 \to \mathbf{v}_9)$, and $L(\mathbf{c}_1 \to \mathbf{v}_{17})$.

Fig. 5.12 shows the architecture of the check node c_1 implementing equations (5.42). The differential messages generated by the variable nodes are selected and the permutations and summations are carried out according to the equation (5.42). The summation results are then processed by individual MP units as described in (5.42). The output produced by the respective MP units (z^+ and z^- in (5.42)) are then propagated to the neighboring variable nodes.

Fig. 5.13a shows the current-mode implementation of the equations (5.42). The currents $I_{L_i^{+(-)}}$ in Fig. 5.13a represent the differential variables $L_i^{+(-)}$ in equation (5.42), $i = 1, 2, 3$. The hyper-parameter γ in equation (5.42) is represented by the current I_γ. The circuit has two operational modes: (a) when the reset signal RST is set to logic low, transistor N_1 pulls the gate voltage of transistors N_2-N_6 low and the output current I_{z^+} is set to zero (implying no message is conveyed); (b) when the reset signal RST is set to logic high, the gate voltage of N_1 set to V_γ which determines the current I_γ. At equilibrium, the output current I_{z^+} is determined by the MP condition (3.1).

The PMOS diodes P_1 through P_4 ensures a unidirectional flow of current thus implementing the rectification operation in the equation (3.1). The subtraction operations in equation (3.1) are implemented by transistors N_2-N_5 and the summation constraint is implemented using the Kirchhoff's current summation at the node K. However, the diodes introduce a threshold voltage drop

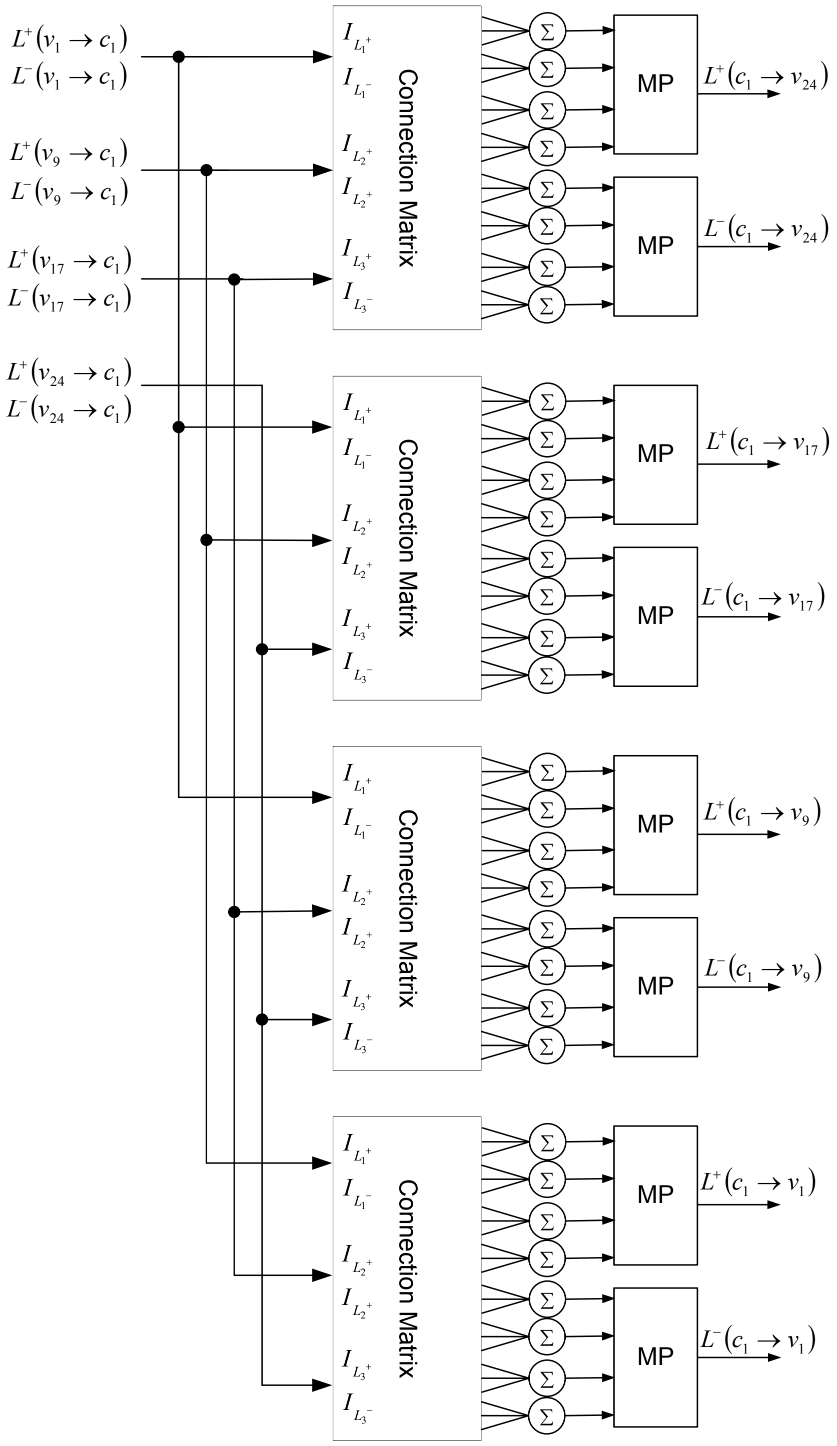


Figure 5.12: Architecture of the check node c_1 in Fig. 5.1

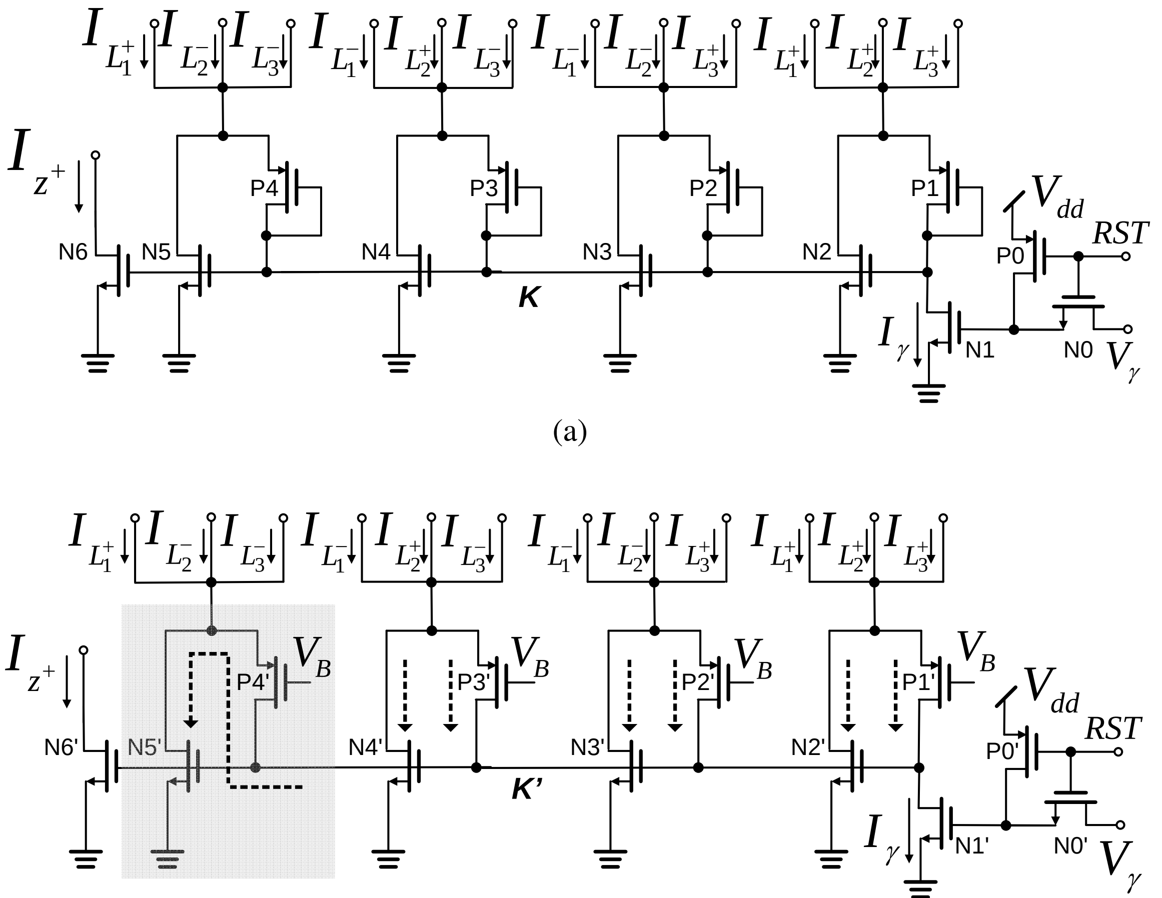


Figure 5.13: (a) Circuits for implementing the check node module using the MP algorithm:(a) design used in this paper; and (b) an alternate low-voltage circuit.

requiring a larger supply voltage ($\geq 2V_{th} + V_{dsat}$). An alternative low-voltage implementation is shown in Fig. 5.13b which consists of a cascode PMOS $P_1^{'}$ through $P_4^{'}$. The implementation eliminates the threshold voltage drop but potentially suffers from current leakage between neighboring stages as shown in Fig. 5.13b. This situation could occur when one of the branch currents is much larger than the other. In the proposed implementation, we have chosen the circuit shown in Fig. 5.13a. Since the operation of the circuit is based on the Kirchhoff's current law, it is functional irrespective of the biasing condition of the transistors. Consequently, the magnitude of current can vary in a wide range from pA to μA, which only effects the operational speed of the circuit. However, the matching between the transistors N_2-N_5 is important because it ensures that the constraint in equation (3.1) is accurately satisfied.

5.4.2 Implementation of the variable node

We will use the variable node v_1 to illustrate the basic operation as described by equation (5.2). The architecture of the variable node is shown in Fig. 5.14 and the circuit implementation for computing the message propagated from variable node v_1 to check node c_{20} is shown in Fig. 5.15. The currents $I_{v_1}^{+}$ represent the positive part of differential message $L(v_1)$ ($L(v_1) = L^{+}(v_1) - L^{-}(v_1)$). Similarly, $I_{c_1 \to v_1}^{+}$ and $I_{c_8 \to v_1}^{+}$ represent the $L^{+}(c_1 \to v_1)$ and $L^{+}(c_8 \to v_1)$. These currents are summed at node B and mirrored through transistors P_0 and P_1. Similarly, the current through P_2 equals the sum of all the negative portion of the differential message. For brevity, we denote the two summed currents as I^+ and I^- as shown in Fig. 5.15, where

$$\begin{aligned} I^+ &\Rightarrow L^+(\mathbf{v}_i) + \sum_{\mathbf{c}_k \in \mathbf{V}_{i\sim j}} L^+(\mathbf{c}_k \to \mathbf{v}_i), \\ I^- &\Rightarrow L^-(\mathbf{v}_i) + \sum_{\mathbf{c}_k \in \mathbf{V}_{i\sim j}} L^-(\mathbf{c}_k \to \mathbf{v}_i). \end{aligned} \tag{5.43}$$

The two currents I^+ and I^- are compared at node A and when I^+ is greater than I^-, the current through P_5 equals to the positive part of the differential message propagated from the variable node to the check node. A similar circuit computes the negative part of the differential message. Thus, we realize the differential form of (5.2) as

$$\begin{aligned} L^+(\mathbf{v}_i \to \mathbf{c}_j) &\Leftarrow \begin{cases} I^+ - I^-, & \text{if } I^+ - I^- > 0. \\ 0, & \text{otherwise.} \end{cases} \\ L^-(\mathbf{v}_i \to \mathbf{c}_j) &\Leftarrow \begin{cases} 0, & \text{if } I^+ - I^- > 0. \\ I^+ - I^-, & \text{otherwise.} \end{cases} \end{aligned} \tag{5.44}$$

Note, the transistors P_3 and P_4 in Fig. 5.15 act as current limiters where the common-mode current and hence the total power dissipation of the decoder can be limited using the voltage V_B.

5.5 Measurement Results and Discussions

Fig. 5.16 shows a system level schematic of the (32,8) LDPC decoder corresponding to the Tanner-graph shown in Fig. 5.1. Besides the variable node and the check node modules, the decoder also integrates the following modules to facilitate output buffering and testing through an external digital interface (FPGA in this paper):

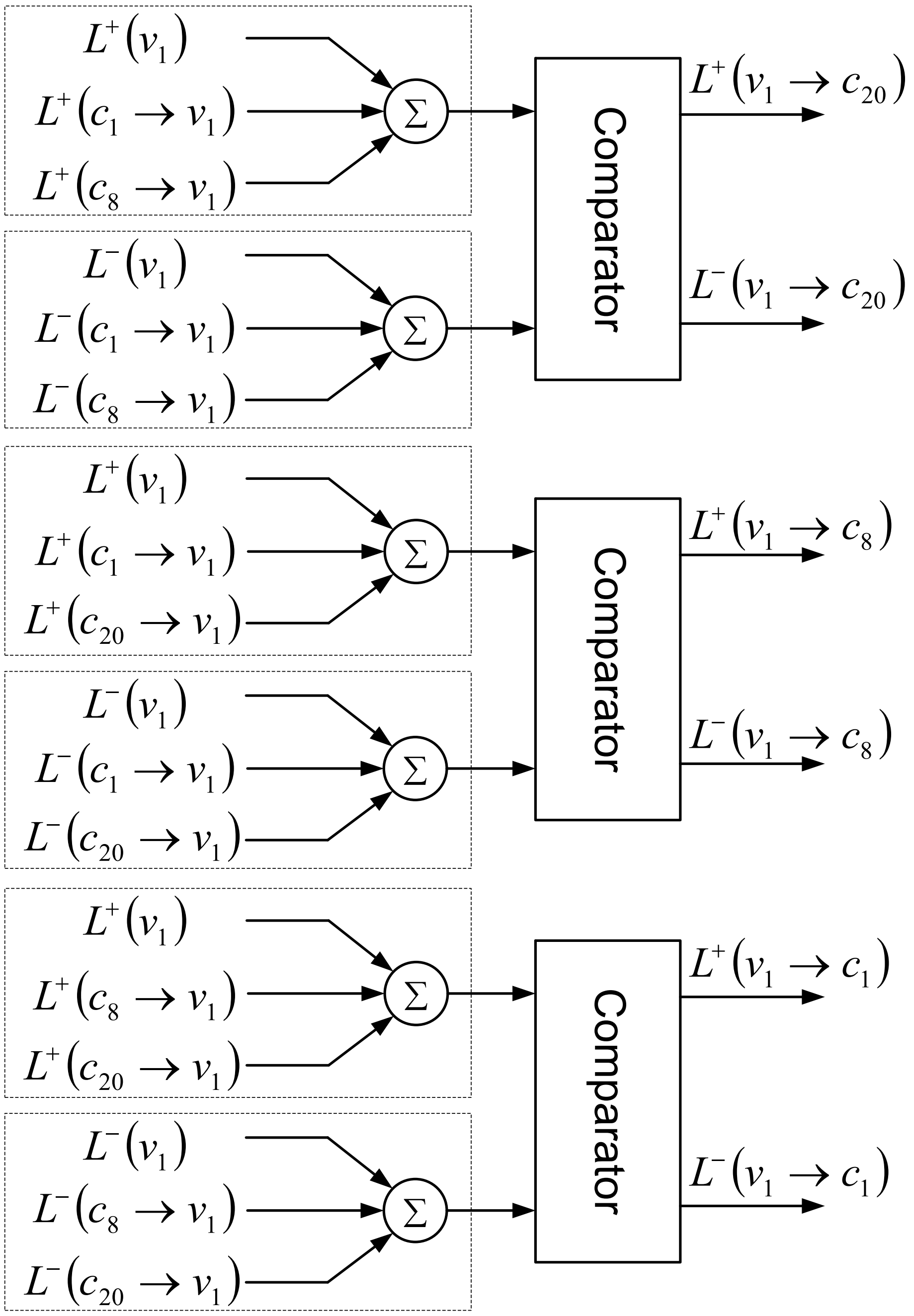


Figure 5.14: Architecture of variable node v_1 in Fig. 5.1

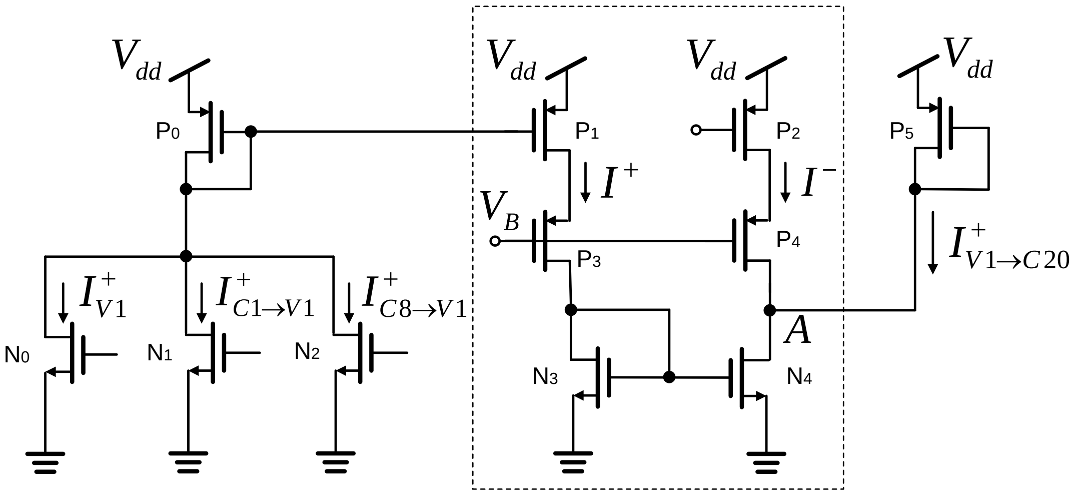


Figure 5.15: Circuit to implement variable node

(a) **A 6-bit current-mode digital-to-analog converter (DAC)**, used to digitally programming initial messages (currents) in equation (5.43). Its MSB of the DAC is denoted by d_1 whereas the LSB is denoted by d_6. The architecture of the current DAC (shown in Fig. 5.16 is based on the popular resistive divider [95] where the current through each branch is twice as large as its neighbor. The output as determined by the bits d_1–d_6 is a binary weighted sum of currents which are mirrored into each of the variable node modules.

(b) **Output comparators**, which are used to compute the decoded bits by comparing the differential variable node currents when the decoder has reached equilibrium. For example, if I_V^+ is greater than I_V^-, the output of the comparator Q is logic "high" and indicative of the bit "0" being decoded. Note that the comparator needs to be reset before each comparison is made.

(c) **Digital sample-and-hold input buffer** is a shift register chain which converts the serial input D_{in} to the decoder IC into parallel bit slices that are processed by the DAC module. As shown in Fig. 5.16, there are two sets of non-overlapping control signals for the shift register chain: $\Phi_1/\overline{\Phi_1}$ and $\Phi_2/\overline{\Phi_2}$. Φ_1 and Φ_2. In one period, when (Φ_1 = 1 and Φ_2 = 0), the previous stage data d_{i-1} is sampled; and when (Φ_1 = 1 and Φ_2 = 0), the sampled data is held during while the DAC module processes the latched bits.

(d) **Digital sample-and-hold output buffer** is also a shift register chain which sample the parallel decoded bits (from the comparator modules) and convert them into a bit-serial format Q_{out}.

The microphotograph of a prototype MP-based LDPC decoder fabricated in a 0.5μm standard CMOS process is shown in Fig. 6.20. Table 6.3 summarizes the main specifications of the chip.

Table 5.2: Specifications of the MP-based analog LDPC decoder

Fabrication Process	Standard CMOS $0.5\mu m$
Die Size	$3000\mu m \times 3000\mu m$
Number of Transistors	11424 (6336 PMOS, 5088 NMOS)
Number of Variable Nodes	32
Number of Check Nodes	24

The measurement setup used to evaluate the fabricated chip is shown in Fig. 5.18. The decoder chip is hosted on a daughter board which is then mounted on a test station mother board. A second mountable daughter board hosts a field-programmable gate array (FPGA) which is responsible for clock generation and data synchronization. A sample timing diagram for all the digital signals used by the LDPC chip (shown in Fig. 5.16) and generated by the FPGA are shown in Fig. 5.19. The FPGA selectively programs the on-chip DACs to emulate data received over an additive white Gaussian noise (AWGN) channel after which the FPGA enables the analog decoding core. The output of the decoder is latched on the comparator after a pre-determined *decoding time* and the latched bit-stream is serially retrieved by the FPGA. As shown in Fig. 5.18, the experimental setup is also controlled through a Visual Studio interface on a PC. The C script emulates the AWGN

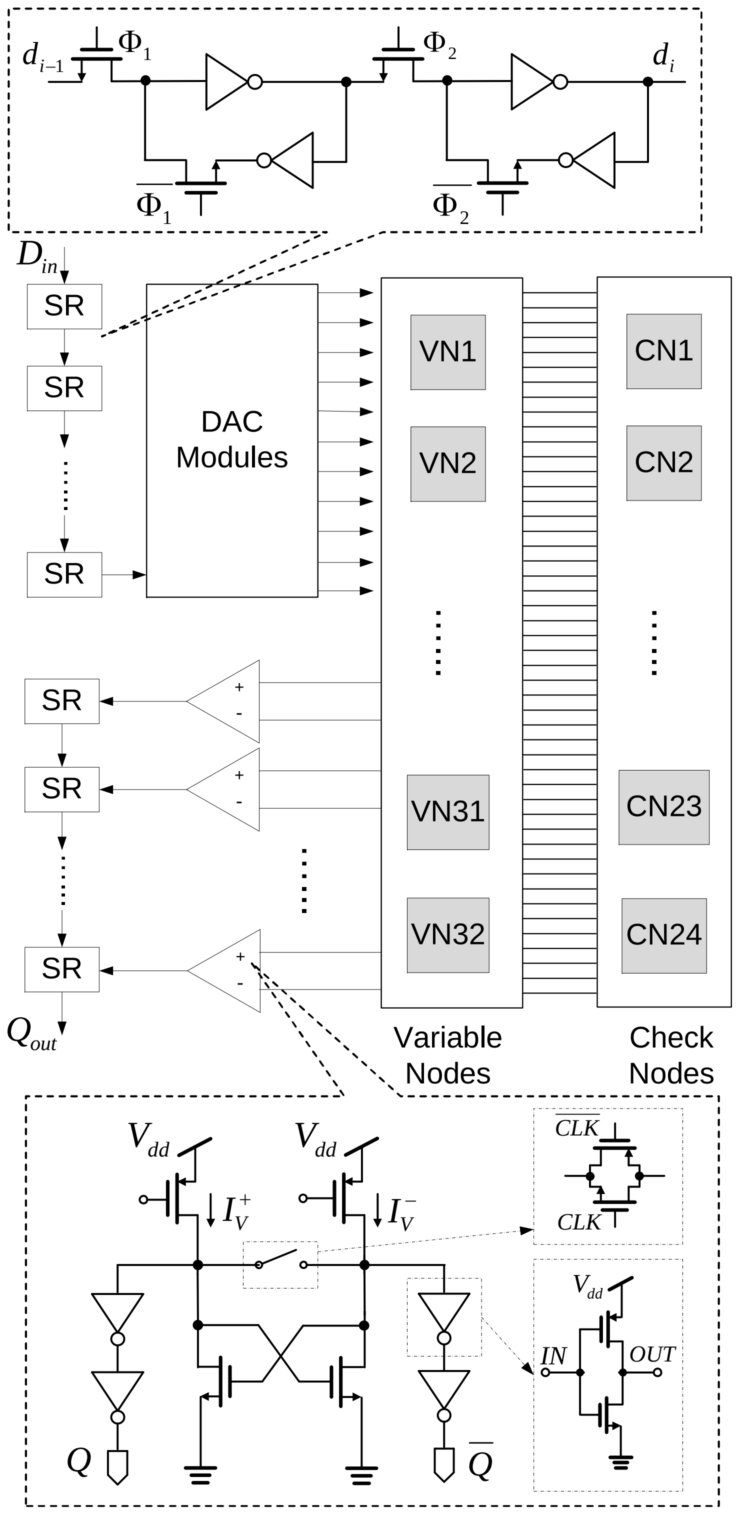


Figure 5.16: System level architecture of the (32,8) LDPC decoder

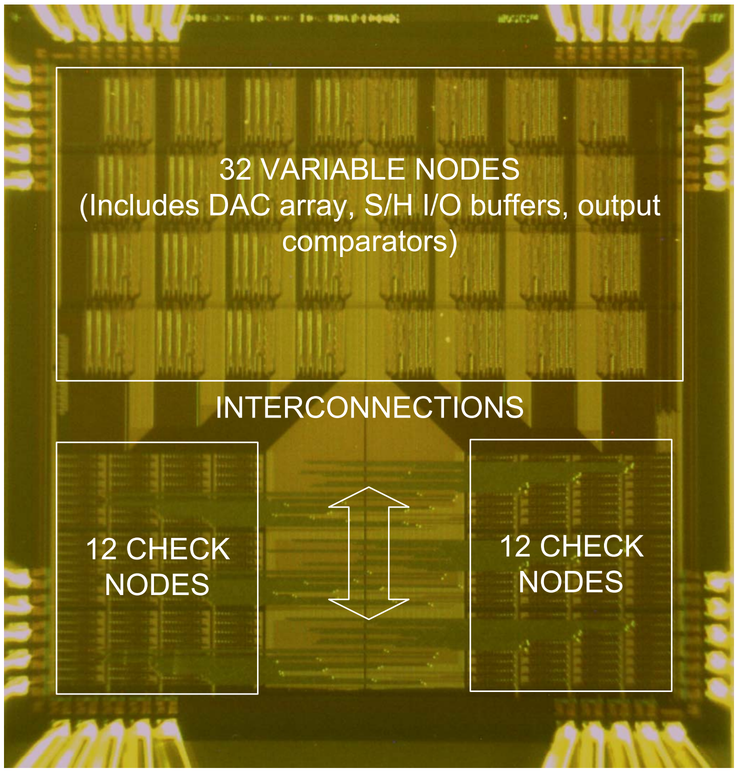


Figure 5.17: Die microphotograph of the chip

channel and generates 6-bit signal values at different SNR levels. These bit patterns are then stored on the SDRAM of the FPGA which uses the values to perform real-time decoding. After each noisy transmission, decoded bits are retrieved from the LDPC chip and stored on the FPGA's SDRAM. At the end of each Monte-carlo run, the Visual Studio interface transfers the data logged on the SDRAM and computes the overall BER. The MATLAB interface is also used to adapt the bias voltages of the decoder through a National Instruments data acquisition card.

Fig. 5.20 compares the BER performance of a software implementation of the MP-based decoder and the measured results obtained from the fabricated prototype. For this experiment the decoding throughput was set to 320 Kbps and the hyper-parameter γ was optimally chosen based on iterative experiments as described later. At low SNR, the hardware implementation outperforms its software counterpart which could be attributed to the continuous-time dynamics of the analog

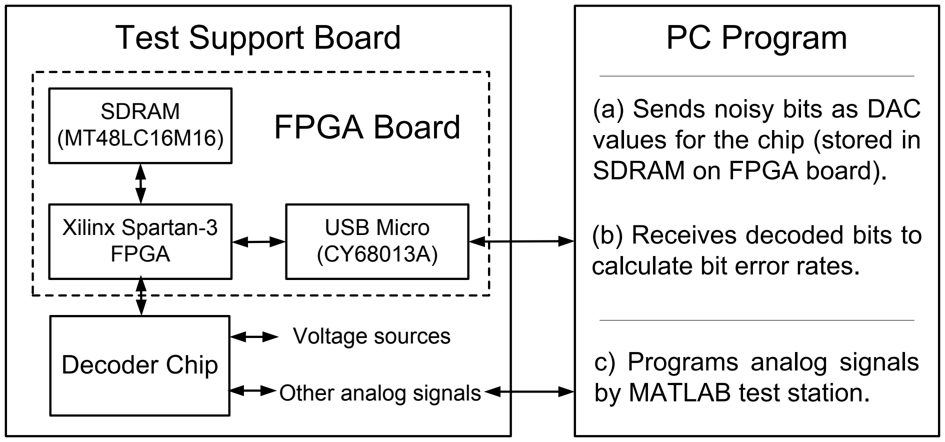

Test Support Board
FPGA Board
SDRAM (MT48LC16M16)
Xilinx Spartan-3 FPGA
USB Micro (CY68013A)
Decoder Chip
Voltage sources
Other analog signals
PC Program
(a) Sends noisy bits as DAC values for the chip (stored in SDRAM on FPGA board).
(b) Receives decoded bits to calculate bit error rates.
c) Programs analog signals by MATLAB test station.

Figure 5.18: Experimental setup of the chip

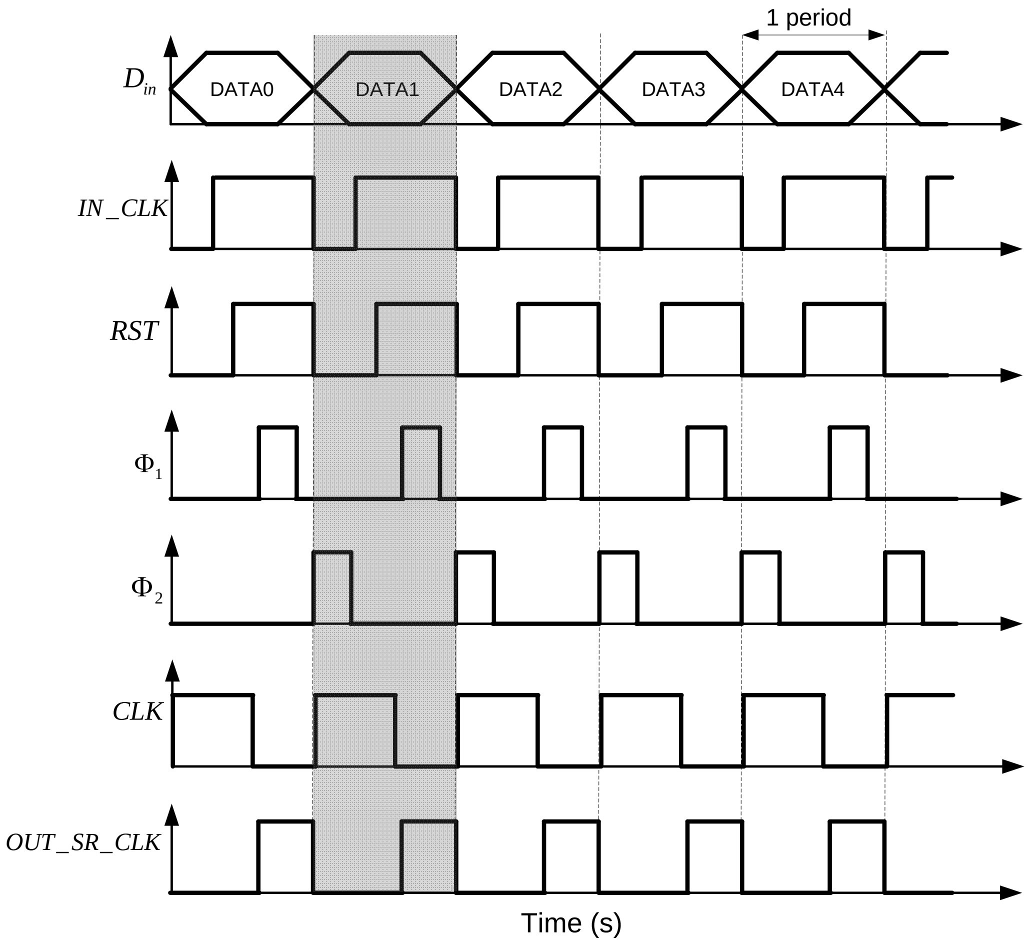


Figure 5.19: Timing diagram of digital signals used to control the operation of the LDPC decoder chip

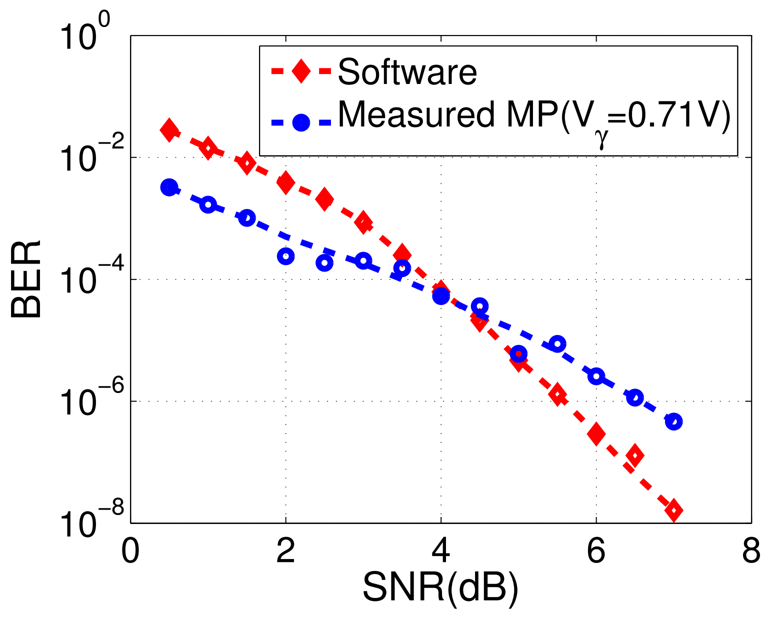


Figure 5.20: Comparison of BER of the MP-based LDPC decoder using software simulation and measurements from the fabricated prototypes.

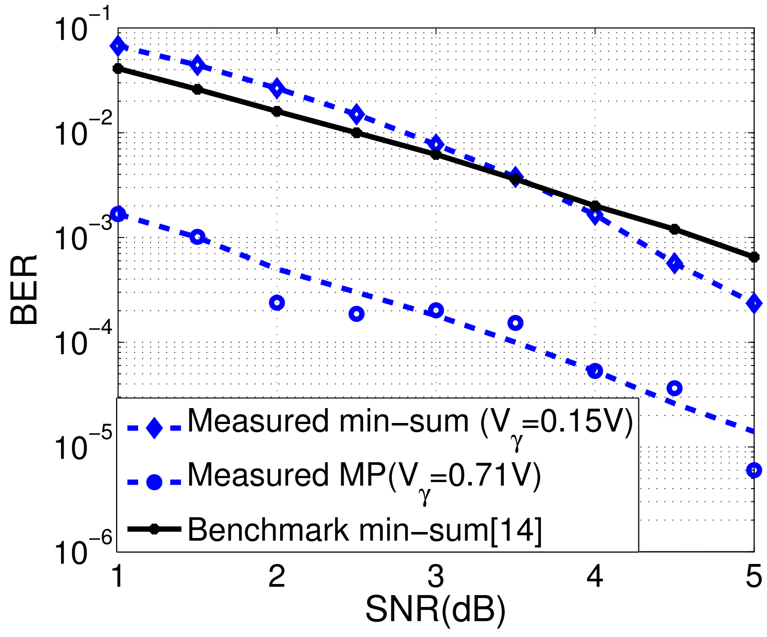


Figure 5.21: Comparison of BERs of the measured MP-based LDPC decoder and a benchmark min-sum LDPC decoder.

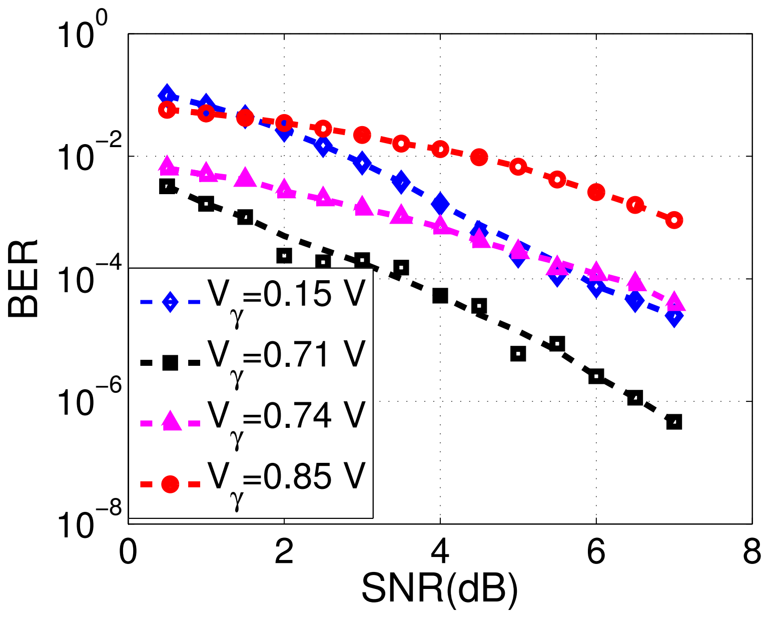


Figure 5.22: Comparison of measured BER performances for different values of the hyper-parameter γ.

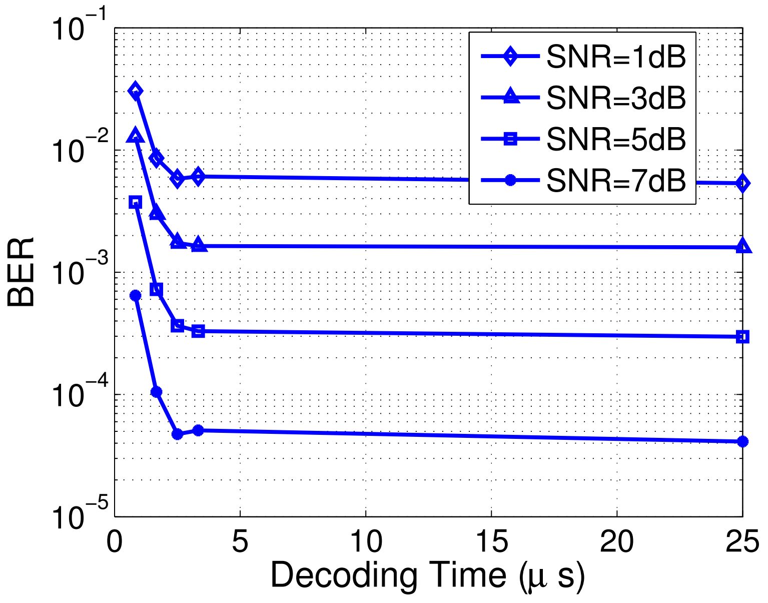


Figure 5.23: Measured BER performances of the MP-based LDPC decoder ($V_\gamma = 0.74V$) for different decoding times (throughput)

decoder. However, at high SNR the software implementation outperforms its hardware counterpart which could be attributed to the limited dynamic range of the programming DACs and due to the offset and gain errors introduced by the current mirrors.

Fig. 5.21 shows the measured BER curves which are obtained under the conditions: (a) when the system is configured to operate as a min-sum decoder ($\gamma \approx 0$) [45] and the decoding throughput is set to 320 Kbps; and (b) when the MP-based decoder is configured with an optimal setting of the hyper-parameter $\gamma > 0$. For comparison we have included BER results reported for a state-of-the-art analog min-sum decoder [77]. The results in Fig. 5.21 show that for $\gamma = 0$, the performance of MP-based min-sum LDPC decoder is inferior to the benchmark min-sum decoder [77] at low SNR, however, at high SNR ($>$ 3.5 dB) the performance of MP min-sum decoder outperforms the benchmark. When $\gamma > 0$, the results show that the MP-based LDPC decoder outperforms the benchmark min-sum decoder by more than $3dB$.

Fig. 5.22 shows the measured BER curves for different values of the hyper-parameter $V\gamma$. Again, for this experiment the decoding throughput is set to 320 Kbps. As $V\gamma$ increases (γ increases), it can be seen that the BER of the MP-based LDPC decoder first improves and then degrades, which is consistent with the BER-SNR-γ trade-off previously demonstrated only using simulation results (shown in Fig. 5.8b).

Fig. 5.23 shows the measured BER performance (under different SNRs) when the decoding time (inverse of decoding speed) is varied. For this experiment, $V\gamma$ is held constant at 0.74V (correspondent to one of the four curves shown in Fig. 5.22). It is seen that as the decoding time is increased, the BER performance improves, which is consistent with the response of previously reported analog decoders [77]. When the decoding time is reduced to less than 2.5 μs (data throughput 12.8 Mb/s), the BER performance starts to decreases rapidly. When decoding time is as low as 833 ns (data throughput 38.4 Mb/s), the BER (measured at SNR 7dB) could be 13 times higher than the BER measured at data throughput of 12.8 Mb/s.

Table 5.3 compares the measured specification of the fabricated MP-based LDPC decoder to the specifications of different digital and analog decoders reported in literature.

The comparison shows that the MP-based LDPC decoder has the second highest throughput amongst the reported analog decoders. However, the turbo decoder with the highest throughput [96] has an energy efficiency much lower than the MP-based design. Moreover, the turbo decoder has a longer codeword length than ours. The benchmark min-sum LDPC decoder [77] has the same codelength as the proposed MP-based LDPC decoder. However, the former measured the energy efficiency based on the power consumption for the whole chip, while the latter based on the core, which makes the comparison difficult. However, it can be seen that the throughput of the MP-based LDPC decoder is more than twice that of the benchmark min-sum LDPC decoder. In terms of energy efficiency, the MP-based LDPC decoder has the second highest energy efficiency. Whereas the (8,4) Hamming Trellis graph decoder, which has the highest energy efficiency, has a much lower throughput than this implementation. In terms of the silicon area, the proposed implementation achieves integration density comparable to that of [77], considering the differences in the respective technology feature size.

The table of comparison also shows that state-of-the-art digital decoders can also enjoy high energy-efficiencies as the analog decoders. This is because these implementations exploit highly parallel architecture [72, 73], early termination techniques [97], post-processing methods [98] and aggressive feature and voltage scaling. However, it should be noted that unlike analog decoders, digital decoders require an analog-to-digital converter to digitize the analog channel information. For instance, the digital decoder reported in [98] requires 4 to 6-bit digital inputs and the energy efficiency metric (pJ/bit) reported in table 5.3 does not incorporate the power dissipation of the ADC.

Table 5.3: CMOS Digital (top 4) and Analog (bottom 6) Decoders

Author	Code	CMOS Technology	Core Area (mm^2)	Power (mW)	Throu-ghput (Mb/s)	Energy Efficiency (nJ/b)
Digital						
Blanksby et al. [73]	R=$\frac{1}{2}$, N=1024, LDPC	0.16 μm, 1.5 V	52.5	690	500	1.4
Bickerstaff et al. [72]	R=$\frac{1}{2}:\frac{14}{16}:\frac{1}{16}$ N=2048,LDPC	0.18 μm 1.8 V	14.3	787	320	2.46
Darabiha et al. [97]	R=$\frac{11}{15}$ N=660 LDPC	0.13 μm, 1.2 V 0.6 V	7.3	518@4dB 398@5.5dB	2440 480	0.156@4dB 0.12@5.5dB
Zhang et al. [98]	R=0.84 N=2048 RS-LDPC	65 nm 0.7 V 1.2 V	6.67	144 2800	6670 47700	0.0215 0.0587
Analog						
Gaudet et al. [96]	R=$\frac{1}{3}$,N=48, Turbo	0.35 μm, 3.3 V	1.32	185	13.3	13.9
Winstead et al. [99]	(8,4) Hamming Tail-biting	0.5 μm, 3.3 V	0.083	1 (core)	1	1
Amat et al.[100]	R=$\frac{1}{3}$,N=132, Turbo	0.35 μm, 3.3 V	4.1	6.8 (core)	2	3.4
Winstead et al. [78]	(8,4) Hamming Trellis Graph Factor Graph	0.18 μm, 1.8 V	0.002 0.02	0.15 0.807	3.7 3.7	0.04 0.22
Hemati et al. [77]	(32,8) LDPC	0.18 μm, 1.8 V	0.57	5 (chip)	6	0.83
Gu et al. this work	(32,8) LDPC	0.5 μm, 3.3 V	5.4	1.254 ($V\gamma$=0.85V) 1.683 ($V\gamma$=0.74V) 1.914 ($V\gamma$=0.71V) 7.755 ($V\gamma$=0.15V) (core)	12.8	0.098 0.1315 0.1495 0.6059

Chapter 6

An analog SVM implemented on MP-based PWL circuits

6.1 History of SVM

SVM theory provides a principled approach for designing classifiers (binary and multi-class) by having a strong foundation in statistical learning theory. Due to their excellent generalization ability and modest requirements on the size of training data, SVMs have been successfully applied to numerous real-time recognition tasks [101]. Even though several digital implementations of SVMs have been reported in literature [102], they have been mapped onto analog VLSI using translinear CMOS circuits [103, 104] only recently. Here, we will re-formulate SVMs to operate in a log-likelihood domain, and apply MP-based transforms to map the architecture onto CMOS circuits. The result is a general-purpose micro-power SVM hardware that can be used in applications ranging from communications to biometrics.

6.2 An MP-based SVM

In its general setting, a binary SVM computes a decision score $F \in \mathbb{R}$ for every feature vector presented $\overrightarrow{z} \in \mathbb{R}^{\mathbb{N}}$ and is given by:

$$F = \sum_{i=1}^{N} \alpha_i K(\overrightarrow{x_i}, \overrightarrow{z}) + b \tag{6.1}$$

$K : \mathbb{R}^{\mathbb{N}} \times \mathbb{R}^{\mathbb{N}} \rightarrow \mathbb{R}$ denotes a kernel function computed between an input vector $\overrightarrow{z}$ and an array of templates (support vectors) $\overrightarrow{x_i} \in \mathbb{R}^{\mathbb{N}}$. The parameters $\alpha_i \in \mathbb{R}$ and $b \in \mathbb{R}$ are obtained through a supervised training procedure [105]. Even though several choices of kernel functions exist, a second-order polynomial kernel given by $K(\overrightarrow{x_i}, \overrightarrow{z}) = [(\overrightarrow{x_i} \cdot \overrightarrow{z}) + \beta]^2$ has been chosen. All the parameters of the SVM are represented in a differential form as $\alpha_i = \alpha_i^+ - \alpha_i^-$ and $b = b^+ - b^-$, which leads to a differential representation of equation (6.1) as:

$$F = F^+ - F^-, \tag{6.2}$$

where $F^+ = \sum_i \alpha_i^+ K(\overrightarrow{x_i}, \overrightarrow{z}) + b^+$ and $F^- = \sum_i \alpha_i^- K(\overrightarrow{x_i}, \overrightarrow{z}) + b^-$.

The inner product computation of $\overrightarrow{x_i}$ and $\overrightarrow{z}$ can also be achieved by turning them into differential forms:

$$\begin{aligned}\overrightarrow{x_i}^T \cdot \overrightarrow{z} &= \sum_{j=1}^{N} (x_{ij}^+ - x_{ij}^-)(z_j^+ - z_j^-) \\ &= \sum_{j=1}^{N} (x_{ij}^+ z_j^+ + x_{ij}^- z_j^-) - \sum_{j=1}^{N} (x_{ij}^+ z_j^- + x_{ij}^- z_j^+)\end{aligned} \tag{6.3}$$

Here x_{ij} and z_j are elements of vector $\overrightarrow{x_i}$ and $\overrightarrow{z}$. (6.3) can be approximated with MP units expressed by:

$$
\begin{aligned}
z^{+} \approx \mathcal{M}(&\log x_{i1}^{+} + \log z_{1}^{+}, \ldots, \log x_{iN}^{+} + \log z_{N}^{+},\\
&\log x_{i1}^{-} + \log z_{1}^{-}, \ldots, \log x_{iN}^{-} + \log z_{N}^{-}, \gamma)\\
z^{-} \approx \mathcal{M}(&\log x_{i1}^{+} + \log z_{1}^{-}, \ldots, \log x_{iN}^{+} + \log z_{N}^{-},\\
&\log x_{i1}^{-} + \log z_{1}^{+}, \ldots, \log x_{iN}^{-} + \log z_{N}^{+}, \gamma)
\end{aligned}
\qquad (6.4)
$$

For a classification task, only the sign of F is important, and using the monotonic property of the $log(.)$ implies $sign(F) = sign(log(F^{+}) - log(F^{-}))$. Therefore, the decision generated by the SVM can be expressed as

$$
sign(F) = sign(z^{+} - z^{-}). \qquad (6.5)
$$

The system architecture and circuit diagram of MP-based SVM are shown in Fig. 6.1.

An MP-based SVM architecture can be partitioned into three major sub-systems as shown in figure 6.1. It consists of a log-MAP matrix-vector multiplier that computes compressive inner-product kernels based on margin propagation. The outputs of the matrix-vector multiplier are scaled in log domain, which is equivalent to a cubic transformation of an inner-product kernel. The scaled factors are then presented to an output normalization stage that computes the decision function according to equation (6.5).

A current-mode circuit for implementing a log inner product computation $log(x^{T}y)$ based on MP circuits is also shown in figure 6.1. At the core of the network is the basic MP cell shown in Fig. 4.2. Currents representing the logarithm of vector elements are summed according to Kirchhoff's current law. A constant current sink with magnitude γ constraints the total current at the gate of

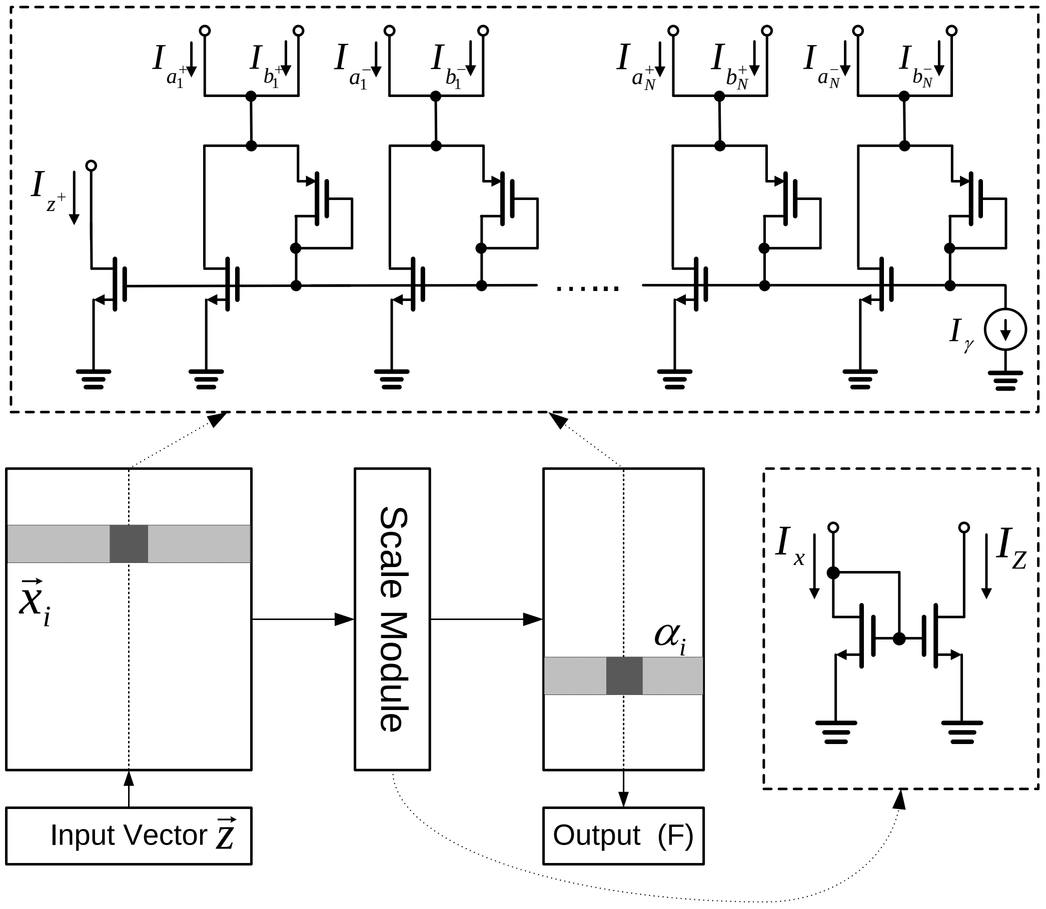


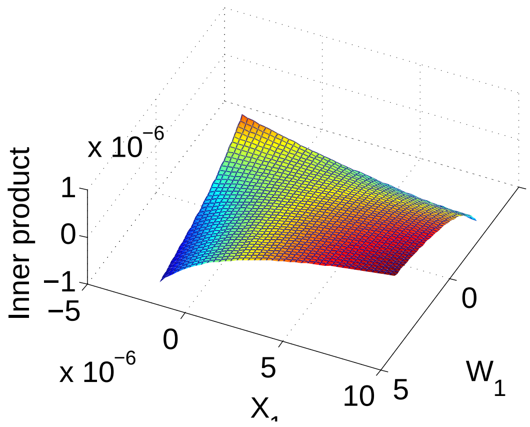
Figure 6.1: The circuit diagram of the MP-based SVM

output transistor. Thus the reverse water-filling criterion is implemented. For the SVM sub-system in Fig. 6.1, the circuit is replicated for each template (support vector). The output current from the first stage is mirrored and scaled to an outer normalization stage that implements another MP normalization procedure. Consequently, the outer normalization stage is also implemented using the reverse water-filling circuit in Fig. 4.2.

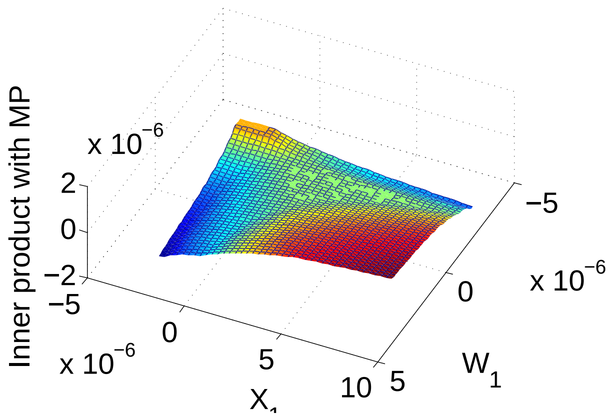
The SVM parameters α_i and x_i are stored as currents on floating gate transistors, illustrated in the highlighted box of figure 4.2. Each of the storage cells are individually addressable via a serial shift register chain. Programming is achieved by using hot-electron injection in conjunction will global tunneling [106]. There are 14 reverse water-filling cells per support vector in the log-MAP matrix-vector multiplier. The number of cells in the output normalization sub-system is proportional to the total number of support vectors. Since the positive and negative class labels must be split to maintain positive current flow during analog computation, a fully-differential implementation of the output stage is required to compute the decision score.

6.3 Simulation result

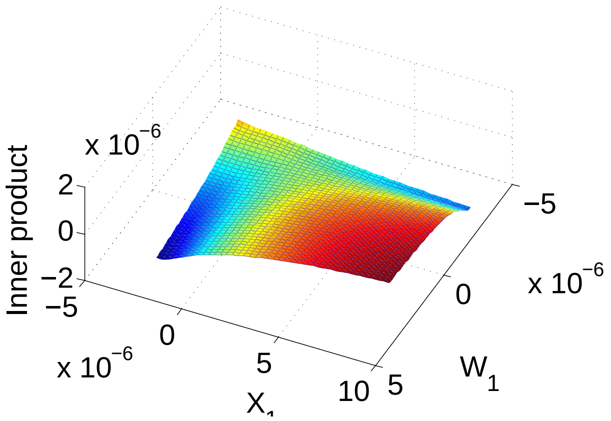
Fig. 6.2 shows the PWL approximation effect of MP-based algorithm on inner product, expressed by equation (4.34), with $N = 2$. Fig. 6.2a displays the result computed by “log-sum-exp” function. Fig. 6.2b displays the result computed by MP-based function simulated in Matlab. Fig. 6.2c displays the result computed by MP-based function simulated in Cadence.



(a)



(b)



(c)

Figure 6.2: Multiplication computed by (a) "log-sum-exp" function (b) MP-based function with $\gamma = 1.2$ (in Matlab) (c) MP-based function with $\gamma = 1.2$ (in Cadence)

6.4 Circuit Implementation

The prototype fabricated in a 0.5 μm CMOS process has integrated an array of floating gate transistors that serve as storage for up to 740 SVM parameters as well as novel circuits that have been designed for interfacing with an external digital processor. These include a novel current-input current-output logarithmic amplifier circuit that can achieve a dynamic range of 120dB while consuming nanowatts of power and a novel varactor based temperature compensated floating-gate memory that demonstrates a superior programming range than other competitors.

6.4.1 floating-gate memory cells

The prototype uses floating-gate memory array to store SVM parameters. The system architecture and the basic circuit of the floating-gate memory cells is shown in Fig. 6.3. The prototype has embedded floating-gate memory array, temperature compensation circuitry, logarithmic amplifier circuitry (translinear circuitry). It also sets aside connection pin to Keithley for current calibration and measurement as well as the pin for floating-gate programming. The memory array is composed of 740 floating-gate memory cells, which are distributed into 18 rows and 114 columns. Each cell is addressable by their address switches, Row_select and Column_select1-3. These switches are set by programming shift registers. As can be seen from Fig. 6.3, the system is also transferable between two states by setting the signal RUN. Once RUN is set to logic high, the parameters stored in memory cells in terms of current are flowing into the MP-based computational circuit and the computation is carried out. However, once RUN is set to low, the stored current is led into Keithley so that calibration and measurement are executed.

The current through P_2 represents the support vectors. As is seen from Fig. 6.3, P_2 and P_1 are a pair of floating gate transistors. Hence the current through P_2 is tunable through charge injection and tunneling. During current injection, the memory cell to be programmed is selected by the address switches. A negative voltage is applied to Program_pin so that charge can be injected into the gate of P_1 and P_2. During charge tunneling, a high voltage is applied to V_{tunnel}. Vc is a voltage applied to the floating-gate control capacitor Cc, which can be used to tune the gate voltage of P_1 and P_2 and hence control the current flow through them. Vx is the voltage applied to the varactor C_v, which is used to tune the value of C_v. This property is used in temperature compensation to protect the memory array from the attack of temperature variation. We will talk about this feature into detail in later sections. The current through P_3 represents the input vectors. It is controlled by Vdd_translinear and Vg_translinear, which are transmitted from reference circuitry, i.e. a current-input current-output logarithmic amplifier circuitry.

6.4.2 Current-input current-output logarithmic amplifier circuit

Logarithmic amplifiers are used for generating output signals that are proportional to the logarithm of the input [107]. Due to the compressive nature of the logarithm function, such an amplifier is useful for processing wide dynamic range signals like sensory (auditory or visual) signals or multi-path signals in radio-frequency and radar applications.

The most popular technique for implementing logarithmic amplifiers is a transimpedance based approach that exploits the exponential dependence between the current and the voltage across a p-n junction diode, a bipolar transistor [14] or a MOSFET in sub-threshold region [108, 109]. Based on this approach, amplifiers with input dynamic range greater than 100dB have been reported. However, there are three disadvantages of using a transimpedance based approach: (a) the output of a transimpedance logarithmic amplifier is a voltage signal, which implies that the output dynamic range is limited by the supply voltage; (b) due to the dependence on the current-to-voltage relationship, transimpedance logarithmic amplifiers are sensitive to temperature variations and therefore requires additional compensation circuitry; and (c) for current-mode analog VLSI computation, the use of transimpedance logarithmic amplifiers requires a linear transconductor stage to convert the output voltage into currents.

In our study, we propose the design of a current-input, current-output logarithmic amplifier, which by construction in insensitive to variations in temperature. At the core of the proposed design

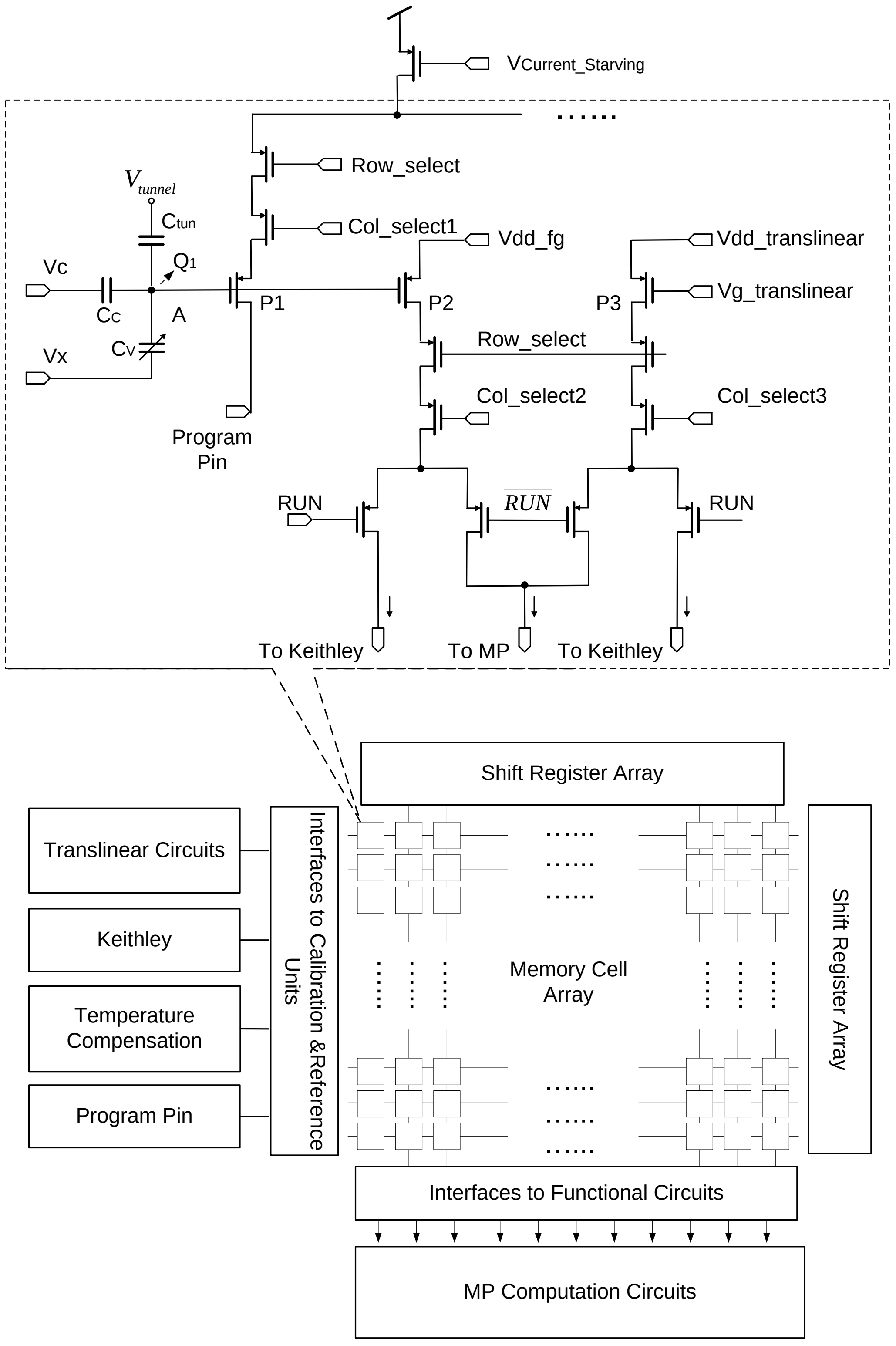


Figure 6.3: The system architecture and basic circuit of floating-gate memory cells.

is a Translinear Ohm's law principle which exploits floating-voltage sources and linear/non-linear resistive elements embedded within a translinear loop. By exploing multiple translinear networks, temperature compensation is achieved through a resistive cancellation technique. We verify the functionality of the circuit using measured results from fabricated prototypes and justify the results using mathematical models and analysis. We also extend the design of the logarithmic amplifier to be fully programmable and digitally addressable, where all the current references have been made programmable using analog floating-gates and the input to the amplifier is made digitally programmable using a standard ladder-based digital-to-analog converter DAC.

6.4.2.1 Translinear Ohm's Law based Logarithm Computation

The concept of translinear Ohm's law is an extension of the celebrated translinear principle [14] which exploits the exponential relationship between voltages and currents in certain devices (diodes, BJTs and sub-threshold MOSFETs). Translinear principle has been successfully used for implementing linear analog computation like matrix-vector multiplication [17] and for implementing non-linear analog computation like quadratic kernels [61]). Translinear Ohm's law is first explained in this section using an example circuit shown in Fig. 6.4 which consists of several diodes D_{1-4} acting as translinear elements connected to a floating-voltage source ΔV and a memoryless, non-linear circuit element N. The non-linear circuit element N is characterized by a function $f(.)$ which models relationship between the current flowing through N (I_N) and the voltage across N (V_N) according to

$$I_N = f(V_N). \tag{6.6}$$

If the voltage drop across each diode in Fig. 6.4 is denoted by $V_{1–4}$ then,

$$V_1 + V_2 + \Delta V = V_3 + V_4 + V_N, \tag{6.7}$$

which after using the translinear diode equation of $I_{1\text{-}4} = I_S \exp\left(V_{1\text{-}4}/U_T\right)$ leads to,

$$I_1 \cdot I_2 \cdot \exp\left(\frac{\Delta V}{U_T}\right) = I_3 \cdot I_4 \cdot \exp\left(\frac{V_N}{U_T}\right), \tag{6.8}$$

and hence,

$$\begin{aligned} V_N &= \Delta V + U_T \ln\left(\frac{I_1 \cdot I_2}{I_3 \cdot I_4}\right) && (6.9) \\ I_N &= f\left(\Delta V + U_T \ln\left(\frac{I_1 \cdot I_2}{I_3 \cdot I_4}\right)\right) && (6.10) \end{aligned}$$

U_T in equations 6.8 and 6.10 refers to the thermal voltage which is a linear function of absolute temperature and is approximately equal to 26mV at room temperature (25^{o}C).

If currents I_{1-4} satisfy the relation $I_1 \cdot I_2 = I_3 \cdot I_4$ then,

$$I_{\text{N}} = f\left(\Delta V\right). \tag{6.11}$$

If N is a resistor with resistance R, then $f\left(\Delta V\right) = \Delta V/R$ which when inserted in equation (6.11) leads to an equivalent Ohm's law (Note that the current I_{N} is not directly drawn from V_{N}). If N is a general type of resistor (linear and non-linear) with $f(0) = 0$, then f can be approximated using first-order and second-order Taylor series terms as $f(\Delta V) \approx f'(0)\Delta V + f''(0)/2(\Delta V)^2$.

Inserting the approximation in equation (6.11) leads to

$$I_{\mathrm{N}} = f'(0)\Delta V + f''(0)/2(\Delta V)^2. \tag{6.12}$$

Note that $f'(0)$ has units of a transconductance and equation (6.12) is only satisfied when the translinear condition $I_1 \cdot I_2 = I_3 \cdot I_4$ is satisfied, hence the name "translinear Ohm's Law". For the sake of simplicity and to minimize mathematical clutter we will assume $f''(0) \approx 0$ for the rest of the analysis and we will reintroduce the higher-order terms to understand the limitations of the proposed method.

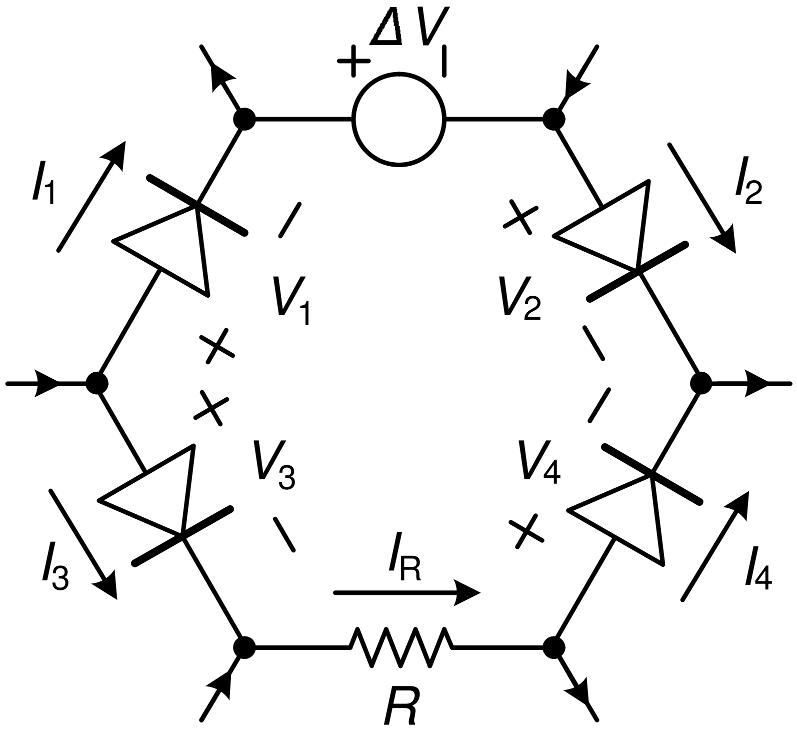


Figure 6.4: The basic concept of translinear Ohm's law.

The translinear Ohm's Law is now applied towards approximating the logarithm of a current using the circuit in Fig. 6.5. The circuit is is derived from a translinear circuit which was reported in [61] where the transistors M_1 to M_4 are biased in weak-inversion and form the translinear-loop.

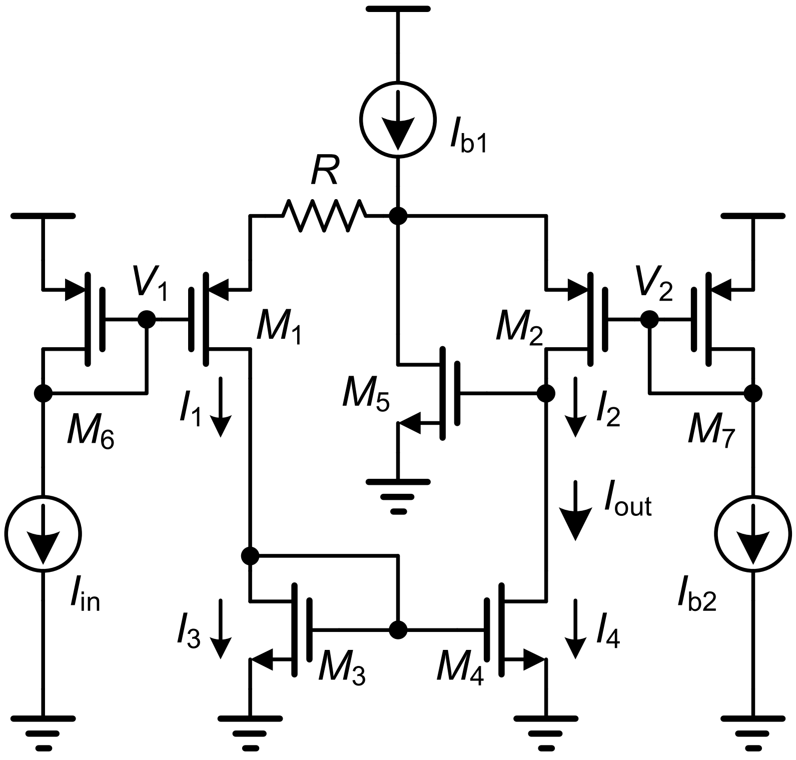


Figure 6.5: Schematic of the translinear logarithmic current-to-current converter.

The drain-to-source voltages for all transistors are larger than 100mV, in which case the transistors satisfy the following translinear relation [110]:

$$NMOS : I_n = S_n I_{D0n} e^{\left(V_{Gn} - V_{Tn}\right)/n_n U_T} e^{-V_{Sn}/U_T}, \tag{6.13}$$

$$PMOS : I_p = S_p I_{D0p} e^{\left(-V_{Gp} + V_{Tp}\right)/n_p U_T} e^{V_{Sp}/U_T}, \tag{6.14}$$

where $S_{n,p}$, $V_{Tn,p}$, $n_{n,p}$, U_T, $V_{Gn,p}$, and $V_{Sn,p}$ are the aspect ratio, the threshold voltage, the sub-threshold slope, the thermal voltage, gate and source voltage referred to bulk potential (V_{dd} or gnd) for nMOS and pMOS transistor respectively. The transistor M_5 serves as a feedback element which reduces the output impedance at the drain of M_2. If the sizes of the transistors are considered to be equal, the current mirror formed by M_3 and M_4 ensures $I_1 \cdot I_2 = I_3 \cdot I_4$. Then, using the translinear Ohm's law the output current I_{out} can be expressed as

$$I_{out} = f'(0) V \left(I_{in} \right), \tag{6.15}$$

where the floating-voltage source $V \left(I_{in} \right)$ equals the difference in gate voltages of transistors M_1 and M_2, which are mirrored from M_6 and M_7 and can be expressed as

$$V \left(I_{in} \right) = n \cdot U_T \cdot \log \left(\frac{I_{in}}{I_{b2}} \right). \tag{6.16}$$

which leads to

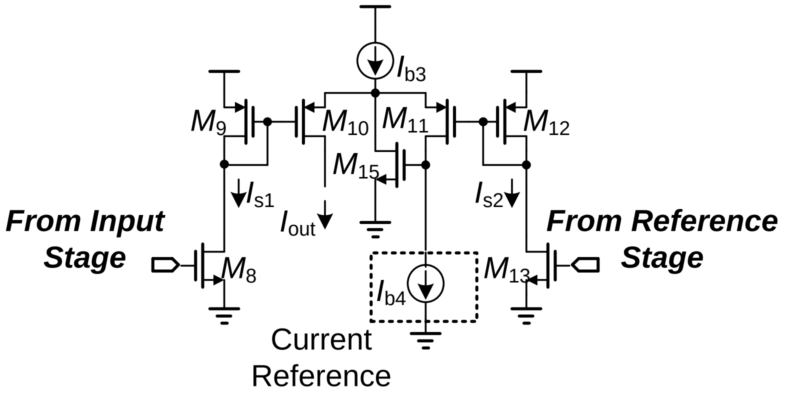
$$I_3 = n \cdot f'(0) \cdot U_T \cdot ln \left(\frac{I_1}{I_2} \right). \tag{6.17}$$

Thus, the output current is proportional to the logarithm of the input current under the condition $I_{in} > I_{b2}$ which is required for the circuit to be operational.

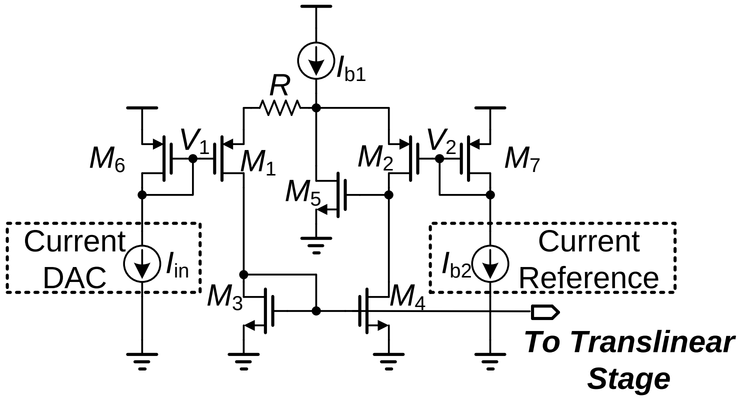
Equation (6.16) consists of several temperature dependent and non-linear parameters which includes: (a) the thermal voltage U_T; (b) the transconductance parameter $f'(0)$; and (c) the sub-threshold slope n. However, all the parameters affect only the gain of the amplifier and hence can be potentially cancelled.

6.4.2.2 Complete Logarithmic Amplifier Circuit

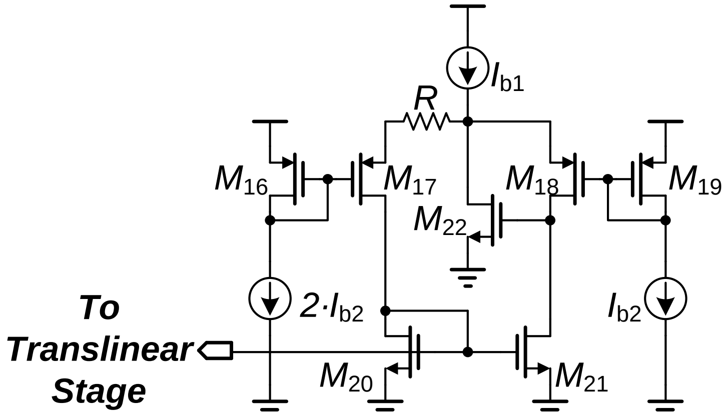
Fig. 6.6(a) shows a complete implementation of a temperature insensitive logarithmic amplifier based on the proposed translinear Ohm's law principle. It consists of an input stage, a reference stage and a translinear stage. The input stage and the reference stage is formed using the basic circuit shown in Fig. 6.6. Based on equation (6.16), the output current generated by the input stage



(a)



(b)



(c)

Figure 6.6: Proposed logarithmic amplifier: (a) basic circuit of translinear stage; (b) basic circuit of input stage; (c) basic circuit of reference stage.

I_{in} is given by

$$I_x = n \cdot f'(0) \cdot U_T \cdot \log\left(\frac{I_{in}}{I_{b2}}\right). \tag{6.18}$$

For the reference stage the ratio of the currents I_1 and I_2 are set to be 2 using the current mirrors M_1,M_2 and M_3. Thus the reference current I_{ref} is given by

$$I_{ref} = n \cdot f'(0) \cdot U_T \cdot \log(2) \tag{6.19}$$

$$I_{ref} = 1.44 \cdot n \cdot f'(0) \cdot U_T. \tag{6.20}$$

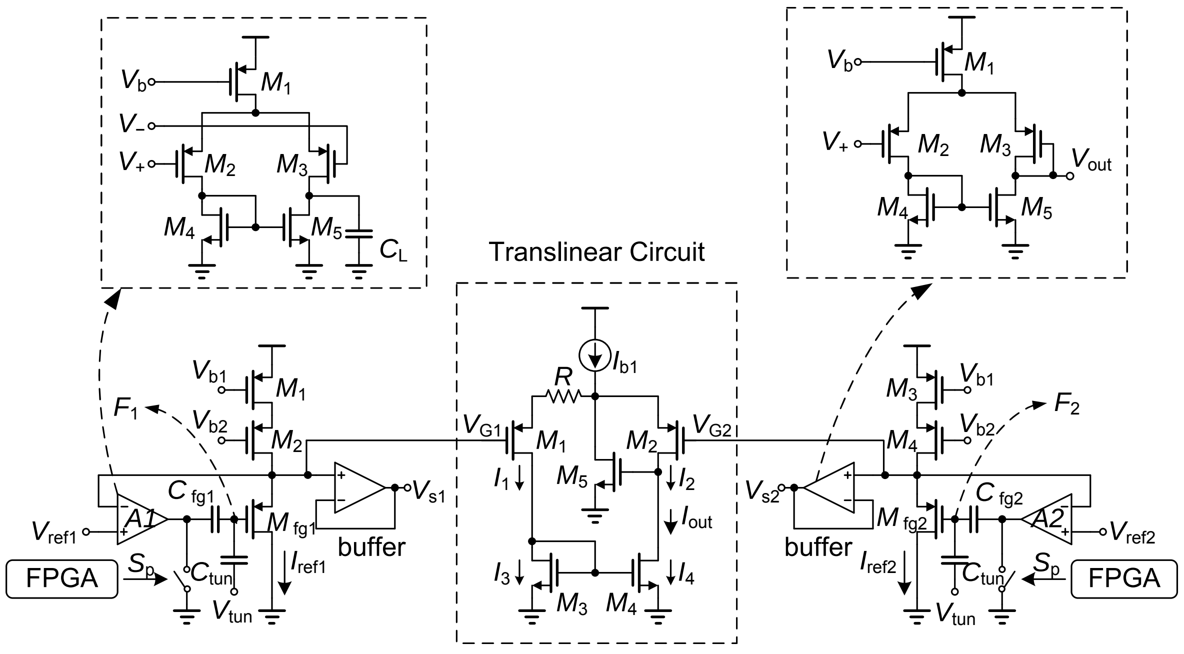
The translinear stage forms a translinear loop using the gate-to-source terminals of the pMOS transistors M_1-M_4 which leads to $I_{out} \cdot I_{ref} = I_{gain} \cdot I_x$. Thus, using equation (6.16) Note that I_{b3} in Fig. 6.6 is an external biasing current to establish the translinear loop which is usually 10 times larger than I_{b4}. M_{15} is added at the input of I_{b4} to ensure M_{11} to be in saturation region. Thus, using equation (6.16) the output current I_{out} can be expressed as

$$I_{out} = I_{gain} \cdot \frac{I_x}{I_{ref}} \tag{6.21}$$

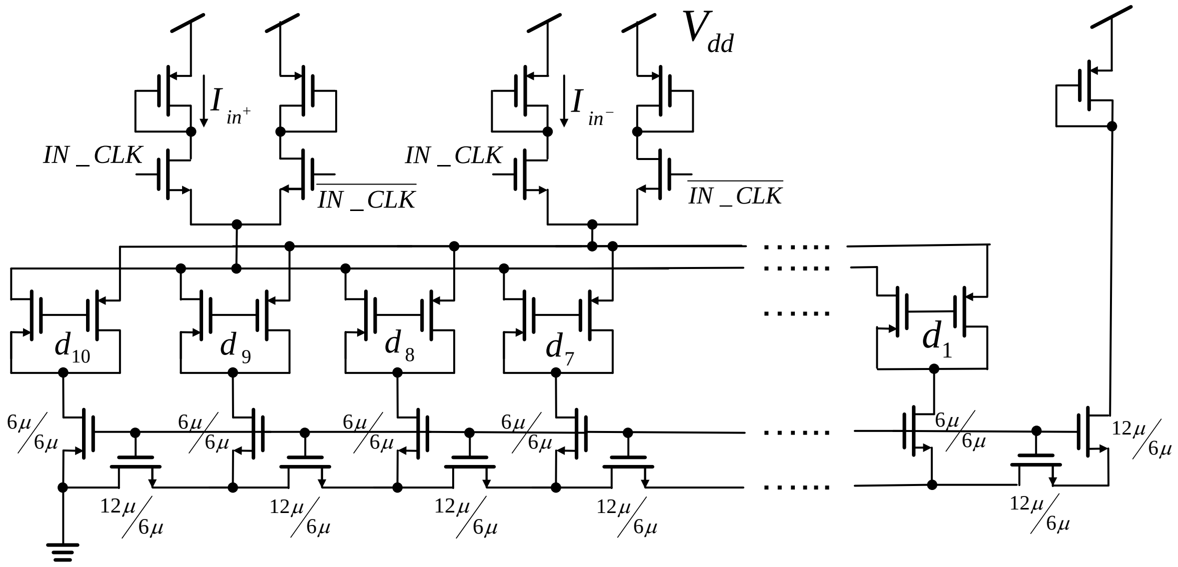
$$I_{out} = 1.44 \cdot I_{gain} \cdot \log\left(\frac{I_{in}}{I_{b2}}\right). \tag{6.22}$$

Thus, if the currents I_{gain} and I_{b2} are independent of temperature, I_{out} is theoretically invariant with respect to temperature. Also, if the resistances R_1 and R_2 are matched all the terms dependent on the resistance gets potentially cancelled out. The current I_{gain} determines the gain of the amplifier and the current I_{b2} determines the minimum input current of the amplifier.

Circuit in Fig. 6.6 requires current sources I_{ref} and I_{b4} that can be programmed to different



(a)



(b)

Figure 6.7: Basic circuit of (a) current reference; (b) current DAC.

amplifier settings. It turns out that the core circuit shown in Fig 6.5 could also be used for designing programmable current references as shown in Fig. 6.7a. If the gate voltages of M1 and M2 are given by V_{G1} and V_{G2}, then the reference current is given by $I_{out} = (V_{G1} - V_{G2})/R$. If the resistance R is temperature compensated and the voltage difference $V_{G1} - V_{G2}$ are independent of temperature then I_{out} will also be temperature compensated. Note that once I_{out} is generated on-chip, different values of currents can be generated using a current mirrors. The voltages V_{G1} and V_{G2} are generated using programmable floating-gate voltage references which was reported in [111]. In this section, we briefly describe the operating principle of the voltage reference and the readers are referred to [111] for additional details. If the floating-gate transistor M4 is matched to the current source transistor M1, the output voltage V_{G1} is given by

$$V_{\mathrm{G1}} = V_{\mathrm{dd}} - nV_{\mathrm{b1}} + n\frac{Q_1}{C_{\mathrm{T}}}, \tag{6.23}$$

where n is the sub-threshold slope factor, V_{dd} and V_{b1} is the supply and biasing voltages, Q_1 the charge on the floating-gate F and C_{T} is the total capacitance at node F. Thus,

$$V_{\mathrm{G1}} - V_{\mathrm{G2}} = n\frac{Q_1 - Q_2}{C_{\mathrm{T}}}. \tag{6.24}$$

The charge Q_1 and Q_2 are programmed using a linear injection technique [111], where the opamps A_1 and A_2 implement an active feedback that maintains the source current, the source and the gate voltage of M_1 and M_2 constant. This elements any non-linear factors that affect hot-electron injection onto the gate, thus achieving a stable and controllable injection rate. Thus the rate and resolution at which Q_1 and Q_2 remains constant throughout the programming range which is rail-to-rail. Programming is achieved by periodically enabling the feedback loop by opening the switch

S_1 (controlled by an FPGA). Again, for the sake of brevity, we have omitted details of hot-electron injection programming which can be found in [111].

Programming of the input current I_{in} is achieved using a 10-bit current-mode DAC as shown in Fig. 6.6. The MSB of the DAC is denoted by d_10 whereas the LSB is denoted by d_1. The architecture of the current DAC (shown in Fig. 6.7 (b)) is based on the popular resistive divider [95] where the current through each branch is twice as large as its neighbor. The output as determined by the bits d_10–d_1 is a binary weighted sum of currents which are mirrored into the input stage. The digital inputs of d_10 through d_1 are transmitted through a shift register chain, which converts the serial input D_{in} into parallel bit slices that are processed by the DAC module.

6.4.3 Adaptive, varactor-driven temperature compensation circuitry

Floating-gate transistors serve as an attractive medium for non-volatile storage of analog parameters. However, conventional current memories based on floating-gate transistors are sensitive to variations in temperature, therefore limiting their applications to only controlled environments. In our research, we propose a temperature compensated floating-gate array that can be programmed to store currents down to picoampere level. At the core of the proposed architecture is a control algorithm that uses a varactor to adapt the floating-gate capacitance such that the temperature dependent factors can be effectively canceled. As a result, the stored current is theoretically a function of a reference current and the differential charge stored on the floating-gates. We validate the proof-of-concept using measurement results obtained from prototype current memory cells fabricated in a 0.5μm CMOS process.

6.4.3.1 Varactor based Floating-gate Temperature Compensation

The architecture of the proposed varactor-based floating-gate current memory cell is shown in Fig. 6.8. It consists of a reference cell formed by two floating-gate transistors P_1 and P_2. Note that in addition to the usual control-gate capacitor C_c and tunneling capacitor C_{tun}, all the floating-gate transistors have a varactor C_v connected to their respective gates. The currents through P_1 and P_2 (measured using an on-chip analog-to-digital converter) are controlled by a control module which sets the control-gate voltage V_c and tuning voltage V_x using an off-chip digital-to-analog converter. If the charge on the floating-gate nodes A and B are denoted by Q_1 and Q_2, then the gate voltages for the pMOS transistors P_1 and P_2 are given by

$$V_A \approx \frac{Q_1 + C_c \cdot V_c + C_v \cdot V_x + C_{tun} \cdot V_{tunnel} + C_b \cdot V_{dd}}{C_T}, \tag{6.25}$$

$$V_B \approx \frac{Q_2 + C_c \cdot V_c + C_v \cdot V_x + C_{tun} \cdot V_{tunnel} + C_b \cdot V_{dd}}{C_T}. \tag{6.26}$$

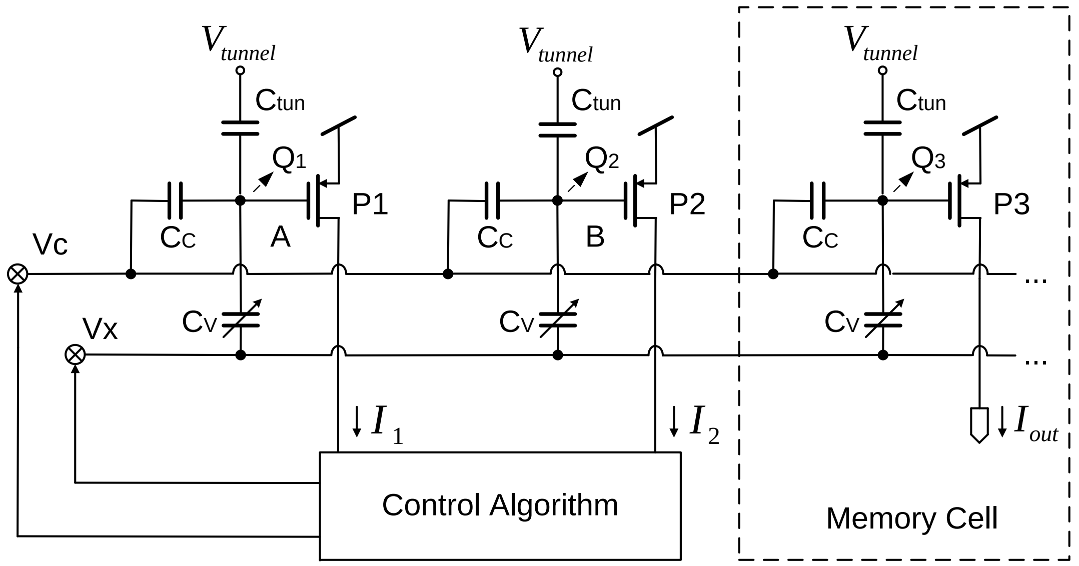


Figure 6.8: Proposed floating-gate based current reference and current memory cells

When P_1 and P_2 are biased in weak-inversion, the respective ratio of their drain-currents I_1

and I_2 is given by

$$\frac{I_1}{I_2} = \exp\left(\kappa \frac{Q_2 - Q_1}{C_T U_T}\right). \tag{6.27}$$

The control module in Fig. 6.8 monitors I_1 and I_2 and ensures that the their ratio remains constant across temperature variations by tuning the voltage V_x which changes the capacitance C_T. Letting $K = I_1/I_2$, then the parameter $C_T U_T$ is also held constant according to

$$C_T U_T = \kappa \frac{Q_2 - Q_1}{\log(K)} \tag{6.28}$$

which leads to the expression for the output current I_{out} generated through the memory cell as

$$\frac{I_1}{I_{out}} = \exp\left[\log(K)\frac{Q_3 - Q_1}{Q_2 - Q_1}\right]. \tag{6.29}$$

Thus, if the current I_1 is compensated for temperature, then according to equation (6.29), the current through the memory cells are also temperature compensated. In principle, I_1 could be generated by a bandgap reference circuit [59]. For this implementation, we tried to maintain I_1 to be constant by co-adapting the control-gate voltage V_c along with the tuning voltage V_x.

6.4.3.2 Varactor Implementation and Second-order Effects

The varactor has been realized using a MOS capacitor which operates in an accumulation mode [112]. The cross-sectional architecture and the layout of the MOS-cap is shown in Fig. 6.9 along with the floating-gate transistor, the control-gate and the tunneling-node. The use of MOS-cap as a varactor introduces second-order effects into the equation (6.29):

- Derivation of equation (6.29) assumed that the floating-gate transistors including the variable capacitors C_v are matched with respect to each other. Unfortunately, each floating-gate node is programmed to a different potential, which will lead to mismatch in C_v. In weak-inversion, however, the difference between respective gate-voltages is small (less than 100mV), which reduces the effect of the mismatch.

- The variation of C_v with respect to the tuning voltage V_x is non-linear as shown in the measured response in Fig. 6.21. Therefore, V_x and V_c has to be biased properly to ensure: (a) sub-threshold biasing of the transistors; and (b) high-sensitivity of C_v with respect to V_x.

- The MOS-cap C_v also varies with temperature. However, the temperature model of a MOS varactor is too complex for any closed-form calibration, which motivates adapting V_x in an online manner.

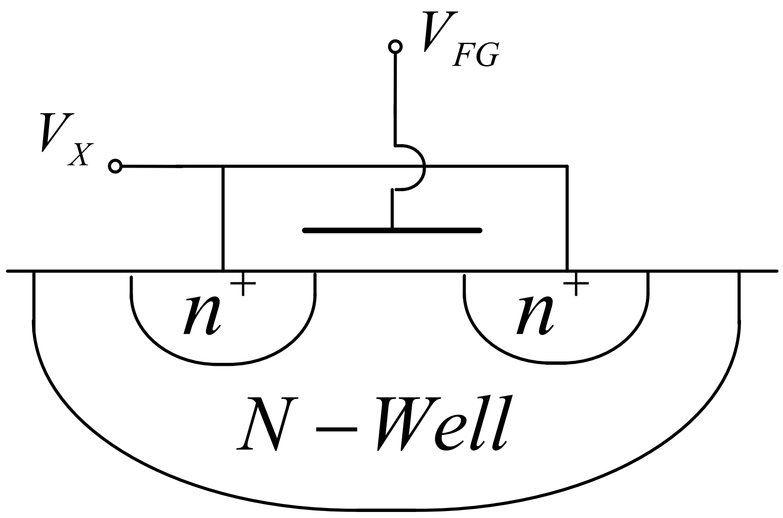
6.5 Measurement Results

The prototype is still being measured for the SVM computation result.

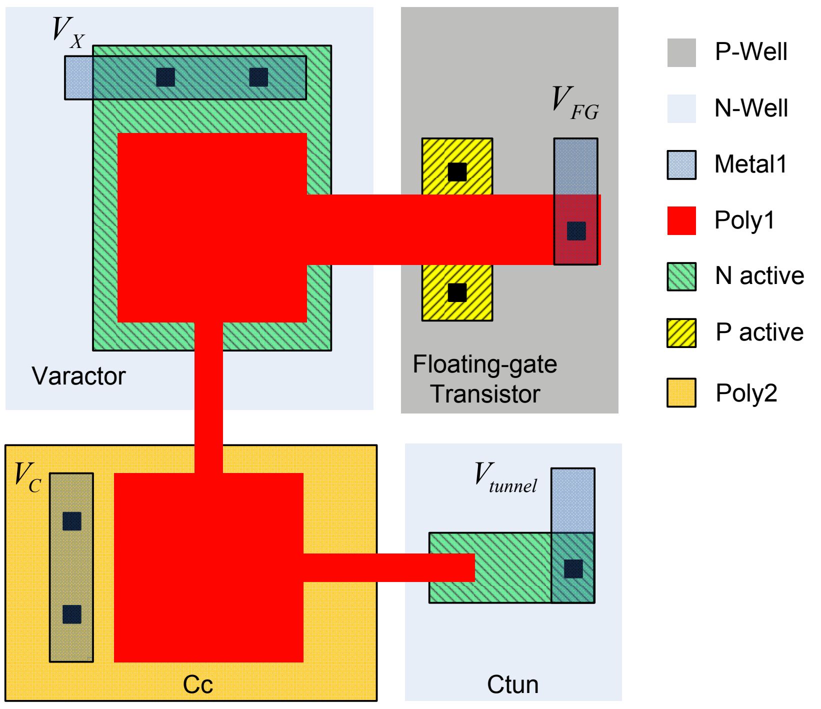
6.5.1 Measurement results for floating-gate memory

As is seen through Fig. 6.3, the programming of floating-gate cells are controlled by Vcurrent_starving, Vdd_fg, V_c, V_x and the charge injection voltage applied at program pin. In the first set of experiment, we vary one of the above parameters and keep the others fixed to verify the functionality of these control parameters.

Fig. 6.10 demonstrates the current through three different floating-gate transistors when different gate voltages are applied to the current starving transistor P_0 during charge injection. These



(a)



(b)

Figure 6.9: (a) Cross-sectional architecture of MOS varactor (b) Layout for memory cell in Fig. 6.8

three cells are tunneled until their current is lower than picoampere. As Vcurrent_starving is decreasing, the charging velocity is increasing. This is because the injection current through the transistor is increased by lowering the gate voltage of P_0.

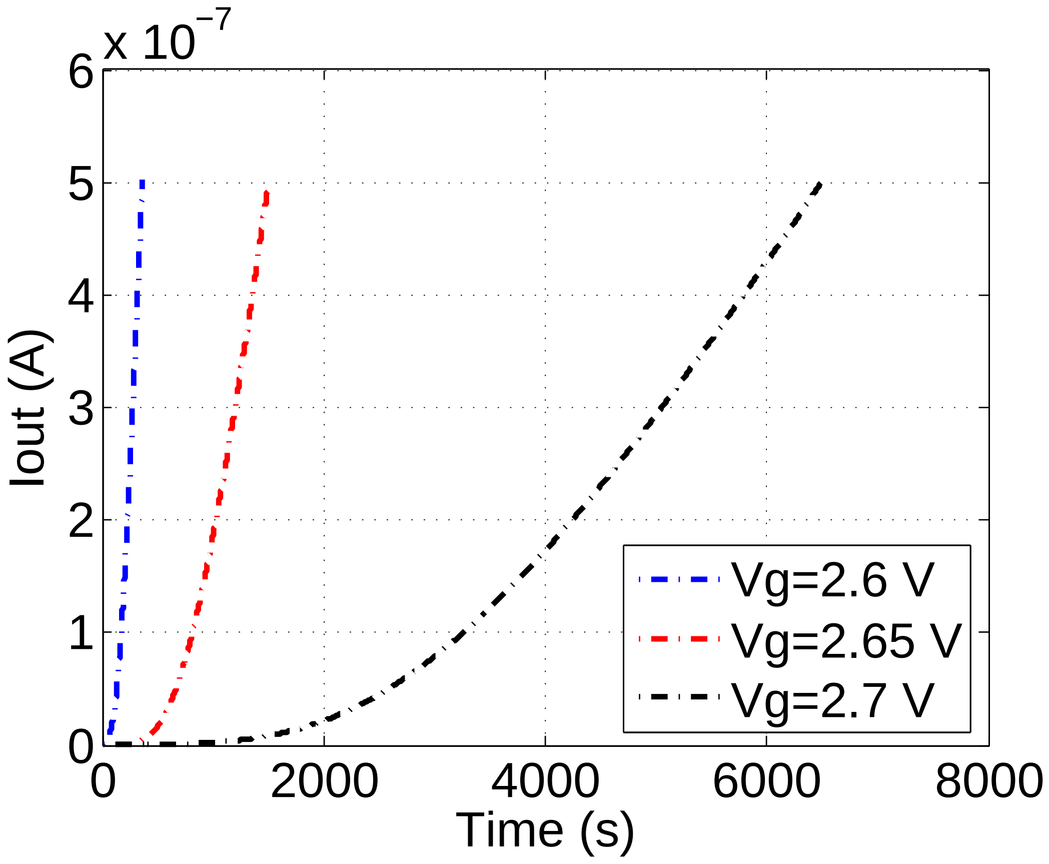


Figure 6.10: Floating-gate transistor current with respect to current starving voltage

Fig. 6.11 shows the current through three different floating-gate transistors when different Vdd_fg are applied. It is seen that as Vdd_fg is increasing, the charging velocity is increasing as well. The phenomenon is consistent with the principle that the injection current is increased with a higher V_{SG} across the floating-gate transistor.

Fig. 6.12 shows the current through three different floating-gate transistors when different V_xs are applied. It is seen that as V_x is increasing, the charging velocity is decreased. Since increasing V_x will increase the gate voltage of the floating-gate transistor, the phenomenon is consistent with

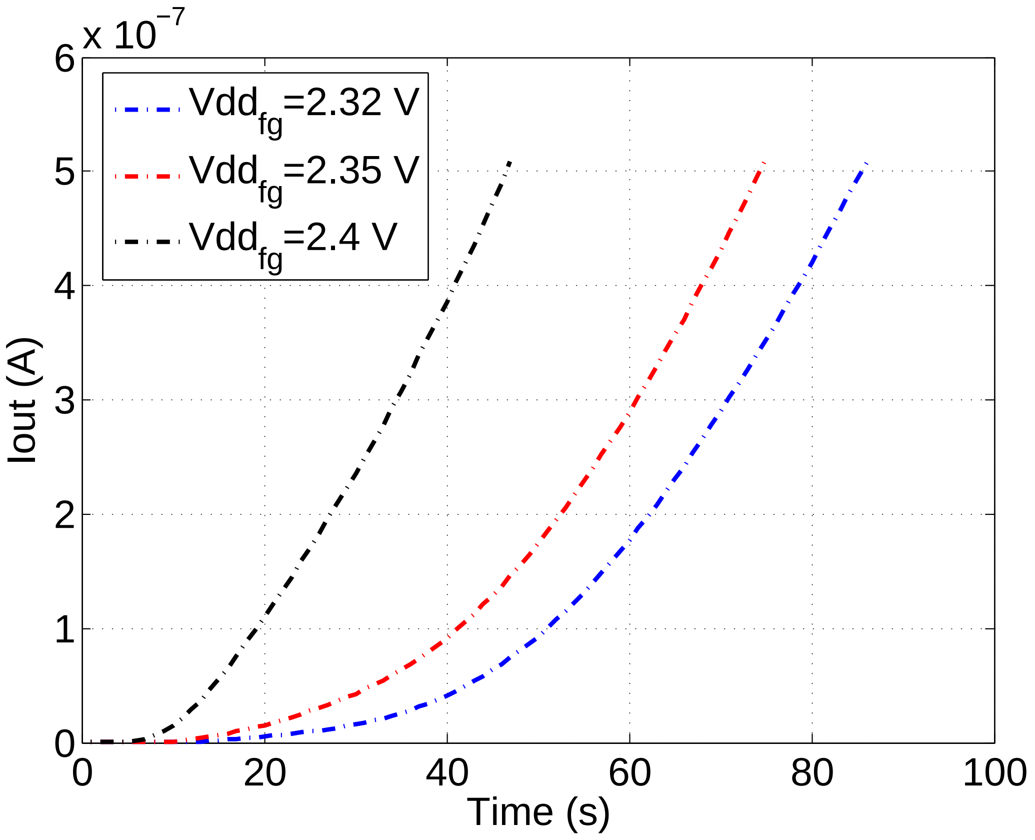


Figure 6.11: Floating-gate transistor current with respect to Vdd_fg

the principle that the injection current is proportional to V_{SG} across the floating-gate transistor.

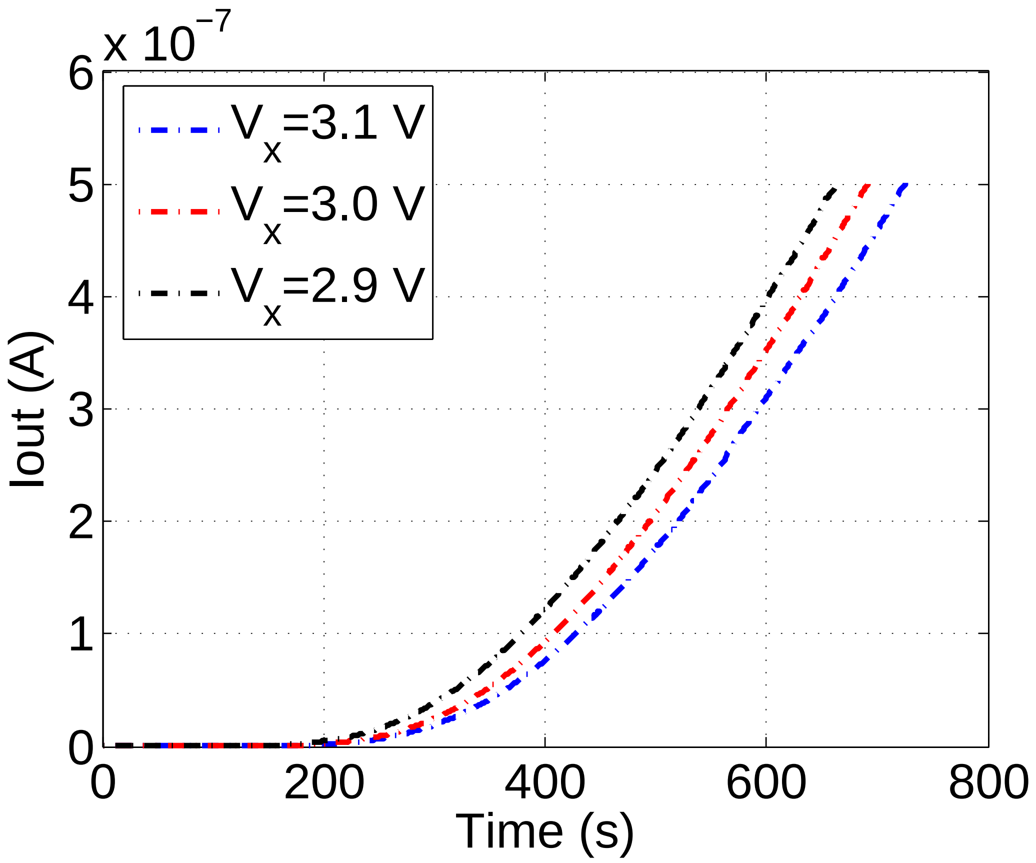


Figure 6.12: Floating-gate transistor current with respect to V_x

Fig. 6.13 shows the precise programming procedure by tuning the voltage applied at program pin, i.e., by tuning the injection current. The methodology is: first charge the cell to some threshold value that is approaching the desired value, then reduce V_{PROD} to allow a lower injection current, which ensures a fine step size. For this experiment, the desired value is 6.3 μA, and the threshold is set at 6.1 μA. It is seen that the current is increasing rapidly as V_{PROD} is as low as as -3.3V; however, as the current is approaching 6.1 μA, V_{PROD} is switched to -2.8V. As a result, the current through the floating-gate transistors are programmed to 6.3 μA precisely, as is seen from Fig. 6.13.

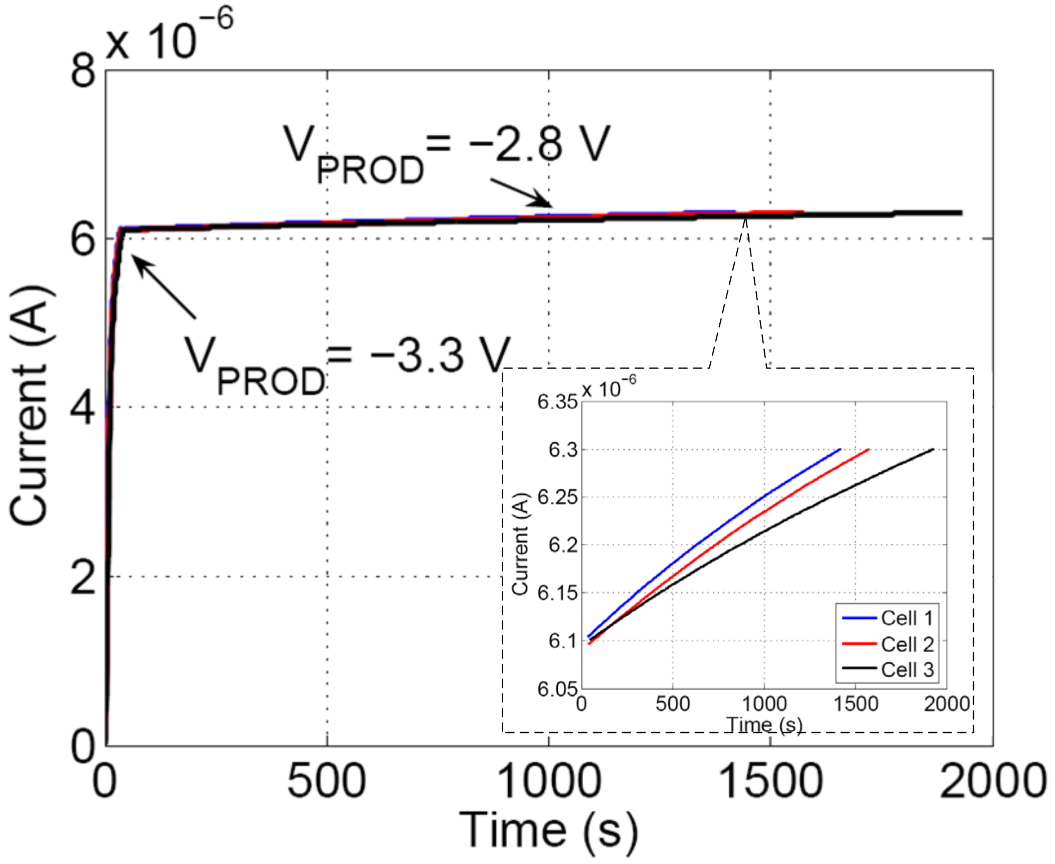


Figure 6.13: Precise programming procedure for floating-gate transistors

6.5.2 Measurement results for current-input current-output CMOS logarithmic amplifier

The microphotograph of a prototype current-input current-output CMOS logarithmic amplifier fabricated in a 0.5μm standard CMOS process is shown in Fig. 6.14. Table 6.3 summarizes the main specifications of the chip.

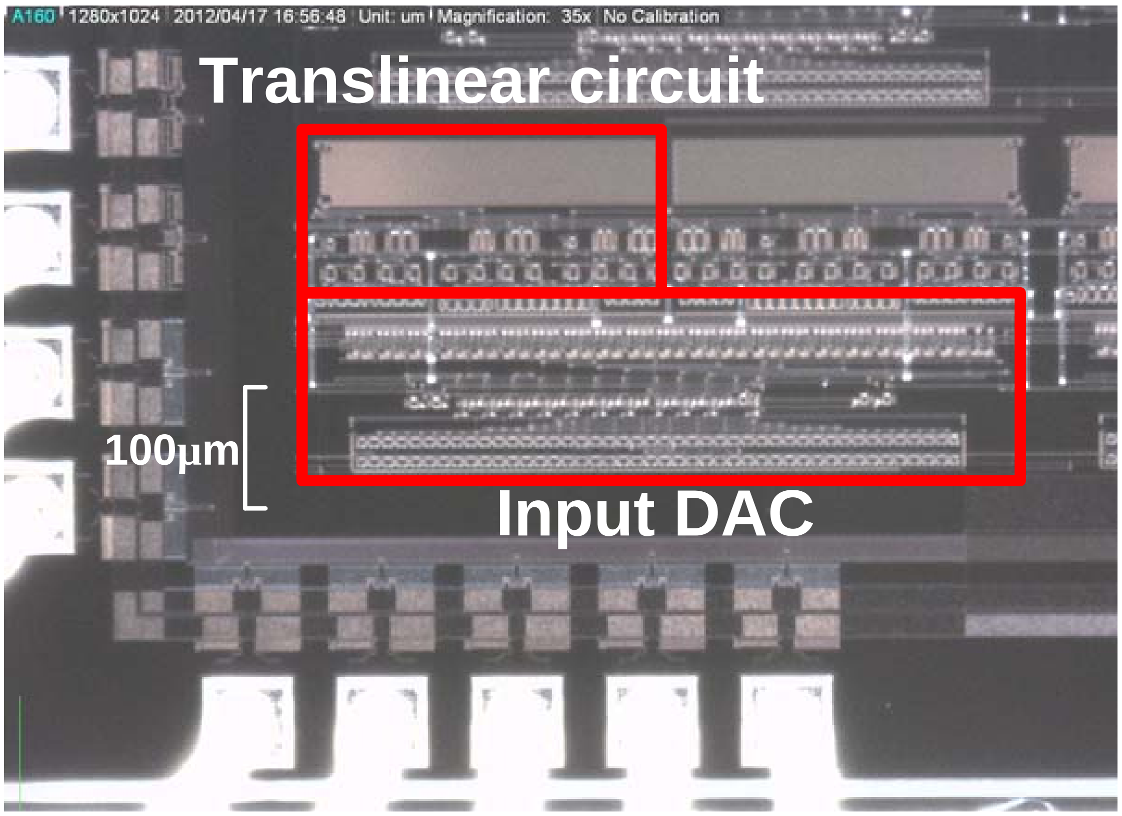

A160 1280x1024 2012/04/17 16:56:48 Unit: um Magnification: 35x No Calibration
Translinear circuit
100µm
Input DAC

Figure 6.14: Die microphotograph of the chip.

6.5.2.1 Measurement results for the current reference

Fig. 6.15 shows the measured output current I_{out} (Refer to Fig. 6.7 (a)) versus the differential gate voltage V_1-V_2 (Refer to Fig. 6.7 (a)). During this experiment, the switches S_ps are set to be off so that both V_1 and V_2 are determined by the V_{ref} of the opamps. V_1 is varied while V_2 is

Table 6.1: Specifications of the current-input current-output CMOS logarithmic amplifier

Fabrication Process	Standard CMOS $0.5\mu m$
Die Size	$3000\mu m \times 3000\mu m$
$M_{1\text{-}4,\,6\text{-}7}$ (refer to Fig. 6.5)	10μm/5μm
M_5 (refer to Fig. 6.5)	30μm/1.5μm
R (refer to Fig. 6.5)	10MΩ

fixed. It is seen from Fig. 6.15 that I_{out} is proportional to the differential voltage, which proves that equation (6.11) holds for this circuit.

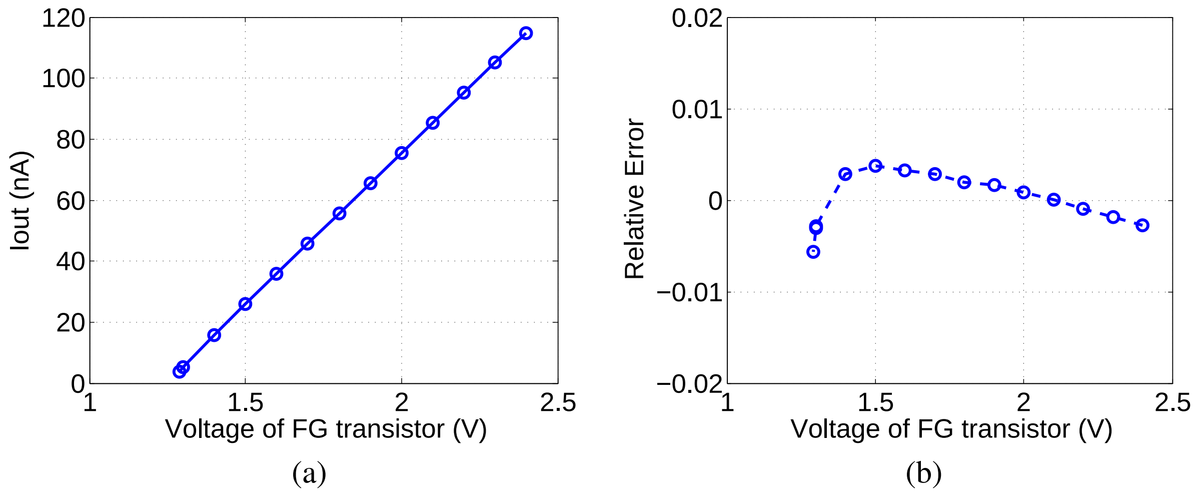


Figure 6.15: (a) Measurement current response and (b) relative error for the current reference.

6.5.2.2 Measurement results for the logarithmic amplifier

The measured result of the circuit is shown in Fig. 6.16. The input current was varied from 100pA to 40nA and for each measurement with different value of I_{b4} (Refer to Fig. 6.6). The results show a close agreement between the measured result and the mathematical model given by equation (6.22). The results also demonstrate that a larger dynamic range can be achieved by increasing I_{b4}.

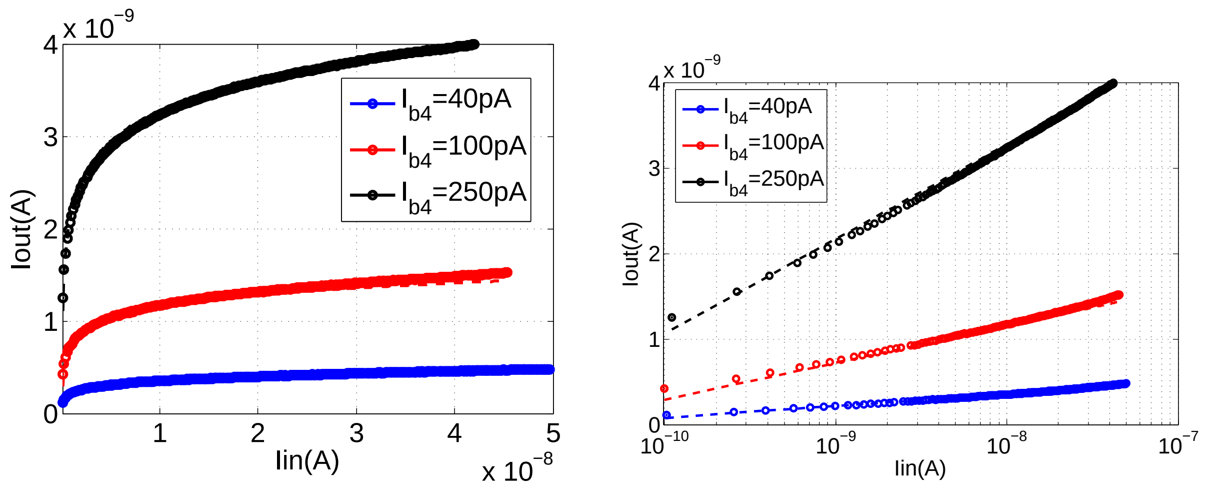


Figure 6.16: Measurement results for the logarithmic current conversion.

6.5.2.3 Noise Analysis

When the current memory cell is operating at sub-nA current levels, the thermal noise and flicker noise (due to sub-threshold MOSFETs and the resistor) effects the accuracy of the current read-out from the cell. Under DC condition, flicker noise is dominant. The power spectral density (PSD) of flicker noise for MOS transistors biased in subthreshold region can be expressed by

$$S_{I_D} = \frac{KF'}{WLC_{ox}} \cdot \frac{I_D^2}{f}. \tag{6.30}$$

Note KF' is a process-dependent constant; C_{ox} is the gate oxide capacitance per unit area; W and L represent the width and length of the transistor respectively; I_D is the drain current flowing through the transistor; and f represents the frequency. Equation (6.30) indicates that the noise spectral density will increase along with the biasing current.

To verify equation (6.30), a set of experiment is performed in which I_{in} and I_{out} are varied whereas I_{b4} and I_{b2} are fixed. The noise is obtained based on the standard deviation of the 100 current samples for each pair of I_{in} and I_{out}. Fig. 6.17 (a) shows the noise of I_{out} is proportional to the magnitude of I_{out}, and Fig. 6.17 (b) shows the corresponding SNRs.

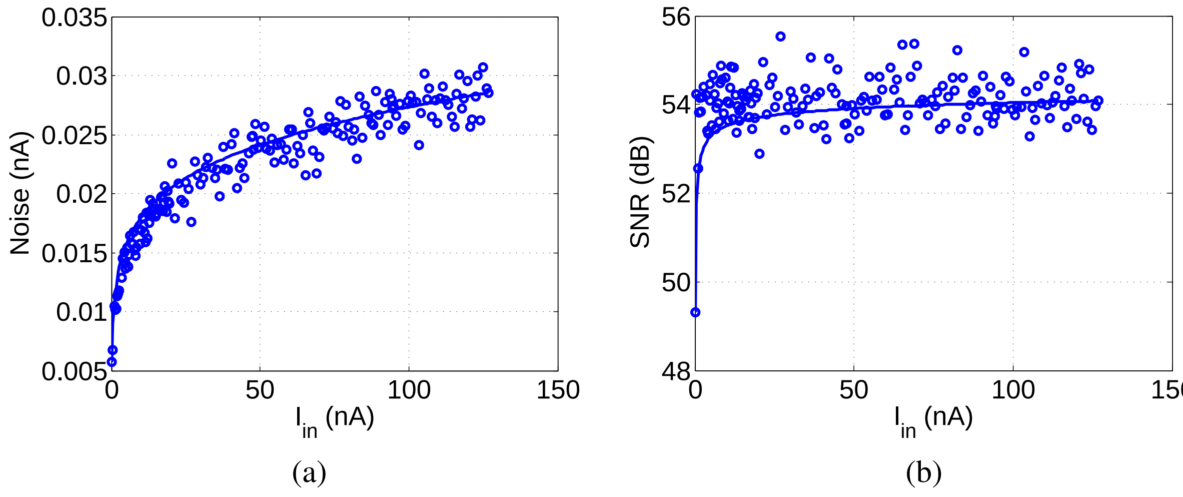


Figure 6.17: (a) Noise of I_{out} versus I_{in}; (b) SNR of I_{out}.

6.5.2.4 Temperature characteristics

The underlying principle of the translinear Ohm's law intrinsically leads to a temperature compensated operation. According to equation (6.22), the temperature coefficient of I_{out} is determined by the temperature characteristics of I_{b4}, I_{b2}, and I_{in}. Assume that I_{b4} and I_{b2} are not temperature compensated whereas I_{in} is temperature compensated. Hence I_{b4} and I_{b2} are written as functions of temperature $I_{b4}(T)$ and $I_{b2}(T)$. Then equation (6.22) is rewritten as:

$$I_{out}(T) = I_{b4}(T) \cdot \frac{ln(I_{in}) - ln(I_{b2}(T))}{ln2}. \tag{6.31}$$

For a specific temperature T_0, three $I_{out}(T_0)$ are sampled as

$$\begin{aligned} I_{out_1}(T_0) &= I_{b4}(T_0)\cdot\frac{ln(I_{in_1}) - ln(I_{b2}(T_0))}{ln2}, \\ I_{out_2}(T_0) &= I_{b4}(T_0)\cdot\frac{ln(I_{in_2}) - ln(I_{b2}(T_0))}{ln2}, \\ I_{out_3}(T_0) &= I_{b4}(T_0)\cdot\frac{ln(I_{in_3}) - ln(I_{b2}(T_0))}{ln2}. \end{aligned} \tag{6.32}$$

Then it can be deduced that

$$\frac{I_{out_1}(T_0) - I_{out_2}(T_0)}{I_{out_1}(T_0) - I_{out_3}(T_0)} = \frac{\frac{I_{b4}(T_0)}{ln2}\cdot ln\left(\frac{I_{in_1}}{I_{in_2}}\right)}{\frac{I_{b4}(T_0)}{ln2}\cdot ln\left(\frac{I_{in_1}}{I_{in_3}}\right)} = \frac{ln\left(\frac{I_{in_1}}{I_{in_2}}\right)}{ln\left(\frac{I_{in_1}}{I_{in_3}}\right)} \tag{6.33}$$

As is seen from equation (6.33), the temperature related items are canceled. Hence $\frac{I_{out_1}(T)-I_{out_2}(T)}{I_{out_1}(T)-I_{out_3}(T)}$ is temperature compensated only if I_{in} is temperature compensated.

A corresponding experiment with temperature varying from 27^{o}C to 57^{o}C is conducted. The experimental result is shown in Fig. 6.18. It is seen that in the weak-inversion regime, the current ratios are approximately invariant with respect to temperature. That validates the relationship given by equation (6.33). Based on the first order polynomial for the data shown in Fig. 6.18, a 230 ppm/K is achieved over the temperature range of 27^{o}C to 57^{o}C.

6.5.2.5 Speed analysis

The speed of the circuit is limited by current. According to equation (6.22), the speed limit factor could be I_{in}, I_{b4}, and I_{b2}. The following experiment is conducted to figure out the bandwidth variation with respect to these factors. In the experiment, I_{in} and I_{b4} are varied whereas I_{b2} is fixed to be 100pA. The result is demonstrated in Fig. 6.19. Against Fig. 6.19 (a), it is seen that

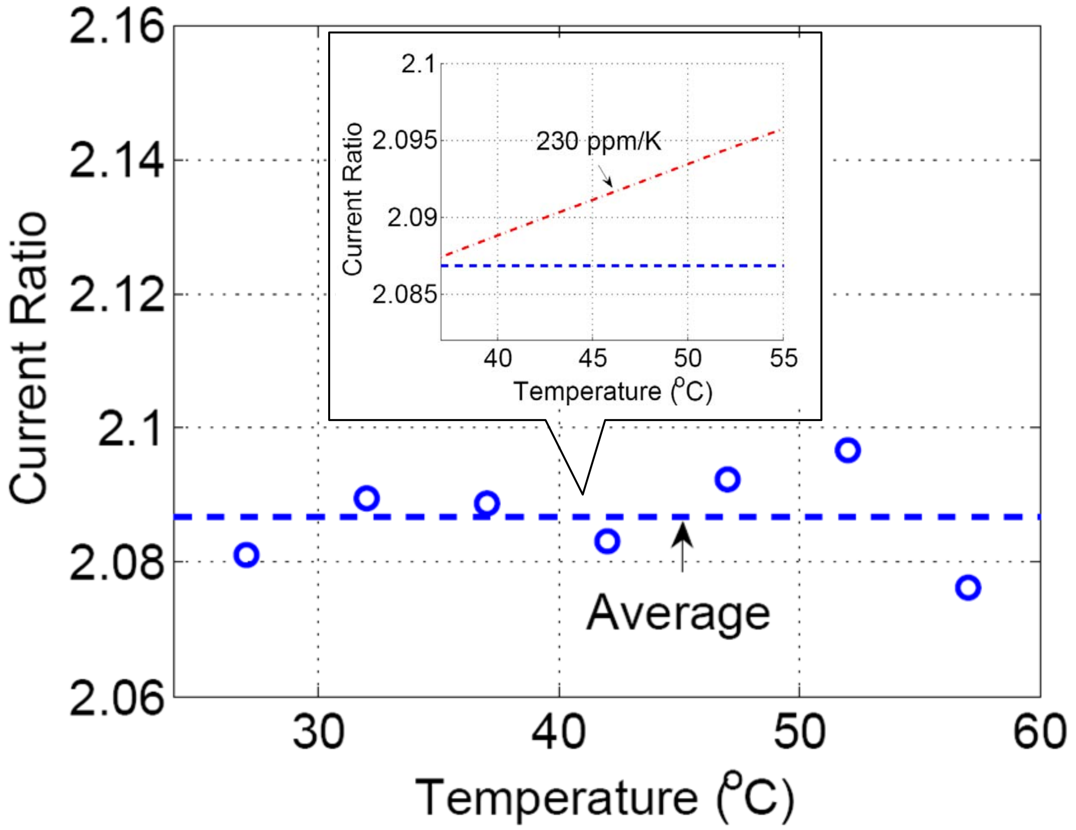
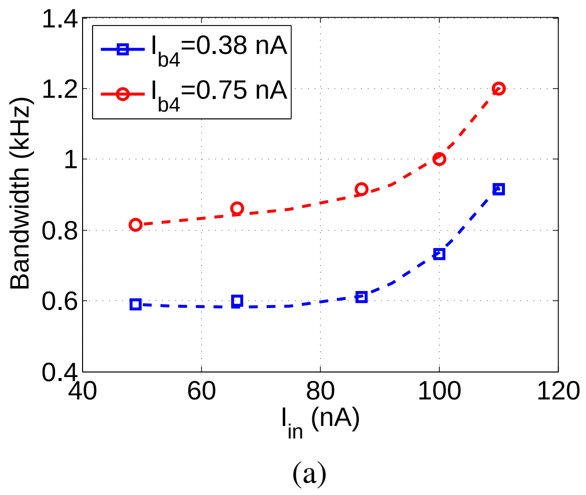


Figure 6.18: Measured response plotting $\dfrac{I_{out_1}(T) - I_{out_2}(T)}{I_{out_1}(T) - I_{out_3}(T)}$ under different temperatures.

when I_{b4} is set to 0.38nA, when I_{in} varies from $50nA$ to $110nA$, the bandwidth is increased from 0.6kHz to 0.9kHz. However, when I_{b4} is set to 0.75nA, when I_{in} varies from $50nA$ to $110nA$, the bandwidth is increased from 0.8kHz to 1.2kHz. From Fig. 6.19 (b), it is seen that when I_{in} is set to $85nA$, bandwidth is monotonically increased from 0.45kHz to $1kHz$ as I_{b4} varies from 0.25nA to 2.75nA; however, when I_{in} is set to $110nA$, bandwidth increases from 0.73kHz to 1.1kHz as I_{b4} varies in the same range. These phenomena demonstrate that the bandwidth is increased when bias current are increased.



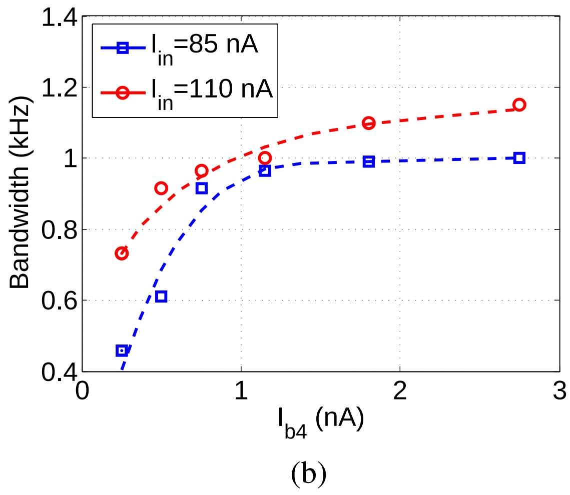


Figure 6.19: Measured response plotting bandwidth with respect to current.

Table 6.2 compares this work with some of the references.

6.5.3 Measurement results for varactor-driven temperature compensation circuitry

An array of varactor-based FG current memory cells has been prototyped in a 0.5μm standard CMOS process. Table 6.3 summarizes the specification of the fabricated prototype and Fig. 6.20 shows its micrograph. Programming of the FG memory cells is based on the hot-electron injection

Table 6.2: Comparison of performance

Reference	[113]	[114]	[115]	[108]	This work
Conversion	current-in voltage-out	current-in voltage-out	current-in voltage-out	current-in voltage-out	current-in current-out
Process	$1.6\mu m$	$1.5\mu m$	Discrete	$0.5\mu m$	$0.5\mu m$
Area	-	$2.9 \times 3.7mm^2$	-	$91 \times 75\mu m^2$	$290 \times 140\mu m^2$
Supply	-	± 6 V	-	3.3 V	3.3 V
DR	100dB	114dB	80dB	140dB	120dB
Bandwidth	$>$10Hz	1kHz	25kHz	$>$3.5kHz	1.2kHz
Power	10pA-1μA	30mW	-	0.1μW-33μW	1nW-10μW
ppm/K	uncompensated	uncompensated	uncompensated	uncompensated	230

and Fowler-Nordheim (FN) tunneling, whose details have been presented elsewhere [61, 116] and has been omitted here for the sake of brevity.

Table 6.3: Specification of the fabricated prototype

Fabrication Process	Standard CMOS $0.5\mu m$
Die Size	$3000\mu m \times 3000\mu m$
Size of P_1-P_3	$6\mu m/3\mu m$
C_c	100fF

In the first set of experiments, the floating-gate voltage V_A was measured (using a voltage buffer shown in Fig 6.8) when V_x was tuned and $V_c = 1.2V$. The ratio $\frac{C_v}{C_c}$ was determined based on the sensitivity analysis of the current I_1 with respect to V_x and V_c. The measurement result in Fig. 6.21 shows that the capacitance is inversely proportional to V_x and shows the largest change happens at $V_x = 1.25V$. The experiment also demonstrates that C_v can be varied over a large dynamic range with respect to V_x, which should be sufficient to compensate for large temperature variations.

For the next set of experiment, we measured the current ratios under four different tempera-

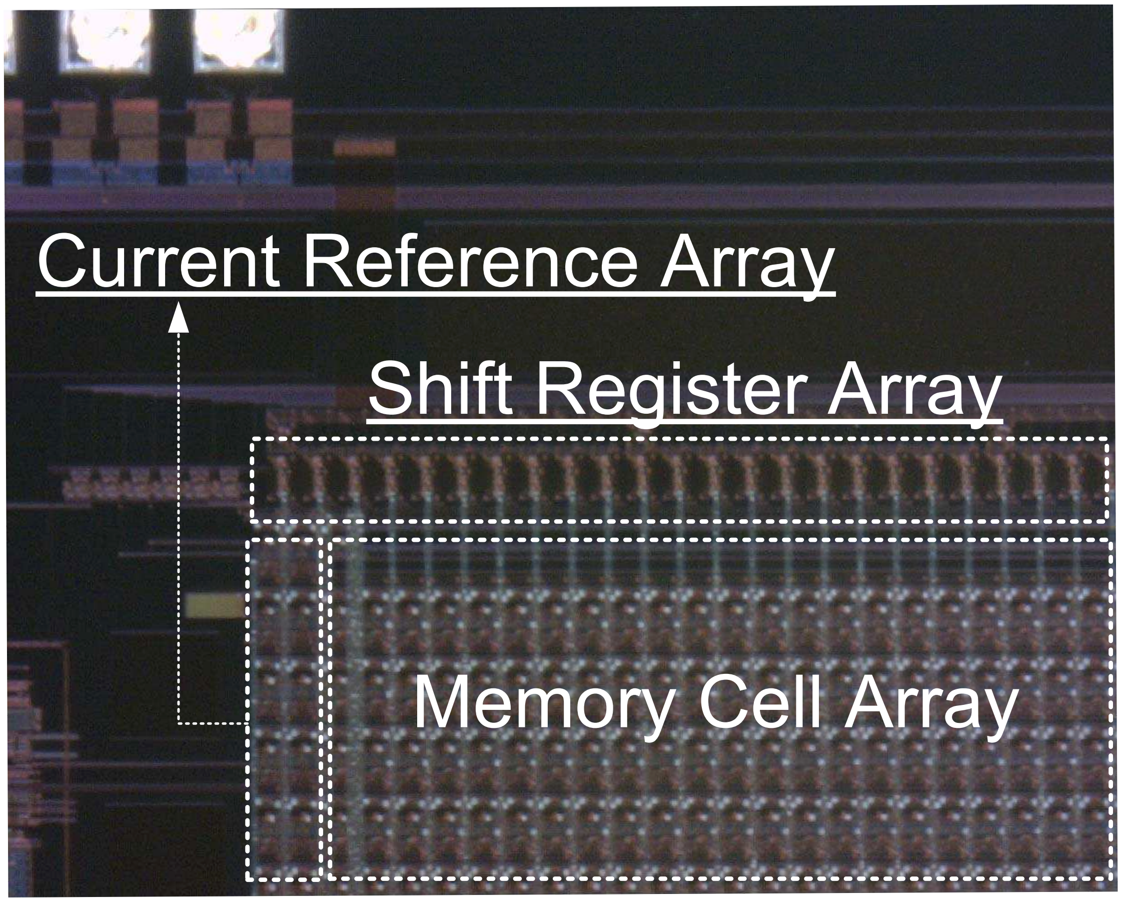


Figure 6.20: Die microphotograph of the chip

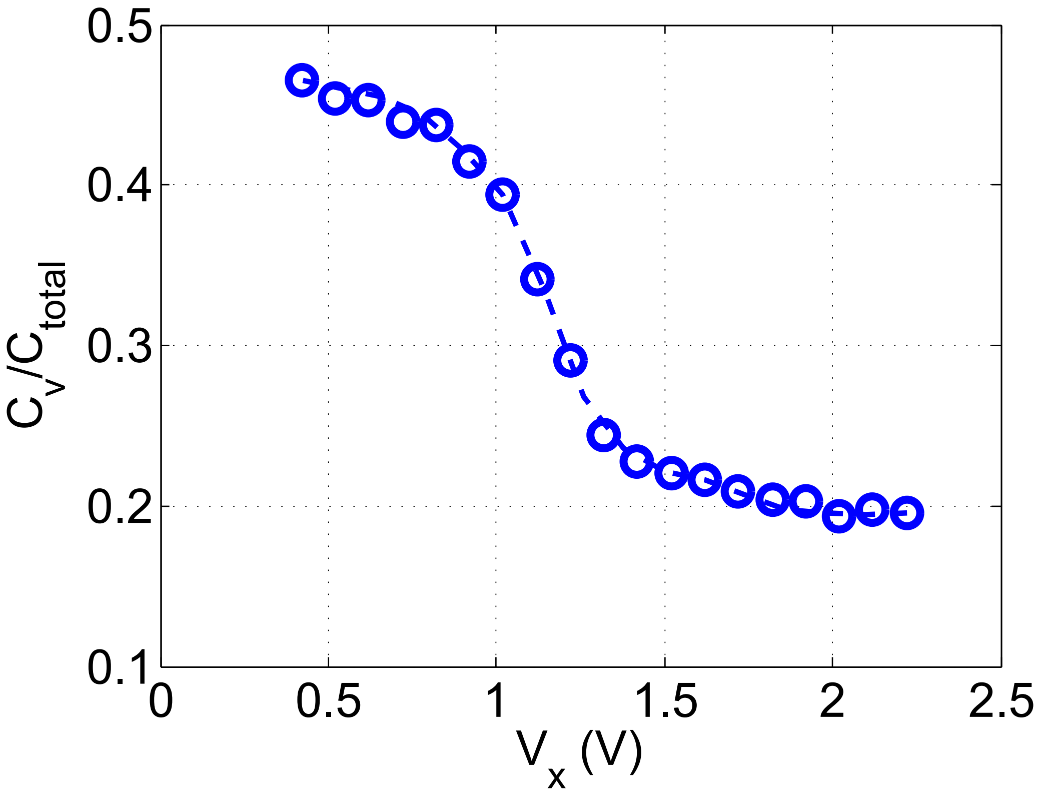
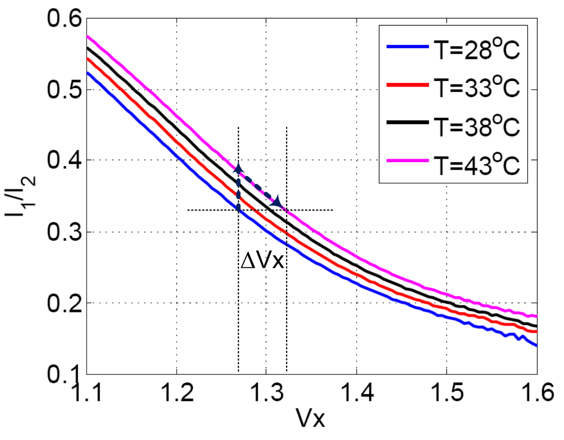
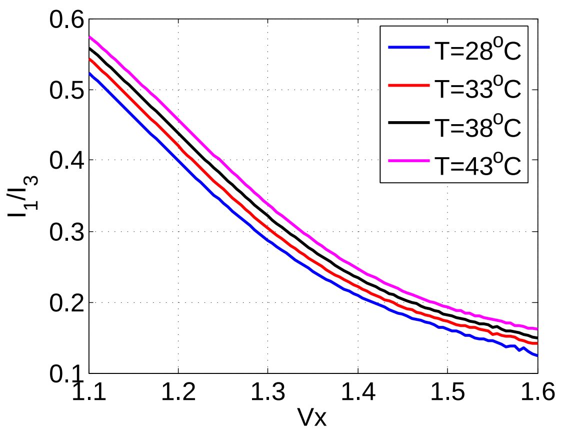


Figure 6.21: The C-V curve of MOS capacitor

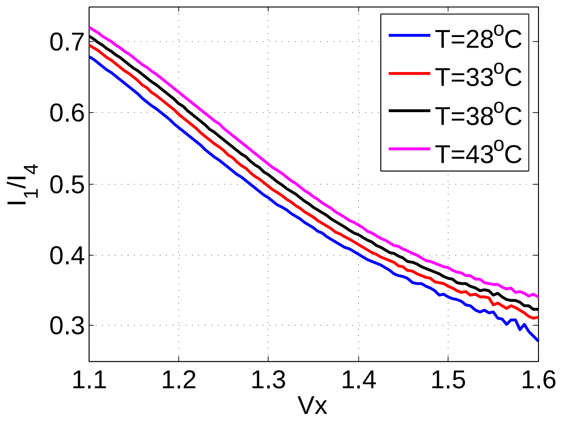
tures, namely, 28°C, 33°C, 38°C, and 43°C, when the voltages V_C is fixed to be 2V and V_x is varied. The results shown in Fig. 6.22(a) - (e) demonstrate that with the increase in temperature, the respective current ratios decrease as indicated by the relationship in equation (6.28). The control algorithm then varies V_x to ensure that the ratio $\frac{I_1}{I_2}$ is maintained to be constant. The trajectory of the V_x traversed by the control algorithm is shown in Fig. 6.22(a) - (e) shows the corresponding value of the ratio $\frac{I_1}{I_{out}}$ (Here I_{out} is represented by I_3-I_6 from different memory cells). The compensated results with V_x biased at 1.2 V and 1.3V are demonstrated in Fig. 6.23. A 150 ppm/K is achieved over the temperature range of 28^oC to 43^oC. It is seen that the current through other memory cells (I_3-I_6) can be effectively compensated with respect to temperature once $\frac{I_1}{I_2}$ is constant, no matter how different the current is from I_1. This result validates the proof-of-concept temperature compensation for the proposed floating-gate current memory.



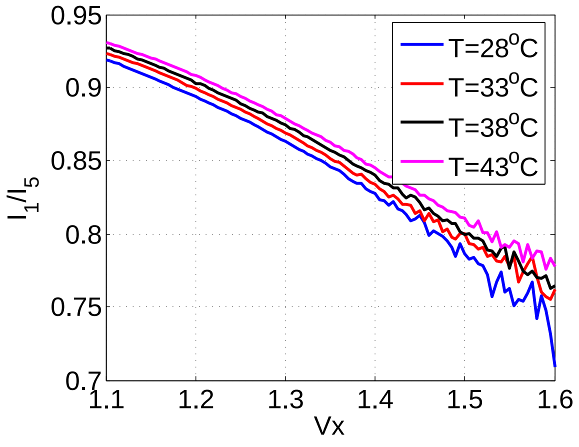
(a)



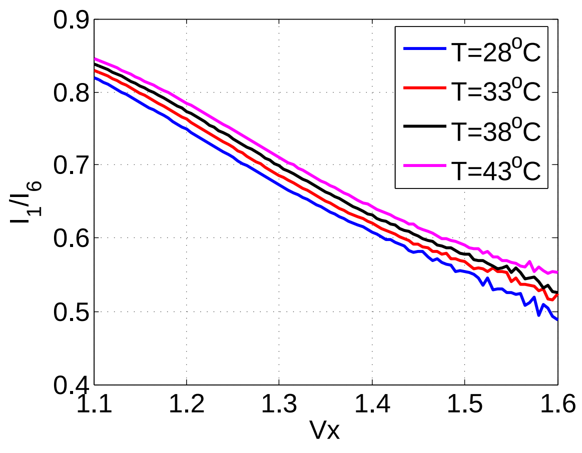
(b)



(c)

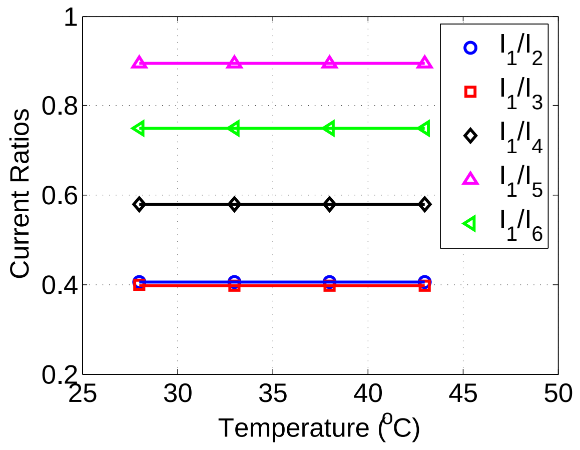


(d)

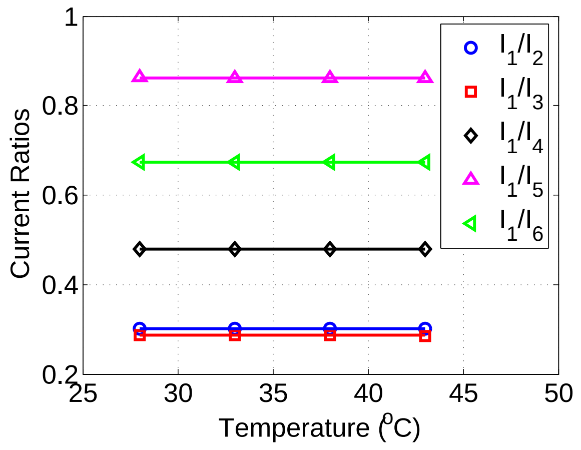


(e)

Figure 6.22: Measured response plotting (a) $\frac{I_1}{I_2}$ (b) $\frac{I_1}{I_3}$ (c) $\frac{I_1}{I_4}$ (d) $\frac{I_1}{I_5}$ (e) $\frac{I_1}{I_6}$ with respect to V_x under 28°C, 33°C, 38°C, and 43°C.



(a)



(b)

Figure 6.23: Measured response plotting temperature compensated current ratios under V_x of (a)1.2V (b)1.3V.

Chapter 7

Conclusions and future work

The major conclusion of this thesis is that it is feasible to use MP algorithm as an effective alternative for analog computation. The MP algorithm can be implemented with current-mode analog VLSI circuit. The basic MP circuit can be utilized to synthesize a variety of static analog computation circuits. Also MP circuit serves as a fundamental circuit unit in the MP-involved applications.

The proposed MP algorithm is a novel analog computation algorithm with the underlying principle of universal conservation law. In contrast with the state-of-the-art transliner circuits dependent on physics of electronic devices, MP algorithm is dependent on the physical law, which facilitates it independence on transistor operation regions. As a consequence, MP circuit has a wide dynamic range, and is capable of trade off one performance with another when required due to its wide operation region. Also the underlying principle of universal conservation law enables MP algorithm to be implemented in a diversity of devices, not limited to electronic device (e.g. mass, time, resistance, etc.). Since current-mode circuits is the most easy and mature way of implementation, in this dissertation, we adhere to current-mode implementation.

One characteristic of MP algorithm is that by using MP algorithm, the PWL approximation of "log-sum-exp" function can be achieved. MP algorithm also has a couple of mathematical properties, which facilitate the MP-based computations. Another characteristic of MP algorithm lies in the implementation based on reverse water-filling criterion, which enables MP to be implemented in analog domain easily.

The two characteristics mentioned above entail diverse analog arithmetic computations to be synthesized on basic MP circuit. In this study, the investigated computations include addition, subtraction, multiplication, division, power, polynomial, tanh, inner product. These computations are ubiquitous in applications such as neural network, communications, machine learning, and the like.

The two MP-based application examples brought out in this research study are an analog LDPC decoder and an analog SVM. For each application, the mathematic model built upon MP algorithm is illustrated, the simulation results are presented, the system architecture and circuit implementation based on MP circuit are demonstrated, and the measurement results are presented as well.

7.1 Summary of contributions

The contributions of this dissertation are summarized here:

- Propose an analog signal processing (ASP) algorithm named margin propagation (MP) as an efficient piece-wise linear (PWL) approximation technique to a "log-sum-exp" function and bring out its current-mode analog circuit implementation. Since MP algorithm involves only addition, subtraction and threshold operations and hence can be implemented using universal conservation principles like the Kirchoff's current law. Its analog circuit implementation is bias-scalable in different operation regions. As a result, the MP circuit can be adaptive to various application requirements since operating in different biasing condition is beneficial for different specifications.

- Synthesize bias-scalable analog computational circuits for a wide-range of linear and non-linear mathematical functions based on MP principle. We have implemented addition, subtraction, multiplication, division, power and polynomial computation, etc. with MP-based circuits. An important attribute of MP based synthesis is that all computations are performed in the logarithmic domain, as a result of which multiplications map into additions, divisions map into subtractions and power computations map into multiplications. This simplifies the implementation of many complicated non-linear functions. All the computational circuits can be operated in weak inversion, moderate inversion, and strong inversion regions.

- Design and implement an analog (32,8) MP-based LDPC decoder. One fascinating property is its capability of trading off BER performance with energy efficiency due to the tunable hyper parameter γ in MP algorithm, which has been verified by the (32,8) MP-based LDPC decoder prototype. The prototyped MP-based LDPC decoder can achieve an energy efficiency of 100nJ/bit while an optimal configuration can also deliver up to 3 dB improvement in BER compared to the benchmark min-sum LDPC decoder.

- Design and implement an analog energy-scalable MP-based support vector machine (SVM). The prototype stores 2052 SVM parameters into floating-gate memory array and is fully programmable. Due to the using of MP computation circuit, SVM is bias-scalable.

- Design and implement a novel current-input current-output logarithmic amplifier circuit Unlike the traditional transimpedance-based logarithmic amplifier circuit, our design based on translinear Ohm's law directly generates currents as a logarithmic function of the input current, which exhibits a dynamic range of 120dB and a temperature sensitivity of 230 ppm/K, while consuming less than 100nW of power.

- Design and implement a varactor based temperature compensation circuitry for floating-gate memory array By adapting the floating-gate capacitance, this design enables temperature dependent factors be effectively canceled, which solve the variation of current stored in floating-gate memory due to the temperature effect. A temperature sensitivity of 150 ppm/K is achieved.

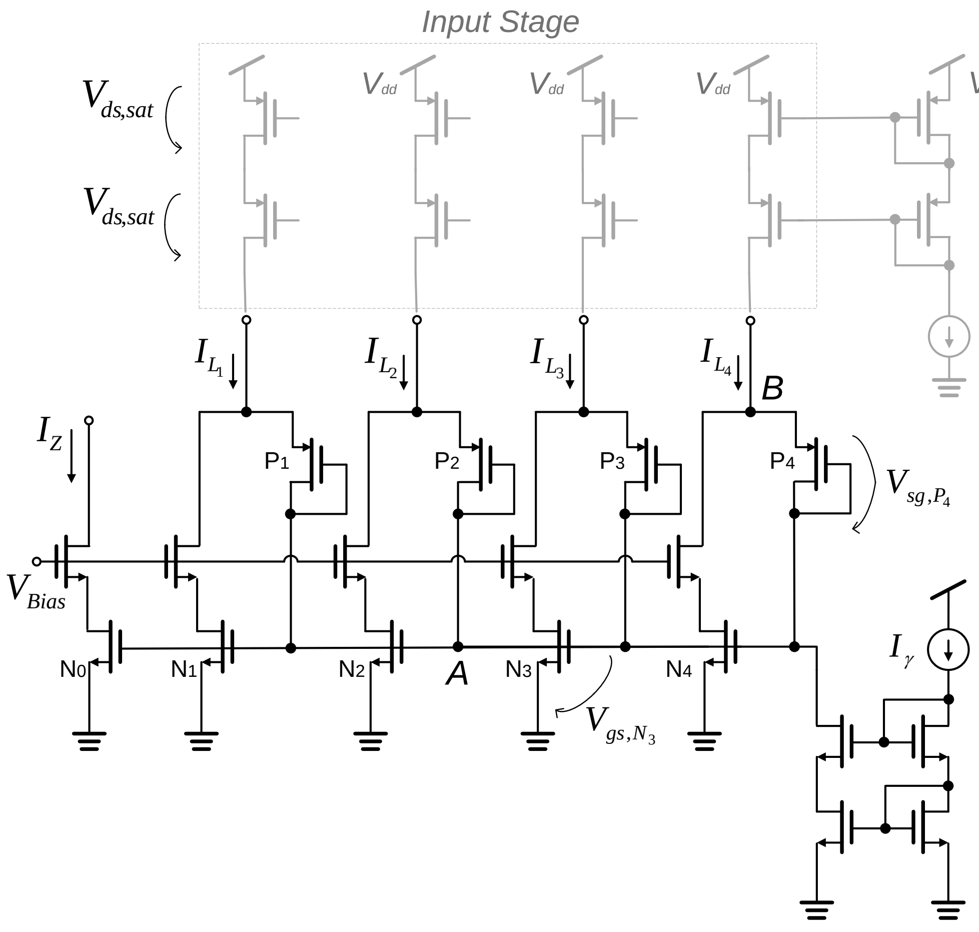


Figure 7.1: Improved MP circuit with cascode current mirrors

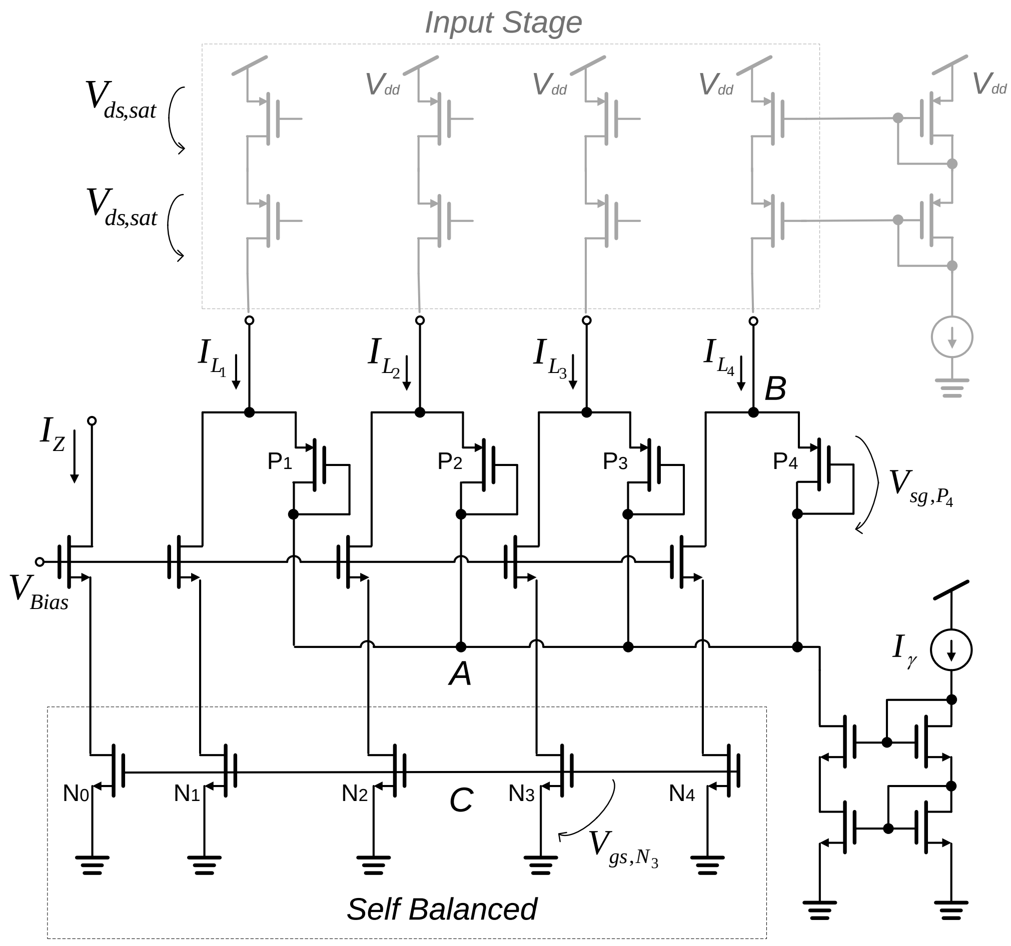


Figure 7.2: Decoupled MP circuit to prevent gate leakage

7.2 Future directions

Some aspects can be further improved or remain undiscovered for MP algorithm and MP circuits:

- Novel methodologies to implement MP algorithm. MP algorithm is scalable to different analog structures ranging from silicon transistors to microfluidic devices. This dissertation is adhered to current-mode since it is the easiest way of implementation. In future research, the physical parameters to represent value can be time (to improve energy efficiency), charge (to improve the precision), or any other characteristic parameters dependent on the application requirements.

- Ultra-low power implementation for MP algorithm. In this thesis, all the prototypes are capable of operating with current magnitude as low as nano-ampere. However, since MP algorithm is based on conservation law, it is possible to function even under the current magnitude of femto-ampere. For this improvement, specific circuit design techniques are supposed to be developed.

- Precision improvement for MP algorithm. In this thesis, one intuitive solution to improve the PWL approximation effect of MP to “log-sum-exp” function is proposed based on an experiment presented in chapter 3: to increase the number of operands involved in the MP computation. The accurate mathematical design for operands including the optimal number of operands and the value of the operands, is a promising topic for research.

7.2.1 Potential solutions for ultra-deep-submicron computational circuits using margin propagations

7.2.1.1 Cascode current mirrors

As has been discussed in [117], the performance of MP circuit is limited by the accuracy of "current mirrors" composed of N_0 through N_4 in Fig. 4.2. However, when MP circuit is scaled down to nano-scale, the inaccuracy in current replicating, which is caused by systematic mismatch (due to different V_{DS}s), is getting more pronounced.

Several literature have reported methodologies to improve the precision of ultra-deep-submicron current mirrors under low-voltage circumstances [118, 119, 120, 121, 122, 123]. Most of them are focused on cascoding scheme. Based on the cases of low-voltage high-precision current mirrors, the simplest method of improving the accuracy of nano-scale MP circuit is to utilize cascode technique. A basic nano-MP circuit with cascode current mirrors is displayed in Fig. 7.1. To achieve low-voltage implementation, it is also feasible to introduce other techniques reported in [118, 119, 120, 121, 122, 123].

7.2.1.2 Gate decoupling

The gate leakage is another concern for nano-scale MP circuits since it may cause the malfunction for the MP circuit. As we can see through Fig. 7.1, gate leakage is liable to occur at node A, which would affect the output current mirrored from $N1$ through N_4. To avoid the liability, a possible solution is proposed, which "decouples" the gates of $N1$ through N_4 from node A. The improved topology is demonstrated in Fig. 7.2. KCL still holds at node A; the difference lies in that the gates of transistors N_0 through N_4 are decoupled from node A and connected to node C. In this way, the gate voltages of N_1 through N_4, which used to be driven by currents through P_1-P_4, is self-balanced by the current through themselves. This is made possible due to the hot carrier effect in nano-scale circuit.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] A. Graupner, J. Schreiter, S. Getzlaff, and R. Sch$\ddot{u}$ffny, "CMOS image sensor with mixed-signal processor array," *IEEE J. Solid-State Circuits*, vol. 38, no. 6, pp. 948 – 957, Jun. 2003.

[2] P. Hasler, P. D. Smith, D. Graham, R. Ellis, and D. V. Anderson, "Analog floating-gate, on-chip auditory sensing system interfaces," *IEEE sensors journal*, vol. 5, no. 5, pp. 1027–1034, Oct. 2005.

[3] P. Hasler and D. V. Anderson, "Cooperative analog-digital signal processing," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 4, 2002, pp. IV–3972–IV–3975.

[4] G. Frantz, "Digital signal processor trends," *IEEE Micro*, vol. 20, no. 6, pp. 52 – 59, Dec. 2000.

[5] H. Suzuki, H. Takata, H. Shinohara, E. Teraoka, M. Matsuo, T. Yoshida, H. Sato, N. Honda, N. Masui, and T. Shimizu, "1.047Ghz, 1.2V, 90nm CMOS, 2-way VLIW DSP core using saturation anticipator circuit," in *2006 Symposium on VLSI Circuits Digest of Technical Papers*, 2006, pp. 152 – 153.

[6] T. S. Hall, C. M. Twigg, J. D. Gray, P. Hasler, and D. V. Anderson, "Large-scale field-programmable analog arrays for analog signal processing," *IEEE Trans. Circuits Syst. I*, vol. 52, no. 11, pp. 2298–2307, Nov. 2005.

[7] C. Huang, N. Lajnef, and S. Chakrabartty, "Calibration and characterization of self-powered floating-gate usage monitor with single electron per second operational limit," *IEEE Trans. Circuits Syst. I*, vol. 57, no. 3, pp. 556–567, Mar. 2010.

[8] F. Silveira, D. Flandre, and P. G. A. Jespers, “A floating-gate MOS learning array with locally computed weight updates,” *IEEE J. Solid-State Circuits*, vol. 31, no. 9, pp. 1314–1319, Sep. 1996.

[9] G. E. Moore, “Cramming more components onto integrated circuits,” *Electronics*, vol. 38, no. 8, pp. 114–117, apr. 1965.

[10] J. W. Cooley and J. W. Tukey, “An algorithm for the machine calculation of complex fourier series,” *Math. Computation*, vol. 19, pp. 297–301, Apr. 1965.

[11] A. Bandyopadhyay, J. Lee, R. W. Robucci, and P. Hasler, “MATIA: A programmable 80 μW/frame cmos block matrix transform imager architecture,” *IEEE J. Solid-State Circuits*, vol. 41, no. 3, pp. 663–672, Mar. 2006.

[12] N. Zhang, C. Teuscher, H. Lee, and B. Brodersen, “Architectural implementation issues in a wideband receiver using multiuser detection,” in *Proc. 36th Annu. Allerton Conf. Communication, Control, and Computing*, Sep. 1998, pp. 765 – 771.

[13] R. Sarpeshkar, “Efficient precise computation with noisy components: Extrapolating from an electronic cochlea to the brain,” Ph.D. dissertation, Dept. Comput. Neur. Syst., California Institute of Technology, Pasadena, CA, 1997.

[14] B. Gilbert, “Tranlinear circuits: a proposed classification,” *Electronics Letters*, vol. 11, no. 1, pp. 14–16, 1975.

[15] ——, “A high-accuracy analog multiplier,” in *Proc. IEEE ISSCC*, 1974, pp. 128 – 129.

[16] S. Hemati and A. Banihashemi, “Full CMOS min-sum analog iterative decoder,” jun. 2003, pp. 347 – 347.

[17] E. Seevinck and R. J. Wiegerink, “Generalized tranlinear circuit principle,” *IEEE J. Solid-State Circuits*, vol. 26, no. 8, pp. 1198–1202, Aug. 1991.

[18] W. Gai, H. Chen, and E. Seevinck, “Quadratic-translinear CMOS multiplier-divider circuit,” *Electron. Lett.*, vol. 33, no. 10, pp. 860–861, May. 1997.

[19] B. A. Minch, P. Hasler, and C. Diorio, “Multiple-input translinear element networks,” *IEEE Trans. Circuits Syst. II*, vol. 48, no. 1, pp. 20–28, Jan. 2001.

[20] R.W.Adams, “Filtering in the log domain,” in *63rd AES conf.*, New York, May 1979.

[21] J. Mulder, A. C. van der Woerd, W. A. Serdijn, and A. H. M. van Roermund, "An RMS-DC converter based on the dynamic translinear principle," *IEEE J. Solid-State Circuits*, vol. 32, no. 7, pp. 1146–1150, Jul. 1997.

[22] Y. P. Tsividis, "Externally linear integrators," *IEEE Trans. Circuits Syst. II*, vol. 45, no. 9, pp. 1181–1187, Sep. 1998.

[23] B. Gilbert, "Translinear circuits: A historical overview," *Analog Integrated Circuits and Signal Processing*, vol. 9, no. 2, pp. 95–118, 1996.

[24] B. A. Minch, "MOS translinear principle for all inversion levels," *IEEE Trans. Circuits Syst. II*, vol. 55, no. 2, pp. 121–125, Feb. 2008.

[25] E. A. Vittoz, "Analog VLSI signal processing: Why, where, and how?" *Analog Integr. Circuits Signal Process*, vol. 6, no. 1, pp. 27–44, 1994.

[26] B. A. Minch, "Synthesis of static and dynamic multiple-input translinear element networks," *IEEE Trans. Circuits Syst. I*, vol. 51, no. 2, pp. 409–421, Feb. 2004.

[27] A. G. Andreou, K. A. Boahen, P. O. Pouliquen, A. Pavasovic, R. E. Jenkins, and K. Strohbehn, "Current-mode subthreshold MOS circuits for analog VLSI neural systems," *IEEE Trans. Neural Netw.*, vol. 2, no. 2, pp. 205–213, Mar. 1991.

[28] A. G. Andreou and K. A. Boahen, "Translinear circuits in subthreshold MOS," *Analog Integrated Circuits and Signal Processing*, vol. 9, no. 2, pp. 141–166, 1996.

[29] P. Hasler, C. Diorio, and B. A. Minch, "A four-quadrant floating-gate synapse," in *Proc. IEEE International Symposium on Circuits and Systems*, vol. 3, Jun. 1998, pp. 29–31.

[30] C. Diorio, "Neurally inspired silicon learning: From synapse transistors to learning arrays," Ph.D. dissertation, Department of Electrical Engineering,California Institute of Technology, Pasadena, CA, 1997.

[31] B. A. Minch, "Multiple-input translinear element log-domain filters," *IEEE Trans. Circuits Syst. II*, vol. 48, pp. 29–36, Jan. 2001.

[32] ——, "Construction and transformation of multiple-input translinear element networks," *IEEE Trans. Circuits Syst. I*, vol. 50, pp. 1530–1538, Dec. 2003.

[33] J. Mulder, W. A. Serdijn, A. C. van der Woerd, and A. H. M. van Roermund, “A generalized class of dynamic translinear circuits,” *IEEE Trans. Circuits Syst. II*, vol. 48, no. 5, pp. 501–504, May. 2001.

[34] D. Frey, “Log-domain filtering: An approach to current-mode filtering,” *Proc. Inst. Electron. Eng.*, vol. 140, no. 6, pp. 406–416, Dec. 1993.

[35] ——, “A general class of current mode filters,” in *Proc. IEEE ISCAS*, vol. 2, 1993, pp. 1435–1438.

[36] ——, “Exponential state space filters: A generic current mode design strategy,” *IEEE Trans. Circuits Syst. I*, vol. 43, pp. 34–42, Jan. 1996.

[37] C. Toumazou and T. S. Lande, “Micropower log-domain filter for electronic cochlea,” *Electron. Lett.*, vol. 30, no. 22, pp. 1839–1841, Oct. 1994.

[38] M. Punzenberger and C. C. Enz, “Low-voltage companding currentmode integrators,” in *Proc. IEEE ISCAS*, 1995, pp. 2112–2115.

[39] D. Perry and G. W. Roberts, “Log-domain filters based on LC ladder synthesis,” in *Proc. IEEE ISCAS*, 1995, pp. 311–314.

[40] J. Mulder, A. C. van der Woerd, W. A. Serdijn, and A. H. M. van Roermund, “A current-mode companding $\sqrt{x}$-domain integrator,” *Electron. Lett.*, vol. 32, no. 3, pp. 198–199, Feb. 1996.

[41] W. A. Serdijn, M. Broest, J. Mulder, A. C. van der Woerd, and A. H. M. van Roermund, “A low-voltage ultra-low-power translinear integrator for audio filter applications,” *IEEE J. Solid-State Circuits*, vol. 32, pp. 577–581, Apr. 1997.

[42] W. A. Serdijn, J. Mulder, A. C. van der Woerd, and A. H. M. van Roermund, “A wide-tunable translinear second-order oscillator,” *IEEE J. Solid-State Circuits*, vol. 33, no. 2, pp. 195–201, Feb. 1998.

[43] A. Payne, A. Thanachayanont, and C. Papavassilliou, “A 150-MHz translinear phase-locked loop,” *IEEE Trans. Circuits Syst. II*, vol. 45, no. 9, pp. 1220–1231, Sep. 1998.

[44] T. Lin and A. Payne, “Translinear sinusoidal frequency tripler,” *Electron. Lett.*, vol. 38, no. 10, pp. 441–442, May. 2002.

[45] M. Gu, K. Misra, H. Radha, and S. Chakrabartty, "Sparse decoding of low density parity check codes using margin propagation," in *Proc. of IEEE Globecom*, Nov. 2009.

[46] ——, "Sparse decoding of low-density parity-check codes based on margin propagation," *IEEE Trans. Commun.*, 2010, submitted.

[47] W. E. Ryan, "An introduction to LDPC codes," in CRC *Handbook for Coding and Signal Processing for Recoding Systems*. Taylor, CRC Press, 2004, ch. 36.

[48] F. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 498–519, Feb. 2001.

[49] F. shi Lai and C.-F. E. Wu, "A hybrid number system processor with geometric and complex arithmetic capabilities," *IEEE Trans. Comput.*, vol. 40, no. 8, pp. 952–962, Aug. 1991.

[50] E. I. Chester and J. N. Coleman, "Matrix engine for signal processing applications using the logarithmic number system," in *Proc. IEEE Application-Specific Syst., Architectures and Processors*, Jul. 2002, pp. 315–324.

[51] J. N. Coleman, E. I. Chester, C. I. Softley, and J. Kadlec, "Arithmetic on the European logarithmic microprocessor," *IEEE Trans. Comput.*, vol. 49, no. 7, pp. 702–715, Jul. 2000.

[52] H. Kim, B.-G. Nam, J.-H. Sohn, J.-H. Woo, and Hoi-Jun, "A 231-MHz, 2.18-mW 32-bit logarithmic arithmetic unit for fixed-point 3-D graphics system," *IEEE J. Solid-State Circuits*, vol. 41, no. 11, pp. 2373–2381, Nov. 2006.

[53] B.-G. Nam, H. Kim, and H.-J. Yoo, "A low-power unified arithmetic unit for programmable handheld 3-D graphics systems," *IEEE J. Solid-State Circuits*, vol. 42, no. 8, pp. 1767–1778, Aug. 2007.

[54] K. H. Abed and R. E. Siferd, "CMOS VLSI implementation of a low-power logarithmic converter," *IEEE Trans. Comput.*, vol. 52, no. 11, pp. 1421–1433, Nov. 2003.

[55] S. L. SanGregory, C. Brothers, and D. Gallagher, "A fast,low-power logarithm approximation with CMOS VLSI implementation," in *Proc. IEEE Midwest Symp. Circuits and Systems*, Aug. 1999, pp. 388–391.

[56] Y.-H. Lam, W.-H. Ki, and C.-Y. Tsui, "Integrated low-loss CMOS active rectifier for wirelessly powered devices," *IEEE Trans. Circuits Syst. II*, vol. 53, no. 12, pp. 1378–1382, Dec. 2006.

[57] A. Gore, S. Chakrabartty, S. Pal, and E. C. Alocilja, "A multichannel femtoampere-sensitivity potentiostat array for biosensing applications," *IEEE Trans. Circuits Syst. I*, vol. 53, no. 11, pp. 2357–2363, Nov. 2006.

[58] T. Serrano-Gotarredona, B. Linares-Barranco, and A. G. Andreou, "A general translinear principle for subthreshold MOS transistors," *IEEE Trans. Circuits Syst. I*, vol. 46, no. 5, pp. 607–616, May. 1999.

[59] B. Razavi, *Design of analog* CMOS *integrated circuits*. McGraw-Hill Science/Engineering/Math, Aug. 2000.

[60] P. R. Kinget, "Device mismatch and tradeoffs in the design of analog circuits," *IEEE J. Solid-State Circuits*, vol. 40, no. 6, pp. 1212–1224, Jun. 2005.

[61] S. Chakrabartty and G. Cauwenberghs, "Sub-microwatt analog VLSI trainable pattern classifier," *IEEE J. Solid-State Circuits*, vol. 42, no. 5, pp. 1169–1179, May. 2007.

[62] K. Kang and T. Shibata, "An on-chip-trainable gaussian-kernel analog support vector machine," *IEEE Trans. Circuits Syst. I*, vol. 57, no. 7, pp. 1513–1524, Jul. 2010.

[63] S. Chakrabartty and G. Cauwenberghs, "Gini-support vector machine: Quadratic entropy based multi-class probability regression," *Journal of Machine Learning Research*, vol. 8, pp. 813–839, 2007.

[64] R. G. Gallager, *Low Density Parity-Check Codes*, R. G. Gallager, Ed. MIT Press, Cambridge, MA, 1963.

[65] E. T. S. I. (ETSI), "Digital Video Broadcasting (DVB) second generation framing structure, channel coding and modulation systems for broadcasting, interactive services, news gathering and other broadband satellite applications; tm 2860rl dvbs2-74r8," www.dvb.org.

[66] B. Bangerter and et al., "Wireless technologies: High-throughput wireless LAN air interface," http://www.intel.com/technology/itj.

[67] Y. Rashi and et al., "LDPC: Efficient alternative FEC for the TFI-OFDM PHY proposal," http://www.grouper.ieee.org/groups/802/15/pub.

[68] R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inf. Theory*, vol. 27, no. 5, pp. 533–547, Sep 1981.

[69] J. Hagenauer, E. Offer, and L. Papke, "Iterative decoding of block and convolutional codes," *IEEE Trans. Inf. Theory*, vol. IT-42, no. 3, pp. 429–445, Mar. 1996.

[70] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Trans. Inf. Theory*, vol. 45, no. 2, pp. 399–431, Mar. 1999.

[71] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman, "Improved low-density parity-check codes using irregular graphs," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 585–598, Feb. 2001.

[72] M. M. Mansour and N. R. Shanbhag, "A 640 Mbps 2048-bit programmable LDPC decoder chip," *IEEE J. Solid-State Circuits*, vol. 41, pp. 684–698, Mar. 2006.

[73] A. J. Blanksby and C. J. Howland, "A 690-mW 1-Gb/s 1024-b, rate-1/2 low-density parity-check code decoder," *IEEE J. Solid-State Circuits*, vol. 37, no. 3, Mar. 2002.

[74] A. Darabiha, A. C. Carusone, and F. R. Kschischang, "A 3.3-Gbps bit-serial block-interlaced min-sum LDPC decoder in 0.13-μm CMOS." IEEE Custom Integrated Circuits Conference, Sep. 2007, pp. 459–462.

[75] T. Brandon, R. Hang, G. Block, V. C. Gaudet, B. Cockburn, S. Howard, C. Giasson, K. Boyle, P. Goud, S. S. Zeinoddin, A. Rapley, S. Bates, D. Elliott, and C. Schlegel, "A scalable LDPC decoder ASIC architecture with bit-serial message exchange," *Elsevier Integration-the VLSI Journal*, vol. 41, no. 3, pp. 385–398, May. 2008.

[76] J. Hagenauer and M. Winklhofer, "The analog decoder," in *Proceedings of IEEE International Symposium on Information Theory (ISIT '98)*, Cambridge, Mass, USA, Aug. 1998, p. 145.

[77] S. Hemati, A. Banihashemi, and C. Plett, "An 80-Mb/s 0.18-μm CMOS analog min-sum iterative decoder for a (32,8,10) LDPC code," *IEEE J. Solid-State Circuits*, vol. 41, pp. 2531–2540, Nov. 2006.

[78] C. Winstead, N. Nguyen, V. Gaudet, and C. Schlegel, "Low-voltage CMOS circuits for analog iterative decoders," *IEEE Trans. Circuits Syst. I*, vol. 53, pp. 829–841, Apr. 2006.

[79] V. Gaudet, "Energy efficient circuits for LDPC decoding." CMOS Emerging Technologies, 2007.

[80] A. Liveris, Z. Xiong, and C. Georghiades, “Compression of binary sources with side information at the decoder using LDPC codes,” *Communications Letters, IEEE*, vol. 6, no. 10, pp. 440–442, Oct 2002.

[81] M. Sartipi and F. Fekri, “Source and channel coding in wireless sensor networks using ldpc codes,” *Sensor and Ad Hoc Communications and Networks, 2004. IEEE SECON 2004. 2004 First Annual IEEE Communications Society Conference on*, pp. 309–316, Oct. 2004.

[82] M. Potluri, S. Chilumuru, S. Kambhampati, and K. Namuduri, “Distributed source coding using non-binary ldpc codes for sensor network applications,” *Information Theory, 2007. CWIT ’07. 10th Canadian Workshop on*, pp. 105–109, June 2007.

[83] D. MacKay and R. Neal, “Near shannon limit performance of low density parity check codes,” *Electronics Letters*, vol. 33, no. 6, pp. 457–458, Mar 1997.

[84] S.-Y. Chung, J. Forney, G.D., T. Richardson, and R. Urbanke, “On the design of low-density parity-check codes within 0.0045 db of the shannon limit,” *Communications Letters, IEEE*, vol. 5, no. 2, pp. 58–60, Feb 2001.

[85] ITRS, “International technology roadmap for semiconductors,” http://www.itrs.net, 2010.

[86] T. Richardson and R. Urbanke, “The capacity of low-density parity check codes under message-passing decoding,” *IEEE Trans. Inf. Theory*, vol. 47, pp. 599–618, 2001.

[87] T. Richardson, M. Shokrollahi, and R. Urbanke, “Design of capacity approaching irregular low-density parity-check codes,” *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 619–637, Feb. 2001.

[88] D. J. C. MacKay, *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, 2003, ch. 6, pp. 557–572, http://www.inference.phy.cam.ac.uk/mackay/itila/.

[89] D. L. Donoho, “Compressed sensing,” *IEEE Trans. Inf. Theory*, vol. 52, no. 4, Apr. 2006.

[90] L. Mancera and J. Portilla, “L0-norm-based sparse representation through alternate projections,” in *13th International Conference on Image Processing (ICIP), 2006*. Atlanta, GE (USA), October 2006.

[91] S.-Y. Chung, T. Richardson, and R. Urbanke, “Analysis of sum-product decoding of low-density parity-check codes using a gaussian approximation,” *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 657–670, Feb. 2001.

[92] P. Rusmevichientong and B. Van Roy, "An analysis of belief propagation on the turbo decoding graph with gaussian densities," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 745–765, Feb 2001.

[93] A. Amraoui, "Ldpcopt," http://ipgdemos.epfl.ch/ldpcopt/.

[94] O. C. Ibe, *Fundamentals of applied probability and random processes*, 1st ed. Academic Press, 2005.

[95] C.-H. Lin and K. Bult, "A 10-b, 500-MSample/s CMOS DAC in 0.6 mm^2," *IEEE J. Solid-State Circuits*, vol. 33, no. 12, pp. 1948 – 1958, Dec. 1998.

[96] V. Gaudet and G. Gulak, "A 13.3-Mb/s 0.35μm CMOS analog turbo decoder IC with a configurable interleaver," *IEEE J. Solid-State Circuits*, vol. 38, no. 11, pp. 2010–2015, Nov. 2003.

[97] A. Darabiha, A. C. Carusone, and F. R. Kschischang, "Power reduction techniques for LDPC codes," *IEEE J. Solid-State Circuits*, vol. 43, no. 8, pp. 1835 – 1845, Aug. 2008.

[98] Z. Zhang, V. Anantharam, M. J. Wainwright, and B. Nikolić, "An efficient 10GBASE-T ethernet LDPC design with low error floors," *IEEE J. Solid-State Circuits*, vol. 45, no. 4, pp. 843 – 855, Apr. 2010.

[99] C. Winstead, J. Dai, S. Yu, C. Myers, R. Harrison, and C. Schlegel, "CMOS analog MAP decoder for (8,4) Hamming code," *IEEE J. Solid-State Circuits*, vol. 39, no. 1, pp. 122–131, Jan. 2004.

[100] D. Vogrig, A. Gerosa, A. Neviani, A. Amat, G. Montorsi, and S. Benedetto, "A 0.35-μm CMOS analog turbo decoder for the 40-bit rate 1/3 UMTS channel code," *IEEE J. Solid-State Circuits*, vol. 40, pp. 753–762, Mar. 2005.

[101] E. Osuna, R. Freund, and F. Girosi, "Training support vector machines: An application to face detection," *Comput. Vis. Pattern Recogn.*, pp. 130–136, 1999.

[102] R. Genov and G. Cauwenberghs, "Charge-mode parallel architecture for matrix-vector multiplication," *IEEE Trans. Circuits Syst. II*, vol. 48, no. 10, pp. 930–936, 2001.

[103] S. Chakrabartty and G. Cauwenberghs, *Sub-Microwatt Analog* VLSI *Support Vector Machine for Pattern Classification and Sequence Estimation.* Cambridge: MIT Press, 2005, ch. Adv. Neural Information Processing Systems (NIPS2004), p. 17.

[104] ——, “Sub-microwatt analog VLSI trainable pattern classifier,” *IEEE J. Solid-State Circuits*, vol. 42, no. 5, pp. 1169–1179, May. 2007.

[105] B. Boser, I. Guyon, and V. N. Vapnik, “A training algorithm for optimal margin classifier,” in *Proc. 5th Annu. ACM Workshop Computational Learning Theory*, 1992, p. 14452.

[106] A. Bandyopadhyay, G. J. Serrano, and P. Hasler, “Adaptive algorithm using hot-electron injecton for programming analog computational memory elements within 0.2% of accuracy over 3.5 decades,” *IEEE J. Solid-State Circuits*, vol. 41, no. 9, pp. 2107–2114, Sep. 2006.

[107] W. L. Barber and E. R. Brown, “A true logarithmic amplifier for radar IF applications,” vol. 15, no. 3, pp. 291–295, Jun. 1980.

[108] A. Basu, R. W. Robucci, and P. E. Hasler, “A low-power, compact, adaptive logarithmic transimpedance amplifier operating over seven decades of current,” *IEEE Trans. Circuits Syst. I*, vol. 54, no. 10, pp. 2167–2177, Oct. 2007.

[109] C. D. Holdenried, J. W. Haslett, J. G. McRory, R. D. Beards, and A. J. Bergsma, “A DC-4-GHz true logarithmic amplifier: Theory and implementation,” *IEEE J. Solid-State Circuits*, vol. 37, no. 10, pp. 1290–1299, Oct. 2002.

[110] E. Vittoz and J. Fellrath, “CMOS analog integrated circuits based on weak inversion operations,” *IEEE J. Solid-State Circuits*, vol. 12, no. 3, pp. 224–231, Jun. 1977.

[111] C. Huang, P. Sarkar, and S. Chakrabartty, “Rail-to-rail, thermal-noise limited, linear hot-electron injection programming of floating-gate voltage bias generators at 13-bit resolution,” *IEEE J. Solid-State Circuits*, vol. 46, no. 11, pp. 2685–2692, Nov. 2011.

[112] R. L. Bunch and S. Raman, “Large-signal analysis of MOS varactors in CMOS-G$_m$ LC VCOs,” *IEEE J. Solid-State Circuits*, vol. 38, no. 8, pp. 1325–1332, Aug. 2003.

[113] T. Delbruck and D. Oberhoff, “Self-biasing low power adaptive photoreceptor,” in *Proc. IEEE Int. Symp. Circuits and Systems*, vol. 4, May. 2004, pp. 844–847.

[114] D. H. K. Hoe and D. B. Ribner, “An auto-ranging photodiode preamplifier with 114 db dynamic range,” *IEEE J. Solid-State Circuits*, vol. 31, no. 2, pp. 187–194, Feb. 1996.

[115] M. J. Hayes, “A nonlinear optical preamplifier for sensing applications,” *IEEE Trans. Circuits Syst. I*, vol. 49, no. 1, pp. 1–9, Jan. 2002.

[116] A. Basu, S. Brink, C. Schlottmann, S. Ramakrishnan, C. Petre, S. Koziol, F. Baskaya, C. M. Twigg, and P. Hasler, "A floating-gate-based field-programmable analog array," *IEEE J. Solid-State Circuits*, vol. 45, no. 9, pp. 1781–1794, Sep. 2010.

[117] M. Gu and S. Chakrabartty, "Synthesis of bias-scalable CMOS analog computational circuits using margin propagation," *IEEE Trans. Circuits Syst. I*, vol. 59, no. 2, pp. 243–254, Feb. 2012.

[118] E. S*ä*ckinger and W. Guggenb*ü*hl, "A high swing, high impedance MOS cascode circuit," *IEEE J. Solid-State Circuits*, vol. 25, no. 1, pp. 289–298, Feb. 1990.

[119] T. Itakura and Z. Czarnul, "High output-resistance CMOS current mirrors for low-voltage applications," *IEICE Trans. Fundamentals*, vol. E80-A, no. 11, pp. 230–232, Jan. 1997.

[120] F. You, S. H. K. Embabi, J. F. Duque-Carrillo, and E. S*á*nchez-Sinencio, "An improved tail current source for low voltage applications," *IEEE J. Solid-State Circuits*, vol. 32, no. 8, pp. 1173–1180, Aug. 1997.

[121] T. Serrano and B. Linares-Barranco, "The active-input regulated cascode current-mirror," *IEEE Trans. Circuits Syst. I*, vol. 41, no. 6, pp. 464–467, Jun. 1994.

[122] J. Ram*í*rez-Angulo, R. G. Carvajal, and A. Torralba, "Low supply voltage high-performance CMOS current mirror with low input and output voltage requirements," *IEEE Trans. Circuits Syst. II*, vol. 51, no. 3, pp. 124–129, Mar. 2004.

[123] X. Zhang and E. I. El-Masry, "A regulated body-driven CMOS current mirror for low-voltage applications," *IEEE Trans. Circuits Syst. II*, vol. 51, no. 10, pp. 571–577, Oct. 2004.