ACS: Asynchronous Computing with Streams

Patricia Gonzalez-Guerrero

A Dissertation Presented to the Graduate Faculty

of the University of Virginia in Candidacy for the Degree of

Doctor of Philosophy

University of Virginia

December, 2019

Abstract

The internet of things (IoT) is shaping the way we live and the evolution of society. Recent prototypes demonstrated continuous monitoring of fitness activities, health indicators, environmental factors and industrial processes, with sensors that operate under restrictive power budgets ($\sim$100-200μW) harvested from piezo, thermo, solar or even bioelectric sources. These applications are enabled by aggressively lowering the power supply (>0.3V), sampling (>1KHz) and processing rates (>200KHz). However, those constraints are at odds with the next generation of IoT devices. We expect the IoT sensor to interact with us, follow voice commands, stream video, recognize faces and even emotions in real time, all powered by machine learning algorithms. More importantly, we expect all these demanding applications, under the restrictive power budget and form-factor of a typical IoT sensor.

To address the challenge of high performance and low power at the IoT sensor node, we propose asynchronous computing with streams (ACS), a novel paradigm that leverages the advantages of previous computing with streams approaches such as synchronous stochastic computing (SSC), while addressing its drawbacks. In SSC data is encoded in streams of bits decreasing circuit complexity, generating savings in power and area. However, the high cost of generating these streams and the long computation latency eclipses any potential savings. To address these drawbacks, we propose a new efficient asynchronous architecture for the IoT sensor node with end to end processing on streams. This approach, avoids costly binary-stream-binary conversion and waives the need of clock generation and distribution. In this dissertation we explore asynchronous stochastic computing (ASC) and stochastic computing with $\Sigma\Delta$ streams, both paradigms falling under the umbrella of ACS. We first develop a theoretical framework based on 2 state continuous time Markov Chains to model the dynamics of asynchronous stochastic streams and estimate the computing error due to random fluctuation. Furthermore, we address the need for efficient stream generation circuits by exploring asynchronous sigma delta modulators as stream generator for ACS. We also propose an end to end asynchronous architecture including interfaces with the memory and the IO. Finally, To demonstrate the effectiveness of ASC, we implement 1) an image processing algorithm: Gamma correction, 2) A signal processing algorithm: Fast Fourier Transform and 3) a machine learning algorithm: Decision Trees.

Acknowledgments

This is by far my favorite part of this book. This PhD left me with two lessons: (1) dreams do come true and (2) they do come true thanks to the paradoxical crazy situations that bring us together in one way or another.

To my adviser, Professor Mircea R. Stan, I of course have to thank him all the guidance and financial support. However, the most important lesson I take from him is his unwavering faith in humanity and his kindness towards all. These two qualities are hard to find in a culture where competition is pervasive. I am sure that if all of us acted more like him, the world would be a better place. To my committee, Professors Lorena Anghel, Nikhil Shukla, Steven M. Bowers and Stephen G. Wilson, I am grateful for the time you put into the meetings and calls to answer all my questions. I would like to specially thank Professor Wilson for all the time we spent thinking about $\Sigma\Delta$ streams error. It was a joy and a blessing to observe and learn from a person with such attention to detail, to carefully analyze every aspect of a problem. More importantly his commitment to guide and help students is unique and I am extremely thankful for this.

To the current members of my research lab: Tommy Tracy II, Sergiu Mosanu, Vaibhav Verma, Junhan Han, Nazmus Sakib, Robert E. West II *a.k.a.* Trey, Ceylan Morgul, Rahul Sreekumar, Yunfei Gu and Mateja Putic, and the ones that already graduated, Xinfei Guo, Pi, and Alec Roelnek. I thank you all for the laughs in the happy moments, but also the laughs in the really stressful moments which is only possible with such a great bunch of people. To Xinfei Guo thanks for enlighten me about how to write papers, create plots, tables and many other tricks. To Tommy Tracy II thanks for being so supportive and hop on some of my crazy ideas. To Alec Roelnek thanks for being always so extremely funny. To Vaibhav thanks for making us do push-ups in the middle of the day. To Rahul, thanks for all the help, support and commitment to the ACS cause.

To my parents Nohora Guerrero and Enrique Gonzalez, I thank you for every day you went to work while extremely ill so we can have an education. Thank you for giving us the hope of a different future even-though there were not many opportunities. Thank you for breaking the chains of ignorance. Thank you for dreaming for us and with us. I am really-truly standing on the shoulders of giants. To my siblings Kike and Dani Gonzalez thanks for being my engine, my fuel and my reason.

To the little family from the land of magical-realism: Juliana, Jaidy, Edna, Yuly, Mariana, Paula, Natali, and Dona Berta, thanks for the good memories and funny stories in New-year, Christmas, Thanksgiving, Graduations, Accomplishments, Birthday parties and so on.

To Olga, Matt and little Hanna thank you for all the yoga, hiking and camping adventures. To Gerardo, Kristy and little Sophi thank you for the afternoons of food, wine and laughing. To Ritu for her love when I needed it the most. To Andrea King for all the care and fun moments around delicious food.

My heart cherishes all of you.

Last but not least, I am eternally grateful to Carlos A. Polanco. It was your support, patience, and commitment throughout all the difficulties of this PhD journey which made it possible. Thanks for all the advice, constructive and objective criticism at every step of this process. Thank you for your example of unwavering discipline and tenacity. I am so lucky I had you throughout all this and I can't wait for the next crazy adventure we decide to embark on.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Gaines [10] introduced SC around the 1960s as an efficient way to deal with algorithms that require large arrays of elements such as adders and multipliers. At that time, the size of the arrays were severely limited by area and power consumption. More than 50 years later these limitations have become relevant again with big data driven technologies, like machine learning (ML) and extremely constrained energy-scavenging systems that will shape the Internet of Things (IoT) [13], smartdust [14] [15], and edge intelligence (EI) [16] .

Smartdust, IoT, and EI, terms coined around 1990, 1999 and 2019 respectively, share a common base: the deployment of billions of interconnected wireless sensor nodes with applications in the military, health management, smart farming and environmental monitoring among many others. Around 2003, news were spreading that *by 2013 there would be one trillion of interconnected autonomous sensors* [17] [18]. It is hard to argue that we had hit that goal by today (2019), and given the technical challenges that still need to be addressed, we keep on moving the target date to some years in the future. Similarly, even-though SSC yields spectacular power savings in the computing units, major drawbacks end-up increasing the overall cost of a SSC-based system to the point it becomes impractical.

1.1.1 Ultra-low power sensors challenges

Smartdust was originally envisioned as billions of autonomous, tiny micro-electro-mechanical (MEM) $1mm \times 1mm \times 1mm$ devices, a.k.a. motes, featuring basic computation resources and the ability to detect and communicate with other motes in the vicinity. Individual motes would self-organize into *ad-hoc* computer networks capable of relaying data using wireless technology [19].

Although, several technical challenges are still to be addressed to deploy the smartdust technology, advancements in fabrication, integration and batteries have already enabled the IoT: a conglomeration of all kinds of objects with the ability to connect to the internet such as mobile-phones, smart refrigerators or coffee makers, self-driving cars and fitness monitors, among others. At first sight, the main difference between a mote and an IoT-sensor is a form-factor, way larger than $1mm \times 1mm \times 1mm$ [14] [15]. However, the evolution of IoT technology is uncovering the most important difference: the data flow across the IoT ecosystem. Typically an IoT-sensor is *dumb*, meaning that it only collects and transmits health, location, or environmental data, just to name a few examples. Advanced data processing relies on the intelligence of the Cloud, where data is aggregated, storaged and analyzed enabling patterns recognition, or action triggering such as turn off lights, lock a door, or emergency protocols. By 2017, we started to observe the IoT relatively "*dumb*" [16], relatively large and energy unaware sensors slowly entering the market with products such as virtual assistants or fitness monitors. Those IoT sensors don't have form-factor limitations, power consumption restrictions, particularly demanding processing requirements or require a person to periodically replace/recharge a battery (1 day/1 week). As the IoT looks into other applications where the energy available is unreliable and scarce, the form factors are reduced for accessibility or wearability and the cost of constantly replacing/recharging batteries in embedded systems deployed in masses becomes unfeasible, the “dumb" sensor model can't keep up. Challenges such as: 1) applications requiring low latency, that is, real-time conversion of data to information, 2) energy scarcity for the sensors 4) privacy and 5) security still remain to be addressed to achieve the mark of one trillion autonomous sensors wirelessly connected [16].

Moving closer to the original mote, and addressing the challenges of current IoT solutions, the scientific community is moving towards EI which stresses the importance of two main features for wireless sensor nodes:

1. Energy harvesting capabilities: Replacing or recharging batteries for one trillion devices deployed in the field is not a practical or economical assumption. Thus the node should be designed to operate under harvested, intermittent and restricted energy conditions.

2. Adding “intelligence”, which in this context means processing capabilities, to the sensor node addresses some of the IoT challenges in the following ways:

- **Energy scarcity:** One of the most expensive operations in a wireless sensor is the transmission of raw-data which can take up to 60% of the power budget in a node [20] [21]. Thus, adding local capabilities to transform raw data into information for classification, decision making, context detection or analytics, dramatically reduces the volume of data that needs to be transmitted to a central node. Adding intelligence to the node assume that the cost of local processing is less than that of wireless transmission of raw data.
- **Real time:** Many applications require responsive local decision-making [22] and thus, real-time local computing. For example in cases where the required responsiveness does not tolerate the latency associated with communication with the Cloud for decision-making.
- **Security and Privacy:** For many applications, locally collected data are best secured by keeping that data local and not sending it to the Cloud [16]. For example, Apple's Siri virtual assistant performs keyword spotting locally to detect the trigger phrase "hey Siri" [23].

Increasing the computing power while operating from scavenged energy looks like two opposite directions. Recent work showed that key spotting, as the one used by Apple's Siri, using machine learning algorithms requires 2W and 431mJ of power and energy respectively [24], while the power budget for a battery-less IoT sensor is no more than 200μW. This is, 10K times less than the required power to run the ML algorithm on a Raspberry Pi [25].

1.1.2 Ultra-low power sensors state of the art

To achieve ultra-low-power operation, current approaches heavily rely on reducing the power supply to 0.3V-0.45V where yield, reliability and performance get compromised to meet the power constrains. Table 1.1 summarizes examples of motes/IoT-sensors showcased at the International Solid-State Circuits Conference (ISSCC) in 2018 and 2019. A remarkable example of low power consumption is the implantable glucose monitoring system that senses, digitize and transmit information operating at 0.3V consuming only 1.15μW, however, this sensor does not include any processing capabilities [26]. Processing capabilities are showcased in the general purpose processor that operate at 0.45V with energy harvested from a solar cell, however the frequency of operation is limited to only 4Hz [27]. As the sensor increases the processing capabilities and include the power consumption of the transducers (image processing and multi-sensor platform in Table 1.1) the power consumption jumps to the order of mW. These examples discussed above have one common feature, data representation is base 2 digital binary, moreover, the image processing, the multi-sensor, and the general purpose platforms follow a very traditional Von Neumann architecture.

In contrast, Table 1.1 shows the metrics of performance for a system designed for photoplethysmography (PPG) [28] and speech processing [29] with a processing power of $44\mu W$ and $1\mu W$ respectively. In both cases the key feature for low power operation is enabled by the data representation. Contrary to Von Neumann architectures, in this case periodic sampling is substituted by signal dependent schemes, where sampling is triggered irregularly and occurs when a specified event, defined by its amplitude variation is detected [30]. That is, the analog to digital converter (ADC) is replaced by an analog to events converter, originating architectures with very different data-flows. These last sensors examples, show that to achieve the final $1\mu W$ power consumption budget we may need to overhaul the sensor architecture and re-think the data to information flow.

#	Purpose	Building Components	Processor	Tech.	Area	Memory size	Vdd [V]	Freq. [Hz]	Power
1 [27]	General	MCU, Energy harvesting (solar), Battery	MSP430	180nm	5.33mm^2	2KB	0.2-1.2	2	595pW 0.45V
2 [31]	Image Processing	MCU, CNN accelerator, crypto-engine, AFE, Wup RX, PMU, Battery, voltage regulator, BLE radio, CMOS camera, Energy harvesting (solar)	Generic	FinFET 14nm	33mm$\times$33mm	512kB	0.55-1.7	200k-250M	0.2mW iddle, 25mW Peak
3 [26]	Glucose monitoring	DS-ADC + Clock + AFE + TX + FIFO	No	65nm	0.7mm$\times$1.3mm	-	0.3	256K	1.15 μW
4 [29]	Speech processing	LNA, 16 filter banks, BNN, microphone	No	180nm	1.6mm$\times$1.5mm	No	0.55-0.6	-	1μW
5 [28]	Pulseoximetry	AFE, a PPG-to-clock converter, digital Heart Beat Locked Loop	No	180nm	-	-	3.3	-	44μW
6 [32]	Body multi-sensor node	Accelerators, Bluetooth, AFEs, Clock generation, transducers, Digital IO	ARM cortex	55nm	4.3mm$\times$4.3mm	192kB, Flash	0.8-2.8	-	2mW

Table 1.1: Summary of metrics of performance for IoT-sensors/motes reported in ISSCC in 2018 and 2019.

1.1.3 SSC potential to enable ultra-low power computing

SSC is a candidate to enable high-performance low-power consumption processing capabilities for the energy-constrained wireless node. In SC, a number g is associated with the probability p of an event to occur. p is the probability of having a high (P) voltage on a stream of bits that can only take two voltages high (P) or low (N). The streams are encoded in either unipolar ($N = 0$ and $P = 1$) or bipolar ($N = -1$ and $P = 1$) representations. To properly process information using SC, the input streams should be uncorrelated or independently generated. Under that assumption, a stream multiplier can be implemented with a simple AND gate (Figure 1-1a) or an adder with a multiplexer. Figure 1-1c shows an example for multiplication. The stream X is generated with probability $p_x = 0.5$ and stream Y with probability $p_y = 0.5$. The output is the stream C with probability $p_c = 0.25$. Given the minimum footprint of its computing units SC has the potential to enable on node high-performance, low-power data processing for energy constrained systems.

Stochastic computing is by definition a synchronous approach where each bit is the result of a random experiment happening at the positive edge of the clock. Several researchers have shown SSC applications demonstrating more than 90% of logic simplification and robustness in the presence of bit errors [2] [33] [34]. Although very appealing, SSC has three major drawbacks: (1) All SSC approaches in literature focus on replacing the processing core with an stochastic computing core while keeping the remaining blocks of the system the same. To do this all the digital binary data is converted to streams using a Digital to Stochastic Converter, *a.k.a* Stochastic Number Generator (SNG), and Stochastic to Digital Converters. Figure 1-1d shows the block diagram of a typical SNG. A Linear Feedback Shift Register (LFSR) is used to generate pseudo-random numbers (rn) that are compared with the digital input p_g. The output of the comparator is P if p_g is larger than rn otherwise is N. In this approach, stochastic number generation can use up to 80% of the total area and 67%-92% of the total power consumption in a SC system [2]. (2) Since the streams are randomly generated, these must be long enough to obtain a minimum level of accuracy. Long streams have a negative effect in latency and energy consumption. (3) A SSC based sensor requires the generation and distribution of a very fast clock to make up for the latency penalty incurred by the long streams of bits. The combination of these factors results in latency, power and energy metrics 48X, 1.3X and 457X larger than the binary counterpart [35]. These reasons make stochastic computing as we know it, very promising but impractical for the large majority of applications.

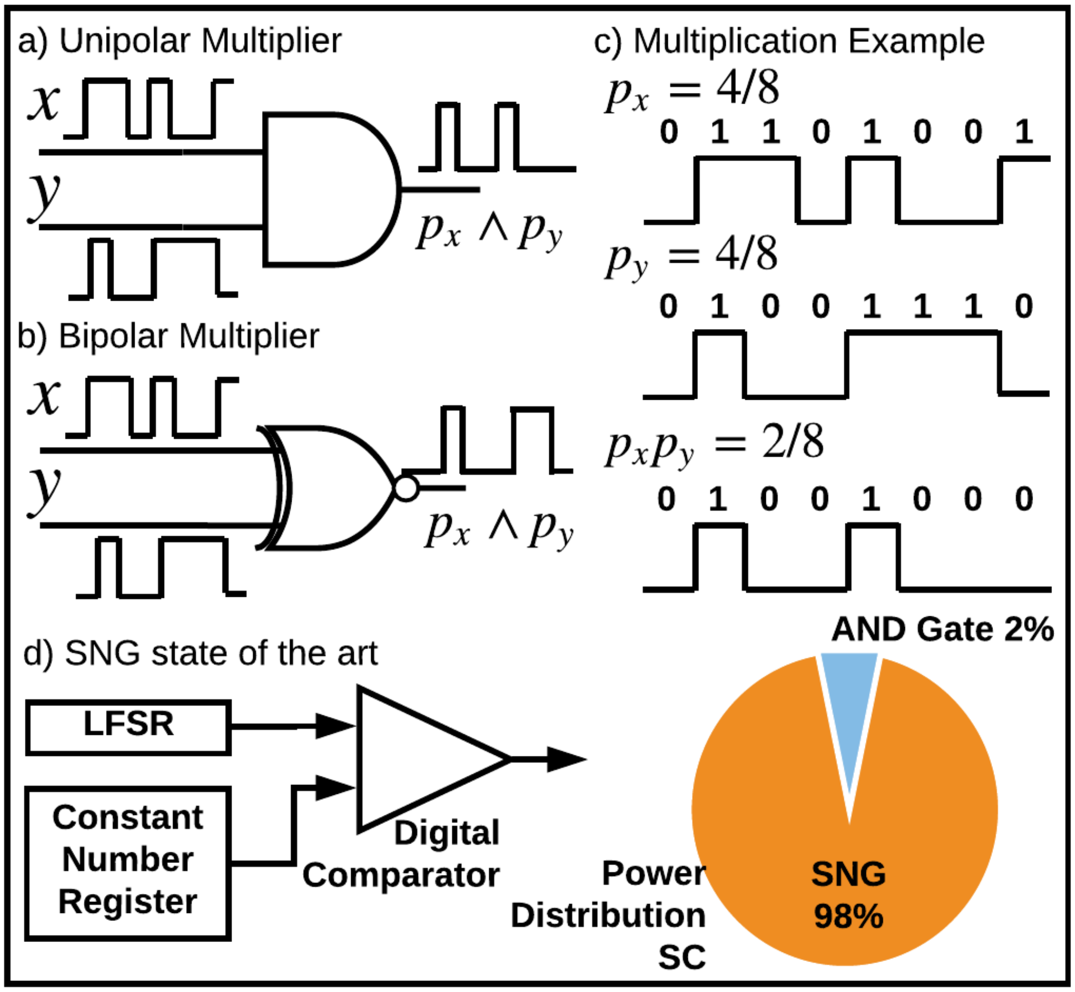

Figure 1-1: Stochastic multipliers for a) unipolar b) and bipolar representation. c) Unipolar multiplication example. d) SNG state of the art

1.2 Thesis Contribution: asynchronous computing with streams

In this dissertation, we introduce asynchronous computing with streams (ACS) as a way to leverage SSC advantages while addressing its drawbacks. In SSC, a number g is associated with **the count** of Ps or “1s" in a stream of bits. In contrast for ACS data is mapped in the following way:

Given a stream that can be in two states {N, P}, we map a number g to the percentage of **the time** the stream is at state P

This change in the stream definition originates ACS, and enables the overhaul of the wireless sensor architecture to met the restrictive power constrains. The main contributions of this thesis are summarized as follows.

1.2.1 Theoretical foundations for Asynchronous Stochastic Computing

Recent work [36] suggested that synchronization requirements between streams for SSC can be relaxed, thus it was proposed to split clock domains down to the level of a handful of gates. In this way, local blocks rely on locally, independently and inexpensively generated clocks instead of a centrally synchronized one [36]. In this work, we propose to eliminate all system clocks and synchronization requirements originating asynchronous stochastic computing (ASC) where the streams are continuous-in-time instead of discrete. Given the continuous nature of this approach, the SSC-Bernoulli model does not explain the dynamics for ASC. Thus, we need to find an appropriate model to explain ASC and the error due to random fluctuations. Figure 1-2 contrast an asynchronous with a synchronous stochastic stream for $p = 0.75$. Notice that for ASC the transitions occur at random continuous-times.

In this dissertation, we develop the theoretical foundations for ASC based on two-state-continuous-time Markov chains (2CTMC) and evaluate the metrics of performance for the basic computing units. We calculate the generation and computation error associated with random fluctuations for asynchronous streams. These foundations enables a fair comparison between SSC and ASC.

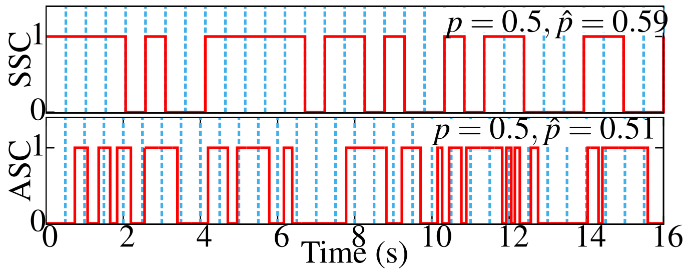

Figure 1-2: Examples for SSC and ASC for $p = 0.5$. Dashed lines represent the positive edge of the clock for SSC.

1.2.2 Asynchronous stream generator

The asynchronous nature of the ASC yields savings in power and energy compared with SSC and typical digital approaches. Encouraged by these results, the next question to address is how to generate asynchronous streams. SSC has historically use LFSRs and comparators to generate the stochastic streams (Figure 1-1d). However, the generation of stochastic streams is one of the major obstacles for stochastic computing because it increases the system cost by 80%-90% [2] and requires expensive stream-digital-stream conversion back-and-forth.

To address the stream generation question, we realized that the strict stochastic condition for the streams can also be relaxed, opening the door for non entirely stochastic yet cost-effective stream generation approaches. Inspired by current architectures for systems on chip (SoC) with sensing capabilities, we propose to use an asynchronous sigma delta modulator (AΣΔM), to generate asynchronous streams. We evaluate the ΣΔ streams using stochastic computing building blocks and analyze the metrics of performance for the approach, originating SC-AΣΔM. Notice that ASC and SC-AΣΔM fall under the umbrella of ACS.

1.2.3 Memory element for ACS

Computing with asynchronous streams has been limited to single stage simple combinationa logic. However, more advanced functions such as, stochastic-tanh (*stanh*) based on state machines [37], or low-error-adders [38] [39] [40] require the use of clocked flip-flops as memory elements. The need for a clock makes this approach not compatible with an asynchronous paradigm, thus we introduce the use of a capacitor embedded in a feedback loop, as a memory-element for continuous time streams and the implementation of a low-error adder for the ACS paradigm.

1.2.4 Applications

To evaluate the performance of ACS compared with the synchronous counterpart and the typical binary approach, we propose, implement and analyze three applications: 1) an image processing algorithm : gamma correction; 2) a digital signal processing algorithm: Fast Fourier Transform; and 3) random forest (Raf) inspired, machine learning (ML) algorithms.

1.2.5 Wireless sensor architecture

Typical SSC approaches rely on multiple digital-to-stochastic, and stochastic-to-digital converters to integrate the stochastic computing units with different components of the architecture. The cost of these interfaces completely eclipses any savings in area and power performance. We propose a novel sensor architecture that takes advantage of the asynchronous nature of the streams and minimizes digital-to-stochastic-to-digital interfaces. Also, we explore the performance of three of those interfaces with: 1) analog inputs (ADC), 2) memory and 3) communication interface.

1.3 Thesis Organization

Following this introduction, Chapter 2 includes a brief description of the theoretical foundations for SSC developed by Gaines [10] in the 60s. Then we develop the theoretical foundations for ASC and use it to understand the metrics of performance for this novel computing paradigm. Chapter 2 will hopefully leave you intrigued given the potential savings in power and energy of ASC.

In Chapter 3 we describe how to generate asynchronous streams. Typical SSC approaches use different sources of noise or pseudo-noise to create the stochastic streams. Examples of these are LSFRs or magnetic tunnel junction devices [41]. Since these extra components increase the energy, power and area cost of the system, in Chapter 3 we propose to use an asynchronous ΣΔ modulator to generate continuous-time asynchronous-streams. These streams can be connected directly to stochastic computing units, originating stochastic computing with asynchronous ΣΔ modulators (SC-AΣΔM). This chapter evaluates the metrics of performance for this approach.

In Chapter 4 we explore applications using ACS. We start our discussion with an FFT implementation, one of the most popular yet expensive algorithms in signal processing. We discuss the performance for ACS when multi-stage logic is required and the implementation of a low-error adder as a capacitor embedded in a feedback loop for the ACS paradigm. As part of the last application, we analyze the cost of large arrays of asynchronous computing elements and the use of near non-volatile, analog-memory (NAM) computing, with an implementation of a machine learning algorithm: Random Forest (RaF).

Finally, in Chapter 5 we discuss some future directions for the evolution of ACS.

Chapter 2

Asynchronous Stochastic Computing (ASC)

In SC a multiplier is a single AND gate, while an adder is a multiplexer [10]. Both circuits cost less than $\sim 7\%$ of the total area and power of conventional binary implementations. This characteristic makes SC extremely attractive for applications that require large arrays of computing elements like digital signal processing [11,38,42,43], Artificial Intelligence algorithms [44–48], and the Internet of Things [49].

The key for the area and power efficiency for SSC is in the unique data representation, which maps a quantity g to the probability p of a stream to be in one of two states {N,P}. Figure 1-2a shows an example of a typical stochastic stream generated with probability $p = 0.5$ and $N = 0, P = 1$. SSC is an approximate paradigm, thus there is an error associated with the stream generation that propagates through the different layers of computation. For example, the stream in Figure 1-2a is generated with $p = 0.5$ but the estimated probability $\hat{p}$ from the stream is $\hat{p} = \frac{1}{L}\sum_{i=1}^{L} X_i = 0.59$, where X_i is the stream at clock cycle i and $L = 32$. This generation and computation error can be derived by modeling the synchronous streams as L consecutive Bernoulli trials [10].

The transitions from one state to another in the stochastic stream occur at discrete moments of time nT where T is the clock period and $n = 1, 2, \ldots$. The need for a clock means that similarly to binary computing, SSC rely on a robust yet expensive clock distribution network (CDN) to counteract the increased susceptibility to process-voltage-temperature (PVT) variations, aging, device mismatch and lowering the power supply to near-threshold and sub-threshold levels [50–52]. The energy cost and low throughput due to long streams [2], on top of the power-area-design cost of the CDN, compromise the initial area and power savings.

ASC is a novel paradigm that leverages SSC advantages, eliminates the cost of CDN and asynchronous-handshake-circuitry, and has the potential to reduce the long latency required to obtain acceptable accuracy. In ASC the generating probability p is mapped to a continuous time (τ_P) the stream remains at state P. As we discuss on the remaining of this chapter, this subtle change in the probability definition results in better accuracy, latency and consequently energy metrics. Figure 1-2b shows an example of an asynchronous stochastic stream for $p = 0.5$. Notice that the transitions occur at random continuous times.

To quantify the advantages of ASC over SSC, we describe the asynchronous stochastic streams as Two-state Continuous Time Markov Chains (2CTMC) (section 2.2), and derive analytical expressions for the error associated with basic SC building blocks (section 2.3). We validate our model by comparing it with numerical simulations of 2CTMC and Bernoulli trials. Using the probability foundations, developed in this chapter, we estimate savings in energy/latency of 20%-50% for ASC compared with SSC for multiplication and gamma correction (section 2.4). Based on Spice-level simulations and post-(P&R) analysis (section 2.5) of an artificial neural network (ANN) implementation we estimate that ASC has the potential of 33%-44%, 10%-55%, and 50% savings in power, latency and energy respectively compared with the SSC approach. These savings makes ASC a good candidate to implement ultra-low-power machine learning accelerators for systems with restricted power budgets such as sensors for the IoT.

2.1 Synchronous Stochastic Computing

Any type of computer that interfaces with the real world deals with analog inputs and outputs. In a typical binary system an Analog to Digital Converter samples an analog signal at a fixed rate to generate a deterministic set of bits [10]. Figure 2-1a shows a typical fixed point representation of the number 4 using 8 bits. In SSC, a number g with range $0 < g < V$ is mapped to the probability $p = g/V$. In turn p is reflected on the probability that the logic level is “P" in a stream of bits where each bit last one clock cycle [10]. Figure 2-1b shows examples of typical stochastic streams with $p = 0.25$, $P = 1$, and $N = 0$.

The sequences shown in Figure 2-1b are $L = 8$ successive Bernoulli trials, hence, we can define the random variable (RV):

$$X_{ssc} = \text{number of samples being 1 in a sequence.} \tag{2.1}$$

where the sub-index “ssc" indicates that it is valid for the SSC approach. X_{ssc} has a bi-

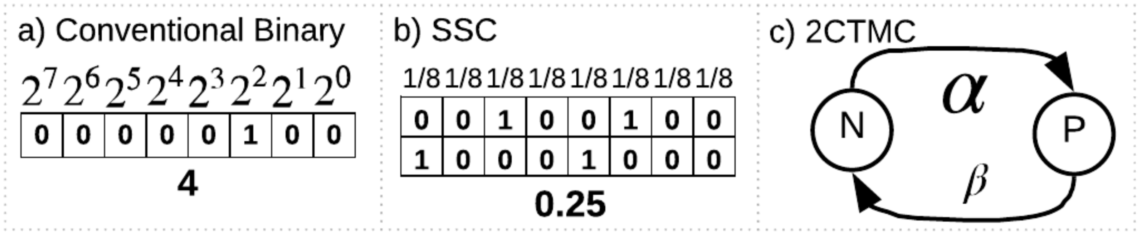

Figure 2-1: a) Typical binary data representation. b) Synchronous stochastic streams with generating probability $p = 0.25$. c) Two state continuous time Markov chain model for ASC.

nomial distribution with expected value $\mathrm{E}(X) = Lp$. From a Bernoulli sequence only the estimate $\hat{p}_{ssc}$ of p can be obtained. This estimate can be expressed as $\hat{p}_{ssc} = X_{ssc}/L$. The expected value of $\hat{p}_{ssc}$ is $\mathrm{E}[\hat{p}_{ssc}] = p$ and the associated variance $\mathrm{VAR}[\hat{p}_{ssc}]$:

$$VAR[\hat{p}_{ssc}] = \frac{p(1-p)}{L}. \tag{2.2}$$

The error between the estimated probability $\hat{p}_{ssc}$ and the generating probability p is given by the standard deviation: $\sigma(\hat{p}_{ssc}) = \sqrt{\mathrm{VAR}[\hat{p}_{ssc}]}$ [10].

2.2 Asynchronous Stochastic Computing

In ASC the number g is mapped to the probability of **the time** the stream is at logic level “P", thus we define a RV X_{asc}:

$$X_{asc} = \text{Total time the stream is at logic level “P"}. \tag{2.3}$$

This random experiment can be described as a two-state continuous time Markov chain (2CTMC), which is a continuous time stochastic process $\{X_t\}$ with the Markov property. $\{X_t\}$ holds the complete record of logic levels occupied by the stream at all times [53] [54] and there are only two possible states $\{1, 0\}$. If at time t the stream is at state 0(1), it remains in that state for a random time that is exponentially distributed with parameters $\tau_0 = 1/p$ and ($\tau_1 = 1/(1-p) = 1/q$). The asynchronous stream is fully described by the transition matrix [53]:

$$P(t) = \begin{bmatrix} q + pe^{-\frac{t}{pq}} & p - pe^{-\frac{t}{pq}} \\ q - qe^{-\frac{t}{pq}} & p + qe^{-\frac{t}{pq}} \end{bmatrix} \tag{2.4}$$

Similarly to the synchronous case, we can only estimate from the stream the generating probability as:

$$\hat{p}_{asc} = \frac{1}{t_L} \int_0^{t_L} \{X_t\} dt \tag{2.5}$$

where the sub-index "asc" indicates that it is valid for the ASC approach, and t_L is the stream length in seconds. Using 2.4 and 2.5, we obtain the expected value $\mathrm{E}[\hat{p}_{asc}]$ as:

$$\mathrm{E}[\hat{p}_{asc}] = \frac{1}{t_L} \int_0^{t_L} \mathrm{E}[\{X_t\}] dt = p \tag{2.6}$$

Since the generation error is proportional to the variance we calculate $\mathrm{VAR}[\hat{p}_{asc}]$ as:

$$\begin{aligned} \mathrm{VAR}[\hat{p}_{asc}] &= \mathrm{E}[\hat{p}_{asc}^2] - \mathrm{E}[\hat{p}_{asc}]^2 \\ &= \frac{2p^3q^3}{t_L^2}\left(e^{\frac{-t_L}{pq}} - 1\right) + \frac{2p^2q^2}{t_L} \end{aligned} \tag{2.7}$$

To verify equations 2.6 and 2.7, we compare them against the $\mathrm{E}[\cdot]$ and $\mathrm{VAR}[\cdot]$ calculated from 20 simulated streams using 2CTMC generators developed in Matlab (Figure 2-2a) . The experimental $\mathrm{E}[\hat{p}]$ will converge to the analytical $\mathrm{E}[\hat{p}]$ as the number of samples N in the experiment increases. The ASC models agree with the generated asynchronous stochastic streams.

To compare ASC with SSC, Figure 2-2a also shows SSC's $\mathrm{E}[\cdot]$ and $\mathrm{VAR}[\cdot]$ for both, simulated streams and the expressions derived from the probability model described in [10]. For a fair comparison, we assume that SSC is equivalent to ASC if the stochastic streams for both approaches have a similar average number of transitions. For example, L=16 (SSC) is equivalent to a computing time of $t_L = 8$ (ASC), given a clock-cycle of 0.5 s. This clock-cycle is an arbitrary choice to simplify the mathematics but can be scaled to μ/n/p s to match current technologies clock frequencies. As we will further discuss in section 2.4 ASC yields a slightly smaller error due to the absence of quantization error.

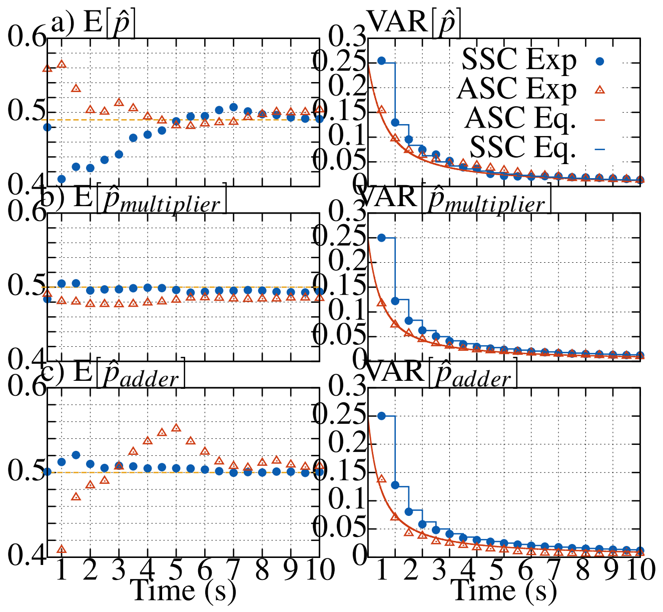

Figure 2-2: This figure compares the theoretical models (Eq.) with the simulated streams (Exp.) for ASC and SSC. Synchronous streams are modeled as L consecutive Bernoulli trials, while asynchronous streams are modeled as 2-state-continuous-time-Markov-chains of length t_L. The expected value E[$\hat{p}$] is at the left while the VAR[$\hat{p}$] is at the right. a) Stream generation for $p = 0.5$. b) Multiplication for $p_x = 0.7$ and $p_y = 0.7$. c) Scaled addition for $p_x = 0.5$ and $p_y = 0.5$ with $p_a = 0.5$

2.2.1 Data Representation

Similar to SSC, in ASC the analog input g can be mapped in two ways:

Unipolar

In a unipolar representation, g in the range $0 < g < V$ is mapped to the range $[0,1]$ which is the natural range for probability [10]. Thus $p = g/V$, and E[$\hat{p}_{asc}$], VAR[$\hat{p}_{asc}$] are given by (2.6) and (2.7) respectively.

Bipolar

Several applications require the use of negative numbers or a bipolar representation of data. In the bipolar representation, the analog input g in the range $-V < g < V$ is mapped to a binary stream using the transformation [10]

$$p = \frac{g}{2V} + \frac{1}{2} \tag{2.8}$$

In this case the maximum level V is represented by a stream that is 1 all the time. The minimum level -V is a stream that is 0 all the time. Finally, for $g = 0$ the stream is generated with $p = 0.5$. From (2.8), $g/V = 2p - 1$ thus an estimate for g/V is given by $\hat{g}/V = 2\hat{p} - 1$ [10] and

$$\text{VAR}[\hat{g}/V] = 4VAR[\hat{p}] = 2(1 - (g/V)^2)(\frac{1}{t'_L} + \frac{e^{-t'_L} - 1}{t'^2_L}) \tag{2.9}$$

with $t'_l = 4Vt_L/(V^2 - g)$.

2.3 Operations

In section 2.2 we introduced an analytic model to describe asynchronous stochastic streams and their properties. In this section, we use this mathematical model to find the computing error due to random fluctuation for the stochastic multiplier and adder (Figure 2-3). To find this error, we use the estimated probability at the output of the computing block $\hat{p}_z$. Thereafter, we calculate the expected value (E[·]) and variance (VAR[·]). In this chapter we showcase the analysis for the unipolar case, however, the error for the bipolar inverter, multiplier (2-3c) and adder can be derived using the transformation in (2.8) following a similar methodology. Our model confirms that asynchronous and synchronous SC share the basic combinational computing elements and enables the comparison between ASC and SSC developed in section 2.4.

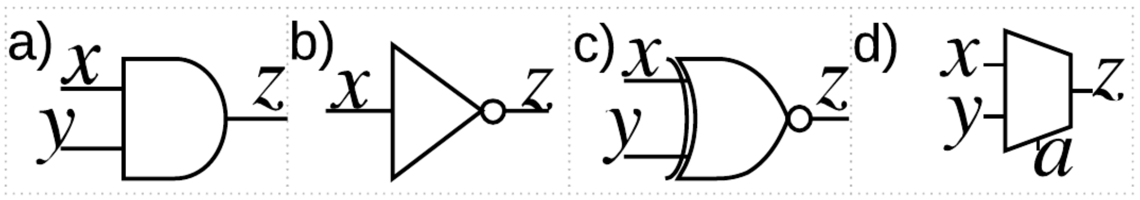

Figure 2-3: The stochastic computer basic building blocks. a) Unipolar multiplier with streams x and y as inputs b) Inverter with stream x as input c) Bipolar multiplier with streams x and y as inputs. d) Scaling adder with streams x and y as inputs

2.3.1 Unipolar multiplier

Two continuous-time streams $\{X_t\}$ and $\{Y_t\}$ with generating probabilities p_x and p_y can be multiplied by using an AND gate (figure 2-3a). The estimated probability at the output of the AND gate is:

$$\hat{p}_z = \frac{1}{t_L}\int_0^{t_L}\{X_t\}\{Y_t\}dt \tag{2.10}$$

The expected value for $\hat{p}_{and}$ is given by:

$$\mathrm{E}[\hat{p}_z]_{xy} = p_x p_y \tag{2.11}$$

and the variance is:

$$\begin{aligned}\mathrm{VAR}[\hat{p}_z]_{xy} = {} & p_x^2\mathrm{VAR}[\hat{p}_y] + p_y^2\mathrm{VAR}[\hat{p}_x] \\ & + \frac{2a^3}{b^2t_L^2}\big(e^{\left(\frac{-bt_L}{a}\right)} - 1\big) + \frac{2a^2}{bt_L}\end{aligned} \tag{2.12}$$

with VAR$[\hat{p}_y]$, VAR$[\hat{p}_x]$ given by 2.7, $a = p_x q_x p_y q_y$, $b = p_x q_x + p_y q_y$ and t_L is the computing time. Figure 2-2b shows E$[\hat{p}_z]$ and VAR$[\hat{p}_z]$ versus t_L for $p_x = p_y = 0.7$. We validate our expression with the same approach used in section 2.2. The simulated streams match the curves described by 2.12 and 2.11. We also show the expectance and variance for SSC multiplication [10].

2.3.2 Unipolar weighted adder

Weighted addition for two asynchronous stochastic streams can be implemented using a 2:1 Multiplexer (Figure 2-3d), where $\{A_t\}$ is a third stochastic stream that connects the output with one input at a time. The generating probability for $\{A_t\}$ is p_a, a scaling factor used to guarantee that the adder result is always within the probability valid range $[0,1]$. For random streams $\{X_t\}$, $\{Y_t\}$, and $\{A_t\}$ the estimated probability at the output of the 2:1 Multiplexer for ASC can be found as:

$$\hat{p}_z = \frac{1}{t_L}\int_0^{t_L}\{A_t\} + (1-\{A_t\})\{Y_t\}dt \tag{2.13}$$

The expected value is:

$$\mathrm{E}[\hat{p}_z]_{xy} = p_a p_x + (1-p_a)p_y = p_a p_x + q_a p_y \tag{2.14}$$

In particular, when $p_a = 0.5$ we obtain the average adder $\mathrm{E}[\hat{p}_{add}] = \frac{p_X + p_Y}{2}$. The variance can be calculated as:

$$\begin{aligned} \mathrm{VAR}[\hat{p}_z]_{xy} = \mathrm{VAR}[\hat{p}_{mult}]_{ax} + \mathrm{VAR}[\hat{p}_{mul}]_{\bar{a}y} \\ - 2p_x p_y \mathrm{VAR}[\hat{p}_a] \end{aligned} \tag{2.15}$$

with $\mathrm{VAR}[\hat{p}_{mult}]_{ax}$, $\mathrm{VAR}[\hat{p}_{mult}]_{\bar{a}y}$ given by 2.12 and $\mathrm{VAR}[\hat{p}_a]$ given by 2.7. Figure 2-2 shows the $\mathrm{E}[\hat{p}_z]$ and $\mathrm{VAR}[\hat{p}_z]$ for the addition versus t_L. Similarly to previous examples we validate our mathematical expressions with simulated streams.

Table 2.1: Compares SSC and ASC probability models. We also show the expected value $\mathrm{E}[\cdot]$ and variance $\mathrm{VAR}[\cdot]$ for different stochastic operations.

Parameter		**SSC**	**ASC**
Probability Model		Binomial Distribution	2 State Continuous time Markov Chain
Random Variable		X_{ssc} discrete	X_{asc} continuous
Estimated probability $\hat{p}$		$\frac{1}{L}\sum_i^L X_i$	$\frac{1}{t_L}\int_0^{t_L}\{X\}dt$
Stream Generation	$\mathrm{E}[\hat{p}]$	p_g	p_g
	$\mathrm{VAR}[\hat{p}]$	$\frac{p_g(1-p_g)}{L}$	$\frac{2p^3q^3}{t_L^2}\left(e^{\frac{-t_L}{pq}} - 1\right) + \frac{2p^2q^2}{t_L}$
Multiplication	$\mathrm{E}[\hat{p}_{AND}]$	$p_x \times p_y$	$p_x \times p_y$
	$\mathrm{VAR}[\hat{p}_{AND}]$	$p_x\mathrm{VAR}[\hat{p}_y]$ + $p_y\mathrm{VAR}[\hat{p}_x]$ + L $\mathrm{VAR}[\hat{p}_x]\mathrm{VAR}[\hat{p}_y]$	$p_x^2\mathrm{VAR}[\hat{p}_y] + p_y^2\mathrm{VAR}[\hat{p}_x]\frac{2a^3}{b^2t_L^2}\left(e^{\left(\frac{-bt_L}{a}\right)} - 1\right) + \frac{2a^2}{bt_L}$
Addition	$\mathrm{E}[\hat{p}_{ADD}/2]$	$\frac{p_x+p_y}{2}$	$\frac{p_x+p_y}{2}$
	$\mathrm{VAR}[\hat{p}_{ADD}/2]$	$p_a\mathrm{VAR}[\hat{p}_x]$ + (1+p_a)$\mathrm{VAR}[\hat{p}_y]$ + $(p_x - p_y)^2$ $\mathrm{VAR}[\hat{p}_a]$	$\mathrm{VAR}[\hat{p}_{mult}]_{ax} + \mathrm{VAR}[\hat{p}_{mul}]_{\bar{a}y} - 2p_xp_y\mathrm{VAR}[\hat{p}_a]$
Inversion	$\mathrm{E}[\hat{p}_{INV}]$	$-p_x$	$-p_x$
	$\mathrm{VAR}[\hat{p}_{INV}]$	$\frac{p_g(1-p_g)}{L}$	$\mathrm{VAR}[\hat{p}]$

2.4 Asynchronous versus Synchronous SC

In sections 2.2, 2.3 we observed that ASC yields a small accuracy improvement. In this section, we show that this small improvement yields 20%-50% for the overall system accuracy/latency. To obtain a fair comparison between SSC and ASC we apply the same methodology used in section 2.2.

Quantization error for stream generation

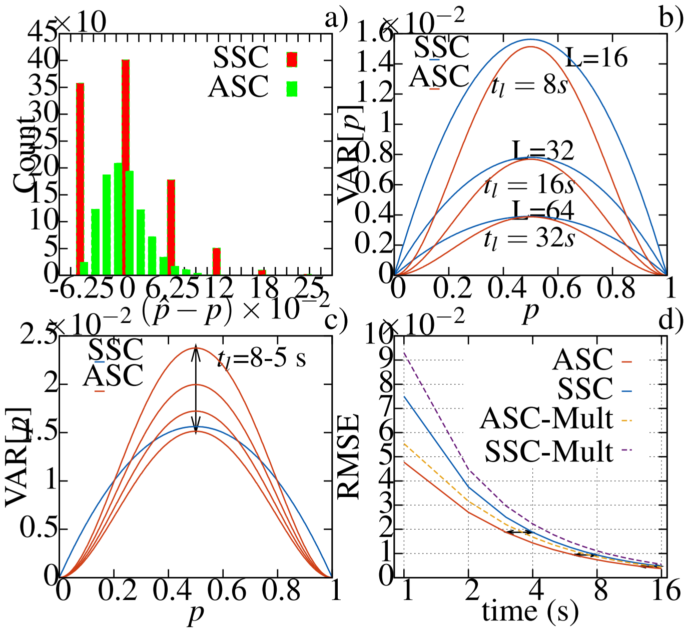

Figure 2-4: a) Error distribution for $p = 0.0625$ with L=16 (SSC) and $t_L = 8s$ (ASC). b) SSC and ASC VAR[$\hat{p}$] for different computation times. b) VAR[$\hat{p}$] for L=16 (SSC) and for $t_L = 8s, 7s, 6s, 5s$ (ASC) d) RMSE versus time for stream generation and multiplication

For SSC, $\hat{p}$ is mapped to a discrete space $S = [0, 1/L, 2/L, ..., L/L]$. In contrast for ASC, $\hat{p}$ is mapped to the real numbers in the interval [0,1]. Intuitively, this difference between continuous and discrete spaces yields a more accurate representation of the continuous analog input. As an example, Figure 2-4a shows the histogram for the distance $d = p - \hat{p}$ for 1000 SSC and ASC streams when $p = 0.0625$. For SSC d is quantized and is -0.0625 for $\sim$350 streams. In contrast, for ASC d is distributed in the interval $[-0.625 - 0.1]$ resulting in less than 5 streams having an error as high as -0.0625. The advantage of ASC over SSC for all ps can be observed in the VAR[$\cdot$] (Figure 2-4b). ASC yields lower variance as p tends to the extremes given the freedom for when to transition.

This feature enables a reduction in latency for ASC. Figure 2-4d shows the root mean squared error (RMSE) versus time (L) calculated over all possible ps representable by a particular L. ASC requires 20% less time than the SSC counterpart, while obtaining the same average error. To understand this, let's take a look at Figure 2-4c, where we compare VAR[$\hat{p}_{ssc}$] for L=16 with VAR[$\hat{p}_{asc}$] for t=8,7,6,5. For a small reduction in the computing time, for example t=7 s, VAR[$\hat{p}_{ssc}$]<VAR[$\hat{p}_{asc}$] around $p \approx 0.5$, however, this is balanced out as p moves to the edges and VAR[$\hat{p}_{ssc}$]>VAR[$\hat{p}_{asc}$] yielding a smaller RMSE for ASC. The balancing effect dissapears when VAR[$\hat{p}_{asc}$]>VAR[$\hat{p}_{ssc}$] for the major part of the curve.

Quantization error for Multiplication

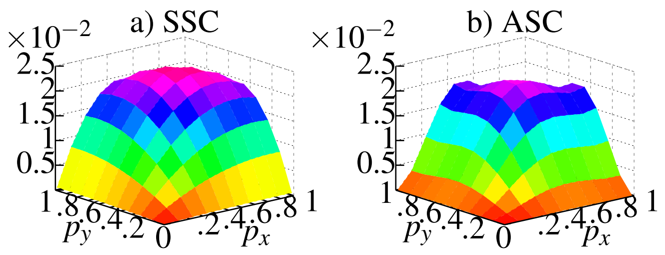

Figure 2-5: Compares the variance for synchronous (a) and asynchronous (b) multiplication.

Figures 2-5a and 2-5b show the variance for ASC and SSC multiplication for t=8s and L=16. Similar to the stream generation error, VAR[$\hat{p}_{asc}$] <VAR[$\hat{p}_{ssc}$]. To evaluate the overall advantage of ASC multiplication, we calculate the RMSE versus time (Figure 2-4d). There is also a slight improvement in accuracy that reflects in a reduction in latency of 20% for ASC.

Quantization error for Gamma Correction

To evaluate how ASC performs for an application, we implement gamma correction, an algorithm used to control the brightness of an image. Each pixel in the image is transformed by $V_{out} = V_{in}^{\gamma}$ [55]. We use a compact SC architecture of gamma correction based on Bernstein polynomials [2], depicted in Figure 2-6a. The coefficients $b_0 = 0.0955$, $b_1 = 0.7207$, $b_2 = 0.3476$, $b_3 = 0.9988$, $b_4 = 0.7017$, $b_5 = 0.9695$, and $b_6 = 0.9939$ approximate $\gamma = 0.45$ [2]. We implement Continuous Time Markov Chain Generators (CTMCG) in Matlab to generate the continuous time streams. Figure 2-6b shows the output of the stochastic architecture for gamma correction obtained at different times. Notice that ASC yields a better image quality in less computing time. To quantify the latency savings we calculate the RMSE versus time for both SSC and ASC (Figure 2-6c). ASC yields at least 50% of savings in latency for the same SSC accuracy.

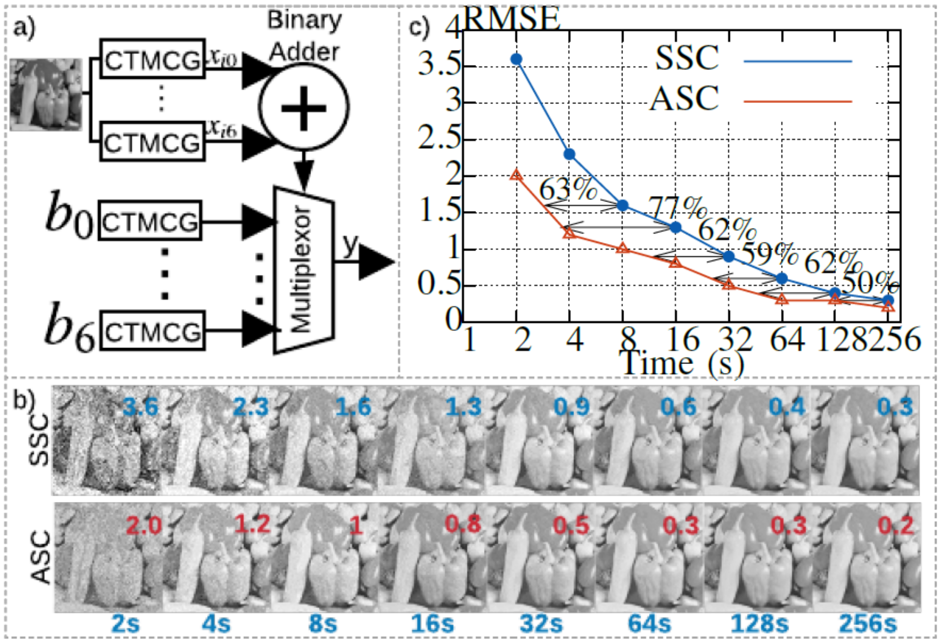

Figure 2-6: Gamma correction: a) Block diagram. b) Results for different times using SSC and ASC. c) RMSE versus time

2.5 Energy and power considerations

Developments in machine learning, especially in ANN algorithms, have shown commercial applicability in video and natural-language processing [56]. However, an ANN usually requires massive, power-hungry computational resources typically found only in datacenters [57, 58]. Among several efforts to reduce the power/energy/area cost of ANN, is the development of SSC-ANN [44–48].

We use an ANN as an example to evaluate how ASC measures against SSC in terms of power, area and energy. Figure 2-7a shows the block diagram for a typical ANN composed of interconnected neurons. Each neuron consist of (1) synapses that multiply inputs by weights and (2) sigmoid-like activation function (σ) applied on the sum of the output of the synapses [59]. Figure 2-7b shows the stochastic implementation of the neuron. While the bipolar multipliers and adders are common for the ASC and SSC approaches, there is not an ASC equivalent for the activation function. In SSC, the activation function is the stochastic $\tanh(N,x)$, implemented as a state machine with N states [37]. Unfortunately such implementation requires clocked flip-flops not compatible with ASC. In this work, we use a charge pump integrator to implement the activation function (Figure 2-7c). A charge pump consist of a current source (I1) and sink (I2) of equal magnitude separated by switches s1 and s2. The inverter acts as a current comparator [60]. In the neuron, the output of the adder controls s1(s2) such that charge is pumped in and out of the capacitor Cp. Figure 2-7d shows the estimated average $\hat{p}_o$ of the output of the activation function for different values of I1(I2). Figure 2-8 shows an example of the streams at different stages of the ASC-neuron obtained from Spice level simulations, in particular, notice the stream nature for the output of the activation function (Figure 2-8i). We implement a single layer of a fully connected ANN for both SSC (multiplier-adder-activation function) and ASC (multiplier-adder) using SystemVerilog. We run synthesis and P&R for both approaches, using a FinFET1X technology foundry models for three different power supplies. $Vdd3$ is the technology's nominal power supply, $Vdd2 = Vdd3 - 0.1$ and $Vdd1 = Vdd2 - 0.1$. For the purposes of P&R, we model the ASC activation function as an inverter (switches s1 and s2). Since the ASC activation function is not included in the P&R flow, we compare its metrics of performance using Spice-level simulations (Table 2.2). These results can be extrapolated to several layers, given the regularity of the ANN architecture.

2.5.1 Power and area analysis

The dynamic power $P_{dyn} = C_L V_{dd}^2 f_{0\to1}$ dissipated by the computing units (section 2.3) is in average the same for both ASC and SSC, because the circuit capacitance C_L and

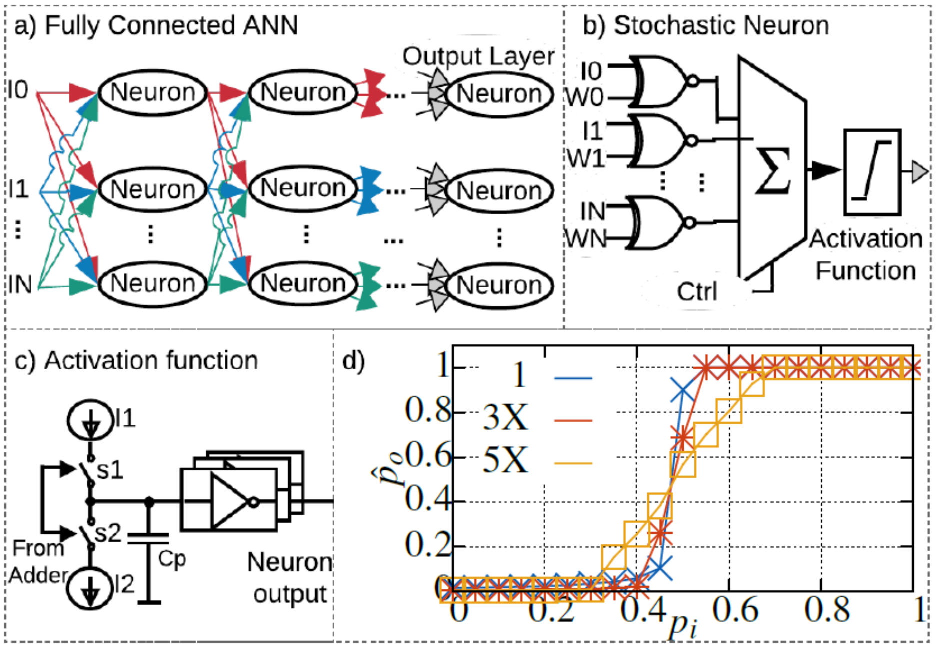

Figure 2-7: a) Block diagrams for ANN. b) Stochastic implementation of a neuron c) Sigmoid-like asynchronous activation function. d) Output of the asynchronous activation function for different input currents. The smaller current (1) results in a sharp transition while increasing the current 3X and 5X, results in a smoother transition.

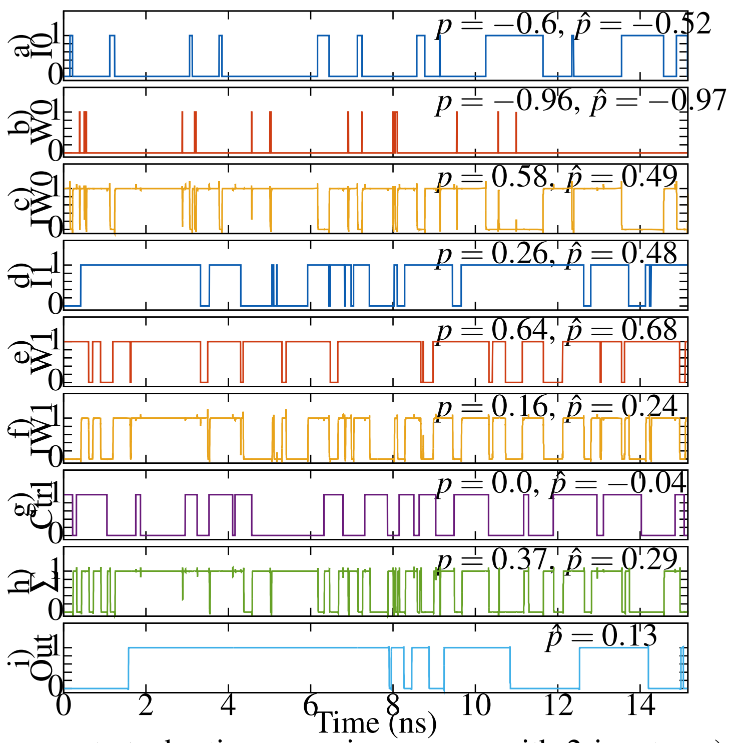

Figure 2-8: Streams at stochastic computing neuron with 2 inputs. a) Input 0 (I0). b) Weight 0 (W0). c) Multiplication IW0=I0×W0. d) Input 1 (I1). e) Weight 1 (W1). f) Multiplication IW1=I1×W1. g) Auxiliar stream generated with $p = 0$ to control the stochastic adder. h) Adder result ($\sum$). i) Activation function output.

the power supply V_{dd} are identical, and we assume a comparable average transition rate ($f_{0\rightarrow 1}$). However, ASC has the potential for power savings due to the elimination of the CDN (typical digital paradigm) or handshake signals (typical asynchronous paradigm).

Figure 2-9a shows the gate count for ASC and SSC after P&R. SSC requires in average 5X more buffers and 9X more inverters than ASC to compensate for PVT variations. The increment of buffer and inverters is more evident as the power supply is reduced. For $Vdd1$ SSC requires at least 18X more inverters than ASC, while for $Vdd3$ SSC requires only 5X more inverters than ASC. This increment in the number of buffers/inverters can be directly translated to an increment in power/area consumption. Figure 2-9c shows the percentage of the power dedicated solely to the CDN for the SSC approach. Given the absence of CDN in ASC we can infer savings in power of at least 33%-44%.

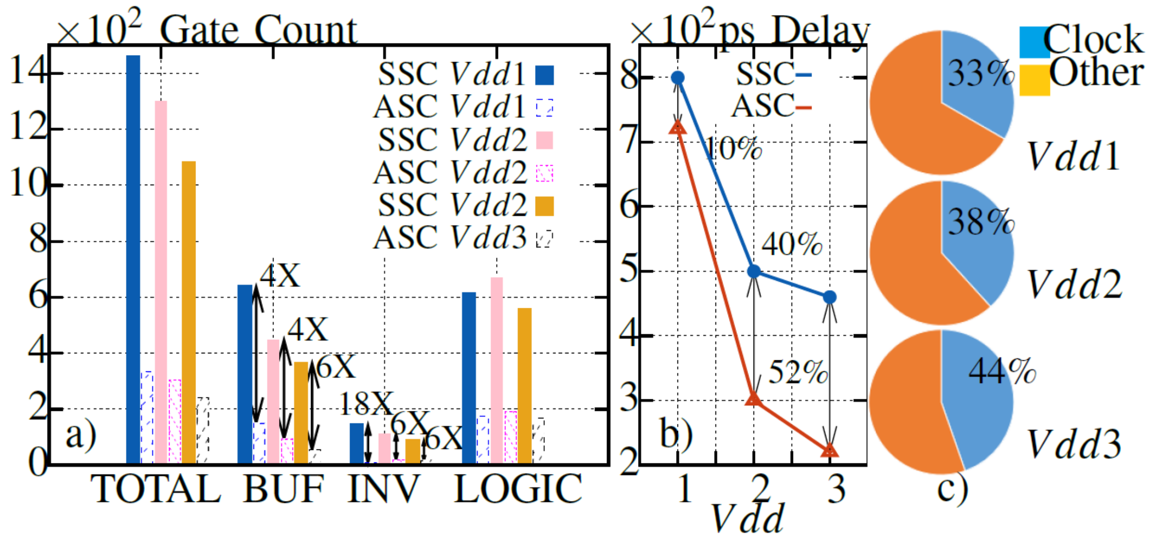

Figure 2-9: a) Block diagrams for ANN. b) Stochastic implementation of a neuron c) Sigmoid-like asynchronous activation function. d) Output of the asynchronous activation function for different input currents. The smaller current (1) results in a sharp transition while increasing the current 3X and 5X, results in a smoother transition.

2.5.2 Delay

In section 2.3, we assume ideal circuits with zero propagation delay t_p. For real circuits $t_p \neq 0$ and depends on the logic-gate output load and PVT variation. Figure 2-9b shows the minimum propagation delay t_p for both ASC and SSC obtained after P&R. Notice that for $Vdd1$, the difference in t_p is only 10% given the more pronounced effect of PVT variation for both approaches. However as the power supply increases the propagation delay for ASC is 52% smaller than for SSC.

2.5.3 Energy

Dynamic energy can be estimated as $E_{dyn} = P_{dyn}t_L$. A reduction in either P_{dyn} or t_L, yields equivalent savings in energy. From the previous discussion about power and delay we can argue at least 10%, or up to 52% savings in energy due to the latency reduction and the CDN elimination.

To estimate the energy consumption for both approaches we obtain the metrics of performance (Table 2.2) from Spice-level simulations for the ASC and SSC neuron with two inputs. ASC saves 50% of energy compared with the SSC for a stochastic neuron implementation.

Table 2.2: Metrics of performance for a SSC and ASC stochastic neuron building components such as: multiplication ($\times$), addition ($+$), and the sigmoid function (σ). Power and delay obtained for maximum frequency of operation at $Vdd2$. The streams were generated assuming the same average number of transitions for both approaches.

	ASC				SSC			
	$\times$	$+$	σ	Total	$\times$	$+$	σ	Total
Latency (ns)	9.6				16			
Power (μW)	5.4	3.3	7.9	16.6	2.9	3.2	1.4	7.5
Energy (fJ)	52	32	75	160	47	51	224	322

2.6 Conclusion

This chapter develops theoretical foundations for ASC, modeling the asynchronous streams stochastic behavior with a 2CTMC. Using our model, we analyze the error due to random fluctuations for the stochastic multiplier and adder. One of the major roadblocks for SSC is the long computation latency required to obtain acceptable accuracy. A comparison between the probability models shows that ASC yields 20% and 50% of savings in latency for multiplication and gamma correction respectively. Moreover one of the most attractive features of ASC is the elimination of CDN or asynchronous hand-shake circuits. For a single layer in an ANN, we find that ASC yields savings of at least 33%-44% and 10%-50% in power and latency respectively after P&R. Spice-level simulations for a typical stochastic neuron show that ASC saves 50% in energy compared with SSC. The savings in power, latency and energy makes ASC an alternative to reduce the power consumption of complex machine learning algorithms, enabling artificial intelligence in power constrained environments such as the Internet of Things.

Chapter 3

Asynchronous computing on ΣΔ streams

3.1 Introduction

The previous chapter introduced asynchronous stochastic computing and analyzed the computing error due to random fluctuations, finding that ASC yields savings in power, latency and energy for the stochastic computing units compared with the SSC paradigm. In this chapter we shift our attention towards how to generate these asynchronous streams.

The main advantages of SC are the simplicity of the logic required for otherwise complex functions and the robustness in the presence of bit errors (flipped-bits) [2] [33] [34]. However, one of the roadblocks for SC mainstream adoption is the stream generation. Currently, the streams are generated using LFSRs to generate pseudo-random numbers (Figure 1-1d). In this approach, SNG can use up to 80% of the total area and 67%-92% of the total power consumption in a SC system [2].

In this work, we propose to apply SC directly on ΣΔ streams in order to minimize the power associated with SNG. This approach is specially advantageous for Systems on Chip (SoCs) with sensing capabilities which already include a ΣΔ modulator as part of the Analog to Digital Converter (ADC) (Figure 3-1a). Instead of adding the LFSR based SNG to convert the binary data back to a stream (Figure 3-1a), we propose to simplify the node by computing directly on the streams generated by the ΣΔ modulator (Figure 3-1b). Our approach eliminates the power and area associated with the SNG and digital filters in the ADCs. To further minimize the power consumption of the proposed system, we use a current-based asynchronous ΣΔ modulator. Operating in current mode simplifies the building circuits for the modulator and the asynchronous nature posses several benefits described in Section 3.2.3.

Applying SC directly on the ΣΔ streams is not a straightforward solution because those

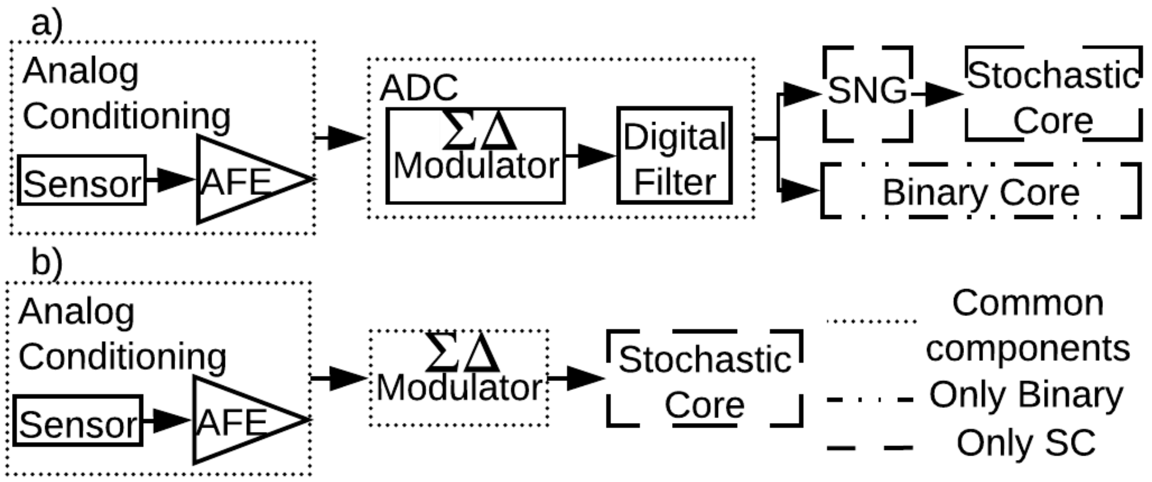

Figure 3-1: In this figure we identify the common components between the typical approach and our approach. a) Typical SoC with a ΣΔ ADC and a digital core for on node processing. For SC, a SNG is added before the stochastic core. b) SC-SD.

streams are highly correlated. For example, multiplying ΣΔ streams x, y with an AND gate, when $x = y$, yields to $p_{x \wedge y} = p_x = p_y$ instead of $p_{x \wedge y} = p_x p_y$. To address these challenges, in this work, we propose for the first time, to the best of our knowledge, to use SC directly on pulse density modulated (PDM) streams by :

1. Using the AΣΔM as a low power solution for the analog to streams generator. (Section 3.2.3)
2. Exploring reducing the correlation between PDM streams generated with ΣΔ modulators in two ways (Section 3.4):
   (a) Taking advantage of on-chip process variation as a source of randomness for ΣΔ streams.
   (b) Shifting the modulator natural frequency of oscillation (ω_c).

We demonstrate the potential of these ideas to achieve low energy SC-inspired computation by implementing a widely used image processing algorithm (gamma correction) with the streams generated from AΣΔMs (Section 3.5). We implement the Gamma correction application in an industry-standard 1xFinFET technology. Simulation results show 98%-11% savings in the total system latency, and, 50%-38% in power savings when compared with the binary counterpart or the SC-LFSR approach. Due to the fact that the front end stages and the SNG circuits have been simplified, we believe that similar savings can be achieved by implementing the proposed techniques on other applications as well.

3.2 Background

3.2.1 Computation on ΣΔ streams

Different approaches have been proposed for operating directly on ΣΔ streams with 1-bit stream processors. One approach is to modify the structure of the modulator to implement complex functions like filtering [61] [62]. However the output of the modulator is not the raw stream but a modified version. A second approach operates over the ΣΔ streams using counters or transversal filters [63] [64] [40]. Both approaches are difficult to generalize for different applications and some include the use of analog blocks that increase the complexity of the design. Non of these approaches apply SC for processing the ΣΔ streams. Analog to stochastic converters in literature, combining ADC with LFSRs do not improve power-area metrics [65] [66]. To the best of our knowledge, our work is the first to apply stochastic computing directly on PDM streams as generated with an asynchronous ΣΔ modulator, and to discuss methods to lower the correlation among the streams.

3.2.2 Pulse modulation

ΣΔ modulators generate Pulse Density Modulated (PDM) streams, which encode the input into the density of pulses as shown in Figure 3-2. More formally, PDM sets the duty cycle δ and the angular frequency $\omega = 2\pi/T$ of the output stream according to [67]:

$$\delta = \frac{I+1}{2} \qquad \frac{\omega}{\omega_c} = 1 - I^2 \qquad \text{for } |I| < 1 \tag{3.1}$$

with ω_c the natural frequency of the modulator or the frequency of oscillation when the input is zero [67], T the period and I the input normalized by its maximum value. Note that the average value of the output stream, which represent its probability, is given δv_h.

3.2.3 The asynchronous ΣΔ modulator

To implement the ΣΔ modulator we adopt two design techniques that present several advantages over more conventional clocked plus switched-capacitors schemes:

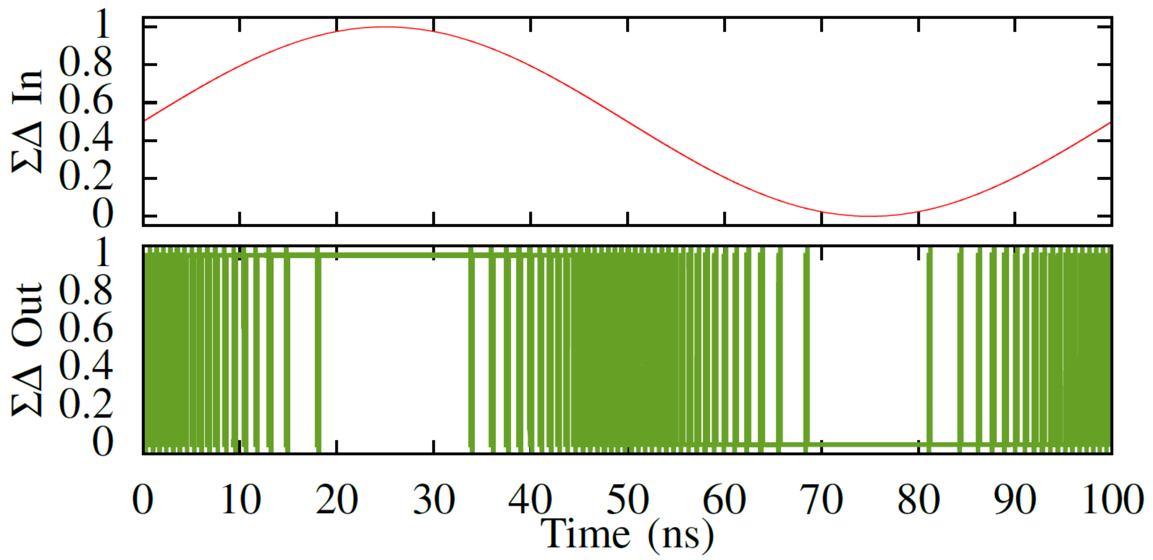

Figure 3-2: ΣΔ streams (PDM) example for a sine wave. Notice that ω and δ change with the input. The maximum frequency occurs when the input is 0.5, on the other hand the frequency is zero when the input is at its minimum or maximum.

1. *Current-based design*, which reduces area and power by eliminating switched capacitors and high gain active components. Reducing the circuit capacitance to only the parasitic capacitance associated with transistors allows an increase in the sampling rate [68] [69].
2. *Asynchronous modulation*, which switches only when the input integrator crosses a threshold instead of at each clock cycle, reducing the activity factor (α). This combined with a simpler integrator leads to power and area savings. Moreover, the lack of sampling and hold circuits eliminates errors due to charge injection, switch non-linearity, clock feed-through and finite settling time [70] [30].

Figure 3-3e shows the block diagram of the AΣΔM. It includes an integrator, a quantizer, a feedback element and an adder. The modulator takes the difference between the input signal X_{in} and the output signal V_{out} and integrates it until the comparator (quantizer), threshold is reached. Then the output is reversed and the integration starts again. While the pulse at the output has a constant amplitude of v_h, the transition time related with ω and pulse width related with δ depend on the input signal dynamics, thus the stream is discrete in amplitude but continuous in time.

Each building block of the AΣΔM is broken down in circuit elements in Figure 3-3. By

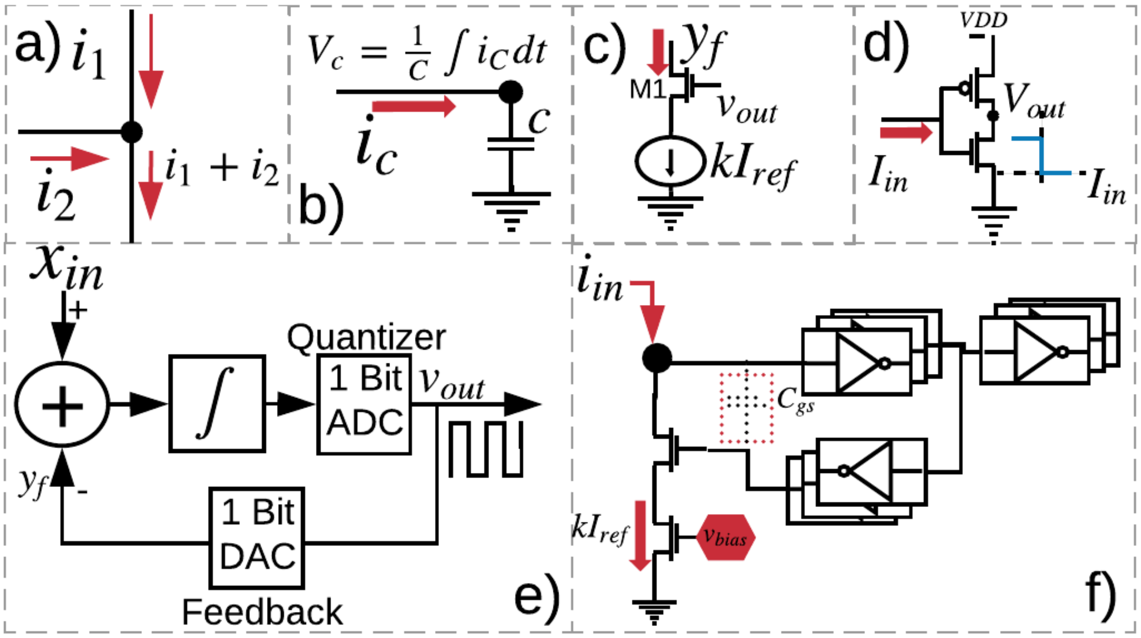

Figure 3-3: Current Based AΣΔM building circuits: a) Adder, b) Integrator, c) Feedback, d) Comparator. e) ΣΔ Modulator block diagram. f) Current based AΣΔM architecture.

Table 3.1: Area-Power summary for LFSR-SNG and AΣΔM

SNG	Area (μm^2)	Power (μW) @1GHz	Technology
LFSR	22	138	1XFinFET
AΣΔM	9	26	1XFinFET

Kirchhoff law, the current adder is just connecting the currents to the same node (Figure 3-3a). The integrator is a capacitor (Figure 3-3b) whose voltage V_c is proportional to the integral of its current i_c. The current comparator is a CMOS inverter (Figure 3-3d) [71]. I_{in} greater than zero charges the input capacitor C_{gs} until V_{gs} is higher than the inverter threshold voltage, causing the output of the inverter to go low. On the other hand, I_{in} less than zero pulls the input to ground, hence the output will be high. Finally, the digital-to-analog feedback circuit, shown in Figure 3-3c, has two possible states. When the output is low, transistor M1 is off and the current is zero. When the output is high, transistor M1 is on and current is kI_{ref} where k denotes the feedback gain.

3.3 Modulator architecture

We propose to implement the AΣΔM shown in Figure 3-3e using the architecture described in Figure 3-3f. To adjust the time to start the integration cycle again, we use inverter chains. The capacitor used for integration is the comparator input capacitance C_i. Minimum size inverters are used to isolate the modulator from the output. We use a state of the art 1xFinFET process and simulate it using Cadence Spectre. The area and power metrics are summarized in Table 3.1. The modulator's power consumption depends on the input current I_{in}. When the input current is at its minimum or maximum, $\alpha = 0$ and the power consumed by the modulator is $6.8\mu W$. When the modulator operates at the natural frequency $\alpha = 1$ the power consumed is $47\mu W$. The calculated average power is $26\mu W$. For comparison we implement a LFSR based SNG in SystemVerilog and run synthesis and place and route using the cadence tool flow to obtain power and area metrics. Compared with a single LFSR based SNG implemented in the same technology the AΣΔM approach can save up to 81% in power consumption. In applications where more than one LFSR is used, we can replace each LFSR by a modulator thus similar savings are expected.

To improve the modulator's signal to noise response at the expense of an increment in latency and circuit area we add a 3 fF capacitor as the integrator. The AΣΔM is designed in Cadence-Virtuoso using a FinFet1X technology. Figure 3-4 shows the frequency and power consumption for the AΣΔM versus power supply Vdd. For comparison, we include the frequency and power consumption for a SNG used in SSC, for $Vdd = 0.7$ V. Figure 3-4 shows that the natural frequency (f_c) of the modulator and the frequency for the SNG are comparable. On the other hand, the power consumption, of the SSC-SNG is at least two orders of magnitude larger. Section 3.9 further compares the energy consumption for SSC, binary and SC-AΣΔM.

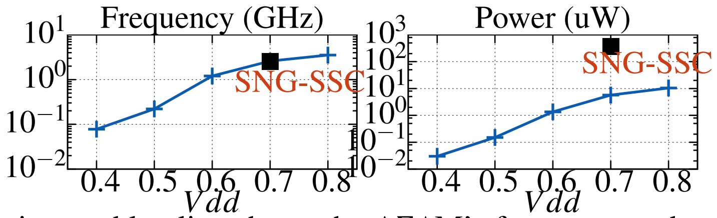

Figure 3-4: Continuous blue line shows the AΣΔM's frequency and power consumption versus Vdd. The black square shows the frequency and power consumption for the SNG used in SSC at Vdd=0.7 V.

3.4 Correlation between ΣΔ streams

In certain cases we can directly multiply two ΣΔ streams with the stochastic multiplier (AND/XOR gate). Figure 3-5 shows PDM streams x and y generated by identical ΣΔ modulators where $p_x = 0.6$, $p_y = 0.5$, and the output of an stochastic multiplier $p_{x\wedge y} = p_x p_y = 0.3$. The key for obtaining the correct multiplication is that T_x and T_y are relatively prime, similar to Pulse Width Modulated (PWM) streams running at different frequencies [72]. Contrary to PWM, the frequency of PDM streams is not constant but varies with the input (Equation 3.1), thus we cannot guarantee inharmonicity between streams for all the cases. For example, when $x = y$ (Figure 3-7a) the $p_{x\wedge y} = p_x = p_y$ instead of $p_x p_y$.

The accuracy for stochastic multiplication or other stochastic functions depends on how uncorrelated are the input streams. To quantify correlation between streams, we calculate the Stochastic Computing Correlation (SCC) [34] for streams x and y, with probabilities p_x and p_y:

$$SCC(X,Y) = \begin{cases} \frac{p_{x\wedge y} - p_x p_y}{min(p_x, p_y) - p_x p_y} & p_{x\wedge y} \geq p_x p_y \\ \frac{p_{x\wedge y} - p_x p_y}{p_x p_y - max(p_x + p_y - 1, 0)} & \text{otherwise,} \end{cases} \tag{3.2}$$

where the probability p is defined as $p = \frac{1}{T}\int_0^T x(t)dt$ and $x(t)$ is a stream of bits. Intuitively, the SCC indicates how similar streams are to each other. $SCC = +1$ means maximum similarity between streams. $SCC = -1$ means minimum similarity, and, $SCC = 0$ means

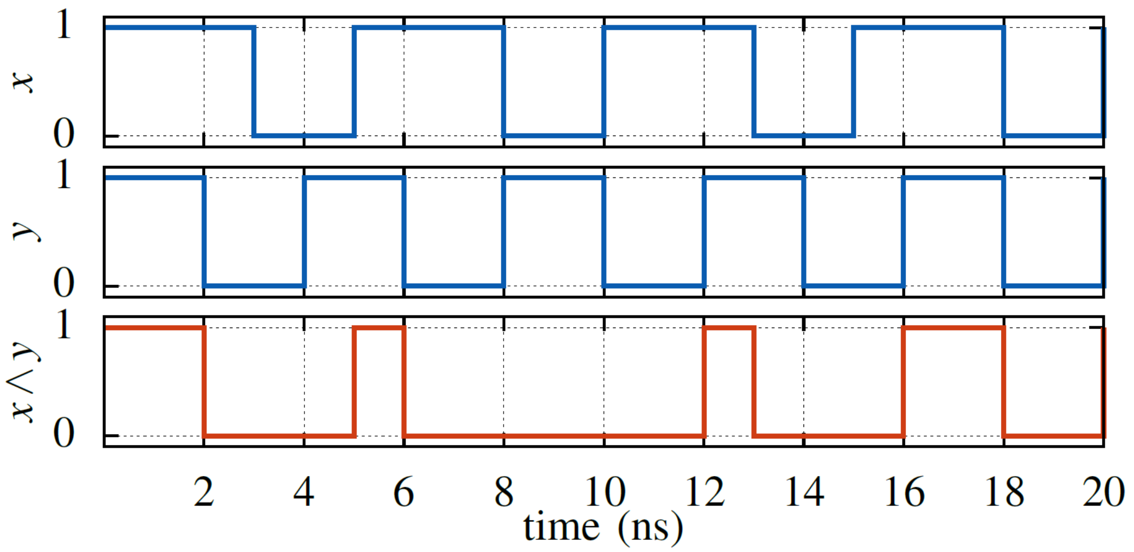

Figure 3-5: Stochastic multiplication with PDM streams for $p_x = 0.6$, $p_y = 0.5$ and $p_x p_y = 0.3$. For this example, the streams x and y are generated with two identical modulators.

the sequences are uncorrelated. In this section we explore how to make $SCC = 0$ for $\Sigma\Delta$ streams relying on: Process variations (Section 3.4.1) and shifting the modulator's natural frequency ω_c (Section 3.4.2).

3.4.1 Process variations

The random nature of process variations [73] inspires us to explore its potential as a source of randomness for stream generation. To that end, we run Monte Carlo (MC) simulations of two identical A$\Sigma\Delta$Ms connected to an AND gate as shown in Figure 3-6a (ignore for now V_{b1} and V_{b2}). The system was simulated using Cadence Virtuoso with the foundry provided MC models. The SCC was calculated with p_x, p_y, and $p_{x\wedge y}$ obtained as the average voltage at the output of each modulator and the AND gate respectively. Figure 3-7b shows x, y, and $x \wedge y$ streams for one MC run, where the streams are different enough so $p_{x\wedge y} \approx p_x p_y$. Figure 3-8 shows the MC results distribution for two examples: a) $p_x = p_y = 0.83$, and b) $p_x = p_y = 0.28$ with the corresponding $p_{x\wedge y}$ and SCC. Note that the mean (μ) for x and y in both examples is the expected p_x, p_y. The variance (σ^2) is less than $10mV$.

The multiplication error depends on the streams associated probability p, since a larger p result in a lower relative error. For example when $p_x = p_y = 0.83$ (Figure 3-8a), 77% of

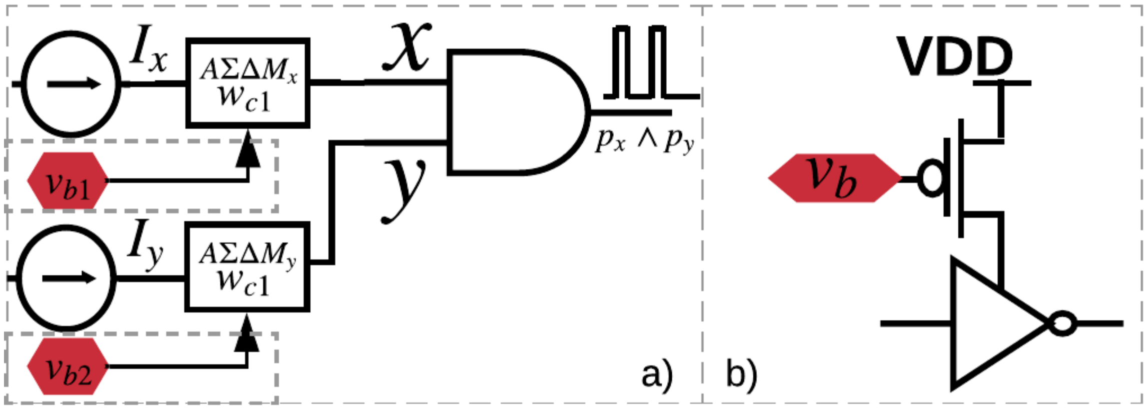

Figure 3-6: a) Circuit setup for correlation study. b) Starved inverter to control the delay in the AΣΔM, thus the natural frequency ω_c.

the samples have a multiplication error lower than 3% while only 3% of the samples have an error larger than 5%. On the other hand, when $p_x = p_y = 0.23$ (Figure 3-8b) only 20% of the samples have a multiplication error of less than 3% and 61% of the samples present an error larger than 5%. Those examples show different multiplication error for the same on-chip process variations.

In contrast with the multiplication error, SCC gives more cohesive results (Figure 3-8). In both examples, *SCC* indicates that from solely process variation we expect $\sim 50\%$ of the samples to have a correlation as low as $|SCC| < 0.1$ and $\sim 88\%$ of the samples have a $|SCC| < 0.3$.

Depending on the application, process variation alone might not lower the correlation between streams enough to yield an acceptable error. Figure 3-8a shows how process variation keeps the stochastic multiplication error between PDM streams $< 5\%$. On the other hand, Figure 3-8b shows that process variation lowers the correlation between streams but do not guarantee a low enough SCC. In the case of multiplying small stochastic numbers the requirement for $SCC \to 0$ is critical for a good stochastic performance and process variation alone do not suffice.

3.4.2 Frequency shifting

Although process variation cannot 100% guarantee $SCC \to 0$, it causes the stream's frequency to shift, which in turn reduces the correlation between streams, compare Figure3-7b (streams affected by process variation) with Figure3-7a (ideal case). Inspired by this observation, we can deliberately decrease *SCC* by using different natural frequencies ω_{c1}

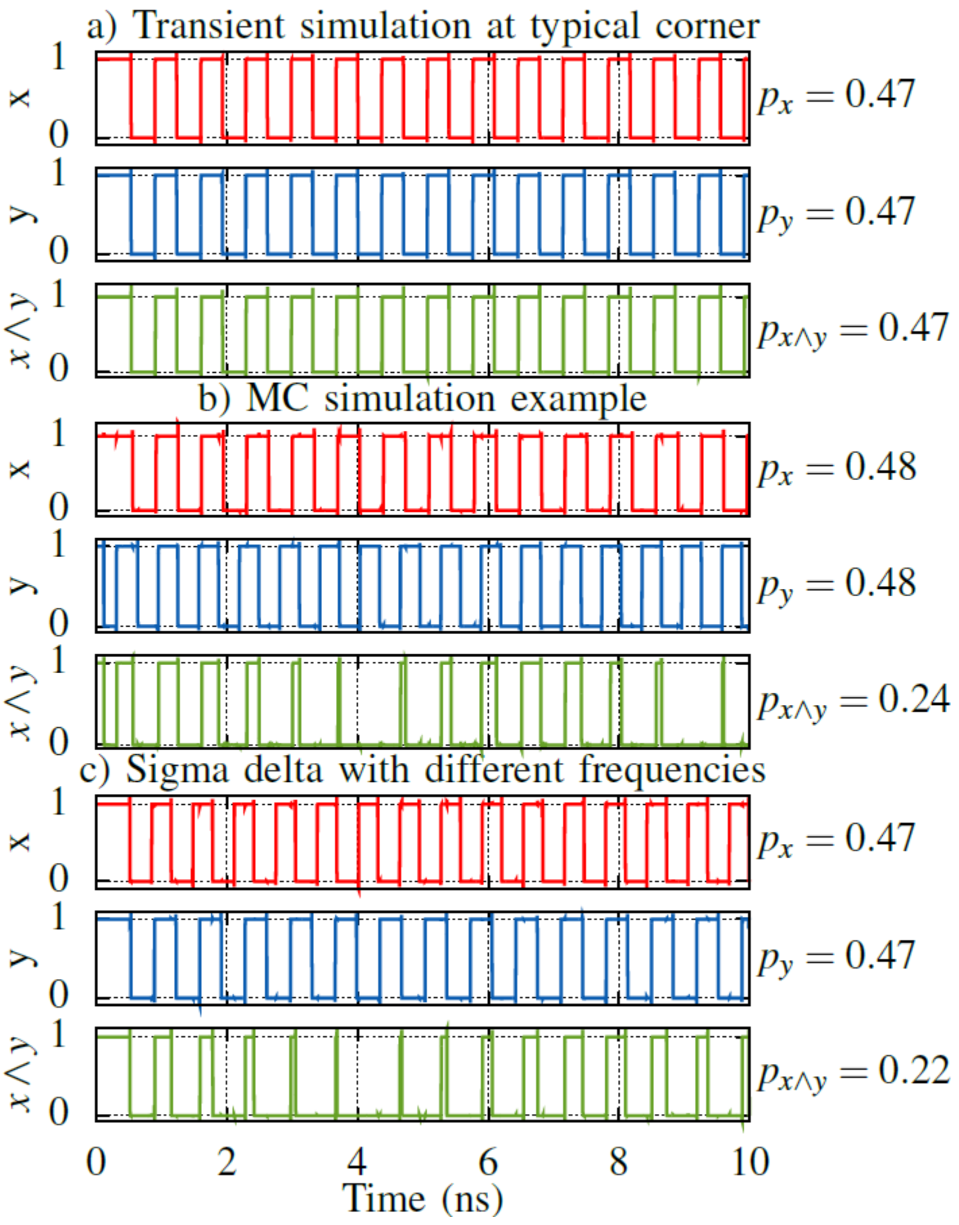

Figure 3-7: Transient simulation results showing ΣΔ Streams x and y generated with two AΣΔMs and the output of the stochastic multiplier (AND gate). a) Two identical modulators. b) Two identical modulators affected by process variation (MC simulation results) c) Two modulators set with different natural frequencies $\omega_{c1} \neq \omega_{c2}$

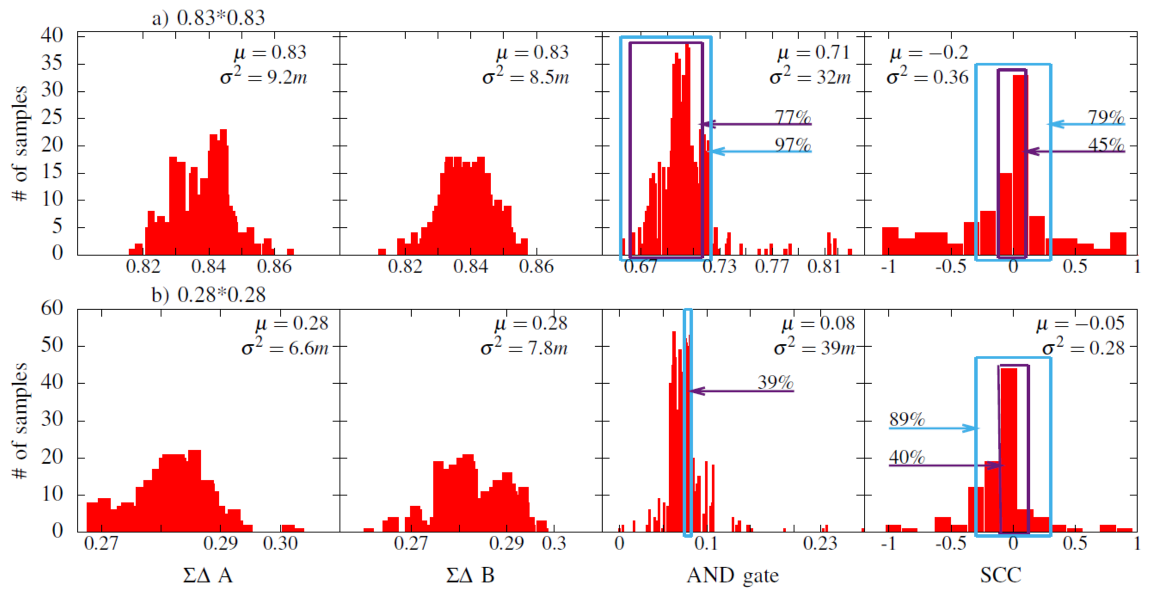

Figure 3-8: Example of Monte Carlo runs for a) $p_x = p_y = 0.83$ and b) $p_x = p_y = 0.28$. From left to right, the figures show p_x, p_y, $p_{x \wedge y}$ and SCC distributions.

and ω_{c2} for the modulators generating the streams x and y. To implement this, we replace the regular inverters in the AΣΔM delay chain (Figure 3-3f) for starved-inverters (Figure 3-6b) controlled by a bias voltage v_b. Different voltages for v_b result in different ω_c. Figures 3-9a and 3-9d show the frequency and δ of the AΣΔM set to operate at different natural frequencies ω_{c1}, ω_{c2}, ω_{c3}. The difference between V_{b1} and V_{b2} is $100mV$ and between V_{b1} and V_{b3} is $500mV$. Although the frequency of operation changes, the duty cycle δ (Figure 3-9d) which is related to the probability p or stream average (Figure 3-9c) remains unaffected.

Shifting the natural frequency ω_c of the modulators decreases the correlation between streams which is reflected in lower error levels for SC. We use the setup shown in Figure 3-6a with the modified AΣΔM including the variable delay chain. Figure 3-7c shows an example of streams generated with this setup for $v_{b1} \neq v_{b2}$. In this case, $SCC = 0.008$ and the $p_x \wedge p_y = p_x p_y = 0.22$. We repeat this simulation using 1000 randomly generated pairs of inputs, calculate the multiplication error as Error $= \frac{p_x p_y - p_{x \wedge y}}{p_x p_y}$ and get the error histogram envelopes (Figure 3-10a). We define $\beta = \omega_{c1}/\omega_{c2}$ as the ratio between the modulators natural frequencies. Notice that for modulator x and y in Figure 3-6a v_{b1} and v_{b2} are fixed, which in turn fixes ω_{c1} and ω_{c2}. With this we guarantee $\omega_{c1} \neq \omega_{c2}$ however

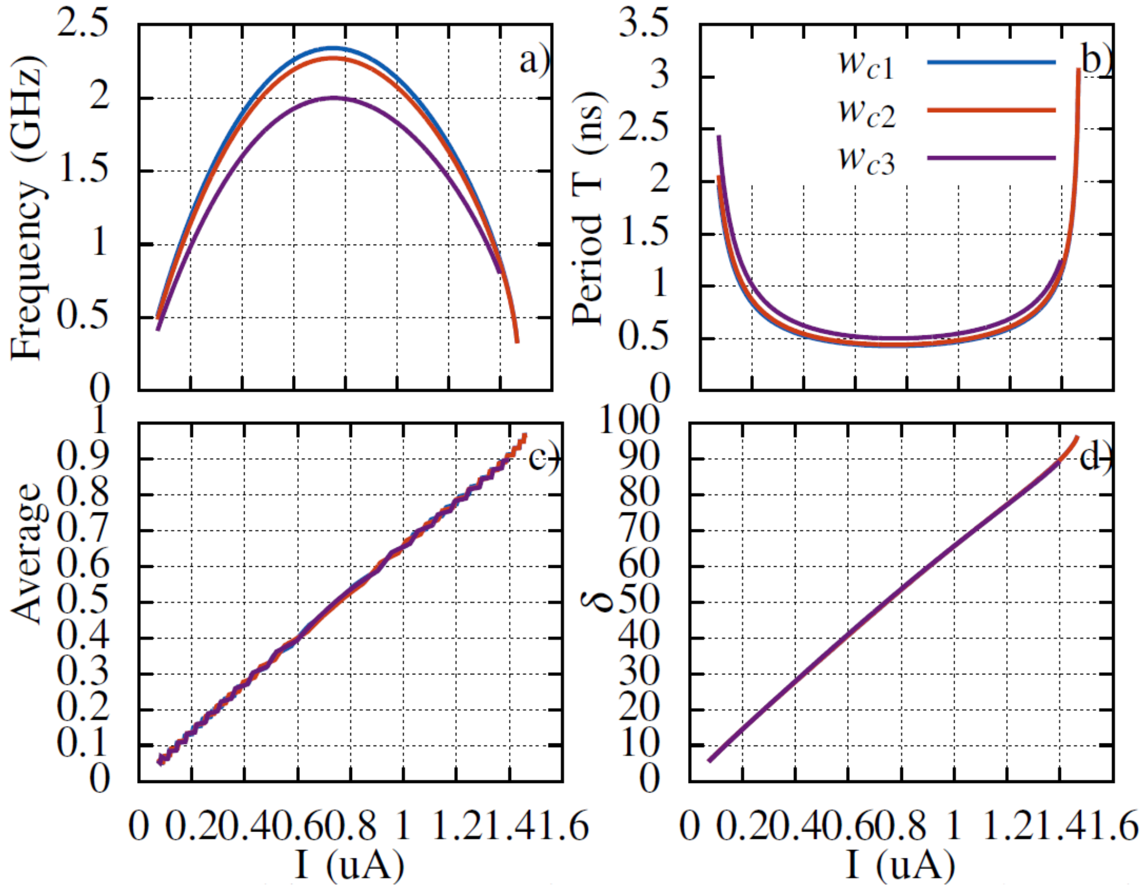

Figure 3-9: Modulator response to the input current. a) Frequency, b) Period, c) Average (p), and d) duty-cycle (δ)

the frequency ω for each PDM stream changes with the input as described in equation 3.1.

Our simulations confirm that using modulators with different natural frequencies reduces the multiplication error. Figure 3-10a, compares the error distribution for $\beta = 0.9$ and $\beta = 1$. When $\beta = 1$ we obtain a wider distribution whose μ is far from zero, caused by pairs where $\omega_{c1} \sim n\omega_{c2}$. On the other hand, for $\beta = 0.9$ the error distribution narrows down around zero, such that 80% of the samples have an error less than 0.04 and only 9% of the samples have an error larger than 0.1. The reduction in multiplication error when using different natural frequencies shows that PDM streams are good candidates for SC.

To evaluate how the error depends on the difference between modulations frequencies, we use a first order Verilog-A model to vary β within $(0-1]$. We randomly generate 1000 pairs of inputs and calculate the Root Mean Squared Error (RMSE) for each β (Figure 3-10b). We find that increasing the distance between ω_{c1}, and ω_{c2} reduces the error except for an increment observed when $\omega_{c1} = n\omega_{c1}$ or $T_{c1} = (1/n)T_{c2}$. Those peaks in the error result from an increase of the total number of cases with harmonic input frequencies. On the other hand, we observe a drop in the error even for ω_{c1} close to ω_{c2} ($\beta = 0.9$). Finally, the minimum error happens when $\beta = 0.1$ which increases the operation time.

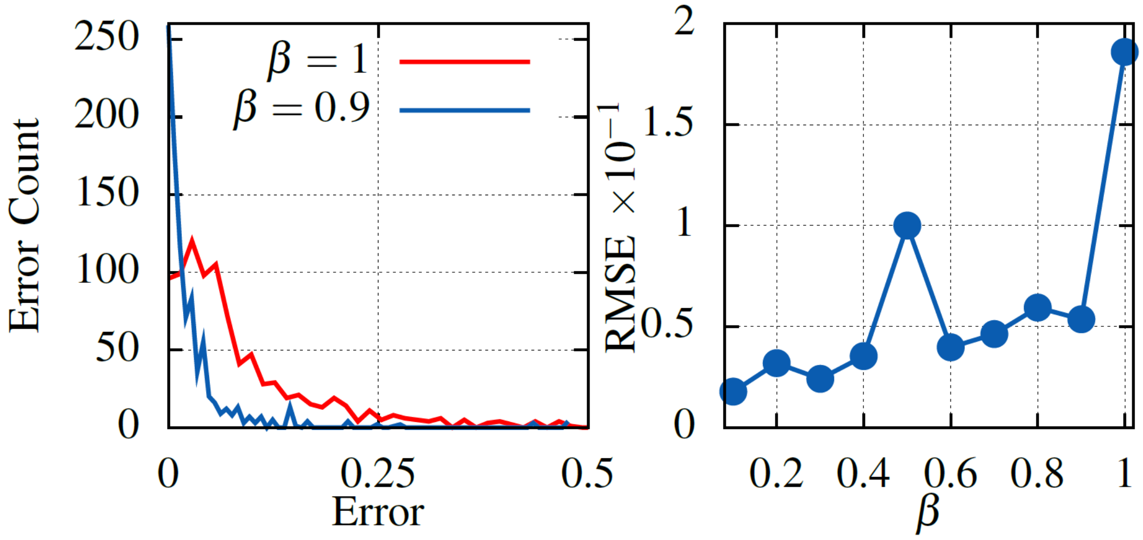

Figure 3-10: a) Error count for $\beta = 1$ and $\beta = 0.9$. b) RMSE in stochastic multiplication calculated varying β from 0.1 to 0.9 for 1000 randomly generated pairs. The maximum error happens when both A$\Sigma\Delta M$ have the same natural frequency ω_c.

The results in this section show that shifting ω_{c1} from ω_{c2} lowers the correlation between the streams enough so we can use SC techniques directly on PDM streams as generated with a $\Sigma\Delta$-Modulator. Even for ω_{c1} being very close to ω_{c2} we still obtain a RMSE of 0.05 for stochastic multiplication.

3.5 Evaluation results

In section 3.4 we explored how to lower the correlation between PDM streams. We found that setting $\omega_{c1} \neq \omega_{c2}$ the streams generated by two asynchronous modulators have a low SCC index. For the specific case of stochastic multiplication we showed that we reduce the RMSE by 70% without increasing the stream length (usually done in typical SC).

Stochastic computing has great potential for being used in several applications, and more complex SC applications require more than two SNGs. In these cases, we would want to replace each SNG by an AΣΔM. We envision that the significant power and area savings reported in section 3.2.3 still apply. To evaluate these ideas we implement the Gamma Correction algorithm. This algorithm requires the generation of 13 different streams. In the remaining of this section we will describe the implementation and explore the area, power, and energy metrics.

Gamma correction is a popular image processing algorithm used to code and decode luminance and tristimulus values, defined by the relation $V_{out} = V_{in}^{\gamma}$ [55]. We use a compact SC architecture of gamma correction based on Bernstein polynomials [2], depicted in Figure 3-11. The coefficients $b_0 = 0.0955$, $b_1 = 0.7207$, $b_2 = 0.3476$, $b_3 = 0.9988$, $b_4 = 0.7017$, $b_5 = 0.9695$, and $b_6 = 0.9939$ approximate $\gamma = 0.45$ [2]. The input x_i corresponds to each pixel in the image to be processed. To generate the coefficients we use 7 AΣΔMs. Since this is a multiplexer based operation, the correlation between the multiplexer inputs does not affect the output accuracy [74]. Thus, all the AΣΔM generating the coefficients are set to operate at the same natural frequency ω_{c1}. To generate x_i we use one more AΣΔM operating at a different natural frequency ω_{c2}. Non-correlated versions of x_i are created by using different delayed versions of x_i [10]. Notice that in this system we use 8 AΣΔMs in total, the streams generated from the pixels P_i and the coefficients b should have a low SCC thus we use two different natural frequencies $\omega_{c1} \neq \omega_{c2}$ as discussed in section 3.4. Two different bias voltages $v_{b1} \neq v_{b2}$ are used to enable this. All the stream frequencies therefore depend on the natural frequency of the modulators ω_{c1} or ω_{c2} which are constant and the input as described in equation 3.1 which varies with the corresponding coefficient or pixel value.

We implement this SC gamma correction architecture using a 1xFinFET technology and simulate the circuit using Cadence Spectre. The power and latency metrics are extracted from spectre level simulation results. For comparison, we also design using SystemVerilog and run synthesize and place and route for the LFSR-SC approach and the typical binary approach. To estimate the power, area, latency and energy metrics we use the post place-and-route results obtained from the Cadence Innovus (P&R tool). The metrics for the three implementations are summarized in Table 3.2. Figure 3-11 shows the original image, the output of the stochastic gamma correction circuit, and the Matlab gamma correction results for comparison. The calculated ERMS with our approach is only 2.54%.

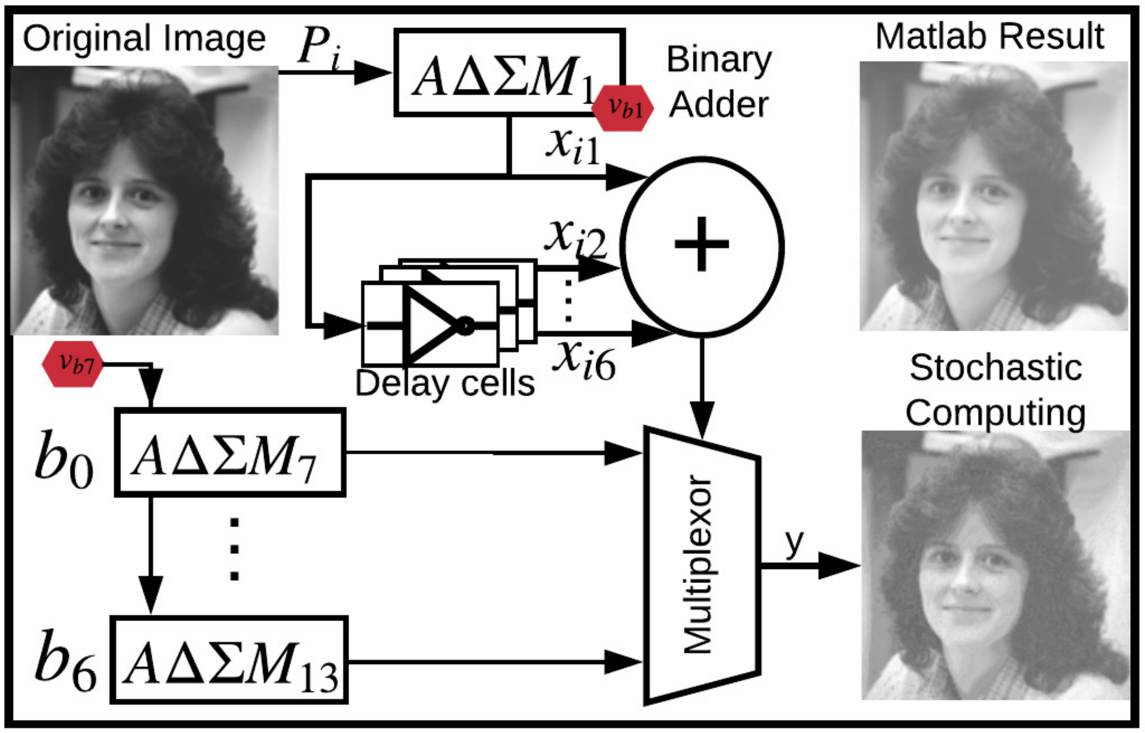

Figure 3-11: Gamma Correction architecture showing the original image, and for comparison, the image after applying gamma correction in Matlab and with our approach.

Table 3.2: Metrics for gamma correction. *At maximum frequency of operation.

Approach	Area (μm^2)			Gate Count			Power (μW)			Energy (pJ)	Delay (ns)	Technology	Latency (ns)
	core	SNG	total	core	SNG	Total	core	SNG	Total				
Conventional-Binary	72	-	145	422	-	422	824	-	824	0.49	0.5	FinFET	4.3
Stochastic+LFSR	5	185	190	13	362	375	51	1047	1098	224	204	FinFET	207.8
Stochastic+PWM [72]	76	678	754	-	-	-	-	-	1690	3	1.8	45nm	-
This Work	5	91	100	13	-	-	51	254	305	1.16	3.8	FinFET	3.8

Latency: We estimate 97% and 11% latency savings when compared with the SC-LFSR and the binary approach respectively. The savings result from using an analog to stochastic converter instead of the typical analog to binary to stochastic approach. Figure 3-12 compares the latency for the three approaches. t_{ADC} is the *ADC* conversion time, and t_b, t_{SC} are the binary and stochastic processing times. In SC-SD, the stream length (t_{ADC}) depends on the T for the modulators. While processing $\Sigma\Delta$ streams with SC only takes t_{ADC}, in a binary system processing a sample takes t_b longer.

Power: Operating on PDM streams as generated with the A$\Sigma\Delta$M saves power following the SC original intention. Table 3.2 shows that a typical SC-LFSR gamma correction design consumes up to 25% more power than its binary counterpart and 97% is dedicated to SNG. With our approach we replace the LFSRs with A$\Sigma\Delta$Ms. This approach lowers the power by up to 62% when compared with the binary counterpart.

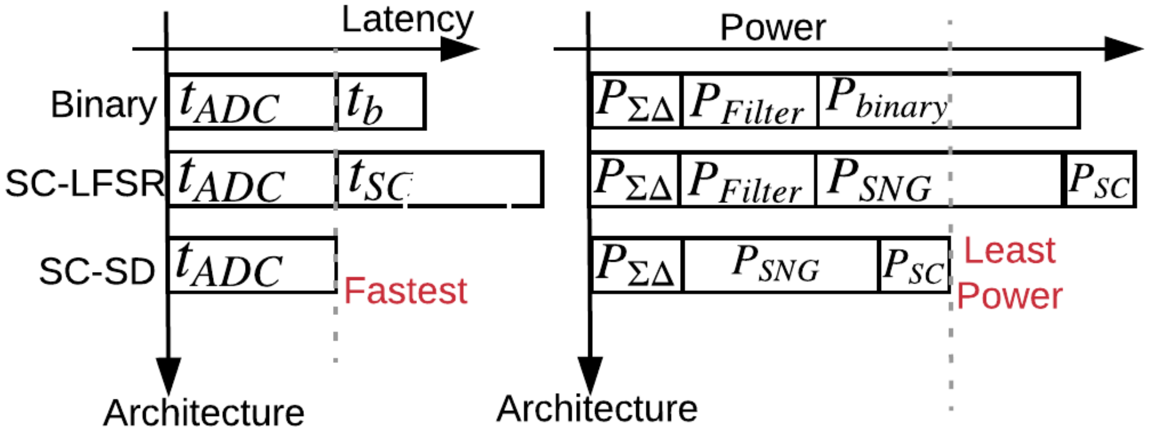

Figure 3-12: Power and latency considerations

Energy: In this work, we lower the power and total system latency. Thus, contrary to any other SC architecture, we expect energy savings in spite of processing streams instead of compact binary numbers. Simplified energy expressions for binary (E_{binary}), SC-LFSR (E_{SC}) and a SC-SD (E_{SCSD}) are:

$$E_{binary} = P_{\Sigma\Delta}t_{ADC} + P_{Filter}t_{ADC} + P_{binary}t_b \tag{3.3}$$

$$E_{SC} = P_{\Sigma\Delta}t_{ADC} + P_{Filter}t_{ADC} + P_{SNG}t_{SC} + P_{SC}t_{SC} \tag{3.4}$$

$$E_{SCSD} = P_{\Sigma\Delta}t_{ADC} + P_{SNG}t_{ADC} + P_{SC}t_{ADC} \tag{3.5}$$

where $P_{\Sigma\Delta}$, and P_{Filter} are the power associated with the modulator and filters in the ADC.

P_{SNG}, and, P_{SC} are the power associated with the SNG and the stochastic core and P_{binary} is the power consumed by a typical binary core. As usually presented in the literature, the energy reported in Table 3.2 for the binary approach only includes $P_{binary}t_b$, neglecting the energy associated with the analog to binary conversion. We estimate this neglected energy to be about $12pJ$ per clock-cycle. For this, we implement a 32-tap Finite Impulse Response (FIR) filter with one multiplier (typically used for ΣΔ ADCs) [75] using SystemVerilog. We run synthesis using the 1xFinFET library and find the energy associated with this filter. After the neglected terms are included, we find that SC-SD saves up to 90% in energy when compared with the binary counterpart.

3.6 Theoretical foundations of computing with asynchronous ΣΔ streams

Figure 3-6a shows a typical stochastic bipolar multiplier (XNOR gate) connected to AΣΔMs and Figure 3-13 shows a multiplication example. In this example, the AΣΔMs generate streams $x(t)$, $y(t)$ with $p_x = -7/8$, $p_y = 3/8$ and periods $T_x \approx 0.1\mu$s and $T_y \approx 0.4\mu$s. Since $T_x \neq T_y$, the two streams drift relative to each other over time. The drift pattern repeats at the least common multiple (LCM) of the periods T_p =LCM(T_x, T_y)= $1.28\mu s$ (marked by a green line in Figure 3-13), that is, the output of the XNOR gate is periodic with period T_p.

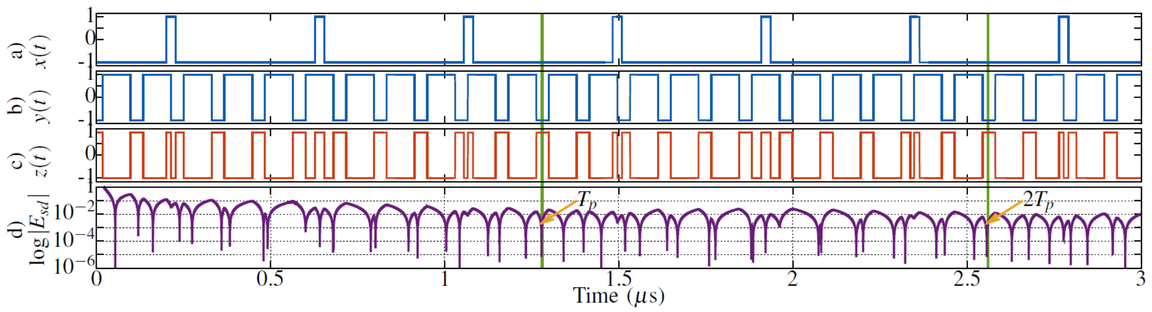

Figure 3-13: Continuous-time output streams of AΣΔMs for a) $p_x = -7/8 = -0.875$ and b) $p_y = 3/8 = 0.375$. c) output of XNOR gate $x(t) \odot y(t)$ and d) Multiplication error versus time. Example generated with ideal Matlab model.

To obtain the output of the multiplier p_z, we integrate the output stream $z(t)$ over the computing time t. Thus, $p_z(t) = \frac{1}{t}\int_0^t x(t') \odot y(t')dt' = \frac{1}{t}\int_0^t z(t')dt'$ and the computing error is $E_{sd}(t) = |p_z(t) - p_x p_y|$ (Figure 3-13d). The error decreases with time and becomes quite small. In the following sections we (1) quantify the magnitude of the error, that is, how small is small? and (2) study the error-latency trade-off. These contributions allow us to identify the necessary computing time given an accuracy requirement.

Although in this work we focus our efforts in the bipolar stochastic multiplier our results can be extended to other combinational stochastic logic as we will show in section 3.9.

3.7 Theoretical model for SC-AΣΔM

To find the asymptotic computation error for SC-AΣΔM, we note the AΣΔ streams periodicity and work in the Fourier space. For a stream $x(t)$, the Fourier series expansion is given by

$$x(t) = p_x + 4\delta_x \sum_{n=1}^{\infty} \operatorname{sinc}(n\delta_x) cos\left(2\pi \frac{n}{T_x} t\right), \tag{3.6}$$

where $\operatorname{sinc}(n\delta_x) = \frac{\sin(n\pi\delta_x)}{n\pi\delta_x}$, and the period T_x and duty-cycle δ_x depend on the modulator's input p_x (equation 3.1). The output of the XNOR gate $z(t)$ is the multiplication of the Fourier series expansions for streams $x(t)$ and $y(t)$. Therefore, the multiplication error E_{sd-F} can be expressed as

$$\begin{aligned} E_{sd-F}(t) = \frac{1}{t} \Bigg| & 4p_x\delta_y \sum_{m=1}^{\infty} \operatorname{sinc}(m\delta_y) \int_0^t \cos\left(2\pi \frac{m}{T_y} t'\right) dt' \\ & + 4p_y\delta_x \sum_{n=1}^{\infty} \operatorname{sinc}(n\delta_x) \int_0^t \cos\left(2\pi \frac{n}{T_x} t'\right) dt' \\ & + 8\delta_x\delta_y \sum_{n=1}^{\infty} \sum_{m=1}^{\infty} \operatorname{sinc}(n\delta_x) \operatorname{sinc}(m\delta_y) \\ & \left[\int_0^t \cos\left(2\pi \left(\frac{n}{T_x} + \frac{m}{T_y}\right) t'\right) \right. \\ & \left. + \cos\left(2\pi \left(\frac{n}{T_x} - \frac{m}{T_y}\right) t'\right) dt' \right] \Bigg|. \end{aligned} \tag{3.7}$$

E_{sd-F} oscillates over time, due to the mixing of cosine integrals that causes maximums and minimums that depend heavily on the signals' phase. However as t increases, the fluctuations die off as $1/t$, settling around an asymptotic error.

To find this asymptotic error, we observe that the first and second terms in (3.7) integrate to zero for multiples of T_x and T_y respectively. For the third term, $\cos(2\pi(\frac{n}{T_x}+\frac{m}{T_y})t')$ integrates to zero at $t = T_p = \text{LCM}(T_x, T_y) = NT_x = MT_y$ with N, M integers. However, $\cos(2\pi(\frac{n}{T_x}-\frac{m}{T_y})t') = 1$ when $\frac{n}{T_y} = \frac{m}{T_x}$ which occurs at multiples of T_p and pairs $(n,m) = (kM, kN)$ with $k = 1,2,3...$ Therefore, the asymptotic error for a given pair of $\Sigma\Delta$ streams $x(t)$, $y(t)$ is

$$E_{sd-F}(T_p) = E_{sd-Tp} = 8\delta_x\delta_y \sum_{k=1}^{\infty} \text{sinc}(kM\delta_x)\,\text{sinc}(kN\delta_y). \tag{3.8}$$

Equation (3.8) constitutes a fundamental limit to the achievable accuracy in SC-A$\Sigma\Delta$M. Contrary to SSC, as t increases the error does not go to zero. In many cases this error is negligible. For the example in Figure 3-13, $T_p = 1.28\mu s = 3T_x = 11T_y$ and $E_{sd-T_p} = 0.0023$ after adding phase adjustment. In other cases, such as $p_x = p_y$, the error is unacceptably large. The magnitude of this error is determined by δ_x, δ_y and (M,N). Since δ_x, δ_y depend on the inputs, we do not have control over them. However maximizing M,N for all cases minimizes the computation error as described in the following sections.

3.8 Latency-Error analysis for SC-A$\Sigma\Delta$M

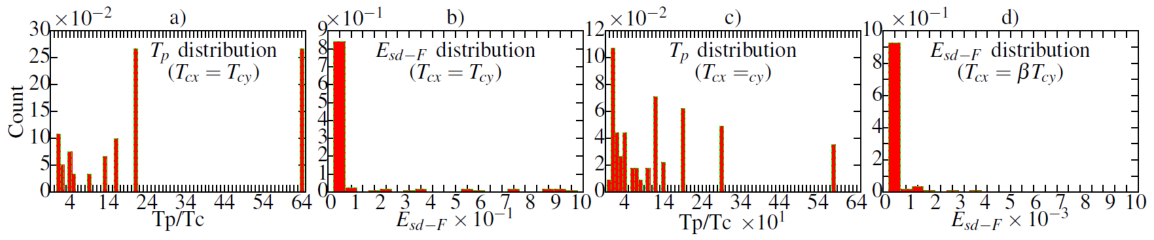

Figure 3-14: Histograms for a) Common period T_p distribution for identical modulators, that is $T_{cx} = T_{cy} = T_c$. b) Error distribution for identical modulators. c) Common period T_p distribution for modulators with different natural frequencies $T_{cx} = \beta T_{cy}$ and $\beta = 0.9$. d) Error distribution for modulators with different natural frequencies. In these figures we set r=16. Note scale changes from case to case.

In the previous section we found an expression for the asymptotic error E_{sd-T_p} for stochastic multiplication given two A$\Sigma\Delta$ streams. E_{sd-T_p} occurs at multiples of the common period T_p. To estimate the average multiplication error over all input pairs, we have to find all possible common periods given all possible inputs. Since, the number of inputs is infinite, to simplify our analysis, we assume p_x, p_y are rational numbers represented as fractions q_x/r and q_y/r respectively. Notice, that any real number can be approximated with a fractional expression; thus our assumption does not affect the generality of our analysis. r defines a grid for the possible numbers that can be represented. For example, $r = 8$ yields 8 different numbers which is equivalent to a resolution of 3 bits in typical binary computing or a stream length of 8 bits in SSC. We can express the stream's periods in terms of q_x, q_y and r using (3.1) as

$$T_x = \frac{r^2}{r^2 - q_x^2} T_{cx} \qquad\qquad T_y = \frac{r^2}{r^2 - q_y^2} T_{cy} \tag{3.9}$$

with $1/T_{cx}$ and $1/T_{cy}$ the natural frequencies of the AΣΔMs generating streams $x(t)$ and $y(t)$. With this, we can analyze the average error given an input set.

3.8.1 Identical AΣΔMs

If the two modulators are identical, that is under ideal conditions with no process-voltage-temperature variation, some input pairs yield an unacceptable large error. To understand which pairs results in this large error, we define T_c as $T_c = T_{cx} = T_{cy}$. Thus T_p is given by

$$T_p = \text{LCM}(T_x, T_y) = \frac{r^2 T_c}{\text{gcd}(r^2 - q_x^2, r^2 - q_y^2)} \tag{3.10}$$

with gcd the greatest common divisor. Figure 3-14a shows the distribution of all possible common periods for $r = 16$. In this case, the maximum T_p is $64T_c$. Figure 3-14b shows the distribution for the asymptotic error E_{sd-F} for $r = 16$. Although the error for most input pairs is concentrated around 0.025, some pairs result in unacceptably large error (close to 1). These high errors occur when $(N, M)s$ are small, for example, when T_x is multiple of T_y. In particular, when $T_x = T_y$, $(N, M) = (1, 1)$ the error can go as high as 0.97.

3.8.2 Shifting the natural frequency of the modulators

To address the low accuracy caused by small (N, M) pairs we can shift the natural frequency of one modulator [35]. To quantify this improvement, we define the common period for a pair of inputs as

$$T_p = LCM\left(\frac{b}{(r^2 - q_x^2)a}, \frac{1}{r^2 - q_y^2}\right) r^2 T_c \tag{3.11}$$

with $T_c = T_y = \beta T_x$ and the shift ratio $\beta = b/a$ a rational number. Figure 3-14c shows the distribution of all possible common periods for $r = 16$ and $\beta = 0.9$. Comparing Figure 3-14a (obtained with identical modulators) and Figure 3-14c, the distribution of common periods increases by an order of magnitude. This increment lessens the computing error, because the values for (N, M) increase as well. Figure 3-14d shows a reduction in the maximum error of at least an order of magnitude, eliminating the catastrophic error observed when the modulators operate at the same natural frequency.

Figure 3-15 shows the maximum common period versus β for different grids r. The βs that result in sharp spikes, e.g. $\beta = 1$, $\beta = 0.5$, should be avoided because small T_p causes large computing error. Furthermore, similar to SSC, increasing the resolution requirements increases computing time. For instance, to obtain the best case error for $r = 256$, the computing time should be larger than $10^6 T_c$.

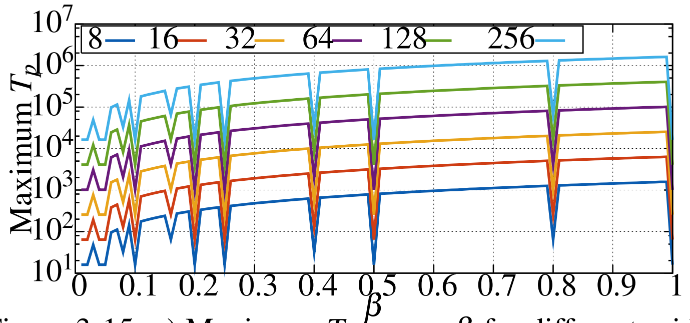

Figure 3-15: a) Maximum T_p versus β for different grids r.

3.8.3 Error-Latency trade-off

The previous sections discussed the fundamental limits for SC-AΣΔM accuracy. However, we made two unrealistic assumptions: 1) For any pair of inputs the common period is known *a priori*, and 2) the computing time can be extended to the longest T_p, which for $r = 256$ is in the order of $10^6 T_c$. Both assumptions are not practical for a circuit implementation. In this section, we analyze the error versus computation time for $t < T_p$ using SSC as a benchmark for the expected computing error.

Given an accuracy requirement, SC-AΣΔM yields at least 70% of latency savings compared with SSC. Figure 3-16a shows the error distribution for SSC (E_{ssc}) calculated for 16-bit-long streams. The root mean squared error (RMSE) is 0.24. For a fair comparison between E_{sd} and E_{ssc}, we obtain the computing time t for SC-AΣΔM that results in a similar RMSE. Figure 3-16b shows the error distribution for SC-AΣΔM with $t = 4.57 T_c$, the minimum common period. 70% of the samples result in $E_{sd-F} < 0.05$ and less than 1%

of the samples result $0.3 < E_{sd-F} < 0.45$ which increases the overall error. Nevertheless, the RMSE for SC-AΣΔM is comparable with SSC.

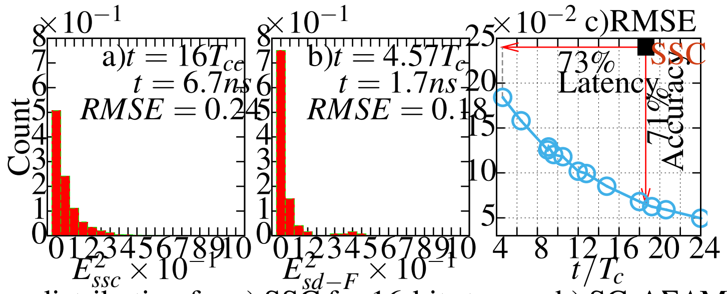

Figure 3-16: Error distribution for a) SSC for 16-bit stream. b) SC-AΣΔM for $t = 4.57T_c < T_p$. c) RMSE versus time for SC-AΣΔM compared with SSC for r=16. For these plots r=16 and $\beta = 0.9$.

Figure 3-16c shows the RMSE for SC-AΣΔM versus time calculated for $r = 16$. Compared with SSC, SC-AΣΔM yields 73% less latency for the same RMSE. Similarly, SC-AΣΔM yields 71% better accuracy when the computation runs the same time.

3.9 Circuit design

To validate our theoretical model, we simulate the setup shown in Figure 3-6 using Cadence-Spectre and calculate the multiplication error. Figure 3-17 shows an example for $x(t)$, $y(t)$, $z(t) = x(t) \odot y(t)$, $p_z(t)$ and the error components versus time. Approximating $T_x = 351$ ps and $T_y = 392$ ps the common period is $T_p = 6.7$ ns $= 19T_x = 17T_y$ (green lines in Figure 3-17).

Besides the error studied in the previous sections, the initial transient at the outputs of the modulators yields an additional error in the multiplication result. For the example in Figure 3-17, this transient lasts for $t_{ic} = 236$ ps. At this time (t_{ic}) the streams start a periodic behavior. The error due to this initial condition is given by $E_{ic}(t) = t_{ic}/t$ for $t > t_i$ and shown in Figure 3-17f. Only after t_{ic} the streams $x(t)$, and $y(t)$ can be described by 3.6. In consequence, the total computing error E_{sd} can be expressed as

$$E_{sd} = \begin{cases} |\pm 1 - p_x p_y| & \text{if } t \leq t_{ic} \\ |E_{sd-F} + E_{ic}| & \text{if } t > t_{ic} \end{cases} \tag{3.12}$$

with E_{sd-F} given by (3.7). Figure 3-17f shows the total error E_{sd} and the error obtained at multiples of T_p (E_{sd-T_p}) excluding the initial transient, both from simulations. We also

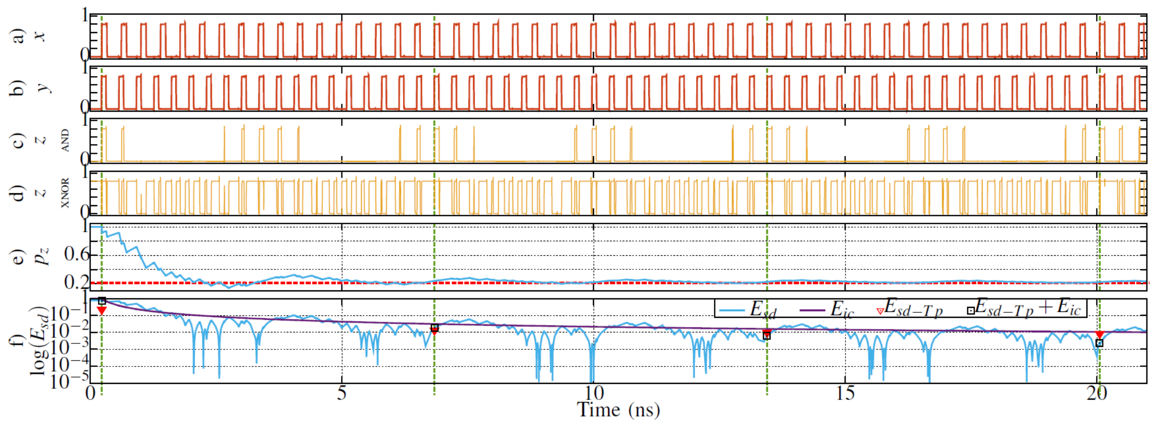

Figure 3-17: Spice simulation for stochastic multiplication with AΣΔ streams. The AΣΔMs are designed in cadence using a FinFET1X technology for $\beta = 0.9$. a),b) Outputs for the AΣΔMs generated with $p_x = p_y = -0.46$. c) AND gate output. d) XNOR gate output e) Estimated probability $\hat{p}_z$ at the output of the XNOR gate. The expected value is $p_z = 0.21$ f) Error for multiplication.

show the calculated E_{ic}. Notice that $E_{sd-Tp} + E_{ic}$ at multiples of T_p matches the total error obtained from the circuit simulation.

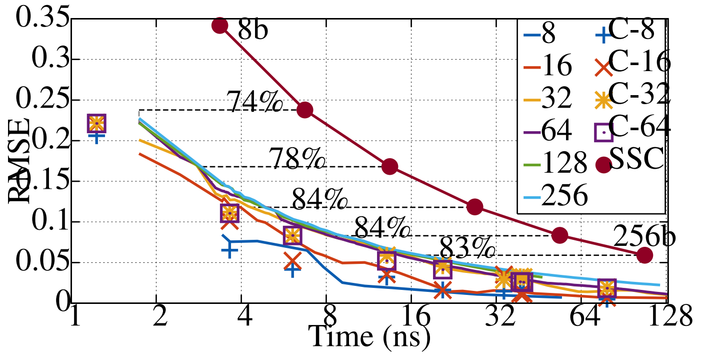

Figure 3-18: RMSE calculated for multiplication using all possible pairs of inputs given a grid r versus time with $\beta = 0.9$. Continuous lines shows the results from the ideal model (Matlab) ignoring initial conditions and phase shift between signals. Individual samples marked as “r Circ", are the results from Spice simulations including the initial condition effect. The figure includes the SSC error for comparison.

3.9.1 Error-latency trade-off for multiplication

To obtain a general estimate of the error we calculate the multiplication RMSE for all the possible pair of inputs in the grids given by $r = 8, 16, 32, 64, 128$ using our Matlab model (solid lines, Figure 3-18). For comparison, Figure 3-18 also shows with symbols the RMSE obtained from Spice level simulations for grids $r = 8, 16, 32, 64$. The slight difference between the Spice-level simulations and the ideal model is due to the effect of the initial conditions and phase between the multiplier input signals. We also show the RMSE for SSC since it is the error benchmark. Given the same RMSE, SC-AΣΔM yields at least 74% latency savings compared with SSC.

3.9.2 Error-latency trade-off for gamma correction

The results obtained for stochastic multiplication with AΣΔMs can be extended to other combinational logic. As an example, we implement gamma correction, an image processing algorithm used to code and decode luminance and tristimulus values, defined by the relation $V_{out} = V_{in}^{\gamma}$ [55]. We implement the compact SC architecture from [2] and [35] using a FinFet1X technology and simulate the circuit using Cadence Spectre. Figure 3-19 compares the original image, the output of the gamma correction algorithm as calculated with Matlab (golden case), and the results from the SC-AΣΔMs architecture obtained at different computing times. One interesting property for SSC is progressive precision or sketching. Since shorter sequences yields lower accuracy estimates of the computation, the accuracy can be dynamically adjusted without radical modifications in the hardware, that is just adjusting the computing time. This property has applications in image processing and computer vision [76] [77]. Figure 3-19 shows that similar to SSC, shorter computing times with SC-AΣΔM yields lower accuracy estimates of the computation. For our example we have an estimate of the result as early as 772ps. Also, increasing the computing time beyond 7ns does not yield significant improvements in the RMSE. Although, using the maximum common period for all possible combinations, results in computing times unfeasible large. the majority of common periods are still concentrated below 10ns.

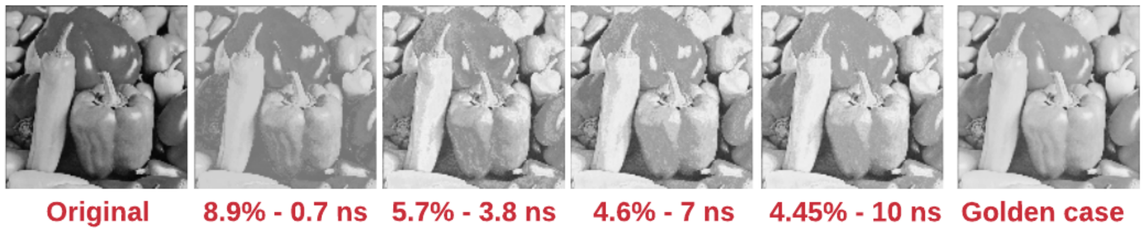

Figure 3-19: Gamma correction algorithm results obtained at different computing times.

3.9.3 Energy considerations

Table 3.3 summarizes the power, latency and energy metrics for the binary, SSC and SC-AΣΔM approaches using multiplication and gamma correction as examples. As expected the power savings at the core are more than 93%, at expense of latency and computing accuracy. For a fair comparison between the binary and the SC-AΣΔM architectures (Figure 3-1) we use the ADC metrics reported in [78] for a SAR architecture designed in a FinFET14 nm technology. The core and SNGs were synthesized using Cadence-Genus to operate at maximum frequency. All circuits operate at a power supply of 0.7V. Even though the latency for SC-AΣΔM is 5X-10X larger compared with the binary approach, we find energy savings of 89% and 79% for multiplication and gamma correction respectively. The energy savings are mainly due to the elimination of the ADC and the simplicity of the AΣΔM used to generate the streams.

Table 3.3: Energy, power and latency metrics for the conventional-binary approach, SSC and SC-AΣΔM. VDD=0.7V maximum frequency for circuits designed in FinFET1X technology. *ADC metrics are common for Binary and SSC

	Latency (ns)	Power (mW)	Energy (pJ)	Latency (ns)	Power (mW)	Energy (pJ)
	Multiplier			**Gamma Correction**		
Binary						
ADC* [78]	1.05	2.26	2.38	1.05	2.26	2.38
Core	0.58	0.17	0.1	0.5	0.82	0.41
Total	1.63	2.43	2.48	1.55	3.08	2.79
SSC						
SNG	108	0.73	79	204	1.05	214
Core	108	0.005	0.54	204	0.03	6.5
Total	109	3	82.1	205	3.34	223
SC-AΣΔM						
AΣΔM	16	0.01	0.2	7.3	0.04	0.33
Core	16	0.005	0.08	7.3	0.03	0.23
Total	16	0.016	0.27	7.3	0.07	0.56

3.10 Conclusions

SC is one of the most attractive paradigms for ultra-low power, error tolerant, approximate computing, since hardware expensive functions can be implemented with only $\sim 5\%$ of the area required for binary computing. However, SNG and the increment in processing time ultimately leads to more power, area and energy numbers. In this work, we propose asynchronous computing on PDM streams generated with a low power AΣΔM as an analog to streams converter. We design and simulate our architecture using an industry-standard 1xFinFET technology. Simulations show that by using our AΣΔM instead of a typical SNG scheme we obtain power and area savings of up to 81% and 60%. Since PDM streams can be highly correlated, we study on-chip process variations as a source of randomness for the stream generation. Based on these results we propose to set the different modulators in the system to run at different but fixed natural frequencies. Simulation results show 98%-11% savings in the total system latency, and, 50%-38% in power savings when compared with the binary counterpart or the previous stochastic approaches.

We also analyze the error-latency trade off for SC-AΣΔM multiplication using the Fourier series expansion for the ΣΔ-streams. Our theoretical model demonstrates that there is an asymptotic error or bias for SC-AΣΔM. For the majority of inputs this bias is small, yielding superior accuracy compared with SSC. This bias depends on the relation between the streams' periods T_x, T_y and it is obtained at $T_p =$LCM(T_x, T_y). However, T_p can be unfeasibly large, thus we analyze the error-latency trade-off for SC-AΣΔM using SSC as the benchmark. To validate the theoretical model we use Spice-level circuits simulations using a FinFet1X technology. Our model and simulations results show that multiplication with SC-AΣΔM yields at least 70% of latency savings obtaining the same SSC accuracy performance, or 70% better accuracy when the computing time is similar to SSC. Although the latency is 5X-10X longer when compared with the binary approach, the power efficiency of SC-AΣΔM yields savings in energy of at least 79% in gamma correction. The energy savings make the approach ideal for the constrained conditions of the IoT sensors.

Chapter 4

Applications

Computation with continuos time streams has been limited to single-stage combinational logic, because more complex algorithms require the incorporation of a memory element. In SSC the memory elements are just flip-flops, which unfortunately require synchronized streams. Yet, algorithms like the FFT, one of the most used and expensive operations in digital signal processing, could benefit from the ACS savings in area, power and latency. Therefore, we need to find an effective, yet inexpensive way to incorporate a memory element in asynchronous computing units.

In this chapter we first introduce a digital implementation of a Fast Fourier Transform accelerator used for wheezing detection. This, help us to describe the limitations of the ultra-low power digital paradigm. With the lessons learned from the typical implementation, we delve into the ACS based FFT implementation and an ACS low-error adder. Finally we present the ACS-NAM architecture for random forest inspired machine learning algorithms.

4.1 An ultra-low-power dual-phase latch based digital accelerator for continuous monitoring of wheezing episodes

A wheeze is a continuous lung sound that if continuously monitored can provide crucial information for medical diagnosis and treatment of Asthma or Obstructive Pulmonary Diseases (OPD). Although the Internet of Things (IoT) has the potential to enable wheezing continuous monitoring, an IoT device must be extremely energy-efficient since it is powered from small batteries or even harvested energy. Some prototypes for wheezing detection have been demonstrated in Field Programmable Arrays (FPGAs) [79], however,

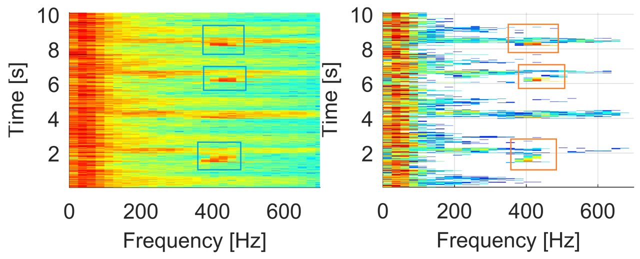

Figure 4-1: Example of a Spectrogram obtained after STFT calculation. a) Matlab results b) Our accelerator results.

the power consumption for these is still two or three orders of magnitude higher than what can be supported by harvesting energy from the environment. In this section we present an ultra low power accelerator to generate the STFT representation of the respiratory signal used to monitor and detect wheezes. The input for our accelerator is a respiratory signal recording that can be obtained by placing a microphone in the trachea region [80]. A wheeze in this signal, is a superimposition of sinusoidal sounds that last more than 150 ms and are located beyond 100 Hz in the frequency domain. To process the respiratory signal, the STFT is used to generate the spectrogram shown in Fig. 4-1a for further analysis [81]

4.1.1 Hardware optimizations

A block diagram of the accelerator is depicted in Fig. 2a and it is composed of a Fast Fourier Transform (FFT), input, and output controllers. The three controllers share access to 4 multipliers and three memory banks that are multiplexed in time to reduce the energy consumption due to leakage current associated to idle components.

The input controller multiplies the respiratory signal with a set of programmable coefficients for windowing and takes care of window overlapping. The output controller reads the FFT results from the memory banks and uses the multipliers to get the square of the Power Spectral Density (PSD^2). The FFT controller is used to calculate the Fast Fourier Transform for each window. A third memory bank A1 is included to keep the overlapped section of the data for the next window calculation. The register banks are implemented with positive level latches [82].

The rest of this section describes other key optimizations implemented in this work:

Architecture Width: We choose a word width of 13 bits for internal FFT calculations and an input word width of 10 bits to avoid overflow errors. Simulations show that a sampling frequency of 1.5 KHz with an FFT core of 64 points is sufficient to extract the wheezing information.

Near Threshold Operation: The switching energy can be calculated as $E_{active} = \alpha C Vdd^2$, where ($\alpha$) is the activity factor, (C) is the switched capacitance and Vdd is the power supply. Since the energy consumption is proportional to Vdd^2 we lower Vdd up to the near-threshold region where we found the minimum energy operation point for our accelerator.

Dual Phase Implementation: Lowering the Vdd, although powerful for energy reduction, increments the effects of Process, Voltage and Temperature (PVT) variations by up to 20X [83]. To address the variation effect, without increasing the energy consumption we adopt a dual phase latch based design style [52]. To implement our accelerator we convert flip flop registers into their master and slave latches and use commercial tools for time borrowing optimization.

FFT core adaptation: Since the Fourier Transform is one of the fundamental operations in digital signal processing a lot of work has been done to optimize its performance. In this work we adopt the implementation described in [84] which presents an efficient addressing scheme for "in-place" memory access. This implementation, however, requires delays between the write and read addresses to compensate the butterfly computation time. These delays are a hazard when operating at low Vdd because of the increased PVT variation effect. For our implementation, we take advantage of the dual phase clocking scheme. The address generated by the FFT core is buffered during the positive level of Clock 1 and is used for reading data from the memory asynchronously. This data will be used by the FFT core to perform the butterfly calculation during the positive level of clock 2.

4.1.2 Results and discussion

We build our accelerator using SystemVerilog, and synthesize it using a GF-130nm library optimized for sub-threshold operation. We compare our dual phase latch based accelerator with a regular flip-flop implementation [84]. We simulate the netlist after place and route for $Vdd = 0.3V - 0.8V$ and find the maximum operating frequency per each VDD point.

Figure 4-1 compares the output of the the STFT accelerator (4-1b) with the Matlab spectrogram function (4-1a). To calculate the PSD^2 we scale the FFT results before the final multiplication. This, avoids overflow in this last stage of the calculation. As a result, a good portion of the spectrogram is zeroed in discarding only the non wheezing related information.

Figures 4-2c and 4-2b compare the Energy vs. Vdd for the control logic and register files. The control logic shows up to 15% in savings for voltages lower than $0.6V$. Below

0.6V a wider shoulder between clock phases is used to compensate for longer delays due to the reduced power supply. Therefore there is no gain in performance but savings in energy due to the absence of clock buffering and gates up-sizing. On the other hand, for voltages above 0.6V the shoulder required between clock phases is reduced. For this reason, there is an increment in performance of up 25% ($Vdd = 0.8V$) and the energy in the latch based implementation moves towards the flip-flop based implementation.

Figure 4-2b compares the energy consumption of the flip-flop memory bank with the latch memory bank. When using latches for the memory banks, we get up to 80% savings in energy consumption. Since latches have around 35% less transistors than flip flops, latches leak less current. Also, we see less active energy for the latch based register file. In this case, the enable signal (G) going to the latch is not switching for every clock cycle as it is the equivalent clock (CK) signal in the flip-flop. To improve the flip-flop bank energy consumption clock gating per row can be implemented [82]. Figure 4-2b shows that clock gating, decreases the energy consumption at higher voltages when the active energy dominates but increases the energy consumption at lower voltages when leakage is dominant.

Figure 4-2d compares the total energy for both implementations including control logic and register banks. The total energy savings when including memory banks and control logic can be up to 50%. The major contributer to the energy savings is the latch based memory included as part of the design.

4.1.3 Conclusion

In this section we present an ultra low power accelerator for STFT calculation, optimized for wheezing detection. We implement a two phase latch based scheme to counteract PVT variation in the sub-threshold and near-threshold regime. The accelerator consumes 3.3 pJ/cycle, and 155 nJ/FFT at the minimum energy point located at 0.5V with a maximum operating frequency of 163 KHz. To evaluate its performance we compare it with a flip flop implementation and we save up to 50% in energy consumption. We take advantage of the dual phase clock timing of our accelerator to read the data from memory during clock 1 and write data to memory during clock 2, avoiding the need to add hazardous delays between the read/write memory address buses while we still have a simplified "in place" memory address generator.

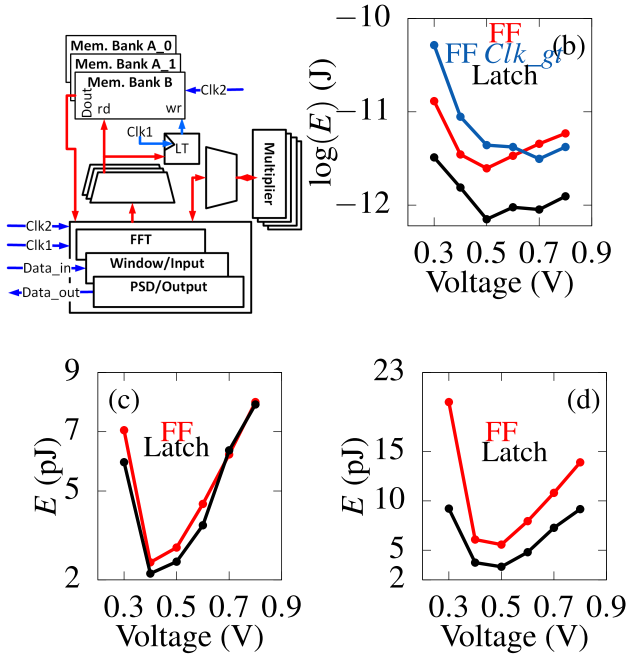

Figure 4-2: a) Block Diagram for the accelerator. b), and, c) Compares the Energy vs. VDD for the accelerator control logic and memory banks. d) Compares Total energy of the dual phase latch vs flip-flop implementations

4.2 ACS-FFT: Area-Efficient Low-Latency FFT Design Based on Asynchronous Stochastic Computing

Computation on asynchronous streams has been limited to single-stage combinational logic, because more complex algorithms require the incorporation of a memory element. In SSC the memory elements are just flip-flops, which unfortunately require synchronized streams. We need to find an effective, yet inexpensive way to incorporate a memory element in asynchronous computing units.

In this section, we propose to use a capacitor embedded in simple feedback loop as the asynchronous memory element. Based on this idea, we design a low-error asynchronous adder that is able to keep track of the addition carry. Then, we implement an ASC-FFT architecture based on this adder using a FinFET1X technology. We compare its accuracy, area, and latency, against conventional FFT architectures and find that the ASC-FFT shows better area-delay performance than pipelined, and minimum hardware FFT architectures

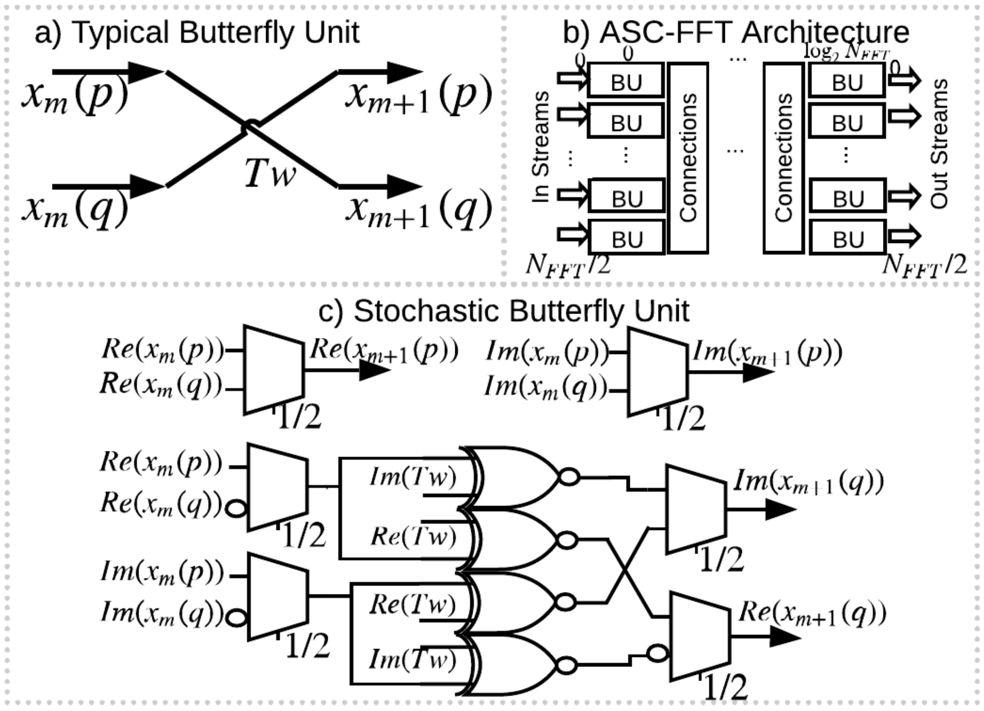

Figure 4-3: a) Buttterfly unit as described in equation 4.2 b) ASC-FFT architecture c) Stochastic Implementation of the BU

for $N_{FFT} \geq 32$ and $N_{FFT} \geq 128$ respectively, where N_{FFT} is the FFT size.

4.2.1 Asynchronous Stochastic Computing FFT Architecture

The radix-2 FFT is a multistage algorithm based on butterfly units (BUs) described by:

$$x_{m+1}(p) = x_m(p) + x_m(q) \tag{4.1}$$

$$x_{m+1}(q) = [x_m(p) - x_m(q)]\, e^{-j\frac{2\pi}{N}nk} \tag{4.2}$$

and shown in Fig. 4-3a, where $m = 1, ..., \log_2(N_{FFT})$ labels the stage, $x(p)$ and $x(q)$ label the inputs for a particular BU, N_{FFT} is the FFT size, $x_{m+1}(p)$ and $x_{m+1}(q)$ are the BU complex outputs. Finally, $e^{-j\frac{2\pi}{N_{FFT}}nk}$ are the twiddle (Tw) coefficients with $k, n = 0, ..., N_{FFT} - 1$. Directly mapping conventional multipliers and adders to the typical stochastic versions (Fig.4-3c) yields poor SNR due to the adder output being divided by two which we refer as the adder scaling effect [85]. The authors in [38] present a SSC-FFT, which achieves better SNR response due to a non-scaling adder whose output result is $p_x + p_y$ instead of $(p_x + p_y)/2$. This adder, relies on a digital counter (flip-flops) that keeps track of carry bits. Unfortunately, this SSC-FFT implementation still suffers from the drawbacks described before, expensive SNG and long latency. To address these challenges, we propose an efficient non-scaling asynchronous adder for CTS that exploits the simplicity of a current based design approach.

Asynchronous non-scaling adder

Fig. 4-4a shows an example of SSC streams addition. A(t)+B(t) at each clock cycle can be: ① N+N, ② N+P=0 or ③ P+P. In case ③(①) the output stream can only take one P(N). The other P(N) is accumulated in a clocked counter as a carry. A carry can be released when case ② occurs and the counter decreases (increments) by one [38] [39] [40]. For ASC streams the time the inputs are in a given state {N, P} is not quantized, thus counting carry bits would require an unfeasible fast clock. Instead, we propose to use an integrator that accumulates “time" when ③(①) and decrements when the carry “time" is sent to the output.

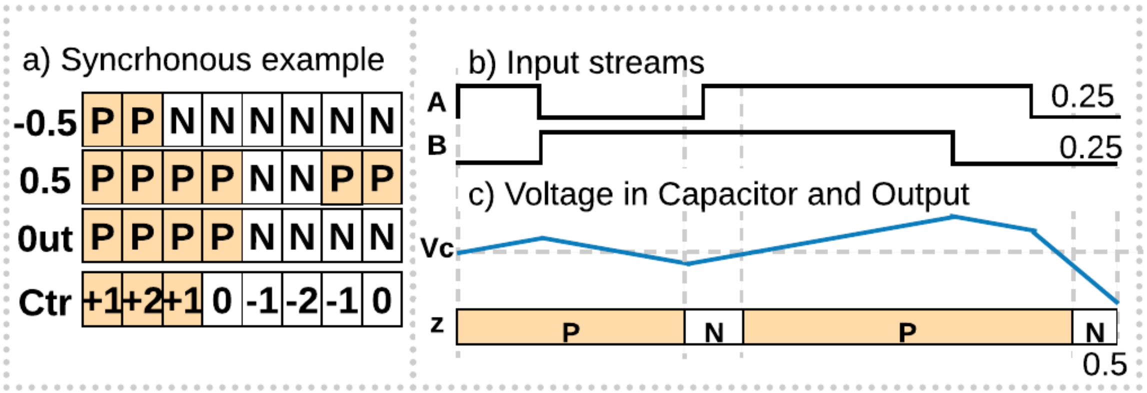

Figure 4-4: a) Example of non-scaling synchronous addition with digital integrator. b),c) Example of asynchronous adder, input streams (b) and evolution of the voltage in the capacitor and output stream (c).

This adder can be implemented with the block diagram in Fig. 4-5a. The integrator output can be expressed as:

$$y(t) = y(t_o) + \frac{1}{k}\int_{t_0}^{t}\Big(A(\tau) + B(\tau) - y_f(\tau)\Big)d\tau \tag{4.3}$$

where $y(t_0)$ is the initial condition on the integrator and k is the integrator constant. The integrator output is converted to an asynchronous stream by a quantizer. If $y(t) > Thr$, $z(t) = \text{P}$, otherwise $z(t) = \text{N}$. $y_f(t)$ is the output of the system converted back to the input units.

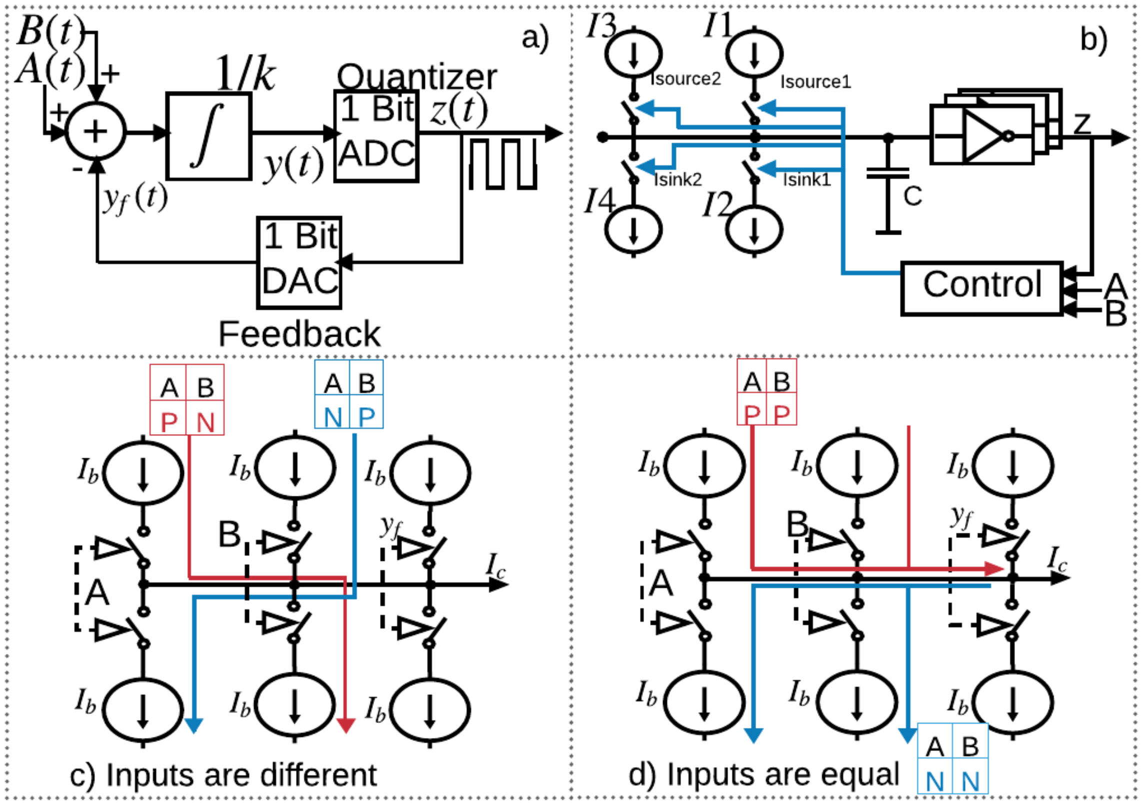

Figure 4-5: Proposed asynchronous adder. $A(t)$ and $B(t)$ are the continuous time input streams and $Z(t)$ is the continuous time output stream. a) Block diagram. b) Architecture. c), d) Current adder theory. c) Current flux when inputs are PN or NP. d) Current flux when inputs are PP or NN.

The main SC advantage is the computing units low hardware cost (Fig. 1-1). To keep this advantage, we adopt a current based methodology, eliminating the need of switched capacitors and high gain active components like operational amplifiers. The adder's architecture is depicted in Fig. 4-5b. To implement the addition of the inputs and the feedback path, we adopt a current based adder. By Kirchhoff law, this adder is just connecting the currents to the same node. Fig. 4-5c, d shows one current mirror per input stream $A(t)$, $B(t)$ and a third current mirror for $y_f(t)$. Fig. 4-5c shows the current flow when the inputs are PN (NP). In this case the net current is zero and the input for the integrator depends entirely on $y_f(t)$. On the other hand, Fig. 4-5d shows the current flow when the input streams are either PP or NN. In these cases, the current from $A(t)+B(t)$ is $\pm 2I_b$ and the input to the integrator can be $\pm I_b$ or $\pm 3I_b$ depending on $z(t)$. The current integrator is implemented as a capacitor given its current-voltage relation $V_c = \frac{1}{c}\int I_c dt$. The current-comparator can be implemented as an inverter [71].

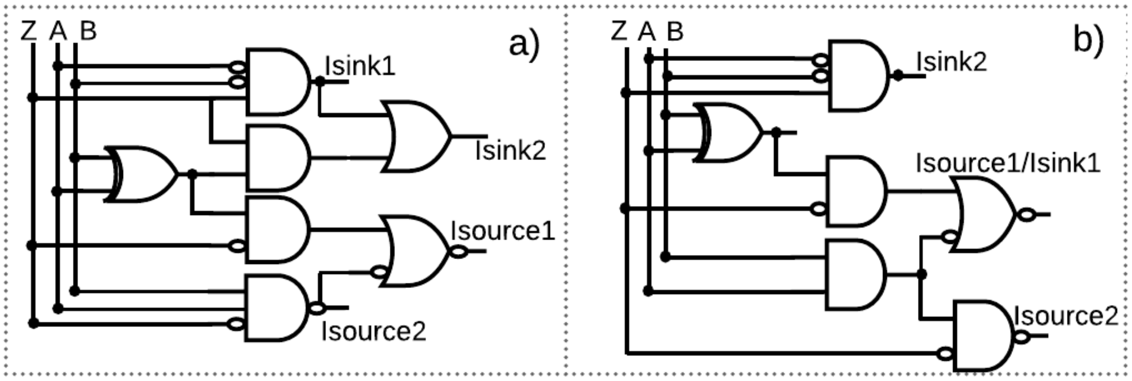

Figure 4-6: Control logic for scaling (a) and non-scaling (b) adder.

Fig. 4-4b shows the adder operation. In this example $A(t)$ and $B(t)$ are streams with associated probability of $p_A = 0.25$ and $p_B = 0.25$. Fig. 4-4c shows the evolution of the voltage in the capacitor V_c (integrator) and the output of the comparator $z(t)$. The slope for V_c is given by the integrator input current and can be either ± 1 or ± 3. Table 4.1 shows the slope for all the different combinations of $A(t)$, $B(t)$ and the output of the modulator $z(t)$. Based on this we can reduce the number of current supplies from 6 to 4 as shown in Fig. 4-5b. When the slope is +3 (-3) current sources I1, I3 (I2, I4) are active at the same time. The control logic to activate the current sources is shown in Fig. 4-6a which is implemented based on Table 4.1.

Asynchronous scaling-adder

We can implement a low-error scaling adder ($\frac{A(t)+B(t)}{2}$) following the same reasoning used to design the non-scaling version. Based on the implementation in Fig. 4-5b, we modify the capacitor charging slopes to ± 2 and ± 1 depending on $A(t)$, $B(t)$ and $z(t)$. To adapt for the new slopes I1 and I2 become I_b instead of $2I_b$.

Fig. 4-7 shows the scaling adder operation. In this example $A(t)$ and $B(t)$ are streams with associated probability $p_A = 0.25$ and $p_B = -0.25$. Fig. 4-7c shows the evolution of

Table 4.1: Truth table for control signals for the asynchronous adder.

Z	Inputs		Slope		Active Source	
	A	B	Scaling	Non-Scaling	Scaling	Non-scaling
N (P)	N (P)	N (P)	0	-1 (1)	None	I2 (I1)
N (P)	P (N)	P (N)	2 (-2)	3 (-3)	I1,I3 (I2,I4)	I1,I3 (I2,I4)
N (P)	x	x	1	1 (-1)	I1 (I2)	I1 (I2)

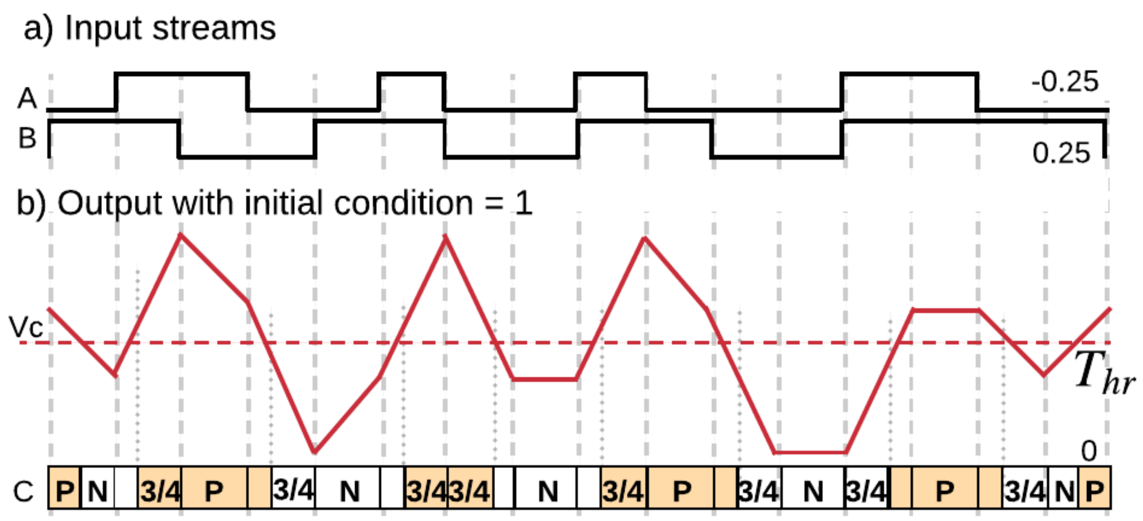

Figure 4-7: Example for average adder. a) Input streams $A(t)$ and $B(t)$. b) Voltage in the capacitor and output stream for initial condition $V_c > Thr$

the voltage in the capacitor V_c (integrator) and the output of the comparator. For different initial conditions, the evolution of V_c is different but the expected result is correct. Notice that if the input streams are PP(NN) and the output is P(N) there is not accumulation in the integrator. On the contrary, if the output stream is N(P) the integrator accumulates the input current until the comparator switches to P(N).

Asynchronous FFT Architecture

We replace the multiplexer based stochastic adder with our asynchronous adders and implement the FFT architecture shown in Fig. 4-3b. We scaled the outputs at each FFT stage by two to avoid overflow errors, and both inputs and twiddle coefficients are CTS generated with AΣΔM. In the following section, we compare the area and delay performance for the proposed ASC-FFT with SSC-FFT and typical FFT pipelined (optimized for throughput) [86] and one-BU (optimized for area) [84] architectures .

4.2.2 results

FFT SNR Perfomance: We implement the ASC-FFT architecture (Fig. 4-3b) using a FinFET1X technology. We run SPICE level simulations and calculate the average and the FFT SNR for the resulting streams. Fig. 4-8a shows the SNR versus N_{FFT} for different SC-FFT implementations (SSC [38], un-even, and Direct [85]). The direct BU SSC implementation is not practical given the quick performance degradation as N_{FFT} increments [85]. The SNR performance for the ASC-FFT is better than the un-even SSC-FFT architecture for $N_{FFT} \geq 8$ when the scaling degrading effect on the SNR is more notoriuous for the un-even architecture. Fig. 4-8b shows the ASC-FFT SNR versus delay for $N_{FFT} = 8, 16, 32, 64$. Increasing the computation time has a positive effect on the overall computation accuracy. However, to obtain an accuracy comparable with SSC-FFT architectures ($SNR \approx 20dB$) the ASC-FFT requires 3X less computing time. Table 4.2 summarizes the performance results for the ASC adder and FFT.

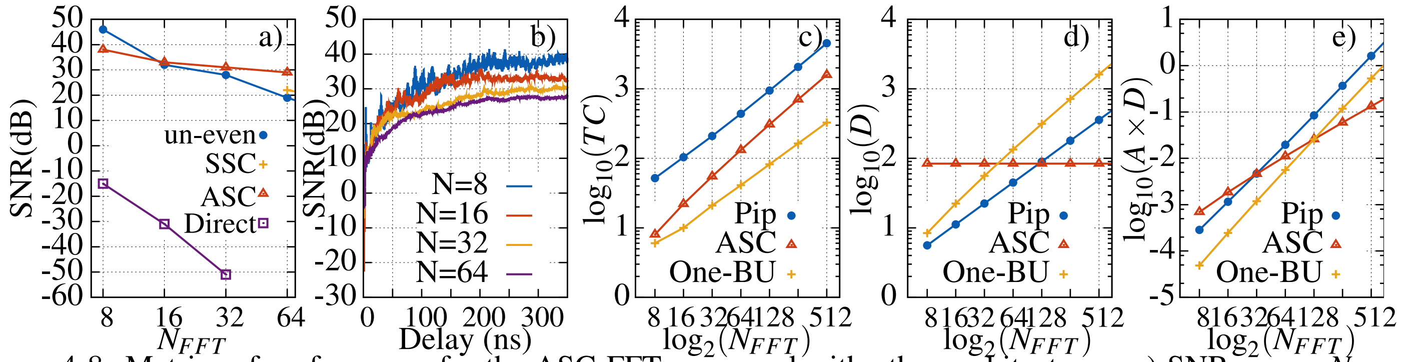

Figure 4-8: Metrics of performance for the ASC-FFT compared with other architectures. a) SNR versus N_{FFT} for different SSC-FFT architectures b) SNR versus Delay varying N_{FFT}s. Transistor count (c), delay (d) and delay-area product (e) for ASC-FFT and conventional FFT architectures versus N_{FFT}.

Table 4.2: Hardware Performance Comparison. *Valid for L=4 bits. **At maximum frequency of operation and N_{FFT}=64

FFT/Adder Implementation	**Adder transistor count**	**Multiplier transistor count**	**# Clock cycles**	**Latency for FFT (ns)****
Binary-One-BU [84]	444	3379	192	135
Binary-Pipelined [87]	444	3379	64	45
SSC D.Adder [39]*	146	-	-	-
SSC Digital$\Sigma\Delta$ [40]	240	10	1024	328
SSC Dual-Line [85]	156	10	1024	256
ASC (This Work)	108	10	-	84

Hardware Cost: The ASC adder has 76% and 24% less hardware cost than a conventional binary adder and other stream-adder approaches respectively (see transistor count (TC) in Table 4.2). The hardware cost for the conventional FFT architectures depends on the number of adders, multipliers and registers between stages. We compare the TC neglecting interconnection and routing logic for different FFT architectures in Fig. 4-8c. The ASC-FFT hardware cost is placed between the pipelined and the one-BU architecture. Even though in the ASC and pipelined FFT architectures, the number of adders and multipliers is growing with N_{FFT}, the registers in the pipelined architecture grows faster (2*N/3-2) [86] while in the ASC-FFT there is no memory involved in the computation. For the one-BU architecture, on the other hand, the number of computing elements is constant and only the registers grow with N_{FFT}.

Delay: For conventional and SSC FFT architectures, we estimate the delay as $cc \times T_{min}$ where cc is the number of clock cycles to finish the FFT computation, which grows with N_{FFT} and T_{min} is determined by the worst propagation delay in the logic data-path. To keep the comparison fair, for the ASC-FFT architecture we choose a computation time that results in the same SNR performance as the SSC-FFT architecture (latency column in table 4.2). Fig. 4-8d shows delay (D) versus N_{FFT}. The computation delay for the ASC-FFT architecture is better for $N_{FFT} \geq 32$ and $N_{FFT} > 128$ than the One-BU and the pipelined architecture respectively.

Fig. 4-8e shows that the AxD for the ASC-FFT is better for $N_{FFT} > 16$ and $N_{FFT} > 32$ than the pipelined and the one-BU architectures respectively.

4.2.3 Conclusion

Recent work demonstrated that using CTS reduces the latency, power and energy consumption for SC, opening the door for a new paradigm: ASC. However, this asynchronous approach has been limited to simple logic which constrains its application. In this work, we first present a low-error asynchronous adder for CTS that incorporates a capacitor embedded in a feedback loop as a memory element to keep carry “time". This adder, enables more complex functions such as the ASC-FFT presented here. Our adder requires 70%-30% less transistors when compared to conventional and SC adders respectively. Besides, the ASC-FFT shows 3X less latency than SSC-FFTs. Furthermore, there is a significant advantage in hardware cost when compared with pipelined architectures. Overall, simulation results show that ASC-FFT outperforms both pipelined and one-BU FFT architectures for $N_{FFT} > 32$ and $N_{FFT} \geq 128$ in terms of area-delay efficiency. Due to improved area and latency performance, ASC-FFT architecture can be embedded as a promising accelerator for energy and area constrained systems such as sensors and IoT devices.

4.3 Towards low power machine learning using asynchronous computing with streams

4.3.1 Summary

We combine near-analog-memory (NAM) computing and asynchronous-computing-with-streams (ACS) to obtain a NAM-ACS based architecture for the next generation of sensors for the internet of things (IoT), smartdust, or edge intelligence (EI). ACS has the potential to enable ultra-low power, massive computational resources required to execute on-node complex machine learning (ML) algorithms; While NAM addresses the memory-wall which increases the power and latency of ML and other complex functions. In ACS an analog value is mapped to an asynchronous stream that can take one of two values (v_h, v_l). This stream-based data representation enables area-power efficient computing units such as the multiplier implemented as an AND gate yielding savings in power of $\sim$90% compared with digital approaches. NAM-computing and ACS have one common issue, the cost of analog-to-digital and digital-to-streams converters sky-rocket the power, latency and energy cost making the approaches impractical. Our NAM-ACS architecture simplifies the sensor architecture and eliminates expensive conversions, enabling an end-to-end processing on asynchronous streams data-path. In this work, we tailor the NAM-ACS based architecture for random forest (RaF), a ML algorithm, chosen for its ability to classify using a reduced number of features. Our simulations show that the NAM-ACS approach enables 75% of savings in power compared with a single ADC obtaining a classification accuracy of 85% using a RaF inspired algorithm.

4.3.2 Introduction

The IoT, smartdust and EI have one common factor the deployment of trillions of ultra-low-power, interconnected wireless sensors. Moreover, due to the elevated cost and long latency incurred in raw data transmission to the cloud, the next generation of these sensors are expected to locally process data such as images and speech in real time using complex machine learning algorithms [56, 88, 89]. Unfortunately, current examples of these advanced sensors can drain a battery in only 40 minutes of use [57], given the restrictive form factor due to safety concerns and to enable wearability. To meet the expectations of long battery life and small form factors for smart sensors, three challenges need to be addressed: (A) the high power and area cost of parallel intensive computational resources; (B) the high power (power wall) and latency (memory wall) cost of data transfer from memory to processing units; and (C) the high power cost of analog-to-digital conversion (ADC) required as part of the sensor read-out circuits. In this work we propose an asynchronous computing with streams, ACS, based architecture that addresses all three challenges.

Computing with streams, instead of the typical approach based on digital numbers, addresses the need for ultra-low-power parallel computing units (challenge A). One of the most popular versions of this computing paradigm is synchronous stochastic computing (SSC[1]) [10] [90], where digital numbers are mapped to long streams of bits (Figure 4-9). Several SSC-based architectures for image processing algorithms have demonstrated power savings in the computing units of $\sim 90\%$ [1, 2, 91] compared with typical digital approaches. Despite the power efficiency of these computing units, the SSC paradigm incurs in higher power, area and energy cost due to the extra circuits required for the interaction between the computing units and the conventional parts of the sensor such as ADC, digital memory and output interfaces (Section 4.3.3).

ACS has the potential to leverage SSC advantages while addressing its drawbacks. In ACS an analog quantity is mapped to an asynchronous stream (Section 4.3.3). Although previous work demonstrated significant savings in power and energy compared with SSC and typical digital approaches, for applications such as multiplication, gamma correction

[1]Commonly known in literature as SC, we make emphasis in the synchronous nature of Gaines [10] approach.

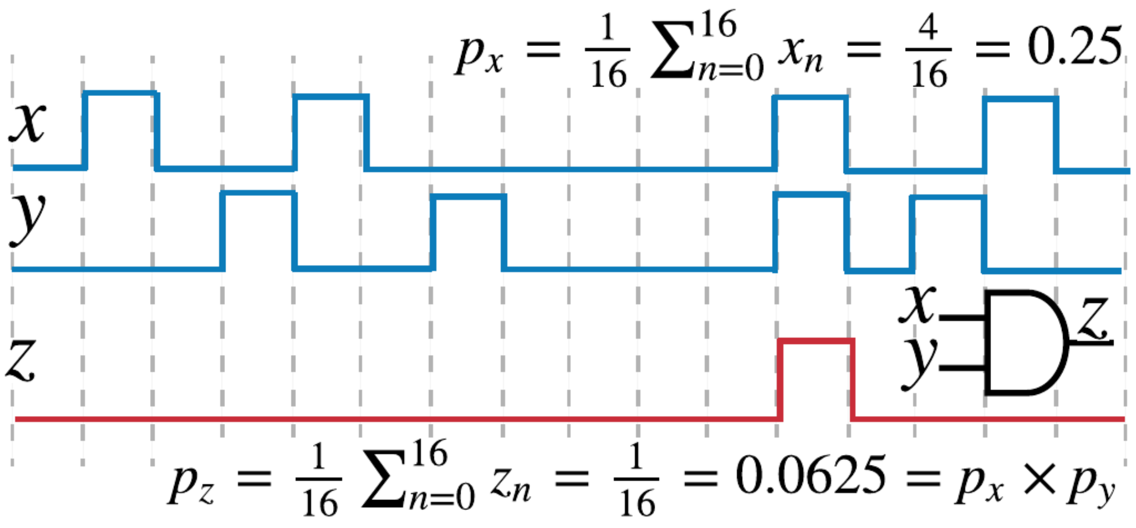
$p_x = \frac{1}{16}\sum_{n=0}^{16} x_n = \frac{4}{16} = 0.25$
x
y
z
x
y
z
$p_z = \frac{1}{16}\sum_{n=0}^{16} z_n = \frac{1}{16} = 0.0625 = p_x \times p_y$

Figure 4-9: Streams x and y represent number $p_x = p_y = 0.25$. An stochastic multiplier can be implemented using an AND gate. Stream z is the output of the AND gate and represents number $p_z = 0.0625 = p_x \times p_y$. Dashed lines represent the clock.

and fast Fourier transform [35] [11] [92] [93], the challenge to achieve seamless integration between the computing units and all the different building blocks in the architecture still remains. Similarly to SSC, unless this challenge is addressed, its power/energy/area cost will be prohibitively high.

In this work, we propose a sensor architecture based on ACS (Section 4.3.4), focusing on the interface between the computing units with the memory (challenge B) and the read-circuit (challenge C). We adopt a memory-centered architecture using analog-memory instead of SRAM or DRAM. Typically, to integrate analog memory with other digital components in the system, several ADCs are required which account for 58%, 31% [3] and 50% [7] of the total power, area and energy respectively (Section 4.3.3). Similarly, the sensor read interface requires several ADCs, which account for 33%-85% of the total power [94] for this interface (Section 4.3.3). In this work, we replace the ADCs with power/energy efficient asynchronous sigma delta modulators (AΣΔMs) which convert the analog inputs to streams of ones and zeros. The elimination of complex ADCs combined with the simplicity of the AΣΔMs avoids any expensive digital→streams→digital conversion achieving an end-to-end processing on asynchronous streams data-path.

We evaluate the performance of our architecture using RaF inspired algorithms, such as ADABoost and boosted-regression-trees (BTR) for image classification (Section 4.3.4). Other approaches for artificial vision sense, record, and process an entire image using complex classifiers trained with massive amounts of data. However, the concept of *active vision* suggests that biological vision systems do not store complete images of the world but focus on few visual features needed for a particular task [95]. RaF enables the use of at least 75%-97% less features compared with other machine learning algorithms such as neural networks [96] [97] enabling the application of active vision. Moreover, RaF is widely used today because of its high accuracy across a large array of applications, low complexity for implementation, low training time, high-throughput inference, and robustness against noise and over-fitting. Previous work using RaF inspired algorithms demonstrated 90%+ accuracy using only 30-200 [96] [97] features, out of a total of 784 features for an image in the MNIST database [98].

The framework to evaluate our ACS-based architecture uses ideal models for the analog memory and sensors. The interfaces and the ACS units are designed and simulated in Cadence-Virtuoso using a foundry provided FinFET1X[2] technology. Training for RaF algorithm is performed off-line to obtain the decision thresholds (Section 4.3.5). Our work shows that using a NAM-ACS architecture results in savings in latency, power and energy compared with an ADC based architecture. Assuming a memory read time of 100ns [3], our simulations yields savings of 49%-99% at the analog-memory interface depending on the memory-array size. Moreover the power efficiency of the memory interface combined with the parallel asynchronous computing on streams cores is 1.6X-3.6X better than the power efficiency of a single ADC. Notice that while the ADC is only doing conversion, our approach includes data processing as well. Finally, for a memory of size 64×128 and assuming 100 ns of memory-read-time [3], the accuracy obtained using BRT is 85% and the power of the AΣΔM interface combined with the computing unit enables 75% and 25% of savings in power and latency compared with the cost of single ADC.

4.3.3 Motivation and Background

Addressing challenge (A) with asynchronous computing with streams

Synchronous Stochastic Computing: One approach that is gaining popularity to address challenge (A), the high power and area cost of massive amounts of computational resources, is to encode data in streams of ones and zeros instead of the typical digital representation [35] [10] [72]. In SSC, the most popular of these approaches, a number is represented as the average of a stream of bits. This data representation enables extremely efficient computing units. Figure 4-9 shows a stochastic multiplier (AND gate) and a

[2]In modern technologies the node number does not refer to any one feature in the process, and foundries use slightly different conventions; we use 1x to denote the 14/16nm FinFET nodes offered by the foundry.

stochastic multiplication example for 16-bit streams. Notice that the stochastic multiplier requires 6 transistors while an 8-bit digital multiplier requires 3000. This contrast yields savings in power and area of at least 90%.

Although promising, the SSC approach is not widely adopted for three major reasons. First, expensive stochastic number generators (SNG) are required to form random streams. Figure 4-10b shows an example of a SSC image sensor and the circuit used for the image processing algorithm gamma correction. The gamma correction circuit yields savings of 93% and 96% in power and area compared with the typical digital approach [72]. However, to generate the random streams of bits, SNGs with linear feedback shift registers (LFSRs) are required. The SNGs account for 87% of the system cost [2] eclipsing the savings achieved by the efficient processing units. Second, long streams are required to obtain acceptable accuracy. For example, SSC implementations of finite impulse response filters [99] and fast Fourier transform [100] require 8000 and 1200 bits, respectively, to obtain acceptable accuracy, negatively impacting the latency of the system. Finally, the fast clock generation and distribution networks required to ameliorate the impact on the latency performance increase the design complexity and power consumption.

Asynchronous Computing with Streams: In ACS an analog value g is mapped to an asynchronous stream that can take two possible values v_h and v_l, and data is mapped to the time the stream is at v_h instead of the probability of a discrete event to occur [92] [93]. To generate asynchronous streams we use a current-based-asynchronous $\Sigma\Delta$ modulator (A$\Sigma\Delta$M) [35] which architecture is shown in Figure 4-11a. The A$\Sigma\Delta$M takes an input current I_{in} such that $I_{in}/I_{max} = p$ with I_{max} the maximum current in the system. Then, the modulator generates a continuous-time, asynchronous and periodic stream such that p is encoded in the average value of the stream [67]. Figure 4-12 shows the output stream of the A$\Sigma\Delta$M for a triangular waveform. The frequency is maximum for $p = 0$ and minimum for $p = \pm 1$, while the duty-cycle is 50% for $p = 0$ and 0%(100%) for $p = -1(1)$.

The output of the A$\Sigma\Delta$Ms can be directly connected to the computing units as shown in Figure 4-11b, achieving an end-to-end asynchronous computing on streams data-path. The combination of a clock-less design and the asynchronous nature of the modulator enables a practical implementation of ACS. Previous work demonstrated ACS multiplication, gamma correction, low error addition and fast Fourier transform [35] [11] with significant savings in area, power, latency and energy.

ACS shares most of the advantages of asynchronous systems such as high performance and low-power consumption. In a synchronous system, either SSC or digital computing, the maximum frequency of a global clock is determined by the single path with the worst propagation delay in-between registers. Faster paths, have to wait until the arrival of the

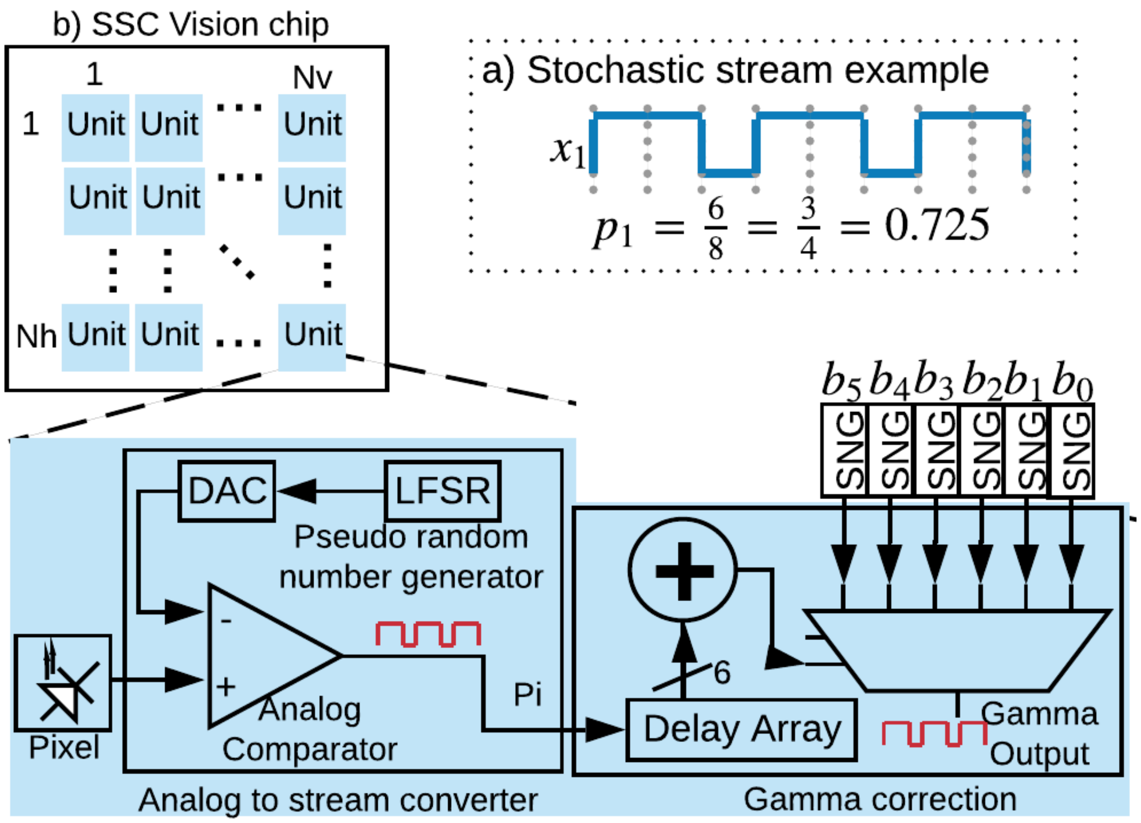
b) SSC Vision chip
1
Nv
1
Nh
Unit
a) Stochastic stream example
x_1
$p_1 = \frac{6}{8} = \frac{3}{4} = 0.725$
$b_5\,b_4\,b_3\,b_2\,b_1\,b_0$
SNG
DAC
LFSR
Pseudo random number generator
-
+
Analog Comparator
Pi
Pixel
Analog to stream converter
6
Delay Array
Gamma Output
Gamma correction

Figure 4-10: State of the art SSC vision chip composed of an array of Nh×Nv units. Each unit has a pixel, a SSC-readout circuit which in this case is the analog-to-stream-converter and SNG [1], and the SSC processing units for gamma correction [2].

positive edge of the clock for the next step. In contrast for asynchronous computing the next step starts immediately after the previous step finishes without waiting for a slower central clock. That is, the information propagates through the logic as fast as each individual path permits [101]. Moreover, in a synchronous system the clock is switching non-stop, even for inactive paths, consuming active power continuously. An asynchronous system only consume active power when completing a task returning to an almost not dissipating state afterwards [101]. An example of this, is the AΣΔM (Figure 4-11) which switches only when the input integrator crosses a threshold instead of at each clock cycle, reducing the activity factor (α) [30].

A caveat for asynchronous systems is the need of completion and hand-shake circuits that indicate when the computation is done. As we will discuss in section 4.3.4, for ACS

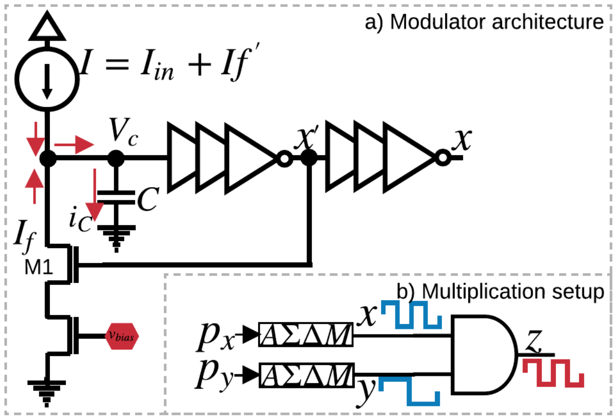

Figure 4-11: a) Current-based asynchronous ΣΔ modulator (AΣΔM) architecture. b) Multiplication setup using ΣΔ streams

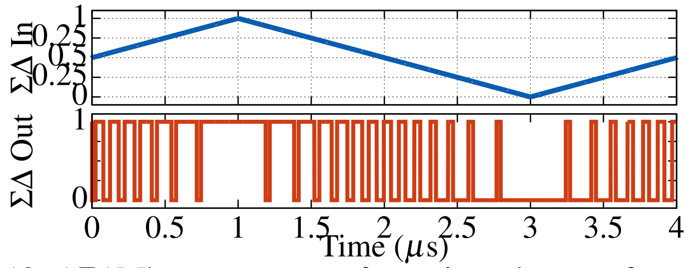

Figure 4-12: AΣΔM's output stream for a triangular waveform at the input.

there is no need for these supporting circuits.

Addressing challenge (B) with near analog memory computing

A Memory-centric architecture, is one of the options to address challenge B, reducing the data movement and power-latency overhead. One approach packs large amounts of memory close to the processing elements such as is done in Google's TPU [58]. However, data transfers are not fully eliminated, thus the memory/power wall is not addressed. A second approach, brings the computing units adjacent to the memory core, yet limitations in the memory technology constrain the processing capabilities that can be implemented [102]. A third approach uses the memory architecture for storage and computing (*in-situ*). Prior work demonstrated *in-situ* SSC on DRAM using clever optimizations for data representation to address the latency penalties inherent to SSC [103]. However, the cost of analog-binary-stream-binary conversion makes the approach prohibitively expensive for an energy constrained sensor.

One promising approach for processing near memory replaces the typical digital memory with non-volatile analog memory. This approach increases memory density, and facilitates node recovery for ultra-low power IoT sensors prone to blackout periods [7]. However, to be fully compatible with digital architectures, ADCs are included in the memory to obtain a digital output (Figure 4-13a). These ADCs become a bottleneck for the resoruce-constrained system, because they consume 58%, 31% [3] and 50% [7] of the total power, area and energy respectively. In this work, we assume the presence of an analog memory and address the cost of the memory peripheral circuits by replacing typical ADCs with AΣΔMs (Section 4.3.4).

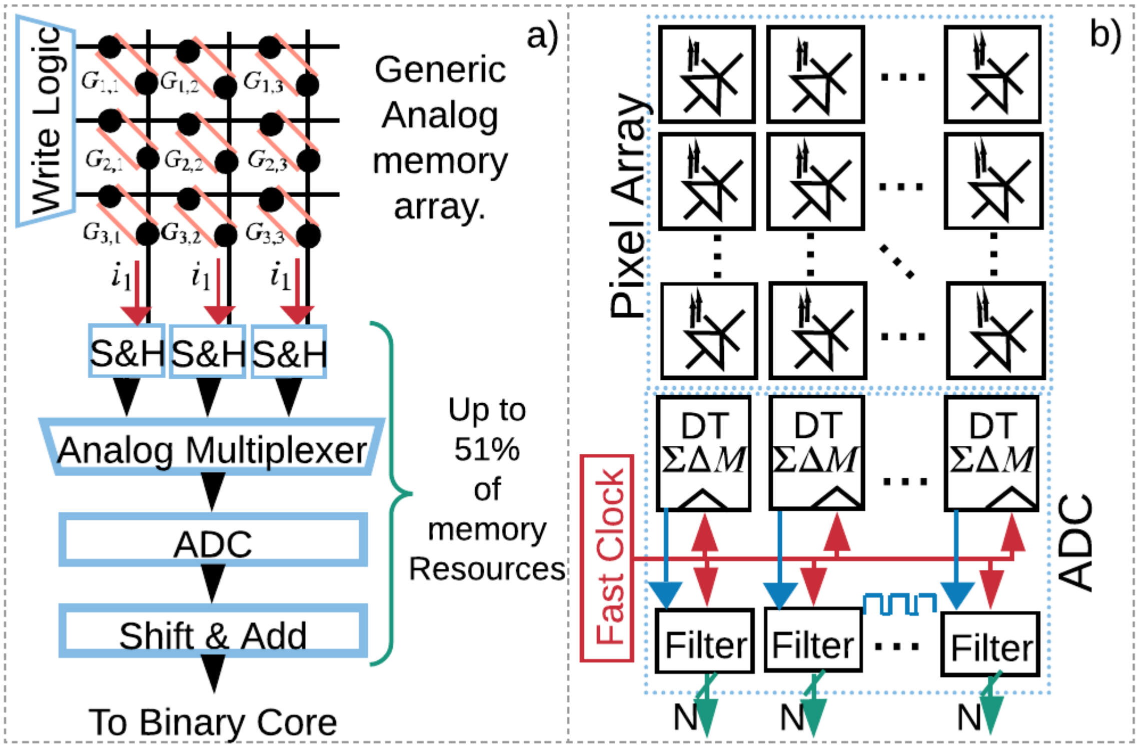
a)
Write Logic
$G_{1,1}$ $G_{1,2}$ $G_{1,3}$
$G_{2,1}$ $G_{2,2}$ $G_{2,3}$
$G_{3,1}$ $G_{3,2}$ $G_{3,3}$
i_1 i_1 i_1
Generic Analog memory array.
S&H S&H S&H
Analog Multiplexer
ADC
Shift & Add
To Binary Core
Up to 51% of memory Resources
b)
Pixel Array
DT ΣΔM
Fast Clock
ADC
Filter
N

Figure 4-13: a) Typical Analog memory architecture. Promising options for analog memory technologies are memristors [3] [4], floating-gates [5], Ferro electric field effect transistor (FeFET) [6] and Flash [7]. b) Simplified block diagram for an image sensor. The read-circuit includes a column parallel ADC. Previous work have demonstrated DT-ΣΔ ADCs as a feasible low-power option [8]

Addressing challenge (C)

Figure 4-13b shows the block diagram of a typical image sensor composed of 3 elements: (1) a pixel array that converts light intensity into analog signals. (2) Column-parallel ADCs convert the analog information to a digital format with N bits; and (3) the control logic (not shown) and auxiliary circuits that orchestrate the sensor operation. A major bottleneck for low power operation for the sensor, is the column-parallel ADCs which consume 33%-85% of the total power [94]. Several ADC architectures such as cyclic, successive approximation register and flash have been demonstrated. As an example, Figure 4-13b shows the sensor using discrete-time ΣΔ ADCs (DT-ΣΔ-ADC) [8]. Section 4.3.4 describes our solution.

4.3.4 Algorithm and Architecture

In this section we start by describing the Random Forest algorithm. Then we delve into the details of each of the components for the proposed NAM-ACS based architecture, shown in Figure 4-14.

The algorithm: Random Forest

The Random Forest [104] algorithm is a machine learning model composed of an ensemble of randomly trained, binary decision trees. Each tree is trained independently from a random subset of the training data made with replacement, called bagging [105], and is grown by recursively choosing a split feature from a random subset of the feature space, and splitting the training data-points along a threshold. This threshold comparison for the split feature is captured as a split node in the decision tree, whose left child corresponds to the next state if the threshold qualification is met, and the right to the contrary. Each leaf node in a tree represents a classification result [9]. Figure 4-15a shows an example for a decision tree used to classify an input sample into 3 different classes: Class0-Class3. Figure 4-15b shows an input sample, *a.k.a* feature vector, conformed of features p1-p4.

At run-time, each tree yields a classification result for the input feature vector. Starting at the root node, a root-to-leaf path is traversed based on the values of the features of the input sample. Figure 4-15a highlights the root-to-leaf path for the feature vector shown in Figure 4-15b. Since each of the split outcomes is mutually exclusive, there is only one root-to-leaf path per tree which can be traversed for any input feature vector [9].

To address the nonuniform memory access patterns of tree traversal algorithms and load balancing, we adopt a RaF-AP [9] implementation based on Automata Processing

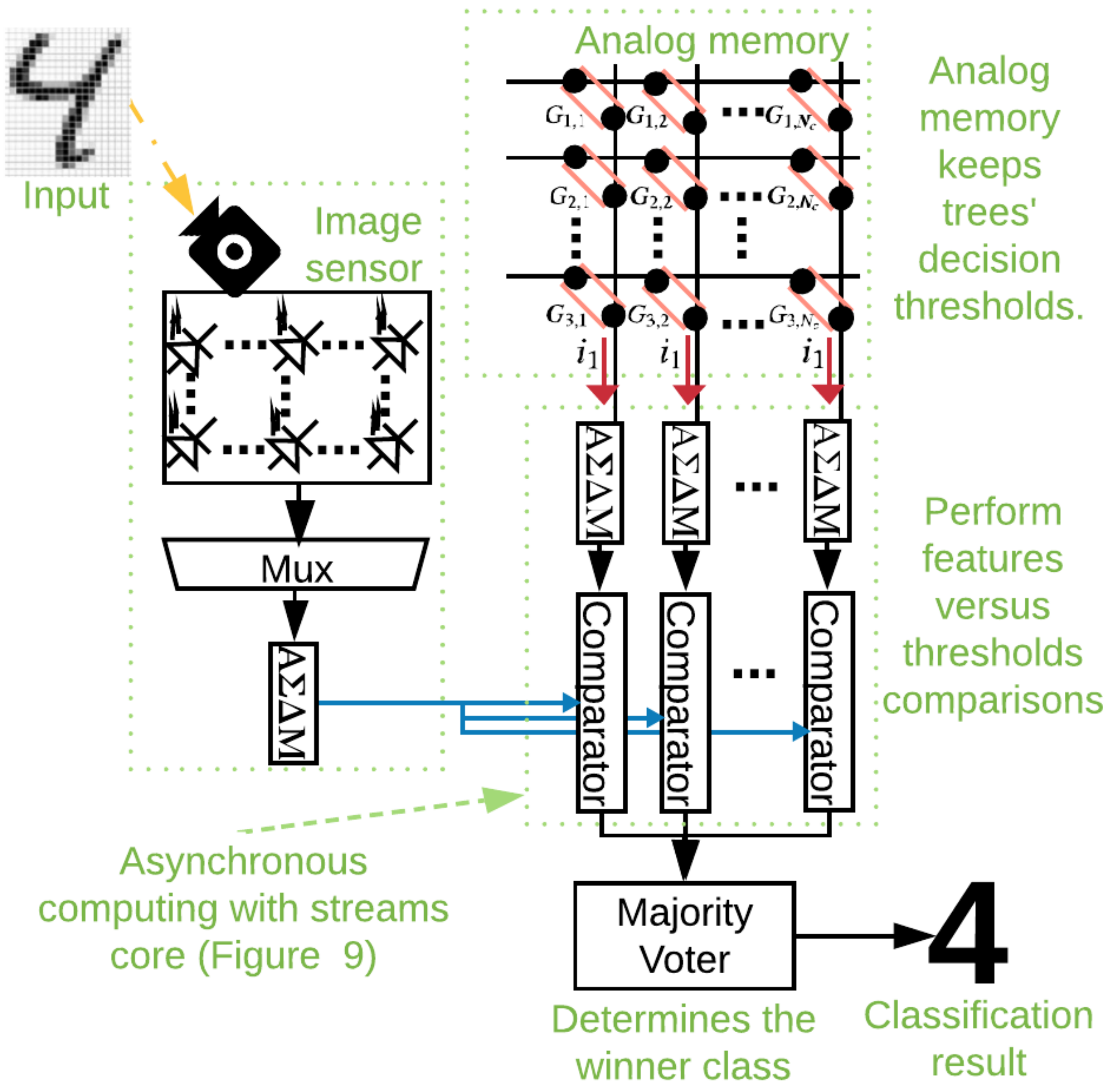

Figure 4-14: Image sensor architecture with on-node image classification capabilities using the machine learning algorithm Random Forest.

(AP). In RaF-AP each root-to-leaf path in a decision tree is represented as a chain of feature evaluation nodes (Figure 4-16). Each evaluation node represents one side of the decision tree's split node, and all possible paths are translated into ordered chains, enabling both parallel and streaming execution of all decision trees simultaneously. This architecture enables complete inference in a single stream of the feature vector without the need

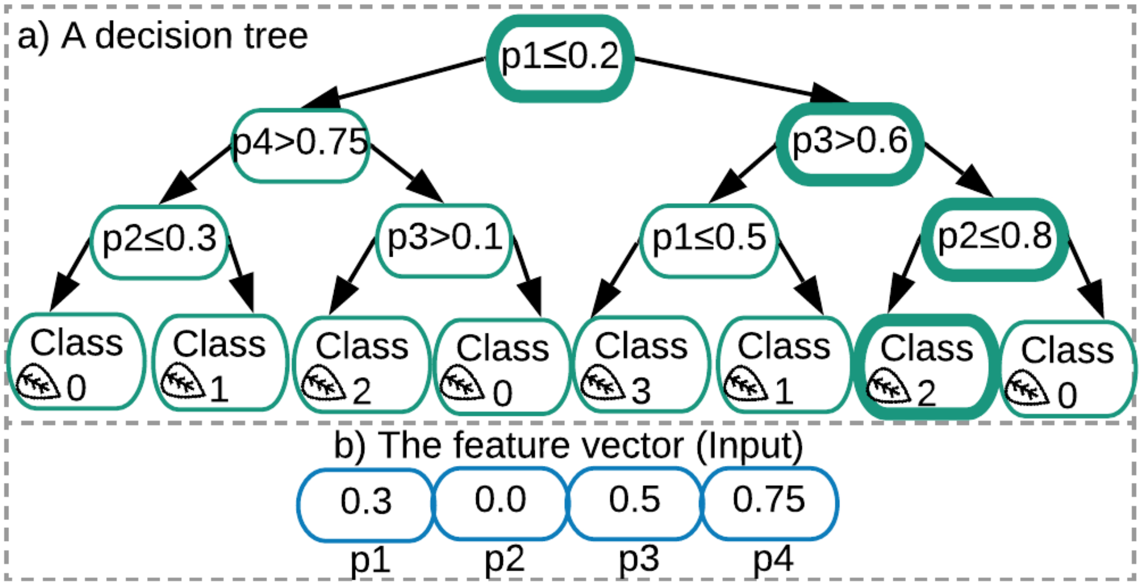

Figure 4-15: a) Single Tree example and b) the input feature vector.

of buffering or data movement across the architecture.

Two other decision tree-based algorithms: Adaboost, for Adaptive Boosting, and Boosted Regression Trees using boosting [106] instead of bagging. Unlike with Random Forests where the decision trees independently learn the same distribution, and the resulting independent classifications are combined with a voter, Adaboost and Boosted Regression Tree algorithms have decision trees that are much weaker learners that are summed together to create a stronger learner. Inference is made in the same way as with RaF with a tree traversal; the only difference is how the results are combined in the final stage of the algorithm.

Architecture description

The block diagram for the proposed architecture is shown in Figure 4-14. An analog memory is used to keep the thresholds obtained after offline training. Notice, that each column in the memory is mapped to a chain of features, which in turn represents a class (Figure 4-16). The analog output of each column is converted to a stream using column-parallel AΣΔMs. These streams are connected directly to the column-parallel computing units (comparators), which compare the input features with the thresholds coming from the memory one row at a time. After evaluating all the rows the output of the comparator is “1"

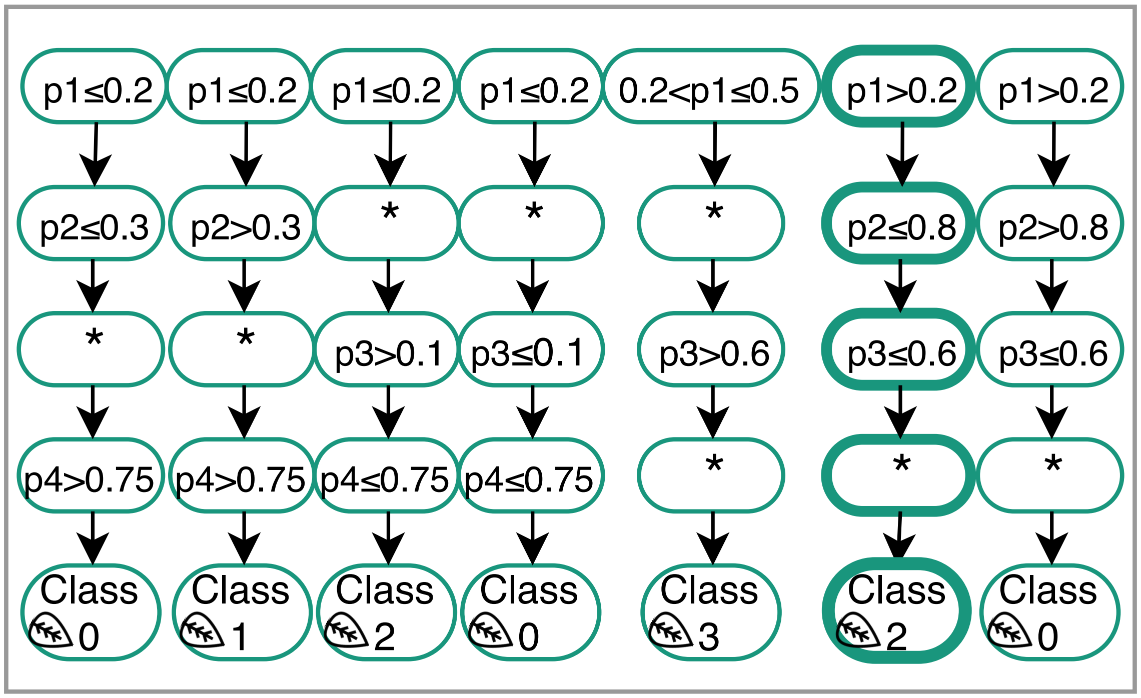

Figure 4-16: Implementation of RaF that enables parallel execution of decision trees [9]. Each node has a high and low threshold, see row 1, column 5. In this example, the thresholds are omitted when they are the minimum or maximum values.

when the feature set belongs to the class associated with that column, and “0" otherwise. A majority voter decides the class with more votes among the trees.

The efficiency of this architecture is due to two key factors. (1) A reduced set of features required for classification (Section 4.3.6). (2) The end to end asynchronous computing with streams approach enabled by the ACS paradigm combined with the use of the NAM-processing approach.

On the remaining of this section we describe the computing units for asynchronous streams and in section 4.3.6 we analyze the cost of the analog memory interface using AΣΔMs combined with the computing units.

Computing units for asynchronous streams

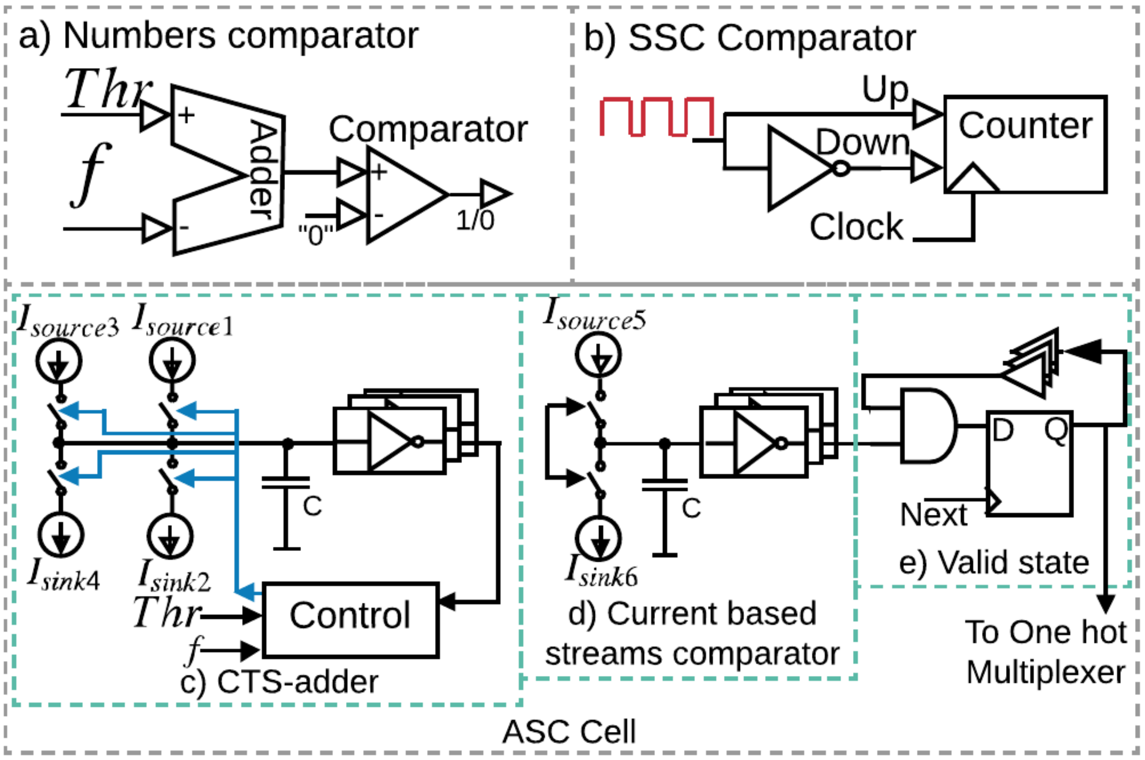

Figure 4-17: a) Conventional hardware implementation to compare two arbitrary numbers. b) SSC sign detector [10] c) Adder for asynchronous streams [11]. d) Current comparator. e) D-flipflop to keep the column status. The combination of the CTS-adder and the current comparator form the continuous time streams comparator.

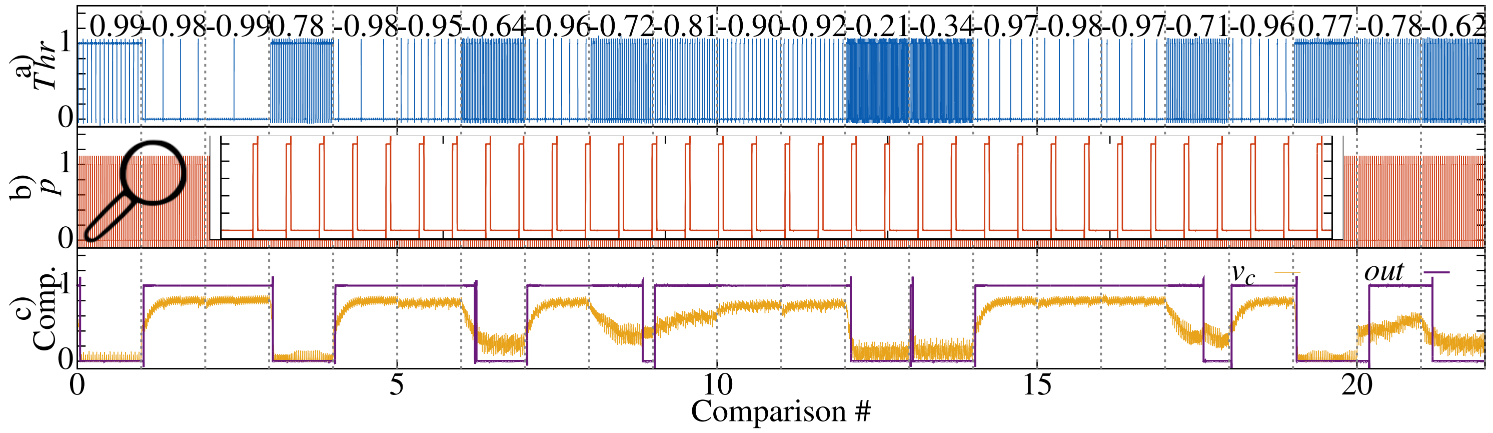

Figure 4-18: Comparator operation example keeping the input feature ($p = -0.73$) constant and varying the threshold (Thr) for 22 different comparisons. If $p \geq Thr$ the comparator output (out) is "1" a), b) Threshold (Thr) and feature (p) to be compared. c) Comparator signals: Voltage in the capacitor (v_c) and final comparator output.

Processing the data coming from the pixel array requires massive hardware and software capabilities specially when using ML algorithms. To demonstrate the potential of ACS for ML we use a parallel array of computing units. Each unit compares the stream coming from the pixel array, that is the feature p, with the thresholds Ths from the RaF algorithm coming from the analog memory, that is:

$$z = \begin{Bmatrix} 1 & \text{if } p > Th \\ 0 & \text{if otherwise} \end{Bmatrix} \tag{4.4}$$

where z is the computing unit output. A comparison between two numbers is typically implemented with an adder to obtain $(Th - p)$, followed by a comparator with zero as shown in Figure 4-17a. Figure 4-17c shows the adder for asynchronous streams, composed of a capacitor embedded in a feedback loop, acting as an asynchronous memory element. The function of the capacitor is to keep track of the addition carry significantly reducing the error associated with computing with streams [11].

A sign detector can be implemented with an integrator as shown in Figure 4-17b [10]. Typically in SSC, an integrator is implemented as a clocked-counter that increases when the input stream equals 1 or decreases when the input stream is 0. The quantity in the counter represents the integral of the input. Eventually, the counter will become full or empty depending on the sign of the input quantity [10]. A digital counter requires a clock which is not compatible with the continuous time nature of the asynchronous streams. To implement the asynchronous comparator, we use a current based integrator instead (Figure 4-17d). The voltage in the capacitor acts as continuous-time state-keeper that increases(decreases) when the input stream is 1(0). The inverters convert Vc to 0 or 1. We design the computing units in Cadence-Virtuoso using a FinFET1x technology and perform spectre level simulations to obtain the metrics of interests. Figure 4-18 shows the output of the unit comparing a fixed feature p with 22 different thresholds Thr. The output of the comparator is "1" when $p > Thr$.

Analog memory for thresholds and pixel array considerations

Analog memory: Advancements in non volatile memory (NVM) technologies such as resistive-RAM (RRAM) [4] [3], Ferro-electric-Fet (FeFET) [5] and Floating-gate (FG) [7], enable the fabrication of analog memories with 2-8 bits of resolution. Table 4.3 summarizes the recent analog memory implementations showing improvements in all metrics of performance. All numbers are taken directly from the published literature. The summary shows that leakage power is reduced by 98% for FeFET, PCM and RRAM compared with

SRAM. Power per cell is reduced by 99.9% and 54% for FG and RRAM respectively compared with eDRAM or SRAM. Analog memories also have comparable retention time compared to conventional DRAM and SRAM memories. Although the current reading and writing speeds are slower than DRAM and SRAM, they are still fast compared to flash memories [107], and the significant power and density advantages made them great candidates for budget-constrained applications. Unfortunately, the ADC-based interface, used for digital compatibility, is expensive and a major drawback for the adoption of this technology. In this work we propose to use column-parallel AΣΔM as the output interface for the analog memory, in this way we overcome the ADC bottleneck, and the analog memory output is converted to asynchronous streams for the ACS units (Figure 4-14). Section 4.3.6 will discuss the memory interface cost in detail.

***The image sensor*:** Similar to the analog memory, we can simplify the read-out sensor interface. Notice in Figure 4-13b that the output of a DT-ΣΔ modulator for the DT-ΣΔ-ADC is a stream of bits, thus to use a stream based approach, we can remove the digital filter and operate directly on the streams. Furthermore, we can replace the DT modulator with an AΣΔM. Figure 4-14 shows a generic pixel array with the proposed interface. Given that for this RaF inspired architecture, we only evaluate one pixel at a time, a single AΣΔM multiplexed among the pixels of interest suffices. Table 4.4 compares the power consumption of a single AΣΔM with an ADC, both designed in the same technology. The power consumption for the asynchronous modulators is at least one order of magnitude less than a SAR-ADC designed in the same technology.

4.3.5 Experimental Setup

Machine learning

We use the Scikit-Learn [113] Python machine learning framework to train our decision tree models. Random Forest, Boosted Regression Tree, and ADABoost Regression models are each parameterizable by the number of decision trees in their ensemble as well as the maximum number of leaves per tree.

One parameter that is uniquely important for this RaF implementation is the number of features considered by the models. This parameter determines the depth of each of our decision chains as well as the run-time for inference by setting the stream length. To set the maximum number of features considered by each model, we train a model with all features, and then select the top N features based on feature importance values. We then transform our training data and retrain the models with this reduced set of features. Figure 4-19 shows a heatmap of the feature importance obtained from the RaF model when

Table 4.3: Metrics for Memories.

	Type	Feature size ****	Array size	Bits per cell	# Bytes	Area	Power		Power per cell	Read time	Write time	Retention time
							Active	Leakage				
1	FeFET* [6] [107]	180nm	-	5	-	1071 μm^2	-	35.29 μW	-	45ns	65ns	10 years
2	PCM [108] [107]	45nm	256 $\times$ 256	8	32KB	0.043 mm^2	16.1 mW	35.29 $\mu W*$	245 nW	12ns	100ns	>10 years
3	RRAM [3] [107]	9nm	128 $\times$ 128	2	4KB	25 μm^2	300 nW	35.29 μW*	9.1 nW	50ns	0.3ns	>10 years
4	FG [109] [5] [110]	20nm	4 $\times$ 8	8	32B	490 μm^2**	1.5 nW ***	-	5.8 pW	-	-	-
4	eDRAM [3] [107]	36-65nm	-	1	64KB	0.083 mm^2	10.35 mW	-	20 nW	2-10ns	2-10ns	>10 years
6	SRAM [6] [107]	45nm	-	1	-	10311 μm^2*	0.9mW	1.1 mW*	19.09nW [111]	0.2ns	0.2ns	-

* Results obtained for MNIST classification using multilayer perceptron [6].
** Metrics for single memory cell [109].
*** Memory power consumption for a 16×16 image classification [5].
**** Feature size is compared against 45nm CMOS technology in this table. Although changes are expected due to advances in both CMOS and memory technologies, the relative relationship will stay similarly.

trained with all 784 features. The lighter the pixel color the most important the feature is for accurate classification. The closest the pixel is to black the less information it conveys. Using the top most relevant features for classification enables a significant reduction in the hardware size and consequently the system power consumption and latency. During run-time, we choose the top N features, which in turn determines the length of the chains and number of rows in the memory.

Once the model has been trained, we provide the Scikit-Learn model to the RFAu-

Table 4.4: Metrics for individual components of the system

		Power Supply (V)		
		0.6	0.7	0.8
Memory Interface				
ADC [112]	Sample rate (GS/s)		0.95	1.2
	Power (mW)		2.26	3.1
	Resolution		10	10
AΣΔM	Power (μW)	0.984	9.28	17.28
	Max. Frequency (MHz)	337		
ACS Computing Unit Power (μW)				
Adder		1.4	15	29
Comparator		0.96	5.2	12.6
Valid		0.01	0.13	0.13
Total		4.81	41.2	83

tomata [114] tool. This automata transformation tool based on Tracy, Fu, et al's [9] work, converts decision tree ensembles into an automata representation that consist in a range per node (Figure 4-16). In this work, Each node (range) is mapped to two cells in a memory row to keep the low and high thresholds.

Circuit simulation

To evaluate our architecture we use ideal current sources to model the memory and pixel arrays. We design the computing units and the AΣΔMs using foundry provided models for a FinFET1X technology. This work focuses on the trade-offs when using a massive amount of parallel computing units and modulators. For functional verification we use the thresholds obtained from off-line training for MNIST data-base classification to run circuit level simulations using Cadence-Spectre.

Finally, independently of the architecture of choice, given a digital data representation data-path, on-node real-time image classification requires several ADCs. Table 4.4 shows the metrics of performance for an ADC designed in a FinFET 14nm technology [112], which we use as a baseline for comparison.

4.3.6 Results

This section describes the metrics of performance for the NAM-ACS architecture. There is a power-latency trade-off that depends on the memory-array size and the power supply

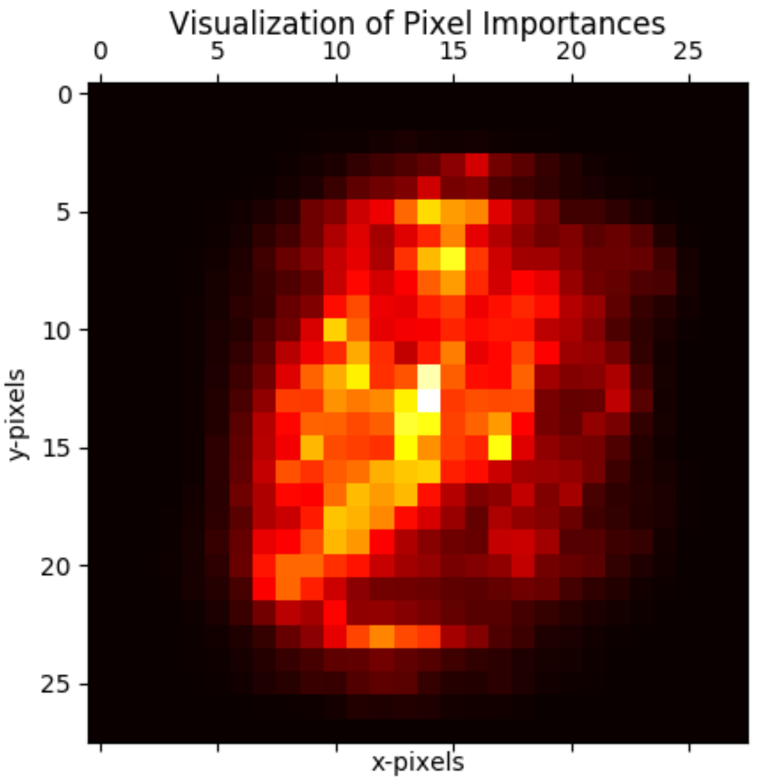

Figure 4-19: A visualization of the importance of pixels for classifying the image. Black is assigned to unimportant features that can be discarded, while white represents the most important features.

Vdd (Note that the metrics of performance reported for the ACS architecture include the cost of processing the data). However, even when the power consumption of the ACS architecture (AΣΔMs + ACS-cores) is maximum, its energy performance is comparable with the energy consumed by a single ADC.

To evaluate the ACS-architecture performance, we calculate the power, latency, and power-delay-product (PDP) from circuit level simulations for two different cases where the slowest component in the system dominates the minimum latency:

- *Case I:* The latency is determined by the memory-read time. Using a memristor analog memory we assume a read-time of 100 ns [3] [4].
- *Case II:* The system latency is defined by the ACS latency, which depends on the maximum frequency of operation of the ACS-core.

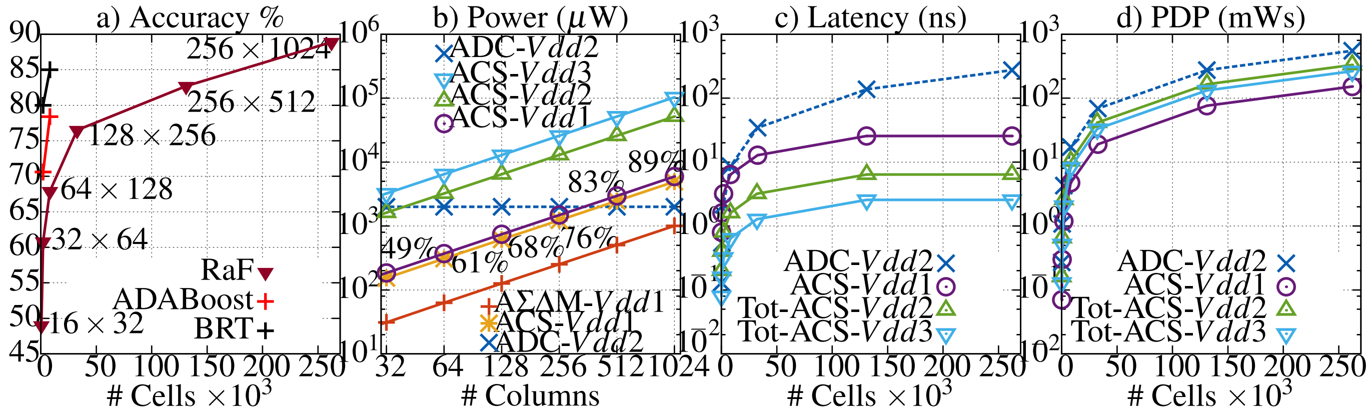

Figure 4-20: Metrics of performance for the ACS architecture. a) Accuracy versus # of cells. b) Power consumption versus number of columns for the memory interface (ASDMs) and ACS-cores for Vdd_1, f_1. Also the combined power consumption of the ACS-cores and the ASDMs for three different operating points. $Vdd1 < Vdd2 < Vdd3$ c) Latency versus # of cells for three different operating points. d) Product delay product for three different operating points.

Accuracy

Figure 4-20a shows the accuracy results for our trained models as a function of the number of cells (# Cells) in the memory array. The # Cells is obtained as the array's number of rows multiplied by the number of columns. Table 4.5 shows the results of a parameter exploration of the number of trees and number of leaves per tree and the classification accuracy for MNIST using RaF. The maximum number of features is set to 256 because we determined there not to be an increment in accuracy beyond this point.

We found that to maximize our accuracy, we had to trade off the number of leaves per tree, with the number of trees in the ensemble. Adding more leaves to the tree resulted in a model that better fit its random subset of the training data, while adding more trees reduces the effect of variance incurred by overfitting the subset.

Figure 4-20a also shows the accuracy results for ADABoost and BRT regression models. The memory size required is 4X-8X smaller compared with RaF for the same level of accuracy at the expense of an increment in the majority voter complexity.

Power consumption

Figure 4-20b compares the power consumption for the memory interface (AΣΔMs) and the ACS-cores with the power consumption of a single ADC. A reasonable assumption in an analog-memory-based architecture, given the large power consumption of typical ADCs, is the use of a single ADC per array (Figure 4-13a) [3], thus the average power for an ADC based interface (dashed blue line) is constant and independent of the array size.

For the ACS architecture we can vary the voltage and frequency (Vdd, f) to meet a latency requirement. When the latency of the system is determined by the analog-memory read-time (*Case I*), we find (Vdd_1, f_1) such that the computation is completed in 100 ns. Figure 4-20b shows a monotonic increment in the power consumption for the ACS-cores and the memory interface versus the number of columns (# Columns) in the memory array. Compared with a single ADC, the total power consumption of the NAM-ACS architecture is at least 1 order of magnitude smaller for an array with 32 columns, and comparable for an array of 512 columns. That is, ACS enables 512 parallel comparisons for the cost of only one of the ADCs required in previous architectures. This demonstrates the potential for ACS to enable massive computational resources for low-power ML algorithms.

When the latency of the system is determined by the ACS-cores (*Case II*), the maximum latency is achieved at the technology nominal power supply (Vdd_3, f_3). Figure 4-20b shows the combined power consumption of the ACS-cores and memory interface (ACS-$Vdd2$), which quickly grows faster than the consumption of a single ADC. As we discuss

Table 4.5: Accuracy obtained for different number of features, trees and leaves per tree, with the corresponding memory array size required for RaF

Memory array size (# rows $\times$ # columns)	Features (# rows)	Trees	Leaves per Tree	Accuracy
16×32	16	1	16	49%
	16	2	8	47.60%
	16	4	4	42.40%
32×64	32	1	32	60.80%
	32	2	16	60.40%
	32	4	8	53.40%
	32	8	4	50.90%
64×128	64	1	64	67.20%
	64	2	32	67.90%
	64	4	16	67.87%
	64	8	8	60.97%
	64	16	4	54.40%
128×256	128	1	128	74.90%
	128	2	64	74.60%
	128	4	32	76.50%
	128	8	16	72.70%
	128	16	8	66.90%
	128	32	4	56.50%
256×512	256	1	256	78.30%
	256	2	128	79.70%
	256	4	64	82.70%
	256	8	32	81.30%
	256	16	16	76.50%
	256	32	8	70.40%
	256	64	4	59%
256×1024	256	8	128	88.90%
	256	4	256	88.30%

below, this comes with a significant reduction in the system latency.

Latency

Figure 4-20c shows the latency versus number of cells. For the ACS architecture the latency is proportional to the number of rows (features p) multiplied by the computation time. For comparison, we also show the memory read-time for an ADC based analog-memory architecture (Figure 4-13a) where the latency is proportional to the number of cells multiplied by the ADC conversion time.

When the system latency is limited by the analog-memory read-time, the ACS-cores operate at $(Vdd1, f1)$. We find that for a memory array of 8X16 the latency for the ACS architecture is 6X times longer than for the ADC based approach. For a memory array with 8K cells (64X128), the latency for both approaches is comparable, and beyond this point ACS yields better latency than the ADC based approach. Since RaF for MNIST classification does not yield any significant accuracy improvement when increasing the number of features beyond 256 (section 4.3.6), the maximum number of rows in the memory array is 256 and the latency is constant beyond this point.

When the system latency is limited by the ACS-cores (Vdd_3, f_3), the latency is 1.6X-207X better than the ADC approach (Figure 4-20b). The improvements in system latency have a significant impact on the overall energy consumption of the system as we discuss below.

Power-Delay-Product

A good estimate of the power efficiency of a system is given by the power-delay-product (PDP) shown in Figure 4-20d. In an ACS system limited by the memory read-time, the PDP for the combined ACS memory interface and ACS-cores is 3X smaller than the power-efficiency of a single ADC. In a system limited by the ACS-cores latency, eventhough there is a significant increment in the power consumption, the PDP product for the memory interface combined with the ACS-cores is still 2X better than the PDP of a single ADC.

Table 4.6 summarizes the metrics of performance for RaF, ADABoost and BRT for the best accuracy obtained. Notice that ADABoost and BRT enable a reduction of 96% in the memory size for a similar accuracy compared with RaF. This reduction enable savings of up to 87% in power and latency at the expense of a small increment in the majority voter complexity.

Table 4.6: Best accuracy for RaF, ADABoost and BRT, with the corresponding power and latency estimates. The cost of the majority voter, pixel array is not included since it is the same cost for conventional architectures.

		RaF	ADABoost	BRT
Accuracy		88.9%	78.4%	85%
Memory Size		256×1024	64×128	64×128
Power (mW)	Memory [5]	15×10^{-3}	47×10^{-6}	47×10^{-6}
	Memory interface	1	0.125	0.125
	Sensor interface	0.001	0.001	0.001
	ACS-core	4.9	0.616	0.616
	Total	5.9	0.742	0.742
Latency (us)		275	17	17
PDP (nWs)		152	2.38	2.38

4.3.7 Conclusions

The next generation of sensors for the IoT, smartdust or EI not only will operate under constrained power budgets, but will also require sophisticated processing capabilities for machine learning algorithms. In this paper we propose a NAM-ACS architecture for RaF-inspired machine learning algorithms. This architecture enables an end-to-end asynchronous computing on streams data-path leveraging the advantages of other computing on streams paradigms such as SSC while addressing its drawbacks. To enable the end-to-end asynchronous data-path we introduce an analog-memory-centric architecture. Typical architectures using analog memories require ADCs to interface with the other blocks in the system, making these ADCs power and latency bottlenecks. In our work all the ADCs are eliminated and replaced with parallel AΣΔM. The cost of the parallel components increases with the memory size. However, for a memory of size 64×128 and assuming 100 ns of memory-read-time [3], the accuracy obtained using BRT is of 85% and the power of the AΣΔM interface combined with the computing unit enables 75% and 25% of savings in power and latency compared with the cost of single ADC. This demonstrate the potential for ACS to enable ultra-low-power machine learning for the energy constrained sensors that will conform the IoT, smartdust or EI.

Chapter 5

Conclusions and Future Directions

Wireless sensors for the IoT, smartdust or EI must operate under power and energy constrained conditions, yet the next generation of these sensors are expected to perform power-hungry and complex computational tasks. To conciliate these two contradictory needs, power-hungry computational tasks and ultra-low-power consumption, we propose a computing paradigm where data is mapped to asynchronous streams: ACS.

We first introduce asynchronous stochastic computing (ASC) and develop its theoretical foundations. We evaluate the performance of three applications. (1) A multiplier implemented with a single AND gate. (2) Gamma correction: and image processing algorithm and (3) a single layer for an artificial neural network. We find significant savings in latency, power and energy compared with SSC.

Given the encouraging results, we embark on the search for an asynchronous stream generator. In this dissertation we propose asynchronous computing on pulse density modulated streams generated with a low power AΣΔM as an analog to streams converter. We use circuit simulations and develop a theoretical model to understand the error boundaries for this approach. Our theoretical model demonstrates that there is an asymptotic error or bias for SC-AΣΔM. Our model and simulations results show that multiplication with SC-AΣΔM yields at least 70% of latency savings obtaining the same SSC accuracy performance, or 70% better accuracy when the computing time is similar to SSC.

To implement complex operations, that go beyond single stage combinational logic, we propose a capacitor embedded in a feedback loop, as a memory element. The memory element enables the implementation of an asynchronous low-error adder that we use later to implement the ASC-FFT enabling savings in area and latency compared with typical digital implementations.

The next generation of ultra-low-power sensors demand complex processing for machine learning algorithms. To address this need we propose a NAM-ACS architecture for RaF-inspired machine learning algorithms. The architecture features an analog-memory-centric architecture. Typical architectures using analog memories require ADCs to interface with the other blocks in the system, making these ADCs power and latency bottlenecks. In our work all the ADCs are eliminated and replaced with parallel AΣΔM.

The ideas described in this dissertation originates a novel sensor architecture that features end-to-end asynchronous processing on streams. This architecture eliminates the need for a clock generation unit and costly clock distribution networks. Moreover, the sensor is simplified by eliminating expensive stream-digital-stream conversions.

The significant savings in latency, area, power and energy demonstrate the potential of asynchronous computing on streams to reduce the power consumption and enable ultra-low-power operation for the wireless sensors that will conform the smartdust, IoT and EI.

5.1 Future Directions

5.1.1 Asynchronous ΣΔ Modulator metrics of performance

In this dissertation we use a simple version of an asynchronous ΣΔ modulator architecture. We designed the architecture in cadence virtuoso using a FinFET1X technology and run simulations using foundry models. We also developed a continuous time Matlab model used for algorithm evaluation. These efforts, were targeted towards obtaining metrics of performance such as power, energy and delay, setting aside non-ideal effects given the AΣΔMs low sensitivity to circuit imperfections.

A necessary next step is a study of the effect of process, voltage and temperature variation using post-layout circuit simulations, Matlab models and post-silicon verification. Ideally, as there is no quantiser in the system, the AΣΔM do not suffer of quantisation errors. Hence, in theory, the signal-to-noise ratio (SNR) can be very high, even for a first-order system [30]. In practice the signal-to-noise-distortion ratio (SNDR) depends on the circuit deviations from the ideal behavior. Thus, an analysis of the effects of these non-idealities is required to define and measure a metric analogous to the effective number of bits in traditional ADCs. Examples of these non-idealities are:

- Transistor miss-match in the current mirrors used to charge and discharge the passive integrator (capacitor).

- Output current dependency of the load. A cost versus performance evaluation for different current mirror architectures would give a better understanding of the most suitable architecture depending on the application requirements.
- The effect of thermal and flicker noise as a fundamental limit for the AΣΔM performance.
- Propagation delay in the comparator.
- Variation in the propagation delay.

Also, a good portion of the trade-offs and circuit knobs are tied to the particular application requirements. For example, the final implementation of the AΣΔM as the output interface will be informed by the analog memory characteristics. To the best of our knowledge some work has been done from the signals and system perspective to understand the behavior of the modulator, yet a complete analysis of the circuit design trade-offs could result in a new PhD dissertation.

5.1.2 Design automation tools

One of the main advantages of digital versus analog design is the level of automation in the design and verification flows, which enables the development of complex systems (billion of transistors) with high probability of success.

In order to validate our ideas we looked into some applications that included multistage circuits (FFT) or high parallelism (RaF) using custom made scripts that generate circuits for spectre level simulations. However, as the complexity of the system increases and the necessity to go through the complete design flow for chip fabrication arises, some form of design automation is required. A custom design flow, similar to analog circuits, would make the approach unfeasible as we increase the level of complexity.

To address this, we envision mixed-signal elements such as the low power adder, the asynchronous stochastic tanh and the asynchronous comparator, being implemented following standard cells dimensions and being included as part of the libraries used for synthesis and PnR. The success of such endeavor depends on the ability to minimize the effect of circuits non-idealities and the accuracy constrain determined by each particular application.

5.1.3 Wireless interface

An ACS based sensor should be capable of energy efficient wireless communication. One of the options to achieve ultra-low power wireless transmission is Ultra-Wide-Band Impulse-Radio (IR-UWB). An IR-UWB system is characterized by the low complexity of the transmitter and receiver units, achieving high- energy-efficiency. In IR-UWB, each symbol (in the simplest case a bit) is encoded in a train of extremely short pulses which results in a bandwidth wider that 500MHz [115]. Since the IR-UWB scheme is impulse based, the transmitter and receiver units are simplified due to the elimination of up-down converters. Typical IR-UWB implementations are optimized for synchronous streams of bits which requires accurate clock sources. Moreover, successful transmission requires the use of expensive synchronization schemes between the transmitter and receiver unit. In an ACS architecture data is mapped to asynchronous streams, thus there is not a definition of bit, the presence of clock or the notion of clock synchronization. Thus, an ACS based architecture requires an asynchronous, non-coherent wireless communication scheme. An option to address this requirement is shown in Figure 5-1, and Figure 5-2 show the waveforms at different stages of the architecture.

The transmission unit receives data from the ACS-core (Figure 5-7) or directly from an AΣΔM (Figure 5-1). Figures5-2a,b show the analog input and the stream output for the AΣΔM. A delay-cell (Figure 5-1a) is used to generate a short-pulse at each rising (“R") and falling (“F") edge of the asynchronous stream. The short-pulse activates a current-starved VCO to start an oscillation (Figure5-2c). This oscillation shaped by the effect of the antenna, approximates the Gaussian-like train of pulses typically used for UWB communication . To differentiate between the edge polarity, “R" and “F" are transmitted with center frequencies $f1$ and $f2$ respectively (Figure5-2c-zoom-in).

The receiver is shown in Figure 5-1b. A low-noise-amplifier (LNA) amplifies the signal captured by the antenna (Figure5-2d). Two band-pass filters centered around frequencies $f1, f2$ respectively, differentiates between the “R" and “F" information. Non-coherent energy detectors with level cross sampling are used to detect the incoming UWB-pulses and generate a single pulse (Figure5-2e) feasible for the energy detector. The outputs from the energy detectors are sent to a Set-Reset (SR) flip-flop to reconstruct the asynchronous ΣΔ stream (Figure5-2f).

Figure 5-3 shows the circuit implementation for the short-pulse generator built as a series of 3 current-starved inverters. The generator’s pulse-width is programmable in two ways. (1) The voltages $V_b, \bar{V}_b$ control the inverters current through transistors M1-M3, P1-P3, and (2) the variable-capacitors modify the inverter load [12]. The output of the short-pulse generators, activate a 3-stages current-starved VCO. A header-footer com-

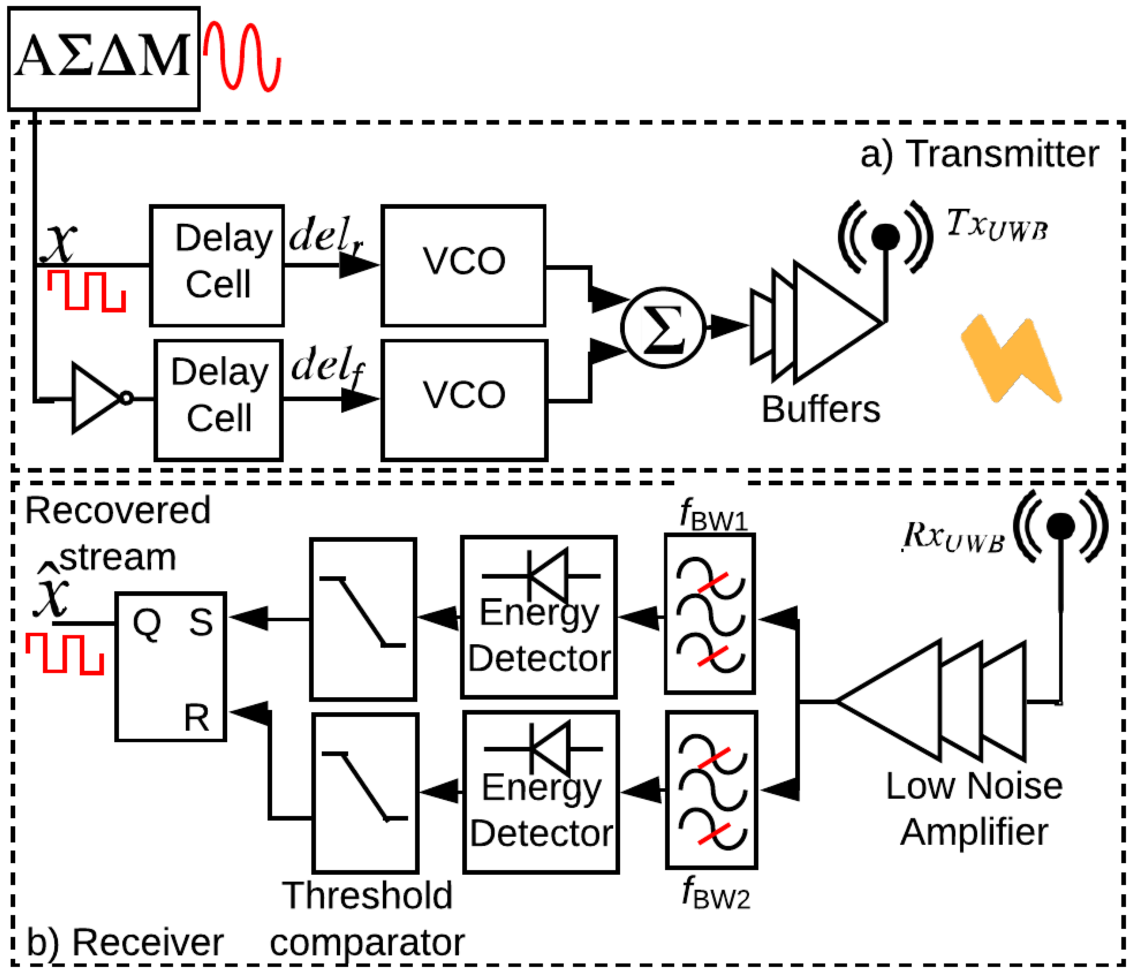

Figure 5-1: A possible implementation for a transceiver (transmission-reception units) for ACS streams. Adapted from [12].

bination, M4-M6, P4,P6, power-gates the VCO lowering the power consumption to the minimum when no pulse is transmitted, while bias-voltages $V_{f1}, \bar{V}_{f1}$ and $V_{f2}, \bar{V}_{f2}$ control the frequency of oscillation of the VCOs. Each VCO drives a high-impedance (High-Z) inverter which output is connected together to generate the combined IR-UWB signal to be transmitted. As part of preliminary explorations, we design the TX unit in FinFET1X and 65nm technologies. Figure 5-4 shows the PSD for the circuits designed in FinFET1X. Notice the peak frequencies located at $f1 = 5.3$ GHz and $f2 = 5.3$ GHz the frequency of oscillation for the VCOs. Continuing this work requires the pulse engineering at the

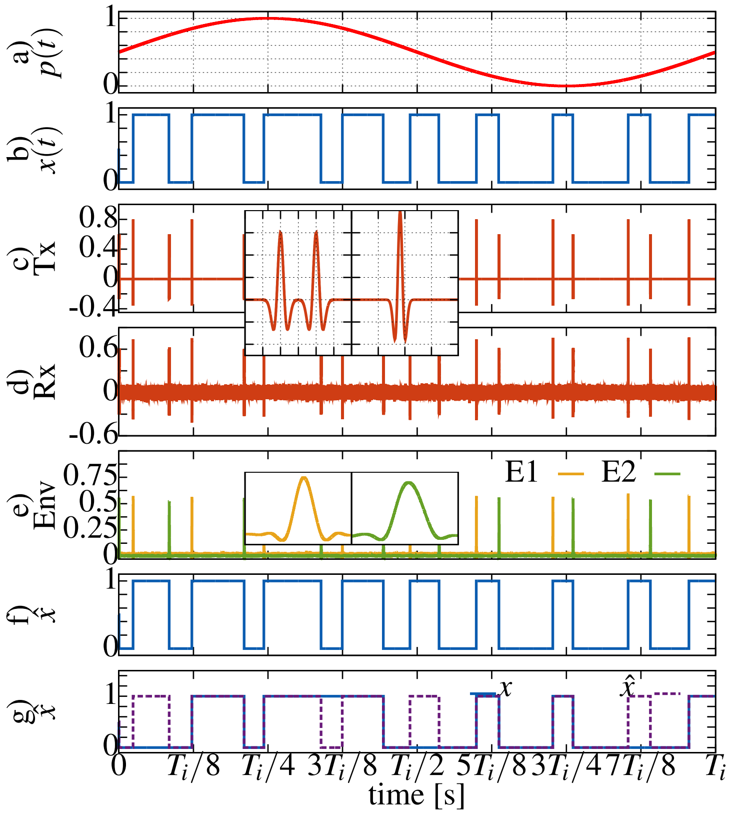

Figure 5-2: a) Analog input for the AΣΔM. A sine wave with period T_i, b) Output of the AΣΔM with a natural frequency $f_c = f_i \times 10$. c) Tx signal generated at the transmitter. d) Rx signal at the receiver. e) Output for both envelope detectors. f) Recovered ΣΔ stream at the output of the SR flip-flop. g) Recovered ΣΔ stream in the presence of noise. Dashed lines represent the expected stream, continuous line is the recovered streams.

following stages including the output buffer and the antenna to reduce the power emission at the bands below 1.61GHz. Table 5.1 summarizes the design parameters and metrics of performance for both technologies.

The average power and energy consumption of the wireless transmitter depends on the data rate which in turn depends on the application. As an example, health-care applications such as electrocardiogram (ECG) and electroencephalogram (EEG) typically require a sample rate of 600Ms/s, the maximum frequency of the signal of interest is $\approx 70Hz$, and the required ADC resolution is 8-12 bits [116]. In ACS there is not a definition of bit. However, to set the system requirements from the application perspective we could derive the Effective Number of Bits (ENoB) for the AΣΔM after applying a band-pass filter. Fig-

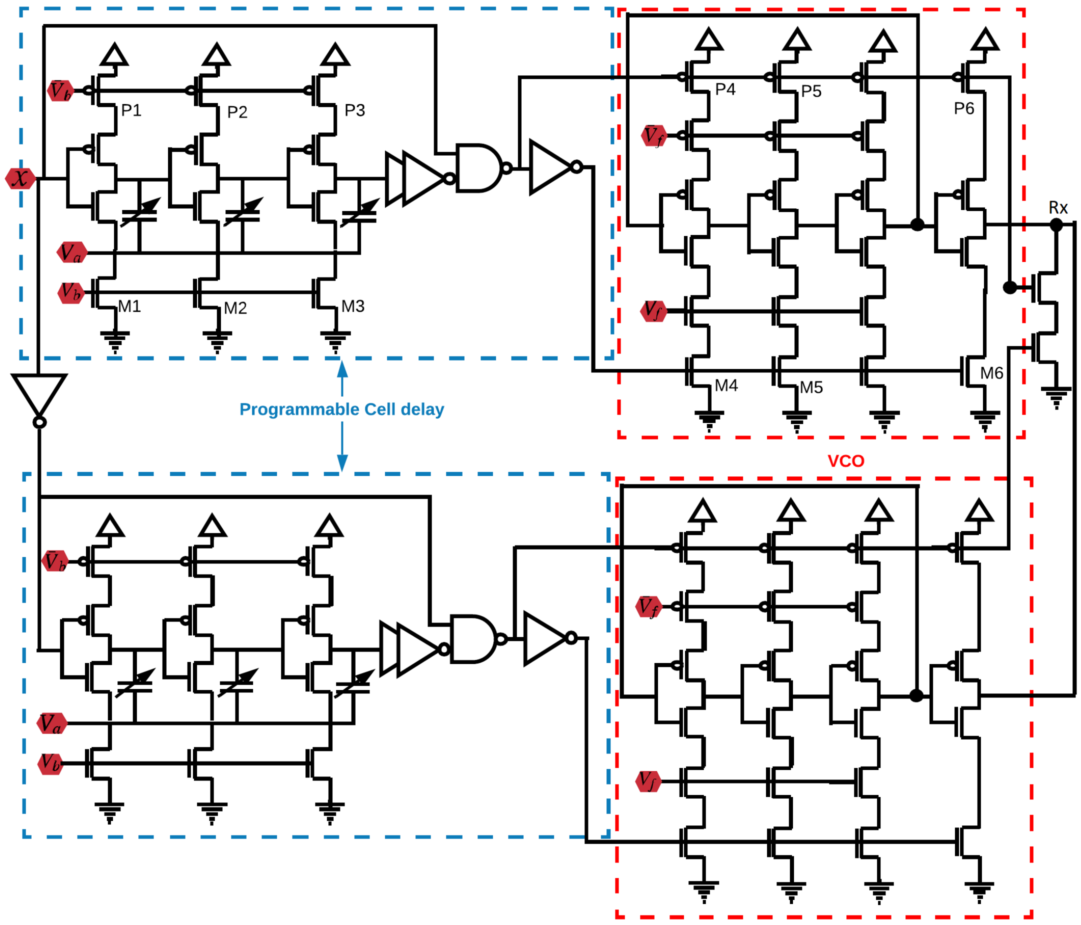

Figure 5-3: Transceiver scheme for the proposed approach to an asynchronous, non-coherent IR-UWB radio for an ACS architecture.The input x is an asynchronous stream. The output is the Rx signal to be fed to a buffer driving an antenna.

ure 5-5a,b shows the power spectral density (PSD) calculated for a ΣΔ stream modulated with a sine wave (Figures 5-2a,b). The input sine waves used to generate Figures 5-5a,b have frequencies of $f_i = f_c/10$ and $f_i = f_c/2$ with f_c the natural frequency of the modulator. Similar to the discrete-time ΣΔ modulator, a band-pass filter, can be placed at the output of the ASDM to obtain the effective-number-of-bits (ENoB). Figure 5-5c shows ENoB for different input frequencies and different filters centered around f_i with bandwidths ranging from $0.1 \times f_i$ to $0.7 \times f_i$. Typical health-care applications require an ENoB of 8-12 bits, thus to build a system tailored towards these applications we can design the

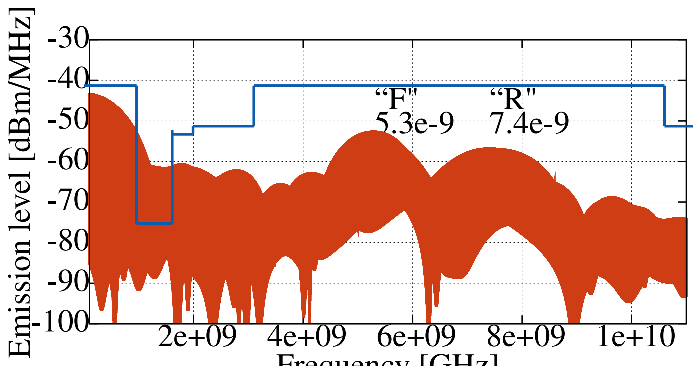

Figure 5-4: Power spectral density (PSD) for signal at the output of the high-Z buffer simulated in Cadence-Spectre with a FinFET1X technology.

Parameter	**Technology**	
	65nm	FinFET1x
R Frequency	**	7.4 GHz
F Frequency	**	5.3 GHz
Power Supply	1.0	0.8 V
Pulse Power	**	6.88μ W
Average power*	**	$15n$ W

Table 5.1: Transmitter parameters for 65nm and FinFET1x technologies. *The average power of transmitting a square wave with a frequency of 1MHz. **Work in progress leaded by Rahul Sreekumar, member of HPLP research group.

AΣΔM natural frequency $f_c = 10 \times f_i$. That is, $f_i = 100KHz$ and $f_c = 1MHz$. Setting f_c sets the maximum frequency of operation in the system, thus the power-supply and metrics of performance such as power, latency and energy.

The continuous time representation has the potential to reduce the transmission power consumption by 66%-77% compared with a fully digital approach. Figure 5-2b shows the ΣΔ stream generated with a modulator designed such that $f_c = 10 \times f_i$. A naive analysis shows that transmitting one period of a sine wave requires 16 pulses (8 "F" and 8 "R"). Using a fully digital approach, assuming a sampling rate of 660S/s [116] and the generation of a pulse per bit in the data representation, the number of pulses required is at least 48-72 for 8-12 bits of resolution respectively. This analysis ignores the cost of synchronization preambles. Our preliminary results show that the pulse generation average power, for an ACS system operating at the maximum frequency f_c is $15nW$.

The naive analysis presented on this section is an introduction to all the work that lies

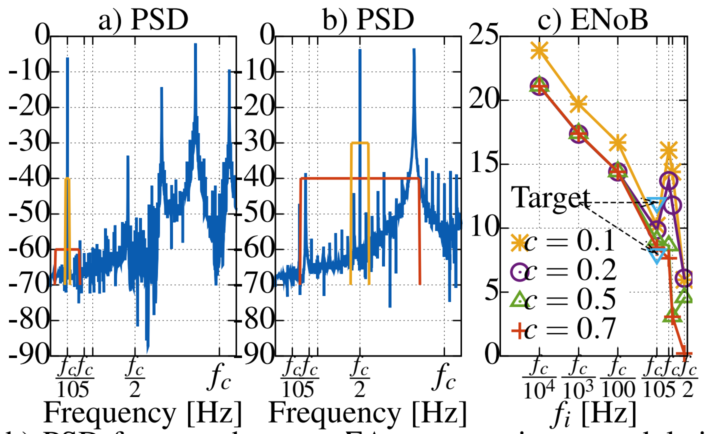

Figure 5-5: a), b) PSD for asynchronous ΣΔ stream using a modulating sine wave with frequency $f_i = \frac{f_c}{10}$ and $f_i = \frac{f_c}{2}$ respectively. The figure also shows ideal pass-band filters frequency response to recover the sine-wave from the ΣΔ stream. c) ENoB versus f_i for different pass-band filters band-width calculated as $c \times f_i$.

ahead:

- The continuous-time nature of the data representation in ACS, requires to re-think the metrics of performance for the ACS-based wireless communication system. Although the pulses sent resembles a digital system, the information each pulse represents in not a bit. Typical approaches can be modeled and compared using metrics such as energy-per-bit or bit-error-rate. Thus further theoretical developments are required to obtain metrics that explains the dynamics of the system. A simple approach to measure the cost and performance of the system is to use the sine-wave as the information unit and compare different systems by the cost of transmitting it.

- Preliminary results show that using a $f_c = 10 \times f_i$ relation for the modulators natural frequency and the maximum input frequency can yield an ENoB of 10 to 12. However, we do not study the band-pass filter characteristics or limitations of hardware implementation. Possible options are the time-to-digital converters usually accompanying the AΣΔM when used as an ADC.

- A not-explored hypothesis is the increase (decrease) of robustness against pulse transmission errors for the ACS wireless transmission unit. An exploration is required to measure the quality of the information received, for example the sine wave, in the presence of missed pulses of falsely detected pulse. Figure 5-2g shows examples of both cases.

5.1.4 Building the first Asynchronous computing on streams based mote

Smartdust, IoT and EI has applications in farming, healthcare, smart cities, transportation, and inventory among others. Each application present different constrains that dictate the characteristics of the smartdust mote. To validate the ideas discussed on this dissertation we need to fabricate and verify the first prototype of a wireless sensor using and asynchronous computing on streams paradigm.

This dissertation focused on developing the theoretical foundations for ACS and understanding its metrics of performance. We need to put together all the pieces to build an ACS based, wireless sensor architecture for the IoT, smartdust or EI.

A SSC based architecture

SSC [10] has potential for spectacular power and area savings in the computing units. However a SSC based architecture would replace conventional digital implementations of computing units with their stochastic counterparts and add several digital-to-stream (stochastic number generator, binary to streams) and stream-to-digital converters at the interfaces between the computing units and the remaining blocks in the architecture (Figure 5-6). As discussed in Chapter 3 one of the major drawbacks for SSC is the elevated cost of these interfaces.

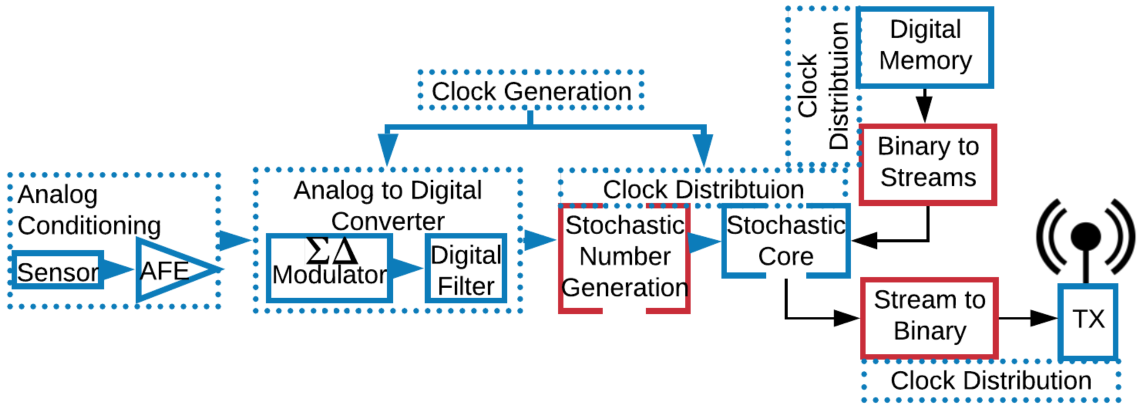

Figure 5-6: A wireless sensor architecture using SSC. In red all the digital-streams-digital-conversion required for SSC-digital integration

An ACS based architecture

Our goal is to achieve seamless node integration in order to capitalize the power savings provided by the stream-based computing units. To do this, we eliminate the digital-stream-digital conversion steps in the architecture, aiming for an end-to-end asynchronous computing with streams data-path. Figure 5-7 shows the proposed sensor architecture.

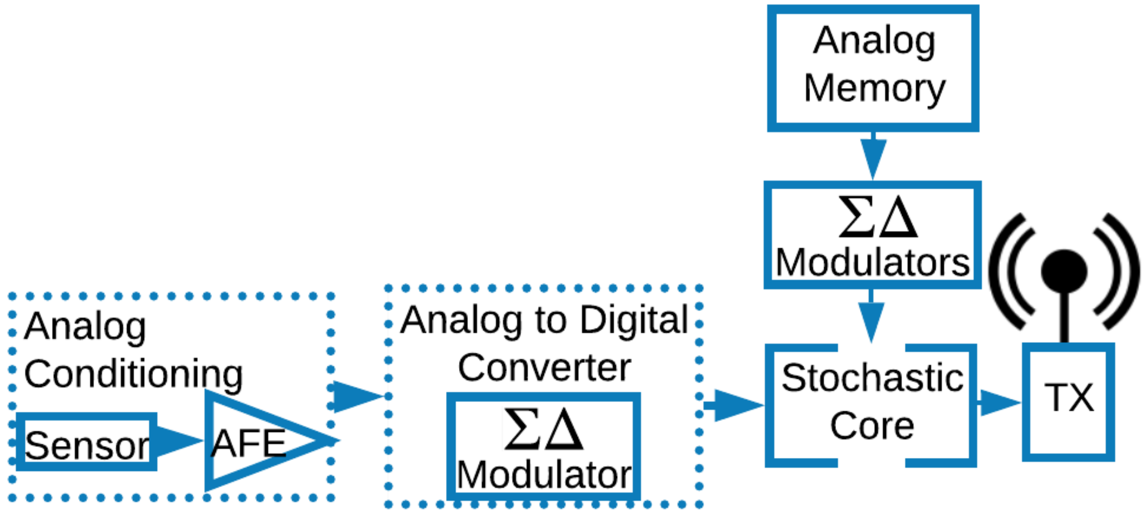

Figure 5-7: Our approach

As discussed in Chapter 3, the ADC is replaced by an AΣΔM which generates continuous time streams from an analog input. The AΣΔM is attractive, for the simplicity of its architecture, the absence of a clock, and the reliability for low current and supply voltages [67]. In Chapter 3, we also discuss how the ΣΔ streams can be connected directly to the stochastic core. Moreover, in Chapter 4 we explore the stochastic core for applications such as FFT and RaF. In section 4.3 we discuss the use of analog-memory and near-analog memory computing using ACS. Section 5.1.3 gives a brief glimpse for the wireless communication system requirements and considerations.

Appendix A

List of publications

A.1 Peer-reviewed Journals

[J1] P. Gonzalez-Guerrero, T. Tracy, X. Guo, M. Lejani, K. Skadron, M. Stan, "Towards low power machine learning using asynchronous computing with streams," *In preparation*

[J2] P. Gonzalez-Guerrero, M. Stan, "Area-Efficient Low-Latency FFT Design Based on Asynchronous Computing with streams," Transactions on Circuits and Systems I (TCAS-I), January 2020. *Submitted*

[J3] X. Guo, V. Verma, **P. Gonzalez-Guerrero**, S. Mosanu, M. Stan, "Back to the Future: Digital Circuit Design in the FinFET Era," Journal of Low Power Electronics (JOLPE), Vol. 13, No. 3, pp. 338–355, 2017.

A.2 Peer-reviewed Conferences

[C1] P. Gonzalez-Guerrero, T. Tracy, Guo Xinfei, M. Stan, "Towards low power machine learning using asynchronous computing with streams", The tenth International green and sustainable computing conference (IGSC), 2019. *Accepted*

[C2] P. Gonzalez-Guerrero, M. Stan, "Asynchronous Stochastic Computing", Asilomar conference on signals, systems, and computers, 2019. *Accepted*, **Nominated best paper award**.

[C3] P. Gonzalez-Guerrero, S. Wilson, M. Stan, "Error-latency trade-off for computing on asynchronous sigma delta streams", IEEE Systems on Chip conference (SoCC), 2019.

[C4] P. Gonzalez-Guerrero, X. Guo, M. Stan, "ASC-FFT: Area-Efficient Low-Latency FFT Design Based on Asynchronous Stochastic Computing", 2019 Latin American Symposium on Circuits and Systems (LASCAS). **Best paper award**.

[C5] P. Gonzalez-Guerrero, X. Guo, M. Stan, "SCSD: Towards Low Power Stochastic Computing Using Sigma Delta Streams ", 2018 IEEE International Conference on Rebooting Computing (ICRC).

[C6] X. Guo, V. Verma, **P. Gonzalez-Guerrero** and M. Stan, “When “things” get older - Exploring Circuit Aging in IoT Applications,” In Proc. of International Symposium on Quality Electronic Design (ISQED), Santa Clara, CA, 2018.

[C7] P. Gonzalez-Guerrero, M. Stan, "Ultra-low-power dual-phase latch based digital accelerator for continuous monitoring of wheezing episodes", IEEE SOI-3D-Subthreshold Microelectronics Technology Unified Conference (S3S), 2017.

Bibliography

[1] A. Alaghi et al. Stochastic circuits for real-time image-processing applications. In *2013 50th ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6, May 2013.

[2] Weikang Qian et al. An Architecture for Fault-Tolerant Computation with Stochastic Logic. *IEEE Transactions on Computers*, 60(1):93–105, 1 2011.

[3] A. Shafiee et al. Isaac: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars. In *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, pages 14–26, June 2016.

[4] M. Hu et al. Dot-product engine for neuromorphic computing: Programming 1t1m crossbar to accelerate matrix-vector multiplication. In *2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6, June 2016.

[5] J. Lu et al. A 1 tops/w analog deep machine-learning engine with floating-gate storage in 0.13 μm cmos. *IEEE Journal of Solid-State Circuits*, 50(1):270–281, Jan 2015.

[6] Matthew Jerry, Pai-Yu Chen, Jianchi Zhang, Pankaj Sharma, Kai Ni, Shimeng Yu, and Suman Datta. Ferroelectric FET analog synapse for acceleration of deep neural network training. In *2017 IEEE International Electron Devices Meeting (IEDM)*, pages 1–6. IEEE, 12 2017.

[7] M. Reza Mahmoodi et al. An ultra-low energy internally analog, externally digital vector-matrix multiplier based on nor flash memory technology. In *Proceedings of the 55th Annual Design Automation Conference*, DAC ’18, pages 22:1–22:6, New York, NY, USA, 2018. ACM.

[8] Y. Chae et al. A 2.1 m pixels, 120 frame/s cmos image sensor with column-parallel-delta-sigma-adc architecture. *IEEE Journal of Solid-State Circuits*, 46(1):236–247, Jan 2011.

[9] Tommy Tracy, Yao Fu, Indranil Roy, Eric Jonas, and Paul Glendenning. Towards machine learning on the automata processor. In *International Conference on High Performance Computing*, pages 200–218. Springer, 2016.

[10] B. R. Gaines and B. R. Stochastic computing. In *Proceedings of the April 18-20, 1967, spring joint computer conference on - AFIPS '67 (Spring)*, page 149, New York, New York, USA, 1967. ACM Press.

[11] L. Patricia Gonzalez-Guerrero et al. ASC-FFT: Area-Efficient Low-Latency FFT Design Based on Asynchronous Stochastic Computing. In *2019 Latin American Symposium on Circuits and Systems (LASCAS)*. IEEE, 2019.

[12] W. Tang and E. Culurciello. A non-coherent fsk-ook uwb impulse radio transmitter for clock-less synchronization. In *2011 IEEE International Symposium of Circuits and Systems (ISCAS)*, pages 1295–1298, May 2011.

[13] Kevin Ashton et al. That 'internet of things' thing. *RFID journal*, 22(7):97–114, 2009.

[14] Kris Pister. Smart dust. https://people.eecs.berkeley.edu/~pister/SmartDust/SmartDustBAA97-43-Abstract.pdf, 2001.

[15] Joseph M Kahn, Randy H Katz, and Kristofer SJ Pister. Next century challenges: mobile networking for "smart dust". In *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 271–278. ACM, 1999.

[16] A. Keshavarzi and W. van den Hoek. Edge intelligence—on the challenging road to a trillion smart connected iot devices. *IEEE Design Test*, 36(2):41–64, April 2019.

[17] Alfred Hermida. Caution over computerised world. http://news.bbc.co.uk/2/hi/technology/3340491.stm, 2003.

[18] The economist. The smart dust revolution. https://www.economist.com/news/2003/11/20/the-smart-dust-revolution, 2003.

[19] Azonano. Precision agriculture - nanotech methods used, such as 'smart dust', smart fields' and nanosensors. https://www.azonano.com/article.aspx?ArticleID=1318#_What_is_\T1\textquoteleftSmart_Dust\T1\textquoterightandWhoInvent, 2005.

[20] Alicia Klinefelter, Nathan E. Roberts, Yousef Shakhsheer, Patricia Gonzalez, Aatmesh Shrivastava, Abhishek Roy, Kyle Craig, Muhammad Faisal, James Boley, Seunghyun Oh, Yanqing Zhang, Divya Akella, David D. Wentzloff, and Benton H. Calhoun. 21.3 A 6.45&amp;#x03BC;W self-powered IoT SoC with integrated energy-harvesting power management and ULP asymmetric radios. In *2015 IEEE International Solid-State Circuits Conference - (ISSCC) Digest of Technical Papers*, pages 1–3. IEEE, 2 2015.

[21] Yanqing Zhang, Fan Zhang, Yousef Shakhsheer, Jason D. Silver, Alicia Klinefelter, Manohar Nagaraju, James Boley, Jagdish Pandey, Aatmesh Shrivastava, Eric J. Carlson, Austin Wood, Benton H. Calhoun, and Brian P. Otis. A Batteryless 19 $\mu$W MICS/ISM-Band Energy Harvesting Body Sensor Node SoC for ExG Applications. *IEEE Journal of Solid-State Circuits*, 48(1):199–213, 1 2013.

[22] Y. Nait Malek, A. Kharbouch, H. El Khoukhi, M. Bakhouya, V. De Florio, D. El Ouadghiri, S. Latre, and C. Blondia. On the use of iot and big data technologies for real-time monitoring and data processing. *Procedia Computer Science*, 113:429 – 434, 2017. The 8th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN 2017) / The 7th International Conference on Current and Future Trends of Information and Communication Technologies in Healthcare (ICTH-2017) / Affiliated Workshops.

[23] Apple. Machine learning journal. https://machinelearning.apple.com/2017/10/01/hey-siri.html, 2017.

[24] Raphael Tang, Weijie Wang, Zhucheng Tu, and Jimmy Lin. An Experimental Analysis of the Power Consumption of Convolutional Neural Networks for Keyword Spotting. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5479–5483. IEEE, 4 2018.

[25] Raspberry Pi. Teach, Learn, and Make with Raspberry Pi. https://www.raspberrypi.org/.

[26] A. F. Yeknami, X. Wang, S. Imani, A. Nikoofard, I. Jeerapan, J. Wang, and P. P. Mercier. A 0.3v biofuel-cell-powered glucose/lactate biosensing system employing a 180nw 64db snr passive sigma delta adc and a 920mhz wireless transmitter. In *2018 IEEE International Solid - State Circuits Conference - (ISSCC)*, pages 284–286, Feb 2018.

[27] L. Lin, S. Jain, and M. Alioto. A 595pw 14pj/cycle microcontroller with dual-mode standard cells and self-startup for battery-indifferent distributed sensing. In *2018*

IEEE International Solid - State Circuits Conference - (ISSCC), pages 44–46, Feb 2018.

[28] D. Jang and S. Cho. A 43.4μw photoplethysmogram-based heart-rate sensor using heart-beat-locked loop. In *2018 IEEE International Solid - State Circuits Conference - (ISSCC)*, pages 474–476, Feb 2018.

[29] M. Yang, C. Yeh, Y. Zhou, J. P. Cerqueira, A. A. Lazar, and M. Seok. A 1μw voice activity detector using analog feature extraction and digital deep neural network. In *2018 IEEE International Solid - State Circuits Conference - (ISSCC)*, pages 346–348, Feb 2018.

[30] Dariusz Kościelnik and Marek Miśkowicz. Asynchronous Sigma-Delta analog-to digital converter based on the charge pump integrator. *Analog Integrated Circuits and Signal Processing*, 55(3):223–238, 6 2008.

[31] T. Karnik, D. Kurian, P. Aseron, R. Dorrance, E. Alpman, A. Nicoara, R. Popov, L. Azarenkov, M. Moiseev, L. Zhao, S. Ghosh, R. Misoczki, A. Gupta, M. Akhila, S. Muthukumar, S. Bhandari, Y. Satish, K. Jain, R. Flory, C. Kanthapanit, E. Quijano, B. Jackson, H. Luo, S. Kim, V. Vaidya, A. Elsherbini, R. Liu, F. Sheikh, O. Tickoo, I. Klotchkov, M. Sastry, S. Sun, M. Bhartiya, A. Srinivasan, Y. Hoskote, H. Wang, and V. De. A cm-scale self-powered intelligent and secure iot edge mote featuring an ultra-low-power soc in 14nm tri-gate cmos. In *2018 IEEE International Solid - State Circuits Conference - (ISSCC)*, pages 46–48, Feb 2018.

[32] M. Konijnenburg, R. van Wegberg, S. Song, H. Ha, W. Sijbers, J. Xu, S. Stanzione, C. van Liempd, D. Biswas, A. Breeschoten, P. Vis, C. Van Hoof, and N. Van Helleputte. 22.1 a 769μw battery-powered single-chip soc with ble for multi-modal vital sign health patches. In *2019 IEEE International Solid- State Circuits Conference - (ISSCC)*, pages 360–362, Feb 2019.

[33] Peng Li, David J Lilja, Weikang Qian, Kia Bazargan, and Marc D Riedel. Computation on Stochastic Bit Streams Digital Image Processing Case Studies. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 22(3):449–462, 3 2014.

[34] Armin Alaghi and John P. Hayes. Exploiting correlation in stochastic circuit design. In *2013 IEEE 31st International Conference on Computer Design (ICCD)*, pages 39–46. IEEE, 10 2013.

[35] Patricia Gonzalez-Guerrero, Xinfei Guo, and Mircea Stan. SC-SD: Towards Low Power Stochastic Computing using Sigma Delta Streams. In *2018 IEEE International Conference on Rebooting Computing (ICRC)*. IEEE, 2018.

[36] M. H. Najafi, D. J. Lilja, M. Riedel, and K. Bazargan. Polysynchronous stochastic circuits. In *2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 492–498. IEEE, 1 2016.

[37] B. D. Brown and H. C. Card. Stochastic neural computation. i. computational elements. *IEEE Transactions on Computers*, 50(9):891–905, Sep. 2001.

[38] Bo Yuan et al. Area-Efficient Scaling-Free DFT/FFT Design Using Stochastic Computing. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 63(12):1131–1135, 12 2016.

[39] Ensar Vahapoglu and Mustafa Altun. Accurate Synthesis of Arithmetic Operations with Stochastic Logic. In *2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pages 415–420. IEEE, 7 2016.

[40] Yifei Liu, Paul Furth, and Wei Tang. Hardware-Efficient Delta Sigma-Based Digital Signal Processing Circuits for the Internet-of-Things. *Journal of Low Power Electronics and Applications*, 5(4):234–256, 11 2015.

[41] N. Onizawa, D. Katagiri, W. J. Gross, and T. Hanyu. Analog-to-stochastic converter using magnetic-tunnel junction devices. In *2014 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH)*, pages 59–64, July 2014.

[42] Arash Ardakani et al. Hardware implementation of FIR/IIR digital filters using integral stochastic computation. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6540–6544. IEEE, 3 2016.

[43] Ran Wang et al. Design and evaluation of stochastic FIR filters. In *2015 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM)*, pages 407–412. IEEE, 8 2015.

[44] K. Kim et al. Dynamic energy-accuracy trade-off using stochastic computing in deep neural networks. In *2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6, June 2016.

[45] K. Sanni et al. Fpga implementation of a deep belief network architecture for character recognition using stochastic computation. In *2015 49th Annual Conference on Information Sciences and Systems (CISS)*, pages 1–5, March 2015.

[46] A. Ren et al. Designing reconfigurable large-scale deep learning systems using stochastic computing. In *2016 IEEE International Conference on Rebooting Computing (ICRC)*, pages 1–7, Oct 2016.

[47] Z. Li et al. Dscnn: Hardware-oriented optimization for stochastic computing based deep convolutional neural networks. In *2016 IEEE 34th International Conference on Computer Design (ICCD)*, pages 678–681, Oct 2016.

[48] J. Li et al. Towards acceleration of deep convolutional neural networks using stochastic computing. In *2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 115–120, Jan 2017.

[49] P. Gonzalez-Guerrero et al. Scsd: Towards low power stochastic computing using sigma delta streams. In *2018 IEEE International Conference on Rebooting Computing (ICRC)*, pages 1–8, Nov 2018.

[50] Xinfei Guo et al. Back to the future: Digital circuit design in the finfet era. *Journal of Low Power Electronics*, 13(3):338–355, 2017.

[51] X. Guo et al. When things get older: Exploring circuit aging in iot applications. In *2018 19th International Symposium on Quality Electronic Design (ISQED)*, pages 296–301, March 2018.

[52] Yanqing Zhang et al. Hold time closure for subthreshold circuits using a two-phase, latch based timing method. In *2013 IEEE SOI-3D-Subthreshold Microelectronics Technology Unified Conference (S3S)*, pages 1–2. IEEE, 10 2013.

[53] Nicolas Privault. *Notes on Markov chains*. Open Library, 2011.

[54] Solomon Frederick. *Probability and Stochastic Processes*. Prentice-Hall, Inc, Englewood Cliffs, 1987.

[55] Charles A. Poynton. SMPTE Tutorial: Gamma and its Disguises: The Nonlinear Mappings of Intensity in Perception, CRTs, Film, and Video. *SMPTE Journal*, 102(12):1099–1108, 12 1993.

[56] BP. Brain power. http://www.brain-power.com/, 2019.

[57] Robert LiKamWa et al. Draining our glass: An energy and heat characterization of google glass. In *Proceedings of 5th Asia-Pacific Workshop on Systems*, APSys '14, pages 10:1–10:7, New York, NY, USA, 2014. ACM.

[58] Norman P. Jouppi et al. In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the 44th Annual International Symposium on Computer Architecture*, ISCA '17, pages 1–12, New York, NY, USA, 2017. ACM.

[59] S. L. Bade and B. L. Hutchings. Fpga-based stochastic neural networks-implementation. In *Proceedings of IEEE Workshop on FPGA's for Custom Computing Machines*, pages 189–198, April 1994.

[60] J. Ramirez-Angulo. A compact current controlled cmos waveform generator. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 39(12):883–885, Dec 1992.

[61] D.A. Johns and D.M. Lewis. Design and analysis of delta-sigma based IIR filters. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 40(4):233–240, 4 1993.

[62] Naman Saraf, Kia Bazargan, David J Lilja, and Marc D Riedel. IIR filters using stochastic arithmetic. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2014*, pages 1–6, New Jersey, 2014. IEEE Conference Publications.

[63] F. Maloberti. Non conventional signal processing by the use of sigma delta technique: a tutorial introduction. In *IEEE International Symposium on Circuits and Systems*, volume 6, pages 2645–2648. IEEE, 1992.

[64] Victor da Fonte Dias. Signal processing in the sigma-delta domain. *Microelectronics Journal*, 26(6):543–562, 9 1995.

[65] SL Toral, JM Quero, JG Ortega, and LG Franquelo. Stochastic A/D sigma-delta converter on FPGA. In *Circuits and Systems, 1999. 42nd Midwest Symposium on*, volume 1, pages 35–38. IEEE, 1999.

[66] Armin Alaghi, Cheng Li, and John P Hayes. Stochastic circuits for real-time image-processing applications. In *Proceedings of the 50th Annual Design Automation Conference on - DAC '13*, page 1. ACM Press, 2013.

[67] E. Roza. Analog-to-digital conversion via duty-cycle modulation. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 44(11):907–914, 1997.

[68] S.J. Daubert and D. Vallancourt. A transistor-only current-mode Sigma Delta modulator. In *Proceedings of the IEEE Custom Integrated Circuits Conference*, pages 1–24. IEEE, 1991.

[69] D.G. Nairn and C.A.T. Salama. Algorithmic analogue/digital convertor based on current mirrors. *Electronics Letters*, 24(8):471, 1988.

[70] Dazhi Wei, Vaibhav Garg, and John G Harris. An asynchronous delta-sigma converter implementation. In *IEEE International Symposium on Circuits and Systems*, page 4. IEEE, 2006.

[71] P.J. Crawley and G.W. Roberts. Switched-current sigma-delta modulation for A/D conversion. In *[Proceedings] IEEE International Symposium on Circuits and Systems*, volume 3, pages 1320–1323. IEEE, 1992.

[72] M. Hassan Najafi, Shiva Jamali-Zavareh, David J. Lilja, Marc D. Riedel, Kia Bazargan, and Ramesh Harjani. Time-Encoded Values for Highly Efficient Stochastic Circuits. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 25(5):1644–1657, 5 2017.

[73] Jung Hwan Choi, Jayathi Murthy, and Kaushik Roy. The effect of process variation on device temperature in finFET circuits. In *IEEE/ACM International Conference on Computer-Aided Design*, pages 747–751. IEEE, 2007.

[74] Hideyuki Ichihara, Shota Ishii, Daiki Sunamori, Tsuyoshi Iwagaki, and Tomoo Inoue. Compact and accurate stochastic circuits with shared random number sources. In *International Conference on Computer Design*, pages 361–366. IEEE, 10 2014.

[75] Kun Lin, Kan Zhao, E. Chui, A. Krone, and J. Nohrden. Digital filters for high performance audio delta-sigma analog-to-digital and digital-to-analog conversions. In *Proceedings of Third International Conference on Signal Processing (ICSP'96)*, volume 1, pages 59–63. IEEE.

[76] Mohammed Alawad et al. Sketching computation with stochastic processing engines. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 13(3):46, 2017.

[77] A. Alaghi et al. Stochastic circuits for real-time image-processing applications. In *2013 50th ACM/EDAC/IEEE Design Automation Conference (DAC)*, May 2013.

[78] L. Kull et al. A 10b 1.5gs/s pipelined-sar adc with background second-stage common-mode regulation and offset calibration in 14nm cmos finfet. In *2017 IEEE International Solid-State Circuits Conference (ISSCC)*, pages 474–475, Feb 2017.

[79] Bor-Shing Lin and Tian-Shiue Yen. An FPGA-based rapid wheezing detection system. *International journal of environmental research and public health*, 11(2):1573–93, 1 2014.

[80] Syamimi Mardiah Shaharum, Kenneth Sundaraj, and Rajkumar Palaniappan. A survey on automated wheeze detection systems for asthmatic patients. *Bosnian journal of basic medical sciences*, 12(4):249–55, 11 2012.

[81] Styliani A. Taplidou and Leontios J. Hadjileontiadis. Wheeze detection based on time-frequency analysis of breath sounds. *Computers in Biology and Medicine*, 37(8):1073–1083, 8 2007.

[82] Pascal Meinerzhagen, S. M. Yasser Sherazi, Andreas Burg, and Joachim Neves Rodrigues. Benchmarking of Standard-Cell Based Memories in the Sub-$V_{\rm T}$ Domain in 65-nm CMOS Technology. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 1(2):173–182, 6 2011.

[83] Mingoo Seok, Gregory Chen, Scott Hanson, Michael Wieckowski, David Blaauw, and Dennis Sylvester. CAS-FEST 2010: Mitigating Variability in Near-Threshold Computing. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 1(1):42–49, 3 2011.

[84] Xin Xiao et al. An Efficient FFT Engine With Reduced Addressing Logic. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 55(11):1149–1153, 11 2008.

[85] Bo Yuan et al. Area-Efficient Error-Resilient Discrete Fourier Transformation Design using Stochastic Computing. In *Proceedings of the 26th edition on Great Lakes Symposium on VLSI*, pages 33–38. ACM, 2016.

[86] J.A. Johnston. Parallel pipeline fast fourier transformer. *IEE Proceedings F Communications, Radar and Signal Processing*, 130(6):564, 1983.

[87] Liang Yang et al. An efficient locally pipelined FFT processor. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 53(7):585–589, 7 2006.

[88] R. LiKamWa et al. Redeye: Analog convnet image sensor architecture for continuous mobile vision. In *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, pages 255–266, June 2016.

[89] H. Li, K. Ota, and M. Dong. Learning iot in edge: Deep learning for the internet of things with edge computing. *IEEE Network*, 32(1):96–101, Jan 2018.

[90] Bingzhe Li, M Hassan Najafi, and David J Lilja. Low-cost stochastic hybrid multiplier for quantized neural networks. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 15(2):18, 2019.

[91] P. Li, D. J. Lilja, W. Qian, K. Bazargan, and M. D. Riedel. Computation on stochastic bit streams digital image processing case studies. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 22(3):449–462, March 2014.

[92] Patricia Gonzalez-Guerrero, Stephen G. Wilson, and Mircea R Stan. Error-latency Trade-off for Asynchronous Stochastic Computing with ΣΔ Streams for the IoT. In *System on chip conference*. IEEE, 2019.

[93] Patricia Gonzalez-Guerrero and Mircea R. Stan. Asynchronous Stochastic Computing. In *Asilomar conference on signals systems and computers*. IEEE, 2019 (Submitted).

[94] Robert LiKamWa et al. Energy characterization and optimization of image sensing toward continuous mobile vision. In *Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '13, pages 69–82, New York, NY, USA, 2013. ACM.

[95] J Kevin O'regan. Solving the" real" mysteries of visual perception: The world as an outside memory. *Canadian Journal of Psychology/Revue canadienne de psychologie*, 46(3):461, 1992.

[96] Irina Burciu, Thomas Martinetz, and Erhardt Barth. Sensing forest for pattern recognition. In Jacques Blanc-Talon, Rudi Penne, Wilfried Philips, Dan Popescu, and Paul Scheunders, editors, *Advanced Concepts for Intelligent Vision Systems*, pages 126–137, Cham, 2017. Springer International Publishing.

[97] J. Wadden, T. Tracy, E. Sadredini, L. Wu, C. Bo, J. Du, Y. Wei, J. Udall, M. Wallace, M. Stan, and K. Skadron. Automatazoo: A modern automata processing benchmark suite. In *2018 IEEE International Symposium on Workload Characterization (IISWC)*, pages 13–24, Sep. 2018.

[98] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

[99] R. Wang, J. Han, B. Cockburn, and D. Elliott. Design and evaluation of stochastic fir filters. In *2015 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM)*, pages 407–412, Aug 2015.

[100] B. Yuan, Y. Wang, and Z. Wang. Area-efficient scaling-free dft/fft design using stochastic computing. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 63(12):1131–1135, Dec 2016.

[101] C. H. Van Berkel, M. B. Josephs, and S. M. Nowick. Applications of asynchronous circuits. *Proceedings of the IEEE*, 87(2):223–233, Feb 1999.

[102] Y. Kang, W. Huang, S. Yoo, D. Keen, Z. Ge, V. Lam, P. Pattnaik, and J. Torrellas. Flexram: Toward an advanced intelligent memory system. In *2012 IEEE 30th International Conference on Computer Design (ICCD)*, pages 5–14, Sep. 2012.

[103] S. Li, A. O. Glova, X. Hu, P. Gu, D. Niu, K. T. Malladi, H. Zheng, B. Brennan, and Y. Xie. Scope: A stochastic computing engine for dram-based in-situ accelerator. In *2018 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 696–709, Oct 2018.

[104] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[105] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[106] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[107] Mohammed Affan Zidan, Hossam Aly Hassan Fahmy, Muhammad Mustafa Hussain, and Khaled Nabil Salama. Memristor-based memory: The sneak paths problem and solutions. *Microelectronics Journal*, 44(2):176–183, 2013.

[108] M. Le Gallo, A. Sebastian, G. Cherubini, H. Giefers, and E. Eleftheriou. Compressed sensing recovery using computational memory. In *2017 IEEE International Electron Devices Meeting (IEDM)*, pages 28.3.1–28.3.4, Dec 2017.

[109] J. Lu and J. Holleman. A floating-gate analog memory with bidirectional sigmoid updates in a standard digital process. In *2013 IEEE International Symposium on Circuits and Systems (ISCAS2013)*, pages 1600–1603, May 2013.

[110] F Merrikh Bayat, Xinjie Guo, HA Om'Mani, N Do, Konstantin K Likharev, and Dmitri B Strukov. Redesigning commercial floating-gate memory for analog computing applications. In *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1921–1924. IEEE, 2015.

[111] Raj Johri, Ravindra Singh Kushwah, Raghvendra Singh, and Shyam Akashe. Modeling and simulation of high speed 8t sram cell. In *Proceedings of Seventh International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA 2012)*, pages 245–251. Springer, 2013.

[112] L. Kull et al. A 10b 1.5gs/s pipelined-sar adc with background second-stage common-mode regulation and offset calibration in 14nm cmos finfet. In *2017 IEEE International Solid-State Circuits Conference (ISSCC)*, pages 474–475, Feb 2017.

[113] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[114] Tommy Tracy. Rfautomata. `https://github.com/tjt7a/ANMLZoo/tree/master/RandomForest/code`, 2019.

[115] Huseyin Arslan, Chen Zhi-Ning, and Di Benedetto Maria-Gabriella. *Ultra Wideband Wireless communication*. John Wiley & Sons, 2006.

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