An Asynchronous Spike Event Coding Scheme for Programmable Analogue Arrays and its Computational Applications

Luiz Carlos Paiva Gouveia

A thesis submitted for the degree of Doctor of Philosophy.
The University of Edinburgh.
September 2011

Abstract

This work is the result of the definition, design and evaluation of a novel method to interconnect the computational elements - commonly known as Configurable Analogue Blocks (CABs) - of a programmable analogue array. This method is proposed for total or partial replacement of the conventional methods due to serious limitations of the latter in terms of scalability.

With this method, named Asynchronous Spike Event Coding (ASEC) scheme, analogue signals from CABs outputs are encoded as time instants (spike events) dependent upon those signals activity and are transmitted asynchronously by employing the Address Event Representation (AER) protocol. Power dissipation is dependent upon input signal activity and no spike events are generated when the input signal is constant.

On-line, programmable computation is intrinsic to ASEC scheme and is performed without additional hardware. The ability of the communication scheme to perform computation enhances the computation power of the programmable analogue array. The design methodology and a CMOS implementation of the scheme are presented together with test results from prototype integrated circuits (ICs).

Declaration of originality

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, that this work was developed in the School of Engineering at The University of Edinburgh and that this work has not been submitted for any other degree or professional qualification except as specified.

Luiz Carlos Paiva Gouveia

Acknowledgements

During this thesis project, many people helped to create an enjoyable and collaborative environment within the Institute for Micro and Nano Systems (IMNS). Although all of them have been important for different reasons, I would like to thank particularly to the following friends.

Dr. Alister Hamilton, who was a great supervisor, offering me total freedom and conditions to the development of the thesis. I am also thankful to his patience. Dr. Thomas Jacob Koickal, who acted as an informal second supervisor. He helped me with bits of everything, from English revision to presentation skills, from project incomes prediction to equipment acquisition and much more. Dr. Martin Reekie, who offered me his time and his deep knowledge on analogue circuits. Dr. Keith Muir, who kindly offered me his PCB layout from his project, which I used as a starting point for my ons PCB designs. He instructed me on the joys of PCB design and FPGA programming tools. He also advised about test equipment. Dr. Katherine Cameron, who helped me on conducting me throughout the School procedures and locations and her coordination of the Paper Club, where lots of ideas were shared. Dr. Simeon Bamford, with whom I shared lots of experiences and had many productive discussions about the thesis, from the incomes to several technical details. He also redesigned the AER circuitry used on this thesis. Dr. Giacomo Indiveri, Dr. Shih-Chii Liu and Dr. Tobi Delbrück from ETH Neuroscience Center in Zurich. They made available the AER circuitry used by Dr. Simeon Bamford in this thesis. They were also very kind and helpful whenever I meet them in conferences and workshops. Dr. Kwabena Boahen of Stanford University for clarifications about the AER protocol. Everyone that has worked with me in room 2.2 of Alrick Building for a light, friendly and productive environment.

I would also like to thank to the Engineering and Physical Sciences Research Council (EPSRC) which funded this project under the grant to University of Edinburgh (EP/C015789/1) and the U.S. National Science Foundation and other sponsors for financial support to the 2007 Telluride Neuromorphic Engineering Workshop.

Although not have contributed directly to the thesis itself, my special thanks to my wife Andreia Nobre for her love, motivation, dedication and patience. Also to my son Fabio, who took away many hours of my sleep, but always paid back with a smile.

Contents

List of Figures

List of Tables

Acronyms and abbreviations

ADM	Asynchronous Delta Modulation
AER	Address Event Representation
ANN	Artificial Neural Networks
ASDM	Asynchronous Sigma Delta modulation
ASEC	Asynchronous Spike Event Coding
ASIC	Application Specific Integrated Circuit
BSD	Binary Spike Delta modulation
BSSD	Binary Spike Sigma-Delta modulation
CAB	Configurable Analogue Block
CMOS	Complimentary Metal-Oxide-Substrate
ENoB	Effective Number of Bits
CE	Channel Efficiency
FPAA	Field Programmable Analogue Array
FPGA	Field Programmable Gate Array
IC	Integrated Circuit
I&F	Integrate-and-fire neuron
LUT	Look-up Table
MOS	Metal-Oxide-Substrate
OP-AMP	Operational Amplifier
OTA	Operational Transconductance Amplifier
PCB	Printed Circuit Board
TSD	Ternary Spike Delta modulation
TSSD	Ternary Spike Sigma-Delta modulation
VLSI	Very Large Scale Integration

Chapter 1
Introduction

1.1 Motivation

Electronic engineering attempt to realize mathematical models using electrical devices configured and connected appropriately. In general such models can be implemented using digital, analogue or combined approaches. Each approach has their own paradigms, techniques, advantages and limitations.

Analogue systems use continuous variables to represent information whilst digital ones uses discrete - usually binary - numbers. The continuous characteristic of the information implies analogue systems are more vulnerable to a wide number of physical effects than digital ones. Therefore the digital approach tends to make complex system designs easier and faster than the analogue approach. This characteristic helps to explain the ever-increasing popularity of digital designs.

The robustness of digital designs allows for automation and flexibility and both of them are difficult to achieve with analogue designs. These characteristics allow for complex systems like digital Central Processing Units (CPUs), micro-controllers, Digital Signal Processors (DSPs) and Field-Programmable Gate Arrays (FPGAs). In particular, FPGAs are digital systems used specially for rapid prototyping. They contain a large number of basic digital circuits which can be configured and connected to implement specific functions. Doing this is much faster than designing a different system whenever you need to implement a different function.

With these crucial advantages of digital circuits, analogue designs survive due to specific applications. In particular, analogue circuits are used to interface with real world because most of its information — measurements and controls — are continuous variables as well. Moreover analogue designs usually present a more fitted solution than digital circuits because they are usually smaller, spend less power and presents a higher processing speed for the same applications.

Therefore, if the main reason that drives the popularity of the digital circuits is design automation and operation flexibility, it is a reasonable ambition to incorporate these characteristics to an analogue design. This would shorten analogue design cycles, provides reconfiguration levels similar to the digital case and, furthermore, will benefit from the analogue properties listed in the previous paragraph.

Some efforts have been made on the automation field, as in Analogue-to-Digital Converters (ADC) designs [1]. Regarding the flexibility, studies have been made on analogue CPUs [2, 3] and on programmable analogue arrays.

Several researchers in institutions and companies have been trying to achieve a level of programmability in analogue systems similar to digital arrays. Some had coined terms as “Field-Programmable Analogue Arrays” (FPAA) [4] and “Field-Programmable Mixed-signal Arrays” (FPMA) [5] to define the class of circuits that are the analogue and mixed-signal counterparts of the digital FPGA, respectively.

In general these architectures are built from a number of basic programmable processing blocks, known as Configurable Analogue Blocks (CABs). The configurable blocks are then interconnected using specific configurable signal routing. This signal routing is usually implemented in a similar fashion as it is in a digital array, i.e., using switched-matrices: a collection of wires and switches connected in a special pattern, defined by design requirements. However this routing method imposes serious limitations to the number of CABs allowed in the array.

The limitation on the scalability of programmable analogue circuits was the main motivation for this thesis whose objectives are described in next section. To achieve this objectives, other research fields with similar properties were investigated, mainly the systems developed to implement neuromorphic circuits which are reviewed in the next chapter.

1.2 Statement of hypothesis

The main objectives of this thesis are:

1. to propose an alternative method for analogue communication between functional blocks in a low-to-medium resolution, large-scale programmable analogue arrays context and

2. to demonstrate that this method is able to perform a set of computations on analogue signals independently of or when combined with the configurable analogue blocks.

1.3 Thesis overview

The chapters of this thesis are as follows:

The next chapter presents a brief history of FPAAs and the current state of those systems, focusing on the communication methods used on them. The limitations of the current communication strategies are highlighted. Paradigms of neuromorphic systems leading to an alternative communication methods are therefore reviewed.

A novel method to interconnect the CABs of a programmable analogue array is then introduced in the third chapter. The heart of this new architecture lies on the appropriate choice of the coding scheme. A set of asynchronous pulse-based differential methods are also considered. A figure of merit is used to choose the most suitable option for the specifications.

The fourth chapter introduces computational operations that can be performed by the proposed communication method and are demonstrated with chip results.

The fifth chapter presents the design flow and parameters of the chosen coding scheme. It also describes CMOS circuits designed for the implementation of the method with chip results being shown.

Conclusions and possible future work are them discussed in the sixth and last chapter.

Chapter 2
Communication in programmable analogue arrays

2.1 Introduction

The first step taken in this work to achieve the goals stated in the introduction was a comprehensive bibliographic review of work done in the programmable analogue arrays field up to date. This review is synthesized in this chapter.

The general concept and reasons for using programmable analogue arrays are presented. Both commercial and academic programmable analogue arrays developed so far are revisited, categorised and analysed. A list of implementations is given with their main characteristics and differences.

The issue of the transmission of information both between the elements inside the array and with the external world is studied. Despite being the most conventional method to perform this communication, voltage or current signal representation using switch matrices for signal routing present some limitations. An alternative method based on timing rather than voltage or current is presented.

Using this alternative method, a novel programmable analogue array communication architecture is described.

2.2 Programmable analogue architectures

Analogue circuits are much less robust than digital ones, because of their greater sensitivity to noise, cross-coupling, process and temperature drifts among others. This characteristic makes more difficult to change parameters and functionality of analogue circuits without degrading the system performance. Therefore most of analogue circuits are the result of full-custom, application-specific designs — presenting no reconfiguration capabilities — to provide high-performance operations.

The greater flexibility offered by digital systems has made them very popular. Digital computers have replaced their analogue counterparts a long time ago and most of the signal processing is done in digital domain.

However analogue circuits have intrinsic advantageous characteristics comparing to digital circuits, in general being faster, smaller and less energy demanding [6, 7]. Naturally, these advantages lead engineers and researchers to a quest to develop analogue architectures that also provide the high degree of flexibility experienced by FPGAs, DSPs or digital microprocessors [2].

Similarly to FPGAs, there are a wide range of potential applications for programmable analogue systems, including low-power computing [8], remote sensing [9], rapid prototyping [10].

Although it was written in 1998, a good review of patents and academic and commercial circuits point is presented in [5]. In this work, the authors first described the general idea of the Field Programmable Analogue Array (FPAA), proposed some classifications and gave examples of implementations. Usually, the characteristics of programmable analogue architectures can be defined using different classifications. Some of these classifications are:

Programming capability - A system can be defined according to how many times it can be programmed. Different systems vary from being only programmable once, like fused or anti-fused architectures [11] and metal-mask programmable analogue arrays, to allowing several — possible infinite — reconfigurations, like switched-capacitor circuits [12] and neural networks [13].

Programming method - This classification defines how to change the system behaviour. These changing can be obtained by either direct programming, where the designer or designer tool has the detailed knowledge of the signal flow [4], or learning and adapting techniques, where the system is seen as a “black box”. Usually, the latter is achieved using genetic algorithms [14, 15] or neural networks [16, 17].

Structure flexibility - A system can also be characterized according to its versatility. Some architectures allow only their parameters to be programmable, like programmable automatic gain control (AGC) amplifiers and adaptive filters [18] whilst other systems allow the signal path be changed [19], by changing the interconnection of different circuits.

Granularity - The basic idea behind FPAAs is the use of basic units — usually known as

Configurable Analogue Blocks (CABs) — to implement a range of functions. In general, the complexity of these basic blocks is a trade-off between performance and flexibility. The block complexity can range from fine granularity, where the basic blocks are basic components, like transistors, resistors and capacitors [9], to coarse granularity, with more complex circuits like capacitively coupled current conveyors (C4) and vector-matrix multipliers [20].

Signal representation - Signals need some sort of physical representation to be processed by electronic circuits. Analogue circuits are defined as operating in one or more domains, usually voltage, current or charge domain. Voltage domain is the classical choice for analogue engineers, but current and charge techniques have their common applications, like power control and CCD image sensors, respectively. Recently, timing has been used as another possible representation. This representation and its benefits and drawbacks will be presented in detail in the next section.

Other FPAA characteristics can be used to define other classifications. For instance, signal timing characteristics employed, by using discrete or continuous time circuitry [21, 22]; design techniques implemented, for instance sub-threshold transistor operation [4] or fully-differential versus single-ended circuit structures; and signal routing approaches, e.g. global, hierarchical or local routing [13].

Since the 1990s lot of work has been done to develop analogue architectures with a functional philosophy similar to the digital FPGAs. This work resulted in various techniques and some commercial products. It is worth describing in brief some commercial products that appeared in the last two decades. Some of them were discontinued, like the first three examples, but some still are available:

- Totally Reconfigurable Analogue Circuit (TRAC®) IC family from Zetex Semiconductors Inc. - Fast Analogue Solutions branch of Zetex group introduced this technology about twenty years ago. This architecture operates in continuous mode and is essentially a collection of operational amplifiers configured as one of a set of predefined functions to process analogue signals [23]. For instance, TRAC020LH [24] is a chip with twenty CABs where each one can be configured to perform summation (of two signals), negation, logarithm compression, anti-log expansion, rectification, amplification, differentiation and integration (the last three operations performed using external components, like resistors and capacitors). Further operations are possible combining them, like RMS conversion, filtering and others. Every CAB input and output are available outside and internal routing is done connecting theses CABs terminals.

- Field Programmable System on a Chip (FIPSOC) from Sidsa: This system is a mixed mode programmable circuit originally aimed to general front end and data acquisition [25]. The chip contains twelve differential amplifiers combined in four input channels, four comparators, one analogue multiplexer, ADC/DAC blocks and a digital micro-controller and memory [26]. The system is configured by the micro-controller or the internal logic using the ADC and DAC blocks. Signal routing is performed by the analogue multiplexer.

- In-system programmable analogue circuit (ispPAC$^{®}$) family from Lattice Semiconductor Corp.^{TM}- These circuits are hierarchically built from basic cells (PACell^{TM}) grouped in functional modules (PACblock^{TM}) and using an Analogue Rooting Pool (ARP) to connect PACell and PACblocks inputs and outputs, DACs and the device pins [27]. Each product in the family was designed to one specific function and each one has its own PACells [28]. PACells of ispPAC$^{®}$10 — targeted to signal conditioning functions, like amplification and filtering — consist of four PCAblocks containing instrumentation and summing amplifiers and arrays of capacitors. Elements like comparators and polarity switches are added to the ispPAC10's PACells to include non-linear processing in ispPAC$^{®}$20 architecture. IspPAC$^{®}$30's PACells include multiplying DACs, which make it a FPMA.

- Mixed-Signal Programmable System-on-Chip (PsoC$^{®}$) from Cypress Semiconductor^{TM}- It is a family of architectures comprising both digital and analogue programmable blocks, with supporting circuits, as SRAMs, clock generators and micro-controllers (M8, 8051 or ARM). Each block of PSoC$^{®}$1 sub-family consists of one operational amplifier. Additional circuitry determines whether they will operate either in continuous-time or discrete-time fashion. Rather than providing universal connectivity and switch programming, each block presents multiplexers sourcing the amplifier inputs, resistor strings, capacitor terminals from its neighbour blocks outputs [29]. Block designs were optimized to support a few key functions such as a delta sigma modulator, a gain amplifier, digital-to-analogue converter (DAC), or differencing amplifier [30].

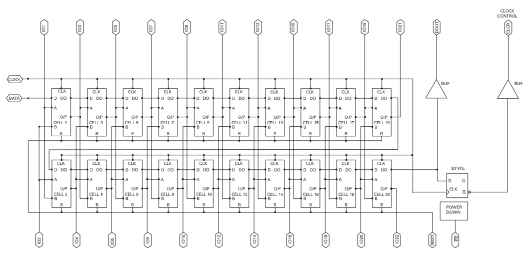
- Dynamically programmed Analogue Signal Processors (dpASP) and Field Programmable

Analogue Arrays (FPAA) from Anadigm® - The former Motorola's reconfigurable analogue group [31] designed new system consisting of a matrix of fully Configurable Analogue Blocks (CABs), surrounded by programmable interconnect resources and analogue inputs and output cells with active elements and supporting circuits. The first generation of these systems presented up to twenty CABs [32], but current products work with just four CABs [33] in order to present greater signal-to-noise ratio and bandwidth. Anadigm's products are based on switched-capacitor circuit techniques. Therefore, the core of their CABs is one operational amplifier and a programmable bank of capacitors and they are surrounded by a fabric of programmable interconnect resources, such as bandgap circuits, clock generators and lookup tables. These CABs are routed inside and to external pins using local and global switch-matrices.

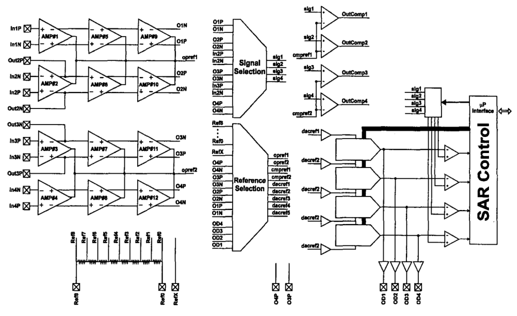
- Cellular Visual Microprocessor (CVM) from AnaLogic Computers Ltd. (Eutecus Inc. in USA) - This system is a visual sensor and processor based on Cellular Neural Networks (CNN) technique which is explained further in this section. For instance, ACE16k[1] is an 128x128 Focal-Plane Analogue Programmable Array Processor to work with high-speed and moderate accuracy (around 8bits) requirements [34]. Each ACE16k's CAB is built with several different circuits like analogue multipliers, non-linear dynamic blocks, analogue memory, optical detection circuits and others. Each CAB is analogue connected with other eight adjacent CABs and digitally connected with column ADC and DAC.

The previous list of commercial FPAAs are not exclusive, as there are other systems that incorporate some analogue array processing. Figure 2.1 shows diagrams of cited architectures whilst table 2.1 shows a comparison between the commercial products using the classifications presented before.

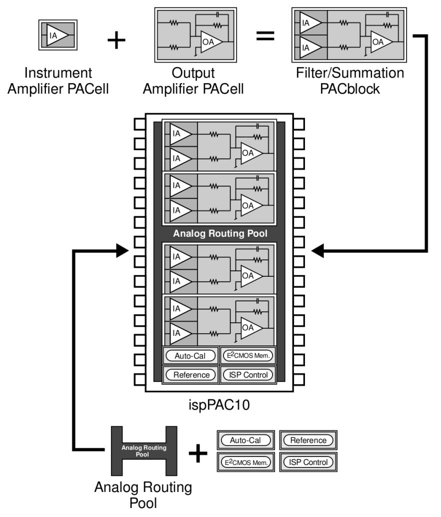
[1]ACE16k is the analogue core of the CVM system and was designed by Instituto de Microelectrónica de Sevilla (CNM-CSIC), Spain.



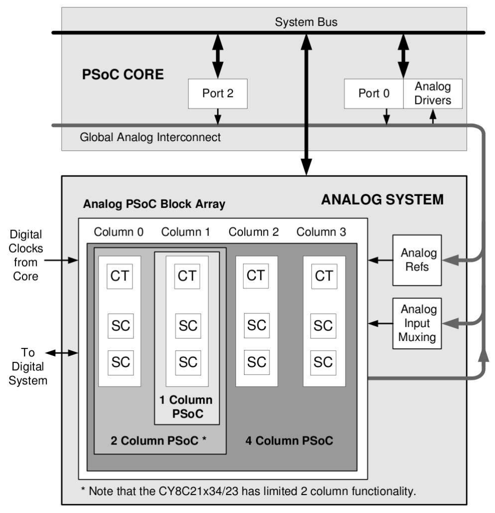
(a)



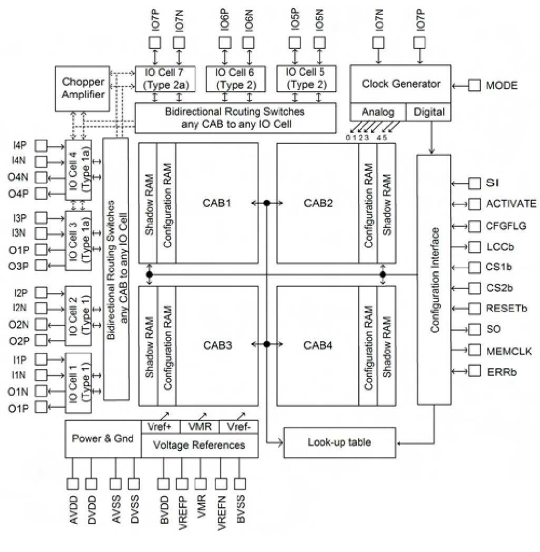
(b)



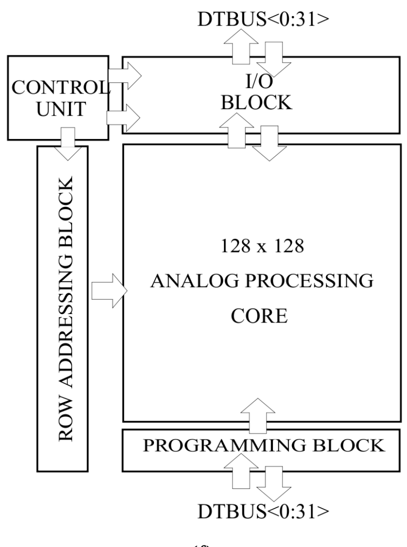
(c)



(d)



(e)



(f)

Figure 2.1: *Architecture diagrams of the commercial FPAAs cited in this work. (a) TRAC020LH [24] from Zetex, (b) FipSoc [26] from Sidsa, (c) ispPAC10 [27] from Lattice, (d) PSoC CY8C27x43 [29] from Cypress, (e) AN231E04 [33] from Anadigm and (f) ACE16k [34] from CNM-CSIC.*

Architecture	Type	Method	Flexibility	Granularity	Routing	Applications
Zetex's TRAC	ANA	Direct	PAR (using external components), FUN and ROUT	Medium (op-amp)	External	General signal processing. Able to implement functions like filtering, amplification, envelope and peak detection.
Sidsa's FIPSOC	MS	Direct	PAR and ROUT	Coarse (op-amp, DAC and ADC)	Internal multiplexer	General purpose front-end for signal conditioning and data acquisition.
Lattice's ispPAC	ANA/ MS	Direct	PAR, FUN and ROUT	Medium (op-amp)	Crossbar/ switch matrices	General signal processing with different architectures for specific applications, e.g. signal conditioning, control loop and monitoring and analogue front-ends.
Cypress's PsoC	MS	Direct	PAR, FUN and ROUT	Medium (op-amp, comparators)	Local multiplexers (PSoC1), multiplexers and crossbars (PSoC3)	General signal processing with different architectures for specific applications, e.g. touch screen sensors, LED Back-light, Motor Control, Power Management and Gyro Sensing.
Anadigm's dpASP	ANA/ MS	Direct	PAR, FUN and ROUT	Coarse (op-amp, comparator, SAR)	Switch matrices	General signal processing like signal conditioning, filtering and process control.
AnaLogic's CVM	ANA	Direct/ Adaptation	PAR and FUN	Coarse (multipliers, memories, photo-detectors)	Crossbar/ direct connection	Complex image processing implemented locally like terrain feature classification, multi-target tracking and optical flow calculation.

Table 2.1: *Characterization of commercial programmable analogue architectures. These systems are classified according to its signal processing type - either Analogue (ANA) or Mixed-Signal (MS); programming method; flexibility for configure CABs parameters (PAR), functions (FUN) and routing (ROUT); the granularity level; signals routing techniques and target applications. All architectures present infinite programmable capability and signals are represented in voltage-domain.*

In academia, research groups in these areas were identified with projects related to FPAA field:

- Electronic Systems Design Group at University of Southampton, where continuous time FPAA architectures were developed. This group was active until 2008. In [35], the group presented the continuous-time Hierarchical Field Programmable Analogue Array (HFPAA). This system presents a Differential Difference Amplifier (DDA) as the CAB. This is an example of mixed signal representation due its dual voltage/current output mode. The architecture uses hierarchical interconnection switches to achieve maximum routing capability between CABs and minimum number of routing resources.

- Cooperative Analogue and Digital Signal Processing (CADSP) at Georgia Technology Institute, where Dr. Paul Hasler's group continues the development of floating gate technique in FPAA [36, 37, 38] architectures and in the neuromorphic systems. By using floating gates as a switch for signal routing or even to reconfigure the function parameters, the group aims at very high-density field-programmable analogue arrays. Several CABs architectures can be implemented using this technique. For instance, in [20], two CABs were implemented, containing a mixture of fine-grained (MOSFETs and capacitors), medium-grained (OTAs), and coarse-grained (capacitively coupled current conveyors) computational blocks. Signals are routed using crossbars and switch-matrices using floating gate devices.

- NASA's JPL Evolvable Hardware Laboratory, where Dr. Adrian Stoica's group works on adaptive techniques for the use in analogue processing [9]. Their work objective is to develop a class of self-configurable and evolvable hardware, which adapts to its working environment to obtain optimal signal processing and provides fault tolerant functionality. In special the group has been working with Field Programmable Transistor Array (FPTA), where the CABs are fine-grained components like MOSFETs. In the Evolutionary Oriented Reconfigurable Architecture [39], CABs are made of 8 transistors, being 4 PMOS and 4 NMOS, and 24 switches. CABs are divided in clusters, where each cluster presents CABs with different transistor sizes. The routing are performed by switches and multiplexers. FPTA has also been the subject of other groups in University of Heldelberg [40], whose architecture is based on all-PMOS and all-NMOS CABs configurable in up to 75 different sizes in a checker-board pattern and routed using also switch-matrices and multiplexers. Recently, JPL group has designed another architecture — Self-Reconfigurable Analogue Array (SRAA) [41] — which presents a medium coarse granularity (OTA).

- Dr.-Ing. Joachim Becker from Department for Microelectronics at the University of Ulm. His Ph.D. studies at Institute of Microsystem Technology (IMTEK) with supervision of Prof. Yiannos Manoli resulted in an architecture based on digitally configurable transconductors. These continuous-time CABs presents a number (normally seven), binary weighted sizes G$_m$-cells that can be turned on or off. These cells are configured in a parallel fashion to obtain different transconductance. The CABs are also used to routing the signals inside the system and, therefore, avoiding the use of switches in the signal path [42]. A more recent implementation of this architecture use floating-gates to add current programmability and a 3-bit capacitor array [43].

- Prof. Leon O. Chua on Control, Robotics & Biosystems group at University of California Berkeley works on Cellular Neural (or Non-linear) Networks (CNN). The Universal Machine version of this architecture (CNN-UM) is based on the concept of specific connectivity model and analogue circuit dynamic with continuous valued state variables [44]. Signals are routed through direct, local interactions within a finite radius, however further cells can be "virtually" connected due to dynamic propagation. The research of CNN techniques and applications involved other groups and researchers like Dr. Támas Roska at Neural Computing Lab at Hungarian Academy of Sciences. This architecture is chiefly used to model physical phenomena, neuromorphic control and different applications like visual processing [45].

The diagrams for these architectures are shown in figure 2.2. All approaches mentioned so far face the problem of sensibility of analogue signal to interferences in different ways, either by careful layout routing [27], using specific circuit techniques [20] or trying to limit the scope of routing [13, 42]. This research differs from these approaches as we aim to use timing representation for convey analogue variables instead of traditional voltages and currents. The inspiration for this idea came from in a specific class of systems described next.

2.2.1 Neuromorphic systems as FPAAs

While general purpose FPAAs architectures have been continuously undermined by successive failures and struggle on producing viable and popular market solutions, at least one application field has provided a better prospectus: artificial neural networks with neuromorphic systems in particular.

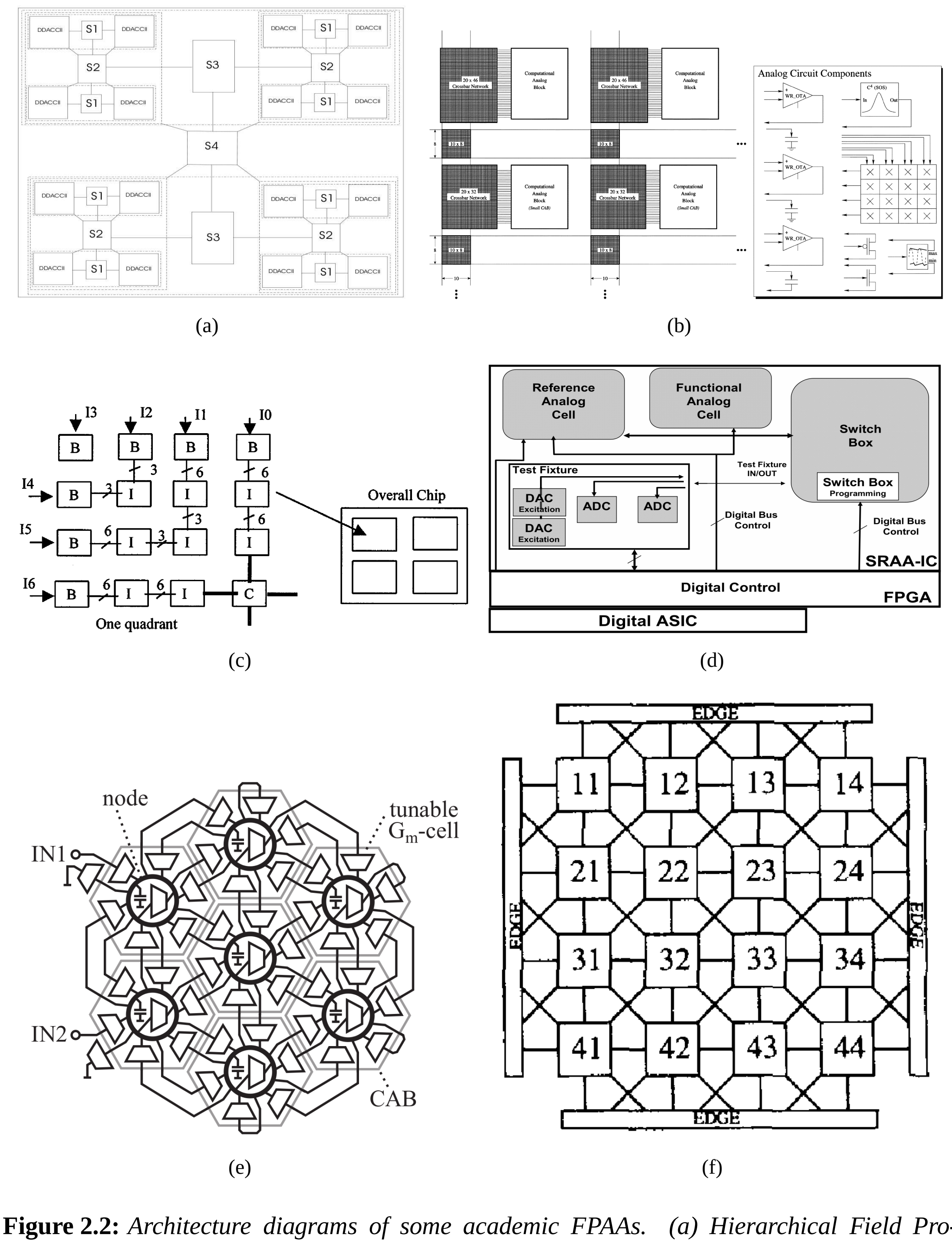


Figure 2.2: *Architecture diagrams of some academic FPAAs. (a) Hierarchical Field Programmable Analogue Array [46], (b) Large scale FPAA as in [20], (c) Field Programmable Transistor Array (FPTA) presented in [9], (d) Self-Reconfigurable Analog Array (SRAA) [41], (e) the G_m-C based Field Programmable Analogue Array with floating gate transistors [43] and (f) analogue programmable Cellular Neural Network (CNN) chip [45].*

Artificial Neural Networks (ANNs) are computational architectures where each element perform a similar computational role of biological neurons and synapses. The computation is highly associated with the interconnection between the elements in the network rather than with the elements functionality. Usually its configuration is adaptive and it is defined though the use of learning algorithms.

Although ANN can be implemented in software, its hardware implementations (HNN) lead to greater computational performance [47]. These hardware architectures can be implemented either using specific designed IC (ASIC) or a more generic platform. Examples of ASIC ICs include digital-based designs [48, 49], analogue [50, 51], mixed-mode [52] or even optical [53] designs. More recently, artificial neural networks designed on programmable platforms have appeared, both in digital [54], analogue [55, 16] and mixed-mode domains [17].

ANN computations are biologically inspired but they intend to replicate the computational functions of biological neurons rather than reproduce in detail their working principles. In other words, ANN aims to mimic the biological neurons computational principles but not their structures. For this goal the concept of neuromorphic systems was created. Neuromorphic systems are VLSI systems designed to mimic at least some functional and computational properties of the biological nervous systems [6].

Most of the neuromorphic systems are classified as Spike Neural Network (SNN), based on the concept that the information is transferred between neurons using temporal information conveyed on the onset of spikes [59]. Accurate representations of the neuron functionality may include complex and non-linear mathematical models like the Hodgkin and Huxley or Izhikevich [60] models of ions channels on the neuron membrane. As an example of VLSI implementation of such ion channels is the Field Programmable Neural Array (FPNA) [56]. It is an array of neurons (one per line) with sub-threshold circuits emulating dendrite sections (columns) and the soma. Regarding the connectivity, in this circuit external inputs and soma outputs are routed back to dendrite sections throughout cross-bars.

However most of the neuromorphic architectures are based on the simpler integrate-and-fire (I&F) representation of the neuron [61]. From this principle, very simple basic blocks were designed to represent neurons and synapses [62, 63, 64]. Although some recent approaches use digital architectures, traditionally these circuits were designed using analogue circuitry mainly [65, 57] due their smaller sizes.

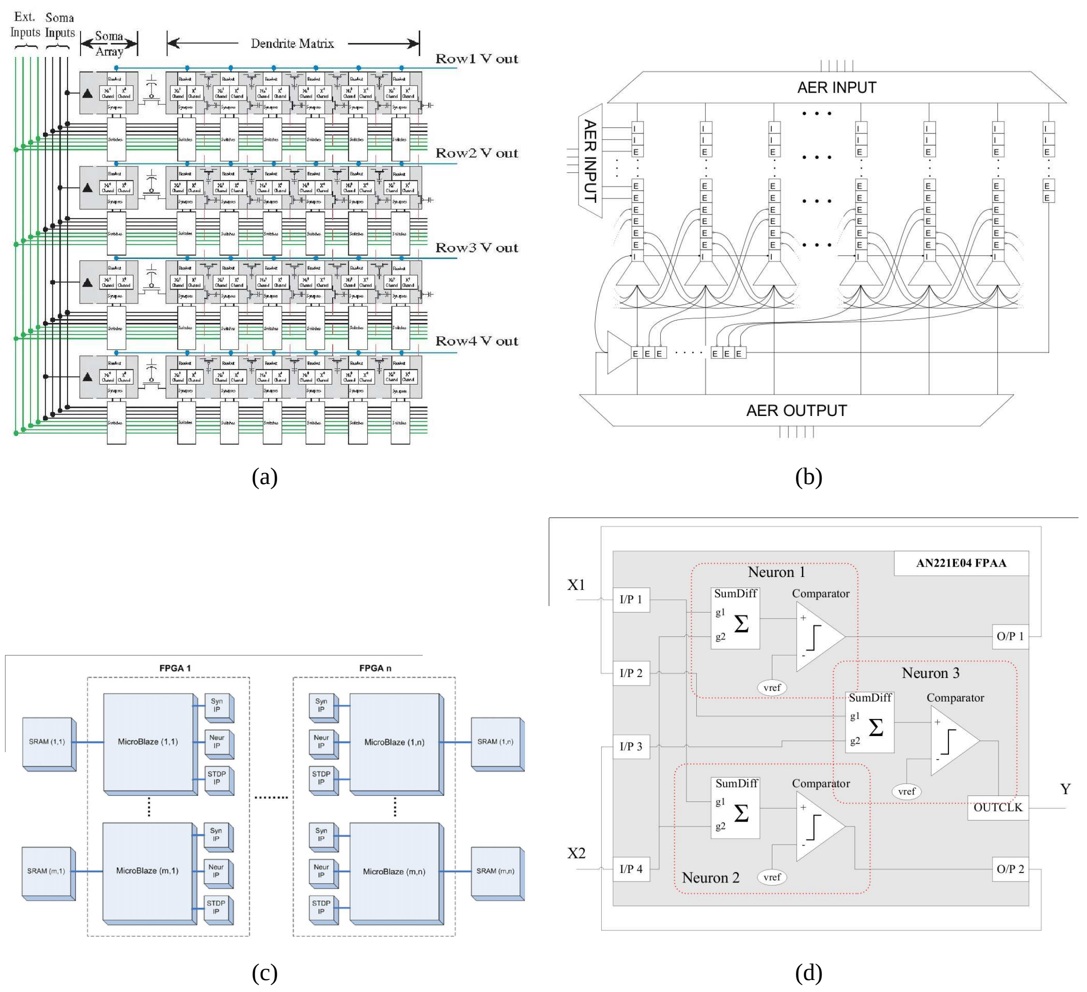


Figure 2.3: *Examples of neuromorphic architectures. (a) Field programmable neural array (FPNA) presented in [56] is an array of sub-threshold circuits to mimic ion channels presented in neuron membrane; (b) an event-based VLSI network of integrate-and-fire (I&F) neurons from [57] with neurons (trapezoids) and excitatory (E) and inhibitory (I) synapses; (c) a FPGA-based neuromorphic architecture as in [58] and (d) a neuromorphic system implemented into the AN221E04 FPAA from Anadigm Inc. presented in [55].*

Most neuromorphic systems are very specialised programmable arrays, where neurons, synapses and other functional blocks act as CABs. In [57] the system is built from 32 neurons with 22 synapses each. However these systems can also be implemented using generic programmable arrays, both digital — DSPs [66] and FPGAs [58, 67] — and analogue [68, 69]. The last two systems implement an I&F SNN and a pulsed coupled oscillator, respectively, into Anadigm's FPAAs. Due to limited hardware resources on these FPAAs, the I&F SNN system were implemented using multiple devices. Figure 2.3 shows some ANN and neuromorphic systems.

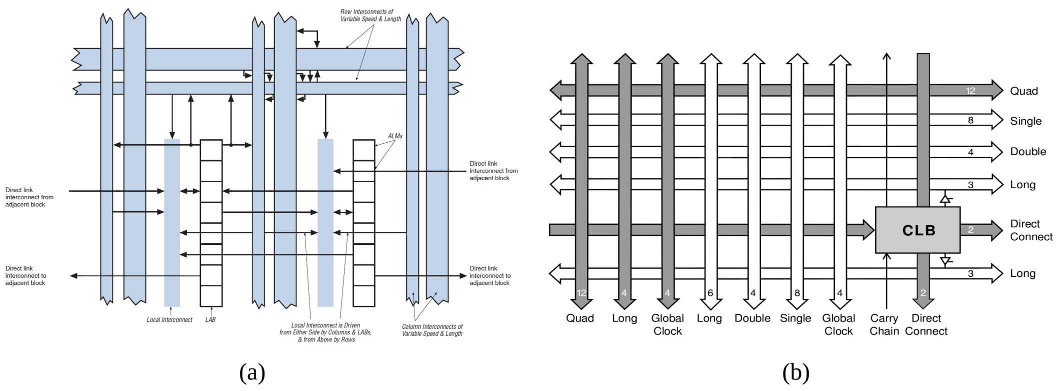

Row Interconnects of Variable Speed & Length
ALMs
Direct link interconnect from adjacent block
Direct link interconnect to adjacent block
Local Interconnect
LAB
Local Interconnect is Driven from Either Side by Columns & LABs, & from Above by Rows
Column Interconnects of Variable Speed & Length
(a)
Quad
Single
Double
Long
Direct Connect
Long
CLB
Quad
Long
Global Clock
Long
Double
Single
Global Clock
Carry Chain
Direct Connect
(b)

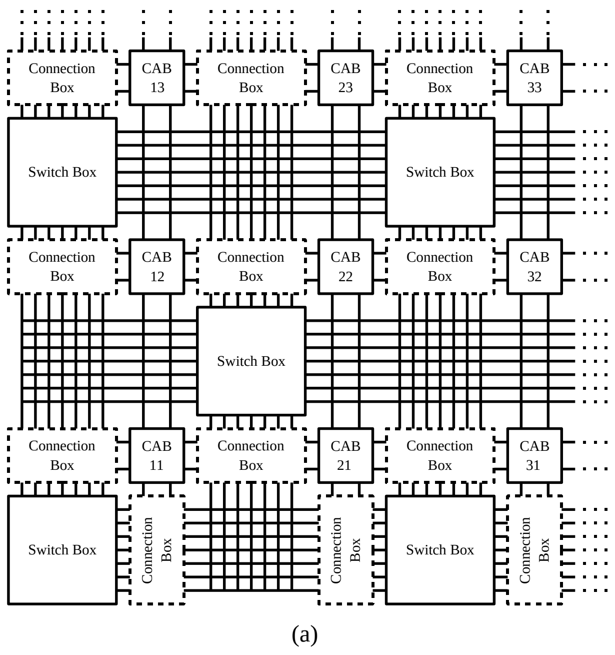
Figure 2.4: *Altera and Xilinx FPGA routing styles. (a) Altera's Stratix II architecture presented in [70] and (b) the Xilinx XC4000 family as presented in [71].*

Despite all these interesting and innovative “analogue programmable architectures”, the main contribution to this thesis from neuromorphic systems is the most common method (Address Event Representation — AER, described on next chapter) used to rout information among its elements, which differs from the conventional analogue routing methods used so far.

2.3 Analogue information routing

As said before, one of the main issues faced by programmable analogue systems is the communication of information between their computational blocks. The main characteristics of analogue routings are their degree of connectivity, signal integrity and power and area used by them. The ideal routing architecture would present a maximum degree of connectivity, connecting every signal of any CAB to any other CAB in the system. It would also convey the signals with no degradation, either from the routing itself or from other signals, and present the minimum area and power consumption overhead.

In FPGAs the signals are usually routed using switch-based routers and buffers with each vendor presenting different routing implementations. Xilinx implements island-style architectures with each logic block surrounded by connection blocks which connect them to different wires segments that end at switching blocks. The different wire segments allows for short, direct interconnection for speed optimization to long wires to large fan-out and clock signals. Local and global connections are used by Altera FPGAs as depicted in figure 2.4(a) whilst a Xilinx routing example is illustrated in figure 2.4(b).



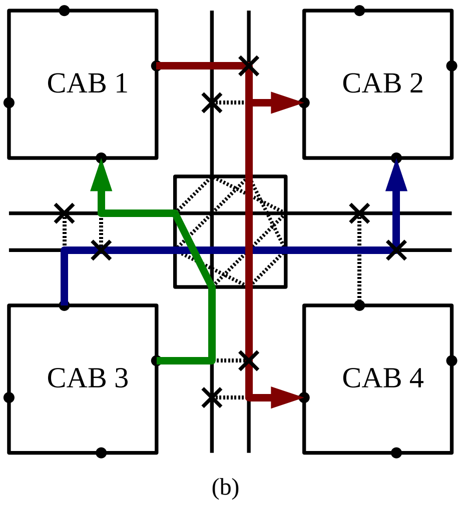


Figure 2.5: *FPAA routing example. (a) An example of a programmable analogue array as presented in [73] and (b) an example of CABs connection pattern.*

Similar approaches are used by most of the programmable analogue systems [72], with the figure 2.5 showing an example architecture and connection examples. Examples of commercial systems include architectures from Lattice, Cypress and Anadigm as shown in table 2.1. Academic examples that use switch-based routers are HFPAA [35] and floating-gate based FPAAs [20], among others.

Using switch matrices and crossbars leads to some design issues like connectivity and signal integrity. The degree of connectivity is related to the power and area usage[2]. The degree of connectivity (fanout) presented by these techniques are given by Rent's rule, after E.F. Rent's empirical work on sockets for digital computers to IBM [74] and can be applied even for brain connectivity [75]. In one interpretation of this rule [76], if a group of elements is arbitrary bounded, then it defines the number of links crossing this boundary:

$$P = P_0 N^b \tag{2.1}$$

with P being the number of links (wires), P_0 being the number of pins of each N elements

[2]In digital systems, timing is also affected by the degree of connectivity

inside the boundary and b is an empirical constant dependent of specificity and optimization of the system. For programmable gate arrays b is close to 1 [75] where different algorithms are employed to optimise number of switches and wire lengths.

Considering a chip with N elements, Sivilotti [76] defined β as the fanout of each element:

$$\beta = \frac{N_s}{{P_0}^2 N} \tag{2.2}$$

where N_s is the number of switches to connect these elements. In a non-hierarchical connection pattern with $P_0 N$ inputs and P outputs then applying equation 2.1:

$$N_s = P_0^2 N^{b+1} \tag{2.3}$$

which gives $\beta = N^b$. For full connectivity, each element needs to connect to any other element, giving $b = 1$. Therefore, the number of switches needed for a full connectivity using crossbars increases proportionally to the square of the number of elements. Hierarchical approaches can reduce this number [76].

The switch-based analogue routing is used by many analogue programmable arrays implementations. As crossbars are efficient only for small size architectures [75], current systems tend to present a small number of CABs [77].

Other issues appear with the use of switches for routing analogue signals and one is related to the linearity of switches. When implemented with MOS transistors, the switch resistance is heavily dependent of the signal amplitude, which leads to signal distortion. CMOS transmission gates and, more recently, floating-gate transistors can reduce but not eliminate this limitation [20], as shown in figure 2.6. However, floating-gate transistors require highly complex programming methods due the high voltage required for the generation of tunnelling effect.

In a programmable array a signal is normally connected to several other CABs, i.e. the signal has a high fanout. These CABs are located at different distances from the signal origin and, therefore, the connections present different dynamic responses. These responses are due to different capacitances and resistances (wire lengths and number of switches) of each connection. These differences results in destination-dependent signal distortion due to the different delays (phase shift) applied to the signal.

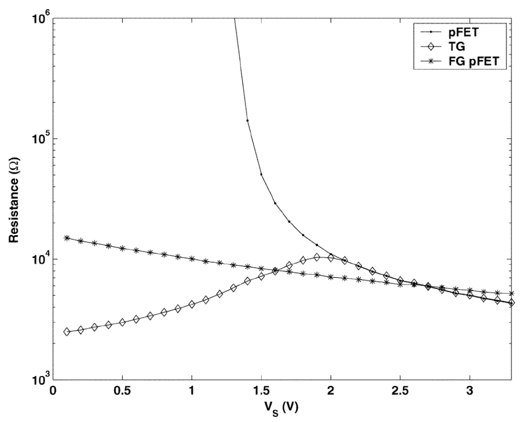


Figure 2.6: *Non-linear switch resistance. Typical MOS switch (pFET), transmission gate (TG) and floating-gate switch (FG pFET) resistance as a function of input signal amplitude (Vs), from [20]. This input amplitude dependence contribute to signal distortion on switch-based routing.*

In a switch-based routing, signals are transmitted using wires laid orthogonally. Capacitive coupling between these wires leads to another source of distortion known as inter-channel interference. Longer the routing path, higher is the probability of a signal suffer interference from other signals.

When a high fanout is required, CABs are designed with buffers for their output signals. As these buffers are designed to meet the worst case condition (highest fanout) their inclusion leads to high power consumption.

If crossbars are in one extreme point presenting full connectivity, in the other extreme are systems where each cell is allowed to connect only to its neighbours, like in CNN topologies [44] and in hexagonal FPAAs [42]. Whilst these approaches avoid most of the area overhead of routing-dedicated circuitry, they tend to use the CABs themselves as routing channels. Unless these systems target very specific applications, they tend to require a high level of CABs used exclusively for routing.

2.3.1 Timing communication

Analogue signals conveys information which using a limited range of continuous values changing in continuous or discrete times. The continuous characteristics of its amplitude, although allows for more compact information representation, make its storage and transmission less accurate due to presence of noise and distortion among others.

Modulation methods are used to increase the efficiency of such signals because they use a more robust carrier for transmission. Information is then coded by changing one or more characteristics of the carrier. Because this carrier is more easily detected and measured, its use can reduce information loss common to analogue signals.

Modulations are usually classified into digital, analogue and pulse modulations. In analogue modulations the carrier is also an analogue signal with well defined characteristics such a sine wave. The information is then coded into the carrier's amplitude (AM), frequency (FM) or phase (PM). Although it presents greater immunity then the original representation, these modulations are also susceptible to the same effects due its analogue characteristics. Digital modulations also use analogue carriers but coded with a digital representation according to the information to be transmitted. It requires analogue-to-digital conversions of the information before signal transmission, therefore losing the advantages of the analogue representation.

In contrast, the carrier in the pulse modulations is usually a signal with a limited number of states. The information is coded into the switching time or switching frequency of these states. Binary state representation presents a higher amplitude separation and, therefore, an easier distinction between them. This distance has the potential to offer good noise immunity [78] and, therefore, an easier routing in a programmable analogue array architecture when compared to the use of an analogue signal. Inaccuracies appears on the switch timing (jitter) rather than in the carrier amplitude.

Examples of pulse modulations are Pulse Frequency and Rate Modulation (PFM/PRM), Pulse Delay or Position Modulation (PDM), Pulse Width Modulation (PWM), Pulse Code Modulation (PCM) and Stochastic Pulse Modulation (SPM) and are shown in figure 2.7. A detailed review of these modulations is found in [79], where the author did an extensive analysis of the mentioned modulations regarding on accuracy, multiplexing and power dissipation, and in [80], where the modulations are considered in neuromorphic realm.

The idea of using timing as the key parameter in analogue processing is not new, and it is more clear if we consider PWM modulations. The PWM is used in D-class audio amplifiers, which are more energy-efficient than other power amplifiers classes, electric motor drive control and communications, for instance.

As an analogue signal representation, timing representation coding has been used by neuromorphic systems groups in their quest to mimic the biological neuron [81]. The coding is associated

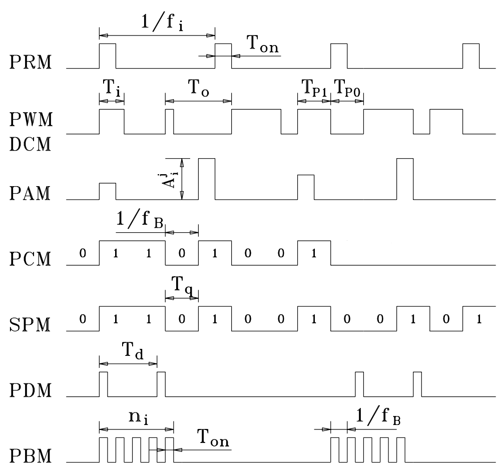


Figure 2.7: *Some pulse stream modulations. From the top: Pulse Rate Modulation (PRM), Pulse Width Modulation (PWM), Pulse Amplitude Modulation (PAM), Pulse Code Modulation (PCM), Stochastic Pulse Modulation (SPM), Pulse Delay Modulation (PDM) and Pulse Burst Modulation (PBM). Adapted from [79].*

with spike trains in this specific case. A great number of circuits modelling specific properties of the neurons have been published [82, 83], whilst others focus on its computational potentials [59]. Currently, these researches find applications on field of neuro-inspired circuits, like auditory [84], olfactory [85] and visual [86] systems.

Other two groups — the Hybrid Group at University of Florida (Dr. John G. Harris) and the Bionet Group at University of Columbia (Dr. Aurel A. Lazar) — have developed similar works also using spike coding. Their researches focus on mathematical foundations and implementations of time encoding machines using irregular sampling techniques [87, 88]. Both uses the integral characteristic of integrate-and-fire neurons models to perfectly reconstruct an analogue signal from a spike train. This reconstruction is based on finding the weights of a coefficient matrix.

By using timing representation, the signal dynamic range is increased. Voltage and current dynamic ranges tend to be reduced with the evolution of the CMOS fabrication technologies and system requirements, as low voltage supplies and low power consumption are required. On the other hand, timing dynamic range tends to increase as the IC fabrication technology delivers faster and smaller transistors.

One limitation of timing to express analogue variables is regarding to its continuity. Because of the analogue (voltage or current) to time conversion, there is always a period of time allowed

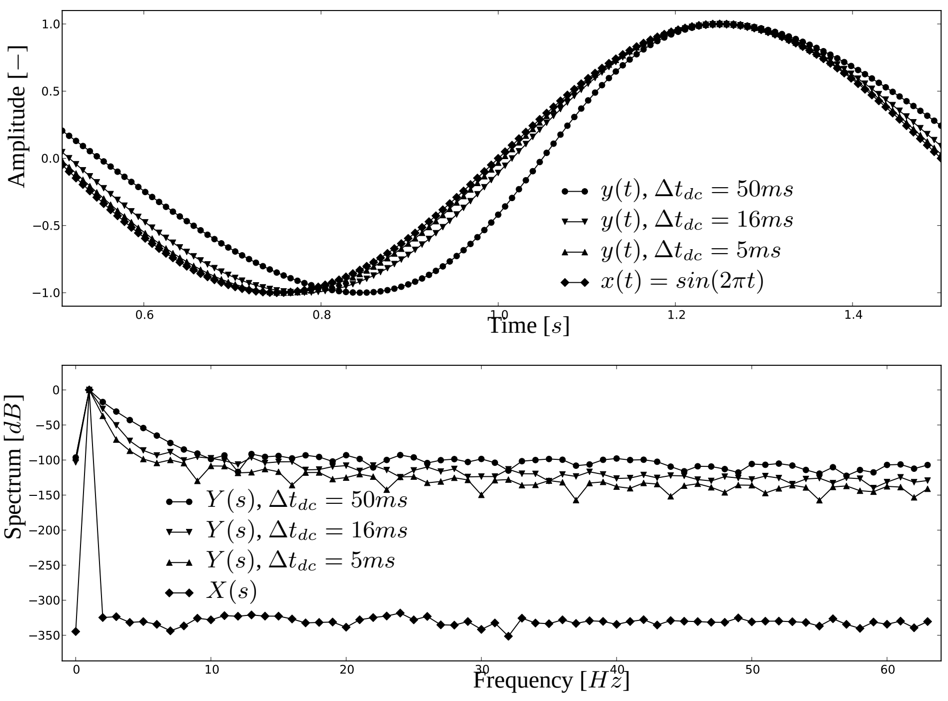


Figure 2.8: *PDM distortion. Example of distortion due to asynchronous pulse distance modulation (PDM). Each $y(t)$ is the modulated output for the sine wave $x(t)$ with different timing characteristics. Δt_{dc} represents the mean time distance of events for each case. A constant ratio between the timing dynamic range to mean time distance of 1.8 was used. In other words, for $y(t)$ modulated with $\Delta t_{dc} = 50ms$, the highest $x(t)$ value presents a distance of $50ms - 1.8 * 50ms/2 = 5.0ms$ whilst for the lowest value of $x(t)$ the distance is $50ms + 1.8 * 50ms/2 = 95.0ms$. As the output $y(t)$ is updated at different times, this output is a distorted version of the input $x(t)$.*

to represent the measured value. Therefore information cannot be measured continuously and therefore every timing modulation is time-sampled. This sampling time can be fixed — as in clocked systems — or variable, allowing another measurement after a previous one has finished.

Clocked versions — specially PWM-based ones — have been used for transmission of analogue information in arrays [21, 89]. However, the requirement of a global clock signal to synchronize the transmission leads to greater power consumption and issues such as clock skew, noise [90], metastability and high levels of electromagnetic interference (EMI) compared to variable-timed sampled (asynchronous) systems [91].

However continuously-update variable-timed sampled systems leads to signal distortion due to phase shift generated by different timing windows, as shown in figure 2.8 for PDM modulation.

A special class of timing modulations are known as differential modulations. The information transmitted with these methods is related to the previous measurements rather than the information itself. Because of this property these modulations need some type of memory to store previous information.

Delta-based modulations [92] are differential modulations where the conveyed information is restricted to represent an increase or decrease by a fixed small amount. In the simplest case, a simple bit is enough to represent these binary states. In their asynchronous versions of these modulations, each timing event does not convey signal amplitude information, but rather the instant when the signal amplitude has changed by a fixed value and the direction of this change. Some of these modulations are studied in more details in the next chapter.

2.4 Summary

In this chapter, a brief history of the analogue programmable arrays was presented. This type of circuits have been struggling to obtain the same level of maturity and popularity as the digital arrays. One of the reasons is the relatively small array size in the available analogue arrays and it is due mainly to the intercommunication between the array elements.

The problem of intercommunication between each block inside an array is an important issue in digital arrays but rather more problematic in its analogue counterpart. In digital domain the used area, power overhead, the degree of flexibility in the communication pattern and the delays inserted in the signal path are the main aspects to be considered in the communication design. In analogue arrays, these aspects are added to intrinsic analogue challenges of the signal distortion, white noise and interference.

The use of communication methods similar to the digital arrays present a great limitation on the practical size of the analogue array and power consumption. In the next chapter a time based alternative to overcome this essential limitation is proposed.

Chapter 3

Asynchronous spike event coding scheme: the communication method

3.1 Introduction

In the previous chapter a detailed review of the field of programmable analogue arrays were presented. The limitations of the current communication mechanisms were explained and a different approach using time as the information representation was suggested.

In this chapter a novel programmable analogue communication architecture is proposed. This architecture uses timing-encoded signals to convey information between the CABs within the array and outside the system.

Firstly, the communication method used for the proposed programmable architecture is presented and its functionality is explained. In this chapter the communication aspects of the method is studied whilst its computational properties are presented in chapter 4. Address Event Representation (AER) constitutes an important part of the method and therefore is presented in this chapter as well.

The core of such method, the timing coding, is presented. A specific timing coding scheme, the Ternary Spike Delta (TSD) modulation is used in this work. However other timing schemes are also able to be integrated into the architecture. A set of these alternative methods are presented in the following sections.

Finally, with different timing coding candidates being available, a method to evaluate the performance of each coding schemes is needed. The evaluation is performed by measuring its *Channel Efficiency* (CE) together with other constraints, mainly their computational properties. The outcome of such evaluation reveals that the TSD coding is theoretically the most suitable to be implemented.

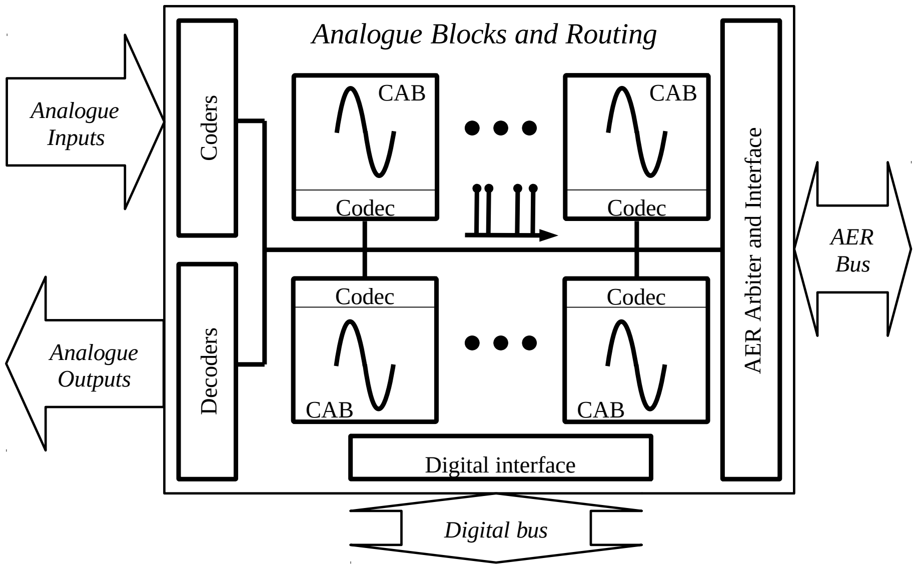


Figure 3.1: *Proposed communication architecture. The array of Configurable Analogue Blocks (CABs) is connected using an asynchronous digital channel. Asynchronous Spike Event Coding (ASEC) coders and decoders (codec) are also used to interface the array with external circuitry.*

3.2 Asynchronous spike event coding scheme

From the review of the previous and existing FPAA architectures, the inter CABs communication was identified as an important issue and one of the main limitations of FPAA performance. In this work a novel architecture is proposed based on the timing communication.

The architecture is designed to be flexible regarding the CAB's internal functionality or implementation as long their inputs and outputs signals respect the limitations of the communication scheme. The system is flexible enough to even allow a hybrid implementation of communication methods. For instance, a system can use conventional communication method of analogue switch matrices to transmit information inside a small-size cluster of CABs and the timing method used to communicate between clusters.

In the architecture shown in figure 3.1, CABs in an array are virtually interconnected using a common asynchronous digital channel. The communication between those CABs is the role of the Asynchronous Spike Event Coding (ASEC) communication scheme [93]. This scheme is the result of the *association* of the AER communication method with the TSD modulation which perform the conversion of analogue signals into timing information and vice-versa. The TSD modulation output is used to trigger AER communication signalling in the digital channel.

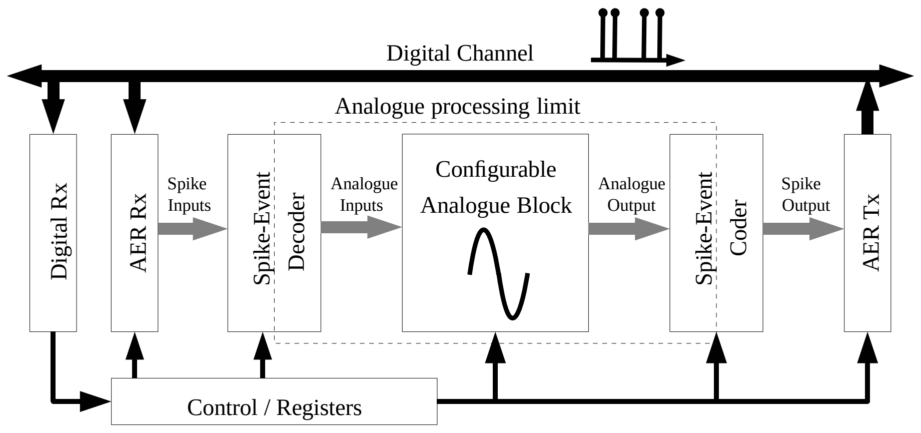


Figure 3.2: *ASEC-CAB interface diagram. Block diagram of the Asynchronous Spike Event Coding (ASEC) scheme interfacing with a CAB. With this architecture, analogue signals are limited to the CAB realm, reducing the levels of interference.*

ASEC conversion consists of one *Spike Event*[1] coder and decoder pair for each CAB as shown in figure 3.2. These coders and decoders work on the onset of specific events: whenever the CAB output analogue signal is found in certain conditions for the coders and on the common channel state for the decoders.

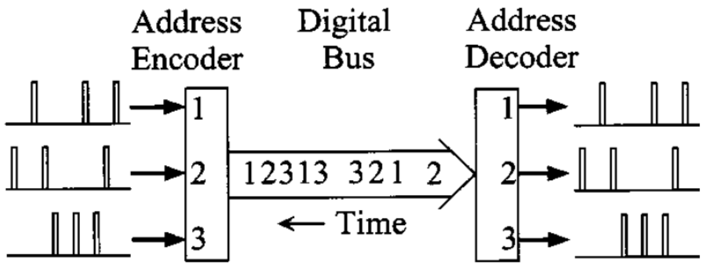
The transmission of these spike events is implemented using the common digital channel rather than dedicated interconnections. Because these events are asynchronous, the AER protocol — widely used in neuromorphic designs [94] — is an appropriate choice[2] for the management of information flux inside the array. This protocol has been used to convey analogue information in the past, as in [96].

3.3 The channel component: Address event representation

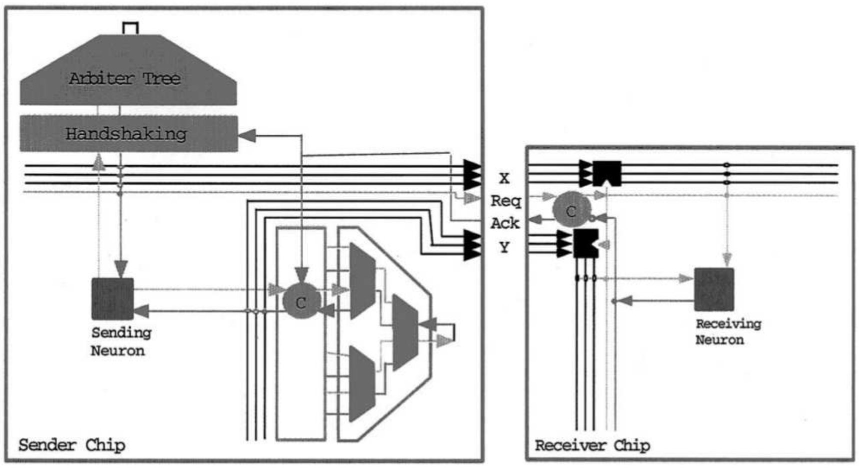
As said in the previous chapter, neuromorphic systems try to mimic structures and functionalities of biological neuronal systems into VLSI technology. Similar to biological neurons, most of such systems use spike representation for transmit information between their elements. However real neurons can be connected to hundreds or thousands of other neurons in a 3-D

[1]The "Spike" part of the name is derived from the neuromorphic systems, where the communication method was inspired. In these systems, as well as in actual neurons, an abrupt change of the a variable state (the membrane potential) encodes information in the instant it has happen rather than the change itself.

[2]In [95], Boahen performed a statistical evaluation of the AER protocol against other asynchronous protocols like ALOHA and CSMA. He concluded that arbitrated channels provide a communication throughput five fold than an non-arbitrated (ALOHA) channel.



(a)



(b)

Figure 3.3: *Address Event Representation. (a) Address Event Representation (AER) working principle and (b) conventional AER architecture from [95]. All events generated by the neurons (or CABs in our case) are coded according to the neuron address (an unique identification), multiplexed in time and then decoded at the receiving side. The conventional architecture comprises of horizontal and vertical arbiter trees and handshaking signalling circuitry.*

configuration whilst individual elements in analogue VLSI systems are connected in a 2-D silicon IC. A solution to overcome this limitation is to create "virtual connections" between those elements. An asynchronous communication system which implement these connections is the *Address Event Representation* (AER).

Original AER is a point-to-point asynchronous handshaking protocol for transmission of digital words using a Multiple Access Channel (MAC) common to every element in the array. The information coded in these transmitted digital words represents the identification (address) of either the transmitting or receiving CAB, depending on the implementation. The former case will be used in the following description.

The communication initiates whenever an element (a CAB in a FPAA) in the array generates an event. This element requests a permission to access the digital bus. If granted by the arbiter, its address is written on the digital bus, broadcasting the event to other elements in the array. An AER router is responsible for distributing these events to the appropriate receiver using internal or external LUTs, for instance. In this work, we use an external FPGA to route the spikes between coders and decoders. After the target element had acknowledged the reception of the event, the bus is freed to further utilization.

The asynchronous nature of the AER protocol greatly preserves the information conveyed in the time difference between events. The main source of inaccuracy between the generated and the received time difference is found in the presence of event collisions. Because the access to the channel is asynchronous and random, different elements may try to access the channel simultaneously. The AER protocol offers mechanisms to handle these spike collisions. Usually, a unique and central arbiter is used to manage collisions. In this case, collisions are resolved by an arbiter by queueing and transmitting successively all the spike events involved in a collision. Although this process can be made relatively faster than the signal, it is the main source of distortion due to the communication channel.

The use of AER protocol allows random time-sharing of the same physical channel between multiple CABs thereby avoiding an exponential increase in the number of physical connections — a limiting factor in the realization of large programmable analogue arrays. Our system has an interconnect complexity of O($\sqrt{N}$) [97], whereas analogue interconnect using switch matrices have a complexity of O(N^2). The area used to implement crossbars and switch matrices increases as O(N^2) as well, whilst our architecture has a proportional increase in circuit area, increasing as O(N). In other words, this architecture is more suitable to larger arrays.

Spike events are essentially asynchronous and robust digital signals that are easy to route on shared channels, not only between CABs, but also between ICs providing improved scalability. When extended to inter-chip communications the interconnect complexity is O($\log_2 N$).

Since its introduction in Mahowald's work [65] different versions of AER protocol have been proposed to overcome some limitations. Boahen's group have developed a serial version to improve its scalability [98] as well John Lazzaro in [99]. Andreou worked on increasing its speed [100] whilst Brajovic [101] presented an error correction codes method to avoid the use of arbiters. Appendix C presents a novel implementation with a distributed arbitration.

3.4 The coding component: Ternary spike delta modulation

The coding scheme used in this work is based on a derivation of the delta modulation. Delta modulation and its derivatives have been receiving multiple denominations in the time. The derivation used in this thesis is named Ternary Spike Delta (TSD) modulation. Being a derivation of the delta modulation, general aspects of the delta modulation will be analysed first. It will be useful also to describe other methods evaluated in this chapter. Because these other methods can also be linked with the AER communication, they are called spike delta modulations in this work.

The working principle of delta modulations is based on limiting the error $e(t)$ between the input signal $x(t)$ and an internal variable $z(t)$:

$$|e(t)| = |x(t) - z(t)| \leq e(t)_{max} \tag{3.1}$$

using a negative feedback loop in the modulator or *coder*. In other words, these modulations work by forcing a feedback signal $z(t)$ to track the input signal.

The error is sensed by a (normally 1-bit) quantizer system in order to produce the coder output $y(t)$ whilst the internal variable $z(t)$ is generated by the integration this output signal:

$$z(t) = k_i \int y(t)dt \tag{3.2}$$

where k_i is the integration gain.

The communication process is completed at the demodulator or *decoder* side, where the signal $z(t)$ is replicated as $z_R(t)$

$$z_R(t) = k_i \int y_R(t)dt \tag{3.3}$$

where $y_R(t)$ are the spikes at the decoder input.

In order to obtain this replication, an ideal channel *CH*[3] is considered, with $y_R(t) = y(t)$.

[3]The main characteristic of an ideal channel for this thesis is the fact that the spike time intervals from the input to the output of the channel are the same. In other words, the channel does not change the time elapsed between spikes generated by the coder.

Furthermore, both coder and decoder integrators are required to present the same initial condition and gain. Otherwise, $z_R(t)$ and $z(t)$ will present different DC levels and amplitudes, respectively.

Finally, a better approximation $x_R(t)$ of the input signal is obtained by averaging the reconstructed signal $z_R(t)$. A low-pass filter (LPF), whose impulse response function is $h(t)$, can be used for this purpose:

$$x_R(t) = h(z_R(t)) \approx \overline{z_R(t)} = \overline{x(t) - e(t)}. \tag{3.4}$$

Equations 3.1 and 3.4 show that the difference between the reconstructed signal and the input signal is bounded by the maximum allowed error $e(t)_{max}$.

These methods are known as delta modulations because the feedback control updates the feedback signal by a fixed amount *delta* (δ), which is a function of the resolution of the converter.

If the system is designed to provide N_b bits of resolution, then:

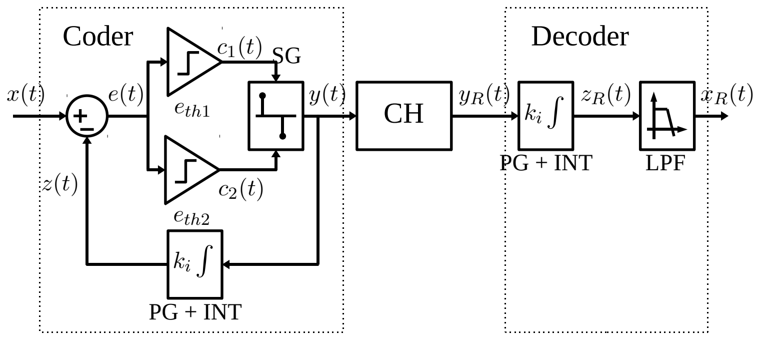
$$e(t)_{max} = \delta = \frac{\Delta x_{max}}{2^{N_b}} \tag{3.5}$$

where $\Delta x_{max} = x_{max} - x_{min}$ is the maximum amplitude variation of the input signal. The parameter δ, known as tracking or quantization step, is used in quantizer and integrators designs to limit the error to its maximum.

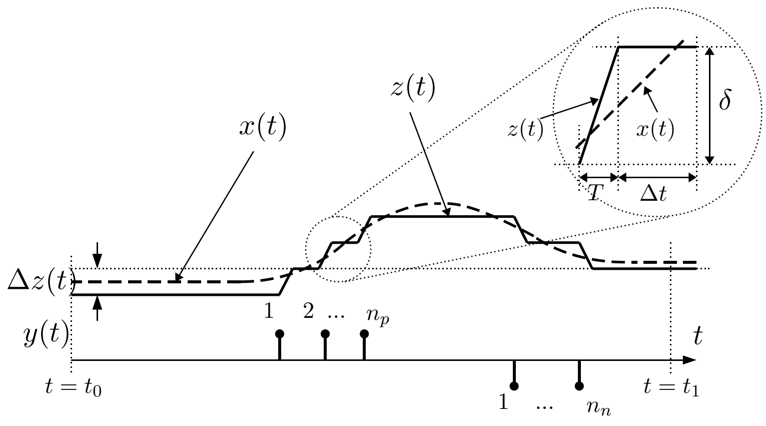
In the PWM version the quantized output is the output of the coder, i.e. $y(t) = Q(e(t))$. However, in spike-based versions the output of the quantizer(s) trigger(s) a spike generator (SG) such as $y(t) = SG(Q(e(t)))$.

Ternary Spike Delta (TSD) modulation was presented by [102] and further analysed by [103][4]. This modulation is similar to the schemes described in [104], [105], and [106] which are based on the principle of irregular sampling used to implement asynchronous A/D converters, for instance. A similar version was also used in [107] and by Miskowicz (re-branded as Send-on Delta) [108] for low-bandwidth, low-power sensors.

[4]Although the scheme studied in these works are PPM-based, they were named Asynchronous Delta Modulation (ADM).



(a)



(b)

Figure 3.4: *Ternary Spike Delta modulation - TSD. (a) is the block diagram of coder and decoder. It presents two comparators with different thresholds (e_{th1} and e_{th2}), a Spike Generator (SG), Pulse Generators (PG), integrators (INT), the channel (CH) and the decoder Low-Pass Filter (LPF). Signal $x(t)$ is the coder input signal while the feedback signal $z(t)$ is an approximated copy of $x(t)$ generated from the spike output $y(t)$. Signals $z_R(t)$ and $y_R(t)$ are, ideally, copy of signals $z(t)$ and $y(t)$ respectively on decoder side. Decoder output $x_R(t)$ is an approximation of $x(t)$. (b) presents illustrative waveforms of important signals. The spike train $y(t)$ presents three states: up, down and null. T is the pulse period, Δt is the inter-spike period and δ is the tracking step.*

In ternary modulations, the output present three possible states [102, 109]. These are used to indicate the spike onset and its "signal". Therefore, the signal transmitted is represented by "positive" and "negative" spikes and each resulting in an increment or decrement of δ, respectively. As the decrement in $z(t)$ is also controlled by spikes, the absence of spikes in this modulation indicates no change in the current value of $z(t)$.

Because of this three state output, it is not possible to use only one 1-bit quantizer (comparator). Another comparator is inserted in the coder loop and the figure 3.4(a) presents the coder block diagram of TSD modulation. This extra comparator also senses the error signal $e(t)$ but its comparison threshold e_{th2} is different[5]. The comparator outputs are combined within the SG block to generate the proper spikes $y(t)$, if needed. These spikes are outputted to the channel CH and to the Pulse Generator (PG) block. The respective pulse is then sourced to the integrator INT to produce the proper increase or decrease in $z(t)$ and $z_R(t)$.

The difference of thresholds Δe_{th} impacts the performance of the coder and its ideal value is the same as the tracking step, i.e., $\Delta e_{th} = \delta$. This ideal value is used in the following analysis whilst deviations from this will be considered in the chapter 5.

Although the integrator can present different gains for positive and negative spikes, only the case where the gains are symmetric is considered. For the following analysis the figure 3.4(b) is used as an example. The function of the integrator is to update $z(t)$ by a fixed amount and, therefore, it works as an analogue accumulator.

After analysing the figure 3.4(b), one can verify that the variation in the feedback signal $z(t)$ since the initial time t_0 is:

$$\Delta z(t) = \delta(n_p - n_n) \tag{3.6}$$

where n_p is the number of positive spikes and n_n of negative ones generated since t_0. According to equation 3.6, the value of the feedback signal $z(t)$ in TSD depends only of the number of positive and negative spikes since t_0, but not of the actual time t.

Although the TSD was adopted as the analogue coding method to the ASEC scheme, other five methods were also considered throughout this thesis. This methods are also presented in this chapter to be evaluated against the TSD. The next two methods are also based on delta modulation whilst the others are based on sigma-delta modulations.

[5]For the sake of simplicity, in this study it was considered that e_{th1} and e_{th2} are symmetric.

3.5 Alternative modulations

3.5.1 Binary spike delta modulation

Variations on the delta modulation model can be achieved by selecting the number of output states. In this thesis, the options for the amount of output states were limited to two (*binary*) and three (*ternary*) as in the TSD case described before. Although others are possible and are derivations of these two basic types, they increase the complexity of the communication scheme.

In Binary Spike Delta (BSD) modulation [103], the output is represented by only two states: either there is a spike or there are no spikes at all. For every spike generated, the feedback signal $z(t)$ increases by a fixed amount[6] δ. On the other hand, when there are no spikes, $z(t)$ decreases by a constant rate α_d. The diagram of the BSD modulation is showed in figure 3.5(a).

In this case the error signal $e(t)$ is compared against a fixed threshold e_{th} using only one comparator[7]. Therefore, the comparator output $c(t)$ is high when the $e(t) - e_{th} > 0$ and low otherwise. Every time $c(t)$ turns high, a spike is generated by the spike generator block (SG) as shown in figure 3.5(b), which is a detailed illustration of a possible waveform.

Being a delta modulation, the signal $z(t)$ in BSD method is generated by the feedback integrator and its variation from an initial time t_0 is

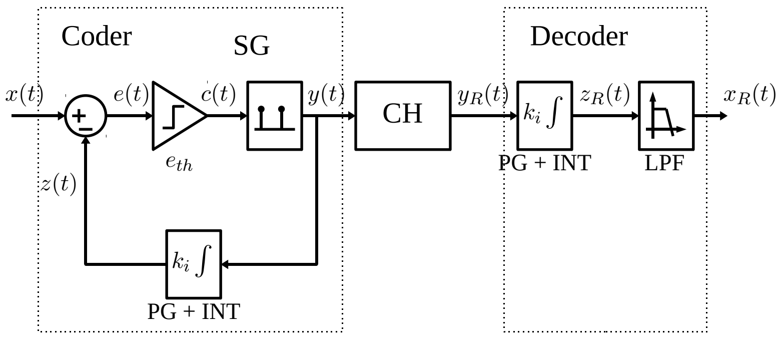
$$\begin{aligned}\Delta z(t) &= \Delta z(t_{n-1}) + \delta - k_i(t - t_{n-1}) = \cdots \\ &= n\delta - k_i(t - t_0) \end{aligned} \tag{3.7}$$

where n is the number of spikes generated since t_0. That is, the feedback signal is a function of the number of spikes and the time elapsed since the initial time t_0.

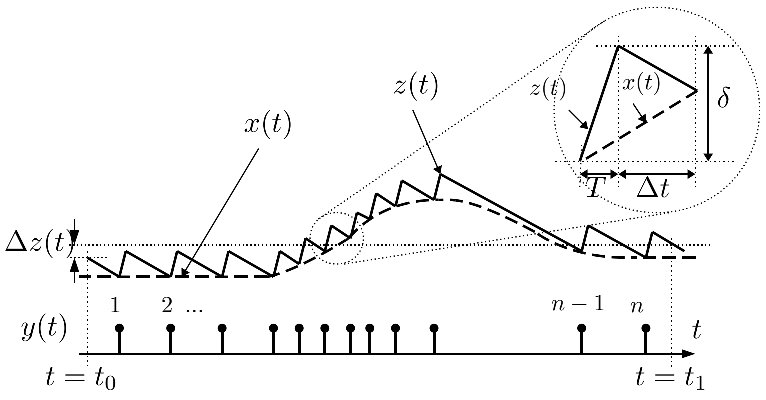
Different processes can be used to produce this dynamic in the integrator output. One possible method is to generate a fixed width (T) pulse from each output spike to create a fixed increment on $z(t)$, using the Spike Generator (SG) in figure 3.5(a). Also this spike generator would provide an appropriate DC level (p_{dc}) to decrement $z(t)$ proportionally (α_d) to the time between

[6]Other implementations can use the symmetrical process, i.e. $z(t)$ decreases by δ where there are spikes and increase when there is not. However the analysis is similar to the one presented.

[7]The choice of comparator threshold e_{th} impacts the average (DC) error $e(t)$ and should be designed to be $\delta/2$ to cancel this DC error. Figure 3.5(b) shows the effect for $e_{th} = 0$: a constant and positive DC shift of $\delta/2$.



(a)



(b)

Figure 3.5: *Binary Spike Delta modulation - BSD. (a) presents the BSD functional block diagram. Signals descriptions are similar to the ones in figure3.4(a). (b) shows example waveforms of the relevant signals. Similarly to TSD, there may be a DC offset between $x(t)$ and $z(t)$. In this case the DC offset is approximately $\delta/2$.*

spikes. In this case

$$k_i = \frac{\delta}{p_s T} = \frac{\alpha_d}{p_{dc}} \tag{3.8}$$

where p_s is the pulse amplitude triggered by the spike $y(t)$. As the variation of signal $z(t)$ during the period Δt is controlled by the pulse DC level, then

$$p_{dc} = \frac{\Delta z(t)}{k_i \Delta t}. \tag{3.9}$$

In order to obtain the minimum number of spikes, the following condition is needed:

$$\alpha_d = -\frac{|\Delta z(t)|_{max}}{\Delta t_{max}} \approx -\left|\dot{x}(t)\right|_{max} = -\alpha_s. \tag{3.10}$$

From equations 3.8 to 3.10, the DC level is

$$p_{dc} = -\frac{T\alpha_s}{\delta} p_s. \tag{3.11}$$

A PWM-like version can also be implemented using spikes and this implementation is described next.

3.5.2 Asynchronous delta modulation

Asynchronous Delta Modulation (ADM) was presented by [103], where it was referred as *Asynchronous PLM*, and [110], presented as *hybrid PLM-FM or rectangular-wave modulation*.

This modulation presents a two-state output pulse, with variable width and frequency. This bipolar representation is preferably used because it is adequate to digital channels. When used in a spike-based communication system, each spike can specify the time of each coder output $y(t)$ switch.

The block diagram of this modulation is presented in figure 3.6(a). The comparator output $c(t)$ controls SG and PGs converts each spike into a PWM ($\pm$b) signal representation to apply to both integrators.

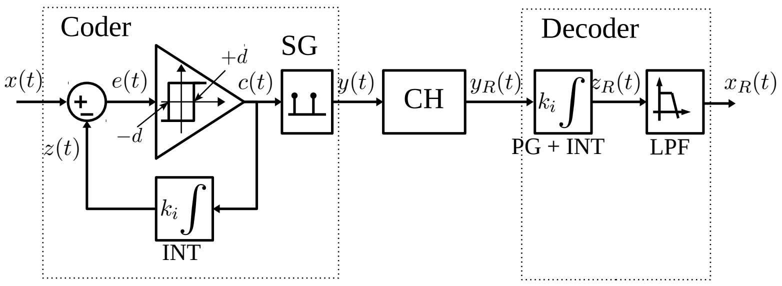
The variation on the integrator $z(t)$ output since the initial time t_0 is

$$\Delta z(t) = \sum_{i=1}^{n} (-1)^{i+1} \alpha_d (t_i - t_{i-1}) \tag{3.12}$$

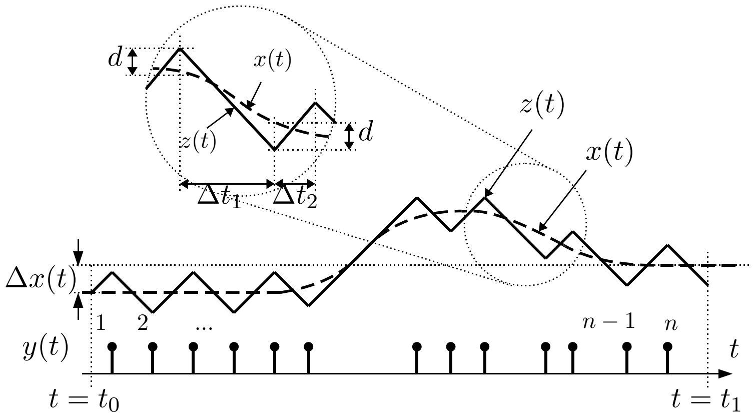
with α_d being the $z(t)$ update rate. Figure 3.6(b) shows an example of the behaviour of some signals of the modulation.

The coder uses a *hysteresis* comparator as the amplitude-to-time converter. In fact the comparator limits the error using its hysteresis window. Considering a comparator with a symmetrical hysteresis window, the error is limited by $-d \leq e(t) \leq d$ where $-d$ and d define the comparator hysteresis. Whenever the error reaches the limits of the window, a spike is generated and the feedback loop acts bringing the error signal inside the window. The comparator hysteresis is designed as

$$d = \delta/2. \tag{3.13}$$



(a)



(b)

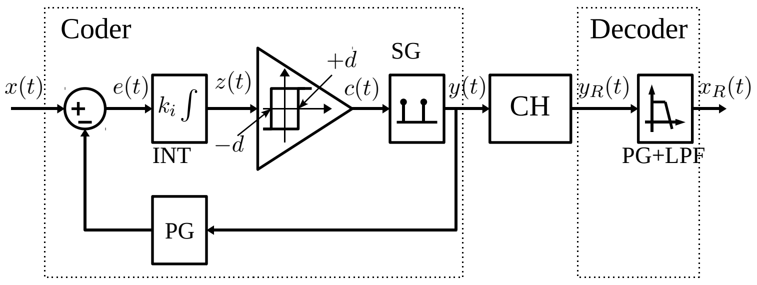
Figure 3.6: *Asynchronous Delta Modulation - ADM. (a) is the block diagram of the ADM modulation. As for BSD and TSD cases, Spike Generators (SG) are used to generate the pulses used as the input of integrators. In (b), Δt_1 and Δt_2 are the spike time intervals which defines the low and high periods for the Pulse Generator (PG) at the input of the integrators.*

For the correct working of the coder, the maximum speed which the signal $z(t)$ increases or decreases has to be greater than the speed of the signal. If α_s is the absolute value of maximum derivative of the input signal, i.e., $\alpha_s = |\dot{x}(t)|_{max} \leq \alpha_d$, then the integrator gain must be

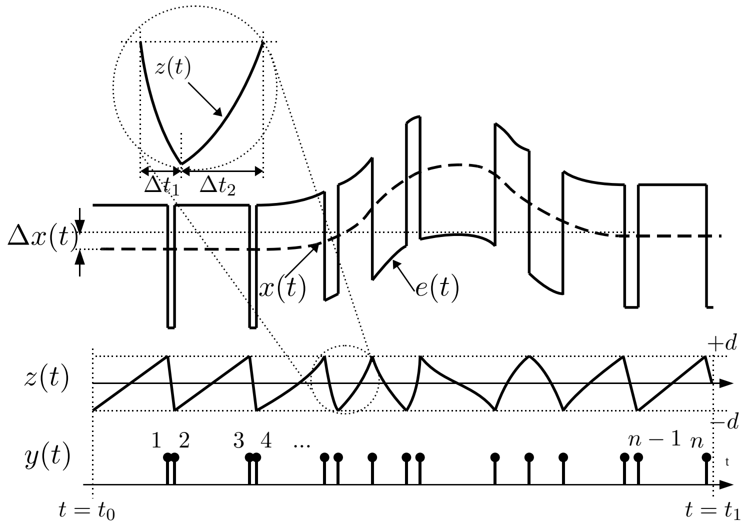
$$k_i \geq \frac{\alpha_s}{b}. \tag{3.14}$$

If this condition is not satisfied, the signal $z(t)$ will not able to track $x(t)$ all the times. This effect, common to every delta-based modulation, is known as slope overload [111].

Up to this point only spike versions of the PWM delta modulations have been discussed. These modulations are prone to slope overload, i.e., the feedback signal $z(t)$ may not be able to track the input signal $x(t)$ if this signal rises or fall very quickly. In other words, the coder works



(a)



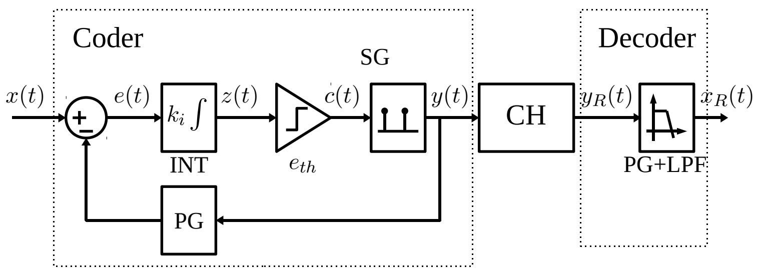
(b)

Figure 3.7: *Asynchronous Sigma Delta modulation - ASDM. (a) Block diagram and (b) example waveform with the important signals. Signal $z(t)$ tracks the input signal $x(t)$. The spike train $y(t)$ is used to replicate the tracking signal $z_R(t)$. The output signal $x_R(t)$ is a filtered version of $x(t)$.*

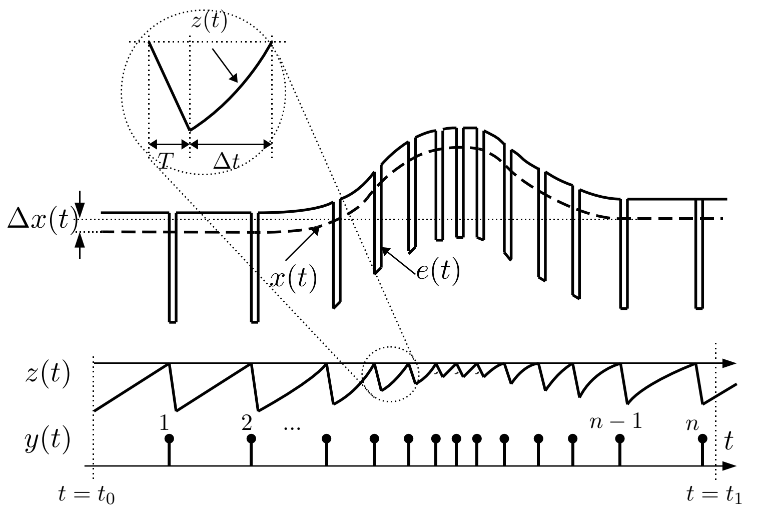
well up to a maximum input bandwidth. Alternative methods to avoid this effect are based on sigma-delta modulations and are presented in the next subsection.

3.5.3 Spike sigma-delta modulations

All methods evaluated in this chapter are spike differential modulations. These modulations present two different types of feedback control loop in the coding and transmission mechanism. Therefore they are classified according to the type of feedback loop. In delta modulations the integrator is in the signal feedback path whilst the sigma-delta modulations present the integrator on the forward path, before the quantizer.



(a)

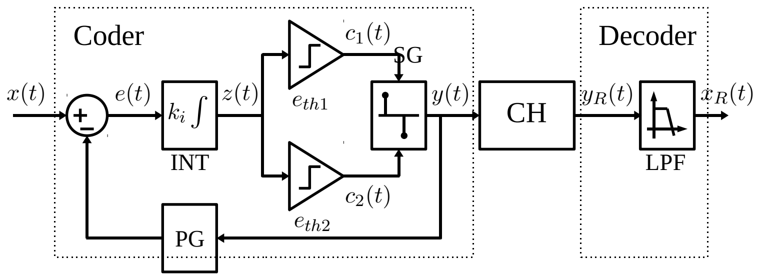


(b)

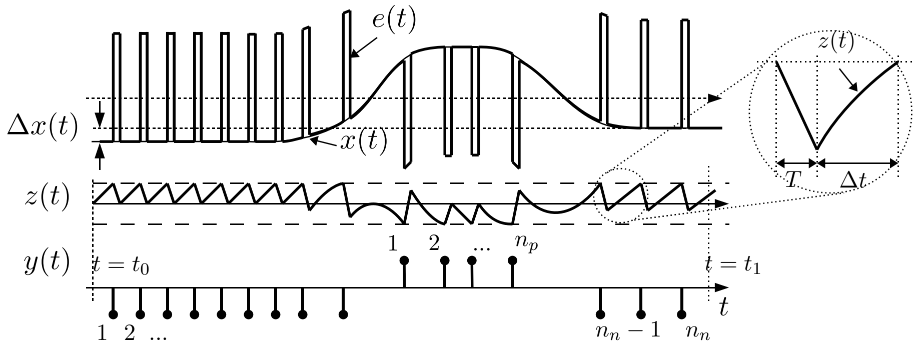
Figure 3.8: *Binary Spike Sigma Delta modulation - BSSD. (a) BSSD block diagram and (b) an example waveform. Signal $z(t)$ tracks the input signal $x(t)$. The spike train $y(t)$ is used to replicate the tracking signal $z_R(t)$. The output signal $x_R(t)$ is a filtered version of $x(t)$.*

The slope overload present in delta modulations can be minimized by increasing the tracking step or reducing the loop delay. The first impacts the system resolution and the second is limited by the technology used. Other methods include change the topology as in adaptive delta modulations (Continuously-Variable Slope Delta modulation [112], for instance) with requires a great complexity, or using sigma-delta modulations.

Sigma delta modulation is a variation of the delta modulation, where a block diagram reduction is performed in such way that only one integrator is used, by merging coder and decoder integrator and moving it to another position in the loop [92]. This removes the problem of mismatch between integrators found in delta modulations. In this new position of the integrator in the feedback loop, the quantizer codes the *integral* of this error instead of the error signal.



(a)



(b)

Figure 3.9: *Ternary Spike Sigma Delta modulation - TSSD. (a) Block diagram with (b) an illustrative waveform. Signal $z(t)$ tracks the input signal $x(t)$. The spike train $y(t)$ is used to replicate the tracking signal $z_R(t)$. The output signal $x_R(t)$ is a filtered version of $x(t)$.*

The main advantage of this modulation is known as noise shaping. Noise shaping is the processing of moving the quantization error to high frequencies. Therefore, the low pass filter at the decoder removes this high frequency components, resulting in a better Signal-to-Noise Ratio (*SNR*). However, this advantage is not present in the asynchronous version because there is no sampler inside the loop [113].

Asynchronous Sigma Delta Modulation (ASDM) was presented in [110] and since then some studies have been carried out on this technique [114, 115, 113, 91]. Likewise in the case of Spike Delta modulations, different output states are possible and, again, just binary and ternary are analysed. The block diagrams of the former, the Binary Spike Sigma-Delta (BSSD) modulation and the latter, the Ternary Spike Sigma-Delta (TSSD) modulation, are shown in figures 3.8(a) and 3.9(a), respectively. However, the implement ion of these modulations as presented in the figures are impractical, due to high amplitudes required for the signal $y(t)$, specially for higher resolutions.

3.6 Evaluation of the communication methods: Why TSD?

The Ternary Spike Delta (TSD) modulation (the heart of the SEC scheme) and other alternative differential methods were described in the previous sections. Having different characteristics and properties, a common measurement is required to quantitatively compare these modulations. The channel efficiency measure is defined next followed by simulation results. Although those channel efficiency expressions for each method was derived, a direct comparison is difficult to visualize. Therefore, numerical simulations of models for each modulation were used to help in the evaluation.

3.6.1 Channel efficiency

In this architecture, the usage of the common channel is the most important factor. In this context, it is desirable that a communication method uses as few as possible the channel to convey the signal with a specific quality (resolution) to an efficiently utilization of the channel capacity. For this thesis, Channel Efficiency (CE) is defined as

$$CE = \frac{f_{in} \times 2^{ENoB}}{N_s} \tag{3.15}$$

where N_s is the number of spikes per second transmitted when a sine wave with frequency f_{in} is used as the input signal $x(t)$ and $ENoB$ is the coder resolution expressed in *Effective Number of Bits*.

According to this definition, each modulation is evaluated according to the number of spikes generated by a specific signal for a fixed resolution coding. In other words, the smaller the spike frequency (for a specific input frequency), the better is the coder performance. This metric was used because the utilization of the communication channel is the main limitation foreseen for large analogue arrays using such modulations. This definition resemble a common ADC figure of merit cited in [116] and the one presented in [106] for asynchronous ADCs with the power being replaced by the spike frequency.

Approximated instantaneous spike frequency expressions for each modulation is presented in table 3.1. It also presents the input signal conditions needed for the maximum and minimum number of spikes, the number of spikes and measurement for the channel efficiencies. Derivations of table 3.1 results can be found in the appendix A.

Modulation	TSD	BSD	ADM	ASDM	BSSD	TSSD
$f_{sk}(t)$	$\frac{\lvert\dot{x}(t)\rvert}{\delta}$	$\frac{\dot{x}(t) - k_i\, p_{dc}}{\delta - k_i\, p_{dc}\, T}$	$\frac{k_i^2 - [\dot{x}(t)]^2}{2\, k_i\, \delta}$	$\frac{b^2 - [x(t)]^2}{4\, \delta\, b\, k_i}$	$\frac{1}{T}\frac{x(t) - y_{(min)}}{\Delta y}$	$\frac{1}{T}\frac{\lvert x(t)\rvert}{y}$
$x(t)\rvert_{f_{sk}(t)=f_{sk(max)}}$	$\lvert\dot{x}(t)\rvert_{(max)}$	$\dot{x}(t) = -k_i\, p_{dc}$	$\lvert\dot{x}(t)\rvert = 0$	$x(t) = 0$	$x(t)_{(max)}$	$\lvert x(t)\rvert_{(max)}$
$x(t)\rvert_{f_{sk}(t)=f_{sk(min)}}$	$\dot{x}(t) = 0$	$\dot{x}(t) = k_i\, p_{dc}$	$\lvert\dot{x}(t)\rvert = k_i$	$\lvert x(t)\rvert_{(max)} \le b$	$x(t) = y_{(min)}$	$x(t) = 0$
$f_{sk}(t)\rvert_{x(t)=x_{DC}}$	0	$\frac{1}{T - \frac{\delta}{k_i\, p_{dc}}} \simeq \frac{f_{sk(max)}}{2}$	$\frac{k_i}{2\, \delta} = f_{sk(max)}$	$\frac{b^2 - (x_{DC})^2}{4\, \delta\, b\, k_i}$	$\frac{1}{T}\frac{x_{DC} - y_{min}}{\Delta y}$	$\frac{1}{T}\frac{\lvert x_{DC}\rvert}{y}$
$f_{sk}(t)\rvert_{x(t)=0}$	0	$\frac{1}{T - \frac{\delta}{k_i\, p_{dc}}} \simeq \frac{f_{sk(max)}}{2}$	$\frac{k_i}{2\, \delta} = f_{sk(max)}$	$\frac{b}{4\, \delta\, k_i}$	$\frac{1}{T}\frac{\lvert y_{(min)}\rvert}{\Delta y}$	0
N_s	$2\, f_{in}\, 2^{N_b}$	$\pi\, f_{in}\, 2^{N_b}$	$\frac{\pi}{2}\, f_{in}\, 2^{N_b}$	$\frac{b^2 - 0.5A^2}{2\, d\, b\, k_i}$	$\frac{A}{T\, \Delta\, y}$	$\frac{2A}{\pi\, T\, y}$
CE	$\frac{1}{2} = 0.5$	$\frac{1}{\pi} \approx 0.32$	$\frac{2}{\pi} \approx 0.63$	$\frac{2\, d\, b\, k_i\, f_{in} 2^{N_b}}{b^2 - \frac{A^2}{2}}$	$\frac{T\, \Delta\, y f_{in}}{\delta}$	$\frac{\pi\, T\, y\, f_{in}}{2\, \delta}$

Table 3.1: *Spike frequency and channel efficiency comparison.* $f_{sk}(t)$ *is the approximate instantaneous spike frequency,* $x(t)\rvert_{f_{sk}(t)=f_{sk(max/min)}}$ *is the condition for maximum/minimum spike frequency,* $f_{sk}\rvert_{x(t)=x_{DC}}$ *is the spike frequency for DC input signals,* N_s *is the number of spikes per second for an sine wave input* $x(t) = A\ \sin(\omega_{in} t)$ *and* CE *is the channel efficiency for each modulation. The TSD modulation is the only that presents no spikes for any DC input signal.*

As expected the spike frequency of delta-based modulations is a function of the input signal *derivative*. But the frequency characteristics are very different: whilst the maximum frequency when using ADM is met with a DC input signal, the same signal will not generate spikes on the TSD modulation (and about half of the maximum for the BSD case).

In details, the ADM output frequency is proportional to the square of the input derivative. Therefore this frequency reaches its minimum with fast changing input signals and there is a *self-oscillation frequency* (f_0) for DC input signals[8]. Alternatively, the spike frequency for BSD modulation is a function of the input derivative. The maximum spike frequency is obtained when the input signal *rises* at its maximum rate. When the input signal *falls* at maximum rate, the spike frequency is the minimum. For TSD, the spike frequency is a function of the *absolute* input derivative[9]. Consequently, there are no spikes when the signal does not change[10].

Considering the sigma-delta-based modulations, their spike frequency is a function of the amplitude of the input signal. It is proportional to the square of the input signal amplitude in ASDM method; to the amplitude itself in BSSD case and to the absolute value for TSSD. As for the delta-based cases, these differences reflect on the conditions where the method will produce the minimum and maximum output frequency. For instance, whilst a null input signal will generate no spikes in a TSSD method, the same signal will generate spikes at the maximum frequency for ASDM method. The opposite is true when the input signal reaches its maximum or minimum: no or few spikes for ASDM and maximum output frequency for TSSD.

All these three modulations present a self-oscillation frequency for DC input signals [118], with the exemption of TSSD when the input amplitude is null (although very low self-oscillation frequencies can be obtained for the other methods in extreme cases, i.e., $|x(t)| = x_{max}$). Only the TSD modulation has no self-oscillation frequency for any constant input.

Regarding the number of spikes generated per second N_s, TSD method generate more spikes ($4/\pi$) than ASM method and less ($2/\pi$) than BSD method. The number of spikes per second is a function of the signal amplitude in sigma-delta versions, with the binary version (BSSD) also presenting $\pi/2$ times more spikes per second than the ternary case (TSSD).

[8] In synchronous delta modulation systems, this effect is also known as granular noise [117].

[9] For comparison purposes, it was shown [102] that the frequency of asynchronous version of TSD is $\frac{\pi}{3\sqrt{2}}$ smaller than the synchronous version.

[10] In fact, no spikes are generated in TSD modulation when the change in the input signal $x(t)$ is smaller than the quantization step δ

Comparing delta with sigma-delta modulations using the table 3.1 is not straightforward. Numerical simulations were performed to visualize this comparison. The results are presented in the next subsection.

3.6.2 Comparison results and discussion

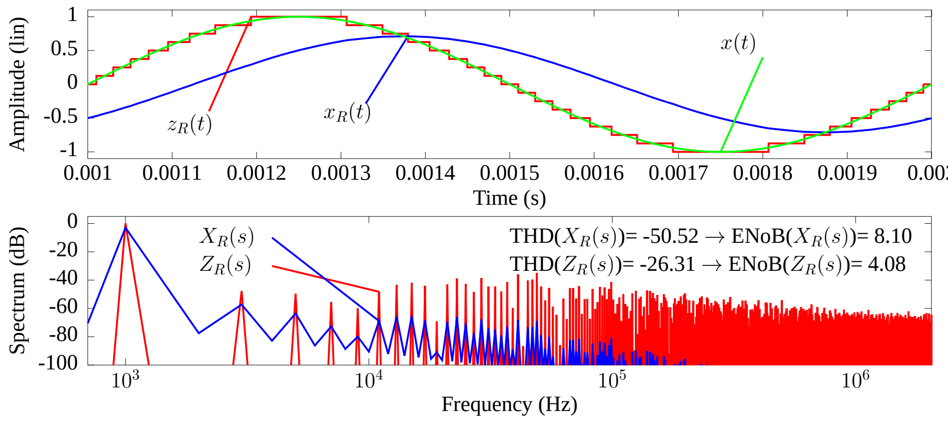
Models of all modulations were designed for resolutions spanning from 3 to 8 bits and simulated using a 1-kHz sine wave. The conversion results and the frequency response for the 4-bit resolution case and frequency spectrum are plotted in figures 3.10 and 3.11 for delta-based and sigma-delta-based modulations, respectively.

In both figures, two graphs are reserved for each modulation. The first one shows the coder input signal $x(t)$, the decoder output signal $x_R(t)$ and, for delta-based cases, the output of decoder integrator $z_R(t)$[11] in time domain. Due the filter design, where the pole is selected to be the same as the input signal frequency, all signals $x_R(t)$ present a 3 dB attenuation and a 45°-phase shift. The second graph corresponds to the frequency characteristics $X(s)$, $X_R(s)$ and $Z_R(s)$ of the same signals, respectively, having the fundamental component at 1 kHz.

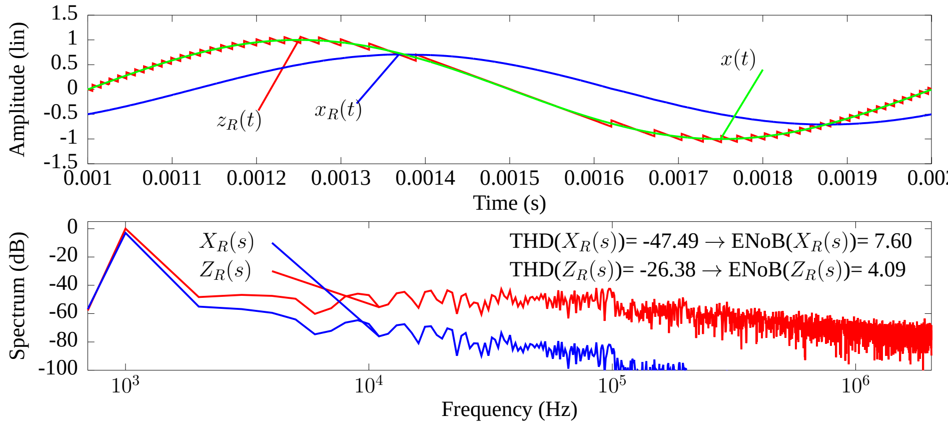
Although the modulations are asynchronous by definition, all simulations were performed using sampled signals. According to the sampling theory, frequency components of a sampled signal which are greater than the Nyquist frequency $f_s/2$ will be mirrored (around Nyquist frequency) and then added into the lower frequency components. Therefore, by performing sampling on the signals involved, the simulation provides worse results than in reality. To minor this effects, one can increase the sampling frequency, i.e., increase the *Oversampling Ratio* (OSR). By doing that, the amount of the frequency components mirrored to baseband $(f \leq f_s/2)$ will decreased, considering that all modulations present a low-pass filter at the output. For the simulations presented here, the sampling frequency used was 16.78 GHz for an input signal frequency of 1kHz (OSR = 2^{24}).

Also using these simulations, the number of spikes per second N_s generated were measured and they are presented in figure 3.12(a), whilst figure 3.12(b) shows the calculated N_s using equations on table 3.1. The difference is due to the fact that some expressions on table 3.1 are approximations from the actual expressions.

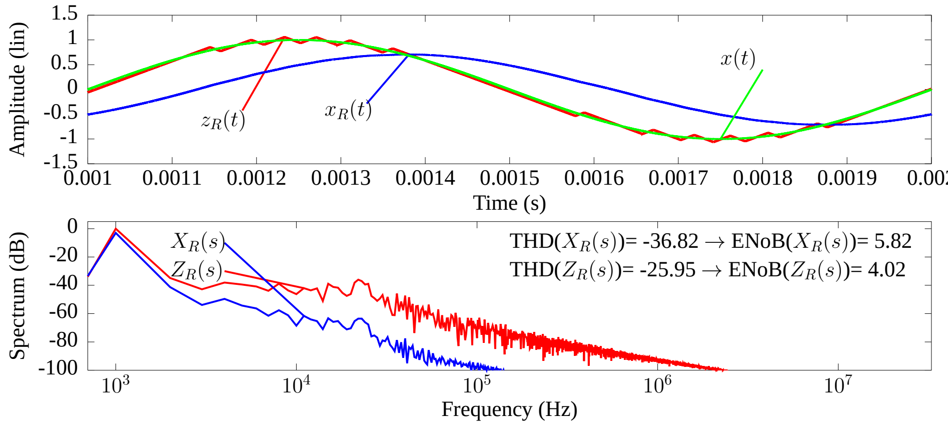
[11] Different to sigma-delta modulations, where the decoder filter is compulsory, its implementation is optional (although desirable) in delta modulations. Therefore, signal $z_R(t)$ is also showed to comparison purposes.



(a)

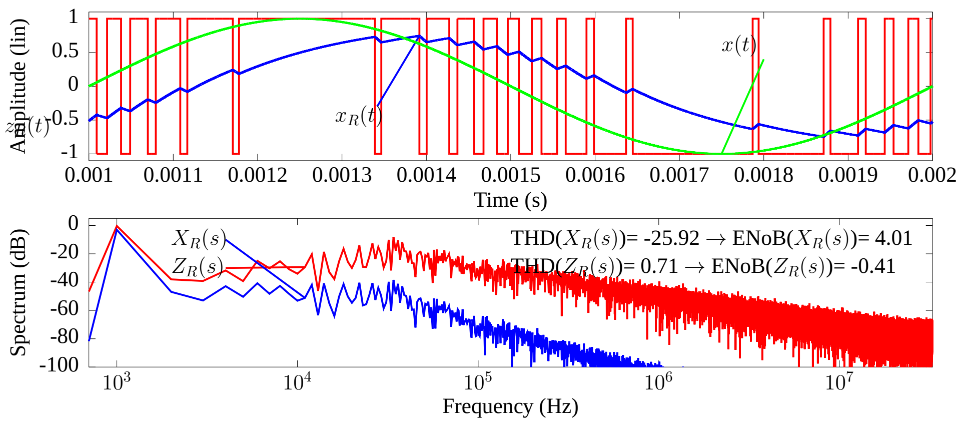


(b)

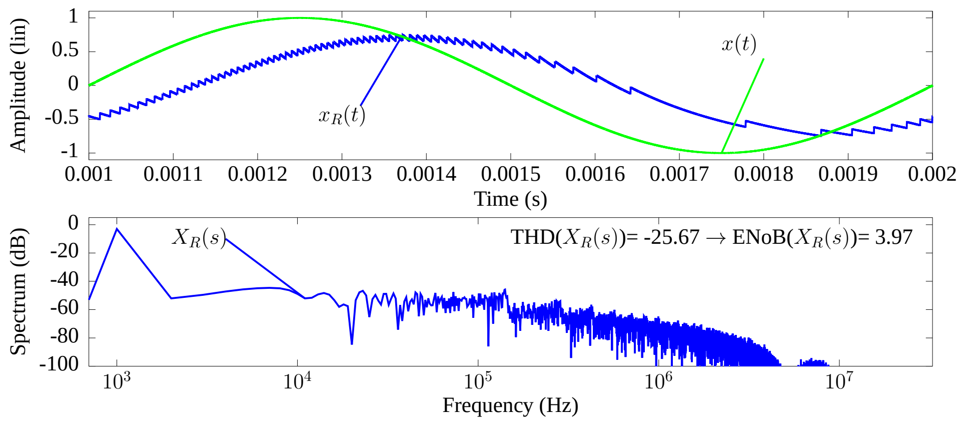


(c)

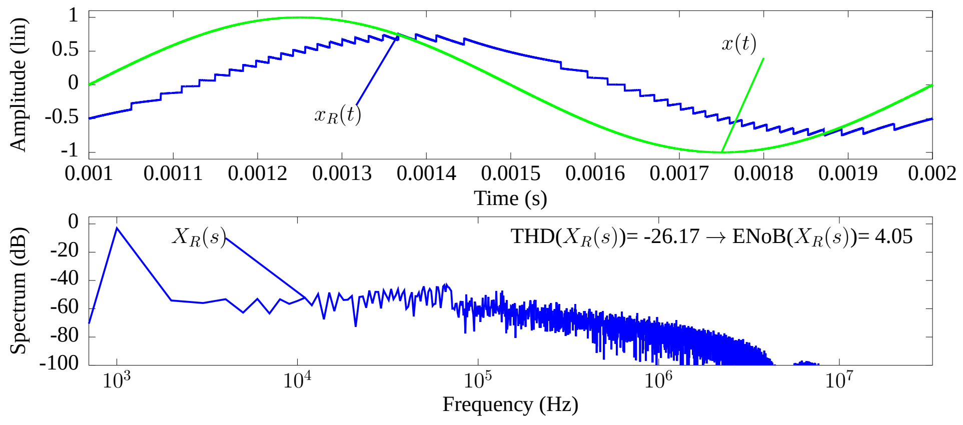
Figure 3.10: *Numerical simulation of the delta modulations models. The signal reconstruction and the frequency spectrum of this reconstruction are shown for each modulation: (a) TSD, (b) BSD and (c) ADM. Signal $z_R(t)$, as well $z(t)$, is the result of the* discrete *output spikes and tracks the* continuous *input signal $x(t)$. How close is this tracking depends on the system resolution.*



(a)



(b)



(c)

Figure 3.11: *Numerical simulation of the sigma delta modulations models. The signal reconstruction and the frequency spectrum of this reconstruction are shown for each modulation: (a) ASDM, (b) BSSD and (c) TSSD. For the first case (ASDM), this figure also shows the pulse signal produced by the incoming spikes $y_R(t)$. This intermediate signal, named $z_R(t)$, is the conventional asynchronous sigma-delta output signal.*

A comparison of the channel efficiencies of the modulations from simulations can be visualized in figure 3.13. In this figure, the graphs (a) and (b) present the results of *normalised* channel efficiency considering the output $x_R(t)$ whilst the graphs (c) and (d) show *actual* channel efficiencies using the signal $z_R(t)$. Normalisation is need for $x_R(t)$ because the design was performed to obtain N_b on the signal $z_R(t)$ for delta-based modulations. Therefore the computation of channel efficiency considering $x_R(t)$ would lead to efficiencies greater than unity because of the decoder filter.

By the analysis of these graphs and the table 3.1, the ADM method presents the highest channel efficiency of delta-based cases studied. However this is a valid result when considering the signal $z_R(t)$. For the results based of the signal $x_R(t)$, the TSD presents a better efficiency for low resoltions. This is mainly because the segments of the TSD $z_R(t)$ resembles square waveforms, presenting greater harmonic components at high frequency spectrum than the triangular shape of ADM $z_R(t)$. Therefore the effect of the low-pass filter is more significant. For higher resolutions the effect of the filter is not so important because the harmonics amplitude decreases, i.e., the signal $z_R(t)$ becomes more similar to the input signal and the better channel efficiency of the ADM starts to reflect in the signal $x_R(t)$ as well.

Besides presenting the better channel efficiency figures for low resoltions, the TSD modulation was chosen as the modulation for the communication method because of other two factors:

1. TSD modulation is the only modulation studied that presents no spikes for constant input signals. This analogue array is intended to work for low-frequency input signals and it is expected that many of the input signals might present no activity during a considerable period of time. By making this assumption, it is likely that a system with TSD modulation implemented will generated the lowest communication traffic.

2. As it will be shown in the computational characteristics in chapter 4, TSD modulation presents some properties that enable an inherent signal computation.

3.7 Summary

An alternative communication method to the commonly used switch matrices was introduced: the *Asynchronous Spike Event Coding* (ASEC) scheme. The method relies on an efficient method of analog-to-timing conversion and on an asynchronous communication protocol (AER).

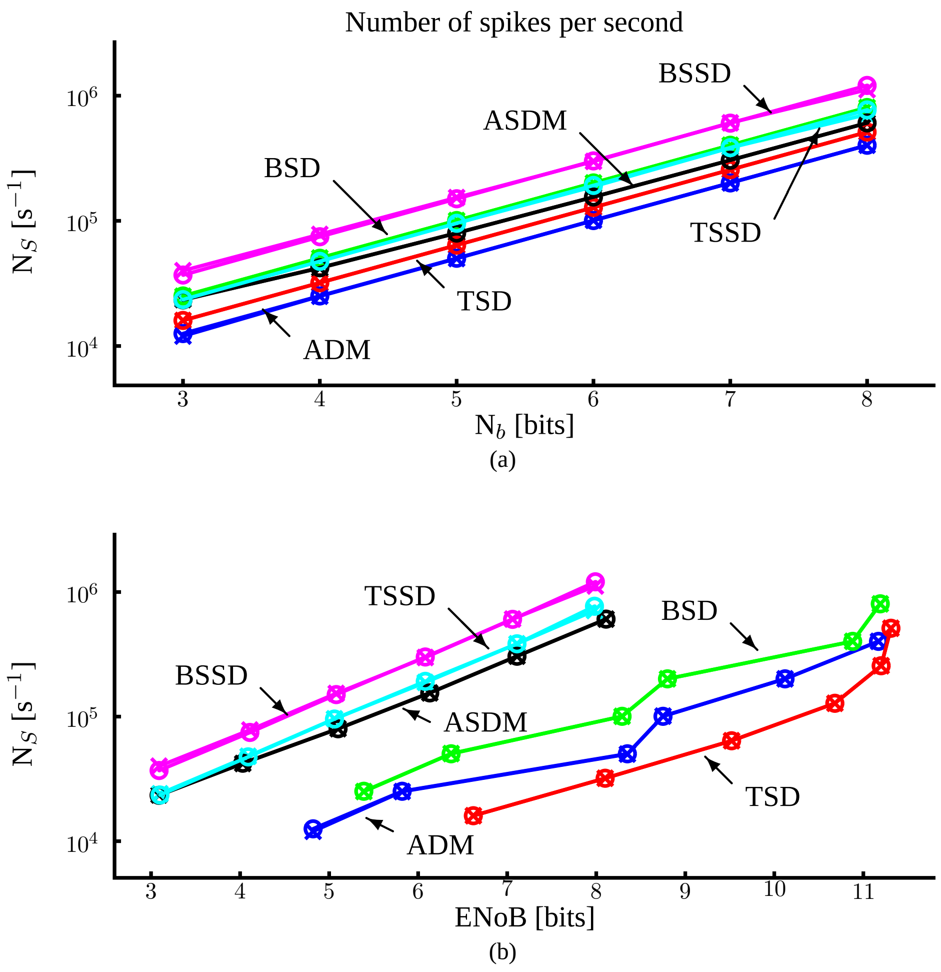


Figure 3.12: *Number of spikes per second (N_s) for the communication methods. Output activity of all coding methods studied. The graphs present both the calculated values of N_s according to table 3.1 (indicated with the marker 'o') and measured from simulations (marker 'x'). The difference between calculated and measured lies on the fact the equations in table 3.1 are approximated expressions. (a) are the results when using the designed resolution N_b and (b) when using the output $ENoB$. Differences are more visible on delta-based methods because the design is performed to using the required resolution on $z_R(t)$ rather than on the coder output $x_R(t)$. Markers 'x' and 'o' are equals for delta-based modulations on (b) because no calculations were performed for the filter output $x_R(t)$.*

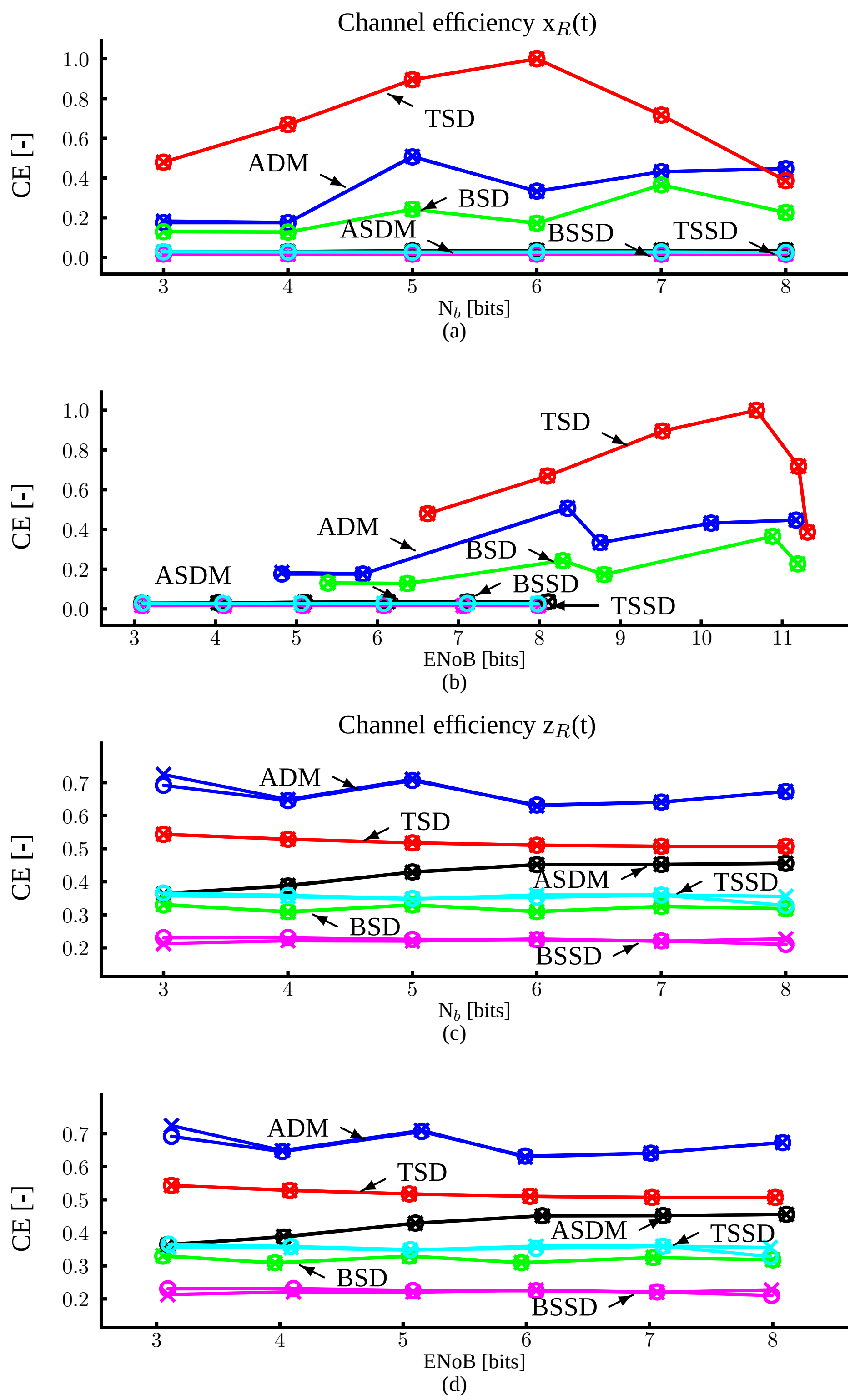


Figure 3.13: *Channel Efficiency (CE) of the communication methods. (a) calculated (see table 3.1) and simulated (b)* normalised *channel efficiency of each communication method in terms of the designed resolution N_b and the resolution at the output $x_R(t)$. Graphs (c) and (d) show the* actual *channel efficiency when signal $z_R(t)$ is considered. Lines market with 'x' and 'o' have similar meaning as in figure 3.12.*

The conversion method was selected among several differential asynchronous pulse modulations which were presented and analysed. These modulations are spike-based derivations of the well-known PWM-based delta and sigma-delta modulations.

Because our architecture uses a common digital channel, the main criteria to evaluate these modulations is the channel activity required by each of them. A measurement for the channel utilisation (channel efficiency) was defined as the amount of spike generated for a specific decoder output resolution. For each modulation studied, table 3.1 presents analytical expressions for the spike activity.

From these expressions one can notice that TSD is the only one that does not output spikes for constant input signals. This fact adds to the highest channel efficiency for low resoltions from the simulations and its unique computational properties as the main reasons for the selection of the TSD as communication method for the architecture.

In this thesis, there is *no* claim about the novelty of the analogue modulation used (TSD). The novel aspects introduced in this thesis are the TSD usage in an analogue array architecture working in association with the AER protocol — *Asynchronous Spike Event Coding* (ASEC) scheme — and the analysis of the computational properties of such scheme as presented in chapter 4.

Apart the alternative methods evaluated here, other variations of the differential modulations are possible. These include Asynchronous Pulse Length Modulation (A-PLM) and Asynchronous Bipolar Pulse Length Modulation (AB-PLM) as presented in [103] and asynchronous versions of the Continuously Variable Slope Delta (CVSD) modulation. The adequacy of these methods would be evaluated in future works.

In the next chapter the computation properties of the ASEC will be demonstrated with examples of some elementary and complex functions.

Chapter 4
Computation in communication with asynchronous event coding scheme

4.1 Introduction

In the chapter 3 a novel analogue communication method for analogue arrays based on asynchronous events and using a common channel to transmit these events were presented. Analogue-to-timing conversion methods were studied and evaluated. The TSD modulation was selected among other methods according to a figure of merit, being the core of the ASEC scheme. In this chapter, it will be shown that the ASEC may be configured to perform a set of analogue computations, corresponding to the second part of the statement of hypothesis in section 1.2.

Although the first electronic computations were performed by analogue circuits, digital techniques gradually become more popular over the years, mainly due to its reliability, simple configuration and a wide range of computations performed by a small set of basic blocks. Consequently, analogue computation had became restricted to a small set of operations. In particular, these operations perform low complexity functions requiring real-time processing, low-power consumption, low circuits area. Moreover, analogue circuits are used to interface real world signals, who present a continuous-value representation. Current examples of analogue computations are found in various types of sensors [94, 119, 120] and array-based applications [121, 13]. However analogue computation has potential benefits compared against its digital counterpart. When considering low to moderate (SNR up to 12-bit) resolution signals, analogue computation could be tens of thousands times more efficient and less costly than custom digital processing [7, 6].

Analogue numeric representation is based on continuous physical properties which leads to a limited precision. Analogue computations are generally performed using voltages or currents as the physical representation of the real numbers. These quantities are not appropriate to represent high dynamic ranges within low-voltage, low-power circuits.

Alternatively, the use of timing as the numerical representation for computations has been studied [21, 122] and presents an important advantage due to the evolution of the CMOS fabrication. Pushed by the digital processor market requirements, CMOS technology features tend to decrease to allow smaller and faster circuits. Considering the current applications, this evolution represents a further increase of the allowed dynamic range when using timing representation[1].

One of the first uses of timing processing was presented by Prof. Tsividis [18], who developed a filter structure where the filter coefficients were set by switching times. In his work, Prof. Tsividis showed that using a couple of switches and a low pass filter (LPF), it is possible to perform a multiplication of a signal by a constant (gain). This kind of operation, the summation of gains is used extensively in artificial neural networks (ANN) [123].

More recently, research groups started to use timing as signal representation to develop analogue processing. Pulse Width Modulation (PWM) was used to implement switched filters [21] and to carry out arithmetic operations [124], signal converters and information storage [125]. Other modulations have been used in specific applications as voice and sound and video processing [126], but a few groups are keen to use them in a more generic scope within analogue processing.

One of them, the Analog VLSI and Biological Systems Group at MIT, leaded by Dr. Sarpeshkar, presented a hybrid computation technique [127]. He defined hybrid computation as a type of processing that combines the analogue computation primitives with the digital signal restoration and discretisation using spikes as a way to transform a real (analogue variable, the inter-spike intervals) number into an integer (discrete digital, the number of spikes) number.

However all these methods, apart the last one, have been designed for and used in particular applications, presenting a flexibility degree very limited to a generic analogue array. And Dr. Sarpeshkar's method requires accessory circuits to perform different operations.

The asynchronous spike event coding (ASEC) communication method presented in chapter 3 was found to be suitable to perform a series of generic computation on analogue signals in an analogue array. Moreover, all computations are realized with the circuits already implemented for the communication process. The idea underlying the computational properties of the communication scheme and some examples are described next.

[1]This assumption is valid if considering the same signals used in the current computations. Many of these signals are measures of physical properties and therefore, their dynamics are fixed. For instance, audio signals are useful only up to 20kHz.

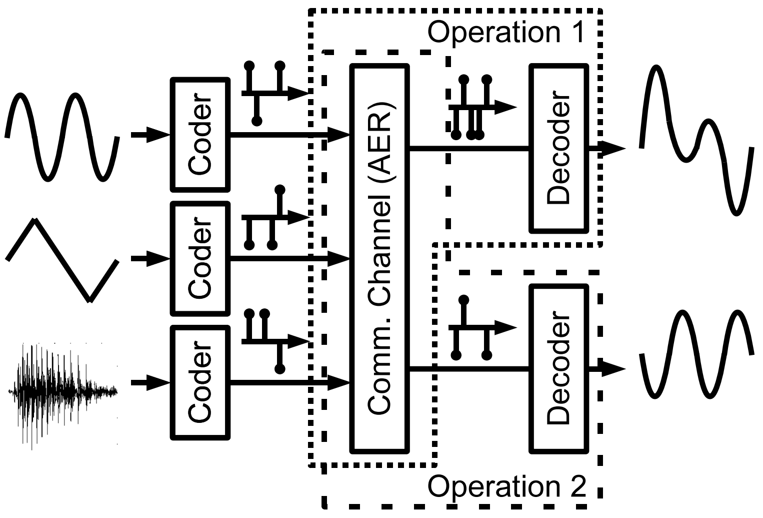


Figure 4.1: *Computation in the communication scheme. Analogue signals are converted into spikes by the coders and arithmetic computation may be performed by the AER and decoder combination. The type of computation performed is defined by programming the AER routing and decoder parameters.*

4.2 Computational framework

In general, programmable analogue arrays perform computations using the exiting functionalities of the programmed configurable analogue blocks (CABs), whilst the role of the communication interconnect is to route signals between those CABs and to external world. However, the computation power of a programmable array can be enhanced if the communication channel can be used to perform computations without additional overheads, as performed by the ASEC scheme presented in this work.

In the architecture proposed in chapter 2, CABs are interconnected using the ASEC scheme. The communication scheme consists of a spike event coder, AER and spike event decoder as shown in figure 4.1. The spike event coder encodes the analogue signals into asynchronous discrete amplitude signals (spike events) which are then routed to target CABs using an asynchronous event representation (AER) protocol. The decoder at the target CABs decodes the spike event signals received from the AER.

This communication scheme allows a set of arithmetic operations to be performed. For instance, in figure 4.1, the top decoder outputs a summation of two inputs (sine and triangular waves) and the second one provides a negation of one of the inputs (sine wave) using the same communication channel. These operations are performed simply by programming parameters of the communication channel. Other operations can be implemented by combining the channel reconfigurability with the analogue signal processing capability of the CABs. The computational realm of this communication scheme can enhance the computing power of the programmable array without using additional hardware.

Moreover, because of the shared nature of the channel, meaning it can enable multiple communications virtually at the same time, the computation operations can also be performed simultaneously. This intrinsic computation capability allows a simpler implementation than other pulse based approaches, [128] as an example. Basic arithmetic operations will be the first examples to be presented in the next sections.

4.3 Arithmetic operations

In this section a set of fundamental arithmetic computations performed by the communication scheme are presented. Each operation is illustrated by simulation of the mathematical models presented in the Appendix B. Actual chip results will be presented in chapter 5.

The following results were obtained for a 4-bit resolution conversion. Finally, the outputs waveforms shown in the figures correspond to the decoder filter input $z_R(t)$ rather than the coder output $x_R(t)$ for simplicity as explained in the next chapter.

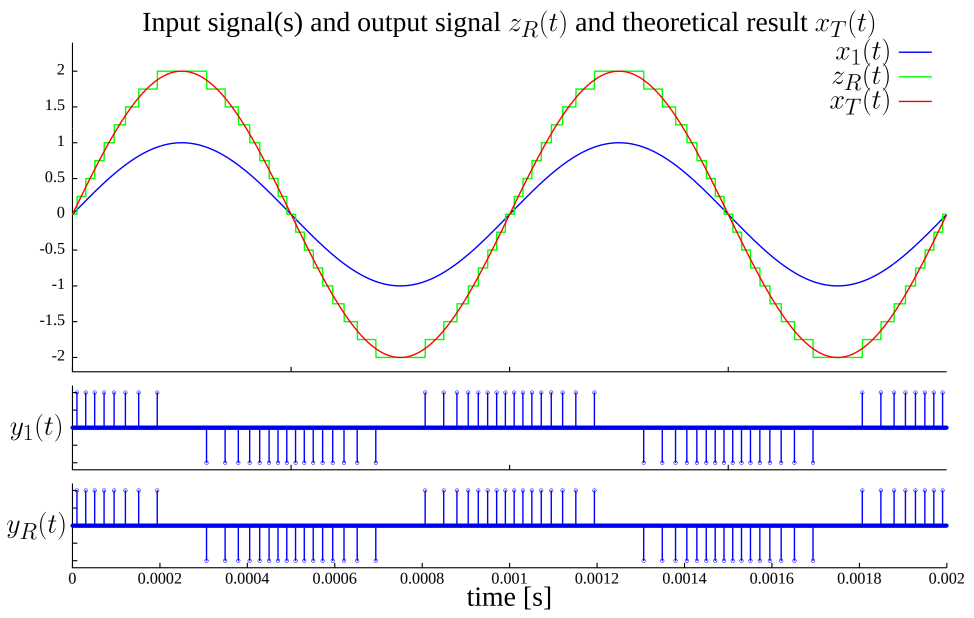
4.3.1 Gain operation

A common operation in any analogue processing is to change the amplitude of a signal, i.e., provide a gain G to the signal such as

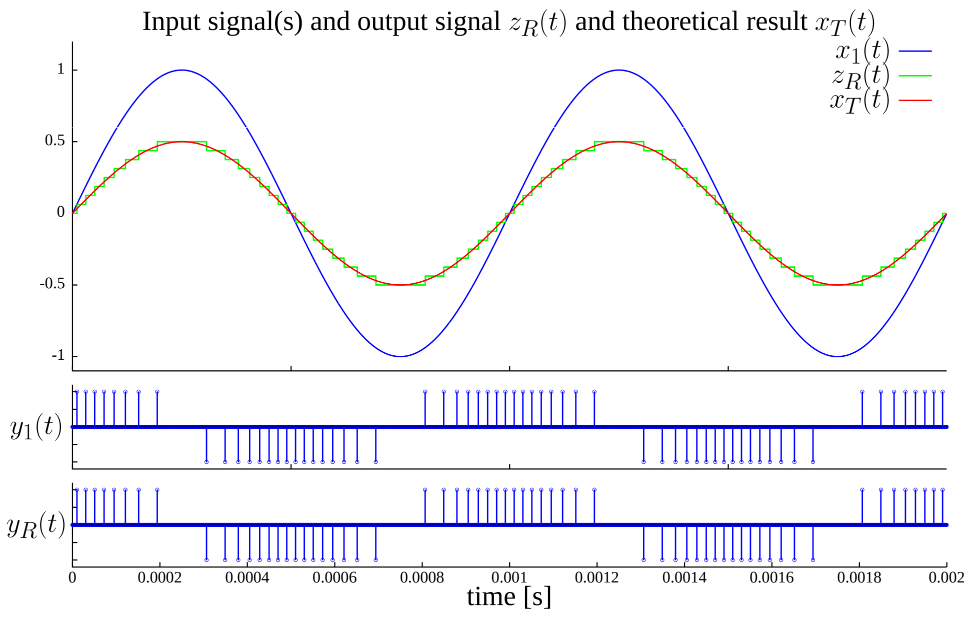
$$x_{gain}(t) = G \times x(t). \tag{4.1}$$

The gain may increase (amplification) or decrease (attenuation) the amplitude of the signal. Both gain types are possible using the proposed architecture.

Equation 3.6 is valid for both coder and decoder blocks. However, if coder and decoder are designed to present different tracking steps then the decoder output will be proportional rather



(a)



(b)

Figure 4.2: *Gain operation. The input sine wave signal $x(t)$ is amplified by a factor of 2 in (a) and attenuated by 2 in (b), at the decoder $z_R(t)$. The gain operation is obtained by selecting the ratio of the tracking steps of coder and decoder. This operation will result in signal amplification by setting the tracking step in decoder greater than in the coder. Otherwise, it results in signal attenuation.*

than equal to the coder input. This proportionality is given by the ratio of both tracking steps

$$G = \frac{\Delta x_{gain}(t)}{\Delta x(t)} \approx \frac{\Delta z_R(t)}{\Delta z(t)} = \frac{\delta_R}{\delta}. \tag{4.2}$$

Having different tracking steps, the ASEC scheme will result in the coder and decoder working with different resolutions. However this difference would be attenuated by a proper decoder filter design. Figure 4.2(a) shows simulations for an amplification of a sine wave by a factor of 2 whilst an attenuation by the same factor is shown in figure 4.2(b).

4.3.2 Negation operation

A second fundamental operation on analogue signals is changing their polarity resulting in the signal negation:

$$x_{neg}(t) = -x(t). \tag{4.3}$$

According to equation 3.6, the signal value at any instant t (knowing the initial condition of coder and decoder) is a function of the number of events transmitted and their type. Setting the AER router to interchange the addresses of positive $y_p(t)$ and negative $y_n(t)$ spikes[2] such operation is performed. In other words, the AER router perform the address operation

$$\begin{aligned} y_p(t) &\rightarrow y_n(t) \\ y_n(t) &\rightarrow y_p(t) \end{aligned} \tag{4.4}$$

resulting in the signal at the decoder output $z_R(t)$ being an inverted version of the input signal $z(t)$. From equation 3.6

$$\Delta x_{neg}(t) \approx \Delta z_R(t) = -\Delta z(t) = \delta(n_n - n_p). \tag{4.5}$$

Since $z(t)$ and $z_R(t)$ are the quantized versions of $x(t)$ and $x_R(t)$, respectively, the operation of the equation 4.3 is obtained. An example using a sine wave signal is shown in figure 4.3.

[2]The coder output $y(t)$ as presented in the chapter 3 is actually the concatenation of both positive and negative spikes, i.e., $y(t) = y_p(t) \cup y_n(t)$.

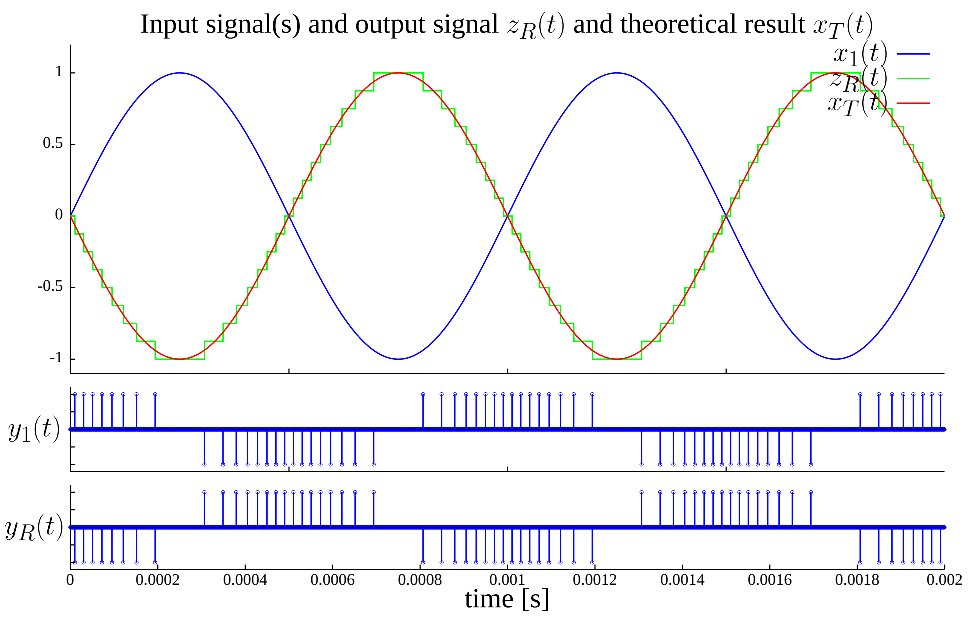


Figure 4.3: *Negation operation. Snapshot of the input sine wave $x(t)$ and the respective negated signal $z_R(t)$. Signal negation is perform by interchange the positive $y_p(t)$ and negative $y_p(t)$ spikes.*

4.3.3 Modulus

Another unary function commonly presented is the modulus of a signal. The modulus of a signal is the magnitude of such signal, defined as

$$x_{abs}(t) = |x(t)|. \tag{4.6}$$

Such function in the proposed architecture is possible using the algorithm

$$\begin{aligned} y_p(t) \rightarrow y_p(t),\ y_n(t) \rightarrow y_n(t) \quad if \quad sig(x(t)) = +1 \\ y_p(t) \rightarrow y_n(t),\ y_n(t) \rightarrow y_p(t) \quad if \quad sig(x(t)) = -1. \end{aligned} \tag{4.7}$$

where $sig(x(t))$ represents a control variable which is a function of the signal $sig(x(t)) = x(t)/|x(t)|$.

This algorithm can be implemented in different two ways. The first with an analogue comparator and a second using a digital counter. In both cases, the result in the decoder is the modulus of the signal. This operation is illustrated in figure 4.4 for a sine wave signal.

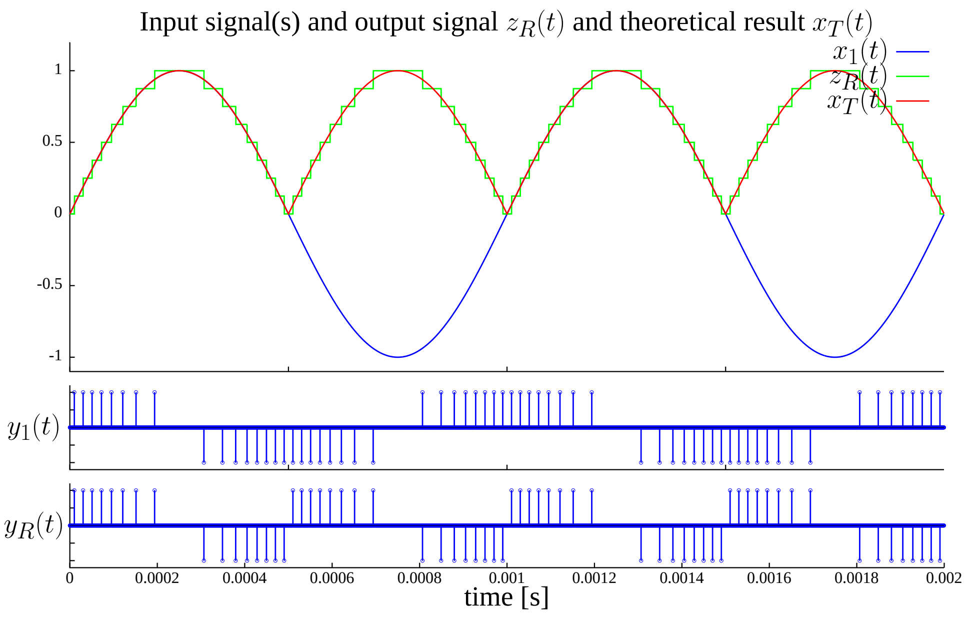


Figure 4.4: *Modulus operation. In this configuration, the modulus of the input signal $x(t)$ is outputted as the result of selectively applying the negation operation. An digital counter or a analogue comparator can be used to control the periods when the negation is performed.*

In the first method, the original signal is compared against a reference signal representing the zero value of the original signal. This comparator can be included in the same or in a distinct CAB. The comparator output signal $x_c(t)$ is coded and transmitted to the communication channel. The AER router controller reads this signal and set (+1) or reset (-1) the variable $sig(x(t))$ on positive and negative spikes reception respectively. If this variable is reset, the AER router performs the negation operation on the original signal as described in subsection 4.3.2. The operation is then performed using the algorithm

The second method employs a digital counter to measure the positive and negative spikes generated by the original signal. In order to work properly, the initial condition of the signal should represent the zero value. If so, the counter is incremented for each positive spike and decremented for each negative one. Every time the counter value changes from a negative value to a positive and vice-versa, the variable $sig(x(t))$ is updated.

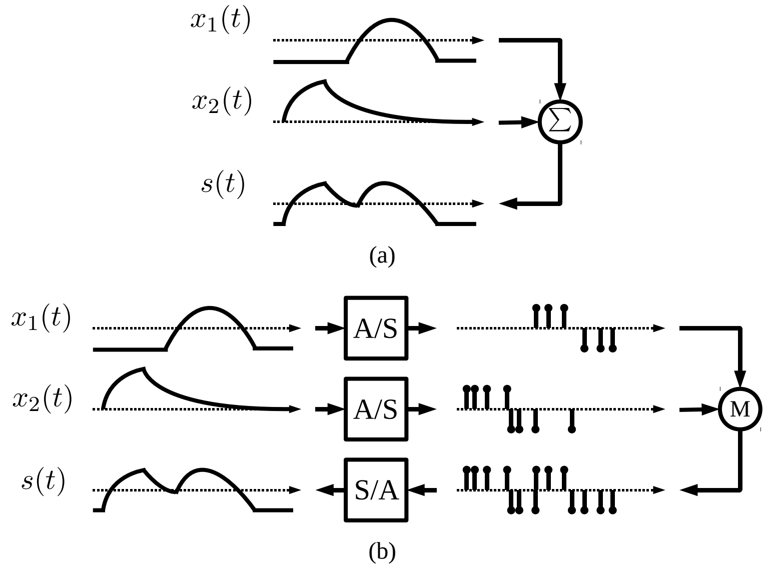


Figure 4.5: *Signal summation in analogue (a) and spike coding event systems (b). Analogue summation can be implemented in diverse forms, being the simplest the current summation according to Kirchoff's current law on a node. The summation on the proposed architecture is performed by simply* merging *the spikes generated by the input signals (operands). In (b), A/P and P/A stand for Analogue-to-Spike and Spike-to-Analogue conversions respectively.*

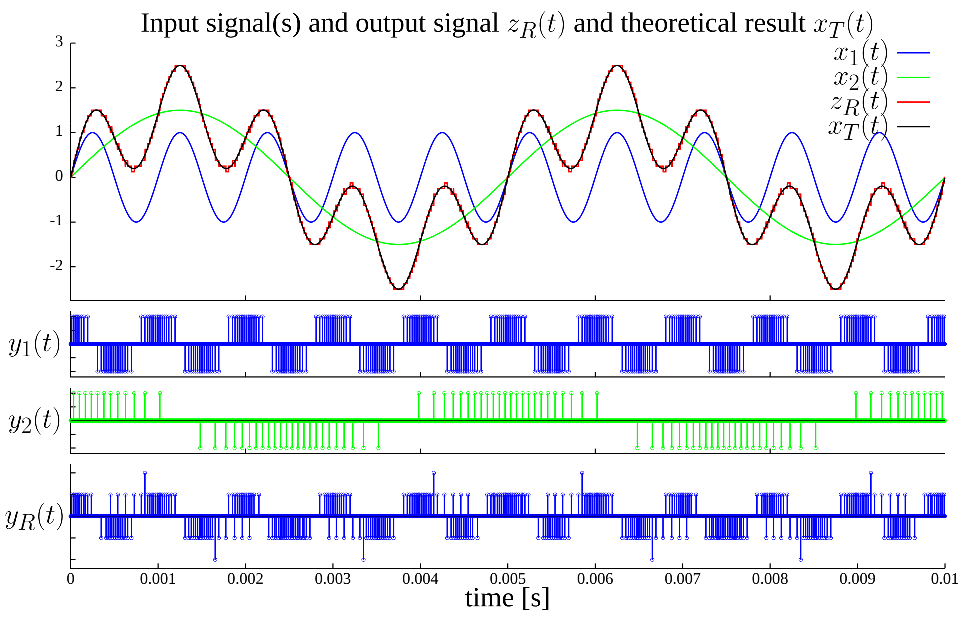
4.3.4 Summation and subtraction

Beyond these unary operations, arithmetic operations involving two or more signals are also required by any generic analogue computation system. The fundamental arithmetic operation involving multiple signals is the summation of these signals. Figure 4.5 shows the conventional concept of performing summation on analogue signal and in the proposed architecture. The summation signal $x_{sum}(t)$ with N signal operators is

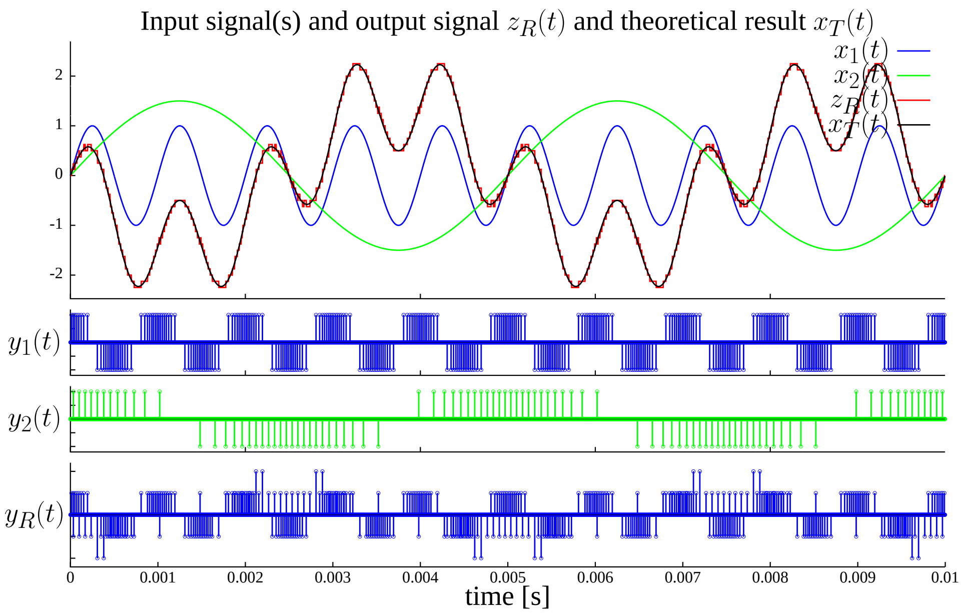
$$x_{sum}(t) = \sum_{i=1}^{N} x_i(t). \tag{4.8}$$

In order to implement this operation with the ASEC scheme, equation 3.6 is considered again. From this equation, it is possible to demonstrate that the summation signal of N operators is obtained at the decoder output $x_{sum}(t)$ by simply *merging* of their respective output spikes

$$x_{sum}(t) \approx z_R(t) = \delta \left(\sum_{i=1}^{N} n_{ip} - \sum_{i=1}^{N} n_{in} \right) + z_R(t_0) \tag{4.9}$$



(a)



(b)

Figure 4.6: *Summation and subtraction operations. (a) show the summation of two input sine waves, $x_1(t)$ and $x_2(t)$. Signal $z_R(t)$ is the respective summation whilst $x_T(t)$ is the ideal summation result. The subtraction of the same input signals $x_1(t) - x_2(t)$ is shown in (b). The summation is performed by the concatenation $y_R(t)$ of the spikes from all operands, i.e., $y_1(t)$ and $y_2(t)$ in this case whilst subtraction also requires the negation operation. Spikes collisions are identified whenever $y_R(t)$ presents twice the amplitude in these figures.*

where n_{ip} and n_{in} are the number of positive and negative spikes, respectively, received from the i^{th} operand after the initial instant t_0.

In this architecture, it is performed by routing the spikes of all operands to a same decoder, with CAB input (after the Spike-to-Analogue conversion) being the result. Simulation results showing the summation of two sine wave signals $x_1(t)$ and $x_2(t)$ are shown in figure 4.6(a), with the decoder integrator output $z_R(t)$. The signal $x_{sum}(t)$ is the theoretical result.

Subtraction $x_{sub}(t)$ may be performed applying both the summation and negation operations outlined before and an example is depicted in figure 4.6(b). Other methods to realize the summation in timing domain are also presented in appendix D.

4.3.5 Multiplication and division

Whilst some operations are performed using asynchronous spike event coding and decoding methods together with the AER router, the range of possible computations can be expanded when the communication scheme is combined with the functionality of the CABs.

This is one example of this cooperation. Once both summation and subtraction operations are performed by the ASEC scheme, multiplication and division are expected to be also possible to implemented in this architecture. This assumption is fulfilled using the logarithmic property in

$$x_{mult}(t) = \prod_{i=1}^{N} x_i(t) = \exp\left(\sum_{i=1}^{N} \ln\ x_i\right). \tag{4.10}$$

This property establish that multiplication (and division) operation can be computed from summation (and subtraction) operation if the operands $x_i(t)$ are submitted to a logarithm compression and the summation result expanded in an exponential fashion. Although this compression and expansion are not performed by the ASEC scheme itself, if this operations are implemented inside the CABs, both multiplication and division operations would be available in the array.

The conventional method to generate this logarithm compression are using the circuits in the figure 4.7. They are known as logarithmic and exponential amplifier respectively [129]. Although they present a simple topology, the diode resistance is usually highly dependent of the temperature. These circuits were not implemented for this thesis.

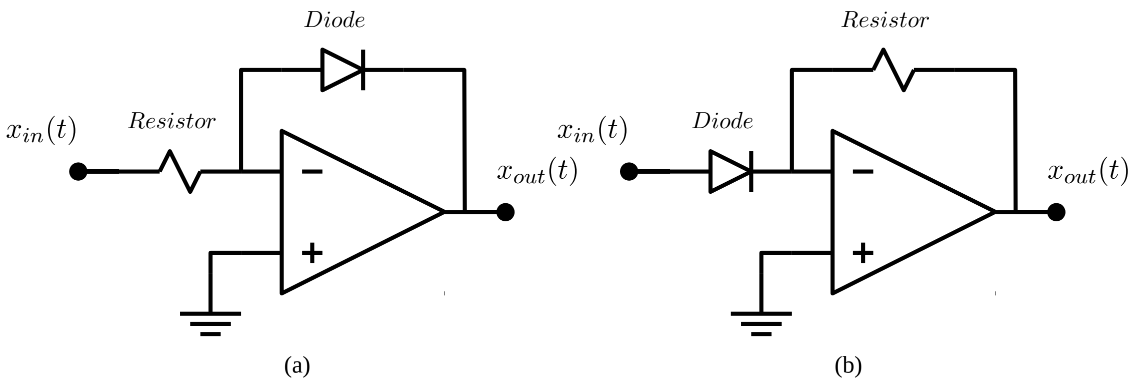


Figure 4.7: *Logarithm and exponential amplifiers. These figures show traditional circuits used to realize both logarithm (a) and exponential (b) operations on analogue signals. This circuits were not implemented in any chip designed for this thesis. These circuits can be implemented inside the array CABs to allow multiplication and division operations.*

4.3.6 Average operation

Once the gain and the summation operations are available, another possible operation is to perform the average of several inputs in each instant given by

$$x_{avg}(t) = \frac{1}{N}\sum_{i=1}^{N} x_i(t). \tag{4.11}$$

To implement this operation, the ASEC uses the same summation and gain procedures, where the gain G defined in subsection 4.3.1 is defined by

$$G = \frac{1}{N}. \tag{4.12}$$

Figure 4.8 presents an example of such computation using the same signals of the summation operation with the gain $G = 0.5$. This expression indicates that the minimum possible amplitude for the tracking step limits the maximum number of operators in this operation.

4.4 Signal conversion

The main property of a programmable analogue array is to process analogue signals using its CABs. However it might be of some interest to having a digital representation of the analogue signals, both for storage and transmission purposes, for instance. Analogue-to-Digital (ADC)

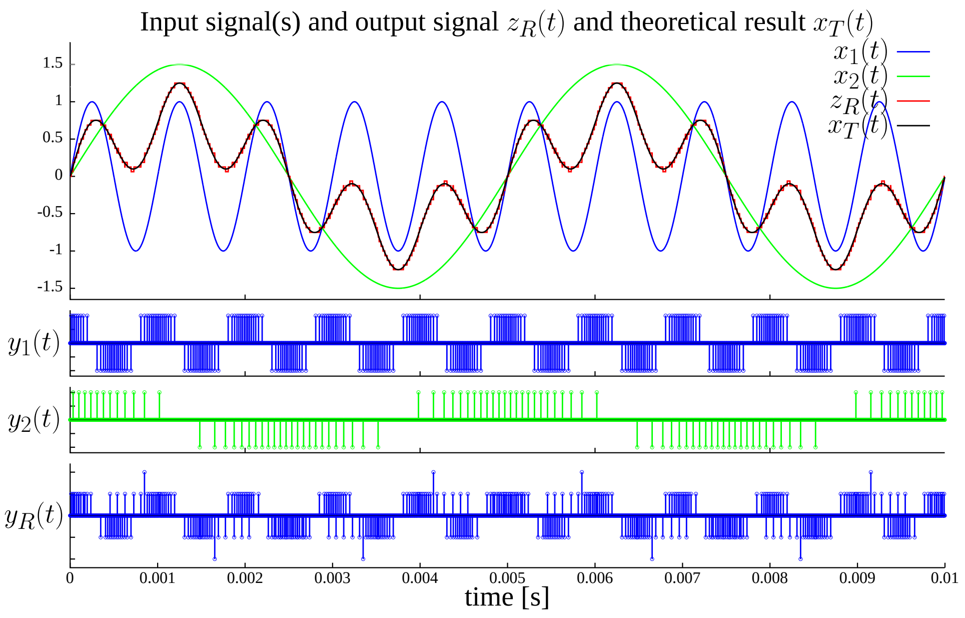


Figure 4.8: *Average operation. This figure shows the input signals $x_1(t)$ and $x_2(t)$ the respective Average signal $z_R(t)$. This operation is perform by combining the summation and gain operations, where the gain factor is the inverse of the number of operands.*

implementations based upon asynchronous modulations were presented in [113, 127, 130, 131].

Realizing data conversion on the proposed architecture can be achieved with the use of Timing-to-Digital (TDC) and Digital-to-Time (DTC) Converters. In its simplest implementation, TDC circuits consist of a digital counter operating at a specific frequency, which determines the converter resolution. Its input is an asynchronous digital signal which triggers the conversion process: read (and store) the current counter value and then reset it. The digital value represents the ratio between the interval of two consecutive inputs and the counter clock period:

$$X_i = \frac{t_i - t_{i-1}}{t_{clk}} \tag{4.13}$$

where X_i is the digital representation of the time interval $t_i - t_{i-1}$.

Therefore an Analogue-to-Digital (ADC) conversion is implemented as follows. Because each spike transmitted using the ASEC scheme presents the same signal magnitude change, the TDC can be used to read the output spikes $y(t)$ generated by a ASEC coder with input $x(t)$. An extra bit is therefore needed to represent the polarity of the spike. For instance, the least significant bit may indicate the spike polarity.

As it was said before, for constant input signals the ASEC coder will not generate any spikes. In this case, the digital counter can overflow and the information would be lost. One method to avoid this is to ensure the registry of these instants, storing or transmitting the maximum or minimum counter value, for instance. These value will add up until an incoming spike stops and resets the counter.

Another method, more complex, would be dynamically changing the tracking step of the ASEC coder, in the same fashion as the Continuously Variable Slope Delta (CVSD) modulation. Every time the counter has overflowed, the tracking step would be reduced (halved, for instance) to capture small signal variations. This approach will increase the digital word size because the step size have to be stored too. On the other hand, less digital words would be generated.

The complementary process can be used to perform an Digital-to-Analogue conversion (DAC). In this case a counter (DTC) is loaded with a digital word and then starts to count backwards until reaches the minimum value, typically zero, and reloads the next digital word. At this point a spike with the correct polarity is transmitted to an ASEC decoder throughout the AER communication channel to (re)generate the analogue signal.

These applications can be used to implement complex systems that could not be fit in just one programmable analogue array. Furthermore, this works only for non real-time systems.

4.5 Other operations

4.5.1 Shift keying modulations

Specific applications can also be realized with the proposed architecture. A first example is phase shift keying. Phase shift keying is a digital modulation where the information is coded in the phase of the carrier. For instance the Binary Phase Shift Keying (BPSK) is defined as

$$s(t) = z(t) = \begin{cases} a\ sin(\omega_c t) & if \quad b_i = 0 \\ a\ sin(\omega_c t + 180^\circ) & if \quad b_i = 1 \end{cases} \tag{4.14}$$

where b_i is the bit to be transmitted, ω_c is the angular frequency of the carrier and a is a function of the energy per-symbol and the bit duration [132].

This function can be implemented by providing the carrier (sine wave) signal to the input of

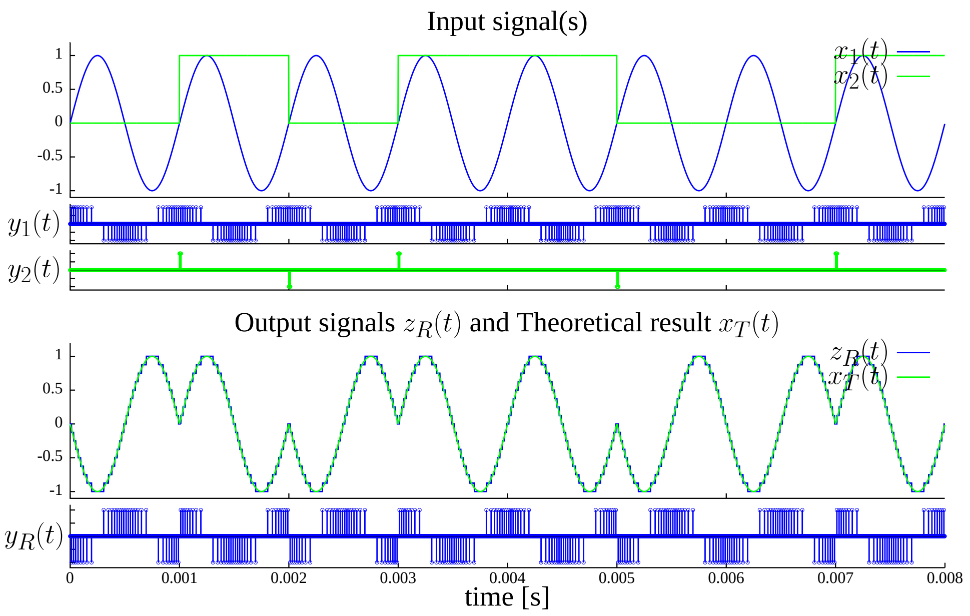


Figure 4.9: *Binary phase shift keying. This modulation is implemented by routing the spikes geberated by the sine wave carrier $x_1(t)$ according to the digital input $x_2(t)$. In this example the modulated digital word in 01011001.*

the coder and routing the spikes according to the digital value to be transmitted. For instance, whilst the input bit is zero, the carrier is replicated at the decoder output, but when the input bit is one, a negation function is performed on the carrier, i.e.

$$\begin{aligned} y_p(t) \rightarrow y_p(t),\ y_n(t) \rightarrow y_n(t) \quad if \quad b_i = 0 \\ y_n(t) \rightarrow y_p(t),\ y_p(t) \rightarrow y_n(t) \quad if \quad b_i = 1. \end{aligned} \tag{4.15}$$

Figure 4.9 shows the modulation result $z_R(t)$ for an input word $x_2(t) = 01011001$.

Quadrature phase shift keying (QPSK) is possible using the summation operation with another sine wave (90° phase shifted) and the generation of a fixed number of successive spikes to quickly change the carrier phase due to signal discontinuity on the second sine wave.

This function has a similar working principle as the function modulus in section 4.3.3. The difference is that the AER routing is controlled by an external signal $b(t)$ rather than the input signal itself.

4.5.2 Weighted summation

The last operation included in this thesis is the weighted summation. This operation, along a non-linear transfer function, is the functional core of the Artificial Neural Networks (ANN) systems. Other applications for the weighted summation include bias compensation in statistics, centre of mass calculation of a lever, among others. The expression for weighted summation is

$$WS(t) = \sum_{i=0}^{N} w_i(t)\, x_i(t). \tag{4.16}$$

The implementation using the proposed communication method is straightforward by combining the gain and the summation operations[3].

One advantage of implementing such computation in this architecture is the possibility of implement massive ANNs in real-time, because of the system inherit scalability. On the down side, because different inputs $x_i(t)$ are multiplied by different and usually changing gains $w_i(t)$, the decoder tracking step δ_R need to be changed before receiving the correspondent spike. This results in a further delay between the instant when AER routing computes the target address and the moment when the ASEC decoder implement the update on its output. This delay is needed to process this tracking step programming. This usually requires waiting for a DAC controlling the decoder integrator to settle to a new value.

4.6 Summary

This chapter was dedicated to present and demonstrate the computational aspects of the ASEC scheme. More than just act as an asynchronous communication method, this method offers the possibility of implement elementary but important operations on analogue signals.

These operations consist of arithmetic operations such as gain, negation, summation, subtraction, mean and weighted summation (linear operators). These operations are realized by either programming the AER routing addresses or by selecting appropriate coder and decoder parameters. The design of these parameters will be detailed in the next chapter.

These operations do not require any additional hardware than the ones required to implement

[3]This method do not work when changing the gain for constant signals.

the communication itself. Other methods to perform such operations usually require dedicated circuits, increasing the circuit area and limiting the circuitry flexibility.

Other operations are possible depending upon the hardware available. This hardware can be presented either inside the analogue processing elements (CABs), as in multiplication operation case (logarithmic amplifiers) and modulus operation (comparator), or external to the array, as digital counters required to modulus and data conversion.

The subject of the next chapter is the physical implementation of the ASEC scheme and the discussion about its characteristics and limitations.

Chapter 5
Asynchronous spike event coding scheme: VLSI design and characterisation

5.1 Introduction

In chapter 3 the spike event coding method was introduced with the TSD as the analogue coding method. It was selected among other methods according to its null spike frequency for constant input signals, its performance against a figure of merit and its computational properties.

In this chapter, an analogue VLSI design approach for the spike event coding method is presented and the functionality of its physical implementation is tested [133]. Evaluation and characterisation of the spike event coding scheme were performed with the design of two proof-of-concept integrated circuits (ICs).

The first prototype IC firstly introduced in [134] includes the basic blocks of TSD coder presented in chapter 3. This prototype was designed mainly to the test the functionality of each TSD block in particular and perform signal communication evaluation. As no AER router was implemented in this IC, the pulses originated from the spikes were externally (wired) routed.

The second IC was presented in [135] and includes a number of spike event coders and decoders and the AER circuitry needed to interconnect them. The function of this IC was to test the complete asynchronous spike event coding scheme by implementing on-chip AER communication protocol and, in particular, its computational properties described in chapter 4.

For this thesis, no specific CABs were designed, although a new circuit for neuromorphic applications have been developed by the research group [64]. The proposed communication method is independent on the CAB functionality and design used in the analogue array.

The first part of this chapter will describe the most important parameters in the spike event coding scheme design. Next, VLSI implementation of each functional block of the scheme is presented. The chapter ends with the discussion of test results obtained from the ICs.

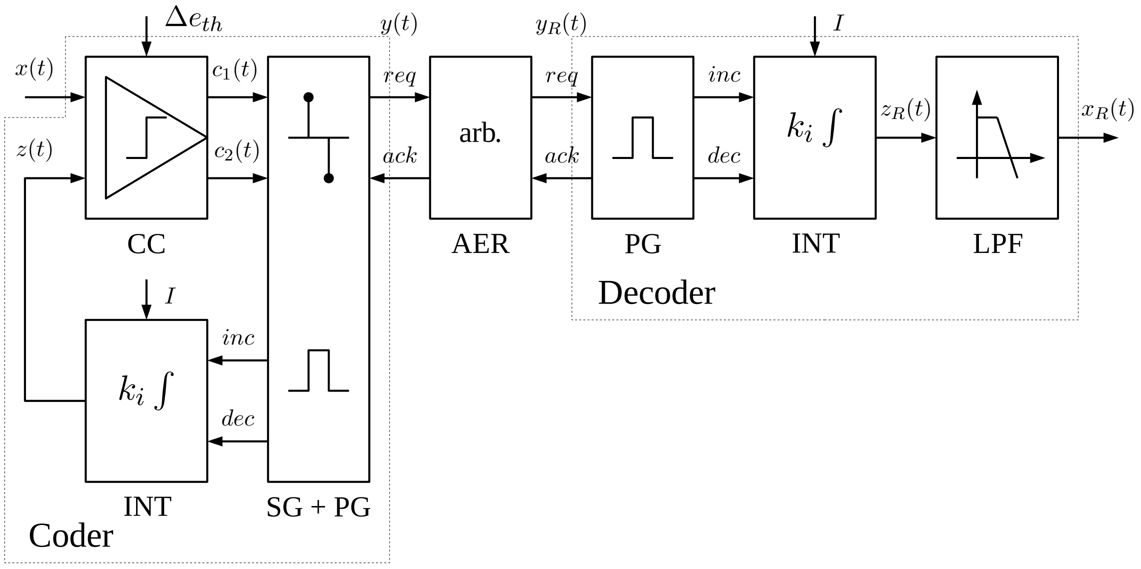


Figure 5.1: *Block diagram of the ASEC implementation. Asynchronous Spike Event Coding is the association of the TSD modulation with AER protocol channel included. CC is the* composite comparator *that perform the function of both comparators in figure 3.4(a). This blocks outputs signals $c_1(t)$ and $c_2(t)$ according to the input signal $x(t)$ and the system parameter $\Delta e_{th} = e_{th1} - e_{th2}$. SG is the* Spike Generator *and PG is the* Pulse Generator *which are merged into a single circuit in the coder side. SG generates the AER request signal req and waits for the acknowledge signal ack from the arbiter in the AER channel. On the reception of ack, PG outputs the integrator input pulses inc and dec. The gain k_i, which defines the tracking step δ value, of the* Integrator *block INT is set by the current I.*

5.2 ASEC implementation

As stated in section 3.2, the Asynchronous Spike Event Coding (ASEC) scheme uses the TSD modulation to perform the analogue-to-time conversion and the AER protocol to deliver and rout the spikes in the analogue array. The TSD block diagram was presented in figure 3.4(a). The ASEC block diagram presented in figure 5.1 is the result of the rearrangement of the TSD block diagram presented in figure 3.4(a) and the ASEC-CAB interface diagram presented in figure 3.2. This rearrangement was perform to better understand of the the implementation of each of the blocks designed.

The comparator, spike generator, pulse generator and integrator circuit design are further described along this chapter. Before the analysis of these blocks individually, the parameters of the ASEC and their design method are explained.

5.2.1 Design parameters of the ASEC

The first step in the design of the coder is to define the main[1] parameter of comparators of figure 3.4(a), i.e., the threshold e_{th1} and e_{th2}. Absolute values of these thresholds impact on the DC level of both $z(t)$ and $z_R(t)$. For a null DC level error, i.e., $\overline{e(t)} = 0$, these thresholds must be symmetrical ($e_{th1} = -e_{th2}$). A positive DC error is produced when $|e_{th1}| > |e_{th2}|$. Similarly $|e_{th1}| < |e_{th2}|$ results in a negative error. More importantly, the *difference* between the comparators thresholds holds a relation with the coding resolution as

$$\Delta e_{th} = e_{th1} - e_{th2} = \delta \tag{5.1}$$

where δ is the tracking step and it is a function of the integrator gain k_i

$$k_i = \frac{\delta}{T} \tag{5.2}$$

and T is the duration of the pulses produced by the pulse generator block. The parameter T is designed according to the predicted input signals, system size and overall configuration because these characteristics determines the channel communication activity.

The communication channel may be overloaded when the activity of all coders exceed the channel capacity. In this implementation, the spike generator sets a minimum period for the interval between two successive output spikes as a preventive method to avoid this condition. This “refractory period” is given by

$$\Delta t_{(min)} = \frac{1}{f_{sk(max)}} - T. \tag{5.3}$$

Setting the period as a multiple of the spike “width” $\Delta t_{(min)} = k\,T$ and using the specification of the maximum derivative of the input, the spike width is determined from

$$T = \frac{\delta}{(k+1)\,|\dot{x}(t)|_{(max)}}. \tag{5.4}$$

The definition of $\Delta t_{(min)}$ also provides an estimative about the limitations imposed to the input

[1]For this application, where speed in most crucial.

signal. From equation 5.4 the frequency of an input sine wave $x(t) = A\ \sin(\omega_{\ in}t)$ is

$$f_{in} = \frac{\omega_{in}}{2\pi} = \frac{1}{2\pi}\frac{\delta}{A}f_{sk(max)} = \frac{1}{2\pi}\frac{1}{2^{N_b}}f_{sk(max)}. \tag{5.5}$$

In other words, once the system parameters are defined, the maximum input frequency is inversely proportional to the signal amplitude. Whenever the input frequency is greater than the value defined by equation 5.5, the system will present slope overload [92]. The slope overload refers to the maximum rate that the coder can update the feedback signal $z_R(t)$.

Finally, the pole of the decoder first order low pass filter (LPF) is a key design parameter as it improves the resolution by attenuating the undesirable out-of-band high frequency harmonics generated during the decoding process.

5.3 Circuits design

The implementation of communication scheme in figure 5.1 makes use of a small number of circuits to implement each block: comparator, spike generator and integrators. In the next subsections, design methodology is presented in more detail for each of these circuits. The decoder low-pass filter was not implemented on-chip, because it is an optional part of the circuit which increases the method resolution. Therefore an offline, software-based implementation was used instead.

These ICs were designed using a 3.3V, 4 metals, single poly, 0.35μm CMOS process. The voltage domain was used to represent the analogue signals involved in the communication process. This choice would allow an easier integration with the CAB proposed in [64].

5.3.1 Comparator

In the block diagram presented in figure 3.4(a), signal subtraction and comparison functions are performed by different blocks, but they are implemented as a single circuit: the comparator.

The design of comparators can be implemented using a pre-amplifier (PA) followed by a decision circuit (DC) and an output buffer (OB) [136] as shown in figure 5.2. The pre-amplifier circuit provides a gain on the input signal enough to reduce the impact of the mismatch on the decision circuit, which outputs a digital-like transition signalling the comparison. Finally the

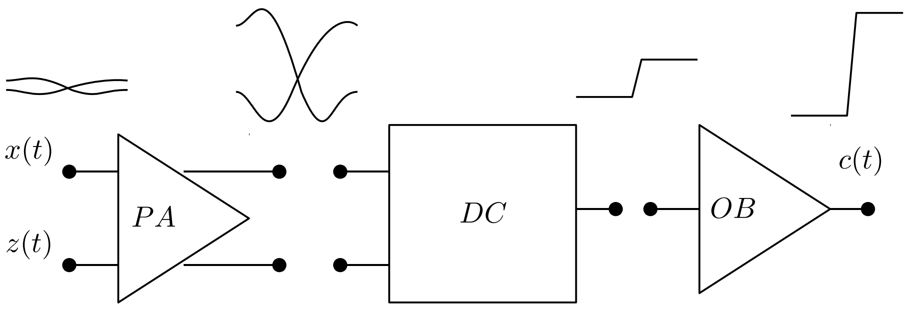


Figure 5.2: *Block diagram of a typical comparator. The input signals difference is magnified by the Pre-Amplifier (PA) and then used by the Decision Circuit (DC) to provide a comparison signal. The Output Buffer (OB) keeps the transition times independent of the load.*

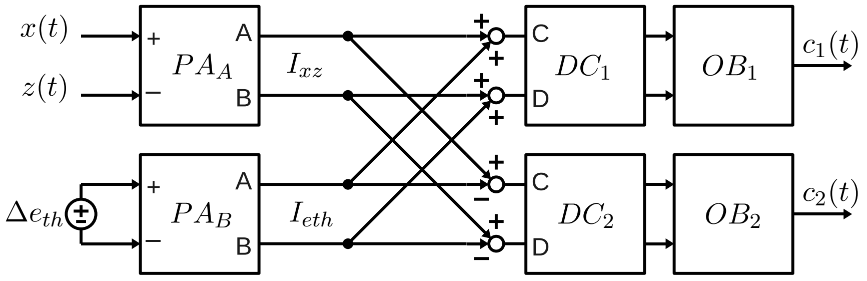


Figure 5.3: *Block diagram of the compound comparator. Pre-amplifier PA_A outputs a current ΔI_{xz} according to the inputs $x(t)$ and $z(t)$, whilst pre-amplifier PA_B generate a fixed offset current I_{eth}. PA_B outputs are added or subtracted from PA_A outputs and the results are applied to the respective decision circuits (DC) and output buffers (OB).*

output buffer provides the current needed to keep the rising and falling times short for any load.

To provide the required Δe_{th}, capacitive or resistive dividers can be used at the comparator input nodes. However, these dividers compromise the input impedance of the circuit [137].

Another method to provide Δe_{th} is to implement offset comparators. Composite transistors can be used to provide this difference [137], however this topology suffers from low input dynamic range. A programmable offset can also be generated by another pre-amplifier which provides a respective I_{eth} on the decision circuit input [138]. Both impedance dividers and offset comparators allow the use of continuous values for Δe_{th}.

In this implementation both outputs $c_1(t)$ and $c_2(t)$ are generated by a compound comparator as in figure 5.3. Instead of using four pre-amplifiers with two sensing the inputs $x(t)$ and $z(t)$ and two providing different offsets (two sets for each output), only two pre-amplifiers are needed.

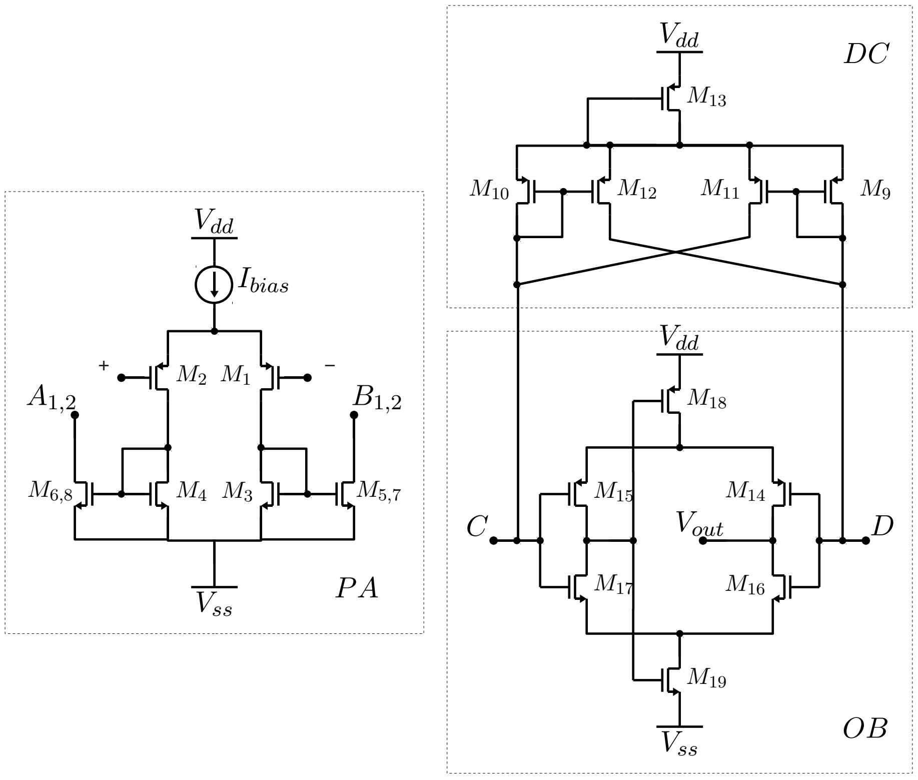


Figure 5.4: *Circuit schematic of each compound comparator blocks. Transconductance preamplifier (PA, left), decision circuit (DC, top right) and output stage (OB, bottom right).*

The pre-amplifier PA_A outputs a differential current $\Delta I_{xz}(t)$ as the result of the amplification of the difference between $x(t)$ and $z(t)$, i.e., the error signal $e(t)$ in figure 3.4(a). The capacitive load on the input nodes $x(t)$ and $z(t)$ is reduced when only one pre-amplifier is used. The other pre-amplifier (PA_B) provides a differential current I_{eth} according to voltage Δe_{th} on its inputs.

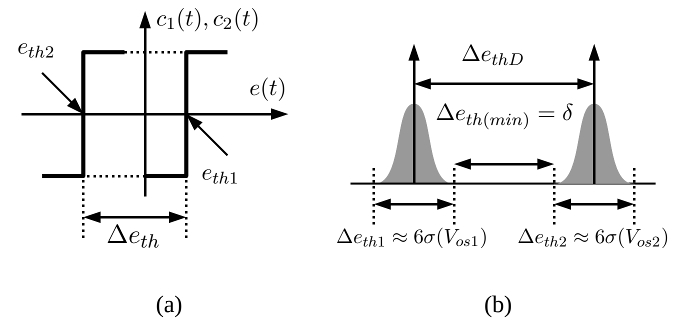
The results of adding ($I_{xz} + I_{eth}$) and subtracting ($I_{xz} - I_{eth}$) these currents are forwarded to the decision circuits to speed up the comparison result. Finally, output buffers generate the digital outputs.

Figure 5.4 presents the pre-amplifier, the decision and the output buffer schematic circuits implemented on IC. The pre-amplifier is a transconductance amplifier with two identical differential output currents at nodes (A_1, B_1) and (A_2, B_2). The decision circuit is a positive feedback circuit and the output buffer is a self-biased amplifier [139].

The transistor sizes for this implementation are presented in table 5.1. The comparator was designed to provide a hysteresis smaller than the tracking step to help avoiding excessive

Transistors	**Size** [μm / μm]
$M_1 - M_2$	20.0 / 1.0
$M_4 - M_3$	6.0 / 0.8
$M_5 - M_8$	6.0 / 0.8
$M_9 - M_{10}$	2.0 / 2.0
$M_{11} - M_{12}$	1.8 / 2.0
M_{13}	10.0 / 0.35
$M_{14} - M_{17}$	2.0 / 0.35
M_{18}	2.0 / 2.0
M_{19}	0.8 / 2.0

Table 5.1: *Transistor sizes of the compound comparator.*



$c_1(t), c_2(t)$
e_{th2}
$e(t)$
e_{th1}
Δe_{th}
(a)
Δe_{thD}
$\Delta e_{th(min)} = \delta$
$\Delta e_{th1} \approx 6\sigma(V_{os1})$
$\Delta e_{th2} \approx 6\sigma(V_{os2})$
(b)

Figure 5.5: *Comparator thresholds design and mismatch. (a) Comparator transfer functions and (b) comparator thresholds variations Δe_{th1} and Δe_{th2} used to calculate the designed threshold difference Δe_{thD}.*

switching due to noise. This hysteresis is generated with the size difference between transistors $M_9 - M_{10}$ and $M_{11} - M_{12}$.

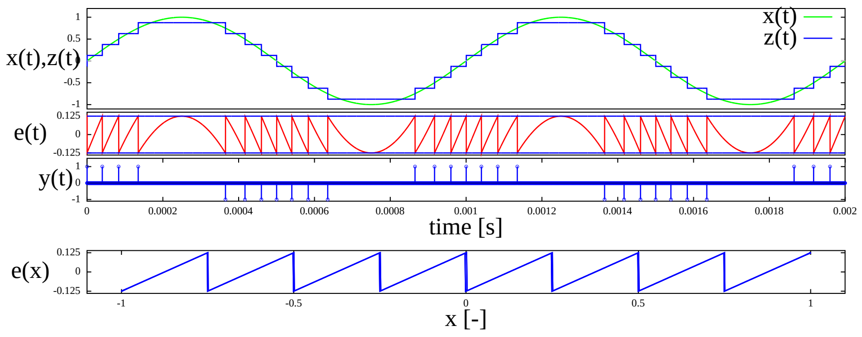
Non-idealities

For an optimum performance, the tracking step δ generated at the integrator output equals to difference between the thresholds Δe_{th} of the comparators. However, due to process mismatches [140], comparators offsets V_{os1} and V_{os2} may vary from the *designed* value Δe_{thD}, as shown in figure 5.5(b). Hence, the *actual* Δe_{th} is bounded, for a 6σ variation (99.7%), by

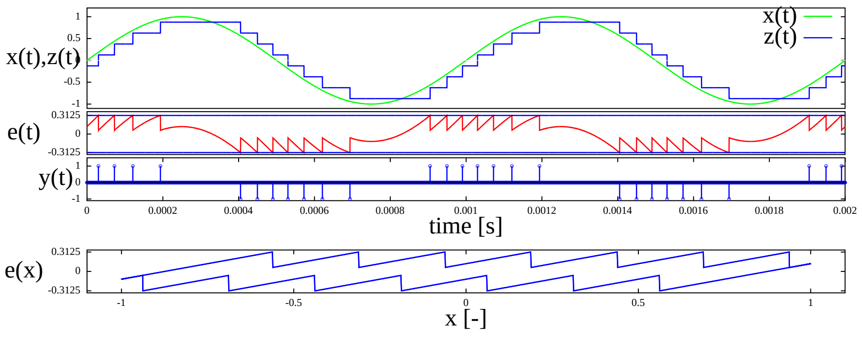
$$\Delta e_{thD} + 3\sigma_{os} \geq \Delta e_{th} \geq \Delta e_{thD} - 3\sigma_{os} \tag{5.6}$$

where $\sigma_{os} = \sigma(V_{os1}) + \sigma(V_{os2})$ for uncorrelated variables and $\sigma(V_{os1})$ and $\sigma(V_{os2})$ are the standard deviations of the comparator offsets V_{os1} and V_{os2} respectively.

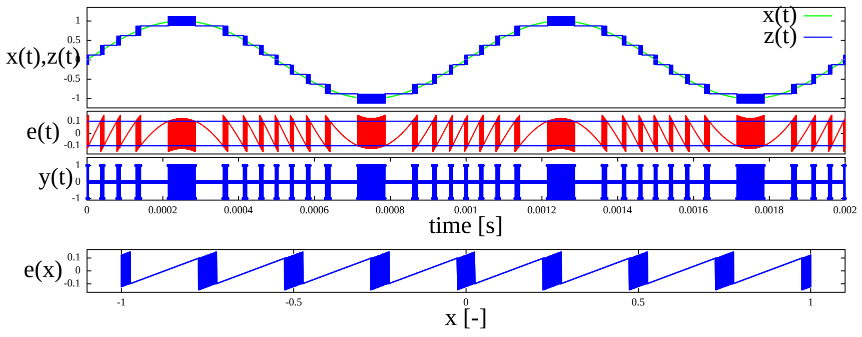
Therefore a design margin is required because of the random offsets due to process variations.



(a)



(b)



(c)

Figure 5.6: *Comparator model simulation with mismatch. TSD model results for 3-bit resolution with different mismatches between tracking step δ and comparator thresholds difference Δe_{th}. (a) $\Delta e_{th} = \delta$, (b) $\Delta e_{th} = 2.5 * \delta$ and (c) $\Delta e_{th} = 0.8 * \delta$.*

For instance, if $\delta < \Delta e_{th}$ the feedback signal $z(t)$ is delayed and distorted, in particular close to the signal inflection points. Conversely, $z(t)$ will oscillate for $\delta > \Delta e_{th}$. The comparators thresholds difference is designed to meet the following safety margin

$$\Delta e_{thD} \geq \delta + 6\sigma_{os} \tag{5.7}$$

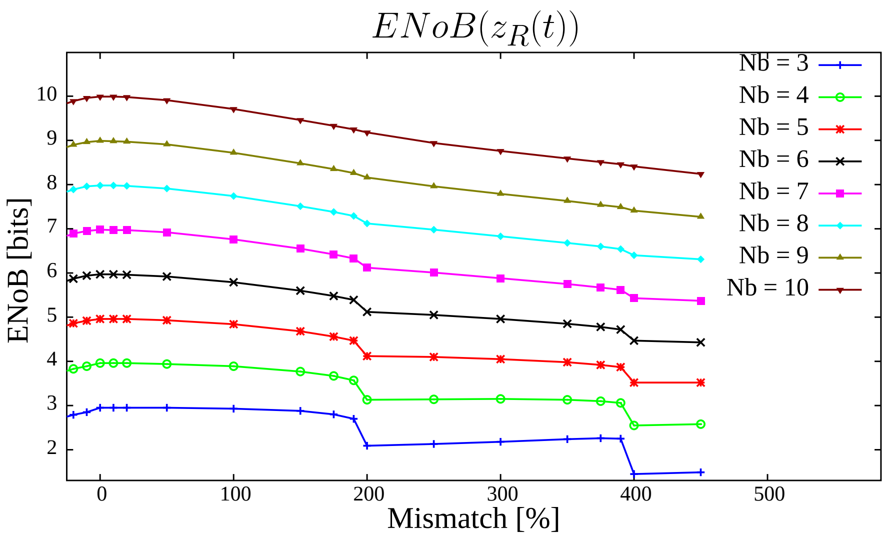


Figure 5.7: *Impact of comparator mismatch on the resolution. Simulation results for TSD model with different mismatches between tracking step δ and comparator thresholds difference Δe_{th}. Measured ENoB in $z_R(t)$ signal.*

Figure 5.6 illustrates the effects of this mismatch when Δe_{th} is equal, greater and smaller than δ. This results were obtained from the TSD model simulation results for a 3-bit resolution. Figure 5.7 shows the influence of this mismatch type on the resolution.

5.3.2 Spike and pulse generators

The spike generator block can provide either a positive or a negative spike according to the output state of the comparator. When the error $e(t) > \Delta e_{th}/2$, a positive spike is transmitted. Similarly, a negative spike is generated when $e(t) < -\Delta e_{th}/2$. Otherwise, no spikes are transmitted. From these spikes correspondent pulses are generated both at coder and decoder. Figures 5.8(a) and 5.8(b) are the block diagrams of the spike and pulse generators on the coder and the pulse generator on the decoder respectively. Figures 5.8(c) and 5.8(d) show an theoretical example of the behaviour of the control signals.

The arbiter senses which comparator outputs $c_1(t)$ and $c_2(t)$ changed first. It avoids further interferences, such as noise, on these signals before the update cycle of the signal $z(t)$ has finished. Arbiter output triggers the spike generator which starts the handshaking communication with the AER arbiter setting the *req* signal. On receipt of the *ack* signal, the pulse generator is activated and provides a pulse *inc* or *dec* to the integrator according to the arbiter output.

The pulse generator block includes programmable delay circuits for the generation of T and

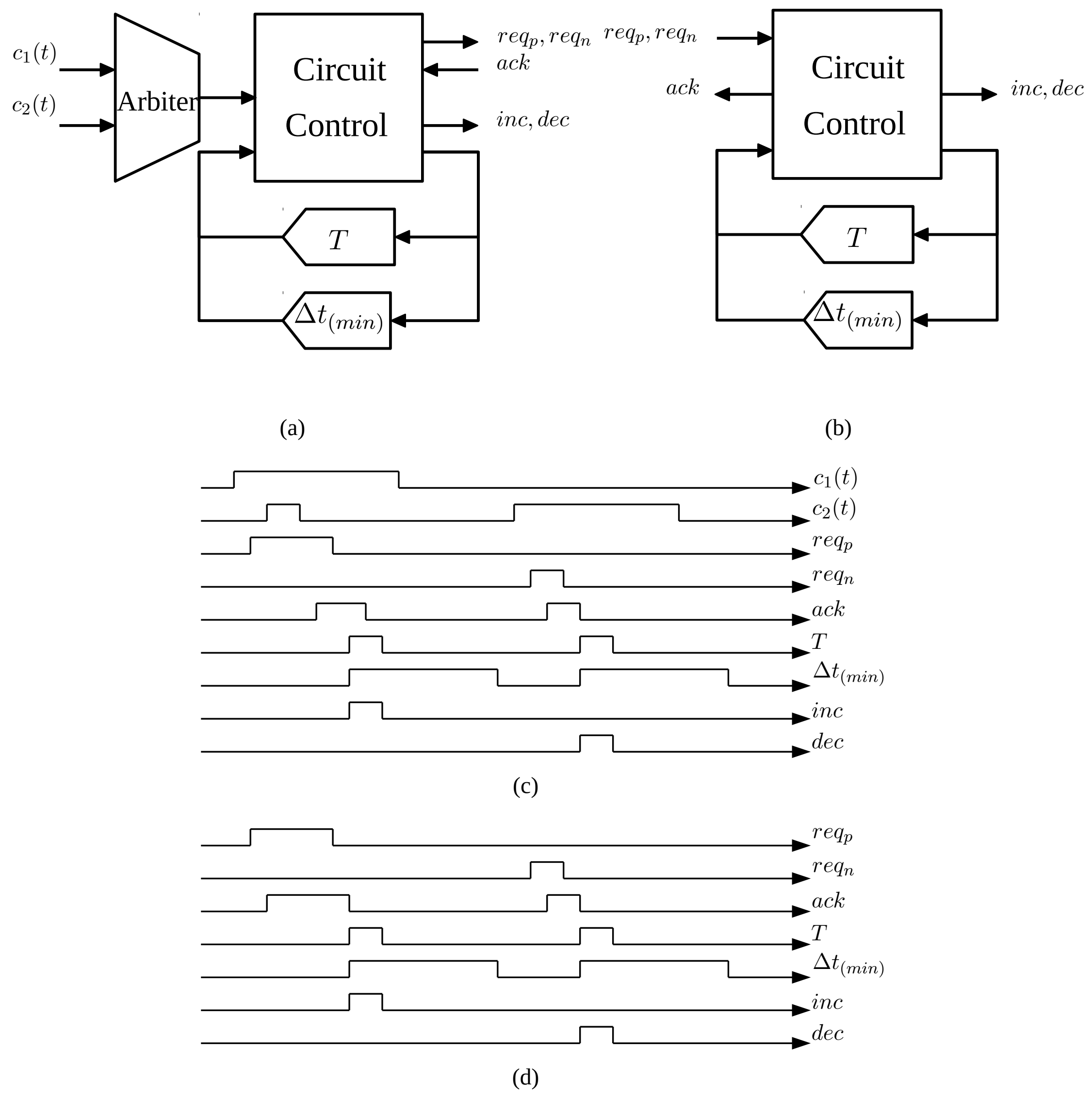


Figure 5.8: *Spike (SG) and pulse (PG) generators. Coder combined spike and pulse generators (a) and decoder pulse (b) generator block diagrams. (c) is an theoretical example of timing diagram of the block in the coder while (d) shows timing diagrams on the decoder. In the coder an arbiter selects between $c_1(t)$ and $c_2(t)$ inputs and starts the handshaking signalling. After the handshaking is completed, either a inc or dec pulse is generated using delay blocks for T and $\Delta t_{(min)}$.*

$\Delta t_{(min)}$ time intervals. These blocks were implemented as a current integrator feeding a chain of inverters. This is a suitable implementation although more efficient methods have been available [141]. The control logic for this circuit was implemented using a technique developed to design asynchronous digital circuits [142] and showed in appendix E. Although the circuit in [107] presents simpler hardware, it lacks noise tolerance provided by the arbiter.

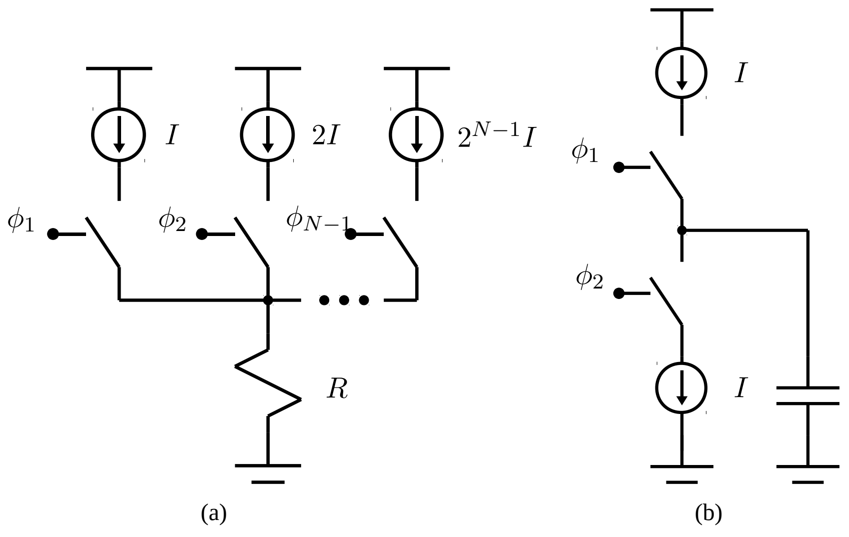


Figure 5.9: *SCI examples. (a) Unipolar version as used in current-steering DACs and (b) bipolar version used as used in signal modulation.*

5.3.3 Integrator

The integrator is the last block remaining to be analysed. Together with the comparator threshold difference Δe_{th}, its gain k_i defines the resolution of the coder.

In the first IC, an integrator based on the switched current integration (SCI) technique [124] was designed. This implementation, used in charge pump circuits, was also used in other modulation schemes [91, 124] and a unipolar version of this type of circuit driving resistors is used in steering current cells of some Digital-to-Analogue Converters (DACs) [143] as shown in figure 5.9.

The schematic of the implemented SCI circuit is presented in figure 5.10 and it works as follows. When a negative spike arrives at the integrator input, the dec signal goes high for an interval T, allowing currents $1.5I$ and $0.5I$ to flow in transistors M_{22} and M_{23}, respectively. Similarly, for the case of a positive spike arrival, transistors M_{20} and M_{21} provide symmetrical operation with inc and $\overline{inc}$ signals. Therefore, the resulting current I that discharges or charges the integrating capacitor C_{int} is given by

$$I = C_{int}\,\frac{\delta}{T} \tag{5.8}$$

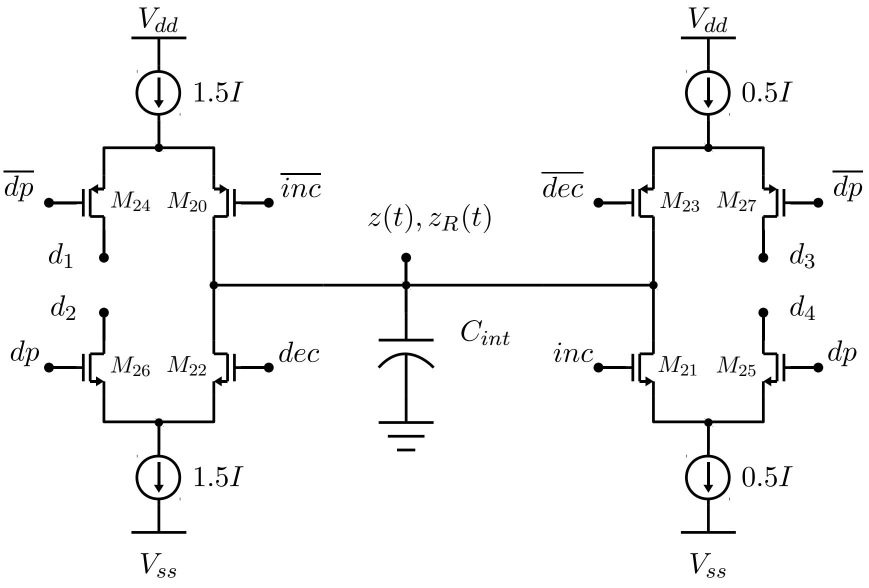


Figure 5.10: *Schematic of the integrator based on a SCI technique. When a positive spike arrives, M_{20} and M_{21} turn on and $1.5I$ and $0.5I$ charges and discharges the capacitor C_{int}, respectively, resulting in a voltage increment of δ given by (5.8). For negative spikes, the complementary process decreases the capacitor voltage. In the absence of spikes, currents are drawn to low impedance nodes ($d_1 - d_4$). The schematic of the delay generators is shown in appendix E.*

From equation 5.2 the designed integrator gain may be obtained as

$$k_i = \frac{I}{C_{int}}. \tag{5.9}$$

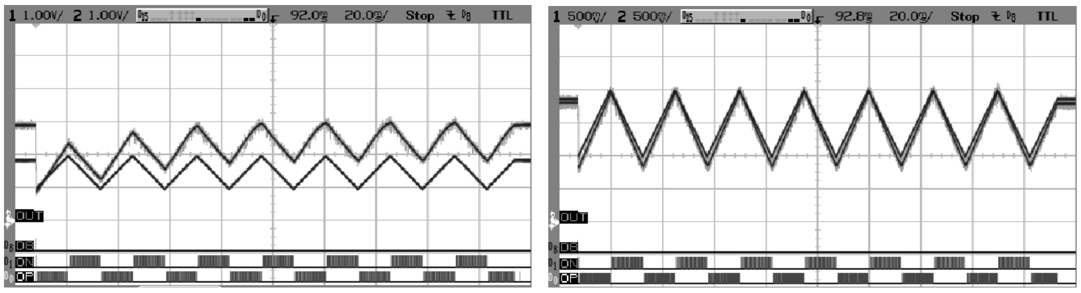
When there are no spikes from the spike generator, currents are driven to low impedance nodes ($d_1 - d_4$) through $M_{24} - M_{27}$ by setting the signal dp high.

Another design approach would be switching on and off the current mirror itself instead of driving the current to low impedance nodes. This solution would result in a lower power consumption but switching the voltage reference at the gate terminal would also result in temporary errors that would corrupt the tracking step amplitude.

The use of two different branches (M_{22} and M_{23} or M_{20} and M_{21}) to both charge and discharge the capacitor reduces charge injection on the integration nodes $z(t)$ and $z_R(t)$ [144] at the cost of doubling the power consumption required. If switches M_{20} and M_{21} (M_{22} and M_{23}) have the same dimensions, the charges injected from the gate to drain capacitance of the complimentary switches cancel each other.

Component	**Size**	**Unit**
Current Sources	20.0 / 2.0	μm / μm
Cascode Curr. Sources	10.0 / 0.35	μm / μm
Current Sinks	20.0 / 2.0	μm / μm
Cascode Curr. Sinks	10.0 / 0.35	μm / μm
$M_{20} - M_{27}$	1.5 / 0.35	μm / μm
C_{int}	1	pF

Table 5.2: *Components sizes of the SCI based integrator.*



(a) (b)

Figure 5.11: *SCI divergence. (a) Decoder integrator output with inc and dec swing set to original values 1.65V and (b) Decoder integrator output with inc swing reduced to 0.9V.*

Table 5.2 presents transistors and capacitor sizes designed for this work. The transistors of the current sources and sinks are the most important considering the circuit mismatch, and therefore their sizes were appropriately chosen to minimize this effect.

According to (5.8), the tracking step δ_d is a function of the bias current I. Therefore the gain operation described in 4.3.1 is implemented by setting different bias currents I for coder and decoder integrators.

Non-idealities

According to equation 5.8, the integrator based on the SCI technique presents three different source of mismatches between coder and decoder.

The first is in the pulse generation by the time delays in the pulse generator block. Because of process variations, the current source, the capacitor and the inverters threshold can vary and then provide different pulse width T. The second mismatch source is due to the size of the capacitor C_{int} in each integrator. The third is the mismatch between current sources that provide the currents $1.5I$ and $0.5I$ in figure 5.10. Moreover the mismatch between current sources and current sink may cause a difference between the increase and decrease tracking steps. Cascoded current mirrors were used to reduce the current mismatch.

The resulting difference may impact the functionality of the scheme. For instance, whilst the coder integrator will still track the input signal, the decoder integrator might present a different behaviour, normally saturating at either of the power supplies.

A solution for this erratic behaviour was found by changing the amplitude of the inc and dec signals. Normally, these signals presents a full or half-scale amplitude[2]. By changing the ON state amplitude, for instance limiting the dec signal amplitude to a value V_{dec} closer to M_{22} threshold voltage, we can force the current mirror output transistor, which provides the $1.5I$ current, to leave the saturation region and to enter into the linear region, thus reducing the effective mirrored current. The IC provides a mechanism to change this amplitude. This calibration mechanism can thus compensate for all 3 mismatches sources.

Figures 5.11(a) and 5.11(b) shows the decoder behaviour before and after this solution. In the former both inc and dec signals swing amplitude is 1.65V as designed originally whilst in the last the amplitude swing for the signal inc was reduced to 0.9V.

5.3.4 Filter

Filter provides the averaging function in equation 3.4 by removing high component frequencies. Ideally, the filter should provide total rejection of out-of-band harmonics with zero in-band attenuation. However, the practical implementation of this characteristic is unrealisable. The actual design of the filter is a trade-off between the amount of harmonic reduction and distortion caused by the phase shift of each component frequency. In practice, the cut-off frequency of the low pass filter ω_p is designed to be greater than or equal to the input signal bandwidth.

The filter is also a key factor for the delay between the coder input $x(t)$ and the decoder output $x_R(t)$. For a sine wave, choosing the pole to be $\omega_p = \omega_{in}$, the filter imposes a 45° phase shift from the signal $z_R(t)$. When applying low-frequency input signals results in the delay due to phase shift being greater than delays due to the modulator loop, $\Delta t_{(min)}$, and AER arbitration process. For instance, the conversion of a tone signal ($\omega_{in} = 2\,\pi\ 4$ kHz) using 8-bit resolution imposes $\Delta t_{(min)} \approx$ 300ns $\ll$ 31.2μs (45° phase shift).

[2]In this implementation, the control signals were originally designed for half-scale voltage amplitude, i.e., $\overline{inc}$ and $\overline{dec}$ swing from ground to V_{dd} while inc and dec from ground to V_{ss}.

5.3.5 AER communication

The AER communication method was introduced in the second IC to fully implemented the asynchronous spike event coding scheme. Although the arbitration process was implemented inside the IC, the routing control was implemented in an external FPGA (Xilinx XC3S1000-4FG320). Therefore this IC presents two sets of control signals (request and acknowledge) and addresses bus: one set from the IC to the FPGA and the other in the opposite direction.

5.4 IC results

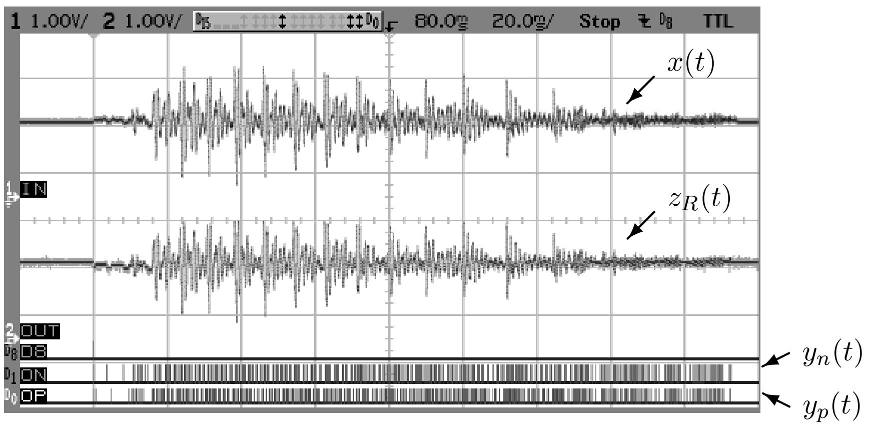
In this section IC results to demonstrate the communication aspects of the event coding scheme are presented. Two different input signals were used to test the communication system: a speech signal and a single tone (sine wave) signal. Measured IC results for the computation capabilities of the communication scheme presented in chapter 4 are shown as well.

5.4.1 Demonstration: speech input

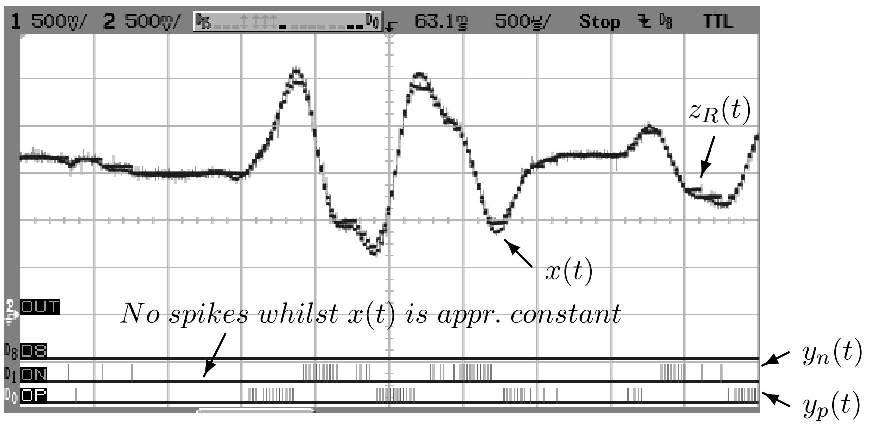
A short speech signal of about 140ms was used to demonstrate the coding functionality. This signal was originally sampled at 44.1 kSps with 8-bit resolution. The coding system was programmed to provide a resolution of 4 bits in order to demonstrate the coding properties.

Figure 5.12(a) was obtained from the first IC and shows the coder input signal $x(t)$ (top), the decoder integrator output $z_R(t)$ (middle) and the coder output spikes $y(t)$ (bottom, with the negative spikes first). The same signals are presented in detail in figure 5.12(b) where $x(t)$ and $z_R(t)$ are overlapped. This figure demonstrates the asynchronous nature of the communication scheme and the absence of coder output spikes when the signal is constant or when its change is smaller than Δe_{th}.

The figure 5.13(a) is the test with the second IC. With the AER implemented in this IC, this figure shows the request, acknowledge signals and both inbound and outbound spike addresses. It is also possible to notice the difference between the 8-bit resolution input signal $x(t)$ and the 4-bit resolution signal $z_R(t)$.



(a)



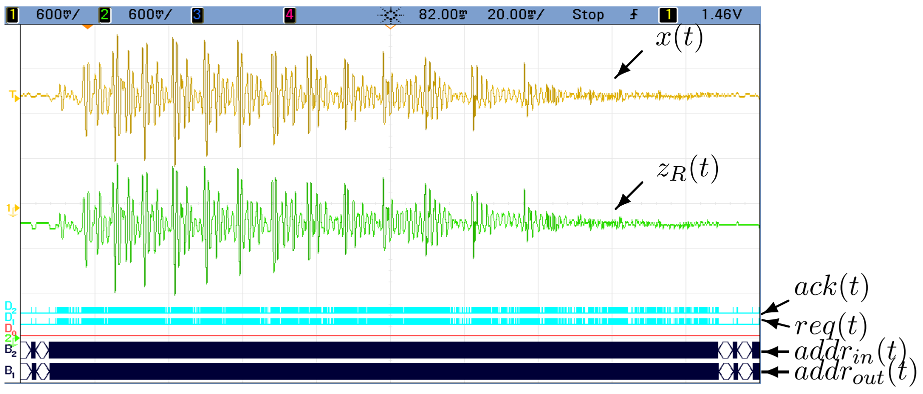
(b)

Figure 5.12: *Demonstration with an audio signal for the first implementation. (a) Coder input $x(t)$ and decoder integrator output $z_R(t)$ for a 2.5Vpp speech signal. Negative $y_n(t)$ and positive $y_p(t)$ output coder spikes are also shown at the bottom of the figure. (b) A detailed view of the same waveforms to show the absence of spikes during the periods when the input signal $x(t)$ is approximately constant, i.e., it changes less than the coder resolution.*

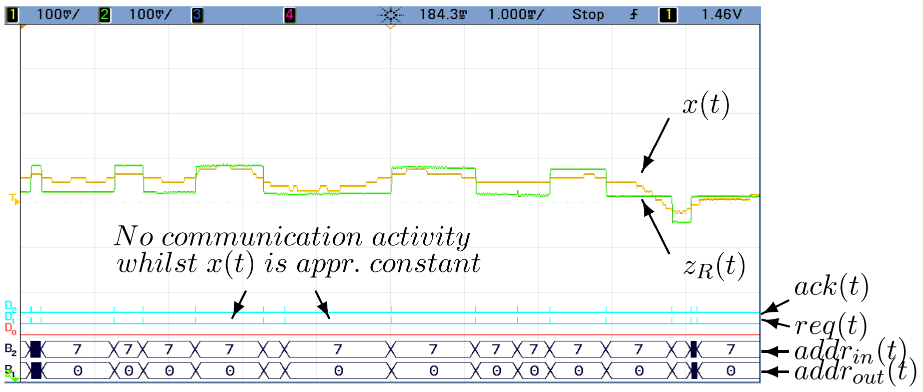
5.4.2 Resolution: single tone input

The resolution of the system was measured using a sine wave input signal with an amplitude of 2.0 V_{pp} and frequency of 20 Hz (f_{in}). The sine wave was sampled at 44.1 kSps and the coder was programmed to provide a 4-bit resolution. A snapshot of the input and output signals is presented in figure 5.14(a). Offline filtering results are shown in the figure 5.14(b) for a digital LPF with a cut-off frequency of 20 Hz, the same frequency as the input signal.

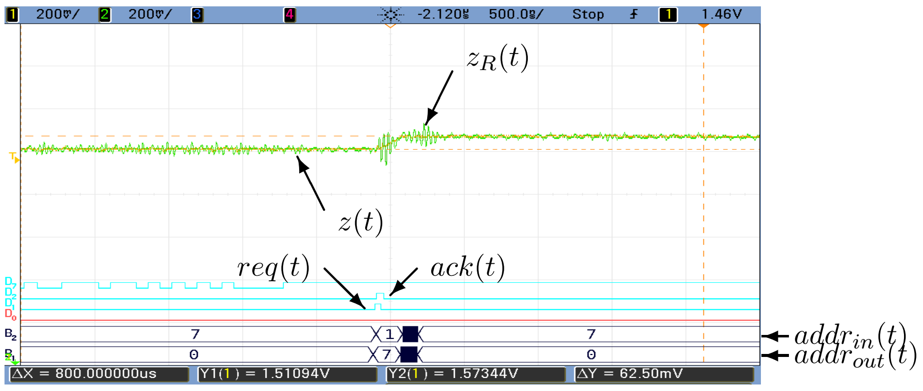
The total harmonic distortion (THD) and resolution of the system was measured using this sine wave input signal. The measured THD of the pre-filter signal is -26 dB which corresponds to



(a)

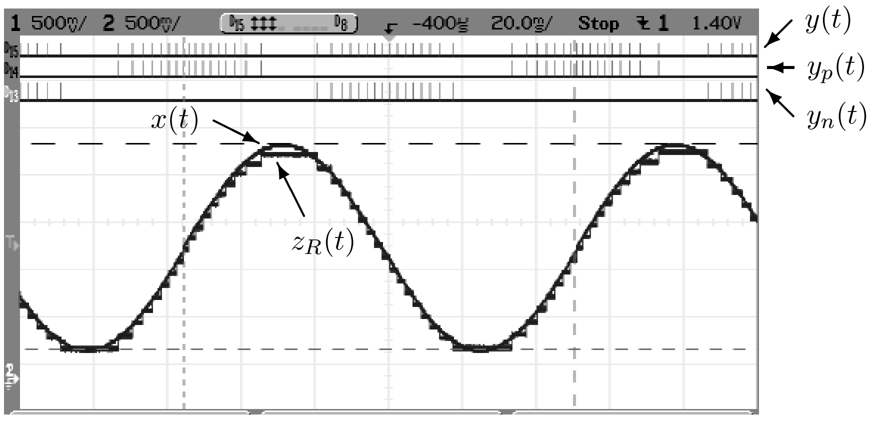


(b)

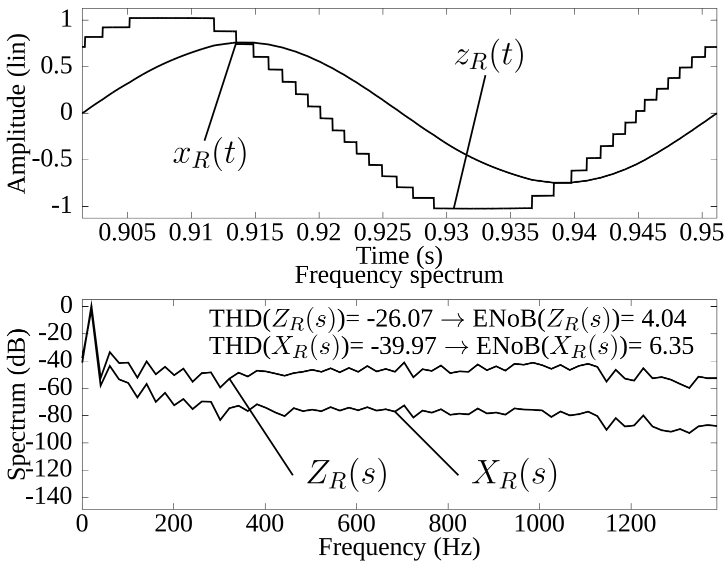


(c)

Figure 5.13: *Demonstration with an audio signal for the second implementation. This figure presents the results from the second chip using the same test set-up of figure 5.12. Similar results were achieved but instead of the spikes generated, it shows the handshaking AER signals (request, acknowledge and spike addresses). In (b) is clear the difference between the 8-bit resolution input $x(t)$ and the 4 bit resolution output $z_R(t)$. In $addr_{out}(t)$, the absence of spikes is represented by the address 0 whilst the same condition is represented in the $addr_{in}(t)$ bus as 7. (c) shows the internal signals $z(t)$ and $z_R(t)$ and the AER control and address buses at the handshaking instant, when a positive spike transmission occurs.*



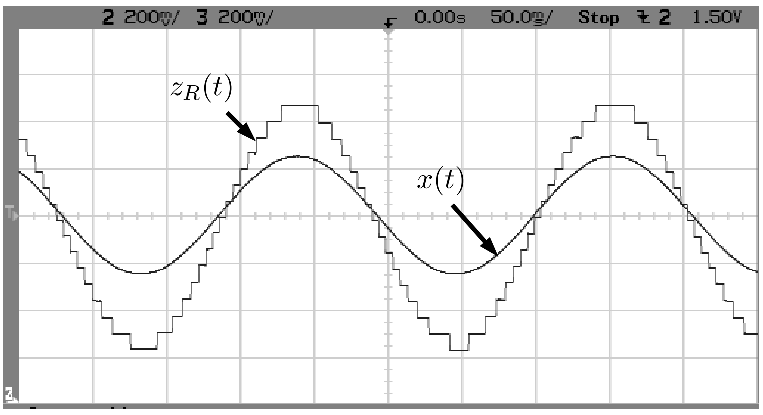
(a)



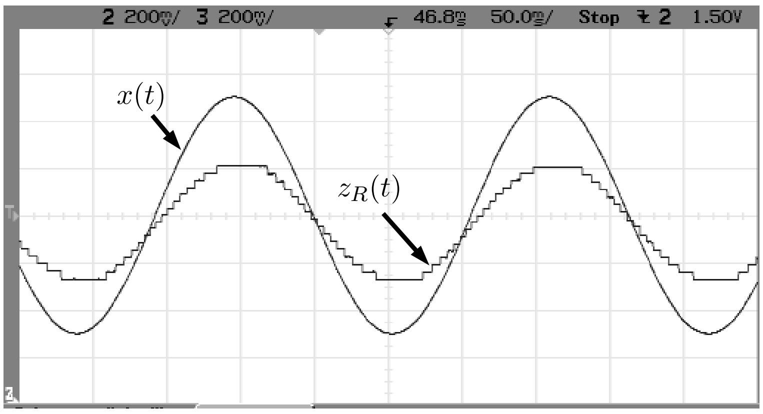
(b)

Figure 5.14: *Resolution of the coding of the first implementation. (a) Oscilloscope snapshot with the input $x(t)$ (2.0V_{pp} sine wave), decoder integrator output $z_R(t)$, negative $y_n(t)$ and positive $y_p(t)$ spikes from the coder output. (b) Reconstructed $z_R(t)$ from ADC data and decoder output $x_R(t)$ from the software filter (top). The low pass filter cut-off frequency is the same as the input signal. The frequency spectrum $Z_R(s)$ and $X_R(s)$ of the filter input and output (bottom) computed.*

a measured resolution of 4.04 bits. The measured THD of the post-filter output, with a filter cut-off frequency equal to f_{in}, improves to -40 dB which corresponds to a measured resolution of 6.35 bits. This resolution improvement causes attenuation and phase shift of the signal. When the cut-off frequency is increased to 10 f_{in}, the measured resolution is equal to 4.93 bits. The low pass filter cut-off of f_{in} yields better resolution attributable to greater attenuation of



(a)



(b)

Figure 5.15: *IC results for the gain operation. The input $x(t)$ is simular to the one in figure 4.2. 5.15(a) is the illustrates the amplification while 5.15(b) shown the results when the ASEC is configured to perform signal attenuation.*

harmonics due to the lower frequency pole of the low pass filter. Resolution is limited by mismatch in the circuit implementation and may be improved by further work.

5.4.3 Computation: gain, negation, summation and bpsk

Computation properties of the ASEC were explained in chapter 4. These explanations were illustrated with software simulations of mathematical models. In this chapter IC results are presented for some of the those computations. Figure 5.15 corresponds to the gain operation while figures 5.16 and 5.17 correspond to negation and summation (and subtraction) operations. Figure 5.18 is the IC result for the BPSK. Similar inputs from the simulation were used for the IC results.

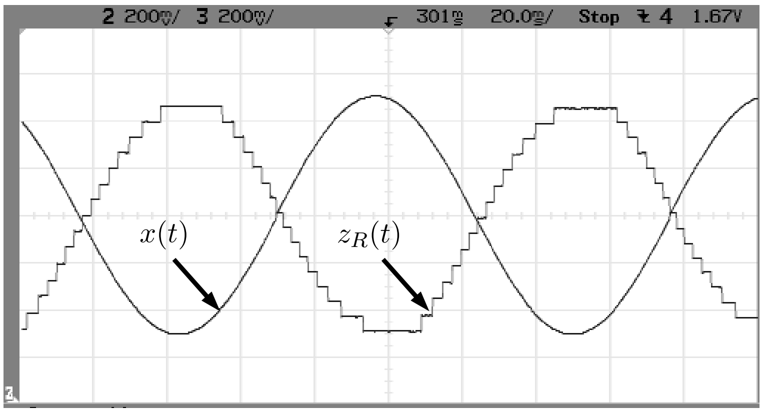
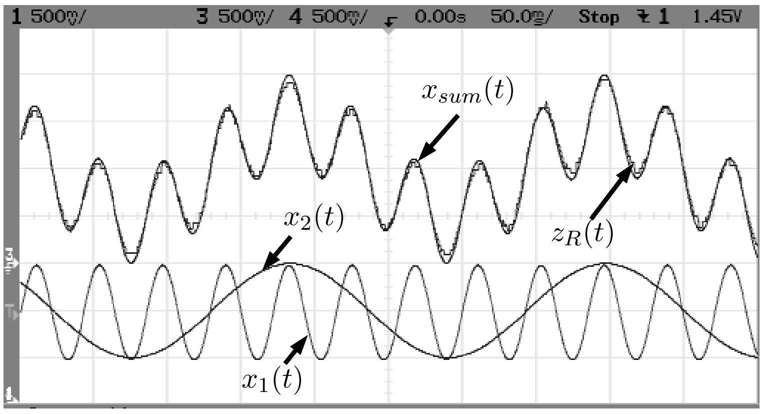
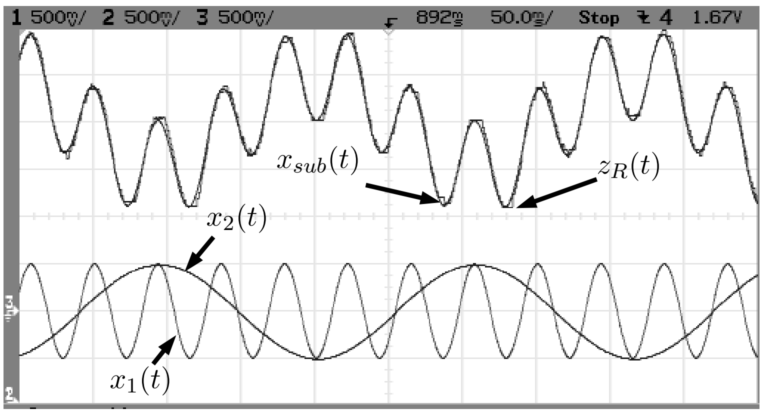


Figure 5.16: *IC results for negation operation. The ASEC circuits were configured to produce an negated version of the sine wave $x(t)$, as suggested in 4.3.2 and figure 4.3.*



(a)



(b)

Figure 5.17: *Summation and subtraction operations results from chip. This figures present chip results similar to the simulation results in figure 4.6 where (a) show the summation $z_R(t)$ of two input sine waves, $x_1(t)$ and $x_2(t)$, with same amplitude but different frequencies (23Hz and 4.7Hz) and $x_{sum}(t)$ being the ideal summation. The subtraction of the same input signals $x_1(t) - x_2(t)$ is shown in (b), where $x_{sub}(t)$ is the ideal result. Scope DC levels were shifted to better illustration.*

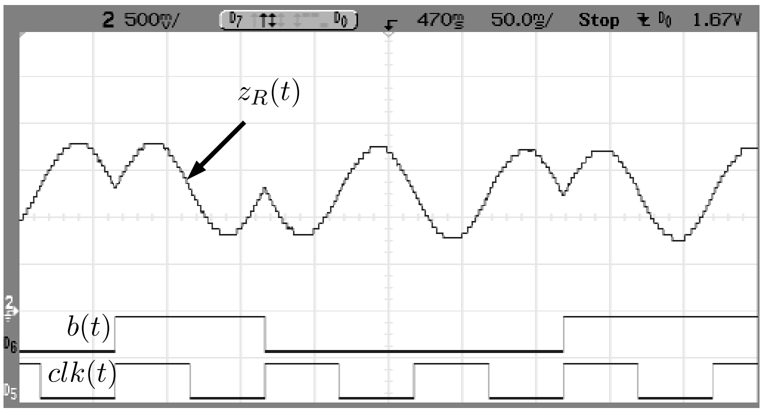


Figure 5.18: *Results of the binary phase shift keying operation from IC. In this snapshot, the input digital word $b(t)$ to be modulated by the method is 010011. $clk(t)$ frequency is the same of the sine carrier (not shown).*

5.4.4 Area and power consumption

As said before, the ASEC coder and decoder were layout in a 0.35μm CMOS technology process. Figure 5.19 shows the snapshot of the ASEC coder layout. The main blocks described earlier were highlighted in the figure to illustrate the size proportion of the blocks. The layout of the decoder is similar with the exemption of the comparator and some minor differences in the digital control.

Figure 5.20 and 5.21 show photographs of the first and second designed test IC respectively. The total area of the second IC and the area of each block is presented in table 5.3 together with the total power consumption and that for each block. All area measurements and power consumption refers to the second test IC with the exceptions of the individual power consumption per block.

5.5 Summary

A VLSI implementation of the spike event coding scheme was presented in this chapter. The steps and constraints involved in the scheme design were explained and detailed.

The communication scheme presents a simple topology with few functional blocks: comparator, integrator and digital communication. Well known topologies were used to design these blocks including a compound comparator to avoid excessive input capacitance loads and asynchronous-based digital design to the AER handshake interface.

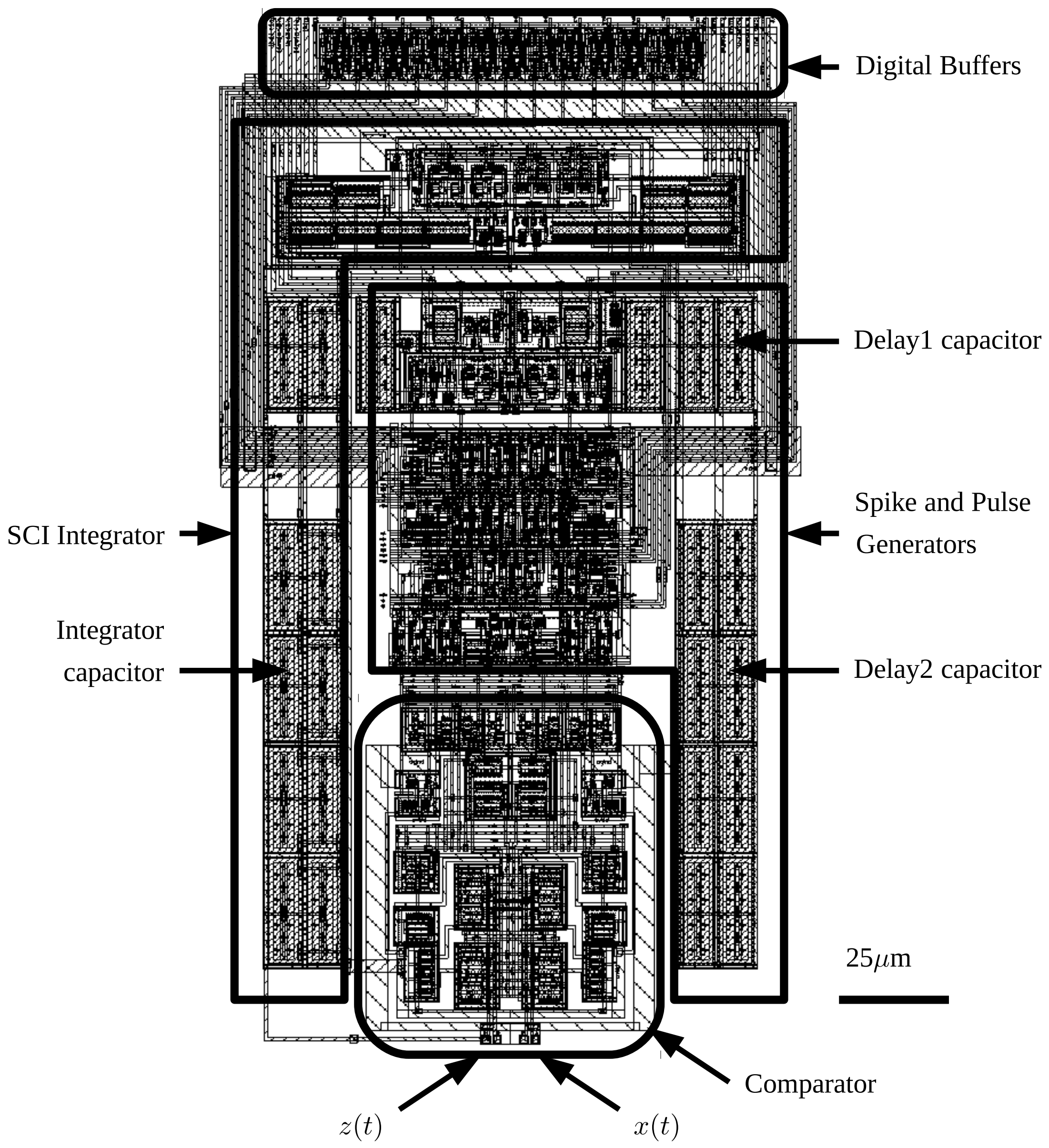


Figure 5.19: *ASEC coder layout. This figure shows the layout of ASEC coder with the main blocks highlighted. The coder occupies an area of approximately of 0.03 mm^2 (120μm x 240μm). ASEC decoder presents similar design excluding the comparator.*

A two-step approach was adopted to implement this VLSI design. In the first stage, the components were placed in a first IC to test each component separately and as the complete coder. The last stage, a larger IC including the AER features (arbiter and address coding/decoding) and a small array of eight coder and decoders, four with SCI based circuits and four with switched-capacitor versions.

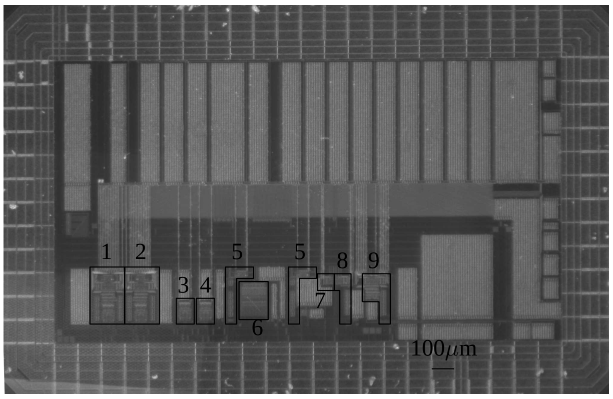


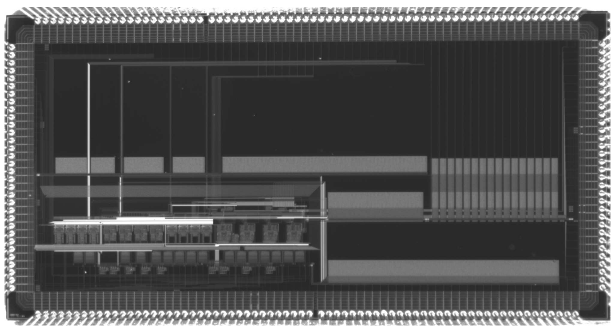
Figure 5.20: *First test chip photograph. The test chip area is 5 mm^2. The circuits are highlighted as: (1) coder with no hysteresis comparator, (2) coder with hysteresis comparator, (3) comparator with no hysteresis, (4) comparator with hysteresis, (5) SCI (integrator), (6) analogue buffer, (7) delay circuit for pulse width, (8) delay circuit for inter-spike period and (9) spike generator. The area of each coder is 0.03 mm^2 and each decoder (with no LPF) is 0.02 mm^2.*

Parameter		**Value**	**Unit**
Technology		CMOS 0.35μm 1P4M	-
Power Supply		3.3	V
Area	Coder	0.03	mm^2
	Decoder	0.02	
	AER	0.025	
	Circuitry	4.8	
	IC	29.25	
Power Consumption	Coder	400	μW
	Comparator	340	
	Decoder	50	
	IC	2700	

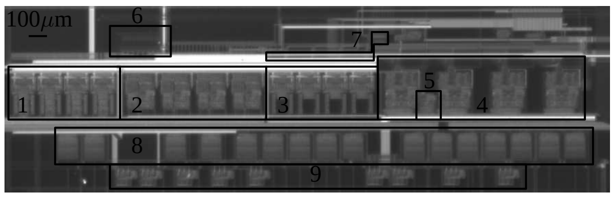
Table 5.3: *Test IC characteristics*

The functionality was tested with an audio sample signal and a tone signal to measure the decoder output resolution, with and without the output filter. This implementation of the ASEC is prone to mismatches both in comparator and in the integrator blocks and monte-carlo simulations were performed to verify the dimensioning of the circuits. Mismatch in the comparator results in an input offset which ultimately leads to input signal distortion.

In the integrators, mismatch effects result in different output values $z(t)$ and $z_R(t)$ for these blocks, being the most extreme condition when the decoder output tends to one of the power supplies compromising the whole functionality. As this effect is most unpredictable and undesirable, two different topologies were designed: one based upon SCI circuits and another based



(a)



(b)

Figure 5.21: *Second test chip photograph. The test chip (a) is pad-limited and the circuitry (b) occupies a quarter (5 mm^2) of the die. The highlighted areas are: (1) four SCI based coders, (2) four switched capacitor based coders, (3) four SCI based decoders, (4) four switched capacitor based decoders, (5) switched capacitor integrator, (6) AER arbiter, (7) AER receiver, (8) analogue buffers, and (9) current mirrors. The areas are presented in table 5.3.*

on a switched-capacitor techniques presented in Appendix F.

Although the SCI integrator presents more potential mismatch sources, the VLSI circuitry implemented in this work provided a calibration method to compensate the imperfection. Such compensation method is not available in the switched-capacitor topology and it was not possible to stabilized it.

In the next chapter, when discussing the next steps for this work, other possible solutions to solve the mismatch problem will be addressed.

Chapter 6
Summary and conclusions

In this last chapter, a summary of the work undertaken for this thesis is presented, followed by extra work research which would be interesting to carry on in the future. The chapter ends with final considerations and conclusions about the work performed.

6.1 Review of the thesis

The two main objectives of this thesis were introduced in the chapter 1: the use of a specific time-based method to transmit analogue information between the elements (CABs) of a programmable analogue array and the use of such method to perform computation over the communication channel with no hardware overhead.

The first step to accomplish these objectives was to review the programmable analogue arrays field in chapter 2. This review was useful to explain the reason why using timing rather than conventional voltage, current or charge representation is beneficial for this application. The main justification is centred on the fact that conventional methods are susceptible to several effects, like IR voltage drops, electromagnetic interference, cross-talk among others, which imposes heavy limitations about the scalability of the array.

In this chapter, a time-based alternative method — the Asynchronous Spike Event Coding (ASEC) scheme — was proposed as the *association* of the Ternary Spike Delta modulation (TSD) with the Address Event Representation (AER) communication method widely used in neuromorphic systems. TSD is the core of this method and its working principle was explained in chapter 3, together with other similar timing methods.

Besides presenting better communication characteristics — Channel Efficiency (CE) — for low resolutions than other spike modulations studied here, the ASEC presents very attractive computational properties as explained in chapter 4. In this chapter, it was demonstrated a set of arithmetic operations applied on analogue signals which are realizable with the proper programming of both TSD decoders and the AER routing map. In other words, no extra and specialized hardware is required. These operations include gain, negation, summation, modulus, mean, signal conversion (ADC and DAC) and others. More operations can be implemented if the computational properties of the ASEC is associated with the ones of the CABs, like multiplication and division.

Finally, in the previous chapter (5) presented the physical VLSI implementation of the ASEC scheme. The basic operational blocks of the TSD - comparators, spike generator, integrator and filter - were described in detail. To implement the dual comparator, a composite design was developed to allow programmable tracking steps keeping the input load the minimum. The spike generator is a purely digital circuit which is in charge of the AER handshaking signalling and the integrator pulse timings. The integrator was implemented as a switched current integration (SCI) circuit[1]. Although this circuit presents high sensitivity to process mismatches, the implemented version allows for external compensation mechanisms. And the filter was implemented as an off-chip, off-line first order digital filter.

Experimental results of two different ICs were presented in that chapter, with both functional and resolution test set-ups. While the first IC implemented only the TSD modulation, the second IC integrated AER channel too, implemented a full version of the ASEC scheme.

6.2 Further work

At the end of this thesis, some directions of future work can be suggested. The first group of suggestions are corrections and improvements on the work done and the second group refers to additional test and experiments to help in a better appreciation of the proposed method.

The first work to be considered is the redesign of test set-up. It was designed to make use of previous IMNS test set-ups and expertise. The test set-up was designed to use two or three printed-circuit boards. The first one is the FPGA evaluation kit (Opalkelly XEM3010) to implement the test control and measurement, the second is a redesign of a previous data acquisition board and the third was a daughter board to the second chip socket. Although the second board was designed to be the more generic possible, its size and complexity lends to great noise levels. The board noise path has to be re-evaluated and the number of active components shall be reduced. The daughter board, for instance, includes 56 DACs and 16 ADCs. Noise reduction

[1]An additional switched-capacitor based integrator was also designed and is presented in the appendix F

is paramount to correctly determine the maximum possible resolution of the ASEC scheme.

One method to reduce the number of these active components is to include calibration mechanisms. The function of a big number of DACs on the PCB is to generate variable analogue bias values to both types of integrators implemented to compensate for mismatch. Several compensating methods were considered after the design of the second IC. The most promising techniques are based on calibration of the current mirrors, such as in [145] and [146]. Source degeneration of either the input or the output transistor is the working principle of both techniques. However the solution presented in [146] allows for digitally-controlled compensation and does not require periodically sampling as in [145].

To reduce the noise introduced by the test set-up, the next task would be to repeat the tests with the switched capacitor integrator. On the confirmation of the circuit sensitivity to noise, the redesign of the switch capacitor integrator in the appendix F is needed. Such effort is justified because it presents a small number of elements sensitive to mismatch than the SCI version, although it normally requires larger circuit area than the last. More studies on technology scaling and PVT variations could be addressed in further work.

In addition to the corrections needed to fully quantify the ASEC coder, next likely steps are to obtain IC results for some of the operations listed in chapter 4. Some of these operations were not tested on IC because they required specific hardware such as multiplication or the first method to implement the modulus function. On the other side, other operations can be implemented with the current ICs but with different or improved test set-up. For instance, the weighted summation would be realizable with a better FGPA programming code.

Additional time can also be spent on the identification and study of new operations performed with the ASEC scheme as well new uses for the proposed communication scheme. This scheme is highly suitable to sensors applications. As an example, ASEC has been used associated with MEMS cantilevers to implement biologically inspired microphones [147]. Also the work presented in [107] can be further improved with the ASEC scheme.

Finally, for sake of completeness, a complete FPAA can be designed including selected CABs circuits. A first prototype has been designed using Dr. Thomas Koickal’s programmable circuit block for reconfigurable neuromorphic systems, as proposed in [148] and in [64].

6.3 Final considerations and conclusion

In summary, this project aimed to study the viability of using asynchronous timing modulations as a viable alternative to the classical approach of dedicated channels in programmable analogue arrays. The outcome was the creation and definition of an event-based communication method, named asynchronous spike event coding scheme and the identification that such method can perform a set of arithmetic operations.

The working principle of this communication scheme was demonstrated as the association of a analogue-to-spike and spike-to-analogue coding methods with an asynchronous digital communication channel (AER). Spike events are essentially asynchronous and robust digital signals that are easy to route on shared channels, not only between CABs, but also between ICs providing improved scalability. Such scalability is also improved because the novel method presents smaller area increment than the traditional switched-matrix approach.

In this method, spike events are transmitted asynchronously and power dissipation is dependent upon signal activity. No spike events are generated when the input signal is constant. This is in direct contrast to other pulse based programmable analogue arrays. This characteristic has a deep impact on the communication channel bandwidth.

The intrinsic computation capability of the spike event coding scheme was also demonstrated. This provides basic arithmetic operations essentially in the communication channel, without the need for additional CABs thereby enhancing the computing capability and flexibility of a programmable analogue array.

Two ICs were designed to serve as proof-of-concept of such communications scheme, with the design step of the scheme parameters being explained. The results of these ICs prove the feasibility of these method.

Appendix A
Analysis of spike frequency and channel efficiency

For the analyses presented here, it is assumed input signals can be considered approximately constant during the interval between 2 or 3 successive spikes. In other words, the input signal $x_R(t)$ frequency is slower when compared with the spike frequency. For all methods considered in chapter 3, the output spike frequency is given by the inter-spike period

$$f_{sk} = \frac{1}{\Delta t_1 + \Delta t_2} \tag{A.1}$$

where Δt_1 is the interval T (the spike ”width”) and Δt_2 is Δt (the inter-spike interval) for the spike modulations (BSD, TSD, BSSD and TSSD).

It is also used the same measurement N_s used in [102], where it was named $N\{g(1)\}$, for the number of spikes by second generated when applying a sine wave signal $x(t) = A\ \sin(\omega_{in} t)$ to each of the methods inputs

$$N_s = N\{g(1)\} = \int_0^1 f(t)dt \tag{A.2}$$

A.1 Delta based modulations

ADM

For the analysis of Asynchronous Delta Modulation (ADM), I will use the figure A.1(a) is used. The slope of the signal $z(t)$ is given by

$$\alpha_1 = \dot{z}(t) \approx \frac{2\,d + \Delta x(\Delta t_1)}{\Delta t_1} \tag{A.3}$$

given that δ is given by equation 3.13 and

$$\dot{x}(t)|_{\Delta t_1} \approx \frac{\Delta x(\Delta t_1)}{\Delta t_1} \Rightarrow \Delta t_1 \approx \frac{\delta}{\alpha_1 - \dot{x}(t)|_{\Delta t_1}} \tag{A.4}$$

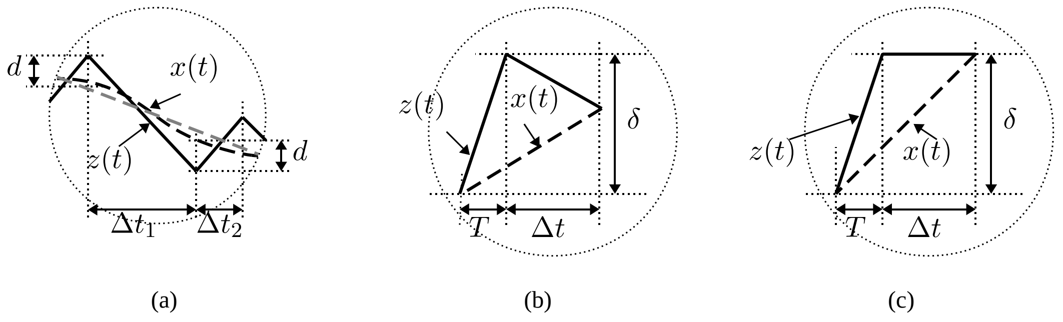


Figure A.1: *Waveform details of (a) ADM, (b) BSD and (c) TSD. This figure present the same wave details shown in the insets of figures 3.6(b), 3.5(b) and 3.4(b), respectively. In figure (c), the DC level of the feedback signal $z(t)$ is shifted to simplify the analysis.*

Using similar analysis for the period Δt_2 and considering $\alpha = k_I\, y(t)$, then $\alpha = \alpha_1 = -\alpha_2$ and the frequency is given by [110]

$$f_{sk}(t) \approx \frac{k_I^2 - [\dot{x}(t)]^2}{2\, k_I\, \delta} \tag{A.5}$$

considering that the output amplitude is the unit. Therefore, ADM presents no spikes when the absolute derivative of the input signal equals the integration gain k_I and the maximum spike frequency $\left(\frac{k_I}{2\delta}\right)$ when the input signal is constant.

Reminding that ADM needs two spikes for each period, the number of spikes N_s is

$$N_s = 2\int_0^1 \frac{k_I^2 - [A\omega\cos(\omega\, t)]^2}{2\, k_I\, \delta} \approx \frac{k_I}{\delta} - \frac{A^2\,\omega^2}{2\,\delta\, k_I} \tag{A.6}$$

If $\omega \gg 1$ and $k_I = \dot{x}(t)|_{(max)} = A\,\omega$, then

$$N_s \approx \frac{A\,\omega}{2\,\delta} = \frac{\pi}{2}\, f_{in}\, 2^{N_b} \tag{A.7}$$

where δ is given by equation 3.5. Therefore, the channel efficiency for this modulation is

$$CE_{ADM} = \frac{f_{in} 2^{N_b}}{\frac{\pi}{2}\, f_{in}\, 2^{N_b}} = \frac{2}{\pi} \tag{A.8}$$

BSD

The analysis of the spike frequency for BSD modulation is similar to the ADM presented before. By inspecting the figure A.1(b) and considering that the input signal derivative is approximately constant during the interval of two spikes, the input dynamics is

$$\dot{x}(t) \approx \frac{\delta + \alpha\,\Delta t}{T + \Delta t} \tag{A.9}$$

where $\alpha = \dot{z}(t)$. As the period is defined as $T + \Delta t$, the spike frequency is given by

$$f_{sk}(t) \approx \frac{\dot{x}(t)}{\delta + \alpha\,\Delta t} \tag{A.10}$$

Equation A.10 is not useful because both Δt and $\dot{x}(t)$ are functions of time. If we rewrite equation A.9 in terms of Δt and replace into equation A.1, the output spike frequency can be expressed as

$$f_{sk}(t) \approx \frac{\dot{x}(t) - \alpha}{\delta - \alpha\,T} \tag{A.11}$$

According to equation A.11, there will be no spikes just when $\dot{x}(t) = \alpha$, i.e., the integrator gain $k_I = \dfrac{\alpha}{p_{dc}}$ is just sufficient to follow the input signal when this is decreasing. The maximum frequency occurs when the derivative of the input signal is maximum.

The number of spikes in one second is, applying equation A.11 in equation A.2,

$$N_s = \int_0^1 \frac{A\,\omega\cos(\omega\,t) - \alpha}{\delta - \alpha\,T} = \frac{A\sin(\omega) - \alpha}{\delta - \alpha\,T} \tag{A.12}$$

From the condition to obtain the minimum number of spikes, for a sine wave $\alpha = -\dot{x}(t)|_{(max)} = A\,\omega$. Also considering $\omega \gg 1$, equation A.12 became

$$N_s \approx \frac{1}{T + \dfrac{\delta}{A\,\omega}} = \pi\,f_{in}\,2^{N_b} \tag{A.13}$$

using equation 3.5 and assuming that $\omega \ll \dfrac{1}{2^{N_b}\,T}$.

Applying equation A.13 into equation 3.15, the channel efficiency for BSD modulation is

$$CE_{BSD} = \frac{f_{in} 2^{N_b}}{\pi \, f_{in} \, 2^{N_b}} = \frac{1}{\pi} \tag{A.14}$$

TSD

Figure 3.4(b) is used to determine the spike frequency of the TSD modulation. It can be redrawn as in figure A.1(c) for better illustration, by shifting the DC level of signal $z(t)$. From the figure it can be shown that

$$|\dot{x}(t)| \approx \frac{|\Delta x(t)|}{T + \Delta t} \tag{A.15}$$

As the signal amplitude variation $\Delta x(t)$ during an inter-spike period is equal to δ, the spike frequency of the TSD modulation

$$f_{sk}(t) \approx \frac{|\dot{x}(t)|}{\delta} \tag{A.16}$$

Studying the equation A.16, one can verify that the minimum frequency occurs when the input is constant and the maximum when the absolute value of input derivative is maximum.

Applying equation A.2, TSD's number of spikes in one second equals to

$$N_s = \int_0^1 \frac{A \, \omega}{\delta} \, |\cos(\omega \, t)| \, dt \approx 4 \, \frac{A}{\delta} \, \frac{\omega}{2 \, \pi} = 2 \, f_{in} \, 2^{N_b} \tag{A.17}$$

which is the same result found in [102]. The channel efficiency in equation 3.15 turns to be

$$CE_{BSD} = \frac{f_{in} 2^{N_b}}{2 \, f_{in} \, 2^{N_b}} = \frac{1}{2} \tag{A.18}$$

A.2 Sigma-delta based modulations

ASDM

Asynchronous Sigma Delta modulation analysis were already presented by some works, such as [114] and [91] and for convenience is shown here.

By observing the figure A.2(a), we notice the hysteresis comparator switches the coder output either when an increasing integrator output $z(t)$ reaches $+\delta$ or a decreasing $z(t)$ reaches $-\delta$.

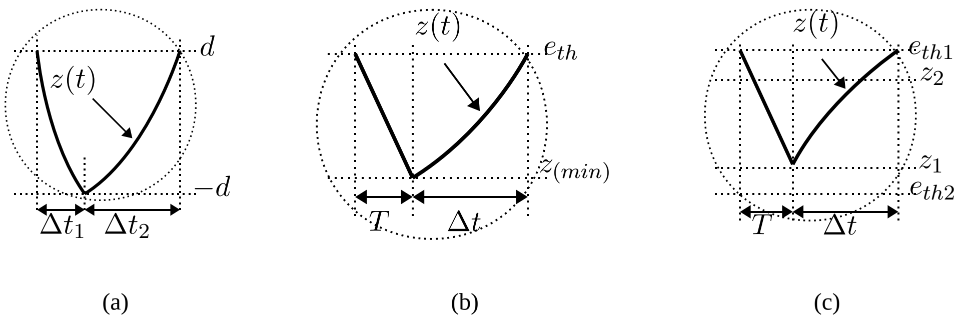


Figure A.2: *Waveform details of (a) ASDM, (b) BSSD and (c) TSSD. This figure present the same wave details shown in the insets of figures 3.7(b), 3.8(b) and 3.9(b), respectively. In (a), the input signal $x(t)$ is DC shifted for better illustration.*

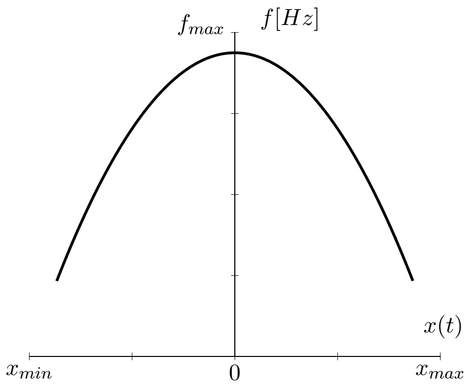


Figure A.3: *Spike frequency as a function of input signal for ASDM modulation.*

For the increasing $z(t)$ interval Δt_1 [149]:

$$d - (-d) = \frac{1}{k_i} \int_{t_1}^{t_1+\Delta t_1} [b + x(t)]\, dt \Rightarrow \Delta t_1 = \frac{2dk_i}{b + x(t)} \tag{A.19}$$

and similarly for the interval Δt_2

$$\Delta t_2 = \frac{2dk_i}{b - x(t)} \tag{A.20}$$

The switching frequency is then given by [91]:

$$f_{sk}(t) = \frac{b^2 - [x(t)]^2}{4dbk_i} \tag{A.21}$$

Analysing equation A.21, we can notice the maximum frequency occurring when the input signal is null. This is named oscillatory or idling frequency $f_{sk0} = \dfrac{b}{4dk_i}$. On the other hand, the minimum when the input signal is maximum or minimum as it is show in figure A.3.

Using equation A.21 and equation A.2 and considering that for each period there are two spikes, the number of spikes per second for a sine wave signal is

$$N_s = \frac{1}{2\ d\ b\ k_i}\left\{b^2 - \frac{A^2}{2}\left[1 - \frac{1}{2\omega}\sin(2\omega)\right]\right\} \approx \frac{b^2 - \dfrac{A^2}{2}}{2\ d\ b\ k_i} \quad (\text{for } \omega \gg \frac{1}{2}) \tag{A.22}$$

using the floowing trigonometric property

$$\int \sin^2(ax)dx = \frac{1}{2} - \frac{1}{4a}\sin(2ax) + C \tag{A.23}$$

Consequently, the channel efficiency for this modulation is given by

$$CE_{ASDM} = \frac{2\ d\ b\ k_i\ f_{in}2^{N_b}}{b^2 - \dfrac{A^2}{2}} \tag{A.24}$$

when replacing d as in equation B.3 with $x(t) = 0$ and $\tau = (2\pi f_{in})^{-1}$. The ASDM channel efficiency tends to 0.423 as the resolution increase. This is independent of the input frequency if the filter pole is a function of it.

BSSD and TSSD

Binary and Ternary Spike Sigma Delta modulations analyses are similar to the analysis of the ASDM. However, differently from ASDM, these modulations presents a fixed interval T on the onset of the spike. For BSSD, during this period $z(t)$ reaches the minimum value $z_{(min)}$:

$$z_{(min)} - e_{th} = \frac{1}{k}\int_{t_1}^{t_1+T}\left(x(t) - y_{(max)}\right)dt. \tag{A.25}$$

On the other hand, the maximum value for $z(t) = V_{th}$ is reached at Δt

$$e_{th} - z_{(min)} = \frac{1}{k}\int_{t_2}^{t_2+\Delta t}\left(x(t) - y_{(min)}\right)dt. \tag{A.26}$$

Combining equation A.25 and equation A.26, the switching frequency is then given by

$$f_{sk}(t) = \frac{1}{T}\frac{x(t) - y_{(min)}}{\Delta y} \tag{A.27}$$

where $\Delta y = y_{(max)} - y_{(min)}$.

For BSSD, the frequency is a linear function of the input signal $x(t)$. Therefore the maximum frequency occurs when the input signal is also maximum whilst the minimum occurs when the input signal is minimum and equal to $y_{(min)}$. Because $y_{(min)}$ can be designed to be the minimum value of $x(t) = x_{(min)}$,

$$N_s = \frac{1}{T\Delta y}\left\{\frac{A}{\omega}\left[1-\cos(\omega)\right] - y_{(min)}\right\} \approx \frac{A}{T\Delta y} \quad \text{(for } \omega \gg 2\text{)} \tag{A.28}$$

In TSSD case

$$z_1 - e_{th1} = \frac{1}{k}\int_{t_1}^{t_1+T}\left[x(t) - y_{(max)}\right]dt \tag{A.29}$$

for positive spikes and

$$z_2 - e_{th2} = \frac{1}{k}\int_{t_2}^{t_2+T}\left[x(t) - y_{(min)}\right]dt \tag{A.30}$$

for negative spikes. For the period Δt, when no spikes are outputted

$$e_{th1(2)} - z_{1(2)} = \frac{1}{k}\int_{t_3(4)}^{t_3(4)+\Delta t} x(t)dt \tag{A.31}$$

From equation A.29 to equation A.31, the instantaneous spike frequency is

$$f_{sk}(t) \approx \frac{1}{T}\frac{|x(t)|}{y} \tag{A.32}$$

when $y = y_1 = -y_2$.

As shown in equation A.32 the frequency is proportional to the absolute value of the input amplitude and its maximum is $\frac{A}{Ty}$ and the minimum is 0 when $x(t) = 0$.

Integrating equation A.32 over time gives the number of spikes per second N_s

$$N_s \approx \frac{2A}{\pi T y} \quad (\text{for } \omega \gg 2). \tag{A.33}$$

Finally, the channel efficiencies of BSSD and TSSD are

$$CE_{BSSD} = \frac{T \Delta y f_{in} 2^{N_b}}{A} = \frac{T \Delta y f_{in}}{\delta} \tag{A.34}$$

and

$$CE_{TSSD} = \frac{\pi T y f_{in} 2^{N_b}}{2A} = \frac{\pi T y f_{in}}{2\delta} \tag{A.35}$$

respectively, considering $\delta = \dfrac{A_{dec}}{2^{N_b}}$ and $A_{dec} = 0.5A$ due to 3-dB filter attenuation and equation B.3 with $x(t) = 0$ and $\tau = \dfrac{1}{2\pi f_{in}}$. As the resolution increase, the BSSD and TSSD channel efficiencies tend to constant values of 0.212 and 0.333 respectively, regardless the resolution and input signal frequency.

Appendix B
Spike modulations numerical models

All analyses showed in chapters 3 and 4 were performed using numerical models. Theses models were written in Matlab® and Octave languages syntax.

Signals are represented by a 2-line matrix: the first lines refers the time whilst the second is the signal amplitude. Variables are as follows:

Variable	Description
Dx	Maximum amplitude variation of the input signal $x(t)$, i.e., $\Delta x(t)_{max}$.
freq	Input bandwidth. The frequency of $x(t)$ when considering $x(t) = A\sin(\omega t)$.
Nb	Designed resolution of the output signal $x_R(t)$ ($z_R(t)$ for delta modulations).
margin	Margin (percentage) of $y(t)$ amplitude for some modulations. Default is 0.
mismatch	Mismatch on the comparator(s) threshold. Default is 0.
A	Sine wave amplitude. i.e., A = $\Delta x(t)_{max}/2$.
w	Angular frequency, i.e., w = $2 * \pi *$ freq.
steps	Maximum amplitude variation to the pulse-induced change ratio. steps = 2^{Nb}.
pole	Pole of the first order low pass filter. In the simulations results, pole = freq.
fs	Sampling frequency of all signals.
Wn	Normalized pole of the low pass filter in z-domain. Wn = pole/(fs/2).
b,a	Numerator and denominator of the Butterworth filter. [b,a] = butter(1,Wn).

Table B.1: *Code global variables.*

Following octave codes refers to the design (*par.m), coding (*coder.m) and decoding (*decoder.m) for each of the spike modulations studied. Parameter calculation for delta based modulations follows the equations showed along the thesis. For sigma-delta cases, a brief explanation of the approximated equations is provided.

TSD Modulation

```
function TSDpar
% TSD coder: Parameters calculation
    delta    = 2*A / steps;
    dxdt_max = A * w;        % maximum derivative for x(t) = A sin(wt)
    refrac   = delta/dxdt_max - td;
    zi       = -2*delta/4;  % integrator initial condition
    if pole/freq < 0.5        % set decoder filter to -90 phase change
        dec = pole/freq;
        attdB = 20*log10(dec);
        att = 10^(attdB/20);
        xri = (-A + zi*0) * att;
    else
        xri = zi;
    end
    nspk     = 2 * steps * freq;      % theoretical # spikes/s
    FoM      = Nb/log2(nspk/freq);    % theoretical FoM
end
```

Design parameters for TSD models

```
function TSDcoder
% TSD modulation: coder
    znext    = zi;
    trf      = -refractory;
    Deth     = (1 + mismatch/100)*delta;
    for n = 1:size(x,2);
        zcurr = znext;                % feedback signal z(t)
        errn  = x(2,n) - zcurr;       % error signal e(t)
        if x(1,n) - trf >= refractory
            if errn > Deth/2
                y(n) = 1;
            elseif errn < -Deth/2
                y(n) = -1;
            else
                y(n) = 0;
            end
        else
            y(n) = 0;
        end
        znext = zcurr + y(n)*delta; % integrator
        trf   = x(1,n);
    end
end
```

TSD coder model

```
function TSDdecoder
% TSD modulation: decoder
```

```
    fs     = 1 / ( y(1,2)- y(1,1) );
    zr(1) = zi;
    zr     = [zi delta*cumsum( y(2,:) ) + zi ];  % integrator modelled as an
        accumulator
    zr     = [y(1,:); zr(1:numel(zr)-1)];
    xr     = [zr(1,:); filter(b, a, zr(2,:), xri)];  % filtered data
end
```

TSD decoder model

BSD Modulation

```
function BSDpar
% BSD coder: Parameters calculation
    delta    = 2*A / steps;
    dxdt_max = A * w;        % maximum derivative for x(t) = A sin(wt)
    refrac   = 1/2 * delta / dxdt_max;
    kd       = dxdt_max;    % integrator gain equals dxdt_max
    nspk     = pi * freq * steps;   % theoretical # spikes/s
    FoM      = Nb/log2(nspk/freq);  % theoretical FoM
end
```

Design parameters for BSD models

```
function BSDcoder
% BSD modulation: coder
    td      = x(1,2) - x(1,1);
    znext   = 0;
    trf     = -refractory;
    for n = 1:size(x,2)
        zcurr = znext;                  % feedback signal z(t)
        errn = x(2,n) - zcurr;          % error signal e(t)
        y(n) = ( (errn - off/2) >= 0 ); % output signal y(t)
        % feedback update
        if (y(n) ~= 0) && (x(1,n) - trf > refractory)
            trf = x(1,n);
            znext = zcurr + delta;
        else
            znext = zcurr - kd*td;
            y(n) = 0;
        end
    end
    y = [x(1,:); y(1:size(y,2))];
```

```
end
```

BSD coder model

```
function BSDdecoder
% BSD modulation: decoder
    td    = y(1,2) - y(1,1);
    zrant = 0;
    for n = 1:size(y,2);
        if ( y(2,n) > 0 )
            zr(n+1) = zr(n) + delta;
        else
            zr(n+1) = zr(n) - kd*td;    % integrator
        end
    end
    zr    = [y(1,:); zr(1:size(zr,2)-1)];
    xr    = [ zr(1,:); filter(b, a, zr(2,:)) ]; % filtered data
end
```

BSD decoder model

ADM Modulation

```
function ADMpar
% ADM coder: Parameters calculation
    q        = 2*A / steps;
    d        = margin*0.5*q;     % appr. q = d-(-d)
    dxdt_max = A * w;            % maximum derivative for x(t) = A sin(wt)
    b        = margin * A;
    kd       = dxdt_max;         % integrator gain equals dxdt_max
    nspk     = pi/2 * freq * steps; % theoretical # spikes/s
    FoM      = Nb/log2(nspk/freq);  % theoretical FoM
end
```

Design parameters for ADM models

```
function ADMcoder
% ADM modulation: coder
    td       = x(1,2) - x(1,1);
    znext    = 0;
    if x(2,1) > 0   % initial condition of y(t)
        yant = -b;
    else
```

```
        yant = b;
    end
    for n = 1:size(x,2)
        zcurr   = znext;              % feedback signal z(t)
        errn    = x(2,n) - zcurr;    % error signal e(t)
        ys(n)   = 0;
        if (errn - d) >= 0
            yn    = b;                % PWM "output"
            if yant ~= yn
                ys(n) = 1;            % Spike output
            end
        elseif (errn + d) <= 0
            yn    = -b;               % PWM "output"
            if yant ~= yn
                ys(n) = -1;           % Spike output
            end
        else
            yn    = yant;
        end
        znext = zcurr + kd*yn*td;    % integrator
        yant  = yn;                   % PWM output
    end
    y = [x(1,:); ys(1:size(ys,2))];
end
```

ADM coder model

```
function ADMdecoder
% ADM modulation: decoder
    td = y(1,2) - y(1,1);
    % compute initial condition of PWM y(t)
    indp = find(y(2,:) > 0);
    indn = find(y(2,:) < 0);
    yant = 2*(indp(1) > indn(1)) - 1;
    zr(1) = 0;
    for n = 1:size(y,2)
        if y(2,n) ~= 0
            yant = y(2,n);
        end
        yr(n) = yant;
        % integrator
        zr(n+1) = zr(n) + kd * yant * td;
    end
    xr    = [ zr(1,:); filter(b, a, zr(2,:)) ]; % filtered data
end
```

ADM decoder model

ASDM Modulation

One characteristic common to the sigma-delta versions of spike modulation is that the output is the filtered result of a sequence of pulses. Considering a first order filter, its step (with amplitude U) response is given by:

$$v(t) = v(\infty) + [v(0) - v(\infty)]\, e^{-\dfrac{t}{\tau}} \tag{B.1}$$

where τ is filter time constant, $v(0)$ is the filter initial condition and $v(\infty)$ is the final value of the filter output given by $v(\infty) = U\tau$.

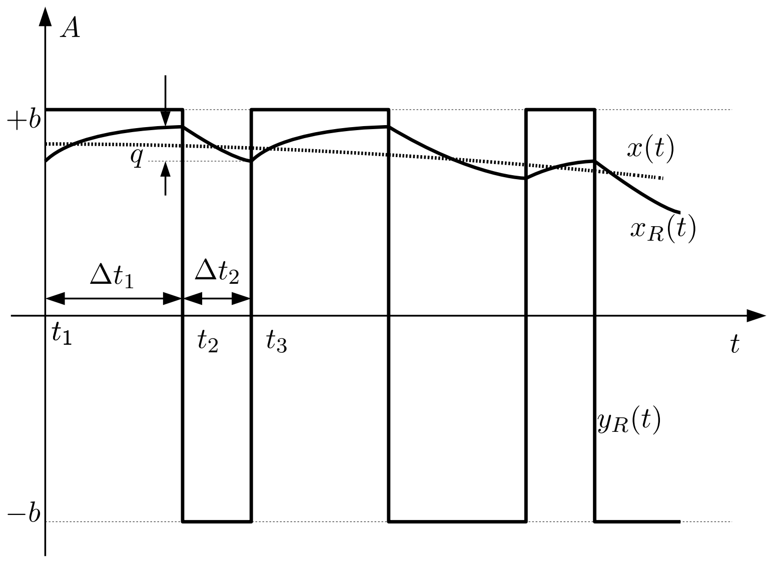


Figure B.1: *Asynchronous sigma-delta modulation output. Illustration of ASDM output signals behaviour.*

Figure B.1 is an illustration of the behaviour of the filter output for a binary pulse such as it happens in ASDM modulation. The filter output rises during the interval Δt_1 and falls during Δt_2. We we first consider the interval Δt_1, being the case for the interval Δt_2 presenting symmetric expressions. From equation B.1:

$$x_R(t_1+\Delta t_1) = x_R(\Delta t_1) + q/2 = b + [x_R(\Delta t_1) - q/2 - b]\, e^{-\dfrac{t_1-\Delta t_1-t_1}{\tau}} \Rightarrow$$

$$\Delta t_1 = \tau \ln \frac{b - x_R(\Delta t_1) + q/2}{b - x_R(\Delta t_1) - q/2} \tag{B.2}$$

where $x_R(\Delta t_1)$ is the middle point of $x_R(t)$ during this interval and q, the variation on the filter output amplitude during the interval Δt_1, is defined by design. Reminding that equation A.21 refers to the pulse frequency, we can combine it with equation B.2 (and the equivalent for Δt_2):

$$\Delta t_1 + \Delta t_2 = \frac{4\,d\,k\,b}{b^2 - x(t)^2} \approx \tau \ln \frac{(b+q/2)^2 - x_R(t_3-t_1)^2}{(b-q/2)^2 - x_R(t_3-t_1)^2} \Rightarrow$$

$$d = \frac{b^2 - x(t)^2}{4\,k\,b} \tau \ln \frac{(b+q/2)^2 - x(t)^2}{(b-q/2)^2 - x(t)^2} \tag{B.3}$$

considering that input signal $x(t)$ is approximately constant during $\Delta t_1 + \Delta t_2$ and $\overline{x_R(t_3-t_1)} \approx x(t)$.

Expressions for other sigma-delta modulations (BSSD and TSSD) are similar but with one interval fixed (T).

```
function  ASDMpar
% ASDM coder: Parameters calculation
    k      = 1;              % integrator gain
    Ad     = 0.5*A;          % amplitude at decoder H(w) = -6dB
    q      = 2*Ad / steps;  % maximum error in the receive side (happens
        close to x=0)
    q      = 1.333*q;        % correction factor
    b      = A + q/2;
    tau    = 1/(2*pi*pole);
    % d = (b^2-x^2)/(4kb) * tau *ln( ((b+q/2)^2 -x^2) / ((b-q/2)^2 -x^2) )
    x      = 0;              % worst case
    A      = (b + q/2)^2 - x^2;
    B      = (b - q/2)^2 - x^2;
    C      = (b^2 - x^2) / (4*k*b);
    d      =  C * tau * log(A/B);
    nspk   = (b^2 - 0.5*A^2) / (2*d*k) % theoretical # spikes/s
    FoM    = Nb/log2(nspk/freq);       % theoretical FoM
end
```

Design parameters for ASDM models

```
function ASDMcoder
% ASDM modulation: coder
    td    = x(1,2) - x(1,1);
    znext = 0;
    yn    = 1;
    for n = 1:size(x,2)
        zcurr = znext;
        errn  = x(2,n) - b*yn;          % error signal e(t)
        znext = zcurr + 1/k*errn*td;    % integration signal z(t)
        ys(n) = 0;
        if znext >= d
            yn = 1;
        elseif znext <= -d
            yn = -1;
        else
            yn = yant;
        end
        if yant ~= yn   % spike generation
            ys(n) = yn;
        end
    end
    y = [x(1,:); ys(1:size(ys,2))];
end
```

ASDM coder model

```
function ASDMdecoder
% ASDM modulation: decoder
    xri   = 0;
    yant  = 0;
    % compute initial condition of PWM y(t)
    indp = find(y(2,:) > 0);
    indn = find(y(2,:) < 0);
    yant  = 2*(indp(1) > indn(1)) - 1;
    for n = 1:size(y,2);
        if y(2,n) ~= 0
            yant = y(2,n);
        end
        yr(n) = b*yant;
    end
    % filtered data
    xr = [ y(1,:); filter(b, a, yr, xri) ];
end
```

ASDM decoder model

TSSD Modulation

```
function TSSDpar
% TSSD coder: Parameters calculation
    k        = 1;           % integrator gain
    Ad       = 0.5*A;       % amplitude at decoder H(w) = -6dB
    q        = 2*Ad / steps;
    q        = 1.33*q;      % correction factor
    tau      = 1/(2*pi*pole);
    % T = 2 * tau *( 1 - |x|/q * ln( (|x|+q/2) / (|x|+q/2) ) )
    x        = A;           % worst case
    a1       = abs(x) + q/2;
    a2       = abs(x) - q/2;
    a3       = abs(x)/q * log(a1./a2);
    T        = 2 * tau * (1 - a3);
    % round T to be multiple of dt
    kT       = round(abs(T)/dt);
    T        = kT*dt;
    ya       = q*tau/T + abs(x) - q/2;   % pulses amplitude
    delta    = margin * ya * T/k;
    nspk     = 2*A / (pi*T*ya);          % theoretical # spikes/s
    FoM      = Nb/log2(nspk/freq);       % theoretical FoM
end
```

Design parameters for TSSD models

```
function TSSDcoder
% TSSD modulation: coder
    td     = x(1,2) - x(1,1);
    kT     = floor(T/td);    % number of samples for the pulse T
    spk    = 0;
    y(1)   = 0;
    znext  = 0;
    for n = 1:size(x,2)
        zcurr   = znext;
        yf      = ya * y(n);      % feedback gain
        errn    = x(2,n) - yf;    % error signal err(t)
        znext   = zcurr + 1/k * errn * td;   %integration signal z(t)
        % feedback generator
        if spk > 0      % pulse hasn't finished yet
            y(n+1) = y(n);
            spk    = spk - 1;
        elseif abs(znext) > delta/2     % start pulse
            spk    = kT;
            y(n+1) = sign(znext);
        else
            y(n+1) = 0;
        end
    end
    y  = [x(1,:); y(1:size(y,2)-1)];
    ys = abs(y(2,:)) .* (y(2,:) - circshift(y(2,:),[0,1])); % converting to
        spikes
    y  = [y(1,:); ys];
end
```

TSSD coder model

```
function TSSDdecoder
% TSSD modulation: decoder
    td    = y(1,2) - y(1,1);
    kT    = floor(T/td);    % number of samples for the pulse T
    xri   = 0;
    indy  = find(abs(y(2,:)) == 1); % find spikes
    yi    = y(2,:);
    for n = 1:numel(indy)   % converting from spikes
        yi( indy(n) : indy(n)+kT-1 ) = y(2, indy(n));
    end
    s1 = [ y(1,:) ; ya*yi ];     % apply gain and dc value to the spikes
    xr = [ y(1,:) ; filter(b, a, s1(2,:), xri ) ];  % filtered data
end
```

TSSD decoder model

BSSD Modulation

```
function BSSDpar
% BSSD coder: Parameters calculation
    k       = 1;            % integrator gain
    Ad      = 0.5*Dx/2;     % amplitude at decoder H(w) = -6dB
    q       = 2*Ad / steps;
    q       = 1.33*q;       % correction factor
    xmin     = -Dx/2;
    yo       = xmin - q/2;
    tau      = 1/(2*pi*pole);
    % T = 2 * tau *( 1 - (x-yo)/q * ln( 1 + q/(x - xmin)) )
    x        = 0.0;           % worst case
    a1       = log(1 + q./(x - xmin));
    a2       = (x - yo)/q;
    a3       = 1 - a2.*a1;
    T        = 2 * tau * a3;
    % round T to be multiple of dt
    kT       = round(abs(T)/dt);
    T        = kT*dt;
    ys       = q*tau./T + x - q/2;   % pulses amplitude
    nspk     = A/(T*(ys-yo));        % theoretical # spikes/s
    FoM      = Nb/log2(nspk/freq);   % theoretical FoM
end
```

Design parameters for BSSD models

```
function BSSDcoder
% BSSD modulation: coder
    td     = x(1,2) - x(1,1);
    kT     = floor(T/td);   % number of samples for the pulse T
    znext  = 0;
    spk    = 0;
    y(1)   = 0;
    for n = 1:size(x,2)
        zcurr = znext;
        yf    = ys*y(n) + yo*(1-y(n)); % feedback gain
        errn  = x(2,n) - yf;           % error signal e(t)
        znext = zcurr + 1/k*errn*td;   % integration signal z(t)
        % feedback generator
        if spk > 0            % pulse hasn't finished yet
            y(n+1) = y(n);
            spk    = spk - 1;
        elseif znext >= 0   % start pulse
            spk    = kT;
            y(n+1) = 1;
        else
            y(n+1) = 0;
        end
    end
    y  = [x(1,:); y(1:size(y,2)-1)];
    ys = ( y(2,:) - circshift(y(2,:),[0,1]) );  % converting to spikes
    y  = [y(1,:); (ys + abs(ys))/2];
end
```

BSSD coder model

```
function BSSDdecoder
% BSSD modulation: decoder
    td    = y(1,2) - y(1,1);
    kT    = T/td;
    xri   = 0;
    indy  = find(y(2,:) == 1);  % find spikes
    yi    = y(2,:);
    for n = 1:numel(indy)       % converting from spikes
        yi( indy(n) : indy(n)+kT-1 ) = y(2, indy(n));
    end;
    s1 = [ y(1,:); ys*yi + yo*(1-yi) ];     % apply gain and dc value to the
         spikes
    xr = [ y(1,:); filter(b, a, s1(2,:),  xri ) ];  % filtered data
end
```

BSSD decoder model

Appendix C
A distributed arbitration for AER protocol

In chapter 2 it was defined the Address Event Representation (AER) protocol as the handling mechanism to communicate spikes in the programmable analogue architecture.

AER is the de-facto standard communication protocol for neuromorphic systems [95]. It uses digital buses to broadcast spikes to every element is the system. Each spike is identified by a digital code representing the address of the sender or receiver elements.

However, sharing the same media (the digital bus) asynchronously can lead to transmission collisions. The system can just ignore such collisions, eventually leading to loss of information [101], or can handle these collisions by implementing some sort of arbitration. The arbitration process catches all the spikes involved in a collision and retransmit either one or all events sequentially.

Usually, arbitration is implemented in with a centralized arbiter connected to every element in the system. This appendix presents a distributed alternative for the centralized arbitration.

C.1 Distributed arbitration description

In centralized arbitration, each element or CAB, referred as node in this appendix, asks the arbiter for permission to write to the bus every time it generates an event. The arbiter grants or not this permission to the node according to the bus state. This communication between node and arbiter is performed using asynchronous request and acknowledge signals.

In this implementation of distributed arbitration, each node manage event collisions autonomously. This implementation presents two buses: a *wired-OR* and a *wired-AND*. Every time a node produces an event it directly write its own address on both buses and wait for a confirmation signal set by all other elements on the array whilst monitoring the buses. On the reception of this signal, the transmission was successful and another event can be transmitted.

However if another node writes its address before the confirmation signal, the value of both addresses will differ and then all nodes start an internal process to solve the conflict.

Three different methods to handle collisions were identified, ordered in increasing complexity:

1. ignore the collision and the events involved on that, signalling to the system that it is an invalid address;
2. identify which addresses have collided, pick one up and discard the others or
3. try to resend sequentially these event addresses when the address bus is idle again.

The first two options can lead to information loss or distortion, which can be intolerable to the system. The third option implies in information loss or distortion only if the information coding used requires exact time generation (i.e. Pulse Position Modulation).

In case of options 2 and 3, if a collision happens, each node decides if it will keep writing its address in the buses. After successive interactions, just one will keep writing its address to the buses. When this happens, the address buses values are identical and the system could use it as a valid event transmission.

In this work, a state machine was implemented to control the arbitration process. Besides the two buses the system uses some asynchronous signals to control each node state machine. Some of them are dual (wired-OR and wired-AND) control signals: Address Write (WR), address WRite Acknowledge (WRA), address ConFlicT detection (CFT) and address ReaD (RD); and one is a single address VALidation (VAL) control. This dual logic control signals are used to synchronize the internal state machines of each node.

This arbitration can be understood as an error detection protocol, with some differences from linear block codes. In linear bock codes, the receiver(s) try to identify valid codes from the (possible noisy) input data using a parity-check matrix. The transmitted data is greater, in number of bits, than the information itself because the extra bits are used to error detection. Greater the code size, greater is the probability of error detection and correction [150].

Here, both the emitter(s) and receiver(s) try to identify the error in the data, but only the emitter(s) try to correct the error, rewriting the information as many times as necessary. At each iteration, each emitter will rewrite it according to some predefined criteria. In this first implementation, only two arbitration criteria are available: the node with the smallest or the biggest address will prevail. The following example uses the smallest address criterion, but further considerations are similar for the second case.

A functional example

Now we describe the functionality of the arbitration process for a system using 3-bit address size for its elements. Consider an array of 4 nodes: node A which address is 001, node B (010), node C (011) and node D (100). The idle state values for the wired-OR and the wired-AND buses are, respectively, 000 and 111. These two values are reserved and cannot be used to address nodes. Such example was simulated and the results are presented in figure C.1.

A conflict between spikes generated by nodes B, C and D was forced to give a comprehensive description of the arbitration process. After this nodes write their addresses, the buses values will be 111 (wired-OR) and 000 (wired-AND). As they are different from each other and are not the same of the idle state, each node in the system is aware of an invalid address in the buses due to a collision.

From this point, if the system uses the conflict options 2 and 3, each node that has written its address (candidate node) will decide if it will keep writing its address or not.

Because of the most significant bit (MSB) of the wired-AND bus is zero (000), the node D understands that a node with a lower address was involved in the collision. Therefore, the node will give up of being a candidate, stop writing its address in the buses. The nodes B and C will keep writing their address in the next iteration, as both could have generated this zero. After both have rewritten their addresses, the resulting address buses values will be 011 and 010, which indicates a new collision.

In the next step, both nodes will look at the second MSB, which in this case is one in both buses. As both have generated these ones, both nodes will keep writing their addresses in the next iteration.

The last step will take a look at the least significant bit (LSB). It is one in the wired-OR bus and zero for the wired-AND bus. The node C will give up of writing its address because it could not have generated the zero in the wired-AND bus. Only the node B will keep writing its address and, this time, both addresses have the same value and each node in the system acknowledge that it is a valid address and can read it. Therefore, this arbitration requires, in the worst case, a number of iterations equals to the bus size to resolve a conflict, i.e., $\log_2 N$ with $N - 2$ being

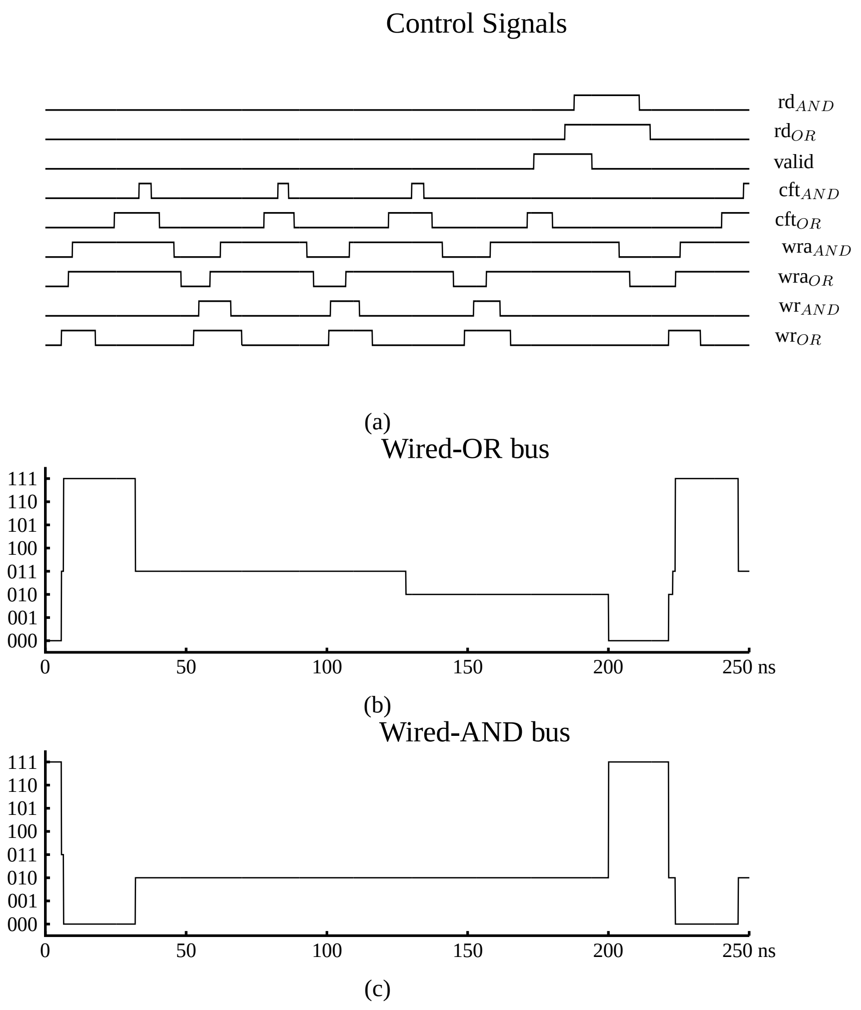


Figure C.1: *Distributed arbiter example waveform. (a) shows the control signals and (b) and (c) are the wired-OR and the wired-AND address buses, respectively. The collision happens at approx. 10ns when the signal* wr_{OR} *goes high for the first time and is resolved at approx. 195ns when the signal* valid *goes down. Probability of event generation was set high (approx. 50 %) to force a collision.*

the maximum number of nodes supported by the bus size.

C.2 Comparison with other AER arbitration techniques

We made some preliminary and generic considerations about the benefits and disadvantages to others arbitration techniques, considering overhead area, power consumption, latency and scalability.

Circuit area and power consumption

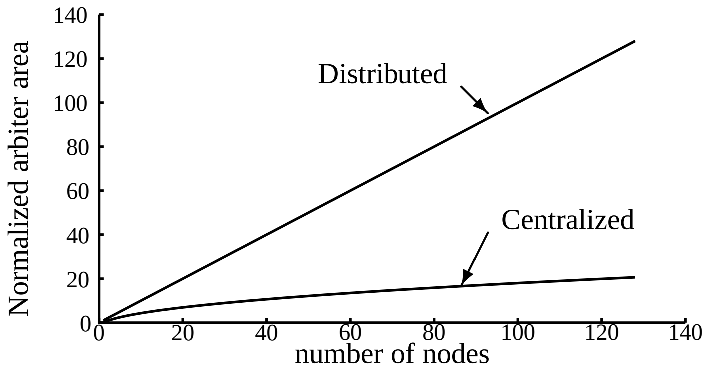


Figure C.2: *AER arbiter area. Normalized area required by* centralized *and* distributed *arbiter implementations. In the first the arbiter area grows proportionally to the root square number of nodes in the system whilst the second requires an area proportional to this number of nodes.*

Area comparison is not straight forward. It depends on process and design, among other factors. The total area was split in area occupied by the circuits and area occupied by signal routing.

In this approach, as each node has it own state machine to arbitrate collisions, the overhead in area increases proportionally to number of nodes N. In a classical AER implementation, the arbiter is implemented as a binary three of basic 2-input arbiters. The total number of arbiters need is given by the sum of a geometric series given by:

$$N_{arb(CA)} = \left(2^{L+1} - 1\right) + \left(2^{C+1} - 1\right) \tag{C.1}$$

which, if considering a square array with N nodes, results in:

$$A_{CA} = 2\left(\sqrt{N} - 1\right) \tag{C.2}$$

as the normalized area expression for the classical centralized arbiter case. The difference between the area requirements for both implementation in terms of the number of nodes in the array is illustrated figure C.2.

Furthermore, the output transistors of the node drive wires shared with $\sqrt{N}$ other nodes transistor drains plus the arbiter gate in the centralized arbiter version. In the distributed case the output transistors drive wires shared with all other $N-1$ nodes, both drains (transmission) and gates (receptions). It implies in greater transistors area and power consumption.

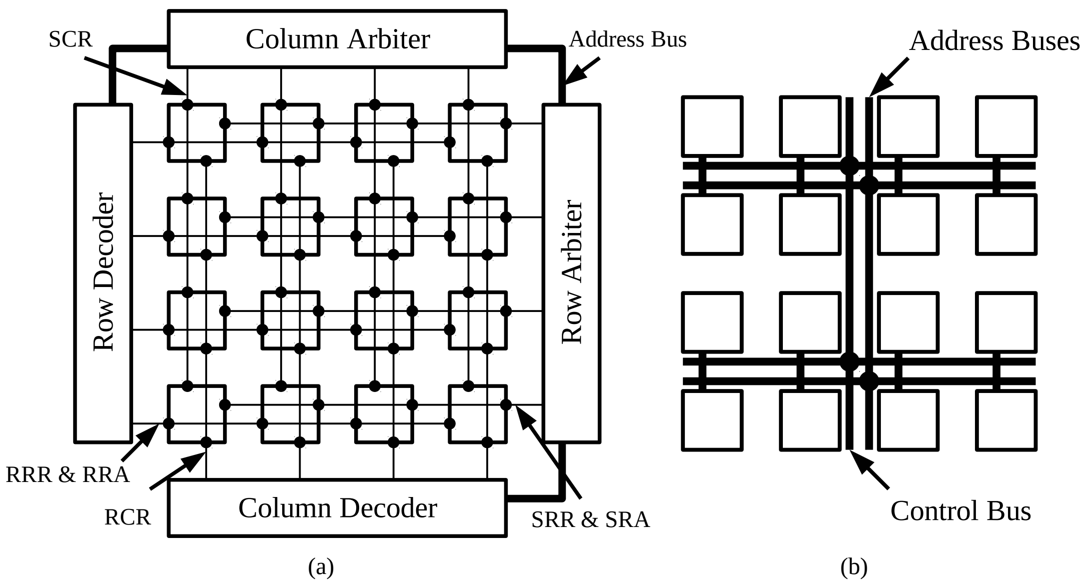


Figure C.3: *Example of routing placements. (a) Reduced version of original centralized arbiter with handshaking signals: Sender Column Request (SCR), Sender Row Request (SRR) and Acknowledge (SRA), Receiver Row Request (RRR) and Acknowledge (RRA) and Receiver Column Request (RCR) signals common to every node. (b) distributed arbiter implementation with both control and address buses common to every node.*

Routing area

The basic handshaking implementation of AER protocol requires one request signal and one acknowledge signal to each transmitting node in the system. This signals are connected to an arbiter which choose one of the nodes, in case of conflict, to transmit its address. For a system with N_T transmitting nodes the total amount of communication wires is $2N_T$.

However, if such nodes are aligned in rows and columns like in an array, all nodes in the same line or column can share this wire. The smaller number of request and acknowledge signals happens when we distribute the N_T nodes in a squared array configuration [95], i.e. the number of lines is equal to the number of columns. In this case, two request and two acknowledge signals are needed (one pair for all nodes in the column and another for the nodes in the same row. This configuration leads to a total of $4\sqrt{N_T}$ control signals. Further developments reduced this number to $3\sqrt{N_T}$, concentrating the acknowledge information in just one signal.

Similar number is found in the receiving side[1] totalling $3(\sqrt{N_T} + \sqrt{N_R})$ signals. for the

[1]The original application of the AER protocol was to connected two different systems, usually in different ICs,

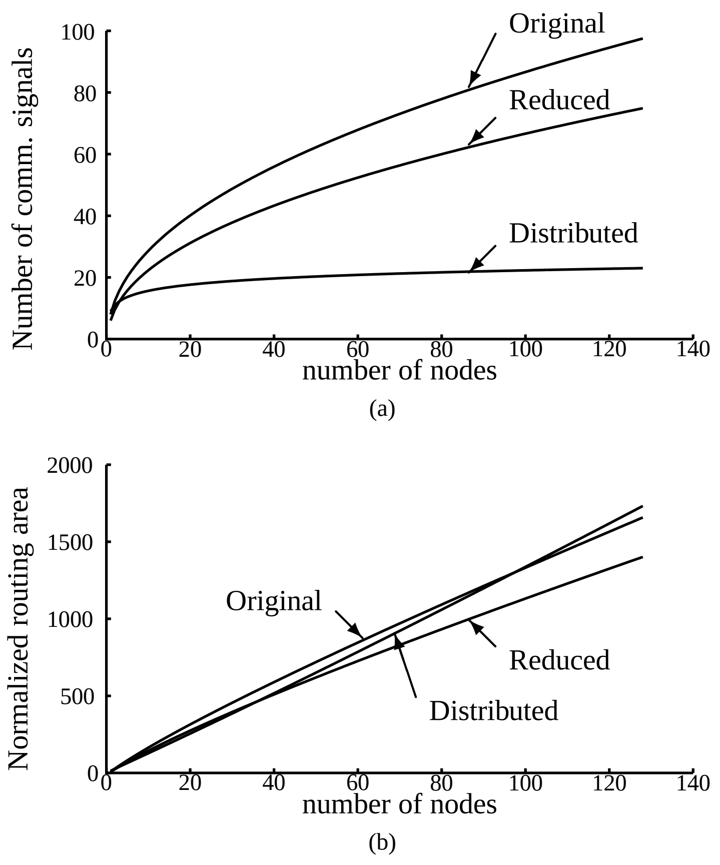


Figure C.4: *AER number of signals and routing area for a square array. Three different AER architectures were compared. The first is the* original *central arbiter. It uses four request and four acknowledge signals and* $\log_2(N)$ *address bus size. The second is a* reduced *version with just two acknowledge signals. The third is the* distributed *arbiter version presented in this appendix. Although this version presents a small number of control signals (a) for large number of elements, it also requires more area (b) to implement the signal routing.*

case where transmitting signals are also potential receivers, the number is $6\sqrt{N}$. The bus is accessible only by the arbiter and address decoder. A typical placement is showed in figure C.3.

Therefore, the area occupied by such squared distribution of nodes is:

$$A_{rout(CA)} \propto \sqrt{N}\left[\left(3\sqrt{N} + 4\log_2 N\right)(k_1 + k_2)\right] \tag{C.3}$$

where L, the number of lines, is $\sqrt{N}$ for a squared layout, and k_1 and k_2 are the height and width of each node, respectively.

with one only sending spikes and the other only receiving them.

In the distributed arbiter implementation, the system needs nine control signals and one dual address bus. Therefore, the routing area is:

$$A_{rout(DA)} \propto (2\log_2 N + 9)\left(0.5Nk_2 + k_1\sqrt{N}\right) \tag{C.4}$$

As shown in figures C.4(a) and C.4(b), the distributed arbiter implementation presents a smaller number control signals. However it requires a larger area to layout them across the circuit for a squared layout.

Another comparison, not so direct, is with non-arbitrated implementations based on error correction algorithms. In this kind of systems, we need an information code greater than the address only, to permit the inclusion of parity bits. The size of the transmitted data depends on the size of the system, the probability of loss accepted and the system activity.

For example, in [101], for 255 nodes operating at 12 x 107 events/second with a probability of loss of 1%, the system needs 32 bits. In the implementation presented here needs 25 bits only. As the signal routing is identical, this implementation demands less area than in [101].

Latency

The distributed AER arbiter is implemented using asynchronous state machines and, therefore, most of the states transitions depend on any indication of every node in the system. The minimum latency is 12 states and the maximum (in case of conflict) is $9\ log_2N + 12$ states. The maximum latency does not depend on the number of nodes in conflict but on their addresses, i.e. smaller (or bigger) is the addresses of at least 2 nodes greater is the latency.

In the centralized version, there is no need for state machines and, therefore the system can runs in a continuous fashion. As the circuit only depends of the responses given by the origin and target nodes, the total latency is in the worst case equal to latency from the distributed version.

Scalability

The arbiter circuit of the centralized implementation of AER is a circuit separated from the nodes of system. This means that when two or more IC are used to create a greater system, an external arbiter need to be added too, as in SCX system [97]. In this implementation, no extra hardware is needed, as long you provide address buses big enough to support all nodes.

Appendix D
Additional summation methods using timing modulations

It was demonstrated in chapter 4 that the asynchronous event spike coding functional principles lead to a natural and straightforward implementation of several arithmetic computations. In this appendix, it is shown the methods to implement the summation with other timing modulations.

D.1 PFM, PDM and PPM

When coding signals using Pulse Frequency Modulation (PFM), the information is represented by the number of spikes n transmitted over a timing window T_M. Considering a linear transformation, the summation is then given by:

$$f_{sum} = \sum_{i=1}^{N} (f_i - f_0) + f_0 = \frac{1}{T_M} \left[\sum_{i=1}^{N} n_i - (N-1)n_0 \right] \tag{D.1}$$

Knowing the number of signals N to be summed up and the number of spikes which represents the zero n_0 of the signal, a counter can be used to perform the summation. The counter, which is previously reset to an initial value, is incremented by each spike during the timing window. This process is quantized, synchronous, purely digital and power-hungry.

In the Pulse Delay Modulation (PDM) case, the information is coded in the time delay between two successive spikes. The summation is represented using the following equation:

$$\Delta_{sum} = \sum_{i=1}^{N} \Delta t_i - (N-1)\Delta t_0 \tag{D.2}$$

Distinctive methods can be used to realize this expression: an integrator with a leakage current corresponding to the zero value or using a Time-to-Digital Converter (TDC). The first outputs an analogue results in contrast to the digital output provided by the last method. Therefore this operation is essentially continuous (analogue method) or quantized (digital method), asynchronous, noisy but power efficient with the analogue method.

Finally, in the Pulse Position Modulation (PPM) method, the phase between a spike and a reference clock is used to code the signal the information. The expression for the summation operation is equal to equation D.2, with Δt_i representing the time difference of the spike and the clock rather than the difference for successive spikes. Therefore, methods similar to the PDM case can be used here. Other methods that outputs spikes as well will be detailed in the next section. Differently from the PDM, PPM is a synchronous method.

D.2 Examples of circuits for PPM

The use of PPM is used instead of PDM because we need a time window (regular sampling, synchrony) with this circuits. The reason is because we cannot allow more than two pulses from the same signal arrive during the computation. Also, only positive representations can be used with these circuits. As the computations are performed in analogue domain, the undesirable effect of error propagation from pulse to analogue and analogue to pulse conversions is present.

Summation circuit version 1

This circuit is a derivation of the circuit presented by John Harris in [122] used to compute weighted averages. In this circuit there are N current sources: one for each operand as shown in figure D.1(a) with a continuous current sink I removing charge from the capacitor. The incoming spikes control the switches operations in the following way: from the beginning of the cycle, all the signals ϕ are active, allowing all N current sources to charge the capacitor C and, therefore, increasing the voltage $V_{int}(t)$. On the detection of the spike from the operand k, the respective signal ϕ_k turns off. When all spikes have arrived, this current sink works alone, removing charge from the capacitor until it comes back to its initial state.

From figure D.1(b), the integrating voltage after k spikes is given by

$$\begin{aligned} V_{int}(\Delta t_k) &= \frac{I}{C}\left[N-(k-1)\right](\Delta t_k - \Delta t_{k-1}) + V_{int}(\Delta t_{k-1}) \\ V_{int}(\Delta t_N) &= \frac{I}{C}\sum_{i=1}^{N}\Delta t_i \end{aligned} \tag{D.3}$$

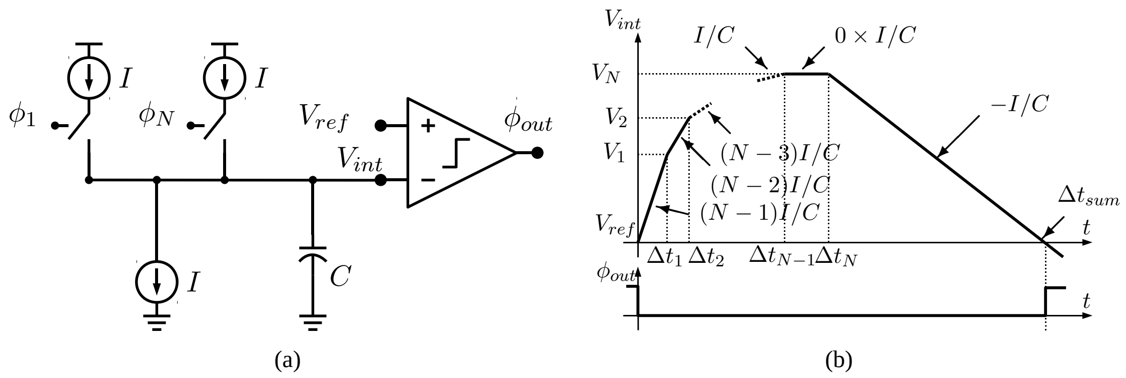


Figure D.1: *PPM summation circuit and waveform: version 1. (a) circuit to perform the summation of multiple signals. (b) waveform example.*

and the summation result is the time elapsed to charge up to $V_{int}(\Delta t_N)$ and discharge back to the reference voltage $V_{ref} = 0$ such as

$$\Delta t_{sum} = \sum_{i=1}^{N} \Delta t_i \tag{D.4}$$

This implementation has the advantage of the comparator output transition represents the true summation result, i.e., it does not present any summed constants to the result as in [122]. However the number of current sources (and power consumption) increase with the number of operators and only positive operands are allowed (1-quadrant summation).

Summation circuit version 2

This PPM summation circuit is also based on integrating voltages like the previous one. However it does not require the same amount of current sources as the previous circuit, as shown in figure D.2(a). Only one current source I_{up}, triggered by *any* incoming spike charges the capacitor C whilst another I_{dn} discharges it.

In fact this circuit can perform the summation operation on two different modes. In the first case the control signal ϕ_N is always set, meaning the integrating node is always being discharged by the current I_{dn}. For each input spike detected, the signal ϕ_P connects the current source to the integration node for a fixed time T. Being $I_{up} \gg I_{dn}$, the integrating node is charged during this time T. Because the current I_{up} is proportional (α) to the time Δt_k, the voltage on

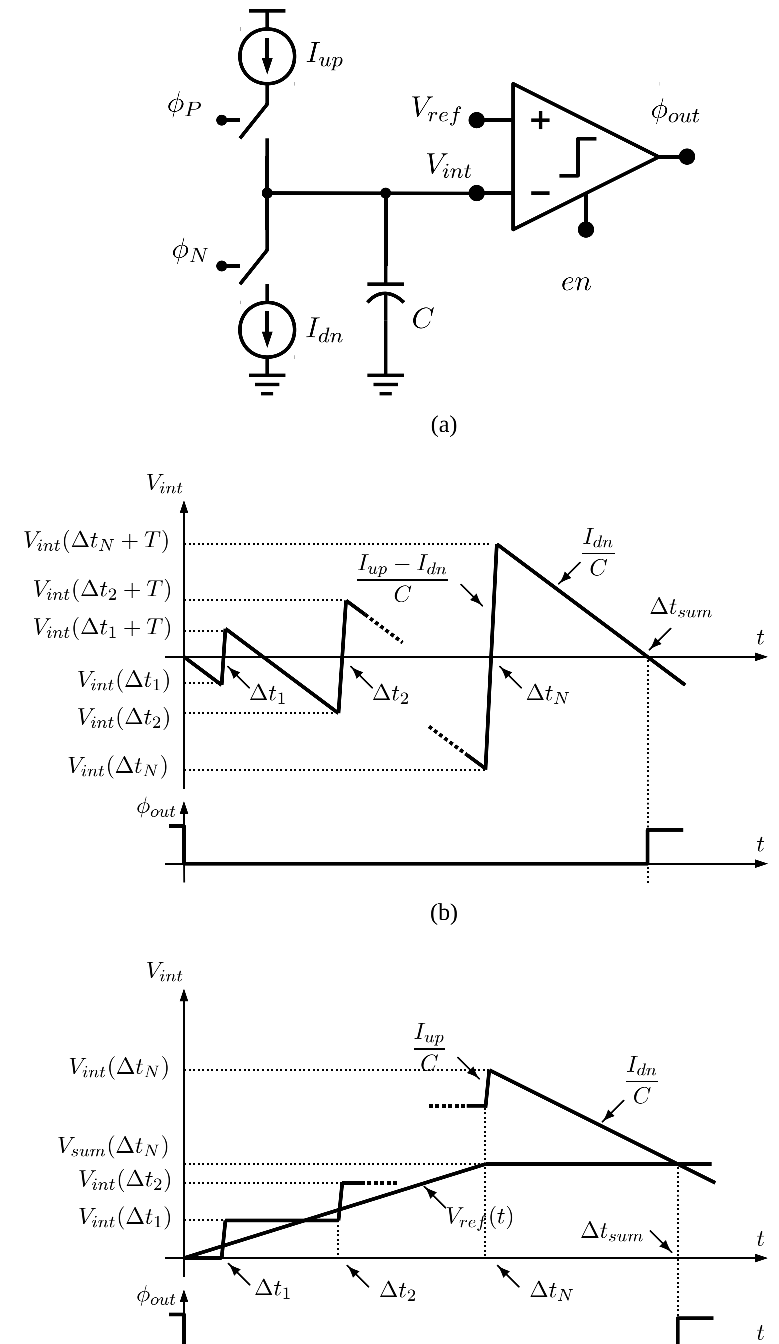


Figure D.2: *PPM summation circuit and waveform: version 2. (a) circuit to perform the summation of multiple signals. (b) waveform example for the case 1 and (c) for the case 2.*

the integrating node after k spikes is

$$\begin{aligned} V_{int}(\Delta t_k) &= V_{int}(\Delta t_{k-1}+T) - \frac{I_{dn}}{C}\Delta t_k \\ V_{int}(\Delta t_{k-1}+T) &= V_{int}(\Delta t_{k-1}) + \frac{T}{C} I_{up}(\Delta t_k) \\ I_{up}(\Delta t_k) &= \alpha \Delta t_k \end{aligned} \tag{D.5}$$

and the comparator output switches[1] at a time proportional to the summation of the incoming spikes as

$$\begin{aligned} \Delta t_{sum} &= \frac{\alpha T}{I_{dn}} \sum_{i=1}^{N} \Delta t_i \\ \Delta t_{sum} &= \sum_{i=1}^{N} \Delta t_i \text{ if } \alpha = \frac{I_{dn}}{T} \end{aligned} \tag{D.6}$$

Although this circuit has the advantage of can received a variable number of input spikes (number of operands), it still presents a high static current consumption. To reduce this current consumption, the second mode can be implemented using the same circuit. In this mode, the signal ϕ_N is set only after the last spike has arrived. This last spike also stops the comparator reference $V_{ref}(t)$ integration. In this mode, equations D.5 and D.6 can be rewritten as

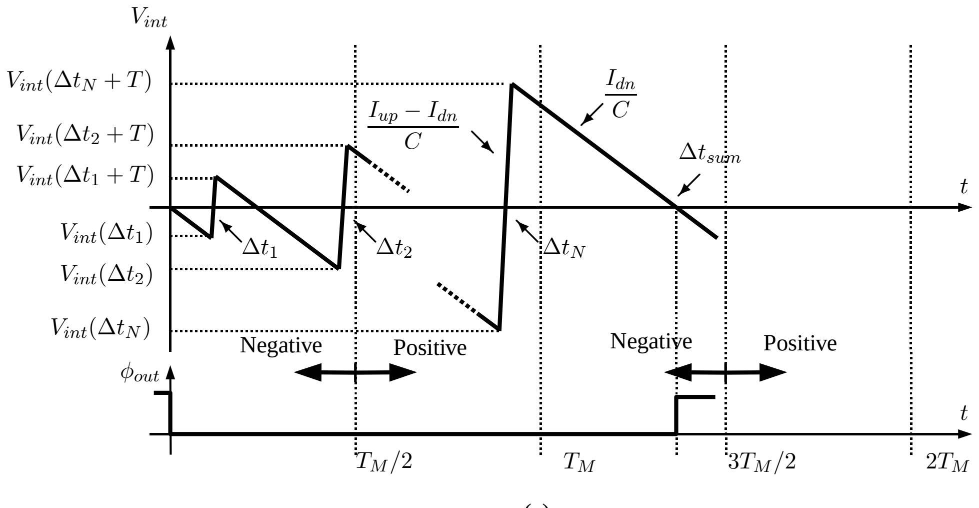
$$\begin{aligned} V_{int}(\Delta t_k) &= V_{int}(\Delta t_{k-1}+T) - \frac{T}{C} I_{up}(\Delta t_k) \\ I_{up}(\Delta t_k) &= \alpha \Delta t_k \end{aligned} \tag{D.7}$$

and

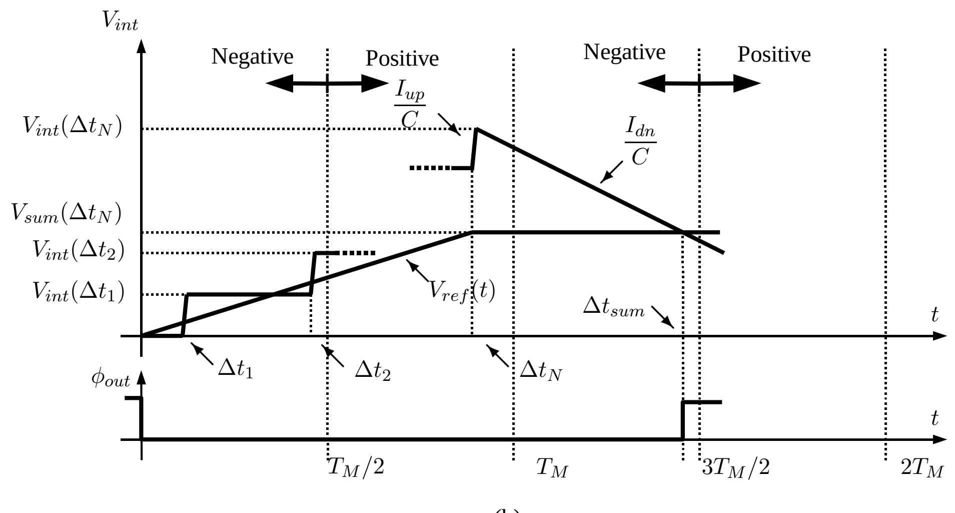
$$\begin{aligned} \Delta t_{sum} &= \Delta t_N + \frac{\alpha T}{I_{dn}} \sum_{i=1}^{N} \Delta t_i - \frac{C}{I_{dn}} V_{ref}(\Delta t_N) \\ \Delta t_{sum} &= \sum_{i=1}^{N} \Delta t_i \text{ if } \alpha = \frac{I_{dn}}{T} \text{ and } V_{ref}(\Delta t_N) = \frac{I_{dn}}{C}\Delta t_N \end{aligned} \tag{D.8}$$

respectively. Figures D.2(b) and D.2(c) show a 1-quadrant summation, but the method can be adapted to performed a 4-quadrant summation if one period T_M latency is allowed as in figures D.3(a) and D.3(a). In other words, the results appear only in next period.

[1]In fact the comparator outputs switches every time the integrating node voltage crosses the reference voltage V_{ref}. To avoid this, the comparator can can enable using the signal en active only after the last spike has arrived.



(a)



(b)

Figure D.3: *PPM quadrant summation summation circuit and waveform: version 2. 4-quadrant version of the summation methods using the circuit in figure D.2(a). (a) waveform example for the case 1 and (b) for the case 2.*

D.3 Summation expressions for the methods studied

Here are the expression derived from the communication methods presented in chapter 3. As already shown, the summation expression for the TSD modulation is given by:

$$x_{sum}(t) \approx z_R(t) = \delta \left(\sum_{k=1}^{N} n_{kp} - \sum_{k=1}^{N} n_{kn} \right) + z_R(t_0) \tag{D.9}$$

where n_{kp} and n_{kn} are the number of positive and negative spikes, respectively, received from the k^{th} operand after the initial instant t_0. Meanwhile, for the BSD modulation, this operation is given by:

$$x_{sum}(t) \approx z_R(t) = \delta \sum_{k=1}^{N} n_k - k_i N\,(t - t_0) + z_R(t_0) \qquad \text{(D.10)}$$

with k_i being the integrator gain. Differently from the TSD case, in BSD modulation the summation is a function of the current instant. For the ADM modulation, the expression is:

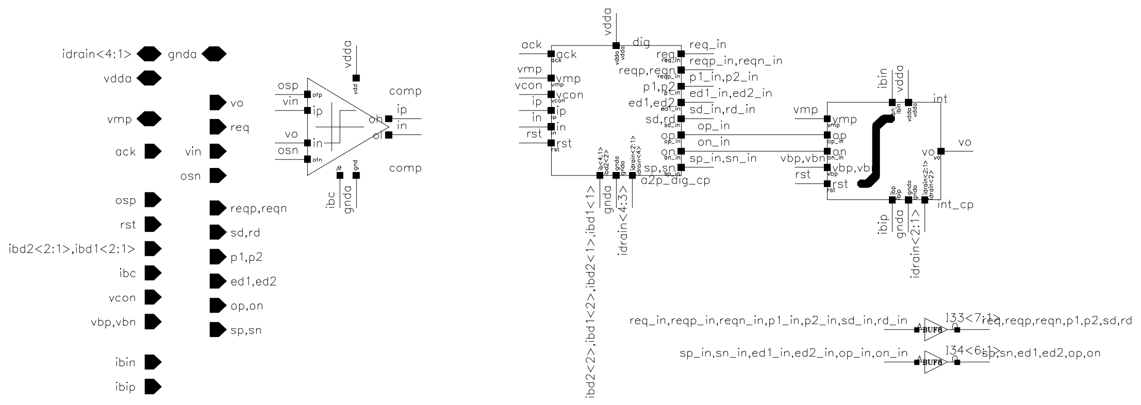
$$x_{sum}(t) \approx z_R(t) = k_i b \sum_{k=1}^{N} \left[\sum_{m_k=1}^{M} (-1)^{m_k+1} (t_{m_k} - t_{m_k-1}) \right] + z_R(t_0) \qquad \text{(D.11)}$$

meaning that the summation, as any individual signal, is a function of all spike timings and therefore can not be implemented as just the merge of all spikes as it is in the TSD modulation.

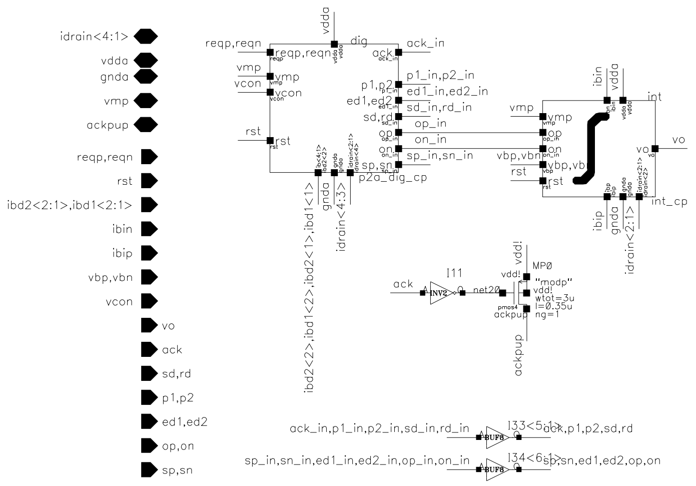
Appendix E
Schematics of the ASEC circuits

The circuits used in this thesis are presented in this appendix, with the exception of the AER circuitry which was designed and cordially lend by Giacomo's Invivieri Group at ETH Zurich and were adapted to the project by Simeon Braford. All circuits were designed in Cadence® Electronic Design Automation (EDA) suit.

Figures E.1(a) and E.1(b) presents the top schematic for the ASEC coder and decoder respectively. The main blocks on the coder are the compound comparator (figure E.2) the spike and pulse generators (figure E.3) and the integrator (figure E.7). The schematic of the digital circuits presented in spike and pulse generators are also presented. Figure E.4 shows the schematic of the arbiter while the schematic of the c-miller elements are also presented in figure E.5. The schematic of the delay circuits for the pulse generator block are presented in figure E.6.

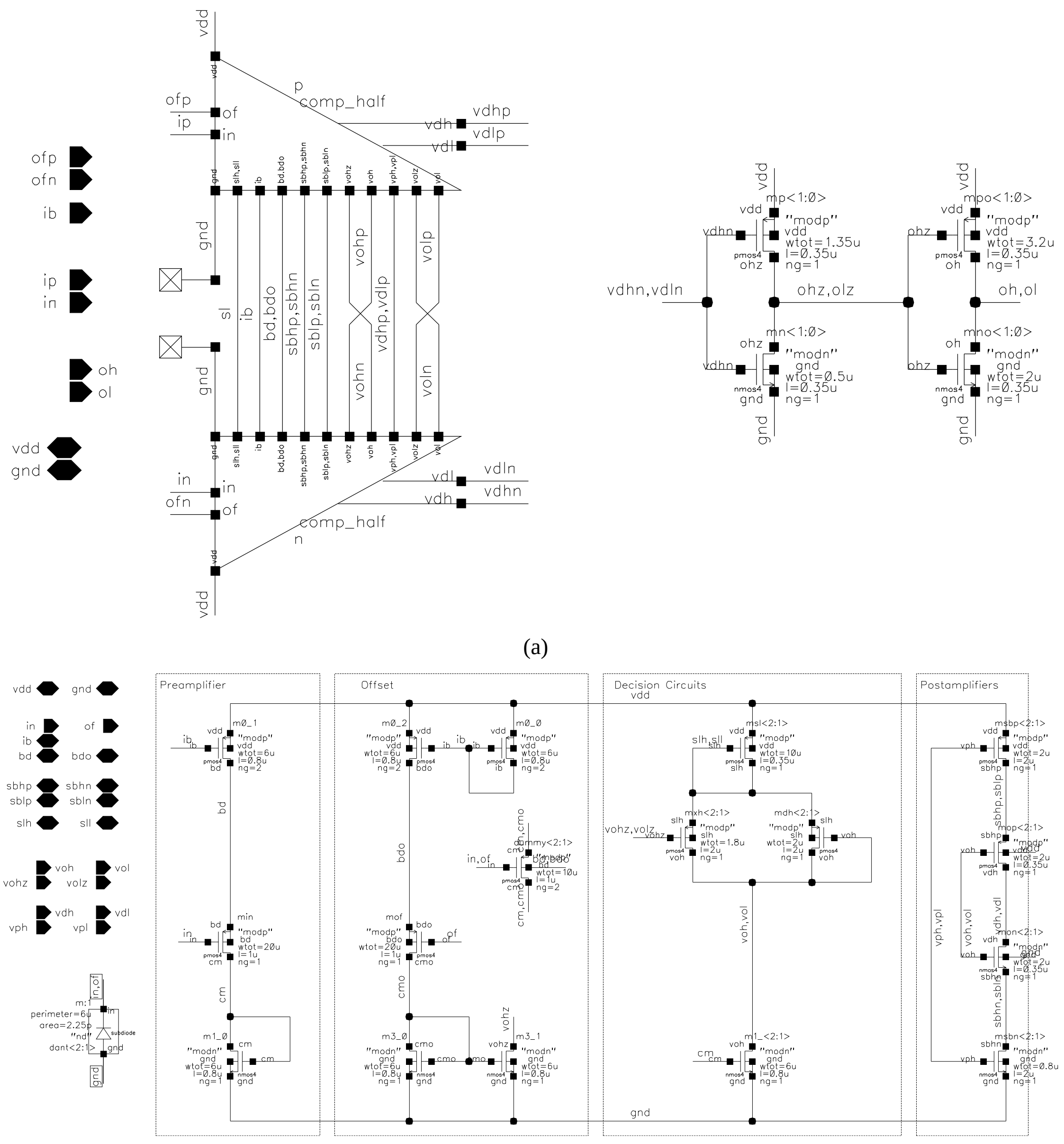


(a)



(b)

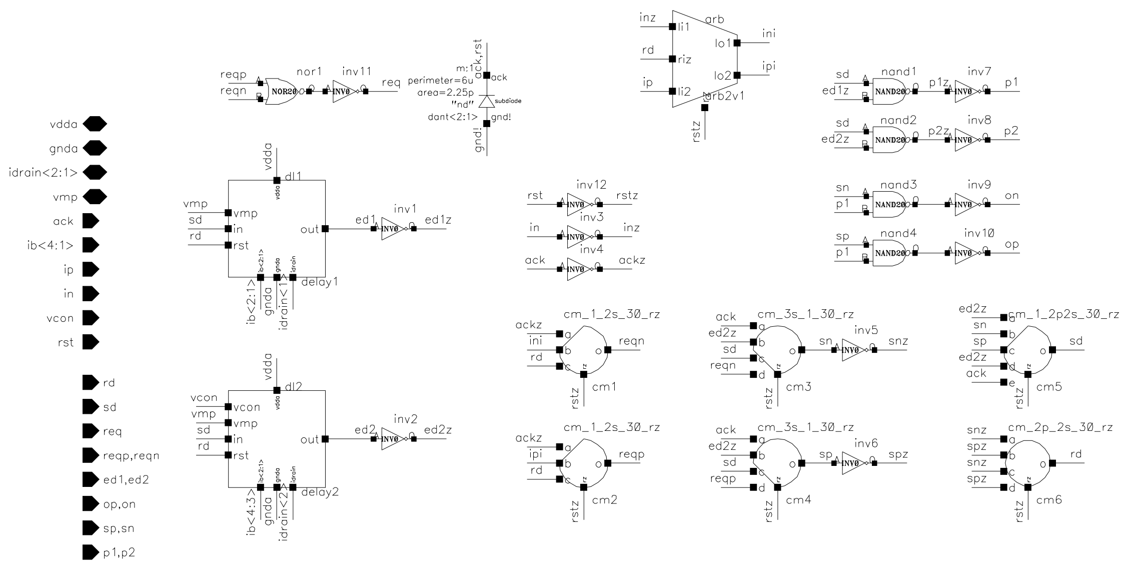
Figure E.1: *Schematics of the ASEC coder and decoder. The ASEC coder (a) is build from three fundamental blocks as show in chapter 5: Comparator (*comp*), digital control (*dig*) and integrator (*int*). Inverters are used as buffers for test. The decoder (b) two main blocks are the digital control (*dig*) and the integrator (*int*). The low pass filter was implemented off-chip. The pull-up transistor* MP0 *sets the global acknowledge signal.*



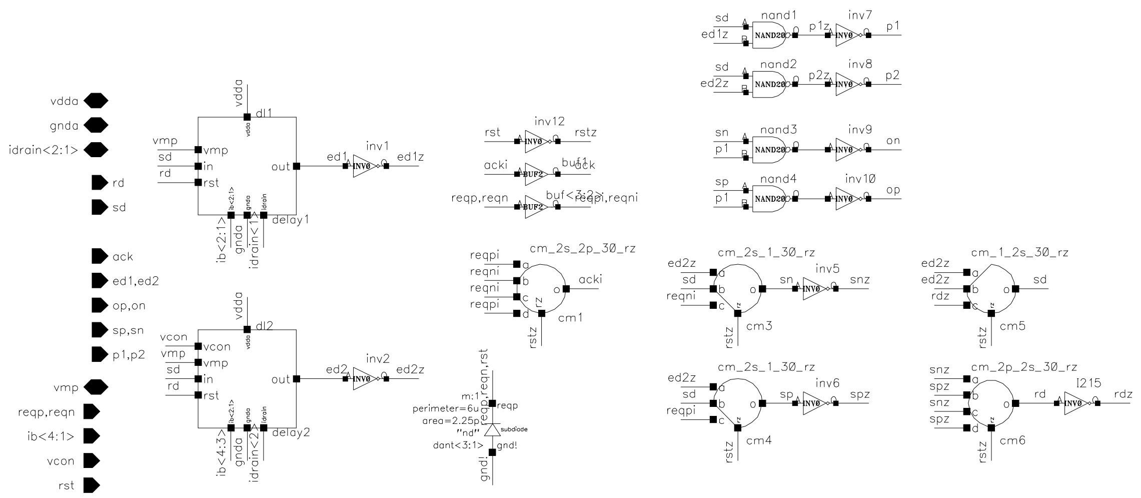
(a)

(b)

Figure E.2: *Schematic of the compound comparator. In order to guarantee the symmetrical design of the comparator (a), it was divide in two identical halves. Therefore, and modification on one side (the transistor sizes of the input differential pair, for instance) is performed on both sides. (b) shown the circuits inside each half. Components are grouped into the blocks of figure 5.2, i.e., pre-amplifier, decision circuit and output buffer. Offset group is the pre-amplifier for the* Δe_{th}.

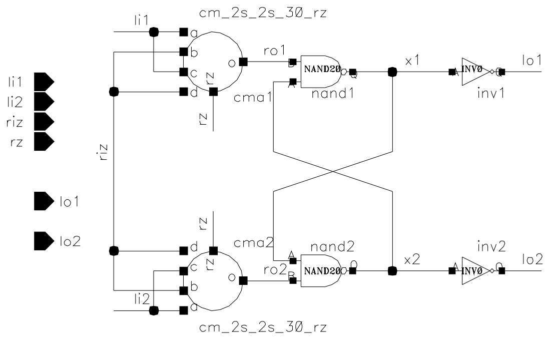


(a)

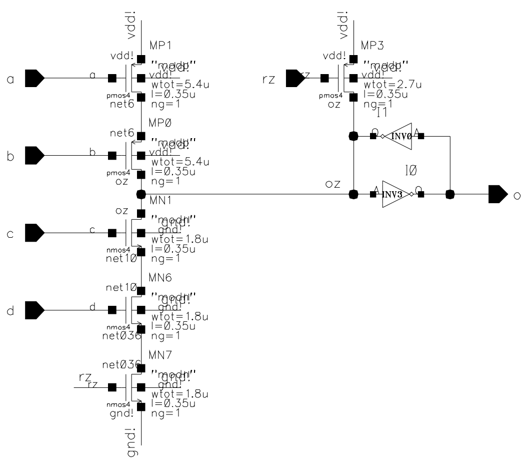


(b)

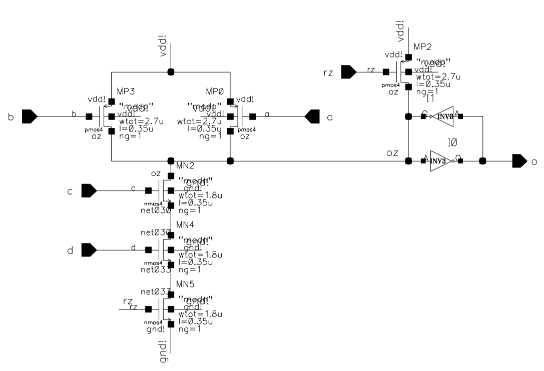
Figure E.3: *Schematics of the digital control circuits. The combination of the spike and pulse generator is show in (a) while pulse generator at the decoder in shown in (b). Common blocks include the delay blocks (*dl1 *and* dl2*) to generate the input integrator pulses. Digital control logic is asynchronous and were designed using the method presented by [142]. A set of different C-miller elements (*cm_*_*_30_rz*) were used to use the design method.*



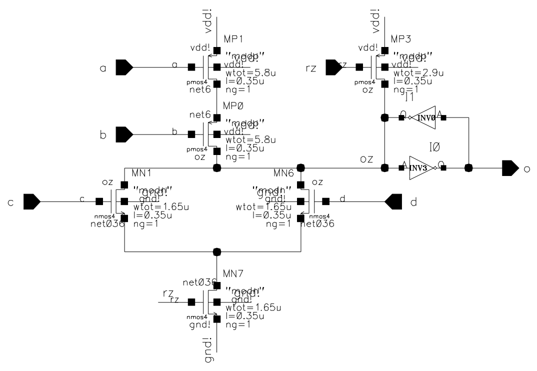
(a)



(b)

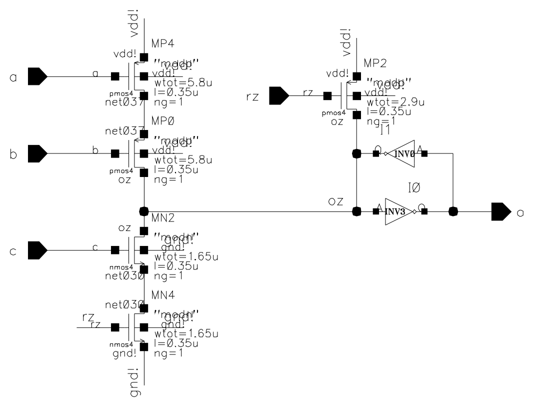


(c)

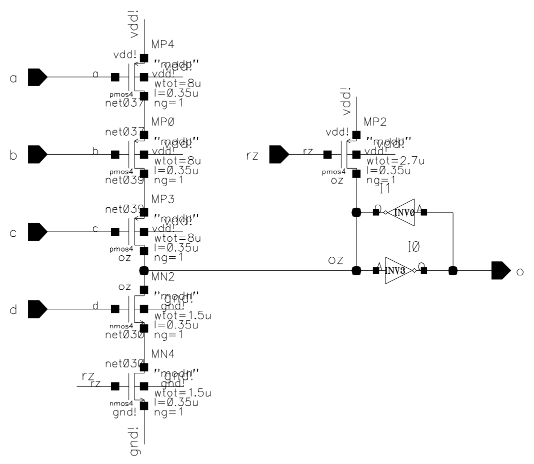


(d)

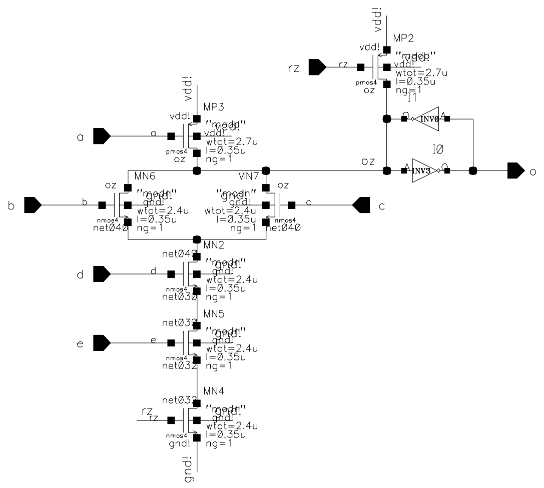
Figure E.4: *Schematics of the spike generator arbiter and C-miller elements. (a) presents the schematic of the arbiter in figure E.3(a). Other figures show the different c-miller elements used in the spike and pulse generator blocks. The elements in these figures ((b) to (d)) are, in order,* cm_2s_2s_30_rz, cm_2p_2s_30_rz *and* cm_2s_2p_30_rz.



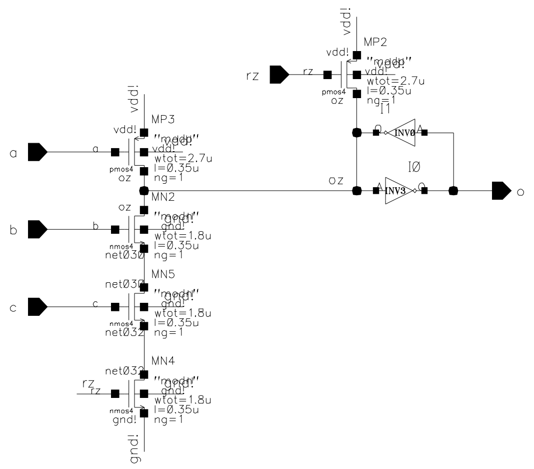
(a)



(b)

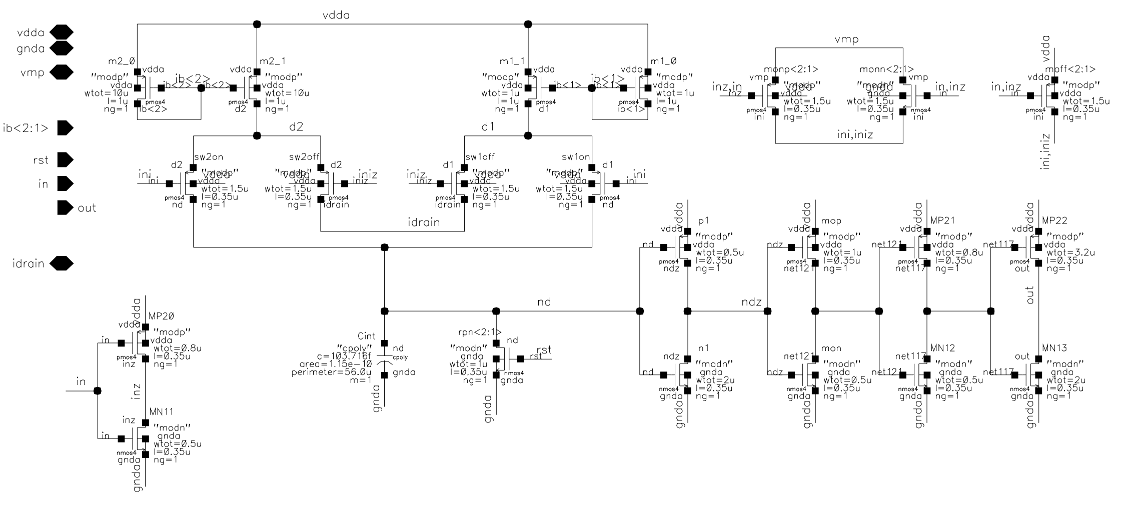


(c)

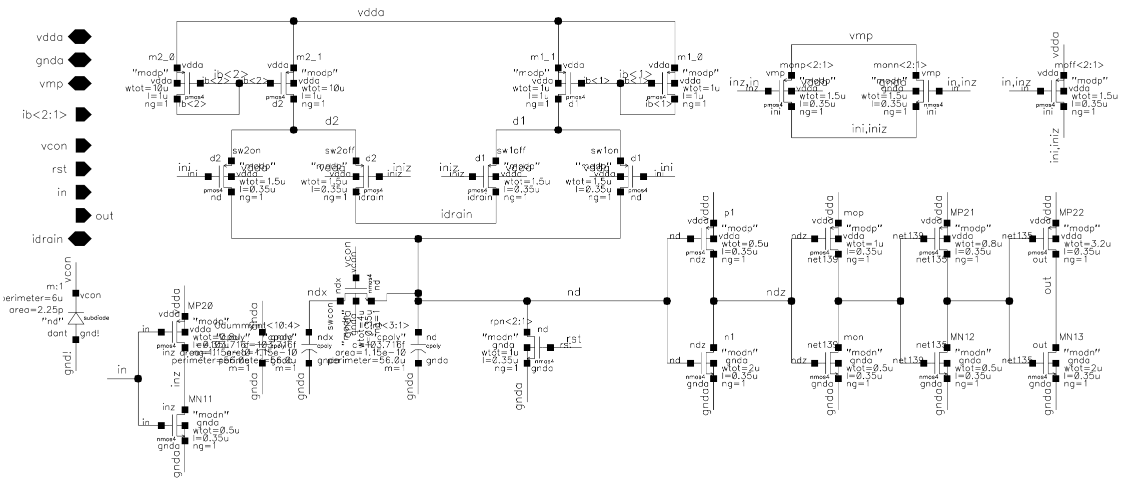


(d)

Figure E.5: *Schematics of the C-miller elements (cont.). Figures ((b) to (d)) show the c-miller elements* cm_2s_1_30_rz, cm_3s_1_30_rz, cm_1_2p2s_30_rz *and* cm_1_2s_30_rz *respectively.*



(a)



(b)

Figure E.6: *Schematics of the digital control delays circuits. These figures presented the schematic of the delays circuit for the the pulse generator blocks. These circuits works integrating a current into a capacitor C_{int}. This current is the summation of two externally supplied currents. Two different current mirrors provide a greater range of input current while keeping the transistors in the operational region. The voltage in the capacitor is connected to a chain of inverter which acts as a voltage buffer and reduce the transition times. (a) generates the pulse width T while the "refractory period" Δt_{min} is produced by the circuit in (b), which presents a programmable capacitor to increase the range of the interval periods between successive pulses.*

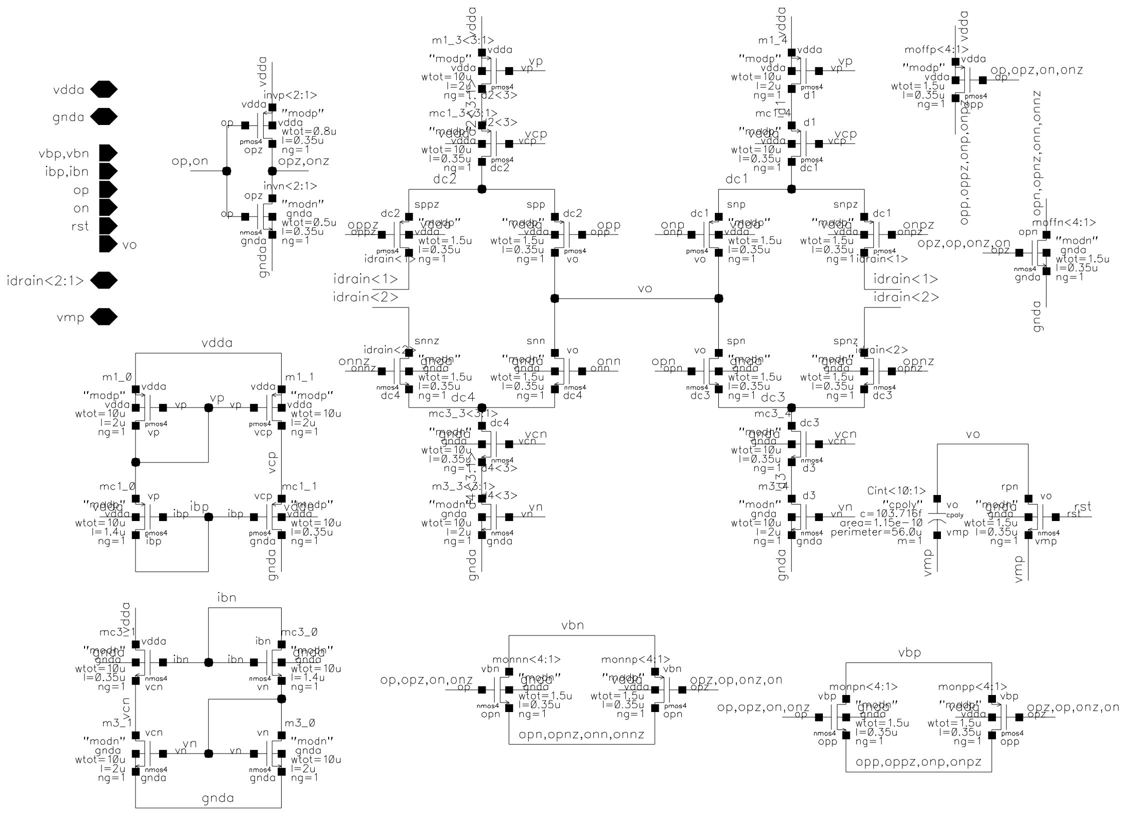


Figure E.7: *Schematic of the SCI integrator. The schematic is similar to figure 5.10 with the input current mirror and integration control signals bias being explicit. The input currents are both sink from the node* ipb *and source to node* ibn*. In the first IC, just one current was supplied being the complimentary current generated inside the integrator. In the second IC, both currents were controlled outside the IC to study the mismatch on the generation of this reference current. These currents generates the bias voltages* vp *and* vn *to the current mirrors and voltages* vcp *and* vcn *to the cascode transistors of these current mirrors. The gate voltage of the control switches* sp* *and* sn* *are set by the external voltages* vbn *and* vbp *respectively though the transistor* mo**. These voltages can be calibrated to compensate for the integrator mismatches. A 1 pF capacitor is placed at the output integrator node* vo *to hold the node voltage.*

Appendix F

Switched-capacitor integrator design

Due to the great sensibility to circuit mismatches presented in SCI topology another integrator topology was included in the second chip. This topology is based in switched-capacitor design techniques [151, 152] and a wide variety of designs for integrators have been studied and compared, as in [153] and [154], for instance.

The topology used in the second chip was proposed by [156] as a version of the basic stray-insensitive integrator [155] with offset-compensation, as seen in figure F.1. Other topologies for offset compensation have been studied, but the one in figure F.1(b) was used as the base configuration for integrator included in the second chip because of its simplicity.

The circuit implemented is the inverter integrator shown in figure F.2(a) and the phases sequence required to control the circuit is presented in figure F.2(b). The spike generator presented in 5.3.2 was adapted to output these non overlapped phases instead of the signals inc and dec.

Before either $c_1(t)$ or $c_2(t)$ are set by the comparator, the integrator samples the amplifier input offset in the capacitor C_o and resets the input capacitor C_s. Whenever any of the comparator outputs goes high, the new spike generator finishes the offset sampling by turning off the switches 3 and 5 and starts sampling the voltage v_d. The integrator output voltage is:

$$z(t) = z(t - \Delta t) + \frac{C_s}{C_i}\, v_d \tag{F.1}$$

where $v_d = v_{dec}$ if $\phi_{1p} = 1$ or $v_d = -v_{inc}$ if $\phi_{1n} = 1$. Therefore, these voltages are given by:

$$v_{dec} = \delta_d\, \frac{C_i}{C_s} \tag{F.2}$$

With this topology, the number of mismatch error sources are reduced. The timing is not

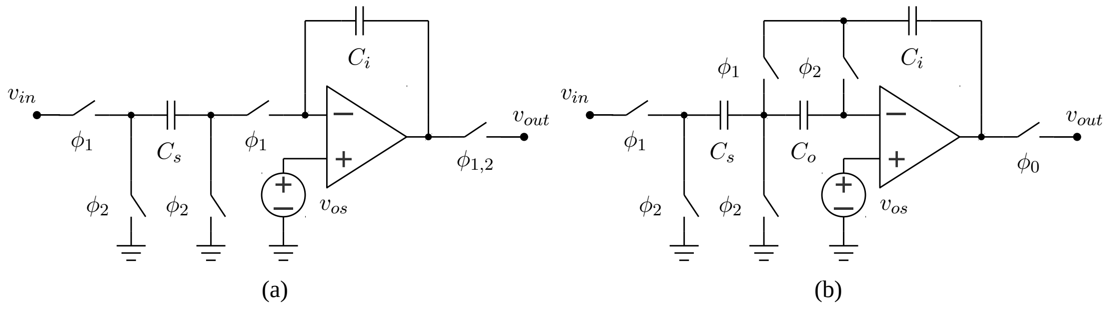


Figure F.1: *Switched-capacitor integrator topologies. Inverted versions of (a) stray-insensitive integrator [155] and (b) offset-compensated integrator. The output switch phase ϕ_0 is equal to ϕ_2 in [156] and equal to ϕ_1 in [153]. In this topology, the capacitor C_o stores the amplifier offset voltage v_{os} during the phase ϕ_1 for the compensation during phase ϕ_2.*

Component	Size	Unit
Switches PMOS	0.5 / 0.35	μm / μm
Switches NMOS	0.5 / 0.35	μm / μm
C_i	100	fF
C_o	100	fF
C_s	300	fF

Table F.1: *Components sizes of the switched-capacitor based integrator.*

critical, as long the circuit time constant is smaller than the phases width[1]. The coder/decoder capacitor mismatch of the SCI circuits is replaced by the mismatch of the *ratio* of the capacitors C_i and C_s, which is considerably less than the former.

Although the switched-capacitor based integrator is a know better option for mismatch problems, the implementation of this circuit showed to be more sensible to noise effects then the SCI implementation. In switched-capacitor circuits, the decoder output eventually saturates in one power supply rail. This effect is presented in figure F.3 where (a) represents the decoder output for the designed values of v_{inc} and v_{dec} and (b) is the best result obtained when changing both v_{inc} and v_{dec}[2].

[1]The time constant was designed to be five times smaller than the pulse width to obtain a final error $< 1\%$.

[2]Switched-capacitor based integrators were implemented in the second chip where all bias signals are controlled by 12-bit resolution DACs. In other words 1 LSB change in v_{inc} or v_{dec} was not enough to obtain a stable output for the decoder integrator.

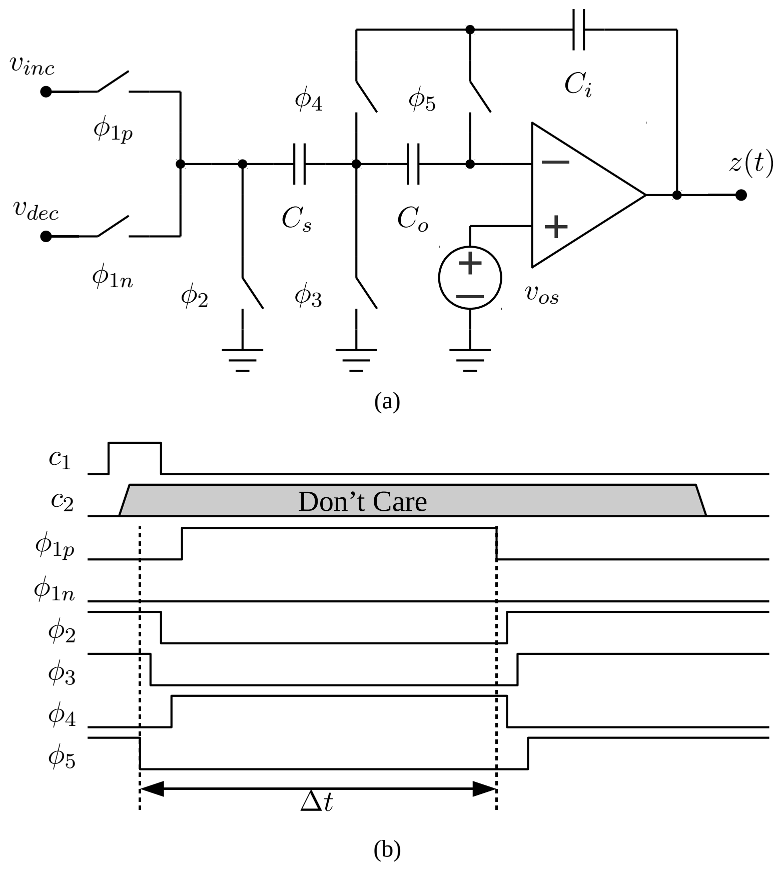


Figure F.2: *Switched-capacitor integrator. (a) Circuit schematic and (b) typical behaviour of the control phases and output signal. The output switch phase ϕ_0 is equal to ϕ_2 in [156] and equal to ϕ_1 in [153]. In this topology, the capacitor C_o stores the amplifier offset voltage v_{os} during the phase ϕ_1 for the compensation during phase ϕ_2.*

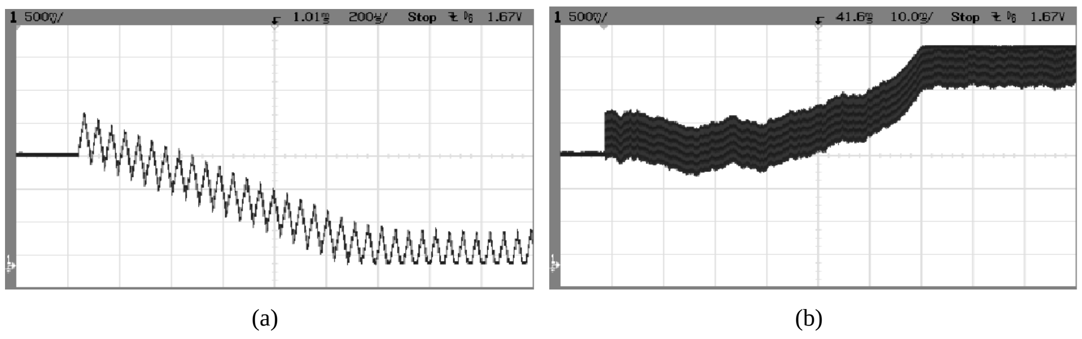


Figure F.3: *Switched-capacitor integrator divergence. (a) Decoder integrator output with designed values for $vinc$ and $vdec$ and (b) Decoder integrator output with $vinc$ and $vdec$ tuned (best fit). With the best fit settings, the integrator randomly saturates at different power rails.*

Appendix G
Publications

As the direct result of this thesis or in work associate with it, a series of academic papers were published in peer-reviewed conferences and journals. These papers are listed below and were included in here for completeness.

Journal papers

- L.C. Gouveia, T.J. Koickal, and A. Hamilton, “An Asynchronous Spike Event Coding Scheme for Programmable Analog Arrays,” IEEE Transactions on Circuits and Systems I: Regular Papers, 2010, pp. 1364-1367.
- T. Koickal, L. Gouveia, and A. Hamilton, “Bio-inspired Event Coded Configurable Analog Circuit Block,” Evolvable Systems: From Biology to Hardware, vol. 5216, 2008, p. 285-295.
- T.J. Koickal, L.C. Gouveia, and A. Hamilton, “A programmable spike-timing based circuit block for reconfigurable neuromorphic computing,” Neurocomputing, vol. 72, 2009, pp. 3609-3616.

Conference papers

- L.C. Gouveia, T.J. Koickal, and A. Hamilton, “Computation in communication: Spike event coding for programmable analog arrays,” Proceedings of 2010 IEEE International Symposium on Circuits and Systems, Paris: IEEE, 2010, pp. 857-860.
- L.C. Gouveia, T.J. Koickal, and A. Hamilton, “A CMOS implementation of a spike event coding scheme for analog arrays,” 2009 IEEE International Symposium on Circuits and Systems, Taipei: IEEE, 2009, pp. 149-152.
- L. Gouveia, T. Koickal, and A. Hamilton, “An Asynchronous Spike Event Coding Scheme for Programmable Analog Arrays,” 2008 IEEE International Symposium on Circuits and Systems, Seattle: IEEE, 2008, pp. 1364-1367.

- T.J. Koickal, A. Hamilton, and L.C. Gouveia, “Programmable Analog VLSI Architecture Based upon Event Coding,” NASA/ESA Conference on Adaptive Hardware and Systems, Los Alamitos, CA, USA: IEEE, 2007, pp. 554-562.
- T.J. Koickal, L.C. Gouveia, and A. Hamilton, “A programmable time event coded circuit block for reconfigurable neuromorphic computing,” Proceedings of the 9th international work conference on Artificial neural networks, San Sebastian, Spain: Springer-Verlag, 2007, pp. 430-437.

An Asynchronous Spike Event Coding Scheme for Programmable Analog Arrays

Luiz Carlos Gouveia, Thomas Jacob Koickal, and Alister Hamilton

***Abstract*—This paper presents a spike time event coding scheme for transmission of analog signals between configurable analog blocks (CABs) in a programmable analog array. The analog signals from CABs are encoded as spike time instants dependent upon input signal activity and are transmitted asynchronously by employing the address event representation protocol (AER), a widely used communication protocol in neuromorphic systems. Power dissipation is dependent upon input signal activity and no spike events are generated when the input signal is constant. Computation is intrinsic to the spike event coding scheme and is performed without additional hardware. The ability of the communication scheme to perform computation will enhance the computation power of the programmable analog array. The design methodology and analog circuit design of the scheme are presented. Test results from prototype chips implemented using a 3.3-V, 0.35-μm CMOS technology are presented.**

***Index Terms*—Address event representation (AER), analog very large scale integration (VLSI), array signal processing, asynchronous delta modulation, field programmable analog arrays (FPAAs).**

I. Introduction

A FIELD programmable analog array (FPAA) consists of several configurable analog blocks (CABs) that are programmed to perform analog computation and signal processing functions [1]–[9]. An important design problem in an FPAA is the communication of analog signals between CABs. Previous implementations of FPAAs used crossbars or switch matrices [3], [4]. These approaches suffer from signal distortion due to voltage drops, parasitic capacitances along the wires and switches, and through signal interference [4]. Some of these issues can be reduced using floating gate technology [7] or by limiting the communication range to neighboring CABs [8], for instance.

In an alternative approach [9], pulse-width modulated signals were used for transmitting analog information in the array. However, this approach requires a global clock signal to synchronize the transmission leading to greater power consumption and issues such as clock skew, noise, metastablity, and high levels of electromagnetic interference (EMI) compared to asynchronous systems [10].

Recently, event driven signal processing techniques based upon signal dependent sampling strategies has received increasing research interest [11]–[13]. A motivation for this type of study is drawn from biology where the brain processes signals in the analog domain and transmits them as time events [14]. In event-driven sampling schemes, the intersampling intervals are quantized instead of the signal amplitude and an event is triggered whenever the input signal crosses prespecified levels along the amplitude domain [15]. In [16], signal recovery from time encoded signals has been demonstrated and this method has been extended to recover information from spiking neurons [17].

In this paper we present a spike event coding scheme for transmitting analog signals between CABs in a programmable analog array. In this scheme analog signals are encoded as spike timed events and are transmitted between CABs using the address event representation (AER) protocol [18]. There are many benefits in using an event-based coding scheme. First, event coding transmits signals as asynchronous, essentially digital spikes triggered by input signal activity. This reduces power consumption and results in better resource utilization when signal activity is low or static. Second, in contrast to synchronous signal processing, event-based processing benefits from immunity to metastable behavior, low crosstalk, and freedom from clock skew. Third, event-based processing transmits signals as digital spikes and hence are easy to route, not only between CABs but also between multiple chips allowing greater scalability.

In general, programmable analog arrays perform computations using programmed configurable analog blocks, while the role of the communication interconnect is to route signals between programmed analog blocks. However, the computation power of a programmable array can be enhanced if the communication channel can perform computation without additional overheads, as performed by the spike event communication scheme presented here.

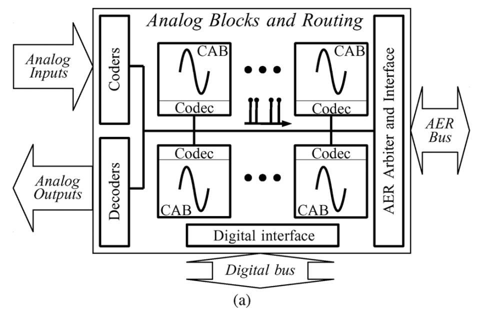
This communication scheme allows a set of fundamental arithmetic operations to be performed, for example gain, negation, and summation. These fundamental operations are performed simply by programming parameters of the communication channel. More complex operations can be implemented either by programming the channel and/or by programming the CABs. The computation capability of this communication scheme can enhance the computing power of the programmable array without using additional hardware.

The paper is organized as follows. The architecture is described in Section II. In Section III, the spike event coding

Manuscript received May 05, 2010; revised September 13, 2010; accepted September 27, 2010. This work was supported by the Engineering and Physical Sciences Research Council (EPSRC) under research grant (EP/C015789/1) to the University of Edinburgh. This paper was recommended by Associate Editor V. Gaudet.

The authors are with the Institute for Micro and Nano Systems, Joint Research Institute for Integrated Systems, School of Engineering, University of Edinburgh, Edinburgh EH9 3JL, U.K. (e-mail: l.gouveia@ed.ac.uk; thomas.koickal@ee.ed.ac.uk; alister.hamilton@ee.ed.ac.uk).

Digital Object Identifier 10.1109/TCSI.2010.2089552



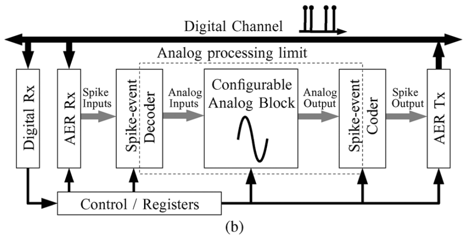


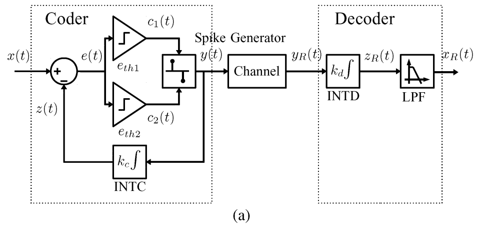
Fig. 1. (a) Architecture diagram. The array of configurable analog blocks (CABs) is connected using an asynchronous digital channel. (b) Block diagram of the spike event communication interface between a CAB and the digital channel.

scheme is introduced and the design methodology is explained. The computation properties of the communication scheme are introduced in Section IV. The circuit design and analog VLSI implementation of the spike event coding scheme are presented in Section V and Section VI contains experimental test results from a prototype chip. Section VII contain a discussion of the method and conclusions.

II. Architecture Description

In the proposed architecture shown in Fig. 1(a), CABs are interconnected using an asynchronous *spike event communication scheme* [19]. The communication scheme consists of spike event coders and decoders to interface each CAB to an asynchronous digital channel as shown in Fig. 1(b). For each transmission, the spike event coder of the transmitter CAB encodes the analog signals into asynchronous discrete amplitude signals (spike events) which are then delivered to receiver(s) CAB(s), where the decoder(s) convert the signal back to analog representation.

The transmission of these spike events is implemented using a digital bus instead of analog interconnections. Because events are asynchronous, the *address event representation* (AER) protocol [18]—widely used in neuromophic designs—is an appropriate choice. The asynchronous nature of the AER protocol preserves the information conveyed in the time difference between events. It can also handle possible collisions of simultaneous spikes.



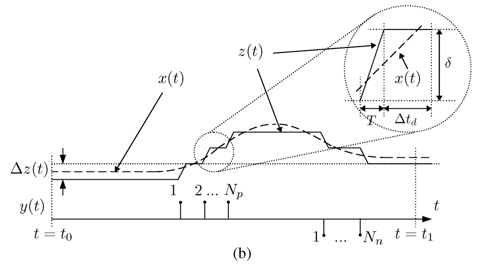


Fig. 2. (a) Event coding block diagram and (b) typical waveform diagram. In the inset, δ is the tracking step, T is the spike "width" and Δt_d is the time interval between successive spikes.

III. Spike Event Coding Scheme

The spike event coding scheme is shown in Fig. 2(a). This scheme is based on asynchronous delta modulation [20] and schemes that use the principle of irregular sampling as in [11], [13] and [15], where it was used to implement asynchronous A/D converters.

The spike event coder operates by generating a signal similar to the input signal. In other words, a feedback signal $z(t)$ is forced to track the input signal $x(t)$ by bounding the error $e(t)$ between them:

$$e(t) = x(t) - z(t) = x(t) - k_c \int y(t)dt \tag{1}$$

where $y(t)$ is the coder output. In this paper, this output is represented either by positive or negative pulses with a short and fixed duration (spikes). These spikes are produced by the spike generator and transmitted both to the communication channel and to the input of the feedback integrator (INTC).

Each positive or negative spike results in an incremental or decremental change δ (tracking step) at the output of the feedback integrator:

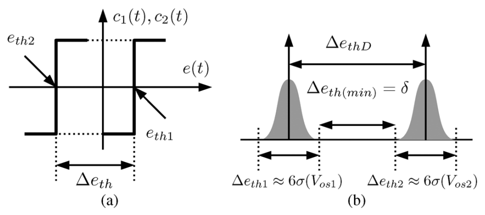
$$\Delta z(t) = \delta(N_p - N_n) \tag{2}$$

where N_p (N_n) is the number of previous positive (negative) spikes since t_0 as shown in Fig. 2(b).

On the receiver side, the analog output $x_R(t)$ is given by

$$x_R(t) = LPF\left(z_R(t)\right) \approx \overline{z_R(t)} = k_d \overline{\int y_R(t)dt} \tag{3}$$

with the decoder low-pass filter (LPF) removing high-frequency harmonics and averaging the signal $z_R(t)$, which is the result of integrating the incoming spikes $y_R(t)$.



$c_1(t), c_2(t)$
e_{th2}
$e(t)$
e_{th1}
Δe_{th}
(a)
Δe_{thD}
$\Delta e_{th(min)} = \delta$
$\Delta e_{th1} \approx 6\sigma(V_{os1})$
$\Delta e_{th2} \approx 6\sigma(V_{os2})$
(b)

Fig. 3. Comparator thresholds design. (a) Comparator transfer functions and (b) comparator thresholds variations Δe_{th1} and Δe_{th2} used to calculate the designed threshold difference Δe_{thD}.

Considering an ideal channel, i.e., $y_R(t) = y(t)$, the spikes are also transmitted to the decoder integrator INTD of the receiver CAB. The gain k_d of this integrator may be the same as the coder feedback integrator (INTC) k_c. Therefore, $x_R(t)$ is

$$x_R(t) \approx k_c \overline{\int y(t)dt} = \overline{z(t)} = \overline{x(t) - e(t)}. \tag{4}$$

Therefore, the maximum difference between the decoder output $x_R(t)$ and coder input $x(t)$ is $|e(t)|_{\max}$, which is defined by the specification of the system resolution:

$$|e(t)|_{\max} = \delta = \frac{\Delta x(t)_{\max}}{2^{N_B} - 1} \tag{5}$$

where $\Delta x(t)_{\max}$ is the input dynamic range and N_B is the desired resolution in bits.

In an AER system, one of the most important parameter is the output spike frequency. For the chosen coder, the output spike frequency is a function of the magnitude of the input derivative

$$f_{\text{spike}} = \frac{|\dot{x}(t)|}{\delta}. \tag{6}$$

From (6), we see that this event coding scheme presents no output activity when the input signal is constant and does not exhibit self-oscillatory behavior as in [10], [20]. This characteristic is beneficial in a programmable analog framework where significant number of slow or non-active signals, like bias signals may be present.

Next, we discuss the design process of the spike event coder and decoder.

A. Design

The first step in the design of the event coder is to design the comparators of Fig. 2(a), i.e., the threshold limits e_{th1} and e_{th2}, as shown in Fig. 3(a). The difference between the comparators thresholds $\Delta e_{th} = e_{th1} - e_{th2}$, as shown in Fig. 3(b), define the tracking step δ.

Ideally, $\Delta e_{th} = \delta$. However, due to process mismatches, comparators offsets V_{os1} and V_{os2} may vary from the designed value Δe_{thD} [21]. Hence, the *actual* Δe_{th} is bounded by

$$\Delta e_{thD} + 3\sigma_{os} \geq \Delta e_{th} \geq \Delta e_{thD} - 3\sigma_{os} \tag{7}$$

where $\sigma_{os} = \sigma(V_{os1}) + \sigma(V_{os2})$. $\sigma(V_{os1})$ and $\sigma(V_{os2})$ are the standard deviations of the comparator offsets V_{os1} and V_{os2}, respectively. Therefore, the comparators thresholds difference is designed to meet the specification

$$\Delta e_{thD} \geq \delta + 6\sigma_{os}. \tag{8}$$

The design of the integrator gain k_c (and k_d) is also based on the tracking step

$$k_c = \frac{\delta}{T} \tag{9}$$

where T is the width of the pulses from the spike generator block. This parameter is designed according to the input signal and the AER system characteristics.

In order to reduce the overload of the communication channel, this implementation of the spike generator sets a minimum period for the interval between two successive output spikes. This "refractory period" is given by $\Delta t_{d(\min)} = 1/f_{spike(max)}$. Setting the period as a multiple of the spike "width" $\Delta t_{d(\min)} = kT$ and using the specification of the maximum derivative of the input, the spike width is determined

$$T = \frac{\delta}{(k+1)\,|\dot{x}(t)|_{\max}}. \tag{10}$$

The definition of $\Delta t_{d(\min)}$ also provides an estimative about the limitations of the input signal. For instance, from (10), the frequency of an input sine wave $x(t) = A\ \sin(\omega_{\text{in}} t)$ is:

$$f_{\text{in}} = \frac{\omega_{\text{in}}}{2\pi} = \frac{1}{2\pi}\frac{\delta}{A} f_{\text{spike(max)}}. \tag{11}$$

In other words, once the system parameters are defined, the maximum input frequency is inversely proportional to the signal amplitude. Whenever the input frequency is greater than the value defined by (11), the system will present slope overload. The slope overload refers to the maximum rate that the coder can update the feedback signal $z_R(t)$.

Finally, the pole of the decoder first-order low-pass filter (LPF) is a key design parameter as it improves the resolution by attenuating the undesirable out of band high-frequency harmonics generated during the decoding process. Ideally, the filter should provide total rejection of out of band harmonics with zero in-band attenuation. However, the practical implementation of this characteristic is unrealizable. The actual design of the filter is a tradeoff between the amount of harmonic reduction and distortion caused by phase shift of each component frequency. In practice, the cutoff frequency of the low-pass filter ω_p is designed to be greater than or equal to the input signal bandwidth.

The filter is also a key factor for the delay between the coder input $x(t)$ and the decoder output $x_R(t)$. By choosing the pole to be $\omega_p = \omega_{\text{in}}$, for instance, the filter imposes a 45° phase shift from the signal $z_R(t)$. When applying low-frequency input signals, the delay due to phase shift is greater than delays due to the modulator loop, $\Delta t_{d(\min)}$, and AER arbitration process. For instance, the conversion of a tone signal ($\omega_{\text{in}} = 2\pi 4$ kHz) using 8-bit resolution imposes $\Delta t_{d(\min)} \approx 300$ ns $\ll 31.2$ μs (45° phase shift).

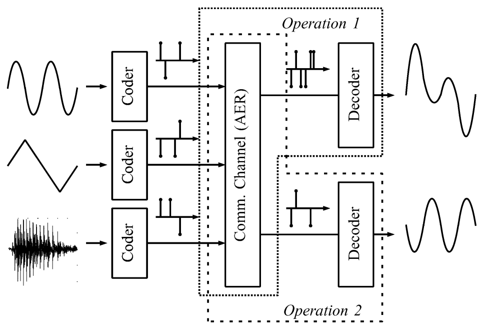


Fig. 4. Computation in the communication scheme. Analog signals are converted into spikes by the coders and arithmetic computation may be performed by the AER and decoder combination. The type of computation performed is defined by programming the AER routing and decoder parameters.

IV. Arithmetic Operations

The spike event coding scheme allows not only communication between CABs, but it is also capable of performing computation without additional hardware. Arithmetic operations are performed by configuring a combination of decoder and/or channel routing parameters. Fig. 4 illustrates the basic concept. This intrinsic computation capability allows a simpler implementation than other pulse-based approaches, for example [22].

In this section, we show a basic set of computations performed by this scheme and, for each of these operations, we show snapshots of chip results. From this set of basic computations more complex functions can be implemented. The following results were obtained for a 4–bit resolution design. The output waveforms shown correspond to the decoder filter input $z_R(t)$. The details of the chip implementation are shown in Section V.

A. Gain

A common operation in any analog processing is to amplify a signal, i.e., provide a gain G to the signal

$$x_{R_{\text{gain}}}(t) = G \times x(t). \tag{12}$$

The gain may increase (amplification) or decrease (attenuation) the amplitude of the signal. Both gain types are possible using the proposed architecture.

The expression in (2) is valid for both coder and decoder blocks. However, if coder and decoder are designed to present different tracking steps δ, then the decoder output will be proportional but not equal to the coder input. This gain is given by the ratio of both tracking steps amplitudes

$$G = \frac{\Delta z_{R_{\text{gain}}}(t)}{\Delta z(t)} = \frac{\delta_D}{\delta_C}. \tag{13}$$

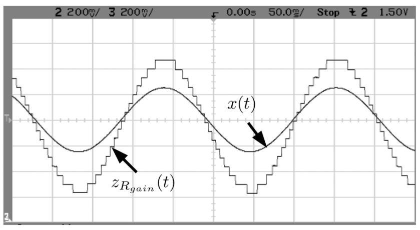


Fig. 5. Gain operation. The input sine wave signal $x(t)$ is scaled by a factor of $G = 2$ in this example, producing the decoder signal $z_{R_{\text{gain}}}(t)$. The gain operation is obtained by changing the ratio of the tracking steps of the coder δ_c and decoder δ_d.

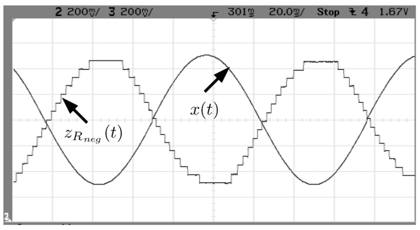


Fig. 6. Negation operation. Snapshot of the input sine wave $x(t)$ and the negated signal $z_{R_{\text{neg}}}(t)$. The negation is performed by the interchange of the positive and negative spikes using the AER router.

Fig. 5 shows chip result for an amplification of a sine wave by a factor of 2.

B. Negation

A second fundamental operation on analog signals is changing their polarity, i.e., signal negation

$$x_{R_{\text{neg}}}(t) = -x(t). \tag{14}$$

According to (2), the signal value at any instant t (knowing the initial condition of coder and decoder) is a function of the number of events transmitted and their type. By setting the AER router to interchange the addresses of positive and negative spikes, the signal at the decoder output $x_R(t)$ is a negated version of the input signal $x(t)$

$$\Delta z_{R_{\text{neg}}}(t) = -\Delta z(t) = \delta(N_n - N_p). \tag{15}$$

An example using a sine wave signal is shown in Fig. 6.

C. Summation

Beyond these unary operations, arithmetic operations involving two or more signals are also required by generic systems. The fundamental arithmetic operation involving multiple signals is the summation of these signals:

$$s(t) = \sum_{i=1}^{j} x_i(t). \tag{16}$$

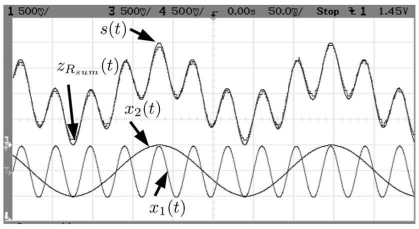


Fig. 7. Summation operation. Summation of two input sine waves, $x_1(t)$ and $x_2(t)$, with the same amplitude but different frequencies (23 and 4.7 Hz) and their respective summation $z_{R_{\rm sum}}(t)$. $s(t)$ is the ideal summation of the two signals.

Again considering (2), the summation signal $s(t)$ of j operators is given by

$$s(t) = x_{R_{\rm sum}}(t) = \delta\left(\sum_{i=1}^{j} N_{pi} - \sum_{i=1}^{j} N_{ni}\right) + x_{R_{\rm sum}}(t_0) \tag{17}$$

where N_{pi} and N_{ni} are the number of positive and negative spikes, respectively, received from the ith operand after the initial instant t_0.

The chip results showing the summation of two sine wave signals $x_1(t)$ and $x_2(t)$ are shown in Fig. 7, together with the decoder integrator output $z_{R_{\rm sum}}(t)$. The signal $s(t)$ represents the ideal summation result. Subtraction operations may be performed by combining the summation and negation operations outlined above.

These basic operations may be combined to compute more complex computations in a programmable analog array. For example, the weighted sum operation commonly used in artificial neural networks

$$f(t) = \sum_i w_i x_i(t) \tag{18}$$

is a combination of summation and gain operations.

All of the operations above are performed using asynchronous spike event coded signals and by AER router and decoder programming. However, the range of possible computations is expanded when the communication scheme is combined with the functionality of the CABs [23]. For instance, if CABs were designed to perform logarithmic compression and exponential expansion, as in [1], both multiplication and division can be implemented using the summation operation according to the logarithmic property

$$x_{R_{\rm mult}}(t) = \prod_i x_i(t) = \exp\left(\sum_i \ln x_i(t)\right). \tag{19}$$

Since the AER protocol is used to transmit spike events, collisions during access to the channel are possible and a 1-persistent arbiter is used to resolve them. Collisions lead to an error in the summation process. This is studied in more detail in [19].

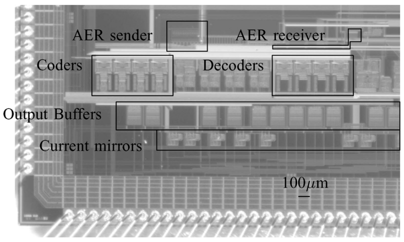


Fig. 8. Second test chip photograph. The test chip is pad-limited and the circuitry occupies a quarter (approximately 5 mm^2) of the die. Highlights show four coders, four decoders, output buffers, current mirrors, AER sender, and AER receiver. The area of each coder is approximately 0.03 mm^2 and each decoder (with no LPF) is approximately 0.02 mm^2.

TABLE I
SECOND TEST CHIP CHARACTERISTICS

Parameter		**Value**	**Unit**
Technology		CMOS 0.35μm 1P4M	-
Power Supply		3.3	V
Area	Coder	0.03	mm^2
	Decoder	0.02	
	AER	0.025	
	Circuitry	4.8	
	Chip	29.25	
Power Consumption	Coder	400	μW
	Comparator	340	
	Decoder	50	
	Chip	2700	

V. Analog VLSI Implementation

The spike event coder and decoder were implemented in a first chip for validation of the spike event coding scheme [24]. A second test chip, capable of performing computation in addition to communication, was implemented containing an array of four coders, four decoders and an AER router. A chip photograph of this second test chip is shown in Fig. 8 and a summary of the characteristics of this chip is presented in Table I.

In this section, we describe the spike event coder and decoder circuits implemented on both chips: comparators, spike generator, and integrators. The coder and decoder integrators are implemented using the same circuit design. The decoder LPF was implemented off-chip, using an offline digital filter.

A. Comparators

The comparators were implemented using a preamplifier (PA) followed by a decision circuit (DC) and an output buffer (OB) [25]. To provide the required Δe_{th}, capacitive or resistive dividers can be used at the comparator input nodes. However, these dividers compromise the input impedance of the circuit.

Another method to provide Δe_{th} is to implement offset comparators. Composite transistors can be used to provide the offset [26]; however, this topology suffers from low input dynamic range. A programmable offset can also be generated by another preamplifier which provides a respective $\Delta I_{\rm off}$ on the decision circuit input [27]. Both impedance dividers and offset comparators allow continuous resolution values to be used.

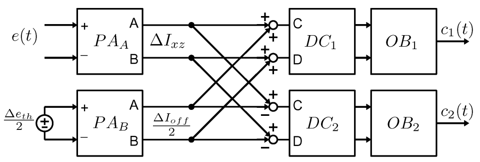


Fig. 9. Block diagram of the compound comparator. Preamplifier PA_A outputs a current ΔI_{xz} according to the error signal $e(t) = x(t) - z(t)$, while preamplifier PA_B generates a fixed offset current $\Delta I_{\text{off}}/2$. PA_B outputs are added or subtracted from PA_A outputs and the results are applied to the respective decision circuits (DC) and output buffers (OB).

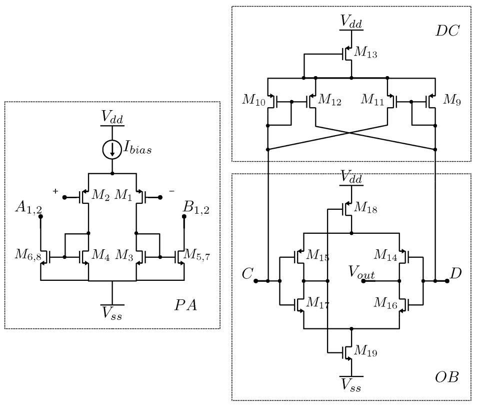


Fig. 10. Circuit schematic of each of the compound comparator blocks. Transconductance preamplifier (PA, left), decision circuit (DC, top right) and output stage (OB, bottom right).

In this implementation, both outputs $c_1(t)$ and $c_2(t)$ are generated by a compound comparator shown in Fig. 9. Instead of using four preamplifiers, with two sensing the inputs $x(t)$ and $z(t)$ and two providing different offsets, we use only two preamplifiers.

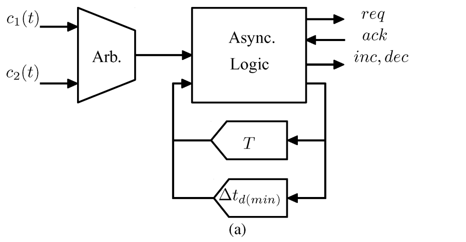
The preamplifier PA_A outputs a differential current ΔI_{xz} as the result of the comparison between $x(t)$ and $z(t)$. Thus, the capacitive loads of these nodes are reduced by using only one preamplifier. The other preamplifier (PA_B) provides a differential current $\Delta I_{\text{off}}/2$ corresponding to the input voltage $\Delta e_{th}/2$.

The input to the decision circuit DC_1 is $\Delta I_{xz} + \Delta I_{\text{off}}/2$ and to DC_2 is $\Delta I_{xz} - \Delta I_{\text{off}}/2$. The decision circuits speed up the comparison result. Finally, output buffers generate the digital outputs.

Fig. 10 presents the schematics of the preamplifier, the decision and the output buffer schematic circuits implemented on chip. The preamplifier is a transconductance amplifier with two identical differential output currents at nodes (A_1, B_1) and (A_2, B_2). The decision circuit is a positive feedback circuit and the output buffer is a self-biased amplifier [28].

B. Spike Generator

The spike generator block can provide either a positive or a negative spike according to the output state of the comparator. When the error $e > \Delta e_{th}/2$, a negative spike is transmitted. Similarly, a positive spike is generated when $e < -\Delta e_{th}/2$. Otherwise, no spikes are transmitted.



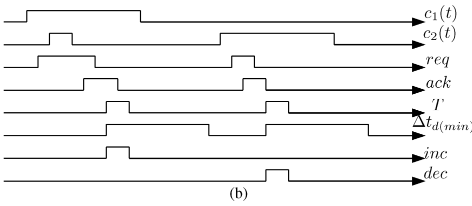


Fig. 11. (a) Block diagram and (b) an example timing diagram of the spike generator. An arbiter selects between $c_1(t)$ and $c_2(t)$ inputs and starts the handshaking signaling. After the handshaking protocol is completed, either a inc or dec pulse is generated using delay blocks for t and $\Delta t_{d(\min)}$.

The spike generator block also includes programmable delay circuits for the generation of T and $\Delta t_{d(\min)}$ time intervals. Fig. 11(a) is the block diagram of the spike generator and Fig. 11(b) shows an example of the signals behavior.

The control circuitry for this logic was implemented using a technique for the design of asynchronous digital circuits [29]. These spikes are transmitted across the channel using AER.

C. Integrators

We implemented the integrator block using a charge pump integrator as shown in Fig. 12. A unipolar version of this type of circuit driving resistors is used in steering current cells of some digital-to-analog converters (DACs) [30].

In this implementation, a spike with width T is used in the integrator block to increment or decrement $z(t)$ by δ. According to (6) the coder output spike frequency is a function of the magnitude of the input derivative and the coder integration step.

For an optimum performance, the tracking step δ is equal to the difference between the thresholds Δe_{th} of the comparators. However, a design margin is required because of the random offsets present in the comparator due to process variations. For instance, setting $\delta < \Delta e_{th}$, more output spikes will be needed for the feedback signal $z(t)$ to track the input signal $x(t)$. Conversely, $z(t)$ will oscillate for $\delta > \Delta e_{th}$. Therefore, the *designed* tracking step (δ_d) is the same as given by (8).

The integrator circuit in Fig. 12 works as follows. When a negative spike arrives at the integrator input, the dec signal goes high for an interval T, allowing currents $1.5I$ and $0.5I$ to flow in transistors M_{22} and M_{23}, respectively. Similarly, for the case of

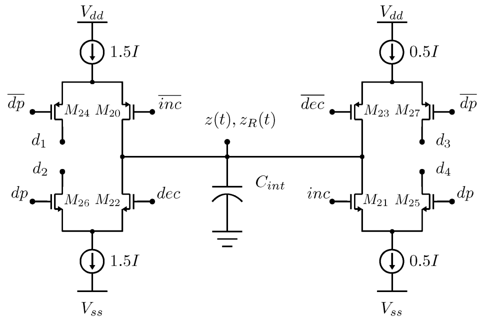


Fig. 12. Schematic of the integrator based on a charge pump technique. When a positive spike arrives, M_{20} and M_{21} turn on and $1.5I$ and $0.5I$ charges and discharges the capacitor C_{int}, respectively, resulting in a voltage increment of δ_d given by (20). For negative spikes, the complementary process decreases the capacitor voltage. In the absence of spikes, currents are drawn to low impedance nodes $(d_1 - d_4)$.

a positive spike arrival, transistors M_{20} and M_{21} provide symmetrical operation with inc and $\overline{inc}$ signals. The resulting current I that discharges or charges the integrating capacitor C_{int} is given by

$$I = C_{int} \times \delta_d / T. \tag{20}$$

From (9), the designed integrator gain may be obtained. When there are no spikes, currents are driven to low impedance nodes $(d_1 - d_4)$ through $M_{24} - M_{27}$ by setting the signal dp high.

The use of two different branches (M_{22} and M_{23} or M_{20} and M_{21}) to both charge and discharge the capacitor reduces charge injection on the integration nodes $z(t)$ and $z_R(t)$ [31] at the cost of doubling the power consumption required. If switches M_{20} and M_{21} (M_{22} and M_{23}) have the same dimensions, the charges injected from the gate to drain capacitance of the complimentary switches cancel each other.

According to (20), the tracking step δ_d is a function of the bias current I. Therefore, the gain operation described in Section IV-A is implemented setting different bias currents I for coder and decoder integrators.

D. AER Channel

The AER protocol employs asynchronous transmission of digital words using a multiple access channel common to every transmitter and receptor in the array. The information transmitted has a unique identification (address) of either the coder or the decoder, depending on the implementation.

An AER router is responsible for distributing these events to the appropriate receivers using internal or external LUTs, for instance. In this paper, we use an external FPGA to route the spikes between coders and decoders. Because each coder transmits two types of spikes (positives and negatives), two different addresses were used in this work for each coder.

Spike collisions during access to the channel are possible and an arbiter is used to resolve them. In our implementation, spike collisions are resolved by a 1-persistent arbiter by queueing and transmitting successively all the spike events involved in the collision [19]. The effect of such collisions is an error in signal

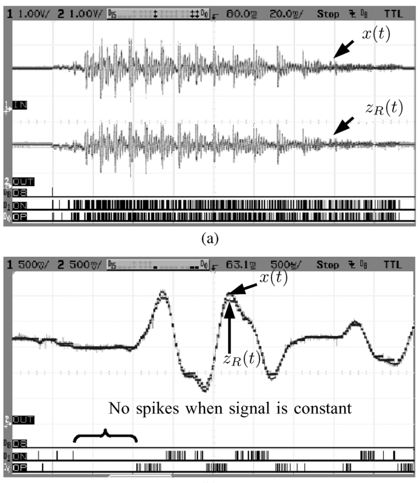


Fig. 13. (a) Coder input $x(t)$ and decoder integrator output $z_R(t)$ for a $2.5V_{pp}$ speech signal. Negative (ON) and positive (OP) output coder spikes are also shown at the bottom of the figure. (b) A detailed view of the same waveforms to show the absence of spikes during the periods when the input signal is constant.

$z_R(t)$ during the time the spikes are being recovered from the queue. Although the decoded value after the conflict resolution is correct, this temporary error causes distortion on the decoded waveform [19].

VI. Chip Results

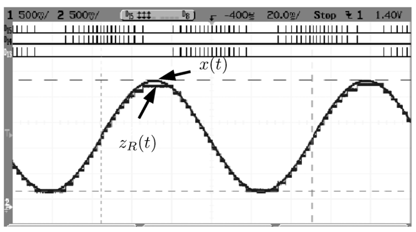
In this section, we present chip results to demonstrate the communication aspects of the event coding scheme. Measured chip results for the computation capabilities of the communication scheme were presented in Section IV and are shown in Figs. 5–7. Two different input signals were used to test the communication system: a speech signal and a sine wave.

A. Response to a Speech Signal

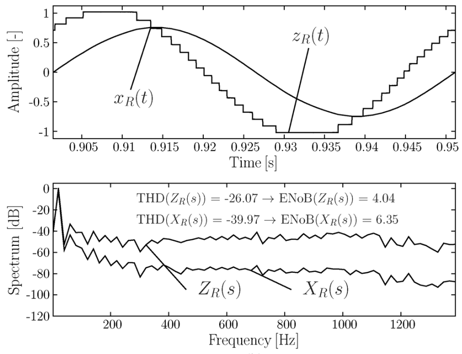
A speech signal sampled at 44.1 kSps with 8 bit resolution was used and the coder was designed to provide a resolution of 6 bits. The coder input signal $x(t)$, the decoder integrator output $z_R(t)$ and the coder output spikes $y(t)$ are shown in Fig. 13(a). The same signals are presented in detail in Fig. 13(b). This figure demonstrates the asynchronous nature of the communication scheme and the absence of coder output spikes when the signal is constant or when its change is smaller than Δe_{th}.

B. Response to a Sine Wave

The resolution of the system was measured using a sine wave input signal with an amplitude of 2.0 V_{pp} and frequency of 20 Hz (f_{in}). The sine wave was sampled at 44.1 kSps and the coder was designed to provide a 4-bit resolution. A snapshot of the input and output signals is presented in Fig. 14(a). Offline filtering results are shown in the Fig. 14(b) for a digital LPF with a cutoff frequency of 20 Hz, the same frequency as the input signal.



(a)



(b)

Fig. 14. (a) Snapshot of $2.0V_{pp}$ sine wave input $x(t)$ and decoder integrator output $z_R(t)$ and negative (ON) and positive (OP) spikes from the coder output (bottom). (b) $z_R(t)$ and the low pass filter output $x_R(t)$ (top). The low pass filter cutoff frequency is the same as the input signal. The frequency spectrum $Z_R(s)$ and $X_R(s)$ of the filter input and output (bottom).

The total harmonic distortion (THD) and resolution of the system were measured using this sine wave input signal. The measured THD of the pre-filter signal is -26 dB which corresponds to a measured resolution of 4.04 bits. The measured THD of the post-filter output, with a filter cutoff frequency equal to f_{in}, improves to -40 dB which corresponds to a measured resolution of 6.35 bits. This resolution improvement causes attenuation and phase shift of the signal. When the cutoff frequency is increased to 10 f_{in}, the measured resolution is equal to 4.93 bits. The low pass filter cut-off of f_{in} yields better resolution attributable to greater attenuation of harmonics due to the lower frequency pole of the low pass filter. The maximum resolution is mainly defined by mismatch and noise in the circuit implementation and may be improved in further work.

VII. Discussion and Conclusion

In this paper, we have presented a spike event coding scheme for communicating analog signals between CABs in a programmable analog array. The design methodology and circuit implementation of the communication scheme were presented together with results from fabricated chips.

In our method, spike events are transmitted asynchronously and power dissipation is dependent upon signal activity. No spike events are generated when the input signal is constant. This is in direct contrast to other pulse-based programmable analog arrays, [9] for example.

We have shown that the proposed communication scheme is capable of asynchronous transmission of analog signal information between CABs using the AER protocol. The use of spike event coding and the AER protocol allows time-sharing of the same physical channel between multiple CABs thereby avoiding an exponential increase in the number of analog connections—a limiting factor in the realization of large programmable analog arrays. Our system has an interconnect complexity of $\mathrm{O}(\sqrt[2]{N})$, whereas analog interconnect using crossbars and switch matrices have a complexity of $\mathrm{O}(N^2)$ [32], [33]. The circuit area consumed by the implementation of crossbars and switch matrices increases as $\mathrm{O}(N^2)$, while our architecture has a proportional increase in circuit area, increasing as $\mathrm{O}(N)$. In other words, this architecture is more suitable to large arrays.

Spike events are essentially asynchronous and robust digital signals that are easy to route on shared channels, not only between CABs, but also between chips providing improved scalability. When extended to interchip communications the interconnect complexity is $\mathrm{O}(\log_2 N)$.

We have demonstrated the intrinsic computation capability of the spike event coding scheme. This provides basic arithmetic operations essentially in the communication channel, without the need for additional CABs thereby enhancing the computing capability of a programmable analog array.

Acknowledgment

The authors would like to thank S. Bamford for the design of the AER circuitry. The authors would also like to acknowledge support from the Scottish Funding Council for the Joint Research Institute for Integrated Systems which is part of the Edinburgh Research Partnership in Engineering and Mathematics (ERPem).

References

[1] D. L. Grundy, “A computational approach to VLSI analog design,” *J. VLSI Signal Process.*, vol. 26, no. 12, pp. 1860–1867, Dec. 1991.

[2] A. Bratt and I. MacBeth, “Dpad2A field programmable analog array,” *Analog Integrated Circuits Signal Process.*, vol. 17, no. 1/2, pp. 67–89, Sep. 1998.

[3] E. K. Lee and P. G. Gulak, “A CMOS field-programmable analog array,” *IEEE J. Solid-State Circuits*, vol. 26, no. 12, pp. 1860–1867, Dec. 1991.

[4] D. R. D’Mello and P. G. Gulak, “Design approaches to field-programmable analog integrated circuits,” *Analog Integrated Circuits Signal Process.*, vol. 17, no. 1–2, pp. 7–34, Sep. 1998.

[5] F. Baskaya, D. V. Anderson, and S. K. Lim, “Net-sensitivity-based optimization of large-scale field-programmable analog array (FPAA) placement and routing,” *IEEE Trans. Circuits and Syst. II, Exp. Briefs*, vol. 56, no. 7, pp. 565–569, Jul. 2009.

[6] D. Varghese and J. N. Ross, “A Continuous time hierarchical field programmable analogue array for rapid prototyping and hierarchical approach to analogue systems design,” in *Proc. 18th Annu. ACM Symp. Integr. Circuits Syst. Design*, Florianopolis, Brazil, 2005, pp. 248–253.

[7] T. S. Hall, C. M. Twigg, P. Hasler, and D. V. Anderson, “Developing large-scale field-programmable analog arrays,” in *Proc. 18th Int. Parallel Distrib. Process. Symp.*, Santa Fe, NM, 2004, pp. 142–147.

[8] J. Becker and Y. Manoli, “A continuous-time field programmable analog array (FPAA) consisting of digitally reconfigurable G_M-cells,” in *Proc. IEEE 10th Int. Symp. Asynchronous Circuits Syst.*, Crete, Greece, 2004, pp. 1092–1095.

[9] K. Papathanasiou, T. Brandtner, and A. Hamilton, "Palmo: Pulse-based signal processing for programmable analog VLSI," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 49, no. 6, pp. 379–389, Jun. 2002.

[10] D. Kościelnik and M. Miśkowicz, "Asynchronous Sigma-Delta analog-to digital converter based on the charge pump integrator," *Analog Integr. Circuits Signal Process.*, vol. 55, no. 3, pp. 925–1030, Jun. 2008.

[11] Y. W. Li, K. L. Shepard, and Y. P. Tsividis, "Continuous-time digital signal processors," in *Proc. IEEE 11th Int. Symp. Asynchronous Circuits Syst.*, New York, 2005, pp. 138–143.

[12] Y. P. Tsividis, "Event-driven, continuous-time ADCs and DSPs for adapting power dissipation to signal activity," in *Proc. IEEE Int. Symp. Circuits Syst.*, Paris, France, 2010, pp. 3581–3584.

[13] Y. P. Tsividis, "Mixed-domain systems and signal processing based on input decomposition," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 53, no. 10, pp. 2145–2156, Oct. 2006.

[14] W. Gerstner, "Spiking neurons," in *Pulsed Neural Networks*, W. Maass and C. Bishop, Eds. Cambridge, MA: MIT Press, 1998, pp. 4–53.

[15] E. Allier, G. Sicard, L. Fesquet, and M. Renaudin, "A new class of asynchronous A/D converters based on time quantization," in *Proc. IEEE 9th Int. Symp. Asynchronous Circuits Syst.*, Vancouver, BC, Canada, 2003, pp. 196–205.

[16] A. A. Lazar and L. T. Toth, "Perfect recovery and sensitivity analysis of time encoded bandlimited signals," *IEEE Trans. Circuits Syst. I, Fundam. Theory Appl.*, vol. 51, no. 10, pp. 2060–2073, Oct. 2004.

[17] D. Wei and J. G. Harris, "Signal reconstruction from spiking neuron models," in *Proc. IEEE Int. Symp. Circuits Syst.*, Vancouver, BC, Canada, 2004, pp. 353–356.

[18] K. Boahen, "Point-to-point connectivity between neuromorphic chips using address events," *IEEE Trans. Circuits Syst. II, Analog Digital Signal Process.*, vol. 47, no. 5, pp. 416–434, May 2000.

[19] L. Gouveia, T. Koickal, and A. Hamilton, "An asynchronous spike event coding scheme for programmable analog arrays," in *Proc. IEEE Int. Symp. Circuits Syst.*, Seattle, WA, 2008, pp. 1364–1367.

[20] H. Inose, T. Aoki, and K. Watanabe, "Asynchronous delta-modulation system," *Electron. Lett.*, vol. 2, no. 3, pp. 95–96, Mar. 1966.

[21] J. B. Shyu, G. C. Temes, and F. Krummenacher, "Random error effects in matched MOS capacitors and current sources," *IEEE J. Solid-State Circuits*, vol. 19, no. 6, pp. 948–955, Jun. 1984.

[22] B. D. Brown and H. C. Card, "Stochastic neural computation I: Computational elements," *IEEE Trans. Comput.*, vol. 50, no. 9, pp. 891–905, Sep. 2001.

[23] T. Koickal, L. Gouveia, and A. Hamilton, "A programmable spike-timing based circuit block for reconfigurable neuromorphic computing," *Neurocomputing*, vol. 72, no. 16–18, pp. 3609–3616, Oct. 2009.

[24] L. Gouveia, T. Koickal, and A. Hamilton, "A CMOS implementation of a spike event coding scheme for analog arrays," in *Proc. IEEE Int. Symp. Circuits Syst.*, Taipei, Taiwan, 2009, pp. 149–152.

[25] R. J. Baker, H. W. Li, and D. E. Boyce*, CMOS: Circuit Design, Layout and Simulation*. New York: IEEE Press, 1998.

[26] A. A. Fayed and M. Ismail, "A high speed, low voltage CMOS offset comparator," *Analog Integr. Circuits Signal Process.*, vol. 36, no. 3, pp. 267–272, Sep. 2003.

[27] D. Yaklin, "Offset Comparator With Common Mode Voltage Stability," U.S. Patent 5,517,134, May 14, 1996.

[28] M. Bazes, "Two novel fully complementary self-biased CMOS differential amplifiers," *IEEE J. Soild-State Circuits*, vol. 26, no. 2, pp. 165–168, Feb. 1991.

[29] A. Martin, "Translating concurrent programs into VLSI chips," in *Proc. 4th Int. PARLE Conf. Parallel Architectures Lang. Eur.*, Paris, France, 1992, pp. 515–532.

[30] J. Wikner and N. Tan, "Modeling of CMOS digital-to-analog converters for telecommunication," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 46, no. 5, pp. 489–499, May 1999.

[31] H. Pan, "Method and System for a Glitch-Free Differential Current Steering Switch Circuit for High Speed, High Resolution Digital-to-Analog Conversion," U.S. Patent 7,071,858, May 4, 2006.

[32] S. Deiss, R. Douglas, and A. Whatley, "A pulse-coded communications infrastructure for neuromorphic systems," in *Pulsed Neural Networks*, W. Maass and C. Bishop, Eds. Cambridge, MA: MIT Press, 1998, pp. 157–178.

[33] V. Beiu and W. Ibrahim, "Does the brain really outperform Rent's rule?," in *Proc. IEEE Int. Symp. Circuits Syst.*, Seattle, WA, 2008, pp. 640–643.

Luiz Carlos Gouveia received the B.Sc. and M.Sc. degrees in electrical engineering from Federal University of Rio de Janeiro, Rio de Janeiro, Brazil, in 1995 and 2001, respectively. He is currently working towards the Doctoral degree in electrical engineering at the University of Edinburgh, Edinburgh, U.K.

He was a VLSI Engineer at COPPETEC, Brazil, from 1996 to 1997, a Network Engineer at Telefonica Celular, Brazil, from 1997 to 1999, and an Analog VLSI Engineer at ChipIdea S.A., Portugal, from 2002 to 2006. He holds a Research Assistant position at the University of Glasgow, Glasgow, U.K. His current interests include low-power analog circuits, programmable analog VLSI, image sensor systems, and bioinspired systems.

Thomas Jacob Koickal received the Ph.D. degree in electronics and electrical communication engineering from the Indian Institute of Technology, Kharagpur, India.

From 1998 to 2002, he was a Scientist at the Control and Guidance Design Group, Indian Space Research Organization, Trivandrum. Currently, he is a Researcher at the Institute of Micro and Nano Systems, University of Edinburgh, Edinburgh, U.K. His research interests include bioinspired modeling, time-based computation, neuromorphic systems, MEMS/CMOS multi-sensory systems, and programmable analog VLSI.

Alister Hamilton is a Senior Lecturer in the discipline of Electronics in the School of Engineering, University of Edinburgh, Edinburgh U.K., where he has worked since 1988. His research interests are in the implementation of neural networks and neuromorphic systems in analog VLSI and in novel design strategies for programmable analog arrays.

Computation in Communication: Spike Event Coding for Programmable Analog Arrays

Luiz Carlos Gouveia, Thomas Jacob Koickal and Alister Hamilton
Institute for Micro and Nano Systems - School of Engineering and Joint Research Institute for Integrated Systems
University of Edinburgh, EH9 3JL, U.K.
Email: L.Gouveia@ed.ac.uk, {Thomas.Koickal, Alister.Hamilton}@ee.ed.ac.uk

***Abstract*— This paper presents the computation properties of an asynchronous spike event coding scheme employed for communicating signals between analog blocks in a programmable array. The computation is intrinsic to the spike event communication scheme and is performed without additional hardware. The ability of the communication scheme to perform computation will enhance the computation power of the programmable analog array. Test results from a chip implemented using $0.35\mu m$ CMOS technology are presented.**

I. Introduction

A field programmable analog array (FPAA) consists of a number of configurable analog blocks (CABs) that are programmed to perform a set of computations. This set of computations is defined by the target application of the system. For instance, a neuromorphic application requires integration (for an integrate-and-fire neuron) [1] and summation (for the summation of presysnaptic spikes into the soma) [2], among others. Likewise, an array designed to perform more generic analog signal processing requires amplification, filtering and summation [3]. In general, these systems perform computations by programming the configurable analog blocks in the array, while the role of the communication interconnect is to route the signals between analog blocks. The computation power of a programmable array can be enhanced if the communication channel can perform computation without additional overheads.

The authors proposed a novel analog array architecture in [4]. This architecture uses an asynchronous spike event coding scheme [5] to communicate analog signals between CABs. This is distinct from conventional communication methods using voltage or current with crossbars or matrix switches [6] to represent signal amplitudes. It is also distinct from our previous work where analog signals were encoded using synchronous pulse width modulated signals [7]. The use of asynchronous spike event coding leads to the ability of our system to perform computation in the communication channel.

In this paper, we show the computation properties of the spike event communication scheme. The communication scheme can perform a set of fundamental arithmetic operations for example gain, inversion and summation, without using extra hardware. These fundamental operations are performed simply by programming parameters of the communication channel. From these basic operations, more complex operations can be implemented either by programming the channel and/or by programming the CABs. The computation capability of this communication scheme can enhance the computing power of the programmable array without using additional hardware. The communication scheme is implemented in $0.35\mu m$ CMOS technology and chip measurement results showing the computation properties are presented. In the following section, communication system architecture is described.

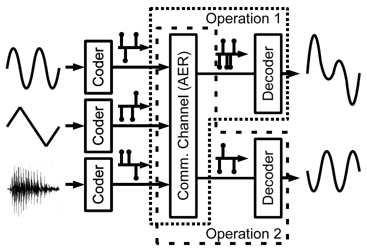
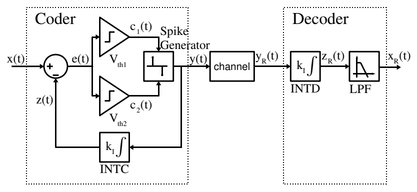


Fig. 1. Computing over the channel. Analog signals are converted into spikes by the coders. Available computations are performed using the decoders and the shared communication channel. The type of computations performed is defined by programming the decoders and channel parameters.

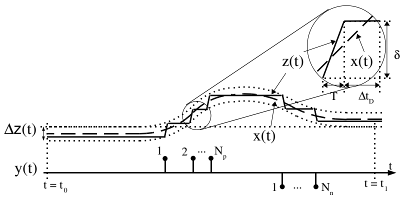
II. Architecture Description

In the proposed architecture, CABs are interconnected using an *asynchronous spike event communication scheme* [5]. The communication scheme consists of a pair of spike event coder and decoder for each CAB and the channel as shown in Fig.1. The spike event coder encodes the analog signals into asynchronous discrete amplitude signals (spike events) which are routed to target CABs using the Asynchronous Event Representation (AER) protocol. The decoder at the target CABs decodes the spike event signals received from the AER bus.

A set of fundamental arithmetic computations can be performed by configuring the AER router and the decoder parameters as explained in Section III. For instance, in Fig. 1, the top decoder performs a summation of two inputs and the second one provides a negation of one of the inputs using the same channel. In this section, we give a brief description of the spike event coding scheme and the AER protocol.



(a)



(b)

Fig. 2. (a) Event coding block diagram and (b) an example showing waveforms of predict behavior for important signals. In the inset, δ is the tracking step, T is the spike "width" and Δt_D is the time interval between successive spikes.

A. Spike Event Coding Scheme

The spike event coding scheme is a communication method based on the principle of irregular sampling schemes described in [8] and [9]. The scheme provides an efficient utilization of resources and lower power consumption. Further, the events are transmitted digitally providing improved scalability in building large arrays.

The block diagram of the spike event coding scheme is shown in Fig. 2(a). The event coder tracks the input signal by bounding an error signal given by:

$$e(t) = x(t) - z(t) = x(t) - \int y(t)dt \tag{1}$$

where $e(t)$ is the error between the analog input signal $x(t)$ and coder feedback integrator (INTC) output $z(t)$. The coder output spikes $y(t)$ are sent to the communication channel and to the input of the feedback integrator. The decoder output $x_R(t)$ is an analog signal given by:

$$x_R(t) = LPF(z_R(t)) \approx \overline{z_R(t)} = \overline{\int y_R(t)dt} \tag{2}$$

If the channel is ideal, $y_R(t) = y(t)$ then

$$x_R(t) \approx \overline{\int y(t)dt} = \overline{z(t)} = \overline{x(t) - e(t)} \tag{3}$$

The error between the decoder output $x_R(t)$ and coder input $x(t)$ is bounded, $|e|_{max} \leq \delta$, where δ, the tracking step or quantization error, is a system parameter.

The outputs of the event coder are represented by positive and negative fixed short duration pulses (spikes). These spikes are generated by the spike generator when the comparators change their states. Each positive or negative spike generates an incremental or decremental change (δ) at the output of the both integrators ($INTC$ and $INTD$).

The amplitude change in the integrator is given by:

$$\Delta z(t) = \delta\ (N_p - N_n) \tag{4}$$

where N_p (N_n) is the number of previous positive (negative) spikes since t_0 as shown in Fig. 2(b). In other words, knowing the initial condition of coder and decoder, the signal value at any instant t is given by the number of events of each type transmitted. Further details about design of the spike event coding scheme may be found in [5] and [11].

B. Communication Channel Protocol: AER

The signal to be transmitted between CABs is a series of asynchronous spike time events generated by the coder block. The transmission of these events, represented by discrete amplitude signals, may be implemented using a digital bus instead of analog interconnections. By time-sharing the same physical channel an exponential increase in the number of analog connections can be avoided - a limiting factor in the realization of large programmable analog arrays. As these events are asynchronous, the AER protocol [10], widely used in neuromophic designs, is an appropriate choice because it avoids the synchronization of events and therefore preserving the information conveyed in the time difference between events.

The AER protocol employs asynchronous transmission of digital words using a multiple access channel (MAC) common to every transmitter and receptor in the array. The information transmitted has a unique identification (address) of either the coder or the decoder, depending on the implementation. An AER router is responsible for distribute these events to the appropriate receivers using internal or external LUTs, for instance.

For this work, different addresses for the positive and negative spikes for each coder are transmitted.

III. Arithmetic Operations

In this section we show a set of computations performed by the communication scheme. For each operation, we show the measured chip results. From these basic set of operations more complex operations can be achieved. Details of the circuits used to implement the coder and decoder together with chip results were presented in [11].

Another chip has been designed containing a larger array containing 4 coders, 4 decoders and an AER router. Results from this chip are reported here for the first time. The following results were obtained for a 4-bit resolution design. The outputs waveforms shown correspond to the decoder filter input $z_R(t)$.

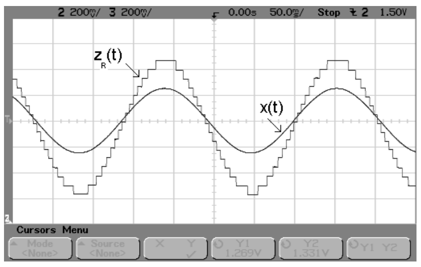


Fig. 3. Gain operation. The input sinusoid signal $x(t)$ is amplified by 2 at the decoder $z_R(t)$. The gain operation is obtained by changing the ratio of the tracking steps of coder and decoder.

A. Gain

A common operation in any analog processing is to change the intensity of a signal, i.e., provide a gain G to the signal:

$$x_R(t) = G \times x(t) \tag{5}$$

The gain may increase (amplification) or decrease (attenuation) the amplitude of the signal. Both gain types are possible using the proposed architecture.

Eq. 4 is valid for both coder and decoder blocks. However, if coder and decoder are designed to present different tracking steps δ then the decoder output will be proportional to the coder input. This gain is given by the ratio of both tracking steps amplitudes:

$$G = \frac{\Delta z_R(t)}{\Delta z(t)} = \frac{\delta_D}{\delta_C} \tag{6}$$

Fig. 3 shows the chip result for an amplification of a sinusoid by a factor of 2.

B. Negation

Another fundamental operation on analog signals is to change their polarity, i.e., signal negation:

$$x_R(t) = -x(t) \tag{7}$$

According to Eq. 4, the signal value at any instant t (knowing the initial condition of coder and decoder) is a function of the number of events transmitted and their type. By setting the AER router to interchange the addresses of positive and negative spikes, the signal at the decoder output $x_R(t)$ is a negated version of the input signal $x(t)$:

$$\Delta z_R(t) = -\Delta z(t) = \delta \left(N_n - N_p\right) \tag{8}$$

An example using a sinusoid signal is shown in Fig. 4.

C. Summation

Beyond these unary operations, arithmetic operations involving two or more signals are also required by a generic

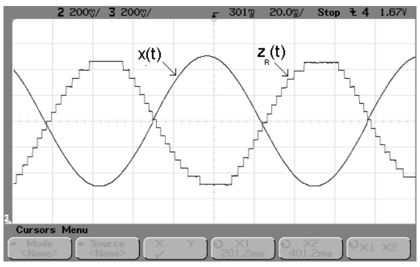


Fig. 4. Negation operation. Snapshot of the input sinusoid $x(t)$ and the negated signal $z_R(t)$. The negation is perform by interchange the positive and negative spikes using the AER router.

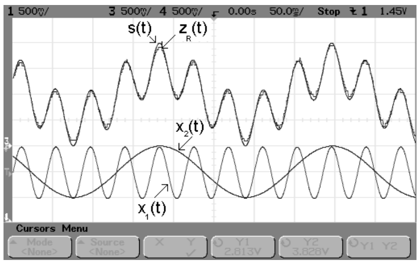


Fig. 5. Summation operation of two input sinusoids, $x_1(t)$ and $x_2(t)$, with same amplitude but different frequencies (23Hz and 4.7Hz) and the respective summation $z_R(t)$ with $s(t)$ being the ideal summation.

system. The fundamental arithmetic operation involving multiple signals is the summation of these signals:

$$s(t) = \sum_{i=1}^{j} x_i(t) \tag{9}$$

Again considering Eq. 4, the summation signal $s(t)$ of j signals is given by:

$$s(t) = \delta \left(\sum_{i=1}^{j} N_{pi} - \sum_{i=1}^{j} N_{ni} \right) + s(0) \tag{10}$$

where N_{pi} and N_{ni} are the number of positive and negative spikes, respectively, received from the ith operand and $s(0)$ is the initial state of the decoder.

Chip results showing the summation of two sinusoid signals $x_1(t)$ and $x_2(t)$ are shown in Fig. 5, together with the decoder integrator output $z_R(t)$. The signal $s(t)$ represents the ideal summation result.

Subtraction operation is performed by combining the summation and negation operations outlined above.

D. Examples of applications

These basic operations allow to configure an array of programmable analog blocks to compute more complex tasks.

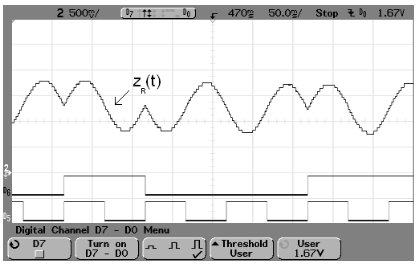


Fig. 6. Binary phase shift keying. This modulation is implemented by routing the spikes according to the digital input (D6). D5 frequency is the same of the sinusoid carrier.

A first example is phase shift keying (PSK). Phase shift keying is a digital modulation where the information is coded in the phase of the carrier. For instance, *binary* phase shift keying (BPSK) is defined as follows:

$$s(t) = \begin{cases} a\,sin(\omega_c t) & if \quad b_i = 0 \\ a\,sin(\omega_c t + 180°) & if \quad b_i = 1 \end{cases} \tag{11}$$

where b_i is the bit to be transmitted, w_c is the angular frequency of the carrier and a is a function of the energy-per-symbol and the bit duration [12].

This function can be implemented by providing the carrier (sinusoid) signal to the input of the coder and routing the spikes according to the digital value to be transmitted. For instance, while the input bit is zero, the carrier is replicated at the decoder output, but when the input bit is one, a negation function is performed on the carrier. Fig. 6 shows the modulation result $z_R(t)$ for a input word of 010011.

Quadrature phase shift keying (QPSK) is implemented summing two carriers with 90° phase shift.

Another application example of our scheme is to implement the weighted sum operation commonly used in artificial neural networks. A weighted summation is a combination of summation and gain operations:

$$f(t) = \sum_i w_i x_i(t) \tag{12}$$

All of the operations above are performed using asynchronous spike event coded signals and by AER router and decoder programming. However, the range of possible computations is expanded when the communication scheme is combined with the functionality of the CABs [4]. For instance, if CABs were designed to perform logarithmic compression and exponential expansion, both multiplication and division can be implemented using the summation operation according to the logarithmic property:

$$x_R(t) = \prod_i x_i(t) = e^{\sum_i log\, x_i(t)} \tag{13}$$

IV. Conclusions

In this paper we presented the principles and results from a CMOS implementation of a communication strategy intended for use in programmable analog arrays that is also capable of performing fundamental arithmetic operations (gain, negation and summation). The architecture uses an asynchronous spike event coding scheme to communicate between analog blocks. This scheme allows the user to select which operations to be performed by simply programming the channel router and decoder parameters, either before or even during the process.

No extra dedicated hardware is needed to perform these operations and any decoder in the array is able to perform any of the operations. More complex and specialized functions are available by combining these fundamental operations, like PSK and weighted summation. Other applications, like multiplication, are possible when combined with the functions performed by the analog blocks.

Acknowledgment

The authors would like to thank the supports from Engineering and Physical Sciences Research Council and the Scottish Funding Council for the Joint Research Institute with the Heriot-Watt University, part of the Edinburgh Research Partnership in Engineering and Mathematics (ERPem) and to Simeon Bradford for the design of the AER arbiter.

References

[1] G. Indiveri, E. Chicca and R. Douglas, "A VLSI array of low-power spiking neurons and bistable synapses with spike-timing dependent plasticity," *IEEE Trans. on Neural Networks*, vol. 17, no. 1, pp. 211-221, Jan. 2006.
[2] R. Z. Shi and T. Horiuchi, "A summating, exponentially-decaying cmos synapse for spiking neural systems," *Adv. in Neural Info. Proc. Syst.*, vol. 16, 2004.
[3] Lee, E. and Hui, W., "A Novel Switched-Capacitor Based Field-Programmable Analog Array Architecture," *Analog Integr. Circuits Signal Processing*, vol. 17, no.1-2, pp. 35-50, 1998.
[4] T. Koickal, L. Gouveia and A. Hamilton, "A programmable spike-timing based circuit block for reconfigurable neuromorphic computing," *Neurocomputing*, vol. 72, no. 16-18, pp. 3609-3616, 2009.
[5] L. Gouveia, T. Koickal and A. Hamilton, "An asynchronous spike event coding scheme for programmable analog arrays," in *Proc. of the IEEE Int. Symp. on Circuits and Systems*, pp.1364-1367, 2008.
[6] D. D'Mello and P. Gulak, "Design approaches to field-programmable analog integrated circuits," in *Special Issue on Programmable Analog Systems, Analog Integrated Circuits and Signal Processing*, vol. 17, no. 1-2, pp. 7-34, Sep. 1998.
[7] K. Papathanasiou, T. Brandtner and A. Hamilton, "Palmo: pulse-based signal processing for programmable analog VLSI," in *IEEE Trans. on Circuits and Systems-II, Analog and Digital Signal Processing*, vol. 49, no. 6, pp. 379-389, Jun. 2002.
[8] Y. Li, K. Shepard and Y. Tsividis, "Continuous-time digital signal processors," in *Proc. of IEEE Int. Symp. on Asynchronous Circuits and Systems*, pp. 138-143, Mar. 2005.
[9] E. Allier, G. Sicard, L. Fesquet and M. Renaudin, "A new class of asynchronous A/D converters based on time quantization," in *Proc. of Int. Symp. on Async. Circuits and Systems*, pp. 196-205, May 2003.
[10] K. Boahen, "Point-to-point connectivity between neuromorphic chips using address events," in *IEEE Trans. on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 47, no. 5, pp. 416-434, May 2000.
[11] L. Gouveia, T. Koickal and A. Hamilton, "A CMOS Implementation of a Spike Event Coding Scheme for Analog Arrays," in *Proc. of the IEEE Int. Symp. on Circuits and Systems*, pp. 149-152, 2009.
[12] S. G. Wilson, *Digital modulation and coding*, Upper Saddle River, NJ: Prentice-Hall Inc., 1996.

A CMOS Implementation of a Spike Event Coding Scheme for Analog Arrays

Luiz Carlos Gouveia, Thomas Jacob Koickal and Alister Hamilton
Institute for Micro and Nano Systems - School of Engineering and Joint Research Institute for Integrated Systems
University of Edinburgh, EH9 3JL, U.K.
Email: L.Gouveia@ed.ac.uk, {Thomas.Koickal, Alister.Hamilton}@ee.ed.ac.uk

***Abstract*— This paper presents a CMOS circuit implementation of a spike event coding/decoding scheme for transmission of analog signals in a programmable analog array. This scheme uses spikes for a time representation of analog signals. No spikes are transmitted using this scheme when signals are constant, leading to low power dissipation and traffic reduction in a shared channel. A proof-of-concept chip was designed in a 0.35μm process and experimental results are presented.**

I. Introduction

Configurable Analog Blocks (CABs) are the basic processing units of Field Programmable Analog Arrays (FPAAs) and are configured to perform different types of analog functions [1]. The communication between CABs is an important issue because traditional techniques like crossbars or switch matrices [2] degrade the transmitted signals. These techniques cause distortion of the signal due to voltage drops, parasitic capacitances along wires and the switches and through signal interference, limiting the size of analog arrays.

Pulse modulations are an alternative to these techniques. They map the amplitude of analog signals onto the timing domain. The use of these discrete-amplitude continuous-time modulations allows greater system scalability than analog routing methods. Synchronous versions were used in analog arrays [3], but as they require a global clock signal, these modulations suffer from clock skew and high power consumption.

Asynchronous modulations are used in some biological inspired systems [5] [6]. Recently, a *spike event coding scheme* was proposed by the authors to transmit analog signals between CABs [4]. It presents advantages over other pulse modulations, such as transmission of information on demand and, therefore, reduction in communication traffic, and low energy dissipation, freedom from clock skew and low crosstalk due to asynchronous coding.

In this paper we present a CMOS implementation of the spike event coding scheme with programmable resolution. First we review the working principles of the scheme. Later, we describe the design parameters and the circuits used to realize the coder. Finally, we present results from a tested chip.

II. Spike Event Coding Scheme

The spike event coding scheme [4] is shown in Fig. 1(a). The spike event coder operates by generating a signal similar to the input signal. In other words, a feedback signal $z(t)$ is

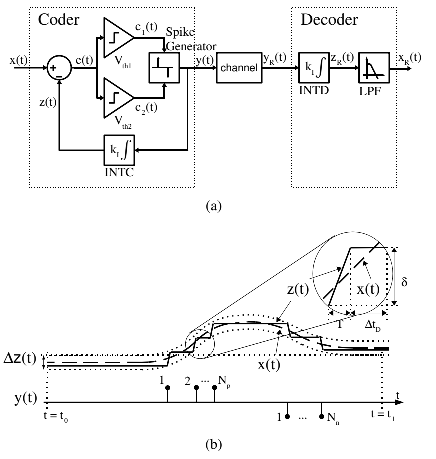


Fig. 1. (a) Block diagram of the spike event coding scheme and (b) an example of the behavior of the main signals. δ is the tracking step, T is the spike width and Δt_D is the time interval between successive spikes.

forced to track the input signal $x(t)$ by bounding the error $e(t)$ between them:

$$e(t) = x(t) - z(t) = x(t) - \int y(t)dt \qquad (1)$$

where $y(t)$ is the coder output. This output is represented either by positive or negative pulses with a short and fixed duration (spikes). These spikes are produced by the spike generator and transmitted both to the communication channel and to the input of the feedback integrator (INTC).

Each positive or negative spike results in an incremental or decremental change δ (tracking step) at the output of the feedback integrator:

$$\Delta z(t) = \delta\,(Np - Nn) \qquad (2)$$

where Np (Nn) is the number of previous positive (negative) spikes since t_0 as shown in Fig. 1(b).

Considering an ideal channel, the spikes are also transmitted to the decoder integrator INTD, which presents the same gain K_I of the coder feedback integrator INTC. The decoder output $x_R(t)$ is given by:

$$x_R(t) \approx \overline{\int y(t)dt} = \overline{z(t)} = \overline{x(t) - e(t)} \quad (3)$$

with the decoder Low Pass Filter (LPF) removing high frequency harmonics and averaging the signal $z_R(t)$. Therefore, the maximum difference between the decoder output $x_R(t)$ and coder input $x(t)$ is $|e(t)|_{max}$, which is defined by the specification of the spike event scheme resolution:

$$|e(t)|_{max} = \frac{\Delta x(t)_{max}}{2^{N_B} - 1} \quad (4)$$

where $\Delta x(t)_{max}$ is the input dynamic range and N_B is the desired resolution in bits.

III. Analog VLSI Implementation

The spike event coding scheme was implemented in a proof-of-concept chip for validation of the scheme. The layout dimensions of the spike event coder are 240μm x 120μm using AMS 0.35μm CMOS process.

In this section we describe the spike event coder and decoder circuits implemented on chip: comparators, spike generator and integrators. Both coder and decoder integrators were implemented using the same design. The decoder LPF was implemented off-chip, using an offline digital filter.

A. Comparators

In the block diagram in Fig. 1(a), the error $e(t)$ is limited by two comparators with different thresholds (V_{th1} and V_{th2}):

$$|e| \leq \Delta V_{th} = |V_{th1} - V_{th2}| \quad (5)$$

From (4) and (5), the resolution of the spike event coding scheme is a function of the thresholds of both comparators.

The design of the comparators can be implemented using a preamplifier (PA) followed by a decision circuit (DC) and an output buffer (OB) [7]. To provide the required ΔV_{th}, capacitive or resistive dividers can be used at the comparator input nodes. However, these dividers compromise the input impedance of the circuit.

Another method to provide ΔV_{th} is to implement offset comparators. Composite transistors can be used to provide the offset [8], however this topology suffers from low dynamic range. A programmable offset can also be generated by another preamplifier which provide a respective ΔI_{off} on the decision circuit input [9]. Both impedance dividers and offset comparators allow continuous resolution values to be used.

In this implementation, both outputs $c_1(t)$ and $c_2(t)$ are generated by a compound comparator in Fig. 2. Instead of using four preamplifiers, with two sensing the inputs $x(t)$ and $z(t)$ and two providing different offsets, we use only two preamplifiers. The preamplifier PAA outputs a differential current ΔI_{xz} as the result of the comparison between $x(t)$ and $z(t)$. Thus, the capacitive loads of these nodes are reduced by using only one preamplifier. The other preamplifier (PAB) provides a differential current $\Delta I_{off}/2$ according to $\Delta V_{th}/2$ voltage on the inputs. The results of adding ($\Delta I_{xz}+\Delta I_{off}/2$) and subtracting ($\Delta I_{xz}-\Delta I_{off}/2$) these currents are forwarded to the decision circuits to speed up the comparison result. Finally, output buffers generate the digital outputs.

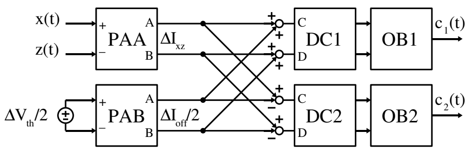


Fig. 2. Block diagram of the compound comparator. Preamplilfier PAA outputs a current ΔI_{xz} according to the inputs $x(t)$ and $z(t)$, while preamplifier PAB generate a fixed offset current $\Delta I_{off}/2$. PAB outputs are added or subtracted from PAA outputs and the results are applied to the respective decision circuits (DC) and output buffers (OB).

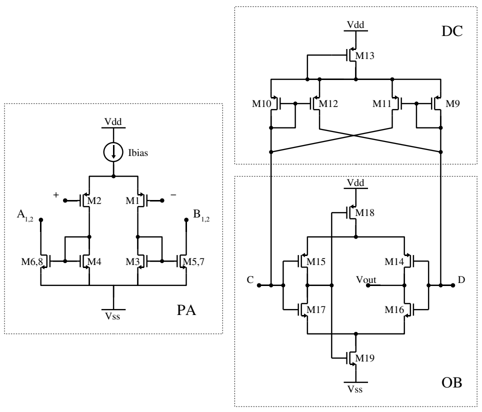


Fig. 3. Circuit schematic of each of the compound comparator blocks. Transconductance preamplifier (PA,left), decision circuit (DC,top right) and output stage (OB,bottom right).

Fig. 3 presents the preamplifier, the decision and the output buffer schematic circuits implemented on chip. The preamplifier is a transconductance amplifier with two identical differential output currents at nodes (A_1, B_1) and (A_2, B_2). The decision circuit is a positive feedback circuit and the output buffer is a self-biased amplifier [10].

B. Spike Generator

The spike generator block can provide either a positive or a negative spike according to the output state of the comparator. When the error $e > \Delta V_{th}/2$, a negative spike is transmitted. Similarly, a positive spike is generated when $e < -\Delta V_{th}/2$. Otherwise, no spikes are transmitted. The control circuitry for this logic was implemented using a technique for the design of asynchronous digital circuits [11]. These spikes can be transmitted using switch matrices or any asynchronous Medium Access Channel (MAC) protocol, like AER [12].

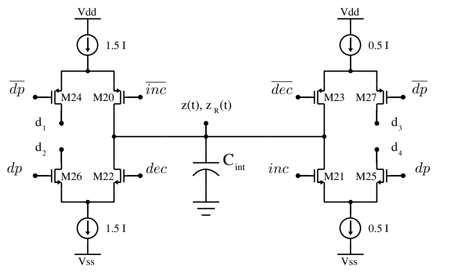


Fig. 4. Schematic of the integrator based on a charge pump technique. When a positive spike arrives, M20 and M21 turn on and $1.5I$ and $0.5I$ charges and discharges the capacitor C_{int}, respectively, resulting in a voltage increment of $\delta_d = T \times I/C_{int}$. For negative spikes, the complementary process decreases the capacitor voltage. In the absence of spikes, currents are drawn to low impedance nodes (d1-d4). The schematic of the delay generators is not shown.

C. Integrators

In this implementation, the spike width T is used in the integrator block to increment or decrement $z(t)$ by δ. A minimum interspike interval $\Delta t_{Dmin} = kT$ is also introduced in the design to avoid overload on the communication channel ("refractory period"). The coder output spike frequency is a function of the magnitude of the input derivative and the coder integration step:

$$f = \frac{\left|\frac{dx(t)}{dt}\right|}{\delta} = \frac{1}{T + \Delta t_D} \quad (6)$$

The first conclusion from (6) is that no spikes are transmitted when the input derivative is zero, i.e. the input signal is constant. This is true only after the feedback signal $z(t)$ has tracked the input signal $x(t)$, i.e. $e(t) \leq \Delta V_{th}$.

We also conclude from (6) that the maximum output frequency occurs when the input signal presents its maximum absolute input derivative. From the specification of this derivative, Δt_{Dmin} is defined and pulse width T is:

$$T = \frac{\delta}{(k+1)\left|\frac{dx(t)}{dt}\right|_{max}} \quad (7)$$

The integrator block includes programmable delay circuits for the generation of T and Δt_{Dmin} time intervals.

For an optimum performance, the tracking step δ is equal to the difference between the thresholds ΔV_{th} of the comparators. Setting $\delta < \Delta V_{th}$, more output spikes will be needed for the feedback signal $z(t)$ track the input signal $x(t)$. Conversely, $z(t)$ will oscillate for $\delta > \Delta V_{th}$. However, a design margin is required because of the random offsets present in the comparator due to process variations. Therefore, the *designed* tracking step is:

$$\delta_d \leq \Delta V_{th} - 6\sigma(V_{os}) \quad (8)$$

where $\sigma(V_{os})$ is the comparator offset standard deviation.

We implemented the integrator block using a charge pump integrator as shown in Fig. 4. A unipolar version of this type of circuit driving resistors is used in steering current cells of some Digital-to-Analog Converters (DACs).

When a negative spike arrives at the integrator input, the dec signal turns high during an interval T, allowing currents $1.5I$ and $0.5I$ flowing in transistors M22 and M23, respectively. Similarly, for the case of a positive spike arrival, transistors M20 and M21 provide symmetrical operation with inc and $\overline{inc}$ signals. The resulting current I that discharges or charges the integrating capacitor C_{int} is given by $I = C_{int} \times \delta_d/T$. Therefore, the integration gain is given by $K_I = \delta_d/T$. When there are no spikes, currents are driven to low impedance nodes (d_1-d_4) through M24-M27 by setting the signal dp high.

The use of two different branches (M22 and M23 or M20 and M21) to both charge and discharge the capacitor reduces charge injection on the integration node [13] at the cost of doubling the power consumption required. If switches M20 and M21 (M22 and M23) have the same dimensions, the charges injected from the gate to drain capacitance of the complimentary switches cancel each other.

IV. Chip Results

We used two different input signals to test the chip: a speech signal and a sine wave. The speech signal was sampled at 44.1 kSps with 8 bit resolution. The coder was designed to provide a resolution of 6 bits. The coder input signal $x(t)$, the decoder integrator output $z_R(t)$ and the coder output spikes $y(t)$ are shown in Fig. 5(a). The same signals are presented in detail in Fig. 5(b) to show the absence of coder output spikes when the signal is constant or when its change is smaller than ΔV_{th}.

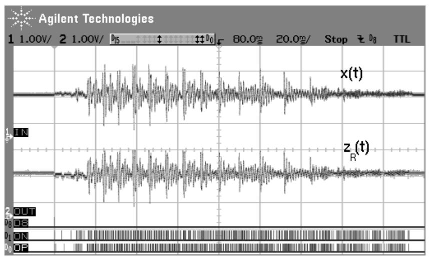
The resolution of the system was measured using a sine wave input signal. The sine wave presents 1.0 Vpp amplitude and 4.4 kHz frequency (f_{in}) sampled at 555 kSps and the coder was designed to provide a 4 bit resolution. A snapshot of the input and output signals is presented in Fig. 6(a). Offline filtering results are shown in the Fig. 6(b) for a digital LPF with a cutoff frequency of 4.4 kHz, the same frequency as the input signal.

The measured resolution of the pre-filter signal is 3.83 bits. The resolution increases to 6.97 and 5.29 bits for post-filter signals, using filter cut-off frequencies equals to f_{in} and $10f_{in}$, respectively. However, this resolution improvement causes attenuation and phase shift of the signal. These results are similar to the simulation values presented in [4]. The measured power consumption of the coder is 0.4 mW, with approximately 90% of if used by the comparator.

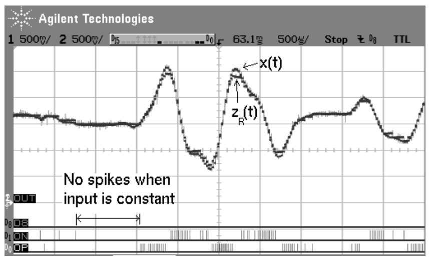
V. Conclusions

In this paper we described a CMOS implementation of a spike event coding scheme. This scheme is intended to be used to transmit analog signals inside a FPAA and/or between different FPAAs using asynchronous events (spikes).

This scheme presents an efficient utilization of channel resources and lower power consumption because no output activity is present when signals are constant. A chip was designed

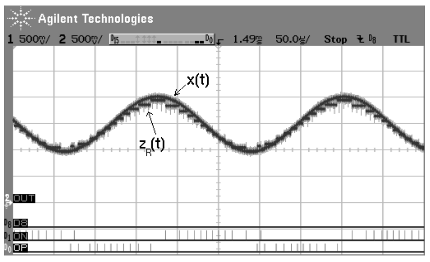


(a)

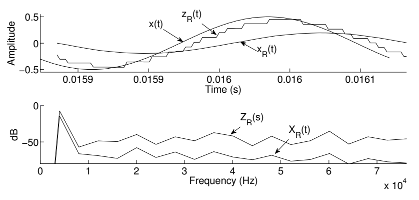


(b)

Fig. 5. (a) Coder input $x(t)$ and decoder integrator output $z_R(t)$ for a 2.5Vpp speech signal. Negative (ON) and positive (OP) output coder spikes are also shown at the bottom of the figure. (b) A detailed view of the same waveforms to show the absence of spikes during the periods when the input signal is constant.



(a)



(b)

Fig. 6. (a) Snapshot of 1.0Vpp sine wave input $x(t)$ and decoder integrator output $z_R(t)$ and negative (ON) and positive (OP) spikes from coder output (bottom). (b) $x(t)$, $z_R(t)$ and the low pass filter output $x_R(t)$ (top). The low pass filter cut-off frequency is the same as the input signal. The frequency spectrum $Z_R(s)$ and $X_R(s)$ of the filter input and output (bottom).

in a 0.35μm process and the experimental results validate the circuit design. This scheme is being implemented within a small array of CABs developed by the authors [14] [15].

Acknowledgment

The authors would like to thank the supports from Engineering and Physical Sciences Research Council and the Scottish Funding Council for the Joint Research Institute with the Heriot-Watt University, part of the Edinburgh Research Partnership in Engineering and Mathematics (ERPem).

References

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

[2] D'Mello, D. R. and Gulak, P. G., "Design approaches to field-programmable analog integrated circuits". *Special Issue on Programmable Analog Systems, Analog Integrated Circuits and Signal Processing*, Hingham: Kluwer Academic Publishers, vol. 17, no. 1-2, pp. 7-34, September 1998.

[3] Papathanasiou, K., Brandtner, T. and Hamilton, A., "Palmo: pulse-based signal processing for programmable analog VLSI". *IEEE Transactions on Circuits and Systems-II, Analog and Digital Signal Processing*, vol. 49, no. 6, pp. 379-389, June 2002.

[4] Gouveia, L., Koickal, T. and Hamilton, A., "An asynchronous spike event coding scheme for programmable analog arrays", in *Proceedings of the IEEE International Symposium on Circuits and Systems*, 2008, pp.1364-1367.

[5] Lazar, A.A. and Toth, L.T., "Perfect recovery and sensitivity analysis of time encoded bandlimited signals", *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol.51, no.10, pp. 2060-2073, October 2004.

[6] Wei, D. and Harris, J.G., "Signal reconstruction from spiking neuron models". *Proceedings of the International Symposium on Circuits and Systems*, vol. 5, pp. 353-356, May 2004.

[7] Baker, R.J., Li, H.W. and Boyce, D.E., *CMOS: Circuit Design, Layout and Simulation*. New York:IEEE Press, 1998.

[8] Fayed, A.A. and Ismail, M., "A High Speed, Low Voltage CMOS Offset Comparator", *Analog Integrated Circuits and Signal Processing*, vol. 36, no. 3, pp. 267-272, September 2003.

[9] Yaklin, D., "Offset Comparator with Common Mode Voltage Stability", U. S. Patent 5517134, May 1997.

[10] Bazes, M., "Two novel fully complementary self-biased CMOS differential amplifiers", *IEEE Journal of Soild-State Circuits*, vol. 26, pp. 165-168, February 1991.

[11] Martin, A., "Translating Concurrent Programs into VLSI Chips", in *Proceedings of the 4th International PARLE Conference on Parallel Architectures and Languages EuropeI*, London:Springer-Verlag, 1992, pp. 515-532.

[12] Boahen, K., "Point-to-point connectivity between neuromorphic chips using address events", *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 47, no. 5, pp. 416-434, May 2000.

[13] Pan, H., "Method and system for a glitch-free differential current steering switch circuit for high speed, high resolution digital-to-analog conversion", US Patent US20060092062, May 2006.

[14] Koickal, T., Hamilton, A. and Gouveia, L., "Programmable analog VLSI architecture based upon event coding", *Second NASA/ESA Conference on Adaptive Hardware and Systems*, pp. 554-562, August 2007.

[15] Koickal, T., Hamilton, A. and Gouveia, L., "Bio-inspired Event Coded Configurable Analog Circuit Block", *Evolvable Systems: From Biology to Hardware*, LNCS 5216, Berlin:Springer-Verlag, 2008, pp. 285-295.

An Asynchronous Spike Event Coding Scheme for Programmable Analog Arrays

Luiz Carlos Gouveia, Thomas Jacob Koickal and Alister Hamilton
School of Engineering and Electronics
University of Edinburgh, EH9 3JL, U.K.
Email: L.Gouveia@ed.ac.uk, {Thomas.Koickal, Alister.Hamilton}@ee.ed.ac.uk

***Abstract*— This paper presents a spike event coding scheme for the communication of analog signals in programmable analog arrays. In the scheme presented here no events are transmitted when the signals are constant leading to low power dissipation and traffic reduction in analog arrays. The design process and the implementation of the scheme in a programmable array context are explained. The validation of the presented scheme is performed using a speech signal. Finally, we demonstrate how the event coded scheme can perform summation of analog signals without additional hardware.**

I. Introduction

A field programmable analog array (FPAA) consists of several configurable analog blocks (CABs) that are programmed to perform specific signal processing functions. An important design problem in a FPAA is the communication of analog signals between CABs. Previous implementations of FPAAs used crossbars or matrix switches [1] [2]. These approaches suffer from signal distortion due to voltage drops, parasitic capacitances along the wires and switches and through signal interference. In an alternative approach [3], pulse width modulated signals were used for transmitting analog information in the array. However, this approach requires a global clock signal to synchronize the transmission.

Recently, asynchronous signal processing based on signal dependent sampling strategies has received increasing interest [4]. A motivation for this type of study is drawn from biology where the brain processes signals in analog domain and transmits them as time events [5]. In event based sampling schemes, the intersampling intervals are quantized instead of the signal amplitude and an event is triggered whenever the input signal crosses prespecified levels along the amplitude domain [6]. In [7], signal amplitude information is coded into the timing sequence and this scheme has been extended to recover information from spiking neurons [8].

In this paper we present a spike event coding scheme for transmitting analog signals between CABs. There are some benefits in using an event based coding scheme in a programmable analog array. First, an event coding transmits signals based on demand. This leads to a better utilization of resources due to traffic reduction. Second, in contrast to synchronous signal processing, event based processing benefit from low energy dissipation, freedom from clock skew, immunity to metastable behavior and low crosstalk. Third, event based processing transmits signals as digital spikes and hence it is more suited to communication between distant CABs, even in different ICs, than analog signals [1] [2], allowing greater scalability.

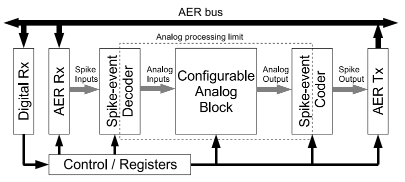


Fig. 1. Block diagram of the spike event communication interface to a configurable analog block (CAB). Spike events are transmitted across the array using AER protocol.

II. Spike Event Coding Scheme

A. Description

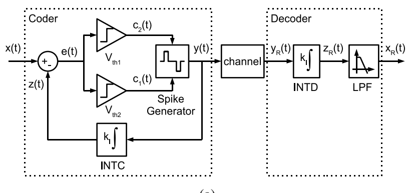
The block diagram representation of a CAB with the spike event communication interface is shown in Fig. 1. Analog signals from the CAB form the input to the spike event coder. In a programmable array context, these spikes can be transmitted between CABs using an AER protocol [9]. The destination CAB reads spikes through an AER receiver and these spike inputs are converted to analog signals at the spike event decoder. The control registers are used for configuration and control of the circuit block.

The spike event coding scheme is shown in Fig. 2(a). This scheme is based on the principle of irregular sampling schemes described in [4] and [6], where it was used to implement asynchronous A/D converters. The event coder tracks the input signal by bounding an error signal given by:

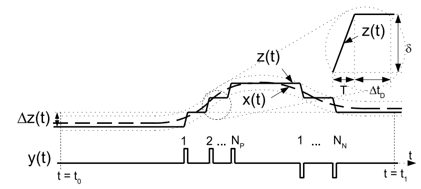
$$e(t) = x(t) - z(t) = x(t) - \int y(t)dt \tag{1}$$

where $e(t)$ is the error between the analog input signal $x(t)$ and coder feedback integrator (INTC) output $z(t)$. The coder output spikes $y(t)$ are sent to the communication channel and to the input of the feedback integrator. The decoder output $x_R(t)$ is an analog signal given by:

$$x_R(t) = LPF(z_R(t)) \approx \overline{z_R(t)} = \overline{\int y_R(t)dt} \tag{2}$$



(a)



(b)

Fig. 2. (a) Event coding block diagram and (b) waveforms example. An example of predict behavior of some signals are shown in b), where δ is the tracking step, T is the spike width and Δt_D is the time interval between successive spikes.

If the channel is ideal, $y_R(t) = y(t)$ and

$$x_R(t) \approx \overline{\int y(t)dt} = \overline{z(t)} = \overline{x(t) - e(t)} \qquad (3)$$

The error between the decoder output $x_R(t)$ and coder input $x(t)$ is bounded, $|e|_{max} \leq \delta$, where δ, the tracking step or quantization error, is a system parameter.

The outputs of the event coder are represented by positive and negative fixed short duration pulses (spikes). These spikes are generated by the spike generator when the comparators change their states. Each positive or negative spike generates an incremental or decremental change (δ) at the output of the both integrators (INTC and INTD). Although δ can be varied based on the characteristics of the input signal, in this paper we consider the case for fixed δ only.

The change in the output of both integrators is given by:

$$\Delta z(t) = \delta\ (Np - Nn) \qquad (4)$$

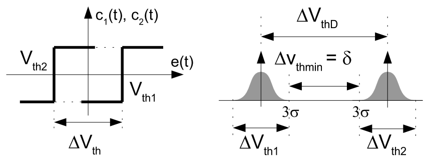
where $Np(Nn)$ is the number of previous positive (negative) spikes since t_0 as shown in Fig. 2(b).

B. Design

In this section we discuss the design process of the spike event coder and decoder. The first step in the design of the event coder is to determine the tracking step δ:

$$\delta = \frac{\Delta x(t)_{max}}{2^{N_B}} \qquad (5)$$

where $\Delta x(t)_{max}$ is the input dynamic range and N_B is the desired resolution in bits.



(a) (b)

Fig. 3. Comparators threshold design. a) Comparators transfer function and b) Comparator offsets ΔV_{th1} and ΔV_{th1} are used to design the threshold difference ΔV_{thD}.

The tracking step is used to design the comparators thresholds difference $\Delta V_{th} = V_{th1} - V_{th2}$, as shown in Fig. 3(a). Ideally, this difference is equal to the tracking step δ. However, due to the comparators offset the actual ΔV_{th} is bounded ($\Delta V_{thD} + 6\sigma \geq \Delta V_{th} \geq \Delta V_{thD} - 6\sigma$) by a function of the comparator offset standard deviation σ and the designed thresholds difference ΔV_{thD} (Fig. 3(b)). Therefore, the comparators thresholds difference is designed to meet the specification:

$$\Delta V_{thD} \geq \delta + 6\sigma \qquad (6)$$

Another design parameter is the spike width T and it is designed according to the input signal and the AER system characteristics. In order to reduce the overload of the communication channel, the spike generator sets a minimum period for the interval between two successive output spikes. This "refractory period" is given by $\Delta t_{Dmin} = kT$. Using Δt_{Dmin} and the specification of the maximum derivative of the input $\left|\frac{dx(t)}{dt}\right|_{max}$, the spike width is determined:

$$T = \frac{\delta}{(k+1)\left|\frac{dx(t)}{dt}\right|_{max}} \qquad (7)$$

In an AER system, one of the most important specification is the output spike frequency. The coder output spike frequency is a function of the magnitude of the input derivative:

$$f = \frac{1}{T + \Delta t_D} = \frac{\left|\frac{dx(t)}{dt}\right|}{\delta} \qquad (8)$$

From (8), we see that this event coding scheme presents a null output activity when the input signal is constant. This characteristic is beneficial in a programmable analog framework where significant number of bias signals are present.

From (7) and (8), the maximum output frequency is:

$$f_{max} = \frac{1}{(k+1)T} \qquad (9)$$

The spike width T and the tracking step δ are used to design the coder and decoder integrator gains given by $K_I = \frac{\delta}{T}$.

Finally, the pole of the decoder low pass filter (LPF) is a key design parameter as it improves the resolution by attenuating the undesirable out-band high frequency harmonics generated during the decoding process. Ideally, the filter should provide total rejection of out-band harmonics with zero in-band attenuation. However, practical implementation of this characteristic being unrealizable, the dominant pole of the filter is designed to be near the cutoff frequency ω, for an input signal with bandwidth of ω.

III. Simulation Results

The event coding scheme was simulated using a speech signal and a pure tone as the coder input. In order to depict the coder functionality clearly, the coder was implemented to provide resolution of 4 bits.

Response to a Speech Signal: The response of the spike event coding to the speech signal is shown in Fig. 4. The speech signal is shown in Fig. 4(a). The decoded signal $x_R(t)$ at the output of a first-order LPF shows a close match with the input speech signal $x(t)$ (see expanded plot Fig. 4(b)). The pole of the LPF was designed to be at 4 kHz (allocated voice bandwidth). Fig. 4(c) and 4(d) demonstrate two important characteristics. First, Fig. 4(c) shows the error $e(t)$ is bounded by ΔV_{th}. Second, Figs. 4(b) and 4(d) show that no spikes are transmitted when the input signal is relatively constant thereby reducing the communication traffic and leading to a better utilization of the resources.

THD measurement: The THD of the system was measured using a 4 kHz sine wave as the input signal. Two LPFs were designed to demonstrate the influence of the pole design: one with the pole at 4 kHz (LPF1) and the second at 40 kHz (LPF2). The coder input $x(t)$, the decoder integrator output $z_R(t)$ and the LPF1 and LPF2 outputs $x_{R1}(t)$ and $x_{R2}(t)$, respectively, are shown in Fig. 5(a). Fig. 5(b) shows the frequency spectrum of the output signals.

The specified resolution of spike event coder is obtained at the decoder integrator output (4.0 bits). The resolution increases to 5.3 bits and 6.7 bits using the filters LPF2 and LPF1, respectively. As stated in Section II, the improvement in resolution in LPF1 is attributed to the larger attenuation of the harmonics because the pole is designed at a lower frequency.

The influence of the refractory period Δt_{Dmin} on the coder performance is shown in Fig. 5(a). Because the initial state of the coder integrator was set to zero and $x(t_0) = 1$, the error signal $e(t)$ is initially greater than δ. The error decreases for each successive output spike occurrence. By choosing $\Delta t_{Dmin} \approx 4.6\mu s$ and $T \approx 100ns$ and according to (9), the maximum output spike frequency is 213 kHz. Therefore, the refractory period Δt_{Dmin} determines the initial tracking speed of the coder.

IV. Computational Application

In a programmable analog array, an important requirement is the capability of adding analog signals; the summation of the outputs of hundreds or thousand synapses in a neuromorphic system is an example. Due to the large number of operators, it is desirable that this operation can be performed without additional hardware like the summation of currents in analog

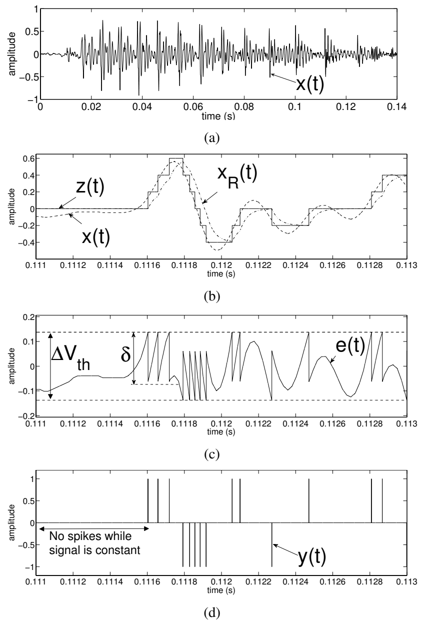


Fig. 4. Example of coding and decoding of a speech signal. (a) The complete speech signal $x(t)$. (b) The expanded plot showing the decoding $x_R(t)$ of the speech signal and the integrator output $z(t)$. (c) The error signal $e(t)$ bounded by the difference of comparators thresholds ΔV_{th}. (d) The spike event coder output $y(t)$.

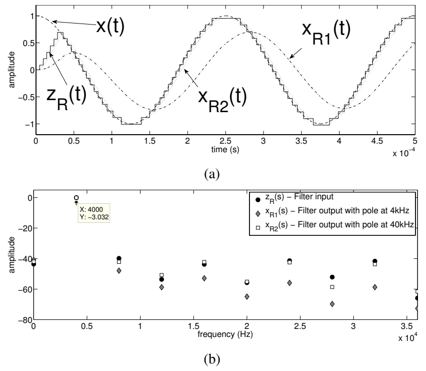


Fig. 5. THD simulation. (a) Decoder output for a 4 kHz sine wave. The integrator output $z(t)$ tracks the input $x(t)$. The signals $x_{R1}(t)$ and $x_{R2}(t)$ are the outputs of filters: the first with the pole at 4kHz and the second at 40kHz. The initial state of both integrators are set to zero. (b) The frequency spectrum of LPF input, $Z_R(s)$ and outputs $X_{R1}(s)$ and $X_{R2}(s)$ were computed for the last period (250 μs < t < 500 μs).

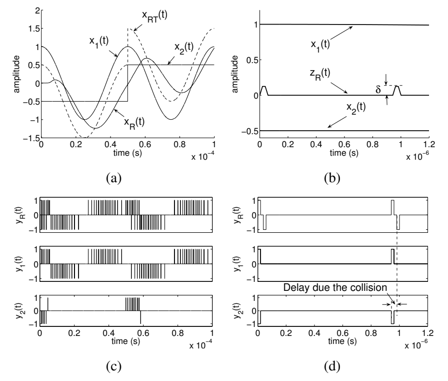


Fig. 6. Example of summing operation. The inputs $x_1(t)$ and $x_2(t)$, the decoder output $x_R(t)$ and a predicted result $x_{RT}(t)$ are shown in graphs a) and b). The spike outputs, $y_1(t)$ and $y_2(t)$, and an AER arbiter output $y_R(t)$ are shown in c) and d). The graphs b) and d) show of the effect of spike collisions: an error with amplitude δ appears on the output $x_R(t)$ between the transmissions of $y_1(t)$ and $y_2(t)$, which were involved in a collision.

domain [10]. Here we show how summation is performed with event coding without any extra hardware.

Using (4), the summation signal $s(t)$ with j operators is:

$$s(t) = \delta \left(\sum_{i=1}^{j} N_{pi} - \sum_{i=1}^{j} N_{ni} \right) \tag{10}$$

where N_{pi} and N_{ni} are the number of positive and negative spikes, respectively, received from the ith operator.

Since AER protocol is used to transmit spike events, collisions during the access to the channel are possible and an arbiter is used to resolve them. Collisions lead to an error in the summation process. This error is given by $\epsilon = \delta\ (N_{pc} - N_{nc})$, where $N_{pc}(N_{nc})$ is the number of positive (negative) spikes in the collision. One possible resolution of the conflicts is performed by queuing and transmitting successively all events involved in the collision (1-persistent). This method was used for the simulation.

Simulation: The results showing the summation of a sine signal $x_1(t)$ and a step signal $x_2(t)$ are shown in Fig. 6, together with the decoder output $x_R(t)$ and the predicted result $x_{RT}(t)$. The coders outputs $y_1(t)$ and $y_2(t)$ and the decoder input $y_R(t)$ are shown in Fig. 6(c).

The expanded results in Fig. 6(b) show the effect of spike collision (Fig. 6(d)) in the summation result using a 1-persistent arbiter: an error (with amplitude δ in this case) between the transmission of the spikes $y_1(t)$ and $y_2(t)$.

The decoder output $x_R(t)$ follows the predicted result $x_{RT}(t)$, except for the limited tracking time in the step signal coding and for the spike collisions in the AER bus.

V. Conclusions

In this paper we presented a spike event coding scheme for the communication of analog signals in a programmable array. The scheme transmits spike events based on input signal activity thereby providing efficient utilization of resources and lower power consumption. Further the events are transmitted digitally providing improved scalability in building large programmable arrays. The methodology of the scheme and the parameters design process were presented. The functionality of the event coded scheme was validated through simulations. We demonstrated how event coding can be used to add analog signals without extra hardware; an important feature in programmable analog systems. Currently the circuits of the spike event communication interface are being implemented on a chip to interface CABs in a programmable array developed by the authors [11].

Acknowledgment

The authors would like to thank Engineering and Physical Sciences Research Council (EPSRC), for supporting this work under the grant to University of Edinburgh (EP/C015789/1)

References

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

[2] D. R. D'Mello, and P. G. Gulak, *Design approaches to field-programmable analog integrated circuits*. In Special Issue on Programmable Analog Systems, Analog Integrated Circuits and Signal Processing, Hingham: Kluwer Academic Publishers, vol. 17, no. 1-2, pp. 7-34, September 1998.

[3] K. Papathanasiou, T. Brandtner and A. Hamilton, *Palmo: pulse-based signal processing for programmable analog VLSI*. In IEEE Transactions on Circuits and Systems-II, Analog and Digital Signal Processing, vol. 49, no. 6, pp. 379-389, June 2002.

[4] Y. W. Li, K. L. Shepard and Y. P. Tsividis, *Continuous-time digital signal processors*, In Proceedings of eleventh IEEE International Symposium on Asynchronous Circuits and Systems, pp. 138-143, March 2005.

[5] W. Gerstner, *Spiking neurons*. In Pulsed neural networks. W. Maass and C. Bishop eds., Cambridge: MIT Press, pp. 4-53, 1998.

[6] E. Allier, G. Sicard, L. Fesquet and M. Renaudin, *A new class of asynchronous A/D converters based on time quantization*, In Proceedings of Ninth International Symposium on Asynchronous Circuits and Systems, pp. 196-205, May 2003.

[7] A. A. Lazar, L. T. Toth, *Perfect recovery and sensitivity analysis of time encoded bandlimited signals*, IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, vol.51, no.10, pp. 2060-2073, October 2004.

[8] D. Wei and J. G. Harris, *Signal reconstruction from spiking neuron models*. In Proceedings of the International Symposium on Circuits and Systems, vol. 5, pp. 353-356, May 2004.

[9] K. Boahen, *Point-to-point connectivity between neuromorphic chips using address events*. In IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, vol. 47, no. 5, pp. 416-434, May 2000.

[10] G. Indiveri, E. Chicca, R. Douglas, *A VLSI array of low-power spiking neurons and bistable synapses with spike-timing dependent plasticity*, IEEE Transactions on Neural Networks, vol. 17, no. 1, pp. 211-221, Jan. 2006.

[11] T. Koickal, A. Hamilton and L. Gouveia, *Programmable analog VLSI architecture based upon event coding*, Second NASA/ESA Conference on Adaptive Hardware and Systems, pp. 554-562, August 2007.

References

[1] C. Azzolini, D. Vecchi, A. Boni, and G. Chiorboli, “High-level Accurate Model of High-resolution Pipelined ADC’s,” in *IEEE Instrumentation and Measurement Technology Conference Proceedings*, no. April, pp. 261–265, IEEE, Apr. 2006.

[2] P. Dudek and P. Hicks, “A CMOS general-purpose sampled-data analog processing element,” *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 47, pp. 467–473, May 2000.

[3] J. Mills and C. Daffinger, “An analog VLSI array processor for classical and connectionist AI,” in *Proceedings of the International Conference on Application Specific Array Processors*, no. 812, pp. 367–378, IEEE Comput. Soc. Press, 1990.

[4] E. Lee and P. Gulak, “A CMOS field-programmable analog array,” *IEEE Journal of Solid-State Circuits*, vol. 26, no. 12, pp. 1860–1867, 1991.

[5] D. D’Mello and P. Gulak, “Design Approaches to Field-Programmable Analog Integrated Circuits,” *Analog Integrated Circuits Signal Process.*, vol. 17, pp. 7–34, 1998.

[6] C. Mead, “Neuromorphic electronic systems,” *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1629–1636, 1990.

[7] R. Sarpeshkar, *Efficient precise computation with noisy components : extrapolating from an electronic cochlea to the brain.* Ph.d., California Institute of Technology, 1997.

[8] P. Hasler, “Low-power programmable signal processing,” in *Proceedings of the Fifth International Workshop on System-on-Chip for Real-Time Applications*, (Washington, DC, USA), pp. 413 – 418, IEEE Computer Society, 2005.

[9] A. Stoica, R. Zebulum, D. Keymeulen, R. Tawel, T. Daud, and A. Thakoor, “Reconfigurable VLSI architectures for evolvable hardware: from experimental field programmable transistor arrays to evolution-oriented chips,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 9, pp. 227–232, Feb. 2001.

[10] T. Hall and C. Twigg, “Field-programmable analog arrays enable mixed-signal prototyping of embedded systems,” in *48th Midwest Symposium on Circuits and Systems, 2005*, vol. 1, pp. 83–86, 2005.

[11] R. T. Edwards, K. Strohbehn, S. E. Jaskulek, and R. Katz, “Analog module architecture for space-qualified field-programmable mixed-signal arrays,” in *2nd annual Military and Aerospace Applications of Programmable Devices and Technologies Conference*, (Laurel, MD), 1999.

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

[13] T. Roska and L. O. Chua, "The CNN universal machine: an analogic array computer," *Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 40, pp. 163–173, 1993.

[14] C. Santini, J. Amaral, M. Pacheco, M. Vellasco, and M. Szwarcman, "Evolutionary analog circuit design on a programmable analog multiplexer array," in *Proceedings of the IEEE International Conference on Field-Programmable Technology*, pp. 189–196, IEEE, 2002.

[15] M. A. Terry, J. Marcus, M. Farrell, V. Aggarwal, and U.-M. O'Reilly, "GRACE : Generative robust analog circuit exploration," *Lecture notes in computer science*, vol. 3907, pp. 332–343, 2006.

[16] P. Dong, G. Bilbro, and M.-Y. Chow, "Implementation of Artificial Neural Network for Real Time Applications Using Field Programmable Analog Arrays," in *Proceedings of the IEEE International Joint Conference on Neural Network*, pp. 1518–1524, IEEE, 2006.

[17] S. Bridges, M. Figueroa, D. Hsu, and C. Diorio, "Field-Programmable Learning Arrays," *Neural Information Processing Systems*, vol. 15, pp. 1155–1162, 2003.

[18] Y. Tsividis, "Signal Processors with Transfer Function Coefficients Determined by Timing," *IEEE Transactions on Circuits and Systems*, vol. 29, no. 12, pp. 807–817, 1982.

[19] C. Looby and C. Lyden, "Op-amp based CMOS field-programmable analogue array," *IEE Proceedings - Circuits, Devices and Systems*, vol. 147, no. 2, p. 93, 2000.

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

[21] K. Papathanasiou, T. Brandtner, and A. Hamilton, "Palmo: pulse-based signal processing for programmable analog VLSI," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 49, no. 6, pp. 379–389, 2002.

[22] R. Chawla, *Power-efficient analog systems to perform signal-processing using floating-gate MOS device for portable applications*. Phd, Georgia Institute of Technology, 2004.

[23] C. Evans-Pughe, "Programmable Analog: More Gain, Less Pain," 2001.

[24] Z. Plc., "Totally Re-Configurable Analog Circuit - TRAC020LH." 1999.

[25] E. Cantó, J. Moreno, J. Cabestany, I. Lacadena, and J. Insenser, "Implementation of Virtual Circuits by Means of the FIPSOC Devices," *Field-Programmable Logic and Applications: The Roadmap to Reconfigurable Computing*, vol. 1896, pp. 87–95, 2000.

[26] J. Faura, I. Lacadena, A. Torralba, and J. Insenser, "Programmable analog hardware: a case study," in *IEEE International Conference on Electronics, Circuits and Systems*, vol. 1, pp. 297–300, IEEE, 1998.

[27] L. Semiconductor Corp., "ispPAC Overview," 2001.

[28] E. Ramsden, “The ispPAC family of reconfigurable analog circuits,” in *Proceedings of the 3rd NASA/DoD Workshop on Evolvable Hardware*, (Long Beach, CA), pp. 176–181, IEEE Comput. Soc, 2001.

[29] C. S. Corporation, “PSoC Programmable System-on-Chip Technical Reference Manual,” 2010.

[30] M. Mar, B. Sullam, and E. Blom, “An architecture for a configurable mixed-signal device,” *IEEE Journal of Solid-State Circuits*, vol. 38, pp. 565–568, Mar. 2003.

[31] A. Bratt and I. Macbeth, “DPAD2A field programmable analog array,” *Analog Integrated Circuits and Signal Processing*, vol. 17, no. 1, pp. 67–89, 1998.

[32] M. Jankovec and M. Topic, “Analog circuit development system,” in *The IEEE Region 8 EUROCON 2003. Computer as a Tool.*, vol. 1, pp. 125–129, IEEE, 2003.

[33] A. Inc., “AN231E04 Datasheet Dynamically Reconfigurable dpASP,” 2008.

[34] G. Linan, R. Dominguez-Castro, S. Espejo, and A. Rodriguez-Vazquez, “ACE16K: An advanced focal-plane analog programmable array processor,” in *Proceedings of the 27th European Solid-State Circuits Conference*, pp. 201–204, 2001.

[35] D. Varghese and J. Ross, “A continuous-time hierarchical field programmable analogue array,” in *Proceedings of the IEEE Custom Integrated Circuits Conference*, pp. 190–193, IEEE, 2005.

[36] T. Hall, P. Hasler, and D. Anderson, “Field-Programmable Analog Arrays: A Floating-Gate Approach,” in *Proceedings of the 12th International Conference on Field-Programmable Logic and Applications*, (London, UK), pp. 424—-433, Springer-Verlag, 2002.

[37] P. Hasler, B. A. Minch, and C. Diorio, “Adaptive circuits using pFET floating-gate devices,” in *Proceedings 20th Anniversary Conference on Advanced Research in VLSI*, pp. 215–229, IEEE, Mar. 1999.

[38] M. R. Kucic, *Analog computing arrays*. Phd, Georgia Institute of Technology, 2004.

[39] R. Zebulum, A. Stoica, and D. Keymeulen, “The design process of an evolutionary oriented reconfigurable architecture,” in *Proceedings of the 2000 Congress on Evolutionary Computation*, pp. 529–536, IEEE, 2000.

[40] J. Langeheine, J. Becker, S. Folling, K. Meier, and J. Schemmel, “A CMOS FPTA chip for intrinsic hardware evolution of analog electronic circuits,” in *Proceedings 3rd NASA/-DoD Workshop on Evolvable Hardware*, pp. 172–175, IEEE Comput. Soc, 2001.

[41] A. Stoica, D. Keymeulen, R. Zebulum, M. Mojarradi, S. Katkoori, and T. Daud, “Adaptive and evolvable analog electronics for space applications,” *Evolvable Systems: From Biology to Hardware*, vol. 4684, pp. 379–390, 2007.

[42] J. Becker and Y. Manoli, “A continuous-time field programmable analog array (FPAA) consisting of digitally reconfigurable Gm-cells,” in *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 1, pp. I–1092–5, IEEE, May 2004.

[43] F. Henrici, J. Becker, S. Trendelenburg, D. DeDorigo, M. Ortmanns, and Y. Manoli, “A Field Programmable Analog Array using floating gates for high resolution tuning,” in *Proceedings of the IEEE International Symposium on Circuits and Systems*, pp. 265–268, IEEE, May 2009.

[44] L. Chua and T. Roska, “The CNN paradigm,” *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 40, pp. 147–156, Mar. 1993.

[45] P. Kinget and M. S. J. Steyaert, “A programmable analog cellular neural network CMOS chip for high speed image processing,” *IEEE Journal of Solid-State Circuits*, vol. 30, pp. 235–243, Mar. 1995.

[46] D. Varghese and J. N. Ross, “A continuous-time hierarchical field programmable analogue array for rapid prototyping and hierarchical approach to analogue systems design,” in *Proceedings of the 18th annual symposium on Integrated circuits and system design*, (New York, NY, USA), pp. 248–253, ACM, 2005.

[47] J. Misra and I. Saha, “Artificial neural networks in hardware: A survey of two decades of progress,” *Neurocomputing*, vol. 74, pp. 239–255, May 2010.

[48] E. Swartzlander Jr and R. Jones Jr, “Digital neural network implementation,” in *Proceedings of the 11th Conference on Computers and Communications*, pp. 722–728, IEEE, 1992.

[49] S. Bettola, “High performance fault-tolerant digital neural networks,” *IEEE Transactions on Computers*, vol. 24, pp. 483–363, Mar. 1998.

[50] M. Holler, S. Tam, H. Castro, and R. Benson, “An electrically trainable artificial neural network (ETANN) with 10240 ”floating gate” synapses,” in *Proceedings of the International Joint Conference on Neural Networks*, pp. 191–196, IEEE, June 1989.

[51] V. Koosh, *Analog computation and learning in VLSI*. Phd, California Institute of Technology, 2001.

[52] A. Almeida and J. Franca, “A mixed-mode architecture for implementation of analog neural networks with digital programmability,” in *Proceedings of International Conference on Neural Networks*, pp. 887–890, IEEE, Oct. 1993.

[53] S. Skinner, J. Steck, A. Cruz-Cabrera, and E. Behrman, “Optical hardware backpropagation neural network,” in *Proceedings of the International Joint Conference on Neural Networks*, vol. 4, pp. 2351–2356, Ieee, 1999.

[54] J. Liu and D. Liang, “A Survey of FPGA-Based Hardware Implementation of ANNs,” in *Proceedings of International Conference on Neural Networks and Brain*, vol. 2, pp. 915–918, IEEE, 2005.

[55] J. Maher, B. Ginley, P. Rocke, and F. Morgan, “Intrinsic Hardware Evolution of Neural Networks in Reconfigurable Analogue and Digital Devices,” in *IEEE Symposium on Field-Programmable Custom Computing Machines*, pp. 321–322, IEEE, Apr. 2006.

[56] E. Farquhar, C. Gordon, and P. Hasler, “A field programmable neural array,” in *IEEE International Symposium on Circuits and Systems*, pp. 4114 – 4117, IEEE, 2006.

[57] E. Chicca, G. Indiveri, and R. Douglas, “An event-based VLSI network of integrate-and-fire neurons,” in *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 5, pp. V–357–60, IEEE, 2004.

[58] L. Maguire, T. Mcginnity, B. Glackin, A. Ghani, A. Belatreche, and J. Harkin, “Challenges for large-scale implementations of spiking neural networks on FPGAs,” *Neurocomputing*, vol. 71, pp. 13–29, Dec. 2007.

[59] W. Maass, “Computing with spikes,” *Special Issue on Foundations of Information Processing of TELEMATIK*, vol. 8, no. 1, pp. 32–36, 2002.

[60] E. M. Izhikevich, “Which model to use for cortical spiking neurons?,” *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, vol. 15, pp. 1063–70, Sept. 2004.

[61] R. Jolivet, A. Rauch, H. R. Lüscher, and W. Gerstner, “Integrate-and-Fire models with adaptation are good enough,” *Advances in neural information processing systems*, pp. 595 – 602, 2006.

[62] A. van Schaik, “Building blocks for electronic spiking neural networks.,” *Neural networks : the official journal of the International Neural Network Society*, vol. 14, no. 6-7, pp. 617–28, 2006.

[63] G. Indiveri, “A low-power adaptive integrate-and-fire neuron circuit,” in *IEEE International Symposium on Circuits and Systems*, vol. 4, pp. 820–823, IEEE, May 2003.

[64] T. J. Koickal, L. C. Gouveia, and A. Hamilton, “A programmable time event coded circuit block for reconfigurable neuromorphic computing,” in *International work conference on Artificial neural networks*, vol. 1, (San Sebastian, Spain), pp. 430–437, Springer-Verlag, 2007.

[65] M. Mahowald, *VLSI analogs of neuronal visual processing: a synthesis of form and function*. Phd, California Institute of Technology, 1992.

[66] C. Wolff, G. Hartmann, and U. Ruckert, “ParSPIKE-a parallel DSP-accelerator for dynamic simulation of large spiking neural networks,” in *Proceedings of the 7th International Conference on Microelectronics for Neural, Fuzzy and Bio-Inspired Systems*, pp. 324–331, IEEE Comput. Soc, 1999.

[67] R. Guerrero-Rivera, A. Morrison, M. Diesmann, and T. C. Pearce, “Programmable logic construction kits for hyper-real-time neuronal modeling.,” *Neural computation*, vol. 18, pp. 2651–79, Nov. 2006.

[68] P. Rocke, B. Mcginley, J. Maher, F. Morgan, and J. Harkin, “Investigating the suitability of FPAAs for evolved hardware spiking neural networks,” *Evolvable Systems: From Biology to Hardware*, vol. 5216, pp. 118–129, 2008.

[69] Y. Maeda, T. Hiramatsu, S. Miyoshi, and H. Hikawa, “Pulse coupled oscillator with learning capability using simultaneous perturbation and its FPAA implementation,” in *ICCAS-SICE International Joint Conference*, no. 1, pp. 3142–3145, IEEE, Aug. 2009.

[70] A. Corporation, “Stratix II Architecture, Stratix II Device Family Data Sheet,” 2007.

[71] Xilinx, “XC4000E and XC4000X Series Field Programmable Gate Arrays,” 1999.

[72] Y. Creten, J. De Hert, O. Charliert, P. Merken, J. Putzeys, and C. Van Hoof, “Demonstration of an 4.2 K analog switch matrix in a standard 0.7 μ CMOS process,” *Journal de Physique IV (Proceedings)*, vol. 12, pp. 211–213, May 2002.

[73] J. Luo, *Circuit Design and Routing For Field Programmable Analog Arrays*. Dissertation, University of Maryland, 2005.

[74] E. Rent, “Microminiature packagingLogic block to pin ratio,” 1960.

[75] V. Beiu and W. Ibrahim, “Does the brain really outperform Rents rule?,” *2008 IEEE International Symposium on Circuits and Systems*, pp. 640–643, May 2008.

[76] M. A. Sivilotti, *Wiring Considerations in Analog VLSI Systems, with Application to Field-Programmable Networks*. Ph.d., California Institute of Technology, 1991.

[77] F. Baskaya, D. V. Anderson, and S. K. Lim, “Net-Sensitivity-Based Optimization of Large-Scale Field-Programmable Analog Array (FPAA) Placement and Routing,” *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 56, pp. 565–569, July 2009.

[78] G. Pratt, *Pulse computation*. Phd, Massachusetts Institute of Technology, 1989.

[79] L. Reyneri, “Theoretical and implementation aspects of pulse streams: an overview,” in *7th International Conference on Microelectronics for Neural, Fuzzy and Bio-Inspired Systems*, (Granada), pp. 78–89, 1999.

[80] A. F. Murray, D. Del Corso, and L. Tarassenko, “Pulse-stream VLSI neural networks mixing analog and digital techniques.,” *IEEE transactions on neural networks*, vol. 2, pp. 193–204, Jan. 1991.

[81] R. Vogelstein, U. Mallik, and G. Cauwenberghs, “Silicon spike-based synaptic array and address-event transceiver,” in *IEEE International Symposium on Circuits and Systems*, vol. 5, pp. 385–388, IEEE, May 2004.

[82] G. Indiveri, “VLSI reconfigurable networks of integrate-and-fire neurons with spike-timing dependent plasticity,” *The Neuromorphic Engineer Newsletter*, vol. 2, no. 1, pp. 4–7, 2005.

[83] R. Jolivet, A. Rauch, H. R. Lüscher, and W. Gerstner, “Integrate-and-Fire models with adaptation are good enough: predicting spike times under random current injection,” *Advances in neural information processing systems*, vol. 18, pp. 595–602, 2006.

[84] W. Gerstner, R. Kempter, J. L. van Hemmen, and H. Wagner, “A neuronal learning rule for sub-millisecond temporal coding.,” *Nature*, vol. 383, pp. 76–81, Oct. 1996.

[85] T. J. Koickal, A. Hamilton, S. L. Tan, J. A. Covington, J. W. Gardner, and T. C. Pearce, “Analog VLSI Circuit Implementation of an Adaptive Neuromorphic Olfaction Chip,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 54, pp. 60–73, Jan. 2007.

[86] S. J. Thorpe, A. Delorme, and R. Vanrullen, “Spike-based strategies for rapid processing Spike-based strategies for rapid processing,” *Field-Programmable Logic and Applications: The Roadmap to Reconfigurable Computing*, vol. 14, no. 33, pp. 715–726, 2001.

[87] A. Lazar and L. Toth, “Perfect recovery and sensitivity analysis of time encoded bandlimited signals,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 51, no. 10, pp. 2060–2073, 2004.

[88] D. Wei and J. Harris, “Signal reconstruction from spiking neuron models,” in *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 5, pp. 353 – 356, IEEE, May 2004.

[89] T. Morie, S. Sakabayashi, H. Ando, M. Nagata, and A. Iwata, “Pulse Modulation Circuit Techniques for Nonlinear Dynamical Systems,” in *International Symposium on Nonlinear Theory and its Application*, pp. 447–450, 1998.

[90] L. F. Cristofoli, A. Henglez, J. Benfica, L. Bolzani, F. Vargas, A. Atienza, and F. Silva, “On the comparison of synchronous versus asynchronous circuits under the scope of conducted power-supply noise,” in *2010 Asia-Pacific International Symposium on Electromagnetic Compatibility*, pp. 1047–1050, Ieee, 2010.

[91] D. Kościelnik and M. Miśkowicz, “Asynchronous Sigma-Delta analog-to digital converter based on the charge pump integrator,” *Analog Integrated Circuits and Signal Processing*, vol. 55, pp. 223–238, Oct. 2007.

[92] S. Park, “Principles of sigma-delta modulation for analog-to-digital converters,” *Motorola Application Notes APR8*, 1999.

[93] L. C. Gouveia, T. J. Koickal, and A. Hamilton, “An Asynchronous Spike Event Coding Scheme for Programmable Analog Arrays,” in *IEEE International Symposium on Circuits and Systems.*, (Seattle), pp. 1364–1367, IEEE, May 2008.

[94] S.-C. Liu and A. V. Schaik, “AER EAR: A Matched Silicon Cochlea Pair With Address Event Representation Interface,” *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 54, no. 1, pp. 48–59, 2007.

[95] K. Boahen, “Point-to-point connectivity between neuromorphic chips using address events,” *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 47, no. 5, pp. 416–434, 2000.

[96] P. Häfliger and E. Aasebø, “A rank encoder: Adaptive analog to digital conversion exploiting time domain spike signal processing,” *Analog Integrated Circuits and Signal Processing*, vol. 40, no. 1, pp. 39–51, 2004.

[97] S. Deiss, R. Douglas, and A. Whatley, “A Pulse-Coded Communications Infrastructure for Neuromorphic Systems,” in *Pulsed Neural Networks* (W. Maass and C. Bishop, eds.), vol. chapter6pa, ch. 6, pp. 157–178, MIT Press, 1999.

[98] P. Merolla, J. Arthur, B. Shi, and K. Boahen, “Expandable Networks for Neuromorphic Chips,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 54, no. 2, pp. 301–311, 2007.

[99] J. Lazzaro and J. Wawrzynek, “A multi-sender asynchronous extension to the AER protocol,” in *16th IEEE Conference on Advanced Research in VLSI*, pp. 158–169, IEEE, Mar. 1995.

[100] J. Georgiou and A. Andreou, “High-speed, address-encoding arbiter architecture,” *Electronics Letters*, vol. 42, no. 3, pp. 170–171, 2006.

[101] V. Brajovic, “Lossless non-arbitrated address-event coding,” in *IEEE International Symposium on Circuits and Systems*, vol. 5, pp. 825–828, IEEE, May 2003.

[102] H. Inose, T. Aoki, and K. Watanabe, “Asynchronous delta-modulation system,” *Electronics Letters*, vol. 2, no. 3, p. 95, 1966.

[103] J. Das and P. Sharma, “Some asynchronous pulse-modulation systems,” *Electronics Letters*, vol. 3, p. 284, 1967.

[104] Y. Li, K. Shepard, and Y. Tsividis, “Continuous-time digital signal processors,” in *IEEE International Symposium on Asynchronous Circuits and Systems*, pp. 138–143, IEEE, Mar. 2005.

[105] V. Balasubramanian, A. Heragu, and C. Enz, “Analysis of ultralow-power asynchronous ADCs,” in *Proceedings of IEEE International Symposium on Circuits and Systems*, pp. 3593–3596, IEEE, May 2010.

[106] E. Allier, G. Sicard, L. Fesquet, and M. Renaudin, “A New Class of Asynchronous A/D Converters Based on Time Quantization,” in *9th International Symposium on Asynchronous Circuits and Systems*, vol. 12, pp. 197–205, IEEE Computer Society, 2003.

[107] D. Chen, Y. Li, D. Xu, J. Harris, and J. Principe, “Asynchronous Biphasic Pulse Signal Coding and Its CMOS Realization,” in *2006 IEEE International Symposium on Circuits and Systems*, (Island of Kos), pp. 2293–2296, IEEE, 2006.

[108] M. Miskowicz, “Send-On-Delta Concept: An Event-Based Data Reporting Strategy,” *Sensors*, vol. 6, pp. 49–63, Jan. 2006.

[109] T. Kato and K. Miyao, “Modified hysteresis control with minor loops for single-phase full-bridge inverters,” in *IEEE Industry Applications Society Annual Meeting*, pp. 689–693, IEEE, Oct. 1988.

[110] J. Das and P. Sharma, “Rectangular-wave modulation - a hybrid PLM-FM system,” *Electronics Letters*, vol. 2, no. 1, p. 7, 1966.

[111] Y. Zhu and S. Leung, “A nonuniform sampling delta modulation technique,” in *Proceedings of TENCON’94 - 1994 IEEE Region 10’s 9th Annual International Conference on: ’Frontiers of Computer Technology’*, pp. 708–712, IEEE, 1994.

[112] J. Greefkes and K. Riemens, “Code modulation with digitally controlled companding for speech transmission,” *Philips Technical Review*, pp. 335–353, 1970.

[113] E. Roza, “Analog-to-digital conversion via duty-cycle modulation,” *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 44, no. 11, pp. 907–914, 1997.

[114] C. Kikkert and D. Miller, “Asynchronous delta sigma modulation,” in *Proceedings of the IREE (Australia)*, vol. 36, pp. 83–88, 1975.

[115] J. Tripathi, “Rectangular-Wave Modulation,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-19, no. 2, pp. 317–320, 1983.

[116] R. Walden, “Analog-to-digital converter survey and analysis,” *IEEE Journal on Selected Areas in Communications*, vol. 17, pp. 539–550, Apr. 1999.

[117] T. Hawkes and P. Simonpieri, “Signal coding using asynchronous delta modulation,” *IEEE Transactions on Communications*, vol. 22, no. 3, pp. 346–348, 1974.

[118] D. Koscielnik and M. Miskowicz, “Asynchronous Sigma-delta Modulator Based On The Charge Pump Integrator,” in *Proceedings of the International Conference on Mixed Design of Integrated Circuits and System*, pp. 234–238, June 2006.

[119] B. Choubey and S. Collins, “Models for Pixels With Wide-Dynamic-Range Combined Linear and Logarithmic Response,” *IEEE Sensors Journal*, vol. 7, pp. 1066–1072, July 2007.

[120] T. J. Koickal, A. Hamilton, T. C. Pearce, and S. L. Tan, “Analog VLSI design of an adaptive neuromorphic chip for olfactory systems,” in *Proceedings of IEEE International Symposium on Circuits and Systems.*, vol. vol, pp. 4547–4550, 2006.

[121] A. Kramer, “Array-based analog computation,” *IEEE micro*, vol. 16, pp. 20–29, Oct. 1996.

[122] V. Ravinuthula and J. G. Harris, “Time-based arithmetic using step functions,” in *Proceedings of the International Symposium on Circuits and Systems*, vol. 1, pp. I–305–I–308, May 2004.

[123] A. Murray, “Pulse techniques in neural VLSI: a review,” in *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 5, (San Diego, CA), pp. 2204–2207, IEEE, 2002.

[124] M. Nagata, “PWM signal processing architecture for intelligent systems,” *Computers & Electrical Engineering*, vol. 23, pp. 393–405, Nov. 1997.

[125] A. Iwata and M. Nagata, “A concept of analog-digital merged circuit architecture for future VLSI’s,” *Analog Integrated Circuits and Signal Processing*, vol. 11, pp. 83–96, Oct. 1996.

[126] J. Caves, S. Rosenbaum, L. Sellars, C. Chan, and J. Terry, “A PCM voice codec with on-chip filters,” *IEEE Journal of Solid-State Circuits*, vol. 14, pp. 65–73, Feb. 1979.

[127] R. Sarpeshkar and M. O’Halloran, “Scalable hybrid computation with spikes,” *Neural computation*, vol. 14, pp. 2003–38, Sept. 2002.

[128] B. Brown and H. Card, “Stochastic neural computation. I. Computational elements,” *IEEE Transactions on Computers*, vol. 50, no. 9, pp. 891–905, 2001.

[129] R. Hughes, *Logarithmic amplification: with application to radar and EW*. Norwood, MA, USA: Artech House, Inc., 1986.

[130] M. Kurchuk and Y. Tsividis, "Signal-dependent variable-resolution quantization for continuous-time digital signal processing," in *IEEE International Symposium on Circuits and Systems*, (Taipei), pp. 1109–1112, IEEE, May 2009.

[131] W. Tang and E. Culurciello, "A pulse-based amplifier and data converter for biopotentials," in *IEEE International Symposium on Circuits and Systems*, pp. 337–340, IEEE, May 2009.

[132] S. Wilson, *Digital modulation and coding*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1995.

[133] L. C. Gouveia, T. J. Koickal, and A. Hamilton, "An Asynchronous Spike Event Coding Scheme for Programmable Analog Arrays," *IEEE Transactions on Circuits and Systems I: Regular Papers*, pp. 1364–1367, May 2010.

[134] L. C. Gouveia, T. J. Koickal, and A. Hamilton, "A CMOS implementation of a spike event coding scheme for analog arrays," in *IEEE International Symposium on Circuits and Systems*, (Taipei), pp. 149–152, IEEE, May 2009.

[135] L. C. Gouveia, T. J. Koickal, and A. Hamilton, "Computation in communication: Spike event coding for programmable analog arrays," in *IEEE International Symposium on Circuits and Systems*, (Paris), pp. 857–860, IEEE, May 2010.

[136] R. J. Baker, H. W. Li, and D. E. Boyce, *CMOS Circuit Design, Layout, and Simulation*. New York: Wiley-IEEE Press, 2nd ed., 1997.

[137] A. A. Fayed and M. Ismail, "A High Speed, Low Voltage CMOS Offset Comparator," *Analog Integrated Circuits and Signal Processing*, vol. 36, no. 3, pp. 267–272, 2003.

[138] D. A. Yaklin, "Offset comparator with common mode voltage stability," Sept. 1996.

[139] M. Bazes, "Two novel fully complementary self-biased CMOS differential amplifiers," *IEEE Journal of Solid-State Circuits*, vol. 26, no. 2, pp. 165–168, 1991.

[140] J.-B. Shyu, G. Temes, and F. Krummenacher, "Random error effects in matched MOS capacitors and current sources," *IEEE Journal of Solid-State Circuits*, vol. 19, pp. 948–956, Dec. 1984.

[141] M. Kurchuk and Y. Tsividis, "Energy-efficient asynchronous delay element with wide controllability," in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, pp. 3837–3840, IEEE, 2010.

[142] A. Martin, "Translating concurrent programs into VLSI chips," in *PARLE '92 Parallel Architectures and Languages Europe* (D. Etiemble and J.-C. Syre, eds.), vol. 605, (Paris), pp. 513–532–532, Springer Berlin / Heidelberg, 1992.

[143] J. Wikner and N. Tan, "Modeling of CMOS digital-to-analog converters for telecommunication," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 46, pp. 489–499, May 1999.

[144] H. Pan, "Method and system for a glitch-free differential current steering switch circuit for high speed, high resolution digital-to-analog conversion," June 2005.

[145] C. Taillefer, “Current mirror compensation for transistor mismatch,” in *IEEE International Symposium on Circuits and Systems*, vol. 2, pp. 509–512, Presses Polytech. Univ. Romandes, 2000.

[146] J. A. Lenero-Bardallo, T. Serrano-Gotarredona, and B. Linares-Barranco, “A Calibration Technique for Very Low Current and Compact Tunable Neuromorphic Cells: Application to 5-bit 20-nA DACs,” *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 55, pp. 522–526, June 2008.

[147] T. J. Koickal, R. Latif, L. C. Gouveia, E. Mastropaolo, S. Wang, A. Hamilton, R. Cheung, M. Newton, and L. Smith, “Design of a Spike Event Coded RGT Microphone for Neuromorphic Auditory Systems,” *Proceedings of IEEE International Symposium on Circuits and Systems.*, 2011.

[148] T. J. Koickal, L. C. Gouveia, and A. Hamilton, “Bio-inspired Event Coded Configurable Analog Circuit Block,” *Evolvable Systems: From Biology to Hardware*, vol. 5216, pp. 285–295, 2008.

[149] T. Green and B. Williams, “Spectra of Delta-Sigma Modulated Inverters : An Analytical Treatment,” *IEEE Transactions on Power Electronics*, vol. 7, no. 4, pp. 644–654, 1992.

[150] S. Lin and D. Costello, *Error control coding: fundamentals and applications*, vol. 326. Englewood Cliffs, NJ: Prentice-Hall, Inc., Apr. 1983.

[151] R. Gregorian, K. Martin, and G. Temes, “Switched-capacitor circuit design,” *Proceedings of the IEEE*, vol. 71, no. 8, pp. 941–966, 1983.

[152] U. Seng-Pan, R. Martins, and J. Franca, “Highly accurate mismatch-free SC delay circuits with reduced finite gain and offset sensitivity,” in *Proceedings of the 1999 IEEE International Symposium on Circuits and Systems VLSI (Cat. No.99CH36349)*, pp. 57–60, IEEE, 1999.

[153] W.-H. Ki and G. Temes, “Offset-compensated switched-capacitor integrators,” in *IEEE International Symposium on Circuits and Systems*, (New Orleans), pp. 2829–2832, IEEE, 1990.

[154] C. Enz and G. Temes, “Circuit techniques for reducing the effects of op-amp imperfections: autozeroing, correlated double sampling, and chopper stabilization,” *Proceedings of the IEEE*, vol. 84, no. 11, pp. 1584–1614, 1996.

[155] K. Martin and A. S. Sedra, “Strays-insensitive switched-capacitor filters based on bilinear Z-transform,” *Electronics Letters*, vol. 15, no. 13, p. 365, 1979.

[156] K. Nagaraj, J. Vlach, T. Viswanathan, and K. Singhal, “Switched-capacitor integrator with reduced sensitivity to amplifier gain,” *Electronics Letters*, vol. 22, no. 21, p. 1103, 1986.