DISS. ETH NO. 24905

Event-Based CMOS Circuits for a Class of Belief-Propagation Models

A thesis submitted to attain the degree of

DOCTOR OF SCIENCES of ETH ZURICH

presented by

Chen-Han Chien

MSc. Electrical Engineering, National Tsing Hua University, Taiwan

born on *27.05.1984*

citizen of Taiwan, Republic of China (R.O.C)

accepted on the recommendation of

PD. Dr. Shih-Chii Liu
Prof. Dr. Tobi Delbruck
Prof. Dr. Hans-Andrea Loeliger

2018

Acknowledgements

Pursuing a PhD in a foreign continent far away from home is a memorable journey I could never imagine before. Along this journey, I am grateful to many people, without whom I cannot reach this far.

First and foremost, I would like to thanks my supervisor **PD Dr. Shih-Chii Liu** for her guidance, countless discussions and valuable suggestions on my research work and academic writing skills. Apart from that, she gave me a timely support when my research was not moving forward. I would also like to thank **Prof. Tobi Delbruck** for his generous help and straightforward feedback on my research project, especially during the circuit test on the USBAERmini2 board. I would also like to thank **Prof. Hsin Chen** for introducing me to the Institute of Neuroinformatics. Without his recommendation, this adventure would have never started.

I'm very grateful to **Minhao Yang**, with whom I had endless discussions about circuits and have gained a much better understanding of many circuits thanks to his profound technical knowledge. I also enjoy all the lunch time discussions with him about topics in the life. I would also like to thank **Adrain Huber** for the feedback and the translation of my thesis. My gratitude also goes to the people working in iniLabs for the consistent technical support. **Chenghan Li** introduced me to the DVS and APS technologies and provided me helpful feedback of my thesis. **Luca Longinotti** developed the firmware of the motherboard and **Vicente Villanueva** provided guidance on the PCB layout.

Last but not least, my parents are always my strongest support. I am grateful to my parents, **Chun-Hsun Chien** (簡俊勳) and **Su-Mei Kuo** (郭素美), for their infinite and eternal support. I want to express my heartfelt gratitude to my beloved wife **Ho-Hui Wang** (王荷惠). Your unconditional love and company in this foreign country make me never alone along this journey. You make me feel at home in a foreign continent. I really enjoy the time in these years with your company, for which I am endlessly grateful!

Abstract

Abstract Bayesian networks are often used to describe how brains can perform inference. Methods of transforming these abstract models to spiking neural networks that can perform inference are still scarce. A recently proposed model called the event-based Belief-Propagation (BP) model shows how inference can be carried out by using the distribution of interspike intervals in spike trains as the messages. Because the simulation times of a factor graph that uses this model can be very long, this thesis proposes an analog Very-Large-Scale Integration (aVLSI) version of this model as one method of speeding up the computation times. The electronic model will be a useful addition to the neuromorphic effort in building spiking neural network systems.

This thesis describes one hardware implementation of this event-based BP model, which uses both a Field-Programmable Gate Array (FPGA) and a mixed analog-digital Application-Specific Integrated Circuit (ASIC) chip developed in a 0.35um CMOS process technology. It describes the challenges in implementing the various circuit blocks of this stochastic model which includes the critical hazard function needed for the neuron to generate stochastic spikes following a defined probability distribution. Many of these circuit blocks did not exist in any form at the start of the thesis because most of the focus in the neuromorphic community is on spiking neural network chips that do not include a stochastic component. Therefore, this thesis presents possible solutions for implementing the event-based stochastic model in hardware.

The hardware system developed in this work is based on an architecture of the event-based BP model that is partitioned into a Landscape Sampling (LS) block and a Random Sampling (RS) block. The input spike trains carrying the BP messages are processed by the LS block that implements the constraint function of a defined factor node. The LS block outputs a message-combined probability distribution that is used by the RS block to produce the stochastic output spikes using the implemented hazard function.

The thesis considers the practical challenges of mapping the theoretical model to aVLSI circuits, the possible methods for generating on-chip noise sources, and the subsequent partitioning of the hardware system into an FPGA and an ASIC. The factor graphs constructed by the event-based BP model under the constraints of the hardware are validated through simulations and then applied in two tasks 1) object tracking using an event-based Kalman filter and 2) data reconstruction using the event-based Continuous Restricted Boltzmann Machine (CRBM). These applications are examples of possible applications of the hardware system. The thesis shows the capability of the final hardware system in implementing graphs with arbitrary variable distributions for its inputs and using constraint functions such as “plus” and “equality”. Measured results show that the BP hardware consumes 6.32 mW of power with 0.046 mW of power per RS channel on the ASIC.

Zusammenfassung

Abstrakte Bayessche Netze werden häufig verwendet, um Inferenz in Gehirnen zu erklären. Methoden, die solche abstrakten Modelle in gepulste neuronale Netze übersetzen, sind jedoch noch selten anzutreffen. Ein kürzlich vorgeschlagenes Modell, welches unter dem Namen ereignisbasiertes Belief-Propagation (BP) bekannt ist, zeigt auf, wie Inferenz erfolgen kann, wobei die Zeitintervall-Verteilung zwischen konsekutiven Pulsen die zu übertragende Nachricht darstellt. Da Faktorgraphen, die solche Modelle verwenden, eine lange Simulationszeit aufweisen, wird in dieser Arbeit eine analoge Very-Large-Scale Integration (VLSI) Variante dieses Modells eingeführt, um die Berechnungszeit zu verkürzen. Das elektronische Modell ist eine nützliche Ergänzung zu neuromorphen Ansätzen, gepulste neuronale Netze zu entwickeln.

Diese Arbeit beschreibt eine Hardware-Implementierung dieses ereignisbasierten BP-Modells, wobei sowohl ein Field-Programmable Gate Array (FPGA) als auch ein Mixed Analog-Digital Application-Specific Integrated Circuit (ASIC) Chip verwendet werden; der ASIC Chip ist für die 0.35um CMOS Prozesstechnologie entworfen. Die Arbeit beschreibt weiterhin die Schwierigkeiten, die bei der Implementierung der verschiedenen Schaltkreiskomponenten dieses stochastischen Modells entstanden; dies beinhaltet die wichtige Ausfallrate, die vom Neuron verwendet wird, um stochastische Pulse zu generieren, die einer definierten Wahrscheinlichkeitsverteilung folgen. Viele dieser Schaltkreiskomponenten existierten zu Beginn dieser Arbeit nicht, da ein Schwerpunkt innerhalb der neuromorphen Entwicklergemeinschaft darin liegt, gepulste neuronale Netze zu entwickeln, welche keine stochastischen Komponenten enthalten. Diese Arbeit zeigt daher Möglichkeiten auf, um ereignisbasierte stochastische Modelle in Hardware zu implementieren.

Das im Rahmen dieser Arbeit entwickelte Hardware-System basiert auf einer Architektur des ereignisbasierten BP-Modells, welche in einen Landscape Sampling (LS) Block und einen Random Sampling (RS) Block zerlegt ist. Die Eingangspulszüge, die die BP-Nachrichten tragen, werden vom LS-Block verarbeitet, welcher die

Nebenbedingungsfunktion eines definierten Faktorknotens implementiert. Der LS-Block gibt eine Wahrscheinlichkeitsverteilung aus, die den verknüpften Eingangsnachrichten in einen Knoten entspricht, und die im RS-Block dazu verwendet wird, um stochastische Ausgangspulse mittels der implementierten Ausfallrate zu erzeugen.

Diese Arbeit untersucht die praktischen Herausforderungen, die bei der Abbildung des theoretischen Modells auf einen aVLSI-Schaltkreis entstehen, die Möglichkeiten, um On-Chip Rauschquellen zu generieren, sowie die nachfolgende Zerlegung des Hardware-Systems in eine FPGA- und eine ASIC-Komponente. Die Faktorgraphen, welche im Rahmen des ereignisbasierten BP-Modells entwickelt werden, und die Hardware-Beschränkungen berücksichtigen, werden mittels Simulationen validiert. Sodann werden sie auf zwei Aufgaben angewendet: 1) Objektverfolgung mittels eines ereignisbasierten Kalmanfilters; 2) Datenrekonstruktion mittels einer ereignisbasierten Continuous-Restricted Boltzmann-Maschine (CRBM). Diese Anwendungen sind Beispiele möglicher Applikationen des Hardware-Systems. Diese Arbeit zeigt die Fähigkeit des entwickelten Hardware-Systems auf, Graphen mit beliebigen Eingangs-Zufallsvariablen-Wahrscheinlichkeitsverteilungen zu simulieren, die Nebenbedingungsfunktionen wie "Plus" und "Gleichheit" benutzen. Messungen ergeben, dass die BP-Hardware 6.32 mW Leistung aufnimmt, wobei 0.046 mW Leistung pro RS-Kanal des ASICs aufgenommen werden.

Contents

List of Figures

List of Tables

Chapter 1 Introduction

In parallel with neural network modeling [1]–[4] efforts in the neuroscience field to understand how brains compute, hardware engineers are developing dedicated or general-purpose silicon neuron processors that allows these models to be computed in parallel and in some cases with reduced power efficiency than running on the computer. Stochastic models are promising candidates for explaining how brains perform inference [5]–[13]. However, the hardware implementations of stochastic models of brain computation are still relatively scarce. Among them, the recently proposed event-based belief-propagation (BP) model uses biologically plausible signals (spikes) and employ the local and parallel computation seen in brains. This thesis revolved around the work in building a hardware system based on a class of BP models aimed at speeding up the simulation time. Because of the use of spikes in the model, the system can be interfaced with the event-based vision [14], [15] and cochlea [16], [17] sensors being developed in the neuromorphic community.

1.1 Probability Inference in the Brain

Various studies [5]–[13] have reported experiments that show that the brain processes information in a Bayesian way. The inference tasks in these experiments include the cue combination task [5]–[8] and computing the a posteriori information from the prior information and sensory input [9]–[13]. To give an example of how humans perform inference, if a man wants to travel during the Christmas holidays and wants to leave by sometime and has to be at his family home by 12 midnight, he makes an estimation based on the information at hand. He can obtain the current traffic news from the TV or radio, or he can use the information about the traffic situation from the past two days and from the same time period in the previous year. He can also use the current weather condition in his estimation. The information obtained at hand will help him make a decision that will get him to the place in time and with the least amount of hours on the highway. This example illustrates that the information processed in the brain can be seen as a form of inference.

Many researchers have worked on Artificial Neural Network (ANN) models that mimic the architecture of the neural brain in solving various problems. A well-known model, for example, in machine learning is the Restricted Boltzmann Machine (RBM) [2], [18] which is used for a classification task involving handwritten digits. This stochastic model can learn a probability distribution of its inputs. An RBM has a two-layered fully connected architecture (one visible and one hidden) with symmetric weights and no interconnection within a layer. Since the neurons are not connected within a layer, the output of each neuron is conditionally independent of one another for a given input. The weights are updated using the training algorithm called Contrastive Divergence (CD) [19]. One extension of the RBM is the Deep Belief Network (DBN) [2] that consists of several stacks of RBMs whose outputs serve as the inputs of the next layer. The RBM and DBN are suitable for learning the probabilities of the input, either in a supervised or unsupervised manner. These networks are widely used in speech recognition [20], [21], and feature extraction and classification with unlabeled data [22]–[24]. Another variant of the RBM is the Continuous RBM (CRBM) [25]. Compared to the binary output of RBM, this model directly uses the analogue values of the output of the sigmoid as the input to the next layer. The CRBM is suitable for modelling continuous asymmetric data [25], [26]. In addition, in some studies [27], [28], the RBM in a DBN is replaced by a CRBM if continuous-valued inputs or real-valued neurons are required.

Spiking Neural Networks (SNNs), the next generation of the ANNs, have a bio-inspired neural network structure that is closer to the architecture of spiking neurons in the brain. In a SNN, neurons communicate by transmitting spikes (or events) to each other [29]. Variants of SNNs are constructed based on different neuron models. Some neuron models are based on biophysical and biochemical experiments, such as the Hodgkin-Huxley model [4] that describes the dynamics of the ion channels and their influence on the membrane potential of the neuron. The Hodgkin-Huxley equation takes into account the membrane capacitance, ion currents, conductances, and channel efficacies in describing the neuron responses to its inputs. Even if the Hodgkin-Huxley model can successfully explain neuron's response such as the spiking properties under different stimuli, its complexity makes this model difficult to scale up to a large network. The Izhikevich model [30], a simplified spiking model, is capable of producing several spiking patterns, e.g. regular spiking, fast spiking and bursting, by using only four parameters. Another simplified model called the Leaky Integrate-and-Fire (LIF) model consists of simply a capacitor, a resistor and a threshold detection. The membrane potential is described by the voltage output of the resistive-capacitive circuit. Once the membrane potential crosses threshold, a spike is triggered and the voltage is reset to a resting potential. The form of the spike

can reduced to a digital pulse. In a LIF, the neuron firing depends on the differential equations governing the change in the membrane potential dependent on the neuron parameters and its input. Instead of a voltage-dependent model, another model called the Spike Response Model (SRM) [29], [31] uses the timing of the input spikes since the last output spike to determine the change in the membrane potential. Once the membrane potential crosses threshold, a next output spike is triggered. In the SRM, the spike timing matters the most while the form of the spike does not carry information.

In recent years, studies show that SNNs can perform well in various tasks, e.g. classification [32]–[35], recognition [36], [37], reconstruction [1] or a decoder for brain-machine interfaces [38]. Some studies also demonstrated that ANNs can be converted to SNNs in the deep learning field. [39] demonstrated that a converted spiking DBN can achieve the classification accuracy of > 94% on the MNIST dataset consisting of 70,000 handwritten digits [40]. The accuracy of the event-driven DBN is only 1% lower than that of the time-stepped DBN. This network has been trained using CD to fuse in real-time, spiking inputs from Dynamic Vison Sensor (DVS) [14] and AER-EAR silicon cochlea [16] in a multi-sensory classification task. [41] presents an event-driven approach of CD that allows online training of the RBM. The classification accuracy on the MNIST by using the event-driven CD (91.9%) is close to the accuracy by using the standard CD (92.6%). In [42], a converted spike-based CNN is implemented for object recognition. Compared to the original CNN, this study shows that the spike-based CNN is two orders of magnitude more energy-efficient with little loss in accuracy.

How neurons transmit information in spike trains is not yet fully understood. The widely used coding schemes are either rate or temporal codes [29]. In rate coding, the information is encoded in the average mean firing rate within a time window. It is successfully used to explain the experimental results on motor systems such as the stretch receptor in a muscle spindle [43]. Some SNN models [1], [37], [39], [41], [44] use rate coding to pass information. However, if neurons in the brain really adopt this coding scheme, spike timing plays no role in spike trains and the various activities in different trials can only be considered as noise. Ideally, the mean firing rate can be sufficiently measured in two consecutive spikes if the period of firing spikes is constant. Due to noise, the mean firing rate has to be averaged within a time window. However, rate coding hardly explains some behaviors such as human ability to recognize images in a few hundred milliseconds [45], [46]. Some studies [47] argue that rate coding is still plausible if a population of neurons is used. It allows generating sufficient spikes in a short period so that neurons could respond quickly.

Another assumption is that the information is encoded in spike timings rather than the rate. In temporal coding, the information is contained in the exact arrival time or the first spike timing since the last spike. Many studies in biological experiments [48]–[51] show that temporal coding can explain the quick responses of neurons to stimuli. Some models are built based on temporal coding [32], [34]–[36].

A class of temporal coding uses InterSpike Intervals (ISIs) to encode information. Some biological studies show that neurons could pass information using spikes preceded by ISIs [52]–[54]. These studies evoke the question of what is the algorithmic meaning of the ISIs and how they play a role during the information processing in neurons. In [55], Steimer et al. proposed an event-based Belief-Propagation (BP) model that uses ISIs in spike trains to pass information among neurons. In this model, a spike is treated as a random sample, whose numerical value is given by the spikes preceding ISI. Each spike train is assumed to follow a renewal processes and hence the sequences corresponds to a sequence of independent random numbers where each spike's label corresponds to an ISI random number as shown in Figure 2.3. This model can be interpreted as a class of graphical models, which not only are a method of presenting variables' dependencies but facilitate the development of the algorithms for probability inference. Graphical models implement the functionality of a system with multiple input variables where the network function can be decomposed into a composition of functions on smaller subset of variables. The Forney Factor Graph (FFG) [56], [57] is one instantiation of graphical models where the nodes of the graph represent the functions of subsets of variables; and messages are transferred between the nodes using methods such as BP [57], [58]. The message-passing scheme in the algorithm of a BP model employs local and parallel computation, which seems biologically plausible because a global observation is not needed. Steimer's model, a variant of a FFG, is an event-based BP model that uses a temporal coding scheme based on ISIs in spike trains to communicate messages between the nodes of the graph.

Research on SNN models is getting popular. It is not only because they are more brain-inspired but because they provide an efficient transmission in terms of the power dissipation and fault tolerance. In the simplified models such as LIF model, spikes (or events) interpreted as digital pulses are transmitted in an asynchronous fashion. That is, the outputs of neurons are silent if the input stimuli is not strong enough to trigger a spike. In addition, a digital pulse can be transmitted a longer distance better than an analog signal. It is especially useful when the voltage headroom is decreasing along with the downsizing of transistors. These properties allow to explore an avenue to feasible hardware implementations, e.g. silicon-based

neuromorphic substrate [59], of large scale neural networks. The author describes next the work on the hardware implementations of ANNs and SNNs.

1.2 Hardware Implementation of Artificial Neural Network

Different from CPUs with a von Neumann structure [60], the massively parallel operation of ANNs' and SNNs' hardware with a simple processing unit is able to solve tasks more efficiently in real-time processing. In [61], [62], the nose-on-a-chip sensor with CRBM allows diagnosing ventilator-associated pneumonia rapidly before going to a doctor for further diagnosis. The functionality of the chip was verified in clinical trials. The accuracy reaches 95.73% on 74 samples as experimental group and 43 samples as control group with 1.27 mW of power. A sparse SNN hardware [63] with 256 neurons is capable of learning and reconstructing images at high speed (140 Mpixel/s throughput) and low power (6.67 mW). In [64], by using the neural engineering framework, the system can achieve a pattern recognition up to 96% on the MNIST dataset. In [65], the CNNs simulated on the computer, Field-Programmable Gate-Array (FPGA) implementation and Application Specific Integrated Circuit (ASIC) are compared for the performance on the number of frames processed per second (fps). It shows that with the same image size and the same filter size, the performance (fps) of the ASIC CNN is better than the FPGA and the FPGA is better than the computer. Minitaur [66], an event-driven neural network hardware accelerator implemented on FPGA, performs 18.73 million postsynaptic updates per second consuming just 1.5 W of power and reaches 92% accuracy on the MNIST dataset and 71% accuracy on the 20 newsgroups classification data set. The system also demonstrates a robustness to noise and maintains a 70% accuracy even when the input contains 80% noise. In [67], an ASIC implementation of a spike-based learning algorithm with 16 neurons and 128 synapses per neuron is capable to classify complex patterns of mean firing rates in real time.

In addition, several groups have worked on large-scale general-purpose spiking neuromorphic hardware. Neurogrid [68] integrates axons, synapses, and dendritic trees in an analog manner within 16 Neurocores, each of which consists 256×256 silicon neuron array. This board containing one million neurons allows for a complexity of neural computation with only 3 W of power. SpiNNaker [69] is a massively parallel ARM processor based system, where each board contains 48 nodes and each node has an 18 ARM processor, to provide a flexible simulator. ROLLS [70] using long-term and short-term plasticity synapse with 256 neurons and 128K analog synapses can perform simple classification tasks after training, such as the

classification of car and motorbike by using the average firing rate of the output neurons. TrueNorth [71] with 1 million spiking digital neurons and 256 million digital synapses can correctly classify moving pedestrians, cyclists and cars in real time. A spiking RBM has been mapped on the neuromorphic TrueNorth system by using a noisy threshold model to implement the Gibbs sampler on the digital neurons [72]. There are other ways of performing approximate inference in graphical models through spiking neurons. One proposed method is that of neural sampling [73]–[76] in a graph with binary nodes and where the nodes are represented by spiking neurons. This form of sampling belongs to the Markov chain Monte Carlo (MCMC) technique and is similar to Gibbs sampling. This sampling scheme has been demonstrated on the SpiNNaker hardware system [77], [78]. These general-purpose large-scale neuromorphic hardware provide a platform for various models facilitating neural computations.

There are also other designs that implement graphical models. In [79], Loeliger et al. demonstrated an analog circuit that implements the Sum-Product Rule (SPR) where probabilities are represented by currents. The product term in the SPR is achieved by using a current multiplier with the structure of a Gilbert multiplier and operating in subthreshold regime or by using bipolar transistors. The summing term is achieved by summing currents together. A Gilbert multiplier can be regarded as a factor node whose constraint function is defined by the input connections of this circuit. [79] demonstrates two example factor nodes, which are soft exclusive-or gate and component-wise product, with two inputs distribution and one output distribution represented by currents. The distributions used in this analog circuit are not limited to binary distributions. A fundamental circuit architecture was presented in [79], showing the concept of how a network can be built to process a multi-value probability distribution (i.e. probability mass function). Another work [80] implemented the min-sum (or max-product) algorithm in analog circuits, which also use currents to represent distributions. By using min-sum algorithm, [80] claims that standard CMOS technology (instead of BiCMOS) and the conventional biasing method (i.e. transistors operated above threshold) can be used for realizing the analog circuits so that the cost during manufacturing and the mismatch between blocks can be reduced.

1.3 Thesis Contribution and Organization

As mentioned in the sections above, the increasing availability of different neural network hardware platforms that are implemented through either custom mixed-mode analog/digital or digital Very-Large-Scale Integration (VLSI) spiking neuron

arrays or on FPGA have allowed the validation of various neuroscience and machine learning spiking models for practical applications. The use of stochastic models (mainly RBM and its variants are applied) on neuromorphic spiking platforms is still relatively scarce although stochasticity by itself could be useful, e.g. to decorrelate the firing of neurons in a population [81]. **This thesis sets out to examine the feasibility of realizing event-based probabilistic computation in hardware through spiking events.** The work described in this thesis is based on Steimer's model [55] that provides a link between both stochastic neural networks and FFGs to spiking network computation. Because this stochastic model cannot be easily implemented on the currently available spiking network hardware platforms, an explicit implementation of the event-based message passing scheme is presented in this thesis and was published in [82]. This analog VLSI (aVLSI) implementation is based on direct correspondence between the fundamental equations from renewal theory and the physical behavior of aVLSI circuit elements. This circuit can generate sequences of arbitrarily distributed random numbers that are confined to the positive real axis. In this thesis, the VLSI message-passing circuit in [82] is extended to an ASIC chip with an array of 16 channels that produce output messages. The calculations of the analog messages carried by the ISIs of the input spike trains and the output of the factor functions are carried out using an FPGA for flexibility in constructing graphs with different factor functions. The circuit details of the ASIC chip in a 2-poly-4-metal 0.35um CMOS process are described in the thesis. Model simulations in Matlab and measurements from the combined ASIC + FPGA system used in the construction of example factor graphs are also presented in the thesis. The thesis is organized as follows.

Chapter 2 first introduces the Forney Factor Graph (FFG) and some notions related to this work. The chapter also describes the event-based belief propagation model associated with the renewal theory, the discrete-time approximation and the event-driven message-passing algorithm. It also presents the architecture of the event-based factor node which is composed of two blocks, the Landscape Sampling (LS) block and the Random Sampling (RS) block. The LS block receives the input events and the constraint function is defined here. The RS block is used for generating the output spikes based on the message-combined probability distribution. The hardware implementation is based on the required two blocks in this architecture.

Chapter 3 presents the model simulation results. Before realizing the event-based BP model in hardware, the model validity is verified by building up factor graphs with the architecture described in Chapter 2. The simulation is first don on

the RS block to demonstrate the feasibility of the discrete-time approximation. Different equations used to compute the hazard are simulated. Among them, the one suitable for the hardware can be determined by comparing their output ISI distributions to the input probability distribution. Then, a single factor node combining the LS and the RS blocks is simulated with several constraint functions defined in this factor node. Finally, the chapter presents two applications: 1) object tracking of a falling tennis ball by an event-based Kalman filter and 2) data reconstruction by an event-based CRBM.

Chapter 4 elaborates on the hardware implementation of the event-based BP model including the system architecture and circuit blocks. It describes in detail the LS implemented on the FPGA and a random number generator circuit needed for producing stochastic output events in this model. It also presents two variants of the RS circuits. The measurement results on the ASIC and the entire system are described here along with the detailed circuit descriptions. This chapter also gives examples of networks using several factor nodes implemented on this hardware system.

Chapter 5 gives a conclusion of this thesis work and an outlook of possible future directions on this work.

Chapter 2 Structure of Belief-Propagation Model

This chapter first mentions some basic notions of the Forney Factor Graph (FFG) that are useful for understanding the event-based Belief-Propagation (BP) model and the work described in the later chapters. More details of FFGs can be found in [57], [58]. Then, the event-based BP model proposed by Steimer et al. [55] is introduced. The Matlab model and hardware system described in this thesis are both based on the architecture introduced here. A part of the text in this chapter comes from the paper (the title is "Hardware Implementation of an Event-Based Message Passing Graphical Model Network") published in the IEEE Transactions on Circuits and Systems I (TCASI) in 2018 [83]. The author has only used the text related to the work contributed by the author in the paper.

2.1 Forney Factor Graph

A FFG is a graph-based representation of a factorized joint probability distribution, such that the nodes of the graph correspond to the nonnegative factors of the factorization. The edges in turn correspond to those variables, on which the factor functions they are connected to, depend on. An example of a factor graph representing the factorized joint probability of the variables described in (2.1), is shown in Figure 2.1. The nodes of the graph correspond to the individual factors (f_1 to f_6) of the factorization in (2.1) and the edges correspond to the variables. It is assumed in the remainder of the thesis that the variables are discrete and therefore only summations, rather than integrations, are needed for marginalization. The marginal probability of each variable can be computed by summing over all variables except for the desired variable. The marginal probability, e.g. of X_3, is shown in (2.2) with a normalizing factor described in (2.3).

$$f\left(x_1,\mathrm{K}\,,x_6\right)=f_1\left(x_1\right)f_2\left(x_2\right)f_3\left(x_1,x_2,x_3\right)f_4\left(x_4\right)f_5\left(x_3,x_4,x_5\right)f_6\left(x_5,x_6\right) \quad (2.1)$$

$$p(x_3) = \sum_{\substack{x_1,\mathrm{K},x_6 \\ except\ x_3}} f(x_1,\mathrm{K},x_6) / Norm \tag{2.2}$$

$$Norm = \sum_{x_1,\mathrm{K},x_6} f(x_1,\mathrm{K},x_6) \tag{2.3}$$

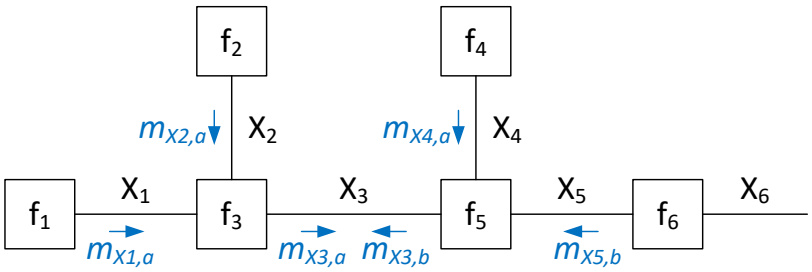


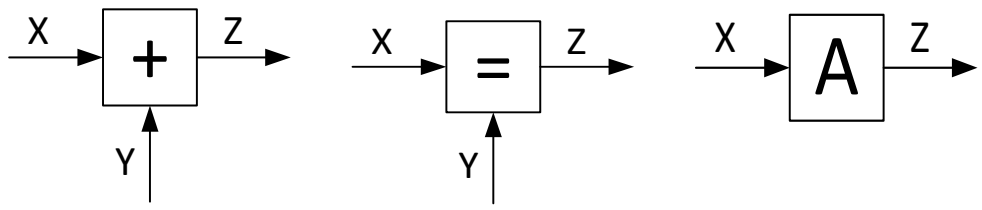
Figure 2.1 A factor graph generated from the factorized joint probability in (2.1).

Using the BP approach, marginalization can be performed in a more efficient way than in (2.2), by exchanging 'messages' between adjacent nodes along the connecting edge (variable). These messages can be interpreted as probability mass functions that depend on the connecting variable and, for the graph in Figure 2.1, are computed following the set of equations in (2.4). As Figure 2.1 shows, a message *m* (blue arrows) which is sent from one factor to one of its neighbors is formed by the product of all input messages into the sending node (except for the message coming in along the same edge as *m*) and the factor function represented by the sending node. The resulting product is then summed across all variables connected to the sending node, except the variable (edge) the output message is passed along. The method of computing output messages in this way is called the sum-product rule (SPR). Note that in (2.4) the output message of each equation is equal to its right side up to a scale factor $Norm_{Xi}$, which is formed by summing along the output variable X_i. The marginal probability of X_3 in Figure 2.1, i.e. $p(x_3)$, is obtained by multiplying the messages from the left and right sides of the edge associated with X_3 as shown in (2.5).

$$
\begin{aligned}
m_{X1,a}(x_1) &\propto f_1(x_1) \\
m_{X2,a}(x_2) &\propto f_2(x_2) \\
m_{X3,a}(x_3) &\propto \sum_{x_1,x_2} f_3(x_1,x_2,x_3)\, m_{X1,a}(x_1)\, m_{X2,a}(x_2) \\
m_{X3,b}(x_3) &\propto \sum_{x_4,x_5} f_5(x_3,x_4,x_5)\, m_{X4,a}(x_4)\, m_{X5,b}(x_5) \\
m_{X4,a}(x_4) &\propto f_4(x_4) \\
m_{X5,b}(x_5) &\propto \sum_{x_6} f_6(x_5,x_6)
\end{aligned}
\tag{2.4}
$$

$$
p(x_3) \propto m_{X3,a}(x_3)\, m_{X3,b}(x_3) \tag{2.5}
$$

Although the belief propagation approach presents an advantage in that the bidirectional messages on all edges are formed by summations across only a subset of all variables and can be computed in parallel, the massive cost involved in computing the SPR is still required. This cost can be reduced by considering only Gaussian distributed variables and restricting the defined constraint functions to a few types, e.g. plus, equality, and gain as shown in [58]. Factor nodes for these functions (see Figure 2.2) are described in (2.6), and the unidirectional output messages, for instance, are computed as shown in (2.7). As a result, the mean and variance of the distributions are the only parameters needed during message passing and the message computation rules can easily be tabulated [57], [58]. This constraint reduces the computational load. However, for messages of arbitrary distributions and for more complex user-defined functions, the full calculation of the SPR is needed.



(a) (b) (c)

Figure 2.2 Symbols of specific factor nodes. The arrow indicates the direction of message passing. The nodes define the (a) plus, (b) equality, and (c) gain constraint functions with variables *X*, *Y* and *Z*.

$$f_{plus}(x,y,z)=\delta\left(z-\left(x+y\right)\right)$$
$$f_{equality}(x,y,z)=\delta(x-y)\delta(y-z) \tag{2.6}$$
$$f_{gain}(x,z)=\delta\left(Ax-z\right)$$

$$m_Z(z)\propto\sum_x\sum_y f_{plus}(x,y,z)\,m_X(x)\,m_Y(y)$$
$$m_Z(z)\propto\sum_x\sum_y f_{equality}(x,y,z)\,m_X(x)\,m_Y(y)=m_X(z)\,m_Y(z) \tag{2.7}$$
$$m_Z(z)\propto\sum_x f_{gain}(x,z)\,m_X(x)=m_X\left(\frac{z}{A}\right)$$

2.2 Event-Based Belief-Propagation Model

This section introduces the main concept of the event-based BP model proposed by Steimer et al. [55]. Instead of representing the SPR as a list of probabilities, [55] showed how analog messages can be represented as a list of InterSpike Intervals (ISIs) (i.e. ISI samples) of any two consecutive events in the input and output spike streams of the factor nodes. This model avoids the normal expensive SPR computation. The SPR summations are solved in an implicit way by means of Monte Carlo sampling. In addition, the factor graph in this formulation is not limited to the use of Gaussian messages. The event-based belief propagation approach bears some similarity to the sequential Monte Carlo sampling method [84], [85], where many particles are used to approximate the sampled input distribution. This message passing formulation is inspired by experimental evidence that shows that populations of neurons can be sensitive to the timing of their inputs and that the input to a neuron can depend on the spike input frequency or the difference between the arrival time of spikes [46], [86]. The mechanism of the event-based BP model in [55] is explained as follows.

2.2.1 Renewal Theory

The message is encoded in the ISIs of the spikes or events as shown in Figure 2.3. Given a probability distribution representing a message, each ISI value is a random sample from this distribution. Within a finite time window, *W*, the statistics (empirical distribution) of the samples approximates the true ISI distribution representing the message.

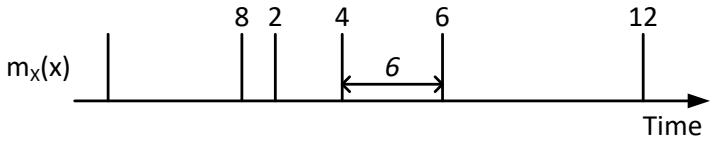


Figure 2.3 Proposed message encoding principle for random number sequence. Each spike of a train is provided with an analog label, whose value corresponds to the length of the ISI preceding the spike, i.e. to the difference in spike time between the considered spike and its predecessor (see numbers above the spikes in arbitrary units). These analog values are therefore samples of the ISI distribution underlying the spike train. The spike train is renewal, i.e. all ISIs are independent samples.

It is a stochastic, event-based process, called renewal process, such that the temporal difference (ISI) between the times of occurrence of two successive events (spikes) follows some given distribution. Importantly, for a process to be renewal, two different ISIs must be statistically independent. In particular, this means that the time of each spike depends only on the time of the latest previous spike and not on the rest of the process' history. In other words, the process is 'renewed' after each spike, thereby explaining the term 'renewal process'.

Besides the ISI distribution $p(t-t_0)$ (where t_0 is the time of the last spike), renewal theory is based on two other fundamental quantities, the survivor function $S(t-t_0)$ and the hazard function $h(t-t_0)$. $S(t-t_0)$ is the process' probability of 'surviving' until time t without firing any further spike, given the last spike has happened at t_0. The equation is shown in (2.8). The hazard $h(t-t_0)$ in turn can be interpreted as a conditional instantaneous firing rate, i.e. the probability of firing within an infinitesimally small interval around t, given that the last spike occurred at t_0. A fundamental result of renewal theory establishes the relation in (2.9) between $p(t-t_0)$, $S(t-t_0)$ and $h(t-t_0)$ [29], [87].

$$S\left(t-t_0\right)=1-\int_{t_0}^{t} p\left(t'-t_0\right)dt' \tag{2.8}$$

$$p\left(t-t_0\right)=S\left(t-t_0\right)\cdot h\left(t-t_0\right) \tag{2.9}$$

Equation (2.9) has a quite straightforward interpretation; Given that the last spike happened at t_0, the probability of an ISI of length $t-t_0$ is equal to the (joint) probability of not firing until t, times the probability of firing at t. To avoid cluttered notation, for the rest of the thesis it is assumed that the last spike has happened at

$t_0 = 0$). Combined with (2.8) and (2.9) this allows for an alternative expression of the hazard function in (2.10). The details of the derivation are given in Appendix A.1.

$$\begin{aligned} h(t) &= \frac{p(t)}{1-\int_0^t p(t')\,dt'} \\ &= p(t)\cdot\exp\left(\int_0^t h(t')\,dt'\right) \end{aligned} \tag{2.10}$$

Equation (2.10) is this continuous recursive form of the hazard function implemented in VLSI circuits, the output of which is then used as an instantaneous firing rate input to a spike-generator circuit.

2.2.2 Discrete-Time Approximation

To generate independent output ISI samples based on the renewal theory, the hazard value, *h(t)*, is used to compare with the samples from a uniform random variable. To realize this random variable on VLSI hardware, the current circuits in the author's best knowledge are clock-driven. Note that it is not claimed that all the random number generators are clock-driven but those with a uniform distribution are so (details in Chapter 5). Therefore, new random numbers are only sampled at finite, discrete time steps, which undermine the continuous time assumption implicit in (2.10). This section presents a condition imposed on the mean time t_{mean} of the input probability distribution *p(t)*, defined in (2.11), and the time step Δt, which is the period of the sampling clock of the random number generator, such that (2.10) is still approximately valid. Here summarize the key properties of the discrete-time approximation Please refer to Appendix A.2 for the detail definitions and descriptions.

$$t_{mean} = E[T] = \int_0^\infty t p(t)\,dt \tag{2.11}$$

In order to produce output spikes that follow a given instantaneous rate profile (e.g. a hazard function), it is necessary to compare the product $h(t)\Delta t$ with the sample "*nx(t)*" of a uniform random variable "*NX*", whose range is between [0,1], on time $t = i\Delta t$, where $i \in N_0 = \{\{0\} \cup N\}$. The probability of generating a spike event is equal to the probability of $nx(i\Delta t)$ being smaller than $h(i\Delta t)\Delta t$, and is hence given in (2.12).

$$\Pr\left(h(i\Delta t)\Delta t > nx(i\Delta t)\right) = h(i\Delta t)\Delta t \tag{2.12}$$

Equation (2.12) is the discrete-time approximation of some instantaneous firing rate, which is given by the hazard function. Using this comparison scheme, the

output ISI distribution $p_{ISI}(n\Delta t)$, where an event happens on $t = n\Delta t$ and no events happen before $t = n\Delta t$, can be written as (2.13). Equation (2.13) is the finite time approximation to (2.10). If Δt is small enough compared to t_{mean}, p_{ISI} will approximate p. In a case where p is a regularly-step-staircase (RSS) probability distribution (see Figure A.1), p_{ISI} is even equal to p for any values of Δt is. For the derivation of (2.13) in different conditions of p, please refer to Appendix A.2.

$$\begin{aligned} p_{ISI}(n\Delta t)\Delta t &= h(n\Delta t)\Delta t \cdot \left(\prod_{i\in\{1,\ldots,n-1\}} \left(1-h(i\Delta t)\Delta t\right) \right) \\ &\approx p(n\Delta t)\Delta t, \text{ if } \Delta t << t_{mean} \end{aligned} \tag{2.13}$$

2.2.3 Message Passing

The section first explains how the message passing using the event-based belied-propagation model where a message is encoded in a number of ISIs in a spike train can approximate the messages computed using the SPR and then gives an architecture of a factor node.

Note that the values of ISIs dealt with in this thesis are in the range of natural number, i.e. the value of each ISI $\in N$, where $N = \{1,2,3,\ldots\}$, as shown in Figure 2.3. Therefore, the probability distribution of a message such as $m_X(x)$ is a probability mass function and the possible values of variable X is natural numbers. The real time scale of an ISI in seconds is represented as t_{ISI} which is equal to the product of the ISI's value and time step Δt, i.e. $t_{ISI} = ISI\Delta t$. The maximum ISI in a spike train is defined as ISI_{max} in natural number and t_{ISImax} in real time scale.

The notion of the expectation of a random variable is introduced first. In (2.14) the expectation of a random variable whose possible values are in nature numbers, i.e. $x \in N$, is defined as the sum of the products of the random variable's all possible values and its probabilities [88]. Alternatively, summing all samples' values and divided by the total number of the samples, named *Norm*, is another solution to find out the expectation if the number of sample approaches to the infinity. Otherwise, the mean calculated using the samples is only an approximation on the expectation with a finite number of samples. Similarly, the expectation of a function of two random variables, whose samples are both natural numbers, is defined in (2.15). If the two random variables are independent, the joint probability distribution $p(x,y)$ can be written as $p(x)\times p(y)$. The expectation of $g(X,Y)$ can also be approximated using the sample pairs (x,y) assuming that the number of sample pairs are sufficient.

$$E(X) = \sum_x xp(x) \approx \frac{\sum_{\text{all samples of x}} x}{Norm} \tag{2.14}$$

$$E\left[g(X,Y)\right] = \sum_{x,y} g(x,y)\,p(x,y) = \sum_x \sum_y g(x,y)\,p(x)\,p(y) \approx \frac{\sum_{\text{all sample pairs of (x,y)}} g(x,y)}{Norm} \tag{2.15}$$

$$m_Z(z) \propto \sum_x \sum_y f(x,y,z)\,m_X(x)\,m_Y(y) \tag{2.16}$$

The form of (2.15) shows a similarity with the general form of the SPR described in (2.16). Given a value of variable *Z*, e.g. *z* = *i*, the three-variable function *f(x,y,z)* is simplified as a two-variable function $f_{z=i}(x,y)$. Computing $m_Z(i)$ now is similar to computing the expectation of function $f_{z=i}(x,y)$. Therefore, $m_Z(i)$ can be approximated using the samples pairs *(x,y)*. By changing the value *i* from 1 to *n* and reusing the samples pairs, an unnormalized message (or histogram) $um_Z(z)$, composed of [$um_Z(1)$,$um_Z(2)$,…, $um_Z(n)$], can be established as shown in (2.17). $um_Z(i)$ can be understood as the value at *i*-th bin of the histogram. After normalization, message m_Z is obtained as shown in (2.19). Because messages m_Z is a probability distribution whose area in total has to be 1, the normalizing term *Norm* should be a sum of um_Z as shown in (2.18). Hence, it is necessary to compute *Norm* as well. Function *F(x,y)*, a sum of function *f(x,y,z)* with respect to variable *Z*, is used to obtain the value of *Norm* from the samples pairs *(x,y)* as shown in (2.18). In addition, um_Z and *Norm* can be computed in parallel by sharing the same sample pairs.

$$\begin{aligned} um_Z(z=1) &= \sum_x \sum_y f_{z=1}(x,y)\,m_X(x)\,m_Y(y) = \sum_{\text{all sample pairs of (x.y)}} f_{z=1}(x,y) \\ um_Z(z=2) &= \sum_x \sum_y f_{z=2}(x,y)\,m_X(x)\,m_Y(y) = \sum_{\text{all sample pairs of (x.y)}} f_{z=2}(x,y) \\ &\text{M} \\ um_Z(z=n) &= \sum_x \sum_y f_{z=n}(x,y)\,m_X(x)\,m_Y(y) = \sum_{\text{all sample pairs of (x.y)}} f_{z=n}(x,y) \end{aligned} \tag{2.17}$$

$$\begin{aligned} Norm &= \sum_{z} um_Z(z) \\ &= \sum_{z}\sum_{x}\sum_{y} f(x,y,z)\, m_X(x)\, m_Y(y) \\ &= \sum_{x}\sum_{y} F(x,y)\, m_X(x)\, m_Y(y) \\ &= \sum_{\text{all sample pairs of (x,y)}} F(x,y) \end{aligned} \tag{2.18}$$

$$m_Z(z) = \frac{um_Z(z)}{Norm} \tag{2.19}$$

Figure 2.4(a) shows an example of a factor node block, where the ISIs in spike trains $spike_X$ and $spike_Y$ encode messages m_X and m_Y. Therefore, samples described above are represented by ISIs and the value *i* from 1 to *n* is represented by the ISI's value from 1 to ISI_{max}. The ISIs from $spike_X$ and $spike_Y$ are paired and sent to the constraint function *f(x,y,z)*. Given some ISI value *i* of variable *Z*, Figure 2.4(b) shows the landscape of function $f_{z=i}(x,y)$ as an example (Note that Figure 2.4(b) is taken from Steimer et al [55]). The value $um_Z(i)$ is computed using function $f_{z=i}(x,y)$ and the sample pairs. Meanwhile, *Norm* is also computed using function *F(x,y)* and the ISI pairs. By changing *i* from 1 to ISI_{max} and divided by *Norm*, the output message $m_Z(z)$ is obtained. Then, a spike train, $spike_Z$, which encodes message m_Z in the ISIs are generated using the discrete time approximation. With the sample-based approach, the event-based BP model avoids using the SPR for computing messages.

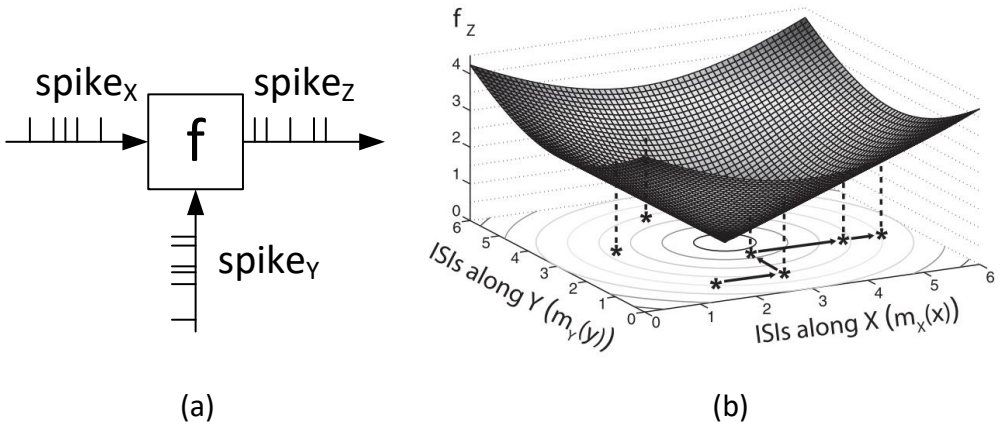


Figure 2.4 Concept of the event-based BP model. (a) Factor node with three edges *X*, *Y*, *Z*, where the messages are passed along the arrows using spike trains $spike_X$, $spike_Y$ and $spike_Z$. (b) Every input pair *(x,y)* samples the function *f* while *Z* = z.

The architecture of a factor node is described as follows. A unidirectional message passing is composed of a Landscape Sampling (LS) block and a Random Sampling (RS) block as shown in Figure 2.5. A complete factor node for this example is made up of three such circuits in order to compute the messages in both directions for all three variables. In the LS block, samples consist of pairs of the most recent input ISIs of variables *X* and *Y*. These samples *(x,y)* are both sent to the factor's function, *f(x,y,z)* and the summation function *F(x,y)* in the LS block. The summation function *F(x,y)* is the sum of *f(x,y,z)* with respect to the output variable, *Z*, and is used as a normalizing term. Once all pairs in a time window, *W*, are computed, the message-combined distribution m_Z is computed from the histogram of *f(x,y,z)* normalized by the value of *F(x,y)*. The message m_Z is then sent to the RS block that generates the output spikes. The hazard function and a uniform random number generator are required in this block.

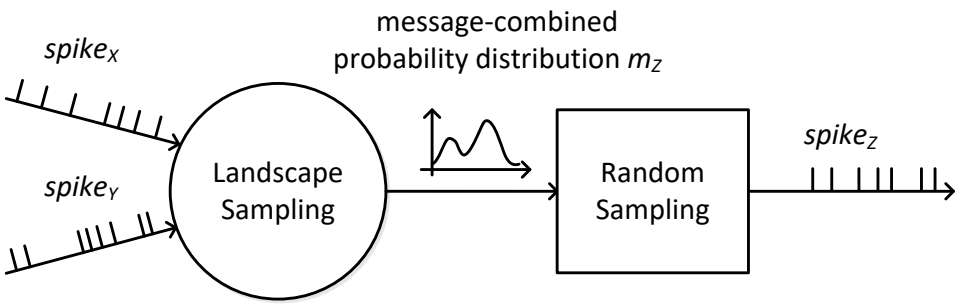


Figure 2.5 Proposed message passing scheme. The factor's function *f* is implemented in a LS block which produces the probability distribution of message m_Z used by the RS block to generate the spiking output.

Chapter 3 Event-Based Belief-Propagation Model Simulation

Before diving into the hardware implementation, the behavior of the event-based Belief-Propagation (BP) model through simulations in Matlab is verified. First, the RS block is simulated to confirm the validity of the discrete-time approximation. Then a single factor node combining the Landscape Sampling (LS) block and the Random Sampling (RS) block is simulated. Lastly, two applications are presented, one is the event-based Kalman filter for object tracking and another is the event-based Continuous Restricted Boltzmann Machine (CRBM).

3.1 Random Sampling Block Validation

As mentioned in Section 2.2.2, the output InterSpike Interval (ISI) distribution p_{ISI} approximates the input probability distribution p_{in} using (2.10) and (2.12). Also, in Appendix A.2, different types of the input probability distribution $p_{in}(t)$ and different methods to compute the hazard $h(t)$ are discussed. In brief, the output ISI distribution p_{ISI} can be equal to p_{in} if $p_{in}(t)$ is a regularly-step-staircase (RSS) probability distribution and the hazard is computed using the original definition (A.5) or the continuous recursive form (A.6). If, on the other hand, $p_{in}(t)$ is not a RSS probability distribution or the discrete recursive form (A.7) is used, p_{ISI} is only an approximation of p_{in} under the condition that time step $\Delta t \ll t_{mean}$, where t_{mean} is defined as the mean time of $p_{in}(t)$ as shown in (2.11). Note that in the following sections, the input probability distribution is represented as p_{in} instead of p used in Chapter 2 and Appendix. Literally, they are identical. The subscript is intended to make the notation easier.

The influence on p_{ISI} using different equations to update hazard is demonstrated through simulation. The three ways to update hazard are the original definition (A.5), the continuous recursive form (A.6) and the discrete recursive form (A.7). First, a discrete-value input signal representing a RSS p_{in} is provided as shown in Figure 3.1(a). It is a triangle waveform from $t = 1\Delta t$ to $17\Delta t$. In the following Matlab simulations, Δt

is defined as 1. The output ISI distribution is obtained by counting the ISIs in the output spike train with time window W = 1,000,000. Since the discrete-time approximation described in Appendix A.2 does not consider the effect of the finite number of samples, enough ISIs (= 70,000) are collected by setting a large W to compute the histograms.

In Figure 3.1(a), p_{ISI1} is the output ISI distribution using the original definition to compute the hazard; p_{ISI2a} and p_{ISI2b} are the output ISI distributions using the discrete recursive form with different updating steps, Δt and $0.1\Delta t$, respectively. The purpose of providing different updating steps is that continuous-time simulations are not possible on the computer so p_{ISI} using the continuous recursive hazard is obtained indirectly by reducing the time step. Note that if there is no spike after sequentially providing the entire p_{in} (which means the hazard is always smaller than the random sample), the RS block is forced to generate a spike at ISI = 25.

From the shape of the output ISI distributions in Figure 3.1(a), it can be easily seen that p_{ISI1} is the most similar to p_{in}. In addition, although both p_{ISI2a} and p_{ISI2b} have distortions (some counts are located at ISI = 25), the latter is more similar to p_{in} than the former. Note that the Kullback-Leibler (KL) will be used to show the similarity of two distributions later. Here the author briefly point out that the difference of p_{ISI} is visible to the eye. The reason for the distortion is explained as follows. In the discrete recursive form, it is assumed that the hazard value stays constant during the time between two consecutive updates, i.e. $t \in [i\Delta t,(i+1)\Delta t]$ or $[0.1i\Delta t,0.1(i+1)\Delta t]$ depending on the updating step, despite the fact that the hazard is continuously changing. Therefore, the hazard computed from the discrete recursive form is smaller than the one from the original definition as shown in Figure 3.1(b). In the beginning, i.e. the first few bins, the difference of the hazard is not obvious. However, the recursive property enlarges the difference, leading to distinct hazard values at the end. It can be seen that h_{ISI1} at ISI = $17\Delta t$ is 1 which guarantees a spike must be produced (the range of the random sample nx is [0,1]) while h_{ISI2a} at ISI = $17\Delta t$ is only 0.64. Therefore, all p_{ISI}(t) in Figure 3.1(a) approximate p_{in}(t) when t is small but some of them, especially p_{ISI2a}, do not approximate well at the end.

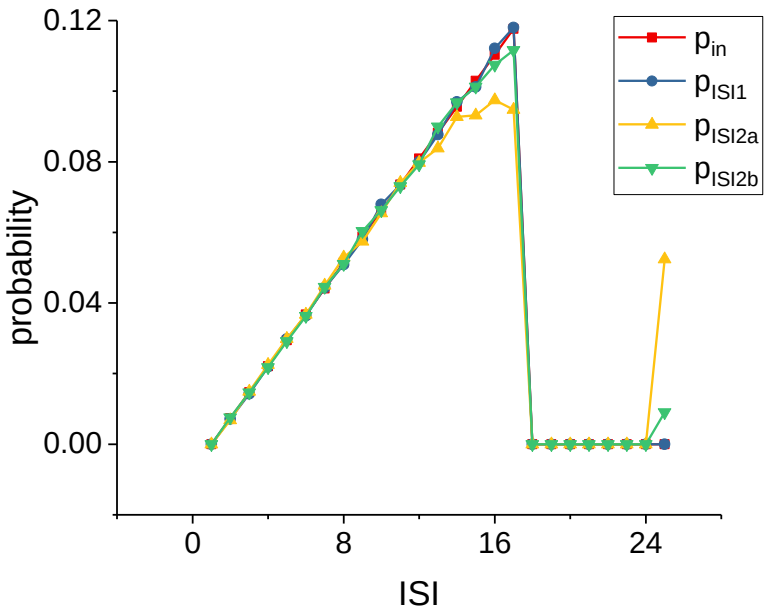
Table 3.1 shows the Kullback-Leibler (KL) divergence [89] in different updating steps. The definition of KL divergence, $D_{KL}(P\|Q)$, is shown in (3.1). It characterizes the disparity between the empirical distribution $Q(i)$ and the theoretical (or ideal) distribution $P(i)$. This disparity is caused by the finite set of samples that are used by $Q(i)$ for a representation of $P(i)$. The Q in this case is represented as p_{ISI} and the P is represented as p_{in}.

$$D_{KL}\left(P \| Q\right) = \sum_{i} P(i) \log \frac{P(i)}{Q(i)} \tag{3.1}$$

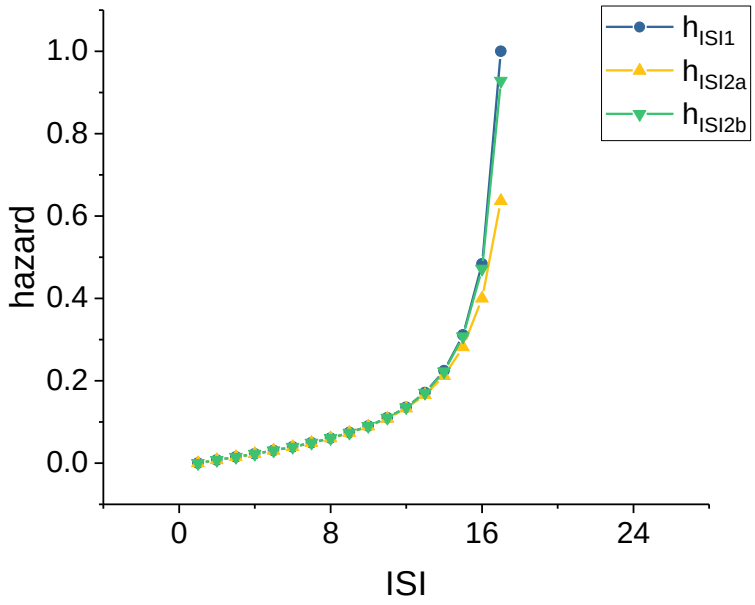
Table 3.1 shows that minimizing the updating step will help decrease the error so that p_{ISI} approximates p_{in}. When minimizing the updating step, the discrete recursive form to compute the hazard approaches to the continuous recursive form. The experimental results indicate that it is better to use either the original definition or the continuous recursive form to update the hazard. If, on the other hand, the discrete recursive form is used, the updating step has to be set as small as possible. As the result, the original definition is used to compute the hazard in the simulations and the continuous recursive form is used for the hardware implementation in Chapter 4.

Output	Hazard Updating Step	KL Divergence
p_{ISI1}	Δt	1.37E-4
p_{ISI2a}	Δt	0.081
p_{ISI2b}	0.1Δt	0.013
p_{ISI2c}	0.01Δt	0.0013
p_{ISI2d}	0.001Δt	3.04E-4
p_{ISI2e}	0.0001Δt	1.54E-4

Table 3.1 KL divergence between the output ISI distribution and the input distribution. (Δt = 1)



(a)



(b)

Figure 3.1 (a) Output ISI distributions over 70,000 samples using different equations to compute the hazard. p_{ISI1} is obtained using the original definition while p_{ISI2a} and p_{ISI2b} are obtained using the discrete recursive form with different updating steps, Δt and $0.1\Delta t$. (b) Corresponding hazards for the three methods. The definitions of the subscripts are the same as in (a).

3.2 Factor Node

The LS and RS blocks are combined to form a unidirectional factor node. The LS takes input spikes and generates a message-combined probability distribution according to the constraint function *f*. This message-combined probability distribution is sent to the RS block which produces the output spikes. This section models the functionality of the event-based BP model using different constraint functions in a factor node and shows an example of a simple network.

The first factor node demonstrated in Figure 3.2(a) is the **equality** constraint node whose function $f_{equality}$ is shown in (2.6). The arrows indicate the directions of the message passing. Because the inputs of the factor node have to be spike streams, two RS blocks in front of the equality constraint node is placed to generate the spike trains carrying the corresponding messages m_X and m_Y. Figure 3.2(b),(c) show the defined probability distributions of the two messages in RS blocks in in red and the output ISI distributions in blue

The output message m_Z of the equality constraint node is shown in Figure 3.2(d). The red curve shows the probability distribution computed from the SPR in (2.7) and the blue curve shows the output ISI distribution by counting the ISIs in the spike train of *Z*. As the result shown in (2.7), the output message is the product of two input messages. Since a triangular-shaped distribution for message m_X and a V-shaped distribution for message m_Y is provided, message m_Z should show a two-bump distribution. As Figure 3.2(d) shows, the output ISI distribution approximates the SPR result.

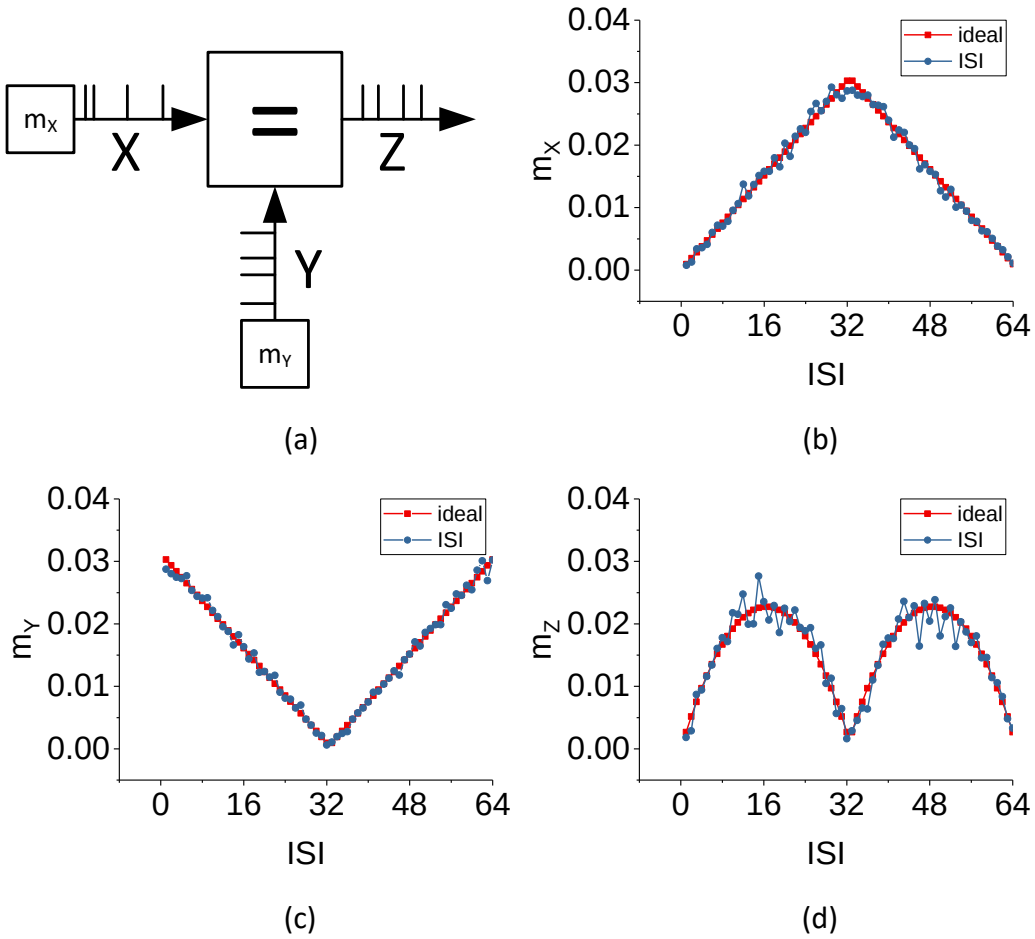


Figure 3.2 (a) Unidirectional factor node with the equality constraint function. The input spike trains are generated from two RS blocks. (b) Distribution of message m_X. The red curve shows the defined probability and the blue curve shows the output ISI distribution from the spike train with 18,000 samples. (c) Distribution of message m_Y (d) Distribution of message m_Z

The second factor node is the **plus** constraint function f_{plus} shown in Figure 3.3(a). Both message m_X and m_Y are defined as uniform distributions ranging from [1,32] as shown in Figure 3.3(b) and (c). In (2.7), the output distribution should be the result of the convolution of two inputs. Therefore, message m_Z should be a triangular-shaped distribution. As Figure 3.3(d) shows, the output ISI distribution of message m_Z (the blue curve) approximates the SPR result (the red curve).

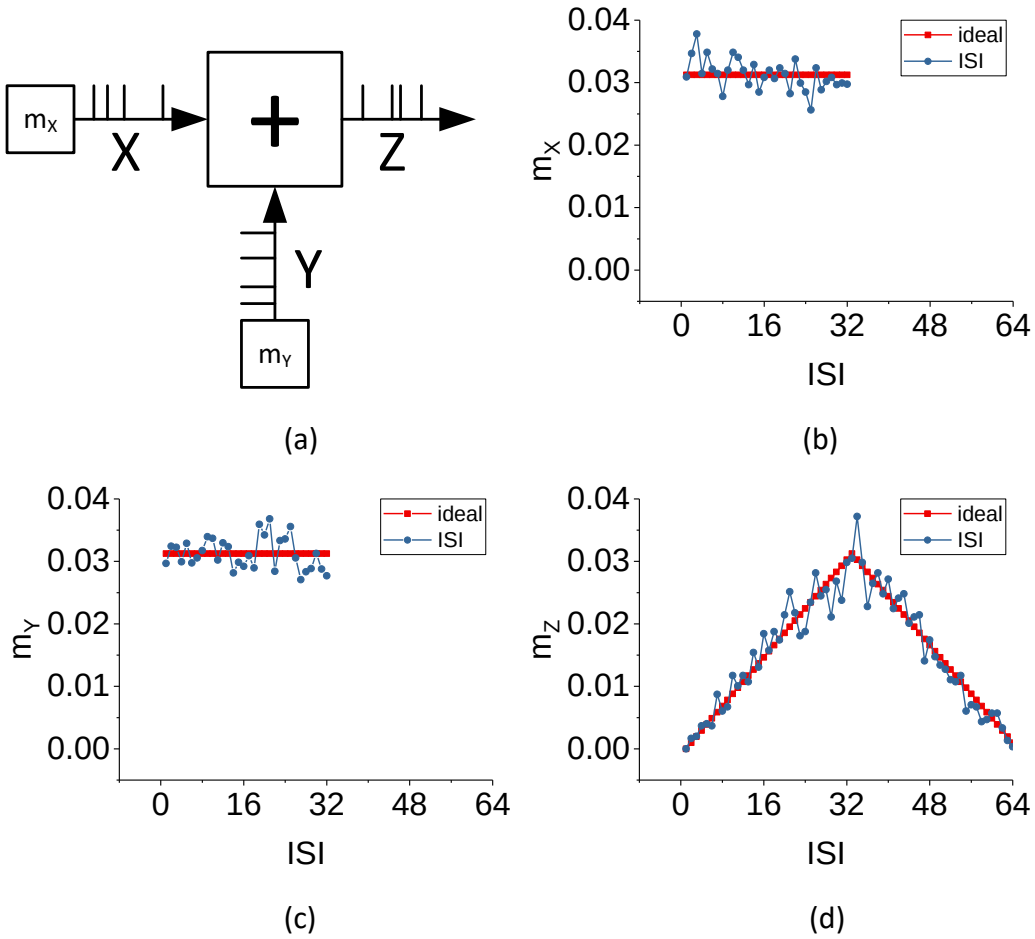


Figure 3.3 (a) Unidirectional factor node with the equality constraint function. The input spike trains are generated from two RS blocks. (b) Distribution of message m_X. The red curve shows the defined probability and the blue curve shows the output ISI distribution from the spike train with 11,000 samples. (c) Distribution of message m_Y (d) Distribution of message m_Z

Next, a simple network with two factor nodes and four RS blocks is constructed as shown in Figure 3.4(a). This simulation demonstrates that a similar result can be obtained using the event-based BP model. Because the messages in this network are restricted to binary, the possible ISIs in a spike train are either 1 or 2, which maps the value of 0 or 1 in the binary domain. One of the factor nodes is the **equality** constraint node and the other is the **xor** constraint node defined in (3.2). The output message can be computed using the SPR in (3.3).

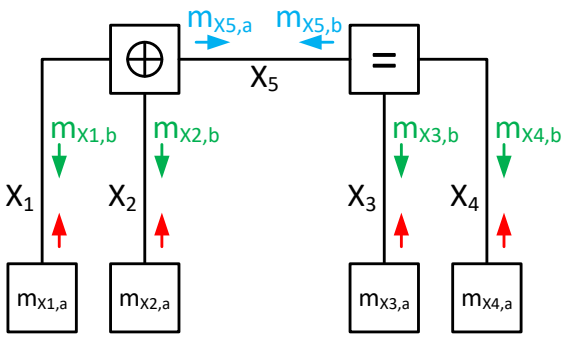
$$f_{xor}(x,y,z)=\delta(x\oplus y\oplus z) \tag{3.2}$$

$$\begin{pmatrix} m_Z(1) \\ m_Z(2) \end{pmatrix} = \begin{pmatrix} m_X(1)m_Y(1) + m_X(2)m_Y(2) \\ m_X(1)m_Y(2) + m_X(2)m_Y(1) \end{pmatrix} \tag{3.3}$$

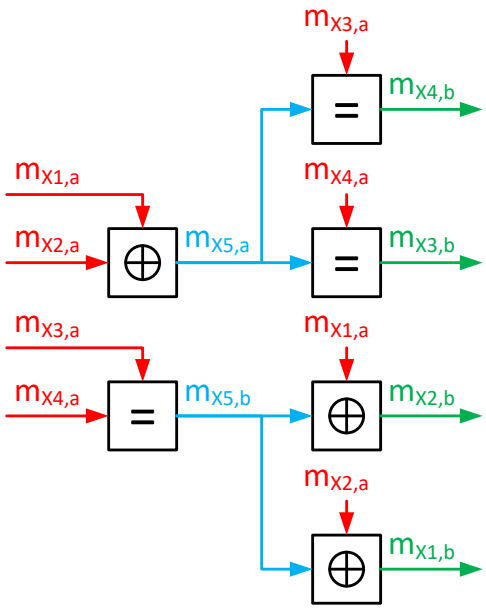
The network is further expanded to the one shown in Figure 3.4(b) in order to obtain the output ISI messages from both directions. First, the spike trains with messages $m_{X1,a}$ to $m_{X4,a}$ are generated using the four RS blocks. These spike trains are sent to the xor and equality constraint node for producing the spike trains with messages $m_{X5,a}$ and $m_{X5,b}$, respectively. Next, the spike trains carrying messages $m_{X5,a}$ and $m_{X3,a}$ are sent to the equality constraint node which generates the spike train carrying message $m_{X4,b}$ and so on as Figure 3.4(b) shows. Finally, all messages of variables X_1 to X_5 are obtained. With these messages, to compute the marginal probability of any of the variables is possible using an equality constraint node as shown in Figure 3.4(c) because the marginal probability of a variable is proportional to the product of both directional messages (see (2.5)).

By setting the binary distributions of messages $m_{x1,a}$ to $m_{x4,a}$ in the RS blocks and following the message passing described above, the marginal probabilities p_{x1} to p_{x4} can be obtained. Messages $m_{x1,a}$ to $m_{x4,a}$ are set as follows: [$m_{x1,a}$,$m_{x1,a}$,$m_{x1,a}$,$m_{x1,a}$] = [(0.9,0.1),(0.9,0.1),(0.1,0.9),(0.9,0.1)]. The spike train that carries the binary message only use two ISI values, i.e. ISI = 1 or 2. The binary value “0” is represented by ISI = 1 while the binary value “1” is represented by ISI = 2.

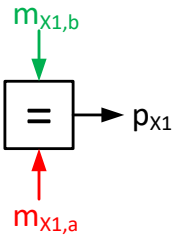
Table 3.2 shows the binary distributions of p_{x1} to p_{x4} computed by the SPR and the ISI counting in the spike trains. In general, increasing the number of ISIs leads to a better approximation between p_{ISI} and the theoretical probability distribution from the SPR. Using the KL divergence to index the similarity of two distributions as shown in Figure 3.5, the trend that increasing the number of ISIs leads to the decrease of the KL divergence can be seen. The suddenly drops of the curves in Figure 3.5(a) are because their ISI distributions just match the theoretical one in this trial. The positons of these drops vary from trial to trial as shown in another trial of Figure 3.5(b).



(a)



(b)

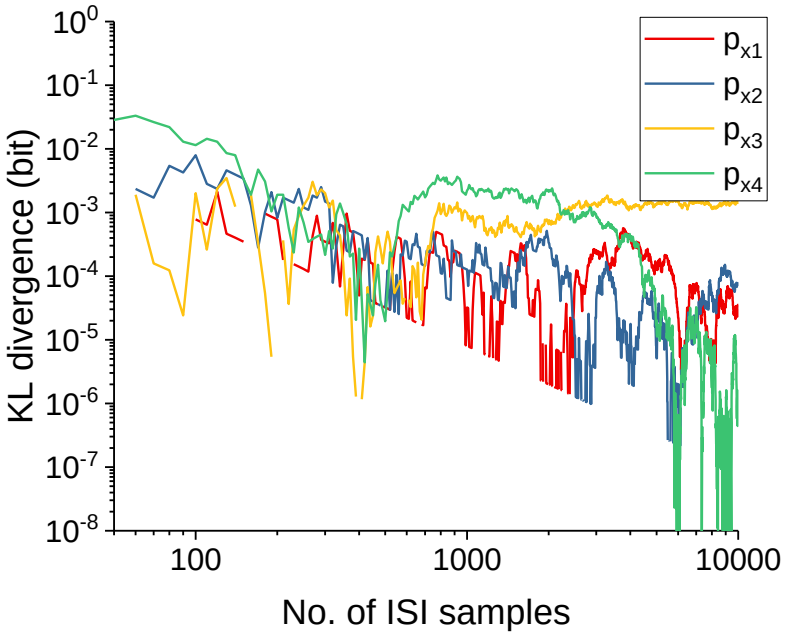


(c)

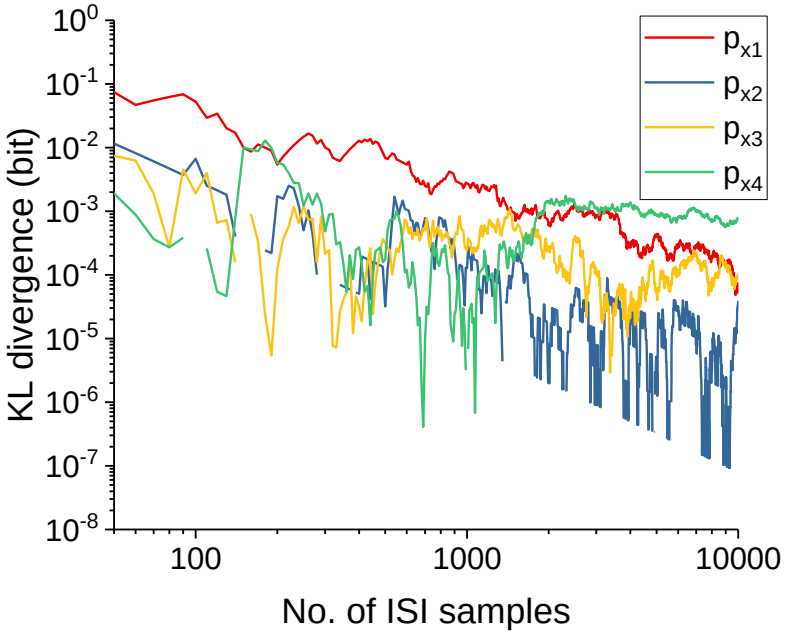
Figure 3.4 (a) Example network with two factor nodes and four semi-factor nodes. (b) Expanded form for computing messages in both directions. (c) Computing the marginal probability of variable X_1 using one equality constraint node to combine the messages from two directions.

	SPR	Event-Based Model (# of ISIs)		
		100	1000	10000
p_{x1}	(0.9,0.1)	(0.89,0.11)	(0.902,0.098)	(0.898,0.102)
p_{x2}	(0.9,0.1)	(0.93,0.07)	(0.896,0.104)	(0.897,0.103)
p_{x3}	(0.82,0.18)	(0.84,0.16)	(0.834,0.166)	(0.837,0.163)
p_{x4}	(0.82,0.18)	(0.77,0.23)	(0.841,0.159)	(0.821,0.179)

Table 3.2 Probability distributions of messages computed from the SPR and the event-base BP model and their dependence on the number of ISIs.



(a)



(b)

Figure 3.5 KL divergences of the marginal probailities p_{X1} to p_{X4} in (a) trial 1 and (b) trial 2.

3.3 Applications

The model investigated in Section 3.2 is now applied to two specific problems: The Kalman filter for object tracking and the CRBM for data reconstruction.

3.3.1 Object Tracking

Because of the ubiquitous noise in our environment, our observations (measurements) or predictions of the targeted physical quantities such as the position are both inaccurate. Such inaccuracy might propagate over time resulting in a significant error. The Kalman filter [90], [91], a state-space model, provides a method to estimate some physical quantities using the information coming from both the prediction and measurement. The equation of the Kalman filter describing the transition of the state is presented in (3.4),

$$\begin{aligned} x_t &= F_t x_{t-1} + B_t u_t + w_t \\ z_t &= H_t x_t + v_t \end{aligned} \tag{3.4}$$

where

x_t: state vector at time t

u_t: vector containing any control inputs

F_t: state transition matrix

B_t: control input matrix

w_t: vector containing the process noise term

z_t: vector of measurements

H_t: transition matrix mapping the state vector into the measurement domain

v_t: vector containing the measurement noise term

Since the true state of the physical quantities (or the system) of interest cannot be directly observed, Kalman filter provides an algorithm that computes the estimate considering the information from the prediction, which is based the model of the system, and the noisy measurement. The true state can be estimated using two quantities, a posteriori estimate $\hat{x}_{t|t}$ and a posteriori error covariance matrix $P_{t/t}$. The process involves a prediction base on $\hat{x}_{t-1|t-1}$ and $P_{t-1/t-1}$ from previous time step to

compute $\hat{x}_{t|t-1}$ and $P_{t/t-1}$ as shown in (3.5) and an update from the measurement to obtain $\hat{x}_{t|t}$ and $P_{t/t}$ as shown in (3.6).

$$\begin{aligned} \hat{x}_{t|t-1} &= F_t \hat{x}_{t-1|t-1} + B_t u_t \\ P_{t|t-1} &= F_t P_{t-1|t-1} F_t^T + Q_t \end{aligned} \tag{3.5}$$

where

$\hat{x}_{t|t-1}$: predicted state

$P_{t/t-1}$: predicted covariance matrix

Q_t: covariance matrix of w_t

$$\begin{aligned} \hat{x}_{t|t} &= \hat{x}_{t|t-1} + K_t \left(z_t - H_t \hat{x}_{t|t-1} \right) \\ P_{t|t} &= P_{t|t-1} - K_t H_t P_{t|t-1} \end{aligned} \tag{3.6}$$

where

K_t: Kalman gain (= $P_{t|t-1} H_t^T \left(H_t P_{t|t-1} H_t^T + R_t \right)^{-1}$)

R_t: covariance matrix of v_t

In addition, the Kalman filter can be mapped into a Forney Factor Graph (FFG) previously shown in [57]. Equation (3.4) can be interpreted as a FFG as shown in Figure 3.6. $\hat{x}_{t-1|t-1}$ and $P_{t-1/t-1}$ in (3.5) and $\hat{x}_{t|t}$ and $P_{t/t}$ in (3.6) are computed in the relative positions of the graph as shown in red. However, the state update is computed in a different way in a FFG in [57], [58]. Assuming all the noise sources have a Gaussian distribution, Loeliger et al. show that the messages in this network can be represented by mean vector and covariance matrix. The message passing in terms of computing the output message of a factor node can be computed using an established table, where the required constraint functions such as plus, equality and gain are described.

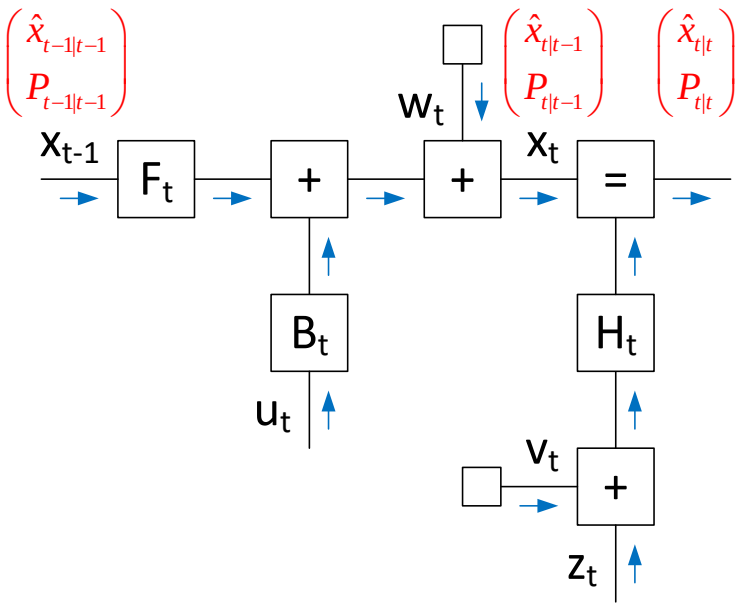


Figure 3.6 Kalman filter represented by a FFG. The next state is updated by the messages along the blue directions. The parameters in red indicates the relative positions of the quantities computed in (3.5) and (3.6).

In this section, an event-based Kalman filter is constructed using the factor nodes in the event-based BP model described in Section 3.2. The event-based Kalman filter is used for one-dimensional object tracking that tracks the position of a falling tennis ball. The measurement is obtained from the Dynamic Vision Sensor (DVS) [14], an event-based retina that generates events when any of the pixels detects a temporal contrast change above a threshold. The DVS reduces data redundancy by producing asynchronous scene reflectance temporal contrast address-events. Figure 3.7 shows the structure of a DVS pixel. Incident light photocurrent is first generated by the photodiode and is converted to a voltage logarithmically in the photoreceptor. After a voltage amplifier amplifying the logarithmic voltage, the voltage change is compared to the ON and OFF thresholds. Then, the AER logic produce an ON or OFF event and reset the voltage amplifier. The AER representing Address Event Representation (AER) [92]–[95] is a four-phase handshaking protocol developed for transmitting and receiving the addressed-events asynchronously.

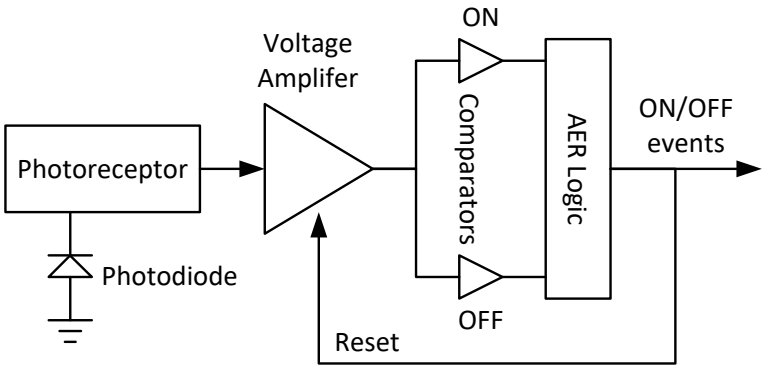


Figure 3.7 Structure of a DVS pixel.

The equations for calculating the position, *s*, and the velocity, *v*, in a state space model with a constant acceleration is shown in (3.7). Note that Δt_{state} in (3.7) is distinct from Δt. The former is the time interval between two consecutive states, called the state step, and the latter is the minimum unit time of an ISI value that associated to the previous simulations. Similar to Figure 3.6, the equation can be also factorized as shown in Figure 3.8. Figure 3.8(a) shows the factor graph representing variable *s* and Figure 3.8(b) shows the factor graph representing variable *v*.

$$\begin{aligned} s_t &= s_{t-1} + v_{t-1} \cdot \Delta t_{state} + 0.5 \cdot a \cdot \Delta t_{state}^2 \\ v_t &= v_{t-1} + a \cdot \Delta t_{state} \end{aligned} \tag{3.7}$$

In Figure 3.8(a), the position $s_{est,t}$ at state t is estimated according to the prediction $s_{pre,t-1}$, which is also the estimation of previous state $t-1$, and the current DVS measurement $s_{mea,t}$. The prediction noise ns_t is generated by a RS block that the input probability distribution is defined as a Gaussian-distributed and the function f_s derived from (3.7) is defined in (3.8). Function f_s is similar to a plus constraint function that sums the input ISIs together. However, the number of inputs of the factor node can be more than two because the message passing in this task is unidirectional. The ISIs in $v_{pre,t-1}$ has to multiply the state step Δt_{state} before summing. The last term of (3.8) is a time-invariant value so that it can be treated as a bias term inside the function instead of an input. Of course, a gain constraint node can also be used to represent Δt_{state}. Velocity $v_{pre,t-1}$ first goes to the gain node and the output spike train of the gain node is then sent to the node with function f_s for summing. Here, all variables are put together in one node to simplify the network.

$$f_s = \delta\left(s_{pre,t} - \left(s_{pre,t-1} + ns_t + v_{pre,t-1} \cdot \Delta t_{state} + 0.5 \cdot a \cdot \Delta t_{state}^2\right)\right) \tag{3.8}$$

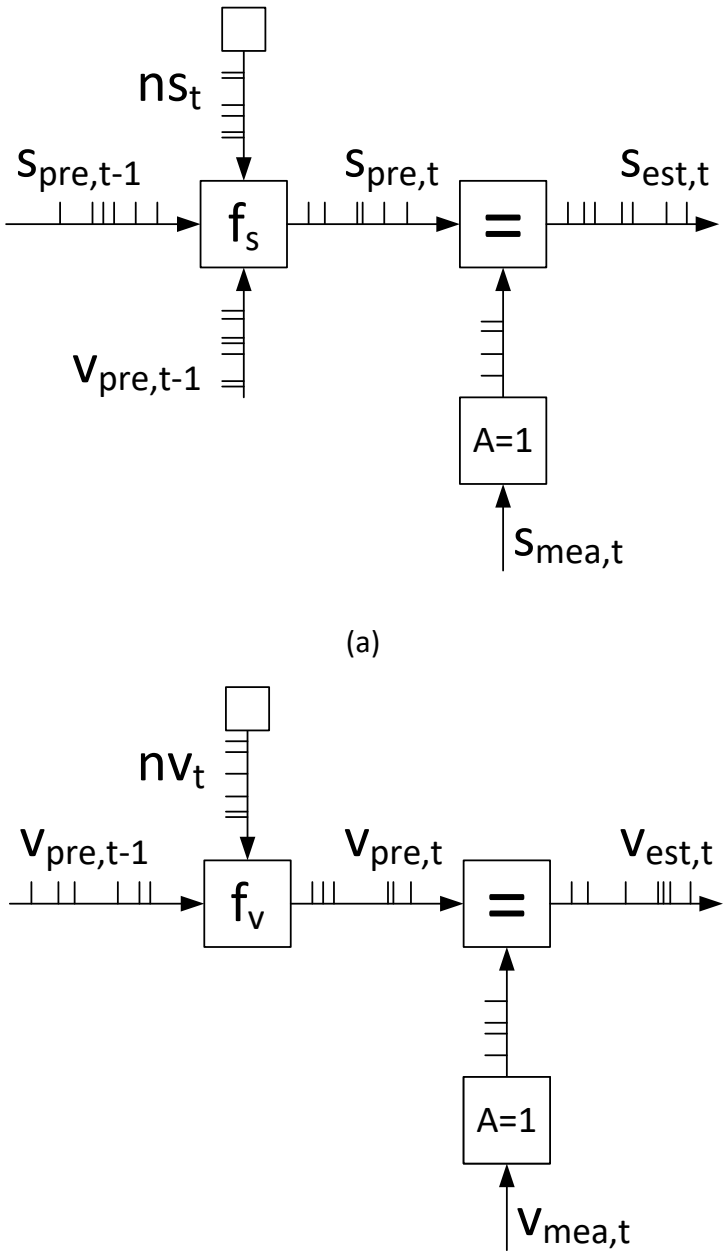


Figure 3.8 Factor Graphs for the task of tracking the falling tennis ball. The graph for (a) the position tracking and (b) the velocity tracking.

The measurement of the position $s_{mea,t}$ is contributed by the DVS that producing the events once detecting the temporal contrast. The information of the positon is represented by the addresses of the events. Therefore, in this example, the value of an ISI represents the address of an event. As shown in Figure 3.8(a), the addressed

events are first sent to a unity-gain (A = 1) node to compute the probability distribution of the addresses and, then, the unity-gain node generates an output spike train according to this distribution. Ideally, the DVS events' addresses can directly form a spike train where the addresses are represented as ISIs. Then the equality constraint node in Figure 3.8(a) use this spike train as its input. However, the number of events might not be sufficient when the tennis ball is just falling because of a low velocity. The equality node is unable to compute the output message. Therefore, a unity-gain node between the DVS events and the equality node can keep producing output spikes within the defined time window.

In Figure 3.8(b), nv_t is generated by a RS block that the input probability distribution is defined as a Gaussian-distributed as well. Function f_v is defined in (3.9). Since a sensor that measures the velocity is unavailable, artificial data is used to represent the measurement of the velocity $v_{mea,t}$. The artificial data are obtained by adding the values of the Gaussian noise on the theoretical velocity computed by (3.7). If the velocity can be extracted from a future event-based sensor, this artificial block can be possibly replaced.

$$f_v = \delta\left(v_{pre,t} - \left(v_{pre,t-1} + nv_t + a \cdot \Delta t_{state}\right)\right) \tag{3.9}$$

Table 3.3 shows the parameters used for this tracking task. The state is updated every 0.01 s meaning that the DVS events are collected during a duration of 0.01 s. Because the position is estimated from the addresses of the DVS events in pixel coordinates, i.e. an ISI represents the location of the pixel that produces the event, the prediction is also computed in pixel coordinates for consistency. Therefore, the unit of the acceleration is set as pixel/s^2 instead of m/s^2.The ratio of the two units is computed according to the range of the falling distance covered by the 128 pixels of the DVS sensor.

Parameter	Quantity	Unit
ISI time step Δt	1	
time window W	10000	
state step Δt_{state}	0.01	s
DVS pixel number	128	pixel
falling distance	0.81	m
accerleration a	$9.8\times\frac{128}{0.81}=1549$	pixel/s^2
standard deviation of ns	2	pixel
standard deviation of nv	2	pixel/s

Table 3.3 Parameters for object tracking

Figure 3.9 shows a 0.01 s time slice of the DVS event output when the tennis ball was falling. The events in white and black represent the OFF and ON temporal contrast change events, respectively. Since only one-dimensional movement along the y axis is considered, the events happening in the same row are all considered as the same addressed event. The numbering of the DVS pixel along the y axis is 1 at the bottom and 128 at the top as shown in Figure 3.9. The ON and OFF events produced at the edge of the ball are due to the intensity contrast change. Both types of events are considered in estimating the position. Figure 3.10 shows the measured and estimated positions at different time (or states). A red dot represents the average of all addresses of the DVS events within 0.01 s while a blue dot represents the average of all ISIs in the spike train s_{est} within the time window W. The ball is placed and dropped around position 120 in the image. This value is used as the initial predicted position. However, the measured position (~= 95) is far from it due to the low velocity in the beginning. The events caused by the object movement are not sufficient so that the random events of the DVS play a role. s_{est} considers both the prediction and the DVS measurement so the estimation curve (the blue curve) approaches to the theoretically quadratic curve better than only the measurement considered (the red curve).

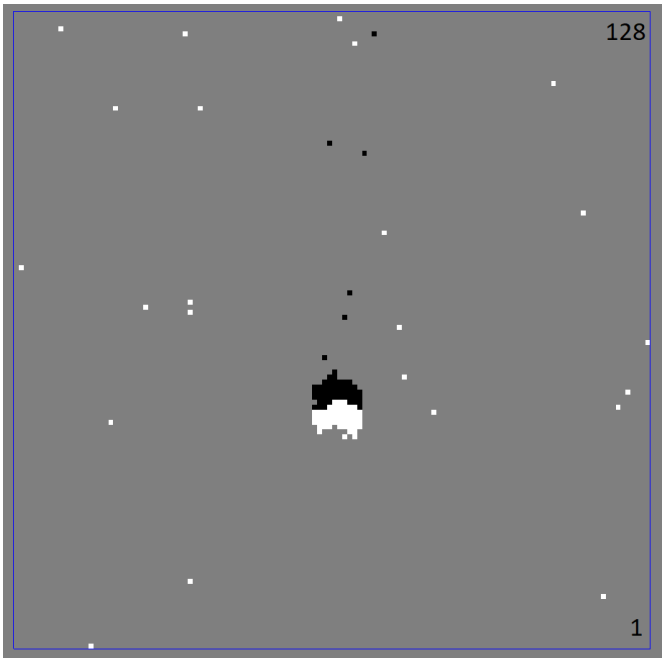


Figure 3.9 Screenshot of the DVS at a moment of the tennis ball falling.

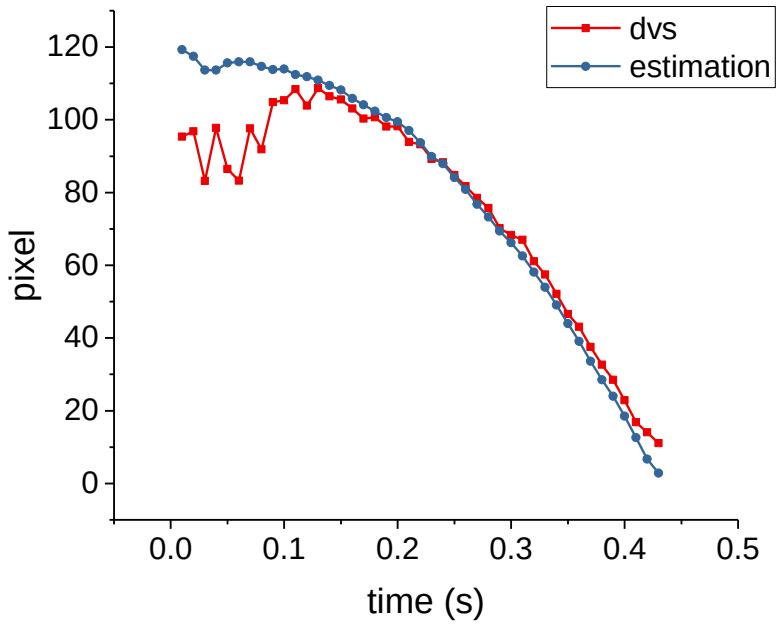


Figure 3.10 Position of the ball as a function of time. Each red dot represents the average of all event addresses in a 0.01 s time slice. Each blue dot represents the average of ISIs in the spike train s_{est} within the time window W.

Figure 3.11(a) shows the probability distribution of all addresses of the DVS events at t_{state} = 0.02, 0.12, 0.22, 0.32 and 0.42 s, respectively, while Figure 3.11(b) shows the probability distribution of all ISIs in the spike train s_{est} in the same condition of t_{state}. When t_{state} = 0.02s, the ball is about to fall. The probability distribution (the red in Figure 3.11(a)) of the DVS events' addresses at this t_{state} does not reflect the position of the tennis ball due to the low velocity. However, the estimated distribution shown in Figure 3.11(b) in red is relatively concentrated because of considering both the prediction and the measurement using the equality node. These random events from the DVS are thought to be filtered out by the equality node. The reason is explained as follows. Because the output probability distribution of the equality node theoretically is the product of the two input distributions, those non-overlapped areas in the two input distributions will output 0 after the product. The initial predicted position is set to pixel 120. Therefore, the random events are filtered out after the equality node. The measured distributions in other states also show a little scatter in Figure 3.11(a) but these noise events are again filtered out in the estimated distributions, i.e. the ISI distribution of spike train s_{est}, in Figure 3.11(b).

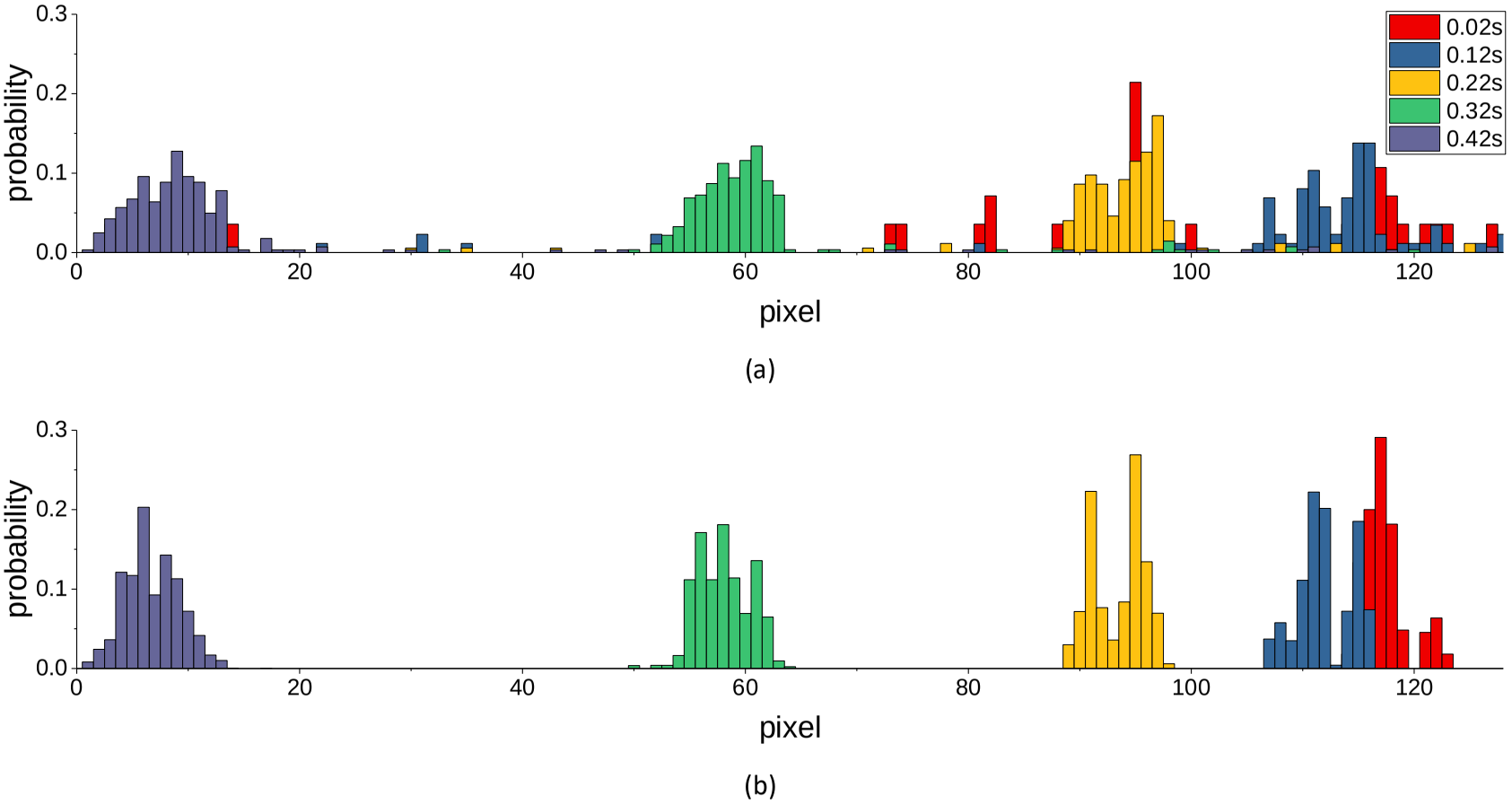


Figure 3.11 Probability distributions in different states. The tennis ball is placed around pixel 120 in the beginning and starts falling freely toward pixel 1. The histogram is obtained by (a) counting all addresses of the events from the DVS and (b) the ISIs in the spike train s_{est} in the time window W.

3.3.2 Data Reconstruction with an Event-Based CRBM

The second example is the event-based CRBM. This event-based version is converted from the continuous-valued CRBM. That is, information is transmitted using spike streams rather than continuous-valued variables. This section first briefly introduces the CRBM by starting with the RBM.

The RBM is a fully-connected network with two layers of neurons, a visible and a hidden layer. Figure 3.12 is an example of a RBM with two visible, two hidden and two bias neurons (v_0 and h_0). There is a weight with each connection. For example, w_{12} represents the bidirectional weight between neurons v_1 and h_2. Each neuron, except for the bias neuron, generates a binary output sampled from some probability distribution composed of two components, $p(s_j=0|\mathbf{s})$ and $p(s_j=1|\mathbf{s})$, as described in (3.10). $p(s_j=1|\mathbf{s})$, where **s** represents an output vector of the neurons in the previous layer, indicates the probability of neuron s_j generating an output sample of 1 under the condition that every neuron in the previous layer contributes its "one" output value to vector **s**. For example, $p(h_2=1|\mathbf{v}=[v_0,v_1,v_2])$ means that the probability of generating an output sample of 1 from neuron h_2 is determined by the three output values of neurons v_0 to v_2. The three values are multiplied with their weights and then sent to a sigmoid function φ_j, the output of which is $p(h_2=1|\mathbf{v})$. Note that the bias neuron always generates a constant output value of 1.

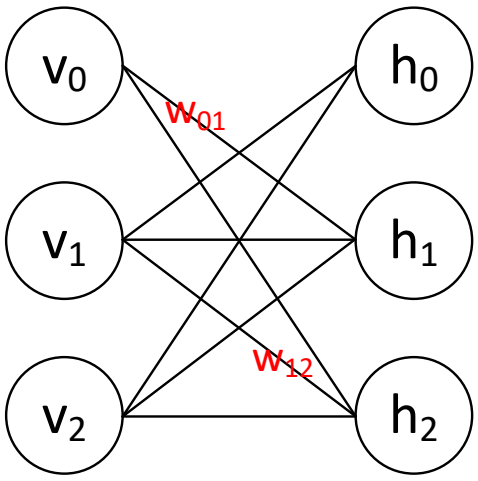


Figure 3.12 RBM with two visible and two hidden neurons. Neuron v_0 and h_0 are bias neurons.

$$
\begin{aligned}
p\left(s_j=1\,|\,s\right) &= \varphi_j\left(\sum_i w_{ij}s_i\right)\\
p\left(s_j=0\,|\,s\right) &= 1-p\left(s_j=1\,|\,s\right)\\
\varphi_j\left(x\right) &= \frac{1}{1+\exp\left(-x\right)}
\end{aligned}
\tag{3.10}
$$

In a RBM, the output is sampled from $p(s_j=1|\boldsymbol{s})$, which is the sigmoid output in (3.10). Therefore, the output of a RBM neuron is either 0 or 1. A CRBM [25] is a modified RBM model, where the output is not limited to the binary. Instead of using the sampled value in RBM, neurons in a CRBM directly send the output of the sigmoid to the next layer. A Gaussian noise is added to the input as shown in (3.11), where σN_j indicates a Gaussian noise source with mean of 0 and variance of σ; ϑ_L and ϑ_H are the lower and upper bounds of the sigmoid; a_j defines the slope of the sigmoid. In the case that $[\vartheta_L,\vartheta_H,a_j]$ = [0,1,1], (3.11) is similar to (3.10) plus an input noise. The stochastic behavior of a RBM is thought to originate from the sampling of the sigmoid output while that of a CRBM is thought to originate from the input Gaussian noise.

$$
\begin{aligned}
s_j &= \varphi_j\left(\sum_i w_{ij}s_i+\sigma N_j(0,1)\right)\\
\varphi_j\left(x\right) &= \theta_L+(\theta_H-\theta_L)\cdot\frac{1}{1+\exp\left(-a_j x\right)}
\end{aligned}
\tag{3.11}
$$

The author chose the event-based BP model on a CRBM rather than a RBM because a CRBM fits the coding scheme of the event-based BP model better. A sample in the event-based BP model is represented as an ISI in the range of $[1,ISI_{max}]$. while the sigmoid output of a CRBM is in the range of $[\vartheta_L,\vartheta_H]$. These two ranges can be associated. For example, in the case that $[1,ISI_{max}]$ = [1,33] and $[\vartheta_L,\vartheta_H]$ = [0,1], the sigmoid outputs can be represented by ISIs with a 5-bit resolution (= $(ISI_{max}-1)/(\vartheta_H-\vartheta_L)$). Of course, a RBM can also be implemented using the event-based BP model. However, the outputs of a RBM are limited to only two ISI values, i.e. ISI = 1, 2, to represent the binary values. No matter which model is implemented, the architecture of a factor node in the event-based BP model does not change (the LS + the RS block). Therefore, it is more efficient to implement a CRBM than a RBM because more information is contained in the ISIs. [25] shows that a CRBM with fewer neurons can do better than a RBM in a reconstruction task. The reason could be that the output of the sigmoid is not quantized in a CRBM so more information is preserved.

Figure 3.13 shows the FFG of the hidden layer in a CRBM with two visible and two hidden neurons. The inputs to the hidden layer are spike trains that the ISIs represent the sigmoid outputs of the visible layer and the Gaussian noise sources. The visible layer has a similar architecture to the hidden layer shown in Figure 3.13 except that the inputs come from the sigmoid outputs of the hidden layer. Ideally, gain constraint nodes have to be added to the architecture in Figure 3.13 to represent the weights, a plus constraint node for summing and a constraint node with the sigmoid function according to (3.11) so that the internal messages are transmitted by spikes. However, this implementation does not work when the author looked into details. The reason is explained as follows.

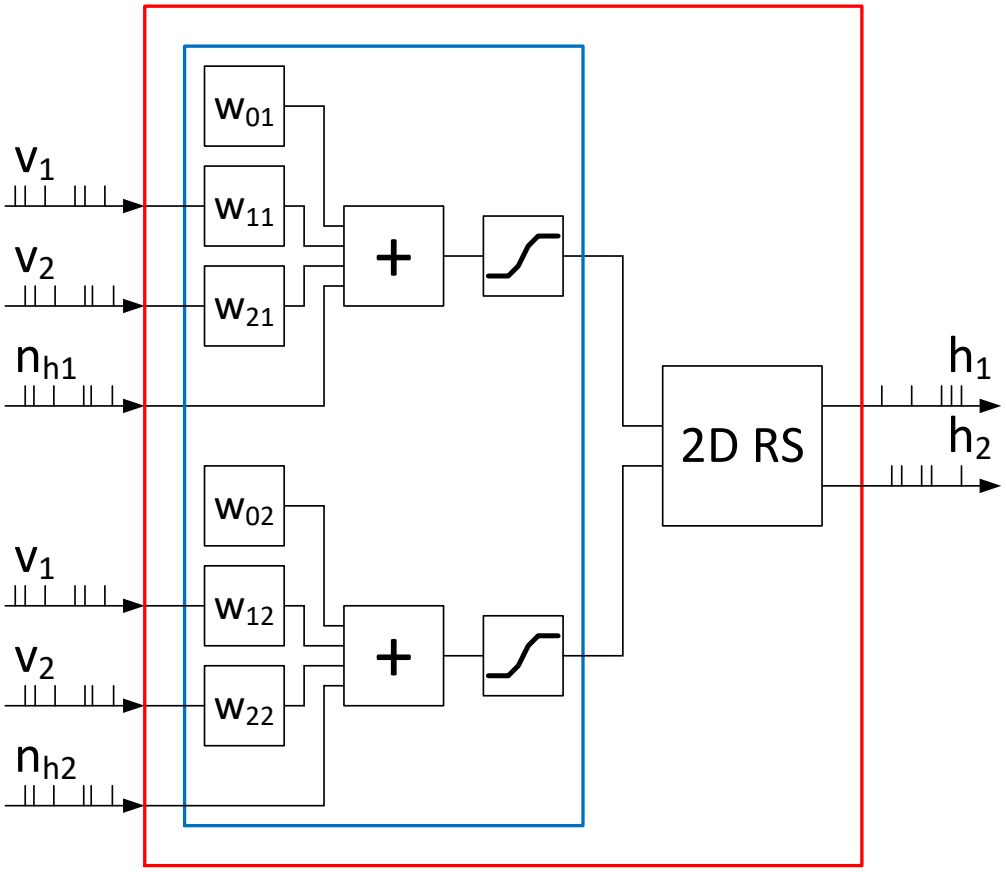


Figure 3.13 FFG of the hidden layer in a two-visible-two-hidden-neuron CRBM.

In a RBM or a CRBM, the neurons in a layer are conditionally independent of each other. That is, given an input vector, each neuron generates its output sample independently of the other neurons in the same layer. The assumption of conditional independence holds only if all neurons in a layer take in one input vector each time and produce one output vector. For example, if there are 1000 input vectors, they

are fed separately to the layer over 1000 time steps. Meanwhile, the neurons in this layer generate 1000 output vectors in response to each input vector. However, a factor node in the event-based BP model uses all the samples at once, builds up a probability distribution from these samples, and then generates all output samples through an output spike train. Therefore, the neurons in a layer cannot be conditionally independent of one another.

To solve the problem that all input vectors represented as ISIs in spike trains are sent to the factor nodes at once, a message-combined joint probability constructed by input vectors is used. Then, this joint probability is sent to a multi-dimensional RS block for producing output spike trains. In this example, the joint probability is two dimensional because both the visible and hidden layers have only two neurons. The entire layer is considered as one factor node (the red box in Figure 3.13) whose constraint function is defined as (3.12) derived from (3.11). This factor node also has two blocks, the LS and the RS blocks. The LS block is shown in the blue box containing all operations and the RS block is a two-dimensional RS (2D RS) block. The hazard function can be still used for generating the output spikes in 2D RS although it is described for a one-dimensional probability distribution. The trick is to use the definition of conditional probability described in (3.13). After the LS block (the blue box in Figure 3.13), a two-dimensional probability distribution $p(h_1,h_2)$ is obtained. The marginal probability $p(h_1)$ and the conditional probability $p(h_2|h_1)$ can be computed according to $p(h_1,h_2)$. Then, an ISI sample for neuron h_1 is first generated using $p(h_1)$. With the ISI_{h1}, an ISI sample for neuron h_2 can be generated using $p(h_2|h_1=ISI_{h1})$. By iterating this procedure, the output spike trains for both neurons h_1 and h_2 is produced. In the implementation, the histograms of $p(h_1)$ and $p(h_2|h_1)$ are directly constructed instead of having $p(h_1,h_2)$ first.

$$f\left(v_1,v_2,h_1,h_2\right)=\delta\left(h_1-\sum_{i=0,1,2}\phi_1\left(w_{i1}v_i\right)-n_{h1}+h_2-\sum_{i=0,1,2}\phi_2\left(w_{i2}v_i\right)-n_{h2}\right) \quad (3.12)$$

$$p\left(x,y\right)=p\left(x\right)p\left(y\,|\,x\right) \quad (3.13)$$

This event-based CRBM is used to perform the same reconstruction task described in [25] and redrawn in Figure 3.14(a). The parameters are first learned from the CRBM and then used in the event-based CRBM. The bit resolution of the parameters is limited after learning so that the performance of the model with lower bit resolution parameters can be inspected. These lower bit resolution parameters could be needed in a future hardware implementation. The bit resolutions are shown in Table 3.4. The mapping of the sigmoid outputs $[\vartheta_L,\vartheta_H]$ and the ISI values $[1,ISI_{max}]$ is from [-1,1] to [1,65].

Parameter	Resolution (bit)
Sigmoid output	5
Weight, w	8
Slope of the sigmoid, a	5

Table 3.4 Resolutions of the parameters in CRBM

Table 3.5 shows the parameter values after learning. The weights of the interconnections are shown in blue and the slope of the sigmoid and the standard deviation of the Gaussian noise in each neuron are shown in red. All noise sources have a standard deviation of 0.1.

2×2 CRBM	h_0	h_1 a_{h1} (0.7188) σ_{h1} (0.1)	h_2 a_{h2} (15.7813) σ_{h2} (0.1)
v_0	NA	w_{01} (-0.3594)	w_{02} (-0.0586)
v_1 a_{v1} (2.1875) σ_{v1} (0.1)	w_{10} (0.2422)	w_{11} (1.1445)	w_{12} (0.3945)
v_2 a_{v2} (1.1875) σ_{v2} (0.1)	w_{20} (0.1016)	w_{21} (1.4531)	w_{22} (-0.5469)

Table 3.5 Parameters after learning

The reconstruction from different epochs are shown in Figure 3.14. In the beginning, 1000 $ISIs_{v1}$ and 1000 $ISIs_{v2}$ form two output spike trains for two visible neurons. Here, the sample number of 1000 is picked because the probability distribution of each neuron can be demonstrated easily. Any number of samples is possibly given. Both ISIs, shown in Figure 3.14(b), are generated randomly by a uniform random variable whose values range from [1,65]. After the initial spike trains are given, the model runs by itself without any new external inputs. That is, the inputs in the second epoch come from the hidden layer and so on. In each epoch, the factor node (either visible or hidden) generates two spike trains of 1000 ISIs each.

After a few iterations (Figure 3.14(c)-(e)), these ISIs converge to two groups similar to the result in [25].

The training data in Figure 3.14(a) is generated by two 2-dimensional Gaussian bumps, where the standard deviations of the two bumps are 0.25 and 0.08 along x axis in Figure 3.14(a) and are 0.1 and 0.1 along y axis. Figure 3.15 shows the ISI distributions of the visible neurons in different epochs. It can be seen that after 15 epochs, the ISI distributions of neuron v_1 and v_2 reflect the differences of the standard deviations in training data.

The ISI distributions of the hidden neurons in different epochs are shown in Figure 3.16. The distribution of neuron h_2 is similar to a binary distribution where most of the ISIs are located at two sides while the distribution of neuron h_1 at the end of the simulation looks like a Gaussian distribution. The reason for the distinct distributions of two hidden neurons is: Neuron h_2 help the visible layer identify two separated groups and neuron h_1 is to support the variation in each group. However, to understand how ISIs are separated in two groups with different standard deviations at the end of the simulation, it is needed to trace the computation along with the iterations.

In this application, a CRBM is turned into an event-based CRBM by treating the entire layer (both visible and hidden) as a factor node. In this way, a group of input vectors, i.e. all ISIs in input spike trains, can be addressed at once. However, this method limits the number of neurons in each layer. For example, if there are three neurons, s_1 to s_3, in a layer, the joint probability represented by the definition of the conditional probability is $p(s_1)\times p(s_2|s_1)\times p(s_3|s_1,s_2)$. To store the values in memory is expensive and the waiting time for the RS block to produce all output events becomes longer. So far the solution to make the neurons in a layer conditionally independent is unknown. Nevertheless, the RS block is still useful. In a RBM, sampling on the sigmoid output is required. The RS block can be used to produce the output sample of each neuron independently. Since each neuron only produces one binary sample each time, the RS block can be reused for the other neurons.

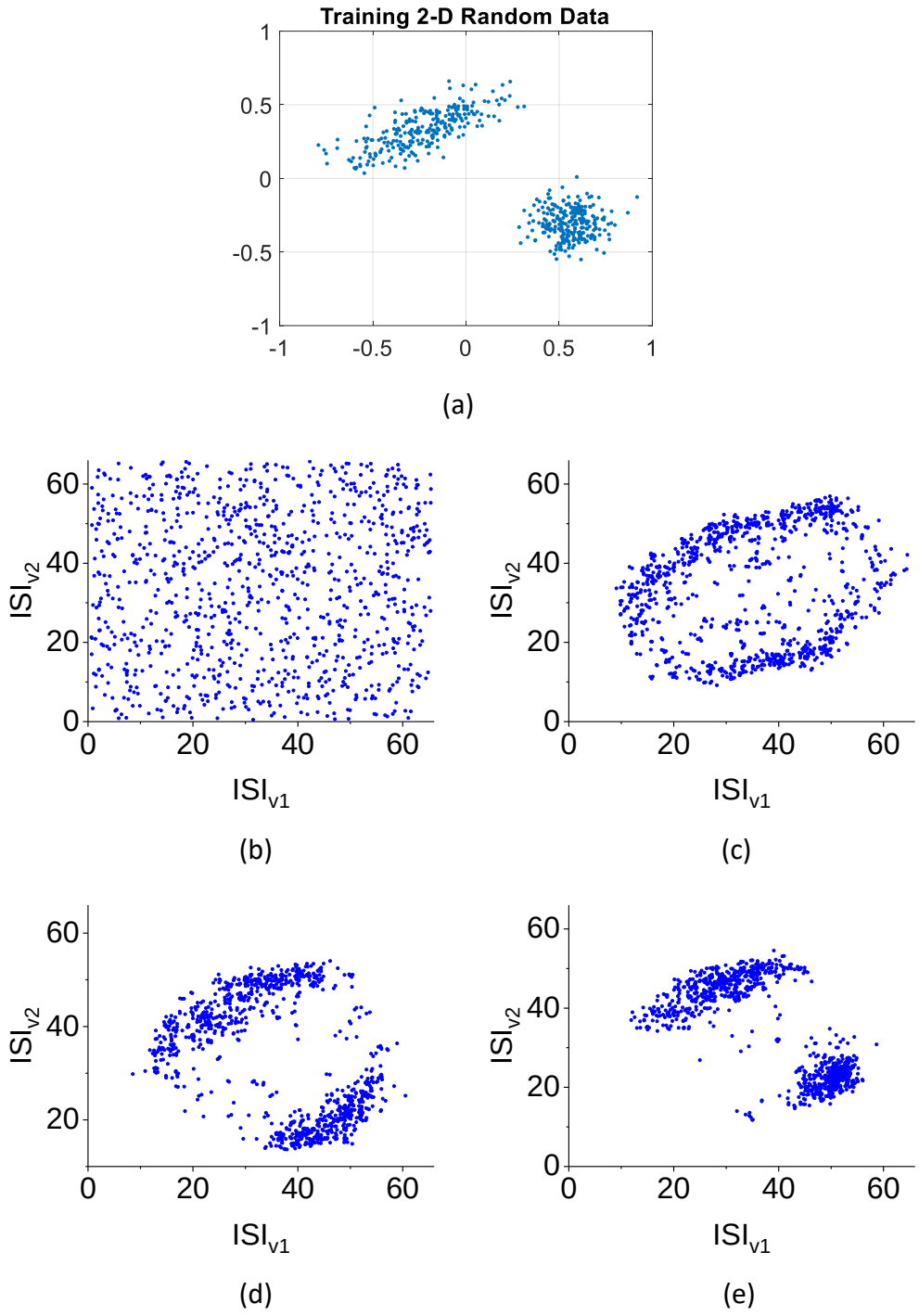


Figure 3.14 Data reconstruction in the event-based CRBM. (a) 2D training data. The reconstruction from the (b) 1st epoch (c) 2nd epoch (d) 5th epoch and (e) 15th epoch.

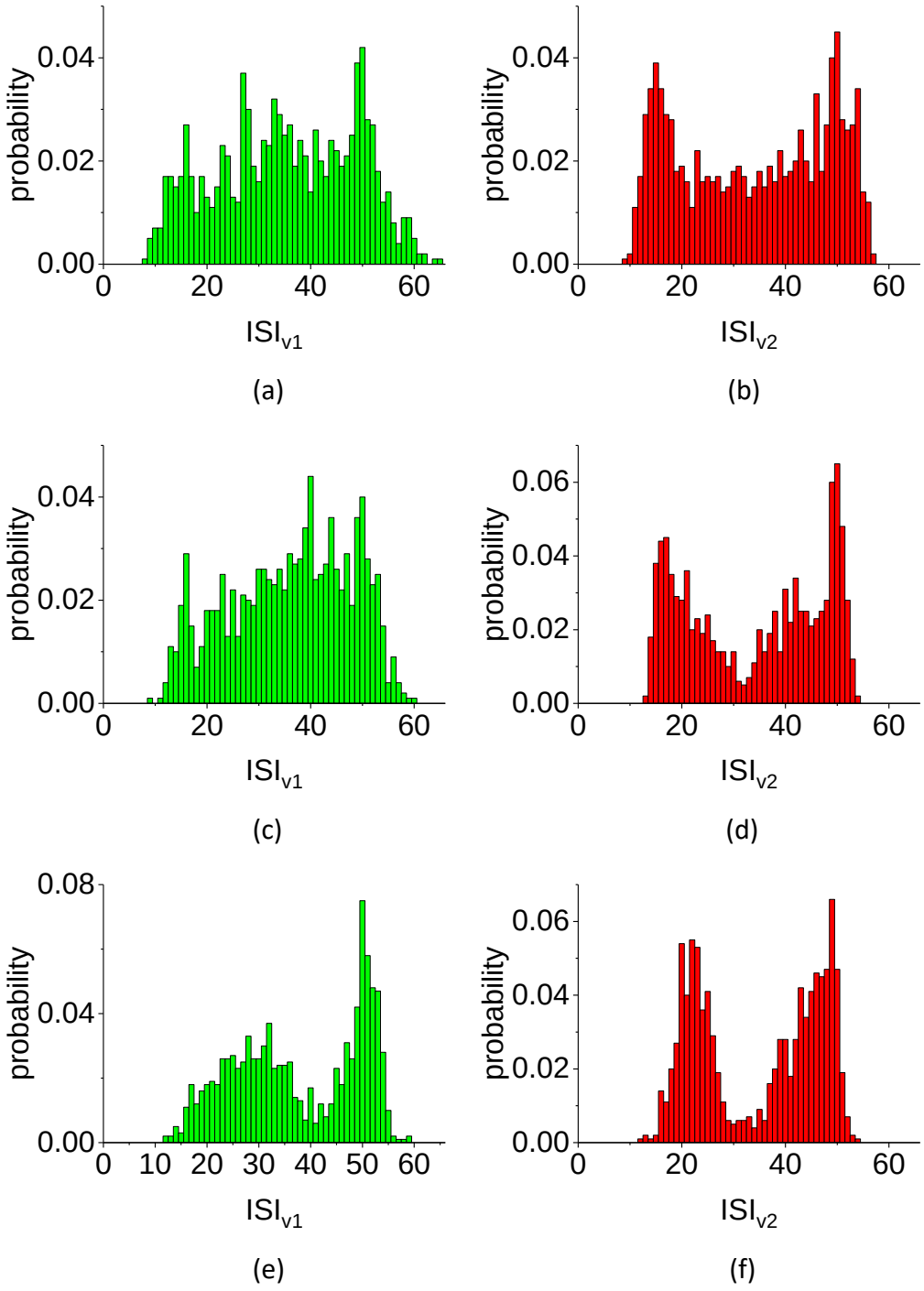


Figure 3.15 ISI distribution of (a) neuron v_1 at 2nd epoch, (b) neuron v_2 at 2nd epoch, (c) neuron v_1 at 5th epoch, (d) neuron v_2 at 5th epoch, (e) neuron v_1 at 15th epoch, and (f) v_2 at 15th epoch.

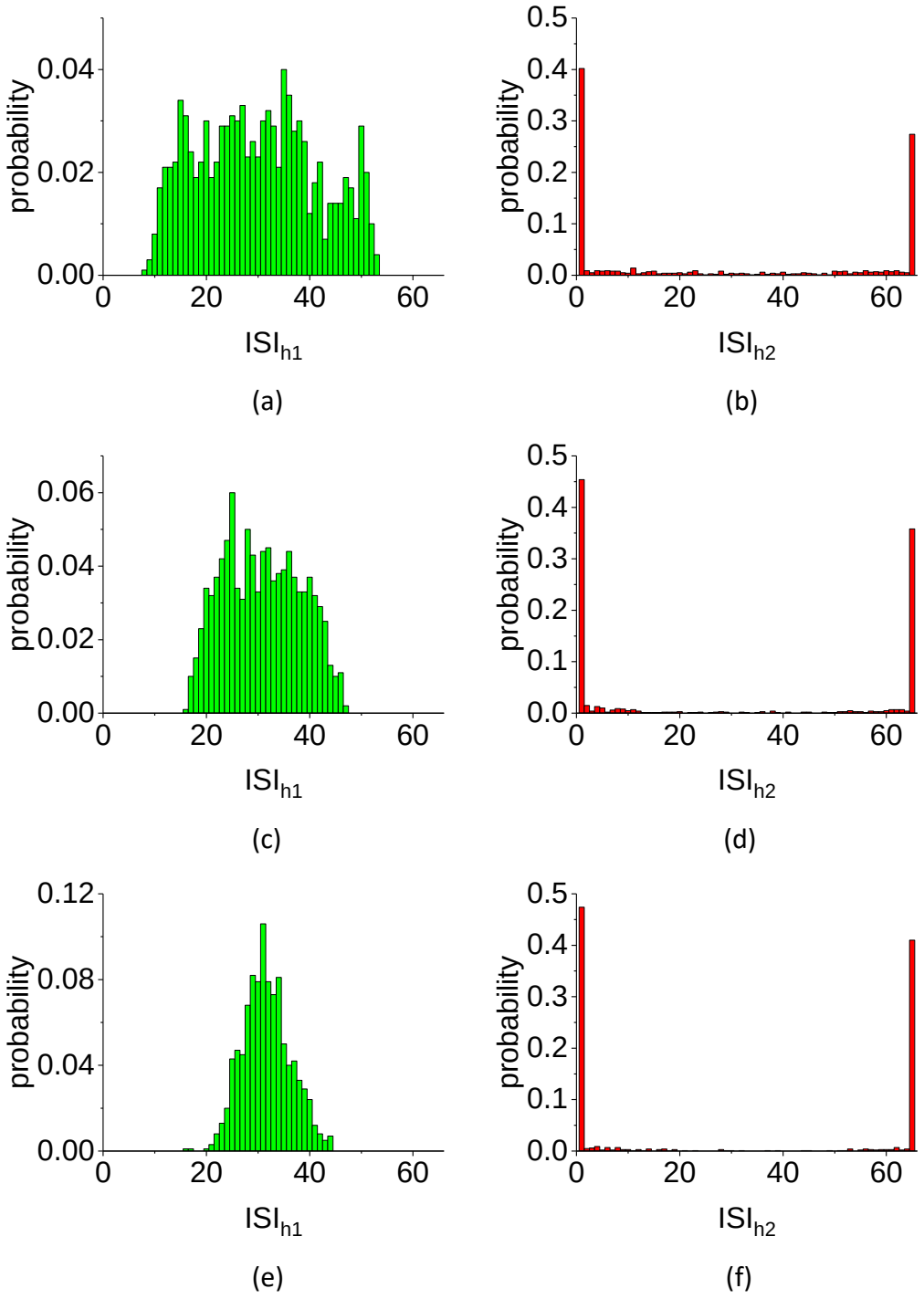


Figure 3.16 ISI distribution of (a) neuron h_1 at 2nd epoch, (b) neuron h_2 at 2nd epoch, (c) neuron h_1 at 5th epoch, (d) neuron h_2 at 5th epoch, (e) neuron h_1 at 15th epoch, and (f) h_2 at 15th epoch.

Chapter 4 Factor Node Hardware

From the model simulations described in Chapter 3, it is confirmed that the event-based Belief-Propagation (BP) model can be applied on the given examples. This chapter describes the hardware realization of the event-based BP model. Since the stochastic model cannot be easily implemented on the currently available spiking network hardware platforms, a VLSI prototype is designed to demonstrate of this model. A large part of this chapter comes from two papers, one (the title is "A Neuromorphic VLSI Circuit for Spike-Based Random Sampling") published in the IEEE Transactions on Emerging Topics in Computing (TETC) in 2015 [82] and the other (the title is "Hardware Implementation of an Event-Based Message Passing Graphical Model Network") published in the IEEE Transactions on Circuits and Systems I (TCASI) in 2018 [83]. The author has only used the text related to the work contributed by the author in the papers.

4.1 System Architecture

The hardware system of the event-based belief propagation model is shown in Figure 4.1. It follows the system architecture described in Figure 2.5. The Landscape Sampling (LS) block is implemented on an FPGA for flexibility and the Random Sampling (RS) block is implemented as an ASIC designed in a 0.35um 2-poly 4-metal CMOS process. With the 16 RS channels on the ASIC chip and the 16 LS channels on the FPGA, factor graphs with up to a maximum of 16 output messages can be configured using this system. The communication from the RS array to the LS array is via the **AER Transmitter** block in the RS chip and the **AER Receiver** block in the LS on the FPGA and using the asynchronous address event representation (AER) protocol [92]–[95]. The connections between the **AER Receiver** and the LS channels are defined in the **Connection Table** block of the FPGA. The **Data Encoder** block combines the messages (m_Z) of the individual LS channels into a single output stream of pulses transmitted consecutively from each channel (see Section 4.2 for details).

This single train of output pulses is then transmitted to the **Data Decoder** block within the RS chip. This block remaps the output spike trains to the corresponding RS channels. In addition, the system has two possible sources of random number generators. The first is through the linear feedback shift registers (LFSRs) and off-chip digital-to-analog converters (DACs) which convert the digital output of the LFSR to analog signals, $V_{nx,s,ext}$, and the second is a pseudo random number generator (RNG) [96] array on the chip providing individual uniform random variables, $V_{nx,s,int}$, for the 16 RS channels. The LS array uses the main FPGA clock (Clk_{main}) for the FPGA modules and also generates three different clocks (Clk_h for the RS channels, Clk_{RNG} for the RNG array, and Clk_{config} for the **Data Decoder** block) using a frequency-divider module. The LS channels use the FPGA clock Clk_{main} to accelerate counting of the InterSpike Interval (ISI) statistics and the RS channels use a frequency divided-down clock Clk_h needed for the longer time constants of the aVLSI circuits.

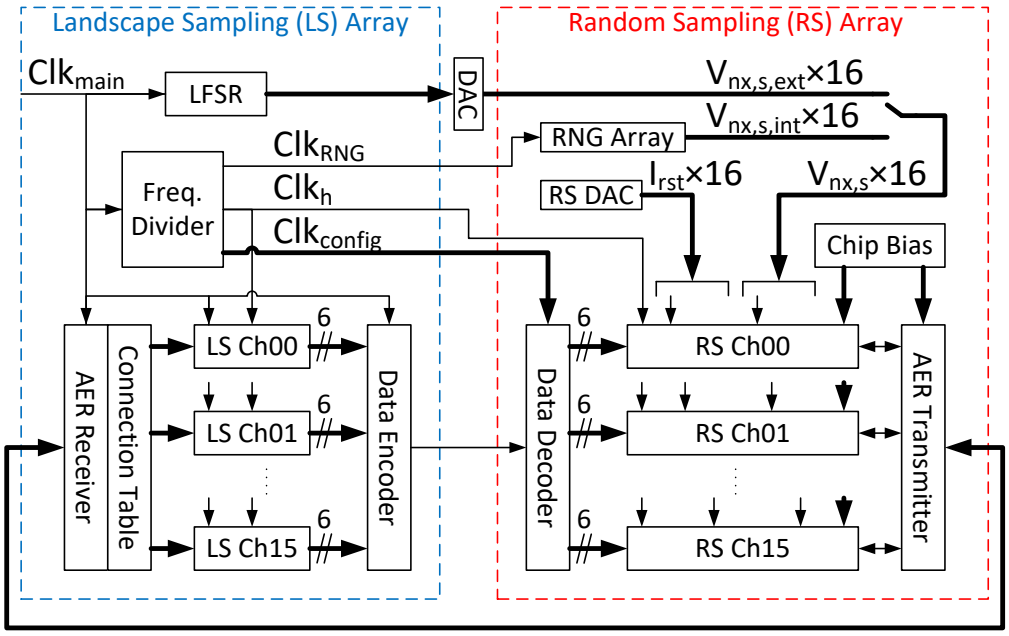


Figure 4.1 System architecture consists of two blocks. Left (dotted blue box), the LS array with 16 channels and right (dotted red box), the RS array also with 16 channels.

4.2 Landscape Sampling

Figure 4.2 shows the structure of one LS channel. The building blocks and signal flow between the blocks follow the basic mechanism described in Section 2.2. The

LS channel is designed for a two-input-one-output or a one-input-one-output message passing. The maximum input number is limited to two in the implementation to reduce structural complexity. However, the input number can be increased in future implementations.

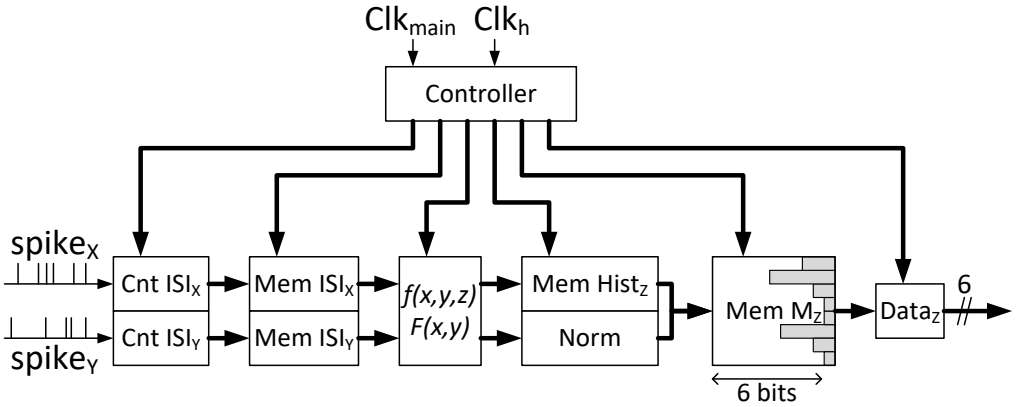


Figure 4.2 Structure of one LS channel

The asynchronous spike trains, $spike_X$ and $spike_Y$, from the AER Receiver block carry the messages, $m_X(x)$ and $m_Y(y)$, of the two variables, *X* and *Y*. The **Cnt ISI$_X$** and **Cnt ISI$_Y$** modules in Figure 4.2 measure the ISIs of the spikes in the event streams of *X* and *Y* respectively. The module starts a counter as soon as a spike arrives. The counter increments in unit time steps, Δt, and stops when the next spike arrives. The measured ISI value is then stored in the memory modules, **Mem ISI$_X$** and **Mem ISI$_Y$**, respectively. At the end of the time window defined in the **Controller** module, the ISIs stored in the memory modules are transferred to the **Function** module which is programmed with the desired factor's constraint function *f(x,y,z)* and summation function *F(x,y)*. Function *f(x,y,z)* in the hardware implementation is limited to delta functions. To ensure enough input samples for extracting the statistics of the distribution, sample pairs *(x,y)* are generated from all combinations of ISIs in the **Mem ISI$_X$** and **Mem ISI$_Y$** modules. This method is slightly different from the pairing method described in [55], where the samples consist of only pairings of the latest ISI values. The histogram counts of *f(x,y,z)* and of *F(x,y)* for normalization are saved to the **Mem Hist$_Z$** and **Norm** modules, respectively. Message m_Z is stored in the **Mem M$_Z$** module after normalization. The index number of ISI bins, *ind*, is in the range of ISI values varying from (*ind* = 1) to (*ind* = ISI_{max}). The value in each m_Z bin is represented by 6 bits, the bit width of the **Mem M$_Z$** module. The picture in the **Mem M$_Z$** module as shown in Figure 4.2 demonstrates an example of a possible probability distribution of the message m_Z. To consume less FPGA resource, the memory values

in **Mem M_Z** module are only updated every time the count of $F(x,y)$ reaches 2^k, where $k \in N$. In this way, normalization only involves a bit shift. The **Data$_Z$** module transmits the value of the individual m_Z bin in a sequential manner starting from ind = 1 to ind = ISI_{max} to the **Data Encoder** block in Figure 4.1. Each **Data$_Z$** module in a channel produces a 6-bit value. The **Date Encoder** block combines the 6-bit data value from all 16 channels into a single output, i.e. [$data_{Z0}$,$data_{Z1}$,$data_{Z2}$,…,$data_{Z15}$], corresponding to a 96-bit data stream. Once an output spike is generated from the corresponding RS channel and sent back to the targeted LS channel through AER interface, the **Data$_Z$** module restarts to transmit the bin value from ind = 1. In addition, the **Data$_Z$** module only sends out the non-zero values stored in the **Mem M_Z** module. The distance between two non-zero values, i.e. the number of bins between the two values, is recorded and used to calculate the correct value of an ISI. The reason for implementing this scheme is related to the design of the RS channel (see Section 4.5.1).

Note that ISI_{max} is always an integer value while t_{ISImax} defined as $ISI_{max} \times \Delta t$ represents the actual ISI time in seconds. This definition is adopted throughout in the following sections.

4.3 Random Number Generator

Before going to the circuit details of the RS block, this section describes a pseudo RNG circuit [96] used for the RS block. The uniform property of the output distribution of the RNG circuit, the simple structure, and a tunable range of the output samples are the reasons that this circuit was chosen.

4.3.1 Discrete-Value Approach

The RNG circuit contains a number of cells in a ring structure as shown in Figure 4.3. The output of each cell is served as the input of the next cell and then the last one comes back to the initial one as a ring. Each cell can be regarded as a random variable that generates samples with a uniform distribution. The output samples is computed based on the method of the delta-sigma automata that follows the rules in (4.1), where $x_n(k)$ indicates the output value x in the n-th cell at the k-th state. The value of cell n at state $k+1$ can be regards as a function of the values of cells n and $n-1$ at the previous state k. In this function, the output is derived from the difference between the summation of these two cells and the sign of them. The function $f(x)$, called quantization residue map as used in single-bit delta-sigma modulation, is shown in Figure 4.4(a), where the input x represents the sum of the two cells and the gray area depicts the range of the possible input and output values.

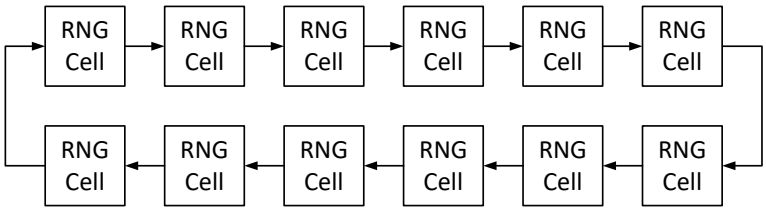


Figure 4.3 Ring structure of the RNG with 12 cells.

$$x_n(k+1) = f\left(x_n(k) + x_{n-1}(k)\right)$$
$$f(x) = x - sign(x) \tag{4.1}$$

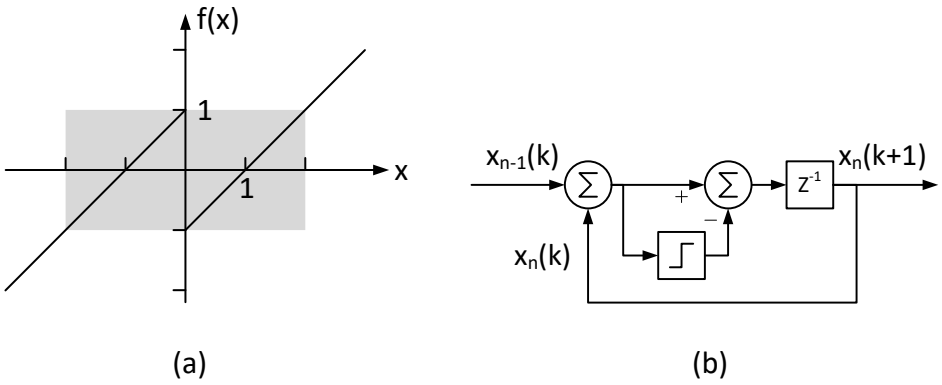


Figure 4.4 (a) Quantization residue map. (b) Block diagram of one RNG cell.

The block diagram in Figure 4.4(b) exhibits one RNG cell based on (4.1). The output value will be updated iteratively. Figure 4.5(a) shows the details of the block diagram, where V_{in} is related to x_{n-1} connected to the previous cell and V_{outA} is related to x_n connected to the next cell. The maximum and minimum values of the output are controlled by V_{RNGp} and V_{RNGn}, respectively. Voltage V_m is set to the average of V_{RNGp} and V_{RNGn}. The switches are controlled by the pulse signals with the patterns shown in Figure 4.5(b), where a four-phase operation (i.e. phase a,b,c,d,a,b,…) is repeated. Basically, an output sample is produced every cycle (i.e. four phases) that means the cell moves to the next state k+1 after a cycle. However, the samples used in the RS block are picked every two cycles that is controlled by signal PICK in Figure 4.5(b) and the value is represented as V_{outB} in Figure 4.5(a). The reason will be explained in the measurement results in the next section.

In the four-phase operation, the RNG cell controlled by the switching signals has different connections as shown in Figure 4.6. In the first phase (Figure 4.6(a)), the

cell samples the values from the previous cell and itself by storing the charges on capacitors C_1 and C_2. Here these two capacitors are set to the same values and the virtual ground indicates V_m. In the second phase (Figure 4.6(b)), the sum of these two values is executed by transferring the charge from capacitor C_1 to C_2. Later, in the third phase (Figure 4.6(c)), the inverted sign of this summed value is computed by using an inverter and storing in capacitor C_1. Finally, in the fourth phase (Figure 4.6(d)), the combination of the sum and its inverted sign is executed. Therefore, a voltage sample is produced on the forth phase of a cycle. The circuit operation in a RNG cell follows the computation of the block diagram in Figure 4.4(b). Every cell has the identical structure. The samples are collected and shown in the measurement results.

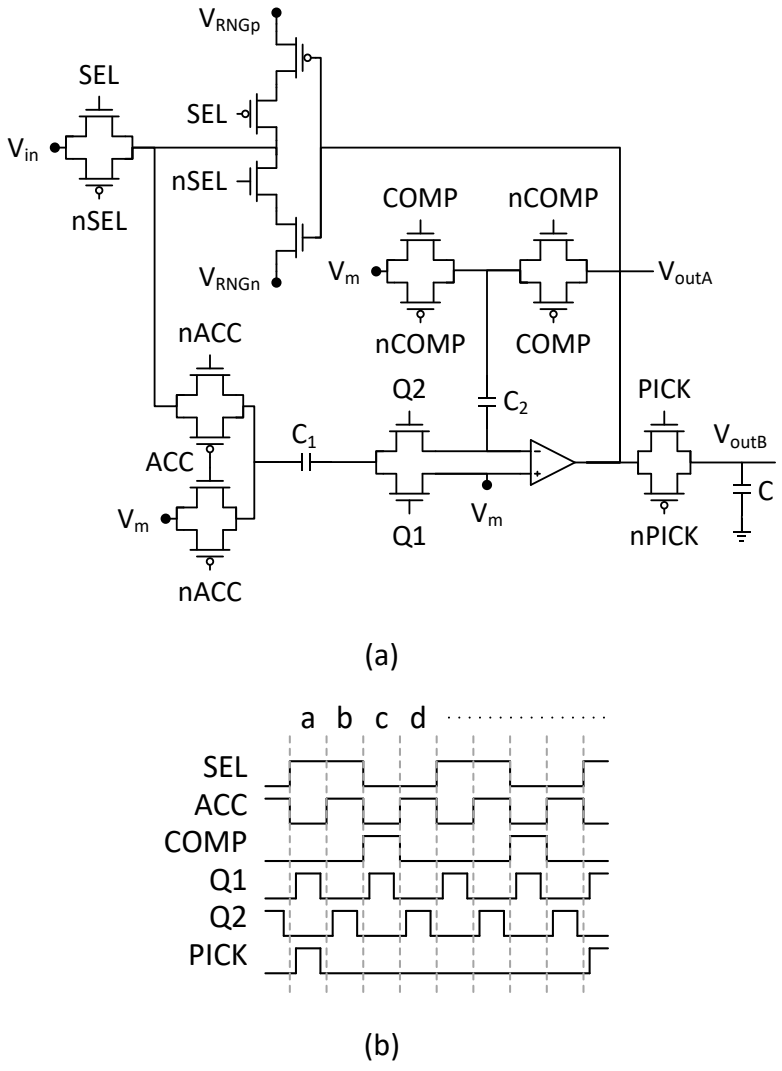


Figure 4.5 One RNG cell. (a) Circuit structure. (b) Control signals.

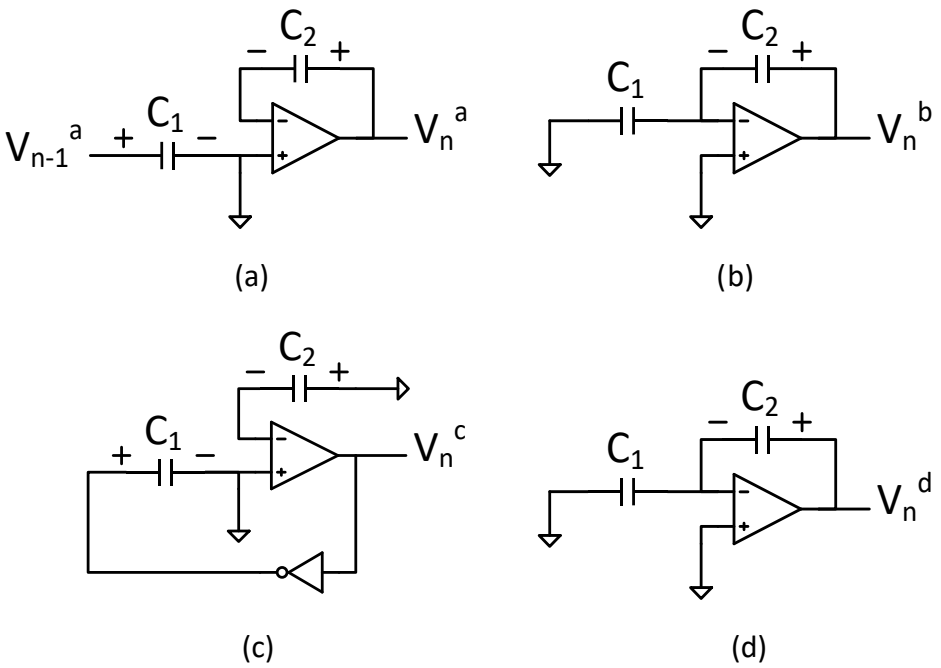


Figure 4.6 Four phases of a RNG cell.

4.3.2 Measurement Results

The micrograph of the RNG in a test chip is shown in Figure 4.13. The first measurement result presents the distributions of the output samples of three RNG cells in a 12-cell ring structure in Figure 4.7. The maximum and minimum voltages, i.e. V_{RNGp} and V_{RNGn}, are set to 2.55 and 0.55 V. The distributions of these three cells approximate a uniform distribution with a little distortion at the values close to the maximum and minimum voltages.

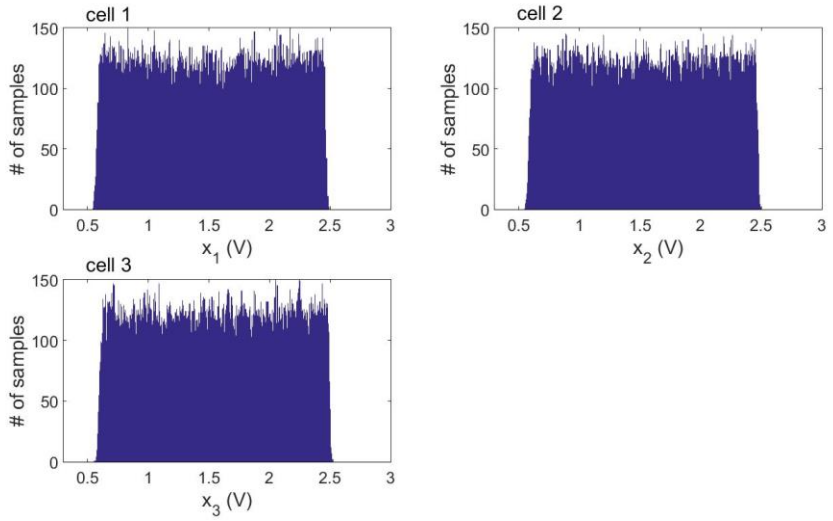


Figure 4.7 Distribution of the random outputs of cells 1, 2 and 3 in the RNG ring.

The second experiment shows the correlations between cells or states in X-Y plots in Figure 4.8. The first row of Figure 4.8 shows the correlation of "cells 1 and 3", "cells 2 and 3", and "cells 3 and 3" that the sample sequences of the three cells, i.e. x_1, x_2 and x_3, are generated in parallel. The second row shows the correlations of these three cells and cell 3 with x_3 delayed one state. The third row shows the correlations that x_3 delayed two states. Each cell is highly uncorrelated to other cells or states except the previous cell and state, i.e "$x_2(k)$ to $x_3(k+1)$" and "$x_3(k)$ to $x_3(k+1)$". This phenomenon is reasonable since the updated value in (4.1) is derived from both of them. The pseudo-number generator can be achieved under this approach choosing which cell desired and avoiding its neighbor state; that is, it is suggested to pick [$x_1(k)$,$x_2(k)$,$x_3(k)$,…] and then jump to [$x_1(k+2)$,$x_2(k+2)$,$x_3(k+2)$,…]. Therefore, the samples are picked every two cycles as described in Section 4.3.1.

Figure 4.9 describes the normalized mutual information [89] of the sample sequences in three neighbor cells to show the independence between them. The sample sequence x_3 is delayed from 0 to 49 states and the normalized mutual information is computed from x_1 and the delayed x_3, and so on. It is clear that the normalized mutual information of x_3 and itself without delaying is 1 so the first data point of the third graph in Figure 4.9 is out of the range in y axis. Besides this data point, only other two points show a higher mutual information, which are "x_2 and the one-state-delayed x_3" and "x_3 and the the one-state-delayed x_3". The results is consistent with that in Figure 4.8.

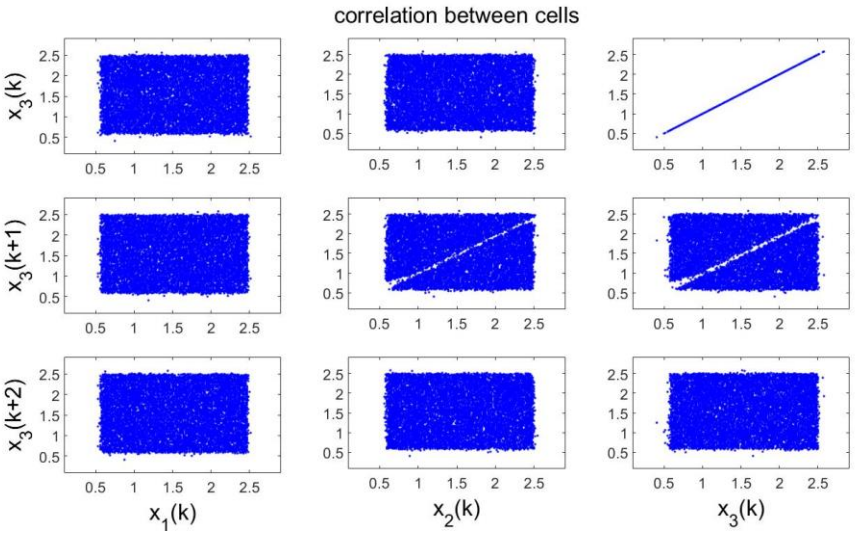


Figure 4.8 Time-space correlogram of different cells or states.

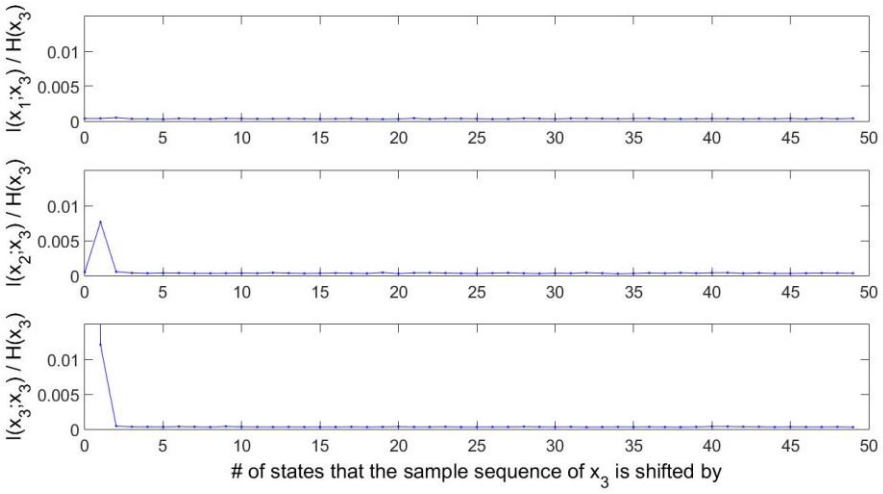


Figure 4.9 Mutual information of the output sequences of cells 1 and 3, cells 2 and 3, cells 3 and 3 with the output sequence of cell 3 delayed from 0 to 49 states. The mutual information is normalized by the entropy of x_3.

The last experiment shows if the sample sequence of a cell is deterministic. A same initial value is given to all cells before closing the loop. Figure 4.10 shows that after a few stats, the output patterns of cell 1 are distinct in different trials even though the same initial condition is given. The variation is contributed by the noise in the device, e.g. the kTC noise or thermal noise. Therefore, the pseudo RNG employed in this thesis is not entirely deterministic, which is useful to run this circuit

for a long time. After describing the operation of the RNG circuit and showing the measurement results of it, the next section describes how the RS block is implemented, where the RNG will be part of the RS circuit.

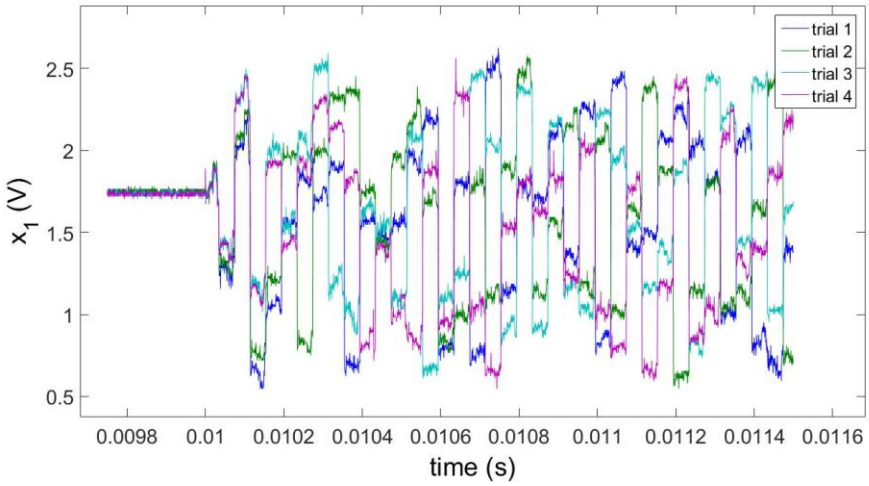


Figure 4.10 Output sequences of cell 1 in different trials.

4.4 Continuous-Input Random Sampling

To realize the RS block, two approaches have been implemented. This section describes the first version of the RS block [82], which uses the continuous-value signal as the input probability distribution, i.e. the time course of the input probability distribution is a continuous value. In Section 4.5, the input signal is changed to the discrete-value input which makes the implementation of the multi-channel RS array easier. The circuit structure of the latter is also partially modified because of the change in the design of the RS block.

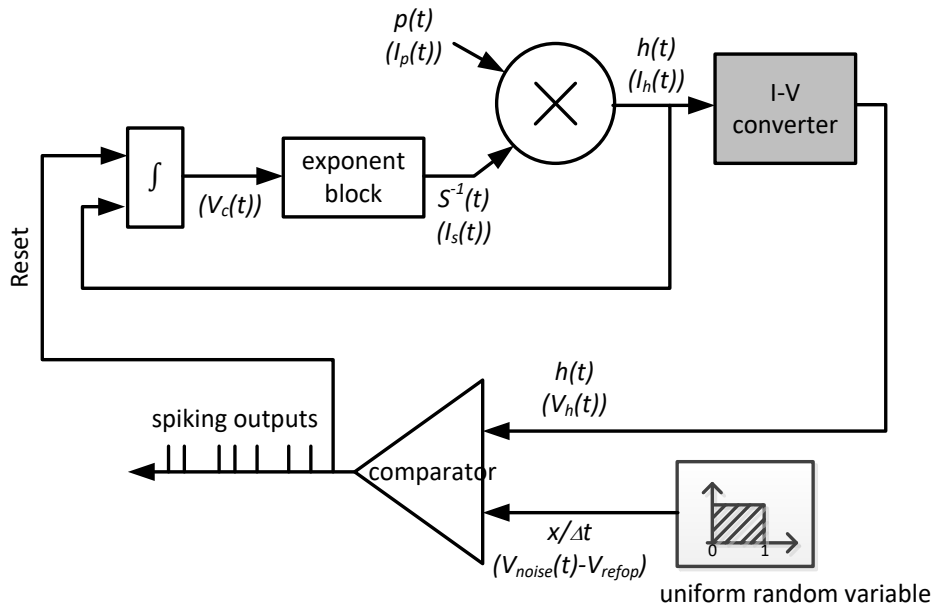


Figure 4.11 Functional implementation of the theoretical continuous-input RS. The variables within parentheses correspond to circuit variables in Figure 4.12. The gray block indicates the additional block needed for the VLSI circuits.

To implement ISI-based random sampling in an aVLSI circuit, the first problem to address is how one can express the terms such as probabilities and distributions (i.e. *p(t)*, *S(t)* and *h(t)*) in (2.10) as corresponding variables in a VLSI circuit. As Section 3.1 shows, in order to make the discrete-time approximation valid, it is recommended to either use the original definition or the continuous recursive form in (2.10) for the hazard update. However, the original definition of the hazard requires a division. To realize such a mathematical operation in a hardware is challenging. Instead, the continuous recursive form can be implemented by an analog circuit. To simplify the circuit design, the input probability distribution *p(t)*, the hazard *h(t)* and the inverse survivor function $S^{-1}(t)$ are all represented as currents $I_p(t)$, $I_h(t)$ and $I_s(t)$ respectively. Figure 4.11 shows the functional blocks and Figure 4.12 shows the transistor circuit schematics of Figure 4.11. The circuit is composed of an integrator, an exponential operator, a current multiplier, a current-to-voltage (IV) converter, a RNG [96], and a voltage comparator. Besides the RNG and the comparator, all of these components serve the purpose of implementing (2.10). The details are described in the following sections.

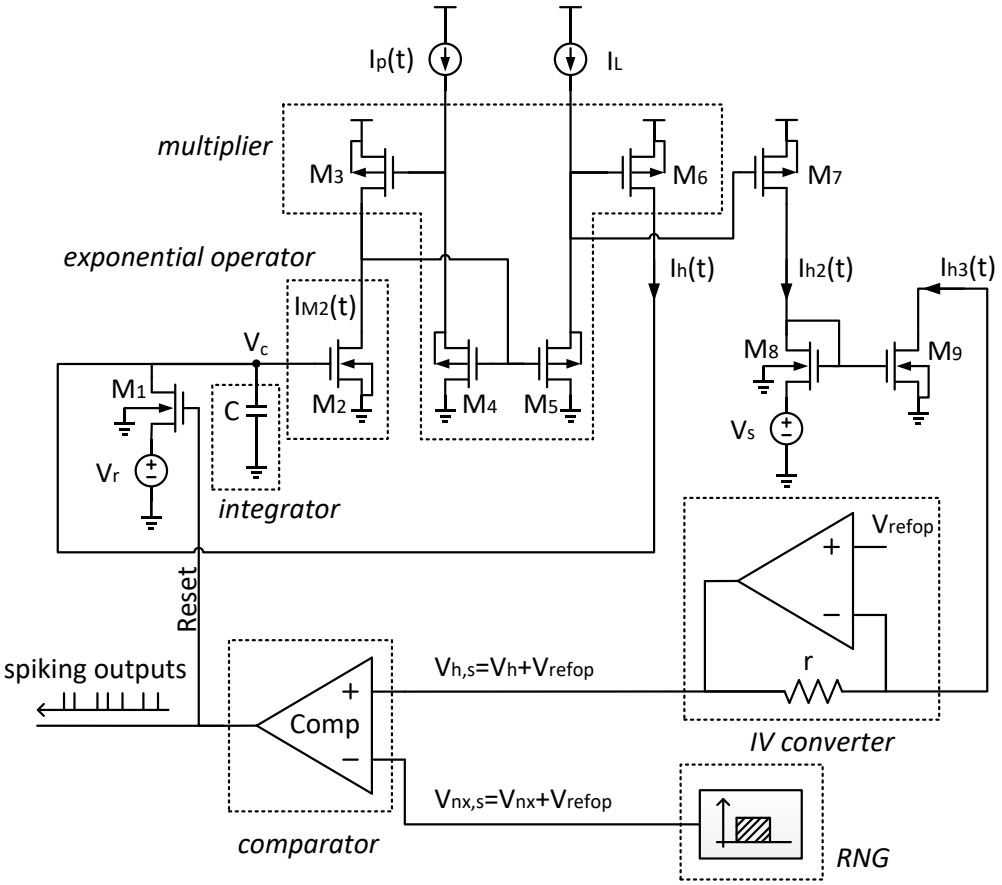


Figure 4.12 Schematics of the continuous-input RS circuit.

4.4.1 Hazard Core

The Hazard Core block includes integrator, exponential operator and multiplier in Figure 4.12. There are several mathematical operations needed in (2.10) that must be physically implemented: Integration, the natural exponential function and multiplication. The hazard is represented by a proportional current $I_h(t)$, whereas $V_c(t)$ represents the integration of $I_h(t)$ over time by a capacitor C. The V_c node drives the gate of transistor M_2 which is operated in the subthreshold regime [97], where there is an exponential relationship between the input gate voltage V_c and the drain current $I_{M2}(t)$. Finally, a current mode translinear multiplier composed of four transistors (M_3 to M_6) [97] generates the product of the time-dependent input current $I_p(t)$ and $I_{M2}(t)$. Note that $I_{M2}(t)$ is the same as the survival current $I_S(t)$. This product becomes the hazard current $I_h(t)$ that charges V_c. The constant current I_L is used to normalize $I_h(t)$.

Transistor M_1 as a switch resets V_c back to a reset value, V_r, once a spike event is generated. A non-zero V_r is necessary to keep the transistors of the multiplier from going out of saturation region right after reset. Another advantage of a non-zero V_r is that it decreases the circuit time constant by bringing the currents in transistors M_2 to M_5 quickly back to their operating ranges. Therefore, the hazard function is implemented by a circuit block consisting of a capacitor C, six transistors (M_1 to M_6), along with a V_r bias and a current bias I_L. Mathematically, $I_h(t)$ can be expressed as (4.2), where κ_{M2}, $I_{0,M2}$ represent the gate-coupling coefficient and off current of transistor M_2 in subthreshold respectively, and U_T is the thermal voltage.

$$I_h(t) = \frac{I_p(t) \cdot I_{0,M2}}{I_L} \cdot \exp\left(\frac{\kappa_{M2}}{U_T} \left(V_r + \frac{1}{C} \int_0^t I_h(t')dt' \right) \right) \tag{4.2}$$

To show its similarity with (2.10), this equation can be further reduced as (4.3), where α, β and I_1 are represented in (4.4) to (4.6).

$$\beta I_h(t) = \alpha I_p(t) \cdot \exp\left(\int_0^t \beta I_h(t')dt' \right) \tag{4.3}$$

$$\alpha = \frac{\kappa_{M2} I_1}{C U_T I_L} \tag{4.4}$$

$$\beta = \frac{\kappa_{M2}}{C U_T} \tag{4.5}$$

$$I_1 = I_0 \exp\left(\frac{\kappa_{M2} V_r}{U_T} \right) \tag{4.6}$$

Hence, the probability density $p(t)$ and the hazard $h(t)$ are equivalent to $\alpha I_p(t)$ and $\beta I_h(t)$ respectively as shown in Table 4.1. Given a desired $p(t)$, the appropriate $I_h(t)$ is given by the straightforward mapping from (2.10) to (4.3). This, however, requires a determination of α, which is difficult, because κ_{M2} cannot be reliably measured in the fabricated circuit. Hence, in Section 4.4.4.1 a method that allows for an indirect determination of α is presented based on the measured output spikes.

Math. Symbol	Phys. Symbol	Parameter
p	αI_p	$\alpha = \frac{\kappa_{M2} I_1}{C U_T I_L}$
h	βI_h	$\beta = \frac{\kappa_{M2}}{C U_T}$

Table 4.1 Mapping the mathematical variables on the hardware

4.4.2 IV Converter

Because the output of the RNG is a voltage and the output of the hazard block is a current, an IV converter is utilized to convert $I_h(t)$ to the voltage $V_h(t)$ so that these two variables can be compared.

First, the current $I_{h2}(t)$ flowing through transistor M_7 in Figure 4.12 has the same magnitude as $I_h(t)$ assuming no transistor mismatch. This current is further mirrored as a current sink through transistors M_8 and M_9 before it is converted to a voltage $V_h(t)$ using an op-amp with a feedback resistor r. In addition, the current $I_{h3}(t)$ to the converter can be an amplified copy of $I_{h2}(t)$ by increasing the bias voltage V_s. In subthreshold regime of transistor M_8 and M_9, the relation of two currents is shown in (4.7). Therefore, $V_h(t)$ can be expressed by V_s, r, $I_h(t)$ (assume $I_{h2} = I_h$ neglecting the transistor mismatch) in (4.8). The purpose of including V_s is to adjust the effective resistance R_{eq} after fabrication, rather than using the fixed resistance r:

$$I_{h3}(t) = I_{h2}(t) \cdot \exp\left(\frac{V_s}{U_T}\right) \tag{4.7}$$

$$\begin{aligned} V_h(t) &= I_h(t) \cdot r \cdot \exp\left(\frac{V_s}{U_T}\right) \\ &= I_h(t) \cdot R_{eq} \end{aligned} \tag{4.8}$$

Because of the reference voltage V_{refop}, applied at the positive input of the op-amp, the output of the converter is a shifted voltage $V_{h,s}(t) = V_{refop} + V_h(t)$. This offset V_{refop} is effectively cancelled by setting the minimum value of the RNG output $V_{nx,s}(t)$ to V_{refop}. In a RNG (see Section 4.3.1), the possible value of $V_{nx,s}(t)$ is in the range of $[V_{RNGn}, V_{RNGp}]$, which is determined by a lower bias V_{RNGn} and a higher bias V_{RNGp}. In this case, $V_{RNG,n} = V_{ref,op}$. Note that $V_{nx}(t)$ is the value without considering the offset,

i.e. $V_{nx}(t) = V_{nx,s}(t)-V_{refop}$. Therefore, the possible value of $V_{nx}(t)$ ranges between $[0,V_{nx,max}]$, where $V_{nx,max} = V_{RNGp}-V_{RNGn}$.

4.4.3 Comparator

The comparator is an open-loop two-stage op-amp. Once $V_{h,s}(t)$ is higher than $V_{nx,s}(t)$, a spike is generated and the system is reset. The pulse width of the spike defines the settling time of the entire circuit during reset. The settling time is determined by the time taken for the output of the multiplier, i.e. V_c, to return to its initial value during reset. Therefore, the pulse width has to be adjustable in the comparator. The probability of generating an event by comparing the outputs of the IV converter and the RNG is shown in (4.9).

$$\Pr(\text{spike at time t}) = p(V_h(t) > V_{nx}(t)) = \frac{V_h(t)}{V_{nx,\max}} \tag{4.9}$$

In Section 2.2.2, the discrete-time approximation is used to generate spikes as shown in (2.12). The probability of generating a spike event is $h(t)\Delta t$. Compare to (4.9), the mathematical and physical relation is shown in (4.10). Then, (4.11) is derived from (4.8), (4.10) and Table 4.1. This equation shows an important message that V_s cannot be adjusted arbitrarily once Δt and $V_{nx,max}$ are determined. Otherwise the output ISI distribution does not approximate the input.

$$h(t)\Delta t = \frac{V_h(t)}{V_{nx,\max}} \tag{4.10}$$

$$R_{eq} = r \cdot \exp\left(\frac{V_s}{U_T}\right) = \beta \cdot \Delta t \cdot V_{nx,\max} \tag{4.11}$$

4.4.4 Measurement Results

The design of the continuous-input RS circuit in an aVLSI chip was done prior to the design of the hardware system described in Section 4.1. The purpose is to ensure that the analog VLSI circuit performs as expected and is feasible to implement as a multi-channel RS chip. Therefore, the circuit in Figure 4.12 was fabricated in a 0.35 um 2-poly 4-metal CMOS technology before the hardware system. The microphotograph of the chip is shown in Figure 4.13. Several test results are presented in this section. First, the parameter α defined in (4.4) is measured and the output of the RNG is characterized. Then, the circuits' ability to sample reliably from

two different input distributions *p(t)* that were specified through different externally applied input currents $I_p(t)$ is demonstrated.

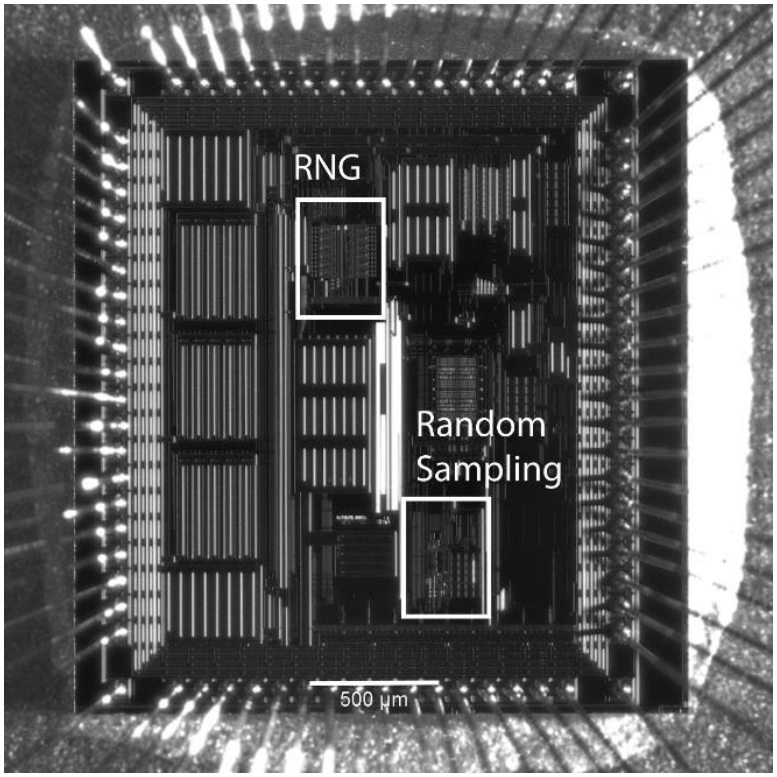


Figure 4.13 Microphotograph of a test chip which holds various test circuits not used in this work. The RS and RNG circuits are outlined in white rectangles.

4.4.4.1 Parameter α

Once a desired input probability distribution *p(t)* has been chosen, the scale of the corresponding input current $I_p(t)$ is determined by the choice of α. This section presents a method for determining α based on the measured output spikes.

The case of a constant input current I_p corresponds to a uniform input distribution *p(t)* with a constant value p_c as shown in Figure 4.14. Because the area underneath the curve has to be 1, p_c is given by $1/t_{ISImax}$, where t_{ISImax} indicate the maximum ISI time in seconds, i.e. $ISI_{max} \times \Delta t$. For a value of I_p, t_{ISImax} can be measured from the output by setting V_{nx} directly to $V_{nx,max}$. By doing so, it is guaranteed that all ISIs of the output spikes should be t_{ISImax} and hence the value of t_{ISImax} has been measured.

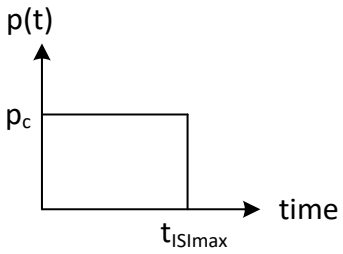


Figure 4.14 Uniform input distribution *p(t)* by providing constant input current I_p. The constant value of *p(t)*, i.e. p_c, is equal to $1/t_{ISImax}$ following the rule that the total area underneath the curve is 1.

By sweeping through various I_p values while measuring the corresponding t_{ISImax}, p_c vs I_p can be plotted as shown in Figure 4.15. Based on the relation that $p(t) = \alpha Ip(t)$, the slope of this curve is equal to the value of parameter α and can be determined by linear regression. The result of the described procedure is $\alpha = 4\times10^9$ 1/C at V_r = 50 mV as shown in Figure 4.15.

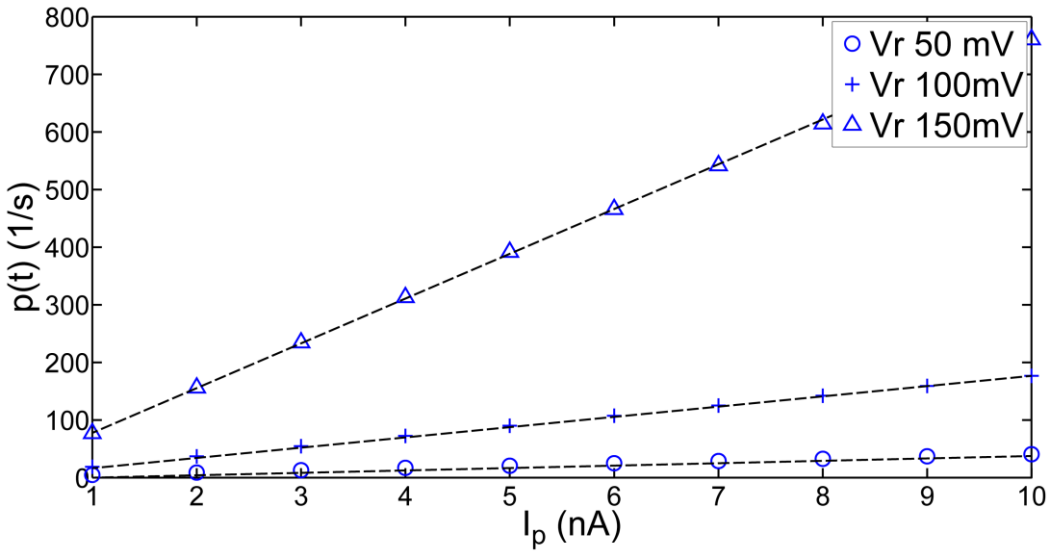


Figure 4.15 Measured *p(t)* (= p_c) vs I_p for a constant I_p. The slope of each curve as extracted from the fit (dotted line) denotes the factor α. Here, α values are extracted for V_r = 50, 100, 150 mV.

4.4.4.2 Empirical Output ISI Distribution

The results from specifying two different input distributions are presented. The first is a uniform distribution and the second is an exponential distribution. Table 4.2 provides an overview over the used values of the adjustable circuit parameters together with the sizes of various circuit components. For the circuit consisting of transistors M_3 to M_6 in Figure 4.12 to act as a multiplier, these transistors need to be operated in the subthreshold regime where the current is exponential to the gate voltage. The currents in this region is usually in the pico to nano amp range. Therefore, in this case, $I_p(t)$ is restricted to a current range of (< 100nA).

Symbol	Block	Value	Unit
$I_p(t)$	Hazard Core	<100	nA
C	Hazard Core	12	pF
V_r	Hazard Core	0.05	V
V_{refop}	IV Converter	~1	V
V_s	IV Converter	~0.18	V
r	IV Converter	500	kΩ
V_{RNGp}	RNG	2.6	V
V_{RNGn}	RNG	1	V
$V_{nx,max}$	RNG	1.6	V
Δt	RNG	64	us

Table 4.2 Physical values of the components and the parameters

Because in the circuit the shape of an input distribution is defined by a current time course, a uniform distribution corresponds a constant input current. For I_p = 1.25 nA, the ISI probability distributions of the input, $p(t)$, and output, $p_{ISI}(t)$, are both shown in Figure 4.16(a). The red line shows the input distribution $p(t)$, while the empirical output distribution $p_{ISI}(t)$ is displayed as blue histogram bars, which were obtained by normalizing the ISI histogram count. Given $\alpha = 4\times10^9$ 1/C from the slope in Figure 4.15 with V_r = 50 mV, $p(t)$ can be predicted to be αI_p = 5 s^{-1}. Because the integration of $p(t)$ over time should be 1, t_{ISImax} should be 0.2 s, which corresponds to a firing rate $v = 2/t_{ISImax}$ = 10 Hz. In another trial based on the uniform distribution, the input current I_p was set to 2.5 nA, which is equivalent to a predicted t_{ISImax} = 0.1 s and v = 20Hz. The resulting $p(t)$ and $p_{ISI}(t)$ are shown in Figure 4.16(b).

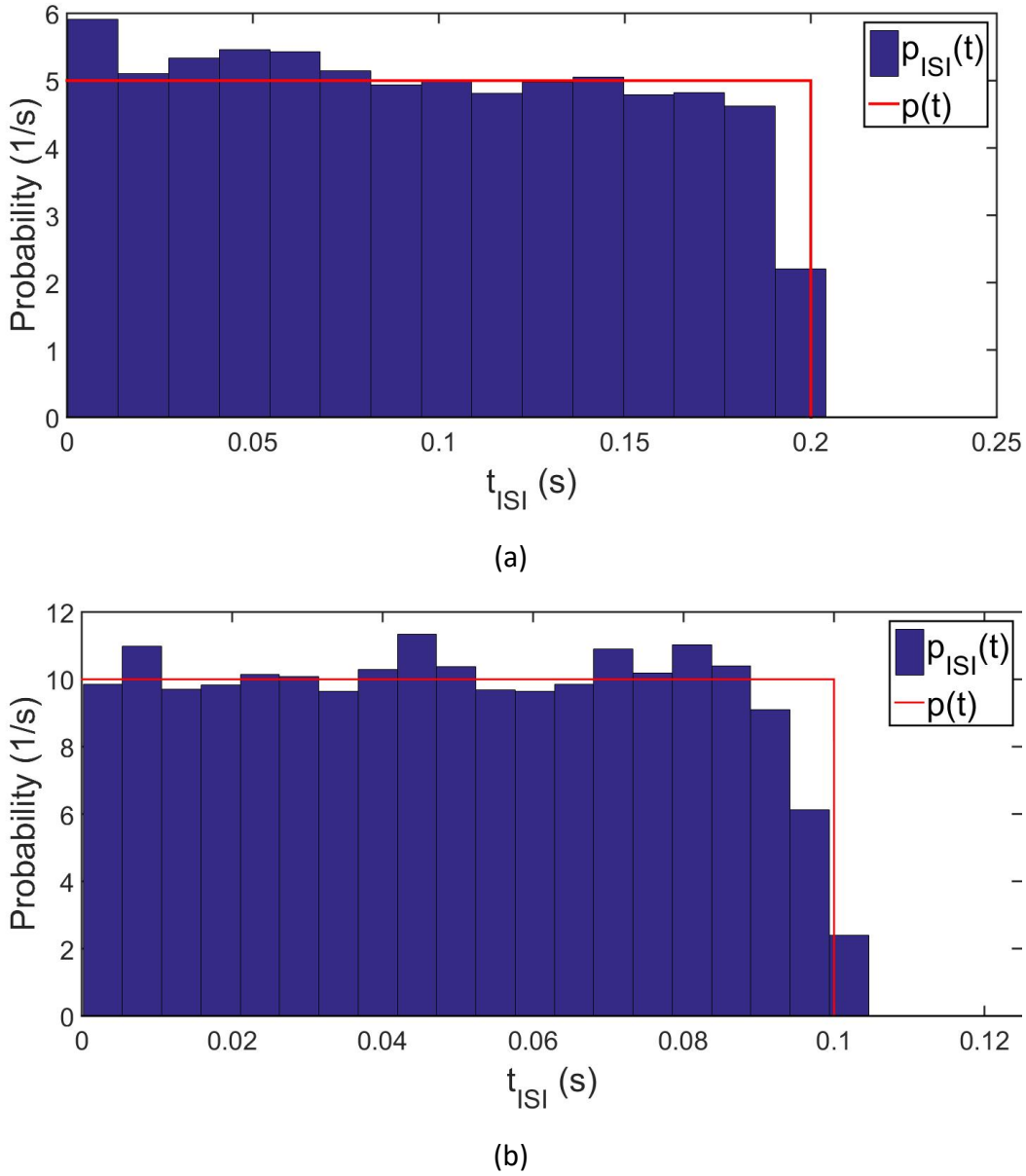
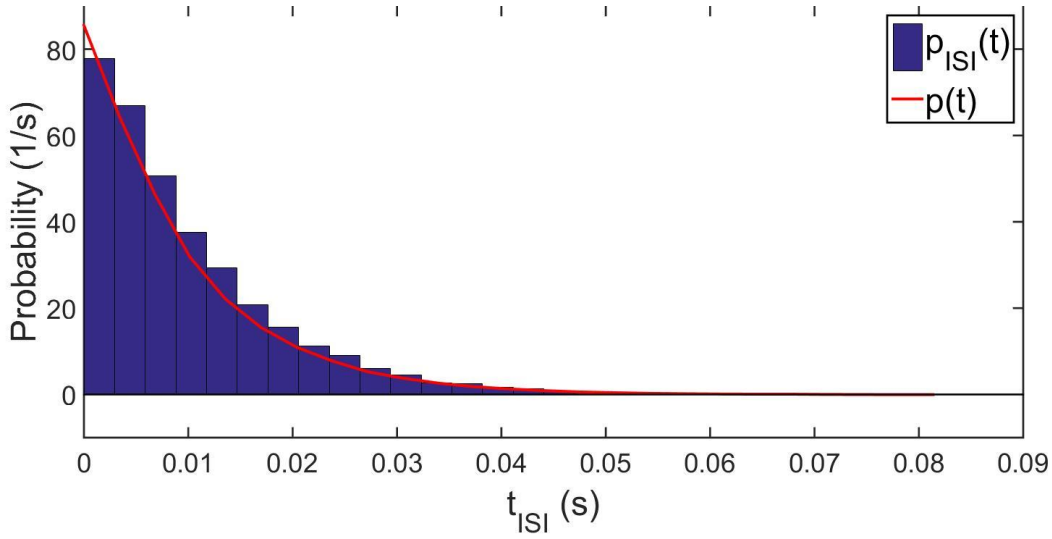


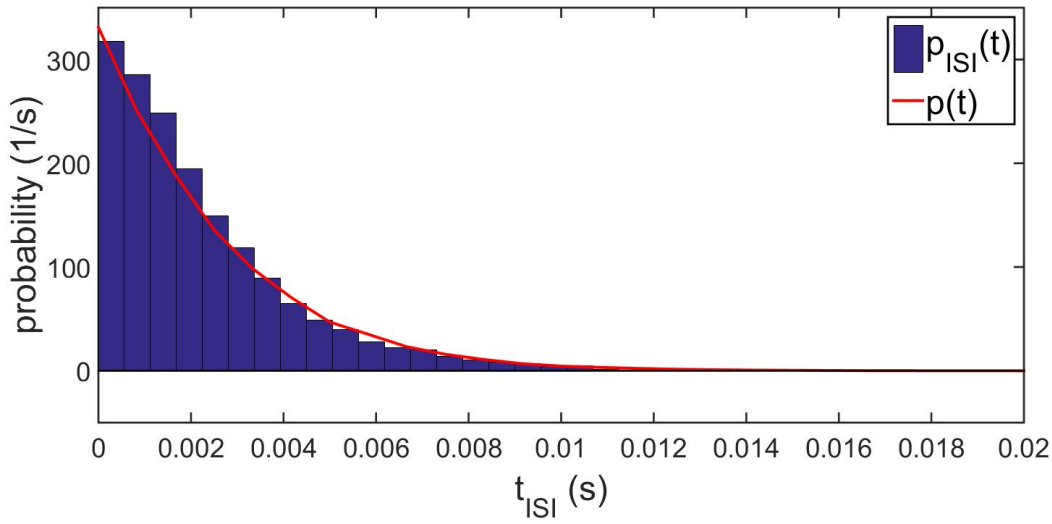
Figure 4.16 Two output ISI distributions for two different input currents. The red line represents the input distribution *p(t)*. The input current I_p is (a) 1.25 nA and (b) 2.5 nA.

In a second set of experiments, the input currents were based on decaying exponentials rather than constants. The initial value $I_p(0)$ was first set to 21.4 nA which, using $\alpha = 4\times10^9$ 1/C, should induce a $p(0) = \nu = 86$ s^{-1} (see Figure 4.17(a)). On the other hand, setting $I_p(0) = 82.8$ nA in the second experiment predicts $p(0) = \nu = 311$ s^{-1}, the results of which are shown in Figure 4.17(b). Since $I_p(0)$ is larger during

the second experiment, the exponential decay time constant $\tau = 1/p(0)$ in Figure 4.17(b) is smaller compared to that of Figure 4.17(a).



(a)



(b)

Figure 4.17 The exponential probability distribution of the output ISIs. The initial input current $I_p(0)$ is (a) 21.4 nA and (b) 82.8 nA. The equivalent input distributions $p(t)$ are plotted in red.

Because more complex inputs could not be generated using the current experimental setup, the circuit simulations is performed based on a bimodal $I_p(t)$ (see Figure 4.18, red line). In this case, the induced output ISI distribution $p_{ISI}(t)$ can be

computed from the simulated hazard current $I_h(t)$. In turn, the corresponding $I_{p,rec}(t)$ (blue line) can be reconstructed from $p_{ISI}(t)$ and compared with the actual input distribution $I_p(t)$. More specifically, the reconstructed current $I_{p,rec}(t)$ was given by $I_{p,rec}(t) = p_{ISI}(t)/\alpha$, where $p_{ISI}(t)$ was obtained from a numerical evaluation based on the hazard function $h(t) = \beta I_h(t)$. Rather than collecting actual, simulated ISIs in a histogram, such a procedure was necessary due to the lengthy simulation time needed to collect enough spikes for computing the output ISI distribution. As Figure 4.18 shows, the reconstructed $I_{p,rec}(t)$ displays a similar waveform as the original input $I_p(t)$.

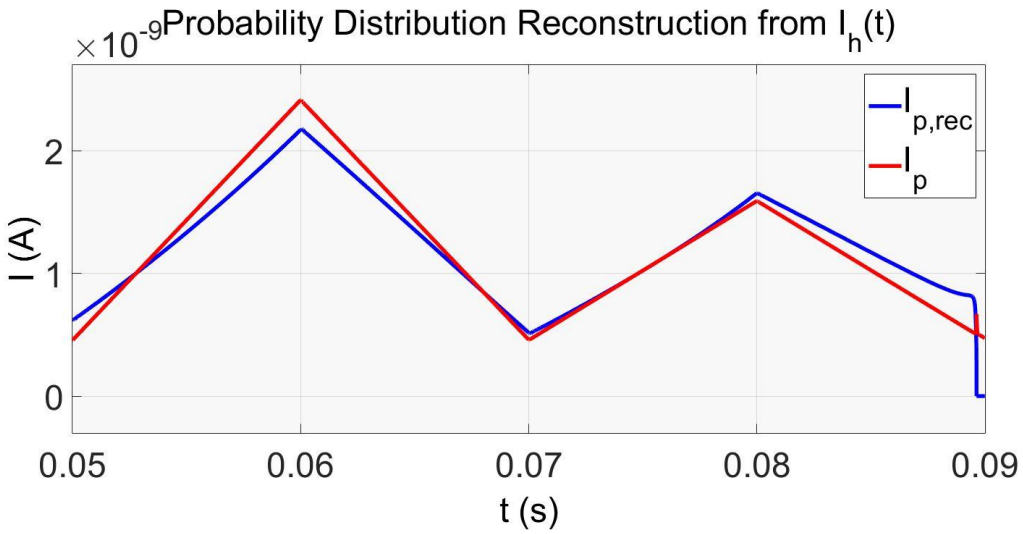


Figure 4.18 Simulation results showing the reconstruction of a more complex probability distribution from the hazard circuit.

4.5 Discrete-Input Random Sampling

When using the continuous-input RS circuit described in Section 4.4 as one channel to create a multiple-channel array, some difficulties were noticed. First, the linear range of the multiplier composed of a translinear loop with four transistors operated in the subthreshold regime (see Figure 4.12) is hard to define. To ensure multiplication holds, the input current I_p has to be small in order to make the gate-source voltage $V_{gs,M4}$ lower than the threshold voltage $V_{th,M4}$. However, $V_{th,M4}$ which depends on the transistor doping varies from one channel to another. A general range of I_p across channels is hard to find. The threshold voltages of the other three

transistors also have similar problems. Namely, the current multiplier that fulfills $I_{M2} \times I_p = I_L \times I_h$ is only valid in a certain current range. In [17], the translinear loop is used for the current multiplier in 64 channels. However, [17] limits the current ratio such as I_p/I_L in a small range of 1.11 and change the transistor sizes every four channels. In this way, the current multiplier can be guaranteed to function as expected. As the input of the RS circuit, I_p represents a regularly-step-staircase (RSS) input probability distribution $p_{in}(t)$ which is unable to be limited in a small range. The second difficulty is that an extra digital-to-analog converter (DAC) for I_p is required. The LS channels implemented on the FPGA send out $p_{in}(t)$ as a sequence of bits following the time step Δt (Section 4.2). Therefore, an extra DAC for each channel is required to generate the analog current I_p from digital bits. This DAC consumes chip resources. This chapter presents a modified version of the RS circuit to address the two described difficulties so that a multi-channel implementation is feasible. In addition, the channel AER in each RS channel is placed so that the channel can communicate with chip-level AER block (see Figure 4.1). The circuit blocks are shown in Figure 4.19.

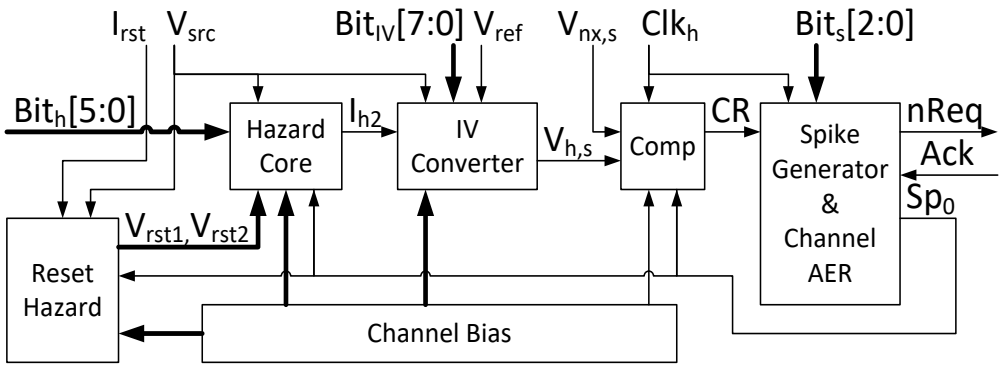


Figure 4.19 Structure of one RS channel

The input of the RS circuit comes from the **Data$_Z$** module of the LS (see Figure 4.2). The 6-bit data in each bin is represented as $Bit_h[5:0]$ in Figure 4.19. Similar to the structure of the continuous-input RS, the circuit includes a hazard function generator, a comparator, and a reset, which are implemented by the **Hazard Core**, **Comp** and **Reset Hazard** blocks (see Figure 4.19) respectively. Because the output of the **Hazard Core** is a current in the aVLSI implementation while the comparator in the **Comp** block is a voltage-input comparator, an **IV Converter** block is needed. In addition, a channel AER represented as the **Spike Generator & Channel AER** block in each RS channel is also required. This block not only communicates with the **Chip AER Transmitter** block, it also controls the pulse width of the feedback spike signal

Sp_0 that defines the settling time of the entire circuit during reset The details of blocks are explained as following subsections.

4.5.1 Hazard Core

This section describes the circuit modifications which improve on the multiplier circuit and also avoid the use of the extra DAC. In the continuous-input RS circuit, the input current I_p(t) can be a continuous-value signal. However, such flexibility is unnecessary because the LS and RNG circuits are time stepped; that is, their outputs are both discrete values that are updated on every $i\Delta t$, where $i \in \{0,1,2,\ldots\}$. In addition, note that $I_p(t)/I_L$ in (4.2) can be treated as a time-varying unit-less factor $N(t)$. The input probability distribution is therefore represented by $N(t)$ instead of $I_p(t)$. The data from the LS FPGA block, i.e. $Bit_h[5:0]$ in Figure 4.19, can be directly used as a digital input of the multiplier. The structure of the multiplier is modified from a translinear loop circuit to a switch-controlled current-mirror array (also called a current-mode DAC). Using the current-mirror array, the reliability of the multiplier is improved and the input range can be defined easily. The extra DAC is also now unnecessary. In this approach, the hazard is still updated using the continuous recursive form while the input is provided as discrete values, i.e. $p_{in}(t)$ is a RSS probability distribution, as the condition in (A.6). The circuit details of the new **Hazard Core** block are shown in Figure 4.20. Similar to the structure of the continuous-input RS, capacitor C_1 is used for the integration term and transistor M_{a1} operates in the subthreshold regime to fulfill the exponential term.

The current-mode 6-bit DAC consists of switches S_{h5} to S_{h0} and transistors M_{b1} to M_{b9}. The transistor size ratios of transistor M_{b1} to M_{b9} with the same length are [2:32:16:8:4:2:1:1:1]. The switches are controlled by a 6-bit RS input $Bit_h[5:0]$. The currents in the branches with high input bits of Bit_h sum into a current splitter composed of transistors M_{c1} to M_{c4}. The remaining currents flow through transistor M_{c5}. When all input bits are low ($Bit_h = 0$), the summed current, I_{hsum}, to the splitter shrinks down to off-current level, resulting in a large time constant and slowing the speed. Transistor M_{b8} provides a offset current to the summed current to prevent this situation. Transistor M_{b9} is also added to deal with the case when all input bits are high. Thus, including the default current from transistor M_{b8}, the possible 6-bit values are shifted from [0,63] to [1,64]. That is, $N(t) = Bit_h(t)+1$. In Section 4.2, it is described that the **Data$_Z$** module in the LS channel only sends out non-zeros values stored in the **Mem M$_Z$** module. The reason for this scheme is to compensate for the shifted value of $N(t)$. For example, if there is a value “29” in the **Mem M$_Z$** module,

the **Data$_Z$** module will send out the value of "28" as Bit_h such that $N(t)$ will restore the original value of "29".

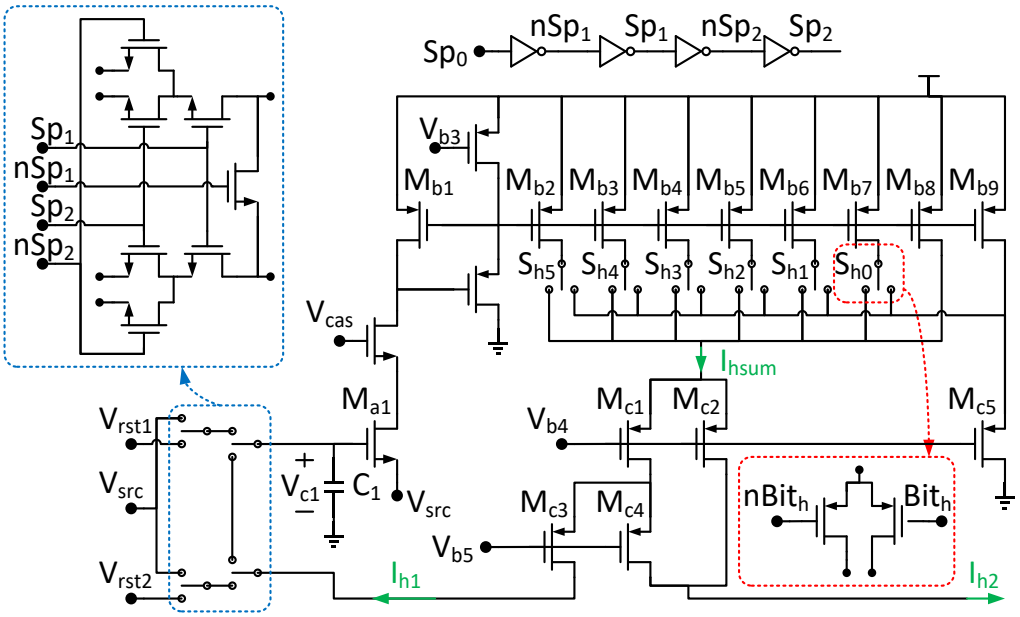


Figure 4.20 CMOS circuit details of the Hazard Core block

The current splitter ratio of the currents through transistors M_{c1} and M_{c2} and through transistors M_{c3} and M_{c4} is [1:7]. The summed current, I_{hsum}, is divided into I_{h1}, the feedback current for integration, and I_{h2} for the **IV Converter** block. Due to the current splitter ratio, I_{h2} is 63×I_{h1}. This formulation simplifies the design of the **IV Converter** block which will be explained in Section 4.5.2.

The other switch group in the blue box of Figure 4.20 controls the integrated voltage on C_1, i.e. V_{c1}, and therefore the drain current of transistor M_{a1}. When there is no spike, I_{h1} charges C_1 continuously. Once a spike, Sp_0, happens through the **Spike Generator & Channel AER** block, the feedback loop is opened through Sp_1, Sp_2, nSp_1 and nSp_2. The latter signals are derived from the Sp_0 using inverters with different timing delays in order to minimize the charge injection effect on C_1. As a result, V_{c1} is reset to V_{rst1} and I_{h1} is shorted to V_{rst2}, provided from the **Reset Hazard** block described in Section 4.5.5. Both V_{rst1} and V_{rst2} are generated by an input reset current I_{rst} as depicted in Figure 4.24. Therefore I_{rst} sets the reset value of V_{c1}, and therefore the initial drain current of M_{a1}. A small initial current corresponds to a longer integration time to reach the same current value. Also, a small input N results in a smaller I_{hsum} and a longer time to reach the same current value. Therefore, t_{ISImax} (= ISI_{max}×Δt) is defined both by I_{rst} and N.

In this design, I_{rst} is set to ~256 pA so that I_{h1}, at N = 1, is only ~2 pA. This setting not only reduces the final power consumption of the circuit but keeps V_{c1} lower than the threshold voltage of transistor M_{a1} before reset. This condition guarantees that the subthreshold voltage-to-current exponential equation is valid. In addition, an offset voltage, V_{src}, is added to decrease the effect of the off currents from the transistor switches during integration and to increase the accuracy of the initial current of M_{a1} during reset [98]. The hardware mapping equation of the circuit in Figure 4.20 is shown in (4.12), where κ_{Ma1} is the gate-coupling coefficient of M_{a1}, $N(t)$ denotes the dynamic input in the range of [1,64] and the denominator of 128 comes from the transistor sizing ratio of M_{b1} to M_{b9} and the current splitter. I_{h1} and N are proportional to h and p_{in} respectively through (4.13) and (4.14). Note that $N(t)$ is updated from the LS channel on every $i\Delta t$. As shown in (4.14), for a fixed value of N, the numerical value of the probability p_{in} can still vary depending on I_{rst}. The smaller I_{rst}, the smaller the p_{in} is, therefore the larger $t_{ISI,max}$ is. This also explains why $t_{ISI,max}$ is defined by both I_{rst} and N from the theoretical perspective.

$$I_{h1}(t)=\left(\frac{I_{rst}}{128}\right)\cdot N(t)\cdot\exp\left(\frac{\kappa_{M_{a1}}}{C_1U_T}\int_0^t I_{h1}(t')dt'\right) \tag{4.12}$$

$$h(t)=\frac{\kappa_{M_{a1}}}{C_1U_T}\cdot I_{h1}(t) \tag{4.13}$$

$$p_{in}(t)=\frac{\kappa_{M_{a1}}}{C_1U_T}\cdot\frac{I_{rst}}{128}\cdot N(t) \tag{4.14}$$

4.5.2 IV Converter

As mentioned in Section 2.2, in order to generate random spikes, the value of h has to be compared to samples in the range of [0,1/Δt] drawn from a uniform distribution. These samples are represented as a voltage V_{nx}, which is updated on every $i\Delta t$. On the chip, I_{h1} needs to be converted to a voltage through a given resistance R_{eq} for comparison to V_{nx}. However, implementing a physical linear resistor on the chip for I_{h1} is infeasible because of the large value of R_{eq}. For example, given a constant input N, the maximum hazard current of I_{h1} in (4.12) can be computed as (4.15). This maximum value happens when V_h (= $I_{h1}\times R_{eq}$) reaches $V_{nx,max}$ whereupon a spike is generated. The ISI value carried by this spike is then t_{ISImax}. If the condition that [N,ISI_{max},I_{rst}] = [64,16,256 pA] is set, then $I_{h1,max}$ = 2048 pA according to (4.15). Assume V_{nx} ranges from [0,1.25 V] so $V_{nx,max}$ is 1.25 V. The required R_{eq} as calculated by $V_{nx,max}/I_{h1,max}$, is equal to 610 MΩ.

$$I_{h1,\max} = I_{h1}\left(t_{ISI\,max}\right) = \left(\frac{I_{rst}}{128}\right) \cdot N \cdot ISI_{\max} \tag{4.15}$$

Such large R_{eq} is difficult to realize physically in a VLSI chip. The required resistance, however, can be reduced by amplifying I_{h1}. It is similar to what was done in the continuous-input RS circuit. The scheme in that is to provide a source voltage V_s for a current mirror as shown in Figure 4.12 so that I_{h2} can be amplified by adjusting V_s. However, the gain of the amplified current varies between channels due to process mismatch. To realize a multi-channel RS array, it is inconvenient to have a separate voltage source per channel. Therefore, this section describes another solution that not only reduces the value of the resistor that is physically implemented on the chip but allows to calibrate the mismatch of the resistor channel by channel.

The current splitter, mentioned in Section 4.5.1 creates a current with an amplification of I_{h1}, by first making $I_{h2} = 63 \times I_{h1}$. Then I_{h2} is amplified further by 32x through a current mirror circuit leading to a resistor R_1 of only 340 kΩ, which occupies a layout area of 71×28 um^2. To calibrate the resistance variation due to process mismatch, the output transistor of the 32x current mirror circuit is divided, as shown in Figure 4.21, into several transistor branches which can be turned on or off by switches S_{IV7} to S_{IV0}. The ratio of M_{d1} to M_{d10} with the same length is [4:128:64:32:16:8:4:2:1:1]. $Bit_{IV}[7:0]$, shown in Figure 4.19, represents the 8 bits to control the switches. The amplified current, I_{hamp}, is converted to a hazard voltage, V_h, by the physical resistor R_1. That is, $V_h = I_{hamp} \times R_1 = I_{h2} \times (1/4) \times Bit_{IV} \times R_1 = I_{h1} \times (63/4) \times Bit_{IV} \times R_1$, where the value of 63 is contributed by the current splitter in the **Hazard Core** block and the factor of 4 is from the transistor sizing ratio of M_{d1} to M_{d10}. Theoretically, $R_{eq} = 63 \times 32 \times R_1$, where $Bit_{IV} = 128$.

In Figure 4.21, the output voltage $V_{h,s}$ is compared with $V_{nx,s}$, a voltage-shifted version of V_{nx}, in order to ease the design complexity of the RNG circuit and the comparator. By definition, a spike is generated when $V_h > V_{nx}$. However, $V_{h,s}$, used for comparison, is not only a shifted signal but has an opposite sign to V_h because $V_{h,s} = V_{ref} - V_h$, where V_{ref} is a bias voltage. Theoretically, another current mirror stage is needed to change the direction flow of I_{hamp}. However, adding one more stage not only increases power dissipation but, more importantly, brings in more mismatch. The details of how to use $V_{h,s}$ and $V_{nx,s}$ for comparison without adding one more stage are as follows.

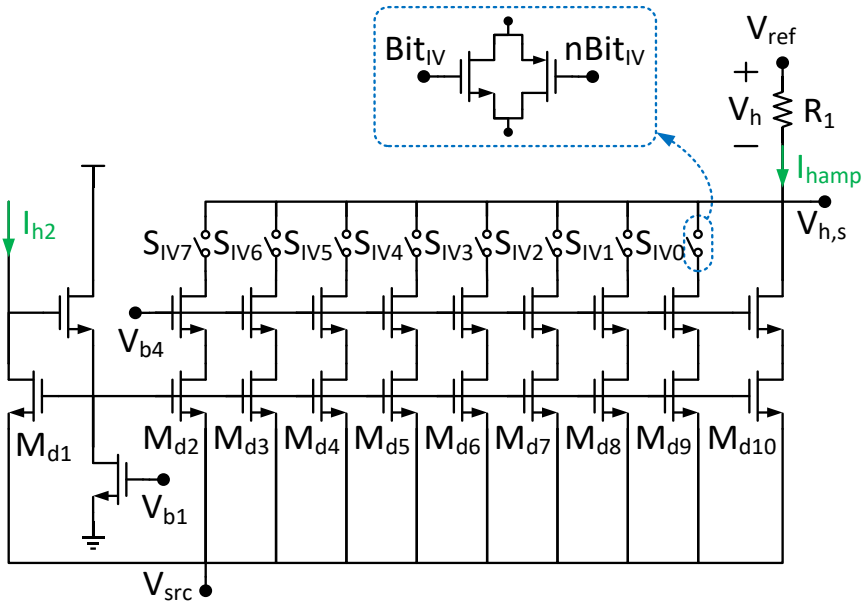


Figure 4.21 CMOS circuit details of the IV Converter block

V_{nx} is generated from a uniform distribution in the range of $[0,V_{nx,max}]$ while the shifted-version $V_{nx,s}$ (the practical output of the RNG) = $V_{nx}+V_{os}$, where V_{os} is an user-defined offset voltage. That is, $V_{nx,s}$ is in the range of $[V_{os}, V_{os}+V_{nx,max}]$. Then, the derivation is given in (4.16). The reason for $Pr(V_h>V_{nx}) = Pr(V_h>V_{nx,max}-V_{nx})$ is that the value of $V_{nx,max}-V_{nx}$ still follows a uniform distribution in $[0,V_{nx,max}]$. Assigning $V_{ref} = V_{nx,max}+V_{os}$ leads to the result in (4.16). According to the equation, the way to generate a spike can be changed to compare $V_{h,s}$ to $V_{nx,s}$. Once $V_{h,s}$ is less than $V_{nx,s}$, a spike is generated. Note that $V_{nx,s}$ is produced by the RNG whose outputs range between $[V_{RNGn},V_{RNGp}]$ (see Section 4.3.1). Therefore, $V_{os} = V_{RNGn}$ and $V_{nx,max} = V_{RNGp}-V_{RNGp}$ and $V_{ref} = V_{RNGp}$.

$$\begin{aligned}\Pr\left(V_h > V_{nx}\right) &= \Pr\left(V_h > V_{nx,\max} - V_{nx}\right) \\ &= \Pr\left(V_{ref} - V_{h,s} > V_{nx,\max} + V_{os} - V_{nx,s}\right) \\ &= \Pr\left(V_{h,s} < V_{nx,s}\right)\end{aligned} \tag{4.16}$$

4.5.3 Comparator

The **Comp** block shown in Figure 4.22 achieves three goals. First, it compares the hazard value $V_{h,s}$ to $V_{nx,s}$. Second, because the random samples are produced on every $i\Delta t$, the comparison result (CR) should be aligned with this time interval. Third,

during the generation of the spike, the block should be disabled. That is, *CR* should stay high when there is a spike.

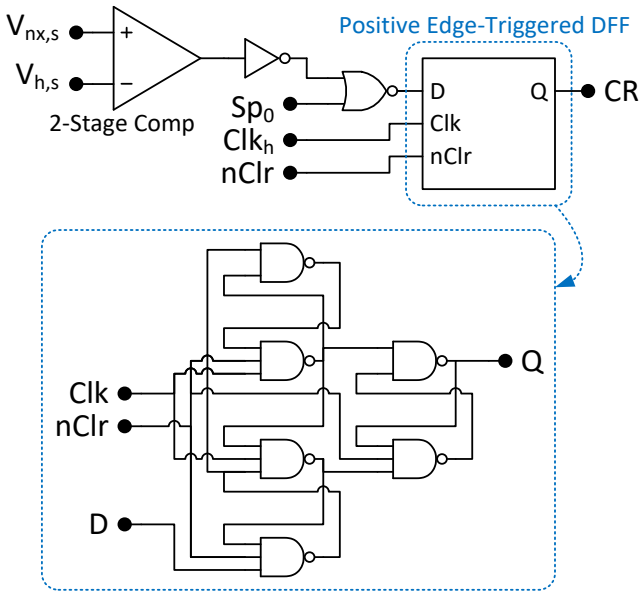


Figure 4.22 Circuit details of the Comp block

The first goal is achieved using a simple two-stage op-amp. As shown in (4.16), the compared result is high when $V_{h,s} < V_{nx,s}$. Therefore, $V_{h,s}$ is connected to the negative terminal of the comparator as shown in Figure 4.22. The second and third goals are achieved by implementing a positive edge-triggered D flip-flop with a clock, Clk_h, and a feedback spike pulse, Sp_0, as inputs. Clk_h is generated from the LS array with a period the same as Δt. The disable period is determined by the pulse width of Sp_0, which is generated from the **Spike Generator & Channel AER** block. The default pulse width is set to $2\times\Delta t$.

4.5.4 Spike Generator & Channel AER

In this block as shown Figure 4.23, the channel AER communicates with the top level **AER Transmitter** block (see Figure 4.1) through the *nReq* and *Ack* signals. The *nReq* signal becomes low active if the output of the **Comp** block (*CR*) is high. The channel AER is also using the same positive edge-triggered D flip-flop in Figure 4.22. The output of the de-multiplexer, Sp_0, has a pulse width that is determined by selecting one of the outputs of six JK flip-flops. These gates produce outputs that are multiples of either 1, 2, 4, 8, 16 or 32 periods of Clk_h. Based on the circuit simulations,

a pulse width of two clock periods is sufficient to allow the voltages and currents of the circuits in the **Hazard Core** block to return to their initial values.

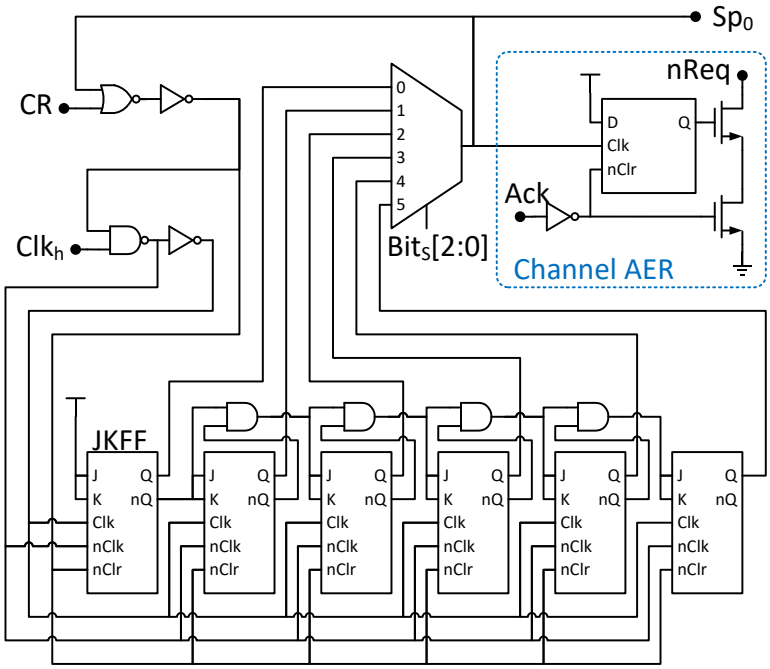


Figure 4.23 Circuit details of the Spike Generator & Channel AER block

4.5.5 Reset Hazard

The **Reset Hazard** block acts to bring V_{c1} and the drain current of transistor M_{a1} in the **Hazard Core** block (see Figure 4.20) back to their initial values. They are controlled by I_{rst} that is generated from an on-chip RS DAC. Each RS channel has a DAC in order to control I_{rst} independently. The structure of the **Reset Hazard** block is shown in Figure 4.24. The two op-amps have the same design with a simple five-transistor structure, where the current source is gate-controlled by Sp_0. If there is no spike pulse, the op-amps are turned off to save power. Transistor M_{a2} is designed to have the same dimension as transistor M_{a1} in the **Hazard Core** block. The voltage source V_{src} is also shared with the same bias in the **Hazard Core** block. Therefore, the initial drain current of M_{a1} should be expected to be I_{rst}. The voltage V_{c1} in the **Hazard Core** block is reset to the initial voltage V_{rst1}. The second reset source V_{rst2} is used to sink the current I_{h1}. Two op-amps are required because if there is only one op-amp, the feedback loop will continue to make I_{h1} increase causing the op-amp to fail in resetting V_{c1}.

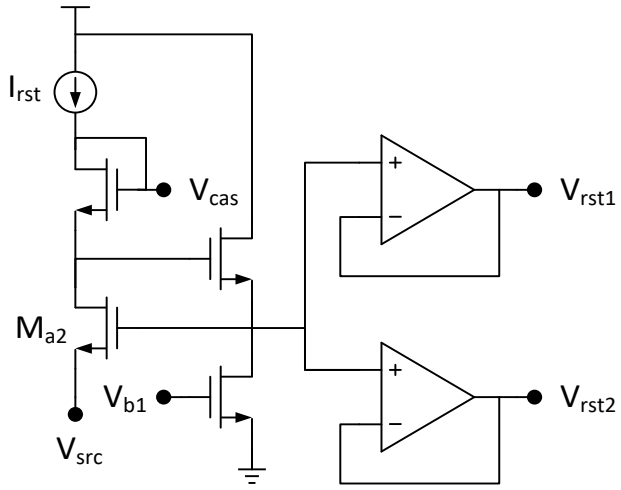


Figure 4.24 CMOS circuit details of the Reset Hazard block

4.5.6 Channel Bias

The **Channel Bias** block shown in Figure 4.25 provides all required biases in the RS channel. Some biases, i.e. V_{b1} and V_{b3}, are generated by a common current source $I_{b,src}$ (Figure 4.25 (a)) from one shared on-chip DAC for all RS channels. The other biases, i.e. V_{b4} and V_{b5}, are generated by a PMOS chain as shown in Figure 4.25(b). There are seven equal-dimension bulk-source-tied transistors connected from V_{dd} to Gnd.

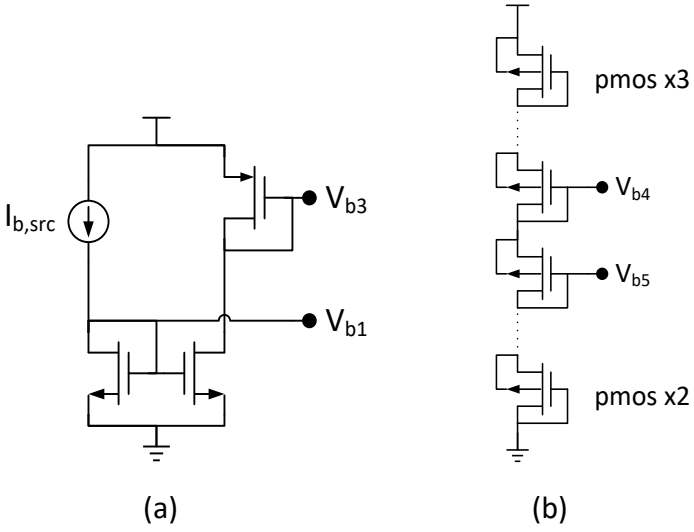


Figure 4.25 CMOS circuit details of the Channel Bias block

Table 4.3 shows the mapping of the parameters from the mathematical values to the physical values in the hardware. Note that the mapping of the probability distribution p_{in} is dependent on both N and I_{rst} as shown in (4.14). The relationship between the hazard value, h, and the hazard current, I_{h1}, is defined in (4.13).

Math. Symbol	Math. Value	Physical Symbol	Physical Value	Unit
p_{in}	[0,1]	$N \times I_{rst}$	$[1,64] \times I_{rst}$	
-		I_{rst}	~300	pA
h	[0,1/Δt]=[0,1]	V_h	[0,1.25]	V
-		$V_{h,s}$	[1.25,2.5]	V
nx/Δt	[0,1/Δt]=[0,1]	V_{nx}	[0,1.25]	V
		$V_{nx,s}$	[1.25,2.5]	V
Δt	1	Δt	64	us
		pulse width of Sp_0	128	us
		Clk_h	0.0156	MHz
		Clk_{main}	0.5	MHz
		Clk_{RNG}	0.5	MHz
		Clk_{config}	1.56	MHz
		C_1	4	pF
		R_1	340	kΩ
		V_{ref}	2.5	V
		V_{src}	0.3	V

Table 4.3 Mapping table of discrete-time RS

4.5.7 Measurement Results

The discrete-input RS circuit is chosen to realize the multi-channel RS ASIC chip. The chip microphotograph is shown in Figure 4.26. Because t_{ISImax} of each channel is controlled by I_{rst} (see Section 4.5.1), 16 on-chip RS DACs is implemented, allowing to tune the I_{rst} value of each individual channel. The remaining subsections present the detailed chip characterization results and measurements from the chip.

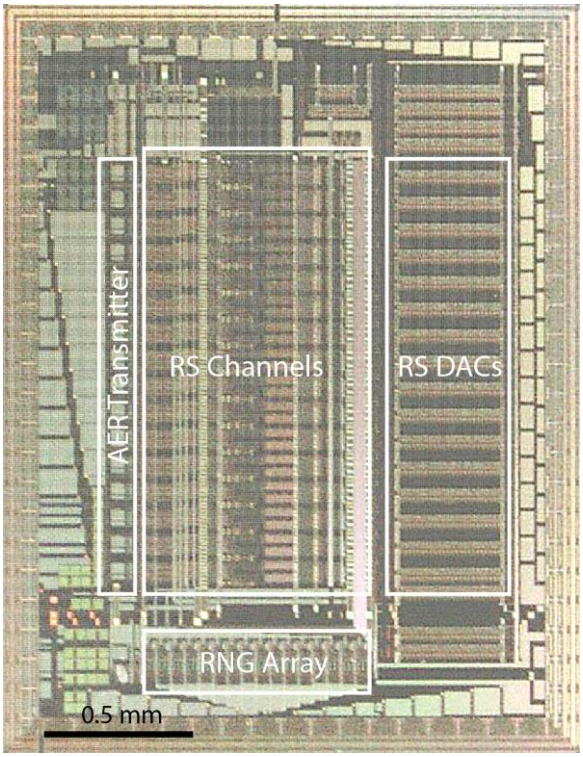


Figure 4.26 Chip microphotograph of the RS array with 16 RS channels, 16 RS DACs for the reset currents, the RNG array for 16 random sources and the top-level chip AER transmitter. The bias generator occupies the remaining area. The chip areas is 2.16×2.74 mm^2

4.5.7.1 Linearity of the Current Mirror Array

The linearity of the current mirror array (or the current-mode DAC) in the **Hazard Core** block is estimated indirectly by measuring the output ISIs. In this experiment, V_{nx} is set by an external voltage source instead of a RNG.

It is known that an output spike is generated when $V_h > V_{nx}$. By assigning a constant value to the input N and assigning V_{nx} to $V_{nx,max}$, the output spikes are fired constantly with a roughly fixed period, i.e. ISIs are constant. The ISI in this case is ISI_{max}. Averaging on these ISIs and changing the input N from 1 to 64 leads to the result as shown in the red cure of Figure 4.27. The average ISI is inversely proportional to N according to (4.15). After taking the reciprocal of the average ISI, the linearity of the current mirror array is shown in the blue curve. As expected, the change of the current multiplier from a translinear loop to a current mirror array results in an improved linearity (i.e. the ratio of the maximum and minimum value of N is increased to 64) and a well-defined input range.

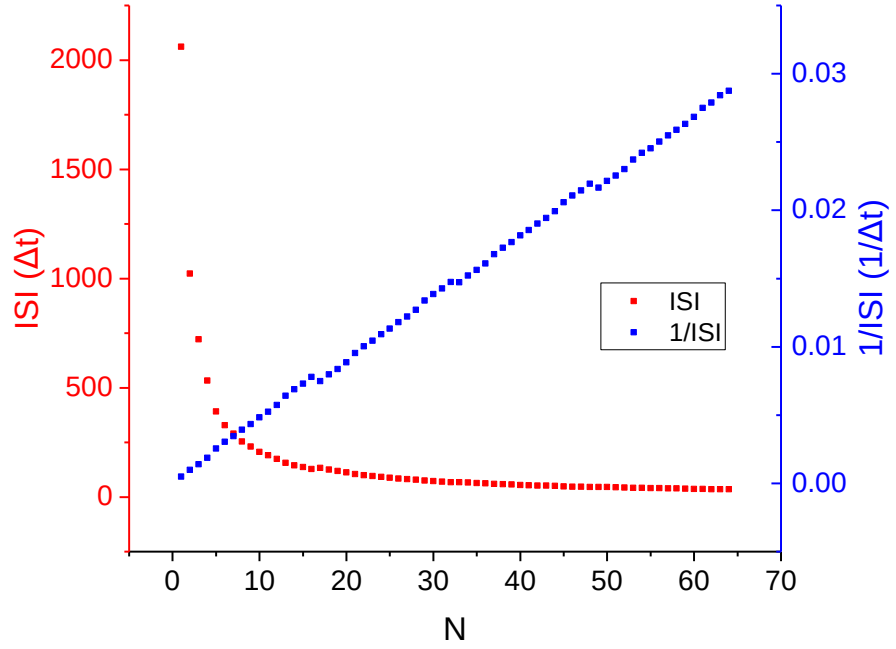


Figure 4.27 Linearity of the current mirror array in the Hazard Core block.

4.5.7.2 Accumulation of the Hazard

This experiment shows how the hazard accumulates in the RS chip given a uniform distribution, which means a constant input *N*, as an example. The relation between the input probability and time in the time-course uniform distribution is shown in Figure 4.14. To generate such distribution, it is only needed to provide a constant input *N* and the system will go back to the initial state by itself once a spike happens. It is not needed to cut the input value when the time reaches t_{ISImax} because a spike must happen no longer than t_{ISImax}.

The value of the hazard *h* can be derived by measuring V_h. This voltage can be indirectly estimated by measuring the output ISIs of one RS channel as a function of V_{nx} as shown in Figure 4.28.Note that in this case V_{nx} is set by an external voltage source instead of a RNG. Because an output spike is generated when $V_h > V_{nx}$, the accumulation time of V_h increases as V_{nx} increases. When a spike is generated, V_h just surpasses V_{nx}, i.e. $V_h \sim= V_{nx}$. Therefore, Figure 4.28 also demonstrates how the hazard accumulates over time in the case of a uniform input distribution. This can be seen by reversing the x and y axes of the plot. The ISI_{max} for each value of *N* happens when V_h reaches $V_{nx,max}$ (= 1.25 V). In Figure 4.28, the ISI_{max} value (~= 32) for $N = 64$ (the red curve) is a half of the one (~= 64) for $N = 32$ (the blue curve) and is a quarter of the

one (~= 128) for N = 16 (the yellow curve) and so on. The experimental result follows the prediction in (4.15) that $N \times ISI_{max}$ is a constant. Note that V_h and V_{nx} are used in this section in order to facilitate the explanation on the accumulation of the hazard instead of using $V_{h,s}$ and $V_{nx,s}$. In fact, $V_{nx,s}$ are provided in this experiment and V_{nx} are obtained by subtract the offset V_{os} from $V_{nx,s}$.

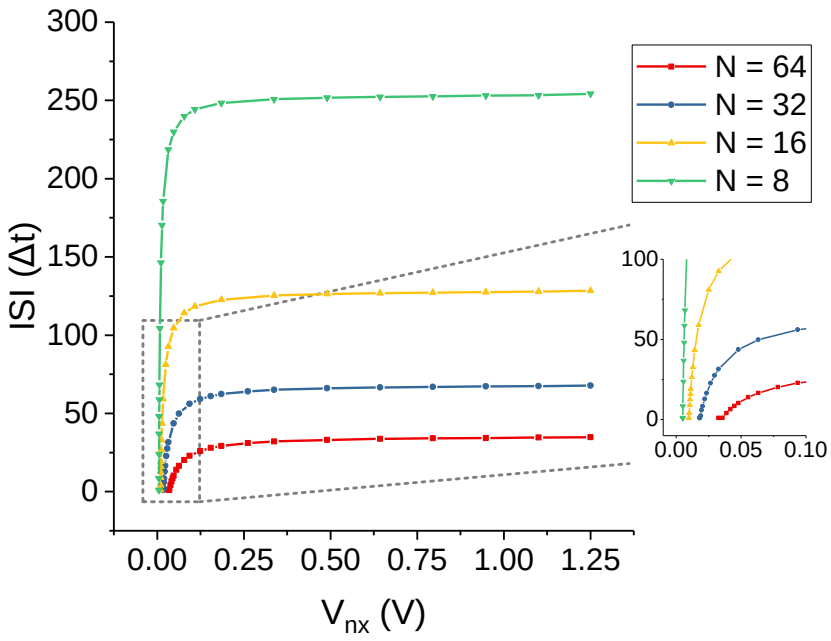


Figure 4.28 Dependence of output ISIs on V_{nx}

The inset of Figure 4.28 shows an expanded view of the curves in the dotted box. Here, it is clear that using the smallest value of N corresponding to a larger t_{ISImax} (= $ISI_{max} \times \Delta t$) also means that the initial V_h is small. The small initial V_h causes that the voltage changes very little over a large range of ISI values. To detect a small change in V_h, a random source with high bit resolution is required. The required resolution will be discussed in the next section.

4.5.7.3 Effect of the Non-Ideal Random Source

In the previous experiment, it is shown that V_h increases slowly in the constant input case and is also possible to change slowly with other input waveforms. Therefore, the random source generating samples of V_{nx} needs to have high resolution and low distortion. In this experiment, whether the output ISI distribution of the RS channel is affected by the non-ideal random source or not is tested. Again, a uniform distribution is provided and the output ISI distribution by collecting the

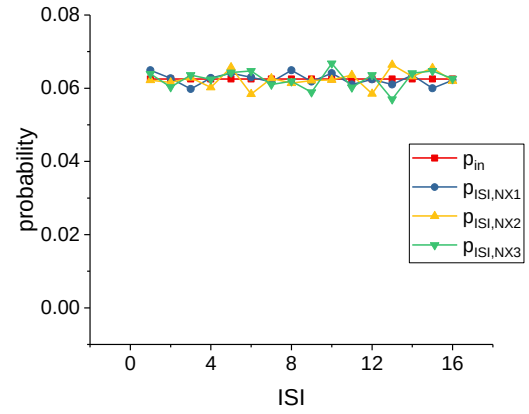
output ISIs is measured. $V_{nx}(t)$ in this experiment is considered as independent samples of a random source that are drawn on every $i\Delta t$.

Given a uniform input distribution, the output ISIs between 1 to ISI_{max} should be equiprobable. In the inset of Figure 4.28, it can be seen that V_h increases slowly before the time reaches t_{ISImax}. During integration, V_h is continuously compared to V_{nx} so the small changes of V_h could only be distinguished with a relatively high accuracy V_{nx}.In this chip, the random source either comes from the internal on-chip RNG or external off-chip LFSR. Both random sources applied on the chip are non-ideal: The RNG circuit has a non-ideal uniform distribution. The distribution of the RNG measured in [82] shows a Gaussian distortion similar to Figure 4.29(c). The LFSR as implemented in the FPGA can only have finite resolution in the distribution values. The effect of these non-idealities on the output ISI distribution is first explored in a Matlab simulations of three forms of the random source: An ideal uniform random source, *NX1*; a uniform random source, *NX2* (Figure 4.29(c)), where a 0.03-sigma Gaussian random value is added to the samples; and a random source with a 8-bit resolution, *NX3*, i.e. the output of *NX3* is quantized in one of the 8-bit values. The impact of these different random sources on the output distribution is dependent on the resolution of the ISI bins. In the case when ISI_{max} = 16, corresponding to the input probability p_{in} = 0.0625 (in the Matlab simulations, Δt = 1 so t_{ISImax} = 16 and p_{in} = 1/16), the output ISI distributions p_{ISI} of all three sources as shown in Figure 4.29(a) are very similar. However in the case of a large ISI_{max}, corresponding to an even smaller p_{in}, only the distribution of *NX1* is similar to the input as shown in Figure 4.29(b).

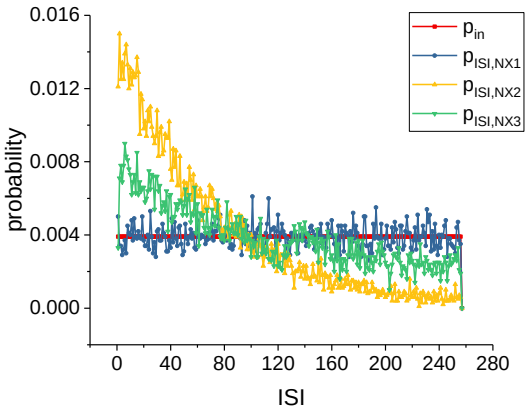
Figure 4.30 shows measured test results from one RS channel of the fabricated chip. Note that I_{rst} is tuned so that ISI_{max} = 16 given a constant input N = 64 (p_{in} = 0.0625). The bit resolution of the external LFSR random source is set to 14 bits. The analog output after passing the LFSR output through a 14-bit off-chip DAC is used as the random number output instead of the internal RNG output. When ISI_{max} is small (= 16), the output distributions from using either the external or internal random sources looks similar to the input distribution as shown in Figure 4.30(a). When ISI_{max} is increased to 256, the 14-bit LFSR in Figure 4.30(b) performs much better than the on-chip RNG output that is similar to the simulation results in Matlab. The KL divergence (see (3.1), where P refers to p_{in} and Q refers to p_{ISI}) generated by using the external (LFSR) and the internal (RNG) random sources are shown in Figure 4.30(c). As expected, the value of the KL divergence increases with ISI_{max} given a fixed number of samples. However, the values using the internal random source is consistently higher across all ISI_{max} values. This observation suggests that the

Gaussian distortion seen in the RNG distribution has a worse effect on the output ISI distribution than the finite bit resolution of the LFSR.

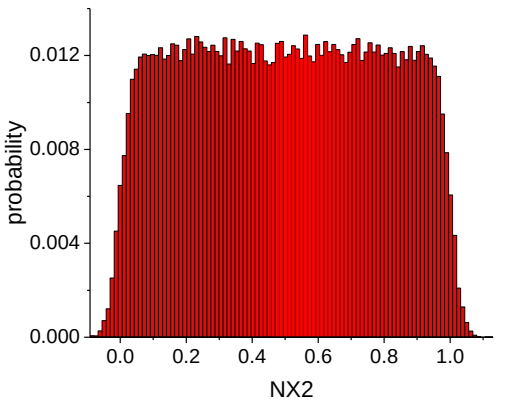
This experiment demonstrates that the Gaussian distortion from the RNG affects the output distribution as ISI_{max} increases. It is difficult to reduce this non-ideality in a simple way after fabrication. In contrast, the resolution of LFSR can be adjusted to an arbitrary number of bits if the corresponding DAC can be found. A 14-bit LFSR with ISI_{max} = 128 leads to the KL divergence of 0.002 as shown in Figure 4.30(c). Therefore, in the system level test combining the LS and RS array to form a message passing network described in Section 4.6, the random sources are all provided from the external LFSR.



(a)

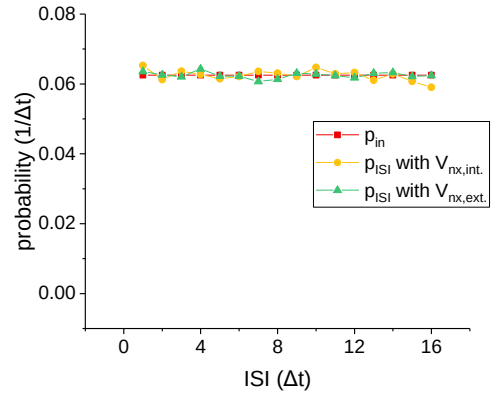


(b)

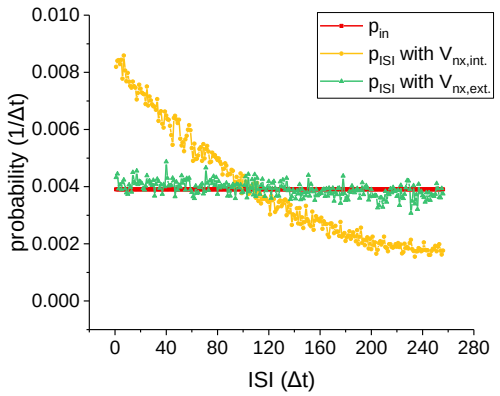


(c)

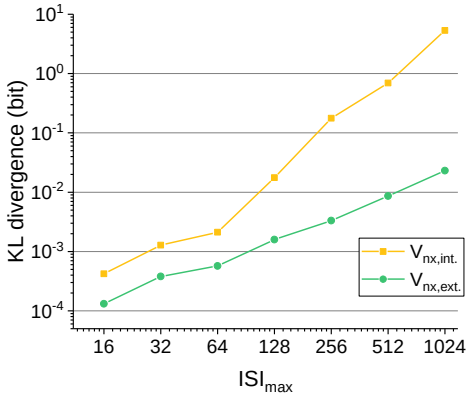
Figure 4.29 Output ISI distributions p_{ISI} over 10,000 samples as simulated in MATLAB. The uniform input probability p_{in} is (a) 0.0625 and (b) 0.0039. (c) The distribution of the random source *NX2*.



(a)



(b)



(c)

Figure 4.30 Output ISI distributions over 80,000 samples as measured from one RS channel using a constant input N = (a) 64 and (b) 4. The corresponding mathematical input probability p_{in} is (a) 0.0625 and (b) 0.0039. (c) The KL divergences of the output ISI distributions over the 80,000 samples with the internal (RNG) and external (LFSR) random sources.

4.5.7.4 Calibration of the Equivalent Resistance

In Section 4.5.2 the way to compute the resistance R_{eq} is presented by a special case of (4.15). In fact, R_{eq} can be theoretically formulated. Combining (4.10) and (4.13), (4.17) is obtained. Then, the resistance is shown in (4.18).

$$\frac{V_h}{V_{nx,\max}} = h(t)\Delta t = \frac{\kappa_{M_{a1}}}{C_1 U_T} \cdot I_{h1}(t) \cdot \Delta t \tag{4.17}$$

$$R_{eq} = \frac{V_h}{I_{h1}} = \frac{\kappa_{M_{a1}}}{C_1 U_T} \cdot V_{nx,\max} \cdot \Delta t \tag{4.18}$$

In this equation, the value of R_{eq} is set by $V_{nx,max}$ and Δt. The ratio of the current mirror in the **IV Converter block** is calibrated by *BitIV[7:0]* so that the equivalent resistance of the circuit (= $R_1 \times (63/4) \times Bit_{IV}$) matches the theoretical R_{eq} in (4.18). Otherwise, the output ISI distribution does not approximate the input. Ideally, Bit_{IV} = 128. Figure 4.31 demonstrates how the calibration affects the ISI output distribution p_{ISI}. Given a uniform input distribution (N = 32), only p_{ISI} with Bit_{IV} = 107 is close to p_{in}. For those Bit_{IV} < 107, the RS channel tends to generate spikes with larger ISIs because the equivalent resistance is not big enough. On the other hand, when Bit_{IV} > 107, more spikes are prone to be generated with smaller ISIs.

Because of circuit mismatch, the calibration has to be done separately in each channel. A memory block that stores the value for each channel would have been useful. Unfortunately, the author overlooked putting such block in this chip so only maximally 6 out of the 16 channels that have similar values of Bit_{IV} can be chosen to construct a factor graph.

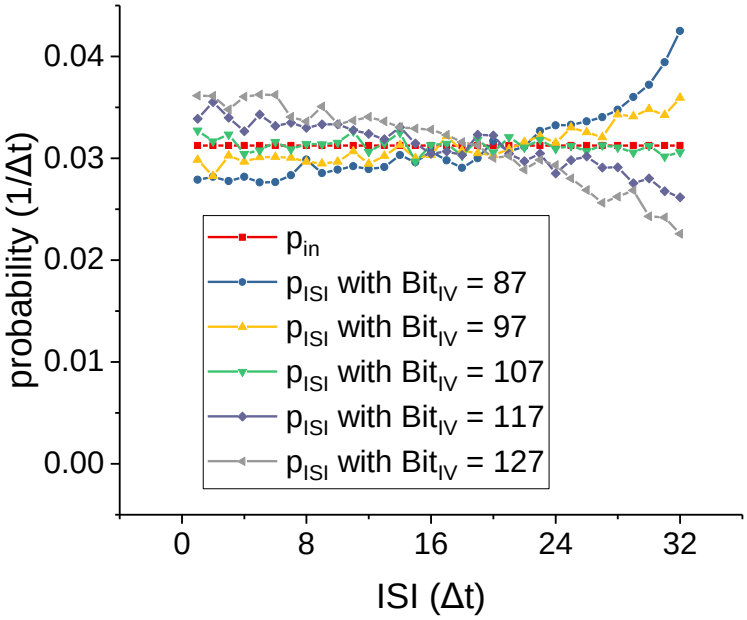


Figure 4.31 Effect of the calibration on the output ISI distributions. Given a constant input $N = 32$, p_{ISI} is obtained from one RS channel over 80,000 samples. The corresponding mathematical input probability p_{in} is 0.031.

4.6 Test Results of Message Passing in VLSI Factor Graphs

In the experiments demonstrated in this section, the results are obtained from the complete system that the RS array is implemented as an ASIC chip and the LS array is implemented on the FPGA. The architecture of the hardware system is shown in Figure 4.1 and the PCB is shown in Figure 4.32. As the measurement results presented in Section 4.5.7.3 and Section 4.5.7.4, the random source for each channel comes from a 14-bit LFSR with an off-chip DAC and only 6 out of 16 RS channels can be used. Also, ISI_{max} of an ISI distribution is no more than 128. Each LS-RS-combined channel represents a unidirectional factor node and several channels make up a unidirectional factor graph. The results on a few factor graphs are presented here. Note that the total time window, W_T, is based on both the number of collected samples and the mean of the ISI distribution, ISI_{mean}. For example, a uniform distribution with $ISI_{max} = 32$ has $ISI_{mean} = 16.5$. If collecting 100,000 samples is desired, then the average W_T is $16.5 \times 100000 \times \Delta t = 105.6$ s for $\Delta t = 64$ us. The time window W of the LS channel is defined as 1.05 s, and the **Mem ISI$_X$** or **Mem ISI$_Y$** module has

only 512 entries (see Figure 4.2). The output samples generated by the RS channel is based on the message-combined distribution, which is computed using the $(512)^2$ sample pairs and stored in the **Mem M_Z** module. To obtain 100,000 output samples in W_T, the LS channel is reused several times until the number of output samples is reached. In each new time window, the **Mem ISI_X**, **Mem ISI_Y** and **Mem M_Z** modules are reset.

Figure 4.32 The PCB of the hardware system consisting of an FPGA LS an ASIC RS. The occupied area of the PCB is 127×118 mm^2. The FPGA used is Lattice Semiconductor LFE3-70EA-FN484.

Note that the term of "ISIs" denotes ISI samples and the term of "ISI value" denotes some bin value in an ISI distribution in the following measurements.

The first factor graph shown in Figure 4.33(a) is composed of two RS channels and one LS-RS-combined channel. The two RS channels generate the output spike trains $spike_X$ and $spike_Y$ on the edges X and Y along the arrows based on the messages m_X, and m_Y defined in the channels. Messages m_X, and m_Y are assigned to two uniform distributions as shown in Figure 4.33(b),(c) in red and their ISI distributions obtained by counting the ISIs of $spike_X$ and $spike_Y$ are shown in blue. The constraint function of the LS-RS-combined channel is assigned to a switching-gain function

defined in (4.19). The switching-gain node sets a gain of 1 on the ISIs of $spike_X$ if the ISIs of $spike_Y$ are between [1,8], and a gain of ½ if the ISIs of $spike_Y$ are between [9,16].

Figure 4.33(d) shows the theoretical distribution (the red curve) of messages m_Z computed using the SPR and the ISI distribution (the blue curve) obtained by counting the ISIs of the output spike train $spike_Z$. The ISI distribution approximates the theoretical distribution as expected. Because messages m_X and m_Y are both uniform, the probability value of message m_Z for the ISI value between [1,16] is approximately three times as large as the probability value for the ISI value between [17,32]. Figure 4.33(e) shows the KL divergence of the three messages between the theoretical and the ISI distribution as a function of the number of ISIs. The KL divergence of message m_Y is the lowest value all the time. The reason is that ISI_{max} (= 16) of message m_Y is the smallest among the three messages. Given the same number of ISIs, the ISI distribution with a smaller ISI_{max} can approximate the theoretical one better because of more samples in each bin.

$$f(x,y,z)=\begin{cases}\delta(x-z) & \text{if } 1\le y\le 8\\ \delta(0.5x-z) & \text{if } 9\le y\le 16\end{cases} \tag{4.19}$$

The second experiment demonstrates that the constraint function can be defined flexibly in the FPGA. The factor graph having the same structure to the first graph is shown in Figure 4.34(a). Only the constraint function defined in the LS-RS-combined channel is changed. Messages m_X and m_Y are assigned to two uniform distributions as shown in Figure 4.34(b),(c) in red and their ISI distributions obtained by counting the ISIs of $spike_X$ and $spike_Y$ are shown in blue. The constraint function f of the switching node is set to function f_A in (4.20) if the ISIs of $spike_Y$ are between [1,8] while function f is set to function f_B in (4.21) if the ISIs of $spike_Y$ are between [9,16].

Because messages m_X and m_Y are both uniform, the theoretical distribution of message m_Z computed using the SPR is a combination of a V-shaped distribution for the ISI value between [1,16] and a triangular-shaped distribution for the ISI value between [34,48] as shown in Figure 4.34(d) in red. The ISI distribution in blue also shows the similar result to the theoretical one as expected. Figure 4.34(e) shows the KL divergence of the three messages as a function of the number of ISIs. As the same reason in the first factor graph, the KL divergence of message m_Y is the lowest value all the time because of the smallest ISI_{max} among the three messages.

$$f_A(x,y,z)=\begin{cases}1, & \text{if } 1\le x\le 8 \quad \& z=1\\ & \text{or if } 9\le x\le 15 \quad \& z=2\\ & \text{or if } 16\le x\le 21 \& z=3\\ & \text{or if } 22\le x\le 26 \& z=4\\ & \text{or if } 27\le x\le 30 \& z=5\\ & \text{or if } 31\le x\le 33 \& z=6\\ & \text{or if } 34\le x\le 35 \& z=7\\ & \text{or if } x=36 \quad \& z=8\\ & \text{or if } x=37 \quad \& z=10\\ & \text{or if } 38\le x\le 39 \& z=11\\ & \text{or if } 40\le x\le 42 \& z=12\\ & \text{or if } 43\le x\le 46 \& z=13\\ & \text{or if } 47\le x\le 51 \& z=14\\ & \text{or if } 52\le x\le 57 \& z=15\\ & \text{or if } 58\le x\le 64 \& z=16\\ 0, & \text{else}\end{cases} \tag{4.20}$$

$$f_B(x,y,z)=\begin{cases}1, & \text{if } x=1 \quad \& z=34\\ & \text{or if } 2\le x\le 3 \quad \& z=35\\ & \text{or if } 4\le x\le 6 \quad \& z=36\\ & \text{or if } 7\le x\le 10 \quad \& z=37\\ & \text{or if } 11\le x\le 15 \& z=38\\ & \text{or if } 16\le x\le 21 \& z=39\\ & \text{or if } 22\le x\le 28 \& z=40\\ & \text{or if } 29\le x\le 36 \& z=41\\ & \text{or if } 37\le x\le 43 \& z=42\\ & \text{or if } 44\le x\le 49 \& z=43\\ & \text{or if } 50\le x\le 54 \& z=44\\ & \text{or if } 55\le x\le 58 \& z=45\\ & \text{or if } 59\le x\le 61 \& z=46\\ & \text{or if } 62\le x\le 63 \& z=47\\ & \text{or if } x=64 \quad \& z=48\\ 0, & \text{else}\end{cases} \tag{4.21}$$

The third factor graph shown in Figure 4.35(a) consists of three RS channels and three LS-RS-combined channels. The three RS channels generate the output spike trains $spike_U$, $spike_V$ and $spike_W$ on the edges *U*, *V* and *W* along the arrows based on

the messages m_U, m_V, and m_W defined in the channels. Messages m_U and m_V are assigned to the same uniform distribution with ISI_{max} = 16 similar to the graph in Figure 4.34(c). Message m_W is assigned to a V-shaped distribution as shown in Figure 4.35(b) in red. The three LS-RS-combined channels are defined as a plus node, an equality node and a half-wave rectified linear unit (ReLU) node. The first two constraint functions are defined in (2.6) and the third one is defined in (4.22).

Because messages m_U and m_V are both uniform, the theoretical distribution of message m_X computed using the SPR is triangular shaped as shown in Figure 4.35(c) in red. The ISI distribution (the blue curve) approximates the theoretical one as expected. In addition, as the result in (2.7), message m_Y is the product of messages m_X and m_W using the SPR. Therefore, the theoretical distribution of message m_Y is a two-bump-shaped distribution as shown in Figure 4.35(d) in red and the ISI distribution (the blue curve) also approximates the theoretical one. Finally, the half-wave ReLU node shifts message m_Y toward the left with 10 ISI values to form message m_Z. The theoretical distribution of message m_Z computed using the SPR is shown in Figure 4.35(e) in red and the ISI distribution is shown in blue. Figure 4.35(f) shows the KL divergence of the six messages as a function of the number of ISIs. The KL divergences of messages m_Y and m_Z are much larger than messages m_W and m_X all the time. The reason is that the equality node filters out many input ISIs. Function $f_{equality}$ (see (2.6)) compares the ISIs of $spike_X$ and the ISIs of $spike_W$. If their values are not the same, the sample pairs are discarded. In the experience, many pairs are discarded. The histogram formed from the remaining ISIs is used to generate the resulting output spike train $spike_Y$. The resulting output ISI distribution has a worse approximation to the theoretical distribution than other constraint functions such as function f_{plus}, where ISIs are not discarded. This error then propagates to the next factor node for generating the spike train $spike_Z$.

$$f_{rectifier}(y,z)=\begin{cases}0 & \text{if } y\le 10\\ \delta(y-z-10) & \text{if } y>10\end{cases} \tag{4.22}$$

The system specifications are given in Table 4.4. The power consumption of the entire chip includes the power of the 16 on-chip DACs which are needed to adjust the reset current, I_{rst}, of each channel separately.

Specification	Quantity	Unit
Process	AMS 2P4M 0.35um	
Chip Area	2.16×2.74	mm^2
Chip Area (One RS Channel)	0.78×0.09	mm^2
Number of Channel	16	
Power of the RS chip	6.32	mW
Power of the RS single channel	0.046	mW
Supply Voltage	3.3	V
System Clock, Clk_{main}	10	MHz
System Clock, Clk_h	0.0156	MHz
Δt	64	us
Time Window W	1.05	s
Mem ISI	7×512	bit
Mem M_Z	6×128	bit
ISI Range	1 to 128	Δt
RS Input Range	0 to 63	
Reset Current, I_{rst}	~300	pA
Random Source $V_{nx,s}$ Range	1.25 to 2.5	V
LFSR Resolution	14	bit

Table 4.4 System specification

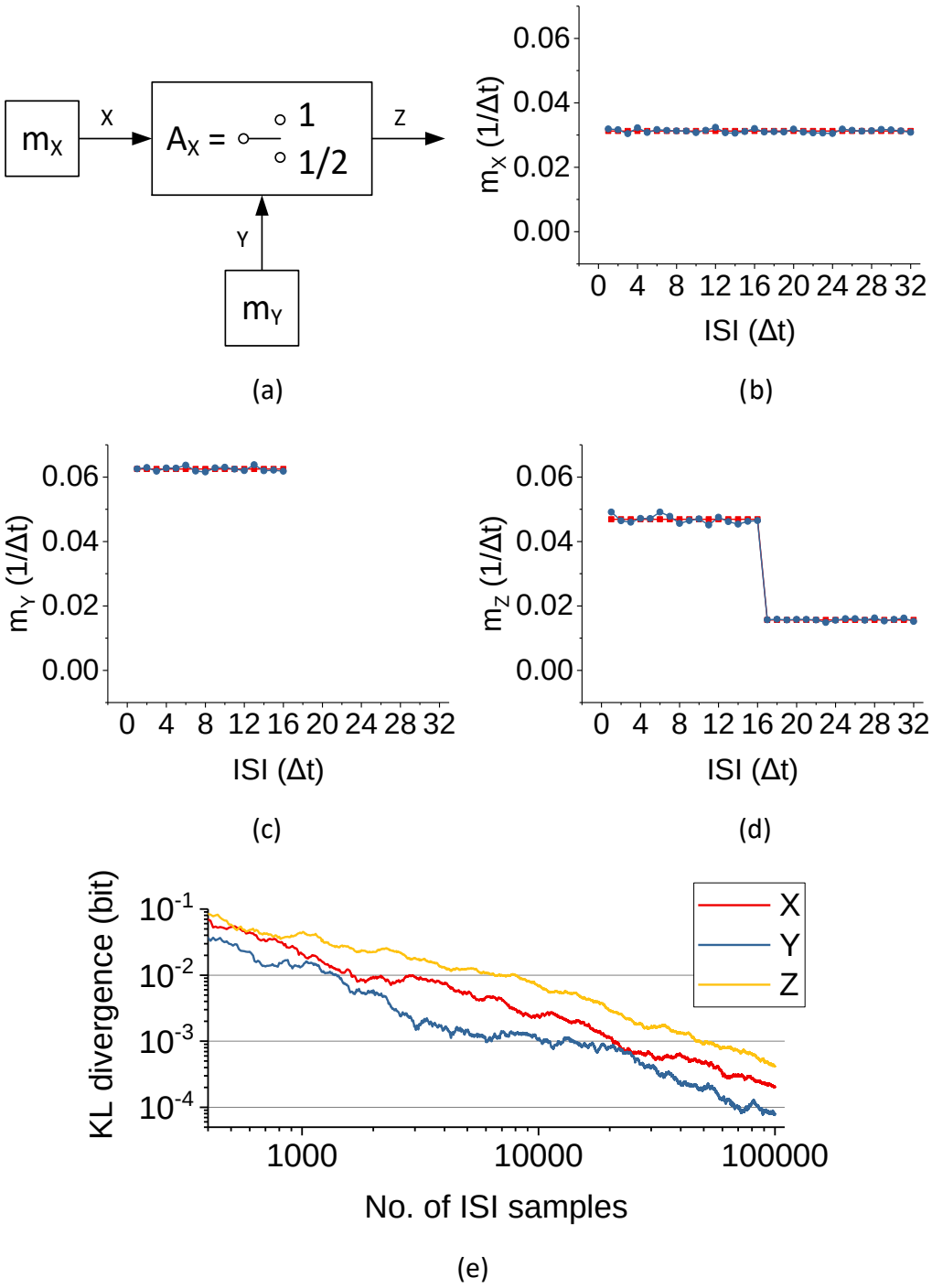


Figure 4.33 (a) Factor graph consisting of two RS channels, one LS-RS-combined channel, three variables *X*, *Y*, *Z* and the messages along the arrows. The messages of (b) *X* (c) *Y* (d) *Z* along the arrows, with the theoretical distribution in red and the output ISI distribution in blue over 100,000 samples. (e) KL divergences of the three messages as a function of the number of ISIs.

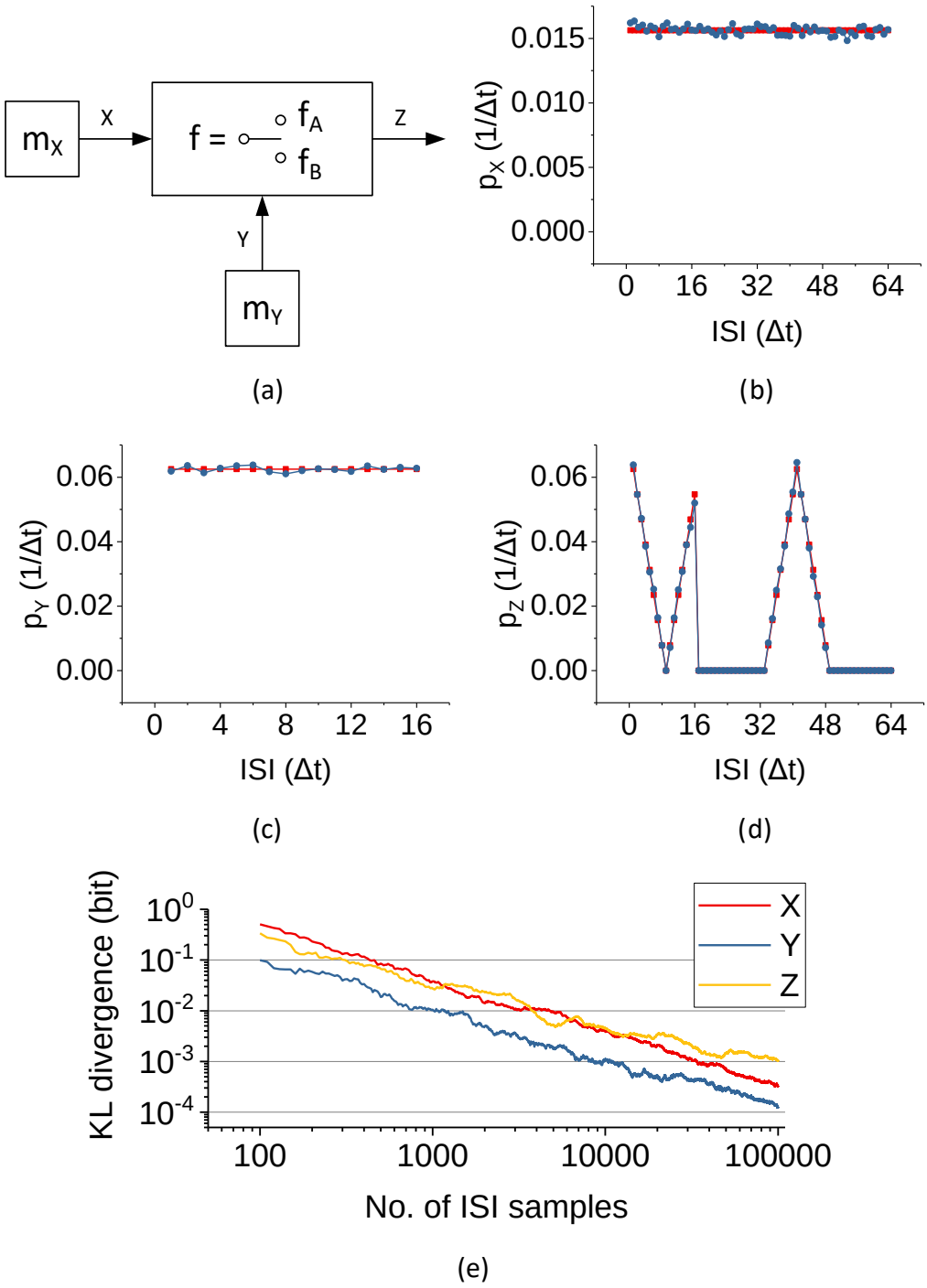


Figure 4.34 (a) Factor graph consisting of two RS channels, one LS-RS-combined channel, three variables *X*, *Y*, *Z* and the messages along the arrows. The messages of (b) *X* (c) *Y* (d) *Z* along the arrows, with the theoretical distribution in red and the output ISI distribution in blue over 100,000 samples. (e) KL divergences of the three messages as a function of the number of ISIs.

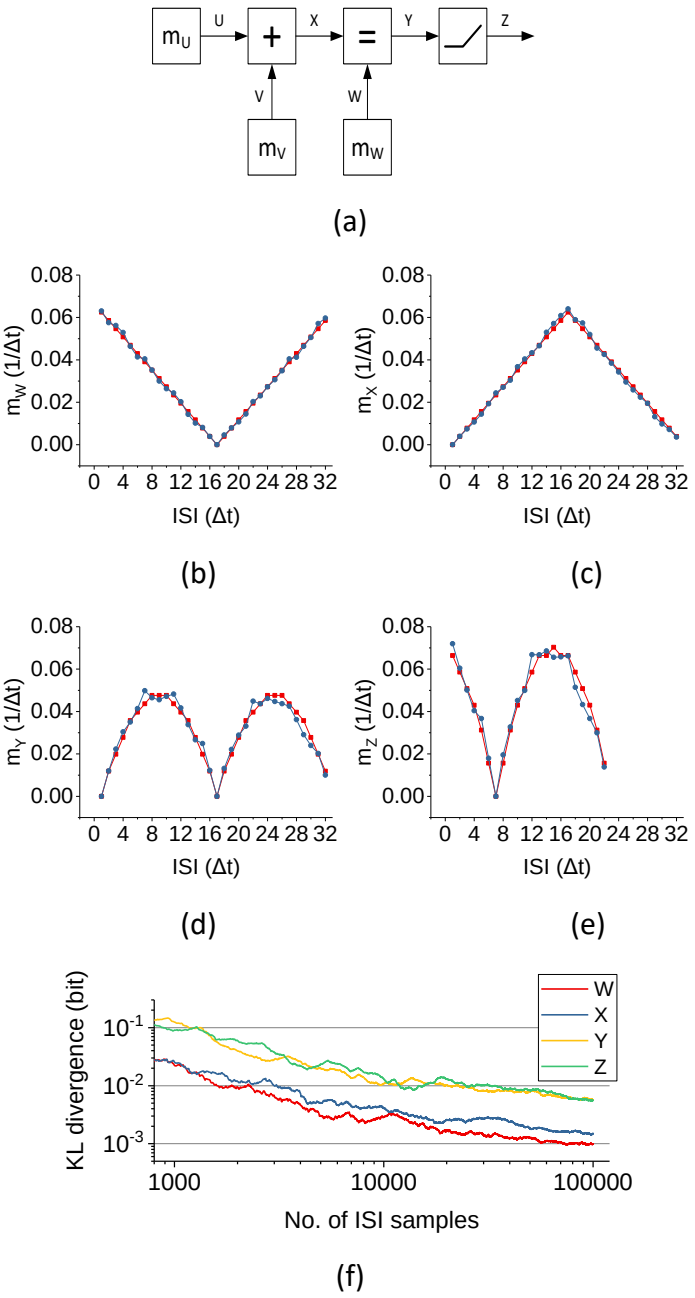


Figure 4.35 (a) Factor graph consisting of three RS channels, three LS-RS-combined channels and variables *U*, *V*, *W*, *X*, *Y*, *Z*, and the messages passing along the arrows. The messages of (b) *W* (c) *X* (d) *Y* (e) *Z* along the arrows with the theoretical distribution in red and the output ISI distribution in blue over 100,000 samples. (f) KL divergences of the four messages as a function of the number of ISIs.

Chapter 5 Conclusion and Future Work

This chapter presents conclusions on the study and the hardware implementation of the event-based Belief-Propagation (BP) model. Section 5.1 describes the considerations in implementing the different components of the BP model through aVLSI circuits and the theoretical approximations because of the constraints of the hardware system. Section 5.2 describes the improvements on the aVLSI design especially in consideration of a larger-scale multi-channel BP system in the future.

5.1 Conclusions of Hardware Design

This thesis presented a hardware system that implements a set of unidirectional factor nodes in the event-based BP model, where the messages are transmitted using spikes. The system consists of two components: The first component, i.e. the Landscape Sampling (LS) array, is implemented on an FPGA. This block records the values of Interspike Intervals (ISIs) from the input spike trains and produces the message-combined distributions using the recorded ISIs and the defined factor constraint functions. The second component is the Random Sampling (RS) array that can produce up to 16 output messages in the form of spike trains. This component is implemented on an ASIC chip fabricated in a 0.35um CMOS process. The output messages produced by this block are encoded in the ISIs of the output spike train that are sampled from the message-combined distributions based on the renewal theory.

Because the original definition of the hazard (A.5) requires a division which is difficult in aVLSI circuits, the continuous recursive form (A.6) was come up with. To ensure that the continuous recursive form leads to the same output ISI distribution as using the original definition, we performed the discrete-time simulations described in Chapter 3. The simulation results lead to the conclusion that if the input probability distribution $p_{in}(t)$ is a regularly-step-staircase (RSS) distribution, using

either the original definition (A.5) or the continuous recursive form (A.6) of the hazard makes the output ISI distribution p_{ISI} equal to p_{in}. If, on the other hand, $p_{in}(t)$ is not a RSS probability distribution or the discrete recursive form (A.7) is used, p_{ISI} only approximates p_{in} under the condition that the time step $\Delta t << t_{mean}$, where t_{mean} is the mean time of $p_{in}(t)$ defined in (2.11). In the case where p_{ISI} only approximates p_{in}, p_{in}(t) are limited to certain kinds of distributions with long t_{mean} compared to Δt. Binary probability distributions, for example, are unable to use the discrete-time approximation. These simulations help the author determine the specifications of an aVLSI circuit design that is used to realize the continuous recursive form of the hazard.

As presented in Chapter 4, the ASIC chip occupies the area of 2.16×2.74 mm^2 in a 0.35 um 2-poly-4-metal CMOS technology and was mounted on a system board with the FPGA of Lattice Semiconductor LFE3-70EA-FN484 that implements the LS channels as shown in Figure 4.32. The power consumption of the ASIC chip is 6.32 mW with 0.046 mW per RS channel. The remaining power consumption of the ASIC excluding the RS channels comes from the RS DACs and the bias generators.

The reason why the RS circuit is implemented in aVLSI is not only because analog circuits are suitable for the continuous recursive form of the hazard but the mathematical terms of the hazard function in (A.6) can easily be realized by analog components, such as a current charging a capacitor (representing the integration) and a transistor operating in subthreshold regime where the current is exponentially dependent on the gate voltage. The multiplication term can be realized by the gain function of a current-mirror circuit array. However, in the current system, the speed of the ASIC is limited by the time constants needed for the analog circuits of the RS channel. It is the reason why Δt is defined in the range of microseconds, i.e. Δt = 64 us.

Because of the clocked nature of both the LS FPGA and the RNG circuit, the ISI samples in the implementation were constrained to integer values. However this constraint allows the author to design a simple counter for a specific range of integer values and also allowed one to compute the histograms of two functions $f(x,y,z)$, the message-combined distribution before normalization, and $F(x,y)$, the normalization term, in parallel (see Figure 4.2). The possible integer values of ISI in the hardware system were in the range of [1,128] (i.e. $t_{ISI} \in [\Delta t, 128\Delta t]$, where time step Δt = 64us). When increasing the maximum ISI, i.e. ISI_{max}, an on-chip RNG with an ideal uniform distribution or an off-chip LFSR with higher bits is needed. Defining ISI_{max} = 128 and using a 14-bit off-chip LFSR with a 14-bit off-chip DAC leads to an acceptable value of the KL divergence of 0.002 (see Figure 4.30(c)).

The input distribution to the discrete-input version of the RS circuit cannot be arbitrary because the ratio between the maximum and minimum values of $p_{in}(t)$ can only go up to 64. In a RS channel, there are only two parameters, I_{rst} and Bit_{IV} (see Figure 4.19), that need adjusting. The reset current I_{rst} is used to define ISI_{max} while Bit_{IV} is used to calibrate the equivalent resistance for converting the hazard current I_h to the hazard voltage V_h. This calibrated resistance has to match the resistance R_{eq} calculated in (4.18) so that p_{ISI} approximates p_{in} well as measured by the KL divergence.

In the event-based BP model, having a sufficient number of ISIs in spike trains is significant for representing the carrying messages. To approximate a distribution well, the value of ISI_{max} determines the number of ISIs required. For example, a good approximation (the KL divergence is 0.001) can be made over 1000 ISIs in a factor graph with binary variables (i.e. ISI_{max} = 2) (see Figure 3.5). However, if ISI_{max} = 32, the number of ISIs needs to increase to 16000 so that a similar KL divergence value can be achieved. In Section 4.6, 100,000 ISIs is used to represent the messages. With ISI_{max} = 32, the time for collecting 100,000 sequential samples would take 105.6 s.

Since the implemented event-based BP hardware model in this thesis only produces unidirectional messages, the messages for the opposite direction in a bidirectional link between two nodes can be computed by reusing factor nodes. Sometimes a delay node is needed to compute an output message correctly. For example, in the event-based Kalman filter, the spike train carrying the messages of the prediction and the spike train carrying the messages of the observation should arrive at the equality node (see Figure 3.8) in the same time window so that the correct distribution is reflected in the output spike train. The number of nodes in the prediction and observation paths should be the same. If not, a delay node, i.e. a unity-gain constraint node, should be added to the path with fewer nodes.

Some circuit constraints in terms of sample number, speed, a good random source and the mismatch between channels can be possibly improved so that a larger-scale multi-channel system is feasible. The modifications are discussed in the next section.

5.2 Hardware Improvements and Outlook

Several improvements that address the constraints of this tested prototype hardware system are proposed and discussed in this section. First, as mentioned in Section 5.1, each RS channel need two parameters for channel calibration, i.e. I_{rst} and Bit_{IV}. The 8-bit parameter Bit_{IV} that is used to calibrate the equivalent resistance to

match the theoretical value can be stored in eight single bit latches for each channel. Although calibration between channels is still necessary, the latches for all channels only consume little resource. As for the other parameter I_{rst}, 16 on-chip RS DACs are implemented to generate 16 individual I_{rst} for the RS channels in the current ASIC. A large percentage of the overall chip power consumption is contributed by the RS DACs (see Table 4.4). I_{rst} defines the initial current of transistor M_{a1} (see Figure 4.20). In the **Hazard Reset** block (see Section 4.5.5), I_{rst} first defines the reset voltage V_{rst1} of the output buffer and then V_{rst1} controls the gate of transistor M_{a1}. Once a spike happens, the gate voltage of transistor M_{a1} is reset to V_{rst1} and the initial current of transistor M_{a1} is reset to I_{rst} within the pulse width of the spike due to the strong driving capacity of the output buffer for V_{rst1}. However, the mismatch of the transistors in the buffer causes V_{rst} to vary between channels, resulting in different initial currents of transistor M_{a1} even though the same current I_{rst} is provided. This is the reason for separated RS DACs required. It is not feasible to have individual RS DACs for all channels in a large-scale multi-channel implementation. The author proposes a modification in the circuit to remove the individual DACs for I_{rst} and a global I_{rst} is explained as follows.

The buffer in Figure 4.24 for producing V_{rst1} is removed and the gate of the transistor M_{a1} in Figure 4.20 is controlled by I_{rst} using a diode-connected transistor M_{a2} as shown in Figure 5.1. Because the transistor M_{a1} has to be operated in the subthreshold regime over a range of currents, the width/length ratio sizes of transistors M_{a1} and M_{a2} has to be large. Because the area of the transistors are also large, the process mismatch between these two transistors is low. In addition, adding a cascade transistor above transistor M_{a2} and placing transistors M_{a1} and M_{a2} close enough in the layout, the error of the current mirror can be improved. In the current ASIC, I_{rst} was set to a small current value (~256 pA) in order to save the power. Such a small current cannot support the operation of transistor M_{a2} (see Figure 5.1) in a saturation region, i.e. V_{ds} > 100 mV. In addition, a buffer was required in the current design due to the small value I_{rst}. The source bias V_{src} was also added in order to increase the precision of the initial current of transistor M_{a1} to be I_{rst} [98]. If I_{rst} is increased to a sufficient value, e.g. 100 times larger = 25.6 nA, the diode-connected transistor M_{a2} will be in saturation and I_{rst} will be large enough to reset the gate of transistor M_{a1} within a period that is determined by the pulse width of the output spike. Also, V_{src} can be set to Gnd; therefore, it can be removed. Moreover, increasing I_{rst} leads to another improvement: Speed.

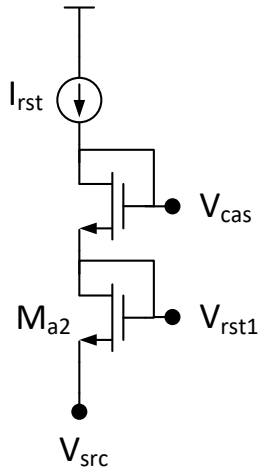


Figure 5.1 Modified Reset Hazard block. I_{rst} is increased and V_{src} is set to Gnd.

Derived from (2.12), (4.13) and (4.15), Δt associated with the circuit parameters can be represented as (5.1). This equation shows that Δt can be reduced by increasing I_{rst} and, therefore, the system speed can be increased. A simple measurement from the current hardware system was done by reducing Δt by a quarter in the FPGA and increasing I_{rst} by 4x in the ASIC. Figure 5.2 shows that p_{ISI} approximates p_{in} for different input probability distributions. Collecting the same number of ISIs is four times faster than before. Of course, increasing I_{rst} will incur a larger power consumption but the system needs a short time for collecting the same number of samples, therefore, leading to approximately the same energy consumption. Another solution to increasing the speed is to reduce the size of the capacitor C_1 according to (5.1). The drawback is that more kTC noise [99] will be brought in during reset, leading to a noisy initial current flowing through transistor M_{a1}. In the current design, capacitor C_1 is at 4 pF, which can be possibly reduced.

$$\Delta t = \frac{1}{h_{\max}} = \frac{128 \cdot C_1 \cdot U_T}{\kappa_{M_{a1}} \cdot I_{rst} \cdot N \cdot ISI_{max}} \tag{5.1}$$

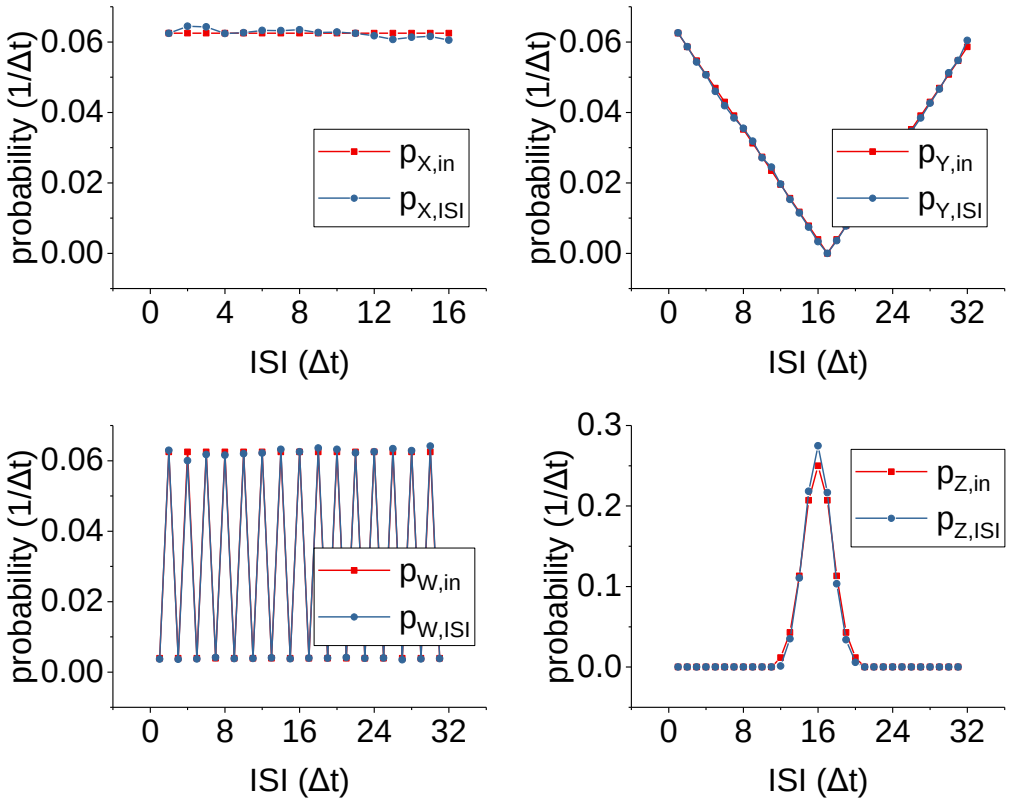


Figure 5.2 Output ISI distributions over 100,000 ISIs as measured from a RS channel using four different input distributions. *Δt* is set to 16 us.

For p_{ISI} to approximate p_{in}, a number of ISIs is required especially when ISI_{max} is large. In Section 5.1, given a uniform p_{in} and ISI_{max} = 32, the time taken to collect 100,000 samples would take 105.6 s. This lengthy accumulation time can be reduced by having multiple inputs or nodes representing one distribution. Similar to cortical neurons which have a large number of input neurons [29], the time required to collect 100,000 samples, for example, can be reduced to 1.056s with 100 inputs. However, this means that the chip area of one RS channel will have to be reduced so that more channels can be placed within a fixed chip area.

Similar to other stochastic models such as RBMs, the event-based BP VLSI model requires a pseudo-number generator for the sampling process. Several aVLSI implementations of pseudo-number generators have been proposed over the years and have focused on either digital noise (noisy bits) [100], [101], uniform distributions [102], or other types of predefined, fixed distributions [103]–[105], or true random number generator circuits which generate discrete, Bernoulli and quasi-continuous, exponential random variables [106]. In [107], a single-photon avalanche

diode (SPAD) was used as an aVLSI noise source and connected to a spike-response neuron model [29] implemented on an FPGA. Digital random number generators have been used as a cheap method of creating random connections for a network using the Neural Engineering Framework [64]. However, previous approaches have focused on random number generator in hardware that were based on fixed predefined distributions. An on-chip pseudo-random generator [96] is used in the continuous-Input RS circuit [82] described in Section 4.4, but because of the non-ideal Gaussian nature of the output distribution caused by the switched capacitors, the author uses an LFSR implemented on the FPGA for the results reported in Section 4.5.7.

A random number generator circuit that produces a less distorted uniform distribution can be designed using larger component sizes in a future design. A further option is to use a current-mode random number generator circuit so that the IV converter is not required. However, amplifying the hazard current I_{h1} is still needed in order to reduce the time constant caused by parasitic impedance. The result of the comparison between the hazard current and the current coding the random number value would have to be completed within Δt. In addition, the output distribution of this new random number circuit should be ideally uniform otherwise there will be a similar problem to the case where the non-ideal Gaussian distribution of the random number generator circuit changes the output ISI distribution.

A future extension of the current hardware would be to configure this system to implement a network that performs inference on the output of event-based sensors such as the DVS [15] and the cochlea which generates asynchronous outputs [108]. Similar to the model simulation applications in Section 3.3.1, this hardware system can be combined with the DVS in a tracking task such as predicting the position of an object across the field of view of the retina.

Appendix A Theoretical Basis

A.1 Recursive Form of the Hazard Function

The hazard *h* is defined as the function of the input probability *p* as shown in (A.1). The product of the hazard *h(t)* and the interval *dt*, i.e. *h(t)dt*, represents the probability of an event that happens between [*t*,*t*+*dt*] given that no events happen before *t*. Here only *t* >= 0 is considered; that is, *p(t)* and *h(t)* only have values since *t* >= 0. The hazard can also be represented as a continuous recursive form as shown in (A.4). The derivation is shown as follows. First, the survival function *S(t)* that describes the probability of no events happening before *t* is defined in (A.2). Then, the hazard *h(t)* can be represented by only the survival function *S(t)* as shown in (A.2). The result in (A.2) leads to (A.3). By replacing the denominator of (A.1) to (A.3), the continuous recursive form of the hazard is obtained in (A.4). Therefore, the hazard can be computed by either the original definition (A.1) or the continuous recursive form (A.4). Their values are the same.

$$h(t) = \frac{p(t)}{1 - \int_0^t p(t')\,dt'} \tag{A.1}$$

$$\begin{aligned} h(t) &= \frac{p(t)}{S(t)}, \text{ where } S(t) = 1 - \int_0^t p(t')\,dt' \\ &= -\frac{d}{dt}\log\left(S(t)\right) \end{aligned} \tag{A.2}$$

$$S(t) = \exp\left(\int_0^t -h(t')\,dt'\right) \tag{A.3}$$

$$h(t) = p(t)\exp\left(\int_0^t h(t')\,dt'\right) \tag{A.4}$$

Now a particular case is considered where *p(t)* is a regularly-step-staircase (RSS) probability distribution, whose value is only changed on time $t = i\Delta t$, where $i \in N_0$ =

{{0} ∪ *N*} and *Δt* is defined as the time step, as shown in Figure A.1. The reason for discussing this case is that this form of *p(t)* corresponds to how the equivalent signal provided to the Random Sampling (RS) block.

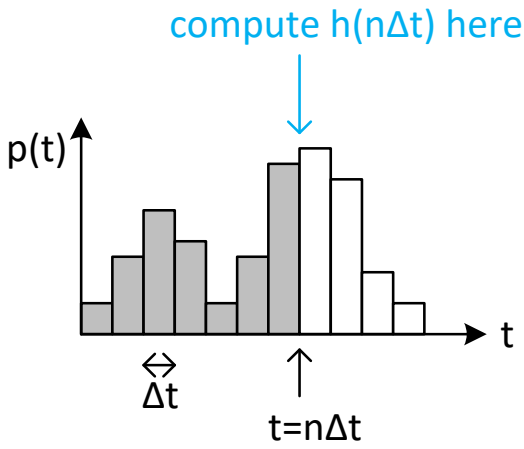


Figure A.1 Input RSS probability distribution *p(t)* whose values are only changed on time *t* = *iΔt*.

To compute the hazard *h(t)* on *t* = *nΔt*, i.e. *h(nΔt)*, (A.1) can be written as (A.5), where the hazard can be computed using only the values of *p(t)* on *t* = 0, 1*Δt*, 2*Δt*, …, *nΔt*. If using the continuous recursive form, *h(nΔt)* needs to be computed in continuous time as shown in (A.6). The value of the hazard from (A.5) is equal to the one from (A.6). However, if *h(nΔt)* is computed using the discrete form as shown in (A.7), its value only approximates to the one from the original definition under the condition that $\Delta t << t_{mean}$, where t_{mean} indicates the mean time of the RSS probability distribution *p(t)* and is defined in (A.8). Therefore, it cannot be thought intuitively that (A.5) will lead to the result of the discrete recursive form on (A.7).

$$h(n\Delta t) = \frac{p(n\Delta t)}{1 - \sum_{i \in \{0,\ldots,n-1\}} p(i\Delta t)\Delta t}, \text{ where } h(0) = p(0) \tag{A.5}$$

$$h(n\Delta t) = p(n\Delta t)\exp\left(\int_0^{n\Delta t} h(t')dt'\right) \tag{A.6}$$

$$h(n\Delta t) \approx p(n\Delta t)\cdot\exp\left(\sum_{i \in \{0,\ldots,n-1\}} h(i\Delta t)\Delta t\right), \text{ if } \Delta t << t_{mean} \tag{A.7}$$

$$t_{mean} = E[T] = \int_0^\infty tp(t)\,dt \tag{A.8}$$

A.2 Output ISI Probability Distribution

This section presents that the output InterSpike Interval (ISI) distribution p_{ISI} approximates the input probability distribution p using the discrete-time approximation. Given a uniform random variable "*NX*", the probability of generating a spike on $t = i\Delta t$ is equal to $h(i\Delta t)\Delta t$ (see Figure A.2 for the concept), and can be written as (A.9), where "$nx(i\Delta t)$" indicate the only one sample from variable *NX* on $t = i\Delta t$ and variable *NX* only produces a new sample on $t = i\Delta t$. Thus, the hazard can be seen as the instantaneous firing rate on $t = i\Delta t$.

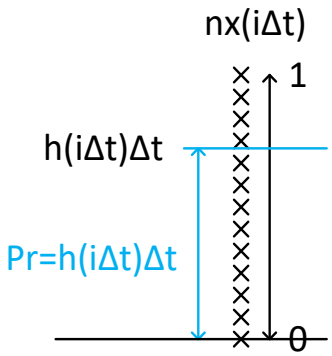


Figure A.2 ***nx(iΔt)* is a sample on *t* = *iΔt*. Its value is samples uniformly between [0,1]. The probability of generating a spike on *t* = *iΔt* is equal to *h(t)Δt*.**

$$\Pr\left(h(i\Delta t)\Delta t > nx(i\Delta t)\right) = h(i\Delta t)\Delta t \tag{A.9}$$

To generate a spike on $t = n\Delta t$ implies that the hazard is smaller than nx when $t = i\Delta t$, where $i \in \{0,\ldots,n-1\}$, and surpasses nx on $t = n\Delta t$. Therefore, the probability of an event happening on $t = n\Delta t$ can be written as a series product of the probabilities $(1-h(i\Delta t)\Delta t)$ and $h(n\Delta t)\Delta t$ as shown in (A.10), where p_{ISI} indicates the output ISI distribution.

First, the condition where $p(t)$ is a RSS probability distribution is discussed. Replacing the hazard in (A.10) by (A.5) leads the result in (A.11) that shows p_{ISI} is equal to p using either the original definition or the continuous recursive form of the hazard. Second, it is assumed that $p(t)$ is still a RSS probability distribution but the hazard is computed using the discrete recursive form as (A.7). By replacing the hazard in (A.10) by (A.7) and using the Taylor series in (A.13), (A.12) shows that p_{ISI} is

only the approximation of p under the condition that $\Delta t << t_{mean}$. Therefore, when using the discrete-time approximation to generate output spikes whose ISIs encode the input RSS probability distribution, it is recommended to compute the hazard using the original definition or the continuous recursive form. Third, if $p(t)$ is a continuous-value probability distribution, p_{ISI} is only an approximation of p no matter how the hazard is computed. The approximation is valid under the condition that $\Delta t << t_{mean}$.

$$p_{ISI}(n\Delta t)\Delta t = \left(\prod_{i\in\{0,\ldots,n-1\}} \left(1-h\left(i\Delta t\right)\Delta t\right) \right) \times h\left(n\Delta t\right)\Delta t \tag{A.10}$$

$$\begin{aligned} p_{ISI}(n\Delta t)\Delta t &= \left(1-p\left(0\right)\Delta t\right) \times \left(1-\frac{p\left(1\Delta t\right)\Delta t}{1-p\left(0\right)\Delta t} \right) \times \mathrm{K} \times \\ &\quad \left(1-\frac{p\left(\left(n-1\right)\Delta t\right)\Delta t}{1-\sum_{i\in\{0,\ldots,n-2\}} p\left(t'\right)\Delta t} \right) \times \left(\frac{p\left(n\Delta t\right)\Delta t}{1-\sum_{i\in\{0,\ldots,n-1\}} p\left(t'\right)\Delta t} \right) \\ &= \left(1-p\left(0\right)\Delta t\right) \times \left(\frac{1-\sum_{i\in\{0,1\}} p\left(i\Delta t\right)\Delta t}{1-p\left(0\right)\Delta t} \right) \times \mathrm{K} \times \\ &\quad \left(\frac{1-\sum_{i\in\{0,\ldots,n-1\}} p\left(i\Delta t\right)\Delta t}{1-\sum_{i\in\{0,\ldots,n-2\}} p\left(i\Delta t\right)\Delta t} \right) \times \left(\frac{p\left(n\Delta t\right)\Delta t}{1-\sum_{i\in\{0,\ldots,n-1\}} p\left(i\Delta t\right)\Delta t} \right) \\ &= p\left(n\Delta t\right)\Delta t \end{aligned} \tag{A.11}$$

$$\begin{aligned} p_{ISI}(n\Delta t)\Delta t &= h\left(n\Delta t\right)\Delta t \cdot \exp\left(\ln\left(\prod_{i\in\{0,\ldots,n-1\}} \left(1-h\left(i\Delta t\right)\Delta t\right) \right) \right) \\ &= h\left(n\Delta t\right)\Delta t \cdot \exp\left(\sum_{i\in\{0,\ldots,n-1\}} \ln\left(1-h\left(i\Delta t\right)\Delta t\right) \right) \\ &= h\left(n\Delta t\right)\Delta t \cdot \exp\left(-\sum_{i\in\{0,\ldots,n-1\}} \left(h\left(i\Delta t\right)\Delta t + \frac{\left(h\left(i\Delta t\right)\Delta t\right)^2}{2} + \mathrm{L} \right) \right) \\ &\approx h\left(n\Delta t\right)\Delta t \cdot \exp\left(-\sum_{i\in\{0,\ldots,n-1\}} h\left(i\Delta t\right)\Delta t \right), \text{ if } \Delta t << t_{mean} \\ &= p\left(n\Delta t\right)\Delta t \end{aligned} \tag{A.12}$$

$$\log(1-x) = -\sum_{n=1}^{\infty} \frac{x^n}{n} = -x - \frac{x^2}{2} - \frac{x^3}{3} - \mathrm{L} \tag{A.13}$$

Bibliography

[1] J. Zylberberg, J. T. Murphy, and M. R. DeWeese, “A sparse coding model with synaptically local plasticity and spiking neurons can account for the diverse shapes of V1 simple cell receptive fields,” *PLOS Comput. Biol.*, vol. 7, no. 10, p. e1002250, Oct. 2011.

[2] G. E. Hinton, S. Osindero, and Y.-W. Teh, “A fast learning algorithm for deep belief nets,” *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, May 2006.

[3] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back, “Face recognition: A convolutional neural-network approach,” *IEEE Trans. Neural Netw.*, vol. 8, no. 1, pp. 98–113, Jan. 1997.

[4] A. L. Hodgkin and A. F. Huxley, “A quantitative description of membrane current and its application to conduction and excitation in nerve,” *J. Physiol.*, vol. 117, no. 4, pp. 500–544, Aug. 1952.

[5] M. O. Ernst and M. S. Banks, “Humans integrate visual and haptic information in a statistically optimal fashion,” *Nature*, vol. 415, no. 6870, p. 429, Jan. 2002.

[6] S. Gepshtein and M. S. Banks, “Viewing geometry determines how vision and haptics combine in size perception,” *Curr. Biol.*, vol. 13, no. 6, pp. 483–488, Mar. 2003.

[7] D. Alais and D. Burr, “The ventriloquist effect results from near-optimal bimodal integration,” *Curr. Biol.*, vol. 14, no. 3, pp. 257–262, Feb. 2004.

[8] U. Beierholm, L. Shams, W. J. Ma, and K. Koerding, “Comparing Bayesian models for multisensory cue combination without mandatory integration,” in *Advances in Neural Information Processing Systems 20*, J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, Eds. Curran Associates, Inc., 2008, pp. 81–88.

[9] P. Mamassian and M. S. Landy, “Observer biases in the 3D interpretation of line drawings,” *Vision Res.*, vol. 38, no. 18, pp. 2817–2832, Sep. 1998.

[10] P. Mamassian and M. S. Landy, “Interaction of visual prior constraints,” *Vision Res.*, vol. 41, no. 20, pp. 2653–2668, Sep. 2001.

[11] J. Feldman, “Bayesian contour integration,” *Percept. Psychophys.*, vol. 63, no. 7, pp. 1171–1182, Oct. 2001.

[12] Y. Weiss, E. P. Simoncelli, and E. H. Adelson, “Motion illusions as optimal percepts,” *Nat. Neurosci.*, vol. 5, no. 6, p. 598, Jun. 2002.

[13] K. P. Körding and D. M. Wolpert, “Bayesian integration in sensorimotor learning,” *Nature*, vol. 427, no. 6971, p. 244, Jan. 2004.

[14] P. Lichtsteiner, C. Posch, and T. Delbruck, “A 128 x 128 120 dB 15 us latency asynchronous temporal contrast vision sensor,” *IEEE J. Solid-State Circuits*, vol. 43, no. 2, pp. 566–576, Feb. 2008.

[15] R. Berner, C. Brandli, M. Yang, S. C. Liu, and T. Delbruck, “A 240x180 10mW 12us latency sparse-output vision sensor for mobile applications,” in *2013 Symposium on VLSI Circuits*, 2013, pp. C186–C187.

[16] S. C. Liu, A. van Schaik, B. A. Minch, and T. Delbruck, “Asynchronous binaural spatial audition sensor with 2x64x4 channel output,” *IEEE Trans. Biomed. Circuits Syst.*, vol. 8, no. 4, pp. 453–464, Aug. 2014.

[17] M. Yang, C. H. Chien, T. Delbruck, and S. C. Liu, “A 0.5 V 55 uW 64 x 2 channel binaural silicon cochlea for event-driven stereo-audio sensing,” *IEEE J. Solid-State Circuits*, vol. 51, no. 11, pp. 2554–2569, Nov. 2016.

[18] P. Smolensky, “Information processing in dynamical systems: Foundations of harmony theory,” Cambridge, MA, USA: MIT Press, 1986, pp. 194–281.

[19] G. E. Hinton, “Training products of experts by minimizing contrastive divergence,” *Neural Comput.*, vol. 14, no. 8, pp. 1771–1800, Aug. 2002.

[20] H. Lee, P. Pham, Y. Largman, and A. Y. Ng, “Unsupervised feature learning for audio classification using convolutional deep belief networks,” in *Advances in Neural Information Processing Systems 22*, Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, Eds. Curran Associates, Inc., 2009, pp. 1096–1104.

[21] G. E. Dahl, D. Yu, L. Deng, and A. Acero, “Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition,” *IEEE Trans. Audio Speech Lang. Process.*, vol. 20, no. 1, pp. 30–42, Jan. 2012.

[22] G. E. Hinton and R. R. Salakhutdinov, “Reducing the dimensionality of data with neural networks,” *Science*, vol. 313, no. 5786, pp. 504–507, Jul. 2006.

[23] D. F. Wulsin, J. R. Gupta, R. Mani, J. A. Blanco, and B. Litt, “Modeling electroencephalography waveforms with semi-supervised deep belief nets: fast classification and anomaly measurement,” *J. Neural Eng.*, vol. 8, no. 3, p. 036015, 2011.

[24] H. Larochelle and Y. Bengio, “Classification using discriminative restricted Boltzmann machines,” in *Proceedings of the 25th International Conference on Machine Learning*, New York, NY, USA, 2008, pp. 536–543.

[25] H. Chen and A. F. Murray, “Continuous restricted Boltzmann machine with an implementable training algorithm,” *IEE Proc. - Vis. Image Signal Process.*, vol. 150, no. 3, pp. 153–158, Jun. 2003.

[26] T. B. Tang and A. F. Murray, “Adaptive sensor modelling and classification using a continuous restricted Boltzmann machine (CRBM),” *Neurocomputing*, vol. 70, no. 7, pp. 1198–1206, Mar. 2007.

[27] J. Chao, F. Shen, and J. Zhao, “Forecasting exchange rate with deep belief networks,” in *The 2011 International Joint Conference on Neural Networks*, 2011, pp. 1259–1266.

[28] Y. Chen, X. Zhao, and X. Jia, “Spectral-spatial classification of hyperspectral data based on deep belief network,” *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.*, vol. 8, no. 6, pp. 2381–2392, Jun. 2015.

[29] W. Gerstner and W. M. Kistler, *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Cambridge University Press, 2002.

[30] E. M. Izhikevich, “Simple model of spiking neurons,” *IEEE Trans. Neural Netw.*, vol. 14, no. 6, pp. 1569–1572, Nov. 2003.

[31] W. Gerstner, “Time structure of the activity in neural network models,” *Phys. Rev. E*, vol. 51, no. 1, pp. 738–758, Jan. 1995.

[32] Q. Yu, H. Tang, K. C. Tan, and H. Yu, “A brain-inspired spiking neural network model with temporal encoding and learning,” *Neurocomputing*, vol. 138, pp. 3–13, Aug. 2014.

[33] J. J. Wade, L. J. McDaid, J. A. Santos, and H. M. Sayers, “SWAT: A spiking neural network training algorithm for classification problems,” *IEEE Trans. Neural Netw.*, vol. 21, no. 11, pp. 1817–1830, Nov. 2010.

[34] R. Gütig and H. Sompolinsky, “The tempotron: A neuron that learns spike timing-based decisions,” *Nat. Neurosci.*, vol. 9, no. 3, p. nn1643, Feb. 2006.

[35] F. Ponulak and A. Kasiński, “Supervised learning in spiking neural networks with ReSuMe: Sequence learning, classification, and spike shifting,” *Neural Comput.*, vol. 22, no. 2, pp. 467–510, Oct. 2009.

[36] S. G. Wysoski, L. Benuskova, and N. Kasabov, “Fast and adaptive network of spiking neurons for multi-view visual pattern recognition,” *Neurocomputing*, vol. 71, no. 13, pp. 2563–2575, Aug. 2008.

[37] J. M. Brader, W. Senn, and S. Fusi, “Learning real-world stimuli in a neural network with spike-driven synaptic dynamics,” *Neural Comput.*, vol. 19, no. 11, pp. 2881–2912, Sep. 2007.

[38] J. Dethier, P. Nuyujukian, S. I. Ryu, K. V. Shenoy, and K. Boahen, “Design and validation of a real-time spiking-neural-network decoder for brain-machine interfaces,” *J. Neural Eng.*, vol. 10, no. 3, p. 036008, Jun. 2013.

[39] P. O’Connor, D. Neil, S.-C. Liu, T. Delbruck, and M. Pfeiffer, “Real-time classification and sensor fusion with a spiking deep belief network,” *Front. Neurosci.*, vol. 7, 2013.

[40] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[41] E. Neftci, S. Das, B. Pedroni, K. Kreutz-Delgado, and G. Cauwenberghs, “Event-driven contrastive divergence for spiking neuromorphic systems,” *Front. Neurosci.*, vol. 7, 2014.

[42] Y. Cao, Y. Chen, and D. Khosla, “Spiking deep convolutional neural networks for energy-efficient object recognition,” *Int. J. Comput. Vis.*, vol. 113, no. 1, pp. 54–66, May 2015.

[43] E. D. Adrian and Y. Zotterman, “The impulses produced by sensory nerve-endings,” *J. Physiol.*, vol. 61, no. 2, pp. 151–171, Apr. 1926.

[44] J. A. Pérez-Carrasco *et al.*, “Mapping from frame-driven to frame-free event-driven vision systems by low-rate rate coding and coincidence processing-

application to feedforward ConvNets," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 11, pp. 2706–2719, Nov. 2013.

[45] C. Keysers, D.-K. Xiao, P. Földiák, and D. I. Perrett, "The speed of sight," *J Cogn. Neurosci.*, vol. 13, no. 1, pp. 90–101, Jan. 2001.

[46] F. Rieke, D. Warland, R. de R. van Steveninck, and W. Bialek, *Spikes: Exploring the Neural Code*, Reprint edition. A Bradford Book, 1999.

[47] W. Gerstner, "Population dynamics of spiking neurons: fast transients, asynchronous states, and locking," *Neural Comput.*, vol. 12, no. 1, pp. 43–89, Jan. 2000.

[48] T. Gollisch and M. Meister, "Rapid neural coding in the retina with relative spike latencies," *Science*, vol. 319, no. 5866, pp. 1108–1111, Feb. 2008.

[49] P. Reinagel and R. C. Reid, "Temporal coding of visual information in the thalamus," *J. Neurosci.*, vol. 20, no. 14, pp. 5392–5400, Jul. 2000.

[50] W. Bair and C. Koch, "Temporal precision of spike trains in extrastriate cortex of the behaving macaque monkey," *Neural Comput.*, vol. 8, no. 6, pp. 1185–1202, Aug. 1996.

[51] C. Weng *et al.*, "Temporal precision in the neural code and the timescales of natural vision," *Nature*, vol. 449, no. 7158, p. 92, Sep. 2007.

[52] J. Y. Shih, C. A. Atencio, and C. E. Schreiner, "Improved stimulus representation by short interspike intervals in primary auditory cortex," *J. Neurophysiol.*, vol. 105, no. 4, pp. 1908–1917, Apr. 2011.

[53] W. M. Usrey, J.-M. Alonso, and R. C. Reid, "Synaptic interactions between thalamic inputs to simple cells in cat visual cortex," *J. Neurosci.*, vol. 20, no. 14, pp. 5461–5467, Jul. 2000.

[54] W. M. Usrey, J. B. Reppas, and R. C. Reid, "Paired-spike interactions and synaptic efficacy of retinal inputs to the thalamus," *Nature*, vol. 395, no. 6700, pp. 384–387, Sep. 1998.

[55] A. Steimer and R. Douglas, "Spike-based probabilistic inference in analog graphical models using interspike-interval coding," *Neural Comput.*, vol. 25, no. 9, pp. 2303–2354, Sep. 2013.

[56] G. D. Forney, "Codes on graphs: Normal realizations," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 520–548, Feb. 2001.

[57] H. A. Loeliger, "An introduction to factor graphs," *IEEE Signal Process. Mag.*, vol. 21, no. 1, pp. 28–41, Jan. 2004.

[58] H. A. Loeliger, J. Dauwels, J. Hu, S. Korl, L. Ping, and F. R. Kschischang, "The factor graph approach to model-based signal processing," *Proc. IEEE*, vol. 95, no. 6, pp. 1295–1322, Jun. 2007.

[59] G. Indiveri *et al.*, "Neuromorphic silicon neuron circuits," *Front. Neurosci.*, vol. 5, 2011.

[60] J. von Neumann, "First draft of a report on the EDVAC," *IEEE Ann. Hist. Comput.*, vol. 15, no. 4, pp. 27–75, 1993.

[61] J. H. Wang, C. T. Tang, and H. Chen, "An adaptable continuous restricted boltzmann machine in VLSI for fusing the sensory data of an electronic nose," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 4, pp. 961–974, Apr. 2017.

[62] K. T. Tang *et al.*, "A 0.5V 1.27mW nose-on-a-chip for rapid diagnosis of ventilator-associated pneumonia," in *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, 2014, pp. 420–421.

[63] P. Knag, J. K. Kim, T. Chen, and Z. Zhang, "A sparse coding neural network ASIC with on-chip learning for feature extraction and encoding," *IEEE J. Solid-State Circuits*, vol. 50, no. 4, pp. 1070–1079, Apr. 2015.

[64] R. Wang, C. S. Thakur, G. Cohen, T. J. Hamilton, J. Tapson, and A. van Schaik, "Neuromorphic hardware architecture using the neural engineering framework for pattern recognition," *IEEE Trans. Biomed. Circuits Syst.*, vol. 11, no. 3, pp. 574–584, Jun. 2017.

[65] C. Farabet, B. Martini, P. Akselrod, S. Talay, Y. LeCun, and E. Culurciello, "Hardware accelerated convolutional neural networks for synthetic vision systems," in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, 2010, pp. 257–260.

[66] D. Neil and S. C. Liu, "Minitaur, an event-driven FPGA-based spiking network accelerator," *IEEE Trans. Very Large Scale Integr. VLSI Syst.*, vol. 22, no. 12, pp. 2621–2628, Dec. 2014.

[67] S. Mitra, S. Fusi, and G. Indiveri, "Real-time classification of complex patterns using spike-based learning in neuromorphic VLSI," *IEEE Trans. Biomed. Circuits Syst.*, vol. 3, no. 1, pp. 32–42, Feb. 2009.

[68] B. V. Benjamin *et al.*, "Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations," *Proc. IEEE*, vol. 102, no. 5, pp. 699–716, May 2014.

[69] S. B. Furber, F. Galluppi, S. Temple, and L. A. Plana, "The SpiNNaker project," *Proc. IEEE*, vol. 102, no. 5, pp. 652–665, May 2014.

[70] N. Qiao *et al.*, "A reconfigurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128K synapses," *Front. Neurosci.*, vol. 9, 2015.

[71] F. Akopyan *et al.*, "TrueNorth: Design and Tool Flow of a 65 mW 1 Million Neuron Programmable Neurosynaptic Chip," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 34, no. 10, pp. 1537–1557, Oct. 2015.

[72] B. U. Pedroni *et al.*, "Mapping generative models onto a network of digital spiking neurons," *IEEE Trans. Biomed. Circuits Syst.*, vol. 10, no. 4, pp. 837–854, Aug. 2016.

[73] D. Pecevski, L. Buesing, and W. Maass, "Probabilistic inference in general graphical models through sampling in stochastic networks of spiking neurons," *PLOS Comput. Biol.*, vol. 7, no. 12, p. e1002294, Dec. 2011.

[74] L. Buesing, J. Bill, B. Nessler, and W. Maass, "Neural dynamics as sampling: A model for stochastic computation in recurrent networks of spiking neurons," *PLoS Comput. Biol.*, vol. 7, no. 11, p. e1002211, Nov. 2011.

[75] W. Maass, "Noise as a resource for computation and learning in networks of spiking neurons," *Proc. IEEE*, vol. 102, no. 5, pp. 860–880, May 2014.

[76] M. A. Petrovici, J. Bill, I. Bytschok, J. Schemmel, and K. Meier, "Stochastic inference with deterministic spiking neurons," *ArXiv13113211 Cond-Mat Physicsphysics Q-Bio Stat*, Nov. 2013.

[77] D. R. Mendat, S. Chin, S. Furber, and A. G. Andreou, "Neuromorphic sampling on the SpiNNaker and parallella chip multiprocessors," in *2016 IEEE 7th Latin American Symposium on Circuits Systems (LASCAS)*, 2016, pp. 399–402.

[78] D. R. Mendat, S. Chin, S. Furber, and A. G. Andreou, "Markov chain Monte Carlo inference on graphical models using event-based processing on the spinnaker neuromorphic architecture," in *2015 49th Annual Conference on Information Sciences and Systems (CISS)*, 2015, pp. 1–6.

[79] H. A. Loeliger, F. Lustenberger, M. Helfenstein, and F. Tarkoy, "Probability propagation and decoding in analog VLSI," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 837–843, Feb. 2001.

[80] S. Hemati and A. H. Banihashemi, "Iterative decoding in analog CMOS," in *Proceedings of the 13th ACM Great Lakes Symposium on VLSI*, New York, NY, USA, 2003, pp. 15–20.

[81] T. J. Hamilton, S. Afshar, A. van Schaik, and J. Tapson, "Stochastic electronics: A neuro-inspired design paradigm for integrated circuits," *Proc. IEEE*, vol. 102, no. 5, pp. 843–859, May 2014.

[82] C. H. Chien, S. C. Liu, and A. Steimer, "A neuromorphic VLSI circuit for spike-based random sampling," *IEEE Trans. Emerg. Top. Comput.*, vol. PP, no. 99, pp. 1–1, 2016.

[83] C. H. Chien, L. Longinotti, A. Steimer, and S. C. Liu, "Hardware implementation of an event-based message passing graphical model network," *IEEE Trans. Circuits Syst. Regul. Pap.*, vol. PP, no. 99, pp. 1–14, 2018.

[84] A. Doucet, S. Godsill, and C. Andrieu, "On sequential Monte Carlo sampling methods for Bayesian filtering," *Stat. Comput.*, vol. 10, no. 3, pp. 197–208, Jul. 2000.

[85] O. Cappe, S. J. Godsill, and E. Moulines, "An overview of existing methods and recent advances in sequential Monte Carlo," *Proc. IEEE*, vol. 95, no. 5, pp. 899–924, May 2007.

[86] M. E. Larkum, J. J. Zhu, and B. Sakmann, "Dendritic mechanisms underlying the coupling of the dendritic with the axonal action potential initiation zone of adult rat layer 5 pyramidal neurons," *J. Physiol.*, vol. 533, no. Pt 2, pp. 447–466, Jun. 2001.

[87] D. R. Cox and P. A. W. Lewis, *The Statistical Analysis of Series of Events*. Chapman and Hall, 1966.

[88] S. M. Ross, *A First Course in Probability*. Pearson Prentice Hall, 2010.

[89] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. Wiley-Interscience, 2006.

[90] R. E. Kalman, "A new approach to linear filtering and prediction problems," *J. Basic Eng.*, vol. 82, no. 1, pp. 35–45, Mar. 1960.

[91] R. Faragher, “Understanding the basis of the Kalman filter via a simple and intuitive derivation,” *IEEE Signal Process. Mag.*, vol. 29, no. 5, pp. 128–132, Sep. 2012.

[92] K. A. Boahen, “A burst-mode word-serial address-event link-I: Transmitter design,” *IEEE Trans. Circuits Syst. Regul. Pap.*, vol. 51, no. 7, pp. 1269–1280, Jul. 2004.

[93] K. A. Boahen, “A burst-mode word-serial address-event link-II: receiver design,” *IEEE Trans. Circuits Syst. Regul. Pap.*, vol. 51, no. 7, pp. 1281–1291, Jul. 2004.

[94] K. A. Boahen, “A burst-mode word-serial address-event link-III: analysis and test results,” *IEEE Trans. Circuits Syst. Regul. Pap.*, vol. 51, no. 7, pp. 1292–1300, Jul. 2004.

[95] K. A. Boahen, “Communicating Neuronal Ensembles between Neuromorphic Chips,” in *Neuromorphic Systems Engineering*, Springer, Boston, MA, 1998, pp. 229–259.

[96] G. Cauwenberghs, “Delta-sigma cellular automata for analog VLSI random vector generation,” *IEEE Trans. Circuits Syst. II Analog Digit. Signal Process.*, vol. 46, no. 3, pp. 240–250, Mar. 1999.

[97] S.-C. Liu, J. Kramer, G. Indiveri, T. Delbruck, and R. Douglas, *Analog VLSI: Circuits and Principles*. Cambridge, Mass: A Bradford Book, 2002.

[98] B. Linares-Barranco and T. Serrano-Gotarredona, “On the design and characterization of femtoampere current-mode circuits,” *IEEE J. Solid-State Circuits*, vol. 38, no. 8, pp. 1353–1363, Aug. 2003.

[99] B. Razavi, *Design of Analog CMOS Integrated Circuits*. McGraw-Hill Education, 2000.

[100] W. T. Holman, J. A. Connelly, and A. B. Dowlatabadi, “An integrated analog/digital random noise source,” *IEEE Trans. Circuits Syst. Fundam. Theory Appl.*, vol. 44, no. 6, pp. 521–528, Jun. 1997.

[101] M. Bucci and R. Luzzi, “Fully digital random bit generators for cryptographic applications,” *IEEE Trans. Circuits Syst. Regul. Pap.*, vol. 55, no. 3, pp. 861–875, Apr. 2008.

[102] C.-C. Wang, J.-M. Huang, H.-C. Cheng, and R. Hu, “Switched-current 3-bit CMOS 4.0-MHz wideband random signal generator,” *IEEE J. Solid-State Circuits*, vol. 40, no. 6, pp. 1360–1365, Jun. 2005.

[103] P. Xu, Y. l Wong, T. K. Horiuchi, and P. A. Abshire, “Compact floating-gate true random number generator,” *Electron. Lett.*, vol. 42, no. 23, pp. 1346–1347, Nov. 2006.

[104] P. Dudek and V. D. Juncu, “An area and power efficient discrete-time chaos generator circuit,” in *Proceedings of the 2005 European Conference on Circuit Theory and Design, 2005.*, 2005, vol. 2, p. II/87-II/90 vol. 2.

[105] J. Holleman, S. Bridges, B. P. Otis, and C. Diorio, “A 3 uW CMOS true random number generator with adaptive floating-gate offset cancellation,” *IEEE J. Solid-State Circuits*, vol. 43, no. 5, pp. 1324–1336, May 2008.

[106] B. Marr and J. Hasler, “Compiling probabilistic, bio-inspired circuits on a field programmable analog array,” *Front. Neurosci.*, vol. 8, 2014.

[107] T. Clayton *et al.*, “An implementation of a spike-response model with escape noise using an avalanche diode,” *IEEE Trans. Biomed. Circuits Syst.*, vol. 5, no. 3, pp. 231–243, Jun. 2011.

[108] M. Yang, C. H. Chien, T. Delbruck, and S. C. Liu, “A 0.5V 55uW 64x2-channel binaural silicon cochlea for event-driven stereo-audio sensing,” in *2016 IEEE International Solid-State Circuits Conference (ISSCC)*, 2016, pp. 388–389.

Curriculum vitae

Chen-Han Chien

Born: 27 May, 1984, Taiwan

Citizenship: Taiwan, Republic of China (R.O.C.)

Contact: shinta0527@gmail.com

Education

03.2012 - 02.2018	Ph.D student, Information Technology and Electrical Engineering, Swiss Federal Institute of Technology in Zurich (ETH Zurich), Switzerland
09.2006 - 10.2008	Master of Science, Electrical Engineering, National Tsing Hua University (NTHU), Taiwan
09.2002 - 06.2006	Bachelor of Science, Electrical Engineering, National Tsing Hua University (NTHU), Taiwan

Work Experience

03.2012 - 02.2017	Research PhD Student, Institute of Neuroinformatics, ETH Zurich and UZH
02.2014 - 07.2014	Teaching Assistant for Neuromorphic Engineering, Institute of Neuroinformatics, ETH Zurich and UZH
02.2010 - 02.2011	Hardware Engineer, ZyFLEX (subsidiary company of ZyXEL), Taiwan

10.2008 - 09.2009 Compulsory Military Service, Second Lieutenant of the Army, Taiwan

09.2006 - 08.2007 Teaching Assistant for Analog IC Design, Department of Electrical Engineering, National Tsing Hua University, Taiwan

IC Design Project

2016 Event Based CMOS Circuit for a Class of Belief-Propagation Models

2015 A Fully Differential LNA with 0.5V Supply Voltage in "A 0.5V 55uW 64x2-channel Binaural Silicon Cochlea for Event-driven Stereo-audio Sensing" project

2013 A Neuromorphic VLSI Circuit for Spike-Based Random Sampling

2008 A Stochastic System on a Chip basing on the Diffusion Network

2006 Delta-sigma Cellular Automata for Analog VLSI Random Vector Generation

Publication

C. H. Chien; L. Longinotti; A. Steimer; S. C. Liu, 'Hardware Implementation of an Event-Based Message Passing Graphical Model Network', *IEEE Trans. Circuits Syst. Regul. Pap.*, vol. PP, no. 99, pp. 1–14, 2018.

C. H. Chien, S. C. Liu, and A. Steimer, "A neuromorphic VLSI circuit for spike-based random sampling," *IEEE Trans. Emerg. Top. Comput.*, vol. PP, no. 99, pp. 1–1, 2016.

C. H. Chien, C. C. Lu and H. Chen, "Mapping the Diffusion Network into a stochastic system in Very Large Scale Integration," in *The 2010 International Joint Conference on Neural Networks (IJCNN)*, 2010, pp. 1-7.

M. Yang, C. H. Chien, T. Delbruck, and S. C. Liu, "A 0.5 V 55 uW 64 x 2 channel binaural silicon cochlea for event-driven stereo-audio sensing," *IEEE J. Solid-State Circuits*, vol. 51, no. 11, pp. 2554–2569, Nov. 2016

M. Yang, C. H. Chien, T. Delbruck, and S. C. Liu, "A 0.5V 55uW 64x2-channel binaural silicon cochlea for event-driven stereo-audio sensing," in *2016 IEEE International Solid-State Circuits Conference (ISSCC)*, 2016, pp. 388–389.