FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Address-Event Based Communication Between Spiking Neural Networks (SNN) Computing Cores

Rodrigo Miguel Guerra da Mota de Almeida Azevedo

Master in Electrical and Computers Engineering

Supervisor: Vítor Grade Tavares

October 31, 2023

Abstract

Impulse-based computing, currently at its peak in spiking neural networks (SNNs), is emerging as the best candidate for the way information is interpreted and analysed by neural networks. Currently, there are some current real-world applications of SNNs in real-time image and audio processing, for example in computer vision, even though the literature on practical applications remains sparse. The massive dimension and complexity that these new spiking neural networks can assume, raises important challenges in hardware design, as it must be able to deal with the number of interconnections that grow exponentially with the network complexity, as well as the amount of memory required. The concept of a division of a neuromorphic processor into cores that facilitates the realization of these systems seeks to resolve this computing issue. The complexity of the network can then grow by including multiple cores on a single integrated circuit. This segmented design is best supported by the implementation of a network-on-chip (NoC) protocol based on Address-Event Representation (AER) for inter-core and inter-chip communication. This work exclusively focuses on the thorough examination and enhancement of the AER process to achieve efficient and scalable communication, without delving into its integration or connection with the NoC. The results obtained from this work demonstrate the feasibility of designing a chip capable of accommodating the growing complexity of spiking neural networks. Furthermore, this research has showcased that multiple approaches, including those with and without serialization, are not only achievable but also offer promising avenues for the realization of scalable and complex SNNs. The results of this work show that multiple fully-digital approaches can be used as possible solutions to this intercore communication problem.

Resumo

A computação baseada em impulsos, atualmente no seu auge nas *Spiking Neural Networks* (SNN), está a emergir como a melhor candidata para a forma como a informação é interpretada e analisada por redes neuronais. Atualmente, existem algumas aplicações reais de SNNs no processamento de imagem e áudio em tempo real, por exemplo, na visão computacional, porém a literatura sobre estas aplicações práticas continua a ser escassa. A enorme dimensão e complexidade que estas novas redes neuronais *spiking* podem assumir levanta importantes desafios ao nível da conceção do hardware, uma vez que este deve ser capaz de lidar com o número de interligações de neurónios que cresce exponencialmente com a complexidade da rede, bem como com a quantidade de memória necessária. O conceito de dividir um processador neuromórfico em núcleos visa solucionar os desafios computacionais inerentes a esses sistemas. A complexidade da rede pode então aumentar com a inclusão de vários núcleos num único circuito integrado. Este design segmentado é melhor suportado pela implementação de um protocolo de *Network-on-Chip* (NoC) baseado na *Address-Event Representation* (AER) para comunicação entre núcleos e entre chips. Este trabalho foca-se no estudo e otimização do processo AER para uma comunicação eficiente, abrindo caminho para SNNs escaláveis e complexas. Os resultados obtidos com este trabalho demonstram a viabilidade de conceber um chip capaz de acomodar a crescente complexidade das redes neuronais. Além disso, esta pesquisa mostrou que várias abordagens, incluindo aquelas com e sem serialização, não só são viáveis, mas também oferecem caminhos promissores para a realização de SNNs escaláveis e complexas. Os resultados deste trabalho mostram que várias abordagens totalmente digitais podem ser usadas como possíveis soluções para este problema de comunicação entre núcleos.

Agradecimentos

I would like to express my heartfelt appreciation to those who have played a role in this final academic journey that was my dissertation.
My sincere gratitude goes to Professor Vítor Grade Tavares for all the guidance, mentorship, and patience. Their expertise has been invaluable in shaping my research.
I also want to thank the Ph.D. students in the microelectronics laboratory that helped me with their insightful advice, feedback and support throughout this process.
To my family, your love and encouragement have been my anchor. Thank you for standing by me.
To my friends, thank you for the good times and the provided much-needed breaks and energy to continue this journey.
Lastly, I extend my thanks to all who have contributed in various ways to my success. Your support has been a driving force.

With gratitude,

Rodrigo Azevedo

This work was partially supported by European Structural and Investment Funds in the FEDER component, through the Operational Competitiveness and Internationalization Programme (COMPETE 2020) [Project no 047264; Funding Reference: POCI-01-0247-FEDER-047264]

‘"It may seem difficult at first, but everything is difficult at first." ’

Miyamoto Musashi

Contents

List of Figures

List of Tables

Abbreviations and Symbols

ACK	Acknowledgement Signal
AER	Address-Event Representation
AI	Artificial Intelligence
ANN	Artificial Neural Network
ASEC	Asynchronous Spike Event Coding
ASIC	Application Specific Integrated Circuit
CAD	Computer-Aided Design
CASE	Computer-Aided Software Engineering
CDC	Clock Domain Crossing
CLK	Clock signal
CMOS	Complementary Metal-Oxide Semiconductor
CTS	Clock Tree Synthesis
DLL	Delay Locked-Loop
DRC	Design Rule Checking
EDA	Electronic Design Automation
EMI	Electromagnetic Interference
FF	Flip-Flop
FIFO	First In - First Out
FPAA	Field-Programmable Analogue Array
FPGA	Field-Programmable Gate Array
FSM	Finite-State Machine
GPU	Graphics Processing Unit
HDL	Hardware Description Language
IDE	Integrated Development Environment
I/O	Input/Output
IP	Intellectual Property
LVDS	Low Voltage Differential Signalling
LUT	Look-up Table
MMMC	Multi-Mode Multi-Corner
NC	Neuromorphic Computing
NoC	Network on Chip
PISO	Parallel In Serial Out
PLL	Phase Locked-Loop
PVT	Process, Voltage, Temperature
REQ	Request Signal
RTL	Register Transfer Level
RX	Receiver
SDF	Standard Delay Format
SerDes	Serializer-Deserializer chip or part of an FPGA
SIPO	Serial In Parallel Out

SNN	Spiking Neural Networks
STPD	Spike-Timing Dependent Plasticity
TB	Testbench
TDM	Time Domain Multiplexing
TX	Transmitter
VLSI	Very-Large-Scale Integration

Chapter 1

Introduction

1.1 Context

Over the last two decades, the field of Machine Learning (ML) has evolved so much to meet the ever-increasing demands of complex applications. To execute these applications, hardware developments were also done in order to meet these objectives, be it the need for more memory and quicker processing. The Von Neumann architecture, which is the foundational architecture for most conventional computers and computing systems today, possesses some limitations that prevent it from being an ideal solution to run these ML applications. These limitations include memory bottlenecking due to the shared bus for the memory and computer processing unit (CPU), and very limited parallelism, due to its sequential nature, with instructions executed one after another.

Graphics Processing Units (GPUs) are not built using this architecture and do not possess these limitations and issues. GPUs are designed with a different architecture known as the SIMD (Single Instruction, Multiple Data) architecture or parallel processing architecture. This architecture is optimized for handling massive parallel computations, which is especially useful for parallel computing tasks like ML applications, making them the most well-suited solution for this field. And this is the reason why GPUs have witnessed their own advancements in recent years, as to cater specifically to the demands of machine learning. And although they offer a high degree of parallelization that allows for quicker processing, they fall very short when it comes to power efficiency when compared to specialized hardware solutions.

These specialized hardware solutions are designed from the ground up to optimize power efficiency and performance for machine learning tasks. They often incorporate custom architectures and dedicated hardware components for specific machine-learning operations. By tailoring the hardware to the specific requirements of machine learning, these solutions can achieve significant improvements in power efficiency while still providing high-performance capabilities.

This is the reason why, in recent years, neuromorphic computing has been a fast-growing research field that has gained a lot of focus. In this field, the main goal is the design of hardware systems that mimic neural architectures. It is known that advancements in this field are sure to

improve how computers interpret and analyse information. Recently, impulse-based computing, such as Spiking Neural Networks (SNNs), has emerged as the best candidate for higher-level processing [7].

This is because neuromorphic systems running this type of computing have the potential to be much more energy efficient than traditional computers. Proof of this efficiency is our very own brain, which is able to perform complex computations using very little energy, and these developed neuromorphic systems are designed to mimic this efficiency. Furthermore, neuromorphic computing systems may be better at handling certain types of tasks and data than traditional computers. For example, they may be better at tasks that involve learning from experience, pattern recognition, or real-time processing, making them ideal for the greater objective of achieving Automated Perception, this work is inserted in.

1.2 Problem and Motivation

Spiking Neural Nets (SNNs) exhibit a remarkably high degree of parallelism and asynchronous nature, like the biological cognitive and sensory systems. These systems are much more efficient and have the potential to be much more computationally powerful than current computer architectures. But a problem arises when trying to develop such systems, in comparison to natural biological systems, where neuron cells are very densely interconnected, and where each cell may have thousands or even tens of thousands of connections in the 3-D space, cutting-edge semiconductor technologies have limited physical interconnection resources that do not allow for this. This limits the physical interconnection of neurons in 2-D silicon systems to practically only a few neighbours.

However, since the available bandwidth in the interconnect paths of these systems is significantly higher than in the biological ones, bus sharing, such as the one supported by Address Event Representation encoding (AER), can be used to overcome this limitation. The main observation that works supports this AER protocol is that neurons fire ‘events’ or spikes at a low frequency (biological neurons might have average firing rates of 100 Hz) when compared to the available bandwidth of the shared bus. Algorithms based both on continuous-valued “intracellular” signals and discrete spiking events have already been created in this way, and while analogue computations may be performed better at the cellular level, we argue that it is advantageous to implement spike-based learning rules in the address domain for large scale systems.

Many research groups are currently investigating communication protocols for the electronic implementation of neural processing. One approach, the address-event representation (AER) protocol, has attracted the interest of several research groups, and has presented itself as the best architecture suitable for large asynchronous circuits such as the ones running the SNNs.

Previous research has extensively demonstrated the potential of Address Event Representation (AER) in constructing large-scale networks with versatile synaptic connectivity. These networks exhibit comparable performance to those constructed using conventional pair connectivity. The key reason behind this success lies in the inherent scalability of AER-based systems.

A notable advantage of AER is its ability to accommodate diverse types of signals represented as spikes. Unlike other approaches that make assumptions about the nature of the encoded signals, AER can effectively handle any measure of cellular activity. This remarkable flexibility enables the development of learning mechanisms that closely mimic the biological processes observed in living organisms, thereby achieving a high degree of biological realism.

By leveraging AER, researchers can design networks with configurable synaptic connections, allowing for precise control over the connectivity patterns. This reconfigurability grants the freedom to experiment with various network architectures, enabling investigations into the impact of specific connectivity motifs on network dynamics and functionality. Additionally, AER facilitates the implementation of learning algorithms that can adapt to a wide range of signal representations, making it an attractive framework for exploring the computational capabilities of neural systems.

The scalability and flexibility offered by AER-based systems have far-reaching implications for neural network research. These systems provide a pathway to constructing large-scale networks that can emulate the intricate connectivity and learning mechanisms observed in biological systems. By facilitating the integration of complex learning algorithms, AER empowers researchers to delve deeper into the mechanisms underlying cognitive processes and develop biologically inspired artificial intelligence models. Ultimately, the use of AER holds great promise in advancing our understanding of neural systems and paving the way for the development of highly efficient and adaptable computational architectures.

1.3 Goals

The goal of this dissertation was to design, develop and evaluate hardware capable of implementing the AER protocol in a neuromorphic system with learning founded on this notion of spike-timing-dependent plasticity (STDP). Taking that into consideration, the objective can be broken up into smaller goals and points:

- **Data Transmission**: Efficient and low-latency transmission and reception of address events are critical in scenarios where preserving timing information is crucial, such as in the context of Spike-Timing-Dependent Plasticity (STDP) neurons. To achieve this objective, it is essential to minimize processing overhead to avoid introducing additional latency. By doing so, the timing information carried by the address events can be accurately maintained for subsequent utilization by STDP neurons.

- **Data processing**: Although requiring little processing, the solution will need to address events quickly and accurately, in order to extract the relevant information from them. This might involve using specialized circuits to perform tasks such as address decoding, and event buffering.

- **Power Considerations**: Addressing power consumption management challenges associated with the hardware implementation. Implementing energy-efficient circuit designs, and power management techniques.

- **Evaluation and Performance Analysis**: Conducting thorough evaluations and performance analysis of the developed hardware solution. This includes benchmarking the system's performance in terms of spike rate compatibility and efficiency in data transmission.

In summary, the design of hardware for AER-based neuromorphic systems necessitates a careful balance between high-speed digital signalling, efficient data processing, and effective power and thermal management. By addressing these considerations, we can create robust and scalable hardware architectures that effectively harness the power of AER for advanced neuromorphic computing applications.

1.4 Structure of Dissertation

Besides the introduction, this dissertation contains 5 more chapters.

Chapter 2 of the work provides an overview of the theoretical aspects and fundamental knowledge that are crucial for understanding the scientific or engineering area within which the research is contextualized. This chapter aims to establish a solid foundation of knowledge that readers can refer to throughout the rest of the work.

Chapter 3 serves as a platform to discuss the state-of-the-art and existing solutions within the scientific community. It is within this chapter that the researcher sets objectives for the preliminary work, building upon the knowledge and insights gained from the previous chapter. The primary goal of Chapter 3 is to provide readers with an in-depth understanding of the current state of research in the field. By reviewing the existing body of work, the chapter highlights the gaps, limitations, and opportunities for further exploration and improvement.

Chapter 4 focuses on the development and results of the solution. It outlines the objectives and requirements that guided the development process and explains the step-by-step creation of the solution. The chapter also highlights any challenges encountered during development and discusses the outcomes achieved through implementation. It emphasizes the relationship between the solution and the observed results, including any unexpected findings or areas for improvement. Additionally, it highlights the tools, technologies, and resources that have been employed.

Finally, in the concluding chapter, Chapter 5, the dissertation offers a comprehensive summary of the problem understanding, an analysis of the results obtained, and the contributions made to the scientific community. This chapter serves as the culmination of the research journey, providing a concise and conclusive overview of the conducted work and potential avenues for future research.

Chapter 2

Background

This section aims to provide a more detailed explanation of the concepts mentioned in the previous section. The objective is to enhance understanding of the forthcoming work and its development.

2.1 Neuromorphic Computing

Neuromorphic computing is a branch of computer science and engineering that aims to mimic the structure and functionality of the human brain by developing hardware and software systems inspired by biological neural networks. The key concept in neuromorphic computing is the use of spikes, also known as action potentials, which are the electrical signals used by neurons in the brain to communicate with each other. Spiking neural networks (SNNs) are computational models that capture the timing and rate of these spikes to process information.

In neuromorphic computing, spikes are used as a fundamental unit of communication and computation. Instead of representing information as binary values (0s and 1s) like in traditional digital computers, spiking neural networks represent information as spikes that occur at specific times. This event-driven processing allows for highly efficient and parallel computation, as spikes are only generated when relevant information is present, reducing the computational load and energy consumption.

2.2 Spiking Neural Networks (SNNs)

Spiking Neural Networks (SNNs) are a type of artificial neural network (ANN) inspired by the behaviour of biological neurons. Unlike traditional ANNs, which typically use continuous-valued activation functions and propagate information through real-valued activations, SNNs operate on discrete events called spikes, resembling the action potentials in biological neurons.

The fundamental unit of computation in an SNN is a spiking neuron, which accumulates input signals over time and generates output spikes when a certain threshold is reached. The timing of

spikes carries important information, allowing SNNs to represent and process temporal information naturally. This aspect is particularly useful for modelling time-varying and event-driven data, such as sensory inputs or sequential data.

SNNs are typically composed of interconnected layers of spiking neurons. Each neuron receives input spikes from its predecessors and propagates its output spikes to the neurons in the subsequent layer. The connections between neurons, known as synapses, have associated weights that regulate the strength of the connections between neurons. The output spike generation is determined by the neuron's activation function, which can be based on factors such as the current input, the neuron's membrane potential, or a combination of these factors. All of this described behaviour is depicted in Figure 2.1.

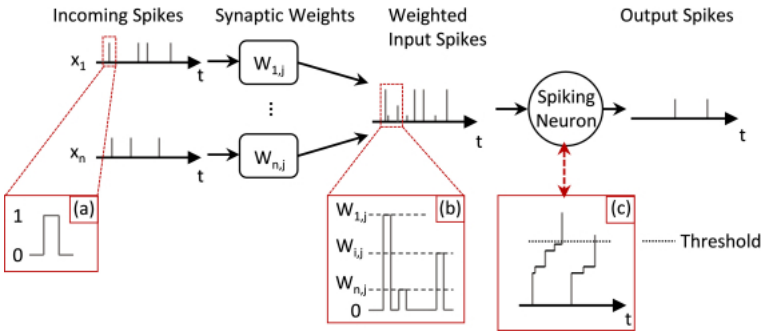


Figure 2.1: Schematic of SNN spike transmission flow [1]

The learning capabilities in SNNs are often achieved through a mechanism called Spike-Timing-Dependent Plasticity (STDP), which adjusts the synaptic weights based on the precise timing of pre- and post-synaptic spikes. STDP allows SNNs to adapt their connections based on the temporal relationships between spikes, facilitating the learning of temporal patterns and enabling unsupervised learning. SNNs have gained significant attention in neuromorphic computing due to their ability to process information in an event-driven and asynchronous manner, which is closer to how biological systems operate. They are particularly well-suited for tasks involving temporal data processing, such as sensory processing, time-series analysis, pattern recognition, and event-based control.

In recent years, there have been advancements in hardware implementations specifically designed for SNNs, such as neuromorphic chips, which aim to exploit the efficiency and parallelism offered by spiking computations. These developments have opened up new opportunities for energy-efficient and real-time neuromorphic computing systems based on SNNs.

2.3 Crossbar

The crossbar architecture plays a vital role in neuromorphic computing systems running SNNs, providing a versatile and densely interconnected network. Comprised of rows and columns of processing elements (see Figure 2.2), the crossbar acts as a programmable grid-like structure,

emulating synaptic connections found in biological neural networks. In the crossbar architecture, inputs and outputs are sent through rows and columns, respectively. These pulses, upon leaving the crossbar, are encoded into addresses using the Address Event Representation (AER) protocol. Inputs (presynaptic events) are encoded with their corresponding addresses and sent through rows, while outputs (postsynaptic events) generate event signals indicating spike occurrences and are sent through columns. This integration enables efficient and parallel communication of spike events within the crossbar architecture, closely resembling the synaptic connections and spike-based activity found in biological neural networks. The conductance of the memristors, the building block of these crossbars, can be dynamically adjusted to strengthen or weaken synaptic connections based on the timing of spike events, mimicking the spike-timing-dependent plasticity observed in biological systems.

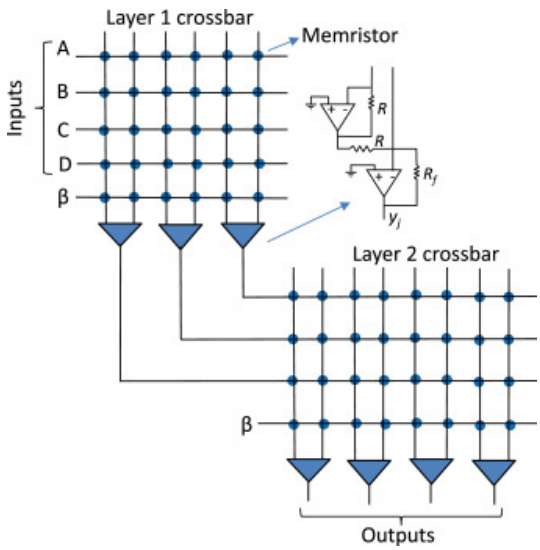


Figure 2.2: Multi-layer Memristors Crossbar schematic [2]

One significant advantage of the crossbar architecture is its compatibility with the Address Event Representation (AER) protocol. This integration of the crossbar architecture with the AER protocol offers several benefits. Firstly, the parallel nature of the crossbar enables the simultaneous processing of multiple spike events, effectively harnessing the massive parallelism observed in neural networks. The direct mapping of synaptic connections onto the crossbar eliminates the need for complex routing, enabling efficient spike communication and reducing latency in AER-based systems.

In essence, the AER protocol simplifies the communication process by using address and event signals, while the crossbar architecture enables parallel processing and efficient connectivity. Together, they enable spike-based communication within the neuromorphic system, facilitating real-time information exchange and efficient computation.

2.4 Time Domain Multiplexing (TDM)

Neuromorphic engineers have successfully adopted time-division multiplexing (TDM) as a technique to achieve massive connectivity in neuromorphic systems. TDM, which has been widely used in telecommunications [8] and computer networks [9], provides an efficient and scalable solution for connecting large numbers of neurons or neuromorphic components.

The inspiration for using TDM in neuromorphic systems comes from its proven success in areas where communication is key. TDM allows multiple signals to share a single transmission medium in these domains by allocating specific time slots to each signal (Seen in Figure 2.3). This enables efficient use of available resources and facilitates high-speed data transmission.

Similarly, in neuromorphic systems, TDM allows for the connection of numerous neurons or neuromorphic components by allocating specific time slots for their communication. Instead of using separate dedicated channels or connections for each neuron, TDM enables the sharing of limited resources among a large population of neurons, thereby achieving massive connectivity.

By using TDM, neuromorphic engineers can efficiently utilize the available resources, reduce wiring complexity, and enable high-density connectivity in neuromorphic systems. This approach not only enhances the scalability of neuromorphic architectures, but also facilitates the implementation of large-scale neural networks that can simulate complex brain-like functionalities.

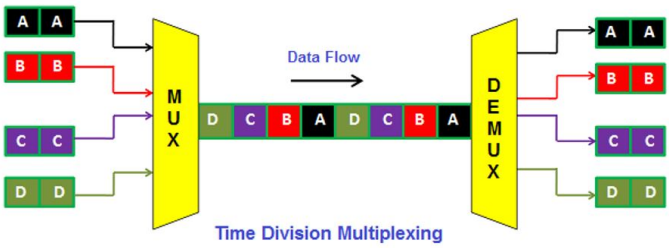


Figure 2.3: Time-Domain Multiplexing

2.5 Address-Event Representation protocol

The mentioned address-event representation (AER) is a protocol that uses TDM to overcome the extensive normal pair connectivity. In an AER system, activity is coded in the form of spikes and is transmitted from one array into the next. In AER, spikes are transmitted as fast as possible once they occur, and the time of the spike is not explicitly encoded, hence the saying that "time represents itself". If we were to make use of the normal approach to send these spike signals, we would use one wire for each pair of neurons, requiring N wire connections for N cell pairs connected. This is not a viable approach in a large complex system with a great number of neuron connections due to the limitations of interconnections of the current semiconductor technologies. The AER system overcomes this by encoding the location of the spiking neuron and transmitting it over a shared data bus, making it more scalable and ideal for larger systems. The inherent scalability of

AER arises from its event-driven nature. In traditional digital systems, information is processed sequentially and synchronized through a clock signal, limiting the system's scalability due to the need for global synchronization. In contrast, most AER systems operate asynchronously, with information encoded as events that are communicated between nodes only when relevant.

Like many asynchronous communication protocols, the handshaking signals REQ and ACK are used to ensure that only one cell pair is using the data bus at a time.

The REQ (Request) signal is used by a cell or entity that intends to transmit data or request a specific operation. When a cell wants to transmit data, it asserts the REQ signal to indicate its intention to use the data bus. The assertion of the REQ signal serves as a request for access to the data bus, informing other cells that the requesting cell wants to transmit data.

On the other hand, the ACK (Acknowledgement) signal is used to acknowledge the request made by a cell and grant access to the data bus. When a cell receives a REQ signal from another cell, it evaluates its own state to determine if it can grant access to the data bus. If the cell is not using the data bus and is ready to receive data, it asserts the ACK signal in response to the REQ signal. The asserting of the ACK signal acknowledges the request and grants permission for the requesting cell to use the data bus.

The handshaking process between cells typically involves a combination of the REQ and ACK signals. When a cell wants to transmit data, it asserts the REQ signal, indicating its desire to use the data bus. The receiving cell(s) monitor the REQ signal and respond with an ACK signal if they are available to receive the data. If multiple cells assert the REQ signal simultaneously, a priority scheme may be used to determine which cell gets access to the data bus. Once the requesting cell receives the ACK signal, it can proceed to transmit the desired data or perform the requested operation on the data bus. This ensures that only one cell pair has control of the bus at any given time, preventing conflicts and maintaining proper data integrity. After the data transmission is complete, the cells may release the bus, and the process repeats for subsequent data transmissions.

This handshaking process ensures proper synchronization and arbitration of the data bus, preventing conflicts and enabling reliable communication between cells in the system. This described behaviour is shown in figure 2.4.

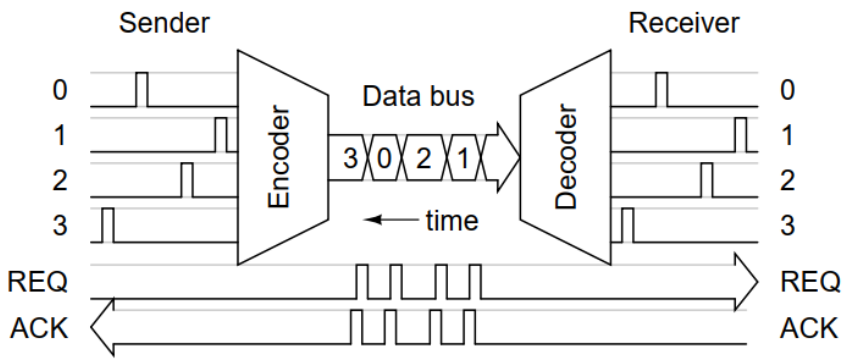


Figure 2.4: AER Behaviour [3]

This approach reduces the required number of interconnection wires from N to approximately $\log_2(N)$. The spike is uniquely composed by the location of its sender, which is explicitly encoded as an address, and the time that it occurred, which might not be explicitly encoded since events are communicated in real-time. This encoded spike is called an address event.

This address event carries information about the target address, which identifies the destination. The target address specifies the location or identifier of the neuron or processing element that should receive the event.

The source address, although not explicitly included in the address-event itself, is implicitly determined by the originating neuron. This is the reason why an address-event in the AER protocol primarily only includes the target address, which indicates where the event should be routed within the neuromorphic system.

So, in summary, the Address Event Representation (AER) protocol is considered a very efficient communication scheme used in the context of neuromorphic computing. It encodes the timing and location of spikes into digital signals. By transmitting spikes asynchronously and in parallel, if multiple channels are being used, AER enables low-latency communication between different components, closely resembling the communication patterns of biological neural networks.

2.6 Network-on-Chip (NoC)

Network-on-Chip (NoC) is a communication infrastructure used in integrated circuits (ICs) to facilitate efficient data transfer between different modules or processing resources within the chip. It replaces traditional bus-based architectures with a packet-switched network. It achieves this by incorporating multiple routing modules strategically placed throughout the circuit. These routing modules efficiently route information signals within the chip using packets and time-division multiplexing (TDM). In essence, NoCs function similarly to internet routing protocols. An example of a mesh topology is depicted in Figure 2.5, where each resource module is connected to an Interconnect IP (IIP).

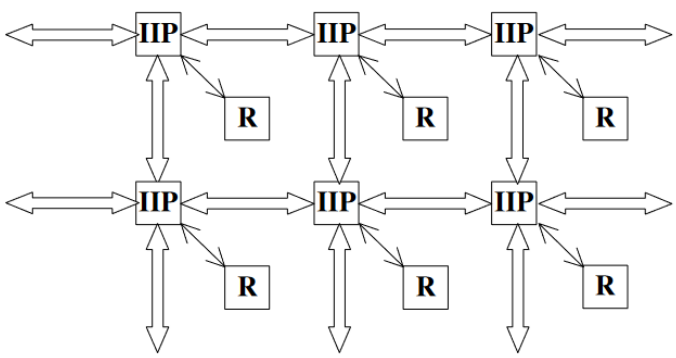


Figure 2.5: 2-D mesh topology of NoC [4]

Their usability is currently being studied in the context of neuromorphic computing. These NoCs provide connectivity by offering a scalable and flexible means of connecting the numerous processing elements, such as neurons and synapses, that comprise neuromorphic chips. Furthermore, NoCs offer low latency and high bandwidth communication channels, which are vital for real-time processing and rapid data transfer. This characteristic is particularly beneficial for neuromorphic chip designs, where quick and efficient communication between different elements is critical for achieving accurate neural network computations. This when paired with the AER protocol, due to its properties, allows for a great number of advantages such as the reduction of bandwidth requirements in interconnects, asynchronous communication of the system, and overall provides flexibility and scalability.

In this work, the primary focus will be on utilizing the AER (Address Event Representation) protocol to establish communication between chips running SNNs and interchip applications. The emphasis will be on exploring how AER can facilitate interchip communication, rather than delving deeply into the specific usage of AER within the NoC interconnects themselves. However, it is important to note that the AER protocol inherently offers a range of advantages to this connectivity realm, as previously mentioned.

2.7 Spike Timing-Dependent Plasticity (STDP)

Spike-timing-dependent plasticity is the biological process that adjusts the strength of connections between neurons in the brain. This process is based on the Hebbian learning theory. The change in strength of a connection bases itself on the relative timing of a particular neuron output and input spikes, these are also called presynaptic and postsynaptic events. Simply put, under an STDP process, if an input spike to a neuron tends to occur immediately before that neuron's output spike, then that particular input is made stronger, and if an input spike tends to occur immediately after an output spike, then that particular input is made weaker, hence the name: "spike-timing-dependent plasticity". This plasticity makes it so that inputs that might be the cause of the post-synaptic neuron's excitation are made even more likely to contribute to it in the future, whereas inputs that are not the cause of the post-synaptic spike are made less likely to contribute to it in the future. The amount of strengthening or weakening the synaptic connection goes through is dependent on the time interval between the two events, with the following formula being used to calculate the change in the weight of the connection:

$$\Delta w = \begin{cases} -\eta[\tau_- - (t_{\text{pre}} - t_{\text{post}})] & 0 \leq t_{\text{pre}} - t_{\text{post}} \leq \tau_- \\ \eta[\tau_+ + (t_{\text{pre}} - t_{\text{post}})] & -\tau_+ \leq t_{\text{pre}} - t_{\text{post}} \leq 0 \\ 0 & \text{otherwise} \end{cases}$$

Figure 2.6: Weight update formula [3]

where t_{pre} and t_{post} denote the time stamps of presynaptic and postsynaptic events.

This synaptic connection updating rule is achieved by further enhancing the AER architecture with two synaptic event queues. One for the presynaptic events and one for the postsynaptic ones. When an event occurs, its address is entered into the appropriate queue along with an associated value initialised to τ+ or τ-. This value is decremented over time [5]. Then for each type of event, we go through each queue backwards to find the correspondent spikes and increment/decrement the appropriate weights, strengthening or weakening the corresponding neuron connections.

2.8 Digital Design Methodology

The digital design flow plays a crucial role in the creation and implementation of digital systems, such as integrated circuits (ICs). It encompasses a series of steps and methodologies that are followed to transform an initial concept or idea into a fully functional digital product. Fundamentally understanding the digital design flow is essential because it serves as the foundation for the subsequent processes and methodologies discussed in this dissertation.

The digital design flow, excluding the processes related to manufacturing part, typically involves the following stages, represented in Figure 2.7:

- **Specification**: In this initial stage, the requirements and specifications of the digital system are defined. This includes determining the functionality, performance goals, power constraints, and other design specifications.

- **Architectural Design**: Based on the specifications, the system architecture is designed at a high level. This involves identifying the key components, their interconnections, and defining the overall structure of the system.

- **Logic Design**: The logical behaviour of the system is defined in this stage. It involves the creation of a digital circuit representation using hardware description languages (HDLs) like VHDL or Verilog. The design is divided into smaller modules or blocks, and the interconnections between these blocks are established.

- **Verification**: Once the logic design is complete, verification techniques are employed to ensure the correctness of the design. This involves performing various tests, simulations, and formal verification methods to detect and fix any design issues or bugs.

- **Synthesis**: In the synthesis stage, the RTL (Register Transfer Level) description of the design is transformed into a gate-level representation. Synthesis tools map the logical description into a network of standard cells from a chosen library, optimizing for factors like performance, area, and power consumption.

- **Physical Design**: Physical design deals with the layout and placement of the synthesized design onto a silicon die. It involves floor planning, placement of cells, routing of interconnects, and other physical optimization techniques. The output is a Physical Design File that represents the final layout of the design.

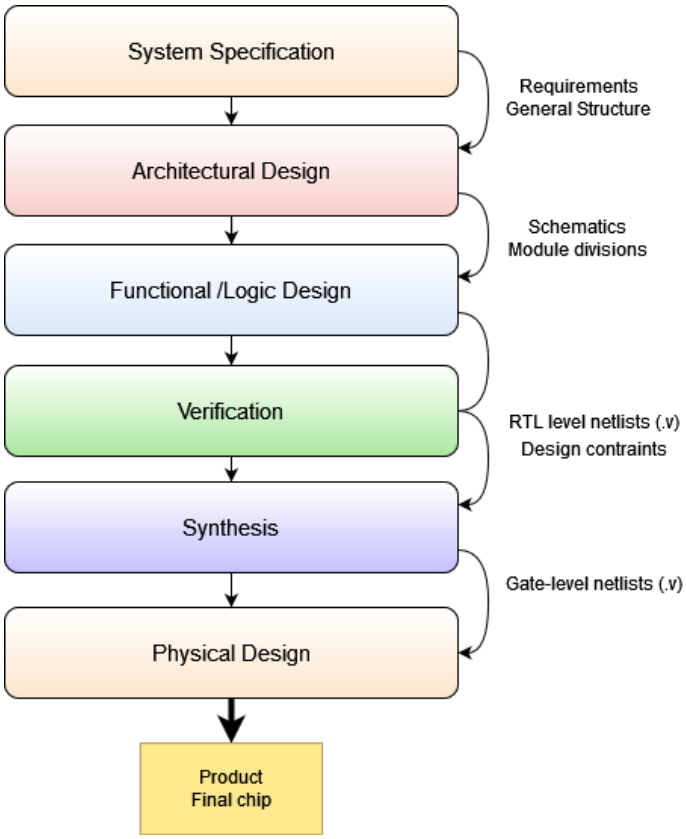


Figure 2.7: Digital Design Flow Schematic

These brief mentions of each stage of the process serve to provide an understanding of the entire design process and enable readers to follow the subsequent chapters that delve into specific aspects or methodologies within this flow. It sets the context for discussing topics related to digital system design throughout the dissertation. More specific details regarding the CAD (Computer-Aided Design) tools used in the digital design flow will be explained in Chapter 3 and 4.2. These chapters will delve into the software tools and methodologies employed to facilitate various stages of the design process.

2.9 Standard Cells

Standard cells play a pivotal role in modern digital design methodology, serving as pre-designed, reusable, and well-characterized digital logic building blocks. These cells consist of fundamental logic gates like NAND, NOR, AND, OR, flip-flops, and other combinational and sequential elements. Meticulously crafted to adhere to specific design rules and compatibility guidelines set by semiconductor foundries, their standardized nature allows seamless assembly and interconnection to form complex circuits and systems with remarkable efficiency.

Designed to optimize performance, area, and power consumption, standard cells strike a balance between these trade-offs. They offer predictable and consistent performance, simplifying the

design process and enabling foundries to achieve reliable production. The reusability of standard cells promotes design reuse, significantly reducing design time, verification efforts, and overall time-to-market for integrated circuits.

In the digital design methodology flow, standard cells come into play during the RTL (Register Transfer Level) synthesis stage, where Electronic Design Automation (EDA) tools convert high-level architectural specifications into RTL representations. At this point, the RTL code is mapped to specific standard cell instances from a technology library, each representing a predefined logic function. These cells are characterized and optimized for performance, area, and power consumption.

Subsequently, during RTL-to-Gate-Level synthesis, the design undergoes further optimization using various EDA tools, taking into account power, timing, and area considerations. The final step in the process involves the physical design phase, where the placement and routing of standard cell instances are meticulously determined to create the chip's layout.

Throughout the entire digital design methodology, standard cells ensure a streamlined and efficient process, facilitating design reuse, enabling quick iterations, and empowering designers to create advanced integrated circuits that meet the demands of modern technology. Their indispensable role in the semiconductor industry continues to drive innovation and progress, paving the way for increasingly sophisticated electronic devices.

Chapter 3

State-of-the-Art

In this chapter, a discussion of the various approaches by researchers and the scientific community are taken into consideration when developing their AER solutions is presented.

3.1 Introduction

In the last few years, there has been a great interest in the field of neuromorphic computing and exploring its potential for implementation. Studies have been conducted to investigate the AER protocol and its suitability for large-scale neuromorphic systems with a high number of neurons. These studies have led to significant advancements in neuromorphic hardware, paving the way for the development of more efficient, scalable, and capable AER-based systems. However, it is worth noting that detailed information regarding the specific implementations of these advancements is relatively scarce within the scientific community. While there has been progress in the design of AER hardware, the specific technical details and methodologies employed in these implementations are not extensively documented or readily available. This limited accessibility to implementation specifics may be due to various factors, including proprietary considerations, ongoing research and development efforts, or simply the fast-paced nature of the field.

Nonetheless, the research community continues to explore and propose new designs, learning rules, and network architectures that harness the unique capabilities of AER-based systems. While the exact implementation details may not be widely disseminated, the general trend suggests a focus on improving data transmission and processing speed, enhancing the accuracy, reliability, and robustness of the system, and exploring innovative learning mechanisms.

The objective of this project is to devise a solution that leverages the most effective elements from currently established architectures while ensuring feasibility within the desired timeline.

3.2 Solution Aspects

In this section, the main aspects surrounding the solutions researched will be discussed. Many solutions have been developed, and although somewhat similar in their approaches to solving the

problem, they possess some differences in extra techniques and designs. The main differences between these approaches are based on the following:

- Hardware platform: Different articles may describe AER implementations that use different hardware designs, depending on the specific requirements of the system. For example, some implementations may use specialized circuits such as ASICs to improve the speed and accuracy of data processing, while others may rely on more adaptive hardware like FPGAs and FPAAs.

- Data transmission: The way in which data is transmitted in an AER-based system can also vary between articles. Some implementations of articles seen, use high-speed digital signalling techniques, such as LVDS or M-LVDS [6] [10] paired with SerDes, while others use more simple or low-power approaches like using $\log_2(N)$ wires to send the addresses between cores.

- Learning algorithms: The specific learning algorithm used in an AER-based system can also vary between articles. Some implementations may use STDP or other types of plasticity-based learning, while others may use reinforcement learning, unsupervised learning, or other types of algorithms.

- Network architecture: The specific network architecture used in an AER-based system can also vary between articles. Some implementations may use fully connected architectures, while others may have used more sparsely connected architectures. This makes a difference in the number of addresses used, and how they are encoded.

Both the learning algorithms and the network architecture are out of the main scope of this work, but will be considered when developing a solution. Having said that, the following aspects are important techniques that stand out in the research articles pertaining to this topic.

3.2.1 Hardware

From the conducted research, it has been found that there are more generalized solutions that do not require specific hardware and can be implemented on both low-cost FPGAs and, potentially, ASICs. However, among the implemented approaches, there are those that utilize FPGAs for adaptive hardware design [11] [12], others that make use of FPAAs [13], and additional approaches that leverage other adaptive hardware platforms [14][15][16].

FPGAs are digital circuits that can be programmed to perform a wide range of tasks, including implementing digital logic circuits, digital signal processing, and microcontrollers. FPGAs are widely used in a variety of applications, including in the field of neuromorphic computing. FPAAs are similar to FPGAs, but they are designed to perform analogue functions such as filtering, amplifying, and converting signals. FPAAs are often used in applications where it is necessary to process analogue signals, such as in sensors and instrumentation. FPGAs are often preferred

when flexibility, digital processing, and complex control logic are important factors in these neuromorphic computing applications. On the other hand, FPAAs are more suitable for applications that require analogue signal processing, energy efficiency, and mixed-signal integration. The choice between the two depends on the specific needs and constraints of the application at hand.

While FPGAs and FPAAs have their advantages, the use of ASICs (Application-Specific Integrated Circuits) deserves strong consideration for neuromorphic computing applications. ASICs are custom-designed chips optimized for specific tasks, resulting in unparalleled performance and computational efficiency. In applications where emulating brain-like functionality can be computationally demanding, ASICs' power efficiency is crucial for handling large-scale neural networks effectively. Moreover, ASICs offer low-latency performance, making them well-suited for real-time processing, which is essential in various fields such as robotics and autonomous vehicles. Their scalability, combined with the ability to tailor the design to meet the specific requirements of a neuromorphic computing system, allows for the implementation of large-scale neural networks in a single chip. While ASIC design and fabrication may involve higher initial costs, they become cost-effective when mass-produced, making ASICs a viable option for large-scale deployments.

3.2.2 Data transmission

When it comes to data transmission, as mentioned before, some approaches in articles simply transmit the addresses to the receiver using $\log_2(N)$ wires, where N represents the number of neurons fully connected to another N neurons, and others use more complex approaches such as using a SerDes (Serializer-Deserializer chip) paired with LVDS (Low Voltage Differential Signalling) to achieve using much fewer wires and keeping higher speeds.

3.2.2.1 SerDes

SerDes stands for Serializer/Deserializer and refers to a technology used for transmitting and receiving high-speed serial data streams over longer distances, typically between integrated circuits ICs or across communication interfaces.

The purpose of SerDes technology is to convert parallel data into a serial data stream at the transmitting end, and then convert it back to parallel data at the receiving end, as shown in Figure 3.1. This enables the transmission of a large amount of data over a smaller number of physical wires or channels, reducing the complexity and cost of the interconnects.

The serializer takes parallel data, typically in the form of multiple bits, and converts it into a high-speed serial data stream. This serialized data is transmitted through a communication channel, such as LVDS in many cases. At the receiving end, the deserializer receives the serial data stream and converts it back into parallel data, reconstructing the original data pattern.

SerDes technology can be utilized in conjunction with Address Event Representation (AER) for communication within and between neuromorphic chips. Within a single neuromorphic chip (intra-chip communication), SerDes technology can be employed to transmit AER-encoded events between different neurons or synapse circuits. The parallel outputs of neurons or synapses can be

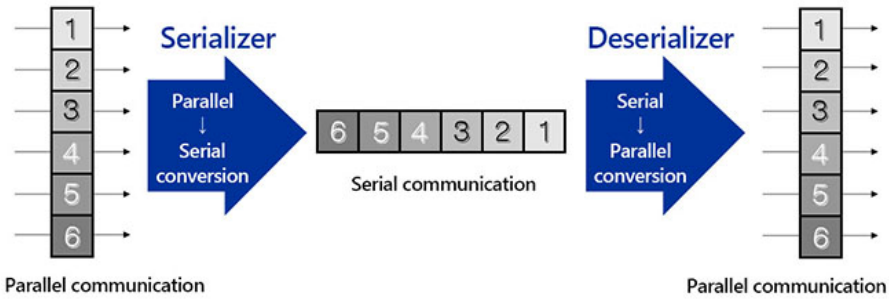


Figure 3.1: Basic SerDes operation

serialized into a high-speed serial data stream using a SerDes circuit, enabling efficient transmission of events across the chip. This serialization allows for a reduction in the number of interconnects required, leading to lower power consumption and improved overall chip performance.

3.2.2.2 LVDS

In LVDS (Low-Voltage Differential), the data is encoded as the voltage difference between two signal lines: the positive (P) and the negative (N) lines. The voltage on the positive line is higher than on the negative line for a logic high (1), while the voltage on the negative line is higher for a logic low (0). The voltage difference between the P and N lines represents the data being transmitted.

LVDS uses a low voltage swing, typically around 350 mV, which allows for low power consumption and minimizes electromagnetic interference (EMI). The differential signalling scheme also helps in reducing noise susceptibility, as noise affects both the P and N lines equally and the receiver only looks at the voltage difference between them. An example of an LVDS schematic is shown in Figure 3.2.

To ensure reliable data transmission, LVDS typically requires a balanced transmission line with controlled impedance. It is particularly well-suited to applications where low power consumption and high data rates are required, such as in portable and battery-powered devices. This is why LVDS requires a driver to encode the data into the appropriate voltage levels for transmission. The driver is responsible for converting the digital data signal into the complementary voltage signals on the positive (P) and negative (N) lines.

M-LVDS is an extension of LVDS that allows multiple devices to communicate with each other using a shared communication channel. This makes it suitable for use in multipoint communication systems.

In the context of neuromorphic computing, LVDS and M-LVDS are often used in the design of the Address Event Representation (AER) protocol [6][10] [17]. This use of LVDS and M-LVDS in the AER protocol allows neuromorphic devices to transmit and receive data at high

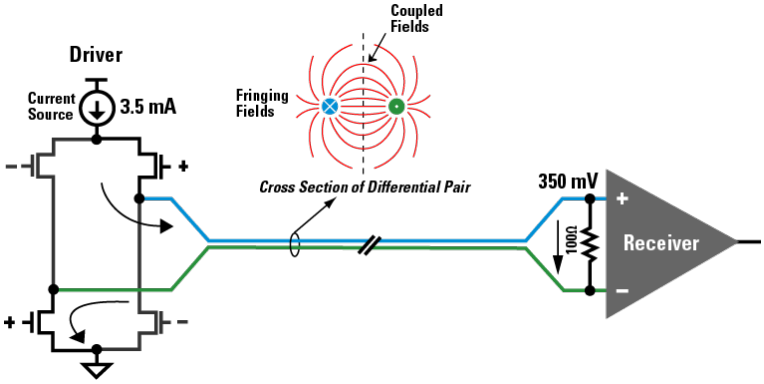


Figure 3.2: Schematic depicting basic LVDS circuit operation

speeds and long distances than when designed without it. It also offers a great resistance to the electromagnetic interference (EMI). All these advantages make it for interchip communication.

3.2.2.3 Manchester Coding

Manchester coding is a differential encoding scheme used for reliable data transmission in digital communication systems. It is characterized by its ability to synchronize the receiver's clock with the sender's clock, ensuring accurate decoding of the transmitted data. In Manchester coding, each bit of the original binary data is represented by a transition between two voltage levels. The two voltage levels, typically denoted as high and low, are assigned specific meanings. There are two conventions, but the most common one is such that a high-to-low transition represents a binary 0, while a low-to-high transition represents a binary 1, as a part of standard IEEE 802.3.

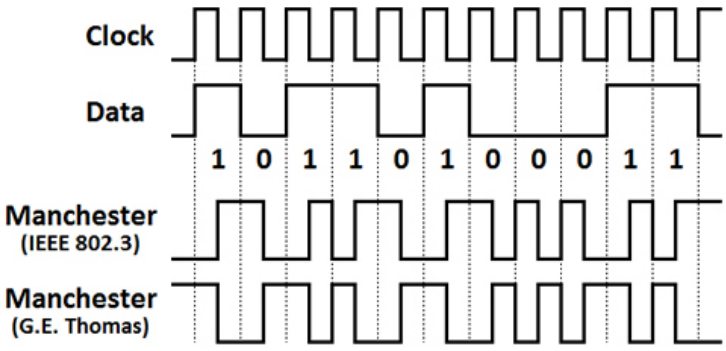


Figure 3.3: Manchester coding

The encoding process involves dividing each bit period into two equal time intervals. The first half of the bit period represents the value of the bit being transmitted, while the second half represents its complement. To transmit a binary 0, a high-to-low transition occurs in the middle of the bit period, while a low-to-high transition represents a binary 1, as seen in Figure 3.3. This differential encoding approach provides several advantages. Firstly, it ensures that there is at least one transition during each bit period, facilitating clock recovery at the receiver's end. This eliminates

the need for a separate clock signal for synchronization purposes. Secondly, the transitions carry the data, reducing the reliance on the absolute voltage levels, which makes Manchester coding more resilient to noise and channel distortion.

At the receiver's end, the Manchester-encoded signal is sampled at the midpoint of each bit period. By comparing the voltage level at the midpoint with the previous sample, the receiver can accurately decode the original binary data.

This is why Manchester coding often plays a crucial role in data transmission on AER (Address-Event Representation) solutions with serializers [6] due to its ability to provide clock synchronization and resistance to noise makes it an effective solution for ensuring reliable data transmission.

3.2.3 AER enhancements

In its original formulation, AER implements a one-to-one connection topology, which is appropriate for mimicking the optic and auditory neural systems [18] [11] [19]. However, there are implementations that make use of other elements to achieve better results and allow for more complex neural circuits.

Additional methods are required to create more complex neural circuits capable of convergent and divergent connectivity between cells. To facilitate these methods, Address-Event Representation (AER) enhancements have been developed, incorporating memory-based projective field mapping. These enhancements enable routing an address-event to multiple receiver locations by utilizing a look-up table (LUT). Each row of the LUT contains information about the sender's location, destination(s), and connection polarity, which dictates if it contributes to the connection and the connection strength. This enhanced architecture also makes use of an integrate-and-fire address-event transceiver (IFAT) and some other support circuitry [20] [21]. An example of a system with this support circuitry is shown in Figure 3.4.

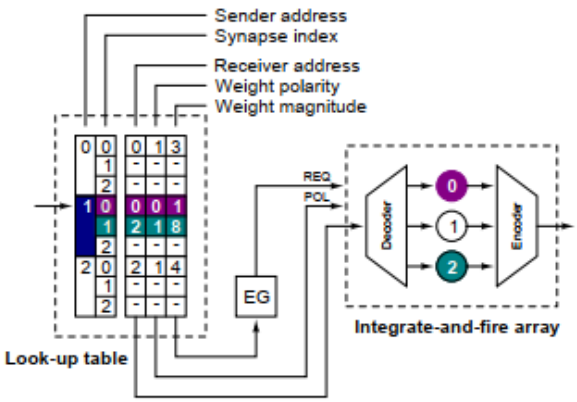


Figure 3.4: Schematic of the enhanced AER architecture [5]

This can also be helpful when this system is being integrated through a NoC architecture. The routing algorithms can be designed to leverage the information stored in the LUT of the enhanced

AER system, as shown in Figure 3.5. These routing algorithms determine the optimal path for address-events to traverse through the neural network cores, ensuring that the events reach the desired multiple receiver locations efficiently.

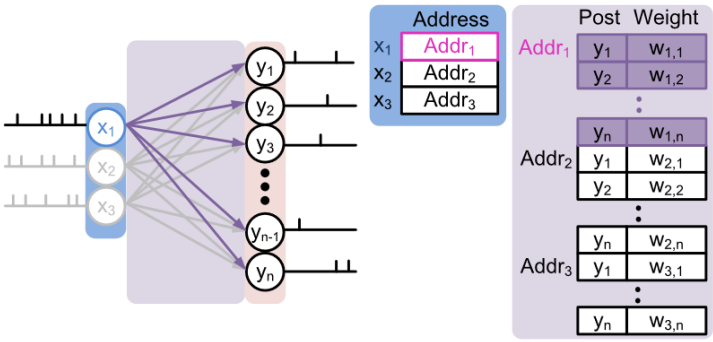


Figure 3.5: Address-based routing for spikes [5]

This enhanced system possesses other advantages, such as the capability of emulating leakage like in real neurons by regularly sending inhibitory events, which, in turn, by changing the frequency of these decay events, allows for the warp of the timescale.

3.2.4 Arbiters and Encoders

It is important to point out how encoders and arbiters play a vital role in AER systems by facilitating the placement of events on the shared data bus. To ensure effective competition with biological systems, these arbiters must exhibit high efficiency and operate at optimal speeds. It is crucial for these arbiters to be highly efficient and operate at high speeds without introducing any delays to the overall process. The integration of encoders and arbiters into a single module called "encoding-arbiters" can offer several advantages in AER systems. Combining these functionalities can lead to improved efficiency, reduced latency, and enhanced scalability. By integrating encoding and arbitration into a single module, the overall system complexity can be reduced, resulting in streamlined operations.

In terms of spike queuing, it is true that short-duration events have a minimal chance of overlapping, and even if they do occur simultaneously in multiple nodes, their arrangement can be flexibly adjusted to maintain the integrity of information. Similar to real neurons, where the firing rate is limited, the occurrence of synchronous events can be rearranged in a way that they happen in quick succession, ensuring minimal information loss. In these scenarios, arbiters play a crucial role by coordinating the scheduling and sequencing of these events, ensuring fair access to the shared resources and resolving any conflicts that may arise. Arbiters help manage the allocation of time slots or resources among the nodes, ensuring efficient and coordinated communication within the neuromorphic system [22].

Encoding arbiters often become performance bottlenecks, particularly in scaling up to handle complex and large-scale systems with lots of bits and possible addresses. Extensive research has been conducted to address these challenges, as evident in notable studies like [20], [23] and [24].

The objective of this research is to overcome the bottlenecks associated with encoding arbiters, enabling the development of more sophisticated and expansive AER systems.

3.3 Conclusion

Research has been conducted on state-of-the-art techniques, and the aim is to incorporate the most effective and optimal elements discovered and, with that in mind, design and implement a circuit capable of effectively implementing an Address-Event Representation (AER) protocol in a neuromorphic system.

To ensure flexibility in meeting project requirements, a custom hardware ASIC chip will be developed, providing greater freedom in design and adaptation. This aligns with the advantages of using ASICs in neuromorphic computing, as they offer purpose-built performance and power efficiency. Their scalability further allows for handling increasing computational demands without external components. Moreover, exploring the capabilities that an ASIC can enable opens up new possibilities for customization and innovation. With the ability to design specialized hardware for specific tasks, the project can explore architectures and approaches, pushing the boundaries of neuromorphic computing. As such, by using custom ASIC development, the project gains a configurable and efficient approach to meet its specific goals.

For the address-encoding arbiters, a simple fixed-priority approach will be adopted. The primary objective is to ensure efficient handling of a great number of addresses, all while keeping the delays of the operation to a minimum. In terms of data transmission, an initial approach based on multiple wires with minimal delay will be implemented. This basic transmission method will be iteratively improved with the ultimate goal of incorporating a SerDes that can be paired with an LVDS interface to allow for communication across great distances while still maintaining great speeds. This advanced setup will enable communication between the sender and receiver using only a few wires, while maintaining a high-speed connection capable of handling the spike rate effectively. Due to the complexity involved in enhancing the AER protocol and the challenges associated with complex routing and NoC, it is unlikely that these enhancements can be fully implemented within the desired timeframe of this project. Nevertheless, integrating and expanding upon these enhancements would be a valuable addition to the overall project.

Chapter 4

Development and Results

In this chapter, the development of the system will be explained, presenting and explaining the methodology and tools used and starting point before presenting each of the iterations and their weak and strong points, as well as the results obtained.

4.1 Methodology

Inspecting the problem at hand, the steps to create a solution should be the following, drawing upon the digital design methodology outlined in Subsection 2.8:

1. **Define Requirements:** Clearly define the requirements and functionalities of the AER protocol chip and the technology that will be used in the design. Determine what it should do, including its input/output behaviour, timing constraints, and performance specifications. This step is crucial to ensure a clear direction for the design process.

2. **Create Verilog Models:** Develop Verilog models that capture the behaviour and functionality of the AER protocol chip. Break down the design into modules and submodules as needed. Implement the necessary logic and algorithms to handle address encoding, data transmission, and any other required functionalities.

3. **Verification and refinement:** Create comprehensive testbenches to validate Verilog model correctness and functionality. Employ appropriate test stimuli for model simulation to ensure their expected performance. Address and resolve any issues discovered during testing. If the Verilog models pass the initial testing phase, proceed to an iterative refinement process. Follow the plan detailed in the conclusion section (Section 3.3). Identify and implement improvements or optimizations to the design. Continuously test, iterate, and refine the models until they meet the specified performance and functionality standards.

4. **Synthesize the Design:** Once the Verilog models have been refined and tested successfully, proceed to synthesize the design. Use the synthesis tool provided (Cadence Genus) to convert the behavioural description into a gate-level netlist. This step generates the digital logic implementation of the AER protocol chip.

5. **Layout and Physical Design:** Take the synthesized netlist and perform the layout and physical design of the chip using Cadence Innovus. This involves mapping the logic gates onto the physical cells of the target technology, considering factors such as timing, power, and area constraints. Create the necessary interconnections and routing structures.

6. **Verify Again:** After the layout is complete, perform testing to ensure that the physical design meets the functional requirements.

7. **Iterate and Make Changes:** If any issues or shortcomings are identified during the testing phase, make the necessary changes to the layout or design. Iterate as needed to address any problems and improve the chip performance.

4.2 Tools used

The main tools that will be used to design and ultimately create the final chip will be the following:

- **Xilinx Vivado** [25]: Xilinx Vivado is a versatile software suite specifically crafted for FPGA and PLD development. It provides engineers with an integrated development environment (IDE) and robust behavioural simulation capabilities. This tool will be utilized to create initial versions of Verilog models and conduct behavioural simulation tests on these models throughout the iterative development process. The reason for utilizing this tool was due to the experience with the IDE user interface.

- **Cadence Xcelium** [26]: Cadence Xcelium is a high-performance digital simulation tool used for verifying and validating digital designs at the RTL, gate-level and physical level stages of development. It offers fast simulation speeds, advanced debugging features, and support for mixed-signal simulations. This tool will be used to simulate the obtained models during the various stages and iterations of the development.

- **Cadence Genus** [27]: Cadence Genus is a synthesis tool offered by Cadence Design Systems, an industry-leading electronic design automation (EDA) company. Genus is used in digital IC design to convert RTL (Register Transfer Level) descriptions written in hardware description languages (such as Verilog or VHDL) into gate-level representations. It facilitates this logic synthesis by utilizing the technology standard cells provided, generating gate-level netlists for simulation and testing (with tools like Cadence Xcellium), and producing netlists that will later be used during the layout process. The tool reports will also provide valuable information for analysis and refinement of the synthesis results.

- **Cadence Innovus** [28]: Cadence Innovus is a physical implementation tool used for transforming gate-level netlists into optimized physical design layouts, considering timing, power, and area constraints. It performs tasks such as physical synthesis, placement, and clock tree synthesis. This will be used to handle the layout of the standard cells that will be used in

creating the final chip and create the final netlists that will be used for simulations at the physical level and ensure the final product has been developed correctly at the final stage.

4.3 Requirements

The requirements established for this project were intentionally designed to provide a certain degree of flexibility, allowing for adaptability and optimization in various aspects. One critical requirement was the minimum clock frequency of 100MHz, which was set to ensure that the system could handle the input spike rate efficiently. By operating at this frequency, the circuit could effectively process and transmit spikes in a timely manner, accommodating the real-time demands of the neuromorphic system and its learning algorithms.

In addition to the clock frequency, a critical requirement for the system was the ability to receive spikes with a maximum width of 40 nanoseconds as input and accurately reconstruct them as in the output lines.

Another of the established goals of this project was to design a system capable of effectively handling 128 I/O spike lines while ensuring reliable spike processing and keeping delays to a minimum.

It is important to emphasize that besides these requirements that were set, circuit optimization was a key goal in this work. The primary focus was to minimize both the circuit footprint and the delays encountered during spike arrival and transmission. By reducing the circuit size, valuable resources could be conserved, enhancing cost-effectiveness and enabling the integration of additional components or modules within the system. Furthermore, minimizing the delays between spike arrival and transmission was crucial to preserving the temporal accuracy of information flow in the neuromorphic system.

4.4 Starting point

During the first stage of the development, an initial design was formulated. This initial design primarily focused on implementing the fundamental functionalities of the AER protocol in a synchronous manner by utilizing a clock signal for coordination between the components and without focusing too much on timing and area optimizations. One of the goals of this initial approach was to be able to handle conflicts that could arise when multiple spikes were generated by employing a simple fixed priority arbiter to select a spike. The chosen spike would then be encoded into an address and forwarded to a module responsible for transmitting the address to the receiver. The module utilized REQ and ACK signals to ensure the correct transmission and reception of the address. The receiver module would decode the address and generate an identical spike, transmitting it through the corresponding line indicated by the received address. This behaviour is illustrated through the following Figure 4.1. Based on the requirements, it was established that the spikes would maintain a constant width of 40 nanoseconds, equivalent to 4 clock cycles when the clock is operating at a frequency of 100 MHz, and consequently, to successfully reconstruct these spikes

received, the decoder will have to send this to another module that would generate a spike based on the spike line that would be set to one. This spike generator would make use of a clock signal at the same frequency of 100MHz and generate the spikes based on that.

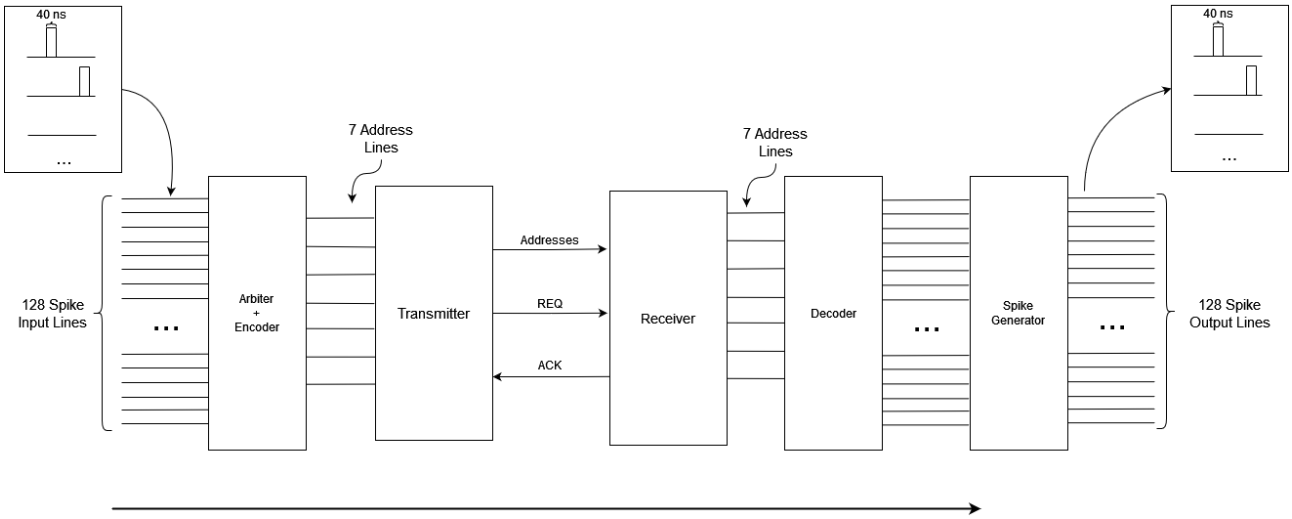


Figure 4.1: First sketch of the design of the AER system

From this analysis of the first sketch, a model can be created that seeks what was set from the initial sketch and integrates the clock signal to better understand the relations between the modules and how they will use this clock signal. Thus, a first design was developed as a behavioural Verilog model, which implemented these features in a simple way, intending to serve as an initial stepping stone, a starting point, intending to refine and expand upon it to develop an enhanced AER module.

The goal was to create a more advanced module that, through the use of more complex features, could significantly reduce delays, minimize the area occupied, and optimize the utilization of wires.

By analysing and studying this initial design, valuable insights could be gained regarding the different components and their interactions within the AER RX and TX modules. This understanding allowed for a logical division of these components into distinct modules, each serving a specific purpose and contributing to the overall functionality of the system. The modular approach facilitated better organization and scalability for future improvements of each module. This first design is represented in the following Figure 4.2.

This design was also done dynamically, allowing for the N number of input spike lines to be introduced as a parameter on the top module, and its submodules would be adapted to that number. This flexibility is an advantage since it easily allows studying the scalability of the system by increasing the number of needed lines.

Relatively to the implementation of this model, this initial model consisted of a simple fixed priority arbiter and an encoder to convert the selected signal from the arbiter to an address. This address is then put on a simple latch element ("out") that could be reset and that would output the addresses through $\log_2(N)$ wires to the RX module. The behaviour of this latch is controlled by

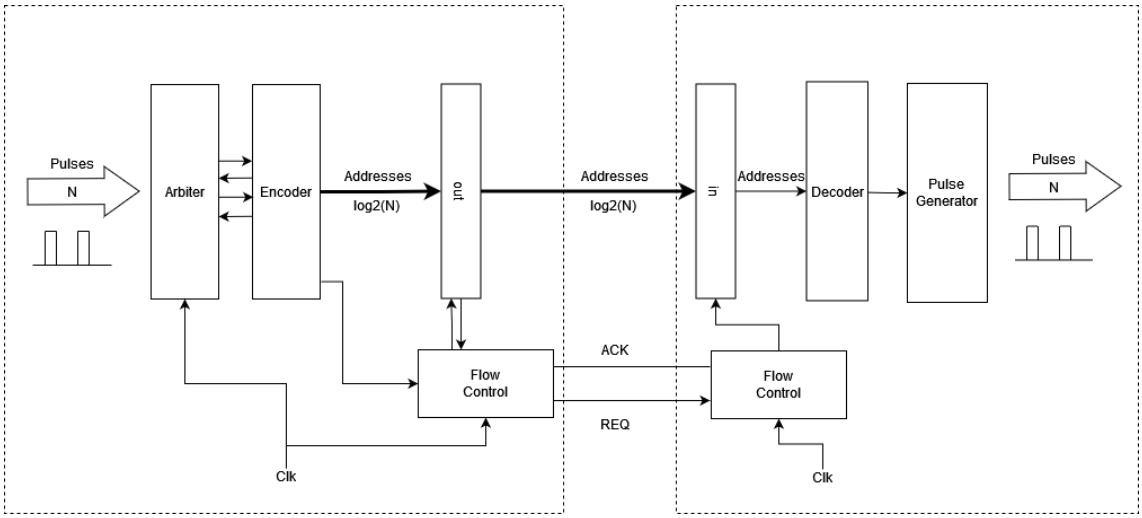


Figure 4.2: Schematic of a very preliminary design model of the AER system

another module that acts as flow control, using the handshaking signals REQ and ACK to ensure the TX and RX do not have conflicts and respect their respective timing constraints. Both the Encoder and Flow Control modules would use TX clock. On the RX side, the same latch and flow control module principles would be used to ensure the address was received correctly. And then, from the latch, the decoder would decode the address and generate the spike in the correct output spike line. The clock signals in both the RX and TX do not need to be synchronized and can be completely different, as the handshaking protocol ensures the data is received correctly between the two independent modules.

This works as a "pseudo-asynchronous" system since it makes use of the asynchronous handshaking protocol through the use of REQ and ACK, but is limited by the clock to iterate through the flow in most elements. This approach might seem strange at first, but the reason for using an asynchronous protocol in a synchronous system was to later remove the clock and create an asynchronous system paired with a synchronous transmission system. Also, due to the fact that at this time, the clock speed is to be much faster than the spike generation and synaptic decrement in plasticity rate, the delay is introduced by inserting a clock in the system.

This initial model would later change through the iterative refinement process that is aimed to enhance the AER module capabilities, but the principles and flow would remain the same.

After the first sketch of the design, some iterations were developed, each with a specific goal and advantage in mind. These iterations were not very similar to the initial sketch, as they sought to simplify and reduce the modules to their simplest possible, as to try and not be too heavy on area and on delays. The most notable changes were merges into a system composed of fewer modules.

4.5 First AER Iteration: Foundational Approach

4.5.1 Architecture

The initial iteration was aligned with the original draft since it maintained most of the transmission principles from the original sketch and was used for $\log_2(N)$ lines transmission of addresses between the TX and RX modules. This, in one way, constitutes a great advantage as it is the simplest way to send the address and offers little to no delay due to the quick processing of the simple control sender module.

However, it has many address lines used, which constitutes a disadvantage. The abundance of wires in integrated circuits (ICs) poses several drawbacks that significantly affect overall chip performance. The extensive wiring leads to signal delays, impeding the speed and efficiency of data transmission within the IC, which can hinder the chip functionality and responsiveness. Furthermore, an excess of wires results in a slight increase in power consumption attributed to resistive losses. Additionally, the excessive wiring occupies valuable space on the chip, limiting the room for integrating additional components and functionalities, thereby restricting the chip overall capabilities.

The biggest differences from the initial model are how the flow control and transmission are implemented. In this system, a single module called **Control Sender** is used to do both the flow control, handshaking, and the addresses' transmission.

In order to code 128 possible spike positions with 7 bits, the first line address is to be coded as the binary code 0000000, this causes a problem since there is no way to distinguish it from when no spike is being transmitted as the address lines are also set at 0000000 during inactivity. To solve this problem, an enable signal is created by the priority encoder to signal the control sender module that an address is being transmitted, as seen in Figure 4.3. This works the same way as the REQ signal to signal an address is being sent to the RX module.

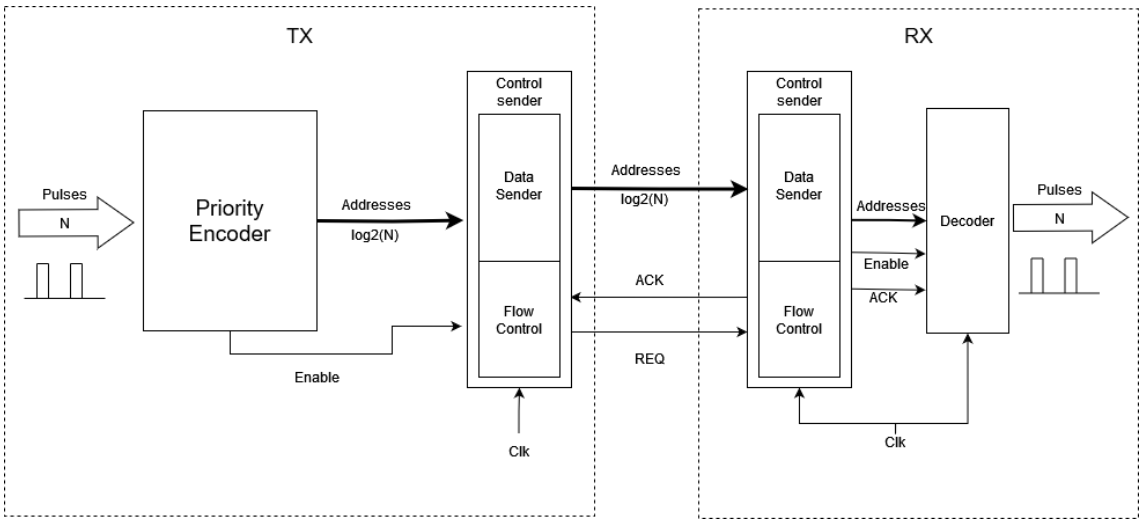


Figure 4.3: Schematic of the 1st iteration of the AER system

In this iteration, the **Priority Encoder** waits until a spike is received, encodes it into the corresponding address and generates the signal *enable*. If two or more spikes appear at the same instant, it will decide which one is to be encoded through its fixed priority. It also has no buffer for spikes that might occur in quick succession, which means that if two spikes come one after another, it will lose the second as long as the first is still being sent by the sender. On the one hand, this allows for the system to be quicker and smaller, but on the other hand, if the rate of neuron activation is high enough compared to the clock and transmission speed between TX and RX, it might lose a lot of information, and a buffer such as simple FIFO might be worth implementing.

This **Control Sender** module makes use of the ENABLE and ACK signals to evaluate when it can enable the output latch and change the input. It also checks if the input has changed and does not transmit the new address until the transmission of the previous address is over. This module was done with a minimum number of registers in order to minimize the number of clock cycles needed to transmit the address, due to the importance of timing preservation for STPD learning.

The **Control Receiver** receives the REQ signal to know that the address is currently being sent through the address lines. It then stores it and responds with the ACK signal to the transmitter, and the transmission is finished. It then sends the stored address to the decoder.

The **Decoder** was implemented as a simple state machine, seen in Figure 4.4, where the ACK signal starts. When it receives it, the address is decoded to know which spike output should be used. Subsequently, through the use of a counter, it sets the decoded spike line to HIGH for the number of cycles decided by the parameter *NUM_CYCLES_PER_SPIKE* (in this case 4 cycles, due to the 100 MHz clock frequency). After completing the designated task, the module initiates a reset process and enters a waiting state, anticipating the reception of the ACK signal. Upon receiving the ACK signal, the module proceeds to restart the FSM, resuming its operation or transitioning to the next state in the process.

This AER iteration also has the advantage of being reliable and robust to clock jitter. This is due to its pseudo-asynchronous properties, which come from the usage of REQ and ACK that ensure that the signal is always received and makes the transmission independent of their clock relation.

4.5.2 Behavioural Simulation Results

As previously stated, the environment used to develop the initial Verilog models and conduct the behavioural simulations was the Xilinx Vivado. The following waveform captures are from it.

Some extensive testing was done on the system. A testbench was developed for each module to ensure they were operating correctly on a local level within the system. But to ensure the system worked correctly, a bigger top module testbench was developed, presented in the following results and waveforms.

In this simulation, the system was tested for several random input spikes and checked to see if it would be able to output the correct one according to the fixed priority of the encoder. It was also tested with two different clock signals at the TX and RX modules, with different phase differences,

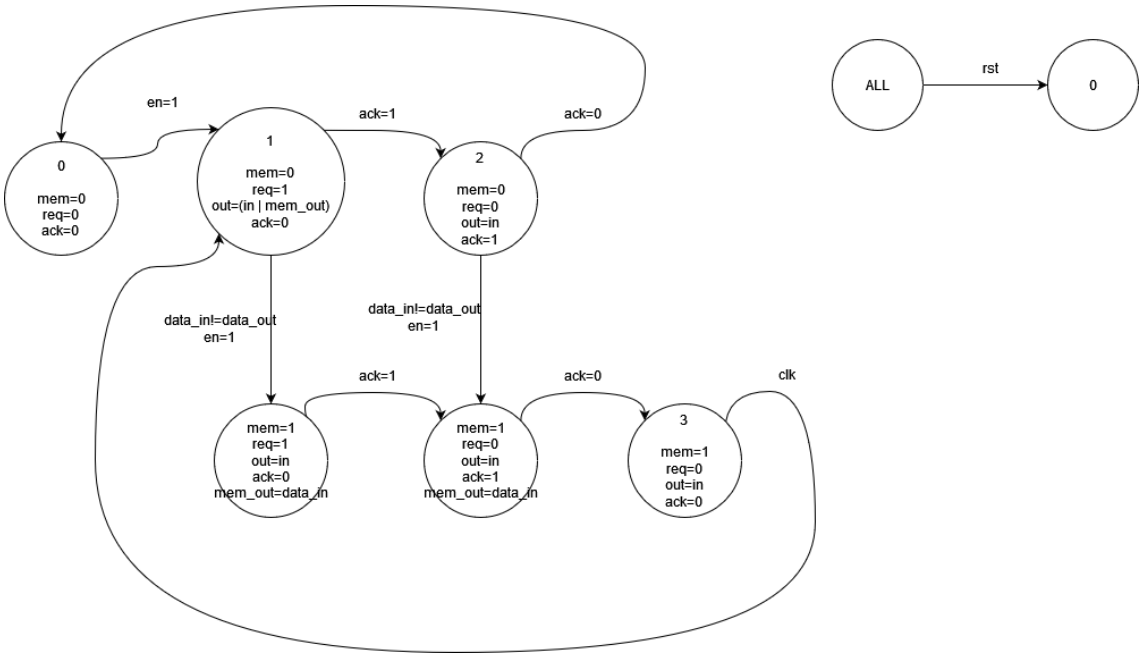


Figure 4.4: FSM of Decoder

to understand better if the system is robust enough to handle this difference through the use of the REQ and ACK signals.

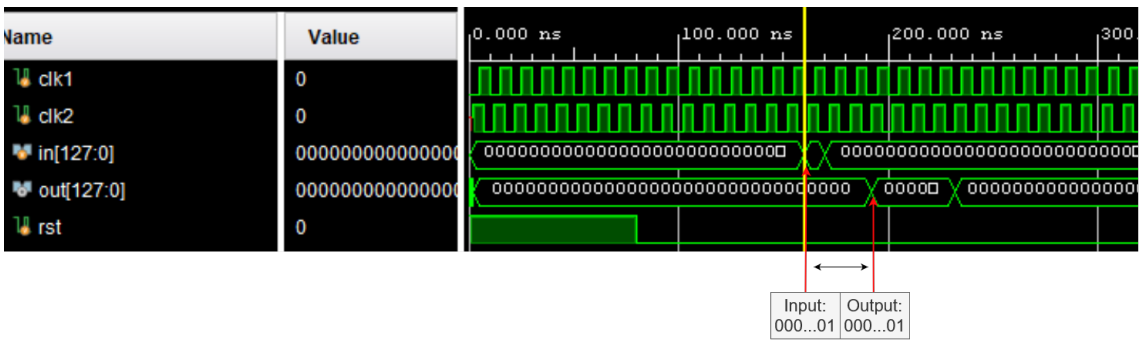


Figure 4.5: Waveform of AER iteration 1 - Delay from spike input and output

Based on the simulation results, shown in Figure 4.5, it is evident that this system exhibits the worst approximate delay of 4 clock cycles or 40ns, starting from the moment the spike arrives until the transmission initiates it as an output.

The testing also covered the possibility of multiple equal spikes being sent in quick succession, and also the possibility of having spikes arriving at a time that was not exactly at the positive edge of a clock signal. Both these possibilities do not constitute a problem due to the robustness of the system. As long as the spike received is longer than a clock cycle, which it should be (spikes are expected to have a constant width of 40ns and the cycles 10ns at 100MHz), the system only needs a clock transition to detect and store a spike and later transmit it, as seen in Figure 4.6, where the circle 1 highlights the input spike being one cycle long, and the circle 2 highlights the behaviour

of REQ and ACK and enable signals.

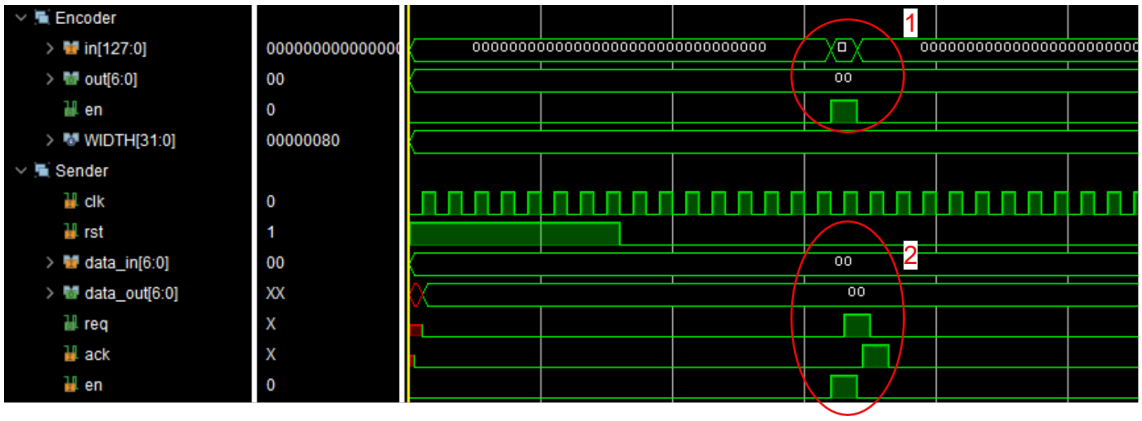


Figure 4.6: Waveform of AER iteration 1 - Management of small spikes

If two spikes arrive, one right after the other, the system is robust enough to handle and transmit both, without losing information, but will create a clock period of inactivity between them on the output, as shown in Figure 4.7 by the coloured input and their corresponding output spikes.

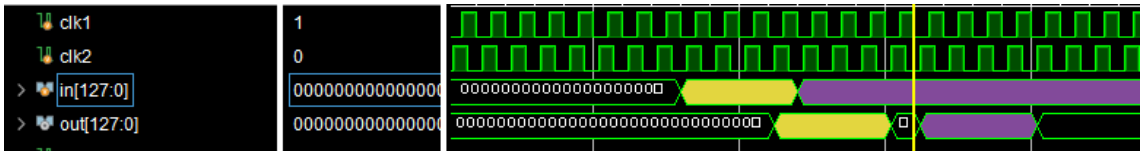


Figure 4.7: Waveform of AER iteration 1 - Demonstration of how the system handles two consecutive spikes

The clock period of inactivity is created due to the execution of a reset on the counter that generates the spikes after a spike has been generated. The described scenario is depicted in Figure 4.8 within the "Decoder" module, particularly in its counter register. In the initial transmission highlighted by the first white circle, only one address is received, evident from the number of ACKs and REQs received also. In the second white circle, two addresses arrive consecutively, and it becomes apparent that the counter signal needs to reset to zero before generating another spike, highlighted by the red circle. The figure also provides insights into the different states and their corresponding actions.

4.5.3 Synthesis and Report Analysis

The synthesis process was efficiently executed using the Genus tool, which transformed the Verilog RTL code into a gate-level netlist. This netlist was based on the 130 nm technology standard cells provided for the design. Throughout this process, various constraints were carefully taken into account, including the specific technology requirements and user-defined constraints.

To ensure the robustness and accurate performance of the design, essential constraints were specified, such as clock jitter and skew, input and output delays, and input and output load capacitance. By incorporating these constraints, the Genus tool was able to optimize the design to meet

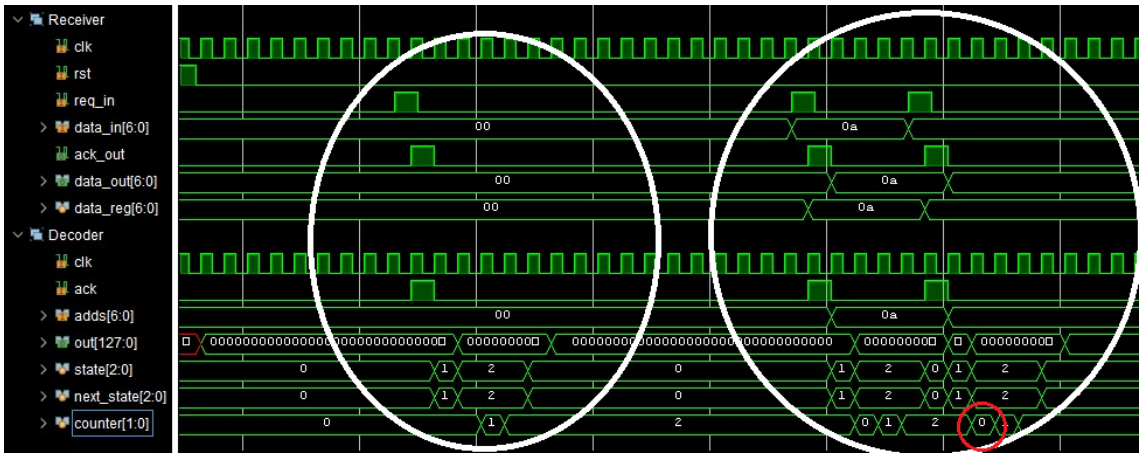


Figure 4.8: Waveform of AER iteration 1 - RX modules signals

the desired timing characteristics, enabling better synchronization and helping to achieve timing closure. High-level schematics of the products of this synthesis for both the TX and RX modules are depicted in the following Figure 4.9

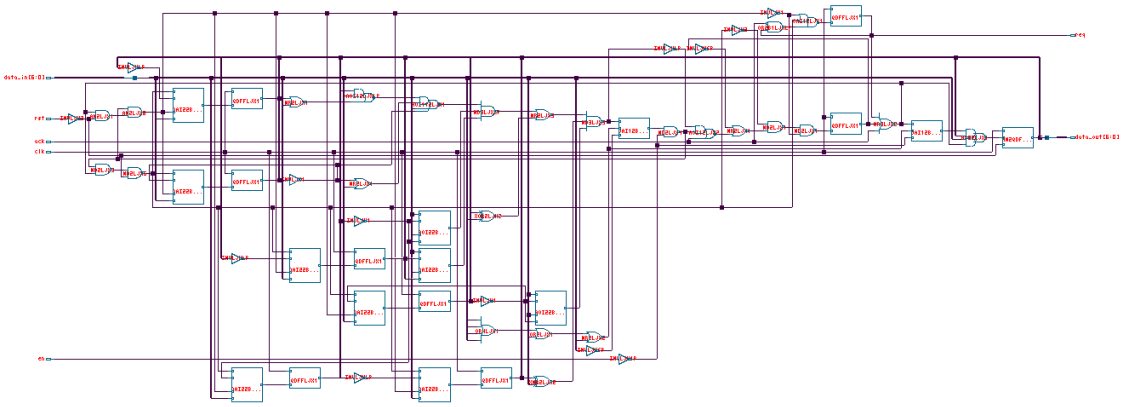


Figure 4.9: Schematic of the TX module post-synthesis with the cells and their connections

It also served as a comprehensive design analysis tool, inspecting the design for potential structure issues or challenges, which allowed for the integrity and reliability of the resulting netlists to be ensured. Another asset offered by the Genus tool was its ability to generate detailed reports of various aspects of the design. These reports were helpful during the optimization phase, offering information related to timing, area utilization, and other key design metrics.

After going through the cyclic process of optimization and redesigning the initial model, the reports for both the TX and RX modules indicated that there were no structural issues. The resulting netlists complied with timing rules and area requirements, validating the final system design.

From the report analysis, it is also important to note that the reports also confirmed that the system would operate without any problems at the intended clock speed of 100MHz. Also, there was no negative slack, suggesting that the design could potentially handle even higher clock speeds

without performance issues. However, it is essential to recognize that the critical path in the developed systems was identified as the clock fanout, which might benefit from improvement through Clock Tree Synthesis (CTS) techniques, but since for this purpose, the designed chip is relatively simple and does not need to operate at higher speeds, no CTS was conducted.

Another noteworthy application of the Genus tool was to determine the maximum achievable frequency before encountering a negative timing slack in any of the TX or RX modules. The results revealed that both the RX and TX modules could operate at frequencies up to 340 MHz without causing any malfunctioning due to negative timing slack, with the TX module being the limiting one. This remarkable finding exceeded the desired frequency by more than threefold. Leveraging this higher frequency could prove advantageous as it enables higher transmission speeds, addressing the challenge of potential spike loss during data transmission. The increased frequency allows for faster data transmission, which, in turn, enhances the efficiency and reliability of spike transmission in the system. Additionally, operating at a higher frequency can provide enhanced temporal resolution, which is important in the context of STDP learning algorithms. STDP relies on precise timing relationships between neural spikes, and the improved temporal resolution enabled by the higher frequency can facilitate more accurate and effective STDP-based learning processes.

Additionally, during the synthesis process, a comprehensive Multi-Mode Multi-Corner (MMMC) analysis was employed to ensure the chip design robustness and reliability under a wide range of operating conditions and process variations. These corner variations arise from variations in Process, Voltage, and Temperature (PVT). In the MMMC analysis, three distinct corners were considered: Slow, Nominal, and Fast. In the Nominal corner, the voltage used was set to 1.2V, and the temperature was maintained at 25ºC, representing typical operating conditions. The Nominal Corner provides an essential benchmark to assess the chip performance under standard scenarios. In contrast, the Slow Corner introduced more challenging conditions. The voltage used was reduced to 0.9V, and the temperature was elevated to 125ºC, simulating conditions that push the chip to its limits. Evaluating the chip functionality under the Slow corner allowed us to understand how it would perform in less favourable scenarios.

Despite the corner conditions in PVT, this meticulous design and synthesis process ensured that the fundamental chip layout and architecture remained consistent throughout all the corners. This stability across process variables is a testament to the robustness of the design approach and its ability to withstand variations in manufacturing conditions.

After completing the synthesis iterations and refining the netlists, the subsequent step involved creating a physical-level model of the final system through layout design. The netlists obtained from the synthesis steps served as the blueprint for placing and routing the system components on the chip.

4.5.4 Layout and Simulation Results

When it comes to the layout process, all design layouts created were done in a consistent and systematic manner. Each layout followed a standardized set of steps, and as a result, scripts were

developed and used to streamline the overall process of creating and optimizing the final designs. These scripts are available in the annex.

The approach used for floorplanning involved creating an initial floorplan with a square site core positioned at the centre of the layout and leaving 10 μm spacing around all sides, reserved for power planning. This initial floorplan not only defined the specific type of site core but also determined the percentage of core utilization. Subsequently, power rings were constructed around the design, maintaining the specified spacing around the core, to ease the distribution of VCC and GND signals. This technique is commonly used in integrated circuit layout design, where a ring-shaped structure is created around the perimeter of the layout.

The ring surrounding the design includes both VCC (power supply) and GND (ground) rings. The ring around design with VCC and GND proves to be an effective technique for providing a stable and uniform power distribution, reducing noise, and enhancing signal integrity within an integrated circuit. It helps create a more robust and reliable design by providing a well-defined power and ground structure. This design can also be complemented through vias or metal traces created throughout the circuitry, called stripes. These connections ensure the power supply and ground are distributed uniformly throughout the IC. A layout schematic of this approach on the Innovus program can be seen in the following Figure 4.10.

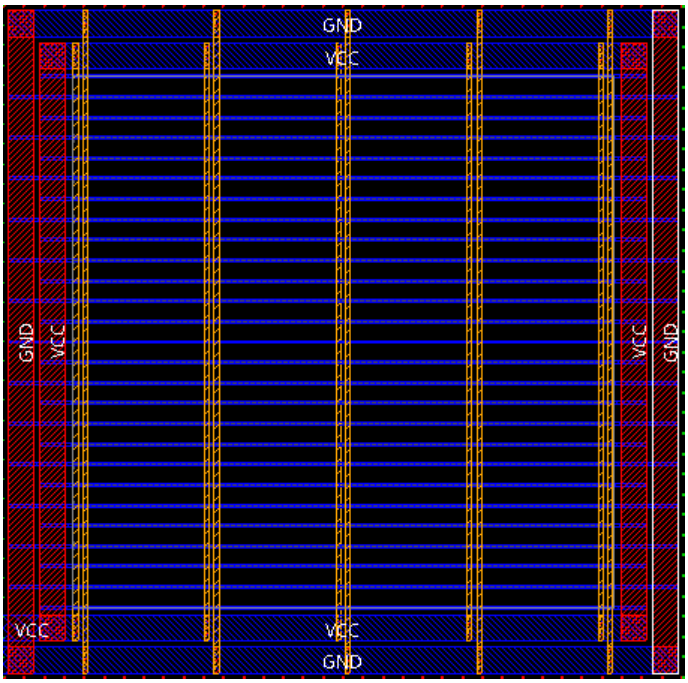


Figure 4.10: Layout schematic of the power planning

After establishing the power delivery framework for the chip, the next step involved distributing the pins for all signals. To accommodate the substantial size of 128 Lines of the IN array in the TX module and the OUT array in the RX module, these pins were spread along the chip Top, Right, and Bottom edges. By distributing them across these edges, the aim was to optimize the routing process and prevent congestion along any single edge.

In addition to the IN and OUT arrays, the remaining signals, including REQ, ACK, OUT (TX), and IN (RX), were distributed along the remaining Right edge. This deliberate distribution approach can be observed in Figures 4.11 and 4.12. Such a distribution scheme avoided clustering connections exclusively on one edge, consequently enhancing the overall routing feasibility of the layout process. Spreading the pins along multiple edges also achieved a balanced and streamlined layout, promoting smoother signal flow and reducing potential routing challenges. This pin distribution strategy significantly contributed to the overall optimization and performance of the chip design and allowed for a quicker layout routing time. This distribution can be seen in Figure 4.11.

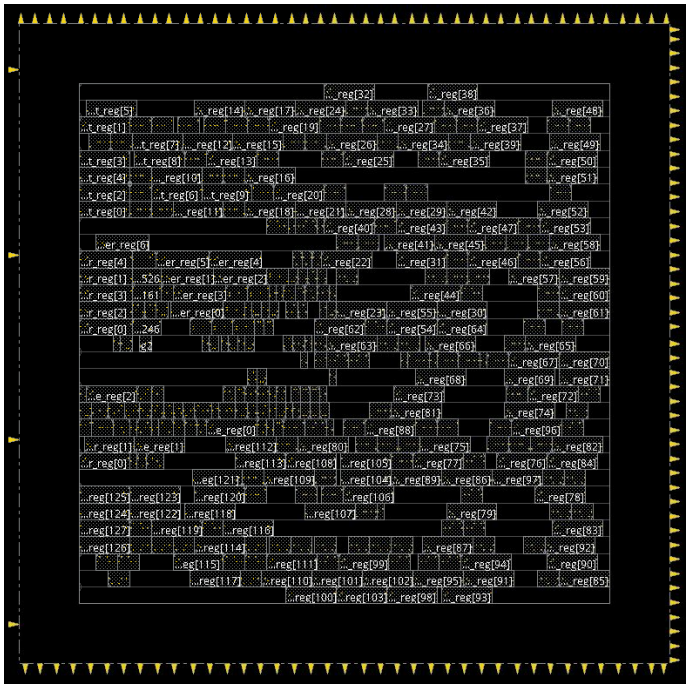


Figure 4.11: Innovus snippet of layout with distributed pins and standard cells placed in lines

Some pins were added to the chip in the bottom right corner to enable the VCC and GND power framework. A close-up snippet of this can be seen in Figure 4.12.

After defining the pin locations on the chip, the subsequent step involved placing the standard cells using the automated 'placeDesign' command in Innovus. This command leverages Innovus' advanced placement algorithms and optimization techniques to position the cells within the chip floorplan effectively. The placement process takes into account various factors, including signal timing, power consumption, area utilization, and design constraints, ensuring a well-balanced and optimized solution.

The 'placeDesign' command analyses the design requirements and objectives, thereby minimizing the need for manual intervention in the placement process. While users have some flexibility to influence the placement of a module cells on the core, it was unnecessary to intervene manually in this particular case. The automated placement algorithms successfully determined the optimal positions for the technology cells. This step also creates some preliminary routing

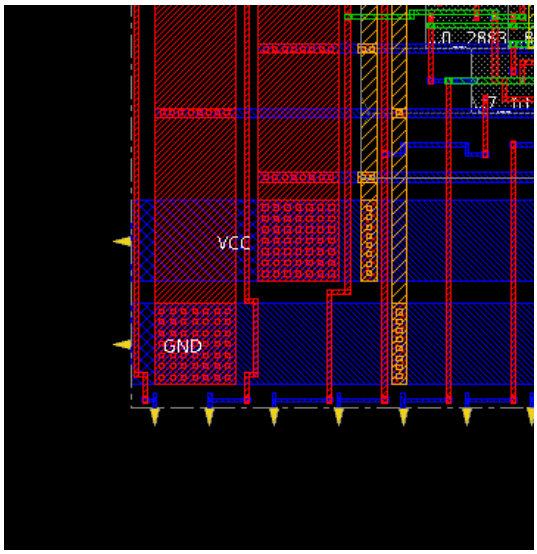


Figure 4.12: Innovus snippet of layout with pins as pads for VCC and GND input

between cells in order to have an idea of how the placement will affect routing. The standard cells placement on the chip core is represented in the following Figure 4.11.

Following the placement of the technology cells, the subsequent step involved routing connections between them, ensuring their proper connectivity to the power infrastructure. This process involved establishing the necessary interconnections while adhering to design rules and constraints. By carefully routing the connections, the design achieves efficient signal propagation and reliable power distribution throughout the chip. There are some options that dictate how this routing is done, but in this case, a timing-focused approach was selected, in order to preserve the timings as much as possible. A snippet of the layout post routing is represented in Figure 4.13.

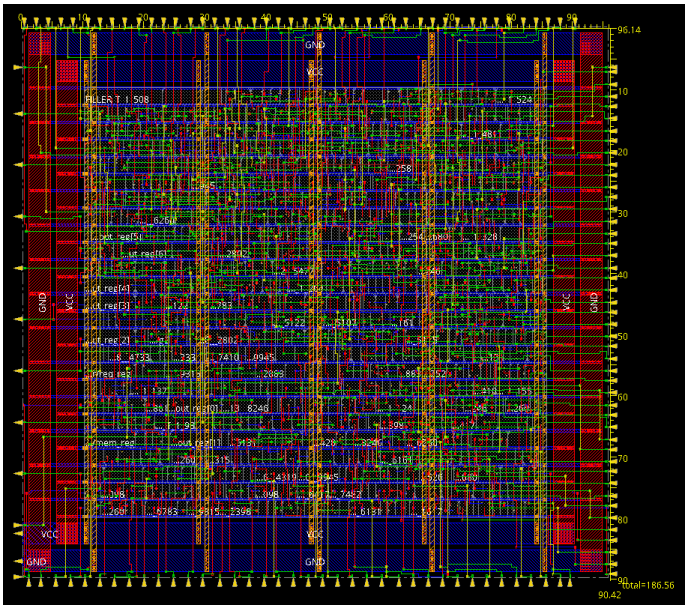


Figure 4.13: Innovus snippet of layout displaying all metal layers post routing

The next crucial step in the chip design process is to add filler cells to fill the empty spaces present in the core. These filler cells serve an essential purpose in improving chip density and manufacturability. They are strategically placed between the technology cells to optimize the layout. It also improves manufacturability by providing uniform spacing between technology cells, resulting in smoother fabrication processes, reduced lithographic distortions, higher chip yield, and overall better quality. An example of a chip after the filler cell placement has been done is in Figure 4.14. An important thing to note is the dimensions of the final chips, which for the TX was $96 * 90 \mu m$, and RX was around $108 * 104 \mu m$.

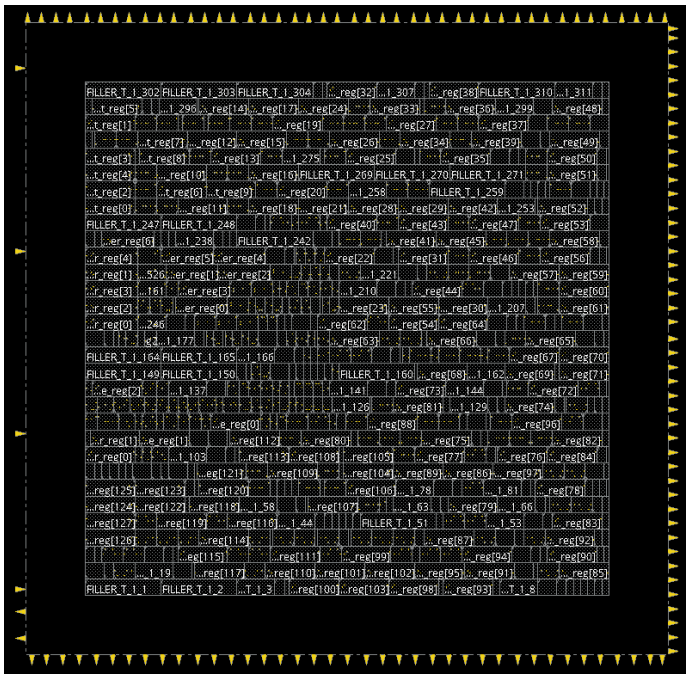


Figure 4.14: Innovus snippet of layout post filler cells placement

As mentioned before, Xcelium allows for physical-aware simulations, allowing you to consider the physical characteristics and effects of the design during simulation. It does this through the use of Standard Delay Format (SDF) files, which describe the delays that exist within the modules and their interconnects. This tool and setup were used to validate the final design of each iteration. These final simulations were done using the previously used testbenches to compare if the behaviour of the system was not heavily affected by the physical limitations. Since the layout of both the TX and the RX modules were done separately, the delay and noise on the transmission between the modules were not considered in the merged SDF files generated by the layout. As such, this delay limitation had to be added by creating delays between the OUT and IN cells of the TX and RX modules in the testbench. The delay chosen between these cells was 10 ns, equivalent to a clock of travel time between the two.

The results of these post-layout simulations showed that both the TX and RX modules were correctly synthesized, and were capable of encoding the spikes received and sending the addresses

while using the REQ and ACK signals. It was also possible to conclude that the delays of the physical constraints, do not have a great impact on the transmission process when compared to the ideal case presented in Section 4.5.2, which makes sense when considering the pseudo-asynchronous taken.

4.6 Second AER Iteration - Leveraging SerDes-Assisted Communication with CLK Line

This iteration was built upon the first, with the intention of solving its greatest issue, the number of wires used to connect both the TX and RX. It seeks to do this by using a SerDes, which would serialize the addresses and later deserialize them, effectively decreasing the number of wires used from $\log_2(N)$ to only one to send the data.

In the first part of the SerDes, the serializer is placed on the end of the TX as a PISO module and the deserializer at the beginning of the RX module as a SIPO module.

4.6.1 Architecture

Relatively to the last iteration, not much was changed in the **Priority Encoder** and **Decoder**, as they are not directly connected to the serialization, deserialization and flow control processes. The new module is shown in the following Figure 4.15:

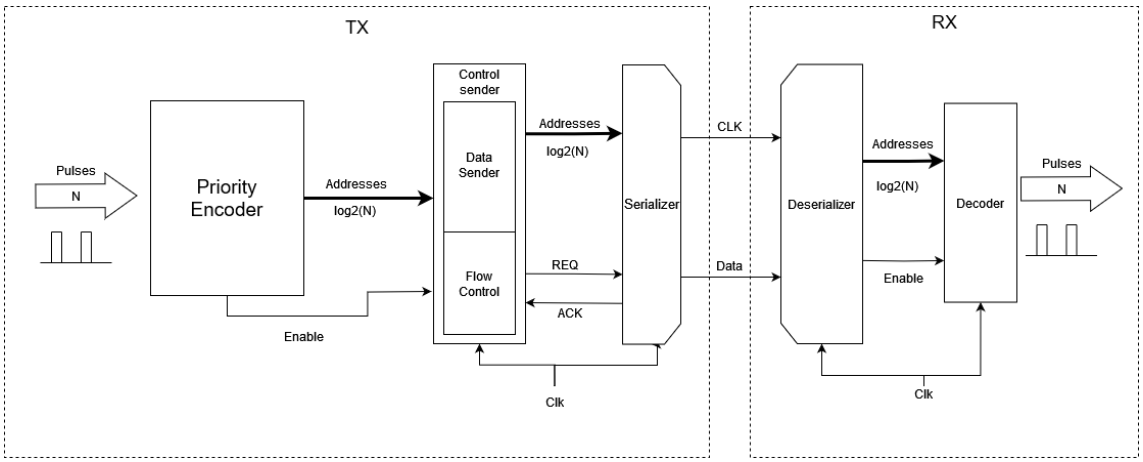


Figure 4.15: Schematic of the 2nd iteration of the AER system

As seen, the SerDes makes use of two signals CLK and the serialized DATA. The CLK is sent to simplify and ease the deserialization process, as then, no complex CDR techniques are needed, such as clock recovery and Locked-loops such as DLLs and PLLs, this can be described as "source-synchronous communication". In this approach, the TX sends the CLK that it used to serialize the addresses and transmits it in a separate line along the serialized data. The deserializer then uses this clock received to store the individual bits in a shift register to have the complete

address after the number of clocks corresponding to the width of the address. Both these modules are depicted in the following Figures 4.16 and 4.17.

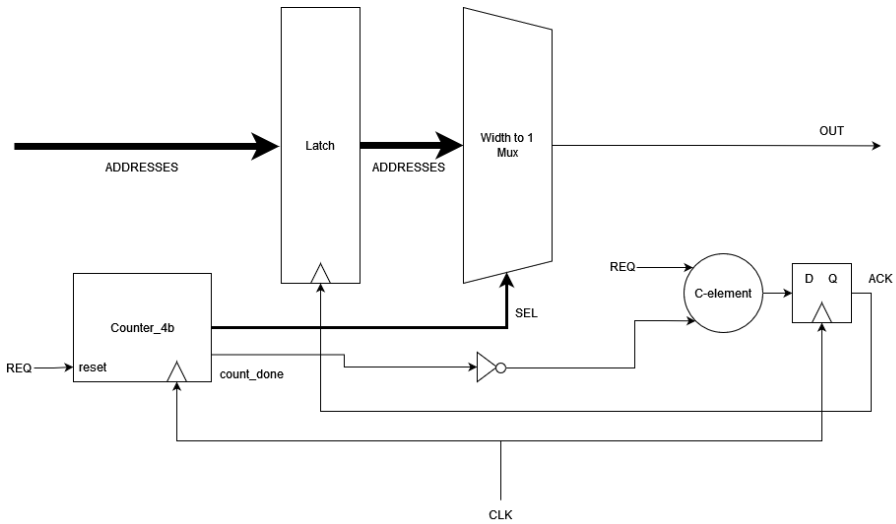


Figure 4.16: Address serializer circuit with compatibility for flow control

It is possible to see that this implementation of the serializer also makes use of the REQ and ACK signals for flow control and compatibility with the **Control Sender** module. This flow control is made possible through the use of a basic C-element and a counter-done flag, which were both obtained through a behavioural design through Verilog, instead of a custom approach. The basic idea behind using a C-element is to create a feedback path where the output of a gate or logic circuit is fed back into its own input. However, it is important to note that this behaviour design approach can be more prone to issues such as race conditions, glitches, and timing violations. Relatively to the counter-done flag, it serves to inform the rest of the system if already gone through the width of the address.

In this implementation, the ACK signal is initially set to 1 throughout the transmission of the address and remains in that state until the completion of the transmission. It effectively serves as an indicator of ongoing activity during the transmission process. The ACK signal is then set to 0 after the transmission has finished, marking the end of the activity.

Relatively to the deserializer, it consists of a shift register and a counter that ensures the output is only kept and sent after a certain number of clock cycles. This is to ensure all registers contain the correct value. This shift register consists of an array of FFs with a width the same as the address. This described system is depicted in Figure 4.17. After this value has been sent to another memory element for the posterior decoding, all the registers are cleared in order to ensure synchronization and to initialize this shift register array for the address transmission.

It is important to note, that this simple receiving system is not very robust to clock jitter, as it has no countermeasures to ensure clock synchronicity with the data. This was studied in the following subsection through the creation of a difference between the phases of the clock and serialized data. But to ensure that both CLK and DATA signals would be sent at the same time,

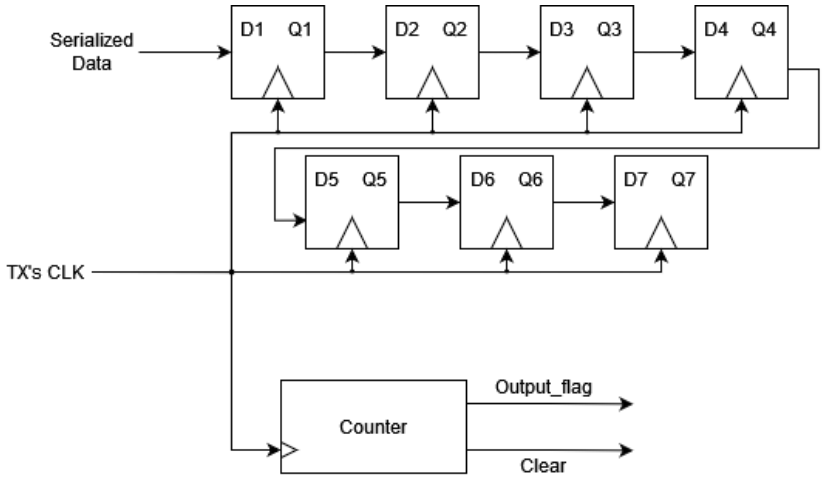


Figure 4.17: Deserializer circuit

a register would be used just before sending the clock. By passing the clock through a register, the clock edge is delayed by one cycle. In this "register-based clock synchronization", the register acts as a buffer, providing synchronization and reducing the possibility of metastability or timing violations for the receiving part.

4.6.2 Behavioural Simulation Results

The Behavioural testing was done similarly to the ones for the previous iteration, with the objective of understanding if the system worked correctly in ideal conditions and testing the robustness of the system in possible signal delays.

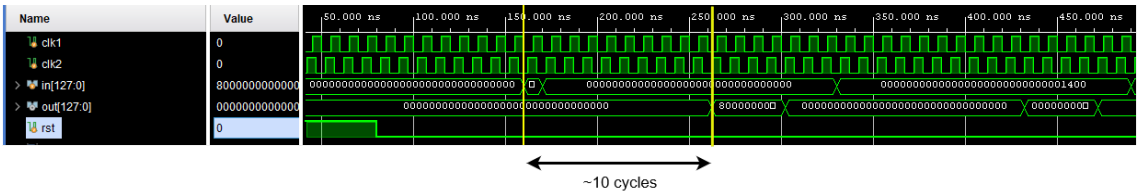


Figure 4.18: Waveform of AER iteration 2 - Total transmission delay

Based on the simulation results, it is evident that this system exhibits an approximate delay of 10 clock cycles (as seen in Figure 4.18), starting from the moment the spike arrives until its transmission initiates as an output. This is almost double the total transmission delay of the first iteration. Still, it is worth noting that this one needs at least 7 (the width of the address) clock cycles to serialize and transmit it, which leaves the same processing delay as the previous one.

From the behaviour simulation, it is also possible to see that all the signals are being generated correctly from the serializer, as depicted in Figure 4.19

In the obtained waveform snippet from the testing, it is apparent that there were two transmissions originating from two different addresses. It is worth noting that the transmission follows a specific bit order, moving from the Least Significant Bit (LSb) to the Most Significant Bit (MSb).

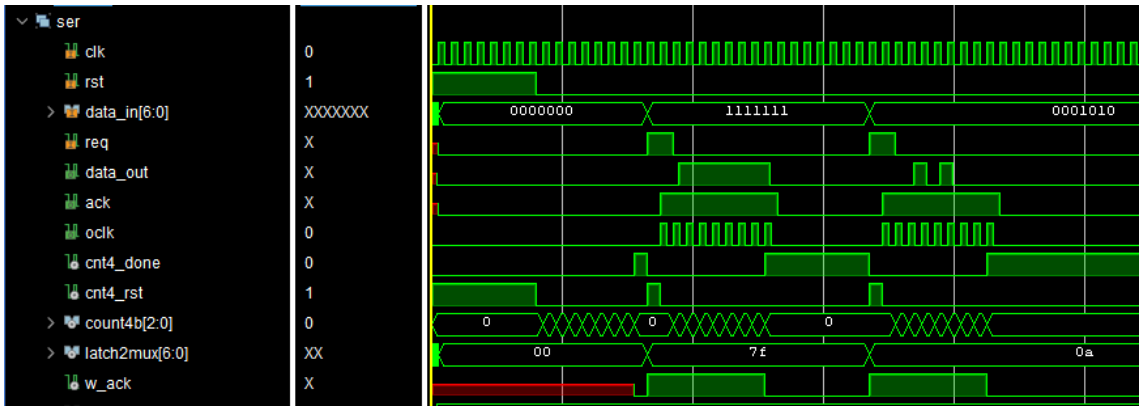


Figure 4.19: Waveform of AER iteration 2 - Serializer signal behaviour

This characteristic is also evident in how the second address is represented in the *data_in* signal as 0001010, which is subsequently encoded in the *data_out* signal as a sequence of bits: 0, 1, 0, 1, 0, 0, 0.

Relatively to the **Deserializer** module, the tests also showed that its behaviour is according to the design.

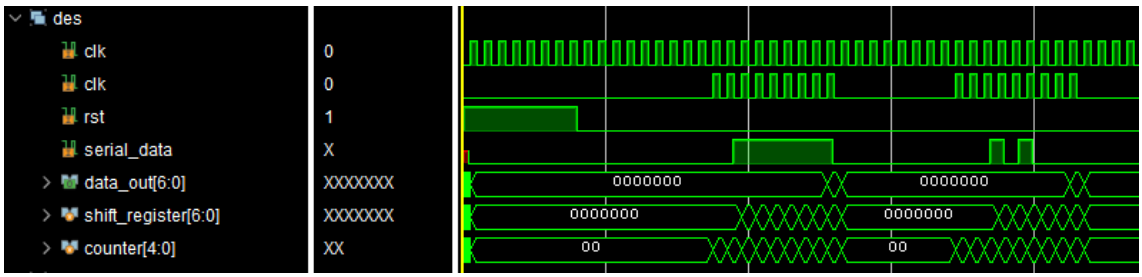


Figure 4.20: Waveform of AER iteration 2 - Deserializer signal behaviour

The serial data of the two signals is the same one sent by the TX in the last Figure. This deserialization is done through the *counter* register that keeps track of how many bits it reads, and when it is supposed to output the address stored in the bit *shift_register* to the **Decoder** module. This outputting moment can be identified in the waveform as when *data_out* changes for a cycle before being set to zero again, as dictated by the counter module.

4.6.3 Synthesis and Report Analysis

In the second iteration of the synthesis process using the Genus tool, gate-level netlists were generated for the TX and RX modules based on the same 130 nm technology standard cells. The various constraints given to design the system, including clock jitter, skew, input and output delays, and load capacitance, stayed the same as the previous iteration.

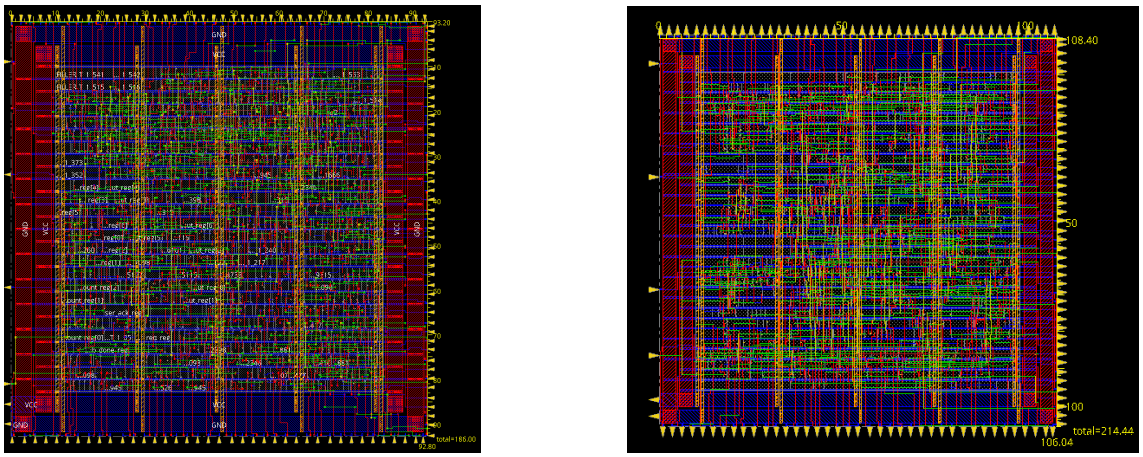
Repeating the same process as before, after the iterative optimization and redesigning, the reports for both the TX and RX modules indicated no structural issues, and the resulting netlists

complied with timing rules and area requirements, successfully validating the final system gate-level design. Notably, the reports confirmed that the system could operate without issues at the intended clock speed of 100 MHz, and there was no negative slack, suggesting potential for even higher clock speeds without performance problems, which was also researched. The results of this research were highly promising, revealing that both the RX and TX modules could operate at frequencies up to 410 MHz until negative timing slack was found.

The resulting netlists from these steps were then used in the posterior step of doing the layout and creating a physical-level model of the final system.

4.6.4 Layout and Simulation Results

For the layout, the same steps as the first iteration were followed. These processes resulted in the layouts presented in Figure 4.21. The TX chip has dimensions of approximately $92 * 93 \mu m$, while the RX is slightly bigger with $108 * 106 \mu m$. It is important to note that this area could be improved by using better technology.



(a) Layout of `tx_iter2`

(b) Layout of `rx_iter2`

Figure 4.21: Layouts of `tx_iter2` and `rx_iter2`

The post-layout simulations for both the TX and RX chips confirmed that both physical chips operated as anticipated when subjected to simulations accounting for actual physical delays, aligning with the behaviour simulated in the ideal scenario that did not consider delays as described by the Verilog model.

4.7 Third AER Iteration - Integration of Complex SerDes System

Building upon the previous iteration, this new system seeks to rectify its limitations. The primary shortcoming lies in the utilization of source-synchronous communication, which creates the need for utilizing an additional interconnect and also makes it more susceptible to the effects of clock jitter. To overcome these challenges, the improved system embraces a more complex SerDes

architecture, which ultimately seeks to mitigate these concerns by employing Manchester Coding (see Subsection 3.3) and Clock-Data Recovery techniques.

4.7.1 Architecture

Similarly to the last iteration, this development maintained most of the encoding modules and data control aspects, as seen in Figure 4.22.

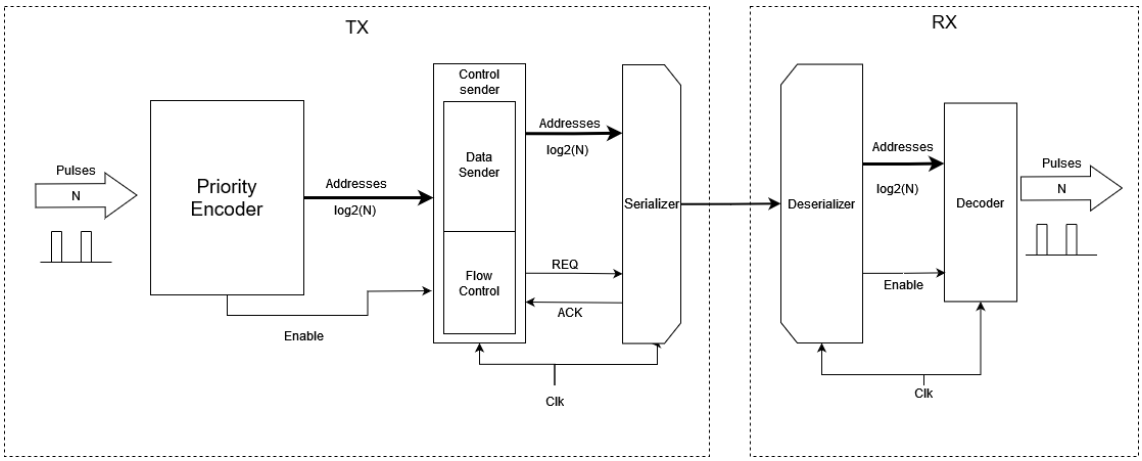


Figure 4.22: Schematic of the 3rd iteration of the AER system

Regarding serialization, the previous module had to be expanded to generate a preamble before sending each address. This preamble initiates and facilitates the clock recovery process before the data arrives. A module that encodes the serialized data into Manchester code was also developed. This module uses the serialized data, the ACK activity signal, and, evidently, the CLK signal.

The improved serializer is depicted in Figure 4.23.

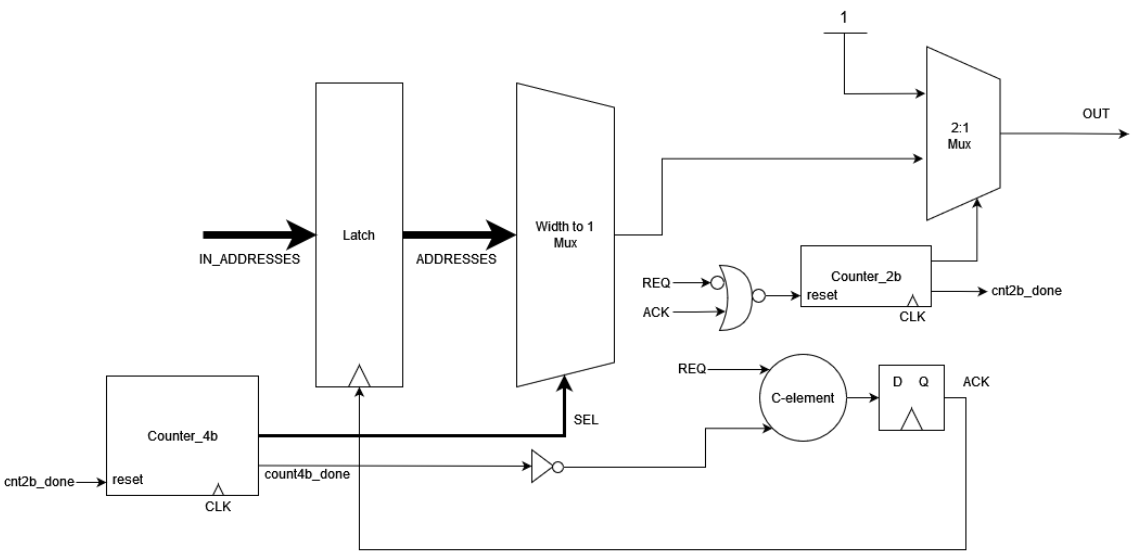


Figure 4.23: Improved serializer circuit with preamble generation

Within this module, key enhancements were introduced, namely the inclusion of a counter and a 2-to-1 multiplexer to allow for the generation of a 4-bit preamble. This preamble, comprised of consecutive 1s, needed adjustments to various reset flags and related elements. The number of bits used in the counter that creates this preamble can be configured through the module parameter *PREAMBLE_NBITS*. Once subjected to Manchester coding, this preamble assumes a clock-like form, which will later aid the deserializer in accurately recovering the original clock frequency. The Manchester coding was done through the circuit shown in Figure 4.24.

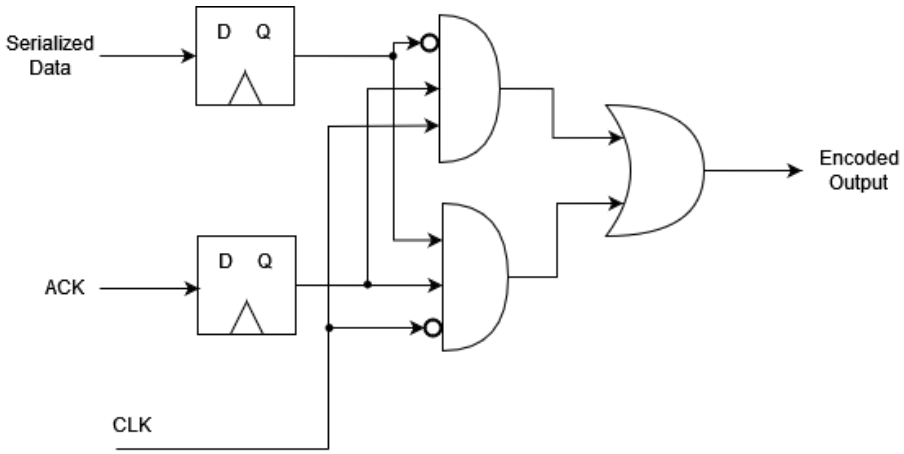


Figure 4.24: Manchester coder module schematic

Significant changes and advancements were made in the realm of the **Deserializer**. To facilitate an efficient Clock and Data Recovery (CDR) process, an analogue-based module was developed using Verilog-A. This module plays a pivotal role in restoring the clock signal by implementing a Delay-Locked Loop (DLL). The module generates a delayed version of the originally received clock by employing a delay line. The module determines the appropriate delay needed by comparing the original clock and the delayed copy. To achieve this, a charge pump is utilized, which actively adjusts the voltage applied to the delay line until the delay aligns with a period of the original clock. These efforts collectively contribute to the successful recovery and synchronization of the clock signal in the Deserializer module. This module was the same as the one used in the article [6], depicted in Figure 4.25.

4.7.2 Behavioural Simulation Results

4.7.2.1 Digital Part - TX

The TX module, designed in a fully digital manner, underwent thorough testing following the same process as the previous iterations. The primary objective of this testing was to validate the correct behaviour of the preamble and the Manchester coding functions, while also assessing their resilience to clock jitter.

The simulation results are presented in the following Figure 4.26

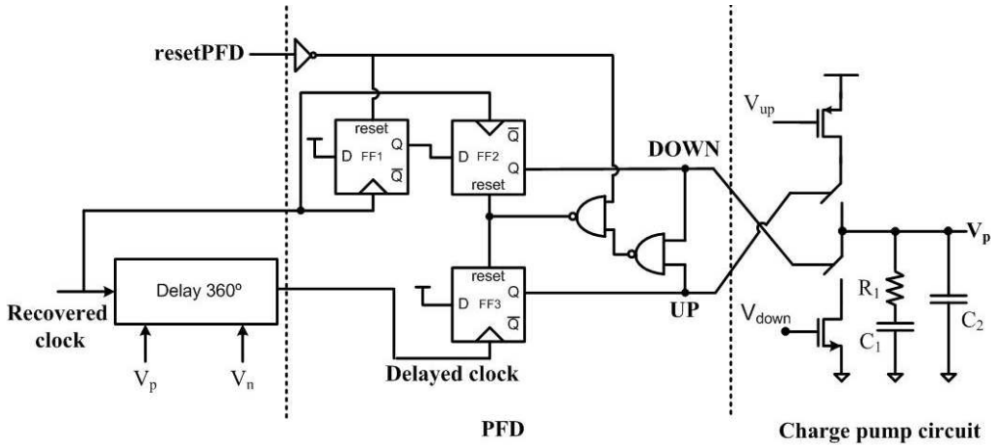


Figure 4.25: Delay-Locked Loop [6]

In the obtained waveform snippet from the testing, it is apparent from the signals highlighted in red that the spike signal ***in*** is converted to a serialized output, delayed by 3 clock signals.

Relatively to the serialization process, initially, the data begins as an address resulting from the previous encoding and is visually represented in purple, indicating its presence in the latch of the serializer. Subsequently, this address is serialized and combined with the 4-bit preamble, highlighted in blue.

Finally, the serialized data, comprising the address and the preamble, is passed through the Manchester encoder module. Utilizing the clock signal, the encoder transforms the blue signal into the red signal *out*, completing the serialization process. It is also possible to verify that if any spikes appear during the transmission of a previous spike, they will be discarded. These discarded spikes are highlighted and crossed in red.

4.7.2.2 Analogue Part - DLL

To create the analogue module that would interface with the rest of the circuit, a Verilog-A behavioural model was created. This module sought to mimic the behaviour of the DLL depicted in Figure 4.25. To develop this module, the Cadence Virtuoso was used.

The model is constituted by a delay element that mimics the delay line of multiple delay inverters. This delay element module takes as input a delay value, and an input clock signal, and has as output a delayed version of this same input clock signal.

The rest of the DLL consisted of an analogue loop that calculates the difference between the phase of the delayed clock and the input clock, and based on the error between them, it calculates the new delay as the error times as parameter DELAY_STEP. It is also important to note that this process is simplified since the clock input frequency the DLL tries to achieve is known (in this case, 100 MHz).

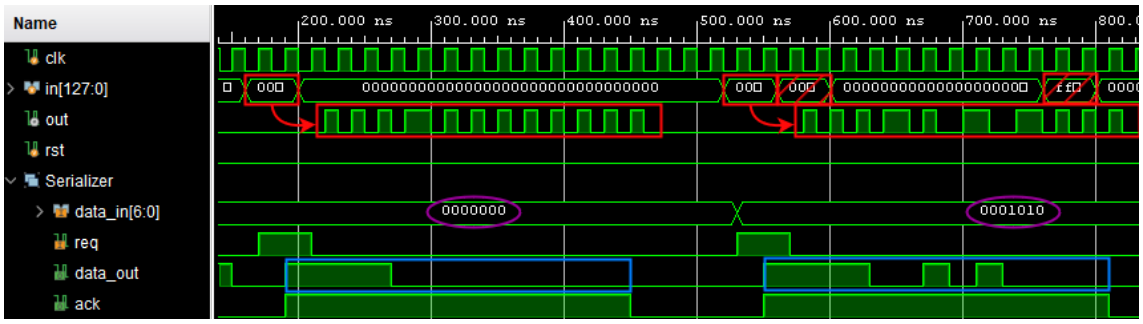


Figure 4.26: Waveform of AER Iteration 3 - Serializer signal behaviour

4.7.2.3 Mixed Signal Simulation

The execution of mixed-signal simulations was not feasible during the project current phase, primarily due to the intricate and time-consuming nature of the testing process. The complexities involved in setting up such simulations required substantial resources and time, which were limited in the current scope of the project. However, it is essential to acknowledge that this aspect remains a valuable avenue for future exploration and development. However, it is important to emphasize that this limitation is not a critical hindrance to the overall project objectives.

The current design of the DLL (Delay-Locked Loop) has been limited to the behavioural level, which means that the full chip design could not be achieved without a physical implementation of the DLL. Despite this constraint, it is crucial to recognize that the focus of the project has been on developing and validating the behaviour of the DLL, establishing its functionality and performance through behavioural simulation.

While mixed-signal simulations are undoubtedly valuable for assessing the interaction between analogue and digital components in chip design, the primary goal of the project was to explore potential transmission approaches. The DLL does not play an important role in this research, as this research is to understand better if the approaches are viable in a more general sense. The goal of this initial stage was to validate the functionality and performance of the DLL at a high level rather than diving into mixed-signal intricacies. This allowed the research to focus on broader aspects of transmission approaches without being constrained by the complexities associated with mixed-signal simulations.

With this said, looking ahead, the inclusion of mixed-signal simulations in future work would enhance the design overall validation and verification. This would allow a more comprehensive assessment of the entire chip behaviour, including analogue and digital interactions, ensuring a robust and accurate system performance representation.

4.7.3 Synthesis and Report Analysis

In the third iteration using the Genus tool, the same steps were taken for the TX module. After the same analysis was done, the system demonstrated stability at a maximum frequency of 300 MHz, which is a bit slower than the previous iteration. This decrease in frequency is due to the additional

circuitry used to support the preamble generation. It is important to note that although this is slower than the previous iteration, it is still threefold the value of the initial proposed required frequency.

For the RX module, the synthesis step was skipped, as its RTL-level module primarily consisted of a decoder module. This decoder module effectively generated spikes after the addresses following CDR on the DLL were decoded.

As per the rest of the report analysis, limited additional insights are available beyond the successful stability achieved by the system, even at higher frequencies. However, it is essential to recognize the significance of these findings, as they underscore the efficiency and reliability of the neuromorphic computing solution.

The resulting netlists from these steps were then used in the posterior step of doing the layout and creating a physical-level model of TX module.

4.7.4 Layout and Simulation Results

Since the RX part of the system is mainly composed of the analogue behaviour model, only the layout for the TX part was done. This layout process followed the same set of steps as previous iterations. This final design has a dimension of $103 * 101\ \mu m$.

Regarding the simulation, this TX design was also tested using the physical level delays. The simulations were done in the same manner as the previous iterations and the results showed that the module was capable of encoding, serializing and outputting the data with the correct preamble preceding it.

4.8 Comparison of results obtained

After all design and final optimizations have been done, the final chips designed have the following characteristics.

Table 4.1: Performance values for each chip

Iter	Lines used	Total Delay	Maximum Frequency of System
1	log2(N) = 7	31,009 ns	340 MHz
2	2	102,261 ns	410 MHz
3	1	140 ns[1]	300 MHz

[1] Ideal case, since it was not possible to test the physical delay.

Table 4.2: Area, Number of Cells used and Power results

		Area (μm)	**Number of Cells Used**	**Power Consumption** (W)
Iter1	TX	$96*90$	984	2.11169e-04
	RX	$108*104$	728	7.77260e-04
Iter2	TX	$92*93$	974	2.12669e-04
	RX	$108*106$	711	9.68466e-05
Iter3	TX	$103*101$	1421	4.72530e-04

Based on the results obtained, a comparison between the iterations provides valuable insights into the strengths and weaknesses of each approach. It allows us to understand the trade-offs and determine the ideal use case for each iteration.

Specifically, concerning the values obtained for the Total Delay and Maximum Frequency metrics, it is evident that the chips developed perform as desired and align well with the intended behaviour. Furthermore, these results demonstrate that there is still potential for achieving higher frequency speeds of at least threefold the initially desired frequency. This increase in frequency might be especially beneficial for the second and third iterations, which rely more on the frequency than the first due to the time spent during the serialization process.

When comparing the chip area of both Iteration 1 and 2 TX circuits, they are quite similar in size. The RX versions of both iterations also share this similarity in size. However, it is worth mentioning that Iteration 3's TX circuit is slightly larger. The increase in size can likely be attributed to the need to allocate space for the preamble generation circuitry. To address the timing dependency of this area, it may be beneficial to explore and adopt a less timing-dependent approach.

Another critical point of data is the maximum frequency of each TX-RX system. It is evident that the iteration with the greatest speed is iteration 2, but all iterations are capable of speeds threefold of the one initially defined.

It is possible to summarize the ideal use cases and scenarios for each AER iteration as follows:

- **First AER iteration (Foundational Approach) Ideal Use Case:** The first AER iteration, with a foundational approach, is suitable for applications where simplicity and low complexity are prioritized.

  Small-scale neuromorphic systems: When the number of neurons and size of address word used is relatively small, and the system complexity is limited. In scenarios where data transmission rates are not of utmost importance, the focus shifts to prioritizing the preservation of timing and speed between cores. Instead of heavily emphasizing demanding communication requirements, the primary concern becomes maintaining synchronization and efficient processing among the cores. This approach acknowledges that the seamless coordination of actions between cores holds greater significance than rapid data transfer in specific applications. By adopting this strategy, the system can strike a balance between effective intercore

communication and optimized performance, ensuring smooth and reliable operations for the specific application at hand.

- **Second AER iteration (SerDes-Assisted Communication with CLK Line) Ideal Use Case:** The second AER iteration, leveraging SerDes-assisted communication with a CLK line, is more suitable for applications that require higher data transmission rates and more efficient communication.

  Medium-scale distance neuromorphic systems: When the number of neurons increases beyond what the first iteration can efficiently handle. Moderate communication demands: For scenarios where data transmission rates need to be improved compared to the first iteration, but do not require the most advanced communication capabilities.

- **Third AER iteration (Complex SerDes System) Ideal Use Case:** The third AER iteration, integrating a complex SerDes system, is designed for applications that require even higher data transmission rates and more advanced communication capabilities.

  Large-scale neuromorphic systems: When dealing with a significant number of neurons and complex neural networks, demanding a more sophisticated communication infrastructure. High communication demands: For applications where data transmission rates must be at their highest, the communication system needs to handle complex data streams efficiently, and where the number of lines used makes a significant difference in routing and total area used.

By comparing the strengths and weaknesses of each iteration, it is easier to make informed decisions about which AER iteration best suits the specific requirements of a neuromorphic system, whether it is a small, simple system with low demands or a large-scale, complex one with high communication requirements, the most appropriate iteration can be chosen to achieve optimal performance and scalability.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

In conclusion, this dissertation has explored the field of neuromorphic computing and specifically focused on designing and implementing hardware architectures for Address-Event Representation (AER) based systems, with a special focus on the transmission aspects. This research aimed to develop a solution that leverages the unique capabilities of AER to create efficient, scalable, and high-performance neuromorphic systems.

This dissertation discusses the theoretical background of neuromorphic computing, including the concepts of SNNs and the use of spikes as the fundamental unit of communication and computation. The AER protocol's suitability for large-scale neuromorphic systems has also been explored, highlighting its advantages in terms of event-driven processing, parallel computation, and reduced energy consumption.

The research conducted in this dissertation has contributed to the existing body of knowledge by proposing a novel hardware architecture for AER-based transmission systems. Several key considerations, including high-speed digital signalling and efficient processing, guided the design of this architecture. Addressing these considerations resulted in multiple robust and scalable hardware solutions that effectively harness the power of AER in intercore and interchip transmission for advanced neuromorphic computing applications.

The development and implementation of the proposed hardware architecture involved several stages, including system design, physical design, and verification. Each stage required careful planning, iteration, and the use of specialized tools and methodologies. The results obtained through the implementation of the solution have demonstrated its effectiveness in terms of spike rate compatibility, efficiency in data transmission, and overall system performance.

As mentioned before, in this research, various hardware architectures were proposed, each tailored to specific scenarios and strengths. The approaches ranged from prioritizing simplicity and synchronization in small-scale systems to enhancing data transmission rates and communication efficiency in medium-scale applications. For large-scale neuromorphic systems with high communication demands, the proposed hardware architecture integrated a complex communication

system to handle data streams.

The outcomes of this research have demonstrated the effectiveness of these hardware architectures in their respective domains. However, it is to note that the results achieved could be further enhanced when coupled with newer and smaller technology.

5.2 Future Work

As a future work note, one promising avenue for further exploration is the implementation of mixed-signal simulation in the third iteration. Integrating mixed-signal simulation would enable the seamless combination of analogue and digital elements within the design environment, presenting exciting opportunities to achieve even better results in the realm of neuromorphic computing. By incorporating analogue elements alongside digital components, researchers can harness the benefits of both domains, leveraging the efficiency and speed of digital processing and the continuous and energy-efficient nature of analogue signals. This hybrid approach could lead to novel and highly efficient neuromorphic systems, unlocking new possibilities for understanding brain-inspired computation and pushing the boundaries of artificial intelligence. The integration of mixed-signal simulation in the third iteration would undoubtedly contribute to advancing the field of neuromorphic computing, opening new avenues for groundbreaking research and applications.

Another potential direction for future research, integrating AER with NoC architectures, holds significant promise in advancing the field of neuromorphic computing. Combining AER's efficiency and event-driven nature with the scalable and low-power communication capabilities of NoCs can lead to the creation of powerful and flexible neuromorphic systems. The seamless integration of AER and NoC would enhance inter-core connectivity, enabling rapid and dynamic data transmission among processing elements while maintaining synchronization and timing critical for neuromorphic networks.

Finally, in this dissertation, the process of digital design post-Verilog creation could have been significantly streamlined by incorporating more scripts and adopting better file organization and manipulation practices. By doing so, future researchers working on similar topics could benefit from a well-defined guide, optimizing their workflow and expediting their design process. Also, creating a guide based on this improved workflow would serve as a valuable resource for future researchers working in the same environment with the same tools, enabling them to build upon this work efficiently and foster innovation in the field of neuromorphic computing and digital hardware design.

References

[1] Maxence Bouvier, Alexandre Valentian, Thomas Mesquida, Francois Rummens, Marina Reyboz, Elisa Vianello, and Edith Beigne. Spiking neural networks hardware implementations and challenges: A survey. *J. Emerg. Technol. Comput. Syst.*, 15(2), apr 2019. URL: https://doi.org/10.1145/3304103, doi:10.1145/3304103.

[2] Raqibul Hasan, Tarek M. Taha, and Chris Yakopcic. On-chip training of memristor crossbar based multi-layer neural networks. *Microelectronics Journal*, 66:31–40, 2017. URL: https://www.sciencedirect.com/science/article/pii/S0026269216301732, doi:https://doi.org/10.1016/j.mejo.2017.05.005.

[3] J. Lazzaro, J. Wawrzynek, M. Mahowald, M. Sivilotti, and D. Gillespie. Silicon auditory processors as computer peripherals. *IEEE Trans. Neural Networks*, 4(3):523–528, 1993.

[4] Jian Liu, Li-Rong Zheng, and Hannu Tenhunen. Interconnect intellectual property for network-on-chip (noc). *Journal of Systems Architecture*, 50(2):65–79, 2004. Special issue on networks on chip. URL: https://www.sciencedirect.com/science/article/pii/S1383762103001152, doi:https://doi.org/10.1016/j.sysarc.2003.07.003.

[5] R. Jacob Vogelstein, Francesco Tenore, Ralf Philipp, Miriam S. Adlerstein, David H. Goldberg, and Gert Cauwenberghs. Spike Timing-Dependent Plasticity in the Address Domain. 2002.

[6] Carlos Zamarreno-Ramos, Rafael Serrano-Gotarredona, Teresa Serrano-Gotarredona, and Bernabe Linares-Barranco. Lvds interface for aer links with burst mode operation capability. In *2008 IEEE International Symposium on Circuits and Systems*, 2008. doi:10.1109/ISCAS.2008.4541500.

[7] Wolfgang Maass. Networks of spiking neurons: The third generation of neural network models. *Neural Networks*, 10(9):1659–1671, 1997. URL: https://www.sciencedirect.com/science/article/pii/S0893608097000117, doi:https://doi.org/10.1016/S0893-6080(97)00011-7.

[8] M Schwartz. Telecommunication networks: Protocols, modeling, and analysis. 1987.

[9] A S Tanenbaum. Computer networks, 2nd edition- prentice-hall international. 1989.

[10] L.Miró-Amarante, A. Jiménez-Fernández, A. Linares-Barranco, F. Gómez-Rodríguez, R. Paz, G. Jiménez, A. LVDS Serial AER Link performance. 2007.

[11] Jongkil Park and Sang-Don Jung. Presynaptic spike-driven spike timing-dependent plasticity with address event representation for large-scale neuromorphic systems. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2020. doi:10.1109/TCSI.2020.2966884.

[12] Fernando Pérez-Peña, Arturo Morgado-Estévez, and Alejandro Linares-Barranco. Interspikes-intervals exponential and gamma distributions study of neuron firing rate for svite motor control model on fpga. *Neurocomputing*, 149:496–504, 2015. URL: https://www.sciencedirect.com/science/article/pii/S0925231214010388, doi:https://doi.org/10.1016/j.neucom.2014.08.024.

[13] Luiz Carlos Gouveia, Thomas Jacob Koickal, and Alister Hamilton. An asynchronous spike event coding scheme for programmable analog arrays. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2011. doi:10.1109/TCSI.2010.2089552.

[14] Andres Upegui, Yann Thoma, Eduardo Sanchez, Andres Perez-Uribe, Juan Manuel Moreno, and Jordi Madrenas. The perplexus bio-inspired reconfigurable circuit. pages 600–605, 2007. doi:10.1109/AHS.2007.105.

[15] J. Manuel Moreno, Jordi Madrenas, and Lukasz Kotynia. Synchronous digital implementation of the aer communication scheme for emulating large-scale spiking neural networks models. In *2009 NASA/ESA Conference on Adaptive Hardware and Systems*, pages 189–196, 2009. doi:10.1109/AHS.2009.14.

[16] Simone Aiassa, Paolo Motto Ros, Guido Masera, and Maurizio Martina. A low power architecture for aer event-processing microcontroller. pages 1–4, 10 2017. doi:10.1109/BIOCAS.2017.8325170.

[17] Paolo Motto Ros, Marco Crepaldi, Chiara Bartolozzi, and Danilo Demarchi. Asynchronous dc-free serial protocol for event-based aer systems. pages 248–251, 12 2015. doi:10.1109/ICECS.2015.7440295.

[18] Fopefolu Folowosele, Jonathan Tapson, Mark Vismer, and Ralph Etienne-Cummings. Wireless systems could improve neural prostheses. *Spie Newsroom*, 01 2007. doi:10.1117/2.1200709.0854.

[19] Jilin Zhang, Jinsong Wei, and Hong Chen. An address event representation circuits design with rotation priority against pulse collision. 2019. doi:10.1109/EDSSC.2019.8754504.

[20] J. Georgiou and A.G. Andreou. High-speed, address-encoding arbiter architecture. 2006.

[21] J. Lazzaro and J. Wawrzynek. A multi-sender asynchronous extension to the aer protocol. In *Proceedings Sixteenth Conference on Advanced Research in VLSI*, pages 158–169, 1995. doi:10.1109/ARVLSI.1995.515618.

[22] Misha Anne Mahowald. *VLSI Analogs of Neuronal Visual Processing: A Synthesis of Form and Function*. PhD thesis, USA, 1992. UMI Order No. GAX92-32201. doi:10.5555/168951.

[23] Xuan-Thuan Nguyen, Hong-Thu Nguyen, and Cong-Kha Pham. A scalable high-performance priority encoder using 1d-array to 2d-array conversion. *IEEE Transactions on Circuits and Systems II: Express Briefs*, PP:1549–7747, 02 2017. doi:10.1109/TCSII.2017.2672865.

[24] Joshi, Kiran Raj, Johnson, and Louis G. Design of 128 bit round robin priority encoder. 07 2004. doi:https://hdl.handle.net/11244/300897.

[25] Xilinx Vivado - Accessed: May 2023. URL: https://www.xilinx.com/products/design-tools/vivado.html.

[26] Cadence Xcelium - Accessed: May 2023. URL: https://www.cadence.com/en_US/home/tools/system-design-and-verification/simulation-and-testbench-verification/xcelium-simulator.html.

[27] Cadence Genus - Accessed: May 2023. URL: https://www.cadence.com/en_US/home/tools/digital-design-and-signoff/synthesis/genus-synthesis-solution.html.

[28] Cadence Innovus - Accessed: May 2023. URL: https://www.cadence.com/en_US/home/tools/digital-design-and-signoff/soc-implementation-and-floorplanning/innovus-implementation-system.html.