PRACTICAL APPROACHES TOWARDS CLOCK SKEW MANAGEMENT IN LARGE SCALE DIGITAL ACCELERATORS FOR RF EMULATION

A Dissertation
Presented to
The Academic Faculty

By

Prachi Sinha

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in the
School of Electrical and Computer Engineering

Georgia Institute of Technology

May 2023

PRACTICAL APPROACHES TOWARDS CLOCK SKEW MANAGEMENT IN LARGE SCALE DIGITAL ACCELERATORS FOR RF EMULATION

Thesis committee:

Dr. Saibal Mukhopadhyay
Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Tushar Krishna
Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Justin Romberg
Electrical and Computer Engineering
Georgia Institute of Technology

Date approved: April 28, 2023

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my advisor, Dr. Saibal Mukhopadhyay, for his guidance, support, and encouragement throughout this research project. His curiosity, passion, and enthusiasm for research were infectious, and I came out of every meeting with renewed energy. He will always be a source of motivation in my career. Dr. Mukhopadhyay's door was always open whenever I had a question pertaining to my research, and he has been a great advisor and mentor.

I would also like to thank my dissertation committee members, Dr. Tushar Krishna and Dr. Justin Romberg, for taking time out of their busy schedules to review my work. Their feedback and suggestions were invaluable in helping me to improve my thesis.

This thesis could not have been possible without the help and support of my lab members, Nael Mizanur Rahman and Sudarshan Sharma. They provided me with invaluable guidance and support throughout the research process and were always present to support me whenever I hit any roadblocks. I would also like to thank my lab mates for creating a conducive and encouraging environment in our shared workspace. My time at Georgia Tech was made memorable by my friends Shambavi Ganesh, Murali Sethuraman, and Hemant Kumawat, who were my support system in Atlanta. They provided me with much-needed positivity and undying support throughout my time in graduate school.

Finally, I would like to thank my family for their love and support. My parents have always been there for me, and they have always believed in me. Their sacrifices have made my success possible. I am also grateful to my sister for her love and kindness.

I am truly grateful to all of the people who have helped me along the way. Their belief in me has made this accomplishment possible. A special thanks to Shantam, for always being there for me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

This research discusses approaches for clock skew management in large-scale digital accelerators for RF emulation design. A high-performance computing near-memory accelerator pro-typed for RF emulation is considered. Digital Application Specific Integrated Circuits (ASICs) have become essential tools for the hardware acceleration Digital ASICs have become essential tools for the hardware acceleration

ASICs have become crucial for the hardware acceleration of complex mathematical models and algorithms due to their ability to achieve faster processing than general-purpose CPUs and GPUs. Emulating RF interactions between dynamic objects at high bandwidth requires distributed processing nodes with substantial computing capacity. In such large-scale designs, clock distribution networks must maintain global synchronization while minimizing skew and delay. A widely practiced approach of skew mitigation apart from reducing operating frequency is adding delay buffers. This can prove to have a considerable power overhead as a bigger skew requires many buffers and results in trade-offs during the physical design of digital ASICs.

A mesochronous scheme is implemented which integrates the clock distribution with the data bus and uses phase detection for synchronization on a functional model of an Radio-Frequency (RF) accelerator. This work estimates an increase of 0.6% in power consumption and a 3.8% increase in the area when the proposed scheme is integrated with the existing accelerator designs.

CHAPTER 1

INTRODUCTION AND BACKGROUND

In recent years, Digital Application Specific Integrated Circuits (ASICs) have become essential tools for the hardware acceleration of complex mathematical models and algorithms, primarily due to their ability to achieve orders of magnitude faster processing as compared to general-purpose CPUs and GPUs. With the rapid advancement of technology and the growing complexity of computational tasks, the demand for highly efficient and specialized processing solutions has increased tremendously. One particular area of interest is the emulation of Radio Frequency (RF) interactions between dynamic objects [1] at very high bandwidth, which necessitates distributed processing nodes, each with a substantial compute capacity, to provide accurate and timely results. However, as the size of the scenario and the complexity of the object being emulated grows, the size of the Application-Specific Integrated Circuits (ASICs) responsible for managing these computations expands accordingly.

For large-scale designs, digital accelerators and systems are no longer limited to single-clocked designs. While synchronous digital circuits still dominate, many modern designs incorporate multiple clock domains operating at varying frequencies [2]. Despite the benefits of global synchronization, such as ease of design and availability of CAD tools, it can hinder the maximum achievable clock frequency due to physical limitations. The clock distribution networks need to maintain global synchronization while minimizing skew and delay. Despite their widespread use in clock distribution networks, H-tree implementations [3] often rely on symmetric layouts for optimal performance, limiting design placement flexibility. This constraint can be particularly restricting for the designers. Moreover, global synchronization implemented with delay-matched clock trees has many drawbacks. It needs more metal layers and results in a high process cost [4].

To address signal delay issues, wider metal wires are often used, which further contributes to higher power consumption in the clock distribution network. As a result, alternative clock distribution schemes can allow for greater flexibility in design placement depending on the ASIC design.

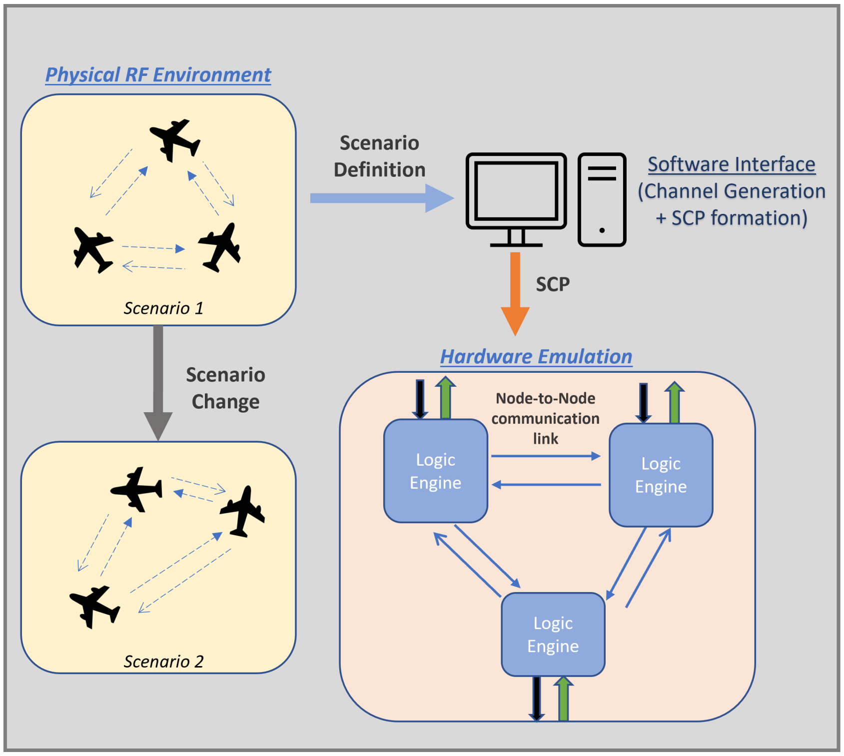


Figure 1.1: RF system Overview

RF emulator accelerators are specialized hardware devices that enable faster and more accurate simulation of wireless communication systems in the physical world as seen in Figure 1.1. They are designed to emulate the behavior of radio-frequency (RF) systems, including transmitters, receivers, and channels. By providing a faster and more efficient alternative to software-based simulation tools, RF emulators can significantly reduce development time and costs for wireless communication products. There is a possibility of FPGA implementation [5] of RF channel emulators to provide repeatability, high accuracy, reliability, and dynamic testing ability of different scenarios [5]. There is a need for custom accelerators due to the inability of FPGA testbeds to achieve both high computational throughput and low computational latency simultaneously. [6] proposes a novel architecture for an RF emulation accelerator and presents a digital design prototyping of the architecture. The design in [6] is a high-performing compute in near-memory accelerator operating a 2.5GHz and implemented on 28nm CMOS technology and is under consideration in the thesis and will henceforth be mentioned by its project name ARION.

The implementation of RF accelerators is a challenging task due to the complexity of the mathematical models used to simulate RF objects and their interactions, resulting in large and computationally intensive designs. The ARION design scales up in size as the number of RF objects or nodes increases. This blow-up in size is evident; a 4 object system has a projected area of 3.9 mm^2, and an 8 node system scales up to 29.6 mm^2 indicating an approximate 600% increase [6]. The increase in the area of the chip forces us to address increasingly complex timing issues to achieve timing closure which might not have been evident in a smaller single-node system.

Moreover, the synchronous blocks in design vary in function and design such that the clock reaching each block from the VCO suffers from a considerable and non-uniform inertial delay in each of the blocks. This can be seen in Figure 1.2 which highlights the inertial delay due to varying design sizes. The difference between arrival times leads to clock skew in synchronous designs which when not accounted for, results in incorrect operation and metastability in data transfer.

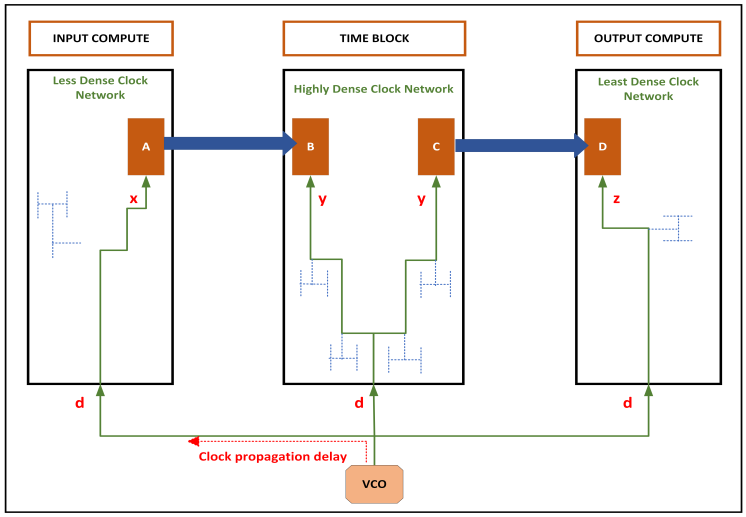


Figure 1.2: Inertial delay in sub-modules of accelerator design due to large scale design of RF accelerator not to scale

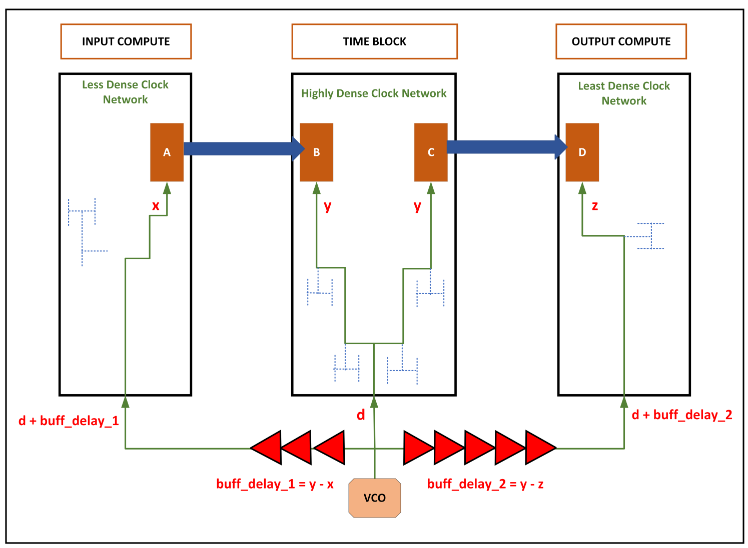


Figure 1.3: Voltage controlled delay Buffers to equalise skew

The widely practiced approach to reduce clock skew in the accelerator is to add delay buffers in clock tree lines (Physical Design CTS). Delay buffers help adjust arrival times of the clock signals as seen from Figure 1.4. This introduces three major issues - large violations require a very large number of delay buffers. Secondly, to ensure the post-silicon ability to tune each clock chain needs a separate power supply and power delivery network which gets very complex very fast. Lastly, power consumption increases considerably due to high switching activity at high frequency. Though the buffering strategy to achieve the desired timing performance achieves the purpose, it is a highly tedious and power-hungry approach. Furthermore, the placement and routing of the delay buffers must be carefully optimized to ensure that the added delay is distributed evenly across the clock tree and that it does not introduce new sources of skew or violate other design constraints. This can be a challenging and iterative process and requires careful consideration and balancing of the conflicting design constraints such as timing, power consumption, and area overhead. the impact on other design parameters especially the target frequency requires iterative effort and sometimes can also be a limitation.

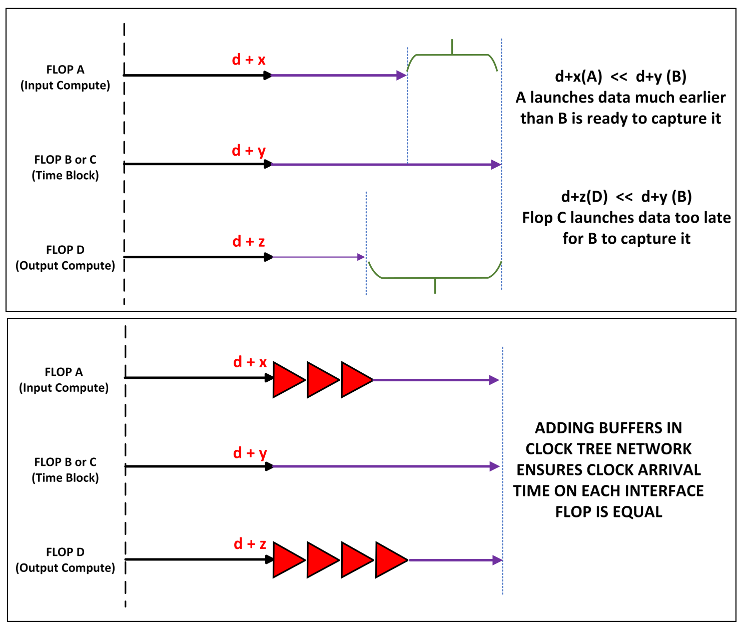


Figure 1.4: Representation of clock arrival time before and after delay buffer

Synchronization hierarchy defines multiple schemes depending on event-clock relation. Synchronous clocks are derived from the same source (PLL or crystal) and have the same phase but may or may not have the same frequency. Alternatively, clocks are asynchronous if they originate from different crystals and have an arbitrary phase difference.

An asynchronous scheme when thought of as an alternative, typically involves some form of handshaking either at the flit level or packet level, which leads to a drop in interface throughput or stalls the system. Both of these scenarios are unfavorable as the RF accelerators emulator represents continuously moving object EM waves emitters.

The design of large-scale System-on-Chips (MPSoCs) can incorporate the Globally Asynchronous Locally Synchronous (GALS) design paradigm to reduce power consumption. Network-on-Chips (NoCs) in particular tend to employ multiple GALS techniques to interconnect numerous CPU cores within a single chip [7].

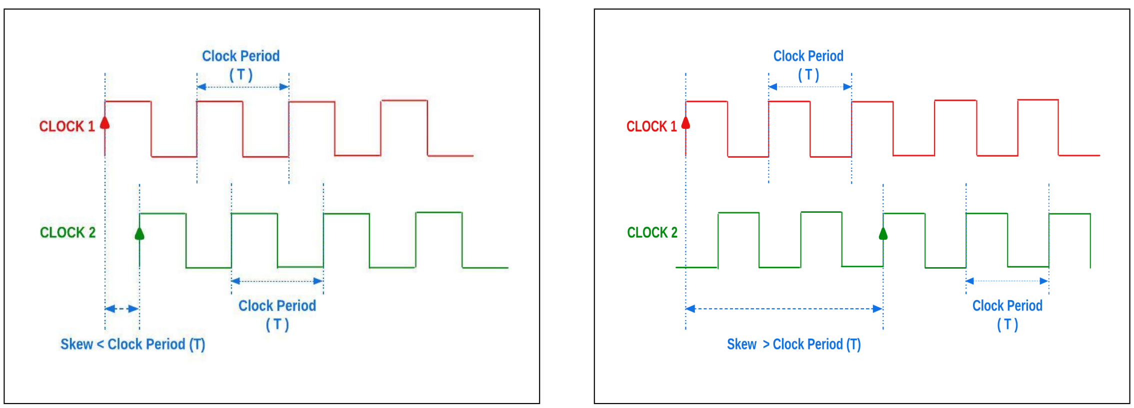
Mesochronous synchronization takes a middle ground between synchronous and asynchronous styles of design. The clocks operate with the same frequency but there may incur an arbitrary but constant phase difference between the clock arrival edge. Mesochronous clocking appears to be a situation we see in most systems, where the synchronous blocks are the sub-modules on the chip clocked by the same source but data flops are suffering through a constant phase difference due to network delay. The skewed clock situation in large designs can thus be recognized as a mesochronous synchronization problem.

This study aims to investigate design approaches to handle skews in large-scale signal processing-based RF accelerators. Considering the requirements of a high-performance RF digital system, it is necessary to have loss-less transmission while maintaining the required throughput of the system. The upcoming chapters talk more about the findings justifies the approach chosen with a proof of concept to mitigate the effects of skew in large-scale systems.

CHAPTER 2

METHODOLOGY

This study characterizes observable skew as the phase difference between the clocks is less than a single-cycle as shown in Figure 2.1 or multi-cycle skew. ARION employs a built-in mechanism of a synchronous external set being propagated to all the blocks to minimize the skew and limit it to a single clock cycle. This study largely focuses on cases when the skew is a fraction of the total clock period. This chapter gives an overview of the system at hand and explains the proposed approach to manage clock skew.



(a) Single-Cycle Clock Skew (b) Multi-cycle Clock Skew

Figure 2.1: Characterisation of Clock Delay

2.1 Accelerator Design

This section provides an overview of the design and architecture under consideration, as proposed in [6].

The ARION chip is specifically designed to accurately simulate the real-world scenario of multiple EM waves interacting with RF objects. To conduct this study, a system of four on-chip objects was considered to provide realistic emulation of complex electromagnetic interactions. Figure 2.2 presents a comprehensive overview of the design and modeling of

ARION, a high-performance computing near-memory accelerator for RF emulation. The architecture and key features of ARION are discussed in detail, along with the scenario realization approaches used to emulate real-world RF object interactions.

ARION models each object as a compute node or accelerator logic engine simultaneously interacting with each other based on scenario programming. The accelerator operates in transmitting and receiving mode to combine signals from other nodes with its local RF attributes, applying physical models and generating real-time signals for other nodes and their local receivers. For communication between nodes, on-chip wires are utilized for single-chip implementations with multiple nodes. To adapt to dynamic changes in the emulated scenario, external control software translates the physical environment to scenario configuration packets (SCPs) and re-configures the accelerator through a Scenario Update (SU) signal every 1 ms without stalling real-time operation [6]. The SU signal control loads the already buffered computation coefficients as per the new scenario programming.

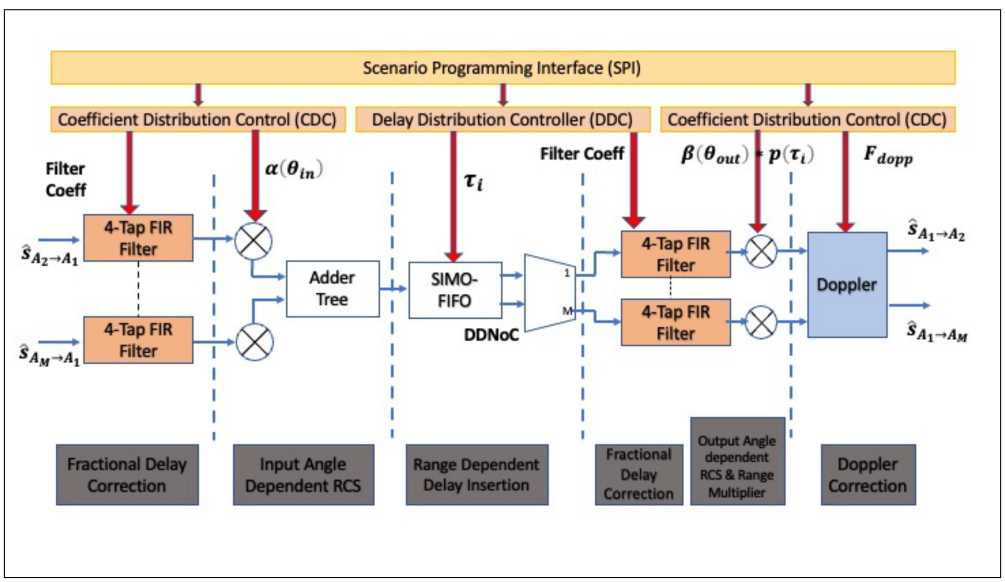


Figure 2.2: ARION Design

The node's architecture is designed to emulate various physical factors, such as path delay, path loss, RCS, antenna gains, input/output fractional delay filters, and Doppler correction. To achieve this, the architecture is divided into two types of blocks: the compute block (data path) and the time block (control path). The data path contains all the compute modules responsible for signal processing, while the control path includes a FIFO for efficient data buffering based on direct path delay[6].

1. Compute Block: The compute engine solves one part of a signal processing-based mathematical model which characterizes the RF system signal processing design processes signal values based on the physical phenomena that would occur in the real world with other RF objects. To account for fractional delay correction, a 4-tap FIR filter is used, followed by RCS gain multiplication that is dependent on the incident angle of the radar[6].

2. Time Block: The time block or the control block is a single-in-multiple-out (SIMO) FIFO simultaneously emulates signal propagation delay to multiple objects or nodes. The propagation delay is programmable and scenario-dependent. The time block is also responsible for sending out a pulse every millisecond to load new scenario coefficients[6].

The time block houses Single-Input Multiple-Output (SIMO) SRAM memory [6] to simulate the delay experienced during signal reflection. Time block experiences the most significant skew in the flops due to clock inertial lag because of its large size. The primary challenge is to accurately capture data at the expected clock edge of the next module without causing data metastability.

The data flow on the chip is from the input compute block to the time block, and then to the output compute block. Additionally, the scenario update signal, originating from the time block every millisecond, must be transmitted to each compute block to ensure that computations are carried out with the updated scenario coefficients. ARION starts computation when an external start signal is observed in the system. This work proposes a strobe-based clock distribution to ensure that clock skew is handled and the data is being latched correctly at the clock edge of receiving flop.

2.1.1 Testbench Automation

ARION is a full-scale, 4 object, tape-out ready chip, and thus it included Design For Test (DFT) modules to program physical-world parameters through the Scenario programming interface (SPI) [6]. The coefficients are loaded through the SPI which takes in bit-wise values through a single pin at a much slower speed through a serial-to-parallel interface. Serial programming is performed through the on-chip pin operating at a much slower clock rate than the chip frequency. Within a single scenario, multiple coefficients need to be programmed for each object in a single scenario. Because of the multi-object nature of the design, to load scenario coefficients thousands of serial programming is needed to be done.

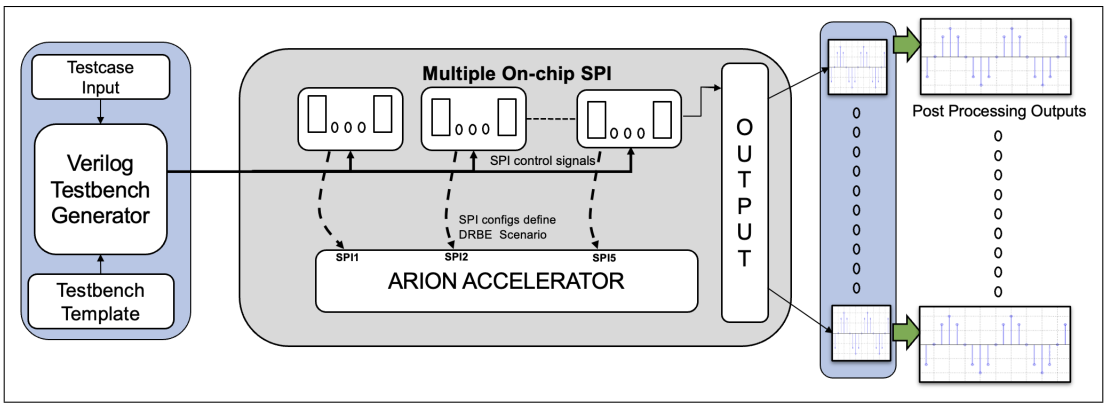


Figure 2.3: Automated Testbench Generation Flow

Figure 2.3 illustrates a Python-based automation environment that generates a Verilog-based testbench for serially loading configurations onto the chip. The automation script took a coefficient input file generated from external software, and a basic testbench template to produce a testbench. There are multiple such serial-to-parallel interfaces on the ARION test chip, one for each of the blocks in the design, and hence an automated approach saves time and effort. It is to be noted that the design of this accelerator is not a contribution of this thesis but the in-house automation flow contributed to the functional verification efforts of the ARION design and also in testing. Results obtained from using this flow can be found in [8].

2.2 Single-Cycle Skew

The proposed methodology to handle skew is inspired by [9], to distribute clock signals in the system alongside data and control values. The approach involves adding a strobe signal to the data distribution channel between the blocks, which serves as a local clock. This allows for the data to be re-timed at the receiving block with respect to the clock signal of the same phase as launched data, thus reducing the risk of metastability. This clock distribution signal is referred to as a "strobe" signal and it runs at half the frequency of the clock.

The proposed synchronization scheme is presented in Figure 2.4. A strobe travels with the data channel from the transmitting to the receiving block. Having the strobe signal with half the switching frequency as the clock is advantageous as the data and strobe would be undergoing the same delay, switching once in the span of one clock period.

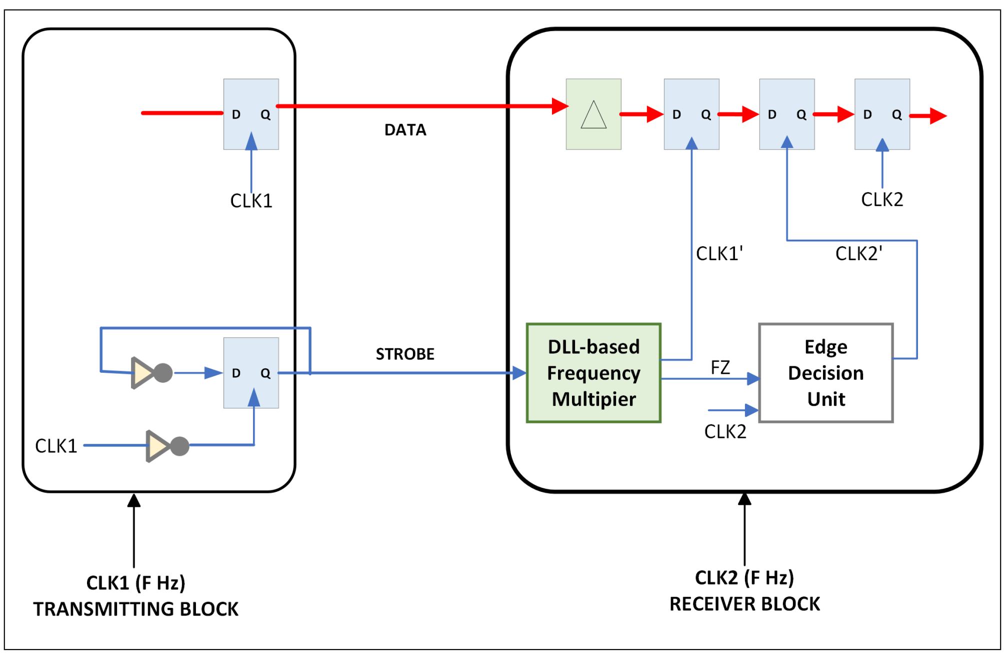
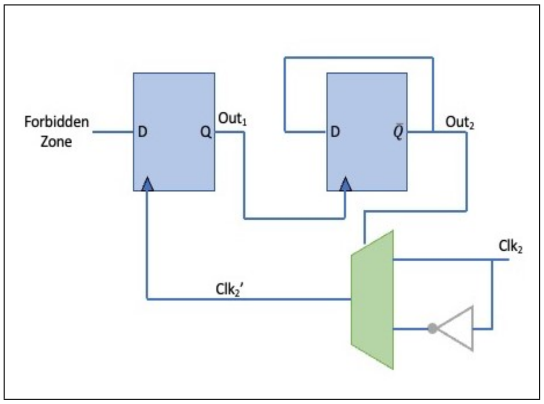


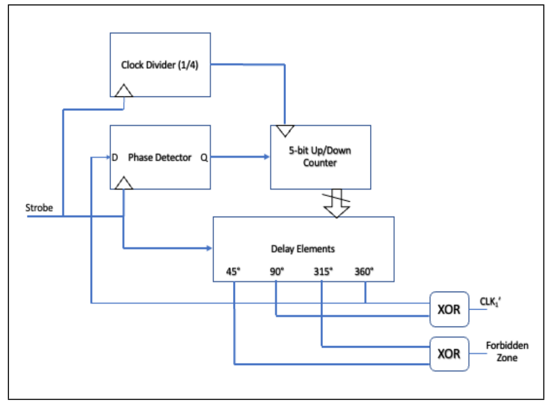
Figure 2.4: Proposed Mesochronous synchronization scheme between a transmitter and receiver module

On the transmitting block, a binary divider is used to generate the strobe signal along with the continuous stream of data flowing to the receiving block.

To avoid metastability at the receiver module, the incoming strobe signal is utilized to reconstruct the sender clock (CLK1') with the same frequency as the data and local clock at the receiver. A Delay-Locked Loop (DLL) based frequency-multiplier (Figure 2.5) generates CLK1' from the strobe and also produces a failure (FZ) signal.



(a) Edge Detection Unit



(b) Delay Locked Loop implementation as presented in [**meso**]

Figure 2.5

In [4] failure zone is defined as the time window within which the input data should be stable to avoid metastability and can be thought of as the sum of the setup and hold time of flip-flops. However, detecting the setup time and the hold time separately and adding them together using a circuit is impractical. [4] defines a more realistic failure zone is defined as a forbidden time window for the half-swing transient point of the input. When the input reaches the half-swing transient point within the failure zone, edge-triggered flip-flops or latches may malfunction.

A DLL is a feedback system often used for the synchronization of signals. DLLs can produce a 90-degree phase-shifted version of the incoming strobe signal. The delay elements in DLL maintain a fixed phase relation between the incoming strobe and output signal which can simply be XOR-ed to reconstruct the sender clock (CLK1') [9]. This study focuses on the possible implementation of mesochronous clocking in ARION and hence assumes a behavioral implementation of the DLL to achieve proof of concept. DLL design is outside the scope of this work. The failure zone signal is also generated using a behavioral implementation and adding up the setup and hold time of edge-triggered flip flops in 28nm technology.

Each edge of the strobe produces a positive edge of the reconstructed CLK1' clock to latch the incoming data which can be buffered to match the delay of the frequency multiplier. An edge decision unit is used in [9] to implement edge adaptability of the local clock in the receiver unit. The negative edge of CLK2 is used to re-time the data with the local clock in the case where the positive edge of CLK2 falls in the data transition zone denoted by the failure zone signal. A last stage of pipe-lining is added to latch the data at the positive edge of CLK2, the receiver local clock.

The negative skew faced by the Scenario Update (SU) signal in theory can undergo double-flopping to avoid metastability. There are other more complex and demanding techniques known to synchronize a pulse but as the SU is single clock period long and occurs once every scenario, it could be safely latched using the brute-force nature of two back-to-back edge-triggered flip flops.

Figure 2.6 shows an implementation of the above-mentioned mesochronous scheme in ARION design, between the input compute and time block. For this study, a functional model of ARION was considered as the compute-heavy design resulted in longer simulation time and other tool-related limitations.

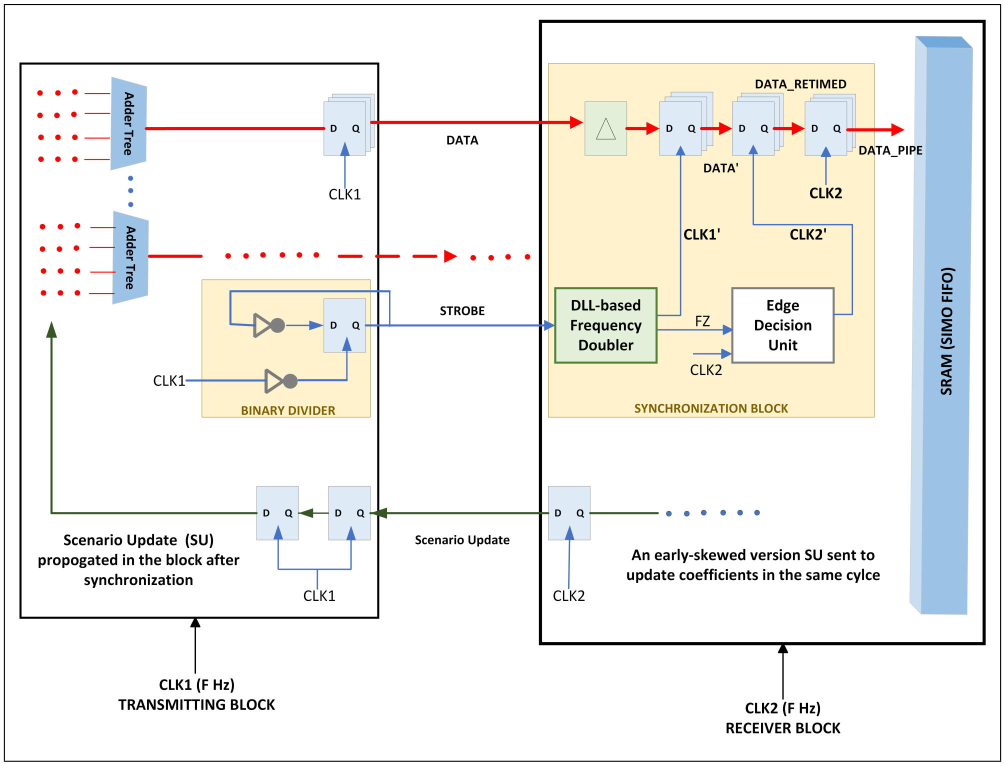


Figure 2.6: Integrated mesochronous scheme with ARION input compute and time block

The functional testbed of ARION is explained in detail in the next chapter of this document.

2.3 Multi-Cycle Skew

Some initial exploration was done for a more generalized scenario where multi-cycle skew can be seen in clocks traveling a long distance on-chip and doesn't utilize an external signal to start operation on-chip. In this case, it is assumed that the clock at the time block lags and the clock at the compute leads, and the difference between the clock edge arrival between two clocks is a constant multiple of the clock period.

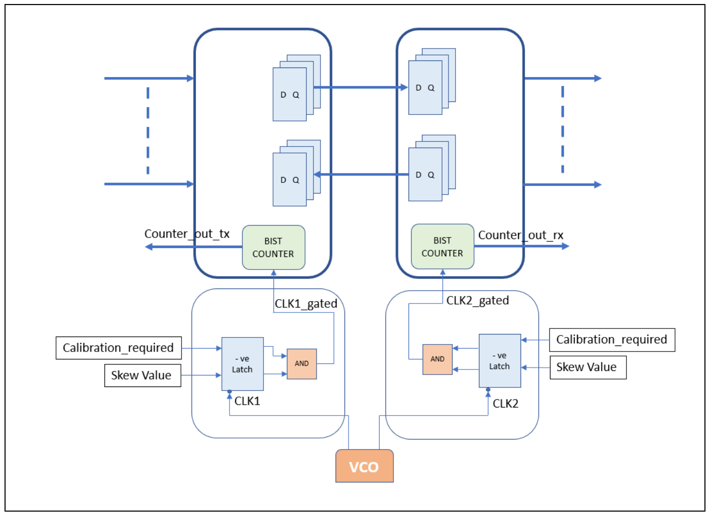


Figure 2.7: A possible BIST approach with clock gating control

In an ideal scenario, if the skew of leading clocks as compared to the lagging clock is known, edge alignment can be achieved by gating the leading clock. One potential post-silicon solution to determining the skew value in a system, which can be a challenging task, is to use a Built-in-Self-Test (BIST) approach. Here, the clock can be gated based on a programmable value through an input to the chip as seen in Figure 2.7. The chip once connected to the clock can go through a calibration phase during the run. A counter is observed in each of the synchronous blocks which starts counting cycles at the same absolute time. This experiment can run for a specific amount of time and the counter values can be observed in the end. The difference in counter value for each of the blocks will give us the skew. This skew can now be programmed back into the chip to gate the leading clocks and align the edge with the leading clock.

A functional implementation of this approach was carried out on the same testbed or testing model used for the single-skew scenario.

CHAPTER 3
RESULTS

This chapter discusses the results obtained from the implementation of mesochronous synchronization. To provide proof of concept, the methodology was first implemented and verified functionally through RTL simulations. Once the function was verified, a post-synthesis netlist was generated to perform timing simulations using post-synthesis standard delay files.

3.1 Testing Model

A simplified but functional model of the accelerator as seen in Figure 3.1 was used as a test bed to verify the approach of single-cycle skew. The motivation behind creating a functional model is the long simulation run time of the accelerator and also complex black-box debugging. A less computation-intensive but functionally equivalent version of ARION was considered which helped in the easy observation of timing issues and concentrate on interfacing signals.

A simplified input compute block that generates 1024 integer input samples was used, as opposed to the accelerator design which has a separate block to generate IEEE 16-bit floating point real and imaginary samples. While the input samples in the accelerator go through a 4-tap FIR filtering and an adder tree, these components were not included in the functional dummy module created for testing. This is mainly because the work analyzes the signal at the interface stage after the computation, and the computation process in itself is outside the scope of this experiment.

In this chip design, the Single-In Multiple-Out (SIMO) FIFO is utilized to store 1024 32-bit samples per object using foundry SRAMs. However, for the purposes of this experiment, a smaller design was used to implement a synthesizable synchronous FIFO. This

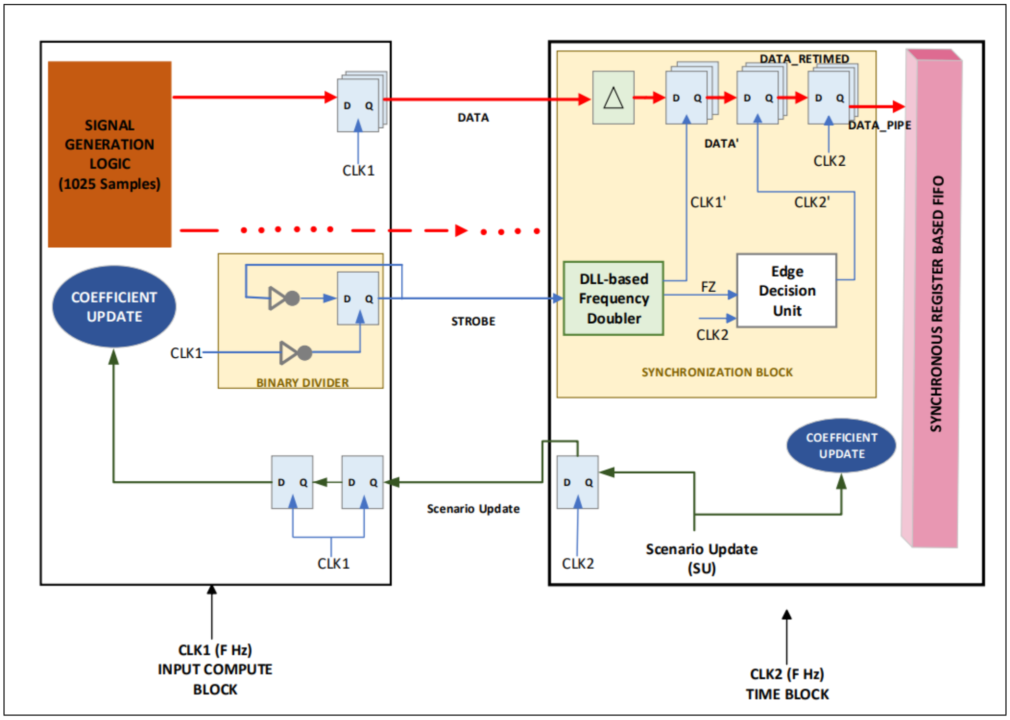


Figure 3.1: A simplified functional model of ARION

design provides a programmable delay of up to 1024 clock cycles per object before the data is read out to the output compute block. To enable SIMO operation, multiple read pointers were implemented in the FIFO. The output compute block in accordance with the design is modeled to consolidate all the signal components for each object using a weighted adder.

3.2 Behavioral Models

A behavioral model of DLL is used to introduce phase shifted version of the input reference signal, this is the case of the strobe signal. An XOR-gate base frequency multiplication is performed with the reference signal and a 90-degree shifted version of the signal. Through observation for 28nm CMOS technology edge-triggered devices, an approximate window of metastability is chosen to ± 27 around the signal edge. Literature survey on 28nm CMOS implementation of DLLs indicates power consumption of DLL in the 1.5mW to 20mW range and area occupied to be in between 0.07mm^2 and 0.0056mm^2(Table 3.1).

Table 3.1: Reported Power and Area of existing DLLs implementations on 28nm CMOS Technology

PRIOR WORK	FREQUENCY RANGE	POWER	AREA
DIGITAL MULTIPLYING DELAY-LOCKED USING PWM [10]	2.4 GHz	1.5 mW AT $1V$	0.024mm^2
PHASE ROTATOR DELAY-LOCKED [11]	2.4GHz - 8GHz	20.2 mW	0.027mm^2
FDSOI ULTRA LOW POWER FACTORIAL DLL (2021) [12]	1.5GHz - 2.0GHz	0.37 mW AT $1V$	0.07mm^2
ULTRA COMPACT ALL-DIGITAL MDLL [13]	1.55GHz - 3.35GHz	1.45 mW AT 0.8V	0.0056mm^2

3.3 Simulation Results

Simulation of a case where the skew is between the blocks is post-synthesis netlist and SDF of the functional model of ARION. The SDF files capture the standard cell delays, both sequential and combinational, along with the rise and fall time specifications which are inherently added to the propagation delay of the signals.

Figure 3.2 show clocks for each of the blocks in the ARION operating at the same frequency but with different inertial delays, which causes them to have different skews. The input block generates and transmits data to the time block after some computation as the start signal is asserted.

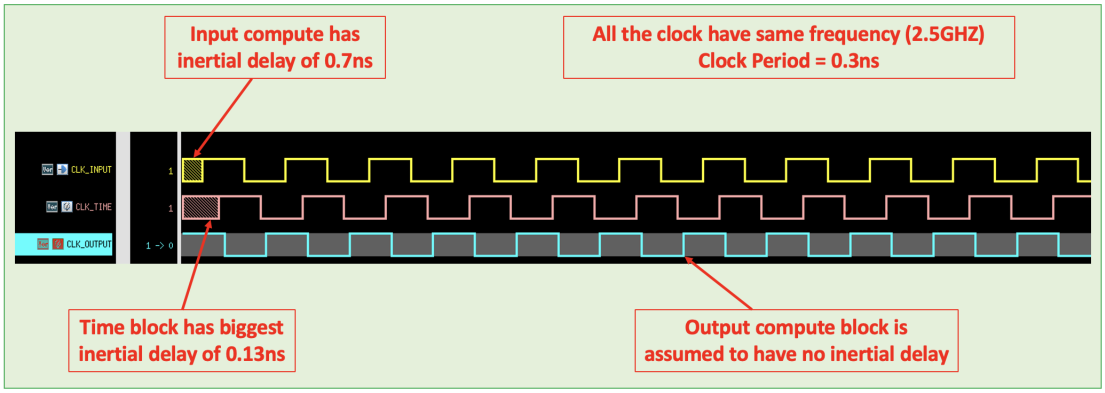


Figure 3.2: Clock given to each of the blocks in the system with varying inertial delay to simulate different clock arrival times. The clock is leading in computing blocks with respect to the lagging clock seen in the time block.

1-bit strobe and 32-bit wide signals corresponding to each scatterer (reflective surfaces on an object) travel from the leading input compute block to the lagging time block. The strobe is generated at the negative edge of the input block clock CLK1. At the receiver, i.e time block, the first stage of data latching is at the positive of the reconstructed clock - CLK1' (Figure 3.3) Figure 3.3 to Figure 3.5 shows the timing diagram of data flowing from the input compute interface to the time block.

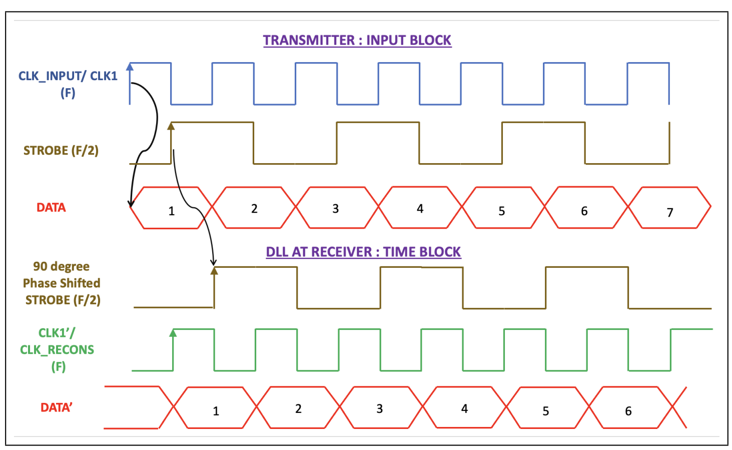


Figure 3.3: Data is being sent from the input block to the receiver along with the clock strobe, samples at the negative edge of local CLK1. At the Receiver, the DLL produces phases shifted version of the strobe to reconstruct the transmitter clock

Further at the receiver, the second stage of sampling happens either at the positive or negative edge of the local clock at the time block. The edge decision unit (Figure 2.5) determines if the next positive edge of CLK2 falls in the failure zone (FZ) of data. If that's the case as seen in Figure 3.4 negative edge of CLK2 or clock CLK2' is chosen. Thus, edge adaptability at each cycle helps avoid metastability when re-timing the data to receiving block clock.

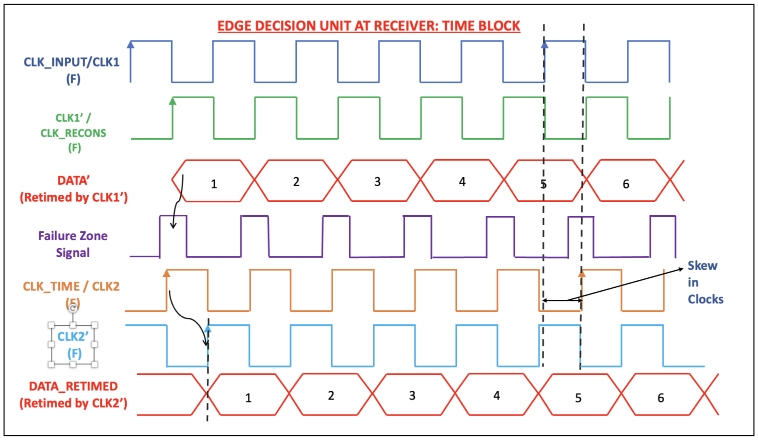


Figure 3.4: The data is first sampled at the positive edge of the reconstructed transmitter clock. CLK2(CLK$_T IME) edge falls in the failure zone signal, the data is sampled at the negative edge of CLK2$

In the last, to latch the data to the positive edge of CLK2, an additional stage of pipelining is included.

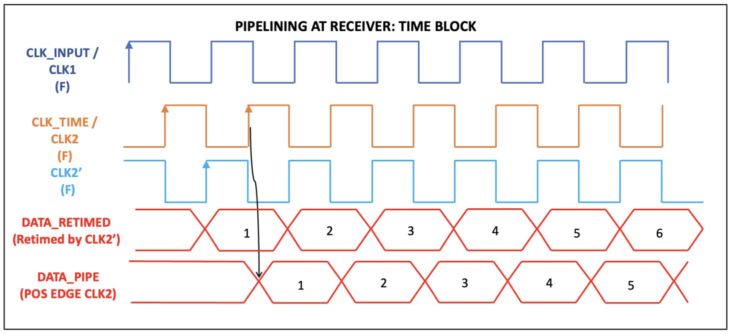


Figure 3.5: Last stage of pipelining aligns the data to the positive edge of CLK2(CLK$_T IME)$

The timing simulation shown illustrates the nature of data capture in the ARION test module with and without mesochronous synchronization. In Figure 3.6 it can be seen that when skew in the clocks is not taken care of causes setup and hold failures in the system.

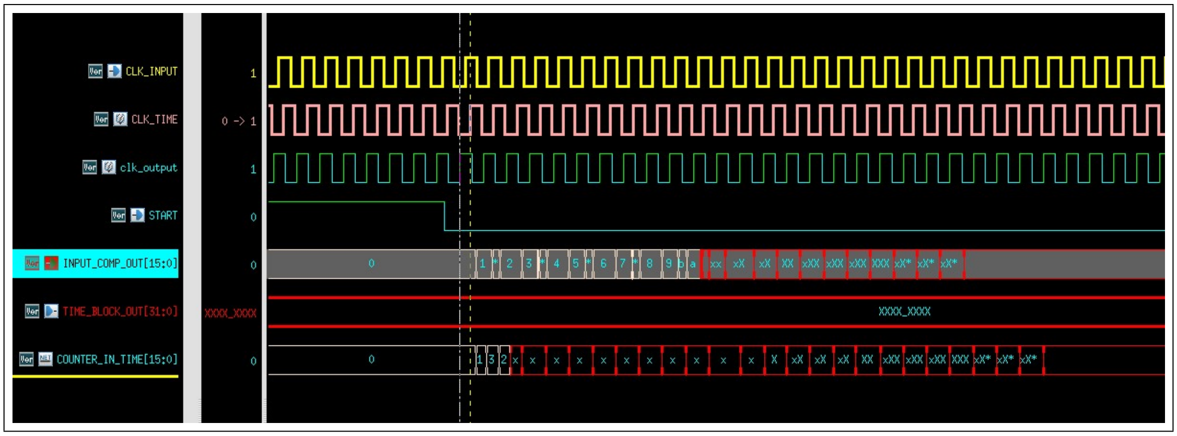


Figure 3.6: Timing Simulation on Functional Model: Errors in data propagation when clocks skew between each block are not managed.

Due to the nature of the design operating in the GHz range, an extra stage of the pipeline was added to meet timing. In Figure 3.7, the wide shot shows the time block realizing the propagation delay in the incoming signal. The re-timing of data takes 2 cycle extra for synchronization at the time block (receiving block) as it is sampled at multiple stages. A deterministic hardware latency is incurred due to the synchronization. This affects the functioning of ARION and could lead to erroneous computation as the data as the time block is simulating larger delay than programmed. Keeping this in mind, the functional model of time block accounts for this latency as seen in the zoomed-in image on the right of Figure 3.7.

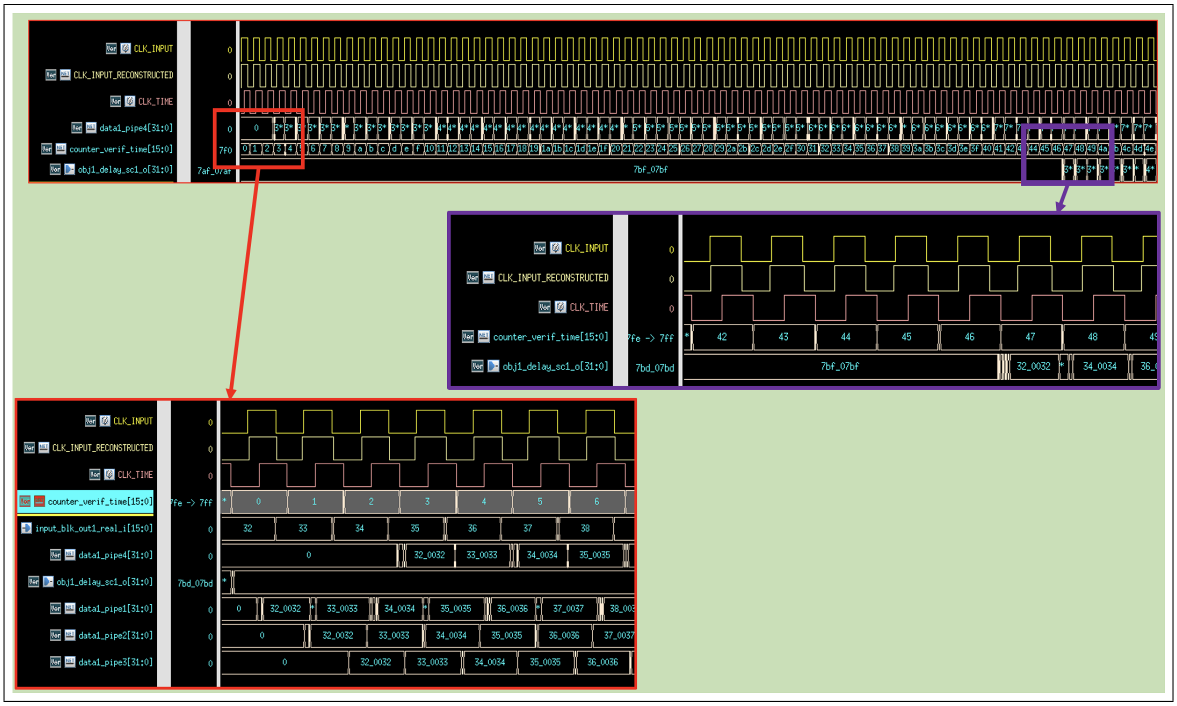


Figure 3.7: Timing Simulation on Functional Model: The propagation delay of 70 cycles is programmed in the time block. Data sent by the input block at cycle 0 is consumed by the time block at cycle 3 (as indicated by the free-running counter), and is then sent out by cycle 70, demonstrating successful mitigation of incurred hardware latency.

ARION sends out a scenario update signal going from time block to each of the blocks for scenario change every millisecond. As the clock at the time block lags as compared to clocks at the compute block, SU faces negative skew. A double flop synchronization at the receiving block is able to remedy the skew.

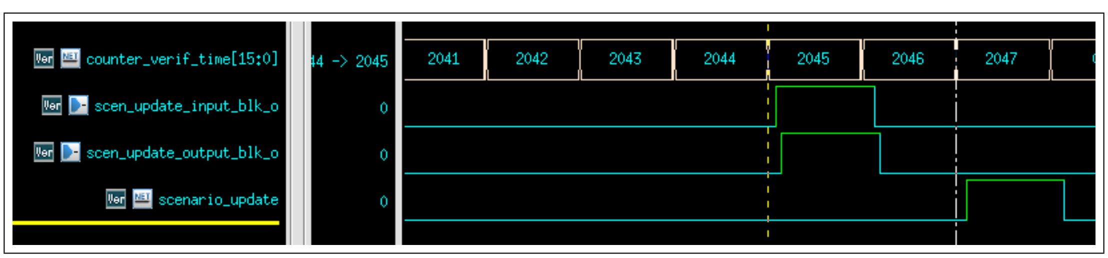


Figure 3.8: Timing Simulation on Functional Model: SU signal is asserted 2 cycles earlier for the compute blocks to ensure latching of new coefficients at the correct edge across the system

Furthermore, using Post-PnR hard macros of the ARION design was simulated, and a top-level RTL wrapper was created and subjected to synthesis after integrating the soft IP designed for the synchronization scheme on top.

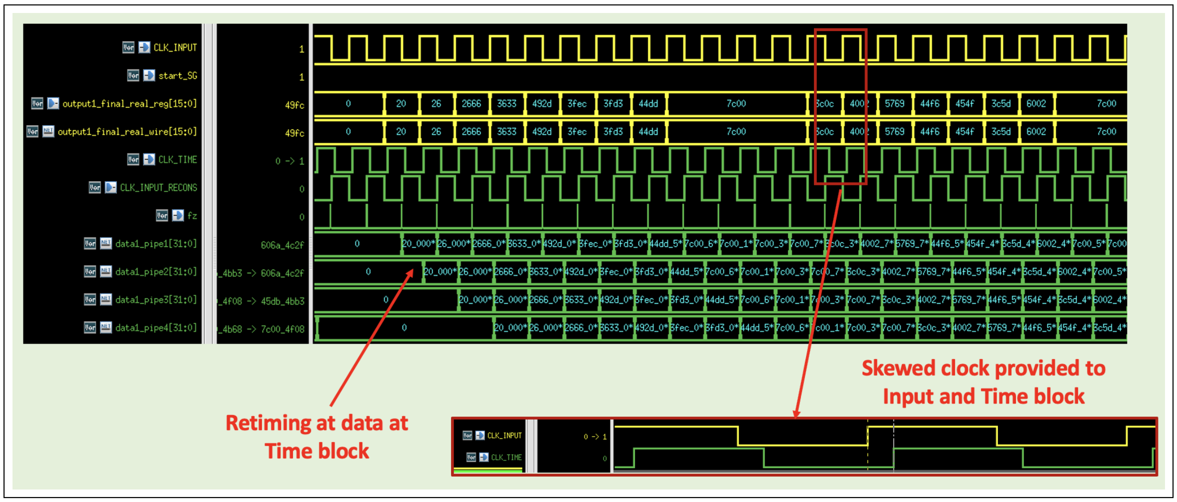


Figure 3.9: Simulation shows re-timing of data at time block after mesochronous implementation in ARION

Lastly, timing simulation results are shown in Figure 3.10 of an initial exploration of BIST proposed in Figure 2.7 to perform the post-silicon multi-cycle skew correction. A calibration phase is introduced to detect the inertial delay of each clock with respect to absolute time. Once the skew between clocks is determined, it can be externally programmed with control signals to indicate the lagging module. This clock gating hypothesis is simulated on the functional test bed of ARION.

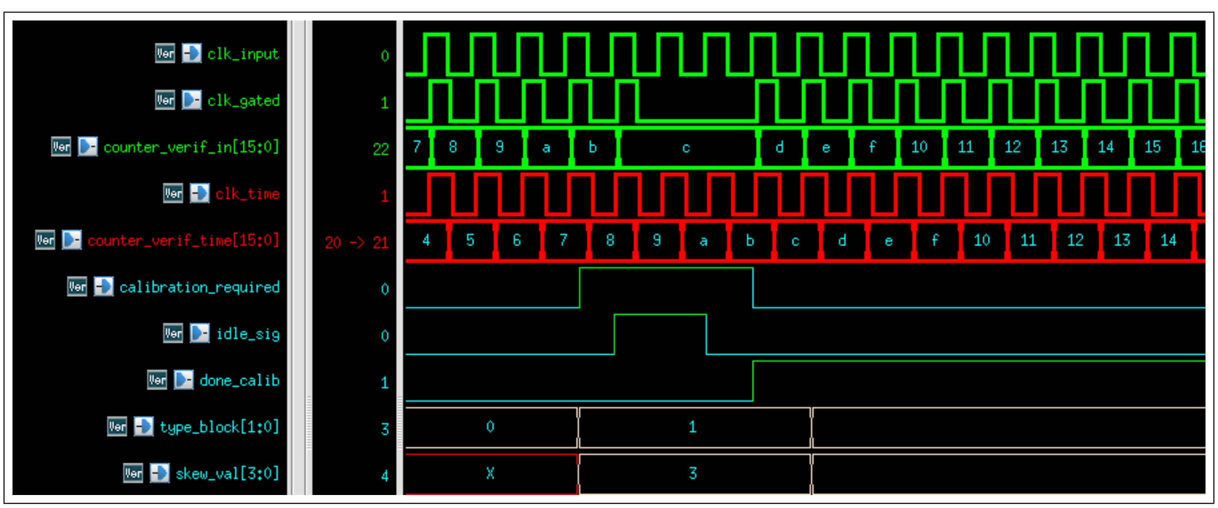


Figure 3.10: Timing simulation multi-cycle clock delay

3.4 Area and Power

This subsection provides an estimate of power and area overhead that would incur with the implementation of the synchronization scheme proposed. The calculation is done by assuming an implementation of DLL proposed in [10] and the original area from the tape-out ready prototype of the 4-object ARION system. Additionally, the post-synthesis area and power estimate of the Edge Decision Unit and mesochronous logic (sequential and combinational) are presented in the table below.

Table 3.2: Estimated Power and Area of Individual Modules

MODULE	POWER	AREA
INPUT COMPUTE	$2.8\ W$	$2\ mm^2$
OUTPUT COMPUTE	$2.8\ W$	$2\ mm^2$
TIME BLOCK	$2.1\ W$	$4.7\ mm^2$
EDGE DETECTOR	$0.2811\ mW$	$9.95\ \mu m^2$
FREQUENCY MULTIPLIER[10]	$1.5\ mW$	$0.024\ mm^2$
MESOCHRONOUS LOGIC	$16.3142\ mW$	$1.335\ \mu m^2$

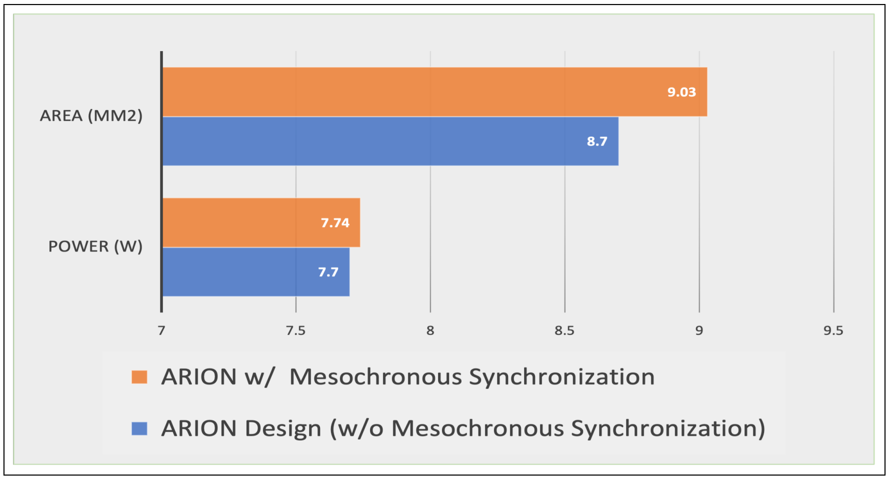


Figure 3.11: Power and Area overhead after adding mesochronous synchronization to ARION design

Considering the worst-case estimate for power and area overhead for a potential PnR design, the aforementioned estimates of synthesized design as seen in Table 3.2 is doubled. **With the strobe-based approach, a power increase of 0.6% and area overhead of 3.8% is observed.**

CHAPTER 4

DISCUSSION

In this work, the issue of clock skew in large-scale RF accelerators was investigated. As the design operates as a synchronous system with a single clock domain, the clock skew can be viewed as a phase difference and hence becomes a mesochronous clocking synchronization problem. The design specifies an external start signal as a brute force active-low SET for the whole system, ensuring that the skew experienced in local clocks is a fraction of the time period and not more than that. This in-built feature of the design supports the decision to explore the proposed design approach to minimize clock skew and ensure proper synchronization in the ARION RF accelerator.

To validate the initial proof of concept, a simplified functional model of the ARION was developed and implemented as the actual design of the accelerator is out of scope. Moreover, a less compute-intensive model of the actual design reduced the simulation run time considerably and facilitate development and debugging. The functional module was verified using standard cell timing analysis on 28nm technology at a clock frequency of 2.5GHz. The clock skew between the synchronous blocks was intentionally introduced and tested to be more than 50 percent of the clock cycle to verify the successful synchronization and achieve timing closure. Once the proof of concept was validated, mesochronous synchronization was introduced as a soft IP in the actual ARION 4-node prototype. Timing simulations were performed with SDF obtained after synthesis which does not include the interconnect delay and could be the reason for some timing failures seen in the trials with ARION design.

A brief table of a literature survey is presented in APPENDIX A. It is seen that many other mesochronous clocking implementations required a ”stall” signal or some form of handshaking, which would negatively impact system throughput. RF accelerators are known to have large-scale layouts due to the compute-intensive nature of mathematical models used to represent real-world systems. [14] proposed a dual-clock FIFO, by achieving synchronization explicitly through flow-control signals. Continuous systems like ARION cannot tolerate stall or back pressure. Taking this into consideration, a strobe-based approach for phase detection and correction for mesochronous clocking was picked. This method involves evaluating metastability at every clock edge and requires dynamic delay matching within the module. The complexity and logic-heavy design of the ARION chip further increase the area by 3.8% after strobe-based implementation. One drawback of using a phase-matching circuit like DLL is the need for delay matching between the data and DLL at every capturing interface which could make timing closure difficult. Mesochronous logic is widely used in NoCs with big processing units but most of the approaches involve a flow control mechanism that cannot be accommodated while emulating a continuously moving object.

CHAPTER 5

CONCLUSION

This research highlights the importance of clock skew management in large-scale digital accelerators for RF emulation design. ASICs have become crucial for hardware acceleration due to their faster processing than general-purpose CPUs and GPUs. Emulating RF interactions at high bandwidth requires distributed processing nodes with substantial computing capacity. To maintain global synchronization while minimizing skew and delay, a mesochronous scheme is implemented, which integrates the clock distribution with the data bus and uses phase detection for synchronization. This approach works well for custom designs as the design uses delay lines and doesn't require flow control which maintains the throughput. The re-timing of data introduces hardware latency in the system and would still demand efforts in timing closure and delay matching in large-scale digital RF emulators. This study estimates an increase in power consumption and area of 0.6 % and 3.8 % respectively.

REFERENCES

[1] J. Seo *et al.*, “A configurable architecture for efficient sparse fir computation in real-time radio frequency systems,” in *2022 IEEE/MTT-S International Microwave Symposium - IMS 2022*, 2022, pp. 998–1001.

[2] P. Teehan, M. Greenstreet, and G. Lemieux, “A survey and taxonomy of gals design styles,” *IEEE Design Test of Computers*, vol. 24, no. 5, pp. 418–428, 2007.

[3] K. Han, A. B. Kahng, and J. Li, “Optimal generalized h-tree topology and buffering for high-performance and low-power clock distribution,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 2, pp. 478–491, 2018.

[4] F. Mu and C. Svensson, “Self-tested self-synchronization circuit for mesochronous clocking,” *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 48, no. 2, pp. 129–140, 2001.

[5] J. Matai, P. Meng, L. Wu, B. Weals, and R. Kastner, “Designing a hardware in the loop wireless digital channel emulator for software defined radio,” in *2012 International Conference on Field-Programmable Technology*, 2012, pp. 206–214.

[6] M. Mukherjee and et al., “A high-performance computing architecture for real-time digital emulation of rf interactions,” in *2023 IEEE Radar Conference (RadarConf23)*, 2023, pp. 1–6.

[7] J. Ax, N. Kucza, M. Vohrmann, T. Jungeblut, M. Porrmann, and U. Rückert, “Comparing synchronous, mesochronous and asynchronous nocs for gals based mpsocs,” in *2017 IEEE 11th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC)*, 2017, pp. 45–51.

[8] M. Mukherjee and et al., “A near-memory accelerator for real-time emulation of rf interactions,” in *GOMACTech 23*, 2023, pp. 1–6.

[9] B. Mesgarzadeh, C. Svensson, and A. Alvandpour, “A new mesochronous clocking scheme for synchronization in soc,” in *2004 IEEE International Symposium on Circuits and Systems (IEEE Cat. No.04CH37512)*, vol. 2, 2004, pp. II–605.

[10] H. Kim, Y. Kim, T. Kim, H.-J. Ko, and S. Cho, “A 2.4-ghz 1.5-mw digital multiplying delay-locked loop using pulsewidth comparator and double injection technique,” *IEEE Journal of Solid-State Circuits*, vol. 52, no. 11, pp. 2934–2946, 2017.

[11] H. Park, J. Sim, Y. Choi, J. Choi, Y. Kwon, and C. Kim, “A 2.4–8 ghz phase rotator delay-locked loop using cascading structure for direct input–output phase detec-

tion," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 69, no. 3, pp. 794–798, 2022.

[12] K. Hoang, Y. Deval, and F. Rivet, "An ultra-low power dual-ring factorial delay locked loop in 28nm fd-soi technology," in *2016 13th IEEE International Conference on Solid-State and Integrated Circuit Technology (ICSICT)*, 2016, pp. 557–559.

[13] S. Yang, J. Yin, P.-I. Mak, and R. P. Martins, "A 0.0056-mm2 249-db-fom all-digital mdll using a block-sharing offset-free frequency-tracking loop and dual multiplexed-ring vcos," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 1, pp. 88–98, 2019.

[14] D. e. a. Konstantinou, "The mesochronous dual-clock fifo buffer," 1, vol. 28, 2020, pp. 302–306.

[15] R. W. Apperson, Z. Yu, M. J. Meeuwsen, T. Mohsenin, and B. M. Baas, "A scalable dual-clock fifo for data transfers between arbitrary and haltable clock domains," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 15, no. 10, pp. 1125–1134, 2007.

[16] D. Verbitsky, R. (Dobkin, R. Ginosar, and S. Beer, "Starsync: An extendable standard-cell mesochronous synchronizer," *Integration*, vol. 47, no. 2, pp. 250–260, 2014.

[17] D. Ludovici, A. Strano, D. Bertozzi, L. Benini, and G. N. Gaydadjiev, "Comparing tightly and loosely coupled mesochronous synchronizers in a noc switch architecture," in *2009 3rd ACM/IEEE International Symposium on Networks-on-Chip*, 2009, pp. 244–249.

Appendices

APPENDIX A

Table 1: Literature Survey: Existing Synchronization Schemes

Prior Work	**Description**	**Advantages**	**Disadvantages**
FIFO for Asynchronous System [15]	FIFO for multiple clock domain	This approach can handle multicycle latency	Handshaking mechanism Large latency overhead Could decrease throughput
Starsync [16]	Synchronization and buffering happen exclusively	Multi-cycle link delay can be mitigated	Large area overhead & FIFO underutilization
Tightly-coupled Mesochronous Implementation [17]	Synchronization & buffering takes place in single structure	No FIFO used for synchronization hence less area utilization	Can only fix single -cycle timing skew Flow Control Implementation
Mesochronous Dual-Clock Buffer [14]	Synchronization implicitly through explicitly flow control mechanism	Area efficient	Push/Full flow control multi-cycle link delay handled through pipelining by strobe signal