UNIVERSITY OF CALIFORNIA, SAN DIEGO

ArCuS: An Architecture for ASIC Cloud based Servers

A thesis submitted in partial satisfaction of the
requirements for the degree
Master of Science

in

Computer Science (Computer Engineering)

by

Pulkit Bhatnagar

Committee in charge:

Professor Michael Bedford Taylor, Chair
Professor Chung-Kuan Cheng
Professor George M. Porter

2017

The thesis of Pulkit Bhatnagar is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

Chair

University of California, San Diego

2017

DEDICATION

To my parents

EPIGRAPH

There are 10 kinds of people in this world:
those who can count binary, and those who can't.

–Anonymous

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to take this opportunity to thank my advisor Professor Michael Taylor for his constant support and advice throughout the course of my stay at University of California, San Diego. I acknowledge him for giving me an opportunity to be involved in multiple projects in the group. His guidance has been invaluable in pursuing this thesis. His knowledge has always motivated me to learn more and helped me to sharpen my problem solving skills.

Credit is also due to my thesis committee members, Professor Chung-Kuan Cheng and Professor George M. Porter for taking out time to review this work and provide constructive feedback and comments.

This work would not be possible without the support and discussions with the ASIC Cloud team including Moein, Luis and Harveen of the Bespoke Silicon Group (BSG) at UCSD. Special thanks to Scott for providing CAD and back-end support during the implementation phase and to Chun for helping setup the design flow.

I am grateful to UCSD, all the professors who have taught me and the welcoming staff and students of the Computer Science and Engineering department. I was very unsure about moving to the states and pursuing graduate studies, but when I got an opportunity at UCSD I knew I had to take the chance. Looking back, it now seems to be an obvious choice. I thank UCSD and all my friends in this journey who made this a memorable learning experience. I will cherish this for a very long time to come. Finally, I dedicate this thesis to my mummy and papa for their love and support. I owe all my accomplishments to them.

VITA

2008-2012	Bachelor of Technology (B.Tech) Electronics & Communication Engineering, Institute of Technology & Management, Gurgaon (Maharshi Dayanad University, Rohtak, India)
2012-2015	Senior Design Engineer, STMicroelectronics, Greater Noida, India
2015	ASIC Digital Design Engineer-II, Synopsys Inc., Noida, India
2016	Engineering Intern, Qualcomm Inc., San Diego, CA, USA
2015-2017	Master of Science (MS) Computer Science (Computer Engineering), University of California, San Diego, USA

ABSTRACT OF THE THESIS

ArCuS: An Architecture for ASIC Cloud based Servers

by

Pulkit Bhatnagar

Master of Science in Computer Science (Computer Engineering)

University of California, San Diego, 2017

Professor Michael Bedford Taylor, Chair

As we approach the end of Moore's law, the research focus has shifted towards developing hardware accelerator based designs to achieve higher performance with lower power requirements. The key objective while developing a server architecture is to minimize the Total Cost of Ownership (TCO) while improving both performance and energy efficiency. Thus, the trend in such planetary scale applications has been to move from CPUs, GPUs and FPGAs towards arrays of application specific accelerators, known as ASIC Clouds. In this work we present ArCuS, a hardware infrastructure for developing **Ar**chitectures for ASIC **C**lo**u**d based **S**ervers. We layout the architectural specifications, RTL description and physical implementation details for the ASIC which aims to serve as a reference and enable rapid prototyping. We

design and characterize the proposed architecture across technology nodes and evaluate various metrics for accelerating bitcoin mining application. Finally, we present the cost-performance trade-offs necessary for determining a cost-optimal ASIC Cloud configuration.

Chapter 1

Introduction

1.1 Motivation

As Moore's Law approaches its limits and the Dark Silicon Apocalypse [3] prevails upon Dennard's scaling [4], the focus on achieving performance and energy efficiency has shifted towards hardware acceleration and specialization [5]. Building heterogeneous architectures have become standard practice to achieve higher performance while maintaining the device power budgets. Modern high-end mobile SoCs like Qualcomm Snapdragon [6][7] and Apple A10 Fusion utilize a co-processor based design integrating CPU, GPU and multiple DSP cores on a single chip. However, it is not always economically viable to integrate accelerators on mobile SoCs given the increased IP cost, area and power constraints.

We are starting to see a similar trend in datacenter architectures. Packing more and more CPU cores on the servers have lead to increased concerns over cooling and maintenance of the warehouses. To achieve the desired performance improvement GPUs (Graphic Processor Units) and FPGAs (Field Programmable Gate Arrays) have started to gain popularity for search engines and machine learning applications. Companies like Google [8] and Baidu [9] have deployed GPUs for neural networks, while Microsoft (Bing) [10] has promoted FPGA based customizable servers as also seen in banking [11] and High Frequency Trading [12] applications.

Application Specific Integrated Circuits (ASICs) have proven to provide better energy efficiency with reduced area. ASICs are fully customizable logic blocks suited

for highly specialized applications giving flexibility in memory configurations, IOs and packaging options. Magaki et. al. [1] introduced **ASIC Clouds** which are purpose built datacenters with arrays of ASIC accelerators aimed at optimizing the TCO (Total Cost of Ownership) of servers. This accelerator based datacenter specialization approach specially makes sense for high volume recurring workloads. Applications such as Youtube video transcoding, Facebook face recognition, Apple's Siri, Google Home and Amazon Alexa process hundreds and thousands of requests per second. These are essentially repetitive computations over millions of users, making custom server designs viable. ASIC clouds provide the flexibility to build custom PCB (Printed Circuit Boards), cooling systems, Power Delivery Networks (PDN), DRAM interfaces and minimalistic IOs based on the application requirements. Companies like Google [13], Amazon, Facebook and Intel [14] have identified this and are developing custom designs for multiple applications. Bitcoin Miner ASICs [15] have been around for sometime now. These can be found deployed today as 130nm down-to 16nm AISCs. Miners are mostly full-custom build and proprietary.

1.2 Thesis Contributions

Today, most of the server-end designs are abstract and company proprietary. This thesis is an effort to develop and realize an infrastructure based on the ASIC cloud [1] model. We layout the functional specifications of the communication among the ASICs at the server level, develop fully-synthesizable RTL model and implement it on various technology nodes ranging from 180nm down-to 16nm for a comparative analysis. All this is done with scalability and flexibility in mind targeting need of minimal changes for each application. We leverage and build upon the open source IP database developed by Bespoke Silicon Group [16] at University of California, San Diego. We hope that this would provide an easily accessible starting point for the research community for developing custom cloud infrastructures. We finally comment on the performance and the cost effectiveness of our approach.

1.3 Organization of the Thesis

The thesis is organized as follows: Chapter 2 discusses related work in the field of custom server designs and builds a background of the ASIC Cloud architecture. In chapter 3 we define the specifications and present the architectural view of ArCuS at the server and ASIC level. Chapter 4 provides details of the ASIC micro-architecture describing the building blocks (IP cores) and the implementation methodology. Chapter 5 summarizes the results and we finally conclude the thesis with a discussion of future efforts in Chapter 6.

Chapter 2

Background

This section gives a high level view of state-of-the-art datacenter and accelerator research and an overview of the ASIC cloud architecture.

2.1 Related Work

The move towards specialization is thought of as one of the most promising directions for future of computer architecture research to combat the effects of dark-silicon [3][17]. Numerous specialized processors have been proposed in literature for cryptography, signal processing, natural language processing, physical and scientific computations, neural networks, encoding and graphics. Authors in [5][18][19] suggests use of automatically synthesized accelerator cores that trades-off area for energy efficiency in processors. Shao et. al. [20] gives a taxonomy of various accelerators in literature and develops an infrastructure for accelerator research.

As computation moves to the cloud, datacenter have become a focus for the research community. Borosso et. al [21] presents a compendium of datacenter design and operation. The era of using general purpose CPUs and even multiple commodity class PCs for search engines as used by Google [22] a decade ago has come to an end. Low power Embedded processors have been studied for warehouse-computing [23]. Special Database processors like Q100 [24] and dataflow computers [25] have also been proposed. Application of FPGAs as a replacement of CPUs and GPUs in datacenters have also been studied. Microsoft's Catapult [10] examines a reconfigurable FPGA based

accelerator for ranking algorithms. Several others have applied FPGAs for Memcached [26][27], Clustering [28] and Web-searches.

Some of the older ASIC supercomputers span from the GF11 [29] for physics simulations to the Anton [30] deployed for molecular dynamic simulations. Google has reported development of Tensor Processing Unit (TPU) [13] for machine learning applications. Amazon Web Services unveiled Annapurna ASIC [31] for faster networking. Magaki et. al. [1] proposed an ASIC cloud architecture for datacenters focusing on recurring workloads. Khazraee et. al. [2] gives a rigorous analysis on technology node selection for realizing the ASIC cloud system for optimal NRE. We take this concept forward and present a hardware infrastructure for realizing ASIC clouds based datacenter.

2.2 ASIC Clouds Overview

This sub-section is a primer of the high level architecture of an ASIC Cloud, Figure 2.1 as proposed in [1]. The machine room at the datacenter has multiple 42U or 48U racks. Each rack is connected through high speed Ethernet to the external network. The racks have a central Ethernet switch (Top-of-Rack switch) which connects each of the server blades in the rack. The server has a power supply unit (PSU) and cooling system. The off-PCB interface (10G Ethernet, PCI-e or other point-to-point links) delivers data to the server controller (FPGA or a micro-controller) which has access to an array of specialized ASICs. Each of the ASIC consists of multiple accelerators called as the Replicated Compute Accelerators (RCA for brevity). The off-chip interface connects to the ASIC router and the controller which distributes the workload among the available RCAs through the internal network. Each ASIC may have on-chip clock generator or PLL, thermal sensor and power grid. In case of memory intensive applications, the DRAMs could be shared among RCAs with a memory controller on each ASIC.

In this work we focus on the server level configuration and the protocol for data transfers. Then we look into a scalable ASIC design specifically for logic intensive processing like bitcoin mining. The same idea could be easily extended to other types of applications.

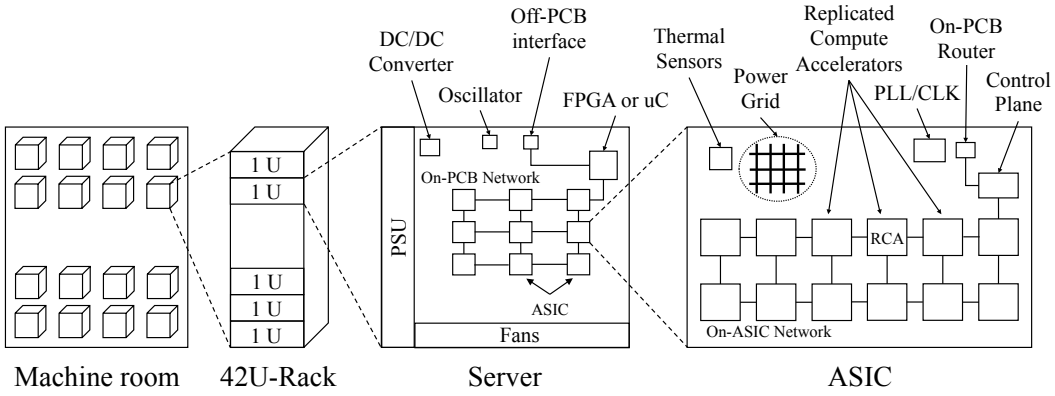

Figure 2.1: High-Level Abstract Architecture of an ASIC Cloud [1].

2.3 Bitcoin Mining

Bitcoin (BTC or ₿) is one of the most widely used cryptocurrency system. The transactions and distribution of new bitcoins are controlled by decentralized computations known as *mining*. It is a computationally intensive operation that takes a 512bit *block* and performs a series of SHA-256 operations on it. The result is compared to some reference value to check if it has a predefined number of starting zeroes. Every such attempt is called a *hash*. Billions of hashes reuse the same block and performs this operation. The number of hashes that a machine performs is termed as the *hashrate*, generally reported in gigahash per second (GH/s). Each SHA-256 round takes up 64-cycles making each hash operation 128-cycles long. This is a computationally intensive and repetitive workload making it a good candidate for parallelism. Taylor [32] examines the transition of bitcoin miner hardware from FPGA to GPUs and ASICs. Today, Cloud mining has become de facto standard. We take up the bitcoin application as our base-case for ASIC cloud acceleration. Note that the Bitcoin application requires no inter-chip communication and has a very high power density being logic intensive with little on-chip memory.

Chapter 3

The ArCuS Framework

In this section we present the ArCuS framework, the complete specifications for the hardware needed to develop the ASIC Cloud infrastructure. The focus of this work is limited to the server level design. We touch upon the interface between the servers at the rack level but do not delve into the networking part as it is outside the scope of this work. Our goal is to:

- Propose an on-PCB network topology and communication protocol.
- Define the interfaces between the controller/FPGA and the ASICs.
- Establish a sequence for reset and system boot-up.
- Specify the data transfer (request and reply) packet formats.
- Govern the distribution/scheduling of workload amongst the ASICs and the RCAs.

3.1 Server Architecture

The server is a part of the 42U rack with multiple ASICs which connects to the outside network via a high-speed off-PCB interface like 1/10/40G Ethernet as shown in Figure 2.1. There are several factors to consider for the choice of the server level controller. It directly depends upon the rack-level architecture, which must be optimized for cost and reliability. At the server, in addition to providing the Ethernet capability, it

is essential to have reliable packet delivery with special mechanisms at the network (IP) and transport layer (TCP, UDP). One option to realize this is to have a low cost CPU that performs the required networking in software taking care of the reliability of data transfers at the TCP (Transmission Control Protocol) level and handles the scheduling at the server level. This type of system saves the hassle of designing the TCP (or other reliable protocols) stack in hardware and is easy to use. However, this ends up increasing the total cost of the cloud with each server costing around $400. Some newer CPU chips (such as the Intel Atom C3000) have been found to provide such capabilities at a much lower cost. The other option is to have a reprogrammable FPGA along with the TCP offload engine (TOE) core to take care of the network reliability. The Ethernet MAC IP is easily available with most FPGAs as an open core. The controller and scheduler logic can be easily programmed in the logic blocks. This type of system is a cheaper solution and can be realized by spending around $250 per server considering amortization of the TOE IP cost over multiple ASICs. However, all this comes with the added complexity of the hardware. Lastly, one intermediate approach is to have distributed networking inside the rack. At the rack-level we can implement a fully-reliable network framework using dedicated CPUs while at the server level we can then use a light weight lower-reliability protocol such as UDP (User Datagram Protocol). The UDP layer is easier to design for the FPGA boards while also supporting the required bandwidth. The cost of the server CPUs is amortized across the rack and hence turns out to be a reasonable and cost-effective solution.

In the following discussion, we assume that the FPGA is connected to the external network and is the entry point receiving the work packets in the appropriate format after processing. The FPGA serves both as the off-PCB interface with the high speed, high bandwidth Ethernet support, and as the controller for monitoring the task distribution among the available ASICs. The FPGA controller and the AISCs communicate using the on-PCB network, discussed next. Each ASIC is an array of identical Accelerators or RCAs.

3.1.1 Network Topology

Various network configurations were considered for the on-PCB interconnect. Although a point-to-point network would provide minimum latency, it is not scalable as the number of ASICs per server increases. Assume a single-ended 32-bit data channel with clock and valid, 3-bit Test bus and 4-bit external reference clocks and control bits per ASIC. Figure 3.1 shows as the number of ASICs increases, the number of pins at the controller in a star network becomes unreasonably high. The situation is even worse if we consider differential signalling for data transfer. The number of IOs on the FPGA side could be increased by interfacing a low-end, low-cost FPGA with the main FPGA board (eg: Xilinx Spartan-6 series can provide IOs as low as $0.2/differential pair). However, there are overheads of interfacing the two FPGAs but this could be a consideration for high performance latency sensitive applications. For our application, to limit the number of IOs on the FPGA, we choose a *Daisy Chained* network for the ASICs. Figure 3.2 shows a high level schematic of the server. Typically, each column can have up to 32 ASICs or 63 ASICs per FPGA controller. The links are unidirectional and the details of the communication protocol are provided in later sections. In this configuration, Figure 3.1 shows only a marginal increase in pin count due to the test and clock signals overhead per ASIC.

The communication link between the FPGA and the ASIC is based on a DDR (Double Data Rate) source synchronous channel, *bsg_comm_link*. Here a light weight version, *bsg_comm_link_lite* is implemented which provides a simplex rather than full-duplex link as needed for this network. The link has various channel configurations and is fully customizable. In this application we have a 32-bit data channel sent along with source-synchronous clock and valid. The detailed description of the *bsg_comm_link_lite* module is provided in chapter 4.

3.1.2 FPGA Controller

The FPGA controller provides reconfigurability and easy access to IP cores provided by the vendors for quick development of our application. There are a large variety of FPGAs in the market and the decision for board selection was based on the design

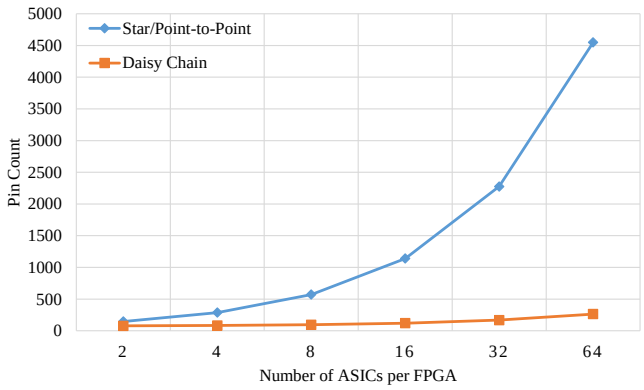

Figure 3.1: **Pin Count vs. ASICs per FPGA**: This shows the comparison of the pin requirements of the Star and Daisy-Chain network topologies for the FPGA. It is observed that the IO demands for a star network increases exponentially as the ASICs per server increases.

requirements while achieving minimal cost. We considered Xilinx series-7 [33] family for this analysis. The board must support at least 10G Ethernet for a high speed, high bandwidth connectivity to the external network. The network topology and the interface discussed earlier requires at least 264 single ended IOs. The type of IOs must be taken into consideration. Xilinx boards provides High Range (HR) IOs supporting up to 3.3V and High Performance (HP) IOs for 1.8V supply. The Gate Count, logic blocks and Block RAM must be sufficient to pack the networking and controller logic. Also, there must be support for high speed memory interface like DDR3/DDR4 for applications requiring use of DRAM. Table 3.1 shows a comparison of some of the candidate Xilinx boards from each of the series-7 family meeting the specifications. Artix-7 is identified as a good fit for the application addressing all the requirements (except 10G) at a low cost. The 10G Ethernet capability can be added by using the XAUI Attachment Unit Interface with external XAUI to SFI (SerDes framer Interface) transceiver with an additional cost of only $20-$30 per server.

Table 3.1: Xilinx Series-7 FPGA Comparison

Family	**Part**	**UserIO (Banks)**	**HS XCVR**	**10G**	**Price** [34]
Artix-7	XC7A75T	300 (6)	8 (6.6Gbps)	No	$134.26
Kintex-7	XC7K160T	400 (8)	8 (12.5Gbps)	Yes	$310.70
Virtix-7	XC7VX330T	600 (12)	20 (13.1Gbps)	Yes	$2,542.41

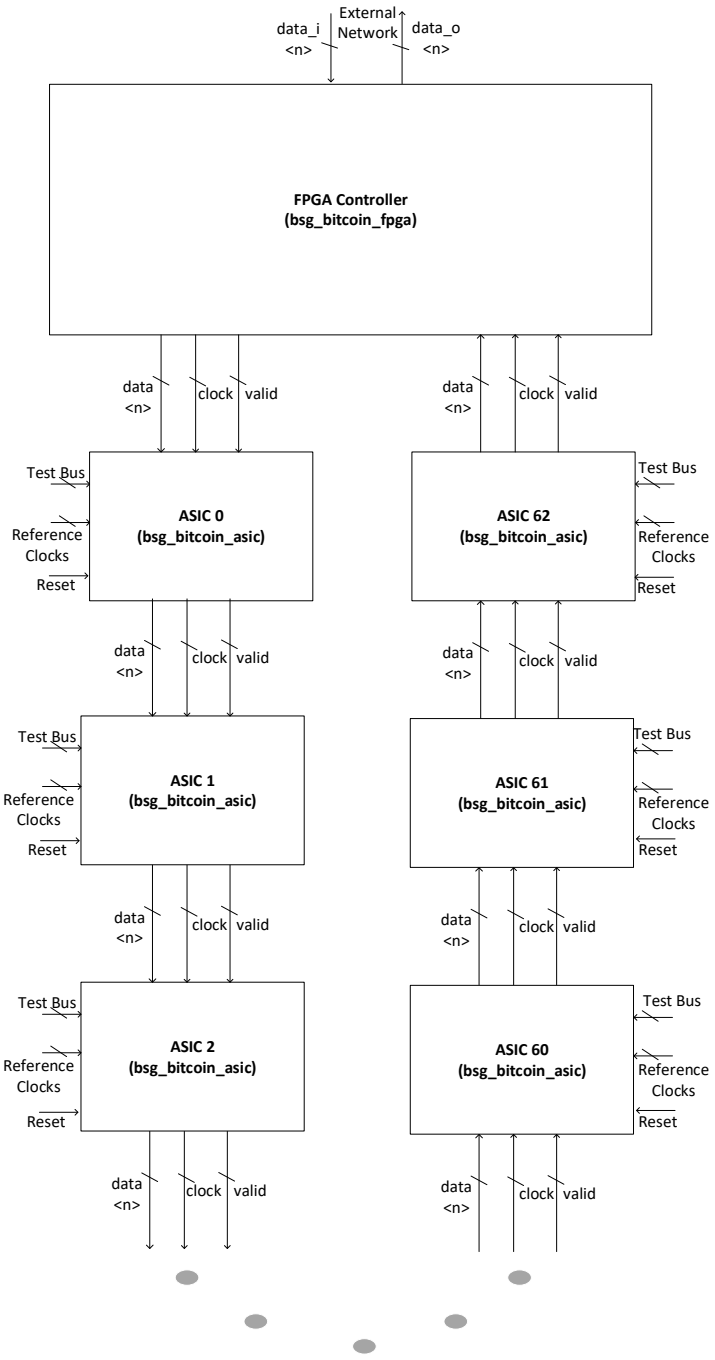

Figure 3.2: **Network Topology at Server Level**: FPGA controller is the gateway to the data received from the external network. The on-PCB network connects the ASICs in a *Daisy-Chain* topology to achieve reduced pin count at the FPGA controller while still maintaining reasonable latency.

3.1.3 ASICs

The ASICs consists of up to maximum of 512 RCA cores. In this work we consider the case of Bitcoin Miner to develop the RCA which is logic intensive and does not require external DRAM accesses. The same architecture could be extended for applications such as litecoin, while video transcoding and neural network applications could be targeted by appropriate memory interfacing with the ASICs and the FPGA controller. Each ASIC has one input and one output interface which is daisy chained to its neighbour as shown in Figure 3.2. The Micro-architectural details of the ASIC are discussed in chapter 4.

3.2 Specifications and Packet Formats

We now define some of the basic functional specifications of our system including the reset, boot-up, initialization, scheduling and the packet formats over the network.

3.2.1 Reset and Boot-up

The FPGA controller can issue an asynchronous reset to all the ASICs. As the communication link is implemented as a unidirectional channel, the need for calibration is avoided. However, sufficient time must be given for the communication links to set up and initialize before driving any data. On ASIC reset, each of the ASIC's internal clock generator must be auto configured to a predefined frequency (lowest frequency). The clock generator frequency can be configured later using *bsg_tag* after reset sequence is complete. The *bsg_tag* is a test interface similar to JTAG that we use for system initialization. To setup the IDs of each ASIC, the FPGA sends streams of bits to each ASIC *bsg_tag* master. The ASIC interprets the set_id command and sets its ID. The FPGA controller notes the number of ASICs available in the chain. Once the ASIC IDs are setup, we can send the clock configuration signals over *bsg_tag* to set the internal clock generator frequencies. Details of these modules are provided in subsequent sections.

3.2.2 Scheduling

The scheduling and distribution of workload across the ASICs and the accelerators (RCA) is partly handled by the controller. To avoid the large memory requirements at the FPGA, this task is distributed rather than centralized. The controller/FPGA maintains an ASIC free list count in memory. It keeps track of the number (only count) of available accelerators (RCAs) within each ASIC rather than which RCA in which ASIC is available. On reset, the counters are set to all ones meaning all RCA units are available. This reduces the memory requirements at the FPGA drastically. Considering maximum 512 RCAs per ASIC, there is only one 9-bit (log_2512) counter per ASIC. For 63 ASICs that evaluates to a requirement to record only 9*63 = 567-bits or approximately **70 bytes/FPGA** in comparison to about 8kB if a centralized approach was used, Figure 3.3(a). Each ASIC is now responsible to keep track of which specific RCA is loaded and which ones are available. This reduces unnecessary computations in the central controller. The ASIC controllers maintains a 1-bit/RCA list, requiring a maximum of 512-bits or **64 bytes/ASIC**, Figure 3.3(b). The same distributed technique propagates to the rack level scheduler and this provides scalability in the design.

Similar approach is taken for designing the buffers for storing the responses of the RCAs. The ASIC controller stores each RCAs response in a dedicated memory till the resources are available to forward the results to the FPGA. This distributed result buffering again reduces the memory requirements at the FPGA. We can now maintain a simple buffer for each of the ASIC in the chain which could be calculated by the response time of each task and number of hops to reach the FPGA controller. The free list is also updated as the responses are received at the FPGA.

3.2.3 Packet Formats

The standard packet length for on-PCB communication between FPGA and ASICs is set to 80-bits. Each packet contains 64 bits of data and 16 bits of address and control information. Two type of packets are defined, one originating from the FPGA controller with the requested task and the other is the ASIC response, which is the result of computation returned back to the FPGA.

ASIC FreeList	FPGA Controller (Bank n)			
	ASIC 0	ASIC 1		ASIC 62
# RCA Units Available	1_1111_1101	1_1111_1111	…	1_1111_1111

$\log_2 512*63 \approx 70$ bytes

(a)

RCA FreeList	ASIC 0				
	RCA 0	RCA 1	RCA 2		RCA 512
RCA Available	0	0	1	…	1

$1*512 \approx 64$ bytes

(b)

Figure 3.3: **Scheduler Bookkeeping**: (a)The FPGA controller maintains counts of the available RCAs in each ASIC. ASIC0 has 510 RCA available. (b)The ASIC controller maintains a 1bit free list for each of the RCAs. RCA0 and RCA1 are busy while others are available in ASIC0.

Request Packet: The request packet fields are described in Table 3.2. Multiple such packets could be required for a single transaction depending upon the application requirements of the accelerator (RCA) inputs. Figure 3.4 shows the packet format.

Reply Packet: The reply packet is very similar to the request packet except that the destination address is always fixed at (111111) to denote that the packet is to be routed to the FPGA, Table 3.3. The six bits ([72:67]) are used to store the source ASIC ID so that the FPGA can keep track of which task has finished and take appropriate actions. Figure 3.5 shows the packet format.

The packet formats internal to the ASIC network are detailed in the next chapter where we explore the micro-architecture and various building blocks of the ASIC.

Table 3.2: Request Packet Description

Field	# Bits	Description
Control/Reserved	1	The MSB is reserved for 1bit control.
Destination Address	6	Destination ASIC address. Maximum 63 ASICs could be addressed in a single chain. (One address is reserved for the FPGA)
Command & Address	9	Configuring and controlling of individual ASICs.
Data	64	Input data from the FPGA to the ASIC.

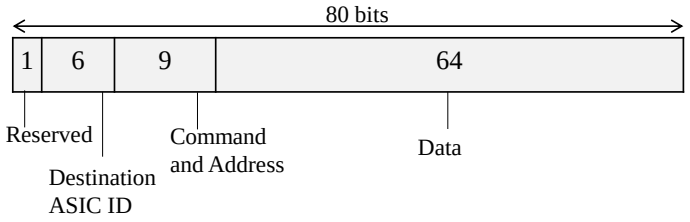

Figure 3.4: Request Packet Format

Table 3.3: Reply Packet Description

Field	# Bits	Description
Control/Reserved	1	The MSB is reserved for 1bit control.
Destination Address	6	Destination is fixed to FPGA address (111111).
Source Address	6	Source ASIC address.
Opcode	3	Reserved for future use.
Data	64	Output data from the ASIC to FPGA.

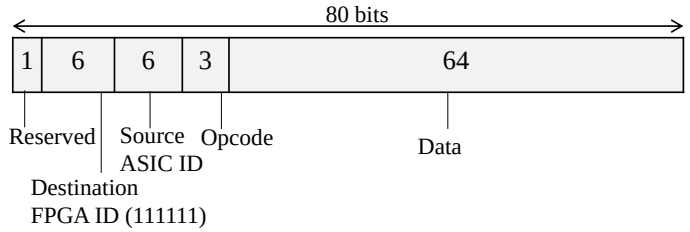

Figure 3.5: Reply Packet Format

Chapter 4

Microarchitecture and Design

Based on the defined specifications for the ASIC cloud server (ArCuS) in previous sections, we provide the microarchitectural details of the ASIC. We re-use and augment an open source IP infrastructure developed by Bespoke Silicon Group [16] at University of California, San Diego. The repository provides a RTL library of general purpose IP cores. The naming convention of the modules follow the group's prefix convention (*bsg_<module_name>*). Finally, we comment on selecting technology nodes for implementation of the ASIC.

4.1 ASIC Microarchitecture

Figure 4.1 shows a block level view of the ASIC (*bsg_bitcoin_asic*). Although this architecture is extensible to a wide variety of applications, this particular example considers bitcoin miner application. Each ASIC has a single-ended 32-bit data input and output interface (*bsg_comm_link_lite*) with per channel clock and valid. The ASIC has independent internal clock generators for IO and Core clock (*bsg_clk_gen*). A test interface (*bsg_tag*) similar to JTAG has been provided for configuration of the ASIC. The controller (*bsg_asic_controller*) and the routing logic (*bsg_mesh_router* and *bsg_fsb*) is responsible to schedule and distribute work to the available RCAs (*bsg_bitcoin_miner*). The system is developed with an intention of scalability and ease of re-use without the need of any vendor licensing except for synthesis, where foundry library must be provided. This makes our system ideal for exploration and use in the research community.

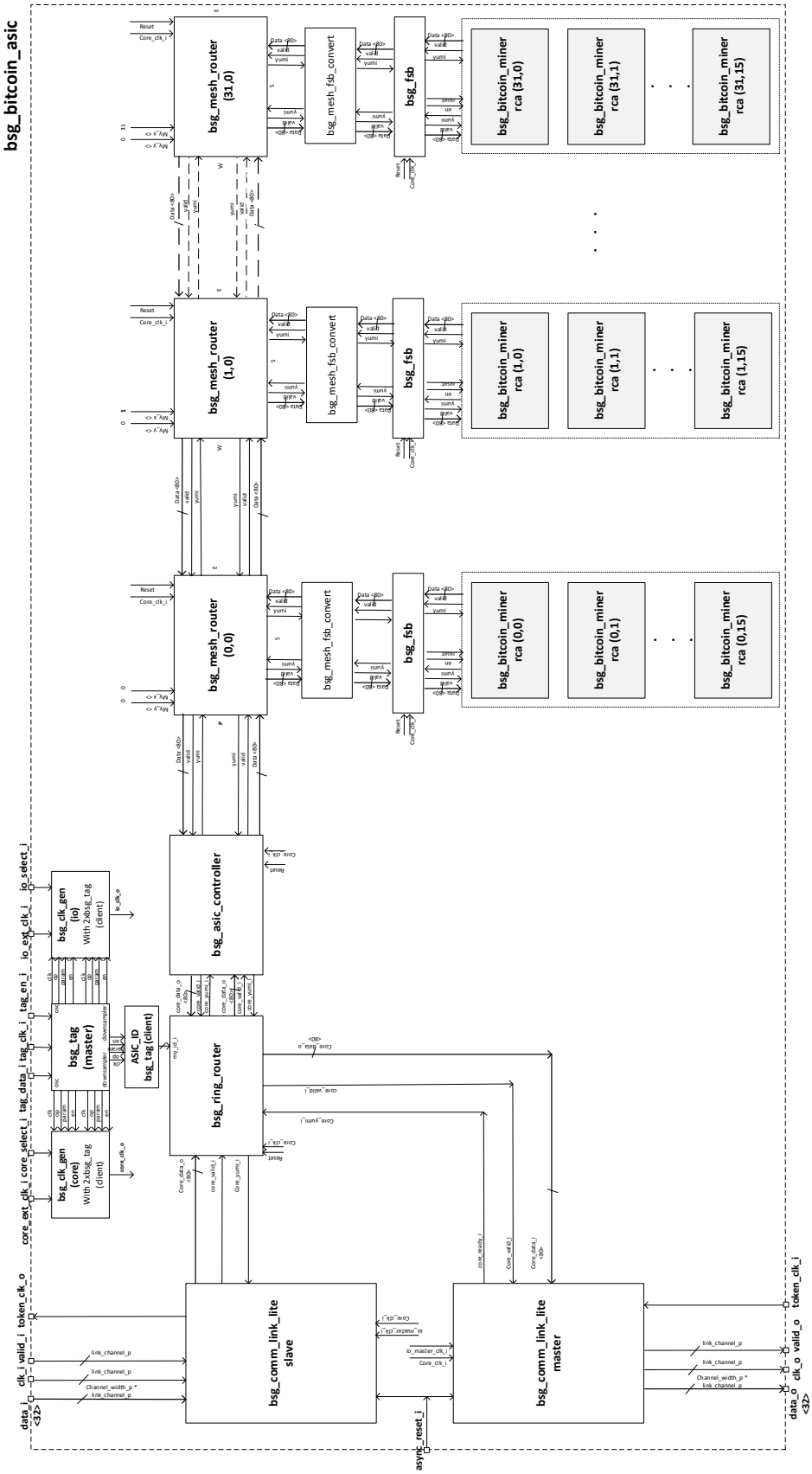

Figure 4.1: **High Level ASIC Microarchitecture:** Shows the main modules instantiated inside the ASIC. The *bsg_comm_link_lite* is the ASIC interface connecting to the internal router and controller. The network contains lanes of RCAs each connected as a mesh to the *bsg_fsb* and *bsg_mesh_router* nodes.

4.1.1 On-PCB Interface

As introduced in chapter 3, the on-PCB network communicates via a light weight communication link *bsg_comm_link_lite*. It is an unidirectional implementation of the full-duplex source-synchronous communication channel. It is master-slave based link which provides multiple user configurations. The interface can be split over multiple link channels of custom width. Each link channel is source-synchronous and is sent with a clock and valid signal. In the given application we have chosen a 32-bit data interface which can be implemented as a single link of 32-bits or 4-link channels of 8-bits each. The multiple link configuration provides a fault tolerant design. The *bsg_comm_link* undergoes a calibration phase on reset to determine the number of active channels and configures the internal blocks accordingly. Thus, in case of failure of any of the links, the interface adopts appropriately and remains fully functional. This calibration requires a full-duplex configuration to setup the link channels. The *bsg_comm_link_lite* is a light weight version which bypasses the calibration phase to enable use of the master and slave blocks independently on different chips (ASIC or FPGA).

Figure 4.2(a) shows the schematic of *bsg_comm_link_lite*. It consists of three major sub-modules: Kernel, S-box and the Fuser. The kernel is the IO side interface with the Source Synchronous Input (SSI) in Slave and a Source Synchronous Output (SSO) in Master. This block is responsible for the clock domain crossing from the IO clock to the Core clock using an asynchronous-FIFO. The data channel is DDR (Double Data Rate) w.r.t the incoming clock and DDR to SDR conversion also takes place in the kernel. Figure 4.2(b) shows the DDR data at the IO side for one 8-bit link channel. The data aligned to the core clock then passes through the S-box and assembler where it is assembled into (from) the core channels from (to) the link channels format. In ArCuS, the standard core channel width is set to 80bits within the ASIC. So, the assembler (Slave) takes in 32 bit data and converts it to 80bit format every 2.5 packets (2.5*32=80) at the input. The Master has a similar functionality in the opposite direction. The *bsg_comm_link_lite* provides a means for high speed and reliable single-ended data transfer without the need of SERDES and other high speed IOs.

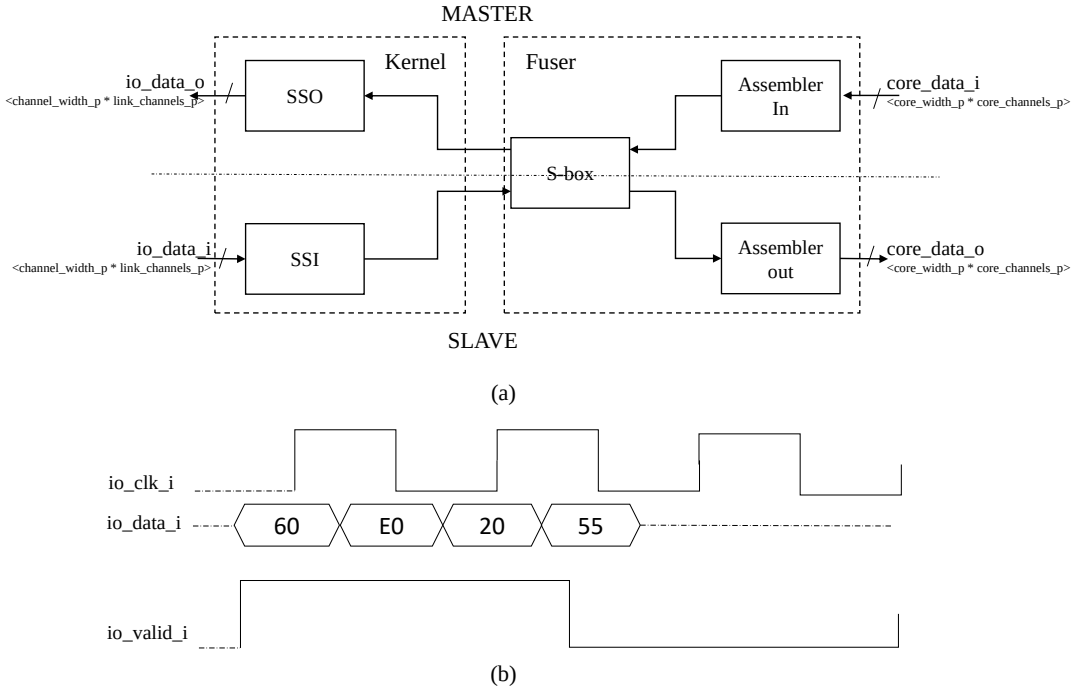

Figure 4.2: bsg_comm_link_lite: (a) Shows the schematic view of the bsg_comm_link_lite with IO and core side data interfaces. (b) DDR data on IO for one 8-bit link channel with clock and valid.

4.1.2 Test and Configuration

The *bsg_tag* module provides a simple serial on-chip configuration network. It is similar to the JTAG interface. This module's functionality is split into a Master and Client block. The ASIC instantiates a tag master module with serial data, clock and enable interface. One Master can communicate with multiple client modules. Master receives 1 bit data, decodes it and sends to respective client via 2 bit interface, op and param (data). A stream of 0's is used to reset the block. The internal zero counter counts till a preset maximum packet length and is then ready for data. The data format expected by the master is as follows. After reset, send a '1', followed by Header with the {<node_id>, <data_not_reset>, <payload_length>} and finally the <payload> for <payload_length> cycles. The master then sends data to the client based on the node_id. The client has logic for clock domain crossing (CDC) and a set procedure for reset. The data received at the client can then be used to configure other blocks and setup the ASIC ID on boot-up. Figure 4.3 shows the schematic and interface of the *bsg_tag* modules.

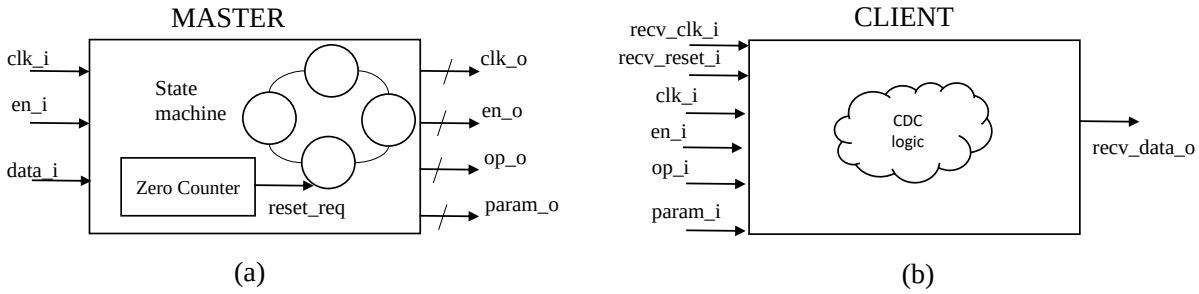

Figure 4.3: bsg_tag: (a) Master has a serial input interface and can handle multiple clients. (b) Client is used for configuring a block while essentially taking care of CDC.

4.1.3 Clock Generators

The ASIC uses two on-chip clock generators *bsg_clk_gen*. These are all-digital glitch-free inverter chain based circuit with down-sampling control. The clock is designed to be atomically updated between any of its values without glitching. The clock generator is divided into three sub-modules: Coarse Delay Tuner (CDT), Fine Delay Tuner (FDT) and Atomic Delay Tuner (ADT) as shown in Figure 4.4. The FDT provides fine delay control by utilizing the different loads on each of the clock paths which can be configured depending on the desired frequency. The ADT provides atomic switching between frequencies and consists of the delay chain. Finally, the CDT provides a coarse delay control by including 0,2,4 or 6 inverters in the clock path. To avoid any variation in duty-cycle, special clock inverters must be used which are generally provided in the standard cell library. The control inputs for frequency settings are synchronized to the internal clock. Each clock generator has a bsg_tag client instance to enable configuration through the tag inputs as discussed in section 4.1.2. Special down-samplers are also provided for a wider frequency range. The exact frequency range and the resolution (step size) depends on the technology library and the delays of the standard cells used. Option is also provided to by-pass these clock generators and use an external reference clock for core and IO with appropriate select inputs at the ASIC level. This may be useful for running RTL simulations to avoid using standard cell library and SDFs.

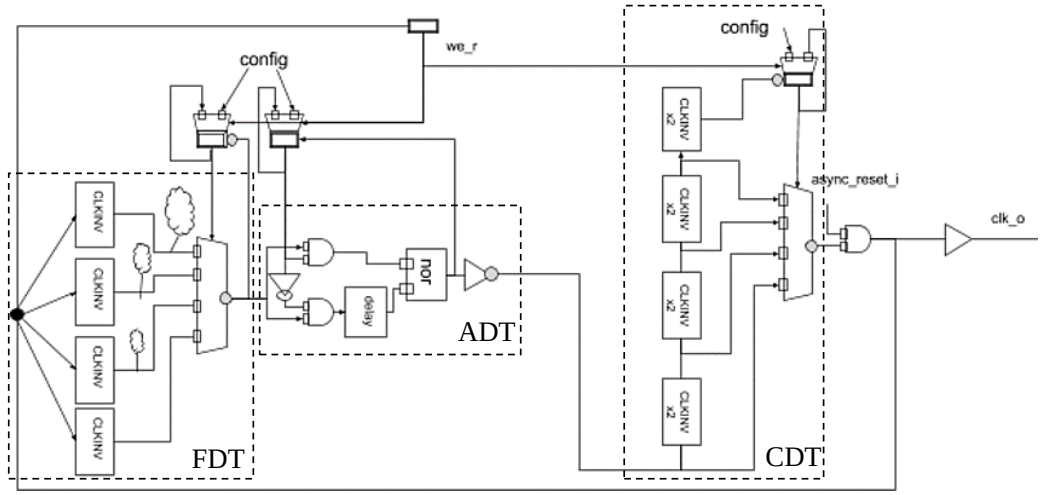

Figure 4.4: bsg_clk_gen Oscillator: The clock generator has a Fine Delay Tuner, Atomic Delay Tuner and a Coarse Delay Tuner to achieve glitch-free operation with large dynamic frequency range.

4.1.4 ASIC Controller

Once the incoming packet address is matched to the ASIC ID at the router, the *bsg_asic_controller* has the responsibility to assign the work to a free RCA. The controller also takes care of the packet format conversion which is compatible with the internal routing network. The controller receives a 80-bit packet in format defined in section 3.2.3, Figure 3.4. It maintains an ASIC level free list of RCAs as discussed in section 3.2.2. It checks for the available RCA in the list, changes the packet format and routes the required number of packets to occupy the RCA. The internal network uses the *bsg_mesh_router* which is detailed in next section. Once the RCA completes its execution, the controller accumulates the result packet and stores it in memory (one result per RCA). The packet is then formatted back to the On-PCB network format (Figure 3.5) with source address as the ASIC ID and is routed to the FPGA.

4.1.5 On-ASIC Routing Network

The accelerators (RCAs) are arranged in several lanes within the ASIC. There could be a maximum of 32 lanes, each with 16 RCAs (therefore, 512 RCA maximum). The 32 lanes are connected via *bsg_mesh_router* with address specified as (X,Y). The *bsg_mesh_router* is a general purpose router with capability of routing in all four di-

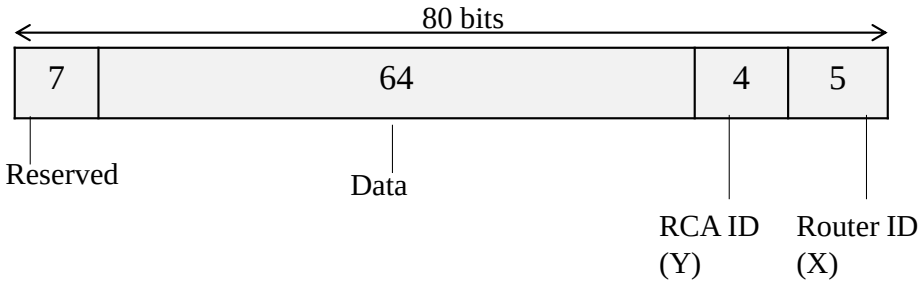

Figure 4.5: Mesh Router Format

rections (E,W,N,S) using dimensional ordered algorithm and one link to connect the processor (P). Here, we utilize the mesh router to route only in following directions: 1) West to East 2) West to South 3) South to West. The *bsg_asic_controller* is connected at port P of the router (0,0). All the unused router links are stubbed to avoid additional logic during synthesis. The routers provide buffering in form of FIFOs to handle traffic during routing. Figure 4.5 shows the packet format for the *bsg_mesh_router*. The X and Y co-ordinates are used for routing the packet to the appropriate lane.

Once the packet reaches the specified router lane (X), it is routed to the South (S) port. The RCA lanes are connected via *bsg_fsb* module. The fsb provides a star interconnection to each of the RCA. The request packet format issued by the fsb master is shown in Figure 4.6. The important field to note is the 4-bit destination address which is same as the Y-coordinate in the original mesh packet. The conversion of mesh to fsb packets is taken care by *bsg_mesh_fsb_convert*. After the RCA finishes the computation, the fsb client (RCA) pushes the result back to the fsb master. Figure 4.7 shows the response packet with the complete source address (X,Y). This packet is then directed to *bsg_asic_controller* i.e. (0,0) through the mesh routers. The *bsg_asic_controller* collects the result packet and updates the free list accordingly.

4.1.6 RCA: Bitcoin Miner

The final piece of the ASIC is *bsg_bitcoin_miner*, the accelerator. Section 2.3 gives an overview of bitcoin mining. Figure 4.8 shows the mining algorithm. The block header contains the version, previous hash, time-stamp, bits and nonce. The MSB of the block is constant across attempts and is precomputed. The nonce and initial hash

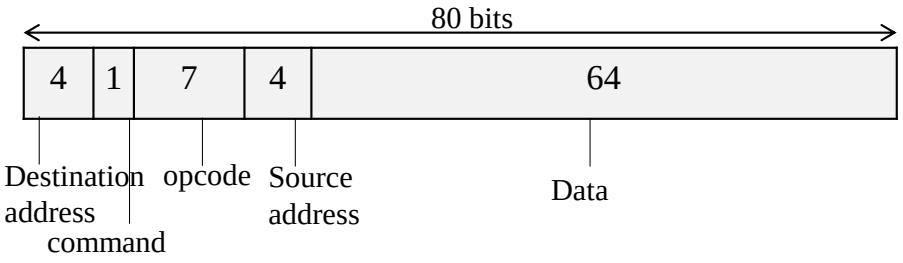

Figure 4.6: FSB Master Request Packet Format

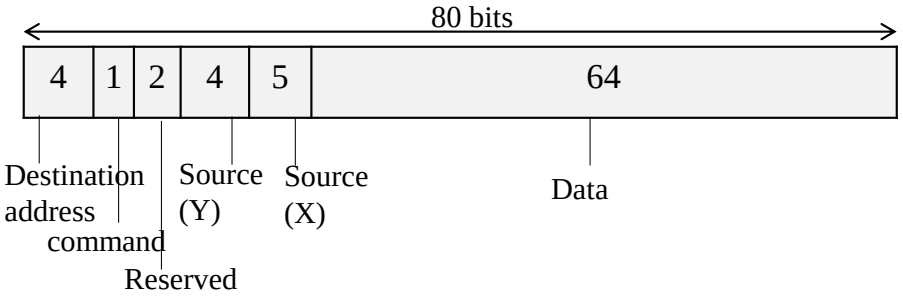

Figure 4.7: FSB Client Response Packet Format

go through two SHA-256 rounds. Billions of attempts are made until the hash is found (a number less than the reference value) or attempt fails. The hardware implementation is fully pipelined and is split in two 64-cycle stages. Thus, one mining latency is 128 cycles with one hash each cycle. The highlighted stages in Figure 4.8 makes up the RCA logic with each taking a 256bit hash and 512bit block input. A total of 768 bits data is needed to initiate the miner. This evaluates to 12 80-bit packets (12*64=768). The packet counting and buffering is taken care at the RCA input by a n:1 (12:1) module. Similarly, the output is a 256 bit hash value which equals 4 80-bit packets (4*64=256) which are accumulated by 1:n (1:4) module at RCA output.

4.2 Simulation Environment

The overall simulation environment of the design looks similar to Figure 3.2. The testbench setup for single ASIC is shown in Figure 4.9. Any number of ASICs could be instantiated for testing. The FPGA logic is implemented for simulations purposes only. It assumes that data after processing from the Ethernet is available in 80-bit

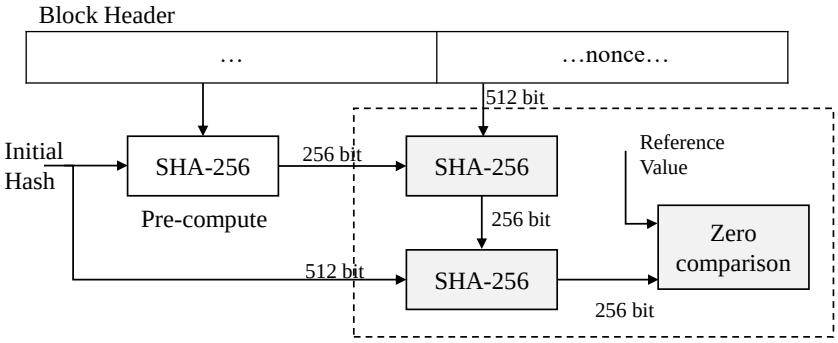

Figure 4.8: **Hashing:** The RCA incorporates two pipelined SHA-256 units with value comparison.

format. The FPGA controller checks the ASIC available RCA counter and routes the packet to the available ASIC through the *bsg_comm_link_lite* master interface. The ASIC after processing the data returns the results back to the FPGA which is received by the *bsg_comm_link_lite* slave. once all the packets are received, the ASIC free list is updated accordingly. Some counters have also been added for measurements in the simulation environment.

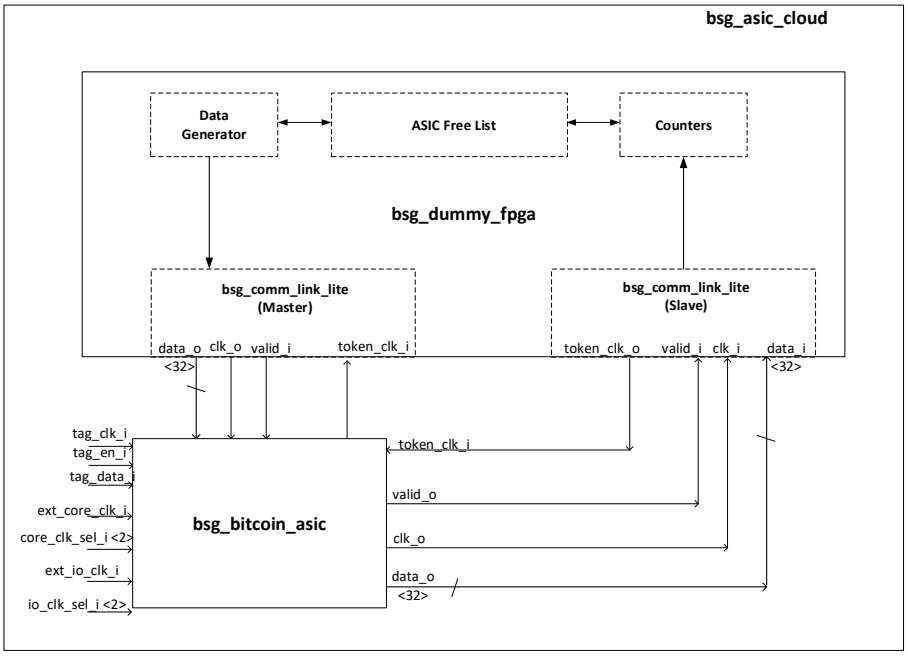

Figure 4.9: ASIC Cloud Simulation Environment

4.3 Tech-Node for Implementation

The main criteria for selecting an appropriate technology node for implementation is to optimize the Total Cost of Ownership (TCO) of the datacenter. The ASIC development cost is part of the Non-Recurring Engineering (NRE) cost which includes the labor, CAD tool licencing, IP and mask cost. Authors in [1] introduce a two-for-two rule which suggests that the TCO of the datacenter must be at least 2x the NRE investment for ASIC-based cloud to make sense. Khazraee et. al. [2] provide a detailed analysis for manufacturing, licensing and other NRE costs for a spectrum of technology nodes and applications. In general, it is observed that smaller technology nodes (16nm/20nm) leads to sub-optimal TCO and hence authors promote use of mature nodes like 65nm due to reduced mask and wafer costs. Next, we study various performance and cost metrics across technology nodes for the proposed system.

Chapter 5

Results

We developed the RTL for the proposed ArCuS architecture and evaluated it in various technology nodes to compare power, area and performance. Simulations were performed in Synopsys VCS, synthesized using Design Compiler (DC) and physical design, placement and routing done with Synopsys Galaxy Design Platform (IC Compiler). TSMC and UMC foundry models were used to implement the design. The nominal design parameters are shown in Table 5.1 with respective voltage and frequencies[2].

5.1 Utilization

The occupancy and utilization of the RCAs is directly dependent on the latency of each operation. We define utilization as the ratio of the actual number of RCAs used to the total number of available RCAs. These numbers are reported at both the ASIC (Figure 5.1) and server level (Figure 5.2) as a function of the RCA latency. The total available RCAs at ASIC level is assumed to be 512. Each ASIC adds a latency of 15 cycles/hop for the passing packets. Considering only one lane of 63 ASICs, each server can have up to 32,256 RCAs. It is noticed that to saturate one ASIC with 512 RCAs the

Table 5.1: Process-Voltage-Frequency

Tech Node	**180nm**	**130nm**	**90nm**	**65nm**	**40nm**	**28nm**	**16nm**
Voltage (V)	1.8	1.2	1.0	1.0	1.0	1.0	0.8
Frequency (MHz)	50	75	90	100	120	150	170

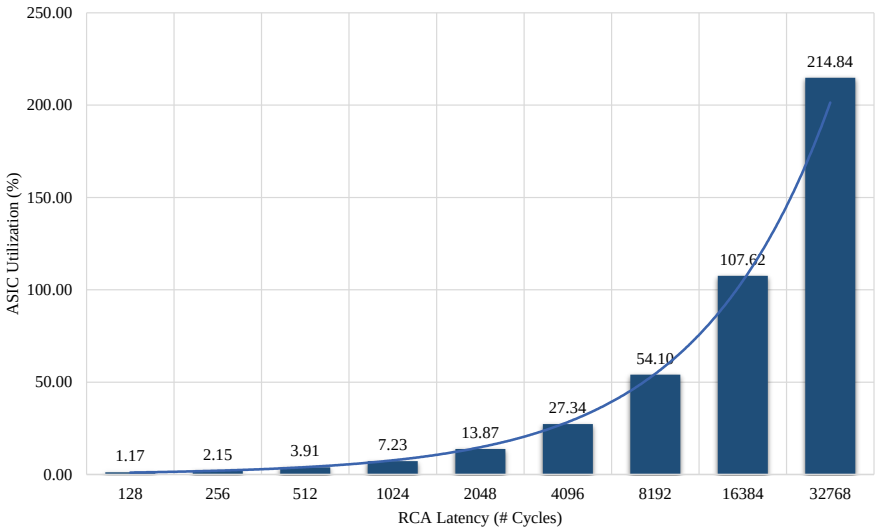

Figure 5.1: **ASIC utilization**: The percentage RCA utilization at ASIC level as a function of RCA latency. Latency of around 16k cycles is desired to achieve 100% utilization of the ASIC with 512 RCAs.

latency of the accelerator is close to 16k cycles while to fully-exploit the server with 63 ASICs, a latency of close to 1 Million cycles per operation is required. This could be an important factor while deciding the server and ASIC configurations depending on the application and anticipated workload.

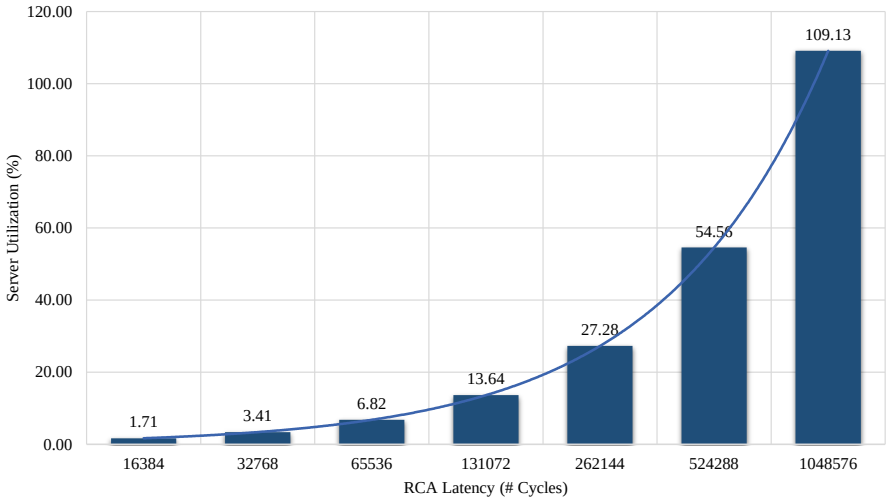

Figure 5.2: **Server utilization**: The percentage RCA utilization at Server level as a function of RCA latency. Latency of around 1M cycles achieves 100% utilization of a server with 63 ASICs.

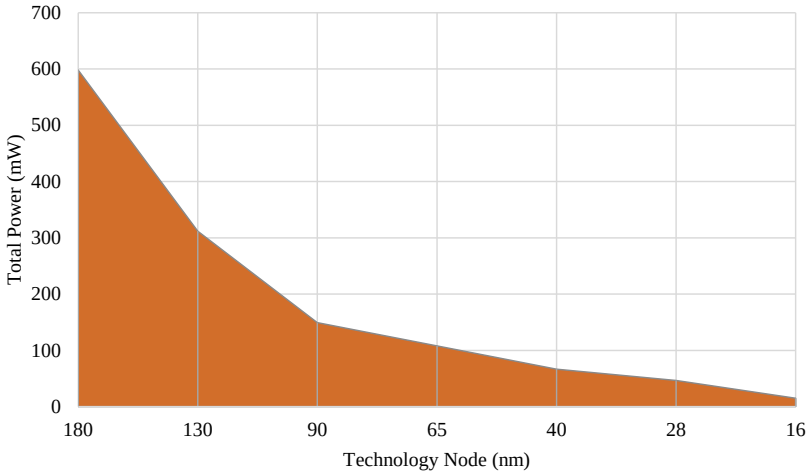

Figure 5.3: **Total Power**: Shows the total ASIC power with only 1RCA instance across tech-nodes.

5.2 Chip Power

The post-route power numbers of the ASIC with only one RCA is shown in Figure 5.3 across all technology nodes. There is a decrease of ≈40X total power when moving from 180nm to 16nm, while paying for the increased power density of 3.2X. It is noted that the power gain between 65nm and 28nm is not significant as the supply voltages did not scale appreciably in these nodes. The distribution of the total power shown in Figure 5.4 suggests that ASIC is dominated by the accelerator power. The bitcoin miner takes up 85% of the chip power while the infrastructure overhead of ArCuS is only 15% which is mainly due to the pads, on-PCB link and clock generators.

The total power per ASIC across technology nodes is shown in Figure 5.5. Power numbers are plotted on a logarithmic scale to accommodate the large difference in magnitudes on the same graph. We have considered no die-size limitations while evaluating the number of RCAs per ASIC which is addressed later.

5.3 Area Analysis

Figure 5.6 shows the ASIC area with one RCA instance across all technology nodes. The estimated gate count of the design post-synthesis is around 500K and similar to power, the area of the chip is dominated by the RCA. Only 13% of the total area is contributed by the communication links, pads, buffers, controller and the routing net-

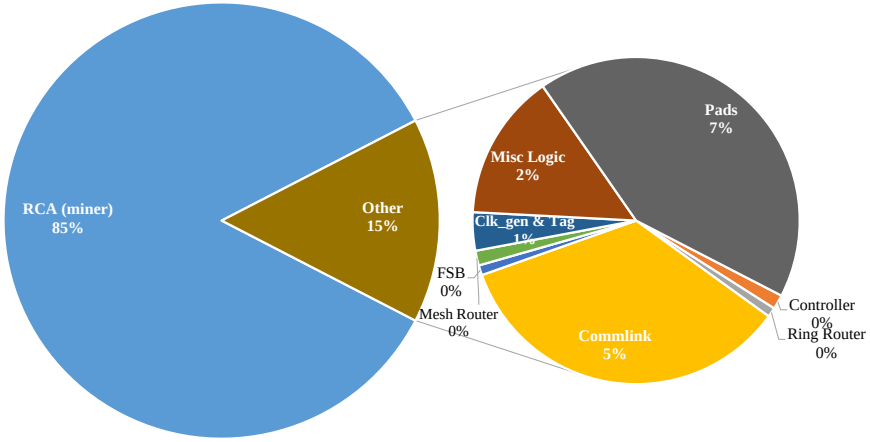

Figure 5.4: **Power Distribution**: Distribution of total power in ASIC with 1 RCA. The architectural overhead is only fraction of the total power. The accelerator power dominates the total chip power.

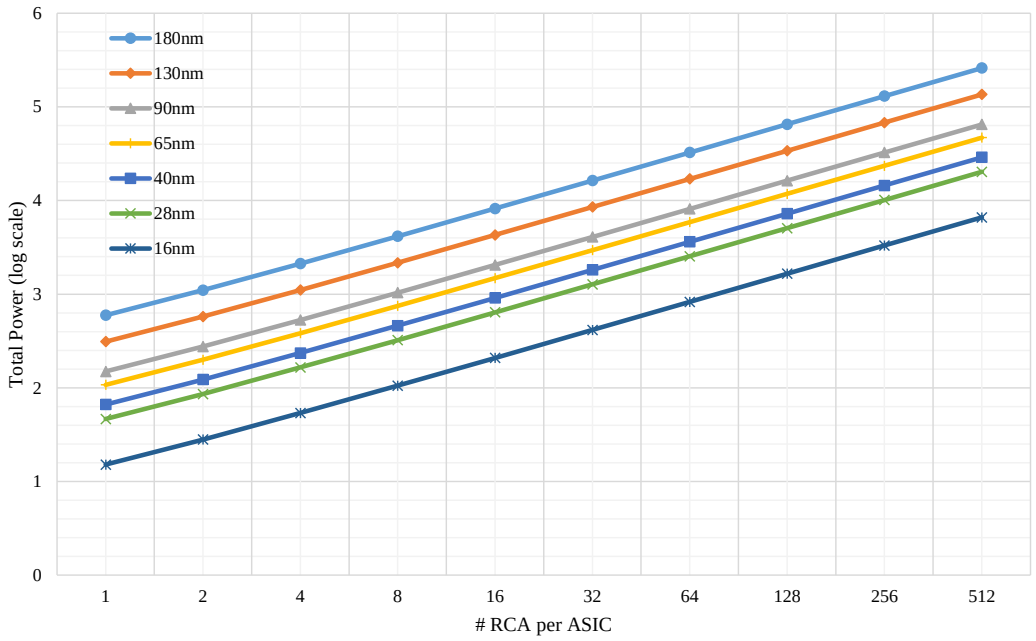

Figure 5.5: **Power vs. RCA count**: Shows the power in log scale at various technology nodes with different RCA per ASIC configurations.

work. Note that the clock generators are build only using standard cell clock inverters, mux and basic logic gates, hence have insignificant area, Figure 5.7. For the purposes of comparison of the area across technology nodes, Figure 5.8 shows the log of area with increasing RCA count per ASIC. In the analysis so far, the die size constraints have not been taken into consideration. It is essential to account for the die sizes and amount of logic that can be packed in the ASIC. Figure 5.9 is a bar plot of the maximum

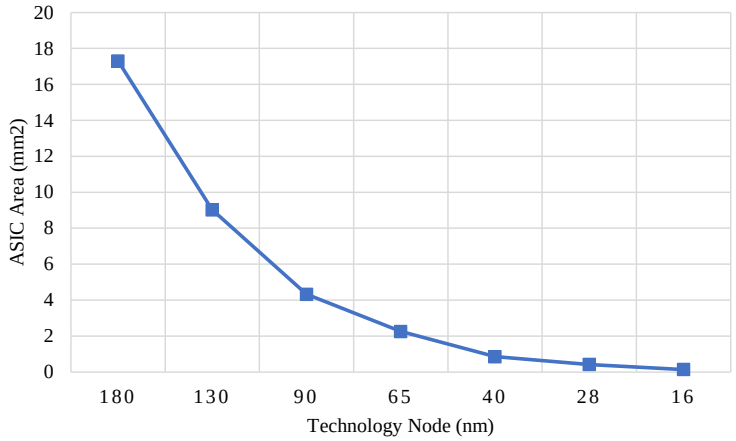

Figure 5.6: **Total Area**: Shows the total ASIC area with only 1RCA instance across technology nodes.

RCA count per ASIC possible for different die sizes at each technology node. We have considered five die sizes from as small as 5mmx5mm to as large as 18mmx18mm. For technologies 40nm and above, it is seen that it is not possible to fit in 512 RCA per ASIC even in 330mm^2 die and larger dies of up to 600mm^2 need to be considered. We have saturated the RCA count plots to 512 as it is the maximum supported RCA per ASIC in the proposed architecture. Die size of 200mm^2 is sufficient to pack 512 bitcoin miners at 28nm, while only 64mm^2 die is able to fit those at 16nm.

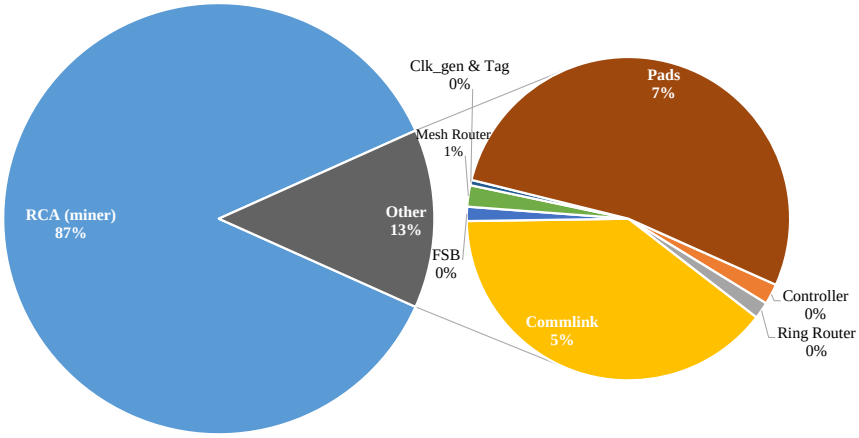

Figure 5.7: **Area Distribution**: Distribution of total area in ASIC with 1 RCA. The architectural overhead is only fraction of the total area. The accelerator dominates the total chip area.

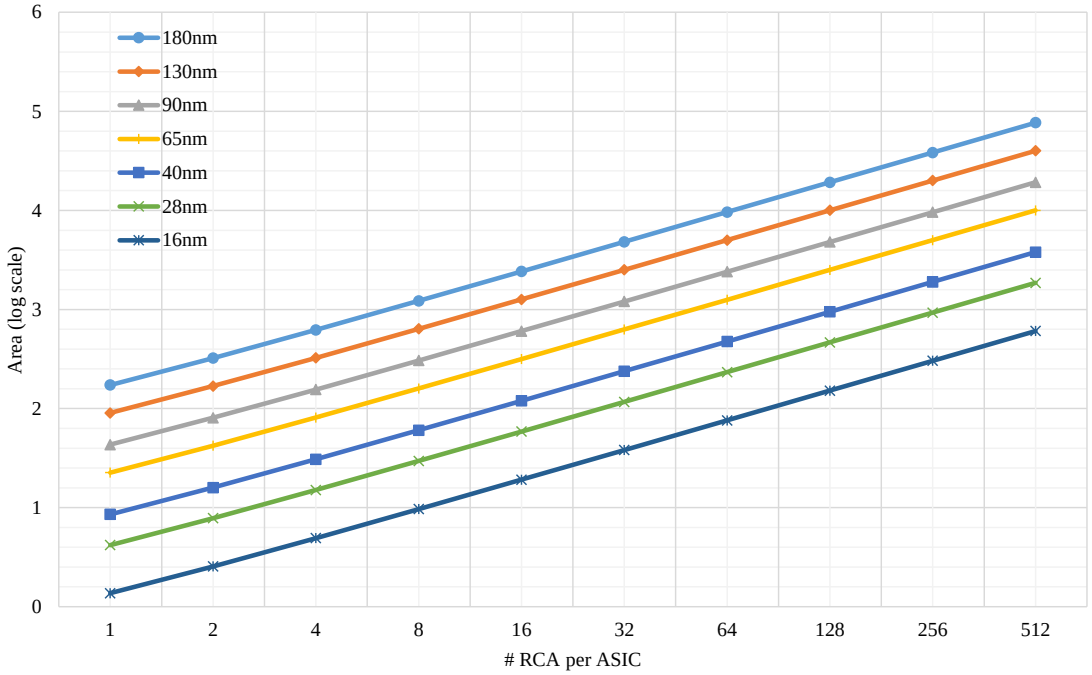

Figure 5.8: **Area vs. RCA count**: Shows the area in log scale at various technology nodes with different RCA per ASIC configurations.

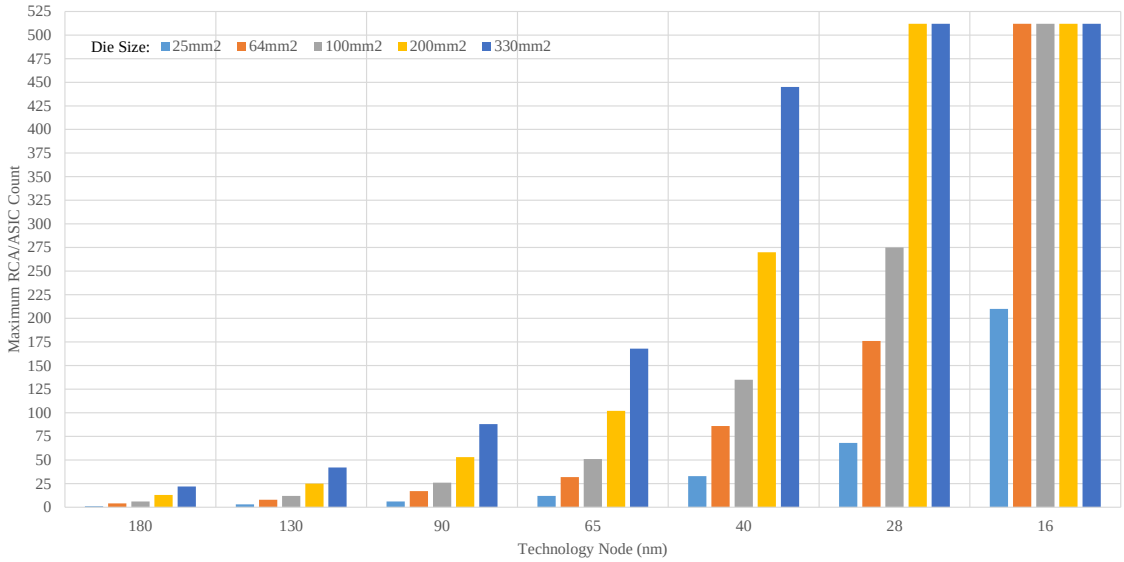

Figure 5.9: **RCA per ASIC:** Shows the amount of RCAs that can be packed in ASIC for various different die sizes at each technology node. The bar plots are saturated at 512 RCAs as supported by ArCuS.

5.4 Performance and Cost

Finally, we look into some important performance metrics for the ASIC cloud. We study a server configuration subject to following constraints. The maximum die size is set to 330mm^2 and maximum of 512 RCAs per die are supported. Each server has 12 lanes with 10 ASICs each. The server configuration and the essential parameters are summarized in Table 5.2. The performance metric in case of bitcoin mining is the hash rate or number of hashes per second. The hash rate is directly dependent on the core frequency and the number of RCAs on each server. Figure 5.10 shows the hash rate per server in Gigahash per second (GH/s). 28nm and 16nm servers clearly outperform others with hash rates reaching close to 10TH/s per server owing to higher density (therefore more RCAs) and faster clock speeds.

The die cost is evaluated based on the technology node and the die area. The total server cost in Table 5.2 models the silicon cost, package cost, heat sinks, fans, DC-DC and Power Supply units. This cost model is based on the authors findings in [2]. The energy efficiency of the server is calculated in terms of GH/s per Watt. Similarly, the operations by the total server cost defines the hashrate per $. The trend in Figure 5.11 shows an increase in efficiency and decline in power and cost requirements per GH/s with technology nodes. This is mainly due to the advantages of technology scaling, higher RCA density in a given die area, reduced total power and higher frequencies.

To be profitable in datacenters, it is not only essential to have higher performance but to maintain a reasonable Total Cost of Ownership. Although, from the metrics in Figure 5.10 and Figure 5.11 it is evident that 16nm gives the best cost and energy efficiency, it might not be profitable enough to build custom chips at advanced nodes.

Table 5.2: ArCuS: Representative Server Configuration

Tech node	**180nm**	**130nm**	**90nm**	**65nm**	**40nm**	**28nm**	**16nm**
Die Size (mm^2)	330	330	330	330	330	200	64
Die Cost ($)	9.90	15.50	16.96	17.24	25.44	23.97	10.86
GHps/Server	142	378	950	2,016	6,408	9,918	11,016
W/Server	1351.92	1854.75	2684.94	3700.19	6029.47	5224.95	2926.08
$/Server	1915.33	2656.07	2846.59	2882.96	3967.60	3773.18	2038.02
GHps/W	0.105	0.203	0.354	0.545	1.063	1.898	3.765
GHps/$	0.074	0.142	0.334	0.699	1.615	2.628	5.405

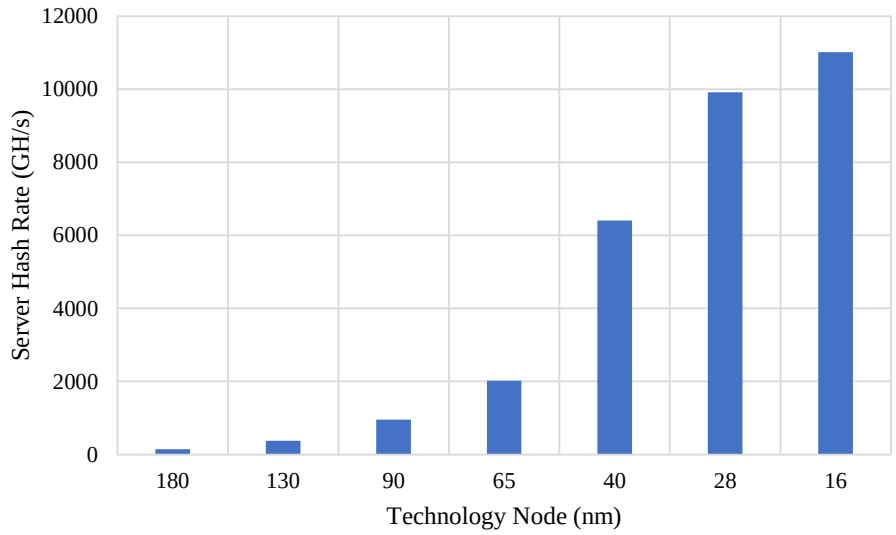

Figure 5.10: **Performance (GHps/Server)**: The hash-rate for each server on different technology nodes corresponding to configuration in Table 5.2 (Higher is better).

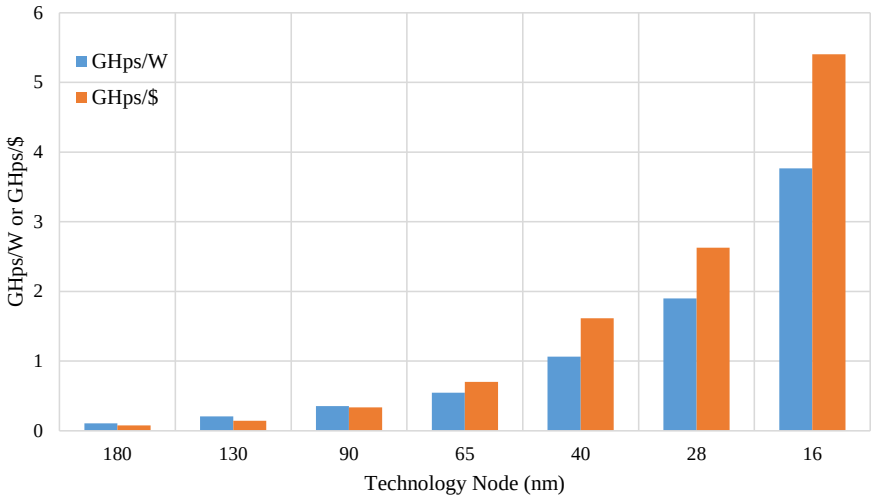

Figure 5.11: **Efficiency Metrics**: The plot shows the energy efficiency (GHps/Watt) and cost efficiency (GHps/$) for the server configuration at all technology nodes (Higher is better).

Figure 5.12(a) shows the variation of the total cost per server with technology nodes for our selected configuration. The absolute cost is noted to increase from 180nm to 40nm ASIC clouds. This is due to the increased die costs and the power per server, which increases the cooling and electricity requirements for the datacenter. However at 28nm and below, higher density leads to a reduced die area and hence reduction in overall cost. However, the trade-off comes in when we start to consider the Non-Recurring

Engineering costs to develop ASICs at lower technology nodes. Figure 5.12(b) shows the dramatic increase in NRE due to design and IP licensing costs when migrating from older to newer technology nodes [2]. Studying various performance and cost metrics, we can conclude 28nm is the candidate node for our application achieving high performance with reasonable NRE and cost per server. No doubt ASICs outperform CPUs and GPUs, however it is essential to consider all these effects while selecting the datacenter configurations and identify when it is worth to migrate to custom cloud architectures.

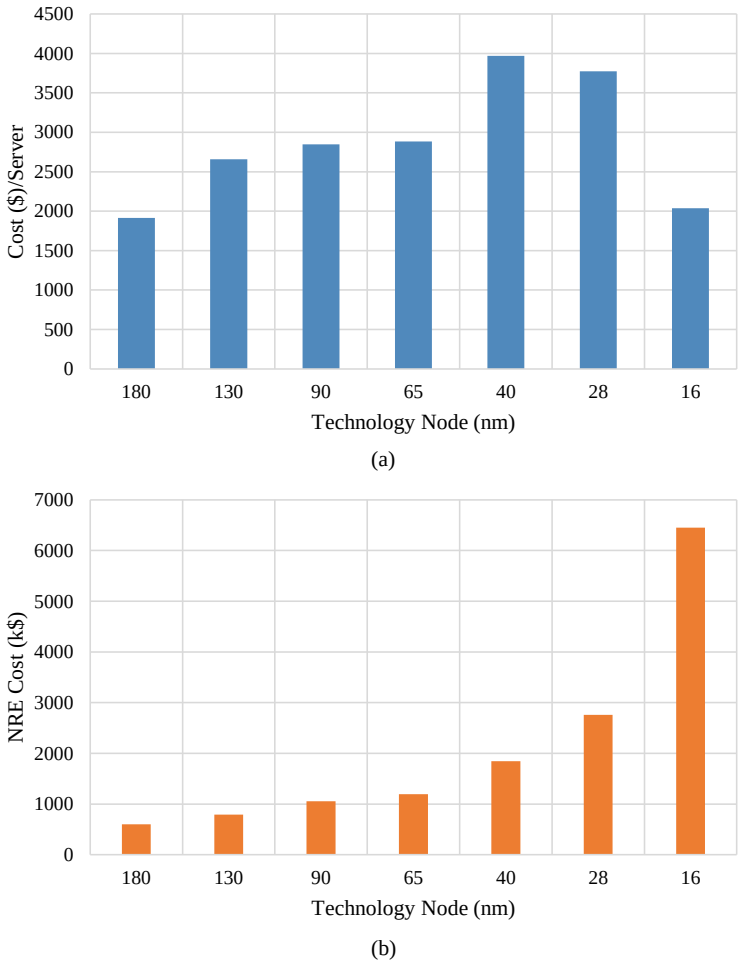

Figure 5.12: **Cost**:(a) Shows the variation of the cost per server across technology nodes (Lower is better). (b) The Non-Recurring Engineering (NRE) costs for each technology node [2] (Lower is better).

Chapter 6

Conclusions and Future Directions

6.1 Conclusion

Datacenter architectures have taken central stage as more applications move towards the cloud. In this thesis we explore the ideas proposed in [1] and present a light weight Architecture for ASIC Cloud based Server: ArCuS. We developed the specifications for communication at the server-level and laid out the microarchitectural details of the ASIC for bitcoin mining application. The complete ArCuS infrastructure has been built using open-source repository developed by University of California, San Diego with an aim to keep the design easily accessible to all research groups. We characterize the proposed architecture over various technology nodes from 180nm down-to 16nm. We examine area, power, cost and performance of ArCuS by considering a typical use-case in the server.

The performance and efficiency numbers at lower technology nodes outperform the older nodes due to the higher density, lower power and frequency scaling. However, considering the total cost of the server and the NRE effort involved in the design of the ASICs, the decision of selecting a technology node becomes non-trivial. Authors in [1] [2] have given detailed analysis of optimizing TCO and NRE over many applications. It has been observed and re-enforced by the findings in this work that the most advanced technology nodes might not always be profitable to consider for ASIC cloud applications.

6.2 Future Directions

This is an exciting time for server architecture researchers as more and more companies have identified the movement towards custom ASICs in datacenters in future. However, the intricate implementation details remain hidden from the research community as these enterprises prefer to remain confidential due to stringent competition. This thesis is an early effort to develop an open infrastructure for ASIC cloud applications with potential for further improvements.

In this work we implemented a logic intensive application with minimalistic memory requirements. Addressing the memory (DRAM) architecture and related communication protocols for the ASIC cloud remains an open challenge. Another aspect to consider, as identified earlier is the networking outside the server. It is an essential component to build the ASIC cloud datacenter. Depending upon the network interface and the delivery of workload packets, necessary modifications and design decisions are needed at the server controller (FPGA). We touch upon similar issues briefly in Chapter 3. We hope ArCuS will be one of the many server architectures to come that would promote development of new ASIC cloud applications and speed up the research in this domain.

Bibliography

[1] Ikuo Magaki, Moein Khazraee, Luis Vega Gutierrez, and Michael Bedford Taylor. ASIC clouds: specializing the datacenter. In *Proceedings of the 43rd International Symposium on Computer Architecture (ISCA)*, pages 178–190. IEEE Press, 2016.

[2] Moein Khazraee, Lu Zhang, Luis Vega Gutierrez, and Michael Bedford Taylor. Moonwalk: NRE Optimization in ASIC Clouds or, accelerators will use old silicon. In *Proceedings of the twenty second edition of ASPLOS on Architectural support for programming languages and operating systems*. ACM, 2017.

[3] Michael B Taylor. Is dark silicon useful? Harnessing the four horsemen of the coming dark silicon apocalypse. In *Design Automation Conference (DAC), 2012 49th ACM/EDAC/IEEE*, pages 1131–1136. IEEE, 2012.

[4] Robert H Dennard, Fritz H Gaensslen, V Leo Rideout, Ernest Bassous, and Andre R LeBlanc. Design of ion-implanted MOSFET's with very small physical dimensions. *IEEE Journal of Solid-State Circuits*, 9(5):256–268, 1974.

[5] Ganesh Venkatesh, Jack Sampson, Nathan Goulding, Saturnino Garcia, Vladyslav Bryksin, Jose Lugo-Martinez, Steven Swanson, and Michael Bedford Taylor. Conservation cores: reducing the energy of mature computations. In *Proceedings of the fifteenth edition of ASPLOS on Architectural support for programming languages and operating systems*, pages 205–218. ACM, 2010.

[6] Adam Kerin. True Heterogeneous Computing. `https://www.qualcomm.com/news/onq/2013/09/03/true-heterogeneous-computing`, 2013 (accessed February 24, 2017).

[7] Mark Shedd. Snapdragon 820 and Kryo CPU: heteroeneous computing and the role of custom compute. `https://www.qualcomm.com/news/snapdragon/2015/09/02/snapdragon-820-and-kryo-cpu-heterogeneous-computing-and-role-custom`, 2015 (accessed February 24, 2017).

[8] Leverage GPUs on Google Cloud for machine learning and scientific computing. `https://cloud.google.com/gpu/`, (accessed February 24, 2017).

[9] Nicole Hemsoth. Baidu Looks to Next Generation Deep Learning Accelerators. https://www.nextplatform.com/2016/01/21/baidu-looks-to-next-generation-deep-learning-accelerators/, 2016 (accessed February 24, 2017).

[10] A. Putnam, A. M. Caulfield, E. S. Chung, D. Chiou, K. Constantinides, J. Demme, H. Esmaeilzadeh, J. Fowers, G. P. Gopal, J. Gray, M. Haselman, S. Hauck, S. Heil, A. Hormati, J. Y. Kim, S. Lanka, J. Larus, E. Peterson, S. Pope, A. Smith, J. Thong, P. Y. Xiao, and D. Burger. A reconfigurable fabric for accelerating large-scale datacenter services. In *Computer Architecture (ISCA), 2014 ACM/IEEE 41st International Symposium on*, pages 13–24. IEEE, 2014.

[11] Stephen Weston. FPGA Accelerators at JP Morgan Chase, Stanford Computer Systems Colloquium. https://www.youtube.com/watch?v=9NqX1ETADn0, 2011 (accessed February 24, 2017).

[12] John Lockwood. Low-Latency Library in FPGA Hardware for High-Frequency Trading, Hot Interconnects. https://www.youtube.com/watch?v=nXFcM1pGOIE, 2012 (accessed February 24, 2017).

[13] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI). Savannah, Georgia, USA*, 2016.

[14] Quentin Hardy. Intel Betting on (Customized) Commodity Chips for Cloud Computing. https://bits.blogs.nytimes.com/2014/12/19/intel-betting-on-customized-commodity-chips-for-cloud-computing/?_r=0, 2014 (accessed February 24, 2017).

[15] List of Bitcoin Mining ASICs. https://en.bitcoin.it/wiki/List_of_Bitcoin_mining_ASICs, Retrieved 2016, (accessed February 24, 2017).

[16] Bespoke Silicon Group repository. https://bitbucket.org/account/user/taylor-bsg/projects/PROJ. (Accessed February 24, 2017).

[17] Michael B Taylor. A landscape of the new dark silicon design regime. *IEEE Micro*, 33(5):8–19, 2013.

[18] Venkatesh, Ganesh and Sampson, Jack and Goulding-Hotta, Nathan and Venkata, Sravanthi Kota and Taylor, Michael Bedford and Swanson, Steven. QsCores: Trading dark silicon for scalable energy efficiency with quasi-specific cores. In *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture (Micro-44)*, pages 163–174. ACM, 2011.

[19] Nathan Goulding-Hotta, Jack Sampson, Ganesh Venkatesh, Saturnino Garcia, Joe Auricchio, Po-Chao Huang, Manish Arora, Siddhartha Nath, Vikram Bhatt, Jonathan Babb, Steven Swanson, and Michael Bedford Taylor. The greendroid mobile application processor: An architecture for silicon's dark future. *IEEE Micro*, 31(2):86–95, 2011.

[20] Yakun Sophia Shao and David Brooks. Research Infrastructures for Hardware Accelerators. *Synthesis Lectures on Computer Architecture*, 10(4):1–99, 2015.

[21] Luiz André Barroso, Jimmy Clidaras, and Urs Hölzle. The datacenter as a computer: An introduction to the design of warehouse-scale machines. *Synthesis lectures on computer architecture*, 8(3):1–154, 2013.

[22] Luiz André Barroso, Jeffrey Dean, and Urs Holzle. Web search for a planet: The Google cluster architecture. *IEEE micro*, 23(2):22–28, 2003.

[23] Kevin Lim, Parthasarathy Ranganathan, Jichuan Chang, Chandrakant Patel, Trevor Mudge, and Steven Reinhardt. Understanding and designing new server architectures for emerging warehouse-computing environments. In *ACM SIGARCH Computer Architecture News*, volume 36, pages 315–326. IEEE Computer Society, 2008.

[24] Lisa Wu, Andrea Lottarini, Timothy K Paine, Martha A Kim, and Kenneth A Ross. Q100: The architecture and design of a database processing unit. *ACM SIGPLAN Notices*, 49(4):255–268, 2014.

[25] Oliver Pell and Oskar Mencer. Surviving the end of frequency scaling with reconfigurable dataflow computing. *ACM SIGARCH Computer Architecture News*, 39(4):60–65, 2011.

[26] Sai Rahul Chalamalasetti, Kevin Lim, Mitch Wright, Alvin AuYoung, Parthasarathy Ranganathan, and Martin Margala. An FPGA memcached appliance. In *Proceedings of the ACM/SIGDA international symposium on Field programmable gate arrays*, pages 245–254. ACM, 2013.

[27] Maysam Lavasani, Hari Angepat, and Derek Chiou. An FPGA-based in-line accelerator for Memcached. *IEEE Computer Architecture Letters*, 13(2):57–60, 2014.

[28] Hanaa M Hussain, Khaled Benkrid, Ahmet T Erdogan, and Huseyin Seker. Highly parameterized K-means clustering on FPGAs: Comparative results with GPPs and GPUs. In *Reconfigurable Computing and FPGAs (ReConFig), 2011 International Conference on*, pages 475–480. IEEE, 2011.

[29] John Beetem, Monty Denneau, and Don Weingarten. The GF11 supercomputer. In *ACM SIGARCH Computer Architecture News*, volume 13, pages 108–115. IEEE Computer Society Press, 1985.

[30] David E. Shaw, Martin M. Deneroff, Ron O. Dror, Jeffrey S. Kuskin, Richard H. Larson, John K. Salmon, Cliff Young, Brannon Batson, Kevin J. Bowers, Jack C. Chao, Michael P. Eastwood, Joseph Gagliardo, J. P. Grossman, C. Richard Ho, Douglas J. Ierardi, István Kolossváry, John L. Klepeis, Timothy Layman, Christine McLeavey, Mark A. Moraes, Rolf Mueller, Edward C. Priest, Yibing Shan, Jochen Spengler, Michael Theobald, Brian Towles, and Stanley C. Wang. Anton, a special-purpose machine for molecular dynamics simulation. *Communications of the ACM*, 51(7):91–97, 2008.

[31] Rich Miller. Amazon Building Custom Chips to Accelerate Cloud Networking. `http://datacenterfrontier.com/amazon-building-custom-asic-chips-to-accelerate-cloud-networking/`, 2016 (accessed February 24, 2017).

[32] Michael Bedford Taylor. Bitcoin and the age of bespoke silicon. In *Proceedings of the 2013 International Conference on Compilers, Architectures and Synthesis for Embedded Systems*, page 16. IEEE Press, 2013.

[33] Xilinx Inc. 7 Series FPGAs Data Sheet: Overview. `https://www.xilinx.com/support/documentation/data_sheets/ds180_7Series_Overview.pdf`, 2016 (accessed February 24, 2017).

[34] Digi-key Electronics. `http://www.digikey.com/?&WT.srch=1&gclid=CI2zz_m_rdICFYZsfgodBy8A1A`. (Accessed February 24, 2017).