

**COMPARATIVE ANALYSIS OF MEMORY HIERARCHIES FOR NOVEL  
PARADIGMS IN SUPERCONDUCTING LOGIC FAMILIES**

A Dissertation  
Presented to  
The Academic Faculty

By

Shunzhi Wen

In Partial Fulfillment  
of the Requirements for the Degree  
Bachelor of Science in Computer Science in the  
College of Computing

Georgia Institute of Technology

December 2023

© Shunzhi Wen 2023

**COMPARATIVE ANALYSIS OF MEMORY HIERARCHIES FOR NOVEL  
PARADIGMS IN SUPERCONDUCTING LOGIC FAMILIES**

Thesis committee:

Dr. Thomas M. Conte  
School of Computer Science  
*Georgia Institute of Technology*

Dr. Alexandros Daglis  
School of Computer Science  
*Georgia Institute of Technology*

Date approved: December 14, 2023

## ACKNOWLEDGMENTS

I would like to thank Dr. Thomas M. Conte for enlightening my passion for this field and providing key guidance in developing this work as part of the SYNDRA project. I would also like to thank another member of the CRNCH lab who is also the main coordinator of SYNDRA, Pulkit Gupta, for his insights and guidance that are vital to this work.

I give special thanks to other members in the SYNDRA group, Ryan Lynch and Devin Pohl for developing the compiler for SYNDRA and generating the benchmark traces to be used by this work, and Cynthia Wang for developing the original OOO simulation framework, which I modified to satisfy the experiment conditions in this work.

Lastly, I want to thank Dr. Alexandros Daglis for reviewing this work and providing valuable feedback and insights in developing this thesis.

## TABLE OF CONTENTS

<b>Acknowledgments</b> . . . . .	iii
<b>List of Tables</b> . . . . .	vii
<b>List of Figures</b> . . . . .	viii
<b>List of Acronyms</b> . . . . .	x
<b>Chapter 1: Introduction</b> . . . . .	1
<b>Chapter 2: Superconducting Electronics Background</b> . . . . .	3
2.1 Superconducting Electronics (SCE) . . . . .	3
2.1.1 Foundations of SCE and Primitives . . . . .	3
2.2 Various Superconducting Logic Families . . . . .	5
2.2.1 Rapid Single Flux Quantum (RSFQ) . . . . .	5
2.2.2 Reciprocal Quantum Logic (RQL) . . . . .	6
2.3 Advantages of SCE over CMOS . . . . .	7
2.4 Limitations of Superconducting Logic Families on Architectural Design . . . . .	8
2.5 Summary . . . . .	9
<b>Chapter 3: Architecture and Cache Background</b> . . . . .	11
3.1 Brief Introduction to VLIW and Dataflow Architectures . . . . .	11

3.2	Static sYnchronous Dataflow Risc Architecture (SYNDRA)	11
3.3	Banked Cache	12
3.4	L2 Cache	13
<b>Chapter 4: Novel Work</b>		<b>14</b>
4.1	Routing Unit Design	14
4.1.1	Cache Bank Selection Policy	14
4.1.2	Two Pseudo-Random Algorithms for PRIM	15
4.2	Memory Channel Interleaving	16
4.3	Load/Store Queue (LSQ)	17
4.4	Non-Blocking Cache	17
4.5	Cache Organization with a SCE Memory Primitive	18
<b>Chapter 5: Advanced Memory Technologies</b>		<b>20</b>
5.1	Possibility for Larger Cache Size on $20 \times 20$ Chiplet	20
5.2	L2 Cache Organization	20
5.3	DRAM Organization and Interconnect Latency	21
<b>Chapter 6: Methodology</b>		<b>22</b>
6.1	Out-of-Order (OOO) Execution Simulator	22
6.2	Simulation Configurations	23
6.3	Workloads and Evaluation Criteria	24
<b>Chapter 7: Experiments and Results</b>		<b>25</b>
7.1	Behavior of Banked Cache in CMOS	25

7.2	Impact of L2 Cache Configurations . . . . .	26
7.3	Impact of Cache Bank Configurations and Selection Policy . . . . .	27
7.4	Impact of Load/Store Queue (LSQ) Size . . . . .	31
7.5	Impact of MSHR and the Non-Blocking Mechanism . . . . .	33
7.6	Comparison between Baseline, Optimal, and Larger Cache for $20 \times 20$ Chiplets . . . . .	34
<b>Chapter 8: Conclusion . . . . .</b>		<b>36</b>
8.1	Summary . . . . .	36
8.2	Advantages and Limitations . . . . .	36
8.3	Future Work . . . . .	37
<b>References . . . . .</b>		<b>38</b>

## LIST OF TABLES

6.1	Simulation Configurations for SCE . . . . .	23
6.2	Simulation Configurations for CMOS . . . . .	23

## LIST OF FIGURES

2.1	SFQ Pulse . . . . .	4
4.1	Circuit Design of an 8-KiB Direct-Mapped Cache . . . . .	19
4.2	Circuit Design of an 8-KiB 2-Way Set-Associative Cache . . . . .	19
7.1	IPC Speedup for CMOS Cache Configs . . . . .	25
7.2	IPC Speedup for Single-Ported L1 Cache with Different L2 Configs . . . . .	26
7.3	IPC Speedup for 4 L1 Cache Banks with Different L2 Configs . . . . .	27
7.4	IPC Speedup for Different Cache Bank Configs (CMOS) . . . . .	27
7.5	L1 Miss Ratio for Different Cache Bank Configs (CMOS) . . . . .	28
7.6	IPC Speedup for Different Cache Bank Configs (High L2 Latency) (SCE) . . . . .	28
7.7	IPC Speedup for Different Cache Bank Configs (Low L2 Latency) (SCE) . . . . .	29
7.8	L1 Miss Ratio for Different Cache Bank Configs (High L2 Latency) (SCE) . . . . .	29
7.9	L1 Miss Ratio for Different Cache Bank Configs (Low L2 Latency) (SCE) . . . . .	30
7.10	Conflict Ratio for Different Cache Bank Configs (High L2 Latency) (SCE) . . . . .	30
7.11	Conflict Ratio for Different Cache Bank Configs (Low L2 Latency) (SCE) . . . . .	31
7.12	IPC Speedup for Different LSQ Sizes with SIM . . . . .	32
7.13	IPC Speedup for Different LSQ Sizes with H-Matrix . . . . .	32
7.14	IPC Speedup for Different MSHR Sizes with SIM . . . . .	33

7.15 IPC Speedup for Different MSHR Sizes with H-Matrix . . . . .	33
7.16 IPC Speedup for Baseline vs Optimal vs Larger Cache for $20 \times 20$ Chiplets	34

## LIST OF ACRONYMS

<b>3T-eDRAM</b>	3-Transistor Embedded DRAM
<b>AC</b>	Alternating Current
<b>ALU</b>	Arithmetic Logic Unit
<b>CMOS</b>	Complementary Metal-Oxide-Semiconductor
<b>DC</b>	Direct Current
<b>DRAM</b>	Dynamic Random Access Memory
<b>DSFQ</b>	Dynamic Single Flux Quantum
<b>EDP</b>	Energy Delay Product
<b>ERSFQ</b>	Energy-efficient Rapid Single Flux Quantum
<b>GF</b>	Galois Field
<b>I-Poly</b>	Irreducible Polynomial
<b>JJ</b>	Josephson Junction
<b>JTL</b>	Josephson Transmission Line
<b>LSQ</b>	Load/Store Queue
<b>LV-RSFQ</b>	Low-Voltage Rapid Single Flux Quantum
<b>MSHR</b>	Miss Status Handling Register
<b>OOO</b>	Out-of-Order
<b>PRIM</b>	Pseudo-Randomly Interleaved Memory
<b>RF</b>	Radio Frequency
<b>RQL</b>	Reciprocal Quantum Logic
<b>RSFQ</b>	Rapid Single Flux Quantum
<b>SCE</b>	Superconducting Electronics

**SFQ** Single Flux Quantum

**SIM** Sequentially Interleaved Memory

**SRAM** Static Random Access Memory

**SYNDRA** Static sYnchronous Dataflow Risc Architectures

**VLIW** Very Long Instruction Word

## SUMMARY

This thesis aims to propose a memory system solution to alleviate the constraints of superconducting logic families on architectural design. Due to the limitations on circuit density and fanout, multi-ported caches are not feasible in the realm of superconducting electronics. As an alternative, the project focuses on a banked cache system allowing independent access to memory partitions per bank, enabling parallel memory accesses. With advanced cache bank selection policies, like Pseudo-Randomly Interleaved Memory (PRIM), we can achieve load balance across cache banks with strong robustness against various access patterns. Other designs such as Load/Store Queue (LSQ) and Miss Status Handling Register (MSHR) are employed to further improve the efficiency of memory accesses. Additionally, a comprehensive examination of memory technologies is conducted to fulfill the needs of each level in the memory hierarchy. The primary goal of this work is to illuminate the challenges faced by the memory system for superconducting logic architectures and shed light on a viable approach to solve it.

# CHAPTER 1

## INTRODUCTION

Memory systems have played a critical role in advancing the field of computing. They provide a means for storing and retrieving data quickly and efficiently, which is essential for modern computing applications. Over the years, memory systems have evolved significantly, with improvements in both capacity and performance.

However, the development of memory systems is not without its challenges. One of the main constraints is the limited bandwidth and high power consumption of traditional memory technologies, such as Dynamic Random Access Memory (DRAM) and Static Random Access Memory (SRAM). Attempts to decrease the power consumption of computational circuits and devices has led to the exploration of new technologies. One such technology is Superconducting Electronics (SCE). Superconducting circuits are based on the principles of superconductivity, enabling them to operate with much lower power consumption than traditional Complementary Metal-Oxide-Semiconductor (CMOS) circuits. However, the use of superconducting logic families also imposes constraints on the design of memory systems, for example, prohibiting the design of a performant multi-ported cache. Therefore, this work aims to design a high-performance, energy-efficient memory system achievable within the constraints of superconducting logic.

To solve the problem, this work will first introduce the background of SCE and the constraints of superconducting logic families (chapter 2). Then, chapter 3 briefly describes the targeted architecture, a basic scheme for L1 cache, and a L2 memory technology that works in cryogenic environment. Chapter 4 explores the landscape of possible memory hierarchies that respect the constraints and proposes relevant factors to experiment with. Chapter 5 provides more details about memory technologies available to this work. In chapter 6 and chapter 7, workloads of interest will be characterized, and their performances

with different cache hierarchies will be evaluated by creating a simulation framework that is cognizant of architectural and technological constraints. Finally, chapter 8 will summarize the designs, discuss their advantages and limitations, and envision future work.

## CHAPTER 2

### SUPERCONDUCTING ELECTRONICS BACKGROUND

#### 2.1 Superconducting Electronics (SCE)

This section introduces the foundations of Superconducting Electronics (SCE), reviews existing superconducting logic families, and discusses the advantages and limitations of superconducting compared with traditional CMOS logic gate circuits.

##### 2.1.1 Foundations of SCE and Primitives

Superconductors are materials that exhibit superconductivity, the ability to conduct electricity with zero resistance. These materials can express superconductivity when brought below a certain temperature, the “critical temperature”. A material’s critical temperature is unique to its chemistry, structure, and the pressure applied to it, with the most well-known superconductors exhibiting critical temperatures ranging from near absolute zero (millikelvins) to 203 K [1]. Using this property of superconductors, the Josephson effect that describes the tunneling of superconducting current between two superconductors separated by a barrier [2], and Maxwell’s Equations, we can build circuits that operate differently from traditional CMOS and achieve lower power consumption.

This work characterizes superconducting circuits as being comprised of three basic components: superconducting loops, Josephson Junctions (JJs), and Single Flux Quantum (SFQ) digital logic. There is also current supply, interconnect, and phase shift elements, which are less important for understanding the basic principles of SCE.

A **superconducting loop** [1] can store magnetic flux, which sets it apart from a loop made of normal, resistive material. As long as the loop remains in the superconducting state, the current can circulate indefinitely. A superconducting loop functions as an ideal

capacitor, but instead of storing charge, it stores magnetic flux .

**Single Flux Quantum (SFQ)** [1] refers to a quantized magnetic flux (Figure 2.1) that serves as a signal flowing between components. Various representations of digital “0” and “1” can be achieved by manipulating the presence, absence, polarization, or location of magnetic flux quanta within a circuit element.

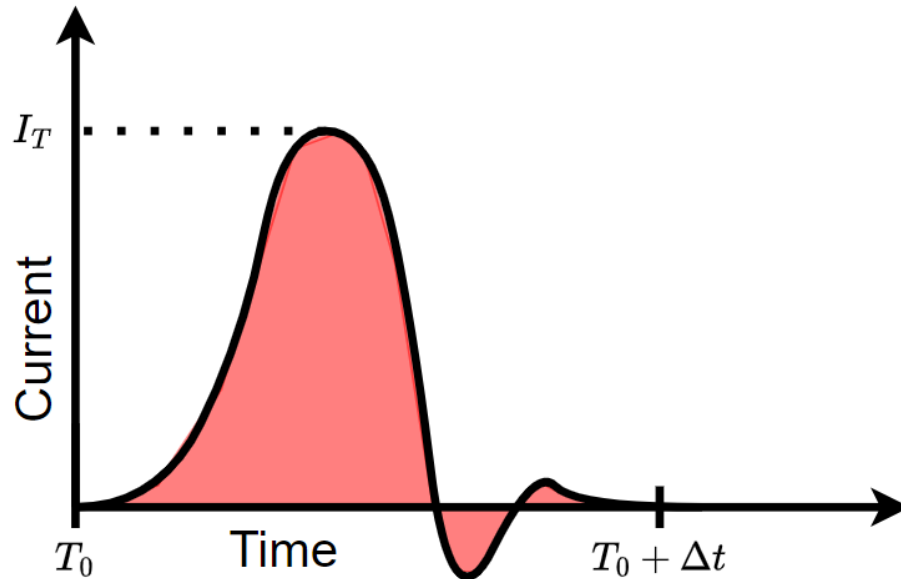


Figure 2.1: A current vs time graph of an SFQ pulse where the area under the curve is directly proportional and approximate to the flux quantum.

**Josephson Junctions (JJs)** [1] are used to control the flow of current by enabling switching behavior akin to CMOS transistors. A JJ consists of a thin insulative or non-conductive film sandwiched by superconductors. When a current exceeding the critical current ( $I_c$ ) is applied to the junction, the junction “switches” and pairs of superconducting electrons (Cooper pairs) can quantum tunnel through the thin barrier layer, enabling a superconducting current to flow between the junction’s contacts without any voltage drop. In the case of SCE when the junction switches, an SFQ pulse is emitted to be read by the next component.

With these components, we can build gates in superconducting circuits and realize the functionality of traditional semiconductor digital circuits.

## 2.2 Various Superconducting Logic Families

Since the 1980s, many different superconducting logic families have been developed. They differ in how they communicate between gates (using SFQ or currents), power supply, static and dynamic power, and number of JJs per logic gate. This work will focus on SFQ-based logic families, such as Rapid Single Flux Quantum (RSFQ) [3] and Reciprocal Quantum Logic (RQL) [4].

### 2.2.1 Rapid Single Flux Quantum (RSFQ)

RSFQ [3] was one of the earliest superconducting logic families, and it became popular in the 1990s. Despite its high energy efficiency compared to other technologies available at the time, taking advantage of that efficiency was of less concern than the clock speed advantages. Individual T-flip flops in RSFQ have been shown to reach a maximum operating speed of 770 GHz [5], making it the fastest digital circuit technology ever developed. While the energy efficiency of RSFQ is significantly better than CMOS, it could be further improved as the static power dissipation is about 100 times the dynamic power dissipation [6]. Another challenge is the nature of the RSFQ gates which require resetting each gate's state after every clock cycle. All RSFQ gates have inherent states to represent "0" and "1" based on the resistive/superconducting states of JJs, whose reset is triggered by a Radio Frequency (RF) current supply. This can lead to significant overhead in the wiring area and limitations on the clock frequency due to the punch-through effect that the junction may switch to the other resistive branch when the RF supply frequency exceeds the order of 1 GHz [7].

Variants of RSFQ tried to solve these problems. Low-Voltage Rapid Single Flux Quantum (LV-RSFQ) connects inductors in series with current supply resistors to reduce the static power dissipation at the cost of lower clock frequency and additional circuit area [8]. Energy-efficient Rapid Single Flux Quantum (ERSFQ) further replaced the resistors with

current-limiting JJs to eliminate static power dissipation [6]. It used phase balancing to eliminate parasitic circulating currents and employed bias inductors with lower inductance to reduce power consumption [9]. Dynamic Single Flux Quantum (DSFQ) addressed the issue of resetting by enabling self-resetting of the gates after keeping the states for about 25 ps and eliminating the necessity of a clock signal per gate [10].

### 2.2.2 Reciprocal Quantum Logic (RQL)

The primary distinction between RQL [4] and RSFQ lies in their power supply schemes. In RSFQ, DC power is applied to JJs in parallel via bias resistors. Conversely, in RQL, Alternating Current (AC) power is applied in series through bias transformers [9].

The proposed power supply scheme for RQL offers several key advantages. It eliminates static power dissipation by removing the need for DC bias current and bias resistors, increasing energy efficiency. This scheme also addresses the issue of large magnetic fields caused by returning bias current in RSFQ circuits [11]. Additionally, the serial bias supply simplifies circuit design by maintaining a low bias current amplitude independent of the number of JJs. This removal of the parallel bias current enables much more complex circuits with a greater total number of JJs/gates as otherwise the bias current amplitude would be too great and introduce substantial noise. The bias current also serves as the clock signal, eliminating the need for a separate SFQ clock distribution network [9]. Lastly, the AC clock signal ensures stability and a much lower bit error rate than RSFQ due to a higher operating margin [4]. Overall, the proposed power supply scheme enhances the performance and efficiency of RQL circuits. However, there is a drawback associated with the power supply scheme as well. To achieve a proper power supply, high-frequency power splitters are required, which can occupy a significant amount of area [9].

The total power dissipation of RQL is similar to that of ERSFQ circuits. Both exhibit minimal static power dissipation, but during data propagation processes, dynamic power dissipation arises from the switching of Josephson junctions.

### 2.3 Advantages of SCE over CMOS

Superconducting architectures offer advantages in both energy efficiency and operating speeds.

In all SFQ-based logic families, the flux is quantized to an SFQ, like a pulse, so the energy used to encode an SFQ is necessarily small. The energy required to generate an SFQ ( $\Delta E \approx I_c \Phi_0$ ) is of the order of  $2 \times 10^{-19}$  J for the critical current  $I_c = 100 \mu A$  [7]. The low energy for SFQ pulse generation allows a reduced energy expense on carry it on a wire.

While this work does not consider supercomputers, a substantial demand for improved energy efficiency is shown by advancements in the energy efficiency of superconductors as reported by Green500. According to the Green500 list released in June 2023 [12], the most recent supercomputer “Frontier” has demonstrated a power consumption of 22703 kW and energy efficiency of 52.592 GFLOPS/W. Compared with its predecessor “Summit” whose energy efficiency was 14.719 GFLOPS/W, efficiency has improved significantly.

Superconducting processors have demonstrated a potential for much higher energy efficiency. A research group designed a 32-bit RQL-based Arithmetic Logic Unit (ALU) at 16.3 GHz with energy efficiency of 4.88 PetaOPS/W [13]. Considering that every Watt dissipated within a cryostat can require 500 Watts of energy to cool, the energy efficiency can still reach 9.76 TOPS/W. The advantages of intrinsic elements in superconducting devices over CMOS are not obvious when cryocooling is taken into account [1], and it is difficult to find a corresponding CMOS ALU to compare against. However, when analyzing the energy efficiency of interconnects, superconducting devices have demonstrated advantages in both energy consumption and clock frequency over CMOS by orders of magnitude. While CMOS has energy consumption of  $\sim 10^{-12}$  J and clock frequency of  $\sim 1$  GHz, RQL shows  $\sim 10^{-18}$  J and over 10 GHz.

Meanwhile, heat dissipation also becomes a limiting factor for clock frequency. With

CMOS, as the transistors become smaller, the power density increases, which requires better cooling systems and reasonable clock frequency to make the heat manageable. Therefore, today's most advanced chips still maintain the clock frequency at  $\sim 4$  GHz [14].

In contrast, the properties of superconductors and the way that signals are propagated in SCE result in much less power dissipation, thus allowing for higher clock frequency than CMOS. RSFQ undeniably stands out as the fastest digital technology achieved thus far, with the ability to attain a clock frequency of  $\sim 70$  GHz in the current technology node. Moreover, by increasing the critical current density to  $0.5 \text{ mA}/\mu\text{m}^2$  and beyond, a clock frequency over 100 GHz becomes achievable [14]. However, to fulfill the maximum frequency, the amount of work that can be accomplished in each cycle is reduced, so the final speedup is still limited by the depth to which pipelines can be constructed in SCE.

## 2.4 Limitations of Superconducting Logic Families on Architectural Design

Superconducting logic families imposes two primary constraints on an architecture: low circuit density and lack of fanouts.

The density of superconducting circuits is still significantly less than that of CMOS. The current superconducting film thickness can only achieve 60 nm (compared to 3 nm for CMOS) [14], and the majority of RSFQ and RQL circuits are fabricated on a 200-300 nm node. Based on comparative results on the synthesis of a 32-bit ALU, an RQL-based ALU would take at least 4x the area of an equivalent ALU in 28nm CMOS [13] [15].

The lack of fanouts is caused by the limited drive strength. In general, all SFQ-based logic families have gates with low drive strength and require the use of Josephson Transmission Lines (JTLs) to drive the gates. A JTL comprises two JJs and can effectively limit clock skew and jitter. Given the limited drive strength, it becomes necessary to have one JTL for each output load to ensure proper operation [16].

When a gate requires multiple fanouts, a specific SFQ gate known as a splitter cell is inserted at the gate's output, which produces more fanouts. To accommodate additional

fanouts, a tree structure of splitter cells must be employed, allowing for branching and supporting multiple connections [17]. Traditional splitters only have a fanout of 2, so if we have many components that require multiple fanouts, the binary trees of splitter cells will take significant area. For example, a mux that selects between 2 64-bit values would require the 1-bit select signal to be replicated 64 times for each bit of the value inputs, which needs a tree of 63 2-fanout splitters. Recent work mitigates the problem by designing an 8-output splitter cell with fewer JJs than a tree of traditional splitters to produce 8 outputs [18]. However, when designing some complex components like multi-ported memory, the structure to accommodate multiple fanouts in each cell can still lead to very inefficient area utilization. Since chip space is limited due to the low circuit density, the number of boosters and splitters must be balanced with other critical components to reach the best performance and retain fabrication feasibility.

## **2.5 Summary**

While superconducting logic families offer several advantages over CMOS in terms of energy efficiency and operating speeds, the design of superconducting architectures faces limitations, including circuit density and fanout constraints. Superconducting circuit density is order-of-magnitude lower than that of CMOS, thus making the chip space very precious. The gates of all superconducting logic families have limited drive strength and necessitate the use of JTLs. The fanout count in SFQ-based logic families is typically limited to one, requiring the insertion of splitter cells and potentially leading to increased area utilization. Balancing the number of boosters and splitters with other critical components becomes crucial due to precious chip space.

A traditional multi-ported cache would require substantial insertion of splitter cells for every bit storage cell in a memory array. Additionally, the cost of additional decoders and select lines would drastically increase the area required due to the magnification of parasitic effects in superconductivity. Due to these costs, the area afforded for a multi-ported cache

can be better spent on computational units if cache organization is improved to mitigate the performance degradation from the lack of the multi-ported cache. Techniques such as banked cache with advanced cache bank selection policies, and non-blocking cache can be applied to alleviate the effects of a limited number of fanouts. By exploring various cache organization configurations, we are able to compare their effects on the performance of the entire system given our target workloads.

Despite these limitations, superconducting technologies show promising potential for higher energy efficiency and faster operating speeds. Through innovative approaches and careful consideration of factors such as buffer integration and fanout management, we can enable higher and more consistent performance with the same cache volume.

## **CHAPTER 3**

### **ARCHITECTURE AND CACHE BACKGROUND**

#### **3.1 Brief Introduction to VLIW and Dataflow Architectures**

Very Long Instruction Word (VLIW) architecture [19] defines the grouping of independent instructions into a VLIW multi-op via static scheduling. This approach eliminates the necessity for hardware to dynamically schedule instructions or check dependencies. Instead, the hardware ensures that each multi-operation executes in lock-step, while the compiler handles instruction scheduling based on dependencies between instructions. This strategy reduces hardware complexity by eliminating the need for dynamic scheduling and dependency checking.

Dataflow architecture [20] triggers the execution of an instruction solely upon the availability of its input data. Each instruction specifies output arcs directing to subsequent instructions that use the output as an operand. In this type of architecture, an arc can be expressed as a coordinate that describes the absolute or relative position of an instruction that depends on this instruction's data. This design effectively mitigates VLIW's bottleneck associated with a centralized register file, enabling the exploitation of maximum instruction-level parallelism within the constraints of the hardware.

#### **3.2 Static sYnchronous Dataflow Risc Architecture (SYNDRA)**

SYNDRA is an architecture built for SFQ-based superconducting logic families that aims to take advantage of their unique properties to achieve a lower Energy Delay Product (EDP) than CMOS architectures. EDP is the product of the total energy and time spent on executing applications. This metric demonstrates the energy efficiency of the architecture given certain workloads.

By symbiotically combining the properties of Very Long Instruction Word (VLIW) and dataflow architectures, SYNDRA alleviates the constraints on fanout by eliminating multi-ported structures, and it reduces hardware complexity by distributing the scheduling work to the compiler. It has a frontend with an instruction cache that prepares a VLIW multi-op composed of 4 instructions for each cycle. Using the dataflow approach, each instruction can specify at most 2 output arcs pointing to the future instructions that will use its result. The instructions in a multi-op are assigned to appropriate execution chiplets that each contain a complete set of function units. If the instruction is a load or store instruction, it will be forwarded to its corresponding cache chiplet by a routing unit based on the address generated. Each cache chiplet will handle accesses assigned to it independently and return the data requested by load instructions to an output queue once ready. The output queue will forward data to the appropriate execution chiplet at the time specified by the load instructions, satisfying the dependence arc that was encoded in the load instruction. In each cycle, if any instruction in a multi-op has an unready operand due to a cache miss, it will stall the frontend and execution chiplets and wait for the data from cache chiplets. This architecture uses a 3 phase clock at 5 GHz, and thus operates at an effective frequency of 1.67 GHz. This architecture is currently in development and as such additional details are not yet available.

### **3.3 Banked Cache**

While the multi-ported cache is infeasible in implementation in superconducting logic families, we need to figure out a method for the SYNDRA memory system to support accesses in parallel to avoid the bottleneck here. One promising design to overcome this bottleneck is to use a banked cache. By partitioning the entire memory address space into sections, one can map each section to a cache bank and therefore allow parallel accesses to different memory sections. The methods for mapping partitioned memory address space include Sequentially Interleaved Memory (SIM) and Pseudo-Randomly Interleaved Mem-

ory (PRIM) [21], which will be further discussed in chapter 4. As a frontend to the cache banks, a routing unit is responsible for assigning the memory access to a specific section to its corresponding cache bank. Several factors can affect the performance of cache banks: the number/size/organization of cache banks, cache bank selection policy, and the size of load/store queues.

### **3.4 L2 Cache**

In order to build a 512-KiB 8-way set-associative cache per L1 cache bank operating at 5 GHz as the L2 cache for SYNDRA, there is a need for a memory technology characterized by minimized latency and communication overhead to the processor within a cryogenic setting. Cryocache [22], a cache organization tailored for use within a 77K temperature environment, demonstrates promise by employing 3-Transistor Embedded DRAM (3T-eDRAM), exhibiting latency of 8 cycles. The feasibility and ramifications of integrating this technology as our L2 cache solution will be investigated.

## CHAPTER 4

### NOVEL WORK

This chapter explores the design of some core components ensuring parallel memory accesses. Within the scope of SCE, where latency in memory hierarchy is much longer and circuit density is much lower than that in CMOS, how these limitations would impact our design choices must be understood in depth. While multi-ported cache in CMOS has addressed the problem of parallel memory accesses, it is not supported in SCE, leading to an exploration for a different approach. This chapter proposes a novel solution that works within the constraints of SCE on fanouts, with techniques like advanced cache bank selection policies, memory channel interleaving, load/store queuing, and non-blocking cache to allow for efficient memory access within the system.

#### 4.1 Routing Unit Design

The routing unit takes memory access from the address generation stage and routes it to the appropriate cache bank determined by a particular cache bank selection policy. It is also responsible for serializing conflicts to the same cache bank. For the discussion below, assume  $n$  is the bit size of the block address, and  $m$  is the bit size of the bank ID.

##### 4.1.1 Cache Bank Selection Policy

The most common policy for cache bank selection is Sequentially Interleaved Memory (SIM) which uses the  $m$  lower bits of block address as the bank ID for  $2^m$  cache banks. This approach is simple for implementation and requires minimum overhead in the routing unit. However, one remaining drawback is that when the workload exhibits an access pattern with certain strides, all traffic is routed to the small set of cache banks, resulting in a degradation of performance due to under-utilization of caches. To improve the robustness

of the memory system when facing various access patterns, Pseudo-Randomly Interleaved Memory (PRIM) [21] can be adopted to reduce the imbalance in cache bank utilization caused by these access patterns. Instead of only using the lower bits as the bank ID like SIM, PRIM uses a pseudo-random algorithm with all or a significant portion of the block address to generate the bank ID.

#### 4.1.2 Two Pseudo-Random Algorithms for PRIM

As shown in [21], one approach is based on the Irreducible Polynomial (I-Poly) in the Galois Field (GF). Each GF is composed of a finite set of numbers, and it defines special rules for multiplication, addition, subtraction, and division. For I-Poly, each block address,  $A = \langle a_{n-1}, \dots, a_1, a_0 \rangle$  in bits, can represent a unique polynomial  $A(x) = a_{n-1} \times x^{n-1} + \dots + a_1 \times x^1 + a_0 \times x^0$ . Then, we take polynomial arithmetic modulo in  $GF(2^n)$  (where 2 is the modulo of this GF, and n is the maximum degree of polynomial) for the polynomial represented by the block address with a special divisor. In  $GF(2^n)$ , addition is equivalent to XOR, and multiplication is equivalent to AND. After computing the resulting polynomial, we convert it back to an integer, which becomes the bank ID for this block address.

Here is an example. Block address 37 ( $100101_2$ ) can represent polynomial  $(x^5 + x^2 + x^0)$ . If we choose 5 ( $101_2$ ) as the divisor, then the operation would be  $(x^5 + x^2 + x^0) \bmod (x^2 + x^0)$  with polynomial arithmetic modulo in  $GF(2)$ . This computation can be conducted with a series of XOR operations using the original binary values. We align the highest bit 1 of the divisor with that of the block address and perform an XOR. We repeat this step until the result contains only m bits (2 in this example):

$$100101_2 \oplus 101000_2 = 1101_2$$

$$1101_2 \oplus 1010_2 = 111_2$$

$$111_2 \oplus 101_2 = 10_2$$

Finally, 2 ( $10_2$ ) will be the bank ID for block address 37.

As pointed out in [21], the key to the success of this algorithm is to pick an appropriate divisor for the modulo operation. The best candidates are irreducible polynomials in  $\text{GF}(2^n)$  that are not divisible by any other non-zero order polynomials in  $\text{GF}(2^n)$ . To serve  $2^m$  cache banks, we must choose irreducible polynomials with an order of  $m$  as the divisor to ensure that the resulting bank ID has  $m$  remaining bits.

Another approach is based on H-Matrix [21]. H-Matrix is a carefully designed boolean matrix of size  $(m, n)$ . For each memory access, it computes the vector-matrix product of the block address and the H-Matrix, where multiplication and addition are replaced with AND and XOR respectively. The  $m$ -bit result is the bank ID of that memory access.

## 4.2 Memory Channel Interleaving

The cache bank selection policies above also enable memory channel interleaving, such that each cache bank also directly corresponds to a memory channel. Given  $n$  bits of block address and  $m$  bits of bank ID, each memory channel will use an address of  $(n - m)$  bits. The only condition to enable memory channel interleaving is that no two accesses, with different block addresses but the same memory channel address, are routed to the same cache bank. In other words, if we ensure that given the same higher  $(n - m)$  bits of block address, the mapping between lower  $m$  bits of block address and bank ID is a bijection, then we can use the higher  $(n - m)$  bits as the memory channel address.

For SIM, since the bank ID uses lower  $m$  bits of block address directly, the mapping is a bijection, apparently.

For I-Poly-based PRIM, we can replace lower  $m$  bits of block address with zeros and take a modulo of the modified polynomial, which will produce intermediate  $m$  bits to be XORed with the original lower  $m$  bits to result in the bank ID. When the higher  $(n - m)$  bits stay the same, the intermediate  $m$  bits should be the same. Since XOR with a constant is a bijective operation, the final output bank ID will be unique for distinctive lower  $m$  bits of block address.

For H-Matrix-based PRIM, it imposes a constraint on the H-Matrix selection that the lower  $m \times m$  partition of H-Matrix must be non-singular. Therefore, when the lower  $m$  bits of block address are modified, the bank ID resulting from matrix multiplication will be different.

### 4.3 Load/Store Queue (LSQ)

There is a LSQ on the input port of each cache bank, which buffers accesses from the routing unit. A cache bank can only accept one memory access request at each clock cycle. When multiple execution chiplets generate memory accesses to the same cache bank at the same clock cycle, the extra accesses must be buffered. Otherwise, we need to stall all execution chiplets until the accesses are handled by the cache bank, to ensure the strict timing of instructions. We only stall when any LSQ is unable to hold accesses in the next clock cycle. For our weakly ordered memory model, the LSQ allows bypassing values of stores in the queue to younger loads.

### 4.4 Non-Blocking Cache

While the blocking cache will trigger a stall immediately at a cache miss, the non-blocking cache allows execution to continue until an instruction uses a value that was missed but remains unresolved. The purpose is to overlap program execution with cache miss penalty as much as possible and hope the cache miss is resolved before the program needs to use the data.

To implement a non-blocking cache, we need a data structure called Miss Status Handling Register (MSHR). To reduce hardware complexity, we adopt a partitioned variant of MSHR [23]. Each MSHR entry holds the address and status of the missed block, and it also contains a table of load/store entries for subsequent accesses to the same block, which records the instruction type, block offset, and destination. At a cache miss, it first checks if there exists an MSHR entry for that block. If it exists, it only allocates a load/store entry

in the found MSHR. Otherwise, it allocates a new MSHR entry and also a load/store entry for the access. If it fails to allocate a new MSHR entry or a load/store entry in an existing MSHR, it stops taking new accesses.

Once a block is returned from the next level in the cache hierarchy, it compares the block address with the ones in MSHR entries. At the matching MSHR, it forwards the data to the output queue or modifies it according to the instruction types of the load/store entries in the order in which they arrived. When it finishes all load/store entries, it writes the block in the cache and deallocates the MSHR entry.

#### **4.5 Cache Organization with a SCE Memory Primitive**

Our cache will take the banked cache approach, and each cache bank will take its own chiplet. Through our industry partner, we know that a  $10mm \times 10mm$  chiplet can hold a 4-KiB tag and data store unit with 3-cycle pipelined array access and other wires. This will be our primitive to build our cache bank. The cache chiplet size for Syndra will be expected to be  $20mm \times 20mm$ , which can hold at least 2 4-KiB primitives and wires to manage routing and other work. So our basic cache organizations include 8-KiB direct-mapped and 8-KiB 2-way set-associative. Each of them will have an access time of 4 cycles, as shown in Figure 4.1 and Figure 4.2, and the total L1 latency will be 8 cycles at 5 GHz counting the communication latency between chiplets. While 2-way set-associative cache requires one more tag comparator, it allows for a higher associativity that is more tolerant to index conflicts, therefore potentially lowering the miss ratio and improving the performance.

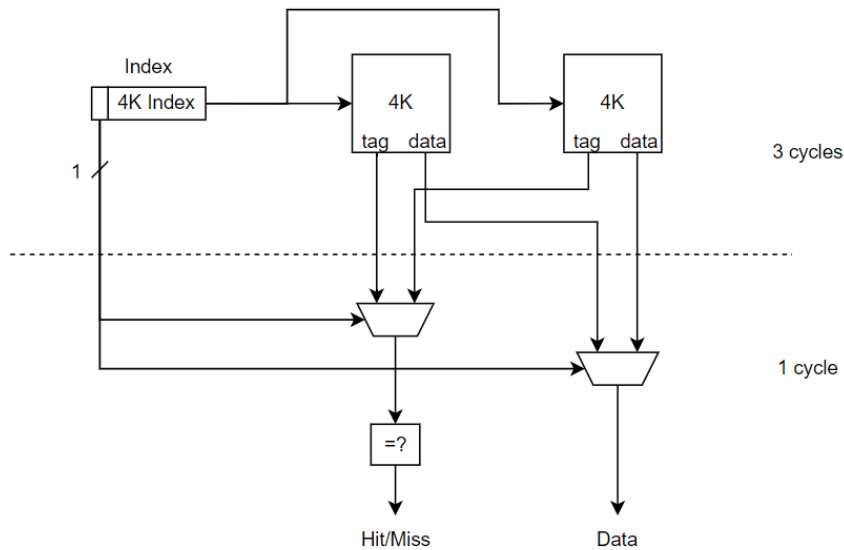


Figure 4.1: A circuit design of an 8-KiB direct-mapped cache based on 2 4-KiB primitives. It uses the lower 12 index bits to read tag and data from the 4-KiB primitives, and selects which tag/data to be used based on the highest bit. Then, it will compare the tag and determine if the access is a hit or miss. It allows for 3-cycle pipelined array access and 1-cycle tag comparison and word select, summing up to 4-cycle access time.

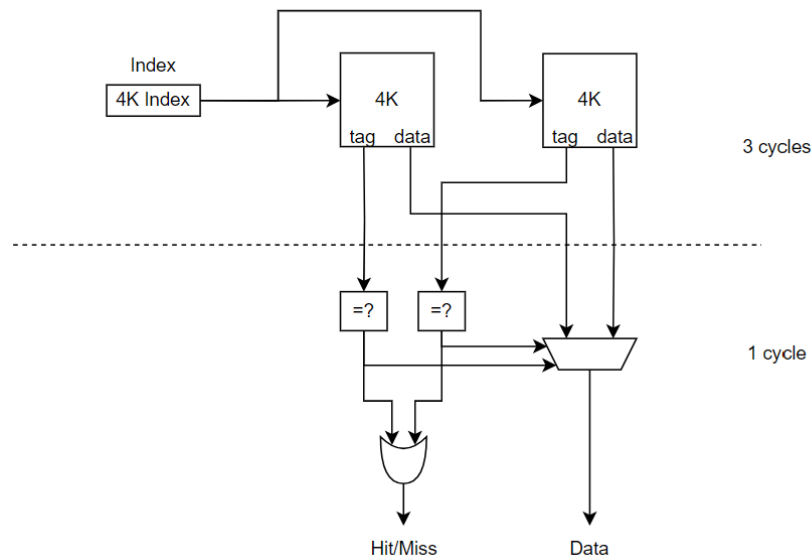


Figure 4.2: A circuit design of an 8-KiB 2-way set-associative cache based on 2 4-KiB primitives. It reads tag and data from two primitives and performs tag comparison in parallel. Then, it uses the tag comparison results to select data with a one-hot mux. The tag comparison results will be ORed to determine if the access is a hit or miss. It allows for 3-cycle pipelined array access and 1-cycle tag comparison and word select, summing up to 4-cycle access time.

## CHAPTER 5

### ADVANCED MEMORY TECHNOLOGIES

#### 5.1 Possibility for Larger Cache Size on $20 \times 20$ Chiplet

It might be possible to have more than 2 4-KiB cache primitives on our  $20 \times 20$  chiplet. The  $10 \times 10$  chiplet has a 1mm perimeter that is unused due to area requirements for testing and wiring, so only  $64mm^2$  are used for the 4-KiB cache primitive.  $10 \times 10$  is used due to constraints on the mask for memory test chiplets but not fundamental fabrication constraints. For the  $20 \times 20$  chiplet, when accounting for the 1mm perimeter, it gives us an effective chiplet area of  $324mm^2$ , which might hold 3 or 4 cache primitives including extra wiring area cost. Optimistically, each cache bank can have a cache size of 16 KiB with an access time of 5 cycles (1 more cycle than 8-KiB caches due to the larger mux).

#### 5.2 L2 Cache Organization

Since all cache bank selection policies above support memory channel interleaving, we can implement a 512-KiB 8-way set-associative L2 cache per L1 cache bank, which is logically equivalent to a unified 2-MiB 8-way set-associative 4-ported cache.

L2 cache will be a traditional blocking cache. Given that L1 cache banks are non-blocking, only one access to the missed block in L1 cache bank will propagate to L2 cache, while subsequent accesses will be handled by the same MSHR entry, so L1 cache banks will not be blocked by L2 cache. Therefore, making L2 cache non-blocking will not substantially contribute to performance. As mentioned in chapter 3, we can use a 3T-eDRAM Cryocache configuration to build our L2 cache, which will be deployed at a temperature of 77K. Alternatively, we can place CMOS-based L2 cache at the room temperature of 300K. Since the two options differ in access time and communication latencies at temperature

barriers, we must compare their effects on the performance before we can make a decision.

### **5.3 DRAM Organization and Interconnect Latency**

Our simulated cryogenic environment uses DDR3 based DRAM at room temperature, which has an expected latency of approximately 100 ns. This DDR3 memory is connected to an FPGA that interfaces with logic in the cryostat over a SerDes connection with a bandwidth of 10 GB/sec and a latency of 31 ns. This means that the round trip time from an L2 miss to a DRAM response is about 162 ns. Inside the cryostat, we support 2 thermal domains at 4 Kelvin and 77 Kelvin. It is expected that the latency between these 2 domains is close to 8 ns, but at worst will be 31 ns. Thus, the round trip time from an L1 miss to a L2 hit response is either 16 ns or 62 ns plus the latency of the L2 access itself.

## **CHAPTER 6**

### **METHODOLOGY**

This chapter introduces the research methodology, including an overview of the innovative superconducting architecture design, description of the simulation framework, identification of workloads of interest, and the evaluation criteria to be employed for comparison.

#### **6.1 Out-of-Order (OOO) Execution Simulator**

Since the compiler for SYNDRA is still under development, we build an OOO simulator based on Tomasulo's algorithm as a idealistic model for parallel execution and demonstrate the effects of different memory systems on performance. Similar to the role of SYNDRA compiler, the OOO scheduling queue is able to exploit the ideal instruction-level parallelism in a small window. To make it more similar to the behavior of SYNDRA, several modifications are applied to the original design. The single multi-ported cache is replaced with a routing unit and banked cache. Meanwhile, the function units are made capable of performing all operations but can only issue one instruction to each in a clock cycle, in order to mimic the behavior of SYNDRA execution chiplets. Moreover, the scheduling window is modified such that only a window of 16 trace instructions at a time can be considered for OOO scheduling. This is different from the size of the scheduling queue, which only determines the total number of instructions that can be considered for OOO execution, but not their proximity to each other. To capture this significant constraint in VLIW scheduling, we ensure that an instruction cannot be dispatched if it is 16 trace instructions away from the oldest instruction in the scheduling queue.

Table 6.1: Simulation Configurations for SCE

Parameters	Possible Options		
Superscalar Width	4		
Number of L1 Cache Banks	1	4	
L1 Cache Bank Size	32 KiB	8 KiB	
L1 Cache Organization	2-Way	Direct Mapped	2-Way
L1 Cache Bank Ports	1		
L1 Latency	3		
Cache Bank Selection Policy	SIM, I-Poly, H-Matrix		
LSQ Size	1, 4, 8, 16		
MSHR Size $\times$ L/S Entries per MSHR	$1 \times 1$ , $1 \times 4$ , $2 \times 4$		
L2 Cache Size	Disabled, 512 KiB 1-Ported per Bank, 2 MiB 4-Ported Unified		
L2 Cache Organization	8-Way		
L2 Latency	30, 105		
DRAM Latency	270		

Table 6.2: Simulation Configurations for CMOS

Parameters	Possible Options		
Superscalar Width	4		
Number of L1 Cache Banks	1	4	
L1 Cache Bank Size	32 KiB	8 KiB	
L1 Cache Organization	2-Way	Direct Mapped	2-Way
L1 Cache Bank Ports	4	1	
L1 Latency	2		
Cache Bank Selection Policy	SIM, I-Poly, H-Matrix		
LSQ Size	1		
MSHR Size $\times$ L/S Entries per MSHR	$1 \times 1$		
L2 Cache Size	512 KiB 4-Ported Unified		
L2 Cache Organization	8-Way		
L2 Latency	12		
DRAM Latency	165		

## 6.2 Simulation Configurations

All simulation configurations for SCE are listed in Table 6.1. To study the effects of each component on the performance of overall design, we set other configurations to minimum when focusing on a single parameter. The target architecture uses a 3-phase clock, and thus an instruction is issued every 3 cycles at 5 GHz. In order to facilitate that in the simulator,

we set the memory latency to the number of cycles for this phased clock (1.67 GHz) and also set the simulation clock frequency accordingly. Also, we will make a comparison between cache configurations in CMOS at room temperature (Table 6.2) to understand how a multi-ported cache performs compared to a banked cache with the same total cache size.

### **6.3 Workloads and Evaluation Criteria**

The preliminary target workloads of this project include digital signal processing and neuromorphic computing (linear algebra). They are also the workloads targeted by SYNDRA. Our benchmarks include Linpack, Finite Impulse Response, Tiled Matrix Multiplication, Streamed Matrix Multiplication, and Triangular Solver.

For performance metrics, we will base our evaluation and analysis on the IPC, demonstrated as speedup over a baseline in each experiment. While not ideal indicators for performance in a superscalar simulation, cache miss ratio and bank conflicts can have substantial impact in VLIW architectures due to static scheduling. We refer to the percentage of accesses that are blocked from being issued due to bank conflicts as the bank conflict ratio, which allows us to relate this metric between benchmarks. As such, we will show both miss ratio and bank conflict ratio when exploring L1 cache configurations. The main concern for this work is to evaluate the performance of the memory system and prevent it from becoming a bottleneck, so its EDP is not evaluated. Therefore, IPC is chosen as the primary indicator, and cache miss ratio and bank conflict ratio also help demonstrate the effectiveness of cache organizations and cache bank selection policies.

## CHAPTER 7

### EXPERIMENTS AND RESULTS

#### 7.1 Behavior of Banked Cache in CMOS

In CMOS, the multi-ported cache can satisfy the memory bandwidth for most applications at a low hardware cost, and therefore the banked cache design is not necessary and might even reduce performance due to bank conflicts. As shown in Figure 7.1, a unified 32-KiB 8-way set-associative 4-ported cache outperforms the other two banked cache design with different cache organizations.

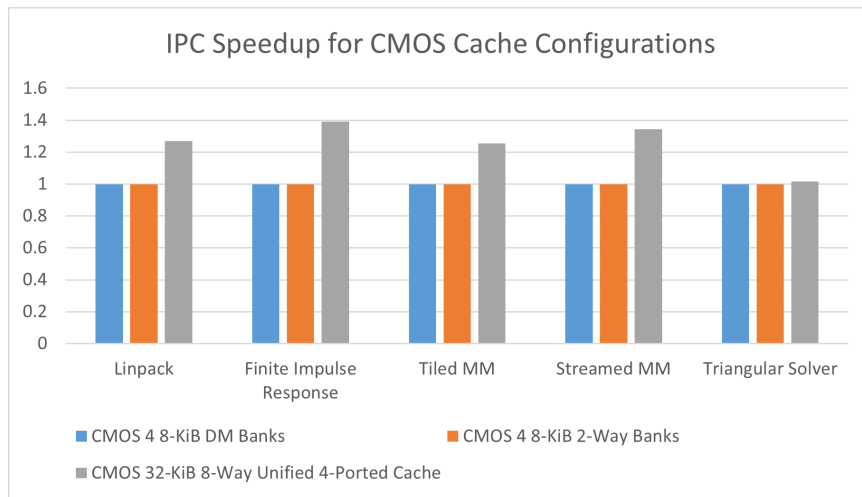


Figure 7.1: This bar graph demonstrates the IPC speedup of 4-ported cache over the banked cache design with similar total size in CMOS.

However, a multi-ported cache is infeasible in superconducting logic families, so we must study the impact of various banked cache configurations and other techniques on the performance.

## 7.2 Impact of L2 Cache Configurations

To gauge the impact of the L2 cache, we compare the performance of a unified 32-KiB 2-way set-associative L1 cache with no L2, and a 2-MiB 8-way set-associative L2 cache with high/low latency configurations.

To understand its impact on the banked cache, we also have a similar setup for 4x8-KiB 2-way set-associative L1 cache banks using SIM and evaluate its performance with no L2, 512-KiB 8-way set-associative L2 cache per bank, and a unified 2-MiB 8-way set-associative L2 cache. The two L2 latencies are also considered.

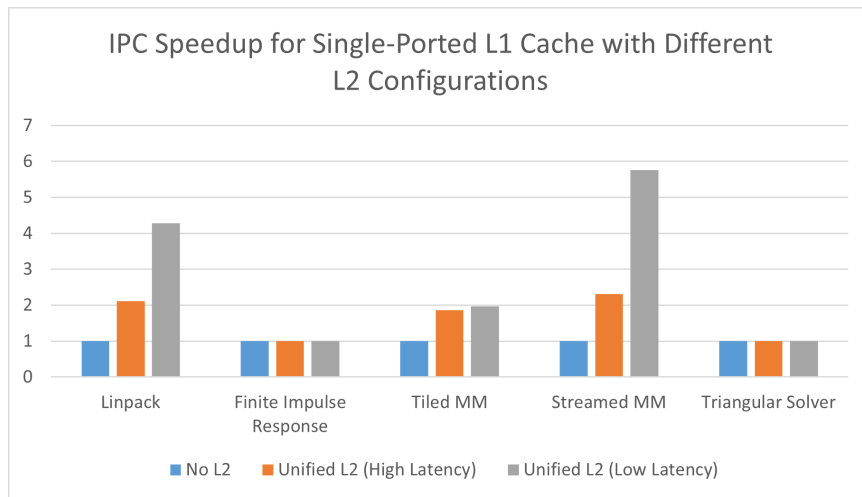


Figure 7.2: This bar graph demonstrates the IPC speedup of L2 cache configurations over the baseline where L2 is disabled for a unified 32-KiB 2-way set-associative L1 cache.

The result in Figure 7.2 and Figure 7.3 shows that the existence of L2 and its latency have a significant influence on the performance of applications whose working set is larger than the size of L1. As cache bank selection policy is deterministic, the L2 utilization of each bank is the same when compared to a unified L2, as shown in Figure 7.3.

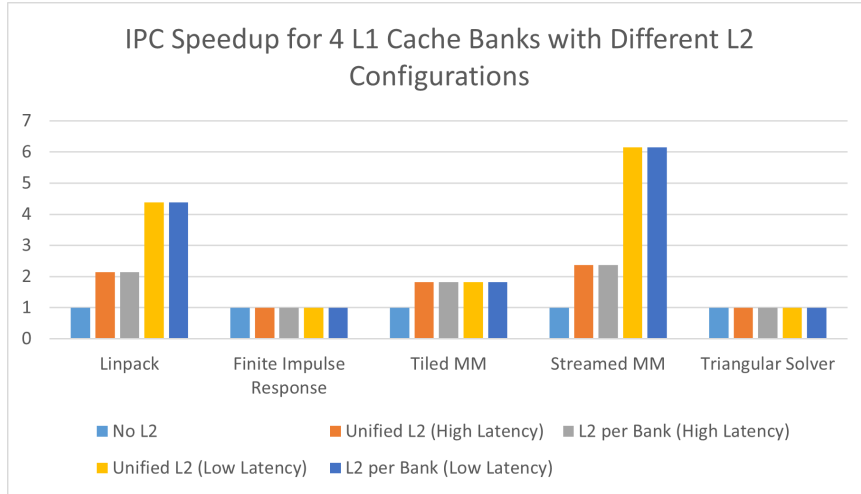


Figure 7.3: This bar graph demonstrates the IPC speedup of L2 cache configurations over the baseline where L2 is disabled for 4x8-KiB 2-way set-associative L1 cache banks.

### 7.3 Impact of Cache Bank Configurations and Selection Policy

We evaluate the performance of 2 cache organizations: 4x8-KiB direct-mapped cache banks and 4x8-KiB 2-way set-associative cache banks. For each configuration, we test 3 cache bank selection policies (SIM, I-Poly, and H-Matrix) independently. The experiments are conducted with CMOS memory hierarchy latency, high L2 latency (SCE), and low L2 latency (SCE) respectively.

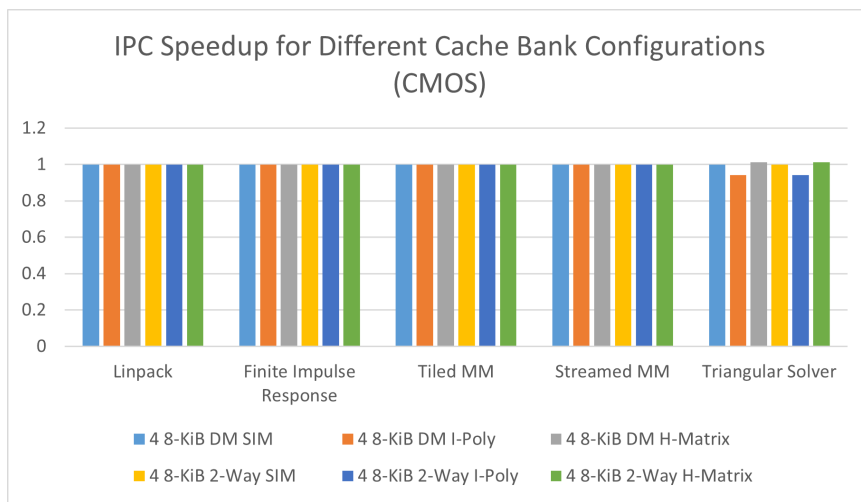


Figure 7.4: This bar graph demonstrates the IPC speedup of L1 cache configurations over the baseline of 4x8-KiB direct-mapped cache banks with CMOS memory hierarchy latency.

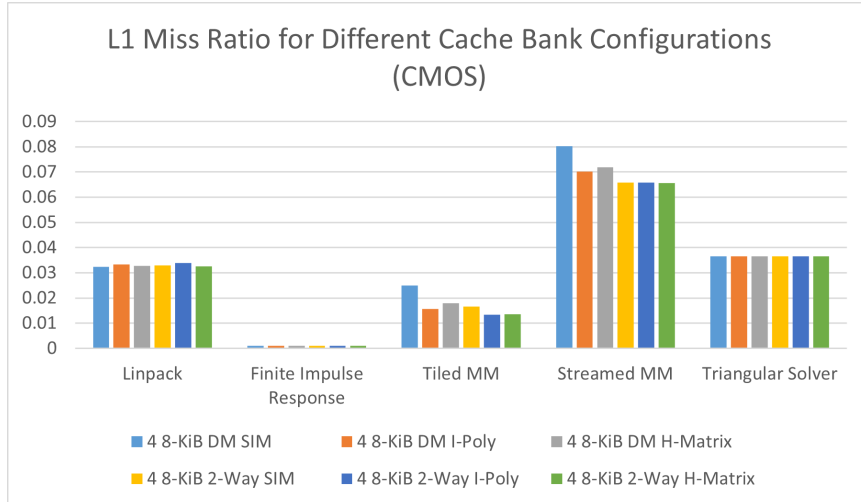


Figure 7.5: This bar graph demonstrates the L1 miss ratios for L1 cache configurations with CMOS memory hierarchy latency.

As shown in Figure 7.4, cache organization and cache bank selection policy have almost no influence on the performance of each benchmark with CMOS memory hierarchy latency, while Figure 7.5 indeed demonstrates a lower miss ratio of 2-way set-associative cache in general. This could be explained by the relatively small L2 latency hiding the effect of L1 misses on the performance.

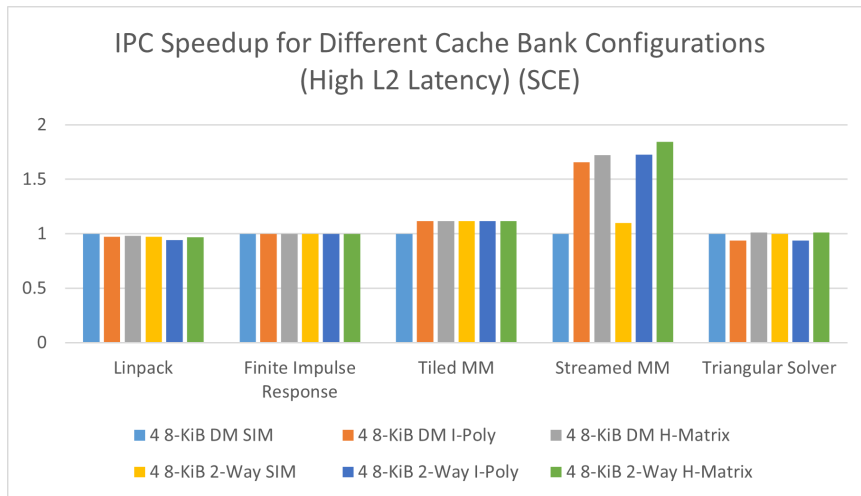


Figure 7.6: This bar graph demonstrates the IPC speedup of L1 cache configurations over the baseline of 4x8-KiB direct-mapped cache banks with high SCE L2 latency.

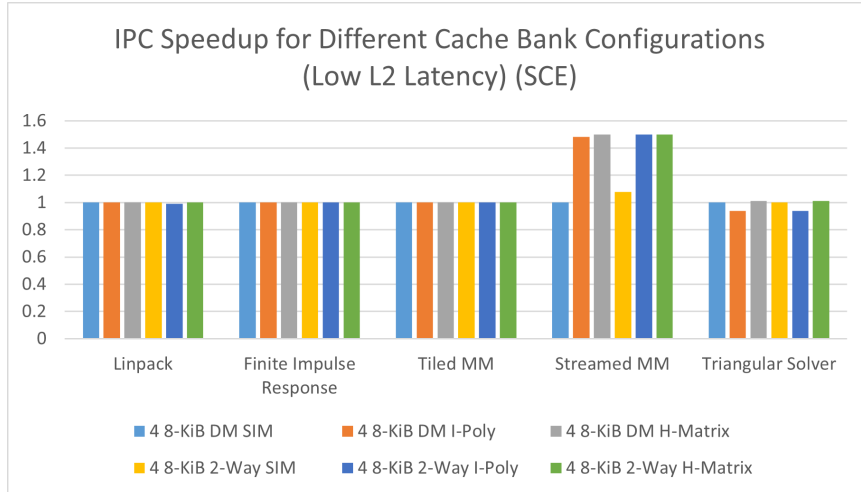


Figure 7.7: This bar graph demonstrates the IPC speedup of L1 cache configurations over the baseline of 4x8-KiB direct-mapped cache banks with low L2 latency in SCE.

With higher L2 latencies in SCE, the effect of L1 misses is more distinctive. For benchmarks that exhibit strongly strided memory access patterns, like Streamed MM, 2-way set-associative cache performs better than direct-mapped cache, and I-Poly and H-Matrix are more robust than SIM, with Figure 7.6 and Figure 7.7 showing an IPC speedup greater than 1, and Figure 7.8 and Figure 7.9 showing a decreased miss ratio.

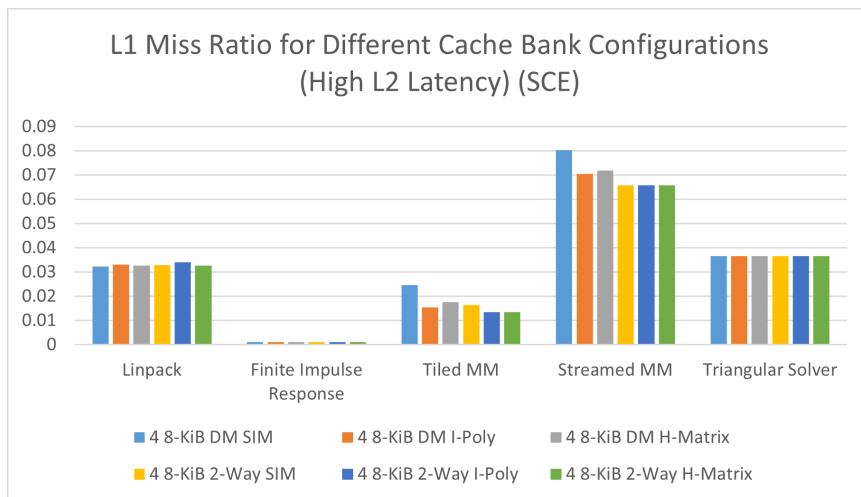


Figure 7.8: This bar graph demonstrates the L1 miss ratios for L1 cache configurations with high SCE L2 latency.

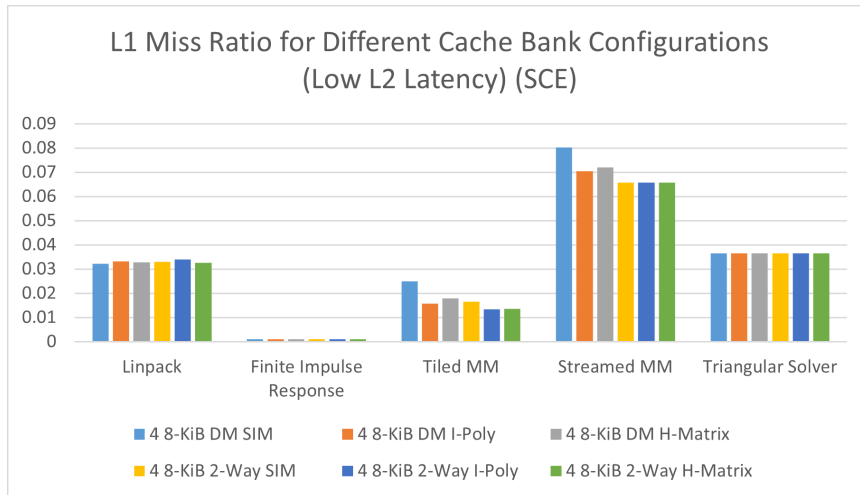


Figure 7.9: This bar graph demonstrates the L1 miss ratios for L1 cache configurations with low L2 latency in SCE.

The bank conflict ratios shown in Figure 7.10 and Figure 7.11 are strong evidence to indicate the reduced bank conflicts using PRIM on benchmarks with strided access patterns. The speedup on IPC is more noticeable with increased L2 latency.

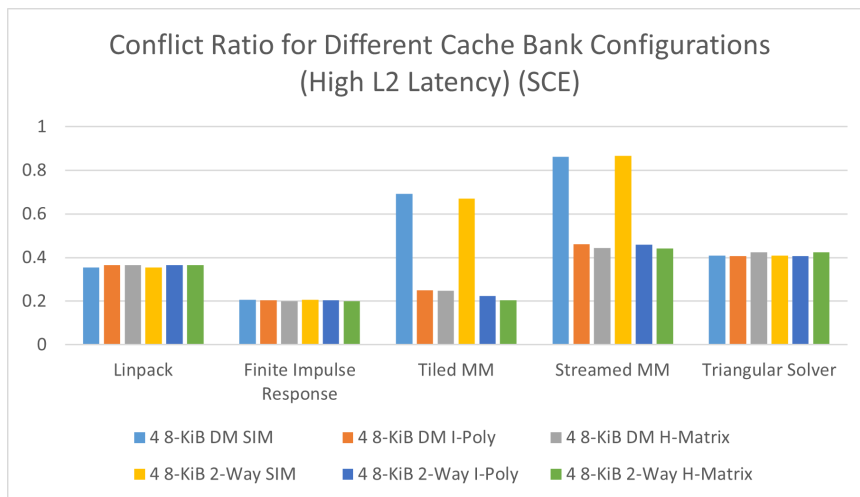


Figure 7.10: This bar graph demonstrates the bank conflict ratios for L1 cache configurations with high L2 latency in SCE.

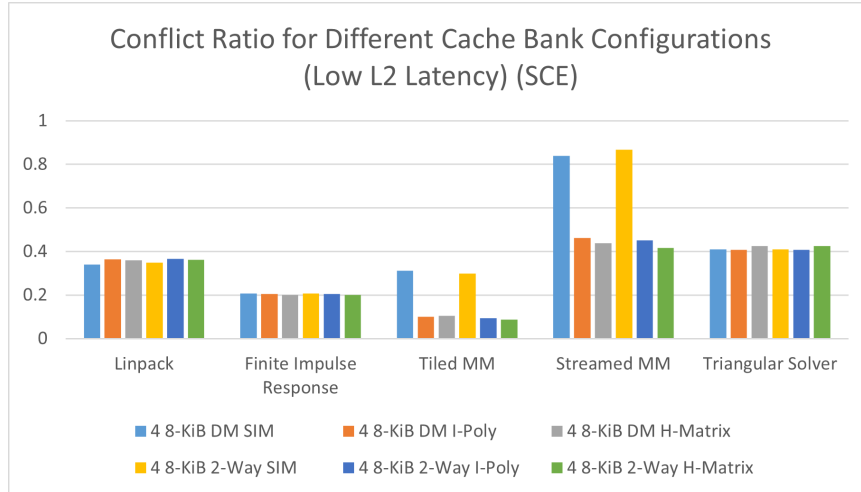


Figure 7.11: This bar graph demonstrates the bank conflict ratios for L1 cache configurations with low L2 latency in SCE.

We also observe that I-Poly doesn't help with Triangular Solver. This benchmark has a small loop with a loop-carried dependence, and thus there is less opportunity for memory access parallelism to provide speedup. It is likely that the small advantage from the H-Matrix approach is due to a reduction in conflicts within a single loop iteration.

Given that we have evaluated the impacts of L2 latency and cache organization, to limit the search space of the following experiments, we will focus on low L2 latency and 4x8-KiB 2-way set-associative cache banks. Meanwhile, I-Poly and H-Matrix share a similar behavior facing strided memory access patterns, but the hardware overhead to implement I-Poly leads to much higher latency than H-Matrix, so we will drop I-Poly in the experiments below.

#### 7.4 Impact of Load/Store Queue (LSQ) Size

In this section, we examine the influence of LSQ size on the previously determined optimal cache organization (4x8-KiB 2-way set-associative cache banks) using SIM or H-Matrix with low L2 latency. We will explore the influence of varying LSQ sizes (1, 4, 8, and 16 entries) on IPC.

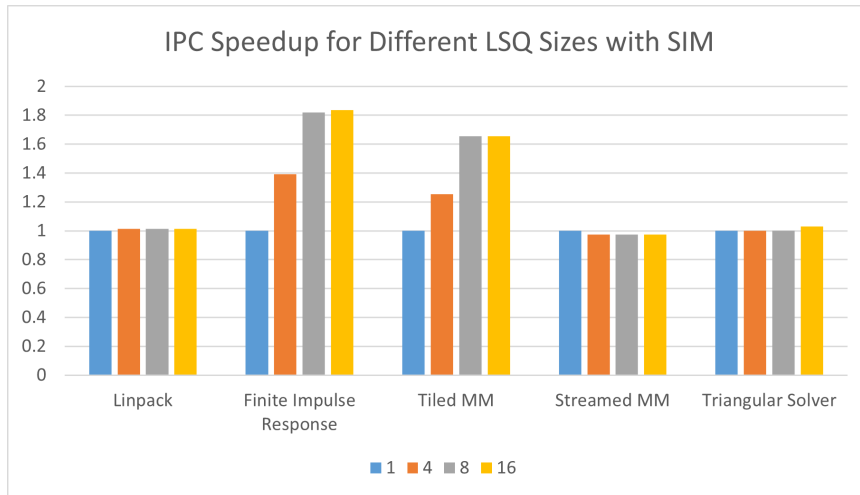


Figure 7.12: This bar graph demonstrates the IPC speedup of more LSQ entries over the baseline of 1 LSQ entry with SIM and low L2 latency.

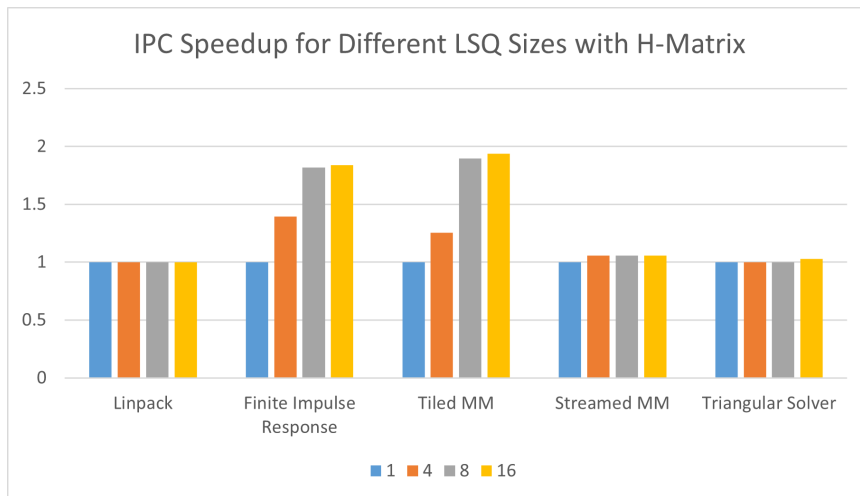


Figure 7.13: This bar graph demonstrates the IPC speedup of more LSQ entries over the baseline of 1 LSQ entry with H-Matrix and low L2 latency.

As shown in Figure 7.12 and Figure 7.13, with more LSQ entries, it allows for more chances of bypassing, therefore increasing the performance of some benchmarks. However, since we imposed a limitation on the proximity in instruction scheduling window, 16 LSQ entries cannot result in more speedup than 8 LSQ entries.

## 7.5 Impact of MSHR and the Non-Blocking Mechanism

For this experiment, we show the speedup of  $1 \text{ MSHR} \times 4$  load/store entries and  $2 \times 4$  over the baseline  $1 \times 1$  to evaluate the impact of different MSHR setups on the performance. Other configurations are the same as the previous section: 4x8-KiB 2-way set-associative cache banks, SIM/H-Matrix, and low L2 latency.

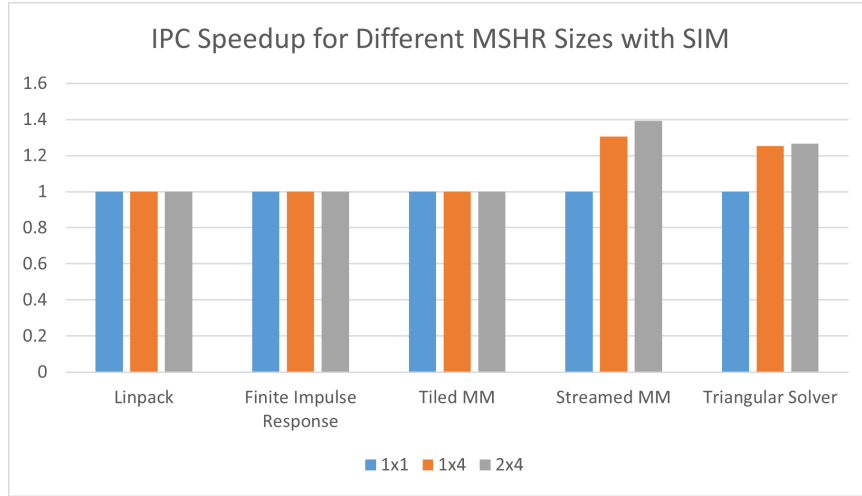


Figure 7.14: This bar graph demonstrates the IPC speedup of more MSHR entries over the baseline of  $1 \text{ MSHR} \times 1$  load/store entry with SIM and low L2 latency.

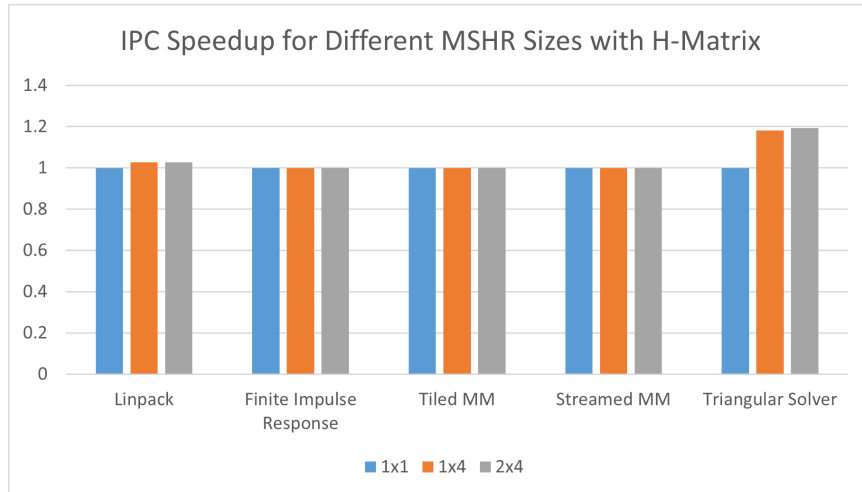


Figure 7.15: This bar graph demonstrates the IPC speedup of more MSHR entries over the baseline of  $1 \text{ MSHR} \times 1$  load/store entry with H-Matrix and low L2 latency.

Figure 7.14 and Figure 7.15 illustrate an improvement in IPC with more MSHR entries on some benchmarks. Similar to the effect of LSQ size, the small scheduling window size limits the full potential of MSHR, which can allow following independent instructions to be issued. This will result in a greater impact on the SYNDRA architecture with its VLIW compiler.

## 7.6 Comparison between Baseline, Optimal, and Larger Cache for $20 \times 20$ Chiplets

In this section, we will determine the overall speedup of our optimal configurations from all experiments above over the baseline. Our optimal configuration is 4x8-KiB 2-way set-associative cache banks using H-Matrix with low L2 latency, 16 LSQ entries, 2 MSHR  $\times$  4 load/store entries. The baseline configuration is 4x8-KiB direct-mapped cache banks using SIM with low L2 latency, 1 LSQ entry, 1 MSHR  $\times$  1 load/store entry, which is the banked cache design with the minimum hardware complexity.

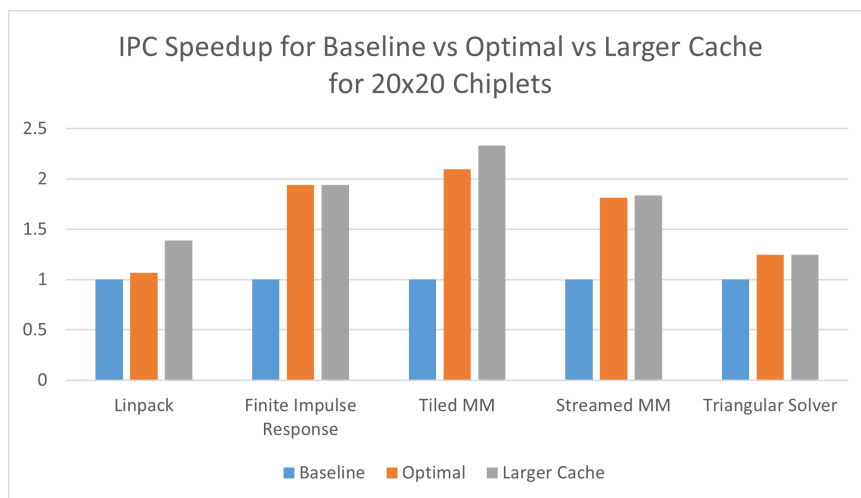


Figure 7.16: This bar graph demonstrates the IPC speedup of the optimal configuration (4x8-KiB 2-way set-associative cache banks using H-Matrix with low L2 latency, 16 LSQ entry, 2 MSHR  $\times$  4 load/store entries) and that with potentially larger cache size for  $20 \times 20$  chiplets over the baseline of 4x8-KiB direct-mapped cache banks using SIM with low L2 latency, 1 LSQ entry, 1 MSHR  $\times$  1 load/store entry.

To explore the potential effect of 20x20 chiplets with a larger cache size (16 KiB per bank), we set up the experiment with the previously identified optimal configuration, except for a larger cache bank size.

Figure 7.16 demonstrates that the optimal configuration has significant speedup over the baseline across most benchmarks, and a larger cache size on  $20 \times 20$  chiplets can further improve the performance.

## **CHAPTER 8**

### **CONCLUSION**

#### **8.1 Summary**

This research project underscores the imperative of optimizing superconducting logic architectures in light of inherent limitations such as circuit density and fanout constraints. The focus revolves around addressing the challenges associated with the lack of multiported caches in superconducting logic families through finding alternatives for parallel memory accesses to mitigate potential bottlenecks. Banked caching is proposed as a solution, where the memory address space is partitioned, allowing parallel access to different memory sections. Different cache bank selection policies, SIM and PRIM, are compared. PRIM emerges as a promising technique to enhance robustness against various access patterns. The overarching goal is to harness the advantages of superconducting technologies, such as energy efficiency and faster operating speeds, while navigating the challenges through innovative strategies like cache banks and PRIM. This approach seeks to demonstrate a pathway to minimize the negative influence of the constraints of superconducting logic families.

#### **8.2 Advantages and Limitations**

Utilizing cache banks in conjunction with PRIM increases the bandwidth for simultaneous memory accesses under the limitations of superconducting logic families, while ensuring a consistent access time regardless of diverse access patterns within targeted workloads. Along with other techniques like LSQ and MSHR, the optimal design achieves 1.6x speedup on average over the baseline design in our benchmarks. Nevertheless, it is essential to acknowledge that the hardware overhead associated with PRIM could potentially

introduce latency, and I-Poly approach would have a significantly longer critical path than H-Matrix approach due to the hardware complexity to implement the algorithm. The long latency could be mitigated by using fewer bits in the block address to generate the bank ID at a cost of robustness against various access patterns.

### **8.3 Future Work**

Exploring an alternative cache bank selection policy distinct from PRIM based on I-Poly/H-Matrix could potentially yield improved robustness while simultaneously mitigating hardware overhead concerns. To rigorously assess and compare the effectiveness of different cache bank selection policies, comprehensive experiments will be conducted leveraging the SYNDRA simulator, which simulates an idealized execution model of SYNDRA, for evaluating the proposed cache bank selection policies. By examining these aspects, the research aims to provide valuable insights into optimizing memory systems for superconducting processors and advancing the understanding of their behavior under various configurations.

## REFERENCES

- [1] “Cryogenic Electronics and Quantum Information Processing,” in *2021 IEEE International Roadmap for Devices and Systems Outbriefs*, Nov. 2021, pp. 1–93.
- [2] B. D. Josephson, “Possible new effects in superconductive tunnelling,” *Physics Letters*, vol. 1, no. 7, pp. 251–253, Jul. 1962.
- [3] O. Mukhanov, V. Semenov, and K. Likharev, “Ultimate performance of the RSFQ logic circuits,” *IEEE Transactions on Magnetics*, vol. 23, no. 2, pp. 759–762, Mar. 1987, Conference Name: IEEE Transactions on Magnetics.
- [4] Q. P. Herr, A. Y. Herr, O. T. Oberg, and A. G. Ioannidis, *Ultra-Low-Power Superconductor Logic*, arXiv:1103.4269 [cond-mat], Mar. 2011.
- [5] W. Chen, A. Rylyakov, V. Patel, J. Lukens, and K. Likharev, “Rapid single flux quantum T-flip flop operating up to 770 GHz,” *IEEE Transactions on Applied Superconductivity*, vol. 9, no. 2, pp. 3212–3215, Jun. 1999, Conference Name: IEEE Transactions on Applied Superconductivity.
- [6] M. Volkmann, A. Sahu, C. Fourie, and O. Mukhanov, “Implementation of Energy Efficient Single Flux Quantum (eSFQ) Digital Circuits with sub-aJ/bit Operation,” *Superconductor Science and Technology*, vol. 26, Sep. 2012.
- [7] K. Likharev and V. Semenov, “RSFQ logic/memory family: A new Josephson-junction technology for sub-terahertz-clock-frequency digital systems,” *IEEE Transactions on Applied Superconductivity*, vol. 1, no. 1, pp. 3–28, Mar. 1991, Conference Name: IEEE Transactions on Applied Superconductivity.
- [8] M. Kuniyoshi *et al.*, “Investigation of Timing Parameters in Single-Flux-Quantum Circuits Using Low Critical-Current Junctions and Low Bias Voltages,” *IEEE Transactions on Applied Superconductivity*, vol. 31, no. 5, pp. 1–5, Aug. 2021, Conference Name: IEEE Transactions on Applied Superconductivity.
- [9] I. I. Soloviev, N. V. Klenov, S. V. Bakurskiy, M. Y. Kupriyanov, A. L. Gudkov, and A. S. Sidorenko, “Beyond Moore’s technologies: Operation principles of a superconductor alternative,” *Beilstein Journal of Nanotechnology*, vol. 8, pp. 2689–2710, Dec. 2017, arXiv:1706.09124 [cond-mat].
- [10] S. V. Rylov, “Clockless Dynamic SFQ and Gate With High Input Skew Tolerance,” *IEEE Transactions on Applied Superconductivity*, vol. 29, no. 5, pp. 1–5, Aug. 2019, Conference Name: IEEE Transactions on Applied Superconductivity.

- [11] X. Peng *et al.*, “High-Speed Demonstration of Bit-Serial Floating-Point Adders and Multipliers Using Single-Flux-Quantum Circuits,” *IEEE Transactions on Applied Superconductivity*, vol. 25, no. 3, pp. 1–6, Jun. 2015, Conference Name: IEEE Transactions on Applied Superconductivity.
- [12] TOP500, *Green500 Data*, Jun. 2023.
- [13] M. Dorojevets, Z. Chen, C. L. Ayala, and A. K. Kasperek, “Towards 32-bit Energy-Efficient Superconductor RQL Processors: The Cell-Level Design and Analysis of Key Processing and On-Chip Storage Units,” *IEEE Transactions on Applied Superconductivity*, vol. 25, no. 3, pp. 1–8, Jun. 2015, Conference Name: IEEE Transactions on Applied Superconductivity.
- [14] S. K. Tolpygo, “Superconductor Digital Electronics: Scalability and Energy Efficiency Issues,” *Low Temperature Physics*, vol. 42, no. 5, pp. 361–379, May 2016, arXiv:1602.03546 [cond-mat].
- [15] O. Chen *et al.*, “Adiabatic Quantum-Flux-Parametron: Towards Building Extremely Energy-Efficient Circuits and Systems,” *Scientific Reports*, vol. 9, no. 1, p. 10 514, Jul. 2019, Number: 1 Publisher: Nature Publishing Group.
- [16] S. S. Tannu, P. Das, M. L. Lewis, R. Krick, D. M. Carmean, and M. K. Qureshi, “A case for superconducting accelerators,” in *Proceedings of the 16th ACM International Conference on Computing Frontiers*, ser. CF ’19, New York, NY, USA: Association for Computing Machinery, 2019, pp. 67–75, ISBN: 978-1-4503-6685-4.
- [17] G. Pasandi, A. Shafaei, and M. Pedram, “SFQmap: A Technology Mapping Tool for Single Flux Quantum Logic Circuits,” in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, ISSN: 2379-447X, May 2018, pp. 1–5.
- [18] J. Volk, G. Tzimpragos, A. Wynn, E. Golden, and T. Sherwood, “Low-Cost Superconducting Fan-Out with Cell I<sub>S</sub> Ranking,” *IEEE Transactions on Applied Superconductivity*, vol. 33, no. 6, pp. 1–12, Sep. 2023, arXiv:2206.07817 [cs].
- [19] J. A. Fisher, “Very Long Instruction Word architectures and the ELI-512,” in *Proceedings of the 10th annual international symposium on Computer architecture*, ser. ISCA ’83, New York, NY, USA: Association for Computing Machinery, 1983, pp. 140–150, ISBN: 978-0-89791-101-6.
- [20] A. Veen, “Dataflow Machine Architecture.,” *ACM Comput. Surv.*, vol. 18, pp. 365–396, Dec. 1986.
- [21] B. R. Rau, “Pseudo-randomly interleaved memory,” in *Proceedings of the 18th annual international symposium on Computer architecture*, ser. ISCA ’91, New York,

NY, USA: Association for Computing Machinery, 1991, pp. 74–83, ISBN: 978-0-89791-394-2.

- [22] D. Min, I. Byun, G.-H. Lee, S. Na, and J. Kim, “CryoCache: A Fast, Large, and Cost-Effective Cache Architecture for Cryogenic Computing,” in *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS ’20, New York, NY, USA: Association for Computing Machinery, 2020, pp. 449–464, ISBN: 978-1-4503-7102-5.
- [23] D. Kroft, “Lockup-free instruction fetch/prefetch cache organization,” in *Proceedings of the 8th annual symposium on Computer Architecture*, ser. ISCA ’81, Washington, DC, USA: IEEE Computer Society Press, 1981, pp. 81–87.