

Stony Brook University



OFFICIAL COPY

The official electronic file of this thesis or dissertation is maintained by the University Libraries on behalf of The Graduate School at Stony Brook University.

© All Rights Reserved by Author.

Energy-Efficient Wide Datapath Integer Arithmetic Logic Units Using Superconductor Logic

A Dissertation Presented

by

Christopher Lawrence Ayala

to

The Graduate School

in Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

in

Computer Engineering

Stony Brook University

December 2012

Copyright by
Christopher Lawrence Ayala
2012

Stony Brook University

The Graduate School

Christopher Lawrence Ayala

We, the dissertation committee for the above candidate for the
Doctor of Philosophy degree, hereby recommend
acceptance of this dissertation.

Mikhail Dorojevets – Dissertation Advisor
Associate Professor, Department of Electrical and Computer Engineering

Sangjin Hong – Chairperson of Defense
Associate Professor, Department of Electrical and Computer Engineering

Emre Salman
Assistant Professor, Department of Electrical and Computer Engineering

Jennifer L. Wong
Assistant Professor, Department of Computer Science
Stony Brook University

This dissertation is accepted by the Graduate School.

Charles Taber
Interim Dean of the Graduate School

Abstract of the Dissertation

Energy-Efficient Wide Datapath Integer Arithmetic Logic Units Using Superconductor Logic

by

Christopher Lawrence Ayala

Doctor of Philosophy

in

Computer Engineering

Stony Brook University

2012

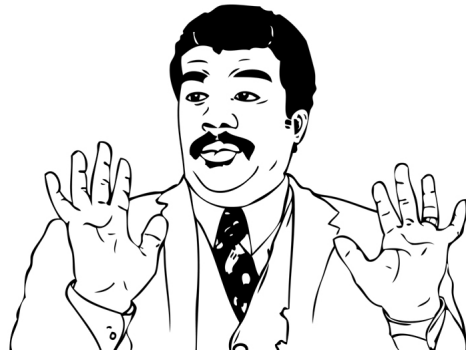
Complementary Metal-Oxide-Semiconductor (CMOS) technology is currently the most widely used integrated circuit technology today. As CMOS approaches the physical limitations of scaling, it is unclear whether or not it can provide long-term support for niche areas such as high-performance computing and telecommunication infrastructure, particularly with the emergence of cloud computing. Alternatively, superconductor technologies based on Josephson junction (JJ) switching elements such as Rapid Single Flux Quantum (RSFQ) logic and especially its new variant, Energy-Efficient Rapid Single Flux Quantum (ERSFQ) logic have the capability to provide an ultra-high-speed, low power platform for digital systems.

The objective of this research is to design and evaluate energy-efficient, high-speed 32-bit integer Arithmetic Logic Units (ALUs)

implemented using RSFQ and ERSFQ logic as the first steps towards achieving practical Very-Large-Scale-Integration (VLSI) complexity in digital superconductor electronics. First, a tunable VHDL superconductor cell library is created to provide a mechanism to conduct design exploration and evaluation of superconductor digital circuits from the perspectives of functionality, complexity, performance, and energy efficiency. Second, hybrid wave-pipelining techniques developed earlier for wide datapath RSFQ designs have been used for efficient arithmetic and logic circuit implementations. To develop the core foundation of the ALU, the ripple-carry adder and the Kogge-Stone parallel prefix carry look-ahead adder are studied as representative candidates on opposite ends of the design spectrum. By combining the high-performance features of the Kogge-Stone structure and the low complexity of the ripple-carry adder, a 32-bit asynchronous wave-pipelined hybrid sparse-tree ALU has been designed and evaluated using the VHDL cell library tuned to HYPRES' gate-level characteristics.

The designs and techniques from this research have been implemented using RSFQ logic and prototype chips have been fabricated. As a joint work with HYPRES, a 20 GHz 8-bit Kogge-Stone ALU consisting of 7,950 JJs total has been fabricated using a 1.5 μm 4.5 kA/cm² process and fully demonstrated. An 8-bit sparse-tree ALU (8,832 JJs total) and a 16-bit sparse-tree adder (12,785 JJs total) have also been fabricated using a 1.0 μm 10 kA/cm² process and demonstrated under collaboration with Yokohama National University and Nagoya University (Japan).

*Sa aking mga magulang.
To my parents.*



Watch out guys...

Contents

Abstract	iii
Contents	vii
List of Figures	x
List of Tables	xiv
Acknowledgments	xvi
Vita	xxi
Publications	xxii
1 Introduction	1
1.1 Motivation	1
1.1.1 CMOS Scaling Limits	1
1.1.2 Significance of Arithmetic Logic Units	3
1.1.3 Benefits and Opportunities in Superconductor Technology	4
1.2 Research Outline and Goals	6
1.3 Superconductor Logic	7
1.3.1 Latching Logic	7
1.3.2 Rapid Single Flux Quantum Logic	8
1.3.3 Energy-Efficient Rapid Single Flux Quantum Logic . .	12
1.3.4 General Challenges for Superconductor Technology . .	12
1.4 Overview of Prior Work in Superconductor Electronics	14
1.4.1 Adders and ALUs	14
1.4.1.1 “Push-Forward” RSFQ Carry-Save Serial Adders	14
1.4.1.2 Case Study of Fast Pipelined Parallel Adders in RSFQ	15
1.4.1.3 1-bit RSFQ ALU with a 3-Input XOR Gate .	15
1.4.1.4 4-bit RSFQ ALU with Half Adder Cells . . .	16

1.4.1.5	4-bit RSFQ Digit-Serial Adder	16
1.4.1.6	100 GHz RSFQ Bit-Serial Adder	17
1.4.2	Microprocessors	18
1.4.2.1	Fujitsu's 8-bit DSP Microprocessor	18
1.4.2.2	FLUX-1 Microprocessor	19
1.4.2.3	CORE1 Microprocessor	21
1.4.2.4	20 GHz 8-bit RSFQ Frontier Datapath	22
2	Development of Efficient Techniques for VLSI Superconductor Design	23
2.1	Superconductor Cell Library and Design Tools	23
2.1.1	SBU Tunable VHDL Cell Library	23
2.1.1.1	Purpose and Overview	23
2.1.1.2	Acknowledgments	27
2.1.2	CONNECT Cell Library	28
2.1.2.1	Purpose and Overview	28
2.1.2.2	Acknowledgments	30
2.1.3	Summary of RSFQ Logic Cells	32
2.2	Asynchronous Hybrid Wave-Pipelining	35
3	Superconductor Ripple-Carry Adder	41
3.1	Goals and Challenges	41
3.2	Ripple-Carry Adder Concept	42
3.3	RSFQ Study	43
3.3.1	Design Overview	43
3.3.2	Simulation Results and Discussion	46
4	Superconductor Kogge-Stone Adder and ALU	53
4.1	Goals and Challenges	53
4.2	General Kogge-Stone Adder Structure	54
4.3	RSFQ Study	58
4.3.1	Design Overview	58
4.3.2	Simulation Results and Discussion	65
4.4	Joint SBU-HYPRES Project: An 8-bit Kogge-Stone ALU Implementation Using the 1.5 μm 4.5 kA/cm ² HYPRES Process	73
4.4.1	Design Flow	73
4.4.2	Simulation Results	75
4.4.3	Low-Frequency Testing	77
4.4.4	High-Frequency Testing	81
4.4.4.1	High-Speed Input/Output Interfaces	81
4.4.4.2	High-Speed Testing Results	84

5	Superconductor Hybrid Sparse-Tree Adder and ALU	87
5.1	Goals and Challenges	87
5.2	Sparse-Tree Structure	88
5.3	RSFQ Study	90
5.3.1	Design Overview	90
5.3.2	Simulation Results	94
5.3.3	Discussion	101
5.4	Adder and ALU Design Implemented Using the CONNECT Cell Library for the 1.0 μm 10 kA/cm ² Process	105
5.4.1	Goals and Challenges	105
5.4.2	Simulation Results	114
5.4.2.1	16-bit HSTA Results	114
5.4.2.2	8-bit HSTALU Results	116
5.4.3	Experimental Testing	117
5.4.4	Chip Testing Results	122
5.4.4.1	16-bit HSTA	122
5.4.4.2	8-bit HSTALU	129
6	Conclusions	132
6.1	Completed Work	132
6.2	Future Work	134
	Bibliography	135

List of Figures

1.1	Clock rate and power for various Intel microprocessors.	2
1.2	Predicted US electricity use for data centers.	3
1.3	Thermal temperature maps of the execution cores in Intel microprocessors.	4
1.4	Views of the Josephson junction (JJ) superconductive device.	5
1.5	DC I-V characteristics of JJs.	8
1.6	JTL and PTL connections used in superconductor logic.	9
1.7	RSFQ D flip-flop.	11
1.8	Microphotographs of the CSSA.	14
1.9	Schematic of the 1-bit ALU slice based on a 3-input XOR.	15
1.10	4-bit RSFQ ALU built with half adder cells.	16
1.11	4-bit digit-serial adder.	17
1.12	100 GHz bit-serial adder.	18
1.13	Microphotograph of Fujitsu's 8-bit DSP based on latching logic.	19
1.14	The FLUX-1 8-bit RSFQ microprocessor	20
1.15	Microphotograph of CORE1 γ 8 mm ² chip.	22
2.1	Normal distribution of delays obtained from a DFF cell simulation.	24
2.2	DFF symbol used in schematics.	25
2.3	FSM of the DFF cell.	26
2.4	Logic simulation waveform of the DFF cell.	26
2.5	Cross section of the HYPRES standard Niobium process.	27
2.6	Cross section of the Japanese ISTEK Advanced Process (ADP 2.2).	29
2.7	Logic simulation in Cadence NC-Verilog.	29
2.8	Grid-based approach used in the CONNECT cell library.	30
2.9	Example of how the schematic and physical layout correspond to each other when designed using the CONNECT cell library. Note that PTL and via cells (pink and purple lines) are also placed at the schematic level as well.	31
2.10	CFF symbol used in schematics.	36

2.11	FSM of the CFF cell used in asynchronous wave-pipelining. . .	36
2.12	Logic simulation waveform of a CFF cell.	36
2.13	An example of asynchronous wave-pipelining with trailing reset waves.	38
2.14	Co-flow clocking versus wave-pipelining.	40
3.1	Example of a 4-bit RCA structure composed of 4 FA cells. The inset shows how one would implement an FA cell using CMOS logic gates.	43
3.2	Schematic of an RSFQ RCA adder.	44
3.3	T1 symbol used in schematics.	44
3.4	FSM of the T1 cell used in the RCA structure.	45
3.5	Logic simulation waveform of the T1 cell.	45
3.6	Latency distribution of the 32-bit RCA. The least significant bit is bit index 0. The latency is measured from the assertion of input signals to the arrival of outputs at the “sum” port. . .	47
3.7	Cell-wise breakdown of the 32-bit RCA for both the complexity and bias current of the logical design. The results do not include additional JJs to distribute bias current in ERSFQ logic. . . .	49
3.8	Categorical breakdown of the 32-bit RCA for both the complexity and bias current of the logical design.	50
3.9	Breakdown comparison of both raw and adjusted bias JJ counts with respect to the total design complexity of the 32-bit RCA implemented in ERSFQ logic.	51
4.1	Example of a 16-bit KSA structure.	57
4.2	INIT blocks which can be logically interchanged to obtain either an ALU (a) or an adder (b). These are the Red cells that reside in the Initialization stage.	62
4.3	Logical schematics for the prefix tree and summation blocks. The Green and Black cells of the Prefix Tree are built using CM blocks (a). The Gray and Blue cells are built using CM_BUFF blocks (b). The Orange cells in the Summation stage are built using SUM blocks (c).	63
4.4	Latency distribution of the 32-bit KSALU. The least significant bit is bit index 0. The latency is measured from the assertion of the “ready” signal to the arrival of outputs at the “sum” port. . .	68
4.5	Cell-wise breakdown of the 32-bit KSALU for both the complexity and bias current of the logical design. The results do not include additional JJs to distribute bias current in ERSFQ logic.	69

4.6	Categorical breakdown of the 32-bit KSALU for both the complexity and bias current of the logical design.	70
4.7	Breakdown comparison of both raw and adjusted bias JJ counts with respect to the total design complexity of the 32-bit KSALU implemented in ERSFQ logic.	71
4.8	ALU_INIT for the 8-bit ALU with HYPRES.	74
4.9	Simulated latency results of the 8-bit ALU design after post-layout verification.	76
4.10	Simulated processing rate results of the 8-bit ALU design after post-layout verification.	77
4.11	Microphotograph of the 8-bit ALU chip for low-frequency testing using the HYPRES 1.5 μm 4.5 kA/cm ² technology.	78
4.12	ADD operation during functional low-frequency testing.	79
4.13	Low-frequency functional testing of logical operations.	80
4.14	Block diagram of the ALU for high-speed testing and the I/O interfaces.	82
4.15	8-bit ALU chip with high-speed testing circuits.	83
4.16	Correct logical operations at 20 GHz when A is fixed to 255 and B is toggled at low-speed between 255 and 0.	84
4.17	20 GHz operation of two critical cases of the ADD operation where A is a fixed value and B is modulated between 1 and 0.	85
4.18	20 GHz operation showing correct functionality of ADD, AND, XOR and ADD for fixed values of A=101 and B=45.	86
5.1	The structure of a 16-bit sparse-tree adder.	89
5.2	The structure of a RSFQ 16-bit hybrid sparse-tree adder.	92
5.3	Logic schematic blocks that are unique to the sparse-tree structure.	93
5.4	Latency distribution of the 32-bit HSTALU. The least significant bit is bit index 0. The latency is measured from the assertion of the “ready” signal to the arrival of outputs at the “sum” port.	95
5.5	Cell-wise breakdown of the 32-bit HSTALU for both the complexity and bias current of the logical design. The results do not include additional JJs to distribute bias current in ERSFQ logic.	97
5.6	Categorical breakdown of the 32-bit HSTALU for both the complexity and bias current of the logical design.	98
5.7	Breakdown comparison of both raw and adjusted bias JJ counts with respect to the total design complexity of the 32-bit HSTALU implemented in ERSFQ logic.	99

5.8	Latency breakdown of the KSALU and HSTALU.	103
5.9	Processing rate and power consumption comparison.	103
5.10	HSTA/HSTALU sub-blocks re-designed using the CONNECT cell library.	106
5.11	Simulation of the clock generator at different bias voltages. . .	109
5.12	Schematics of the supplemental circuits to facilitate high-speed testing of the 16-bit HSTA.	110
5.13	Layout and microphotograph of the 16-bit HSTA chip for low-frequency testing.	111
5.14	Layout and microphotograph of the 8-bit HSTALU chip for low-frequency testing (Delta chip).	112
5.15	Layout and microphotograph of the 16-bit HSTA with on-chip high-frequency test circuits (Eagle chip).	113
5.16	DC bias margins for the 16-bit HSTA (Eagle chip).	114
5.17	DC bias margins for the 8-bit HSTALU (Delta chip).	116
5.18	Block diagram of the testing environment.	120
5.19	Equipment used for experimental testing.	121
5.20	16-bit HSTA chip for low-frequency testing wire-bonded to a chip holder.	122
5.21	Low-frequency random test #1, A = 56811, B = 14643, and Sum = 71454.	124
5.22	Low-frequency random test #2, A = 8724, B = 50892, and Sum = 59616.	125
5.23	Low-frequency random test #3, A = 13982, B = 64973, and Sum = 78955.	126
5.24	Low-frequency random test #4, A = 44636, B = 9199, and Sum = 53835.	127
5.25	Low-frequency serial output of the shifter for input A (blue trace) and B (yellow trace) on the 16-bit HSTA chip designed for high-speed testing. Both input shifters were initialized to all logical 1's (16-bit) so we expected to see 16 transitions each yet we only observed 13 for input A and none for input B. . .	128
5.26	Waveforms demonstrating various operations of the 8-bit ALU.	130

List of Tables

2.1	Key characteristics of the HYPRES standard Niobium process.	26
2.2	Key characteristics of the Japanese ISTEK Advanced Process (ADP 2.2).	28
2.3	Listing of RSFQ logic cells used through out the design of the adder and ALU.	32
3.1	Comparing T1's counting behavior with the full adder cell.	46
3.2	Latency distribution of the 32-bit RCA with the average latencies calculated across all bits.	48
3.3	Breakdown of bias JJs for the 32-bit RCA for ERSFQ logic.	51
3.4	Summary of the key simulation results for the 32-bit RCA.	52
4.1	The Boolean equations for the Kogge-Stone adder using triple-rail encoding.	56
4.2	Full adder decomposition into logical functions using Cin as a control signal.	58
4.3	Instructions decoded into the control signals for the ALU.	64
4.4	Listing of PTL interconnect lengths for the 32-bit KSALU.	65
4.5	Processing rate of the 32-bit KSALU.	66
4.6	Latency distribution of the 32-bit KSALU with the average latencies calculated across all bits.	67
4.7	Breakdown of bias JJs for 32-bit KSALU for ERSFQ logic.	71
4.8	Summary of the key simulation results for the 32-bit KSALU.	72
4.10	PTL length and delay breakdown for each stage of the 8-bit ALU.	73
4.9	Instructions decoded into direct control signals for the ALU.	75
4.11	Josephson junction complexity of the 8-bit ALU design.	76
4.12	Categorical complexity of the 8-bit ALU design.	77
5.1	Listing of PTL interconnect lengths for the 32-bit HSTALU.	91
5.2	Processing rate of the 32-bit HSTALU.	94
5.3	Latency distribution of the 32-bit HSTALU with the average latencies calculated across all bits.	96

5.4	Breakdown of bias JJs for 32-bit HSTALU for ERSFQ logic. .	99
5.5	Summary of the key simulation results for the 32-bit HSTALU.	100
5.6	Summary of the 3 design studies.	102
5.7	Stage-by-stage latency breakdown analysis of the KSALU and HSTALU.	104
5.8	Comparison of power and rate of the KSALU, HSTALU and an Intel Pentium 4 ALU	105
5.9	Clock generator high-frequency characteristics obtained from numerical simulation.	108
5.10	Breakdown of complexity for the 16-bit HSTA with on-chip high-speed testing circuits.	115
5.11	Summary of the 16-bit HSTA.	115
5.12	Breakdown of complexity for the 8-bit HSTALU.	116
5.13	Summary of the 8-bit HSTALU.	117
5.14	List of testing equipment.	118
5.15	Test vectors supplied to the 8-bit HSTALU with waveform outputs shown on Figure 5.26 on page 130.	131

Acknowledgments

Quite possibly the most difficult section I need to write for this dissertation is this one. To even fathom the countless people who have helped me in one way or another throughout this long journey is overwhelming. Sitting down at this very moment and reflecting on the many memories of joy, frustration, and success I have shared with the people I have encountered over the past several years has left me emotionally exhausted. My only hope is that I do not leave anyone out as I finish writing what is my way of expressing my utmost gratitude. But to anyone I may have overlooked, I truly appreciate our time together, whether it was ephemeral or lasting.

During the start of the Fall 2007 semester as an undergraduate senior in the Computer Engineering program, my goal was to complete my combined Master of Science and Bachelor of Engineering degree as fast as possible and find a job. Being in close proximity to New York City, the hub of international finance, it made the most sense to me to leverage my broad background in software engineering to pursue a career most likely as a technology analyst for a reputable banking firm, rather than use my hardware engineering skills for the handful of small defense companies on Long Island and the Tri-State area. I had absolutely no intention to earn a Ph.D.

It was during this semester when I met my advisor, Mikhail Dorojevets, for the first time during his undergraduate course in Computer Architecture. He immediately caught my attention during his introductory lectures, a historical perspective on the different computers built over the past century. He briefly showed a photograph of a prototype 20 GHz 8-bit FLUX-1 microprocessor he designed, certainly an impressive sight but I quickly dismissed it as something that is far too complex for a mere student, such as myself, to comprehend. As a senior, it was time to choose a capstone design project as part of the graduation requirement. Looking at the list of available projects, I narrowed it down to two: (1) a small scale GPU implemented on an FPGA or (2) gate-level design of ultra-fast asynchronous processor units using Rapid Single Flux Quantum (RSFQ) logic, Mikhail's project. I was always very interested in GPUs but being the naïve undergraduate student I was, the words "ultra-fast,"

“asynchronous,” “processor,” and most especially “Quantum” easily captivated me. I approached Mikhail during his office hours to discuss his senior design project. I recall going over concepts such as passive transmission lines, CFFs, and wave-pipelining. Then right on the spot, he started testing my knowledge of the carry look-ahead adder, a topic which at the time I had not looked at since my Digital Systems Design class over 2 years ago. Struggling mightily at first, I was able to convince him that I was a student capable of doing the project. Little did I know it was these first steps that propelled me into the path for this Ph.D.

The immense knowledge, creative intuition and unorthodox approaches are all valuable gifts that Mikhail has given me. I can say with great confidence that if I did not pursue his senior design project, I most likely would not have had the challenging yet stimulating research experience that I cherish today. His hands-on guidance, unrelenting conviction and strive for perfection has sharpened my skills over the years. As a friend, I have always taken his insightful advice to heart and I tremendously appreciate his outlook for my professional career. For going above and beyond what an advisor does for his students, I am extremely grateful.

Of course this dissertation would not be possible without the help of the other members of my defense committee: Sangjin Hong, Emre Salman and Jennifer Wong. I also thank Alex Dobioli who was on my preliminary defense committee. Their feedback, patience and encouragement are all greatly appreciated. And for her punctuality, attention to detail, and for guiding me through the labyrinth of paperwork and graduation formalities, I thank Rachel Ingrassia of the Electrical and Computer Engineering department.

No research experience is complete without sharing the trials, tribulations and triumphs with your fellow colleagues, and at the Ultra High Speed Computing Laboratory, your colleagues are also among your friends. First and foremost, a special thanks to (now Dr.) Artur Kasperek with whom I have journeyed with from the very beginning of my graduate study. His cordial personality, helpfulness and his uncanny ability to navigate effortlessly within the Linux environment has truly made my time at the laboratory a lively one. How he managed to complete his Ph.D. while working full-time at Motorola and looking after his growing family is probably another dissertation he can write about. Artur, I am sorry I was not available to support you during your defense but I am so happy you came through. I am sure you are having a wonderful time back in your home country of Poland with your family and I wish you nothing but the best going forward. Thanks to Kruti Shah, Prachi Bemalkhedkar, Subramaniyan Venkatachalam, Swati Shah and Surabhi Garg for the harmonious team work that allowed us to successfully complete the numerous design studies we were set out to achieve. I hope your newly estab-

lished careers lead you all to lives of great fulfillment ahead of you. To Zuoting Chen and Hao Chen, our time has been relatively brief together but as the laboratory's next generation of students, I want to wish you both the best of luck and I have no doubt that the two of you will have the same rewarding research experience as I had.

Collaboration is an important and necessary aspect in doing worthwhile research. I thank the designers, testers, the fabrication team and all the other members at HYPRES, particularly Timur Filippov who worked closely with us on the 20 GHz 8-bit KSALU. His timely feedback and thoroughness contributed to the overall success of the chip development and demonstration.

Japan, the "Land of the Rising Sun," is a very fascinating country that I had admired from afar. I have always thought I would only get a chance to visit there maybe once in my lifetime. I ended up traveling there 4 times now. I first want to thank Nobuyuki Yoshikawa of the Yoshikawa Laboratory at Yokohama National University for always welcoming me in open arms. His everlasting enthusiasm and conviviality are traits that even his students inherit, both past and present. I thank Yuki Yamanashi for his amicable help, useful suggestions and his strong appetite for a good drink, always leaving me to wonder if I will wake up on time for my departure flight back to the US the next morning. From Nagoya University, I express my gratitude to Akira Fujimaki and particularly Masamitsu Tanaka for his extensive help on the CONNECT cell library, his fully comprehensive replies to e-mails, and the bar hopping memories we share after the conference sessions of the day.

Nothing will ever make me forget the friends I have made throughout my multiple stays in Japan. I deeply appreciate the valuable assistance of Kohei Ehara, Taiichi Kato, Yasuhiro Shimamura and Yoshihiro Takahashi in testing the 8-bit HSTALU and 16-bit HSTA. It was a tremendous joy conversing with them while going through the torturous wait of cooling the test chips down to 4.2 K. I relish the interesting interactions I had with Kazuki Aoki who has demonstrated an astounding knowledge of American pop culture. Please do not be ashamed of your English, it is actually very good! In 2009, I attended the 12th International Superconductive Electronics Conference (ISEC) in Fukuoka, Japan where I first met Naoki Takeuchi. At that time, both of us were finishing up our M.S. degrees but Naoki already secured a position in industry. While I really cherished our time together during the conference, I was almost sure there would not be an opportunity to meet him again after he starts his career. And yet, almost 3 years later, we met again as Ph.D. students. I sincerely thank him for his remarkable friendship and his illuminating explanations of Japanese culture, in addition to sharing his own experiences in industry and his underlying reasons for returning to graduate school. Naoki, I hope you will have a fruitful experience during your visit to UC Berkeley and

I wish you the best of luck in completing your Doctoral degree. It was also an incredible pleasure befriending Qiuyun Xu whose passionate interest in a plethora of subjects, profoundly deep conversations, and an understanding ear has left a lasting impression on me. I am sure whatever path you decide to embark on will aid you in your search for self-realization and contentedness. And to all the other current and former members of the Yoshikawa, Yamanashi and Fujimaki Laboratories, I thank you for the amiable atmosphere and I send you my kind regards.

I have always felt obtaining experience in industry while still in school is key to establishing a promising career upon graduation. My time at NVIDIA has done just that. I thank my manager and computer arithmetic expert, Stuart Oberman, as well as my mentor Michael Siu for trusting my abilities to do research, RTL design and a little software engineering on projects that were vital to obtaining the “SoL” goals of the Streaming Multiprocessor ASIC team. The entire experience of working and learning with the very best engineers in industry on next generation GPU products was a dream fulfilled.

It was only a few years ago when I never really considered going into academia to do research and teach as a career. While I still slightly hold that same notion, it was my role as a teaching assistant for Mikhail’s undergraduate Computer Architecture course that has softened my once firm belief. I thank all the students with whom I had a wonderful opportunity interacting and sharing my own undergraduate experiences with. They have all certainly opened my eyes to the joys of teaching and have made me at least consider the possibility of going into academia.

An honorable mention goes to astrophysicist Neil deGrasse Tyson whose illustrated liking from a popular “meme” is the frontispiece of this dissertation. His gesture of astonishment comes from an interview where he explains how Sir Isaac Newton invented calculus before turning 26 years old. Selecting this “meme” as my frontispiece is my way of poking fun at myself for finally finishing this dissertation. At moments when I find myself frustrated and discouraged, I can always Google one of Neil’s many inspiring talks to remind myself of why I decided to pursue a STEM field degree and career path.

The friendships that I have forged during my undergraduate and graduate studies at Stony Brook, including those already mentioned above, are very dear to me. I treasure the famous bull sessions I had with both past and present fellow Ph.D. students, particularly Shung Han Cho, Cristian Ferent, Dongsoo Kim, Varun Subramanian and Anurag Umbarkar. Standing outside of CEWIT or the Light Engineering building, discussing our crazy decision to do a Doctorate degree and our tentative plans for the future has brought much needed catharsis for all of us. I also value the friendships of John Cheng, Travis Choberka, Michael Co, Alba Escobar, Kailash Ganapathi,

Anna Gromadzka, Munirah Hasan, Huy Huynh, Vaibhav Laghane, Jun Lin, Ikechukwu Okoligwe Jr., Glenn Roach, Frederick Rubino, Aditya Tagat and Hirdeepsinh Vansia, all of whom I have shared memorable experiences with at Stony Brook. Outside of Stony Brook, I am deeply grateful for the friendships of Angelo “Jello” Andrada, Clarissa “Cl@rissa and BFFEVAR!” Baquiran, Matthew “UnknownUser1486” Butcher, Maria DeVera, Carlavee Ervas, Jack “Jack Fury” Fiore, Matthew Miller, Lenima Wright, the Hopkins sisters – Jessica, C.J. and Cathy, and the Reyes brothers – Chris “BlueberryAttack”, Dustin and Chuckie. I thank each and every one of them for their outpouring support and wonderful memories.

And finally, I am forever indebted to those who have been there for me since day one: my dearest family. To my parents Noel Ayala and Angeles Armada Ayala, it is your unending, nurturing love that has made me the person that I am today. The elusive balance of discipline and parental affection must have been very difficult to find but nonetheless you have succeeded. You have instilled in me a hard working attitude while always encouraging me to pursue my interests and ambitions. I will never take for granted the extraordinary parents I have. Mom and dad, thank you so much for everything. A “flesh-pound” thank you goes out to my brother Michael “MastaShakes” with whom I have shared countless memories with from completing mundane errands together, shooting “hoop dreams” in the pool on those humid NY summer days, to holding down the barricade against a massive horde of zombies during our gaming sessions. Your companionship, camaraderie and support throughout all these years have been immeasurable. I also thank my grandparents Efren Ayala, Luisa Ayala and Natividad Armada. They were all influential figures during my formative years and I truly appreciate their love and boundless support. Lolo Efren, we all miss you so much and it pains me that you are no longer here to see me finally finish but I know you are very proud of me. And warm regards to all my extended family members, thank you for your support as well, and after completing this dissertation I hope to visit you all again back in the Philippines.

Christopher Lawrence Ayala
East Setauket, New York USA
December 2012

Vita

Christopher Lawrence Ayala was born in Brooklyn, New York, USA on July 20, 1986 to Noel Ayala and Angeles Armada Ayala. He began his Bachelor of Engineering degree in Computer Engineering with a minor in Computer Science at Stony Brook University in August 2004. In May 2007, he was admitted into the combined Master of Science and Bachelor of Engineering program in Electrical & Computer Engineering. He graduated *Magna Cum Laude* from the combined program in May 2009 and in the same year he was admitted into the Ph.D. program in Computer Engineering with a minor area in Circuits and VLSI. Since June 2008, he has been working as a research assistant under Dr. Mikhail Dorojevets, conducting studies on digital systems and architectures implemented in various families of superconductor logic. He passed the Computer Engineering Qualifying Exam in April 2010 and successfully defended his research proposal in May 2011. During the summer of 2012, he interned at NVIDIA for the GPU ASIC group where he researched and implemented architectural power optimization techniques for the floating-point datapaths of the next generation Streaming Multiprocessor. He is a Student Member of IEEE (S'07), Eta Kappa Nu and Tau Beta Pi Honor Societies, the latter of which he was the President for two terms from 2006-2008 and an advisor from 2008-2009.

Publications

The following is a list of publications that are a result of the research conducted for this dissertation:

1. M. Dorojevets, **C. L. Ayala**, N. Yoshikawa, and A. Fujimaki, “16-bit wave-pipelined sparse-tree RSFQ adder,” *IEEE Transactions on Applied Superconductivity*, vol. 23, Jun. 2013, (accepted).
2. M. Dorojevets, **C. L. Ayala**, N. Yoshikawa, and A. Fujimaki, “8-bit asynchronous sparse-tree superconductor RSFQ arithmetic logic unit with a rich set of operations,” *IEEE Transactions on Applied Superconductivity*, vol. 23, Jun. 2013, (accepted).
3. T. V. Filippov, A. Sahu, A. F. Kirichenko, I. V. Vernik, M. Dorojevets, **C. L. Ayala**, and O. A. Mukhanov, “20 GHz operation of an asynchronous wave-pipelined RSFQ arithmetic-logic unit,” *Physics Procedia*, vol. 36, pp. 59-65, 2012.
4. M. Dorojevets, **C. L. Ayala**, and A. K. Kasperek, “Data-flow microarchitecture for wide datapath RSFQ processors: design study,” *IEEE Transactions on Applied Superconductivity*, vol. 21, no. 3, pp. 787-791, June 2011.
5. T. Filippov, M. Dorojevets, A. Sahu, A. Kirichenko, **C. Ayala**, and O. Mukhanov, “8-bit asynchronous wave-pipelined RSFQ arithmetic-logic unit,” *IEEE Transactions on Applied Superconductivity*, vol. 21, no. 3, pp. 847-851, June 2011.
6. M. Dorojevets, **C. Ayala**, and A. Kasperek, “Development and evaluation of design techniques for high-performance wave-pipelined wide datapath RSFQ processors,” in *Proc. 12th Int. Superconductive Electronics Conf.*, Fukuoka, Japan, 2009, SP-P46.
7. M. Dorojevets and **C. Ayala**, “Logical design and analysis of a 32/64-bit wave-pipelined RSFQ adder,” in *Proc. 2nd Superconducting SFQ VLSI Workshop*, Fukuoka, Japan, 2009, pp. 15-16, 06.

Chapter 1

Introduction

Outline

1.1 Motivation	1
1.1.1 CMOS Scaling Limits	1
1.1.2 Significance of Arithmetic Logic Units	3
1.1.3 Benefits and Opportunities in Superconductor Technology	4
1.2 Research Outline and Goals	6
1.3 Superconductor Logic	7
1.3.1 Latching Logic	7
1.3.2 Rapid Single Flux Quantum Logic	8
1.3.3 Energy-Efficient Rapid Single Flux Quantum Logic	12
1.3.4 General Challenges for Superconductor Technology	12
1.4 Overview of Prior Work in Superconductor Electronics	14
1.4.1 Adders and ALUs	14
1.4.2 Microprocessors	18

1.1 Motivation

1.1.1 CMOS Scaling Limits

CMOS has been the predominant technology for digital integrated circuits since the semiconductor industry transitioned from NMOS technology in the early 1980s. Present day microprocessors are now manufactured using a 22 nm CMOS process and the International Technology Roadmap for Semiconductors

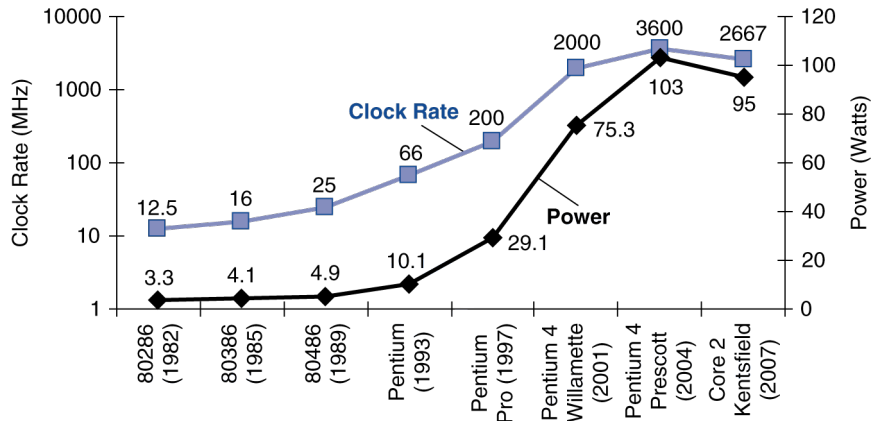


Figure 1.1: Clock rate and power for various Intel microprocessors [1].

(ITRS) 2011 expects the technology to scale down to 18 nm in 2014 and 15 nm in 2016 [2]. Intel has a more optimistic outlook, expecting to release products manufactured using the 14 nm process in 2014 and 10 nm in 2016 as they attempt to stay true to their “Tick-Tock” model [3]. However, the ITRS has forecasted that by 2020, it would be necessary to integrate multiple novel devices with CMOS to achieve properties beyond the ones of which scaling can provide. Furthermore, CMOS has also reached what is called the “power wall” [1, 4]. Figure 1.1 shows how power dissipation has leveled off in recent years. Increasing the power further would lead to expensive cooling solutions and circuit reliability issues. In the past, the primary source of power dissipation is dynamic power which is consumed during switching events in the circuit. Dynamic power is described in Equation 1.1, where P is the dynamic power, C is the capacitive load per transistor, V is the supply voltage and f is the switching frequency.

$$P = CV^2 f \quad (1.1)$$

Engineers were able to improve clock rates while keeping power consumption under control by lowering the supply voltage, which was made possible with each incremental improvement in technology. The effect was especially significant as power is a function of the voltage squared. But today, further lowering of the voltage has now increased static power consumption due to leakage. One type of leakage, known as subthreshold leakage, has been reduced by scaling down gate oxide thickness which in turn, also improved device performance and reduced short channel effects. However, this scaling also increased another type of leakage known as gate leakage. To overcome gate leakage, the CMOS process has recently been modified to use a higher

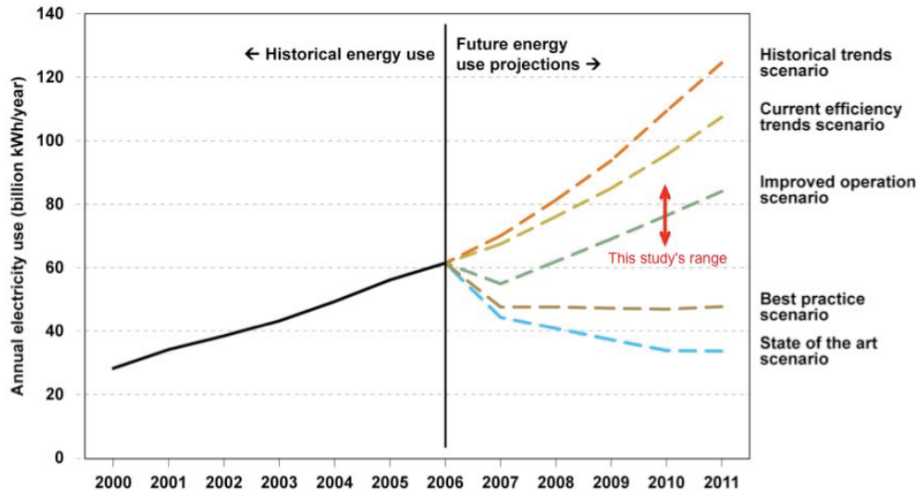


Figure 1.2: Predicted US electricity use for data centers and the range estimated in the study from [5].

permittivity (high K) gate dielectric material such as hafnium dioxide instead of silicon dioxide [6, 7, 8]. Combined with the transition from polysilicon gates to metal gates in the 45 nm process, the gate leakage has been substantially reduced [9]. Even more so, the emergence of multigate devices such as fin field-effect transistors (FinFETs) [10, 11, 12] and Intel’s tri-gate, high K transistors [13, 14, 15] have enabled the production of the current 22 nm process and opened a path for scaling further to smaller feature sizes. However, it has become increasingly difficult to reach each new technology node and it is unclear just how much further CMOS can scale over the next decade.

In niche markets such as data centers and supercomputers, CMOS has an ambitious task of providing long term support for these applications [16, 17]. With cloud computing, social networking, and full-fledged software applications making the transition to web-based thin client paradigms, there is an ever growing demand for data centers [18, 19, 20]. Figure 1.2 on page 3 shows that the electricity use of data centers will continue to increase beyond 100 billion kWh/year at its current rate [5, 21, 22]. In an effort to reduce power consumption while maintaining the high-performance needs of today’s data-centric world, alternative computing technologies should be considered.

1.1.2 Significance of Arithmetic Logic Units

The Arithmetic Logic Unit (ALU) is one of the most important components of a microprocessor as it is responsible for performing the actual execution of most

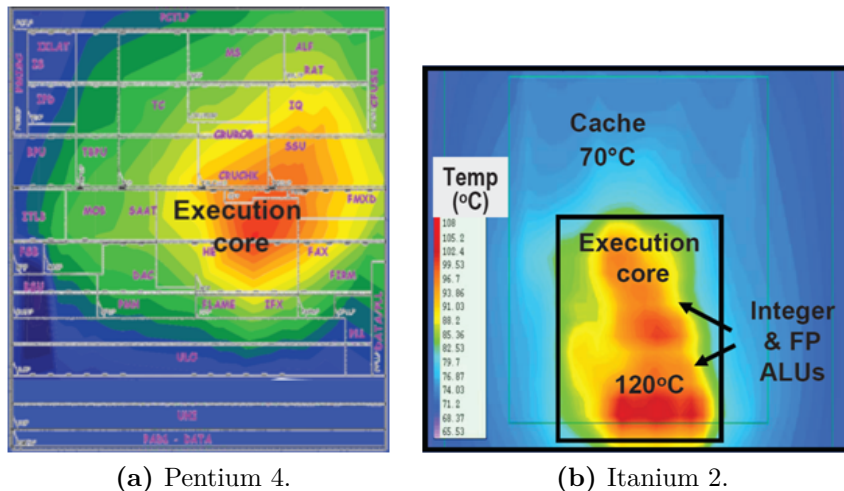


Figure 1.3: Thermal temperature maps of the execution cores in Intel microprocessors [23]. © 2003 IEEE.

instructions in a computer program. It is also one of the most performance-limiting units [24]. Being able to generate the lower-order 32-bits of the ALU output is essential in early address generation and executing consecutive operations. Furthermore, ALUs are also responsible for the thermal hotspots and sharp temperature changes within the execution core, making it one of the highest power-density locations on the processor as seen in Figure 1.3 on page 4. It is common for modern processors today to feature multiple ALUs to execute operations in parallel which exacerbates the power problem even further, impacting circuit reliability and increasing cooling costs. Thus it is important to seek energy-efficient ALU designs that can satisfy both high-performance requirements and low power dissipation.

1.1.3 Benefits and Opportunities in Superconductor Technology

With CMOS technology reaching its scaling limit, researchers have been looking for the next device which can be used to design digital circuits for high-performance and low power consumption. One candidate is the Josephson junction (JJ). The Josephson effect was predicted in 1962 [27, 28, 29], which is the phenomenon of electric current across two weakly coupled superconductors separated by a thin insulating barrier known as the Josephson junction in Figure 1.4 on page 5. For the effect to take place, the JJ must be cooled sufficiently to its critical temperature to reach the superconductive state. Niobium is usually the superconductor material of choice for JJs and it has a

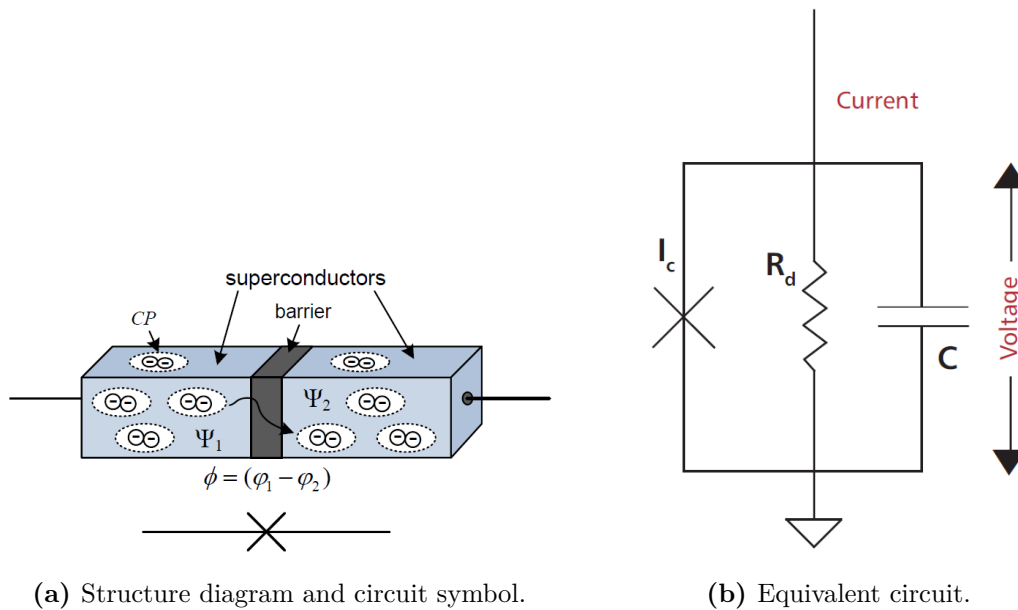


Figure 1.4: Views of the Josephson junction (JJ) superconductive device [25, 26].

critical temperature of approximately 9.2 K. The barrier is typically made of Aluminum-oxide. The thickness of the barrier must be approximately 1 nm or less, so that both normal electrons and Cooper pairs (superconducting electron pairs) can tunnel through the barrier [30, 31, 32].

A tunnel junction between normal metals (non-superconducting) is equivalent to a resistor shunted by a capacitor. However, if the metals are superconductors then Cooper pairs can tunnel at the same rate as normal electrons. When zero voltage is applied to the JJ, only Cooper pairs can tunnel whereas when a voltage is applied across the junction, both Cooper pairs and normal electrons can tunnel. This creates a third parallel channel to the normal tunnel junction circuit model as shown in Figure 1.4b on page 5, where I_c is the critical current (maximum current at zero voltage) of the junction, C is capacitance of the junction and R_d is the resistance of the junction.

In the late 1960s, it has become evident that JJs are suitable for extremely fast processing of digital signals. Furthermore, the fabrication technology for JJ integrated circuits is simpler than present day semiconductor fabrication, as there is no need to dope materials [33]. The speed of superconductor logic is extremely fast with gates operating as fast as 770 GHz [34], coupled with low power consumption. Additionally, there are some levels of interoperability with CMOS circuitry, a technique used in hybrid JJ-CMOS memory [26, 35, 36].

Because JJs are based on superconductivity, it is necessary to have a cryogenic environment in order to have the superconductive effect take place. Even with the cryostat devices used to reach temperatures of around 4.2 K, the total power of a complete system (circuit and cryostat) is still low enough to be an attractive alternative to CMOS.

With the development of superconductor logic, research is necessary to review design techniques and microarchitectures. What may have worked for CMOS may not be easily transferable in superconductor logic and that new systematic approaches may be necessary to provide both a scalable and energy-efficient logic design.

1.2 Research Outline and Goals

The overall goal of this research is to push the limits of the current state-of-the-art in superconductor technology through the design study of ALUs. We want to explore the design space of using this technology and find out what really can be achieved. There are three aspects we wish to accomplish at the bleeding edge:

1. Functional complexity - Move away from the simpler bit-serial designs currently popular in this field, and move towards wide datapath 32-bit designs similar in complexity of CMOS VLSI.
2. Clock rate & throughput - Achieve clock rates never demonstrated before in wide datapath design in the range of 20-30 GHz, resulting in larger throughput than ultra-fast bit-serial designs.
3. Latency - Keep latency below 600 ps, aiming for around 500 ps overall for 32-bit designs.
4. Fabrication technology - Use the most advanced superconducting circuit foundries in the world.

The remainder of this chapter provides an introduction of the different superconductor logic families and the various design challenges one must overcome in this technology. A brief overview of past work in the realm of adders, ALUs and microprocessors are also surveyed.

In Chapter 2, methodologies on how to conduct design studies on wide datapath circuits implemented using superconductor logic are discussed. Specifically, the development of a tunable VHDL cell library provides the capability for logical simulation of circuits and the gathering of statistics such as latency, processing rate, complexity, bias current, and power. Also in this chapter, we

briefly go over the asynchronous hybrid wave-pipelined approaches to achieve scalable wide datapath designs.

Three 32-bit wide adder cores are studied as the foundation for our ALU: the ripple-carry adder in Chapter 3, the Kogge-Stone adder in Chapter 4 and the hybrid sparse-tree adder in Chapter 5. The ripple-carry adder provides a starting point in the study to grasp how the simplest adder structure in CMOS can be designed in superconductor logic. In contrast to the ripple-carry adder, the Kogge-Stone adder is then studied as a complex, high-performance alternative. By combining techniques from the ripple-carry adder and the Kogge-Stone adder, a hybrid sparse-tree adder is developed to create an energy-efficient, high-performance core for an ALU design. In all three studies, we compare their characteristics in both RSFQ and ERSFQ logic with a focus on ALU development for the Kogge-Stone and hybrid sparse-tree structures.

In addition to the design studies carried out in this research, physical implementations and prototype chip demonstrations have been completed, specifically:

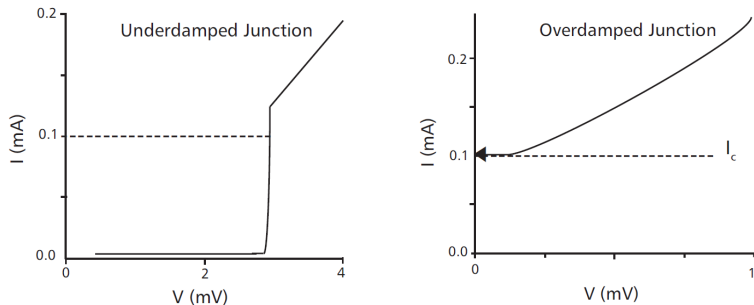
1. An 8-bit implementation of the Kogge-Stone ALU as a joint effort between SBU and HYPRES using $1.5\ \mu\text{m}$ $4.5\ \text{kA}/\text{cm}^2$ technology.
2. A 16-bit hybrid sparse-tree adder and an 8-bit hybrid sparse-tree ALU as a collaborative effort with Yokohama National University and Nagoya University using $1.0\ \mu\text{m}$ $10\ \text{kA}/\text{cm}^2$ technology.

Finally, Chapter 6 makes some final conclusions of the results obtained from this research, followed by a brief outlook on further research directions in superconductor electronics.

1.3 Superconductor Logic

1.3.1 Latching Logic

The first family of superconductor logic involved JJs in the voltage-state or latching mode whose I-V characteristics are illustrated in Figure 1.5a on page 8 [26, 37, 38]. It has a multivalued and hysteretic behavior such that the junction switches from $V = 0\ \text{mV}$ to $V = V_g$ at $I = I_c$, where V_g is the energy gap of the material ($\sim 2.7\ \text{mV}$ for Niobium). The junction can be reset back to the zero-voltage state when current is reduced to almost zero. This behavior provides a two-state voltage logic similar to CMOS. This approach was very popular during the 1970s and 1980s for superconductor computing projects at IBM and Japan. However, it required an AC power system in order to reset the junction back to the zero-state. Eventually this technology was dropped



(a) Voltage-state latching JJs. (b) SFQ-based non-latching JJs.

Figure 1.5: DC I-V characteristics of JJs [26].

as demonstrations showed operating rates close to 1 GHz, which was relatively fast at that time when compared to CMOS, but was very difficult to go any higher [39].

1.3.2 Rapid Single Flux Quantum Logic

Rapid Single Flux Quantum (RSFQ) logic has been developed since 1985 as a step towards a drastic increase in the operation speed of JJ digital circuits [27, 40] in contrast to the slower latching logic. It is based on JJs shunted by a resistor. The I-V curve of this approach is illustrated on Figure 1.5b on page 8 and it shows that its operation is non-hysteretic and single-valued. Furthermore, these devices are DC powered instead of AC powered as in latching logic. In RSFQ, digital logic is represented in the form of single flux quantum (SFQ) pulses which are very short (picosecond) voltage pulses $V(t)$ of a quantized area (Equation 1.2, where Φ_0 is the magnetic flux quantum, h is Planck's constant and e is the charge of the electron [27]). These SFQ pulses, compared to voltage levels in latching logic, are more naturally generated through JJs. They can also be reproduced, amplified, memorized and processed by JJ-based logic circuits.

$$\int V(t)dt = \Phi_0 = \frac{h}{2e} \approx 2.07 \text{ mV} \times \text{ps} \quad (1.2)$$

SFQ pulses can be transferred between logic gates through two types of connections: (1) passive transmission lines (PTLs, Figure 1.6c on page 9) or (2) active Josephson transmission lines (JTLs, Figure 1.6a on page 9). PTLs provide fast connections for long distance ballistic propagation of SFQ pulses at a velocity of $\sim 100 \mu\text{m}/\text{ps}$ [41, 43, 44]. Active JJ-based drivers (TX) and receivers (RX) are required to enter and leave the PTLs respectively, introducing

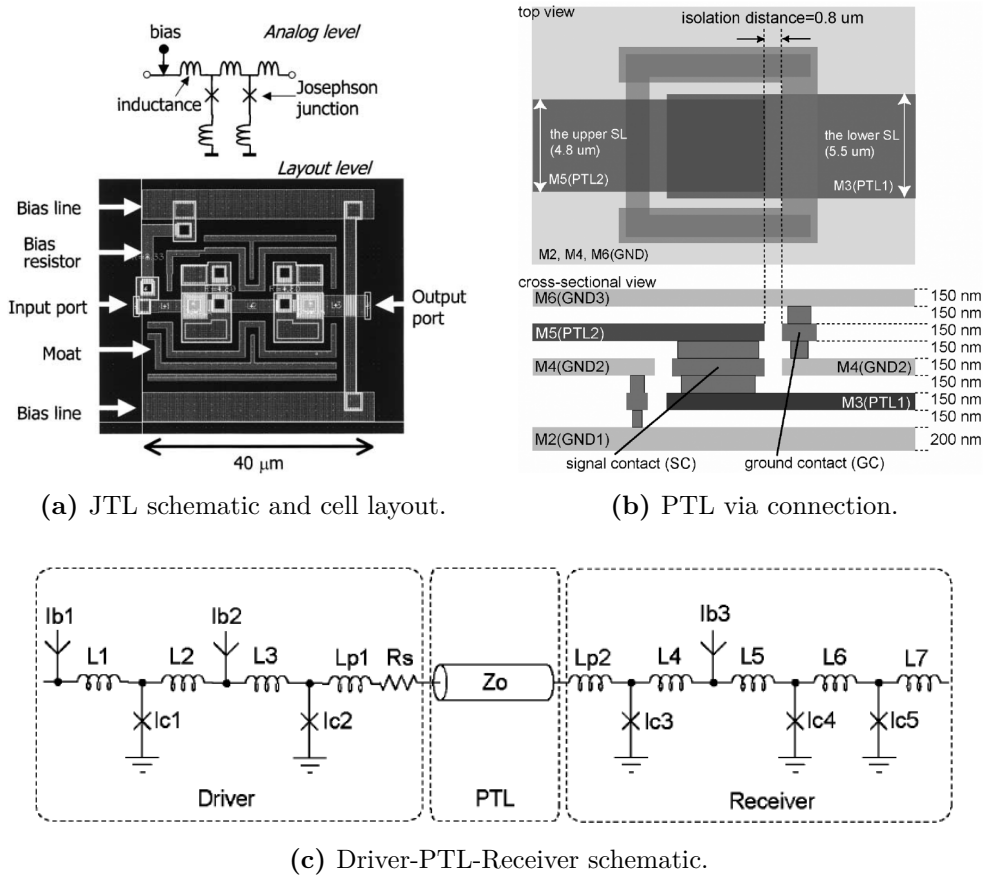


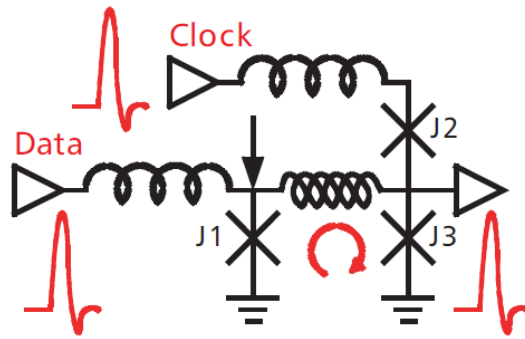
Figure 1.6: JTL and PTL connections used in superconductor logic [41, 42].
 © 2009 IEEE.

a propagation delay overhead before benefiting from the high-speed, low-loss transmission of PTLs. Since PTLs are usually on a different layer than the active JJs, they can be treated as an additional interconnect metal layer that is analogous to the multi-metal process of CMOS. Some processes, such as the ISTEK 10 kA/cm² advanced process, support two PTL layers with via connections as shown in Figure 1.6b on page 9 [41]. JTLs, on the other hand, are typically used for local connections between cells or as delay elements. They use the same plane as gates so they also compete for area, and as active circuit elements with a propagation delay, short distances should only be covered (i.e. short enough to have a propagation delay less than the TX/RX pair for PTLs) if the goal is to achieve the fastest connection.

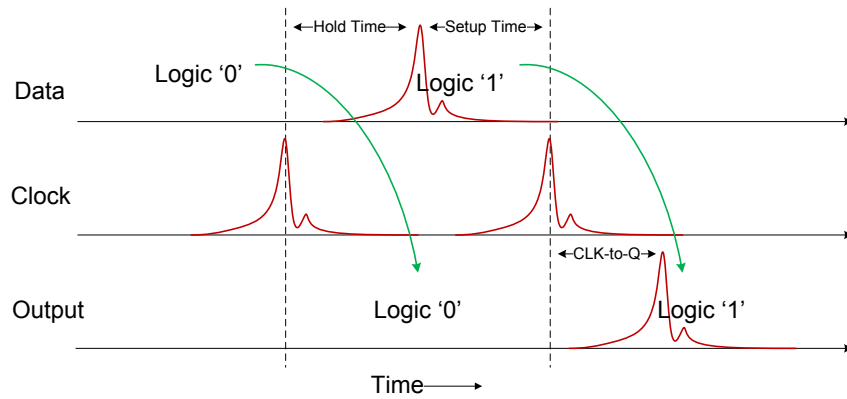
RSFQ is generally a positive-type logic in the sense that data is easier to process if it is a logical ‘1’ determined by the arrival of an SFQ data pulse. In this regard, RSFQ has a harder time dealing with logical ‘0’ or the absence of an SFQ pulse. This absence must be taken with reference to some clock pulse. All gates that must perform some type of inversion such as an inverter or an XOR gate, requires a clock input to differentiate whether data is truly a logical ‘0’, or data still has yet to arrive. Ensuring data is sent before the clock pulse arrives at a gate solves this problem.

At the circuit level, RSFQ gates are built with multiple JJs, inductors and bias resistors. A simplified schematic of a D flip-flop (DFF) gate is shown in Figure 1.7a on page 11. It includes an input junction J1, an inductor to store an SFQ pulse, and a decision-making comparator composed of J2 and J3. When an SFQ pulse appears at the data input, J1 will switch and store one flux quantum in the inductor between J1 and the J2/J3 pair. This stored flux quantum adds $\frac{\Phi_0}{L}$ amount of current in J3. Should a clock pulse arrive in this state, then an SFQ pulse is transmitted to the output and the DFF is set back to the zero-state. If there was no flux quantum stored in the DFF when a clock pulse arrives, then the current in J3 is insufficient to switch and thus no SFQ pulse is transmitted [26]. Figure 1.7b on page 11 shows the logic coding and timing diagram of SFQ pulses as they arrive into a DFF gate.

Bias resistors are used as current distributing elements to supply current to the JJs in RSFQ gates. A majority of the power dissipated in RSFQ comes from these bias resistors in the form of static power. Dynamic switching power contributes only ~1% of the total power consumption of circuits, particularly in large scale designs [45]. Total power of the circuit, assuming it is already at 4.2 K without any additional cooling equipment (P_{cryo}), can be estimated by Equation 1.3, where I_{bias} is the total bias current of the circuit and V_{bias} is the bias voltage fixed at 2.6 mV for the HYPRES 4.5 kA/cm² process, and 2.5 mV for the ISTEK 10 kA/cm² process. To determine the total “plug-in” power at room temperature (P_{room}) by taking into account the power to cool the



(a) Circuit schematic.



(b) Logic coding and timing.

Figure 1.7: RSFQ D flip-flop [27, 25, 26].

circuit down to 4.2 K, we simply multiply the power at 4.2 K by the cryostat efficiency of $1000 W_{room}/W_{cryo}$ as shown in Equation 1.4.

$$P_{cryo} = I_{bias}V_{bias} \quad (1.3)$$

$$P_{room} = P_{cryo} \left(1000 \frac{W_{room}}{W_{cryo}} \right) \quad (1.4)$$

1.3.3 Energy-Efficient Rapid Single Flux Quantum Logic

Energy-Efficient Rapid Single Flux Quantum (ERSFQ) logic aims to eliminate static power consumption altogether [46, 45]. The bias resistors used to distribute current to the logic gates are now replaced with JJs to take advantage of the critical current I_c as a natural current limiter. To use this biasing scheme, the voltage on the power line must be equal or greater than the maximum possible DC voltage used to power the circuit. This maximum DC voltage (V_{bias}) is determined by its clock frequency as shown in Equation 1.5, where f_c is the clock frequency that the circuit is designed to run at and $\Phi_0 \approx 2.07 \text{ mV} \times \text{ps}$ [46]. The ERSFQ bias voltage is in the range of 20-100 μV for clock frequencies in the range of 10-50 GHz, as opposed to the fixed 2.5-2.6 mV bias voltages used in RSFQ. If we substitute Equation 1.5 into Equation 1.3, we obtain the power consumption of ERSFQ at 4.2 K (Equation 1.6). To determine the power at room temperature, we can use Equation 1.4 in the same way as we did for RSFQ.

$$V_{bias} = \Phi_0 f_c \quad (1.5)$$

$$P_{cryo} = I_{bias} \Phi_0 f_c \quad (1.6)$$

1.3.4 General Challenges for Superconductor Technology

There are several challenges that designers must deal with when working with superconductor technology for digital circuits:

- Temperature induced fluctuations: The logic gates in superconductor logic do not have fixed propagation delays [47, 48, 49]. They are influenced by thermal fluctuations resulting in delay jitter. Thus it is necessary to provide reliable synchronization techniques for designs, particularly for ultra-high-speed, wide datapath RSFQ processors of which are more prone to these fluctuations [26, 50].

- Flux trapping: Trapped magnetic flux or frozen flux can degrade circuit performance and cause operation to malfunction [51, 52]. This typically occurs during the temperature transition into the superconducting state. The first step in minimizing this effect is to shield the circuit from the Earth magnetic field with Mu-metal shielding. The second, and most effective step, is to provide areas on superconductor chips to create special traps or moat structures in the ground planes and other large superconducting films to keep frozen flux far from the operating circuits [53, 54, 55].
- Overcoming large latencies and delay overhead: Much of the latency in a large circuit is typically due to splitting signals, especially asynchronous “ready” signals and broadcasted control signals. In RSFQ, splitting a signal into two requires at least an active splitting element. In HYPRES technology, it is necessary to add an additional JTL to re-amplify the signal after two levels of splitting, thus compounding the delay overhead even further. Minimizing splitting and overlapping operations to hide large latencies are important [50, 26].
- Static power: A majority of power is dissipated through bias resistors in RSFQ. For high-complexity circuits, static power has become a problem. The zero static power dissipation biasing network in ERSFQ logic [46], described in Section 1.3.3, aims to eliminate this challenge.
- Development of CAD tools for VLSI superconductor circuit design: While there are numerous tools available to simulate and evaluate superconductor circuits at the junction-level [56, 57, 58, 59, 60], there are very few CAD tools which abstracts the physics behind superconductor logic and focuses on logic level design such as [61, 62, 63]. These tools are necessary to evaluate VLSI-level complexity circuits. The SBU tunable VHDL cell library described in Chapter 2 as well as the CONNECT cell library in Japan [64] are among the latest academic efforts in this domain. With respect to commercial tools, NioPulse [65, 66] aims to provide a unified Cadence-like environment for VLSI superconductor circuit design similar to that of [62] and [64].
- Memory: Perhaps the biggest challenge in superconductor technology is memory. It is the performance limiting component in RSFQ processors. Solutions for on-chip or off-chip memory structures that can provide low-latency, high-throughput access is an on-going effort [67, 68, 69, 70, 71]. There have been recent proposals in developing Josephson magnetoresistive random-access memory (JMRAM) as a possible memory structure to couple with superconductor technology [26, 72] and a re-emergence of

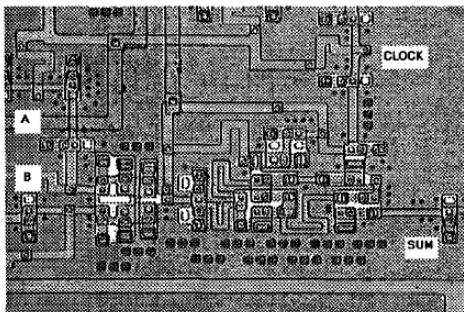
vortex transition memory cells [73] based on previous work from [74, 75].

1.4 Overview of Prior Work in Superconductor Electronics

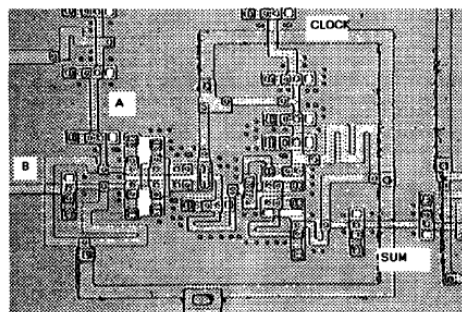
1.4.1 Adders and ALUs

1.4.1.1 “Push-Forward” RSFQ Carry-Save Serial Adders

In 1995, the research group at HYPRES developed a novel “push-forward” design of an RSFQ carry-save serial adder (CSSA) [76]. Two versions were designed, fabricated and successfully tested. The first version (CSSA1) uses traditional RSFQ, whereas the second version (CSSA2) uses a new design approach based on shifting or pushing forward stored data along the storage loops within the RSFQ gate by each incoming SFQ data pulse. Simulations showed a 30 GHz clock frequency for CSSA1 and a 40 GHz clock frequency for CSSA2. Both versions were successfully tested at low frequency with a measured DC bias margin of $\pm 20\%$ for CSSA1 and $\pm 14\%$ for CSSA2.



(a) CSSA1.



(b) CSSA2.

Figure 1.8: Microphotographs of the CSSA [76]. © 1995 IEEE.

1.4.1.2 Case Study of Fast Pipelined Parallel Adders in RSFQ

In 1999, the RSFQ research group at Stony Brook University conducted a design study of Kogge-Stone 32-/64-bit integer adders using concurrent flow sequencing [77]. The results from their study showed a 32-bit and a 64-bit adder having a design complexity of 28820 JJs and 66444 JJs respectively, and a maximum clock rate of 152 GHz for both designs. Their simulation results were obtained for a future $0.8 \mu\text{m}$ Nb-trilayer technology with plans to layout the critical path of the adders in $3.5 \mu\text{m}$ technology but unfortunately no further results were published.

1.4.1.3 1-bit RSFQ ALU with a 3-Input XOR Gate

In 2003, a 1-bit slice of an ALU has been demonstrated by the Superconductivity Research Laboratory, International Superconductivity Technology Center in Tokyo, Japan [78]. Its design is based on a 3-input XOR gate with additional logical gates to perform other functions in parallel, namely: AND, OR, ADD and SUB. These functions are then multiplexed based on the desired function. It consists of 560 JJs in a $1200 \mu\text{m} \times 2600 \mu\text{m}$ area. Simulation showed it can operate up to 50 GHz and experimental measurement of the fabricated chip showed bias margins of $\pm 37\%$ at low speed.

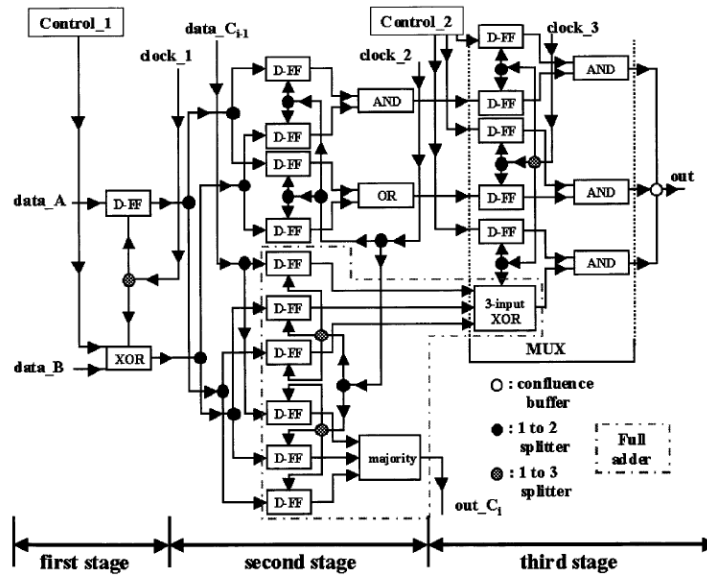


Figure 1.9: Schematic of the 1-bit ALU slice based on a 3-input XOR [78].
© 2003 IEEE.

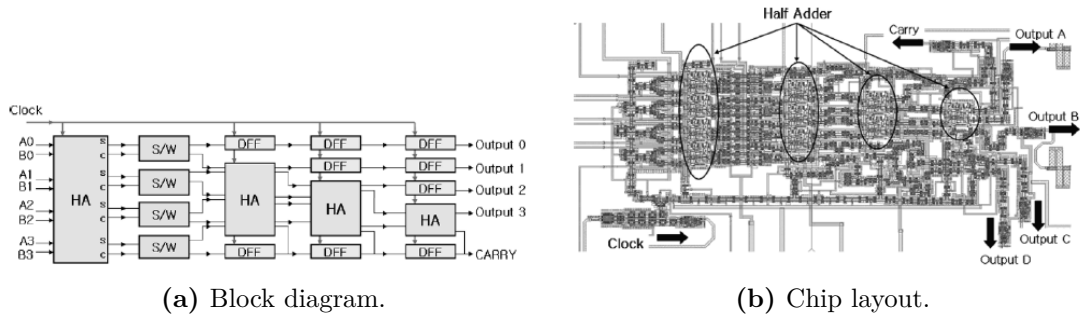


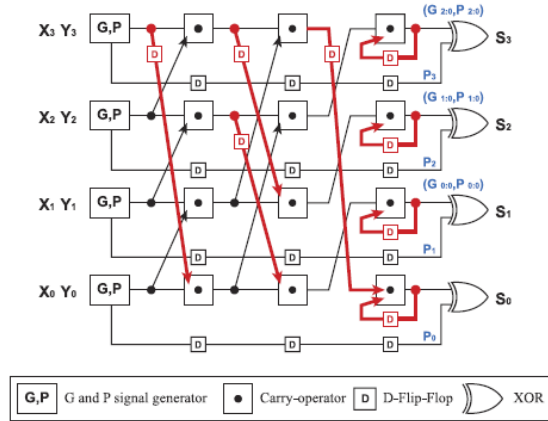
Figure 1.10: 4-bit RSFQ ALU built with half adder cells [79]. © 2005 IEEE.

1.4.1.4 4-bit RSFQ ALU with Half Adder Cells

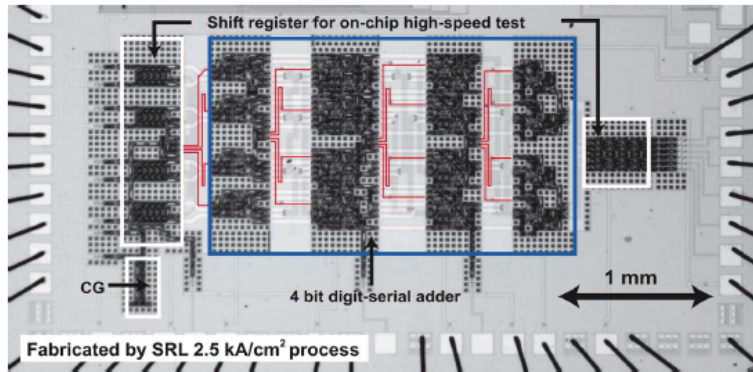
In 2005, a 4-bit ALU has been reported by a group from the University of Incheon, Incheon, Korea [79]. It was designed as part of an effort to develop a superconductive microprocessor. It consists of ten RSFQ half adder cells, four 2×2 switches, and several D flip-flops. The ALU featured four operations: AND, OR, XOR and ADD. The size of the circuit is 3.0 mm x 1.5 mm which is placed in a 5 mm x 5 mm chip. A 1-bit ALU block has been fabricated and successfully tested up to 40 GHz. The 4-bit ALU design was also successfully demonstrated but only at 5 GHz.

1.4.1.5 4-bit RSFQ Digit-Serial Adder

In 2009, a group from Yokohama National University and Nagoya University developed a digit-serial adder architecture [80]. It adopts a carry look-ahead structure to generate carry signals from the digit-serial data and are fed back internally to the next digit-serial data. A 4-bit digit-serial adder has been implemented using a 2.5 kA/cm^2 standard fabrication process. It consists of 2316 JJs and has a bias margin of $\pm 15\%$ at 25 GHz. It showed correct functionality up to 30 GHz.



(a) Digit-serial schematic.

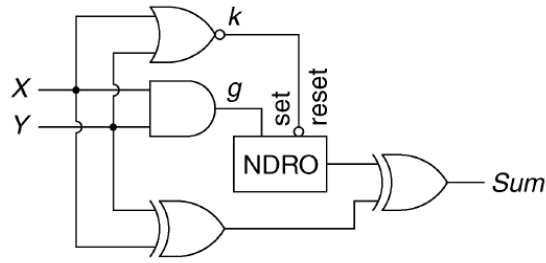


(b) Microphotograph of the chip.

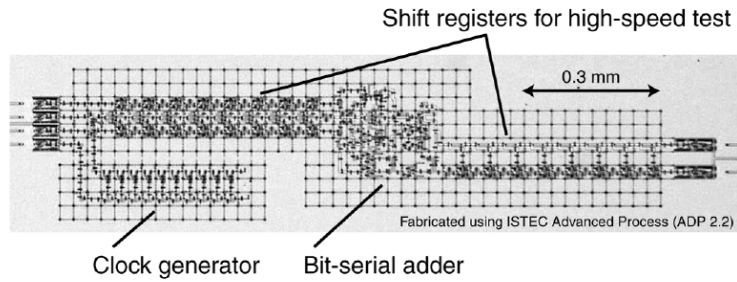
Figure 1.11: 4-bit digit-serial adder [80].

1.4.1.6 100 GHz RSFQ Bit-Serial Adder

In 2011, a collaboration of groups at Nagoya University, Yokohama National University, the Superconductivity Research Laboratory-ISTEC, and Kyoto University developed a bit-serial RSFQ adder with a target clock rate of 100 GHz [81]. The serial adder is designed around the concept of state transitions through the use of an NDRO (non-destructive read-out) cell. The concept was then implemented for serial addition of two 8-bit integers using the ISTEC 10 kA/cm² fabrication process and it has shown sufficient DC bias margins of $\pm 18\%$ at frequencies of up to 60 GHz. Correct operation has been verified up to 93 GHz.



(a) Bit-serial schematic.



(b) 8-bit bit-serial adder chip.

Figure 1.12: 100 GHz bit-serial adder [81]. © 2011 IEEE.

1.4.2 Microprocessors

1.4.2.1 Fujitsu's 8-bit DSP Microprocessor

In 1990, Fujitsu Laboratories designed an 8-bit DSP microprocessor as one of the early attempts to harness JJ technology to build practical chips [39, 82]. It is based on latching logic and uses a total of 23,000 JJs within a 5 mm x 5 mm chip. It has a 64 word x 24-bit instruction ROM, 16 word x 8-bit coefficient ROM, 16 word x 8-bit x 2 data RAM, a 13-bit 16-function ALU [83] and an 8-bit x 8-bit multiplier. It has an estimated maximum clock frequency of 1 GHz and a power consumption of 12 mW which at the time was about 100 times faster and one-tenth the power of conventional CMOS DSPs. All functions have been successfully demonstrated.

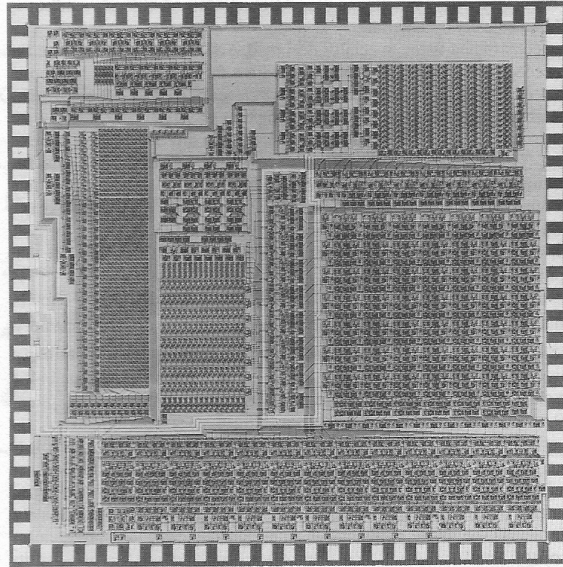


Figure 1.13: Microphotograph of Fujitsu’s 8-bit DSP based on latching logic [39]. © 1992 IEEE.

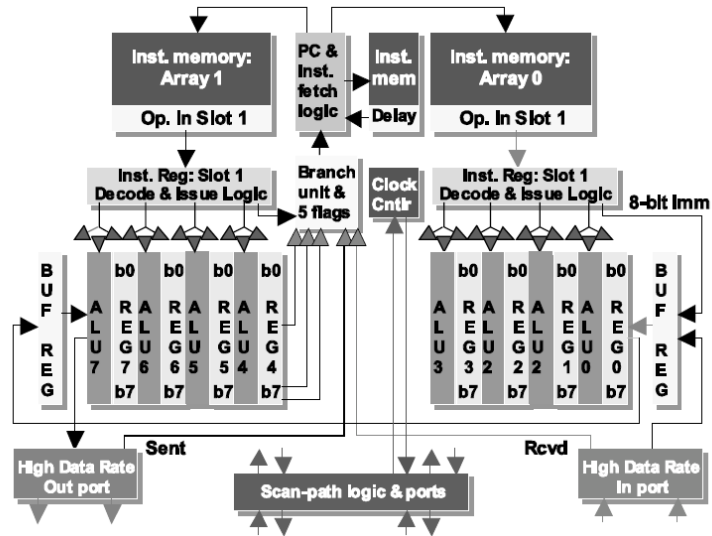
1.4.2.2 FLUX-1 Microprocessor

The FLUX-1 microprocessor is a 20 GHz RSFQ 8-bit processor developed under the collaboration between Stony Brook University (SBU) and TRW (now Northrop Grumman) [26, 84, 85, 86, 87]. The goal was to gain first hand understanding of the architectural and design challenges that must be surmounted for 20+ GHz RSFQ processors. It features the following:

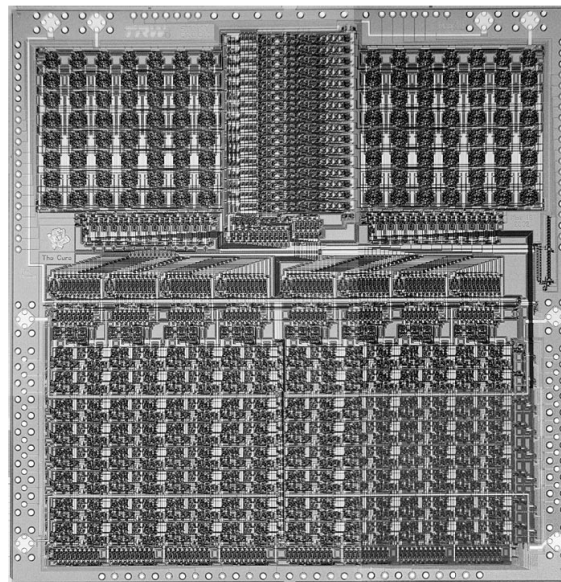
- Ultra-pipelined, 2-3 Boolean operations per stage.
- Two operations per cycle (40 GOPS peak for 8-bit data).
- “Processing in register” - interleaved ALUs and registers.
- Bit-streaming - allowing any operation dependent on another to start as soon as the first bit is available from the preceding, still-in-progress operation.
- Wave pipelining in instruction memory.
- Modular design.
- ~25 control, integer arithmetic, and logical operations (no load/store).

The final chip, FLUX-1R, had 63,107 JJs on a 10.35 x 10.65 mm² die with a power consumption of ~9.5 mW at 4.2 K. The group was able to demonstrate a

1-bit ALU-register block but there were no FLUX-1R chips fully demonstrated before the project ended in 2002.



(a) Block diagram.



(b) Microphotograph of the second chip.

Figure 1.14: The FLUX-1 8-bit RSFQ microprocessor [84]. © 2003 IEEE.

1.4.2.3 CORE1 Microprocessor

The CORE1 project was conducted by Japanese teams from Nagoya, Yokohama, and Hokkaido Universities, the National Institute of Information and Communications Technology at Kobe, and the International Superconductivity Technology Center (ISTEC) Superconductor Research Lab (SRL) at Tsukuba, Japan. It began as a simple processor known as CORE1 α with the following modest features [88]:

- Two 8-bit data registers and a bit-serial ALU.
- 32 byte shift register memory for instructions and data.
- Instruction set of seven 8-bit instructions.
- Non-pipelined processing and control logic.
 - Used 1 GHz system clock and 16-21 GHz local clocks.
 - 1 GHz clock for advancing instructions.
 - Fast local clocks used for bit-serial transfer and processing.
- ~7,220 JJs on a 3.4 x 3.2 mm² die.
- Power consumption rated at 2.3 mW.
- In 2003, it was fully demonstrated.

After demonstration of CORE1 α , a more advanced version was developed: CORE1 β [89]. It features the following:

- Total of 14 instructions in the instruction set.
- Four 8-bit registers.
- 2 cascaded bit-serial ALUs.
- 1 GHz system clock and 21 GHz local clock.
- ~9,498 JJs and power consumption rated at 3.0 mW.
- In 2007, all operations have been completely demonstrated for CORE1 β , final version 9e [90].

Finally, CORE1 γ was developed which integrated pipelining techniques and cache memories [91]:

- 16 byte and 8 byte shift-register-based cache memories for instruction and data.
- Overlaps 4 executed instructions in the pipeline.

- 22,302 JJs on 6.36 mm² area on an 8 mm² die.
- Power consumption estimated at 6.56 mW.
- From 2007–2008, testing has been conducted and functionality of the new System Clock Manager (SCM) and Instruction Cache has been confirmed, but no other results have been published.

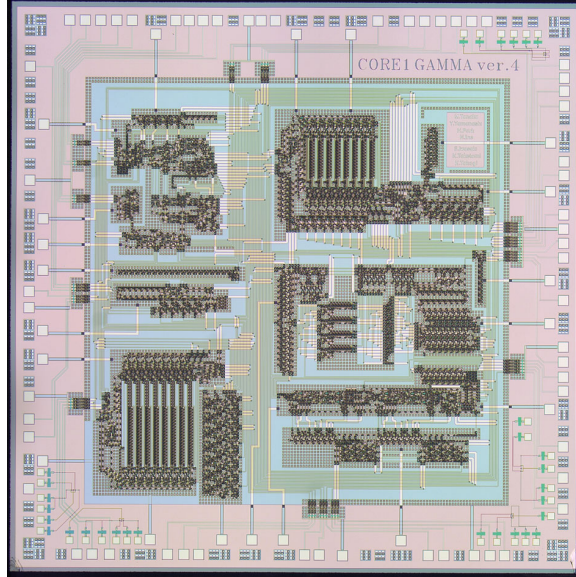


Figure 1.15: Microphotograph of the CORE1 γ 8 mm² chip [91].

1.4.2.4 20 GHz 8-bit RSFQ Frontier Datapath

From 2009–2012, work on an 8-bit RSFQ datapath has been carried out as a joint project between SBU and HYPRES. The datapath adopts an 8-bit version of the 32-bit Frontier data-flow microarchitecture developed at SBU [50]. Using a VHDL cell library tuned to the HYPRES 1.5 μm 4.5 kA/cm² fabrication process, the SBU team completed the cell-level design and verification of the 8-bit datapath. The team at HYPRES completed the physical layout design, fabrication and testing of two key units: an asynchronous wave-pipelined ALU [92, 93] and a multi-port register file [94]. The ALU has been successfully demonstrated to operate at the target clock rate of 20 GHz with $\pm 5\%$ DC bias margins. The register file has been successfully tested at low-frequency with $\pm 4\%$ DC bias margins. Work is still on-going for this project.

Chapter 2

Development of Efficient Techniques for VLSI Superconductor Design

Outline

2.1	Superconductor Cell Library and Design Tools . . .	23
2.1.1	SBU Tunable VHDL Cell Library	23
2.1.2	CONNECT Cell Library	28
2.1.3	Summary of RSFQ Logic Cells	32
2.2	Asynchronous Hybrid Wave-Pipelining	35

2.1 Superconductor Cell Library and Design Tools

2.1.1 SBU Tunable VHDL Cell Library

2.1.1.1 Purpose and Overview

In order to evaluate large-scale designs implemented in new technology, a simulation model needs to be developed. At Stony Brook University (SBU), the Ultra High Speed Computing (UHSC) Laboratory developed a tunable VHDL cell library that logically models superconductor circuits, and allows researchers to design, verify and profile large-scale architectures.

Each cell is described using a behavioral model based on FSMs (finite state machines) and/or logical truth tables when applicable. Parameters such as

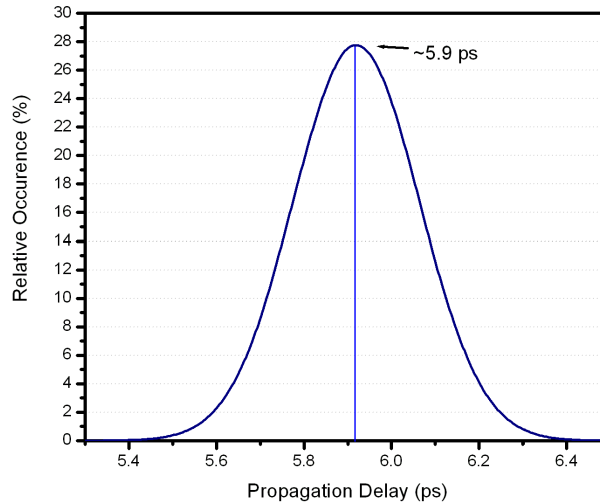


Figure 2.1: Normal distribution of delays obtained from a DFF cell simulation.

timing constraints, delay jitter, bias current, switching energy and complexity are provided by circuit level designers. This information is used by the logic level designers to develop and simulate processor units. These tools not only allow for the simulation and collection of all kinds of statistics for fabrication-ready designs but also provide insight on the technological hurdles that must be overcome for future wide datapath processor designs.

All propagation delays for each cell are modeled using a stochastic approach. Given the average and variance of each propagation delay, a normal distribution of the delays is created to provide a Monte Carlo simulation of the thermal-induced delay fluctuations a circuit would experience (Figure 2.1). The cells also check for any timing violations during simulation and report these violations as a failure. This methodology allows designers to identify weak areas where timing margins are very small. A failure report consists of important information such as what kind of timing constraint has been violated (setup or hold time), where the failure occurred and which inputs caused it. With this information, a logic level designer can identify the source of the failure and what measures should be taken to resolve it. Furthermore, all possible ways a cell can consume dynamic energy are stored in a switching table so that for a given switching event there is a corresponding amount of energy consumption accumulated locally in each cell instance during simulation. After the simulation is over, all cells report their switching energy totals and

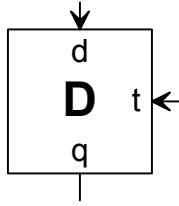


Figure 2.2: DFF symbol used in schematics.

are accumulated to produce the overall switching energy total for the entire simulation. Finally, the cell library provides a set of procedures/functions for the logic level designer to easily obtain the design complexity (i.e. number of JJs, bias current) and estimate the maximum clock rate that the design under test can support.

The simulation waveforms generated from our library look a bit unusual at first. This is because we represent SFQ pulses as logic events, a ‘0’ to ‘1’ transition or a ‘1’ to ‘0’ transition. Any transition represents a pulse, it does not matter whether it was on the rising edge or on the falling edge. This concept can be better explained by examining the RSFQ D flip-flop (DFF). It generally behaves the same way as its CMOS counterpart. The DFF has two inputs, the data input d and the clock input t . Starting from its initial state S_0 as shown in Figure 2.3, if a pulse arrives in input d and afterwards a pulse is applied to clock input t after setup time is satisfied, then a pulse is generated from output q with a clock-to- q delay. If no pulse arrived in input d before the clock pulse, then no output is generated. The simulation waveform for the DFF is shown in Figure 2.4. Notice that in Figure 2.4a it completely disregards the logic level and that it responds to only logic events (rising edge or falling edge transitions) as the representation of SFQ pulses. A cleaner representation is shown in Figure 2.4b which shows only the logic events.

Currently, the SBU VHDL cell library is tuned to the parameters of the HYPRES $1.5\ \mu\text{m}$ $4.5\ \text{kA}/\text{cm}^2$ standard Niobium process whose key characteristics are outlined in Table 2.1 on page 26 and a cross section of the process is illustrated in Figure 2.5 on page 27. Parameters for each logic gate is described through a custom made Standard Parameters File (SPF) which is read by the cell library upon elaboration of a design for simulation. This is an easy way to evaluate designs for different fabrication processes. In fact, an SPF made for a future $20\ \text{kA}/\text{cm}^2$ process was used to evaluate the potential performance of a wide datapath data-flow microarchitecture [50].

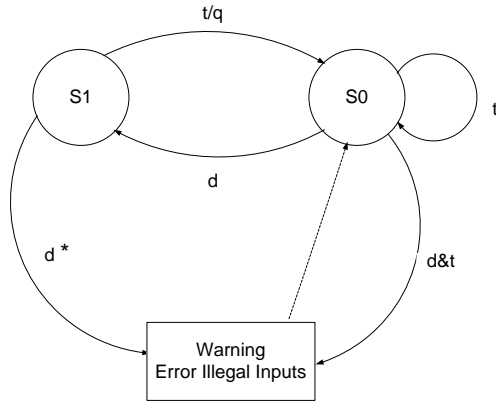
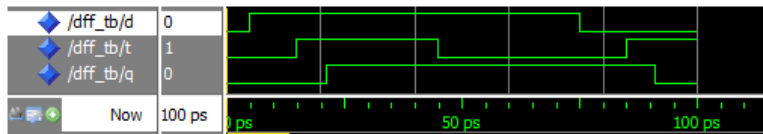
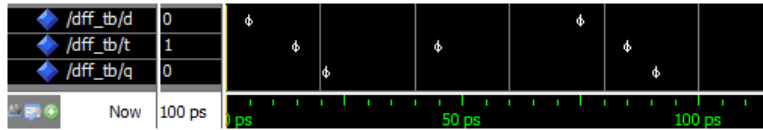


Figure 2.3: FSM of the DFF cell.



(a) Typical waveform view.



(b) Event waveform view.

Figure 2.4: Logic simulation waveform of the DFF cell.

Table 2.1: Key characteristics of the HYPRES standard Niobium process [95].

Minimum Feature Size	1.5 μm
Critical Current Density	4.5 kA/cm^2
Nominal Bias Voltage (for RSFQ)	2.6 mV
Number of Superconducting Layers	4

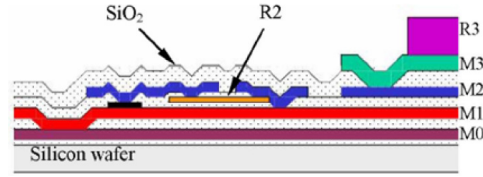


Figure 2.5: Cross section of the HYPRES standard Niobium process [95].

While our approach has delivered fully functional chips [92, 93, 94], it admittedly has some limitations. First, all design descriptions are completed using structural VHDL code (as opposed to dataflow or behavioral code). Logic is designed by hand since no synthesis tools are available, therefore the designer must describe their circuit as structural gates connected to other gates. This is extremely cumbersome for complex blocks as a designer must instantiate each gate and connect all necessary signals by hand. Currently, schematics are drawn without automatic back annotation of VHDL code. It is only used as a visual aid to help code the VHDL description and it must be checked by hand to ensure consistency between the code and schematic. Second, it does not integrate analog circuit level models of cells. This is fine for our purposes but it does not lend itself to becoming a unified simulation suite that can be used at different levels of design abstraction. Lastly, layout views are not incorporated so a physical designer must rely on, interpret, and check for consistency with the manually made logic schematics themselves.

Ultimately, the VHDL cell library is best suited to conduct design studies to profile performance, complexity and power. Combined with powerful VHDL generate statements and configuration packages, different architectures can be quickly explored by changing generic settings and configuration parameters without having to create separate designs by hand.

2.1.1.2 Acknowledgments

The SBU tunable VHDL cell library is a culmination of work from past and present members of the UHSC laboratory under the direction and supervision of Dr. Mikhail Dorojevets. The notable contributors are the following:

- Christopher Ayala (dissertation author): Involved in all aspects of library development, implementation, testing, documentation, and maintenance.
- Artur Kasperek: Implemented some of the gate-level FSMs and added additional helper functions.
- Kruti Shah and Prachi Bemalkhedkar: Contributed to the library documentation.

- Zuoting Chen: Implemented additional features for a new superconductor logic and will take over future development of the library.

2.1.2 CONNECT Cell Library

2.1.2.1 Purpose and Overview

The SFQ CONNECT cell library has been developed by Yokohama National University and Nagoya University in an effort to tremendously speed up the design process of taking a set of functional specifications and performance requirements all the way to chip tape-out [42, 64]. It provides a unified simulation environment in Cadence where analog JJ circuit level simulation of individual cells can be performed through JSIM (SPICE-like simulator for JJ-based circuits) [58, 96]. Layout views are integrated into each cell from which parameters can be extracted and back annotated into the analog model. And most importantly for our purposes, logical Verilog models are also incorporated to facilitate fast logic level simulation of circuits with timing delays and constraints dependent on a globally defined bias voltage. A portion of this research is based on using the CONNECT cell library for the ISTEK 1.0 μm 10 kA/cm² Advanced Process (ADP) whose key characteristics are listed in Table 2.2 on page 28 with a cross section shown in Figure 2.6 on page 29.

The design flow of the CONNECT cell library starts with the schematic capture of the circuit at the gate-level. In the schematic view, cells are placed and oriented next to each other to create logical connections. As shown in Figure 2.9 on page 31, the placement and orientation on the schematic directly correlate with the physical layout view so there is no need to perform a Layout Versus Schematic (LVS) check. PTLs are also placed on the schematic under the cells along two central tracks of a 30x30 μm^2 grid that the gate placements are based on as shown in Figure 2.8 on page 30. Schematics can also be designed modularly through the use of hierarchical blocks. Moat cells should be placed so that they surround the logic circuit to aid in flux trapping and prevent the logic gates from being disturbed. Two to four JTLs should also be placed after DC-to-SFQ converters and before SFQ-to-DC converters which

Table 2.2: Key characteristics of the Japanese ISTEK Advanced Process (ADP 2.2) [97].

Minimum Feature Size	1.0 μm
Critical Current Density	10 kA/cm ²
Nominal Bias Voltage	2.5 mV
Number of Superconducting Layers	9

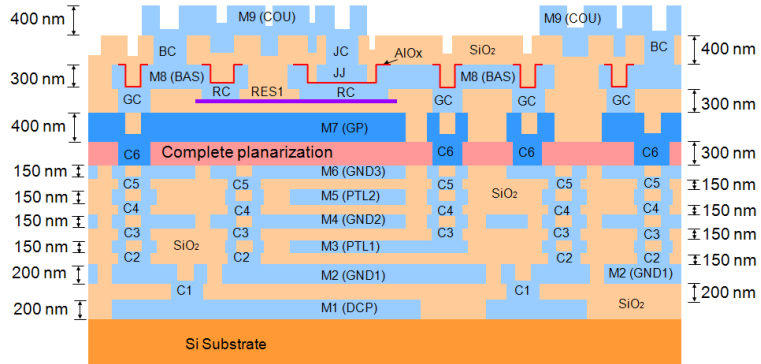


Figure 2.6: Cross section of the Japanese ISTEAC Advanced Process (ADP 2.2) [97].



Figure 2.7: Logic simulation in Cadence NC-Verilog.

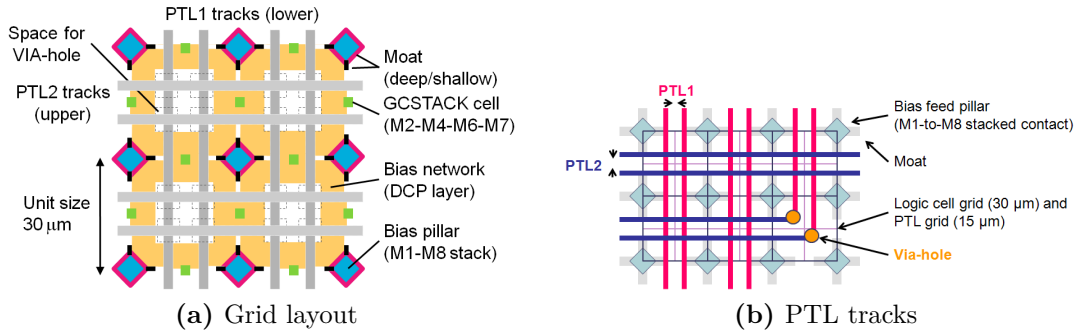


Figure 2.8: Grid-based approach used in the CONNECT cell library [64].

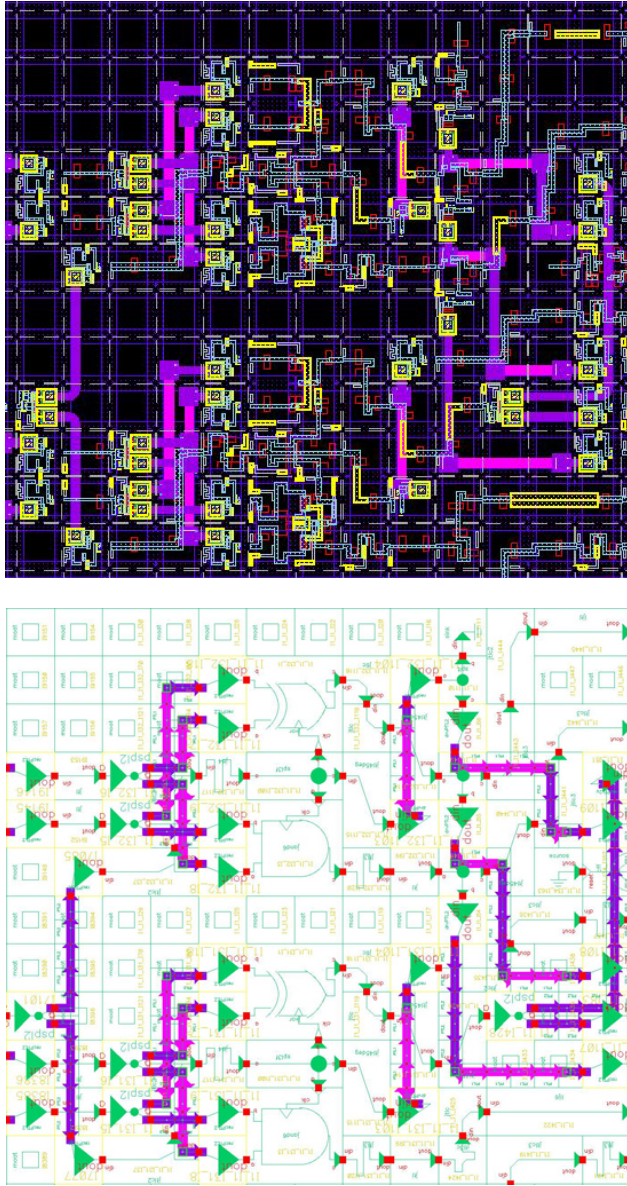
are used to communicate outside the chip.

After the schematic is complete, simulation is conducted through Verilog. Each cell has its own Verilog model with a table of propagation delays depending on the bias voltage. It also includes timing checks for setup and hold violations. The schematic is transformed into a Verilog structural netlist and is brought into the NC-Verilog simulator where a testbench can perform functionality tests. Timings can be observed directly through the waveform viewer as shown in Figure 2.7 on page 29. One of the most important steps in this design-flow is to simulate frequency dependent bias margins, which encompasses testing the design over a range of clock frequencies, varying the bias voltage at each clock frequency and observing whether or not the design works (i.e. no timing violations and is functionally correct) for the given clock frequency and bias voltage.

Once simulated testing and verification are complete, the schematic can be directly converted into a layout and placed on a chip frame. Within the chip frame, I/O and bias pads are placed and wired to the circuit. When the chip layout is complete, a Design Rules Check (DRC) is performed to ensure the circuit did not violate any design rules, particularly spacing. With this final check, a GDSII layout file is generated and submitted to our collaborators in Japan for fabrication.

2.1.2.2 Acknowledgments

The CONNECT cell library was entirely developed by the SFQ research groups at Yokohama National University and Nagoya University. We are grateful they had allowed us to use their library to implement the designs in this research.



(b) Layout view.


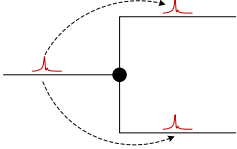
(a) Schematic view.

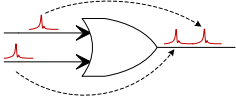
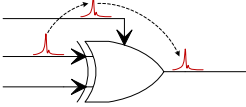
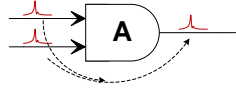
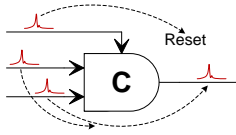
Figure 2.9: Example of how the schematic and physical layout correspond to each other when designed using the CONNECT cell library. Note that PTL and via cells (pink and purple lines) are also placed at the schematic level as well.

2.1.3 Summary of RSFQ Logic Cells

The adder and ALU designs use numerous RSFQ logic cells of which some are quite different from the typical gates found in CMOS technology. Table 2.3 on page 32 lists the major RSFQ cells used throughout this research. A diagram of the SFQ operation and a brief description are provided to help the reader become familiar with the logic cells.

Table 2.3: Listing of RSFQ logic cells used through out the design of the adder and ALU.

Cell Name	Symbol and SFQ Operation	Description
JTL/JL		Josephson transmission line (JTL or JL): Primarily used to connect one logic cell to another. Multiple JTLs can be used to insert additional timing delay or to amplify weak SFQ pulses (e.g. pulses after two levels of splitting).
SPL		Splitter (SPL): Connections in RSFQ are point-to-point so there is no passive way to create a fan-out. Instead, an SPL composed of active JJs is used to create a fan-out of 2. It is an active circuit element that approximately has the same delay as a single-stage JTL. Multiple SPLs can be used to create multiple fan-outs, keeping in mind that two SPLs in series require a JTL to amplify the output (only in HYPRES). A designer must be very careful with timing when SPLs are used.

Cell Name	Symbol and SFQ Operation	Description
MRG		Merger (MRG): Also called confluence buffer (CB) in the CONNECT cell library. It is used to channel two different SFQ paths into a single path. It has a timing requirement such that two SFQ pulses must be separated by T_{MINGAP} . This separation can be realized through multiple JTLs inserted before one of the inputs. MRGs are often used to route multiple SFQ pulses into a T1 or TRS gate.
CXOR		Clocked eXclusive OR (CXOR): A synchronous XOR gate. If, and only if, one SFQ pulse arrives in either input, but not both, before the arrival of clock, then an output is produced after clock arrives.
AAND		Asynchronous AND (AAND): An asynchronous AND gate that requires an SFQ pulse in each of the two inputs to arrive within a certain timing window (T_{MAXGAP}) relative to each other in order to produce an output SFQ pulse.
CFF		Resettable Muller C-flip-flop (CFF): An asynchronous resettable AND gate which will indefinitely wait for an SFQ pulse to arrive in each of the two input ports before it will produce an output pulse. A reset input is available to clear the CFF (throws away the waiting SFQ pulse). We have specifically requested our collaborators to implement this unique gate with a resetting capability as it is a key mechanism for asynchronous wave-pipelined circuits.

Cell Name	Symbol and SFQ Operation	Description
CAND		Clocked AND (CAND): An AND gate whose state is evaluated when a clock input arrives. If an SFQ input pulse arrived in each of the input ports before the clock, then an SFQ pulse is created at the output after the clock pulse arrives. Only available in the CONNECT cell library.
DFF		Delay/data flip-flop (DFF): A flip-flop similar to its CMOS equivalent where a data SFQ pulse is received at input d and then stored until it is read out by the clock pulse.
DFFC		DFF with complementary output (DFFC): A DFF which has both direct and complementary outputs.
D2FF		DFF with 2 clock inputs and 2 outputs (D2FF): It stores a single SFQ pulse from input d and then routes it to either q_0 or q_1 if a clock pulse arrives at t_0 or t_1 respectively. It is effectively a demultiplexer.
T1		Toggle flip-flop with asynchronous carry and synchronous sum (T1): A gate which can be used as a half adder, full adder, counter/compressor or frequency divider. It asynchronously generates an SFQ output on q_0 for every even number of pulses that arrive at input t since the last clock c was applied. It synchronously generates an SFQ output on q_1 after clock c was applied and if only an odd number SFQ pulses have arrived at t since the last clock.

Cell Name	Symbol and SFQ Operation	Description
TRS		Toggle flip-flop with reset and set (TRS): A gate similar to T1 except both outputs are asynchronous. A pulse in the r input will initialize the gate to the 0-state so that the arrival of the next data pulse at t will produce an output at q_0 . A pulse in the s input will initialize it to the 1-state so that the arrival of the next data pulse at t will produce an output at q_1 .

2.2 Asynchronous Hybrid Wave-Pipelining

Traditional synchronous pipelining incorporates the use of latches between stages to separate data moving through the pipeline. Only one set of data can exist for a given pipeline stage. The worst-case path of the longest stage plus some additional overhead from synchronization (clock-to-q, setup time, and clock skew) determines the cycle time, and thus the maximum clock frequency. Synchronous pipelining is very popular in CMOS as clock distribution networks are relatively easy to design when compared to RSFQ. In RSFQ, synchronization overhead can be much larger than the latency of logic between stages for two reasons:

1. Small amount of logic per stage as a result of high clock rates or short cycle times.
2. Large clock skew because splitting SFQ pulses requires active splitting elements with each one introducing significant propagation delay.

Furthermore, there is timing uncertainty in RSFQ circuits due to temperature induced fluctuations mentioned in Section 1.3.4.

Wave-pipelining is a technique in which the intermediate latches are removed [98, 99, 100, 101, 102, 103, 104, 105]. This reduces area, power and overhead from the clock distribution network. The clock rate increases as multiple data waves can exist in any stage (Figure 2.13). There are two main requirements in wave-pipelining [102]:

1. Prevent collisions of unrelated data waves.
2. Equalize delay paths to reduce differences between the shortest and longest delays in the combinational logic.

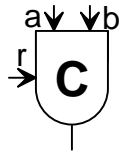


Figure 2.10: CFF symbol used in schematics.

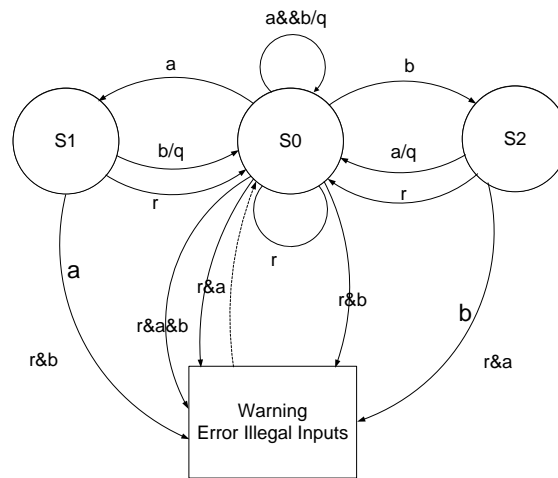


Figure 2.11: FSM of the CFF cell used in asynchronous wave-pipelining.

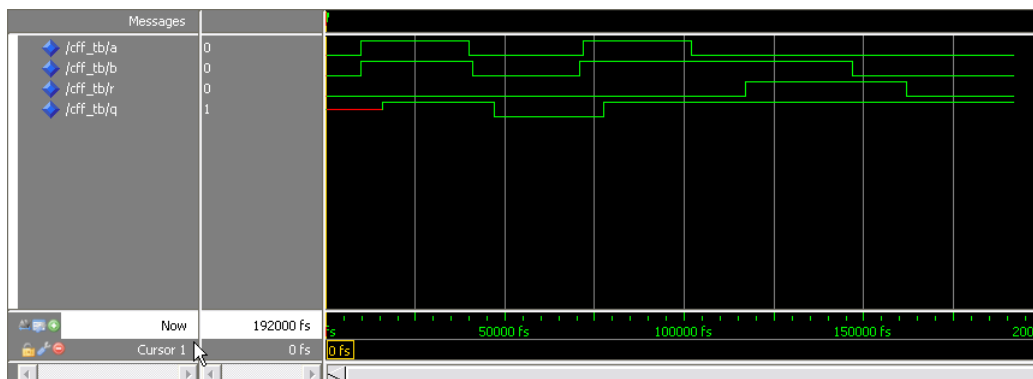


Figure 2.12: Logic simulation waveform of a CFF cell.

Even if we were able to equalize all delay paths, most likely they will not be perfect. There would still be very small differences between the delay paths since gates do not have propagation times that can be divided evenly with JTLs. These differences can accumulate from stage-to-stage, and thus it is also necessary to hold signals before proceeding to the next stage. In RSFQ logic, this is achieved by using a resettable asynchronous Muller C-flip-flop (CFF) [106, 107, 108, 50]. A CFF gate can be thought of as an AND gate in CMOS, with some internal memory. It has two data inputs labeled a and b , as well as one more input named r for reset. CFF will only produce a pulse on output q if it has received one pulse in each of the two data inputs (Figure 2.11). It does not matter when they arrived, they can arrive with a large time separation or at the same time but as soon as the 2nd of the two data pulses arrive, an output pulse from q will be generated after some propagation delay. Since the CFF gate can indefinitely wait for that 2nd pulse to arrive at its other input, it might be possible to receive another pulse in the same input that just received one. It is illegal for multiple pulses to arrive at a single data input and so it is necessary to “clean” or reset the CFF by providing a pulse on the r input before the next data pulse arrives. Figure 2.12 shows the waveform simulation of the CFF. Notice that if an output is generated, it is not necessary to provide a reset before the next data pulses. This is also reflected in the FSM (Figure 2.11). With CFFs, data waves have some level of synchronization which ensures that parts of the wave will not get too far ahead or lag behind. This is especially effective when the encoding of the data wave allows asynchronous propagation and a trailing reset wave is used to clean the CFFs as data passes through, creating a truly data-driven design (Figure 2.13).

Ideally, the maximum processing rate of wave-pipelining ultimately depends on the sum of two timing constraints of the CFF: (1) timing separation between the arrival of data and the trailing reset (setup time) and (2) timing separation of the arrival of reset and the next incoming data (hold time) (Figure 2.14c on page 40). Because we cannot achieve perfect equalization of all combinational paths and the thermal fluctuations influence the gate delays, timing margins must also be added to the timing constraints.

Another popular asynchronous RSFQ sequencing technique is co-flow clocking (Figure 2.14) [109]. In this case, the same clock that is used to move data at the first stage is also used at all subsequent stages to move the same set of data [77, 109]. In other words, only one clock pulse is necessary to move one data wave through all stages of the pipeline. In co-flow clocked designs, it is necessary to have distribution trees at each stage with matching delays to slow down the propagation of the clock pulse as it travels to the next stage. Safety delays are also inserted to provide margins for setup time. And simi-

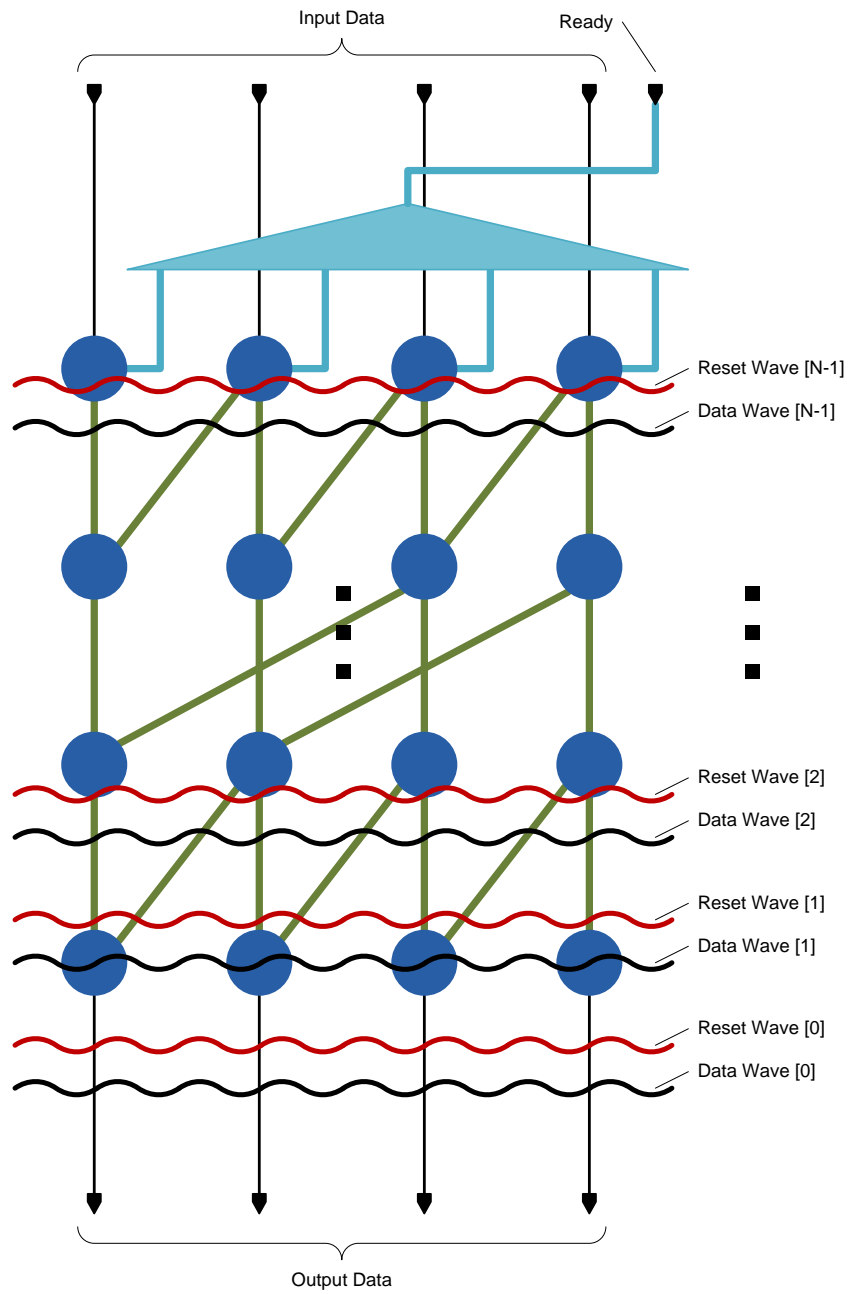
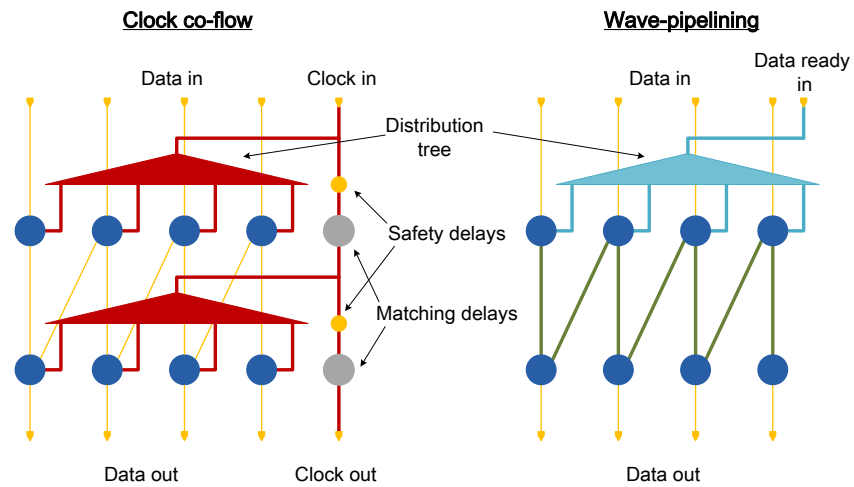


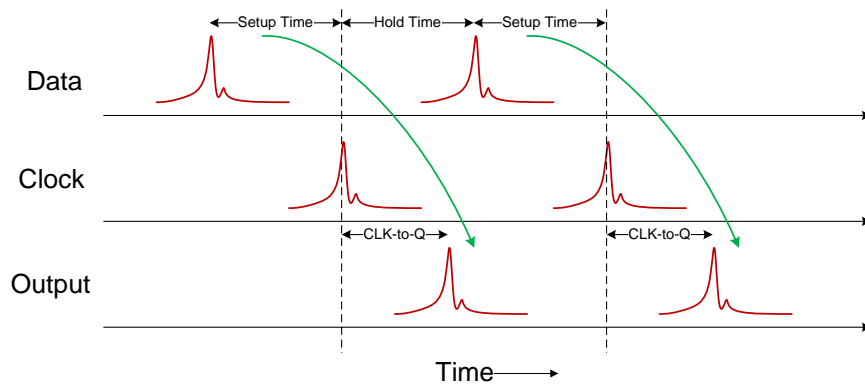
Figure 2.13: An example of asynchronous wave-pipelining with trailing reset waves.

lar to how it is necessary to equalize all data delay paths in wave-pipelining, delays should be inserted on data propagation paths where hold time is a concern. The addition of distribution trees, matching delays, safety delays, and data delays contribute to a substantial amount of complexity and power. The maximum processing rate also depends on setup and hold time but with respect to data and clock as opposed to data and reset in wave-pipelining. Assuming that clock based elements have similar timing constraints to that of CFFs, both co-flow clocking and wave-pipelining should have nearly identical maximum processing rates. The key difference is that wave-pipelining reduces latency by not having to wait for the arrival of the clock to produce an output as shown in Figures 2.14b and 2.14c.

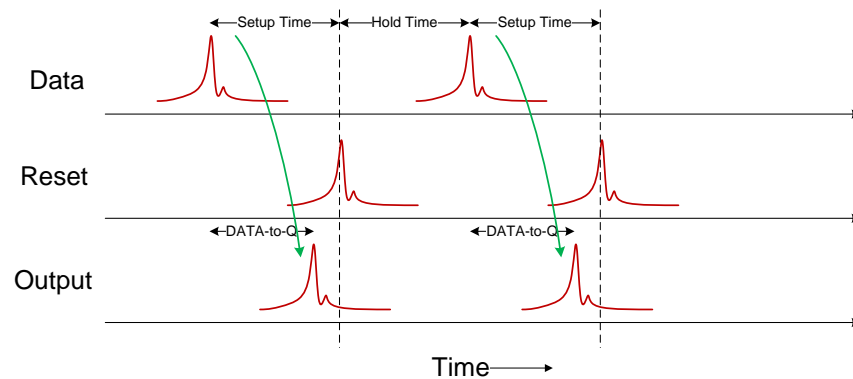
In RSFQ microprocessors, it may be necessary to integrate the two techniques in the form of asynchronous hybrid wave-pipelining, especially in irregular structures where it may not be viable to use pure wave-pipelining techniques. This hybrid approach can be accomplished by using wave-pipelining for the most time critical operations of a circuit and using co-flow clocking for non-critical areas. Furthermore, co-flow clocking can be used sparingly where logic is complex and requires the use of clock-based gates when an asynchronous equivalent is not available.



(a) Structural differences between co-flow clocking and wave-pipelining.



(b) Timing in co-flow clocking.



(c) Timing in wave-pipelining.

Figure 2.14: Co-flow clocking versus wave-pipelining.

Chapter 3

Superconductor Ripple-Carry Adder

Outline

3.1	Goals and Challenges	41
3.2	Ripple-Carry Adder Concept	42
3.3	RSFQ Study	43
3.3.1	Design Overview	43
3.3.2	Simulation Results and Discussion	46

3.1 Goals and Challenges

To gain insight on the design spectrum of adder structures implemented in superconductor logic, we first looked at the simplest case of a 32-bit ripple-carry adder (RCA). This study focuses on a single-cycle, non-pipelined, low-latency adder assuming the structure is constrained to that of an RCA using the HYPRES 1.5 μm 4.5 kA/cm² process. The bit-slice element of the RCA is the full adder which was built using T1 gates to obtain a low-complexity, low-latency structure. Because we decided to eliminate pipelining for this study, only a single binary clocking tree is required without the need to implement additional timing or buffering techniques, thus simplifying the overall design.

The work on the RCA would later on help us make important design trade-off analyses when we explore much more complex structures later on.

3.2 Ripple-Carry Adder Concept

The RCA is the simplest multi-bit adder structure yielding a fast design turnaround time. An N-bit RCA can be constructed using N simple circuit elements known as full adders (FAs), as shown in Figure 3.1 as a 4-bit example. An FA cell adds three one-bit numbers which are typically labeled as A , B , and C_{in} . A and B are the operands, and C_{in} is the carry-in bit, usually from the addition of the preceding bit. With these three inputs, two outputs are generated from the FA cell: S and C_{out} , where S is the sum of the three inputs and C_{out} is the carry-out. Logic equations (3.1) and (3.2) describe how S and C_{out} are generated respectively.

$$S = A \oplus B \oplus C_{in} \quad (3.1)$$

$$C_{out} = (A \wedge B) \vee (A \wedge C_{in}) \vee (B \wedge C_{in}) \quad (3.2)$$

By serially connecting FA cells from the C_{out} of one bit to the C_{in} of the next bit, an RCA structure is generated. The worst-case delay for this structure can be demonstrated by setting operand A to all logical 1's, operand B to all logical 0's and the C_{in} of bit 0 to logical 1. This creates a scenario where the carry must propagate from bit 0 to bit N-1, before the final result is produced. Because of the serial dependence on the carries, it is necessary to wait up to N-carry delays which makes the design relatively slow for a large number of bits. Furthermore, this delay is also roughly equivalent to the cycle time the adder can run at, if no modifications, such as the introduction of pipelining DFFs, are made. Thus, this particular design alone is not scalable for wide datapath, high-performance designs.

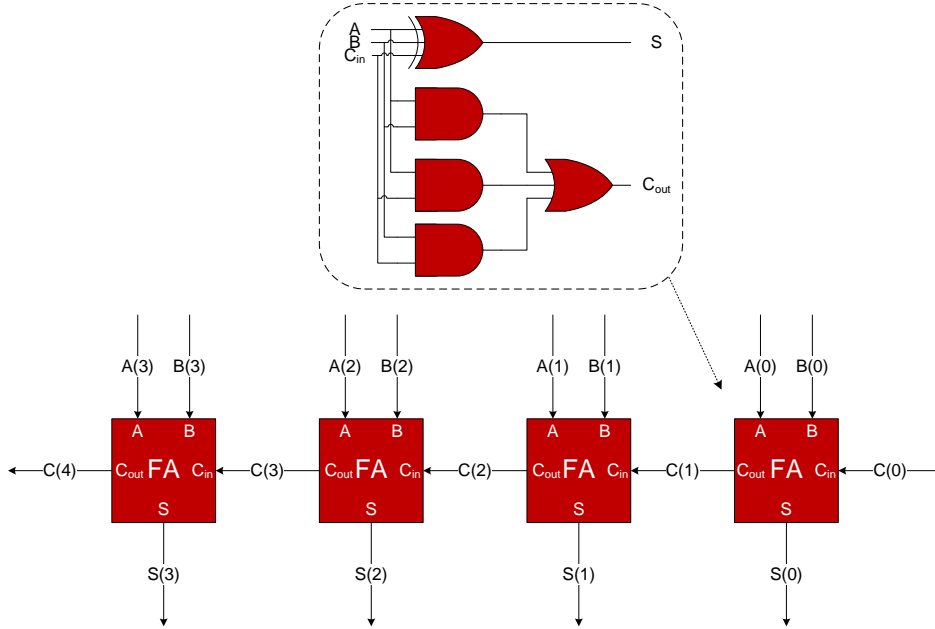


Figure 3.1: Example of a 4-bit RCA structure composed of 4 FA cells. The inset shows how one would implement an FA cell using CMOS logic gates.

3.3 RSFQ Study

3.3.1 Design Overview

Using RSFQ logic to design an RCA, we can exploit the high-speed capabilities of the logic to perform time-multiplexing on the inputs and achieve the same exact behavior of the adder. Instead of a traditional full adder as discussed in Section 3.2, we used a special gate named T1 as a counting circuit to count input pulses. This particular gate is based on one of the fastest digital circuits ever demonstrated using RSFQ technology [34]. The T1 gate has two inputs: t which is the data input, and c which is the clock input used to read-out its current state. It also has two outputs: $q0$ which is generated asynchronously, and $q1$ which is generated synchronously.

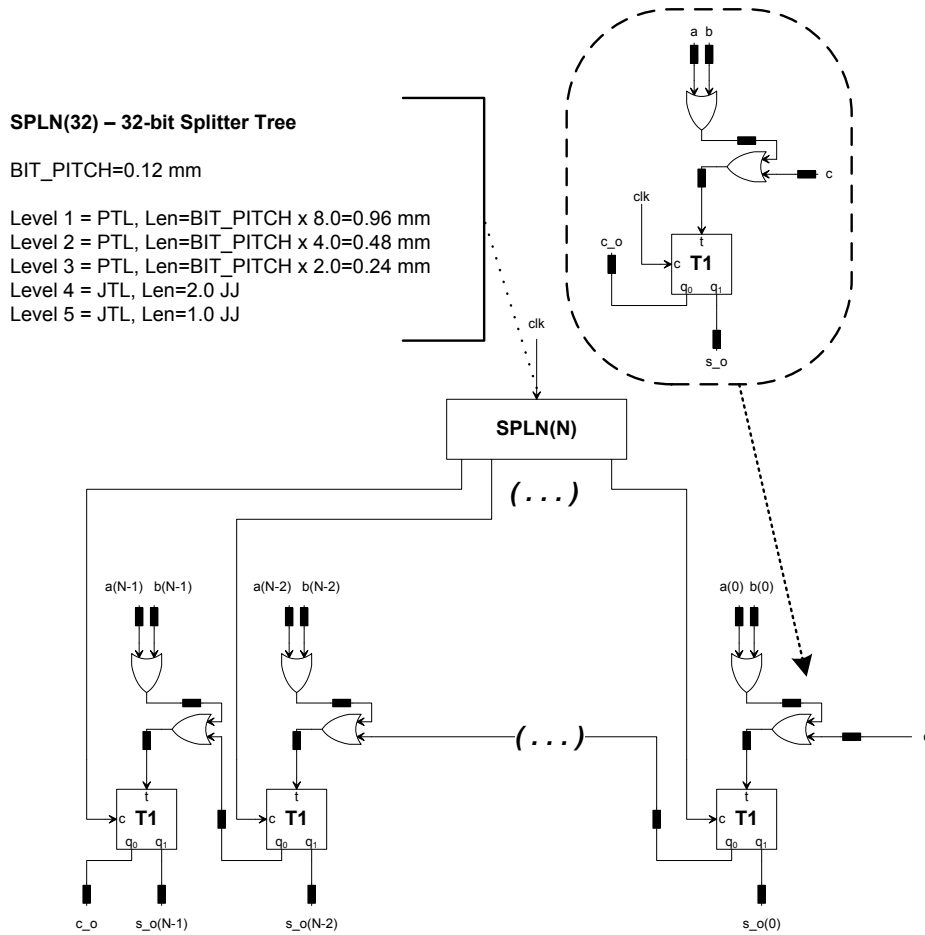


Figure 3.2: Schematic of an RSFQ RCA adder. Inset shows a single RSFQ FA composed of two Merger gates and a T1 gate.

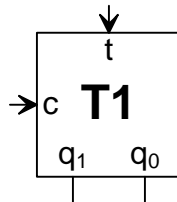


Figure 3.3: T1 symbol used in schematics.

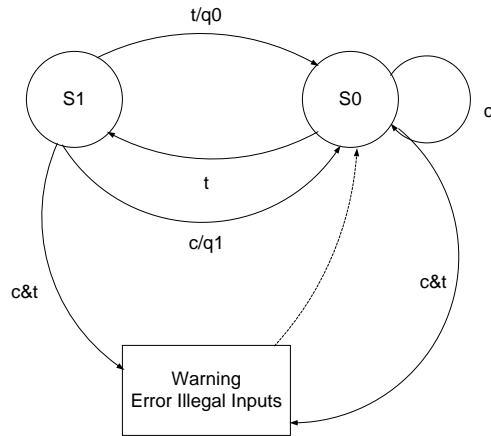


Figure 3.4: FSM of the T1 cell used in the RCA structure.

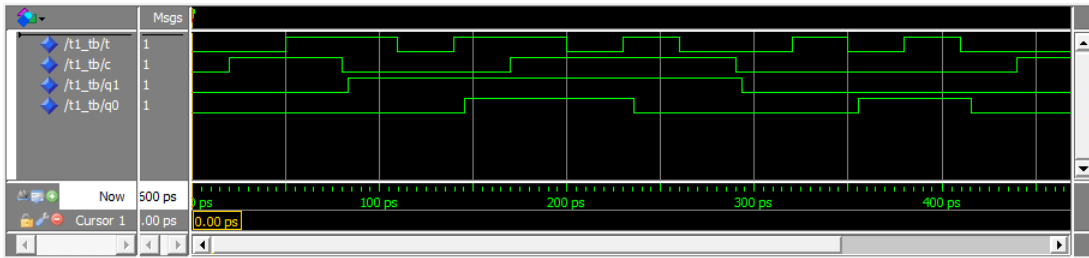


Figure 3.5: Logic simulation waveform of the T1 cell.

The behavior of the T1 gate can be followed on the FSM diagram shown in Figure 3.4 on page 45. Assuming that T1 starts at state S0, if no data pulse arrived at input t and then we apply a pulse on input c some time later (sufficiently far enough to satisfy setup time), no output pulse is generated. If a single data pulse arrived at input t and then we apply a pulse on input c , an output pulse is generated from $q1$. Now if two data pulses arrived with each pulse separated by the minimum time that T1 can process consecutive data pulses, an output pulse is generated asynchronously from $q0$ without any need for clock. T1 can be generalized in the following way:

1. Send N pulses to T1's t input, with each pulse sufficiently separated by the timing constraint of T1.
2. As T1 is processing these N pulses, it will asynchronously generate an output pulse from $q0$ on every even pulse arriving at input t (e.g. 2nd, 4th, 6th, etc. pulse).

Table 3.1: Comparing T1’s counting behavior with the full adder cell.

# of Data Pulses	Full Adder Input Equivalent	Carry-out: Output of $q0$ (asynchronous)	Sum: Output of $q1$ (after clock is applied)
0	A , B , and C_{in} are all zeroes	‘0’	‘0’
1	Only one of the three inputs is logical 1	‘0’	‘1’
2	Two of the three inputs are logical 1	‘1’	‘0’
3	All three inputs are logical 1	‘1’	‘1’

- After N pulses have been processed, a pulse on clock input c can be applied. If N is odd, then an output pulse from $q1$ will be synchronously generated, otherwise no output is generated.

The simulation waveform shown in Figure 3.5 on page 45 shows the behavior of T1 when 0, 1, 2, 3, and 4 data pulses are processed.

In the FA case, the data pulses being processed by T1 are the time-multiplexed arrivals of A , B , and C_{in} . These three inputs are merged together using a Merger gate while still maintaining their time separations along a single stream, and are sent into the t input of T1. For a given clock cycle, only 3 data pulses at most can be processed by T1 when used as a full adder. Table 3.1 on page 46 shows how T1’s counting behavior is equivalent to that of a traditional full adder. These T1 cells as well as appropriate merging circuits and delay elements can be serially connected together in the same way full adders are connected to construct an RCA. This design still deals with the problem of long delays as carry must propagate through N T1-based full adder elements. It also differs from the traditional RCA in that the sum result is read-out by applying a pulse to all of the T1 clock inputs.

3.3.2 Simulation Results and Discussion

Using our VHDL cell library tuned to the HYPRES 1.5 μm 4.5 kA/cm² process, we designed and simulated a 32-bit RSFQ RCA. The results of the sim-

ulation are summarized in Table 3.4 on page 52. Its main advantage is that its complexity is very small which results in a small amount of bias current used to power the circuit. However, these results confirm that RCAs, even when implemented using superconductor technology, do not provide the 20+ GHz processing rate we are aiming to achieve and scaling to higher data widths would decrease the rate even further. Fortunately, some concepts can be taken from this study:

- The counting T1 circuit is applicable in high-speed multipliers as a form of compressors [110].
- Small RCA chains can be used in non-critical paths of a hybrid sparse-tree adder (Chapter 5).

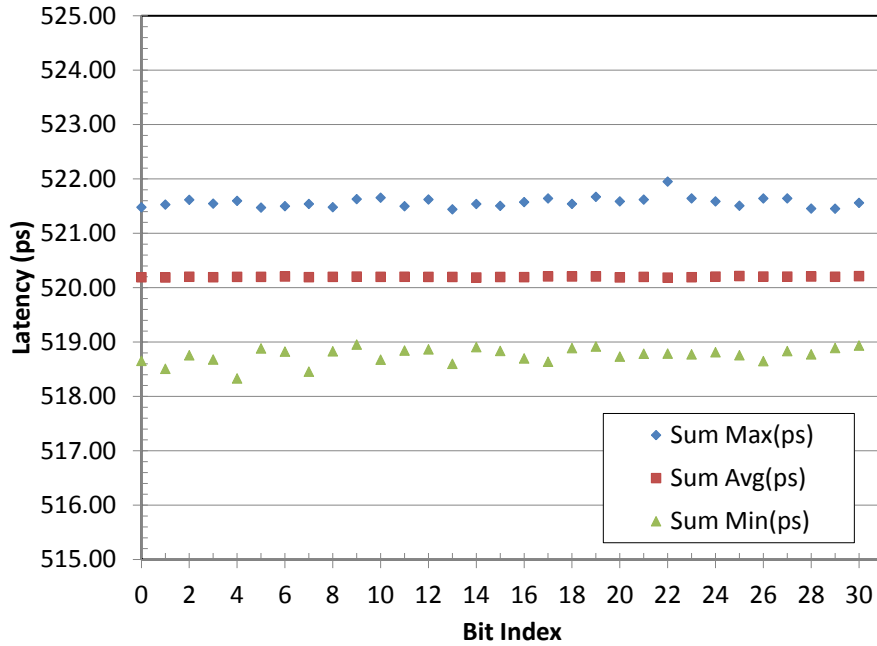
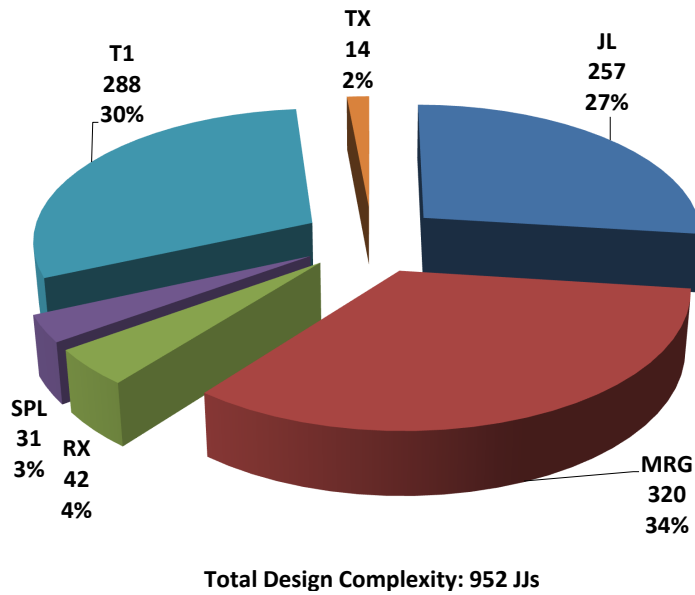


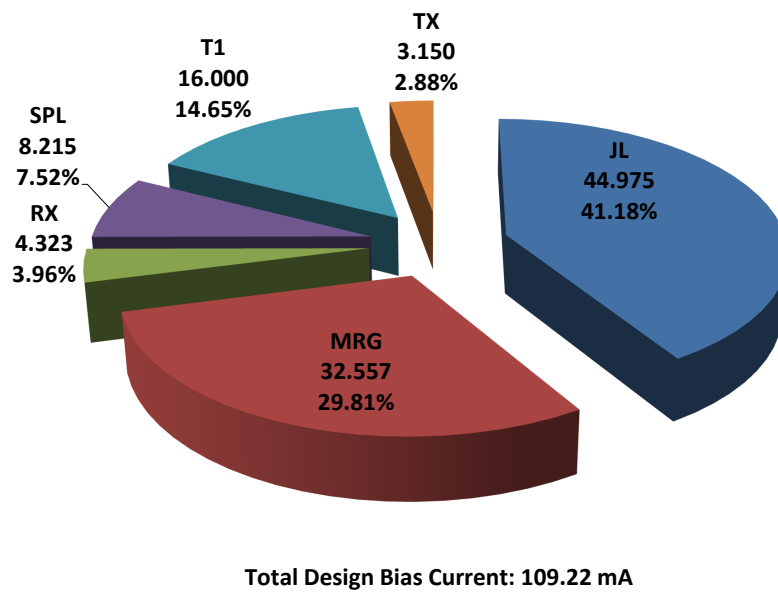
Figure 3.6: Latency distribution of the 32-bit RCA. The least significant bit is bit index 0. The latency is measured from the assertion of input signals to the arrival of outputs at the “sum” port.

Table 3.2: Latency distribution of the 32-bit RCA with the average latencies calculated across all bits.

Bit	Sum Max. (ps)	Sum Avg. (ps)	Sum Min. (ps)
31	521.67	520.20	518.88
30	521.56	520.21	518.94
29	521.45	520.20	518.89
28	521.46	520.21	518.77
27	521.64	520.20	518.83
26	521.64	520.20	518.65
25	521.51	520.21	518.76
24	521.59	520.20	518.81
23	521.64	520.19	518.77
22	521.95	520.18	518.79
21	521.62	520.20	518.79
20	521.59	520.19	518.73
19	521.67	520.21	518.92
18	521.54	520.21	518.89
17	521.64	520.21	518.64
16	521.57	520.19	518.70
15	521.51	520.20	518.84
14	521.54	520.19	518.91
13	521.44	520.20	518.60
12	521.62	520.20	518.87
11	521.50	520.20	518.84
10	521.66	520.20	518.68
9	521.63	520.20	518.95
8	521.48	520.20	518.83
7	521.54	520.19	518.46
6	521.50	520.21	518.83
5	521.47	520.20	518.88
4	521.60	520.20	518.33
3	521.55	520.19	518.68
2	521.62	520.20	518.76
1	521.53	520.19	518.51
0	521.48	520.19	518.65
Average	521.57	520.20	518.76

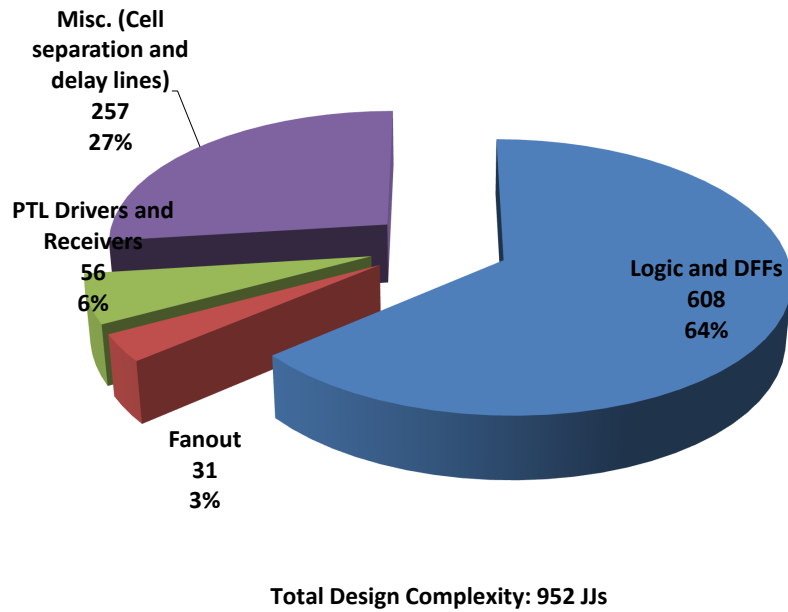


(a) JJ design complexity breakdown of the 32-bit RCA.

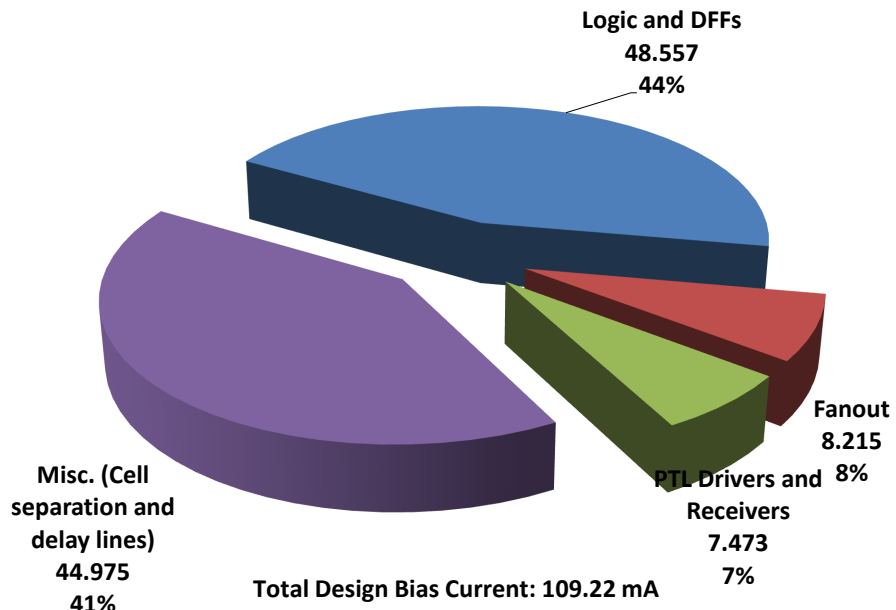


(b) Design bias current breakdown of the 32-bit RCA.

Figure 3.7: Cell-wise breakdown of the 32-bit RCA for both the complexity and bias current of the logical design. The results do not include additional JJs to distribute bias current in ERSFQ logic.



(a) Categorical JJ design complexity breakdown of the 32-bit RCA.



(b) Categorical design bias current breakdown of the 32-bit RCA.

Figure 3.8: Categorical breakdown of the 32-bit RCA for both the complexity and bias current of the logical design.

Table 3.3: Breakdown of bias JJs for the 32-bit RCA for ERSFQ logic. The raw bias JJ count assumes all cells are using non-sharing bias JJs. The adjusted (adj.) bias JJ count assumes that 20% of all Josephson transmission lines (JL cells) are connected to non-sharing bias JJs and the remaining 80% of JL cells share 1 bias JJ for every 2 JL cells.

Cell	Cell Count	Bias JJs/Cell	Total Raw Bias JJs/Cell	Total Adj. Bias JJs/Cell	% Raw Bias JJs/Cell	% Adj. Bias JJs/Cell
JL	257	20%=1, 80%=1/2	257	154	56.11%	43.38%
MRG	64	1	64	64	13.97%	18.03%
RX	14	2	28	28	6.11%	7.89%
SPL	31	1	31	31	6.77%	8.73%
T1	32	2	64	64	13.97%	18.03%
TX	14	1	14	14	3.06%	3.94%
Total:	412		458	355	100.00%	100.00%

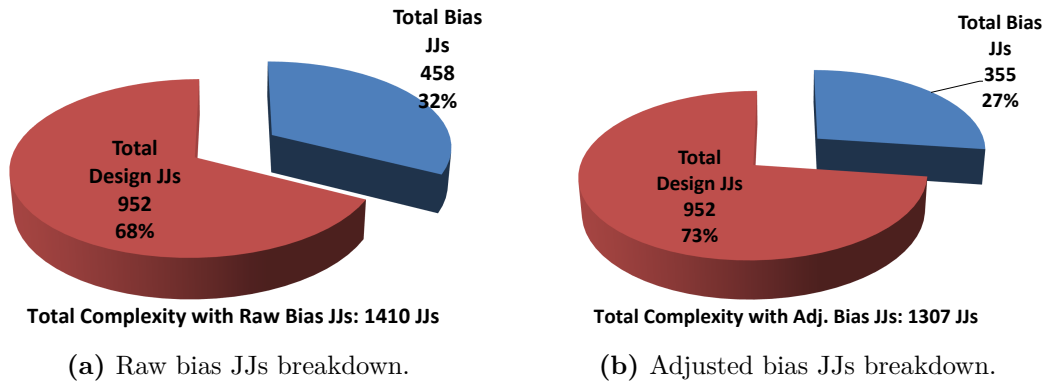


Figure 3.9: Breakdown comparison of both raw and adjusted bias JJ counts with respect to the total design complexity of the 32-bit RCA implemented in ERSFQ logic.

Table 3.4: Summary of the key simulation results for the 32-bit RCA.

(a) Overall summary of the 32-bit RCA.

Data Width	32-bit
Bit Pitch	0.120 mm
Design Complexity	952 JJs
Overall Avg. Latency	520 ps
Max. Processing Rate	2.0 GHz
Total Bias Current	0.109 A

(b) Power related metrics for both RSFQ and ERSFQ implementations of the 32-bit RCA at 4.2 K temperature.

Logic	RSFQ	ERSFQ
Bias Voltage	2.6 mV	4.0 μ V
Total Power	0.284 mW	0.827 μ W
Ops./Watt	7.046 TOPS/W	2418.380 TOPS/W
Energy/Op.	0.142 pJ/op.	0.414 fJ/op.

Chapter 4

Superconductor Kogge-Stone Adder and ALU

Outline

4.1	Goals and Challenges	53
4.2	General Kogge-Stone Adder Structure	54
4.3	RSFQ Study	58
4.3.1	Design Overview	58
4.3.2	Simulation Results and Discussion	65
4.4	Joint SBU-HYPRES Project: An 8-bit Kogge-Stone ALU Implementation Using the 1.5 μm 4.5 kA/cm² HYPRES Process	73
4.4.1	Design Flow	73
4.4.2	Simulation Results	75
4.4.3	Low-Frequency Testing	77
4.4.4	High-Frequency Testing	81

4.1 Goals and Challenges

In the previous chapter, we explored the performance, power and design complexity of the simplest adder structure in superconductor logic. In this chapter, our study is focused on achieving a high-performance (20+ GHz processing rate), pipelined, wide datapath adder that is easy to extend into an ALU. To this end, we have chosen the Kogge-Stone adder (KSA), a parallel prefix carry look-ahead adder. To achieve high-performance, we combined the KSA's

parallel algorithm with a foundation that was designed from the ground up to use wave-pipelining as the vehicle for propagating data signals along the stages of the adder. Using a modular, hierarchical design flow, we developed a Kogge-Stone ALU (KSALU) design that kept internal interfaces exactly the same while only introducing new control signals at the very first stage of the structure. Because the ALU block is intrinsically not suitable for wave-pipelining, we used co-flow clocking strictly within the ALU and relied on wave-pipelining to propagate the signals through the remaining stages to give us an asynchronous hybrid wave-pipelined design.

The aforementioned techniques we used to achieve our design goals were then put to the test in a joint project between SBU and HYPRES where a scaled down implementation of a 20 GHz 8-bit KSALU was completed and demonstrated successfully.

The study of the KSALU allowed us to quantify the characteristics of a high-performance, high-complexity design in contrast to the low-performance, low-complexity RCA.

4.2 General Kogge-Stone Adder Structure

The KSA is a parallel prefix carry look-ahead adder whose concept was developed by Peter Kogge and Harold Stone in 1973 [111]. Today, its structure and its variants are commonly found in high-performance adders in the industry [24, 23]. The primary advantage of this adder is that it can generate carry signals in $O(\log(N))$ time, unlike the RCA which generates them in $O(N)$ time. The adder can be broken down into 3 types of stages:

1. Initialization - Generates bitwise prefix signals. It is the first stage of the KSA.
2. Prefix Tree - Merges prefix signals in a logarithmic fashion to produce group carries. There are $\lceil \log_2(N) \rceil$ number of stages of this type, where N is total number of bits in the adder.
3. Summation - Produces the final sum result. It is the final stage of the KSA.

Table 4.1 on page 56 shows the Boolean equations used to generate the various signals in the KSA using triple-rail encoding. Subscript j refers to the stage index starting with $j = 0$ for the Initialization stage down to the Summation stage where $j = (\lceil \log_2(N) \rceil + 2) - 1$. Subscript i refers to the bit column index where the most significant bit (MSB) is designated as $i = N - 1$ and the least significant bit (LSB) is designated as $i = 0$. The equations use the convention

in which lower-case names refer to bitwise signals and upper-case names refer to group signals. Figure 4.1 on page 57 is a general diagram of a 16-bit KSA. The different cells in the diagram are defined as the following:

- Red Cells - They form the Initialization stage and produce prefix signals for each bit.
- Green Cells - They merge group prefix signals but they no longer have to send their outputs horizontally, only vertically downward to the next stage.
- Black Cells - They merge group prefix signals and they send their outputs in both vertical and horizontal directions.
- Gray Cells - They do not merge group prefix signals but rather pipeline group prefix signals that do not need to be merged anymore. They send their outputs in both vertical and horizontal directions.
- Blue Cells - They are almost the same as Gray Cells except they only transmit the group carry signal to the $i + 1$ bit, while transmitting its p_i signal vertically downward.
- Orange Cells - They form the Summation stage and they calculate the sum for each bit.

While the KSA is synonymous with high performance, it also requires a lot of carry-merge logic (Green and Black Cells) and it has heavy wiring congestion, making it very difficult to implement on an actual chip. Chapter 5 will discuss a variant of the Kogge-Stone structure which ameliorates these shortcomings.

Table 4.1: The Boolean equations for the Kogge-Stone adder using triple-rail encoding.

Initialization Signals	
Carry generate	$g_i = a_i \wedge b_i$
Carry propagate	$p_i = a_i \oplus b_i$
Carry kill	$k_i = \neg(a_i \vee b_i)$
Prefix Tree Signals	
Group carry generate	$G_{j,i} = G_{j-1,i} \vee (P_{j-1,i} \wedge G_{j-1,i-2^{j-1}})$
Group carry propagate	$P_{j,i} = P_{j-1,i} \wedge P_{j-1,i-2^{j-1}}$
Group carry kill	$K_{j,i} = K_{j-1,i} \vee (P_{j-1,i} \wedge K_{j-1,i-2^{j-1}})$
Summation Signals	
Sum (non-LSB)	$s_i = p_i \oplus G_{j-1,i-1}$
Sum (LSB)	$s_i = p_i$

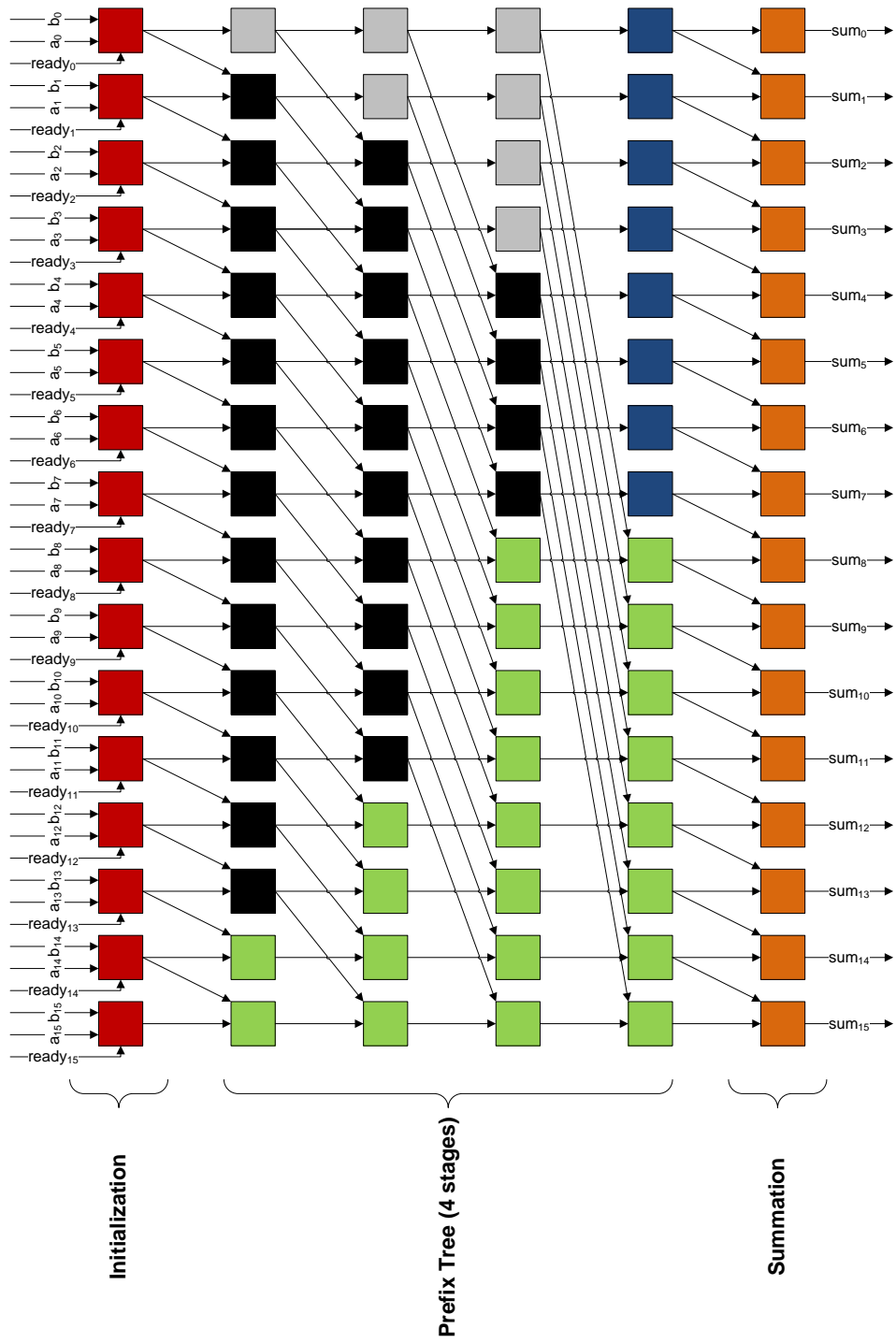


Figure 4.1: Example of a 16-bit KSA structure.

4.3 RSFQ Study

4.3.1 Design Overview

The RSFQ implementation of the KSA fully exploits the benefits of using asynchronous hybrid wave-pipelining techniques. As a multi-stage design, it would have been necessary to provide a complex clock distribution network at each stage of the pipeline if we did not use wave-pipelining. Instead, only the first stage of the adder needs to be clocked and then the data will asynchronously propagate through the remaining stages. Using the CFF gate discussed in Section 2.2, we have an element that can act as both a logic operation (logical AND in the Boolean equations) and a synchronization mechanism to help ensure parts of the wave will not get too far ahead or lag behind. The prefix tree structure features a maximum fan-out of only 2, which is a property very beneficial to RSFQ as splitting requires active circuit elements (pulse splitters) which contribute to the overall delay.

Originally, the RSFQ design of the KSA made full use of all the types of signals listed in Table 4.1 on page 56. However, the design can be simplified further by eliminating the “Kill” signal and instead replace it with a reset wave that is always generated for each data wave. This reset signal trails behind the data wave as the pair propagates through the stages of the adder. As data signals travel through the CFFs, the reset signal that follows them would clean or reset the CFFs for the next data wave. Finally, the reset wave acts as a clock pulse at the Summation stage to read out the sum result from synchronous XOR gates. The logic schematics for the different blocks are shown in Figure 4.2 on page 62 and Figure 4.3 on page 63.

The Kogge-Stone core can also be re-used to design an ALU. By performing

Table 4.2: Full adder decomposition into logical functions using Cin as a control signal.

Cin	A	B	G	P	Logic of G	Logic of P
0	0	0	0	0	$A \wedge B$ (AND)	$A \oplus B$ (XOR)
0	0	1	0	1		
0	1	0	0	1		
0	1	1	1	0		
1	0	0	0	1	$A \vee B$ (OR)	$\overline{A \oplus B}$ (XNOR)
1	0	1	1	0		
1	1	0	1	0		
1	1	1	1	1		

conditional in-place inversions of the two operands via `Inv_a` or `Inv_b` signals, and using a TRS gate (a gate similar to T1 except it can be preset to the state of S0 or S1, and both outputs are asynchronous) to count pulses like an FA (Figure 4.2a on page 62), we can execute a full set of logical operations as well as both addition and subtraction. The logical operations are obtained by decomposing the outputs of the TRS-based pulse counter as shown in Table 4.2 on page 58. The counter produces preliminary G and P signals from which we can obtain AND and XOR core functions if we set the Cin to logical ‘0’ or OR and XNOR core functions if we set the Cin to logical ‘1’. We preset the TRS gate if we want Cin = ‘1’, or we reset it if we want Cin = ‘0’ through the dual-rail `Ctrl_sub/Ctrl_sub_bar` control signals. These logic operators are called core functions because from this set, we can obtain additional logical operations such as NOR and NAND by controlling the inversion of operands. Assuming the TRS is initialized to some Cin value, we must select which logical operator to use from the pair of core functions. We achieve this by multiplexing the preliminary G and P signals with the `Ctrl_xor` signal. If `Ctrl_xor = ‘1’`, then we use the logic operator provided by the preliminary P signal, otherwise we use the logic operator provided by the preliminary G signal. Finally, we must route the logical function to the top-level `P_o` port of the ALU_INIT block by setting the `Ctrl_add` signal to ‘0’. Through the `P_o` port, the result will propagate straight down the remaining stages of the ALU instead of going into the Prefix Tree.

In the case of arithmetic, we must route the preliminary G and P signals directly to the `G_o` and `P_o` ports respectively by setting the `Ctrl_add` signal to ‘1’. All arithmetic operations are based on addition. In the case of subtraction, we perform 2’s complement addition by inverting one of the operands, add +1, and add the other operand in one step as outlined in Equations 4.1 and 4.2.

$$A - B \Leftrightarrow A + \overline{B} + 1 \quad (4.1)$$

$$B - A \Leftrightarrow B + \overline{A} + 1 \quad (4.2)$$

To invert the appropriate operand, we simply set `Ctrl_inv_a` or `Ctrl_inv_b` as necessary. To add +1, we have a special set of control signals reserved for the least significant bit of the ALU to set/reset the TRS called `Lsb_ctrl_sub / Lsb_ctrl_sub_bar`. With the freedom to invert operands and add +1, we obtain the following list of arithmetic operations:

1. ADD A, B $\Leftrightarrow A + B$
2. ADD \neg A, B $\Leftrightarrow B - A - 1$

3. ADD A, $\neg B \Leftrightarrow A - B - 1$
4. ADD $\neg A$, $\neg B \Leftrightarrow -(A + B + 2)$
5. INC_ADD A, B $\Leftrightarrow A + B + 1$
6. SUB A, B $\Leftrightarrow A - B$
7. SUB B, A $\Leftrightarrow B - A$
8. INC_ADD $\neg A$, $\neg B \Leftrightarrow -(A + B + 1)$

All these steps occur within the ALU_INIT block that resides in the Initialization stage. Since the output interface of ALU_INIT is exactly the same as the GPR_INIT block used for a standalone adder, the Prefix Tree does not need to be modified. We only need to supply the additional control signals that the ALU_INIT blocks require.

Table 4.3 on page 64 shows the full list of ALU operations and their associated control signals. The ALU control signals are summarized below:

- Inv_A - If there is a pulse on this signal, it inverts operand A.
- Inv_B - If there is a pulse on this signal, it inverts operand B.
- Ctrl_add - If there is a pulse on this signal, bitwise prefix signals are routed into the Prefix Tree logic.
- Ctrl_xor - If there is a pulse on this signal, XOR-based logic is applied to the operands and sent as p_i . Otherwise, AND-based logic is performed and sent as p_i .
- Ctrl_sub - If there is a pulse on this signal, subtraction-based logic is applied (presetting the TRS gate, or in other words performing “+1” in 2’s complement). For the least significant bit, a Lsb_ctrl_sub signal is used.
- Ctrl_sub_bar - The complement of Ctrl_sub. Also has a corresponding Lsb_ctrl_sub_bar for the least significant bit.

Definition of ports:

- Rdy - The ‘ready’ signal that starts the asynchronous operation of the unit.
- A - Operand input A.
- B - Operand input B.
- Gv/Pv/Rv - Vertical group signals for carry generate, carry propagate and reset, respectively. They form the vertical lines as shown in Figure 4.1 on page 57.

- Gh/Ph/Rh - Horizontal group signals for carry generate, carry propagate and reset, respectively. They form the diagonal lines as shown in Figure 4.1 on page 57.
- pi - Bitwise carry propagate signal. It is calculated during the Initialization stage and is buffered through the remaining stages of the unit until the Summation stage where it is used to calculate the final sum.

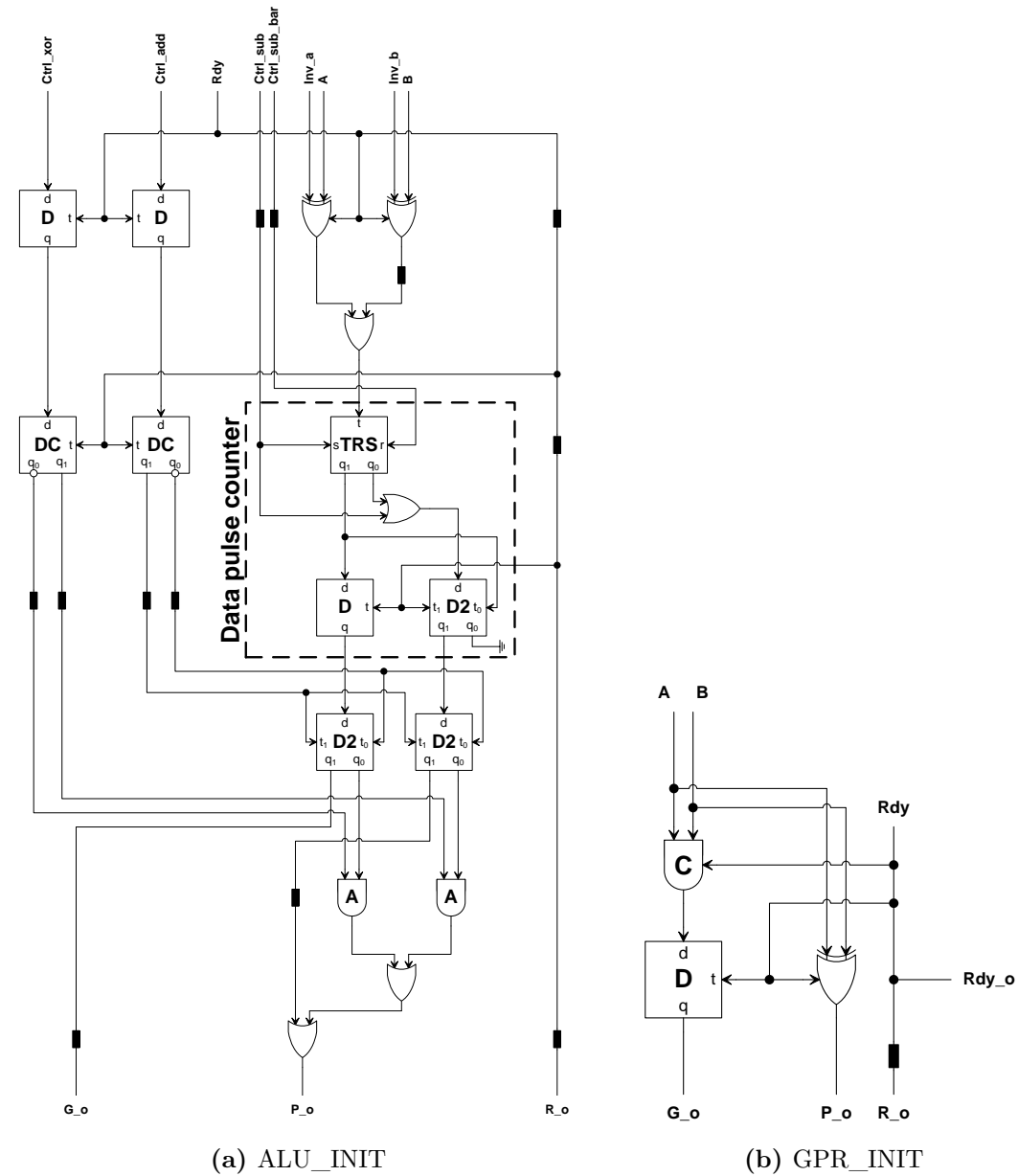


Figure 4.2: INIT blocks which can be logically interchanged to obtain either an ALU (a) or an adder (b). These are the Red cells that reside in the Initialization stage.

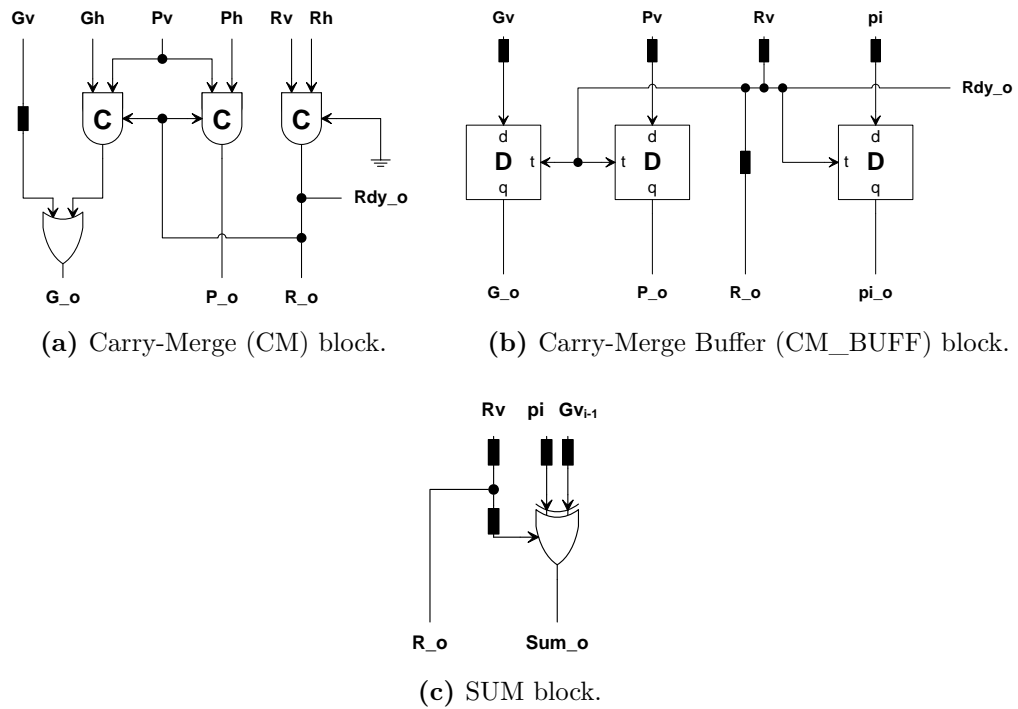


Figure 4.3: Logical schematics for the prefix tree and summation blocks. The Green and Black cells of the Prefix Tree are built using CM blocks (a). The Gray and Blue cells are built using CM_BUFF blocks (b). The Orange cells in the Summation stage are built using SUM blocks (c).

Table 4.3: Instructions decoded into the control signals for the ALU.

ALU Operation	Ctrl_sub	Ctrl_sub_bar	Ctrl_add	Ctrl_xor	Inv_a	Inv_b	Lsb_ctrl_sub	Lsb_ctrl_sub_bar
NOP	0	1	0	0	0	0	0	1
SETIS	0	1	0	0	1	1	0	1
XOR A, B	0	1	0	1	0/1	0/1	0	1
XNOR A, B	0	1	0	1	0/1	1/0	0	1
AND A, B	0	1	0	0	0	0	0	1
AND A, \neg B	0	1	0	0	0	1	0	1
AND \neg A, B	0	1	0	0	1	0	0	1
NOR A, B	0	1	0	0	1	1	0	1
OR A, B	1	0	0	0	0	0	1	0
OR A, \neg B	1	0	0	0	0	1	1	0
OR \neg A, B	1	0	0	0	1	0	1	0
NAND A, B	1	0	0	0	1	1	1	0
ADD A, B	0	1	1	X	0	0	0	1
ADD A, \neg B	0	1	1	X	0	1	0	1
ADD \neg A, B	0	1	1	X	1	0	0	1
ADD \neg A, \neg B	0	1	1	X	1	1	0	1
INC_ADD A, B	0	1	1	X	0	0	1	0
SUB A, B	0	1	1	X	0	1	1	0
SUB B, A	0	1	1	X	1	0	1	0
INC_ADD \neg A, \neg B	0	1	1	X	1	1	1	0

Table 4.4: Listing of PTL interconnect lengths for the 32-bit KSALU.

Stage-to-Stage	Horizontal PTL Length (mm)	Vertical PTL Length (mm)	Total PTL Length (mm)
0 to 1	0.280	0.220	0.5
1 to 2	0.560	0.190	0.75
2 to 3	1.120	0.130	1.25
3 to 4	2.240	0.130	2.37
4 to 5	4.480	0.130	4.61
5 to 6	0.000	0.080	0.08
Total Length (mm)	8.680	0.880	9.56
Total Delay (ps)	86.800	8.800	95.6

4.3.2 Simulation Results and Discussion

Using our VHDL cell library tuned to the HYPRES 1.5 μm 4.5 kA/cm² process, we designed and simulated a 32-bit KSALU. The key results of the simulation are summarized in Table 4.8 on page 72. While latency is comparable to that of the RCA, its main advantage is its high-throughput clock rate due to the Kogge-Stone structure and utilization of asynchronous hybrid wave-pipelining methodologies. Its complexity and bias current is quite large however, and with its dense tree structure, it is difficult to layout for wide data widths. Chapter 5 will discuss a variation of the Kogge-Stone structure and how it is able to improve upon these challenges.

Table 4.5: Processing rate of the 32-bit KSALU. At each processing rate, an initial set of 2000 randomized vectors mixed with worst-case tests is sent into the KSALU to obtain a first-pass sweep of the processing rate. At the maximum passing rate, a second set of 10,000 vectors is sent to exercise the circuit further.

Processing Rate (GHz)	Number Failed Waves	Total Number of Waves
20.0	0	2000
20.4	0	2000
20.8	0	2000
21.3	0	2000
21.7	0	2000
22.2	0	2000
22.7	0	2000
23.3	0	10000
23.8	2	2000
24.4	627	2000
25.0	1997	2000
25.6	2000	2000
26.3	2000	2000

Table 4.6: Latency distribution of the 32-bit KSALU with the average latencies calculated across all bits.

Bit	Sum Max. (ps)	Sum Avg. (ps)	Sum Min. (ps)
31	475.48	471.97	469.00
30	475.05	471.75	469.11
29	475.00	471.55	468.84
28	474.74	471.40	468.45
27	474.44	471.28	468.45
26	474.62	471.26	468.05
25	474.52	471.24	468.02
24	475.40	471.22	467.96
23	474.81	471.22	468.49
22	474.79	471.02	467.97
21	474.46	470.78	467.77
20	474.74	470.63	467.52
19	474.58	470.48	467.44
18	473.97	470.41	467.27
17	473.88	470.38	467.34
16	474.16	470.35	467.02
15	474.00	470.37	467.02
14	473.79	470.13	466.97
13	473.36	469.90	466.58
12	473.34	469.72	465.94
11	473.18	469.56	466.38
10	473.14	469.50	465.64
9	472.96	469.44	466.06
8	473.09	469.41	464.98
7	473.46	469.45	465.53
6	473.27	469.12	465.70
5	472.20	468.68	464.94
4	472.70	468.36	464.70
3	472.07	467.90	463.87
2	471.98	467.52	463.64
1	471.18	467.02	463.24
0	470.70	466.48	462.64
Average	473.72	469.98	466.64

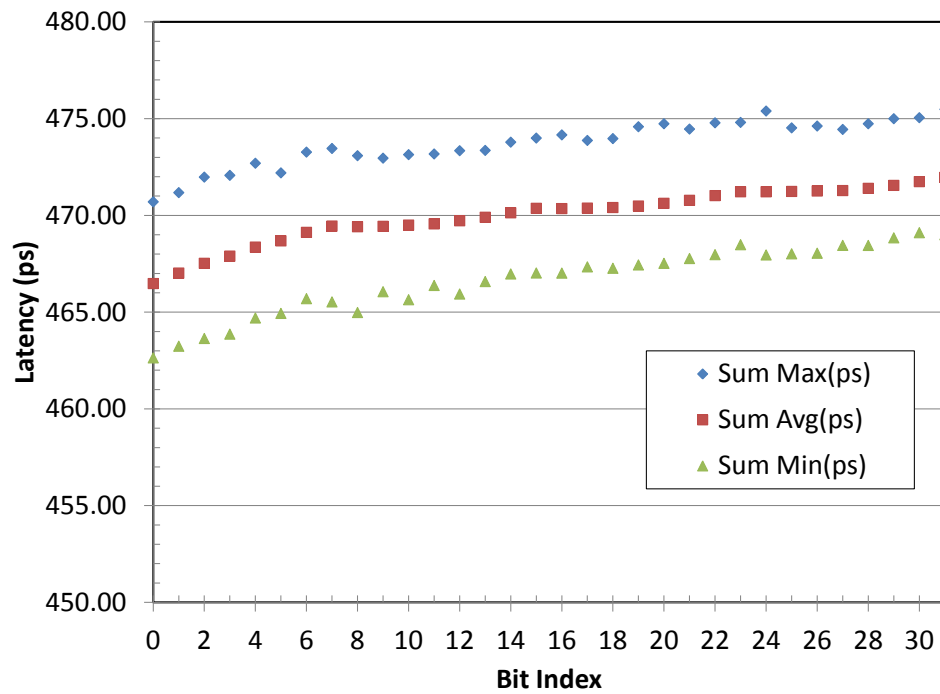
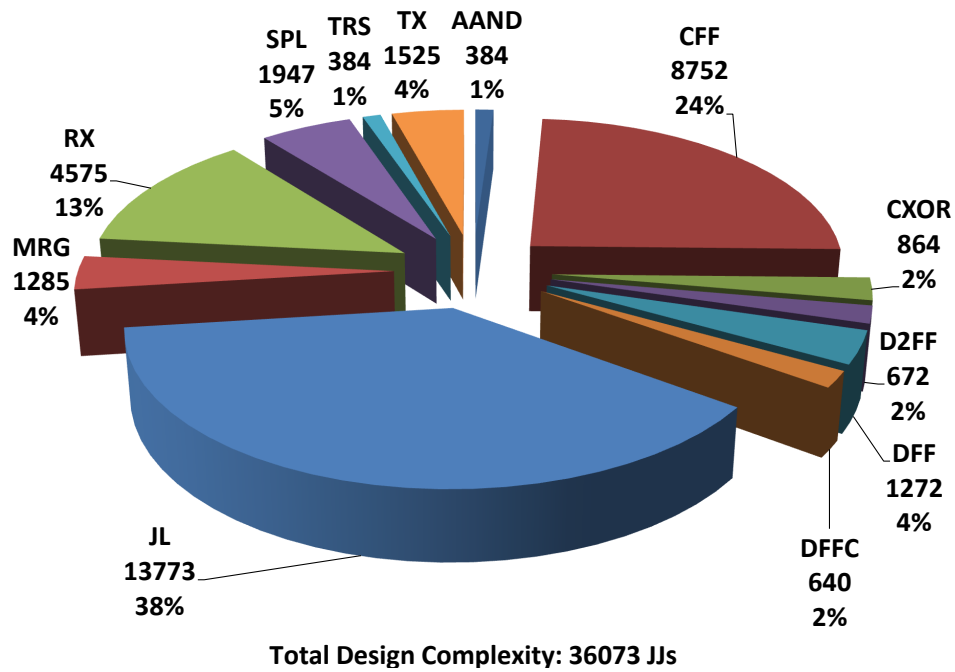
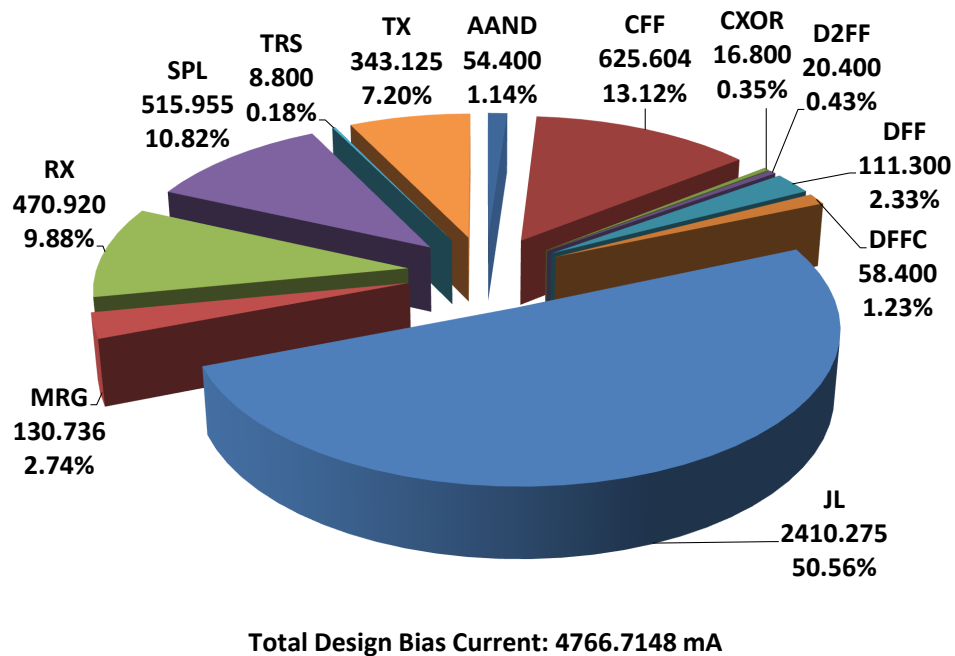


Figure 4.4: Latency distribution of the 32-bit KSALU. The least significant bit is bit index 0. The latency is measured from the assertion of the “ready” signal to the arrival of outputs at the “sum” port.

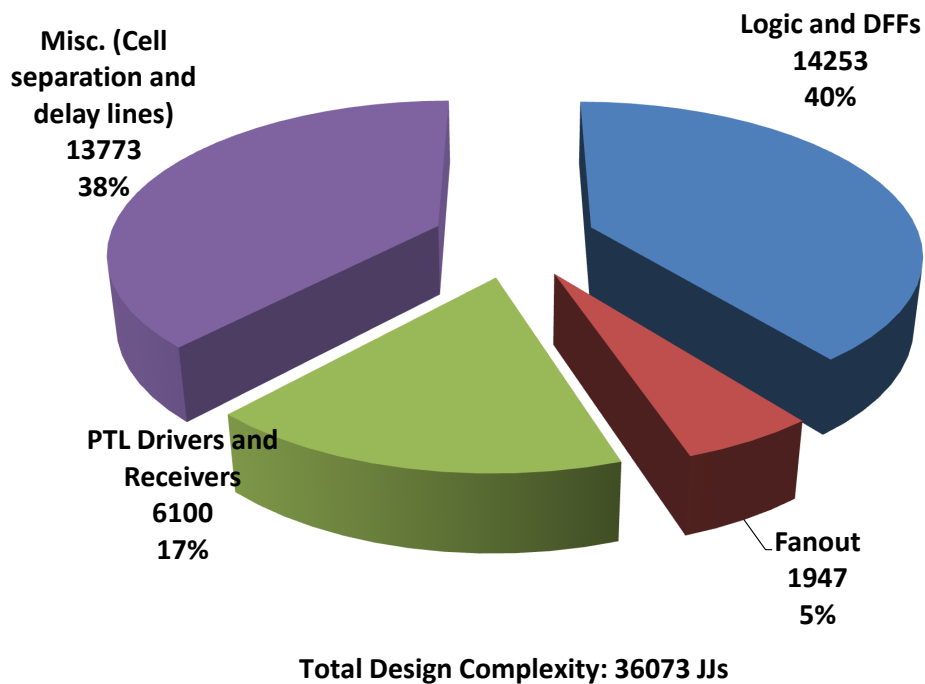


(a) JJ design complexity breakdown of the 32-bit KSALU.

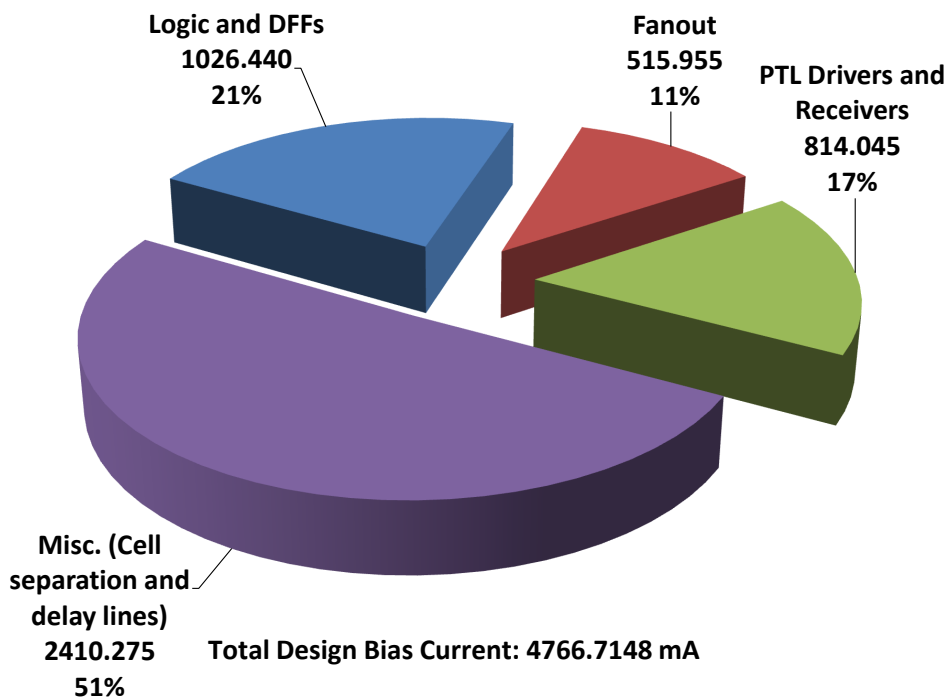


(b) Design bias current breakdown of the 32-bit KSALU.

Figure 4.5: Cell-wise breakdown of the 32-bit KSALU for both the complexity and bias current of the logical design. The results do not include additional JJs to distribute bias current in ERSFQ logic.



(a) Categorical JJ design complexity breakdown of the 32-bit KSALU.



(b) Categorical design bias current breakdown of the 32-bit KSALU.

Figure 4.6: Categorical breakdown of the 32-bit KSALU for both the complexity and bias current of the logical design.

Table 4.7: Breakdown of bias JJs for the 32-bit KSALU for ERSFQ logic. The raw bias JJ count assumes all cells are using non-sharing bias JJs. The adjusted (adj.) bias JJ count assumes that 20% of all Josephson transmission lines (JL cells) are connected to non-sharing bias JJs and the remaining 80% of JL cells share 1 bias JJ for every 2 JL cells.

Cell	Cell Count	Bias JJs/Cell	Total Raw Bias JJs/Cell	Total Adj. Bias JJs/Cell	% Raw Bias JJs/Cell	% Adj. Bias JJs/Cell
AAND	64	1	64	64	0.26%	0.33%
CFE	547	6	3282	3282	13.10%	16.80%
CXOR	96	2	192	192	0.77%	0.98%
D2FF	96	1	96	96	0.38%	0.49%
DFE	318	2	636	636	2.54%	3.26%
DFFC	64	3	192	192	0.77%	0.98%
JL	13773	20%=1, 80%=1/2	13773	8264	54.99%	42.30%
MRG	257	1	257	257	1.03%	1.32%
RX	1525	2	3050	3050	12.18%	15.61%
SPL	1947	1	1947	1947	7.77%	9.97%
TRS	32	1	32	32	0.13%	0.16%
TX	1525	1	1525	1525	6.09%	7.81%
Total:	20244		25046	19537	100.00%	100.00%

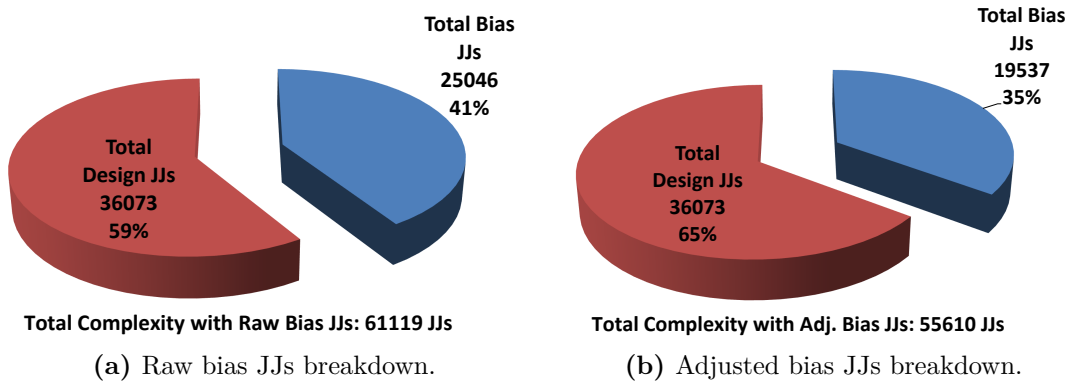


Figure 4.7: Breakdown comparison of both raw and adjusted bias JJ counts with respect to the total design complexity of the 32-bit KSALU implemented in ERSFQ logic.

Table 4.8: Summary of the key simulation results for the 32-bit KSALU.

(a) Overall summary of the 32-bit KSALU.

Data Width	32-bit
Bit Pitch	0.280 mm
Design Complexity	36,073 JJs
Overall Avg. Latency	470 ps
Max. Processing Rate	23.3 GHz
Total Bias Current	4.767 A

(b) Power related metrics for both RSFQ and ERSFQ implementations of the 32-bit KSALU at 4.2 K temperature.

Logic	RSFQ	ERSFQ
Bias Voltage	2.6 mV	46.6 μ V
Total Power	12.507 mW	224.895 μ W
Ops./Watt	1.8463 TOPS/W	103.604 TOPS/W
Energy/Op.	0.537 pJ/op.	9.652 fJ/op.

4.4 Joint SBU-HYPRES Project: An 8-bit Kogge-Stone ALU Implementation Using the 1.5 μm 4.5 kA/cm² HYPRES Process

4.4.1 Design Flow

As part of the joint SBU-HYPRES project, a scaled down version of the RSFQ KSALU study is to be physically implemented and demonstrated using the HYPRES 1.5 μm 4.5 kA/cm² fabrication process (Table 2.1 and Figure 2.5). A simpler ALU design (Figure 4.8 on page 74) was chosen to reduce the number of ALU opcode bits from 5 to 4, which reduced the number of signal lines per bit. Table 4.9 lists the control signals for the ALU operations. Working closely with HYPRES circuit designers, feedback was provided to ensure that the physical implementation was consistent with our VHDL models. This typically involved adjusting JTL lengths as needed in the design and making sure PTL wire lengths were chosen correctly (Table 4.10).

Table 4.10: PTL length and delay breakdown for each stage of the 8-bit ALU.

Stage-to-Stage	Horizontal PTL Length (mm)	Vertical PTL Length (mm)	Total PTL Length (mm)
0 to 1	0.560	0.440	1.000
1 to 2	1.120	0.380	1.500
2 to 3	2.240	0.260	2.500
3 to 4	0.000	0.080	0.080
Total Length (mm)	3.920	1.160	5.080
Total Delay (ps)	39.20	11.60	50.80

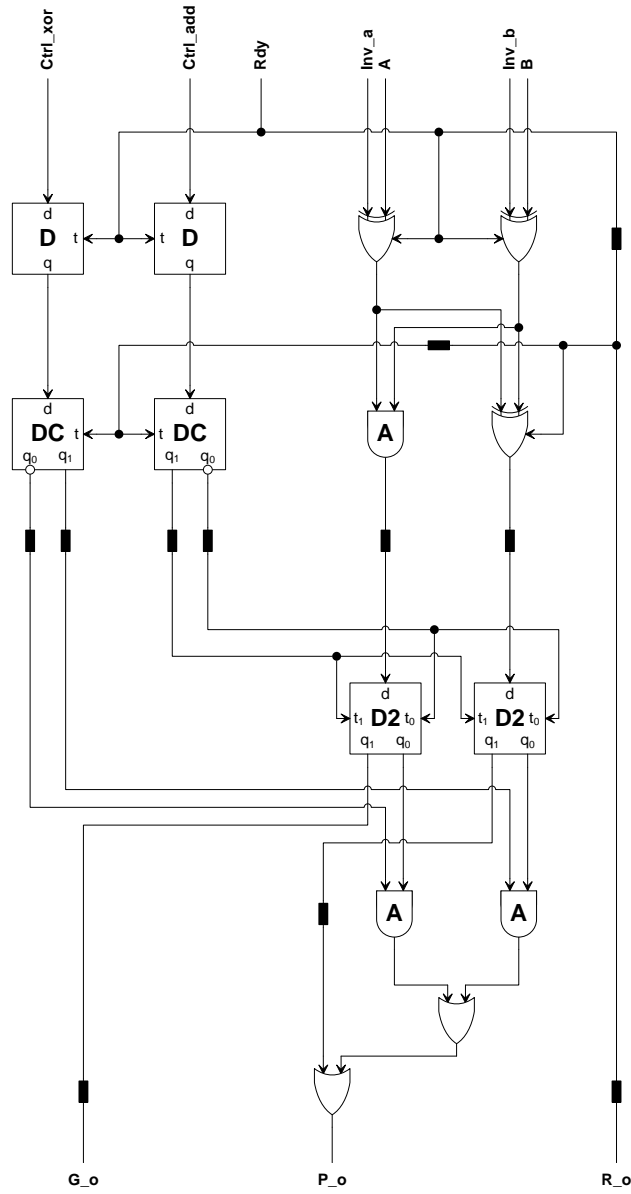


Figure 4.8: ALU_INIT for the 8-bit ALU with HYPRES.

Table 4.9: Instructions decoded into direct control signals for the ALU.

ALU Operation	Ctrl_add	Ctrl_xor	Inv_A	Inv_B
ADD	1	x	0	0
ADD INVERT-A	1	x	1	0
ADD INVERT-B	1	x	0	1
ADD INVERT-AB	1	x	1	1
AND	0	0	0	0
NOR	0	0	1	1
SET1S	0	0	1	1
AND INVERT-A	0	0	1	0
AND INVERT-B	0	0	0	1
XOR	0	1	0	0
XNOR	0	1	0	1
NOP	0	0	0	0

4.4.2 Simulation Results

All results have been obtained through our simulations of the 8-bit ALU design after post-layout verification with HYPRES.

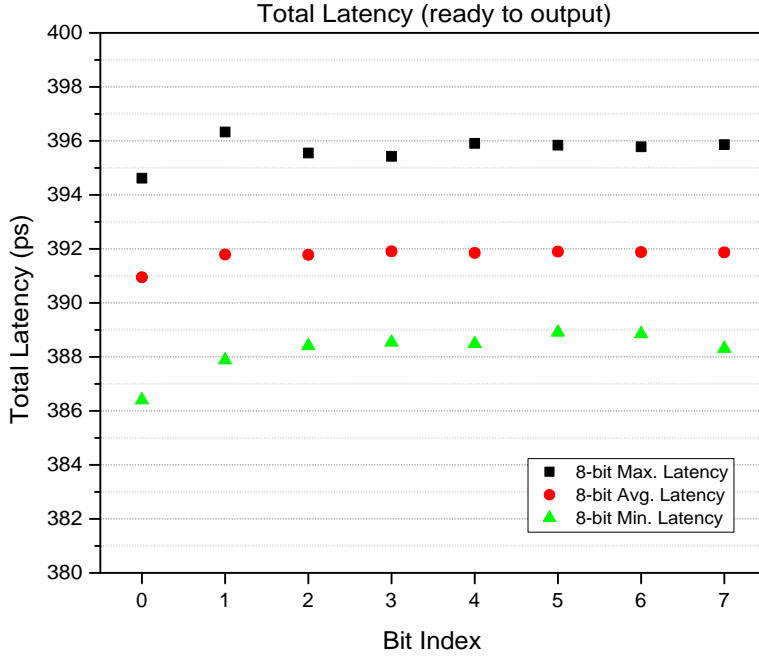


Figure 4.9: Simulated latency results of the 8-bit ALU design after post-layout verification.

Table 4.11: Josephson junction complexity of the 8-bit ALU design.

Cell	Cell Count	JJs/Cell	Total JJs/Cell	%Cells	%JJs
AAND	24	6	144	0.50%	1.97%
CFE	75	16	1200	1.57%	16.40%
CXOR	32	9	288	0.67%	3.93%
D2FF	16	7	112	0.34%	1.53%
DFE	54	4	216	1.13%	2.95%
DFFC	16	10	160	0.34%	2.19%
JL	3651	1	3651	76.67%	49.88%
MRG	33	5	165	0.69%	2.25%
RX	261	3	783	5.48%	10.70%
SPL	339	1	339	7.12%	4.63%
TX	261	1	261	5.48%	3.57%
Total Cells:	4762	Total JJs:	7319	100.00%	100.00%

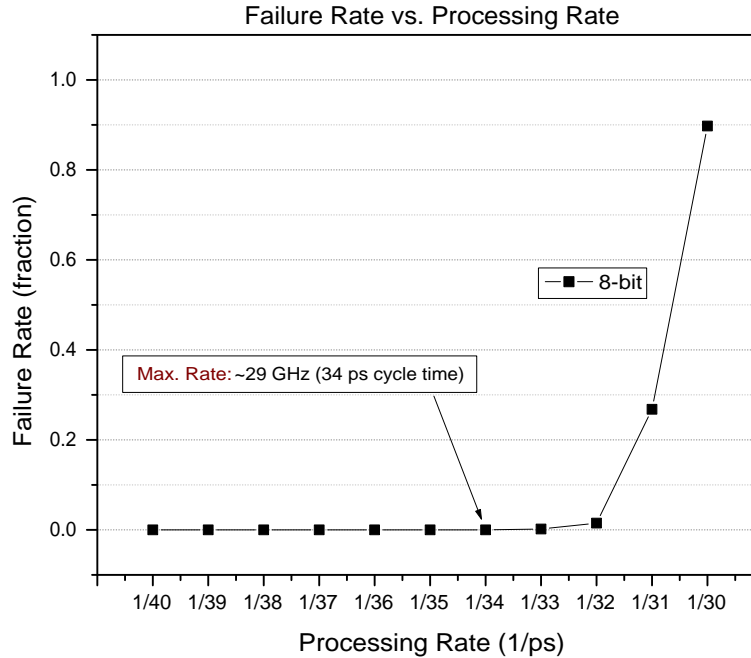


Figure 4.10: Simulated processing rate results of the 8-bit ALU design after post-layout verification.

Table 4.12: Categorical complexity of the 8-bit ALU design.

Category	JJ Count	%JJs
Logic and DFFs	2285	31.22%
Fan-out	339	4.63%
PTL Drivers and Receivers	1044	14.26%
Misc. (Cell separation and delay lines)	3651	49.88%
Total JJs:	7319	100.00%

4.4.3 Low-Frequency Testing

Our colleagues at HYPRES implemented the physical design and testing of two prototype chips of the 8-bit ALU. The first chip consisted of only the 8-bit ALU without any additional circuits for high-speed testing therefore only low-frequency, functional testing was demonstrated for this chip (Figure 4.11 on page 78). This section summarizes the low-frequency results from [92].

Numerous low-frequency functionality tests were performed on the ALU

using the Octopux testing system [112]. The most critical operation to demonstrate is the ADD operation. Figure 4.12 on page 79 shows the ALU correctly adding two 8-bit numbers. A logical '1' appears as a small rectangular pulse whereas a logical '0' shows an absence of this pulse. A dotted blue trace is overlaid to represent a sampled estimation of how the output would actually appear on an oscilloscope (logical '1' represented by a rising or falling edge, a logical '0' represented by no change). Data inputs are first sent into ALU and then a *Ready* pulse is supplied after some delay. The outputs are aligned with their associated *Ready* pulse on the figures.

Further testing included the demonstration of logical operations as shown in Figure 4.13 on page 80. All arithmetic and logical operations show correct functionality at low-frequency.

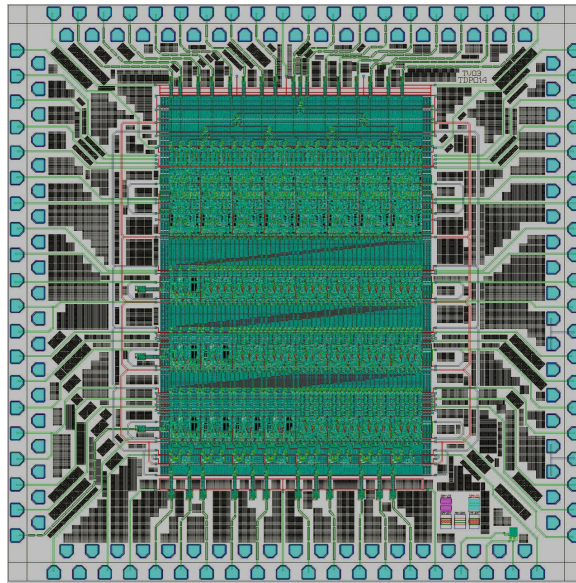
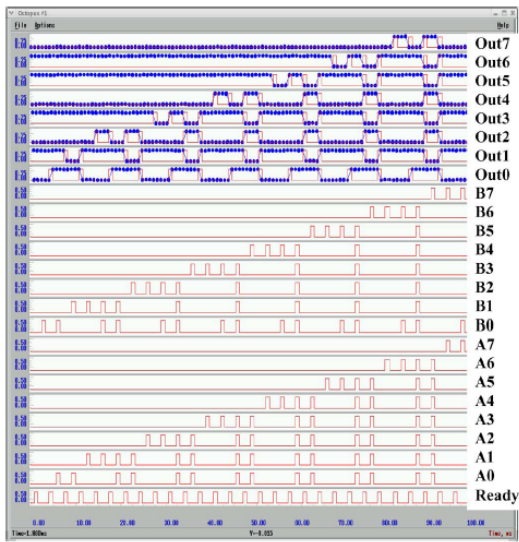
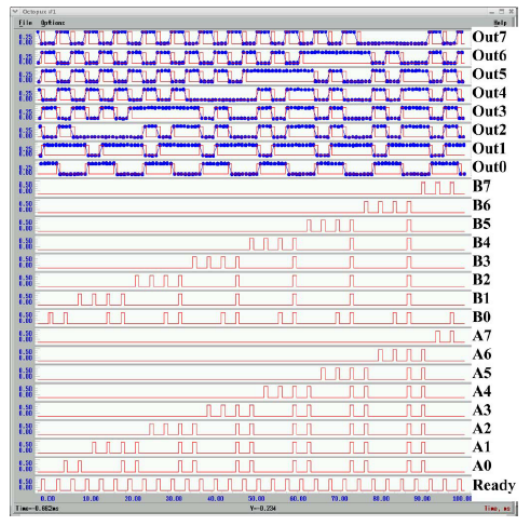


Figure 4.11: Microphotograph of the 8-bit ALU chip for low-frequency testing using the HYPRES $1.5\ \mu\text{m}$ $4.5\ \text{kA}/\text{cm}^2$ technology.

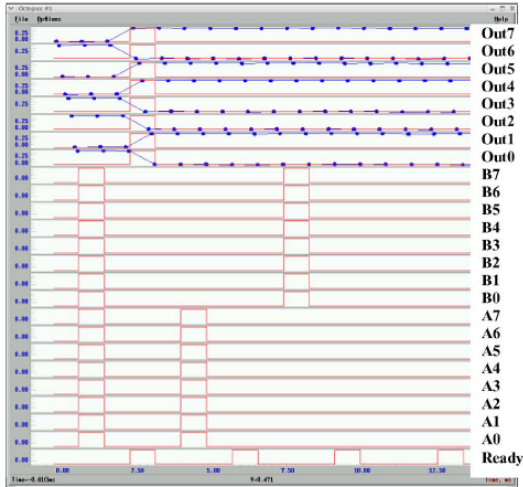


(a) ADD operation.

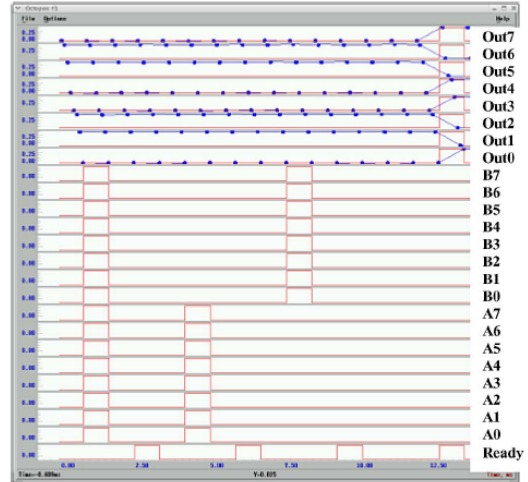


(b) ADD INVERT-AB operation.

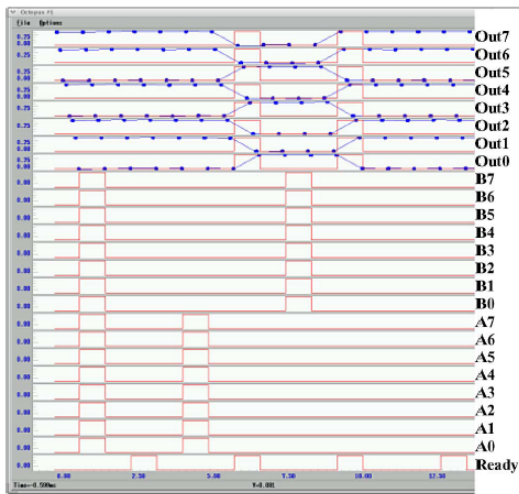
Figure 4.12: ADD operation during functional low-frequency testing [92]. © 2011 IEEE.



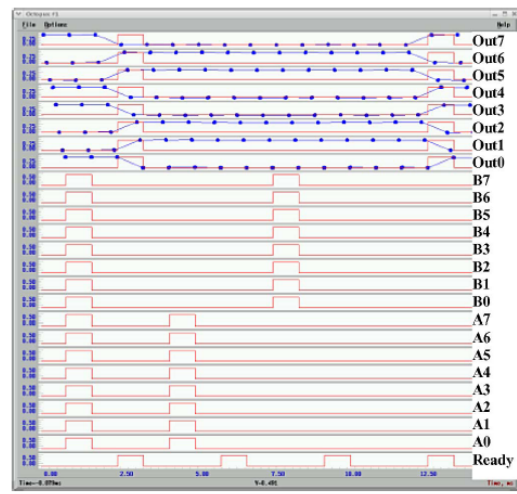
(a) AND.



(b) NOR.



(c) XOR.



(d) XNOR.

Figure 4.13: Low-frequency functional testing of logical operations [92]. © 2011 IEEE.

4.4.4 High-Frequency Testing

Our colleagues at HYPRES have also completed the physical design and testing of a second chip which includes the same 8-bit ALU integrated with additional circuits to conduct high-speed testing (Figure 4.14 on page 82). This section summarizes the high-frequency testing circuits and results from [93].

4.4.4.1 High-Speed Input/Output Interfaces

To provide data operand and control signals at high frequency, a high-speed input interface consisting of SFQ relays was designed. These SFQ relays were controlled by DC or low-frequency bias currents to toggle them on or off. If toggled on, the relay will allow an SFQ pulse to pass through. If toggled off, it will reject any incoming SFQ pulse. Each ALU_INIT bit slice has a total of 6 relays: 2 for data operands A and B, and 4 for control signals. Any set of test vectors and ALU function can be tested by programming the relays. By manipulating the relays at the kHz regime, the changes in outputs can be observed using a low-frequency oscilloscope. A total of 20 pads were needed to control the input interface consisting of 2 x 8-bit inputs and a 4-bit control signal. The high-frequency clock coming from a single pad was distributed to each of the bit slices using a PTL-based splitter tree.

The output interface of the ALU was designed for bit-error rate (BER) measurement at high-frequency. The outputs of the ALU were converted to a dual-rail format so that both direct and complementary signals were available. These signals are transmitted to toggling-type SFQ-to-DC converters [40]. These SFQ-to-DC converters will toggle between 0.0 mV steady-state to 0.5 mV steady-state every time an SFQ pulse is received. When an output is producing a logical '1' at 20 GHz, a low-speed oscilloscope will display that output as the average voltage of the high-speed switching between 0.0 mV and 0.5 mV, resulting in a steady-state 0.25 mV line. When an output is producing a logical '0', the output signal appears as either a 0.0 mV or a 0.5 mV line. To measure BER, a fixed combination of control signals and data vectors were chosen so that the direct and complementary outputs were all stable lines. An error would appear as a sudden transition between the DC voltage states of the SFQ-to-DC converter.

A total of 16 pads were used to provide both direct and complementary 8-bit outputs and an additional two pads were used to monitor the clock and decimated clock. The entire 8-bit ALU and the associated I/O interfaces were integrated onto a 1 x 1 cm² chip (Figure 4.15 on page 83). The ALU without I/O interfaces occupied an area of 4480 μm x 5245 μm .

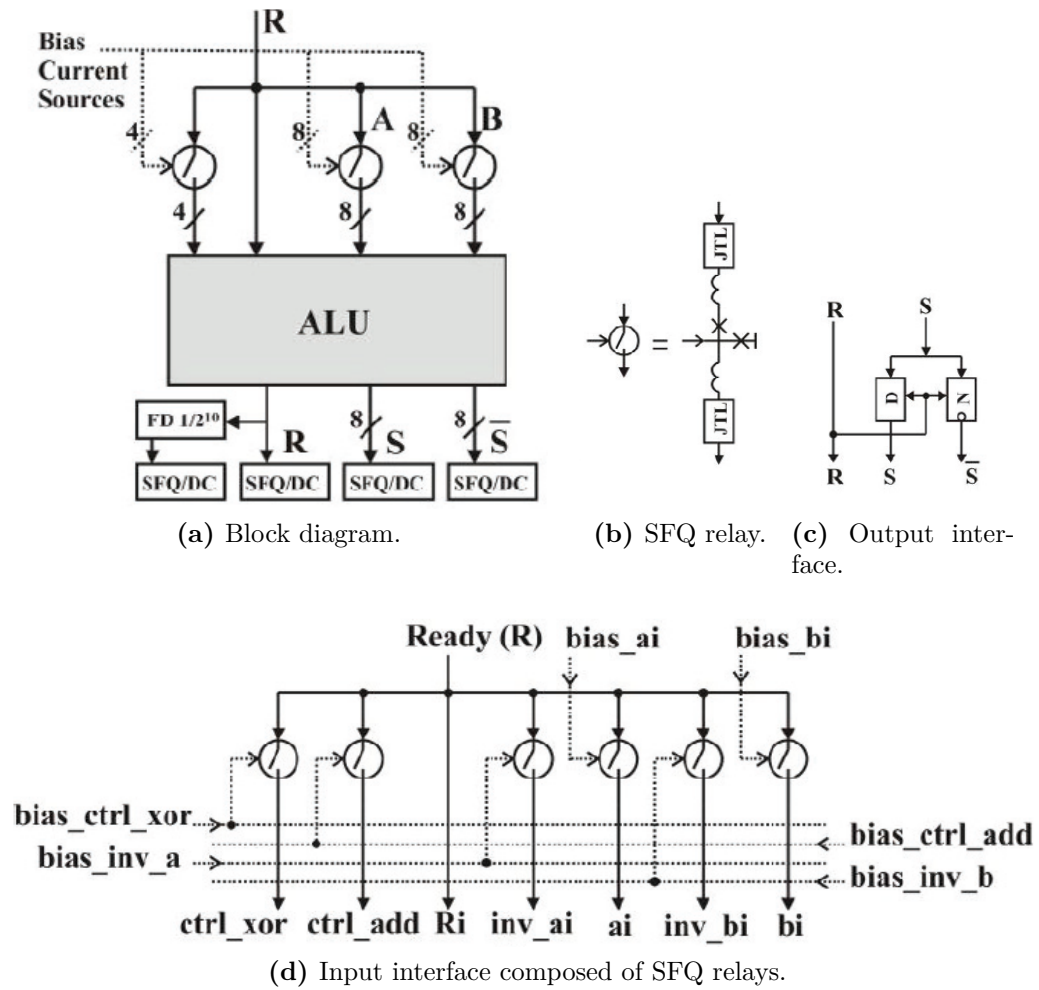


Figure 4.14: Block diagram of the ALU for high-speed testing and the I/O interfaces [93].

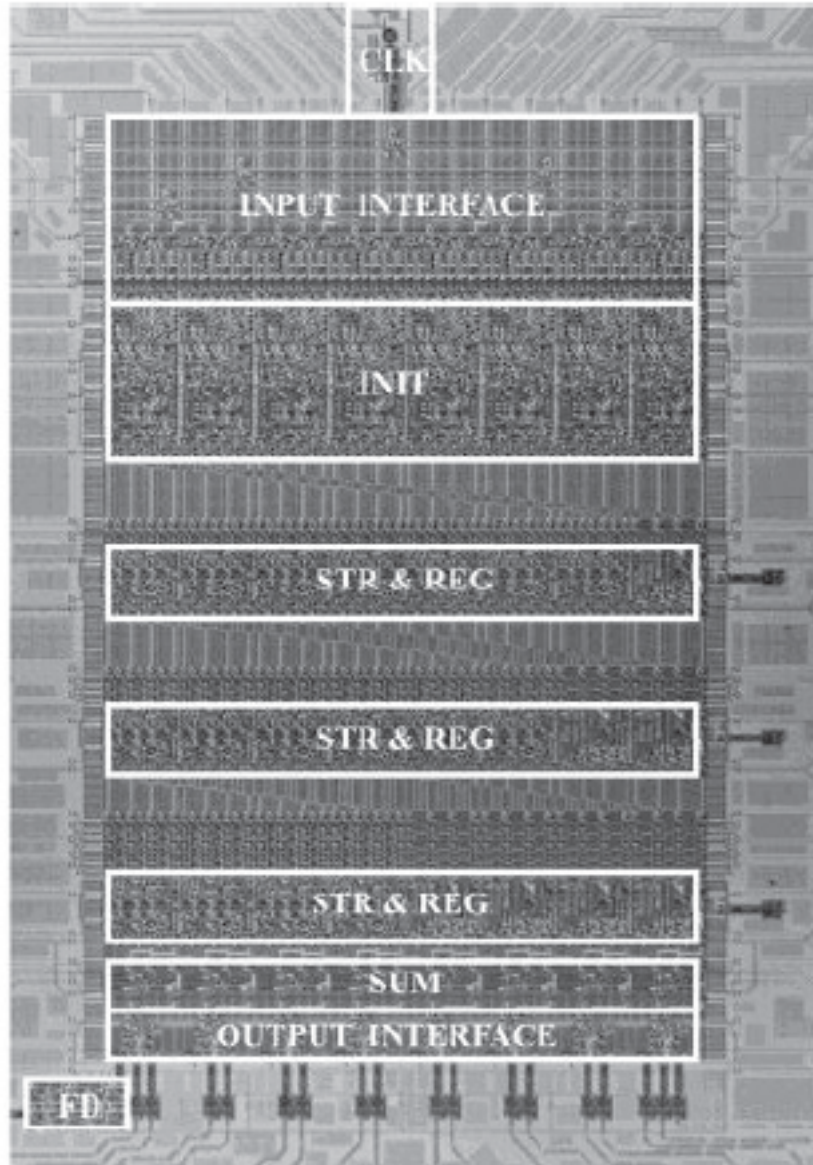


Figure 4.15: 8-bit ALU chip with high-speed testing circuits [93].

4.4.4.2 High-Speed Testing Results

As previously discussed, the oscilloscope waveforms display a steady 0.25 mV line in the case of a logical ‘1’, while a logical ‘0’ displays a randomly selected 0.0 mV or 0.5 mV line depending on the previous state of the SFQ-to-DC converter, rendering an “eye-diagram”.

Figure 4.16 on page 84 shows the demonstration of the logical AND operation functioning correctly when operand A is fixed to 255 and B is modulated between the two values of 255 and 0 at low-frequency. The operation of logical functions solely test the correctness of the ALU block residing in the Initialization stage. Logical results are routed and propagated along the vertical bitwise lines instead of the Prefix Tree thus the Carry-Merge blocks are not exercised for this case.

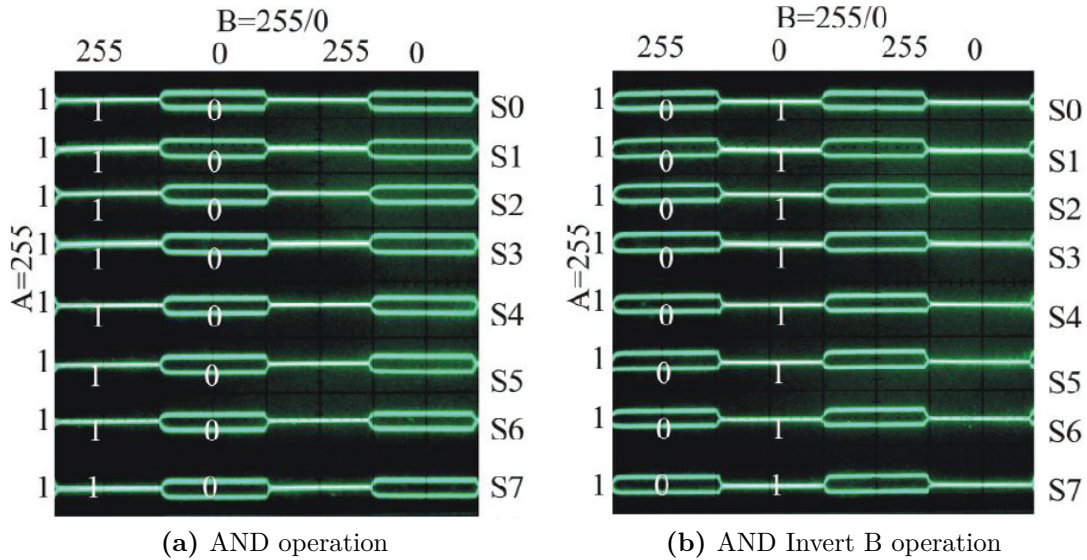


Figure 4.16: Correct logical operations at 20 GHz when A is fixed to 255 and B is toggled at low-speed between 255 and 0 [93].

To demonstrate operation of the Prefix Tree, two critical cases were covered for the ADD operation (Figure 4.17 on page 85). In each case, a carry is generated from the least significant bit and is propagated through all bits except the most significant (Figure 4.17a on page 85) or through all bits completely (Figure 4.17b on page 85). When a carry has to propagate through a bit, it must travel through the horizontal path associated by the stage it is currently at. Having a carry propagate through all bits requires it to travel through what is theoretically the critical path of the Prefix tree. In reality, all propagation paths are matched to the critical path to increase the processing

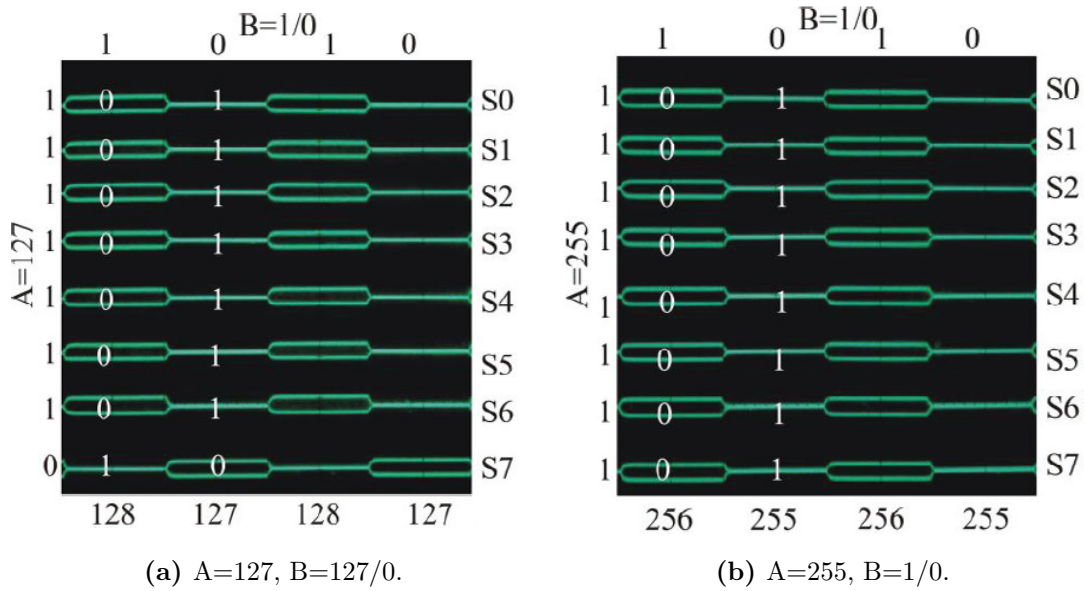


Figure 4.17: 20 GHz operation of two critical cases of the ADD operation where A is a fixed value and B is modulated between 1 and 0. It demonstrates how the carry generated from the least significant bit propagates through most or all bits of the ALU [93].

rate of wave-pipelining. By demonstrating these cases at high-speed, we have shown that wave-pipelining has been correctly implemented.

In the previous waveforms we show the high-frequency operation of the ALU with a fixed function but with changing data values. Figure 4.18 on page 86 shows changing ALU functions but with fixed data values. All cases show correct functionality.

All logic and arithmetic operations have been confirmed to work correctly at the processing rate of 20 GHz. The measured operating margins for the DC bias currents is $\pm 5\%$ and the bit-error rate is estimated to be $\sim 10^{-14}$.

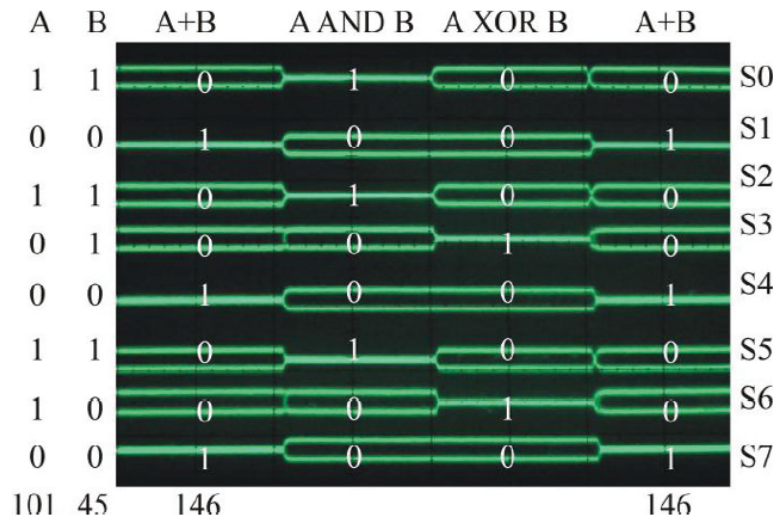


Figure 4.18: 20 GHz operation showing correct functionality of ADD, AND, XOR and ADD for fixed values of A=101 and B=45 [93].

Chapter 5

Superconductor Hybrid Sparse-Tree Adder and ALU

Outline

5.1	Goals and Challenges	87
5.2	Sparse-Tree Structure	88
5.3	RSFQ Study	90
5.3.1	Design Overview	90
5.3.2	Simulation Results	94
5.3.3	Discussion	101
5.4	Adder and ALU Design Implemented Using the CON- NECT Cell Library for the 1.0 μm 10 kA/cm² Process	105
5.4.1	Goals and Challenges	105
5.4.2	Simulation Results	114
5.4.3	Experimental Testing	117
5.4.4	Chip Testing Results	122

5.1 Goals and Challenges

In Chapter 3, we studied the RCA structure as a design candidate for low-complexity at the expense of sacrificing performance. In contrast, we studied the KSALU as a design candidate for high-performance at the expense of high-complexity and power in Chapter 4. In this chapter, our study is now focused on combining the benefits of the RCA and KSALU to develop a 20+ GHz

design but at reduced complexity and power compared to the KSALU. We achieve this by adopting a parallel prefix carry look-ahead hybrid sparse-tree adder (HSTA). All of the design techniques from the KSALU study are still utilized but in the HSTA, a lot of logic has been eliminated. A sparsity-4 arrangement is used to give us a reasonable reduction in complexity without impacting performance significantly. This arrangement also required us to integrate new sub-blocks such as an in-step RCA to pre-calculate the sum for each 4-bit group and a 4-bit carry-skip adder to calculate the final summation. Since the HSTA is also a prefix carry look-ahead structure, it can use the same ALU block from the KSALU to create a hybrid sparse-tree ALU (HSTALU).

Under collaboration with our colleagues in Yokohama National University and Nagoya University, these concepts were then implemented and demonstrated using the SFQ CONNECT cell library for the ISTECH 1.0 μm 10 kA/cm² ADP process in two forms: (1) an 8-bit HSTALU and (2) a 16-bit HSTA.

5.2 Sparse-Tree Structure

The sparse-tree structure is somewhat similar to the Kogge-Stone structure described in Chapter 4. The prefix equations generally remain the same, the only difference is how the carries are merged from stage-to-stage. The Kogge-Stone adder is a sparsity-1 structure, meaning that every carry bit is generated resulting in a simple XOR operation at the final Summation stage. In the sparse-tree adder, it is common to use a sparsity-4 structure [24, 23, 113], which generates a carry for every fourth bit. This sparsity substantially reduces the number of carry merge logic cells in the adder and at the same time also reduces the wiring traffic from stage-to-stage, making it simpler to layout. Figure 5.1 on page 89 shows the typical structure of a 16-bit sparse-tree adder.

The logic cells are the same as the Kogge-Stone cells except the orange blocks are no longer a single two-input XOR but rather a 4-bit carry-select adder. While the sparse-tree is merging carries, two sums are being simultaneously calculated for each 4-bit group: a 4-bit sum assuming there is a carry-in, and a 4-bit sum assuming there is no carry-in. With these two pre-calculated sums arriving at the Summation stage, the true carry-in from the Prefix Tree stage chooses which 4-bit sum to use in each 4-bit group.

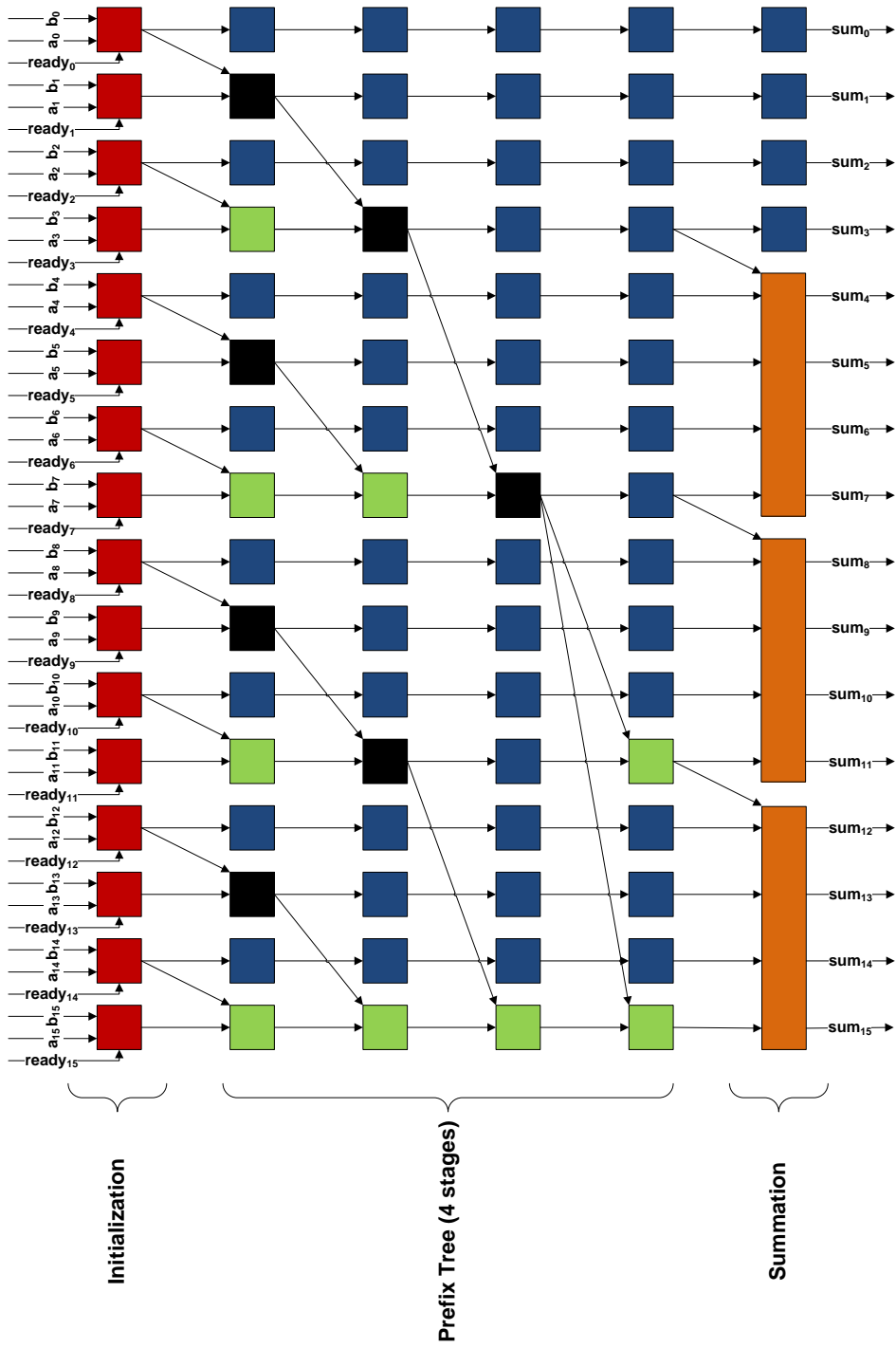


Figure 5.1: The structure of a 16-bit sparse-tree adder.

5.3 RSFQ Study

5.3.1 Design Overview

The implementation of carry select adders in traditional sparse-tree structures works fine in CMOS technology because they are typically designed to perform in a single cycle without the need for pipelining. This approach does not work very well for a high clock rate RSFQ design. It is actually quite expensive to calculate two different 4-bit sums and wave-pipeline them. It requires more vertical stage-to-stage interconnect to transfer these two groups and more JTLs to balance delays across each stage.

Instead, a hybrid sparse-tree adder was developed which involved some modifications with the original sparse-tree adder. In Figure 5.2 on page 92, the bold lines represent prefix carry look-ahead signals, dotted lines represent a ripple carry addition within each 4-bit group, and thin lines represent bitwise signals. Note that the Summation stage is not necessarily the last stage of the adder as lower order bits can be calculated ahead of time and then buffered through the remaining stages. This implementation also reduces the worst-case horizontal PTL length by relying on the Summation block to calculate the final carry-out.

During the first four stages of the adder, we are performing 4-bit ripple-carry addition (1-bit per stage) as shown with logic cells colored with a gray-half. This pre-calculation is akin to calculating the 4-bit sum assuming there is no carry into the 4-bit group and is done by integrating a CXOR gate in the gray-half cells (Figure 5.3a and Figure 5.3b on page 93). These CXORs will calculate the sum for a particular bit using only the bitwise prefix P signal and the carries generated within the group it resides in. The final summation is done as soon as a particular 4-bit group carry-in is calculated (e.g. bits 7 through 4 are calculated one stage earlier because the carry into this group is available one stage earlier) and is completed using carry-skip adders. In each 4-bit group, the second least significant bit produces a group signal called PP. The pipelining of this signal is shown as a Blue Cell with two inputs and outputs. When this PP signal is a logical '1', it means the carry into this 4-bit group will propagate through the lower-order 2-bit half and will be used as a carry into the higher-order 2-bit half. This speeds up calculation of the 4-bit sum as we do not need to wait for carry to truly propagate through the lower-order half, since the ANDing of PP and the carry-in will immediately determine the carry that goes into the higher-order half.

The vertical PTLs have been reduced compared to the KSALU because of less interconnect traffic and channels needed between the stages. The new PTL lengths are shown in Table 5.1 on page 91.

Table 5.1: Listing of PTL interconnect lengths for the 32-bit HSTALU.

Stage-to-Stage	Horizontal PTL Length (mm)	Vertical PTL Length (mm)	Total PTL Length (mm)
0 to 1	0.280	0.060	0.340
1 to 2	0.560	0.060	0.620
2 to 3	1.120	0.060	1.180
3 to 4	2.240	0.060	2.300
4 to 5	3.360	0.060	3.420
5 to 6	0.000	0.280	0.280
Total Length (mm)	7.560	0.580	8.140
Total Delay (ps)	75.60	5.80	81.40

The same ALU implementation discussed in Chapter 4 is re-used in this modified hybrid sparse-tree core.

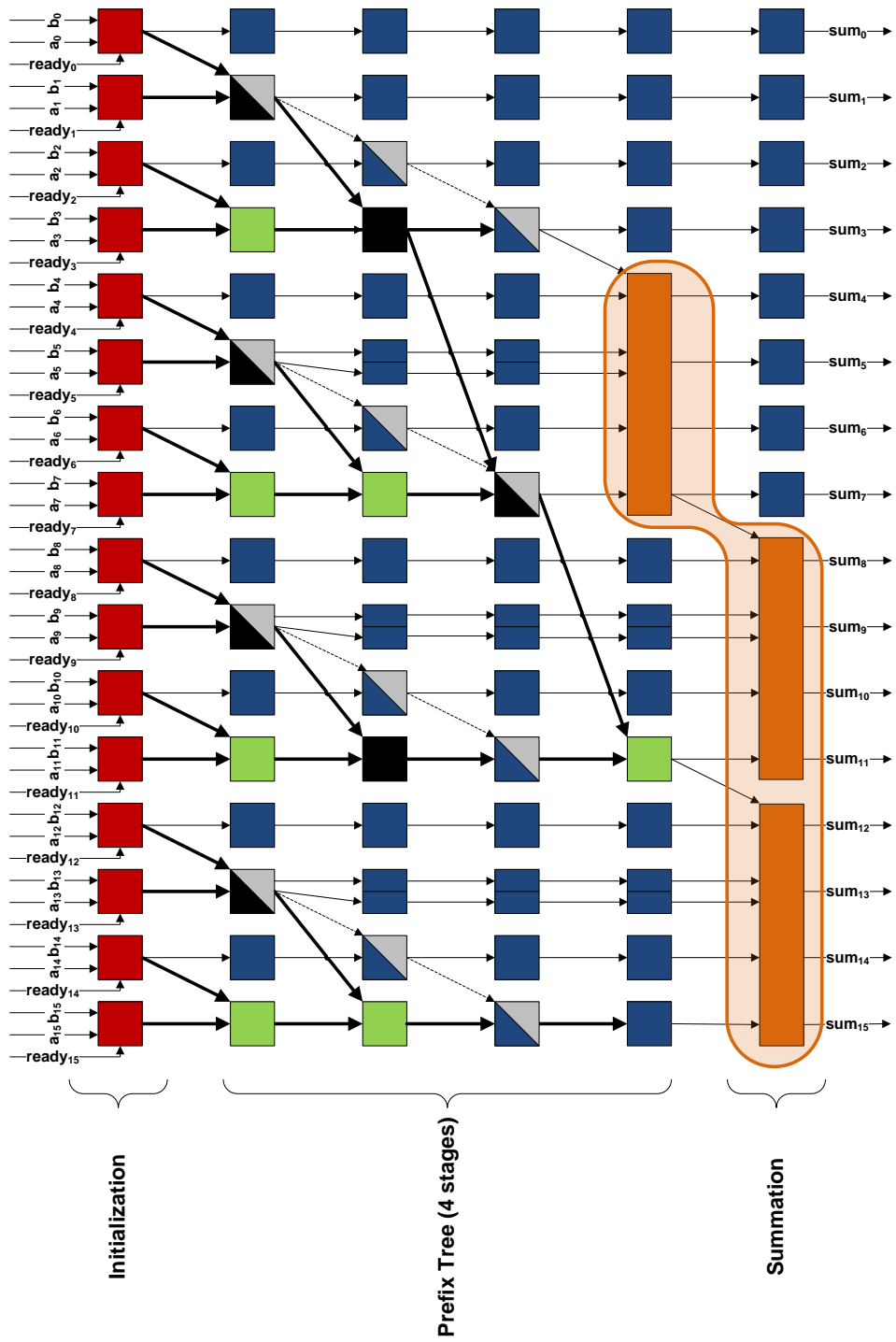


Figure 5.2: The structure of the RSFQ 16-bit hybrid sparse-tree adder.

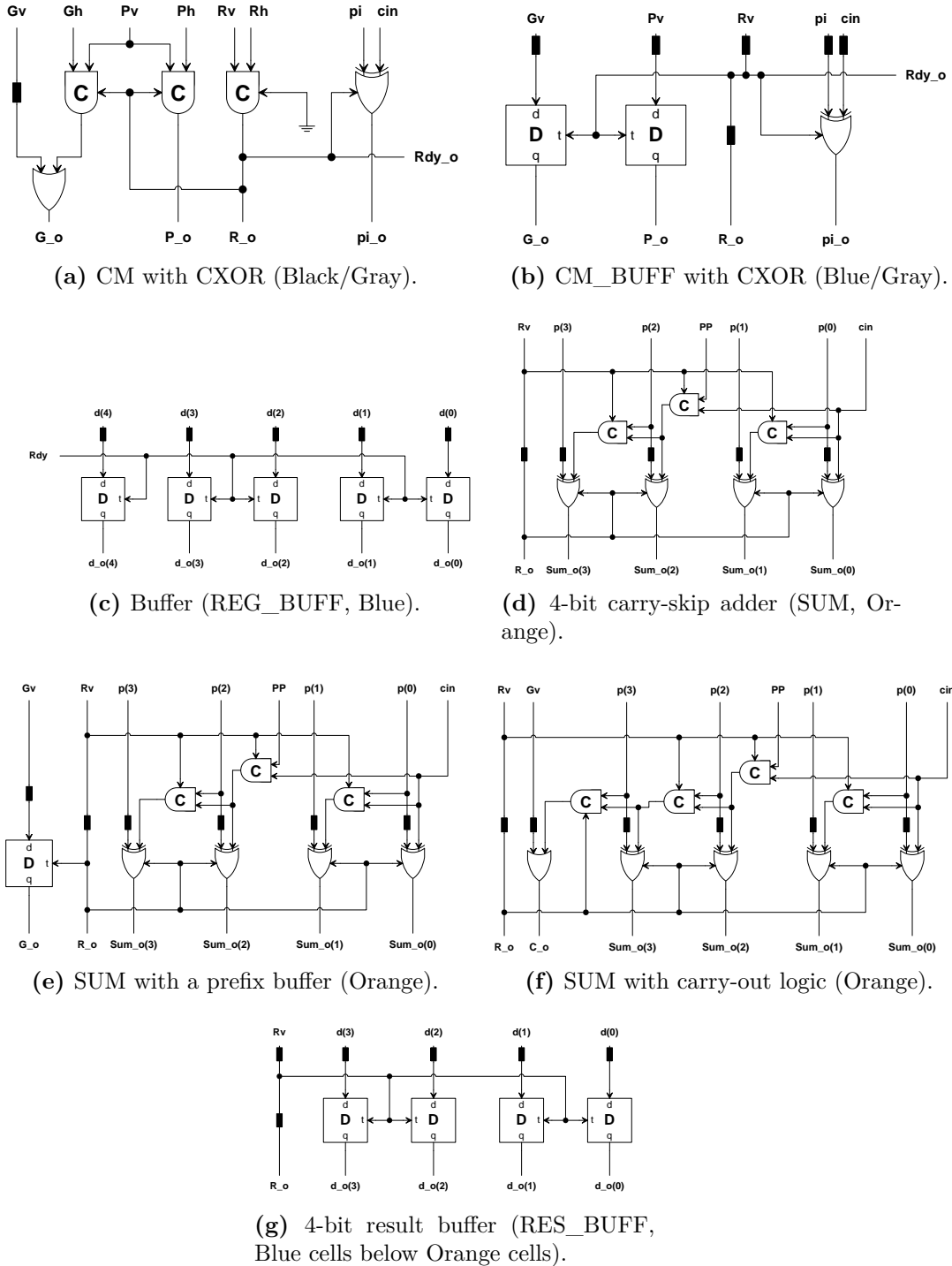


Figure 5.3: Logic schematic blocks that are unique to the sparse-tree structure.

5.3.2 Simulation Results

Using our VHDL cell library tuned to the HYPRES 1.5 μm 4.5 kA/cm² process, we designed and simulated a 32-bit HSTALU. The results of the simulation are summarized in Table 5.5 on page 100. A detailed discussion and comparison of the HSTALU, KSALU and RCA is examined in Section 5.3.3.

Table 5.2: Processing rate of the 32-bit HSTALU. At each processing rate, an initial set of 2000 randomized vectors mixed with worst-case tests is sent into the HSTALU to obtain a first-pass sweep of the processing rate. At the maximum passing rate, a second set of 10,000 vectors is sent to exercise the circuit further.

Processing Rate (GHz)	Number Failed Waves	Total Number of Waves
20.0	0	2000
20.4	0	2000
20.8	0	2000
21.3	0	2000
21.7	0	2000
22.2	0	2000
22.7	0	10000
23.3	3	2000
23.8	15	2000
24.4	713	2000
25.0	1999	2000
25.6	2000	2000
26.3	2000	2000

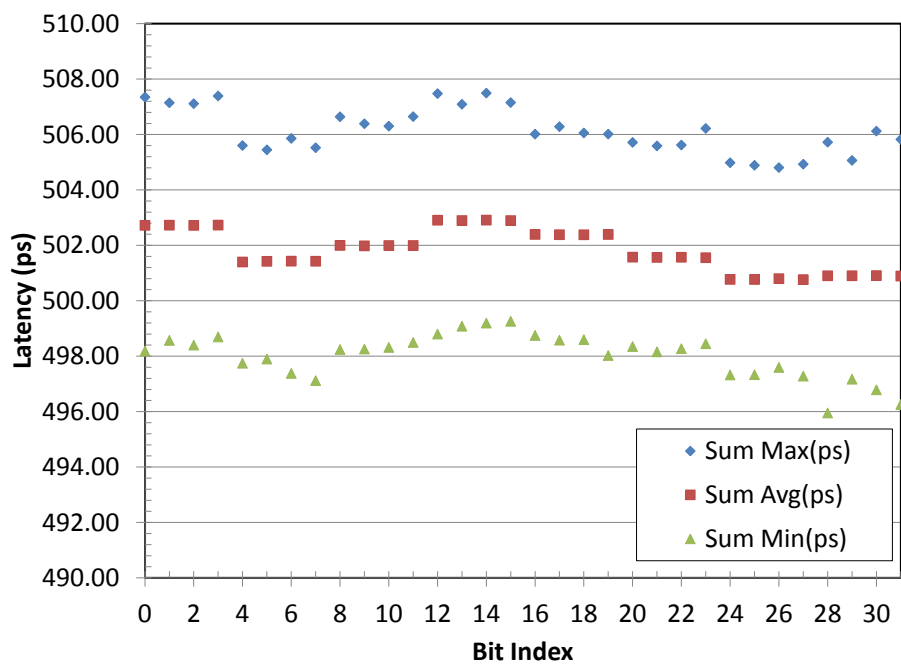
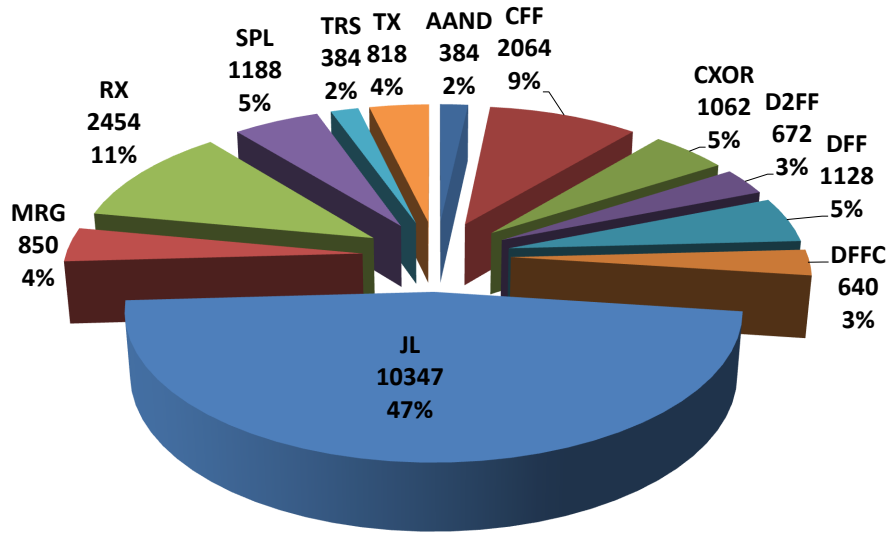


Figure 5.4: Latency distribution of the 32-bit HSTALU. The least significant bit is bit index 0. The latency is measured from the assertion of the “ready” signal to the arrival of outputs at the “sum” port.

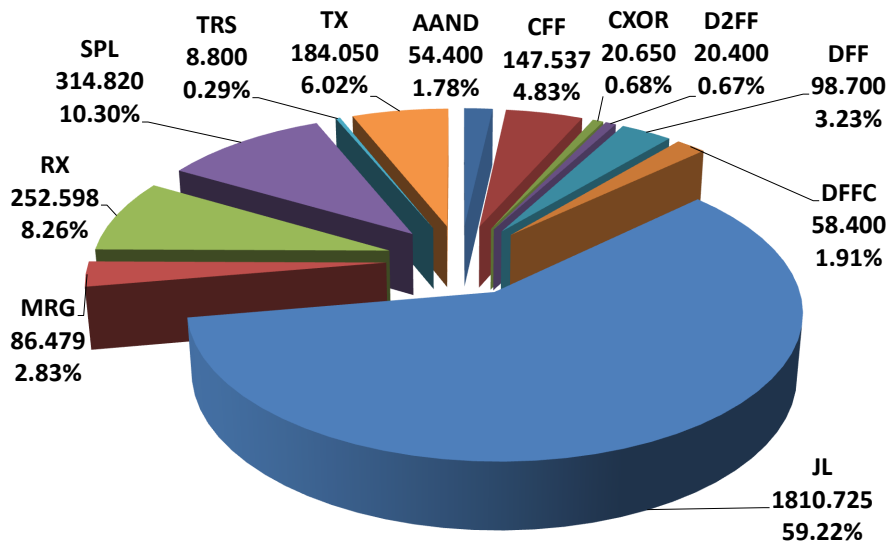
Table 5.3: Latency distribution of the 32-bit HSTALU with the average latencies calculated across all bits.

Bit	Sum Max. (ps)	Sum Avg. (ps)	Sum Min. (ps)
31	505.82	500.89	496.26
30	506.12	500.91	496.79
29	505.07	500.90	497.17
28	505.72	500.90	495.95
27	504.93	500.76	497.28
26	504.81	500.80	497.60
25	504.89	500.77	497.33
24	504.98	500.77	497.32
23	506.22	501.55	498.45
22	505.62	501.57	498.27
21	505.59	501.57	498.16
20	505.72	501.57	498.35
19	506.02	502.39	498.03
18	506.06	502.38	498.60
17	506.28	502.38	498.58
16	506.01	502.39	498.75
15	507.15	502.89	499.26
14	507.50	502.91	499.20
13	507.09	502.90	499.08
12	507.48	502.91	498.80
11	506.65	501.99	498.50
10	506.30	501.99	498.32
9	506.39	501.98	498.26
8	506.64	502.00	498.24
7	505.52	501.43	497.12
6	505.86	501.43	497.38
5	505.45	501.42	497.90
4	505.60	501.40	497.75
3	507.39	502.73	498.70
2	507.11	502.72	498.40
1	507.15	502.73	498.57
0	507.35	502.72	498.19
Average	506.14	501.83	498.02



Total Design Complexity: 21991 JJs

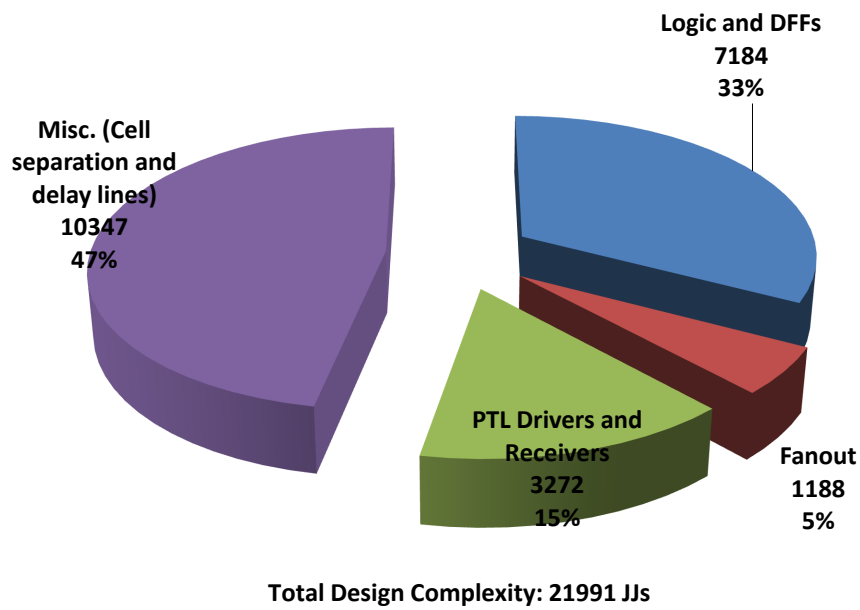
(a) JJ design complexity breakdown of the 32-bit HSTALU.



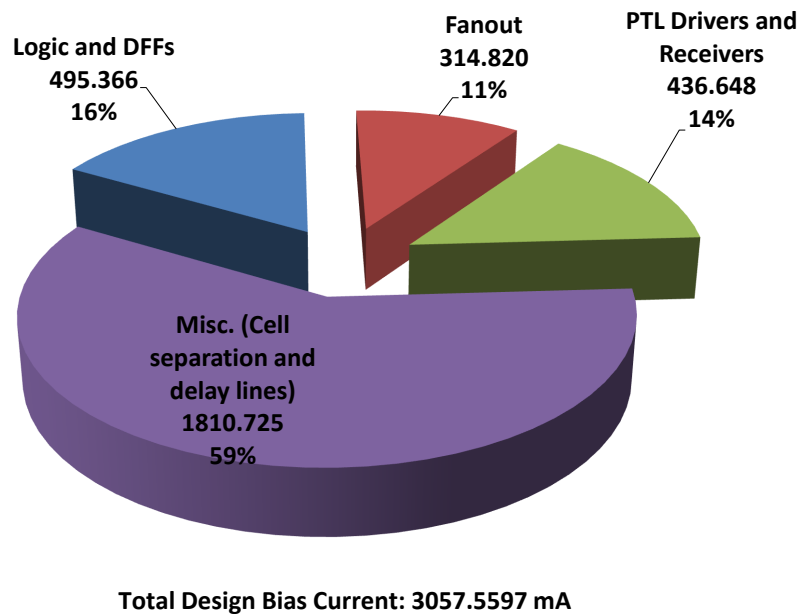
Total Design Bias Current: 3057.5597 mA

(b) Design bias current breakdown of the 32-bit HSTALU.

Figure 5.5: Cell-wise breakdown of the 32-bit HSTALU for both the complexity and bias current of the logical design. The results do not include additional JJs to distribute bias current in ERSFQ logic.



(a) Categorical JJ design complexity breakdown of the 32-bit HSTALU.



(b) Categorical design bias current breakdown of the 32-bit HSTALU.

Figure 5.6: Categorical breakdown of the 32-bit HSTALU for both the complexity and bias current of the logical design.

Table 5.4: Breakdown of bias JJs for the 32-bit HSTALU for ERSFQ logic. The raw bias JJ count assumes all cells are using non-sharing bias JJs. The adjusted (adj.) bias JJ count assumes that 20% of all Josephson transmission lines (JL cells) are connected to non-sharing bias JJs and the remaining 80% of JL cells share 1 bias JJ for every 2 JL cells.

Cell	Cell Count	Bias JJs/Cell	Total Raw Bias JJs/Cell	Total Adj. Bias JJs/Cell	% Raw Bias JJs/Cell	% Adj. Bias JJs/Cell
AAND	64	1	64	64	0.40%	0.53%
CFF	129	6	774	774	4.80%	6.46%
CXOR	118	2	236	236	1.46%	1.97%
D2FF	96	1	96	96	0.60%	0.80%
DFE	282	2	564	564	3.50%	4.71%
DFFC	64	3	192	192	1.19%	1.60%
JL	10347	20%=1, 80%=1/2	10347	6208	64.20%	51.83%
MRG	170	1	170	170	1.05%	1.42%
RX	818	2	1636	1636	10.15%	13.66%
SPL	1188	1	1188	1188	7.37%	9.92%
TRS	32	1	32	32	0.20%	0.27%
TX	818	1	818	818	5.08%	6.83%
Total:	14126		16117	11978	100.00%	100.00%

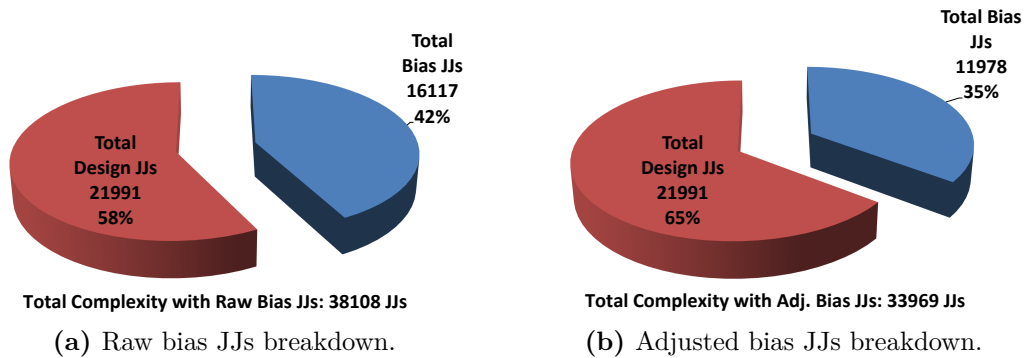


Figure 5.7: Breakdown comparison of both raw and adjusted bias JJ counts with respect to the total design complexity of the 32-bit HSTALU implemented in ERSFQ logic.

Table 5.5: Summary of the key simulation results for the 32-bit HSTALU.

(a) Overall summary of the 32-bit HSTALU.

Data Width	32-bit
Bit Pitch	0.280 mm
Design Complexity	21,991 JJs
Overall Avg. Latency	502 ps
Max. Processing Rate	22.7 GHz
Total Bias Current	3.058 A

(b) Power related metrics for both RSFQ and ERSFQ implementations of the 32-bit HSTALU at 4.2 K temperature.

Logic	RSFQ	ERSFQ
Bias Voltage	2.6 mV	45.4 μ V
Total Power	8.033 mW	141.508 μ W
Ops./Watt	2.826 TOPS/W	160.415 TOPS/W
Energy/Op.	0.354 pJ/op.	6.234 fJ/op.

5.3.3 Discussion

We have conducted design studies for an RCA, KSALU and HSTALU designed for a data width of 32-bits using RSFQ/ERSFQ logic. Table 5.6 on page 102 summarizes the key results of the three design studies conducted in this research. The extremely simple design of the RCA gives it a tremendous advantage in energy efficiency resulting in 2418 TOPS/W, roughly 15–23 times better than the HSTALU and KSALU respectively. However, its processing rate of only 2 GHz is not sufficient for high-performance computing.

The KSALU and HSTALU both have 20+ GHz processing rates due to the use of a high-speed prefix carry look-ahead structure. However, the HSTALU takes on a hybrid approach of combining RCA-like structures within a sparse-tree design resulting in a ~55% improvement in energy efficiency over the KSALU, while only seeing a ~2.6% decrease in processing rate and a ~6.3% increase in latency. In terms of latency, a breakdown of where the KSALU gained advantages is shown in Figure 5.8 on page 103. Table 5.7 on page 104 shows a more detailed analysis of the latency breakdown between these two designs with the corresponding comments below:

1. KSALU gained an additional 2 ps delay because its vertical length is 0.220 mm versus the 0.060 mm in the HSTALU.
2. KSALU gained an additional 2 ps delay because its vertical length is 0.190 mm versus the 0.060 mm in the HSTALU.
3. KSALU lost 2 ps of delay because it lacks the additional splitter that HSTALU needs.
4. KSALU has an additional horizontal PTL length of 1.12 mm (11.2 ps delay) because HSTALU's horizontal interconnect does not need to cross an additional 4-bits. However, HSTALU still requires an additional splitter resulting in a net ~9 ps increase in delay for the KSALU.
5. KSALU does not need the extra 7 ps delay to improve timing and processing rate of the more complex Summation blocks of the HSTALU.
6. KSALU has a much simpler Summation block compared to the 4-bit carry-skip block of the HSTALU resulting in a net loss of 36 ps in delay for the KSALU.

In Figure 5.9 on page 103, we compare the KSALU and HSTALU to a high-performance 9 GHz 65 nm CMOS ALU developed by Intel [114]. Because Intel's ALU contains two 32-bit cores and other peripheral circuits, only 20% of the reported power is compared. Furthermore, the power consumption of the KSALU and HSTALU are multiplied by a factor of 1000 to take into account

Table 5.6: Summary of the 3 design studies.

Units	RCA	KSALU	HSTALU
Data Width	32	32	32
Bit Pitch (mm)	0.12	0.28	0.28
Design Complexity (JJs)	952	36073	21991
Overall Avg. Latency (ps)	520	470	502
Max. Processing Rate (GHz)	2.0	23.3	22.7
Total Bias Current (A)	0.109	4.767	3.058
ERSFQ Power at 4.2 K (μ W)	0.827	224.895	141.508
ERSFQ Energy Efficiency at 4.2 K (TOPS/W)	2418.380	103.604	160.415

a cryostat efficiency of 1000 W/W to cool the circuits to 4.2 K. Taking all this into consideration, the KSALU and HSTALU are both \sim 2.5 times faster than Intel’s ALU while still consuming 9 to 14.5 times less power respectively (Table 5.8 on page 105).

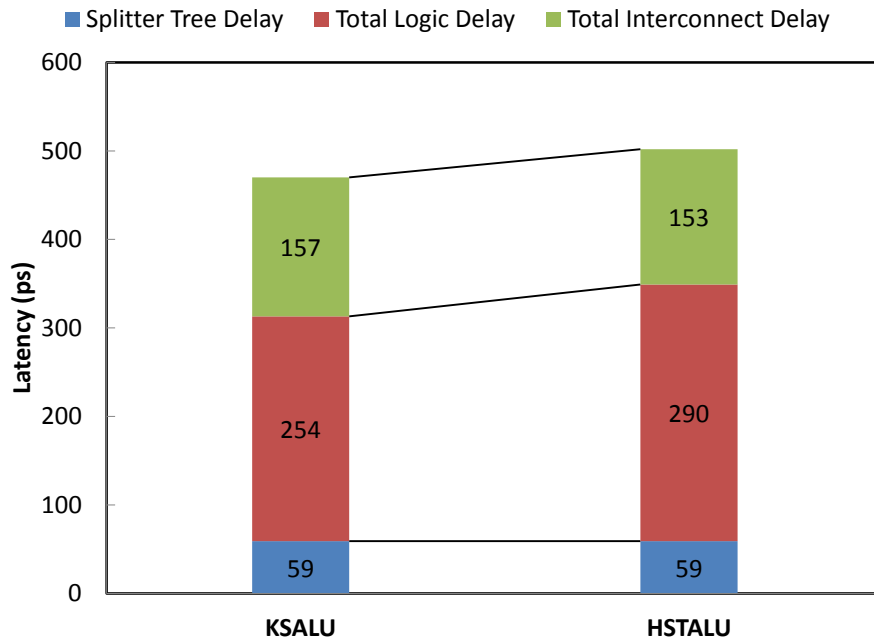


Figure 5.8: Latency breakdown of the KSALU and HSTALU.

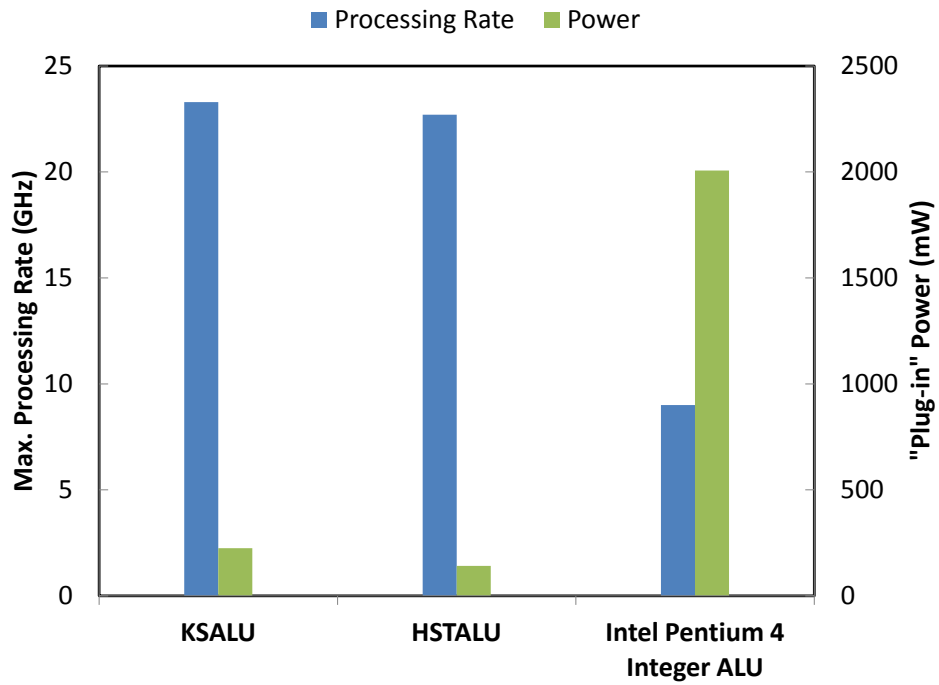


Figure 5.9: Processing rate and power consumption comparison.

Table 5.7: Stage-by-stage latency breakdown analysis of the KSALU and HSTALU.

Stage	Substage	KSALU Delay (ps)	HSTALU Delay (ps)	% KSALU Delay	% HSTALU Delay	Diff (KSALU - HSTALU, ps)	Comment
N/A	Splitter Tree	59	59	12.55%	11.75%	0	
0 - Initialization	Logic	115	115	24.47%	22.91%	0	
	Interconnect	16	14	3.40%	2.79%	2	1
1 - Prefix	Logic	23	23	4.89%	4.58%	0	
	Interconnect	18	16	3.83%	3.19%	2	2
2 - Prefix	Logic	23	23	4.89%	4.58%	0	
	Interconnect	23	23	4.89%	4.58%	0	
3 - Prefix	Logic	23	23	4.89%	4.58%	0	
	Interconnect	34	36	7.23%	7.17%	-2	3
4 - Prefix	Logic	23	23	4.89%	4.58%	0	
	Interconnect	57	48	12.13%	9.56%	9	
5 - Prefix	Logic	23	23	4.89%	4.58%	0	
	Interconnect	9	16	1.91%	3.19%	-7	4
6 - Summation	Logic	24	60	5.11%	11.95%	-36	5
Total		470	502	100.00%	100.00%	-32	

Table 5.8: Comparison of power and rate of the KSALU, HSTALU and an Intel Pentium 4 ALU [114]. For the KSALU and HSTALU, a cryostat efficiency of 1000 W/W is assumed. The total power consumption of the Intel Pentium 4 ALU is actually 10.36 W but it contains 2 x 32-bit cores and additional circuitry so 20% of the total power is compared.

Unit	KSALU	HSTALU	Intel Pentium 4 Integer ALU [114]
Technology	ERSFQ	ERSFQ	65 nm CMOS
Data Width	32-bit	32-bit	32-bit
Clock Rate	23.3 GHz	22.7 GHz	9 GHz
Power	225 mW	142 mW	2.07 W

5.4 Adder and ALU Design Implemented Using the CONNECT Cell Library for the 1.0 μm 10 kA/cm² Process

5.4.1 Goals and Challenges

Using the CONNECT cell library developed by our colleagues at Yokohama National University and Nagoya University, we designed the following units:

1. 16-bit HSTA
 - (a) Chip for functional, low-frequency testing (August 2011)
 - (b) Chip for high-frequency testing (Eagle chip, August 2012)
2. 8-bit HSTALU for functional, low-frequency testing (Delta chip, March 2012)

Both designs were implemented for the ISTEK 1.0 μm 10 kA/cm² ADP process. The goal is to achieve a 30 GHz processing rate for these units and to gain first hand experience in the full flow of taking a design concept from implementation all the way to tape-out and testing. With respect to the 16-bit HSTA, we wanted to achieve a new milestone for SFQ circuits by being the first to demonstrate the largest datapath width for parallel adders in this technology. A total of 5 tape-outs were completed for this research but for this dissertation we will only focus on the 3 chip designs mentioned above.

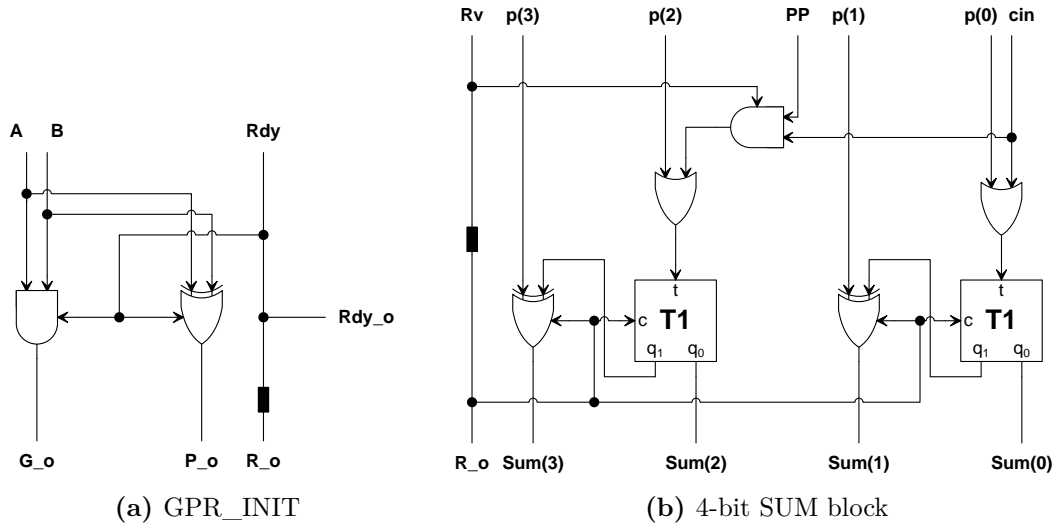


Figure 5.10: HSTA/HSTALU sub-blocks re-designed using the CONNECT cell library.

The microarchitecture of the 16-bit HSTA is exactly as described in Section 5.3 with three major differences. First, the GPR_INIT block (Figure 5.10a on page 106) that creates the bitwise prefix signals at the Initialization stage is simplified slightly as the CONNECT cell library has a convenient CAND (clocked AND) gate. This eliminates the need to generate the prefix G signal using a CFF gate connected to a DFF as it was done in the design study as shown in Figure 4.2b on page 62.

Second, the SUM blocks residing in the Summation stage had to be re-designed. The original 4-bit carry-skip block (Figure 5.3d on page 93) was too slow for a 30+ GHz design and the large amount of logic, particularly the CFFs, made the layout too difficult. Instead, we opted to use a hybrid design of CXORs and T1s (Figure 5.10b on page 106) in a similar arrangement used for the RCA design study in Chapter 3. In the CONNECT cell library, the T1 layout is optimized to connect its carry output to an adjacent CXOR gate resulting in a compact structure. It eliminates the use of multiple CFF gates and required only a single CAND gate to facilitate the carry-skip function. To improve the processing rate of this block, we split the calculation of the 4-bit sum into two stages. The first stage calculates the sum of the lower two bits while preparing the carry going into the upper two bits. The second stage completes the calculation of the upper two bits.

Lastly, the 16-bit HSTA uses a bit pitch of $150\ \mu\text{m}$ since it was designed to be a standalone adder, allowing its pitch to be kept to a minimum without having to match any other units' bit pitch.

To perform high-frequency testing of the 16-bit HSTA, a second chip was developed to include the following 3 supplemental circuits:

1. Clock Generator: To run the adder at high speed, an on-chip clock source was designed. It is a straightforward 16 pulse-train design where a single SFQ pulse is sequentially split 16 times and are all merged together through a sequential chain of MRG gates as shown in Figure 5.12a on page 110. In other words, a single SFQ input pulse will provide a high speed train consisting of 16 SFQ pulses at the output (16 cycles). Each pulse is separated by the same designed delay to obtain the desired clock frequency. The clock generator has 3 modes:
 - (a) Low frequency 1-input pulse, 1-output pulse mode which is especially useful for checking results one at a time on the oscilloscope.
 - (b) 15 GHz to 25.5 GHz mode (Clk<1>) depending on the supplied bias voltage (Figure 5.11 on page 109).
 - (c) 23.5 GHz to 41 GHz mode (Clk<2>) depending on the supplied bias voltage (Figure 5.11 on page 109).
2. High-Speed Input Shift Register: To supply changing input operands at high-frequency, a fast input shifter register was designed for inputs A and B (Figure 5.12b on page 110). It is a 16-bit parallel-load/parallel-output shift register which on each cycle will supply two 16-bit parallel inputs into the adder using the present contents of the register while shifting the contents to the right by 1-bit. The shift register will provide a pair of 16 different 16-bit inputs before it needs to be initialized again with new data.
3. Output Compressor: In order to capture the high-speed waves of the 16-bit outputs, we designed an output compressor which creates an XOR signature of the output (Figure 5.12c on page 110). It is simply a T1 gate for each bit. If an odd number of pulses arrived at a given bit then its corresponding T1 gate will have a logical '1' state, otherwise it will have a logical '0' state. Based on the initial inputs of the input shift register, we can determine the expected XOR signature of the output and compare it with the XOR signature read out from the chip. This simplified the design and testing substantially because we did not need to design a set of output registers to capture each of the 16 waves.

The 8-bit HSTALU also adopted the new SUM sub-block design but the remainder of the microarchitecture is largely the same as it was developed in Section 5.3. The primary difference is that it uses a bit pitch of 300 μm to

Table 5.9: Clock generator high-frequency characteristics obtained from numerical simulation.

Margin	Bias Voltage (mV)	Clk<2> Cycle Time (ps)	Clk<1> Cycle Time (ps)	Clk<2> (GHz)	Clk<1> (GHz)
20.00%	3.00	24.2	39.2	41.32	25.51
18.00%	2.95	24.7	40.0	40.49	25.00
16.00%	2.90	25.0	40.7	40.00	24.57
14.00%	2.85	25.9	41.7	38.61	23.98
12.00%	2.80	26.7	42.9	37.45	23.31
10.00%	2.75	27.5	44.1	36.36	22.68
8.00%	2.70	28.3	45.3	35.34	22.08
6.00%	2.65	29.0	46.4	34.48	21.55
4.00%	2.60	29.7	47.5	33.67	21.05
2.00%	2.55	30.5	48.5	32.79	20.62
0.00%	2.50	31.1	49.5	32.15	20.20
-2.00%	2.45	32.2	51.1	31.06	19.57
-4.00%	2.40	33.2	52.7	30.12	18.98
-6.00%	2.35	34.1	54.2	29.33	18.45
-8.00%	2.30	35.1	55.5	28.49	18.02
-10.00%	2.25	36.1	56.9	27.70	17.57
-12.00%	2.20	37.5	58.9	26.67	16.98
-14.00%	2.15	38.9	60.9	25.71	16.42
-16.00%	2.10	40.1	62.8	24.94	15.92
-18.00%	2.05	41.3	64.6	24.21	15.48
-20.00%	2.00	42.6	66.4	23.47	15.06

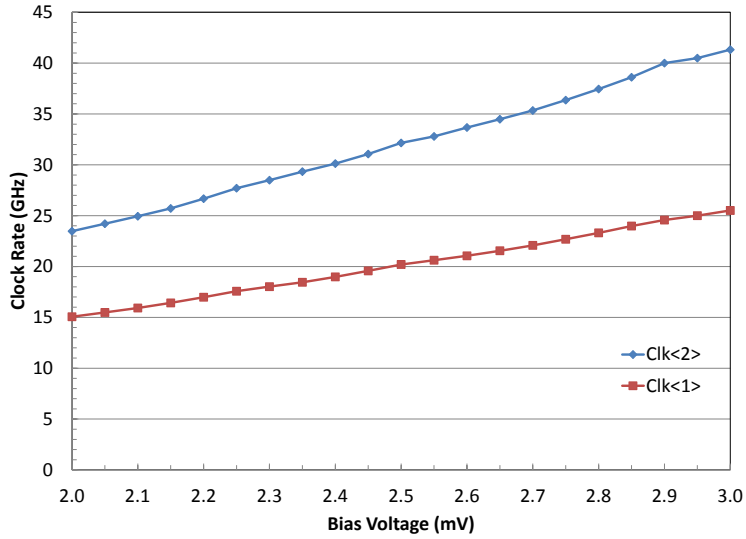
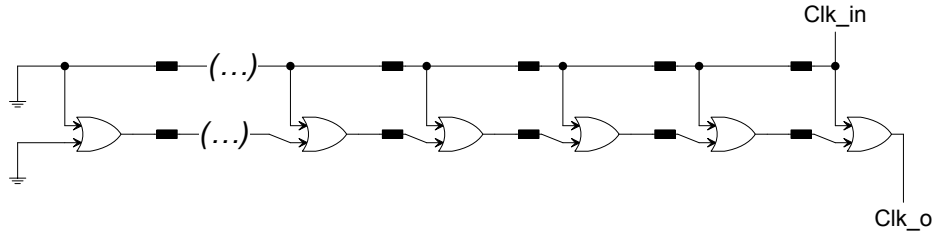
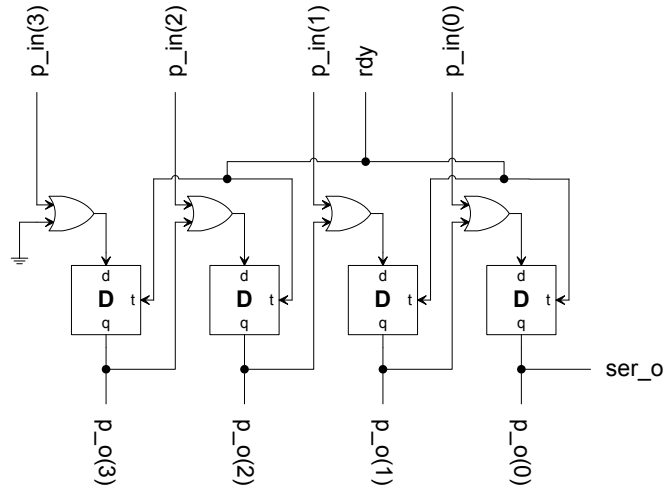


Figure 5.11: Simulation of the clock generator at different bias voltages.

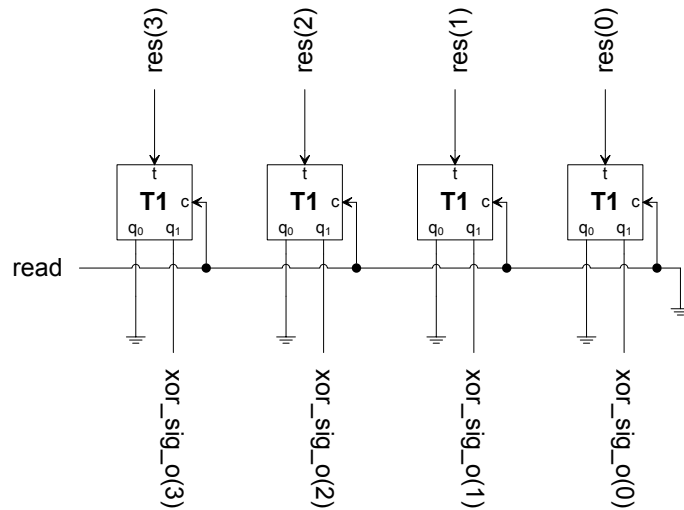
accommodate the logical complexity of the ALU_INIT blocks in the Initialization stage. The larger bit pitch also allowed us to reserve a channel for a return line to propagate the result into a register file unit if the HSTALU was to be integrated into a datapath.



(a) Concept of the clock generator.

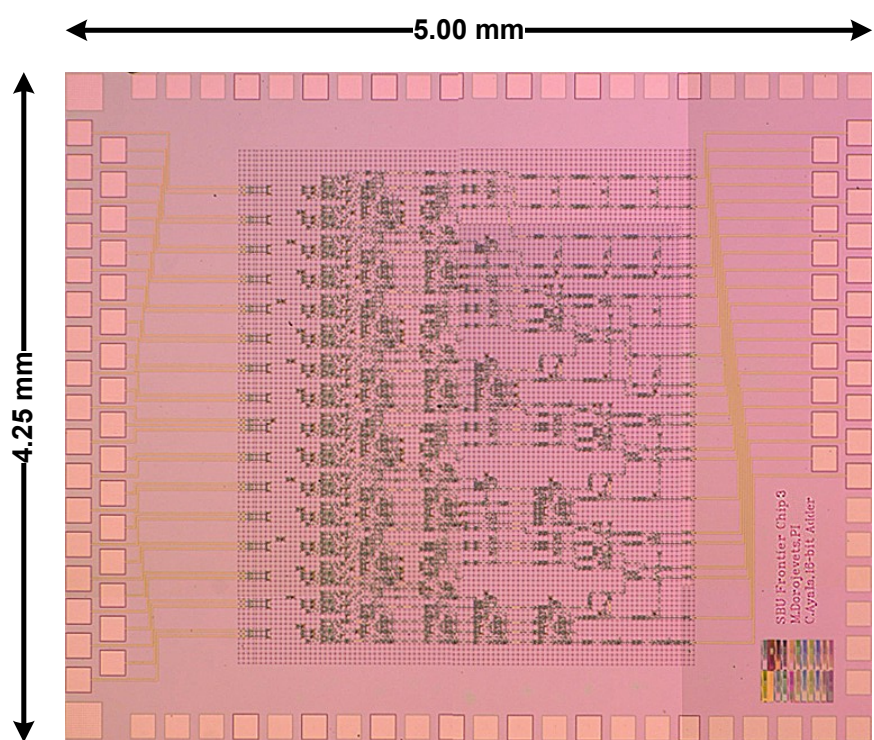


(b) 4-bit example of the input shift register.

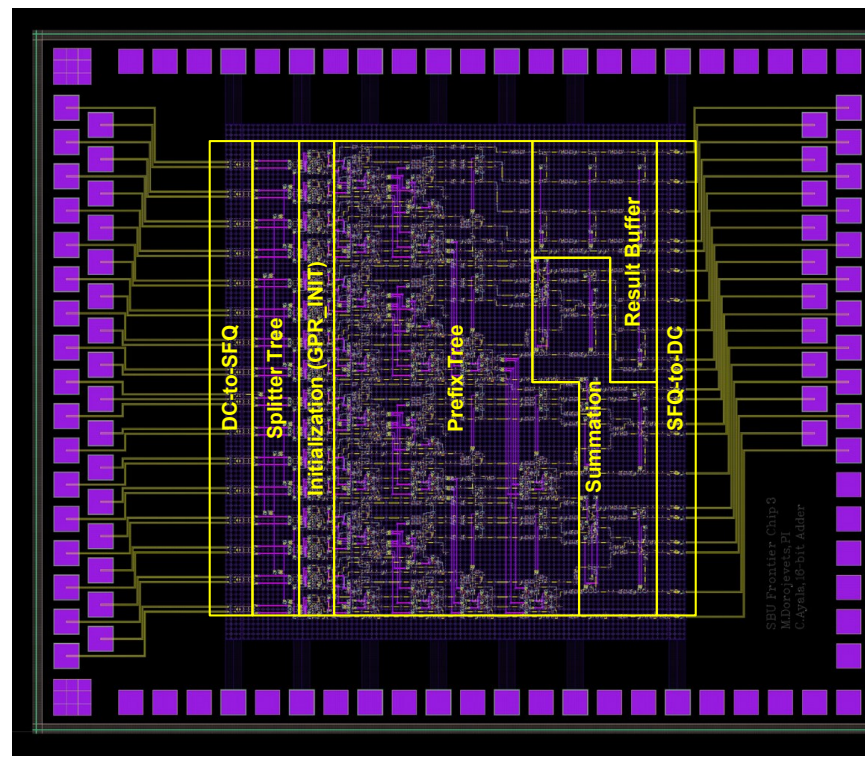


(c) 4-bit example of the output compressor.

Figure 5.12: Schematics of the supplemental circuits to facilitate high-speed testing of the 16-bit HSTA.

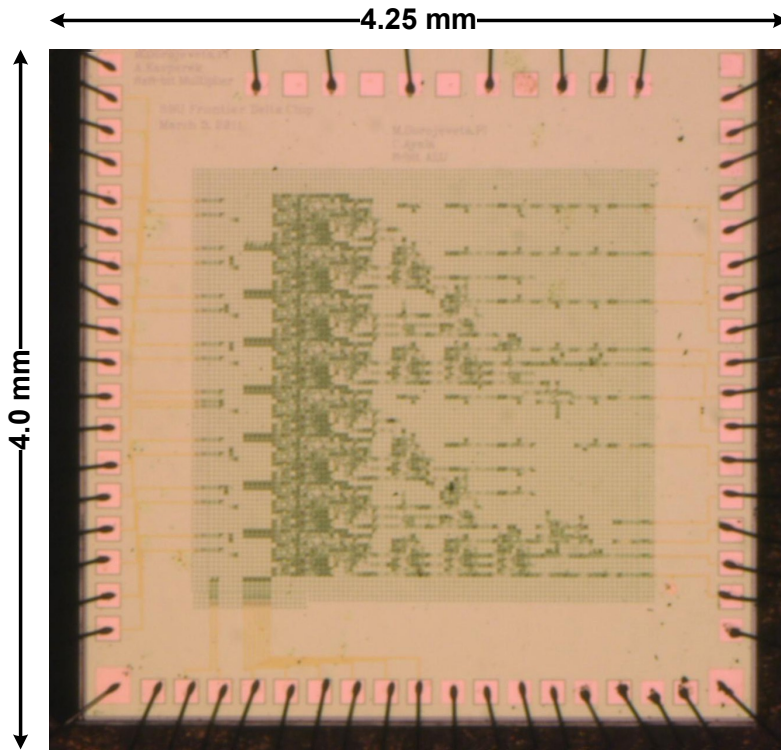


(b) Microphotograph

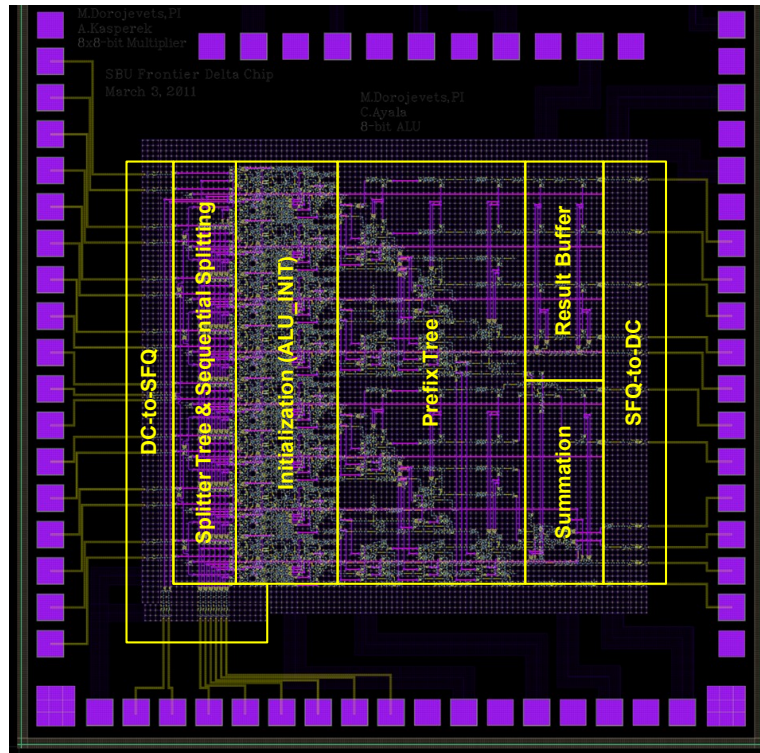


(a) Layout

Figure 5.13: Layout and microphotograph of the 16-bit HSTA chip for low-frequency testing.

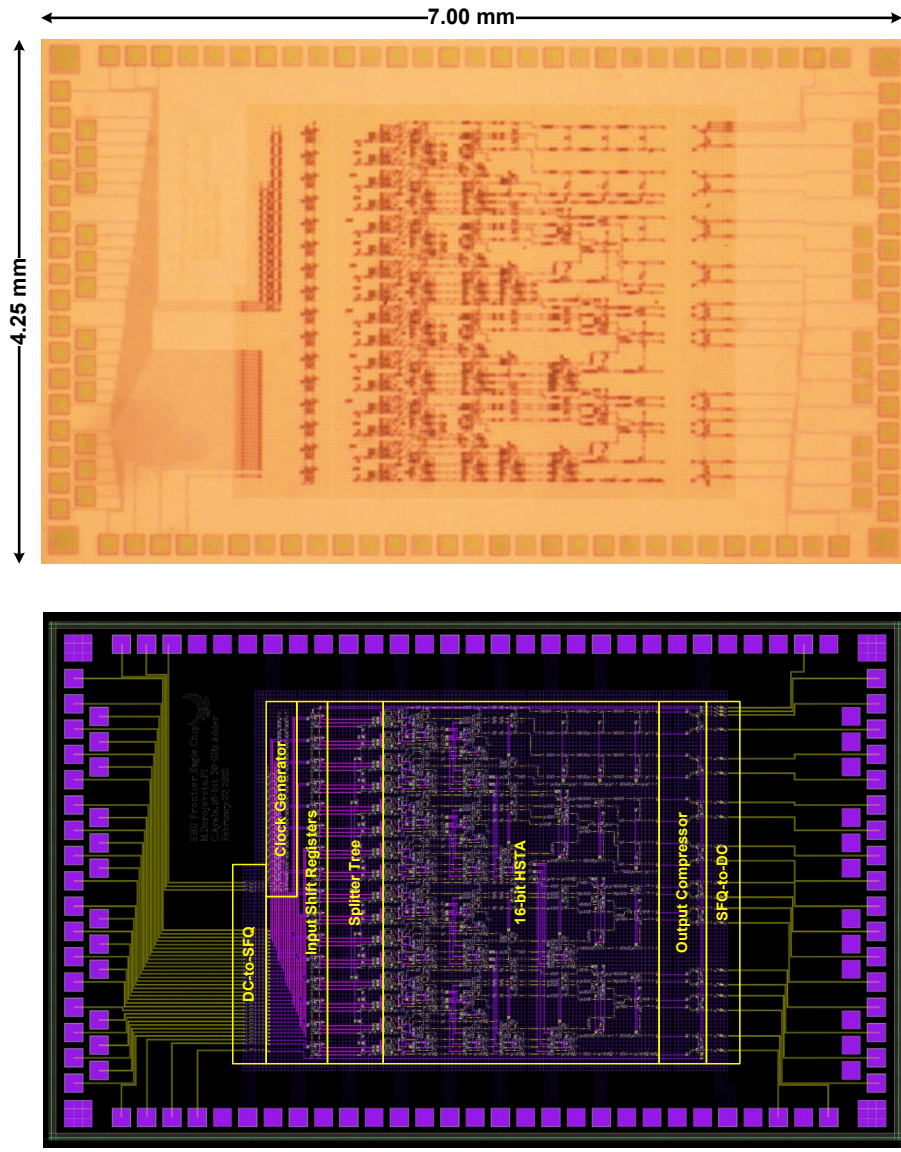


(b) Microphotograph



(a) Layout

Figure 5.14: Layout and microphotograph of the 8-bit HSTALU chip for low-frequency testing (Delta chip).



(a) Layout

(b) Microphotograph

Figure 5.15: Layout and microphotograph of the 16-bit HSTA with on-chip high-frequency test circuits (Eagle chip).

5.4.2 Simulation Results

The CONNECT cell library allowed us to easily evaluate our designs with respect to complexity, bias current, area, latency and performance. An important benchmark for our circuits is the frequency dependent DC bias margins which reveal how robust our designs are to changing bias voltages at different clock rates, a capability that is also provided by the CONNECT cell library. This section summarizes the simulation results of the 16-bit HSTA and 8-bit HSTALU.

5.4.2.1 16-bit HSTA Results

Since the core of the 16-bit HSTA is the same for both the low- and high-frequency test chips, only the results of the latter are summarized here. The breakdown of complexity is detailed in Table 5.10 on page 115 and an overall summary of the design is shown in Table 5.11 on page 115. Figure 5.16 on page 114 illustrates the DC bias margins at different clock rates. At the target clock rate of 30 GHz, we have +20% / -16% margins which is relatively wide for a circuit of this scale.

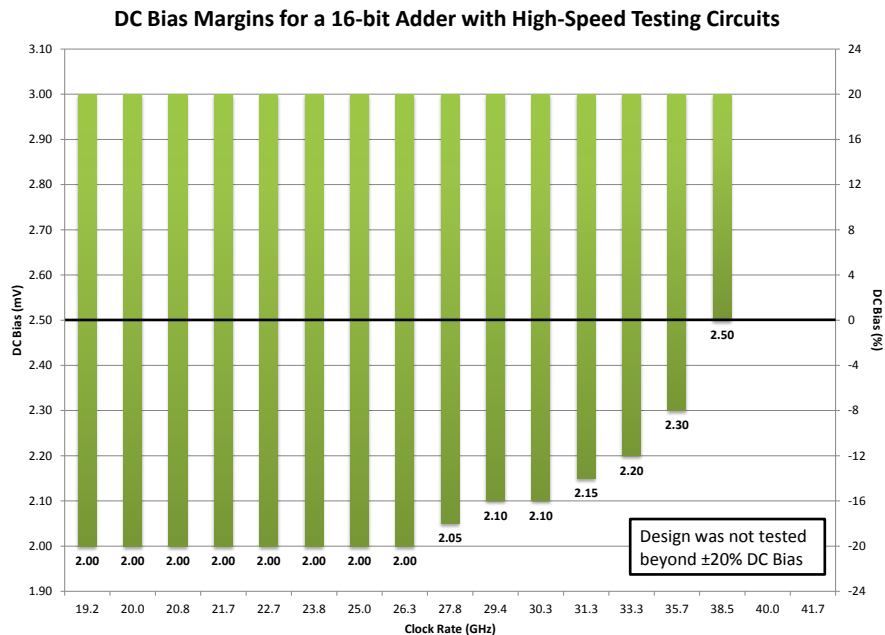


Figure 5.16: DC bias margins for the 16-bit HSTA (Eagle chip).

Table 5.10: Breakdown of complexity for the 16-bit HSTA with on-chip high-speed testing circuits.

Component	Complexity (JJs)	Bias Current (mA)
Clock Generator	828	113.718
Input Shift Register	1072	131.365
Adder - Island 0	3679	430.279
Adder - Island 1	3176	410.307
Adder - Island 2	3086	419.967
Output Compression	584	60.195
DC-to-SFQ	180	14.625
SFQ-to-DC	180	29.622
Total	12785	1610.078

Table 5.11: Summary of the 16-bit HSTA.

Adder Datapath Width	16 bits
Peak Processing Rate	38.5 GHz (simulated)
Latency (2.5 mV bias)	352 ps (simulated)
Complexity (core)	9941 JJs
Complexity (total)	12785 JJs
Bias Current (total)	1.61 A
Area (core)	8.5 mm ²

5.4.2.2 8-bit HSTALU Results

The breakdown of complexity is detailed in Table 5.10 on page 115 and an overall summary of the design is shown in Table 5.11 on page 115. Figure 5.16 on page 114 shows the DC bias margins at different clock rates. At the target clock rate of 30 GHz, we have +20% / -16% margins.

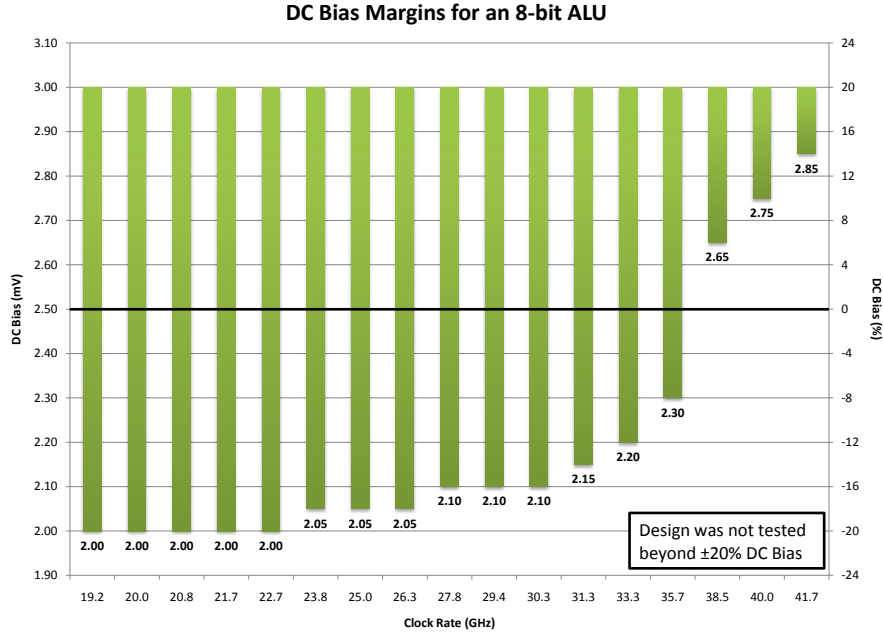


Figure 5.17: DC bias margins for the 8-bit HSTALU (Delta chip).

Table 5.12: Breakdown of complexity for the 8-bit HSTALU.

Component	Complexity (JJs)	Bias Current (mA)
DC-to-SFQ	125	10.16
Main circuit	8832	1098.48
SFQ-to-DC	99	16.29
Total	9056	1124.93

Table 5.13: Summary of the 8-bit HSTALU.

ALU Datapath Width	8 bits
# of Arithmetic Operations	8
# of Logical Operations	12
Peak Processing Rate	42 GHz (simulated)
Latency (2.5 mV bias)	374 ps (simulated)
Complexity (core)	8832 JJs
Bias Current	1.13 A
Area (core)	7.2 mm ²

5.4.3 Experimental Testing

We conducted testing at Yokohama National University in Japan where the Yoshikawa Laboratory provided an assortment of equipment to verify the functionality of superconducting chips and measure the operating margins (Table 5.14 on page 118). Chips are wire-bonded onto a chip holder where it will be seated inside the chamber of a chip probe. A multimeter is used to check for wire continuity to ensure that all pads have a connection through the probe. To screen out the external magnetic field, two layers of Mu-metal magnetic shields surround the chamber of the chip probe. Next, the probe is pre-cooled to $T = 77$ K using liquid Nitrogen. This process is necessary to avoid the rapid temperature change it would experience if the chip was to go directly to $T = 4.2$ K in liquid Helium, which may create unwanted flux trapping and disturb the operation of the circuit. After pre-cooling, the chip probe is once again tested for wire continuity before it is slowly lowered into the Helium Dewar, a process that can take 20-30 minutes to gradually bring the circuit to 4.2 K where the temperature is low enough to enter the superconductive critical temperature of Niobium ($T = 9.2$ K) [115]. Once the chip is sufficiently cooled, the wire connections are checked one last time and then the probe is connected to the testing equipment as shown on Figure 5.18 on page 120.

The circuit is powered on in a specific order. First the SFQ-to-DC circuits are slowly biased up to their expected nominal currents, followed by the main circuit and finally the DC-to-SFQ circuits. Test patterns prepared on the data generator beforehand are transmitted to the circuit and a resulting waveform of the outputs are displayed on oscilloscopes. Due to parameter variations in fabrication, samples of the same chip will most likely not behave the same way. It is sometimes necessary to fine tune the bias supplies to get a stable

Table 5.14: List of testing equipment.

Equipment	Manufacturer and Model	Description
Data generator	Tektronix DG2020A	Generates input data into the test chip.
Input attenuator	Tamagawa Electronics VBA-641A	Lowers the voltage of the incoming input from the data generator to mV levels for superconductor circuits.
Power supply	Kikusui PMR18-2.5DU	Provides DC bias current to the chip.
Low-pass filter	Custom made	Minimizes noise across the DC bias current supplies.
Differential amplifier	Stanford Research Systems SR560	Amplifies output of the superconductor circuit for viewing on oscilloscopes.
Oscilloscope	Agilent Technologies DSO5014A, and others	Displays the output of the chip.
Probe connection box	Custom made	Connects all inputs, outputs, power and ground lines between testing equipment and the chip probe.
Chip probe	Custom made	Links the connection box with the chip to be placed at superconducting temperatures.
Helium Dewar	Cryofab Inc. CMSH 88	Cools down the test chip to $T = 4.2$ K
Magnetic shield	Standard Mu-metal shield	Shields the test chip from external magnetic fields which can disturb the circuit's operation.

output waveform. Too much bias will create an oscillating waveform whereas too little bias will create either no output or an unstable waveform. A logical '1' is represented by a rising or falling edge on the waveforms and a logical '0' is a steady voltage level.

Once a stable output waveform is achieved for a particular test pattern, the DC bias margins can be measured. It is the process of adjusting the bias supply to the absolute maximum and minimum before the stable waveform becomes disturbed either through oscillation or dropped outputs. The maximum and minimum bias current values are then recorded for the present test pattern. We repeat this process for each test case prepared on the data generator and find what are the overlapped margins across all test cases. Where these margins intersect is ultimately the operating margins of the circuit.

It is not always possible to get a stable output waveform on the very first try. Flux trapping is a common problem with superconductor circuits and it is often necessary to deflux or warm-up the chip out of the superconductive state to remove the trapped flux that is hindering operation of the circuit. If a chip is not operating as expected, we try to deflux the chip, slowly cool it again and repeat the test. As a general rule of thumb, we would repeat this cycle at least 2 to 3 times before we deem a chip sample as non-functional and move on to the next one.

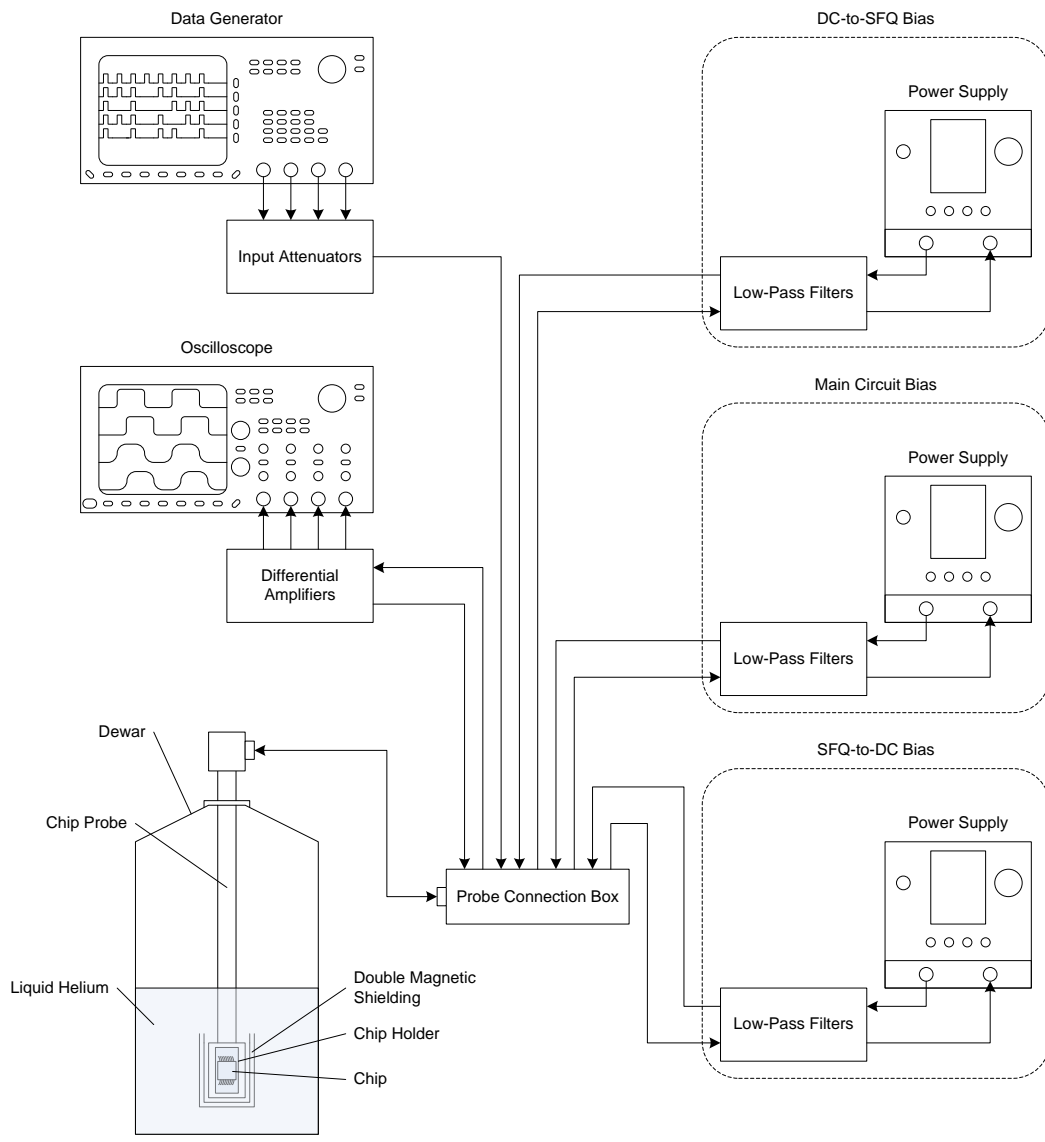


Figure 5.18: Block diagram of the testing environment.



Figure 5.19: Equipment used for experimental testing: (A) data generator, (B) input attenuators, (C) power supplies, (D) low-pass filters, (E) probe connection box, (F) oscilloscopes, (G) differential amplifiers, (H) chip probe, (I) Helium Dewar with chip probe inserted, (J) magnetic shields, and (K) magnetic shields attached to chip probe.

5.4.4 Chip Testing Results

5.4.4.1 16-bit HSTA

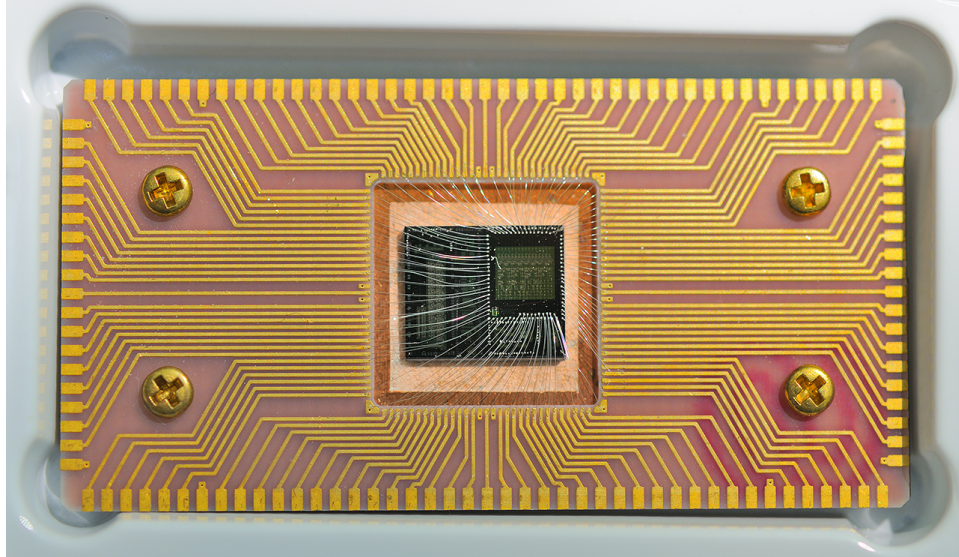


Figure 5.20: 16-bit HSTA chip for low-frequency testing wire-bonded to a chip holder.

As previously mentioned the 16-bit HSTA design underwent two chip designs, the first for low-frequency functional testing (August 2011) and the second for high-speed testing (August 2012). To test the first chip, each data pattern was set with 16-bit input data for both operands A and B followed by Ready after some time delay. The test is sent twice to produce an even number of transitions on the waveform so the oscilloscope can lock onto the output signal. This chip showed excellent results, passing all test cases including the critical case of carry propagating through all bits (all 1's plus 1). Four random cases are shown on Figures 5.21 thru 5.24. The measured DC bias margins intersecting across all cases were $+9.81\%$ / -10.68% .

To test the second chip containing additional circuits for high-speed testing, we loaded 16-bit data in parallel to each of the input shifter registers. After sufficient delay, we supply a trigger pulse into the clock generator to start the 16-pulse high-frequency clock, sending 16 waves of data into the adder with each subsequent wave being the same as the previous but shifted to the right by 1 bit. The output compression circuits create an XOR signature of the 16 outputs generated from the adder. Afterwards, we supply a pulse to read out the states for each compressed output bit. Using a Perl program, we calculate the expected states for each bit and compare them to the output waveform.

Unfortunately, the second chip did not fare so well. In the low frequency operation of this chip, the input shift register for operand A showed a few missing bits in the serial output after supplying all 1's to the register (Figure 5.25 on page 128). The same test for operand B showed no output at all. Running a single-stepping test through each of the 16 test vectors revealed mostly incorrect or unstable output and when running at high-frequency, we observed similar results as expected from our initial low-frequency experiments of this chip.

Based on data and measurements of the fabrication run (named ADP627) for this chip by our colleagues, almost all other circuits also showed malfunction and it was generally agreed that the quality of this fabrication run was at a lower standard of quality than usual [116].

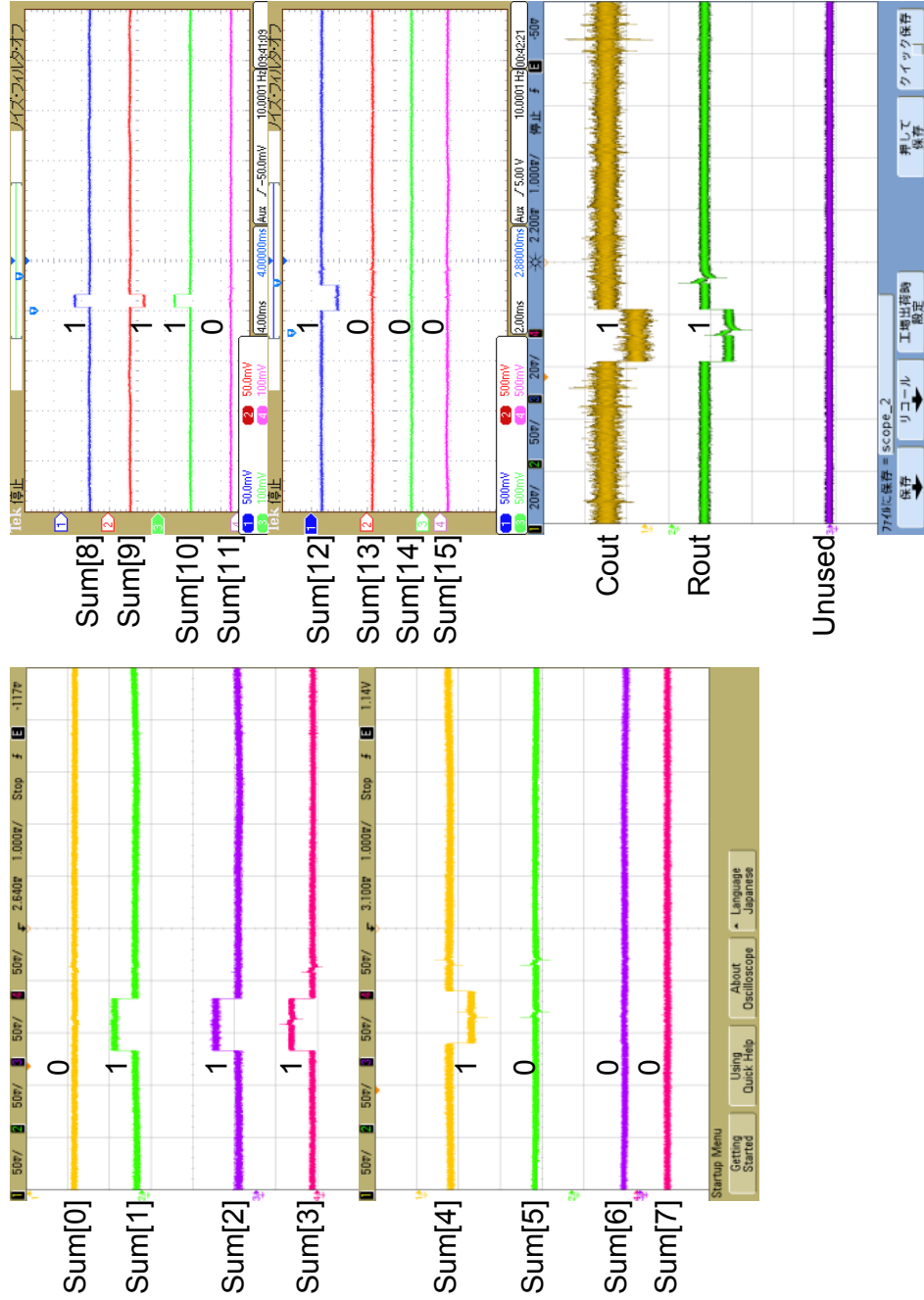


Figure 5.21: Low-frequency random test #1, A = 56811, B = 14643, and Sum = 71454.

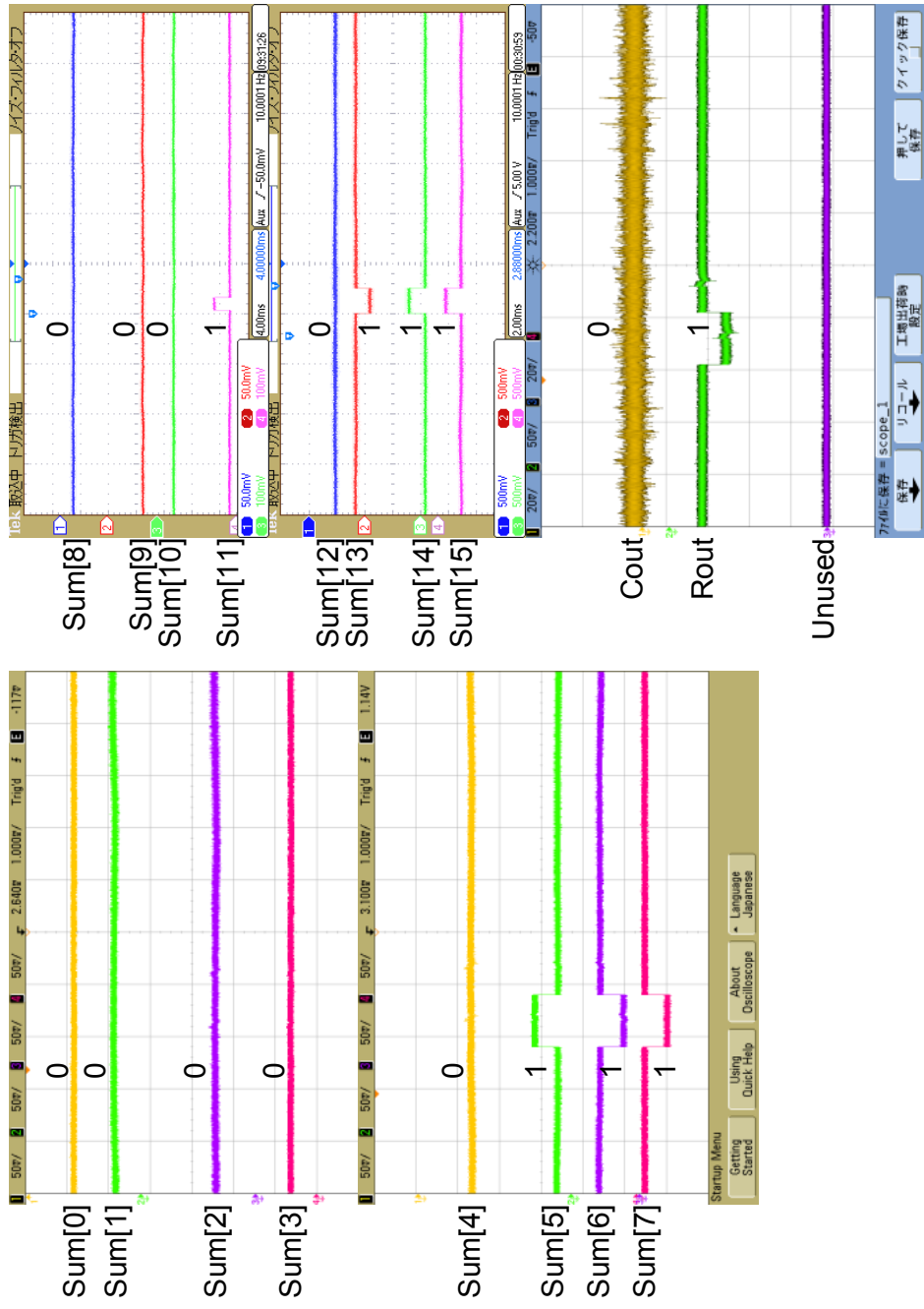


Figure 5.22: Low-frequency random test #2, A = 8724, B = 50892, and Sum = 59616.

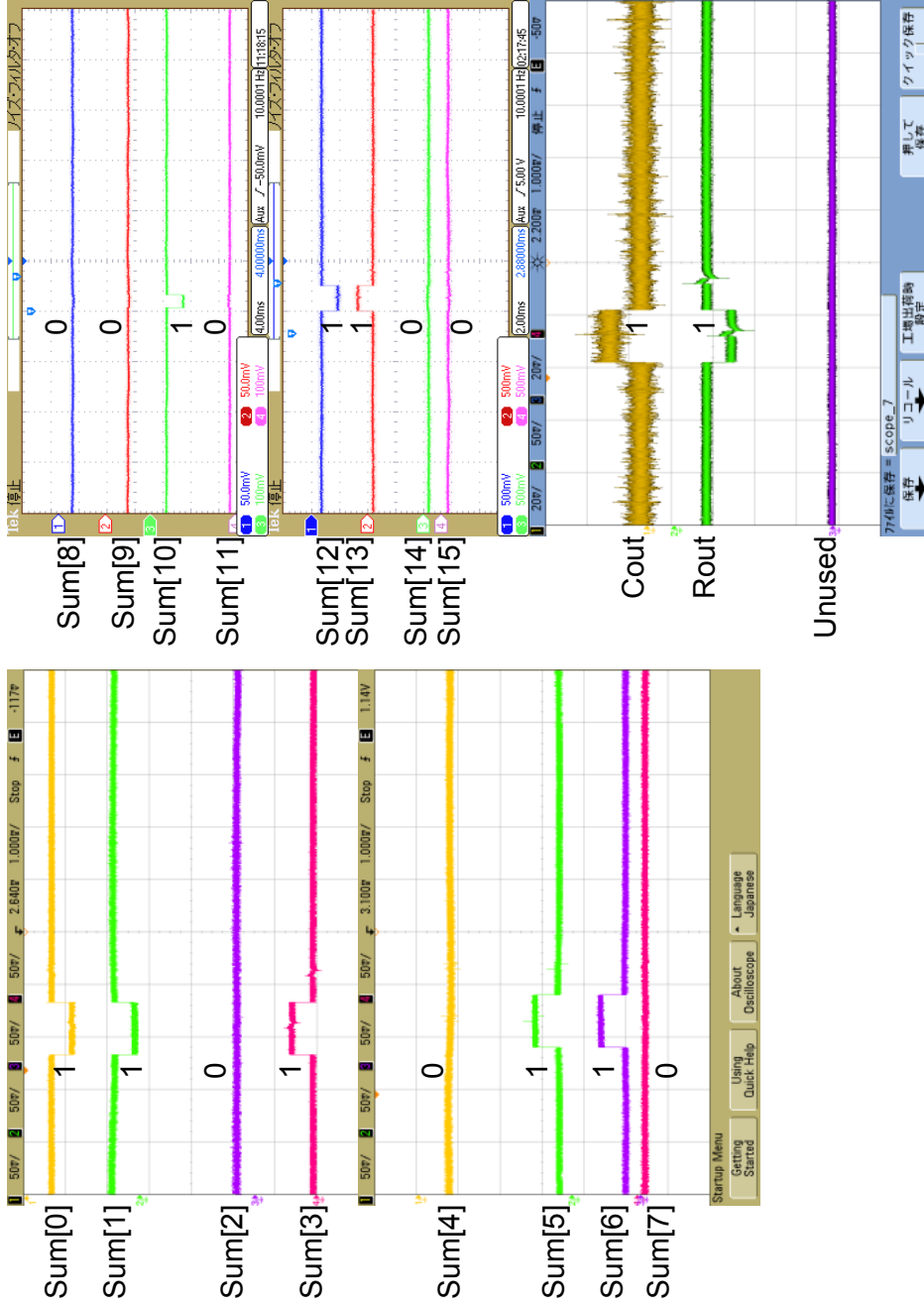


Figure 5.23: Low-frequency random test #3, $A = 13982$, $B = 64973$, and $Sum = 78955$.

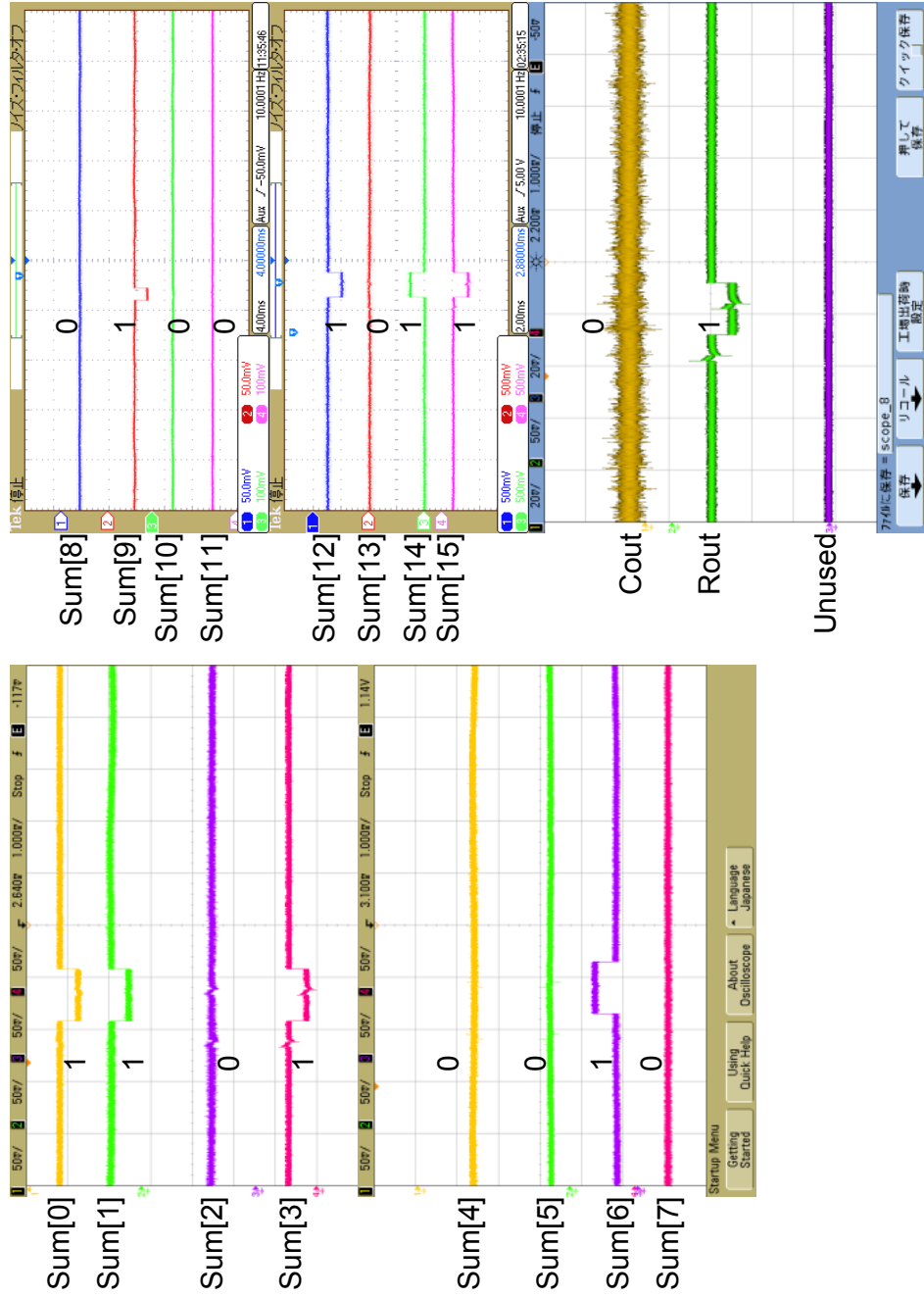


Figure 5.24: Low-frequency random test #4, A = 44636, B = 9199, and Sum = 53835.

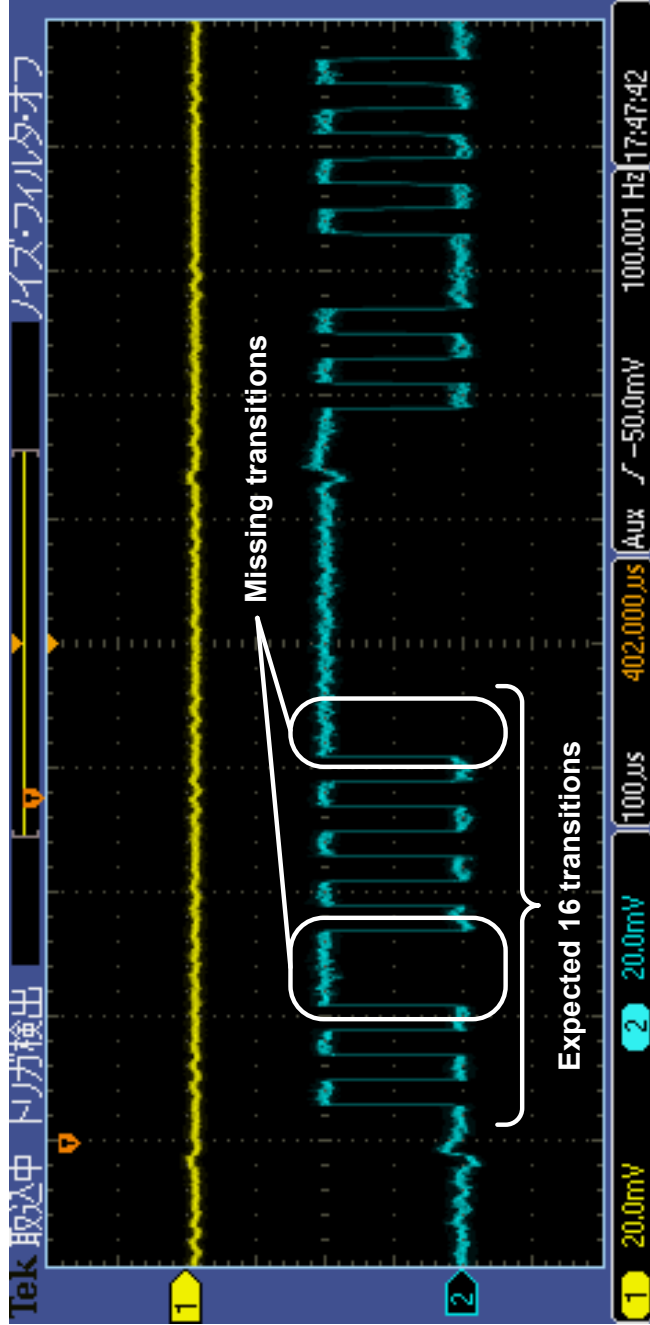


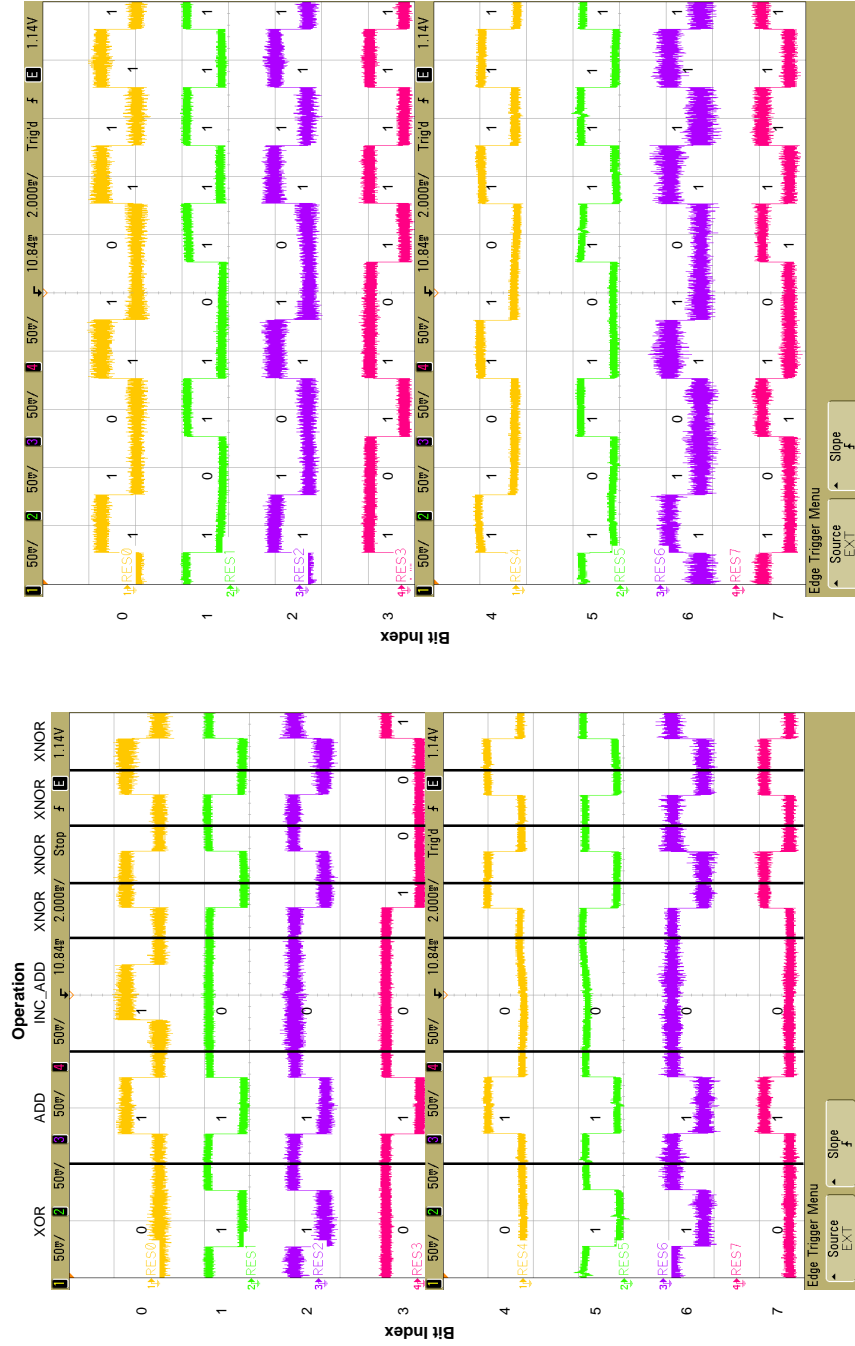
Figure 5.25: Low-frequency serial output of the shifter for input A (blue trace) and B (yellow trace) on the 16-bit HSTA chip designed for high-speed testing. Both input shifters were initialized to all logical 1's (16-bit) so we expected to see 16 transitions each yet we only observed 13 for input A and none for input B.

5.4.4.2 8-bit HSTALU

The 8-bit HSTALU had one chip design capable of only low-frequency functional testing. To test this chip, two 8-bit input operands A and B, along with a 6-bit control signal and an additional 2-bit control signal for the least significant bit were sent by the data pattern generator followed by the Ready signal to start the operation. Table 5.15 on page 131 shows a list of test vectors supplied by the data generator and Figure 5.26 on page 130 shows the resulting output on the oscilloscope.

All operations in the table produced correct results except for XNOR which had difficulty in demonstrating correctness across the 8-bit output simultaneously. We traced the problem to the unstable operation of the XOR gate used to selectively invert operands (Figure 4.2a on page 62). When it receives a pulse in both inputs (i.e. the operand is '1' and the control signal to invert is also '1'), the output incorrectly shows as '1' for most bits. Therefore, we decided to demonstrate the XNOR operation on bit 3 which was the only bit that did not exhibit this problem.

Another malfunction we encountered in this chip is the lack of generating the prefix signal G, a vital signal for arithmetic operations as well as logical operations based on AND and OR. We were able to determine this problem when none of the AND/OR-based logical functions worked and when hand-picked vectors for ADD did not flip bits where a carry was expected. The test vectors on Table 5.15 were selected to avoid the generation of G. We attribute this malfunction to the small operating margins of the TRS gate and combined with the low overall yield of the wafers containing the HSTALU chip samples, the probability of confirming full functionality was low. Despite this, we still showed several operations functioning correctly and we measured the overlapped DC bias margins across these working cases to be $\pm 1.8\%$.



(b) Mixing vectors for a fixed ADD operation.

(a) Mixing of arithmetic and logic operations.

Figure 5.26: Waveforms demonstrating various operations of the 8-bit ALU. Refer to the upper section of Table 5.15 on page 131 to see the test vectors for (a) and the lower section for (b).

Table 5.15: Test vectors supplied to the 8-bit HSTALU with waveform outputs shown on Figure 5.26 on page 130.

Operation	Operand A	Operand B	Output
XOR	0b11001100	0b10101010	0b01100110
ADD	0b00000000	0b11111111	0b11111111
INC_ADD	0b00000000	0b00000000	0b00000001
XNOR	0b00000000	0b00000000	0b11111111
XNOR	0b00000000	0b00001000	0b11110111
XNOR	0b00001000	0b00000000	0b11110111
XNOR	0b00001000	0b00001000	0b11111111
ADD	0b11111111	0b00000000	0b11111111
ADD	0b01010101	0b00000000	0b01010101
ADD	0b10101010	0b00000000	0b10101010
ADD	0b00000000	0b11111111	0b11111111
ADD	0b00000000	0b01010101	0b01010101
ADD	0b00000000	0b10101010	0b10101010
ADD	0b01010101	0b10101010	0b11111111
ADD	0b10101010	0b01010101	0b11111111
ADD	0b00001111	0b11110000	0b11111111
ADD	0b11110000	0b00001111	0b11111111

Chapter 6

Conclusions

Outline

6.1	Completed Work	132
6.2	Future Work	134

6.1 Completed Work

The goal of this research is to establish a new frontier in the development of high-complexity, high-performance designs using energy-efficient superconductor logic. To this end, we have conducted superconductor design studies on three different types of structures, namely: a ripple-carry adder, a Kogge-Stone ALU and a hybrid sparse-tree ALU. All three candidates were designed for a 32-bit data width and simulated using our VHDL cell library tuned to the HYPRES 1.5 μm 4.5 kA/cm² technology for both RSFQ and ERSFQ superconductor logic. Metrics such as latency, maximum processing rate, complexity, bias current and total power were obtained for each design.

The 32-bit superconductor ripple-carry adder has a design complexity of 952 JJs and an average latency of 520 ps. Its maximum processing rate is only 2 GHz but it is a very energy efficient design as it is capable of producing 2,418 TOPS/W at 4.2 K temperature in ERSFQ logic, consuming only 827 nW. Unfortunately, the ripple-carry adder is not very scalable for wide datapath, high-performance architectures. Its concept, however, would prove to be very useful in small ripple-carry adder chains found in the hybrid sparse-tree adder. Furthermore, its use of the T1 gate as a fast counting circuit is applicable for high-speed compressors used in multipliers [110].

On the other end of the design spectrum, a 32-bit Kogge-Stone ALU presents a structure that can easily exploit the advantages of asynchronous

hybrid wave-pipelining techniques for a scalable, high-performance unit. It has a design complexity of 36,073 JJs and an average latency of 470 ps. Its maximum processing rate is 23.3 GHz. The large complexity resulted in a power consumption of 225 μW yielding an energy efficiency of 104 TOPS/W at 4.2 K in ERSFQ logic.

A scaled down 8-bit version of the Kogge-Stone ALU has been physically implemented in RSFQ using the HYPRES 1.5 μm 4.5 kA/cm² technology, as a joint effort between SBU and HYPRES. The chip fully operates at the processing rate of 20 GHz, demonstrating that our asynchronous hybrid wave-pipelining techniques is a suitable approach for designing high-performance circuits.

The Kogge-Stone structure required a very dense tree of carry-merge logic cells and it has a complex stage-to-stage interconnect that is difficult to layout at higher data widths. The 32-bit hybrid sparse-tree ALU resolved these two challenges by incorporating a sparse-tree to generate carries into each 4-bit group while integrating some of the low-complexity techniques of the ripple-carry adder. With this approach, there is an increase in complexity in calculating the final sum as it is necessary to serially pre-calculate the 4-bit sum of each group during the first four stages of the adder and use 4-bit carry-skip adders to produce the final result. Despite these complications, the hybrid structure still resulted in a $\sim 55\%$ improvement in energy efficiency over the Kogge-Stone ALU, while sacrificing only $\sim 2.6\%$ in processing rate and $\sim 6.3\%$ in latency. Overall, the ERSFQ hybrid sparse-tree ALU has a design complexity of 21,991 JJs, an average latency of 502 ps, a maximum processing rate of 22.7 GHz while consuming 142 μW of power resulting in an energy efficiency of 160 TOPS/W at 4.2 K. When compared to a 9 GHz Intel ALU [114], the hybrid sparse-tree ALU is ~ 2.5 times faster and consumes ~ 14.5 times less power after taking into account the additional power to cool the circuit to 4.2 K.

Using SFQ CAD tools developed by our colleagues in Yokohama National University and Nagoya University, we implemented and tested an 8-bit ALU and a 16-bit adder based on the hybrid sparse-tree structure using RSFQ logic in the ISTEK 1.0 μm 10 kA/cm² technology. The target clock rate for both designs is 30 GHz. The 16-bit adder passed all low-frequency tests with a DC bias margin of +9.81% / -10.68% whereas the 8-bit ALU demonstrated several operations at low-frequency with a DC bias margin of $\pm 1.8\%$.

6.2 Future Work

Due to the recently volatile foundry conditions in Japan, it would be worthwhile to have a second fabrication run of the hybrid sparse-tree 8-bit ALU and 16-bit adder. Since the TRS gate of the CONNECT cell library had relatively low margins, it would be interesting to re-design the ALU_INIT block of the 8-bit ALU so that the TRS-based data pulse counter is replaced with a T1-based counter. To preset the T1, we simply include an additional MRG gate at the input to create an extra port to preset. This can also eliminate the Ctrl_sub_bar signal since there would be no need to reset the T1 counter; it is in fact naturally reset when the gate is read out. It also eliminates a D2FF gate to store the preliminary P signal because of the built-in storage of T1. Since the T1 gate has been successfully demonstrated in the Summation blocks of the adder as well as an 8x8 carry-save multiplier [110], this modification will most likely improve the chances of fully demonstrating the ALU, at least for low-frequency testing. The only difficulty is integrating the extra MRG within the already compact ALU_INIT block, but with the elimination of the D2FF gate, extra room in the layout should be available.

In this research, all of the ERSFQ work was only done as a design study to estimate the energy efficiency of our units, and scaled down physical implementations were done in RSFQ. It would be beneficial to see the work physically implemented and tested in ERSFQ to see how well the measured power consumption aligns with our simulations. Also fascinating are the other emerging energy-efficient superconductor logic families such as Reciprocal Quantum Logic (RQL) [117, 118, 119] and Adiabatic Quantum-Flux Parametron (AQFP) logic [120, 121]. Both families have very limited CAD support and it would be interesting to research new approaches on how to efficiently implement large-scale designs in these logic families and perhaps set new benchmarks in low-power computing.

Bibliography

- [1] D. A. Patterson and J. L. Hennessy, *Computer Organization and Design: The Hardware/Software Interface*, 4th ed. Morgan Kaufmann, 2008. (Cited on page 2).
- [2] (2011) International Technology Roadmap for Semiconductors. International Technology Roadmap for Semiconductors. [Online]. Available: <http://www.itrs.net/> (Cited on page 2).
- [3] M. Bohr and K. Mistry. (2011, May) Intel's Revolutionary 22 nm Transistor Technology. Intel Newsroom. Intel. [Online]. Available: http://download.intel.com/newsroom/kits/22nm/pdfs/22nm-Details_Presentation.pdf (Cited on page 2).
- [4] T. Kuroda, "Low-power, high-speed CMOS VLSI design," in *Computer Design: VLSI in Computers and Processors, 2002. Proceedings. 2002 IEEE International Conference on*, 2002, pp. 310 – 315. (Cited on page 2).
- [5] J. G. Koomey, "Growth in data center electricity use 2005 to 2010," Stanford University, Tech. Rep., Aug. 2011. [Online]. Available: <http://www.mediafire.com/file/zzqna34282fr2f/koomeydatacenterelectuse2011finalversion.pdf> (Cited on page 3).
- [6] W. Chen, J. Xu, P. Lai, Y. Li, and S. Xu, "Gate Leakage Properties of MOS Devices with Tri-Layer High-k Gate Dielectric," in *Electron Devices and Solid-State Circuits, 2005 IEEE Conference on*, Dec. 2005, pp. 695 – 698. (Cited on page 3).
- [7] J. Huang, P. Kirsch, J. Oh, S. H. Lee, P. Majhi, H. Harris, D. Gilmer, G. Bersuker, D. Heh, C. S. Park, C. Park, H.-H. Tseng, and R. Jammy, "Mechanisms Limiting EOT Scaling and Gate Leakage Currents of High-/Metal Gate Stacks Directly on SiGe," *Electron Device Letters, IEEE*, vol. 30, no. 3, pp. 285 –287, Mar. 2009. (Cited on page 3).

- [8] C.-C. Lu, K.-S. Chang-Liao, C.-Y. Lu, S.-C. Chang, and T.-K. Wang, "Leakage effect suppression in charge pumping measurement and stress-induced traps in high-k Gated MOSFETs," in *Semiconductor Device Research Symposium, 2007 International*, Dec. 2007, pp. 1–2. (Cited on page 3).
- [9] R. Chau, J. Brask, S. Datta, G. Dewey, M. Doczy, B. Doyle, J. Kavalieros, B. Jin, M. Metz, A. Majumdar, and M. Radosavljevic, "Application of high-[kappa] gate dielectrics and metal gate electrodes to enable silicon and non-silicon logic nanotechnology," *Microelectronic Engineering*, vol. 80, pp. 1 – 6, 2005, 14th biennial Conference on Insulating Films on Semiconductors. [Online]. Available: <http://www.sciencedirect.com/science/article/B6V0W-4GB2584-7/2/e84d38f9af611851a6a00d2e169a4815> (Cited on page 3).
- [10] A. Bhoj and N. Jha, "Design of ultra-low-leakage logic gates and flip-flops in high-performance FinFET technology," in *Quality Electronic Design (ISQED), 2011 12th International Symposium on*, Mar. 2011, pp. 1–8. (Cited on page 3).
- [11] P. Mishra, A. Bhoj, and N. Jha, "Die-level leakage power analysis of FinFET circuits considering process variations," in *Quality Electronic Design (ISQED), 2010 11th International Symposium on*, Mar. 2010, pp. 347–355. (Cited on page 3).
- [12] S. Tawfik and V. Kursun, "Compact FinFET Memory Circuits with P-Type Data Access Transistors for Low Leakage and Robust Operation," in *Quality Electronic Design, 2008. ISQED 2008. 9th International Symposium on*, Mar. 2008, pp. 855–860. (Cited on page 3).
- [13] D. James, "Intel Ivy Bridge unveiled - The first commercial tri-gate, high-k, metal-gate CPU," in *Custom Integrated Circuits Conference (CICC), 2012 IEEE*, Sep. 2012, pp. 1–4. (Cited on page 3).
- [14] C. Auth, C. Allen, A. Blattner, D. Bergstrom, M. Brazier, M. Bost, M. Buehler, V. Chikarmane, T. Ghani, T. Glassman, R. Grover, W. Han, D. Hanken, M. Hattendorf, P. Hentges, R. Heussner, J. Hicks, D. Ingerly, P. Jain, S. Jaloviar, R. James, D. Jones, J. Jopling, S. Joshi, C. Kenyon, H. Liu, R. McFadden, B. McIntyre, J. Neiryneck, C. Parker, L. Pipes, I. Post, S. Pradhan, M. Prince, S. Ramey, T. Reynolds, J. Roesler, J. Sandford, J. Seiple, P. Smith, C. Thomas, D. Towner, T. Troeger,

- C. Weber, P. Yashar, K. Zawadzki, and K. Mistry, “A 22nm high performance and low-power CMOS technology featuring fully-depleted tri-gate transistors, self-aligned contacts and high density MIM capacitors,” in *VLSI Technology (VLSIT), 2012 Symposium on*, Jun. 2012, pp. 131 – 132. (Cited on page 3).
- [15] E. Karl, Y. Wang, Y.-G. Ng, Z. Guo, F. Hamzaoglu, U. Bhattacharya, K. Zhang, K. Mistry, and M. Bohr, “A 4.6GHz 162Mb SRAM design in 22nm tri-gate CMOS technology with integrated active VMIN-enhancing assist circuitry,” in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2012 IEEE International*, Feb. 2012, pp. 230 –232. (Cited on page 3).
- [16] C. Belady and C. Malone, “Data center power projections to 2014,” in *Thermal and Thermomechanical Phenomena in Electronics Systems, 2006. IThERM '06. The Tenth Intersociety Conference on*, 30 2006-June 2 2006, pp. 439 –444. (Cited on page 3).
- [17] P. Krein, “A discussion of data center power challenges across the system,” in *Energy Aware Computing (ICEAC), 2010 International Conference on*, dec. 2010, pp. 1 –3. (Cited on page 3).
- [18] H. Qi and A. Gani, “Research on mobile cloud computing: Review, trend and perspectives,” in *Digital Information and Communication Technology and it's Applications (DICTAP), 2012 Second International Conference on*, May 2012, pp. 195 –202. (Cited on page 3).
- [19] R.-H. Di, H. Lv, T. Wang, X.-H. Zhang, and D.-M. Xiao, “Research on the impact of cloud computing trend on E-government framework,” in *E -Business and E -Government (ICEE), 2011 International Conference on*, May 2011, pp. 1 –4. (Cited on page 3).
- [20] Y. Jadeja and K. Modi, “Cloud computing - concepts, architecture and challenges,” in *Computing, Electronics and Electrical Technologies (ICCEET), 2012 International Conference on*, Mar. 2012, pp. 877 –880. (Cited on page 3).
- [21] S. Sengupta, V. Kaulgud, and V. Sharma, “Cloud Computing Security–Trends and Research Directions,” in *Services (SERVICES), 2011 IEEE World Congress on*, Jul. 2011, pp. 524 –531. (Cited on page 3).
- [22] S. Zhang, S. Zhang, X. Chen, and X. Huo, “Cloud Computing Research and Development Trend,” in *Future Networks, 2010. ICFN '10. Second International Conference on*, Jan. 2010, pp. 93 –97. (Cited on page 3).

- [23] S. Mathew, M. Anders, R. Krishnamurthy, and S. Borkar, “A 4-GHz 130-nm address generation unit with 32-bit sparse-tree adder core,” *Solid-State Circuits, IEEE Journal of*, vol. 38, no. 5, pp. 689 – 695, May 2003. (Cited on pages 4, 54, and 88).
- [24] S. Mathew, M. Anders, B. Bloechel, T. Nguyen, R. Krishnamurthy, and S. Borkar, “A 4-GHz 300-mW 64-bit integer execution ALU with dual supply voltages in 90-nm CMOS,” *Solid-State Circuits, IEEE Journal of*, vol. 40, no. 1, pp. 44 – 51, Jan. 2005. (Cited on pages 4, 54, and 88).
- [25] V. Michal, E. Baggetta, M. Aurino, S. Bouat, and J. Villegier, “Superconducting RSFQ logic: Towards 100GHz digital electronics,” in *Radioelektronika (RADIOELEKTRONIKA), 2011 21st International Conference*, Apr. 2011, pp. 1 –8. (Cited on pages 5 and 11).
- [26] (2005, Aug.) Superconducting Technology Assessment (STA). National Security Agency. [Online]. Available: <http://www.nitrd.gov/pubs/nsa/sta.pdf> (Cited on pages 5, 7, 8, 10, 11, 12, 13, and 19).
- [27] K. Likharev and V. Semenov, “RSFQ logic/memory family: a new Josephson-junction technology for sub-terahertz-clock-frequency digital systems,” *Applied Superconductivity, IEEE Transactions on*, vol. 1, no. 1, pp. 3 –28, Mar. 1991. (Cited on pages 4, 8, and 11).
- [28] B. Josephson, “Possible new effects in superconductive tunnelling,” *Physics Letters*, vol. 1, no. 7, pp. 251 – 253, 1962. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/0031916362913690> (Cited on page 4).
- [29] B. D. Josephson, “The discovery of tunnelling supercurrents,” *Rev. Mod. Phys.*, vol. 46, pp. 251–254, Apr 1974. [Online]. Available: <http://link.aps.org/doi/10.1103/RevModPhys.46.251> (Cited on page 4).
- [30] Z. Bao, M. Bhushan, S. Ran, and J. Lukens, “Fabrication of high quality, deep-submicron Nb/AlOx/Nb Josephson junctions using chemical mechanical polishing,” *Applied Superconductivity, IEEE Transactions on*, vol. 5, no. 2, pp. 2731 – 2734, Jun. 1995. (Cited on page 5).
- [31] L. Lee, E. Arambula, G. Hanaya, C. Dang, R. Sandell, and H. Chan, “RHEA (resist-hardened etch and anodization) process for fine-geometry Josephson junction fabrication,” *Magnetics, IEEE Transactions on*, vol. 27, no. 2, pp. 3133 –3136, Mar. 1991. (Cited on page 5).

- [32] Q. Zhong, W. Cao, J. Li, Y. Zhong, and X. Wang, “Study of dry etching process using SF₆ and CF₄/O₂ for Nb/NbxSi₁/Nb Josephson-junction fabrication,” in *Precision Electromagnetic Measurements (CPEM), 2012 Conference on*, Jul. 2012, pp. 46–47. (Cited on page 5).
- [33] D. K. Brock, “RSFQ Technology: Circuits and Systems,” *Int. J. High Speed Electron. Syst.*, vol. 11, pp. 307–362, 2001. (Cited on page 5).
- [34] W. Chen, A. Rylyakov, V. Patel, J. Lukens, and K. Likharev, “Rapid single flux quantum T-flip flop operating up to 770 GHz,” *Applied Superconductivity, IEEE Transactions on*, vol. 9, no. 2, pp. 3212–3215, Jun. 1999. (Cited on pages 5 and 43).
- [35] K. Fujiwara, Q. Liu, T. Van Duzer, X. Meng, and N. Yoshikawa, “New Delay-Time Measurements on a 64-kb Josephson-CMOS Hybrid Memory With a 600-ps Access Time,” *Applied Superconductivity, IEEE Transactions on*, vol. 20, no. 1, pp. 14–20, Feb. 2010. (Cited on page 5).
- [36] Q. Liu, K. Fujiwara, X. Meng, S. Whiteley, T. Van Duzer, N. Yoshikawa, Y. Thakahashi, T. Hikida, and N. Kawai, “Latency and Power Measurements on a 64-kb Hybrid Josephson-CMOS Memory,” *Applied Superconductivity, IEEE Transactions on*, vol. 17, no. 2, pp. 526–529, Jun. 2007. (Cited on page 5).
- [37] Y. Akahori and K. Hohkawa, “A Josephson dual-phase ac-powered logic network using special latch circuits,” *Electron Devices, IEEE Transactions on*, vol. 32, no. 11, pp. 2339–2344, Nov. 1985. (Cited on page 7).
- [38] P. Arnett and D. Herrell, “Regulated AC power for Josephson interferometer latching logic circuits,” *Magnetics, IEEE Transactions on*, vol. 15, no. 1, pp. 554–557, Jan. 1979. (Cited on page 7).
- [39] S. Hasuo, “High-speed digital circuits for a Josephson computer,” in *Multiple-Valued Logic, 1992. Proceedings., Twenty-Second International Symposium on*, May 1992, pp. 2–8. (Cited on pages 8, 18, and 19).
- [40] V. Kaplunenko, M. Khabipov, V. Koshelets, K. Likharev, O. Mukhanov, V. Semenov, I. Serpuchenko, and A. Vystavkin, “Experimental study of the RSFQ logic elements,” *Magnetics, IEEE Transactions on*, vol. 25, no. 2, pp. 861–864, Mar. 1989. (Cited on pages 8 and 81).
- [41] K. Takagi, M. Tanaka, S. Iwasaki, R. Kasagi, I. Kataeva, S. Nagasawa, T. Satoh, H. Akaike, and A. Fujimaki, “SFQ Propagation Properties in

- Passive Transmission Lines Based on a 10-Nb-Layer Structure,” *Applied Superconductivity, IEEE Transactions on*, vol. 19, no. 3, pp. 617 –620, Jun 2009. (Cited on pages 8, 9, and 10).
- [42] S. Yorozu, Y. Kameda, H. Terai, A. Fujimaki, T. Yamada, and S. Tahara, “A single flux quantum standard logic cell library,” *Physica C: Superconductivity*, vol. 378-381, Part 2, no. 0, pp. 1471 – 1474, 2002. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0921453402017598> (Cited on pages 9 and 28).
- [43] H. Suzuki, S. Nagasawa, K. Miyahara, and Y. Enomoto, “Characteristics of driver and receiver circuits with a passive transmission line in RSFQ circuits,” *Applied Superconductivity, IEEE Transactions on*, vol. 10, no. 3, pp. 1637 –1641, Sep 2000. (Cited on page 8).
- [44] D. Gupta, W. Li, S. Kaplan, and I. Vernik, “High-speed interchip data transmission technology for superconducting multi-chip modules,” *Applied Superconductivity, IEEE Transactions on*, vol. 11, no. 1, pp. 731 –734, Mar 2001. (Cited on page 8).
- [45] O. Mukhanov, “Energy-Efficient Single Flux Quantum Technology,” *Applied Superconductivity, IEEE Transactions on*, vol. 21, no. 3, pp. 760 –769, Jun. 2011. (Cited on pages 10 and 12).
- [46] D. Kirichenko, S. Sarwana, and A. Kirichenko, “Zero Static Power Dissipation Biasing of RSFQ Circuits,” *Applied Superconductivity, IEEE Transactions on*, vol. 21, no. 3, pp. 776 –779, Jun. 2011. (Cited on pages 12 and 13).
- [47] K. Fujiwara, N. Nakajima, T. Nishigai, M. Ito, N. Yoshikawa, A. Fujimaki, H. Terai, and S. Yorozu, “Error rate test of large-scale SFQ digital circuit systems,” *Applied Superconductivity, IEEE Transactions on*, vol. 15, no. 2, pp. 427 – 430, Jun. 2005. (Cited on page 12).
- [48] H. Terai, Y. Hashimoto, S. Yorozu, A. Fujimaki, N. Yoshikawa, and Z. Wang, “The relationship between bit-error rate, operating speed and circuit scale of SFQ circuits,” *Applied Superconductivity, IEEE Transactions on*, vol. 15, no. 2, pp. 364 – 367, Jun. 2005. (Cited on page 12).
- [49] I. Kataeva, H. Akaike, A. Fujimaki, N. Yoshikawa, S. Nagasawa, and N. Takagi, “Clock Line Considerations for an SFQ Large Scale Reconfigurable Data Paths Processor,” *Applied Superconductivity, IEEE Trans-*

- actions on*, vol. 21, no. 3, pp. 809 –813, Jun. 2011. (Cited on page 12).
- [50] M. Dorojevets, C. Ayala, and A. Kasperek, “Data-Flow Microarchitecture for Wide Datapath RSFQ Processors: Design Study,” *Applied Superconductivity, IEEE Transactions on*, vol. 21, no. 3, pp. 787 –791, Jun. 2011. (Cited on pages 12, 13, 22, 25, and 37).
- [51] S. Narayana, Y. Polyakov, and V. Semenov, “Evaluation of Flux Trapping in Superconducting Circuits,” *Applied Superconductivity, IEEE Transactions on*, vol. 19, no. 3, pp. 640 –643, Jun 2009. (Cited on page 13).
- [52] K. Tanaka, T. Morooka, A. Odawara, Y. Mawatari, S. Nakayama, A. Nagata, K. Ikeda, K. Chinone, and M. Koyanagi, “Study of trapped flux in a superconducting thin film-observation by scanning SQUID microscope and simulation,” *Applied Superconductivity, IEEE Transactions on*, vol. 11, no. 1, pp. 230 –233, Mar 2001. (Cited on page 13).
- [53] S. Bermon and T. Gheewala, “Moat-guarded Josephson SQUIDS,” *Magnetics, IEEE Transactions on*, vol. 19, no. 3, pp. 1160 – 1164, may 1983. (Cited on page 13).
- [54] M. Jeffery, T. Van Duzer, J. R. Kirtley, and M. B. Ketchen, “Magnetic imaging of moat-guarded superconducting electronic circuits,” *Applied Physics Letters*, vol. 67, no. 12, pp. 1769 –1771, Sep 1995. (Cited on page 13).
- [55] K. Fujiwara, S. Nagasawa, Y. Hashimoto, M. Hidaka, N. Yoshikawa, M. Tanaka, H. Akaike, A. Fujimaki, K. Takagi, and N. Takagi, “Research on Effective Moat Configuration for Nb Multi-Layer Device Structure,” *Applied Superconductivity, IEEE Transactions on*, vol. 19, no. 3, pp. 603 –606, Jun 2009. (Cited on page 13).
- [56] S. Polonsky, P. Shevchenko, A. Kirichenko, D. Zinoviev, and A. Rylyakov, “PSCAN’96: new software for simulation and optimization of complex RSFQ circuits,” *Applied Superconductivity, IEEE Transactions on*, vol. 7, no. 2, pp. 2685 –2689, Jun 1997. (Cited on page 13).
- [57] S. Whiteley, “Josephson junctions in SPICE3,” *Magnetics, IEEE Transactions on*, vol. 27, no. 2, pp. 2902 –2905, Mar 1991. (Cited on page 13).

- [58] E. Fang and T. V. Duzer, “A Josephson integrated circuit simulator (JSIM) for superconductive electronics applications,” in *Extended abstracts of 1989 International Superconductivity Electronics Conference*, Tokyo, Jun. 1989, pp. 407–410. (Cited on pages 13 and 28).
- [59] H. Topfer, H. Uhlmann, M. Knoll, H. Thiele, and M. Selent, “Design tools for parameter determination and simulation of integrated Josephson structures,” *Applied Superconductivity, IEEE Transactions on*, vol. 5, no. 2, pp. 3345 – 3348, Jun 1995. (Cited on page 13).
- [60] J. Satchell, “Stochastic simulation of SFQ logic,” *Applied Superconductivity, IEEE Transactions on*, vol. 7, no. 2, pp. 3315 –3318, Jun 1997. (Cited on page 13).
- [61] A. Krasniewski, “Logic simulation of RSFQ circuits,” *Applied Superconductivity, IEEE Transactions on*, vol. 3, no. 1, pp. 33 –38, Mar 1993. (Cited on page 13).
- [62] V. Adler, C.-H. Cheah, K. Gaj, D. Brock, and E. Friedman, “A Cadence-based design environment for single flux quantum circuits,” *Applied Superconductivity, IEEE Transactions on*, vol. 7, no. 2, pp. 3294 –3297, Jun 1997. (Cited on page 13).
- [63] K. Gaj, C.-H. Cheah, E. Friedman, and M. Feldman, “Functional modeling of RSFQ circuits using Verilog HDL,” *Applied Superconductivity, IEEE Transactions on*, vol. 7, no. 2, pp. 3151 –3154, Jun 1997. (Cited on page 13).
- [64] H. Akaike, M. Tanaka, K. Takagi, I. Kataeva, R. Kasagi, A. Fujimaki, K. Takagi, M. Igarashi, H. Park, Y. Yamanashi, N. Yoshikawa, K. Fujiwara, S. Nagasawa, M. Hidaka, and N. Takagi, “Design of single flux quantum cells for a 10-Nb-layer process,” *Physica C: Superconductivity*, vol. 469, no. 15-20, pp. 1670 – 1673, 2009, proceedings of the 21st International Symposium on Superconductivity (ISS 2008), Proceedings of the 21st International Symposium on Superconductivity (ISS 2008). [Online]. Available: <http://www.sciencedirect.com/science/article/B6TVJ-4WDNKSJ-28/2/cd4f343102a2c6d5b73e16d46dd31b0d> (Cited on pages 13, 28, and 30).
- [65] NioPulse. NioCAD. [Online]. Available: <http://niocad.com/> (Cited on page 13).
- [66] S. Anders, M. Blamire, F.-I. Buchholz, D.-G. Crete, R. Cristiano, P. Febvre, L. Fritsch, A. Herr, E. Il’ichev, J. Kohlmann, J. Kunert,

- H.-G. Meyer, J. Niemeyer, T. Ortlepp, H. Rogalla, T. Schurig, M. Siegel, R. Stolz, E. Tarte, H. ter Brake, H. Toepfer, J.-C. Villegier, A. Zagorskin, and A. Zorin, “European roadmap on superconductive electronics - status and perspectives,” *Physica C: Superconductivity*, vol. 470, no. 23-24, pp. 2079 – 2126, 2010, <ce:title>European Roadmap on Superconductor Electronics - Status and Perspectives</ce:title>. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0921453410005332> (Cited on page 13).
- [67] H. Jin, K. Kuwabara, Y. Yamanashi, and N. Yoshikaw, “Investigation of Robust CMOS Amplifiers for Josephson-CMOS Hybrid Memories,” *Physics Procedia*, vol. 36, no. 0, pp. 229 – 234, 2012, <ce:title>SUPERCONDUCTIVITY CENTENNIAL Conference 2011</ce:title>. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S1875389212020883> (Cited on page 13).
- [68] O. Mukhanov, A. Kirichenko, T. Filippov, and S. Sarwana, “Hybrid Semiconductor-Superconductor Fast-Readout Memory for Digital RF Receivers,” *Applied Superconductivity, IEEE Transactions on*, vol. 21, no. 3, pp. 797 –800, Jun. 2011. (Cited on page 13).
- [69] T. I. Larkin, V. V. Bolginov, V. S. Stolyarov, V. V. Ryazanov, I. V. Vernik, S. K. Tolpygo, and O. A. Mukhanov, “Ferromagnetic Josephson switching device with high characteristic voltage,” *Applied Physics Letters*, vol. 100, no. 22, pp. 222 601 –222 601–5, may 2012. (Cited on page 13).
- [70] V. V. Ryazanov, V. V. Bol’ginov, D. S. Sobanin, I. V. Vernik, S. K. Tolpygo, A. M. Kadin, and O. A. Mukhanov, “Magnetic Josephson Junction Technology for Digital and Memory Applications,” *Physics Procedia*, vol. 36, no. 0, pp. 35 – 41, 2012, <ce:title>SUPERCONDUCTIVITY CENTENNIAL Conference 2011</ce:title>. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S1875389212020639> (Cited on page 13).
- [71] K. K. Likharev, “Superconductor digital electronics,” *Physica C: Superconductivity*, vol. 482, no. 0, pp. 6 – 18, 2012, <ce:title>2011 Centennial superconductivity conference EUCAS-ISEC-ICMC</ce:title>. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0921453412002481> (Cited on page 13).
- [72] A. Herr, O. Naaman, D. Miller, and Q. Herr, “Josephson Magnetic Random Access Memory (JMRAM),” *Applied Superconductivity, IEEE*

- Transactions on*, vol. xx, p. xx, Oct. 2012, submitted. (Cited on page 13).
- [73] K. Hinago, Y. Yamanashi, and N. Yoshikawa, “Design and component development of DC-powered single-flux-quantum random-access memories using vortex transition memory cells,” *Applied Superconductivity, IEEE Transactions on*, vol. xx, p. xx, Oct. 2012, submitted. (Cited on page 14).
- [74] S. Nagasawa, H. Hasegawa, T. Hashimoto, H. Suzuki, K. Miyahara, and Y. Enomoto, “Superconducting latching/SFQ hybrid RAM,” *Applied Superconductivity, IEEE Transactions on*, vol. 11, no. 1, pp. 533–536, Mar. 2001. (Cited on page 14).
- [75] H. Numata, S. Nagasawa, and S. Tahara, “A vortex transitional memory cell for 1-Mbit/cm² density Josephson RAMs,” *Applied Superconductivity, IEEE Transactions on*, vol. 7, no. 2, pp. 2282–2287, Jun. 1997. (Cited on page 14).
- [76] A. Kirichenko and O. Mukhanov, “Implementation of novel “push-forward” RSFQ Carry-Save Serial Adders,” *Applied Superconductivity, IEEE Transactions on*, vol. 5, no. 2, pp. 3010–3013, Jun. 1995. (Cited on page 14).
- [77] P. Bunyk and P. Litskevitch, “Case study in RSFQ design: fast pipelined parallel adder,” *Applied Superconductivity, IEEE Transactions on*, vol. 9, no. 2, pp. 3714–3720, Jun. 1999. (Cited on pages 15 and 37).
- [78] K. Takahashi, S. Nagasawa, H. Hasegawa, K. Miyahara, H. Takai, and Y. Enomoto, “Design of a superconducting ALU with a 3-input XOR gate,” *Applied Superconductivity, IEEE Transactions on*, vol. 13, no. 2, pp. 551–554, Jun. 2003. (Cited on page 15).
- [79] J.-Y. Kim, S. Kim, and J. Kang, “Construction of an RSFQ 4-bit ALU with half adder cells,” *Applied Superconductivity, IEEE Transactions on*, vol. 15, no. 2, pp. 308–311, Jun. 2005. (Cited on page 16).
- [80] H. Park, Y. Yamanashi, N. Yoshikawa, M. Tanaka, and A. Fujimaki, “Design of fast digit-serial adders using SFQ logic circuits,” *IEICE Electronics Express*, vol. 6, no. 19, pp. 1408–1413, 2009. (Cited on pages 16 and 17).

- [81] M. Tanaka, H. Akaike, A. Fujimaki, Y. Yamanashi, N. Yoshikawa, S. Nagasawa, K. Takagi, and N. Takagi, “100-GHz Single-Flux-Quantum Bit-Serial Adder Based on 10-Niobium Process,” *Applied Superconductivity, IEEE Transactions on*, vol. 21, no. 3, pp. 792–796, Jun. 2011. (Cited on pages 17 and 18).
- [82] S. Kotani, A. Inoue, T. Imamura, and S. Hasuo, “An 8-b Josephson digital signal processor,” *Solid-State Circuits, IEEE Journal of*, vol. 25, no. 6, pp. 1518–1525, Dec. 1990. (Cited on page 18).
- [83] S. Kotani, N. Fujimaki, T. Imamura, and S. Hasuo, “A subnanosecond Josephson 16-bit ALU,” *Solid-State Circuits, IEEE Journal of*, vol. 23, no. 2, pp. 591–596, Apr. 1988. (Cited on page 18).
- [84] M. Dorojevets and P. Bunyk, “Architectural and implementation challenges in designing high-performance RSFQ processors: a FLUX-1 microprocessor and beyond,” *Applied Superconductivity, IEEE Transactions on*, vol. 13, no. 2, pp. 446–449, Jun. 2003. (Cited on pages 19 and 20).
- [85] P. Bunyk, M. Leung, J. Spargo, and M. Dorojevets, “Flux-1 RSFQ microprocessor: physical design and test results,” *Applied Superconductivity, IEEE Transactions on*, vol. 13, no. 2, pp. 433–436, Jun. 2003. (Cited on page 19).
- [86] M. Dorojevets, P. Bunyk, and D. Zinoviev, “FLUX chip: design of a 20-GHz 16-bit ultrapipelined RSFQ processor prototype based on 1.75- μm LTS technology,” *Applied Superconductivity, IEEE Transactions on*, vol. 11, no. 1, pp. 326–332, Mar. 2001. (Cited on page 19).
- [87] M. Dorojevets, “A 20-GHz FLUX-1 superconductor RSFQ microprocessor,” in *Low Temperature Electronics, 2002. Proceedings of the 5th European Workshop on*, 2002, pp. 157–160. (Cited on page 19).
- [88] M. Tanaka, F. Matsuzaki, T. Kondo, N. Nakajima, Y. Yamanashi, A. Fujimaki, H. Hayakawa, N. Yoshikawa, H. Terai, and S. Yorozu, “A single-flux-quantum logic prototype microprocessor,” in *Solid-State Circuits Conference, 2004. Digest of Technical Papers. ISSCC. 2004 IEEE International*, Feb. 2004, pp. 298–529 Vol.1. (Cited on page 21).
- [89] Y. Yamanashi, M. Tanaka, A. Akimoto, H. Park, Y. Kamiya, N. Irie, N. Yoshikawa, A. Fujimaki, H. Terai, and Y. Hashimoto, “Design and Implementation of a Pipelined Bit-Serial SFQ Microprocessor,

- CORE1B,” *Applied Superconductivity, IEEE Transactions on*, vol. 17, no. 2, pp. 474–477, Jun. 2007. (Cited on page 21).
- [90] A. Fujimaki, M. Tanaka, T. Yamada, Y. Yamanashi, H. Park, and N. Yoshikawa, “Bit-Serial Single Flux Quantum Microprocessor CORE,” *IEICE Transactions on Electronics*, vol. E91-C, pp. 342–349, Mar. 2008. (Cited on page 21).
- [91] M. Tanaka, Y. Yamanashi, N. Irie, H.-J. Park, S. Iwasaki, K. Takagi, K. Taketomi, A. Fujimaki, N. Yoshikawa, H. Terai, and S. Yorozu, “Design and implementation of a pipelined 8 bit-serial single-flux-quantum microprocessor with cache memories,” *Superconductor Science and Technology*, vol. 20, no. 11, p. S305, 2007. [Online]. Available: <http://stacks.iop.org/0953-2048/20/i=11/a=S01> (Cited on pages 21 and 22).
- [92] T. Filippov, M. Dorojevets, A. Sahu, A. Kirichenko, C. Ayala, and O. Mukhanov, “8-Bit Asynchronous Wave-Pipelined RSFQ Arithmetic-Logic Unit,” *Applied Superconductivity, IEEE Transactions on*, vol. 21, no. 3, pp. 847–851, Jun. 2011. (Cited on pages 22, 27, 77, 79, and 80).
- [93] T. V. Filippov, A. Sahu, A. F. Kirichenko, I. V. Vernik, M. Dorojevets, C. L. Ayala, and O. A. Mukhanov, “20 GHz Operation of an Asynchronous Wave-Pipelined RSFQ Arithmetic-Logic Unit,” *Physics Procedia*, vol. 36, no. 0, pp. 59–65, 2012, <ce:title>SUPERCONDUCTIVITY CENTENNIAL Conference 2011</ce:title>. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S1875389212020676> (Cited on pages 22, 27, 81, 82, 83, 84, 85, and 86).
- [94] A. F. Kirichenko, T. V. Filippov, A. Sahu, O. A. Mukhanov, M. Dorojevets, and A. K. Kasperek, “Demonstration of RSFQ 8-bit multi-port register file.” in *Proceedings of Applied Superconductivity Conference 2012 (ASC '12)*, Portland, OR., Oct. 2012. (Cited on pages 22 and 27).
- [95] (2008, Jan.) Hypres Niobium Integrated Circuit Fabrication - Design Rules. 175 Clearbrook Road, Elmsford, NY 10523, USA. [Online]. Available: <http://hypres.accountsupport.com/wp-content/uploads/2010/11/DesignRules.pdf> (Cited on pages 26 and 27).
- [96] E. Fang and T. Van Duzer, “An efficient method for finding DC solutions for Josephson circuits,” *Applied Superconductivity, IEEE Transactions on*, vol. 1, no. 3, pp. 126–133, Sep. 1991. (Cited on page 28).

- [97] T. Satoh, K. Hinode, S. Nagasawa, Y. Kitagawa, M. Hidaka, N. Yoshikawa, H. Akaike, A. Fujimaki, K. Takagi, and N. Takagi, “Planarization Process for Fabricating Multi-Layer Nb Integrated Circuits Incorporating Top Active Layer,” *Applied Superconductivity, IEEE Transactions on*, vol. 19, no. 3, pp. 167–170, Jun. 2009. (Cited on pages 28 and 29).
- [98] M. Anbuselvi, S. Salivahanan, and P. Saravanan, “Design and Analysis of Floating Point and Galois Field Multipliers Using Wave-Pipelining,” in *Advances in Computing, Control, Telecommunication Technologies, 2009. ACT '09. International Conference on*, Dec. 2009, pp. 602–604. (Cited on page 35).
- [99] W. Burleson, M. Ciesielski, F. Klass, and W. Liu, “Wave-pipelining: a tutorial and research survey,” *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 6, no. 3, pp. 464–474, Sep. 1998. (Cited on page 35).
- [100] W. Burleston, L. Cotton, F. Klaus, and M. Ciesielski, “Wave-pipelining: is it practical?” in *Circuits and Systems, 1994. ISCAS '94., 1994 IEEE International Symposium on*, vol. 4, Jun. 1994, pp. 163–166 vol.4. (Cited on page 35).
- [101] G. Enrique Fernandez and R. Sridhar, “Dual rail static CMOS architecture for wave pipelining,” in *VLSI Design, 1996. Proceedings., Ninth International Conference on*, Jan. 1996, pp. 335–336. (Cited on page 35).
- [102] J. Levy, J. Nyathi, and J. Delgado-Frias, “High-performance parallel addition using hybrid wave-pipelining,” in *Circuits and Systems, 2005. 48th Midwest Symposium on*, Aug. 2005, pp. 555–558 Vol. 1. (Cited on page 35).
- [103] M. Litvin, S. Mourad, W. Terry, and J. Terry, “Wave Pipelining using Self Reset Logic,” in *Electronics, Circuits and Systems, 2006. ICECS '06. 13th IEEE International Conference on*, Dec. 2006, pp. 1280–1283. (Cited on page 35).
- [104] R. Sever and M. Askar, “8x8-Bit multiplier designed with a new wave-pipelining scheme,” in *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, Jun. 2010, pp. 2095–2098. (Cited on page 35).

- [105] D. Wong, G. De Micheli, and M. Flynn, "Inserting active delay elements to achieve wave pipelining," in *Computer-Aided Design, 1989. ICCAD-89. Digest of Technical Papers., 1989 IEEE International Conference on*, Nov. 1989, pp. 270–273. (Cited on page 35).
- [106] D. E. Muller and W. S. Bartky, "A Theory of Asynchronous Circuits," in *Proc. Int'l Symp. Theory of Switching, Part 1*. Harvard Univ. Press, 1959, pp. 204–243. (Cited on page 37).
- [107] M. Dorojevets and C. Ayala, "Logical design and analysis of a 32/64-bit wave-pipelined RSFQ adder," in *Proceedings of 2nd Superconducting SFQ VLSI Workshop*, ser. O6, Fukuoka, Japan, Aug. 2009, pp. 15–16. (Cited on page 37).
- [108] M. Dorojevets, C. Ayala, and A. Kasperek, "Development and evaluation of design techniques for high-performance wave-pipelined wide datapath RSFQ processors," in *Proceedings of 12th International Superconductive Electronics Conference*, Fukuoka, Japan, Aug. 2009, pp. SP–P46. (Cited on page 37).
- [109] K. Gaj, E. G. Friedman, and M. J. Feldman, "Timing of Multi-Gigahertz Rapid Single Flux Quantum Digital Circuits," *J. VLSI Signal Process. Syst.*, vol. 16, no. 2/3, pp. 247–276, Jul. 1997. [Online]. Available: <http://dx.doi.org/10.1023/A:1007903527533> (Cited on page 37).
- [110] M. Dorojevets, A. K. Kasperek, N. Yoshikawa, and A. Fujimaki, "8x8-bit parallel carry-save superconductor RSFQ multiplier," *Applied Superconductivity, IEEE Transactions on*, vol. 23, p. xx, Jun. 2013, accepted. (Cited on pages 47, 132, and 134).
- [111] P. M. Kogge and H. S. Stone, "A Parallel Algorithm for the Efficient Solution of a General Class of Recurrence Equations," *Computers, IEEE Transactions on*, vol. C-22, no. 8, pp. 786–793, Aug. 1973. (Cited on page 54).
- [112] D. Zinoviev and Y. Polyakov, "Octopux: an advanced automated setup for testing superconductor circuits," *Applied Superconductivity, IEEE Transactions on*, vol. 7, no. 2, pp. 3240–3243, Jun. 1997. (Cited on page 78).
- [113] S. Mathew, M. Anders, R. Krishnamurthy, and S. Borkar, "A 6.5GHz 54mW 64-bit Parity-Checking Adder for 65nm Fault-Tolerant Micro-processor Execution Cores," in *VLSI Circuits, 2007 IEEE Symposium on*, Jun. 2007, pp. 46–47. (Cited on page 88).

- [114] S. Wijeratne, N. Siddaiah, S. Mathew, M. Anders, R. Krishnamurthy, J. Anderson, S. Hwang, M. Ernest, and M. Nardin, “A 9GHz 65nm Intel Pentium 4 Processor Integer Execution Core,” in *Solid-State Circuits Conference, 2006. ISSCC 2006. Digest of Technical Papers. IEEE International*, Feb. 2006, pp. 353–365. (Cited on pages 101, 105, and 133).
- [115] M. Peiniger and H. Piel, “A Superconducting Nb₃Sn Coated Multicell Accelerating Cavity,” *Nuclear Science, IEEE Transactions on*, vol. 32, no. 5, pp. 3610–3612, Oct. 1985. (Cited on page 117).
- [116] M. Hidaka, S. Nagasawa, K. Hinode, and T. Satoh, “Device yield in Nb-nine-layer circuit fabrication process,” *Applied Superconductivity, IEEE Transactions on*, vol. xx, no. xx, p. xx, Oct. 2012, submitted. (Cited on page 123).
- [117] Q. P. Herr, A. Y. Herr, O. T. Oberg, and A. G. Ioannidis, “Ultra-Low-Power Superconductor Logic,” *arXiv.org*, vol. 21240, p. 7, Mar. 2011. [Online]. Available: <http://arxiv.org/abs/1103.4269> (Cited on page 134).
- [118] Q. Herr, “Carry look-ahead adder implemented in Reciprocal Quantum Logic,” *Applied Superconductivity, IEEE Transactions on*, vol. xx, p. xx, Oct. 2012, submitted. (Cited on page 134).
- [119] O. Oberg, Q. Herr, A. Ioannidis, and A. Herr, “Integrated Power Divider for Superconducting Digital Circuits,” *Applied Superconductivity, IEEE Transactions on*, vol. 21, no. 3, pp. 571–574, Jun. 2011. (Cited on page 134).
- [120] N. Takeuchi, K. Ehara, K. Inoue, Y. Yamanashi, and N. Yoshikawa, “Margins and energy dissipation of adiabatic quantum-flux parametron logic at finite temperature,” *Applied Superconductivity, IEEE Transactions on*, vol. xx, p. xx, Oct. 2012, submitted. (Cited on page 134).
- [121] K. Inoue, K. Ehara, N. Takeuchi, Y. Yamanashi, and N. Yoshikawa, “Simulation and experimental demonstration of logic circuits using ultra-low-power adiabatic quantum-flux-parametron,” *Applied Superconductivity, IEEE Transactions on*, vol. xx, p. xx, Oct. 2012, submitted. (Cited on page 134).