博士論文

大規模断熱量子磁束パラメトロン回路設計法の研究

Study on the design methodology of large-scale adiabatic quantum-flux-parametron circuits

指導教官　吉川　信行 教授

2023 年 3 月 10 日

横浜国立大学 大学院理工学府　数物・電子情報系理工学専攻

20QC502

田中　智之

Tomoyuki Tanaka

Abstract

The amount of data handled by humankind is increasing every year. The data center which provides services such as Internet commerce and video streaming services has a market share that is expected to increase by USD 615.96 billion from 2020 to 2025, and the market's growth momentum will accelerate at a compound annual growth rate (CAGR) of 21.98% [1]. This growth means that data centers will deploy many more computers, and one projection predicts that data centers will consume 8% of the earth's electricity in 2030 [2]. The information society faces the problem of power consumption.

CMOS circuits used in computers as arithmetic circuits have a problem with miniaturization. For example, TSMC, a leading foundry for logic circuits, has announced that it will begin mass production of N3 (3 nm) technology in the second half of 2022. However, given that the diameter of an atom is about 0.1 nm, it is clear that further miniaturization will be even more difficult. Computers with low power consumption and high performance are needed in the future. We have been working on the realization of next-generation computers, focusing on adiabatic quantum-flux-parametron (AQFP) circuits. A superconductor circuit is a device based on Josephson junctions, which can operate faster and consume much less power than CMOS circuits[3]. The AQFP circuit is an even more energy-efficient logic device among superconductor circuits, consuming 1/100,000 of the energy of CMOS[4, 5].

There are many hurdles to overcome one by one in order to put AQFP circuits into practical use. This thesis improves the design environment for AQFP circuits by addressing the application of variable-length wiring and methods to achieve the same level of the environment in AQFP circuits as in CMOS circuit design.

Contents

Chapter 1

Intrduction

1.1 Limitation of CMOS based computer

Computers are the infrastructure of human life. Computers control everything and anything, and they have become so important that many people might lose their lives if they suddenly become unable to use computers. Since the invention of the transistor, complementary metal oxide semiconductor-based computers (CMOS) have experienced explosive growth and have contributed significantly to the advancement of human society.

CMOS will soon face a miniaturization problem. The most advanced process rules have reached around 3 nm in 2022, and the leading research laboratory of semiconductors; Interuniversity Microelectronics Centre(imec) claims it will be possible to get 0.2 nm by 2040[6]. Since the atomic radius of silicon and metals used in wiring is about 0.1 nm, the process rule cannot be smaller than 0.1 nm due to physical limitation. This limitation means that semiconductor growth will stagnate in 2040, after which computers may be left to structures that are not CMOS. In other words, near 2040, we will face a problem with the stagnation of providing calculation resources.

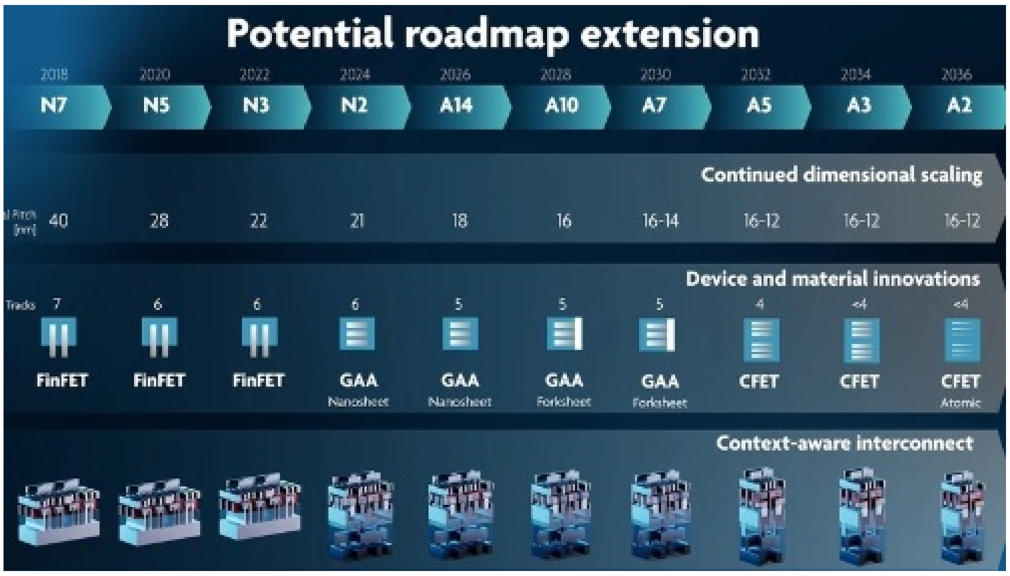

Potential roadmap extension
2018 N7
2020 N5
2022 N3
2024 N2
2026 A14
2028 A10
2030 A7
2032 A5
2034 A3
2036 A2
Continued dimensional scaling
Pitch (nm) 40 28 22 21 18 16 16-14 16-12 16-12 16-12
Device and material innovations
Tracks 7 6 6 6 5 5 5 4 <4 <4
FinFET FinFET FinFET GAA Nanosheet GAA Nanosheet GAA Forksheet GAA Forksheet CFET CFET CFET Atomic
Context-aware interconnect

Figure 1.1: Loadmap of miniaturization of CMOS transistor[6]. They said the process rule of state-of-the-art CMOS will reach 0.2 nm in the 2040s.

1.2 Increased demand for computing resources

Think about the projections for the demand for computing resources: the metaverse with virtual reality and mixed reality, the Internet of things where various devices are connected to the Internet, on-demand video streaming such as Youtube and Netflix, etc. It is not difficult to believe that the demand for computing is only going to increase. Indeed, according to an article reported in Nature in 2018, networks and computers are expected to use 20% of the electricity consumed on the planet in the 2030s, and the increase in computers is considered one of the major obstacles to decarbonization, which is the current trend[2].

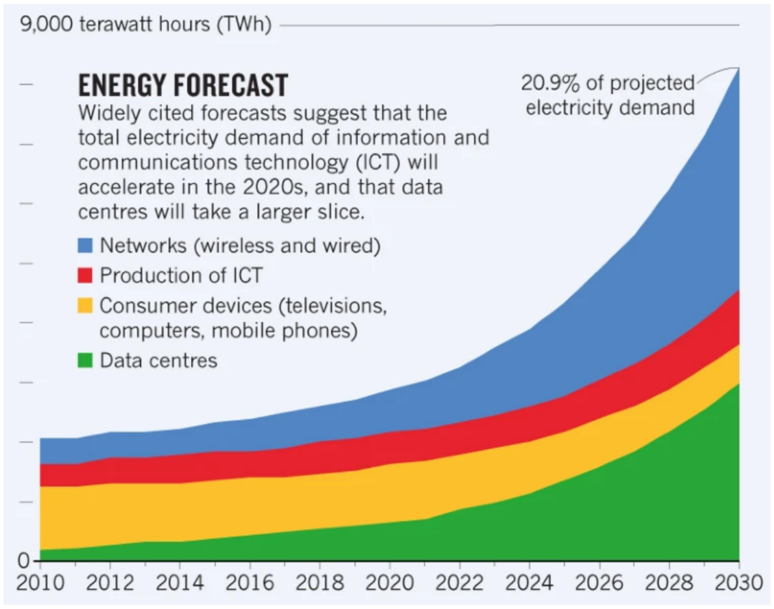


Figure 1.2: Projected power consumption by information and communication devices, which around 2022 was about 10% of total power consumption, but is expected to exceed 20% around 2030[2]. In particular, the infrastructure side, namely networks and data centers, will increase dramatically.

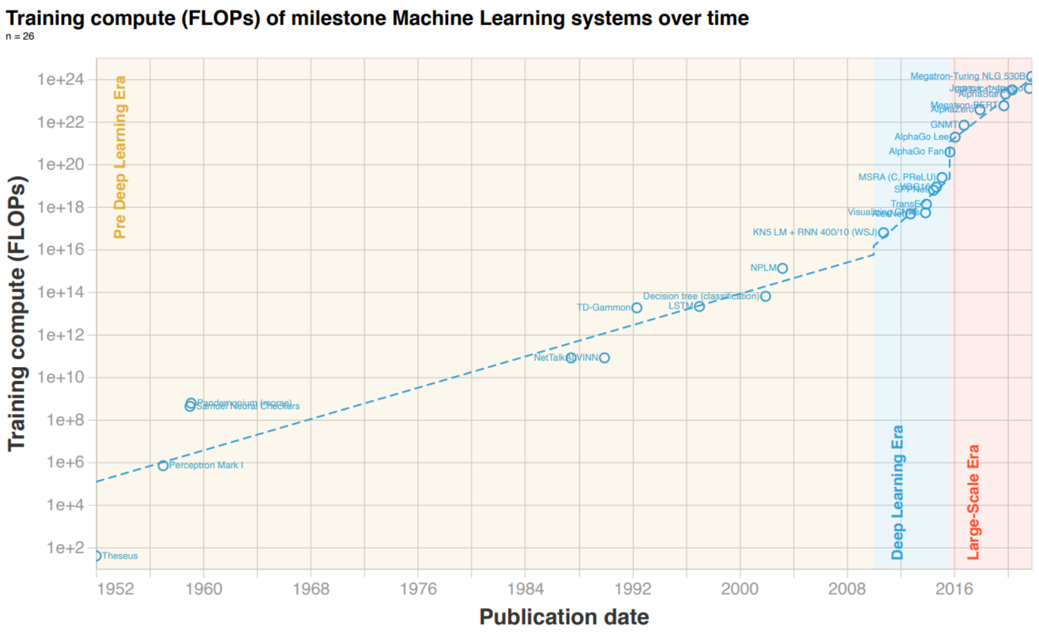


Figure 1.3: A graph showing the increase in learning models, which had been gradually increasing since around 1950, but increased dramatically after 2010. No sign of stagnation in the increase can be observed[7].

Further progress is also needed in terms of computer quality. One application that uses large amounts of computing resources is machine learning. Examples of applications include automated driving and chatbots. In particular, chatbots have made significant progress in recent years, but they are still in their infancy[8]. In order to improve performance, it is necessary to expand learning models. Already the pace of growth of learning models is extremely fast, doubling in size every six months since 2010, and there is no indication yet that this trend will slow down. There is a reason why the field of machine learning requires high-performance computers[7].

1.3 Superconductor post-CMOS computing

Superconductor circuits are a typical example of Post-CMOS circuits. It uses the Josephson junction as an active element for arithmetic operations and has excellent high-speed operation and low power consumption.

1.3.1 Single flux quantum circuit

Single flux quantum (especially RSFQ) circuits are digital devices with excellent high-speed operation, and have been successfully demonstrated in 4-bit microprocessors. In addition, 8-bit microprocessor designs and component operation demonstrations are underway, mainly at Nagoya University and Kyushu University, and a complete 8-bit microprocessor is nearing completion[9, 10]. The operating clock of this processor is 50 GHz, and power consumption is estimated to be 10^{13} ops/W. The operation of a 32-bit arithmetic unit has been verified in simulations, and steady progress is being made toward practical application[11].

If the RSFQ circuit can be put to practical use, it would raise the clock frequency of computers by a factor of 10. This would be a great boon for applications where real-time performance is important, such as weather simulation for weather forecasting.

Unfortunately, however, even if RSFQ circuits are commercialized, decarbonization will not progress.

Superconducting circuits operate at cryogenic temperatures of around 4K, which means that the chips must be constantly cooled. Cooling from room temperature to operating temperature 300 K to 4.2 K requires an extra power consumption of around 400 W to 5000 W[3]. To make a fair comparison with a conventional computer operating at room temperature, this power needs to be added to the power consumption of the circuit.

The Green 500 is a contest for the lowest power consumption of supercomputers, and the winner of the Green 500 announced in the fall of 2022 was a supercomputer called "Henri" installed at the Flatiron Institute in the U.S., with the energy efficiency is 65.091 GFlops/W. The results already show that conventional computers are more energy efficient, although this is not a strict comparison because of the different bit widths and architectures[12].

SFQ circuits also have various innovations to reduce power consumption, but they have problems such as lowering the operating frequency, and SFQ circuits are not a silver bullet that can solve all the problems that computers face[13, 14].

1.3.2 Adiabatic quantum-flux-parametron circuit

In order to balance the problem of increasing the amount of information handled by mankind with power savings, we need digital devices that can operate moderately faster than CMOS circuits and consume three or more orders of magnitude less power than that one. Adiabatic quantum-flux-parametron (AQFP) circuits are an alternative circuit technology to CMOS circuits. AQFP circuits are superconductor circuits that operate at5 GHz to 10 GHz and consume 5 to 6 orders of magnitude less power than CMOS circuits at the same operating frequency[5].

Previously, the AQFP, a 4-bit microprocessor, MANA, was demonstrated in operation[15]. This processor was designed to operate at 5 GHz and consumed 3.2×10^{13} ops/W. To compare this circuit to the RSFQ circuit, it is extended to 8-bit width. Even if the power consumption were increased by a factor of 8 in the pessimistic scenario, the power consumption would still be 4×10^{12} ops/W. The CMOS, RSFQ, and AQFP circuits are compared in table 1.1. Even with rough reasoning, the AQFP circuit can consume up to 100 times less power than the CMOS circuit. In addition, AQFP can operate at up to 10 GHz, which is slow compared to RSFQ circuits, but fast enough compared to current CMOS circuits. In other words, if a computer can be realized using AQFP circuits, the two immediate problems of insufficient computing resources and increased power consumption can be solved.

Table 1.1: Comparison of the architecture of digital device

	Clock frequency	Efficiency at 300 K	The architecture
CMOS[12]	5 GHz	6.5×10^{10} Flops/W	64-bit
RSFQ[9]	50 GHz	10^{10} ops/W	soon 8-bit
AQFP [15]	5 GHz (max. 10 GHz)	4×10^{12} ops/W	4-bit

Basic research on how to drive AQFP circuits and investigation of power consumption has progressed to date, and the next step is to work toward more practical applications. Practical circuits will need to be 10 or 100 times larger than the circuits designed to date, requiring ingenuity in circuit design and software development.

1.3.3 Purpose of this research

The current state of research on AQFP circuits is limited to a small scale compared to RSFQ circuits. This is due to the fact that most of the circuit design needs to be done manually, which requires a long time to design, and because of this, it is difficult to optimize the circuit structure. There have been several studies to improve the efficiency of designing AQFP circuits, but the main focus is on logic synthesis to optimize the graph of the circuit. While this is an important task because graph optimization reduces the area and latency of the circuit, it is more important to generate the physical circuit. Therefore, this study addressed both the routing and placement of AQFP circuits.

1.4 Structure of this paper

This paper consists of three major parts. The first part explains the basic theory of superconducting circuits and the principle of operation of AQFP circuits. Next, it describes the design flow of the AQFP circuit, and finally, it describes the actual circuit's design and measurement results. Chapter 2 explains the basic theory of AQFP and how to design the circuit. Chapter 4 explains to alleviate the limitation of the data propagation length limit of the AQFP circuit by introducing variable wiring height. Chapter 3 describes the automatic design of circuits similar to CMOS circuits to improve the design capability of AQFP circuits dramatically.Chapter 5 reports on the design and operational demonstration of AQFP integer and floating-point adders using the methods proposed in chapters 3 and 4 for designing large AQFP circuits. Chapter 6 reports a summary of the contents of this thesis.

Chapter 2

Theory of adiabatic quantum-flux-parametron

2.1 Introduction

This chapter first provides an overview of adiabatic quantum-flux-parametron (AQFP) circuits and their principle of operation. Then, how the AQFP circuit defines the binary information of '0' and '1' is explained, and the circuit design procedure is described.

Goto et al. at the University of Tokyo developed the quantum-flux-parametron circuit (QFP) in 1985. It can operate at several GHz and has one of the lowest power consumption among superconductor circuits. They have demonstrated the operation of an analog-to-digital converter operating at 18 GHz, high-speed operation at 36 GHz, and the design and operational implementation of an arithmetic logic unit (ALU) [16, 17, 18].

We are investigating adiabatic quantum-flux-parametron (AQFP) circuits, which operate adiabatically in QFP circuits. AQFP circuits have no static power consumption because they are driven by AC flux bias, and their dynamic power consumption is minimized because the potential of the circuit changes adiabatically.

Figure 2.1 shows the schematic of the AQFP circuit, in which the Josephson junction of the RF-SQUID is replaced by a DC-SQUID. The circuit parameters are symmetrical. In addition, since there are no resistors in the circuit, static power consumption is not generated.

2.2 Potential energy of AQFP circuit

In this section, the potential energy of the QFP circuit is explained. figure 2.1 shows the circuit diagram of a QFP circuit, and assuming these relationships $L_{\mathrm{x1}} = L_{\mathrm{x2}} = L_{\mathrm{x}}, L_1 = L_2 = L, k_1 = k_2, J_1 = J_2$ that the QFP circuit has a symmetrical structure, euqation (2.1) shows the potential energy[19, 20].

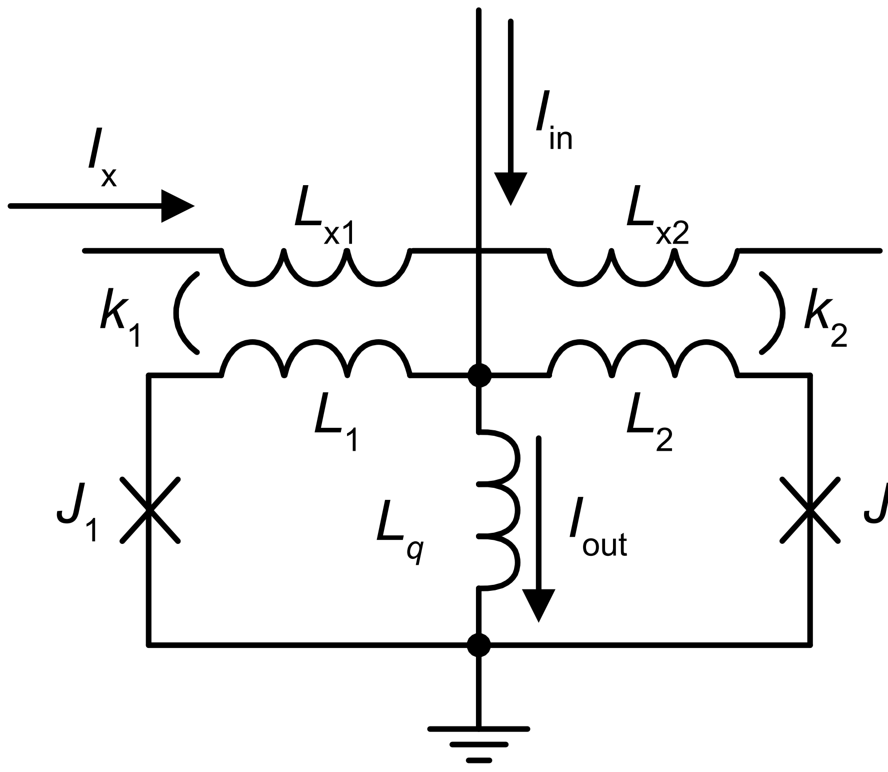


Figure 2.1: Schematic of AQFP circuit. It consists of two superconductive loops with Josephson junctions (x mark in the figure). In addition, unlike RSFQ circuits, for example, there is no static power consumption because the circuit has no resistance.

$$U = E\left[\frac{(\phi_\mathrm{x}-\phi_-)^2}{\beta_\mathrm{L}} + \frac{(\phi_\mathrm{in}-\phi_+)^2}{\beta_\mathrm{L}}2\beta_\mathrm{q} - 2\cos\phi_-\cos\phi_+\right] \tag{2.1}$$

$$E = \frac{I_\mathrm{c}\Phi_0}{2\pi} \tag{2.2}$$

$$\phi_\mathrm{x} = 2\pi\frac{k\sqrt{LL_\mathrm{x}}I_\mathrm{x}}{\Phi_0} \tag{2.3}$$

$$\phi_\mathrm{in} = 2\pi\frac{L_\mathrm{x}L_\mathrm{in}}{\Phi_0} \tag{2.4}$$

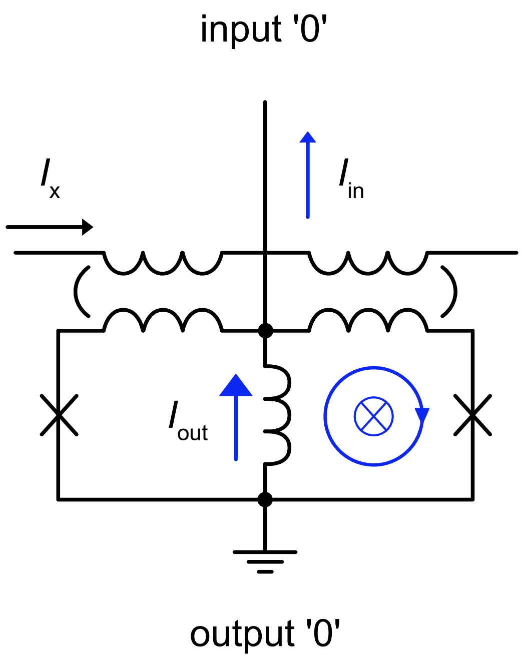
$$\phi_+ = \frac{\phi_1+\phi_2}{2} \tag{2.5}$$

$$\phi_- = \frac{\phi_1-\phi_2}{2} \tag{2.6}$$

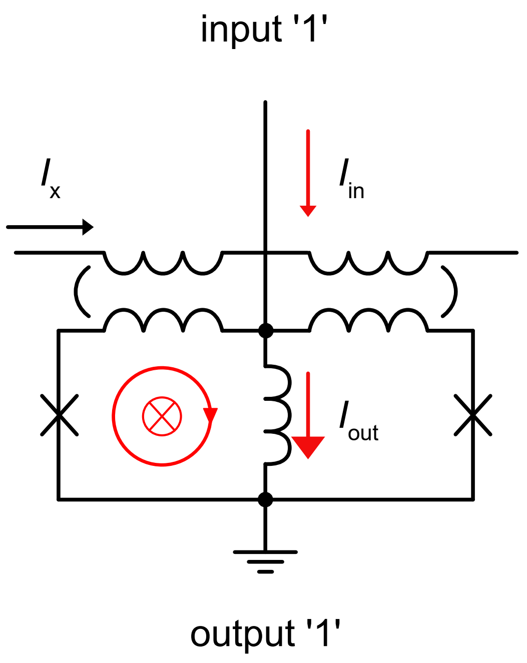
$$\beta_\mathrm{L} = 2\pi\frac{LI_\mathrm{c}}{\Phi_0} \tag{2.7}$$

$$\beta_\mathrm{q} = 2\pi\frac{L_\mathrm{q}I_\mathrm{c}}{\Phi_0} \tag{2.8}$$

I_c is the critical current value of J_1, J_2, and ϕ_1, ϕ_2 are the phase difference of J_1, J_2,

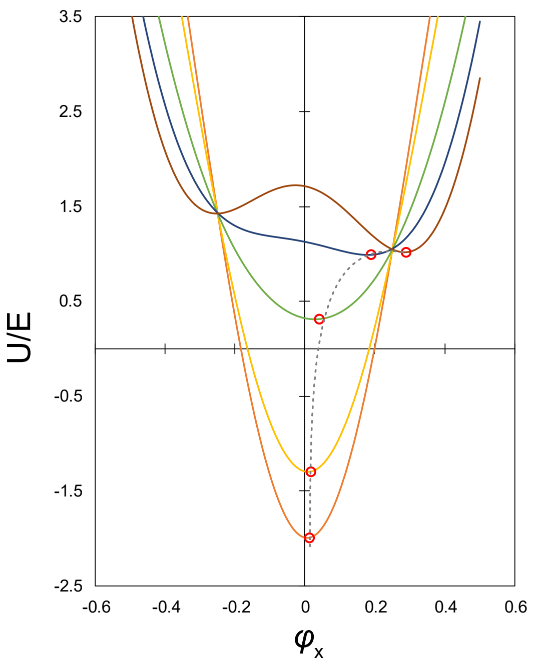


(a) The logic state is ‘0’.

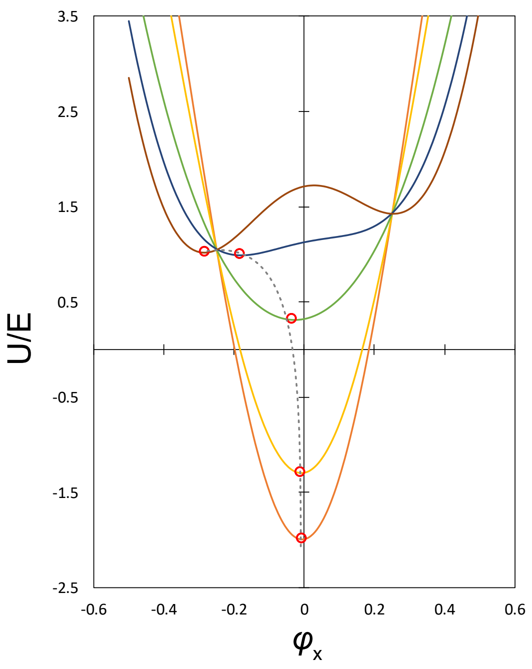


(b) The logic state is ‘1’.

Figure 2.2: Two states of the AQFP circuits. When the current of the center inductance of AQFP flows upwards, this is state “0”. If the current flows downwards, it is state “1”.



(a)



(b)

Figure 2.3: The figure shows the potential change of the AQFP circuit. The trajectory of the potential is shown by red dots and dashed lines. (a) corresponds to a positive input current and (b) corresponds to a negative input current.

respectively. $\beta_\mathrm{L}, \beta_\mathrm{q}$ denote the normalized inductance L, L_q, respectively.

And assuming $\phi_- = \phi_\mathrm{x}$, euqation (2.1) is changed as follow.

$$\frac{U}{E} = \left[\frac{(\phi_\mathrm{in} - \phi_+)^2}{\beta_\mathrm{L} + 2\beta_\mathrm{q}} - 2\cos\phi_\mathrm{x}\cos\phi_+ \right] \tag{2.9}$$

This equation is plotted graphically in figure 2.3. In the initial state, $I_\mathrm{x} = 0$, i.e., $\Phi_\mathrm{x} = 0$, the potential of the circuit has a single-well structure. When I_x is increased, the cosine term in the potential becomes dominant, and the structural changes to a double-well structure. Whether the minimum point of the potential takes a right or left is uniquely determined by the polarity of the current in Φ_in given as the signal input. When the potential tilts to the right, as in Figure figure 2.3(A), it means state "0", and when the potential tilts to the left, as in figure 2.3(B), it means state "1".

Energy dissipation of the AQFP during the clocking cycle E_sw is described below.

$$E_\mathrm{sw} \simeq 2I_\mathrm{c}\Phi_0\frac{\tau_\mathrm{j}}{\tau_\mathrm{x}} \tag{2.10}$$

$$\tau_\mathrm{j} \simeq 2\pi\frac{L_\mathrm{j}}{R} = \frac{\Phi_0}{I_\mathrm{c}R} = \sqrt{\frac{2\pi\Phi_0 C_\mathrm{s}}{\beta_\mathrm{c}J_\mathrm{c}}} \tag{2.11}$$

$$L_\mathrm{j} = \frac{\Phi_0}{2\pi I_\mathrm{c}} \tag{2.12}$$

$$\beta_\mathrm{c} = Q^2 = \frac{2\pi R^2 C I_\mathrm{c}}{\Phi_0} \tag{2.13}$$

$$E_\mathrm{sw} \simeq \frac{2\Phi_0^2}{R\tau_\mathrm{x}} \tag{2.14}$$

L_j, R, C_s,C,J_c and β_c are Josephson inductance, the equivalent resistance of the sub-gap resistance and shunt resistor, junction capacitance per area, junction capacitance, the critical current density and McCumber parameter[21], respectively. τ_x is the duration time of the switching process and τ_j is the time constant that comes from the characteristic of Josephson junctions. As can be seen from the equations (2.10), (2.11) and (2.14), power consumption can be reduced by flowing modification.

- Using high-J_c fabrication process.
- Using high-β_c Josephson junctions. In other words, to use unshunted JJ (to increase shunt resistance).

The typical energy dissipation of AQFP is $E_\mathrm{sw} \simeq 10 \times 10^{-19}$ J for $I_\mathrm{c} = 25\,\mu\mathrm{A}$, $C = 1\,\mathrm{pF}$, $R = 1\,\Omega$[22]. The combination of β_L and B determines the adiabatic change in the potential of the circuit. Decreasing β_L and β_q decreases the energy consumed. Also, decreasing β_L and increasing β_q increases the operating margin of the circuit for the excitation current. There is a trade-off between the operating margin of the circuit and the energy consumption, and the value of β_q determines it. For the AQFP circuit in the AIST HSTP process, a combination of β_L and β_q of 0.2 and 1.6 is used.

2.3 Data propagation of AQFP circuits

Microstrip or strip lines are used for signal propagation in the AQFP circuit. The circuit diagram is shown in figure 2.4. The AQFP circuit has a transformer in the output of the signal current. Since the binary of the AQFP circuit is determined by the polarity of the current, reversing the coupling direction of the transformer allows for NOT operation without additional implementation cost. When the buffer gate switches, a quantum magnetic flux $\Phi_0 = 2.067\,851 \times 10^{-15}$ Wb penetrates the superconducting loop. A circulating current flows through the loop to counteract the magnetic flux, and the current is determined by $I_{\text{loop}} = \Phi_0/L[\text{A}]$. Since the coupling coefficient of the transformer is about $k \simeq 0.5$, the current flowing in the superconducting loop including the signal line is $I_{\text{signal}} \simeq 0.5\Phi_0/L_{\text{stripline}}[\text{A}]$. The input current to the gate I_{signal} must be sufficiently larger than the gray zone of the AQFP buffer gate. The minimum required current has been studied based on simulation and measurement results, and it is assumed to be $I_{\text{th}} = 7\,\mu\text{A}$. Considering the standard microstrip lines in use, the wiring length is about 1000 µm.

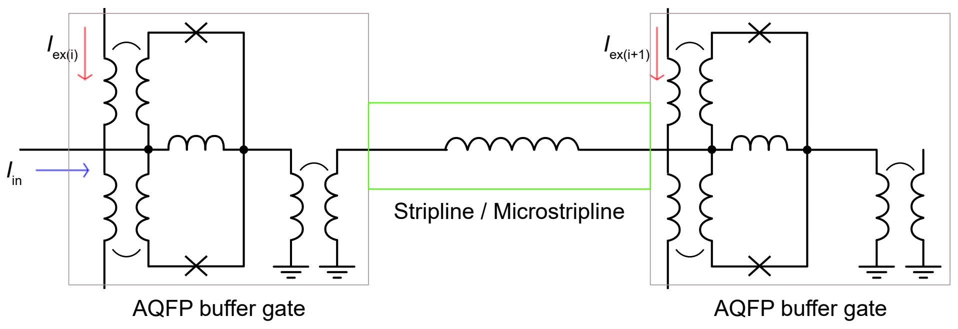


Figure 2.4: Signal propagation of AQFP circuit. The AQFP buffer gate contains a transformer, and the output current of the buffer gate propagates the signal through the transformer and stripline or microstrip line to the next gate. The inductance of the superconducting loop determines the current flowing in the signal line.

2.4 Excitation of AQFP circuit

AQFP circuits can be operated by applying a periodic current, and the current waveform can be sinusoidal, square, or triangular. However, since AQFP circuits can reduce power consumption by slowing the switching process, providing discontinuous waveforms will increase power consumption. Therefore, a sine wave is usually used to excite the circuit. A three-phase excitation scheme was first proposed, in which the phase difference between the excitation currents applied to the sending gate and the receiving gate is 120°. Then, it is followed by a four-phase excitation scheme in which the phase difference is 90° [23]. The four-phase excitation method has a broader timing margin of the circuit, and the stability of the circuit at high-speed operation is improved. There are also proposals for 8- and 12-phase excitation schemes and delay

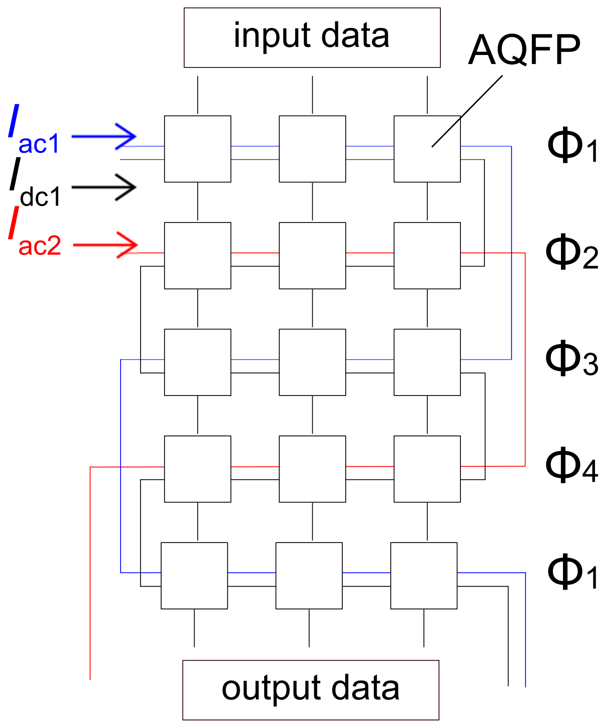


Figure 2.5: Diagram of the circuit configuration for the 4-phase excitation scheme of the AQFP circuit. There are three excitation currents are wired in meander form from left to right. The DC current is wired sequentially from the top to bottom, and the AC currents are wired from top to bottom by skipping one phase in sequence. As shown in figure 2.6, the two currents have a phase difference of $1/2\pi$. The interaction of the AC and DC currents flow directions excites the circuit four times per cycle. Therefore, the i-th and $(i + 4)$-th gates from the top are excited at the same time.

line clock, which uses on-chip delay lines to create a phase difference[24, 25]. However, unless otherwise noted, this paper will focus on the 4-phase excitation scheme. Figures 2.5 and 2.6 shows 4-phase clocking scheme and placement of the gate.

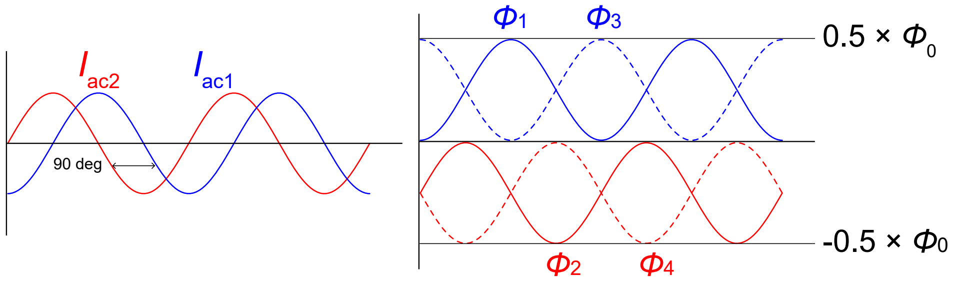


Figure 2.6: A method of creating four phase differences by the interaction of DC and AC currents. The magnitude of the three currents is equivalent to $1/4\phi_0$, and the two AC currents have a phase difference of $1/2\pi$. The DC current is used to provide an offset to the AC current, and when wired as shown in figure 2.5, the AC and DC currents are positively biased when they are in the same direction and negatively biased when they are in opposite directions. Since the AQFP circuit has no polarity, the circuit switches when the total input current exceeds $\pm 1/2\phi_0$, so wiring it as shown in the figure can create four phase differences.

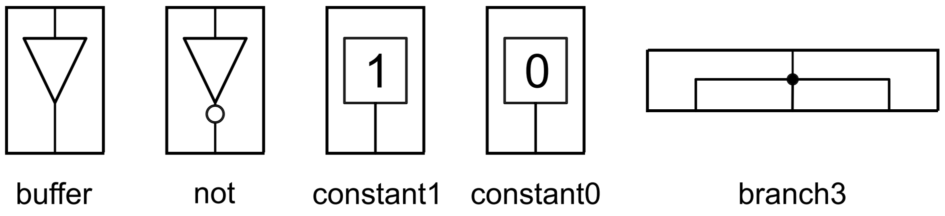


Figure 2.7: Basic symbols for AQFP circuits. From left to right: buffer, not, constant0, constant1 and branch3. Branch circuits have different numbers of branches, branches 2, 3, 4, and 5.

2.5 Logic circuit using the AQFP circuit

2.5.1 Basic cells of AQFP logic circuits

The AQFP circuit has four basic gates [26]. The respective circuit symbols are shown in figure 2.7.

buffer Figure 2.1 shows schematic of buffer gate. When the circuit is excited, a current in the same direction as the input current is generated. Since the gate does not perform logic operations, it is used to synchronize signal timing or to amplify current for long-distance propagation.

not(invert) In figure 2.4, the circuit with a negative coupling coefficient for the output transformer is the inverter gate. A current in the opposite direction of the input current is generated.

constant A gate with no input terminals, only output terminals. A gate in which the loop inductance of the superconducting loop is asymmetric, biasing the ease of switching the left and right Josephson junctions. The const0 gate and the const1 gate are mirror images of each other.

branch A gate for splitting and merging currents. It is used to create branching and majority gates as described below.

The majority gate is used as basic gates for operations in AQFP circuits. A majority gate is a logic circuit that has $2n+1$ fan-in and 1 fan-out. It is a gate that outputs "0" when $n+1$ inputs are "0" and outputs "1" when $n+1$ inputs are "1" and outputs the one with the higher number of logic inputs. So far, 3-input, 1-output and 5-input, 1-output majority logic gates have been designed and included in the cell library [27]. Logic equations of the 3-input majority gate and the 5-input majority gates are shown in euqation (2.15) and euqation (2.16), respectively.

$$Maj(a,b,c) = ab + bc + ca \tag{2.15}$$
$$Maj(a,b,c,d,e) = abc + abd + abe + acd + ace + ade + bcd + bce + bde + cde \tag{2.16}$$

The principle of operation of majority logic gates is explained qualitatively: a three-input majority logic gate has a structure that uses branch3 to join the outputs of three buffer gates. As shown in the figure, when a 3-input majority logic gate receives inputs {“0”, “1”, “1”}, the AQFP circuit assigns the logic state to the direction of current flow, so the direction of the output current is “1” according to Kirchhoff’s law. A 5-input majority logic gate works in the same way by using a 5-input branch cell.

In principle, a majority gate can be created using only a branch circuit, since the current merging is used for the calculation. However, since the input current is determined by the magnitude of the wiring inductance, it is difficult to match the amount of current. For this reason, the majority logic gate in the cell library has a buffer gate attached to the input section to align the input current level. figure 2.9 shows a block diagram of a majority logic gate and its symbol.

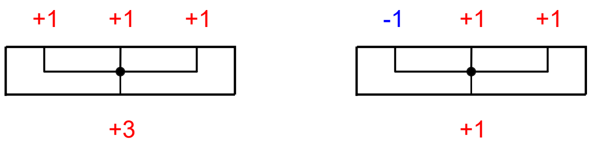


Figure 2.8: Inside the majority logic gate, currents are merged, which represents majority logic. “-1” means upward current and “1” means downward current.

As can be seen from euqation (2.15), the majority logic gate is a combination of AND and OR operations. By performing conversions such as those in equations (2.17) to (2.19), they can be used as majority logic gates AND, OR, and NOR gates. figure 2.10 shows schematic of each gates.

$$\text{AND}: \quad a\cdot 0+0\cdot c+ca=ac \tag{2.17}$$

$$\text{OR}: \quad a\cdot 1+1\cdot c+ca=a+c \tag{2.18}$$

$$\text{NOR}: \quad \neg a\cdot 1+1\cdot\neg c+\neg a\neg c=\neg a\neg c=\neg(a+c) \tag{2.19}$$

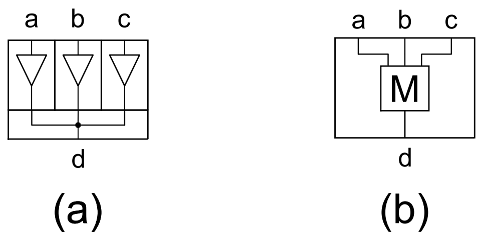


Figure 2.9: (a)Block diagram of a 3-input majority logic gate. buffer is attached to the input of the branch block to align the magnitudes of the input currents. Since the magnitude of the output current of the buffer and the not gate is equal, the majority logic operations with negative inputs are also possible by installing a not gate instead of a buffer gate. (b) Symbol of 3 input majority gate

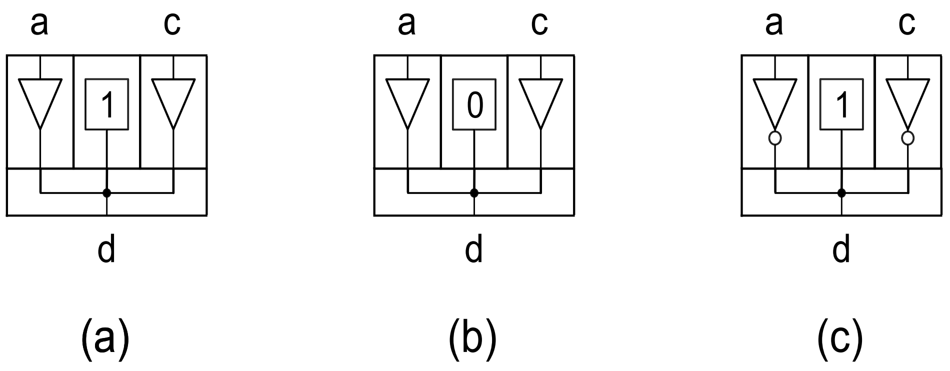


Figure 2.10: Schematic of (a) OR gate, (b)AND gate, and (c) NOR gate. Equations (2.17) to (2.19) bring these conversions. The gate placed center in the schematic is the constant output gate.

2.5.2 Energy dissipation of AQFP logic gate

Evaluating the energy consumption of AQFP circuits is more difficult than for RSFQ or CMOS circuits. This is because AQFP circuits do not have static power consumption, and because it is an adiabatic circuit, all energy received by the circuit is not dissipated during the calculation. The power consumption of an AQFP buffer gate E_{AQFP} is calculated by the following equation, where $I_{\mathrm{L}}(t)$ is the current in the excitation line, $V_{\mathrm{L}}(t)$ is the potential difference of the excitation line, and f is the operating frequency.

$$E_{\mathrm{AQFP}} = \int_0^{\frac{2\pi}{f}} I_{\mathrm{L}}(t) \times V_{\mathrm{L}}(t)dt \tag{2.20}$$

As shown in the figure 2.11, the power consumption of the AQFP buffer gate is proportional to the frequency, and the power consumption is zero when the circuit is excited using infinite time. The majority gate used for logic operations is energy saturated because any buffer whose input direction differs from that of another buffer causes that buffer to operate non-adiabatically. The most power-hungry gate is spl2L, a branch cell using a Josephson junction with a critical current of 100 µA.

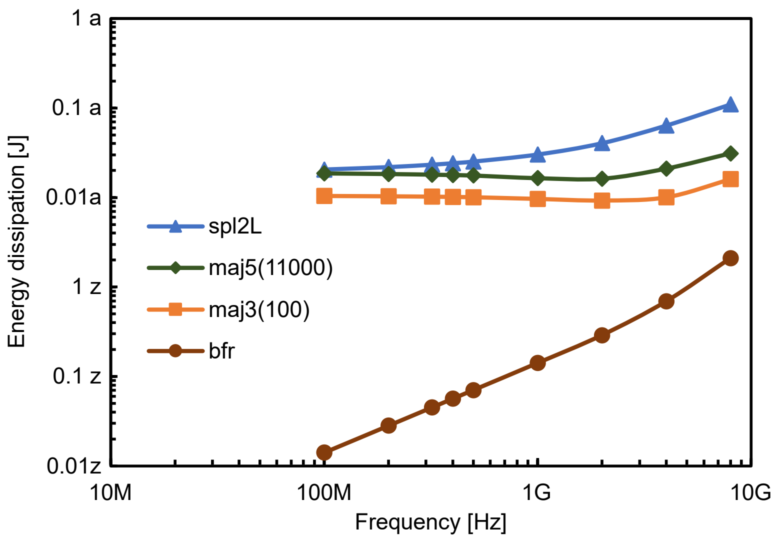


Figure 2.11: The graph shows the frequency dependence of the power consumption of the AQFP circuit. The energy consumption of the AQFP buffer gate is proportional to frequency. AQFP logic gates consume different amounts of energy depending on the input combination, and the combination with the largest energy consumption is shown as a representative example.

In the range above 5 GHz, energy consumption increases a little faster. This is due to the effect of plasma oscillations determined by the parameters of the Josephson junction. This effect is based on the AIST HSTP process and can be mitigated by using a process with a higher critical current density.

Evaluating the energy of gates that perform logic operations is a bit more complicated. This is because energy is exchanged between logic gates, and a method of evaluating energy that takes this effect into account is necessary. A systematic method has been proposed and is described below [28]. First, a somewhat long buffer chain is connected above and below the circuit to be measured and simulated, as shown in the figure 2.12. Total energy E_{tot} and consumed energy in the device under the test(DUT) E_{DUT} can be calculated in following equations.

$$E_{\text{tot}} = E_{\text{ac1}} + E_{\text{ac2}} = \int W_{\text{ac1}} + \int W_{\text{ac2}} \tag{2.21}$$

$$E_{\text{DUT}} = E_{\text{ac1-DUT}} + E_{\text{ac2-DUT}} = \int W_{\text{ac1-DUT}} + \int W_{\text{ac2-DUT}} \tag{2.22}$$

Then, to find a linear approximation function of E_{diff}, the difference of $E_{\text{tot}} - E_{\text{DUT}}$. The constant term of this function is the energy received by the DUT from the outside as signal current. Therefore, the energy consumed by the AQFP circuit, E_{circuit}, is $E_{\text{circuit}}(f) = E_{\text{DUT}}(f) + E_{\text{diff}} + \textit{const}$ Figure 2.13 shows curve of measured energy. The difference of energy E_{diff} becomes almost linear.

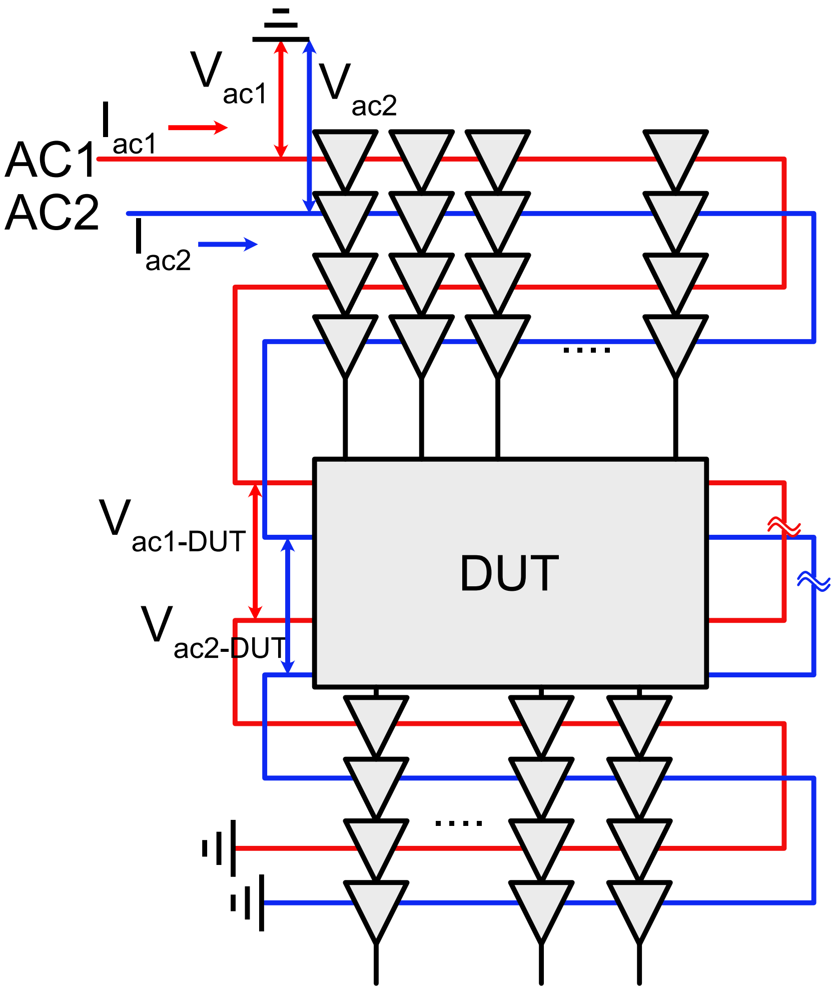


Figure 2.12: Circuit diagram for measuring the energy consumption of an AQFP circuit. A buffer chain is connected to the input and output ends of the Design Under Test (DUT), and the energy of the entire circuit and the energy consumed by the DUT are obtained from the simulation.

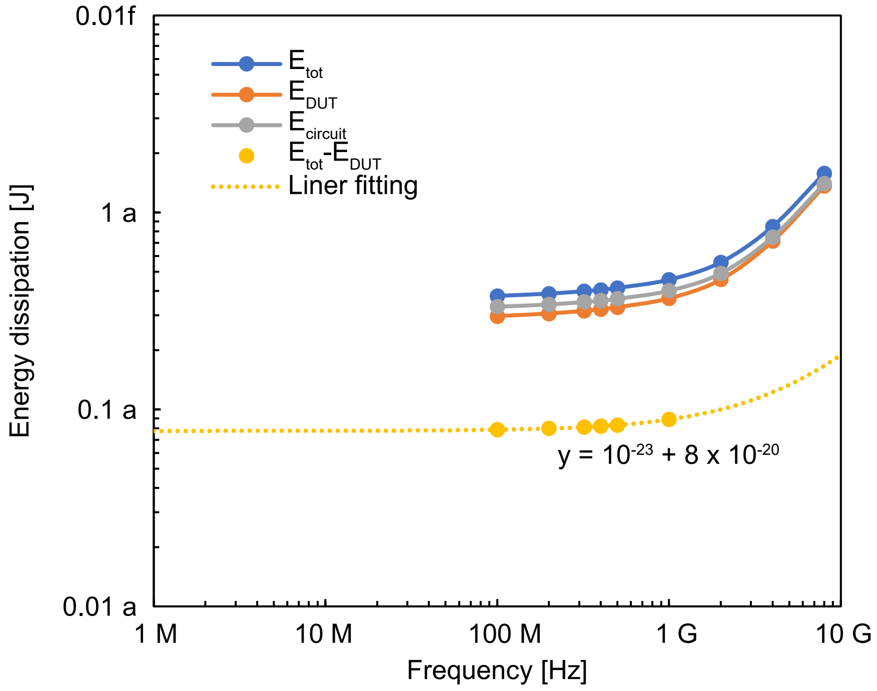


Figure 2.13: An example of how to calculate energy consumption. Calculate the energy consumed by the entire circuit, E_{tot}, and the energy consumed by the DUT, E_{DUT}, and then find the difference between E_{tot} and E_{DUT}, E_{diff}. Since the constant term in the linear approximation of E_{diff} is the energy received and consumed by the DUT from sources other than the excitation line, the sum of E_{DUT} and the constant term is the energy consumed by the circuit, E_{circuit}.

Figures 2.14 and 2.15 show the energy consumption of a 3-input majority logic gate and a 5-input majority logic gate.

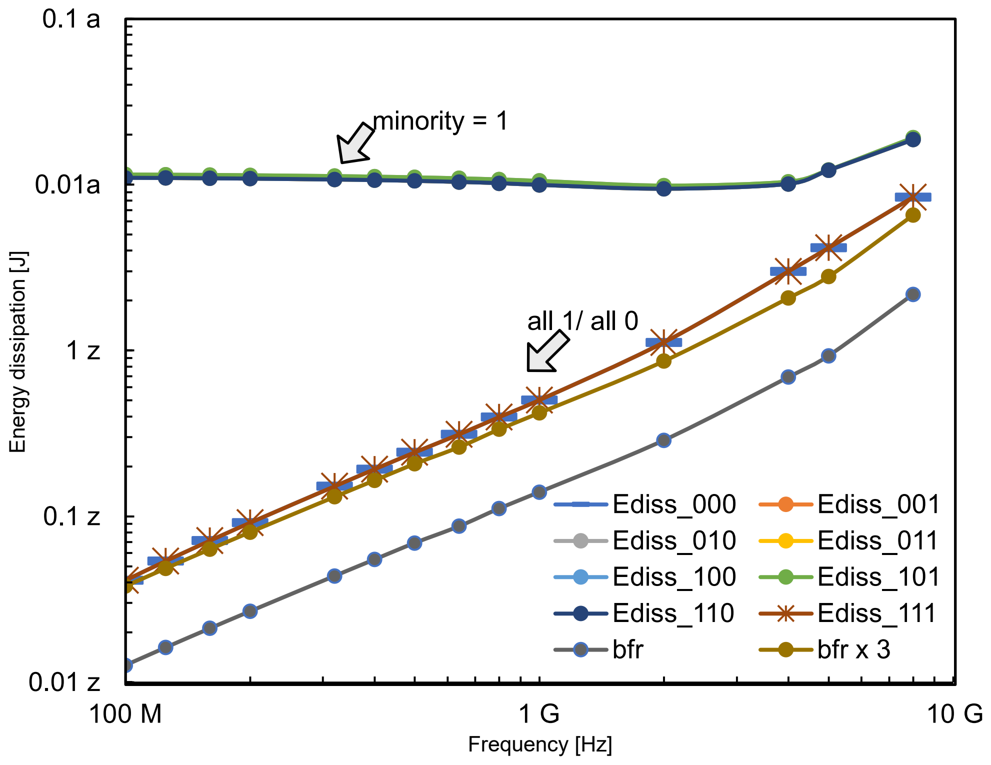


Figure 2.14: The graph shows the frequency dependence of the power consumption of the AQFP 3-input majority gate. The three-input majority logic gate consists of three buffers, and when all inputs are equal ($Ediss_{000}$, $Ediss_{111}$), the energy consumed is approximately equal to the energy of the three buffers. In other cases, the operation is non-adiabatic and there is a lower bound on the energy consumption, which is approximately 0.01 aJ.

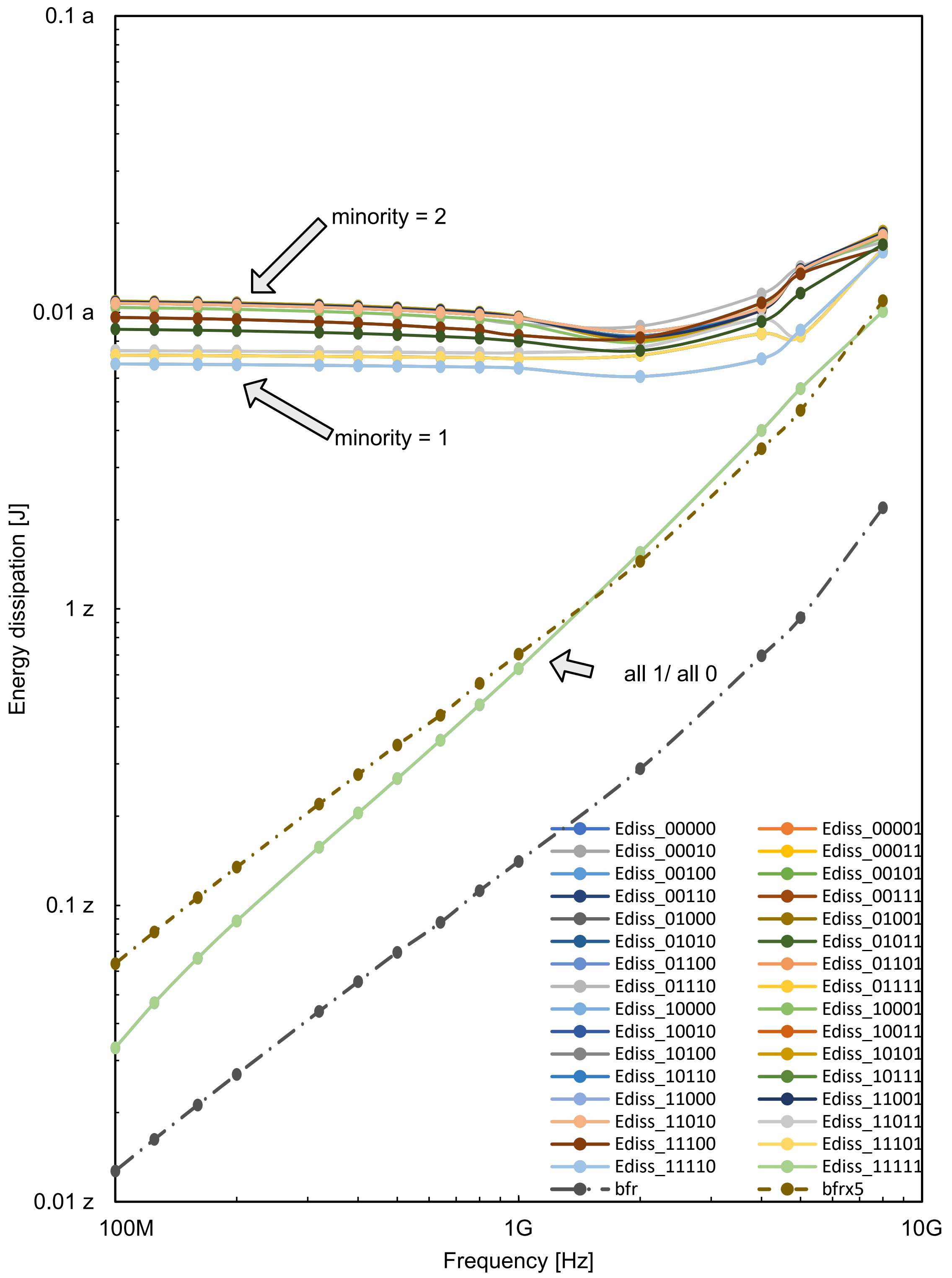


Figure 2.15: The graph shows the frequency dependence of the power consumption of the AQFP 5-input majority gate. A 5-input majority logic gate consists of five buffers, and when all inputs are equal ($Ediss_{00000}$, $Ediss_{11111}$), the energy consumed is approximately equal to the energy of five buffers. It also has two ranks of energy consumption when operating in a non-adiabatic manner, which is determined by the number of minority inputs, and the energy consumption when there are two minority inputs is greater. Minority = 1 and minority = 2 indicate the number of minority inputs.

2.5.3 Design flow of AQFP circuit

AQFP circuit basic cells and logic cells can be combined to create a circuit that operates. A 1-bit full adder is used as an example. The function for a full adder is shown in equations (2.23) and (2.24).

$$C_{\mathrm{out}} = ab + bc + ca \tag{2.23}$$

$$S = a \oplus b \oplus c \tag{2.24}$$

Logic synthesis

$$C_{\mathrm{out}} = Maj\left(a, b, c\right) = ab + bc + ca \tag{2.25}$$

$$S = Maj\left(\neg Maj\left(a, b, c\right), Maj\left(a, b, \neg c\right), c\right) \tag{2.26}$$

a, b denote the inputs, c denotes the lower bit carry input, C_{out} denotes the carry output, and S denotes the sum output.

First, consider what kind of logic gates are used to create a structure that satisfies the logical equation. Since the equation written in equations (2.23) and (2.24) is in disjunctive canonical form, we use methods [29, 30] to transform it into the following equation.

$$S = \begin{cases} \neg ab(a+b) = (\neg a + \neg b)(a+b) = a \oplus b & (c=0) \\ \neg(a+b) + ab = (\neg a \neg b) + ab & (c=1) \end{cases} \tag{2.27}$$

$$= \neg c(a \oplus b) + c(\neg a \neg b) + ab = a \oplus b \oplus c \tag{2.28}$$

This logical equation equations (2.25) and (2.28) are expressed in terms of a network graph as shown in figure 2.16.

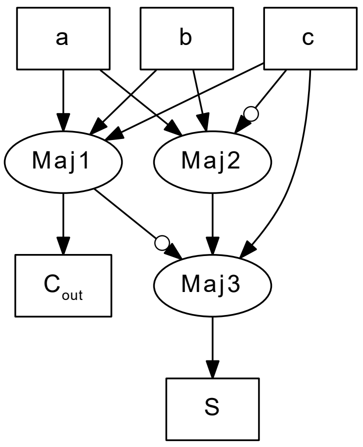


Figure 2.16: Network graph of full adder circuit by using majority logic gate. In this graph negative connection decreases as dot input.

Since the branching of signals in the AQFP circuit requires branching gates (SPL2 and SPL3), the appropriate branching tree is connected below the logic gates. The network graph of the full adder after connecting the branch gates is shown in figure 2.17.

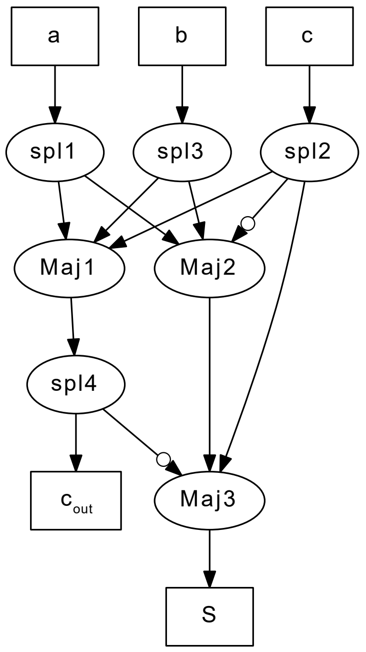


Figure 2.17: Circuit diagram of figure 2.16 with the split cells added.

Finally, a buffer is inserted to adjust the signal timing as shown in figure 2.18.

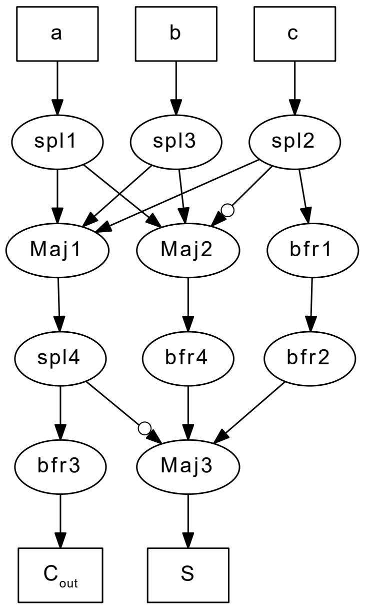


Figure 2.18: Network graph of the full adder, correctly designed as a logic circuit. Signal branch cells and buffer cells to adjust signal timing are inserted. This circuit is referred to as a type-A total adder for convenience.

We refer to this graph structure shown in figure 2.18 of full adder as type-A. It is a well-balanced structure in terms of energy consumed and latency. The graph shown in figure 2.19 is a type-B full adder, which has a shorter latency of C_{out} than type-A. This feature is effective to design ripple carry adder. However, it consumes more energy than type-A due to a large number of branch cells and majority gates. As described above, even circuits with the same logic formulas have various characteristics, and it is very important to consider the implementation that suits the purpose.

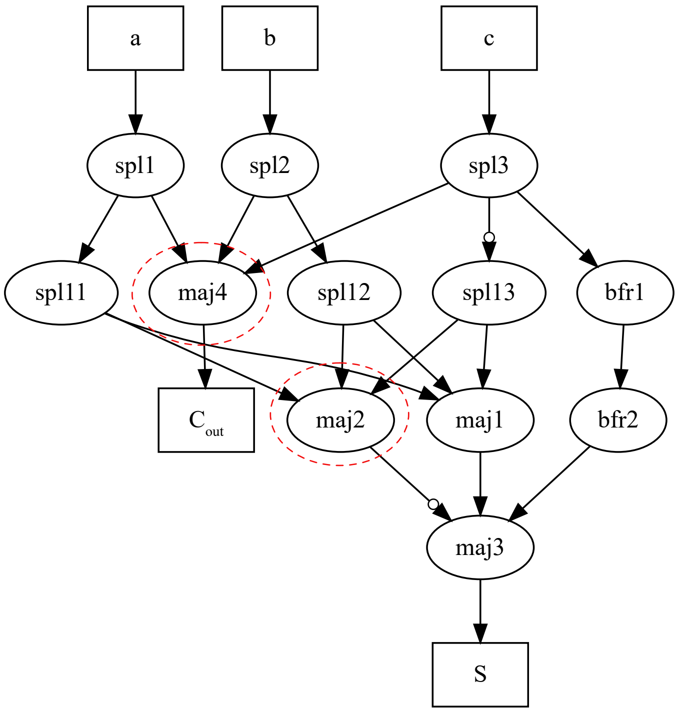


Figure 2.19: Network graph of the full adder called type-B, has a shorter latency of C_{out} than type-A. The two majority gates enclosed by the red dashed line perform the same operation.

Chapter 3

RTL to GDS flow for large AQFP circuits

3.1 Introduction

In previous, various circuits have been designed and demonstrated using AQFP circuits. Examples of large-scale circuits include the 4-bit microprocessor MANA and the 16-bit Kogge-Stone adder[15, 31]. We designed these circuits mostly by hand, which required a long time to design. In particular, we spent much time verifying the designed layout. To use AQFP circuits in practical applications, even larger designs with more complexity are needed. The design environment must be improved to enable efficient circuit design.

There have been reports on the construction of semi-custom design flows for AQFP circuits[32] and methods for efficient placement of standard cell designs [33, 34, 35]. This chapter reports on the construction of a highly integrated full-custom design flow and a top-down design flow that outputs GDS from RTL in a fully automated manner, using Synopsys tools.

3.2 Full-custom environment setup and cell library development

3.2.1 Full-custom setup

We have created an environment that allows designers to design custom AQFP circuits on a schematic layout editor known as Synopsys Custom Compiler. This tool is integrated with an analog/digital simulator and layout verification tool.

For analog simulation, PrimeSim HSPICE is used [36]. Compared to the previously used superconductor circuit analog simulators such as jsim[37], JoSIM[38], and WRSpice, it is a user-friendly simulator because it can use a variety of configurable voltage sources and can interactively set up simulation from the circuit design window [39]. It is also available to simulate circuits with sweeping parameters. This feature is convenient for checking the robustness of the designed circuit against fabrication parameter spread. Digital simulation is also available through PrimeSim XA (mixed signal mode) by invoking the System Verilog views developed for each AQFP logic cell.

We set up the layout versus schematic (LVS) on Custom Compiler for the layout verification tool. It is of great interest to circuit designers to know whether the circuit layout they have created correctly corresponds to a specified schematic. This checking is done carefully, but a minor design error can lead to a large failure[40]. LVS performs a mechanical comparison of the circuit layout and schematic, and plays a vital role in circuit design.

Our LVS tool currently supports checking the topology of graphs extracted from the schematic and layout. Figures 3.1 and 3.2 show the schematic and layout of the buffer gate of the AQFP, and LVS has verified the equivalence of these two. When the circuit is verified to be equivalent, the notification shown in listing 3.1 is obtained. LVS is available not only for the cell designed under the full-custom flow but also for larger circuits designed such as those built from the standard cell library. Inductor parameter checks are normally done via simple sheet inductance calculations which are not suitable for the complex multi-layer inductances that the AQFP has. Integration with a 3D solver such as STAR or InductEx [40] is necessary and will be in consideration in the future.

These developments have made designing AQFP circuits much more convenient than before. We used this environment to create cell libraries for the RTL-to-GDS flow of AQFP circuits.

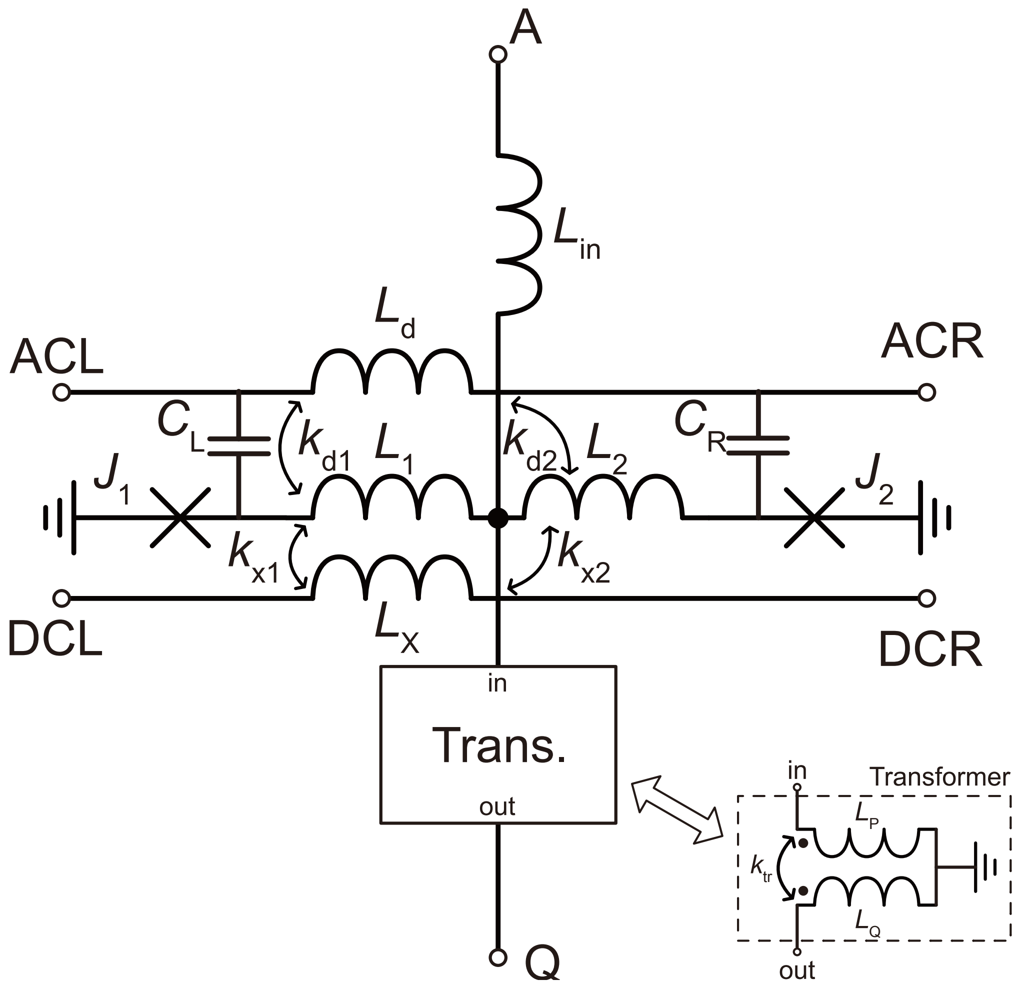


Figure 3.1: Schematic of an AQFP circuit.L_{in} =1.67 pH, $L_1 = L_2$ =1.42 pH, L_x =5.19 pH, L_d =5.17 pH, L_p =8.39 pH, L_s =30.9 pH, $C_L = C_R$ =2.90 fF, $k_{x1} = k_{x2} = -0.22$, $k_{d1} = k_{d2} = -0.14$, $k_{tr} = 0.49$ and critical current of J_1 and J_2 are 50 µA. Parasitic couplings are ommited in this figure.

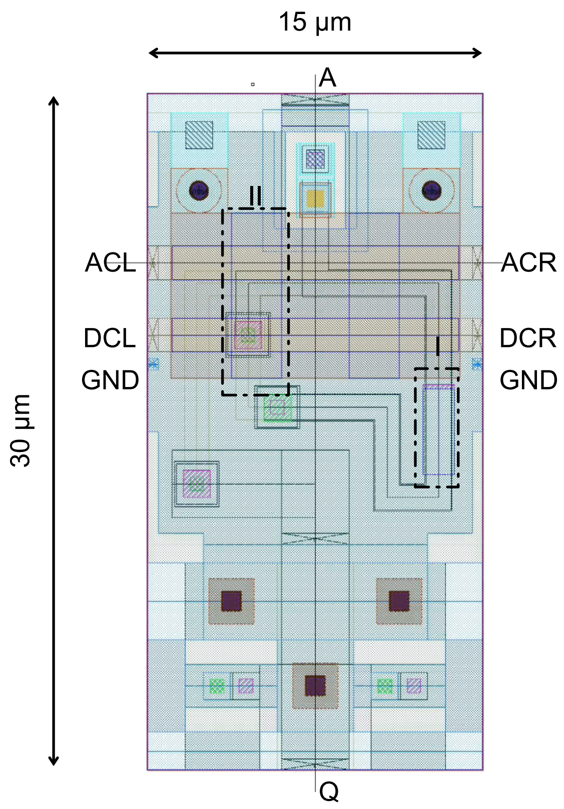


Figure 3.2: Layout view of the buffer gate. ‘A’ is the data input, ‘Q’ is the data output, ‘ACL’ and ‘ACR’ are the left/right AC power-clock ports, and ‘DCL’ and ‘DCR’ are the left/right DC offset ports. The virtual GND lines for LVS in the standard cell design of the AQFP circuit are located below DCL and DCR. Blue rectangles in the dotted line box (I) and (II) are examples of using the inductance annotation layers for LVS. (I) indicates the primary inductor of the transformer and (II) indicates the individual inductance L_1.

3.3 AQFP RTL-to-GDS flow

3.3.1 Cell libarary development

We have designed a standard cell library for the MIT Lincoln Laboratory SFQ5ee process [41] for automated design flows. The critical current density of the Josephson junction (JJ) is 100 A/µm^2, and it has eight superconductor Nb layers available for circuit design [42]. The basic structure of the cell follows previous studies with some improvements to enable the previously mentioned verification flows [43].

We prepared two types of cells with the same function, one is an “odd” cell, and the other is an “even” cell, which is a mirrored image of the odd cell. The clocking of the AQFP circuit uses a method called 4-phase clocking[44], in which the DC bias current and two AC currents with phase differences of 90 degrees are wired in a meander pattern. Therefore, the DC bias currents flow from the left for cells excited by an odd numbered phase and from the right for cells excited by even numbered

Listing 3.1: LVS execution log. It shows the results of comparing the netlist extracted from schematic and layout. The last line's "bfr == bfr" indicates that two netlists are matched.

```
Results Summary
----------------
LVS Device Extraction Error Summary
2 total rules were run.
0 rules NOT EXECUTED.
0 rules have violations.
There are 0 total violations.
Refer to bfr.LAYOUT_errors
----------------
LVS Compare Summary

LVS Compare Result: PASS
TOP equivalence point: [bfr,bfr]

  equivalence points checked: 1
    passed 1
    failed 0

Refer to bfr.LVS_ERRORS for LVS Error Diagnostics

  [PASS]  bfr == bfr    (level 0)
```

phases. Some gates, such as AND/OR gates, have a specific direction for DC bias currents due to the asymmetry of the AQFP circuit inductance in the constant cells needed to perform AND/OR. To treat this problem previously, the designer must manually turn the cell when they placed these gates in even-numbered positions. We decided to add the even cells that were initially flipped over to the standard cell library. Figure 3.3 shows AQFP AND/OR full adder circuit. Cells placed on ϕ_2, ϕ_4 receive DC from right to left and thus "even" cells are chosen from the cell library for these phases. For ϕ_1, ϕ_3, "odd" cells are chosen. Making these odd/even cells explicitly available simplifies the modifications needed to obtain a correct placement.

AQFP circuits also have the problem of low circuit drive power, which is a major obstacle for long internal wiring and large branches/splitters. A circuit called a booster can solve this problem. Since it can reduce circuit latency and area. A four-branch cell using the booster and a long-distance wiring cell are included in the cell library. Further, cells are individually characterized to identify the maximum interconnect length they can drive. This information is included in a Liberty file that describes the driving strength of each cell for use in the top-down design flow.

The larger the circuit, the more the effect of signal delay time needs to be considered. Since AQFP circuits have logic gates connected in a daisy chain to the clock (excitation current), attention must be paid to the clock delay time. The wiring which provides the excitation current is changed from a simple inductor model to an LC ladder model [45]. The excitation current overlaps the inductance of the SQUID part of the AQFP circuit. Therefore, there are both magnetic, and capacitive coupling effects. It is now possible to evaluate the excitation timing discrepancy due to the propagation delay of the excitation current.

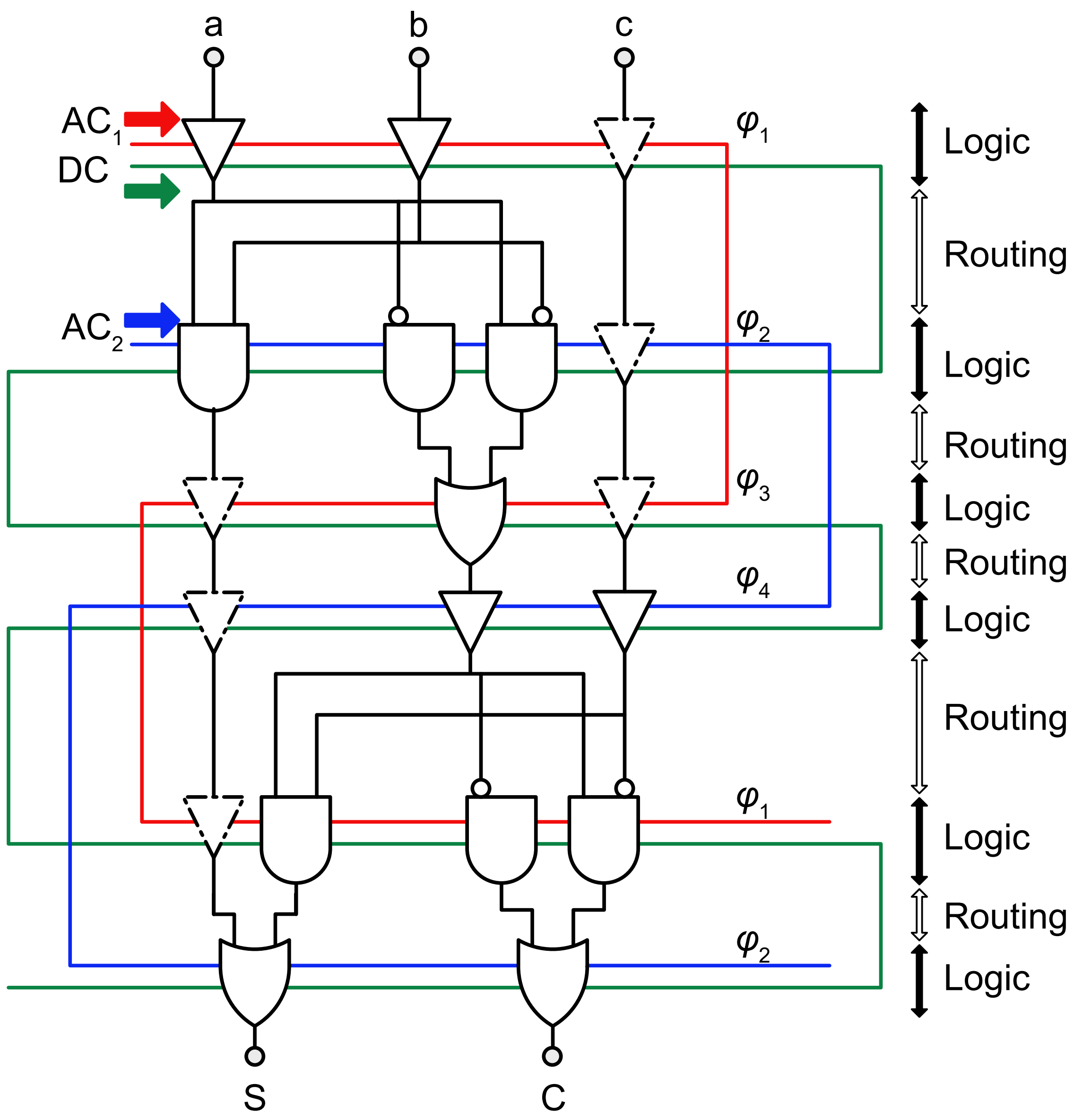


Figure 3.3: Schematic of AQFP AND/OR full adder. Two AC lines and one DC line are wired in a meander shape like this figure. The region of routing and logic cells are placed alternately. The buffer gate depicted by the dashed line is inserted during the path-balancing process. "Even" cells are used which belong to ϕ_2, ϕ_4.

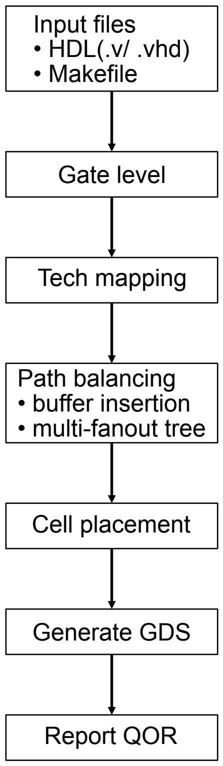


Figure 3.4: Two files are prepared: a Makefile and an HDL file describing the circuit to be designed. The Makefile contains the binary path and the module name, and the top-down flow begins by executing the Makefile with the "make" command.

3.3.2 AQFP RTL-to-GDS flow

Cell placement and routing are done within Synopsys's framework of the Fusion Compiler system. The tool set is highly customizable through TCL scripting, with a very flexible and complete interface to the design database. Our AQFP flow consists of a master script, which calls subsidiary scripts to perform the various tasks, from library compilation through synthesis, formality checking, place and route, and documentation. Figure 3.4 shows a rough flowchart of this top-down flow.

Initially, technology mapping to the AQFP library is performed. Synthesis uses abstract cells that avoid issues particular to AQFP, such as using a multi-phase clock, so mapping to the "real" library cells is required. Once mapped, the next operation is path balancing, ensuring that the combinational cells' inputs originate from the same logic depth within a pipeline. This process involves adding AQFP buffer cells into the design, generally in large numbers. As a by-product, all combinational cells are assigned a logic level number, starting from one at the beginning of each pipeline state.

A floor plan for the design is generated from the per-level gate counts and externally-set variables. This assembles the cell placements into rows, where each logic level has a corresponding cell row, starting from the top. A fixed-height wiring channel separates the rows. As a consequence of path balancing, all routes will connect adjacent rows, like figure 3.3. Cell placement and routing are usually done within this floor plan, minimizing a cost function that depends heavily on the total wire length. The routing flow is timing dependent and can compensate for timing errors by adding buffering or adjusting path length. Our flow uses four physical routing layers and horizontal and vertical strip passive transmission lines. These lines have a known delay proportional to distance. Clock delay is obtained from cell clock terminal placement, as the clocking routes are predictable from the floor plan. Again, the delay is proportional to the distance along the clock lines from a reference point.

Once placement and routing are complete, the GDS layout file and a QoR report are generated. The report identifies possible DRC errors and timing violations. In a typical design, there are often congestion points where routing effectively fails.

3.3.3 Quality of result

Implementation results are shown in table 3.1 and figure 3.5. In the appendix, there are raw data to make this table. We compiled a set of ISCAS-85 benchmarks [46] and multi-bit adders to compare with a previous study, ASAP [33]. Among the ISCAS-85 benchmarks, the C6288 circuit (a 16×16 array multiplier comprising half-adders and full-adders) yielded the largest JJ count at over 220k JJs. To investigate the scaling limitations of the tool, we included a simple, parameterized multi-bit adder with data word sizes ranging from 8-bit to 1024-bit. The comparison results are not completely fair because of the different cell libraries and different hardware used to compute the results, but nonetheless we can gain some insight into the differences between the approaches. Clearly, ASAP is superior in terms of execution speed. The overall JJ usage (fewer JJs is better) is also superior in ASAP as its usage is slightly better for some benchmarks, and much better in others, particularly the C6288 benchmark. One explanation for this is that ASAP was built from the ground up to support AQFP logic specifically. The RTL-to-GDS flow in this study still uses algorithms made for semiconductor circuits but with customized tool control via TCL scripting. For example, ASAP has a more proper treatment of AQFP buffer insertion and optimization (retiming). While buffer insertion is implemented in this study, it was not optimized to the extent ASAP was. Given that buffering can contribute to more than 90% of the JJs of an AQFP circuit [32], the lack of extensive buffering optimization can explain why the JJ usage is higher in this study.

However, our tool has successfully synthesized much larger circuits than ASAP. It consistently demonstrated its capability to realize designs in the range of 100k JJs to over one million JJs. The largest circuit successfully designed is a 1024-bit adder with two million JJs. The design took two days on a server with 1024 GB of memory in the cloud. Further, the internals of the RTL-to-GDS engine is shared between RSFQ and AQFP logic. The only difference is the customization of the flow through scripting. ASAP only supports AQFP logic. Lastly, the top-down design flow is well-integrated with the full-custom flow which provides a seamless user experience compared to the standalone flow available via ASAP.

Table 3.1: Comparison of placement results between ASAP[33] and proposal.

	Proposal		ASAP	
name	#JJs	time (s)	#JJs	time (s)
C17	72	35	50	9
C432	3958	193	2474	26
C499	6656	238	5046	41
C880	7546	363	6480	55
C1355	6584	333	4918	42
C1908	10112	557	5680	60
C2670	16258	570	7884	81
C3540	20520	973	17420	95
C5315	31352	1326	29454	374
C6288	220428	6383	49394	158
C7552	35066	1246	33530	591
8-bit adder	1072	288		
16-bit adder	2966	252		
32-bit adder	8028	936		
64-bit adder	20636	2124		
128-bit adder	53290	4608		
512-bit adder	567616	49536		
1024-bit adder	2275472	187200		

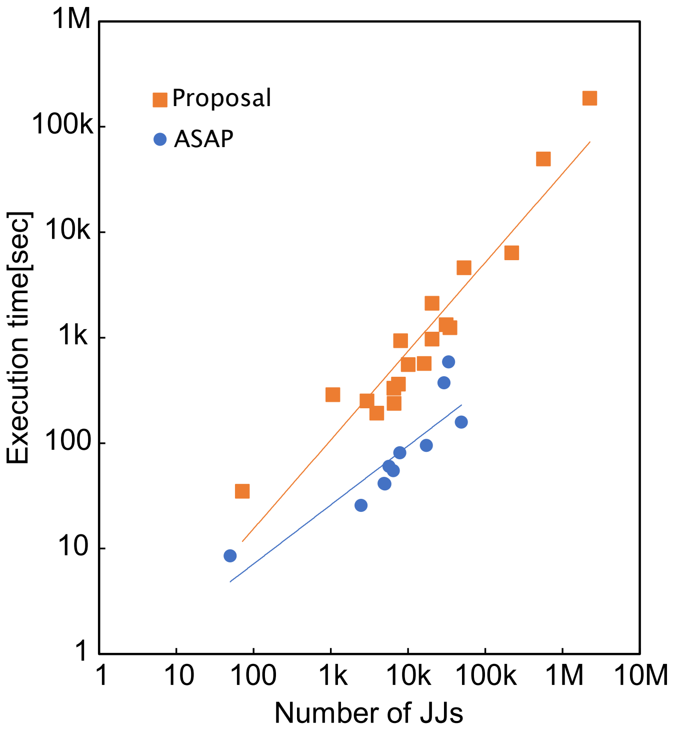


Figure 3.5: Graph of the number of Josephson junctions(JJs) vs. execution time. ASAP in the previous study is superior in both execution time and scalability. However, our method successfully designs circuits with larger JJs.

3.3.4 Discussion

During this study, the following was elucidated in regard to the characteristics of AQFP circuits compared to the conventional RSFQ circuits from perspective of RTL-to-GDS challenges.

In AQFP logic, the gates that make up the architectural latches are also clocked by the global excitation current[47]. These latches are composed of multiple cells and thus require multiple clock phases. It is difficult to resolve the timing of the combinational logic with architectural latch macros composed of multiple gates, so sequential circuits cannot be automatically synthesized at this moment. The quantum-flux-parametron latch (QFPL) is an AQFP-compatible latch without a feedback path, but it requires an additional control line[48] and still requires additional logic gates to interface with it. In addition, there have not been many reports of sequential circuit design using AQFP circuits. Methodologies have been proposed in [49, 50], but it still remains unclear how to properly automate the physical placement of the architectural latches together with the combinational logic, which all can influence the spacing between logic rows and apply even more interconnect constraints on a heavily constrained problem (interconnect length limitation).

In comparison, RSFQ logic has a high degree of freedom with the clock, making it easy to design complex circuits and cells with internal states. However, the overall number of JJs in the circuit increases because of the need for active JJ circuits to distribute the clock. Also, RSFQ logic tends to have more JJs per cell compared to AQFP logic [51]. Thus, synthesis flows for RSFQ logic can more easily approach higher JJ counts for the above reasons, especially because the lower cell count (with more JJs per cell) reduces the computation/memory load for the RTL-to-GDS flow when compared to AQFP logic.

AQFP physical synthesis is hindered by a more heavily constrained placement flow as cells cannot be placed on any arbitrary row, but only on a row that provides the required clock phase it needs to be excited by. For example, an AQFP cell that must be excited by ϕ_2 must be placed only on ϕ_2 equivalent rows in 4-phase clocking. This constrains the placement of the cell to just 25% of the rows available for a given circuit design which may be challenging for a placer to find a good solution.

RSFQ circuits allow transmission over long distances using passive transmission lines (PTLs)[52]. AQFP circuits use buffers as repeaters to transmit across long distances, which increases latency and creates more constraints on the place-and-route solution because the repeaters themselves also require a power-clock. This can be partially mitigated by the aforementioned boosters in Section 3.2. Another approach is to convert the AQFP signal to SFQ pulses to enable ballistic propagation using PTLs. However, the energy consumption of the RSFQ circuit to support this is 1000 times higher than that of the AQFP circuit, limiting its adaptability[53]. An intensive trade-off analysis needs to be conducted to determine what is the most suitable strategy.

Fewer JJs are required to perform arithmetic operations in AQFP logic. This is because it intrinsically uses majority logic as the basis of logic computation. Majority logic has been shown to be much more efficient for arithmetic [54, 29]. That is not to say that RSFQ logic cannot do majority logic, but the implementation of a 3-input majority gate (MAJ3) in RSFQ logic requires 23 JJs [55], while the MAJ3 in AQFP logic requires only 6 JJs [26].

3.4 Conclusion

We developed full-custom and top-down design flows using Synopsys EDA tools for AQFP circuits. The full-custom flow is integrated with simulation and verification tools such as analog/digital simulators and LVS. This provides an improved design efficiency and user experience.

The top-down flow is the first attempt to build a tool that fully automates the design from RTL to physical information that can be taped out. This is the first successful design of AQFP circuits exceeding millions of JJs.

The next steps for this work include the following: (1) enable circuit parameter verification of complex inductor structures by integrating 3D solvers into the LVS flow, (2) provide an easy way to adjust optimization constraints such as relaxing interconnect constraints or area constraints, and (3) provide support for novel clocking methodologies such as power-divider clocking [56] and delay-line clocking [57] as they may provide substantial opportunities to reduce latency and buffering [25, 58].

Chapter 4

Channel routing for AQFP circuits

This chapter describes the left edge algorithm, which is widely used in AQFP circuits, and the Glitter routing algorithm used when the routing width is not constant. Then, the Glitter-based routing inductance optimization algorithm is described and the results of its execution are discussed.

4.1 Routing structure of AQFP cirucits

The HSTP process has four metal layers (layers made of Nb): GP, BAS, COU, and CTL. The strip line structure is a structure in which a ground surrounds the top and bottom of the line through which the signals pass, and the GP layer covers the signal wire as the bottom ground and the COU layer for the top ground. Figure 4.1 shows the layer structure of the stripline at the HSTP process. Figure 4.2 shows the cross-section of signal lines. The CTL layer covers the signal wire as the upper ground, and the BAS and COU layers are used to cross the signals. Therefore, only two layers are available for signal routing.

The channel routing algorithm for AQFP circuits uses the most popular left-edge method[59].

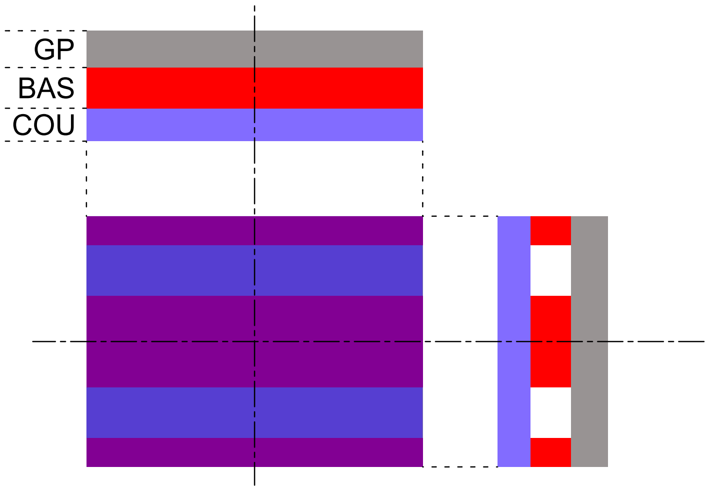


Figure 4.1: AQFP wiring stripline structure. The signal current passes through the BAS layer, with the COU and GP layers covering the top and bottom of the BAS layer. On the sides of the signal lines, the BAS layer wired along the signals connects the top and bottom grounds.

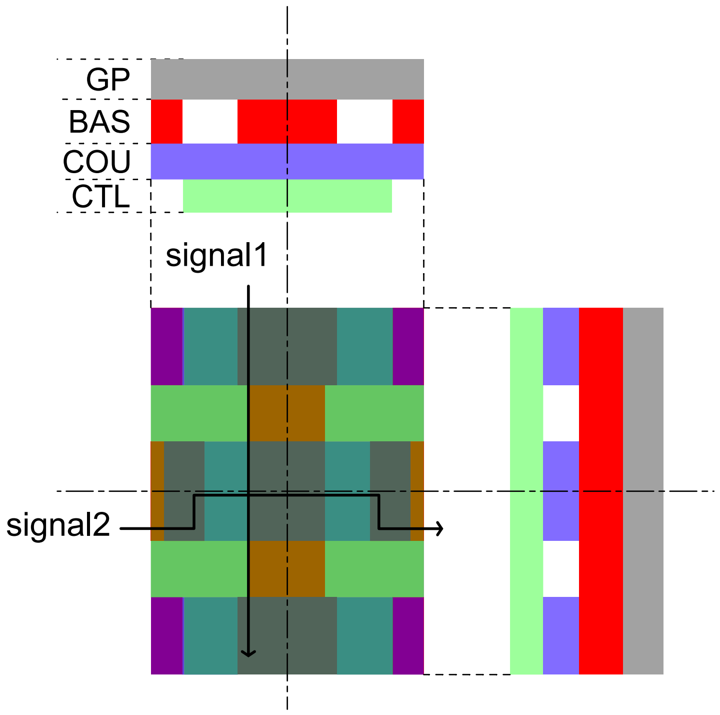


Figure 4.2: Diagram of a cross-section of signal lines in an AQFP circuit. The CTL layer covers the upper side, and the GP layer covers the lower side to protect the signal lines of the BAS and COU layers that cross.

4.1.1 Left edge algorithm

Left edge algorithm(LEA)[60, 61] is so named because it scans the signal edge from the left edge of the routing area and assigns the track (horizontal routing grid in the channel area) that the wires use. The algorithm creates two graphs from the state of the channel region and processes the graphs to perform the routing. figure 4.3 (a) shows an example of a wiring region.

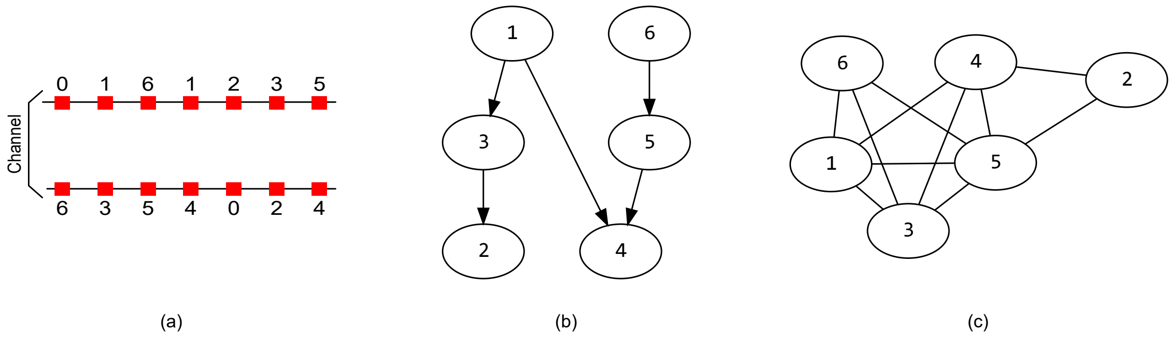


Figure 4.3: (a) Example of the channel area. Numbers indicate the connection relationship of each terminal. 0 indicates no terminal. (b) VCG created from Figure (a). (c) HCG created from Figure (a).

Vertical Constrain Graph

A vertical constrain graph (VCG) is a directed graph for determining the order in which to route. The VCG corresponding to (a) in figure 4.3 is (b). The left edge algorithm has the possibility of incorrect routing depending on the order of track assignment. For example, as shown in figure 4.4 (a), if signal lines '1' and '2' are placed in order from the bottom, there will be no vertical wiring area for signal line '2. This mistake can be solved by wiring signal line '2' first, as in figure 4.4 (b). The wiring order is specified in this situation by adding the edge '1' to '2' to the VGC. Wiring can be done without overlap by sequentially wiring from the signal line corresponding to the node of the leaf of the final created VCG. In the case of Figure 4.3(b), wiring '2' is permanently wired before wiring '1'. In addition, since wiring '2' and '4' have no relationship in VCG, there is no restriction on the wiring order.

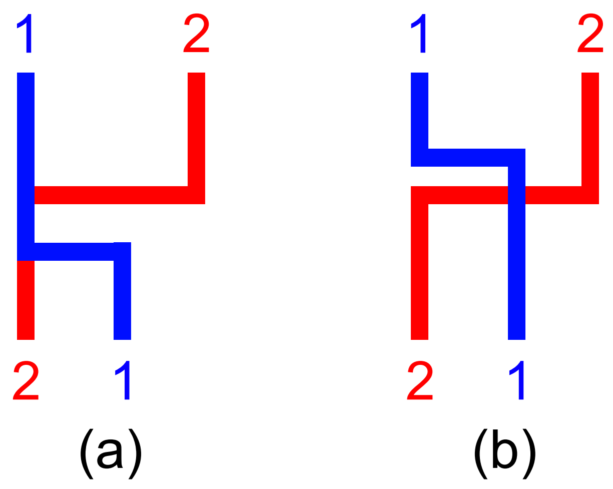


Figure 4.4: Cases where wiring must be constrained using VCG. In case (a), the vertical wiring on the left side is crossed, but this is eliminated in state (b). Changing the order of the horizontal wiring can control the crossing wires.

Horizontal Constrain Graph

The horizontal constrain graph (HCG) is an undirected graph describing the possibility of horizontal wiring overlap. figure 4.3 (c) is the HCG generated from (a). If two nodes of the HCG are adjacent to each other, the signal lines corresponding to those nodes cannot be routed on the same track. In figure 4.5 (a), node '1' and node '2' are connected, and in (b), there is no edge between node '1' and node '2'. In other words, in (b), wires '1' and '2' can be placed on the same track.

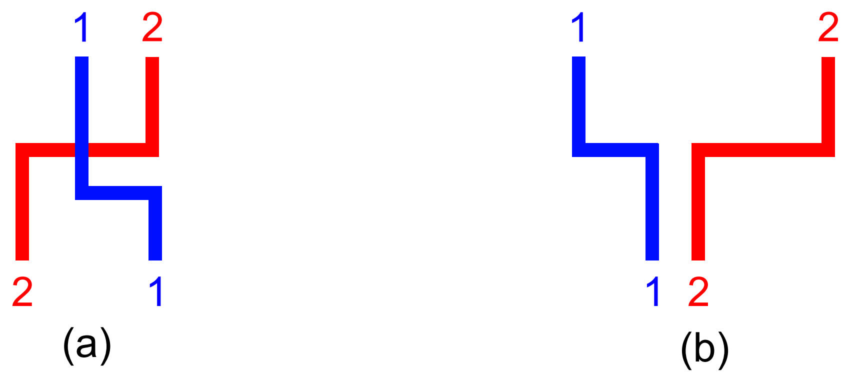


Figure 4.5: (a) has a relationship in the HCG, while (b) does not. In (a), assigning horizontal signal lines to the same track causes overlap, while in (b), this does not occur.

Algorithm flow of left edge algorithm

Algorithm 4.1 shows pseudocode of LEA.

Algorithm 4.1 Left edge algorithm

```
1: repeat
2:     node-set ← set of leaves in VCG
3:     wires ← set of node-set where wire not connected in HCG
4:     Putting wires
5:     Delete wires from HCG and VCG
6: until HCG and VCG are empty
```

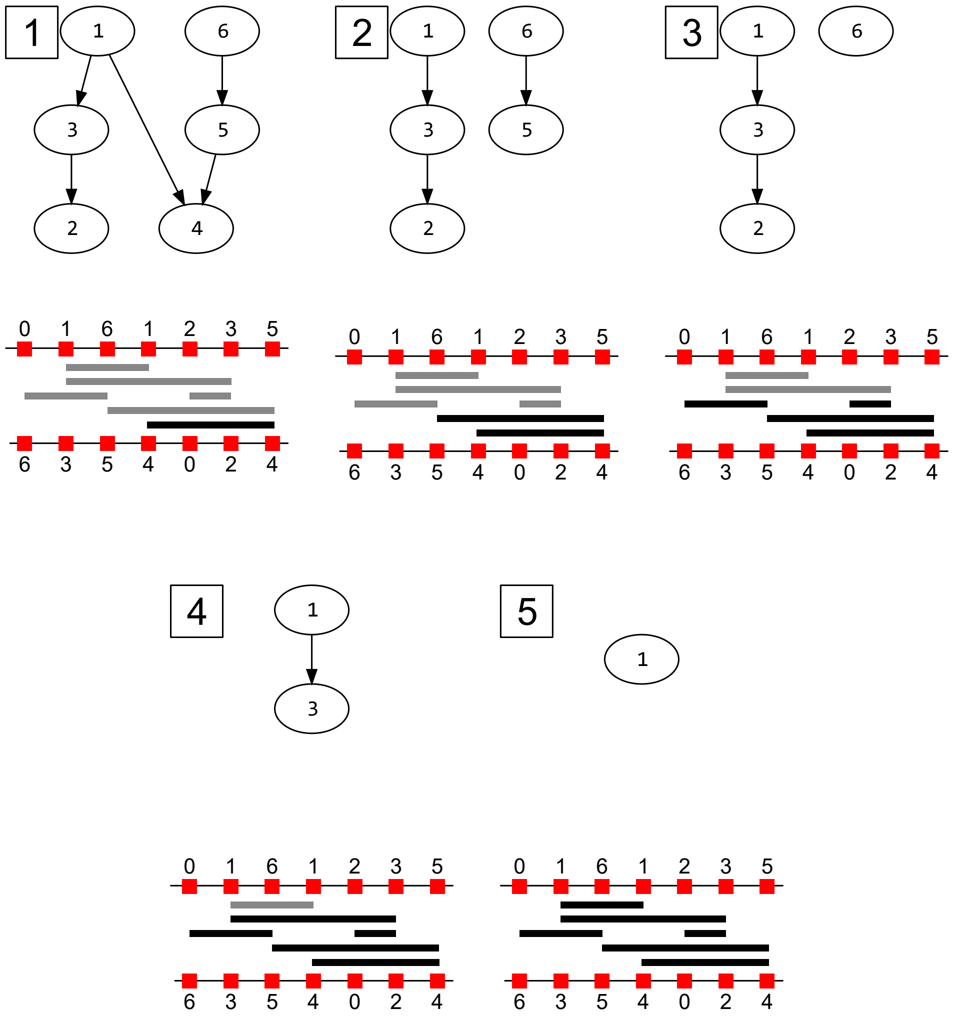
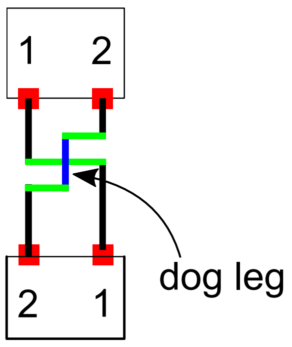
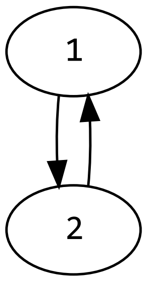


Figure 4.6: The left edge algorithm allocates the wires to the track, and when the VCG is empty, it means that the wire allocation has been completed.

There is a restriction in this algorithm that wiring cannot be performed unless the VCG is a directed acyclic graph (DAG). This means that even if the order of wiring placement is changed, there exists a situation where vertically oriented signal lines overlap as shown in figure 4.7. It is necessary to arrange the logic cells in such a way that this constraint is satisfied.





(a) (b)

Figure 4.7: Wiring example when VCG is not a DAG. (a) If such a combination of terminals is present in the circuit, the wiring will fail if the signal line bends only twice. Therefore, adding a structure called DOGLEG, which wires by bending four times. (b) The VCG generated from (a) has no leaves and cannot be wired because of the circulation.

In the AQFP circuit, the upper and lower cells are positioned at different positions, as shown in figure 4.8, to prevent the VCG from becoming a cyclic graph. The I/O pin spacing of the AQFP circuit is 20 µm, and the width of the signal lines is 10 µm, including the shield. By shifting the placement of the upper and lower cells by 10 µm, the vertical wiring of the upper and lower cells will not collide, which can be attributed to the wiring problem of no VCGs, and the left edge algorithm can always be used for wiring.

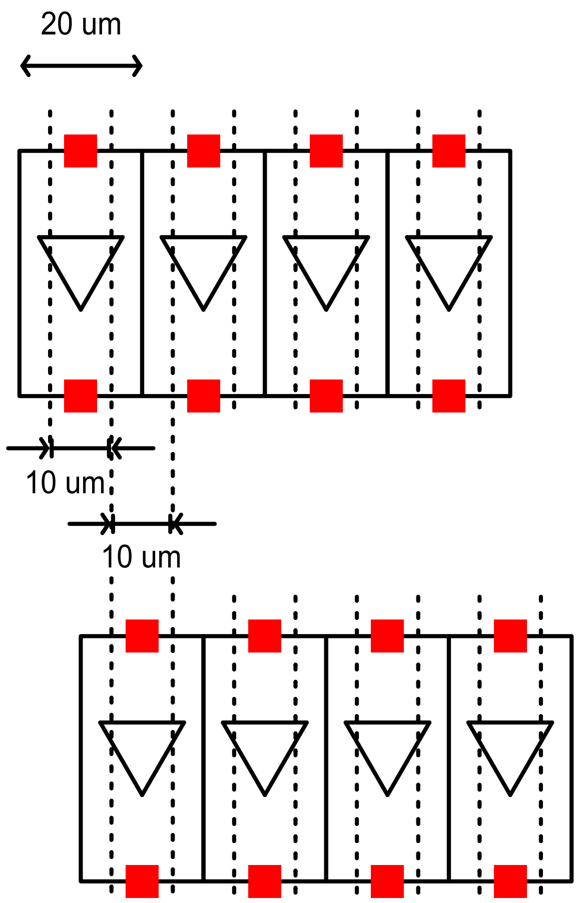


Figure 4.8: The method to avoid making cycle in VCG of AQFP circuit routing problem. By shifting the placement of the upper and lower cells by 10 µm, the vertical wiring of the upper and lower cells will not collide.

4.2 Channel routing optimization with Glitter

A stripline connects the AQFP gate and gate, and the strip line is part of the superconducting loop. Since the flowing current in the superconducting loop is inversely proportional to the inductance of the loop, the current flowing in the long strip line is small. The minimum sensitivity of the AQFP circuit designed for the HSTP process is about 7 µm, which is the current that flows when the parasitic inductance is 50 pH. If the inductance is higher than this, arithmetic errors due to thermal noise will occur frequently, so this is a rule that must be followed in the design. Where μ_0 is permeability in vacuum, w is width of signal wire, d_n is ditstance between signal wire to ground plane, and λ_n is London penetration depth.

$$L_n = \mu_0 \frac{(d_n + 2\lambda_L)\, l}{w} \tag{4.1}$$

As can be seen from euqation (4.1), the parasitic inductance is inversely proportional to the width. If the wiring width is wider, the inductance per length can be reduced, which means that more extended wiring is possible.

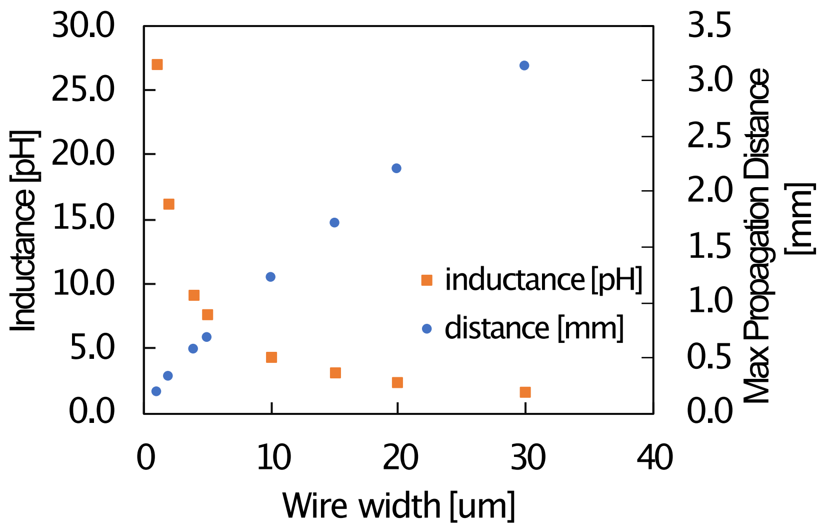


Figure 4.9: Parasitic inductance per 100 µm vs wire width. Inductance can be small when wider wire is used.

The figure shows the relationship between wiring width and inductance per length and between wiring width and maximum wiring length. 3 mm wiring is possible if 30 µm width wiring is used.

4.3 Glitter, variable-width channel router

Using wide signal lines can increase the data transmission distance, but it also increases the area of the circuit. If wide wiring can be used for long-distance signal lines and thin wiring for short distances, data can be propagated over long distances without increasing the circuit area. This technique can be achieved using Glitter[62], an algorithm that enables correct wiring in wiring areas where the wiring width is not constant.

4.3.1 Routing Algorithm

In this section, the Glitter algorithm is briefly explained. Glitter is an algorithm that is an extension of the left edge algorithm and generates a weighted constrain graph (WCG) by generating a weighted VCG and HCG and combining them. Figure 4.11 shows HCG and VCG corresponding to figure 4.10, and figure 4.12 shows WCG from figures 4.10 and 4.11. Since the generated WCG is a graph with a mixture of undirected and directed edges, the orientation of the undirected edges is calculated according to the algorithm to convert the WCG to a DAG. The direction is given such that the channel area is minimized.

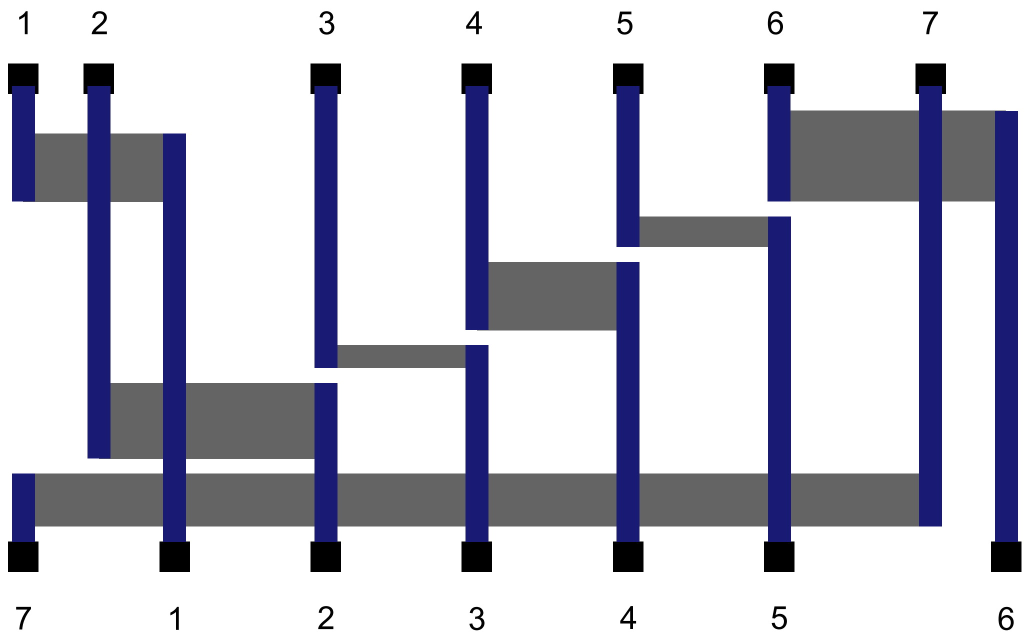
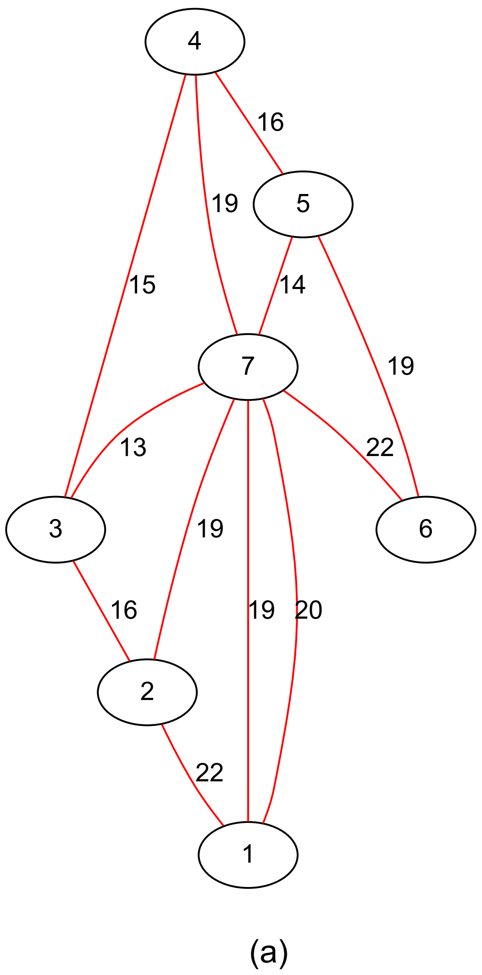


Figure 4.10: The example of routing problem for Glitter. Figures 4.11 and 4.12 is generated from this problem.



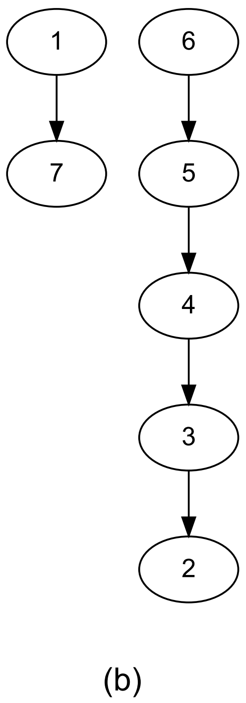
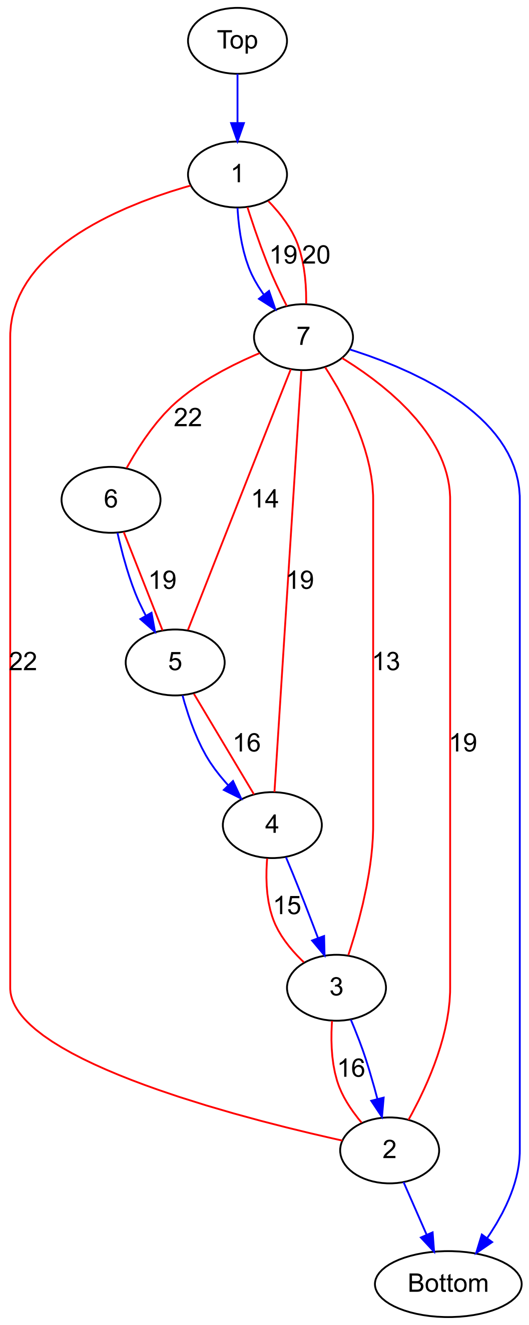
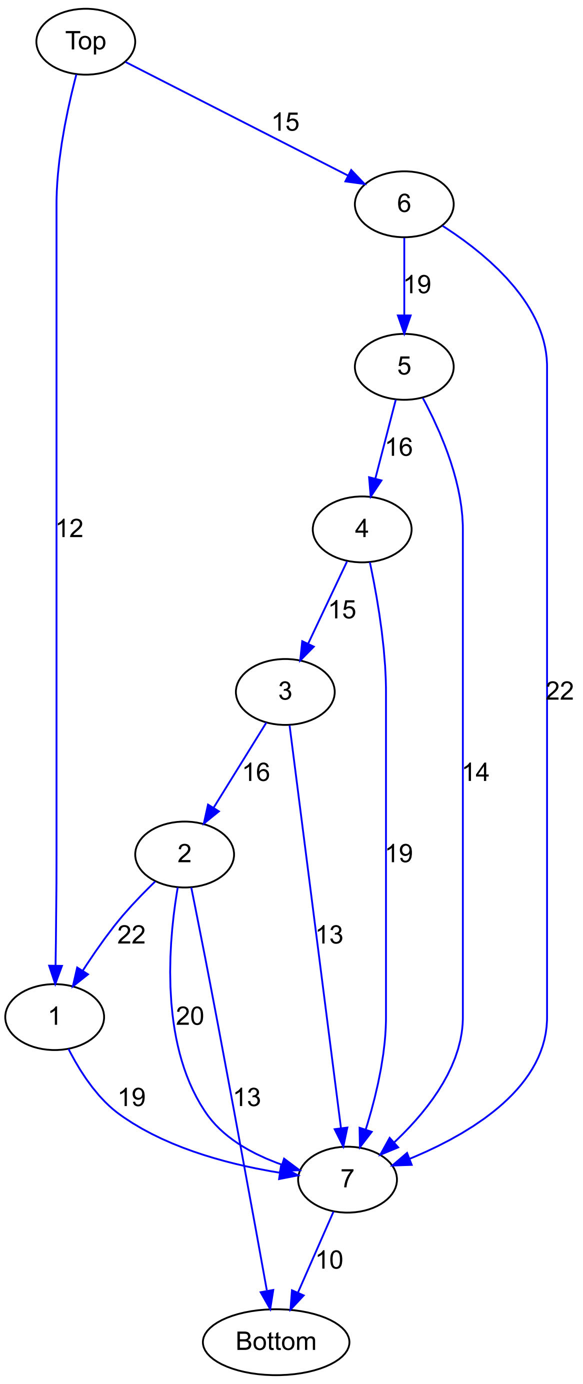


Figure 4.11: (a) HCG of routing problem of figure 4.10. (b) VGC of routing problem offigure 4.10.





(a) (b)

Figure 4.12: (a) Initial state of weighted constrain graph. This graph is generated by merging HCG and VCG of figure 4.11. (b) Final state for WCG. This is DAG. The wire is assigned from the edge with the smallest maximum weight from the top and bottom.

4.3.2 Flow to decide the width of wire

The algorithm for determining the wiring width is described below. The implementation is simple, since previous experimental results have shown that a simple algorithm is sufficient.

Repeat the four steps as shown below to reduce the inductance of the wiring in the channel to 50 pH or less.

1. Determine the required wiring width from the wiring length.
2. Using the Glitter algorithm, determine the wiring sequence. This determines the channel width.
3. Update the channel width. Recalculate the inductance of the wiring.
4. Check if all inductances in the channel are below the target value (50 pH). If confirmed, exit the program; otherwise, return to step 1.

When this algorithm is executed, wiring with shorter distances in the channel becomes thinner, and wiring with longer distances becomes wider than the standard width of AQFP wire. Since the channel length is the sum of the widths of each wire, as the number of thicker wires increases, the channel length also increases. The increase in channel length may cause positive feedback because the wires are incentivized to be thicker. Therefore, thicker wiring in the initial conditions can prevent positive feedback.

4.3.3 Test result

The proposed method was applied to a circuit with about 400 wires, and the results obtained are shown in table 4.1.

Table 4.1: Test result of inductance optimization by using Glitter channel router.

	LEA		Glitter	
#channel	Max inductance pH	# buffers	Max inductance pH	# buffers
1	81.8	44	46.6	0
2	77.4	40	43.6	0
3	94.6	48	41.1	0
4	82.4	42	48	0
5	151	174	71.9	58
6	110.1	78	48.5	0
7	97.3	26	47.5	0
8	79.8	18	46.7	0
9	78.3	10	47.5	0
10	67.8	8	46.4	0
11	41.2	0	41.2	0
Total		488		58

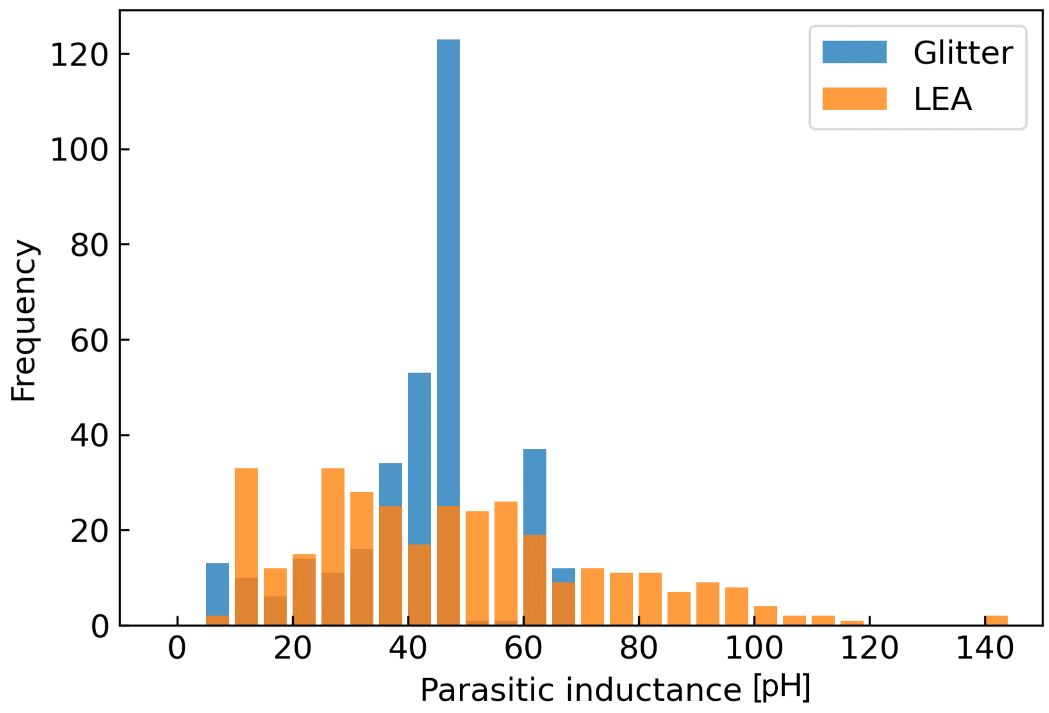


Figure 4.13: Hisgram of parasitic inductance of wire in the benchmark circuits. With the optimization, peaks are visible at less than the threshold value of 50 pH in Glitter, whereas the values were scattered in LEA.

4.4 Conclusion

In this chapter, the routing method of the AQFP circuit is described. First, the wiring structure in the HSTP process and the left edge algorithm, which is the basic algorithm, are described.

There is a strict limit to the wiring distance in AQFP circuits, and this limit is due to the parasitic inductance of the interconnections, which can be extended by using interconnections with low inductance per unit length, i.e., wide interconnections. However, increasing the wiring width leads to an increase in the circuit area, so the wiring algorithm using Glitter that flexibly determines the wiring width is proposed.

As the clock frequency increases, the AQFP gate and the characteristic impedance difference between the wiring and the AQFP gate may reflect due to the difference in impedance[63], which may interfere with signal propagation. Therefore, it is necessary to design wiring that includes a stripline filter and to calculate in advance combinations of wiring widths and distances that cannot be used.

Chapter 5

Ciruit design and test

In this chapter, the actual circuits designed will be discussed.

5.1 Adder(Kogge-Stone adder)

The 16-bit majority-3 KSA and the 8-bit majority-5 KSA were designed and demonstrated in operation. In addition, 4-, 8-, and 16-bit majority-3 and majority-5 KSAs were designed to evaluate their performance in simulation, and a comparison of their performance is also reported.

The adder is the fundamental operation in logic operations. The performance improvement of the adders has a significant impact on the entire system.

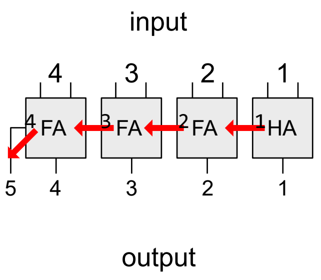
5.1.1 Architecture

There are various adder architectures, including ripple carry adder (RCA), Carry Save Adder (CSA), carry-lookahead adder(CLA), and carry prefix adder (CPA). The CSA and CLA are speed-up techniques for RCA and CPA. Figure 1 shows the architecture of RCA. (a) is a block diagram of the implementation method in CMOS circuits and RSFQ, and (b) is a block diagram of the implementation method in AQFP circuits. The RCA has a structure connecting all full adders' carry signals in sequence. The AQFP circuit cannot perform asynchronous operations, so data must be stored in first-in-first-out memory using buffer gates until the carry signal arrives. Since a large FIFO is required, the area and JJs are inefficient. When RCAs are implemented in an AQFP circuit, the latency is $O(N)$, and the JJs is $O(N^2)$.

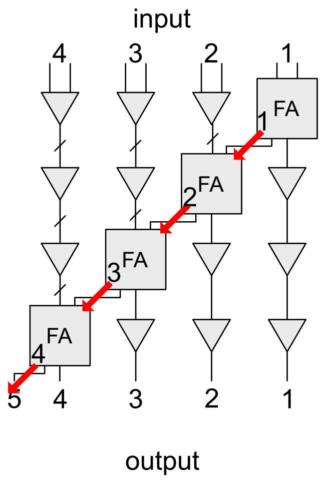
CPA is known as a faster adder than RCA. The following is an overview of how CPA works. Let the two n-bit operands be denoted as $A = (a_{n-1}, a_{n-2}, ...a_0)$ and $B = (b_{n-1}, b_{n-2}, ...b_0)$ and the sum of A and B as $S = (s_n, s_{n-1}, ...s_0)$. And to introduce intermediate answer p_i, g_i and it is defined as

$$\begin{aligned} g_i &= a_i \times b_i \\ p_i &= a_i \oplus b_i \end{aligned} \tag{5.1}$$

g_i equal to "1" indicates carry generates from i-th bit. p_i equal to "1" indicate carry generates if g_{i-1} equal to "1". And also, an operator "o" is introduced which



(a) Block diagram of RCA in CMOS and RSFQ circuits.



(b) Block diagram of RCA in AQFP circuit. A lot of buffer gates (FIFO memory) are needed, which were not shown in (a).

Figure 5.1: Block diagram of AQFP circuits.

works as following equations.

$$(g,p)o(g',p') = (g + g'p', pp') \tag{5.2}$$

This operation works for contentious sequences bit i to bit j and $(g_{i:j}, p_{i:j})$ indicate the result of operation "o" from i to j.

$$(g_{i:j}, p_{i:j}) = (g_i, p_i)o(g_{i-1}, p_{i-1})o...o(g_{j+1}, p_{j+1})o(g_j, p_j) \tag{5.3}$$

It is continuance "o" is also adaptable for $(g_{i:j}, p_{i:j})$.

$$(g_{i:k}, p_{i:k}) = (g_{i:j}, p_{i:j})o(g_{j-1:k}, p_{j-1:k}) \tag{5.4}$$

$g_{i-1:0}$ indicates carry will arrive bit-i th and finally result of addition can be derived by

$$s_i = g_{i-1:0} \oplus p_i \tag{5.5}$$

Finally, the summation result S can be derived by

$$S = (s_n, s_{n-1} \cdots s_1, s_0) = (g_n, g_{n-1:0} \oplus p_n, g_{n-2:0} \oplus p_{n-1}, g_{1:0} \oplus p_1, p_0) \tag{5.6}$$

By setting j to $j = (i + k)/2$ in euqation (5.4), the computation of $g_{i:j}$ and $p_{i:j}$ can be represented by a binary tree. Therefore, the computation time of this adder is $O(log_2 N)$, which is much faster than CSA.

There are many variations of the CSA, which differ in the number of fanouts, the number of wire lengths, and the number of logic gates used. AQFP circuits are suitable for adders with the Kogge-Stone structure, which are CSAs with fewer fanouts because of the low gate driving power.

figures 5.2 and 5.7 shows block diagram of majority-3 based and majority-5 based KSAs, respectively. There is a small difference in architecture, but the algorithm is the same.

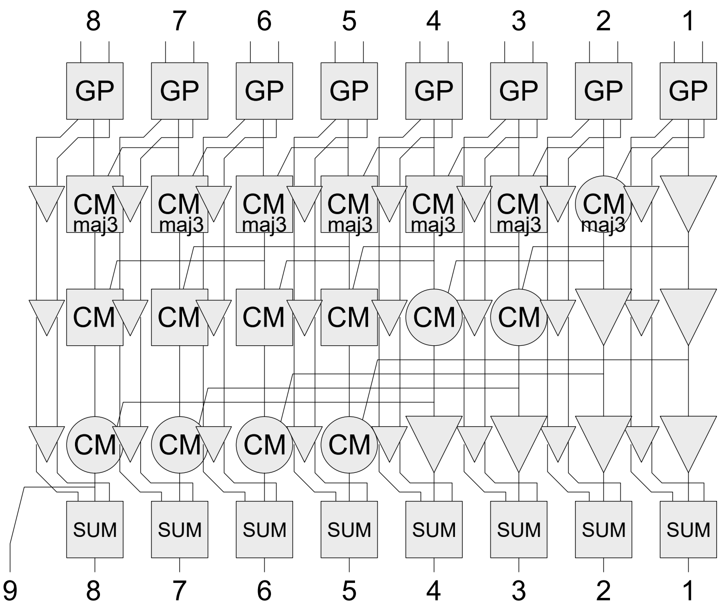


Figure 5.2: Block diagram of 8-bit majority-3 KSA. Logic inside of the circuit is shown in figure 5.3. To deduce latency, the SUM stage uses g_i, p_i, and a buffer chain is required to propagate these signals.

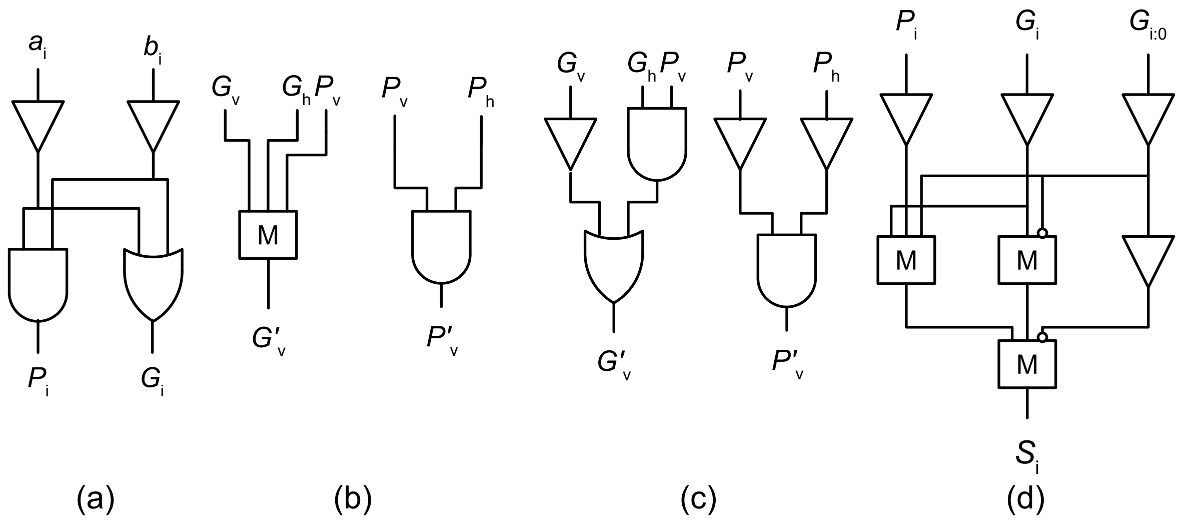


Figure 5.3: Inside of block diagram of majority-3 KSA. (a) GP block. (b)CM maj-3 block. (c) Standard CM block. (d) SUM block.

5.1.2 16-bit majority-3 based KSA

Figure 5.4 shows microphotograph of majority-3 based KSA. And, the circuit parameters are shown in the table 5.1. Figures 5.5 and 5.6 show measurement result.

We tried two sets of tests: critical tests and random tests. The critical tests were five test vectors that we explicitly defined to demonstrate pass-through of propagate signals, generation of carry from every bit, and full-propagation along the carry chain from the least significant bit (LSB) to the most significant bit (MSB). If the carry output is correct in the latter case, it is a very a good indication that the prefix signals are properly propagating along the long interconnects between stages, and that the carry prefix tree is working properly. Table 5.2 lists the critical test vectors we applied and the expected result. We then proceeded to apply a set of random vectors for a total of 110 random additions, as shown in Table 5.3.

figure 5.5 shows the results of the critical tests and 20 random tests, and figure 5.6 shows the results of 90 additional random tests at 100 kHz. The dc-SQUID-based output interface of the KSA produces a unipolar return-to-zero signal. When the output is a logic ‘1’, the output signal rises and then falls back to zero proportional to the applied clock period of the AC clock. When the output is a logic ‘0’, the output signal remains low. We confirmed that both the critical tests and the random tests passed. Even though our experiment was not fully exhaustive, these successful tests provide a strong indication that our design is working correctly.

Some of the outputs were rather noisy with S14, in particular, being very unstable. We attribute this to the measurement equipment condition at the time of this writing. Furthermore, we measured the operating margins of the circuit by adjusting how much we can change the amplitudes of AC1 and AC2 before the circuit malfunctions. The operating range is -4.29 dBm to 0.90 dBm for AC1, and -3.41 dBm to 1.02 dBm for AC2. Both ranges are sufficiently wide.

Table 5.1: Specification of majority-3 based KSA

X mm	2.8
Y mm	3.6
Latency cycle	8.5
JJs	4976

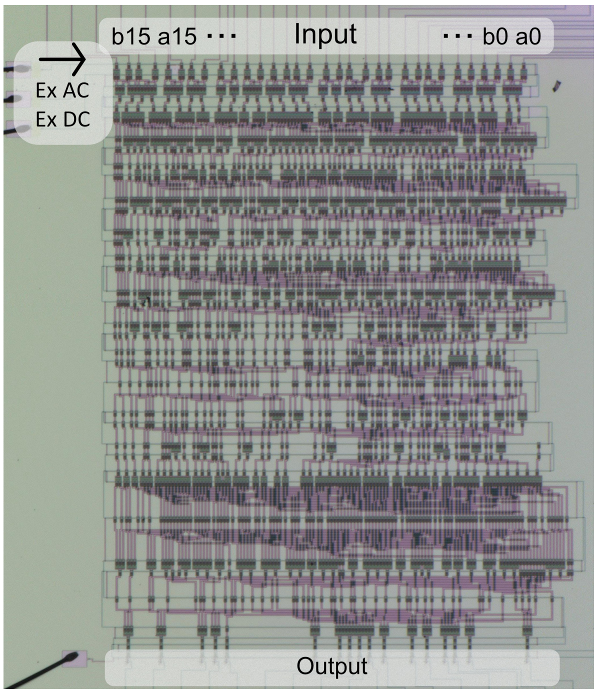


Figure 5.4: Microphoto graph of 16-bit majority-3 based KSA.

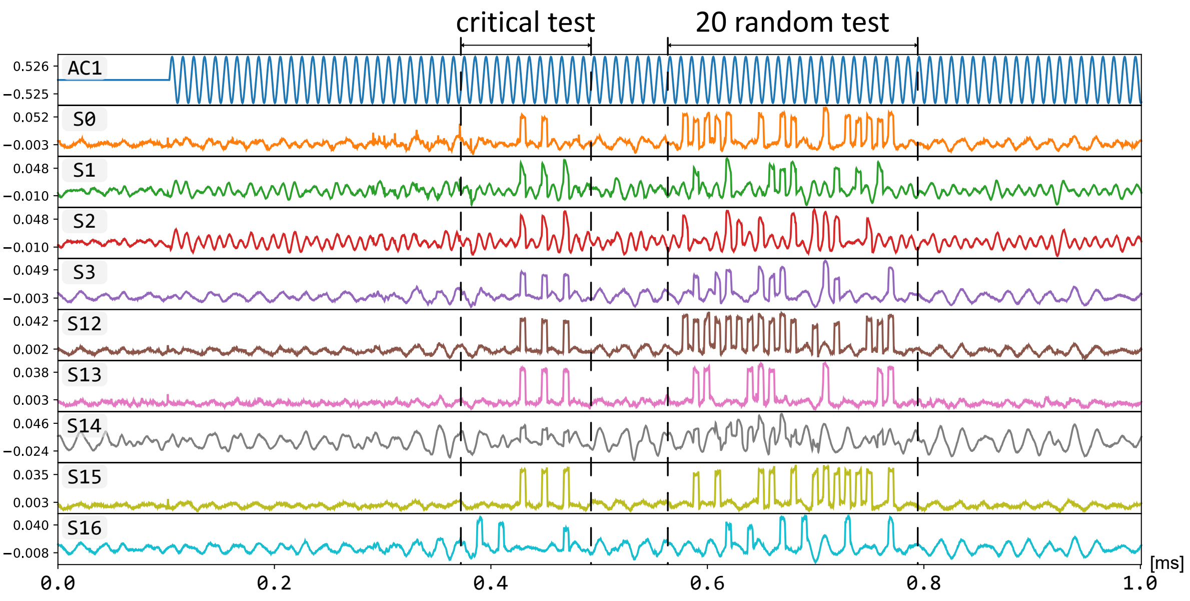


Figure 5.5: Measurement waveform of the 16-bit KSA for 5 critical test vectors shown in Table 5.2 and the first 20 random test vectors shown in Table 5.3. The signals from top to bottom are the excitation current, output bits 0 (LSB) to 3, bits 12-to-15, and the output of the carry (MSB). Critical tests include test cases in which the carry moves from the LSB to the MSB. All tests cases have been validated.

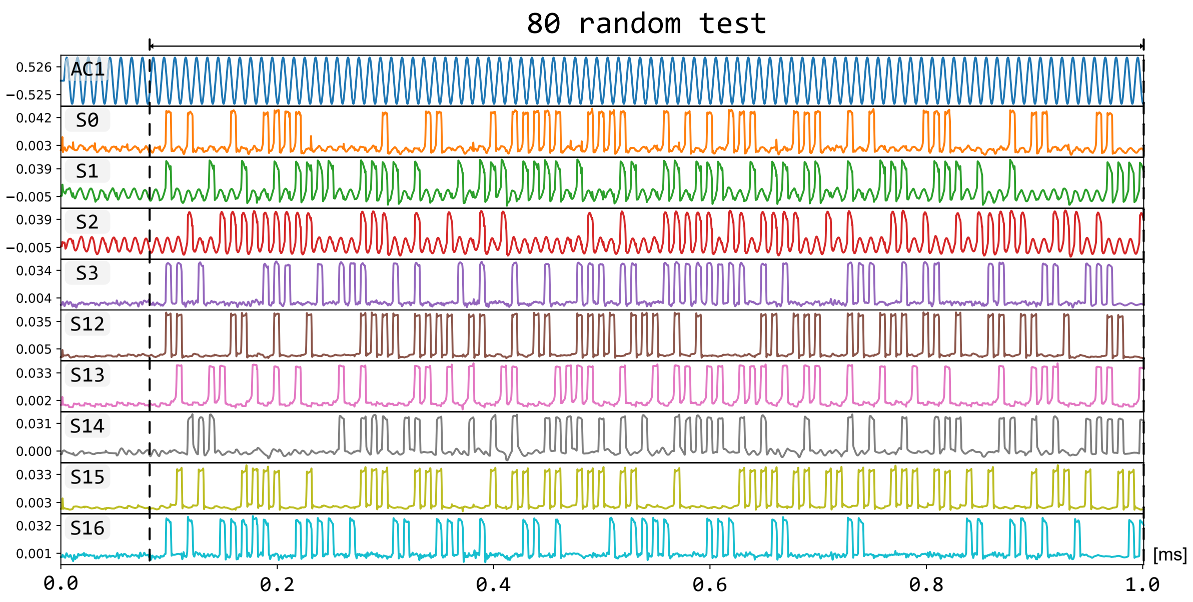


Figure 5.6: Measurement waveform of the 16-bit KSA showing 90 random test vectors listed as 21 through 110 in Table 5.3. The signals from top to bottom are the excitation current, output bits 0 (LSB) to 3, bits 12 to 15, and the output of the carry (MSB). All observable outputs have been validated. The waveform traces have been averaged to show a clearer picture.

Table 5.2: List of critical test vectors applied in experiment.

Op. A	Op. B	Result	Description
0xFFFF	0x0001	0x10000	Propagate carry through all bits
0x0001	0xFFFF	0x10000	Propagate carry through all bits
0xFFFF	0x0000	0x0FFFF	Operand A passthrough
0x0000	0xFFFF	0x0FFFF	Operand B passthrough
0xFFFF	0xFFFF	0x1FFFE	Generate carry at all bits

Table 5.3: Full list of random test vectors applied in experiment.

#	Op. A	Op. B	Result	#	Op. A	Op. B	Result
1	0x00E4A	0x0075B	0x015A5	56	0x06C45	0x068A6	0x0D4EB
2	0x0349A	0x0BDB1	0x0F24B	57	0x0F8A2	0x0F681	0x1EF23
3	0x03291	0x005B0	0x03841	58	0x01514	0x0502C	0x06540
4	0x0AF9C	0x0234D	0x0D2E9	59	0x0FC3D	0x0004D	0x0FC8A
5	0x0D6CC	0x07EB3	0x1557F	60	0x001D2	0x0BBBB	0x0BD8D
6	0x03EDA	0x0120A	0x050E4	61	0x0A57A	0x037FF	0x0DD79
7	0x085FA	0x0F866	0x17E60	62	0x0D00E	0x04853	0x11861
8	0x078A4	0x08059	0x0F8FD	63	0x0D35E	0x019E1	0x0ED3F
9	0x0CA13	0x03397	0x0FDAA	64	0x0D4C9	0x0CA61	0x19F2A
10	0x0E972	0x075A9	0x15F1B	65	0x084AB	0x0D785	0x15C30
11	0x06B60	0x07107	0x0DC67	66	0x0BF4A	0x070B6	0x13000
12	0x04F84	0x0C01C	0x10FA0	67	0x093DC	0x073A3	0x1077F
13	0x0A2E6	0x03A5E	0x0DD44	68	0x0113F	0x0E1FF	0x0F33E
14	0x08C19	0x019D4	0x0A5ED	69	0x02164	0x040F9	0x0625D
15	0x01B0A	0x07944	0x0944E	70	0x0040F	0x0584F	0x05C5E
16	0x0B3FC	0x0CE35	0x18231	71	0x0BDF2	0x0A60D	0x163FF
17	0x060C3	0x02B20	0x08BE3	72	0x0A42F	0x0C89B	0x16CCA
18	0x04FEC	0x04CE9	0x09CD5	73	0x07B1D	0x08CA0	0x107BD
19	0x06661	0x013C2	0x07A23	74	0x0C6D8	0x025C5	0x0EC9D
20	0x0B11F	0x0FF4A	0x1B069	75	0x020D9	0x0876A	0x0A843
21	0x0B380	0x06BFB	0x11F7B	76	0x00BCD	0x0F39F	0x0FF6C
22	0x07C9D	0x03E3B	0x0BAD8	77	0x0BADB	0x0E50B	0x19FE6
23	0x0C93A	0x0836B	0x14CA5	78	0x03CC6	0x0EE08	0x12ACE
24	0x02204	0x0A274	0x0C478	79	0x06D36	0x0312F	0x09E65
25	0x02DEC	0x03876	0x06662	80	0x02DC6	0x0C74D	0x0F513
26	0x06D5B	0x0B819	0x12574	81	0x0335F	0x00204	0x03563
27	0x0F5DB	0x025BA	0x11B95	82	0x012A9	0x076DB	0x08984
28	0x09FE5	0x0F771	0x19756	83	0x069B4	0x01A8C	0x08440
29	0x0E74F	0x0C095	0x1A7E4	84	0x08984	0x0EFEB	0x1796F
30	0x0F4A2	0x09B2B	0x18FCD	85	0x0A5DF	0x0F479	0x19A58
31	0x06230	0x0579F	0x0B9CF	86	0x04F7B	0x0795E	0x0C8D9
32	0x00974	0x00209	0x00B7D	87	0x00210	0x02F62	0x03172
33	0x0C993	0x05F24	0x128B7	88	0x03748	0x0676E	0x09EB6
34	0x0EAB9	0x0B0ED	0x19BA6	89	0x0BDD1	0x013DD	0x0D1AE
35	0x0E074	0x02F86	0x10FFA	90	0x0315A	0x07E86	0x0AFE0
36	0x082E9	0x08309	0x105F2	91	0x0114D	0x00A20	0x01B6D
37	0x063A3	0x004B5	0x06858	92	0x06D6A	0x06551	0x0D2BB
38	0x01895	0x0F683	0x10F18	93	0x02259	0x0BE0A	0x0E063
39	0x03590	0x0C2BE	0x0F84E	94	0x02862	0x02DF4	0x05656
40	0x069C9	0x06D0D	0x0D6D6	95	0x08B3E	0x0FF92	0x18AD0
41	0x0267C	0x0B06B	0x0D6E7	96	0x032AF	0x0CD67	0x10016
42	0x0C602	0x05098	0x1169A	97	0x05C8B	0x042A1	0x09F2C
43	0x0D320	0x06D40	0x14060	98	0x03A3A	0x03AD2	0x0750C
44	0x095FB	0x06603	0x0FBFE	99	0x0CE5D	0x0742A	0x14287
45	0x0513A	0x06B27	0x0BC61	100	0x06619	0x0AFEB	0x11604
46	0x0DD7A	0x0FBD7	0x1D951	101	0x0987D	0x06624	0x0FEA1
47	0x03FDC	0x0F748	0x13724	102	0x0D2C9	0x052C0	0x12589
48	0x0F5B9	0x01601	0x10BBA	103	0x0D7BB	0x00CC1	0x0E47C
49	0x03DF8	0x03EA8	0x07CA0	104	0x00D6F	0x08F55	0x09CC4
50	0x04741	0x0CA1D	0x1115E	105	0x0F4F7	0x00D7D	0x10274
51	0x06063	0x06A60	0x0CAC3	106	0x07391	0x05937	0x0CCC8
52	0x00242	0x02884	0x02AC6	107	0x00B7B	0x059C2	0x0653D
53	0x02BD4	0x0CC85	0x0F859	108	0x032EA	0x04471	0x0775B
54	0x0A4C7	0x0EB0C	0x18FD3	109	0x02DB1	0x0A351	0x0D102
55	0x04E1B	0x0C9B8	0x117D3	110	0x0F361	0x091C1	0x18522

The first 20 vectors in this list correspond to the random tests in 5.6, and the last 90 vectors correspond to 5.5.

5.1.3 8-bit majority-5 based KSA

To further speed up the AQFP adder, a KSA with a 5-input majority logic gate is proposed. Increasing the bit width increases latency due to the increase in the number of CM gate layers. In other words, if the latency of the CM gate can be reduced, a scalable adder can be created. $g + g \times p$ is an operation of the CM gate, which can be calculated with a single 5-input majority logic gate, as shown in euqation (5.9).

Figure 5.7 shows block diagram of AQPF majortiy-5 based KSA. This structure is very similar to designing a KSA in a CMOS circuit. And, figure 5.8 shows inside of each block. The majority- 5 is used in CM block. table 5.4 shows specification of 8-bit majority-5 KSA.

$$\begin{aligned} Maj(a,b,c,c,1) &= ab + abc + ab1 + acc + ac1 + ac1 + bcc + bc1 + bc1 + cc1 && (5.7)\\ &= abc + ab + ac + bc + c && (5.8)\\ &= ab + c && (5.9) \end{aligned}$$

Table 5.4: Specification of majority-5 based KSA

X mm	1.6
Y mm	1.3
Latency cycle	4.25
JJs	1212

The designed KSA was evaluated in liquid helium using an immersion probe. The circuit was run at a low frequency of 100 kHz to verify that the operations were correct and evaluated using 47 test vectors, including random inputs. section 5.1.3 shows the observed waveform of the test circuit. The readout of the AQFP circuit uses a dc-SQUID driven by the same sinusoidal wave as the one feeding the circuit. The output is a return to zero(RTZ) waveform.

In addition, the margin of AC1 and AC2 is -23.6 % to 21.4 % and -5.6 % and 21.1 %, respectively. This margin range is relatively narrow, but it is a typical value.

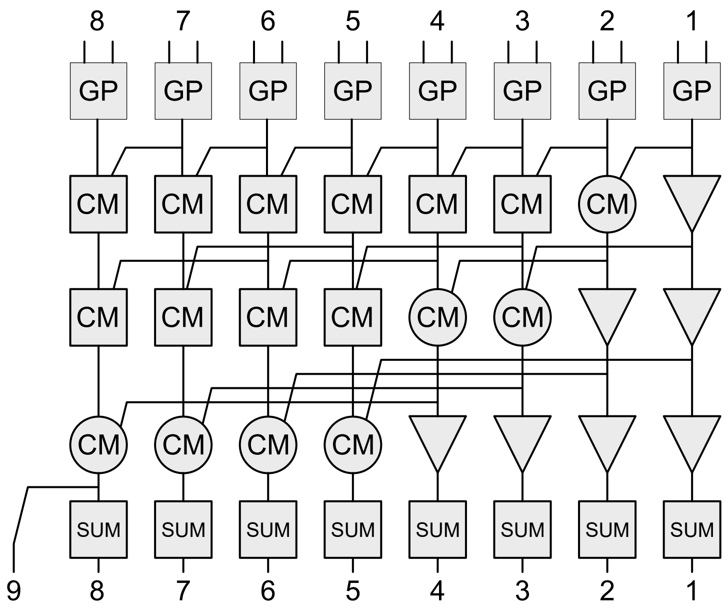


Figure 5.7: Block diagram of 8-bit majority-5 KSA. Logic inside of the circuit is shown in figure 5.8.

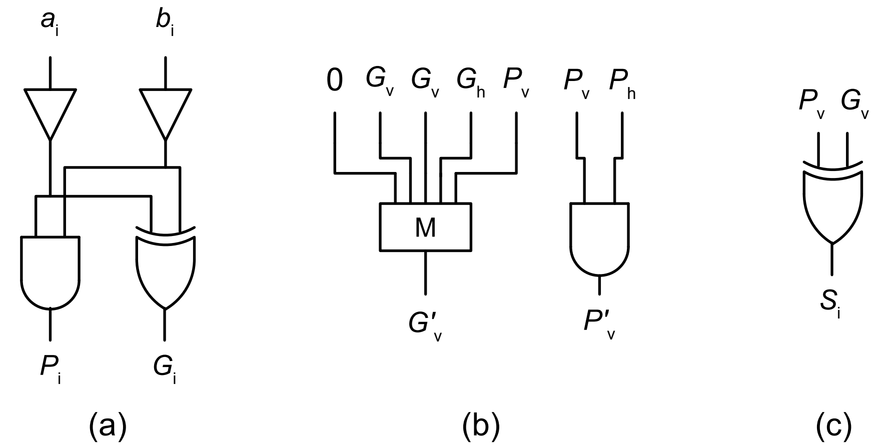


Figure 5.8: Inside of block diagram of majority-5 KSA. (a) GP block. (b) CM block. (c) SUM block. Compared with majority-3 based KSA,By using 5-input majority gate, latency of CM block becomes two to one. To privent using buffer chain to propagate G_i, P_i to SUM block, XOR gate is used GP and SUM block.

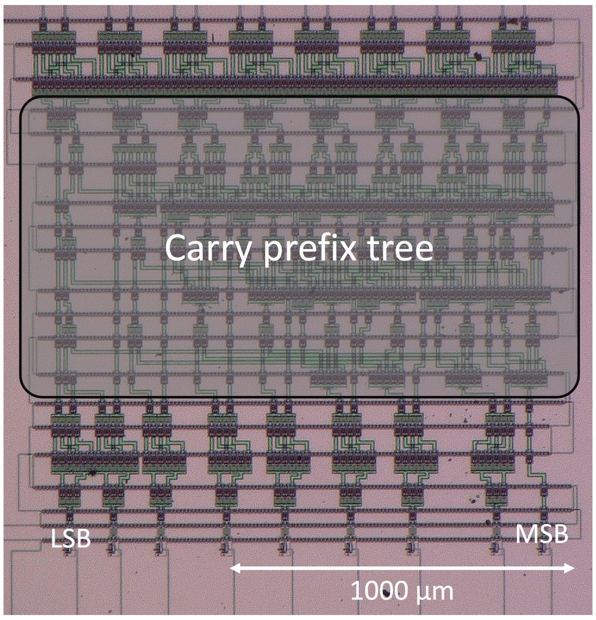


Figure 5.9: Microphoto graph of 8-bit majority-5 based KSA.

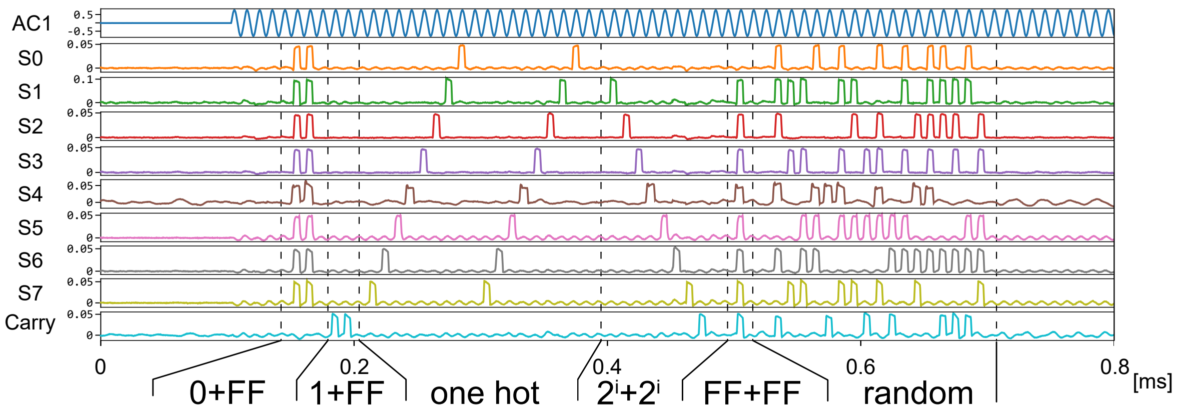


Figure 5.10: Measurement waveform of majority-5 based 8-bit KSA.Passed the critical test and the random test test.

5.1.4 Discussion

Adder performance compared with another circuit techonlogy

Table 5.5 shows performance comparison of adder with several parallel adders under various technologies including superconductor RSFQ/ERSFQ [64], RQL [65], and 90-nm adiabatic CMOS [66]. We decided to compare RSFQ/ERSFQ as it is the most widespread logic family in superconductor electronics. RQL is included to this table as well even though the reported design is only 8-bit because it is an AC-biased logic just like AQFP, and it is also considered a very energy-efficient technology. The 90-nm adiabatic CMOS work reported in [66] is also in the table, because even though the designs have not been experimentally demonstrated, it is one of the more recent works using a 90 nm fabrication process to manufacture devices that operate adiabatically like AQFP. The area, number of Josephson junctions (JJs) when applicable, bias magnitude, target operating frequency, and energy/op are compared.

Firstly, the footprint area of the adder in this study is larger than that of the RSFQ implementation. Compared to RQL, the area of the AQFP circuit is also larger. The primary cause for this is the large output transformers of each AQFP. Section 5.1.4 briefly discusses how this can be improved. The adiabatic CMOS adder occupies even less area despite using a relatively old 90 nm CMOS technology. We expect this difference to grow when considering 7 nm FinFET technology [67]. In a strict discussion, the AQFP circuit may be smaller in area. The AQFP circuit is designed by the HSTP process and has four metal layers. In contrast, the RSFQ circuit used ADP process, which has eight metal layers. Since the wiring in the AQFP circuit occupies more than half of the area, the area can be reduced if it uses a process with more layers. As an optimistic estimate, if we assume that the area can be reduced by about 30%, the area of the circuit will be 7 mm^2, which is smaller than the RSFQ circuit.

In terms of the number of JJs, this study is almost half the number of junctions used in the RSFQ implementation. This may be due to the fact that RSFQ circuits are designed for very high-speed operation, so many JJs are inserted to adjust the delay of data propagation. Furthermore, the clock and reset paths of the wave-pipelined RSFQ adder also require active JJs, whereas the AQFP power-clock acts as both the synchronizing clock and reset. This power-clock is distributed through a meandering microstripline and thus requires no active JJs. RQL, like AQFP, distributes power through AC power-clocks along microstriplines, so it also has a low number of JJs. In our work, 3000 JJs are used for the amplification of signal currents, so there is potential to reduce the number of JJs to around the levels of RQL through more intelligent cell placement and interconnect routing.

The target operating frequency of RSFQ is higher than our AQFP implementation as we need to operate our circuit at relatively low clock rates to remain in the adiabatic regime. Despite this, the AQFP adder is still faster than adiabatic CMOS. When the operating frequency of the adiabatic CMOS adder is increased, the dynamic switching energy becomes more dominant compared to the static power consumption, making it difficult for semiconductor technology to make significant improvements in operating frequency while still operating adiabatically.

Table 5.5: Comparison with other 16-bit parallel adders in literature.

Metric	AQFP (this work)	RSFQ [64]	RQL[c][65]	Adiabatic CMOS [66]
Area	10.1 mm^2	8.5 mm^2	2.8 mm^2	0.0048 mm^2
Complexity	4976 JJs	9941 JJs	815 JJs	972 Trs
Bias	AC 3.0 mA	DC 1.61 A	AC 0.9 mA	DC 1 V
Target Freq.	5 GHz	30 GHz	10 GHz	0.5 GHz
Energy/op	6.97 fJ[a]	3.13 pJ[ab]	90.2 fJ[a]	182 fJ[d]

[a] Includes 1000 W/W cooling efficiency.
[b] Design was originally in RSFQ but we assumed ERSFQ biasing.
[c] Note that this is an 8-bit RQL adder, not 16-bit.
[d] Extrapolated from energy per transistor of shift register in [66].

An important point to emphasize is the energy/op metric. The power consumption in this study was estimated by the product of the power consumption per JJ [5] at the target clock frequency of 5 GHz by the number of JJs in our circuit. For all superconductor circuits in Table 5.5, we also took into account the cooling overhead using a 1000 W/W coefficient [3]. Since the static power consumption of RSFQ is too high, we assumed that the bias network of the design can adopt the ERSFQ approach where bias resistors are replaced by current-feeding JJs for this metric. The power consumption P of ERSFQ is obtained from $P = I \times \Phi_0 \times F$, where I, Φ_0, and F are the bias current of the circuit, the quantum flux, and the operating frequency, respectively [68]. For adiabatic CMOS, a 16-bit KSA design was reported in [66], but its power consumption was not mentioned. Based on the dissipated energy for a 3-bit shift register reported in that work, we assumed a linear extrapolation of the energy/op of the CMOS-based adiabatic KSA in Table 5.5. Our AQFP adder requires five hundred times less power than the ERSFQ implementation, at least fifteen times less power than the RQL implementation, and thirty times less power than the adiabatic CMOS implementation. In the case of semiconductor technology, even 28 nm CMOS does not show substantial improvement over 90 nm technology for adiabatic circuits because the static leakage power becomes more prevalent in the adiabatic operation region as reported in [66]. Considering that our design is more energy-efficient and still operates at a practical clock rate for high-performance computing, this makes AQFP logic a good candidate for building the next generation of low-power supercomputers and data centers.

Performace comparison between majortiy-3 and majority-5 KSA

Table 5.6 shows comparison about specificaton of majority-3 based and majority-5 based KSA. The circuit was redesigned from scratch for an accurate comparison. First, as expected, the majority-5-based KSA had shorter latency. The results showed that when the bit width increased, the reduction also increased. This trend is because the reduction in CM gate latency reduced the latency that depends on the bit width. Also, the number of buffer gates was lower for the majority-5-based KSA than for the majority-3-based KSA. This reduction is because the circuit's latency is reduced, and using XOR inside the GP block reduces the amount of FIFO memory. The majority-5-based KSA has more spl3L, i.e., branch cells with high energy consumption.

figure 5.11 shows a comparison of energy consumption. Averaged power consumption is used for this figure since the energy consumption of the AQFP circuit varies depending on the input signal.

The majority-5-based KSA consumed more energy than the majority-3-based KSA at all bit widths and frequencies. Although there should be a strong correlation between the number of Josephson junctions and energy consumption, the results were inconsistent. This inconsistency suggests that the number of spl3Ls (branching cells with high energy consumption) significantly impacts the energy consumption. figure 5.13 shows the normalized energy consumption of the majority-5-based KSA by the energy consumption of the majority-3-based KSA. When the ratio is greater than 1, the energy consumed by majority-5-based-KSA is greater than that of the majority-3-based-KSA. As the bit width increases, the curve approaches 1, suggesting that the majority-5-based KSA consumes less energy for large adders. In addition, it was shown that the method of estimating energy consumption by the number of Josephson junctions, commonly used in RSFQ circuits, cannot be used in AQFP circuits.

Table 5.6: Cell statistic of each KSA

	bit	bfr	and/or[a]	maj3	maj5	spl2[b]	spl3L[c]	phase	total JJs
majority 3	4	71	14	15	0	32	4	12	388
majority 3	8	237	46	31	0	84	8	16	1120
majority 3	16	736	134	63	0	212	16	21	3110
majority 5	4	51	30	0	5	25	8	12	398
majority 5	8	126	66	0	17	77	16	15	1004
majority 5	16	402	146	0	49	213	32	19	2660

[a] Energy consumption is equal to maj3
[b] 2-branchiing gate with buffer gate
[c] 3-branching gate with bufferL gate

Scaling of the AQFP KSA

We estimate how the adder scales in terms of area and latency when we increase the data word size as shown in figure 5.14. Based on these estimates, we discuss where AQFP can use improvement and how such improvements can be carried out. For the estimation, we assumed a simple structure where the bit slices of KSAs are connected directly below each other, and we also considered the insertion of buffers for long-distance wiring. For example, in the final stage of a 128-bit KSA, the data travels a distance of 64-bitslices. In this case, at least 13 levels buffer insertions will be required. To estimate the area, we calculated the ratio of the area used by the gate to the area used by the internal wiring and excitation lines for the 16-bit KSA we designed in this work, and estimated the area based on that. The number of buffer insertions increases in proportion to the square of the number of bits in the adder. Therefore, the slope of the curve with respect to area becomes a little larger. Compared to the

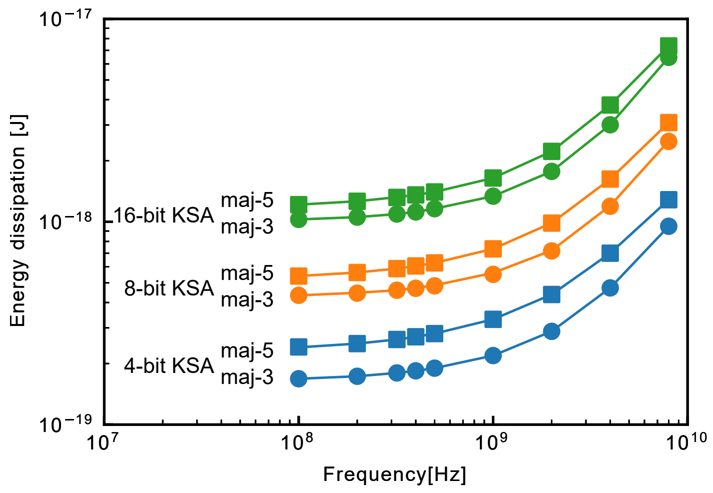


Figure 5.11: Comparison of power consumption of AQFP majority-5 based and majority-3 based KSA. 4, 8, and 16-bit KSAs are shown. We evaluated energy 200 times with different input combinations and averaged it.

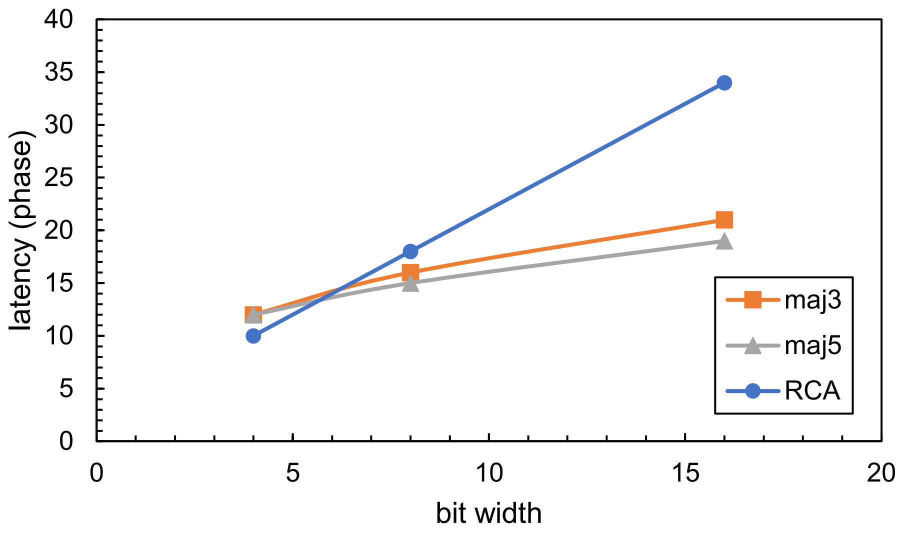


Figure 5.12: Latencty comparison of majority-5 based and majority-4 based KSA. As expected, majority-5 KSA is faster than majority-3 based KSA. In addition, the gap of latency larger when larger bit width.

previous study, the scalability is slightly improved because the gate area is smaller. From these estimates, it was found that, for example, a 64-bit KSA circuit can be implemented on a 1 cm^2 chip. Compared to the CMOS circuits that are widely used today, this area is very large, and efforts are needed to reduce the area. One approach is using a directly coupled quantum-flux-parametron (DQFP) [69, 70] where the large output transformer is completely removed, and logic gates connect directly to each other. This can reduce the cell size in half but unlike conventional AQFP cells, the DQFP buffer and inverter have different core structures, introducing complexities in

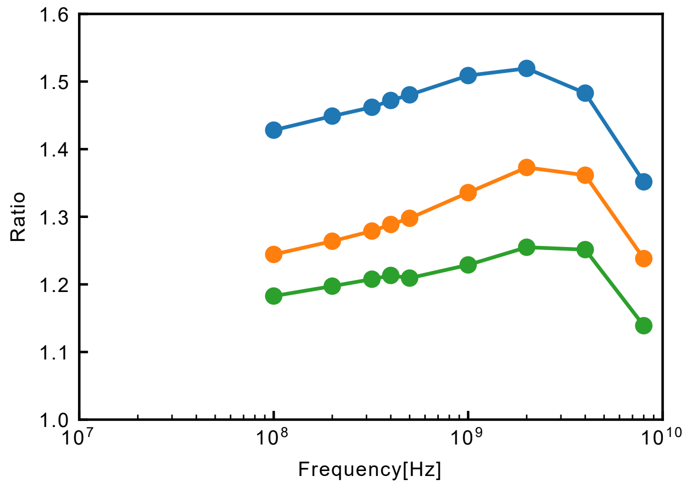


Figure 5.13: Graph of energy consumption of maj5 KSA normalized by energy consumption of mah-3 KSA of the same bit width. As the bit width increases, the difference becomes smaller.

cell library development. Another approach is to use a more advanced process with more layers [43, 71, 72]. With more layers, the transformer and the SQUID can be stacked vertically, roughly halving the footprint of the logic cell. Interconnections of the AQFP can also be stacked, potentially reducing the area further. The DQFP approach can also benefit from more advanced processes since the directly coupled inductor structures can be implemented using dedicated kinetic inductance layers such as those in the MIT Lincoln Laboratory SFQ5ee process [73], which opens more opportunities for further scaling.

The increase in latency for KSA should be logarithmic, but according to our estimates, it is increasing a little bit faster than expected. This is due to our conservative design approach of adding buffers after each logic stage in addition to buffer insertions for long interconnects. In any case, the absolute latency is quite high. We can improve the latency by reducing the phase difference of the excitation current or, in other words, adding more clock phases within the same target clock period. This can be achieved through power clock dividers or delay line clocking [56, 57, 25]. The latency improvement between these techniques is shown in the lower sub-plot of figure 5.14. In the present 4-phase design, the phase difference is 90°, but by using the aforementioned methods, the phase difference can be reduced up to 1/5 of the present design. This means more clock phases become available per cycle and thus data can propagate through more stages of logic in a cycle. Furthermore, we still use the same clock frequency in these methods so the switching energy is unchanged.

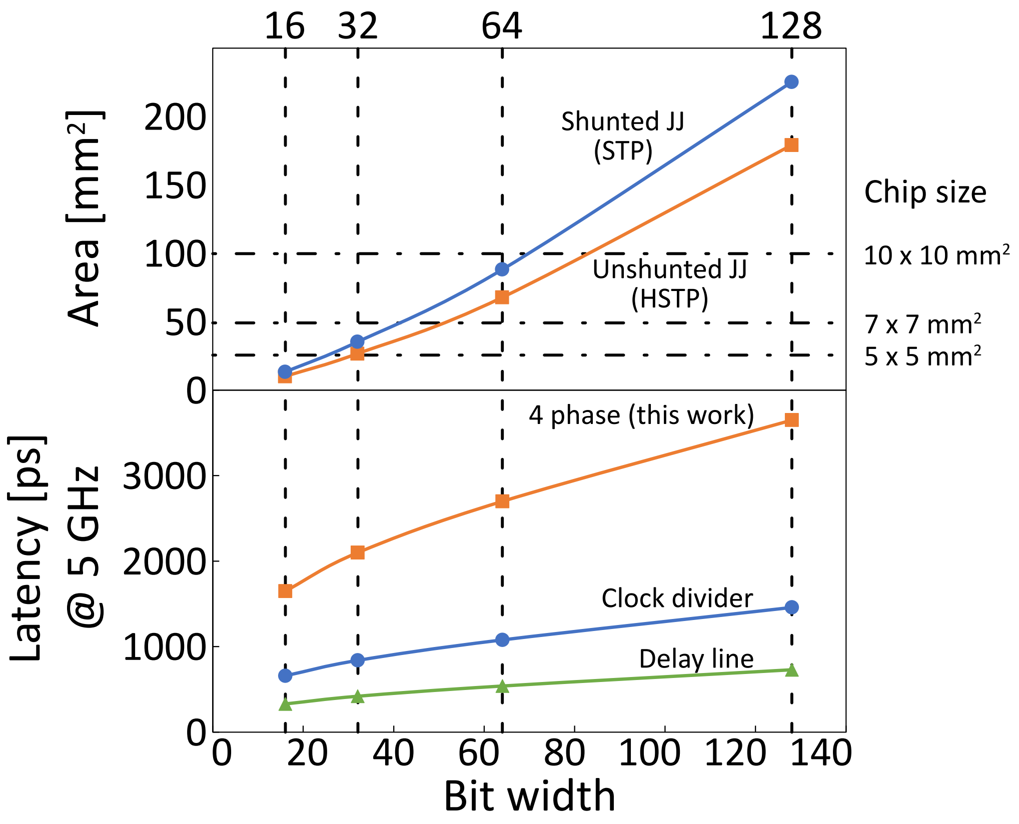


Figure 5.14: Word size scaling of the AQFP KSA for area and latency.

5.2 bfloat 16 floating point adder

For more practical applications, a floating-point adder was designed. Floating-point arithmetic is often used in scientific calculations and machine learning and is characterized by the wider range of values it can represent than integer arithmetic. Floating-point adders are larger in scale than integer adders, making it challenging to implement them in AQFP circuits. Recently, however, a 16-bit wide floating-point format called bfloat-16 has come into use in the field of machine learning. Therefore, we designed a 16-bit floating-point adder to solve the energy consumption problem in machine learning and to be useful as a benchmark to compare the performance with CMOS circuits. Unfortunately, the area of the designed circuit was too large to manufacture.

First, we explain the floating-point format. Widely used is the IEEE 754 single precision format, which consists of a sign part, an exponent part, and a mantissa part. When each parts are denoted as i, j, and k, the value represented by a bit string is calculated by the following equation.

$$(-1)^i \times 2^{(j-127)} \times \left(1 + \left(\frac{k}{2^{-23}}\right)\right) \tag{5.10}$$

By separating the exponent and value, floating-point arithmetic represents a wider range of values than integers with the same number of bits. 32-bit floating-point is overly large to implement in an AQFP circuit. Therefore, we decided to implement a floating-point adder in bfloat-16 format, where the length of the mantissa part is 7 bits. The largest component in floating-point addition is the adder of the mantissa part. The area of the circuit can be significantly reduced by reducing the length of the mantissa to 7 bits.

Figure 5.15 shows block diagram of bfloat-16 adder. It consists of an adder, a subtracter, a shift circuit, a priority encoder, and a comparator. For adder and subtracter, KSA is used written in previous section, section 5.1.

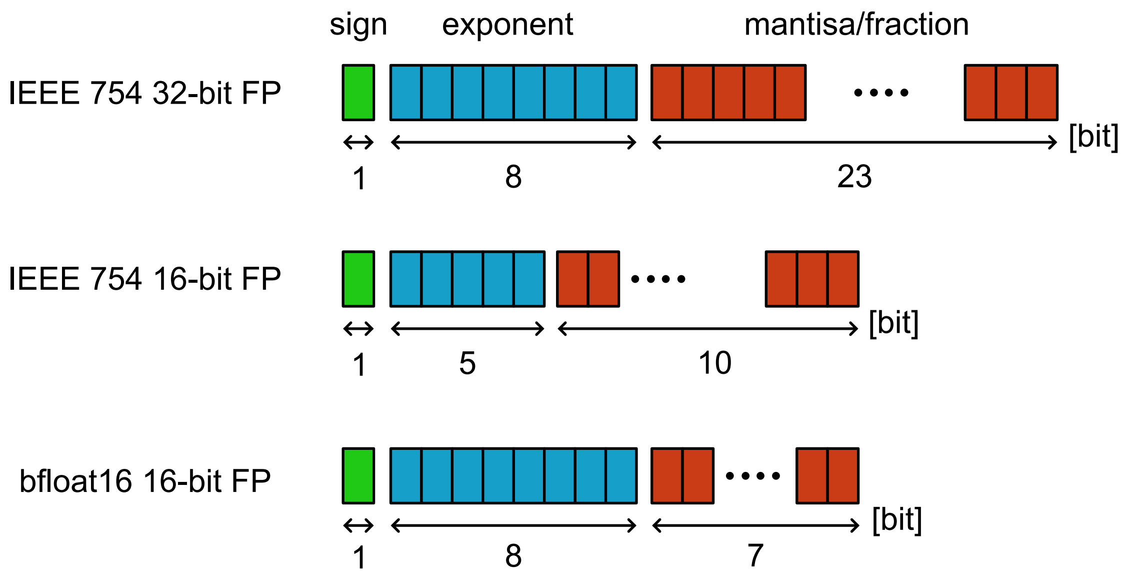


Figure 5.15: Floating-point arithmetic typically uses the IEEE 754 format, a 32-bit format. bfloat-16 has the same dynamic range because its exponential part has the same size as the IEEE 754.

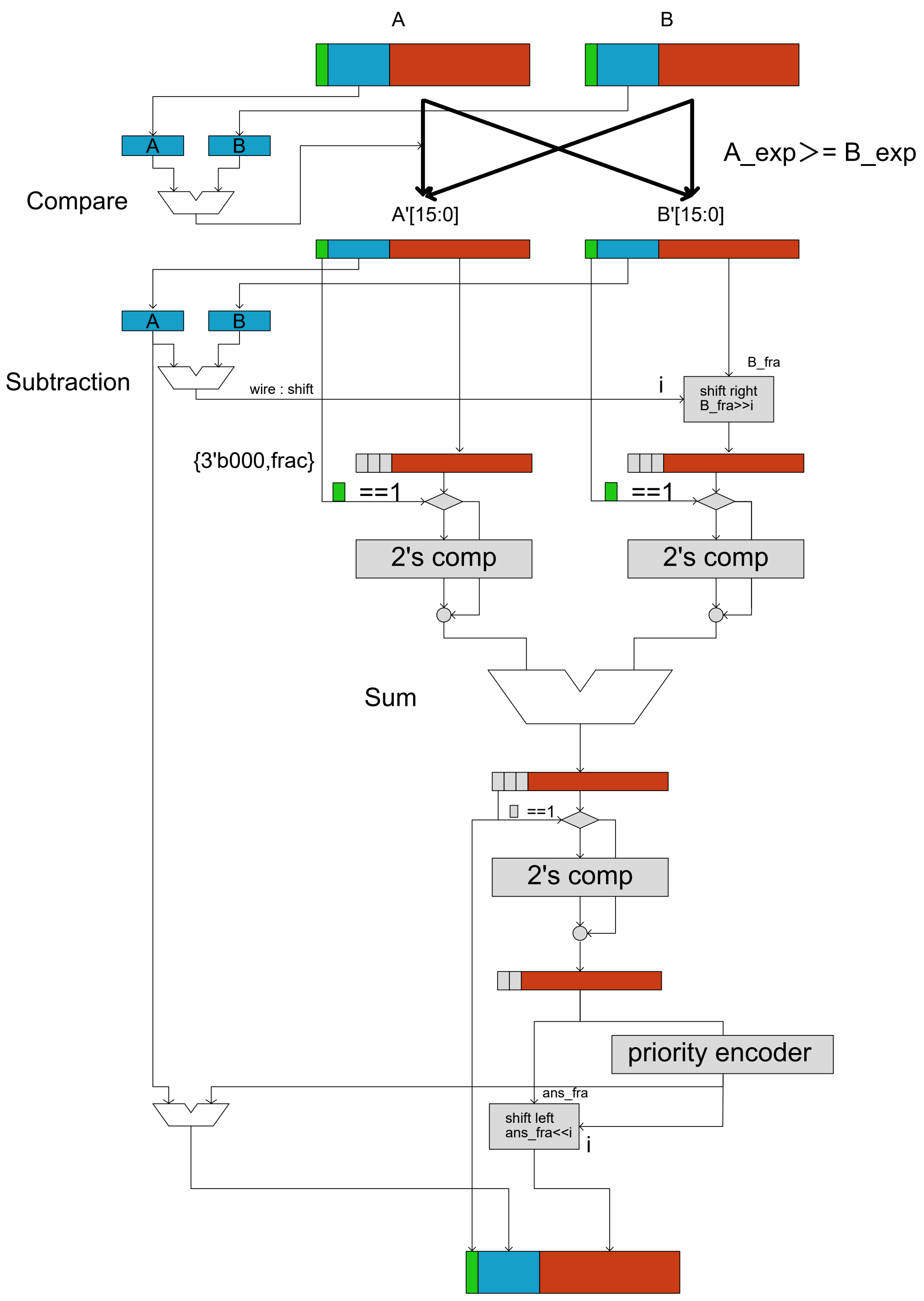


Figure 5.16: Block diagram of bfloat-16 AQFP adder. Green block, blue block, and orange block are denoted sign, exponent, and mantissa part, respectively.

Comparator

The n-bit binary sequence $A = [a_{n-1}, a_{n-2}, \cdots, a_1, a_0], B = [b_{n-1}, b_{n-2}, \cdots, b_1, b_0]$ comparisons are greedy implemented, the computational complexity is $O(n)$, as shown in algorithm 5.1. However, upon careful consideration, this computation can be implemented with a binary tree. In other words, the computational complexity is $O(\log N)$.

algorithm 5.2 shows the implementation with binary trees. Figure 5.17 shows (a) 1-bit and (2) 2-bit comparator. For bfloat-16, 8-bit comparator is required, and it can be realized with 3-stage binary type comparator. The latency of this comparator is approximate half.

Algorithm 5.1 Calculate $A > B$

```
Ensure: $A > B$
1: for i: n-1 to 0 do
2:   if $a_i = b_i$ then
3:     Calculate $[a_{i-1} : a_0] > [b_{i-1} : b_0]$
4:   else if $a_i > b_i$ then
5:     return A is larger than B
6:   else if $a_i < b_i$ then
7:     return B is larger than A
8:   end if
9: end for
```

Algorithm 5.2 Calculate $A > B$ improved version

```
Ensure: $A > B$
1: n ←bit length of A
2: if n = 1 then
3:   return $(eq, large) = (a_0 b_0, a_0 \neg b_0)$
4: else
5:   left ←Calculate $([a_{n-1} : a_{n/2}] > [b_{n-1} : b_{n/2}])$
6:   right ←Calculate $([a_{n/2-1} : a_0] > [b_{n/2-1} : b_0])$
7: end if
8: return $(eq, large) = (left.eq \cdot right.eq, left.eq \cdot right.large)$
```

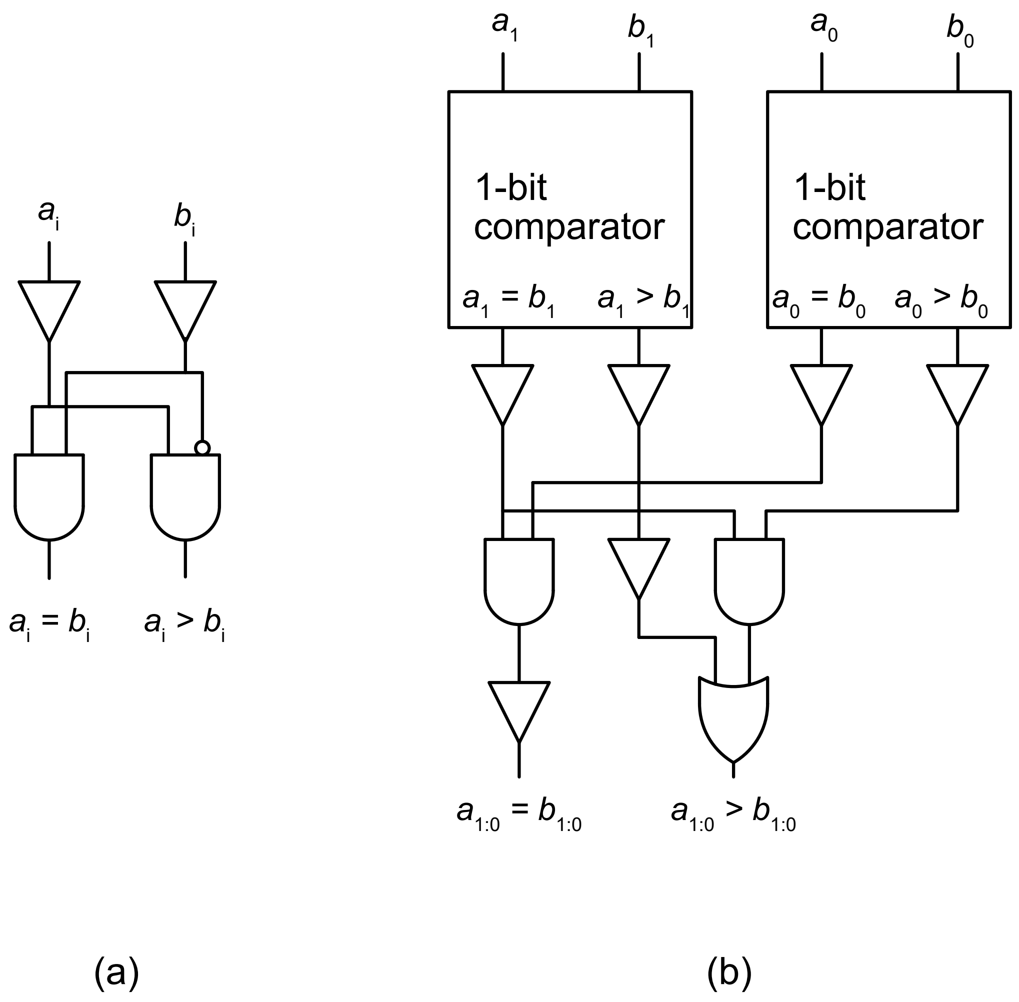


Figure 5.17: Block diagram of bfloat-16 AQFP adder. Green block, blue block, and orange block are denoted sign, exponent, and mantissa part, respectively.

Prioriry encoder

Priority encoder[74] is used to find hidden bits in the result of the addition of mantissa parts. It can look for a “1” at the highest end of a bit sequence table 5.7 shows the truth table of 4-bit priority encoder and figure 5.18 shows schematic of 4-bit priority encoder.

Table 5.7: Truth table of 4-bit priority encoder. V output distinguishes the digit of the priority bit as 0 and no input.

a_3	a_2	a_1	a_0	i_1	i_0	V
0	0	0	0	x	x	0
0	0	0	1	0	0	1
0	0	1	x	0	1	1
0	1	x	x	1	0	1
1	x	x	x	1	1	1

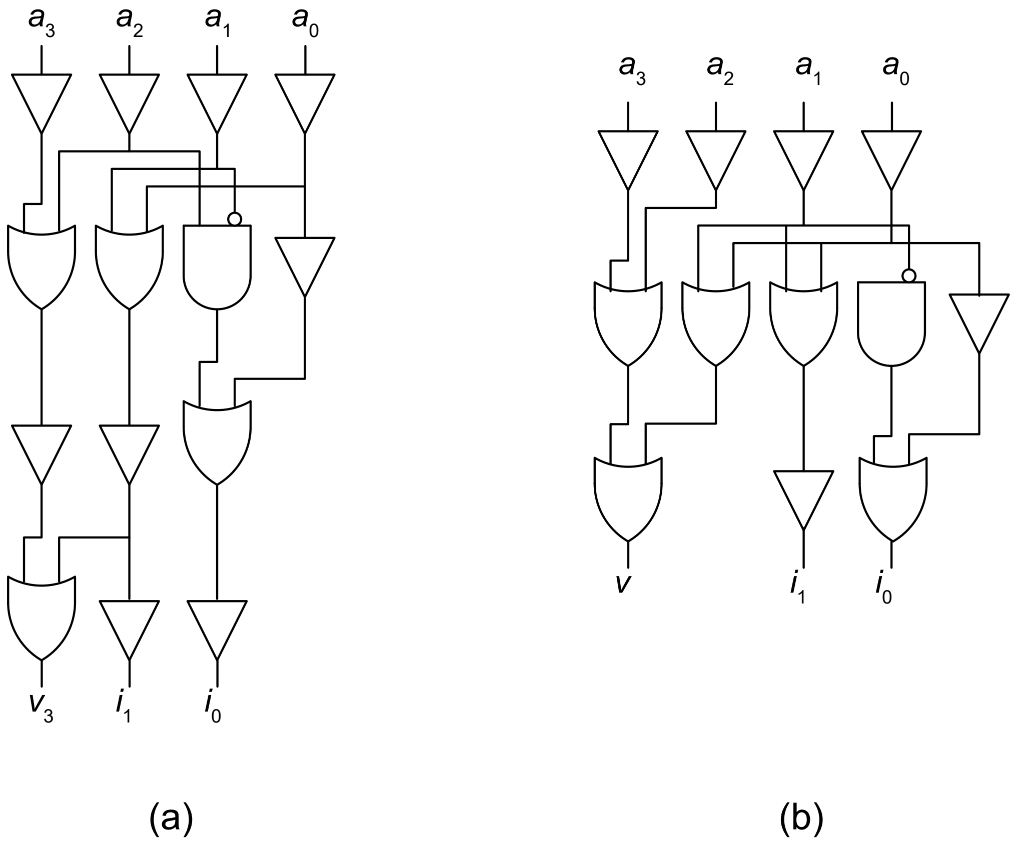


Figure 5.18: Schematic of 4 to 2 priority encoder. (a) Gate minimized version. (b) latency minimized version. If majority-5 gate is available, the latency of I can be reduced two from three.

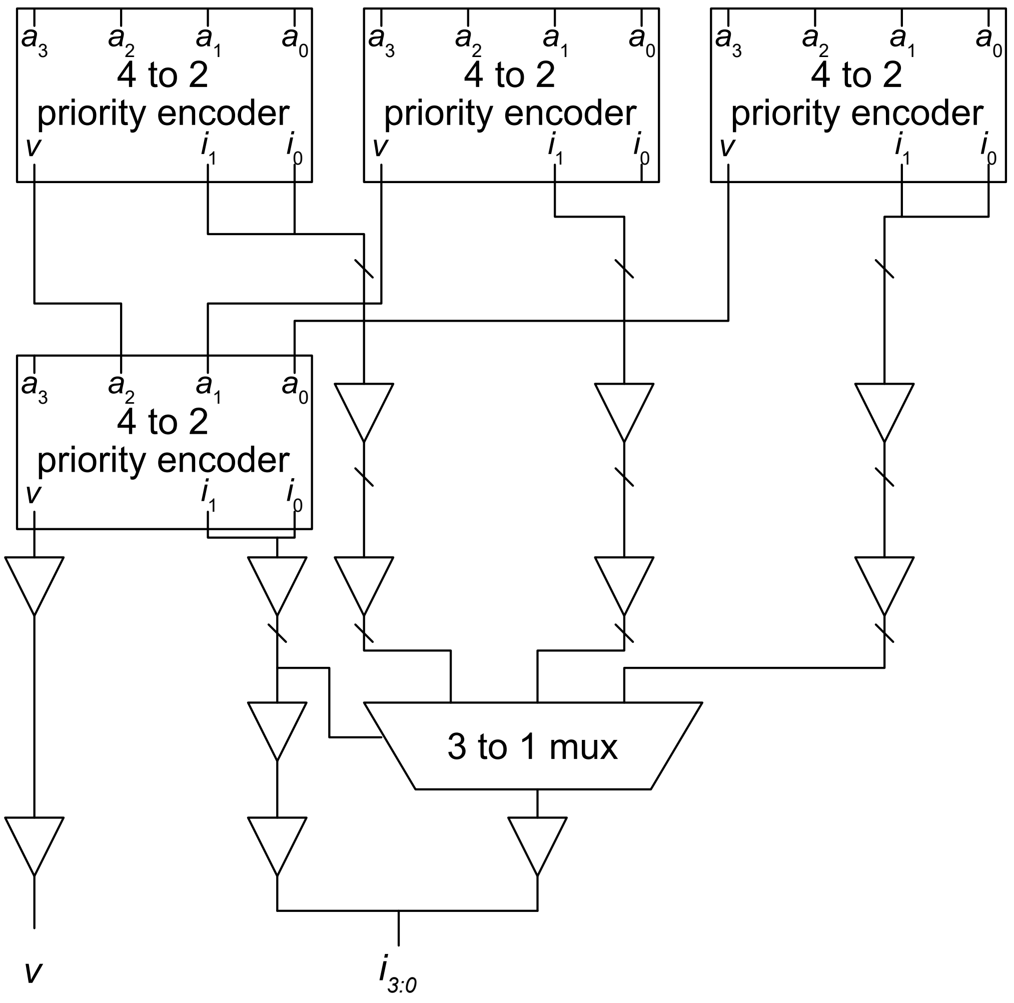


Figure 5.19: Scehmatic of 12 to 4 priority encoder. By combinations of 3 to 1 multiplexer and four priority encoders, the latency and area of this circuit can be reduced.

Shifter

The shifter is used to align the digits in the mantissa. Before addition, it is used to align the digits of operands A and B. After addition, it is used to put the result of the calculation into hidden bit format. For the shifter architecture, a barrel shifter was used. A barrel shifter consists of an array of 2-to-1 MUXes. Table 5.8 and figure 5.20 show the truth table of 2-to-1 multiplexer and the schematic of the 2-to-1 MUX. Figure 2 shows the schematic of the 4-bit shifter.

Table 5.8: The truth table of 2-to-1 multiplexer (MUX)

S	O
0	b
1	a

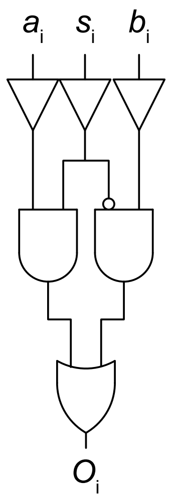


Figure 5.20: Scehmatic of 2-to-1 multiplexer(MUX). When $s_i = 1$, the circuit generates a signal.

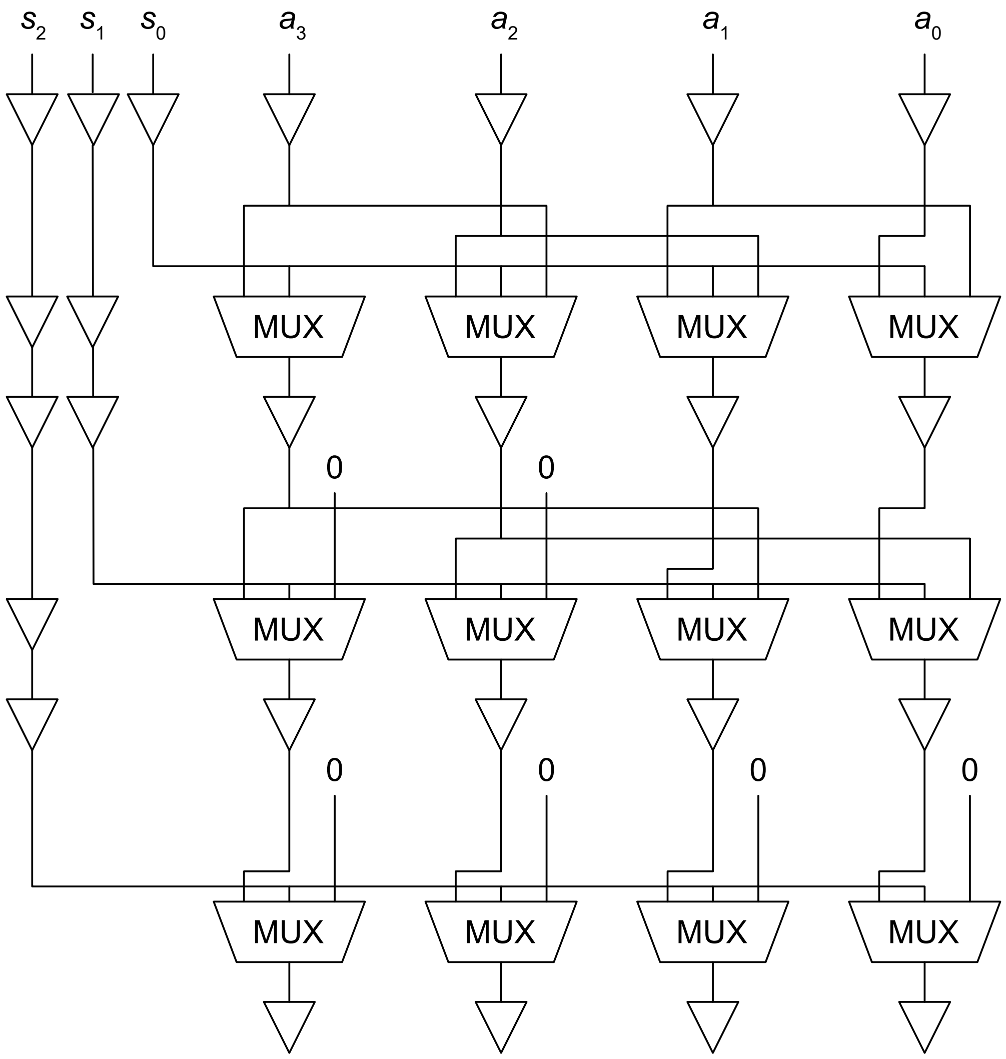


Figure 5.21: Scehmatic of4-bit shifter circuits. It consists of an array of 2-to-1 MUX. The structure of the circuit is simple, but it is required a large foot print.

5.2.1 Discussion

Table 5.9 shows the dimensions of each component of the AQFP floating point adder. Total JJs of the circuit including buffer to consider data timing is 17336. This circuit has 2 shifters, and the shifter has largest footprint.

Table 5.9: AQFP floating point adder Component Dimensions

#	Name	Area mm^2	JJs	Phase
1	Exponent comparator	0.93	450	12
2	Operand swap	1.77	1050	6
3	Expoent subtractor	1.06	540	13
4	Mantissa right shifter	4.14	1724	26
5	Mantissa adder	3.99	2228	20
6	Mantissa 2's compliment	2.48	1142	17
7	Mantissa priority encoder	0.44	420	15
8	Exponent adder	2.56	1556	19
9	Mantissa left shifter	2.91	1372	21

Table 5.10: Comparison with another digital circuit technologies

	AQFP-FP	RSFQ(serial)[75]	CMOS-FP (22nm)[76]
Area	50	–	0.049
JJs	17336	8830	–
Power dissipation	24 fJ/op	16fJ/op	2.18 pJ/op
Operating frequency	5	6	1.85

5.3 Conclusion

In this chapter, integer and floating point adder are designed. Tables 5.5 and 5.10 show standing point of AQFP compared with RSFQ and CMOS circuit.

When compared with RSFQ circits, AQFP has large advantage in energy dissipation but calculation performance is inferior. An application which require time creatical operation, AQFP is not stutable.

When compared with CMOS circuits, AQFP has advantage calculation performance and power dissipation. But, there is extreamly large gap between two circuit technology in footprint area. To compare AQFP to CMOS, efforts to design circuit smaller is required.

Chapter 6

Conclusion

For the design of large AQFP circuits, we constructed an RTL to GDS flow, a technique to generate the physical structure of a circuit from a hardware description language such as Verilog. This development was done through collaborative research, and I was mainly responsible for designing the logic cell library. Using this flow, we have successfully automated the design of circuits with over 1 million Josephson junctions. Since the largest circuits designed to date for AQFP circuits have been around 10,000 junctions, this represents a dramatic improvement in the design capability of AQFP circuits. In both automated and manual design of AQFP circuits, the short wiring length limit is a problem due to the parasitic inductance of the wiring. Therefore, we investigated a method to reduce the inductance per length by making the wiring width variable and assigning wide wiring for long-distance signal propagation. While the conventional wiring length limit was 700um when the wiring width was fixed, the wiring width variable can relax the wiring length limit to 2000um, thereby improving the degree of freedom in circuit design.

The circuit was designed and demonstrated. A 16-bit Kogge-Stone integer adder with 3-input majority logic gates and an 8-bit Kogge-Stone integer adder with 5-input majority logic gates were designed and successfully demonstrated to operate at 100 kHz. These two architectures were compared in terms of latency and energy consumption. It was shown that the maj-5 type adder has better scalability in terms of both energy consumption and latency. The performance of the integer adder in the AQFP circuit was compared to that of RSFQ and CMOS circuits. The RSFQ circuit is superior in terms of operating frequency, and the CMOS circuit is superior in terms of area, but the AQFP circuit is superior in other aspects. A 16-bit floating-point(FP) adder, such as that used in machine learning, was designed. Comparisons were made for the 16-bit FP adder with the CMOS circuit and the RSFQ circuit, and the same trends as for the integer adder were observed.

We have studied the construction of a design method for AQFP circuits. The results of this work have allowed us to design a circuit larger than the chips that can be fabricated with the current fabrication process. The next step will be to improve the circuit design process. Alternate advances in design methods and fabrication methods will be necessary for the future development of AQFP and superconductor circuits.

Acknowledgements

What to write in the chapter called “Acknowledgements” is an open question. Although my doctoral dissertation is to be presented as my sole work, there is no doubt that I could not have submitted this dissertation without the important advisors and colleagues who will be writing it. I would like to write about their contributions and my appreciation for them.

First, I would like to express the utmost gratitude that I can express to my advisor, professor Nobuyuki Yoshikawa. The many research funds you have collected for us have provided me with excellent facilities in our laboratory, and I have been able to conduct the research I wanted to do without any inconvenience. I am also very grateful to you for employing me as a research assistant, which enabled me to spend three years of my doctoral program without any financial worries. In addition, you are a person of great character, very generous, and very poised. There were many challenges that I was able to overcome because you told me it would be okay.

Secondly, I would like to extend the same amount of gratitude to my substantive research advisor, associate professor Christopher Lawrence Ayala, who has been a great help to me in my research. I was not a very good English speaker, but you never gave up and communicated with me. You were extremely good as an advisor and there were many technical issues that I could not have solved without you.

Without professor Nobuyuki Yoshikawa and associate professor Christopher Lawrence Ayala, I would not have enrolled in the Ph.D. course. Meeting both of you is one of the greatest blessings in my life. I am grateful for the opportunity to do research for three years and for the fact that they inspired me to pursue a Ph.D.

I would also like to thank associate professor Yuki Yamanashi, associate professor Naoki Takeuchi, visiting professor Hideo Suzuki, assistant professor Lieze Schindler, Michael Johnston, and associate professor Olivia Chen, department of information science, Tokyo City University. Especially with visiting professor Hideo Suzuki, we worked together to improve the experimental environment in the laboratory, and although nothing remains as a result of our research, we spent a lot of meaningful time together.

I also express my gratitude to the members of Synopsys, who were my partners in the research. It was an exciting experience to learn a lot about the way American companies work and to collaborate with a leading EDA tool company.

As a senior member of the AQFP Microarchitecture group, I would like to express my deepest gratitude to Dr. Ro Saito for his advice on the content of our research. I am also grateful to Takehisa Yamada, Risako Saito, Shohei Takagi, Yu Hoshika, and Yuto Omori, with whom I had a good time as a member of the group.

I am also very grateful to Yuki Hironaka and Taiki Yamae, with whom I have spent

6 very long years. D. program is a lonely existence by nature, but thanks to you guys, I was able to spend my days without loneliness. I am equally grateful to Hongxiang Shen.

It is not possible to write all the names, but I would like to give equal thanks to all the members of Yoshikawa Lab, which has more than 30 members. I know I was a depressing senior, but I am delighted to have spent time with you guys.

In addition, I was hired as a Japan Society for the promotion of science (JSPS) research fellow (DC2) in the third year of my doctoral program. It pains me to have to decline during the period of employment, but I am grateful for the financial support.

For the advancement of science, also I need to say thank you. I needed an excellent translator, spell checker and grammar checker, especially since my English is not good, thanks to DeepL and Grammarly.

And finally, although not a contribution related to science, I would like to thank my father, Katsuyoshi Tanaka, and my mother, Sayuri Tanaka. They were the ones who pushed me to enter the doctoral program, and I am greatly indebted to them for listening to me when I was mentally hard, and for their warm welcome whenever I returned home.

March 2023
Graduate School of Engineering, Yokohama National University
Yoshikawa laboratory
Tomoyuki Tanaka

Appendix A

Synthesis results

List of raw data to create the table, cell-by-cell breakdonw of circuits generated from RTL by Synopsys logic synthesis tools. In this cell library, we are using buffer, inverter, 3-input majority gate, 2-input and gate, 2-input or gate, 2-output splitter gate, 3-output splitter gate, and filler cell are used.

Table A.1: Cell-by-cell breakdown for the C17 benchmark[46]

Cell name	Cell Count	JJs/cell	total JJs
filler	109	0	0
and2_pp_even	3	2	6
bfr_100	13	2	26
bfr_100_even	8	2	16
or2_nn_even	1	6	6
or2_pp	1	6	6
or2_pp_even	1	6	6
spl2_100	3	2	6
Total			72

Table A.2: Cell-by-cell breakdown for the C432 benchmark[46]

Cell name	Cell Count	JJs/cell	total JJs
filler	4953	0	0
and2_nn	1	6	6
and2_nn_even	7	6	42
and2_pn	17	6	102
and2_pn_even	32	6	192
and2_pp	10	6	60
and2_pp_even	8	2	16
bfr_100	674	2	1348
bfr_100_even	704	2	1408
maj3_npp	52	6	312
maj3_ppp_even	3	6	18
or2_pn_even	1	6	6
or2_pp	20	6	120
or2_pp_even	13	6	78
spl2_100	15	2	30
spl2_100_even	45	2	90
spl3_100	37	2	74
spl3_100_even	28	2	56
Total			3952

Table A.3: Cell-by-cell breakdown for the C499 benchmark[46]

Cell name	Cell Count	JJs/cell	total JJs
filler	9136	0	0
and2_nn_even	8	6	48
and2_pn	128	6	768
and2_pn_even	62	6	372
and2_pp	48	6	288
and2_pp_even	25	2	50
bfr_100	992	2	1984
bfr_100_even	936	2	1872
or2_nn	8	6	48
or2_nn_even	8	6	48
or2_pp	32	6	192
or2_pp_even	72	6	432
spl2_100	101	2	202
spl2_100_even	71	2	142
spl3_100	21	2	42
spl3_100_even	84	2	168
Total			6656

Table A.4: Cell-by-cell breakdown for the C880 benchmark[46]

Cell name	Cell Count	JJs/cell	total JJs
filler	11375	0	0
and2_pn	37	6	222
and2_pn_even	30	6	180
and2_pp	77	6	462
and2_pp_even	38	2	76
bfr_100	1254	2	2508
bfr_100_even	1255	2	2510
maj3_npp	6	6	36
maj3_npp_even	5	6	30
maj3_ppp	23	6	138
maj3_ppp_even	24	6	144
or2_nn	9	6	54
or2_nn_even	3	6	18
or2_pn_even	1	6	6
or2_pp	50	6	300
or2_pp_even	59	6	354
spl2_100	72	2	144
spl2_100_even	50	2	100
spl3_100	53	2	106
spl3_100_even	79	2	158
Total			7546

Table A.5: Cell-by-cell breakdown for the C1355 benchmark[46]

Cell name	Cell Count	JJs/cell	total JJs
filler	9173	0	0
and2_nn_even	36	6	216
and2_pn	128	6	768
and2_pn_even	56	6	336
and2_pp	48	6	288
and2_pp_even	28	2	56
bfr_100	988	2	1976
bfr_100_even	936	2	1872
or2_nn	8	6	48
or2_nn_even	8	6	48
or2_pp	32	6	192
or2_pp_even	40	6	240
spl2_100	96	2	192
spl2_100_even	73	2	146
spl3_100	21	2	42
spl3_100_even	82	2	164
Total			6584

Table A.6: Cell-by-cell breakdown for the C1908 benchmark[46]

Cell name	Cell Count	JJs/cell	total JJs
filler	11290	0	0
and2__nn	5	6	30
and2__nn__even	6	6	36
and2__pn	81	6	486
and2__pn__even	53	6	318
and2__pp	37	6	222
and2__pp__even	64	2	128
bfr__100	1903	2	3806
bfr__100__even	1836	2	3672
maj3__nnn__even	1	6	6
maj3__npp	3	6	18
maj3__npp__even	9	6	54
maj3__ppp	5	6	30
maj3__ppp__even	4	6	24
or2__nn	10	6	60
or2__nn__even	10	6	60
or2__pn	1	6	6
or2__pn__even	3	6	18
or2__pp	37	6	222
or2__pp__even	64	6	384
spl2__100	89	2	178
spl2__100__even	36	2	72
spl3__100	55	2	110
spl3__100__even	86	2	172
Total			10112

Table A.7: Cell-by-cell breakdown for the C2670 benchmark[46]

Cell name	Cell Count	JJs/cell	total JJs
filler	16841	0	0
and2_nn	6	6	36
and2_nn_even	4	6	24
and2_pn	79	6	474
and2_pn_even	92	6	552
and2_pp	70	6	420
and2_pp_even	74	2	148
bfr_100	3207	2	6414
bfr_100_even	3029	2	6058
inv	5	2	10
inv_even	13	2	26
maj3_nnn	1	6	6
maj3_nnn_even	2	6	12
maj3_npn	2	6	12
maj3_npp	1	6	6
maj3_npp_even	2	6	12
maj3_ppp	14	6	84
maj3_ppp_even	18	6	108
or2_nn	8	6	48
or2_nn_even	9	6	54
or2_pn	1	6	6
or2_pn_even	3	6	18
or2_pp	83	6	498
or2_pp_even	87	6	522
spl2_100	125	2	250
spl2_100_even	83	2	166
spl3_100	72	2	144
spl3_100_even	75	2	150
Total			16258

Table A.8: Cell-by-cell breakdown for the C3540 benchmark[46]

Cell name	Cell Count	JJs/cell	total JJs
filler	47187	0	0
and2_nn	20	6	120
and2_nn_even	17	6	102
and2_pn	102	6	612
and2_pn_even	81	6	486
and2_pp	178	6	1068
and2_pp_even	183	2	366
bfr_100	3209	2	6418
bfr_100_even	3232	2	6464
maj3_nnn	5	6	30
maj3_nnn_even	14	6	84
maj3_npn	7	6	42
maj3_npn_even	9	6	54
maj3_npp	19	6	114
maj3_npp_even	21	6	126
maj3_ppp	56	6	336
maj3_ppp_even	79	6	474
or2_nn	19	6	114
or2_nn_even	21	6	126
or2_pn	13	6	78
or2_pn_even	9	6	54
or2_pp	129	6	774
or2_pp_even	141	6	846
spl2_100	182	2	364
spl2_100_even	145	2	290
spl3_100	260	2	520
spl3_100_even	229	2	458
Total			20520

Table A.9: Cell-by-cell breakdown for the C6288 benchmark[46]

Cell name	Cell Count	JJs/cell	total JJs
filler	271901	0	0
and2_nn	2	6	12
and2_nn_even	1	6	6
and2_pn	300	6	1800
and2_pn_even	341	6	2046
and2_pp	340	6	2040
and2_pp_even	674	2	1348
bfr_100	48855	2	97710
bfr_100_even	49048	2	98096
inv	10	2	20
inv_even	18	2	36
maj3_nnn	2	6	12
maj3_npn	1	6	6
maj3_npn_even	1	6	6
maj3_npp	315	6	1890
maj3_npp_even	293	6	1758
maj3_ppp	250	6	1500
maj3_ppp_even	343	6	2058
or2_nn	96	6	576
or2_nn_even	125	6	750
or2_pn	22	6	132
or2_pn_even	8	6	48
or2_pp	224	6	1344
or2_pp_even	215	6	1290
spl2_100	708	2	1416
spl2_100_even	478	2	956
spl3_100	1042	2	2084
spl3_100_even	744	2	1488
Total			220428

Table A.10: Cell-by-cell breakdown for the C7552 benchmark[46]

Cell name	Cell Count	JJs/cell	total JJs
filler	55834	0	0
and2_nn	13	6	78
and2_nn_even	16	6	96
and2_pn	181	6	1086
and2_pn_even	292	6	1752
and2_pp	107	6	642
and2_pp_even	134	2	268
bfr_100	6504	2	13008
bfr_100_even	6292	2	12584
inv	1	2	2
inv_even	5	2	10
maj3_nnn	1	6	6
maj3_nnn_even	3	6	18
maj3_npp	23	6	138
maj3_npp_even	12	6	72
maj3_ppp	24	6	144
maj3_ppp_even	19	6	114
or2_nn	46	6	276
or2_nn_even	51	6	306
or2_pn	8	6	48
or2_pn_even	15	6	90
or2_pp	230	6	1380
or2_pp_even	183	6	1098
spl2_100	252	2	504
spl2_100_even	257	2	514
spl3_100	230	2	460
spl3_100_even	186	2	372
Total			35066

Table A.11: Cell-by-cell breakdown for the C5315 benchmark[46]

Cell name	Cell Count	JJs/cell	total JJs
filler	46568	0	0
and2_nn	10	6	60
and2_nn_even	12	6	72
and2_pn	105	6	630
and2_pn_even	215	6	1290
and2_pp	173	6	1038
and2_pp_even	191	2	382
bfr_100	5660	2	11320
bfr_100_even	5514	2	11028
inv	21	2	42
inv_even	25	2	50
maj3_nnn	2	6	12
maj3_nnn_even	3	6	18
maj3_npn	1	6	6
maj3_npp	8	6	48
maj3_npp_even	5	6	30
maj3_ppp	28	6	168
maj3_ppp_even	16	6	96
or2_nn	35	6	210
or2_nn_even	82	6	492
or2_pn	14	6	84
or2_pn_even	17	6	102
or2_pp	198	6	1188
or2_pp_even	205	6	1230
spl2_100	259	2	518
spl2_100_even	169	2	338
spl3_100	256	2	512
spl3_100_even	194	2	388
Total			31352

Table A.12: Cell-by-cell breakdown for the Divisor benchmark

Cell name	Cell Count	JJs/cell	total JJs
bfr_100	76031230	2	152062460
spl3_100	22096	2	44192
spl2_100	17970	2	35940
or2_pp	12585	6	75510
and2_pp	11877	6	71262
and2_pn	10154	6	60924
maj3_ppp	7413	6	44478
maj3_npp	2987	6	17922
or2_nn	2405	6	14430
and2_nn	1277	6	7662
or2_pn	1274	6	7644
maj3_nnn	676	6	4056
inv	316	2	632
maj3_npn	219	6	1314
Total			152448426

Table A.13: Cell-by-cell breakdown for the 8-bit adder

Cell name	Cell Count	JJs/cell	total JJs
filler	1099	0	0
bfr_100	160	2	320
bfr_100_even	143	2	286
spl2_100	25	2	50
or2_pp_even	15	6	90
spl3_100	10	2	20
maj3_ppp_even	10	6	60
spl2_100_even	9	2	18
and2_pp_even	9	2	18
spl3_100_even	9	2	18
or2_pp	8	6	48
or2_nn_even	6	6	36
and2_pp	6	6	36
and2_pn	5	6	30
maj3_ppp	3	6	18
or2_nn	2	6	12
and2_pn_even	2	6	12
Total			1072

Table A.14: Cell-by-cell breakdown for the 16-bit adder

Cell name	Cell Count	JJs/cell	total JJs
filler	3319	0	0
bfr_100	492	2	984
bfr_100_even	461	2	922
spl2_100	47	2	94
or2_pp_even	31	6	186
spl3_100	26	2	52
spl2_100_even	25	2	50
spl3_100_even	25	2	50
and2_pp_even	23	2	46
maj3_ppp_even	23	6	138
or2_pp	20	6	120
maj3_ppp	15	6	90
and2_pn	12	6	72
or2_nn	8	6	48
or2_nn_even	8	6	48
and2_pp	8	6	48
and2_pn_even	3	6	18
Total			2966

Table A.15: Cell-by-cell breakdown for the 32-bit adder

Cell name	Cell Count	JJs/cell	total JJs
filler	9750	0	0
bfr_100	1457	2	2914
bfr_100_even	1360	2	2720
spl2_100	104	2	208
spl3_100_even	68	2	136
or2_pp_even	64	6	384
maj3_ppp_even	63	6	378
spl2_100_even	58	2	116
spl3_100	55	2	110
and2_pp_even	51	2	102
or2_pp	46	6	276
maj3_ppp	37	6	222
and2_pn	28	6	168
or2_nn	18	6	108
and2_pp	14	6	84
or2_nn_even	13	6	78
and2_pn_even	3	6	18
maj3_nnn_even	1	6	6
Total			8028

Table A.16: Cell-by-cell breakdown for the 64-bit adder

Cell name	Cell Count	JJs/cell	total JJs
filler	25092	0	0
bfr_100	3936	2	7872
bfr_100_even	3784	2	7568
spl2_100	212	2	424
spl3_100_even	165	2	330
maj3_ppp_even	140	6	840
spl2_100_even	138	2	276
spl3_100	120	2	240
and2_pp_even	118	2	236
or2_pp_even	115	6	690
or2_pp	113	6	678
maj3_ppp	109	6	654
and2_pn	60	6	360
or2_nn	52	6	312
or2_nn_even	11	6	66
and2_pp	11	6	66
and2_pn_even	3	6	18
maj3_nnn	1	6	6
Total			20636

Table A.17: Cell-by-cell breakdown for the 128-bit adder

Cell name	Cell Count	JJs/cell	total JJs
filler	63315	0	0
bfr_100	10504	2	21008
bfr_100_even	10194	2	20388
spl2_100	508	2	1016
spl3_100_even	379	2	758
maj3_ppp_even	347	6	2082
spl3_100	290	2	580
or2_pp_even	288	6	1728
maj3_ppp	285	6	1710
spl2_100_even	281	2	562
and2_pp_even	184	2	368
or2_pp	183	6	1098
and2_pn	124	6	744
and2_pp	77	6	462
or2_nn_even	71	6	426
or2_nn	54	6	324
maj3_nnn	3	6	18
and2_pn_even	3	6	18
Total			53290

Table A.18: Cell-by-cell breakdown for the 512-bit adder

Cell name	Cell Count	JJs/cell	total JJs
filler	1011857	0	0
bfr_100	130607	2	261214
bfr_100_even	129291	2	258582
spl2_100	2011	2	4022
spl3_100_even	1512	2	3024
maj3_ppp_even	1416	6	8496
spl3_100	1200	2	2400
spl2_100_even	1155	2	2310
or2_pp_even	1147	6	6882
maj3_ppp	1112	6	6672
and2_pp_even	806	2	1612
or2_pp	770	6	4620
and2_pn	496	6	2976
and2_pp	274	6	1644
or2_nn	245	6	1470
or2_nn_even	240	6	1440
maj3_nnn	22	6	132
and2_pn_even	17	6	102
maj3_nnn_even	3	6	18
Total			567616

Table A.19: Cell-by-cell breakdown for the 1024-bit adder

Cell name	Cell Count	JJs/cell	total JJs
filler	4218351	0	0
bfr__100	546227	2	1092454
bfr__100__even	543303	2	1086606
spl2__100	4041	2	8082
spl3__100__even	3019	2	6038
maj3__ppp__even	2921	6	17526
spl2__100__even	2421	2	4842
spl3__100	2412	2	4824
or2__pp__even	2267	6	13602
maj3__ppp	2195	6	13170
and2__pp__even	1615	2	3230
or2__pp	1569	6	9414
and2__pn	999	6	5994
and2__pp	543	6	3258
or2__nn	516	6	3096
or2__nn__even	451	6	2706
and2__pn__even	43	6	258
maj3__nnn__even	32	6	192
maj3__nnn	18	6	108
and2__nn	3	6	18
or2__pn	3	6	18
or2__pn__even	3	6	18
and2__nn__even	3	6	18
Total			2275472

Table A.20: Cell-by-cell breakdown for the 8-bit multiplier

Cell name	Cell Count	JJs/cell	total JJs
filler	9569	0	0
bfr_100	718	2	1436
bfr_100_even	691	2	1382
spl2_100_even	119	2	238
and2_pp	117	6	702
spl3_100_even	112	2	224
or2_pp	87	6	522
spl3_100	82	2	164
or2_pp_even	61	6	366
spl2_100	56	2	112
and2_pp_even	53	2	106
and2_pn_even	39	6	234
maj3_ppp	35	6	210
and2_pn	31	6	186
maj3_ppp_even	31	6	186
or2_nn	22	6	132
or2_nn_even	12	6	72
or2_pn	3	6	18
or2_pn_even	2	6	12
maj3_nnn_even	2	6	12
Total			6314

Table A.21: Cell-by-cell breakdown for the 16-bit multiplier

Cell name	Cell Count	JJs/cell	total JJs
filler	80352	0	0
bfr_100	4028	2	8056
bfr_100_even	3940	2	7880
spl3_100	584	2	1168
and2_pp_even	497	2	994
spl2_100	488	2	976
or2_pp_even	418	6	2508
spl3_100_even	356	2	712
or2_pp	247	6	1482
spl2_100_even	230	2	460
and2_pp	210	6	1260
and2_pn_even	196	6	1176
maj3_ppp_even	165	6	990
and2_pn	153	6	918
maj3_ppp	141	6	846
or2_nn_even	108	6	648
or2_nn	59	6	354
maj3_npp	15	6	90
or2_pn_even	11	6	66
maj3_npp_even	10	6	60
or2_pn	8	6	48
inv_even	7	2	14
maj3_nnn_even	7	6	42
and2_nn_even	6	6	36
and2_nn	3	6	18
maj3_npn_even	3	6	18
Total			30820

Table A.22: Cell-by-cell breakdown for the 32-bit multiplier

Cell name	Cell Count	JJs/cell	total JJs
filler	701827	0	0
bfr_100_even	18575	2	37150
bfr_100	17789	2	35578
spl3_100_even	2327	2	4654
spl2_100_even	2213	2	4426
and2_pp	1949	6	11694
or2_pp	1801	6	10806
and2_pn	1189	6	7134
spl3_100	1088	2	2176
spl2_100	939	2	1878
or2_pp_even	763	6	4578
and2_pn_even	714	6	4284
and2_pp_even	668	2	1336
or2_nn	477	6	2862
maj3_ppp	467	6	2802
maj3_ppp_even	247	6	1482
or2_nn_even	177	6	1062
maj3_npp	150	6	900
maj3_npp_even	135	6	810
and2_nn	60	6	360
or2_pn	59	6	354
or2_pn_even	45	6	270
inv	32	2	64
and2_nn_even	22	6	132
maj3_npn	17	6	102
inv_even	10	2	20
maj3_nnn	5	6	30
maj3_nnn_even	5	6	30
maj3_npn_even	3	6	18
Total			136992

本研究に関する発表

発表論文

採択済み

1. T. Tanaka, C. L. Ayala, and N. Yoshikawa, “A 16-Bit Parallel Prefix Carry Look-Ahead Kogge-Stone Adder Implemented in Adiabatic Quantum-Flux-Parametron Logic,” IEICE Transactions on Electronics, vol. E105.C, no. 6, pp. 270 – 276, Jun. 2022.
2. C. L. Ayala, T. Tanaka, R. Saito, M. Nozoe, N. Takeuchi, and N. Yoshikawa, “MANA: A Monolithic Adiabatic iNtegration Architecture Microprocessor Using 1.4-zJ/op Unshunted Superconductor Josephson Junction Devices,” IEEE J. Solid-State Circuits, vol. 56, no. 4, pp. 1152 – 1165, Apr. 2021.
3. C. L. Ayala, R. Saito, T. Tanaka, C. Olivia, Y. He., N. Takeuchi, and N. Yoshikawa“A semi-custom design methodology and environment for implementing superconductor adiabatic quantum-flux-parametron microprocessors,” Supercond. Sci. Technol., vol. 33, no. 5, p. 054006, Mar. 2020.
4. C. J. Fourie, C. L. Ayala, L. Schindler, T. Tanaka, and N. Yoshikawa, “Design and Characterization of Track Routing Architecture for RSFQ and AQFP Circuits in a Multilayer Process,” IEEE Trans. Appl. Supercond., vol. 30, no. 6, pp. 1 – 9, Sep. 2020.
5. R. Saito, C. L. Ayala, O. Chen, T. Tanaka, T. Tamura, and N. Yoshikawa, “Logic Synthesis of Sequential Logic Circuits for Adiabatic Quantum-Flux-Parametron Logic,” IEEE Trans. Appl. Supercond., vol. 31, no. 5, pp. 1 – 5, Aug. 2021.
6. T. Tanaka, C. L. Ayala, S. Whiteley, E. Milnar, A. Barker, S. Chen, A. Beleov, T. Barbee III, J. Kawa, N. Yoshikawa, “A full-custom design flow and a top-down RTL-to-GDS flow for adiabatic quantum-flux-parametron logic using a commercial EDA design suite,” IEEE Trans. Appl. Supercond.
7. Y. Hironaka, S. Meher, C. L. Ayala, Y. He, T. Tanaka, M. Habib, A. Sahu, A. Inamdar, D. Gupta, and N. Yoshikawa, “Demonstration of interface circuits for adiabatic quantum-flux-parametron cell library using an eight-metal layer superconductor process,” IEEE Trans. Appl. Supercond.
8. M. Johnston, C. L. Ayala, T. Tanaka, N. Yoshikawa, “Analysis and Stabilization of Signal Reflections in Gate-to-Gate Connections for AQFP Circuits,” IEEE Trans. Appl. Supercond.

口頭・ポスター発表

1. 田中 智之, Christopher L. Ayala, 田村 智大 , 齋藤 蕗生 , 伊東 大樹 , 浅井 和人 , 竹内 尚輝 , 吉川 信行「超伝導断熱量子磁束パラメトロン回路によるコンピュータの実現とその現状」、ナノ学会　横浜 2020, ナノ学会 2020 年 5 月
2. 田中 智之, AYALA Christopher, 吉川 信行「断熱量子磁束パラメトロン回路のグリッドレスチャネル配線による長距離配線」、99 回低温工学会, 低温工学・超電導学会　横浜 2020 7 月
3. 山田 剛久, C. L. Ayala, 齋藤 蕗生, 田中 智之, 吉川 信行,「機械学習による断熱量子磁束パラメトロン集積回路の配置順序最適化の改良検討」, 低温工学・超電導学会研究発表会, 低温工学・超電導学会, 2020 年 7 月
4. 齋藤 蕗生, AYALA Christopher L, CHEN Olivia, 田中 智之, 田村 智大, 吉川 信行,「Adiabatic Quantum-Flux-Parametron Logic のためのフィードバック回路の自動合成」, 低温工学・超電導学会研究発表会会, 低温工学・超電導学会 2020 年 7 月
5. 田中 智之, Christopher L. Ayala, 吉川信行,「チャネル配線の寄生インダクタンス最適化による断熱量子磁束パラメトロン回路の小型化」　通信学会ソサイエティ大会　通信学会　2020 9 月
6. 山田 剛久, C. L. Ayala, 齋藤 蕗生, 田中 智之, 吉川信行,「特殊ケースにおける 2 層チャネル配線アルゴリズムの開発」, 電子情報通信学会ソサイエティ大会 通信学会 2020 年 9 月.
7. 齋藤 蕗生, Christopher L. Ayala, Olivia Chen, 田中 智之, 吉川 信行,「AQFP 回路のための N-Phase Clocking に関する研究」, 通信学会ソサイエティ大会, 通信学会, 2020 年 9 月.
8. Tomoyuki Tanaka, Christopher L. Ayala, Ro Saito , Daiki Ito, Kazuhito Asai, Naoki Takeuchi, Nobuyuki Yoshikawa, “Development of an ultra-low power microprocessor using adiabatic quantum-flux-parametron circuits,” 電気学会、新潟 2020 年 9 月
9. Ro Saito, Christopher L. Ayala, Olivia Chen, Tomoyuki Tanaka, Tomohiro Tamura, Nobuyuki Yoshikawa, “Study of clock synchronization for high-level synthesis of adiabatic quantum-flux-parametron sequential logic circuits,” ASC 2020, online, Oct. 2020.
10. T. Tanaka, C. L. Ayala, and N. Yoshikawa, “A 16-bit parallel prefix carry look-ahead Kogge-Stone adder implemented in adiabatic quantum-flux-parametron logic,” ASC 2020, online Oct. 2020
11. Christopher L. Ayala, Ro Saito, Tomoyuki Tanaka, Tomohiro Tamura, Naoki Takeuchi, and Nobuyuki Yoshikawa, “A 4-bit RISC-Dataflow AQFP MANA Microprocessor: Architecture, Design Challenges, and Demonstration,” ASC2020, Online Oct. 2020
12. T. Yamada, C. L. Ayala, R. Saito, T. Tanaka, N. Yoshikawa, “Development of a Generic Two-Layer Channel Routing Algorithm for Adiabatic Quantum-Flux-Parametron Logic Using Advanced Fabrication Process,” The 33rd International Symposium on Superconductivity (ISS), Tsukuba, Japan, Dec. 2020.
13. 田中 智之, AYALA Christopher, 吉川 信行「断熱量子磁束パラメトロン回路の配

置最適化と配線インダクタンス最適化による集積性の改善」、100 回低温工学会 京都 2020 年 12 月

14. 田中 智之, AYALA Christopher, 齋藤 蕗生, 吉川 信行「断熱量子磁束パラメトロン回路用自動設計ツールにおける配線幅・配置最適化」、通信学会総合大会　オンライン 2021 年 3 月
15. 田中 智之, Christopher Ayala, 吉川信行,「断熱量子磁束パラメトロン回路を用いた浮動小数点加算器の設計」, 通信学会ソサイエティ大会 2021 年 9 月
16. Tomoyuki Tanaka, Christopher L. Ayala and Nobuyuki Yoshikawa, "Demonstration of 8-bit Kogge-Stone adder using adiabatic quantum-flux-parametron logic with 5-input majority gate," SSV2021, SSV committee, Nagoya Dec. 2021
17. Tomoyuki Tanaka, Christopher L. Ayala and Nobuyuki Yoshikawa, "Design of adiabatic quantum-flux-parametron bfloat16 floating point arithmetic unit," ISS2021, AIST, Dec. 2021
18. 齋藤 理彩子, 田中 智之, Christopher Ayala, 吉川 信行「断熱量子磁束パラメトロン回路における 5 入力多数決論理ゲートを用いた 8-bit Kogge-Stone 加算器の設計と評価」, 通信学会ソサイエティ大会 2022 年 3 月
19. Tomoyuki Tanaka, Christopher Ayala, and Nobuyuki Yoshikawa, "Investigation of power consumption of adiabatic quantum-flux-parametron Kogge-Stone adder circuits by using 5-input majority gate," 電気学会　種子島 2022 年 9 月
20. Shohei Takagi, Tomoyuki Tanaka, Christopher L. Ayala, and Nobuyuki Yoshikawa, "Design and Analysis of Energy-Efficient Adiabatic Quantum-Flux-Parametron Multiplier Families," SSV2022, Kyoto, Sep. 2022
21. Tomoyuki Tanaka, Sukanya S. Meher, Christopher L. Ayala, Yuki Hironaka, AnubhavSahu, Amol Inamdar, Deepnarayan Gupta, and Nobuyuki Yoshikawa, "Design of a hybrid superconductor logic computation system using single flux quantum and adiabatic quantum-flux-parametron logic families," SSV2022, Kyoto, Sep. 2022
22. Tomoyuki Tanaka, Christopher Ayala, Stephen Whiteley, Eric Mlinar, Aaron Barker, Sam Lo, Jamil Kawa, and Nobuyuki Yoshikawa, "A full-custom design flow and a top-down RTL-to-GDS flow for adiabatic quantum-flux-parametron logic using a commercial EDA design suite," ASC2022, Hawaii, Oct. 2022.
23. Tomoyuki Tanaka, Sukanya Sagarika Meher, Christopher Ayala, Yuki Hironaka, Anubhav Sahu, Amol Inamdar, Deepnarayan Gupta, and Nobuyuki Yoshikawa, "Demonstration of a hybrid superconductor logic computation system using single flux quantum and adiabatic quantum-flux-parametron logic families," ASC2022, Hawaii, Oct. 2022.
24. Christopher Ayala, Tomoyuki Tanaka, Ro Saito, and Nobuyuki Yoshikawa, "An adiabatic quantum-flux-parametron block permutation unit for a superconductor SHA-3 cryptoprocessor," ASC2022, Hawaii, Oct. 2022.
25. Yuki Hironaka, Sukanya Sagarika Meher, Christopher Ayala, Yuxing He, Tomoyuki Tanaka, Mustapha Habib, Anubhav Sahu, Amol Inamdar, Deepnarayan Gupta, and Nobuyuki Yoshikawa, "Demonstration of interface circuits for adiabatic quantum-flux-parametron cell library using an eight-metal layer superconductor process," ASC2022, Hawaii, Oct. 2022.
26. Shohei Takagi, Tomoyuki Tanaka, Christopher Ayala, Nobuyuki Yoshikawa,

“Design of Energy-Efficient Adiabatic Quantum-Flux-Parametron Multiplier Familie,” ASC2022, Hawaii, Oct. 2022.

表彰

1. 低温工学・超電導学会 令和三年度 優秀発表賞
2. 電気学会 Young Researcher English Oral Presentation Award

Bibliography

[1] Technavio, “Data center market by component and geography - forecast and analysis 2022-2026.”

[2] “How to stop data centres from gobbling up the world’ s electricity.” `https://www.nature.com/articles/d41586-018-06610-y`.

[3] D. S. Holmes, A. L. Ripple, and M. A. Manheimer, “Energy-Efficient superconducting Computing—Power budgets and requirements,” *IEEE Trans. Appl. Supercond.*, vol. 23, pp. 1701610–1701610, June 2013.

[4] N. Takeuchi, D. Ozawa, Y. Yamanashi, and N. Yoshikawa, “An adiabatic quantum flux parametron as an ultra-low-power logic device,” *Superconductor Science and Technology*, vol. 26, p. 035010, jan 2013.

[5] N. Takeuchi, T. Yamae, C. L. Ayala, H. Suzuki, and N. Yoshikawa, “An adiabatic superconductor 8-bit adder with 24kBT energy dissipation per junction,” *Appl. Phys. Lett.*, vol. 114, p. 042602, Jan. 2019.

[6] A. Patterson, “Moore’s law could ride euv for 10 more years,” Sep 2021.

[7] J. Sevilla, L. Heim, A. Ho, T. Besiroglu, M. Hobbhahn, and P. Villalobos, “Compute trends across three eras of machine learning,” 2022.

[8] OpenAI, “Chatgpt: Optimizing language models for dialogue,” 2023. `https://openai.com/blog/chatgpt/`.

[9] I. Nagaoka, K. Ishida, M. Tanaka, K. Sano, T. Yamashita, T. Ono, K. Inoue, and A. Fujimaki, “Demonstration of a 52-GHz Bit-Parallel multiplier using Low-Voltage rapid Single-Flux-Quantum logic,” *IEEE Trans. Appl. Supercond.*, vol. 31, pp. 1–5, Aug. 2021.

[10] R. Kashima, I. Nagaoka, M. Tanaka, T. Yamashita, and A. Fujimaki, “64-GHz datapath demonstration for Bit-Parallel SFQ microprocessors based on a Gate-Level-Pipeline structure,” *IEEE Trans. Appl. Supercond.*, vol. 31, pp. 1–6, Aug. 2021.

[11] T. Kawaguchi and N. Takagi, “32-bit ALU with clockless gates for RSFQ Bit-Parallel processor,” *IEICE Transactions on Electronics*, vol. E105.C, pp. 245–250, June 2022.

[12] TOP500, “Green 500, november 2022,” 2022. `https://www.top500.org/lists/green500/2022/11/`.

[13] A. F. Kirichenko, I. V. Vernik, M. Y. Kamkar, J. Walter, M. Miller, L. R. Albu, and O. A. Mukhanov, “Ersfq 8-bit parallel arithmetic logic unit,” *IEEE Transactions on Applied Superconductivity*, vol. 29, no. 5, pp. 1–7, 2019.

[14] M. Dorojevets, Z. Chen, C. L. Ayala, and A. K. Kasperek, “Towards 32-bit energy-efficient superconductor rql processors: The cell-level design and analysis of key processing and on-chip storage units,” *IEEE Transactions on Applied*

Superconductivity, vol. 25, no. 3, pp. 1–8, 2015.

[15] C. L. Ayala, T. Tanaka, R. Saito, M. Nozoe, N. Takeuchi, and N. Yoshikawa, "MANA: A monolithic adiabatic integration architecture microprocessor using 1.4-zj/op unshunted superconductor josephson junction devices," *IEEE J. Solid-State Circuits*, vol. 56, pp. 1152–1165, Apr. 2021.

[16] Y. Harada and J. B. Green, "High-speed experiments on a qfp-based comparator for adcs with 18-ghz sample rate and 5-ghz input frequency," *IEEE Transactions on Applied Superconductivity*, vol. 2, pp. 21–25, March 1992.

[17] M. Hosoya, W. Hioe, K. Takagi, and E. Goto, "Operation of a 1-bit quantum flux parametron shift register (latch) by 4-phase 36-ghz clock," *IEEE Transactions on Applied Superconductivity*, vol. 5, pp. 2831–2834, June 1995.

[18] W. Hioe, M. Hosoya, S. Kominami, H. Yamada, R. Mita, and K. Takagi, "Design and operation of a quantum flux parametron bit-slice alu," *Applied Superconductivity, IEEE Transactions on*, vol. 5, pp. 2992 – 2995, 07 1995.

[19] H. L. Ko and G. S. Lee, "Noise analysis of the quantum flux parametron," *IEEE Transactions on Applied Superconductivity*, vol. 2, pp. 156–164, Sep. 1992.

[20] N. Takeuchi, T. Yamae, C. L. Ayala, H. Suzuki, and N. Yoshikawa, "Adiabatic Quantum-Flux-Parametron: A tutorial review," *IEICE Trans. Electron.*, vol. E105.C, pp. 251–263, June 2022.

[21] D. E. McCumber, "Effect of ac impedance on dc voltage ‐ current characteristics of superconductor weak ‐ link junctions," *Journal of Applied Physics*, vol. 39, no. 7, pp. 3113–3118, 1968.

[22] N. Takeuchi, T. Yamae, S. Hideo, and Y. Nobuyuki, "超低電力コンピューティングのための断熱磁束量子パラメトロンのパラメータ設計法," **電子情報通信学会論文誌** *C*, vol. 11, pp. 329–338, nov 2022.

[23] K. Fang, N. Takeuchi, T. Ando, Y. Yamanashi, and N. Yoshikawa, "Multi-excitation adiabatic quantum-flux-parametron," *Journal of Applied Physics*, vol. 121, no. 14, p. 143901, 2017.

[24] N. Takeuchi, M. Nozoe, Y. He, and N. Yoshikawa, "Low-latency adiabatic superconductor logic using delay-line clocking," *Applied Physics Letters*, vol. 115, no. 7, p. 072601, 2019.

[25] R. Saito, C. L. Ayala, and N. Yoshikawa, "Buffer reduction via n-phase clocking in adiabatic quantum-flux-parametron benchmark circuits," *IEEE Trans. Appl. Supercond.*, pp. 1–1, 2021.

[26] N. Takeuchi, Y. Yamanashi, and N. Yoshikawa, "Adiabatic quantum-flux-parametron cell library adopting minimalist design," *Journal of Applied Physics*, vol. 117, no. 17, p. 173912, 2015.

[27] 山﨑祐一, 山栄大樹, 竹内尚輝, and 吉川信行, "断熱量子磁束パラメトロン回路を用いた 5 入力 majority ゲートの研究," in **信学技報**, vol. 119 of *SCE2019-20*, (茨城), pp. 67–70, 8 月 2019. 2019 年 8 月 9 日 (金) 産業技術総合研究所 (SCE).

[28] T. Yamae, N. Takeuchi, and N. Yoshikawa, "Systematic method to evaluate energy dissipation in adiabatic quantum-flux-parametron logic," *J. Appl. Phys.*, vol. 126, p. 173903, Nov. 2019.

[29] L. Amarú, P. Gaillardon, A. Chattopadhyay, and G. De Micheli, "A sound and complete axiomatization of majority-n logic," *IEEE Transactions on Computers*, vol. 65, pp. 2889–2895, Sep. 2016.

[30] L. Amarù, E. Testa, M. Couceiro, O. Zografos, G. De Micheli, and M. Soeken, "Majority logic synthesis," in *Proceedings of the International Conference on Computer-Aided Design*, ICCAD ' 18, (New York, NY, USA), Association for Computing Machinery, 2018.

[31] T. Tanaka, C. L. Ayala, and N. Yoshikawa, "A 16-bit parallel prefix carry Look-Ahead Kogge-Stone adder implemented in adiabatic Quantum-Flux-Parametron logic," *IEICE Transactions on Electronics*, vol. E105.C, pp. 270–276, June 2022.

[32] C. L. Ayala, R. Saito, T. Tanaka, O. Chen, N. Takeuchi, Y. He, and N. Yoshikawa, "A semi-custom design methodology and environment for implementing superconductor adiabatic quantum-flux-parametron microprocessors," *Supercond. Sci. Technol.*, vol. 33, p. 054006, Mar. 2020.

[33] Y.-C. Chang, H. Li, O. Chenht, Y. Wang, N. Yoshikawa, and T.-Y. Ho, "ASAP: an analytical strategy for AQFP placement," in *Proceedings of the 39th International Conference on Computer-Aided Design*, no. Article 134 in ICCAD '20, (New York, NY, USA), pp. 1–7, Association for Computing Machinery, Nov. 2020.

[34] T. Tanaka, C. L. Ayala, Q. Xu, R. Saito, and N. Yoshikawa, "Fabrication of adiabatic quantum-flux-parametron integrated circuits using an automatic placement tool based on genetic algorithms," *IEEE Transactions on Applied Superconductivity*, vol. 29, pp. 1–6, Aug 2019.

[35] Y. Murai, C. L. Ayala, N. Takeuchi, Y. Yamanashi, and N. Yoshikawa, "Development and demonstration of routing and placement eda tools for large-scale adiabatic quantum-flux-parametron circuits," *IEEE Transactions on Applied Superconductivity*, vol. 27, pp. 1–9, Sep. 2017.

[36] S. C. Lo, A. J. Barker, S. R. Whiteley, E. Mlinar, J. Chen, D. Wu, and K. Singhal, "Simulation methodology for timing analysis and design optimization in digital superconducting electronics," in *2022 23rd International Symposium on Quality Electronic Design (ISQED)*, pp. 33–38, Apr. 2022.

[37] E. S. Fang and T. Van Duzer, "A josephson integrated circuit simulator (JSIM) for superconductive electronics application," in *Extended Abstracts of Int. Superconductivity Electronics Conf.*, pp. 407–410, June 1989.

[38] J. A. Delport, K. Jackman, P. l. Roux, and C. J. Fourie, "JoSIM—Superconductor SPICE simulator," *IEEE Trans. Appl. Supercond.*, vol. 29, pp. 1–5, Aug. 2019.

[39] R. Freeman, J. Kawa, and K. Singhal, "Synopsys' journey to enable TCAD and EDA tools for superconducting electronics," 2020.

[40] C. J. Fourie, "Full-gate verification of superconducting integrated circuit layouts with inductex," *IEEE Transactions on Applied Superconductivity*, vol. 25, no. 1, pp. 1–9, 2015.

[41] S. K. Tolpygo, V. Bolkhovsky, T. J. Weir, A. Wynn, D. E. Oates, L. M. Johnson, and M. A. Gouker, "Advanced fabrication processes for superconducting very large-scale integrated circuits," *IEEE Transactions on Applied Superconductivity*, vol. 26, no. 3, pp. 1–10, 2016.

[42] MIT Lincoln Laboratory, *MIT LL 100 $\mu A/\mu m2$ Superconductor Electronics Fabrication Process SFQ5ee Design Guide*, Dec. 2021.

[43] Y. He, C. L. Ayala, N. Takeuchi, T. Yamae, Y. Hironaka, A. Sahu, V. Gupta,

A. Talalaevskii, D. Gupta, and N. Yoshikawa, “A compact AQFP logic cell design using an 8-metal layer superconductor process,” *Supercond. Sci. Technol.*, vol. 33, p. 035010, Feb. 2020.

[44] N. Takeuchi, S. Nagasawa, F. China, T. Ando, M. Hidaka, Y. Yamanashi, and N. Yoshikawa, “Adiabatic quantum-flux-parametron cell library designed using a 10 kA cm-2niobium fabrication process,” *Superconductor Science and Technology*, vol. 30, p. 035002, jan 2017.

[45] N. Takeuchi, H. Suzuki, C. J. Fourie, and N. Yoshikawa, “Impedance design of excitation lines in adiabatic Quantum-Flux-Parametron logic using InductEx,” *IEEE Trans. Appl. Supercond.*, vol. 31, pp. 1–5, Aug. 2021.

[46] M. C. Hansen, H. Yalcin, and J. P. Hayes, “Unveiling the ISCAS-85 benchmarks: a case study in reverse engineering,” *IEEE Des. Test Comput.*, vol. 16, pp. 72–80, July 1999.

[47] N. Tsuji, N. Takeuchi, Y. Yamanashi, T. Ortlepp, and N. Yoshikawa, “Majority Gate-Based feedback latches for adiabatic quantum flux parametron logic,” *IEICE Transactions on Electronics*, vol. E99.C, no. 6, pp. 710–716, 2016.

[48] N. Tsuji, N. Takeuchi, T. Narama, T. Ortlepp, Y. Yamanashi, and N. Yoshikawa, “Magnetically coupled quantum-flux-latch with wide operation margins,” *Supercond. Sci. Technol.*, vol. 28, p. 115013, Sept. 2015.

[49] T. Yamae, N. Takeuchi, and N. Yoshikawa, “Binary counters using adiabatic Quantum-Flux-Parametron logic,” *IEEE Trans. Appl. Supercond.*, vol. 31, pp. 1–5, Mar. 2021.

[50] R. Saito, C. L. Ayala, O. Chen, T. Tanaka, T. Tamura, and N. Yoshikawa, “Logic synthesis of sequential logic circuits for adiabatic Quantum-Flux-Parametron logic,” *IEEE Trans. Appl. Supercond.*, vol. 31, pp. 1–5, Aug. 2021.

[51] S. S. Meher, J. Ravi, M. E. Çelik, S. Miller, A. Sahu, A. Talalaevskii, and A. Inamdar, “Superconductor standard cell library for advanced EDA design,” *IEEE Trans. Appl. Supercond.*, vol. 31, pp. 1–7, Aug. 2021.

[52] Y. Hashimoto, S. Yorozu, Y. Kameda, and V. K. Semenov, “A design approach to passive interconnects for single flux quantum logic circuits,” *IEEE Trans. Appl. Supercond.*, vol. 13, pp. 535–538, June 2003.

[53] Y. Yamazaki, N. Takeuchi, and N. Yoshikawa, “A compact interface between adiabatic Quantum-Flux-Parametron and rapid Single-Flux-Quantum circuits,” *IEEE Trans. Appl. Supercond.*, vol. 31, pp. 1–5, Aug. 2021.

[54] C. L. Ayala, N. Takeuchi, Y. Yamanashi, T. Ortlepp, and N. Yoshikawa, “Majority-logic-optimized parallel prefix carry look-ahead adder families using adiabatic quantum-flux-parametron logic,” *IEEE Transactions on Applied Superconductivity*, vol. 27, pp. 1–7, June 2017.

[55] H. Cong, M. Li, and M. Pedram, “An 8-b multiplier using single-stage full adder cell in single-flux-quantum circuit technology,” *IEEE Transactions on Applied Superconductivity*, vol. 31, no. 6, pp. 1–10, 2021.

[56] Y. He, N. Takeuchi, and N. Yoshikawa, “Low-latency power-dividing clocking scheme for adiabatic quantum-flux-parametron logic,” *Appl. Phys. Lett.*, vol. 116, p. 182602, May 2020.

[57] N. Takeuchi, M. Nozoe, Y. He, and N. Yoshikawa, “Low-latency adiabatic superconductor logic using delay-line clocking,” *Appl. Phys. Lett.*, vol. 115, p. 072601,

Aug. 2019.

[58] T. Yamae, N. Takeuchi, and N. Yoshikawa, “Adiabatic quantum-flux-parametron with delay-line clocking: logic gate demonstration and phase skipping operation,” *Supercond. Sci. Technol.*, vol. 34, p. 125002, Oct. 2021.

[59] N. A. Sherwani, *Algorithms for VLSI Physical Design Automation.* USA: Kluwer Academic Publishers, 1993.

[60] A. Hashimoto and J. Stevens, “Wire routing by optimizing channel assignment within large apertures,” in *Proceedings of the 8th Design Automation Workshop*, DAC ’ 71, (New York, NY, USA), p. 155 – 169, Association for Computing Machinery, 1971.

[61] S. M. S. . H. Youssef, “Channel routing,” 1995.

[62] H. H. Chen and E. S. Kuh, “Glitter: A gridless Variable-Width channel router,” *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, vol. 5, pp. 459–465, Oct. 1986.

[63] K. Asai, N. Takeuchi, H. Suzuki, Y. Yamanashi, and N. Yoshikawa, “Transmission line effects of long Gate-to-Gate interconnections in adiabatic Quantum-Flux-Parametron logic circuits,” *IEEE Trans. Appl. Supercond.*, vol. 32, pp. 1–7, Oct. 2022.

[64] M. Dorojevets, C. L. Ayala, N. Yoshikawa, and A. Fujimaki, “16-bit Wave-Pipelined Sparse-Tree RSFQ adder,” *IEEE Trans. Appl. Supercond.*, vol. 23, pp. 1700605–1700605, June 2013.

[65] A. Y. Herr, Q. P. Herr, O. T. Oberg, O. Naaman, J. X. Przybysz, P. Borodulin, and S. B. Shauck, “An 8-bit carry look-ahead adder with 150 ps latency and sub-microwatt power dissipation at 10 GHz,” *J. Appl. Phys.*, vol. 113, p. 033911, Jan. 2013.

[66] R. Celis-Cordova, A. O. Orlov, T. Lu, J. M. Kulick, and G. L. Snider, “Design of a 16-bit adiabatic microprocessor,” in *2019 IEEE International Conference on Rebooting Computing (ICRC)*, pp. 1–4, Nov. 2019.

[67] A. Stillmaker and B. Baas, “Scaling equations for the accurate prediction of cmos device performance from 180nm to 7nm,” *Integration*, vol. 58, pp. 74–81, 2017.

[68] D. E. Kirichenko, S. Sarwana, and A. F. Kirichenko, “Zero static power dissipation biasing of RSFQ circuits,” *IEEE Trans. Appl. Supercond.*, vol. 21, pp. 776–779, June 2011.

[69] N. Takeuchi, K. Arai, and N. Yoshikawa, “Directly coupled adiabatic superconductor logic,” *Supercond. Sci. Technol.*, vol. 33, p. 065002, May 2020.

[70] R. Ishida, N. Takeuchi, T. Yamae, and N. Yoshikawa, “Design and demonstration of directly coupled Quantum-Flux-Parametron circuits with optimized parameters,” *IEEE Trans. Appl. Supercond.*, vol. 31, pp. 1–5, Aug. 2021.

[71] L. Schindler, R. van Staden, C. J. Fourie, C. L. Ayala, J. A. Coetzee, T. Tanaka, R. Saito, and N. Yoshikawa, “Standard cell layout synthesis for row-based placement and routing of rsfq and aqfp logic families,” in *2019 IEEE International Superconductive Electronics Conference (ISEC)*, pp. 1–5, 2019.

[72] C. J. Fourie, C. L. Ayala, L. Schindler, T. Tanaka, and N. Yoshikawa, “Design and characterization of track routing architecture for rsfq and aqfp circuits in a multilayer process,” *IEEE Transactions on Applied Superconductivity*, vol. 30, no. 6, pp. 1–9, 2020.

[73] S. K. Tolpygo, V. Bolkhovsky, T. J. Weir, A. Wynn, D. E. Oates, L. M. Johnson, and M. A. Gouker, “Advanced fabrication processes for superconducting very Large-Scale integrated circuits,” *IEEE Trans. Appl. Supercond.*, vol. 26, pp. 1–10, Apr. 2016.

[74] A. M. S. Abdelhadi, *Architecture of block-RAM-based massively parallel memory structures : multi-ported memories and content-addressable memories.* PhD thesis, University of British Columbia, 2016.

[75] X. Peng, Y. Yamanashi, N. Yoshikawa, A. Fujimaki, N. Takagi, K. Takagi, and M. Hidaka, “Design and high-speed demonstration of single-flux-quantum bit-serial floating-point multipliers using a 10ka/cm2 nb process,” *IEICE trans. electron.*, vol. E97.C, no. 3, pp. 188–193, 2014.

[76] S. Mach, F. Schuiki, F. Zaruba, and L. Benini, “FPnew: An Open-Source multiformat Floating-Point unit architecture for Energy-Proportional transprecision computing,” *IEEE Trans. Very Large Scale Integr. VLSI Syst.*, vol. 29, pp. 774–787, Apr. 2021.