Self-Timed Parallel Adders based on DI RSFQ Primitives

Y. Kameda, S. V. Polonsky†, M. Maezawa‡, and T. Nanya
Research Center for Advanced Science and Technology, The University of Tokyo, Meguro-ku, Tokyo 153-8904 Japan
†Physics department, SUNY Stony Brook, Stony Brook, NY 11794 USA
‡Electron Devices Division, Electrotechnical Laboratory, Tsukuba-shi, Ibaraki 305-0045 Japan

***Abstract*—We present two versions of self-timed pipelined parallel carry-look-ahead adders. The adders are designed based on delay-insensitive (DI) rapid single-flux-quantum (RSFQ) primitives. Basic binary gates employ dual-rail encoded data, which include timing information in themselves. One version uses wave pipelining and the other delay-insensitive pipelining with a request-acknowledge data transfer protocol. We show simulation results of 4 to 32-bit adders and their sensitivity to delay variations. Two design schemes are compared in terms of area, speed, robustness, interface and design process for large systems.**

I. Introduction

Increasing device speed causes clock distribution and timing problems. Self-timed or asynchronous design solves these problems by removing a global clock. Asynchronous circuits are designed based on a delay assumption about primitives and wires. The severest one is delay-insensitive (DI) assumption, in which both primitives and wires have finite but unbounded delays. This assumption is effective as wire delays are comparable to primitive delays.

Rapid single-flux-quantum (RSFQ) device [1], one of the superconductive Josephson-junction devices, has a potential for an ultra fast digital device. In RSFQ circuits, data are represented by short voltage pulses instead of voltage levels. Classical synchronous RSFQ circuits use a clock signal to translate a pulse on a data line in a "clock window" as logic "1" and no pulse as logic "0." Because an improper arrival order of data and clock pulses leads to erroneous data transfer, careful delay estimation and clock design are required [2]. Facing timing problems, RSFQ technology has been paying an attention to an asynchronous approach. Dual-rail data encoding enables clock free data transfer [3], [4]. In the dual-rail scheme, a pair of (true- and false-) data lines carry 1-bit binary information. The propagation of a pulse on the true-line or the false-line represents logic "1" or "0" respectively. No race occurs because only one pulse propagates either true- or false-line during 1-bit data transfer.

Manuscript received September 15, 1998.
This work was supported in part by the Ministry of ESSC under Grant-in-aid for Scientific research No. (B) 09480049, by STARC, by DARPA/NSA/NASA via JPL, and by the DoD's University Research Initiative via AFOSR.

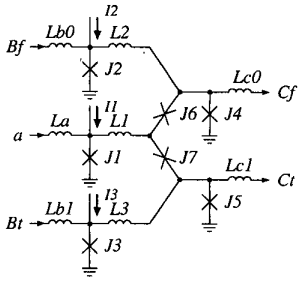

Fig. 1. RSFQ implementation of 1×2-Join.

For dual-rail DI circuits, efficient primitives have been proposed [5]. A 1×2-Join serves as a destructive read-out register with a dual-rail data input/output and a request input. Note that the order of a data pulse and a request pulse is arbitrary. The word "Join" is used in the sense that two pulses meet in the cell. An RSFQ implementation of the 1×2-Join is depicted in Fig. 1. It consists of two pairs of interferometers for output Cf and Ct. One pair of interferometers ($J2$,$L2$,$J4$) for input Bf and ($J1$,$L1$,$J6$,$J4$) for input a are coupled through $J4$ with output Cf. Similarly the other pair ($J3$,$L3$,$J5$) for input Bt and ($J1$,$L1$,$J7$,$J5$) for input a are coupled through $J5$ with output Ct. In its initial state, no persistent superconducting current is present in the interferometer loops. Suppose it receives inputs at Bf and a in this order. The first input at Bf triggers $J2$, but the pulse generated is insufficient to trigger $J4$. Thus the flux quantum arrived at Bf is stored in the loop ($J2$, $L2$, $J4$). When a pulse arrives at a, $J1$ is tripped and a current loop through $J4$ is formed. The addition of current through $J4$ exceeds its critical current and $J4$ trips, resulting in output pulse generation at Cf. It also causes $J7$ to trip, which isolates upper and lower part of the circuit. After an output pulse is emitted at Cf, no persistent superconducting current is present, which means the circuit is now in the initial state. The operation for the other pair or order of inputs is similar. Note that input pair (Bf, Bt) is not allowed since it is invalid in the dual-rail data representation.

A 2×2-Join can execute any 2-input binary operation in combination with mergers (confluence buffers). The 2×2-Join has two dual-rail inputs and four outputs one of which produces a pulse in response to four input patterns. Fig. 2 depicts an RSFQ implementation of the 2×2-Join. A pair of interferometers ($JA0$,$LA0$,$J00A$,$J00$) for input

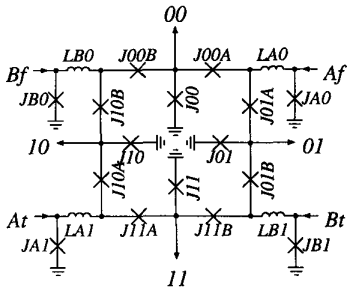

Fig. 2. RSFQ implementation of 2×2-Join.

Af and ($JB0$,$LB0$,$J00B$,$J00$) for input Bf are coupled through $J00$ with output 00. Similarly the circuit has three pairs for the other three outputs. Pulses arrived at Af and Bf trigger $JA0$ and $JB0$ respectively and consequently $J00$, emitting an output pulse at 00. Two junctions $J01A$ and $J10B$ isolate these interferometers from the others.

Using these primitives, two pipelining schemes are possible: wave pipelining [6] and DI pipelining [7]. The former uses no pipeline registers to reduce area and delay. Several data can exist between registers. On the other hand, the latter uses pipeline registers to implement a request-acknowledge data transfer protocol in order that it may tolerate delay variations.

A pipelined circuit is evaluated by two measures regarding to speed. *Latency* is defined as the delay from the input to the output. A more significant value is *cycle time*, which is defined as the time interval of consecutive input or output data. Throughput is the reciprocal of cycle time. Thus, shorter cycle time gives higher throughput.

In this paper, using the above two schemes, we design two versions of pipelined parallel carry-look-ahead adders. We show simulation results of 4 to 32-bit adders and their sensitivity to delay variations. Two design schemes are compared in terms of area, speed, robustness, and design process for large systems.

II. Wave pipelining

Wave pipelining is a technique to achieve high throughput without the use of pipeline registers on critical paths. To prevent data skewing due to different data propagation delays, all the data paths in a pipeline stage are designed to have equal delays. The delay of one stage is set to the maximum delay of all the paths in the stage. For this purpose, matching delays are inserted. Serial connection of these stages compose another stage or a larger circuit. Although all the data paths in a single stage are designed to have equal stage delays, one stage delay could be different from another.

Fig. 3 illustrates a dual-rail AND gate for wave pipelining. Dual-rail inputs A and B are connected to a 2×2-Join. It generates only one pulse out of four outputs in response to any combination of input pulses. Three

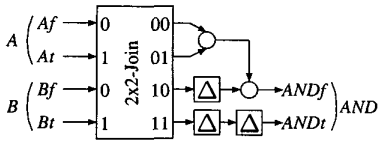

Fig. 3. AND gate for wave pipelining.

outputs, 00, 01, and 10 are merged into primary output $ANDf$ while output 11 is directed to primary output $ANDt$. Matching delays (denoted by Δ) compensate the delays of the mergers.

Data can be fed as early as possible as long as the next data do not catch up with the previous ones, or in other words, data fronts do not overlap. Matching delays try to keep data fronts straight. The maximum delay of all the stages gives the theoretical minimum cycle time of the whole circuit. However, in practice, it is difficult or impossible to keep data fronts straight because global and local process variation shifts delays from their nominal values. Therefore, some timing margin should be added to the theoretical cycle time.

The advantages of this scheme include no races between pulses on data lines as a nature of dual-rail encoding, and no pipeline stages realizing high throughput. However, the circuit is vulnerable to timing violations resulting from delay variations of each primitive, especially for large circuits.

III. Delay-insensitive pipelining

Delay-insensitive (DI) pipelining employs a request-acknowledge data transfer protocol. A functional unit, which is a basic component with *request* and *acknowledge* signals, can send the result if it has received a *request* signal from the unit to which the result is to be sent. Otherwise it waits to send until it receives the *request* signal. After sending the result, the functional unit sends a *ready* signal back to the previous units from which input data come. The *ready* signal is connected to the request input of the previous unit, notifying that the receiver is empty and ready for accepting new data. If a functional unit have multiple data outputs, and therefore multiple *ready* signals corresponding to each output, C-elements (or coincidence junctions) rendezvous all the *ready* signals to generate a *request* signal for the unit.

Typically a functional unit comprises a 2×2-Join (as a binary operator), some mergers and a 1×2-Join (as a pipeline register). The minimum cycle time is given by the total delay of a 2×2-Join, mergers, a 1×2-Join, and another 1×2-Join in the previous stage.

Fig. 4 illustrates a dual-rail AND gate that follows the request-acknowledge protocol. The 2×2-Join and the mergers in the left half implement the AND function as described in Fig. 3. The 1×2-Join temporarily stores the result and releases it after receiving a pulse from $request_{AND}$. The forks (or splitters) and the mergers after the 1×2-Join generate acknowledge signals, $ready_A$

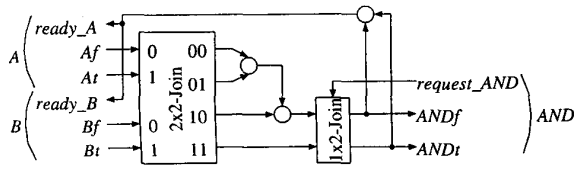

Fig. 4. AND gate for DI pipelining.

and $ready_B$. Because the 1×2-Join synchronizes data and control signals, no matching delays are necessary.

This handshake protocol realizes delay-insensitive systems i.e. the correctness of the operation is independent of both primitive and wire delays. Due to delay insensitivity, it is allowed to put a pipeline register between any two adjacent functional units. Extra pipeline register insertion can improve cycle time by dividing a long wire that deteriorates cycle time.

Robustness is an advantage of this scheme. A whole circuit operates correctly as long as each primitive that composes the circuit works correctly. Moreover, since there are no timing constrains at all, logic design and primitive cell design are independent of each other, achieving high modularity. The disadvantage lies in its area and control overhead (i.e. speed degradation). It requires three lines for 1-bit information including a control line while wave pipelining uses two lines per bit.

IV. Simulation

A. Adder structure

To compare the two methodologies we design several sizes of a parallel carry-look-ahead adder [8], which has a regular structure and small fan-outs. Fig. 5 illustrates a 4-bit carry-look-ahead adder. In general, the n-bit adder consists of $\log_2 n + 2$ levels of regular modules. i-th bit inputs $A(i)$ and $B(i)$ given, the first level (a column of squares) calculates a propagation signal $P(i) = A(i) \oplus B(i)$ and a generation signal $G(i) = A(i) \wedge B(i)$. The second to $(\log_2 n - 1)$-th levels compute a carry signal $C(i)$ from the $P(i)$ and the $G(i)$. A black processor (denoted by a black circle) computes $P' = PH \wedge PL$ and $G' = GH \vee GL \wedge PH$, where PH and GH have the same bit-index and PL and GL have a lower bit-index. A white processor (denoted by a white circle) copies input data to the outputs. The last level (a column of triangles) calculates a sum $F(i) = P(i) \oplus C(i-1)$.

B. Technology assumptions

Table I summarizes the assumptions about the prospective Josephson-junction technology used in the following simulations.

C. Analog simulation

First, primitives are designed and their parameters are optimized so that they have margins of ±30% on global bias current. Primitive delays are calculated for the global bias current factor (XI) of 0.7, 1.0 and 1.3. As XI gets larger, a primitive operates faster. This delay v.s. XI charactristic depends on primitives. Table II shows analog simulation results of five primitives used to compose adders. Note that all the primitives have margins of more than ±30%.

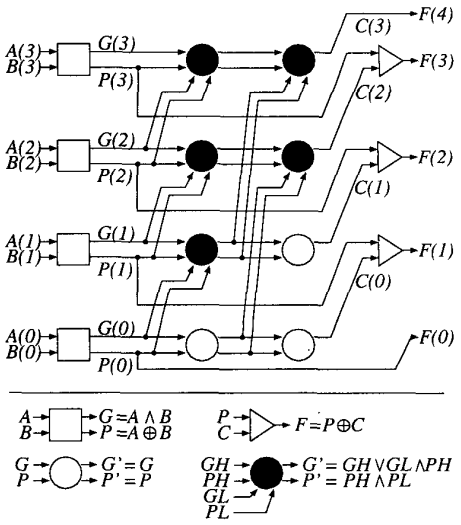

Fig. 5. Block diagram of 4-bit carry-look-ahead adder.

TABLE I
Josephson Junction Technology Assumptions

Minimum Josephson junction size	0.8μm×0.8μm
Critical current density	20KA/cm^2
Design density	10μm×10μm/JJ with wiring
Delay in microstrip lines	0.01ps/μm

D. Digital simulation

With increasing size and complexity of circuits analog simulatin consumes much time. Digital simulation is indispensable in the future design. For this reason, the behavior models for the primitives are written in hardware description languages (HDLs). The models include the delay v.s. XI charactristics. They report an error message if they receive an unspecified input sequence.

Lastly, all higher modules, including a test environment, are written also in HDLs. For each pipelining

TABLE II
Analog Simulation Results Of Primitives

Primitive	# of JJs	Delay (ps)		
		$XI = 0.7$	$XI = 1.0$	$XI = 1.3$
Fork	3	4.8	3.3	2.1
Merge	5	10.5	5.4	3.3
C-element	3	4.8	3.0	2.1
1×2-Join	7	5.1	2.7	1.5
2×2-Join	16	9.0	6.3	4.5

TABLE III
DIGITAL SIMULATION RESULTS OF BLACK PROCESSORS

Design methodology	# of JJs	Area (μm^2)	Latency (ps) $XI =$ 0.7	1.0	1.3	Cycle time (ps) $XI =$ 0.7	1.0	1.3
Wave pipelining	76	87^2	52	27	17	25	12	8
DI pipelining	209	144^2	69	34	22	49	22	13

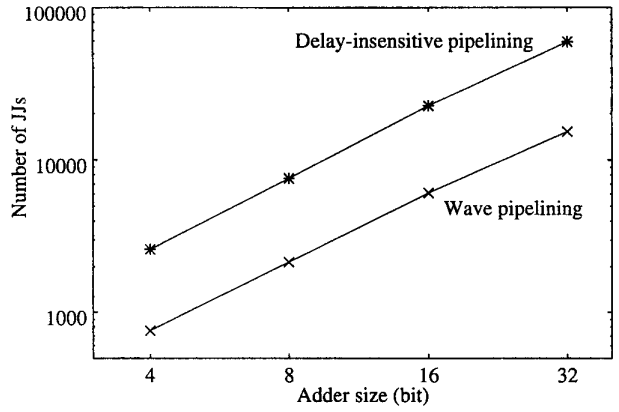

Fig. 6. Number of Josephson junctions in adders.

methodologies, we design four sizes of adders: 4, 8, 16, and 32-bit input widths. The delay in microstrip lines is also taken into consideration. The test environment generates random input patterns, feeds them one by one according to the protocol in question, and verifies the outputs. It also takes statistics on latency and cycle time at the same time.

Digital simulation results of black processors alone are shown in Table III. According to the technology assumptions in Table I, the areas are estimated from the number of required Josephson junctions. The latency and cycle time of output G' are presented because it is slower than output P' and affects the latency and cycle time of whole adders.

Fig. 6 shows the number of Josephson junctions in the adders. Figs. 7 and 8 plot average latency and cycle time for 500 random input patterns respectively with the vertical lines showing their variations. To calculate the cycle time of a wave-pipelined adder, we increase input data rate gradually until the adder can no longer opperate correctly. For DI-pipelined adders, we assume an ideal input source that immediately generates random data in response to a *request* singal, and an ideal output drain that consumes arriving data and sends an *acknowledge* signal with no dealy.

V. COMPARISON

A. Area

We use the number of required Josephson junctions for area comparison. Table III shows that the black processor for wave pipelining uses about three times as many junctions as that for DI pipelining. Therefore, the DI-pipelined adders use about three times as many junctions as the wave-pipelined ones as shown in Fig. 6. This large area overhead lies in the use of pipeline registers in each pipeline stage. A pair of Josephson transmission lines (JTLs) (2 junctions in total) serves as a temporary storage for wave pipelining while a 1×2-Join and control circuitry (14 junctions) do for DI pipelining.

Keeping delay insensitivity, a way to reduce the area of a DI-pipelined adder is to lessen the number of pipeline stages and thus pipeline registers, which improves latency but deteriorates cycle time.

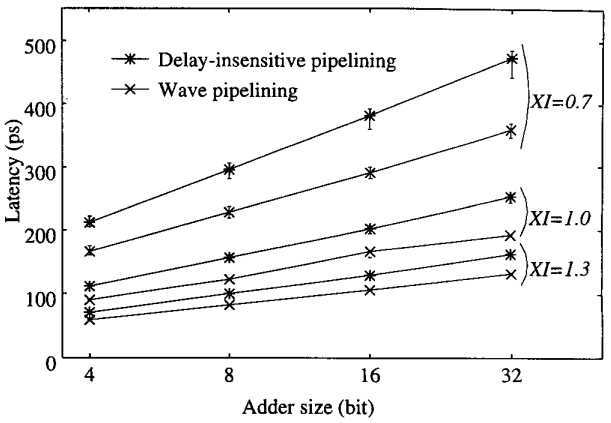

Fig. 7. Latency of adders.

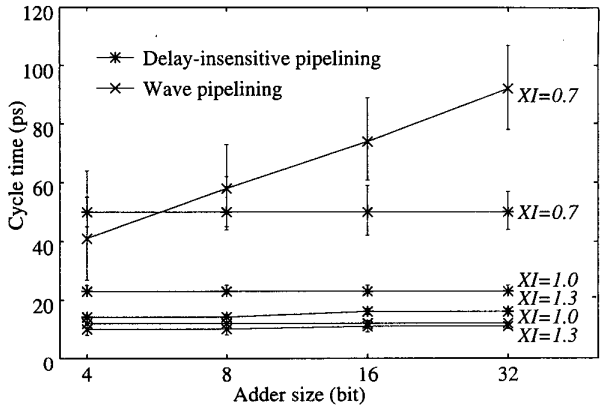

Fig. 8. Cycle time of adders.

B. Latency

The logic depth of a circuit determines its latency. Hence, latency can be roughly estimated to be a delay accumulation from the input to the output. For example, the 4-bit adder has four levels of processors (see Fig. 5). The latency of a black processor (in the middle two levels) is shown in Table III. The first and last levels have

approximately a half latency of the black processor since their logic depth is a half of the black processor's. Consequently, the latency of the adder is roughly estimated to be the triple of the black processor's latency. As the input size doubles, the adder requires one more level of black processors, and accordingly the latency increases. These rough estimations agree with the digital simulation results shown in Fig. 7.

The latency of the wave-pipelined adders is about 80% of that of the DI-pipelined ones. Because, in a DI-pipelined adder, signals go through a pipeline register in each stage, it adds extra time to the latency.

C. Cycle time

We first focus on the cycle time for wave pipelining. If all the delays are set to their nominal values (i.e. XI=1.0), wave pipelining has shorter cycle time for all the sizes than DI pipelining. The cycle time equals to the theoretical minimum, which is the maximum stage delay, in this case, the delay of the AND gate. Although XI changes to 1.3 and primitives can operate faster, cycle time improvement is small. If XI decreases to 0.7, the cycle time increases with the adder size. Moreover the adders cannot operate as fast as the black processor alone (see Table III). Delay variation from the nomial values causes larger difference between maximum and minimum delay in one pipeline stage. Therefore skewing data fronts tend to overlap. A simple way to circumvent this cycle time digradation is to put pipeline registers that keep data fronts straight at the cost of additional area and latency. Although it requiers almost the same area as DI pipelining, circuit size has a less influence on cycle time.

In contrast, the cycle time for DI pipelining is not affected by the circuit size because of localized timing signals. The cicle time of the whole adders and the black processor for DI pipelining is the same. Moreover, the cycle time varies in accordance with XI variations. It reduces by 30% with the increase of XI from 1.0 to 1.3 whereas the cycle time for wave pipeling does by less than 10% for 32-bit adders.

D. Robustness

Asynchronous dual-rail data encoding eliminates races between data and clock signals inherent to synchronous systems. Timing errors can occur within a single data front (intra-data front) as well as between adjacent data fronts (inter-data fronts).

In this sense, a dual-rail asynchronous system is robuster than a synchronous counterpart. Data front overlap in a wave-pipelined circuit is the only source of timing errors. In other words, timing erros occur only between adjacent data fronts (inter-data fronts). However, they can be solved by decreasing data input rate.

For a wave-pipelined adder the probability of inter-data-front error can be estimated analytically [9]. Even a 64-bit adder with resynchronization buffers does not suffer throughput degradation caused by skewing of data fronts for the technologies with Josephson junction variance $\sigma_{JJ} < 3.6\%$.

TABLE IV
OCCURRENCE OF TIMING ERRORS

Design methodology	Timing errors
Clocked pipelining	Inter- and intra-data front
Wave pipelining	Inter-data front
Delay-insensitive pipelining	None

The request-acknowledge protocol enhances the robustness and realizes delay-insensitive data transfer at the cost of area and control overhead.

Table IV summarizes the occurrence of timing errors for three design methodologies.

E. Interface

In a self-timed system, as the vertical lines in Fig. 8 shows, cycle time (or data output rate in this case) varies even though data input rate remains constant. In short the both pipelines are elastic. Some interface technique is necessary to absorb output rate variation to connect one circuit to others.

Several stages of first-in-first-out buffers or FIFOs [10] play its role in a wave-pipelined system. The FIFOs temporarily store the outputs from a circuit in a queue and then make a straight data front.

Connecting handshake circuits is simple: data lines to data lines and *request* signal lines to *acknowledge* signal lines. The request-acknowledge protocol guarantees correct data transfer. No difference exists between internal and external circuit connection.

F. Design process

Design process simplicity is important for large and complicated systems though its quantitative measurement is difficult.

Design of circuits based on the both schemes starts with primitive cell design, which includes layout and parameter optimization. Next, in the case of wave pipelining, matching delays such as JTLs and pipeline registers are inserted. Digital simulation checks timing violations resulting from imbalanced delays. It also calculates the speed and the sensitivity to delay variations.

On the other hand, in the case of DI pipelining, after cell design next comes the design of functional units which are the smallest modules with a *request* input and an *acknowledge* output. Digital simulation is performed for speed estimation, not for timing verification since the system does not have any timing constrains. Due to delay insensitivity, primitive cell design and circuit design are independent of each other.

VI. Conclusion

RSFQ dual-rail gates can be a viable alternative to classical clocked RSFQ gates for large and complex RSFQ systems in the future. At the expense of approximately doubling hardware cost they offer high throughput, tolerance to delay variations, and exceptional ease of design. The choice of wave pipelining or DI pipelining strongly depends on delay controllability. The former is suitable for a small system where delays are manageable while the latter is effective for large and complicated systems.

A natural design is the combination of the two. A large system is divided into small or simple subsystems in which delays are predictable. Wave pipelining methodology is applied to these subsystems. All the subsystems should be connected following the request-acknowledge protocol because the global delays are uncontrollable or unpredictable.

Acknowledgment

The authors would like to thank Prof. K. Likharev and his RSFQ research group for valuable discussions.

References

[1] K. K. Likharev and V. K. Semenov, "RSFQ logic/memory family: A new Josephson-junction technology for sub-terahertz-clock frequency digital systems," *IEEE Trans. Appl. Supercond.*, vol. 1, no. 1, pp. 3–28, Mar. 1991.

[2] K. Gaj, E. G. Friedman, and M. J. Feldman, "Timing of multi-gigahertz rapid single flux quantum digital circuits," *IEEE Journal of VLSI Signal Processing*, vol. 16, pp. 247–276, 1997.

[3] I. Kurosawa, H. Nakagawa, M. Aoyagi, M. Maezawa, Y. Kameda, and T. Nanya, "A basic circuit for asynchronous superconductive logic using RSFQ gates," in *Extended Abstracts of ISEC 95*, Sept. 1995, pp. 204–206.

[4] Z. J. Deng, S. R. Whiteley, and T. Van Duzer, "Data-driven self-timing of RSFQ digital integrated circuits," in *Extended Abstracts of ISEC 95*, Sept. 1995, pp. 189–191.

[5] P. Patra, S. Polonsky, and D. S. Fussell, "Delay insensitive logic for RSFQ superconductor technology," in *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*. Apr. 1997, pp. 42–53, IEEE Computer Society Press.

[6] L. W. Cotten, "Maximum-rate pipeline systems," in *AFIPS Proc. of Spring Join Computer Conference*, May 1969, vol. 34, pp. 581–586.

[7] Y. Kameda, S. Polonsky, M. Maezawa, and T. Nanya, "Primitive-level pipelining method on delay-insensitive model for RSFQ pulse-driven logic," in *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, 1998, pp. 262–273.

[8] R. P. Brent and H. T. Kung, "A regular layout for parallel adders," *IEEE Trans. on Computers*, vol. C-31, pp. 264–260, Mar. 1982.

[9] S. Polonsky, "Performance analysis of 64-bit carry-look-ahead adder based on data-driven dual-rail boolean gates," Tech. Rep. 06, HTMT RSFQ System Group, SUNY at Stony Brook, Jun. 1998 (unpublished).

[10] M. Maezawa and S. Polonsky, "Dual-rail RSFQ shift register on delay-insensitive model and its applications," Tech. Rep. of IEICE, SCE97-29, pp. 19–24, Nov. 1997.