Journal of VLSI Signal Processing 16, 247–276 (1997)
© 1997 Kluwer Academic Publishers. Manufactured in The Netherlands.

Timing of Multi-Gigahertz Rapid Single Flux Quantum Digital Circuits

KRIS GAJ, EBY G. FRIEDMAN AND MARC J. FELDMAN
Department of Electrical Engineering, University of Rochester, Rochester, New York 14627

Received November 24, 1996; Revised December 18, 1996

Abstract. Rapid Single Flux Quantum (RSFQ) logic is a digital circuit technology based on superconductors that has emerged as a possible alternative to advanced semiconductor technologies for large scale ultra-high speed, very low power digital applications. Timing of RSFQ circuits at frequencies of tens to hundreds of gigahertz is a challenging and still unresolved problem. Despite the many fundamental differences between RSFQ and semiconductor logic at the device and at the circuit level, timing of large scale digital circuits in both technologies is principally governed by the same rules and constraints. Therefore, RSFQ offers a new perspective on the timing of ultra-high speed digital circuits.

This paper is intended as a comprehensive review of RSFQ timing, from the viewpoint of the principles, concepts, and language developed for semiconductor VLSI. It includes RSFQ clocking schemes, both synchronous and asynchronous, which have been adapted from semiconductor design methodologies as well as those developed specifically for RSFQ logic. The primary features of these synchronization schemes, including timing equations, are presented and compared.

In many circuit topologies of current medium to large scale RSFQ circuits, single-phase synchronous clocking outperforms asynchronous schemes in speed, device/area overhead, and simplicity of the design procedure. Synchronous clocking of RSFQ circuits at multigigahertz frequencies requires the application of non-standard design techniques such as pipelined clocking and intentional non-zero clock skew. Even with these techniques, there exist difficulties which arise from the deleterious effects of process variations on circuit yield and performance. As a result, alternative synchronization techniques, including but not limited to asynchronous timing, should be considered for certain circuit topologies. A synchronous two-phase clocking scheme for RSFQ circuits of arbitrary complexity is introduced, which for critical circuit topologies offers advantages over previous synchronous and asynchronous schemes.

1. Introduction

The recent achievements of superconductive circuits using *Rapid Single Flux Quantum* (*RSFQ*) logic make this technology a possible candidate to first cross the boundary of 100 GHz clock frequency in a large scale digital circuit. The success of RSFQ circuits is in part due to the unique convention used to represent digital information. Rather than using steady voltage levels, RSFQ circuits use quantized voltage pulses to transmit binary logic state information. This logic scheme has necessarily led to new timing concepts and techniques in order to coordinate the operation of the gates and sub-circuits at multigigahertz frequencies. Nevertheless, the similarities to semiconductor voltage-state timing are strong, and the two technologies can be discussed in the same language.

This paper is written with the intention that both semiconductor and superconductor communities will benefit from the mutual exchange of ideas on the timing of high speed large scale digital circuits. RSFQ designers inherit a broad range of techniques and methods developed over many years by VLSI semiconductor circuit designers. The capability of RSFQ technology offers the semiconductor community an opportunity to be made aware about existing pitfalls in the design and

implementation of clocking schemes at multi-gigahertz clock frequencies, and to benefit from innovative timing schemes that have been proved to work correctly at frequencies as yet unavailable in semiconductor technologies.

1.1. Advantages of RSFQ Logic

The basic concepts and recent progress in RSFQ logic are reviewed in [1–4]. The most significant advantages are high speed, low power, and the potential for large scale integration. Today, relatively complex circuits consisting of roughly 100 clocked gates have been designed and tested at frequencies about 10 GHz by several groups [3, 5–9]. The simplest digital circuit has been demonstrated at 370 GHz [10]. The on-chip power dissipation is negligible, below 1 μW per gate, so that ultra-high device density may eventually be realized. Additional advantages are that RSFQ circuits require only a dc power supply, can employ either an external or an internal clock source, have a negligible bit error rate [11], and the fabrication technology is fairly simple. The primary disadvantages include the necessity of helium cooling, and a relatively underdeveloped fabrication infrastructure. If one recognizes that the standard feature size in today's still primitive superconductive technology is about ten times larger compared to a state-of-the art CMOS process, it is impressive that RSFQ circuits still offer two orders of magnitude speed-up in clock frequency and three orders of magnitude smaller power dissipation [4].

With these features, RSFQ can be established with a relatively modest effort as a technology of choice for high performance digital signal processing [3], wideband communication [12–14], precise high frequency instrumentation [15], and numerous scientific applications [16, 17]. In the longer term, RSFQ may also provide the speed and power characteristics required by general purpose petaflop-scale computing (petaflop = 10^{15} floating point operations per second), which is likely to remain beyond the reach of the fastest semiconductor technologies [18, 19].

The primary immediate application of RSFQ logic is digital signal processing. The current state of RSFQ technology favors the design of circuits with a regular topology, limited control circuitry, a small number of distinct cells, and limited interconnections. The analysis of timing in RSFQ circuits presented in this article focuses on but is not limited to this type of architecture, which is well suited for most DSP functions.

1.2. Introduction to RSFQ Timing

Correct timing is essential to fully exploit the high speed capability of individual RSFQ gates, and to translate this advantage into a corresponding speed-up in the performance of medium to large scale RSFQ circuits. Research in this area has only just started and has been only applied to moderate 100-gate circuits to date. Yet even for this medium scale complexity, the design of effective timing schemes in the multi-gigahertz frequency range is a challenging problem.

Timing methodologies for semiconductor VLSI circuits have been well-established and systematized [20–25]. One approach to superconductor circuit design is to rely on the application of such rules and techniques drawn from the semiconductor literature. More prevalent, however, the RSFQ clocking circuitry is developed specifically for RSFQ logic [3, 26, 27]. In this paper these two approaches are intertwined, and the similarities and the differences between semiconductor and superconductor designs are highlighted.

The emerging novel methodologies for designing the clocking circuitry in RSFQ circuits diverge from and challenge two well established rules used in the design of digital semiconductor circuits. First, the idea of *equipotential clocking*, in which the entire clock distribution network is considered to be a surface that must be brought to a specific state (voltage level) every half clock period. The analog of equipotential clocking for RSFQ circuits requires that only one SFQ pulse is present in the clock path from the clock source to the input of any synchronous RSFQ gate. This is inefficient for RSFQ circuits, in which several consecutive clock pulses can coexist within a path of the clock distribution network. Actually, equipotential clocking is inefficient for the design of ultrafast digital circuits in semiconductor technology as well, and can be easily replaced by the less restrictive *pipelined clocking* as suggested in the literature [28, 29].

Second, the ubiquitous *zero-skew clocking* is not a natural choice for RSFQ circuits. Clocking schemes that offer better performance *or* improved tolerance to process induced timing parameter variations have been proposed and analyzed [3, 26, 27]. These schemes utilize intentional clock skew to trade circuit performance with circuit robustness. Techniques that offer a significant improvement in performance over zero-skew clocking without affecting circuit yield have been developed and applied to RSFQ circuits [27, 30]. Similar schemes have been proposed earlier for semiconductor logic [31–33], but these approaches have not as yet

been widely accepted. The primary reasons are conservative design conventions used within industry, complex design procedures [32, 34, 35], relatively small performance improvements (up to 40%), and difficulties in implementing well-controlled delay lines within semiconductor-based clock distribution networks [22, 35]. The success of RSFQ logic may lead to reconsidering the applicability of these techniques to ultrafast semiconductor circuits.

In addition, the emergence of multi-gigahertz RSFQ logic provides a new perspective on several early continuing controversies concerning the design of high speed digital circuits. A central dilemma is the choice between synchronous and asynchronous clocking [29, 36]. *RSFQ logic is well suited for both types of clocking*. Asynchronous event driven schemes such as *dual-rail logic* or *micropipelines* [37] appear to be easier and more natural to implement in RSFQ circuits than in semiconductor-based logic [1, 26, 38, 39]. The same applies to a *bit-level pipeline* synchronous architecture [40]. *Wave pipelining* used to increase the performance of pipelined semiconductor-based circuits [41, 42] can be used with RSFQ logic. It has also been shown that RSFQ is specifically suitable for the *Residue Number System* (*RNS*) representation of numbers [43, 44]. Operations using this representation are extremely efficient and easier to perform in RSFQ than in semiconductor-based logic but conversion difficulties and multiple frequency clocking will likely limit the use of RNS in mainstream applications.

Most medium-scale RSFQ circuits developed to date are fully synchronous circuits with one phase clocking. This trend is likely to continue, unless the problems with scalability of *multidimensional arrays* and large parameter variations require the application of *asynchronous* or *hybrid, globally asynchronous locally synchronous* schemes. In this paper a new two-phase clocking scheme is introduced which offers advantages in robustness, performance, and design simplicity over the ubiquitous single-phase clocking. However, it is far from clear that these advantages are sufficient for any multiple-phase clocking scheme to justify the device/area overhead inherent in these schemes.

2. RSFQ Logic vs. Semiconductor Logic

In this section the similarities and the differences between RSFQ and semiconductor logic elements are discussed. The most important and fundamental difference between the two technologies appear at the device level, as described in Subsection 2.1. The device level differences affect the gate design, and the basic suite of RSFQ gates differs substantially from those familiar in semiconductor logic design, as seen in Subsection 2.2. For example, several RSFQ gates with no direct analog in semiconductor-based logic appear to be the most natural components of RSFQ circuits for DSP applications [3]. All of these differences between RSFQ and semiconductor logic naturally influence the choice of timing schemes, as discussed in this paper; it will nevertheless become clear that the higher the level of abstraction the less significant the differences become.

2.1. *Differences at the Device Level*

Device and circuit level differences between RSFQ logic and semiconductor-based logic are summarized in Table 1. The primary difference is the use of a two-terminal *Josephson junction* as the basic active

Table 1. RSFQ vs. semiconductor voltage-stage technologies.

Characteristics	RSFQ	Semiconductor logic families
Basic active component	*Josephson junction* (2-terminal)	*Transistor* (3-terminal)
Basic passive component	*Inductance*	*Capacitance*
Information transmitted as	Quantized voltage *pulse*	Voltage *level*
Information stored as	*Current* in the *inductance* loop	*Charge* at the *capacitance*
Basic logic gates	*Synchronous*	*Asynchronous* (combinational)
Gate fanout	1	>1
Parasitic component	Parasitic inductance	Parasitic capacitance and resistance
Passive interconnects	Microstrip lines (only for long connections)	Metal RC lines (only for short connections)
Active interconnects	Josephson transmission lines + splitters	Metal RC lines with buffers

component of superconductor-based circuits, as compared to the three-terminal transistor in semiconductor-based circuits. Josephson junctions support the transmission, storage, and processing of information in RSFQ logic [1].

Magnetic field is quantized in a superconductor. It is natural to convey information in superconducting circuits in the form of quantized voltage pulses, each corresponding to the transmission of a basic quantum of the magnetic field called a *single flux quantum* (SFQ). The area of an SFQ pulse, the voltage integrated over time, is equal to

$$\int V(t)dt = \Phi_0 = h/2e = 2.07\ \text{mV} \cdot \text{ps}, \qquad (1)$$

where h is a Planck's constant and e is the electron charge unit. The shape of an SFQ pulse is shown in Fig. 1(a). The pulse width is in the range of several picoseconds and the pulse height is sub-millivolts for a current niobium-trilayer superconductive fabrication technology [2, 4]. Note that this form of information

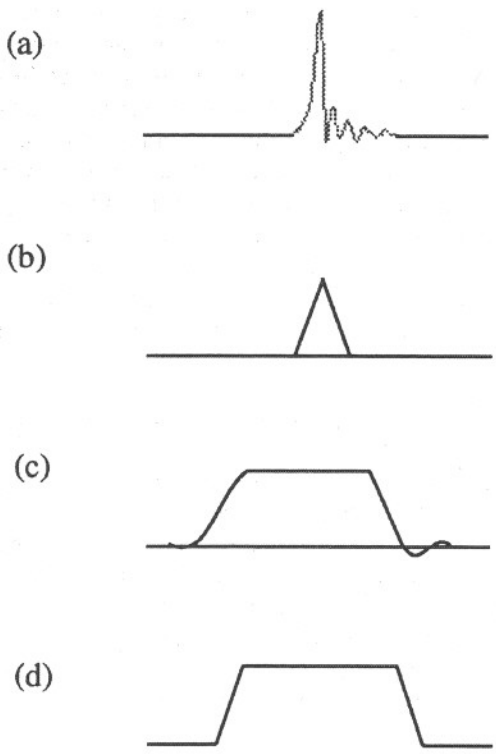


Figure 1. Convention for representation of SFQ pulses in RSFQ and voltage waveforms in voltage-state logic. (a) an SFQ pulse, (b) simplified graphical representation of an SFQ pulse, (c) voltage waveform, (d) simplified graphical representation of a voltage waveform.

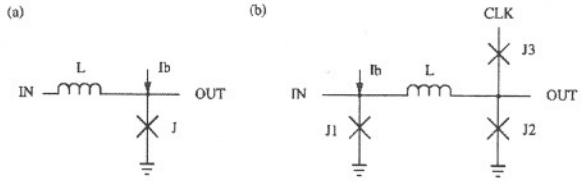


Figure 2. Circuit-level schematic of (a) single stage of a Josephson transmission line (JTL), (b) inductive storage loop including comparator. Notation: Jn—junction, L—inductor, Ib—bias current source.

is measured in fundamental physical constants and is intrinsically digital.

In this paper an SFQ pulse is graphically represented by the symbol shown in Fig. 1(b). Associated with every pulse is a single unique moment in time corresponding to the position of the peak of the pulse voltage. This convention follows the example of the simplified graphical representation of the voltage waveform commonly used in semiconductor digital circuit design, as shown in Figs. 1(c) and (d).

The basic active transmission component of SFQ circuits is called the *Josephson transmission line* (JTL). Single JTL stages (shown in Fig. 2(a)) are connected in series to transmit SFQ pulses without loss over an arbitrary distance. The delay of a single stage is several picoseconds, depending in part on the bias current, and so JTLs provide well-controlled and mutually correlated delays for the design of the clock distribution network. JTLs comprise most of the interconnections in medium to large scale RSFQ circuits, appearing both in the data paths between RSFQ gates and in the clock distribution network.

The basic storage component has the form of an *inductive storage loop* composed of two junctions (J1 and J2) and an inductor (L), as shown in Fig. 2(b). The presence of current in the loop corresponds to the logic state "1". The absence of current corresponds to the logic state "0". The current circulates around the loop without loss, until the state of the loop is evaluated. This evaluation is performed using a *Josephson comparator* which is composed of two serially connected junctions (junctions J3 and J2 in Fig. 2(b)). If the loop contains a logical "1," a pulse at the clock input generates a pulse at the output; if the loop contains a logical "0," no output pulse is generated.

The circuit shown in Fig. 2(b) (a storage loop with a comparator) constitutes the core of the simplest RSFQ clocked gate called a *Destructive Read-Out* cell or *DRO*. The behavior of a DRO for typical input stimuli is shown in Fig. 3(a). Note from Fig. 3(a) and (b) that

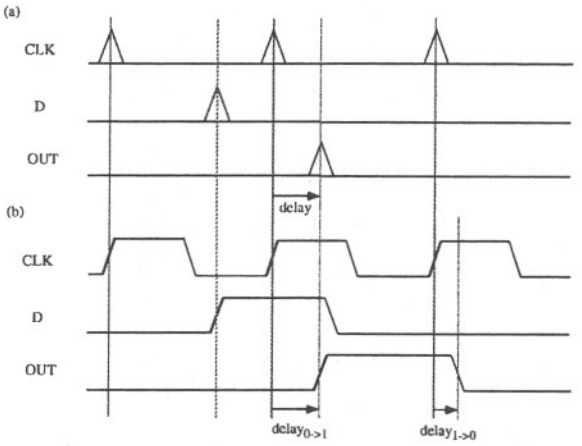


Figure 3. Comparison between the operation of (a) an RSFQ destructive read-out (DRO) cell, and (b) a semiconductor positive edge-triggered *D* flip-flop.

a DRO is the RSFQ analog of the semiconductor edge-triggered *D* flip-flop. The event that changes the state of the *D* flip-flop is the rising edge of a voltage waveform; the corresponding event that changes the state of a DRO is the SFQ pulse.

Basic RSFQ logic gates (e.g., AND, OR, XOR) are composed of a combination of overlapping and interconnected inductive storage loops supplemented with JTL stages and other simple combinations of junctions and inductances [1, 2]. As a result, these gates always contain a clock input used to evaluate the contents of one or several inductive storage loops, and to release the output pulse. Therefore, *most basic RSFQ logic gates are synchronous* as compared to asynchronous combinational semiconductor gates. It is seen that the logic function of an RSFQ gate is inseparable from its storage capability.

The output logic state of an RSFQ gate is clearly determined: an output pulse (or no pulse) following the clock pulse signifies the output logic state "1" (or "0"). However, the RSFQ Basic Convention [1, 45] is required to specify the input logic state of an RSFQ gate: The appearance of a pulse at the data input of the gate in a window determined by two consecutive clock pulses corresponds to a logical "1," the absence of a pulse at the data input in the same window corresponds to a logical "0," as shown in Fig. 4. This convention distinguishes RSFQ from all semiconductor logic families and from other superconductive logic families.

Another important difference among RSFQ and other logic families is the fanout is always equal to one for all RSFQ cells, as compared with fanouts of

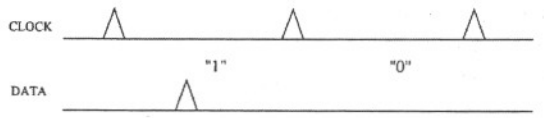


Figure 4. Basic RSFQ convention for representation of logic states.

greater than one for semiconductor logic gates and buffers. Whenever a connection to more than one input is required, a special cell called a *splitter* is used [1]. A splitter repeats at its two outputs the sequence of pulses from its input. As for the JTL, the splitter introduces a significant input- to-output delay that may affect the timing of the circuit. Splitters are inevitable components of RSFQ clock distribution networks.

Another unique feature of this superconductive technology is that an *SFQ pulse can be transferred over large distances with a speed approaching the speed of light*, using passive superconductive *microstrip lines* [1, 2, 46]. This feature was used only recently in the design of an RSFQ clock distribution network [7].

RSFQ is intrinsically a low power technology, but there is an important distinction compared to low power CMOS. In CMOS, the energy is dissipated mainly in the form of a dynamic power during voltage transitions in the circuit nodes. Therefore, the power consumption can be minimized by eliminating redundant activity in the circuit nodes even at the cost of increasing the number of transistors in the circuit. In RSFQ, the energy is consumed primarily in the form of a static power dissipated by current sources providing the bias current to the junctions. Thus, *power consumption is directly proportional to the number of junctions in the circuit.*

2.2. *Function and Complexity of Basic RSFQ Gates*

The logic function of an RSFQ circuit of any complexity can be easily described using a *Mealy state transition diagram* [1], known well from semiconductor digital circuit design. As most RSFQ gates are clocked, these gates contain an internal memory and at least two distinct internal states. A state transition diagram for the DRO cell is shown in Fig. 5(a), together with a symbol of the gate. The nodes of the Mealy diagram correspond to the two distinct logic states of the DRO storage loop. The arrows show transitions that appear as a result of input pulses (including clock pulses). Output data pulses are associated with transitions between states, and for synchronous cells appear as a result of pulses that arrive at the clock input.

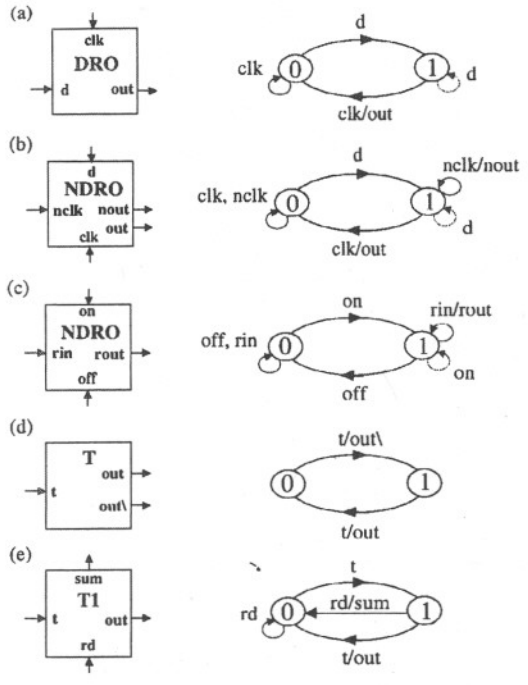


Figure 5. Symbols and Mealy state transition diagrams for basic RSFQ gates: (a) DRO, (b), (c) NDRO, (d) T flip-flop, (e) T1 flip-flop.

The function of most elementary RSFQ gates may be described by analogy to the function of their semiconductor counterparts, as shown in Table 2. Note, however, that this analogy must be correctly understood. The behavior of the two circuits is similar but not identical: the rising edge of a voltage waveform in a semiconductor circuit corresponds to an SFQ pulse in the RSFQ counterpart, as shown in Fig. 3.

Table 2. Semiconductor counterparts of RSFQ gates.

RSFQ gate	Semiconductor counterpart
Clocked	
DRO	*D* flip-flop
NOT	NOT + *D* flip-flop
AND	AND + *D* flip-flop
OR	OR + *D* flip-flop
XOR	XOR + *D* flip-flop
Non-clocked without memory	
Splitter	Buffer with fanout two
Confluence buffer	Event OR
Non-clocked with memory	
NDRO	Transmission gate
Coincidence junction	Muller C-element

Table 3. Complexity of RSFQ gates and CMOS counterparts.

RSFQ gate	# of JJs	CMOS gate	# of transistors
DRO	4	*D* flip-flop	12
NOT	4	NOT + *D* flip-flop	2 + 12
AND	14	AND + *D* flip-flop	6 + 12
OR	8	OR + *D* flip-flop	6 + 12
XOR	7	XOR + *D* flip-flop	6 + 12
T-flip-flop	5	—	—
T1-flip-flop	8	—	—
Confluence buffer	5	OR	6
Splitter	3	Buffer with fanout two	4
NDRO	9	Transmission gate	2

Review articles on RSFQ [1, 2, 47] describe state transition diagrams, circuit level schematics, and device parameters for the majority of basic RSFQ cells. The existing suite of basic RSFQ gates does not include such elementary semiconductor gates as NAND, NOR, and XNOR [48]. This difference occurs since inversion is more difficult to obtain in RSFQ than in voltage stage logic [2]. Also, the relative complexity of various cells differs substantially between the two technologies, as shown in Table 3 [1, 48]. These differences require new design methodologies, including a different set of elementary gates. These differences also make the automated logic synthesis of large RSFQ circuits particularly challenging.

Apart from clocked gates, a basic set of RSFQ cells also includes several non-clocked (asynchronous) cells that are used to build larger synchronous or asynchronous RSFQ circuits. Non-clocked cells without memory include the splitter cell (described above) and the *confluence buffer*. The confluence buffer operates as an asynchronous OR: it passes all pulses from either of its inputs to the output with appropriate delay [1]. The standard implementation of this gate has a significant drawback; it does not allow two input pulses to appear too close in time to each other. If the distance between pulses at the two inputs of the confluence buffer is smaller than the *minimum separation time*, only one pulse will appear at the output.

Most frequently used non-clocked RSFQ gates with internal memory are the NDRO cell [1, 2], T flip-flop [1], and T1 flip-flop [49]. Symbols and state transition diagrams describing each of these cells are shown in Figs. 5(b)–(d) and (e).

The *NDRO* cell can be treated as a simple extension of the DRO cell [1] (Fig. 5(b)). Apart from operating

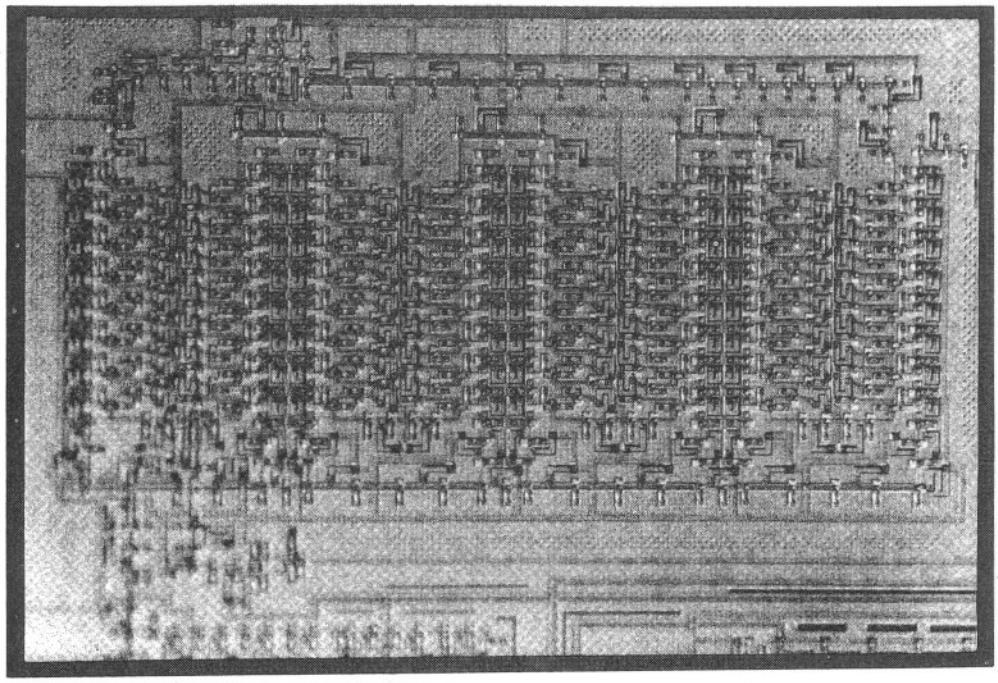


Figure 6. Photograph of the RSFQ circular shift register.

similar to a DRO, it has an additional function associated with an extra non-destructive clock input (nclk) and non-destructive clock output (nout). The non-destructive clock reads the contents of the storage loop to a non-destructive output without changing the internal state of the cell.

Another interpretation of the function of the NDRO cell is given in Fig. 5(c). In this case, the NDRO does not have a previous destructive-read output, and the remaining inputs and outputs have been renamed to better describe this new function. The cell behaves like a CMOS transmission gate [48]. Pulses at inputs ON and OFF permit the gate to transmit, and not transmit, respectively. In the transmitting mode, every pulse from the input RIN propagates with a delay to the output ROUT. In the non-transmitting mode, no pulse appears at the output ROUT regardless of the pulses at the RIN input.

A *T flip-flop* is a modulo two counter that reverses its logical state each time a pulse appears at the T input (Fig. 5(d)). A pulse is generated at its primary output every two input pulses. A T1 flip-flop is an extension of the T flip-flop that permits destructive read-out of an internal state of a T flip-flop to a separate output SUM (Fig. 5(e)).

Other more complex RSFQ cells with sophisticated logic functions have been reported in the literature. These include: a demultiplexer [47, 49, 50], B flip-flop [51], full-adder [1, 49, 52], adder-accumulator [8], carry-save adder [53, 54], and a majority AND gate [52]. Most of these cells cannot be decomposed into simpler RSFQ cells. Special cells with complementary inputs and outputs have been designed to be used with asynchronous dual rail logic [55–58] as described in Section 5. The photograph of a medium size RSFQ circuit—RSFQ circular shift register [59] is shown in Fig. 6.

In most cases, cells specifically designed for RSFQ logic are superior to functionally equivalent cells generated from semiconductor circuit design principles. As an example, in Figs. 7(a) and (b) two equivalent implementations of a half-adder in RSFQ logic are shown. From Table 3, it is seen that the RSFQ-specific implementation results in a circuit with fewer junctions, 20 versus 30 in this case, and thus also a smaller area. A more significant difference between the two implementations, however, is evident when one extends the function of the half-adder to that of a *full adder* or *adder accumulator*. The traditional half-adder in Fig. 7(a) cannot be easily changed; any extension would involve adding several new gates and multiplying the complexity of the circuit. For the RSFQ-specific implementation either modification is small and straightforward (although mutually exclusive, as

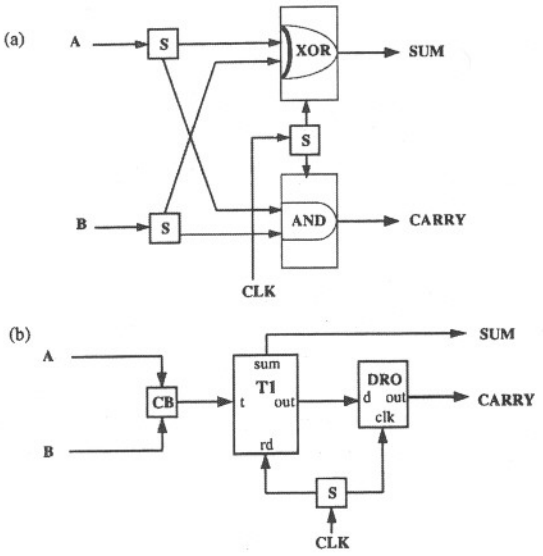


Figure 7. Two implementations of a half-adder in RSFQ logic; (a) based on elementary logic gates; (b) based on gates specific to RSFQ.

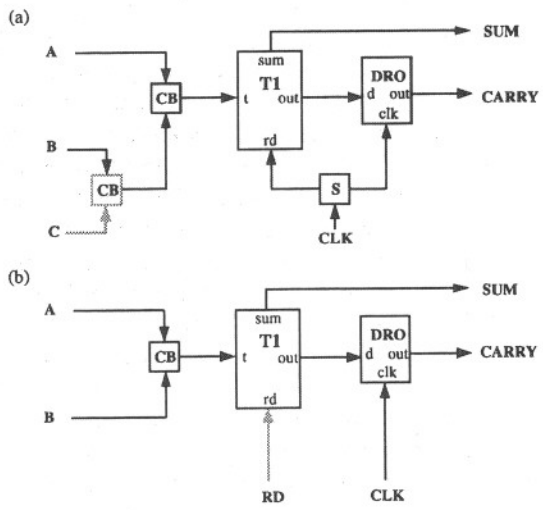


Figure 8. Modification of the half adder to (a) full-adder, (b) adder-accumulator.

a "full adder-accumulator" is not a valid gate). The full adder is obtained by adding a single confluence buffer at the data input as illustrated in Fig. 8(a); the function of an adder-accumulator is created by deleting the splitter at the clock input, and separating the clock (CLK) and read (RD) inputs, as shown in Fig. 8(b).

The best approach for choosing which basic gates should comprise a circuit may depend upon the function of the circuit, for instance digital signal processing vs. general purpose computing and operational unit vs. control unit. For example, the control units in an RSFQ microprocessor can be based on a set of asynchronous *event-driven* (*data-driven*) gates with events represented in the form of SFQ pulses [60–62]. Similarly, the synchronization scheme also influences the most effective suite of basic gates. For instance, asynchronous gates with complementary inputs and outputs are well suited for an asynchronous data-driven synchronization scheme, while basic synchronous RSFQ logic gates (AND, OR, XOR, NOT) are well suited for synchronous bit-level pipelining.

3. Single-Phase Synchronous Clocking

Single-phase synchronous clocking is the form of clocking most frequently used in semiconductor circuit design. Its primary advantages include high performance, design simplicity, small device and area overhead, and good testability. Several authors regarded this kind of clocking as inadequate for ultrafast RSFQ circuits [26, 63]. The main argument used against synchronous clocking is the deteriorating effects of *clock skew* and *phase delay* on circuit robustness and performance. Despite these theoretical limitations, single-phase synchronous clocking has been successfully used in almost all medium to large scale RSFQ circuits developed to date [3, 5–9].

In this section, it is shown that most of the limitations of single-phase synchronous clocking can be easily overcome by applying an appropriate design procedure. In Subsection 3.1, it is shown that using *pipelined* (*flow*) clocking instead of equipotential clocking eliminates the deteriorating effect of *phase delay* (the propagation delay from the clock source to the most remote cell in the clock distribution network) on the circuit performance. In Subsection 3.2, the limitations imposed by the external and internal clock sources are analyzed. In Subsection 3.3, techniques to minimize the effect of *clock skew* on the circuit performance without decreasing the circuit yield and reliability are presented. This discussion is continued in Subsection 3.4 and Subsection 3.5 by analyzing several synchronous clocking schemes with different topologies for the clock distribution network and different values of the interconnect delays. In Subsection 3.6, these clocking schemes are applied to particular circuit—a linear unidirectional pipelined array comprised of N heterogeneous RSFQ cells. A graphical model of the circuit behavior is provided, and the performance of all clocking schemes is compared in terms of circuit throughput and latency. The analysis presented in this section is

extended in Section 4 by taking into account the effects of fabrication process variations.

3.1. Equipotential vs. Pipelined Clocking

Two basic modes of clocking apply to any general *semiconductor* clock distribution network:

Equipotential clocking [20, 25] assumes that a voltage state (voltage level) at the primary clock input does not change until the previous state has propagated through the longest path in the clock distribution network. This limitation has historically been negligible, as the *phase delay*, i.e., the worst case propagation delay in the clock path, was typically much smaller than the limitation imposed on the clock period by the most critical data path between two registers in the circuit. For high speed large scale semiconductor circuits, however, this is no longer true; the limitation imposed by the propagation delay of the clock distribution network becomes a dominant factor which limits the maximum clock frequency in this type of clocking environment [29].

As described in the literature [28, 29], the requirement of equipotential clocking can be substantially relaxed. In clock distribution networks composed of metal interconnections separated by buffers, it is sufficient that the voltage state in a given node in the network does not change until the previous state has propagated past the nearest buffer. A method of clocking that complies with this much less restrictive rule is called *pipelined clocking* [28]. In *pipelined clocking*, several consecutive clock transitions corresponding to several clock cycles may travel simultaneously along the longest path in the clock distribution network.

In RSFQ logic, even for medium size circuits, the propagation delay through the clock distribution network is often several times larger than the worst case data path delay. Two factors contribute to this. First, the clock distribution network is typically composed of JTLs and splitters, each with a delay comparable to the delay of a single RSFQ gate. Multiple JTL stages must be used to cover the physical distance between the clock inputs of neighboring cells. Second, the data path between two clocked RSFQ gates does not contain any combinational logic. Therefore, equipotential clocking is not considered to be a viable solution for medium to large scale RSFQ circuits. Instead, pipelined clocking, referred in RSFQ literature as *flow clocking* [3], is used in all medium to large scale RSFQ circuits developed to date [3, 5–9]. In flow (pipelined) clocking, several consecutive clock pulses travel simultaneously through

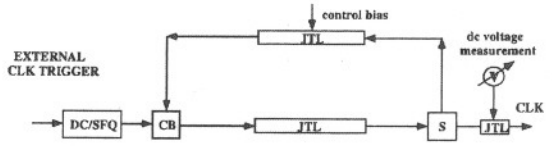


Figure 9. RSFQ clock ring used as an internal clock source. Notation: S—splitter, CB—confluence buffer.

the clock distribution network. In the clock distribution network composed of JTLs and splitters, the only limit on the distance between clock pulses originates from the width of the clock pulse [2] and the effects of the interactions between consecutive pulses [64]. Both limitations are negligible compared to the limitations imposed by the critical data path in the circuit.

3.2. Clock Sources

Additional practical limitations on the maximum clock frequency of RSFQ circuits derive from the characteristics of the available clock sources. When an external clock generator is used, the high-frequency sinusoidal signal must be converted to a string of SFQ pulses using a *DC/SFQ converter* [1, 2]. The maximum frequency of the clock is constrained by the maximum input frequency of the converter. An alternative solution is the use of an on-chip clock generator. An internal clock source can be composed of a JTL ring with a confluence buffer used to introduce the initial pulse to the ring, and a splitter used to read the data from the ring [9, 47, 65], as shown in Fig. 9. The minimum clock period of the ring is limited by the sum of the delays of the splitter and the confluence buffer to less than 100 GHz with current fabrication technology. The other form of on-chip high frequency clock, an overbiased Josephson junction [1], can generate much higher frequencies but it has limitations arising from its relatively large jitter.

3.3. Synchronization of a Pair of Clocked Cells

A variety of clocking schemes (single-, two-, and multiple-phase) and associated storage elements are used in semiconductor logic design [22]. *Single-phase clocking* typically requires the use of either *edge-triggered D flip-flops* or *D latches*. In Section 2, Fig. 3, it was shown that the RSFQ basic storage element, DRO, is the analog of the positive edge-triggered *D* flip-flop. The authors are unaware of any analog of a semiconductor *D* latch in RSFQ logic.

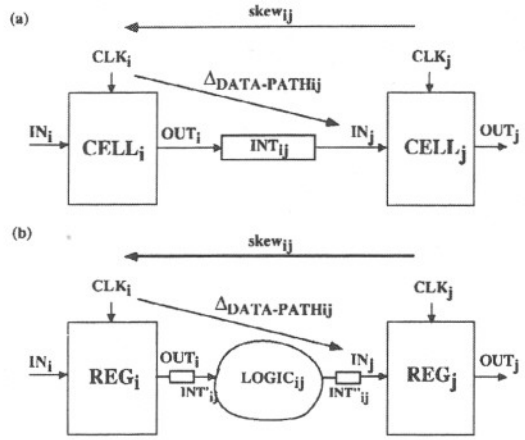


Figure 10. Data path between two sequentially adjacent cells in (a) RSFQ logic; (b) semiconductor logic. Notation: INT_{ij}—interconnection between cells i and j, REG—register composed of D-flip-flops, $LOGIC_{ij}$—combinational logic, $skew_{ij}$—clock skew between cells i and j.

Two storage (clocked) cells that exchange data between each other are called *sequentially adjacent. Conditions for the correct exchange of data between a pair of sequentially adjacent RSFQ cells are identical to the conditions for communicating between two semiconductor positive-edge triggered D flip-flops.* These conditions are demonstrated below.

Schematics of generalized *synchronous data paths* for RSFQ and for semiconductor circuits using D flip-flops are shown in Figs. 10(a) and (b). These schematics are almost identical, apart from two important differences. First, in semiconductor circuits, the actual logic function of the circuit is performed by a combinational path (labeled $LOGIC_{ij}$ in Fig. 10(b)) between the two D flip-flop storage components (labeled REG_i and REG_j in Fig. 10(b)). In RSFQ circuits, the logic function is performed by the cells at the beginning and at the end of the data path (labeled $CELL_i$ and $CELL_j$ in Fig. 10(a)). The logic function of an RSFQ gate is inseparable from the storage capability. Interconnections between cells INT_{ij} are typically composed of a few JTL stages and do not perform any logic function. Second, storage cells at the beginning and at the end of the data paths in semiconductor circuits are typically identical for all data paths within the entire system, and are characterized using a single set of timing parameters (hold time, setup time, and the clock-to-output delay of a D flip-flop). In RSFQ circuits, cells at the beginning and at the end of the data paths are not identical, and change from one data path to the next. The hold and setup times of various RSFQ cells differ substantially.

For both technologies, an important parameter describing the data path is the *clock skew* [20, 21]. Clock skew (denoted $skew_{ij}$) is defined as the difference between the arrival time of the clock signal (SFQ pulse in RSFQ, rising edge of the clock waveform in voltage-state logic) at the clock inputs of the cells at the beginning and at the end of data path (t_{CLK_i} and t_{CLK_j}, respectively). The clock skew between cells i and j is

$$skew_{ij} = t_{CLKi} - t_{CLKj}. \tag{2}$$

Similarly, the *data path delay* (denoted $\Delta_{DATA\text{-}PATH_{ij}}$) is defined as the interval between the moment when the clock arrives at the *clock* input of the first cell (t_{CLK_i}), and the moment when the data appears at the *data* input of the second cell (t_{IN_j}):

$$\Delta_{DATA\text{-}PATH_{ij}} = t_{IN_j} - t_{CLK_i}. \tag{3}$$

Waveforms corresponding to the correct exchange of data between two sequentially adjacent cells in the presence of clock skew are shown in Figs. 11(a) and (b) for voltage state logic and for RSFQ, respectively. From these waveforms, two inequalities that fully describe the timing constraints of the data path between two

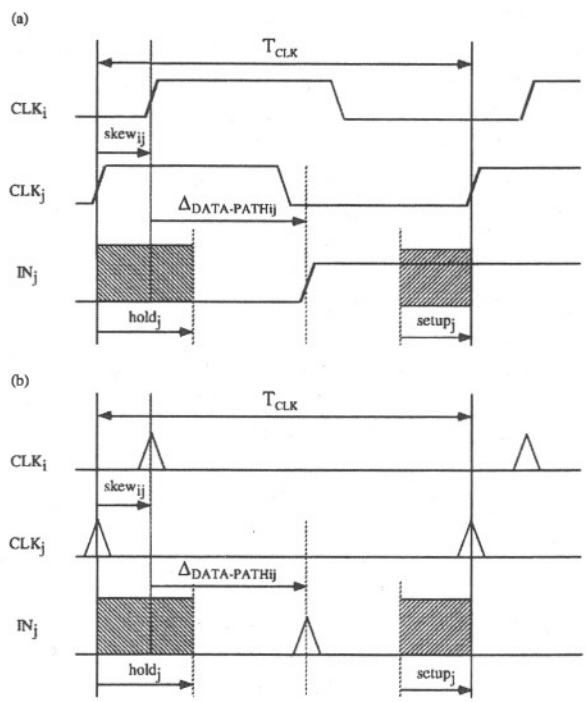


Figure 11. Timing diagram describing the exchange of data between two sequentially adjacent storage cells in (a) semiconductor voltage-state logic, (b) RSFQ logic.

adjacent cells can be derived:

$$skew_{ij} + \Delta_{\text{DATA-PATH}_{ij}} \geq hold_j, \quad (4)$$

$$T_{\text{CLK}} \geq skew_{ij} + \Delta_{\text{DATA-PATH}_{ij}} + setup_j. \quad (5)$$

These inequalities are identical for RSFQ and voltage state logic. The formulas for the data path delay differ between RSFQ and semiconductor technologies. For RSFQ,

$$\Delta_{\text{DATA-PATH}_{ij}} = \Delta_{\text{CELL}_i} + \Delta_{\text{INT}_{ij}}, \quad (6)$$

for a voltage state logic

$$\Delta_{\text{DATA-PATH}_{ij}} = \Delta_{\text{REG}} + \Delta_{\text{LOGIC}_{ij}} + \Delta_{\text{INT}_{ij}}, \quad (7)$$

where Δ_X denotes the delay introduced by the component X.

Using (4) and (5), the dependence between the clock skew and the minimum clock period in the circuit can be determined. Clock skew can be both positive and negative [25]. Positive clock skew increases the minimum clock period [see (5)], but at the same time prevents the possibility of race errors (the propagation of the data through several data paths within one clock period) that occurs when (4) is not satisfied. Negative clock skew decreases the minimum clock period, but makes a violation of the hold time constraint, and thus race errors, more likely.

The operating region of the circuit composed of two sequentially adjacent cells as a function of the clock period and the clock skew between the cells is shown in Fig. 12. The following conclusions can be drawn:

a) Changing the nominal value of the clock skew changes the minimum clock period. The minimum clock period is linearly dependent on the clock skew. *There exist values of clock skew for which the circuit does not work for any* (even an extremely small) *clock frequency.*
b) The minimum clock period is equal to

$$T_{\text{MIN}} = hold_j + setup_j, \quad (8)$$

and is obtained for a clock skew equal to

$$skew_0 = -\Delta_{\text{DATA-PATH}_{ij}} + hold_j. \quad (9)$$

Note that although the hold and setup time may be individually negative, the sum of the hold and setup time is always positive. In contrast, the optimal value of the clock skew, $skew_0$, although typically negative, can be positive for some configurations of RSFQ cells.
c) It can be seen that *zero clock skew is in no respect advantageous compared to other values of clock skew.* It is only a point on a continuum of allowed values of clock skew.

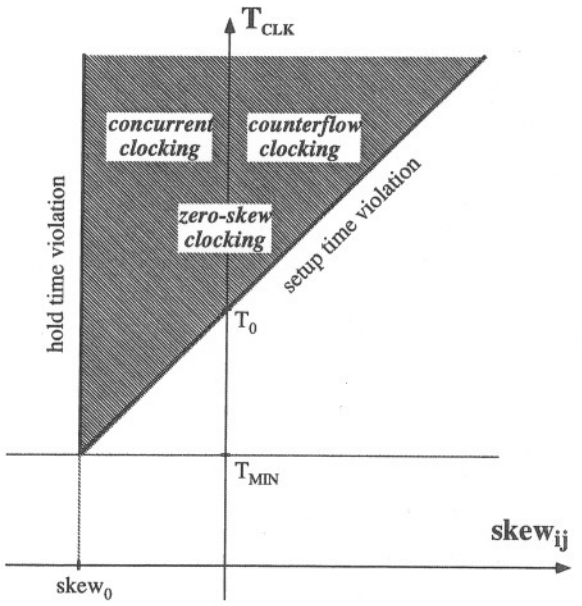


Figure 12. Operating region of a circuit composed of two sequentially adjacent cells as a function of the clock period and the clock skew.

In circuits with a closed data loop [59, 66], the sum of the local clock skews around the loop must be equal to zero. This characteristic however does not imply that all local clock skews must be equal to zero. Local skews may be different in order to minimize the clock period imposed by the most critical data path in the loop (this design procedure is referred to in the literature as "cycle stealing" or as exploiting "useful clock skew" [22, 25]).

Similarly, in many cases the module is a part of a larger (e.g., multi-chip) circuit. If communication between modules is synchronous, the requirement to maintain zero clock skew among all of the inputs and outputs of the module may be imposed [35]. This however does not apply to the current state of RSFQ technology, where the complexity of circuits within a single chip is limited, and the projected inter-chip communication is asynchronous.

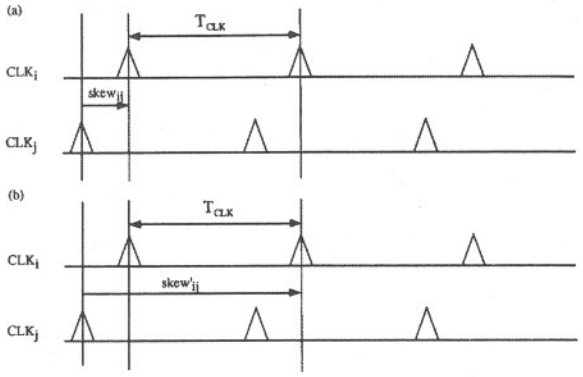


Figure 13. $skew_{ij}$ in (a) and $skew'_{ij}$ in (b) are indistinguishable from the point of view of the circuit operation at clock period T_{CLK}.

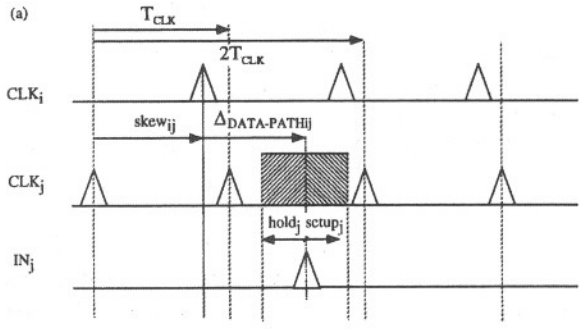


Figure 14. Timing illustration for a circuit operating with a non-conventional value of the clock skew: Counterflow clocking with $k = 1$.

These results can be generalized by the simple observation that for a fixed clock period, values of clock skew that differ by an integer multiple of the clock period are indistinguishable from the point of view of maintaining correct circuit operation, as illustrated in Fig. 13. With this observation, conditions (4) and (5) can be rewritten as follows:

$$skew_{ij} - kT_{CLK} + \Delta_{DATA\text{-}PATH_{ij}} \geq hold_j, \quad (10)$$

$$T_{CLK} \geq skew_{ij} - kT_{CLK} + \Delta_{DATA\text{-}PATH_{ij}} + setup_j. \quad (11)$$

$k = 0$ corresponds to the circuit operating in a standard manner, as shown in Fig. 11(b). The operation of the circuit for the case of $k = 1$ is shown in Fig. 14. The clocking scheme corresponding to $k = -1$ is described in Section 3.4.2. In Fig. 15, the generalized operating region of the circuit composed of two adjacent RSFQ cells as a function of the clock skew and the clock period is shown. Historically, only the operating

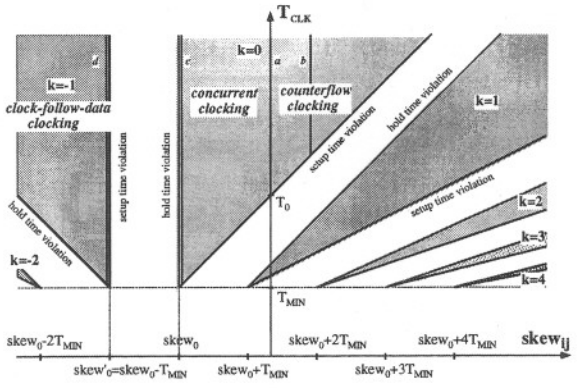


Figure 15. Complete operating space of the data path between two sequentially adjacent cells as a function of the clock frequency and the clock skew. Lines a, b, c, d, correspond to the range of allowed clock frequencies for the circuit with the clock skew fixed to the optimum value for a given clocking scheme (without taking parameter variations into account). a—zero-skew clocking, b—counterflow clocking, c—concurrent clocking, d—clock-follow-data clocking.

region corresponding to $k = 0$ has been used, almost exclusively.

3.4. *Basic Single-Phase Clocking Schemes*

3.4.1. Standard Clocking Modes. The most popular clocking scheme used in *semiconductor* circuit design is single-phase zero-skew equipotential clocking [20, 22]. A clock distribution network used to implement this clocking scheme for a two-dimensional systolic array has the form of an H-tree network consisting of metal lines separated by large-fanout buffers [67, 68], as shown in Fig. 16(a). Buffers within the clock distribution network decrease the time of the clock propagation through the longest path in the network and substantially decrease the requirements on the fanout of the clock source [22]. Nominally, the symmetry of an H-tree clock distribution network assures the simultaneous arrival of the clock signal to the inputs of all the cells in the array. However, in a real circuit there will be timing parameter variations in both the passive and active components of the network, and so the actual clock skew between any two sequentially adjacent cells is randomly distributed around zero [69]. The worst case value of this clock skew depends on the size of the array and on the distribution of the local parameters. This problem is addressed in detail in Section 4.

Zero-skew clocking is relatively easy to implement in RSFQ circuits. In Fig. 16(b), an RSFQ H-tree network composed of JTLs and splitters suited for a square structured systolic array is shown. With some

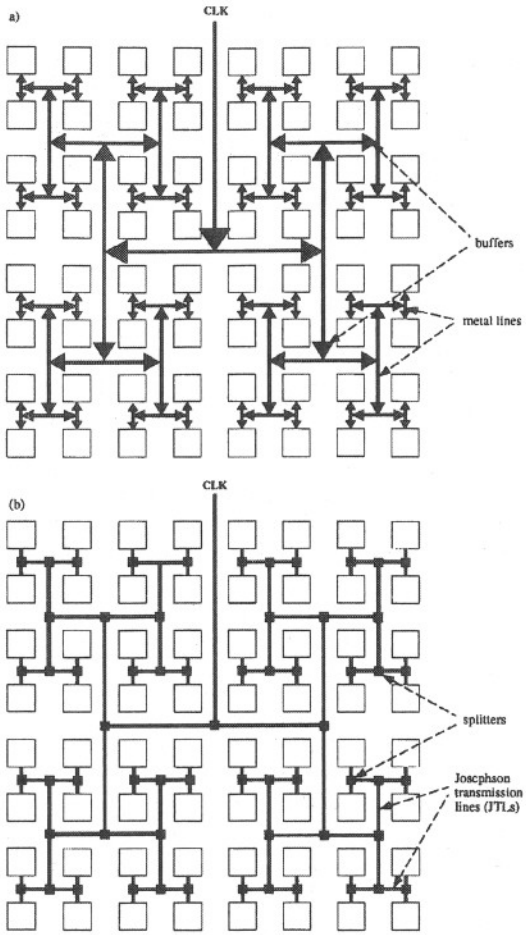


Figure 16. (a) H-tree zero-clock-skew clock distribution network in semiconductor logic. (b) H-tree zero-clock-skew clock distribution network in RSFQ logic.

overhead, similar networks can be build for less symmetric circuit structures. However, as shown in the previous section, zero clock skew is in no respect advantageous to other values of the clock skew. Usually, the *optimum* clock skew for a pair of sequentially adjacent cells, $skew_0$ [defined by (9)], is substantially less than zero. Less commonly, for some configurations of RSFQ cells, $skew_0$ may be positive. In this case a circuit with zero clock skew will not operate correctly for any clock frequency. Note that this situation cannot occur in semiconductor circuits for which the hold time of the edge-triggered D flip-flop is typically equal to zero (and is certainly less than the delay of the D flip-flop) [48], and thus $skew_0$ is always negative.

A general linear pipelined array is shown in Fig. 17(a). When zero-skew clocking is applied the clock distribution network has the form of a binary

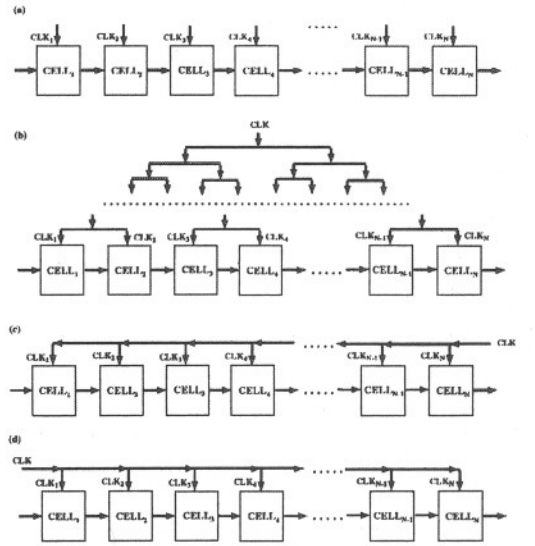


Figure 17. Clocking in the one-dimensional array. (a) General structure of the array; (b) binary-tree zero-skew clocking; (c) straight-line counterflow clocking; (d) straight-line concurrent clocking.

tree as shown in Fig. 17(b). This has two disadvantages. First, the binary tree is composed of a large number of splitters and JTLs. Second, the skew between the clock source and the clock signals arriving at all of the cells in the array is large, and this may affect the synchronization between the array and the other circuits connected to its inputs and outputs.

An alternative to clocking a one-dimensional systolic array with a binary tree structure is *straight-line clocking* [70], in which the clock path is distributed in parallel to the data path of the array. Two types of straight-line clocking can be distinguished. In *counterflow clocking* [1], the clock flows in the direction *opposite* to the data, as shown in Fig. 17(c). In *concurrent clocking* (also referred to as con-flow [3] or concurrent-flow clocking [1, 27]), the clock and the data flow in the *same* direction, as shown in Fig. 17(d).

For straight line clocking the magnitude of the clock skew is equal to the propagation delay through the clock path between two adjacent cells. In RSFQ circuits, this delay is equal to the delay of a single splitter plus the delay of an interconnecting JTL. The *sign* of the clock skew depends upon the relative direction of the clock and data signals, which is opposite for counterflow vs. concurrent clocking.

For counterflow clocking, clock skew is positive. As shown by Eq. (4) and Fig. 12, a violation of the hold time is less likely than for zero-skew clocking. This

characteristic means that counterflow clocking is a robust design strategy—the circuit timing should always be correct at a frequency low enough to satisfy the setup time constraint, even if there are large timing parameter variations. The disadvantage of counterflow clocking is that the minimum clock period of the circuit is larger than for zero-skew clocking by the magnitude of the delay in the clock path.

For counterflow clocked circuits, as shown in (5), the clock skew and hence the propagation delay in the clock path should generally be minimized. This is advantageous because the hold time constraint (4) is typically satisfied even for zero clock skew. Thus counterflow circuits are designed using the minimum number of JTL stages necessary to cover the physical distance between the clock inputs of adjacent cells. A common strategy is to scale the physical dimensions of the JTL (without changing the values of the device parameters) to permit covering the maximum physical distance with the minimum number of JTL stages, and thus with the minimum delay. The correct operating points of the circuit for a fixed clock skew and for clock periods greater or equal to the minimum clock period are indicated by the line *b* in the diagram in Fig. 15.

For concurrent clocking, clock skew is negative. The data released by the clock from the first cell of the data path travels simultaneously with the clock signal in the direction of the second cell. The clock arrives at the second cell earlier than the data. The clock releases the result of the cell operation computed during the last clock cycle, preparing the cell for the arrival of the new data.

Concurrent clocking guarantees greater maximum clock frequency than counterflow or zero-skew clocking. The clock skew in concurrent clocking may be set to the optimum nominal value corresponding to the minimum clock period by choosing an appropriate delay (number of stages) of the interconnect JTL line. The minimum clock period T_{MIN} is given by (8). This limitation is imposed only by the internal speed of the gates, and not by the clock distribution network as in previous schemes. The optimum clock skew is given by (9). Operating points for the optimal clock skew and for clock periods greater or equal to the minimum clock period form the line *c* in the diagram in Fig. 15. The data pulse arrives at the input of the second cell in the worst case data path at the *beginning* of the clock period at the boundary of the *hold* time violation as shown in Fig. 18(a).

In the presence of timing parameter variations affecting both the clock skew and the position of the hold time

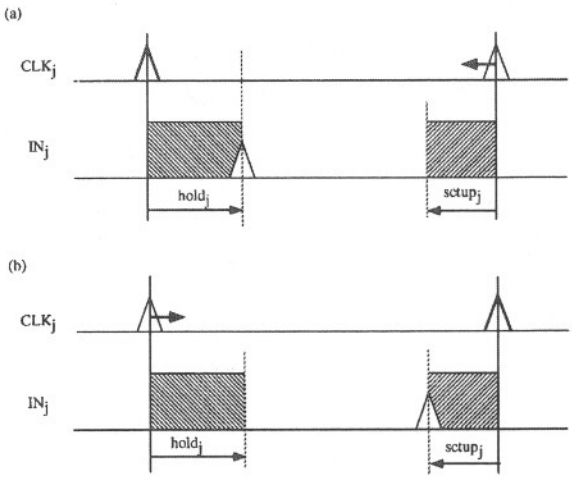


Figure 18. The position of the data pulse within the clock period for the optimal value of the clock skew in (a) concurrent clocking; (b) clock-follow-data clocking.

boundary, the circuit is vulnerable to the hold time violation, which may appear independently of the clock frequency. This is unacceptable, and thus the absolute value of the nominal clock skew must be decreased, as described in detail in Section 4. This leads to a smaller than optimum performance gain and requires a relatively complex design procedure.

Both counterflow and concurrent flow clocking can be generalized to the case of a two dimensional array. The corresponding clock distribution networks have a *corner-based* (*comb*) topology (as in Fig. 27, below).

3.4.2. Clock-Follow-Data Clocking. If the magnitude of the clock skew (the delay in the clock path) is increased in a clock distribution network with the straight-line concurrent clocking topology (Fig. 17(d)), a distinct clocking mode results. In this mode the data signal released by the clock from the first cell of the data path arrives at the second cell earlier than the clock. We call this scheme *clock-follow-data* clocking. [As the topology of the clock distribution network and the sign of the clock skew is the same as in concurrent clocking, clock-follow-data clocking has been previously referred in the literature as con-flow with data traveling faster [3] or simply concurrent-flow clocking [1]. We introduce a separate name for this mode to clearly distinguish it from the typical concurrent clocking scheme].

The operating region of the circuit in the clock-follow-data is described by (10) and (11) with $k = -1$ and is shown in Fig. 15. The typical operation of the circuit is shown in Fig. 19.

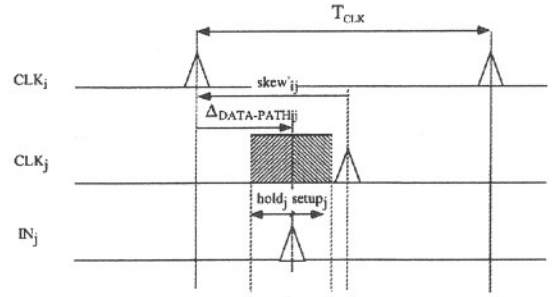


Figure 19. Timing diagram describing the exchange of data between two sequentially adjacent cells in clock-follow-data clocking.

In clock-follow-data clocking a *single* clock pulse carries the data through the whole array of N clocked cells in a time which is independent of the clock period. In concurrent clocking, $N-1$ clock periods are necessary to carry the data through an array comprised of N cells.

The clock skew in clock-follow-data clocking may be set to the optimum value, corresponding to the minimum clock period, by choosing an appropriate number of interconnect JTL stages. The minimum clock period T_{MIN} is the same as for the concurrent clocking mode, and is given by (8). The optimum clock skew in clock-follow-data clocking, $skew'_0$, differs from the optimum clock skew for concurrent clocking, $skew_0$, by the value of the minimum clock period (see Fig. 15), i.e.,

$$skew'_0 = skew_0 - T_{MIN} = -\Delta_{DATA\text{-}PATH_{ij}} - setup_j. \tag{12}$$

In clock-follow-data clocking the data pulse at the input of the second cell in the worst case data path lies at the *end* of the clock period, at the boundary of the *setup* time violation, as shown in Fig. 18(b). This relation means that in the presence of timing parameter variations affecting both clock skew and the position of the setup time boundary, the circuit may exhibit a setup time violation independent of the clock frequency. Therefore, the nominal magnitude of the clock skew must be *increased* above the theoretical optimum (see Fig. 15).

3.5. *Minimum Clock Period in Various Clocking Schemes*

The minimum clock period of the synchronous circuit T^{MIN}_{CLK} is equal to the maximum of limitations $T^{MIN}_{CLK_{ij}}$ imposed by the data paths between any pair of sequentially adjacent cells, $CELL_i$, $CELL_j$, in the circuit, i.e.,

$$T^{MIN}_{CLK} = \max_{ij} \{T^{MIN}_{CLK_{ij}}\}. \tag{13}$$

The minimum clock period imposed by a pair of sequentially adjacent cells is equal to

$$T^{MIN}_{CLK_{ij}} = skew_{ij} + \Delta_{DATA\text{-}PATH_{ij}} + setup_j. \tag{14}$$

Let us consider the minimum clock period for different clocking schemes:

For zero-skew clocking, the minimum clock period is

$$T^{MIN}_{zero\text{-}skew} = \max_{ij} \{\Delta_{DATA\text{-}PATH_{ij}} + setup_j\}. \tag{15}$$

For counterflow clocking, the minimum clock period is equal to

$$T^{MIN}_{counterflow} = \max_{ij} \{\Delta_{CLK\text{-}PATH_{ij}} + \Delta_{DATA\text{-}PATH_{ij}} + setup_j\}, \tag{16}$$

where $\Delta_{CLK\text{-}PATH_{ij}}$ is the delay of the clock path between cells i and j. This delay is typically the delay of one splitter and the minimum number of JTL stages necessary to cover the physical distance between the clock inputs of both cells.

For concurrent clocking and clock-follow-data clocking, with the optimal clock skew between cells given by (9) and (12), respectively, the minimum clock period is

$$T^{MIN}_{concurrent} = T^{MIN}_{clock\text{-}follow\text{-}data} = \max_j \{hold_j + setup_j\}. \tag{17}$$

From (15)–(17),

$$T^{MIN}_{concurrent} = T^{MIN}_{clock\text{-}follow\text{-}data} < T^{MIN}_{zero\text{-}skew} < T^{MIN}_{counterflow}. \tag{18}$$

3.6. *Performance of the Linear Pipelined Array with Synchronous Clocking*

Consider a general linear synchronous array comprised of N heterogeneous cells with distinct timing parameters, as shown in Fig. 17(a). The array processes data in a pipelined fashion. The data is fed to the input of the first cell in the array, and the corresponding result appears at the output of the Nth cell after the appropriate number of clock cycles. The performance of

the pipeline is described using two parameters. The *throughput* is defined as the output rate of the circuit, i.e., the inverse of the time between two consecutive outputs. In a synchronous array, throughput is equal to clock frequency. *Latency* is defined as the total time needed to process the data from the input to the output of the circuit. In an N-cell synchronous array, the latency is defined as an interval between the moment when the clock reads the data into the first cell, and the moment when the clock releases the corresponding result from the last cell of the array.

The behavior and performance of the linear array are analyzed for different clocking schemes using space vs. time diagrams shown in Figs. 21 to 24. In these diagrams, the data flows in two directions, in space—along the vertical axis, and in time—along the horizontal axis. The flow of the data in space corresponds to the data moving from one stage of the pipeline to the next stage as a result of the clock pulse at the cell separating the two stages. Each clock pulse releases the next data. The flow of the data through the data path between two clocked cells i and j is represented by a horizontal bar (rectangle), according to the convention depicted in Fig. 20. In this convention, the time necessary for processing the data within a single stage between cells i and j (interval AD in Fig. 20) is equal to the sum of the propagation delay through the data path [as defined by (6)] and the setup time of the cell j. The shaded part of the rectangle (interval CD in Fig. 20) represents the time interval around the position of the data pulse at the input of the cell j that is forbidden for clock pulses CLK_j. Any clock pulse at the input CLK_j appearing within this interval causes a violation of either the hold or setup time constraint, and thus a circuit malfunction. The first clock pulse that appears at CLK_j after the end of the forbidden interval (marked as the shaded rectangle) transfers the data to the next stage. The preceding clock pulse must appear before the beginning of the forbidden interval.

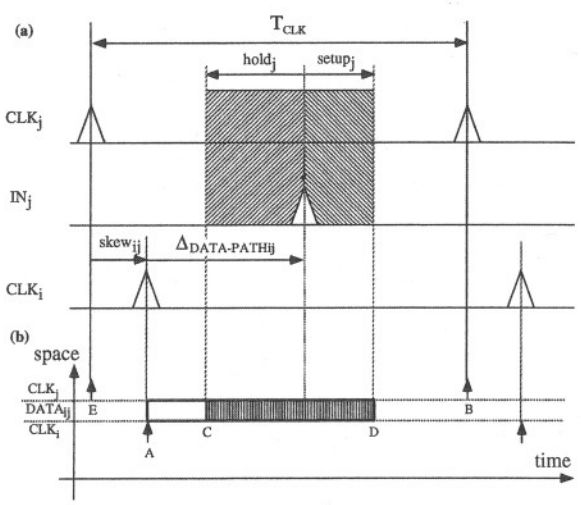


Figure 20. (a) Data flow through the data path between two sequentially adjacent clocked cells, and (b) its simplified graphical representation used in Figs. 21–24.

The operation of the pipeline for zero-skew clocking is shown in Fig. 21. The maximum clock frequency and throughput of the array is determined by the time to process the data through the slowest stage of the pipeline—data path $DATA_{23}$ between cells 2 and 3. Only one data pulse is present in the pipeline stage at any given time.

In Fig. 22, the operation of the circuit for counterflow clocking is shown. The minimum clock period of the circuit is determined by the time to process the data through the slowest stage of the pipeline *plus* the clock skew of this stage. In the most critical data path $DATA_{23}$, CLK_2 initiates processing of the data, and CLK_3 reads the result as soon as this processing is completed. In other non-critical pipeline stages, the data is ready to be transferred to the next stage long before the arrival of the clock pulse.

The operation of the pipeline for concurrent clocking is shown in Fig. 23, and for clock-follow-data clocking in Fig. 24. In both cases, the clock skew of the most critical data path $DATA_{23}$ has been chosen to be the optimal value given by (9) and (12), respectively. For all stages, the next data pulse begins propagating through the data path before the previous pulse *has been transferred* to the next pipeline stage. Additionally, for the slowest stage $DATA_{23}$ (as well as for the data path $DATA_{45}$), the next data pulse starts propagating through the data path before the previous pulse *is ready to be transferred* to the next pipeline stage. From the relation between the clock skews of the critical data path, (9), (12), and from Fig. 13, it can be seen that the timing in the circuit for the minimum clock period in concurrent and clock-follow-data clocking is indistinguishable. As a result, the maximum throughput and the minimum latency are identical in both schemes. This equality between latencies does not hold for clock periods greater than the minimum clock period.

The maximum throughput of the array is equal to the inverse of the minimum clock period. The minimum

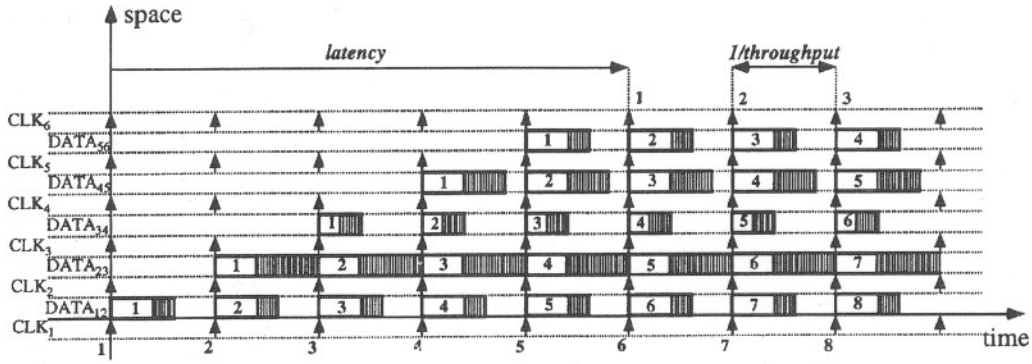


Figure 21. Space vs. time operation of the pipelined one-dimensional array with zero-skew clocking.

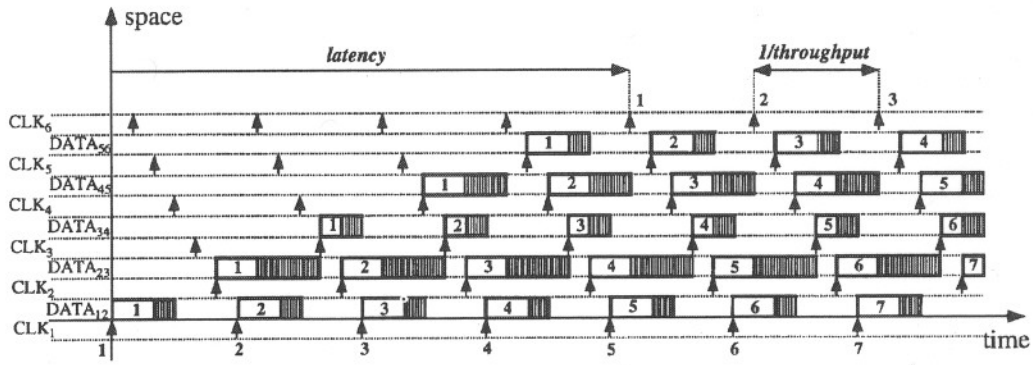


Figure 22. Space vs. time operation of the pipelined one-dimensional array with counterflow clocking.

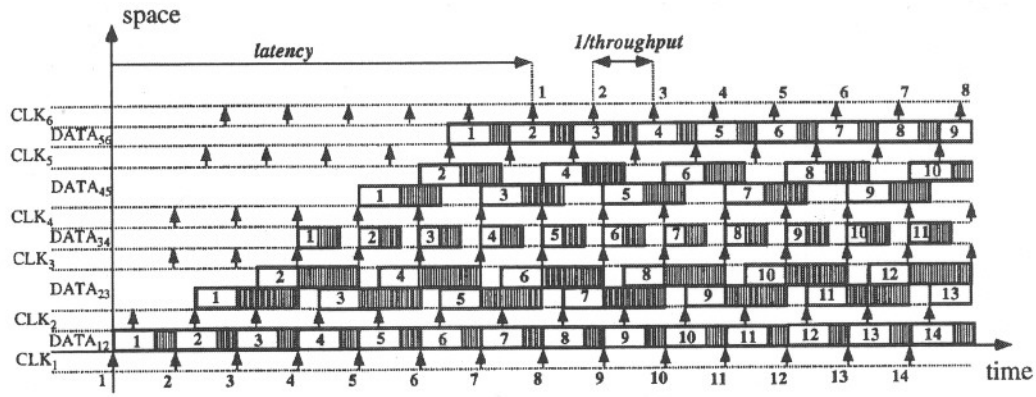


Figure 23. Space vs. time operation of the pipelined one-dimensional array with concurrent clocking.

clock period for each of the clocking schemes discussed in this section is given by (13) with $j = i + 1$. From (18), the relation between the throughputs for each of the different clocking schemes is given by

$$TH_{\text{concurrent}} = TH_{\text{clock-follow-data}} > TH_{\text{zero-skew}} > TH_{\text{counterflow}}. \quad (19)$$

The latency of the circuit for all clocking schemes is given by the following formulae:

$$L_{\text{zero-skew}}(T_{\text{CLK}}) = (N-1) \cdot T_{\text{CLK}}, \quad (20)$$

$$L_{\text{counterflow}}(T_{\text{CLK}}) = (N-1) \cdot T_{\text{CLK}} - \sum_{i=1}^{N-1} \Delta_{\text{CLK-PATH}i(i+1)}, \quad (21)$$

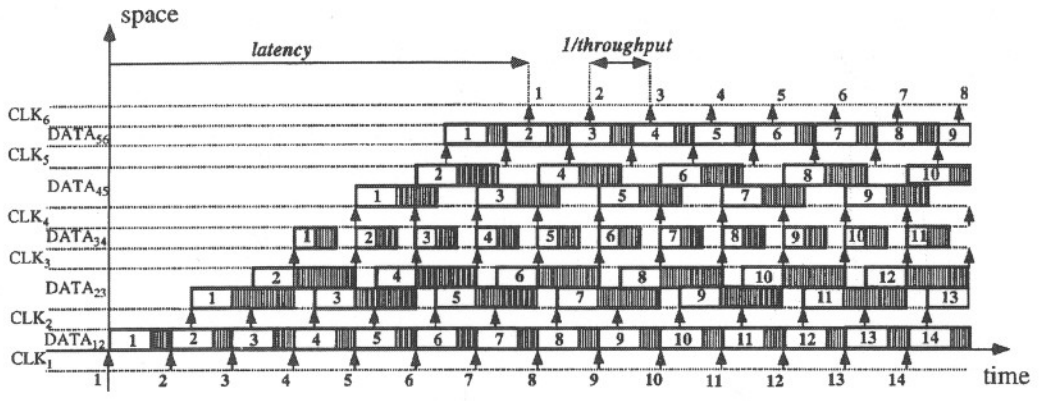


Figure 24. Space vs. time operation of the pipelined one-dimensional array with clock-follow-data clocking.

$$L_{\text{concurrent}}(T_{\text{CLK}}) = (N-1)\cdot T_{\text{CLK}} + \sum_{i=1}^{N-1} |skew_{i(i+1)}|, \quad (22)$$

$$L_{\text{clock-follow-data}}(T_{\text{CLK}}) = \sum_{i=1}^{N-1} |skew'_{i(i+1)}|. \quad (23)$$

Relations between latencies for different clocking modes are not unique. They depend on the parameters of the cells constituting the array and any physical constraints due to the layout. It is possible however to establish these relations unambiguously for most typical parameters.

If $\Delta_{\text{CLK-PATH}_{ij}}$ is constant for all cells, i.e., the physical distance between adjacent cells in the array is the same for all cells, then, from (15), (16), (20), and (21),

$$L_{\text{counterflow}}\left(T^{\text{MIN}}_{\text{counterflow}}\right) = L_{\text{zero-skew}}\left(T^{\text{MIN}}_{\text{zero-skew}}\right). \quad (24)$$

If the clock skew between cells is set to the optimum value which is distinct for concurrent and clock-follow-data clocking, then from (9), (12), (17), (22), and (23),

$$L_{\text{concurrent}}\left(T^{\text{MIN}}_{\text{concurrent}}\right) = L_{\text{clock-follow-data}}\left(T^{\text{MIN}}_{\text{clock-follow-data}}\right). \quad (25)$$

This relation holds despite the fact that in the concurrent scheme, N clock pulses are necessary to drive the data from the input of the first cell to the output of the last cell, while in the clock-follow-data scheme, a single clock pulse drives the data along the entire pipeline.

The minimum latency for concurrent clocking scheme is typically smaller than for zero-skew clocking,

$$L_{\text{concurrent}}\left(T^{\text{MIN}}_{\text{concurrent}}\right) < L_{\text{zero-skew}}\left(T^{\text{MIN}}_{\text{zero-skew}}\right). \quad (26)$$

The latency in all clocking schemes apart from the clock-follow-data scheme is a function of the clock period, and is not defined for the clock periods smaller than the minimum clock period characteristic for each scheme. For clock periods T_{CLK} permitted in all clocking modes (i.e., for T_{CLK} larger than the minimum clock period for counterflow clocking $T^{\text{MIN}}_{\text{counterflow}}$)

$$L_{\text{clock-follow-data}}(T_{\text{CLK}}) < L_{\text{counterflow}}(T_{\text{CLK}}) < L_{\text{zero-skew}}(T_{\text{CLK}}) < L_{\text{concurrent}}(T_{\text{CLK}}). \quad (27)$$

4. Effects of Timing Parameter Variations

The analysis presented in Section 3 concerns the ideal case in which the parameters characterizing devices in the circuit after fabrication are equal to their assumed target values. A more practical design process must account for the effects of process variations on the timing characteristics of a circuit. Taking parameter variations into account results in different *expected* and *worst case* maximum clock frequencies of the circuit and in different optimum values of interconnect delays in the clock distribution network. Including parameter variations in the timing analysis may also lead to the choice of a different synchronization scheme.

Specific features of present day niobium-trilayer technology used to develop medium to large scale RSFQ circuits are described in [71–73]. Two problems must be considered. First, superconducting fabrication

technology is relatively immature compared to well established semiconductor technologies such as CMOS, resulting in much larger parameter variations. Secondly, because of the small volume of the integrated circuits produced by the superconducting foundries, their fabrication process is typically not well characterized.

4.1. *Global vs. Local Timing Parameter Variations*

The minimum clock period of a synchronous circuit using one of the standard clocking schemes described in Subsection 3.4.1 is

$$T_{\mathrm{CLK}}^{\mathrm{MIN}} = skew_{ij} + \Delta_{\mathrm{DATA\text{-}PATH}_{ij}} + setup_j, \quad (28)$$

where the data path between cells i and j is the most critical data path in the circuit. In the presence of parameter variations, the timing parameters included on the right side of (28) can be modeled as random variables. The distribution of these variables is typically assumed to be normal, with mean equal to the nominal value of the timing parameter and standard deviation dependent on the deviations of the fabrication process and the effects of the internal structure of the RSFQ cell [74].

As shown in [74], the timing parameters of the basic RSFQ gates are predominantly affected by *wafer-to-wafer* variations in the *resistance per square* and the *inductance per square*. Other parameters that affect the difference between the actual and nominal values of the timing parameters are the *critical current density* (which affects the electrical characteristics of the junctions) and the global mask-to-wafer biases of the *inductor, resistor*, and *junction sizes* within the circuit.

The effects of deviations in the critical current density and global deviations in the junction size can be significantly decreased by adjusting the *global bias current* that provides the dc power supply to the integrated circuit. Both of these deviations can be approximated for a wafer (or an integrated circuit) using an auxiliary array of test structures, as described in [75]. The bias current can be changed proportionally to the actual values of the critical current density and the normalized junction area [74, 76]. Taking these adjustments into account, a relative 3σ *standard deviation in the delay of the basic RSFQ gates has been estimated to be about* 20% for an existing standard superconductive fabrication process [74]. By taking this result into account, one may estimate the *worst case* minimum clock period of a circuit to be about 20% greater than under nominal conditions.

The other more dramatic effect of parameter variations is the reduction in circuit yield. If for certain actual values of the timing parameters

$$skew_{ij} + \Delta_{\mathrm{DATA\text{-}PATH}_{ij}} \geq hold_j \quad (29)$$

is not satisfied, then the circuit will not work properly for *any* clock frequency. This effect is greatest in the concurrent clocking mode, where the clock skew is chosen to be as close as possible to the boundary corresponding to the hold time violation (Fig. 18(a)). To the extent that the *wafer-to-wafer* variations of global parameters (such as inductance and resistance per square) change all timing parameters proportionally, (29) implies that a violation of the hold time constraint will not result from the global parameter variations. The danger cannot be completely discounted, as the timing parameters included in (29) will not necessarily change in the same proportion. However, changes in the values of these parameters tend to be correlated, which minimizes the effects of global variations on the circuit yield [74].

A more direct deleterious effect on circuit yield results from local *on-chip* variations of the individual parameters, such as the sizes of the junctions, inductors, and resistors, and *on-chip* variations of resistance per square, inductance per square and critical current density. These on-chip variations are typically not well characterized. Preliminary data imply that the local on-chip variations are several times smaller than global wafer-to-wafer parameter variations [71, 75, 77]. Deviations of the timing parameters of various components of the data path that result from local parameter variations are uncorrelated; thus a value of the optimum clock skew for concurrent clocking can be safely chosen according to

$$skew_0^{\mathrm{MIN}} = -\Delta_{\mathrm{DATA\text{-}PATH}_{ij}}^{\mathrm{MIN}} + hold_j^{\mathrm{MAX}}, \quad (30)$$

where the minimum and maximum values are taken to account only for the effects of local parameter variations. Note from (28) that changing the *nominal* value of the clock skew given by (9) to satisfy (30) affects not only the *worst case* but also the *expected* value of the minimum clock period. A similar analysis applies to the clock-follow-data clocking approach.

As a result, in concurrent clocking and clock-follow-data clocking the local parameter variations affect both the *expected* and the *worst case* value of the minimum clock period. Global parameter variations affect

primarily the *worst case* value of the minimum clock period. Both effects are smaller in concurrent clocking vs. clock-follow-data clocking because of smaller absolute delays in the clock paths between sequentially adjacent cells.

In counterflow and zero-skew clocking, global parameter variations typically do not affect the *expected* value of the minimum clock period but change substantially the *worst case* value of the minimum clock period. The effect of local parameter variations is negligible.

4.2. *Local Variations within a Clock Distribution Network*

Heretofore, the clock skew has been assumed to be proportional to the *difference* in delays of the clock paths from the clock source to the inputs of two sequentially adjacent cells. This model of the clock skew is referred to in the literature as a *difference model* [28]. The difference model holds well for straight-line clocking of a linear array, but is inadequate for other topologies such as a binary tree, H-tree, or a *corner clocking* structure, shown in Figs. 25–27 respectively. This is best understood by considering that the *nominal* clock skew between cells X and Y in Figs. 25 and 26 is zero, and in Fig. 27 it is determined only by the segment CC'. However, as a result of local on-chip variations in the clock distribution network the *actual* clock skew between cells X and Y will depend upon the entire crosshatched portion of the network.

Therefore, in order to discuss the clock skew caused by local on-chip variations it is necessary to introduce the more general model of the clock skew called the *summation model* [28]. In the summation model, clock skew is a function of the *sum* of the clock path delays from the nearest common node of the clock distribution network to the inputs of sequentially adjacent cells.

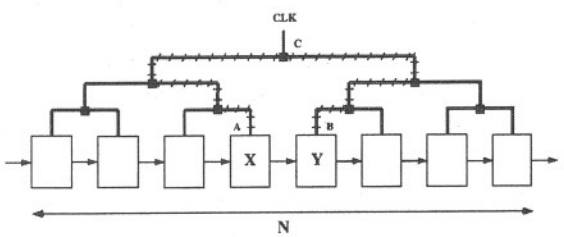


Figure 25. Binary-tree clock distribution network. Local variations in the crosshatched part of the network contribute to the clock skew between cells X and Y.

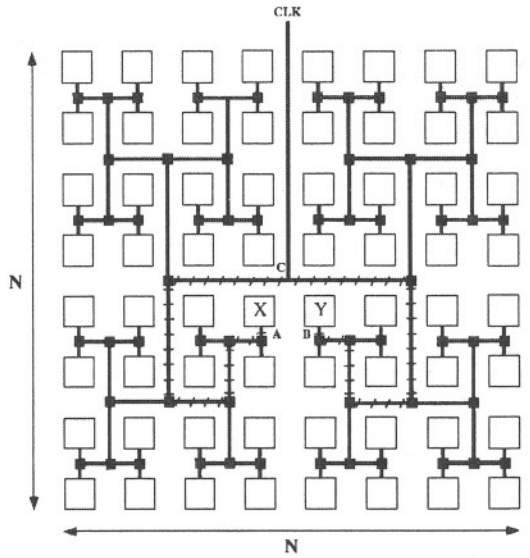


Figure 26. H-tree clock distribution network. Local variations in the crosshatched part of the network contribute to the clock skew between cells X and Y.

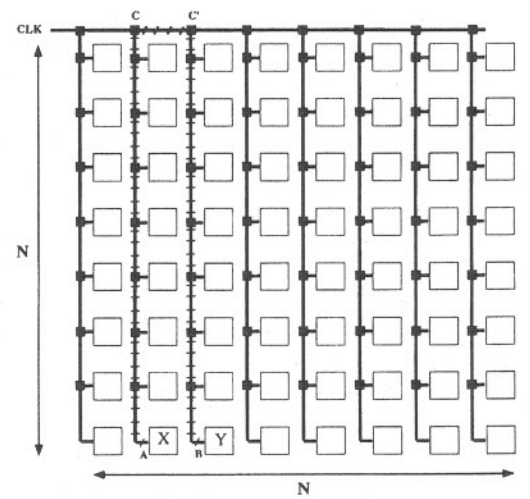


Figure 27. Corner-based clock distribution network. Local variations in the parallel paths CA, C'B contribute to the random clock skew between cells X and Y. The nominal value of the clock skew depends only on the delay CC'. The actual value of this delay changes as a result of both local and global parameter variations.

The effect of the local on-chip variations in the clock distribution network is primarily a function of a network topology, rather than the clocking scheme used within that topology:

For linear arrays, *straight-line* clocking offers an optimum solution in which the difference model of clock

skew applies (see Fig. 29). As a result, this topology of the clock distribution network is perfectly scaleable and works efficiently for an arbitrary number of cells in an array. Asymmetric $M \times N$ systolic arrays with a small constant value of M scale similarly with N to linear arrays. Examples of such circuits include an N-bit serial multiplier ($N \times 3$ array) [3, 5] and an N-bit multiplier-accumulator [$(2N - 1) \times 3$ array] [8].

For the binary tree topology shown in Fig. 25 the data path most critical to local variations is likely to be between cells X and Y. Clock skew is a function of the sum of the path delays CA and CB, between each clock input node and the nearest common ancestor in the binary tree. Therefore, for relatively small N arrays, the clock skew resulting from the local variations in the clock tree increases the *worst case* minimum clock period in the circuit; for large N arrays it may additionally cause an unacceptable reduction in the circuit yield.

The effect of local parameter variations in the clock distribution network on the clock skew is particularly strong for a square array. The worst-case skew grows quickly with the increase in size of the array. Assuming that variations of the clock path delays between any two adjacent cells of the array are independent of each other, the standard deviation of the clock skew for the worst case data path grows proportionally to $\sqrt{N}$, where N is the size of the array [28] (see also [78]). Variations of the resistance per square, inductance per square, and critical current density depend strongly on the physical distance between the corresponding paths of the clock distribution network. For example, the variations tend to be larger in the H-tree network (see Fig. 26), than in the corner-clocked network (see Fig. 27).

Therefore, large-size fully synchronous two-dimensional systolic arrays are difficult to build. Since the local parameter variations are not well characterized to date, it is difficult to judge whether this effect limits the practical sizes of arrays currently developed in RSFQ logic (e.g., 16×16 parallel multiplier described in [1, 2, 26]). Certainly, there exists a limit on the size of a square $N \times N$ systolic array above which synchronous clocking will lead to an unacceptable worst case performance or a very low circuit yield. Depending on the magnitude of the on-chip variations in the timing parameters and the size of the array, either a more conservative clocking scheme (e.g., counterflow vs. concurrent), or a *hybrid* synchronization scheme may be required. In the hybrid scheme presented in [28], an entire array is divided into local synchronous

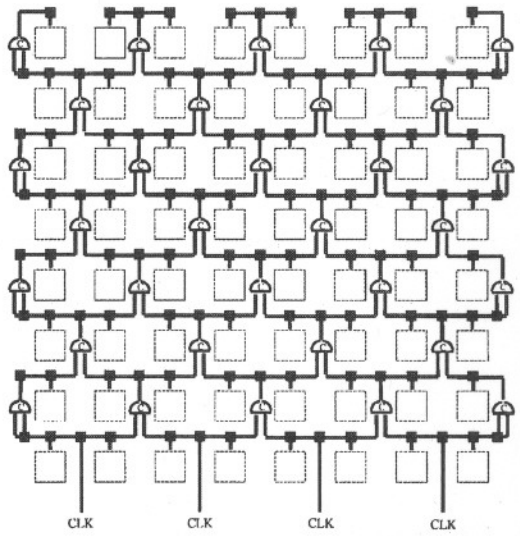


Figure 28. Scheme for resynchronization of the clock signal traveling along different paths in the clock distribution network using coincidence junctions.

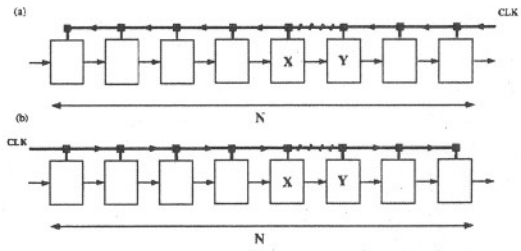


Figure 29. The portions of the straight-line clock distribution network which affect the clock skew between sequentially adjacent cells for (a) counterflow clocking, (b) concurrent clocking.

subarrays with local clocks controlled using an asynchronous handshaking protocol.

Another solution, developed specifically for RSFQ arrays, is described in [1, 26]. In this approach, shown in Fig. 28, clock signals traveling along different paths in the clock distribution network are resynchronized using coincidence junctions. A *coincidence junction* [1, 26, 47] produces an output pulse only after an input pulse has arrived at both of its inputs. Statistically, in the circuit shown in Fig. 28, the clock skew between any two neighboring cells is substantially reduced.

4.3. *Optimal Choice of Interconnect Delays*

A quantitative analysis of the effects of global and local variations on the performance of a circuit and thus also

the optimal choice of interconnect delays is difficult to perform analytically, and usually requires computationally intensive Monte Carlo simulations. These computations can be substantially sped-up by using a behavioral simulation rather than a circuit level simulation, as described in [30, 79]. Another approach, based on an approximate worst case analysis, is presented in [27]. This approach leads to correct but not necessarily optimal solutions. The design of circuits with concurrent clocking is particularly challenging since a *nominal* value of the clock skew must be chosen considering the effects of the global and local parameter variations. An incorrect choice may lead to a large percentage of the integrated circuit not working properly for any clock frequency. Good characterization data of the fabrication process is a necessary condition for a correct quantitative analysis of the circuit performance and the design of the *optimum* clock distribution network.

5. Asynchronous Timing

In semiconductor VLSI, asynchronous timing has been for many years considered a possible alternative to synchronous clocking [20, 37]. Its main advantages include modularity, reliability and high resistance to fabrication process variations. Nevertheless, asynchronous clocking has *not* been widely accepted in semiconductor circuit design due to unsatisfactory performance in terms of area, speed, and power consumption, as well as complicated design and testing procedures [29].

Asynchronous timing requires local signaling between adjacent cells. This signaling is naturally based on the concept of *events* such as *request* and *acknowledge*. In semiconductor logic events are coded using voltage state *transitions* (rising edges in *return-to-zero* signaling, and rising and falling edges in *non-return-to-zero* signaling). Semiconductor logic elements that process voltage transitions (e.g., Muller C-element, Toggle, Select) are complex and slow compared to logic gates that process *voltage levels*. In RSFQ logic, events are coded using SFQ pulses. Asynchronous logic elements that process SFQ pulses (e.g., confluence buffer, coincidence junction) are simple and fast compared to RSFQ logic gates (such as AND, OR, XOR), and therefore RSFQ asynchronous circuits can approach the speed of synchronous circuits. Because of this asynchronous clocking appears to be easier and more natural to implement in RSFQ circuits than in semiconductor voltage-state logic. For complex RSFQ circuits, the disadvantage of larger area and power consumption required for local signaling in asynchronous circuits may be compensated by circuit modularity and the larger tolerance to fabrication process variations.

5.1. *Dual-Rail Logic*

The only asynchronous timing approach reported to be actually used in the design of a large scale RSFQ circuit [16] is based on *dual-rail logic*. Adapting dual-rail logic for use with RSFQ gates has been investigated in [38, 39, 55–58].

In *dual-rail logic*, each signal is transmitted using two signal lines, denoted true- and false-. The appearance of an SFQ pulse on the true-line is defined as the logical "1", and the appearance of the pulse on the false-line as the logical "0". This convention differs significantly from the Basic RSFQ Convention described in Section 2. Therefore, any RSFQ gate which should be used as the core of a dual-rail logic cell must be redesigned by adding special input and output circuitry. First, the gate is extended with a second complementary output OUT\. Each time the cell performs a logic operation, an SFQ pulse is created at one and only one of the cell outputs, OUT or OUT\. Additionally, the cell is supplemented with the input circuitry used to accept dual-rail inputs and to internally generate the clock pulse driving the core RSFQ gate.

The input circuitry for a single-input gate can have a form of a confluence buffer with two delay lines: clock line C-JTL and data line *D*-JTL as shown in Fig. 30(a). A pulse that appears at either input a or $a\backslash$ of the cell generates a pulse at the output of the confluence buffer, CB. This pulse, delayed by the JTL line C-JTL, is used to clock the RSFQ gate. The timing constraints in the circuit are described by

$$\Delta_{\text{D-JTL}} + \text{setup} \leq \Delta_{\text{CB}} + \Delta_{\text{C-JTL}}, \tag{31}$$

$$\Delta_{\text{D-JTL}} + T_{\text{IN}} \geq \Delta_{\text{CB}} + \Delta_{\text{C-JTL}} + \text{hold}, \tag{32}$$

where T_{IN} is the period of the input data signal. From (31) and (32),

$$T_{\text{IN}} \geq \Delta_{\text{CB}} + \Delta_{\text{C-JTL}} - \Delta_{\text{D-JTL}} + \text{hold}. \tag{33}$$

The minimum value of the input period is

$$T_{\text{MIN}} = \text{hold} + \text{setup}, \tag{34}$$

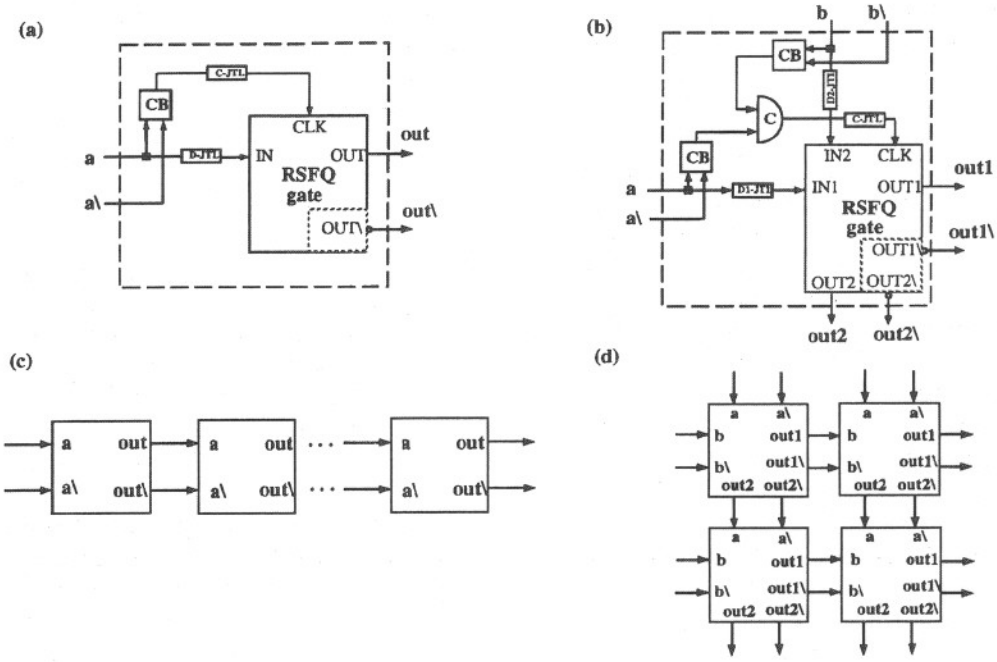


Figure 30. Internal structure of a dual-rail cell based on (a) one-input RSFQ gate, (b) two-input RSFQ gate; and the method of connecting dual-rail cells into (c) a linear array, (d) a rectangular array. Notation: CB—confluence buffer, C-JTL—clock path JTL, D-JTL—data path JTL, C—coincidence junction.

and is obtained for a choice of interconnect delays according to

$$\Delta_{\text{C-JTL}} - \Delta_{\text{D-JTL}} = \text{setup} - \Delta_{\text{CB}}. \qquad (35)$$

For the optimum choice of the interconnect delays, the data pulse appears at the input of the RSFQ gate exactly a setup time before the clock pulse arrives (the same as for clock-follow-data synchronous clocking). This makes the circuit vulnerable to fabrication process variations. The actual optimum value of the interconnect delays $\Delta_{\text{C-JTL}}$ and $\Delta_{\text{D-JTL}}$ must be derived taking parameter variations into account.

Dual-rail cells designed according to these rules can be connected into a linear array with unidirectional data-flow without any additional circuitry, as shown in Fig. 30(c). Note that in this configuration no *acknowledge* signal is used, and the *request* signal does not appear explicitly but rather is integrated with the dual-rail data signals. As a result, the circuit is vulnerable to timing violations resulting from the next data appearing at the cell input before the previous data is accepted. *The maximum input rate of the signal driving the first cell of the array is limited by the maximum input rate of the slowest gate in the array.* If the interval between any two external input data pulses is smaller than the minimum input period for any cell in the array, the timing constraints are violated, leading to a circuit malfunction.

Therefore, the overall performance of this simple array in dual-rail logic in terms of the latency and the maximum throughput is comparable to the performance in synchronous clock-follow-data clocking. The device overhead and design complexity of a dual-rail logic is significantly greater.

In case of a two-input dual-rail cell, the cell input circuitry becomes even more complicated (Fig. 30(b)). The output of the confluence buffer associated with each of the dual-rail inputs feeds the input of the coincidence junction. The coincidence junction generates the clock pulse only after both input data signals have arrived. The maximum input rate of the cell can be derived using an analysis similar to that performed for one-input cells. The important difference is that the maximum data rate for each input depends not only on the internal delays in the circuit but also on the interval between the arrival of the dual-rail data signals at two different inputs of the cell. As a result, the maximum input rate for a gate becomes dependent on the circuitry

surrounding the gate and the timing characteristics of the external input data sources.

A two-dimensional array composed of two-input dual-rail cells is shown in Fig. 30(d). For a square $N \times N$ array, dual rail logic offers a unique advantage by eliminating the effect of clock skew due to local parameter variations in the clock distribution network discussed in Section 4.2. However, disadvantages of the scheme include

a) a large device overhead resulting from using two confluence buffers, one coincidence junction and complementary output circuitry per every two-input gate in the circuit;
b) vulnerability to discrepancies between input rates at any two inputs in the circuit.

5.2. *Micropipelines*

The other asynchronous scheme considered for application in RSFQ one-dimensional arrays is the *micropipeline*. This scheme, known from semiconductor circuit design [37], appears to be easily adaptable to RSFQ logic [1, 26, 60, 61]. The scheme is based on the use of coincidence junctions (Muller C-elements in semiconductor logic) to generate the clock for each cell in the pipeline on the basis of the *request* signal generated by the previous cell in the pipeline, and the *acknowledge* signal generated by the next cell in the pipeline.

From the analysis presented in [37], this scheme does not offer any advantage in speed compared to a fully synchronous methodology (e.g., concurrent clocking). The design of the circuitry for generation signaling events (*acknowledge* and *request*) must take into account the effects of the local timing parameter variations. The disadvantage of the scheme lies in its large device overhead (one coincidence junction plus multiple JTL stages per each clocked cell), and the requisite complex operation.

6. Two-Phase Synchronous Clocking

Two-phase clocking is a common approach used in semiconductor circuit design in which a two-phase master-slave double latch is used as a storage component [22]. Multiple phases of the clock relax the timing constraints in the circuit, and thus increase circuit tolerance to variations in the fabrication process. The disadvantage is the area/device overhead resulting from the second clock path and more complex storage components.

In this section a novel two-phase clocking scheme applicable to RSFQ circuits of any complexity is introduced. We show that high performance, robustness, and design simplicity may justify two-phase clocking despite the area overhead inherent in this scheme.

An initial attempt to apply two-phase clocking to RSFQ circuits was reported in [80]. In this paper, two-phase clocking is used to drive a long linear shift register. The motivation is to assure that the circuit works correctly at a very low clock frequency applied during functional testing, independently of the parameter variations in the circuit. No attempt is made to optimize the performance of the circuit. Two phases of the clock are generated using complementary DC/SFQ converters, and distributed independently along the data path of the shift register. As a result, the design is vulnerable to independent local parameter variations occurring in the two parallel clock paths used to distribute each phase of the clock.

An enhanced version of this *two-phase concurrent clocking* scheme applicable to any general one- and two-dimensional arrays as well as to RSFQ circuits with a less regular topology is presented here. The performance of this scheme is analyzed, and its advantages and disadvantages are compared to single-phase concurrent clocking.

In RSFQ two-phase clocking, the phases of the clock are shifted from each other by half of the clock period, as shown in Fig. 31(a). Both phases of the clock can be generated from one signal with twice the clock frequency using a T flip-flop, as shown in Fig. 31(b). A separate T flip-flop can be associated with each clocked cell in the circuit, or can be used to generate both phases for a whole sequence of clocked cells. In the latter case,

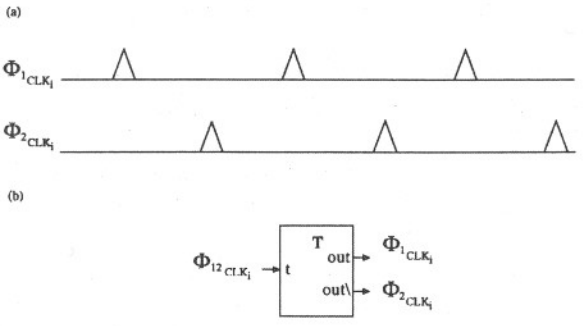


Figure 31. Two-phase clocking in RSFQ logic. (a) phases of the clock; (b) method of generating both phases from a single signal operating at twice the clock frequency.

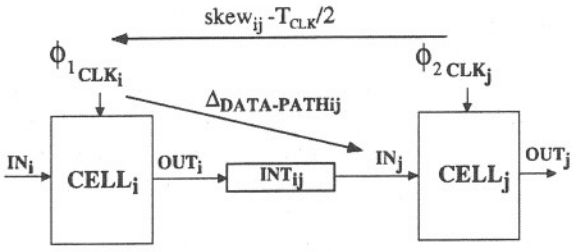


Figure 32. Data path between two physically adjacent RSFQ cells in two-phase clocking.

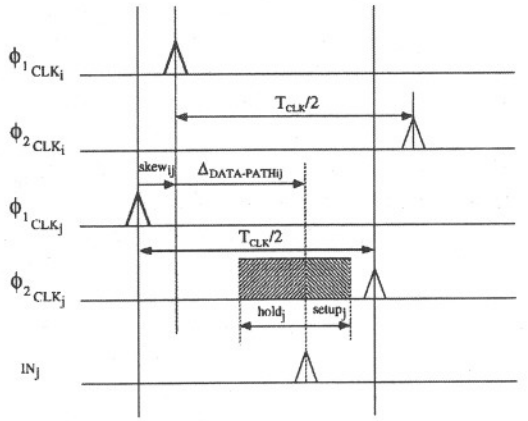


Figure 33. Exchange of data between two physically adjacent cells in two-phase clocking.

both clock phases are distributed independently at an interval between two consecutive T flip-flops.

The data path between two sequentially adjacent cells is shown in Fig. 32. Timing diagrams depicting the exchange of data between two sequentially adjacent cells are given in Fig. 33.

Conditions for the correct operation of the circuit are

$$T_{\text{CLK}}/2 \geq skew_{ij} + \Delta_{\text{DATA-PATH}_{ij}} + setup_j, \quad (36)$$

$$skew_{ij} + \Delta_{\text{DATA-PATH}_{ij}} + T_{\text{CLK}}/2 \geq hold_j. \quad (37)$$

In Fig. 34, the operating region of the circuit as a function of the clock skew and the clock period is shown. By comparing the shape of this operating region with the regions for single-phase clocking presented in Fig. 12, it can be concluded that for two-phase clocking:

a) There does not exist a region of clock skew values for which the circuit does not work for any clock frequency. For any possible value of the clock skew, there exist a minimum clock period above which the circuit works correctly for any clock period.

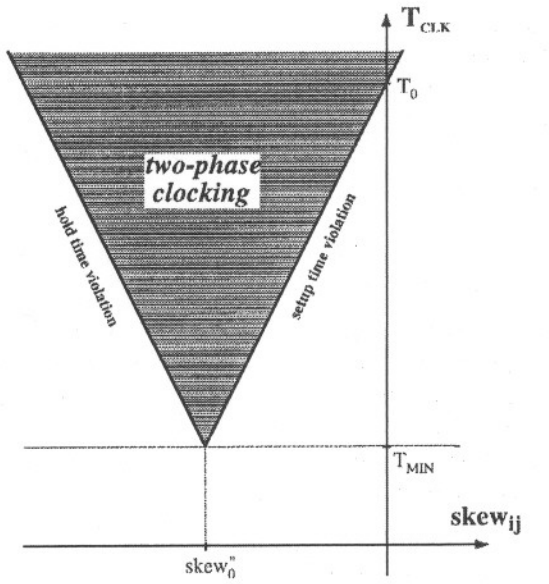


Figure 34. Operating region for two phase clocking of a circuit composed of two sequentially adjacent cells, as a function of the clock period and the clock skew.

b) The minimum clock period in the circuit is limited by

$$T_{\text{MIN}} = hold_j + setup_j, \quad (38)$$

and the optimal choice of the clock skew is

$$skew_0'' = -\Delta_{\text{DATA-PATH}_{ij}} - setup_j/2 + hold_j/2. \quad (39)$$

The optimum clock skew in two-phase clocking, $skew_0''$, is related to the optimum clock skew for single-phase concurrent clocking, $skew_0$, [given by (9)] and single-phase clock-follow-data clocking, $skew_0'$, [given by (12)] according to

$$skew_0'' = \frac{skew_0 + skew_0'}{2} \quad (40)$$

The minimum clock period, without taking parameter variations into account, is identical for all three clocking schemes. The position of the data pulse within the clock period for the optimum value of the clock skew in two-phase clocking is shown in Fig. 35.

c) The optimal value of the clock skew is identical regardless of whether parameter variations are considered. This feature simplifies considerably the design of the circuit by eliminating the need for the computationally intensive Monte Carlo simulations

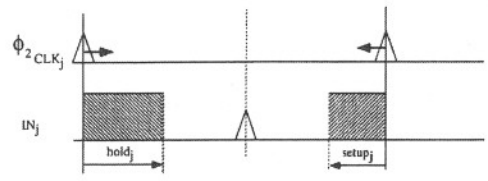


Figure 35. The position of the data pulse within the clock period in two-phase clocking for the optimum value of the clock skew.

necessary to determine the optimal clock skew in single-phase concurrent clocking.

d) The *expected* value of the minimum clock period is the same with or without taking parameter variations into account. As a result, the expected value is smaller than in single-phase concurrent clocking. The worst case value of the clock period in both schemes is comparable.

Thus the advantage of the two-phase synchronous clocking are robustness, high-performance, and the simplicity of the design procedure. The disadvantage of this approach the additional overhead circuitry required to generate and distribute the second phase of the clock.

7. Conclusions

The timing of medium to large scale RSFQ circuits follows the well-established principles and methodologies of the timing applied to high speed VLSI semiconductor-based circuits. There are significant qualitative differences which arise from

a) the lack of purely combinational logic in RSFQ circuits;
b) the low fanout of RSFQ gates;
c) a different suite of elementary gates in the two technologies.

There are other differences which primarily arise from the much greater operating frequencies in RSFQ logic. The most important of these is the inefficiency of applying equipotential clocking to multi-gigahertz large clock distribution networks. Pipelined (flow) clocking should be used instead in both RSFQ and semiconductor-based circuits.

Zero-skew clocking, which is ubiquitous in semiconductor circuits, has no particular advantage when applied to RSFQ logic. Non-zero-skew clocking schemes can be chosen either for superior performance or for extended tolerance to fabrication process variations. Although these advantages may be easier to exploit in RSFQ circuits, the same clocking schemes also apply to the design of high speed semiconductor circuits.

The choice of clocking scheme for a particular RSFQ circuit depends upon:

a) the topology of the circuit (one-dimensional vs. two-dimensional array, regular vs. irregular structure);
b) the performance requirements (throughput, latency) of the circuit;
c) global and local parameter variations in the circuit;
d) complexity of the design procedure (computationally intensive Monte Carlo analysis vs. analytical estimations);
e) the device, area, and power consumption overhead;
f) the complexity of the physical layout.

For circuits which are essentially one-dimensional, $N \times 1$ arrays and asymmetric $N \times M$ arrays with small M, the natural choices are the *straight-line* synchronous clocking schemes. *Counterflow clocking* offers the advantages of high robustness to timing parameter variations, small area, and a simple design procedure, but at the cost of reduced circuit throughput. When the highest clock frequency is of primary concern, *concurrent clocking* should be considered. An aggressive application of this scheme will reduce the expected yield of the circuit unless there is a good quantitative knowledge of the fabrication process variations. The design procedure leading to the optimum solution may require intensive Monte Carlo simulations, although suboptimal solutions can be obtained using simpler analytical methods. Concurrent clocking tends to require a larger number of JTL stages in the clock paths compared to counterflow clocking, and thus a greater overhead in circuit area and in layout complexity is expected.

This paper introduces a new clocking scheme, *two-phase clocking*, which is expected to offer better performance than concurrent clocking, better tolerance to fabrication process variations than counterflow clocking, and an extremely simple design procedure. Also in two-phase clocking, the choice of the optimum interconnects in the circuit does not require any knowledge of the timing parameter variations. Interconnect delays within the clock distribution network are similar to concurrent clocking. The only disadvantage of two-phase clocking is the area overhead resulting from the

necessity to generate both clock phases for every cell of a linear $N \times 1$ array or every column of an asymmetric $N \times M$ array. A single T flip-flop (5 Josephson junctions) per cell or column of cells is sufficient for this purpose. In all of these synchronous schemes applied to $N \times 1$ arrays or asymmetric $N \times M$ arrays with small M, the maximum clock frequency is independent of N.

Asynchronous schemes such as *dual-rail clocking* and *micropipelines* can also be successfully applied to linear and asymmetric arrays, but these schemes do not offer any advantages over synchronous schemes in either performance, robustness, or design complexity. Either scheme can be adjusted (by the appropriate choice of interconnect delays) to provide either the performance equivalent of concurrent clocking or the robustness of counterflow clocking. Both schemes, however, require a significant overhead, which is comparable to or greater than required by two-phase clocking. The design for optimum performance is equally complex as for concurrent clocking and requires good knowledge of the timing parameter variations.

For a two-dimensional symmetric square $N \times N$ arrays the situation is more complicated. The additional effects of the local parameter variations in corresponding paths of the clock distribution network (a *summation model* of the clock skew) must be considered. In all of the synchronous schemes, the performance of the circuit deteriorates with an increase of the array size N by a factor proportional to at least $\sqrt{N}$. Depending on the magnitude of the on-chip variations and the topology of the clock distribution network, these effects may become critical for different sizes of N. In particular, it is possible that the constant factors may be sufficiently small for practical sizes of RSFQ arrays. For all synchronous schemes, the worst case maximum clock frequency deteriorates with increasing N. Additionally, for all single-phase clocking schemes, there exists a value of N above which the yield of the circuit begins to decrease. This value of N is smallest for concurrent clocking and largest for counterflow clocking. In two-phase clocking, increasing the array size deteriorates only the *worst case* circuit performance. Neither the *expected* performance of the circuit nor the functional circuit yield at low speed is affected by an increase of the array size N.

Asynchronous schemes scale better with increasing N. Again, the primary disadvantage of these schemes is the large circuit overhead. These schemes are also more difficult to analyze and test than synchronous schemes. As a result, the use of asynchronous timing methodologies may be limited to circuits of large N. Finally, hybrid synchronization schemes which use asynchronous strategies in tandem with simpler synchronous schemes are likely to be advantageous for large RSFQ circuits.

Acknowledgment

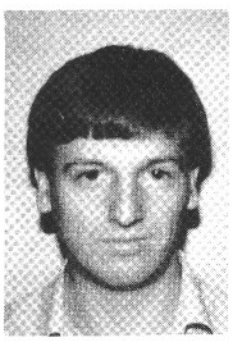
This work was supported in part by the Rochester University Research Initiative sponsored by the US Army Research Office.

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, pp. 3–28, 1991.
2. K.K. Likharev, "Rapid single-flux-quantum logic," in *The New Superconductor Electronics*, H. Weinstock and R. Ralston (Eds.), Kluwer, Dordrecht, pp. 423–452, 1993.
3. O.A. Mukhanov, P.D. Bradley, S.B. Kaplan, S.V. Rylov, and A.F. Kirichenko, "Design and operation of RSFQ circuits for digital signal processing," *Proc. 5th Int. Supercond. Electron. Conf.*, Nagoya, Japan, Sept. 1995, pp. 27–30.
4. K.K. Likharev, "Ultrafast superconcductor digital electronics: RSFQ technology roadmap," *Czechoslovak J. Phys.*, Suppl. S6, Vol. 46, 1996.
5. O.A. Mukhanov and A.F. Kirichenko, "Implementation of a FFT radix 2 butterfly using serial RSFQ multiplier-adders," *IEEE Trans. Appl. Supercond.*, Vol. 5, pp. 2461–2464, 1995.
6. J.C. Lin, V.K. Semenov, and K.K. Likharev, "Design of SFQ-counting analog-to-digital converter," *IEEE Trans. Appl. Supercond.*, Vol. 5, pp. 2252–2259, 1995.
7. V.K. Semenov, Yu. Polyakov, and D. Schneider, "Preliminary results on the analog-to-digital converter based on RSFQ logic," *CPEM96 Conf. Digest Suppl.*, Braunschweig, Germany, June 1996, pp. 15–16.
8. Q.P. Herr et al. "Design and low speed testing of a four-bit RSFQ multiplier-accumulator," *IEEE Trans. Appl. Supercond.*, Vol. 7, 1997.
9. Q.P. Herr, K. Gaj, A.M. Herr, N. Vukovic, C.A. Mancini, M.F. Bocko, and M.J. Feldman, "High speed testing of a four-bit RSFQ decimation digital filter," *IEEE Trans. Appl. Supercond.*, Vol. 7, 1997.
10. P.I. Bunyk et al., "High-speed single-flux-quantum circuit using planarized niobium-trilayer Josephson junction technology," *Appl. Phys. Lett.*, Vol. 66, pp. 646–648, 1995.
11. Q.P. Herr and M.J. Feldman, "Error rate of a superconducting circuit," *Appl. Phys. Lett.*, Vol. 69, pp. 694–695, 1996.
12. D.Y. Zinoviev and K.K. Likharev, "Feasibility study of RSFQ-based self-routing nonblocking digital switches," *IEEE Trans. Appl. Supercond.*, Vol. 7, 1997.
13. Q. Ke, B.J. Dalrymple, D.J. Durand, and J.W. Spargo, "Single flux quantum crossbar switch," *IEEE Trans. Appl. Supercond.*, Vol. 7, 1997.

14. N.B. Dubash, P.-F. Yuh, V.V. Borzenets, T. Van Duzer, and S.R. Whiteley, "SFQ data communication switch," *IEEE Trans. Appl. Supercond.*, Vol. 7, 1997.
15. O.A. Mukhanov and S.V. Rylov, "Time-to-digital converters based on RSFQ digital counters," *IEEE Trans. Appl. Supercond.*, Vol. 7, 1997.
16. A.V. Rylyakov and S.V. Polonsky, "All digital 1-bit RSFQ autocorrelator for radioastronomy applications: Design and experimental results," *IEEE Trans. Appl. Supercond.*, Vol. 7, 1997.
17. A.V. Rylyakov, "New design of single-bit all-digital RSFQ autocorrelator," *IEEE Trans. Appl. Supercond.*, Vol. 7, 1997.
18. G. Taubes, "Redefining the supercomputer," *Science*, Vol. 273, pp. 1655–1657, 1996.
19. G. Gao, K.K. Likharev, P.C. Messina, and T.L. Sterling, "Hybrid technology multithreaded architecture," in *Proc of PetaFlops Architecture Workshop*, to be published; see also the Web site http://www.cesdis.gsfc.nasa.gov/petaflops/peta.html.
20. C. Mead and L. Conway, *Introduction to VLSI Systems*, Addison-Wesley, Reading, MA, 1980.
21. M. Hatamian, "Chapter 6, Understanding clock skew in synchronous systems," in *Concurrent Computations (Algorithms, Architecture, and Technology)*, S.K. Tewksbury, B.W. Dickinson, and S.C. Schwartz (Eds.), Plenum Publishing, New York, pp. 87–96, 1988.
22. H.B. Bakoglu, *Circuits, Interconnections and Packaging for VLSI*, Addison-Wesley, 1990.
23. H.M. Teresa, *Synchronization Design for Digital Systems*, Kluwer Academic Publishers, 1991.
24. E.G. Friedman, "Clock distribution design in VLSI circuits—an overview," *Proc. IEEE Int'l Symp. Circuits Syst.*, pp. 1475–1478, May 1993.
25. E.G. Friedman (Ed.), *Clock Distribution Networks in VLSI Circuits and Systems*, IEEE Press, 1995.
26. O.A. Mukhanov, S.V. Rylov, V.K. Semenov, and S.V. Vyshenskii, "RSFQ logic arithmetic," *IEEE Trans. Magnetics*, Vol. 25, pp. 857–860, 1989.
27. K. Gaj, E.G. Friedman, M.J. Feldman, and A. Krasniewski, "A clock distribution scheme for large RSFQ circuits," *IEEE Trans. Appl. Supercond.*, Vol. 5, pp. 3320–3324, 1995.
28. A.L. Fisher and H.T. Kung, "Synchronizing large VLSI processor arrays," *IEEE Trans. Comput.*, Vol. C-34, pp. 734–740, 1985.
29. M. Afghahi and C. Svensson, "Performance of synchronous and asynchronous schemes for VLSI systems," *IEEE Trans. Comput.*, Vol. C-41, pp. 858–872, 1992.
30. K. Gaj, C.-H. Cheah, E.G. Friedman, and M.J. Feldman, "Optimal clocking design for large RSFQ circuits using Verilog HDL," (in preparation).
31. J.P. Fishburn, "Clock skew optimization," *IEEE Trans. Comput.*, Vol. 39, pp. 945–951, 1990.
32. J.L. Neves and E.G. Friedman, "Topological design of clock distribution networks based on non-zero clock skew," *Proc. 36th Midwest Symp. Circuits Syst.*, pp. 468–471, Aug. 1993.
33. J.L. Neves and E.G. Friedman, "Design methodology for synthesizing clock distribution networks exploiting nonzero localized clock skew," *IEEE Trans. VLSI Syst.*, Vol. 4, pp. 286–291, 1996.
34. J.L. Neves and E.G. Friedman, "Circuit synthesis of clock distribution networks based on non-zero clock skew," *Proc. IEEE Int'l Symp. Circuits Syst.*, pp. 4.175–4.178, June 1994.
35. J.L. Neves and E.G. Friedman, "Automated synthesis of skew-based clock distribution networks," *Int'l J.VLSI Design*, March 1997.
36. S.Y. Kung and R.J. Gal-Ezer, "Synchronous versus asynchronous computation in very large scale integrated (VLSI) array processors," *Proc. of SPIE*, Vol. 341, pp. 53–65, May 1982.
37. I.E. Sutherland, "Micropipelines," *Comm. ACM*, Vol. 32, pp. 720–738, 1989.
38. Z.J. Deng, S.R. Whiteley, and T. Van Duzer, "Data-driven self-timing of RSFQ digital integrated circuits," ext. abstract, *5th Int'l Supercond. Electr. Conf. (ISEC)*, Nagoya, Sept. 1995, pp. 189–191.
39. M. Maezawa, I. Kurosawa, Y. Kameda, and T. Nanya, "Pulse-driven dual-rail logic gate family based on rapid single-flux-quantum (RSFQ) devices for asynchronous circuits," *Proc. 2nd Int. Symposium Advanced Research in Asynchronous Circuits and Systems*, pp. 134–142, March 1996.
40. M. Hatamian and G.L. Cash, "Parallel bit-level pipelined VLSI design for high-speed signal processing," *Proc. IEEE*, Vol. 75, pp. 1192–1202, Sept. 1987.
41. D.C. Wong, G.D. Micheli, and M.J. Flynn, "Designing of high-performance digital circuits using wave pipelining: Algorithms and practical experiences," *IEEE Trans. Comp. Aid. Design Int. Circ. and Syst.*, Vol. 12, pp. 25–46, 1993.
42. D.A. Joy and M.J. Ciesielski, "Clock period minimization with wave pipelining," *IEEE Trans. Comp.-Aid. Design Int. Circ. and Syst.*, Vol. 12, pp. 461–472, 1993.
43. Q. Ke and M.J. Feldman, "Single flux quantum circuits using the residue number system," *IEEE Trans. Appl. Supercond.*, Vol. 5, pp. 2988–2991, 1995.
44. Q. Ke, "Superconducting single flux quantum circuits using the residue number system," Ph.D. Thesis, University of Rochester, 1995.
45. K.K. Likharev, O.A. Mukhanov, and V.K. Semenov, "Resistive single flux quantum logic for the Josephson-junction technology," in *SQUID'85*, Berlin, Germany, W. de Gruyter, pp. 1103–1108, 1985.
46. S.V. Polonsky, V.K. Semenov, and D.F. Schneider, "Transmission of single-flux- quantum pulses along superconducting microstrip lines," *IEEE Trans. Appl. Supercond.*, Vol. 3, pp. 2598–2600, 1993.
47. S.V. Polonsky et al., "New RSFQ circuits," *IEEE Trans. Appl. Supercond.*, Vol. 3, pp. 2566–2577, 1993.
48. N. Weste and K. Eshraghian, *Principles of CMOS VLSI Design—A Systems Perspective*, Addison-Wesley, Reading, MA, 1985.
49. S.B. Kaplan and O.A. Mukhanov, "Operation of a superconductive demultiplexer using rapid single flux quantum (RSFQ) technology," *IEEE Trans. Appl. Supercond.*, Vol. 5, pp. 2853–2856, 1995.
50. A.F. Kirichenko, V.K. Semenov, Y.K. Kwong, and V. Nandakumar, "4-bit rapid single-flux-quantum decoder," *IEEE Trans. Appl. Supercond.*, Vol. 5, pp. 2857–2860, 1995.
51. S.V. Polonsky, V.K. Semenov, and A.F. Kirichenko, "Single flux, quantum B flip-flop and its possible applications," *IEEE Trans. Appl. Supercond.*, Vol. 4, pp. 9–18, 1994.
52. S.S. Martinet and M.F. Bocko, "Simulation and optimization of binary full-adder cells in RSFQ logic," *IEEE Trans. Appl. Supercond.*, Vol. 3, pp. 2720–2723, 1993.

53. A.F. Kirichenko and O.A. Mukhanov, "Implementation of novel 'push-forward' RSFQ carry-save serial adders," *IEEE Trans. Appl. Supercond.*, Vol. 5, pp. 3010–3013, 1995.
54. S.V. Polonsky, J.C. Lin, and A.V. Rylyakov, "RSFQ arithmetic blocks for DSP applications," *IEEE Trans. Appl. Supercond.*, Vol. 5, pp. 2823–2826, 1995.
55. Z.J. Deng, N. Yoshikawa, S.R. Whiteley, and T. Van Duzer, "Data-driven self-timed RSFQ digital integrated circuit and system," *IEEE Trans. Appl. Supercond.*, Vol. 7, 1997.
56. 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 5th Int'l Supercond. Electr. Conf. (ISEC)*, Nagoya, Sept. 1995, pp. 204–206.
57. I. Kurosawa, H. Nakagawa, M. Aoyagi, M. Maezawa, Y. Kameda, and T. Nanya, "A basic circuit for asynchronous superconductive logic using RSFQ gates," *Supercond. Sci. Technol.*, Vol. 8, pp. A46–A49, 1995.
58. M. Maezawa, I. Kurosawa, M. Aoyagi, H. Nakagawa, Y. Kameda, and T. Nanya, "Rapid single-flux-quantum dual-rail logic for asynchronous circuits," *IEEE Trans. Appl. Supercond.*, Vol. 7, 1997.
59. C.A. Mancini, N. Vukovic, A.M. Herr, K. Gaj, M.F. Bocko, and M.J. Feldman, "RSFQ circular shift registers," *IEEE Trans. Appl. Supercond.*, Vol. 7, 1997.
60. P. Bunyk and A. Kidiyarova-Shevchenko, "RSFQ microprocessor: New design approaches," *IEEE Trans. Appl. Supercond.*, Vol. 7, 1997.
61. P. Bunyk and V.K. Semenov, "Design of an RSFQ microprocessor," *IEEE Trans. Appl. Supercond.*, Vol. 5, pp. 3325–3328, 1995.
62. P. Patra and D.S. Fussell, "Conservative delay-insensitive circuits," *Proc. 4th Workshop on Physics and Computation: PhysComp96*, Boston, 1996, pp. 248–259.
63. J. Fleischman and T. Van Duzer, "Computer architecture issues in superconductive microprocessors," *IEEE Trans. Appl. Supercond.*, Vol. 3, pp. 2716–2719, 1993.
64. V.K. Kaplunenko, "Fluxon interaction in an overdamped Josephson transmission line," *Appl. Phys. Lett.*, Vol. 66, pp. 3365–3367, 1995.
65. J.-C. Lin and V.K. Semenov, "Timing circuits for RSFQ digital systems," *IEEE Trans. Appl. Supercond.*, Vol. 5, pp. 3472–3477, June 1995.
66. A. Yu. Kidiyarova-Shevchenko and D. Yu. Zinoviev, "RSFQ pseudo-random generator and its possible applications," *IEEE Trans. Appl. Supercond.*, Vol. 5, pp. 2820–2822, 1995.
67. H.B. Bakoglu, J.T. Walker, and J.D. Meindl, "A symmetric clock-distribution tree and optimized high-speed interconnections for reduced clock skew in ULSI and WSI circuits," *IEEE Int'l Conf. Computer Design*, pp. 118–122, Oct. 1986.
68. M. Shoji, "Elimination of process-dependent clock skew in CMOS VLSI," *J. Solid- State Circuits*, Vol. SC-21, pp. 875–880, 1986.
69. D.C. Keezer, "Design and verification of clock distribution in VLSI," *Proc. IEEE Int'l Conf. Commun. ICC'90*, Vol. 3, pp. 317.7.1–317.7.6, April 1990.
70. M.D. Dikaiakos and K. Steiglitz, "Comparison of tree and straight-line clocking for long systolic arrays," *J. VLSI Signal Processing*, Vol. 3, pp. 1177–1180, 1991.
71. "Hypres niobium process flow and design rules," Available from Hypres, Inc., 175 Clearbrook Road, Elmsford, NY 10523.
72. "TRW topological design rule for Josephson junction technology JJ-110A," available from TRW, One Space Park, Redondo Beach, CA 90278.
73. Z. Bao, M. Bhushan, S. Han, and J.E. Lukens, "Fabrication of high quality, deep-submicron Nb/AlO_x/Nb Josephson junctions using chemical mechanical polishing," *IEEE Trans. Appl. Supercond.*, Vol. 5, pp. 2731–2734, 1995.
74. K. Gaj, Q.P. Herr, and M.J. Feldman, "Parameter variations and synchronization of RSFQ circuits," *Applied Superconductivity 1995*, Institute of Physics Conf. Series #148, Bristol, UK, 1995, pp. 1733–1736.
75. A.D. Smith, S.L. Thomasson, and C. Dang, "Reproducibility of niobium junction critical currents: Statistical analysis and data," *IEEE Trans. Appl. Supercond.*, Vol. 3, pp. 2174–2177, 1993.
76. Q.P. Herr and M.J. Feldman, "Multiparameter optimization of RSFQ circuts using the method of inscribed hyperspheres," *IEEE Trans. Appl. Supercond.*, Vol. 5, pp. 3337–3340, June 1995.
77. L.A. Abelson, K. Daly, N. Martinez, and A.D. Smith, "LTS Josephson junction critical current uniformities for LSI applications," *IEEE Trans. Appl. Supercond.*, Vol. 5, pp. 2727–2730, 1995.
78. S.D. Kugelmass and K. Steiglitz, "An upper bound of expected clock skew in synchronous systems," *IEEE Trans. Comput.*, Vol. 39, pp. 1475–1477, 1990.
79. K. Gaj, C.-H. Cheah, E.G. Friedman, and M.J. Feldman, "Functional modeling of RSFQ circuits using Verilog HDL," *IEEE Trans. Appl. Supercond.*, Vol. 7, 1997.
80. P.-F. Yuh, "Shift registers and correlators using a two-phase single flux quantum pulse clock," *IEEE Trans. Appl. Supercond.*, Vol. 3, pp. 3009–3012, 1993.



Kris Gaj received the M.S. and Ph.D. degrees in Electrical Engineering from Warsaw University of Technology, Poland, in 1988 and 1992, respectively.

He has worked in computer-network security, computer arithmetic, testing of integrated circuits, and VLSI design automation. In 1991 he was a visiting scholar at the Simon Fraser University in Vancouver, Canada, where he worked on the analysis of various BIST (build-in-self-test) techniques for VLSI digital circuits. In 1992-93 he headed a research team at the Warsaw University of Technology developing an implementation of the Internet standard for secure electronic mail (Privacy Enhanced Mail), and software for secure Electronic Data Interchange per UNO standard UN-EDIFACT. He

was a founder of ENIGMA, a company that generates practical software and hardware applications from new cryptographic research.

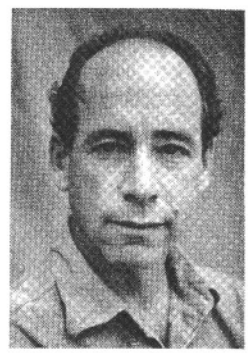
He has been with the Department of Electrical Engineering at the University of Rochester, Rochester, NY, since 1994, where he is a postdoctoral research fellow working on logic-level design and timing analysis of high-speed superconducting circuits. He currently teaches a graduate course on cryptology and computer-network security at the University of Rochester, and supervises student research projects on high-speed implementations of cryptography, VLSI circuit design and superconducting electronics.

He is the author of a book on code-breaking.

Eby G. Friedman was born in Jersey City, New Jersey in 1957. He received the B.S. degree from Lafayette College, Easton, PA. in 1979, and the M.S. and Ph.D. degrees from the University of California, Irvine, in 1981 and 1989, respectively, all in electrical engineering.

He was with Philips Gloeilampen Fabrieken, Eindhoven, The Netherlands, in 1978 where he worked on the design of bipolar differential amplifiers. From 1979 to 1991, he was with Hughes Aircraft Company, rising to the position of manager of the Signal Processing Design and Test Department, responsible for the design and test of high performance digital and analog IC's. He has been with the Department of Electrical Engineering at the University of Rochester, Rochester, NY, since 1991, where he is an Associate Professor and Director of the High Performance VLSI/IC Design and Analysis Laboratory. His current research and teaching interests are in high performance microelectronic design and analysis with application to high speed portable processors and low power wireless comunications.

He has authored two book chapters and many papers in the fields of high speed and low power CMOS design techniques, pipelining and retiming, and the theory and application of synchronous clock distribution networks, and has edited one book, *Clock Distribution Networks in VLSI Circuits and Systems* (IEEE Press, 1995). Dr. Friedman is a Senior Member of the IEEE, a Member of the editorial board of *Analog Integrated Circuits and Signal Processing*, Chair of the VLSI Systems and Applications CAS Technical Committee, Chair of the VLSI track for ISCAS '96 and '97, and a Member of the technical program committee of a number of conferences. He was a Member of the editorial board of the *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, Chair of the Electron Devices Chapter of the IEEE Rochester Section, and a recipient of the Howard Hughes Masters and Doctoral Fellowships, an NSF Research Initiation Award, an Outstanding IEEE Chapter Chairman Award, and a University of Rochester College of Engineering Teaching Excellence Award.
friedman@ee.rochester.edu



Marc J. Feldman received the Ph.D. degree in physics from the University of California at Berkeley in 1975. He worked at Chalmers University in Sweden and at the NASA/Goddard Institute for Space Studies in New York City in the development of superconducting receivers for radio astronomy observatories. He joined the faculty of Electrical Engineering at the University of Virginia in 1985, where he developed a variety of superconducting diodes for receiver applications.

He is now Senior Scientist and Professor of Electrical Engineering at the University of Rochester. Dr. Feldman's current research activities are directed towards the development of ultra-high-speed large-scale digital circuits using superconducting single-flux-quantum logic.