# Timing of Large RSFQ Digital Circuits

Kris Gaj, Eby G. Friedman, and Marc J.†Feldman
Department of Electrical Engineering, University of Rochester, Rochester, NY 14627

**Abstractã This paper analyzes 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 are presented and compared.**

## I. Introduction

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 [1]-[5]. Yet even for this medium scale complexity, the design of effective timing schemes in the multi-gigahertz frequency range is a challenging problem. The choice of clocking scheme for a particular RSFQ circuit is influenced by: (a) the topology of the circuit (e.g., 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.

Timing methodologies for semiconductor VLSI circuits have been well-established and systematized. One approach to superconductor circuit design is to rely on the application of such rules and techniques drawn from the semiconductor literature. More commonly, however, RSFQ clocking circuitry is developed specifically for RSFQ logic [1], [3], [4].

A central dilemma is the choice between synchronous and asynchronous clocking. Because of the high speed of RSFQ circuits, many have argued that some asynchronous scheme *must* be used, because synchronous clocking (using a single global clock signal) will certainly be inadequate for any technology providing such speeds. Nevertheless, synchronous clocking has been successfully used in almost all medium to large scale RSFQ circuits developed to date [4].

## II. Single-Phase Synchronous Clocking

Single-phase synchronous clocking is the form of clocking almost always used in semiconductor circuit design. Its primary advantages include high performance, design simplicity, small device and area overhead, and good testability.

Synchronous clocking works well for RSFQ circuits even at very high speed because RSFQ circuit designs always ignore the well-established clocking technique used in most semiconductor circuits. This is *equipotential zero-skew clocking*, in which every gate is clocked at the same instant, before the clock source generates its next output. 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. It is then natural to choose *pipelined clocking*, where many clock SFQ pulses travel through the clock path in parallel to the data path.

### *A. Synchronization of a Pair of Clocked Cells*

A schematic of *synchronous data path* connecting two RSFQ cells is shown in Fig. 1. An important parameter describing the data path is the *clock skew*. Clock skew (denoted $skew_{ij}$) is defined as the difference between the arrival time of the clock signal at the clock inputs of the cells at the beginning and at the end of data path ($t_{CLKi}$ and $t_{CLKj}$, respectively). The clock skew between cells *i* and *j* is

$$skew_{ij} = t_{CLKi} - t_{CLKj} \, . \tag{1}$$

By analyzing the conditions for the correct exchange of data between two sequentially adjacent cells in the presence of clock skew the minimum clock period in the circuit can be determined. The operating region of an RSFQ 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. 2.

The following conclusions can be drawn: (1) The minimum clock period is linearly dependent on the clock skew. (2) There exist values of clock skew for which the circuit does not work for any (even an extremely small) clock frequency. (3) 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.

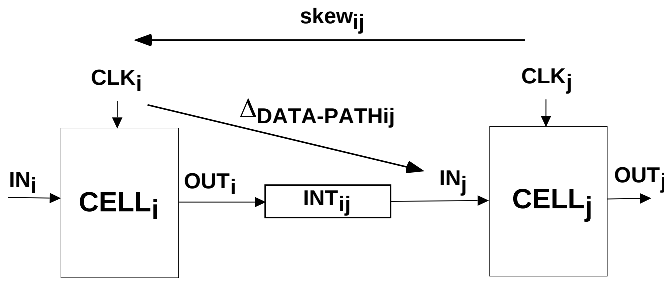

Fig. 1. Data path between two sequentially adjacent cells. Notation: $INT_{ij}$ - interconnection between cells i and j, $CELL_i$, $CELL_i$ - RSFQ cells i and j.

Financial support for this research was provided in part by the University Research Initiative at the University of Rochester, sponsored by the Army Research Office under Grant No. DAAL03-92-G-0112.

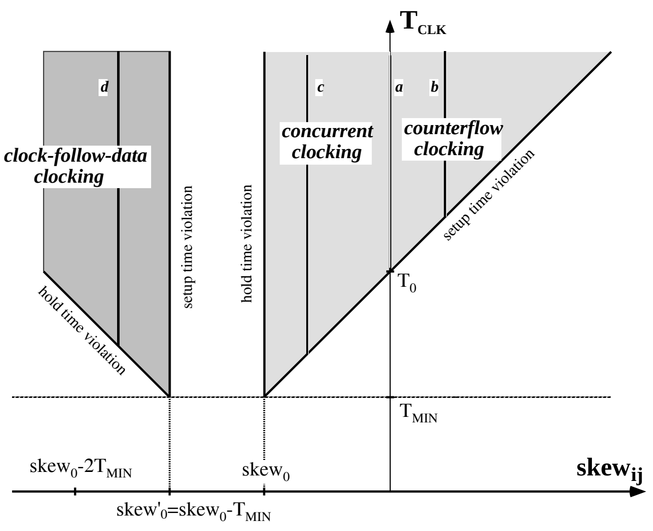

Fig. 2. Complete operating space of the data path between two sequentially adjacent cells as a function of the clock period and the clock skew. Lines a, b, c, d, correspond to the range of allowed clock frequencies for the circuit with the given clocking scheme. a - zero-skew clocking, b - counterflow clocking, c - concurrent clocking, d - clock-follow-data clocking.

## B. Counterflow and Concurrent Clocking

In *counterflow clocking*, the clock flows in the *opposite* direction to the data. Clock skew is positive. As shown by Fig. 2, a violation of the hold time is less likely than for zero-skew clocking. 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.

In *concurrent clocking*, the clock and the data flow in the *same* direction. The data released by the clock from the first cell travels simultaneously with the clock signal towards the second cell in the direction of the data path. The clock arrives at the second cell earlier than the data. The clock releases the result of the cell operation computed during the previous clock cycle, preparing the cell for the arrival of the new data.

This scheme should be used when the highest clock frequency is of primary concern. An aggressive application of this scheme will reduce the expected yield of the circuit, and so there must be 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 [3]. Concurrent clocking tends to require a larger number of Josephson transmission line (JTL) stages in the clock paths compared to counterflow clocking, and thus a greater overhead in circuit area and in layout complexity is expected.

## C. Clock-Follow-Data Clocking

If the magnitude of the clock skew (the delay in the clock path) is increased in a circuit with the concurrent clocking topology, a distinct clocking mode is possible. 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. In clock-follow-data clocking a *single* clock pulse carries the data through the whole array of N sequentially connected clocked cells in a time which is independent of the clock period. Clock-follow-data clocking can be optimized for the same minimum value of the clock period as concurrent clocking (see Fig. 2). It typically requires a larger number of JTL stages and has slightly larger worst case minimum clock period.

## D. The Effect of Local 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* topology offers an optimum solution. This topology is perfectly scaleable and works efficiently for an arbitrary number of cells in an array. Asymmetric MxN systolic arrays with a small value of M scale similarly to linear arrays.

For a two-dimensional symmetric square NxN arrays the effects of the local parameter variations in the paths of the clock distribution network cause additional clock skew, and as a result reduce circuit performance and yield. In all of the synchronous schemes, the performance of the circuit deteriorates 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 will become critical for different sizes of N. In particular, it is possible that these effects may be sufficiently small to realize practical sizes of RSFQ arrays, especially for counterflow clocking.

## III. OTHER SYNCHRONOUS SCHEMES

Two other synchronous clocking schemes have been developed. In *resynchronized clocking* developed specifically for RSFQ arrays [1], clock signals traveling along different paths in the clock distribution network are resynchronized using coincidence junctions. A *coincidence junction* produces an output pulse only after an input pulse has arrived at both of its inputs. Therefore, the clock skew between any two neighboring cells of a large array is substantially reduced. The disadvantages of the scheme include large area overhead and necessity to generate initially synchronized multiple clock signals at the input of the array.

*Two-phase clocking* is a novel clocking scheme adapted from semiconductor logic [5]. This scheme is expected to offer better performance than concurrent clocking, better tolerance to fabrication process variations than counterflow clocking, and an extremely simple design procedure. Its only drawback is the area overhead.

## IV. ASYNCHRONOUS TIMING

Asynchronous timing requires local signaling between adjacent cells. This signaling is naturally based on the concept of *events* such as *request* and *acknowledge*. In RSFQ logic, events are coded using SFQ pulses. Asynchronous logic elements that process SFQ pulses are simple and fast, and therefore asynchronous event driven schemes such as *dual-rail logic* or *micropipelines* appear to be easier and more natural to

TABLE I
FEATURES OF SYNCHRONOUS AND ASYNCHRONOUS TIMING SCHEMES

| timing scheme | sync/ async | speed | robustness | design procedure simplicity | area overhead | suitable for linear Nx1 arrays | suitable for symmetric NxN arrays |
|---|---|---|---|---|---|---|---|
| counterflow | S | − | + | + + | + | + | +/− |
| concurrent | S | + | − | − − | − | + | − |
| clock-follow-data | S | + | − | − − | − − | + | − |
| zero-skew | S | −/+ | − | + | −/+ | + | − |
| resynchronized | S | − | ++ | + | − − | + | + |
| two-phase | S | ++ | ++ | + | − − | + | + |
| dual-rail | A | +/− | + | − | − − | + | + |
| micropipelines | A | − | + | − | − − | + | +/− |

implement in RSFQ circuits than in semiconductor-based logic.

### *A. Dual-Rail Logic*

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. 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 as shown in Fig. 3a. 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. 3b. 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.

For a square NxN 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. However, disadvantages of the scheme include (a) a large device overhead, and (b) vulnerability to discrepancies between input rates at external inputs to the circuit.

### *B. Micropipelines*

*Micropipelines*, known from semiconductor circuit design, appear to be easily adaptable to RSFQ logic. The scheme is based on the use of coincidence junctions 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. 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.

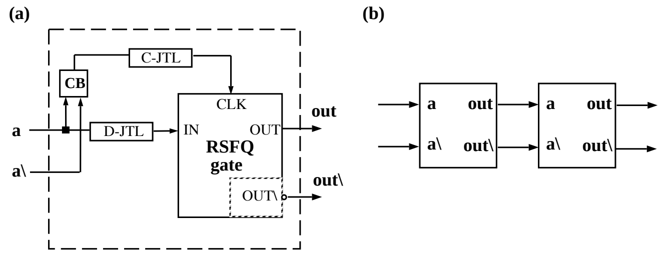

Fig. 3. Internal structure of a dual-rail cell based on one-input RSFQ gate (a), and the method of connecting dual-rail cells into a linear array (b). Notation: CB - confluence buffer, C-JTL - clock path JTL, D-JTL - data path JTL.

## V. SUMMARY

The primary features of all discussed in this paper timing schemes are summarized in Table I. For circuits which are roughly linear (pipeline width << length) the natural choices are the straight-line synchronous clocking schemes: counter-flow or concurrent. Counterflow clocking is preferable for an immature not-well characterized fabrication process with large timing parameter variations. If the performance obtained using this scheme is inadequate then a zero-skew or concurrent clocking should be considered.

For a square NxN arrays asynchronous schemes scale better with increasing N, as the timing signals are generated locally and do not need to travel through the long branches of the clock distribution network. However, resynchronized or double -phase clocking might be easier to design and can be even more robust. Two-phase clocking offers also the best *expected* maximum performance. Finally, hybrid synchronization schemes which use asynchronous strategies in tandem with simpler synchronous schemes are likely to be advantageous for large RSFQ circuits.

## REFERENCES

[1] O. A. Mukhanov, S. V. Rylov, V. K. Semenov, and S. V. Vyshenskii ≥RSFQ logic arithmetic,≤ *IEEE Trans. Magnetics*, vol. 25, 1989, pp. 857-860.
[2] 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, March 1991.
[3] 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, 1995, pp. 3320-3324.
[4] 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.*, pp. 27-30, Nagoya, Japan, Sept. 1995.
[5] K. Gaj, E. G. Friedman, and M. J. Feldman, ≥Timing of multi-gigahertz rapid single flux quantum digital circuits,≤ *Journal of VLSI Signal Processing,* vol. 9, 1997.