OPTIMISED ASYNCHRONOUS TIMING FOR SUPERCONDUCTIVE DIGITAL CIRCUITS

H.R. Gerber, C.J. Fourie and W.J. Perold

Superconducting Systems Laboratory, University of Stellenbosch, PrivateBagX1, Matieland, 7602, SouthAfrica (telephone: +27-21-808-4011; fax: +27-21-808-4981; e-mail: hrgerber@sun.ac.za).

Abstract: Rapid Single Flux Quantum (RSFQ) logic is a digital circuit technology that in recent years has presented itself as an alternative to semiconductors in the application of ultra high speed, very low power applications. The optimal timing of digital circuits operating at hundreds of Gigahertz is still a complex problem for both RSFQ and semiconductor technologies. The fact that most RSFQ gates require a clock signal to function makes this even more complex. Various RSFQ timing schemes have been adapted from semiconductor design methodologies, and some have been designed specifically for RSFQ. Currently, synchronous clocking schemes outperform other schemes, but with the scale of RSFQ circuits ever increasing, the proper use of timing schemes are becoming more crucial. This paper describes a new asynchronous self-timing scheme where the details of clock distribution and clocking are built into the logic gates. Tests were done on the newly developed asynchronous logic gates and a asynchronous full adder was implemented and tested.

Keywords: RSFQ, superconducting electronics, asynchronous logic, VLSI.

1. INTRODUCTION

RSFQ is the first superconducting digital technology that can use asynchronous timing schemes as well as synchronous timing schemes, and the possibility of constructing super-fast large asynchronous processors has therefore attracted significant interest. Since pipelining has become common in digital signal processors and because RSFQ logic naturally lends itself to pipelining, asynchronous RSFQ schemes have largely been overlooked. Furthermore, other well-known advantages of synchronous circuits exist, such as smaller area, improved testability, well established and easier design techniques [1].

This paper will propose a new timing scheme and design methodology that will provide optimal operating speed with minimal overhead. The timing scheme also attempts to make automated layout, placement and routing possible. Automation will greatly simplify the RSFQ design process. The lack of automation in the design of RSFQ circuits is one of the biggest restrictions on the progress that is being made in the field.

2. DIFFERENCES BETWEEN RSFQ AND SEMICONDUCTOR LOGIC

The most apparent difference is the use of Josephson junctions as active components in superconductor based circuits, compared to transistors that are used in semiconductor circuits. Fig 1 shows a schematic representation of the active components. Other differences include the use of pulse based logic rather than voltage state logic; the need for a clock signal for each logic gate and the fact that Josephson junction based circuits have a maximum fan-out of one. A special gate called a splitter is used to obtain a fan-out of two.

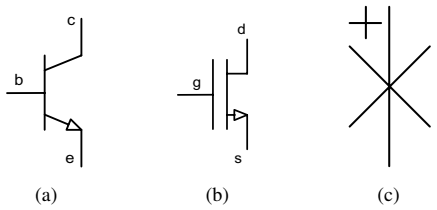


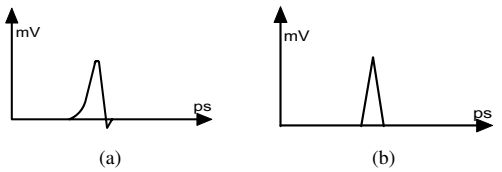
Figure 1: Active components for semiconductor and superconductor circuits (a) bipolar junction transistor, (b) MOSFET and (c) Josephson junction.

These quantised voltage pulses correspond to the transmission of a basic quantum of magnetic flux called a single flux quantum (SFQ). The area of an SFQ pulse is equal to [1,2]:

$$\int v(t)dt = \Phi_0 = h/2e = 2.07mV.ps \qquad (1)$$

Where h is Planck's constant and e is the electron charge unit. Fig. 2 shows an RSFQ pulse, and a simplified representation.

Most of the logic gate components in RSFQ are clock driven synchronous logic elements, whereas most



(a) (b)

Figure 2: RSFQ pulse state logic (a) pulse shape and (b) simplified pulse representation.

semiconductor logic gates do not require a clock signal. Fig. 3 shows the symbols of a selection of RSFQ logic gates [3].

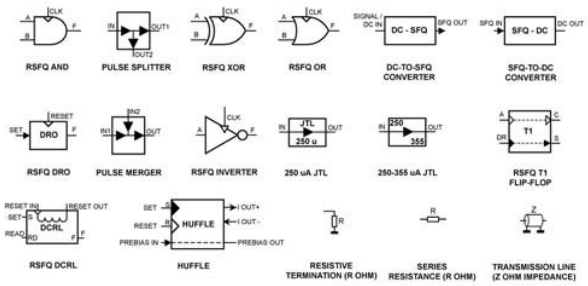


Figure 3: A Selection of RSFQ logic gate symbols

3. CLOCK DISTRIBUTION NETWORKS

Clock skew is an important parameter that describes the data path in both RSFQ and semiconductor based circuits [1] - [6].

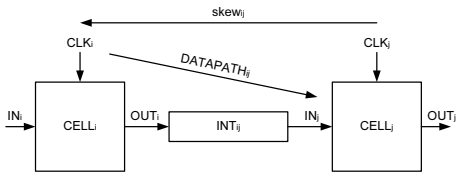


Figure 4: Clock skew between two sequentially adjacent cells i and j

Clock skew is defined as the difference between the arrival time of the clock signal at the clock inputs of the cells at the beginning and the end of the data path [1]. Fig. 4 gives a graphical representation of clock skew.

$$Skew_{ij} = t_{clk_i} - t_{clk_j} \tag{2}$$

The data path delay ($datapath$) 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 input of the second cell (t_{in_j}) [1]:

$$\Delta_{data-path_{ij}} = t_{in_j} - t_{clk_i} \tag{3}$$

From the above equations we can develop two inequality equations that fully describe the timing constraint of the data path for both pulse and voltage state logic:

$$Skew_{ij} + \Delta_{data-path_{ij}} >= hold_j \tag{4}$$

$$T_{clk} >= skew_{ij} + \Delta_{data} + setup \tag{5}$$

and

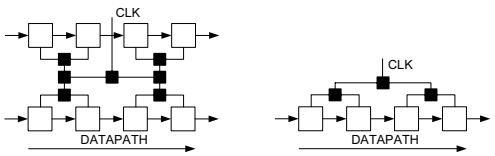
$$T_{min} = hold_j + setup \tag{6}$$

when

$$skew_{ij} = -datapath + hold_j \tag{7}$$

From the last equation it can be seen that cells should be optimised for minimum hold and setup times [1, 4]. It can also be seen that zero clock skew is not always required.

3.1 Binary and H-Tree networks



(a) (b)

Figure 5: Clock distribution scheme (a) H-tree and (b) Binary networks

A clock distribution network used to implement this clocking scheme consists of metal lines separated by large fan-out buffers, as shown in Fig. 5. 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 fan-out of the clock source. Normally the symmetry of the clock distribution network assures the simultaneous arrival of the clock signal to the input of all cells in the array. As the size of the array increases, the size of the distribution network increases exponentially [1]. At some stage, the overhead of the clock distribution network becomes overwhelming and can therefore not be used in large-scale circuits. The binary tree consists of a large number of splitters and Josephson Transmission lines (JTLs). The clock skew between the clock source and the clock signals that arrive at the cells can be considerable and this will create synchronisation problems if the cells are connected to other circuits.

3.2 Straight line clocking

An alternative to the binary tree structure is straight-line clocking [1, 4, 7]. In this scheme the clock distribution runs in parallel with the data path. There are two types of straight line clocking that can be implemented :

1. Concurrent clocking: the clock signals propagate in the same direction as the data.

2. Counterflow clocking: the clock signals propagate in the opposite direction relative to the data.

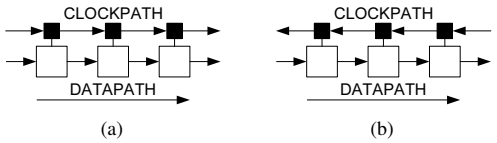


Figure 6: Straight line clocking (a) concurrent and (b) counterflow

Fig. 6 shows concurrent and counterflow clocking. In straight line clocking the magnitude of the clock skew is equal to the propagation delay of the clock signal between two adjacent cells. In concurrent clocking the clock skew is negative and in counterflow clocking it is positive. Straight line clocking has a small clock distribution overhead. Limitations on the operating speed of the circuit is only dependent on the internal speed of the slowest cell in the data path and not the clock distribution network. Optimal timing is however very dependent on the variation of the fabrication process. This leads to a suboptimal utilisation of the scheme. In general straight line clocked circuits can only be clocked at the maximum speed of the slowest component and this causes a lot of delay at the faster components.

4. CLOCKING SCHEMES

Fig. 7 shows where RSFQ asynchronous self-timing (RSFQ-AT) is positioned with respect to other timing schemes [8]. The figure also shows that RSFQ asynchronous self-timing is a combination of both Bundled and Delay insensitive clocking schemes.

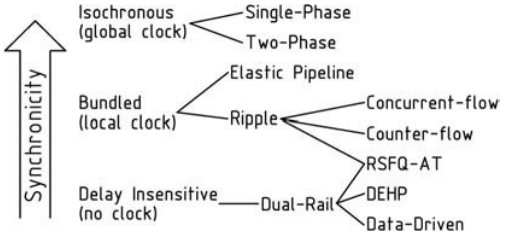


Figure 7: Shown is a taxonomy of RSFQ timing schemes; some are extinct [8]

4.1 *Equipotential Clocking*

The most popular clocking scheme used in semiconductor circuit design is single phase zero-skew equipotential clocking [1, 9]. Equipotential clocking assumes that a voltage state at the primary clock input does not change until the previous state has propagated thought the longest path of the clock distribution network. This limitation has previously been negligible, but with high speed large scale circuits the propagation delay plays a dominant role in the determination of the maximum operating speed [1, 5, 7].

4.2 *Pipelined Clocking*

Pipelined clocking is governed by a less restricted rule that states 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 [1, 7]. In a pipelined clocking scheme multiple clock pulses may travel simultaneously along the same distribution network. In a circuit where the clock distribution networks consist of interconnects that are separated by buffers, pipelined clocking is possible [1].

5. SELF-TIMED CLOCKING

The asynchronous self-timing clocking scheme implementation is similar to the straight line clock-follow-data scheme and is also similar to the dual-rail or data driven clocking scheme [1]. However, all timing requirements and clock distribution are built into the logic gates, and is therefore called self-timing. The incorporation of the clocking dynamics into the gates simplifies large scale design. Optimisation of the circuit can be done without changing the circuit. All the gates must simply be optimised for optimum timing conditions, ensuring that the hold and setup times are not violated. This will increase the layout size of each gate, but it will reduce the overall size of the circuit. It is relatively easy to shrink the size of a single gate, but it is far more difficult to shrink large and complex circuits. Once the gates are optimised and laid out as small as possible, it is a simple matter of cascading the gates to implement a given function.

6. SELF-TIMED CLOCKING IN DETAIL

In the asynchronous self-timed scheme, all SFQ data pulses that propagate through a circuit are accompanied by a clock pulse. The two pulses run in parallel transmission lines in order to reduce clock jitter and clock skew. The concurrent arrival of a clock signal and an SFQ data pulse indicates a logical '1'. The arrival of a clock pulse without any SFQ data pulse indicates a logical '0'. The SFQ pulse is allowed to precede the clock pulse and this will also indicate a logical '1'. If the SFQ data pulse follows the clock pulse, a logical '1' will only be indicated with the arrival of next clock pulse. This will generally cause logical errors. Fig. 8 shows the arrival of pulses at one input of a typical gate.
Ideally, a gate would be optimised so that the clock and data pulse arrives at exactly the same time. This critical timing will reduce delay time. If the clock delay is too large, the gate would have performed its functions and still be waiting for the clock pulse to arrive (positive delay). However this critical timing is dangerous, because any variations in pulse path lengths and fabrication process can cause the pulses to arrive in an incorrect order (negative delay). Negative

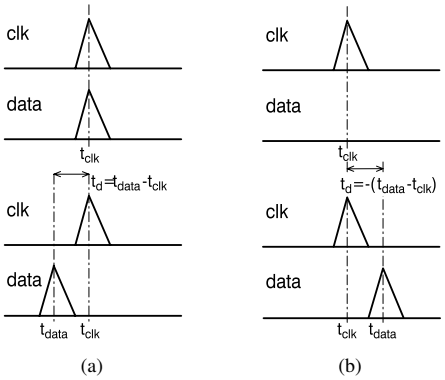


Figure 8: RSFQ-AT pulse representation (a) logical '1' and (b) logical '0'

clock delay will cause the gate to function incorrectly. Reducing the positive clock delay time ($t_{data} - t_{clk}$) for each gate, will therefore increase the maximum circuit operating speed.

7. TIMING GATE

The most important component of the self-timing design approach is the asynchronous Timing gate. The Timing gate applies the correct clock signals to the gate with which it is associated. Fig. 9 shows a block diagram of the construction of a typical two input, single output gate.

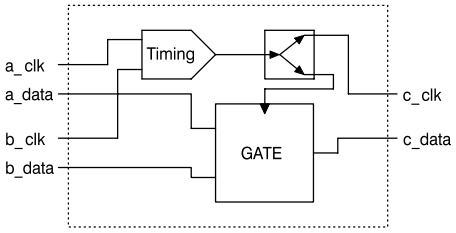


Figure 9: Proposed organisation for the design of asynchronous logic gate

The Timing gate waits for all the input clock signals to arrive, before applying a clock signal to the current gate. The Timing gate must also provide enough delay to the incoming clock pulse to ensure that the logic gate can complete its function before the clock signal is applied. Failing to provide enough delay will cause logical errors. In most cases the Timing gate is implemented using a standard RSFQ Coincidence Buffer [10]. However, if the order of the data inputs are known and fixed, the timing gate can be eliminated, by applying the last arriving clockpulse to the pulse splitter.

Another advantage of this clocking scheme is that it only requires one clock pulse to propagate the data though N levels of logic. Most of the other timing schemes require N-1 pulses (clock cycles). These timing schemes lend themselves to pipelining, but it is a complex task to keep these pipelines full and if they can not be kept full, a speed decrease of up to N-1 is experienced.

Because it might not be possible for data to propagate though N levels of logic in one clock cycle, some level of resynchronisation will be required to synchronise the asynchronous sections of the circuit with the synchronous (globally clocked) sections. A Destructive Readout Register (DRO) can be used to accomplish this resynchronisation. Fig. 10 shows the implementation. This will enable synchronous and asynchronous circuits to interface very effectively. This hybrid implementation allows for very high synchronised clock frequencies to be used, because complex circuits or logic with long data paths can be implemented using an asynchronous approach without affecting the rest of the circuitry.

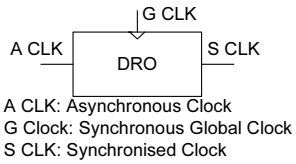


Figure 10: Resynchronisation by using a DRO

8. SFQ DRIVERS AND RECEIVERS

The use of the SFQ drivers and receivers at the input and output of the gates makes it possible to use passive transmission lines to propagate the SFQ pulses between gates [7, 11–13]. This reduces the need for JTLs to convey the SFQ pulses over longer distances. Using less JTLs also reduce clock jitter and delays, because each JTL adds a variable amount of delay.

In this design the SFQ drivers and receiver were designed to drive transmission lines with a characteristic impedance of 4.6Ω. This allows for relatively thin transmission lines that can save space when the circuit is laid out. Fig. 11 shows the SFQ driver circuit. The use of the SFQ drivers and receivers can therefore reduce the overall size of larger circuits. Another advantage of using SFQ drivers and receivers is that the automated layout of passive transmission lines is far simpler than when JTLs are used. All of this will greatly simplify the design process.

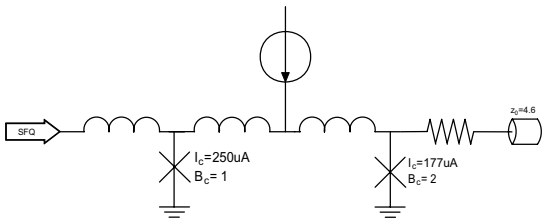


Figure 11: The SFQ driver circuit

Table 1: Adder Comparison

	RSFQ full-adder	RSFQ-AT full-adder
JJ count	101	108
T_c	125ps	62ps
F_{max} GHz	$<$24	$>$24

9. RESULTS

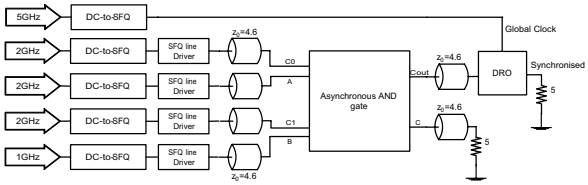


Figure 12: Asynchronous AND gate testbed

The asynchronous self-timing was first implemented in an asynchronous AND gate. Fig. 12 shows how the gate was tested and Fig. 13 shows the simulated results. Through simulation it was found that the gate fails for a negative delay time of more than $7ps$, therefore the only timing consideration when using the asynchronous AND gate is that $t_{data} - t_{clk} > -7ps$. It was also found that the clock and data outputs have a delay of $t_{data} - t_{clk} = -4ps$. Both of these delay times should be optimised to $0ps$. The gate was tested at 1GHz, 10GHz and 20GHz and functioned correctly. With the correct optimisation, the asynchronous AND gate should operate at frequencies of up to 50GHz. The asynchronous AND gate consisted of 32 junctions.

The SFQ driver and receiver circuits functioned correctly and the characteristic impedance of 4.6Ω allows for very thin transmission lines. This is very important for VLSI layouts.

The asynchronous OR and XOR gates were also implemented and tested and all functioned correctly.

The asynchronous self-timing scheme was tested by implementing a 1-bit full adder. Fig. 14 shows the implementation and Fig. 15 shows the simulated results, while Table 1 shows a comparison with a standard RSFQ implementation of the full-adder [14], [15]. With the correct optimisation the full adder should operate at frequencies of up to 20GHz.

The full adder was implemented as it would be in semiconductor technology. The standard RSFQ implementation required a complex clock distribution network, two clock pulses to complete and some delay components had to be included to compensate for a varying data path depth [15].

10. CONCLUSION

The asynchronous self-timing scheme handles both the complexity associated with the clock distribution network

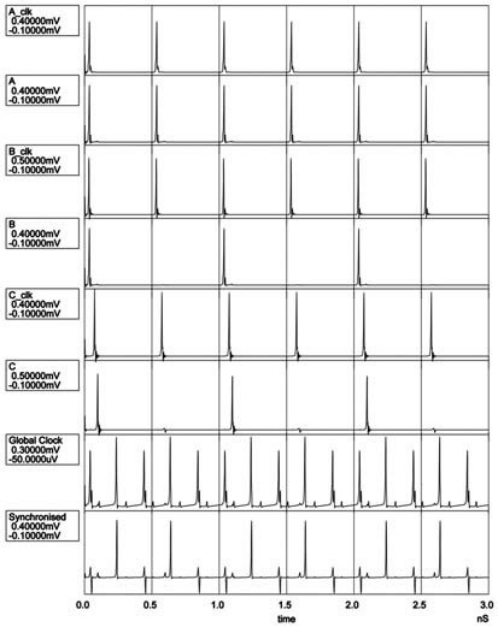


Figure 13: Simulated response of the asynchronous AND gate

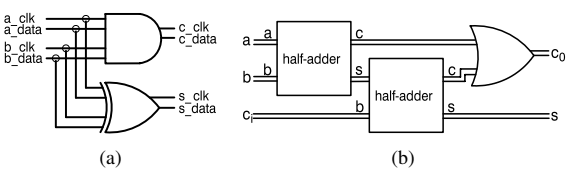


Figure 14: Asynchronous implementation of the (a) half-adder and (b) full-adder

and clocking scheme by incorporating them into the logic gate. This hides the complexities from the circuit designer and makes it possible to design RSFQ circuits with an approach similar to that used in semiconductor circuits. The scheme makes automated CAD tools, for the design and layout, easier to implement.

Circuits or sections of circuits that uses the asynchronous self-timing scheme can easily interface with other circuits that use other schemes, something that is not always possible with other timing schemes.

The asynchronous timing scheme makes it very easy to read pulse based timing diagrams, because the clock pulses indicate the instances when data is valid.

The incorporation of the SFQ drivers and receivers into the gates will greatly reduce the number of JTLs that are required in the signal distribution network. This in turn will reduce the clock jitter that is caused by fabrication variations. The use of passive transmission lines also makes automated placement and routing possible.

In the asynchronous self-timing scheme clock-skew has very little importance. Clock-skew is only considered when new gates are developed; once the gates are optimised, clock-skew can mostly be ignored. Clock-skew that is

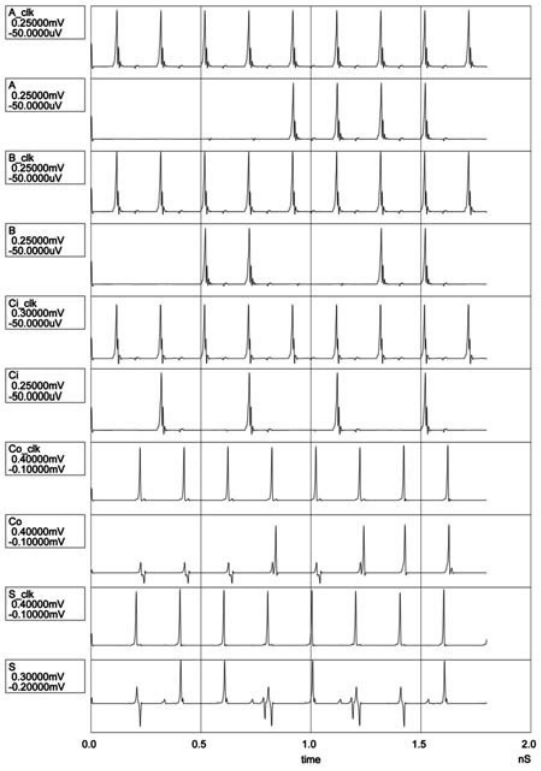


Figure 15: Simulated results of the asynchronous Full Adder gate

caused by the clock distribution network is cancelled out by the self-timing built into the gate. As demonstrated, resynchronisation can be achieved by making use of a DRO gate.

The asynchronous self-timing can be viewed as equipotential clocking that propagates the clock signals through the data path at maximum speed and causes an event when the end of the data path is reached. This event indicates that the circuit has completed its function and that it is ready to perform more functions. The output clock can be fed back to the input clock, and this closed loop clocking will produce circuits that operate at maximum speed, without the need of a global clock.

With careful design of the circuit logic, asynchronous self-timed pipelined clocking can be implemented. This will be the topic of further papers.

REFERENCES

[1] K.Gaj, E. Friedman and M. Feldman, *Timing of MultiGigahertz Rapid Single Flux Quantum Digital Circuits*, Journal of VLSI Signal Processing, vol. 9, 1997.

[2] M. Feldman, *Digital applications of Josephson Junctions*, Progress of Theoretical Physics, 1997.

[3] C. Fourie, *A toolkit for the design of superconducting programmable gate arrays*, PhD dissertation, Department of Electronic Engineering, University of Stellenbosch, South Africa, December 2003.

[4] K. Gaj, E. Friedman and M. Feldman, *A Clock Distribution Scheme for Large RSFQ Circuits*, IEEE Trans. Appl. Supercond., vol. 5, pp. 3320–3324, June 1995.

[5] J. Lin and V. Semenov, *Timing for RSFQ Digital Systems*, IEEE Trans. Appl. Supercond., vol. 5, pp. 3472–3477, September 1995.

[6] K. Gaj, Q. Herr and M. Feldman, *Analysis of Timing Requirements for Basic RSFQ Cells*, Draft version for IEEE Trans. Appl. Supercond.

[7] M. Dorojevets and P. Bunyk, *Architectural and Implementation Challenges in Designing HighPerformance RSFQ Processors*: A Flux1 Microprocessor and Beyond, IEEE Trans. Appl. Supercond., vol. 13, pp. 446–449, June 2003.

[8] Q.P. Herr and P. Bunyk, "Implementation and Application of FirstIn FirstOut Buffers," *IEEE Trans. Appl. Superconduct*., vol. 13, no. 2, pp. 563–566, June 2003.

[9] A. Davis and S. Nowick, *An Introduction to Asynchronous Circuit Design*, September 1997.

[10] SUNY RSFQ Cell Library, Available online at http://pavel.physics.sunysb.edu/RSFQ/Lib

[11] P. Bunyk, M. Leung, J. Spargo and M. Dorojevets, *FLUX1 RSFQ Microprocessor: Physical Design and Test Results*, IEEE Trans. Appl. Supercond., vol. 13, pp. 433–436, June 2003.

[12] M. Dorojevets, P. Bunyk and D. Zinoviev, *FLUX Chip: Design of a 20GHz 16–bit Ultrapipelined RSFQ Processor Prototype Based on 1.75μm LTS Technology*, IEEE Trans. Appl. Supercond., vol. 11, pp. 326–332, March 2001.

[13] Q. Herr, M. Wire and A. Smith, *Ballistic SFQ Signal Propagation OnChip and Chip–to–Chip*, IEEE Trans. Appl. Supercond., vol. 13, pp. 463–466, June 2003.

[14] H. R. Gerber, C. J. Fourie and W.J. Perold, "RSFQAsynchronous Timing (RSFQAT): a New Design Methodology for Implementation CAD Automation", *IEEE Trans. Appl. Superconduct.*, vol. 15, no. 2, pp 272–275, June 2005.

[15] C. Fourie, *A 10GHz Oversampling Delta Modulating Analogue–to–Digital Converter implemented with RSFQ Superconducting Digital Logic*, Masters dissertation, Department of Electronic Engineering, University of Stellenbosch, South Africa, December 2000.