Design of Multiple Fanout Clock Distribution Network for Rapid Single Flux Quantum Technology

Naveen Katam, Alireza Shafaei, and Massoud Pedram
Department of Electrical Engineering, University of Southern California, Los Angeles, CA 90089
{nkatam, shafaeib, pedram}@usc.edu

ABSTRACT

Rapid Single Flux Quantum (RSFQ) logic cells have traditionally been limited to driving one fanout cell only, because of the difficulty in distributing the single flux quantum pulse to multiple fanouts. However, this paper presents a method to modify the standard RSFQ cells at their interface in order to support multiple fanouts. For clock distribution, having multiple fanout drive capability is very important as the RSFQ logic is gate-level pipelined and requires clock for every logic operation. In general, the clock signal is split using splitter cells in order to provide it to different cells in the same clock stage. To support multiple fanouts without using splitter cells, the basic idea is to connect the output of one splitter to more than one gate, and compensate the input current reduction by increasing the bias current of Josephson junctions at the receiving end of the fanout. This helps simplify the clock routing process and reduces area usage, but it also tends to decrease circuit margins. However, we show that the yield is not compromised by our proposed technique and thus go on to present an algorithm for modifying the interface of RSFQ logic cells for this purpose.

1. INTRODUCTION

The demand for high computing performance and energy efficiency has been driving the development of the semiconductor technology for decades. Unfortunately, with increasing challenges to physical scaling of CMOS devices and the conclusive end of Moore's law in sight, there is a significant need to search for new device technologies and circuit fabrics that would allow continuation of performance and energy efficiency scaling beyond the end-of-scaling CMOS. In this context, *superconductive digital electronics* (SDE), especially *single flux quantum* (SFQ), has appeared as a very promising "beyond-CMOS" device technology with a verified speed of 370GHz [1] for simple digital circuits and switching energy per bit of ~10^{-19}J at T=4.2K (liquid helium temperature).

The first SFQ logic family, called *rapid SFQ* (RSFQ), was developed at Moscow State University [2]. This logic was DC powered. Newer DC-powered SFQ logic families such as the dual-rail SFQ [3], *energy-efficient RSFQ* (ERSFQ) [4], and *energy-efficient SFQ* (eSFQ) [5] have been proposed since then. In addition, a new class of AC-powered SFQ logic has emerged as exemplified by *self-clocked complementary logic* (SCCL) [6] and *reciprocal quantum logic* (RQL) [7]. Moreover, the SFQ technology appears to be an excellent choice for beyond CMOS as it is a low power solution with high speed. This is mainly because Josephson junctions (JJs), which are the basic circuit component in SFQ logic, switch quickly (~1ps) and dissipate very little switching energy (~10^{-19}J) [8] at low temperatures.

SFQ technology uses quantized voltage pulses in digital data generation, reproduction, amplification, memorization, and processing [9]. Although extraordinarily promising characteristics have been observed for SFQ logic, many technical challenges, including choice of circuit fabrics and architectures that can utilize SFQ technology as well as development of efficient simulation and design automation techniques and tools for SFQ logic must be undertaken in order for the SFQ logic to become a realistic option for realizing large-scale, high-performance, and energy-efficient computing systems of the future [8].

Although CAD tools for semiconductor technology are very mature, they cannot be directly applied to the design of SFQ circuits. Some of the key differences [10] are: (i) a different representation of logic values (bits), (ii) Different active components (i.e., transistor in CMOS vs. JJ in SFQ), (iii) different basic passive components (i.e., capacitors in CMOS vs. inductors in SFQ), and (iv) different passive and active interconnects (i.e., metal RC lines with buffers in CMOS vs. Josephson transmission lines and micro strip lines in SFQ). Furthermore, at logic and system levels, (v) different suite of basic gates, (vi) different clocking schemes, (vii) the influence of one gate on another connected gate and potential for back propagation of signals, (viii) rather high cost of the biasing network, (ix) high relative cost of interconnect compared to gates, and so on. Unfortunately, the current state of SDE CAD methodologies and tools is underdeveloped and by no means sufficient to build large-scale digital superconductive circuits.

Since SFQ is a pulse based logic, SFQ circuits cannot have a fanout of more than one. To take multiple fanouts from a node in a circuit, one has to split the concerned signal using *splitter* cells of required number. However, using splitter cells increases the delay of circuit and consumes area. RSFQ logic is deeply pipelined and requires clock signal for each logical operation. Thus, when there are more than one logic operations per clock stage, clock pulse has to be split using splitter cells. *A clock stage* contains all cells that need to be executed at the same time in a circuit.

In this paper, we present an approach to modify the standard cells without having to completely modify the logic cell to get more than one fanout from splitter output of clock distribution network. In Section 2, basic concepts of RSFQ logic are explained. In Section 3, clock distribution network and the idea of getting multiple fanout is discussed. Section 4 verifies the idea with simulation results of a test structure.

2. RSFQ LOGIC CONCEPTS

Transfer of SFQ pulses: A key element of SFQ circuits is the *Josephson transmission line* (JTL), which consists of several JJs that are DC-current biased with I_b such that $I_b < I_c$. These JJs are connected in parallel to one another by using inductors as shown in Figures 1(a) and 1(b). The inductance value L must be set to $L^* \sim \frac{\Phi_0}{I_c}$ because if L is much smaller than L^*, an SFQ pulse will be spread over several junctions. On the other hand, if L is much larger than L^*, flux quanta will be stored in a superconducting loop, comprising of two parallel-connected JJs and an intervening L and will not be transferred forward (see the SQUID description below). Now, an input pulse from input A triggers a 2π-leap in J1; next the resulted pulse developed across J1 triggers a 2π-leap in J2 through L2 and the process repeats until the pulse is transferred to the end

of JTL. There is a transfer delay as the pulse is transferred from one JJ to another. To transfer a pulse from one location to two destinations, the SFQ pulse *splitter* is used (cf. Fig. 1(c)). In a splitter, both branches are identical such that the current from input flows equally into both output branches, and the inductance values are chosen such that the Φ_0 is reproduced over appropriate time-width (cf. Fig. 1(d)). Unlike conventional CMOS, in SFQ logic, we must use splitters in order to fanout a source signal to different destinations.

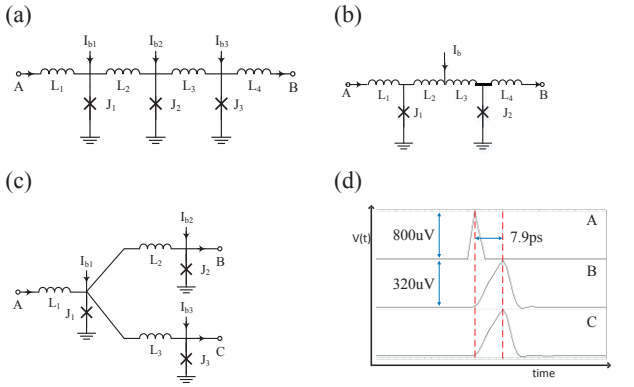

Figure 1: (a), (b) JTL circuits, (c) splitter circuit, and (d) simulation results.

Storing SFQ Pulses: DC SQUID (Superconducting Quantum Interference Device) with inductance L ($LI_c \sim 1.6\ \Phi_0$) [2] is used as a memory element to conserve SFQ pulses. SQUID is a quantizing SFQ loop (details in [11]) and is utilized in one of the fundamental cells of SFQ, D flip-flop (DFF) (cf. Fig. 2), which has two stable states, storing either zero (counterclockwise direction of i_L which is state "0") or one fluxon (clockwise direction of i_L. which is state "1"). State of the loop depends on the input from D. The stored pulse in the loop can be read by using a clock (Clk) signal. Depending upon the state of the loop, the arrival of Clk pulse causes either J2 to leap (if the state is "1") or J3 to leap (if the state is "0"). If J2 leaps, an output pulse will be generated losing the fluxon stored in the loop and resetting it to "0" state. On the other hand, if J3 leaps, no pulse will be generated.

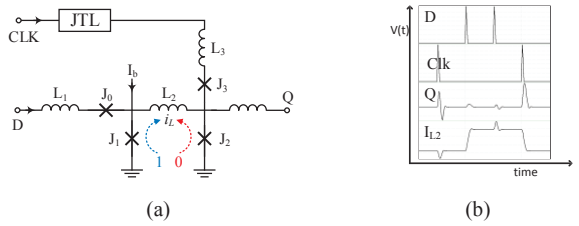

Figure 2: SFQ D flip-flop (J1, J2 and L2 form a SQUID, whereas J0 and J3 mediate pulse propagation), (b) simulation results.

RSFQ Convention of logic states: Signaling protocol of RSFQ is different from ordinary combinational logic because of two reasons [11]: (1) the "return to zero" nature of signals (SFQ pulses), and (2) the intrinsic memory of SFQ loops (SQUIDs) which are part of most RSFQ circuits. Any RSFQ logic gate can functionally be viewed as an implicit coupling of an asynchronous logic with a register (DFF), which works based on the standard protocol that is shown in Fig. 3. A typical RSFQ cell is fed by one or more inputs D_1 to D_n and the clock line. Each cell has typically two stable states and represents a finite state machine. Each clock pulse represents the boundary between consecutive cycles. An input pulse D_i can arrive any time during a clock period; arrival (or non-arrival) of pulse at D_i is treated as "1" (or "0") during that clock period. Inputs cause changes in the state of cell and are consequently conserved until the arrival of the Clk. When the Clk arrives, it resets cell into "0" state producing an output pulses (if the state is "1") at Q_{out}. This is equivalent to gate-level pipelining. As an example of RSFQ gates, OR gate and its simulation result are shown in Fig. 6.

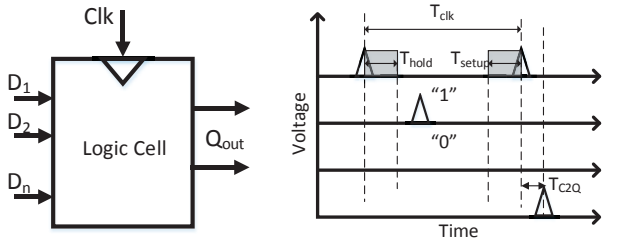

Figure 3. (a) A typical RSFQ cell, and (b) the standard timing protocol of RSFQ cells. The definition of setup time, hold time, clock-2-Q time, and clock period are specified.

3. CLOCK DISTRIBUTION NETWORK

Semiconductor circuits traditionally use equipotential clocking [12], which assumes that the worst-case propagation delay in the clock path is much smaller than the most critical data path delay in the circuit. In contrast, in RSFQ circuits, the propagation delay through the clock distribution network is comparable to the worst-case data path delay between two adjacent cells. The reason is that the clock distribution network is typically composed of splitters and JTLs whose delays are comparable to delays of logic cells. For this reason, flow clocking [13] is used which allows several consecutive clock pulses to travel simultaneously on the distribution network. The timing constraint with this clocking scheme is that before passing the data from one cell (sender) to a next cell (receiver), the receiver cell's calculation that had started as a result of the previous clock signal should have completed so that it is in the reset mode, and hence, ready to receive the next data signal. This ensures proper operation of an RSFQ circuit. Below, two widely-used clocking schemes in RSFQ are introduced:

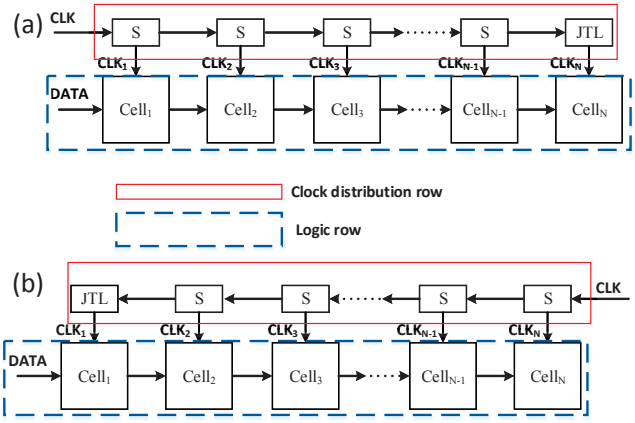

Figure 4: RSFQ clocking schemes: (a) concurrent-flow clocking (clock and data flow in the same direction), and (b) counter-flow clocking (clock and data flow in the opposite directions).

- *Counter-flow clocking:* the clock flows in the opposite direction of data (Fig. 4(b)). This is the most direct way to ensure that the receiving cell is reset before it receives the next data.
- *Con-current clocking*: clock flows in the same direction of the data (Fig. 4(a)).

Both clocking schemes need $N-1$ clock periods (or N clock pulses) to carry the data through N logic cells. The minimum clock period is the maximum of the delay of N cells. A combination of splitter and JTL components makes the clock distribution network. After generating two pulses using a splitter, one pulse is fed to the current cell and the other one is carried to drive other logic cells using an appropriate JTL segment. The split-and-drive operation is repeated until all RSFQ cells receive a clock signal. In the following diagrams, a red rectangle with Clk input is to be taken as a clock distribution row.

3.1 Problem Description

From the discussion on RSFQ cells and clocking schemes, it is evident that a desirable placement for the clock distribution network (a row of JTLs and splitters) is to put it above the row of logic cells, as indicated in the Fig. 4. Otherwise, the timing constraints are not satisfied, or more area is consumed for distributing the clock signal to these rows of logic cells (recall that a signal cannot be taken from one place to another by using a wire; instead JTL should be used to do this, which consumes area).

Since we cannot take multiple fanouts from a cell output, if there are more than one logic cell in a clock stage, we have to resort to either of the following two approaches to distribute the clock to all gates in a stage. (i) Split the clock pulse from the clock distribution network into the required number of fanouts (N) using $N-1$ splitters (cf. Fig. 5(a)). The drawback of this method is that the clock reaches different cells of the same stage at different times which may in turn cause difficulties from the timing perspective. (ii) Use different clock distribution rows for all cells in the same clock stage. The drawback is that the resource usage will be higher (cf. Fig. 5(b)). Suppose (in a circuit), one clock stage has six cells while the rest of the stages have only two cells. Even then, six clock distribution rows have to be run for all the logic cell rows which results in consuming more area than required. Both of these methods have issues. However, the first approach with careful timing seems to be better as it does not consume as much as area as the other one with multiple clock distribution rows. One way to make clock pulse reach all cells in minimum time is to reduce the number of splitter cells used in clock distributing network. We present a new approach to have fanout more than one by modifying the interface of standard cells. This new approach can be used for reducing the clock distribution delay by increasing fanout count of splitter.

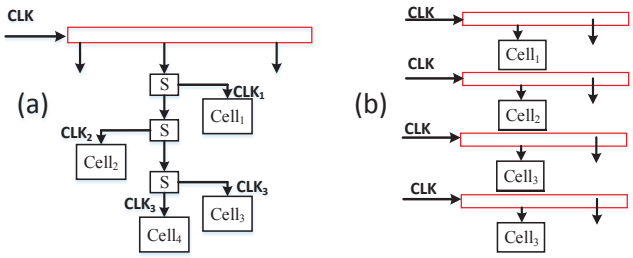

Figure 5: Distribution of clock to logic cells of same clock stage by (a) using splitters (b) using separate clock distribution row for each cell of stage

4. THE PROPOSED APPROACH

In general, it is difficult to connect to arbitrary RSFQ logic cells because of small input and output impedance of circuits. When two gates are connected together without re-optimizing their device parameters, current redistribution can occur between these gates which may in turn disturb the functionality of both gates [14]. A simple approach to solve this problem is to add a few JTL stages (one or two) to the *core of the gate* (part of the gate sufficient for performing the logic function fig. 6(a)) and re-optimize the gate to get its functionality. We call this added JTL stage an *interface*. In the following text, we will call the gate from which a SFQ pulse is going to fanout cells, the providing gate; and fanout cells will be called receiving cells.

In this work, we will try to modify just the *interface* to achieve multiple fanouts. In general, (case of single fanout), all the current from the incoming SFQ pulse goes to the single receiving cell, and it is enough to make the first junction at the interface to leap. Once the first junction of the cell leaps, the pulse propagates through the receiving cell according to its functionality. If there are multiple fanouts, current from the incoming SFQ pulse is shared among multiple fanout cells. Approximately, this current is equally distributed among all fanout cells due to the JTL interface at the entry of these cells which makes the SFQ pulse see equal impedance for all gates [15]. Due to this sharing, the amount of delivered current to the first junction of each fanout gate is significantly reduced. This reduction in current to the fanout gates can alter the functionality of gates and cause circuit failure.

To avoid this failure of a circuit in the case of multiple fanouts, the above said reduction in current has to be compensated. One way to provide extra current is to use exponential JTLs at the output interface of the providing gate. An exponential JTL [16] has critical currents of JJs and inductors in an increasing and decreasing order respectively in order to provide larger current as the output of SFQ pulse. However, the drawback of using an exponential JTL is that it should have at least three JJs and three inductors to amplify current. These extra elements in the path of SFQ pulse add more delay to the signal path apart from increasing the area of cell and power consumption. Hence, exponential JTL is not a feasible solution to generate extra current required to compensate the reduction in current due to sharing of multiple fanout cells. Moreover, we may have to optimize the (inductance and JJ critical current) parameters of core of standard cells. If device parameters of standard cells have to be changed on the fly, then the idea of standard cell design is violated.

A good way to provide extra current from providing cells side to receiving cells without needing to change the design of standard cells is to just change the parameters of the JTL interface of standard cells so that the core of the gate need not be changed. JTL interface has two JJs, one bias current, four inductors. In the following, we discuss a way to increase the fanout of cells just by changing two of these parameters (critical currents of JJ and bias current) of interface JTL.

The basic idea for achieving multiple fanouts is to provide the required extra current through the bias current for interface JTL JJs, so that the first junction of the interface can leap by the onset of incoming pulse. However, we cannot increase the bias current as indefinitely as the bias current always has to be below critical current of Josephson junctions it is biasing. In some cases, increasing of bias current does not help, we have to change the other parameters too (critical current and inductance values.). An algorithm for changing these parameters is given in Fig. 8. This approach will be explored in the context of clock distribution network where multiple cells can be connected to the splitter output. Though this approach can be tried and used for whole RSFQ logic, we believe that maximum benefit of this approach can be seen for clock distribution network. If this approach is used for general cases of RSFQ, we might have to lose advantage of standard-cell design (plug-and-paly) as we might have to look at

every case separately to use multiple fanout. However, in the case of clock distribution tree, there is no possibility of surprise cases and hence we can confidently use this approach. For clock distribution network, this approach can reduce the number of splitter cells used, thereby enabling the clock signal to reach all cells of a single clock stage at approximately the same time and reducing the delay. Though this approach seems to give better results, drawback is that it can give rise to lower yield of the circuit.

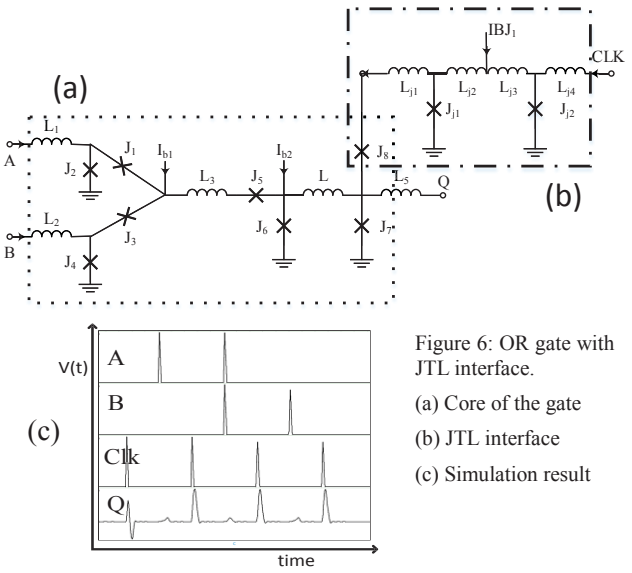

Figure 6: OR gate with JTL interface.
(a) Core of the gate
(b) JTL interface
(c) Simulation result

4.1 Margin and Yield Calculation

Margins of each component parameter of the cell are to be found first before we proceed to the yield calculation. Each RSFQ cell (circuit) consists of three basic device components: JJs, inductors, and resistors. However, resistor values change along with JJ critical current with respect to Stewart-McCumber parameter βc [17] which is same for all junctions in the circuit (βc=2 in our simulations). Another variable parameter is the bias current, which is an input to a JJ. So, the parameters are JJ critical currents, inductance values and bias current. *The margins of a parameter are upper and lower limits for which the circuit will operate, while all other parameters are held at their nominal values* [18]. Each of these parameters have lower and upper bound values, beyond which the circuit ceases to function as expected. For example, for a component value change of -40% to 60%, the gate function remains correct, then the margin with respect to this component value will be $[-40,60]$ and will be represented by a positive margin ($M_{p,i}$) and a negative margin ($-M_{n,i}$) denoting the variability range of that particular component. Example of margins shown in Fig. 7 for OR gate (only critical components are shown).

The question of whether a circuit can work is answered by ensuring that component values are within margins. However, a quantitative estimate of the probability that a circuit will work can only be known by the probability distribution of all of the circuit components. From [19] and [20], the statistics of fabricated JJ critical currents and inductance values of the MIT Lincoln Laboratory process technology (MIT-LL SFQ5ee) are obtained respectively, which follow normal probability distribution. Fab reports of inductor yield is quite high, which makes the parametric yield of inductors very high. Furthermore, we assume that the bias network can be controlled precisely. For these reasons, we have only considered the probability of JJ critical currents in the yield calculations.

For this purpose, success probability of JJ critical current, denoted by p_i, and defined as the probability that the JJ critical current is within its margins (e.g., $[-M_{n,i}, +M_{p,i}]$), is calculated as follows:

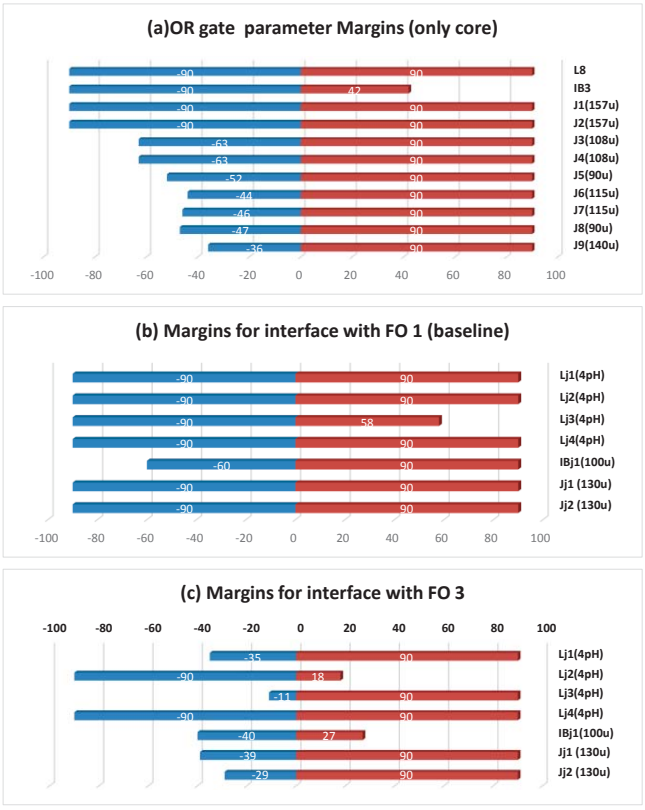

Figure 7: Margins (a) OR gate margins (*only core of the gate*) (b) interface margins with FO 1(baseline case) (c) interface margins with FO 3. Labels on the right side are to be matched with Fig. 6. The values in brackets are the nominal values of concerned device parameter

$$p_i = P(x_{1i} \le X \le x_{2i}) = \frac{1}{2}\left(\text{erf}\left(\frac{x_{2i}-\mu_i}{\sigma_i\sqrt{2}}\right) - \text{erf}\left(\frac{x_{1i}-\mu_i}{\sigma_i\sqrt{2}}\right)\right) \quad (4)$$

where, $x_{2i} = (1+\frac{M_{p,i}}{100})\mu_i$ and $x_{1i} = (1-\frac{M_{n,i}}{100})\mu_i$. By plugging x_{2i} and x_{1i} into Eq. (4), we will have:

$$p_i = \frac{1}{2}\left(\text{erf}\left(\frac{\frac{M_{p,i}}{100}*\mu_i}{\sigma_i\sqrt{2}}\right) + \text{erf}\left(\frac{\frac{M_{n,i}}{100}*\mu_i}{\sigma_i\sqrt{2}}\right)\right) \quad (5)$$

For a circuit with N JJ's, the probability that all JJs are within their margins can be called *yield* (Y), since it gives the probability of success of the circuit. For an N-junction circuit, yield can be calculated as

$$Y = p_1 * p_2 \ldots\ldots p_N = \prod_{i=1}^{N} p_i \quad (6)$$

The above margin calculation and subsequent yield calculation are mainly used for optimization of individual circuit parameters. Though the above yield calculation is still considered for parametric yield calculation of RSFQ circuits, it does not capture the interdependence among different circuit parameters. For this purpose, we have done Monte Carlo simulations [21] for obtaining the yield values after our modifications.

4.2 Algorithm

In this section, we present a general algorithm to modify the interface of any RSFQ cell to be used for multiple fanout. This algorithm takes a fully-functional RSFQ gate with a JTL interface

for single fanout as the input. Critical and bias currents mentioned in the flowchart of Fig. 8 are of the JJs which belong to JTL interface. We use the JTL shown in Fig. 1(b) in this section because it only has one bias current. When critical currents and bias currents are increased as said in algorithm, they should not go beyond their margins of existing functioning gate. All the simulations are done using JSIM tool [22]. The functioning of gate is determined by comparing the characteristics of different components with the known functioning gate. In case of 'No solution can be found', one has to use one of the methods shown in figure 5.

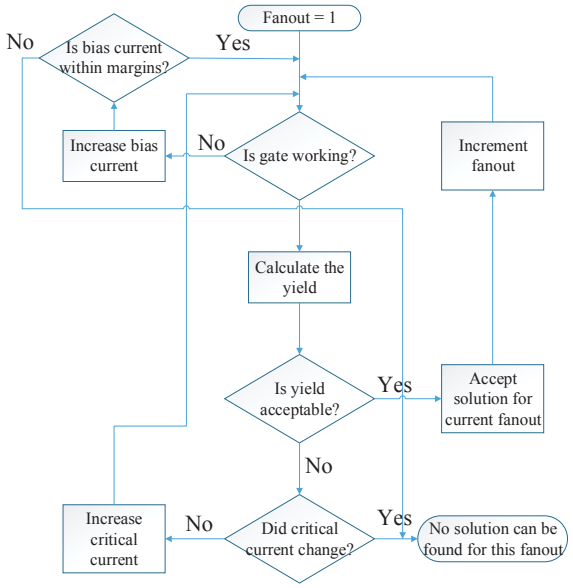

Figure 8: Flowchart for changing the JTL interface to use the cell for multiple fanout.

Here, we just show the optimization of the interface; however, one can optimize all the gate parameters along with modifying JTL interface to achieve a better yield. One can find available tools for circuit optimizer in [23]. We have not proceeded to optimize whole circuit so that the effectiveness of this basic idea can be illustrated. Margins of modified interface for fanout of three are shown in Fig. 7(c). One can observe the reduction in margins compared with the margins of interface of OR gate shown in Fig. 7(b). This reduction results in reduction in yield as per equations (4) and (5). Our algorithm finds whether a solution is acceptable or not based on this yield calculation. In general, we found that for increasing fanout, a lowest possible bias current for a fixed critical current of interface JJs results in higher yield.

5. RESULTS

We have tested the idea of multiple fanout for clock distribution network by increasing the fanout of a splitter output by connecting it to more than one OR gate. Steps mentioned in the above flowchart are followed to modify JTL interface of OR gate for increasing the fanout. Test structures used for presenting the results are shown in Fig. 9. All three test structures drive six OR gates, using splitters with fanout of 1 (FO1), fanout of 2 (FO2), and fanout of 3 (FO3). Results of optimized interface for different fanouts and their yields are given. Yield values calculated from margins as well as Monte Carlo simulation are also reported in this section.

The modified values of JTL interface parameters along with their yield values are given in Table 1. The value of bias current of interface JTL increases from FO1 to FO2 and from FO2 to FO3. As mentioned earlier, even JJ critical current has to be increased from FO2 to FO3. But the drawback is that the yield is reduced subsequently. However, we can see the advantages of multiple fanout from Table 2. Area of different cells are calculated as per the calculations given for resistively shunted (Josephson) junctions and inductors in [19]. Area for test structures comparison is also given in terms of number JJs and total inductance as actual area is technology dependent and parasitic values are added up. Delay values from initial clock input to first splitter to final splitter output in the path (C2FS) and initial clock input to first splitter to last OR gate output (C2FQ) are shown. Delay is reduced from 23ps to 10ps as fanout is increased from 1 to 3. There is an 18% reduction in area of test structure when FO3 is used. This reduction in delay, area and power comes from removal of splitter cells due to increase in fanout as described earlier.

Table 1: Modified JTL interface parameters with yield result

Fanout	Bias current (I_b)	Critical current of JJs (I_c)	OR yield	
			Margin based	Monte Carlo
1	100μA	130μA	0.999	0.999
2	150μA	130μA	0.999	0.972
3	250μA	150μA	0.981	0.865

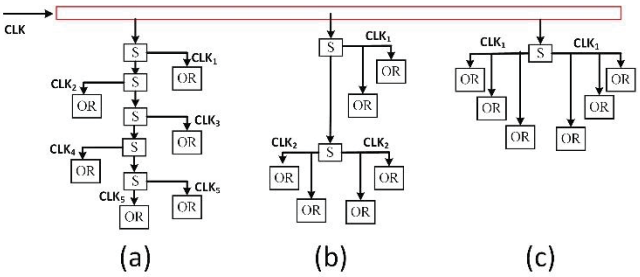

Figure 9: Test structures: all circuits drive six OR gates (a) baseline case: all splitter outputs have fanout of 1 (FO1) (b) all splitter outputs have fanout of 2 (FO 2) (c) all splitter outputs have fanout of 3 (FO 3).

Table 2: Comparison of results and circuit parameters

Test Structure		FO 1 (a) baseline	FO 2 (b)	FO 3 (c)
Delay (ps)	**C2FS**	23.1	11	10
	C2FQ	29	16	14
Static power (μW)		12.3	11.4	10.5
Area	**# of JJs**	81	75	69
	Inductance (pH)	422	390	358
	Approx. (μm²)	1112	1014	916

Table 3: Comparison of metrics to see cumulative advantage

Test Structure		FO 1 (a) baseline	FO 2 (b)	FO 3 (c)
$Yield/Delay$	C2FS	0.43	0.88	0.86
	C2FQ	0.34	0.60	0.61
$Yield/Power$		0.81	0.85	0.82
$Yield/Area$		0.89	0.95	0.94

The above parameters are used to define the following new metrics: $\frac{Yield}{Delay}$, $\frac{Yield}{Power}$, and $\frac{Yield}{Area}$, which are reported in Table 3. These metrics are used to see how far we can sacrifice yield for achieving better results in terms of delay, power, and area. In case, if the values of

these metrics are greater than baseline case, we can be sure that we are getting some benefit out of following the given approach of increasing fanout as we lose to decreasing of yield. Both FO2 and FO3 have cumulative advantage over FO1 with the proposed approach when compared against the drawback of yield in terms of delay, power and area. In case of FO2, the yield is same as single fanout and hence it is a big advantage to use it whenever possible. However, the yield of FO3 is considerably less, which makes it possible to use only for a few cases.

$\frac{Yield}{Delay}$, $\frac{Yield}{Power}$, and $\frac{Yield}{Area}$ metrics show that both FO2 and FO3 have an advantage over the baseline case. We have also tried to modify the interface of OR gate so that it can be used for fanout of 4. This made the margins of inductors of the *core of the gate* to reduce drastically. Hence, we did not present the case as it results in ~50% yield. This leads us to our future work to optimize the whole gate along with the interface so that the yield of gate can be increased with the increased fanout (this changes the design of the standard cells). Any available circuit optimizer software tools [24] [25] were not used in this present work for optimizing the circuit to get better margins (with and without JTL interface). It is possible to get very good yield comparable to single fanout gates if whole gate is optimized when trying to use it for multiple fanout. However, this will create multiple instances of OR gate in the standard cell library and one has to pick the suitable OR gate based on the fanout. The optimum fanout will be FO2 for the present case without changing the core of the gates. For a new SFQ library with different device parameters for core of the gates, optimum fanout may change.

6. CONCLUSION

Josephson junction RSFQ technology is very promising in the wake of high power dissipation of CMOS technology. It has challenges that are not seen in CMOS design like single fanout of RSFQ gates and hence, the design automation of RSFQ circuits has not been achieved. We have demonstrated a first step towards making standard cells that can be used for multiple fanout which can be used in standard cell based design. This approach not only reduces the area consumption, power consumption and delay in clock distribution network, but also helps in placement and routing of clock distribution network. We have introduced new metrics $\frac{Yield}{Delay}$, $\frac{Yield}{Power}$, and $\frac{Yield}{Area}$ to compare the results of single and multiple fanouts which capture both pros and cons of the approach presented in this paper thus giving the cumulative advantage.

Acknowledgement

This work was supported by the Safe and Secure Operations Office of the Intelligence Advanced Research Projects Activity (IARPA) under contract no. FA8750-15-C-0203-IARPA-BAA-14-03.

7. REFERENCES

[1] 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.

[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. Superconduct.*, vol. 1, 1991.

[3] S. Polonsky, "Delay insensitive SFQ circuits with zero static power dissipation," *IEEE Trans. Appl. Supercond.*, Vol. 9, June 1999.

[4] D. Kirichenko D, S. Sarwana S, and A. Kirichenko. "Zero static power dissipation biasing of SFQ circuits." *IEEE Trans. on Applied Supercond.*, 2011;21(3):776–779.

[5] M. Volkmann, A. Sahu, C. Fourie, and O. Mukhanov. "Implementation of energy efficient singleflux quantum digital circuits with sub-aJ/bit operation," *Superconductor Science and Technology*, 2013.

[6] A. H. Silver and Q. P. Herr, "A new concept for ultra-low power and ultra-high clock rate circuits," *IEEE Trans. Appl. Supercond.*, Vol. 11, pp. 333–336, June 2001.

[7] O. T. Oberg, Q. P. Herr, A. G. Ioannidis, and A. Y. Herr, "Integrated power divider for superconducting digital circuits," *IEEE Trans. Appl. Supercond.*, Vol. 21, pp. 571–574, June 2011.

[8] D. S. Holmes, A. L. Ripple, and M. A. Manheimer, "Energy-Efficient Superconducting Computing —Power Budgets and Requirements," *IEEE Trans. Appl. Supercond.*, Vol. 23, No. 3, 2013.

[9] K. Likharev. "Superconductor digital electronics," *Physica C*. 2012;482: 6–18.

[10] K. Gaj, Q. P. Herr, V. Adler, A. Krasniewski, E. G. Friedman, and M. J. Feldman, "Tools for the computer-aided design of multi-gigahertz superconducting digital circuits," *IEEE Trans. Appl. Supercond.*, vol. 9, no. 1, pp. 18–38, Mar. 1999.

[11] P. Bunyk, K. Likharev, D. Zinoviev, "RSFQ technology: Physics and devices", Int. J. High Speed Electron. Syst., vol. 11, Mar. 2001.

[12] C.Mead and L.Conway, Introduction to VLSI systems, Addison-Wesley, Reading, MA, 1980.

[13] Mukhanov, D. Bradley et al. "Design and Operation of RSFQ circuits for Digital Signal Processing," *5th Int. Supercond. Electron. Conf.*, Nagoya, Japan, Sept 1995, pp 27-30.

[14] K. Gaj et al. "Toward a Systematic Design Methodology for Large Multigigahertz Rapid Single Flux Quantum Circuits," *IEEE Trans. Appl. Supercond*., Vol.9, No.3, Sep. 1999, pp. 4591-4606.

[15] B. Dimov, V. Todorov, V. Mladenov, F. H. Uhlmann, "The Josephson Transmission Line as an Impedance Matching Circuit," *WSEAS Trans. Circuits and Systems*, Vol.3, No.5, pp. 1341-1346, 2004.

[16] O. A. Mikhanov, V. K. Semenov and K. K. Likharev, "Ultimate Performance of RSFQ Logic Circuits," *IEEE Trans. Magn.*, Vol. 23, No. 2, pp. 759-762, March 1987.

[17] T. Van Duzer, and C. W. Turner, *Principle of Superconducting Circuits*. New York: Elsevier, 1981.

[18] 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.

[19] S. K. Tolpygo, "Superconductor digital electronics: scalability and energy efficiency issues," *Low Temp. Phys.*, vol. 42, 2016.

[20] S. K. Tolpygo et al., "Inductance of circuit structures for MIT LL superconductor electronics fabrication process with 8 niobium layers," *IEEE Trans. Appl. Supercond.*, Vol.25, No. 3, June 2015.

[21] C. J. Fourie, W. J. Perold, and H. R. Gerber, "Complete Monte Carlo model description of lumped-element RSFQ logic circuits", *IEEE Trans. Appl. Supercond.*, vol. 15, pp. 300–303, 2005.

[22] E. S. Fang and T. Van Duzer, "A Josephson integrated circuit simulator (JSIM) for superconductive electronic applications," in *Ext. Abstracts 4th Int. Supercond. Electron. Conf. (ISEC)*, Tokyo, Japan, 1989, pp. 407–410.

[23] C. J. Fourie and M. H. Volkmann, "Status of superconductor electronic circuit design software," *IEEE Trans. Appl. Supercond.*, vol. 23, no. 3, p. 1 300 205, Jun. 2013.

[24] Q. P. Herr and M. J. Feldman, "Multiparameter optimization of RSFQ circuits using the method of inscribed hyperspheres," *IEEE Trans. Appl. Supercond.*, Vol. 5, pp. 3337–3340, 1995.

[25] S. Polonsky, P. Shevchenko, A. Kirichenko, D. Zinoviev, and A. Rylyakov, "PSCAN'96: New software for simulation and optimization of complex RSFQ circuits," *IEEE Trans. Appl. Supercond.*, Vol. 7, pp. 2685–2689, 1997.