Accurate Margin Calculation for Single Flux Quantum Logic Cells

Soheil Nazar Shahsavani, Bo Zhang, and Massoud Pedram
University of Southern California
{nazarsha, zhan254, pedram}@usc.edu

Abstract—This paper presents a novel method for accurate margin calculation of single flux quantum (SFQ) logic cells in a superconducting electronic circuit. The proposed method can be utilized as a figure of merit to estimate the robustness of a logic cell without the need for expensive Monte-Carlo simulations. This is achieved through efficient state-space exploration of all parameters in the cell structure. Using the proposed approach, distinct parameter dispersion (DPD) based yield of SFQ cells increases by 55% on average, compared with state-of-the-art techniques.

I. INTRODUCTION

Development of the semiconductor technology has been driven by the demand for higher performance and energy efficient computing for decades. Although conventional CMOS-based technology has been able to provide high performance energy computing fast enough to keep up with the demand, it has encountered serious challenges such as high temperature and power consumption [1]. Hence, there is a significant demand to search for new innovations that would permit continuation of performance and energy efficiency scaling to well beyond the end-of-scaling CMOS nodes.

Superconducting electronics (SCE), including rapid single flux quantum (RSFQ) logic family, appears to be one of the most promising technologies to replace CMOS devices, primarily due to fast switching and low energy consumption. This is as a result of special attributes of Josephson junctions (JJs), basic circuit elements in SFQ logic, such as fast switching (∼1 ps) and low switching energy per bit of (∼10⁻¹⁹ J) at low temperatures [2]. In particular, RSFQ technology uses quantized voltage pulses in digital data generation, reproduction, amplification, memorization, and processing [3]. Furthermore, it has been demonstrated that RSFQ circuits are functional at operating frequencies of up to 770 GHz [4].

In spite of many extraordinary characteristics of SFQ logic, such as low energy dissipation and high performance computation, architectures, design automation methodologies, and device fabrication require solutions 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 [2]. The increase in integration density of modern superconducting circuit processes allows the design of increasingly complex circuits. However, design challenges of SFQ logic and tool development has not received much attention in the past decades.

RSFQ gates are implemented using two-terminal Josephson junctions. This characteristic decreases the input and output impedance, increases the sensitivity of the circuit to variations of the component parameters and consequently complicates the design of a single gate [5]. Fabrication process is also significantly different from conventional CMOS process, and process variations are larger in current fabrication processes [5]. Additionally, cell variants are required in cell libraries to allow automated place-and-route, with support for different layouts depending on the configuration of inputs/outputs. Cell variants are used based on trade-offs between propagation speed, operating margins and bias current [6][7].

To develop reliable cell structures prone to process variations, accurate methods for reliability evaluation are necessary. Critical margin calculation and yield analysis using Monte-Carlo (MC) based simulations have been extensively used to estimate the robustness of a logic cell [5][8].

Yield of a cell is defined as the ratio of number of cells operating correctly to the total number of cells. Assuming each parameter in the cell structure to have a normal distribution with specified mean and standard deviation, yield can be quantified using MC simulations. Although calculated yield may be a good indicator of cell reliability, it is computationally expensive. Furthermore, using MC based yield estimation throughout the cell optimization process is inefficient. This is primarily due to the fact that a large number of simulations should be performed at each iteration of the optimization process and after each change in the component parameters. Therefore, it may only be used at the final stage of optimization process to quantify the robustness of the optimized cells.

A simpler method of robustness evaluation called Critical Margin calculation has been widely used in the literature [9][5][8]. In this method, a binary search over a predefined range of values for each parameter is performed while all other parameters are fixed at their nominal values. Every time a parameter is changed, cell is tested. Upper and lower bound values for each parameter calculated in this manner are denoted as parameter margins. Critical margin is calculated as the smallest margin among all parameters in the cell. However, as this method only considers alteration of each parameter while other parameters are fixed at their nominal values, it can not capture the dependence of parameters on each other. Hence, critical margin fails to evaluate the robustness of a cell accurately [8].

The focus of this paper is to propose a new margin calculation method for SFQ cells with large number of parameters. The primary goal of this method is to calculate a set of margins for which the cell yield will be nearly one if all parameters lie within specified margins. To the best of our knowledge, no previous work has been done on accurate margin calculation.
of SFQ cells. The key contributions of this paper can be summarized as follows.

- We present a novel margin calculation algorithm to estimate a set of margins for all parameters in a logic cell such that if all parameters are spread inside these margins, yield is nearly one.
- The proposed algorithm works irrespective of the cell structure and topology.
- The proposed algorithm can be used to evaluate the robustness of a cell throughout the optimization process efficiently, using small number of simulations.

The rest of the paper is organized as follows. Theoretical background are discussed in section II. Proposed margin calculation methodology is presented in Section III. Simulation results are reported in Section IV. Finally, the paper is concluded in Section V.

II. THEORETICAL BACKGROUND

A. SFQ Technology

Information in SFQ technology is represented by short picosecond voltage pulses of quantized area rather than dc voltage (as in CMOS). These SFQ pulses can be generated, reproduced, memorized and amplified using elementary components called Josephson junctions (JJ) [9]. The area of the produced pulses can be calculated as follows.

\[ \int V(t) dt = \Phi_0 \approx \frac{h}{2e} \approx 2.07 \text{mV}.ps \]

(1)

where \( \Phi_0 \) represents a single quantum of superconducting flux, \( V(t) \) denotes voltage across junction, \( h \) is the Planck constant and \( e \) is the electron charge. Current-phase and voltage-phase relation in JJs can be quantified as follows.

\[ \frac{\partial \phi}{\partial t} = \frac{2\pi}{\Phi_0} V(t) \]

(2)

\[ J_s(\phi) = J_c \sin(\phi) \]

(3)

where \( \phi \) is the phase of the junction and \( J_c \) denotes the critical current density. If current density through a JJ is greater than \( J_c \), a voltage pulse \( V(t) \) is formed across JJ. This causes the JJ to exit the superconducting state and enter the normal state. Once JJ returns to the superconducting state, junction goes through a 2\( \pi \)-leap. For a detailed explanation of magnetic flux quantization see [9]. Fig. 1(a) illustrates the schematic view of a splitter cell. It consists of several JJs and inductors. Additionally, JJs are biased by DC-currents, denoted by \( I \). An input pulse triggers a 2\( \pi \)-leap in JJ. Consequently, a pulse is generated which flows through J2 and J3 which in turn generates pulses at outputs B and C (cf. Fig. 1(b)). Both branches of the splitter cell are identical. Hence, current flows equally through both branches. Furthermore, inductance values should be chosen such that \( \Phi_0 \) is reproduced over appropriate time-width window. Operation of splitter cell is shown in Fig. 1(b). Typical cells in SFQ design have a single fan-out. Therefore, to propagate a pulse to multiple cells, splitters should be used. Furthermore, generating multi-input cell structures is not trivial. Input/output interfaces should be modified to be able to support multiple inputs and outputs [7].

\[ \text{margin is defined as set of parameters for which cell functions correctly (i.e., as specified by the HDL model). Likewise, a failing set refers to set of parameters for which cell does not function correctly.} \]

C. Line Search

This is the most popular technique for margin calculation [5]. Margin for each parameter is calculated while others are fixed at their nominal values. Given a stopping criterion and an initial range for each parameter, a binary search is performed to calculate the upper bound and lower bound values for each parameter such that cell functions correctly [10].

Circuit diagram for an AND2 (2-input AND gate) cell is shown in Fig. 2. It has 20 parameters, including different inductance values \( (L1-L8) \), junctions \( (J1-J9) \), and biasing currents \( (I1-I3) \). Calculated margins for all parameters in AND2 cell using Line Search (LS) method is depicted in Fig. 3. As it can be observed, values for parameter \( J8 \) can deviate from -72% to +25% of its nominal value while other parameters are fixed. Binary search for each value is performed using an initial bound of \( \pm 100\% \). As search range drops below the target accuracy search is terminated. Critical margin is defined as the smallest margin among all parameters. In the case of AND2 gate, the critical margin is equal to 25%. Although critical margin is relatively easy to calculate, it is not
In this paper, MC simulations are performed to evaluate the yield of a cell design and an HDL model describing the correct operation of the cell. Details of both approaches are explained in the following subsections.

### A. Individual Parameter Update (IPU)

In this algorithm, a set of simulations are performed to calculate the margin for each parameter. Pseudo code for IPU algorithm is given in Algorithm 1. The process starts by creating variables ($X_j$) representing the value of each component in the cell design. All variables are initialized to their corresponding nominal value (line 0). In each iteration, upper margin for one parameter ($U_j$) is updated using LS method, while other parameters are fixed (line 3). This is achieved by UpdateByLS function which calculates the upper bound for each parameter given input parameters. Next, the mean value for $X_j$, namely $M^j$, is calculated as the average of upper bound value ($U_j$) and previous value ($X_j^{i-1}$) of the corresponding parameter (line 4). A simulation is performed

### D. Yield Analysis

The yield of a cell can be quantified as the number of cells working correctly divided by the total number of cells. To estimate the yield reliably, each parameter should have a statistical distribution specification and a large number of cell simulations should be performed.

One method for relating the margin and yield concepts can be defined as follows. Assume all parameters to have independent normal distributions around their nominal values and standard deviations ($\sigma$) proportional to their corresponding margins. Calculating the yield using these parameter variations can be used to evaluate the accuracy of margin calculation techniques [8].

In this paper, MC simulations are performed to evaluate the accuracy of calculated margins similar to [8]. It is assumed that all parameters have independent normal distribution and each parameter has a mean value equal to its corresponding nominal value. Additionally, each parameter can have two different standard deviation (SD) values. 1) identical parameter dispersion (IPD) refers to the calculation in which all parameters have the same standard deviation. In this analysis, $3\sigma$ value is set to the critical margin of the cell. This method indicates robustness of a cell if all parameters have same variation while parameter spread is limited to smallest margin among all parameters. 2) distinct parameter dispersion (DPD) analysis assumes each parameter has a $3\sigma$ value equal to its own margin. If lower and upper bound margins for each parameter are different, minimum value is chosen as the $3\sigma$ value. In this manner, the probability that MC samples lie within the calculated margins for each parameter is more than 99%. This method is more realistic than IPD as different parameters (e.g., inductors or Josephson junctions) may have different fabrication processes and hence distinct parameter spreads.

For the case of AND2 gate and LS margin calculation method, IPD and DPD based yield values are equal to 92% and 45%, respectively. As it can be observed, DPD based yield is low. This indicates that although the critical margin value calculated by this method may be accurate, reported margin for each parameter is highly optimistic, as a direct consequence of ignoring the dependence of different parameters on each other.

### III. PROPOSED METHOD

Based on the proposed margin and yield definitions, the idea is to calculate a set of margins for all parameters in the design, such that calculated yield is one if all parameters lie within these margins. Our proposed solution is more accurate than LS method as it considers the dependence of variables on each other. In other words, the goal is to calculate a set of margins for which IPD and DPD yields are near one. Proposed method can be directly used as a figure of merit to indicate the robustness of a cell design.

Two approaches are introduced for accurate margin calculation. 1) individual parameter update (IPU) and 2) simultaneous parameter update (SPU).

Inputs to both algorithms are the parameters in the cell design and a HDL model describing the correct operation of the cell. Details of both approaches are explained in the following subsections.
while variable \( j \) is changed to \( M_{X_i} \) and other parameters are fixed. If the simulation for this set of parameters fails, \( M_{X_i} \) value is updated until a passing set is achieved (lines 5-9). Once a mean value \( M_{X_i} \) is found such that cell functions correctly for given set of variables, \( X_j^i \) value is updated to its corresponding mean value \( M_{X_j} \) (line 10). The algorithm continues by calculating the upper bound for next parameter while some of the parameters are updated to their mean values \((X^k_i, \forall k = 1...j-1)\) and others are fixed at their previous values \((X^t_j, \forall t = j...p)\) (cf., Alg. 1 line 3).

Parameter update and cell simulation continues until stopping criteria is satisfied (lines 12-14), that is, once the absolute difference between previous value \((X_j^{i-1})\) and current value \((X_j^i)\) for all parameters is less than a predefined threshold value. Output of IPU algorithm is a set of upper bound margins for all parameters. Lower bound margins can be calculated by replacing lower bound values \((L_j^i)\) by upper bounds in the Alg. 1. If there are \( n \) parameters in the cell design, possible set of values for all parameters forms an \( n \)-dimensional space. Cartesian product of calculated margins defines a hyper-rectangle in \( n \)-dimensional space. If all parameters lie within the given hyper-rectangle, the cell functions as specified in the HDL model.

Assuming a cell contains only three parameters \((X, Y, \) and \( Z)\), IPU algorithm works as follows. First, upper bound for \( X \) is calculated while \( Y = Y^0 \) and \( Z = Z^0 \). Mean value for \( X \) is calculated as average of \( X^0 \) and \( U_{X_k}^0 \). If the cell works with the set of parameters \((X = M_{X_k}^1, Y^0, Z^0)\) \( X^1 \) is updated to \( M_{X_k}^1 \). Otherwise, it continues updating \( M_{X_k}^1 \) until a passing set of parameters is found. Next, the upper bound for \( Y \) is calculated while \( X = X^1 \) and \( Z = Z^0 \). Similarly, \( Y \) is updated to its mean value \( Y = M_{X_j}^1 \) (lines 5-9). Finally, upper bound for \( Z \) is calculated considering \( X = X^1 \) and \( Y = Y^1 \). The algorithm continues updating parameters \( X, Y, \) and \( Z \) until none of them can be updated anymore, or the difference of current and previous values for all parameters drop below a predefined threshold value.

B. Simultaneous Parameter Update (SPU)

Psuedo code for this algorithm is given in Algorithm 2. This is different than the Alg. 1 as it updates all parameters to their corresponding mean value simultaneously rather than individually.

Initially, all parameters are set to their nominal values and LS is performed to calculate the upper bound for each parameter. At each iteration \( i \), mean value \((M_{X_j})\) for all parameters is calculated as the average of their upper bound \((U_{X_j}^{i-1})\) and previous value \((X_j^{i-1})\) (cf., Alg. 2 lines 2-4). If the cell does not function correctly given these mean values, all mean values are updated until a set of passing points is found. This is achieved through averaging mean value and previous value for each parameter (lines 5-9). Once a passing set is found, all parameters are updated to their mean values (lines 10-12). Additionally, upper bounds for all parameters are updated using LS method (line 16, \( UpdAllByLS \) function updates all upper bounds.) Similar to Alg. 1 stopping criteria is met once the absolute difference between previous and current value for all parameters is below the predefined threshold.

Assuming a cell with three parameters \((X, Y \) and \( Z)\), the SPU algorithm works as follows. Initially upper bounds \((U_{X_k}, U_{Y_k}^0, \) and \( U_{Z_k}^0)\) are calculated using LS method. Mean values are calculated as average of upper bound and initial values \((X^0, Y^0, \) and \( Z^0)\). Until cell functions correctly, mean values are updated by averaging mean values and previous values \((e.g., M_{X_j}^1 = \frac{M_{X_j}^1 + X_j^{i-1}}{2})\). Once cell passes the test, all parameters are updated to their mean values and upper bound for each parameter is updated using LS approach. The final \( X_j \) values denote a set of margins representing the upper bounds for all parameters. Similar to Alg. 1, lower bound margins can be calculated by replacing lower bounds \((L_{X_j})\) with upper bounds. Upper bound and lower bound margins for each parameter define an interval. Consequently, a hyper-rectangle is formed as Cartesian product of all intervals. If all parameters lie within this hyper-rectangle, the cell should function correctly.

Alg. 2 uses fewer number of simulations to calculate the margins compared with Alg. 1. The reason is that Alg. 2 updates all parameters at once, rather than individually in the case of Alg. 1. Therefore, it moves faster towards the boundaries of hyper-rectangle of passing points. On the other hand, resulting margins may be more conservative compared with Alg. 1 as it is searches for multiple parameters (i.e., a set of parameters) for which cell functions correctly. This is in contrary with searching for a single parameter by changing individual variables as in Alg. 1.

Intuitively, Alg. 1 is similar to taking small steps toward finding margins (boundaries of hyper-rectangle) whereas Alg. 2 takes larger steps towards boundaries.

Consequently, a hybrid approach is presented to calculate a larger set of margins than that of Alg. 2 using fewer number

Algorithm 1 Individual Parameter Update (IPU)

- Initialize \( X_j^0 \) for \( j = 1...p \)
- Initialize \( j = 1, i = 1, \) stop = false

1: while (!stop) do
2:   for each \( j \) in \( 1...p \) do
3:     \( U_{X_j}^i = UpdateByLS(X_j^i, X_j^{i-1}, \forall k = 1...j - 1, t = j...p) \)
4:     \( M_{X_j}^i = \frac{U_{X_j}^i + X_j^{i-1}}{2} \)
5:     if (!pass(\( X_j^i, M_{X_j}^i, X_j^{i-1}, \forall k = 1...j - 1, t = j...p \)) do
6:       while (!pass(\( X_j^i, M_{X_j}^i, X_j^{i-1}, \forall k = 1...j - 1, t = j + 1...p \)) do
7:         \( M_{X_j}^i = \frac{M_{X_j}^i + X_j^{i-1}}{2} \)
8:       end while
9:     end if
10:    \( X_j^i = M_{X_j}^i \)
11:   end for
12: if (abs(X_j^i - X_j^{i-1}) < threshold, \forall j = 1...p) do
13:   stop = true
14: end if
15: i = i + 1
16: end while
Algorithm 2 Simultaneous Parameter Update (SPU)

Initialize $X_j^0, U_j^0$ for $j = 1..p$
Initialize $j=1, i=1$, stop=false

1: while (stop) do
2:   for each $j$ in $j=1..p$
3:      $M^j_{X_j} = \frac{X_j^{i-1} + X_j^{i-1}}{2}$
4:   end for
5:   if ($\neg$pass($M^j_{X_j}, \forall j = 1..p$)) do
6:     while ($\neg$pass($M^j_{X_j}, \forall j = 1..p$)) do
7:        $M^j_{X_j} = \frac{X_j^{i-1} + X_j^{i-1}}{2}$
8:     end while
9:   end if
10:   for each $j$ in $j=1..p$
11:      $X_j^i = M^j_{X_j}$
12:   end for
13:   if (abs ($X_j^i - X_j^{i-1}$) < threshold, $\forall j = 1..p$) do
14:      stop = true
15:   end if
16:   $U^j_{X_j} = \text{UpdAllByLS}(X_j^i, \forall j = 1..p)$
17:   i = i+1
18: end while

of simulations than Alg. 1.

C. Proposed Hybrid Approach

Hybrid margin calculation (HMC) approach starts by using Alg. 2 to calculate an initial set of margins. To reduce total number of simulations during running Alg. 2, we only calculate upper bounds ($U_j^x$) once and not in every iteration (i.e., line 16 in Alg. 2 is skipped.) Let’s denote these margins as a passing set, namely $P$. Furthermore, a set of mean values for which cell fails to function correctly and is farthest away from $P$ is recorded using Alg. 2. This set of points is called $F$. It is worth mentioning that since initial upper bound margins are calculated using LS method, there is no guarantee that set of upper bound values is a passing set. Hence, this set can be a candidate for $F$ (cf., Section II-C). Also, an empty vector is initialized to keep track of passing sets generated throughout HMC method as $B$ (Alg. 3 line 0).

At first, we find passing sets by changing only one parameter from set $F$ as follows. Mean value for each parameter is calculated as the average of its corresponding $P_j$ and $F_j$ values (line 2). If the cell functions correctly using set of parameters ($X_j^i, M^j_{X_j}, X_j^{i-1}$, $\forall k = 1..j-1, t = j+1..p$), this set is added to vector $B$ (lines 3-5). After iterating all parameters, we check vector $B$. There are two possible cases. 1) Vector $B$ is not empty, hence at least one passing set is found. In this case, function $\text{findOptimalSet}(B)$ finds the passing set for which sum of all margins is maximized. Consequently, optimal set $B_{opt}$ is the output of the algorithm (lines 7-9). 2) Vector $B$ is empty. Hence, we have not found any passing sets yet. In this case, all parameters are set to their corresponding values in $F_j$. At each iteration, we update one parameter to its mean value ($M_j^i$) and check whether the circuit works given updated parameter. If ($X_j^i, X_j^{i-1}$, $\forall k = 1..j, t = j+1..p$) defines a passing set algorithm is terminated and this set is returned (lines 14-17). Otherwise, we continue updating next parameter. If all parameters are updated to their mean value and yet no passing set is found, we update set $F$ by changing all values to their corresponding mean value (lines 19-21). Stopping criteria is met once the absolute difference of all $P_j$'s and $F_j$'s drop below the predefined threshold value. In this situation, set $P$ is returned. This essentially means that margins calculated using Alg. 2 are the largest possible margins and can not be improved anymore.

Calculated margins for AND2 gate (2 input AND gate) using Alg. 3 is depicted in Fig. 4. As it can be observed, there is significant difference between LS method and proposed method especially in lower bound margins of multiple parameters (e.g., $L8$ and $I2$).

IV. SIMULATION RESULTS

To evaluate the effectiveness of the proposed approach and compare it with the conventional LS method, we have used IPD and DPD yield analysis methods (cf., Section II). For this purpose, we have calculated margins as well as critical margin for several logic gates using both proposed HMC and LS methods. We have used PSCAN2 for LS based margin calculation and circuit simulations [11]. For the case of IPD yield
Table I. Results of yield calculation using IPD and DPD methods for different margin calculation algorithms and various SFQ cells. OR2 and OR3 represent 2-input and 3-input OR gates.

<table>
<thead>
<tr>
<th>Cell</th>
<th>IPD Yield (%)</th>
<th>DPD Yield (%)</th>
<th># Simulations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LS</td>
<td>HMC</td>
<td>LS</td>
</tr>
<tr>
<td>AND2</td>
<td>92.1</td>
<td>100</td>
<td>45.3</td>
</tr>
<tr>
<td>AND3</td>
<td>93.7</td>
<td>100</td>
<td>68.5</td>
</tr>
<tr>
<td>OR2</td>
<td>92.5</td>
<td>100</td>
<td>68.7</td>
</tr>
<tr>
<td>OR3</td>
<td>92.5</td>
<td>100</td>
<td>58.0</td>
</tr>
<tr>
<td>INV</td>
<td>92.1</td>
<td>99.5</td>
<td>45.3</td>
</tr>
<tr>
<td>SPLIT</td>
<td>88.8</td>
<td>98.1</td>
<td>79.5</td>
</tr>
<tr>
<td>Average</td>
<td>91.9%</td>
<td>99.6%</td>
<td>60.9%</td>
</tr>
</tbody>
</table>

V. CONCLUSION

This paper presents a novel method for accurate margin calculation of single flux quantum (SFQ) logic cells in a superconducting electronic circuit. The proposed approach can be directly used as a figure of merit to estimate the robustness of a logic cell. This is achieved through efficient state-space exploration of all parameters in the cell structure. Using proposed approach, distinct parameter dispersion (DPD) based yield of SFQ cells increases by 55% on average, compared with state-of-the-art techniques.

VI. ACKNOWLEDGMENTS

The research is based upon work supported by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via the U.S. Army Research Office grant W911NF-17-1-0120. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the ODNI, IARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation herein. The authors would like to thank Naveen Khatam (University of Southern California) for helpful discussion and providing the cell designs used in this paper.

REFERENCES