Josephson Junction Stuck-At Fault Detection in SFQ Circuits

Abdelrahman G. Qoutb, *Student Member, IEEE*, Stephen Whiteley, *Life Member, IEEE*, Jamil Kawa, and Eby G. Friedman, *Fellow, IEEE*

***Abstract*—High reliability is an important requirement for all electronic integrated circuits including superconductive systems. Reliability and yield can be categorized by the failure paths, sequence of faults due to a physical failure, and failure mechanism. Determining the defects and faults is essential to enhance the lifetime of superconductive systems. Josephson junctions (JJs) are the base element of single flux quantum systems. The primary contributions described in this article are a set of JJ-based fault models followed by a methodology for developing high-level fault models. Based on these models, test vectors can be generated to detect the type and/or location of these JJ faults.**

***Index Terms*—Design for testability (DFT), fault models, single flux quantum (SFQ), superconductive digital electronics, superconductive (SC) integrated circuits.**

I. Introduction

THE challenge of achieving high performance with high reliability is escalating due to dimensional scaling, novel materials and devices, and operation in severe conditions (such as extreme cryogenic temperatures and subterahertz frequencies). These reliability challenges, combined with yield issues, are exacerbated by exotic manufacturing technologies.

Single flux quantum (SFQ) logic is a superconductive (SC) technology for low-power, high-performance cryogenic computing. The development of SFQ technology has enabled complex integrated circuits achieving over 11 000 Josephson junctions (JJs) for digital signal processors [1] and similar complexity prototype RSFQ microprocessors [2]. SFQ circuits with a regular layout structure, such as an ac-biased SFQ shift register have reached 800 000 JJs [3], operating at subterahertz clock frequencies. The achievable frequencies and cryogenic environment make SFQ circuits difficult to control via external probing. Prototype evaluation of these circuits, therefore, requires advanced testing methodologies.

Manuscript received 28 September 2022; revised 24 January 2023; accepted 6 February 2023. Date of publication 22 March 2023; date of current version 25 April 2023. This work was supported in part by the National Science Foundation under Grant 2124453, in part by the Intelligence Advanced Research Projects Activity (IARPA) under Grant W911NF-17-9-0001, and in part by the Qualcomm and Synopsys. This article was recommended by Associate Editor Masamitsu Tanaka. *(Corresponding author: Abdelrahman Qoutb.)*

Abdelrahman G. Qoutb and Eby G. Friedman are with the Department of Electrical and Computer Engineering, University of Rochester, Rochester, NY 14627 USA (e-mail: a.qoutb@rochester.edu; friedman@ece.rochester.edu).

Stephen Whiteley and Jamil Kawa are with Synopsys, Inc., Mountain View, CA 94043 USA (e-mail: whiteley@synopsys.com; jamil@synopsys.com).

Digital Object Identifier 10.1109/TASC.2023.3260144

Reliability and yield can be categorized by the failure paths and failure mechanisms. Determining the defects and faults is essential to enhancing the lifetime and testability of SC systems. This capability is achieved by improving the fault coverage, where the system is evaluated to identify the characteristics of the faults, such as the quantity, location, and type. Understanding the behavior of each failure mechanism and the development of effective and reliable methodologies that exploit design for testability (DFT) techniques prior to fabrication are vital to the development of testable SC systems.

The basic elements of SFQ systems are JJs, resistors, inductors, and interconnects [4]. In this article, high-level JJ-based fault models are proposed, and the required test vectors are described to detect the location and type of these faults [5], [6].

Several significant differences exist between conventional transistor-based CMOS fault models and JJ-based SFQ fault models beyond subterahertz clock frequencies and the cryogenic environment. An SFQ signal is represented by the existence of an SFQ pulse not as a voltage level (as in CMOS). In both CMOS and SFQ device-based faults, it is challenging to identify the location and type of faults within a system. The following additional differences prevent the use of standard CMOS-based DFT techniques [6], [7].

1) Only two states exist in SFQ logic, zero (the absence of a pulse) or one (existence of a pulse). In CMOS logic, output states, such as zero, one, and floating may exist.
2) SFQ logic gates are inherently clocked and latched within at least one storage loop, where several clock cycles are required to produce an output [8]. Unlike CMOS, in SFQ systems, additional information, such as the number of cycles, is required by a test controller.
3) Limited fan-out of SFQ gates and flip flops [9]. Splitters are required to provide additional outputs [9].

The previous work on enhancing the testability of SC electronics, emphasizing process variations, defects, and structural testing, has been considered in [6], [10], [11], and [12]. In [10], a methodology is proposed to test timing violations due to process variations. This methodology is achieved by characterizing logic cells due to process variations and identifying the delay excitations [10]. Hence, test patterns are generated that guarantee to excite the worst case delay along each target multicycle path [10]. In [11], fault models are produced based on a large number of simulations. These models are based on variation-induced failures by varying the value of the inductors, resistors, and critical currents [10]. Kerkhof and Speek [12]

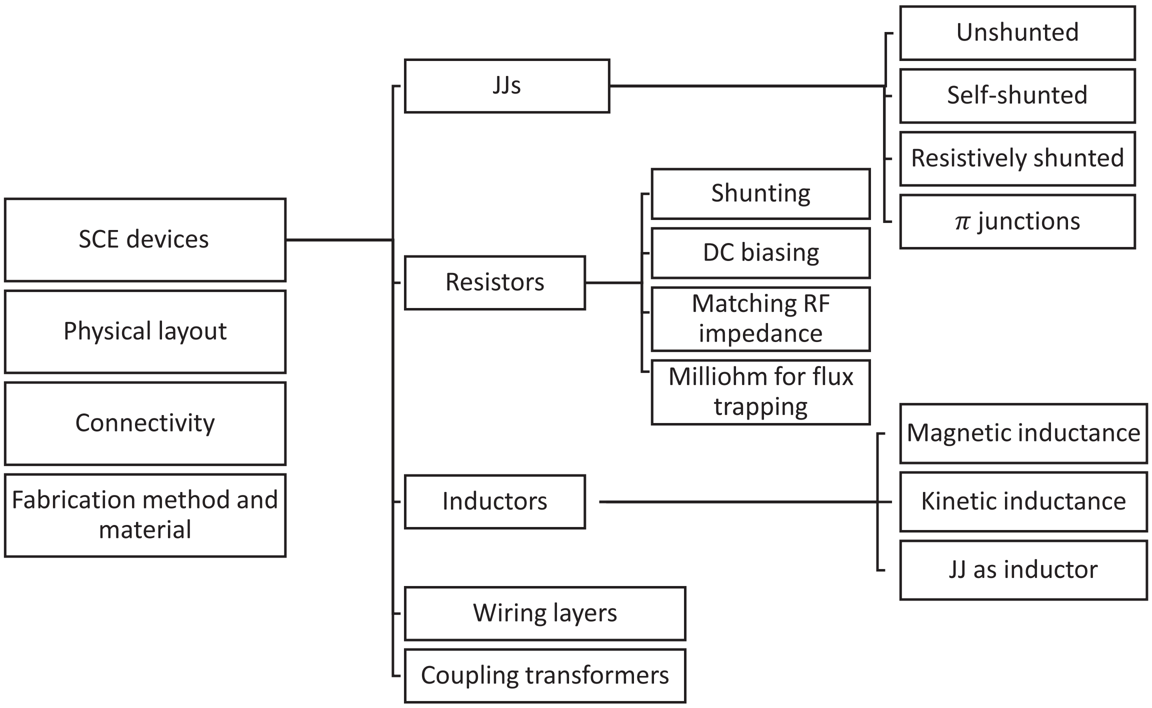

Fig. 1. SFQ fault mechanisms. Component-based faults are attached to a specific SFQ component. High-level models detect the faults associated with resistively shunted JJs. Other SFQ fault mechanisms include faults associated with the physical layout, faults due to connectivity between the devices, and faults due to limitations in the manufacturing process.

suggested defect-oriented testing by introducing possible defects, such as shorts in the same layer, bridges between layers, opens in layers and vias, and pinholes in the thin oxide layer. Hence, elements are inserted to detect these defects (as an example, inserting large and small capacitors to detect pinholes). Based on statistical information from detecting these defects, a model is presented to measure the quality of the fabrication process. From a literature survey, fault models that target specific defect mechanisms are missing. More research in this area is required to produce more efficient SFQ-based test methodologies.

A methodology is proposed here to include DFT within SFQ systems. To the best of our knowledge, this work is the first to describe JJ-based stuck-at faults. This objective is achieved by developing high-level fault models that target JJ-based faults, such as stuck-at in a SC state or an open state. These fault models can be exploited to develop a fault simulation algorithm. The required test vectors to identify the type and location of these sets of faults are generated based on a high-level fault model. A summary of the quality measures of each fault model is discussed in this article. The fault coverage of open-circuit (OC) and short-circuit faults and the location of each logic cell are identified.

Potential faults within SFQ systems are categorized by device-based faults, fabrication-based faults, and dc bias network faults, as shown in Fig. 1. JJ-based fault models are evaluated to generate the required test vectors to determine the location and/or type of defect.

The primary contribution of this article is a test methodology for SFQ systems. High-level JJ-based fault models are described followed by a methodology for developing a fault model to target a specific block or type of fault.

The rest of this article is organized as follows. JJ-based fault mechanisms and related fault models are discussed in Section II. These proposed JJ-based fault models are validated in Section III. The required test vectors to detect and allocate JJ-based faults within an SFQ system are presented in Section IV. The fault coverage of the proposed models are presented in Section V. A methodology to develop a block-level JJ-based fault model to generate the required test vectors is described in Section VI. Finally, Section VII concludes this article.

II. JJ-Based High-Level Fault Models

Multiple types of JJs exist within SFQ systems, such as unshunted, self-shunted, resistively shunted, and π junctions. Resistively shunted JJs are the most advanced fault type within state-of-the-art high-performance SFQ circuits. Faults associated with resistively shunted JJs are the focus of this article.

A faulty resistively shunted JJ has four modes of operation, stuck-at SC, stuck-at resistive, OC, and noisy switching. Further simplifications are necessary to develop JJ-based fault models that support complex testability mechanisms considering millions of JJs.

The great majority of physical failures results in stuck-at shorts and opens [13]. In this article, two JJ-based fault modes are considered, stuck-at SC state and stuck-at OC state.

Multiple physical defects can lead to a JJ stuck in the SC state, such as a JJ with a higher critical current than the expected value. Typical margins for a bias network are 20% to 30% of the critical current [14], [15]. If the critical current of a JJ is above this margin, the device behaves as a stuck-at SC state.

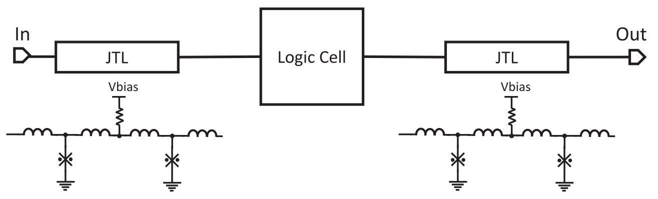

Fig. 2. Configuration of the logic cell under test where a Josephson transmission line is placed at the primary inputs and outputs of the logic cell under test.

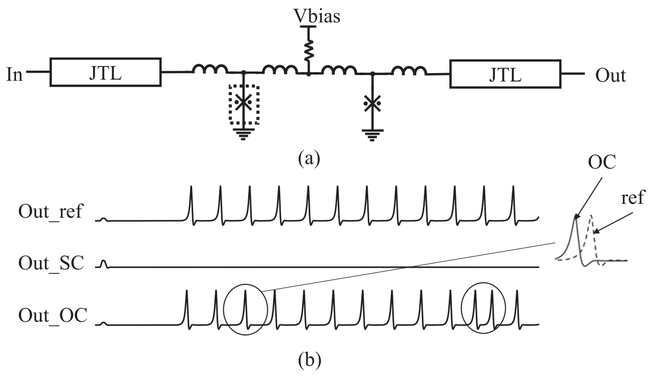

Fig. 3. JTL faults, (a) model of a JTL, and (b) simulation of a JJ stuck-at SC or OC state, indicating additional failure behaviors (circled). The squared JJ is the faulty JJ. Out_ref is the output of a JTL with a reference cell (without faults). Out_SC and Out_OC are, respectively, the output of a JTL with a JJ stuck-at SC state and OC state.

A JJ stuck in the OC state can occur if a break exists in the tunneling barrier or interconnect. In this article, a JJ stuck in the SC state is modeled as a JJ with a high critical current (5 mA) to ensure the operation is in the stuck-at SC state, whereas a JJ stuck-at in the OC state is modeled as an OC.

A. Fault Simulation and Analysis

To develop a high-level fault model of a logic cell with a faulty JJ, the response of a circuit is considered to be only due to one fault type in a single JJ at a time. The logic cells within an SFQ cell library are based on the configuration shown in Fig. 2. A JJ-based fault model is presented for the following SFQ cells: Josephson transmission line (JTL), splitter, DFF, OR, and AND gates.

The simulation environment to model JJ-based faults within a JTL is illustrated in Fig. 3(a). The output of a JTL due to a JJ stuck-at SC state is zero. The output of a JTL with a JJ stuck-at OC state is challenging to detect. This output exhibits a small difference in delay from a reference JTL without faults, as shown in the circled areas depicted in Fig. 3(b).

Two types of JJs exist within a splitter, the driver JJ and the branch JJ. The simulation environment to model JJ-based faults within a splitter cell is shown in Fig. 4. The faulty output of a splitter with the driver JJ stuck in the OC state [see Fig. 4(a)] is similar to a reference output (without JJ faults), as illustrated in Fig. 4(a). In this condition, a stuck-at OC fault in the driver JJ is undetectable. The output of a splitter cell with a branch JJ

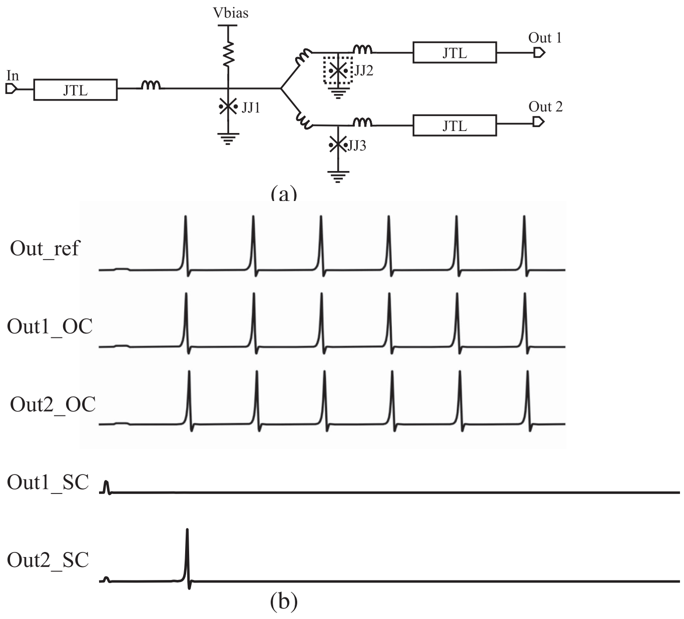

Fig. 4. Splitter faults, (a) splitter cell, and (b) simulation of J2 stuck-at SC or OC state.

stuck in the SC state depends upon the location of the faulty JJ, as shown in Fig. 4(b). The output of a splitter cell, attached to a faulty JJ (stuck in the SC state), exhibits a single pulse followed by zeros, as depicted in Fig. 4(b).

The same procedure is followed to model other SFQ cells, such as DFF, AND, and OR gates. These logic cells have multiinput and multioutput ports, including an input clock signal. To provide a fault model for these gates, the cell under test is analyzed for all input conditions with the clock signal enabled or disabled. Testing all input cases is essential to determine the location and type of faulty JJs within a cell.

At least one storage loop exists in these logic gates. The location of a JJ within these logic gates sets the dependence of the cell function on the clock signal or latching operation. For the example shown in Fig. 5(a), a fault in J2 influences the clock signal, whereas a fault in J1 or J3 affects the storage loop. At least two test vectors are required to detect faults within these clocked logic gates, as discussed in Section III.

As previously discussed in Section II, high-level fault models are based on different cell behaviors caused by JJ-based faults. The fault models of a JTL, splitter, and DFF cell are, respectively, tabulated in Table II(a)–(c). These fault models are independent of technology and/or manufacturing process. As listed in Table II, the output of each logic cell due to a fault in a JJ is compared with a reference cell (without faults). The faulty output is highlighted as a gray cell.

As shown in Fig. 5(b), an OR cell is composed of eight JJs with four control JJs and one DFF, whereas, as illustrated in Fig. 5(c), an AND cell is composed of 11 JJs, forming two DFFs and three control JJs. To provide a high-level JJ-based fault model of both an OR cell and an AND cell, the output is evaluated with and without the clock signal. As an example, when the clock signal is ON, a clock pulse is inserted after each input. High-level JJ-based

TABLE I
HIGH-LEVEL JJ-BASED FAULT MODELS, (A) OR CELL, AND (B) AND CELL

A	B	Ref		J1		J2		J3		J4		J5		J6_w/o _clck		J6_w _clck		J7_w/o _clck		J7_w _clck		J8_w/o _clck		J8_w _clck	
		W_clck	W/o_clck	OC	SC	OC	SC	OC	SC	OC	SC	OC	SC	OC	SC	OC	SC	OC	SC	OC	SC	OC	SC	OC	SC
0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	1	0	1	1	1	0	1	0	1	1	0	1	1	0	1	0	0	0	0	1	1	0	1	0
1	0	1	0	1	0	1	1	1	1	1	0	0	1	1	0	1	0	0	0	0	1	1	0	1	0
1	1	1	0	1	1	1	1	1	1	1	1	0	1	1	0	1	0	0	0	0	1	1	0	1	0

(a)

A	B	Out_ref		J1		J2		J3		J10		J11		J4w/o _clck		J4_w _clck		J5_w/o _clck		J5_w _clck		J6_w/o _clck		J6_w _clck		J7w/o _clck		J7_w _clck		J8_w/o _clck		J8_w _clck		J9_w/o _clck		J9_w _clck	
		W _clck	W/o _clck	OC	SC	OC	SC	OC	SC	OC	SC	OC	SC	OC	SC	OC	SC	OC	SC	OC	SC	OC	SC	OC	SC	OC	SC	OC	SC	OC	SC	OC	SC	OC	SC	OC	SC
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	0	0	1	0	1	1	0	0	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

The faulty output is highlighted as a gray cell.

(b)

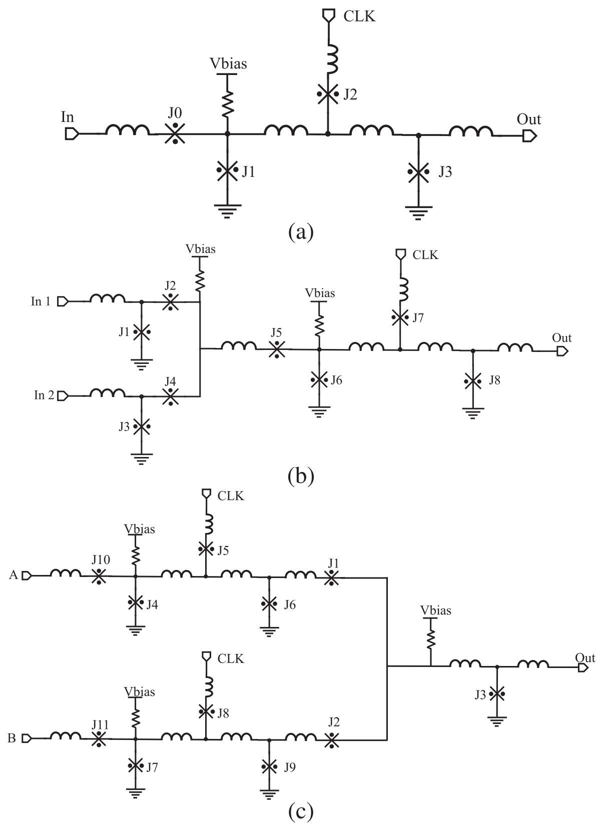

Fig. 5. Circuit structure of (a) DFF, (b) OR cell, and (c) AND cell.

fault models of OR gate and AND gate are, respectively, listed in Table I(a) and (b). Note that a fault in the JJs forming the DFFs within the OR and AND gates may exhibit a different behavior when the clock signal is ON or when it is OFF. However, a fault in the control JJs can only be identified when the clock signal is ON.

III. Validation of Proposed JJ-Based Fault Models

The benchmark circuit shown in Fig. 7 is used here to validate the proposed JJ-based fault models. This validation process identifies the logic paths of JJ-based faults within an SFQ system. As an example, one fault is inserted at J2 within the AND_1 gate. As listed in Table I(b), when J2 is stuck in the SC state within an AND gate, the faulty output is one when $A = 1$ and $B = 0$. As shown in Fig. 6(a), the faulty and reference AND gates produce a correct output when $A = 1$ and $B = 1$. When $A = 1$ and $B = 0$, output AND_1 is one when J2 is stuck in the SC state. This faulty output propagates to the second stage of the benchmark circuit, as shown in Fig. 6(b). Out_Ref is the result of an AND operation between AND_1 and C_1. The second-stage AND gate is free of faults. Hence, the source of the fault is only from AND_1.

Based on these validation results the following: 1) the proposed JJ-based high-level fault models describe the operation of a cell with JJs stuck-at in either the OC or the SC state; 2) the influence of JJ-based stuck-at faults is localized within the gate where the fault exists. Hence, a JJ stuck-at fault in a specific gate does not cause additional faults in other gates. Based on these characteristics, the proposed technique for developing high-level fault models can shift JJ-based faults from a gate-level model to a block-level model. This technique can be used to develop a fault simulation tool for JJ-based stuck-at faults.

IV. Test Vector Generation

The required test vectors that identify the location and/or type of JJ-based fault are based on the proposed fault models. As an example, consider a JJ-based fault model of a JTL, as listed in Table II(a). OC faults cannot be identified since the output of a faulty JTL with J1 stuck-at in the OC state is similar to a reference cell. The output of a faulty JTL with a JJ stuck in the SC state can be identified by detecting zero when a one is

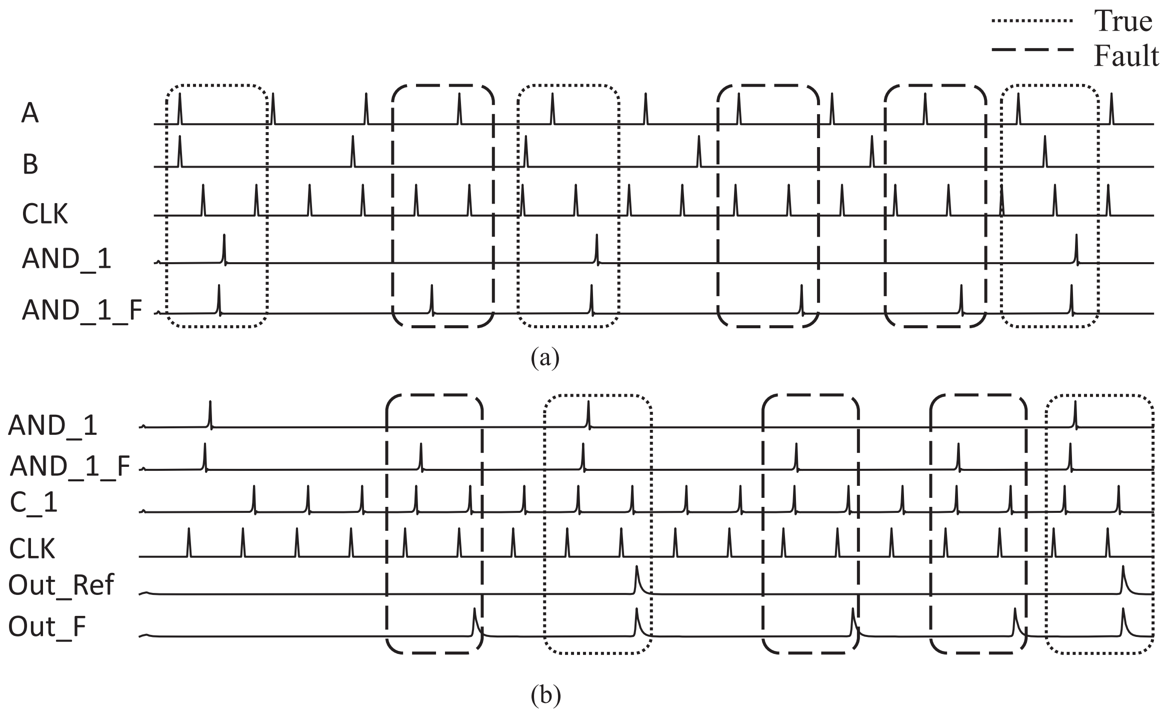

Fig. 6. Validation of the proposed JJ-based fault models where a single JJ fault is inserted at AND_1 gate. J2 is stuck-at SC, as shown in Fig. 7. Output of the reference cell without any faults, (a) at the first stage, faulty output AND_1_F is one when either A = 1 and B = 1 or A = 1 and B = 0, whereas the true operation AND_1 is one only when A = 1 and B = 1. (b) At the second stage, where no faults exist, but the faulty output of the first stage propagates to the second stage. This behavior exemplifies that JJ-based stuck-at faults are localized faults that only affect the operation of a specific cell (the cell with the faulty JJ).

TABLE II
HIGH-LEVEL JJ-BASED FAULT MODELS, (A) JTL, (B) SPLITTER, AND (C) DFF

In	Out		
	Ref	OC	SC
0	0	0	0
1	1	1	0

(a)

	In	Ref		OC		SC	
		Out1	Out2	Out1	Out2	Out1	Out2
J1	0	0	0	0	0	0	0
	1	1	1	1	1	0	0
J2	0	0	0	0	0	0	0
	11	11	11	11	11	00	10
J3	0	0	0	0	0	0	0
	10	11	11	11	11	10	00

(b)

Input	CLK	00	1	010	101
	In	11	0	101	010
Ref		00	0	010	001
J0	OC	00	0	000	000
	SC	00	0	010	001
J1	OC	00	0	010	001
	SC	00	0	000	000
J2	OC	00	0	000	000
	SC	00	0	010	001
J3	OC	01	0	010	001
	SC	00	0	000	000

(c)

The faulty output is highlighted as a gray cell. Two test vectors are required to detect the fault if J2 or J3 is stuck-at SC. Detecting JJ-based faults within a DFF is achieved by applying up to three test vectors to set or reset the stored value within the storage loop of a DFF, regardless of the initial condition before testing.

inserted at the JTL. Another example, listed in Table II(b), is the location of stuck-at SC faults, which can be identified by detecting the two outputs of a splitter. If a one is inserted into a splitter and a zero is detected at the two outputs, a stuck-at SC is identified at the driver JJ. If a one is inserted into a splitter and a one is detected in one branch and a zero in the other branch, a stuck-at SC is detected at the JJ located in the other branch.

A complete list of test vectors to detect the location and/or type of JJ-based fault within an SFQ system is listed in Table III. Based on this list, different types of test vectors can be applied, given as follows.

1) Test vectors can be applied to detect a specific fault at a specific location. For example, detecting if J1 (at a specific location) is stuck-at SC (for a specific fault) within a splitter cell or if J3 is stuck-at OC within a DFF cell.
2) Test vectors can be applied to detect if a specific location has a JJ-based fault (stuck-at SC or OC); for example, detecting if J4 is stuck-at SC or OC fault within an OR gate by applying A=10 and B=00.
3) Test vectors can be applied to detect if a JJ-based stuck-at fault occurs within a cell (without identifying the type or location of the fault); for example, applying A=X and B=X to an OR cell, where X means any value.

V. FAULT COVERAGE OF JJ-BASED FAULTS

A summary of JJ-based faults within an SFQ system is listed in Table IV. Considering specific logic cells, 100% of OC faults and 64% of SC faults can be detected within an AND cell; 100% of JJs stuck in the SC state within a splitter cell can be identified and located. It is, however, challenging to determine the type and/or location of all JJ stuck faults within JTL, DFF, and OR cells.

TABLE III
TEST VECTORS TO DETECT THE LOCATION AND/OR TYPE OF JJ-BASED FAULTS WITHIN AN SFQ CELL

Cell	Detected faults	In Test vector	Ref/ Ref1	Out/Out1	Ref2	Out2
JTL	J1 SC or J2 SC	1	1	0		
Splitter	J1 SC	11	1	00	1	00
	J2 SC	11	11	00	11	10
	J3 SC	11	11	10	11	00
DFF	J1 SC, J2 OC, or J3 SC	CLK 010 IN 101	010	000		
	J3 OC	CLK 00 In 11	00	01		
OR	J1 SC or J4 SC	A 1, B 0	1	0		
	J2 OC or J4 OC	A 0, B 0	0	1		
	J2 SC or J3 SC	A 0, B 1	1	0		
	J6 OC or J8 OC	CLK 0 A 1, B 0	0	1		
	J5 OC, J6 SC, J7 OC, and J8 SC	A 1, B 1	1	0		
AND	J1 SC	A 01, B 10	00	10		
	J2 SC	A 01, B 10	00	01		
	J3 OC	A 01, B 10	00	11		
	J1 OC, J2 OC, J3 SC, J4 OC, J4 SC, J5 OC, J6 OC, J6 SC, J7 OC, J7 SC, J8 OC, J9 OC, J9 SC, J10 OC, or J11 OC	A 1, B 1	1	0		

The JJ labels are illustrated in the circuit structures shown in Figs. 3, 4, 5, 9, and 10.

TABLE IV
FAULT COVERAGE OF JJ-BASED FAULTS WITHIN MULTIPLE SFQ CELLS WHERE THE NUMBER OF JJS WITHIN EACH CELL, TOTAL NUMBER OF JJ FAULTS THAT MAY EXIST, NUMBER OF TOTAL FAULTS THAT CAN BE DETECTED, NUMBER OF ONLY OC OR ONLY SC FAULTS THAT CAN BE DETECTED, AND NUMBER OF SPECIFIC OC OR SC FAULT THAT CAN BE DETECTED AT A SPECIFIC LOCATION

Cell	#JJs	#Faults	Detected faults			Detected location	
			#	OC	SC	OC	SC
JTL	2	4	2	0	2	0	0
Splitter	3	6	3	0	3	0	3
DFF	3	6	4	2	2	1	0
OR	8	16	12	6	6	0	0
AND	11	22	18	11	7	1	2
Total	27	54	39	19	20	2	5
Percent			72%	70%	74%	7%	19%

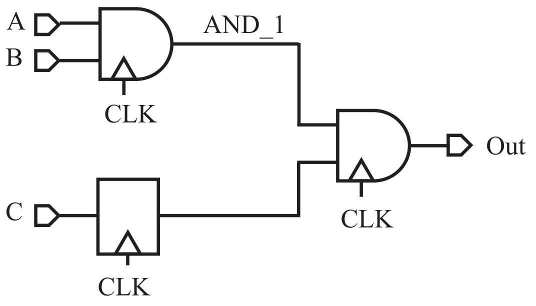

Fig. 7. Benchmark circuit to evaluate JJ-based fault models. A single JJ fault is inserted in a two-level logic cell. The output is compared with the predicted output based on the fault model.

A total of 72% of JJ faults can be identified within an SFQ cell library composed of JTL, splitter, DFF, OR, and AND cells. Only 70% of OC faults can be identified, and 74% stuck-at SC state faults can be determined. The location of only 19% of stuck-at SC state faults can be identified, whereas the location of 7% stuck-at OC faults can be determined.

These numbers do not directly reflect a fault coverage of a logic cell but do reflect an estimate of the fault coverage of JJ-based faults. As an example, for the benchmark circuit presented in Fig. 7, the circuit is composed of two AND gates and one DFF with a total of 25 JJs with a possibility of 50 JJ-based faults. Assuming the ability to detect and observe the primary input and output of all of the logic gates, the following fault coverage can be achieved: 80% of JJ-based faults, 96% of OC faults, and 64% of SC faults.

These low fault coverage results are due to the response of SFQ cells to stuck OC faults. In most cases, JJs stuck-at OC fault pass an SFQ pulse without interruption, such as the JJs within a JTL, as illustrated in Fig. 3(b). In these scenarios, it is challenging to detect stuck-at OC faults through the proposed high-level functional model.

One method to enhance the fault coverage of stuck-at OC faults can be achieved by applying a test methodology to detect and distinguish a transition failure between a faulty output of a

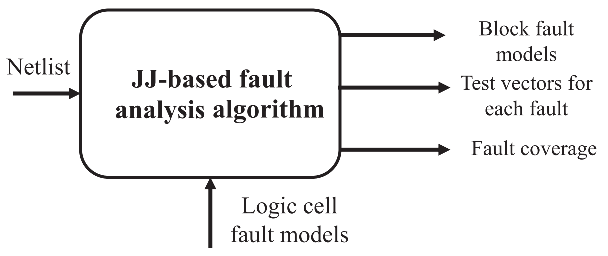

Fig. 8. Block diagram of proposed algorithm to generate test vectors to identify JJ-based faults within an SFQ system.

stuck-at OC fault and a reference cell. As illustrated in Fig. 3(b), a small delay is detected between the faulty output of a JTL with a stuck-at OC fault and a reference cell. Depending upon the location of a JJ, a different delay is detected. Transition faults have been widely used to model stuck-open faults for determining transistor-based faults in CMOS systems [16]. These on-chip transition-based testing mechanisms are challenging to design, require significant power, and increase the test time [16].

It is difficult to identify stuck-at faults at the device level. In CMOS systems, gate-level or node-level stuck-at faults are preferred over transistor stuck-at faults since it is easier to detect and locate faults with high fault coverage [17]. CMOS-based node or gate stuck-at faults simultaneously affect several transistor terminals [13]. High fault coverage is to be expected. For most CMOS stuck-at fault models, the coverage of a transistor-level stuck-at fault is significantly less than the fault model of gate-level stuck-at faults [17]. A higher coverage is obtained for node stuck-at faults than gate stuck-at faults.

VI. JJ-Based Targeted Testing

A test vector generation algorithm is required to target a specific cell and fault. The objective of this algorithm is to provide a block-level fault model to generate test vectors to detect faults and determine the fault coverage within a block, as illustrated in Fig. 8.

To increase the fault coverage of JJ-based faults within an SFQ system, a test methodology is necessary that generates the required test vectors to detect only stuck-at SC faults or only OC faults. As an example, within an OR cell, J1, J2, J3, and J4 stuck-at SC faults can be detected by applying two test vectors (A=10 and B=01) with a clock pulse applied after each of the test vectors.

Pseudocode describing the algorithm generating test vectors to target a specific fault type or specific fault location is shown in Algorithm 1. The time complexity of the proposed algorithm for different benchmark circuits is O(N), where N is the number of gates. The proposed algorithm generates the required test vectors, given as follows.

1) The block under test is evaluated for all possible fault scenarios. A fault is inserted. Each JJ is modeled as either stuck-at SC or OC. A stuck-at OC JJ is modeled as an OC, whereas a stuck-at SC JJ is modeled as a JJ with a high critical current (such as 5 mA).

Algorithm 1: Pseudocode of Algorithm for Generating Test Vectors to Identify JJ-Based Faults.

```
Input: Number of gates $N$, number of input conditions $C$,
    JJ-based fault models of all logic cells $LM$ with JJ
    number $JJn$ under JJ-based fault $f$ of 0 for SC or 1 for
    OC fault, target testing requirements
Output: Block JJ-based fault model $BM$, test vector of
    target testing $Test\_Vector$, undetected faults $DF$, and
    detected faults $DF$
 1: Evaluate the behavior of the netlist with reference cells
    under all input conditions $Out\_Ref$
 2: For $k \leftarrow 1$ to $N$ do          ▷ Evaluate each logic cell
 3:   For $i \leftarrow 1$ to $JJn$ do ▷Evaluate each JJ-based fault
      within the logic cell
 4:     For $j \leftarrow 1$ to $C$ do ▷Evaluate each input condition
 5:       Evaluate the behavior of the netlist with $LM_k$
          under a fault in JJ $i$ with fault type $f$ under input
          condition $j$,
 6:       $BM \leftarrow (Out, k, i, f, j)$
 7:       if $(Out) = Out\_Ref$ then
 8:         $UF \leftarrow (k, i, f, j)$    ▷Undetected fault in $K$
            cell, $i$ JJ, fault $f$, and input condition $i$
 9:         $uf = uf + 1$
10:       else
11:         $DF \leftarrow (Out, k, i, f, j)$    ▷Detected fault in
            $K$ cell, $i$ JJ, fault $f$, and input condition $i$
12:         $df = df + 1$
13:       end if
14:     end for
15:   end for
16: end for
17: Extract $DF$ with $f = 0$ to group stuck-at SC faults
    $BM_{SC}$
18: Extract $DF$ with $f = 1$ to group stuck-at SC faults
    $BM_{OC}$
19: Extract $DF$ regardless $f$ to group a stuck-at a fault
    $BM_U$
20: Generate the test vector for each group
    $Test\_Vector_{SC}, Test\_Vector_{OC}, Test\_Vector_U$
```

2) The output of each fault scenario is compared with a reference output (where no fault is inserted). Undetected faults are those faults where the faulty output is similar to the reference output.
3) All identical faulty outputs due to the same input combination(s) are grouped together. These faults share the same test vectors.

VII. Conclusion

Advanced testing methodologies are required to support complex digital SFQ systems. In this article, JJ-based fault models are proposed for specific gate types. A faulty JJ has four modes of operation, stuck-at SC, resistive, OC, or noisy switching. In total, two JJ-based fault modes are considered in this article, stuck-at SC state and stuck-at OC state. A JJ stuck in the SC state is modeled as a JJ with a high critical current, whereas a JJ stuck in the OC state is modeled as an OC. A high-level JJ-based fault model is presented for the following RSFQ cells: JTL, splitter, DFF, OR, and AND. Test vectors to identify the type and location of a set of faults are generated based on these high-level fault models. The fault coverage of the OC and SC faults and the location of each logic cell are identified; specifically, 72% of JJ-based faults (OC, SC, or both) can be detected within an SFQ system. The fault coverage of a JJ-based fault is 74% of SC faults and 70% of OC faults. While it is challenging to identify the location of OC faults within an SFQ system, all SC faults within a splitter cell can be identified, and the location of 18% of the SC faults within an AND cell can be determined. A methodology is also proposed to develop a block-level fault model to produce the required test vectors to identify the type and location of JJ faults within SFQ systems.

References

[1] H.-D. Hahlbohm and H. Lübbig Eds., *SQUID '85 Superconducting Quantum Interference Devices and Their Applications*. Berlin, Germany: De Gruyter, 1986.

[2] O. A. Mukhanov et al., "Superconductor digital-RF receiver systems," *IEICE Trans. Electron.*, vol. E91-C, no. 3, pp. 306–317, Mar. 2008.

[3] V. K. Semenov, Y. A. Polyakov, and S. K. Tolpygo, "AC-biased shift registers as fabrication process benchmark circuits and flux trapping diagnostic tool," *IEEE Trans. Appl. Supercond.*, vol. 27, no. 4, Jun. 2017, Art. no. 1301409.

[4] T. Jabbari, G. Krylov, S. Whiteley, E. Mlinar, J. Kawa, and E. G. Friedman, "Interconnect routing for large-scale RSFQ circuits," *IEEE Trans. Appl. Supercond.*, vol. 29, no. 5, Mar. 2019, Art. no. 1102805.

[5] G. Krylov and E. G. Friedman, *Single Flux Quantum Integrated Circuit Design*. Cham, Switzerland: Springer, 2022.

[6] G. Krylov and E. G. Friedman, "Design for testability of SFQ circuits," *IEEE Trans. Appl. Supercond.*, vol. 27, no. 8, Dec. 2017, Art. no. 1302307.

[7] M. Bushnell and V. Agrawal, *Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits*. New York, NY, USA: Springer, 2004.

[8] G. Krylov and E. G. Friedman, "Globally asynchronous, locally synchronous clocking and shared interconnect for large-scale SFQ systems," *IEEE Trans. Appl. Supercond.*, vol. 29, no. 5, Aug. 2019, Art. no. 3603205.

[9] T. Jabbari, G. Krylov, J. Kawa, and E. G. Friedman, "Splitter trees in single flux quantum circuits," *IEEE Trans. Appl. Supercond.*, vol. 31, no. 5, Aug. 2021, Art. no. 1302606.

[10] F. Wang and S. K. Gupta, "An effective and efficient automatic test pattern generation (ATPG) paradigm for certifying performance of RSFQ circuits," *IEEE Trans. Appl. Supercond.*, vol. 30, no. 5, Jan. 2020, Art. no. 1300711.

[11] M. Li, F. Wang, and S. Gupta, "Data-driven fault model development for superconducting logic," in *Proc. IEEE Int. Test Conf.*, 2020, pp. 1–5.

[12] H. G. Kerkhoff and H. Speek, "Defect-oriented testing of Josephson logic circuits and systems," *Physica C: Supercond.*, vol. 350, no. 3/4, pp. 261–268, Feb. 2001.

[13] J. Galiay, Y. Crouzet, and M. Vergniault, "Physical versus logical fault models MOS LSI circuits: Impact on their testability," *IEEE Trans. Comput.*, vol. 29, no. 6, pp. 527–531, Jun. 1980.

[14] A. F. Kirichenko et al., "ERSFQ 8-bit parallel arithmetic logic unit," *IEEE Trans. Appl. Supercond.*, vol. 29, no. 5, Aug. 2019, Art. no. 1302407.

[15] G. Krylov and E. G. Friedman, "Design methodology for distributed large-scale ERSFQ bias networks," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 28, no. 11, pp. 2438–2447, Nov. 2020.

[16] S. D. Millman and E. J. McCluskey, "Detecting stuck-open faults with stuck-at test sets," in *Proc. IEEE Custom Integr. Circuits Conf.*, 1989, pp. 22.3/1–22.3/4.

[17] P. Liden and P. Dahlgren, "Coverage of transistor-level and gate-level stuck-at-faults in CMOS checkers," in *Proc. IEEE Int. Symp. Circuits Syst.*, 1995, pp. 2124–2127.