A Tool Kit for the Design of Superconducting Programmable Gate Arrays

Coenrad Johann Fourie

Dissertation presented for the Degree of ***Doctor of Philosophy in Engineering*** *at the University of Stellenbosch*

Promoter:

Prof. W. J. Perold

UNIVERSITEIT
STELLENBOSCH
UNIVERSITY

December 2003

Declaration

I, the undersigned, hereby declare that the work contained in this dissertation is my own original work, unless stated otherwise, and that I have not previously in its entirety or in part submitted it at any university for a degree.

Signature: ....

Date: 31 October 2003

Abstract

The development of a tool kit for the design of superconducting programmable gate arrays (SPGAs) is discussed. A circuit optimizer using genetic algorithms is developed and evaluated. Techniques and a program are also developed for the generation of segmentized 3D models with which to calculate inductance in circuit structures through *FastHenry*. The ability to add random variations to the dimensions of the models is included. These tools are then used to design novel latching elements that allow the construction of reprogrammable Rapid Single Flux Quantum (RSFQ) circuits. A circular process is used, whereby layouts are converted back to circuit diagrams through element extraction, and reoptimized if necessary. Two programmable frequency dividers are then designed; one for testing the routing and switch structures and programming architecture of an SPGA, and another compact one for testing the latching elements and off-chip interface. The dissertation concludes with an overview of the circuits necessary for the implementation of a fully functional SPGA.

Opsomming

Die ontwikkeling van ’n gereedskapstel vir die ontwerp van supergeleier FPGA’s (SPGA’s) word bespreek. Eerstens word ’n stroombaanoptimeerder, wat met genetiese algoritmes funksioneer, ontwikkel en geëvalueer. Daarna word tegnieke en ’n program ontwikkel om driedimensionele segmentmodelle te genereer waaruit *FastHenry* die induktansie van stroombaanstrukture kan bepaal. Die vermoë om toevalsveranderinge by die dimensies van die modelle te voeg, is ook ingesluit. Hierdie gereedskap word dan gebruik om nuwe grendelelemente te ontwerp waarmee herprogrammeerbare *Rapid Single Flux Quantum* (RSFQ) stroombane gebou kan word. ’n Sirkulêre proses word gevolg, waarvolgens uitlegte na stroombaandiagramme teruggeskakel kan word (deur elementonttrekkings) en, indien nodig, heroptimeer kan word. Twee programmeerbare frekwensiedelers word daarna ontwerp; een om die pulsvervoer- en skakelstrukture, asook programmeringsargitektuur van ’n SPGA te toets, en ’n ander, kompakter een om die grendelelemente en warmlogika koppelvlakke mee te toets. Die proefskrif sluit af met ’n oorsig oor die stroombane benodig vir die implementering van ’n volledig funksionele SPGA.

Acknowledgements

Thanks go to everybody who endured countless questions – from the inquisitive and well-prepared down to the stupidly impulsive – from me during the most critical stages of the design and layout of the programmable frequency divider. Oleg Mukhanov, Masoud Radparvar, Steve Kaplan and Sergey Tolpygo of Hypres were particularly harassed through e-mail. Other foreign contributors were Anna Kidiyarova-Shevchenko and Thomas Ortlepp. I am also particularly indebted to Steve Whiteley and his array of programs, as well as support, that made most of the simulations for this project possible.

Local colleagues who endured a barrage of questions include Petrie Meyer, John Cloete, Keith Palmer, Riana Geschke and Ludwig Schwardt.

To Héndri – thanks for the late-night sessions in probability theory.

To Willem Perold, thank you for supporting me in taking my work and theories to faraway conferences.

Lastly, a big thank you to my inquisitive students who asked such hard and uncompromising questions, especially on my approach to circuit modelling.

Contents

List of Figures

List of Tables

Nomenclature

2D	Two-Dimensional.
3D	Three-Dimensional.
b_C	Stewart-McCumber parameter.
CAD	Computer-Aided Design.
CLB	Configurable Logic Block.
CMOS	Complementary Metal-Oxide Semiconductor.
COSL	Complementary Output Switching Logic.
CPU	Central Processing Unit.
dBm	Decibel (referenced to 1 mW).
dc	Direct Current.
DCRL	DC-Resettable Latch.
DRO	Destructive Read-Out register.
EM	Electromagnetic.
f	femto = 10^{-15}.
FDTD	Finite Difference Time Domain.
FPGA	Field-Programmable Gate Array.
FTP	File Transfer Protocol.
GP	Ground Plane.
HUFFLE	Hybrid Unlatching Flip-Flop Logic Element.
IC	Integrated Circuit.
I_C	Critical current of Josephson junction.
J_C	Critical current density of Josephson junction (normally in A.cm^{-2}).
JTL	Josephson Transmission Line.
k	Coupling coefficient for mutual inductance.
L	Confidence interval of Monte Carlo analysis.
LSB	Least Significant Bit.
LUT	Lookup Table.
m	micro = 10^{-6}.
m_0	Permeability of free space ($4\pi\times10^{-7}$ H/m).
MC	Monte Carlo.
Mo	Molybdenium (US spelling: Molybdenum).
MOM	Method of Moments.
MOSFET	Metal-Oxide Semiconductor Field Effect Transistor.
MSB	Most Significant Bit.
n	nano = 10^{-9}.
Nb	Niobium.
l	London penetration depth of a superconductor.
l_{eff}	Effective penetration depth of a thin-film superconductor.
j	Gauge-invariant phase difference across a Josephson junction.
Φ_0	Magnetic flux quantum (2.07×10^{-15} Wb).
p	pico = 10^{-12}.

PFD	Programmable Frequency Divider.
PGA	Programmable Gate Array.
PLD	Programmable Logic Device.
RAM	Random Access Memory.
rf	Radio Frequency.
RSFQ	Rapid Single Flux Quantum.
RSJ	Resistively-Shunted Junction.
s	Standard deviation of a Gaussian random variable.
SFQ	Single Flux Quantum.
SPGA	Superconducting Programmable Gate Array.
Spice	Simulation Program with Integrated Circuit Emphasis.
SQUID	Superconducting Quantum Interference Device.
SRFF	Set-Reset Flip-Flop.
T_C	Critical temperature of a superconductor.
TFF	Toggle Flip-Flop.
TRL	Through-Reflect-Line (calibration technique).
TTL	Transistor-Transistor Logic.
VLSI	Very Large Scale Integration.
y	True statistical yield of Monte Carlo analysis.
y'	Observed yield of Monte Carlo analysis.
Z_0	Characteristic impedance (of transmission line).

Chapter One - Introduction

THE design of superconducting programmable gate arrays requires more than just a basic understanding of PGA theory. It also requires a reliable set of tools for the design, modelling, optimization, and layout of superconducting circuits. This dissertation attempts to combine the key aspects of such a design process into a coherent whole, and to provide the engineer with a tool kit for the realization of SPGAs.

1.1 SUPERCONDUCTING LOGIC FAMILIES

During the last decade the field of superconducting logic circuits has been dominated by the Rapid Single Flux Quantum (RSFQ) family. A review of the theory behind RSFQ, as well as an introduction to RSFQ logic gates and design aspects can be found in the seminal article on the subject by Likharev and Semenov [1].

Several popular RSFQ gates have been available on the Stony Brook University web site [2] for years.

The RSFQ research effort at the University of Stellenbosch has been summarized in [3].

The most important difference between RSFQ and other logic families is that the former is a pulse logic family. Data, or digital bits, are represented by the presence or absence of picosecond voltage pulses. When integrated over time, these pulses are exactly one fluxon in size.

Several lesser known families also exist. One of these, Complementary Output Switching Logic (COSL) [4] [5] [6] [7], is used almost exclusively at Berkeley and Stellenbosch. It is a voltage state logic family, and as such ideal for interfacing the picosecond pulses of RSFQ logic to voltage state semiconductor families.

These two logic families, and especially RSFQ, are used to implement all the logic circuits for this dissertation.

1.2 ELECTRICAL SIMULATION TOOLS, MODELS AND EQUATIONS

All electrical circuit simulations in this dissertation are performed with *WRSpice* [8]. The Josephson junction is always implemented with the RSJ model, in which the basic junction is connected in parallel to a voltage-dependent resistor and a junction capacitance. The implementation of such a *Spice* model is also treated in [9].

When SFQ input pulses are not generated by DC-to-SFQ converters, they are simulated by piece-wise linear voltage sources. Such an SFQ input pulse is constructed as a triangular voltage waveform with a base length of 5 ps and a height of 824 *m*V, so that it integrates to one fluxon.

All circuits developed in this dissertation operate at 4.2 Kelvin. They are also by design reprogrammable, so that bit-error rate is not important at this stage of development. Noise was therefore not modelled in simulations for performance evaluation or optimization.

For the damping of all RSFQ junctions (with the exception of single series junction in the DCRL), the equation [1]

$$b_C = \left(\frac{2e}{\hbar}\right) I_C R_{ef}^2 C \ , \tag{1.1}$$

with

$$R_{ef}^{-1} = R_n^{-1} + R_S^{-1} \tag{1.2}$$

was used.

Here, R_n is the normal resistance of the Josephson junction (16.47 Ω for a 100 *m*m^2 junction in niobium), and R_S is the impedance of the environment connected to the junction. For RSFQ circuits, b_C = 1.

1.3 SUMMARY OF DISSERTATION

This dissertation starts in Chapter 2 with a discussion on genetic algorithms as applied to the optimization of superconducting logic circuits. An optimization program was developed to implement the genetic algorithms, and algorithmic flow charts and screenshots thereof appear in Appendix C. Results obtained with the genetic optimizer are presented in Chapter 2. The compilation of Monte Carlo simulation models [9] is also treated, and the extraction of tolerances from actual layouts is explained.

Chapter 3 contains a thorough investigation into the construction, simulation and verification of 3D models for inductance extraction. The effects of segmentation and reflection plane placement are also discussed, and some well-known structures are analysed. A program was developed to generate the 3D segment models used for inductance calculations, and the construction techniques used therein are detailed in Chapter 3. The chapter ends in an analysis of the true inductance spreads of Hypres circuits through the inclusion of actual design tolerances in the dimensional parameters of the 3D models.

After the treatment of design and layout considerations in the first chapters, Chapter 4 starts with a discussion on the need for novel components to add reprogrammable functionality to RSFQ circuits. The DC-Resettable latch, Current-Set switch, RSFQ-to-COSL converter and HUFFLE are treated – from design, optimization and simulation to layout verification.

In Chapter 5, the novel circuits are used in conjunction with standard RSFQ gates to construct a programmable frequency divider based on the architecture of FPGAs. This is the first step towards the development of a full superconducting programmable gate array, and allows most of the components, circuit configurations and programming structures needed for an SPGA to be tested. The emphasis in this chapter is mainly on developing routing structures and switch blocks, as well as perfecting the access of a low clock frequency controller circuit to the programmable switches.

Since the SPGA-based programmable frequency divider is a large and complex circuit, a smaller PFD was developed to allow fast and easily verifiable testing of the novel latches developed in Chapter 4. The design and simulation of the compact PFD is detailed in Chapter 6. A discussion on layout aspects is also included in this chapter.

In Chapter 7 the remaining considerations for the implementation of a full SPGA are discussed. Using the tools, gates and latches developed in this dissertation, a lookup table programming circuit and an address decoder are designed conceptually. These circuits are also simulated, and some problems regarding their implementation are identified and solved.

The Appendices contain circuit layout masks, circuit diagrams for some of the standard RSFQ logic gates and latches (if they were used in the dissertation), simulation files, flow charts and screenshots for the genetic optimization program, and a collection of 3D models for inductance calculation.

Chapter Two - Circuit optimization through genetic algorithms

This process, no matter how much we intervene in it, is essentially out of our control. Genes mutate, creatures evolve: a new biosphere emerges, and with it a new noosphere. And eventually the designer's minds, along with everything else, have been forever changed.

Kim Stanley Robinson, Green Mars

2.1 INTRODUCTION

NOVEL logic devices in superconducting logic families are normally suboptimal, and need optimization before they can be entered into cell libraries or circuit layouts. Even existing gates and latches need reoptimization when they are mapped to new fabrication technologies, as junction parameters and manufacturing tolerances differ.

Since an integral part of this project is the development of novel gates for the programmable logic circuits, an effective optimizer is needed.

The standard approach to optimizing Rapid Single Flux Quantum (RSFQ) circuits [1] is to maximize the critical margin [10]. A popular technique involving critical margins, and often used for RSFQ circuits, utilizes inscribed hyperspheres [11]. Complementary Output Switching Logic (COSL) circuits, on the other hand, are normally optimized for yield [5] [4] [9], but this merely involves manual tuning of proven sensitive elements such as the input Josephson junction, bias resistor and output dc SQUID [12].

Theoretical circuit yield is determined through Monte Carlo simulations [9], although the classical argument for avoiding yield-based optimization concerns the time required to perform such Monte Carlo simulations with acceptable uncertainty values [11]. However, faster personal computers and simulation software have eroded this barrier, and good results were obtained with yield-based optimization in this project.

Genetic algorithms [13], [14] were utilized because they are well suited to the optimization of complex problems with many parameters, as well as large solution spaces with many local maxima [15].

In this chapter, the implementation of a circuit optimizer based on genetic algorithms is discussed. Algorithmic flow charts for the genetic optimization sequence are shown in Appendix C. Screenshots of the optimization program, developed as part of this dissertation, are also shown in Appendix C.

Results with genetic optimization are also compared to those obtained with random optimization. Both techniques use the theoretical yield derived from Monte Carlo simulations as a direct measure of circuit performance, so that Monte Carlo methods and models are also discussed.

Lastly, margin analysis as an alternative fitness evaluation technique is also explored.

2.2 MONTE CARLO PARAMETERS AND PROCEDURES

Gott würfelt nicht. (God does not play dice.)
Albert Einstein, Creator and Rebel

God not only plays dice. He also sometimes throws the dice where they cannot be seen.
Stephen W. Hawking, Nature, 1975

Einstein would turn over in his grave. Not only does God play dice, the dice are loaded.
Blurb on the discovery of probability mechanics, Sid Meier's Alpha Centauri

Circuit optimization is dependent on reliable information about circuit quality. In this dissertation, circuit quality is expressed as the theoretical yield.

The Monte Carlo analysis has become the de facto technique for the determination of theoretical circuit yield, thereby superseding the less effective technique of margin analysis. Since it is used for the yield calculations and circuit fitness determination discussed here, it needs to be characterized.

After every Monte Carlo simulation, the time-domain output vectors (normally voltage against time, containing digital information as voltage levels or pulses), are evaluated according to a set of predefined constraints. If any output bit is wrong, the simulation is flagged as a failure.

After an entire Monte Carlo analysis is completed, the observed yield is found by dividing the number of functioning circuits through the total number of simulations. The uncertainty interval is calculated for a confidence level of 99 %, as described in [5].

If the observed yield for N Monte Carlo cycles is y', then the uncertainty interval is given by

$$L = 2.6\sqrt{\frac{y'(1-y')}{N}} \ , \qquad (2.1)$$

so that statistical yield y is

$$y = y' \pm L \ . \qquad (2.2)$$

2.2.1 *Uniform test setup*

The theoretical circuit yield calculated through a Monte Carlo analysis is a very good figure of merit. However, in order to allow us to compare the yield results of different circuits, we need to ensure a uniform test setup.

Additionally, information on the performance of each circuit in relation to another is needed in practical circuit optimization and the design of larger systems. Since RSFQ circuits dynamically load each other, they need to be characterized or optimized for all possible interconnections.

The only practical way of ensuring circuit interconnection compatibility is to characterize and optimize circuits for connection to a standard dynamic load. In RSFQ circuits, this standard load is a JTL. For the 1 kA/cm^2 process from Hypres, the standard JTL uses 250 *m*A junctions.

For every Monte Carlo analysis or optimization run, the device under test is connected to 250 *m*A JTLs at every SFQ input and output. In the event that a larger input current is needed (as for the T1 flip-flop and DCRL), or if output currents are above or below standard, the appropriate inputs and outputs are connected through current matching JTLs. Each current matching JTL amplifies or attenuates current by a factor of 1.42 [1].

The use of standard loads for every Monte Carlo analysis also ensures a uniform test setup, even though these standard loads are also subjected to parameter variations. It can well be asked if it is really important that the load devices in a yield simulation should also be varied with manufacturing tolerances. Here no clear rule exists, but since any gate will in practice be connected to dynamic loads that are themselves subject to tolerance effects, these variations are included in all simulations for the determination of circuit yield.

COSL gates use resistive interconnections, and test beds comprised of piece-wise linear voltage sources and resistors are sufficient.

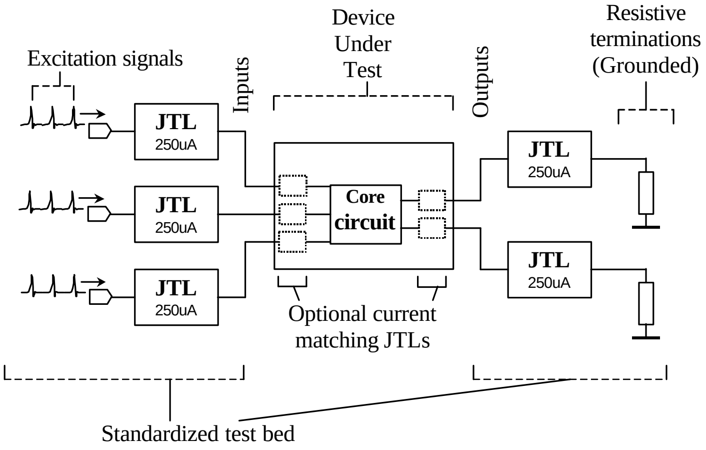


Figure 2.1: Standard test setup for Monte Carlo analysis of RSFQ circuits

2.2.2 Monte Carlo models

Monte Carlo models can be divided into two categories. The first contains the initial models. These are generic in composition, incorporating the known tolerances (global and local) for the design process, and are used for optimization and to predict circuit yield.

The second category contains the more accurate models, which are derived through parameter extractions from the circuit layouts. These models incorporate the actual layout values of and local tolerances specific to each element, and are used to evaluate layout quality.

The latest Hypres layer process specifications are shown in Table 2.1 (adapted from [16]).

The initial tolerance values – both global (chip-to-chip) and local (element-to-element) – for circuits fabricated with the Hypres process are shown in Table 2.2. Global tolerance results from layer thickness variations, and local tolerance from element width or junction area variations.

A Monte Carlo simulation file needs to model both the global and local tolerances. Global tolerance parameters are declared only once, and remain constant in a given circuit. Local tolerance parameters are assigned a new random value every time they are called. Separate models are created for each Josephson junction, so that current density is independent of area and can be varied between junctions. The local variation in junction capacitance does not need to be set, since it depends on the area of the junction alone.

Table 2.1: Hypres layer process specifications

Layer	Bias (wafer-mask) [*m*m]	Tolerance [*m*m]	Physical properties	Thickness [nm]	Deviation [nm]
M0	0.25	± 0.25	Nb, $l_L \cong 112$ nm[*] ± 5 %	100	± 10
I0	0.3	± 0.25	SiO_2, C = 0.277 fF/μm^2 ± 20 %	150	± 15
M1	-0.3	± 0.25	Nb, $l_L \cong 100$ nm[*] ± 5 %	135	± 10
I1A	**	–	–	45	± 5
SiO_2	–	–	C = 0.416 fF/μm^2 ± 20 %	100	± 10
R2	0.2	± 0.25	Mo, R = 1.0 Ω/square ± 20 %	100	± 20
SiO_2	–	–	C = 0.416 fF/μm^2 ± 20 %	100	± 10
I1B	0.2	± 0.25	Hole through above two SiO_2 layers		
M2	-0.5	± 0.25	Nb, $l_L = 90$ nm ± 5 %	300	± 20
SiO2	–		C = 0.08 fF/μm^2 ± 20 %	500	± 40
I2	0.2	± 0.25	Hole through above insulator		
M3	-0.75	± 0.25	Nb, $l_L = 90$ nm ± 5 %	600	± 50
R3	0.0	± 1.0	Ti/Pd/Au, R < 0.1 Ω/square	350	± 60

[*] Effective penetration depth – thin-film correction factor included

[**] $I_c = j_c(A - A_{mis})$, with $A_{mis} = 3.0 \pm 0.5$ *m*m^2

Table 2.2: Tolerance values for Hypres 1kA/cm^2 niobium process

Parameter	Junction critical current density (J_C)	Junction Area	Resistance	Inductance	Junction Capacitance
Global tolerance	10 %[*]	-	20 %	10 %	5 %[*]
Local variation	5 %[*]	5 %[**]	5 %	15 %	***

[*] Tolpygo [17]

[**] Layout variations: only certain values are possible due to grid-snap requirements

[***] Coupled to area variation

An example of an initial Monte Carlo simulation file for *WRSpice* [8] is shown in section B.1.2. Implementation may differ in other programs.

The more exact Monte Carlo analysis derived from parameter extraction, and referred to here as the layout model, is slightly more complex (see section B.1.3 for a *WRSpice* deck file example).

In the layout model, each circuit parameter can be assigned an individual standard deviation calculated from known fabrication tolerances or simulated parameter variations (see section 3.9 for a discussion on simulated inductance variations). Global variations must be retained, since they are the same for all parameters on a chip.

The random variables used to generate the global variations of resistance and critical current are kept exactly as in the initial model. For inductance, global variables are declared for each niobium layer, and the normalized standard deviation for each is calculated from a hundred or more simulations on at least one representative structure in the layer.

For resistance (*R*), the local variation decreases as line width (*W*) increases. The well-known equation for resistance is

$$R = \frac{rL}{W} \ , \tag{2.3}$$

where L is the length of the resistor, and r the sheet resistance in Ω per square.

The incremental change in resistance is larger for an absolute decrease in line width than for the same increase, so that the value of resistance caused by the worst-case width deviation $d_{W,\max}$ is

$$R_{\max} = \frac{rL}{(W - d_{W,\max})} \ . \tag{2.4}$$

The $3s$ variation of a particular resistor is then calculated from

$$\frac{R_{\max}}{R} = \frac{W}{(W - d_{W,\max})} \cong 1 + 3s_{\text{resistance}} \ . \tag{2.5}$$

From (2.5) the local variation ($3s$) of a resistor with width 5.8 *m*m is calculated as 4.50 %, compared to 5.49 % when the width is 4.8 *m*m.

A similar derivation yields the variation in junction area, except that the largest incremental change is for a decrease in junction area, which gives a smaller I_C. Since junction area equals critical current (I_C) divided by current density (J_C), the area variation can be expressed as

$$\frac{I_{C,\min}}{I_C} = \frac{I_C - J_C d_{A,\max}}{I_C} \cong 1 - 3s_{\text{area}} \ , \tag{2.6}$$

where $d_{A,\max}$ is the maximum uncertainty in area, as specified in the process design rules.

Equation (2.6) gives the $3s$ local variation of a 100 *m*A junction in the 1 kA/cm^2 Hypres process, with $\left| d_{A,\max} \right| = 0.5$ *m*m^2, as 5 %. For a 250 *m*A junction in the same process, the $3s$ variation is only 2 %.

The local variation in inductance is more complicated, and is best estimated from observed parameter spreads obtained with computer simulations.

If we define a Gaussian random variable with mean *m* and variance s^2 as

$$X \sim N\left(m, s^2\right) , \tag{2.7}$$

and model the random global and local variations of inductance as normalized Gaussian distributions, we can write

$$X_{g'} \sim N\left(1, s_{g'}^2\right) \tag{2.8}$$

and

$$X_{l'} \sim N\left(1, s_{l'}^{2}\right) . \tag{2.9}$$

In (2.8) and (2.9), $s_{g'}$ is the standard deviation caused by the layer variations, and $s_{l'}$ is that resulting from line width tolerance.

The standard technique for computing the random value of inductance (which is the same as that for resistance and junction area), is through a multiplicative model whereby it is declared as a function of $X_{g'}$ and $X_{l'}$,

$$Y = aX_{g'}X_{l'} , \tag{2.10}$$

where a is the mean value of Y.

Although (2.10) is a very accurate description of what happens on a real chip, the model has a shortcoming.

When layout extraction is performed, the variation of inductance as a result of both global and local tolerances can be determined from numerical simulations (see 3.9, p.51). Since the parameter for global variation – which can be established through similar simulations when conductor widths are held constant – has to be the same for all inductors in a layer, it is only determined once. Simulations also show that Y has a Gaussian distribution, with observed standard deviation s_{obs}. For an inductance value in a Monte Carlo circuit simulation to have a standard deviation of s_{obs}, we need to calculate a $s_{l'}$ that, together with $s_{g'}$, will ensure the correct distribution. Calculation of $s_{l'}$ does not readily follow from (2.10), so that we need a simpler model.

We can start by redefining the global and local random variables (2.8) and (2.9) to have zero means and name them $X_{g''}$ and $X_{l''}$. Now (2.10) becomes

$$Y = a(1 + X_{g''})(1 + X_{l''}) , \tag{2.11}$$

which expands to

$$Y = a(1 + X_{g''} + X_{l''}) + aX_{g''}X_{l''} . \tag{2.12}$$

Equation (2.12) shows that, when zero-mean global and local variables are used, Y can be described in terms of the sum of the independent global and local variables. The last term in (2.12) is very small if the standard deviations of $X_{g''}$ and $X_{l''}$ are much smaller than 1, as is indeed the case in the Hypres process. It can therefore be neglected.

The proposed alternative is therefore an additive model, in which the random variable for inductance (Y) is defined in terms of the sum of statistically independent global (X_g) and local (X_l) variations, so that

$$Y = a\left(1 + X_{g} + X_{l}\right) . \tag{2.13}$$

The mean of Y therefore remains a if X_g and X_l have zero means.

$$X_{g} \sim N\left(0, s_{g}^{2}\right) \tag{2.14}$$

and

$$X_l \sim N\left(0, s_l^2\right) \ . \tag{2.15}$$

Y is produced by a linear transformation of two Gaussian variables, and is therefore also Gaussian. (The proof appears in almost any text on probability theory. See for example [18, p. 147-149]).

$$Y \sim N\left(a, s_{obs}^2\right) \tag{2.16}$$

For completeness, the mean of *Y* is calculated from

$$\overline{Y} = E[a] + E[aX_g] + E[aX_l] = a + aE[X_g] + aE[X_l] = a \ . \tag{2.17}$$

The variance of *Y* equals its second central moment, or

$$s_{obs}^2 = m_2 = E[(Y - \overline{Y})^2] \ , \tag{2.18}$$

which simplifies to

$$s_{obs}^2 = E[Y^2] - \overline{Y}^2 \ . \tag{2.19}$$

The square of (2.13) yields

$$Y^2 = a^2(1 + 2X_g + X_g^2 + 2X_l + X_l^2 + 2X_g X_l) \ . \tag{2.20}$$

The mean of (2.20) is

$$E[Y^2] = a^2 + 2a^2E[X_g] + a^2E[X_g^2] + 2a^2E[X_l] + a^2E[X_l^2] + 2a^2E[X_g X_l] \, . \tag{2.21}$$

The second and fourth terms on the right hand side of (2.21) are zero, as defined in (2.14) and (2.15). The last term describes the correlation between X_g and X_l. Since they are defined to be statistically independent, we can write

$$2a^2E[X_g X_l] = 2a^2R_{X_g X_l} = 2a^2E[X_g]E[X_l] = 0 \, . \tag{2.22}$$

With (2.22) and the definitions in (2.14) and (2.15) substituted into (2.21), the latter reduces to

$$E[Y^2] = a^2 + a^2E[X_g^2] + a^2E[X_l^2] \ . \tag{2.23}$$

The second central moment of X_g can be rearranged in terms of $E[X_g^2]$ so that

$$E[X_g^2] = s_g^2 + \overline{X} = s_g^2 \tag{2.24}$$

A similar equation can be obtained for $E[X_l^2]$, and if substituted into (2.23) along with (2.19) and (2.24), it yields

$$s_{obs}^{2} + a^{2} = a^{2} + a^{2} s_{g}^{2} + a^{2} s_{l}^{2} \tag{2.25}$$

or

$$s_{l} = \frac{\sqrt{s_{obs}^{2} - a^{2} s_{g}^{2}}}{a} = \sqrt{\left(\frac{s_{obs}}{a}\right)^{2} - s_{g}^{2}} \; . \tag{2.26}$$

With (2.26), the standard deviation of the local variation of each inductor in a circuit model can be calculated such that it will, in conjunction with the global variation, give the exact distribution of the observed inductance found from layout extraction. It is important to realize that X_g models the global deviations as caused by layer thickness variations, but that X_l is a function of local variations in conductor width *as well as* any other variations that, together with the global variations, will produce the observed random variable *Y*.

More advanced simulation models derived through parameter extraction utilize transmission line elements to include the effects of distributed capacitance and transmission delays in the Monte Carlo models [19]. The technique is useful for detecting the diminished yields caused by bad layouts, or clock period timing violations resulting from slow or long signal paths. Since it is much slower than conventional Monte Carlo simulations and depends on information about the layout, it was not considered for fitness evaluation in the genetic optimization algorithms.

2.2.3 Trimming

Trimming can improve circuit yield [5] [9] [7]. Though not used during optimization, when information on the worst performance of a circuit is required, it is standard to include trimming when the best expected yield figures for full system circuits are quoted.

The justification for the inclusion of trimmed figures in full circuits is that the bias voltage of a non-functional circuit can be varied to try and make it work.

In COSL circuits, trimming is performed by adding or subtracting current to an input rf SQUID. The current is supplied by a voltage applied over a 50 Ω resistance. Modelling of this trim voltage during Monte Carlo simulations requires the calculation of the effective inductance of the rf SQUID at the start of every Monte Carlo run. The inductance of the Josephson junction is included in this calculation, which is discussed in detail in [7].

For RSFQ circuits, trim is applied to the dc bias voltage. The objective is to cancel the effect of global tolerances on the current bias of grounded Josephson junctions.

A global increase in resistance leads to a directly proportional reduction in the bias current of each junction. This can be counteracted if the dc bias voltage is increased by the same proportion. Additionally, a global increase in the current density of Josephson junctions results in a directly proportional increase in critical current, and reduces the proportion of bias current to critical current. This can also be negated by a proportional increase in the dc bias voltage.

The resulting dc bias applied to any trimmed RSFQ circuit during Monte Carlo simulation, $V_{biastrim}$, is:

$$V_{biastrim} = V_{dcbias} \times R_{tol} \times J_{tol} \tag{2.27}$$

Where:

V_{dcbias} = the nominal dc bias, or 2.6 mV in our circuits

R_{tol} = the normalized multiplier for modelling global resistance change

J_{tol} = the normalized multiplier for modelling global current density change

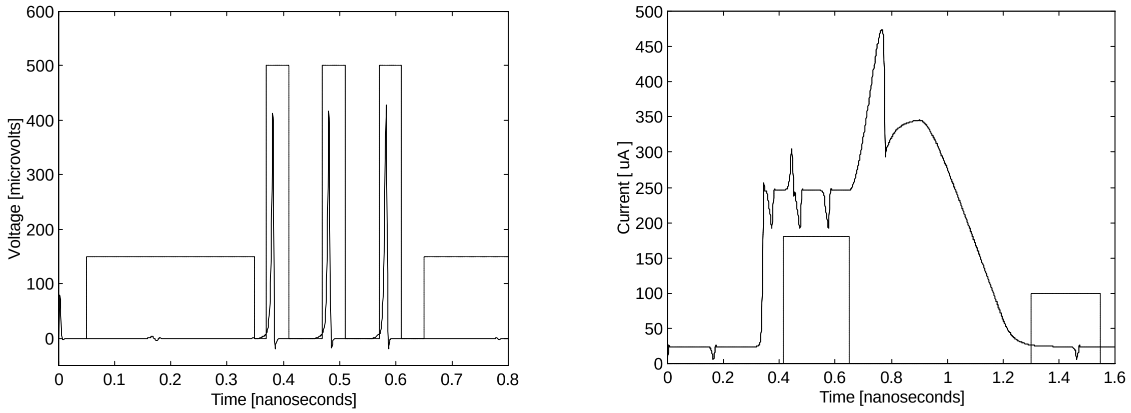
The implementation of voltage trimming in a *WRSpice* circuit file can be seen in section B.1.2.

2.2.4 Failure detection

In this game that we're playing, we can't win. Some kinds of failure are better than other kinds, that's all.
George Orwell, Nineteen Eighty-Four

During a Monte Carlo analysis, hundreds or thousands of simulations, each with random variations on all the elements, are performed on a circuit. The statistical or theoretical yield is then calculated from the observed number of correct simulations, as given by (2.1) and (2.2).

In RSFQ circuits, the voltage output vector of a simulated circuit is integrated over a time window, and only if the integrated value for an expected pulse falls between 1×10^{-15} and 3×10^{-15} V.s (it should be one magnetic fluxon, or 2.07 fWb), is the circuit flagged as correct [20]. Higher values indicate double pulses (a failure), and lower values indicate the absence of a pulse. The evaluation windows are shown (not to vertical scale) in Figure 2.2 (a), and a high window indicates that a pulse is required, whereas low windows indicate that no pulses are allowed. The width of the time windows are also used as constraints, since a logic pulse has to appear within a specified time window in order to satisfy clocking requirements. Slow circuits too are rejected as failures.



KEY: Latch output voltage [——], evaluation windows [– – –]

(a)

KEY: Flux loop current [——], evaluation constraints [– – –]

(b)

Figure 2.2: Evaluation constraints for RSFQ DC-Resettable latch: (a) nominal voltage output vector and (b) nominal flux-loop current

When a current vector is evaluated (to determine flux through a loop), it is averaged over time windows. In Figure 2.2 (b), the current average in the first window must be more than 180 *m*A, corresponding to a set state, whereas the average in the second window must be less than 100 *m*A.

In COSL circuits, the voltage output vector is averaged over time windows. These windows correspond to the clock phase on which a gate output stage operates, and the average voltage must fall within a high or low voltage range, depending on the logic state. The average value of a high output voltage must fall between 800 *m*V and 1.25 mV for one fifth of a clock cycle (denoting the high voltage state). The average value of the low output voltage must remain below 400 *m*V over any one-fifth period of the clock cycle.

2.2.5 *Closed-loop design*

Monte Carlo yield analysis and optimization are repeated as many times as is necessary to yield a reliable circuit that can be constructed with the chosen fabrication process.

Layout, parameter extraction, yield analysis and optimization form a closed-loop cycle that is repeated until a layout is realized that has the desired characteristics.

Some physical limitations that can often only be determined accurately during layout, and can force design changes, include:

- Minimum attainable parasitic inductance.
- Minimum realizable junction area.
- Minimum interconnection distance (affecting inductance).
- Maximum attainable inductive coupling.

These limitations are often factored into the affected element values, or added as extra parasitic elements to circuit models after they show up during layout extraction.

In the entire closed-loop design process, from initial design to final layout, Monte Carlo analyses play a prominent role.

2.3 MARGIN ANALYSIS

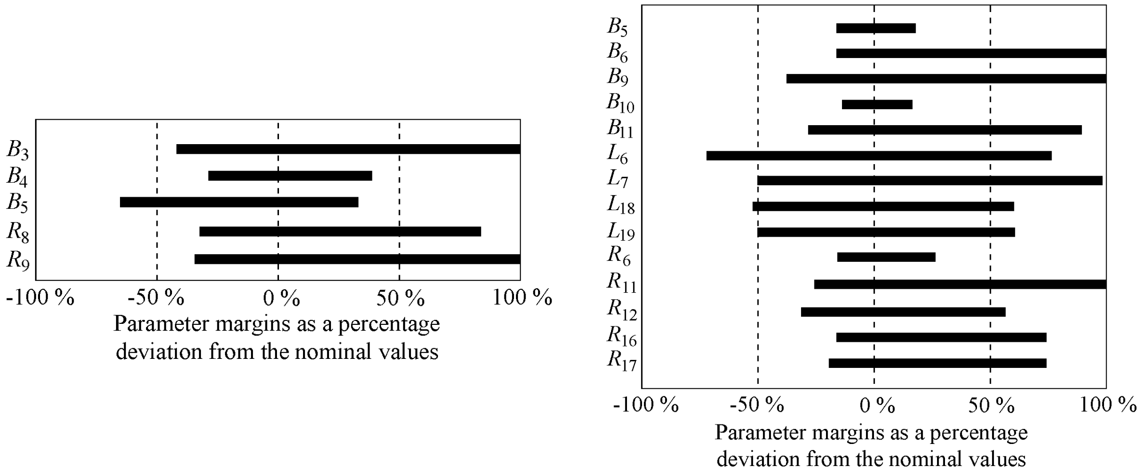
Margin analysis is an old technique used to obtain qualitative information on the effects of parameter variations on circuit operation [10]. It was used extensively when Monte Carlo analyses were still too time-consuming to run repeatedly on a given circuit optimization problem.

Implementation is fairly easy. In a nominal circuit, one element value at a time is changed, a simulation is run, and the output evaluated to determine whether a failure occurred. If not, the element value is adjusted by a larger factor and a new simulation is performed. When a circuit fails, the value of the element under analysis is set exactly halfway between the failed value and the last functioning value.

The process is repeated until the limit value for the given element that denotes the transition from a working circuit to a failure is found. Upper and lower limits are thus found for every element value.

As an example of the use of a margin analysis, compare the results for the HUFFLE and Current-Set switch in Figure 2.3(a) and (b).

Only elements with margins of around ±50 % and less are shown (the elements correspond to the designations in Figure 4.4 and Figure 4.9).



(a) **(b)**

Figure 2.3: Margin analysis results for the most critical parameters of (a) the Current-Set switch and (b) the HUFFLE

For the Current-Set switch, the critical margin is determined by B_4, and for the HUFFLE by B_{10}. It appears that the Current-Set switch is more stable than the HUFFLE (which was confirmed through MC analyses). The most critical elements can now be identified.

The critical margin in RSFQ circuits is often found to be the critical current of a Josephson junction, as with the OR, XOR and NOT-gates in [10], and the Current-Set switch and HUFFLE discussed here. In virtually all other cases, the bias resistors account for the critical margins (all the resistors shown in Figure 2.3(a) and (b) are bias resistors), which is probably why the specification of bias current margins as a measure of circuit performance has remained popular in publications for so long.

Fortunately, global variations account for the largest deviations in resistance, and this can easily be compensated for by trimming the global dc supply voltage. The same technique is used to limit the effect of the global variation of J_C.

It is important to note that the element responsible for the critical margin can change during circuit optimization, so that margin analyses under these conditions must check all suspect elements.

Apart from only giving information on elements in isolation (with a disregard for the effects that element variations have on the margins of others), margin analyses are also not very fast. For Monte Carlo simulations, 100 runs already deliver usable fitness information for optimization (although 441 – the number of runs when *CheckSTP*1 = *CheckSTP*2 = 10 (C.1) – was most often used as a minimum). With a margin analysis, between 15 and 20 runs are normally needed to give upper and lower margins to within 1 % for one element. When several elements are analysed for critical margins, and especially when the simulation engine does not have native margin analysis support, much more time is required than for a Monte Carlo analysis – which then is clearly the most practical method.

2.4 GENETIC ALGORITHMS

Conventional genetic algorithms operate on problems that have been reduced to binary strings [14] through a decoding function that maps the phenotype space (real-world parameters) to the genotype space [13]. These strings, called chromosomes or genomes, are able to reproduce, pair for crossover and undergo random bit mutation. The probability of reproduction is determined by circuit fitness. Pairing and crossover allow the exchange of genetic information, whereas mutation provides a way of introducing random jitter into solutions to prevent convergence on a single local optimum.

Real-valued parameters can also be represented by binary substrings in the chromosome [14, pp. 52-54] [21], as shown in Figure 2.4(a) [20], but this limits the range of the solution. Since the circuit parameters subject to optimization are the real-valued element values (resistance, inductance and critical current), it was opted to map them directly to a genome of real values. This is often differentiated from true genetic algorithms, and referred to as an *evolution strategy* [13]. However, the underlying theory is equivalent, and since the strategy parameters (mutation rate, mutation distribution, fitness curve, etc.) are not yet subjected to evolutionary change themselves, this technique is conventional.

Before the genetic algorithms can start operating, a parent generation that is distributed throughout the solution space is needed. This first generation is created through random variations of the nominal circuit. The entire generation is evaluated for fitness, where fitness is calculated from a cost function and models the probability of survival and reproduction of each individual. A new generation is then created from its ancestors by randomly copying individuals to the new population according to their probability of procreation. Next the resulting individuals are paired. In the implementation discussed in this chapter, all the individuals of a population are paired at random (as opposed to strategies where, for example, stronger individuals pair with weaker ones, or the strongest individuals pair with each other).

Elements from the genomes of each individual in a pair are allowed to exchange at random, unlike the standard practice of punctuated crossover. The probability of crossover is also chosen at random, but remains constant for all elements in a pair of genomes. After crossover, random mutations are allowed. The mutated values are calculated by multiplying the originals with a unity mean Gaussian distribution. The process is illustrated in Figure 2.4(b) [22]. Only two paired individuals are shown after reproduction. Element values contain no multiplier suffixes, and in the example each individual has two resistors, two inductances and two Josephson junction critical current values.

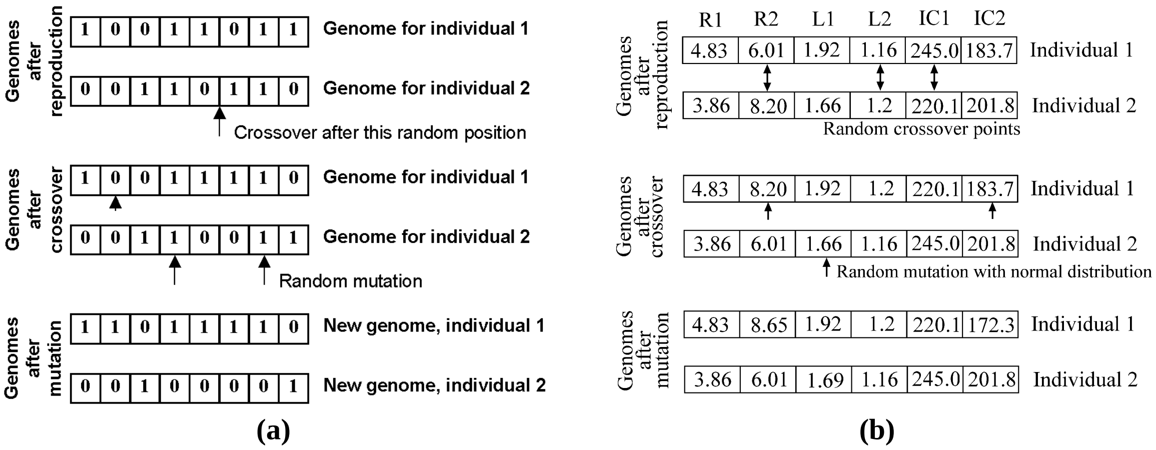


Figure 2.4: Graphical representation of the genetic optimization process using (a) binary string genomes and punctuated crossover and (b) real-valued genomes and random crossover

The new generation is then evaluated for fitness, and the reproduction process repeated. The cycle is continued until an individual with a prescribed yield is spawned, or until the optimizer stalls at some maximum yield.

2.5 FITNESS EVALUATION

Theoretical circuit yield, calculated through a Monte Carlo analysis, is used as an indication of circuit fitness [9]. Since the element variations in such a simulation are based on the manufacturing tolerances of a particular process, circuits are optimized directly for that process. The Hypres 3-micrometre fabrication process [16], for which the tolerances are modelled as Gaussian distributions (see Table 2.2 on p.7 for the list of values), was used for all designs.

2.5.1 Summary of evaluation constraints

Apart from the constraints discussed in section 2.2.4, which are used in all Monte Carlo simulations, other restrictions are necessary to prevent the optimizer from spawning circuits that might seem to work well when they are actually impractical.

- In RSFQ circuits, static current flow between logic gates should be as small as possible. When a circuit is optimized while connected to a JTL, it can evolve to source or sink current from the JTL through the inductive signal interconnection (bias current distribution is determined by the inductance of each zero-resistance path to ground). The JTL is an extremely robust circuit, and can withstand large static loading currents. However, when the offending gate is connected to another less stable circuit, the resulting loading can cause either or both to malfunction. In simulations, circuits are flagged as failures if the absolute value of any static current flowing through a gate interconnection exceeds a chosen value (no clear guidelines exist, so 10 % of the smallest junction critical current is used).
- Junctions that are too small cannot be manufactured, or may be inaccurate. In the Hypres 3-micrometre 1 kA/cm^2 process, it is not advisable to use junctions with a critical current of less that 100 *m*A [16].

2.5.2 Mapping theoretical yield to fitness

Circuit yield is always specified as a percentage, but when used directly as a fitness value it can lead to insufficient differentiation when the yield distribution for all circuits in a population lies between, for instance, 50 % and 70 %. Instead, an adapted scale is utilized, whereby the circuit with the lowest yield is assigned a fitness of 0, whilst the highest yield is translated to a fitness of 1. All other yield values are translated to fitness values between 0 and 1, according to a linear or quadratic function – a parameter set before optimization. The resulting probability distribution function is then normalized to integrate to 1, and used for offspring selection as previously discussed.

It is clear that the weakest individual in a generation is guaranteed to die off, but that the strongest individual does not necessarily survive. Furthermore, even if the strongest individual survives to create offspring, the only way in which an exact replica of its genome can enter the offspring population is if it pairs with itself (possible) and experiences no mutation during the process. In general, however, only genetic material of the strongest individual, and not an exact copy, is transferred to the next generation.

In a technique called elitism [21] the strongest individual is copied into the offspring population to ensure its survival, but it was not used here.

Critical margin analysis is often used as a figure of merit for superconducting circuits and their optimization [1] [10] [11], and could also function as a fitness indicator. However, this technique does not give a direct measurement of the circuit's yield, and gives a poor resolution when the circuit yield is very low. Yet it is useful for the identification of critical elements (see section 2.3).

2.5.3 Noise in yield-based fitness values

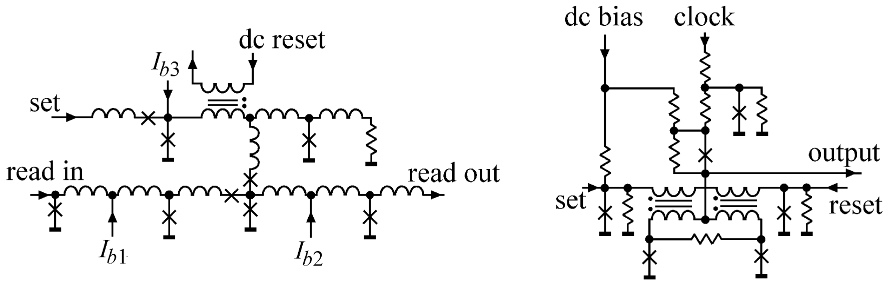
A drawback to fitness values based on yield is the noise inherent to Monte Carlo analyses. This can be reduced by increasing the number of simulations per Monte Carlo analysis (thereby reducing uncertainty), at the cost of increased computing time. Results show that genetic optimization can fall victim to this noise when yield approaches 100 %, as seen in Figure 2.7, or when the yield spread across a generation decreases to the uncertainty limits of the Monte Carlo analysis. The only way to break out of such a local optimum is to step up the rate and standard deviation of mutation, or force a large evolutionary jolt; either by generating a new set of circuits at random from the one with the highest yield, or by applying a manual tweak to a sensitive parameter. The results for the DC-Resettable latch show that the combination of genetic algorithms with large evolutionary shocks (random or manual) increases the effectivity of the optimization process.

2.6 RSFQ DC-RESETTABLE LATCH EXAMPLE

The genetic optimizer was first tested on a novel RSFQ DC-Resettable latch (Figure 2.5(a)), where 32 elements, excluding damping resistors or any parasitics, were optimized [22].

The first functional circuit model had a theoretical yield of only 22.3 ± 9.8 %. The optimization parameters were: population size = 100, mutation probability = 0.03, mutation distribution $(3s)$ = 0.1, fitness evaluation = quadratic, Monte Carlo runs = 225.

Junction damping resistors were automatically adjusted after every junction area alteration in order to set their Stewart-McCumber parameters equal to 1 [1].



(a) **(b)**

Figure 2.5: Simplified schematic circuit diagrams for (a) RSFQ DC-Resettable latch and (b) COSL Set-Reset flip-flop

Figure 2.6 (a) shows the optimization results for the first sequence, which ran for more than 2 weeks on an Intel Pentium III. The gaps at generations 3, 7 and 8 resulted due to data not being checked over weekends. The best offspring circuit had a yield of 55.1 ± 8.6 %.

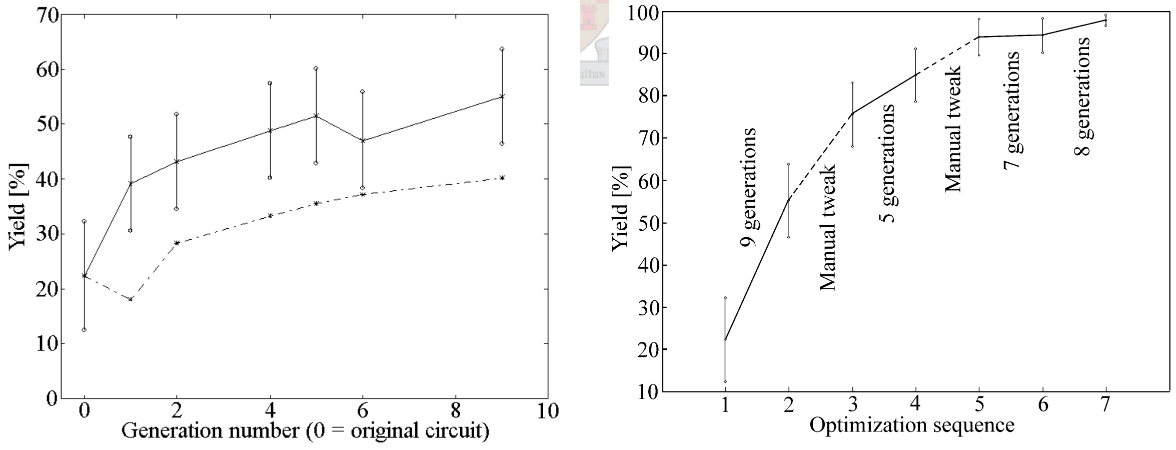
This best circuit was analysed manually, and observed setup failures were linked to the *Set* input inductor (L_1 in Figure 4.1). With this inductor tuned manually (emulating a large mutational jolt), the circuit yield leapt to 75.6 ± 7.5 %.

A second genetic optimization sequence was applied to the tuned circuit, with the parameters similar to those of the first optimization, except for: mutation probability = 0.04, mutation spread ($3s$) = 0.15, vary parasitics = no.

After five generations the yield was up to 84.9 ± 6.2 %. The optimization strategy was then changed to allow the optimizer to operate only on the set-reset stage of the latch. All the element values in the read section were thus locked. Within a few generations the set-reset stage yield reached 100 %, even with $3s$ variations of 30 % and 35 % respectively on the global and local inductance parameters.

The strategy was then reversed by locking the set-reset stage, and allowing the genetic algorithms access to the read stage elements only. Parameters changed from the first sequence were: Population size = 400, mutation probability = 0.05, mutation spread ($3s$) = 0.3, Monte Carlo runs = 361, global tolerance on inductance = 0.3, local tolerance on inductance = 0.35.

After around 8 generations, maximum yield stabilized and showed no further increase. With the Monte Carlo parameters set back to the generic Hypres values, the yield turned out to be 97.71 ± 1.25 %. This is good enough to justify the use of the DCRL in RSFQ circuits, especially since dc bias voltage trimming makes the practical yield better than the theoretical figure, even after the detrimental effects of layout imperfections are factored in (see Table 4.1, p.67). If necessary, feedback loops to counter accidental reset failures can also be introduced into sensitive circuits (section 4.2.1.3, p.58).



(a) **(b)**

Figure 2.6: Genetic optimization results for RSFQ DCRL: (a) yield vs generation number for first sequence and (b) yield vs sequence for entire process

Figure 2.6(b) shows the results for all the genetic optimization runs, as well as the number of generations for each run. The dashed lines represent manual adjustments made to the latch between optimization stages.

2.7 COMPARISON OF GENETIC AND RANDOM OPTIMIZATION METHODS

Genetic algorithms work for circuit optimization, and the results from section 2.6 confirm this. Yet, in order to know if it is worthwhile to use them, we have to compare their results to those obtained from a brute force random search [22].

The straightforward technique for random optimization is to vary the element values of a nominal circuit according to a chosen distribution. Yield is evaluated for each new circuit, and if it is higher than that of the nominal circuit, the new circuit becomes nominal and the process restarts.

The genetic optimizer was compared with the random technique for the optimization of a novel COSL Set-Reset flip-flop, which is shown in Figure 2.5(b). The genetic optimization parameters were: population size = 100, mutation probability = 0.05, mutation distribution ($3s$) = 0.3, fitness evaluation = linear, Monte Carlo runs = 441. The random optimizer also used a population size of 100, and $3s$ element variations of 0.3.

The first implementation of the COSL Set-Reset flip-flop had a yield of 33.1 ± 5.8 %. It was then subjected to optimization with the genetic and random algorithms. Both techniques levelled off after the seventh generation, with the genetic optimizer delivering a best yield of 86.2 ± 4.3 %. The random optimizer produced a best yield of 80.9 ± 4.9 %. The comparative results are shown in Figure 2.7. Note that the population average increases during genetic optimization as weak solutions die off, and also that the best solution died off (or had its best traits genetically diluted) between generations 2 and 3.

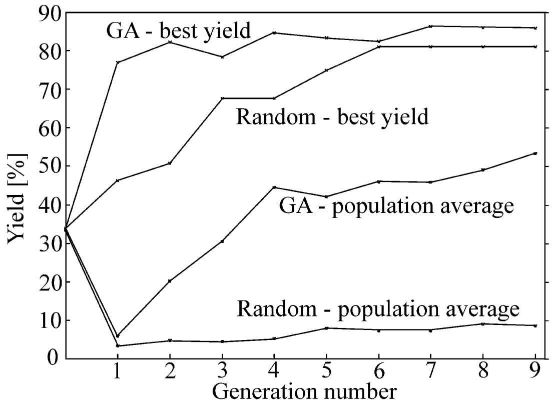


Figure 2.7: Comparative results for a genetic algorithm and random optimization sequence starting with the same unoptimized COSL Set-Reset flip-flop

The mutation distribution was then reduced to 0.1 for further optimization. The random optimizer stalled in a local optimum at 83.2 ± 4.6 %, whereas the genetic optimizer produced a circuit with a yield of 95.6 ± 3.6 % after 12 generations.

Margin analyses on the best result from the genetic and random optimizer show that the former has a critical margin of 12.2 % and the latter one of 8.4 %. While the circuit with better yield also has a better critical margin, it still does not give any indication of the actual yield. This does not disqualify the use of critical margin analysis for optimization, but as discussed in section 2.3, Monte Carlo analyses need fewer simulations.

A comparison between the element values of the best circuit from each optimization process is shown in Figure 2.8, where the elements of the genetic algorithm result are taken as nominal. The large differences in element values confirm that the solution space indeed has multiple local optima.

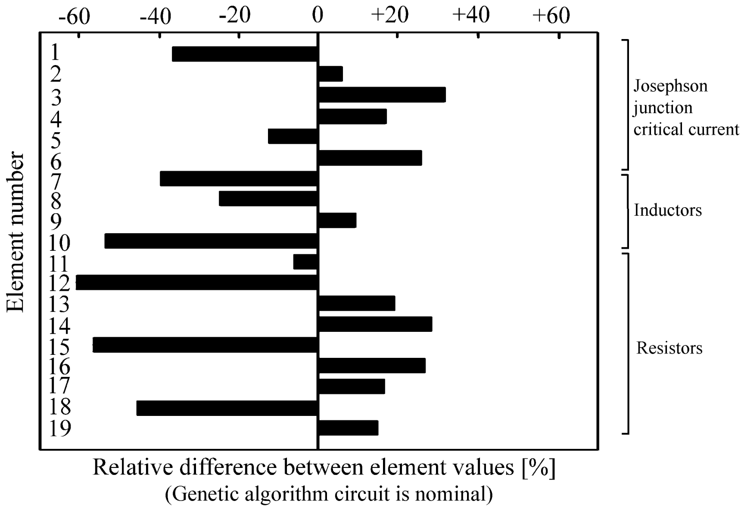


Figure 2.8: Relative difference between the elements of the best COSL Set-Reset flip-flop from the genetic and random optimization processes

In another test, both techniques were applied to the existing COSL negative output OR-gate [7]. With tight evaluation criteria and no voltage trimming the unoptimized yield was 58.7 ± 8.5 %. Optimization parameters were the same as for the COSL Set-Reset flip-flop optimization. Both techniques achieved a yield of 100 %, as shown in Figure 2.9, with the random optimizer defeating genetic algorithms by one generation. The resulting circuit needs no voltage trimming.

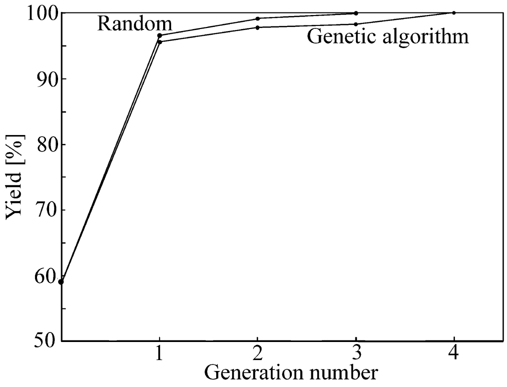


Figure 2.9: Comparative results for genetic and random optimization sequences starting with the same unoptimized COSL negative-output OR-gate

2.8 CONCLUSIONS

Genetic algorithms have been demonstrated to perform well during circuit optimization. Although random searches are efficient at finding several different effective solutions starting from a very suboptimal circuit (provided that enough random circuits are generated), the genetic technique inherits this quality by virtue of the fact that the first generation is spawned at random.

Genetic algorithms have several advantages [22].

- They are easy to implement, requiring minimum mathematical effort.

- Circuit fitness is based on the yield with real tolerance values, so that none of the unknowns of margin analysis remain.
- They readily scale to any N-parameter circuit, even though the size of the search space increases with larger N-values. Circuits with as many as 32 parameters have been optimized.
- They can search a solution space with multiple optima.
- They can optimize themselves if the strategy parameters are coded into the genome.

Genetic algorithms have been used successfully in many research fields [13], but still lack a sound mathematical description that allows designers to calculate the best optimization parameters. Consequently, most parameters are selected at random, and some selections cause genetic algorithms to perform little better, if not worse than random optimization. However, results obtained with genetic algorithms are definitely promising.

A Monte Carlo model based on layout extraction, but fast and easy to simulate in optimization procedures, was also developed to give a very accurate estimate of the yield of an actual circuit.

Chapter Three - Inductance extraction

I keep six honest serving-men
(They taught me all I knew);
Their names are What and Why and When
And How and Where and Who.
Rudyard Kipling, The Elephant’s Child

3.1 INTRODUCTION

INDUCTANCE is a property that is hard to calculate and even harder to estimate; yet it is very important in RSFQ circuits. For the layout of sensitive circuits, especially when manufacturing tolerances are large, the uncertainty in the inductance of conducting structures should be minimal.

For the layout of large circuits, inductance is often estimated from a two-dimensional algorithm [23]. Unfortunately, although fast and easy, this technique is only accurate for uniform microstrip lines over an infinite ground plane (Figure 3.2). The algorithm cannot even accommodate corners, so that corner inductance has to be approximated. The traditional approximation is to multiply the diagonal between the centres of incoming lines, called the effective path length (Figure 3.1) [24], with the per length inductance of the line [12]. However, this is problematic when the corner is not square and the per length inductances of the two arms differ.

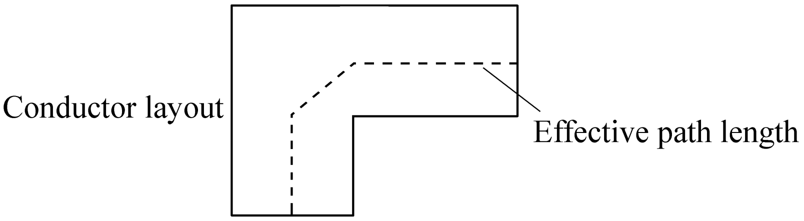


Figure 3.1: Traditional approximation of the effective path length for inductance estimation around a corner in a thin-film conductor

Recent results from numerical analyses suggest that the effective length is less than the popular approximation, and closer to half the side length of the corner square (see section 3.4.2 for a more detailed discussion).

Two-dimensional techniques are also unable to predict the inductance of common structures like vias, tees, Josephson junctions and double-cornered lines with short arms. Figure 3.3 shows some of these structures, with A and A′ as the current entry and exit points, and B the position where the conductor is electrically shorted to B′ on the ground plane.

A scheme for inductance calculation in complex thin-film superconducting structures has been discussed [25]. Guan *et al.* used a modified version of *FastHenry* [26], a program that allows the numerical extraction of the inductance of discretized structures. Tables were generated that list the per square inductance of several line structures with corners, tees and vias. These tables were then used to calculate the inductance of real layout structures. However, in the tightly packed interiors of logic circuits, the close proximity of other conducting structures can influence inductance. Conductors may also consist of several closely spaced corners or vias.

In this chapter, the need for a reliable inductance extraction technique, as well as the implementation thereof with an image-method numerical analysis, are discussed.

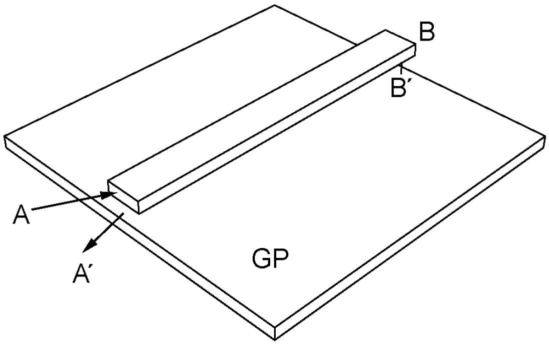
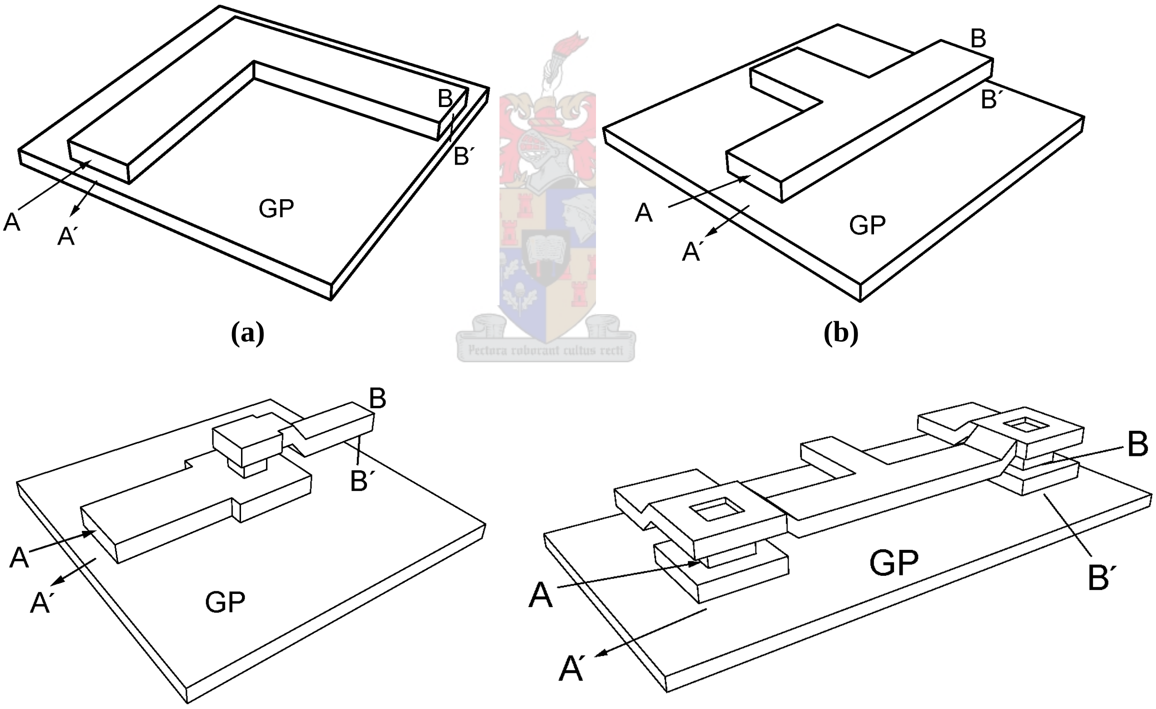


Figure 3.2: Uniform microstrip over ground plane



(a)

(b)

(c)

(d)

Figure 3.3: Common 3D structures requiring inductance calculations are the (a) cornered microstrip, (b) microstrip with tee-in, (c) via-connected microstrips and (d) microstrip connecting two Josephson junctions, including damping resistor covers and dc tee-in

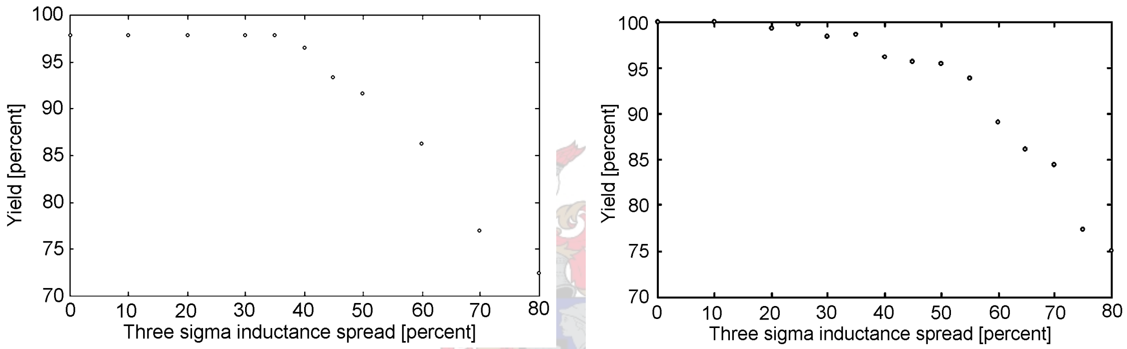
3.2 INDUCTANCE TOLERANCE AND THE EFFECT ON RSFQ CIRCUITS

For the Hypres process [16], the nominal values of resistance and Josephson junction critical current can be established quite accurately during layout. However, the inductance of an etched structure is much more difficult to predict, as detrimental effects caused by fringing, mutual inductance and irregular three-dimensional shapes are often impossible to calculate by analytical means.

Margin analyses often show large margins for inductors, although they can sometimes determine the critical margin, such as in the AND-gate of [10].

However, circuits are more sensitive to inductance than margin analyses imply. The main reason is that changes in inductance from the nominal design values lead to different static current flows in RSFQ gates, thereby affecting junction bias currents. This reduces the Josephson junction critical current margins that will result in switching failures – an effect that escapes detection in margin analyses but shows up through lower yields in Monte Carlo analyses.

As an example, consider the results in Figure 3.4(a) and (b). They were generated through repeated Monte Carlo simulations on the optimized versions of the HUFFLE and DC-Resettable latch (see the circuit diagrams in Figure 4.9 and Figure 4.1), using the generic tolerance model and voltage trimming. The global and local inductance tolerances were merged into a single random variable of which the 3*s* limit was swept from 0 to 80 percent.



(a) **(b)**

Figure 3.4: Yield as a function of inductance spread for (a) HUFFLE and (b) RSFQ DCRL

The Monte Carlo yield results for the HUFFLE in Figure 3.4(a) show yield to decline nearly linearly above a three sigma inductance variation of about 38 %, whilst the critical inductance margin for the same device in Figure 2.3(b) is 50 %. For the DCRL, the critical inductance margin is 42.2 %, although circuit yield starts to decline at a 3*s* inductance variation of about 25 % (Figure 3.4(b)). We can thus conclude that margin analyses do not give the full picture.

Table 3.1: Inductance offset of a 3 *m*m microstrip for worst-case variations in different layers of the Hypres niobium process

Layer	Thickness variation (Global)	Width variation (Local)
M1	+5.49% -4.97%	+7.74% -6.69%
M2	+5.51% -5.33%	+6.68% -5.87%
M3	+5.15% -5.11%	+5.21% -4.69%

Table 3.1 shows the theoretical worst-case inductance offset, due to manufacturing tolerances, of narrow inductors in all the layers of the Hypres process (estimated with

Chang's 2D algorithm [23]). Although lines in M1 can be as narrow as 2.5 *m*m, and those in M2 and M3 only 2 *m*m, these widths are rarely used in physical layouts.

For global variation, the worst case (lowest inductance) was taken as thickest ground layer, thinnest dielectric layer and thickest conductor layer. All parameters were reversed for highest inductance. The worst-case values are conservative – repeated simulations on discretized 3D structures show that the 3*s* limits of global inductance variations (with a Gaussian distribution) are generally about a quarter less than the worst-case values in Table 3.1. (The comparison is valid, as line width has no effect on the global variations.)

The inductance variations used in Monte Carlo analyses are shown in Table 2.2. The ±10 % global variation more than covers the layer thickness tolerance, whereas the ±15 % local variation allows for width tolerance as well as an extra ±7 % to ±10 % uncertainty in the nominal value of an etched inductor. These values are larger than the actual variations expected for the Hypres process, as was found in an analysis of global deviations [27] and will be demonstrated for both global and local deviations in section 3.9. This conservative approach (similar to that of [9], although their values are different) originates from earlier work [3] when inductance and its variations could not yet be established with any certainty, and was retained for consistency between yield results of new and established circuits.

Since Monte Carlo analyses show yield to be sensitive to inductance, reliable inductance calculation techniques are necessary.

3.3 INDUCTANCE CALCULATION TECHNIQUES

3.3.1 Mathematical and 2D approach

The easiest way to calculate the inductance of a two-dimensional superconducting microstrip line remains the analytical technique proposed by Chang [23]. However, as with other analytical techniques, this one is very limited.

If we stay with a 2D problem, but add a second conductor, the problem of determining inductance escalates beyond the abilities of analytical mathematics. A numerical calculation becomes necessary, and once again Chang proposes a technique [28].

The numerical technique provides a fast and easy way to calculate the mutual inductance between multiple superconducting lines, with or without a ground plane. Yet, like the analytical technique, it is unable to handle three-dimensional structures such as corners and vias.

3.3.2 Numerical calculations on 3D models

As mentioned before, the analytical and 2D techniques for inductance calculation are only accurate for very simple structures. Even with the addition of tables for corners and tees they can still, in general, not deliver good solutions. Furthermore they are virtually useless for any 3D structure of which the designer does not have prior knowledge of the inductance characteristics.

When the mutual inductance between lines that do not run parallel to each other for their full lengths (as is the case in any practical layout) needs to be computed, the analytical and 2D techniques fare even worse. Circuit yield is also sensitive to mutual inductance (3.4). In a simulation experiment on the HUFFLE, margin and yield analyses were combined to determine the margins of the mutual inductance coupling coefficient that would result in a noticeable yield reduction for the layout tolerance model. The margin was found to be somewhere between 20 % and 25 %.

It is evident that the structures in Figure 3.3 require 3D modelling of some sort. Several 3D numerical analysis programs are available for the calculation of the inductance of superconducting microstrip lines, but not all can handle the sort of complex structures shown in Figure 3.3. After evaluation of several available programs, it was decided to use *FastHenry* with superconductor support [29] for all inductance calculations.

The generation of segmented 3D structures for *FastHenry*, as well as the evaluation of calculation results are discussed in the rest of this chapter. Firstly, however, it is necessary to show that *FastHenry* will give results similar to those of the other 3D and 2D techniques when applied to a simple problem that all of them can handle.

As an example, consider the calculation of the inductance of the simplest of elements, a straight superconducting microstrip line above a superconducting ground plane. The parameters for the Hypres 3-micrometre Nb process [16] are used (1997 values, Rev. 017); with the conductor in layer M1, and the ground plane in layer M0. For these calculations, line thickness is 200 nm, dielectric thickness is 150 nm and the GP is 100 nm thick. The penetration depth for a niobium thin-film is 90 nm.

In the 2D programs (with the exception of *Induct*), the ground plane is infinite. In the 3D programs the ground plane is chosen to be around 20 *m*m wide. The conductor has a width of 5 *m*m. Since the 2D techniques ignore fringe effects at the ends of the line, the 3D structure is made long (100 *m*m) to limit the detrimental effects of fringing.

The calculations and characteristics of six programs are compared in Table 3.2.

- *Matlab* [30] was programmed with Chang's equation [23].
- *Sline* [31] also uses Chang's equation.
- *Induct* [32] uses Chang's 2D numerical formulation [28].
- *LL* [33] uses a 2D boundary element method to solve the per-length inductance of a superconducting structure.
- *FastHenry* [26] with superconductor support [29] uses a magnetoquasistatic formulation of Maxwell's equations, from which a mesh analysis is created and solved with a multipole-accelerated algorithm.
- *3D-MLSI* [34] utilizes a 3D finite element sheet current method, although 3D structures are limited in complexity.

Table 3.2: Inductance of superconducting microstrip calculated with different programs

Simulation Tool	Engine Type	Result [pH / *m*m]	Comments
Matlab	2D, Analytical	0.07853	Instantaneous
Sline	2D, Analytical	0.07867	Instantaneous
Induct	2D, Numerical	0.07585	Very Fast
LL	2D, Numerical	0.0742	Fast
FastHenry 3.0wr	3D, Numerical	0.07487*	Slow, memory hungry
		0.07583**	
		0.07478***	
3D-MLSI	3D, Numerical	0.08069	Very slow

* Single length-element, 175 filaments, small GP.

**GP segmented as 40×100, line as 7 segments in width, 4×4 filaments. Large GP overhang.

***Method of images. Line segmentation is 7×3×140 in width×height×length.

The analytical two-dimensional techniques agree very well with one another, allow very fast calculation of inductance, and can be implemented readily in CAD packages. During manual layout of superconducting VLSI circuits, these programs are indispensable to the engineer. The major drawback to the 2D techniques is that, although they account for fringe effects at the conductor boundaries, they cannot include fringe effects at the terminal edges. Fortunately, practical inductors in ICs are never open-ended, and always terminate in structures that limit fringing, such as Josephson junction cover pads.

The 3D techniques agree very well with the 2D methods (no solution in Table 3.2 is exactly correct), which gives us more confidence for their application to complex 3D structures. The three different techniques used with *FastHenry* also agree extremely well, with the images method coming to within 0.15 % of the solution obtained for a single line segment and ground plane. The primary reason for the *FastHenry* results being lower than the analytical ones is that fringing caused by the finite line length resulted in a lower calculated inductance. The various techniques used with *FastHenry* are discussed in the rest of this chapter.

As a final comment, the major drawback to any 3D technique is the time that it takes to produce an answer, and the computing resources it ties up while doing so. *FastHenry* gobbles up copious amounts of memory for even modest structures, especially when the discretization density is increased, whereas the limited two-conductor student version of *3D-MLSI* – although less demanding on memory – requires extremely long run-times even on powerful workstations.

3.4 SEGMENTED 3D MODELS

3.4.1 Segments and filaments

As discussed in section 3.3.2, *FastHenry* yields sufficiently accurate results for simple microstrip lines. The program is therefore used for all further inductance calculations in this dissertation. The rest of this chapter is dedicated to the modelling of 3D structures with the explicit goal of feeding the models into *FastHenry* and ensuring reliable calculation results.

FastHenry input files specify each conductor element as a collection of cylindrical segments with rectangular cross section, each of which carries a uniformly distributed current along its length [26] [35]. These segments can also be subdivided into parallel filaments, each with a uniform cross sectional current density (Figure 3.5). For superconducting elements, a penetration depth is specified. Normal conductors such as resistors are given a conductance value.

When complex structures require current flow in different directions within the same conductor, segments have to be created in each axial direction of interest.

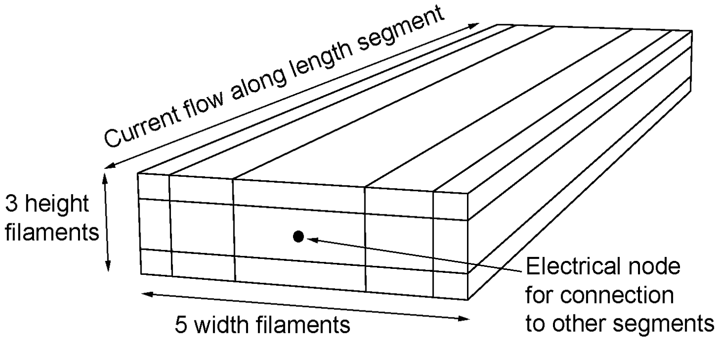
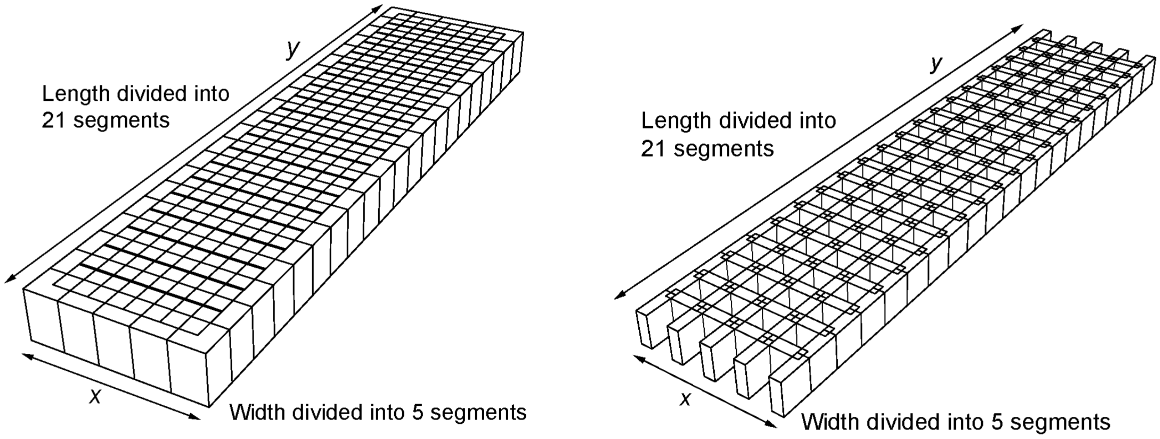

Current flow along length segment
3 height filaments
5 width filaments
Electrical node for connection to other segments

Figure 3.5: *FastHenry* line segment shown with filaments and connection node

Figure 3.6 (a) shows how a line is segmented in the *x* an *y* directions. Each segment can be divided further into filaments, as shown in Figure 3.5. The structure in Figure 3.6 (b) is a graphical representation of the same line with all segment widths reduced to one third of their actual values. This is merely a visual aid to help clarify the segment interconnection pattern.



(a) **(b)**

Figure 3.6: Segmented line (a) as it really looks and (b) graphical representation with segment widths shrunk to one third of their actual values

Finer segmentation and filamentation yields more accurate answers, and normally the calculated inductance values decrease as segmentation density increases. If discretization is sufficiently fine (Teh *et al.* suggest a mesh size equal to the penetration depth for a similar numerical method [36]), a further increase in the density does not lead to a more accurate answer. However, such segmentation strategies are extremely expensive on computer RAM and CPU time. In practice it is easier to calculate the inductance for two or three segmentation densities and then fit the answers to a curve extracted from that of a simple structure in order to estimate the answer for a very fine discretization [37].

Figure 3.7 shows the results of several *FastHenry* simulations on a segmented microstrip with various filamentation densities. The conductor is in layer M2, 5 *m*m wide (7 segments) and 10 *m*m long (14 segments). The ground plane has 40×50 segments in width and length with no filament subdivisions, and the overhang is 10 *m*m on all sides.

As can be seen from the figure, the calculated inductance is lower when more filaments are used, and approaches an asymptote as the number of filaments is increased. In this instance, 2×3 filamentation in width and height already yields an answer that is within 1 % of the asymptote.

Simulations with few line filaments, and more height filaments in the GP, have shown that the values in Figure 3.7 will not decrease by more than 0.35 % for high GP filamentation.

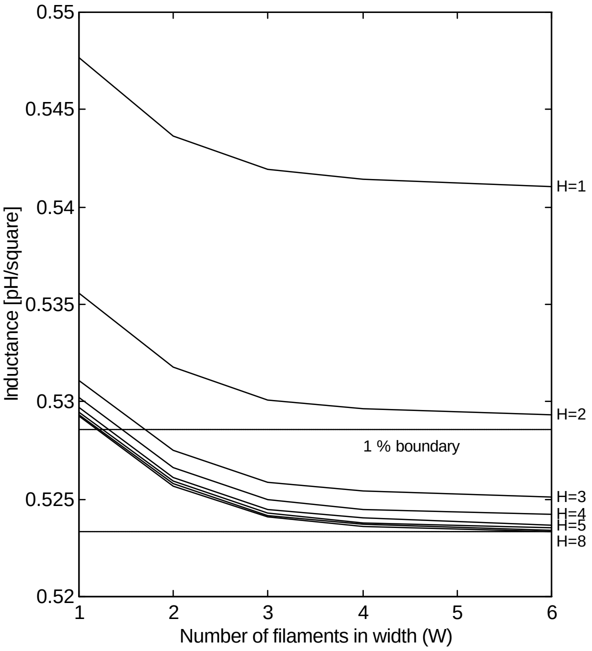


Figure 3.7: Per-square inductance of superconducting microstrip over GP for various filamentation densities

3.4.2 *Current flow considerations – injection and cornering*

When lines are segmented the current entry point cannot be chosen at one node along the edge only, as this forces current to concentrate there and gives a higher inductance value. In line structures, all the nodes along the input edge are set electrically equivalent (see Figure 3.8 [37]). At the furthest edge of any structure, where a connection is needed between a conductor and the current return path, every node along the edge is set electrically equivalent to the corresponding node on the image structure or ground plane; depending on which is used.

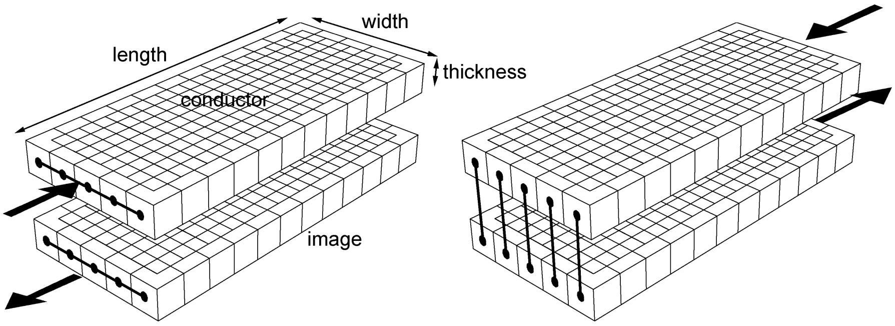


Figure 3.8: Equivalent nodes on segmented line edges (front and rear) for the method of images (thick lines show electrical equivalence, arrows show current injection/extraction)

In RSFQ circuits, inductance is most often calculated between connected Josephson junctions. Each junction is modelled as a via connecting a top and bottom pad (see Figure 3.16 for construction details). The current entry point is the bottom pad of one junction. All the bottom nodes of the via itself, where it ties into the bottom pad, are set electrically equivalent (the path labelled C in Figure 3.9 [24], or the lowest rectangle of nodes in Figure 3.16(a).)

A corresponding set of equivalent nodes on the ground plane or image junction, depending on the technique used, is used for current extraction.

When a Josephson junction forms the end of an inductance structure, the nodes along the path C are not set equivalent, but each connected to the corresponding node on the ground plane or the bottom pad of an image junction.

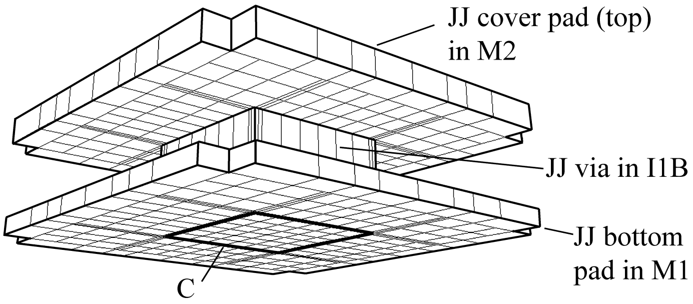


Figure 3.9: Equivalent nodes for current injection into Josephson junction

In structures containing discontinuities such as corners, current flow across the structure is not uniform. This can be visualized with EM simulation software. Current flow shows a peak at the inside of a corner and a null (for unchamfered bends) at the outer vertex [38]. It is therefore important to segmentize sufficiently to allow current to concentrate, for example, at the inner edges of sharp discontinuities.

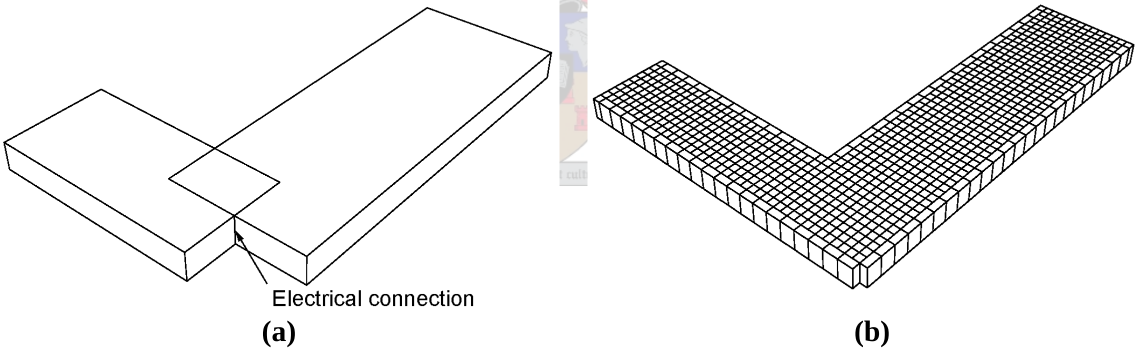


Figure 3.10: Corner structure formed with (a) single-segment lines connected together and (b) segmented lines sharing all corner segments

The corner structure in Figure 3.10 highlights the necessity for sufficient segmentation. When each arm is constructed with only one segment, they are electrically connected at the centre of their coincident edges – even if the segments themselves are subdivided into filaments. This forces current to concentrate at the electrical node in *FastHenry*, whereas in practice the current density is highest at the inside of the corner. The misrepresentation in *FastHenry* leads to a calculated inductance that is too high, especially for structures with short arms. The segmented corner structure in Figure 3.10 (b) supports a varying current density, and allows current to concentrate around the inside of the corner.

Incidentally, it was found through simulations on corner structures such as the one depicted in Figure 3.10 (b) that a better approximation for the effective length of a corner when the arms are long is about 0.5 times the width. This is much lower than the conventional assumption of 0.707 as shown in Figure 3.1, but agrees with the observation made by Teh *et al.* [36]. For very short arm lengths (down to a length-to-width ratio of 1), the calculated effective length of a corner increases slightly to around 0.535 times the width. Teh *et al.* [36] predict a larger increase, but they are not quite clear on the length-to-width ratio. Furthermore, at these very short lengths it becomes increasingly difficult to compensate for the effects of fringing when the inductance analyses are performed. As a design rule-of-thumb, a corner can therefore be approximated as half a square if the arms are longer than about one and a half squares. For shorter arm lengths, numerical calculations would be better.

Single-segment lines such as the one in Figure 3.10 (a) give a corner inductance of about 0.9 squares – most definitely too high.

3.4.3 Basic structures used to build 3D models

Routines were developed that compose segmented structures for common rectangular objects, such as lines, vias, uniform corner or tee-in pads, pads with nodes arranged for connection to vias, and connection strips.

The most commonly used structures are shown in Figure 3.11 [24]. In order to allow current flow in a three-dimensional model, each structure is constructed from many small segments. Horizontal structures such as lines and pads have close-fitting segments in the *x* and *y* directions, as these are the only axial directions in which current can flow. Vias have segments in all three Cartesian directions to allow current flow between layers. Connection strips are the only structures allowed to have elements in directions other than the Cartesian axes, since they can be used to connect conductors on different levels.

Vias are made hollow, with their vertical edge segments at least as thick as the London penetration depth [25]. Simulations on hollow conductors have shown that the loss in accuracy is negligible. The vertical edge segments of vias can therefore be given any width greater than the penetration depth, so that more evenly segmented structures can be constructed.

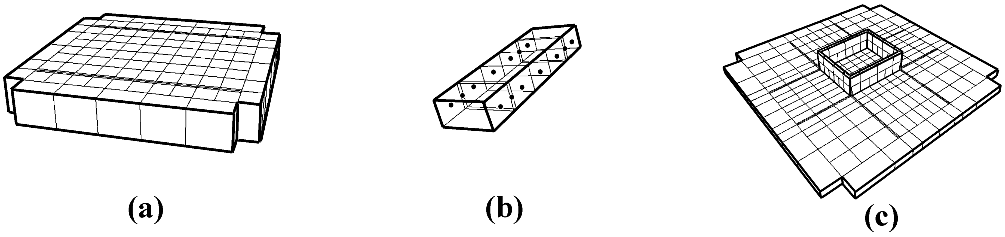

(a) (b) (c)

Figure 3.11: Basic structures used to construct complex 3D models are (a) a transmission line, (b) a connection strip (dots show the location of nodes) and (c) a via mounted on a pad

Segments can also be subdivided into filaments for greater accuracy.

All dimensional parameters, as well as segmentation and filamentation densities, are variable. Any combination of these basic structures, with their appropriate edge nodes connected through short strips, can now be used to create complex conductor shapes. A library of basic inter-junction configurations has been created in this way, and contains a Josephson transmission line (JTL), RSFQ pulse splitter, and several connected junctions with line crossings, layer changes, corners and tee-ins (see Appendix D). When new layouts are created, the appropriate form is selected, dimensional parameters supplied, and a three-dimensional model created automatically. Inductance is then extracted, and the layout dimensions corrected if necessary.

3.4.4 Hollow models for conductors

Hollow models use the fact that most current in superconducting structures flow within the London penetration depth if the thickness of the conductor far exceeds the penetration depth. Simulations of structures that were only built up to the penetration depth (and therefore hollow) showed little deviation from solid structures, thereby vindicating the use of hollow vias.

As an example, the inductance of a solid M3 line with a thickness of 600 nm and 7 height filaments was compared to that of a hollow M3 line. The penetration depth is 90 nm. Even when the hollow line had only 1 height filament for each segment, its inductance was only 0.5 % higher than that of the solid line. When both structures had 7 height filaments, the difference in inductance was a mere 0.28 %. For solid and hollow lines in M2, with a thickness of 300 nm, the difference was four times as large, or 1.1 % when both had equal numbers of filaments.

The conclusion from these results is that hollow models deliver results very close to those of solid models (especially for thicker lines). However, they are too expensive to segmentize properly, and were only used to evaluate the hollow via simplification.

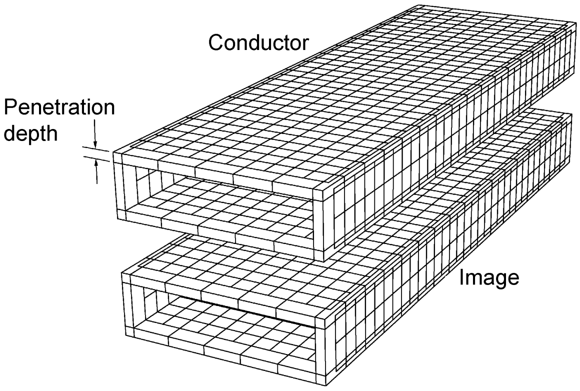


Figure 3.12: Hollow conductor and image

3.5 METHOD OF IMAGES

There is nothing more practical than a good theory.
Leonid Ilich Breznev

Superconducting circuits are normally etched above a large ground plane. The ground plane can be included in simulation models, but has to be finely segmented for accuracy. These segments should be similar in size to those on the most critical conductors. This unfortunately means that, unless very complex non-uniform segmentation algorithms are used, large parts of the ground plane far from the conducting structures waste segments and computing resources.

Fortunately the ground plane can be omitted when the method of images is used (exactly as for antenna radiation pattern calculations [39]). The only requirements are that the ground plane should be large enough (in relation to the inductor) to be considered effectively infinite, and that it should have a high enough conductivity to be considered a perfect conductor. The superconducting ground plane in the Hypres process satisfies both.

Implementation of the method of images is easy – all structures are mirrored to form images. The original structures and their images are then placed at equal distances on both sides of a reflection plane.

The equation for the self-inductance of a conducting loop with magnetic flux passing through it is given by

$$L = \frac{\int_S m_0 \mathbf{H} \cdot d\mathbf{a}}{i}, \tag{3.1}$$

where m_0 is the permeability of free space, **H** is the magnetic field intensity, and i is the current in the loop [39]. It follows from the surface integral in (3.1) that the inductance calculated by the method of images, in which the loop surface area is twice that of a conductor above a ground plane, has to be divided by 2 in order to find the real inductance.

3.5.1 Reflection plane

The placement of the reflection plane in the method of images has a significant effect on the calculated inductance, especially for layers close to the reflection plane. This makes mathematical sense, since these layers leave less area for a loop surface (see equation 3.1), and are therefore more susceptible to area changes brought on by moving the reflection plane.

One of the earliest publications on inductance calculations in superconducting circuits with the method of images used the upper boundary of the GP as the reflection plane [40]. However, this assumes that all the return current flows on the outside of the ground plane (that fills an infinite half plane), and can only be accurate for geometries that are much larger than the penetration depth of the superconducting ground plane.

In practical integrated superconducting circuits, the thin-film conductor, ground plane and dielectric thicknesses are all in the same order of magnitude as the penetration depth. The current that flows underneath the top of the ground plane can therefore not be ignored, as it invalidates the sheet current assumption that underlies the argument of placing the reflection plane at the top of the ground plane.

In a previous paper [25], the reflection plane was set by comparing the results of an image method with that of an accurate ground plane calculation. Unfortunately the positional value was not published. A more recent paper [36] proposes that the reflection plane should lie at the effective penetration depth (l_{eff}) of the superconducting ground plane. The equation for effective penetration depth is

$$l_{eff} = l \coth\left(\frac{d}{l}\right), \tag{3.2}$$

where l is the London penetration depth and d the thickness of the superconducting film.

A structure from [36] was selected, and their results compared to those of the segmented GP method in *FastHenry*. The results are shown in Table 3.3, and are also compared to the analytical solution from Chang [23]. The microstrip parameters are: length = 100 *m*m, width = 5 *m*m, line thickness = 220 nm, GP thickness = 300 nm, dielectric thickness = 177.5 nm, penetration depth of line = 137 nm, penetration depth of GP = 86 nm.

Table 3.3: Microstrip inductance calculated through various techniques

Method	L [pH/*m*m]	Change from Chang method [%]
Chang (*Sline*)	0.0903	-
Teh *et al.*	0.0882	-2.33
Segmented line with GP in *FastHenry*[*]	0.0885	-2.02

[*] Discretization divisions: length×width×height: 140×7×3

The results in Table 3.3 agree well with each other, with both numerical methods yielding answers slightly less than the analytical solution as a result of fringe effects at the line ends.

Figure 3.13 shows a 2D cross section of a superconducting conductor above a ground plane (d is the dielectric thickness), as well as the locations of the reflection plane and the image. The effective height of the conductor above the reflection plane is $h_{c\text{-}rp}$.

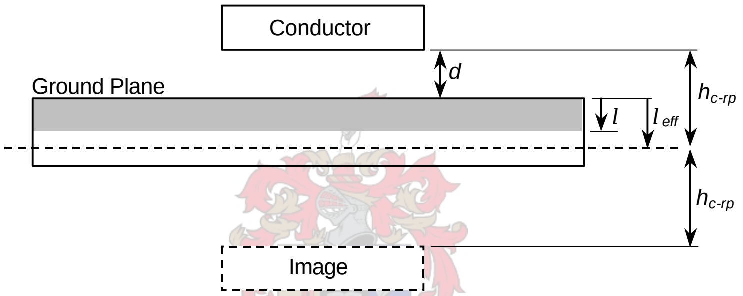


Figure 3.13: Position of conductor above and reflection plane below the ground plane

The next step was to implement the method of images in *FastHenry*, set the reflection plane at l_{eff}, and compare the results of a few line structures with different layer thicknesses to the analytical solutions with Chang's method. The results are shown in Table 3.4. For the microstrip line, width = 5 *m*m and line length = 50 *m*m. The *FastHenry* analyses used discretization divisions (length×width×height) = 70×7×3.

Table 3.4: Microstrip inductance calculated with *FastHenry*, using the method of images and reflection plane at l_{eff} , compared to Chang's analytical method

Structure	t_1 [nm]	t_2 [nm]	h [nm]	l_1 [nm]	l_2 [nm]	L [pH/µm]	Change on Chang [%]
Teh *et al.*	220	300	177.5	137	86	0.08804	-2.51
M1	200	100	150	90	90	0.07473	-5.00
M2	300	100	350	90	90	0.10495	-4.34
M3	600	100	850	90	90	0.16290	-4.01

In Table 3.4, t_1 is the conductor thickness, t_2 the GP thickness, h the dielectric thickness, and l_1 and l_2 the London penetration depths for the conductor and GP respectively.

As can be seen from Table 3.4, the result of a *FastHenry* analysis on the line structure from Teh *et al.* [36] agreed to within 0.2 % of their value.

However, it is also evident that the difference between the numerical and analytical solutions becomes more pronounced for layers that are closer to the GP. Other reflection plane positions were therefore evaluated, and the results are presented in Table 3.5.

Table 3.5: Percentage difference between inductance calculated with the method of images in *FastHenry* and Chang's analytical method for different positions of the reflection plane
Line width = 5 *m*m, line length = 50 *m*m, segmentation divisions (l×w×h) = 70×7×3; Chang is nominal

Reflection plane position	M1	M2	M3
0 (GP surface)	-31.3 %	-20.0 %	-11.0 %
l	-9.97 %	-7.31 %	-5.37 %
l_{eff}	-5.00 %	-4.34 %	-4.01 %
$1.1l_{\text{eff}}$	-2.52 %	-2.85 %	-3.34 %
$1.2l_{\text{eff}}$	-0.0586 %	-1.35 %	-2.67 %
$1.582l_{\text{eff}}$	+9.17 %	+4.26 %	-0.0847 %

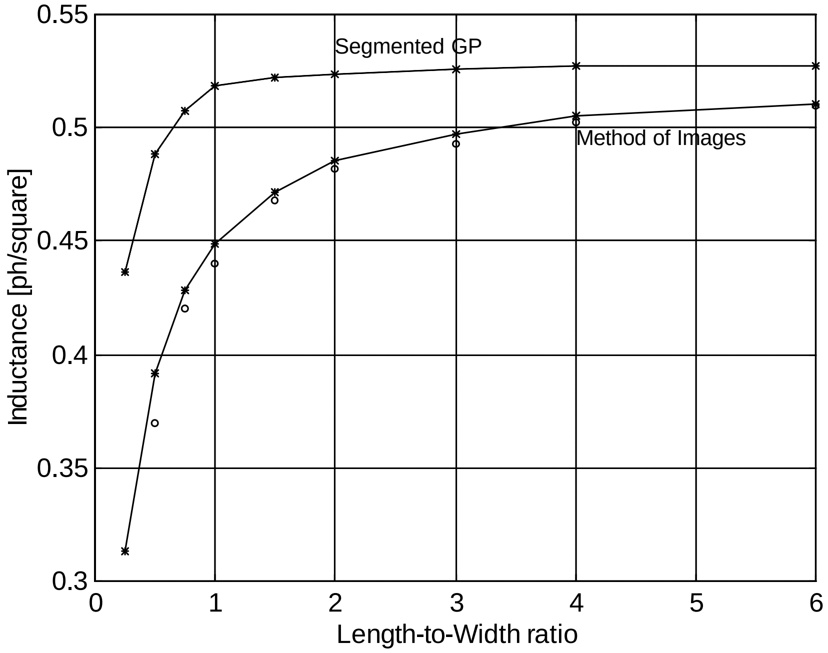
It is clear that the correct position for the reflection plane is either at l_{eff}, or slightly further down at $1.1l_{eff}$. At l_{eff} the difference between the numerical and analytical solutions becomes more pronounced as dielectric thickness decreases (see Table 2.2, p.7 for Hypres layer dimensions), whereas at $1.1l_{eff}$ it is the other way round.

Theory predicts that inductance is decreased by the fringe field factor [23], and that this fringe factor becomes larger when the dielectric thickness, or separation between conductor and GP, is decreased. It is thus expected that structures in M1 should experience a larger relative reduction in inductance due to fringing than similar structures in M2 or M3. Hence the conclusion that l_{eff} is indeed the best position for the reflection plane [37].

As a further evaluation of the performance of the method of images, the calculated per-square inductance of a microstrip was compared with published results for different length-to-width ratios.

Figure 3.14 shows the per-square inductance of an M2 microstrip as a function of the length-to-width ratio, calculated with the method of images and with the reflection plane at l_{eff}. The conductor is in layer M2, 5 *m*m wide, with 7 lateral segments and 6×8 filaments in width and height. Results for a *FastHenry* analysis with a segmented ground plane are included, and the GP overhangs the microstrip by 10 *m*m on all sides. Published results (interpolated) [25] are included as circles.

The method of images results agree very well with the published results of [25], except for large discrepancies below a length-to-width ratio of 1, where fringing effects seriously influence the inductance. The disparity may be caused by a difference in the way that line ends are connected (see Figure 3.8), as it is not known which technique they employed. The good agreement between results for a length-to-width ratio larger than 1 shows that the postulated reflection plane position at l_{eff} yields good results.



KEY: Method of Images [——], Segmented GP [– – –], Guan *et al.* [o]

Figure 3.14: Microstrip inductance per square as a function of length-to-width ratio

Figure 3.14 also shows that the segmented GP does not perform as well as the image method for small structures; most likely because the GP is too small. Better solutions may be found with a non-uniformly discretized GP [35], but since the images method is faster and already accurate enough, the former technique was not investigated.

In RSFQ circuits, conductors are never open-ended as in the models used to generate the results in Figure 3.14, but always terminate in other conductors or Josephson junction pads (see Figure 3.19) Consequently the inaccuracies introduced by the fringe effects of very short inductors will not influence practical inductance calculations.

From the preceding discussions and results, it is evident that the placement of the reflection plane at l_{eff} delivers results that are accurate and reproducible. Henceforth all inductance calculations with the method of images in this dissertation employ this reflection plane model.

3.5.2 Scaling factors for finer discretization

It is often impractical or even impossible to discretize large structures finely enough to yield results that fall within 1 % of the asymptote for sub-penetration depth discretization. Once again a thorough analysis was performed on the same structure used for the calculation of the per-square inductance graph in Figure 3.7, but this time using the method of images. The results for different filamentation densities are shown in Figure 3.15.

The microstrip parameters are, once again: width = 5 *m*m (7 segments), length = 10 *m*m (14 segments). The microstrip is in layer M2.

The inductance values for very high filament counts were extrapolated, because the large number of effective segments were impossible to handle with available computers.

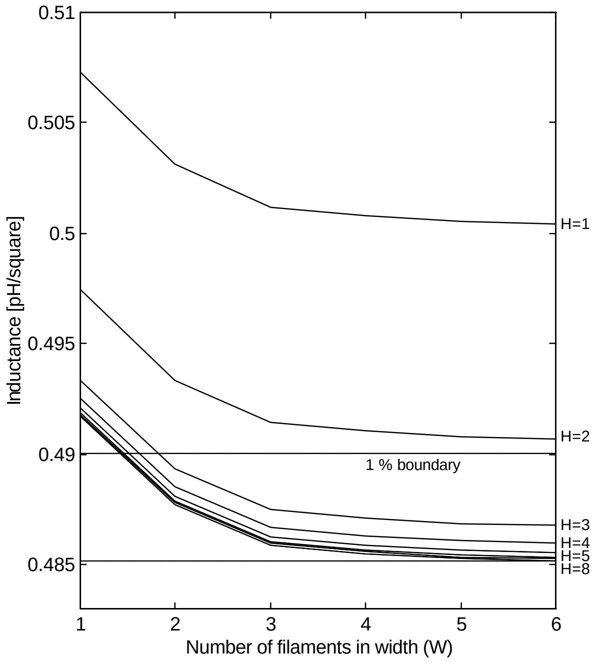


Figure 3.15: Microstrip inductance per square for different numbers of filaments, calculated with *FastHenry* and the method of images

A comparison of the results in Figure 3.7 and Figure 3.15 shows that they agree very well, with the inductance calculated for a filamentation of 1×1 about 4.6 % higher than the asymptote in both cases. The only difference between the two is that they were calculated for lines with different length-to-width ratios, so that the method of images yields an inductance that is consistently about 8 % lower than that calculated with the segmented ground plane. The shorter length-to-width ratio of 2 for the method of images calculation was required to limit the number of filaments necessary for calculating inductance near the asymptote value. The gap between the values calculated with the segmented ground plane and the method of images can be seen clearly in Figure 3.14.

From Figure 3.7 and Figure 3.15 it can be seen that an increase in filaments for the height of the conductor allows a solution to converge to a good value faster than would a similar increase in width filaments. For large structures, a filamentation of 6×8 for width×height is impractical, as this often requires many gigabytes of RAM and very long solution times. In practice the limit that a *FastHenry* simulation can accommodate is about 15 000 elements (segments and filaments). We used filamentation of 1×3 (width×height) for large structures, and 2×3 whenever possible. For 2×3 filamentation, the results are already within 1 % of those for 6×8, and thus good enough for all practical purposes.

The above results were also compared to those for 3D models of complex structures such as the Josephson transmission line (see section 3.6.1), and no discernible differences were detected.

For all further inductance calculations, *FastHenry* results were adjusted by multiplication factors calculated from the data in Figure 3.15 to the expected values for 6×8 filamentation. For example, the inductance of a superconductor microstrip is calculated with the method of images, and the inductance for 6×8 filamentation is then determined by extrapolation (as shown in (B.2) in section B.2.2).

3.6 COMPLEX STRUCTURES

With the basic building blocks defined, and the position of the reflection plane determined, any three-dimensional circuit object can be modelled for inductance extraction.

It is important to note that the calculated inductance through Josephson junctions only accounts for the inductance of the superconducting structures, and does not include the time-dependent junction inductance L_J [41, p. 470] [42, p. 210], defined as

$$L_J = \frac{\Phi_O}{2pI_C \cos j\,(t)}\ . \tag{3.3}$$

However, during circuit simulations L_J is accounted for by the junction model, and we are only interested in the inductance caused by the superconducting structures, as modelled by the 3D structures in this chapter.

Once again, all the layout examples and layer definitions discussed here are for the Hypres 3-micrometre process [16]. However, the techniques are universal and readily extend to other fabrication processes.

The damping resistor covers in layer M2 in are included in all models (see Figure 3.19 (b) for an example), since simulations have shown that these structures can reduce the inductance of a junction pad significantly. A reduction of as much as 8 % has been observed. Omitting these covers can lead to an overestimation of the actual inductance in short intra-gate lines.

Inductance extraction simulations for full three-dimensional models have also revealed that pad inductance is actually higher than what we have always estimated from analytical methods. The inductance of each junction pad in Figure 3.19, for example, is 0.42 pH when calculated from the bottom of the junction via to the start of the adjacent microstrip line (L_1 or L_2). An earlier analytical estimate placed the inductance at approximately 0.2 pH [3].

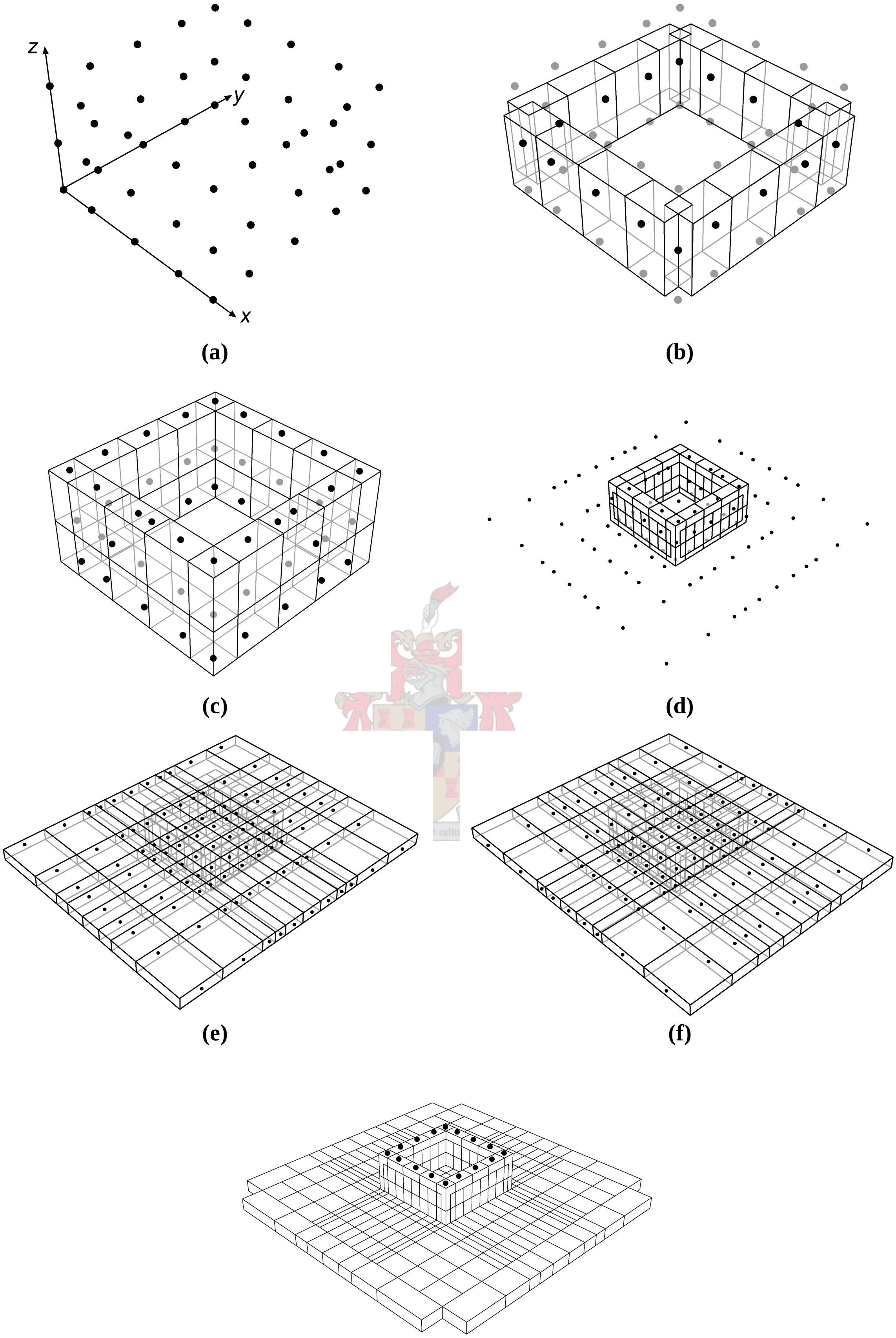
When only part of the inductance of a complex structure needs to be calculated, such as the inductance between J_1 and J_2 in Figure 3.23 (p.45), the current flow path is limited by shorting nodes from the 3D structure to their image counterparts so as to include only the structure of interest.

Complex 3D structures were created for every inductance of interest in the layouts created for this project. Examples of most of them are shown in Figure D.1, and the rest are discussed in the remainder of this section.

An example of how complex structures are formed is represented here as a series of figures. The model represented here contains a Josephson junction, and therefore starts with the definition of a via. Josephson junctions are always defined with the via first to make the assignment of node numbers easier. The vertical dimensions and penetration depth (and therefore the side wall thickness of vias) are stretched for clarity.

Figure 3.16(a) shows the three rectangularly ordered sets of nodes created for the definition of a via. In this case the via forms the I1B connection of a Josephson junction in the Hypres process [16]. Axes are shown to clarify directions. The nodes in the lowest ring will all eventually be set electrically equivalent, and used as the current entry point by *FastHenry*.

After the nodes are declared, the *x*- and *y*-directed segments are added, as shown in Figure 3.16(b). Segments are defined between nodes as shown in Figure 3.5.



(a)

(b)

(c)

(d)

(e)

(f)

(g)

Figure 3.16: Detailed construction of via and bottom pad for a Josephson junction inductance model

The height of the *x*- and *y*-directed segments is calculated so that their upper edges just touch the lower edges of segments in the top cover pad (to be added later), and their lower edges just touch the upper edges of the bottom cover pad segments.

Figure 3.16(c) shows the *z*-directed segments of the via. The middle set of nodes and hidden lines are shown in grey. In Figure 3.16(d), the entire via is shown with solid segments. The next step in the construction of the inductance model is to create the nodes for the bottom pad. These nodes are also shown in Figure 3.16(d). The bottom nodes of the via form part of the set of nodes for the pad, and are shown in grey.

In Figure 3.16(e) and Figure 3.16(f), the bottom pad segments in the *x* and *y* directions are shown. Hidden lines and the via segments are shown in grey. In Figure 3.16(g) the via and bottom pad are shown as they would appear in 3D diagrams in the rest of this dissertation, with hidden lines removed and segments intertwined. The upper nodes through which the via will be connected to a top pad are also shown.

Next, extra nodes are defined to allow the implementation of a top cover pad. The new nodes are shown in black in Figure 3.17(a), whereas the existing upper nodes of the via are shown in grey. Figure 3.17(b) shows the addition of *x*-directed segments (with hidden lines and other segments in light grey). The full Josephson junction model, consisting of a via and two cover pads, is shown in Figure 3.17(c). The nodes between which the top pad is built are shown in grey to indicate their relation to the *x* and *y* segments. Seven nodes are shown in black, as these constitute the coupling position to a transmission line that will be added to the inductance model next. Such a transmission line can connect to the top or bottom pad from any direction.

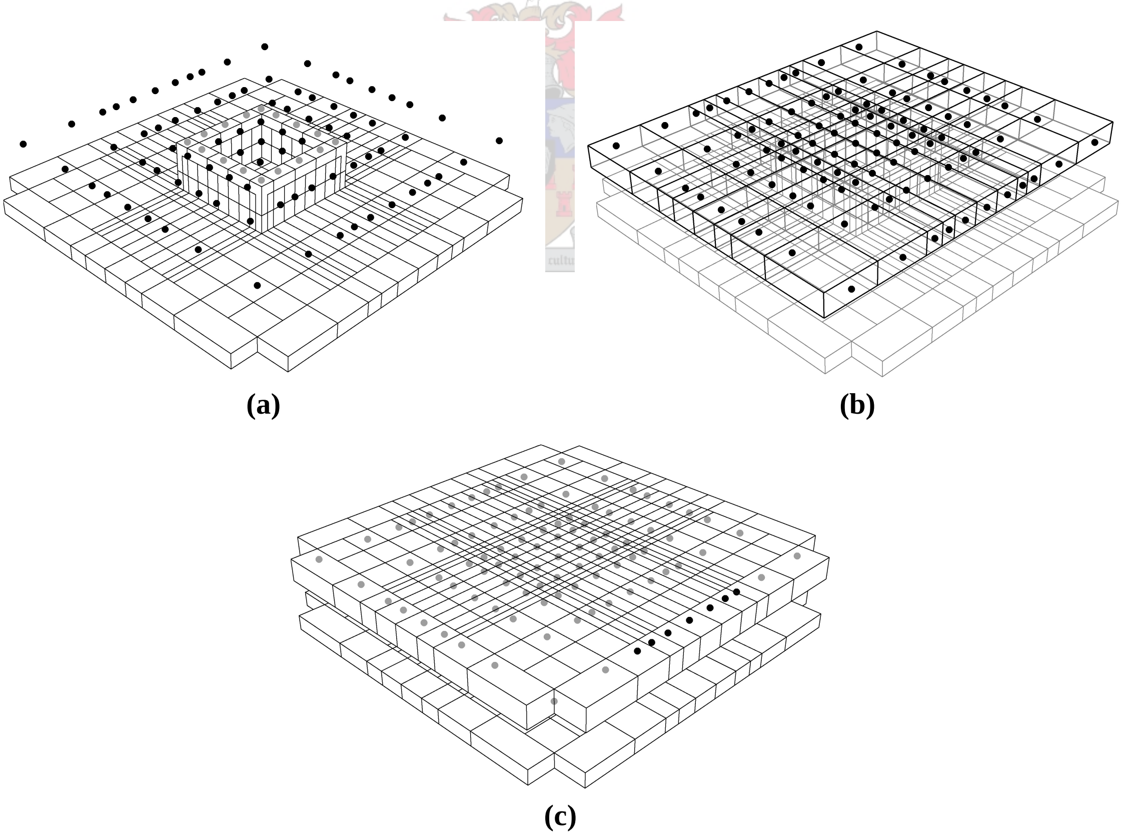


Figure 3.17: Detail on the addition of a top cover pad to complete the Josephson junction inductance model

When a transmission line is connected to the Josephson junction constructed in Figure 3.16 and Figure 3.17, it has to have lateral node spacings equal to that of the Josephson junction top pad (a restriction arising from the rectangular shape of *FastHenry* segments). It is for the purpose of adding a transmission line that the node spacing of the cover pads differ in the *x* and *y* directions.

In Figure 3.18(a), the nodes of a section of transmission line are shown along with the Josephson junction and the 7 nodes through which the line will connect to the top cover pad.

Figure 3.18(b) shows the *x*-directed segments of the transmission line, with hidden lines and the Josephson junction drawn in grey.

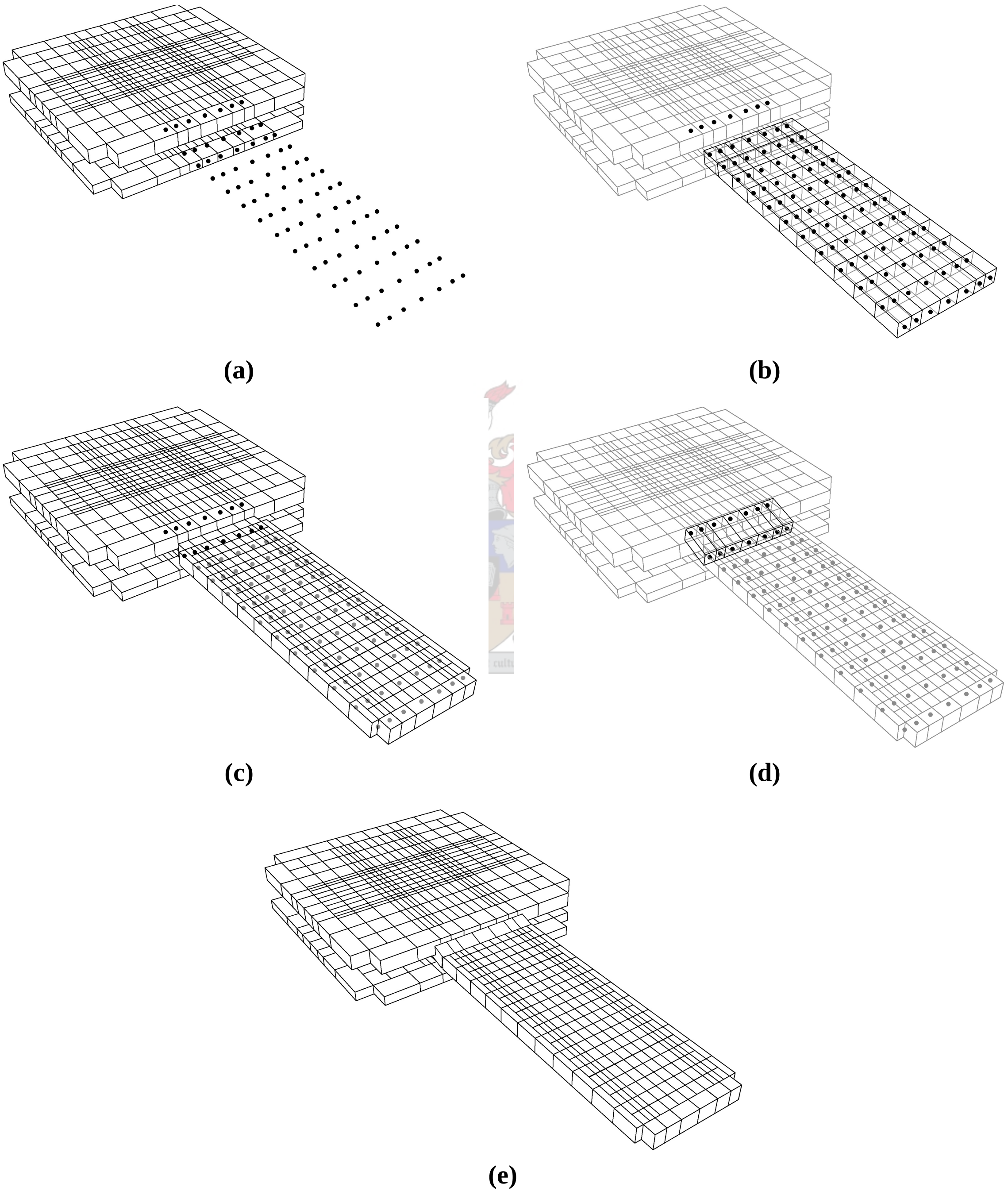


Figure 3.18: Detailed construction of a segmented transmission line and the connection thereof to a Josephson junction for a 3D inductance calculation model

In Figure 3.18(c) the transmission line is shown with both *x* and *y* segments, and hidden lines removed. The black nodes on the top cover pad and transmission line still have to be connected. This is done with a strip of segments as illustrated in Figure 3.18(d).

The completed structure appears in Figure 3.18(e). More transmission line sections can now be added, and each section connected to another by connecting their nearest nodes with segments like those highlighted in Figure 3.18(d). All the models discussed in the rest of this chapter, as well as those in Appendix D, were created like this.

A computer program was developed to handle node and segment placement for the generation of structures as detailed here. Scaling, direction and element numbering are all handled through parameter passing to the subroutines that generate the basic structures (see Figure 3.11), although the parameters are still calculated and controlled by the user who constructs the models.

3.6.1 Josephson transmission line

The first full 3D structure investigated was the standard Josephson transmission line (JTL) shown in Figure 3.19.

The layout structure was designed (through analytical estimations) to have an inter-junction inductance of 3.96 pH. With the full three-dimensional model, inductance was calculated as 4.30 pH, or 8.6 % higher than the design value.

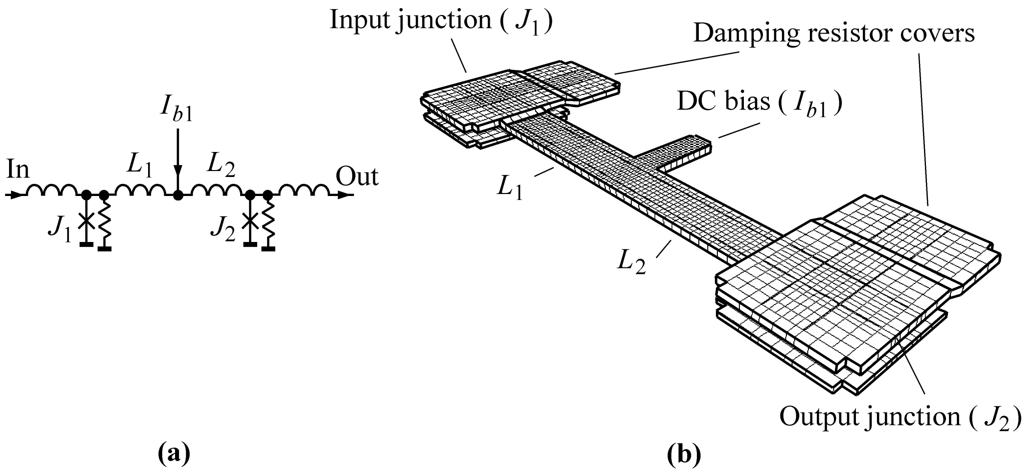


Figure 3.19: (a) Circuit diagram of a JTL and (b) segmented 3D model (with vertical dimensions doubled and image omitted for clarity)

The full three-dimensional model of a JTL without the dc bias tee-in was also used to establish the effect of a small length-to-width ratio on the per square inductance of a microstrip line. It is shown in [25] that per square inductance decreases sharply as a result of fringing when the length-to-width ratio is reduced to less than 2 (as shown in Figure 3.14 for a 5 *m*m wide line in M2). Similar results were obtained for a 4 *m*m wide line in M2 (the standard width for short JTLs in this dissertation), as shown in Figure 3.20. However, Figure 3.20 also shows the per square inductance of the same line when its endpoints are connected to junction pads (as virtually all inductors in RSFQ circuits are). The pads prevent fringing, so that a much flatter curve is obtained.

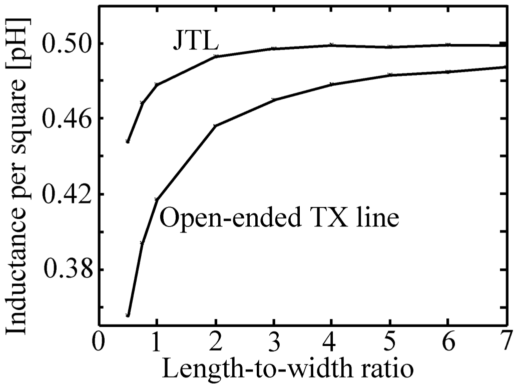


Figure 3.20: A comparison of the effect of small length-to-width ratios on the inductance of both an open-ended and a junction pad-ended microstrip line

The JTL was also used to gauge the effect of the tee and resistor cover pads on inductance. The inductance of each junction pad, from ground right up to where the microstrip lines start was calculated as 0.42 pH. This shows that the pads account for just over 20 % of the total inductance between J_1 and J_2.

When the tee structure is omitted, the total inductance between J_1 and J_2 rises by 0.047 pH, or about 1.2 %. Inductance rises another 0.5 % when the damping resistor covers are removed. These figures may seem small, even though the percentage change increases as inter-junction inductance is decreased. Still, omitting the tee and resistor cover pads from the 3D model introduces an error that is comparable in size to the standard deviation of inductance when all manufacturing tolerances are taken into account (see section 3.9, p.51).

For consistency and increased accuracy, dc tee-ins and resistor cover pads were included in every inductance structure analysed for this project.

A discussion on an inductance spread calculation for the JTL, as well as results thereof, are presented in section 3.9.

3.6.2 RSFQ pulse splitter

Another problematic structure is the RSFQ pulse splitter, as shown in Figure 3.21. The design values for inductance are: L_1 = 1.16 pH, and L_2 = L_3 = 1.64 pH.

A first layout was attempted with inductance values approximated from the Chang analytical model. A full three-dimensional model was then created. With the 3D model, L_1 was calculated as 1.275 pH, L_2 as 1.667 pH and L_3 as 1.666 pH. The simulations showed that L_1 was actually 10 % over the design value, and that the inductor had to be shortened.

The power of the full three-dimensional model is even more evident when the inductance of L_1 is calculated for a three-dimensional structure that omits the dc input and inductors L_2 and L_3. Without these nearby structures, L_1 is calculated as 1.361 pH, or 6.8 % higher than with the full model.

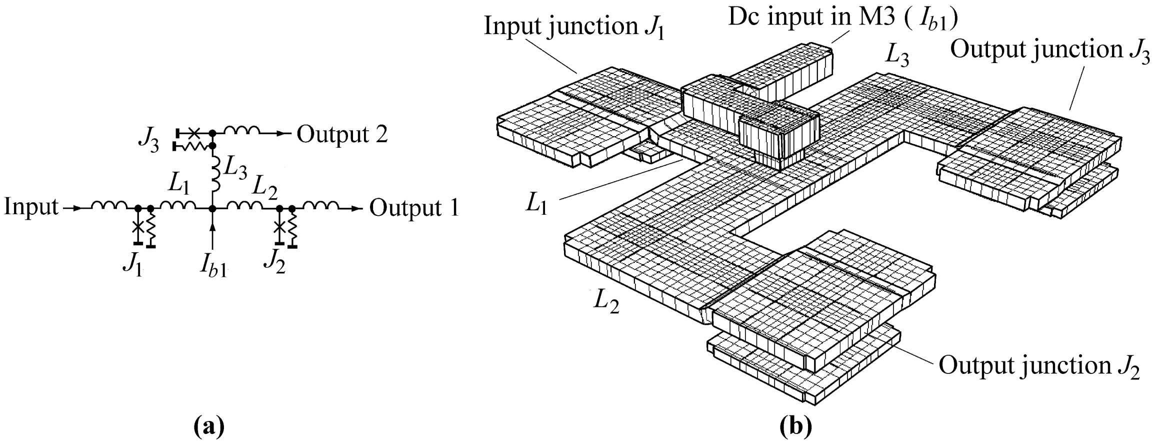


Figure 3.21: (a) Circuit diagram of an RSFQ pulse splitter and (b) segmented 3D model (with vertical dimensions doubled and image omitted for clarity)

3.6.3 Series junctions

Series junctions in RSFQ circuits are always placed between two grounded junctions. In circuit diagrams, the series junction is drawn in series with an inductor.

During layout, the series junction is placed as close as possible to one of the grounded junctions. It is then tempting to use only the remaining line structure between the series junction and the other grounded junction as the inductor, but this neglects the structural inductance of the series junction (caused by the via and pads) and the short connection to the closest grounded junction.

With series junctions, the inductance is calculated through the series junction right up to the junction that goes to ground. In RSFQ circuits, only the series inductance is important, and not how much is located at either end of a series junction. Consequently, parasitic inductance in such a loop can be added to the main inductance to have only one element in a simulation model.

In Figure 3.22 (b), the inductance L_1 has to be calculated from junction J_1, through the M2 connection to J_2, down through the J_2 via, and along the M1 line right up to the via where I_{b1} tees in. (The line in M1 labelled R_2 provides – in the physical circuit – the return path for current flowing from J_2 through the damping resistor R_2. It was included because it could lower inductance L_1.)

The 3D model for the calculation of the inductance of L_1 is shown in Figure 3.23. The current entry and exit points are located on the bottom pads of J_1 and its image respectively, and the bottom nodes of the I_{b1} tee-in via are shorted to their image nodes.

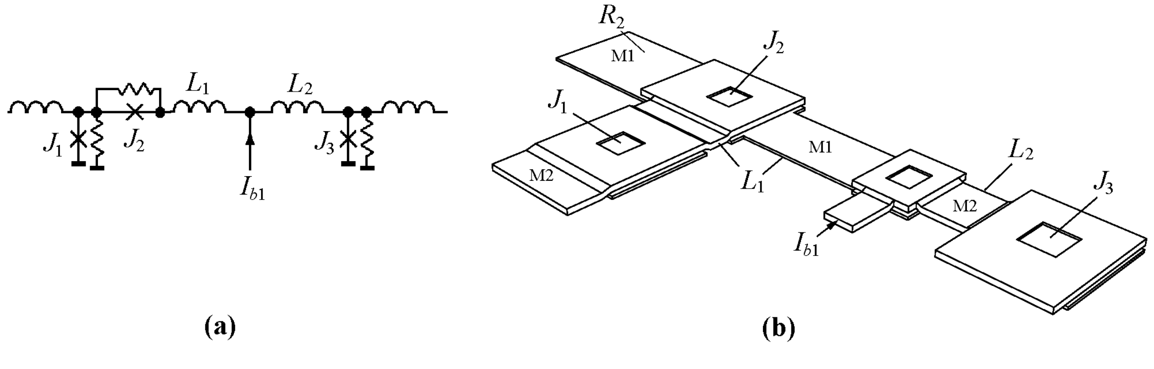


Figure 3.22: (a) Schematic excerpt from HUFFLE circuit showing a series junction (J_2) and (b) the 3D layout view

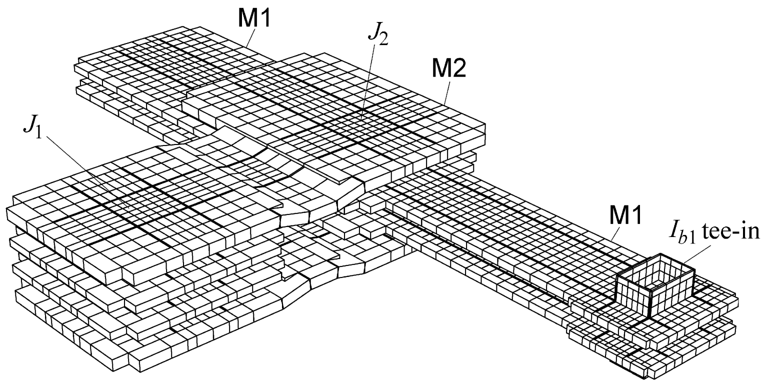


Figure 3.23: 3D inductance model of the series junction in Figure 3.22 with image included (vertical dimensions stretched for clarity)

In the HUFFLE, the series junction J_2 used for this example is placed in series with a 2 pH inductance (excluding the time-variant junction inductance). With the 3D model, the inductance between J_1 and J_2 is calculated as 0.76 pH, which accounts for 38 % of the required series inductance. Therefore, if the inductance between J_1 and J_2 is neglected during layout, and the series inductance taken as the structure between J_2 and the I_{b2} tee-in, the resulting total inductance of 2.76 pH would also be 38 % high.

Naturally, this effect is worse for cases where the series inductance should be even lower.

3.6.4 Damping resistors

As far as can be established from publications by RSFQ research groups, the finite inductance of a damping resistor is largely ignored during simulations. However, layouts are consciously adapted to attempt to keep the inductance as low as possible [2] [3].

As an example of both the effect of careful layout and the usefulness of the 3D inductance calculation strategy, the inductance of the damping resistor of a standard 250 *m*A Josephson junction was calculated.

The damping resistance is 1.21 Ω. Two distinctly different layout possibilities exist – one keeping absolutely within the strict design rules, and one using a tried and tested technique of etching an M2-to-M1 short circuit right over the edge of the resistor [43].

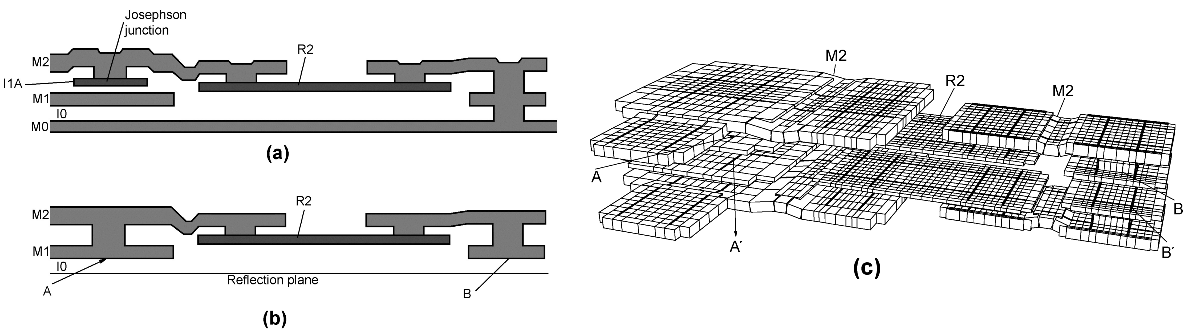


Figure 3.24: (a) 2D cross section of physical layout for a damping resistor created by keeping to conservative design rules, (b) 2D cross section of the inductance calculation model and (c) 3D view of the inductance calculation model (vertical dimensions stretched for clarity)

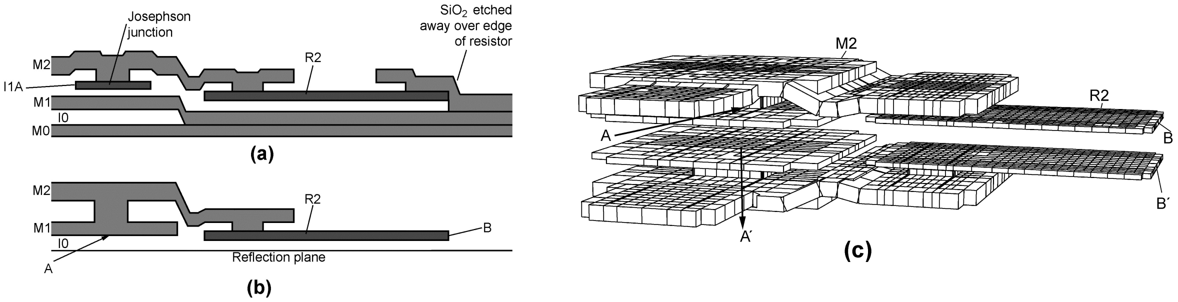


Figure 3.25: (a) 2D cross section of minimum inductance layout for a damping resistor, (b) 2D cross section of the inductance calculation model and (c) 3D view of the inductance calculation model (vertical dimensions stretched for clarity)

Figure 3.24 details the layout of a damping resistor according to the standard design rules. In (b) the simplifications for the inductance calculation model are shown, as well as the current entry point A and the point B where the inductance loop is shorted to the image. The image itself is omitted in (b) to reduce clutter, but can be seen in (c). The 3D view in (c) also shows the current exit point A′ and the loop short point B′ on the image.

Figure 3.25 shows the layout of a damping resistor for minimum inductance. In (b) the simplifications for the inductance calculation model are shown together with the current entry point A and the point B where the inductance loop is shorted to the image. Once again the image is omitted in (b). The 3D view in (c) shows the image, as well as the current exit point A′ and the loop short point B′ on the image.

A few concepts should become clear when Figure 3.25 is studied. For the low-inductance damping resistor layout, the SiO$_2$ layer I0 is etched away beneath the damping resistor, and a layer of metal (M1) is deposited in this hole [3]. This increases the ground plane thickness underneath the resistor to 235 nm, drops the resistor closer to the ground plane, and allows a direct short to ground at the edge of the resistor – all of which reduce inductance.

The vertical location of the resistor structure is modelled as being 100 nm (the thickness of the first SiO$_2$ layer on top of M1) above the ground plane. The thicker ground plane underneath the resistor is incorporated in the position of the reflection plane, since l_{eff} is reduced to 90.98 nm for all structures above the I0 hole (compared to a l_{eff} of 111.9 nm for the M0 ground plane). Since the reflection plane is flat, the change in l_{eff} is modelled by shifting all the structures above the I0 hole closer to the reflection plane. The dimensional stretch is absorbed by the M2 connection between the pads that cover the junction and resistor vias.

The conductivity of molybdenium (the metal used as to implement resistors in Hypres circuits) is 18.7 S.mm^{-1}. The thin-film resistors actually have a conductivity of about one half the value for bulk molybdenium (10 S.mm^{-1} for the 1 Ω per square resistance specified by Hypres [16]), but since the effect on inductance is negligible the bulk value was used for all calculations.

The minimum inductance layout yields an inductance of 0.895 pH, whereas the damping resistor in the strict layout has an inductance of 1.88 pH; or 110 % higher than the minimum inductance layout.

The damping resistor model was also used to calculate the impedance of the large damping resistors in the HUFFLE, all of which were laid out for minimum inductance. The 10 Ω resistor R_3 was calculated to have an inductance of 2.85 pH, and the 5 Ω damping resistor R_9 to have an inductance of 1.66 pH, as opposed to 5.44 pH and 3.45 pH respectively when standard resistor layouts are used.

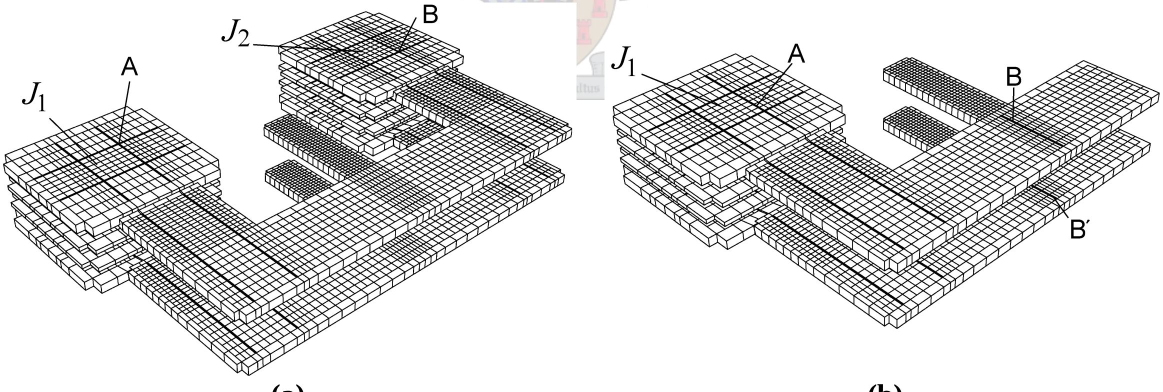
With normal damping resistors, the increased inductance of the conservative layout does not have any noticeable effect on circuit yield. However, the minimum inductance layout is more space effective when resistance values are small.

For resistor values a little larger than standard damping resistors, as in the case for the HUFFLE discussed above, the addition of a finite inductance in series with the resistor to the simulation model has been observed to decrease yield in generic MC models, as will be mentioned in section 4.5. However, for optimized circuits with voltage trimming, no discernible change in circuit yield could be found between circuits utilizing minimum inductance or conservative layouts for damping resistors. This suggests that good RSFQ circuits can tolerate the inductance of such resistors even up to values equal to the interconnection inductances.

Since the resistive elements do not have the penetration depth of superconductors, little change is observed for more filaments. The inductance solutions for long resistors (where the overwhelming contribution to inductance is by the resistor itself, and not by the junction structure and superconducting connections) are not scaled down to the asymptote value discussed in section 3.5.2.

3.6.5 *Simplifying models*

When a model is constructed for, say, a JTL with a U-shaped inductor, the inductance calculated between one junction and a line of nodes in the centre of the U-shape will be exactly half that of the inductance between the two junctions. Complex structures can therefore be broken into smaller sections, as long as a reasonable amount of segments are added beyond the loop termination point in order to avoid fringing.



(a) **(b)**

Figure 3.26: (a) 3D inductance model for U-bended JTL with both junctions and (b) smaller model for partial inductance calculation

The U-bended JTL depicted in Figure 3.26 illustrates this principle. The calculated inductance between junctions J_1 and J_2 in (a) is within a fraction of a percentage of twice the inductance between J_1 and the string of nodes lying on the line labelled B in (b). The microstrip in (b) is continued beyond line B to limit fringing.

3.7 MUTUAL INDUCTANCE

The design methodology for mutual inductance in superconducting circuits is not very well documented.

The standard technique of representing the coupling between inductors in superconducting circuits is to specify the coupling coefficient, *k*, along with the inductance values of each inductor. This simple representation is easy to read and simulate.

In practical layouts, however, inductors seldom share their entire lengths. This means that coupling is only achieved for a part of each inductor, resulting in a lowered effective coupling coefficient. Furthermore, inductance extraction is mostly performed on complete conductors – irrespective of what percentages of their geometries are magnetically coupled.

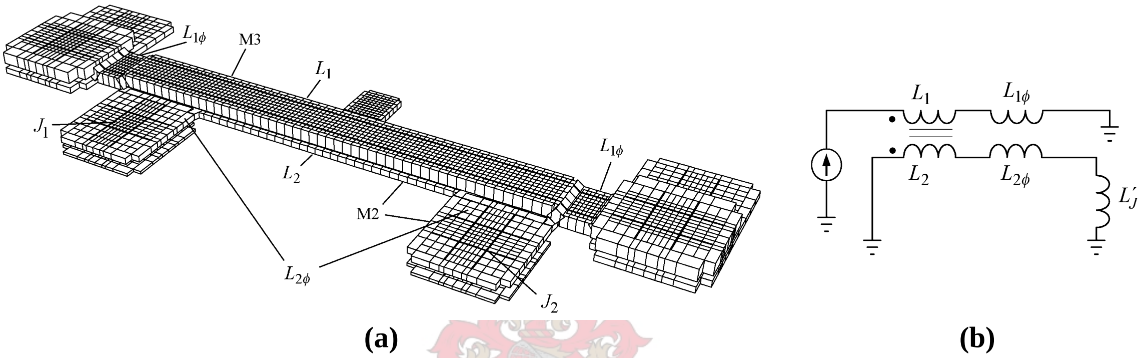


Figure 3.27: (a) 3D model of SQUID loop and control line (vertical dimensions stretched and image omitted for clarity) and (b) lumped element model

Thus, when layout extracted values are compared to those specified in circuit schematics, it is not the effective coupling coefficients that are important, but the ratio of the induced SQUID loop current to the control current.

Figure 3.27(a) shows the 3D model of a control line coupled to a dc SQUID loop. This specific configuration is discussed because it is the most common use for mutually coupled inductors in superconducting logic circuits.

In superconducting electronic circuits, all the materials on a chip are non-magnetic and have permeabilities very close to that of free space. As a result, coupled inductors form linear transformers. Hence the mutual inductance between two conductors (or the coefficient of induced voltage in one loop by a time-varying current in the other) is independent of direction, so that [44]

$$M_{21} = M_{12} = M = k\sqrt{L_1 L_2} \ , \tag{3.4}$$

where L_1 and L_2 are the coupled sections of the control line and SQUID loop respectively, and *k* is the coupling coefficient.

Figure 3.27(b) shows a symbolic representation of the inductive structures in (a). We shall continue to denote the coupled sections of the control line and SQUID loop as L_1 and L_2.

The 3D model (and real layout structure) may also contain (as is the case here) inductive sections that are not coupled. They are designated L_{1f} and L_{2f} here.

Finally, each current path may contain inductive components not reflected in the *FastHenry* output, and which do not affect the calculated mutual inductance value. Such hidden inductances in the control line have no effect on the induced current in the SQUID loop, and can therefore be ignored. However, the Josephson junction inductances in the SQUID loop itself decreases the induced current and must be included in inductance equations. For the general case, and from (3.3), the sum of the time-varying inductances of junctions J_1 and J_2 can be written as

$$L'_J = \frac{\Phi_O}{2pI_{C1}} + \frac{\Phi_O}{2pI_{C2}} \ , \tag{3.5}$$

where the phase difference j is approximated as small enough to make $\cos j$ near unity [7].

From the inductance matrix calculated with *FastHenry* we find

$$L'_1 = L_1 + L_{1f} \ , \tag{3.6}$$

$$L'_2 = L_2 + L_{2f} \ , \tag{3.7}$$

and

$$M' = k'\sqrt{L'_1 L'_2} \tag{3.8}$$

From Figure 3.27(b), and the relationship of voltage in a loop to the time-varying currents in both loops [44], we can write:

$$v_2 = -\left(L'_2 + L'_J\right)\frac{di_2}{dt} + M'\frac{di_1}{dt} \tag{3.9}$$

In a superconductor, the loop in which i_2 circulates is a perfect short circuit, so that $v_2 = 0$. If we substitute this value into (3.9), the result is

$$\left(L'_2 + L'_J\right)\frac{di_2}{dt} = M'\frac{di_1}{dt} \tag{3.10}$$

which can also be written as

$$di_2 = \frac{M'}{L'_2 + L'_J} di_1 \ . \tag{3.11}$$

Integration of (3.11) yields

$$i_2(t) = \frac{M'}{(L'_2 + L'_J)} i_1(t) + I_0 \ . \tag{3.12}$$

The gradient of (3.12),

$$\frac{M'}{L_2'+L_J'}=\frac{k'\sqrt{L_1'L_2'}}{L_2'+L_J'} \ , \tag{3.13}$$

describes the ratio of induced current to control current, and must approach the design value of a circuit. It is also evident from (3.13) that, since the inductances L_1' and L_2' can be larger than L_1 and L_2, the desired induced current ratio can still be obtained – even though the calculated coupling factor k' can be significantly lower than the design value k.

If the ratio calculated with (3.13) is too small, it can be improved by reducing the L_{2f} component of L_2'. Other tweaks tend to cause only small alterations in the ratio. For example, when the per-length inductance of L_2' is increased, the effective coupling factor k' decreases so that (3.13) remains roughly the same.

Now that we can relate extracted parameters to circuit design values for inductive coupling, reliable *FastHenry* models for layout extraction are needed.

The method of images and the placement of the reflection plane are as valid for mutual inductance calculations as they are for self-inductance, so that the same approach to modelling self-inductance structures can be used for mutual inductance calculations in *FastHenry*.

Published results on superconducting coupled inductors are limited to near-unity coupling coefficient spiral inductors and washers – either for impedance calculations [45] or S-parameter modelling [46] – and straight line microstrip results against which to gauge the *FastHenry* 3D models were not available. Thus, in order to verify the *FastHenry* results for coupled microstrips, they were compared to the 2D numerical results obtained with *Induct*.

Table 3.6: Relative differences between inductance values for coupled microstrip lines calculated with *FastHenry* and *Induct*, where the *Induct* results are taken as nominal

Dimensions: L_1 is 6 mm × 40 mm, L_2 is 5.25 mm × 40 mm. Both lines have 5 width segments, each of which has a filamentation of 1×3 in width and height

Layers	L_1 [% difference]	L_2 [% difference]	M_{12} [% difference]	k [% difference]
M3 (L_2) over M2 (L_1)	-4.81	-4.96	-5.60	-2.40
M2 (L_2) over M1 (L_1)	-6.56	-3.74	-5.95	-2.48

Table 3.6 shows the difference between calculated inductance results for microstrip lines in the Hypres process. The *FastHenry* results (obtained through the method of images) are always lower than those of *Induct*, primarily because of the fringe effects in the 3D structures of the former, but also because the latter uses a finite ground plane. Once again the fringe effects are worse for levels closer to ground.

The differences between the 3D and 2D solutions in Table 3.6 agree very well with those observed for self-inductance (which are shown in Table 3.4). The conclusion is that the method of images with a reflection plane at l_{eff} for 3D models in *FastHenry* is equally effective for self- and mutual inductance calculations.

The technique discussed above was used to calculate all k values listed in this dissertation. Two other results also deserve mention.

In practical circuits, the maximum attainable coupling coefficient is limited by the construction process. With the Hypres process, a dc SQUID is almost always created with the loop inductance in layer M2, so that control lines are either in M3 or M1.

The maximum coupling coefficient between an M3 control line and an M2 SQUID loop obtained in this dissertation is 0.58, and requires the control line to intersect the M2 loop very close to, or even right above the Josephson junctions.

The design rule limitation on the proximity of a control line in M1 to the bottom pad of a Josephson junction (also in M1) leads to a lower attainable coupling coefficient. Here the maximum value obtained was 0.35.

Lastly, a 3D model was used to verify the assumed absence of coupling between perpendicularly crossing lines. A model containing an M3 line that perpendicularly crosses the transmission line of the JTL shown in Figure 3.19(b) was analysed, and the ratio of induced to control current (3.13) calculated as smaller than one in ten thousand – even for very wide M3 lines. This shows that the only parasitic component introduced by a perpendicular line crossing is a coupling capacitance [47].

3.8 UTILIZATION OF 3D INDUCTANCE MODELS IN DESIGN PROCESS

Currently, the inductance of every inductor in a layout is verified with the 3D method. This even applies to the sub-picohenry inductances of connections between neighbouring junctions such as a series junction connected to a grounded junction.

Layout is speeded by the reuse of common inductance structures, especially the 4 pH connection inductance between JTLs and other devices with matched 250 *m*A junctions.

If design shortcuts and inductance estimations are used, it is only because 3D simulations have shown them to be very good approximations in certain special cases, such as for straight or single-cornered microstrip lines with length-to-width ratios larger than 2, or connections to universal junction pads.

3.9 SIMULATED INDUCTANCE SPREADS

The development of full 3D models for inductance calculation enables us to study the true effects of manufacturing tolerances on inductance. The validity of the inductance spreads used in Monte Carlo circuit simulations can therefore be verified. A program was developed to automatically generate 3D inductance models with random variations on all the dimensional parameters from a single nominal structure.

Most intergate inductances are in the order of 4 pH, as is the total inductance between junctions J_1 and J_2 in the standard JTL (see Figure 3.19). This model of the inductance between junctions J_1 and J_2, including the effects of the resistor covers and the dc tee-in, was therefore used to calculate a representative figure of inductance spread.

The width of lines L_1 and L_2 was taken as 4 *m*m – the thinnest conductors used for inductive interconnections in any layout for this dissertation. This ensures that the inductance spreads reflect the worst-case possibility.

For the calculation of inductance spread, random variations according to a Gaussian normal distribution are applied to every layer thickness and conductor width in the specific structure. The penetration depth of a niobium thin-film, as well as the effective penetration depth of the ground plane, are also varied, and the results used to set the reflection plane. The new penetration depth is then also used for all the superconducting elements in the *FastHenry* input file.

Table 3.7 shows a generic model for the initialization of the appropriate dimensional parameters (actual implementation depends on the programming language used to generate the *FastHenry* input file). All values are for the 1 kA/cm^2 niobium process from Hypres [16].

The Gaussian function is represented here as *gauss(mean value, standard deviation).*

Table 3.7: Parameter declarations for the generation of 3D inductance models with random dimensional variations

Parameter	Initialization value [*m*m]
`Lambda_niobium`	`Gauss(1,0.05/3)*0.09`
`M0_thickness`	`Gauss(0.1,0.01/3)`
`M1_thickness`	`Gauss(0.135,0.01/3)`
`M2_thickness`	`Gauss(0.3,0.02/3)`
`M3_thickness`	`Gauss(0.6,0.05/3)`
`I0_thickness`	`Gauss(0.15,0.015/3)`
`I1_thickness`	`Gauss(0.1,0.01/3) + gauss(0.1,0.01/3)`
`I2_thickness`	`Gauss(0.5,0.04/3)`
`R_thickness`	`Gauss(0.1,0.02/3)`
`Lambda_effective_GP`	`coth(M0_thickness/Lambda_niobium)*Lambda_niobium`
`M2_line_width`	`Gauss(0,0.25/3) + Nominal_width`*

* Where `Nominal_width` is the nominal width of the conductor; 4 *m*m in this case.

The standard deviation (*s*) of each Gaussian function is obtained by dividing the worst-case variation specified in the Hypres design rules, often referred to as the 3*s* values [9], through 3. For a Gaussian random distribution, 99.74 % of all values lie within ±3σ of the mean [18].

When conductors in layers other than M2 are used, appropriate variables similar to `M2_line_width` are declared and used.

The results of simulations on the inductance of the JTL, using the random variations of Table 3.7, are shown as histograms in Figure 3.28. The histogram in Figure 3.28(a) represents the full process variations, and was compiled from the calculated inductances of 115 JTL structures, each generated with random variations on every layer thickness and line width, as well as on penetration depth. The histogram in Figure 3.28(b) shows the inductance calculation results for 100 JTL structures of which only the layer thicknesses and penetration depth were randomly varied, so that it represents the global variations on inductance.

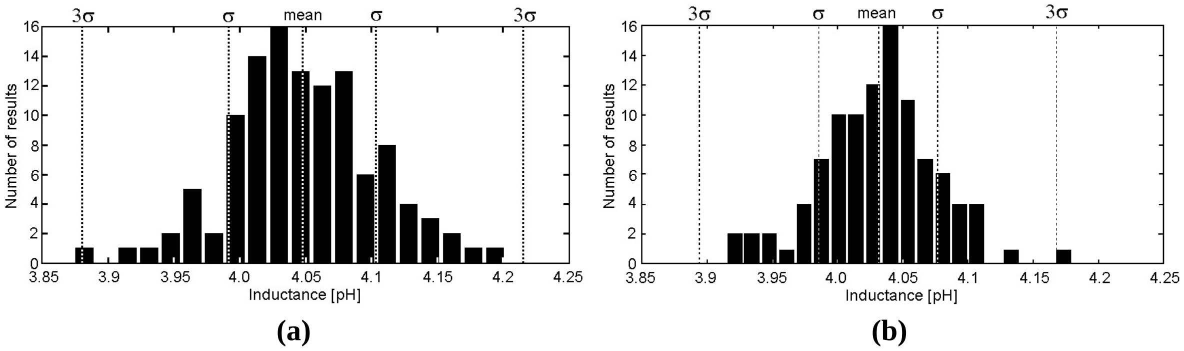


Figure 3.28: (a) Histogram of inductance spread about mean for JTL with full process variations and (b) with global (layer) variations only

From the data presented in Figure 3.28(a), the standard deviation of inductance is calculated as 1.38 %, and the $3s$ limit as 4.14 %. This includes the effects of global (layer) and local tolerances.

Figure 3.28(b) shows the inductance spread for global (layer) variations only. The distribution appears to be Gaussian, as is expected. The standard deviation is calculated as 1.13 %, yielding a $3s$ limit of 3.39 %.

From (2.26), the standard deviation caused by local variations for this JTL is calculated as 0.784 %. The 3σ limit is 2.35 %.

These results are well within the worst-case values predicted in Table 3.1, and show that the simulation values in Table 2.2 are more than adequate.

3.10 CONCLUSIONS

Some inductance calculation techniques [25] use tables generated from several calculation runs to establish relationships between line dimensions and inductance. In real RSFQ structures, all important inductances will always terminate in Josephson junctions. The large pads associated with the JJ structures will inevitably cause large deviations from the expected inductance values of especially short lines, and can therefore not be ignored. It is proposed that all calculations for small inductances, especially intra-gate transmission lines, be performed on structures that are terminated at their ends in Josephson junction structures. Furthermore, since it is supported, all structures such as tee-ins and line crossings that directly affect (by proximity) the inductance of an intra-gate line should be included in the *FastHenry* models used for inductance extraction. This lowers the uncertainty in the inductance values, and can allow highest-yield layouts to be realized.

Exhaustive simulations and the comparison of results calculated with different techniques have showed that segmentized 3D structures using the method of images yield reliable inductance results in *FastHenry*.

Yet, even for sufficient discretization, *FastHenry* is bound to produce inaccuracies. For structures of similar size and segmentation, however, very good relative answers (such as percentage change) can be obtained.

Other numerical inductance calculation techniques have been devised or tried by several researchers. Some reduce computing time and memory requirements through simplifications of the MOM model. Others, like an FDTD technique, require more time and memory as a result of the need to discretize the space between conductors too. In the end, all these techniques are approximations to a problem that is not unique to superconductors, but extends to all fields of VLSI design. The inductance of complex structures cannot be solved analytically, and eventually the design engineer still needs to select a numerical technique that will give a fairly accurate answer in an acceptable time, given the processing power at his disposal.

Since there is no clear advantage regarding accuracy in any of the numerical techniques mentioned above, selection merely depends on calculation speed and the ease of implementation.

3D analyses are time-consuming, both when the 3D models are constructed and when *FastHenry* performs calculations. It is therefore a good design principle to reuse inductors in layouts. When certain inductance values occur very often, such as 2 pH and 4 pH in the RSFQ circuits in this dissertation, a set of interconnection geometries can be created beforehand, and an appropriate geometry selected whenever the specific inductance value is needed.

In this chapter, the use of *FastHenry* and newly developed segmentation routines to model any 3D structure for the extraction of inductance was discussed. Segmentation, filamentation and the placement of the reflection plane for the method of images were all treated in detail, and comparative results used to measure and validate the effectiveness of each. The results presented in this chapter also give an indication of the power of the 3D models, especially when structures contain unconventional discontinuities or are partially coupled to other structures.

Once the accuracy of the models were sufficient, full gate structures were investigated. 3D structures of a complexity not previously attempted for superconducting circuit elements can now be analysed, and mutual and self-inductance values extracted. Eventually, models with dimensional variations according to process tolerances were analysed to establish the real effects that these tolerances have on inductance. It is now possible to calculate the true inductance variations, brought on by manufacturing tolerances, in any integrated circuit structure.

Chapter Four - Novel component design

“Here’s just the thing: just before we’re sent back into the world, the Goddess Meng administers to us a vial of forgetting.”
“I don’t remember that,” Keeper said.
Kim Stanley Robinson, The Years of Rice and Salt

4.1 INTRODUCTION

ALTHOUGH the RSFQ logic family is by now well established, new gates and latches continue to be added to the collection of known cells. Many of these are created through changes to existing cells. An example is the T^2 flip-flop [48] – designed for multiplexing – that is based on the B flip-flop [2] with joint inputs (which is also used for multiplexing [49]).

In this project, a few latches with specialized properties were needed to implement memory functions that would otherwise require combinations of several other standard gates and latches.

These novel latches are also derived from changes to the structures of existing latches, but always with a specific function in mind.

Josephson junctions in circuit diagrams are always indicated by the symbol *B*, after the symbols used in Steve Whiteley’s *WRSpice* [8].

4.2 RSFQ ELEMENTS

4.2.1 DC-Resettable Latch

4.2.1.1 RAISON D'ÊTRE

The first and foremost requirement when reprogrammable logic circuits are designed is the availability of a memory element that can withstand (theoretically, at least) an infinite number of read cycles without data loss.

The secondary requirement is non-volatility. The memory element should be able to retain stored data if dc power is lost either momentarily or for a prolonged period.

A third requirement is obtained from an engineering perspective on the design of large scale reprogrammable circuits. Apart from heating a circuit beyond the critical temperature to destroy stored flux loop currents, a way is needed to electronically erase all elements on a chip or in a memory block. The circuit complexity and die space occupation of RSFQ pulse distribution circuitry dedicated to the reset of programmable elements would make such a technique impractical. A more effective strategy is to bias a single dc line that threads every memory element to induce switching currents and reset all the elements in a single operation.

The first requirement already ruled out the popular and robust Destructive Readout register (DRO) [1] [2], leaving only the Non-Destructive Readout register (NDRO) [2] as a viable candidate. However, the NDRO is not dc resettable, and a new logic circuit, the DC-Resettable latch (DCRL) was designed using the basic structure of the Likharev NDRO [1] as a starting point. Once a functional nominal circuit was obtained, genetic optimization was used to improve the yield and storage stability of the latch.

4.2.1.2 FUNCTIONAL DESCRIPTION

The circuit diagram of the DCRL is shown in Figure 4.1. The element values shown here were derived through a genetic optimization process, except for L_{13} and the coupling coefficient between L_{13} and L_2, which were both extracted from the final circuit layout.

The DCRL consists of a set-reset stage built around a dc SQUID (B_2 and B_3), and a read stage of cascaded JTLs (B_5, B_6, B_8 and B_9).

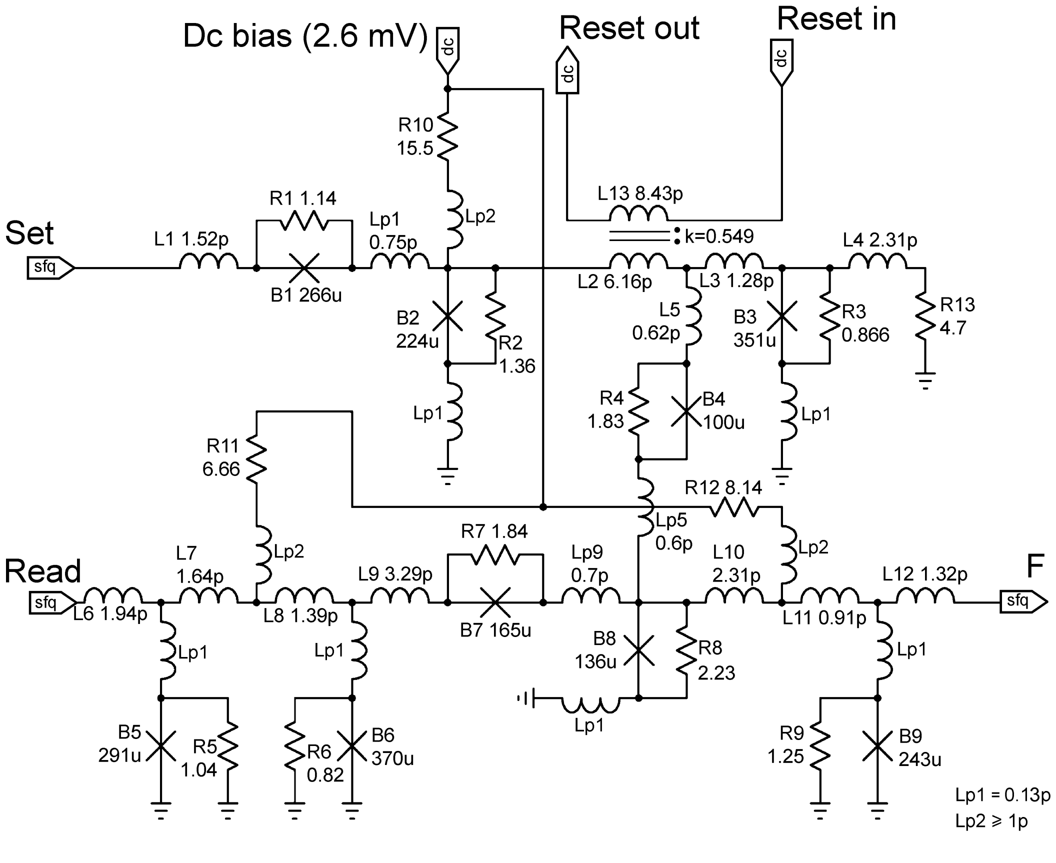


Figure 4.1: Circuit diagram of RSFQ DC-Resettable latch

In the unset state, junction B_2 sinks most of the bias current supplied through R_{10}, while current path B_4-L_5 pumps bias current away from B_8 and into B_3. A *Read* input pulse – amplified by B_5 and B_6 – finds B_8 unbiased and switches series junction B_7 instead, so that no output pulse is produced at *F*.

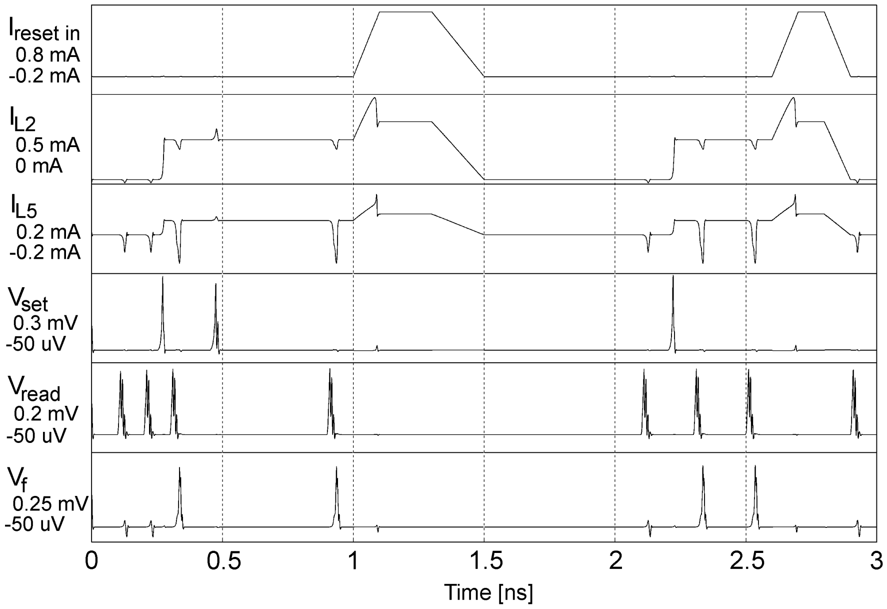
When an SFQ pulse is applied at *Set*, B_2 switches and current is forced to circulate clockwise through L_2, L_3 and B_3. A sufficient fraction of the current is diverted through B_4 and B_8, adding to the bias current from R_{12}, so that B_8 is properly biased at $0.84I_C$. The latch is now set. Further set pulses find B_2 unbiased and switch series junction B_1 instead.

In the set state, a *Read* input pulse proceeds to B_8, which is sufficiently biased to allow switching and pass an SFQ pulse to the output at *F*. Series junction B_4 prevents the switching of B_8 from affecting the current in L_2 and L_3, and the set state persists.

The DCRL is reset by applying a dc control current to *Reset In*. The control current couples to L_2, and the polarity is chosen so that an increase in the control current induces an increase in the current circulating clockwise through L_2. This eventually causes B_3 to switch, and returns the DCRL to the unset state. The SFQ pulse generated by the switching of B_3 is dissipated in R_3 and R_{13}.

Simulation results for the DCRL are shown in Figure 4.2. The dc reset current is 720 *m*A, which is sufficient to reset the latch when no *Read* input pulses are present. If *Read* input pulses are present during the reset operation, the dc reset current can be lowered to 350 *m*A without causing reset failures.

In the simulation results, three *Read* input pulses are applied from about 100 ps to 300 ps. The DCRL is only set after 250 ps, so that only the third *Read* pulse yields an output at *F*. A second *Set* pulse just before 500 ps tests for a possible state change error. Another *Read* pulse at 900 ps is also passed, before the DCRL is reset between 1 ns and 1.5 ns. A *Read* pulse at 2.1 ns then yields no output at *F*. After 2.2 ns, another set-reset cycle is simulated.



Positive terminal for L_2 is next to B_2. Positive terminal for L_5 is at L_2-L_3 tee-junction.

Figure 4.2: Simulated transient response of RSFQ DCRL

Although not shown in Figure 4.2, the dc bias voltage of the DCRL can be removed without destroying the flux in the dc SQUID loop B_2-L_2-L_3-B_3. When the dc bias voltage is restored, all currents return to their correct values with the set state remaining intact. The DCRL is thus a non-volatile memory element.

The first implementation of the DCRL had a yield of only 22 %. It was the first circuit to be optimized with the genetic algorithms discussed in Chapter 2, and a very good example of how powerful the optimizer can be. The optimized circuit has a yield of 100 % with layout extracted values and tolerances, and with voltage trimming. This and other final yield results of all the novel circuits are shown in Table 4.1 on page 67.

4.2.1.3 ADVANCED ADD-ON: FALSE RESET PROTECTION

Monte Carlo simulations on the unoptimized DCRL have shown that it may erroneously reset when an input read pulse leaks through B_4 to the main flux loop. The reset action always requires B_3 to switch, and the resulting output pulse can be used as an alarm signal to off-chip logic, or to set the DCRL again, or both.

A loop reset protection circuit is shown in Figure 4.3, and element designations correspond to those in Figure 4.1. The output of the junction B_3 of the DCRL is connected to a Current-Set switch (see section 4.2.2) that only allows the pulse to pass when a feedback activation control current flows. From there, the pulse proceeds through a pulse splitter. One path leads to fault reporting logic, and the other feeds back to merge with the standard set input of the DCRL.

If the Current-Set switch, pulse splitter and merger circuits (all of which are extremely stable) do not fail, any false reset will feed back to set the DCRL again. Before a programmed reset occurs, the feedback activation current is removed so that the Current-Set switch can suppress the pulse generated by B_3.

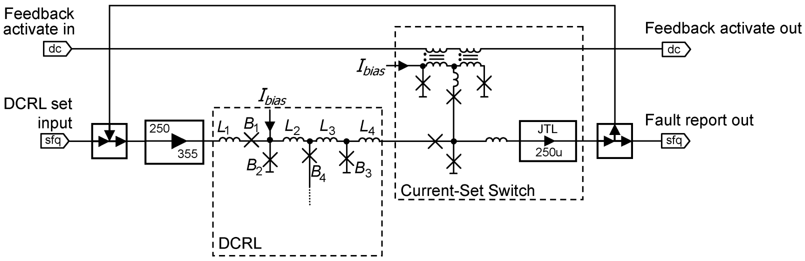


Figure 4.3: Schematic diagram of fault feedback for DCRL loop reset protection

This pulse feedback technique was not added to the design of the PFD discussed in Chapter 6, as the final DCRL has a theoretical yield of 100 %. It was decided to test a physical circuit first and determine if false resets do occur.

4.2.2 Current-Set switch

In reprogrammable circuits, every memory element must be uniquely addressable. Normally, such elements are used as cells in a matrix of row and column programming lines. The use of dc lines and write currents to induce current in memory elements – usually dc SQUIDS or three-junction interferometers – simplifies the design of large circuits. Such dc write currents have been used successfully in the construction of superconducting memory cells and arrays [50] [51] [52].

In this dissertation the programming grid for a programmable circuit is composed of dc current lines for column selection, and SFQ lines for row selection (a clear demonstration of which is shown in Figure 5.10, p. 75). An element is therefore needed that will only allow a received SFQ pulse to pass if it is activated by a control current.

The Current-Set switch shown in Figure 4.4 was designed specifically for this task. The read stage is a cascade of JTLs. The input JTL is stripped down to a single junction (B_5) and the output JTL (B_6 and B_7) performs pulse shaping and matching.

The set stage is formed with a modified dc SQUID (B_1 and B_2) of which the bias current injection point is next to B_2. The SQUID loop (L_1 and L_2) is inductively coupled to a control line (L_3 and L_4).

In the unset state (when no current flows through the control loop L_3 and L_4), junction B_5 is unbiased. An input pulse at *Read* switches the series junction B_4, and no output pulse appears at *F*. The input pulse is thus suppressed or stopped.

When sufficient current is applied at *Set In* (the design specification is for 200 mA or more), the clockwise current induced in the loop L_1-L_2 adds to the bias current flowing through B_2 and causes it to switch. This diverts a constant current of about 200 mA through L_5, B_3 and B_5. The switch is now set.

In the set state, a *Read* input pulse switches the biased junction B_5, and the SFQ pulse propagates to the output *F*. The switch is closed, and any amount of input pulses can pass through.

Switching hysteresis requires the control current at *Set In* to go negative before the SQUID loop will reset, so that the Current-Set switch requires a bipolar control current.

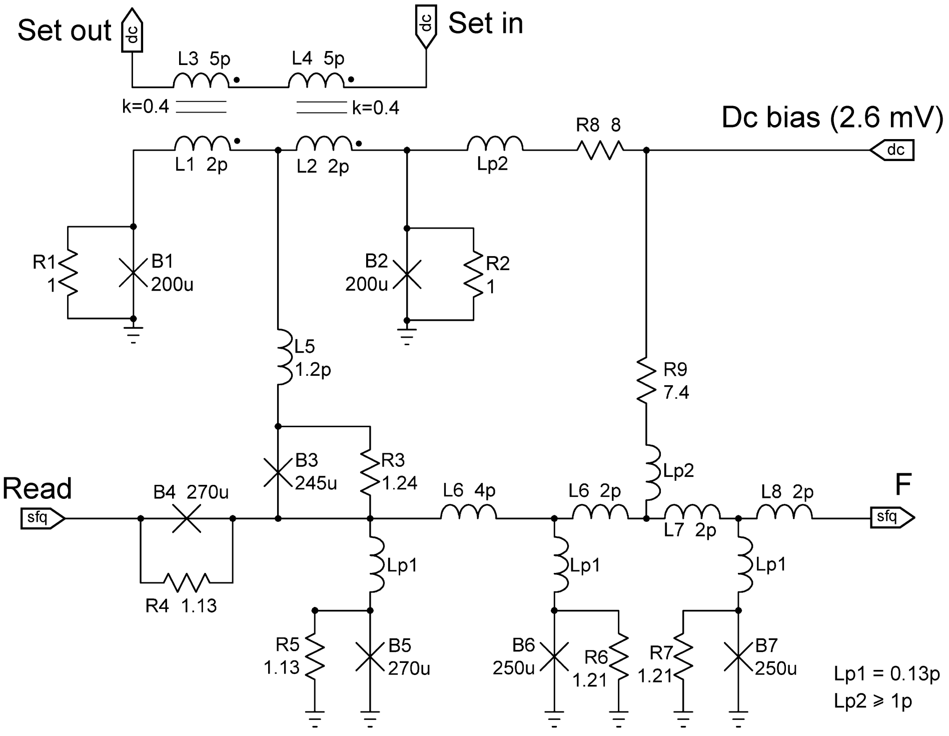


Figure 4.4: Circuit diagram of RSFQ Current-Set switch

Simulation results for the RSFQ Current-Set switch are shown in Figure 4.5. The simulation results clearly show how input pulses are stopped or passed when the Current-Set switch is in the set or unset conditions. The bipolar control current at *Set in* swings between +330 mA and –330 mA, and was generated by a HUFFLE (section 4.4.1). *Read* input pulses were applied in pairs, with the pulses in a pair 100 ps apart, to test for false state changes or sluggish pulse repetition.

The first event, at 150 ps, is when the HUFFLE is preset at start-up (see section 4.4.1). After that, the Current-Set switch is unset, and *Read* pulses are stopped. The switch is set at 1.6 ns, after which two *Read* pulses are passed to *F*. The switch is then reset, and set again right between two *Read* pulses at 4.25 ns. The first pulse is stopped, while the second is passed, and the Current-Set switch therefore functions correctly even at 10 GHz.

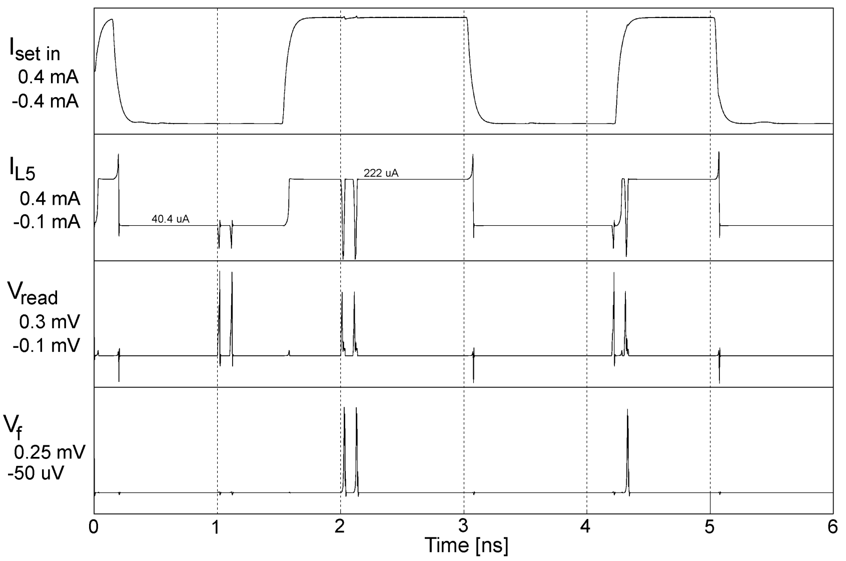


Figure 4.5: Simulated transient response of RSFQ Current-Set switch

4.3 COSL ELEMENTS

4.3.1 Set-Reset flip-flop

The COSL Set-Reset flip-flop is included in this discussion because it was developed as a novel circuit, formed part of the initial set of test circuits used to evaluate the success of the genetic optimization strategy, and forms the basis of a proposed new circuit for the conversion of SFQ pulses to clock-synchronized voltage state logic.

The schematic circuit diagram is shown in Figure 4.6. It is modelled on the standard COSL OR-gate, but there are some important differences.

When the flip-flop is set, it must remain so until a reset signal arrives. In standard COSL gates, two clock signals, 120° out of phase, are used. A COSL input arrives at the peak of the input clock signal, switches the input one-junction SQUID, and diverts control current through an inductor that couples with a dc SQUID. The negative cycle of the input clock resets the input one-junction SQUID.

In the SR flip-flop, the one-junction SQUID is replaced with a two-junction SQUID (B_1-L_1-L_2-B_4). The input clock is replaced with a dc bias voltage that supplies about 220 *m*A through R_7, of which around 200 *m*A flows into B_1 to bias it at $0.8I_C$.

When a COSL input signal is applied through R_3, junction B_1 switches, and the two-junction SQUID is set with current circulating clockwise. Since there is no negative clock cycle to reset the two-junction SQUID, the flip-flop can retain the set state indefinitely. An added advantage to the continuous input bias is that the SR flip-flop can be set asynchronously – unlike ordinary COSL gates.

When the SR flip-flop is set, current flowing through L_1 and L_2 couple into the readout two-junction SQUID formed by B_2-L_3-L_4-B_3. This stage of the SR flip-flop is identical to that of other COSL gates, except for resistor value changes to allow more current into the output two-junction SQUID. When the flip-flop is set, a clock signal at *Clock* causes the output two-junction SQUID to switch and produce a high output at *Q*.

If the SR flip-flop is in the set state, the circulating current through B_4 biases the junction sufficiently for a reset signal applied through R_{10} to cause it to switch. This resets the input two-junction SQUID, and returns the bias current to B_1. The *Reset* signal can also be applied asynchronously.

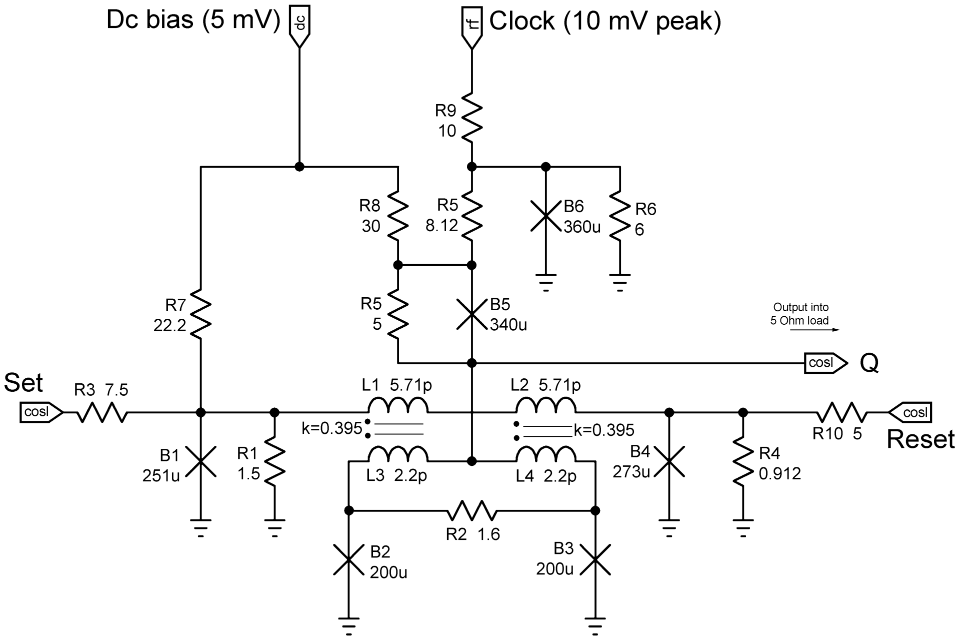


Figure 4.6: Circuit diagram of COSL Set-Reset flip-flop

Simulation results for the COSL SR flip-flop are shown in Figure 4.7. After a *Set* input, an output signal at *Q* appears at every clock cycle. The second *Set* input is used to test for erroneous state changes. After the *Reset* input, no more outputs appear at *Q* until the flip-flop is set again.

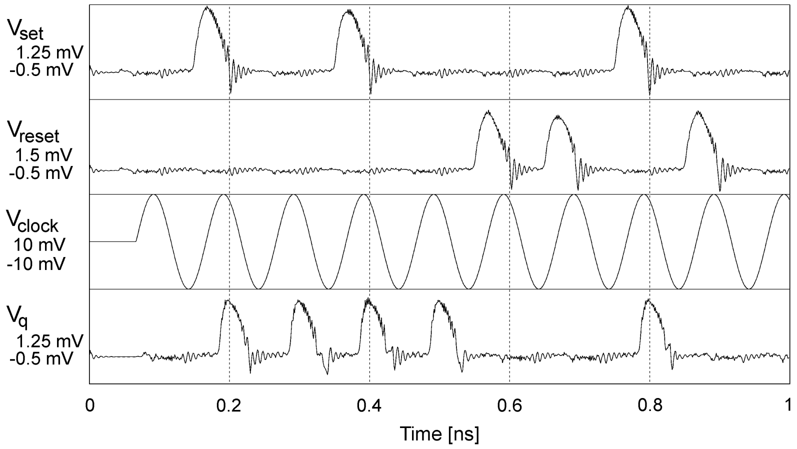


Figure 4.7: Simulated transient response of COSL SRFF

4.3.2 *Toggle flip-flop*

Although better implementations can probably be found, a basic Toggle flip-flop is formed when the *Set* and *Reset* terminals of the COSL SR flip-flop in Figure 4.6 are connected together, and the clock current reduced by around 30 %.

When an input signal is applied, the state of the T flip-flop is inverted. The only practical difficulty is that the combined *Set* and *Reset* inputs require more current than the

200 *m*A that can be delivered with a single COSL output, so that parallel COSL OR-gates are needed to drive the T flip-flop.

The circuit was optimized as an SR flip-flop, and does not really have a good yield as a T flip-flop. However, since it does not form part of the tool kit required for reprogrammable RSFQ circuitry, the T flip-flop was not investigated or optimized further.

Simulation results for the COSL T flip-flop are shown in Figure 4.8. The *Toggle* input feeds both the *Set* and *Reset* pins of the standard COSL SR flip-flop. One clock cycle after a *Toggle* input, the output at *Q* is inverted.

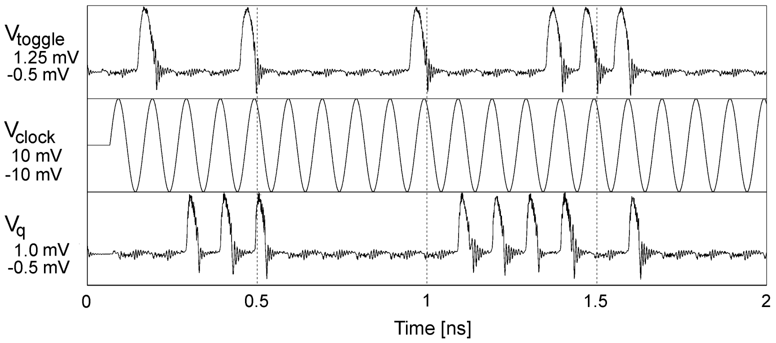


Figure 4.8: Simulated transient response of COSL Toggle flip-flop

4.4 HYBRID LOGIC ELEMENTS

4.4.1 HUFFLE bipolar current switch

When many current lines are required to handle switch addressing, it is impractical to drive each from an off-chip current source. An element is therefore needed that can translate an SFQ address pulse into a dc current.

The most practical element is the Hybrid Unlatching Flip-Flop Logic Element (HUFFLE) [53] [54] [55] [56] [57].

The HUFFLE is ideal for implementing a bipolar current switch. Unfortunately, only the HUFFLE described in [57] uses RSFQ control circuits. However, their implementation uses three-junction flux loops, and it was decided to redesign the standard HUFFLE for RSFQ compatibility – effectively creating a new circuit.

4.4.1.1 FUNCTIONAL DESCRIPTION

Figure 4.9 shows the final circuit diagram of the HUFFLE as it was implemented in the PFD (see Chapter 6). Element values for the set and reset input sections were derived from standard RSFQ gates. The coupling inductances, coupling coefficients and damping resistor inductances L_{S3}, L_{S4} and L_{S9} in Figure 4.9 were obtained through layout extraction, and differ slightly from the values found through genetic optimization (which are not shown here).

The HUFFLE is a bipolar current device, and the positive and negative terminals of *I out* are connected together through a superconducting line that may have a high inductance.

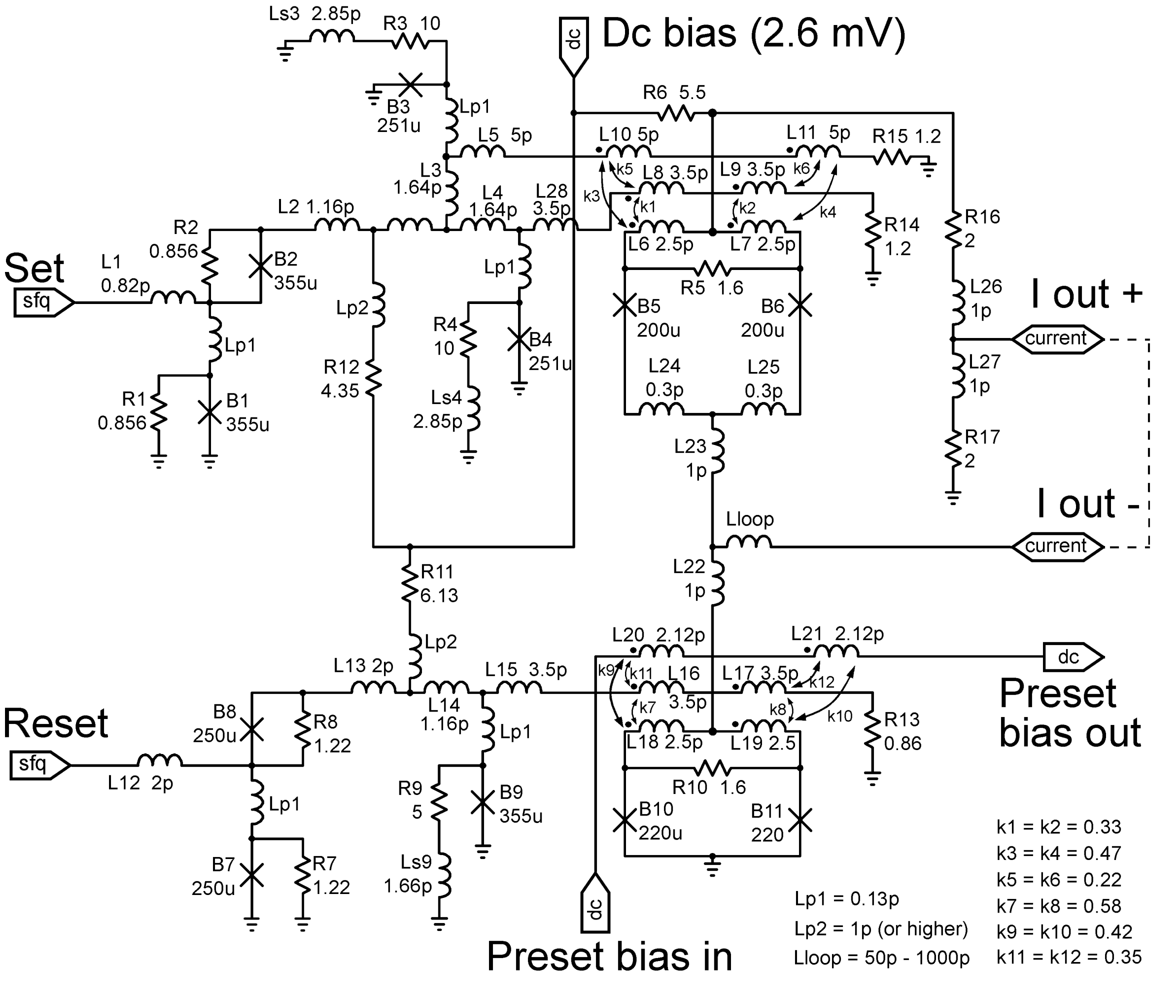


Figure 4.9: Circuit diagram of HUFFLE

Bias current entering through R_6 flows to ground in one of two ways: either through the top dc SQUID (B_5 and B_6), L_{23}, L_{loop} and via *I out* (negative to positive terminals), L_{27} and R_{17}, or through R_{16}, L_{26}, via *I out* (positive to negative terminals), L_{loop}, L_{22} and the bottom dc SQUID. The former corresponds to the unset state, and the latter to the set state.

Current applied to *Preset bias* flows through L_{20} and L_{21}, and is used to initialize the HUFFLE into the unset state. After the initialization pulse, the current is lowered to an idle value (not zero) that assists with further reset actions.

The HUFFLE is set through the application of an SFQ pulse to *Set*. The pulse passes through the buffer junction B_2 and is split into two separate SFQ pulses by junctions B_3 and B_4. Both pulses induce current into L_6 and L_7, and the combined induction current switches the dc SQUID (B_5 and B_6), and causes *I out* to reverse direction and flow through the bottom dc SQUID.

Reset occurs when an SFQ pulse enters *Reset*, passes through the buffer junction B_8 and the amplifier junction B_9, and induces a switching current in L_{18} and L_{19}.

The control current driver junctions B_3, B_4 and B_9 may appear to be underdamped because of the large values of their damping resistors. This is merely a way of diverting more current through the inductive paths to their loads, and each load is designed to facilitate damping. The larger control currents increase the circuit's theoretical yield.

Simulation results for the HUFFLE are shown in Figure 4.10. The *Set* and *Reset* pulses precede state changes, where a positive output current represents the set state, and a negative current the unset state. The preset bias current is ramped up to 950 *m*A at initialization to force the HUFFLE into the unset state, after which it is kept at an idle 143 *m*A. It can also be seen at 5 ns that the preset bias current can be used to reset the HUFFLE during normal operation.

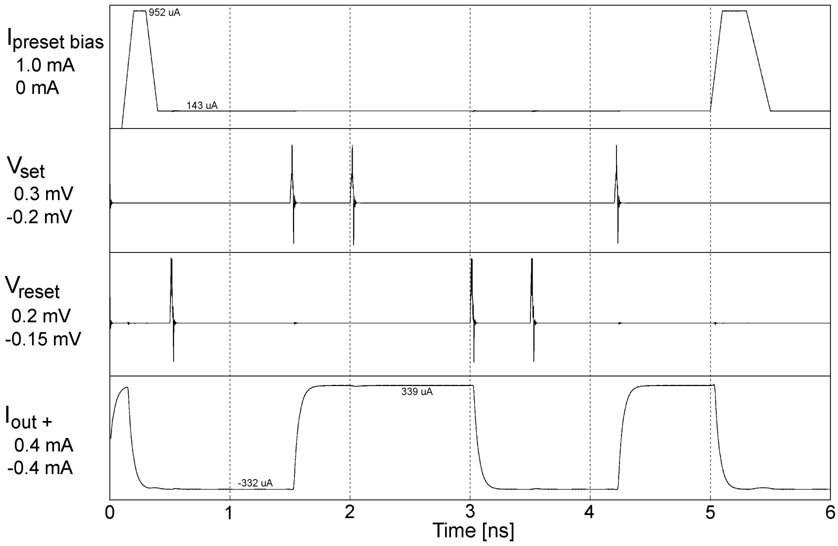


Figure 4.10: Simulated transient response of HUFFLE

4.4.1.2 OTHER USES FOR HUFFLES

Although the HUFFLE was originally developed as a memory element, its primary implementation here is as a bipolar current driver for memory element selection. It will therefore be used at the back end of address decoding logic.

The HUFFLE can be put to other use too. It can for instance be used as an interface to off-chip hot logic [57]. It was used in this way in the compact PFD to be discussed in Chapter 6. For such an implementation, one end of the loop inductance L_{loop} is connected to a cascade of transmission lines and matching resistors that eventually feed the voltage developed across R_{16} and R_{17} into a hot 50 Ω load (see Figure 6.7 on page 90).

Since this voltage signal is in the order of a 1 $\text{mV}_{\text{peak-peak}}$, (Figure 6.8) the 50 Ω load should ideally be the input port of a low-noise amplifier. Matching is also important, as simulations showed that reflections off a mismatched load could spontaneously flip the state of the HUFFLE.

The HUFFLE can also be configured as a T flip-flop. For this operation, the *Set* and *Reset* inputs can be driven simultaneously (through a pulse splitter). This setup is also illustrated in Figure 6.7, and was used in conjunction with the hot logic interface technique discussed above to provide a secondary rf output port for a compact PFD. The T flip-flop dynamics are shown in Figure 6.8.

4.4.2 RSFQ-to-COSL converter

SFQ pulses are virtually invisible to hot logic circuits, so that several techniques have been devised to translate these pulses into voltage state signals. One such a technique was discussed in the previous section, and in [57].

Since COSL is a voltage state logic family, it is a prime candidate for RSFQ to hot logic interfacing. A promising circuit that has previously been studied is the DRO-to-

COSL gate [3]. Unfortunately, this circuit requires a full balanced three-phase clock signal, and requires the RSFQ DRO to supply its SFQ pulse in a very tight time window.

As part of this project, a new RSFQ-to-COSL converter was designed. The schematic circuit diagram is shown in Figure 4.11.

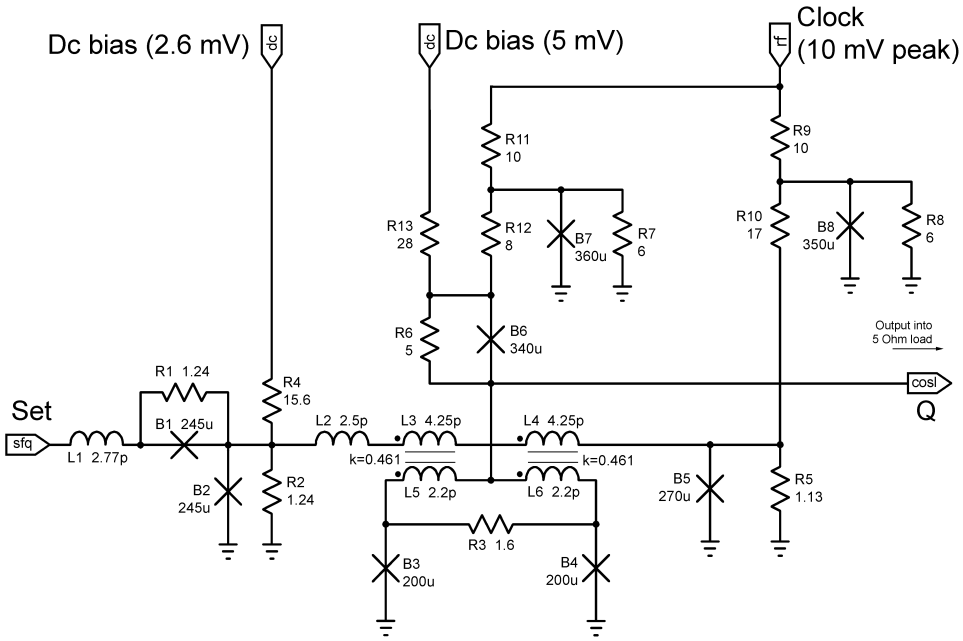


Figure 4.11: Circuit diagram of RSFQ-to-COSL converter

The design is very similar to that of the COSL SR flip-flop discussed in section 4.3.1, except that the input two-junction SQUID is replaced with that of the standard RSFQ DRO (B_2-L_2-L_3-L_4-B_5). So, instead of interfacing a DRO to a COSL OR-gate, the DRO is *integrated* into a COSL gate.

The input two-junction SQUID retains the functionality of the RSFQ DRO, including asynchronous operation and double-set protection via buffer junction B_1. The only limitation is that a *Set* input pulse may not arrive during most of the positive half cycle of the sinusoidal *Clock* input.

Once the DRO is set, control current circulating through the inductive loop formed by L_2, L_3 and L_4 couples into the output two-junction SQUID formed by B_3, L_5, L_6 and B_4. Read-out is performed with a sinusoidal clock as in all other COSL gates.

For use as output buffer, the RSFQ-to-COSL converter must reset itself after a read operation. Since the SFQ input pulses can arrive asynchronously, circuit overhead is reduced if the sinusoidal clock signal doubles as a reset signal. Resistors R_9 and R_{10} feed current from the clock signal to junction B_5, while B_8 acts as a clock shaper. If the DRO stage is in the set state, the current through R_{10} causes B_5 to switch and the gate to reset.

Simulation results for an RSFQ-to-COSL converter driven by a JTL and loaded by a 5 Ω resistor are shown in Figure 4.12. The first two *Set* pulses arrive at the same time during their respective clock cycles. The third *Set* pulse is shifted forwards and the fourth pulse backwards to show the time range over which SFQ input pulses can be accepted and correctly converted to COSL signals.

The RSFQ-to-COSL converter was not used in the compact PFD discussed in Chapter 6, and is therefore not included in the symbol key in Figure 5.1. However, it is used twice in the chip layout shown in Figure A.17, where it can be tested as an individual element before being included in future superconducting circuits.

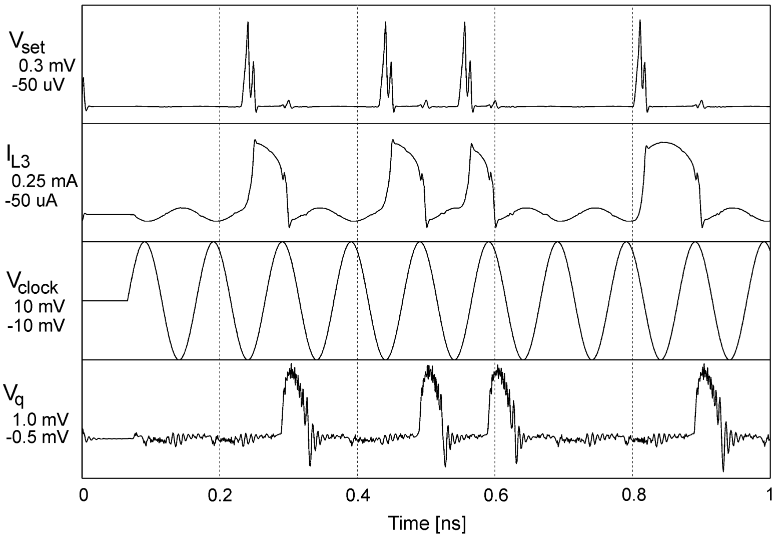


Figure 4.12: Simulated transient response of RSFQ-to-COSL converter

4.5 CONCLUSION – REVIEW OF THE RECURSIVE DESIGN PROCESS

This section concludes the first half of this dissertation, where the low-level design of RSFQ circuits was explored and characterized. The rest of this dissertation builds on the design foundations laid up to here when the design of complex systems is investigated.

In this chapter, latches with non-destructive read functionality and asynchronous reset ability were developed and optimized. These latches perform very well, and will form the basis of reprogrammable systems developed in Chapters 5 to 7.

As an overview of the recursive low-level design process, the development of the HUFFLE is reviewed.

The first implementation of the HUFFLE had an undesirable yield which could not be optimized beyond 55 %. The yield limit was overcome through the use of a new design that utilizes dual-line current feeds on each arm. One of these lines is used for biasing the HUFFLE and presetting it to the default reset value after switch-on. The new architecture was eventually optimized to a yield of 95 %.

However, neither the parasitic effects nor the achievable coupling factors for the mutual inductances were yet known.

For layout, the HUFFLE was structured in order to reduce parasitic inductances. For reasons of practicality, the main inductance loops had to be in M2. This left M1 and M3 for the dual-arm control lines. A 2D analysis showed that the coupling factors could be achieved, but layout according to design rules yielded problems. The best achievable layout was modelled for inductance extraction with *FastHenry*, and it was found that the induced current ratios fell far below the design values. (See section 3.7 for a discussion on the calculation of the induced current to control current ratio.) It also showed finite (and significant) values for the parasitic coupling between the M1 and M3 inductors.

All the extracted values were fed back into the *Spice* simulations, and the actual yield calculated as a dismal 38 %. Analyses showed that not enough current was induced in the set loop of the HUFFLE to allow the control SFQ pulses to switch the HUFFLE.

Several improvements were considered, but most had to be rejected on grounds of layout impracticalities. The eventual solution was to move the damping resistors of the junctions that drive the control currents to the other end of the inductive control lines, so that the junctions would be forced to damp themselves through the control lines, thereby increasing the control currents. For stability, large resistors were added next to each driver junction – 10 Ω for the 250 *m*A junctions and 5 Ω for the 355 *m*A junction. This increased the current diverted through the control lines, but led to flux trapping in the input JTLs of the set and reset stages. The net effect was to decrease the yield, but this was negated when reverse buffering junctions were added to prevent flux trapping.

These changes pushed yield up to 96 %, but another round of layout extractions showed that the large resistors also had large inductances. For the 10 Ω resistor the inductance was calculated as 2.85 pH, compared to 1.66pH for the 5 Ω resistor. Substituted back into the Monte Carlo simulation model, these inductances decreased yield to about 88 %. With voltage trimming and layout extracted *tolerances*, this increased to 98.5 % – high enough to make the HUFFLE a practical circuit.

Table 4.1: Simulated theoretical yield of new circuits with various MC models

Circuit name	Generic MC model (on layout extracted elements) [%]	Generic MC model with voltage trimming [%]	Layout MC model with voltage trimming [%]
RSFQ DCRL	$87.83^{+2.74}_{-2.74}$	$99.94^{+0.06}_{-0.15}$	100
Current-Set switch	100	100	100
COSL SRFF	$76.09^{+2.71}_{-2.71}$	$90.12^{+1.89}_{-1.89}$	$95.54^{+1.31}_{-1.31}$
HUFFLE	$87.93^{+2.73}_{-2.73}$	$97.61^{+1.28}_{-1.28}$	$98.54^{+1.00}_{-1.00}$
RSFQ-to-COSL converter*	$82.51^{+2.41}_{-2.41}$	$91.91^{+1.73}_{-1.73}$	$94.65^{+1.43}_{-1.43}$

* The RSFQ-COSL converter was reoptimized after the extracted model of the final layout in Figure A.6 delivered yields of 68.0 ± 2.96 %, 80.67 ± 2.50 % and 85.78 ± 2.22 % respectively (as a result of a low coupling coefficient).

The generic MC models in Table 4.1 already incorporate the layout extracted values for inductance and mutual coupling. This, in conjunction with the change in design rules (and tolerance specifications) that occurred after the genetic optimization sequences were completed, accounts for the differences between the results reported here and those given for the genetic optimization routines in Chapter 2.

The last column in Table 4.1 lists the results for MC analyses containing not only layout extracted inductance and coupling values, but also extracted tolerances.

Chapter Five - Implementation of Programmable Frequency Divider

Divide et impera. (Divide and rule.)

Attributed to Philip of Macedonia, Julius Caesar, Niccolò Machiavelli and Louis XI of France

Divide and rule, a sound motto. Unite and lead, a better one.

Johann Wolfgang von Goethe

5.1 INTRODUCTION

THE design and construction of a programmable frequency divider, or PFD (not to be confused with phase/frequency detector used in some rf texts) serves quite a few useful purposes. Primarily, however, it is a component used widely for frequency synthesis in communication electronics [58] [59].

Frequency dividers in frequency synthesizers are used to divide the high frequency output of a voltage controlled oscillator by a programmed number for comparison to that of a stable reference oscillator. With prescalers (high frequency dividers forming the first stage of a cascaded divider) able to operate at up to 50 GHz, such as current RSFQ logic circuits, frequency synthesis well into the millimetre band becomes easy.

The development of a PFD would also allow the use and verification of all the elements and routing structures needed for a more ambitious superconducting programmable gate array. It is thus a circuit that will fit (albeit tightly) onto a 25 square millimetre die (3 micrometre process), and, most importantly, can be programmed and tested at any convenient frequency without the need for scarce and expensive measurement equipment.

This chapter deals with the design of such a PFD as a precursor to a superconducting programmable gate array (SPGA), and introduces the switch blocks and programming structures needed for reprogrammable logic ICs. The remaining considerations for the design of full SPGAs are discussed in Chapter 7.

It must be stressed that the PFD treated here is not designed for compactness or simplicity, but specifically with the verification of SPGA concepts in mind. It is therefore primarily a programmable gate array (PGA), with frequency division only of secondary importance.

Circuit diagrams are kept clear from unnecessary clutter by the use of simple symbols, as defined in Figure 5.1. Dc bias connections, with the exception of preset and reset lines, are omitted.

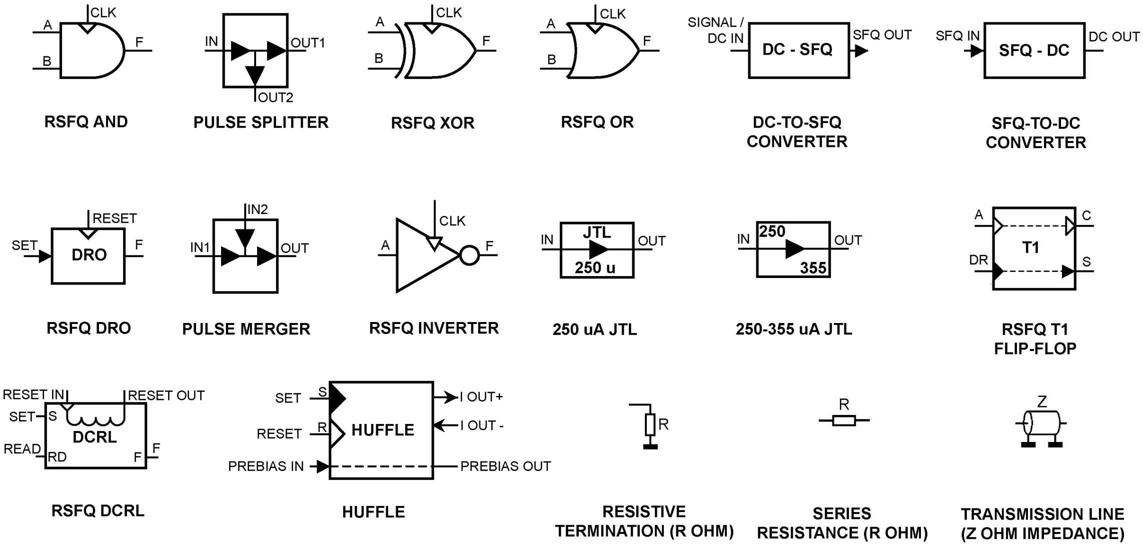


Figure 5.1: Symbol key for schematic circuit diagrams

5.2 THE FPGA AS FOUNDATION TO REPROGRAMMABLE CIRCUITRY

The field-programmable gate array [60] has become the most versatile and popular low-cost integrated circuit for low-volume niche digital applications, and is indispensable in systems that may need rapid reconfiguration or frequent architectural updating.

The composition of an FPGA is simple, yet highly effective, and can therefore serve as the foundation of a superconducting programmable gate array.

The basic structure of an FPGA consists of input and output ports, configurable logic blocks (CLBs) and interconnection resources. A schematic diagram for a general FPGA is shown in Figure 5.2 (modified from [60] and [61]). Crosses indicate programmable switches.

The ports facilitate off-chip interfacing while the logic blocks contain the *architecture* of the FPGA [60].

Implementation of an FPGA in RSFQ is has been considered before [61], but only as a high-level simulation and without addressing low-level design problems. Complications are introduced by the demand for clock signals and synchronization in RSFQ, as well as by the nature of JTLs, and some inventive concepts and designs are necessary to implement a practical SPGA.

The biggest difference between semiconductor and superconductor PGAs lies in the interconnection resources, since the bidirectional wires and MOSFET switches used in semiconductor FPGAs are not available in superconductor architectures.

One way to handle the unidirectional lines inherent to RSFQ circuits is to dedicate entire routing channels (containing many data lines) to a single direction, and alternating these channels between successive rows or columns of CLBs (the technique favoured by [61]). However, this may cause long pulse transit times as data are routed around blocks to access tracks in the other direction – much as traffic in a city built around one-way streets.

The technique implemented here is to alternate the direction of data tracks after every line. It is then possible to turn signals around at every switch box, provided that enough tracks are free to do so.

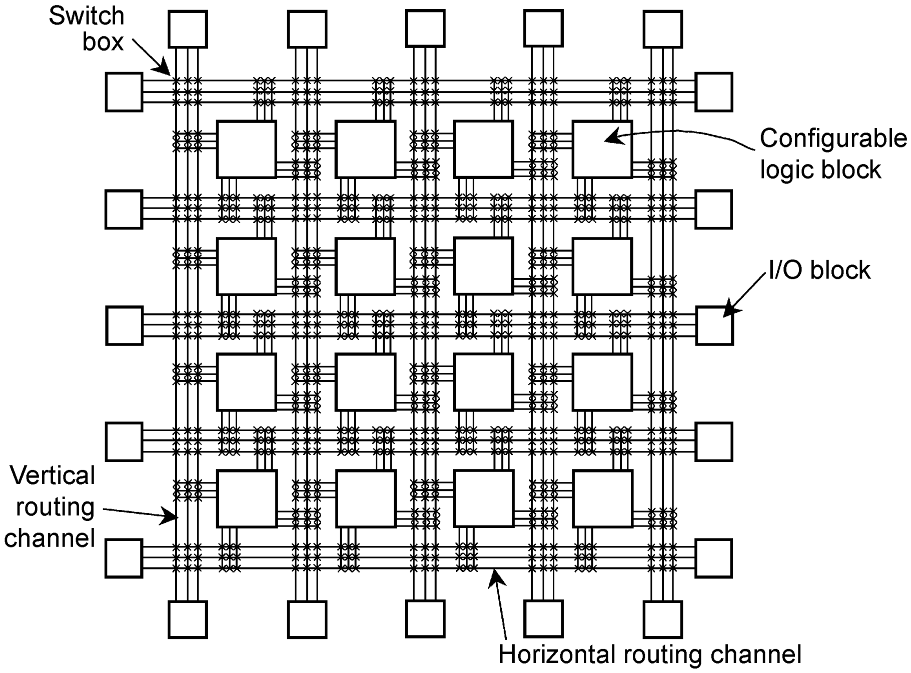


Figure 5.2: Schematic diagram of a general architecture for an FPGA

The critical advantage of reprogrammable circuits based on the interconnection strategy of FPGAs above hard-wired circuits is their heightened tolerance to low-yield gates and random malfunctions. Generally, the more routing paths [62] and possibilities there are, the more fault-tolerant a circuit will be.

Lastly, FPGAs and redundant logic cells also form the basis of techniques proposed to introduce self-healing properties into integrated circuits [63], or allow circuits to evolve their own functions [64]. One day, SPGAs might overcome the obstacles posed by low-yield superconducting logic circuits through self-healing at the expense of circuit space.

5.3 INTERCONNECTION SWITCHES

The interconnection lines in RSFQ circuits are constructed from JTLs, and although the JTL is a bidirectional device, it is always used to transmit pulses in one direction only.

For programmable connections, RSFQ switch blocks are needed. The Crossbar and Inline switches discussed here implement functionality similar to that of semiconductor switches, except for the unidirectional nature of the SFQ data tracks. The T- and Y-switches are RSFQ-specific, and were designed to handle the start or termination points of unidirectional SFQ tracks.

Only the circuit diagrams for each switch are depicted and discussed in this section. The simulation results for all switches are only shown in section 5.7, where these switches form part of the simulation model for the entire PFD.

5.3.1 Crossbar switch

The unidirectional nature of JTLs makes it possible to construct a Crossbar switch that can connect one data line to another so that data can flow from the one line to the other, but not in the opposite direction. When Inline switch elements are integrated into the Crossbar switch, it is even possible to route data so that signals entering on one line will only leave on the other, and vice versa, thereby increasing the routing possibilities of a PGA circuit. However, this requires 4 DCRLs and as many pairs of programming lines – an excessive amount when the space it occupies is taken into account.

Figure 5.3 shows a simplified schematic diagram of such a Crossbar switch, as well as some routing options. Large circles represent switch elements (DCRLs). White switches are unset; black switches are set. Data (SFQ pulse) flow is represented by thick grey lines.

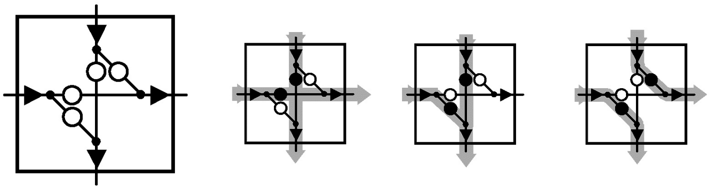
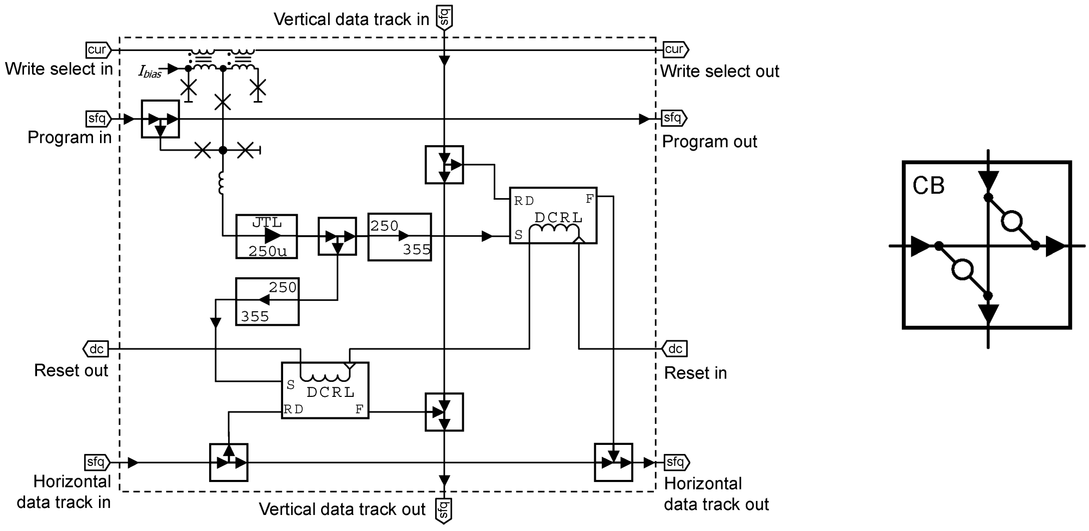


Figure 5.3: Simplified schematic diagram of 4-element Crossbar switch and some data flow possibilities

A more compact implementation with the available circuits requires two DCRLs and only one pair of programming lines. The DCRLs are set simultaneously, so that only two switch states are possible: one in which the horizontal and vertical data tracks are isolated, and one in which a pulse entering on any track leaves on both. The schematic circuit diagram is shown in Figure 5.4. Simulation results for Crossbar switches can be seen in Figure 5.16.



(a) (b)

Figure 5.4: (a) Circuit schematic and (b) data flow schematic symbol of Crossbar switch

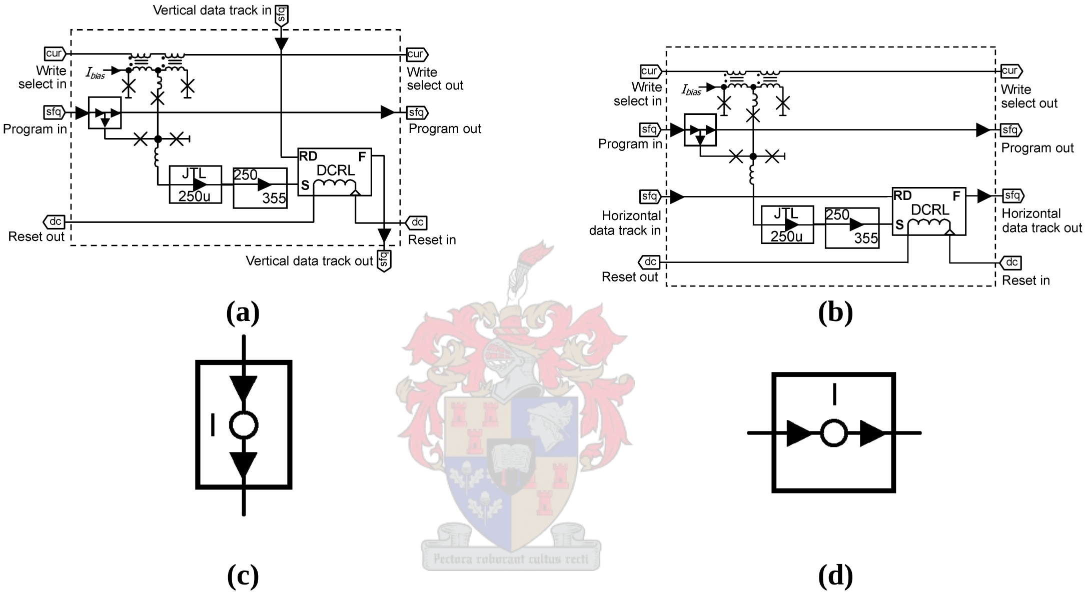
Another implementation of an SFQ Crossbar switch was published by [65]. Their switch is designed for network switching, and only connects one way (horizontal to vertical, or vertical to horizontal – not both). It is also fully SFQ, and lacks the global reset capability desired for SPGA circuits.

5.3.2 Inline switch

Since data tracks occupy a lot of die space, they need to be kept to a minimum in superconducting PGAs. One way to reduce the number of tracks without compromising routing possibilities is to subdivide each track into isolated segments. In this way, different segments of one track can be used to route different signals.

The individual stretches are connected together by Inline switches, which can be closed to allow signal transfer. Figure 5.5 shows schematic circuit diagrams for such switches. Simulation results for these switches are not shown, as they are implicit in those of the Crossbar switch.

Inline switches can also be placed next to Crossbar switches such as the one in Figure 5.4 to regain some of the functionality of the 4-element Crossbar switch in Figure 5.3.



(a) (b)

(c) (d)

Figure 5.5: Inline switch circuit schematics for (a) vertical and (b) horizontal data tracks, and Inline switch schematic symbols for (c) vertical and (d) horizontal data flow

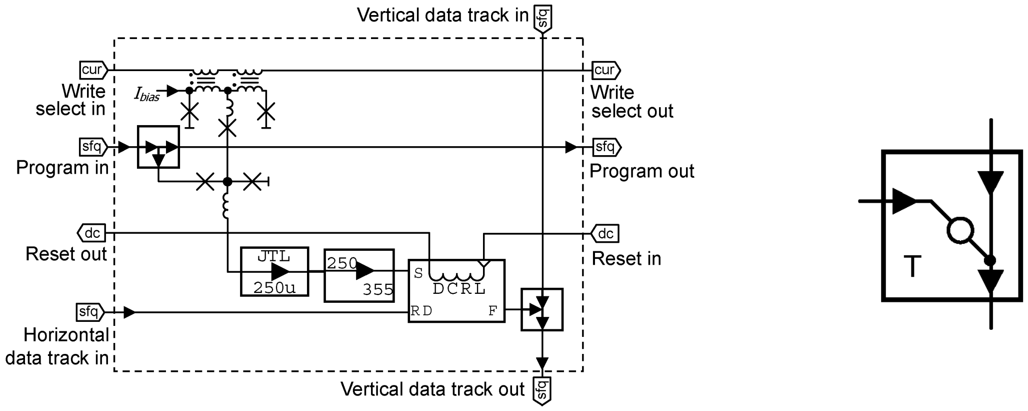
5.3.3 Special Crossbar switches – the T-switch and Y-switch

At the boundaries of a PGA, Crossbar switches can be replaced by more compact switches. These switches are basically Crossbar switches with some data paths removed.

Figure 5.6(a) shows the schematic circuit diagram of a T-switch; in this case one which is used at the terminal point of a left-right horizontal data track.

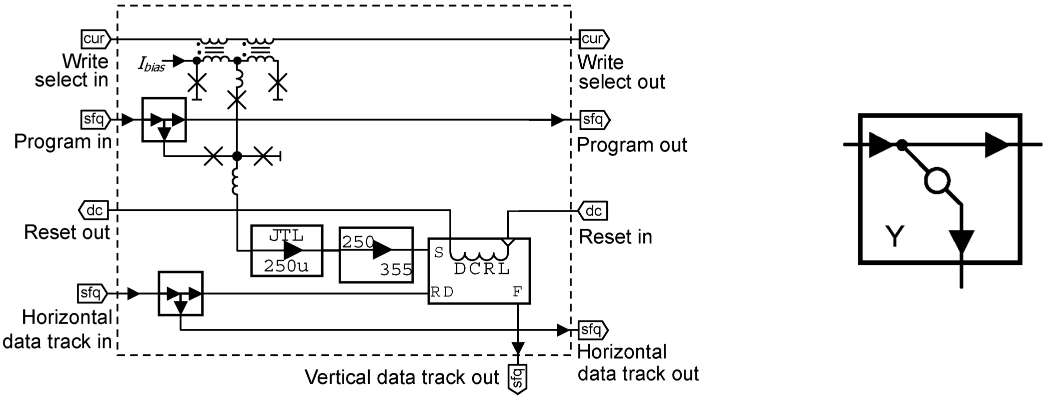
Figure 5.7(a) shows the schematic circuit diagram of a Y-switch. This one is used in the topmost horizontal data track, where the up-down vertical tracks start.

Simulation results for the T-switch are shown in Figure 5.17, and those for the Y-switch in Figure 5.18.



(a) (b)

Figure 5.6: (a) Circuit schematic and (b) data flow schematic symbol of T-switch



(a) (b)

Figure 5.7: (a) Circuit schematic and (b) data flow schematic symbol of Y-switch

5.4 ARCHITECTURE, ROUTING CHANNELS AND SWITCH BLOCKS

The PFD is, conceptually, a very simple circuit. It contains a number of Toggle flip-flops, interconnection lines and programmable switches.

Each T flip-flop divides its input by two, so that division of up to 2^n, where n is the number of T flip-flops, can be achieved by programming the switches to route an incoming signal through the desired number of the T flip-flops.

The flip-flops are flanked on all sides by interconnection lines, much as the standard structure for FPGAs [60].

The Toggle flip-flop used is the T1 flip-flop [2], which differs from other RSFQ T flip-flop implementations in that it also has a destructive read input, and only a complemented asynchronous output (so that from the start-up state it gives an output pulse when the first input pulse is applied, and then only again after the third pulse, the fifth pulse, and so forth). The T1 flip-flop also has a high yield (provided that it is driven with a 250-to-355 *m*A JTL [3]), and only needs a positive bias voltage, as opposed to the bipolar bias requirement of the standard T flip-flop [2].

In order to keep the physical circuit within the boundaries of realizability (with regard to layout), as well as limit the number of off-chip connections, the PFD is limited to four T flip-flops.

The initial design allowed for six horizontal and six vertical interconnection lines, and 144 programmable switches. A schematic representation of the initial PFD design is shown in Figure 5.8.

Switch symbols are simplified, so that Y-, T- and Crossbar switches are indicated by circles, and Inline switches by short perpendicularly crossing lines. The column programming lines are numbered to show that lines 2 and 12 are used to access the switches on the input and output lines. The four switches within the hatched areas also do not lie in line with column programming lines, and are programmed by lines 4, 5, 9 and 10.

This design still uses the initial assumption that the two DCRLs in a Crossbar switch would be accessed and programmed individually.

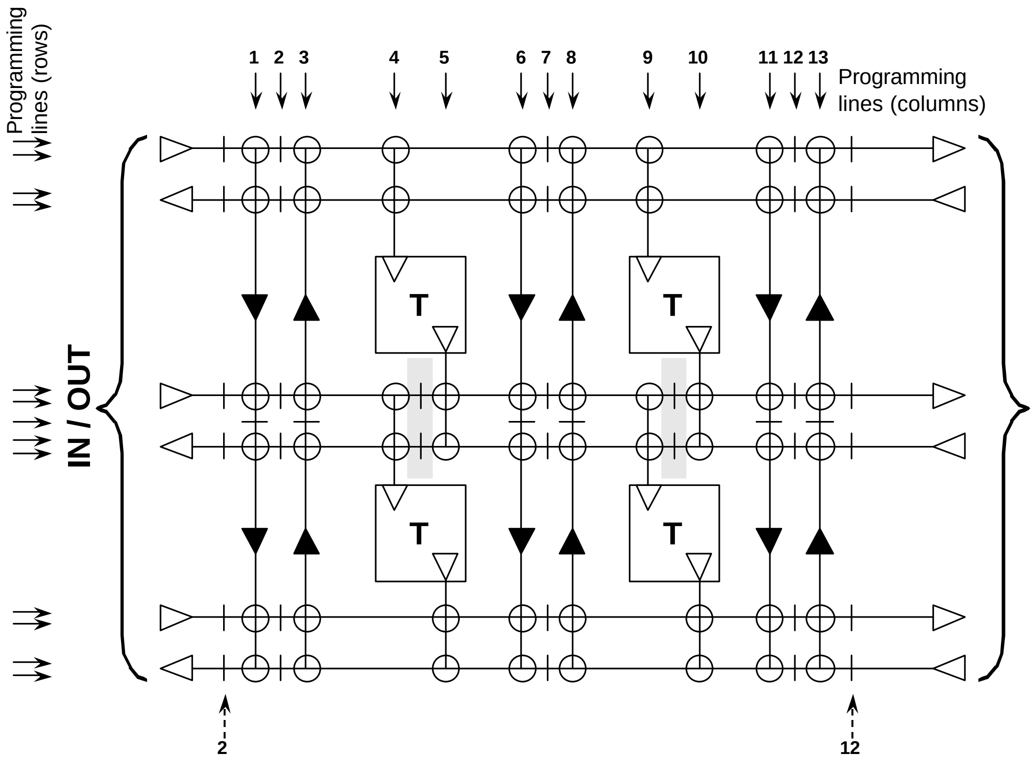


Figure 5.8: Schematic representation of PFD

The initial PFD shown in Figure 5.8 has sufficient redundancy in terms of routing possibilities and for dead switch bypassing. Unfortunately, first experimental layouts showed that it might just not fit onto the 0.25 cm^2, 3-micrometre resolution die from Hypres. The chief culprits are the programming lines, each of which needs a large AND-gate for write selection. A forced redesign thus necessitates a two-pronged optimization scheme: reduce the number of tracks, and hook both switches on a crossbar connection to the same programming lines. The former cuts back the number of switches, thereby reducing programming lines. The latter allows line reuse, and also reduces the number of programming lines needed.

A redesigned, slimmer PFD is shown in Figure 5.9. It has fewer interconnection lines (reducing the number of switches), and also uses the Crossbar switches shown in Figure 5.4(a) of which the DCRLs share the same programming address.

This PFD design has only 66 switches, and the programming grid is composed of 9 bipolar current column lines (driven by HUFFLEs) and 8 SFQ pulse row tracks. Since each column line can only decode 8 unique addresses (one for each SFQ row), lines 3 and 7 cannot address all the switches in their paths. The solution is to use the spare capacity of columns 1 and 2, and 8 and 9 to address the excess switches. This addressing system is shown in Figure 5.10.

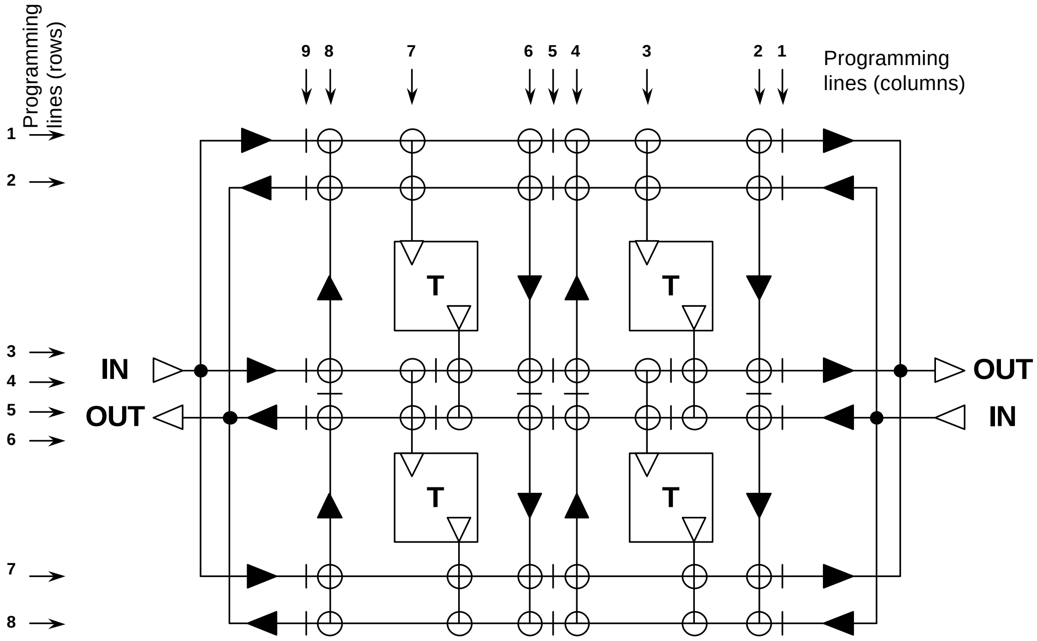


Figure 5.9: Schematic diagram of PFD with reduced number of programming lines

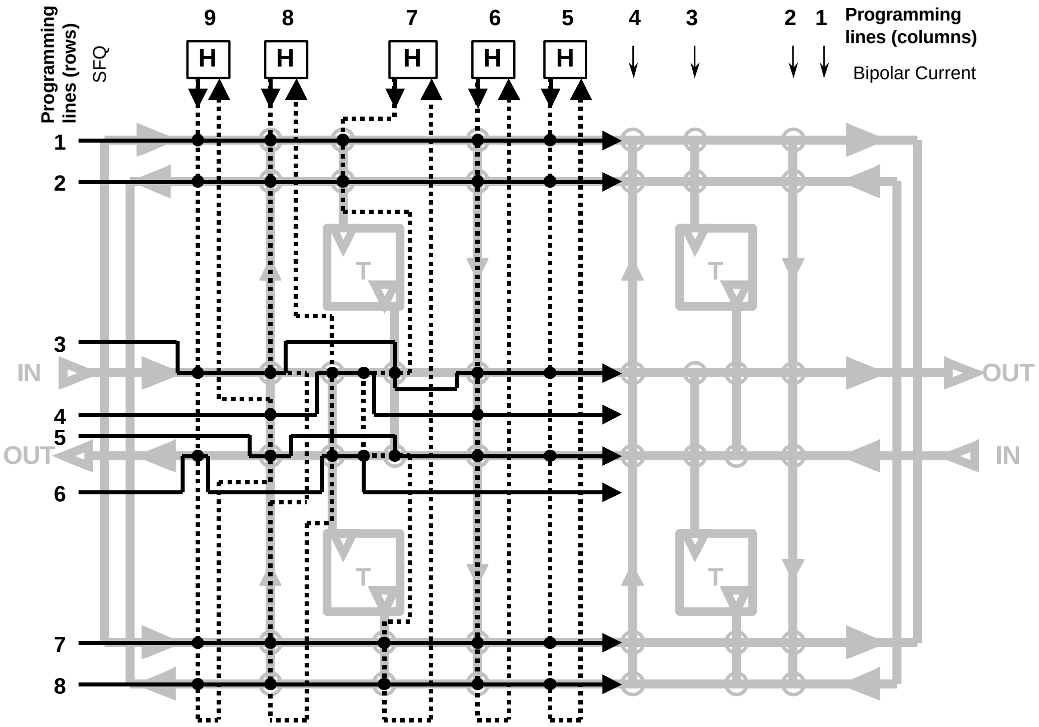


Figure 5.10: Schematic diagram of PFD showing programming line access to switches (lines 4 down to 1 mirror lines 6 to 9, and are omitted for clarity)

The smaller PFD in Figure 5.9 meets all our design criteria and interfacing specifications, and the implementation thereof is discussed in the rest of this chapter. All further instances of the term PFD in this chapter refer to this circuit.

5.5 SWITCH PROGRAMMING ARCHITECTURE

Individual switch access requires either a lot of programming lines, or fewer lines with a lot of decoding logic. The die size and probe construction limits the number of off-chip connections, so that the only available option entails efficient decoding structures.

One way would be to use a 7-bit serial input word with three bits for the horizontal lines (000 to 111), four bits for the vertical lines (0000 to 1000), and then decoding it on-chip for individual switch access. However, this is a mundane action that requires large decoding circuits in the already limited space on the superconducting die, and raises the error probability for switch programming.

Since programming is a low frequency activity that can be managed by semiconductor circuits, it is more practical to move as many as possible of the programming tasks to off-chip logic. The technique employed here is to use a 17-bit input word in which each line and column has its own bit. Such a word requires no on-chip decoding, thus saving valuable chip space.

The full switch programming architecture of the PFD requires 9 column drivers and 8 row drivers, as well as shift registers and clock and write signal distribution circuitry. Instead of showing the large schematic diagram of this entire circuit, a diagram of the programming architecture for a 4×4 switch matrix is shown in Figure 5.11 in order to make the logic flow easier to comprehend. The signal entry points, lag cells in the shift registers, clock and write signal distribution paths and column driver reset circuitry are all exactly the same as for the full switch programming circuit. It can easily be seen in Figure 5.11 how the gates and latches of the column and row drivers are repeated, and the full programming circuit can be deduced from this schematic diagram.

The switch programming sequences for both the full circuit and the smaller 16-switch programmer are shown in Figure 5.12. The operation of the programming circuit can now be explained in terms of the circuit diagram in Figure 5.11, the programming signals in Figure 5.12, and the row and column numbers defined in Figure 5.9 and Figure 5.10.

The programming data word consists of as many bits as there are rows and columns together, or 17 for the full programmer and 8 for the 16-switch programmer.

Let us now only consider the full programmer. The row bits are entered into *Data* first, starting with the highest row bit (denoted R8 in Figure 5.12(a)). A set bit selects the corresponding row. After the lowest row bit (R1), the column bits are entered from highest to lowest (C9 to C1).

Write must be pulsed high in the same clock cycle as the seventeenth or last column bit. The first 16 *Clock* pulses merely shift the data word into a serial shift register. The seventeenth *Clock* shifts the last *Data* bit in, and also moves the column bits into AND-gates that all have one input set by *Write*. After a short internal delay, this *Clock* reads the column driver AND-gates into the *Set* inputs of HUFFLEs. Only one column bit may be set, so that one column of switches is active after the seventeenth *Clock* pulse. The seventeenth *Clock* pulse also releases an internally delayed *Write* pulse to propagate to the inputs of the row driver AND-gates.

The eighteenth *Clock* pulse ANDs the row bits with the delayed *Write* pulse, and releases the row SFQ pulses into the switch matrix. The nineteenth *Clock* pulse resets any set HUFFLEs.

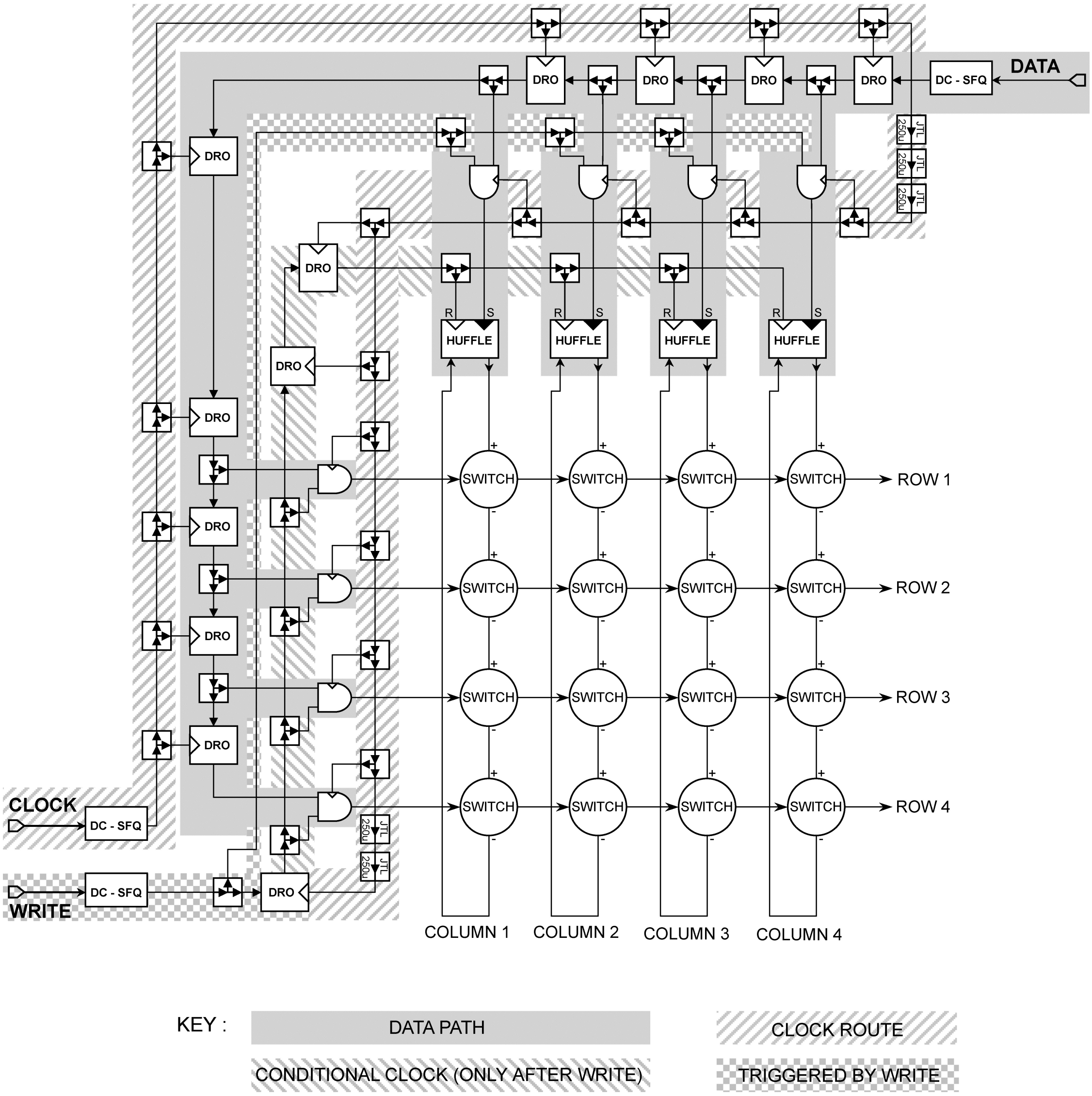
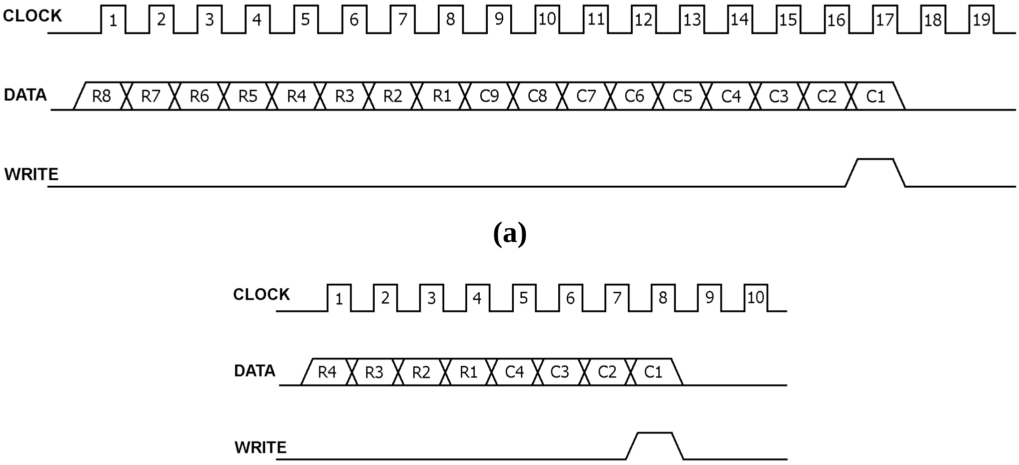


Figure 5.11 Component level representation of 4´4 switch matrix programmer



(a)

(b)

Figure 5.12: Switch programming sequence of PFD with grid of (a) 9 columns and 8 rows and (b) 4 columns and 4 rows

Since the switch matrix is programmed one column at a time, any or all of the row bits but only one columns bit can be set in a programming word. The entire PFD can therefore be programmed with 9 17-bit words in 171 clock cycles.

The programming waveforms shown in Figure 5.12 do not have setup and hold time indications. The programming circuit was designed (through the use of internal delays where necessary) to accept a *Clock* pulses that are exactly coincident in time with *Data* or *Write* pulses. Due to the nature of SFQ pulses and the speed of conversion of a DC-to-SFQ converter, input voltage pulses need only stay high for a few picoseconds. The only timing restriction is the minimum clock cycle, which is limited by the time it takes a pulse to shift through the clock distribution circuitry and the time that the HUFFLEs need to undergo a state change.

Simulation results for the 16-switch programmer (Figure 5.13) show that it can be clocked at 2.5 GHz. The full PFD will therefore function correctly with a programming clock of up to 1 GHz – much faster than the low MHz data rates of commercial microcontrollers such as the Atmel AT89C2051 intended for use as the hot logic controller.

5.6 SIMULATION RESULTS FOR SWITCH PROGRAMMING

Simulation results for the 16-switch programming circuit are shown in Figure 5.13. The input signals are SFQ pulses (the hot logic interface was not simulated). Note that the *Clock* pulses in the simulation are temporally coincident with the *Data* and *Write* pulses. This is not necessary, but shows that no setup time is required between *Data* and *Clock* pulses.

Each switch block consists of a pulse splitter and a Current-Set switch. The pulse splitter allows pulse continuation along the row of switches, and the output of the Current-Set switch is used to set a DCRL memory element. The *Spice* model consists of 2530 elements, 486 of which are Josephson junctions.

The *Data* word is 10110100, so that switches 4, 2 and 1 of column 2 will be set. The *Clock*, *Write* and delayed *Write* signals are also shown, where $V_{writedelay}$ is measured at the *F* output of the DRO closest to the *Write* input in Figure 5.11. The simulation results show how the eight bits of the *Data* word are entered during the first eight *Clock* cycles, and how the *Write* pulse is also entered during the eighth *Clock* cycle. The eighth *Clock* pulse also releases the delayed *Write* signal, which is fed to the row select AND-gates.

All four *Column Select* currents are shown ($V_{colselect1}$ to $V_{colselect4}$), and it is clear that only the second column is activated after the eighth *Clock*. It is reset after *Clock* 10.

The row SFQ pulses are shown as $V_{rowprgrm1}$ to $V_{rowprgrm4}$. The selected row pulses, namely 1, 2 and 4, are released into the switch matrix after the ninth *Clock*. The three switches in rows 1, 2 and 4 of column 2 are then programmed, or set.

Outputs of the Current-Set switches in the switch blocks of rows 1, 2 and 3 are also shown. As an example, $V_{switch11}$ is the output of the Current-Set switch in row 1 and column 1, whereas $V_{switch34}$ is the output of the Current-Set switch in row 3 and column 4. As expected, the Current-Set switches of rows 1 and 2 in column 2 pass their input pulses. The Current-Set switch outputs for row 4 are not shown, as they are identical to those of rows 1 and 2. Since the simulations show that the correct switches are set, it is clear that the programming function is realized correctly.

Observe that the bipolar current $I_{colselect2}$ is reset after the tenth *Clock*.

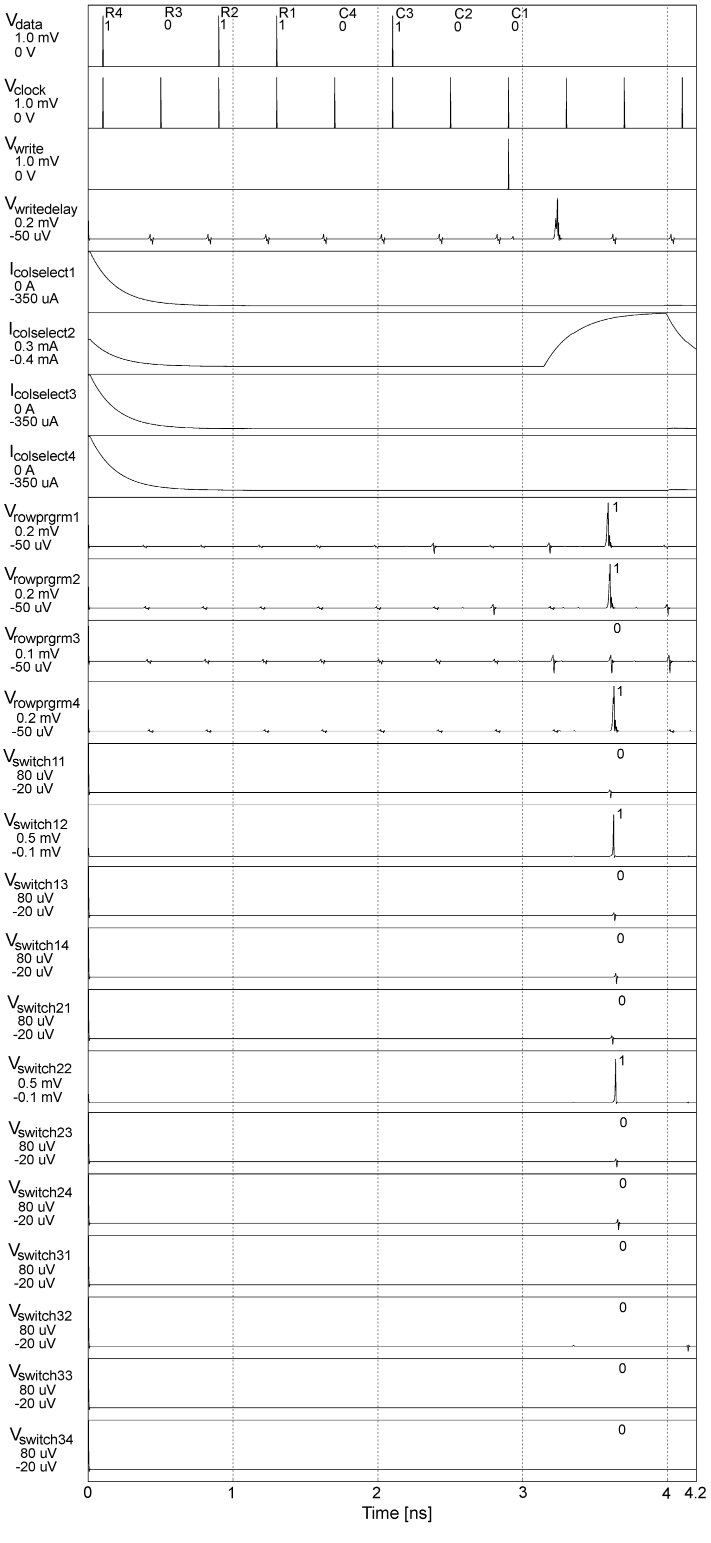
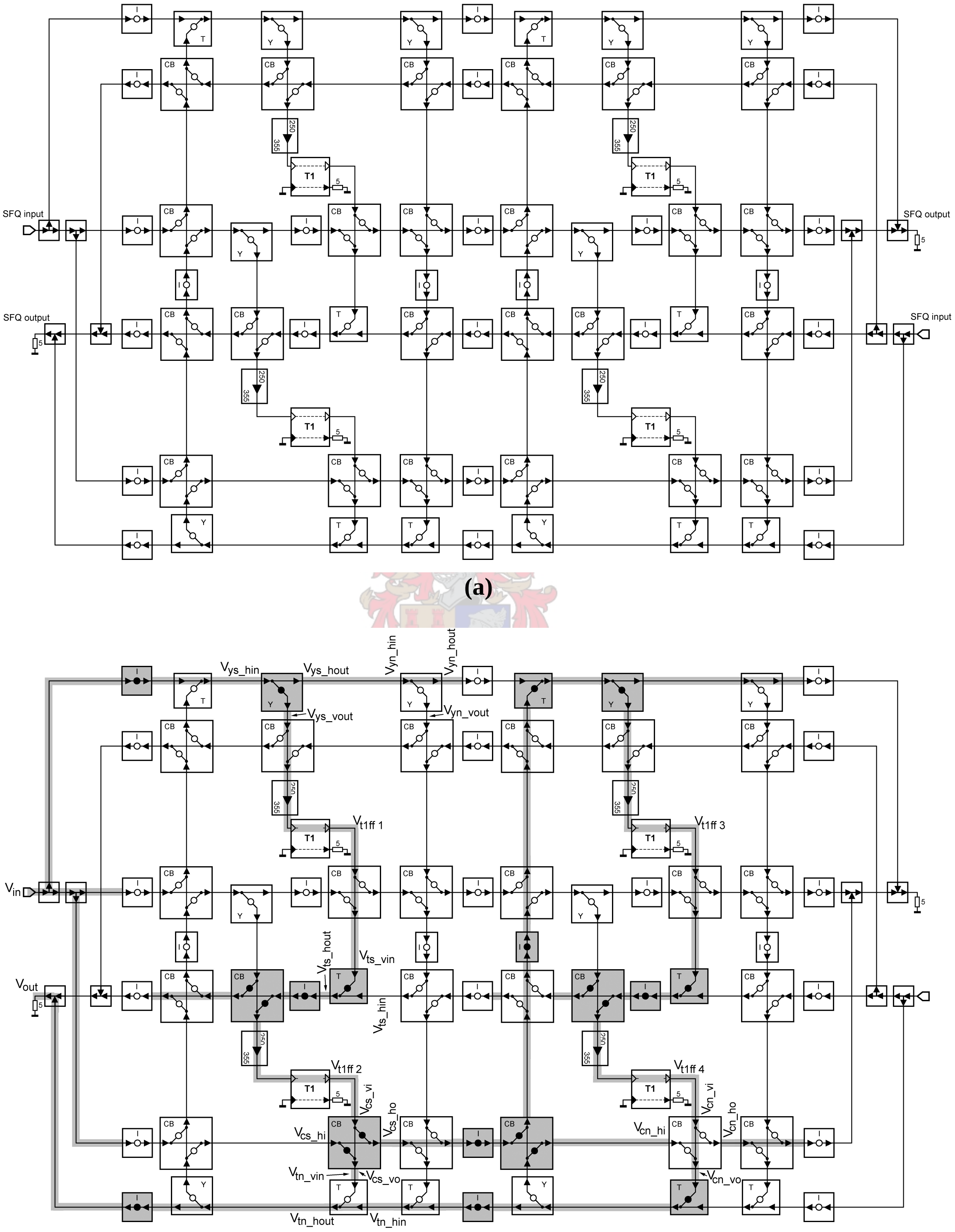


Figure 5.13: Simulated programming of 4´4 switch matrix of a PFD

5.7 SIMULATION RESULTS FOR A PFD CONFIGURED FOR DIVIDE-BY-16 OPERATION

For simulations on the PFD, a *Spice* model was constructed that contains all the Crossbar, Inline, T- and Y-switches as well as T1 flip-flops, pulse splitters and pulse mergers necessary to implement the full PFD. The circuit schematic is shown in Figure 5.14(a).

Figure 5.14(b) shows the set switches and data path when the PFD is configured for divide-by-16 operation. All paths along which data will flow are shown to give an idea of how many track segments are occupied. The simulation model consists of 24147 elements, of which 4623 are Josephson junctions.



(a)

(b)

Figure 5.14: (a) Schematic diagram of simulation model for PFD and (b) schematic diagram showing set switches, signal paths and probe positions for programmed divide-by-16 operation of the PFD

Table 5.1 shows the full set of words needed to program the PFD for the divide-by-16 operation detailed in Figure 5.14(b). For each column, the row bits needed to set the correct switches are also given. Data words are subdivided into groups of four bits for readability.

Table 5.1: Full programming sequence to configure PFD in divide-by-16 mode

Column	Row bits set	Data word (R8-R1,C9-C1)
9	1,8	1000 0001 1000 0000 0
8	6	0010 0000 0100 0000 0
7	1,5,6,7	0111 0001 0010 0000 0
6	–	0000 0000 0001 0000 0
5	7,8	1100 0000 0000 1000 0
4	1,4,7	0100 1001 0000 0100 0
3	1,5,6,8	1011 0001 0000 0010 0
2	6	0010 0000 0000 0001 0
1	–	0000 0000 0000 0000 1

The simulation results of the PFD with switches set as shown in Figure 5.14 are shown in Figure 5.15. Since the circuit model already contains more than 24000 elements, the programming circuitry is not included. Switch programming is thus effected by using piece-wise linear current and voltage sources to set all the required switches during the first 150 ps of the simulation. The programming current and voltage pulse for one such a switch are shown as the top two traces in Figure 5.15 to Figure 5.18.

The simulation results in Figure 5.15 show the progression from input to output, as well as the output voltages of each T flip-flop. The 850 ps delay between $V_{t1ff\,2}$ and $V_{t1ff\,3}$ is a result of the long signal path between the bottom left and top right corners of the PFD.

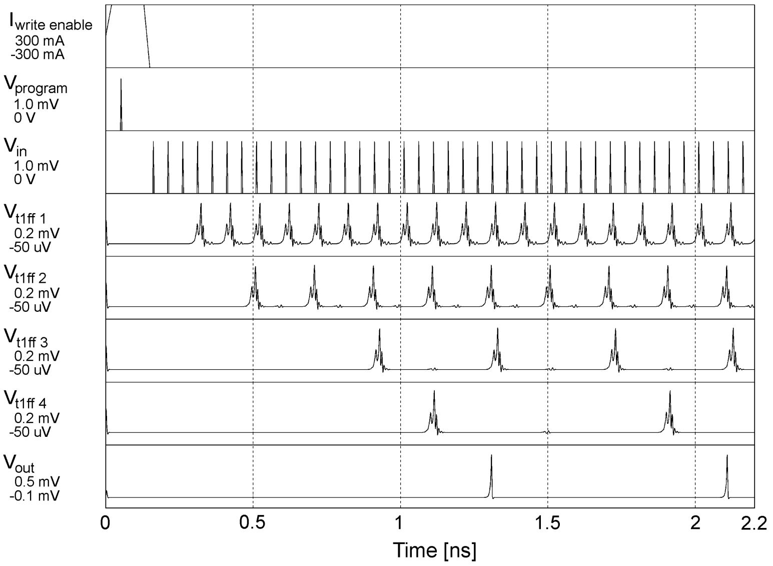


Figure 5.15: Simulated transient response of 4-cell PFD configured for divide-by-16 operation

The total signal delay from input to output is 1.2 ns. When the distance it travels is taken into account, this can rise to 1.3 ns. Extra JTLs for pulse routing over larger distances may push this to 1.4 ns or 1.5 ns.

The programmed divide-by-16 functionality of the PFD is easy to verify. Firstly, there is the simulated programming currents and voltage pulses, shown here by $I_{write\ enable}$ and $V_{program}$. The next trace in Figure 5.15 is the high frequency input at *In*, which has a repetition frequency of 10 GHz. The input signal is then routed to the first T flip-flop, of which the output at 5 GHz is shown in V_{t1ff1}. After the second T flip-flop (V_{t1ff2}), the pulse repetition rate is only 2.5 GHz. The signal is then routed to the third and fourth T flip-flops, of which the outputs are shown as V_{t1ff3} and V_{t1ff4}. It is clear that the output of the fourth T flip-flop has a repetition rate that is 16 times lower than that of the input at V_{in}. This signal is routed to V_{out}, where it appears as a pulse train with a frequency of 625 MHz.

The correct operation of the PFD shows that all the switches and routing structures perform as designed. For the sake of completeness, the simulated response of Crossbar, T- and Y-switches in both the set and unset states are shown here. Voltage and current vectors in Figure 5.16 to Figure 5.18 correspond to the labels in Figure 5.14(b).

The simulated response of the Crossbar switch (Figure 5.4) is shown in Figure 5.16. In (a) the switch is set (observe the programming pulse at 50 ps), so that data entering from any input must leave at both outputs. The simulation shows that data entering on the vertical data track leave on both the vertical and horizontal data tracks.

In Figure 5.16(b) the switch is not set, and the data tracks must be isolated from each other. The simulation results show this to be the case, as pulses on the horizontal data track have no influence on the vertical data track, and vice versa. The Crossbar switch therefore simulates correctly.

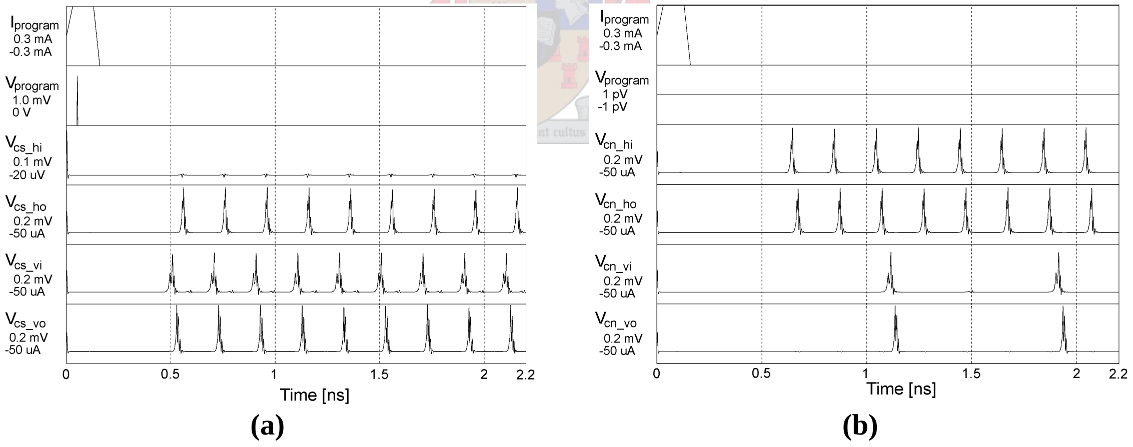
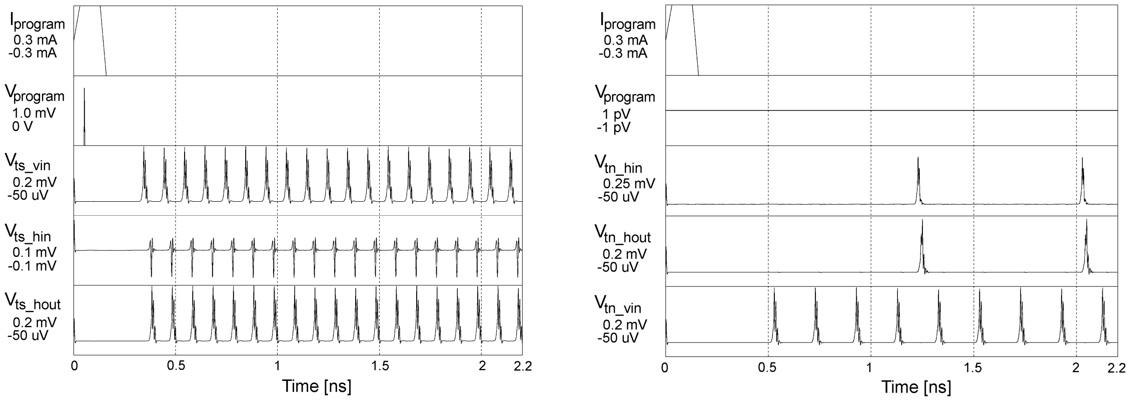


Figure 5.16: Simulated transient response of Crossbar switch when (a) set and (b) not set

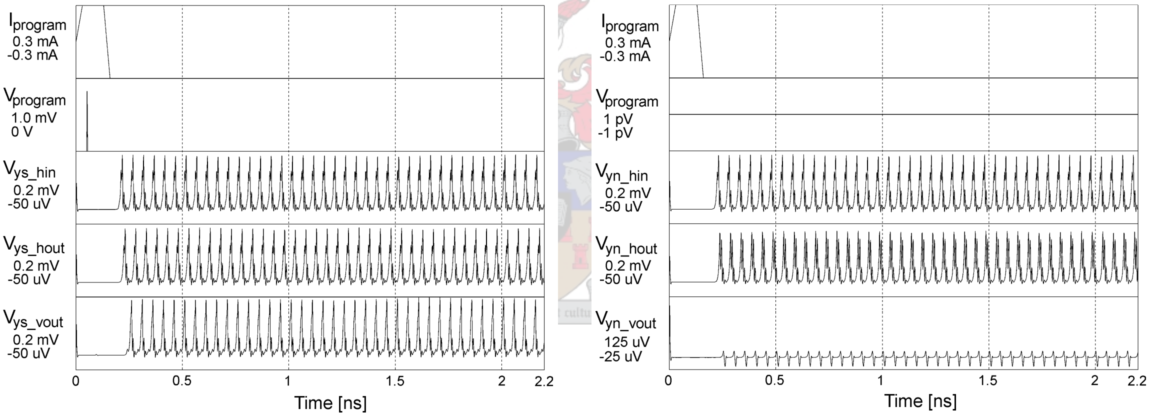
Figure 5.17 shows the simulation results for two T-switches (Figure 5.6) in the PFD. In (a) the switch is set, and data entering from the vertical direction (V_{ts_vin}) leave in the horizontal direction (V_{ts_hout}), which shows that the vertical track is correctly connected to the horizontal one. In (b), another T-switch is in the unset state. Here data entering from the horizontal direction (V_{tn_hin}) pass unhindered to the horizontal output (V_{tn_hout}), whereas data entering from the vertical input never leave the switch. This is also correct, and verifies track isolation when the switch is not set.



(a) **(b)**

Figure 5.17: Simulated transient response of T-switch when (a) set and (b) not set

The simulation results for two Y-switches (Figure 5.7) in the PFD are shown in Figure 5.18. In (a) the Y-switch is set and the input track connected to both outputs, so that data entering from the horizontal direction (V_{ys_hin}) leave in both directions (V_{ys_hout} and V_{ys_vout}). The switch in (b) is in the unset state, so that the vertical data track is isolated from the horizontal one. Data entering from the horizontal direction therefore only leave in the same direction. The Y-switch therefore functions correctly in simulations.



(a) **(b)**

Figure 5.18: Simulated transient response of Y-switch when (a) set and (b) not set

One of the characteristics of programmable circuits based on the interconnection strategy employed in FPGAs is that there are almost always more than way to route a signal for the same function.

As an example, consider the diagrams in Figure 5.19, which show four alternative ways to configure the PFD so that it still performs a divide-by-16 operation between the same input and output ports as with the setup in Figure 5.14(b). Blackened circles indicate set switches.

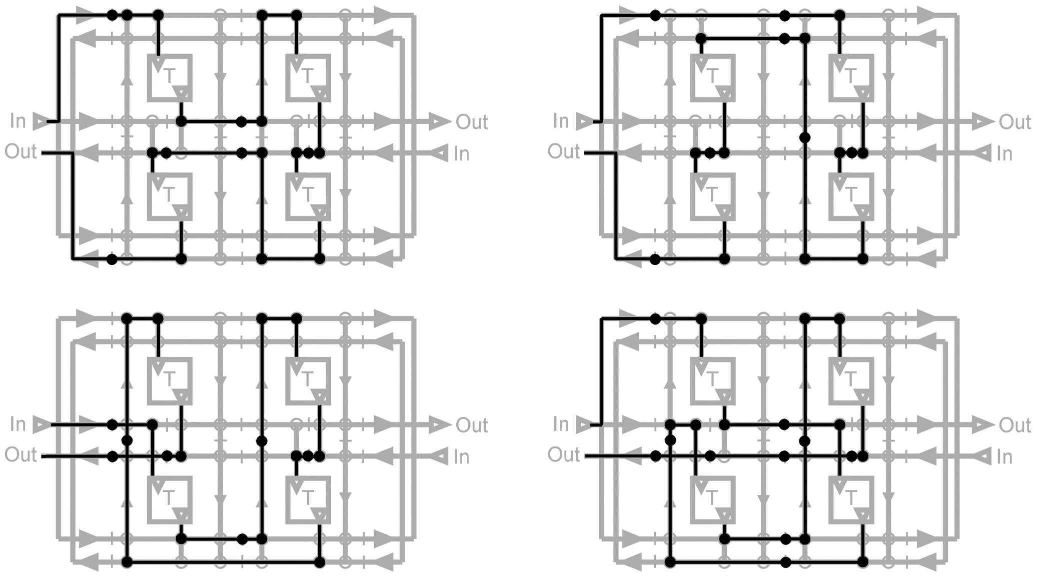


Figure 5.19: Four examples of switch settings and data flow for divide-by-16 operation using the same input and output

5.8 CONCLUSIONS

A complex reprogrammable circuit, the programmable frequency divider, was developed from the basic structure of an FPGA, and implemented using existing and novel RSFQ gates and latches. Simulations show that this circuit indeed performs the operations that it is programmed to do. Although the PFD treated here is the largest circuit of its kind that can be manufactured with the 3-micrometre Nb process from Hypres, it paves the way for the development of a full SPGA as soon as the manufacturing capability becomes available.

The shift register input of the programmable frequency divider also makes it easy to interface with command words originating off-chip, and therefore allows easy programming through slow semiconductor microcontrollers. However, the rf inputs (routed to the array of Toggle flip-flops) run completely asynchronously and can subsequently be clocked from dc to well past 10 GHz. The output should merely be periodic at a frequency that is lower than that of the input by the pre-programmed division factor. The maximum operating frequency of the PGA can therefore be established with ease.

This circuit also provides a platform for easy testing of RSFQ circuits and elements, and the T flip-flop can be exchanged for any other logic block when etching technology allows it.

In short, it has been proved through the designs in this chapter that RSFQ circuits can be designed to have all the flexibility and programmability of conventional semiconductor FPGAs, and that the construction of an SPGA is therefore indeed possible.

Chapter Six - Compact PFD

6.1 INTRODUCTION

AS a first test for the design process, a circuit that is simpler and smaller than the full SPGA-oriented PFD is designed. This circuit allows testing of all the new components, as well as probe characterization.

The primary requirement of this circuit is, however, that it should be programmable by slow semiconducting electronics, and produce outputs that can be measured with basic rf laboratory equipment.

6.2 FUNCTIONAL DESIGN

The compact PFD consists of two Toggle flip-flops connected in cascade. As with the PFD in Chapter 5, the Toggle flip-flop used is the T1 flip-flop [2].

The schematic circuit diagram is shown in Figure 6.1. Operation of the compact PFD is easy to comprehend. An rf input signal at *Rf in* is converted to SFQ pulses. This pulse train is then split, with one arm feeding directly into a programmable switch. The other arm feeds a cascade of T1 flip-flops, of which the outputs are also connected to switches. The desired division factor is obtained by setting one of the three switches, thereby allowing it to pass its input pulse train to the output stage. Here the pulse paths are merged before the output pulses are converted to voltage state signals.

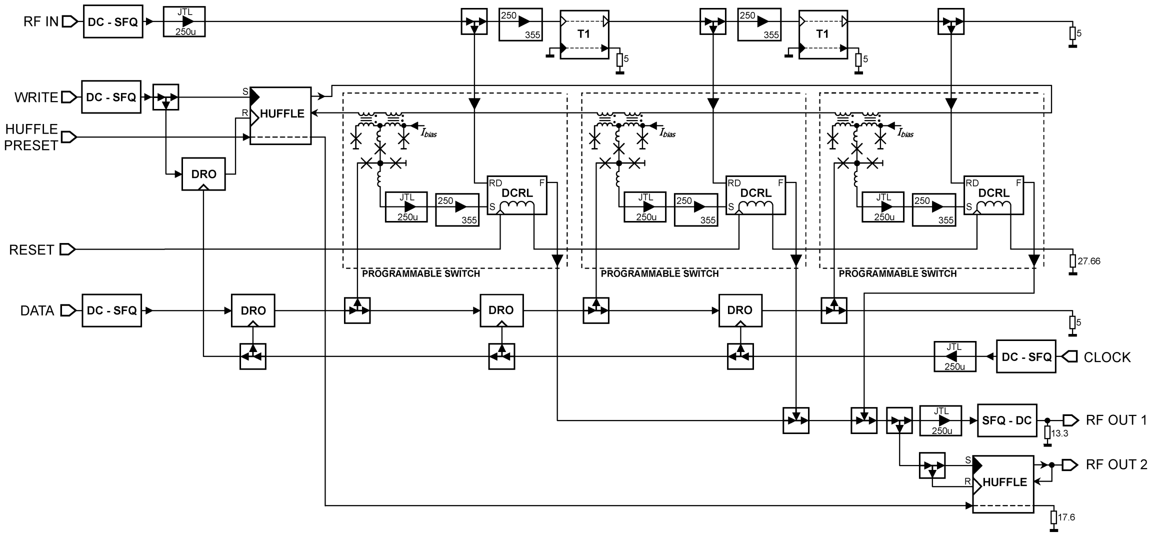


Figure 6.1: Circuit schematic of compact programmable frequency divider

Although there are only two Toggle flip-flops after the RF input, the compact PFD divides this input by 2, 4 or 8. This is because each output converter (thus the SFQ-to-DC converter and the HUFFLE) also operates as a T flip-flop that divides the SFQ output pulse train by a further factor of 2.

The HUFFLE used in this dissertation is a novel circuit that has not been tested physically before, and has a theoretical yield of less than 100 %. It was thus decided to design another compact PFD in which the HUFFLE used for write selection is replaced by a bipolar current line driven by an off-chip source. This circuit is shown in Figure 6.2.

The use of an externally driven write line allows a more reliable verification of the programmable switch circuit.

Both compact PFDs were added to an experimental chip layout (see Figure A.16 and Figure A.17).

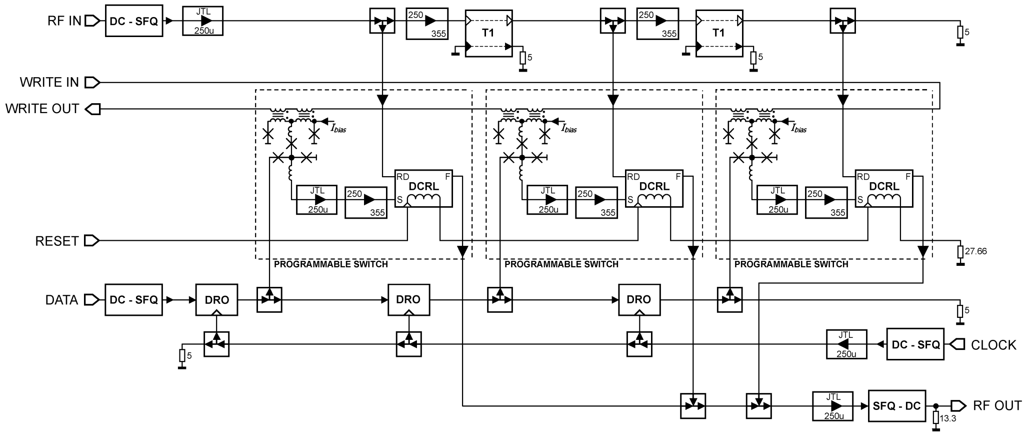


Figure 6.2: Circuit schematic of compact PFD without HUFFLEs

6.3 PROGRAMMING SPECIFICATIONS AND TIMING

The compact PFD does not need a high frequency clock, and can be programmed with low frequency electronics of, for example, the TTL or CMOS semiconductor logic families.

The programming action requires only a 3-bit data word, one write bit (for the compact PFD with HUFFLEs), and a clock signal. Timing diagrams are not shown here, since two clear examples of the programming sequence are shown in the top three traces of Figure 6.4.

It must be noted that the write bit must arrive during the same clock cycle as the third (least significant) bit of the data word, and that correct programming requires that only one bit in the data word is set. There are therefore only four programming possibilities, as listed in Table 6.1.

Table 6.1: Full set of data words and programmable operations for compact PFD

Data word (MSB to LSB)	PFD operation
000	No output
001	Divide by 2
010	Divide by 4
100	Divide by 8

For programming, data words are entered starting with the MSB.

Reset is accomplished by applying a dc voltage (or a voltage pulse) at *Reset*.

6.4 SYSTEM SIMULATIONS

All simulations in the rest of this chapter are on the compact PFD with HUFFLEs, as the version without HUFFLEs is a simplification that functions exactly the same, with the exception of the write signal.

A complete circuit schematic for system simulations is shown in Figure 6.3. This circuit schematic contains more JTLs than that shown in Figure 6.1, since it was found during layout that more JTLs were needed to route SFQ pulses over the required distances. This circuit schematic can be compared to the layout mask in Figure A.16.

The simulation model includes the resistive dividers used to reduce the 5 Volt logic levels of off-chip electronics to the required levels for conversion to SFQ pulses. Since all lines on the probe that connects to the chip are 50 Ω transmission lines, high frequency connections are matched to 50 Ω.

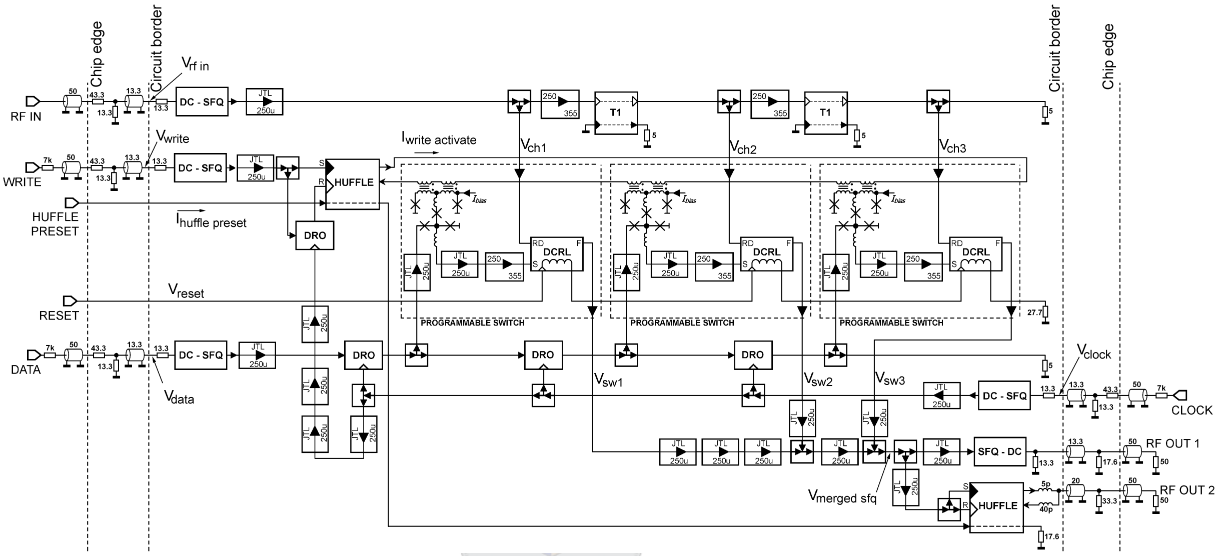


Figure 6.3: Complete circuit diagram for simulation of the compact PFD

The simulation results for the PFD in Figure 6.3 are shown in Figure 6.4. All trace names in Figure 6.4 correspond to the labels in Figure 6.3, except for $V_{rf\ out\ 1}$ and $V_{rf\ out\ 2}$ which are measured at the terminals marked *RF Out* 1 and *RF Out* 2. The simulation spans two programming and reset cycles. The simulated hot logic input signals have a very high frequency, but this is merely to limit the simulation time. In practice, these input signals can go down to dc.

The *Huffle Preset* input is shown as $I_{huffle\ preset}$. Observe the start-up preset pulse, followed by a continuous bias. The start-up transient can be seen in $I_{write\ activate}$.

In the first programming sequence (from 300 ps to 1200 ps), the *Data* input is 010. The selected operation is divide-by-4 (see Table 6.1). Three clock pulses (V_{clock}) are needed to shift the programming data in. The *Write* input pulse during the third clock cycle activates the write sequence by setting a HUFFLE, and allows the third *Clock* input pulse to set the selected DCRLs. The HUFFLE output current, $I_{write\ activate}$, is also displayed in Figure 6.4. It is clear that the bipolar current is set to its positive value after the *Write* input pulse, and returned to the unset negative value by the third and final *Clock* input pulse.

A 10.25 GHz rf input signal (a popular intermediate frequency for millimetre wavelength systems) is then applied at *RF In*, starting at around 1.5 ns. This signal is converted to an SFQ pulse train with a repetition rate of 10.25 GHz, which is shown as

V_{ch1}. The outputs of the T1 flip-flops are shown as V_{ch2} and V_{ch3}. Since the output of a T1 flip-flop has half the frequency of its input, the repetition rates of V_{ch2} and V_{ch3} are 5.125 GHz and 2.563 GHz respectively.

The three switch outputs are shown in V_{sw1}, V_{sw2} and V_{sw3}. Since the second switch is set in the first programming sequence, it lets pulses through (visible in V_{sw2} between 1.75 ns and 3.1 ns).

$V_{merged\ sfq}$ is measured after all three switch outputs are merged together, and represents the SFQ output of the PFD. Between 1.8 ns and 3.2 ns it contains a pulse train with a repetition rate of 5.125 GHz (the same as the output of the second switch).

The output SFQ pulses are converted to voltage state signals by both an SFQ-to-DC converter ($V_{rf\,out\,1}$) and a HUFFLE ($V_{rf\,out\,2}$). In $V_{rf\,out\,1}$, the desired output signal is the envelope of the high frequency carrier. Both output signals have a frequency of 2.563 GHz, and it is clear from the simulation results that the divide-by-4 operation is therefore successful.

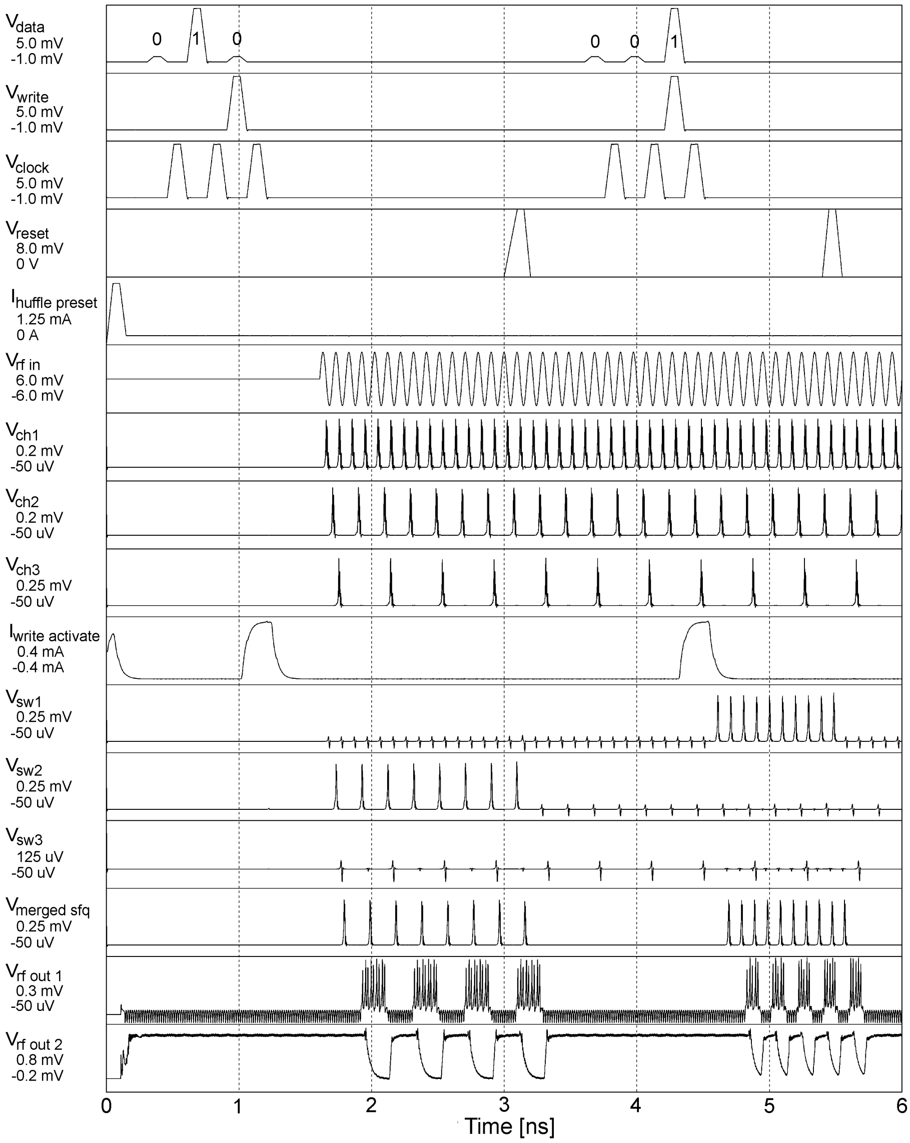


Figure 6.4: Simulated response of compact PFD to programming and reset signals and 10.25 GHz rf input

A reset signal (V_{reset}) is applied after 3 ns. The rf outputs stop soon thereafter, even though an rf input is still applied. A second programming sequence using *Data* input 001 (divide-by-2) is then commenced (between 3.6 ns and 4.5 ns), and the first switch is consequently set. The rf output signals then only have half the frequency of the rf input, or 5.125 GHz. These output signals appear between 4.8 ns and 5.8 ns, after which the PFD is reset once more.

The simulation results clearly show that the compact PFD functions correctly – even with simulated CMOS programming input signals entering through transmission lines.

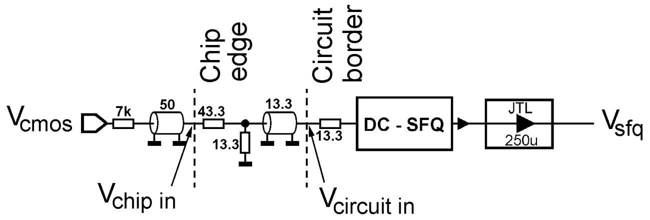


Figure 6.5: CMOS logic level to SFQ pulse interface

Figure 6.5 shows an enlargement of the interface for CMOS logic levels to SFQ pulses. If the CMOS high logic level is lower than 5 V, the 7 kΩ resistor can be adjusted to trim the input voltage so that the DC-to-SFQ converter receives a maximum input current of 360 *m*A. The on-chip 50 Ω matching elements are generic, although only the high frequency *Rf in* line is matched to 50 Ω off-chip.

In order to keep simulation times short, all off-chip 50 Ω transmission lines are modelled to have a delay time of 100 ps, and on-chip transmission lines only 10 ps. Simulation results for the logic conversion circuit in Figure 6.5 are shown in Figure 6.6.

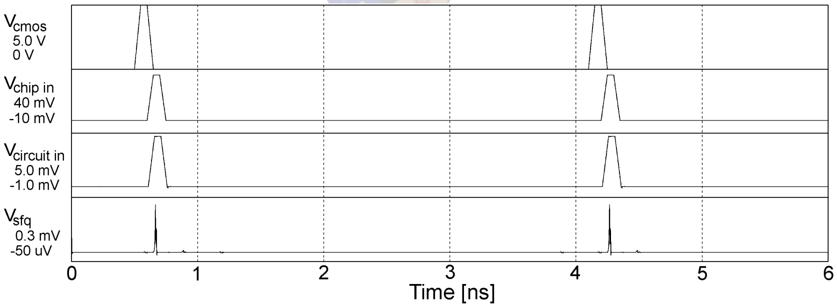


Figure 6.6: Simulated progression from CMOS logic level to SFQ pulse

The compact PFD in Figure 6.3 uses a HUFFLE configured for T flip-flop operation as a hot logic interface (see the discussion in section 4.4.1.2). An extract portraying only the T flip-flop HUFFLE and the voltage state output is shown in Figure 6.7. The impedance of the 20 Ω transmission line was chosen (after simulations on several values) to be small enough to make it practically realizable on chip, but large enough not to load the HUFFLE and cause erroneous state changes.

The output is matched to 50 Ω at the chip edge because it is an rf signal.

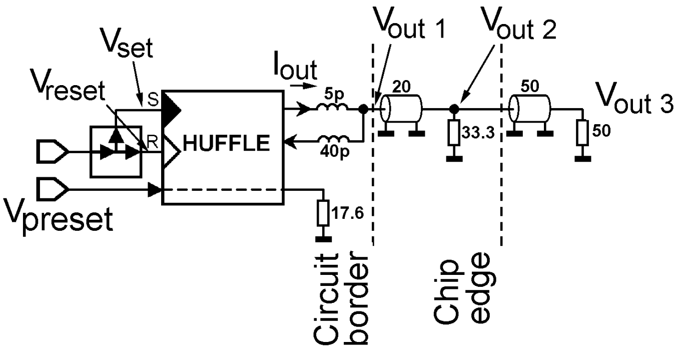


Figure 6.7: Schematic diagram showing HUFFLE connected as T flip-flop with voltage output

Simulation results for the HUFFLE configured as a T flip-flop, and with a voltage output, are shown in Figure 6.8. The results were generated for the same programming sequences as used for those in Figure 6.4. The synchronized *Set* and *Reset* inputs can be seen, as well as the progression from output current to voltage. Once again the transmission line between the circuit border and chip edge has a delay of 10 ps, whereas that of the line between the chip edge and 50 Ω load is 100 ps.

The amplitude of the output voltage ($V_{out\ 3}$) is roughly 300 *m*V$_{\text{peak}}$, or about -60 dBm into a 50 Ω load. Although low noise amplification is clearly needed, the signal is large enough to be detected by a spectrum analyzer.

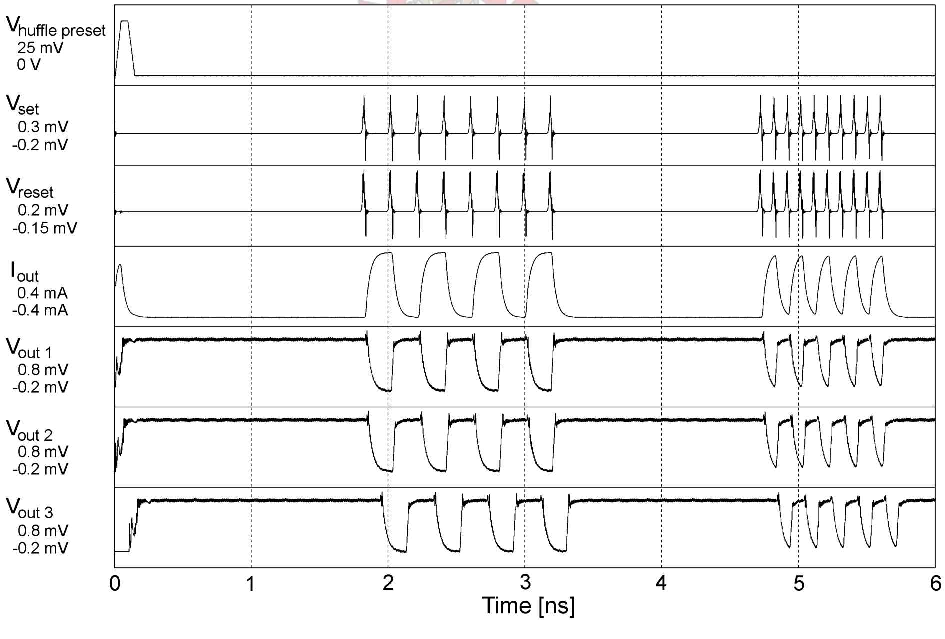


Figure 6.8: Simulated transient response of HUFFLE connected as a T flip-flop with voltage output

6.5 LAYOUT

The University of Stellenbosch does not have an automated layout generator, so that all layouts discussed here and in Appendix A were created manually.

6.5.1 Change of layer process specifications

The old order changeth, yielding place to the new.
Tennyson, The Idylls of the King

In March of 2003, Hypres updated their layer process specifications [16] for the first time in six years. This resulted in a new definition of junction capacitance (up to 50 fF/*m*m^2 for the 1 kA/cm^2 junction, compared to the 38 fF/*m*m^2 used until then in simulation models). The mask-to-wafer junction area bias was also changed, as were several design rules and tolerances. The most important of these is the thickness of layer M1, which is now defined as 135 nm instead of the 200 nm used earlier.

The design rule change necessitated Monte Carlo yield checks on all the circuits developed for this project, as well as changes to all layouts. Fortunately, the effect on inductance was small, with inductors between M2 and ground, and M3 and ground remaining unchanged. Inductance between M1 and ground increased on average by 2.5 %, although the reduction the mask-to-wafer bias from –0.9 *m*m to –0.3 *m*m caused the inductance of some existing layouts to be as much as 10 % high.

The final layouts and circuit diagrams all incorporate the latest design rules. Only some results in Chapters 2 and 3 that were calculated before the design rule change still use the old rules, but conclusions drawn from these results are still valid.

6.5.2 Microstrip discontinuities and matching

In high frequency microstrip lines above a ground plane, such as for the rf inputs in the PFD, reflections off right-angle bends can be reduced if they are chamfered. Such a compensated right-angle bend, with the dielectric and ground plane omitted, is shown in Figure 6.9. The microstrip line has width *w*, and the dielectric has a thickness *h* (the height of the microstrip line above the ground plane). The chamfer makes an angle of 45 degrees with the arms of the microstrip line.

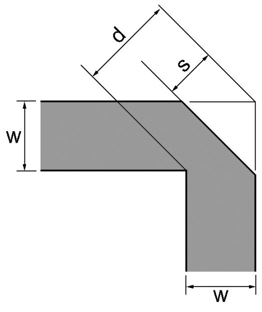


Figure 6.9: Chamfered bend in microstrip line

For optimal compensation when $w/h \geq 0.25$, $1 \leq e_r \leq 25$ and $f \cdot h < 10\,\mathrm{GHz} \cdot \mathrm{mm}$ [66], the relation

$$s/d = 0.52 + 0.65\exp(-1.35w/h) \qquad (6.1)$$

holds true [66] [67].

The Hypres design rules limit the metallic layer masks to a snapped grid of 0.5 *m*m, so that the optimally compensated chamfer can never quite be attained. The mask-to-wafer bias also needs to be accounted for.

Examples of chamfered bends in used in this dissertation can be seen in the layout diagrams in Figure A.16 and Figure A.17. Bends in low frequency (digital input) and dc lines were not compensated.

6.5.3 Discussion on the size of Josephson junction top pads.

Josephson junctions are used here with very large top cover pads (in M2), typically matching the size of the M1 bottom pads. It is common to find that other RSFQ research groups use smaller structures in M2 (see for example the 3D representation in [25]).

The larger top covers are used here to reduce the overall inductance of the connection above the mandatory M1 bottom pad, thereby allowing greater distances between neighbouring junctions.

Furthermore, and as far as is possible, junction top pads are made square. Such pads add the same inductance to any incoming inductor, independent of the direction from which the inductor connects. The standard value for the inductance that a 10 *m*m × 10 *m*m junction cover pad in M2 adds to the inductance between an incoming line and the normal 250 *m*A Josephson junction, has been found from repeated simulations on various structures to vary between 0.3 pH and 0.42 pH, depending on the size of the pad and the width of the transmission line. For fast layout, 0.3 pH is used as a design rule of thumb. Then, once the inductance layout is complete, a *FastHenry* analysis is performed to determine the real inductance.

6.5.4 Moats

Res dura, et regni novitas me talia cogunt molire, et late fines custode tueri.
(Harsh necessity and newness of my kingdom force me to do such things, and to guard all the frontiers.)
Virgil, Aeneid

In order to avoid flux trapping in Josephson junctions when a circuit is cooled to beneath T_C, moats are etched in the ground plane near junctions [68] [69]. These holes trap magnetic flux, and protect the junctions.

The preferred technique here is to use long rectangular moats, for which good results have been obtained [69].

The layout masks in Appendix A clearly show that all the Josephson junctions in the layouts are surrounded by moats. Unlike the moats used by some other researchers, those in the layout of the compact PFD do not surround entire circuit blocks. Moats were kept to between 10 and 20 micrometres in length in order to surround and protect only the Josephson junctions, and limit the possible detrimental effects on the inductance of lines near the moats.

6.6 CONCLUSIONS

A compact PFD has been developed that allows easy testing of the novel latches as well as the concept of reprogrammability. A circuit layout of the compact PFD has been completed, and this layout was then used to create a simulation model that describes the PFD along with its hot logic interfaces.

Simulations were performed to show that the PFD will function correctly, and can be addressed with semiconductor microcontrollers, thus paving the way for the implementation of larger, more complex reprogrammable circuits.

Some layout considerations were also discussed.

Chapter Seven - Towards a full SPGA

And there was something else, the instinct that propelled him out of bed into each unwelcoming day, and that was the desire to know.

Robert Harris, Fatherland

7.1 INTRODUCTION

THERE is considerable interest at the University of Stellenbosch in the design and manufacture of a superconducting programmable gate array (SPGA), and a conceptual discussion on the SPGA has already been done [61]. However, the construction of an SPGA has not been addressed adequately at circuit level before, and address decoding and programming architectures for an SPGA have never before been investigated.

The SPGA requires a considerable amount of logic and routing structures, and cannot yet be implemented in integrated circuits. However, this does not prevent us from exploring the design of an SPGA in expectation of the day when deep submicrometre etching resolutions and self-damped Josephson junctions make the construction and testing thereof possible.

The generic FPGA (see Figure 5.2) as a starting point for the design of an SPGA has already been discussed in section 5.2. Data input, output and routing structures, as well as switches, have also been covered in detail in Chapters 5 and 6. The optimum strategy as far as the number of interconnect resources and switch boxes required to implement a practical SPGA is a research project in itself, and extends beyond the scope of this dissertation. The designs detailed here are therefore first-generation, and merely serve to demonstrate the component level design of SPGAs.

Apart from a short note on switch boxes, this chapter is therefore dedicated to an overview of the building blocks and programming architecture needed to implement a configurable logic block.

7.2 SWITCH BOXES

Switch boxes (see Figure 5.2) occur where routing channels consisting of several data tracks cross each other.

[61] shows a simple switch box implementation, wherein the crossing between two channels – each with four tracks – has only four Crossbar switches. This allows each track in a channel access to only one of those in the other channel, and limits the amount of routing possibilities.

The opposite extreme is to place a Crossbar switch at the crossing of every data track. This will require 16 Crossbar switches for a switch box at a four-track channel crossing. However, it is possible to reduce this number in practice while retaining good routing flexibility [62]. A switch box can be implemented using only the Crossbar and Inline switches developed in Chapter 5, but the optimization thereof for maximum interconnection flexibility goes beyond the scope of this dissertation.

7.3 SUBSYSTEMS OF A CONFIGURABLE LOGIC BLOCK

The CLB performs all logic functions in an FPGA. Each manufacturer of semiconductor FPGAs has a different approach to creating a CLB [60], and the design of a full CLB for an SPGA is a project in itself. The discussion in this chapter will thus be limited to the most important subsystems of a generic CLB, namely the lookup table with its address decoder and programming interface (because they can be constructed with the novel latches developed in Chapter 4).

Any remaining logic functions in a CLB can be realized with the standard set of RSFQ gates and latches [2] [3] and some clever clock-shifting.

7.3.1 Lookup table with 4-to-16 address decoder

The central component of the CLB of a generic FPGA is a lookup table. In such a CLB, four data inputs (labelled A to D) are decoded by a global chip clock to an address in a 16-element lookup table. The corresponding cell in the lookup table is then read out for further processing within the CLB. Since we are for now only interested in the lookup table and address decoder, these components are shown as a simplified schematic circuit diagram in Figure 7.1. Note that reset and preset lines are omitted for clarity.

The lookup table consists of 16 DCRLs, each of which stores one bit and is uniquely addressable. In order to limit the number of components needed to create the address decoder, and also to reduce the decoding operation to a single-clock event, standard RSFQ logic gates such as AND- or OR-gates can be avoided in favour of current coupling.

The technique preferred here is to convert each of the four input bits to a bipolar current through the use of a HUFFLE. The current lines then couple to the SQUID loops of 16 Current-Set switches; each of which is read by the global clock signal. After every switch, one current line changes direction so that each switch is coupled to a unique combination of current lines going up or down. This means that, for any combination of data inputs at *A*, *B*, *C* and *D* in Figure 7.1, only one Current-Set switch will be coupled to four currents in the correct direction to induce switching. This switch then lets the clock pulse through, which reads out the corresponding DCRL in the LUT. The cell addresses are listed in Table 7.2.

Despite the apparent occupation of a lot of die space by the circuit in Figure 7.1, the lines carrying the address decoder currents are very narrow (in the order of 3 micrometres), and 4 of each can pass underneath either side of the dc tee-in of the inter-junction inductance in a long JTL. Pulse routing from the rippled clock line to the DCRL elements in the LUT is therefore not obstructed. Since these lines cross the clock input lines perpendicularly, there are also no stray induced currents that can inhibit the switching action of either the pulse splitters or the switch read inputs.

HUFFLEs are quite useful in small decoders such as this one, where large currents are needed to couple into reading elements. For larger decoders, with more coupling lines, the cumulative uncertainty in the induced current may be too large.

HUFFLEs are also slow elements, with switching speed derating inversely proportional to loop inductance, so that switching speeds for circuits based on the 3 micrometre process may be between 1 GHz and 5 GHz, depending on the inductance of the control current lines.

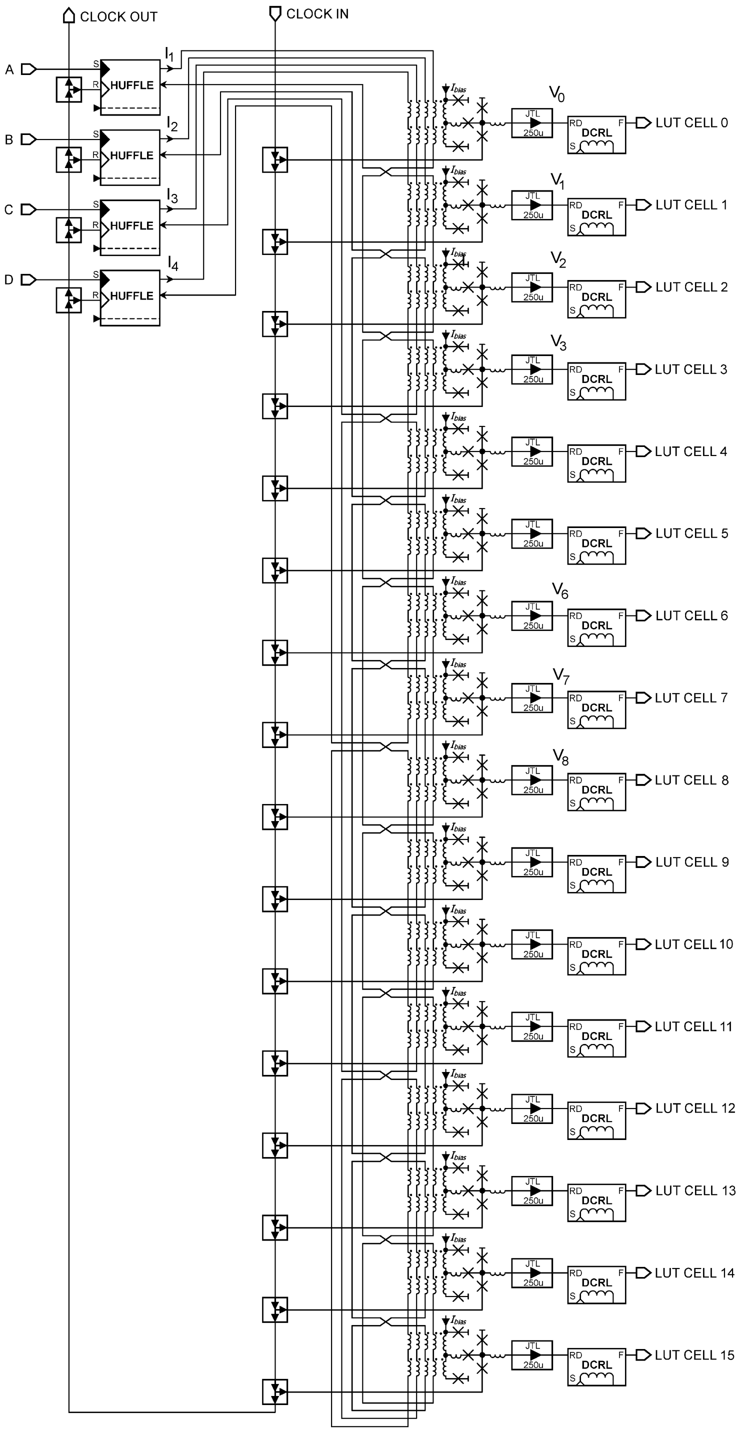


Figure 7.1: Simplified schematic circuit diagram of 4-to-16 address decoder connected to 16-element LUT

A 3D inductance model for the calculation of the self- and mutual inductances of the dc SQUID loop of a Current-Set switch and the four control lines is shown in Figure 7.2.

Calculations on the U-bended model in Figure 7.2 showed that the coupling coefficients between control lines and the SQUID loop A-B are smaller for control lines that are further from the junctions. In this instance, *k* between A-B and I-J is less than half that between A-B and C-D. This is a result of current density in the SQUID loop being higher on the inside of the U-bend, and means that currents flowing through the different control lines will each induce a different amount of current in A-B.

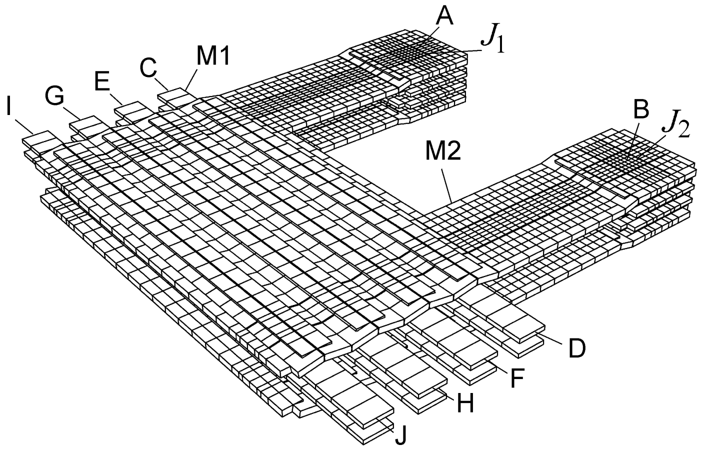


Figure 7.2: 3D inductance calculation model for U-bended dc SQUID loop and 4 control lines, with images included (vertical dimensions stretched for clarity)

A better way to structure the dc SQUID loop is to use symmetry to create nearly equal coupling coefficients. An example of such a structure is shown in Figure 7.3. *FastHenry* indeed shows that the coupling coefficients between each control line and A-B are nearly equal.

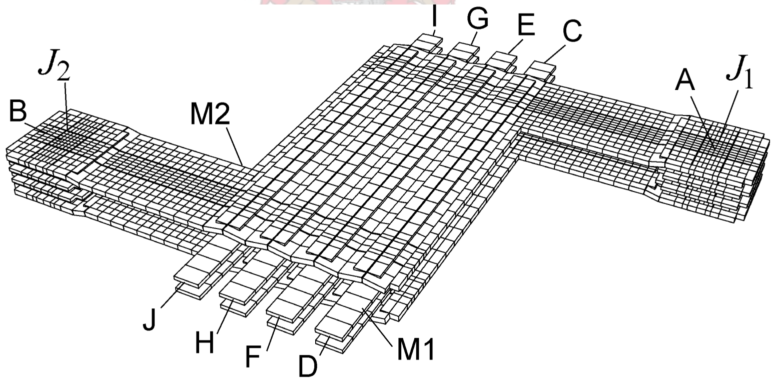


Figure 7.3: 3D inductance calculation model for chicaned dc SQUID loop and 4 control lines, with images included (vertical dimensions stretched for clarity)

Preliminary simulations on an address decoder such as the one in Figure 7.1, with the coupling factors between the SQUID loop and each control line selected to be one quarter of the *k* between a single control line and the loop of a standard Current-Set switch, show that the Current-Set switches select properly. The only problem is that switches for which two or less control currents differ in direction from those of the base (or zero) input state for the CLB do not reset properly when the clock resets the input HUFFLEs. The reason is that the dc SQUID switching curve demands that the total control current must go negative (or close to zero) for the SQUID to reset. This only happens when two or more control currents are reversed.

Several solutions to the reset problem can be implemented. One such solution would be to use pulsed control currents in a fifth control line to reset each switch after every clock cycle. However, this will increase circuit overhead, so that a better solution is simply to deform the SQUID switching curve by the application of a constant control current in a fifth control line. With this bias control technique, the Current-Set switch resets whenever one or more of the control currents flow in the direction opposite to that required for switch selection.

The 3D inductance model for such a SQUID loop is shown in Figure 7.4, with the bias control line clearly visible in the top metal layer.

A discrete logic gate address decoder can also be implemented, but at the cost of added clock cycles and large space occupation.

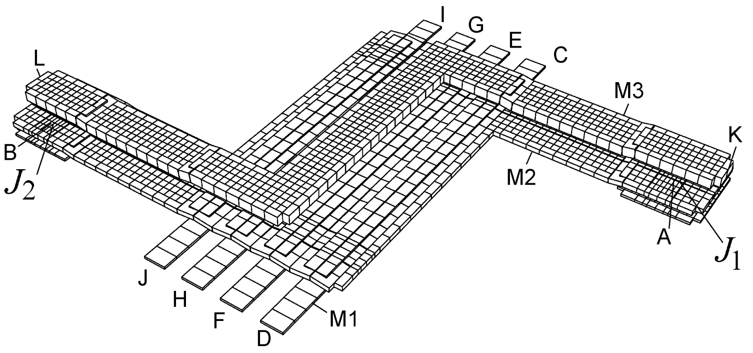


Figure 7.4: 3D inductance calculation model for chicaned dc SQUID loop with 5 control lines (vertical dimensions stretched and images omitted for clarity)

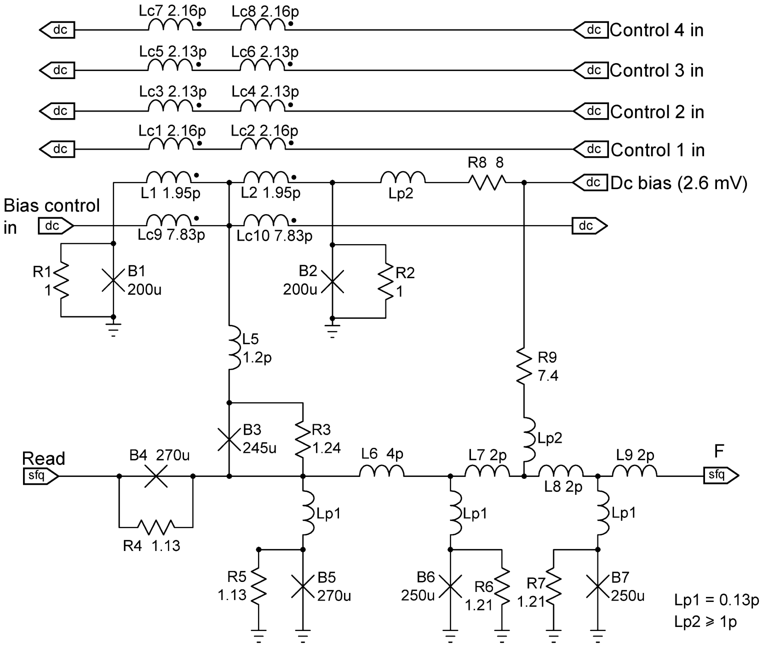


Figure 7.5: Circuit diagram of Current-Set switch altered to interface 4 select control lines and 1 bias control line

A circuit diagram of the Current-Set switch with 5 control lines is shown in Figure 7.5. The coupling coefficients between the SQUID loop and control line inductances are listed in Table 7.1.

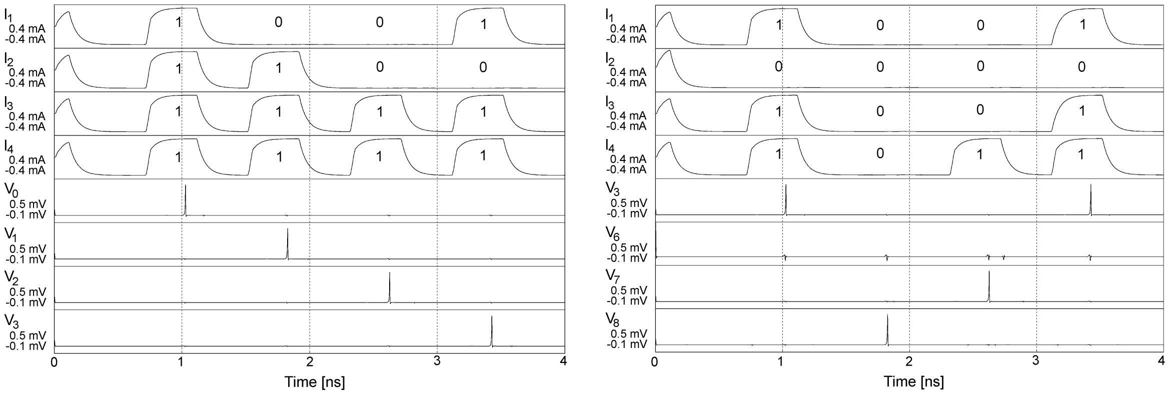
Table 7.1: Coupling coefficients of coupled lines in Current-Set switch with 5 control lines

	SQUID loop	Control line 1	Control line 2	Control line 3	Control line 4	Bias control
SQUID loop	–	0.0963	0.11	0.11	0.0963	0.4364
Control line 1	0.0963	–	0.0639	0.0458	0.037	0.0551
Control line 2	0.11	0.0639	–	0.0593	0.0458	0.0450
Control line 3	0.11	0.0458	0.0593	–	0.0639	0.0450
Control line 4	0.0963	0.037	0.0458	0.0639	–	0.0551
Bias control	0.4364	0.0551	0.0450	0.0450	0.0551	–

Table 7.2: Cell addresses for the 16-element lookup table

LUT cell number	Address $[I_4\ I_3\ I_2\ I_1]$
0	1111
1	1110
2	1100
3	1101
4	1001
5	1011
6	1010
7	1000
8	0000
9	0001
10	0011
11	0010
12	0110
13	0100
14	0101
15	0111

Simulation results for the LUT address decoder in Figure 7.1 are shown in Figure 7.6. The four addresses decoded in Figure 7.6(a) are for cell 0, 1, 2 and 3. In Figure 7.6(b), the addresses for cells 4, 8, 7 and 4 are decoded. The simulated clock frequency is 1.25 GHz. The pulse splitters of the clock distribution section in Figure 7.1 were replaced with concurrent piece-wise linear voltage sources to simplify and shorten the simulation.



(a) **(b)**

Figure 7.6: Simulated response of 4-to-16 address decoder

The clock signal is not shown in Figure 7.6; only the bipolar address currents and a few Current-Set switch outputs.

The bit values of the bipolar address currents I_1, I_2, I_3 and I_4 in Figure 7.6 correspond to the addresses listed in Table 7.2. Although not shown here, clock pulses are applied to the read input of every Current-Set switch in the address decoder every 800 ps, starting at 1 ns. Only one switch at a time is read out, for instance switch (LUT cell number) 0 at 1 ns, when the address is 1111. The simulations show that the address decoder functions correctly, so that it can be used in an SPGA.

7.3.2 LUT programming architecture

A conceptual design for the LUT programming architecture is shown as a schematic circuit diagram in Figure 7.7. The LUT shown here has only 4 elements, but the pattern for extension to 16 elements is obvious. As usual, all the DCRLs share a global reset line, and the HUFFLEs a global preset line. These reset and preset lines are omitted from Figure 7.7 because their connections are obvious.

Outputs are provided for the LUT data and configuration clock signals, so that it is also possible to hook all the CLBs in a column (of an SPGA) to the same long serial register and program an entire CLB column with a single long data word. Such a data word would then be converted to SFQ pulses only once, right before it enters the first CLB.

Alternatively, the LUT data input and configuration clock can be distributed as voltage state signals to every CLB, where a DC-to-SFQ converter can change it to SFQ pulses. In this case all CLBs will receive every data word, but only one will be set to the write mode when the data word has been shifted in.

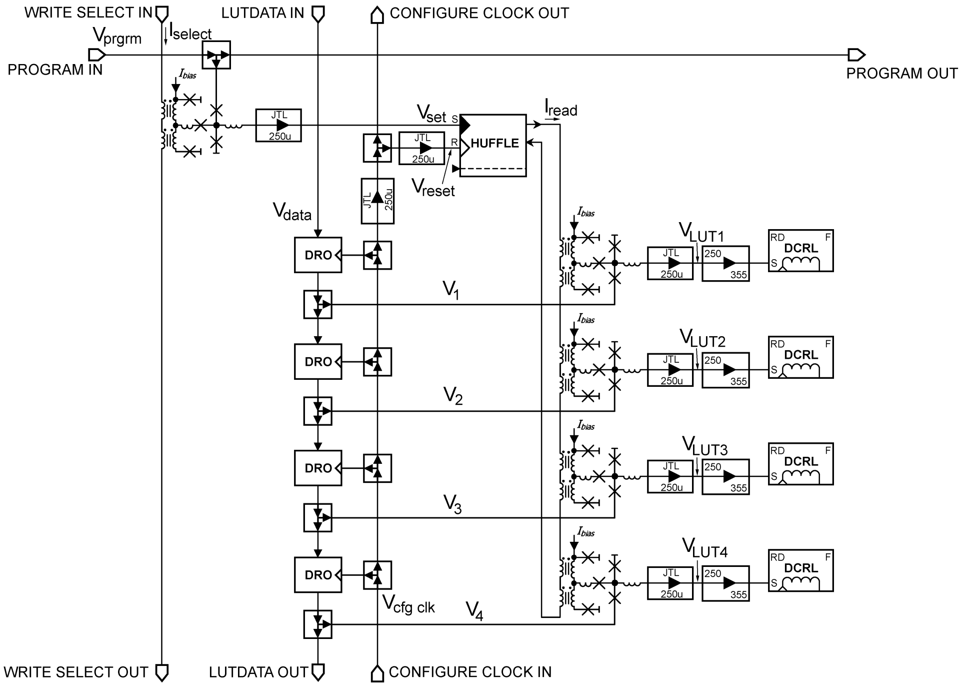


Figure 7.7: Simplified schematic circuit diagram of 4-element LUT programming architecture

Circuit operation is easy to comprehend. The sole purpose of the programming architecture is to route a programming data word to the *Set* inputs of the DCRLs that form the lookup table.

Programming data are entered at *Lutdata In*, and shifted in by *Configure Clock*. A pulse at *Program In* during the last cycle of the configuration clock is used to set the HUFFLE that opens the DCRL *Set* inputs to the programming data. If a switch programming architecture such as that in Figure 5.11 is used, the *Program In* signal will be fed by the SFQ row programming track. In order to let the *Program In* pulse into the CLB, a positive current must be applied to *Write Select In* during the time frame in which the *Program In* pulse is expected. This *Write Select In* current is driven by a bipolar column program line as shown in Figure 5.11. Setting the CLB to the programming mode is therefore equivalent to programming a switch in a switch matrix, as discussed in section 5.5.

The programming sequence for a 16-element LUT, configured in the same way as the 4-element version in Figure 7.7, is shown in Figure 7.8. LUT data are read in starting with that of the last element. Sixteen clock cycles are needed to complete the programming of one LUT. The sixteenth clock also resets the write sequence, which must be started a finite time (t_{D1}) after the application of the fifteenth clock pulse. This time delay is necessary to allow the clock signal to propagate to the *Reset* input of the HUFFLE in the LUT (see Figure 7.7) before the *Program In* pulse arrives at the *Set* input of the same HUFFLE. The time delay depends on layout factors, and a precise value cannot yet be supplied.

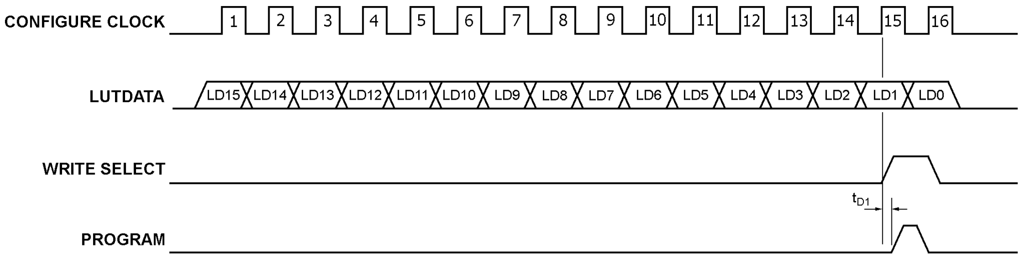


Figure 7.8: Programming sequence of 16-element LUT

Simulation results for the circuit model in Figure 7.7 are shown in Figure 7.9. Trace names correspond to the labels in Figure 7.7. The data input (V_{data}) is 1001.

The other inputs, namely the configuration clock ($V_{cfg\ clk}$), column programming current (I_{select}) and row programming SFQ voltage (V_{prgrm}) are also shown.

The HUFFLE set input (V_{set}) receives an SFQ pulse when I_{select} is positive and an input pulse at *Program In* (V_{prgrm}) is passed through the upper left Current-Set switch in Figure 7.7. This pulse sets the HUFFLE, and I_{bias} flips to its positive value. V_{reset} shows the delayed configuration clock, which serves to reset the HUFFLE and return I_{read} to its negative value. While I_{read} is positive, programming data shifting through the DRO shift register also pass through to the DCRL *Set* inputs in the lookup table. The decoded programming word is visible in V_{LUT1}, V_{LUT2}, V_{LUT3} and V_{LUT4}. The pulse on V_{LUT4} is released earlier than the one on V_{LUT1}, since the configuration clock reaches the last DRO first.

These simulations show that a simple and effective programming circuit for the lookup table of an SPGA does indeed function correctly.

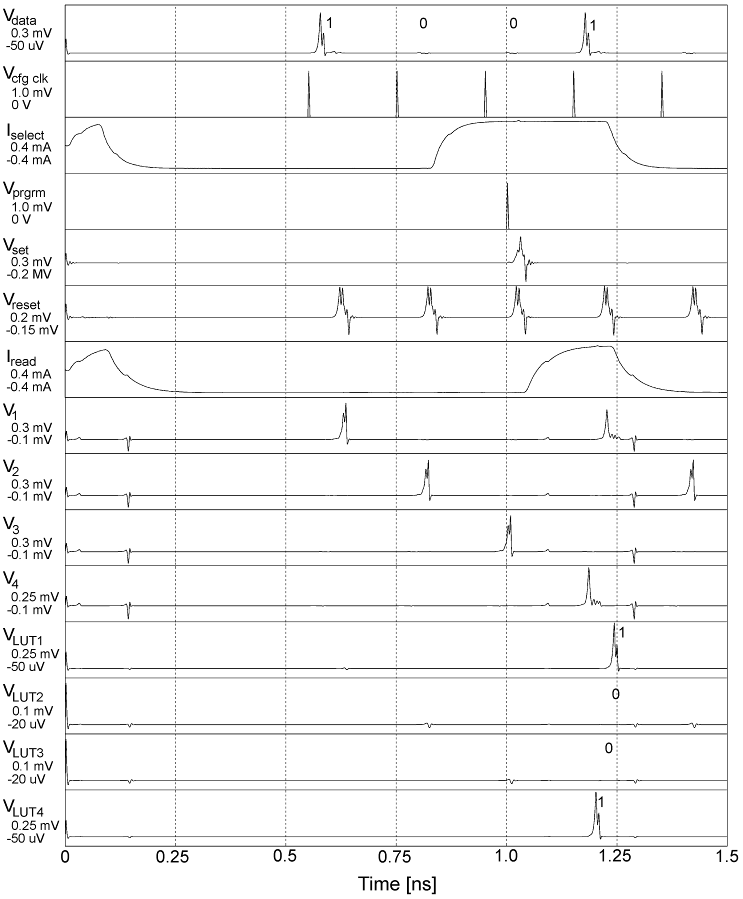


Figure 7.9: Simulated programming of 4-cell LUT

7.4 CONCLUSIONS

The SPGA is still a conceptual circuit, and will remain so until the etching resolution for superconducting circuits improves to sub-micrometre distances. The super tile suggested by [61] can thus not yet be constructed.

However, elementary circuits for constructing lookup tables, address decoders and programming logic for a configurable logic block in an SPGA have been designed and simulated in this dissertation. These circuits utilize the latches introduced in Chapter 4, and build on simulation and layout results obtained from the less complicated programmable devices developed in Chapters 5 and 6. Together with the routing and switching structures discussed in Chapter 5, these circuits form a basis from which SPGAs can be designed when technology finally progresses far enough.

Chapter Eight - Conclusions

This lukewarmness arises… partly from the sceptical temper of men, who do not really believe in new things unless they have been seen to work well.
Niccolò Machiavelli, The Prince

THIS dissertation describes a systematic approach to the design, optimization, layout and verification of superconducting logic circuits based on the RSFQ family.

For circuit optimization, genetic algorithms were implemented from first principles. A program was also developed to handle the entire optimization process, from changing the raw *Spice* circuit file to a Monte Carlo simulation file right through to performing genetic operations and simulations on the circuit files and evaluating the results. Very good results were obtained with these algorithms, even when circuits with very low yields were optimized. The genetic optimizer was also compared to a random optimization technique (which was also programmed into the optimization software), and found to be better. Monte Carlo models based on actual layout tolerances were also developed and implemented.

For the verification of layout quality, with specific emphasis on inductance, routines for the construction of 3D models for use in *FastHenry* were implemented and verified. These routines performed so well that all inductance calculations for actual layout problems were conducted in this way. The program with which the inductance models are generated was also adapted to allow the inclusion of random variations, based on manufacturing tolerances, in the dimensional parameters of the models. In this way it is now possible to establish the true effect of process tolerances on the inductance of a specific layout structure.

With the problems of optimization and layout verification successfully addressed, novel latches for the RSFQ and COSL families, often with non-destructive read capability, were designed and simulated. These latches were then used as the primary building blocks of reprogrammable circuits. The first of these reprogrammable circuits was a programmable frequency divider based on the routing infrastructure of an FPGA. In this way, the first elements needed for a more ambitious SPGA were developed and analysed. A more compact PFD was then developed to ease layout and implementation, and also to provide a test platform for the novel latches.

Finally an overview of the remaining considerations for the design of a full SPGA were treated, and some design problems were identified and solved.

The work in this dissertation proves that fully functional SPGAs can be implemented with RSFQ logic and some hybrid memory elements. A tool kit with clear design examples has also for the first time been assembled for the development of such SPGAs.

References

[1] K. K. Likharev, V. K. Semenov, “RSFQ logic/memory family: A new Josephson-junction technology for sub-terahertz-clock-frequency digital systems,” *IEEE Transactions on Applied Superconductivity*, vol. 1, no. 1, pp. 3-28, March 1991.

[2] Stony Brook University Homepage:
http://pavel.physics.sunysb.edu/RSFQ/RSFQ.html.

[3] C. J. Fourie, *A 10 GHz Oversampling Delta Modulating Analogue-to-Digital Converter Implemented with Hybrid Superconducting Digital Logic*, MScEng thesis, Department of Electronic Engineering, University of Stellenbosch, South Africa, March 2001.

[4] M. Jeffery, W. Perold, T. Van Duzer, “Superconducting complementary output switching logic operating at 5-10 Gb/s,” *Applied Physics Letters*, vol. 69, no. 18, pp. 2746-2748, 28 October 1996.

[5] W. J. Perold, M. Jeffery, Z. Wang, T. Van Duzer, “Complementary output switching logic - A new superconducting voltage-state logic family,” *IEEE Transactions on Applied Superconductivity*, vol. 6, no. 3, pp. 125-131, 3 September 1996.

[6] W. J. Perold, “Complementary Output Switching Logic: a superconducting voltage-state logic family operating at microwave frequencies,” in *Proceedings of the 1998 South African Symposium on Communications and Signal Processing: COMSIG ’98*, pp. 435-440, 1998.

[7] F.J. Rabie, “Superconducting COSL building blocks for ultra-high speed logic circuits,” MScEng Thesis, Department of Electronic Engineering, University of Stellenbosch, South Africa, 1999.

[8] Whiteley Research Inc., 456 Flora Vista Avenue, Sunnyvale, CA 94086. Homepage: http://www.srware.com/.

[9] M. Jeffery, W. J. Perold, Z. Wang, T. Van Duzer, “Monte Carlo optimization of complementary output switching logic circuits,” *IEEE Transactions on Applied Superconductivity*, vol. 8, no. 3, pp. 104-119, September 1998.

[10] C. A. Hamilton, K. C. Gilbert, “Margins and yield in single flux quantum logic,” *IEEE Transactions on Applied Superconductivity*, vol. 1, no. 4, pp. 157-163, December 1991.

[11] Q. P. Herr, M. J. Feldman, "Multiparameter optimization of RSFQ circuits using the method of inscribed hyperspheres," *IEEE Transactions on Applied Superconductivity*, vol. 5, no. 2, pp. 3337-3340, June 1995.

[12] W. J. Perold, private communication.

[13] T. Bäck, U. Hammel, H-P. Schwefel, "Evolutionary computation: comments on the history and current state," *IEEE transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 3-17, April 1997.

[14] J. R. Koza, *Genetic Programming: On the programming of computers by means of natural selection*, The MIT Press, 1992.

[15] K. A. De Jong, "An analysis of the behaviour of a class of genetic adaptive systems," PhD thesis, Dept. Computer and Communication Sciences, University of Michigan, Ann Arbor, MI, 1975.

[16] Hypres, 175 Clearbrook Road, Elmsford, New York 10523, *Niobium Design Rules*, rev. 019, March 2003.

[17] Sergey K. Tolpygo, Director of Fabrication, Hypres Inc., private communication.

[18] P. Z. Peebles, *Probability, Random Variables, and Random Signal Principles*, Third Edition, McGraw-Hill, Inc., 1993.

[19] W. J. Perold, C. J. Fourie, "Modeling superconducting components based on the fabrication process and layout dimensions," *IEEE Transactions on Applied Superconductivity*, vol. 11, no. 1, pp. 345-348, March 2001.

[20] C. J. Fourie, W. J. Perold, "Yield optimization of high frequency superconducting digital circuits with genetic algorithms," *SAIEE Transactions*, vol. 94, no. 2, pp. 11-17, July 2003.

[21] J. M. Johnson, Y. Rahmat-Samii, "Genetic algorithms in engineering electromagnetics," *IEEE Antennas and Propagation Magazine*, vol. 39, no. 4, pp. 7-21, August 1997.

[22] C. J. Fourie, W. J. Perold, "Comparison of genetic algorithms to other optimization techniques for raising circuit yield in superconducting digital circuits," *IEEE Transactions on Applied Superconductivity*, vol. 13, no. 2, pp. 511-514, June 2003.

[23] W. H. Chang, "The inductance of a superconducting strip transmission line," *Journal of Applied Physics*, vol. 50, no. 12, pp. 8129-8134, December 1979.

[24] C. J. Fourie, W. J. Perold, "On using finite segment methods and images to establish the effect of gate structures on inter-junction inductances in RSFQ circuits," *IEEE Transactions on Applied Superconductivity*, vol. 13, no. 2, pp. 539-542, June 2003.

[25] B. Guan, M. J. Wengler, P. Rott, M. J. Feldman, "Inductance estimation for complicated superconducting thin film structures with a finite segment method," *IEEE Transactions on Applied Superconductivity*, vol. 7, no. 2, pp. 2776-2779, June 1997.

[26] M. Kamon, M. J. Tsuk, J. K. White, "FastHenry: A multipole-accelerated 3-D inductance extraction program," *IEEE Transactions on Microwave Theory and Techniques*, vol. 42, no. 9, pp. 1750 – 1758, September 1994.

[27] K. Gaj, Q. P. Herr, M. J. Feldman, "Parameter variations and synchronization of RSFQ circuits," *Applied Superconductivity*, D. Dew-Hughes, Ed. Bristol, U.K.: Inst. Physics, pp. 1733-1736, 1995.

[28] W. H. Chang, "Numerical calculation of the inductances of a multi-superconductor transmission line system", *IEEE Transactions on Magnetics*, vol. MAG-17, no. 1, pp. 764-766, January 1981.

[29] S. R. Whiteley, *FastHenry Version 3.0wr*, February 2001. Available via the Whiteley Research homepage: http://www.srware.com.

[30] The MathWorks, Inc., 24 Prime Park Way, Natick, Massachusetts 01760, *Matlab Version 4 for Windows*, Homepage: http://www.mathworks.com.

[31] S. R. Whiteley, *Sline Version 1.0*, June 1996. Available via the Whiteley Research homepage: http://www.srware.com.

[32] J. Fleischman, *Induct*. Available via the Whiteley Research homepage: http://www.srware.com.

[33] M. M. Khapaev, *LL*. Available from the Moscow state university website: http://www.cmc.msu.ru/vm/sotr/vmhap/ .

[34] M. M. Khapaev, *3D-MLSI*. Available from the Moscow state university website: http://www.cmc.msu.ru/vm/sotr/vmhap/ .

[35] M. Kamon, L. M. Silveira, C. Smithhisler, J. White, *FastHenry User's Guide*, Massachusetts Institute of Technology, 1996.

[36] C. K. Teh, M. Kitagawa, Y. Okabe, "Inductance calculation of 3D superconducting structures with ground plane," *Supercond. Sci. Technol.*, vol. 12, pp. 921-924, 1999.

[37] C. J. Fourie, W. J. Perold, "Reflection plane placement in numerical inductance calculations using the method of images for thin-film superconducting structures," *SAIEE Transactions*, vol. 94, no. 2, pp. 18-24, July 2003.

[38] K. D. Palmer, private communication.

[39] S. Ramo, J. R. Whinnery, T. Van Duzer, *Fields and waves in communication electronics*, Third edition, John Wiley & Sons, Inc., 1994.

[40] L. E. Alsop, A. S. Goodman, F. G. Gustavson, W. L. Miranker, “A numerical solution of a model for a superconductor field problem,” *Journal of Computational Physics*, vol. 31, p. 216-239, 1979.

[41] T. P. Orlando, K. A. Delin, *Foundations of applied superconductivity*, Addison-Wesley Publishing Company, 1991.

[42] T. Van Duzer, C.W. Turner, *Principles of superconductive devices and circuits*, Second Edition, Prentice-Hall, Inc., 1999.

[43] M. Radparvar, Hypres Inc., private communication.

[44] J. W. Nilsson, *Electric Circuits*, Fourth Edition, Addison-Wesley Publishing Company, pp. 506-526, 1993.

[45] G. Hildebrandt, F. H. Uhlmann, G. M. Daalmans, F. R. Bömmel, “A novel approach in calculating V-I curves of a dc-SQUID coupled to a planar input coil,” *IEEE Transactions on Applied Superconductivity*, vol. 6, no. 1, pp. 19-23, March 1996.

[46] A. H. Miklich, J. X. Przybysz, T. J. Smith, “Superconducting thin-film transformers at microwave frequencies,” *IEEE Transactions on Applied Superconductivity*, vol. 9, no. 2, pp. 3062-3065, June 1999.

[47] B. Dimov, H. Toepfer, H. F. Uhlmann, “Analysis of electromagnetic coupling effects in integrated Josephson junction logic devices by the FDTD technique,” *IEEE Transactions on Applied Superconductivity*, vol. 11, no. 1, pp. 1002-1005, March 2001.

[48] A. F. Kirichenko, “High-speed asynchronous data multiplexing/demultiplexing,” *IEEE Transactions on Applied Superconductivity*, vol. 9, no. 2, pp. 4046-4048, June 1999.

[49] L. Zheng, N. Yoshikawa, J. Deng, X. Meng, S. Whiteley, T. Van Duzer, “RSFQ multiplexer and demultiplexer,” *IEEE Transactions on Applied Superconductivity*, vol. 9, no. 2, pp. 3310-3313, June 1999.

[50] I. Kurosawa, A. Yagi, H. Nakagawa, H. Hayakawa, “Single flux-quantum Josephson memory cell using a new threshold characteristic,” *Applied Physics Letters*, vol. 43, no. 11, pp. 1067-1069, 1 December 1983.

[51] A. F. Kirichenko, S. Sarwana, D. K. Brock, M. Radparvar, “Pipelined dc-powered SFQ RAM,” *IEEE Transactions on Applied Superconductivity*, Vol. 11, no. 1, pp. 537-540, March 2001.

[52] M. Morisue, M. Kaneko, H. Hosoya, “A content addressable memory using Josephson Junctions,” *IEEE Transactions on Applied Superconductivity*, vol. 1, no. 1, pp. 48-53, March 1991.

[53] A. F. Hebard, S. S Pei, L.N. Dunkleberger, T.A. Fulton, “A dc-powered Josephson flip-flop,” *IEEE Transactions on Magnetics*, vol. 15, pp. 408-411, 1979.

[54] Y. Hatano, H. Nagaishi, S. Yano, K. Nakahara, H. Yamada, S. Kominami, M. Hirano, “An all dc-powered Josephson logic circuit,” *IEEE Journal of Solid-State Circuits*, vol. 26, no. 8, pp. 1123-1132, August 1991.

[55] Y. Hatano, H. Nagaishi, K. Nakahara, U. Kawabe, “Performance analysis of the Josephson dc flip-flop,” *IEEE Transactions on Applied Superconductivity*, vol. 2, no. 3, pp. 148-155, September 1992.

[56] H. Hasegawa, H. Nagaishi, S. Kominami, H. Yamada, T. Nishino, “A dc-powered Josephson logic family that uses hybrid unlatching flip-flop logic elements (Huffles),” *IEEE Transactions on Applied Superconductivity*, vol. 5, no. 4, pp. 3504-3510, December 1995.

[57] D. F. Schneider, J. C. Lin, S. V. Polonsky, V. K. Semenov, “Broadband interfacing of superconducting digital systems to room temperature electronics,” *IEEE Transactions on Applied Superconductivity*, vol. 5, no. 2, pp. 3152-3155, June 1995.

[58] D. M. Pozar, *Microwave and rf design of wireless systems*, John Wiley & Sons, 2001.

[59] B. Razavi, *Design of analog CMOS integrated circuits*, McGraw Hill, pp.572-574, 2001.

[60] S. D. Brown, R. J. Francis, J. Rose, Z. G. Vranesic, *Field-programmable gate arrays*, Kluwer Academic Publishers, 1992.

[61] P. Gross, *Development of a rapid single flux quantum field programmable gate array*, BEng thesis, Department of Electronic Engineering, University of Stellenbosch, South Africa, 2000.

[62] S. Brown, J. Rose, Z. G. Vranesic, “A detailed router for field-programmable gate arrays,” *IEEE Transactions on Computer-Aided Design*, vol. 11, no. 5, pp. 620-627, May 1992.

[63] D. Mange, E. Sanchez, A. Stauffer, G. Tempesti, P. Marchal, C. Piguet, “Embryonics: a new methodology for designing field-programmable gate arrays with self-repair and self-replicating properties,” *IEEE transactions on Very Large Scale Integration (VLSI) Systems*, vol. 6, no. 3, pp. 387-399, September 1998.

[64] A. Thompson, P. Layzell, R. S. Zebulum, “Explorations in design space: unconventional electronics design through artificial evolution,” *IEEE transactions on Evolutionary Computation*, vol. 3, no. 3, pp. 167-196, September 1999.

[65] Q. Ke, B. J. Dalrymple, D. J. Durand, J. W. Spargo, “Single flux quantum crossbar switch,” *IEEE Transactions on Applied Superconductivity*, vol. 7, no. 2, pp. 2968-2971, June 1997.

[66] R. K. Hoffmann, *Handbook of microwave integrated circuits*, Artech House, Inc., pp. 293-294, 1987.

[67] F. Gardiol, *Microstrip Circuits*, John Wiley & Sons, Inc., p. 88, 1994.

[68] S. Bermon, T. Gheewala, “Moat-guarded Josephson junctions,” *IEEE Transactions on Magnetics*, vol. MAG-19, no. 3, pp. 1160-1164, May 1983

[69] M. Jeffery, T. Van Duzer, J. R. Kirtley, M. B. Ketchen, “Magnetic imaging of moat-guarded superconducting electronic circuits,” *Applied Physics Letters*, vol. 67, no. 12, pp. 1769-1771, 18 September 1995.

Appendix A - Selected circuit schematics and layout masks

THE layout masks shown in this appendix have a visible spot grid of 10 micrometres. The layer key is shown in Figure A.1. All layouts are taken from the compact PFD circuit, so as to include input and output connections.

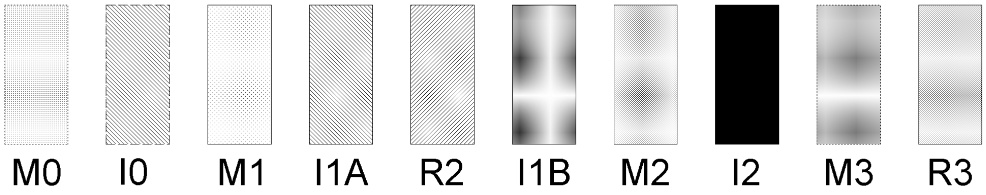


Figure A.1: Layer key for layout masks

A.1 NEW CIRCUITS

A.1.1 RSFQ DCRL

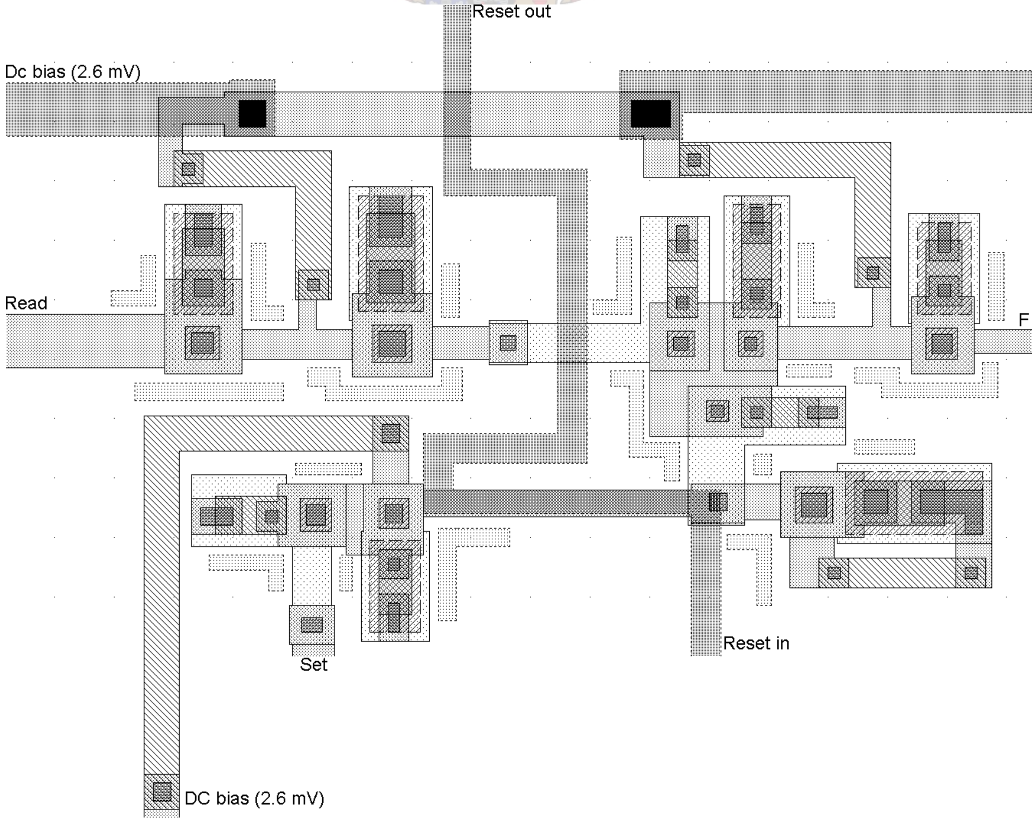


Figure A.2: Layout mask of RSFQ DCRL

A.1.2 Current-Set switch

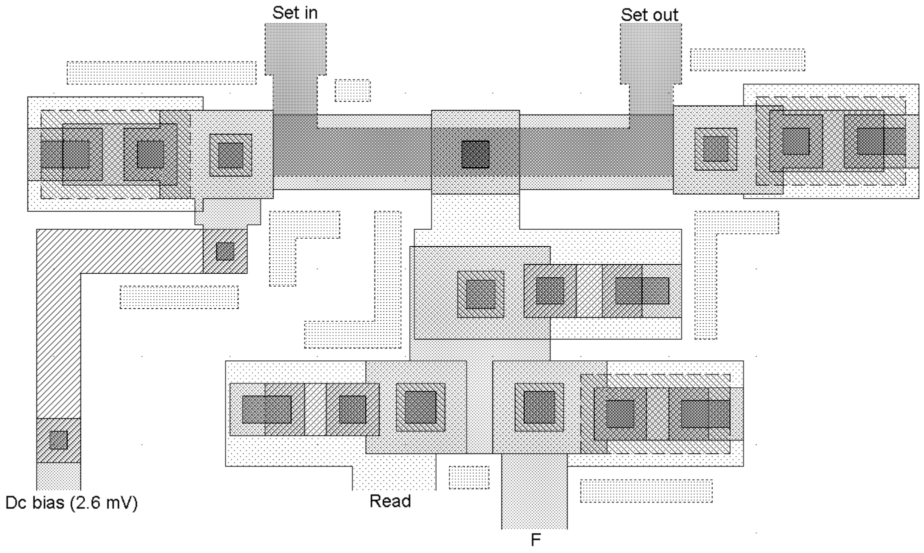


Figure A.3: Layout mask of Current-Set switch

A.1.3 COSL SR flip-flop

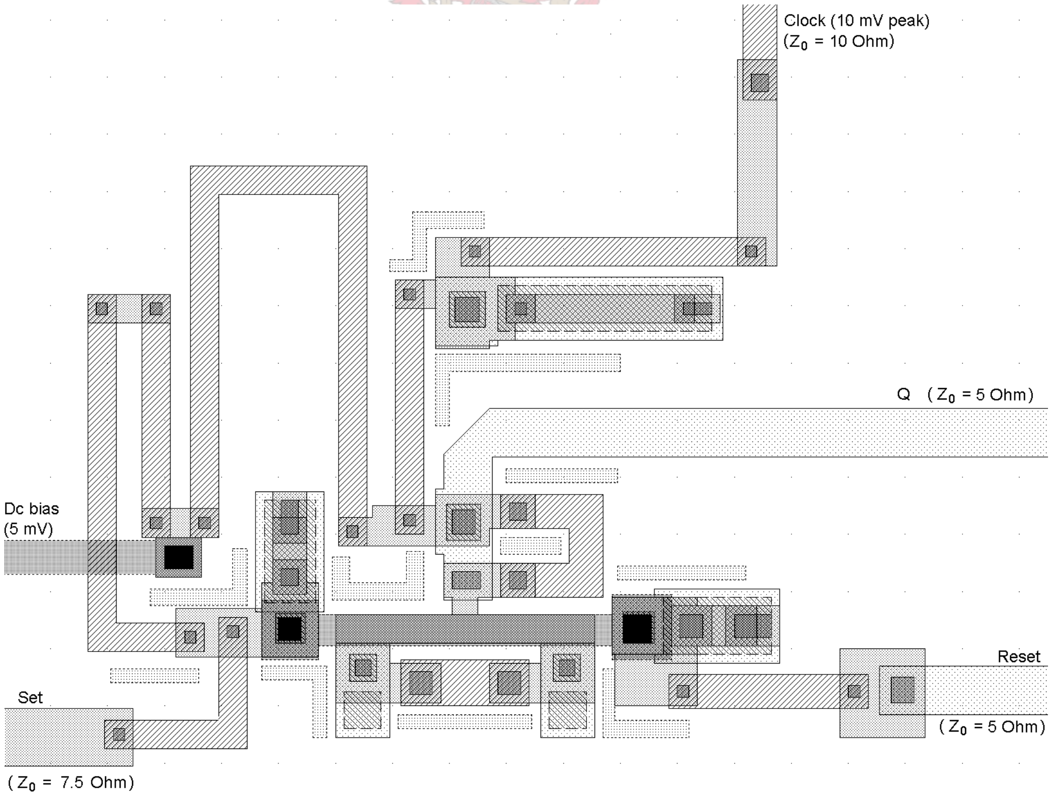


Figure A.4: Layout mask of COSL SRFF

A.1.4 HUFFLE

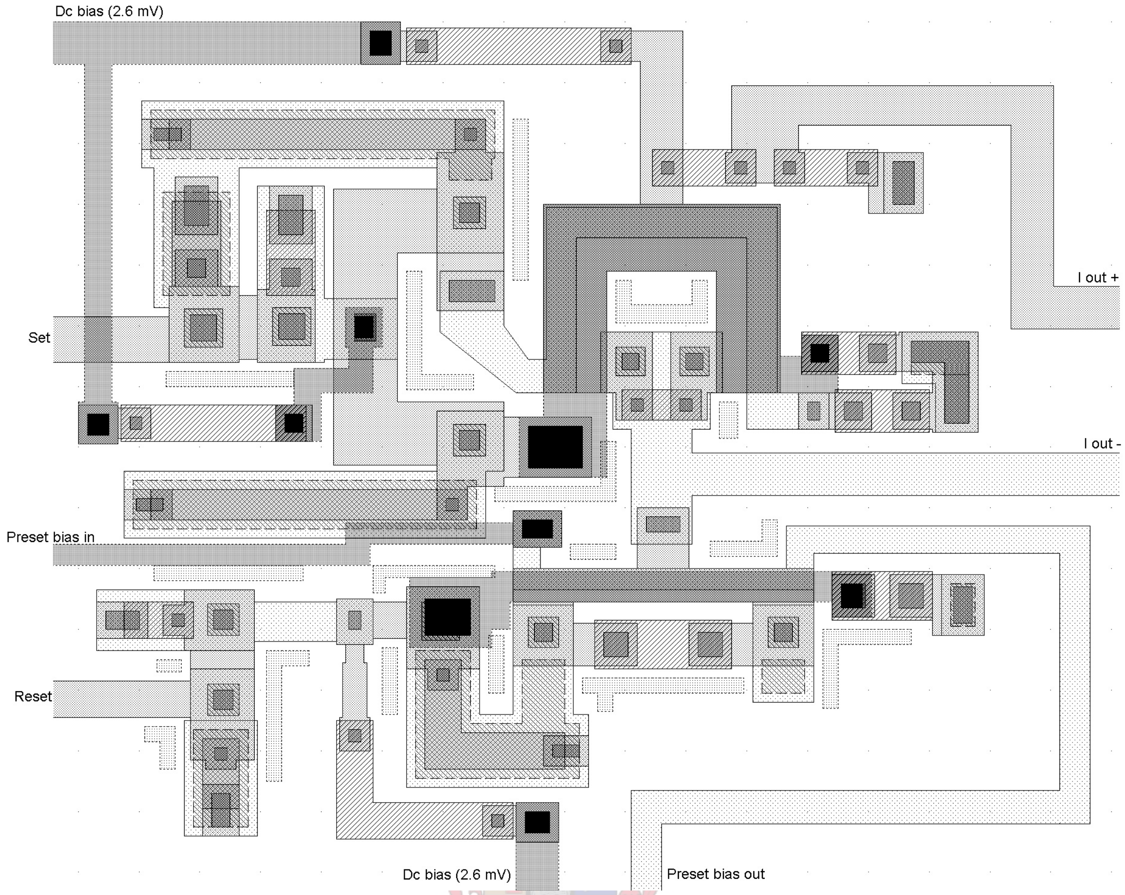


Figure A.5: Layout mask of HUFFLE

A.1.5 RSFQ-to-COSL converter

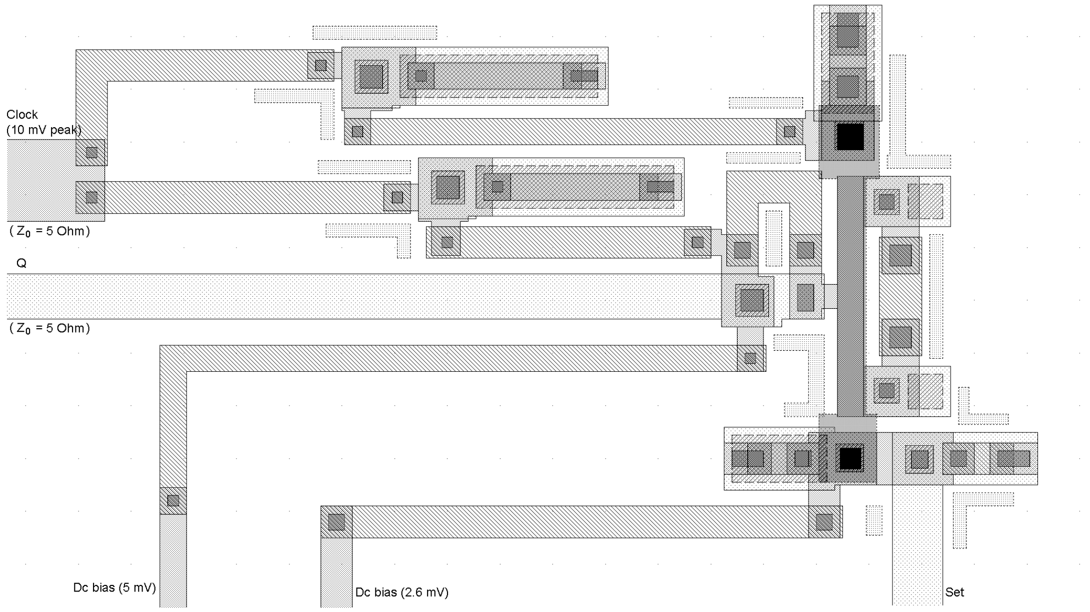
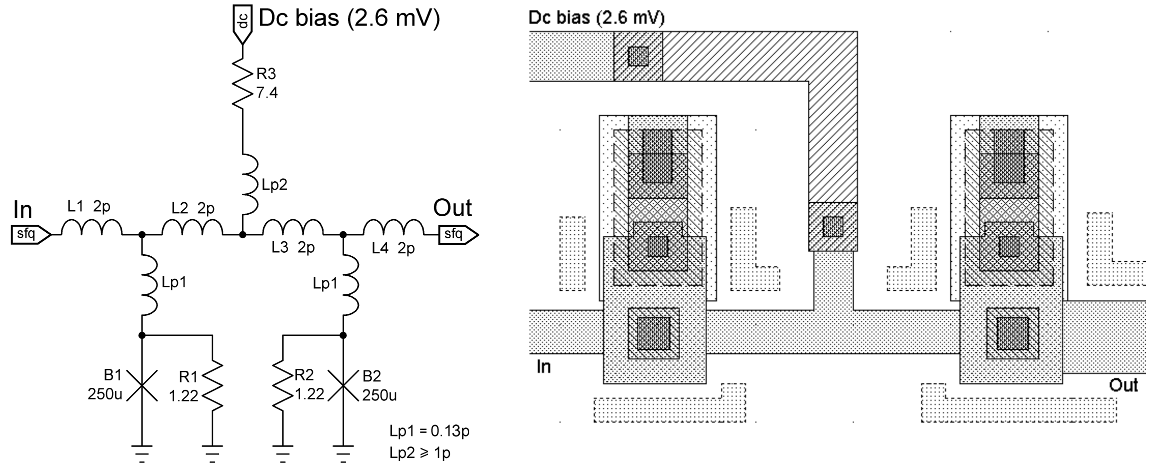


Figure A.6: Layout mask of RSFQ-to-COSL converter

A.2 EXISTING CIRCUITS

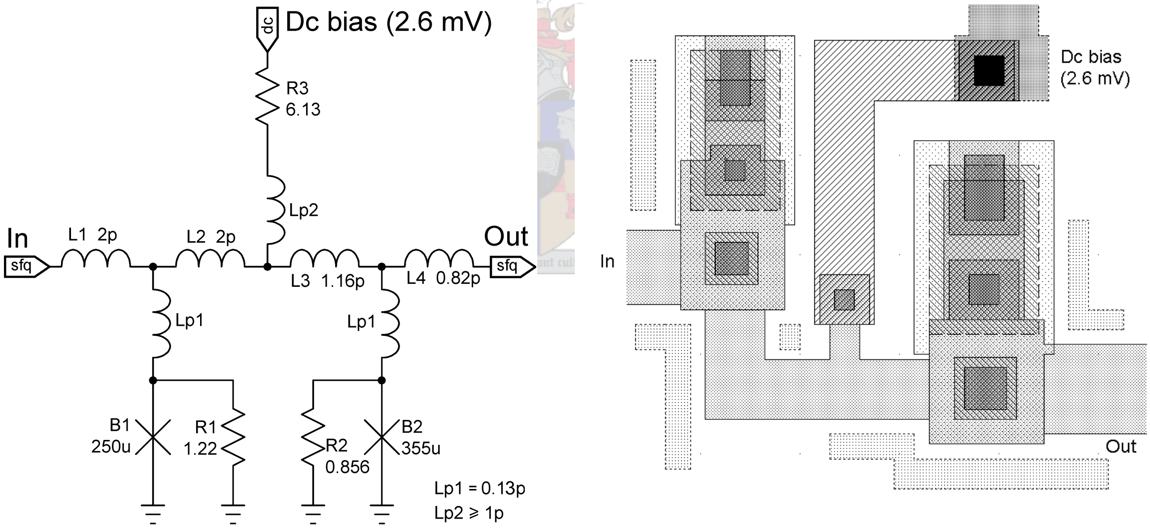
A.2.1 250 mA JTL



(a) (b)

Figure A.7: (a) Circuit schematic and (b) layout mask of 250 mA JTL

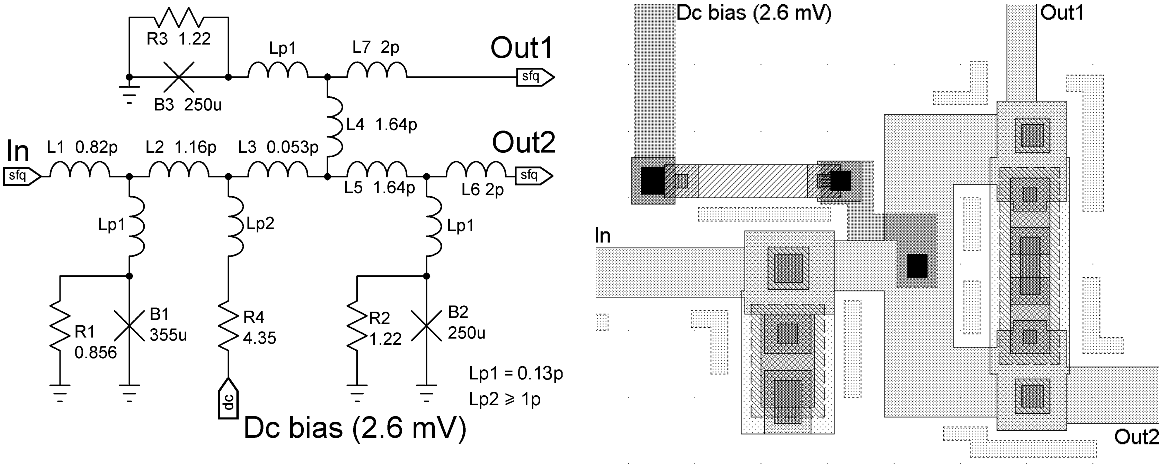
A.2.2 250-355 mA JTL



(a) (b)

Figure A.8: (a) Circuit schematic and (b) layout mask of 250-355 mA JTL

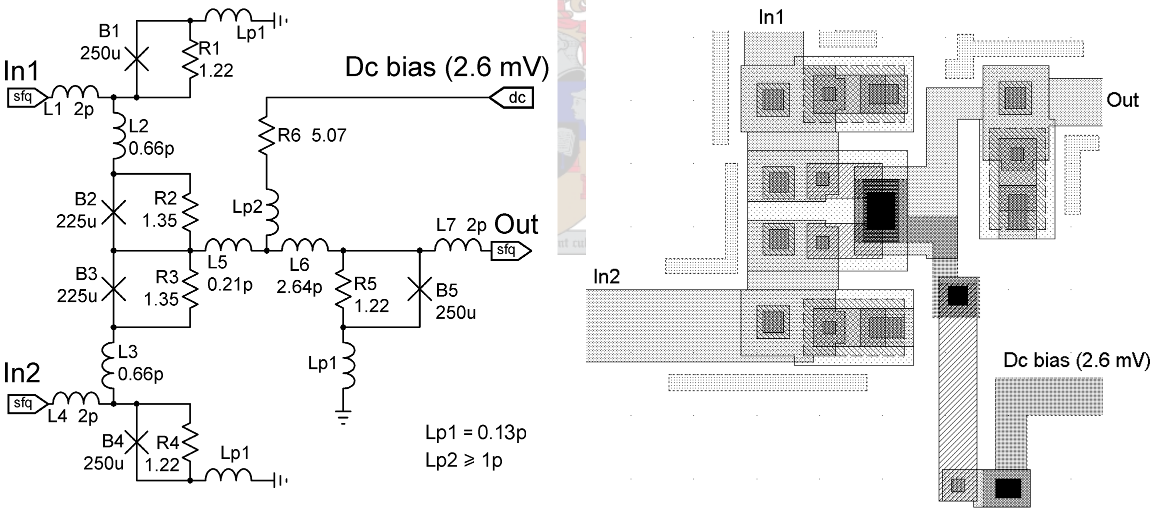
A.2.3 RSFQ pulse splitter



(a) **(b)**

Figure A.9: (a) Circuit schematic and (b) layout mask of RSFQ pulse splitter

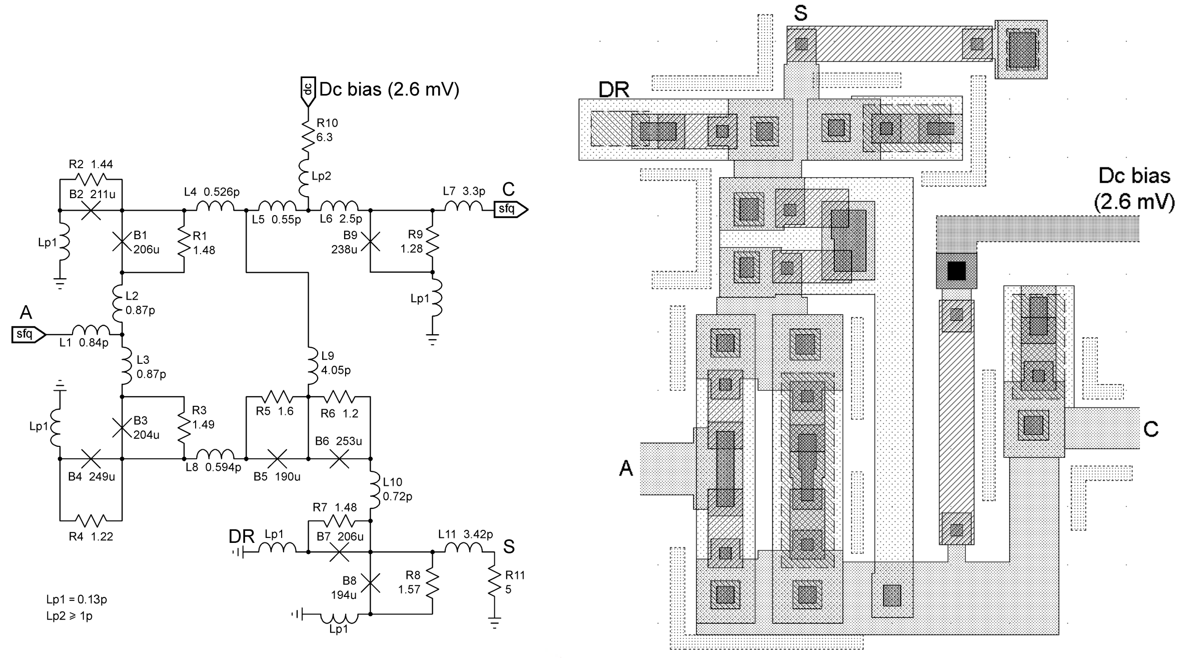
A.2.4 RSFQ pulse merger



(a) **(b)**

Figure A.10: (a) Circuit schematic and (b) layout mask of RSFQ pulse merger

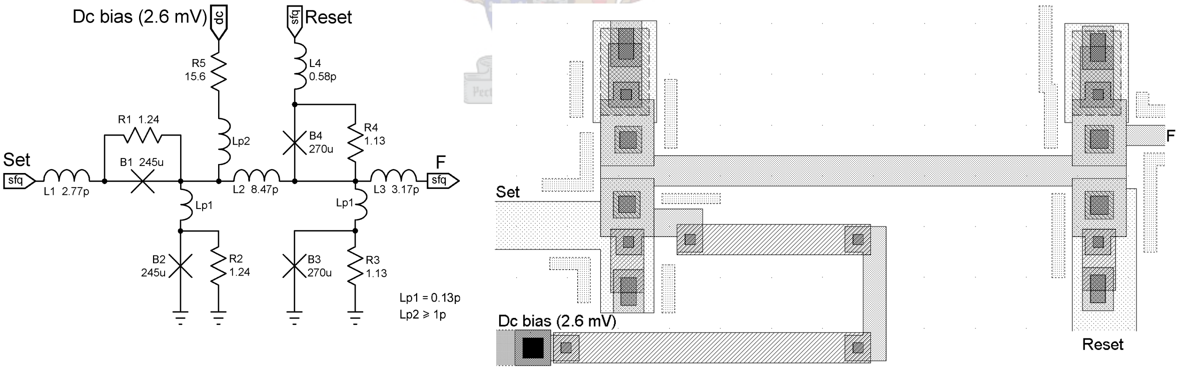
A.2.5 RSFQ T1 flip-flop



(a) **(b)**

Figure A.11: (a) Circuit schematic and (b) layout mask of RSFQ T1 flip-flop

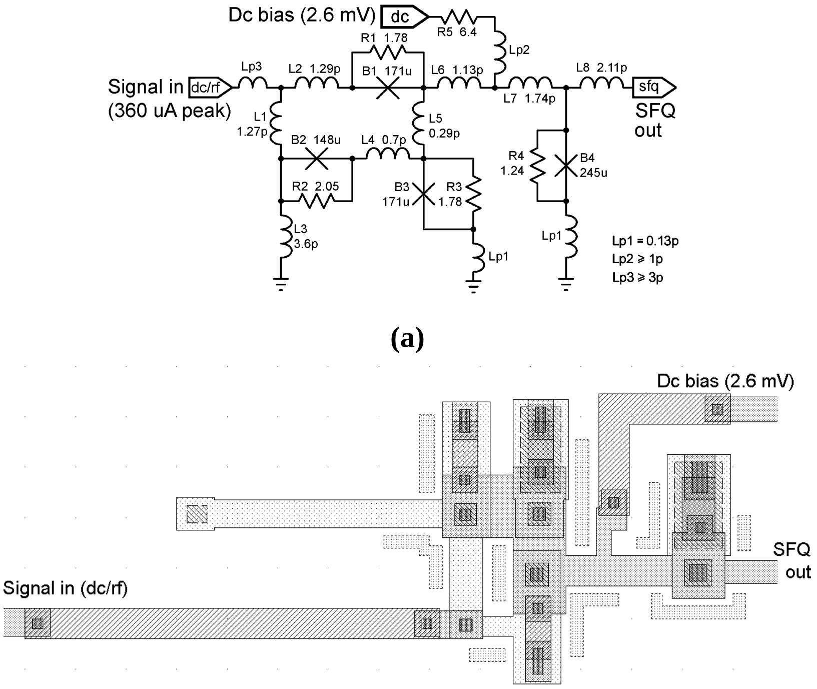
A.2.6 RSFQ DRO



(a) **(b)**

Figure A.12: (a) Circuit schematic and (b) layout mask of RSFQ DRO

A.2.7 DC-to-SFQ converter

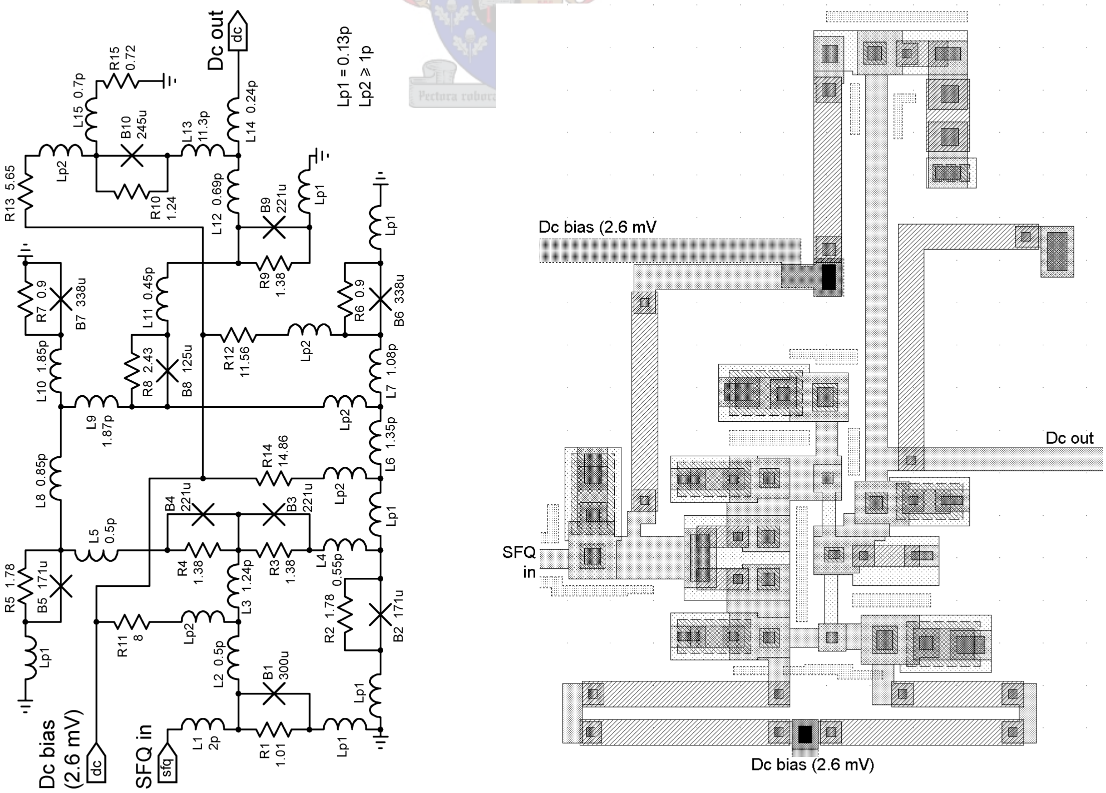


(a)

(b)

Figure A.13: (a) Circuit schematic and (b) layout mask of DC-to-SFQ converter

A.2.8 SFQ-to-DC converter



(a) **(b)**

Figure A.14: (a) Circuit schematic and (b) layout mask of SFQ-to-DC converter

A.3 PFD AND SUBSYSTEMS

A.3.1 Inline switch

Note that the layout of the inline switch differs slightly from the circuit schematic in Figure 5.5(a), in that the pulse splitter for the SFQ program line is moved outside the switch so that the programming pulse does not need to be routed out again. The layout corresponds to the switches in Figure 6.3.

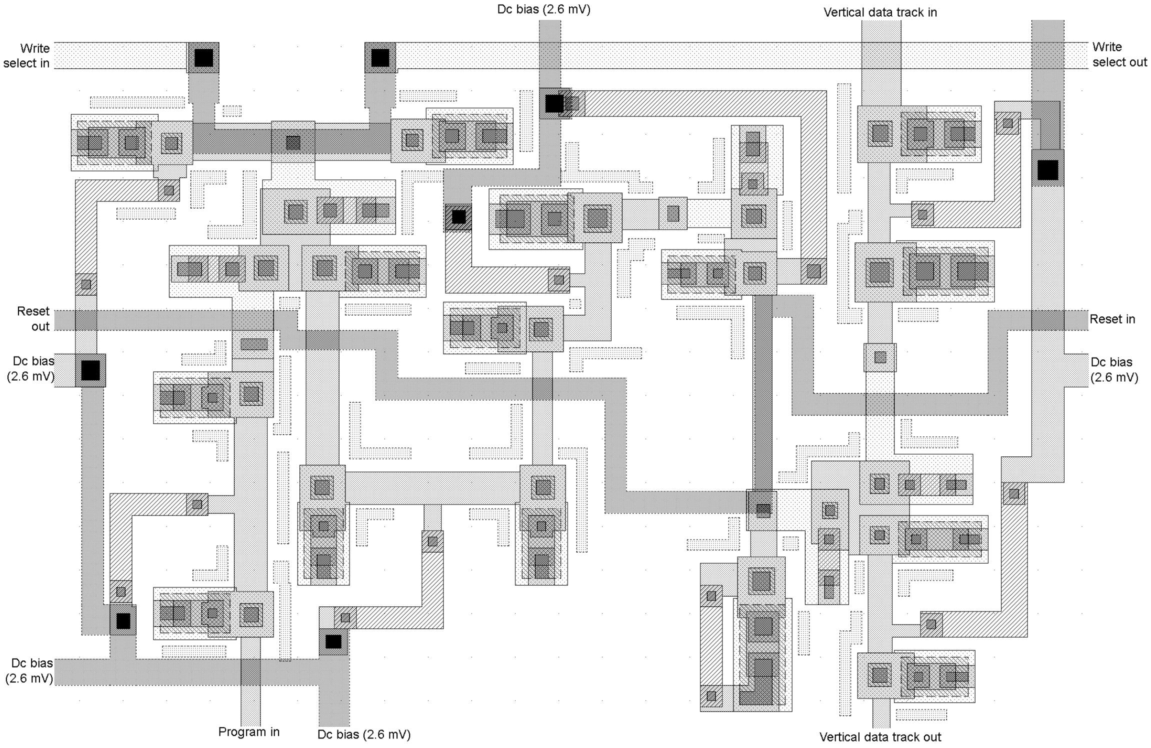


Figure A.15: Layout mask of inline switch for vertical data track

A.3.2 PFD layout

The layout of the compact PFD – with and without HUFFLES – is shown in Figure A.16. Text features etched into the ground plane appear mirrored, since the chip fabrication process mirrors the layout mask to the actual wafer.

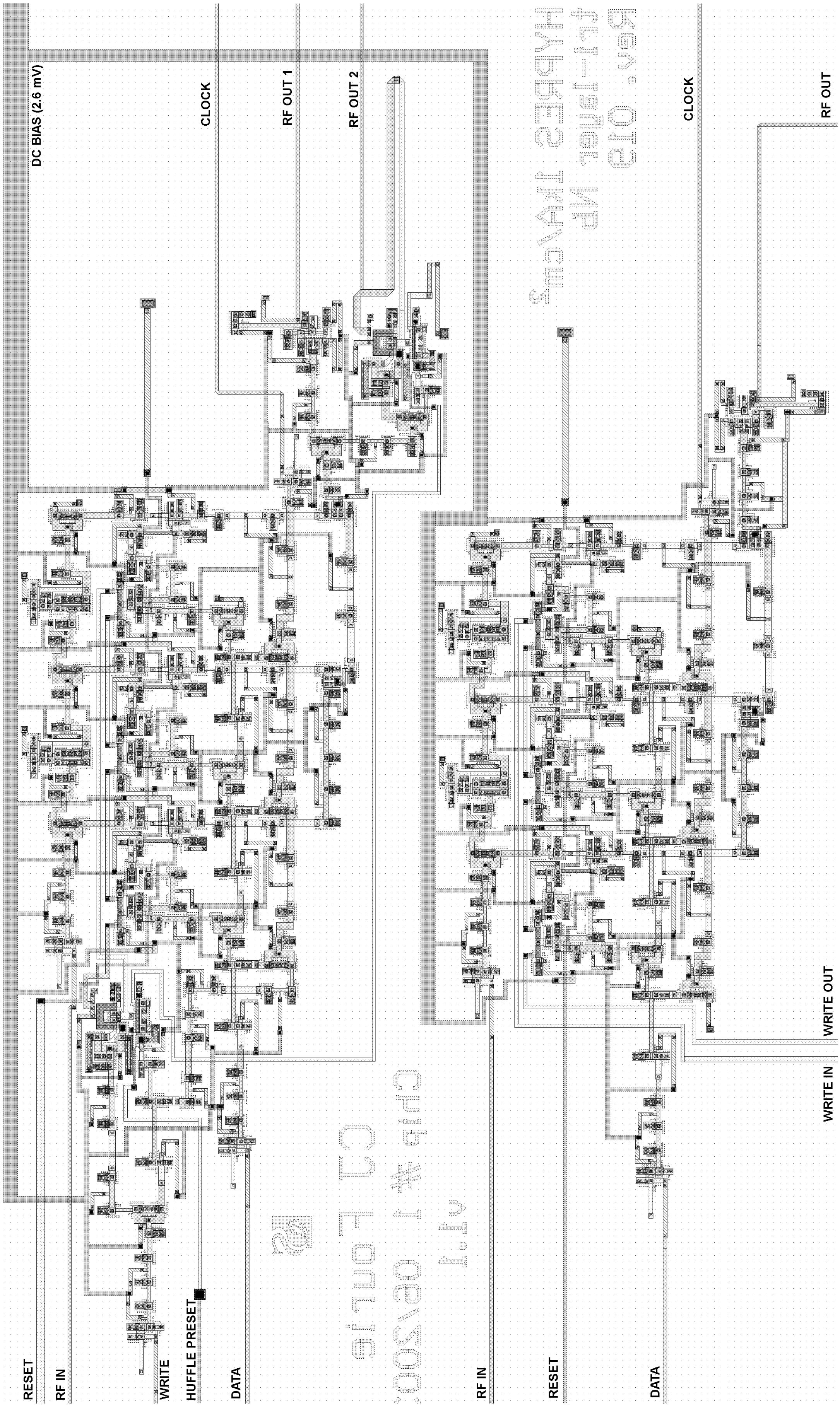


Figure A.16: Layout mask of PFD with HUFFLEs (left) and without HUFFLEs (right)

A.3.3 Complete integrated circuit

The full chip, incorporating both compact PFDs as well as test circuits for the COSL SRFF and RSFQ-to-COSL converter, is shown in Figure A.17.

An SFQ pulse transmission scheme using an existing transmission line driver and receiver [2] is also included for testing.

The remaining structures that do not connect to the standard probe pads are well-defined short circuits, open circuits, 50 Ω loads and through-connected transmission lines, and are to be used for TRL measurements and probe calibrations.

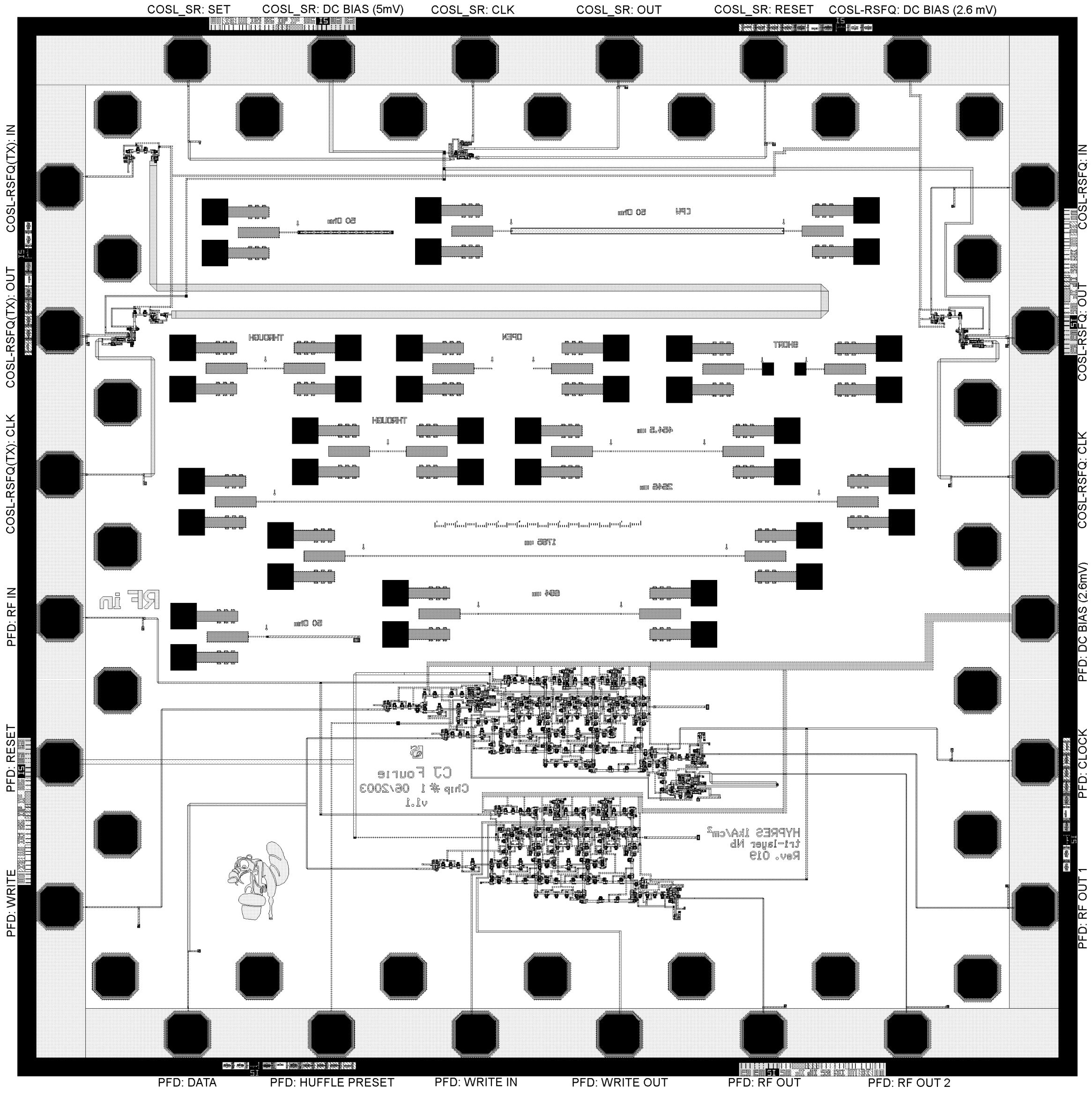


Figure A.17: Layout mask of complete chip for novel components and reprogrammable circuitry test

Appendix B - Selected source code and simulation files

The spice must flow.
Frank Herbert, Dune

B.1 WRSpice

B.1.1 SFQ source

When an SFQ source is desired, a DC-SFQ converter at the input can be replaced by a piece-wise linear voltage source that delivers fluxon-sized input pulses. An example of a 10 GHz pulse train starting at 50 ps is:

```
      Vinput 1 0 pulse 0 824u 50p 2p 3p 0 100p
```

B.1.2 Simulation input file for generic Monte Carlo analysis

The *WRSpice* input file shown in this section is for a single JTL, and demonstrates the setup of a Monte Carlo analysis. The tolerance parameters are derived from Table 2.2, and are implemented as discussed in section 2.2.2.

The total number of simulations equals 441. For this example, the dc bias voltage is trimmed, whereas parasitic inductances are not varied.

Note that a unity area for the Josephson junction model (B-element) corresponds to a critical current of 1000 *m*A (as defined by the *icrit* parameter).

```
* WRSpice Monte Carlo circuit file - single JTL / generic model
.monte
.exec
checkSTP1=10
checkSTP2=10
* global variations
let Jtol = gauss(0.10/3,1)
let Ctol = gauss(0.05/3,1)
let Rtol = gauss(0.20/3,1)
let Ltol = gauss(0.10/3,1)
.endc
.control
if (tsfqpls1*40e-12) < 1.5f or (tsfqpls1*40e-12) > 2.5f
  let checkFAIL=1
end
.endc
.tran 1p 150p 0 0.25p UIC
* local variations
.param Jvar = Jtol*gauss(0.05/3,1)
.param Avar = gauss(0.05/3,1)
.param Rvar = Rtol*gauss(0.05/3,1)
.param Lvar = Ltol*gauss(0.15/3,1)
```

```
.measure tran tsfqpls1 from=50p to=90p avg v(9)
* dc bias voltage is trimmed
V1 4 0 dc $&(2.6m*Rtol*Jtol)
V2 8 0 pulse 0 824u 50p 2p 3p 0 100p
B1 2 0 10 jjhypres1 area=$&(0.25*Avar)
B2 1 0 11 jjhypres2 area=$&(0.25*Avar)
L0 6 5 $&(1.98p*Lvar)
L1 6 2 0.132p
L2 7 1 0.132p
L3 8 6 $&(1.98p*Lvar)
L4 3 5 1p
L5 5 7 $&(1.98p*Lvar)
L6 7 9 $&(1.98p*Lvar)
R0 4 3 $&(7.4*Rvar)
R1 2 0 $&(1.21*Rvar)
R2 1 0 $&(1.21*Rvar)
R3 9 0 5
.model jjhypres1 jj(rtype=1, cct=1, icon=10m, vg=2.8m, delv=0.08m,
+  icrit=$&(1m*Jvar), r0=30, rn=1.64706, cap=$&(5.0p*Ctol))
*Nb 1000 A/cm2   area = 100 square microns
.model jjhypres2 jj(rtype=1, cct=1, icon=10m, vg=2.8m, delv=0.08m,
+  icrit=$&(1m*Jvar), r0=30, rn=1.64706, cap=$&(5.0p*Ctol))
*Nb 1000 A/cm2   area = 100 square microns
```

B.1.3 *Simulation input file for Monte Carlo analysis on layout model*

The *WRSpice* input file in this section features layout extracted tolerances.

```
* WRSpice Monte Carlo circuit file - single JTL / layout model
.monte
.exec
checkSTP1=10
checkSTP2=10
* global variations
let Jtol = gauss(0.10/3,1)
let Ctol = gauss(0.05/3,1)
let Rtol = gauss(0.20/3,1)
let LtolM2 = gauss(0.0339/3,0)
.endc
.control
if (tsfqpls1*40e-12) < 1.5f or (tsfqpls1*40e-12) > 2.5f
  let checkFAIL=1
end
.endc
.tran 1p 150p 0 0.25p UIC
* local variations
.param Jvar = Jtol*gauss(0.05/3,1)
.param Avar250 = gauss(0.05/(2.5*3),1)
.param Rvar5thick = Rtol*gauss(0.055/3,1)
.param Rvar6thick = Rtol*gauss(0.045/3,1)
.param Lvar1 = gauss(0.0235/3,0)
.measure tran tsfqpls1 from=50p to=90p avg v(9)
* dc bias voltage is trimmed
V1 4 0 dc $&(2.6m*Rtol*Jtol)
V2 8 0 pulse 0 824u 50p 2p 3p 0 100p
B1 2 0 10 jjhypres1 area=$&(0.25*Avar250)
B2 1 0 11 jjhypres2 area=$&(0.25*Avar250)
L0 6 5 $&(1.98p*(1 + LtolM2 + Lvar1))
```

```
L1 6 2 0.132p
L2 7 1 0.132p
L3 8 6 $&(1.98p*(1 + LtolM2 + Lvar1))
L4 3 5 1p
L5 5 7 $&(1.98p*(1 + LtolM2 + Lvar1))
L6 7 9 $&(1.98p*(1 + LtolM2 + Lvar1))
R0 4 3 $&(7.4*Rvar5thick)
R1 2 0 $&(1.21*Rvar6thick)
R2 1 0 $&(1.21*Rvar6thick)
R3 9 0 5
.model jjhypres1 jj(rtype=1, cct=1, icon=10m, vg=2.8m, delv=0.08m,
+  icrit=$&(1m*Jvar), r0=30, rn=1.64706, cap=$&(5.0p*Ctol))
*Nb 1000 A/cm2   area = 100 square microns
.model jjhypres2 jj(rtype=1, cct=1, icon=10m, vg=2.8m, delv=0.08m,
+  icrit=$&(1m*Jvar), r0=30, rn=1.64706, cap=$&(5.0p*Ctol))
*Nb 1000 A/cm2   area = 100 square microns
```

B.2 FASTHENRY 3.0WR

B.2.1 Example for microstrip over ground plane

B.2.1.1 INPUT FILE

```
* #1 CJF - superconducting microstripline, low impedance
.Units um
.freq fmin=10e9 fmax=10e9 ndec=1

* conductor
N1 x=0 y=0 z=.25
N2 x=0 y=100 z=.25
E1 N1 N2 w=5 h=.2 nwinc=15 nhinc=10 lambda=.09

* "ground plane"
N3 x=0 y=0 z=-0.05
N4 x=0 y=100 z=-0.05
N5 x=-7.5 y=0 z=-0.05
N6 x=-7.5 y=100 z=-0.05
N7 x=7.5 y=0 z=-0.05
N8 x=7.5 y=100 z=-0.05
E2 N3 N4 w=5 h=.1 nwinc=10 nhinc=5 lambda=.09
E3 N5 N6 w=10 h=.1 nwinc=10 nhinc=5 lambda=.09
E4 N7 N8 w=10 h=.1 nwinc=10 nhinc=5 lambda=.09
.equiv N3 N5 N7
.equiv N4 N6 N8
* short one end
.equiv N2 N4
* input port
.external N1 N3
.end
```

B.2.1.2 FASTHENRY OUTPUT

```
Row 1:  n1  to  n3
Impedance matrix for frequency = 1e+010 1 x 1
             0     +0.468281j
```

B.2.1.3 IMPEDANCE MATRIX TO INDUCTANCE CONVERSION

The impedance matrix contains numbers in real (resistive) and imaginary (reactive) components. In B.2.1.2, the resistance is zero, and the reactance is 0.468281j. Frequency equals 10 GHz. Impedance relates to inductance through

$$Z_L = j2pfL \ , \tag{B.1}$$

so that L = 7.45 pH.

This is a simple, non-segmented structure with sufficient filaments, so that no scaling is necessary.

When the method of images is used, the inductance calculated with *FastHenry* is double that of the same structure over ground, and must be divided by 2 to get the actual value.

B.2.2 Sample output for two-conductor mutual inductance calculation with the method of images

A simulation on the structure shown in Figure D.1(l), with 1×3 filamentation in width and height, yields the following output:

```
Row 2:  n756  to  n2001
Row 1:  n17  to  n1262
Impedance matrix for frequency = 1e+010 2 x 2
             0     +0.812285j            0     +0.521652j
            0     +0.522156j            0      +1.78438j
```

The self-inductance of the conductor in layer M2 can be found from the first impedance value as

$$L_{AB} = \frac{0.812285}{2\times 2\times p\times 10^{10}} \times 0.9834 = 6.357 \text{ pH} \ . \tag{B.2}$$

The extra factor two in the denominator of (B.2) accounts for the image area, whereas the scaling factor (0.9834) is used to convert the inductance value for 1×3 (width and height) segment filamentation to the asymptote value as described in section 3.5.2.

B.3 INDUCT

B.3.1 Example for two microstrips over ground plane

B.3.1.1 INPUT FILE

Filename: "m2m1" (No extension.) Structure: M2 and M1 lines over ground plane.

```
100 -10.0 -0.1  10.0 0.0 0.09 20  6 2 2 3 1
1   -3 0.15 3 0.285 0.09 10 10 2 2 0 2
2   -2.5 0.485 2.5 0.785 0.09 10 10 2 2 0 2
```

B.3.1.2 COMMAND LINE

```
induct < m2m1
```

B.3.1.3 SIMULATION OUTPUT (ONSCREEN)

All results are in pH/*m*m.

```
c:\usr\local\bin>induct < m2m1
Input is of the form:
Conductor number, (x,y) lower left, (x,y) upper right,penetration depth, x divis
ions, y divisioins, x ratio, y ratio,x type, and y type

The input file is from standard input, so use redirection to use a file.
Output is to standard output.
100 -10.0 -0.1  10.0 0.0 0.09 20  6 2 2 3 1
1   -3 0.15 3 0.285 0.09 10 10 2 2 0 2
2   -2.5 0.485 2.5 0.785 0.09 10 10 2 2 0 2
finished subdivide
finished evaluating q matrix
inverted q matrix
reduced q matrix into s matrix
inverted s matrix
===***********Inductance matrix***********===
0.069300        0.054713
0.054713        0.122839
```

Appendix C – Genetic optimization program overview

C.1 ALGORITHMIC FLOW CHARTS

IN this section, the algorithmic flow charts that describe the genetic optimization routine are shown. The flow charts are slightly simplified, as the real program uses a computer network to access a remote *Spice* server, and is therefore event driven.

Figure C.1 shows the start-up routine, which determines whether or not the previous optimization sequence was merely suspended, and, if so, whether the current circuit file is the same as the one previously optimized. This routine then decides whether or not to allow a continued optimization.

Figure C.2 shows how the genetic optimization sequence starts, and how a first generation of children are spawned from random variations on the nominal parent, *Eve*.

Figure C.3 shows how the optimization loop is handled, child for child and generation for generation.

Figure C.4 to Figure C.7 show the most important subroutines. These handle fitness calculation, procreation, pairing and crossover, and mutation.

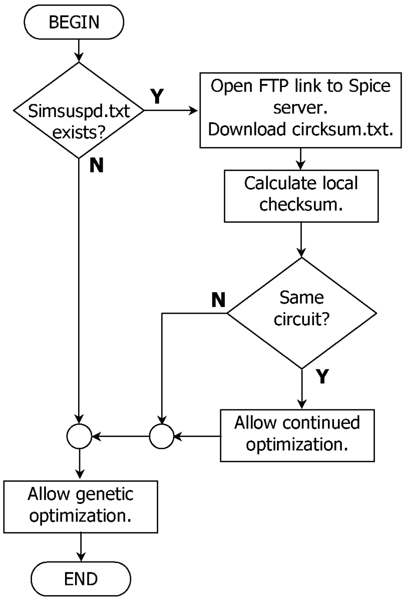

BEGIN
Simsuspd.txt exists?
Y
N
Open FTP link to Spice server. Download circksum.txt.
Calculate local checksum.
Same circuit?
N
Y
Allow continued optimization.
Allow genetic optimization.
END

Figure C.1: Algorithmic flow chart of genetic optimization program activation

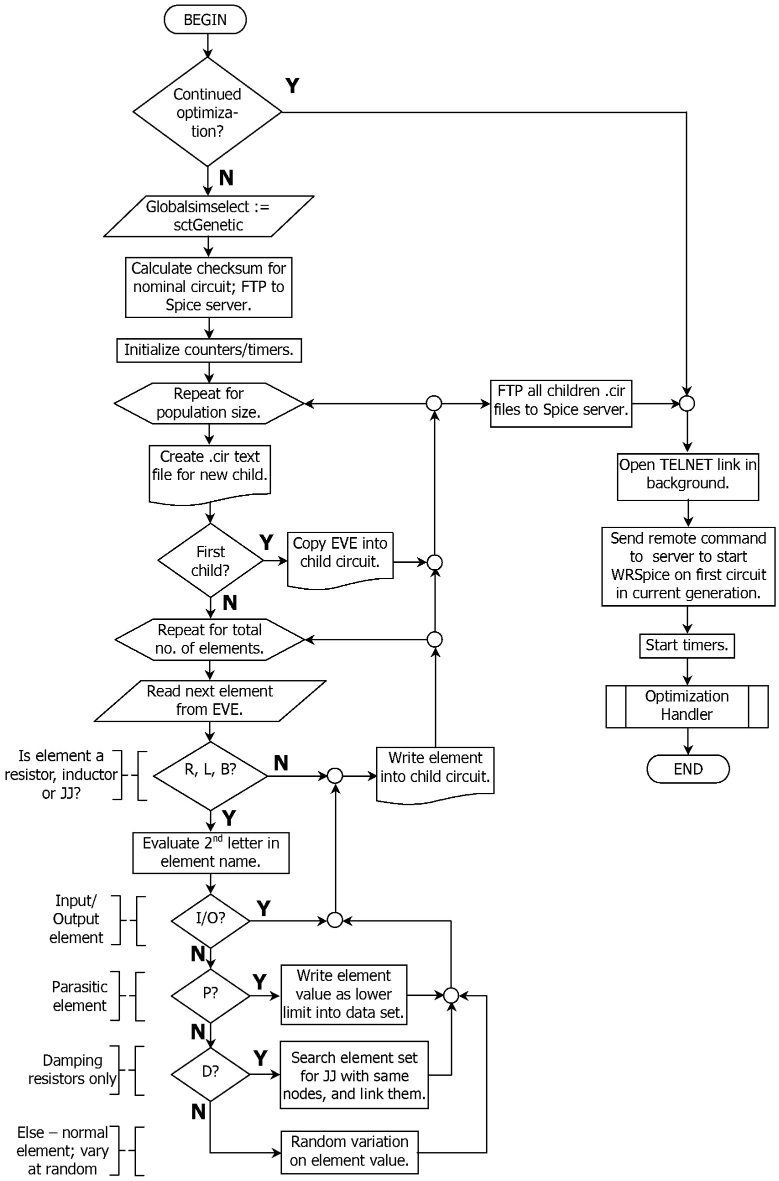

BEGIN
Continued optimiza-tion?
Y
N
Globalsimselect := sctGenetic
Calculate checksum for nominal circuit; FTP to Spice server.
Initialize counters/timers.
Repeat for population size.
FTP all children .cir files to Spice server.
Create .cir text file for new child.
Open TELNET link in background.
First child?
Y
Copy EVE into child circuit.
N
Send remote command to server to start WRSpice on first circuit in current generation.
Repeat for total no. of elements.
Start timers.
Read next element from EVE.
Optimization Handler
Is element a resistor, inductor or JJ?
R, L, B?
N
Write element into child circuit.
END
Y
Evaluate 2nd letter in element name.
Input/ Output element
I/O?
Y
N
Parasitic element
P?
Y
Write element value as lower limit into data set.
N
Damping resistors only
D?
Y
Search element set for JJ with same nodes, and link them.
N
Else – normal element; vary at random
Random variation on element value.

Figure C.2: Algorithmic flow chart of genetic optimization sequence initiation

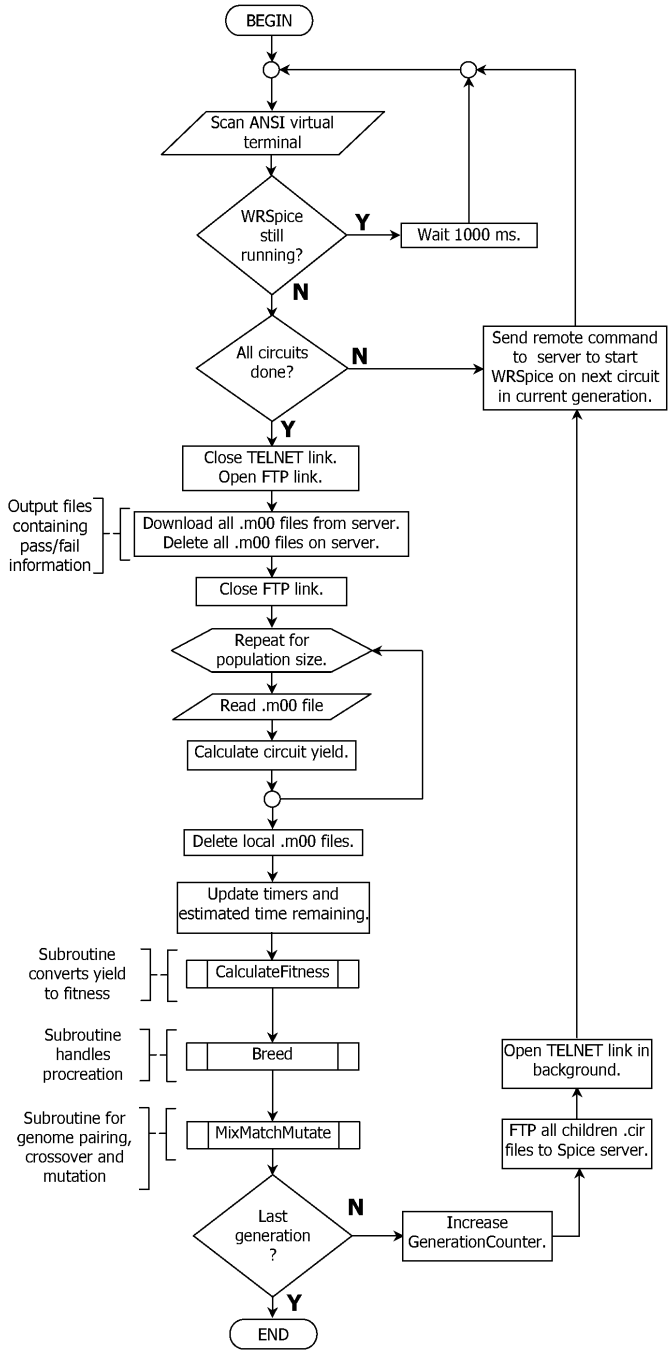

BEGIN
Scan ANSI virtual terminal
WRSpice still running?
Y
Wait 1000 ms.
N
All circuits done?
N
Send remote command to server to start WRSpice on next circuit in current generation.
Y
Close TELNET link. Open FTP link.
Output files containing pass/fail information
Download all .m00 files from server. Delete all .m00 files on server.
Close FTP link.
Repeat for population size.
Read .m00 file
Calculate circuit yield.
Delete local .m00 files.
Update timers and estimated time remaining.
Subroutine converts yield to fitness
CalculateFitness
Subroutine handles procreation
Breed
Subroutine for genome pairing, crossover and mutation
MixMatchMutate
Last generation ?
N
Increase GenerationCounter.
FTP all children .cir files to Spice server.
Open TELNET link in background.
Y
END

Figure C.3: Algorithmic flow chart showing operation of optimization handler

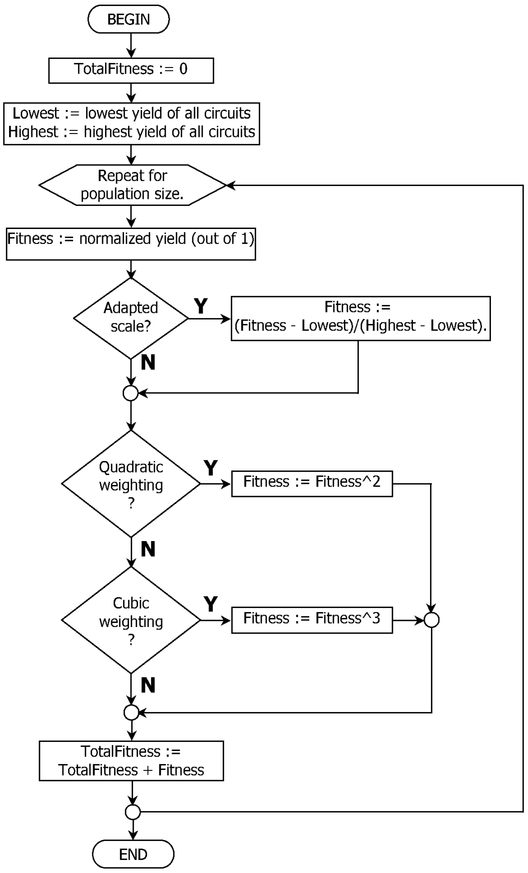

BEGIN
TotalFitness := 0
Lowest := lowest yield of all circuits
Highest := highest yield of all circuits
Repeat for population size.
Fitness := normalized yield (out of 1)
Adapted scale?
Y
Fitness := (Fitness - Lowest)/(Highest - Lowest).
N
Quadratic weighting ?
Y
Fitness := Fitness^2
N
Cubic weighting ?
Y
Fitness := Fitness^3
N
TotalFitness := TotalFitness + Fitness
END

Figure C.4: Algorithmic flow chart of procedure "CalculateFitness"

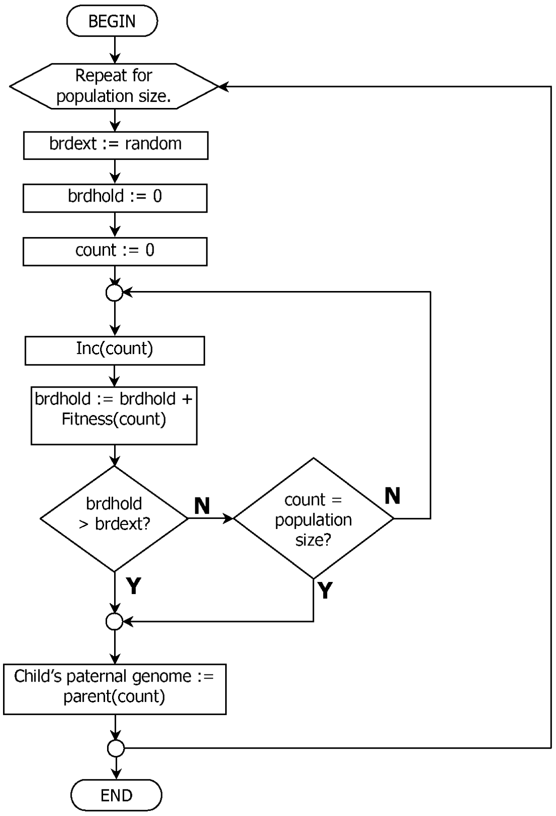

BEGIN
Repeat for population size.
brdext := random
brdhold := 0
count := 0
Inc(count)
brdhold := brdhold + Fitness(count)
brdhold > brdext?
N
count = population size?
N
Y
Y
Child's paternal genome := parent(count)
END

Figure C.5: Algorithmic flow chart of procedure "Breed"

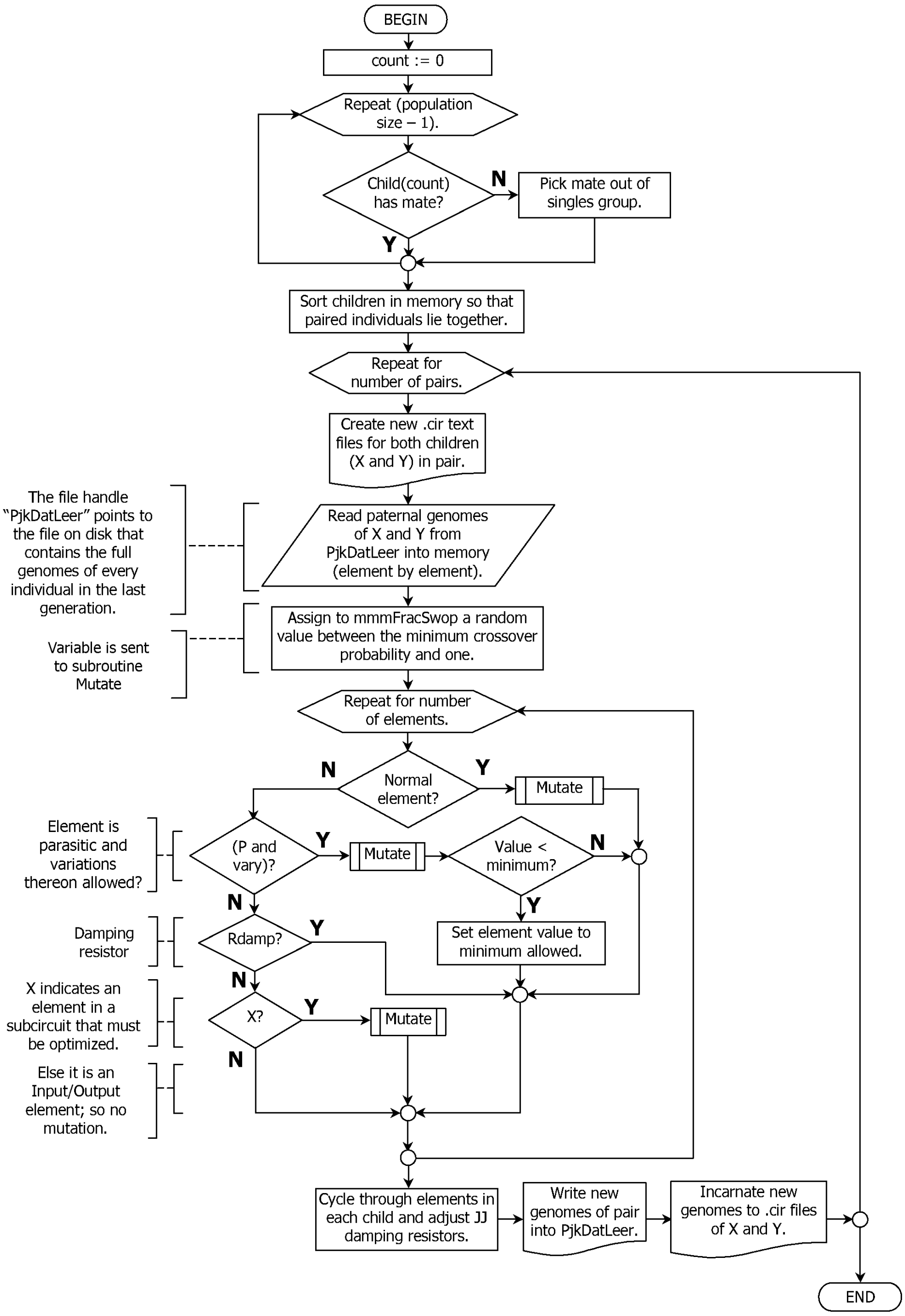

BEGIN
count := 0
Repeat (population size – 1).
Child(count) has mate?
N
Pick mate out of singles group.
Y
Sort children in memory so that paired individuals lie together.
Repeat for number of pairs.
Create new .cir text files for both children (X and Y) in pair.
The file handle "PjkDatLeer" points to the file on disk that contains the full genomes of every individual in the last generation.
Read paternal genomes of X and Y from PjkDatLeer into memory (element by element).
Variable is sent to subroutine Mutate
Assign to mmmFracSwop a random value between the minimum crossover probability and one.
Repeat for number of elements.
N
Normal element?
Y
Mutate
Element is parasitic and variations thereon allowed?
(P and vary)?
Y
Mutate
Value < minimum?
N
N
Y
Damping resistor
Rdamp?
Y
Set element value to minimum allowed.
N
X indicates an element in a subcircuit that must be optimized.
X?
Y
Mutate
N
Else it is an Input/Output element; so no mutation.
Cycle through elements in each child and adjust JJ damping resistors.
Write new genomes of pair into PjkDatLeer.
Incarnate new genomes to .cir files of X and Y.
END

Figure C.6: Algorithmic flow chart of procedure "MixMatchMutate"

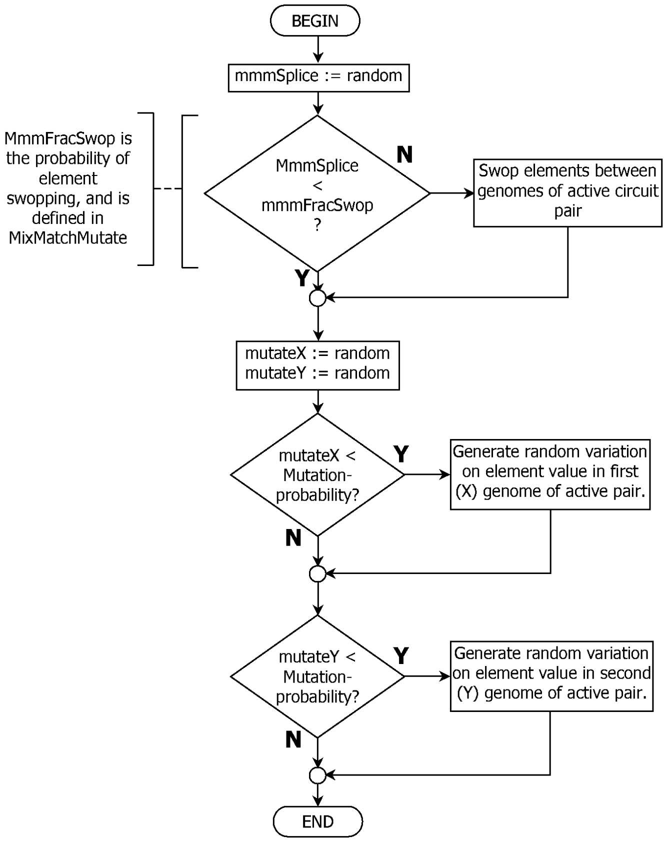

BEGIN
mmmSplice := random
MmmFracSwop is the probability of element swopping, and is defined in MixMatchMutate
MmmSplice < mmmFracSwop ?
N
Swop elements between genomes of active circuit pair
Y
mutateX := random
mutateY := random
mutateX < Mutation-probability?
Y
Generate random variation on element value in first (X) genome of active pair.
N
mutateY < Mutation-probability?
Y
Generate random variation on element value in second (Y) genome of active pair.
N
END

Figure C.7: Algorithmic flow chart of procedure "Mutate"

C.2 SCREENSHOTS AND OPERATING OVERVIEW

Before an optimization sequence can be performed, the *Spice* circuit file (with .cir extension) of the circuit under investigation is loaded into the optimization program, which has provisionally been named *SuperTool*. When *WRSpice* is used, the circuit file can be dumped from the schematic circuit diagram through a single command.

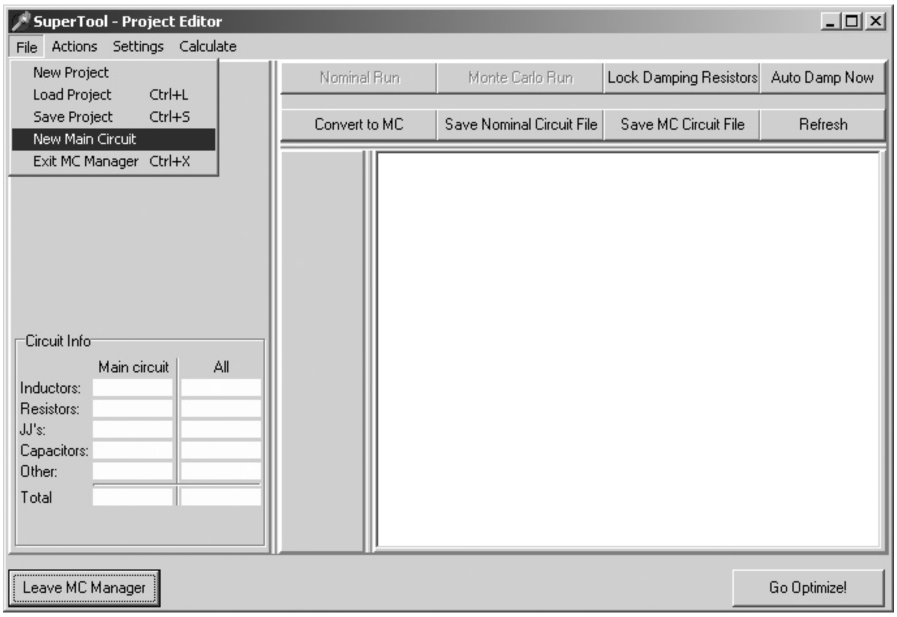


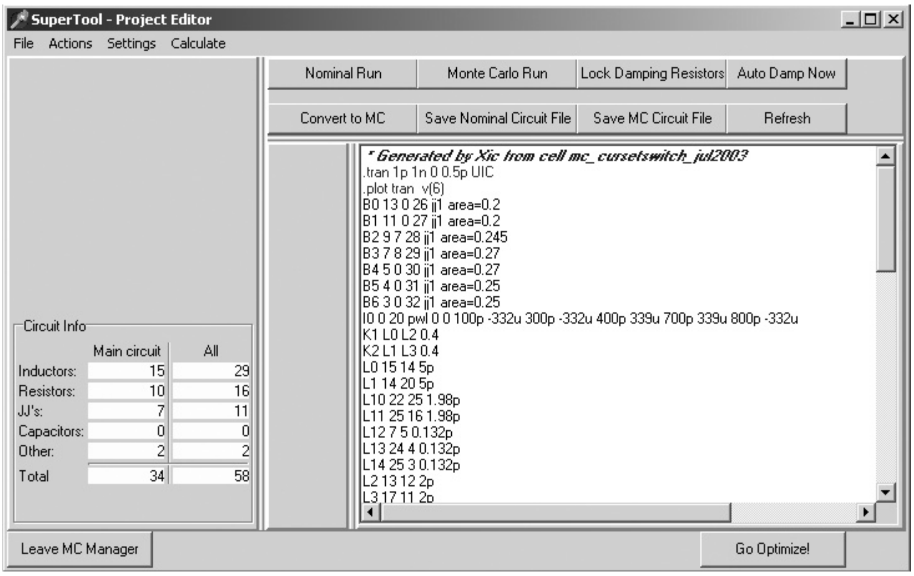
Figure C.8: Loading a new circuit into *SuperTool*

Figure C.8 shows the menu functions of *SuperTool*'s project editor, into which new circuits are loaded. In Figure C.9, the circuit file is loaded and ready for editing. The .plot statement lists all the output variables to be used for evaluating circuit performance. In this case, only the output voltage V_6 will be evaluated.

Some of the information and functions available to the user are also visible in Figure C.9.

The circuit information block on the left side contains the total number of elements in the *Spice* simulation file, with all subcircuit instances included, as well as the number of elements in the main circuit alone. This function was used to count the 2530 elements in the simulation model of the PFD programming circuit, as well as the 24147 elements in simulation model of the PFD core.

The other functions include one that automatically seeks out and links damping resistors to their respective Josephson junctions, as well as another that calculates and automatically sets the value of each damping resistor to obtain a required damping factor. The user can also convert the nominal circuit file to a Monte Carlo simulation file, perform a nominal or Monte Carlo simulation, and save the circuit as either a nominal or a Monte Carlo simulation file. Since project save files always include optimization settings and parameters, the last two functions are useful for creating pure *Spice* circuit files that can be used by other *Spice*-based simulation programs.



Circuit Info

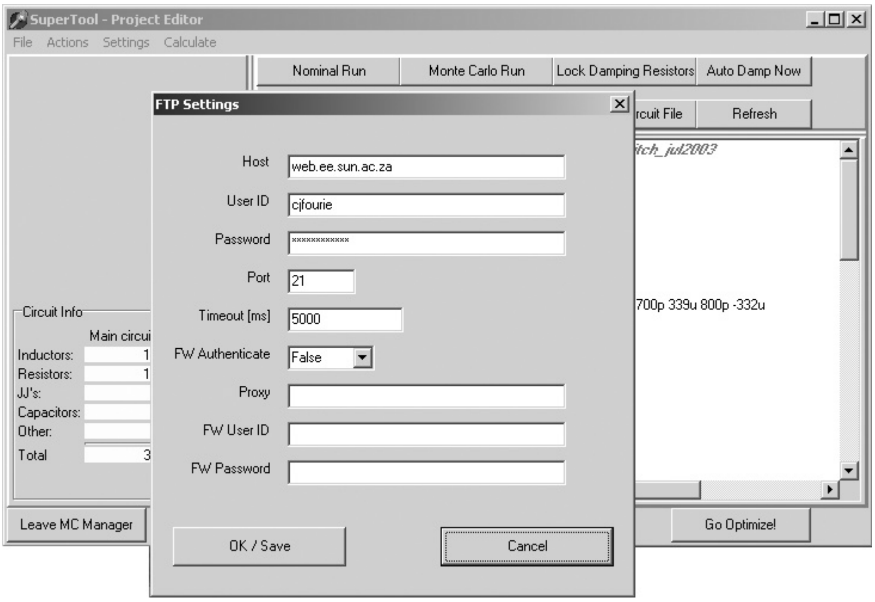
	Main circuit	All
Inductors:	15	29
Resistors:	10	16
JJ's:	7	11
Capacitors:	0	0
Other:	2	2
Total	34	58

```
* Generated by Xic from cell mc_cursetswitch_jul2003
.tran 1p 1n 0 0.5p UIC
.plot tran  v(6)
B0 13 0 26 jj1 area=0.2
B1 11 0 27 jj1 area=0.2
B2 9 7 28 jj1 area=0.245
B3 7 8 29 jj1 area=0.27
B4 5 0 30 jj1 area=0.27
B5 4 0 31 jj1 area=0.25
B6 3 0 32 jj1 area=0.25
I0 0 20 pwl 0 0 100p -332u 300p -332u 400p 339u 700p 339u 800p -332u
K1 L0 L2 0.4
K2 L1 L3 0.4
L0 15 14 5p
L1 14 20 5p
L10 22 25 1.98p
L11 25 16 1.98p
L12 7 5 0.132p
L13 24 4 0.132p
L14 25 3 0.132p
L2 13 12 2p
L3 17 11 2p
```

Figure C.9: *SuperTool* project editor with new circuit file loaded and ready for editing

SuperTool runs under Microsoft *Windows*, while *WRSpice* runs on a Unix platform. Interfacing is handled through Telnet and FTP routines. Figure C.10 and Figure C.11 show how the FTP and Telnet settings are configured.

The "Additional commands" box under the Telnet settings allows the user to specify extra commands that need to be executed every time that *SuperTool* connects to the *Spice* server via Telnet. These commands are most often used to force directory changes.



FTP Settings

Host: web.ee.sun.ac.za
User ID: cjfourie
Password: ************
Port: 21
Timeout [ms]: 5000
FW Authenticate: False
Proxy:
FW User ID:
FW Password:

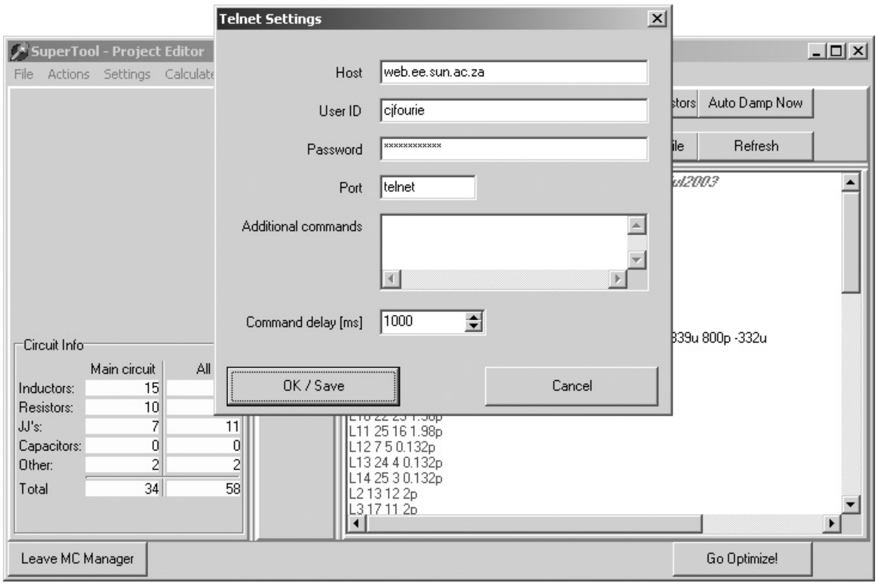
Circuit Info

	Main circui
Inductors:	1
Resistors:	1
JJ's:	
Capacitors:	
Other:	
Total	3

itch_jul2003

700p 339u 800p -332u

Figure C.10: Configuring the FTP settings for *SuperTool*



Telnet Settings

Host: web.ee.sun.ac.za
User ID: cjfourie
Password: ************
Port: telnet
Additional commands:
Command delay [ms]: 1000

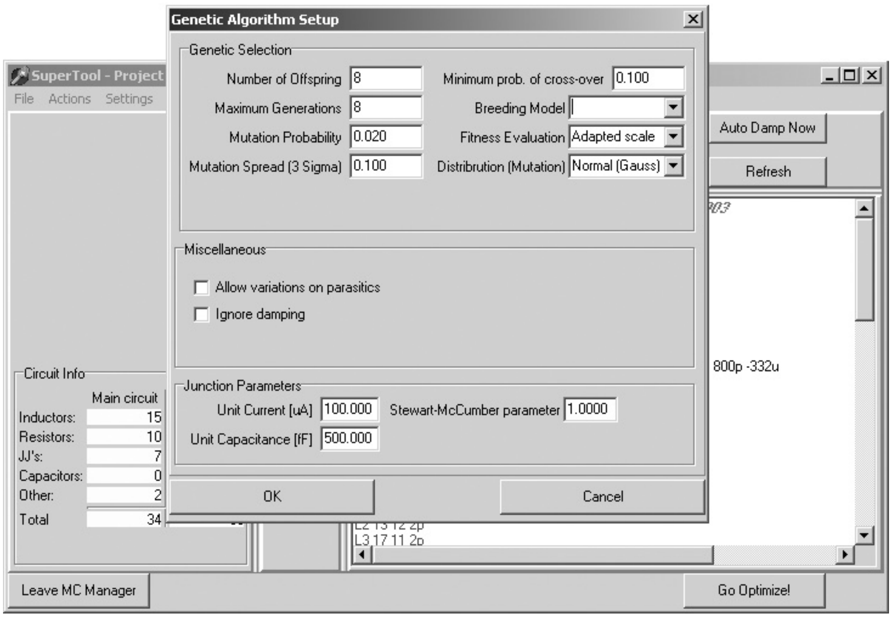
Circuit Info

	Main circuit	All
Inductors:	15	
Resistors:	10	
JJ's:	7	11
Capacitors:	0	0
Other:	2	2
Total	34	58

```
ul2003
339u 800p -332u
L10 22 23 1.98p
L11 25 16 1.98p
L12 7 5 0.132p
L13 24 4 0.132p
L14 25 3 0.132p
L2 13 12 2p
L3 17 11 2p
```

Figure C.11: Configuring the Telnet settings for *SuperTool*

Before optimization can commence, the optimization parameters need to be set. Figure C.12 shows a screenshot of the genetic algorithm setup window.



Genetic Algorithm Setup

Genetic Selection

Number of Offspring	8	Minimum prob. of cross-over	0.100
Maximum Generations	8	Breeding Model	
Mutation Probability	0.020	Fitness Evaluation	Adapted scale
Mutation Spread (3 Sigma)	0.100	Distribution (Mutation)	Normal (Gauss)

Miscellaneous

□ Allow variations on parasitics

□ Ignore damping

Junction Parameters

Unit Current [uA]	100.000	Stewart-McCumber parameter	1.0000
Unit Capacitance [fF]	500.000		

Circuit Info

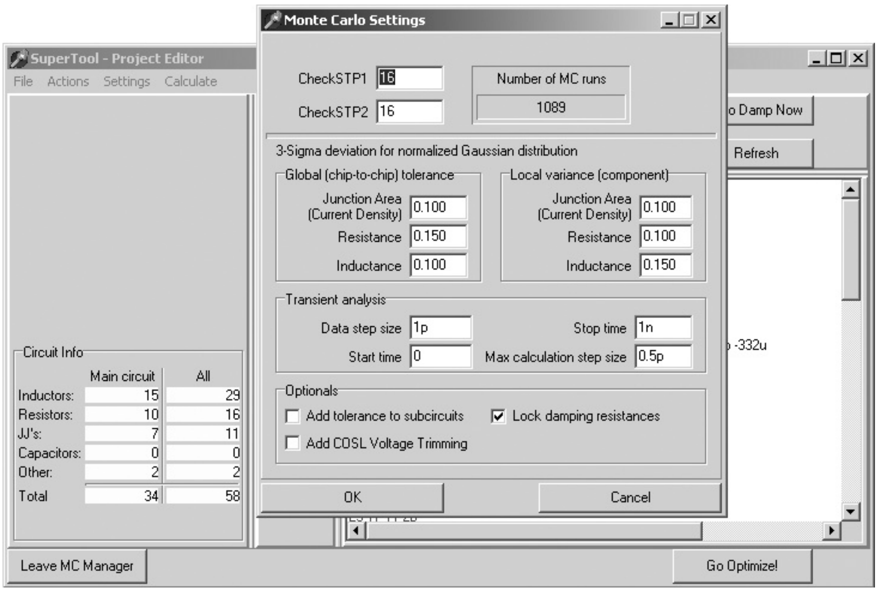
	Main circuit
Inductors:	15
Resistors:	10
JJ's:	7
Capacitors:	0
Other:	2
Total	34

Figure C.12: Genetic algorithm setup window

SuperTool also needs to know what parameter values to use when a nominal circuit file is converted to a Monte Carlo simulation file. Figure C.13 shows the Monte Carlo setup window. The transient analysis parameters are extracted from the active circuit file, but can be edited manually. In that case the new parameters are written over those in the active circuit file. When the active circuit file already is a Monte Carlo simulation file, all other parameters with the exception of "Optionals" are also extracted; else a predefined set of default values are loaded from *SuperTool*'s configuration files.

WRSpice calculates the total number of MC runs from

$$\text{Total runs} = (\mathit{CheckSTP}1 \times 2 + 1) \times (\mathit{CheckSTP}2 \times 2 + 1)\ . \tag{C.1}$$



Monte Carlo Settings

CheckSTP1	16	Number of MC runs
CheckSTP2	16	1089

3-Sigma deviation for normalized Gaussian distribution

Global (chip-to-chip) tolerance		Local variance (component)	
Junction Area (Current Density)	0.100	Junction Area (Current Density)	0.100
Resistance	0.150	Resistance	0.100
Inductance	0.100	Inductance	0.150

Transient analysis

Data step size	1p	Stop time	1n
Start time	0	Max calculation step size	0.5p

Optionals

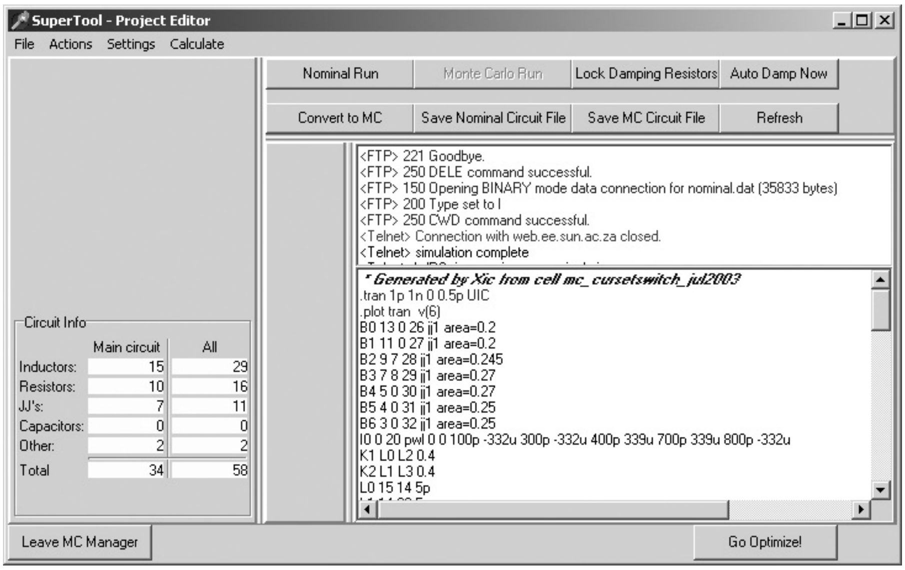
□ Add tolerance to subcircuits ☑ Lock damping resistances

□ Add COSL Voltage Trimming

Circuit Info

	Main circuit	All
Inductors:	15	29
Resistors:	10	16
JJ's:	7	11
Capacitors:	0	0
Other:	2	2
Total	34	58

Figure C.13: Monte Carlo simulation setup window



Circuit Info

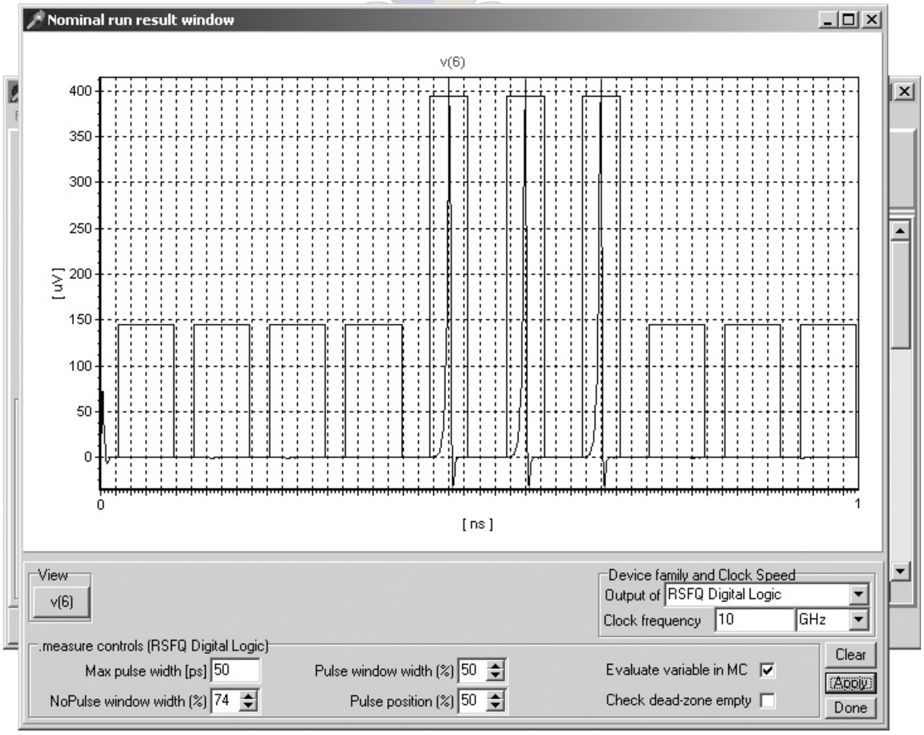
	Main circuit	All
Inductors:	15	29
Resistors:	10	16
JJ's:	7	11
Capacitors:	0	0
Other:	2	2
Total	34	58

```
<FTP> 221 Goodbye.
<FTP> 250 DELE command successful.
<FTP> 150 Opening BINARY mode data connection for nominal.dat (35833 bytes)
<FTP> 200 Type set to I
<FTP> 250 CWD command successful.
<Telnet> Connection with web.ee.sun.ac.za closed.
<Telnet> simulation complete
```

```
* Generated by Xic from cell mc_cursetswitch_jul2003
.tran 1p 1n 0 0.5p UIC
.plot tran  v(6)
B0 13 0 26 jj1 area=0.2
B1 11 0 27 jj1 area=0.2
B2 9 7 28 jj1 area=0.245
B3 7 8 29 jj1 area=0.27
B4 5 0 30 jj1 area=0.27
B5 4 0 31 jj1 area=0.25
B6 3 0 32 jj1 area=0.25
I0 0 20 pwl 0 0 100p -332u 300p -332u 400p 339u 700p 339u 800p -332u
K1 L0 L2 0.4
K2 L1 L3 0.4
L0 15 14 5p
```

Figure C.14: *SuperTool* running a remote simulation on the nominal circuit

Monte Carlo circuit analyses require evaluation criteria to determine whether or not a simulation was successful. Instead of defining these criteria by hand, a nominal simulation run can be used to allow *SuperTool* to propose the criteria. Figure C.14 shows the output when a nominal run was selected, with Telnet and FTP information scrolling in a window above the circuit file body. *SuperTool* automatically generates a nominal *Spice* input file containing a command that requests *WRSpice* to print the simulation output to a file, sends it via FTP to the remote *WRSpice* server, and initiates the simulation via a Telnet command. Upon completion of the simulation, the output file is downloaded via FTP to the local host, and automatically read into the graphic window shown in Figure C.15.



Device family and Clock Speed
Output of RSFQ Digital Logic
Clock frequency 10 GHz

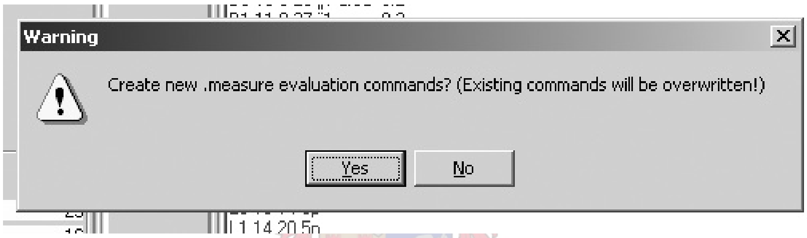
.measure controls (RSFQ Digital Logic)
Max pulse width [ps] 50
NoPulse window width (%) 74
Pulse window width (%) 50
Pulse position (%) 50
Evaluate variable in MC ☑
Check dead-zone empty □

Figure C.15: Nominal simulation output used to configure circuit evaluation criteria

Figure C.15 shows a graph of the output of a nominal simulation. The window can accommodate up to nine graphs. *SuperTool* can then automatically fit evaluation criteria to the data, and can do so for RSFQ, COSL or negative-output COSL voltage traces. It cannot yet handle current traces automatically, as they are not as well defined as the voltages.

In this example, only one output voltage trace is evaluated. The user sets the logic type (RSFQ) and clock frequency (10 GHz). The other parameters can also be changed, but are automatically loaded from configuration files when the graph window is opened. *SuperTool* then searches for every RSFQ pulse by way of numerical integration of the voltage trace with respect to time, and places an evaluation window around the peak of each. Evaluation windows for absent pulses are placed in every empty clock cycle, of which *SuperTool* calculates the temporal positions from the specified clock frequency and the location of known pulses.

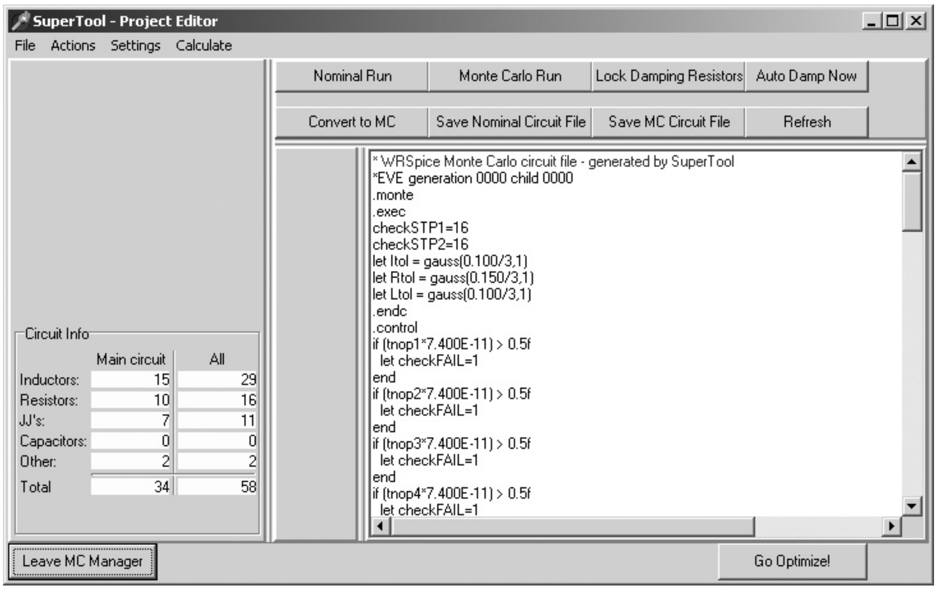
When the graph window is closed, *SuperTool* prompts the user for permission (Figure C.16) to create a new Monte Carlo simulation file with the evaluation criteria it calculated from the nominal simulation data. If the project file is already in Monte Carlo format, the old evaluation commands are replaced with new ones.



Create new .measure evaluation commands? (Existing commands will be overwritten!)

Figure C.16: Prompt for automatic conversion of current file to a new Monte Carlo simulation file upon termination of nominal output graph window

Figure C.17 shows the project editor window containing an automatically created Monte Carlo version of the original nominal circuit file. The user can still edit all the commands and parameters.



Circuit Info

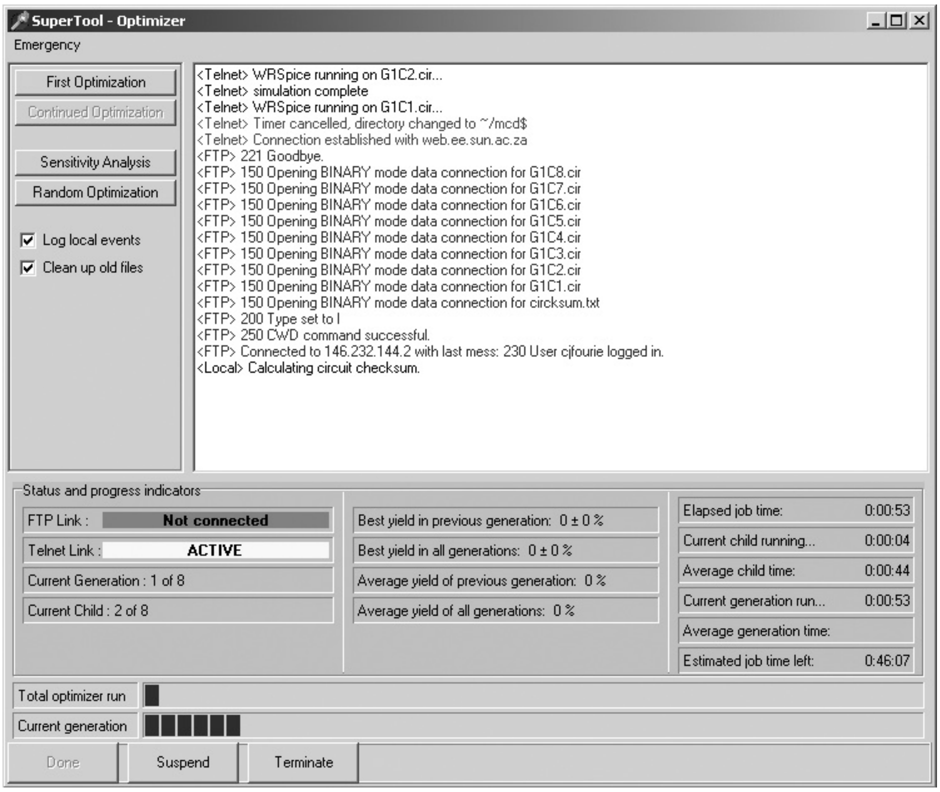
	Main circuit	All
Inductors:	15	29
Resistors:	10	16
JJ's:	7	11
Capacitors:	0	0
Other:	2	2
Total	34	58

```
* WRSpice Monte Carlo circuit file - generated by SuperTool
*EVE generation 0000 child 0000
.monte
.exec
checkSTP1=16
checkSTP2=16
let Itol = gauss(0.100/3,1)
let Rtol = gauss(0.150/3,1)
let Ltol = gauss(0.100/3,1)
.endc
.control
if (tnop1*7.400E-11) > 0.5f
  let checkFAIL=1
end
if (tnop2*7.400E-11) > 0.5f
  let checkFAIL=1
end
if (tnop3*7.400E-11) > 0.5f
  let checkFAIL=1
end
if (tnop4*7.400E-11) > 0.5f
  let checkFAIL=1
```

Figure C.17: *SuperTool* project editor window containing newly created Monte Carlo simulation file

Once the Monte Carlo parameters are added, the circuit can be subjected to optimization. If a previous optimization sequence was left unfinished, *SuperTool* will automatically compare the stored checksum file on the *Spice* server with one generated for the current project, and allow continued optimization if they are the same.

When an optimization sequence is started, random circuit files are generated in accordance with the optimization strategy, and these files sent via FTP to the *Spice* server. Telnet commands are used to command *WRSpice*. A screenshot of the feedback provided during the optimization process (which can run for weeks) is shown in Figure C.18.



☑ Log local events

☑ Clean up old files

```
<Telnet> WRSpice running on G1C2.cir...
<Telnet> simulation complete
<Telnet> WRSpice running on G1C1.cir...
<Telnet> Timer cancelled, directory changed to ~/mcd$
<Telnet> Connection established with web.ee.sun.ac.za
<FTP> 221 Goodbye.
<FTP> 150 Opening BINARY mode data connection for G1C8.cir
<FTP> 150 Opening BINARY mode data connection for G1C7.cir
<FTP> 150 Opening BINARY mode data connection for G1C6.cir
<FTP> 150 Opening BINARY mode data connection for G1C5.cir
<FTP> 150 Opening BINARY mode data connection for G1C4.cir
<FTP> 150 Opening BINARY mode data connection for G1C3.cir
<FTP> 150 Opening BINARY mode data connection for G1C2.cir
<FTP> 150 Opening BINARY mode data connection for G1C1.cir
<FTP> 150 Opening BINARY mode data connection for circksum.txt
<FTP> 200 Type set to I
<FTP> 250 CWD command successful.
<FTP> Connected to 146.232.144.2 with last mess: 230 User cjfourie logged in.
<Local> Calculating circuit checksum.
```

Status and progress indicators

FTP Link : Not connected

Telnet Link : ACTIVE

Current Generation : 1 of 8

Current Child : 2 of 8

Best yield in previous generation: 0 ± 0 %

Best yield in all generations: 0 ± 0 %

Average yield of previous generation: 0 %

Average yield of all generations: 0 %

Elapsed job time: 0:00:53

Current child running... 0:00:04

Average child time: 0:00:44

Current generation run... 0:00:53

Average generation time:

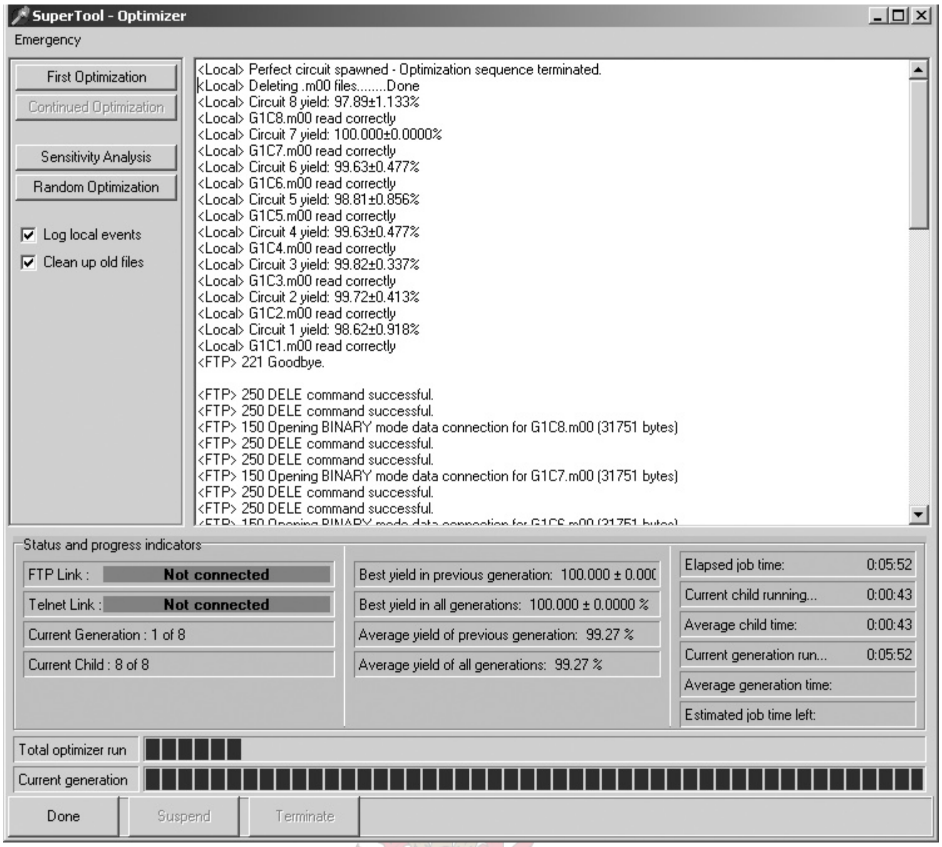
Estimated job time left: 0:46:07

Total optimizer run

Current generation

Figure C.18: Detail of information feedback during genetic optimization process

The optimization process will repeat for as many generations as specified in the optimization parameters, or will terminate when a circuit with a yield of 100 % is created. Figure C.19 shows the output when the latter event occurs (the example used for generating the screenshot operated on a previously optimized gate). After the completion of every generation, the simulation output files are retrieved from the *Spice* server via an FTP connection. The output files are then processed, and the yield for each child calculated automatically. If any child circuit has a yield that is better than any before it, it is copied to a file named “bestyet.cir”, which remains saved until a better circuit is found. This file also contains, in its text header, the numbers of the generation and child it originates from, as well as the yield percentage and uncertainty interval obtained from its Monte Carlo simulation.



☑ Log local events
☑ Clean up old files

```
<Local> Perfect circuit spawned - Optimization sequence terminated.
<Local> Deleting .m00 files........Done
<Local> Circuit 8 yield: 97.89±1.133%
<Local> G1C8.m00 read correctly
<Local> Circuit 7 yield: 100.000±0.0000%
<Local> G1C7.m00 read correctly
<Local> Circuit 6 yield: 99.63±0.477%
<Local> G1C6.m00 read correctly
<Local> Circuit 5 yield: 98.81±0.856%
<Local> G1C5.m00 read correctly
<Local> Circuit 4 yield: 99.63±0.477%
<Local> G1C4.m00 read correctly
<Local> Circuit 3 yield: 99.82±0.337%
<Local> G1C3.m00 read correctly
<Local> Circuit 2 yield: 99.72±0.413%
<Local> G1C2.m00 read correctly
<Local> Circuit 1 yield: 98.62±0.918%
<Local> G1C1.m00 read correctly
<FTP> 221 Goodbye.

<FTP> 250 DELE command successful.
<FTP> 250 DELE command successful.
<FTP> 150 Opening BINARY mode data connection for G1C8.m00 (31751 bytes)
<FTP> 250 DELE command successful.
<FTP> 250 DELE command successful.
<FTP> 150 Opening BINARY mode data connection for G1C7.m00 (31751 bytes)
<FTP> 250 DELE command successful.
<FTP> 250 DELE command successful.
<FTP> 150 Opening BINARY mode data connection for G1C6.m00 (31751 bytes)
```

Status and progress indicators

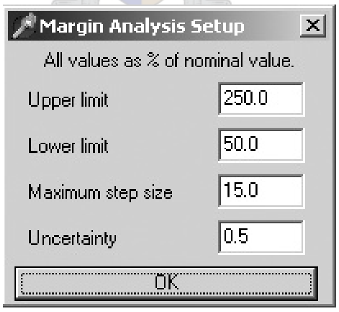
FTP Link : Not connected
Telnet Link : Not connected
Current Generation : 1 of 8
Current Child : 8 of 8

Best yield in previous generation: 100.000 ± 0.000
Best yield in all generations: 100.000 ± 0.0000 %
Average yield of previous generation: 99.27 %
Average yield of all generations: 99.27 %

Elapsed job time: 0:05:52
Current child running... 0:00:43
Average child time: 0:00:43
Current generation run... 0:05:52
Average generation time:
Estimated job time left:

Figure C.19: Detail of output after termination of genetic optimization process

SuperTool also supports margin analyses. It is stilled referred to as a sensitivity analysis in the optimizer window. Figure C.20 shows the parameter configuration window thrown up by *SuperTool* when the margin analysis routine is selected.

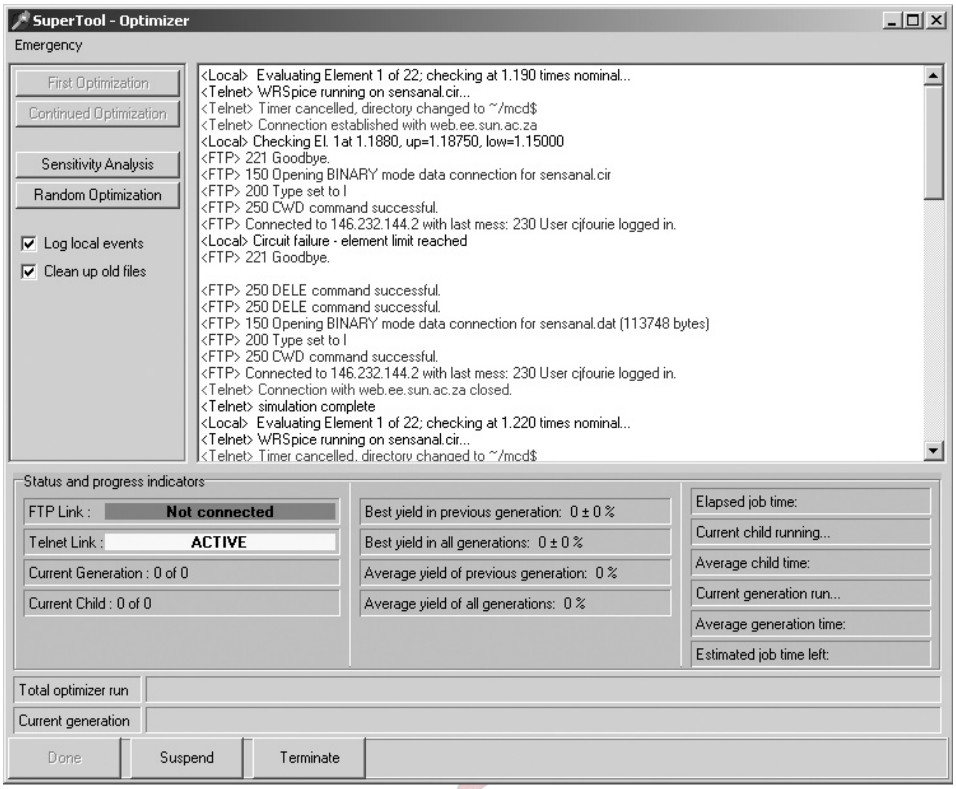


All values as % of nominal value.

Upper limit 250.0
Lower limit 50.0
Maximum step size 15.0
Uncertainty 0.5

Figure C.20: Parameter setup prompt upon selection of margin analysis routine

A screenshot of the feedback provided by *SuperTool* while a margin analysis is running, is shown in Figure C.21. Results are written into a text file that contains a matrix of element numbers and normalized upper and lower boundaries, and can easily be read into *Matlab*.



☑ Log local events
☑ Clean up old files

```
<Local>  Evaluating Element 1 of 22; checking at 1.190 times nominal...
<Telnet> WRSpice running on sensanal.cir...
<Telnet> Timer cancelled, directory changed to ~/mcd$
<Telnet> Connection established with web.ee.sun.ac.za
<Local> Checking El. 1at 1.1880, up=1.18750, low=1.15000
<FTP> 221 Goodbye.
<FTP> 150 Opening BINARY mode data connection for sensanal.cir
<FTP> 200 Type set to I
<FTP> 250 CWD command successful.
<FTP> Connected to 146.232.144.2 with last mess: 230 User cjfourie logged in.
<Local> Circuit failure - element limit reached
<FTP> 221 Goodbye.

<FTP> 250 DELE command successful.
<FTP> 250 DELE command successful.
<FTP> 150 Opening BINARY mode data connection for sensanal.dat (113748 bytes)
<FTP> 200 Type set to I
<FTP> 250 CWD command successful.
<FTP> Connected to 146.232.144.2 with last mess: 230 User cjfourie logged in.
<Telnet> Connection with web.ee.sun.ac.za closed.
<Telnet> simulation complete
<Local>  Evaluating Element 1 of 22; checking at 1.220 times nominal...
<Telnet> WRSpice running on sensanal.cir...
<Telnet> Timer cancelled, directory changed to ~/mcd$
```

Status and progress indicators

FTP Link : Not connected
Telnet Link : ACTIVE
Current Generation : 0 of 0
Current Child : 0 of 0

Best yield in previous generation: 0 ± 0 %
Best yield in all generations: 0 ± 0 %
Average yield of previous generation: 0 %
Average yield of all generations: 0 %

Elapsed job time:
Current child running...
Average child time:
Current generation run...
Average generation time:
Estimated job time left:

Total optimizer run
Current generation

Figure C.21: Detail of information feedback during margin analysis

C.3 PROJECT SAVE FILE HEADER

The project save file (with a .mcg extension) created by *SuperTool* is an ASCII text file that contains all the parameters used for optimization, as well as the entire circuit file in Monte Carlo simulation format.

The header of such a file is shown in this section. The main circuit body is excluded from this text body because a similar instance has already been shown in section B.1.3.

The data file referenced in the second line of the header contains all the elements, their values, suffixes and type designators (normal, parasitic, damping, etc.) for every child circuit in an optimization population. This data file is used by the optimizer, and updated every time a new generation is formed.

The breeding model is still stored as –1, since no specific model (such as elitism) is yet supported. Six fitness models (full or adapted scale, each linear, quadratic or cubic) are defined, and numbered from 0 to 5.

The other parameters are obvious, and Boolean values are stored as 0 (false) or 1 (true).

```
Coenrad J. Fourie - Optimization Project
DataFile=D:\phd programs\sup met tel toets\mcd\huffle_and_switch_mc.dat
# Genetic Algorithm Variables
Offspring=100
Generations=12
Mutation Probability=0.020
Mutation Spread=0.100
Breeding Model=-1
Fitness Model=4
Mutation Distribution=0
```

```
Minimum Cross-over Probability=0.100
Parameter Variance=0
Ignore Damping=1
# Junction Parameters
Unit Current[uA]=100.000
Unit Capacitance[fF]=500.000
Stewart-McCumber parameter=1.0000
# Monte Carlo Variables
CheckStop1=10
CheckStop2=10
Global Junction Area Tolerance=0.100
Global Resistance Tolerance=0.200
Global Inductance Tolerance=0.100
Local Junction Area Tolerance=0.100
Local Resistance Tolerance=0.050
Local Inductance Tolerance=0.150
COSL Voltage Trim=0
Subcircuit Tolerance=0
Add Evaluation Code=1
Deadzone Checking=0
Lock Damping Resistors=1
Main Circuit File Lines=189
# Main Circuit
```

Appendix D – Full set of 3D inter-junction inductance models

ALL the 3D models used for inductance calculation or verification during the layout of the masks shown in Appendix A, with the exception of those already shown in Chapter 3 and Chapter 7, are depicted here. For display purposes all the vertical dimensions are stretched.

All models can be adapted by having their dimensions altered.

We shall first discuss the models used in Chapter 3. The straight interconnection shown in Figure 3.19(b) was built for simulating JTLs. It was also used for all other straight connections in M2, and when shortened (and the dc tee-in omitted) was used for calculating the inductance in one arm of connections containing a series junction.

The model in Figure 3.21(b) was developed for the pulse splitter. With some parts of the geometry cut away, it can also model double-cornered interconnection lines (although none were used in the layout of the PFD).

The series junction model in Figure 3.23 was developed to model the reset input section of the HUFFLE.

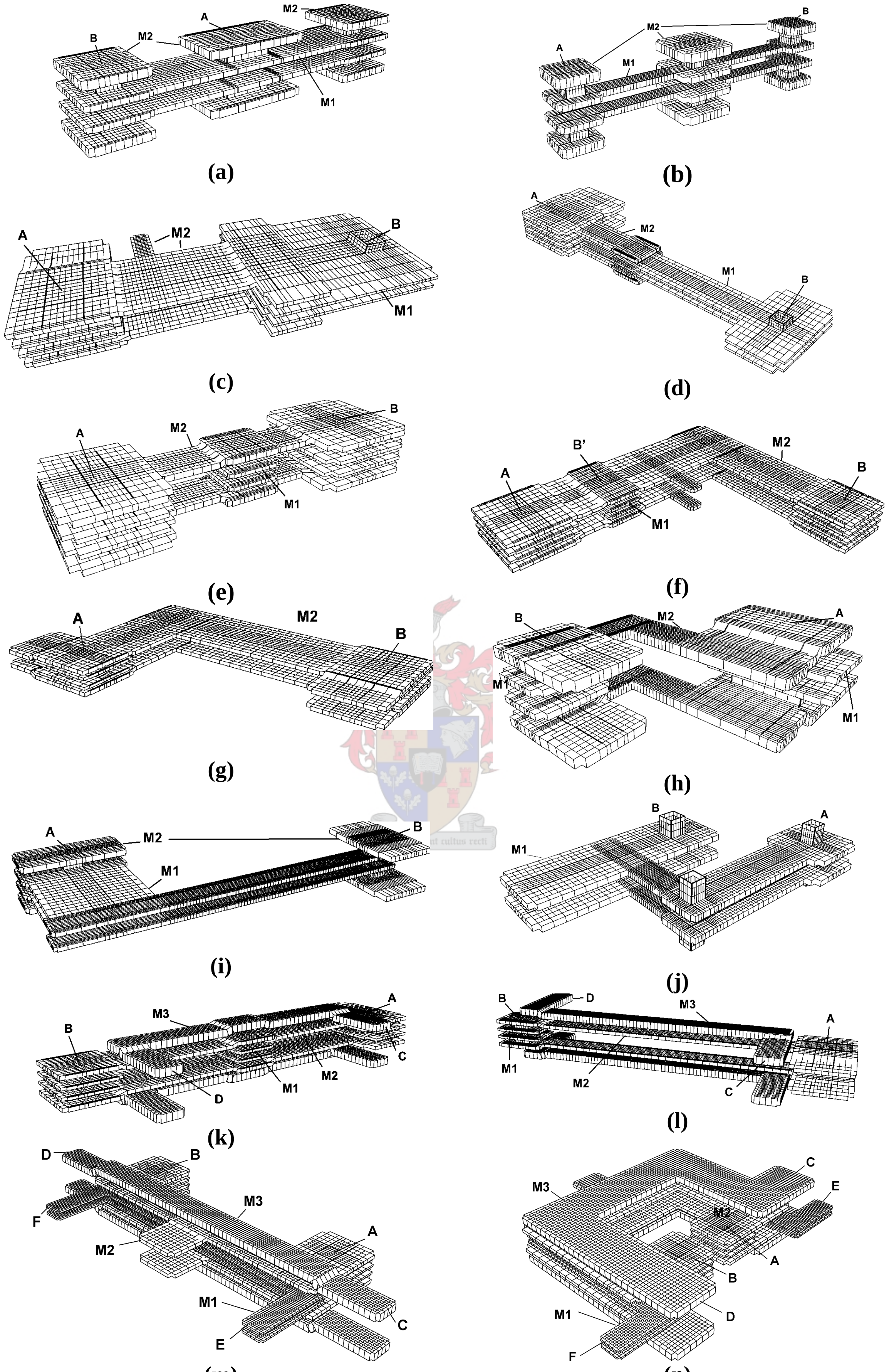
Figure 3.25(c) shows the damping resistor model. It was first used in the 250 mA JTL, and later (with increased length) for modelling the large damping resistors in the HUFFLE.

The U-bended JTL was modelled with the structure shown in Figure 3.26(a).

Figure 3.27(a) shows the model used for calculating self- and mutual inductance in dc SQUID loops and control lines in the COSL SRFF and RSFQ-to-COSL converter.

The other 3D models used in this dissertation are shown in Figure D.1. Of these, (a), (f) and (i) were first used in the T1 flip-flop, (b), (c) and (e) in the SFQ-to-DC converter, (d) and (l) in the DCRL, (g) for PFD interconnects, (h) in the pulse merger, (j) in the DC-to-SFQ converter, (k) in the Current-Set switch and (m) and (n) in the HUFFLE.

The structures in Figure D.1(a) to (j) were developed to model self-inductance between nodes A and B. Those in Figure D.1(k) and (l) model self-inductance for the conductors between nodes A-B and C-D, and also mutual inductance between the two conductors. The three-conductor structures in Figure D.1(m) and (n) model self-inductance for the conductors A-B, C-D and E-F, and the mutual inductance between every pair of conductors.



(a) (b)

(c) (d)

(e) (f)

(g) (h)

(i) (j)

(k) (l)

(m) (n)

Figure D.1: Array of 3D structures used for inductance calculation