MANUAL
to handle JSIM and optimizer program AUTO5 from University of Technology Ilmenau

Version 7.0 - April 2011

written at University of Savoie – France
in the framework of the European FLUXONICS Initiative

****

Manual to handle JSIM and optimizer program AUTO5 developed at the University of Technology of Ilmenau
Manual written at University of Savoie - France in the framework of the European FLUXONICS Initiative

****

File ****** completed by Pascal Febvre on ****, 2010.
Modified by Benedicte NDENDE on ****, 2011.

TABLE OF CONTENTS

I-Procedure to use simulator and optimizer

The procedure to run JSIM simulator and AUTO5 optimizer is based on a set of programs and files. Each of them is described separately in the manual below. The sequence that is needed to do simulations is the following.

I-1 Time-domain simulations:

- create a directory with a name that is the one of the cell under study. Here, we will take the dcsfq cell as an example. All files mentioned below should be in this same directory. testjsim is the directory name in the case of the examples of this manual.

- create the netlist of the cell with extension *.js* (for JSIM): this netlist can have some parameters that are variables. These parameters are letters such as *@A* or *@b* in *.js* netlist file. testjsim.js is the netlist file example of the dcsfq cell. By convention, if there is no parameter in the netlist but only numerical values, the filename is **xxx-t.js**. Example **:** testjsim-t.js

- create the file with the netlist parameters to be inserted in *.js* netlist file. This file also indicates the range of variations of the netlist parameters. This file has the *.para* extension. The name of this file is testjsim.para

- create the file that indicates the directories used to store simulation and optimization files. This file also gives some criteria for digital tests and to plot results with GRACE. This file has the .conf extension. For the dcsfq cell taken as an example the name of the file is: testjsim.conf.

- The two files marginout1.tagr and marginout2.tagr should be present in the main cell directory: they are needed to draw the margins with GRACE.

- Two cases can occur:
  - if the netlist *.js* file has no parameter but only numerical values, you can directly run JSIM with the command : **jsim_n netlist_file > output_file** . Example **: jsim_n testjsim-t.js > testjsim.dat** .

    Click on testjsim.dat to see the result of this command. The output file is not ready to be drawn with GRACE since the netlist is placed as a header before the results of simulation and the statistics of computation are included as a footer. See the next paragraph to see how to display results.
    Note : it is possible to generate automatically the data file without header and footer by inserting the command **.file output_data_file** before the **.print**

commands in the netlist *.js* file. In this case, for instance the command **jsim_n testjsim-t.js > testjsim.dat** will generate the header (netlist) and footer (simulation statistics) without calculation results in *testjsim.dat* file. In this case the results of simulations are in the *output_data_file*.

- If the netlist *.js* file has some parameters, they need to be replaced by their numerical values. This is done with the command : **auto5 netlist_file 2 1 0**. Example : **auto5 testjsim 2 1 0 .**

  The **AUTO5** program is described later in the manual. Briefly the first parameter (2 in this case) means that AUTO5 will do an optimization. The second parameter (1 in this case) means that there is only one attempt which is the first simulation with JSIM (consequently there is no optimization). The third parameter is not used. In the example, the output netlist file with parameters replaced by their numerical values is in */tmp* directory and should be named *testjsim-tmp.js*. The default name and path are defined in testjsim.conf file: see the filename connected to *simcir* variable in *.conf* file. Then the file can be copied back in the main cell directory.

  Example **: cp /tmp/testjsim-tmp.js testjsim-t.js.** Usually these two steps are directly accomplished with the command file named ***A*** that is present in the main cell directory.

  Note: It is possible that you meet some errors when running the command

  auto5 testjsim 2 1 0 because auto5 calls executable programs like *qrun*. In that case, you should verify the rights of all your programs. Make sure that they are able to be executed. If not, enter the command **chmod 777 filename**.

  Example: **chmod 777 qrun.** See the description of the qrun program below.

  Note: the */tmp* directory is used because simulations are faster in RAM. Note also that the **AUTO5** program generates three other files in the */tmp* directory : a test file with extension *.test* not used, and two files with extensions *.dat* and *.ext* . Their names and paths are specified in the *.conf* file (variables *simtest*, *simout* and *extout*, respectively). The file with *.dat* extension (*testjsim.dat* for instance) is the same as the one calculated by JSIM. Again, this file has a header and a footer and is not ready to be drawn with GRACE. See the next paragraph to learn how to display results. The second file with extension *.ext* (check content of *testjsim.ext* for instance) gives the digital state of the circuits at some particular instants, specified in the *.conf* file (variable timestep). See the description of **EXT** program below. Once this first step is achieved with the netlist that includes the numerical values of the parameters, several actions can be done. The simulated output file can just be displayed with GRACE and/or the cell can be optimized and/or simulated for margins or yield. This is accomplished with either the CUTT or the EXT program.

I-2 Display of time-domain results:

By default (unless you use the *.file output_data_file* command before the *.print* commands in the netlist file), JSIM output file with *.dat* extension has a header and a footer that prevent easy automatic plotting of the data. These can be removed with the **CUTT** program with the command : **cutt filename** . Note that the extension *.dat* should NOT be inserted in the filename. Also, in the example, the *.dat* file should be placed in the /tmp directory, as indicated in the *simout* variable of the *.conf* file. Example **: cutt testjsim**. The resulting file with extension *.ext* is also put by default in the /tmp directory in the example (indicated in the *extout* variable of the *.conf* file) and need to be copied back to the cell directory with, for instance : **cp /tmp/testjsim.ext testjsim_signal.ext**

The steps including the replacement of the variables in the netlist file, the simulation of the netlist with JSIM, the removal of the header and footer of the output file and the plot of results on GRACE are usually operated by batch with the command file named ***t*** that is present in the main cell directory. This command file also usually calculates the digital state of the circuit with program EXT, from the same JSIM *.dat* output file. [To launch simulations remotely on MUST server, use ***outt*** to send files and commands to MUST and ***int*** to get back output files after simulation and display them with GRACE.]

I-3 Margins Calculations:

Before doing the calculations of margins, you need to tell the optimizer what are the criteria that are used to state that the cell works properly. To do that, you need to find at least one point of operation with the right nominal values of all the parameters. You can figure out if it works properly by displaying phases and voltages with time-domain simulations, as mentioned above. For instance, if you know that junction n has switched once at time t_1 and twice at time t_2, then its phase should be around 2□ and 4□ respectively for these two instants. Once simulations are performed you need to use the EXT program, which works like the CUTT program (example: **ext testjsim**) but calculates, from the results of JSIM simulations, the normalized phases and voltages for every instant you are interested in (t_1 and t_2 here). Output results are put in a file with *.ext* extension (example: *testjsim.ext*) The phases are normalized to 2□ then rounded to the closest integer, while the voltage is replaced by 1 if it is higher than variable *minonevoltage* (expressed in µV) of the *.conf* file, by 0 if the voltage is lower than variable *maxzerovoltage* and by -5 when the voltage is comprised between *maxzerovoltage* and *minonevoltage*. In this last case, the results say that there is an error since one should not check the voltages (and also phases) when a pulse is passing (or when a junction is switching). The *.ext* file gives, for each instant, the normalized values of all the *.print* commands. **Be careful** to include only phases in *.print* commands of the netlist file because other parameters will be misunderstood by the EXT program. You are only allowed to use a voltage or a current for the last *.print* command in the netlist. This corresponds to the right column of the *.ext* file. If the last parameter you want to display is a phase then you should set *voltageswitch* variable of *.conf* file to 0. If it is a current or a voltage, you need to set *switchvoltage* to 1, so that the EXT program normalizes properly each output phase or

voltage.
Note : The digital tests of voltages is usually used for SFQDC cells.

The times cannot be freely chosen unfortunately, they are necessarily taken during the simulation duration (given by *.tran* command), starting at a value given by the *timestep* variable mentioned in the *.conf* file and with a periodicity of *timestep* as well. Example : if *timestep* is 15 (15 ps) for a total simulation time of 50 ps, the EXT program will check and normalize phases at 15, 30 and 45 ps. Once you are satisfied with the results, the digital state is in file with extension *.ext*. You then need to copy the *.ext* file in a reference file for digital criteria which has the extension *.soll*.
Example : **cp testjsim.ext testjsim.soll**. This testjsim.*soll* file is now your reference for margins and yield calculations.

Margins calculations are done with the AUTO5 program. The syntax is **auto5 testjsim 1 N** where N is the number of iterations. Each parameter of the netlist, whose values are in *.para* file, is used as a nominal value for margins calculations. The corresponding normalized value of this parameter is 1. For each parameter, the optimizer AUTO5 will check the range of values that make the circuit work properly. The normalized range for the search goes from 0.1 to 3. The first iteration picks up a value of the parameter midway between 1 and 3 (2 in this case). It performs JSIM calculations and compares the digital results of *.ext* file with the reference file *.soll*. If results are the same, test is passed and the optimizer continues by picking a value midway between 2 and 3, etc… Otherwise it verifies midway between 1 and 2. Then the same procedure is done for lower values than the nominal normalized parameter, between 0.1 and 1. The total number of iterations is given by N. Consequently the final accuracy is $1/2^N$. All this is done for each parameter indicated in *.para* file, assuming that all the other parameters are at their nominal values. Two files are generated at the end of the process : one with extension *.mar* which has the values of the margins, and a GRACE *.agr* file which allows to directly draw the results with GRACE.

Usually the margins calculations and automatic display with GRACE are directly run with the command file named b that is present in the main cell directory. [To launch simulations remotely on MUST server, use ***outb*** to send commands to MUST and ***inb*** to get back output files after margins calculations and display them with GRACE.]

I-4 Optimization of a cell:

This is done with the AUTO5 program as well. It is assumed that you have already one operation point that works with nominal parameters, and that the digital criteria are already set with the *.soll* file ready, as mentioned in the paragraph about margins calculations. The command to use to optimize the cell, that means : increase the margins and center them around a new nominal point of operation, is as follows : **auto5 filename 2 N x** where filename is the cell name without any extension (example : **auto5 testjsim 2 N x**). At the end of the optimization process, files *filename testjsim .mar* and *filename testjsim .agr* are created. The optimization of AUTO5 is based on the «center of gravity » method : the program selects a « box » whose size is ± x% of the value of the nominal parameter and centered on it. Then, with a Monte-Carlo method, it picks up randomly a value of the parameter inside this box *N* times and checks if the circuit works or fails. Then it calculates the center of gravity for the values of the parameter corresponding to the working circuits and finally gives the new optimized value. Of course this process is done by choosing randomly all parameters that have to be adjsuted, as mentioned in the *.para* file. It gives, as a display, the total number of working circuits among the *N* attempts, as well as the pourcentage of working circuits. You then need to manually replace the new optimized values put in the *filename.mar* file in the corresponding *.para* file. It is better to recalculate margins after this, to check that the optimization went right.

Note: When you execute all those commands (auto5 testjsim 2 1 0, cutt testjsim, ext testjsim)
You should be in the main cell directory.

I-5 Yields Calculations:

To calculate yields, the AUTO5 program selects randomly a set of parameters that are inserted in the netlist file and then simulates the corresponding circuit. The command syntax is **auto5 filename 3 50 20 4**. Note that the name of the file here has no extension.

Example auto5 testjsim 3 50 20 4

3: is to tell that AUTO5 will calculate yields.
50: defines the number of iterations that will be done by AUTO5.
20: is expressed in percentage. It is the maximum standard deviation well known by sigma of a Gaussian law.
4: represents the number of steps to go up to 20%.
In this case for example, it will calculate yields with 4 values of sigma. The first one is for sigma = 5%, then sigma = 10%, 15% and finally 20%. At the end, there are 4 * 50 = 200calculations.
After doing those calculations, AUTO5 places the yield data in a resulting file which has the .yi extension and present in the main cell directory. Click on testjsim.yi to see the file. That file is ready to be displayed with GRACE.
Usually yields calculations are directly run with the command file named d that is present in the main cell directory.

II-JSIM simulator

Syntax of the JSIM netlist file (extension .js)

jsim [-options] [filenames] [-options] [filenames] ... [redirection]

Recognized options are d, r, or nothing, following a '-'.

d	toggle debugging, which dumps a file jsim.dbg (initially off).
r	toggle use of a rawfile jsim.raw for output (initially off).
-	read the standard input.

The d,r options can be combined. The filenames jsim.dbg and jsim.raw are hard coded.

Other tokens on the command line are assumed to be file names to read for input. If no such tokens are found, the standard input is read.

The options operate on files listed to the right of the option list, and are active until changed with another invocation. If no files are listed, the options are read before the standard input is read.

When multiple files are simulated, enabled output to the jsim.dbg and jsim.raw files is appended. The first file to generate such output creates the file. If the file previously existed, it will be clobbered! If the rawfile option is not specified, files will be created in accordance with the .file lines in the input files. These file names should be unique, as new files are opened for each simulation. The .file lines are ignored if the rawfile option is used.

examples:

```
jsim file1 > file2
```

will simulate file 1 and put output results in file 2.

```
jsim file1 -d file2 -d file3
```

will dump debug information while simulating file2.

```
jsim file1 -rd - file3 <anotherfile
```

will simulate file1, turn on debugging and rawfile creation, then simulate anotherfile, then file3.

```
jsim -r <inputfile
```
will simulate inputfile and create a rawfile.

```
jsim -r inputfile
```
same as above

```
jsim - - - <input
```
will run three concatenated input decks read from input.

II-1 JSIM Preliminary Version User's Guide

E. S. Fang and T. Van Duzer
Department of Electrical Engineering and Computer Science
University of California
Berkeley, CA 94720

1. Introduction

JSIM (Josephson SIMulator) is a circuit simulation program for Josephson circuits. Circuits may contain resistors, capacitors, inductors, mutual inductors, independent voltage and current sources, lossless transmission lines and Josephson junctions. At present time, only transient analysis is allowed.

The input format of JSIM is quite similar to SPICE. If you are not familiar with SPICE, it is recommended that you read the SPICE user guide also.

2. Circuit Description

2.1. Resistors

General form : RXXXX N1 N2 VALUE

Example : RC1 12 9 1KOHM

2.2. Capacitors

General form : CXXXX N1 N2 VALUE <IC=VALUE>

Example : C1 10 11 1PF

Initial value may be specified, but currently it
is ignored. This applies to ALL initial values.

2.3. Inductors

General form : LXXXX N1 N2 VALUE <FCHECK> <IC=VALUE>

Example : L2 1 0 2.3PH FCHECK

If FCHECK (flux check) is specified, JSIM will
keep track of the change of flux through the
inductor. It is recommended particularly for
phase mode circuit. In each superconductive loop,
at least one inductor should be considered for
FCHECK option.

2.4. Mutual Inductors

General form : KXXXX LXXXX LYYYY VALUE

Example : K1 L1 L2 0.9

2.5. Independent Voltage Sources

2.5.1. Sinusoidal Sources

General form : VXXXX N1 N2 SIN(VO VA FREQ TD THETA)

Example : V1 1 0 SIN(0 1MV 100MEGHZ 0US 0)

Note VO must be zero.

2.5.2. Pulse Sources

General form : VXXXX N1 N2 PULSE(V1 V2 TD TR TF PW
PER)

Example : V2 2 0 PULSE(0MV 1MV 0PS 2PS 2PS 10PS 50PS)

Note V1 must be zero.

2.5.3. Piece-wise Linear Sources

General form : VXXXX N1 N2 PWL(T0 V0 T1 V1 ....)

Example : V3 3 0 PWL(0PS 0MV 1PS 1MV)

Note T0 and V0 must be zero.

2.6. Independent Current Sources

2.6.1. Sinusoidal Sources

General form : IXXXX N1 N2 SIN(IO IA FREQ TD THETA)

Example : I1 1 0 SIN(0 1MA 100MEGHZ 0US 0)

Note IO must be zero.

2.6.2. Pulse Sources

General form : IXXXX N1 N2 PULSE(I1 I2 TD TR TF PW PER)

Example : I2 2 0 PULSE(0MA 1MA 0PS 2PS 2PS 10PS 50PS)

Note I1 must be zero.

2.6.3. Piece-wise Linear Sources

General form : IXXXX N1 N2 PWL(T0 I0 T1 I1 ....)

Example : I3 3 0 PWL(0PS 0MA 1PS 1MA)

Note T0 and I0 must be zero.

2.7. Josephson Junctions

General form : BXXXX N1 N2 MODNAME <AREA> <CONDEV=DEVNAME> <IC=V0,PHI0>

Example : B1 2 3 JJMOD1 AREA=1.5 CONDEV=L2

MODNAME is the model name, CONDEV is used to simulate the modulation of critical current due to magnetic field, DEVNAME can only be inductors, voltage and current sources.

2.8. Transmission Line

General form : TXXXX N1 N2 N3 N4 LOSSLESS <Z0=VALUE> <TD=VALUE>

Example : T1 1 0 2 0 LOSSLESS Z0=50 TD=100PS

N1 and N2 are nodes for port 1, and N3 and N4 are
nodes for port 2. Default Z0=50ohm, TD=1sec.

2.9. Subcircuit Calls

General form : XYYYY SUBDEFNAME N1 N2 .....

Example : X1 SGA 1 3 4 5

3. Subcircuit Definition

General form : .SUBCKT SUBDEFNAME N1 N2 ......

General form : <circuit elements>

General form : .ENDS

Example : .SUBCKT TEST 1 2 3

Example : R1 1 2 3K

Example : C1 2 3 3PF

Example : .ENDS

4. Model Specification

4.1. Josephson Models

General form : .MODEL MNAME JJ(<PARAM=VALUE>,....)

Example : .MODEL JJMOD1 JJ(VG=2.5MV, CAP=0.6PF, ICRIT=100UA)

RTYPE : quasiparticle model, can be zero or one.
Zero is for zero conductance, and one is for

piece-wise linear conductance curve, default is 0.

CCT : control current type, can be zero or one. Zero is for no control current, and one is for sine x over x, default is 0.

VG : gap voltage, default is 2.8mV.

DELV : gap transition voltage, default is 0.1mV.

ICON : control current scale, default is 1mA.

R0 : subgap resistance, default is 30ohm.

RN : normal resistance, default is 5ohm.

CAP : junction capacitance, default is 2.5pf

ICRIT : critical current, default is 1mA.

5. Transient Analysis Specification

General form : .TRAN PRSTEP TSTOP <TSTART> <MAXTSTEP>

Example : .TRAN 1PS 100PS 20PS 0.5PS

PRSTEP is the printing step.

TSTOP is the stop time.

TSTART is the starting time for printing, default is 0.

MAXTSTEP is the maximum internal time step, default is 1ps.

6. Print and File Specifications

General form : .FILE FILENAME

General form : .PRINT PRTYPE PRNAME <PART>

Example : .FILE OUT1

Example : .PRINT NODEV 2 0

Example : .PRINT DEVV X1_X2_B1

Example : .FILE OUT2

Example : .PRINT DEVI B2 JJTOTAL

Example : .PRINT PHASE B3

Example : .PRINT DEVV T1 PORT1

Example : .PRINT DEVI T1

To print out device voltage or current of a sub-circuit element, just expand the element name by the subcircuit call name. In the above examples, X1_X2_B1, means B1 belong to subcircuit call X2 which is in turn called by X1. This eliminates the need for a long node list in the subcircuit definition in order to print out values associated with subcircuit elements as in SPICE.

JJTOTAL : total current.

JJJOSEPH : Josephson current.

JJCAP : current due to capacitive effect.

JJRESIS : quasi-partical current.

JJALL : print all currents in the order specified.

PORT1 : print port1 of transmission line.

PORT2 : print port2 of transmission line, no port specification prints both ports.

7. Option Specifications

General form : .OPTIONS <PARAM, ...>

Example : .OPTIONS RELTOL=0.01 MAXPHISTEP=1.5

RELTOL=VALUE : relative tolerance, default is 0.001.

PHITOL=VALUE : absolute tolerance for phase, default is 0.0001, (too small, set it to 0.01).

VNTOL=VALUE : absolute voltage tolerance, default is 0.1 uV, only in effect when LTE option is used.

INTOL=VALUE : absolute current tolerance, default is 0.1 uA, only in effect when LTE option is used.

MAXPHISTEP=VALUE : maximum phase change allowed in one time step, default is 1.5.

MAXFLUXSTEP=VALUE : maximum flux change in inductor allowed in one time step. Only applies to those inductors with FCHECK flag specified. Default is 0.5e-15.

LTE : check for local truncation error in choosing step size, default no LTE.

IGWARN : ignore warning and proceed with simulation.

NUMDGT=VALUE : number of digits to be printed, default is 3.

8. Special Files

JSIM will generate some special files. All the special files start with .jsim. They are devlist, devname, model, nodemap and subdef.

9. CAUTION

This is a preliminary version of the program. Many parameters have no defaults, and have to be specified.

10. BUGS

If you find any problems, please send e-mail to esfang@argon.berkeley.edu, include the input deck that causes the problem and a description of the problem. Also you may call (415) 642-0502 and contact Emerson Fang, or write to Prof. Ted Van Duzer at the above address.

II-2 Noise supplement JSIM Preliminary Version User's Guide

J S Satchell
DRA(Malvern)
St. Andrews Rd.,
Worcs, UK
WR14 3PS

1. Introduction

JSIM (Josephson SIMulator) is a circuit simulation program for Josephson circuits. The noise extension adds two new device types, and is intended to be a strict superset of standard JSIM, documented in the file manual.ms.

Modifications, makefile, awk script, this document and the contents of the test directory are

2. Circuit Description

2.1. Independent Voltage Sources

2.1.1. Noise Sources

General form : VXXXX N1 N2 NOISE(VO VA TSTEP TD)

Example : V1 1 0 NOISE(0 1P 1PS 0PS)

Note VO must be zero. VA is the spectral amplitude density of the noise source in Volts/root(Hz). TSTEP is a the maximum time step that will be used, this provides an opportunity to limit ther-rors in the stochastic algorithm. 1pS seems to be reasonable for HTS junction parameters. TD is a time delay before noise starts.

2.2. Independent Current Sources

2.2.1. Noise Sources

General form : IXXXX N1 N2 NOISE(IO IA TSTEP TD)

Example : I1 1 0 NOISE(0 1P 1PS 0PS)

Note IO must be zero. IA is the spectral amplitude density of the noise source in Amps/root(Hz). TSTEP is a the maximum time step that will be used, this provides an opportunity to limit the errors in the stochastic algorithm. 1pS seems to be reasonable for HTS junction parameters. TD is a time delay before noise starts.

BUGS

If you find any problems with the stochastic extension, please send e-mail to satchell@dra.hmg.gb, include the input deck that causes the problem and a description of the problem. Warning: I may ignore you, be too busy or be unable to help.

III-Executable programs on the software JSIM

AUTO5

AUTO5 has been developed at the University of Technology of Ilmenau by Thomas Ortlepp. This program is used to calculate margins of SFQ cells, to optimize them or to calculate their yield. AUTO5 calls the QRUN program at each iteration.

Syntax : AUTO5 *filename N1 N2 [N3] [N4]*
filename is the filename without extension of the cell netlist to be analyzed by AUTO5

N1 is : 1 for margin calculations
2 for optimization with a Monte-Carlo method
3 for yield calculations
N2 is the number of binary steps used for margin calculations (if N1 = 1)

Examples : AUTO5 testjsim 1 10 calculates the margins of the DCSFQ cell with ten binary steps.

Remark: By default, this program works with files that are present in the /tmp directory. Results are also in the same directory (see *testjsim.conf* file).

CUTT

CUTT has been developed at the University of Technology of Ilmenau by Thomas Ortlepp. This program is used to remove header and footer of the output file processed by JSIM.

Syntax : CUTT *filename*

filename is the filename without extension corresponding to *filename.dat* output file processed by JSIM.

Examples : CUTT testjsim

Input file *: testjsim.dat* – the file path and extension of this file are given in *testjsim.conf* file
Output file : *testjsim.ext* – the file path and extension of this file are given in *testjsim.conf* file

EXT

The EXT program acts like the CUTT program. It is required to normalize each output phase and voltage values for times specified in the *timestep* value in the *testjsim.conf* file.

Syntax : EXT *filename*

Example ext testjsim.

The results of this command is that a file name *testjsim.ext* is created phases are normalized to 2π

QRUN

Qrun is a small executable program called by auto5. It takes into account the Johnson noise generated by resistances.

To take into account the Johnson noise, you have to add the file *noise.awkf* in your directory. Make sure that when JSIM calls AUTO5, that noise is taken into account. To do that, you have to modify the file qrun like this. Copy the noise.awkf file in the main cell directory. Make also sure that you can open that file. If not, change rights like this
chmod 777 noise.awkf.

See also the README file for more informations.

IV-Files configurations

IV-1 list of files described previously

The files needed for the simulations of the dcsfq are:

- The netlist called here *testjsim.js*
- The file with parameters *testjsim.para*
- *Testjsim.conf* the AUTO5 program configuration
- *Marginout1.tagr* used by GRACE to plot results

  The files listed above are the essential files to start simulations.

  There are also files for easy simulation:
- File A to copy the file testjsim-tmp.js in testjsim-t.js in the main cell directory
- File t to plot voltages , phases or current results
- File b to calculate margins
- File d to calculate yields

  Other intermediates files are created during simulations:
- *testjsim.dat* which contains the netlist, the results of the simulation and simulations statistics.
- *testjsim.soll* which contains normalized values after margins calculations.
- There is also *testjsim.mar* created during margins calculations.
- And finally there is testjsim.yi created during yields calculations.

IV-2 Schematic of the DCSFQ cell

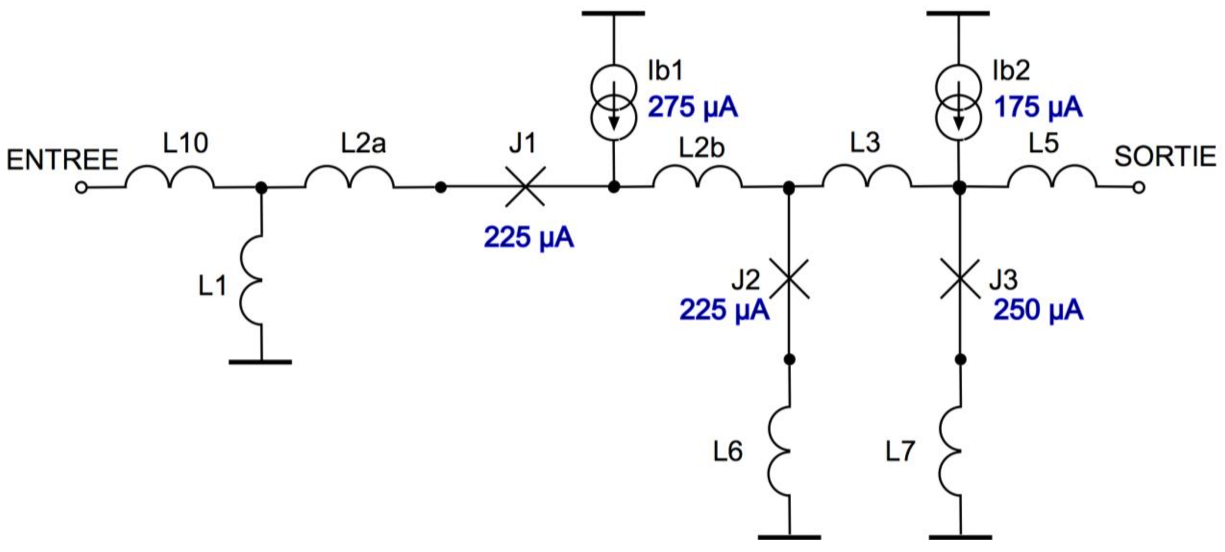


Figure 2 DCSFQ cell. This schematic corresponds to the DCSFQ netlist in the next page(File1).

```
* 12.04.2011 Benedicte NDENDE Modification
*============================================
.SUBCKT DCSFQ  0  30  2
*circuit
L10 30     3       @q pH
L1  3      0       @r pH
L2a 3      4       @s pH
L2b 11     5       @t pH
L3  5      6       @u pH
L4  6      7       @v pH
L5  7      2       @w pH
*junctions
B1   11    4       jdcsfq1
RB1  4     51      @xr
Lp1  51    11      @l pH

B2   5     8       jdcsfq2
RB2  5     81      @yr
Lp2  81    8       @m pH
L6   8     0       @e pH

B3   7     9       jdcsfq3
RB3  7     91      @zr
Lp3  91    9       @n pH
L7   9     0       @f pH
*bias current
ib1  0     11      pwl (0 0 5p @A uA 100n @A uA)
ib2  0     7       pwl (0 0 5p @B uA 100n @B uA)
.MODEL jdcsfq1 JJ(RTYPE=0, ICRIT= @x uA CAP= @xc PF RN=90)
.MODEL jdcsfq2 JJ(RTYPE=0, ICRIT= @y uA CAP= @yc PF RN=90)
.MODEL jdcsfq3 JJ(RTYPE=0, ICRIT= @z uA CAP= @zc PF RN=90)
.ENDS
*===================================================
.SUBCKT JTL  0  1  2
Ljtl1  1      4       @h pH
Ljtl2  4      8       @i pH
Ljtl3  8      5       @j pH
Ljtl4  5      2       @k pH

B1   4      6      jjtl1
RB1  4      9      @xr
Lj1  9      6      @d pH
Lj6  6      0      @g pH

B2   5      7      jjtl2
RB2  5      10     @yr
Lj2 10      7      @o pH
Lj7  7      0      @p pH
* Bias source 2.5mV and 7.41 Ohm
ib1  0     8       pwl (0 0 5p @C uA 100n @C uA)

.MODEL jjtl1 JJ(RTYPE=0, ICRIT= @a uA CAP= @ac PF RN=90)
.MODEL jjtl2 JJ(RTYPE=0, ICRIT= @b uA CAP= @bc PF RN=90)
.ENDS
*  Definition of full circuit
*===================================================
Isig1 0 30 pulse(0 @P uA 500ps 150ps 150ps 0.5ps 500ps)
*                 MAG    TD   TR  TF   PW   PERIOD
*===================================================
XDCSFQ DCSFQ   0 30 2
XJTL1 JTL 0 2 100
XJTL2 JTL 0 100 200
R1 200 0 2
* load with 2 ohms resistor

*.print phase XDCSFQ_B1
*.print phase XJTL1_B2
*.print phase XJTL2_B2

.print DEVI Isig1
.print DEVV XDCSFQ_B1
.print DEVV XDCSFQ_B2
.print DEVV XJTL2_B2

.tran 0.5ps 3000ps 0ps 0.5ps
*   PRSTEP TSTOP TSTART MAXTSTEP
.end
```

NB: Only junctions resistances and capacitances have parameters with 2 letters as their parameter because JSIM with the knowledge of the Mc Cumber parameter and the RnIc value capacitance.The other parameters should have only one letter. Also make sure that a letter is not repeated in file *testjsim.para*

File1: *testjsim.js*. This file is the dcsfq netlist file presented in the previous page (Figure2).

```
1 1.0 fix 0

2 1.0 XIc 0
5 1.0 XIb 0
6 1.0 XLM1 0
7 1.0 XLM2 0

q 0.1   L10   0
r 3.904 L1    0
s 0.604 L2a   0
t 1.126 L2b   0
u 4.484 L3    0
v 0.000 L4    0
w 2.080 L5    0

x 225 jdcsfq1 0
y 225 jdcsfq2 0
z 250 jdcsfq3 0

A 275 Ib1 0
B 175 Ib2 0

h 2.080 Ljtl1 0
i 2.059 Ljtl2 0
j 2.059 Ljtl3 0
k 2.080 Ljtl4 0

e 0.198 L6 0
f 0.110 L7 0
l 1.037 Lp1 0
m 1.037 Lp2 0
n 1.0   Lp3 0

d 1.0 Lj1 0
g 0.214  Lj6 0
o 1.0 Lj2 0
p 0.214 Lj7 0

a 250 Bjtl1 0
b 250 Bjtl2 0

C 350 Ibjtl 0

P 785 Isig1 1
```

File2 *testjsim.para*. This is the file which contains the parameters of the dcsfq cell.

Description of *xxx.para* file

Example:

1 1.0 fix 0   this line should stay like this

2 1.0 XIc 1

3 1.0 XIb 0

- the first figure is a reference number for global variations. It is referred by the local parameters detailed below. For instance the critical current of junction J1 can be varied locally but also globally (then all critical currents of all junctions also vary by a factor given by the second figure (1.0 here)
- the second parameter is the global multiplier (Xic values are multiplied by this value during margin and yield analysis)
- the third parameter is the name of the parameters to be varied globally (Josephson junction currents, bias currents,…)
- the last parameter is 1 when global variations are done during calculations, and 0 when they are not considered.

a 250.0 J1 2

b 175.0 J2 2

- the first parameter (a small cap letter) is the parameter used in *testjsim.js* file
- the second one is the nominal value of the parameter
- the third parameter is the name of the element as displayed in GRACE.
- the last parameter is: 1 when local variations are taken into account
- 0 when local variations are not considered.

  a number >1 when local and global variations are taken into account simultaneously; the number is the first parameter of global parameters (2 for XIc or 3 for XIb in this example)

A 2.207 L1 0

B 0.968 L2a 1

u 250.0 IB1 3

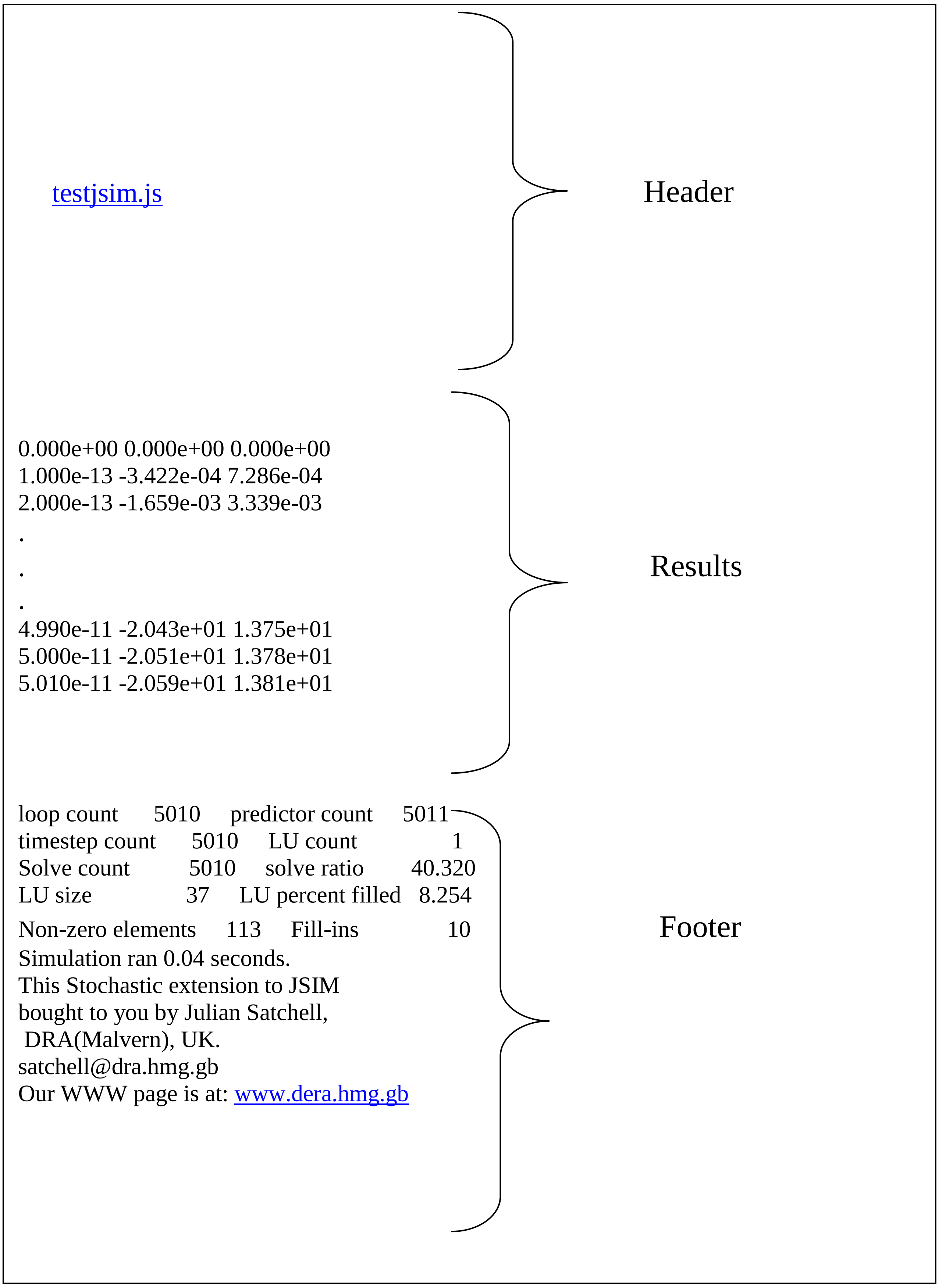
v 100.0 IB2 3

```
%------------------------------------------------------------
% Configuration data: dcsfq
%
%
% Torsten Reich
% 01.04.2003
% modifications: Pascal Febvre
% 14.04.2011: Ndende benedicte
%------------------------------------------------------------
initrand 434375233
%-----------Parameter----------------------------------------
I0RN 0.600e-3          % IcRN product
betac 0.04              % Mc Cumber Parameter
timestep 400        % Time interval for extraction  of digital data (ps)
extstep 0.0010   % Offset for data display with xmGrace
% -- parameters for making binary decisions for digital tests -----
voltageswitch 0
maxzerovoltage 0 % in microvolts
minonevoltage 10 % in microvolts
% if voltageswitch is zero all parameters are assumed to be phases
% if voltageswitch is one the last parameter is assumed to be voltage
%----------------- input data -------------------------------
compin  ../testjsim/testjsim/.soll  % Theoretical values of phases
paracir ../testjsim/testjsim.js    % Netlist with variable parameters
parain  ../testjsim/testjsim.para  %  Netlist parameters
%-----------intermediate results--------------------------------
simout  /tmp/testjsim.dat     % Results from simulation
extout  /tmp/testjsim.ext     % Results from phase extraction
simcir  /tmp/testjsim -tmp.js  % Netlist for simulation
simtest /tmp/testjsim.test    % Test function
%------------ output data -------------------------------------
paraneu ../testjsim/testjsim.para  % Parameter set after Monte-Carlo optimization
cirmar  ../testjsim/testjsim.mar   % Margins in text format
xmgrmar ../testjsim/testjsim.agr   % Margins display with xmGrace
yield   ../testjsim/testjsim.yi    % Results from dispersion analysis
%------------------------------------------------------------
end
% everything after end is ignored
```

File3 *testjsim.conf* which contains the directories used to store simulation and optimization files

```
# Grace project file
#
@version 50102
@page size 792, 612
@page scroll 5%
@page inout 5%
@link page off
@map font 0 to "Times-Roman", "Times-Roman"
@map font 1 to "Times-Italic", "Times-Italic"
@map font 2 to "Times-Bold", "Times-Bold"
@map font 3 to "Times-BoldItalic", "Times-BoldItalic"
@map font 4 to "Helvetica", "Helvetica"
@map font 5 to "Helvetica-Oblique", "Helvetica-Oblique"
@map font 6 to "Helvetica-Bold", "Helvetica-Bold"
@map font 7 to "Helvetica-BoldOblique", "Helvetica-BoldOblique"
@map font 8 to "Courier", "Courier"
@map font 9 to "Courier-Oblique", "Courier-Oblique"
@map font 10 to "Courier-Bold", "Courier-Bold"
@map font 11 to "Courier-BoldOblique", "Courier-BoldOblique"
@map font 12 to "Symbol", "Symbol"
@map font 13 to "ZapfDingbats", "ZapfDingbats"
@map color 0 to (255, 255, 255), "white"
@map color 1 to (0, 0, 0), "black"
@map color 2 to (255, 0, 0), "red"
@map color 3 to (0, 255, 0), "green"
@map color 4 to (0, 0, 255), "blue"
@map color 5 to (255, 255, 0), "yellow"
@map color 6 to (188, 143, 143), "brown"
@map color 7 to (220, 220, 220), "grey"
@map color 8 to (148, 0, 211), "violet"
@map color 9 to (0, 255, 255), "cyan"
@map color 10 to (255, 0, 255), "magenta"
@map color 11 to (255, 165, 0), "orange"
@map color 12 to (114, 33, 188), "indigo"
@map color 13 to (103, 7, 72), "maroon"
@map color 14 to (64, 224, 208), "turquoise"
@map color 15 to (0, 139, 0), "green4"
@reference date 0
@date wrap off
@date wrap year 1950
@default linewidth 1.0
@default linestyle 1
@default color 1
@default pattern 1
@default font 0
@default char size 1.000000
@default symbol size 1.000000
@default sformat "%16.8g"
@background color 0
@page background fill on
@timestamp off
@timestamp 0.03, 0.03
@timestamp color 1
@timestamp rot 0
@timestamp font 0
@timestamp char size 1.000000
@timestamp def "Mon Sep 17 08:28:05 2001"
```

File4 *marginout1.tagr*



testjsim.js

```
0.000e+00 0.000e+00 0.000e+00
1.000e-13 -3.422e-04 7.286e-04
2.000e-13 -1.659e-03 3.339e-03
.
.
.
4.990e-11 -2.043e+01 1.375e+01
5.000e-11 -2.051e+01 1.378e+01
5.010e-11 -2.059e+01 1.381e+01
```

```
loop count       5010    predictor count     5011
timestep count      5010    LU count               1
Solve count         5010    solve ratio       40.320
LU size               37    LU percent filled  8.254
Non-zero elements     113   Fill-ins              10
Simulation ran 0.04 seconds.
This Stochastic extension to JSIM
bought to you by Julian Satchell,
 DRA(Malvern), UK.
satchell@dra.hmg.gb
Our WWW page is at: www.dera.hmg.gb
```

File5 *testjsim.dat* contains the dcsfq netlist, the results of the simulation and also simulations statistics.

```
Auto5 testjsim 2 1 0
cp /tmp/testjsim-tmp.js testjsim-t.js
```

File6 File A usually use to facilitate the copy of the netlist from the /tmp to the main cell directory

```
#!/bin/bash
#file t

# A program inclusion
./A

# cutt program inclusion
jsim_n testjsim-t.js > /tmp/ testjsim.dat
cutt dcsfq
cp /tmp/ testjsim.ext testjsim _signal.ext
xmgrace -autoscale xy -nxy testjsim _signal.ext &

#inclusion du programme ext
jsim_n testjsim -t.js > /tmp/ testjsim.dat
ext testjsim
cp /tmp/ testjsim.ext testjsim.soll

# removal of intermediate files
rm -f /tmp/ testjsim *

#end
```

File7 File t is used for simulations, displaying results and removing intermediate files

```
#!/bin/csh
# file d
# yield analysis


cd /home/benedicte/testjsim

# first parameter for yield analysis
# second parameter for number of attempts
# third parameter for maximum deviation sigma
# fourth parameter for number of different deviations in the sigma range
Auto5 testjsim 3 2000 70 40

# end
```

File8: File d used for yields calculations

Replace csh with your shell if it is not the right one

```
#!/bin/csh
# file b
# margins calculations

cd /home/benedicte/testjsim

# inclusion du contenu du fichier b
Auto5 testjsim 1 30
#xmgrace testjsim.agr  &

# end
```

File9 File b used for margins calculations

The first column specifies the time in which the dcsfq is analysed (variable timestep in the testjsim.conf file) the 3 columns after correspond to the different .print commands in the netlist *testjsim.js*

```
400.000000 0 0 0
800.000000 1 1 1
1200.000000 1 2 2
1600.000000 2 3 3
2000.000000 3 3 3
2400.000000 4 4 4
2800.000000 5 5 5
```

File10 *testjsim.soll*

Initial values

Final values

```
Parameter :'q'   (0.1000,3.0000) -> (0.0100,0.3000)
Parameter :'r'   (0.8142,1.0272) -> (3.1787,4.0101)
Parameter :'s'   (0.1000,3.0000) -> (0.0604,1.8120)
Parameter :'t'   (0.7387,1.7636) -> (0.8317,1.9858)
Parameter :'u'   (0.4065,1.2840) -> (1.8227,5.7573)
Parameter :'v'   (0.1000,3.0000) -> (0.0000,0.0000)
Parameter :'w'   (0.1000,3.0000) -> (0.2080,6.2400)
Parameter :'x'   (0.4166,1.2269) -> (93.7272,276.0510)
Parameter :'y'   (0.9476,1.1471) -> (213.2041,258.0940)
Parameter :'z'   (0.8498,1.3542) -> (212.4543,338.5614)
Parameter :'P'   (0.8188,1.0271) -> (655.0637,821.6498)
```

File11 *testsjims.mar* created contains margins calculated by JSIM

```
5.0  100.00
 10.0  100.00
 15.0  92.50
 20.0  90.00
```

File12 *testjsim.yi* used by GRACE and created after batch command program “d”.

V-AN EXAMPLE

To clarify what we said before, we are going to do all steps from simulation to yields calculations on the dcsfq+ 2JTL + sfqdc circuit.

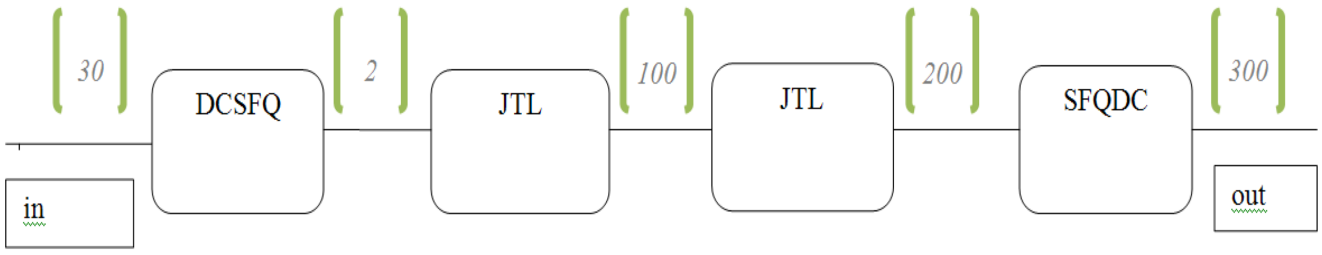


Figure2 schematic of dcsfq+ 2JTL + sfqdc with nodes

The netlist of the dcsfq cell on page 21.

The next figure represents the JTL schematic to be used.

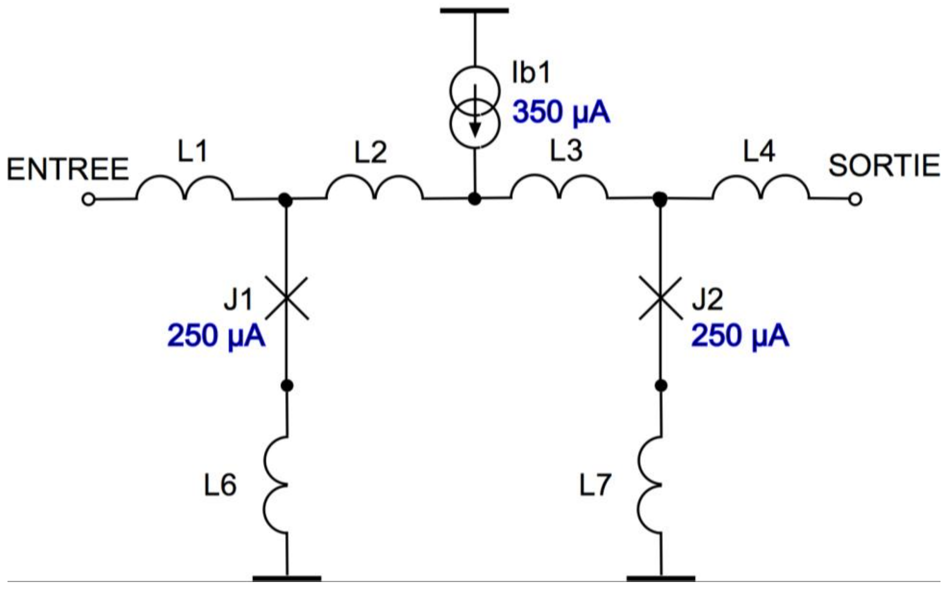


Figure3 JTL schematic

After the JTL cell, the SFQDC cell. Figure4 in the next page is the SFQDC schematic to be used.

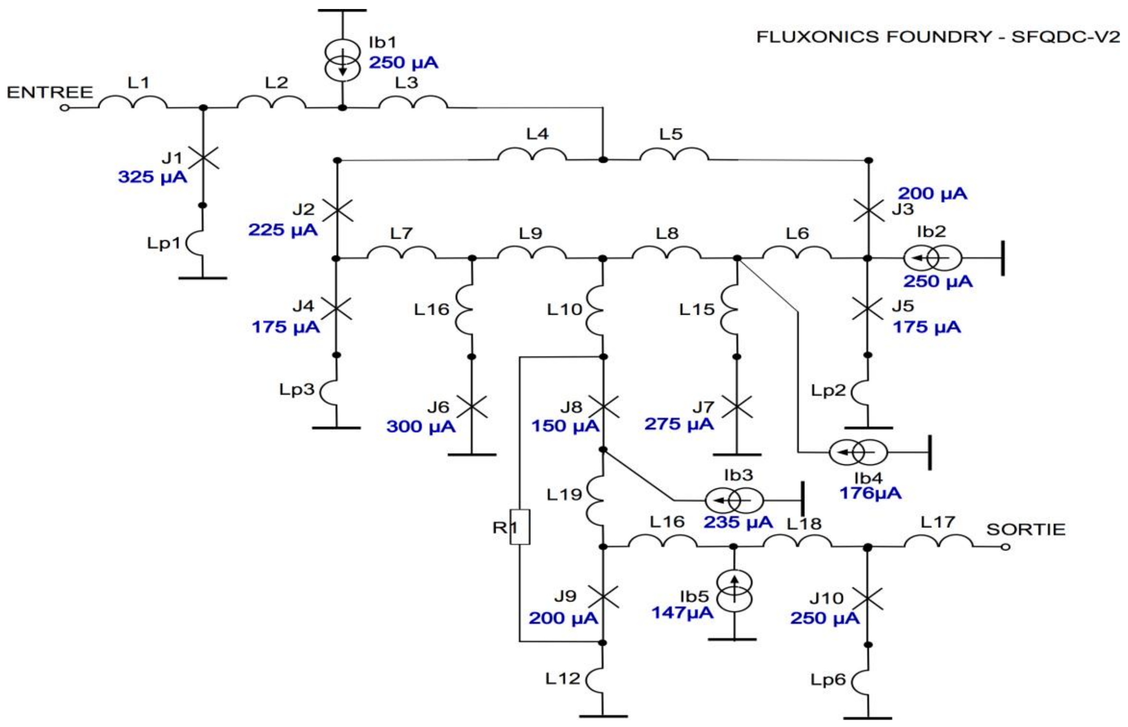


Figure4: sfqdc schematic put after DCSFQ+2JTL

After creating the directory SFQDC, we added all the files necessary for simulations. We also copy the two files marginout1.agr and marginout2.agr in the main cell directory named sfqdc. Three batch command files are also present.

```
auto5 sfqdc 2 1 0
cp /tmp/sfqdc-tmp.js sfqdc-t.js
```

File A

```
#!/bin/bash
#file t
./A
#program cutt inclusion
jsim_n sfqdc-t.js > /tmp/sfqdc.dat
cutt sfqdc
cp /tmp/sfqdc.ext sfqdc_signal.ext
xmgrace -autoscale xy -nxy sfqdc_signal.ext &

# program ext inclusion
jsim_n sfqdc-t.js > /tmp/sfqdc.dat
ext sfqdc
cp /tmp/sfqdc.ext sfqdc.soll

rm -f /tmp/sfqdc*
#end
```

File t

```
#!/bin/bash
# file b
#set path=(home/benedicte/testcellule/sfqdc $path )
#set path=(home/benedicte/Logiciels/jsim_n $path)

cd /home/benedicte/testcellule/sfqdc

# inclusion du contenu du fichier b
auto5 sfqdc 1 30
xmgrace sfqdc.agr  &

# fin inclusion du contenu du fichier b
```

File b

Then, we created the *sfqdc.conf*

```
% Configuration data: dcsfq_cell
% 12.04.2011 Benedicte NDENDE
%------------------------------------------------------------
initrand 434375233
%-----------Parameter----------------------------------------
I0RN 0.256e-3          % I0RN Product
betac 1                % Mc Cumber Parameter
timestep 400         % Time interval for extraction  of digital data (ps)
extstep 0.001   % Offset for data display with xmGrace
%-----------parameters for making binary decisions for digital tests-----------
voltageswitch 1
maxzerovoltage  2% in microvolts
minonevoltage  4% in microvolts
%-----------input data-----------------------------------------
compin  ../sfqdc/sfqdc.soll  % Theoretical values of phases
paracir ../sfqdc/sfqdc.js    % Netlist with variable parameters
parain  ../sfqdc/sfqdc.para  % Netlist parameters
%-----------intermediate results--------------------------------
simout  /tmp/sfqdc.dat     % Results from simulation
extout  /tmp/sfqdc.ext     % Results from phase extraction
simcir  /tmp/sfqdc-tmp.js   % Netlist for simulation
simtest /tmp/sfqdc.test    % Test function
%-----------output data-------------------------------------
paraneu ../sfqdc/sfqdc.para  % Parameter set after Monte-Carlo optimization
cirmar  ../sfqdc/sfqdc.mar   % Margins in text format
xmgrmar ../sfqdc/sfqdc.agr   % Margins display with xmGrace
yield   ../sfqdc/sfqdc.yi    % Results from dispersion analysis
%------------------------------------------------------------
end
% everything after end is ignored
```

We created the SFQDC netlist and added the netlist of DCSFQ+2JTL cell named *testjsim-t.js*.

```
*=========================================
*  SFQDC  JeSEF Technology
*  26.04.2004 Thomas Ortlepp new
*  11.05.2004 small modifications T.O.
*  02.07.2004 Version for Release M
*  15.02.2005 Version 2
*  16.11.2005 Thomas Ortlepp modify
*  Quelle: 550uA
*  21.04.2011 Benedicte NDENDE Modification
*=============================================

.SUBCKT DCSFQ  0  30  2

*circuit
L10 30     3       0.100000pH
L1  3      0       3.904000pH
L2a 3      4       0.604000pH
L2b 11     5       1.126000pH
L3  5      6       4.484000pH
L4  6      7       0.000000pH
L5  7      2       2.080000pH



*junctions
B1   11    4       jdcsfq1
RB1  4     51       1.137778

Lp1  51    11       1.037000pH

B2   5     8       jdcsfq2
RB2  5     81       1.137778

Lp2  81    8       1.037000pH
L6   8     0       0.198000pH

B3   7     9       jdcsfq3
RB3  7     91       1.024000

Lp3  91     9       1.000000pH
L7   9      0      0.110000pH

*bias current
ib1  0     11      pwl (0 0 5p 275.000000uA 100n 275.000000uA)
ib2  0     7      pwl (0 0 5p 175.000000uA 100n 175.000000uA)

.MODEL jdcsfq1 JJ(RTYPE=0, ICRIT= 225.000000uA CAP= 1.131079PF RN=90)
.MODEL jdcsfq2 JJ(RTYPE=0, ICRIT= 225.000000uA CAP= 1.131079PF RN=90)
.MODEL jdcsfq3 JJ(RTYPE=0, ICRIT= 250.000000uA CAP= 1.256755PF RN=90)
.ENDS
```

Model names should be different even if they don't belong to the same Sub-circuit

Here is the JTL description and a part of the SFQDC cell

```
.SUBCKT JTL  0  1  2

Ljtl1  1      4        2.080000pH
Ljtl2  4      8        2.059000pH
Ljtl3  8      5        2.059000pH
Ljtl4  5      2        2.080000pH

B1   4      6       jjtl1
RB1  4      9       1.137778
Lj1  9      6       1.000000pH
Lj6   6      0      0.214000pH
B2   5      7       jjtl2
RB2  5     10       1.137778

Lj2 10      7       1.000000pH
Lj7   7      0      0.214000pH
* Bias source 2.5mV and 7.41 Ohm
ib1  0     8        pwl (0 0 5p 350.000000uA 100n 350.000000uA)

.MODEL jjtl1 JJ(RTYPE=0, ICRIT= 250.000000uA CAP= 1.256755PF RN=90)
.MODEL jjtl2 JJ(RTYPE=0, ICRIT= 250.000000uA CAP= 1.256755PF RN=90)
.ENDS
*=====================================================
.SUBCKT SFQDC  0  1  17

*circuit
L1   1    2       @Q pH
L2   2    3       @R pH
L3   3    4       @S pH
L4   4    5       @T pH
L5   4    6       @U pH
L6   7    9       @V pH
L7   8   10       @W pH
L8   9   11       @D pH
L9  10   11       @E pH
L10 11   12       @F pH
L12 14    0       @G pH
L13 13   16       @H pH
L15  9   18       @I pH
L16 10   19       @J pH
L17 30   17       @K pH
L18 16   30       @L pH
L19 32   13       @M pH

*junctions
B1     2     21     jjth1
RB1    2    210     @mr
Ljp1 210     21     @n pH

B2     6     8      jjth2
RB2    6   810      @vr
Ljp2 810     8      @o pH
```

The end of the sfqdc description is here

```
B4    8   81        jjth4
Ljp4  81  811       @e pH
RB4   811  8        @xr
B5    7   71        jjth5
Ljp5  7  711        @f pH
RB5   711  71       @yr

B6   19     0       jjth6
Ljp6 19    192      @g pH
RB6  192    0       @zr

B7   18     0       jjth7
Ljp7 18    182      @h pH
RB7  182    0       @ar

B8   32   12        jjth8
Ljp8 12   123       @i pH
RB8  123  32        @br

B9   13   14        jjth9
Ljp9 14   134       @j pH
RB9  134  13        @dr

B10   30   31       jjth10
Ljp10 30   311      @k pH
RB10  311  31       @cr

LR1  12   122       @l pH
R1   122   14       @O Ohms

Lp1  21    0    @q pH
Lp2  71    0    @r pH
Lp3  81    0    @s pH
Lp6  31    0    @t pH

*bias current
ib1  0     3      pwl (0 0 5p @A uA 100n @A uA)
ib2  0     7       pwl (0 0 5p @B uA 100n @B uA)
ib4   0    9       pwl (0 0 5p @C uA 100n @C uA)
ib3   0    32      pwl (0 0 5p @X uA 100n @X uA)
ib5   0    16      pwl (0 0 5p @Y uA 100n @Y uA)

.MODEL jjth1 JJ(RTYPE=0, ICRIT= @m uA CAP= @mc PF RN=90)
.MODEL jjth2 JJ(RTYPE=0, ICRIT= @v uA CAP= @vc PF RN=90)
.MODEL jjth3 JJ(RTYPE=0, ICRIT= @w uA CAP= @wc PF RN=90)
.MODEL jjth4 JJ(RTYPE=0, ICRIT= @x uA CAP= @xc PF RN=90)
.MODEL jjth5 JJ(RTYPE=0, ICRIT= @y uA CAP= @yc PF RN=90)
.MODEL jjth6 JJ(RTYPE=0, ICRIT= @z uA CAP= @zc PF RN=90)
.MODEL jjth7 JJ(RTYPE=0, ICRIT= @a uA CAP= @ac PF RN=90)
.MODEL jjth8 JJ(RTYPE=0, ICRIT= @b uA CAP= @bc PF RN=90)
.MODEL jjth9 JJ(RTYPE=0, ICRIT= @d uA CAP= @dc PF RN=90)
.MODEL jjth10 JJ(RTYPE=0, ICRIT= @c uA CAP= @cc PF RN=90)
```

The full circuit description is here.

```
*=====================================================
*  Definition of full circuit
*=====================================================
*signal current

Isig1 0 30 pulse(0 @P uA 600ps 150ps 150ps 0.5ps 500ps)
*                 MAG    TD   TR  TF   PW   PERIOD


*========================================
*=====================================================
XDCSFQ DCSFQ   0 30 2
XJTL1 JTL 0 2 100
XJTL2 JTL 0 100 200
XSFQDC SFQDC 0 200 300
*R2 300 0 0.4
* load with 0.4 ohms resistor

*.print DEVI Isig1
*.print phase XJTL2_B2
*.print phase XSFQDC_B1
*.print phase XSFQDC_B10



.print DEVI Isig1
.print DEVV XJTL2_B2
.print DEVV XSFQDC_B1
.print DEVV XSFQDC_B10
.print NODEV 300 0


.tran 0.5ps 3500ps 0ps 0.5ps
*   PRSTEP TSTOP TSTART MAXTSTEP
.end
```

And finally we have created the *sfqdc.para* file.

```
1 1.0 fix 0
2 1.0 XIc 0
5 1.0 XIb 0
6 1.0 XLM1 0
7 1.0 XLM2 0

Q 1.415 L1   0
R 0.448 L2   0
S 0.659 L3   0
T 1.020 L4   0
U 1.122 L5   0
V 1.489 L6   0
W 0.862 L7   0
D 2.634 L8   0
E 1.590 L9   0
F 0.554 L10  0
G 0.231 L12  0
H 4.951 L13  0
I 0.822 L15  0
J 1.270 L16  0
K 1.084 L17 0
L 1.738 L18  0
M 0.869 L19  0

n 0.930 Ljp1 0
o 1.050 Ljp2 0
p 1.090 Ljp3 0
e 1.150 Ljp4 0
f 1.150 Ljp5 0
g 0.951 Ljp6 0
h 0.980 Ljp7 0
i 1.470 Ljp8 0
j 1.070 Ljp9 0
k 1.080 Ljp10 0
l 0.910 LR1 0
O 0.375 R1 0

q 0.186 Lp1 0
r 0.625 Lp2 0
s 0.545 Lp3 0
t 0.221 Lp6 0

m 325 jjth1 1
v 225 jjth2 1
w 200 jjth3 1
x 175 jjth4 1
y 175 jjth5 1
z 300 jjth6 1
a 275 jjth7 1
b 150 jjth8 1
d 200 jjth9 1
c 250 jjth10 1

A 250 Ib1 1
B 160 Ib2 1
C 176 Ib3 1
X 235 Ib4 1
Y 147 Ib5 1

P 785 Isig1 0
```

Simulations can start. You can enter the command ./t on the main cell directory sfqdc to execute the script t. If you want to simplify more add to the file */.bashrc* the path to your directory.

```
export
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/X11R6/bin:/home/benedi
cte/Logiciels:/home/benedicte/testcellule:/home/benedicte/testcellule/sfqdc:/home/benedicte/te
stcellule/dcsfq
```

Be careful, the first directory in which file t will be founded is the one that ubuntu is going to choose. It can be confusing.
You can plot voltages results now by just entering the letter "t" in the main cell directory.
The results are:

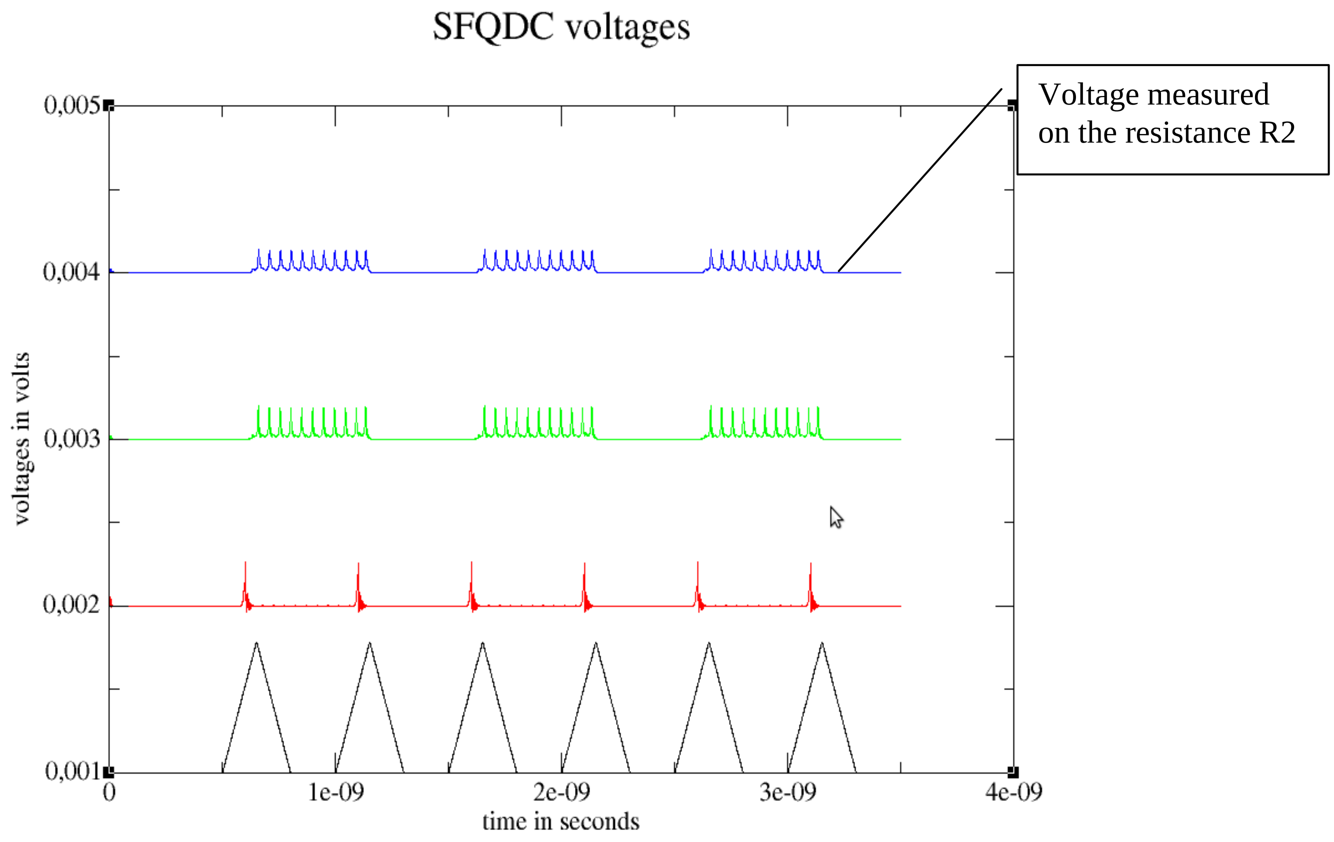


Figure5 SFQDC simulations

Continue with the steps from margins calculations to yields optimization.

For margins, make sure that you comment all the voltages except the voltage of R2. For margins ones obtaines:

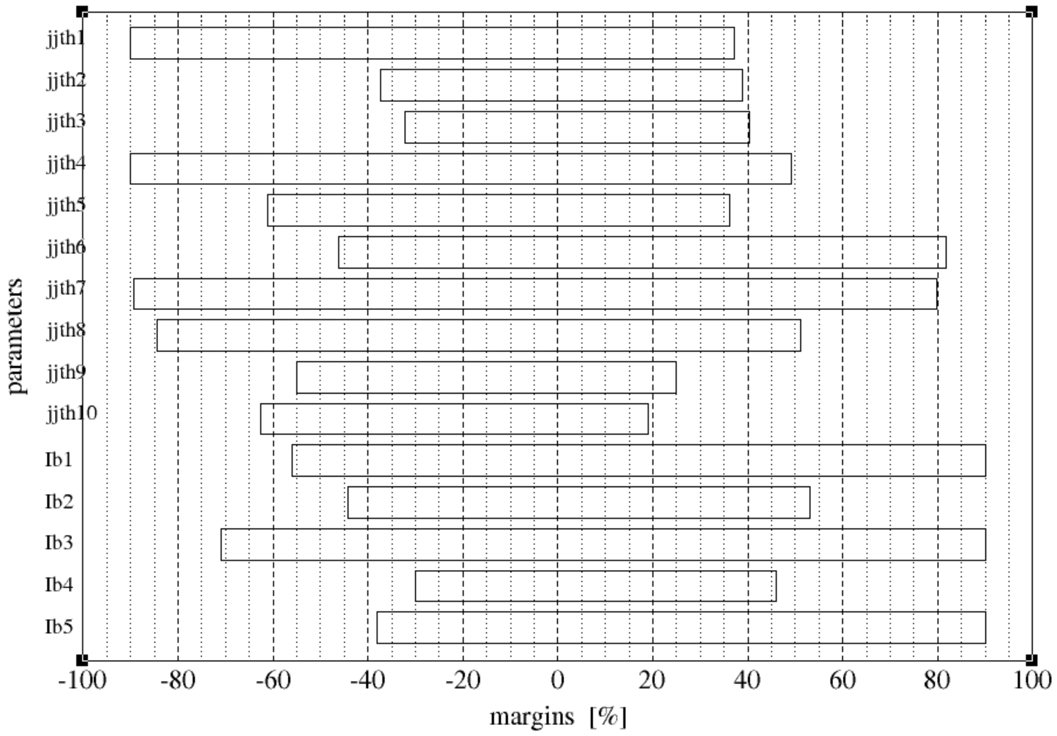


Figure6: sfqdc margins. Only critical and polarization current are variable in the sfqdc.para.

For yields with **auto5 sfqdc 3 1000 28 20**
These are the results we founded:

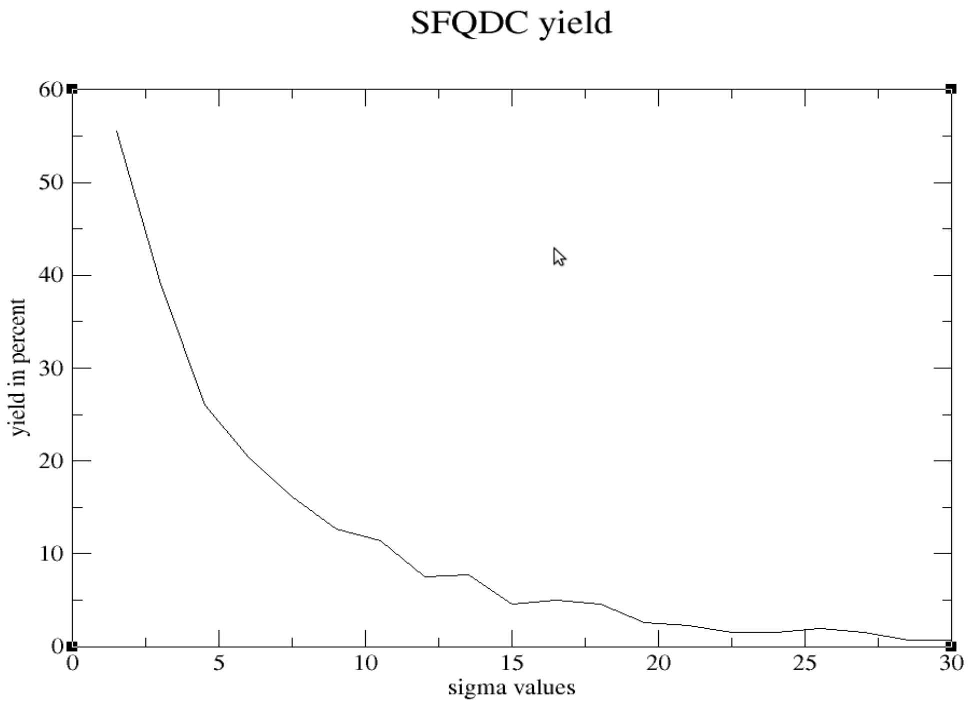


Figure7: sfqdc yields

VI-NOISE

JSIM can take in account the Johnson noise generated by resistors to perform the simulations. Noise inclusion is achieved first by the modification of the file **qrun** located in the folder jsim_n:

DON'T FORGET TO ADD the variable TEMP in the .bashrc file by adding **export TEMP= value**

```
rm $2

#noise inclusion

awk -f noise.awkf temperature=$TEMP < $1 > /tmp/noise.js
cp /tmp/noise.js $1
rm /tmp/noise.js

#end noise inclusion

jsim_n $1 > $2
ext $3
rm $4
tabcomp $3 v >>  $4

# $1  <simcir>
# $2  <simout>
```

Then you will have to add the file **noise.awkf** in your work directory.

Let's now take an example with the case of the DCSFQ cell. Actually, we included the noise for a temperature T=4.2K. What JSIM does actually when including the noise is to add on each resistors a noise source, (here a current one) whose spectral amplitude density can be computed here by the following formula:

$$A=\mathrm{sqrt}(4*k_B*T/R)$$

where R is the value of the resistance, k_B the Boltzmann constant and T the temperature.

Here is the beginning of the net list **dcsfq.js** concerning the DCSFQ circuit

```
*==========================================
.SUBCKT DCSFQ  0  30  2

*CIRCUIT
L10 30     3       @q pH
L1  3      0       @r pH
L2a 3      4       @s pH
L2b 11     5       @t pH
L3  5      6       @u pH
L4  6      7       @v pH
L5  7      2       @w pH

*JONCTIONS

B1   11    4       jdcsfq1
RB1  4     51      @xr
Lp1  51    11      @l pH

B2   5     8       jdcsfq2
RB2  5     81      @yr
Lp2  81    8       @m pH
L6   8     0       @e pH

B3   7     9       jdcsfq3
RB3  7     91      @zr
Lp3  91    9       @n pH
L7   9     0       @f pH

*POLARISATIONS

ib1  0     11      pwl (0 0 5p @A uA 100n @A uA)
ib2  0     7       pwl (0 0 5p @B uA 100n @B uA)

.MODEL jdcsfq1 JJ(RTYPE=0, ICRIT= @x uA CAP= @xc PF RN=90)
.MODEL jdcsfq2 JJ(RTYPE=0, ICRIT= @y uA CAP= @yc PF RN=90)
.MODEL jdcsfq3 JJ(RTYPE=0, ICRIT= @z uA CAP= @zc PF RN=90)
.ENDS
```

We are looking at the values of the critical current of the three junctions. The **.para** file was set with x=y=225uA et z=250uA. Given the Fluxonics Technology with $R_N I_c$ =0,256mV, it is easy to compute the values of RB1, RB2 and RB3, which will be given by JSIM :

RB1=RB2= (0,256e-3)/(225e-6)= 1,13778 Ω

RB3=(0,256e-3)/(250e-6)= 1,02400 Ω

Then, we can compute the spectral amplitude density of the current noise source, which will be added parallel on each resistors :

For RB1 and RB2, A= sqrt(4k_BT/RB1) = sqrt(4*1,38e-23*4,2/1,13778) = 14,28pA/sqrt(Hz)

For RB3, A= sqrt(4k_BT/RB3) = sqrt(4*1,38e-23*4,2/1,02400) = 15,05pA/sqrt(Hz)

We give now the net list with the value replaced (**dcsfq-t.js**) and we verify that the results match with our previous calculations:

```
*==============================================
.SUBCKT DCSFQ  0  30  2

*CIRCUIT
L10 30     3       0.100000pH
L1  3      0       3.904000pH
L2a 3      4       0.604000pH
L2b 11     5       1.126000pH
L3  5      6       4.484000pH
L4  6      7       0.000000pH
L5  7      2       2.080000pH

*JONCTIONS

B1   11    4       jdcsfq1
RB1  4     51      1.137778
iRB1 4 51 NOISE(14.286281p 0.0 1.0p)

Lp1  51    11      1.037000pH

B2   5     8       jdcsfq2
RB2  5     81      1.137778
iRB2 5 81 NOISE(14.286281p 0.0 1.0p)

Lp2  81    8       1.037000pH
L6   8     0       0.198000pH

B3   7     9       jdcsfq3
RB3  7     91      1.024000
iRB3 7 91 NOISE(15.059063p 0.0 1.0p)
Lp3  91    9       1.000000pH
L7   9     0       0.110000pH

*POLARISATIONS

ib1  0     11      pwl (0 0 5p 275.000000uA 100n 275.000000uA)
ib2  0     7       pwl (0 0 5p 175.000000uA 100n 175.000000uA)


.MODEL jdcsfq1 JJ(RTYPE=0, ICRIT= 225.000000uA CAP= 1.131079PF RN=90)
.MODEL jdcsfq2 JJ(RTYPE=0, ICRIT= 225.000000uA CAP= 1.131079PF RN=90)
.MODEL jdcsfq3 JJ(RTYPE=0, ICRIT= 250.000000uA CAP= 1.256755PF RN=90)
.ENDS
```

Noise sources are correctly added with the predicted spectral amplitude density

We give the readme file concerning the **qrun** file, which give more informations concerning especially the possibility to include or not the noise on each resistors, by simply changing their names. Concerning the shunt resistors of junctions : if you have a junction B1 with a shunt resistor called RB1 and you don't want noise to be applied to this shunt resistor, then simply change his name into RZ1.

```
* See below for revised make instructions                    *
/******************** ** ********************/
See the original README file, which I have renamed README.old.
I don't think I have anything to add to it, except I have removed the dos
make files as they would need modifications that I am not competent to make.

I have made a dumb makefile, which does not need the subsidiary ones in
the subdirectories. It also doesn't bother with ar or ranlib.

So now to implement, just edit makefile, (e.g change cc to gcc).
If you are on a machine which does not support random() in its c library,
set -DNORANDOM, and the horrible function rand() will be used. This is
a low quality random number generator, with a spectrum which is
coloured, but is supported on almost all machines. Alternatively
see if your system supports srand48/drand48, and edit the source code
to use those instead.
To improve the interactive feel of my graphics I have added
extra flushing statements, which will incur a small IO
overhead. If this bothers you, comment out the calls to fflush
in file jsimtxt/tran_n.c
An awk script has been provided which adds noise to circuit,
by including noise current source across every resistor.
To use it on a file circuit.js you could try

awk -f noise.awkf temperature=77.36 < circuit.js | jsim_n

As far as I know any of awk, nawk and gawk are OK. If you are
using DOS/Windoze, you are stuck, unless there is a DOS awk
implementation available somewhere on the net. The script is simple
enough that it could be emulated by some other programme if need be.

Resistors whose names start with Z are left alone, so you can
add noise free bias networks, output filters and the like.
There are some tests in the tests subdirectory.

Julian Satchell (satchell@dra.hmg.gb)      (+44) 1684 895003
```