Malt 3.0 Methods and Meaning

Every Option Defined! Nothing Left Out!

v0.1

Quentin Herr

25 July 2021

Meaning

Malt is a parametric yield calculator and optimizer of digital circuits. Malt is named in honor of Malthus, who as first to describe systems in which the demand for resources grows exponentially, while the resources themselves grow linearly. Multi-parameter optimization using limited compute resources is like that.

Algorithm for Calculating Yield. The yield algorithm calculates the multi-dimensional Gaussian integral across the operating region of the circuit. In the simplest case, yield is given by the Normalized Upper Incomplete Gamma Function, Q, which is a generalization of erfc. In N dimensions,

$$\text{Yieldc\_hypersphere}(N,r) = Q(N/2,r**2/2)$$
$$\text{Yieldc\_hypersphere}(1,r) = Q(1/2,r**2/2) = \text{erfc}(r/\text{sqrt}(2))$$

where the operating region of the circuit is defined by a single value, radius r. This reduces to the familiar complementary error function for N=1.

For an arbitrary operating region shape, yield is given by the integrating the Gaussian probability function over all the entire space enclosed by the operating region. Numerical integration can be used to compute the yield of a real circuit, by mapping out the organic shape of operating region piece-by-piece. If the shape of the operating region is well-behaved (see the requirements below), the operating region can be mapped out and partitioned into simplexes using a binary search to find points on the boundary of the operating region. Specifically, the set of points, n, associated with each simplex gives the mean value of the complementary yield, and the sigma. This set of points also gives the value of the solid angle of the simplex, Ω. These terms provide a numerical estimate for the contribution of each simplex:

$$\text{Yieldc}(n) = \Omega(n)\ \text{ave}\,(Q(N/2,r_n**2/2))$$
$$\text{Yieldc\_var}(n) = \Omega(n)\ \text{var}(Q(N/2,r_n**2/2))$$
$$\text{Yieldc} = \Sigma_n\ \text{Yieldc}(n) \pm (\Sigma_n\ \text{Yieldc\_var}(n))^{1/2}.$$

The contribution of each simplex can be used in an adaptive algorithm that determines where to add new points on the boundary.

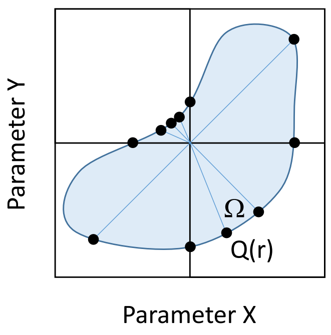

Fig. 1. In the numerical integration, the average Q value of each simplex (defined by N points in N dimension) is weighted by the solid angle Ω. The weighted sigma of the Q values gives an error estimate.

Malt first calculates the margin of the "corner vector" in each quadrant, octant, or orthant, and then calculates yield on a per-orthant basis using the margin of the corner vector to predict the contribution. This is effective to the extent that the parameters are highly interdependent. (If one or more of your parameters is not highly interdependent, so much the better. Remove it from the group and calculate its contribution to yield based on individual parameter margins using the Yieldc_box expression above.) The number of orthants is large, equal to 2^N. Meanwhile, the yield integral is dominated by the points closest to the origin (nominal parameter values), due to the rapid decay of the Gaussian. It means that the problem reduces to evaluating yield only in those orthants that have points closest to the origin.

The algorithm requires that the binary searches used to define the operating region are single-valued. Some different operating regions that illustrate this requirement are shown in Fig. 2.

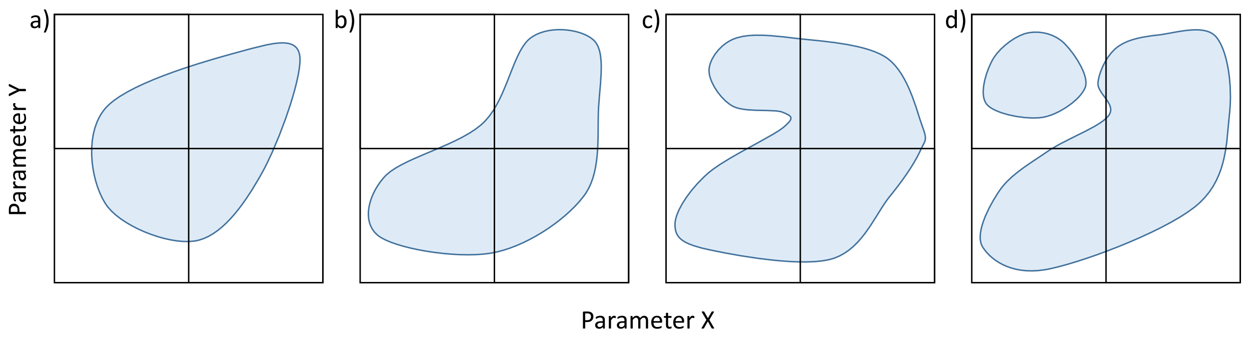

Fig 2. Four operating scenarios are shown: a) convex, b) not convex, but single-valued, c) not single-valued, but simply-connected d) not simply-connected. The optimization algorithm requires that the operating region be convex. The yield algorithm has the less restrictive requirement that the operating region boundary be single-valued for binary searches starting at the origin. This may depend on the positioning of the origin within the operating region.

Algorithm for Parametric Yield Optimization. Malt optimizes parameter values using the design-centering algorithm of Director and Hachtel, IEEE Circuits and Systems, 1977. The algorithm maps out the operating region using the binary search, builds a simplex to approximate the operating region, and inscribes the largest possible hypersphere (effectively a hyperellipsoid) in the simplex using linear programming. The center of the hypersphere corresponds to optimal parameter values. This heuristic is a good approximation to yield optimization as it positions the parameter values as far as possible from the edges of the operating region in multi-parameter space. The complexity of the simplex is large. In eight dimensions, 100 iterations of the binary search might well produce 100,000 simplex elements. Higher dimensions can be tried, but the computation time will increasingly be dominated by simplex operations instead of circuit simulation.

The convexity requirement, illustrated in Fig. 2, bears further discussion. Josephson circuits using SFQ or RQL data encoding are usually well-behaved in this respect, and the optimization is fast and efficient. The algorithm may fail for circuits with multiple internal modes of operation that pass, or circuits with underdamped junctions that have stochastic effects. The routine reports the validity of the convexity assumption after completing parameter optimization, and then recalculates individual parameter margins. This makes the success or failure of the optimization apparent.

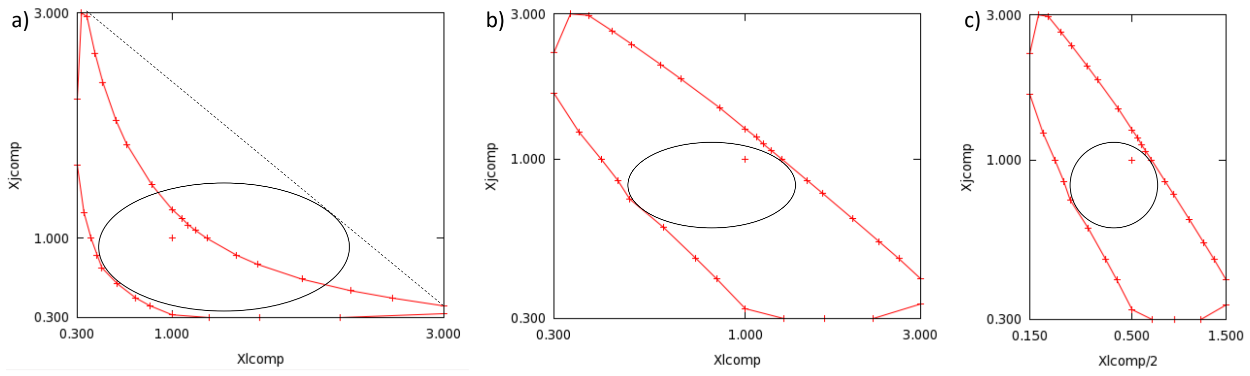

Fig. 3. Malt-space transforms are illustrated. a) Many parameters have an inverse dependence, causing a strongly non-convex operating region. Points interior to the convex simplex will be ignored, leading to a poor approximation. b) Transforming parameters to log space solves gives the desired result. c) The optimization algorithm effectively inscribes an ellipse in the operating region, with axes proportional to the sigma of each parameter. This is actually implemented by inverse scaling of the parameter values in malt space.

Log Space. Converting parameters to log space can help achieve the convexity requirement, as shown in Fig. 3. In fact, most parameters are Gaussian distributed in log space, not linear space. This is generally true of any parameter for which it is unphysical to have both positive and negative values, such as inductors, capacitors, resistors, and junction critical currents. Parameters that are better in normal space include bias currents such as the bias tap offset and input pattern offset. What about the ac clock amplitude? That is better in linear space, However, if you specify the ac clock in dB, you should use log space. In fact, these two paradigms are mathematically equivalent! In linear space, the center between high and low values is the mean. In log space, it is the geometric mean. The transform to log space is

$$f(x) = nom*(log(x/nom)+1.0),$$

where nom is the nominal parameter value. The function maps the nominal value to itself. The function is plotted in Fig. 4.

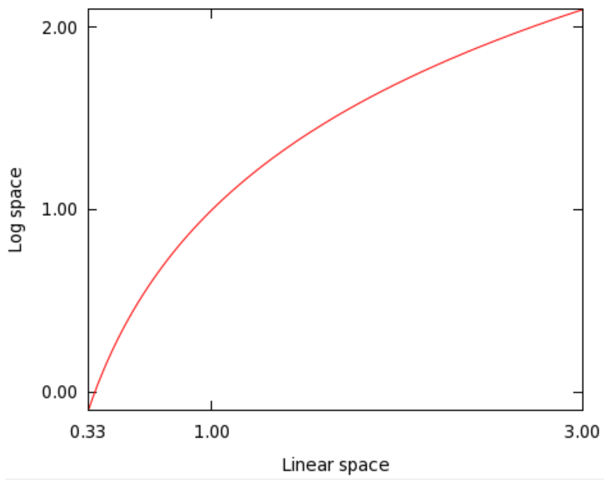

Fig. 4. Figure on the transform in units of nominal. This transform is applied before the scaling with sigma mentioned above.

Linear vs. log-space is on a per-parameter basis and they can be mixed. Either way, high and low margins are quoted both has physical parameter values, and in units of sigma. Margins in units of sigma can be derived from the physical parameter values for both linear and log space as follows:

$$\text{marg\_lin}(x) = (x\text{-nom})/\text{sig}$$
$$\text{marg\_log}(x) = (f(x)\text{-nom})/\text{sig}$$

where x is the high or low physical parameter value, sig is the sigma value. As defined above, nom is the nominal physical parameter value and f(x) is the transform to log space.

Corner parameters. Corner parameters are not explicitly included in the binary search, but are instead alternately set to high and low values. If there are N corner parameters, there are 2^N combinations of high and low, which are the parameter corners. Margins are calculated at each parameter corner in parallel. Only the worst case margin is reported. This means that the operating region of the circuit is the logical intersection of the operating regions at all corners. This is illustrated in Fig. 5. Parameters that have uniform distributions should be specified as corner parameters. This is true for global parameters, where parts that are out-of-tolerance are discarded. Treating globals as corner parameters has an additional advantage for optimization, as their inclusion in this way does not increase the dimensionality of the operating-region simplex.

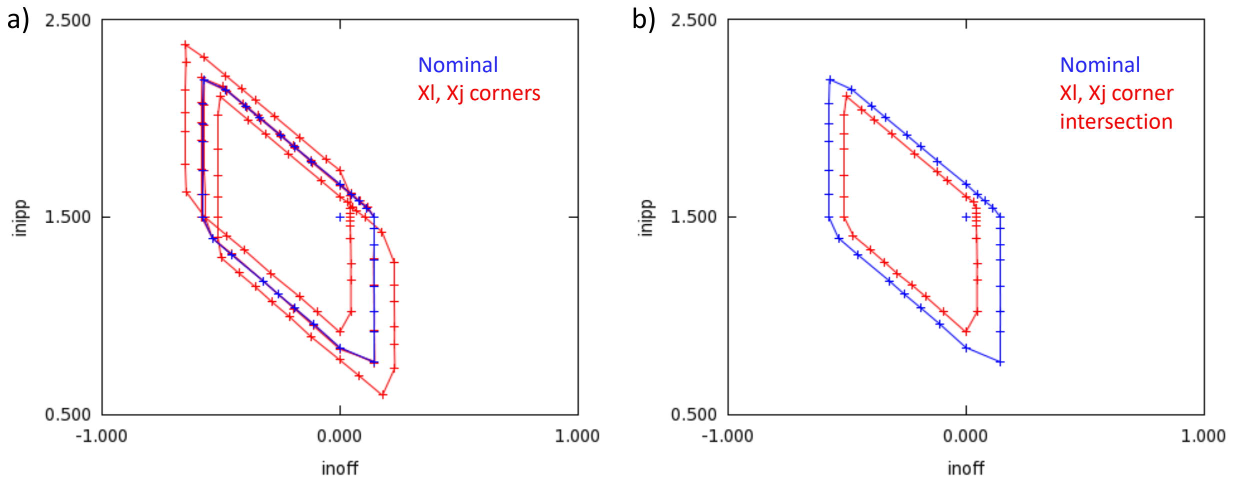

Fig. 5. The operating region of two parameters is shown a) with all other parameters at nominal, and with Xl and Xj set to their parameter corners, each in turn. As shown in b), when corner parameters are specified the intersection of the corner-based operating regions is reported.

Noise and Bit-Error Rate. Margins and yields are calculated in noise-free circuit simulation. The resulting margins correspond to a bit-error-rate of 0.5, whereas large-scale applications may require a bit-error rate per gate of 10^{-24} or even lower. An effective way to prorate the margins and yields for bit-error rate is to 1) measure the bit-error rate to determine the factor by which margins are narrowed between a BER of 0.5 and the desired BER. 2) Scale the sigmas for all parameters up by this factor. This method is shown in Fig. 6.

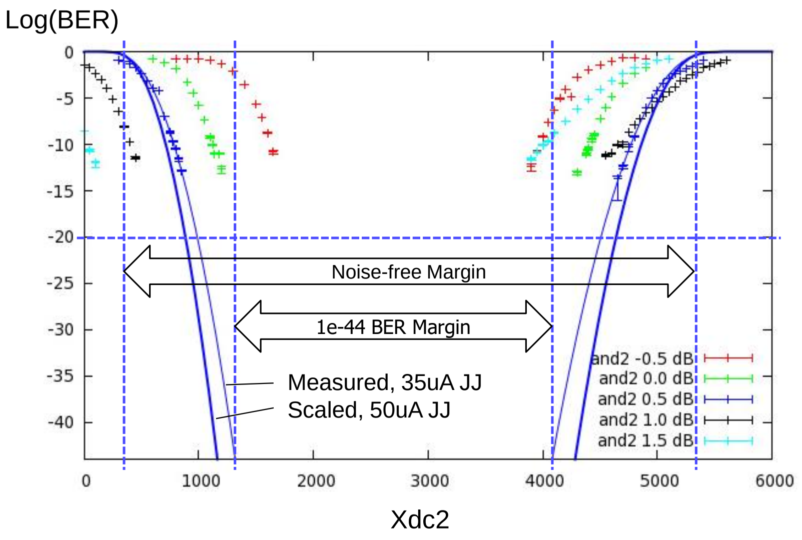

Fig. 6. Measured BER and error-function curve fits are shown for an AND2 gate. The effective margin at a particular BER is always narrower than that given in noise-free simulation, which corresponds to a BER of 0.5. The factor by which margins narrow, F, may be largely independent of the particular knob plotted on the x axis. Yield at a particular BER can be estimated by increasing all device sigmas by a factor of 1/F.

Note that the simplex-based yield estimate method described here can efficiently compute yield for multiple sigma values, so long as they are scaled by a common factor. This only involves reevaluation of the analytic Q function, and does not require addition circuit simulations. Yield as a function of sigma may be prove to be a good fit to the Q function, where both radius and dimension are fitting parameters.

Quick Start Guide

Software. Malt uses itself, WRSpice, gnuplot, the and linux command line. You can type in your own netlist with your favorite text editor or generate it using the native WRSpice schematic editor, but the model-based custom netlister that we have implemented in the Cadence schematic editor environment is highly recommended. All files invoked below are listed out in Appendix 1 and may be included with your malt distribution.

Netlist. Consider the following netlist:

```
netlist.cir
```

which is self-contained except for the definition of some parameter nominal values

```
netlist.nominals
```

and the included model file:

```
my_models_spice.mod
```

which is intended only to serve as a self-contained example for present purposes.

Parameter Definition. The netlist and model file will normally be created by the netlister, but the individual parameters and nominals must be defined by the user by editing the text files.

WRSpice. From the WRSpice command line, type:

```
> source netlist.nominals
> source netlist.cir
> run
> let
> plot v(phi.Xb0.XI1) v(phi.Xb1.XI12)
```

where the "let" command is very useful in figuring out the names of the circuit vectors that you may want to plot. Keep hitting the space bar until you return to the WRSpice prompt.

All these commands and more are included in a WRSpice script. From the WRSpice command line, type:

```
> netlist.run
```

then type

```
> quit
```

to exit.

Malt. If you have the malt executable, invoke it from the command line and it will print the following:

```
MALT 3.0

  Parametric yield optimization utility for use with WRSpice
SYNOPSIS
  malt -d|m|t|2|y|o <circuit_name>[.<ext_1>][.<ext_N>] [--<config.line>]
Options
  -d      Define correct circuit operation
  -m      Calculate individual parameter margins
  -t      Trace nodes at marginal parameter values
  -2      Binary search operating region in 2 dimensions
  -y      Calculate circuit yield using corner analysis
  -o      Optimize yield using simplex approximation
CIRCUIT NAME
  Used to find the .config, .cir, .param, .passf, and .envelope files
```

```
    Name extensions are delimited by periods
    The most specfic .cir, .param, .passf, and .envelope files are used
    The .config files are parsed from most general to most specific
   COMMANDLINE CONFIGURATION OPTIONS
    Lines of a configuration file, delimited by two leading dashes
```

Malt will make use of the netlist file

```
netlist.cir
```

where that .cir file extension is what malt requires for netlist files. The “liberated” parameters, and the node to be checked, are defined in the .config file:

```
netlist.config
```

where the .config file extension is what malt requires for configuration files.

To define correct circuit operation, from the linux command line type:

```
> malt –d netlist
```

and it will 1) plot the nodes at nominal parameter values, 2) plot the envelopes around the nodes that define correct circuit operation, and 3) exit to the WRSpice command line so you can manipulate the plots.

To compute parametric margins, yield, and optimized parameter values, from the linux command line type:

```
> malt –m netlist
> malt –y netlist
> malt –o netlist
```

each in turn. Of course, you could also recompute yield after the optimization by updating the nominal parameter values in the config.

More Malt. Additional features, in no particular order, are as follows.

Slices of the **operating region in two dimensions** are defined in netlist.1.config. To produce the result, from the linux command line type:

```
> malt –2 netlist.1
```

and then:

```
> gnuplot netlist.1.2gnu
```

to view the results. This file is a gnuplot script that can be edited to suit. In particular, changing the pause statements to “ pause -1” will give better interactive results. Two-dimensional margins are a poor substitute for higher-dimensional analysis, but can be useful, particularly for judging the convexity of the operating region.

Some **corner parameters** are defined in netlist.2.config. To recompute margins for this case, type:

```
> malt –m netlist.2
```

and compare the results to that above. The same can be done for yield and optimized parameter values.

A word about **input file precedence** is in order. In the above example, Malt first loads the configuration file netlist.config, then loads the file netlist.2.config, overwriting those fields that are redefined. The final configuration is documented in the file netlist.2.config.m. Malt also looks for the file netist.2.cir, but not finding it, looks for (and finds) the file netlist.cir. All this is described in greater detail in the next section.

The **.param file** can be used to define additional parameters that do mathematical operations on the parameters defined in the netlist. This is demonstrated with the files netlist.3.param and netlist.3.config. Type:

```
> malt –m netlist.3
```

and again compare the results to that above. This example happens to collapse a four parameter corners down to two corners, and also happens to get the same result. In general, any parametric knob the mind can conceive could be achieved here. Use of a separate file is necessary because variables cannot be assigned using mathematical expressions within the netlist.cir file.

The trace routine is used to **visualize circuit behavior at the operating boundary** corresponding to individual parameter margins. For each parameter margin, the waveforms for each node are plotted just inside and just outside the operating boundary. Type:

```
> malt –t netlist.4
```

to generate the plots associated with the reduced set of parameters specified in netlist.4.config. To view the plots, from the wrspice prompt, type:

```
> netlist.4.plot.t
```

which is the generated script written in WRSpice syntax. Delete the multiple *.pass and *.fail files that were generated, to reduce cruft when you are done.

Arbitrary criteria for correct circuit operation can be specified in the **.passf file** in the event that the automatically-generated envelopes are inadequate. An example is given in the files netlist.5.passf and netlist.5.config, dealing with the superconducting-phase waveform generated by the output amplifier. This is illustrated in Fig. 7. Type:

```
> malt –d netlist.5
> malt –m netlist.5
```

to compute margins including these new criteria. Note that only nodes listed in the .config can be referenced by the .passf file, as no other nodes are saved.

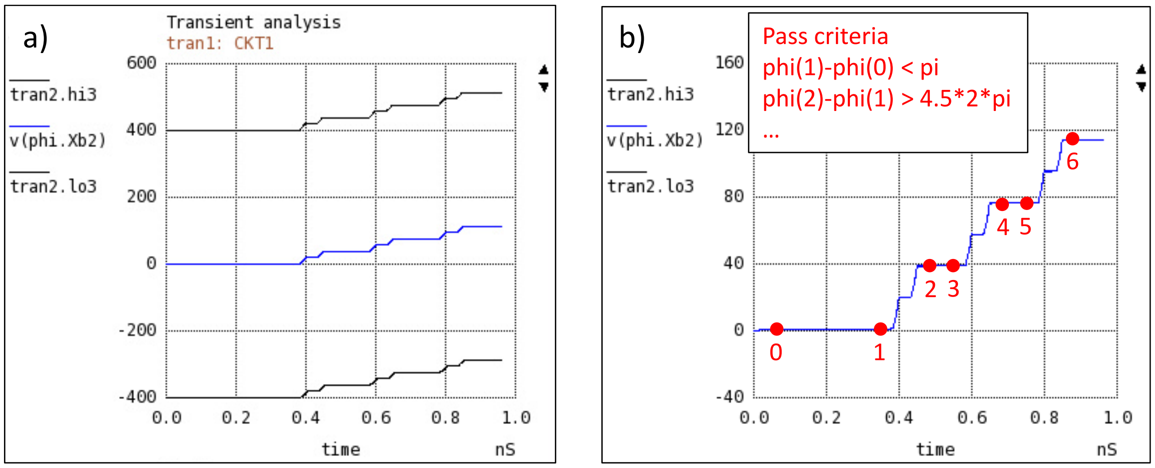

Fig. 7. The phase (time-integral of voltage) waveform of the output is not amenable to envelope definition. a) Instead, the amplitude tolerance on the envelope is set to values that cannot fail. b) The .passf file in this example defines pass criteria in terms of minimum and maximum phase excursions between adjacent checkpoints, as illustrated.

Input and Output File Names

File types are indicated by the file extension.

Configuration file.

- <circuit_name>.config

This file, or file hierarchy, defines the circuit parameters to be analyzed, the nodes to be monitored, and the options for the various analysis types. It cannot seem to find any of the malt.config files.

Circuit file.

- <circuit_name>.cir

This contains the WRSpice circuit netlist, marked up with the parameter variables as needed.

Pass/fail file.

- <circuit_name>.passf

This is an optional file that set pass/fail criteria for correct circuit operation, written in WRSpice control code syntax.

Circuit name extensions.

- <circuit_name>[.<ext_1>][.<ext_N>]

Name extensions, delimited by periods, can be added to the circuit names, allowing for partial updates to the input files. The most specific cir, passf, and (auto-generated) envelope files will be used.

The config files are handled a little differently, as they are parsed in order from most general to most specific, with overwrites occurring where fields are redefined. The final overwrite occurs for option fields specified at the command line. Note that default settings are defined in generically-named file called "malt.config". This file will be auto-generated if it does not exist within the file tree.

Final configuration file.

- <circuit_name>[.<ext_1>][.<ext_N>].config.d|m|t|2|s|y|o

The file configuration is written to the file using the circuit name with all extensions, and the analysis type appended.

Examples

Input files	Malt command	Config hierarchy	Final configuration
jtl.config Jtl.cir	malt –m jtl	malt.config jtl.config	jtl.config.m
jtl.config jtl.2.config jtl.cir	malt –m jtl.2	malt.config jtl.config jtl.2.config	jtl.config.2.m
jtl.config jtl.2.cir	malt –m jtl.2	malt.config jtl.config	jtl.config.2.m
jtl.config jtl.2.cir	malt –d jtl.2 --simulate=0	malt.config jtl.config command line	jtl.config.2.d

Iterate file.

- <circuit_name>[.<ext_1>][.<ext_N>].iterate.2|s|y|o

To interrupt analysis iterations cleanly during runtime, remove the associated iterate file. This applies to the analysis types indicated on the line above.

Output file.

- <circuit_name>.d|m|2|s|y|o

An output file is generated for each analysis type. Additional files are generated for the d, t, and 2 analysis types, as described in the next section.

Options & Outputs

General Options

Accuracy of the binary search.

- binsearch_accuracy=<float>

This number is given as a fraction of sigma for each parameter.

Name of the WRSpice executable.

- spice_call_name=<string>

Normally, "wrspice".

Send all the messages generated by spice to the terminal, yes/no.

- verbose=<int>

Not particularly pleasant, but useful for debug.

Echo output to the terminal, yes/no.

- print_terminal=<int>

Output is printed to the output file in any case.

Define correct circuit operation (-d)

Run the simulation at nominal parameter values, plot the node waveforms and generate envelopes for the waveforms that define correct circuit operation.

Node names that will be saved and monitored. The dt and dx can be included on the same line or listed separately.

- node=<string> , dt=<float>, dx=<float>

The pass/fail criteria for correct circuit operation is defined with envelopes on the waveforms for the listed nodes. The envelope is defined by tracing the waveform with an ellipse, whose time axis is defined by dt and whose amplitude axis is defined by dx. A single set of dt and dx parameters can be applied to all nodes if listed separately, in advance of the nodes.

Time-wise tolerance of the pass/fail envelope.

- dt=<float>

Amplitude-wise tolerance of the pass/fail envelope.

- dx=<float>

Run the nominal circuit simulation yes/no.

- d_simulate=<int>

Normally you need to run the simulation to in order to construct the envelopes. However, if you want to re-construct the envelopes using the same waveforms (but presumably different dt and dx), you can skip the simulation.

Save the generated envelopes, yes/no.

- d_envelope=<int>

Normally, yes, but if you want to try out new envelopes without overwriting the old ones, then no.

The following supplementary output files will be created.

- <circuit_name>.envelope
- <circuit_name>.plot.d

The first file contains the vector data, for internal use only. The second file is a WRSpice script; invoke this from the WRSpice command line to review the waveforms and envelopes.

Alternative to define (.passf)

A hand-written file with a .passf extension can be used to define correct circuit operation instead of, or in conjunction with, the envelope definition. The file is written in wrspice control-code syntax. It is intended to assess node values against some criteria, and set the flag passf to 1 or 0 accordingly, to indicate pass or fail.

Calculate individual parameter margins (-m)

Calculate individual parameter margins using a binary search. Uses the pass/fail criteria defined above. In all binary-search-based routines here and below, normal parameters are margined, while corner parameters are set to the corner values.

Parameters.

- param=<string>, nominal=<float>, min=<float>, max=<float>, sigma=<float>, sig_abs=<float>, include=<int>, logs=<int>, corner=<int>
    - String is the parameter variable name.
    - Nominal is the nominal parameter value.
    - Max and Min are the upper and lower bounds on the binary search.
    - Sigma is the 1-sigma parameter spread as a percentage of nominal. Sig_abs is given as an absolute value. Just one of the two must be set to non-zero.
    - Include: Include the parameter in the analysis yes/no. If no, set its value to nominal and leave it alone.
    - Corner: The parameter is set to the corner values (specified by sigma), as opposed to being included in the binary search, yes/no.
    - Logs: The parameter is defined as having its distribution in log-space, yes/no.

Trace nodes at marginal parameter values (-t)

Generates plots for each node on both sides of the operating boundary for each parameter. Useful for debugging the envelope generation, the parameters margins, or both. The total number of plots is potentially large, scaling as the product of the number of nodes and the number of included parameters.

The options used are those listed for parameter margins analysis.

The following supplementary output files will be created.

- <circuit_name>.<param>.<max|min>.<pass|fail>
- <circuit_name>.plot.t

The first file contains the vector data, for internal use only. The second file is a WRSpice script. Invoke the script from the WRSpice command line to review the waveforms and envelopes.

Binary search operating region in 2 dimensions (-2)

Plot points on the operating boundary in two dimensions using the binary search.

Parameters (see above).

- param=<string>, nominal=<float>, min=<float>, max=<float>, sigma=<float>, sig_abs=<float>, include=<int>, corner=<int>, logs=<string>

Defines the two parameters for the 2D margin run. You can have multiple lines

- param_x=<string>, param_y=<string>

Number of points to generate that define the operating region

- 2D_iter=<int>

The following supplementary output files will be created.

- netlist2.2
- netlist2.2gnu

The first file contains the margin data, for internal use only. The second file is a gnuplot script. Invoke the script from the gnuplot command line to generate the plots. You can edit the script to suit your tastes.

Calculate circuit yield using corner analysis (-y)

Evaluate parametric yield using an adaptive algorithm following an anneal schedule. 1) Calculate 2^N to 3^N margins in N-space where N is the total number of included parameters. Corner parameters are set to the corner values, while normal parameters are margined with a binary search. 2) Continue further until the desired accuracy is achieved. Control parameters are defined below.

Parameters.

- param=<string>, nominal=<float>, min=<float>, max=<float>, sigma=<float>, sig_abs=<float>, include=<int>, corner=<int>, logs=<string>

Step (1) defined above proceeds for 2^N iterations if depth=0, and 3^N iterations if depth=10. Range: 0-10. Default: 5.

- y_search_depth=<int>

Step (1) defined above initially partitions space into equal solid angles if width=9, into equal products of angle and function value if width=0. A larger width value is more exploratory across space, and a smaller value is more adaptive based on existing information. Range: 0-9. Default: 5.

- y_search_width=<int>

Step (1) becomes more adaptive in a certain number of steps. Inherently uninteresting. Range: 1-40. Default: 12.

- y_search steps=<int>

Maximum amount of estimated memory that can be allocated in step (2). This may be an underestimate of true system memory usage. Default: 4194304.

- y_max_mem_k=<int>

Estimated percent accuracy to in order to terminate step (2). Default: 10.

- y_accurancy=<int>

Print every iteration, else print only running most significant iteration. Default: 0.

- y_print_every=<int>

Optimize yield using simplex approximation (-o)
Center the parameter values within a simplex that approximates the operating region. As above, normal parameters are margined, while corner parameters are set to the corner values. Options for controlling the iterations are listed below.

Parameters.

- param=<string>, nominal=<float>, min=<float>, max=<float>, sigma=<float>, sig_abs=<float>, static=<int>, include=<int>, corner=<int>, logs=<string>, nom_min=<float>, nom_max=<float>

Limits on the optimized nominal parameter values.

- Nom_max and nom_min can be set to limit the range of the optimized parameter values.

Maximum number of iterations. Hard stop on this number.

- o_max_iter=<int>

Minimum number of iterations. Analysis will terminate after the last this-many iterations fail to improve the margins by more than y_accuracy.

- o_min_iter=<int>

Maximum number of simplex facets. Hard stop on this number.

- o_max_planes=<int>

Appendix 1: Listing of all the files

See the directory

AA_fluxshut_top_4malt_clean.

netlist.cir

```
* HNL Generated netlist of AA_fluxshut_top
* GLOBAL Parameters
.param amplitude = 0.700  frequency = 10G  m_in = 0.1p
* GLOBAL Scaling Parameters
*.param Xac=1.0
*.param Xlcomp=1.0
*.param Xjcomp=1.0
.param Xl=1.0
.param Xj=1.0
*.param Xpdc=1.0
.param Xa=1.0
.param Xr=1.0
* Timing corner definitions
.param timingX = 1.0
.global 0
* Include Statements
.include my_models_spice.mod
*Subcircuits
.subckt aa_fluxshut a i0 i1 q
XL0 a i0 rql_inductor_scale l=7.4e-12
Loff i0 i1 'm_in'
XL2 i1 net04 rql_inductor_scale l=1.47e-11
XL1 net04 q rql_inductor_scale l=7.4e-12
Xb0 i0 net023 rql_rsj_scale jjmod=rql25 ic=0.07 icrn=1
Xb1 net04 net08 rql_rsj_scale jjmod=rql25 ic=0.07 icrn=1
Lg0 net023 0 1e-12
Lg1 net08 0 1e-12
.ends aa_fluxshut
.subckt g_terminate i ic1=0.050
XR0 i 0 rql_resistor_scale r='0.7/ic1'
.ends g_terminate
*Input stage
L1 na 0 'm_in'
XL0 net017 na rql_inductor_scale l='1e-12*l0'
Xb0 net015 net017 rql_rsj_scale jjmod=rql25 ic='b0' icrn=1
*Flux shuttle stages
XI1 net015 i1a i1b net06 aa_fluxshut
XI2 net06 i2a i2b net07 aa_fluxshut
XI3 net07 i3a i3b net08 aa_fluxshut
XI4 net08 i4a i4b net05 aa_fluxshut
XI5 net05 i5a i5b net09 aa_fluxshut
XI6 net09 i6a i6b net010 aa_fluxshut
XI7 net010 i7a i7b net03 aa_fluxshut
XI8 net03 i8a i8b net029 aa_fluxshut
XI9 net029 i9a i9b net018 aa_fluxshut
XI10 net018 i10a i10b net025 aa_fluxshut
XI11 net025 i11a i11b net020 aa_fluxshut
XI12 net020 i12a i12b net02 aa_fluxshut
*flux shuttle termination
```

```
Xb1 net02 net04 rql_rsj_scale jjmod=rql25 ic='b1' icrn=1
Lg1 net04 0 1e-12
XI13 net02 g_terminate ic1=0.07
*output amp
XL2 net031 net030 rql_inductor_scale l=1e-12
XL3 net032 0 rql_inductor_scale l=1e-12
XL4 net029 net026 rql_inductor_scale l='1e-12*l45'
XL5 net024 net025 rql_inductor_scale l='1e-12*l45'
XL6 net030 oo rql_inductor_scale l=4e-10
XR0 net030 oo rql_resistor_scale r=50
Xb2 net030 net032 rql_rsj_scale jjmod=rql25 ic='b2' \
  icrn='1.0*b2*0.07/(b2*0.07+b34*0.040)'
Xb3 net026 net031 phib3 rql_junction_scale area='b34' jjmod=rql25
Xb4 net024 net031 phib4 rql_junction_scale area='b34' jjmod=rql25
*Input source
Ia 0 na 'phi0/m_in*inoff' +\
pulse(0  'phi0/m_in*inipp' 170ps  20ps  20ps 80ps 1) +\
pulse(0  'phi0/m_in*inipp' 370ps  20ps  20ps 80ps 1) +\
pulse(0  'phi0/m_in*inipp' 570ps  20ps  20ps 80ps 1) +\
0
*Four phase clock sources
I1 i1a i1b sin(0 'phi0/m_in*Xac*amplitude' 10GHz 00ps)
I2 i2a i2b sin(0 'phi0/m_in*Xac*amplitude' 10GHz 25ps)
I3 i3a i3b sin(0 'phi0/m_in*Xac*amplitude' 10GHz 50ps)
I4 i4a i4b sin(0 'phi0/m_in*Xac*amplitude' 10GHz 75ps)
I5 i5a i5b sin(0 'phi0/m_in*Xac*amplitude' 10GHz 00ps)
I6 i6a i6b sin(0 'phi0/m_in*Xac*amplitude' 10GHz 25ps)
I7 i7a i7b sin(0 'phi0/m_in*Xac*amplitude' 10GHz 50ps)
I8 i8a i8b sin(0 'phi0/m_in*Xac*amplitude' 10GHz 75ps)
I9 i9a i9b sin(0 'phi0/m_in*Xac*amplitude' 10GHz 00ps)
I10 i10a i10b sin(0 'phi0/m_in*Xac*amplitude' 10GHz 25ps)
I11 i11a i11b sin(0 'phi0/m_in*Xac*amplitude' 10GHz 50ps)
I12 i12a i12b sin(0 'phi0/m_in*Xac*amplitude' 10GHz 75ps)
*Output source
Iout 0 oo pwl(0 0 20ps 'Xpdc*80uA')
Rout 0 oo 50
*Analysis definition
.tran 1ps 1000ps 0ps uic
.end
```

my_models_spice.mod

```
* MODEL Declarations
*unshunted JJ with scalable parameters
.subckt rql_junction_scale PLUS MINUS PHI area=0.25 jjmod=rql25
b1 PLUS MINUS phi jjmod area='area*Xa*Xj*Xjcomp/timingX'
.ends rql_junction_scale
* rsj with scalable parameters
.subckt rql_rsj_scale PLUS MINUS jjmod=rql25 ic=0.25 icrn=0.7
Xb0 PLUS MINUS phi rql_junction_scale area='ic' jjmod='jjmod'
Xr0 PLUS MINUS rql_resistor_scale r='icrn/ic'
.ends rql_rsj_scale
* inductor models with scalable parameters
.subckt rql_inductor_scale PLUS MINUS l=1p
L0 PLUS MINUS 'l*Xl*Xlcomp/timingX'
.ends rql_inductor_scale
```

```
* resistor models with scalable parameters
.subckt rql_resistor_scale PLUS MINUS r=1
R0 PLUS MINUS 'r*Xr*timingX'
.ends rql_resistor_scale
*JJ models
.model rql25 jj(rtype=1,cct=1,icon=10m,vg=2.6m,delv=0.1m, icrit=1m,r0=40,rn=1.800,cap=0.70p)
* End MODEL Declarations
```

netlist.run

```
*netlist.run
.control
source netlist.nominals
source netlist.cir
run
plot \
v(phi.Xb0) \
v(phi.Xb0.XI1) \
v(phi.Xb1.XI1) \
v(phi.Xb0.XI12) \
v(phi.Xb1.XI12)
set noaskquit
.endc
```

netlist.nominals

```
*netlist
.control
*Parameter values
Xlcomp=1.0
Xjcomp=1.0
Xac   =1.0
Xpdc  =1.0
inoff =0.0
inipp =1.0
b0    =0.080
l0    =5.0
b1    =0.018
b2    =0.070
b34   =0.040
l45   =30.0
.endc
```

netlist.config

```
* Global parameters
param=Xlcomp, nominal=1.0,   min=0.30,  max=3.0,   sigma=4.0, logs=1, corners=1, include=0
param=Xjcomp, nominal=1.0,   min=0.30,  max=3.0,   sigma=4.0, logs=1, corners=1, include=0
param=Xac,    nominal=1.0,   min=0.30,  max=1.7,   sigma=4.0, logs=0, corners=0, include=1
param=Xpdc,   nominal=1.0,   min=0.30,  max=1.7,   sigma=4.0, logs=0, corners=0, include=1,
nom_min=1, nom_max=1
* Individual Variables
*Input waveform offset and peak-to-peak amplitude in units of Phi0
param=inoff,  nominal=0.00,  min=-1.0,  max=1.0, sig_abs=0.04,logs=0, corners=0, include=1
param=inipp,  nominal=1.00,  min=0.0,   max=3.0, sig_abs=0.04,logs=0, corners=0, include=1
*Input parameters
param=b0,     nominal=0.080, min=0.030, max=0.30,  sigma=4.0, logs=1, corners=0, include=1
```

```
param=l0,     nominal=5.00,  min=1.50,  max=15.0,  sigma=4.0, logs=1, corners=0, include=1
*Output parameter
param=b1,     nominal=0.018, min=0.006, max=0.060, sigma=4.0, logs=1, corners=0, include=0
param=b2,     nominal=0.070, min=0.020, max=0.200, sigma=4.0, logs=1, corners=0, include=0
param=b34,    nominal=0.040, min=0.012, max=0.120, sigma=4.0, logs=1, corners=0, include=0
param=l45,    nominal=30.0,  min=10,    max=100,   sigma=4.0, logs=1, corners=0, include=0
*** General Option ***
binsearch_accuracy=0.1
***  Options for Corners Yield  ***
***  ranges: 0-10, 0-9, 1-40
y_search_depth = 5
y_search_width  = 5
y_search_steps  = 12
y_max_mem_k = 4194304
y_accuracy = 10
y_print_every = 0
* Nodes for the Passfail criterion
dx=2
dt=40e-12
node=v(phi.Xb0)
node=v(phi.Xb0.XI1)
node=v(phi.Xb1.XI12)
```

netlist.1.config

```
* Global parameters
param=Xlcomp, nominal=1.0,   min=0.30,  max=3.0,   sigma=4.0, logs=1, corners=0, include=1
param=Xjcomp, nominal=1.0,   min=0.30,  max=3.0,   sigma=4.0, logs=1, corners=0, include=1
param=Xac,    nominal=1.0,   min=0.30,  max=1.7,   sigma=4.0, logs=0, corners=0, include=1
param=Xpdc,   nominal=1.0,   min=0.30,  max=1.7,   sigma=4.0, logs=0, corners=0, include=1,
nom_min=1, nom_max=1
* Individual Variables
*Input waveform offset and peak-to-peak amplitude in units of Phi0
param=inoff,  nominal=0.00,  min=-1.0,  max=1.0, sig_abs=0.04,logs=0, corners=0, include=1
param=inipp,  nominal=1.00,  min=0.0,   max=3.0, sig_abs=0.04,logs=0, corners=0, include=1
*Input parameters
param=b0,     nominal=0.080, min=0.030, max=0.30,  sigma=4.0, logs=1, corners=0, include=1
param=l0,     nominal=5.00,  min=1.50,  max=15.0,  sigma=4.0, logs=1, corners=0, include=1
*Output parameter
param=b1,     nominal=0.018, min=0.006, max=0.060, sigma=4.0, logs=1, corners=0, include=0
param=b2,     nominal=0.070, min=0.020, max=0.200, sigma=4.0, logs=1, corners=0, include=0
param=b34,    nominal=0.040, min=0.012, max=0.120, sigma=4.0, logs=1, corners=0, include=0
param=l45,    nominal=30.0,  min=10,    max=100,   sigma=4.0, logs=1, corners=0, include=0
*** General Option ***
binsearch_accuracy=0.1
***  2D Margins  ***
2D_iter = 32
param_x = Xlcomp, param_y = Xjcomp
param_x = inoff,  param_y = inipp
* Nodes for the Passfail criterion
dx=2
dt=40e-12
node=v(phi.Xb0)
node=v(phi.Xb0.XI1)
node=v(phi.Xb1.XI12)
```

netlist.2.config

```
* Global parameters
param=Xlcomp, nominal=1.0,   min=0.30,  max=3.0,   sigma=4.0, logs=1, corners=1, include=1
param=Xjcomp, nominal=1.0,   min=0.30,  max=3.0,   sigma=4.0, logs=1, corners=1, include=1
param=Xac,    nominal=1.0,   min=0.30,  max=1.7,   sigma=4.0, logs=0, corners=0, include=1
param=Xpdc,   nominal=1.0,   min=0.30,  max=1.7,   sigma=4.0, logs=0, corners=0, include=1,
nom_min=1, nom_max=1
* Individual Variables
*Input waveform offset and peak-to-peak amplitude in units of Phi0
param=inoff,  nominal=0.00,  min=-1.0,  max=1.0, sig_abs=0.04,logs=0, corners=0, include=1
param=inipp,  nominal=1.00,  min=0.0,   max=3.0, sig_abs=0.04,logs=0, corners=0, include=1
*Input parameters
param=b0,     nominal=0.080, min=0.030, max=0.30,  sigma=4.0, logs=1, corners=0, include=1
param=l0,     nominal=5.00,  min=1.50,  max=15.0,  sigma=4.0, logs=1, corners=0, include=1
*Output parameter
param=b1,     nominal=0.018, min=0.006, max=0.060, sigma=4.0, logs=1, corners=0, include=0
param=b2,     nominal=0.070, min=0.020, max=0.200, sigma=4.0, logs=1, corners=0, include=0
param=b34,    nominal=0.040, min=0.012, max=0.120, sigma=4.0, logs=1, corners=0, include=0
param=l45,    nominal=30.0,  min=10,    max=100,   sigma=4.0, logs=1, corners=0, include=0
*** General Option ***
binsearch_accuracy=0.1
***  Options for Corners Yield  ***
***  ranges: 0-10, 0-9, 1-40
y_search_depth = 5
y_search_width  = 5
y_search_steps  = 12
y_max_mem_k = 4194304
y_accuracy = 10
y_print_every = 0
* Nodes for the Passfail criterion
dx=2
dt=40e-12
node=v(phi.Xb0)
node=v(phi.Xb0.XI1)
node=v(phi.Xb1.XI12)
```

netlist.3.config

```
* Global parameters
*these are defined in the .param file now:
*param=Xlcomp, nominal=1.0,   min=0.30,  max=3.0,   sigma=4.0, logs=1, corners=1, include=1
*param=Xjcomp, nominal=1.0,   min=0.30,  max=3.0,   sigma=4.0, logs=1, corners=1, include=1
param=Xljcomp, nominal=1.0,   min=0.30,  max=3.0,   sigma=4.0, logs=1, corners=1, include=1
param=Xac,    nominal=1.0,   min=0.30,  max=1.7,   sigma=4.0, logs=0, corners=0, include=1
param=Xpdc,   nominal=1.0,   min=0.30,  max=1.7,   sigma=4.0, logs=0, corners=0, include=1
* Individual Variables
*Input waveform offset and peak-to-peak amplitude in units of Phi0
param=inoff,  nominal=0.00,  min= -1.0, max=1.0,   sig_abs=0.04, logs=0, corners=0, include=1
param=inipp,  nominal=1.00,  min= 0.0,  max=3.0,   sig_abs=0.04, logs=0, corners=0, include=1
*Input parameters
param=b0,     nominal=0.080, min=0.030, max=0.30,  sigma=4.0, logs=1, corners=0, include=1
param=l0,     nominal=5.00,  min=1.50,  max=15.0,  sigma=4.0, logs=1, corners=0, include=1
*Output parameter
param=b1,     nominal=0.018, min=0.006, max=0.060, sigma=4.0, logs=1, corners=0, include=0
param=b2,     nominal=0.070, min=0.020, max=0.200, sigma=4.0, logs=1, corners=0, include=0
param=b34,    nominal=0.040, min=0.012, max=0.120, sigma=4.0, logs=1, corners=0, include=0
```

```
param=l45,    nominal=30.0,  min=10,    max=100,   sigma=4.0, logs=1, corners=0, include=0
*** General Option ***
binsearch_accuracy=0.1
* Nodes for the Passfail criterion
dx=2
dt=40e-12
node=v(phi.Xb0)
node=v(phi.Xb0.Xl1)
node=v(phi.Xb1.Xl12)
```

netlist.3.param

```
* define some parameters
.control
Xlcomp=Xljcomp
Xjcomp=Xljcomp
.endc
```

netlist.4.config

```
* Global parameters
param=Xlcomp, nominal=1.0,   min=0.30,  max=3.0,   sigma=4.0, logs=1, corners=1, include=1
param=Xjcomp, nominal=1.0,   min=0.30,  max=3.0,   sigma=4.0, logs=1, corners=1, include=1
param=Xac,    nominal=1.0,   min=0.30,  max=1.7,   sigma=4.0, logs=0, corners=0, include=0
param=Xpdc,   nominal=1.0,   min=0.30,  max=1.7,   sigma=4.0, logs=0, corners=0, include=0
* Individual Variables
*Input waveform offset and peak-to-peak amplitude in units of Phi0
param=inoff,  nominal=0.00,  min= -1.0, max=1.0,   sig_abs=0.04, logs=0, corners=0, include=1
param=inipp,  nominal=1.00,  min= 0.0,  max=3.0,   sig_abs=0.04, logs=0, corners=0, include=1
*Input parameters
param=b0,     nominal=0.080, min=0.030, max=0.30,  sigma=4.0, logs=1, corners=0, include=0
param=l0,     nominal=5.00,  min=1.50,  max=15.0,  sigma=4.0, logs=1, corners=0, include=0
*Output parameter
param=b1,     nominal=0.018, min=0.006, max=0.060, sigma=4.0, logs=1, corners=0, include=0
param=b2,     nominal=0.070, min=0.020, max=0.200, sigma=4.0, logs=1, corners=0, include=0
param=b34,    nominal=0.040, min=0.012, max=0.120, sigma=4.0, logs=1, corners=0, include=0
param=l45,    nominal=30.0,  min=10,    max=100,   sigma=4.0, logs=1, corners=0, include=0
*** General Option ***
binsearch_accuracy=0.1
* Nodes for the Passfail criterion
dx=2
dt=40e-12
node=v(phi.Xb0)
node=v(phi.Xb0.Xl1)
node=v(phi.Xb1.Xl12)
```

netlist.5.config

```
* Global parameters
param=Xlcomp, nominal=1.0,   min=0.30,  max=3.0,   sigma=4.0, logs=1, corners=1, include=0
param=Xjcomp, nominal=1.0,   min=0.30,  max=3.0,   sigma=4.0, logs=1, corners=1, include=0
param=Xac,    nominal=1.0,   min=0.30,  max=1.7,   sigma=4.0, logs=0, corners=0, include=1
param=Xpdc,   nominal=1.0,   min=0.30,  max=1.7,   sigma=4.0, logs=0, corners=0, include=1

* Individual Variables
*Input waveform offset and peak-to-peak amplitude in units of Phi0
param=inoff,  nominal=0.00,  min= -1.0, max=1.0,   sig_abs=0.04, logs=0, corners=0, include=1
```

```
param=inipp,  nominal=1.00,  min= 0.0,  max=3.0,   sig_abs=0.04, logs=0, corners=0, include=1
*Input parameters
param=b0,     nominal=0.080, min=0.030, max=0.30,  sigma=4.0, logs=1, corners=0, include=1
param=l0,     nominal=5.00,  min=1.50,  max=15.0,  sigma=4.0, logs=1, corners=0, include=1
*Output parameter
param=b1,     nominal=0.018, min=0.006, max=0.060, sigma=4.0, logs=1, corners=0, include=1
param=b2,     nominal=0.070, min=0.020, max=0.200, sigma=4.0, logs=1, corners=0, include=1
param=b34,    nominal=0.040, min=0.012, max=0.120, sigma=4.0, logs=1, corners=0, include=1
param=l45,    nominal=30.0,  min=10,    max=100,   sigma=4.0, logs=1, corners=0, include=1

*** General Option ***
binsearch_accuracy=0.1

* Nodes for the Passfail criterion
dx=2
dt=40e-12
node=v(phi.Xb0)
node=v(phi.Xb0.XI1)
node=v(phi.Xb1.XI12)
node=v(phi.Xb2), dx=400
```

netlist.5.passf

```
* first line is assumed to be a comment
.control
*zero indexed array. indexed with itime below
*units are ps as defined by the .tran line in the .cir file
compose myt values 90 344 490 544 690 744 890
* minimum phase advance when it is supposed to be on is 4.5*2*pi=28.3 rad
itime=2
dowhile itime < 7
jtime=itime-1
if (v(phi.Xb2)[$&myt[$&itime]]-v(phi.Xb2)[$&myt[$&jtime]]) < 28.3
  echo Generated not enough phase in range $&myt[$&jtime] to $&myt[$&itime]
  failed=1
end
itime=itime+2
end
* maximum phase advance when it is supposed to be off is 3 rad
itime=1
dowhile itime < 6
jtime=itime-1
if (v(phi.Xb2)[$&myt[$&itime]]-v(phi.Xb2)[$&myt[$&jtime]]) > 3
  echo Generated too much phase in range $&myt[$&jtime] to $&myt[$&itime]
  failed=1
end
itime=itime+2
end
.endc
```