A B **C**
D E
F G H I
J K L M
N O P Q
R S T U
V W X Y
Z

An Efficient Solution to Simulate Mixed-Signal Circuits in **C**

Yves Leduc — *Associate Member, Polytech Sophia*
Jacques Mequin — *EDA R&D Consultant*
Xavier Albinet — *EM Microelectronics, Marin*

Agenda

Introduction, the landscape and the methods
...
…
The basics of the NAPA compiler and simulator
...
Hierarchy, cells and data cells
...
C Resources and functions
User functions
Parameters, values or addresses
...
Tools and synchronization
...
Cell generators
Transfer functions, step and impulse responses
...
Simulation in '*Z*' domain
...
Simulation in '*s*' domain, the SARC engine
...
Full scale simulations
...
Additional description of important resources
A list of the instructions, nodes, user functions...

A few references and conclusion

Example, modeling a SWC 'multiply by 2' cell
Example, a simulation of a ΣΔ SWC modulator

Example, the random walk

Example, a 2nd order resonator software solution

Example, an offset compensated SWC integrator

Example, synchronization of FFT and TSNR

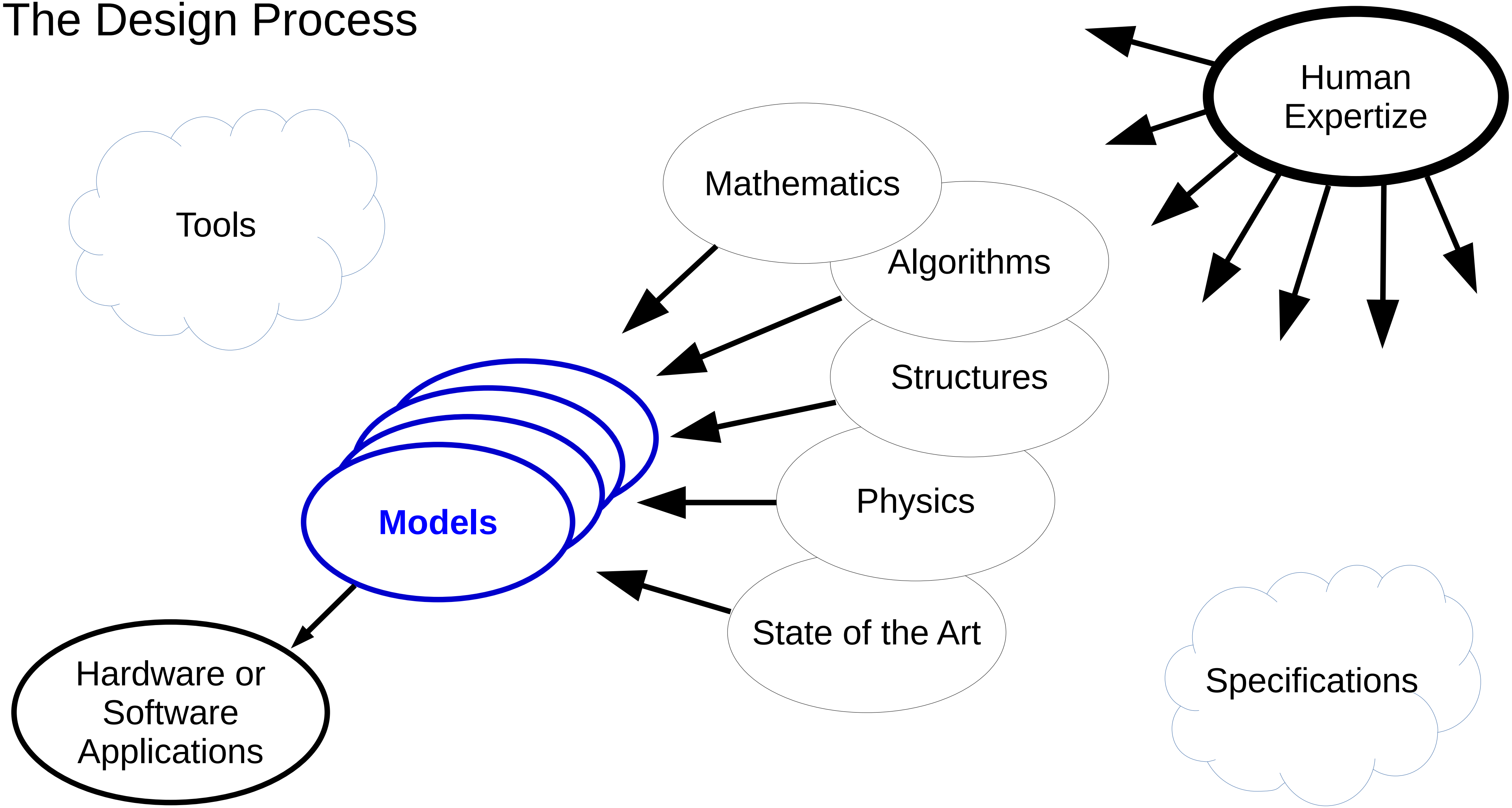
Example, a digital IIR filter

Example, a cascade of 3 analog biquadratic filters

Example, a 1st order ΣΔ SWC modulator

Example, a 3rd order ΣΔ SWC modulator

The Design Process



Modeling is Essential in the Design of a Circuit.

Successive high level abstractions are key for an efficient **Divide and Conquer** strategy.

Modeling creates a structure like the multi-layers of an onion. Each layer brings
new informations about the circuit under design.

Peeling onions makes me cry.
Modeling should be less stressing.

"All models are WRONG, but some are USEFUL"


https://commons.wikimedia.org/wiki/File:GeorgeEPBox.jpg

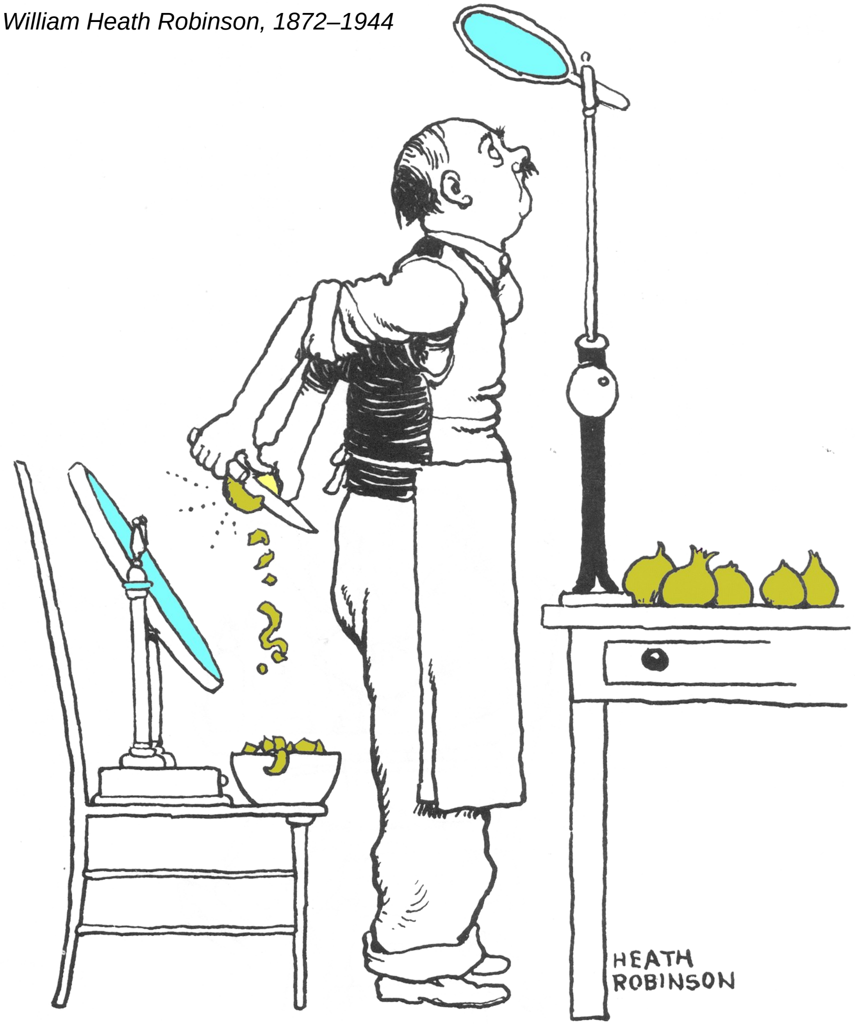
a citation attributed to late George E.P. Box.

George E.P. Box, Statistician, 1919 - 2013
Design of experiments, Bayesian statistics, Time series…

ICI,
Princeton University,
University of Wisconsin–Madison

Contributions: *Response-surface methodology,*
Box–Jenkins method,
Box–Cox transformation …

William Heath Robinson, 1872–1944

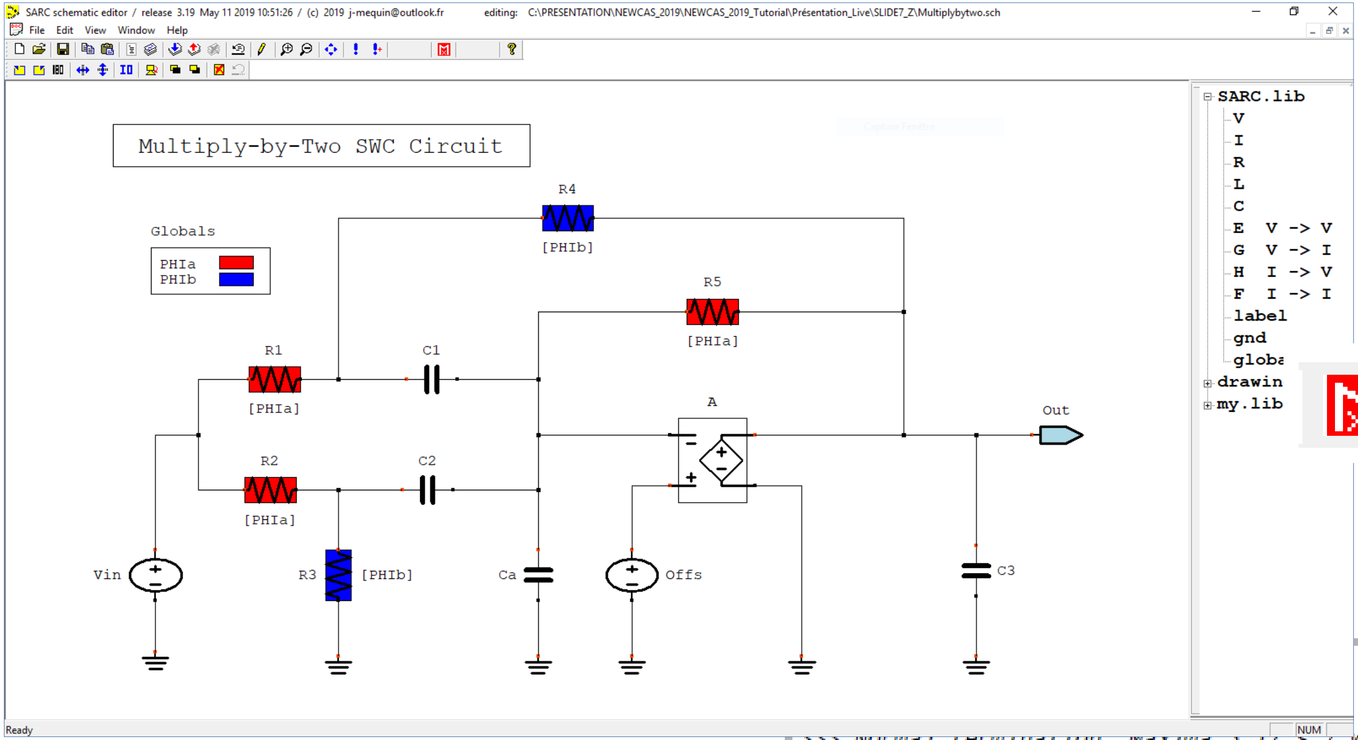


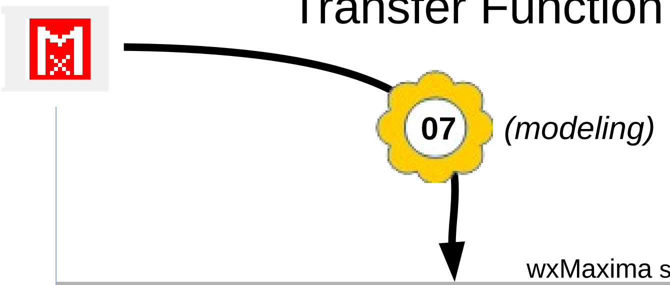
How to Avoid Tears when Peeling Onions

Happily, we have a few solutions to produce models without tears

Modeling in 'Z' of a SWC Circuit

Schematic Editor





wxMaxima screen

wall 5.22 s [95.43,98.39]%

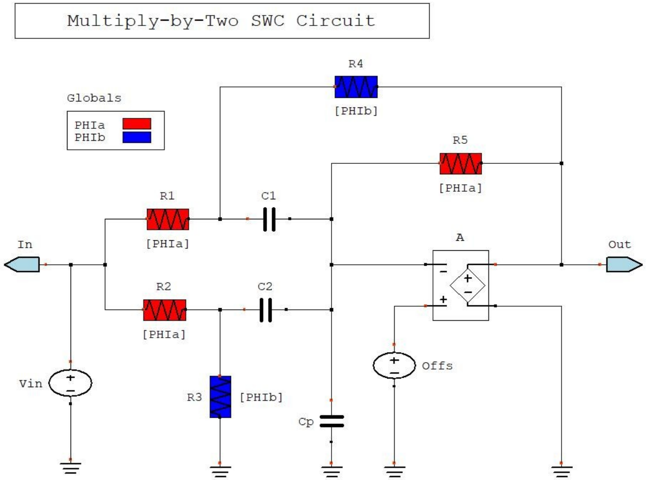
$$\text{*****} \qquad XFER = \frac{(A\,Ca + A\,C2 + A\,C1)\,Offs + ((A^2 + A)\,C2 + (A^2 + A)\,C1)\,Vin}{(A+1)\,Ca + (A+1)\,C2 + (A^2 + 2\,A + 1)\,C1}$$

$$\text{*****} \quad [\text{ No offset }] \quad XFER = \frac{(A\,C2 + A\,C1)\,Vin}{Ca + C2 + (A+1)\,C1}$$

$$\text{*****} \quad [\text{ A infinite }] \quad XFER = \frac{(C2 + C1)\,Vin}{C1}$$

This simple model comforts our understanding and prepares to exhaustive simulations.

Modeling in Z Domain per Phase or per Cycle



wxMaxima screen

per phase

>>> Normal termination Maxima 5.55 s / wall 5.65 s [94.83,98.39]%

***** $XFER[a] = \frac{A\,Offs}{A+1}$

$$XFER[b] = \frac{((A^2+A)\,Ca+(A^2+A)\,C2+(A^2+A)\,C1)\,Offs\,Z+(-A^2\,Ca-A^2\,C2-A^2\,C1)\,Offs+((A^2+A)\,C2+(A^2+A)\,C1)\,Vin}{((A+1)\,Ca+(A+1)\,C2+(A^2+2A+1)\,C1)\,Z}$$

***** [No offset] $XFER[a] = 0$

$$XFER[b] = \frac{(A\,C2+A\,C1)\,Vin}{(Ca+C2+(A+1)\,C1)\,Z}$$

***** [A infinite] $XFER[a] = Offs$

$$XFER[b] = \frac{(Ca+C2+C1)\,Offs\,Z+(C2+C1)\,Vin+(-Ca-C2-C1)\,Offs}{C1\,Z}$$

per cycle

>>> Normal termination Maxima 5.19 s / wall 5.2 s [95.42,98.39]%

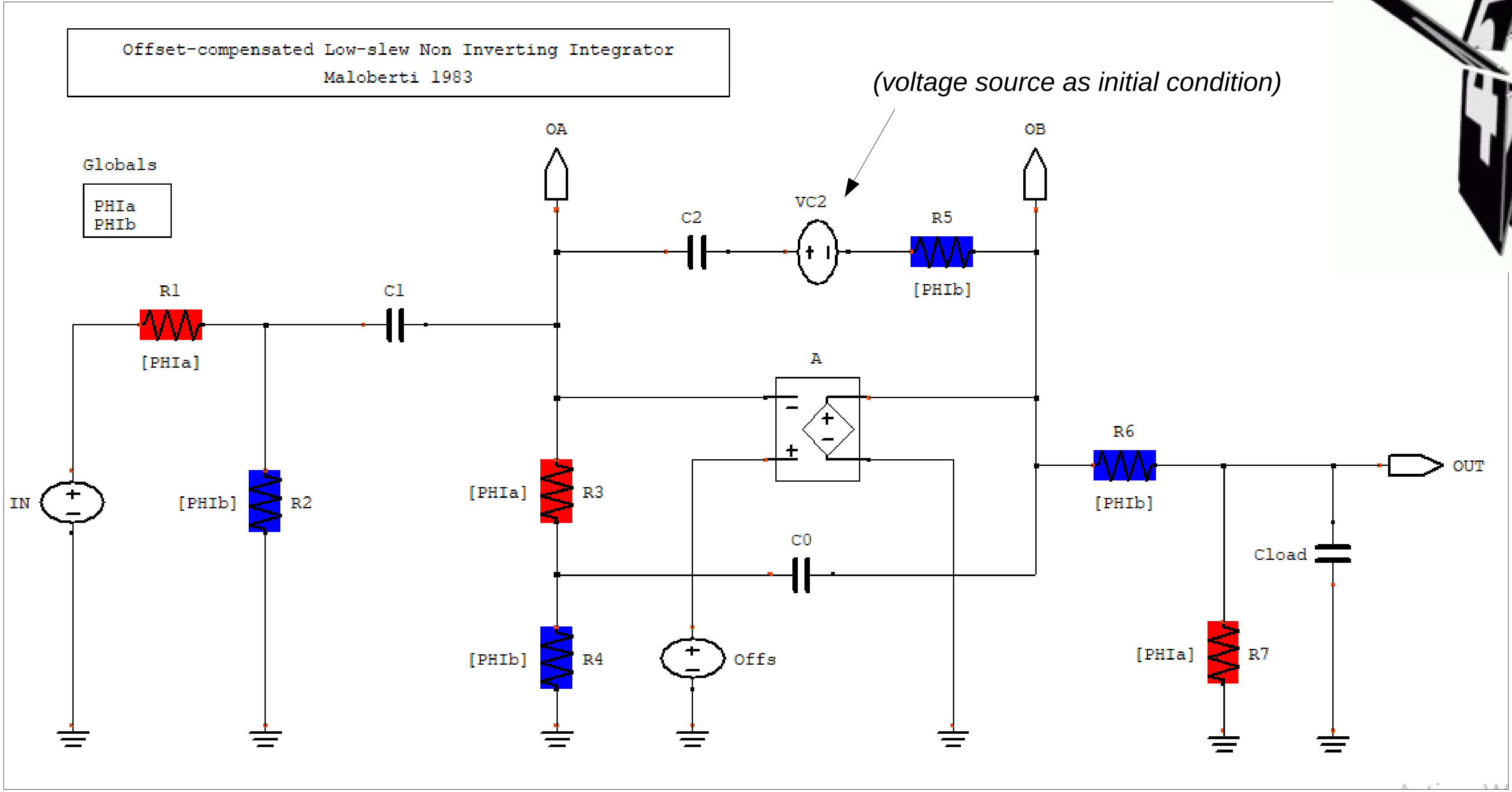
***** $$XFER = \frac{(A\,Ca+A\,C2+A\,C1)\,Offs+((A^2+A)\,C2+(A^2+A)\,C1)\,Vin}{(A+1)\,Ca+(A+1)\,C2+(A^2+2A+1)\,C1}$$

***** [No offset] $$XFER = \frac{(A\,C2+A\,C1)\,Vin}{Ca+C2+(A+1)\,C1}$$

***** [A infinite] $$XFER = \frac{(C2+C1)\,Vin}{C1}$$

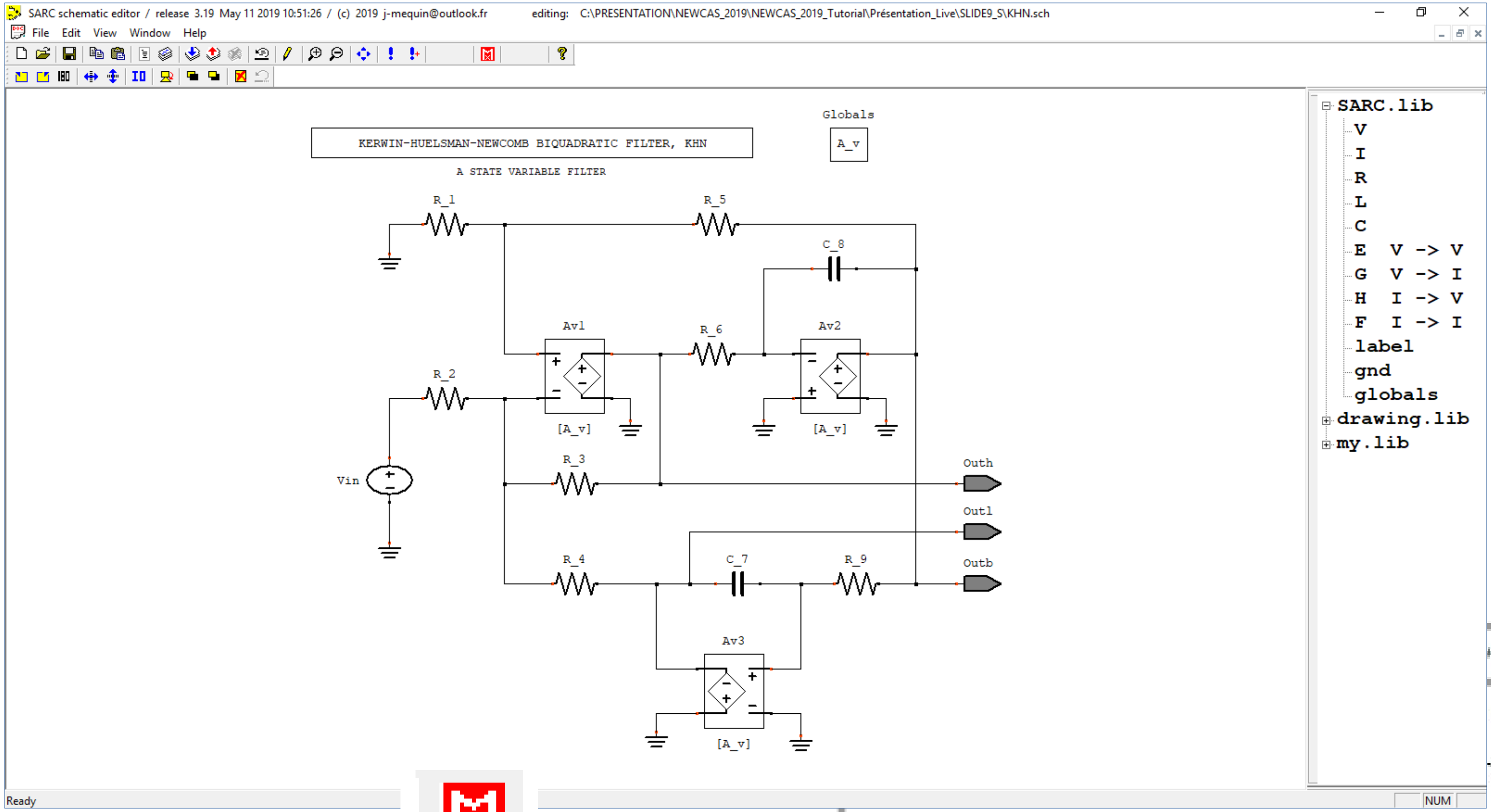
An Example, Modeling and Simulating a SWC Circuit in ‘**s**’

library file ‘/Simulate/NapaDos/Hdr/Max/SWC_Integrator/Maloberti_Integrator1_NI.sch’

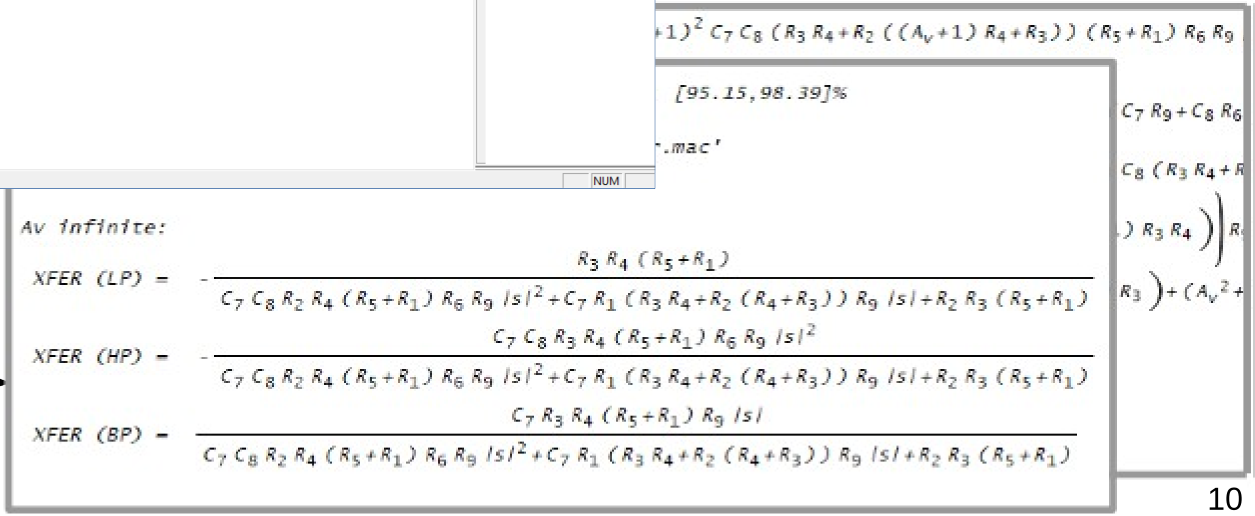


Modeling in ‘**s**’ of a Continuous-Time Circuit

Schematic Editor



wxMaxima screens



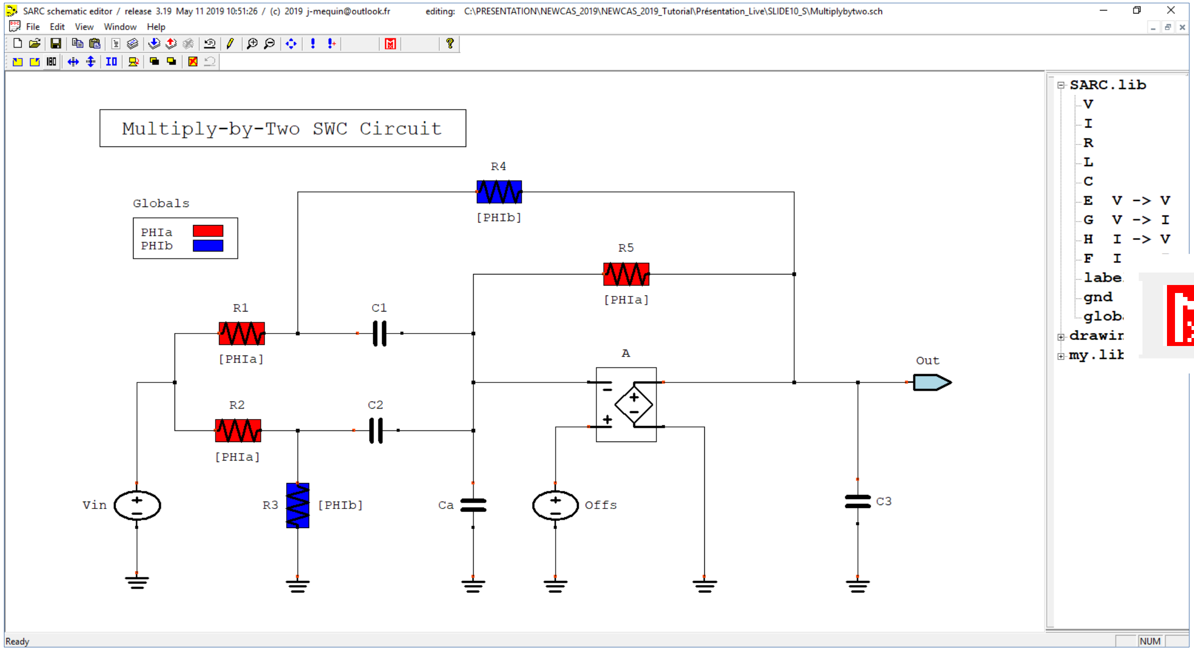
Transfer Functions

10

(modeling)

SWC Circuit Modeled in 's' to Introduce Bandwidth

Schematic Editor



This model needs a more sophisticated simulator engine

wxMaxima screen

>>> Normal termination Maxima 4.62 s / wall 4.67 s [95.61,98.39]%

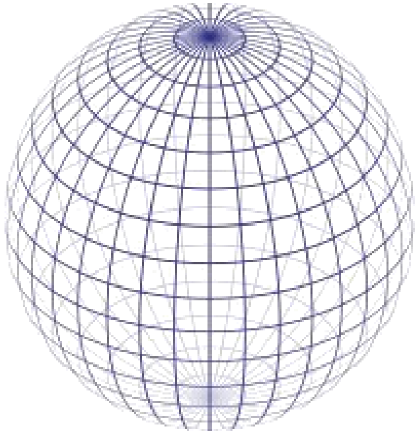
XFER =

$$(C2\ PHIa\ PHIb\ (2\ C1\ PHIb\ RDS + PHIa\ (2\ C1\ (RDS - A\ PHIb) + Ca\ RDS))\ |s|^2 +$$
$$((C2+C1)\ PHIb^2\ RDS + PHIa\ (PHIb\ (3\ C2\ RDS - A\ ((C2+C1)\ PHIb + (C2+C1)\ PHIa)) + ((Ca+2\ C1)\ PHIb + (Ca+C2+C1)\ PHIa)\ RDS))\ |s| + (PHIb+PHIa)\ RDS)\ /\ (C1\ C2\ Ca\ Cload\ PHIa^3\ PHIb^2\ RDS\ |s|^4 + PHIa^2$$
$$PHIb\ ((C1\ (C2\ (3\ Cload + Ca) + Ca\ Cload) + C2\ Ca\ Cload)\ PHIb\ RDS + PHIa\ (C1\ (C2\ (Ca\ (RDS + PHIb) + 2\ Cload\ RDS) + Ca\ Cload\ RDS) + C2\ Ca\ Cload\ RDS))\ |s|^3 + PHIa\ ($$
$$(C1\ (2\ Cload + Ca + 2\ C2) + C2\ (2\ Cload + Ca) + Ca\ Cload)\ PHIb^2\ RDS + PHIa$$
$$(PHIb\ (2\ C1\ Ca\ RDS + (C1\ (Ca + (A+3)\ C2) + C2\ Ca)\ PHIb + (C1\ (Ca + (A+2)\ C2) + C2\ Ca)\ PHIa) + (((2\ Ca + 3\ (C2+C1))\ Cload + C2\ (2\ Ca + 3\ C1))\ PHIb + (C1\ (Cload + Ca + C2) + (Ca+C2)\ Cload)\ PHIa)\ RDS))$$
$$|s|^2 + ((Cload + Ca + C2 + C1)\ PHIb^2\ RDS + PHIa$$
$$(PHIb\ (4\ C2\ RDS + (Ca + (A+2)\ (C2+C1))\ PHIb) + ((2\ Cload + 3\ Ca + 2\ C1)\ PHIb + (Cload + 2\ (Ca+C2) + C1)\ PHIa)\ RDS + PHIa\ ((2\ Ca + (A+3)\ C2 + (2\ A+3)\ C1)\ PHIb + (Ca + C2 + (A+1)\ C1)\ PHIa)))\ |s| +$$
$$(PHIb + PHIa)\ (RDS + (A+1)\ PHIb + (A+1)\ PHIa))$$

Environment, Constraints, Motivation... and Tactics !

What do we need? How to organize a simulation? How to enforce reuse? …

We need to keep a TIGHT CONTROL on the modeling and on the simulations !
We will ENFORCE THE COHERENCE of the netlists.



We need ultimate SPEED.
We have chosen ANSI-**C** as it is fast and is a de-facto standard.

The netlist must be as CONCISE and crystal clear as possible.

We have an imperious constraint: a short term and steady ROI.
We have chosen an INCREMENTAL DEVELOPMENT of the project.

And NO HASSLE. FREEWARE only.

For Portability, ANSI-C ! But ...

C is **verbose**.
As a return of experience, **C** is not suited to describe one-time programs as the coding and debugging are time-consuming. -This is not our job- .

C is **lax**.
We need a strict control of the code as we cannot mix freely analog and digital signals.

C is **too generic**. We need a more specialized language.
Our target is signal processing.

C compiler produces **cryptic error messages**. Debugging is overwhelming.

C is loose and presents surprising **hazards, limitations** ...

2 / 3 = 0
(double) (2 / 3) = 0.0
2.0 / 3 = 0.666..

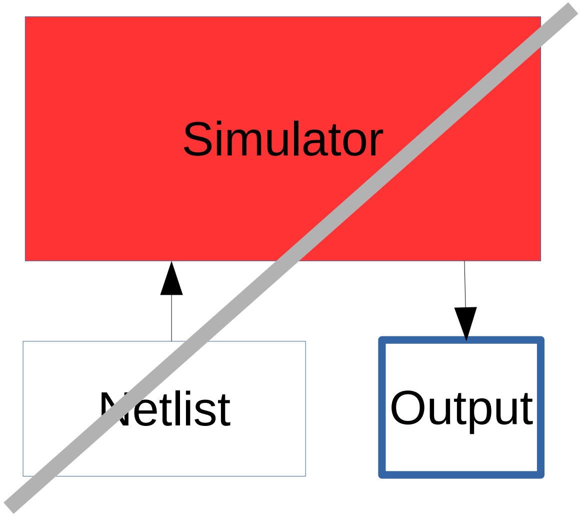
int n = 1;
&n ? no problem
&1 ? ERROR

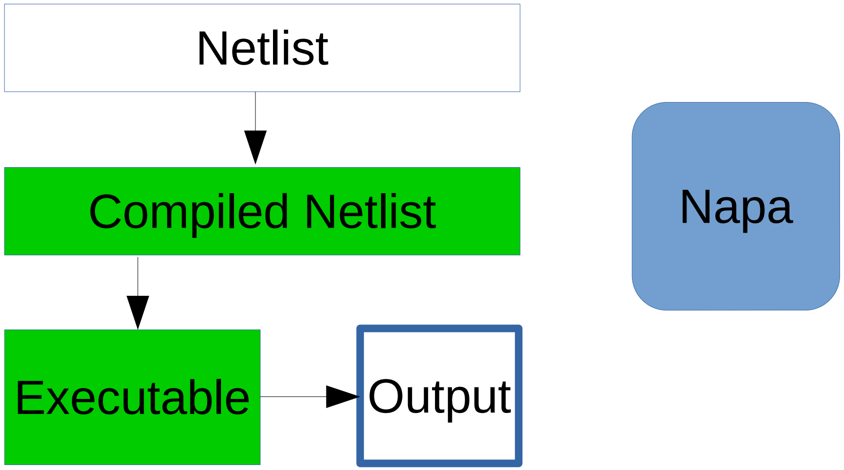
Our Choice, an Ad-Hoc Cycle Driven Simulator

We need to find a golden solution, so that we can have our cake and eat it.

We will not use a generic simulator running netlists but we will build an ad-hoc executable which **IS** the simulator of our netlist.

We will use a language of netlist which will have a direct translation in a streamlined dedicated ‘**C**’ code which will be compiled and executed on the fly.





The Simulations Are Therefore Just **Fast**

Example: **81 TSNR** for a **3rd Order ΣΔ SWC Modulator.**
These TSNR need a total of **81 FFTs of 2^{16} points**.
A simulation of **5.3 millions cycles**.

file ‘tsnr.c’

file ‘tsnr.nap’

```
header  <napatool.hdr>

title   "TSNR Analysis of a MASH 2+1 SWC Sigma-Delta Modulator"

fs      2.0e6

dvar  ampldb LINSWEEP(TOOL_INDEX, -80.0,0.0, 81)  &update &export
dvar    ampl DB2LIN(ampldb, 1.0)                  &update
dvar    freq  1234.56789
dvar      ph  rand_uniform(0.0, _2pi_)

node  vrefh  dc    1.0
node  vrefl  dc   -1.0
node     in  osc  0.0 ampl freq ph                // fast sinewave

node    out  cell sd  "./sd21.net" "./_sd21.dat" in vrefh vrefl

ivar   npts  POWEROF2(16)
tool   tsnr  stdout  out 1 10.0 16.0e3 npts

terminate    0.0 < ampldb

directive  WINDOW   BLACKMAN_HARRIS_7
directive  QUIET
```

compile netlist

15

*compile **C** code & execute*

file ‘./sd21.net’

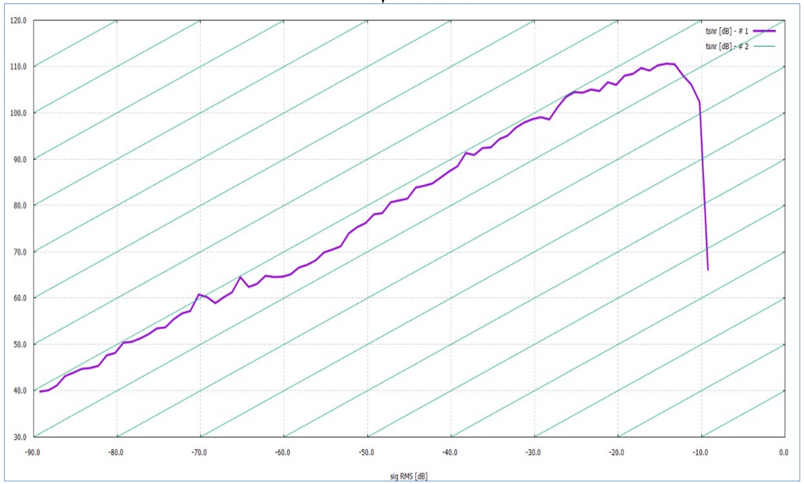
```
cell_interface $out $sdfile $in $vh $vl

data $sdfile   $A $B $C $D $E $F  $op1 $op2 $op3 $cmp4 $cmp5

dvar $i1  $A
dvar $i2  $A/$C
dvar $j1  $B/$A
dvar $j2  2.0*($B/$C)
dvar $k1  $D/($C*I2D($F))
dvar $k2  $D/($B*I2D($F))
dvar $k3  $D/$E
ivar $ia  $F
node $vp  dalgebra  $vh
node $vn  dalgebra  $vl
node $th  average   $vn   $vp
node $s01 cell i1  <Integrator1/d2_a.net>  $op1 $i1 $in  $i2 $s03 0.0
node $s02 cell i2  <Integrator1/d2_a.net>  $op2 $j1 $s01 $j2 $s03 0.0
node $y1  cell c4  <Comparator/2_a.net>   $cmp4 $s02 $th
node $s03 mux       $y1  -$vn -$vp
node $s07 cell i3  <Integrator1/d3_a.net>  $op3 $k1 $s03 $k2 $s02 $k3 $s08
0.0
node $y2  cell c5  <Comparator/2_a.net>   $cmp5 $s07 $th
node $s08 mux       $y2  -$vn -$vp
node $s09 delay     $y1
node $s10 delay     $s09
node $s11 differentiator $y2
node $s12 differentiator $s11
node $s13 gain      $ia  $s12
node $out sum       $s10 $s13
```

screen

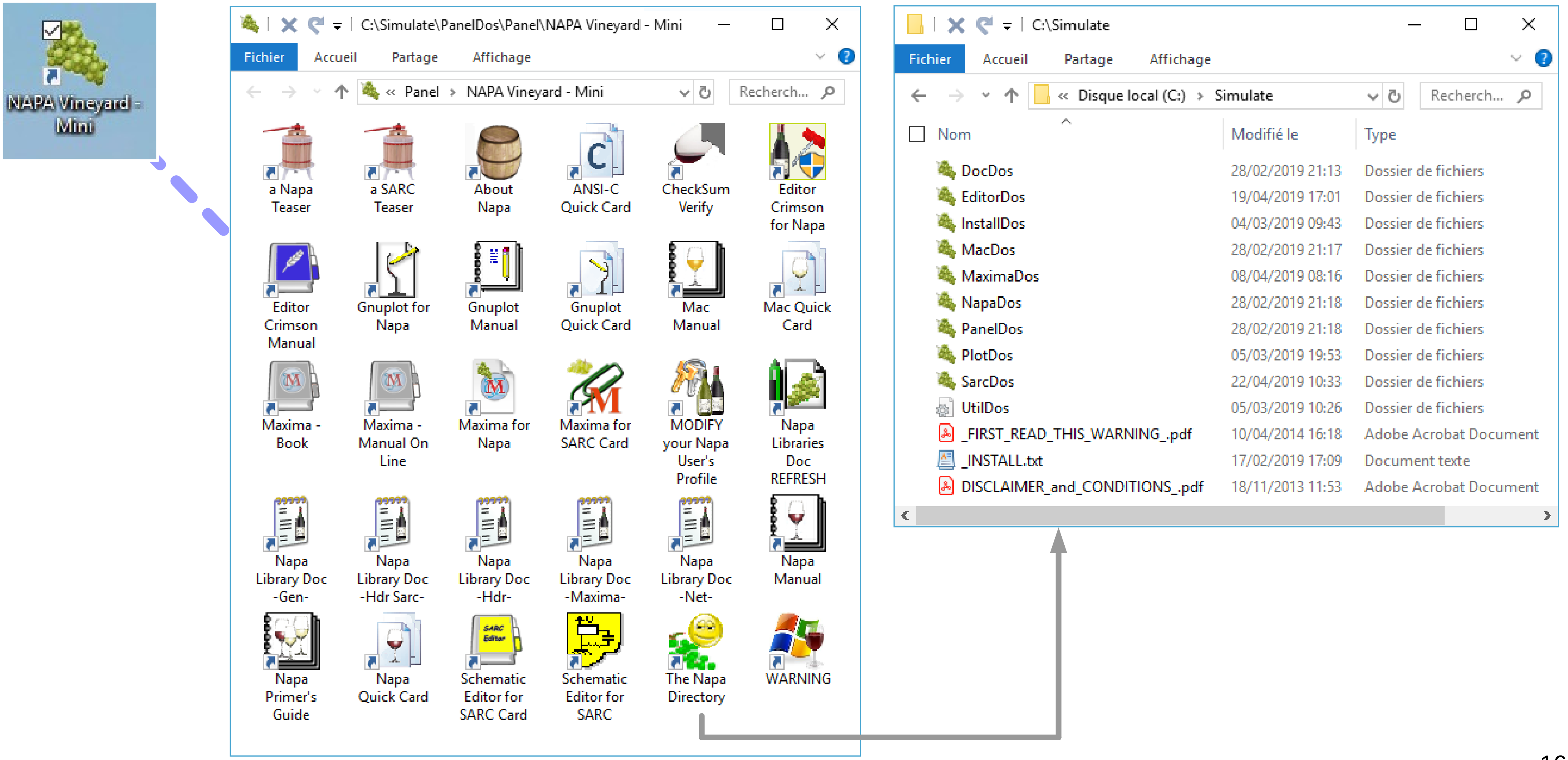
```
****
****  TSNR Analysis of a MASH 2+1 SWC Sigma-Delta Modulator
****

NAPA Tools Information:     (          tsnr[0]) Appending ad hoc template to tsnr data

****  Random Seed [I]   :                778788825 ****
****  Output Tag  [O]   :                254179993 ****

****  NAPA Compiler     :           V4.00 for Win64 ****

****  Main Netlist      :                 TSNR.tmp ****

****  Simulator Time    :              2.65421  s ****
****  Simulator Index   :                5 308 417 ****
****  Tool Index        :                       81 ****

****  Run Time I/O      :                          ****

      ->  stdout                              [  0] ****

****  Stopwatch         :          H00:M00:S00.877 ****

****  Normal Termination                           ****

[TSNR]
```

< 0.9 second

gnuplot screen

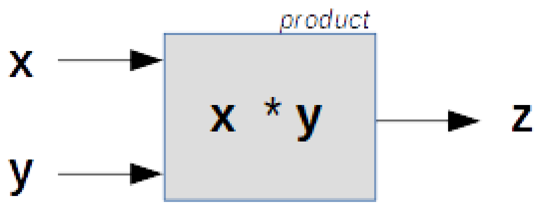


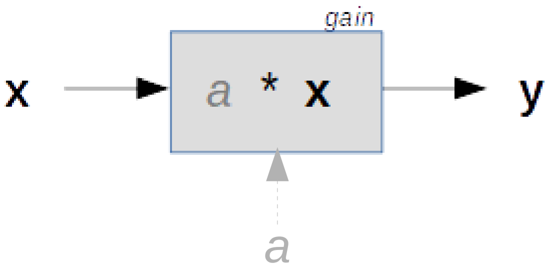
Napa, a Compiler and a Simulator



Mixed Signal: Analog and Digital

https://www.gnu.org/graphics/meditate.fr.html





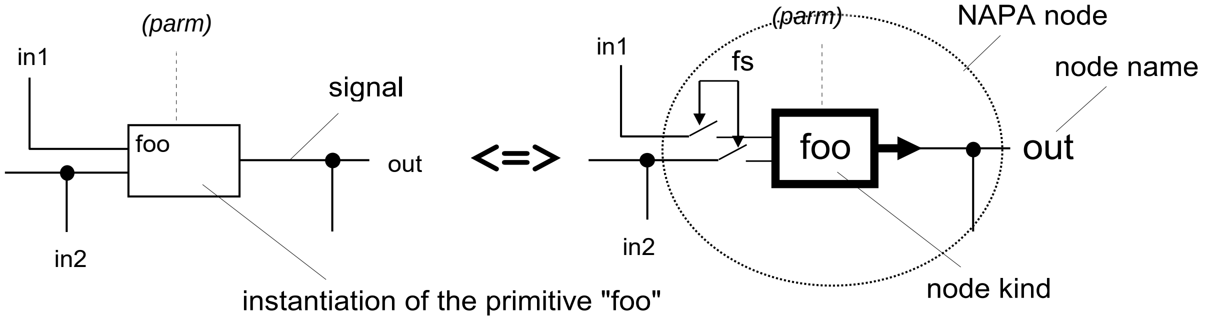
SIGNAL — *Digital or analog signal represented by ‘node ...’*

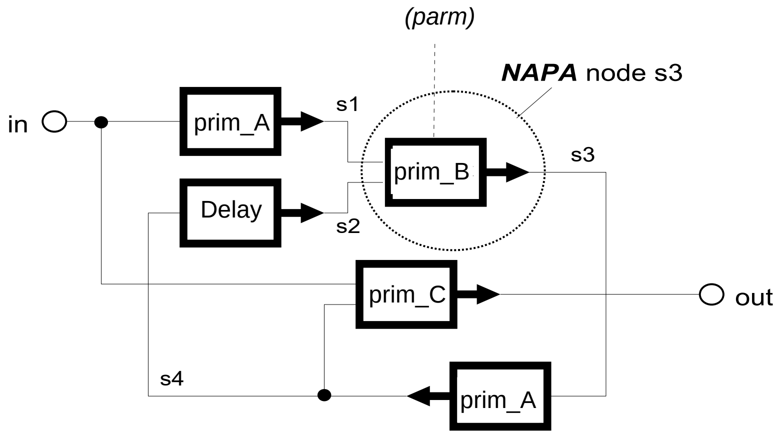
Parameter — *Analog parameter represented by ‘dvar ...’*
Digital parameter represented by ‘ivar ...’

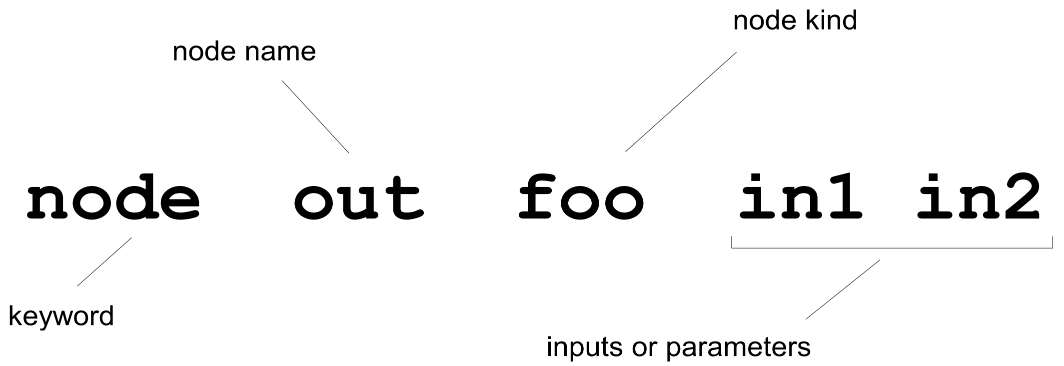
The simulations will rely on the ANSI-C compiler **gcc.**

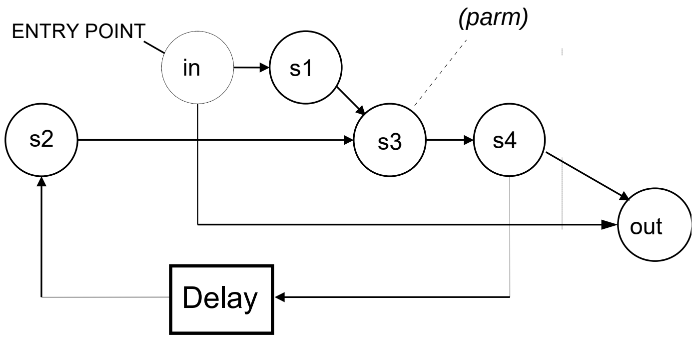
Our choice:			
	analog	double float	for its resolution of **15** digits
	digital	long long int	for its range of **]-2^{63}**, **2^{63}**[

The Nodes

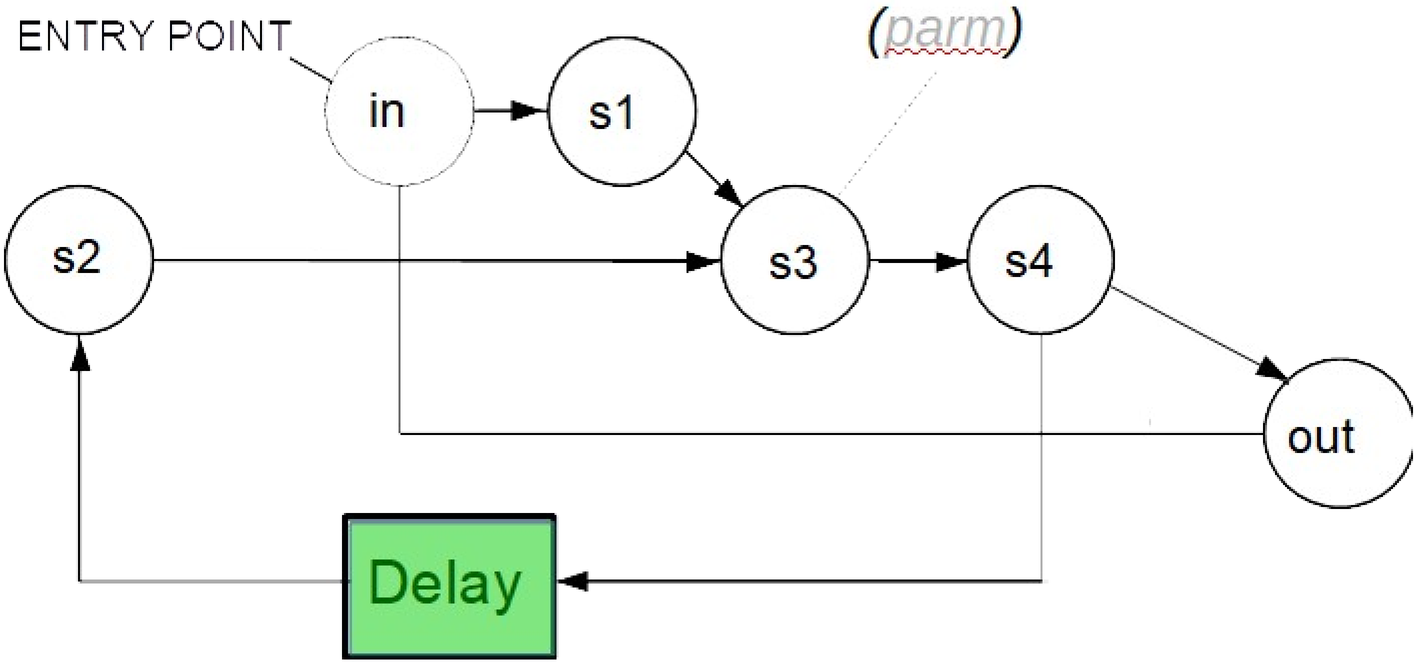

(parm)
in1
signal
foo
out
in2
instantiation of the primitive "foo"
<=>
(parm)
NAPA node
in1
fs
node name
foo
out
in2
node kind


(parm)
NAPA node s3
in
prim_A
s1
prim_B
s3
Delay
s2
prim_C
out
s4
prim_A


node kind
node name
node out foo in1 in2
keyword
inputs or parameters


ENTRY POINT
(parm)
in
s1
s2
s3
s4
out
Delay

Sorting the Nodes for a Linear Computing Flow



loop

TIME …

(update parm)

in (*TIME, ...*)

s1 (in)

s3 (***old***_s2, s1, *parm*)

s4 (s3)

out (in, s4)

new_**s2** (s4)

A Minimalist List of Basic Elements

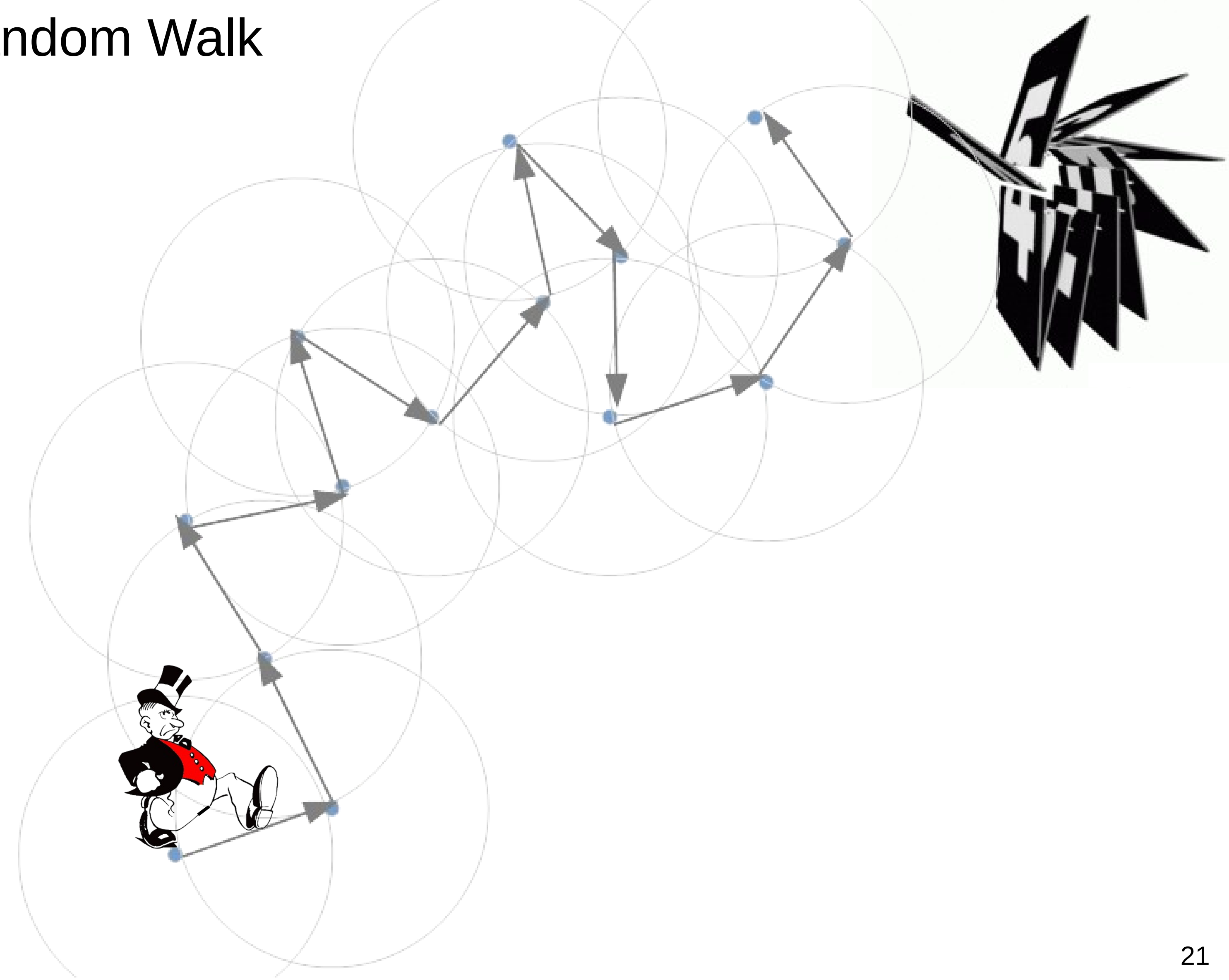
Analog type
Digital type

James Petts from London, England
[CC BY-SA 2.0 (https://creativecommons.org/licenses/by-sa/2.0)]

Z domain models may be built with these elements.

Output	*output* ...
Control	*terminate* ...
Resource	*header* ...
Sampling Frequency	*fs* ...
	ivar ...
	dvar ...
One-lined **C** expression	*node .. ialgebra* ...
	node .. dalgebra ...
	node .. dc ...
+	*node .. sum* ...
	node .. offset ...
x	*node .. prod* ...
	node .. gain ...
Delay	*node .. delay* ...
Initialization	*init* ...

An Example: the Random Walk



Modeling a 2D Random Walk

ANSI-C, C11 code, file ‘random_walk.c’

From the Crimson Editor :

NAPA main netlist, file ‘random_walk.nap’

```
header     <napa.hdr>
header     <Function/random.hdr>

fs         1.0

dvar       stp    0.3

node       phi      dalgebra     rand_uniform(0.0,_2pi_)
node       xstep    dalgebra     stp * sin(phi)
node       ystep    dalgebra     stp * cos(phi)

node       xd       delay   x
node       yd       delay   y

node       x        sum     xd   xstep
node       y        sum     yd   ystep

output     stdout   x   y   phi

terminate 1000LL <= LOOP_INDEX
```

Napa netlist compiler

```
...
#include  "/Simulate/NapaDos/Hdr/napa.hdr"
#include  "/Simulate/NapaDos/Hdr/Function/random.hdr"

double      FSL = 1.0;
double      napa_abs_time = 0.0;
long long   LOOP_INDEX = 0LL;

double      d_var_stp = 0.3;
double      d_node_xd = 0.0;
double      d_node_yd = 0.0;
double      d_node_phi = 0.0;
double      d_node_ystep = 0.0;
double      d_node_y = 0.0;
double      d_node_xstep = 0.0;
double      d_node_x = 0.0;

do {
   napa_abs_time  = ((double) LOOP_INDEX) / FSL;

   d_node_xd = d_node_x;
   d_node_yd = d_node_y;
   d_node_phi = rand_uniform(0.0,_2pi_);
   d_node_ystep = d_var_stp * cos(d_node_phi);
   d_node_y = d_node_yd + d_node_ystep;
   d_node_xstep = d_var_stp * sin(d_node_phi);
   d_node_x = d_node_xd + d_node_xstep;
   fprintf(stdout, "%.15e", napa_abs_time);
   fprintf(stdout, " % .12e % .12e % .12e\n", d_node_x, d_node_y, d_node_phi);
   LOOP_INDEX++;

} while (!(1000LL <= LOOP_INDEX));
...
```

The Control

```
header      <napa.hdr>
header      <Function/random.hdr>

fs          1.0

dvar        stp    0.3

node        phi     dalgebra    rand_uniform(0.0, _2pi_)
node        xstep   dalgebra    stp * sin(phi)
node        ystep   dalgebra    stp * cos(phi)

node        xd      delay  x
node        yd      delay  y

node        x       sum    xd  xstep
node        y       sum    yd  ystep

output      stdout  x  y  phi

terminate  1000LL <= LOOP_INDEX
```

```
...

#include  "/Simulate/NapaDos/Hdr/napa.hdr"
#include  "/Simulate/NapaDos/Hdr/Function/random.hdr"

double      FSL = 1.0;
double      napa_abs_time = 0.0;
long long  LOOP_INDEX = 0LL;

double      d_var_stp = 0.3;

double      d_node_xd = 0.0;
double      d_node_yd = 0.0;
double      d_node_phi = 0.0;
double      d_node_ystep = 0.0;
double      d_node_y = 0.0;
double      d_node_xstep = 0.0;
double      d_node_x = 0.0;

do {
  napa_abs_time  = ((double) LOOP_INDEX) / FSL;

  d_node_xd = d_node_x;
  d_node_yd = d_node_y;
  d_node_phi = rand_uniform(0.0,_2pi_);
  d_node_ystep = d_var_stp * cos(d_node_phi);
  d_node_y = d_node_yd + d_node_ystep;
  d_node_xstep = d_var_stp * sin(d_node_phi);
  d_node_x = d_node_xd + d_node_xstep;
  fprintf(stdout, "%.15e", napa_abs_time);
  fprintf(stdout, " % .12e % .12e % .12e\n", d_node_x, d_node_y, d_node_phi);
  LOOP_INDEX++;

 } while (!(1000LL <= LOOP_INDEX));

...
```

The Initialization and the Loop

```
header     <napa.hdr>
header     <Function/random.hdr>

fs         1.0

dvar       stp    0.3

node       phi     dalgebra   rand_uniform(0.0,_2pi_)
node       xstep   dalgebra   stp * sin(phi)
node       ystep   dalgebra   stp * cos(phi)

node       xd      delay  x
node       yd      delay  y

node       x       sum    xd  xstep
node       y       sum    yd  ystep

output     stdout  x  y  phi

terminate 1000LL <= LOOP_INDEX
```

```
...

#include  "/Simulate/NapaDos/Hdr/napa.hdr"
#include  "/Simulate/NapaDos/Hdr/Function/random.hdr"

double     FSL = 1.0;
double     napa_abs_time = 0.0;
long long  LOOP_INDEX = 0LL;

double     d_var_stp = 0.3;

double     d_node_xd = 0.0;
double     d_node_yd = 0.0;
double     d_node_phi = 0.0;
double     d_node_ystep = 0.0;
double     d_node_y = 0.0;
double     d_node_xstep = 0.0;
double     d_node_x = 0.0;

do {
  napa_abs_time  = ((double) LOOP_INDEX) / FSL;

  d_node_xd = d_node_x;
  d_node_yd = d_node_y;
  d_node_phi = rand_uniform(0.0,_2pi_);
  d_node_ystep = d_var_stp * cos(d_node_phi);
  d_node_y = d_node_yd + d_node_ystep;
  d_node_xstep = d_var_stp * sin(d_node_phi);
  d_node_x = d_node_xd + d_node_xstep;
  fprintf(stdout, "%.15e", napa_abs_time);
  fprintf(stdout, " % .12e % .12e % .12e\n", d_node_x, d_node_y, d_node_phi);
  LOOP_INDEX++;

 } while (!(1000LL <= LOOP_INDEX));

...
```

Initialization

A single loop

Type Determination and Declaration

```
header      <napa.hdr>
header      <Function/random.hdr>

fs          1.0

dvar ──► double     0.3

node   double ◄── dalgebra    rand_uniform(0.0,_2pi_)
node        xstep  dalgebra    stp * sin(phi)
node        ystep  dalgebra    stp * cos(phi)

node   double ◄── delay  double
node        yd     delay  y

node   double ◄── sum  double  double
node        y      sum     yd  ystep

output      stdout  x  y  phi

terminate 1000LL <= LOOP_INDEX
```

3 categories of Nodes: digital, analog or ‘chameleonic’

```
…

#include  "/Simulate/NapaDos/Hdr/napa.hdr"
#include  "/Simulate/NapaDos/Hdr/Function/random.hdr"

double      FSL = 1.0;
double      napa_abs_time = 0.0;
long long  LOOP_INDEX = 0LL;

double      d_var_stp = 0.3;

double      d_node_xd = 0.0;
double      d_node_yd = 0.0;
double      d_node_phi = 0.0;
double      d_node_ystep = 0.0;
double      d_node_y = 0.0;
double      d_node_xstep = 0.0;
double      d_node_x = 0.0;

do {
   napa_abs_time  = ((double) LOOP_INDEX) / FSL;

    d_node_xd = d_node_x;
   d_node_yd = d_node_y;
   d_node_phi = rand_uniform(0.0,_2pi_);
   d_node_ystep = d_var_stp * cos(d_node_phi);
   d_node_y = d_node_yd + d_node_ystep;
   d_node_xstep = d_var_stp * sin(d_node_phi);
   d_node_x = d_node_xd + d_node_xstep;
   fprintf(stdout, "%.15e", napa_abs_time);
   fprintf(stdout, " % .12e % .12e % .12e\n", d_node_x, d_node_y, d_node_phi);
   LOOP_INDEX++;

 } while (!(1000LL <= LOOP_INDEX));

…
```

Sorting and Compilation of the Nodes

```
header     <napa.hdr>
header     <Function/random.hdr>

fs         1.0

dvar       stp     0.3

node       phi     dalgebra    rand_uniform(0.0, _2pi_)
node       xstep   dalgebra    stp * sin(phi)
node       ystep   dalgebra    stp * cos(phi)

node       xd      delay  x
node       yd      delay  y

node       x       sum    xd  xstep
node       y       sum    yd  ystep

output     stdout  x  y  phi

terminate 1000LL <= LOOP_INDEX
```

```
…

#include  "/Simulate/NapaDos/Hdr/napa.hdr"
#include  "/Simulate/NapaDos/Hdr/Function/random.hdr"

double     FSL = 1.0;
double     napa_abs_time = 0.0;
long long  LOOP_INDEX = 0LL;

double     d_var_stp = 0.3;

double     d_node_xd = 0.0;
double     d_node_yd = 0.0;
double     d_node_phi = 0.0;
double     d_node_ystep = 0.0;
double     d_node_y = 0.0;
double     d_node_xstep = 0.0;
double     d_node_x = 0.0;

do {
  napa_abs_time  = ((double) LOOP_INDEX) / FSL;

   d_node_xd = d_node_x;
  d_node_yd = d_node_y;
  d_node_phi = rand_uniform(0.0,_2pi_);
  d_node_ystep = d_var_stp * cos(d_node_phi);
  d_node_y = d_node_yd + d_node_ystep;
  d_node_xstep = d_var_stp * sin(d_node_phi);
  d_node_x = d_node_xd + d_node_xstep;
  fprintf(stdout, "%.15e", napa_abs_time);
  fprintf(stdout, " % .12e % .12e % .12e\n", d_node_x,d_ node_y,d_ node_phi);
  LOOP_INDEX++;

 } while (!(1000LL <= LOOP_INDEX));
...
```

old value of x

new value of x

Introduction of a ‘**C**’ Formula

```
header     <napa.hdr>
header     <Function/random.hdr>

fs         1.0

dvar       stp     0.3

node       phi     dalgebra    rand_uniform(0.0,_2pi_)
node       xstep   dalgebra    stp * sin(phi)
node       ystep   dalgebra    stp * cos(phi)

node       xd      delay  x
node       yd      delay  y

node       x       sum    xd   xstep
node       y       sum    yd   ystep

output     stdout  x  y  phi

terminate 1000LL <= LOOP_INDEX
```

copy,
process
and paste

```
...

#include  "/Simulate/NapaDos/Hdr/napa.hdr"
#include  "/Simulate/NapaDos/Hdr/Function/random.hdr"

double     FSL = 1.0;
double     napa_abs_time = 0.0;
long long  LOOP_INDEX = 0LL;

double     d_var_stp = 0.3;

double     d_node_xd = 0.0;
double     d_node_yd = 0.0;
double     d_node_phi = 0.0;
double     d_node_ystep = 0.0;
double     d_node_y = 0.0;
double     d_node_xstep = 0.0;
double     d_node_x = 0.0;

do {
  napa_abs_time  = ((double) LOOP_INDEX) / FSL;

  d_node_xd = d_node_x;
  d_node_yd = d_node_y;
  d_node_phi = rand_uniform(0.0,_2pi_);
  d_node_ystep = d_var_stp * cos(d_node_phi);
  d_node_y = d_node_yd + d_node_ystep;
  d_node_xstep = d_var_stp * sin(d_node_phi);
  d_node_x = d_node_xd + d_node_xstep;
  fprintf(stdout, "%.15e", napa_abs_time);
  fprintf(stdout, " % .12e % .12e % .12e\n", d_node_x, d_node_y, d_node_phi);
  LOOP_INDEX++;

} while (!(1000LL <= LOOP_INDEX));

...
```

C formula may be directly introduced

in nodes *dalgebra, ialgebra, dc ...*
in instructions *dvar, ivar, init, terminate ...*

Inclusion of a ‘**C**’ Resource

```
header     <napa.hdr>
header     <Function/random.hdr>

fs         1.0

dvar       stp    0.3

node       phi    dalgebra    rand_uniform(0.0, _2pi_)
node       xstep  dalgebra    stp * sin(phi)
node       ystep  dalgebra    stp * cos(phi)

node       xd     delay  x
node       yd     delay  y

node       x      sum    xd  xstep
node       y      sum    yd  ystep

output     stdout  x  y  phi

terminate 1000LL <= LOOP_INDEX
```

```
...

#include  "/Simulate/NapaDos/Hdr/napa.hdr"
#include  "/Simulate/NapaDos/Hdr/Function/random.hdr"

double     FSL = 1.0;
double     napa_abs_time = 0.0;
long long  LOOP_INDEX = 0LL;

double     d_var_stp = 0.3;

double     d_node_xd = 0.0;
double     d_node_yd = 0.0;
double     d_node_phi = 0.0;
double     d_node_ystep = 0.0;
double     d_node_y = 0.0;
double     d_node_xstep = 0.0;
double     d_node_x = 0.0;

do {
  napa_abs_time  = ((double) LOOP_INDEX) / FSL;

  d_node_xd = d_node_x;
  d_node_yd = d_node_y;
  d_node_phi = rand_uniform(0.0,_2pi_);
  d_node_ystep = d_var_stp * cos(d_node_phi);
  d_node_y = d_node_yd + d_node_ystep;
  d_node_xstep = d_var_stp * sin(d_node_phi);
  d_node_x = d_node_xd + d_node_xstep;
  fprintf(stdout, "%.15e", napa_abs_time);
  fprintf(stdout, " % .12e % .12e % .12e\n", d_node_x, d_node_y, d_node_phi);
  LOOP_INDEX++;

} while (!(1000LL <= LOOP_INDEX));

...
```

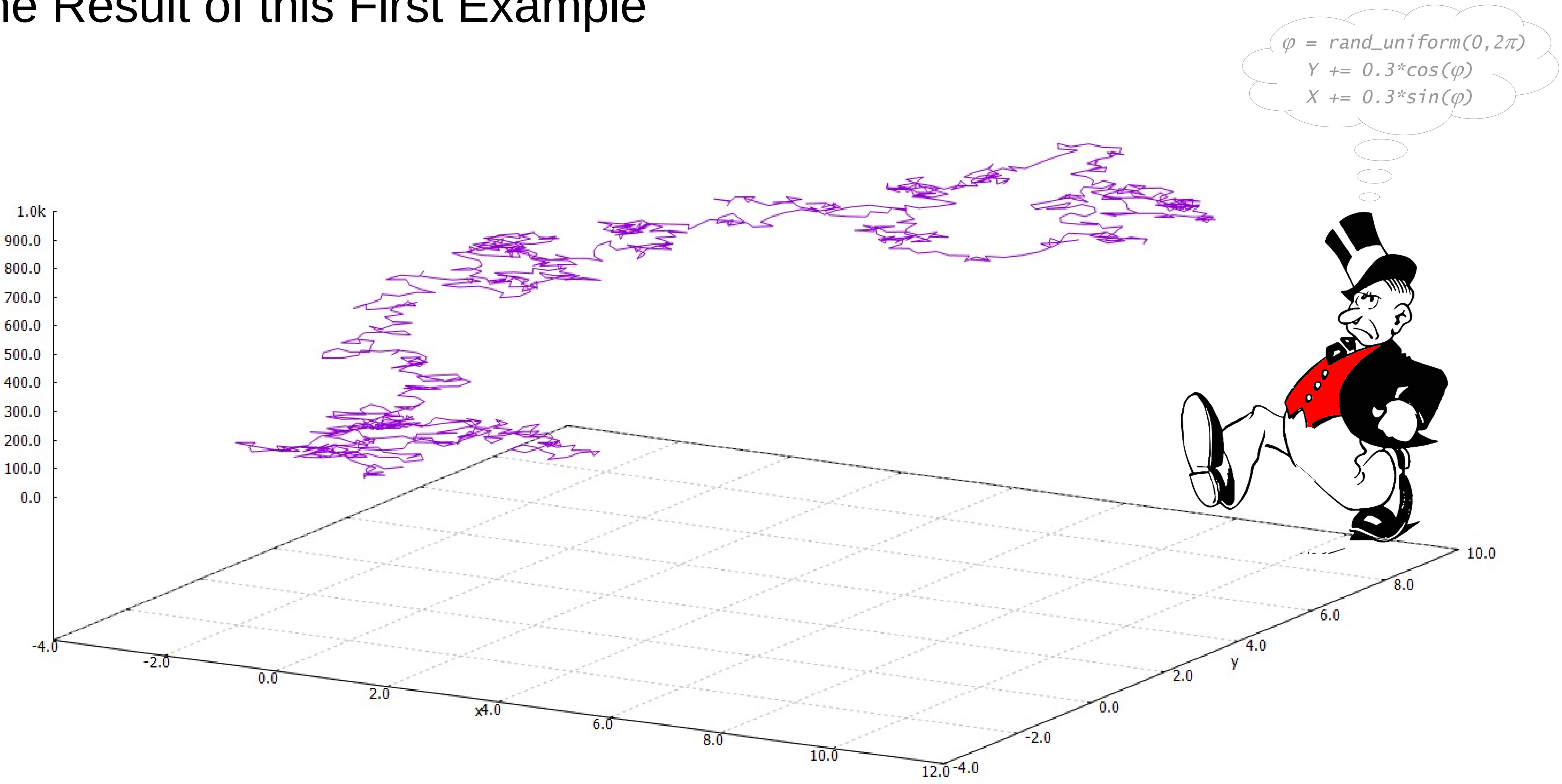
If a function is not a **C** native function, an external resource must be included to the produced **C** code, and therefore will be compiled **with** this code.

Time-Domain Output

```
header      <napa.hdr>
header      <Function/random.hdr>

fs          1.0

dvar        stp     0.3

node        phi     dalgebra    rand_uniform(0.0, _2pi_)
node        xstep   dalgebra    stp * sin(phi)
node        ystep   dalgebra    stp * cos(phi)

node        xd      delay  x
node        yd      delay  y

node        x       sum    xd  xstep
node        y       sum    yd  ystep

output      stdout  x  y  phi

terminate 1000LL <= LOOP_INDEX
```

```
...

#include  "/Simulate/NapaDos/Hdr/napa.hdr"
#include  "/Simulate/NapaDos/Hdr/Function/random.hdr"

double     FSL = 1.0;
double     napa_abs_time = 0.0;
long long  LOOP_INDEX = 0LL;

double     d_var_stp = 0.3;

double     d_node_xd = 0.0;
double     d_node_yd = 0.0;
double     d_node_phi = 0.0;
double     d_node_ystep = 0.0;
double     d_node_y = 0.0;
double     d_node_xstep = 0.0;
double     d_node_x = 0.0;

do {
  napa_abs_time  = ((double) LOOP_INDEX / FSL);

  d_node_xd = d_node_x;
  d_node_yd = d_node_y;
  d_node_phi = rand_uniform(0.0,_2pi_);
  d_node_ystep = d_var_stp * cos(d_node_phi);
  d_node_y = d_node_yd + d_node_ystep;
  d_node_xstep = d_var_stp * sin(d_node_phi);
  d_node_x = d_node_xd + d_node_xstep;
  fprintf(stdout, "%.15e", napa_abs_time);
  fprintf(stdout, " % .12e % .12e % .12e\n",d_node_x, d_node_y, d_node_phi);
  LOOP_INDEX++;

 } while (!(1000LL <= LOOP_INDEX));

...
```

The Result of this First Example



gnuplot screen

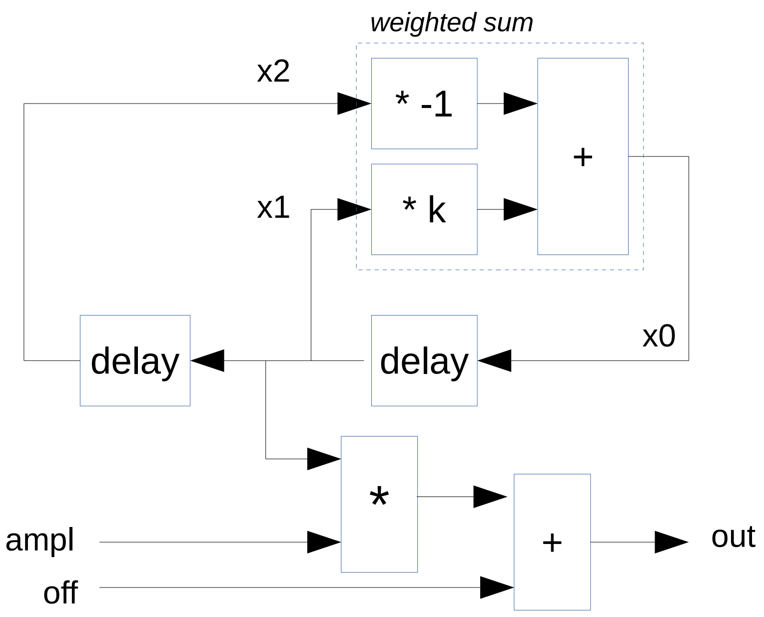
A Few Other Elements

Weighted Sum	*node .. wsum ...*
DC	*node .. dc ...* *node .. const ...*
Sinewave	*node .. sine ...*
Declaration	*declare ...*

Our first challenge: the **sinewave** is using the ANSI-C function ‘**sinl( )**’.
But the trigonometric functions are notoriously slow to compute.

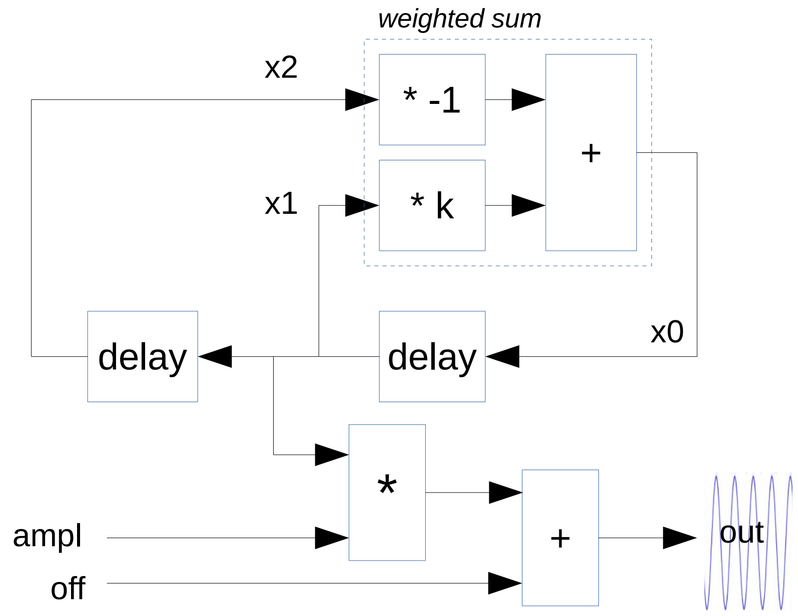
This is a great opportunity for a first exercise.

An Example: a Second Order Resonator



This 2-Pole Resonator is Known to Be an Excellent Oscillator

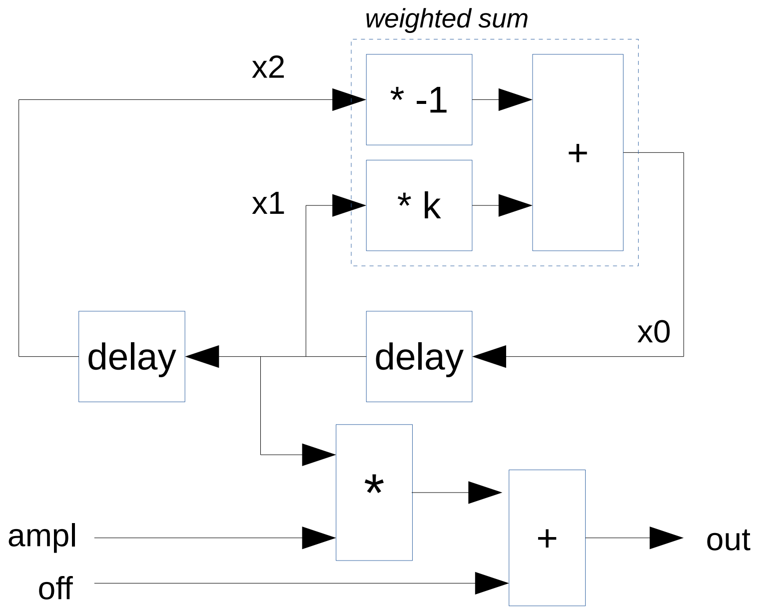
```
#* The resonator is implemented as a 2-pole filter described
#* by the difference equation:
#*
#*    X[n]   = 2.0*cos(2Pi*freq/FSL)*X[n-1] - X[n-2]
#*
#* An initial impulse is required to start the oscillation.
#*
#*    X[n-1] = sin(phase)
#*    X[n-2] = sin(phase - (2Pi*freq/FSL))

...

declare  (analog)   x0                    // the resonator is analog !

dvar   k  2.0 * cos(_2pi_*(freq/FSL))

node  x0  wsum      k  x1   -1.0  x2
node  x1  delay     x0
node  x2  delay     x1
node   s  gain      ampl  x1
node out  offset    off   s

init  x1  sin(phase)                      // output of delay x0 -> x1
init  x2  sin(phase-(_2pi_*(freq/FSL)))   // output of delay x1 -> x2

...
```



This is an Opportunity to Introduce Hierarchy: the Cell

file 'osc.net'

```
cell_interface  $out  $off $ampl  $freq $phase

#* The resonator is implemented as a 2-pole filter described
#* by the difference equation:
#*
#*    X[n]   = 2.0*cos(2Pi*freq/FSL)*X[n-1] - X[n-2]
#*
#* An initial impulse is required to start the oscillation.
#*
#*    X[n-1] = sin(phase)
#*    X[n-2] = sin(phase - (2Pi*freq/FSL))

declare  (analog)    $off  $ampl $freq $phase
declare  (constant)  $freq $phase                // not such modulation

declare  (analog)    $x0                         // the resonator is analog !

dvar   $k  2.0 * cos(_2pi_*($freq/FSL))

node  $x0  wsum       $k  $x1   -1.0  $x2
node  $x1  delay      $x0
node  $x2  delay      $x1
node   $s  gain       $ampl  $x1
node $out  offset     $off   $s

init  $x1  sin($phase)                           // output of delay x0 -> x1
init  $x2  sin($phase-(_2pi_*($freq/FSL)))       // output of delay x1 -> x2
```



The Cell Expansion, a Simple Process

file 'osc.net'

```
cell_interface $out $off $ampl $freq $phase

declare  (analog)     $off  $ampl $freq $phase
declare  (constant)   $freq $phase
declare  (analog)     $x0
dvar     $k   2.0 * cos(_2pi_*($freq/FSL))
node    $x0   wsum        $k  $x1   -1.0  $x2
node    $x1   delay       $x0
node    $x2   delay       $x1
node     $s   gain        $ampl  $x1
node   $out   offset      $off   $s
init    $x1   sin($phase)
init    $x2   sin($phase-(_2pi_*($freq/FSL)))
```

insert

file 'osc.exp'

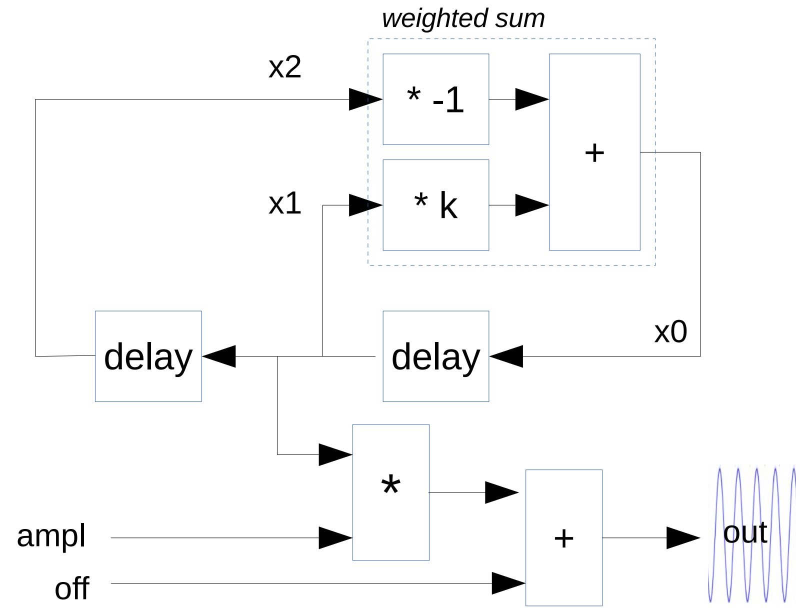
```
title  "a resonator made with a 2-pole filter running at #freq Hz"

header <napa.hdr>
fs     1.0e6
node   out0  sine        off ampl freq phase

#* >>  node  out1 cell pls  "./osc.net"  off ampl freq phase

    declare (analog)        off  ampl freq phase
    declare (constant)      freq phase
    declare (analog)        pls__x0
    dvar    pls__k   2.0 * cos(_2pi_*(freq/FSL))
    node    pls__x0  wsum   pls__k  pls__x1   -1.0  pls__x2
    node    pls__x1  delay  pls__x0
    node    pls__x2  delay  pls__x1
    node    pls__s   gain   ampl  pls__x1
    node    out1     offset off   pls__s
    init    pls__x1  sin(phase)
    init    pls__x2  sin(phase-(_2pi_*(freq/FSL)))

#* <<

node   out2  osc         off ampl freq phase
dvar   per   1.0/freq
dvar   off   0.0
dvar   ampl  1.0
dvar   freq 440.0
dvar   phase  _pi4_
output stdout out0 out1 out2
terminate (2.0*per) <= TIME
```

The expansion,
a job of the PARSER

Let’s Compare both Solutions

file ‘osc.nap’

```
header <napa.hdr>

title  “a resonator made with a 2-pole filter running at #freq Hz”

fs     1.0e6

#* sinewave, method  1
node   out0  sine                  off ampl freq phase

#* sinewave, method 2a and 2b
node   out1  cell pls  “./osc.net”  off ampl freq phase
node   out2  osc                   off ampl freq phase

dvar   per   1.0/freq
dvar   off   0.0
dvar   ampl  1.0
dvar   freq  440.0      // 12 Tone Equal Temperament, A4 @ 440Hz
dvar   phase _pi4_

output stdout  out0  out1 out2

terminate (2.0*per) <= TIME
```



Cross Reference

file 'osc.nap'

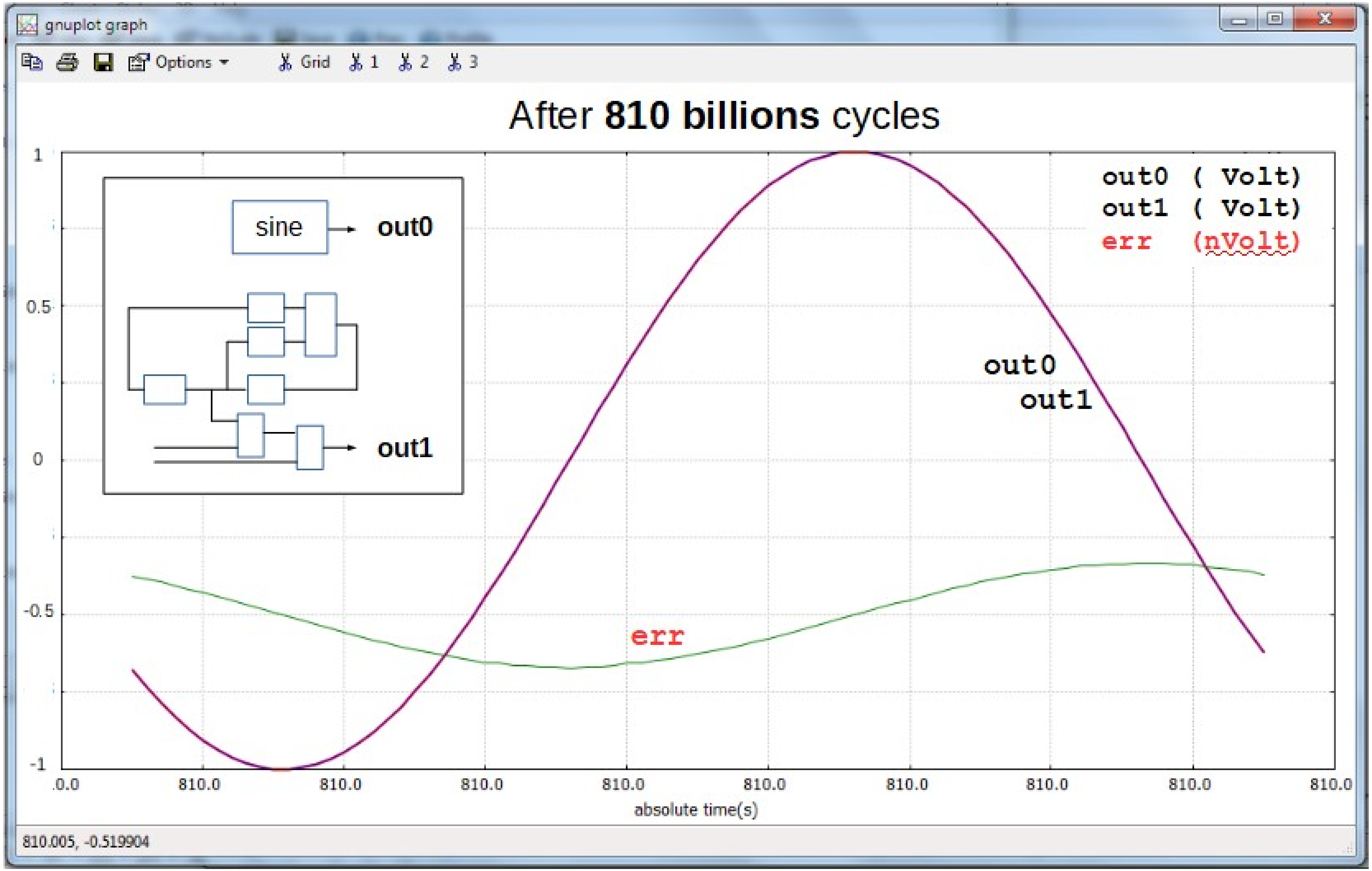
```
1.   header <napa.hdr>
2.
3.   title  "a resonator made with a 2-pole filter running at #freq Hz"
4.
5.   fs     1.0e6
6.
7.   node   out0  sine                       off ampl freq phase
8.   node   out1  cell  pls  "./osc.net"     off ampl freq phase
9.
10.  node   out2  osc                        off ampl freq phase
11.
12.  dvar   per   1.0/freq
13.  dvar   off   0.0
14.  dvar   ampl  1.0
15.  dvar   freq  440.0     // 12  Tone Equal Temperament, A4 @ 440Hz
16.  dvar   phase _pi4_
17.
18.  output stdout out0 out1 out2
19.
20.  terminate (2.0*per) <= TIME
```

file 'osc.net'

```
1.   cell_interface $out $off $ampl $freq $phase
2.
3.   declare  (analog)      $off  $ampl $freq $phase
4.   declare  (constant)    $freq $phase
5.   declare  (analog)      $x0
6.   dvar    $k   2.0 * cos(_2pi_*($freq/FSL))
7.   node   $x0  wsum        $k  $x1   -1.0  $x2
8.   node   $x1  delay       $x0
9.   node   $x2  delay       $x1
10.  node    $s  gain        $ampl  $x1
11.  node  $out  offset      $off   $s
12.  init   $x1  sin($phase)
13.  init   $x2  sin($phase-(_2pi_*($freq/FSL)))
```

```
Administrateur : NAPA Cross Reference: Source File *** osc.nap ***
**************************************************************************************

List of Files
  A. ->   "osc.tmp"
  B. ->   "osc.net"

**************************************************************************************


List of Headers
  A.1              <-   "/Simulate/NapaDos/Hdr/napa.hdr"

Sampling Information
  A.5              <- [ main sampling frequency ]

List of Nodes
  A.7              <-   out0
  A.8     B.11     <-   out1
  A.10             <-   out2
  A.8     B.10     <-   pls__s
  A.8     B.7      <-   pls__x0
  A.8     B.8      <-   pls__x1
  A.8     B.9      <-   pls__x2

List of Variables
  A.14             <-   ampl
  A.15             <-   freq
  A.13             <-   off
  A.12             <-   per
  A.16             <-   phase
  A.8     B.6      <-   pls__k
  A.8     B.12     <- [ init ]
  A.8     B.13     <- [ init ]

List of Declarations
  A.8     B.3      <-   ampl
  A.8     B.3      <-   freq
  A.8     B.4      <-   freq
  A.8     B.3      <-   off
  A.8     B.4      <-   phase
  A.8     B.3      <-   phase
  A.8     B.5      <-   pls__x0

List of IO's
  A.18             <- [ output ]

Terminate
  A.20             <- [ terminate ]

**************************************************************************************
```

Precision


gnuplot graph
Options
Grid
1
2
3
After 810 billions cycles
sine
out0
out1
out0 (Volt)
out1 (Volt)
err (nVolt)
out0
out1
err
1
0.5
0
-0.5
-1
.0.0
810.0
810.0
810.0
810.0
810.0
810.0
810.0
810.0
810.0
absolute time(s)
810.005, -0.519904

Benchmarks

file './osc1.nap'

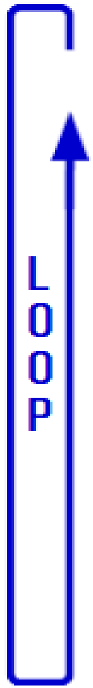
```
header <napa.hdr>

title  "a resonator made with a 2-pole filter"

fs      1.0

node    out   cell s  "./osc.net"  0.0 1.0 440.0 0.0

terminate 1000000000000 <= LOOP_INDEX
```

file './osc2.nap'

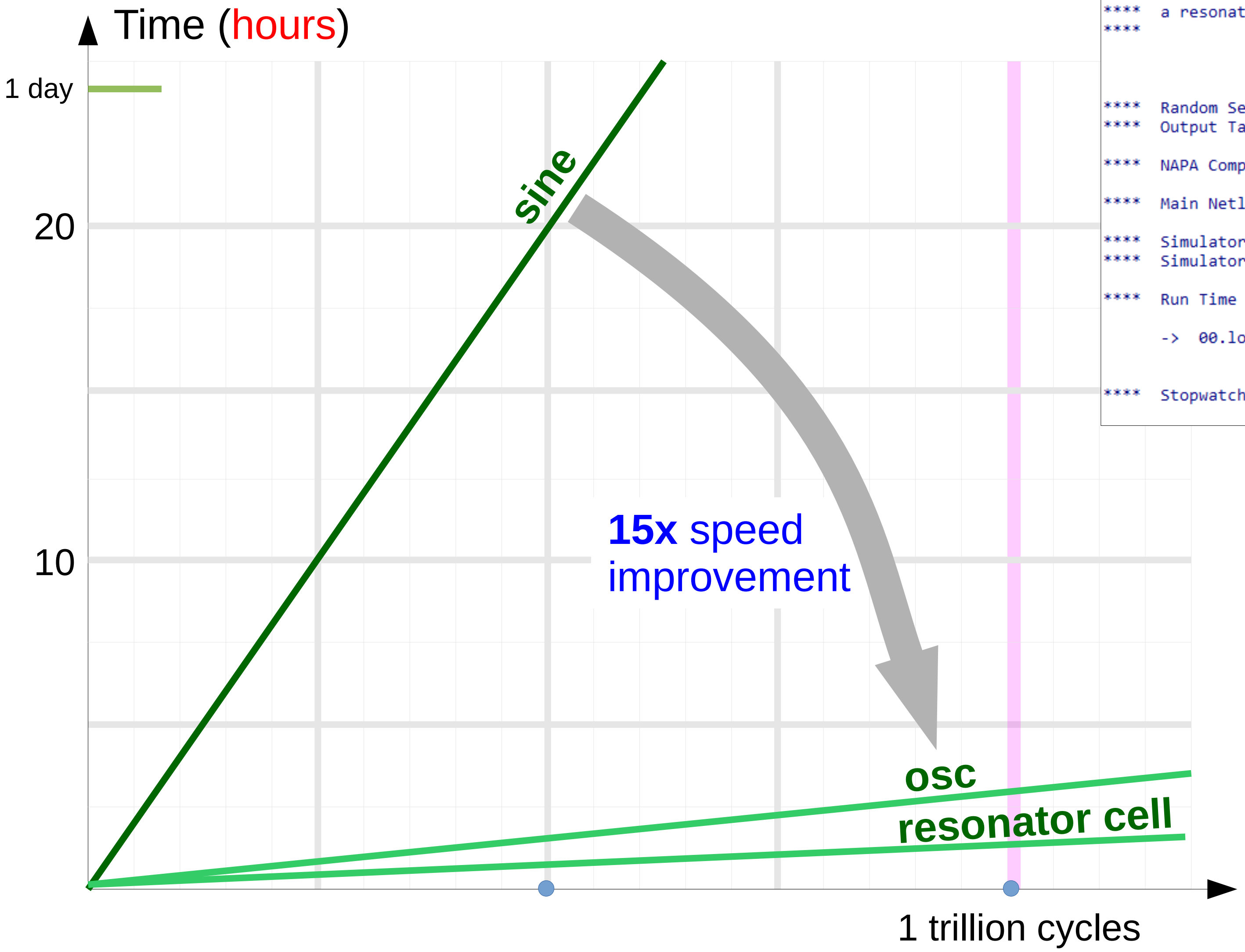
```
header <napa.hdr>

title  "a sinewave"

fs      1.0

node    out   sine   0.0 1.0 440.0 0.0

terminate 1000000000 <= LOOP_INDEX
```

file './osc1.c'

```
...

  /* (start main loop)                              */
  napa_waypoint = 5;

  do {

    napa_abs_time  = napa_abs_loop;

    /* block 1 (update variables) is empty          */

    /* block 2 (update nodes)                       */

    /* always */ {
      d_node_s__x2 = d_node_s__x1;
      d_node_s__x1 = d_node_s__x0;
      d_node_s__x0 = (d_var_s__k) * (d_node_s__x1);
      d_node_s__x0 -= d_node_s__x2;
      d_node_s__s = d_node_s__x1;
      d_node_out = d_node_s__s;
    }

    /* block 3 (output) is empty                    */

    napa_abs_loop++;

  } while (!TERMINATE);

  /* (main loop completed)                          */
  napa_waypoint = 6;

...
```



Speed




```
****
****   a resonator made with a 2-pole filter
****

****   Random Seed [I]   :               778316738 ****
****   Output Tag  [O]   :               779821950 ****

****   NAPA Compiler     :          V4.00 for Win64 ****

****   Main Netlist      :                  00.tmp ****

****   Simulator Time    :             1.00000 Gs ****
****   Simulator Index   :           1 000 000 001 ****

****   Run Time I/O      :                         ****

       ->  00.log                            [   0] ****

****   Stopwatch         :          H00:M00:S09.580 ****
```

More about the Cells

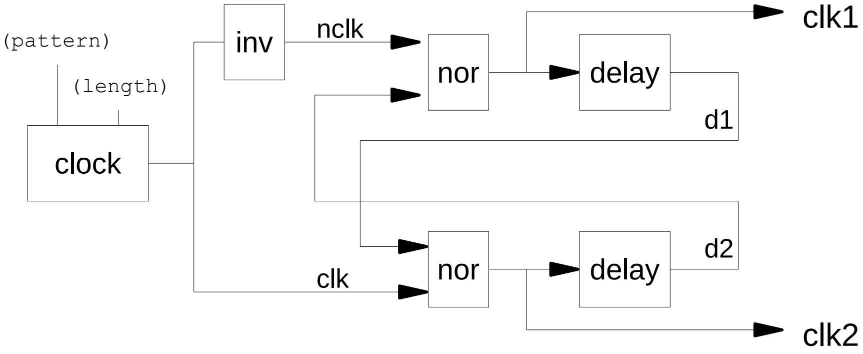
library file '/Simulate/NapaDos/Net/Clock/clock102.net'

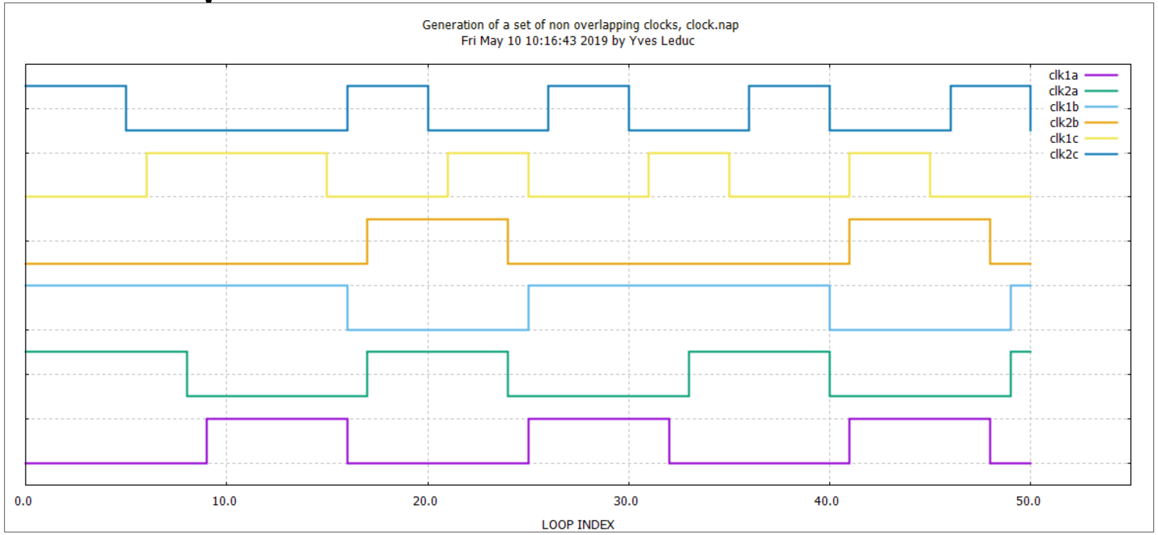
```
cell_interface $dummy $clk1 $clk2 $pattern $length

node   $clk    clock   $pattern $length

node   $nclk   inv     $clk
node   $clk1   nor     $nclk $d2
node   $d1     delay   $clk1
node   $clk2   nor     $clk  $d1
node   $d2     delay   $clk2
```

file 'clock.nap'

```
title   "Generation of a set of non overlapping clocks, $F"

header   <napa.hdr>
fs       1.0

 node  void  cell ca <Clock/clock102.net> clk1a clk2a  "01"    8
 node  void  cell cb <Clock/clock102.net> clk1b clk2b  "110"   8
 node  void  cell cc <Clock/clock102.net> clk1c clk2c  "01.10" 5

output stdout      LOOP_INDEX clk1a clk2a clk1b clk2b clk1c clk2c
terminate  50LL < LOOP_INDEX
```

```
ca__clk = 00000000 11111111 00000000 11111111 00000000 11111..
cb__clk = 11111111 11111111 00000000 11111111 11111111 00000..
cc__clk = 00000 11111 11111 00000 11111 00000 11111 00000 11..
```





A cell may have multiple outputs !
Use the identifier '***void***' for a greater clarity.

A First ‘go with the flow’ Status

Until now, we have defined

the signals and the parameters	*node, dvar, ivar*
a few elements to describe simple netlists	*gain, sum, delay, multiplexer...*
a few simple activation’s elements	*dc, sinewave, ...*
an efficient hierarchical mechanism	*cell*
a simple control of the flow	*do { … } while (!TERMINATE)*

The description is OPEN to host **C** expressions *dc, dvar, ialgebra, terminate, ...*
But we have to manage painfully the resources *header*

We have implemented the capability to CHECK many potential mistakes in the description.
But, we have REDUCED the capability offered by the **C** language.

But we LEFT THE ANALYSIS of the results to external analysis tools through output files.
And it is of course NOT a good idea to postprocess the millions lines of an external file to produce a result.

Additional Control of the **C** Code using the **C** Macro Preprocessor

'mac.nap'

```
...

directive   LOGFILE

directive   MAGIC  0xABADCAFE
directive   str    "Mary has a little lamb"

directive   tangent(x,y)    (sin(x)/cos(y))


debug       FFT  TOOL

...
```

'mac.c'

```
...

#define   LOGFILE

#define   MAGIC   0xABADCAFE
#define   str     "Mary has a little lamb"

#define   tangent(x,y)   (sin(x)/cos(x))


#define   DEBUG_MODE_FFT
#define   DEBUG_MODE_TOOL

...
```

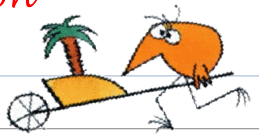
FYI, the '**directive LOGFILE**' triggers the generation of a log file during the simulation.
The code activated by this directive is located in file '*Simulation/Napados/Hdr/napa.hdr*'

Manual or Automatic Inclusion of Header Files

Function '**sqrt( )**', being a native function, does not need any specific action.

The function '**lin2db( )**' is not a native ANSI-C function and the corresponding code must be included.

'manual insertion'



file './lin2db.nap'

```
header     <napa.hdr>
header     <Function/convert1.hdr>

fs         1.0e6

node       ampldb  dalgebra  lin2db(ampl, ref)
node       ampl    dalgebra  sqrt(2.0*(double)(1LL+LOOP_INDEX))
dvar       ref     1.0
output     stderr  ampl  ampldb

terminate 10LL <= LOOP_INDEX
```

library file '/Simulate/Napados/Hdr/Function/convert1.hdr'

```
#ifndef  __FUNCTION_CONVERT1_HDR__
#define  __FUNCTION_CONVERT1_HDR__


double lin2db(double x, double ref);


double lin2db(double x, double ref) {
  if (0.0 >= x) {
    fprintf(stderr, "\nNAPA Run Time Error: (%s)\n", __func__);
    fprintf(stderr,   " Input <%g> is not strictly positive\n",x);
    exit(EXIT_FAILURE);
  }
  if (0.0 >= ref) {
    fprintf(stderr, "\nNAPA Run Time Error: (%s)\n", __func__);
    fprintf(stderr,   " Reference <%g> is not strictly positive\n",ref);
    exit(EXIT_FAILURE);
  }
  return 20.0*log10(x/ref);
}

#endif
```

library file '/Simulate/NapaDos/Hdr/toolbox.hdr'

```
...

#ifdef  COMPILE_lin2db
# include "/Simulate/NapaDos/Hdr/Function/convert1.hdr"
#endif

...
```

library file '/Simulate/NapaDos/Hdr/Function/convert1.hdr'

```
#ifndef   __FUNCTION_CONVERT1_HDR__
#define   __FUNCTION_CONVERT1_HDR__


double lin2db(double x, double ref);


double lin2db(double x, double ref) {
  if (0.0 >= x) {
    fprintf(stderr, "\nNAPA Run Time Error: (%s)\n", __func__);
    fprintf(stderr,   " Input <%g> is not strictly positive\n",x);
    exit(EXIT_FAILURE);
  }
  if (0.0 >= ref) {
    fprintf(stderr, "\nNAPA Run Time Error: (%s)\n", __func__);
    fprintf(stderr,   " Reference <%g> is not strictly positive\n",r);
    exit(EXIT_FAILURE);
  }
  return 20.0*log10(x/ref);
}

#endif
```

file 'lin2db.nap'

```
header     <napa.hdr>
header     <toolbox.hdr>

fs          1.0e6
node        ampl    dalgebra   sqrt(2.0*(double)(1LL+LOOP_INDEX))
node        ampldb  dalgebra   lin2db(ampl, ref)
dvar        ref     1.0
output      stderr  ampl  ampldb

terminate 10LL <= LOOP_INDEX
```

file 'lin2db.c'

```
#define      COMPILE_lin2db

#include  "/Simulate/NapaDos/Hdr/napa.hdr"
#include  "/Simulate/NapaDos/Hdr/toolbox.hdr"
...
do {
    napa_abs_time  = napa_abs_loop * 1.0e-6;

    d_node_ampl   = sqrt(2.0*(double)(1LL+LOOP_INDEX));
    d_node_ampldb = lin2db(d_node_ampl, d_var_ref);
    fprintf(stdout, "%.15Le", napa_abs_time);
    fprintf(stdout, "  % .12e % .12e\n", d_node_ampl, d_node_ampldb);
    abs_loop++;

  } while (!(10LL < LOOP_INDEX));
...
```

The netlist compiler indicates to the **C** compiler that the function '**lin2db( )**' has been detected in the netlist by inserting a **C** preprocessor macro

'**COMPILE_lin2db**'.

By registering once the address of the file containing the code corresponding to the function 'lin2db( )' in a library file '**toolbox.hdr**', there is no more need to add the corresponding 'header' instruction !

The Limitations of **C** expressions

The nodes '*dalgebra*' and '*ialgebra*' suffer from major limitations.

Being a **C** code, only limited verifications are possible.
It is difficult to have a variable number of parameters in functions.
The one-lined **C** expression supports only a single output.

Elaborate functions are difficult to be written.

And the creation of independent instantiations are cumbersome.

file './badcounter.nap'

```
header <napa.hdr>
header "./badcounter.hdr"

fs    1.0e6

ivar v1 counter(1)
node n2 ialgebra counter(2)
node n3 ialgebra counter(3)

output stderr  v1 n2 n3

terminate 4LL < LOOP_INDEX
```

file './badcounter.hdr'

```
#ifndef  __BADCOUNTER_HDR__
#define  __BADCOUNTER_HDR__

double counter(long long stp);

double counter(long long stp) {
  static long long int oldcnt = 0LL;
  long long int cnt;
  cnt     = oldcnt;
  oldcnt += stp;
  return cnt;
}
#endif
```

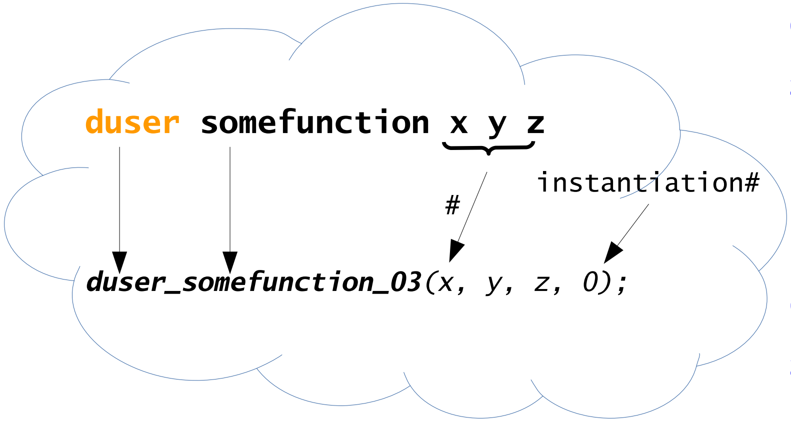
(screen)

```
****
****  BADCOUNTER
****

# (time domain output)
# absolute_time(s)              v1          n2          n3
0.000000000000000e+000           0           1           3
1.000000000000000e-006           0           6           8
2.000000000000000e-006           0          11          13
3.000000000000000e-006           0          16          18
# absolute_time(s)              v1          n2          n3
```

wrong !

Unleashing **C** Expressions: Nodes ‘*duser*’ and ‘*iuser*’ (1/4)

```
header <napa.hdr>
header "./goodcounter.hdr"

fs     1.0

0 node   c1   iuser    goodcounter   10 inc
1 node   c2   iuser    goodcounter
2 node   c3   iuser    goodcounter       -1

ivar   inc   4

output       stderr   c1 c2 c3

terminate  10LL <= LOOP_INDEX
```




```
...
#define   COMPILE_goodcounter   3
```

the number of instantiations of the function in the netlist

```
#include "/Simulate/NapaDos/Hdr/napa.hdr"
#include "./goodcounter.hdr"

long long ivar_inc = 4;

0 check_iuser_goodcounter_02(10, ivar_inc, 0);
1 check_iuser_goodcounter_00( 1);
2 check_iuser_goodcounter_01(-1, 2);

0 init_iuser_goodcounter_02(10, ivar_inc, 0);
1 init_iuser_goodcounter_00( 1);
2 init_iuser_goodcounter_01(-1, 2);

do {
  napa_abs_time = ((double) LOOP_INDEX) / FSL;
0 node_c1 = iuser_goodcounter_02(10, ivar_inc, 0);
1 node_c2 = iuser_goodcounter_00( 1);
2 node_c3 = iuser_goodcounter_01(-1, 2);
  fprintf(stdout, "%.15e", napa_abs_time);
  fprintf(stdout, " % .12e % .12e % .12e\n",i_node_c1,i_node_c2,i_node_c3);
  LOOP_INDEX++;
} while (!(10LL <= LOOP_INDEX));

0 close_iuser_goodcounter_02(10, ivar_inc, 0);
1 close_iuser_goodcounter_00( 1);
2 close_iuser_goodcounter_01(-1, 2);
...
```

file ‘goodcounter.hdr’

```
#ifndef  __GOODCOUNTER_HDR__
#define  __GOODCOUNTER_HDR__

/* ****************************************************************** */
/**   iuser function goodcounter’ accepts 0, 1 or 2 arguments         */
/**     node a iuser goodcounter ini step                             */
/**     node b iuser goodcounter     step          (ini = 0)          */
/**     node c iuser goodcounter                   (ini = 0, step = 1) */

/* ** MACRO FUNCTIONS DEFINITIONS ************************************ */

#define       iuser_goodcounter_02(a,b, c)       iuser_goodcounter(  b, c)
#define check_iuser_goodcounter_02(a,b, c) check_iuser_goodcounter(      )
#define reset_iuser_goodcounter_02(a,b, c) reset_iuser_goodcounter(a,    c)
#define  init_iuser_goodcounter_02(a,b, c)  init_iuser_goodcounter(a,    c)
#define close_iuser_goodcounter_02(a,b, c) close_iuser_goodcounter(      )

#define       iuser_goodcounter_01(a, b)         iuser_goodcounter(  a, b)
#define check_iuser_goodcounter_01(a, b)   check_iuser_goodcounter(      )
#define reset_iuser_goodcounter_01(a, b)   reset_iuser_goodcounter(0,    b)
#define  init_iuser_goodcounter_01(a, b)    init_iuser_goodcounter(0,    b)
#define close_iuser_goodcounter_01(a, b)   close_iuser_goodcounter(      )

#define       iuser_goodcounter_00( a)           iuser_goodcounter(  1, a)
#define check_iuser_goodcounter_00( a)     check_iuser_goodcounter(      )
#define reset_iuser_goodcounter_00( a)     reset_iuser_goodcounter(0,    a)
#define  init_iuser_goodcounter_00( a)      init_iuser_goodcounter(0,    a)
#define close_iuser_goodcounter_00( a)     close_iuser_goodcounter(      )


/* ** FUNCTIONS PROTOYPES ******************************************** */

long long     iuser_goodcounter(              long long stp, int id);
void    check_iuser_goodcounter(              void                 );
void    reset_iuser_goodcounter(long long ini,               int id);
void     init_iuser_goodcounter(long long ini,               int id);
void    close_iuser_goodcounter(              void                 );
```

Static memory allocation

```
/* ** GLOBAL VARIABLES ************************************************ */

long long  goodcounter_cnt[ COMPILE_goodcounter ];


/* ** FUNCTIONS DEFINITIONS ******************************************* */

long long   iuser_goodcounter(long long stp, int id) {
  long long int cnt;
  cnt = goodcounter_cnt[ id ];
  goodcounter_cnt[ id ] += stp;
  return cnt;
}

void  check_iuser_goodcounter(void) {
  return;
}

void  reset_iuser_goodcounter(long long ini, int id) {
  goodcounter_cnt[ id ] = ini;
  return;
}

void   init_iuser_goodcounter(long long ini, int id) {
  reset_iuser_goodcounter(ini, id);
  return;
}

void  close_iuser_goodcounter(void) {
  return;
}


/* ******************************************************************** */

#endif
```

Individual parameter

A Second 'go with the flow' Status

The node '*duser*' and '*iuser*' simplify considerably the code to be written to implement a function.
Not a rocket science, it is simply an efficient 'divide and conquer' approach.

(btw, it is not too far from the concepts of an object oriented language)

This is expanding the netlist language considerably without adding complexity.

We can write functions with a variable number of arguments,
and we can also define easily default values.

iuser function 'goodcounter' *accepts 0, 1 or 2 arguments*

```
node a iuser goodcounter ini step
node b iuser goodcounter     step      <----------  (ini = 0)
node c iuser goodcounter               <----------  (ini = 0, step = 1)
```

We will add now a few other features to simplify the netlist description
and the work of the designer of user's functions !

Unleashing **C** Expressions: Options (3/4)

```
/*  NAPA iuser defined function:  "sequence"
/*
/*  Generates an integer number from an interval of indices.
/*
/*  USAGE:  node  <node_nam>  iuser  sequence       <n2>
/*          node  <node_nam>  iuser  sequence       <n2>  (up)
/*          node  <node_nam>  iuser  sequence       <n2>  (down)
/*          node  <node_nam>  iuser  sequence       <n2>  (shuffle)
/*          node  <node_nam>  iuser  sequence       <n2>  (shuffle)  (aperiodic)
/*          node  <node_nam>  iuser  sequence       <n2>  (shuffle)  (periodic)
/*          node  <node_nam>  iuser  sequence  <n1> <n2>
/*          node  <node_nam>  iuser  sequence  <n1> <n2>  (up)
/*          node  <node_nam>  iuser  sequence  <n1> <n2>  (down)
/*          node  <node_nam>  iuser  sequence  <n1> <n2>  (shuffle)                  */
/*          node  <node_nam>  iuser  sequence  <n1> <n2>  (shuffle)  (aperiodic)     */
/*          node  <node_nam>  iuser  sequence  <n1> <n2>  (shuffle)  (periodic)      */
/*                                                                                   */
/*  Where   n1   long integer, limit min of interval [n1, n2]  (default: 0)    @ init   */
/*          n2   long integer, limit max of interval [n1, n2]                  @ init   */
/*                                                                                   */
/*  Optional qualifiers are '(up)', '(down)', or '(shuffle)'.                  @ init   */
/*  It is possible to periodize the sequence '(periodic)' or '(aperiodic)',    @ init   */
/*  when using the option '(shuffle)'.                                               */
/*                                                                                   */
/*  defaults are '(up)' and '(aperiodic)'.                                           */
```

Options extend considerably the function capabilities.

We propose that the NAPA compiler reduces the programmation effort.

libray file '/Simulate/Napados/Hdr/User/sequence.hdr'

```c
...

int  sequence_typ[ COMPILE_iuser_sequence ];
int  sequence_per[ COMPILE_iuser_sequence ];

void check_iuser_sequence(long n1, long n2, int id) {
  if (n2 <= n1) {
    fprintf(stderr, "\nNAPA Run Time Error: (sequence[%d]) Invalid interval\n", id);
    exit(EXIT_FAILURE);
  }

  sequence_typ[id] = 1;          /*  default is (up)         */
  sequence_per[id] = false;      /*  default is (aperiodic)  */

  if        (ISOPTION("iuser_sequence", id, "up"       )) { sequence_typ[id] =  1;
  } else if (ISOPTION("iuser_sequence", id, "shuffle"  )) { sequence_typ[id] =  0;
  } else if (ISOPTION("iuser_sequence", id, "down"     )) { sequence_typ[id] = -1;    }

  if        (ISOPTION("iuser_sequence", id, "periodic" )) { sequence_per[id] = true;
  } else if (ISOPTION("iuser_sequence", id, "aperiodic")) { sequence_per[id] = false; }

  if     (ISNOTOPTION("iuser_sequence", id             )) {
    fprintf(stderr, "\nNAPA Run Time Error: (sequence[%d]) Option is not valid\n", id);
    exit(EXIT_FAILURE);
  }
  return;
}


long long iuser_sequence(long n1, long n2, int id) {
  long long n;
  ...
  if (1 == sequence_typ[id]) {     /* code for a sequence with a positive slope */
    n = ...
  }
  return n;
}

...
```

('napatool.hdr' combines 'napa.hdr" and 'toolbox.hdr')

'file.nap'

```
header   <napatool.hdr>

fs      1.0
node    sa iuser sequence n1 n2            (down)
node    sb iuser sequence n3   (shuffle) (periodic)
ivar    n1     1
ivar    n2   100
ivar    n3    10
output stdout sa   sb

terminate   20LL   <=   LOOP_INDEX
```

options

"file.c"

```c
...
#define  COMPILE_iuser_sequence   2
...
#define  ISOPTION(f,i,o)       check_for_option(f,i,o)
#define  ISNOTOPTION(f,i)    (ISOPTION(f,i,"_another_") )
...
do {
  ...
  node_sa = iuser_sequence_02(ivar_n1, var_n2, 0);
  node_sb = iuser_sequence_01(ivar_n3, 1);
  ...
} while (!(20LL <= LOOP_INDEX));
...
int check_for_option(char *fun, long id, char *opt) {
  int yesno = false;
  ...
  (ad-hoc crosslist of function instantiations and options)
  ...
  return yesno;
}
...
```

Unleashing **C** Expressions (4/4)

Enabling multiple output functions

file ‘foo.hdr’

```
...

/* An example of a MULTIPLE OUTPUT ‘duser’  function          */

/*  node  tag  duser  foo  x y z                              */

/*  node   na  duser  foo   tag  (outa)                       */
/*  node   nb  duser  foo   tag  (outb)                       */
/*  node   nc  duser  foo   tag  (outc)                       */

#define check_duser_foo_01(a,b)       check_duser_extract_foo(b)
#define check_duser_foo_03(a,b,c,d)   check_duser_compute_foo(a,b,c,d)

#define init_duser_foo_01(a,b)
#define init_duser_foo_03(a,b,c,d)    init_duser_compute_foo(a,b,c,d)

#define reset_duser_foo_01(a,b)
#define reset_duser_foo_03(a,b,c,d)   reset_duser_compute_foo(a,b,c,d)

#define duser_foo_01(a,b)             duser_extract_foo(a,b)
#define duser_foo_03(a,b,c,d)         duser_compute_foo(a,b,c,d)

#define close_duser_foo_01(a,b)
#define close_duser_foo_03(a,b,c,d)

...
```

```
...

int    foo_opt[COMPILE_duser_foo];
double foo_a[COMPILE_duser_foo];
double foo_b[COMPILE_duser_foo];
double foo_c[COMPILE_duser_foo];
...

double duser_compute_foo(double a, double b, double c, int id) {
  ...
  foo_a[id] = ... ;
  foo_b[id] = ... ;
  foo_c[id] = ... ;
  return (double) id;   /* a ‘duser’ function has to return a double! */
}
...

void check_duser_extract_foo(int id) {
  foo_opt[id] =  0;           /* option (outa) is the default         */
  if         (ISOPTION("duser_foo", id, "outa")) {   foo_opt[id] =  0;
  } else if (ISOPTION("duser_foo", id, "outb")) {   foo_opt[id] =  1;
  } else if (ISOPTION("duser_foo", id, "outc")) {   foo_opt[id] =  2;
  }
  if      (ISNOTOPTION("duser_foo", id)) {
    fprintf(stderr, "\nNAPA Run Time Error: (foo[%d]) option not valid\n", id);
    exit(EXIT_FAILURE);
  }
  return;
}
...

double duser_extract_foo(double tag, int id) {
  if         (0 == foo_opt[id]) {  return foo_a[(long) tag];
  } else if (1 == foo_opt[id]) {  return foo_b[(long) tag];
  } else if (2 == foo_opt[id]) {  return foo_c[(long) tag];
  }
}
...
```

the tag is emitted

the tag is received

Constant and Variable Parameters

screen

From the current definitions, ‘***dvar***’ and ’***ivar***’ are -**CONSTANT**- and as they are defined during the initialization.

They are **never updated by default**.

We will use a specific instruction to update them during the simulation.

file ‘quick.nap’

```
header    <napa.hdr>

title     “a quicktest”

fs        1.0

dvar      vtime              TIME
node      ntime    dalgebra  TIME

output    stderr   ntime vtime

terminate 10LL <= LOOP_INDEX
```

a ‘**dvar**’ or a ‘**ivar**’ are computed only at initialization

a ‘**node**’ is computed all along the simulation

Administrateur : NAPA Compile and Run: Source File *** quick.nap ***

```
[quick]  **** MAC Preprocessor Running ****
[quick]  **** NAPA Compiler    Running ****
[quick]  **** GCC Compiler     Running
[quick]  **** Ad Hoc Simulator Running

****
****   a quicktest
****

# (time domain output)
# absolute_time(s)                          ntime                    vtime
0.000000000000000e+000   0.000000000000e+000   0.000000000000e+000
1.000000000000000e+000   1.000000000000e+000   0.000000000000e+000
2.000000000000000e+000   2.000000000000e+000   0.000000000000e+000
3.000000000000000e+000   3.000000000000e+000   0.000000000000e+000
4.000000000000000e+000   4.000000000000e+000   0.000000000000e+000
5.000000000000000e+000   5.000000000000e+000   0.000000000000e+000
6.000000000000000e+000   6.000000000000e+000   0.000000000000e+000
7.000000000000000e+000   7.000000000000e+000   0.000000000000e+000
8.000000000000000e+000   8.000000000000e+000   0.000000000000e+000
9.000000000000000e+000   9.000000000000e+000   0.000000000000e+000
# absolute_time(s)                          ntime                    vtime

****   Random Seed [I]   :            778843862 ****
****   Output Tag  [O]   :            392220309 ****

****   NAPA Compiler     :      V4.00 for Win64 ****

****   Main Netlist      :            quick.tmp ****

****   Simulator Time    :          9.00000   s ****
****   Simulator Index   :                   10 ****

****   Run Time I/O      :                      ****

       -> stderr                          [  0] ****

****   Stopwatch         :      H00:M00:S00.156 ****

****   Normal Termination                       ****
```

No explicit update, no change !

Updating Parameters (1/2)

screen

There are a few specific instructions to control the parameters '**ivar**' and '**dvar**'.

file 'quick.nap'

```
header      <napa.hdr>

title       "a quicktest, updating a parameter"

fs          1.0

dvar        vtime                 TIME     &update
node        ntime       dalgebra  TIME

output      stderr      ntime vtime

terminate   10LL <= LOOP_INDEX
```

Explicit update

```
Administrateur : NAPA Compile and Run: Source File *** quick.nap ***
[quick]  **** MAC Preprocessor Running ****
[quick]  **** NAPA Compiler    Running ****
[quick]  **** GCC Compiler     Running ****
[quick]  **** Ad Hoc Simulator Running ****

****
****   a quicktest
****

# (time domain output)
# absolute_time(s)                                  ntime                 vtime
0.000000000000000e+000    0.000000000000e+000  0.000000000000e+000
1.000000000000000e+000    1.000000000000e+000  1.000000000000e+000
2.000000000000000e+000    2.000000000000e+000  2.000000000000e+000
3.000000000000000e+000    3.000000000000e+000  3.000000000000e+000
4.000000000000000e+000    4.000000000000e+000  4.000000000000e+000
5.000000000000000e+000    5.000000000000e+000  5.000000000000e+000
6.000000000000000e+000    6.000000000000e+000  6.000000000000e+000
7.000000000000000e+000    7.000000000000e+000  7.000000000000e+000
8.000000000000000e+000    8.000000000000e+000  8.000000000000e+000
9.000000000000000e+000    9.000000000000e+000  9.000000000000e+000
# absolute_time(s)                                  ntime                 vtime

****   Random Seed [I]   :              778844075 ****
****   Output Tag  [O]   :              318115499 ****

****   NAPA Compiler     :         V4.00 for Win64 ****

****   Main Netlist      :               quick.tmp ****

****   Simulator Time    :              9.00000  s ****
****   Simulator Index   :                      10 ****

****   Run Time I/O      :                         ****

       ->  stderr                          [   0] ****

****   Stopwatch         :        H00:M00:S00.156 ****

****   Normal Termination                          ****
```

A few short forms which could be applied to a '**ivar**' or a '**dvar**'

... **&update**
... **&constant** (forbids any update)
... &export

Updating Parameters (2/2)

file ‘update.nap’

```
header     <napa.hdr>

title      “variables and events”

fs         1.0

dvar       v1      TIME
update     v1

dvar       v2      TIME     &update

ivar       v3      -1LL
update     v3      LOOP_INDEX / 2LL

event      e4      ISTIME(9.5) || (90.0 < TIME)

output     stderr  v1 v2 v3 e4   when   e4
output     stdout  v1 v2 v3

terminate  100.0 <= TIME
```

equivalent definitions

file ‘update.c’

```
...
#define ISTIME(t)  (((FSL*((t)-TIME))<0.5) && ((FSL*(TIME-(t)))<=0.5))
...
```

screen

```
****
****  variables and events
****


# (time domain output)
# absolute_time(s)                         v1                   v2         v3         e4
1.000000000000000e+001   1.000000000000e+001  1.000000000000e+001          5          1
9.100000000000000e+001   9.100000000000e+001  9.100000000000e+001         45          1
9.200000000000000e+001   9.200000000000e+001  9.200000000000e+001         46          1
9.300000000000000e+001   9.300000000000e+001  9.300000000000e+001         46          1
9.400000000000000e+001   9.400000000000e+001  9.400000000000e+001         47          1
9.500000000000000e+001   9.500000000000e+001  9.500000000000e+001         47          1
9.600000000000000e+001   9.600000000000e+001  9.600000000000e+001         48          1
9.700000000000000e+001   9.700000000000e+001  9.700000000000e+001         48          1
9.800000000000000e+001   9.800000000000e+001  9.800000000000e+001         49          1
9.900000000000000e+001   9.900000000000e+001  9.900000000000e+001         49          1
1.000000000000000e+002   1.000000000000e+002  1.000000000000e+002         50          1
# absolute_time(s)                         v1                   v2         v3         e4


****  Random Seed [I]  :            778870259 ****
****  Output Tag  [O]  :            518885203 ****

****  NAPA Compiler    :      V4.00 for Win64 ****

****  Main Netlist     :           update.tmp ****

****  Simulator Time   :            100.000 s ****
****  Simulator Index  :                  101 ****

****  Run Time I/O     :                      ****

      -> stderr                         [  0] ****
      -> stdout                         [  0] ****


****  Stopwatch        :       H00:M00:S00.312 ****
```

An ‘***event***’ is an ‘***ivar***’ updated automatically.

NB: an ‘***update***’ refers **ALWAYS** to the definition of the parameter and not to any previous update!

An Important Tool: the SARC Engine (Jacques Mequin)

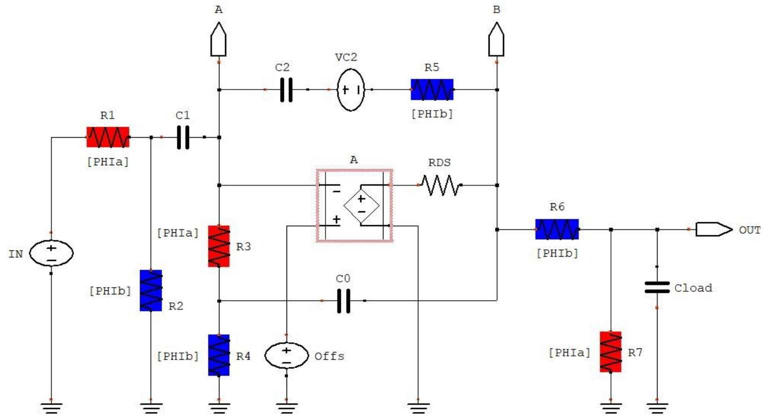
Similarly to the ***Laplace transfer*** generations, the consultant has also developed the generation of ***State-Space models*** (aka *A B C D* matrices)

Consequently, he has implemented a ***Semi-Analytical Recursive Convolution*** algorithm (aka **SARC**) to perform simulations of ***Linear Time Invariant*** circuits as simply as **Matrix Algebra** without the hassle of tuning iteration limits nor convergences

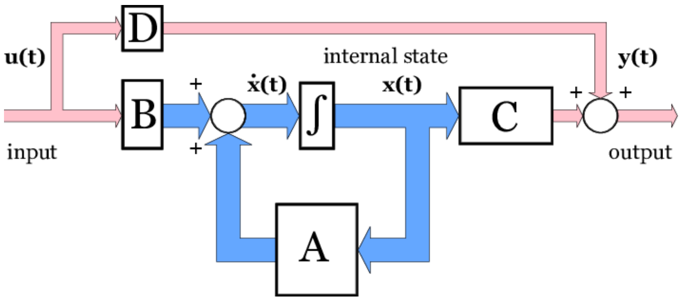
Issued from a ***schematic editor***, the circuit is analyzed and the **A B C D** matrices generated as the result of thousands lines of Maxima

Finally, a ".h" file is produced defining the various arrays of coefficients necessary for the effective simulations

SARC models are multiple-IN/multiple-OUT (MIMO)
Voltages / Currents / Watts / Joules / Fluxes / Charges can be computed
component values can be changed on-the-fly
capacitance charge and inductance flux are preserved
accurate impulse responses can be plotted
characteristic poles can be printed
etc ...



The SARC heuristic and environment deserves by itself a specific presentation



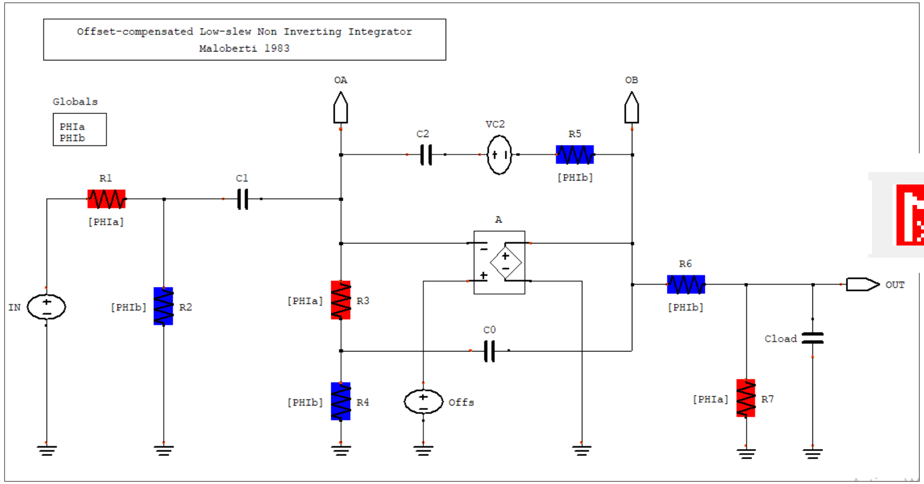
Unlike Laplace (1749), the **State-Space approach** is very recent (1960)..

It is Rudolf Kalman (1930 - 2016) who was one of the greatest gurus in the field of automation ("control engineering"). He used it, for example, during the Apollo space project.

Rudolf Kalman was awarded the National Medal of Science in 2008 by Barack Obama

Modeling a SWC Circuit in ‘s’ Produces a Lot of Parameters

libray file ‘/Simulate/NapaDos/Hdr/Max/SWC_Integrator/Maloberti_Integrator1_NI.sch’



$A = \infty$

wxMaxima screen

$$XFER = (C0\,C1\,C2\,PHIa\,PHIb^2\,(PHIb\,RDS + PHIa\,(RDS - A\,PHIb))\,|s|^3 + C1\,PHIa\,PHIb\,(((C2+C0)\,PHIb + C2\,PHIa)\,RDS - A\,PHIb\,(C2\,PHIb + (C2+C0)\,PHIa))\,|s|^2 - A\,C1\,PHIa\,PHIb\,(PHIb+PHIa)\,|s|)/(C0\,C1$$

$$C2\,Cload\,PHIa\,PHIb^2\,(PHIb+PHIa)\,(PHIb\,RDS + PHIa\,(5\,RDS + (A+2)\,PHIb))\,|s|^4 + PHIb\,(C0\,C1\,C2\,PHIb^3\,RDS + PHIa\,(PHIb\,(5\,C0\,C1\,Cload\,PHIa\,RDS + C2\,(C0\,((A+1)\,Cload + (A+2)\,C1) + C1\,Cload)\,PHIb^2 +$$

$$PHIa\,((C0\,(C1\,(2\,(Cload + (A+2)\,C2) + A\,Cload) + 2\,(A+1)\,C2\,Cload) + (A+4)\,C1\,C2\,Cload)\,PHIb + (C0\,(C1\,(Cload + (A+2)\,C2) + (A+1)\,C2\,Cload) + (A+2)\,C1\,C2\,Cload)\,PHIa)) +$$

$$((C0\,(C1\,(Cload + 7\,C2) + 2\,C2\,Cload) + 2\,C1\,C2\,Cload)\,PHIb^2 + PHIa\,(C2\,(C0\,(4\,Cload + 9\,C1) + 7\,C1\,Cload)\,PHIb + (C0\,(C1\,(2\,Cload + 3\,C2) + 2\,C2\,Cload) + 3\,C1\,C2\,Cload)\,PHIa))\,RDS))\,|s|^3 + (($$

$$(C0\,(2\,C2 + C1) + 2\,C1\,C2)\,PHIb^3 + PHIa$$

$$((C0\,(2\,Cload + 5\,C2 + 6\,C1) + C1\,(Cload + 8\,C2) + 3\,C2\,Cload)\,PHIb^2 + PHIa\,((C0\,(2\,(Cload + 2\,C2) + 5\,C1) + 3\,C1\,(Cload + 2\,C2) + 4\,C2\,Cload)\,PHIb + ((C2+C1)\,Cload + C0\,(C2+C1) + C1\,C2)\,PHIa)))\,RDS +$$

$$PHIb\,((C1 + (A+1)\,C0)\,C2\,PHIb^3 + PHIa\,((C0\,((A+1)\,Cload + 3\,(A+1)\,C2 + (A+2)\,C1) + C1\,(Cload + (A+5)\,C2) + (A+2)\,C2\,Cload)\,PHIb^2 + PHIa$$

$$((C0\,((A+1)\,Cload + 3\,(A+1)\,C2 + (A+3)\,C1) + C1\,(3\,Cload + 2\,(A+3)\,C2) + (2\,A+3)\,C2\,Cload)\,PHIb + (((A+1)\,C2 + C1)\,Cload + C0\,((A+1)\,C2 + C1) + (A+2)\,C1\,C2)\,PHIa)))\,|s|^2 + ($$

$$((3\,C2 + C1 + 2\,C0)\,PHIb^2 + PHIa\,((Cload + 5\,C2 + 3\,(C1 + C0))\,PHIb + (Cload + 2\,C2 + C1 + C0)\,PHIa))\,RDS + ((A+2)\,C2 + C1 + (A+1)\,C0)\,PHIb^3 + PHIa$$

$$((Cload + (3\,A+5)\,C2 + 2\,(2\,C1 + (A+1)\,C0))\,PHIb^2 + PHIa\,((Cload + 4\,(C2+C1) + 3\,A\,C2 + (A+1)\,C0)\,PHIb + ((A+1)\,C2 + C1)\,PHIa)))\,|s| + (PHIb+PHIa)\,(RDS + PHIb + PHIa))$$

$$XFER = -(C0\,C1\,C2\,PHIa^2\,PHIb^3\,|s|^2 + (C1\,C2\,PHIa\,PHIb^3 + (C1\,C2 + C0\,C1)\,PHIa^2\,PHIb^2)\,|s| + C1\,PHIa\,PHIb^2 + C1\,PHIa^2\,PHIb)/((C0\,C1\,C2\,Cload\,PHIa^2\,PHIb^4 + C0\,C1\,C2\,Cload\,PHIa^3\,PHIb^3)\,|s|^3 +$$

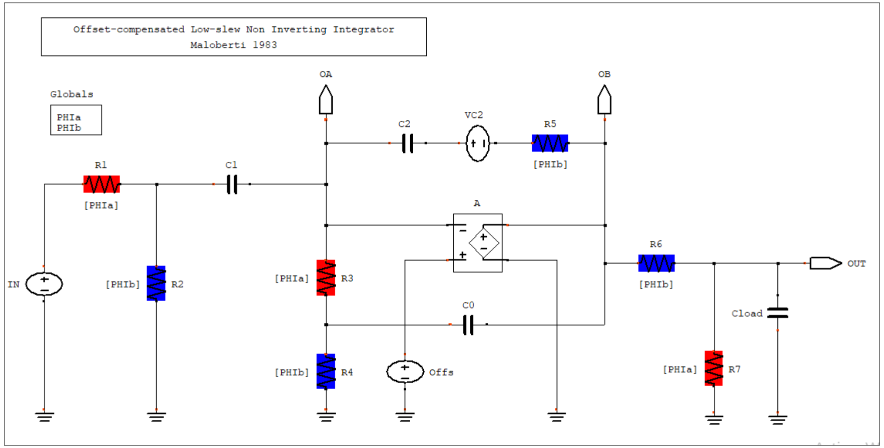
$$((C0\,C2\,Cload + C0\,C1\,C2)\,PHIa\,PHIb^4 + (((C1 + 2\,C0)\,C2 + C0\,C1)\,Cload + 2\,C0\,C1\,C2)\,PHIa^2\,PHIb^3 + ((C1+C0)\,C2\,Cload + C0\,C1\,C2)\,PHIa^3\,PHIb^2)\,|s|^2 +$$

$$(C0\,C2\,PHIb^4 + ((C2+C0)\,Cload + (C1 + 3\,C0)\,C2 + C0\,C1)\,PHIa\,PHIb^3 + ((2\,C2 + C0)\,Cload + (2\,C1 + 3\,C0)\,C2 + C0\,C1)\,PHIa^2\,PHIb^2 + (C2\,Cload + (C1+C0)\,C2)\,PHIa^3\,PHIb)\,|s| + (C2+C0)\,PHIb^3 + (3\,C2 + 2\,C0)$$

$$PHIa\,PHIb^2 + (3\,C2 + C0)\,PHIa^2\,PHIb + C2\,PHIa^3)$$

Passing Parameters by Addresses

library file ':Simulate/NapaDos/Hdr/Max/SWC_Integrator/Maloberti_Integrator1_NI.sch'



file 'myintegrator.dat'

```
data_interface   $C0..2  $Cload  $A $RDS $Offs  $Ron $Roff

dvar  $C0     rand_normal(2.0e-12, 50.0e-15)      // integrator
dvar  $C1     rand_normal(2.0e-12, 20.0e-15)
dvar  $C2     rand_normal(2.0e-12, 20.0e-15)
dvar  $Cload  2.0e-12

dvar  $Ron  100.0                                  // switches
dvar  $Roff   1.0e9

dvar  $Adb   60.0
dvar  $A     DB2LIN($Adb, 1.0)                     // amplifier
dvar  $RDS   10.0e6
dvar  $Offs   1.0e-3
```

$C0..2 is expanded as

$C0 $C1 $C2

library file '/Simulate/NapaDos/Net/SWC_Integrator/Maloberti_Integrator1_NI.net'

```
cell_interface   $dummy  $I  $O $A $B  $Clk1..2  $fildat  $initvalue

data    $fildat  $C0..2 $Cload  $A $RDS $OFFS  $Ron $Roff
dvar    $rsw1    switch_d($Clk1, $Roff, $Ron)  &update
dvar    $rsw2    switch_d($Clk2, $Roff, $Ron)  &update

ganging $miparm[]  $C0..2 $Cload  $rsw1..2  $RDS $A $OFFS

node    $id      duser  sarc Maloberti_Integrator1_NI() $miparm  $I $initvalue
node    ($O)     duser  sarc $id (V@OUT)
node    ($A)     duser  sarc $id (V@A)
node    ($B)     duser  sarc $id (V@B)

...     //  ( a few print here )

header  <Max/SWC_Integrator/Maloberti_Integrator1_NI.hdr>
```

A '***data cell***' is a low level cell specialized in the storage of data

Passing parameters by addresses

2 inputs, multiple outputs duser function 'sarc'

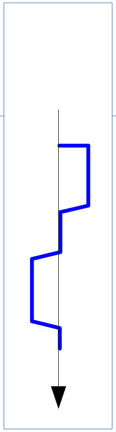
ANSI-C Resource

Running the Simulation

59

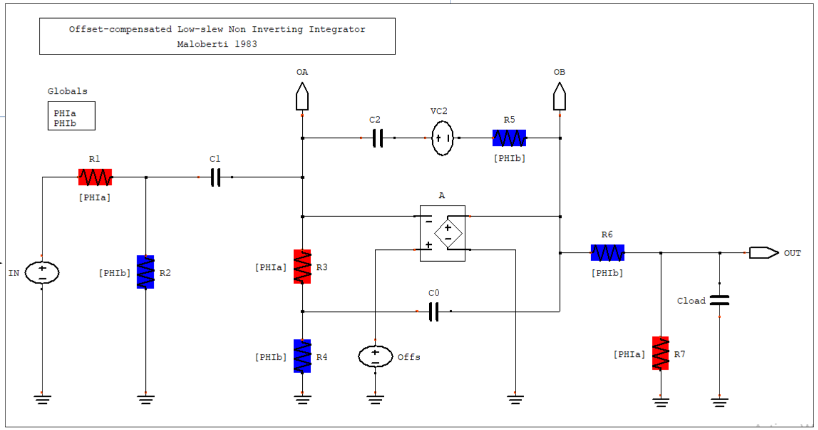
file ‘Maloberti.nap’

```
header     <napatool.hdr>

title      "Simulation of a Maloberti SWC Integrator"

fs         10.0e9

string       fil1  "./mypwl.in"              // the description of the input
string       fil2  "./myintegrator.dat"      // the parameters of my integrator

node Clk1  clock "10"  5000                  // the clocks for the SWC
node Clk2  clock "01"  5000

node  In   cell pwl  <PWL/pwl_d.net>  fil1  1 1.0 0.0  (aperiodic)
node void  cell int  <SWC_Integrator/Maloberti_Integrator1_NI.net> In O A B Clk1..2 fil2  initvalue

dvar   initvalue  0.5

output     stdout  In(_V)  O(_V) A(_V) B(_V)  Clk1..2

terminate 8.0e-6 < TIME

directive LOGFILE
ping
```



file ‘mypwl.in’

```
# PWL (first order interpolation)
#
#  time (s)     vin (V)
   0.0            0.0
   0.01e-6        1.0
   2.60e-6        1.0
   2.90e-6        0.0
   4.60e-6        0.0
   4.90e-6       -1.0
   7.60e-6       -1.0
   7.90e-6        0.0
  99.99e-6        0.0
```



```
****
****   Simulation of a Maloberti SWC Integrator
****

****
****          2.024 pF     <-  int__C0
****          1.997 pF     <-  int__C1
****          2.015 pF     <-  int__C2
****          2.000 pF     <-  int__Cload
****
****          10.00 MOhm   <-  int__RDS
****          1.000 mV     <-  int__OFFS
****
****          500.0 mV     <-  initvalue


****   Random Seed [I]   :               778871003 ****
****   Output Tag  [O]   :               314073249 ****

****   NAPA Compiler     :         V4.00 for Win64 ****

****   Main Netlist      :           Maloberti.tmp ****

****   Simulator Time    :              8.00010 us ****
****   Simulator Index   :                   80002 ****

****   Run Time I/O      :                         ****

       <-  mypwl.in                          [I  ] ****
       ->  Maloberti.log                     [  O] ****
       ->  stdout                            [  O] ****

****   Stopwatch         :         H00:M00:S03.032 ****

****   LOG File Ready    :           Maloberti.log ****


****   Normal Termination                          ****
```

- source:

  file ‘***Maloberti.tmp***’ (from ‘*Maloberti.nap*’)

- one input:

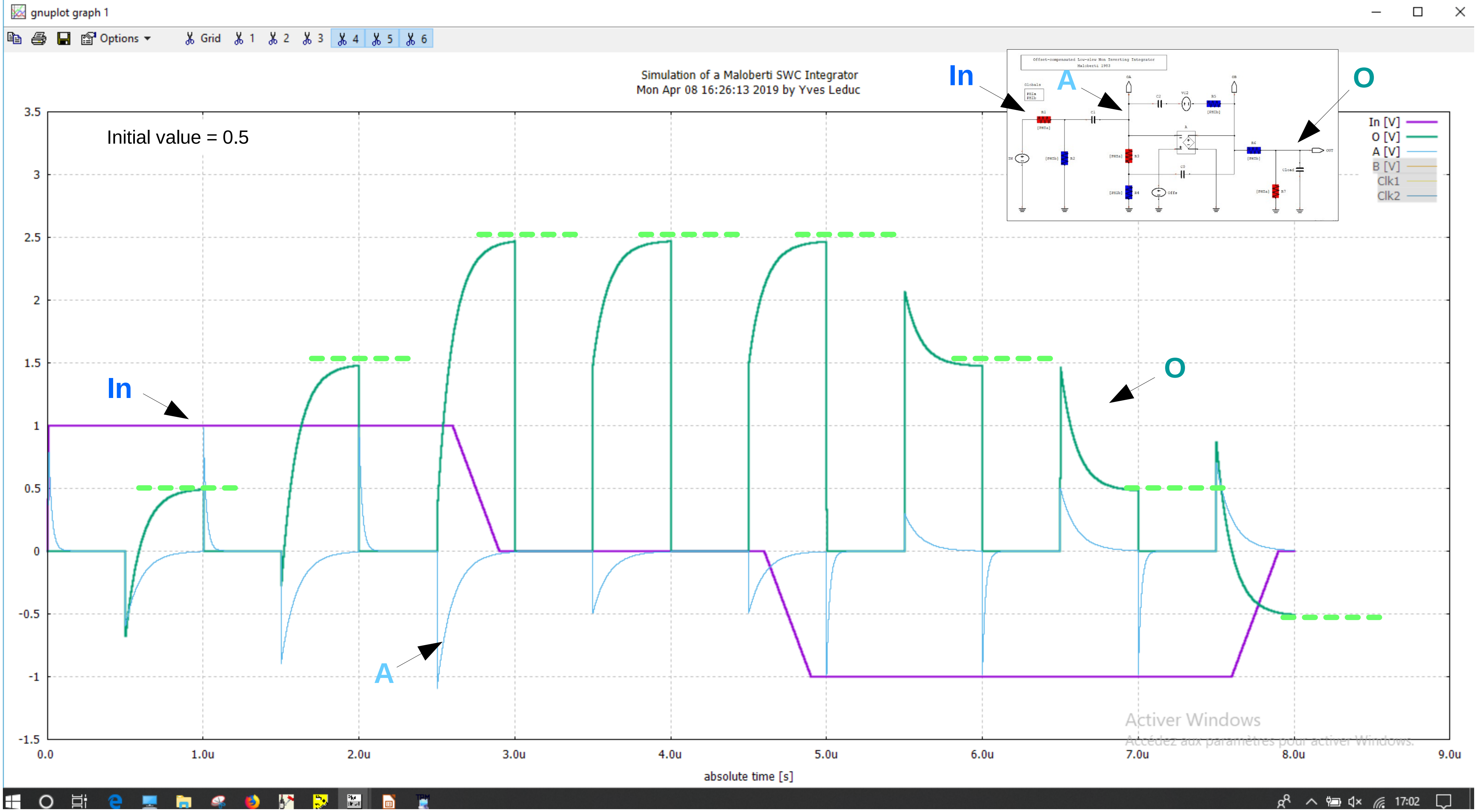
  file ‘***mypwl.in***’

- two outputs:

  a log file ‘***Maloberti.log***’
  ‘stdout’ redirected to file ‘***Maloberti.out***’

- number of cycles: 80 002

- run time: 3.0 seconds


gnuplot graph 1
Options
Grid
Simulation of a Maloberti SWC Integrator
Mon Apr 08 16:26:13 2019 by Yves Leduc
Initial value = 0.5
In
A
O
In [V]
O [V]
A [V]
B [V]
Clk1
Clk2
absolute time [s]
Activer Windows
Accédez aux paramètres pour activer Windows.
17:02

The Files

‘.nap’ file

Mac Preprocessor

‘.tmp’ file

Napa Compiler

‘.c’ file

GCC Compiler (&link)

‘.exe’ file

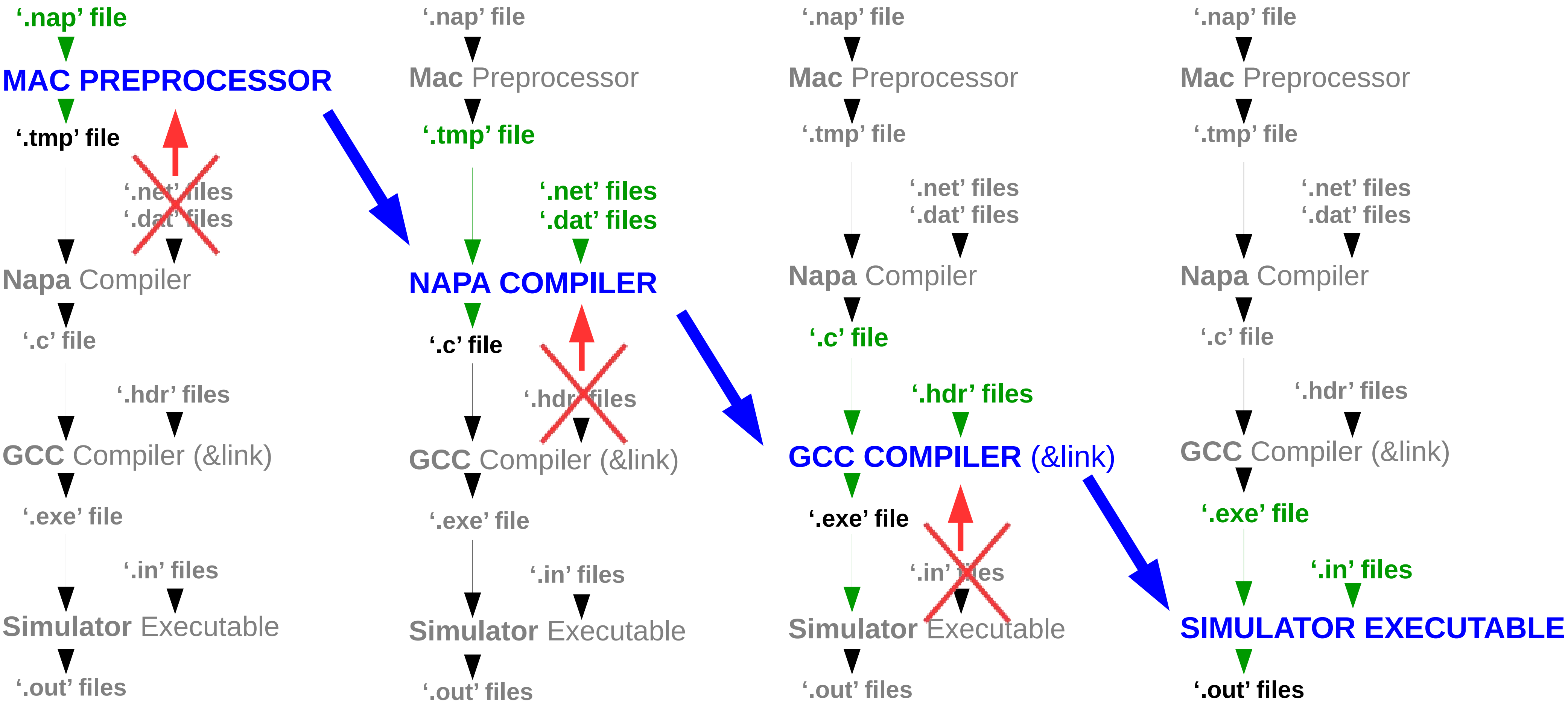
Simulator Executable

‘.out’ files

```
Administrateur: NAPA Compile and Run: Source File *** Maloberti.nap ***

[Maloberti]  **** MAC Preprocessor Running ****
[Maloberti]  **** NAPA Compiler    Running ****
[Maloberti]  **** GCC Compiler     Running ****
[Maloberti]  **** SARC Engine      Linking ****
[Maloberti]  **** Ad Hoc Simulator Running ****


NAPA Ping Information:     function 'duser_pwl()'                  from file "/Simulate/NapaDos/Hdr/User/pwl.hdr"
NAPA Ping Information:     function 'duser_sarc()'                 from file "/Simulate/NapaDos/Hdr/User/sarc.hdr"
NAPA Ping Information:     function 'Maloberti_Integrator1_NI()'   from file "/Simulate/NapaDos/Hdr/Max/SWC_Integrator/MIMO_Maloberti_Integrator1_NI.hdr"
NAPA Ping Information:     function 'print_blank_line()'           from file "/Simulate/NapaDos/Hdr/Function/print_var_and_string.hdr"
NAPA Ping Information:     function 'print_var()'                  from file "/Simulate/NapaDos/Hdr/Function/print_var_and_string.hdr"
NAPA Ping Information:     function 'rand_normal()'                from file "/Simulate/NapaDos/Hdr/Function/random.hdr"
NAPA Ping Information:     function 'switch_d()'                   from file "/Simulate/NapaDos/Hdr/Function/switch.hdr"


 ****
 ****  Simulation of a Maloberti SWC Integrator
 ****

 ****
 ****          2.024 pF     <-  int__C0
 ****          1.997 pF     <-  int__C1
 ****          2.015 pF     <-  int__C2
 ****          2.000 pF     <-  int__Cload
 ****
 ****          10.00 MOhm   <-  int__RDS
 ****          1.000 mV     <-  int__OFFS
 ****
 ****          500.0 mV     <-  initvalue



 ****  Random Seed [I]  :                  778871003 ****
 ****  Output Tag  [O]  :                  314073249 ****

 ****  NAPA Compiler    :             V4.00 for Win64 ****

 ****  Main Netlist     :               Maloberti.tmp ****

 ****  Simulator Time   :                 8.00010 us ****
 ****  Simulator Index  :                      80002 ****

 ****  Run Time I/O     :                            ****

       <-  mypwl.in                              [I  ] ****
       ->  Maloberti.log                         [  O] ****
       ->  stdout                                [  O] ****


 ****  Stopwatch        :             H00:M00:S03.032 ****


 ****  LOG File Ready   :               Maloberti.log ****


 ****  Normal Termination                             ****

[Maloberti]


Press Enter to continue . . .
```

The instruction ‘***ping***’ publishes the location of functions which have been detected in the Napa netlists.

The Simulation Flow



A Third ‘go with the flow’ Status

We have now the elements to activate and run a model.

It is time now to have a solution for the analysis of the results.
We will add a few features to the ‘***duser***’ node concept to build the user defined functions ‘***tool***’.

The analysis tools will be integrated in the simulator as regular nodes.

The Mechanism of the Tool Synchronization

‘tool’ is a user defined function with a synchronization mechanism automatically hooked to the simulator.

A **state machine** is implemented in each tool with 2 states: '**wait**' and '**run**'.

Tasks are numbered. Tools are asked by the simulator to perform a task. Tools are in '**wait**' state until the simulator is sending a message 'start' and all tools are now in '**run**' state.

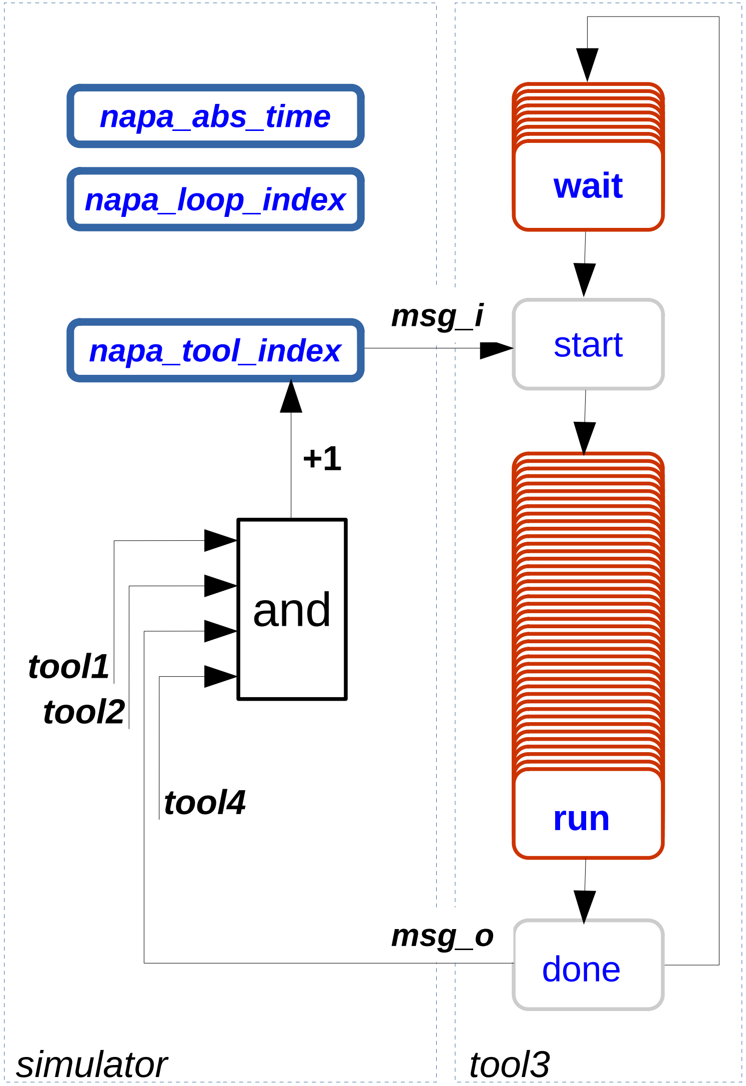
All tools run their own task. The output of the tool is the status of its work. The simulator collects all these status at the end of each simulation cycle.

The simulation continues until all tools have completed the specified task. A tool having accomplished its task stops and is in '**wait**' state.

When all tools have accomplished their task, the simulator sends a message to all of them to start the next task.

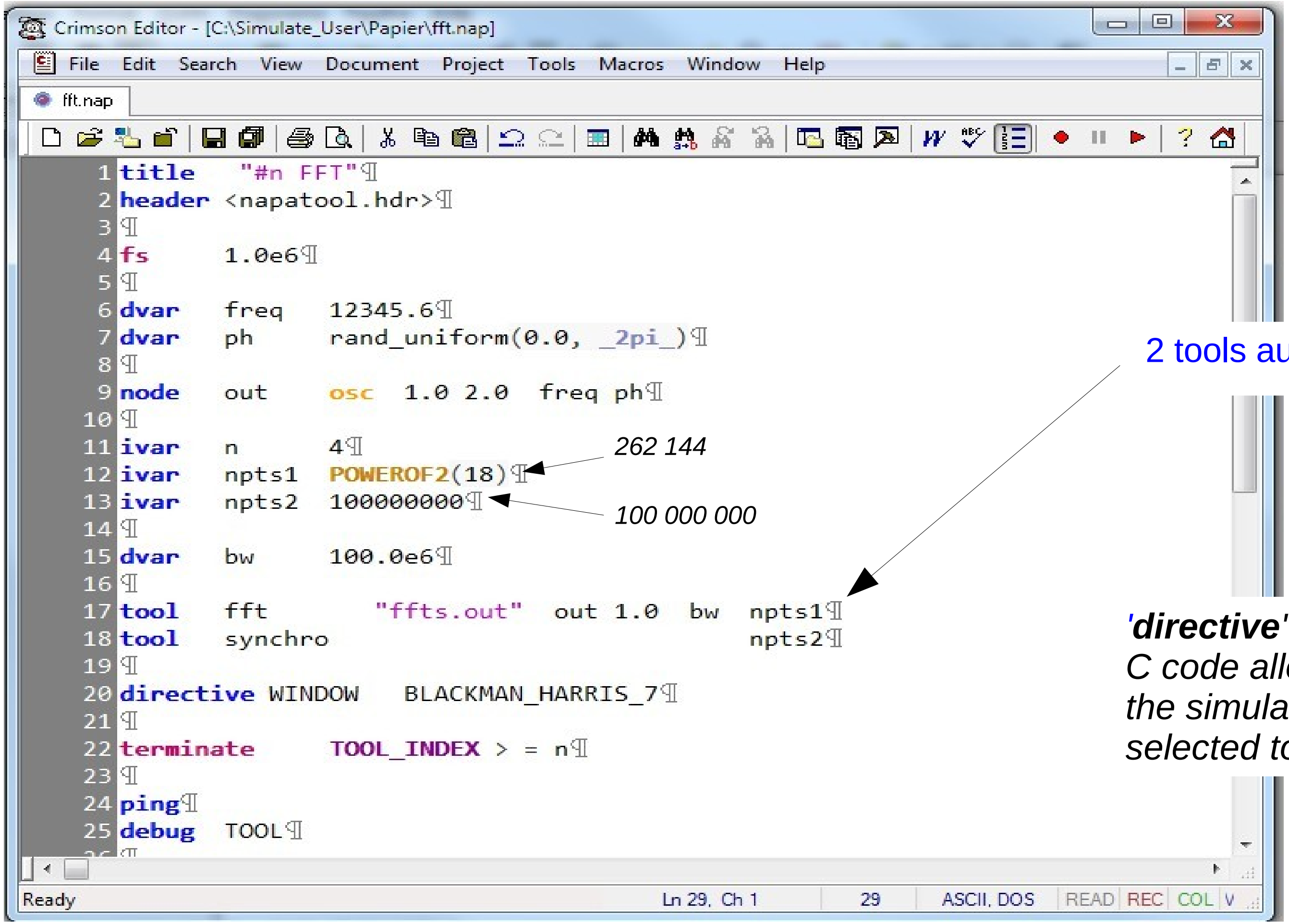
Variable '**napa_tool_index'** handled by the simulator counts the number of tasks already completed and can be used to control the simulation.

(*Note : Macro '**TOOL_INDEX**' is the image of 'napa_tool_index'*)



The Node "*itool*" and its Short Form '*tool*"

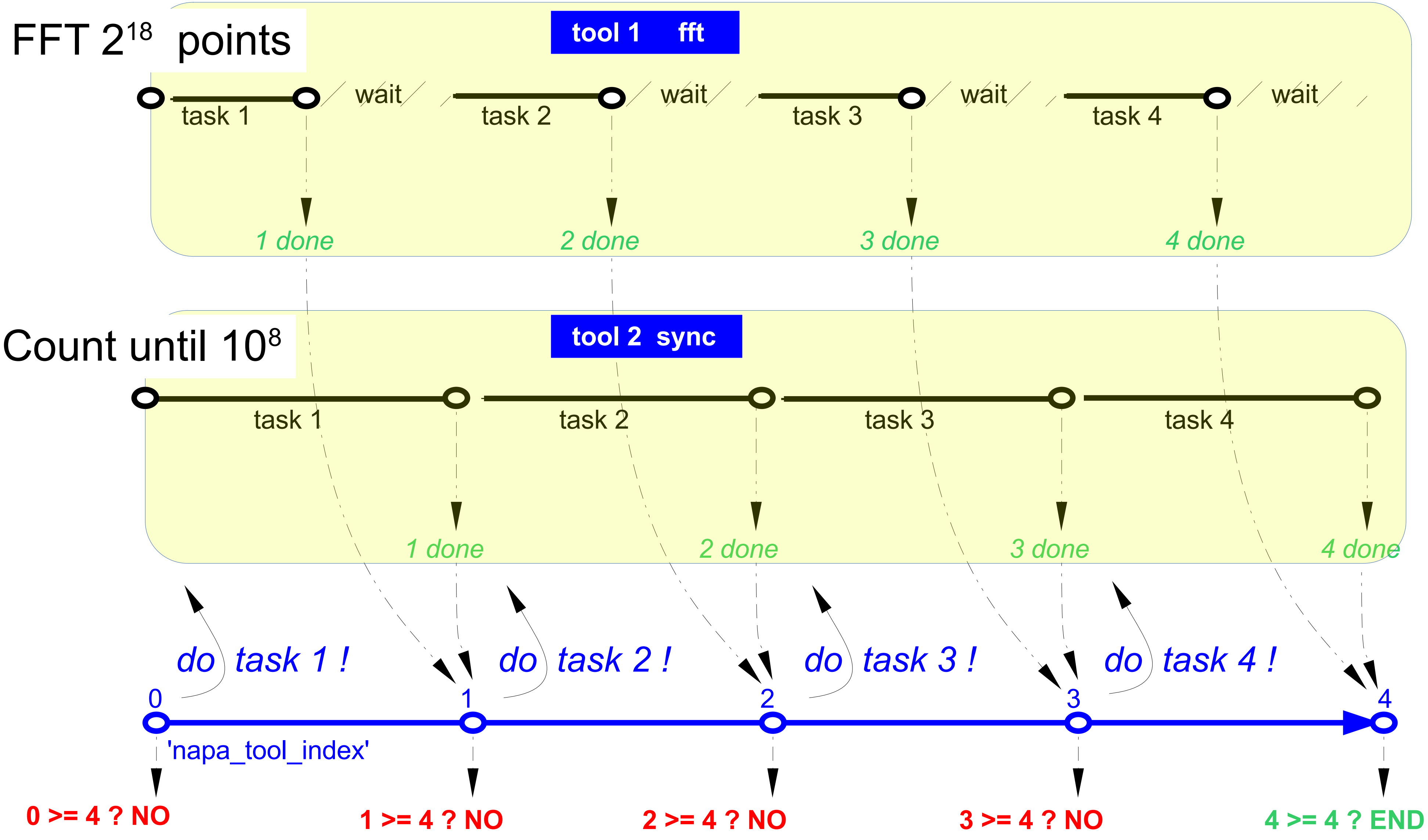
*'**tool**' is a contraction of a regular node syntax: '**node void itool**' and is therefore processed as a node.*



2 tools automatically synchronised

*'**directive**' introduces a macro definition in the C code allowing the preprocessor to configure the simulator. Here a FFT windowing function is selected to replace the default.*

4 FFT of **2^{18}** samples, made every **10^{8}** samples


FFT 2^18 points
tool 1 fft
task 1
wait
task 2
wait
task 3
wait
task 4
wait
1 done
2 done
3 done
4 done
Count until 10^8
tool 2 sync
task 1
task 2
task 3
task 4
1 done
2 done
3 done
4 done
do task 1 !
do task 2 !
do task 3 !
do task 4 !
0
1
2
3
4
'napa_tool_index'
0 >= 4 ? NO
1 >= 4 ? NO
2 >= 4 ? NO
3 >= 4 ? NO
4 >= 4 ? END

The Synchronization: a Cooperation between Simulator and Tools

```
...
#define TOOL_INDEX = napatool
...
do {
napa_abs_time = napa_abs_loop * 1.0e-6L;
...
d_node_out = ... ;
...
napa_msg = &(napa_mailbox[0]);
napa_msg->o = napa_packet;
i_node__void0 = itool_fft_05("ffts.out",d_node_out,1.0,d_var_bw,i_var_npts1, 0);

napa_msg = &(napa_mailbox[
napa_msg->o = napa_packe
i_node__void1 = itool_synchro_01(i_var_npts2, 0);

if ((napa_mailbox[0].o >= napa_packet) && (napa_mailbox[1].o >= napa_packet)) {
napa_rel_loop    = -1.0L;
napa_tool_index = napa_packet;
napa_mailbox[0].i = START;
napa_mailbox[1].i = START;
napa_packet
}
napa_abs_loop
} while (!(1L << TOOL_INDEX));
...
```

The simulator prefills the individual mailboxes

Internally, the tools update the mailboxes to signal the evolution of their work

The simulator tests the content of the mailboxes output which contains the answer of the tools and reacts accordingly

The loop is controlled here by the value of 'napa_tool_index'

Tools, an Example

Sinewaves and White Noise, FFT and TSNR Analysis

70

file “./tool.nap”

```
header <napatool.hdr>
title  "Frequency Domain Analysis"
fs     2.0e+6

dvar   ampl1  0.5
dvar   freq1  2000.0          // fundamental
dvar   ph1    rand_uniform(0.0, _2pi_)

dvar   ampl3  0.05
dvar   freq3  3.0 * freq1     // 3rd harmonic
dvar   ph3    rand_uniform(0.0, _2pi_)

dvar   bwa    10000.0         // bandwidth of interest
ivar   nptsa  POWEROF2(20)

dvar   bwb    10000.0         // bandwidth of analysis
ivar   nptsb  POWEROF2(18)

node   s0     noise 0.0  1.0e-6
node   s1     osc   0.0  ampl1  freq1  ph1
node   s2     osc   0.0  ampl3  freq3  ph3
node   in     sum   s0..2

tool   fft    "fft.out"   in 1.0   bwa   nptsa
tool   tsnr   "tsnr.out"  in 1.0   bwb   nptsb

terminate  3 <= TOOL_INDEX  // 3 FFTs and 3 TSNRs

directive  WINDOW  BLACKMAN_HARRIS_7
#*         debug  IO  TOOL
ping
```

```
[tool]  **** MAC Preprocessor Running ****
[tool]  **** NAPA Compiler    Running ****
[tool]  **** GCC Compiler     Running ****
[tool]  **** Ad Hoc Simulator Running ****


NAPA Ping Information:    function 'itool_fft()'     from file "/Simulate/NapaDos/Hdr/Tool/fft1.hdr"
NAPA Ping Information:    function 'itool_tsnr()'    from file "/Simulate/NapaDos/Hdr/Tool/fft2.hdr"
NAPA Ping Information:    function 'rand_uniform()'  from file "/Simulate/NapaDos/Hdr/Function/random.hdr"

 ****
 ****  Frequency Domain Analysis
 ****


NAPA Tools Information:    (        tsnr[0]) Process  # 000     <- 262143
NAPA Tools Information:    (         fft[0]) Process  # 000     <- 1048575
NAPA Tools Information:    (        tsnr[0]) Process  # 001     <- 1310719
NAPA Tools Information:    (         fft[0]) Process  # 001     <- 2097151
NAPA Tools Information:    (        tsnr[0]) Process  # 002     <- 2359295
NAPA Tools Information:    (         fft[0]) Process  # 002     <- 3145727


 ****  Random Seed [I]  :         777402411 ****
 ****  Output Tag  [O]  :         534469166 ****

 ****  NAPA Compiler    :      V4.00 for Win64 ****

 ****  Main Netlist     :          tool.tmp ****

 ****  Simulator Time   :         1.57286  s ****
 ****  Simulator Index  :         3 145 728 ****
 ****  Tool Index       :                 3 ****

 ****  Run Time I/O     :                   ****

       ->  fft.out                     [  0] ****
       ->  tsnr.out                    [  0] ****


 ****  Stopwatch        :      H00:M00:S01.547 ****


 ****  Normal Termination                    ****


[tool]
```

← 3.1 million cycles

← in 1.5 second

The Synchronization at Work

(NB: run here with instruction '**debug IO TOOL**')

```
NAPA Ping Information:     function 'itool_fft()'     from file "/Simulate/NapaDos/Hdr/Tool/fft1.hdr"
NAPA Ping Information:     function 'itool_tsnr()'    from file "/Simulate/NapaDos/Hdr/Tool/fft2.hdr"
NAPA Ping Information:     function 'rand_uniform()'  from file "/Simulate/NapaDos/Hdr/Function/random.hd

 ****
 ****  Frequency Domain Analysis
 ****


NAPA Debug Information:     (Open   I/O stream )           fft.out for fft             <-  0
NAPA Debug Information:     (Open   I/O stream )          tsnr.out for tsnr            <-  0
NAPA Tools Information:     (            fft[0]) Collect  # 000.000  <- 0
NAPA Tools Information:     (           tsnr[0]) Collect  # 000.000
NAPA Tools Information:     (           tsnr[0]) Process  # 000      <- 262143
NAPA Tools Information:     (           tsnr[0]) End      # 000
NAPA Tools Information:     (            fft[0]) Process  # 000      <- 1048575
NAPA Tools Information:     (            fft[0]) End      # 000
NAPA Tools Information:     (            fft[0]) Collect  # 001.000  <- 1048576
NAPA Tools Information:     (           tsnr[0]) Collect  # 001.000
NAPA Tools Information:     (           tsnr[0]) Process  # 001      <- 1310719
NAPA Tools Information:     (           tsnr[0]) End      # 001
NAPA Tools Information:     (            fft[0]) Process  # 001      <- 2097151
NAPA Tools Information:     (            fft[0]) End      # 001
NAPA Tools Information:     (            fft[0]) Collect  # 002.000  <- 2097152
NAPA Tools Information:     (           tsnr[0]) Collect  # 002.000
NAPA Tools Information:     (           tsnr[0]) Process  # 002      <- 2359295
NAPA Tools Information:     (           tsnr[0]) End      # 002
NAPA Tools Information:     (            fft[0]) Process  # 002      <- 3145727
NAPA Tools Information:     (            fft[0]) End      # 002
NAPA Debug Information:     (Close  I/O stream )           fft.out for fft             <-  3145728
NAPA Debug Information:     (Close  I/O stream )          tsnr.out for tsnr            <-  3145728
```

FFT #0 — **TSNR #0**

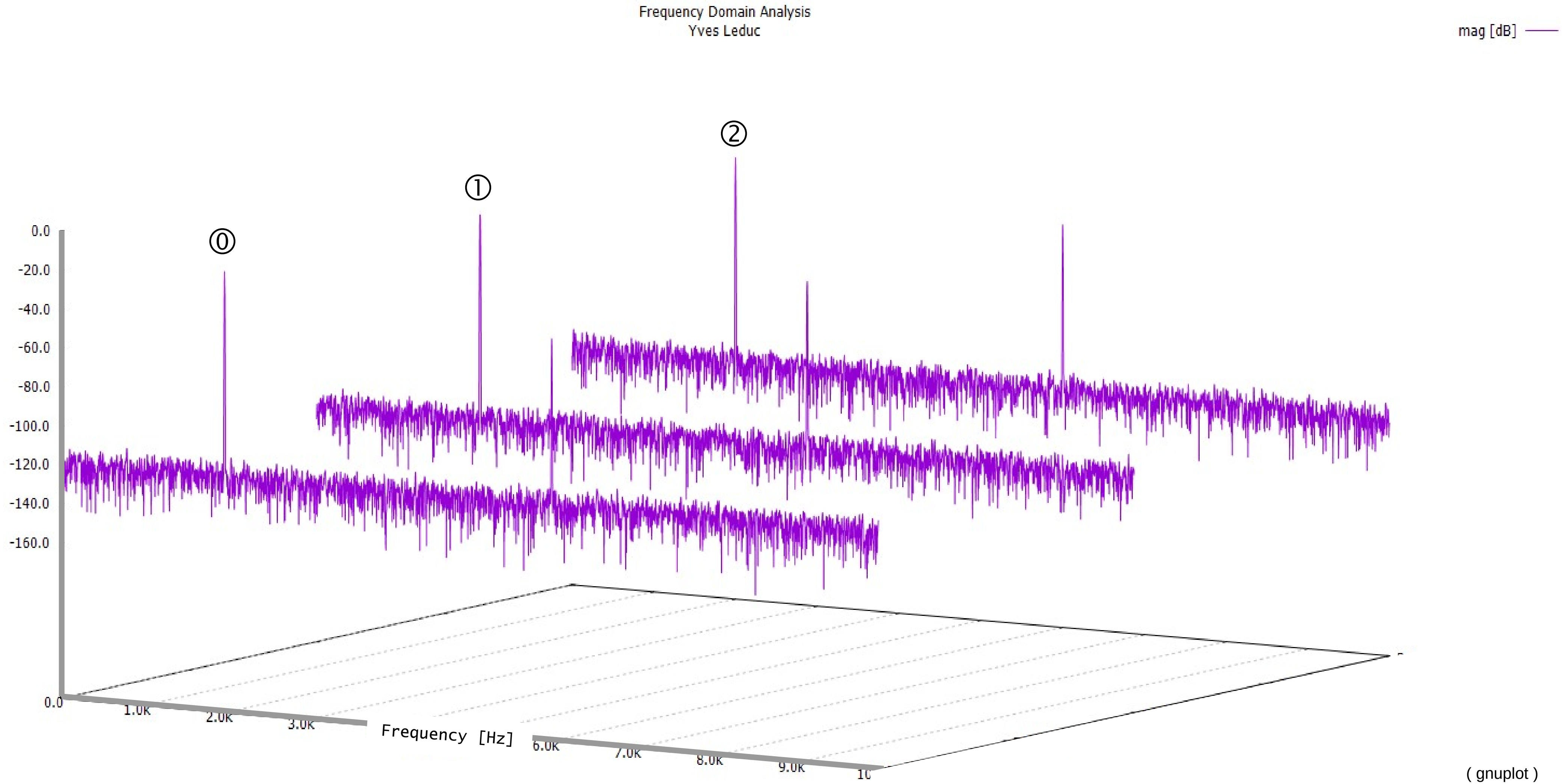
wait !

FFT #1 — **TSNR #1**

wait !

FFT #2 — **TSNR #2**

The Result



(gnuplot)

RMS of 100 FFT’s

file ‘./tool2.nap’

```
header <napatool.hdr>

title   "Frequency Domain Analysis"

fs      2.0e+6

dvar    ampl1   0.5
dvar    freq1   2000.0
dvar    ph1     rand_uniform(0.0, _2pi_)
dvar    ampl3   0.05
dvar    freq3   3.0 * freq1
dvar    ph3     rand_uniform(0.0, _2pi_)
dvar    bw      10000.0
ivar    npts    POWEROF2(18)

node    s0      noise 0.0  10.0e-6
node    s1      osc   0.0  ampl1  freq1   ph1
node    s2      osc   0.0  ampl3  freq3   ph3
node    in      sum   s0..2

tool   fft    "fft100.out"  in 1.0   bw   npts

terminate  1 <= TOOL_INDEX

directive  WINDOW   BLACKMAN_HARRIS_7
directive  NFFT     100

ping
```

These 100 FFT’s are synchronized as a single one

73

```
****
****  Frequency Domain Analysis
****

NAPA Tools Information:     (          fft[0]) Process  # 000.099  <- 262143
NAPA Tools Information:     (          fft[0]) Process  # 000.098  <- 524287
NAPA Tools Information:     (          fft[0]) Process  # 000.097  <- 786431
NAPA Tools Information:     (          fft[0]) Process  # 000.096  <- 1048575
NAPA Tools Information:     (          fft[0]) Process  # 000.095  <- 1310719
NAPA Tools Information:     (          fft[0]) Process  # 000.094  <- 1572863
NAPA Tools Information:     (          fft[0]) Process  # 000.093  <- 1835007
NAPA Tools Information:     (          fft[0]) Process  # 000.092  <- 2097151
NAPA Tools Information:     (          fft[0]) Process  # 000.091  <- 2359295
NAPA Tools Information:     (          fft[0]) Process  # 000.090  <- 2621439
NAPA Tools Information:     (          fft[0]) Process  # 000.089  <- 2883583
NAPA Tools Information:     (          fft[0]) Process  # 000.088  <- 3145727
NAPA Tools Information:     (          fft[0]) Process  # 000.087  <- 3407871
NAPA Tools Information:     (          fft[0]) Process  # 000.086  <- 3670015
NAPA Tools Information:     (          fft[0]) Process  # 000.085  <- 3932159
NAPA Tools Information:     (          fft[0]) Process  # 000.084  <- 4194303
```

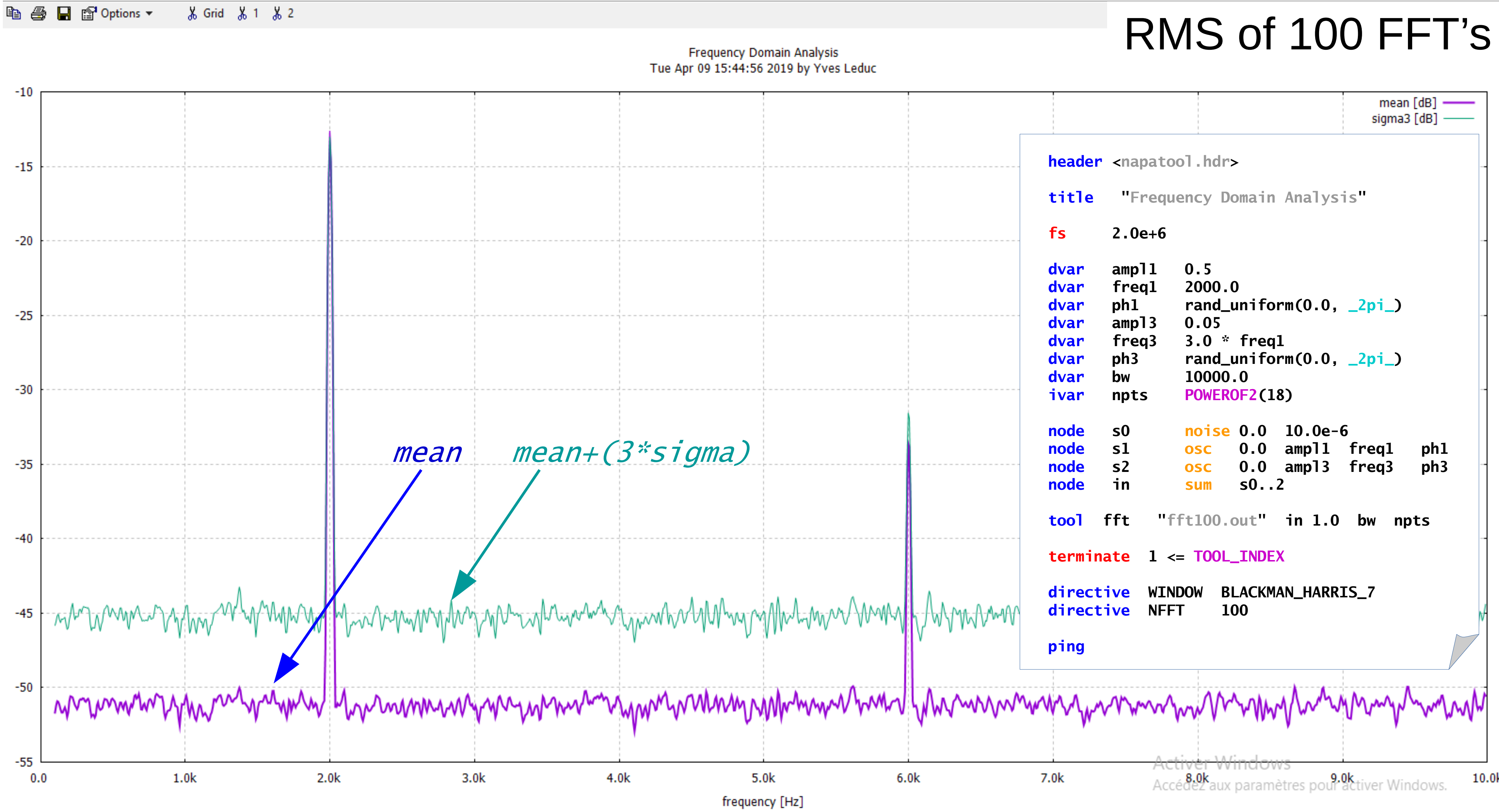
----- **100 RMS of FFT**

```
NAPA Tools Information:     (          fft[0]) Process  # 000.011  <- 23330815
NAPA Tools Information:     (          fft[0]) Process  # 000.010  <- 23592959
NAPA Tools Information:     (          fft[0]) Process  # 000.009  <- 23855103
NAPA Tools Information:     (          fft[0]) Process  # 000.008  <- 24117247
NAPA Tools Information:     (          fft[0]) Process  # 000.007  <- 24379391
NAPA Tools Information:     (          fft[0]) Process  # 000.006  <- 24641535
NAPA Tools Information:     (          fft[0]) Process  # 000.005  <- 24903679
NAPA Tools Information:     (          fft[0]) Process  # 000.004  <- 25165823
NAPA Tools Information:     (          fft[0]) Process  # 000.003  <- 25427967
NAPA Tools Information:     (          fft[0]) Process  # 000.002  <- 25690111
NAPA Tools Information:     (          fft[0]) Process  # 000.001  <- 25952255
NAPA Tools Information:     (          fft[0]) Process  # 000      <- 26214399

****  Random Seed [I]  :          777408748 ****
****  Output Tag  [O]  :           55961787 ****

****  NAPA Compiler    :     V4.00 for Win64 ****

****  Main Netlist     :           tool2.tmp ****

****  Simulator Time   :          13.1072  s ****
****  Simulator Index  :          26 214 400 ****
****  Tool Index       :                   1 ****

****  Run Time I/O     :                     ****

      ->  fft100.out                   [  0] ****

****  Stopwatch        :     H00:M00:S08.901 ****
```

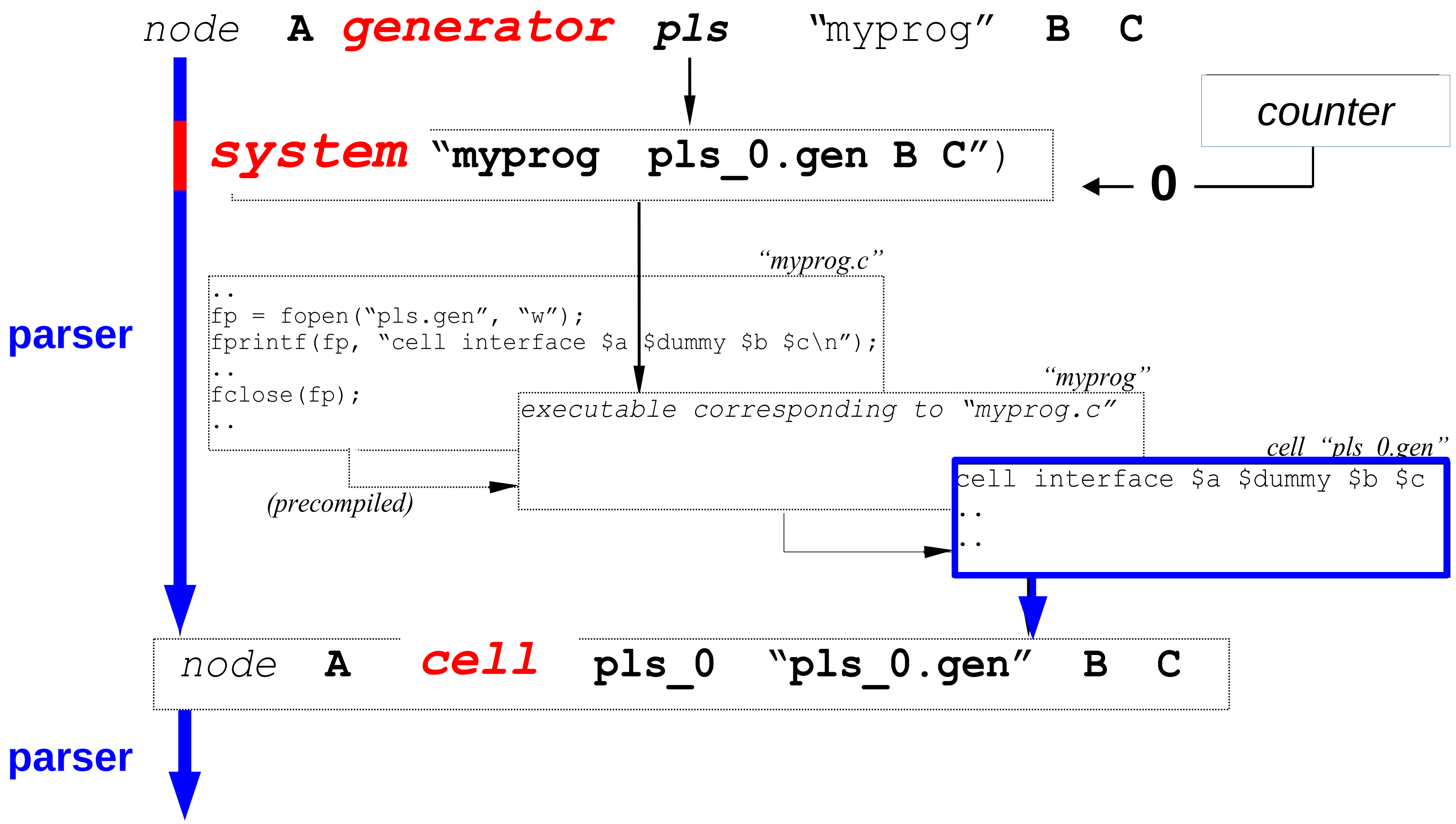
26.2 million cycles →

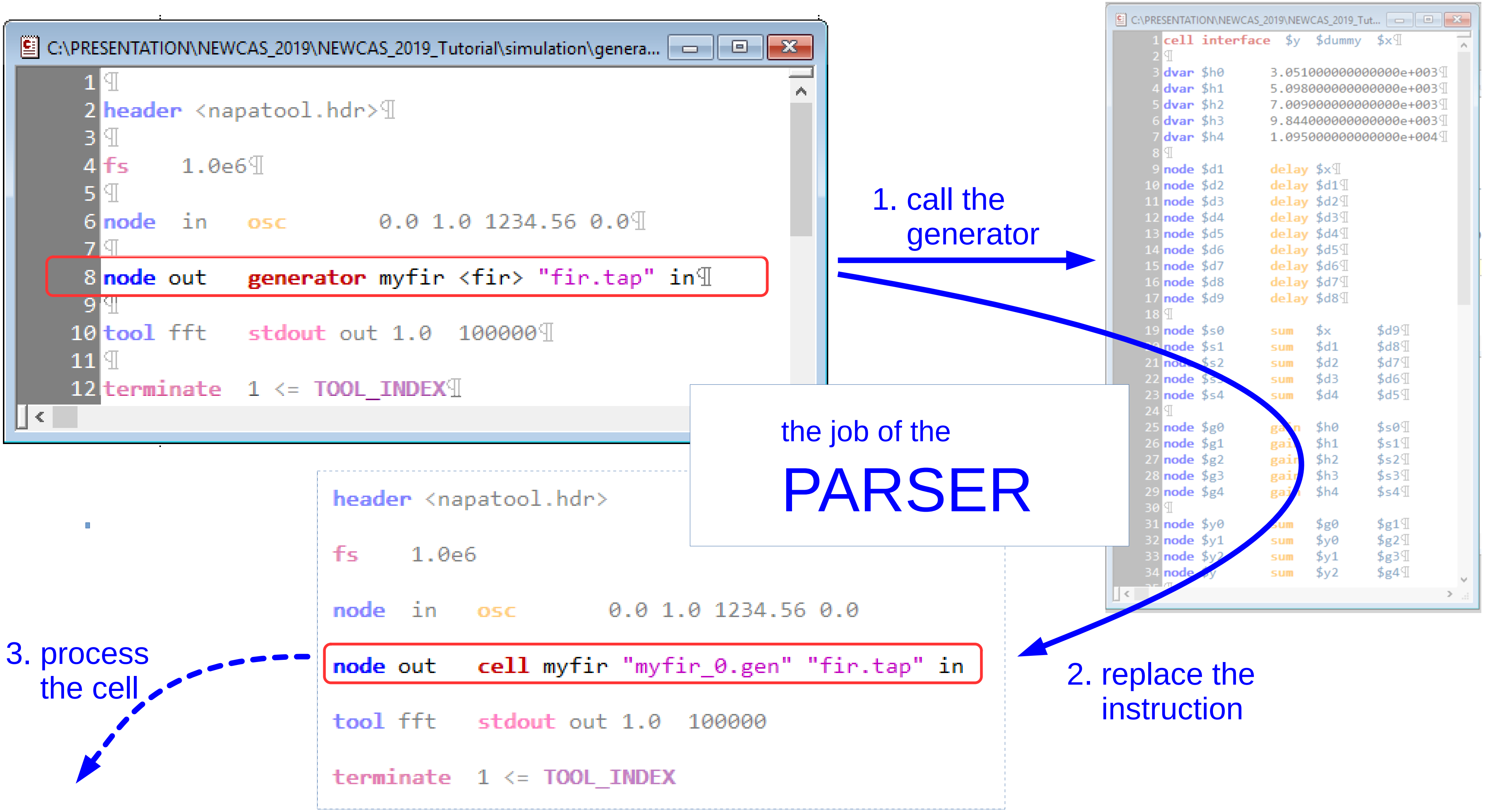
in 8.9 seconds →

RMS of 100 FFT's



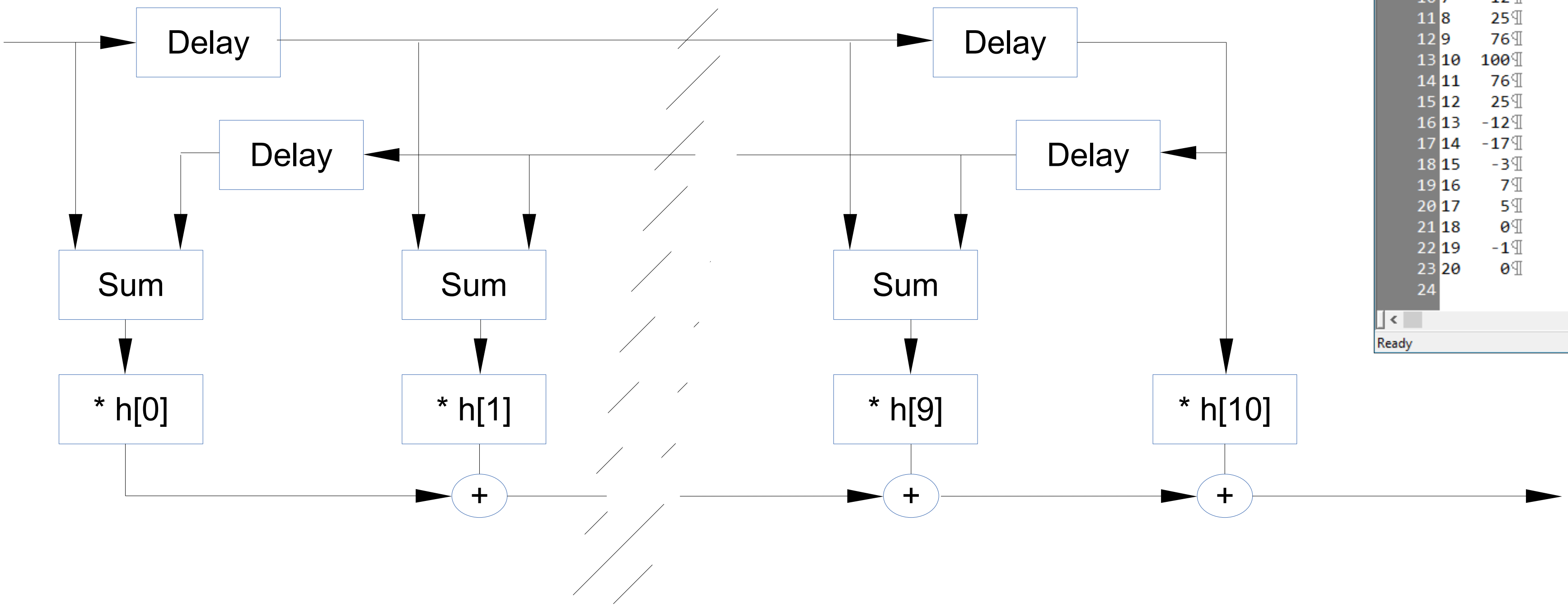
Cell Generator : Create a Cell on the Fly through a System Call

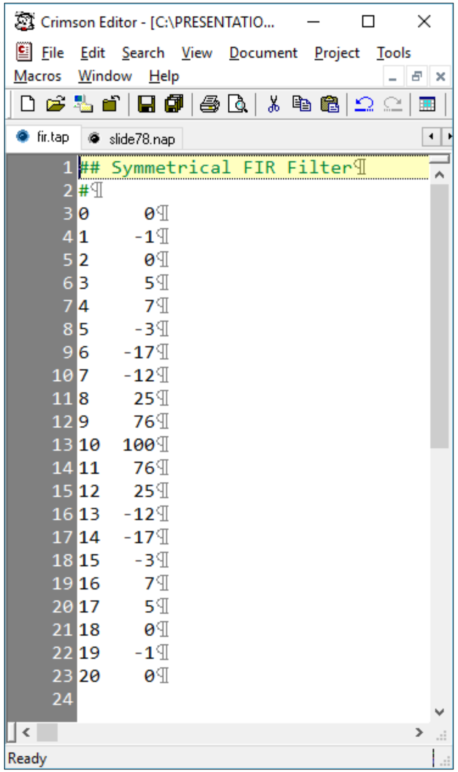



C:\PRESENTATION\NEWCAS_2019\NEWCAS_2019_Tutorial\simulation\genera...
header <napatool.hdr>
fs 1.0e6
node in osc 0.0 1.0 1234.56 0.0
node out generator myfir <fir> "fir.tap" in
tool fft stdout out 1.0 100000
terminate 1 <= TOOL_INDEX
1. call the generator
C:\PRESENTATION\NEWCAS_2019\NEWCAS_2019_Tut...
cell interface $y $dummy $x
dvar $h0 3.051000000000000e+003
dvar $h1 5.098000000000000e+003
dvar $h2 7.009000000000000e+003
dvar $h3 9.844000000000000e+003
dvar $h4 1.095000000000000e+004
node $d1 delay $x
node $d2 delay $d1
node $d3 delay $d2
node $d4 delay $d3
node $d5 delay $d4
node $d6 delay $d5
node $d7 delay $d6
node $d8 delay $d7
node $d9 delay $d8
node $s0 sum $x $d9
the job of the
PARSER
2. replace the instruction
header <napatool.hdr>
fs 1.0e6
node in osc 0.0 1.0 1234.56 0.0
node out cell myfir "myfir_0.gen" "fir.tap" in
tool fft stdout out 1.0 100000
terminate 1 <= TOOL_INDEX
3. process the cell

Cell Generator, an Example

A Digital Folded FIR

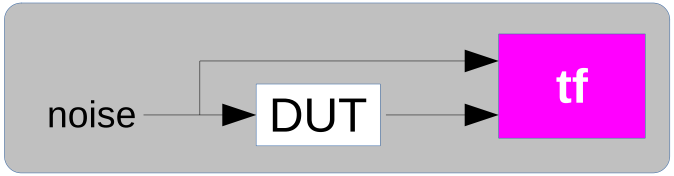




file './fir.nap'

```
title   "Transfer function of a symmetrical FIR"

header   <napatool.hdr>

fs          1.0e6

ivar   npts POWEROF2(16)

node   in    cell        rclk   <Noise/rclock.net>  0.50
node   out   generator   fltr   <fir>   "~/fir.tap"   in

tool   tf   "transfer_function.out"   in 1 out 1   npts

terminate   1 <= TOOL_INDEX

directive   REPEAT   10
ping
```

On the fly cell generation of a filter
using the file "fir.tap" containing the taps



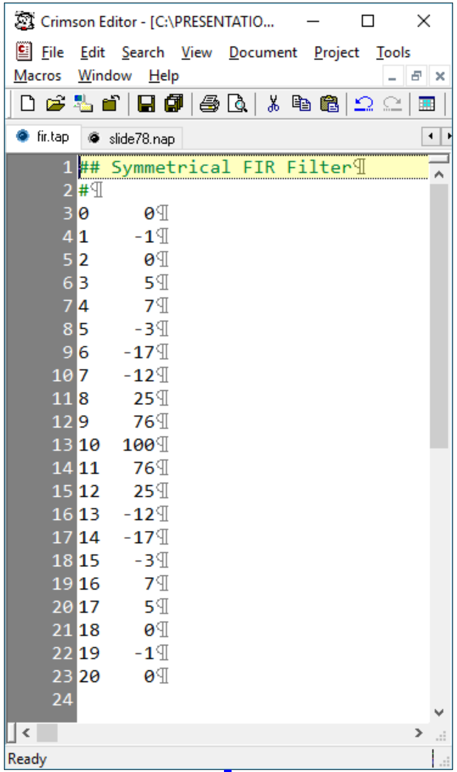
79

```
****
****  Transfer function of a symmetrical FIR
****


NAPA Tools Information:     (             tf[0]) Process
NAPA Tools Information:     (             tf[0]) Process
NAPA Tools Information:     (             tf[0]) Process
NAPA Tools Information:     (             tf[0]) Process  # 000.006  <- 262143
NAPA Tools Information:     (             tf[0]) Process  # 000.005  <- 327679
NAPA Tools Information:     (             tf[0]) Process  # 000.004  <- 393215
NAPA Tools Information:     (             tf[0]) Process  # 000.003  <- 458751
NAPA Tools Information:     (             tf[0]) Process  # 000.002  <- 524287
NAPA Tools Information:     (             tf[0]) Process  # 000.001  <- 589823
NAPA Tools Information:     (             tf[0]) Process  # 000      <- 655359


****  Random Seed [I]   :             778867594 ****
****  Output Tag  [O]   :             227728549 ****

****  NAPA Compiler     :         V4.00 for Win64 ****

****  Main Netlist      :                fir.tmp ****

****  Simulator Time    :            655.359 ms ****
****  Simulator Index   :                655360 ****
****  Tool Index        :                     1 ****

****  Run Time I/O      :                       ****

      ->  transfer_function.out             [  0] ****


****  Stopwatch         :         H00:M00:S01.422 ****


****  Normal Termination                          ****
```

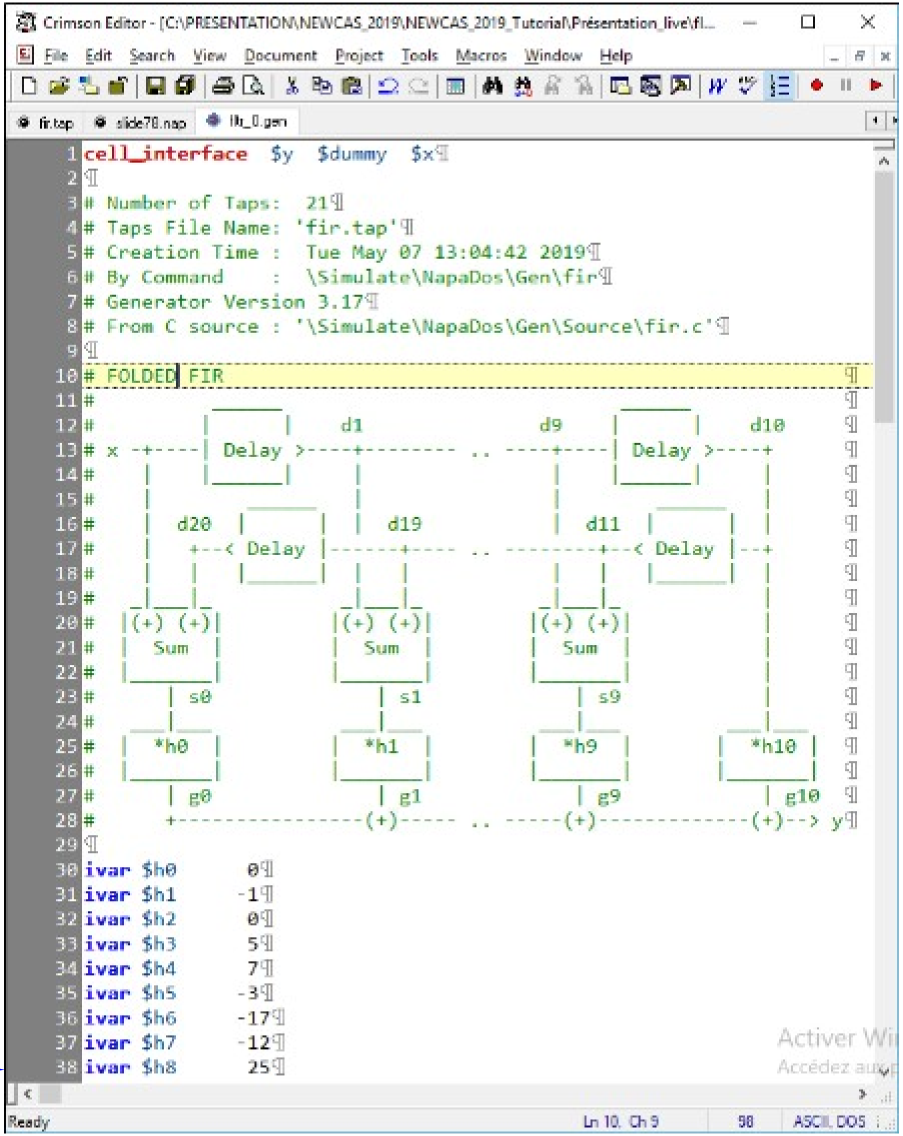
Digital FIR
Cell generator

Transfer function

RMS of FFTs



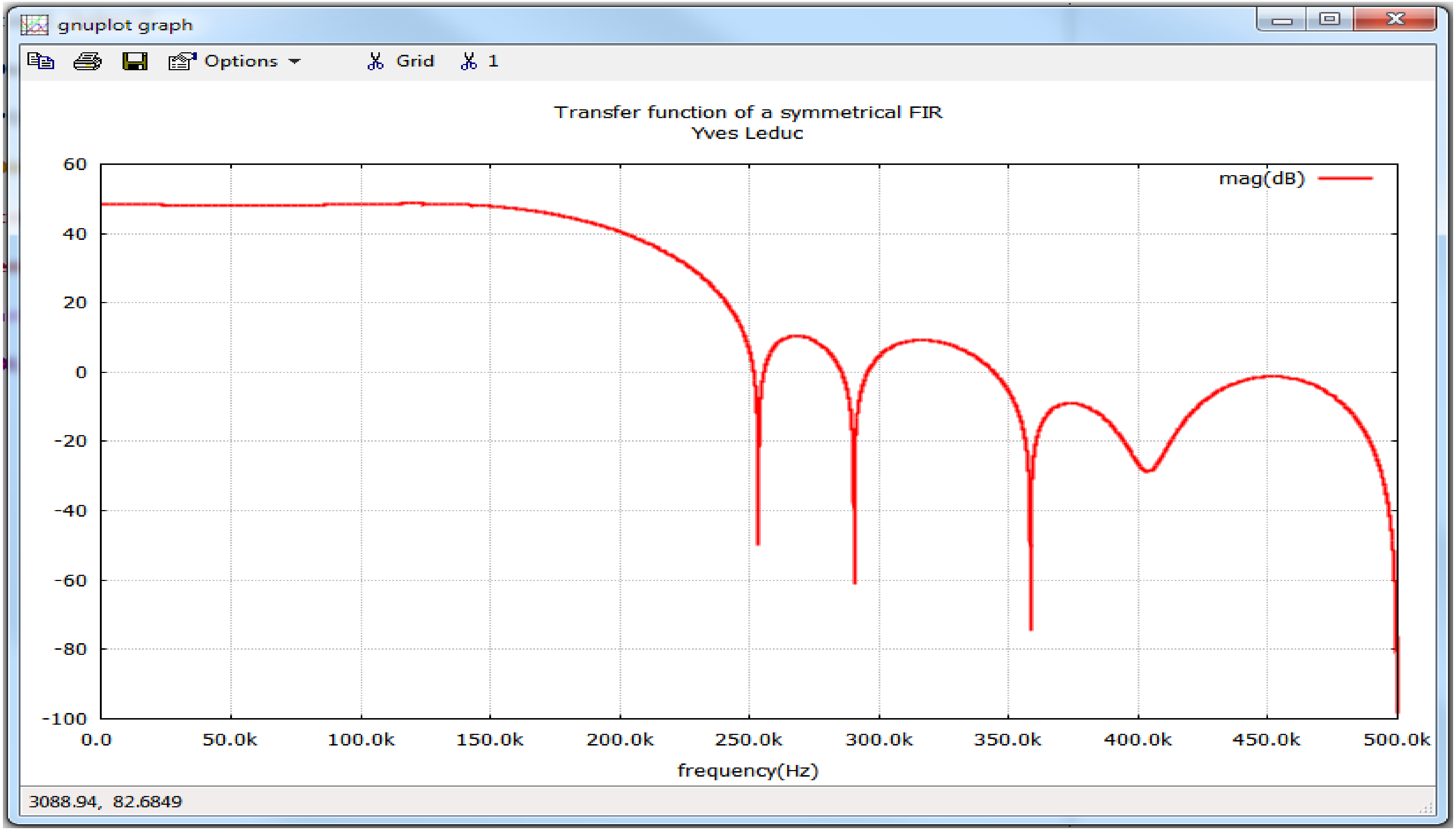
Taps (input)



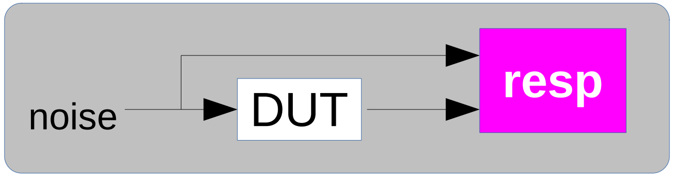
Generated Cell

```
node out generator fltr <fir> "~fir.tap" in
```

*generator **'fir'***


gnuplot graph
Options
Grid
1
Transfer function of a symmetrical FIR
Yves Leduc
mag(dB)
60
40
20
0
-20
-40
-60
-80
-100
0.0
50.0k
100.0k
150.0k
200.0k
250.0k
300.0k
350.0k
400.0k
450.0k
500.0k
frequency(Hz)
3088.94, 82.6849

Step and Impulse Responses



file './fir.nap'

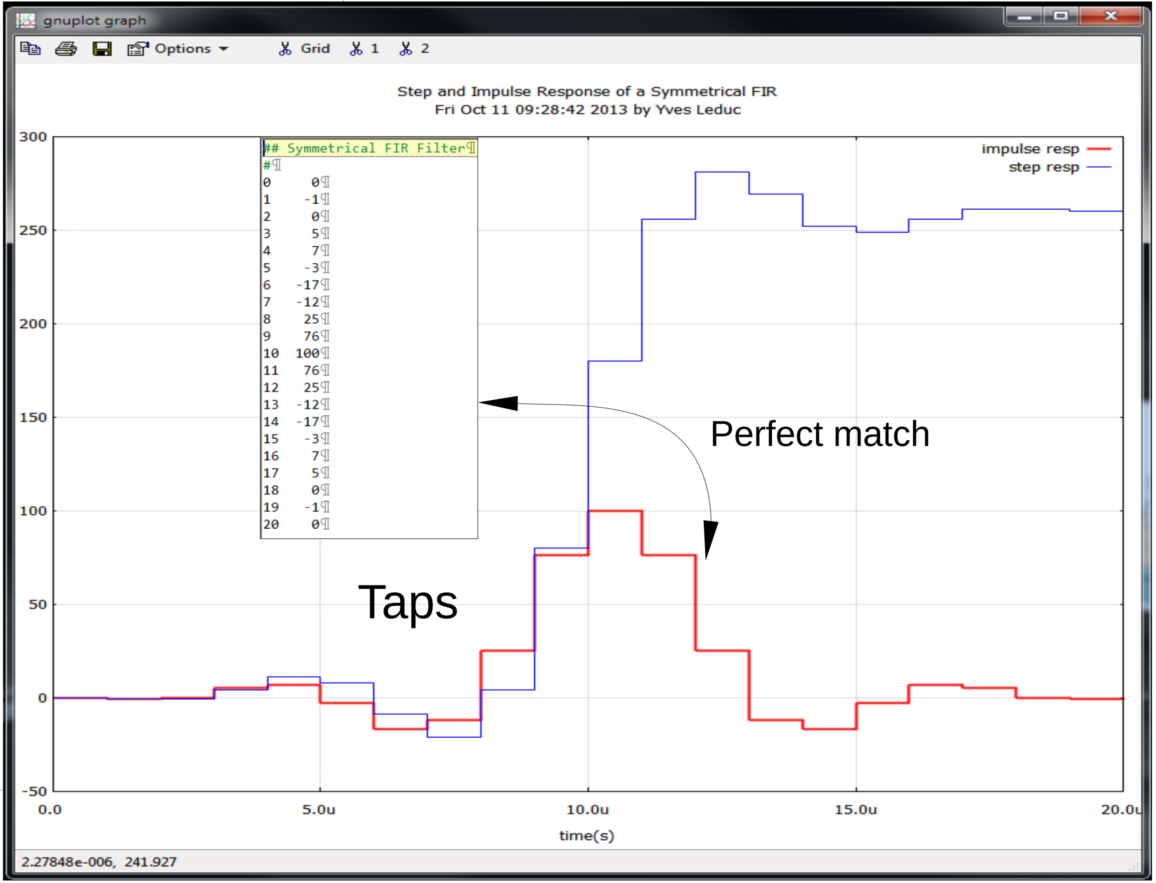
C:\PRESENTATION\NEWCAS_2019\NEWCAS_2019_Tutorial\Présentation_Live\SLIDE83_N\fir.nap

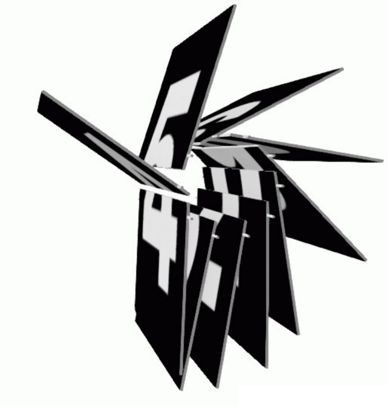
```
title  "Step and Impulse Response of a Symmetrical FIR"

header <napatool.hdr>

fs        1.0e6

directive REPEAT   10

ivar npts  POWEROF2(16)

node in    cell       rclk  <Noise/rclock.net>  0.50
node out   generator  fltr  <fir>    "~/fir.tap"    in

tool resp  "response.out"          in 1 out 1  npts

terminate  1 <= TOOL_INDEX

ping
```

82

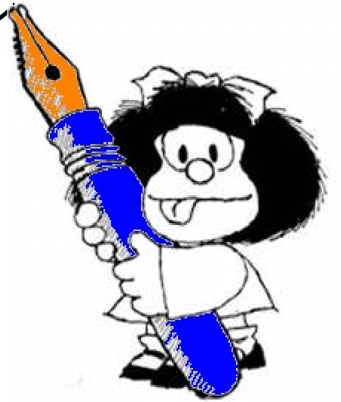
10 FFTs, 1 IFFT

```
NAPA Ping Information:     function 'itool_resp()'      from file "/Simulate/NapaDos/Hdr/Tool/fft4.hdr"
NAPA Ping Information:     function 'rand_bernoulli()'  from file "/Simulate/NapaDos/Hdr/Function/random.hdr"

 ****
 ****  Step and Impulse Response of a Symmetrical FIR
 ****

NAPA Tools Information:     (           resp[0]) Process  # 000.009  <- 65535
NAPA Tools Information:     (           resp[0]) Process  # 000.008  <- 131071
NAPA Tools Information:     (           resp[0]) Process  # 000.007  <- 196607
NAPA Tools Information:     (           resp[0]) Process  # 000.006  <- 262143
NAPA Tools Information:     (           resp[0]) Process  # 000.005  <- 327679
NAPA Tools Information:     (           resp[0]) Process  # 000.004  <- 393215
NAPA Tools Information:     (           resp[0]) Process  # 000.003  <- 458751
NAPA Tools Information:     (           resp[0]) Process  # 000.002  <- 524287
NAPA Tools Information:     (           resp[0]) Process  # 000.001  <- 589823
NAPA Tools Information:     (           resp[0]) Process  # 000      <- 655359

 ****  Random Seed [I]  :           778867323 ****
 ****  Output Tag  [O]  :           360459791 ****

 ****  NAPA Compiler    :       V4.00 for Win64 ****

 ****  Main Netlist     :             fir.tmp ****

 ****  Simulator Time   :          655.359 ms ****
 ****  Simulator Index  :              655360 ****
 ****  Tool Index       :                   1 ****

 ****  Run Time I/O     :                     ****

       -> response.out                  [   0] ****


 ****  Stopwatch        :       H00:M00:S01.391 ****


 ****  Normal Termination                     ****
```



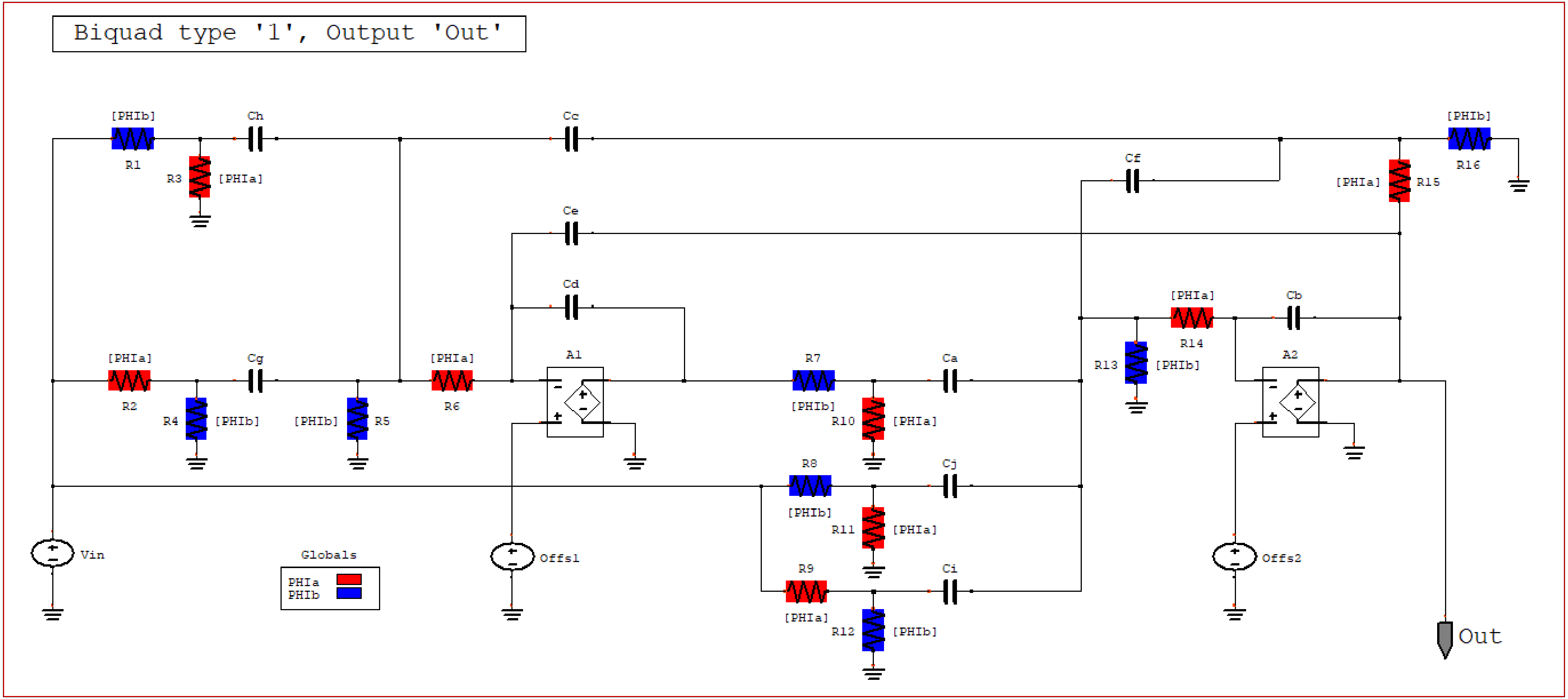


A Cascade of 3 SWC Biquadratic Filters



SWC Generic Biquadratic Filter 85 *(modeling)*

library file '/Simulate/NapaDos/Hdr/Max/Z_SWC_Biquad/Biquad_1a.sch'



SWC Generic Biquadratic Filter, Idealized Transfer Function

wxMaxima 19.01.2x [run_wxMaxima.mac*]

Fichier Edition View Cell Maxima Équations Algèbre Analyse Simplifier List Tracé de courbes Numérique Aide

$$nOffs1_1 = A1\ A2\ Ca\ (Ch + Cg + Ce + Cd + Cc)$$

$$nOffs1_0 = -A1\ A2\ Ca\ (Ce + Cd)$$

$$nOffs2_2 = A2\ (Ch + Cg + Ce + (A1+1)\ Cd + Cc)\ (Cj + Ci + Cf + Cb + Ca)$$

$$nOffs2_1 = -A2\ (((Ce + (A1+1)\ Cd)\ Cj + (Ce + (A1+1)\ Cd)\ Ci + Cb\ (Ch + Cg) + (Ce + (A1+1)\ Cd)\ Cf + (2\ Cb + Ca)\ Ce + (2\ (A1+1)\ Cb + (A1+1)\ Ca)\ Cd + Cb\ Cc)$$

$$nOffs2_0 = A2\ Cb\ (Ce + (A1+1)\ Cd)$$

$$nVin_2 = -A2\ (Ch + Cg + Ce + (A1+1)\ Cd + Cc)\ Ci$$

$$nVin_1 = A2\ ((Ch + Cg + Ce + (A1+1)\ Cd + Cc)\ Cj + (Ce + (A1+1)\ Cd)\ Ci - A1\ Ca\ Cg)$$

$$nVin_0 = -A2\ ((Ce + (A1+1)\ Cd)\ Cj - A1\ Ca\ Ch)$$

$$d_2 = (Ch + Cg + Ce + (A1+1)\ Cd + Cc)\ (Cj + Ci + (A2+1)\ Cf + (A2+1)\ Cb + Ca)$$

$$d_1 = -(((Ce + (A1+1)\ Cd)\ Cj + (Ce + (A1+1)\ Cd)\ Ci + Cb\ ((A2+1)\ Ch + (A2+1)\ Cg) + ((A2+1)\ Ce + ((A1+1)\ A2 + A1 + 1)\ Cd)\ Cf + (2\ (A2+1)\ Cb + (1 - A1\ A2)\ Ca)\ Ce + (A1+1)\ (2\ (A2+1)\ Cb + Ca)\ Cd + ((A2+1)\ Cb - A1\ A2\ Ca)\ Cc)$$

$$d_0 = ((A2+1)\ Cb - A1\ A2\ Ca)\ Ce + ((A1+1)\ A2 + A1 + 1)\ Cb\ Cd$$

... Z response (order 2)

$$Out_Z = \frac{(nOffs2_2\ Offs2 + nVin_2\ Vin)\ Z^2 + (nOffs1_1\ Offs1 + nOffs2_1\ Offs2 + nVin_1\ Vin)\ Z + nOffs1_0\ Offs1 + nOffs2_0\ Offs2 + nVin_0\ Vin}{d_2\ Z^2 + d_1\ Z + d_0}$$

... time response

$$Out_0 = \frac{nOffs2_2\ Offs2_0 + nVin_2\ Vin_0 + nOffs1_1\ Offs1_{-1} + nOffs2_1\ Offs2_{-1} + nVin_1\ Vin_{-1} + nOffs1_0\ Offs1_{-2} + nOffs2_0\ Offs2_{-2} + nVin_0\ Vin_{-2} - d_1\ Out_{-1} - d_0\ Out_{-2}}{d_2}$$

/******************** z2hfile( ) generating C:/Simulate/NapaDos/Hdr/Max/Z_SWC_Biquad/ZTRANS_Biquad_1a.h ********************/

done!

>>> Normal termination Maxima 38.27 s / wall 38.52 s [97.73,98.37]%

$$zsol\_ideal = -\frac{Cd\ Ci\ Vin\ Z^2 + (-Cd\ Cj - Cd\ Ci + Ca\ Cg)\ Vin\ Z + (Cd\ Cj - Ca\ Ch)\ Vin}{(Cd\ Cf + Cb\ Cd)\ Z^2 + (-Cd\ Cf + Ca\ Ce - 2\ Cb\ Cd + Ca\ Cc)\ Z - Ca\ Ce + Cb\ Cd}$$

$$Out = -\frac{(DI)\ \mathbf{Z^2} + (AG-DI-DJ)\ \mathbf{Z} + (DJ-AH)}{(DF+BD)\ \mathbf{Z^2} + (AE+AC-DF-2BD)\ \mathbf{Z} + (BD-AE)} * In$$

SWC Generic Biquadratic Filter, Full Transfer Function in Z Domain

(Opamps with limited gain and offset)

wxMaxima screen

/******************** CYCLE ********************/

$$nOffs1_2 = 0$$

$$nOffs1_1 = A1\,A2\,Ca\,(Ch+Cg+Ce+Cd+Cc)$$

$$nOffs1_0 = -A1\,A2\,Ca\,(Ce+Cd)$$

$$nOffs2_2 = A2\,(Ch+Cg+Ce+(A1+1)\,Cd+Cc)\,(Cj+Ci+Cf+Cb+Ca)$$

$$nOffs2_1 = -A2\,((Ce+(A1+1)\,Cd)\,Cj+(Ce+(A1+1)\,Cd)\,Ci+Cb\,(Ch+Cg)+(Ce+(A1+1)\,Cd)\,Cf+(2\,Cb+Ca)\,Ce+(2\,(A1+1)\,Cb+(A1+1)\,Ca)\,Cd+Cb\,Cc)$$

$$nOffs2_0 = A2\,Cb\,(Ce+(A1+1)\,Cd)$$

$$nVin_2 = -A2\,(Ch+Cg+Ce+(A1+1)\,Cd+Cc)\,Ci$$

$$nVin_1 = A2\,((Ch+Cg+Ce+(A1+1)\,Cd+Cc)\,Cj+(Ce+(A1+1)\,Cd)\,Ci-A1\,Ca\,Cg)$$

$$nVin_0 = -A2\,((Ce+(A1+1)\,Cd)\,Cj-A1\,Ca\,Ch)$$

$$d_2 = (Ch+Cg+Ce+(A1+1)\,Cd+Cc)\,(Cj+Ci+(A2+1)\,Cf+(A2+1)\,Cb+Ca)$$

$$d_1 = -((Ce+(A1+1)\,Cd)\,Cj+(Ce+(A1+1)\,Cd)\,Ci+Cb\,((A2+1)\,Ch+(A2+1)\,Cg)+((A2+1)\,Ce+((A1+1)\,A2+A1+1)\,Cd)\,Cf+(2\,(A2+1)\,Cb+(1-A1\,A2)\,Ca)\,Ce+(A1+1)\,(2\,(A2+1)\,Cb+Ca)\,Cd+((A2+1)\,Cb-A1\,A2\,Ca)\,Cc)$$

$$d_0 = ((A2+1)\,Cb-A1\,A2\,Ca)\,Ce+((A1+1)\,A2+A1+1)\,Cb\,Cd$$

... Z response (order 2)

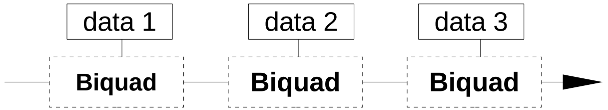
$$Out_Z = \frac{(nOffs2_2\,Offs2+nVin_2\,Vin)\,Z^2+(nOffs1_1\,Offs1+nOffs2_1\,Offs2+nVin_1\,Vin)\,Z+nOffs1_0\,Offs1+nOffs2_0\,Offs2+nVin_0\,Vin}{d_2\,Z^2+d_1\,Z+d_0}$$

SWC Generic Biquadratic Filter, a NAPA Cell in ‘**Z**’ Domain

file ‘./Biquad.net”

```
cell_interface   $Out  $In   $Offs1..2   $Ca $Cb $Cc $Cd $Ce $Cf $Cg $Ch $Ci $Cj  $A1..2

declare (analog) $In $Offs1..2
declare (analog) $Offs1..2
declare (analog) $Ca $Cb $Cc $Cd $Ce $Cf $Cg $Ch $Ci $Cj

dvar  $n3_0  -$A1*$A2*$Ca*($Ce+$Cd)
dvar  $n3_1  $A1*$A2*$Ca*($Ch+$Cg+$Ce+$Cd+$Cc)
dvar  $n3_2  0

dvar  $n2_0  $A2*$Cb*($Ce+($A1+1)*$Cd)
dvar  $n2_1  -$A2*(($Ce+($A1+1)*$Cd)*$Cj+($Ce+($A1+1)*$Cd)*$Ci+$Cb*($Ch+$Cg)+($Ce+($A1+1)*$Cd)*$Cf+(2*$Cb+$Ca)*$Ce     ...
             +(2*($A1+1)*$Cb+($A1+1)*$Ca)*$Cd+$Cb*$Cc)
dvar  $n2_2  $A2*($Ch+$Cg+$Ce+($A1+1)*$Cd+$Cc)*($Cj+$Ci+$Cf+$Cb+$Ca)

dvar  $n1_0  -$A2*(($Ce+($A1+1)*$Cd)*$Cj-$A1*$Ca*$Ch)
dvar  $n1_1  $A2*(($Ch+$Cg+$Ce+($A1+1)*$Cd+$Cc)*$Cj+($Ce+($A1+1)*$Cd)*$Ci-$A1*$Ca*$Cg)
dvar  $n1_2  -$A2*($Ch+$Cg+$Ce+($A1+1)*$Cd+$Cc)*$Ci

dvar  $d_0   (($A2+1)*$Cb-$A1*$A2*$Ca)*$Ce+(($A1+1)*$A2+$A1+1)*$Cb*$Cd
dvar  $d_1   (-($Ce+($A1+1)*$Cd)*$Cj)-($Ce+($A1+1)*$Cd)*$Ci-$Cb*(($A2+1)*$Ch+($A2+1)*$Cg)-(($A2+1)*$Ce+(($A1+1)*$A2+   ...
             $A1+1)*$Cd)*$Cf-(2*($A2+1)*$Cb+(1-$A1*$A2)*$Ca)*$Ce-($A1+1)*(2*($A2+1)*$Cb+$Ca)*$Cd-(($A2+1)*$Cb-$A1*$A2*$Ca)*$Cc
dvar  $d_2   ($Ch+$Cg+$Ce+($A1+1)*$Cd+$Cc)*($Cj+$Ci+($A2+1)*$Cf+($A2+1)*$Cb+$Ca)

node ($Out)  generator  bqd  <iir2>  3 2  $In  $Offs1..2  $n1_0..2 $n2_0..2 $n3_0..2  $d_0..2
```

A Cascade of 3 SWC Biquadratic Filters, the Choice of Capacitances



'./biquad1.dat'

```
data_interface  $Ca $Cb $Cc $Cd $Ce $Cf $Cg $Ch $Ci $Cj

#*  1

dvar  $Ca  0.633641e-12
dvar  $Cb  1.000000e-12
dvar  $Cc  0.207373e-12
dvar  $Cd  1.000000e-12
dvar  $Ce  0.0
dvar  $Cf  0.965438e-12
dvar  $Cg  0.626728e-12
dvar  $Ch  0.419355e-12
dvar  $Ci  0.331797e-12
dvar  $Cj  0.0
```

'./biquad2.dat'

```
data_interface  $Ca $Cb $Cc $Cd $Ce $Cf $Cg $Ch $Ci $Cj

#*  2

dvar  $Ca  0.317010e-12
dvar  $Cb  1.000000e-12
dvar  $Cc  0.257732e-12
dvar  $Cd  1.000000e-12
dvar  $Ce  0.0
dvar  $Cf  0.273196e-12
dvar  $Cg  0.257732e-12
dvar  $Ch  0.0
dvar  $Ci  0.548969e-12
dvar  $Cj  0.548969e-12
```

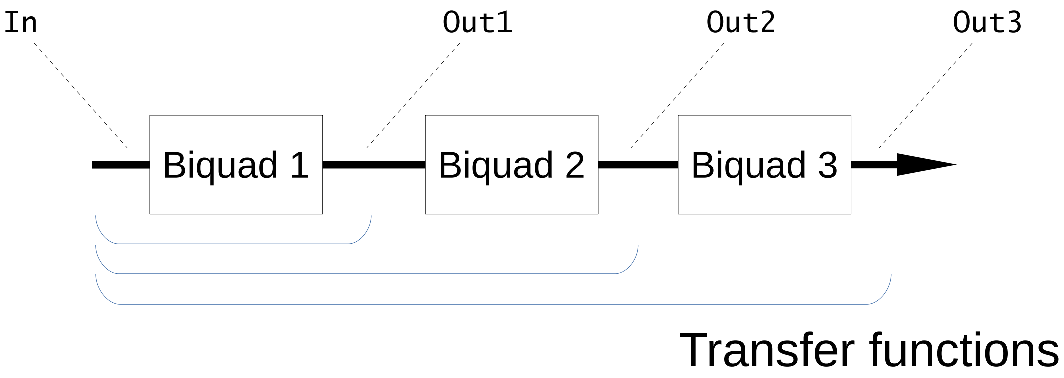
'./biquad3.dat'

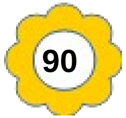
```
data_interface  $Ca $Cb $Cc $Cd $Ce $Cf $Cg $Ch $Ci $Cj

#*  3

dvar  $Ca  0.422692e-12
dvar  $Cb  1.000000e-12
dvar  $Cc  0.225806e-12
dvar  $Cd  1.000000e-12
dvar  $Ce  0.0
dvar  $Cf  0.050056e-12
dvar  $Cg  0.225806e-12
dvar  $Ch  0.0
dvar  $Ci  0.308120e-12
dvar  $Cj  0.308120e-12
```

A Cascade of 3 SWC Biquadratic Filters, the Simulation

file './biquad.nap'

```
header   <napatool.hdr>

fs              100.0e3

dvar     AA1   1000.0
dvar     AA2   1000.0
dvar     AB1   1000.0
dvar     AB2   1000.0
dvar     AC1   1000.0
dvar     AC2   1000.0

node  Offs1A  dc   0.0e-3
node  Offs2A  dc   0.0e-3
node  Offs1B  dc   0.0e-3
node  Offs2B  dc   0.0e-3
node  Offs1C  dc   0.0e-3
node  Offs2C  dc   0.0e-3

data   "./biquad1.dat"    CaA CbA CcA CdA CeA CfA CgA ChA CiA CjA
data   "./biquad2.dat"    CaB CbB CcB CdB CeB CfB CgB ChB CiB CjB
data   "./biquad3.dat"    CaC CbC CcC CdC CeC CfC CgC ChC CiC CjC

node     In    noise 0.0 1.0e-3

node   Out1   cell   bqdA  "./Biquad.net"    In Offs1A Offs2A   CaA CbA CcA CdA CeA CfA CgA ChA CiA CjA   AA1..2
node   Out2   cell   bqdB  "./Biquad.net"  Out1 Offs1B Offs2B   CaB CbB CcB CdB CeB CfB CgB ChB CiB CjB   AB1..2
node   Out3   cell   bqdC  "./Biquad.net"  Out2 Offs1C Offs2C   CaC CbC CcC CdC CeC CfC CgC ChC CiC CjC   AC1..2

ivar   npts   POWEROF2(18)

tool     tf   "temp1.out"  In 1.0  Out1 1.0  1.0e3 10.0e3  npts         // transfer function: In → Out1
post   join   stdout
tool     tf   "temp2.out"  In 1.0  Out2 1.0  1.0e3 10.0e3  npts         // transfer function: In → Out2
post   join   stdout
tool     tf   "temp3.out"  In 1.0  Out3 1.0  1.0e3 10.0e3  npts         // transfer function: In → Out3
post   join   stdout

directive  WINDOW  ROSENFELD
directive  NTF     10

terminate  1 <= TOOL_INDEX
```





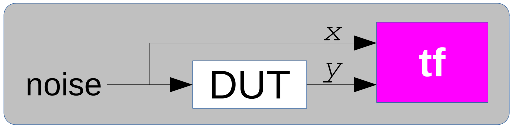
'screen'

```
[Biquad]  **** MAC Preprocessor Running ****
[Biquad]  **** NAPA Compiler    Running ****

NAPA Compiler Information:  (generator)
 Generating cell file  <./bqd_0.gen>, through system call:  '\Simulate\NapaDos\Gen\iir2 bqd_0.gen  3  2  In  Offs1A  Offs2A  bqdA__n1_0  bqdA__n1_1 etc..'


NAPA Compiler Information:  (generator)
 Generating cell file  <./bqdp_1.gen>, through system call:  '\Simulate\NapaDos\Gen\iir2 bqdp_1.gen  3  3  In  Offs1A  Offs2A  bqdA__np1_0  bqdA__np1_1 etc..'


NAPA Compiler Information:  (generator)
 Generating cell file  <./bqd_2.gen>, through system call:  '\Simulate\NapaDos\Gen\iir2 bqd_2.gen  3  2  OutA  Offs1B  Offs2B  bqdB__n1_0  bqdB__n1_1 etc..'


NAPA Compiler Information:  (generator)
 Generating cell file  <./bqdp_3.gen>, through system call:  '\Simulate\NapaDos\Gen\iir2 bqdp_3.gen  3  3  OutA  Offs1B  Offs2B  bqdB__np1_0  bqdB__np1_1 etc..'


NAPA Compiler Information:  (generator)
 Generating cell file  <./bqd_4.gen>, through system call:  '\Simulate\NapaDos\Gen\iir2 bqd_4.gen  3  2  OutB  Offs1C  Offs2C  bqdC__n1_0  bqdC__n1_1 etc..'


NAPA Compiler Information:  (generator)
 Generating cell file  <./bqdp_5.gen>, through system call:  '\Simulate\NapaDos\Gen\iir2 bqdp_5.gen  3  3  OutB  Offs1C  Offs2C  bqdC__np1_0  bqdC__np1_1 etc..'

[Biquad]  **** GCC Compiler     Running ****
[Biquad]  **** Ad Hoc Simulator Running ****

 ****
 ****  BIQUAD
 ****


NAPA Tools Information:     (        tf[0]) Process  # 000.009  <- 262143
NAPA Tools Information:     (   9    tf[1]) Process  # 000.009
NAPA Tools Information:     (        tf[2]) Process  # 000.009
NAPA Tools Information:     (        tf[0]) Process  # 000.008  <- 524287
NAPA Tools Information:     (   8    tf[1]) Process  # 000.008
NAPA Tools Information:     (        tf[2]) Process  # 000.008
NAPA Tools Information:     (        tf[0]) Process  # 000.007  <- 786431
NAPA Tools Information:     (   7    tf[1]) Process  # 000.007
NAPA Tools Information:     (        tf[2]) Process  # 000.007
NAPA Tools Information:     (        tf[0]) Process  # 000.006  <- 1048575
NAPA Tools Information:     (   6    tf[1]) Process  # 000.006
NAPA Tools Information:     (        tf[2]) Process  # 000.006
NAPA Tools Information:     (        tf[0]) Process  # 000.005  <- 1310719
NAPA Tools Information:     (   5    tf[1]) Process  # 000.005
NAPA Tools Information:     (        tf[2]) Process  # 000.005
NAPA Tools Information:     (   4    tf[0]) Process  # 000.004  <- 1572863
NAPA Tools Information:     (        tf[1]) Process  # 000.004
NAPA Tools Information:     (        tf[1]) Process  # 000.003
NAPA Tools Information:     (   3    tf[2]) Process  # 000.003
NAPA Tools Information:     (        tf[0]) Process  # 000.002  <- 2097151
NAPA Tools Information:     (        tf[1]) Process  # 000.002
NAPA Tools Information:     (   2    tf[2]) Process  # 000.002
NAPA Tools Information:     (        tf[0]) Process  # 000.001  <- 2359295
NAPA Tools Information:     (   1    tf[1]) Process  # 000.001
NAPA Tools Information:     (        tf[2]) Process  # 000.001
NAPA Tools Information:     (        tf[0]) Process  # 000      <- 2621439
NAPA Tools Information:     (   0    tf[1]) Process  # 000
NAPA Tools Information:     (        tf[2]) Process  # 000

NAPA Posts Information:     (         join[0]) Append 3 Files as requested

 ****  Random Seed [I]  :             777757025 ****
 ****  Output Tag  [O]  :             474615801 ****

 ****  NAPA Compiler    :         V4.00 for Win64 ****

 ****  Main Netlist     :            Biquad.tmp ****

 ****  Simulator Time   :            26.2144  s ****
 ****  Simulator Index  :             2 621 440 ****
 ****  Tool Index       :                     1 ****

 ****  Run Time I/O     :                       ****

       ->  stdout                        [  0] ****


 ****  Stopwatch        :         H00:M00:S05.469 ****
```

Generation on the fly of 6 cells

3 transfer functions computed 10 times

2.6 millions cycles in 26 seconds



The *Cross Power Spectrum*, G_{xy}, is defined as taking the Fourier Transform of two signals separately and multiplying the result together as follows:

$$G_{xy}(f) = S_x(f)\, S^*_y(f)$$

where * indicates the complex conjugate of the function.

With this function, we can define *the Transfer Function*, H(f), using the cross power spectrum and the spectrum of the input channel as follows:

$$H(f) = \frac{\overline{G_{yy}(f)}}{\overline{G_{xx}(f)}}$$

where $\overline{}$ denotes the average of the function.

The Fundamentals of Signal Analysis
Application Note 243
Agilent Technologies
Innovating the HP Way

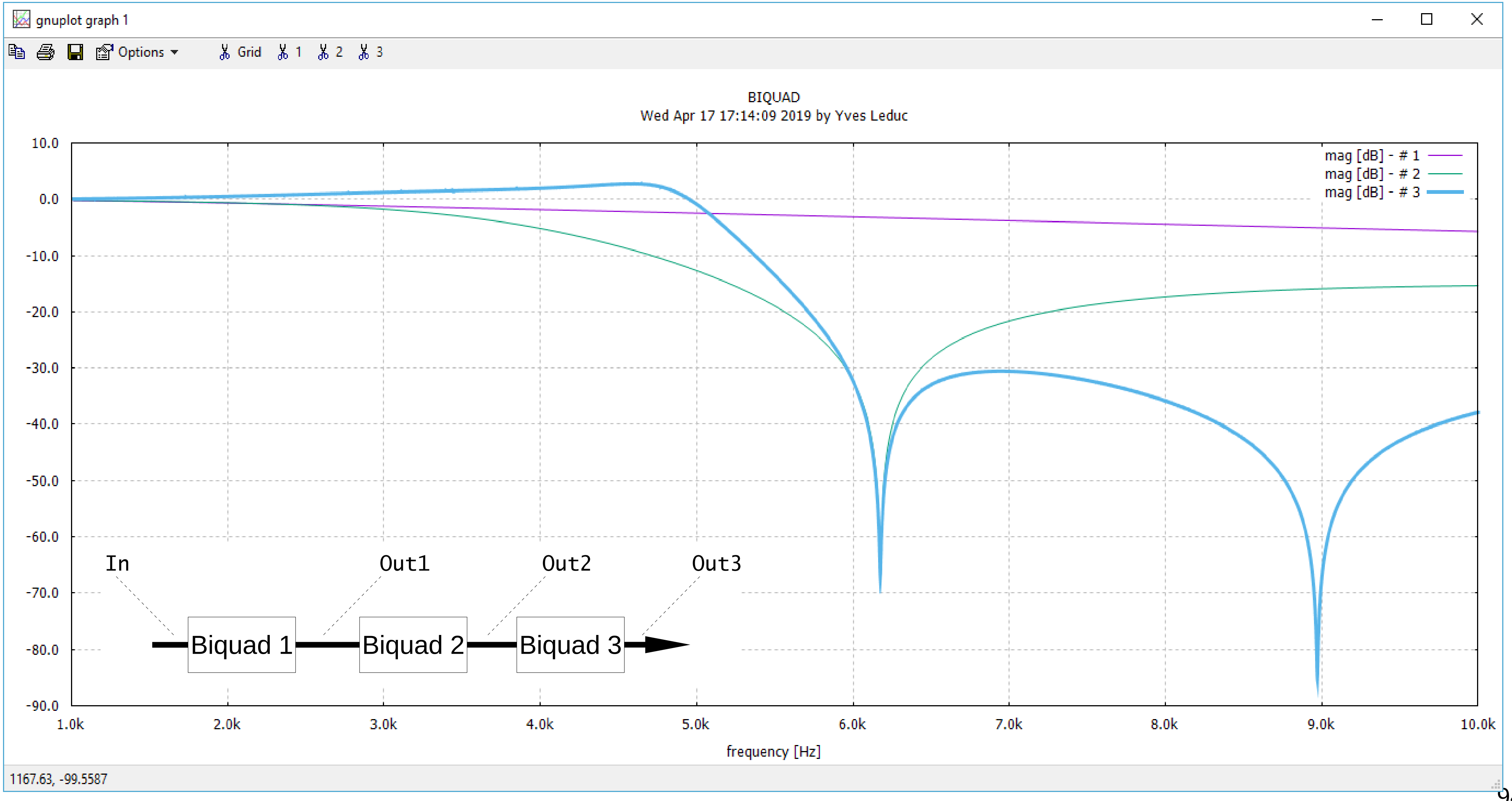
One of the Generated SWC Biquadratic Filter Cells

file './bqd1.gen'

```
cell_interface $out $N_3 $M_2 $in0..2 $n0_0..2 $n1_0..2 $n2_0..2 $d_0..2

#** Analog IIR Structure, 3 inputs 2nd order

#**
#**                  Feedback                  Feedforward
#**
#**     in0 -->(+)----------------+--------->[*n0_2/d_2]--->(+)--> out0
#**             ^                 |  s0_2                    ^
#**             |                 v                          |
#**             |              [DELAY]                       |
#**             |                 |                          |
#**             +---[*-d_1/d_2]<---+--------->[*n0_1/d_2]-----+
#**             |                 |  s0_1                    |
#**             |                 v                          |
#**             |              [DELAY]                       |
#**             |                 |                          |
#**             +---[*-d_0/d_2]<---+--------->[*n0_0/d_2]-----+
#**                                  s0_0
#**     ...
#**
#**     in2 -->(+)----------------+--------->[*n2_2/d_2]--->(+)--> out2
#**             ^                 |  s2_2                    ^
#**             |                 v                          |
#**             |              [DELAY]                       |
#**             |                 |                          |
#**             +---[*-d_1/d_2]<---+--------->[*n2_1/d_2]-----+
#**             |                 |  s2_1                    |
#**             |                 v                          |
#**             |              [DELAY]                       |
#**             |                 |                          |
#**             +---[*-d_0/d_2]<---+--------->[*n2_0/d_2]-----+
#**                                  s2_0
#**
#**     out = out0 + out1 + out2

# **********************************************************************
```

```
node  $ii0 dalgebra    $in0
node  $ii1 dalgebra    $in1
node  $ii2 dalgebra    $in2

# feedforward path coefficients
dvar  $nn0_0  $n0_0/$d_2   &update
dvar  $nn0_1  $n0_1/$d_2   &update
dvar  $nn0_2  $n0_2/$d_2   &update

dvar  $nn1_0  $n1_0/$d_2   &update
dvar  $nn1_1  $n1_1/$d_2   &update
dvar  $nn1_2  $n1_2/$d_2   &update

dvar  $nn2_0  $n2_0/$d_2   &update
dvar  $nn2_1  $n2_1/$d_2   &update
dvar  $nn2_2  $n2_2/$d_2   &update

# feedback path coefficients
dvar  $dd_0   $d_0/$d_2    &update
dvar  $dd_1   $d_1/$d_2    &update

# signal path
node  $s0_0   delay   $s0_1
node  $s0_1   delay   $s0_2
node  $s0_2    wsum     1.0    $ii0    -$dd_0  $s0_0   -$dd_1  $s0_1
node  $out0    wsum   $nn0_0  $s0_0   $nn0_1  $s0_1   $nn0_2  $s0_2

node  $s1_0   delay   $s1_1
node  $s1_1   delay   $s1_2
node  $s1_2    wsum     1.0    $ii1    -$dd_0  $s1_0   -$dd_1  $s1_1
node  $out1    wsum   $nn1_0  $s1_0   $nn1_1  $s1_1   $nn1_2  $s1_2

node  $s2_0   delay   $s2_1
node  $s2_1   delay   $s2_2
node  $s2_2    wsum     1.0    $ii2    -$dd_0  $s2_0   -$dd_1  $s2_1
node  $out2    wsum   $nn2_0  $s2_0   $nn2_1  $s2_1   $nn2_2  $s2_2

node ($out)     sum   $out0..2
```

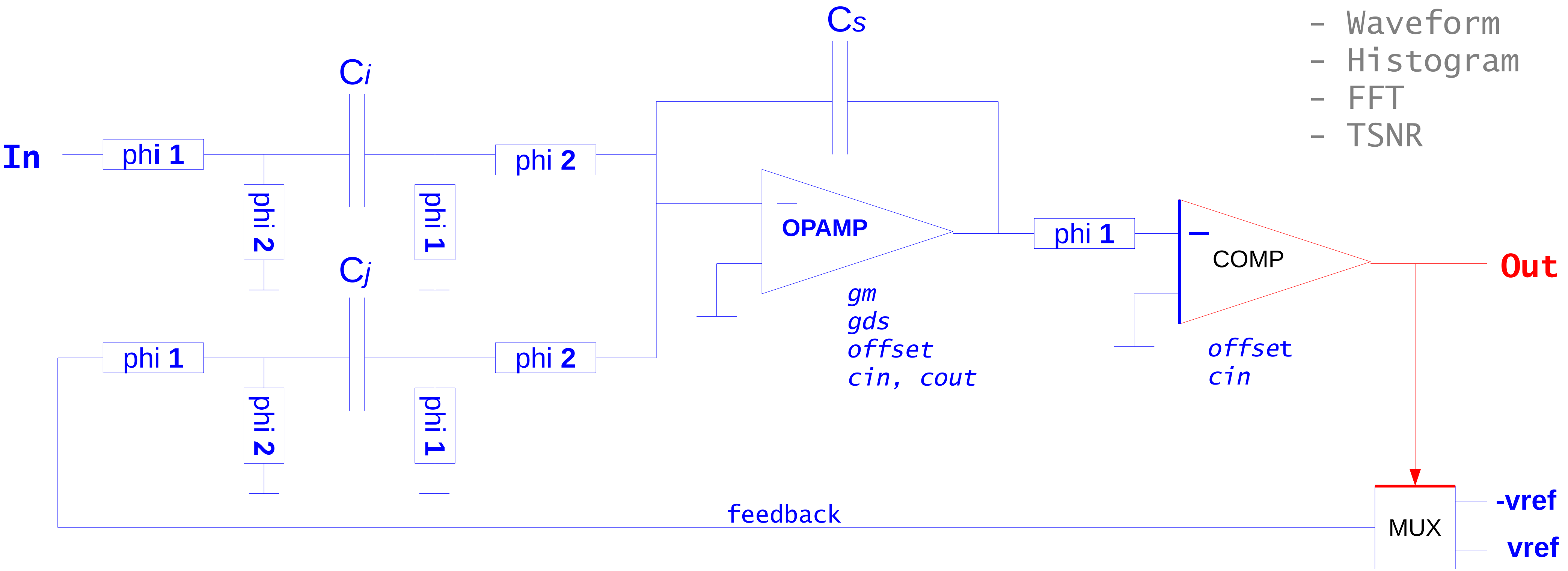
A Cascade of 3 SWC Biquadratic Filters, the Transfer Functions



1^{st} *Order* ΣΔ *Analog Modulator*

Simulating with **SARC** : Precise Simulations at Lower Level

- Waveform
- Histogram
- FFT
- TSNR



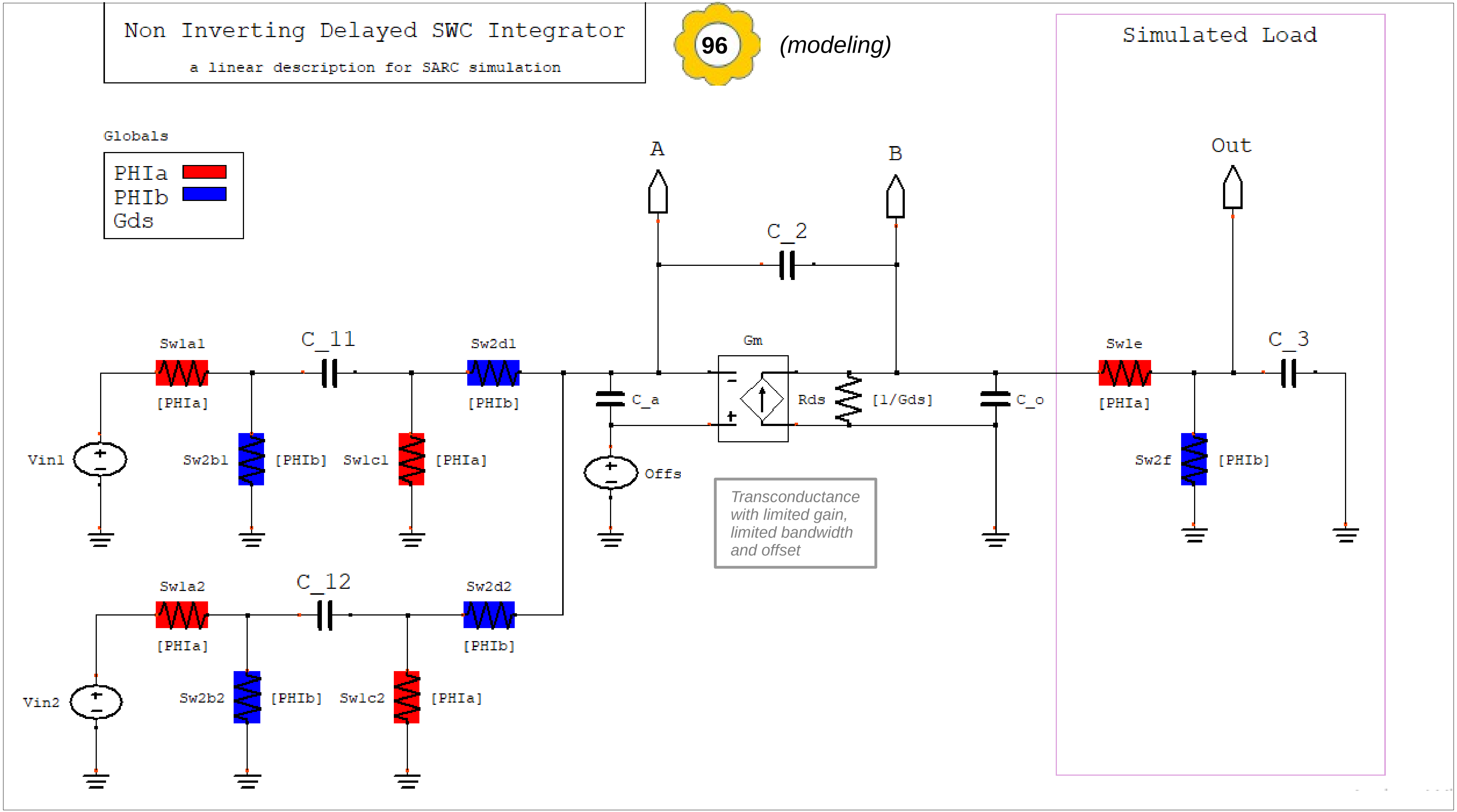
Electrical modeling of the **switched capacitor integrator** :

offset, limited gain, limited bandwidth, parasitic capacitances, switches

Mixed signal modeling of the modulator :

integrator, comparator, multiplexer

~~Z Domain~~ ~~s Domain~~ s Domain *'with care'*


Non Inverting Delayed SWC Integrator
a linear description for SARC simulation
96
(modeling)
Simulated Load
Globals
PHIa
PHIb
Gds
A
B
Out
C_2
Sw1a1
[PHIa]
C_11
Sw2d1
[PHIb]
Gm
Rds
[1/Gds]
C_a
C_o
Sw1e
[PHIa]
C_3
Vin1
Sw2b1
[PHIb]
Sw1c1
[PHIa]
Offs
Sw2f
[PHIb]
Transconductance with limited gain, limited bandwidth and offset
Sw1a2
[PHIa]
C_12
Sw2d2
[PHIb]
Vin2
Sw2b2
[PHIb]
Sw1c2
[PHIa]

SWC ΣΔ Converter with SARC

BIND[..]

$$BIND = \begin{pmatrix} Gds \\ PHIa \\ PHIb \\ Rds = [1/Gds] \\ Sw1a1 = [PHIa] \\ Sw1a2 = [PHIa] \\ Sw1c1 = [PHIa] \\ Sw1c2 = [PHIa] \\ Sw1e = [PHIa] \\ Sw2b1 = [PHIb] \\ Sw2b2 = [PHIb] \\ Sw2d1 = [PHIb] \\ Sw2d2 = [PHIb] \\ Sw2f = [PHIb] \end{pmatrix}$$

STATES[..]

$$STATES = \begin{pmatrix} V(C\_11) & V(C\_12) & V(C\_2) & V(C\_3) & V(C\_a) \end{pmatrix}$$

ESE[..]

$$ESE = \begin{pmatrix} [C_{11}] \\ [C_{12}] \\ [C_2] \\ [C_3] \\ [C_a] \\ [C_o] \end{pmatrix}$$

A[..]

$$A = \begin{pmatrix} -\frac{PHIb + PHIa}{2\, C_{11}\, PHIa\, PHIb} & 0 & 0 & 0 & -\frac{1}{2\, C_{11}\, PHIb} \\ 0 & -\frac{PHIb + PHIa}{2\, C_{12}\, PHIa\, PHIb} & 0 & 0 & -\frac{1}{2\, C_{12}\, PHIb} \\ -\frac{C_o}{2\left(C_a C_o + C_2\left(C_o + C_a\right)\right) PHIb} & -\frac{C_o}{2\left(C_a C_o + C_2\left(C_o + C_a\right)\right) PHIb} & -\frac{C_a\,(Gds\, PHIa + 1)}{\left(C_a C_o + C_2\left(C_o + C_a\right)\right) PHIa} & -\frac{C_a}{\left(C_a C_o + C_2\left(C_o + C_a\right)\right) PHIa} & \frac{C_a\,((Gm + Gds)\, PHIa + 1)\, PHIb^2 + PHIa\left(\left(C_a\, Gds\, PHIa - 2\, C_o + C_a\right) PHIb + PHIa\left(C_a\, Gm\, PHIb - C_o\right)\right)}{\left(C_a C_o + C_2\left(C_o + C_a\right)\right) PHIa\, PHIb\,(PHIb + PHIa)} \\ 0 & 0 & -\frac{1}{C_3\, PHIa} & -\frac{PHIb + PHIa}{C_3\, PHIa\, PHIb} & \frac{1}{C_3\, PHIa} \\ -\frac{C_o + C_2}{2\left(C_a C_o + C_2\left(C_o + C_a\right)\right) PHIb} & -\frac{C_o + C_2}{2\left(C_a C_o + C_2\left(C_o + C_a\right)\right) PHIb} & \frac{C_2\,(Gds\, PHIa + 1)}{\left(C_a C_o + C_2\left(C_o + C_a\right)\right) PHIa} & \frac{C_2}{\left(C_a C_o + C_2\left(C_o + C_a\right)\right) PHIa} & -\frac{C_2\,((Gm + Gds)\, PHIa + 1)\, PHIb^2 + PHIa\left(\left(C_2\, Gds\, PHIa + 2\, C_o + 3\, C_2\right) PHIb + PHIa\left(C_2\, Gm\, PHIb + C_o + C_2\right)\right)}{\left(C_a C_o + C_2\left(C_o + C_a\right)\right) PHIa\, PHIb\,(PHIb + PHIa)} \end{pmatrix}$$

B[..]

$$B = \begin{pmatrix} \frac{1}{2\, C_{11}\, PHIa} & 0 & 0 \\ 0 & \frac{1}{2\, C_{12}\, PHIa} & 0 \\ \frac{C_o}{2\left(C_a C_o + C_2\left(C_o + C_a\right)\right)(PHIb + PHIa)} & \frac{C_o}{2\left(C_a C_o + C_2\left(C_o + C_a\right)\right)(PHIb + PHIa)} & -\frac{C_a\, Gm\, Offs}{C_a C_o + C_2\left(C_o + C_a\right)} \\ 0 & 0 & 0 \\ \frac{C_o + C_2}{2\left(C_a C_o + C_2\left(C_o + C_a\right)\right)(PHIb + PHIa)} & \frac{C_o + C_2}{2\left(C_a C_o + C_2\left(C_o + C_a\right)\right)(PHIb + PHIa)} & \frac{C_2\, Gm\, Offs}{C_a C_o + C_2\left(C_o + C_a\right)} \end{pmatrix}$$

C[..]

$$C = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 & 1 \end{pmatrix}$$

D[..]

$$D = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

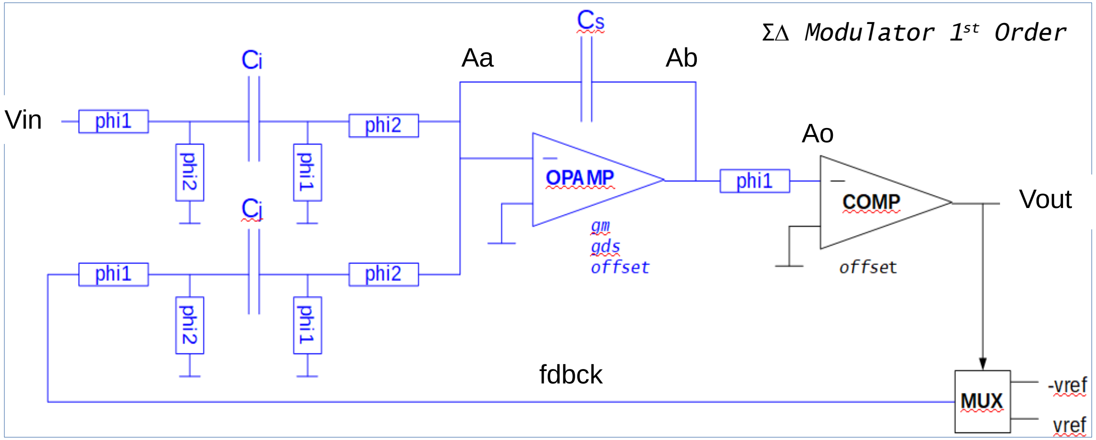
```
******  Output in file  <  C:/PRESENTATION/NEWCAS_2019/NEWCAS_2019_Tutorial/Interactive/Integrator2_NI.hdr  >  *****

>>> Normal termination  Maxima 8.49 s / wall 8.58 s  [96.69,98.38]%
```

file ‘./Integrator2_NI.hdr’

```
...

#define  MIMO_Integrator2_NI_INPUTS      \
   {"Vin1" , "Vin2" }

#define  MIMO_Integrator2_NI_OUTPUTS     \
   {"V(Out)" , "V(A)" , "V(B)" }

#define  MIMO_Integrator2_NI_BOM         \
   {"C11", "C12", "C2", "C3", "Ca", "Co", "Gm", "Offs", "Gds", "PHIa", "PHIb"}

...
```

file './SD1.net'

```
cell_interface   $Vout  $Vin $Vref  $Clk   $Afile $Cfile

title " [ Analog 1st Order SD Modulator with a SARC model ] "

node  $Phi1  dalgebra  $Clk?  $ROff : $ROn
node  $Phi2  dalgebra  $Clk?  $ROn  : $ROff

dvar  $ROn   1.0e3
dvar  $ROff  1.0e9

dvar  $a     1.0              // scaling input
dvar  $g     1.0              // scaling reference

dvar  $Ci    $a*$Cs
dvar  $Cj    $g*$Cs
dvar  $Cs    5.0e-12

dvar  $ibias 125.0e-6         // opamp bias current
data  $Afile $Agm $Agds $OffsA  $Aca $Aco  $ibias

ganging  $sdparm[]  $Ci $Cj $Cs  $Cc  $Aca $Aco $Agm $Agds  $Phi1 $Phi2

node  $tag   duser  sarc   Integrator2_NI()  $sdparm  $OffsA  $fdbck  $Vin
node ($Aa)   duser  sarc   $tag    (V@A)
node ($Ab)   duser  sarc   $tag    (V@B)
node ($Ao)   duser  sarc   $tag    (V@Out)

data  $Cfile        $OffsC $Cc
node  $Vout  comp   $Ao    $OffsC &delayed

node  $fdbck mux    $Vout  $Vref -$Vref

header "./MIMO_Integrator2_NI.hdr"
```



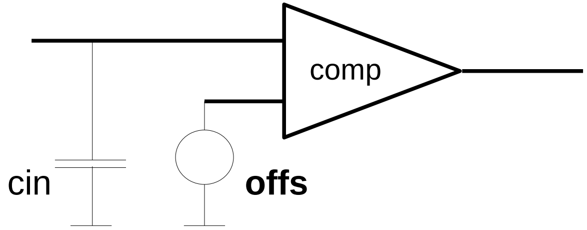
Multiple Inputs Multiple Outputs
duser **sarc** function

SD1 Modulator Cell Netlist

Computing in a Data Cell

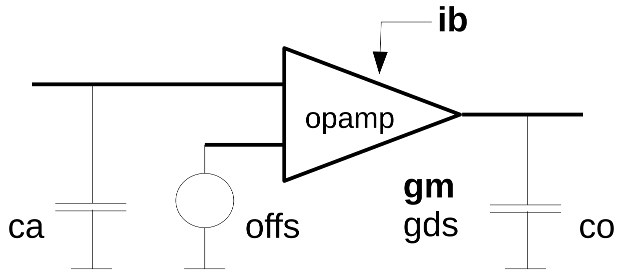
file ‘./comparator.dat’

```
data_interface  $offs  $cin

#* comparator data

dvar ($cin)  0.5e-12  &constant

dvar $offs   rand_normal(0.0, 2.0e-3)
```



file ‘./transconductance_opamp.dat’

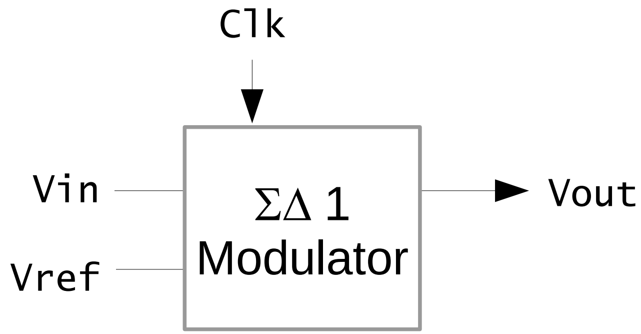
```
data_interface  $gm $gds  $offs  $ca $co  $ib

#* opamp data

dvar  $gds   26.0e-9 - 2.4e-15*$ib - 4.6e-24*$ib*$ib     &update
dvar  $gm    300.0e-6 - 230.0e-9*SQRT($ib) + 1.2e-12*$ib &update

dvar  ($ca)  0.2e-12                                     &constant
dvar  ($co)  0.4e-12                                     &constant

dvar $offs   rand_normal(0.0, 3.0e-3)

declare  (true)  (100.0e-6 <= $ib) && (150.0e-6 >= $ib)
```

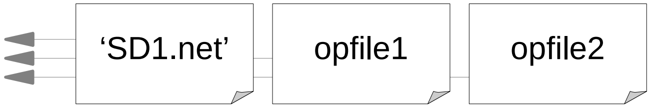


A Time-Domain Simulation

file ‘./SD1.nap’

```
header  <napatool.hdr>

title   "Analysis in Time Domain"

fs      2000.0e6
node    Clk       clock   "01" 500     // 2.0 Mhz clock

string  opfile1  "./transconductance_opamp.dat"
string  opfile2  "./comparator.dat"

node    Vin       dc     (analog)   0.123456789
node    Vref      dc     (analog)   1.0
node    Vout      cell   sd1  "./SD1.net"  Vin Vref  Clk  opfile1..2

output "SD1-time.out" Aa Ab Ao  Vin Vout  Clk

terminate   10.0e-6 <= TIME

alias   Aa   sd1__Aa
alias   Ab   sd1__Ab
alias   Ao   sd1__Ao

debug   SARC
ping
```

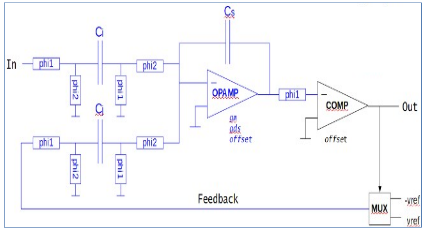


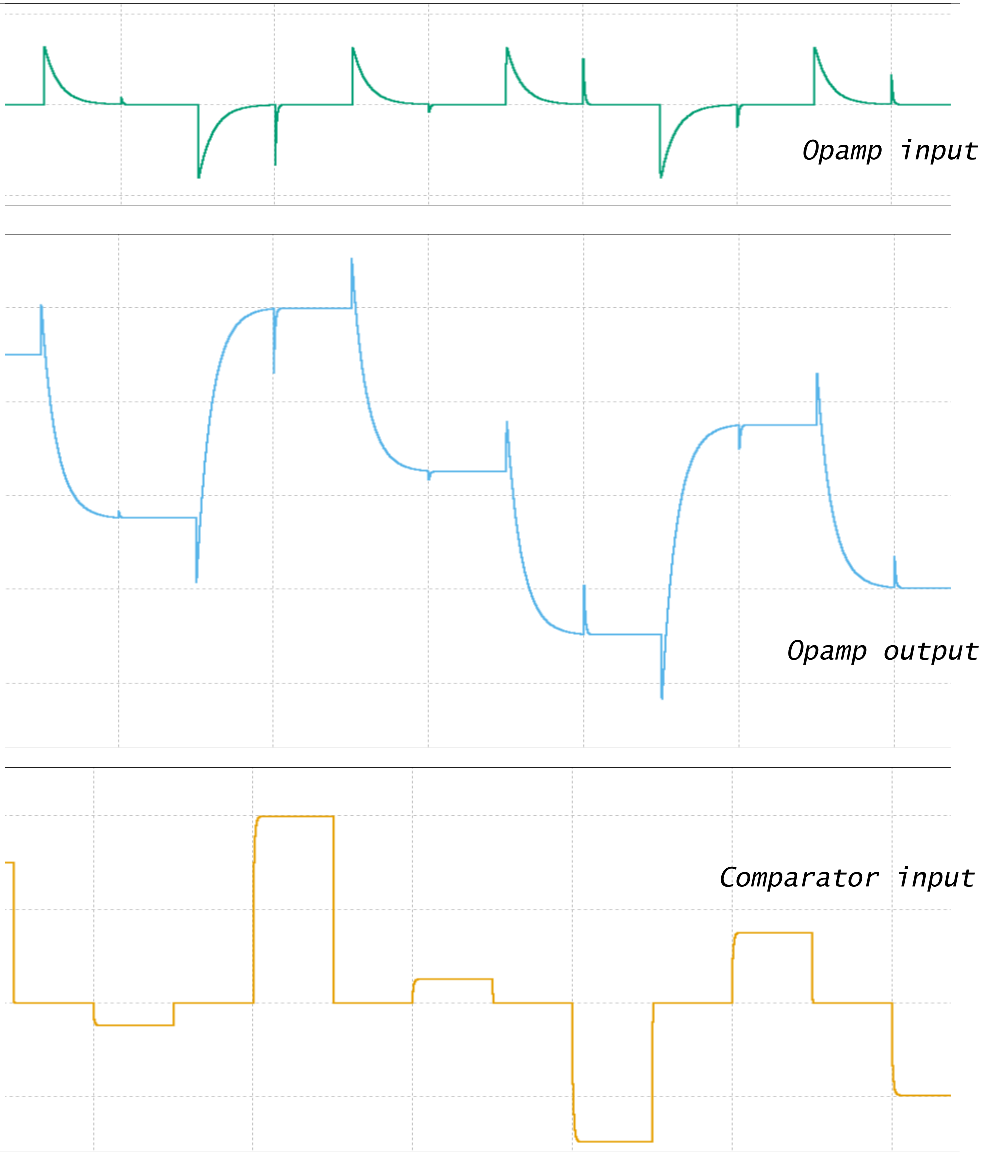


NAPA Simulation (Waveforms)

```
Administrateur : NAPA Compile and Run: Source File *** SD1.nap ***

[SD1]  **** MAC Preprocessor Running ****
[SD1]  **** NAPA Compiler    Running ****
[SD1]  **** GCC Compiler     Running ****
[SD1]  **** SARC Engine      Linking ****
[SD1]  **** Ad Hoc Simulator Running ****

NAPA Ping Information:     function 'duser_sarc()'      from file "/Simulate/NapaDos/Hdr/User/sarc.hdr"
NAPA Ping Information:     function 'Integrator2_NI()'  from file "Integrator2_NI.hdr"
NAPA Ping Information:     function 'rand_normal()'     from file "/Simulate/NapaDos/Hdr/Function/random.hdr"

 ****
 ****  Analysis in Time Domain [ Analog 1st Order SD Modulator with SARC model ]
 ****


NAPA Debug Information:     (           sarc[0])  function 'Integrator2_NI()'

 [ Integrator2_NI       ]     - model file     -

 [ Integrator2_NI       ]     "Integrator2_NI.hdr"

 [ Integrator2_NI       ]     - initialization -

 [ Integrator2_NI       ]     SARC computing rate    = (1 / 500.0 ps) =  2.000 GHz
 [ Integrator2_NI       ]     inputs         { Vin1, Vin2 }
 [ Integrator2_NI       ]     outputs        { V@Out, V@A, V@B }
 [ Integrator2_NI       ]     parameters     { C_11, C_12, C_2, C_3, C_a, C_o, Gds, PHIa, PHIb, Gm, Offs }
 [ Integrator2_NI       ]     parameter  #1                  sd1__C11 =  5.00000 p  ->   C_11
 [ Integrator2_NI       ]     parameter  #2                  sd1__C12 =  5.00000 p  ->   C_12
 [ Integrator2_NI       ]     parameter  #3                   sd1__C2 =  5.00000 p  ->   C_2
 [ Integrator2_NI       ]     parameter  #4                   sd1__C3 =  500.000 f  ->   C_3
 [ Integrator2_NI       ]     parameter  #5                  sd1__Aca =  200.000 f  ->   C_a
 [ Integrator2_NI       ]     parameter  #6                  sd1__Aco =  400.000 f  ->   C_o
 [ Integrator2_NI       ]     parameter  #7                 sd1__Agds =  26.0000 n  ->   Gds
 [ Integrator2_NI       ]     parameter  #8                 sd1__PHIa =  0.00000    ->   PHIa
 [ Integrator2_NI       ]     parameter  #9                 sd1__PHIb =  0.00000    ->   PHIb
 [ Integrator2_NI       ]     parameter  #10                 sd1__Agm =  299.997 u  ->   Gm
 [ Integrator2_NI       ]     parameter  #11               sd1__offsA =  1.93544 m  ->   Offs

 ****  Random Seed [I]  :               777808285 ****
 ****  Output Tag  [O]  :               906065082 ****

 ****  NAPA Compiler    :         V4.00 for Win64 ****

 ****  Main Netlist     :                 SD1.tmp ****

 ****  Simulator Time   :              10.0000 us ****
 ****  Simulator Index  :                   20001 ****

 ****  Run Time I/O     :                         ****

       ->  SD1-time.out                     [   O] ****

 ****  Stopwatch        :          H00:M00:S01.187 ****

 ****  Normal Termination                         ****
```

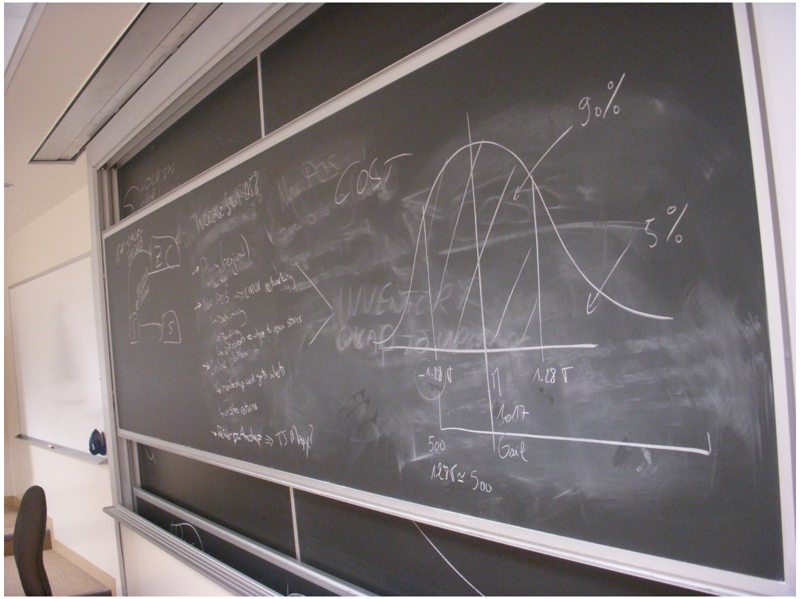
ΣΔ Modulator 1st Order





file ‘./SD1_Histo.nap’

```
header <napatool.hdr>

title  "Histogram Analysis in Time Domain"

fs       2.0e6
node     Clk      clock  "01" 200

string   opfile1  "./transconductance_opamp.dat"
string   opfile2  "./comparator.dat"

dvar     ampldb   -3.0
dvar     ampl     DB2LIN(ampldb, 1.0)
dvar     freq     1234.56789
dvar     ph       rand_uniform(0.0, _2pi_)

node     Vin      osc    0.0  ampl    freq    ph
node     Vref     dc     (analog)     1.0
node     Vout     cell   sd1  "./SD1.net"  Vin Vref   Clk  opfile1..2

tool     histoval stdout Ab Vref  -3.0 3.0  100

terminate    10000000 <= LOOP_INDEX

alias    Aa   sd1__Aa
alias    Ab   sd1__Ab
alias    Ao   sd1__Ao

debug    SARC
ping
```



NAPA Simulation (Histogram)

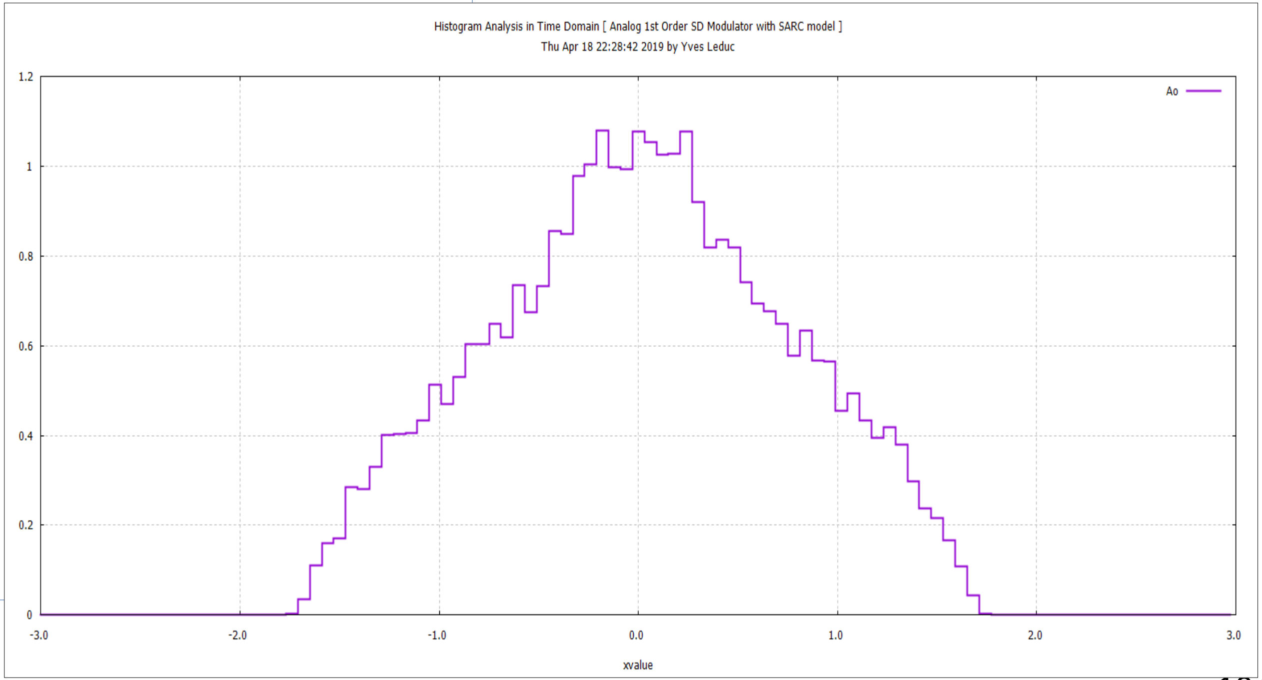
102

Opamp Output, Histogram

```
Administrateur : NAPA Compile and Run: Source File *** SD1_Histo.nap ***

[SD1_Histo]  **** MAC Preprocessor Running ****
[SD1_Histo]  **** NAPA Compiler    Running ****
[SD1_Histo]  **** GCC Compiler     Running ****
[SD1_Histo]  **** SARC Engine      Linking ****
[SD1_Histo]  **** Ad Hoc Simulator Running ****


NAPA Ping Information:     function 'duser_sarc()'      from file "/Simulate/NapaDos/Hdr/User/sarc.hdr"
NAPA Ping Information:     function 'itool_histoval()'  from file "/Simulate/NapaDos/Hdr/Tool/histo1.hdr"
NAPA Ping Information:     function 'Integrator2_NI()'  from file "Integrator2_NI.hdr"
NAPA Ping Information:     function 'rand_normal()'     from file "/Simulate/NapaDos/Hdr/Function/random.hdr"
NAPA Ping Information:     function 'rand_uniform()'    from file "/Simulate/NapaDos/Hdr/Function/random.hdr"

 ****
 ****  Histogram Analysis in Time Domain [ Analog 1st Order SD Modulator with SARC model ]
 ****



 ****  Random Seed [I]  :               777809661 ****
 ****  Output Tag  [O]  :                24529941 ****

 ****  NAPA Compiler    :           V4.00 for Win64 ****

 ****  Main Netlist     :             SD1_Histo.tmp ****

 ****  Simulator Time   :              49.9995 us ****
 ****  Simulator Index  :                  100000 ****
 ****  Tool Index       :                  100000 ****

 ****  Run Time I/O     :                         ****

       ->  stdout                          [   0] ****


 ****  Stopwatch        :           H00:M00:S00.219 ****


 ****  Normal Termination                         ****
```

The histograms are normalized

$$h[i] = \frac{number\_of\_occurrences}{bin\_width * number\_of\_samples}$$



file ‘sd1_FFT.nap’

```
header  <napatool.hdr>

title   "FFT Analysis"

fs      2000.0e6
node    Clk       clock   "01" 500

string  opfile1  "./transconductance_opamp.dat"
string  opfile2  "./comparator.dat"

dvar    ampldb   -3.0
dvar    ampl     DB2LIN(ampldb, 1.0)
dvar    freq     1234.56789
dvar    ph       rand_uniform(0.0, _2pi_)

node    Vin      osc  0.0  ampl  freq   ph
node    Vref     dc   (analog)   1.0
node    Vout     cell sd1  "./SD1.net"  Vin Vref  Clk  opfile1..2

ivar    npts     POWEROF2(16)

decimate 1000  500

tool    fft      stdout    Vout 1  npts

terminate    1 <= TOOL_INDEX

debug   SARC
ping
```

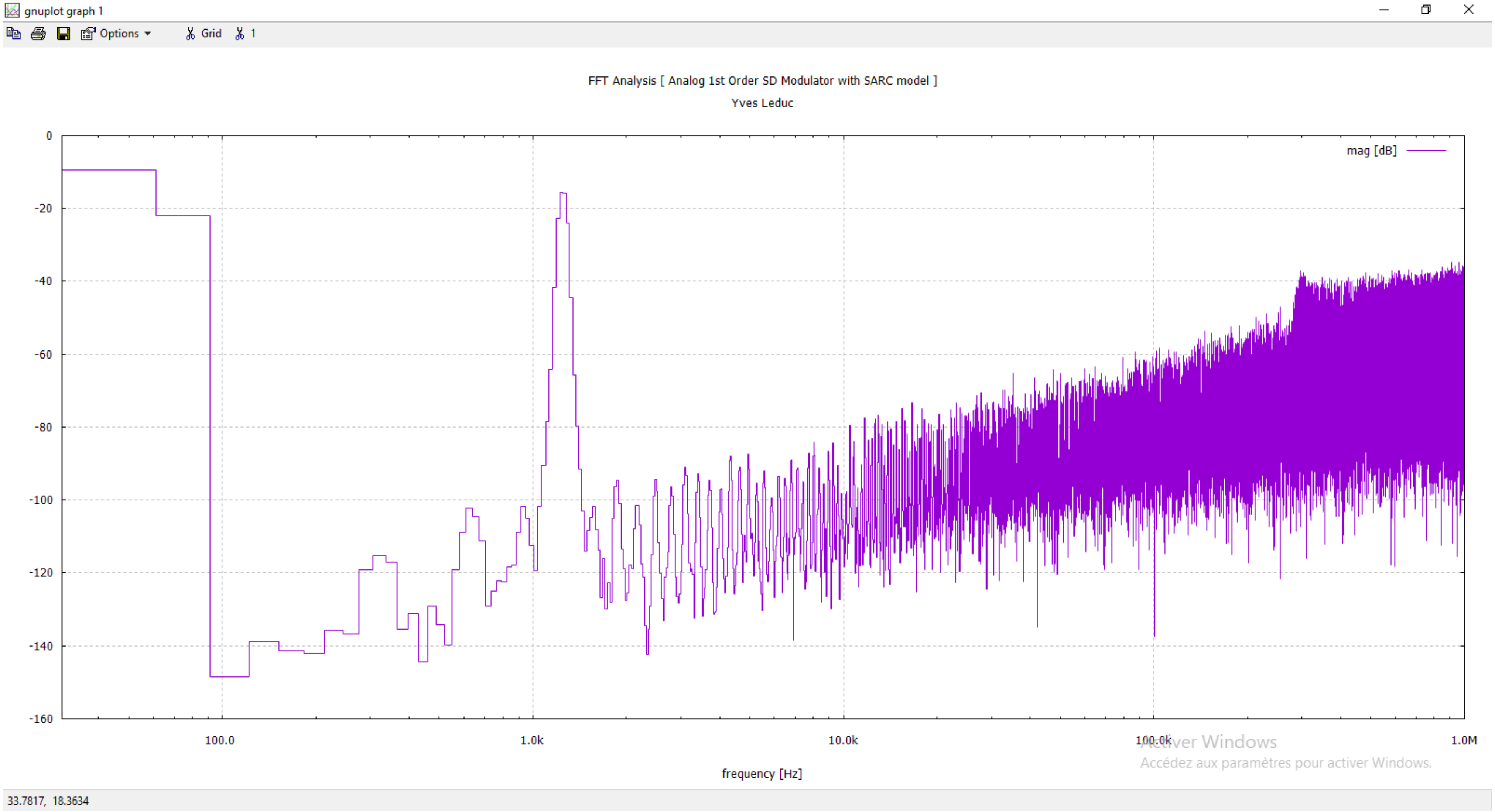
104

```
[SD1_FFT]  **** MAC Preprocessor Running ****
[SD1_FFT]  **** NAPA Compiler    Running ****
[SD1_FFT]  **** GCC Compiler     Running ****
[SD1_FFT]  **** SARC Engine      Linking ****
[SD1_FFT]  **** Ad Hoc Simulator Running ****


NAPA Ping Information:     function 'duser_sarc()'      from file "/Simulate/NapaDos/Hdr/User/sarc.hdr"
NAPA Ping Information:     function 'itool_fft()'       from file "/Simulate/NapaDos/Hdr/Tool/fft1.hdr"
NAPA Ping Information:     function 'Integrator2_NI()'  from file "Integrator2_NI.hdr"
NAPA Ping Information:     function 'rand_normal()'     from file "/Simulate/NapaDos/Hdr/Function/random.hdr"
NAPA Ping Information:     function 'rand_uniform()'    from file "/Simulate/NapaDos/Hdr/Function/random.hdr"

 ****
 ****  FFT Analysis [ Analog 1st Order SD Modulator with SARC model ]
 ****



NAPA Debug Information:     (           sarc[0])  function 'Integrator2_NI()'

 [ Integrator2_NI       ]     - model file     -

 [ Integrator2_NI       ]     "Integrator2_NI.hdr"

 [ Integrator2_NI       ]     - initialization -

 [ Integrator2_NI       ]     SARC computing rate   = (1 / 500.0 ps) =  2.000 GHz
 [ Integrator2_NI       ]     inputs          { Vin1, Vin2 }
 [ Integrator2_NI       ]     outputs         { V@Out, V@A, V@B }
 [ Integrator2_NI       ]     parameters      { C_11, C_12, C_2, C_3, C_a, C_o, Gds, PHIa, PHIb, Gm, Offs }
 [ Integrator2_NI       ]     parameter  #1                     sd1__C11 =  5.00000 p  ->   C_11
 [ Integrator2_NI       ]     parameter  #2                     sd1__C12 =  5.00000 p  ->   C_12
 [ Integrator2_NI       ]     parameter  #3                      sd1__C2 =  5.00000 p  ->   C_2
 [ Integrator2_NI       ]     parameter  #4                      sd1__C3 =  500.000 f  ->   C_3
 [ Integrator2_NI       ]     parameter  #5                     sd1__Aca =  200.000 f  ->   C_a
 [ Integrator2_NI       ]     parameter  #6                     sd1__Aco =  400.000 f  ->   C_o
 [ Integrator2_NI       ]     parameter  #7                    sd1__Agds =  26.0000 n  ->   Gds
 [ Integrator2_NI       ]     parameter  #8                    sd1__PHIa =  0.00000    ->   PHIa
 [ Integrator2_NI       ]     parameter  #9                    sd1__PHIb =  0.00000    ->   PHIb
 [ Integrator2_NI       ]     parameter  #10                    sd1__Agm =  299.997 u  ->   Gm
 [ Integrator2_NI       ]     parameter  #11                  sd1__offsA = -1.12561 m  ->   Offs

NAPA Tools Information:     (            fft[0]) Process  # 000      <- 65535500


 ****  Random Seed [I]  :                 777811527 ****
 ****  Output Tag  [O]  :                 153099792 ****

 ****  NAPA Compiler    :            V4.00 for Win64 ****

 ****  Main Netlist     :               SD1_FFT.tmp ****

 ****  Simulator Time   :               32.7677 ms ****
 ****  Simulator Index  :               65 535 501 ****
 ****  Tool Index       :                        1 ****

 ****  Run Time I/O     :                          ****

       ->  stdout                            [  0] ****


 ****  Stopwatch        :            H00:M01:S43.700 ****


 ****  Normal Termination                          ****
```

65 millions cycles

1 min 44

NAPA Simulation (FFT)


gnuplot graph 1
Options
Grid
1
FFT Analysis [Analog 1st Order SD Modulator with SARC model]
Yves Leduc
mag [dB]
0
-20
-40
-60
-80
-100
-120
-140
-160
100.0
1.0k
10.0k
100.0k
1.0M
frequency [Hz]
Activer Windows
Accédez aux paramètres pour activer Windows.
33.7817, 18.3634

Sampling and Local Sampling Frequencies: FS, FSL

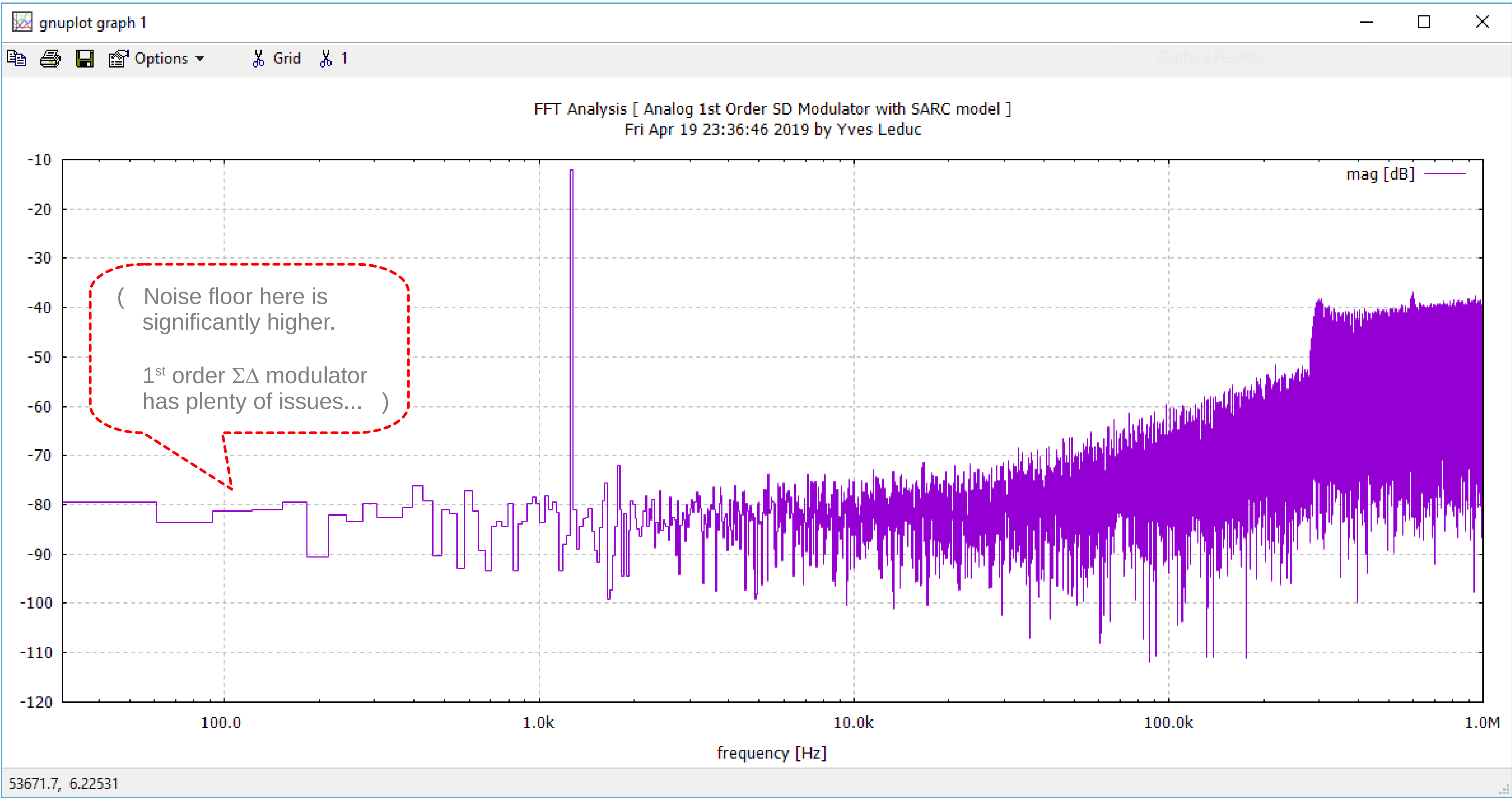
file ‘sd1b_FFT.nap’

```
header <napatool.hdr>

title  "FFT Analysis"

fs        2000.0e6

node     Clk       clock  "01" 500

string   opfile1   "./transconductance_opamp.dat
string   opfile2   "./comparator.dat"

dvar     ampldb    -3.0
dvar     ampl      DB2LIN(ampldb, 1.0)

dvar     ph        rand_uniform(0.0, _2pi_)

node     Vin       osc   0.0  ampl   freq   ph
node     Vref      dc    (analog)    1.0
node     Vout      cell sd1  "./SD1.net"  Vin Vref  Clk  opfile1..2

ivar     npts      POWEROF2(16)

decimate 1000 500

tool     fft       stdout     Vout 1  npts
dvar     freq      coherent(1234.56789, npts)

terminate    1 <= TOOL_INDEX

debug      SARC    COHERENT
directive WINDOW  NONE
ping
```

```
NAPA Ping Information:     function 'duser_sarc()'      from file "/Simulate/NapaDos/Hdr/User/sarc.hdr"
NAPA Ping Information:     function 'itool_fft()'       from file "/Simulate/NapaDos/Hdr/Tool/fft1.hdr"
NAPA Ping Information:     function 'Integrator2_NI()'  from file "Integrator2_NI.hdr"
NAPA Ping Information:     function 'coherent()'        from file "/Simulate/NapaDos/Hdr/Function/coherent.hdr"
NAPA Ping Information:     function 'rand_normal()'     from file "/Simulate/NapaDos/Hdr/Function/random.hdr"
NAPA Ping Information:     function 'rand_uniform()'    from file "/Simulate/NapaDos/Hdr/Function/random.hdr"
 ****
 ****  FFT Analysis [ Analog 1st Order SD Modulator with SARC model ]
 ****


NAPA Debug Information:     (      coherent_freq)
 Sampling frequency      : 2.00000000 MHz
 Frequency target        : 1.23456789 kHz
 Coherent frequency      : 1.25122070 kHz,   Delta:  16.6528131  Hz  ( +1.3% )
   # samples             :      65536
   # periods of signal   :         41
```

- macro **FS** defined here as **2.0e9** (Hz)
- macro **FSL** defined here as **2.0e9** (Hz)

- macro **FS** defined here as **2.0e9** (Hz)
- macro **FSL** defined here as **2.0e6** (Hz)


gnuplot graph 1
Options
Grid
1
FFT Analysis [Analog 1st Order SD Modulator with SARC model]
Fri Apr 19 23:36:46 2019 by Yves Leduc
mag [dB]
-10
-20
-30
-40
-50
-60
-70
-80
-90
-100
-110
-120
100.0
1.0k
10.0k
100.0k
1.0M
frequency [Hz]
(Noise floor here is significantly higher.
1st order ΣΔ modulator has plenty of issues...)
53671.7, 6.22531

NAPA Simulation, TSNR

file ‘sd1_TSNR.nap’

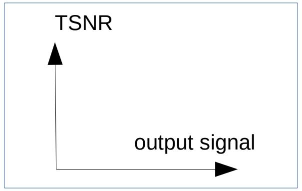
```
header <napatool.hdr>

title  "TSNR Analysis"

fs      200.0e6
node    Clk      clock   "01"   50

string  opfile1 "./transconductance_opamp.dat"
string  opfile2 "./comparator.dat"

dvar    ampldb  LINSWEEP(TOOL_INDEX, -50.0,0.0, 26)  &update &export
dvar    ampl    DB2LIN(ampldb, 1.0)                  &update
dvar    freq    1234.56789                           &constant
dvar    ph      rand_uniform(0.0, _2pi_)             &constant

node    Vin     osc    0.0  ampl  freq   ph
node    Vref    dc     1.0
node    Vout    cell   sd1  "./SD1.net"   Vin Vref  Clk  opfile1..2

ivar    npts    POWEROF2(16)

decimate   100  50

node    Fout    wsum    2 Vout  -1 One
tool    tsnr    stdout  Fout 1  8.0e3 npts  (template)  (psophometr

terminate   0.0 < ampldb

ping
```

```
...
#define  POW10(y)             pow(10.0,(y))
#define  LINDOMAIN(c,b,e)     ((b)+((c)*((e)-(b))))
#define  LOGDOMAIN(c,b,e)     ((b)*POW10((c)*LOG10(((double)(e))/((double)(b)))))
#define  LINSWEEP(c,b,e,n)    LINDOMAIN((((double)(c))/((double)((n)-1L))),(b),(e))
#define  LOGSWEEP(c,b,e,n)    LOGDOMAIN((((double)(c))/((double)((n)-1L))),(b),(e))
...
#define  DB2LIN(x,r)          ((r)*POW10(0.05*(x)))
...
#define  POWEROF2(n)          ((0LL<((long long)(n)))?(1LL<<((long long)(n))):1LL)
...
```

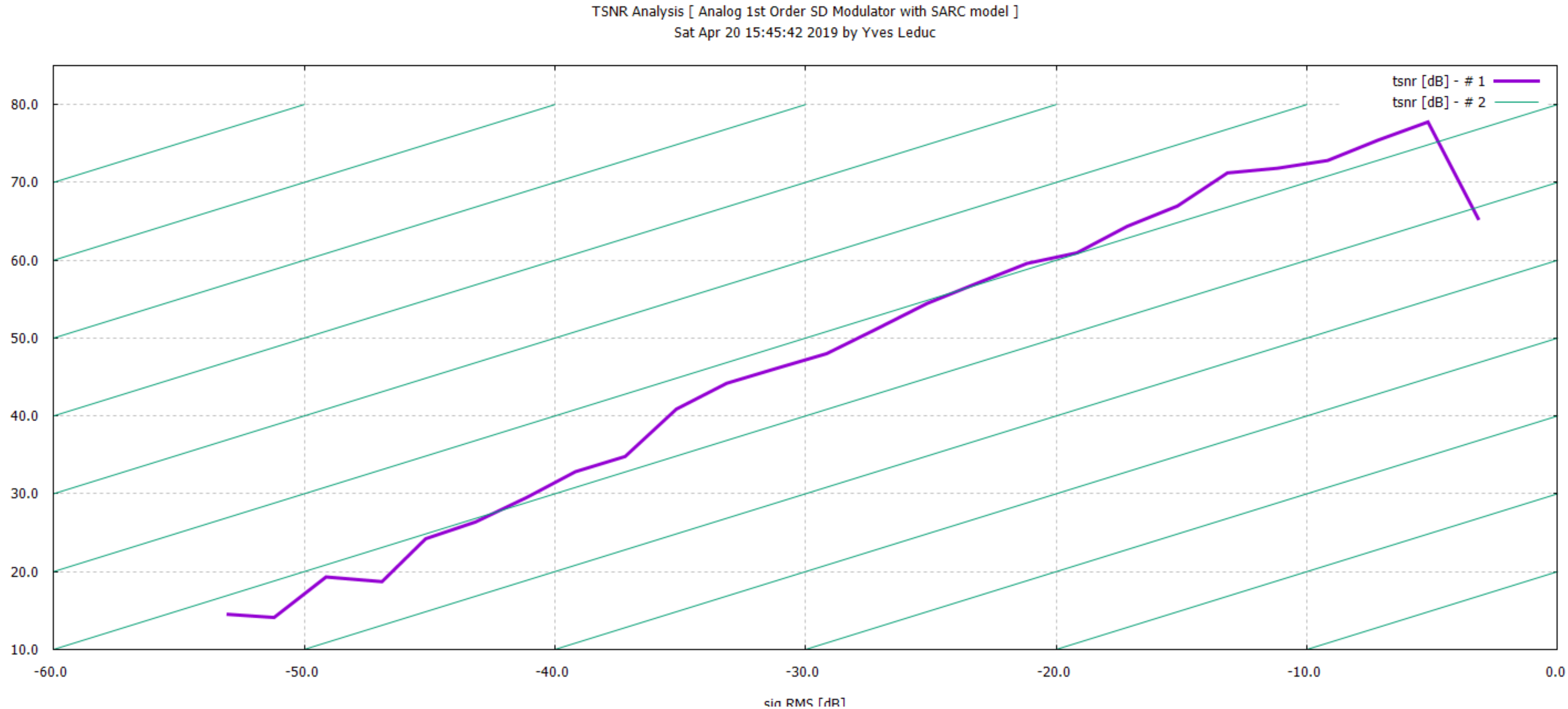
```
NAPA Tools Information:     (        tsnr[0]) Process  # 023      <- 157286350
NAPA Tools Information:     (        tsnr[0]) Process  # 024      <- 163839950
NAPA Tools Information:     (        tsnr[0]) Process  # 025      <- 170393550
NAPA Tools Information:     (        tsnr[0]) Appending ad hoc template to tsnr data

 ****  Random Seed [I]  :            777883971 ****
 ****  Output Tag  [O]  :             68656489 ****

 ****  NAPA Compiler    :       V4.00 for Win64 ****

 ****  Main Netlist     :          SD1_TSNR.tmp ****

 ****  Simulator Time   :            851.968 ms ****
 ****  Simulator Index  :           170 393 552 ****
 ****  Tool Index       :                    26 ****

 ****  Run Time I/O     :                       ****

       ->  stdout                          [  0] ****

 ****  Stopwatch        :       H00:M07:S02.564 ****
```

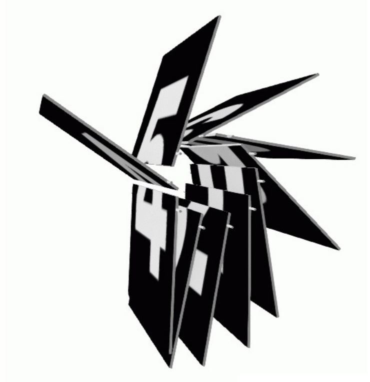
7 minutes ...

TSNR




```
...
tool  tsnr  stdout  Fout 1  8.0e3  npts  (template)
...
```





3^{rd} Order ΣΔ Analog Modulator

Z Domain Model of a SWC Integrator

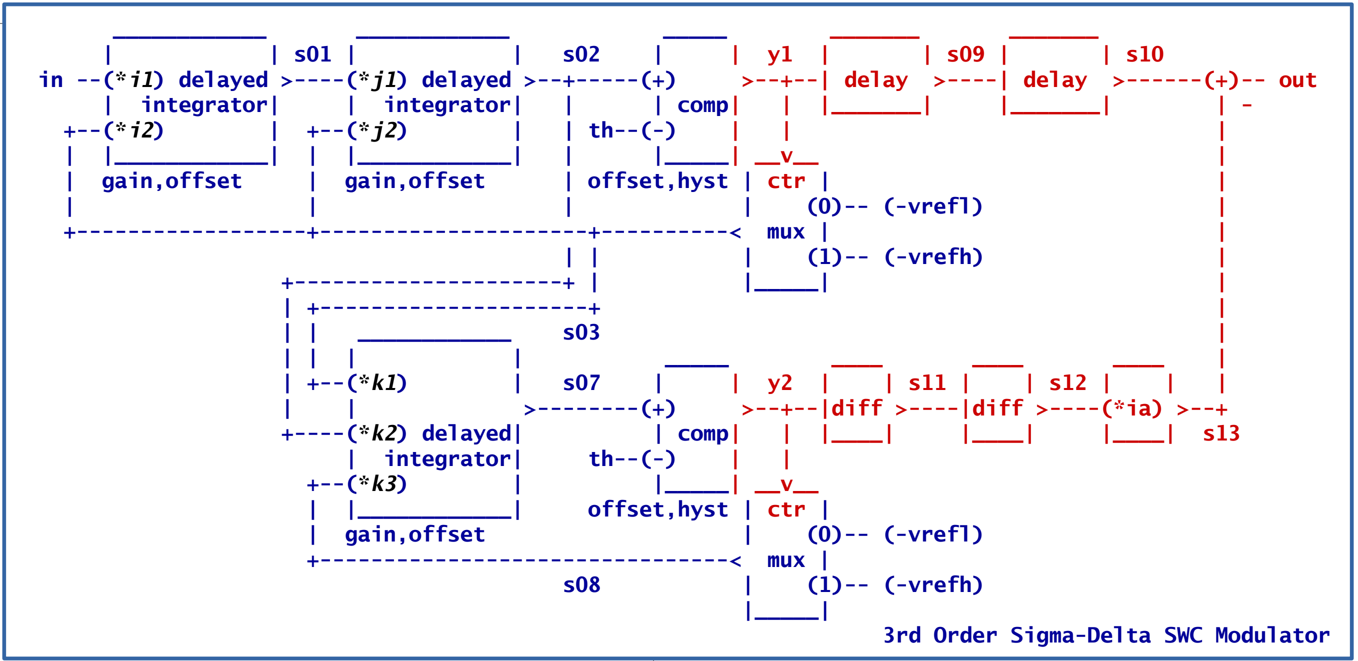
library file '/Simulate/NapaDos/Net/Integrator1/d2_a.net'

```
cell_interface  $out  $datfile  $k1 $in1p  $k2 $in2p  $init

#*                     _____      _______
#*         s02p |     | s02pd |       |
#* in2p --(+)-----|delay >------|  *k2     >-(+)      mu = 1/A
#*           |      |_____|      |_______|   |
#* off ----+          _____      _______     |        k1 = C1/Cs
#*         s01p |     | s01pd |       |     |        k2 = C2/Cs
#* in1p --(+)-----|delay >------|  *k1     >-(+)
#*           |      |_____|      |_______|   |
#* off ----+          _____      _______     |   ____________________
#*                |     | outd  |       |    |  |                    |
#*        +------|delay >------|*(1+mu) >-(+)--|*1/(1+mu+(k1+k2)*mu) >-+-- out
#*        |      |_____|      |_______|   s11|____________________|  |
#*        +--------------------------------------------------------------+

data   $datfile     $gain $off

dvar   $mu    1.0/$gain
dvar   $g11   1.0/(1.0+$mu+(($k1+$k2)*$mu))
dvar   $g13   1.0+$mu

node   $s01p  offset $off $in1p
node   $s01pd delay  $s01p
node   $s02p  offset $off $in2p
node   $s02pd delay  $s02p
node   $s11   wsum   $k1 $s01pd  $k2 $s02pd  $g13 $outd
node   $out   gain   $g11 $s11
node   $outd  delay  $out

init   $outd  $init
```

```
               C2
                ||
in2p --PHIa---+---||---+---PHIb---+
              |   ||   |          |
              |        |          |
             PHIb     PHIa        |
              |        |          |
             _|_      _|_         |              Cs
                                  |              ||
                                  |         +-----||---+
               C1                 |         |    ||   |
                ||                |         |  |\     |
in1p --PHIa---+---||---+---PHIb---+---+--|  \    |
              |   ||   |                   | A  >--+---PHIa---
              |        |            +--|  /
             PHIb     PHIa          |  |/
              |        |           off
             _|_      _|_          _|_
```

file “./SD21.net”

```
cell_interface $out $sdfile $in $vh $vl

data $sdfile            ...
  $A $B $C $D $E $F    ...
  $op1 $op2 $op3 $cmp4 $cmp5  // data files

declare  (true) ISINTEGER($F)

dvar $i1  $A
dvar $i2  $A/$C
dvar $j1  $B/$A
dvar $j2  2.0*($B/$C)
dvar $k1  $D/($C*I2D($F))
dvar $k2  $D/($B*I2D($F))
dvar $k3  $D/$E
ivar $ia  $F

node $vp  dalgebra         $vh
node $vn  dalgebra         $vl
node $th  average          $vn   $vp
node $s01 cell  i1  <Integrator1/d2_a.net>  $op1   $i1  $in   $i2 $s03  0.0
node $s02 cell  i2  <Integrator1/d2_a.net>  $op2   $j1  $s01  $j2 $s03  0.0
node $y1  cell  c4  <Comparator/2_a.net>    $cmp4  $s02 $th
node $s03 mux              $y1   -$vn -$vp
node $s07 cell  i3  <Integrator1/d3_a.net>  $op3   $k1  $s03  $k2 $s02  $k3 $s08  0.0
node $y2  cell  c5  <Comparator/2_a.net>    $cmp5  $s07 $th
node $s08 mux              $y2   -$vn -$vp

node $s09 delay            $y1
node $s10 delay            $s09
node $s11 differentiator $y2
node $s12 differentiator $s11
node $s13 gain             $ia   $s12
node $out sum              $s10  $s13
```



3^{rd} Order SWC ΣΔ Modulator Cell

3rd Order ΣΔ SWC Modulator, Parameters

```
data_interface   $gain  $offset
#*  Opamp of the integrator '1'

dvar  $gain     1000.0
dvar  $offset   rand_normal(0.0, 3.0e-3)
```

```
data_interface   $gain  $offset
#*  Opamp of the integrator '2'

dvar  $gain     1000.0
dvar  $offset   rand_normal(0.0, 3.0e-3)
```

```
data_interface   $gain  $offset
#*  Opamp of the integrator '3'

dvar  $gain     800.0
dvar  $offset   rand_normal(0.0, 5.0e-3)
```

```
data_interface   $hysteresis  $offset
#*  Comparator '4'

dvar  $hysteresis 0.0
dvar  $offset    rand_normal(0.0, 2.0e-3)
```

```
data_interface   $hysteresis  $offset
#*  Comparator '5'

dvar  $hysteresis 0.0
dvar  $offset    rand_normal(0.0, 4.0e-3)
```

file './_sd21.dat'

```
data_interface  $A $B $C $D $E $F $op1 $op2 $op3 $cmp4 $cmp5

#* There are 6 independent parameters (A,B,C,D,E,F)

dvar  $A     1.0   // scaling of 1st integrator stage
dvar  $B     1.0   // scaling of 2nd integrator stage
dvar  $C     1.0   // scaling of 1st quantizer
dvar  $D     1.0   // scaling of 3rd integrator stage
dvar  $E     1.0   // scaling of 2nd quantizer
ivar  $F     2     // interstage attenuation

#*  files containing the opamp characteristics
string  $op1  "./_op1.dat"
string  $op2  "./_op2.dat"
string  $op3  "./_op3.dat"

#*  files containing the comparator characteristics
string  $cmp4 "./_cmp4.dat"
string  $cmp5 "./_cmp5.dat"
```

114

3^{rd} Order ΣΔ SWC Modulator

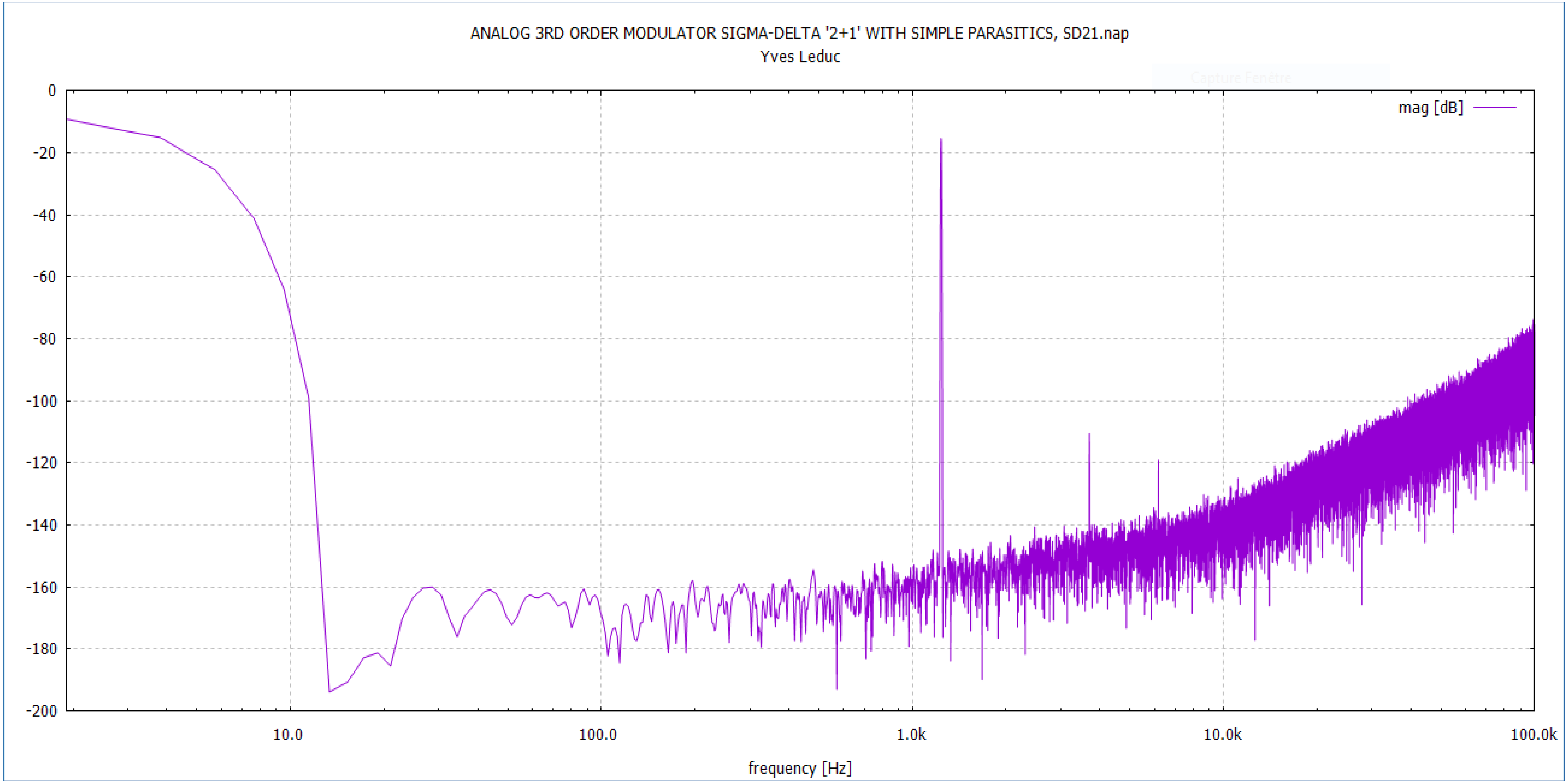
file “./SD21.nap”

```
title  "ANALOG 3RD ORDER MODULATOR SIGMA-DELTA '2+1' WITH SIMPLE PARASITICS, $F"

header <napatool.hdr>

fs      2.0e6

node    vrefh    dc   1.0
node    vrefl    dc  -1.0
node       in   osc   0.0 0.8 1234.56789 0.0
node      out  cell   sd  "./sd21.net"  "./_sd21.dat"  in  vrefh vrefl

ivar      npts  POWEROF2(20)

tool       fft  "fft.out"    out 1 1.0 100.0e3  npts
tool histoval  "histo.out"  A 1.0  -5.0 5.0 100
tool histoval  "histo.out"  B 1.0  -5.0 5.0 100
tool histoval  "histo.out"  C 1.0  -5.0 5.0 100

terminate    1 <= TOOL_INDEX

ping
directive   WINDOW BLACKMAN_HARRIS_7

alias  A  sd__s01
alias  B  sd__s02
alias  C  sd__s07
```

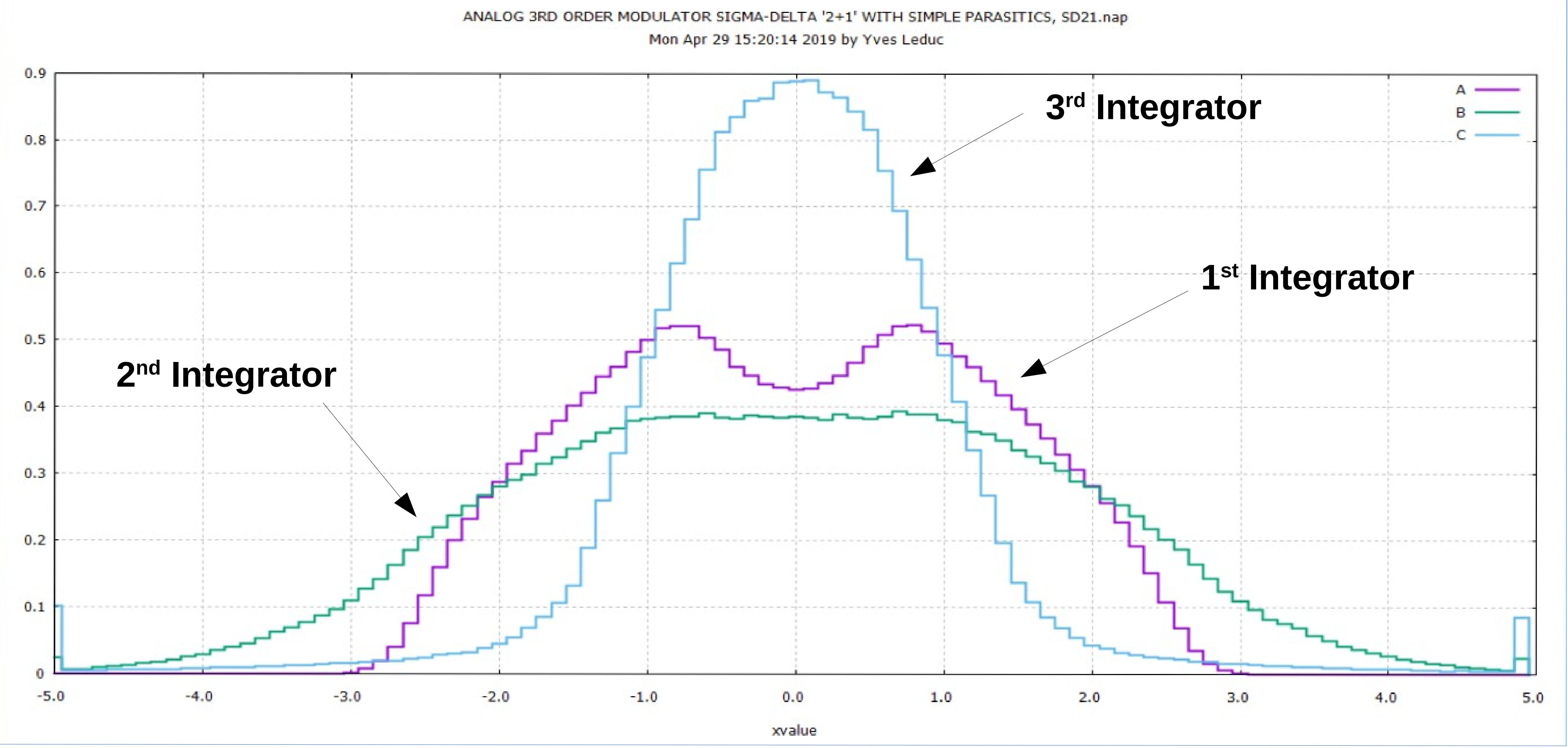
```
****  Random Seed [I]   :           780386915 ****
****  Output Tag  [O]   :            37335564 ****

****  NAPA Compiler     :     V4.02 for Win64 ****

****  Main Netlist      :            SD21.tmp ****

****  Simulator Time    :          524.288 ms ****
****  Simulator Index   :           1 048 576 ****
****  Tool Index        :                   1 ****

****  Run Time I/O      :                     ****

      ->  fft.out                        [  0] ****
      ->  histo.out                      [  0] ****


****  Stopwatch         :     H00:M00:S01.569 ****


****  Normal Termination                      ****
```

1.6 second

3rd Order ΣΔ SWC Modulator, 1 Million points FFT



3^{rd} Order ΣΔ SWC Modulator, 1 Million Samples Histograms



3rd Order ΣΔ SWC Modulator

file "./TSNR.nap"

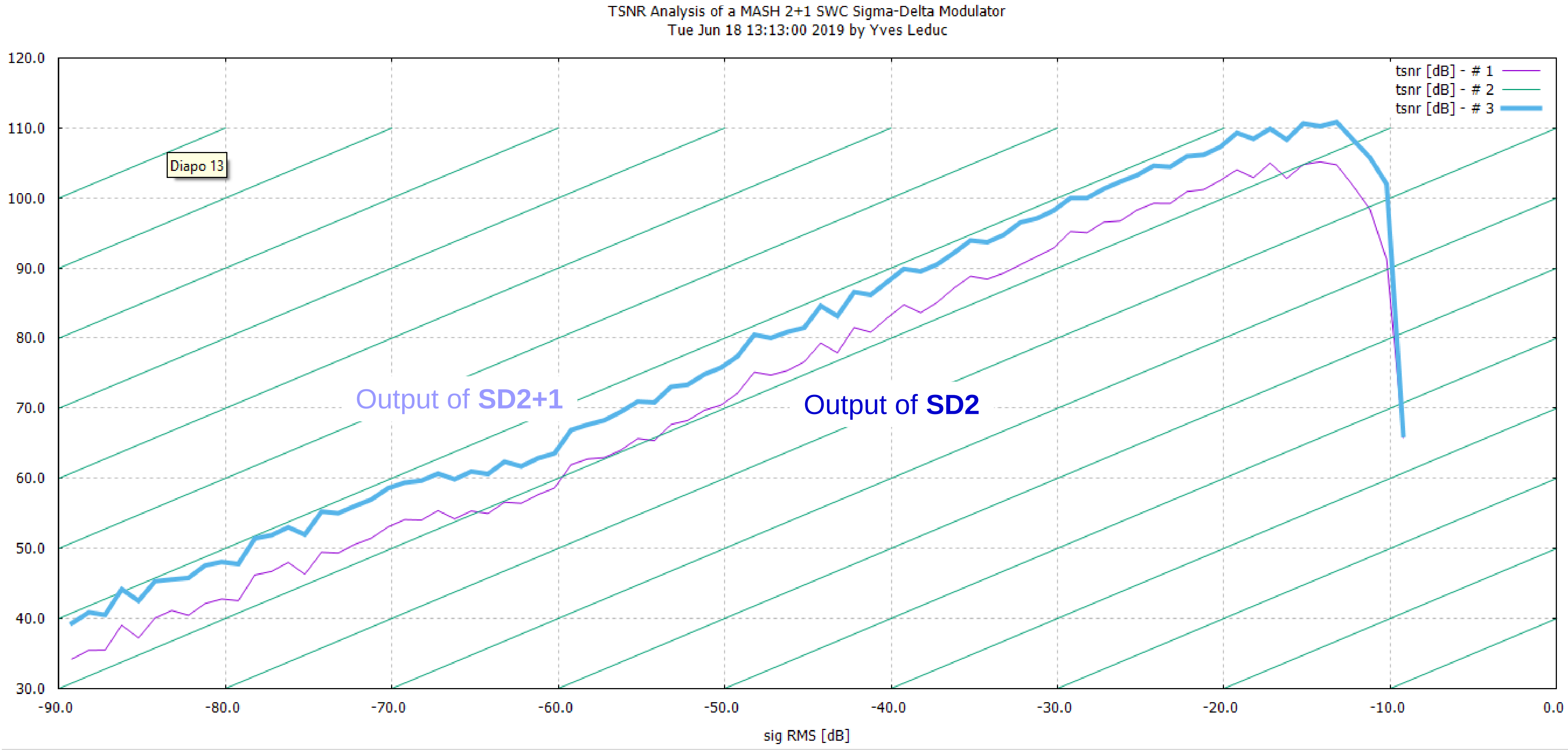
```
header <napatool.hdr>

title  "TSNR Analysis of a MASH 2+1 SWC Sigma-Delta Modulator"

fs     2.0e6

dvar ampldb   LINSWEEP(TOOL_INDEX, -80.0,0.0, 81) &update &export
dvar   ampl   DB2LIN(ampldb, 1.0)                 &update
dvar   freq   1234.56789
dvar     ph   rand_uniform(0.0, _2pi_)

node  vrefh   dc    1.0
node  vrefl   dc   -1.0
node     in   osc   0.0 ampl  freq ph
node    out   cell   sd  "./sd21.net"  "./_sd21.dat" in vrefh vrefl

ivar    npts  POWEROF2(16)

tool    tsnr  "tmp2.out" O2 1 10.0 16.0e3 npts (psophometric) (templ
post    join  stdout    (keep)
tool    tsnr  "tmp3.out" O3 1 10.0 16.0e3 npts (psophometric)
post    join  stdout    (keep)

terminate     0.0 < ampldb

alias     O2  sd1__y1    // output of SD2
alias     O3  out        // output of SD2+1

ping
directive     WINDOW  BLACKMAN_HARRIS_7
directive     QUIET
```

```
****
****  TSNR Analysis of a MASH 2+1 SWC Sigma-Delta Modulator
****

****            1.000 k       <-  A1
****            1.000 k       <-  A2
****            800.0         <-  A3

****            556.6 uV      <-  Offs1
****           -1.068 mV      <-  Offs2
****            4.323 mV      <-  Offs3

NAPA Tools Information:       (          tsnr[0]) Appending ad hoc template to tsnr data

NAPA Posts Information:       (          join[0]) Append 2 Files as requested

****  Random Seed [I]   :               778781753 ****
****  Output Tag  [O]   :               106702948 ****

****  NAPA Compiler     :          V4.00 for Win64 ****

****  Main Netlist      :                TSNR.tmp ****

****  Simulator Time    :              2.65421  s ****
****  Simulator Index   :               5 308 417 ****
****  Tool Index        :                      81 ****

****  Run Time I/O      :                         ****

      <>  tmp2.out                          [I/O] ****
      <>  tmp3.out                          [I/O] ****
      ->  stdout                            [  O] ****

****  Stopwatch         :          H00:M00:S02.424 ****

****  Normal Termination                          ****
```

< 2.7 seconds

162 FFTs on 2^{16} points
5.3 millions samples

3rd Order ΣΔ SWC Modulator, TSNR



Conclusions

30 years after the first prototype, NAPA has proven an impressive extensibility and its ability to support state-of-the-art mixed signal and mixed technology applications.

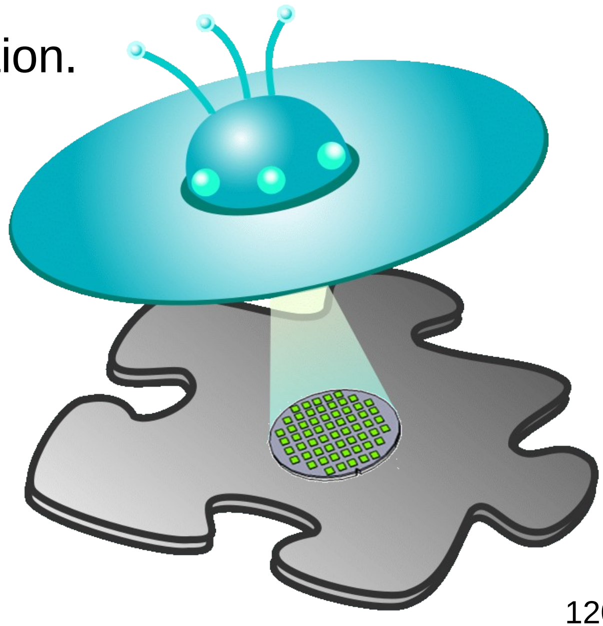
It is built with a solid foundation and takes profit of the side effects of its building block.

It is fast, so fast.

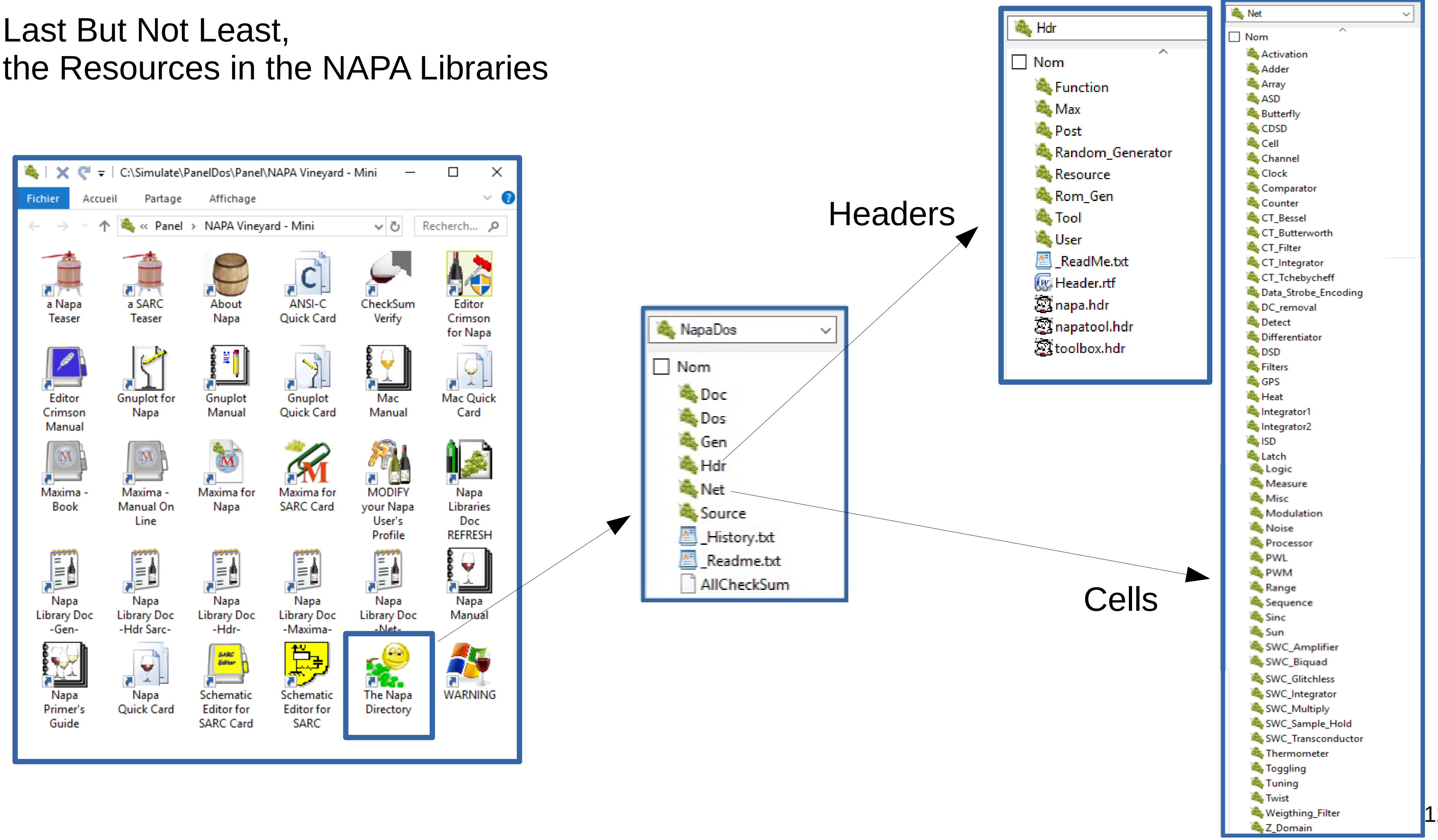
It is not verbose, still it lets the user a solid control of the simulation.
It is wide open and has a crystal clear implementation.

It is free, there is no hassle to manage licenses and therefore can be deployed without any constraint.

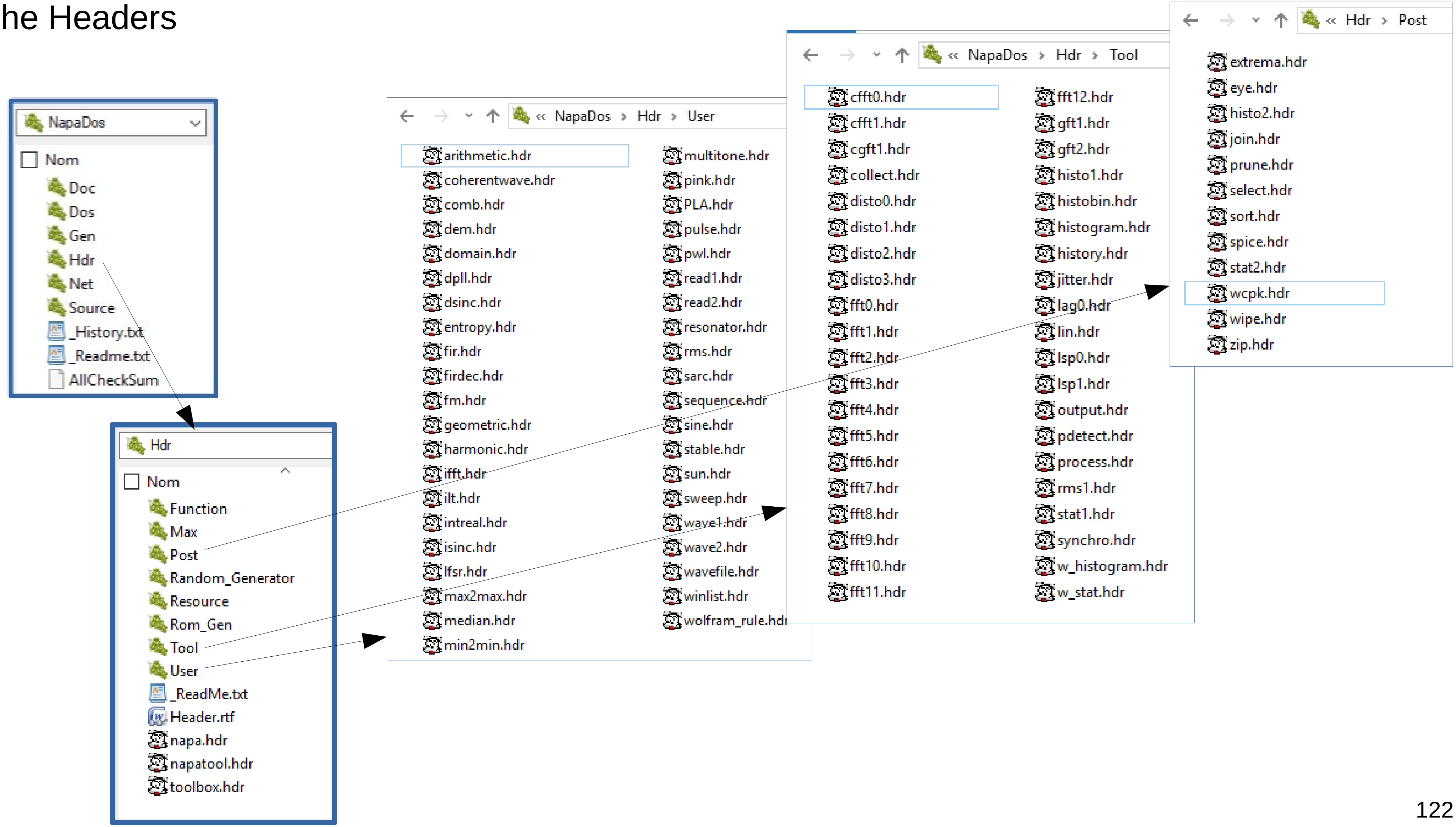
It is ready to welcome new functions and new features in applications yet to be invented.



Last But Not Least,
the Resources in the NAPA Libraries



The Headers



NapaDos
Nom
Doc
Dos
Gen
Hdr
Net
Source
_History.txt
_Readme.txt
AllCheckSum
Hdr
Nom
Function
Max
Post
Random_Generator
Resource
Rom_Gen
Tool
User
_ReadMe.txt
Header.rtf
napa.hdr
napatool.hdr
toolbox.hdr
« NapaDos › Hdr › User
arithmetic.hdr
coherentwave.hdr
comb.hdr
dem.hdr
domain.hdr
dpll.hdr
dsinc.hdr
entropy.hdr
fir.hdr
firdec.hdr
fm.hdr
geometric.hdr
harmonic.hdr
ifft.hdr
ilt.hdr
intreal.hdr
isinc.hdr
lfsr.hdr
max2max.hdr
median.hdr
min2min.hdr
multitone.hdr
pink.hdr
PLA.hdr
pulse.hdr
pwl.hdr
read1.hdr
read2.hdr
resonator.hdr
rms.hdr
sarc.hdr
sequence.hdr
sine.hdr
stable.hdr
sun.hdr
sweep.hdr
wave1.hdr
wave2.hdr
wavefile.hdr
winlist.hdr
wolfram_rule.hdr
« NapaDos › Hdr › Tool
cfft0.hdr
cfft1.hdr
cgft1.hdr
collect.hdr
disto0.hdr
disto1.hdr
disto2.hdr
disto3.hdr
fft0.hdr
fft1.hdr
fft2.hdr
fft3.hdr
fft4.hdr
fft5.hdr
fft6.hdr
fft7.hdr
fft8.hdr
fft9.hdr
fft10.hdr
fft11.hdr
fft12.hdr
gft1.hdr
gft2.hdr
histo1.hdr
histobin.hdr
histogram.hdr
history.hdr
jitter.hdr
lag0.hdr
lin.hdr
lsp0.hdr
lsp1.hdr
output.hdr
pdetect.hdr
process.hdr
rms1.hdr
stat1.hdr
synchro.hdr
w_histogram.hdr
w_stat.hdr
« Hdr › Post
extrema.hdr
eye.hdr
histo2.hdr
join.hdr
prune.hdr
select.hdr
sort.hdr
spice.hdr
stat2.hdr
wcpk.hdr
wipe.hdr
zip.hdr

alias
array
assert
call
command_line
comment
data
debug
decimate
declare
directive
drop
dump
dvar
error
event
export
format
fs
ganging
gateway
header
init
Inject
input
cell_interface
data_interface
interlude
interpolate
ivar
load
napa_version
node
nominal
num_initial
opcode
output
ping
post
random_seed
restart
stuck
string
synchronize
terminate
title
tool
ts
update
void
warning
#*

<instruction_keyword>   [<parameter...>]

Instructions

Nodes

node <node_name> <kind> [<parm...>] [<node_name...>]

adc
algebra
alu
and
average
bshift
btoi
buffer
bwand
bwbuffer
bwinv
bwnand
bwnor
bwnot
bwor
bwxnor
bwxor
cell
change
clip
clock
comp
copy
cosine

dac
dalgebra
dc
delay
differentiator
div
dtoi
dtool
duser
equal
fzand
fzbuffer
fzinv
fzor
fznand
fznor
fznot
gain
generator
hold
ialgebra
integrator
Inv
itob

itod
itool
iuser
latch
lshift
max
merge
min
mod
rom2
rshift
rshift1
rshift2
sign
sine
square
step
sub
sum
toggle
track
triangle
trig
uadc

udac
wsum
xnor
xor
zero

User, Tool and Post Functions

iuser_arithmetic_average
duser_arithmetic_average
duser_coherentwave
duser_comb
iuser_dem
duser_dpll
duser_dsinc
duser_entropy
duser_fir
iuser_fir
duser_fir_in
duser_fir_out
duser_fm
iuser_fm
duser_geometric_average
duser_harmonic_average
duser_ifft
duser_ilt
iuser_lfsr
duser_max2max
iuser_max2max
duser_median
iuser_median
duser_min2min
iuser_min2min
duser_intreal
duser_multitone
duser_pink
iuser_PLA
duser_pulse
duser_pwl
iuser_pwl
duser_read
duser_read2
iuser_read
iuser_read2
duser_resonator
duser_rms_average
iuser_sequence
duser_sarc
duser_sine
iuser_stable
duser_sun
duser_synchro_lindomain
duser_synchro_logdomain
duser_synchro_linsweep
duser_synchro_logsweep
duser_synchro_readsweep
iuser_wave1x16_in
iuser_wave1x16_out
iuser_wave2x16_in
iuser_wave2x16_out
duser_win_list
iuser_wolfram_rule

(in '/Simulate/NapaDos/Hdr/User/..')

itool_autocorr
itool_cfft
itool_cgft
itool_cwin
itool_disto
itool_enbw
itool_fft
itool_fft_cs
itool_freq
itool_gdel
itool_gft
itool_hdecomp
itool_histobin
itool_history
itool_histogram
itool_histoslp
itool_histoval
itool_icn
itool_i2decomp
itool_i3decomp
itool_im2
itool_im3
itool_lag
itool_lin
itool_lsp
itool_lspwin
itool_output
itool_pdetect
itool_ps
itoll_ptf
itool_quinn2
itool_resp
itool_resp2_i
itool_resp2_o
itool_rms
itool_sinewave
itool_statslp
itool_statval
itool_synchro
itool_tdecomp
itool_tf
itool_tf2_i
itool_tf2_o
itool_tfp
itool_tsnr
itool_win
itool_xcorr

(in '/Simulate/NapaDos/Hdr/Tool/..')

post_extrema
post_eye
post_histo
post_join
post_prune
post_select
post_sort
post_spice
post_stat
post_wcpk
post_wcpk_pwl
post_wipe
post_zip

(in '/Simulate/NapaDos/Hdr/Post/..')

Registered C Functions

arithmetic_mean( )
arithmetic_geometric_mean( )
AWG_d( )
AWG_n( )
AWG_s()
bessel_i( )
bessel_j( )
bessel_k( )
bessel_y( )
c2f( )
c2k( )
centroidal_mean( )
choice_between_i( )
choice_between_d( )
choice_between_s( )
coherent( )
coherent_lindomain( )
coherent_linsweep( )
coherent_logdomain( )
coherent_logsweep( )
cmp3( )
compress_A_law( )
compress_mu_law( )
contraharmonic_mean( )
db2lin( )
db2np( )
db2pow( )
d2i( )
dec2bin( )
deg2rad( )
diode_Iv( )
diode_Ri( )
diode_Rv( )
diode_Vi( )
dirac( )
dirac2( )
ET12( )
expand_A_law( )
expand_mu_law( )
f2c( )
factorial( )
gaussian( )
geometric_mean( )
halton( )
hard_limiter( )
harmonic_mean( )
heronian_mean( )
GCD( )
i2d( )
isign( )
ispowerof( )
ispowerof2( )
ispowerof10( )
k2c( )
LCM( )
lin2db( )
lindomain( )
linsweep( )
logdomain( )
log_factorial( )
logn( )
logsweep( )
next_hailstone_number( )
np2db( )
parallel_R( )
parallel_C( )
parm_for_rand_uniform( )
parm_for_rand_normal( )
parm_for_rand_rayleigh( )
pow2db( )
P2t( )
kt( )
powerof( )
powerof2( )
powerof10( )
prompt_for_double( )
prompt_for_long( )
prompt_for_yes_no( )
QR_01( )
QR_01_x( )
QR_gaussian( )
QR_normal( )
QR_uniform( )
QR_uniform_x( )
rad2deg( )
rand_01( )
rand_01_x( )
rand_bernoulli( )
rand_binomial( )
rand_chisquare( )
rand_equilikely( )
rand_erlang( )
rand_exponential( )
rand_gaussian( )
rand_halfnormal( )
rand_geometric( )
rand_lognormal( )
rand_normal( )
rand_pascal( )
rand_poisson( )
rand_rayleigh( )
rand_uniform( )
rand_uniform_x( )
randomize_array( )
reldif( )
rnoise( )
root_mean_square( )
round_it( )
serie_R( )
serie_C( )
serie_L( )
sinc( )
smooth_limiter( )
soft_limiter( )
stuck_array( )
switch_i( )
switch_d( )
t2p( )
thermometric( )
vandercorput( )
vt( )

(in ‘/Simulate/NapaDos/Hdr/Function/..’)

ABS( )
CLIP( )
COS( )
DB2LIN( )
DB2POW( )
DEG2RAD( )
D2I( )
FSS( )
I2D( )
IO_MANAGER( )
ISDELAYED( )
ISEQUAL( )
ISEVEN( )
ISINSIDE( )
ISINTEGER( )
ISOPTION( )
ISNOTOPTION( )
ISODD( )
ISOUTSIDE( )
ISNOTEQUAL( )
ISNOTSMALL( )
ISSMALL( )
ISTIME( )
ISPOWEROF2( )
LENGTH( )
LIN2DB( )

LINDOMAIN( )
LINSWEEP( )
LOG( )
LOG10( )
LOGDOMAIN( )
LOGSWEEP( )
MIN( )
MAX( )
MODULO( )
NIS( )
PING( )
POW( )
POW10( )
POW2DB( )
POWEROF2( )
PS( )
RAD2DEG( )
RAND_01( )
RAND_01_X( )
ROOT( )
SEGMENT_CONDITION( )
SIGN( )
SIN( )
SQRT( )
STS( )
TIMER( )

A_CONSTANT( )
ARITHMETIC_MEAN( )
ARITHMETIC_GEOMETRIC_MEAN( )
C0( )
C2F( )
C2K( )
CENTROIDAL_MEAN( )
CONTRAHARMONIC_MEAN( )
D2I( )
EV( )
EPSILON0( )
F2C( )
G( )
GEOMETRIC_MEAN( )
H( )
HARMONIC_MEAN( )
HERONIAN_MEAN( )
I2D( )
ISPOWEROF( )
ISPOWEROF2( )
ISPOWEROF10( )
K( )
K2C( )
KT( )
LOGN( )
ME( )

MU_CONSTANT( )
MU0( )
POWEROF( )
POWEROF10( )
print_blank_line( )
print_line_of_chars( )
print_line_of_stars( )
print_newline( )
print_string( )
print_var( )
print_dvar_and_string( )
print_dvar( )
print_ivar_and_string( )
print_ivar( )
Q( )
RNOISE( )
RSWITCH( )
ROOT_MEAN_SQUARE( )
SET_STOP_REQUEST( )
STOP_REQUEST( )
SYSTEM_TIME( )
VT( )
Z0( )

(in '/Simulate/NapaDos/Hdr/...')

Built-In and Registered C Macro Functions

Built-In C Macro Constants and Variables

ABS_LOOP_INDEX
ABS_TIME
ANALOG_INI
ANTITHETIC
ASSERT_FLAG
CELLS_LIB
CODE
CREATED
DIGITAL_INI
ERROR_FLAG
FALSE
FS
FSL
GENERATORS_LIB
HEADERS_LIB
I_FORMAT
LOOP_INDEX
NAPA_JOB_ID
NAPA_VERSION
NAPA_WAYPOINT
NO
NUM_INITIAL
NUM_OF_SEGMENTS
NUM_OF_TIME_OUTPUTS
ORIGIN
PERIODIC
PLATFORM
RANDOM_SEED
REF_TIME
REL_LOOP_INDEX
REL_TIME
R_FORMAT
SEGMENT
SEPARATOR
S_FORMAT
SHORT_TITLE
SIM_RATE
SOURCE
STL
TERMINATE
TIME
TITLE
TRUE
USER
WALL_CLOCK
X_FORMAT
YES

Built-In C Constants

START = 1LL
STOP = 0LL

_pi_ = 3.141592653589793
_pi2_ = 1.570796326794897
_pi4_ = 0.7853981633974483
_pi8_ = 0.3926990816987242
_2pi_ = 6.283185307179586
_e_ = 2.718281828459045

_PI_ = 3.141592653589793239L
_PI2_ = 1.570796326794896619L
_PI4_ = 0.7853981633974483096L
_PI8_ = 0.3926990816987241548L
_2PI_ = 6.283185307179586477L
_E_ = 2.718281828459045235L

EPSILON = 2.0e-015

Some Useful Resources to Build User and Tool Functions

IO Manager ***Example:*** *(void) IO_MANAGER(OPENWRITE, &fp, "ffta", ".out", 2)*

```
(long long) IO_MANAGER(CLOSE,      &fp,  file_name, file_suffix,  function_id)
(long long) IO_MANAGER(DEBUG,      &fp,  file_name, file_suffix,  function_id)
(long long) IO_MANAGER(DELETE,     &fp,  file_name, file_suffix,  function_id)
(long long) IO_MANAGER(FREE,       &fp,  file_name, file_suffix,  function_id)
(long long) IO_MANAGER(OPENAPPEND, &fp,  file_name, file_suffix,  function_id)
(long long) IO_MANAGER(OPENREAD,   &fp,  file_name, file_suffix,  function_id)
(long long) IO_MANAGER(OPENWRITE,  &fp,  file_name, file_suffix,  function_id)
(long long) IO_MANAGER(QUERY,      &fp,  file_name, file_suffix,  function_id)
(long long) IO_MANAGER(REWIND,     &fp,  file_name, file_suffix,  function_id)
(long long) IO_MANAGER(REWRITE,    &fp,  file_name, file_suffix,  function_id)
```

Dynamic Memory Allocation Manager ***Example:*** *(void) DA_MANAGER(FREE, &arr, 0LL, id)*

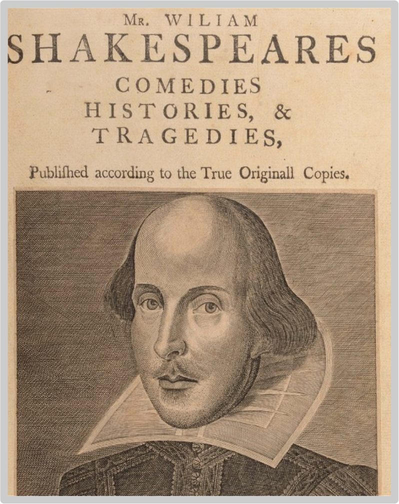
```
(long long) DA_MANAGER(ALLOCATE,   &array, number,  function_id)        HA/DA/FA/LA/IA/SA/CA/PA/GA
(long long) DA_MANAGER(FREE,       &array, number,  function_id)
(long long) DA_MANAGER(QUERY,      &array, number,  function_id)
(long long) DA_MANAGER(RESET,      &array, number,  function_id)
```

Sine and Cosine Table Manager ***Example:*** *(void) SC_MANAGER(FREE, &sc, 1000000LL, 3)*

```
(long long) SC_MANAGER(ALLOCATE,   &sintable, &costable, number,  function_id) SC/S/C
(long long) SC_MANAGER(FREE,       &sintable, &costable, number,  function_id)
(long long) SC_MANAGER(QUERY,      &sintable, &costable, number,  function_id)
```

The Last Words

" Come, come, good model is a good familiar creature if it be well used " [1]



[1] 309 Come, come, good wine is a good familiar creature
310 if it be well used"

William Shakespeare (1564-1616)
Othello, II. Iii

[NAPA] *Yves Leduc*
http://www.borogoves.eu/NAPA Release nnn *<<< download NAPA from the web*

http://www.borogoves.eu/NAPA Teaser.pdf
http://www.borogoves.eu/NAPA Card.pdf
http://www.borogoves.eu/NAPA Primer.pdf
http://www.borogoves.eu/NAPA Reference.pdf
http://www.borogoves.eu/MAC Card.pdf
http://www.borogoves.eu/MAC Reference.pdf

[SARC] *Jacques Mequin*
« Semi-Analytical Recursive Algorithms in Mixed Signals Simulations »
to be published

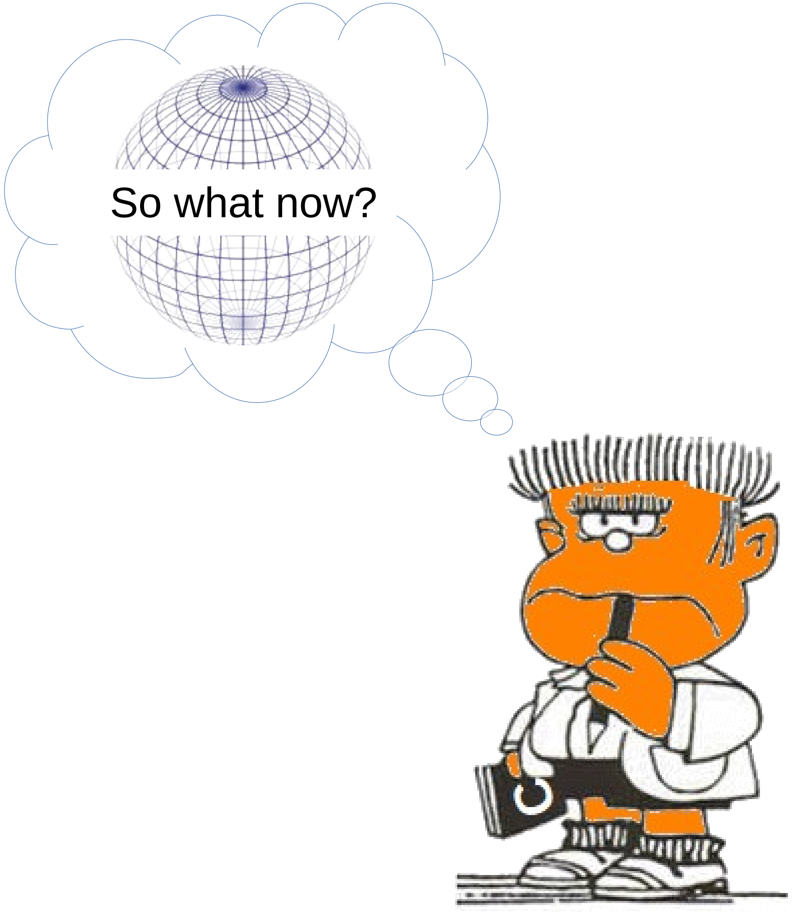
[ANSI C11] ISO/IEC 9899:2011
https://www.iso.org/standard/57853.html

[CRIMSON EDITOR]
http://www.crimsoneditor.com/

[GNUPLOT]
http://www.gnuplot.info/

[MINGW 64]
https://mingw-w64.org/

[WXMAXIMA and MAXIMA]
https://wxmaxima-developers.github.io/wxmaxima/



Email: yves.leduc@borogoves.eu

and free

An Efficient Solution to
Simulate Mixed-Signal Circuits in **C**

Questions ?

POLYTECH NICE-SOPHIA
POLYTECH'LAB
Université Nice Sophia Antipolis