1. **Theory behind closed loop stability analysis**

It all started with the below paper published by Michael Tian in 2001:

<https://kenkundert.com/docs/cd2001-01.pdf>

While I’m not 100% sure of all the details in this paper, the essence of it is show in Fig.7 below:

A diagram of a circuit

Description automatically generated

The paper introduces a double injection technique consisting of a voltage probe (vprb) and current probe (iprb) placed in the loop. To perform the technique we run 2 ac analysis. In the first, we set the ac value of vprb to 1 and the ac value of iprb to 0. In the second, the ac values of the 2 sources are reversed. Justin gives a nice explanation of this at the below link:

<http://education.ingenazure.com/ac-stability-analysis-ngspice/>

The loop gain characteristics are then quantified via the below equation (which is the main contribution of the paper):

A math equation with numbers

Description automatically generated with medium confidence

So what do A, B, C and D mean in (30)?

As already mentioned, the technique performs 2 ac analysis:

ac1: |vprb| = 1, |iprb| = 0

ac2: |vprb| = 0, |iprb| = 1

The parameters of (30) are thus defined as:

A = current through iprb at ac2

B = current through iprb at ac1

C = voltage at vprb at ac2

D = voltage at vprb at ac1

As per Fig.1, the currents flow into the -ve terminal of iprb which in spice means they are -ve. Therefore, for elegance, which will become apparent later, we define:

A\* = -A

B\* = -B

Now we are ready to break down the equation and make practical use of it. To do this, we refer to Frank Widermans excellent link below:

<https://groups.io/g/LTspice/message/4141>

In this link he shows the following:

Substuting in A\* and B\*, after some re-arranging, gives:

In Justins example he makes the following definitions:

A\*: let ip21 = ac2.i(vprobe1)

B\*: let ip11 = ac1.i(vprobe1)  
C: let vprb2 = ac2.probe

D: let vprb1 = ac1.probe

This allows him to define T as:

let av = 1/(1/(2\*(ip11\*vprb2-vprb1\*ip21)+vprb1+ip21)-1)

Placing the substitutions for A\*, B\*, C, D into (5) give the above expression exactly.

Personally I don’t bother re-defining things and just create the expression as follows:

let av = {1/(1/(2\*(ac1.i(v.xstb.Vi)\*ac2.v(xstb.x)-ac1.v(xstb.x)\*ac2.i(v.xstb.Vi))+ac1.v(xstb.x)+ac2.i(v.xstb.Vi))-1)}

This expression can be taken at face value for now as it will make more sense when we discuss how to implement this technique.

1. **Implementing closed loop stability analysis in ngspice / xschem**

Below compares Justins implementation of the technique with what was published. You will see they only differ by the buffer (Vprobe2) placed by Justin. Unsure exactly why this is but this is the version we are going to implement.

A diagram of a device

Description automatically generated A diagram of a diagram

Description automatically generated with medium confidence

Instead of having to place down voltage / current sources every time we want to perform a closed loop stability measurement, it is much more efficient to place the above in a subckt and instantiate that instead. This is exactly what is done in Cadence with their iprobe symbol, shown.

A screenshot of a computer

Description automatically generated

To create the subckt we are going to follow Roberts naming convention in the below post:

[ngspice / Feature Requests / #34 Request for Stability Analysis (sourceforge.net)](https://sourceforge.net/p/ngspice/feature-requests/34/)

This means renaming Justins nets / probes as follows:

A diagram of a mathematical equation

Description automatically generated with medium confidence

So we simply create an empty schematic in xschem with 2 pins (a and b) which we then make into a symbol, as shown below:

A screen shot of a computer

Description automatically generated

Note: See below link for how to create symbols from schematics in xschem:

<https://xschem.sourceforge.io/stefan/xschem_man/creating_symbols.html>

Next, descend into the symbol, press ‘q’ and enter the below text which defines the subckt:

A screenshot of a computer

Description automatically generated

Now, everytime you want to perform closed loop stability analysis, insert this probe and name it “xstb”.

At this point it is worth revisiting the below expression:

let av = {1/(1/(2\*(ac1.i(v.xstb.Vi)\*ac2.v(xstb.x)-ac1.v(xstb.x)\*ac2.i(v.xstb.Vi))+ac1.v(xstb.x)+ac2.i(v.xstb.Vi))-1)}

Now it can be understood as follows:

A\*: ac2.i(v.xstb.Vi) … probe current through Vi in xstb from the 2nd ac analysis

B\*: ac1.i(v.xstb.Vi) … probe current through Vi in xstb from the 1st ac analysis  
C: ac2.v(xstb.x) … probe voltage at node x in xstb from the 2nd ac analysis

D: ac1.v(xstb.x) … probe voltage at node x in xstb from the 1st ac analysis

By substituting the above into (5) (shown below for reference), you will arrive at my expression in ngspice:

So in ngspice you will place this expression but only after both ac analyses have been done. The first ac analysis is where we set the ac value of v.xstb.Vi to 1 and the ac value of i.xstb.Ii to 0, as shown below (note we save the results to raw file tb\_OTA\_1stage\_ac1.raw).

\*\* 2. AC ANALYSIS \*\*

alter i.xstb.Ii acmag=0

alter v.xstb.Vi acmag=1

ac dec 10 1 100G

remzerovec

write tb\_OTA\_1stage\_ac1.raw

The 2nd ac analysis is where we set the ac value of v.xstb.Vi to 0 and the ac value of i.xstb.Ii to 1, as shown below. It is after we have performed this 2nd ac analysis that we can use our Tian equation after which we write the results to raw file tb\_OTA\_1stage\_ac2.raw.

alter i.xstb.Ii acmag=1

alter v.xstb.Vi acmag=0

ac dec 10 1 10G

let av = {1/(1/(2\*(ac1.i(v.xstb.Vi)\*ac2.v(xstb.x)-ac1.v(xstb.x)\*ac2.i(v.xstb.Vi))+ac1.v(xstb.x)+ac2.i(v.xstb.Vi))-1)}

remzerovec

write tb\_OTA\_1stage\_ac2.raw

Below exemplifies the use of the probe in a circuit with the resulting code shown in the Appendix. Note: There is a lot going on in this code so I have highlighted in bold the sections relevant to this closed loop stability analysis.

A screenshot of a computer game

Description automatically generated

1. **Viewing results from the closed loop stability analysis in xschem**

As per section 2, the closed loop characteristics are stored as variable “av” saved in file tb\_OTA\_1stage\_ac2.raw. To view this data in xschem, open the waveviewer, create a plot and open tb\_OTA\_1stage\_ac2.raw as per below:

**A screenshot of a computer

Description automatically generated**

Note: For information on xschems waveviewer, the reader is referred to the following link:

<https://xschem.sourceforge.io/stefan/xschem_man/graphs.html>

In the waveviewer, double click and select “av”. In the below we wanted to plot the closed loop phase response and so inputted ph(av).

Note: To get horizonal cursors we inputted “ <y-evel>”. E.g. for a horizontale cursor placed at -45deg, input “ -45”. To place a vertical cursor press ‘a’. To place a 2nd vertical cursor, and make a differential measurement with the first, press ‘b’. Then to remove the cursors, simply place ‘a’ and ‘b’ again.

A screenshot of a computer

Description automatically generated

In the below we wanted to plot the closed loop magnitude response and so inputted “av db20()”.

A screenshot of a computer

Description automatically generated

1. **Creating expressions from the closed loop stability analysis in xschem / ngspice**

Ngspice is very good at dealing with measures. Below shows example measures related with determining the various characteristics of a loops response.

\*\* 3. MEASURES \*\*   
  
let n45\_rads = -45\*(pi/180)   
meas AC Av\_0 FIND vdb(av) AT=10   
echo --   
meas AC BW WHEN vp(av)=n45\_rads CROSS=1   
echo --   
meas AC Av\_BW FIND vdb(av) WHEN vp(av)=n45\_rads CROSS=1   
echo --   
meas AC ULGF WHEN vdb(av)=0   
echo --   
meas AC ULGF\_phi\_rads FIND vp(av) WHEN vdb(av)=0 CROSS=1   
let ULGF\_phi\_deg = ULGF\_phi\_rads\*(180/pi)   
print ULGF\_phi\_deg   
let PM = 180+ULGF\_phi\_deg   
print PM   
echo --   
\* putting in -170 since this is a 2nd-order system so doesnt ever achieve -180   
let n180\_rads = -170\*(pi/180)   
meas AC GM FIND vdb(av) WHEN vp(av)=n180\_rads CROSS=1

Anything behond this and you can search various measures in the forum and (or) post to the forum itself.

Below shows the ngspice output for these measures:

A computer screen shot of a program

Description automatically generated

There are many more expressions which can be done in ngspice that are documented in the manual and discussed in various posts of the forum.

Note you can create expressions in xschem. But this expression editor is intended to create expressions which reutrns a single waveform and not a single waveform number.

1. **Making the stb probe available for global usage in xschem**

To avoid having to copy the loopgainprobe schematic and symbol to the directory of every tb you are using it in, it is more efficient to place it in the global directory found at:

/usr/local/share/xschem/xschem\_library/devices

To do this, first you need to change ownership of this dir to yourself to allow you to copy files into it. This is done using the “chown” command (short for change ownership) as follows:

**sudo chown -R <username> /usr/local/share/xschem/xschem\_library/devices**

**To find your username (if you don’t already know it) just type whoami into a terminal. For me it is slice making the below command:**

**sudo chown -R slice /usr/local/share/xschem/xschem\_library/devices**

Now simply copy loopgainprobe.sch and loopgainprobe.sym into the dir as normal. Result will be as follows:

A screenshot of a computer screen

Description automatically generated

From now on insert the loopgainprobe.sym from this dir as per below:

A screenshot of a computer

Description automatically generated

1. **Example**

All files necessary to run the above closed loop stability analysis have been checked into the below location in github:

<https://github.com/SLICESemiconductor/OpenSourceTool_Examples/tree/main/Running_closed_loop_stability_analysis_in_ngspice_through_xschem>

1. **Appendix**

\*\* sch\_path: /home/slice/xschem/tb\_OTA\_1stage/Stefans\_eg/ota\_vloga/tb\_OTA\_1stage.sch

\*\*.subckt tb\_OTA\_1stage

Vvssa vssa GND 0

Vvdda vdda vssa xvdda

Vvinp vinp vssa xvinp

xota vdda vout vfb vinp ibias vssa OTA\_ideal\_order1\_elmod

C2 vout vssa {xCload} m=1

**xstb vfb vout loopgainprobe**

Iisnk vdda ibias xibias

\*\*\*\* begin user architecture code

\* Models

\* Note: I name my nmos models as nmos and pmos ones as pmosx (just messing a bit)

.model nmos nmos level=54 version=4.8.2

.model pmosx pmos level=54 version=4.8.2

\* Parameters

.param xvdda = 1.8

.param xvinp = 1

.param xibias = 2u

.param xCload = 0p

.param xgm\_dp = 1e-3

.param xRin = 1/xgm\_dp

.param xCout = 1p

.param xRout = 1e6

.model OTA\_vcvs OTA\_vcvs

\* vlogA instantiation

\*\* 1. DCOP analysis \*\*

\*\* must save the below for DCOP analysis to be back annotated onto the schematic

.option savecurrents

.save

+ @m.xota.m1[vgs]

+ @m.xota.m1[vth]

+ @m.xota.m1[gds]

+ @m.xota.m1[gm]

+ @m.xota.m1[id]

+ @m.xota2.m1[vgs]

+ @m.xota2.m1[vth]

+ @m.xota2.m1[gds]

+ @m.xota2.m1[gm]

+ @m.xota2.m1[id]

+ @m.xota.m2[vgs]

+ @m.xota.m2[vth]

+ @m.xota.m2[gds]

+ @m.xota.m2[gm]

+ @m.xota.m2[id]

+ @m.xota2.m2[vgs]

+ @m.xota2.m2[vth]

+ @m.xota2.m2[gds]

+ @m.xota2.m2[gm]

+ @m.xota2.m2[id]

+ @m.xota.m3[vgs]

+ @m.xota.m3[vth]

+ @m.xota.m3[gds]

+ @m.xota.m3[gm]

+ @m.xota.m3[id]

+ @m.xota2.m3[vgs]

+ @m.xota2.m3[vth]

+ @m.xota2.m3[gds]

+ @m.xota2.m3[gm]

+ @m.xota2.m3[id]

+ @m.xota.m4[vgs]

+ @m.xota.m4[vth]

+ @m.xota.m4[gds]

+ @m.xota.m4[gm]

+ @m.xota.m4[id]

+ @m.xota2.m4[vgs]

+ @m.xota2.m4[vth]

+ @m.xota2.m4[gds]

+ @m.xota2.m4[gm]

+ @m.xota2.m4[id]

+ @m.xota.m5[vgs]

+ @m.xota.m5[vth]

+ @m.xota.m5[gds]

+ @m.xota.m5[gm]

+ @m.xota.m5[id]

+ @m.xota2.m5[vgs]

+ @m.xota2.m5[vth]

+ @m.xota2.m5[gds]

+ @m.xota2.m5[gm]

+ @m.xota2.m5[id]

+ @m.xota.m6[vgs]

+ @m.xota.m6[vth]

+ @m.xota.m6[gds]

+ @m.xota.m6[gm]

+ @m.xota.m6[id]

+ @m.xota2.m6vgs]

+ @m.xota2.m6[vth]

+ @m.xota2.m6[gds]

+ @m.xota2.m6[gm]

+ @m.xota2.m6[id]

.control

pre\_osdi OTA\_vcvs.osdi

save all

op

write tb\_ota\_1stage\_op.raw

**\*\* 2. AC ANALYSIS \*\***

**alter i.xstb.Ii acmag=0**

**alter v.xstb.Vi acmag=1**

**ac dec 10 1 100G**

**remzerovec**

**write tb\_OTA\_1stage\_ac1.raw**

**alter i.xstb.Ii acmag=1**

**alter v.xstb.Vi acmag=0**

**ac dec 10 1 10G**

**\* use this line if you want the phase response to start at 180deg**

**\*let av = {1/(1-1/(2\*(ac1.i(v.xstb.Vi)\*ac2.v(xstb.x)-ac1.v(xstb.x)\*ac2.i(v.xstb.Vi))+ac1.v(xstb.x)+ac2.i(v.xstb.Vi)))}**

**\* use this line if you want the phase response to start at 0deg (more conventional and directly corresponds to Franks derivation)**

**let av = {1/(1/(2\*(ac1.i(v.xstb.Vi)\*ac2.v(xstb.x)-ac1.v(xstb.x)\*ac2.i(v.xstb.Vi))+ac1.v(xstb.x)+ac2.i(v.xstb.Vi))-1)}**

**remzerovec**

**write tb\_OTA\_1stage\_ac2.raw**

**\*ngspice plots**

**\*plot vdb(av) xlog**

**\*plot {(180/pi)\*vp(av)} xlog**

**\*plot 180\*cph(av)/pi**

**\*\* 3. MEASURES \*\***

**let n45\_rads = -45\*(pi/180)**

**meas AC Av\_0 FIND vdb(av) AT=10**

**echo --**

**meas AC BW WHEN vp(av)=n45\_rads CROSS=1**

**echo --**

**meas AC Av\_BW FIND vdb(av) WHEN vp(av)=n45\_rads CROSS=1**

**echo --**

**meas AC ULGF WHEN vdb(av)=0**

**echo --**

**meas AC ULGF\_phi\_rads FIND vp(av) WHEN vdb(av)=0 CROSS=1**

**let ULGF\_phi\_deg = ULGF\_phi\_rads\*(180/pi)**

**print ULGF\_phi\_deg**

**let PM = 180+ULGF\_phi\_deg**

**print PM**

**echo --**

**\* putting in -170 since this is a 2nd-order system so doesnt ever achieve -180**

**let n180\_rads = -170\*(pi/180)**

**meas AC GM FIND vdb(av) WHEN vp(av)=n180\_rads CROSS=1**

setplot

.endc

\*\*\*\* end user architecture code

\*\*.ends

**\* expanding symbol: loopgainprobe.sym # of pins=2**

**\*\* sym\_path: /usr/local/share/xschem/xschem\_library/devices/loopgainprobe.sym**

**\*\* sch\_path: /usr/local/share/xschem/xschem\_library/devices/loopgainprobe.sch**

**.subckt loopgainprobe a b**

**\*.iopin b**

**\*.iopin a**

**\*\*\*\* begin user architecture code**

**Ii 0 x DC 0 AC 0**

**Vi x a DC 0 AC 1**

**Vnodebuffer b x 0**

**\*\*\*\* end user architecture code**

.ends

\* expanding symbol: OTA\_ideal\_order1\_elmod.sym # of pins=6

\*\* sym\_path: /home/slice/xschem/tb\_OTA\_1stage/Stefans\_eg/ota\_vloga/OTA.sym

\*\* sch\_path: /home/slice/xschem/tb\_OTA\_1stage/Stefans\_eg/ota\_vloga/OTA\_ideal\_order1\_elmod.sch

.subckt OTA\_ideal\_order1\_elmod vdda vout vinn vinp ibias\_1u vssa

\*.ipin vssa

\*.ipin vinp

\*.ipin ibias\_1u

\*.ipin vinn

\*.ipin vdda

\*.opin vout

E1 net1 vssa vinp vinn 1

R1 net2 vssa {xRin} m=1

F1 vdda net3 Vidp 1

Vidp net1 net2 0

R2 vout vssa {xRout} m=1

C2 vout vssa {xCout} m=1

Viout net3 vout 0

R3 ibias\_1u vssa 1MEG m=1

.ends

.GLOBAL GND

.end