

Xschem has the capability to display both static and transient DCOPs. By static DCOPs I mean the standard form commonly used. By transient DCOPs I mean dragging a cursor across a transient waveform and seeing the DCOPs change as a function of time. This is quite unique to xschem (I havent seen it in any of the other “big 3”) which in my opinion is a really effective debug feature. I will first start by showing how to perform static DCOPs before moving onto transient ones.

## 1. Static DCOPs:

### 1.1 Save DCOPs in ngspice:

Before your .control statement, setup ngspice to save the DCOPs. For example, the below saves the “vth” of transistor “m1” inside instance “xota” and the “gm” of transistor “m2” inside instance “xota2”.

```
** 1. DCOP analysis **
```

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

```
.option savecurrents
```

```
.save
```

```
+ @m.xota.m1[vth]
```

```
+ @m.xota.m2[gm]
```

As you can see, the syntax is: @m.<instance\_name>.<device\_name>[DCOP]. All the usual DCOPs can be saved where at present those which I save are shown below:

```
+ @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.xota2.m5[vdsat]
```

The above saves all the required DCOPs for device “m5” in instance “xota2”.

Then inside your .control statement, perform an OP analysis and write the results to a raw file in binary (which it will be by default). See below example where I write my op results to file “tb\_ota\_1stage\_op.raw”.

```
op
```

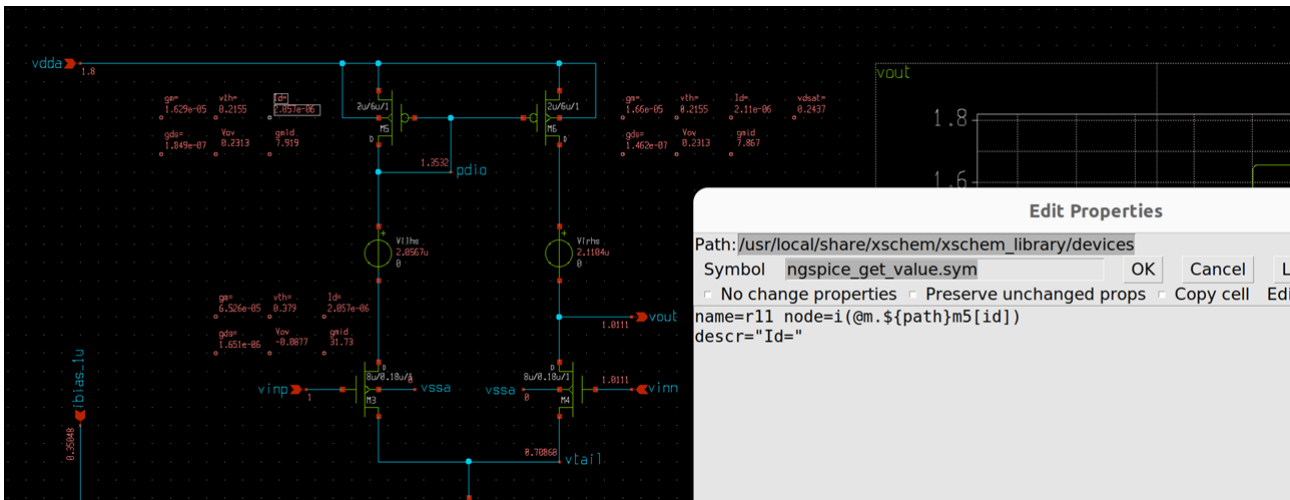
```
write tb_ota_1stage_op.raw
```

### 1.2 Setup xschem DCOP schematic:

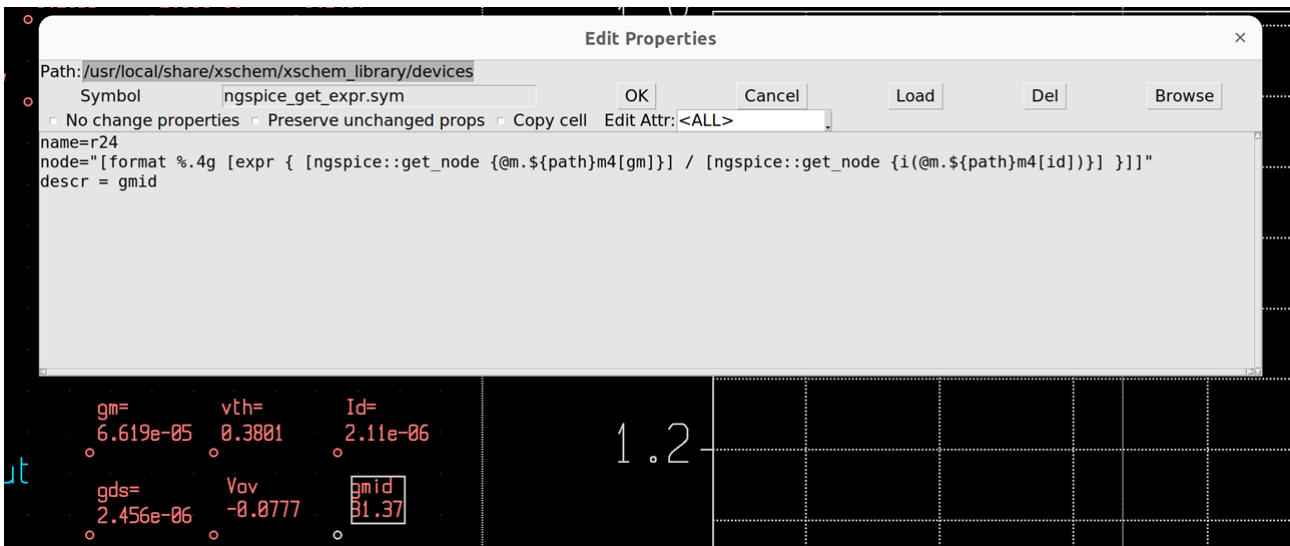
xschem deals with DCOPs slightly differentially than other tools in that it uses specific instances, callend “ngspice\_get\_value.sym” to retrieve the required DCOP value and display it on the schematic. As these symbols must be explicitly added by the user, it makes sense to create a schematic specific for DCOPs which will have all these symbols added. In addition, a DCOP specific schematic also lends itself to transient DCOPs better.

So the first thing you do is copy your schematic and save as schematic\_DCOP.

In this schematic place as many “ngspice\_get\_value.sym” instances as the number of DCOPs you require.



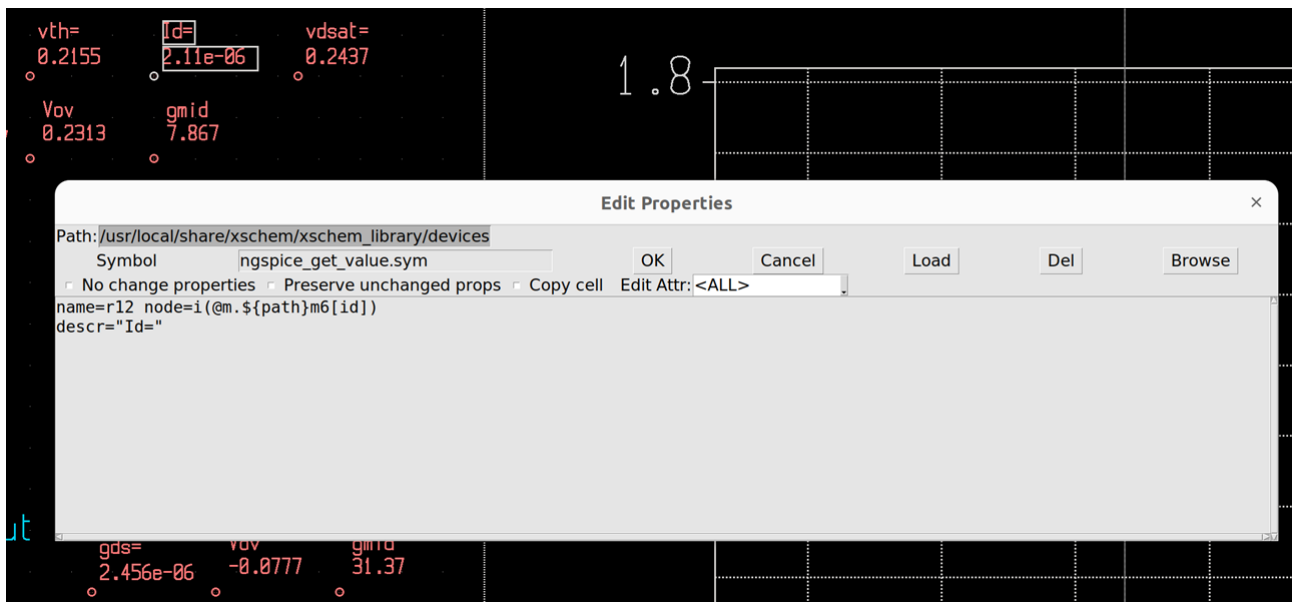
First thing you will notice about the raw code inside these instances is they are very specific .tcl commands. E.g. see below for measuring the gm/id of a device:



In a nutshell, you don’t want to be having to play around too much with this code. Therefore my advice is to take all the DCOP symbols used in the example, whose git hub check in details are given in section 3.

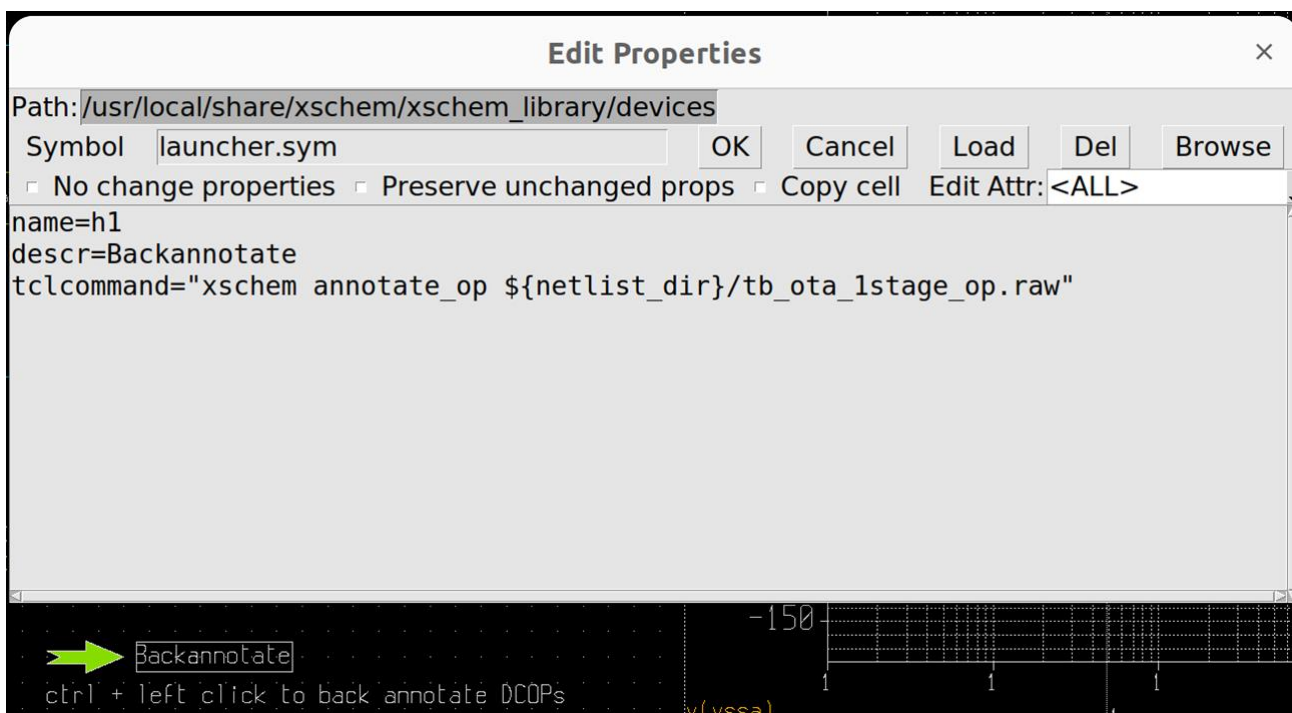
Then you simply copy and paste them as needed where the only thing you would need to change is the device name.

E.g. In the following screenshot, “id” for “m6” is displayed. Say I have “m32” somewhere else whose “id” I want to display. I simply copy this “ngspice\_get\_value.sym”, paste it beside “m32” and change “m6” to “m32” in the tcl command (but nothing else). Then assuming you have the line `+ @m.<instance_name>.m32[id]` contained in your ngspice file, the drain current through m32 will be displayed beside it.



### 1.3 Setup xschem tb to load DCOP results:

To load the DCOP results, you need to include a “launcher” in the tb toplevel. Below shows the launcher.sym which is labelled “Backannotate” to load in DCOP results written into raw file “tb\_ota\_1stage\_op.raw”.



I generally keep “Backannotate” as the launcher description and even include the text file below to avoid me forgetting to have to ctrl + left click to load in the results.

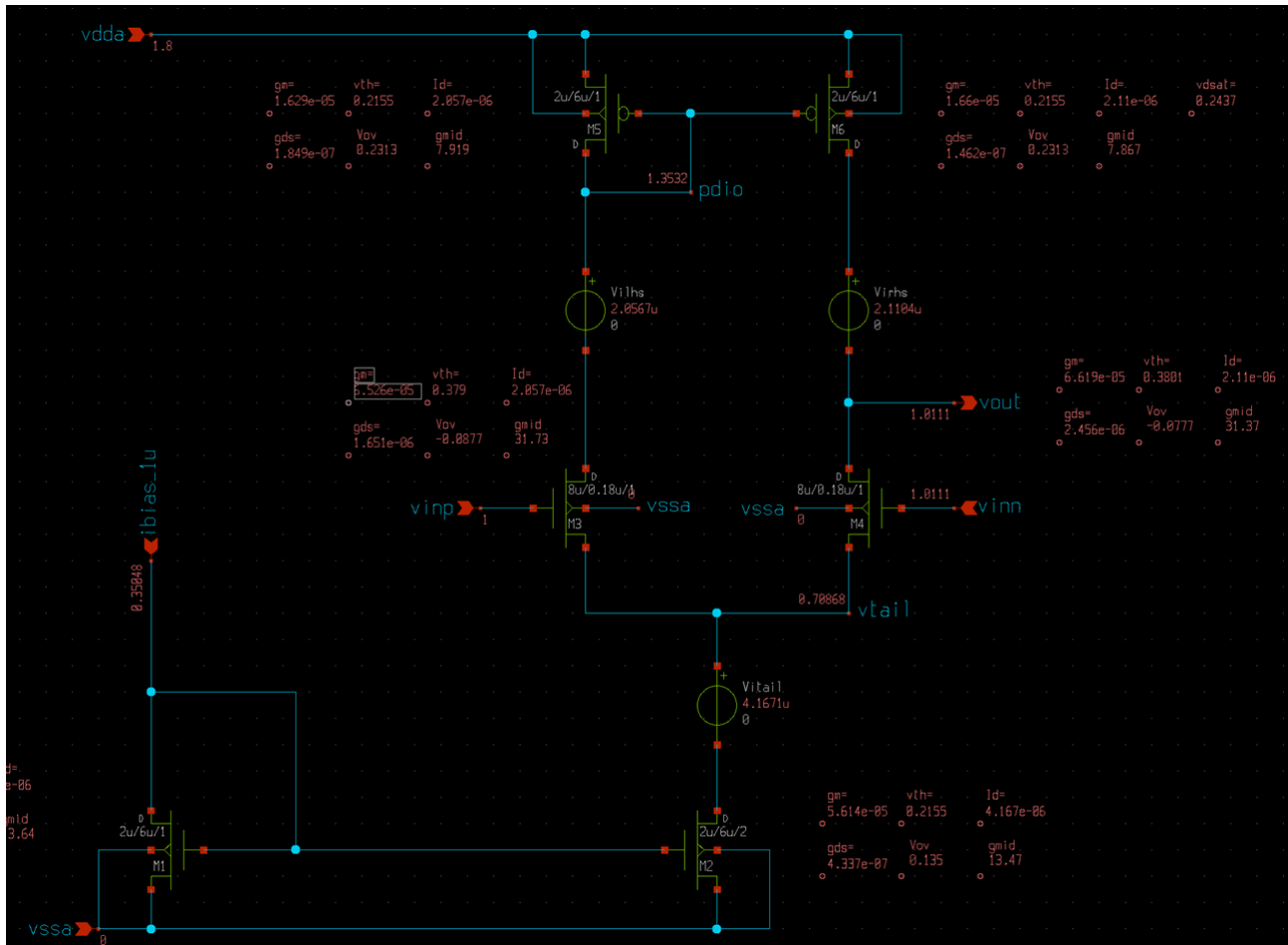
Obviously when using the launcher, take care to ensure it is calling the correct raw file. For example, the above launcher calls the raw file “tb\_ota\_1stage\_op.raw” since that is where I wrote my OP values to in my ngspice file, with the below code:

```

op
write tb_ota_1stage_op.raw
  
```

## 1.4 View DCOP results in xschem:

As per section 1.3, simply ctr+left click on the launcher and you should see something like the below:



Note how M6 was the only device I saved a vdsat for as I was investigating saving vdsat at the time.

You have a lot of flexibility here in that you can choose which DCOPs to display and which not to display (just make sure your ngspice has saved the DCOP of interest, with the shown code). But I like this approach as it gives you total control into which/how and in what order you want your DCOPs displayed.

## 2. Transient DCOPs:

### 2.1 Save DCOPs in ngspice:

Assuming you are already set up to save static DCOPs as described in section 1.1, when performing your transient sim simply use the “set appendwrite” command to append the DCOPs to each time step in your transient sims.

Below exemplifies this where I am appending transient DCOP values to my original file specified to save static DCOPs in (“tb\_ota\_1stage\_op.raw”).

## **\*\* 2. TRAN ANALYSIS \*\***

```
remzerovec
set appendwrite
tran 1n $&xtsim
write tb_ota_1stage_op.raw
unset appendwrite
```

Note the use of “unset appendwrite” to stop appending any other data to this file.

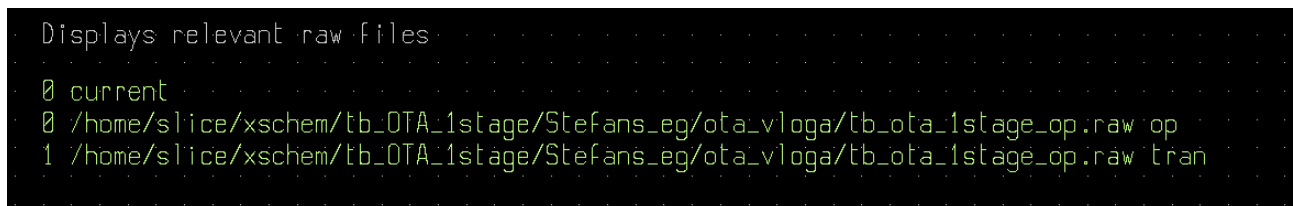
### **2.2 Setup xschem tb to load DCOP results:**

At this point, think of your raw file as consistent of 2 sets of DCOP values – static and transient ones. You therefore need to be able to select which set to use.

To do this you first need to be able to display both options of the raw file to select. This requires you to use, what I call the “xschem raw info” block. This is merely a text file with the below .tcl code in it:

```
tclval([xschem raw info])
```

After running the sim and loading the relevant data (explained later), this will display both sections of the raw file.

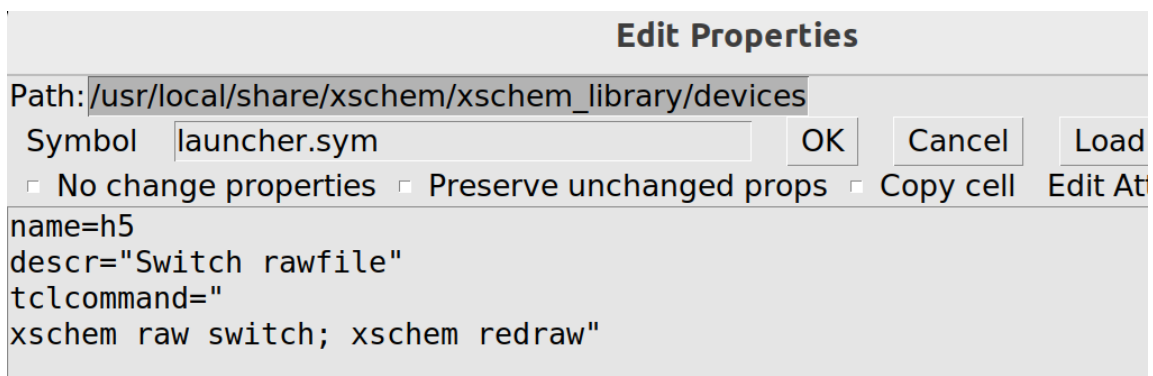


```
Displays relevant raw files:
0 current
0 /home/slice/xschem/tb_OTA_1stage/Stefans_eg/ota_vloga/tb_ota_1stage_op.raw op
1 /home/slice/xschem/tb_OTA_1stage/Stefans_eg/ota_vloga/tb_ota_1stage_op.raw tran
```

Again, to avoid having to deal with .tcl code, it is best just to copy this block from the example, whose git hub check in details are given in section 3.

The above shows the real purpose of the “xschem raw info” block – to display the index of the current simulation. In the above it was ‘0’. With ‘0’ set as current, using the backannotate launcher described in section 1.3, a left click + ctrl operation would display the static DCOPs, since index ‘0’ corresponds to such data in the raw file.

If you want to display transient DCOPs you need to change this index to ‘1’. To do this, you need the below launcher.



Edit Properties			
Path:	/usr/local/share/xschem/xschem_library/devices		
Symbol	launcher.sym	OK	Cancel
<input type="checkbox"/> No change properties <input type="checkbox"/> Preserve unchanged props <input type="checkbox"/> Copy cell <input type="checkbox"/> Edit At			
name=h5			
descr="Switch rawfile"			
tclcommand="			
xschem raw switch; xschem redraw"			

The nice thing about this launcher is you don't have to edit any .tcl code (unlike the backannotate launcher where you need to edit the raw file name). So again to avoid getting involved with .tcl code, I advise to simply copy this launcher from the example, whose git hub check in details are given in section 3.

Together with the xschem raw info text field, the below should now appear in your tb:



As this launcher is simply to change the index of the current simulation in the “xschem raw info” block, you simply need to ctrl + left click on it to toggle the index value between 0 and 1. In the above screen shot I have ctrl + left clicked once to change the index to ‘1’.

Next you need to insert a graph which is going to be used to display the transient waveform, as shown below:

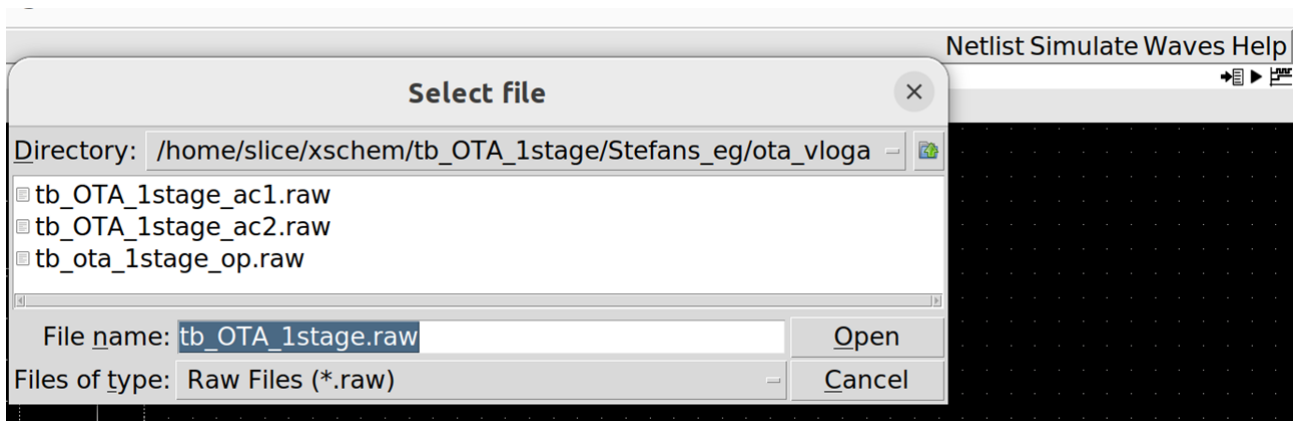


As shown above you need to specify the correct raw file which of course is the one you appended the DCOP data in, in section 2.1. In addition to this you need to add “\$netlist\_dir/” before the raw file name (*honestly I’m a little unsure why you need to do this but it works when I do, as per Stefans guidance*).

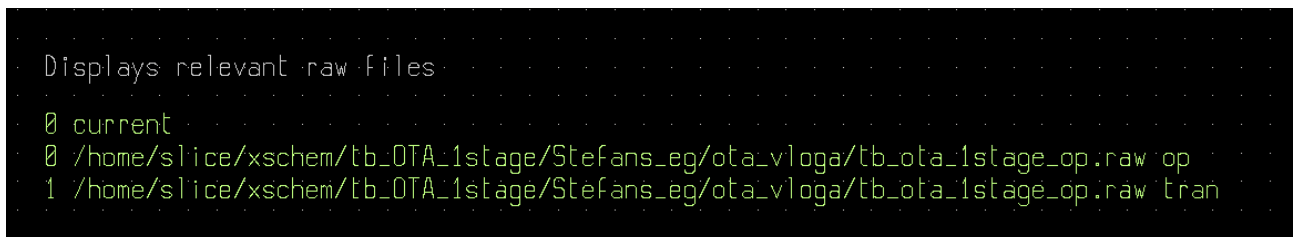
Now that you have appended data in ngspice, include the suitable launchers / text files / graph in xschem, you are ready to view the DCOP results in xschem.

### 2.3 View DCOP results in xschem:

Start by netlisting / simulating. When complete, load in the correct raw file using Waves/**Op**/**<raw\_filename>**. I have highlighted “Op” as it is important you load in the correct version of the waveform which is done by selecting “Op”. Once you do this, a pop up will appear asking you to select the raw file. Obviously this needs to be the raw file set up in section 2.1. In the below example, we can see we have 3 raw files – 2 for AC analysis (stb analysis) and 1 containing the transient DCOP values.



Once you select open the following will appear in the xschem raw info block:

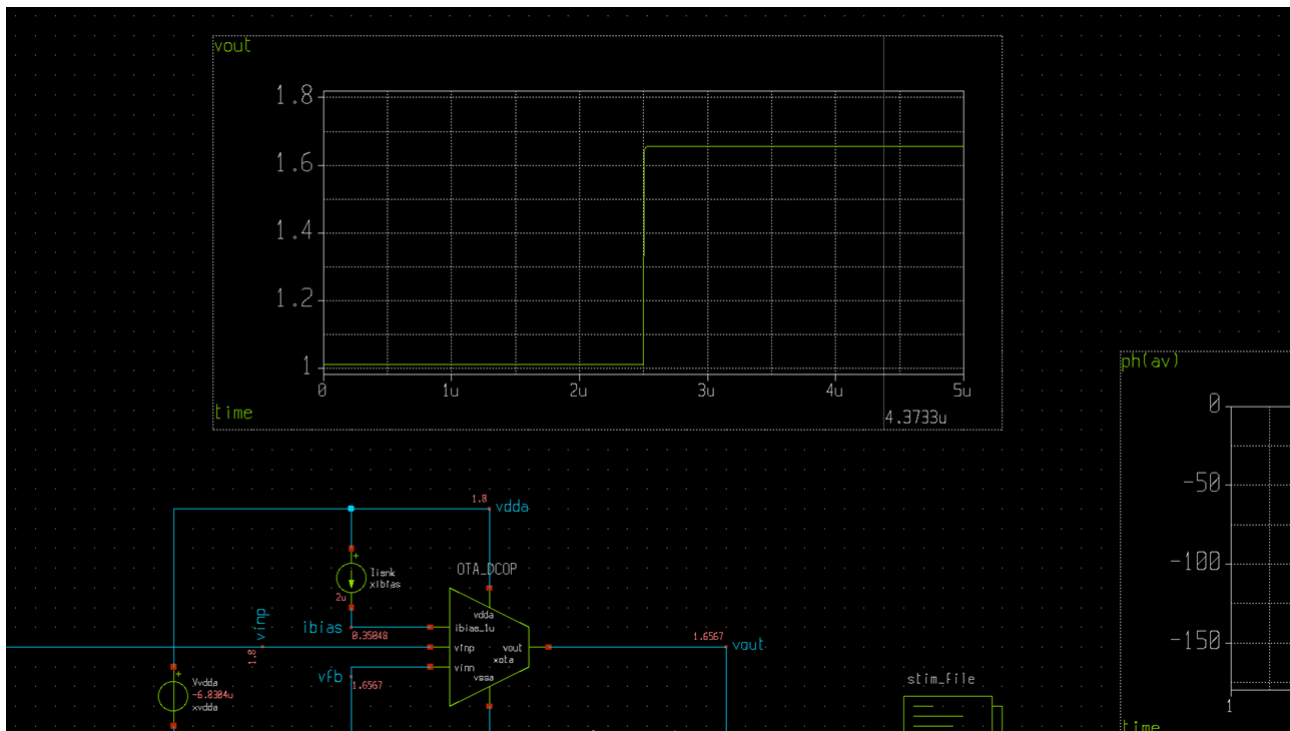


By default the static DCOP will be set as the current index. It is good practice to leave this selected at the start and ctrl + left click the back annotate launcher to view the static DCOPs, as shown below:

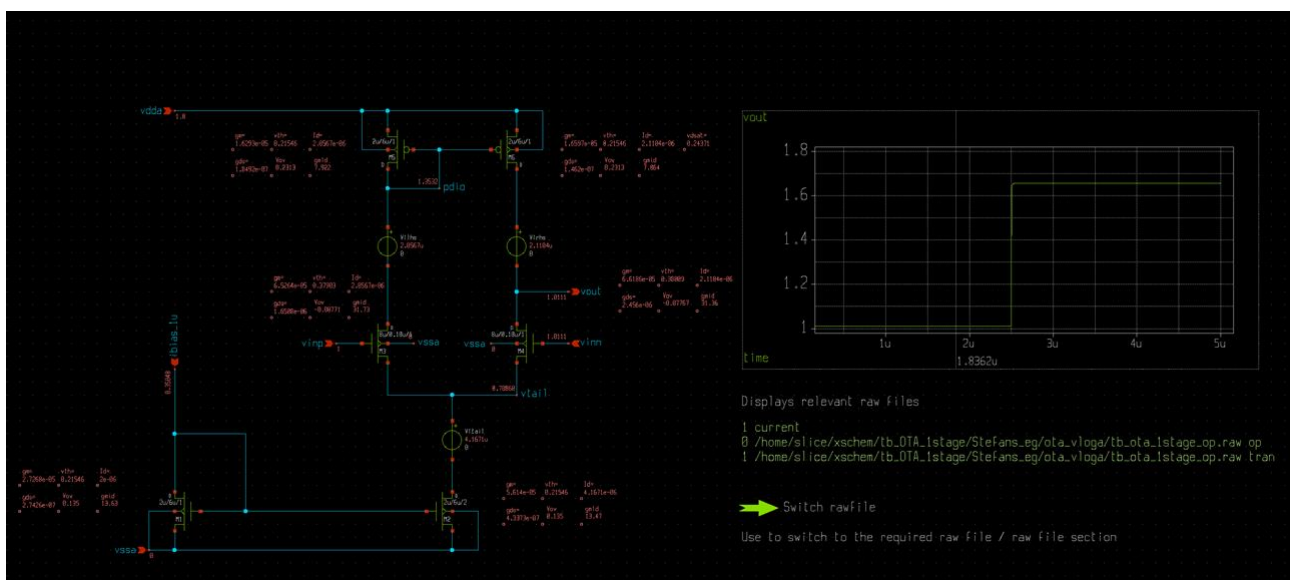




Click inside the graph and select 'b' to display a cursor. Now simply drag the cursor back and forth to see the DCOPs automatically update.



The only issue here is that since our OTA is inside a symbol, the only DCOPs we are seeing changing are those of the top level transient voltages. As we are more interested in viewing the DCOPs of the amplifier, simply copy the xschem raw info / xschem raw switch and transient graph into the schematic\_DCOP version of the sub-block you want to analyse. This should look like the below:



First view the static DCOPs by setting index to '0'. Once happy with those, using xschem raw switch, change the index to '1' and drag the 'b' cursor back and forth along the transient plot to see the DCOPs update in real time.


This is exactly what we want as it makes for a very powerful debug tool. Suppose something strange is happening at 3us in your sim. By simply running a DCOP and dragging the cursor to 3us you can see exactly what is happening. Other vendors of course can do this but you need to set up “AC at tran” which can only display DCOPs at discrete time points. The above lets you move about which in my opinion leads to a quicker debug.



### 3. Example:










All relevant files to perform static and transient DCOP analysis in ngspice / xschem are checked into the below location in github:

[https://github.com/SLICESemiconductor/OpenSourceTool\\_Examples/upload/main/Analysing\\_DCOPs\\_in\\_ngspice\\_xschem](https://github.com/SLICESemiconductor/OpenSourceTool_Examples/upload/main/Analysing_DCOPs_in_ngspice_xschem)

Note: All the pre-ceeding example screensnshots have been taken from these files.

OpenSourceTool\_Examples / Analysing\_DCOPs\_in\_ngspice\_xschem / 

 SLICESemiconductor Add files via upload e59b15d · now  History

Name	Last commit message	Last commit date
 ..		
 OTA.sch	Add files via upload	now
 OTA.sym	Add files via upload	now
 OTA_DCOP.sch	Add files via upload	now
 README	Update README	4 minutes ago
 loopgainprobe.sch	Add files via upload	now
 loopgainprobe.sym	Add files via upload	now
 tb_OTA_1stage.sch	Add files via upload	now
 xschemrc	Add files via upload	now

README 