Xyce® circuit simulator on a Raspberry Pi

Xyce is a state-of-the-art massively parallel circuit simulator that also runs in serial mode, and there is an open source version developed by Sandia National Laboratories, supported by the Department of Energy. Xyce is a fully modern, open source version of SPICE, the ubiquitous circuit simulation code. Now that the faster model 4 B is out, I thought it would be interesting to port the Xyce circuit simulator code to a Raspberry Pi. It works fine on other versions of Linux, so I didn't think it would be an issue, though there are challenges associated with the limited memory on a Pi. Here is the process to get it built on a Pi, recognize that this will take about a half day even on a Model 4B (2 GB). These instructions are for Version 7.2 of Xyce, as the specific Trilinos library needed changes with version, but any updates should be well explained in the Xyce build instructions. Be sure you are using a 16 GB or larger card or an external drive (strongly suggested). I use a solid state USB drive with 64 GB of storage. This build has also been checked with the Buster version of Raspbian on a Model 4B/8GB with the beta 64 bit kernel.

Starting with the latest full Buster distribution, first, customize the system with raspi-config, set up your wireless, and then update the system:

```
$ sudo apt-get update
$ sudo apt-get upgrade -y
```

Now install some needed non-standard utilities:

```
$sudo apt-get install gfortran cmake bison flex libsuitesparse-dev libtool python-scipy git
```

We need MIT’s fftw fast Fourier transform library. Use the Chrome browser to do all the downloading. Go to http://www.fftw.org/download.html and grab the 3.3.8 release via tar.gz. This will put the file in your Downloads folder which won't exist unless you used the browser for a download at least once. (The github version requires a lot of additional software.) You can navigate to the Downloads directory in a terminal window and unzip and extract it by:

```
$ cd ~/Downloads
$ tar xzf fftw-3.3.8.tar.gz
```

Now build and install fftw which only takes about 5 minutes:

```
$ cd fftw-3.3.8
$ ./configure
$ make
$ sudo make install
```

Next we need the Trilinos libraries, a very powerful set of serial and parallel solvers. Go to https://xyce.sandia.gov/ and log in or create an account (free). Navigate to the Download section and open the Building Guide link in the second paragraph. Scroll down a bit to Build Trilinos from source code and install it, and click on that link. There is a github link to the appropriate source code, version 12.12.1, so you can download it (github may or may not have easy access to older versions of the code). You want this particular version for Xyce 7.2 as later versions aren’t guaranteed to work with Xyce, nor are earlier version. This is a big file and it will take a few minutes to download, be patient. Go back to the download section and download the files containing the documentation Xyce_Docs-

7.2.tar.gz, they will be placed in the Downloads directory. This is a moderate download and will be quick.

Now to unzip the file Trilinos files.

$ cd ~/Downloads
$ tar xzf Trilinos-trilinos-release-12-12-1.tar.gz

The last command places the source code in ~/Downloads/Trilinos-trilinos-release-12-12-1, but we want to do an “out of source build”, so we create a directory to do this.

$ cd ~
$ mkdir Trilinos12.12
$ cd Trilinos12.12

Xyce only uses some of Trilinos, and their build page has a suggested reconfigure file. I cut and pasted the original into an editor window from the https://xyce.sandia.gov/documentation/BuildingGuide.html page. The SRCDIR line will need updating if you cut and paste from the Building Guide. Here is the code for a serial build for Linux that works for the Raspberry Pi that should be placed in a file named reconfigure (or copy it from github at https://github.com/wpballa/Xyce):

$ nano reconfigure

```
#!/bin/bash
SRCDIR=$HOME/Downloads/Trilinos-trilinos-release-12-12-1
ARCHDIR=$HOME/XyceLibs/Serial12.12.1
FLAGS="-O3 -fPIC"
cmake \
-G "Unix Makefiles" \
-DCMAKE_C_COMPILER=gcc \
-DCMAKE_CXX_COMPILER=g++ \
-DCMAKE_Fortran_COMPILER=gfortran \
-DCMAKE_CXX_FLAGS="$FLAGS" \
-DCMAKE_C_FLAGS="$FLAGS" \
-DCMAKE_Fortran_FLAGS="$FLAGS" \
-DCMAKE_INSTALL_PREFIX=$ARCHDIR \
-DCMAKE_MAKE_PROGRAM="make" \
-Dtrilinos_ENABLE_CXX11=ON \
-DTrilinos_ENABLE_NOX=ON \
-DTrilinos_ENABLE_LOCA=ON \
-DTrilinos_ENABLE_EpetraExt=ON \
-DEpetraExt_BUILD_BTF=ON \
-DEpetraExt_BUILD_EXPERIMENTAL=ON \
-DEpetraExt_BUILD_GRAPH_REORDERINGS=ON \
-DTrilinos_ENABLE_TrilinosCouplings=ON \
-DTrilinos_ENABLE_Ifpack=ON \
-DTrilinos_ENABLE_Isorropia=ON \
-DTrilinos_ENABLE_AztecOO=ON \
-DTrilinos_ENABLE_Belos=ON \
-DTrilinos_ENABLE_Teuchos=ON \
-DTeuchos_ENABLE_COMPLEX=ON \
-DTrilinos_ENABLE_Amesos=ON \
-DAmesos_ENABLE_KLU=ON \
-DAmesos_ENABLE_UMFPACK=ON \
```

```
-DTrilinos_ENABLE_Sacado=ON \
-DTrilinos_ENABLE_ALL_OPTIONAL_PACKAGES=OFF \
-DTPL_ENABLE_AMD=ON \
-DAMD_LIBRARY_DIRS="/usr/lib" \
-DTPL_AMD_INCLUDE_DIRS="/usr/include/suitesparse" \
-DTPL_ENABLE_UMFPACK=ON \
-DUMFPACK_LIBRARY_DIRS="/usr/lib" \
-DTPL_UMFPACK_INCLUDE_DIRS="/usr/include/suitesparse" \
-DTPL_ENABLE_BLAS=ON \
-DTPL_ENABLE_LAPACK=ON \
$SRCDIR
```

Once you have saved this file, do the following to make the file executable and then execute it:

$ chmod u+x reconfigure
$ ./reconfigure

If the build complains, it probably means you have some difference in the source code directory name (t for T, - for _, or the like), so fix the reconfigure file, resave it, and rerun it.

Next is to actually build the code, this takes around 45 minutes on a Raspberry Pi Model 4B, so start the make and do something else for a while. The make install is pretty quick once make is done.

$ make
$ sudo make install

Now Trilinos is all set so we could download Xyce from the https://xyce.sandia.gov/downloads/sign-in.html page. BUT, the Xyce team recommends using the github version. Still you need to go to the Sandia page to download the documentation which we did earlier. *Next we need to get the source code from GitHub.* Then put the code in a temporary directory so we can perform the build. We will also unpack the documentation and get the test suite.

$ cd ~
$ mkdir XyceTemp
$ cd XyceTemp
$ git clone https://github.com/Xyce/Xyce.git
$ cd Xyce
$ ./bootstrap
$ cd ~
$ mkdir Xyce7.2
$ cd Xyce7.2
$ tar xzf ~/Downloads/Xyce_Docs-7.2.tar.gz
$ git clone https://github.com/Xyce/Xyce_Regression.git

I made the following a file to be easier to edit as versions change, but the line to execute is:

$ nano config

```
#!/bin/bash
$HOME/XyceTemp/Xyce/configure \
ARCHDIR=$HOME/XyceLibs/Serial12.12.1 \
CPPFLAGS="-I/usr/include/suitesparse -I/usr/local/include" \
```

```
CXXFLAGS="-O3 -std=c++11" \
ADMS_CXXFLAGS="-O3 -std=c++11" \
LDFLAGS="-L/usr/local/lib" \
CFLAGS="-O3"
```

Note that the options use the letter O after the dash and then a number (3) to set optimization levels. We no longer have to turn off optimization on the complicated ADMS models to speed the compile with the latest code improvements. Make the config file executable and then execute it with:

$ chmod u+x config
$ ./config

Now we compile and install, again this takes significant time, about an hour.

$ make
$ sudo make install

Finally we want to test things with the regression suite. We haven't installed the library modules, so we turn off those checks with the –taglist option.

$ cd ~/Xyce7.2/Xyce_Regression/TestScripts
$ ./run_xyce_regression --taglist -library

Again, the regression suite takes a while but should process with no failures or warnings on a 64 bit kernel, and some warnings and failures on a 32 bit kernel. The issue seems to be around the precision for some of the tests. The output files will be in the ~/Xyce7.2/Xyce_Regression/Results directory.

I use LibreOffice Calc, a spreadsheet program to view .prn file output as a graph. You may need to install LibreOffice to use it, but it is a wonderful program to have and is installed by default with Buster.

My favorite test circuit file involves the OP27 operational amplifier…

$ nano OP27.cir

```
* OP27
*--------------------------------------------------------------------------------
*
*--------------------------------------------------------------------------------

.PRINT TRAN DELIMITER=TAB V(101) V(Vout)
.TRAN 1u 10m

*** Capacitors ***
*V(+)            3
*V(-)            5
*IN(+)           1
*IN(-)           2
*OUT             53
*Vos Trim1       67 (not used-leave floating)
*Vos Trim2       68 (not used-leave floating)

*** Capacitor Data ***
```

```
C_C1 30 63 79.6p
C_C10FB Vout 2 5.3n IC=0
C_C11IN 100 0 5.5n IC=-5
C_C2 3 13 159p
C_C3 41 64 104p
C_C4 30 58 21.9p

*** Input components ***
C_C5INP 1 0 147n
C_C6P1 3 0 47u
C_C7P2 3 0 10n
C_C8P3 5 0 47u
C_C9P4 5 0 10n

*** Transistors ***
Q_Q1 1 1 2  QNPN 1.7453
Q_Q10 5 10 9  QVPNP 1.4492
Q_Q11 10 11 12  QNPN 0.6283
Q_Q12a 13 2 11  QNPN 1.7535
Q_Q12b 13 2 11  QNPN 1.7535
Q_Q13a 14 0 11  QNPN 1.7535
Q_Q13b 14 0 11  QNPN 1.7535
Q_Q14 5 11 12  QVPNP 1.2262
Q_Q15 5 11 15  QVPNP 1.2262
Q_Q16 16 12 15  QNPN 1.7535
Q_Q17 17 12 15  QNPN 1.7535
Q_Q18 3 14 17  QNPN 0.7757
Q_Q19 3 13 16  QNPN 0.7757
Q_Q2 2 2 1  QNPN 1.7453
Q_Q20 11 18 20  QNPN 2.4241
Q_Q21 15 18 19  QNPN 1.212
Q_Q22 21 18 24  QNPN 1.212
Q_Q23 22 18 25  QNPN 0.6283
Q_Q24 23 18 26  QNPN 2.5133
Q_Q25 29 17 27  QLPNP 1.1111
Q_Q26 30 16 28  QLPNP 1.1111
Q_Q27 30 32 33  QNPN 0.6283
Q_Q28 29 32 34  QNPN 0.6283
Q_Q29 3 29 32  QNPN 0.6283
Q_Q3 8 4 3  QLPNP 1.1111
Q_Q30 39 38 35  QLPNP 1.6799
Q_Q31 40 38 36  QLPNP 1.2968
Q_Q32 41 38 37  QLPNP 2.0631
Q_Q33 31 42 39  QLPNP 1.875
Q_Q34 42 42 39  QLPNP 416.7m
Q_Q35 21 21 3  QLPNP 0.5787
Q_Q36 43 21 3  QLPNP 0.5787
Q_Q37 43 21 3  QLPNP 0.5787
Q_Q38 43 21 3  QLPNP 0.5787
Q_Q39 3 22 18  QNPN 496.4m
Q_Q4 7 4 3  QLPNP 1.1111
Q_Q40 4 44 5  QNPN 0.6283
Q_Q41 45 45 44  QNPN 0.6283
Q_Q42 23 23 46  QLPNP 1.1111
Q_Q43 22 23 47  QLPNP 1.1111
Q_Q44 3 41 48  QNPN 3.2593
Q_Q45 40 48 49  QLPNP 1.1111
Q_Q46 5 48 50  QVPNP 49.1598
Q_Q47 51 48 50  QNPN 0.6283
```

```
Q_Q48 48 40 52  QNPN 1.7453
Q_Q49 40 40 5  QNPN 0.6283
Q_Q5 6 4 3  QLPNP 1.1111
Q_Q50 55 54 Vout  QNPN 0.6283
Q_Q51 3 55 54  QNPN 6.4198
Q_Q52 56 50 Vout  QLPNP 1.1111
Q_Q53 5 51 43  QVPNP 4.7655
Q_Q54 51 60 62  QNPN 1.7453
Q_Q55 3 30 60  QNPN 0.6283
Q_Q56 5 42 61  QVPNP 1.2607
Q_Q6 2 9 8  QLPNP 1.1111
Q_Q7 1 9 7  QLPNP 1.1111
Q_Q8 10 9 6  QLPNP 1.1111
Q_Q9 5 6 4  QVPNP 1.2262
Q_Qz1 3 69 67  QNPN 0.6283
Q_Qz2 3 71 69  QNPN 0.6283
Q_Qz3 3 72 71  QNPN 0.6283
Q_Qz4 3 74 72  QNPN 0.6283
Q_Qz5 3 81 68  QNPN 0.6283

*** Resistors ***
*** Resistor Data ***
R_R1 19 5 536.25
R_R10 34 5 89.4
R_R11 33 5 89.4
R_R12 3 35 137.5
R_R13 3 36 754.3
R_R14 3 37 107.6
R_R15 45 22 44.9183k
R_R16 46 42 1.485k
R_R17 47 42 1.485k
R_R18 45 3 608.7223k
R_R19 52 5 693
R_R2 20 5 536.25
R_R20 49 Vout 804.4
R_R21 Vout 54 9.8
R_R22 Vout 50 9.8
R_R23 56 57 555
R_R24 57 32 360
R_R25 55 43 1.2375k
R_R26 58 41 266.5
R_R27 59 32 4.7614k
R_R28 59 5 1.7679k
R_R29 59 60 4.4314k
R_R3 20 5 536.25
R_R30 61 38 330
R_R31 62 5 41.3
R_R32 41 51 74.5
R_R33 14 63 88
R_R34 14 64 577.5
R_R35 3 65 2.55k
R_R36 65 67 2.55k
R_R37 3 66 2.55k
R_R38 66 68 2.55k
R_R39 67 70 158.6
R_R4 24 5 536.25
R_R40 70 69 158.6
R_R41 69 71 158.6
R_R42 71 72 158.6
```

```
R_R43 71 72 158.6
R_R44 72 73 158.6
R_R45 72 73 158.6
R_R46 14 75 9.15k
R_R47 13 76 9.15k
R_R48 81 76 9.15k
R_R49 73 75 9.15k
R_R5 25 5 429
R_R50 80 81 158.6
R_R51 80 81 158.6
R_R52 79 80 158.6
R_R53 79 80 158.6
R_R54 78 79 158.6
R_R55 77 78 158.6
R_R56 68 77 158.6
R_R57 73 74 158.6
R_R58 73 74 158.6
R_R59 1 0 1k
R_R6 25 26 742.5
R_R60 2 100 1k
R_R61 100 101 1k
R_R62 Vout 100 1k
R_R7 27 31 707.1
R_R8 28 31 707.1
R_R9 29 30 25.692k

*** Sources ***
*** Voltage Data ***
V0 3 0 DC 12
V1 5 0 DC -12
VIN 101 0 sin(0V 5V 1kHz)

*** Start Model Definitions ***
*** use some other transistor derived rad parameters

.MODEL QNPN NPN ( LEVEL  = 1
+ IS     = 1.85277E-16     BF     = 89.2            NF     = 0.9975
+ BR     = 0.505           NR     = 0.995604        ISE    = 5.24807E-16
+ NE     = 1.912           ISC    = 1.35479E-15     NC     = 1.03338
+ VAF    = 309.217         VAR    = 24.388          IKF    = 5.24807E-3
+ IKR    = 0.0191342       RB     = 100
+ RBM    = 0.1             IRB    = 1.01E-6         RE     = 4.21456
+ RC     = 197.969         TF     = 1E-10
+ XTF    = 1               ITF    = 0.01            VTF    = 5
+ PTF    = 20              TR     = 1E-8            XTB    = 0
+ EG     = 1.17
+ CJE=1.06p VJE=0.77 MJE=0.246 CJC=1.59p VJC=0.785 MJC=0.194 CJS=5p VJS=0.708
MJS=0.304)

.MODEL QLPNP PNP (LEVEL  = 1
+ IS     = 5.999635E-15    BF     = 378.2669158     NF     = 1.0841971
+ BR     = 135.482426      NR     = 1.05            ISE    = 1.06722E-15
+ NE     = 1.5497752       ISC    = 2.664285E-15    NC     = 1.4111948
+ VAF    = 39.1181         VAR    = 25.9038257      IKF    = 2.3926E-4
+ IKR    = 3.745563E-5     RB     = 342.1806875
+ RBM    = 100             IRB    = 4.75632E-4      RE     = 0
+ RC     = 0               TF     = 1E-10
+ XTF    = 1               ITF    = 0.01            VTF    = 5
+ PTF    = 20              TR     = 1E-8            XTB    = 0
```

```
+ EG     = 1.17
+ CJE=1.38p VJE=0.596 MJE=0.052 CJC=2.08p VJC=0.744 MJC=0.134
+ CJS=2.18p VJS=1.062 MJS=0.017)

.MODEL QVPNP PNP (LEVEL  = 1
+ IS     = 1.910728E-15    BF     = 3.609126E3      NF     = 0.9978895
+ BR     = 0.3369848       NR     = 0.9742775       ISE    = 4.341141E-17
+ NE     = 1.1174228       ISC    = 5.397116E-16    NC     = 1.0055126
+ VAF    = 44.0519562      VAR    = 19.6963495      IKF    = 1.95239E-4
+ IKR    = 1.488406E-4     RB     = 2.120287E4
+ RBM    = 0               IRB    = 1.020318E-8     RE     = 0.4715013
+ RC     = 115.7868121     TF     = 1E-10
+ XTF    = 1               ITF    = 0.01            VTF    = 5
+ PTF    = 20              TR     = 1E-8            XTB    = 0
+ EG     = 1.17
+ CJE=1.42p VJE=0.851 MJE=0.058 CJC=3.62p VJC=0.672 MJC=0.189 )

*** End Model Definitions ***
.END
```

Save this as OP27.cir. Usually I create a projects directory in the Xyce subdirectory for all my circuits, with a subdirectory for each circuit as shown below.

$ cd ~/Xyce7.2
$ mkdir projects
$ cd projects
$ mkdir op27
$ cd op27

Save the OP27.cir file here.

Now you can run Xyce by

$ Xyce -l op27.log -o op27.prn OP27.cir

-l defines the log file, -o defines the output file and the final argument is the input netlist.

This will only take a few seconds to execute. You can inspect the op27.log file to see how things went. The prn file is compatible with LibreOffice Calc, so fire up LibreOffice Calc to open the op27.prn file and visualize the output by selecting the time and voltages, and creating a graph (x-y line). You can clearly see the inverting nature of this circuit setup.

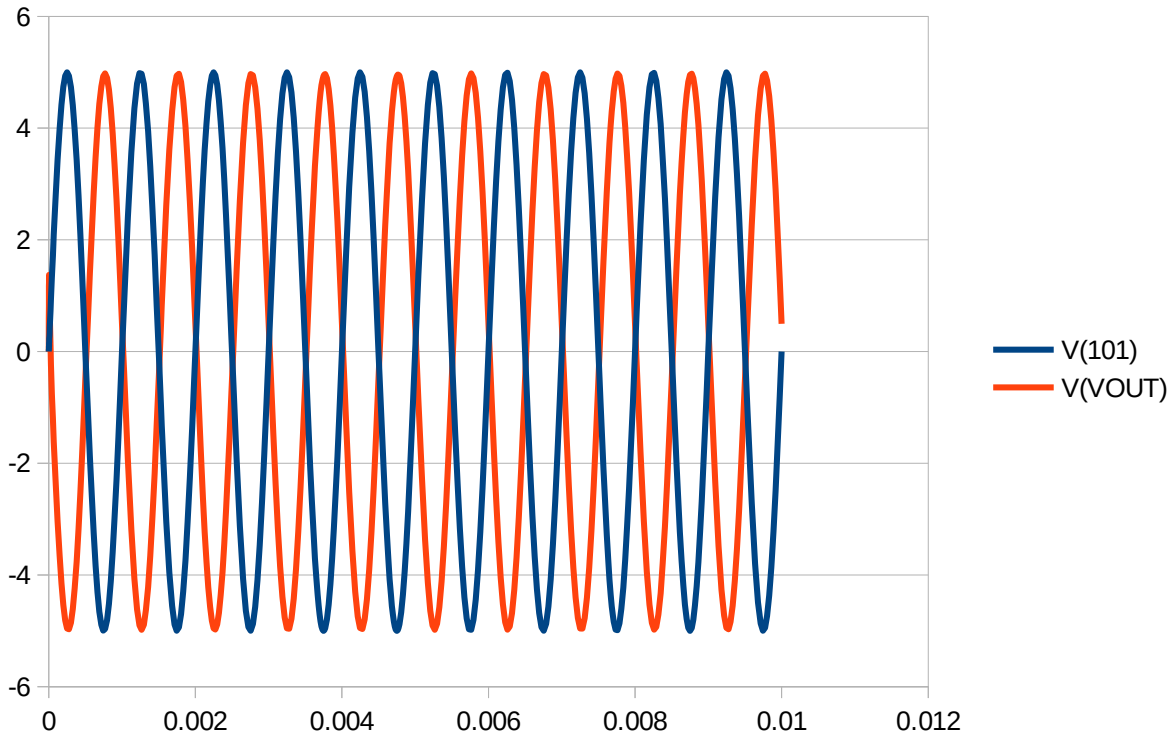

Illustration 1: Output of OP27 circuit simulation

Xyce will run most standard netlists for SPICE with just a few minor modifications. The Xyce Reference Guide in the Documentation folder has the specific changes needed for several popular circuit simulators.

And now you have a state of the art circuit simulator on your Raspberry Pi for your next circuit design project. There is an active Google Group that discusses implementing various models that you can join and potentially get some help with your particular issues.