

## Xyce<sup>®</sup> 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 3 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 3 B. These instructions are for Version 6.7 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 16GB or larger card or an external drive (strongly suggested). I use a solid state USB drive with 64 Gb of storage. This build has only been checked with Jessie.

Starting with the latest full Jessie 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
```

Next, we need to increase the swap file size for later compilation. You can probably revert back to 100Mb after everything is compiled, but you can leave the swap large if you desire:

```
$ sudo nano /etc/dphys-swapfile
```

Change the CONF\_SWAPSIZE from 100 to 2048 (the maximum possible), control-o to write the file, and control-x to exit. Now, to make the change effective, and this will take a minute:

```
$ sudo dphys-swapfile setup
```

Next we need the fftw fast Fourier transform library. I fire up the Chrome browser to do all the downloading. Go to <http://www.fftw.org/download.html> and grab the 3.3.6 release via tar.gz if you are doing it from the Raspberry Pi browser. 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-3.3.6-pl2.tar.gz
```

Now we build and install fftw:

```
$ cd fftw3-3.3.6-pl2
$ ./configure
$ make
```

```
$ sudo make install
```

Next we must get the Trilinos libraries, a very powerful set of serial and parallel solvers. Go to <https://trilinos.org/> and navigate to the Download and Previous Releases and on the right side the Pages section to find the version 12.6.3 link. You want this version for Xyce 6.7 as later versions aren't guaranteed to work with Xyce. You do need to sign up to get access, but it is free. This is a big file and it will take a few minutes to download. Be patient.

Now to unzip the file Trilinos files.

```
$ cd ~/Downloads
$ tar -xzf trilinos-12.6.3-Source.tar.gz
```

The last command places the source code in ~/Downloads/trilinos-12.6.3-Source, but we want to do an "out of source build", so we create a directory to do this.

```
$ cd ~
$ mkdir Trilinos12.6
$ cd Trilinos12.6
```

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. You also need to disable the c++11 standard as shown below. 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=$HOME/Downloads/trilinos-12.6.3-Source
ARCHDIR=$HOME/XyceLibs/Serial
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=OFF \
-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_Ipack=ON \
-DTrilinos_ENABLE_Isorropia=ON \
```

```

-DTrilinos_ENABLE_Aztec00=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 source for Source), so fix the reconfigure file, resave it, and rerun it.

Next is to actually build the code, this takes around 1 ½ hours on a Raspberry Pi Model 3B, 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 download Xyce from the <https://xyce.sandia.gov/downloads/sign-in.html> page. Once you register for free, they will open the real downloads page. Grab the files containing the source code Xyce-6.7.tar.gz from the previous releases page as well as the regression suite Xyce\_Regression-6.7.tar.gz and the documentation Xyce\_Docs-6.7.tar.gz, they will be placed in the Downloads directory. These are moderate downloads, so it will take a few minutes. Move there and unzip them, we will do another out of source build.

```

$ cd ~/Downloads
$ tar xzf Xyce-6.7.tar.gz

$ cd ~
$ mkdir Xyce6.7
$ cd Xyce6.7

```

The next line places the documentation in the proper Xyce folder.

```

$ tar xzf ~/Downloads/Xyce_Docs-6.7.tar.gz

```

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

```
$ nano config
```

```
#!/bin/bash
$HOME/Downloads/Xyce-6.7/configure \
ARCHDIR=$HOME/XyceLibs/Serial \
CPPFLAGS="-I/usr/include/suitesparse" \
CXXFLAGS="-O3" \
ADMS_CXXFLAGS="-O0" \
LDFLAGS="-L/usr/local/lib" \
CFLAGS="-O3" \
--disable-adms_sensitivities
```

Note that the options use the letter O after the dash and then a number (3 or 0) to set optimization levels. We turn off optimization on the complicated ADMS models to speed the compile with option -O0. The last line also speeds up the compile (otherwise it is impossible due to exceeding the swap space available. 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. Still in the Xyce-6.7 directory, we unpack the regression suite.

```
$ tar xzf ~/Downloads/Xyce_Regression-6.7.tar.gz
$ cd Xyce_Regression-6.7/TestScripts
$ ./run_xyce_regression
```

Again, the regression suite takes a while. You may fail a few tests related to calling Xyce as a library if you changed any directories, and you may fail the huge test problem (RCladder) which uses more memory than the Pi has, but the vast majority should pass.

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 Jessie.

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 QVNP 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 QVNP 1.2262
Q_Q15 5 11 15 QVNP 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 QLNP 1.1111
Q_Q26 30 16 28 QLNP 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 QLNP 1.1111
Q_Q30 39 38 35 QLNP 1.6799
Q_Q31 40 38 36 QLNP 1.2968
Q_Q32 41 38 37 QLNP 2.0631
Q_Q33 31 42 39 QLNP 1.875
Q_Q34 42 42 39 QLNP 416.7m
Q_Q35 21 21 3 QLNP 0.5787
Q_Q36 43 21 3 QLNP 0.5787
Q_Q37 43 21 3 QLNP 0.5787

```

Q\_Q38 43 21 3 QLPNP 0.5787  
 Q\_Q39 3 22 18 QNPN 496.6m  
 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 6.6198  
 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         TFE     = 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 QVNP 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 ~/Xyce6.7
$ mkdir projects
$ cd projects
$ mkdir op27
$ cd op27
```

Save the OP27.cir file here.

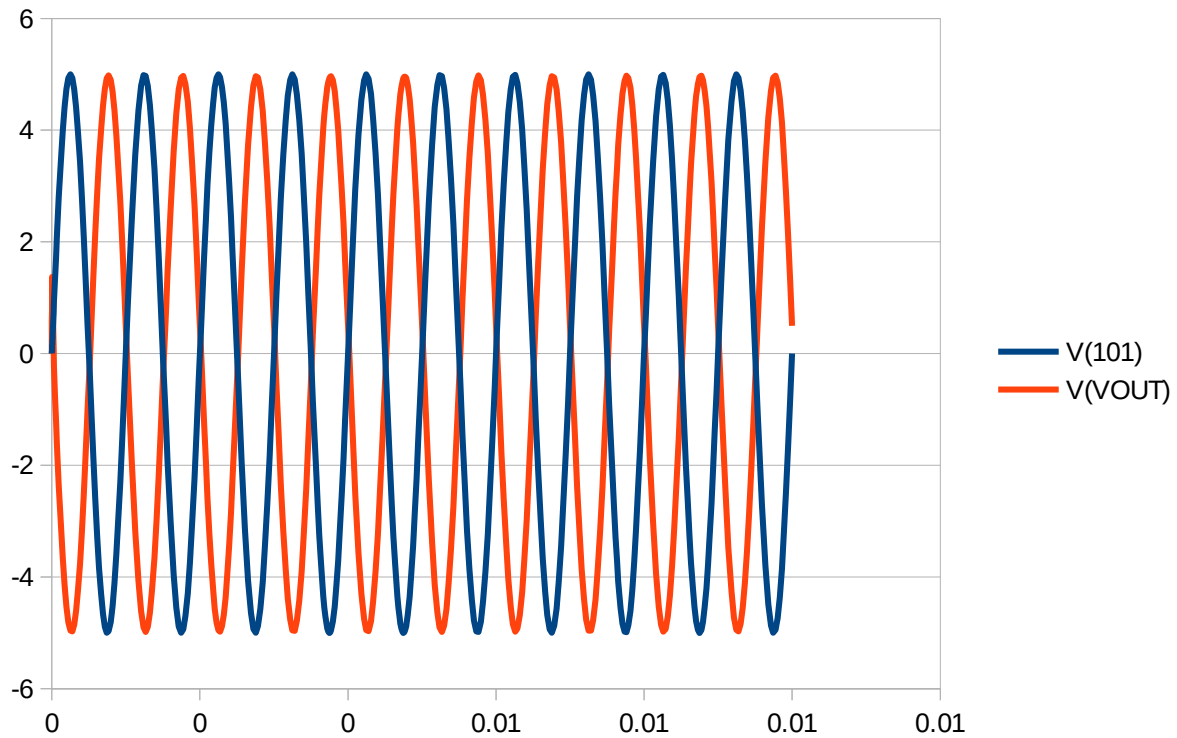
Now you can run Xyce by

```
$ 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.