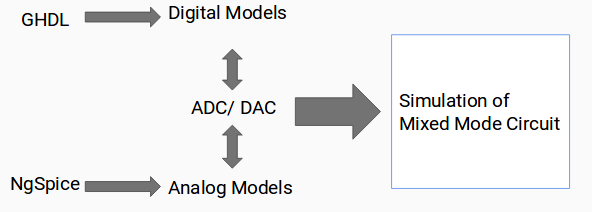
**NGHDL OVERVIEW :**

****

**What exactly interfacing of ngspice ghdl do?**

**→**Ngspice support mixed mode simulation. It can simulate both digital and analog component.

**→**Ngspice has something called model which define the functionality of your circuit,which can be used in the netlist.

**→**For example you can create adder model in ngspice and use it in any circuit netlist of ngspice.

**Now the question is if we already have digital model in ngspice why this interfacing ?**

**→**Well in ngspice it is little tediouse to write your digital model. But many people are familiar with ghdl and can easily write the vhdl code.

**→**So the idea of interfacing is just to write ghdl code for a model and install it as dummy model in ngspice. So whenever ngspice look for that model it will actually call the ghdl to get the result.

**Pre-requisites -**

**→***Ubuntu 16.04* - You can try it on other version and let us know

**→***Python 2.7*

**→***PyQt4*

**→***Ghdl* - Since PPA and package not available, needed to build from source using LLVM backend

**→***NgSpice* - Need to add custom models to ngspice using xspice, so need to build from source

**How to install?**

1. Clone this repository.

2. Run `./install-nghdl.sh` It will install ngspice from source code and put it in $HOME.

**Few words about installed code structure -**

**→**Ngspice will be installed in home directory $HOME. If you already have ngspice-26 directory there it will take its backup.

**→**Source code for all other file will be present in ~/.esim-nghdl

**→**symlink nghdl is stored in /usr/local/bin

**NGHDL WORKING :**

**User**

**Upload VHDL code**

**Add model to Schematic→Convert KiCad to NgSpice**

1. Digital model for NgSpice (from VHDL) :

Creates a dummy model for ngspice and Generation of C filesfor simulation :

a. Model directory created

b. Model added to modpath

c. Model files generated as follows -

i. Cfunc.mod

Defines NgSpice Model

ii. ifspecs.ifs (Converted to C as well)

iii. Nghdl server script (start\_server.sh)

iv. sock\_pkg.vhdl

d. Runs “*make”* and *“make install”* to generate code-models for NgSpice

The above mentioned task is done by two code files in ***eSim-1.1.2 --> nghdl --> src***:

a. **model-generation.py**

b. **ngspice\_ghdl.py**

2. NgSpice Simulation :

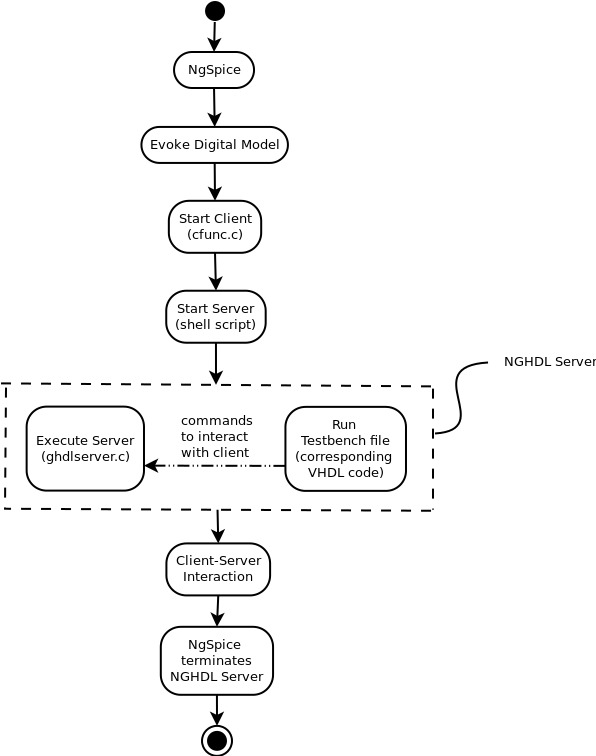
a. NgSpice runs *“cfunc.c”* of the respective model which acts as a client.

b. This client starts server shell script (***start\_server.sh***) which executes Testbench file linked with the Server file (***ghdlserver.c***).

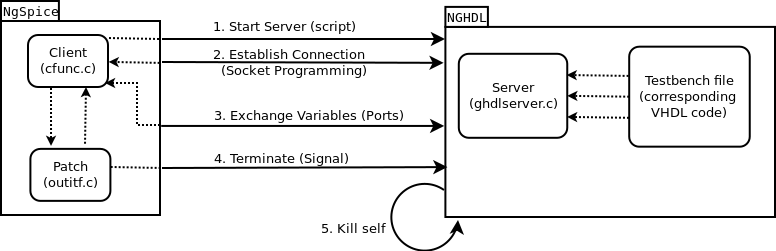
c. Testbench acts as a controller of NGHDL Server and commands the server file to connect and interact with the client.

d. NgSpice sends a signal to the Server file to terminate.

e. The Server file, in turn, terminates itself due to which Testbench also gets closed.

****

Client – Server Interaction :

**How to use?**

1. Run nghdl in command terminal.

2. Upload your vhdl file.

3. Model will be created with your name of your vhdl file.

4. You can use this model in your netlist.

**LIMITATION -**

1. You can use maximum 64 output ports in your file.

2. For a pulsating input, it works for only the first instance given to the model.

3. We can use only one code model of such type in our netlist.

**FUTURE WORK -**

1. Make changes to have output for pulsating input (multiple events).

2. Making changes to include use of more than one code models.