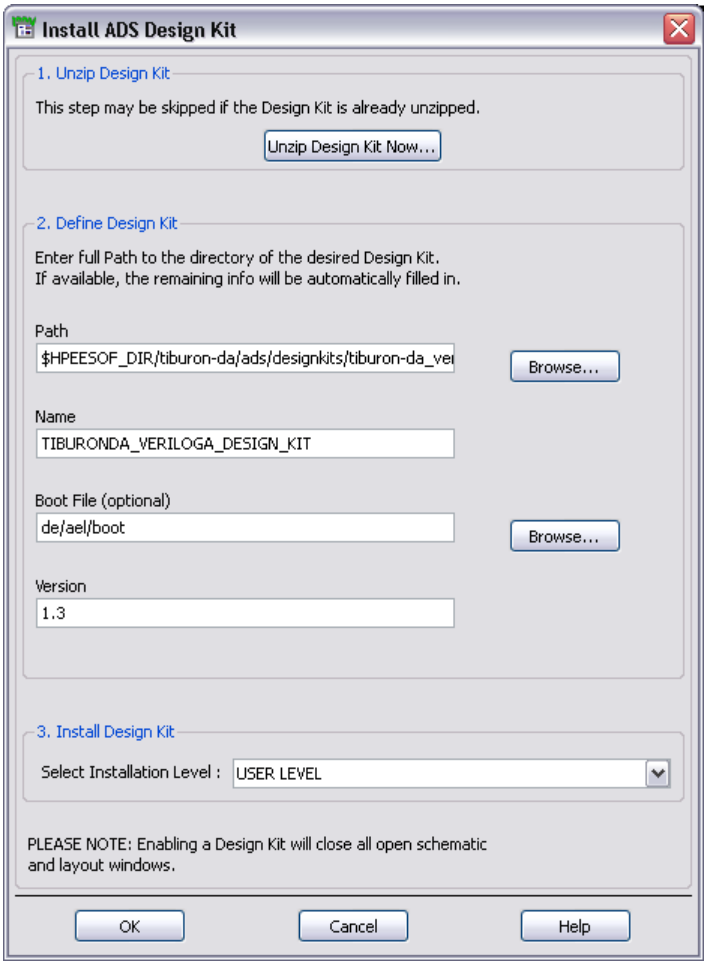
STEP-BY-STEP INSTRUCTIONS Verilog-A + ADS

Tutorial how to make Verilog-A in ADS and add created models to the simulation

1) Install design kit to run the verilog-A (instructions taken from the http://www2.ece.ohio-state.edu/~bibyk/ee894z/verilogaADS.pdf)
   a) Run ADS and in the 'main' window follow "DesignKit" -> "install design kits"
   b) Navigate in the 'path' field to "$HPEESOF_DIR/tiburon-da/ads/designkits/tiburon-da_veriloga" (or just type this into the field right away). E.g.: C:\ADS2009\tiburon-da\ads\designkits\tiburon-da_veriloga
   c) The fields should auto-load. Now finish the installation (OK).



2) Create a verilog-A file for example

DCO.va:

```
`include "constants.vams"
`include "disciplines.vams"
module DCO(b0,b1,b2,b3,sigOut);
input b0,b1,b2,b3;
output sigOut;
electrical sigOut;
voltage b0,b1,b2,b3;
integer progr_freq_select,n;
real period_out,freq_out,next,tt,temp;
parameter real logic_high=1.2,logic_low=0,logic_threshold=0.6;
parameter real t_delay=3p,t_rise=1p,t_fall=1p;
parameter real f_delta=5G;
parameter real f_min=10G,f_max=25G;
        analog begin
        progr_freq_select=(V(b0)>logic_threshold)+2*(V(b1)>logic_threshold)+4*(V(b2)>logic_threshold)+8*(V(b3)>logic_threshold);
                period_out=1/(f_min+progr_freq_select*f_delta);
                //for debugging purposes
                if(temp!=1/period_out) begin
                        temp=1/period_out;
                        $display("selected frequency is ",temp);
                        $display("selected state is ",progr_freq_select);
                end
                //set initial conditions of the time step (i.e. 1/2 period)
                @(initial_step) begin
                        next=0.5*period_out;
                        //$display("at initial_step next is now ", next);
                end
                //alternate the output level at rate of 1/2 of the selected period
                @(timer(next)) begin
                        n=!n;
                        //$display("the value of n is ",n);
                        next=next+0.5*period_out;
                        //$display("the value of next is ",next);
                end
                //rise and fall times dynamically update depending on output period
                tt=0.005*period_out;
                //output the DCO signal
                V(sigOut)<+transition(n ? logic_high : logic_low,0,tt);
                //$display("the value of sigOut is ",V(sigOut));
        end
endmodule
```

**********Note, The module name is also DCO**********

3) Create a new project and name it for example DCO_simple

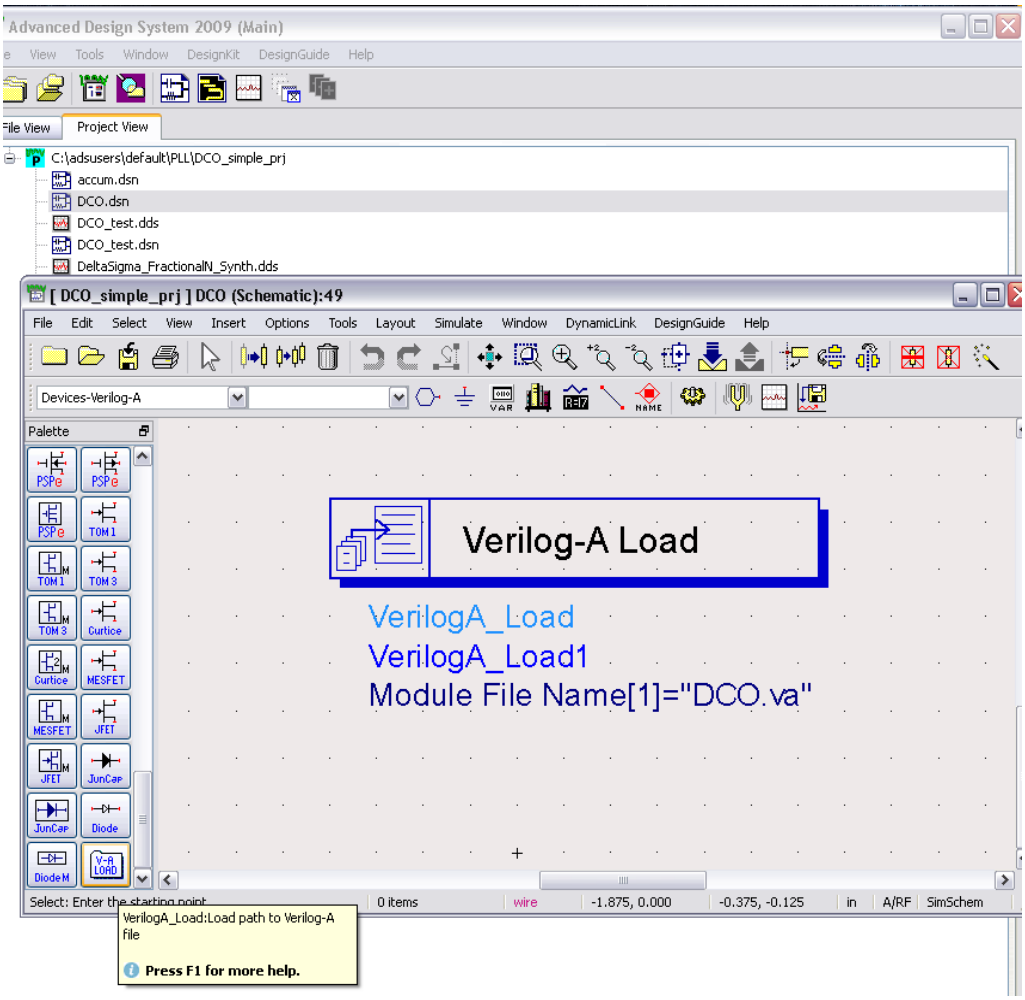
4) Create a new design and name for example DCO (schematic)

5) Navigate on your computer to the directory where this project was created (e.g. C:/adsusers/default/DCO_simple_prj)

6) create a folder “veriloga” and place your DCO.va file there.

7) Go back to the design (schematic) window and select the component palette called “Devices – Verilog - A”

8) From this palette select the “Verilog-A Load” component



9) Provide the file name “DCO.va” to the loader

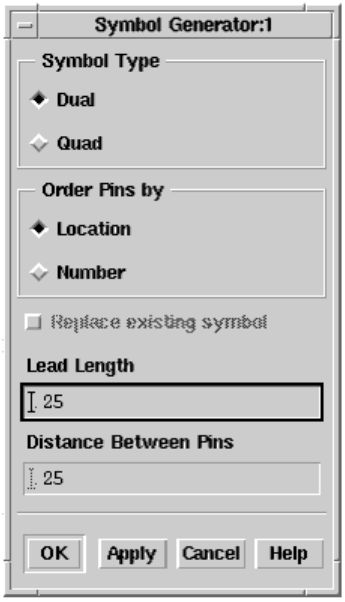
10) You can create the symbol at this point but cannot use the model yet

    a) Assuming the “DCO.dsn” file is still open, go to the “View->Create/Edit…”

Drawing a New Symbol

To draw a new symbol for your design kit:

1. Choose **File** > **Open Project** to open a project with a new schematic window. For more information on ADS projects, refer to Managing Projects and Designs.
2. Access the ADS Symbol Editor by choosing **View** > **Create/Edit Schematic Symbol** . The Symbol Generator dialog box appears.

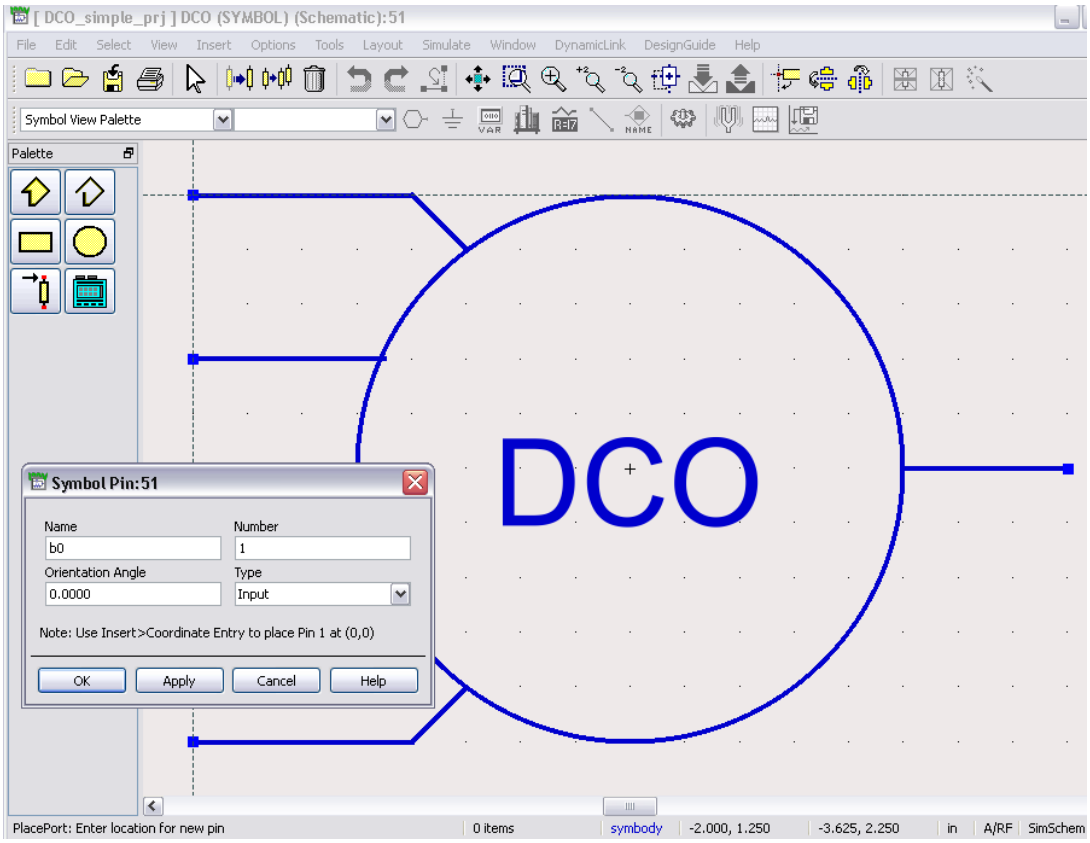


3. Close the Symbol Generator dialog and draw the symbol by hand.
4. See also, Setting the Design Type for Symbols.

For more information on creating symbols in ADS, refer to Drawing a Custom Symbol and/or ADS Schematic Symbol & Bitmap Creation.

> **Note**
> Use of the Symbol Generator is not appropriate for design kit development because the symbols that are automatically generated by the system are just box-like symbols intended for representing a hierarchical design in ADS. For design kits, you will want to create a more meaningful symbol that properly represents the component for which it is a symbol.

b) Draw the symbol you like and add pins with the same names as they were declared in your file (e.g. b0,b1,b2,b3,sigOut) and with appropriate modifiers (input/output)



11) Save this file and navigate to the project directory i.e. "C:/adsusers/default/DCO_simple_prj" and find the folder "networks" locate the AEL file for this DCO ("DCO.ael", may need to run a dummy simulation which will fail just to force ADS to create these files) and open it with text editor to modify it as follows:

```
create_item("DCO",                    // name
        "DCO",                                                  //label
        "DCO",                  // prefix
        0,                              // attributes
        NULL,                           // netlist priority
        "v_dco",                // icon name
        standard_dialog,                // dialog name
        "*",                            // dialog data
        ComponentNetlistFmt,            // netlist format
        "DCO",                  // netlist data
        ComponentAnnotFmt,              // display format
        "DCO",                  // symbol name
        no_artwork,                     // artwork type
```

```
        NULL,                           // artwork data
        ITEM_NOSUBNET_HEADER_EX,
        create_parm("b0","DCO b0", PARM_OPTIMIZABLE | PARM_STATISTICAL, "StdFileFormSet", UNITLESS_UNIT,
prm("StdForm","1.0")),
        create_parm("b1","DCO b1", PARM_OPTIMIZABLE | PARM_STATISTICAL, "StdFileFormSet", UNITLESS_UNIT,
prm("StdForm","1.0")),
        create_parm("b2","DCO b2", PARM_OPTIMIZABLE | PARM_STATISTICAL, "StdFileFormSet", UNITLESS_UNIT,
prm("StdForm","1.0")),
        create_parm("b3","DCO b3", PARM_OPTIMIZABLE | PARM_STATISTICAL, "StdFileFormSet", UNITLESS_UNIT,
prm("StdForm","1.0"))
        );
```

BELOW ARE THE MEANING OF THE ABOVE FUNCTIONS

Item Definition

All components that exist in ADS are added by calling an AEL function called **create_item( )** . Each component can have a number of parameters, which are added to the system by calls to the function **create_parm( )** . The combination of these calls is referred to as the *item definition* . A user interface is provided to set up some of this information, but it also usually requires some additional manual editing. The tutorial gave an example of this process.

In addition to parameters and their default values, the item definition in ADS contains information about a component that is necessary for display on the schematic. There is also information included that is needed for simulation. The following section describes the syntax and the arguments of the **create_item( )** function that are important for design kits. Additional information on the **create_item( )** command is available in Chapter 14 of the *AEL* documentation.

The syntax for the **create_item( )** command is shown here. The square brackets indicate optional parameters, but are not entered in the actual function call.

```
create_item(name, desc, prefix, attrib, priority, iconName, dialogCode,
            dialogData, netlistFormat, netlistData, displayFormat,
            symbolName,artworkType, artworkData [, extraAttribute, cbList,
            parameterN]);
```

An example similar to that in the tutorial is repeated here for reference. The callback and **create_parm( )** information is incomplete but is sufficient for demonstration purposes. The parameter descriptions are included in The create_item() Parameter Descriptions.

```
create_item("mykit_npn", "MYKIT Nonlinear Bipolar Transistor", "BJT", NULL,
            NULL, NULL, standard_dialog, CmpModelNetlistFmt,
            ComponentAnnotFmt, SYM_mykit_npn, no_artwork, NULL,
            ITEM_PRIMITIVE_EX, list(callback1, callback2, etc),
            create_parm("parm1",...), create_parm("parm2",...));
```

The create_item() Parameter Descriptions

Argument	Description
name	Name of the component. It must be completely unique in the system. For design kits, it should include a company and/or process reference.
desc	Description of the component. 80 character limit. This shows up in the edit component dialog as well as in the balloon help when the cursor is positioned over the component in the palette.
prefix	The prefix is typically 1-3 characters but can be longer. It is used to create a unique ID when the component is placed in a schematic.
attrib	This is a special attribute code. The AEL manual contains a list of codes in Table 14-1. Each code has a numerical value assigned to it. Use of these codes is explained below. A value of 0 or NULL indicates that no special codes are set.
priority	Netlisting priority. Set the netlisting priority to NULL or -1 for all components except the netlist include components. For netlist include components, set the netlisting priority to 0 (see Avoiding Illegal Nested Subcircuits.)
iconName	Not used for design kit components - set it to NULL.
dialogCode	This tells the system which dialog to open for editing of parameters and other component attributes. It is usually set to standard_dialog, which is a constant value. A constant value is a special value that the system knows about. It is not a string. Do not put it in quotes.
dialogData	Not used for design kit components - set it to "".
netlistFormat	This tells the system what format to use when the component is output to a netlist. Two constant values are defined that should be sufficient for most design kit components. These are not strings so do not put them in quotes. These netlist format values are:
netlistData	Not used for design kit components - set it to "".
displayFormat	This tells the system how to display the component annotation on the schematic. Component annotation consists of the component name and unique ID. For design kit components, use the value ComponentAnnotFmt. This is a constant known to the system, not a string. Do not put quotes around it.
symbolName	This is the name of the symbol that was created for the component. It is the name of the design file containing the symbol graphics, without the .dsn extension. Note that this name must be different than the name of the component.
artworkType	This tells the system what type of artwork is associated with the component. The following values are used. The equivalent constant value in parentheses can also be used.
artworkData	If artworkType = 0 or 3, set this to NULL.
extraAttribute	This is another attribute code. The AEL manual contains a list of these codes in Table 14-2. Each code has a numerical value assigned to it. Use of these codes is explained below. A value of 0 or NULL indicates that no special codes are set.
cbList	List of netlist callbacks. See Netlist Callbacks for details.
parameterN	All parameters of the component, each defined by a separate call to create_parm(). These are not presented in the AEL list() syntax like the callbacks are. Instead they are just listed one after the other, separated by a comma, as if they were regular arguments. See The create_parm() Parameter Descriptions for details on the create_parm() function.

Parameter Definition

This section describes the syntax and arguments for the **create_parm( )** function, which is used as an argument in the **create_item( )** function, described above. More information is included in Chapter 14 of the *AEL* documentation.
The syntax for the **create_parm( )** command is shown here. The square brackets indicate optional parameters, but are not entered in the actual function call.

```
create_parm(name, desc, attrib, formSet, unitCode, defaultValue [,cbList]);
```

An example similar to that in the tutorial is repeated here for reference. The callback information is incomplete but is sufficient for demonstration purposes. The parameter descriptions are included in The create_parm() Parameter Descriptions.

```
create_parm("Model", "Model Instance Name", 0, "StdFileFormSet", -1,
            prm("StdForm", "BJTM1"),
            list(callback1, callback2, etc));
```

The create_parm() Parameter Descriptions

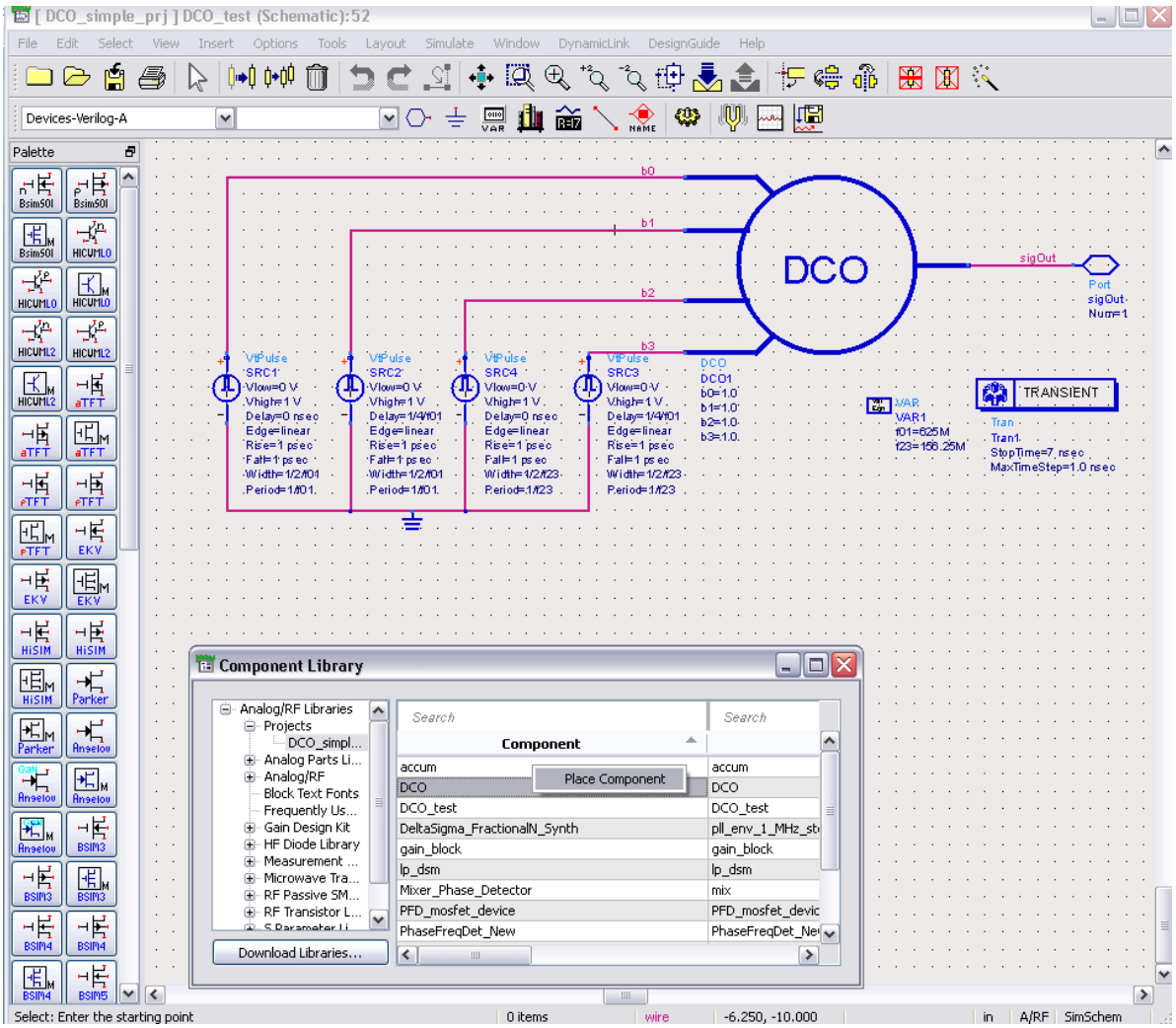
Argument	Description
name	Name of the parameter.
desc	Description of the parameter. This text is seen by the user in the edit parameter dialog when placing a component or modifying its parameter values.
attrib	Special attribute code. See "How to Use Attribute Codes" above. The list of commonly used parameter attributes is given in the AEL manual in the create_parm() section. Design kit components usually do not need a special attribute code. Set the value to 0.

Notes:

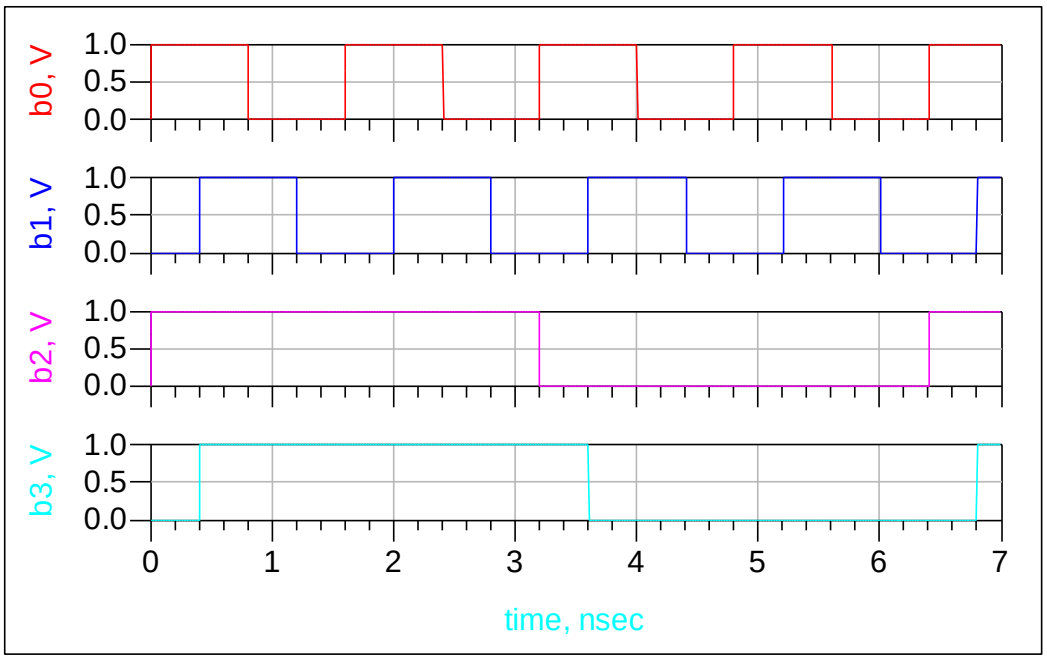
One attribute that may be useful in design kits is PARM_NOT_EDITED. This will prevent the user from modifying the parameter and is typically used for dependent parameters whose values are set by callbacks. Do not use this attribute in design kits that are used in ADS versions prior to ADS 2002C. Prior to that release, callbacks could not modify a parameter with this attribute.
The code for PARM_NOT_ON_SCREEN_EDITABLE may be used in conjunction with formsets. This will prevent the user from typing in a value that is not in the preset list, but it will also prevent the user from using the up/down arrows to scroll through the values on-screen. |

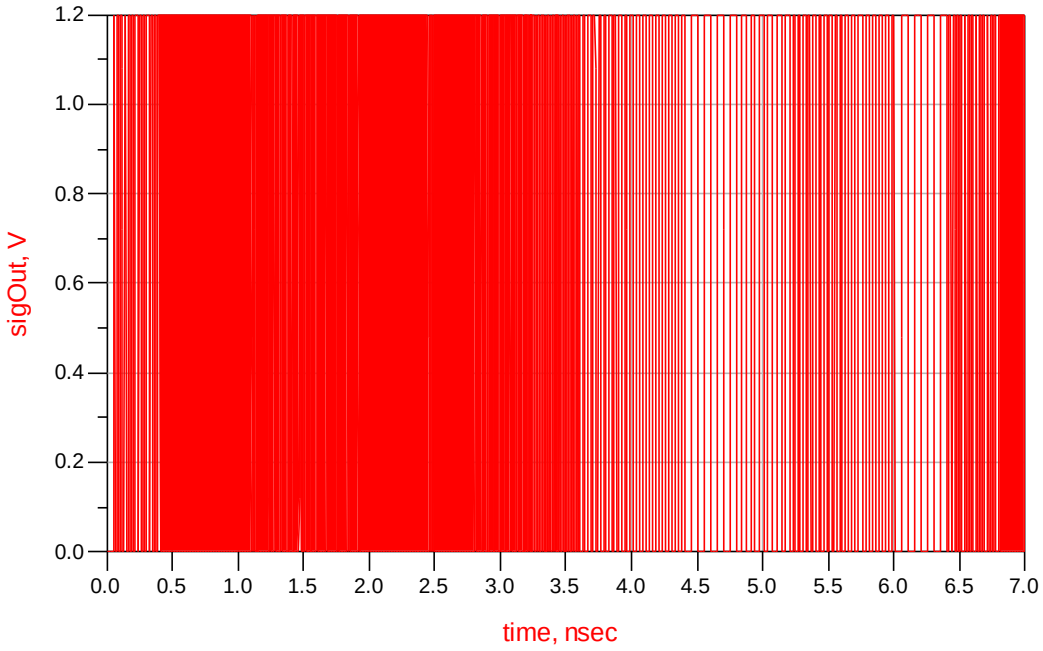
formSet	Forms are used as a way to present the user with a list of values. A parameter value is then limited to those defined in a form set. List the name of the form set here. See Forms and Formsets, or Chapter 14 of the AEL manual.
unitCode	The following is the list of unit types for ADS components. STRING_UNIT = -2 UNITLESS_UNIT = -1 FREQUENCY_UNIT = 0 RESISTANCE_UNIT = 1 CONDUCTANCE_UNIT = 2 INDUCTANCE_UNIT = 3 CAPACITANCE_UNIT = 4 LENGTH_UNIT = 5 TIME_UNIT = 6 ANGLE_UNIT = 7 POWER_UNIT = 8 VOLTAGE_UNIT = 9 CURRENT_UNIT = 10 DISTANCE_UNIT = 11 TEMPERATURE_UNIT = 12 DB_GAIN_UNIT = 13
defaultValue	defaultValue is optional. If specified, it needs to be in the form of a value returned from the prm() function. The prm() function generates an acceptable default value for parameters with different form sets. If a formset is not set for the parameter, use "StdForm" as in the example prm("StdForm", "BJTM1");.
cbList	List of parameter callbacks. See Parameter Callbacks for details.

12) Now you can create a new design (schematic) called “DCO_test.dsn” and add your DCO model to it “Insert”->”Component”->”component Library”.



13) Add the sources and terminations, simulate.





STILL TO FIGURE OUT

How to tie the external parameters to the internal functions (e.g. b0,b1,b2,b3 modifiers near the symbol)

Things that helped me

1) http://cp.literature.agilent.com/litweb/pdf/ads2008/usrguide/ads2008/Working_with_Symbols.html#WorkingwithSymbols-DrawingaCustomSymbol

2) http://cp.literature.agilent.com/litweb/pdf/ads2008/dkarch/ads2008/Basic_Parts_of_an_ADS_Design_Kit.html

3) http://www.edaboard.com/viewtopic.php?t=164139&highlight=veriloga+ads

4) http://edocs.soco.agilent.com/display/ads2009/Using+Verilog-A+and+Verilog-AMS+in+Advanced+Design+System

5) http://edocs.soco.agilent.com/display/ads2009/Getting+Started+with+Verilog-A+and+Verilog-AMS+in+Advanced+Design+System#GettingStartedwithVerilog-AandVerilog-AMSinAdvancedDesignSystem-InstallingVerilogA%2FAMSDevicesProvidedinaDesignKit

6) http://edocs.soco.agilent.com/display/ads2009/Introduction+to+AEL