Application Note

DxDesigner as a Mixed-Language Schematic Capture for SMASH™

Desert Microtechnology Associates, Inc.
16611 North 91st Street, Suite 103
Scottsdale, AZ 85260-1523 USA
Phone: 480/515-0005 FAX: 480/515-0788
Email: ngilbert@desertmicro.com

1. Introduction

This application note demonstrates through an example how DxDesigner (which will be referred to as ViewDraw in this document) can be used as a schematic capture tool to generate netlists that are easily compatible with the suggested file format in SMASH™. An example project directory called "`Viewnet`" will be used to illustrate the proposed design flow. This directory is available upon request on the SMASH™ website as a zip file and should be downloaded, unzipped and placed in the reader's simulation directory before continuing if the reader wishes to run the example step by step.

The operating system used for this example was Windows XP. It has been assumed that the reader is a current user of ViewDraw and SMASH™ and therefore has a basic understanding of how they work. The reader should refer to the ViewDraw manuals or/and SMASH™ manuals if further clarification is needed.

The example project includes a very elementary design of a 2-bit Flash ADC that is used throughout this document.

- The various directories in the example project are explained in section 2.
- The schematic entry and what each symbol represents are described in section 3.
- Netlisting the schematic is performed with the SPICELink program in ViewDraw and is described in section 4.
- Section 5 discusses the SMASH™ specific netlist (`.nsx`) and pattern (`.pat`) file formats and how to simulate the generated netlist through SPICELink with SMASH™.
- The simulation is performed and illustrated in section 6.
- The possibility for cross-probing and back-annotation thanks to the flexibility of ViewDraw is explained in section 7.

2. Description of the directories

The "`Viewnet`" example directory contains two sub-folders called "`Schematic`" and "`Smash`". These directories contain all of the necessary files to view the Flash ADC example and simulate it using SMASH™.

- The "`Schematic`" directory contains the key design schematics and Verilog code that can be used with other directories and can be considered as the library directory.
- The "`Smash`" directory has all the testbench schematics, symbols, netlist and pattern files, Verilog-A files, and Verilog files that are specific to the design or are used for simulation only.

Instructions to setup these directories are described in the following sections.

2.1 Setting up Project Files

A project file is required by ViewDraw to start work on a group of schematics. The project information resides in "`Smash`" directory and should be selected with the open project option using the Dashboard in ViewDraw from the start menu: *Programs → eProducts Designer → Dashboard*. Once the Dashboard is opened and the "`Viewnet`" directory has been placed in your simulation directory, you may open the project.

The opened project will automatically become the active project. The project can be opened by selecting *File → Open → Project*. You should then type in the path to the "`Smash`" folder, which should be something like "`C:\simul\Viewnet\Smash`". If you expand the project, two libraries should be found along with the other folders in that directory. The Dashboard window is shown in **Figure 1**.

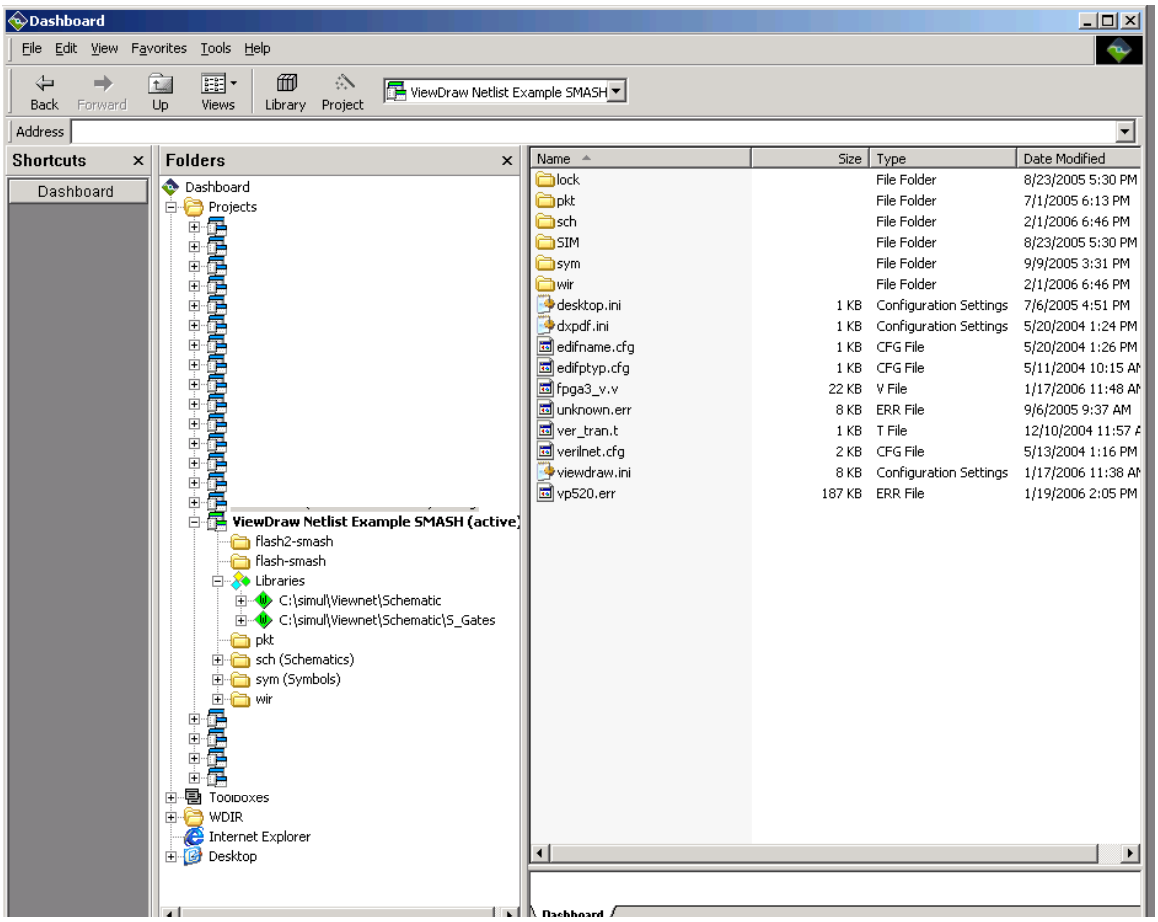


Figure 1: Example Dashboard where the "`Smash`" directory is opened.

The project information is stored in the file "`viewdraw.ini`" and can be edited in the Dashboard or any text editor.

Note:
You will need to modify the path of directories at the end of "`viewdraw.ini`" if it is different from "`C:\simul\`".

2.2 The Schematics Directory

In **Figure 1**, two libraries are included in the project, both of which are in the schematics directory. The schematics directory contains all of your design schematics and symbols. The schematics files are found in the folder called "`sch`" and symbols are in the "`sym`" folder. The "`S_Gates`" folder is explained in the next section. The folder "`wir`" contains the wire files used by ViewDraw to create the netlists. The schematics included in the directory are called "`decode`" and "`Vref`." Any additional design schematics would be added to this directory.

The symbols have the same name as their schematics and have the symbol value `COMPOSITE` as will be explained in section 3. The other symbols used for this design are included in this same folder. Any additional symbols needed for the design should be added to this directory. This directory can then be used as a library for other designs in the project or for other projects.

2.3 S-GATES Directory

In addition to the "`sch`" and "`sym`" folders, the "`S_Gates`" folder is included in the schematics directory where the logic gate symbols reside. The logic gates in this simulation are implemented with Verilog code which will be referenced by the symbols. The code for all of these gates is located in the file "`logic.v`" This file can be updated with the reader's own Verilog code. This file may be compiled as a global library in SMASH™ to increase speed of project compilation for larger designs (please refer to §9.7 Storing a module as a library element of the SMASH™ Reference Manual). This was not done in this example for sake of maintaining simplicity. The other folders are again used by ViewDraw for netlisting.

2.4 Smash Directory

The "`Smash`" directory can also contain schematics and symbols but is primarily thought of as the simulation directory, and contains testbench schematics (called "`flash`" for this example). In addition to the project file, this directory contains all necessary simulation files. The "`flash.cir`" file is produced by the ViewDraw netlister SPICELink. The file "`wspice_S.cfg`" is used by SPICELink to configure the netlist for SMASH™ simulations. The "`flash.nsx`" and "`flash.pat`" files are used in SMASH™.

The models for the transistors used in this example are found in the file "`TSMC_0um35_N88Y_sim.mod`". The zip file "`flash1.zip`" contains an example of the flash design with the comparator block modeled with Verilog-A instead of SPICE. The SMASH™ simulation files will be added to this directory and other testbench simulation netlists generated from ViewDraw will show up in this directory as well.

3. Schematic Entry

Now that the schematics and simulation directories are understood, the actual schematic entry can be explored in ViewDraw. Since the "`Smash`" project is now set as the active project, all of its schematics and symbols can be viewed with ViewDraw. It can be opened with the Dashboard or the start menu. In the Dashboard, expand the "toolbox" icon, click on the "board-level (PCB) Design" icon and double click on the icon "DxDesigner" at the top of the name window on the right of the Dashboard. Or it may be opened from the start menu: *Programs → eProducts Designer → Analog & Digital Simulation → Batch Mode → DxDesigner*.

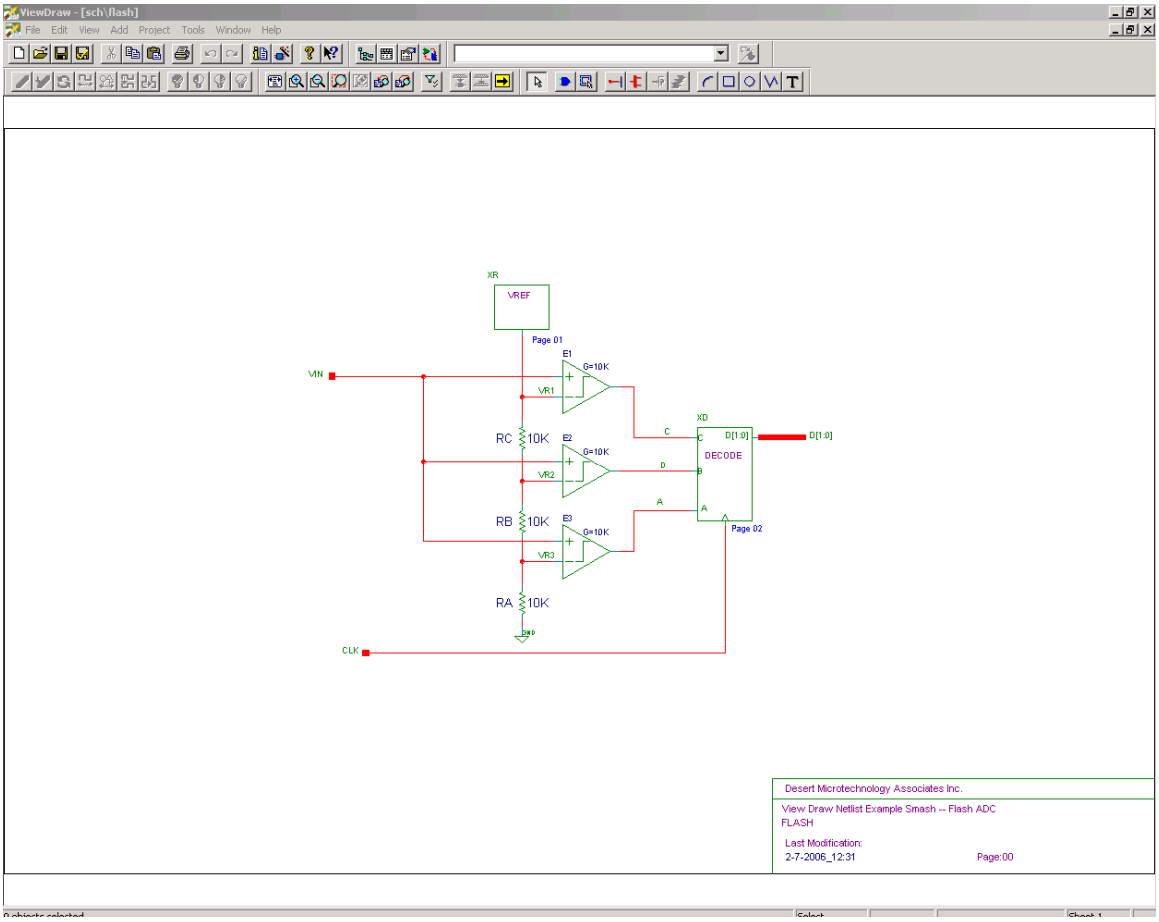


Figure 2: Schematic of simple flash design to illustrate netlisting for SMASH simulation.

Once ViewDraw is open, the testbench schematic can be viewed with the open icon by selecting the schematic in the "`Smash`" directory. The schematic is shown in **Figure 2**. Use the menu or hotkeys to modify the schematic if desired. Refer to the help menus for further clarifications or to create a new schematic. The following sections discuss how these schematics and symbols are organized to netlist SMASH compatible schematics.

3.1 Verilog Symbols

The Verilog symbols are located in the thermometer to binary block called `DECODE`. By right clicking on the `DECODE` symbol and pushing into the schematic, the logic symbols can be found. By double clicking on one of the logic symbols, the attributes of the block can be viewed. By pushing into the symbol, these attributes can be modified.

Note:
To create a new symbol the reader should refer to the ViewDraw manuals.

The symbol is set up to netlist a subcircuit call which will reference the Verilog code in the "`S_Gates`" directory. These Verilog blocks can reference any Verilog code and take on any shape or sized desired by the user. **Figure 3** shows the attributes associated with the latch to netlist the correct subcircuit. The first attribute listed is the MODEL parameter which matches the module name in the Verilog code. The next attribute is the ORDER attribute which adds the model name to the end of the node declarations.

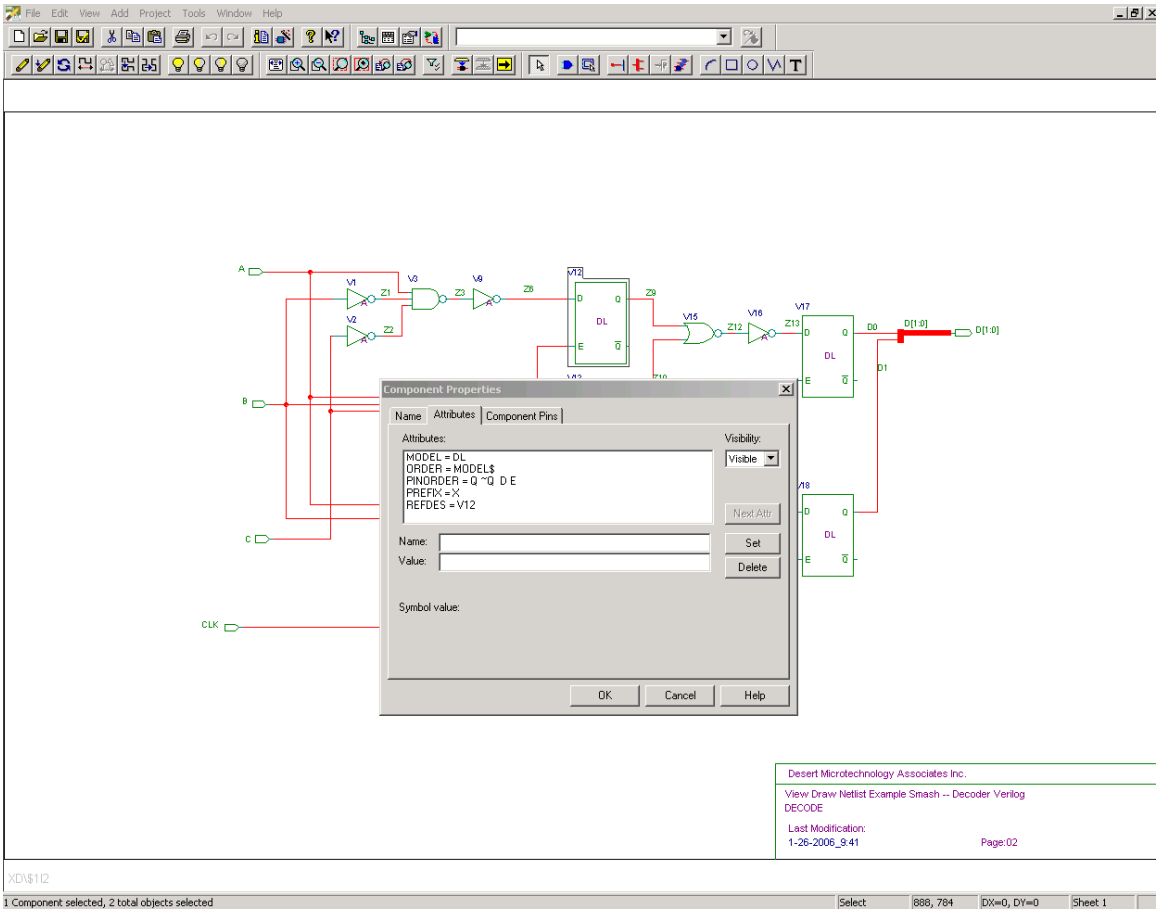


Figure 3: Attributes for Verilog Symbol.

If any parameters are passed into the Verilog module, they must be included with a "`parameter_name=`" value in the `ORDER` attribute and attached to the symbol. The `PINORDER` attribute is used to insure that the port declarations of the subcircuit call correspond to the declaration order in the module. The required prefix for a subcircuit is added with the `PREFIX` attribute and numbering can be maintained with the `REFDES` attribute. Notice that in **Figure 3**, the `REFDES` value used is a "`V`" to denote a Verilog block.

As Shown in **Figure 3**, the attributes for the Verilog Symbol should be as follows:

Name	**Value**
MODEL	<model_name>
ORDER	MODEL$ <Parameter_name>=
<parameter_name>	<Default value for parameter>
PINORDER	<List of pins in same order as Verilog module>
PREFIX	X
REFDES	V?

The values in < > must be adjusted by the user.

In **Figure 4**, the D-Latch symbol is shown in editing mode. Note the symbol type. Since there is no schematic associated with this symbol, the symbol type parameter, circled, must be set to MODULE. For the symbols that have a schematic associated with them, such as the DECODE and VREF symbols, this block is set to COMPOSITE. Composite blocks are automatically netlisted as sub-circuits.

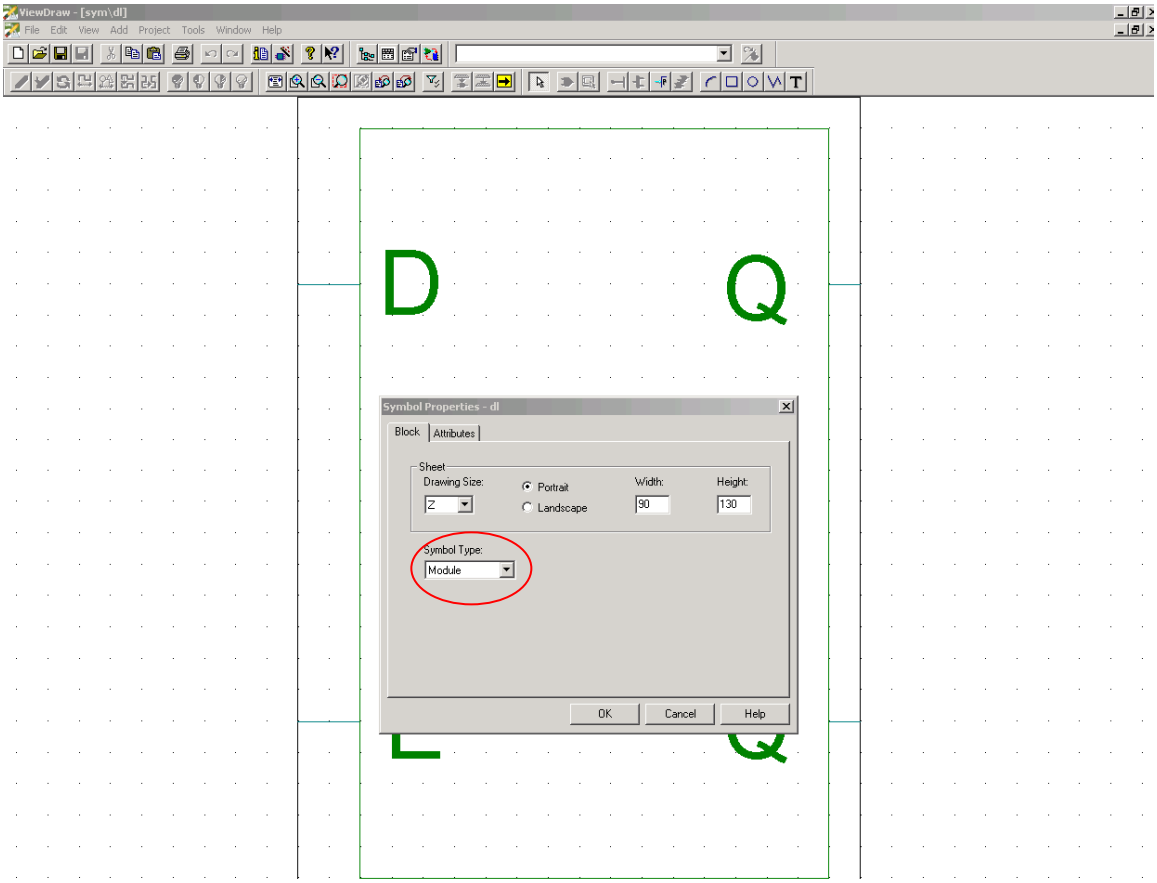
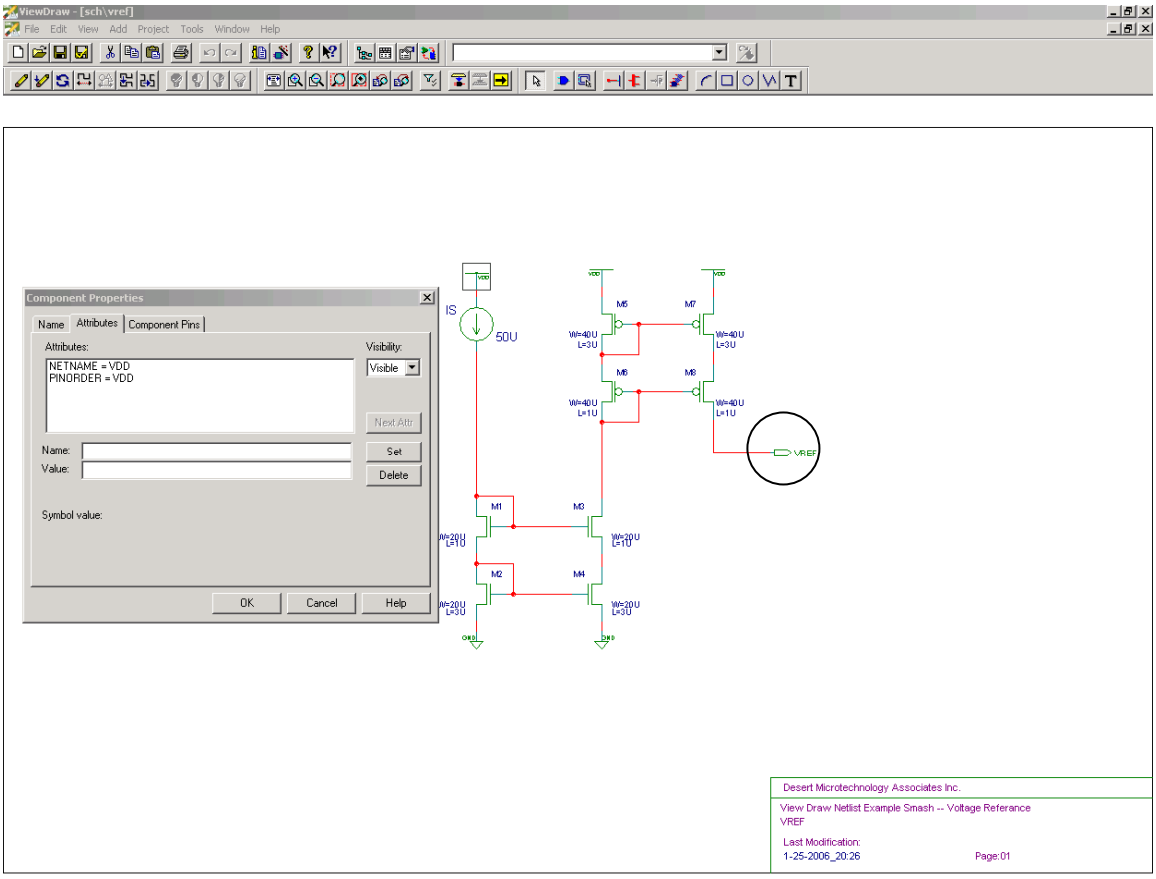


Figure 4: D-Latch symbol with block attributes shown.

3.2 SPICE Symbols

The rest of the symbols in this example are SPICE devices. These devices are also modules and the attributes are very similar to the Verilog symbol shown in **Figure 3**. The main difference is that the `PREFIX` attribute is changed to match the SPICE device. For example, resistors have a `PREFIX` of "`R`" and MOS devices have a `PREFIX` of "`M`". The comparators are implemented with E-elements with HSPICE syntax and the reader can review the symbol and netlist for further details.

The example schematic "`flash1`" uses Verilog-A code for a latched comparator to implement the comparator block instead of SPICE elements. The schematics for the VREF are shown in **Figure 5** where the supply and ground symbols are also shown. The power and ground symbols have a symbol type of `PIN` since they represent global nets. The ports to connect the schematic to the symbol pins are included and are circled in **Figure 5**. The labels of these ports need to match the names of the pins on the composite symbols.



XR\VDD

Figure 5: Schematic of VREF demonstrating the attributed of global pins and showing port symbols.

3.3 Mixed-Mode Schematics

The mixed-mode (mixed-signal and mixed-language) schematic shown in **Figure 2** illustrates SPICE and Verilog blocks in the same schematic. The digital analog interface is defined by the global attributes set by the SMASH™ simulator or added to the `flash.pat` file.

To specify the attributes for a given node, an interface device can be created using a symbol with the appropriate attributes to netlist a device like that which is explained in § 7 Mixed-Signal and Multi Language Interfaces of the SMASH™ Reference Manual. This is left as an exercise for the reader. All of the symbols in the design netlist actually produce SPICE parameters.

The Verilog is referenced through the subcircuit calls, and is represented as a schematic in **Figure 3** since SMASH™ will create an analog/digital interface when a subcircuit referencing a digital module is connected to a SPICE node. Therefore, the Verilog code for the `DECODE` block is represented as a schematic which is connected though the subcircuit connections in the SPICE netlist and all of the signals in the block essentially stay digital. The nodes `VIN` and `CLK` are left unconnected in the schematic because the input voltage stimulus for both nodes is found in the `flash.pat` file.

4. SPICELink and Configuration file

The SPICELink tool creates the netlist for simulation form the top level schematic found in **Figure 2**. In order to netlist the schematic properly, the configuration file must be created. As explained, this file is located in the "`Smash`" directory. The configuration file "`wspice_S.cfg`" is listed below:

```
|wspice_S.cfg
ADD >>> SPICE
ADD * Netlist for Simulation with SMASH
ADD *
ADD .global vdd
ADD *
ADD *
CALC_AREA 1
CALC_RES 0
AD 0.9U*W
AS 0.9U*W
PS 1.8U+2*W
PD 1.8U+2*W
NRD 0.5U/W
NRS 0.5U/W
PPARNAMS AS,AD
UPARNAMS PS,PD
ELEMENTS RCLVIEFGHSQDMXN
COMMANDS .SUBCKT .ENDS .INCLUDE
STRIP COMMENTS
STRIP_COMS *
```

The command "`ADD`" adds the text to the netlist. The necessary SPICE indicator is placed first to tell SMASH™ it is looking at SPICE code. A few comments and the global power supply are then added. The next parameters are used to add the drain and source area and perimeter estimates for the MOS devices and lists of elements and commands are the final commands of any consequence.

Note:
For more explanations on the configuration file syntax, refer to the SPICELink manual.

Once the configuration file has been completed correctly, the SPICELink program can be run and the schematic can be netlisted. It is suggested that the schematics checker program be run first before running SPICELink. This program insures that all of the library directories and *.wir files are created and in the correct locations to generate the schematics. It also looks for any errors in the schematics that would prevent the generation of a proper netlist.

The schematics check is under the tools menu which is the same place where SPICELink can be launched. Step by step screen shot are shown in Figures 6 to 9 illustrating all of the setting to for SPICELink.

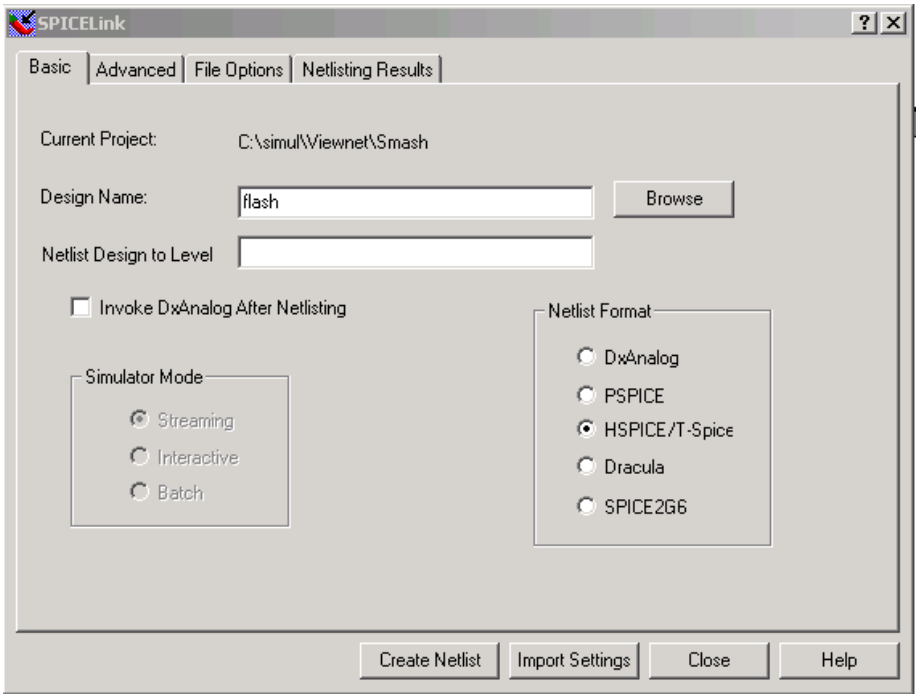


Figure 6: Basic SPICELink settings. Notice that for this example the netlist format should be HSPICE/T-SPICE (i.e. in `.cir`).

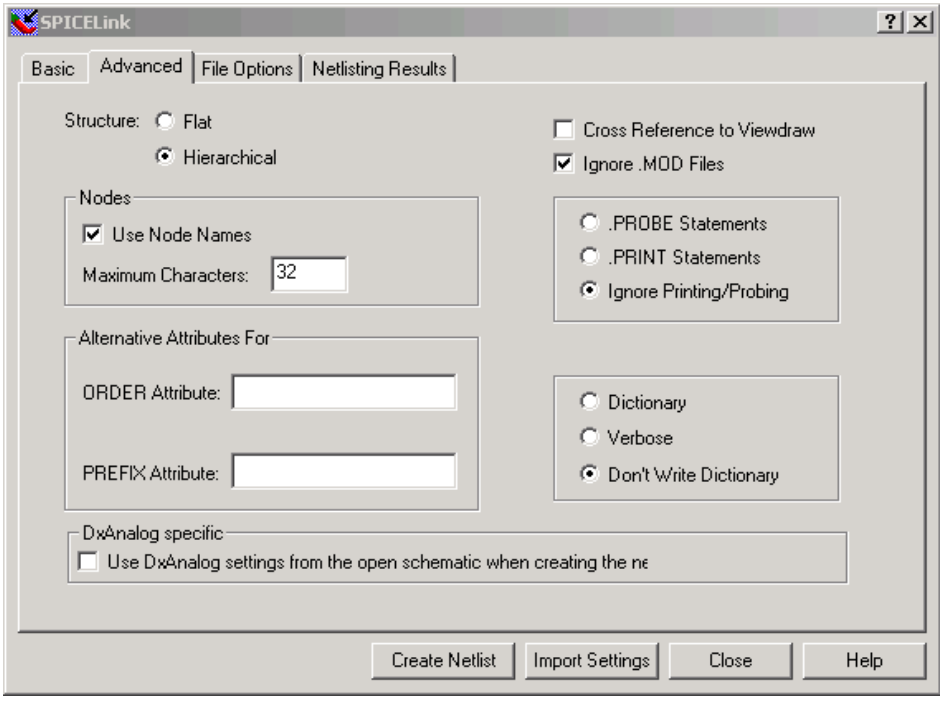


Figure 7: Advanced SPICELink settings. All of these options should be implemented exactly as listed here to insure a proper netlist.

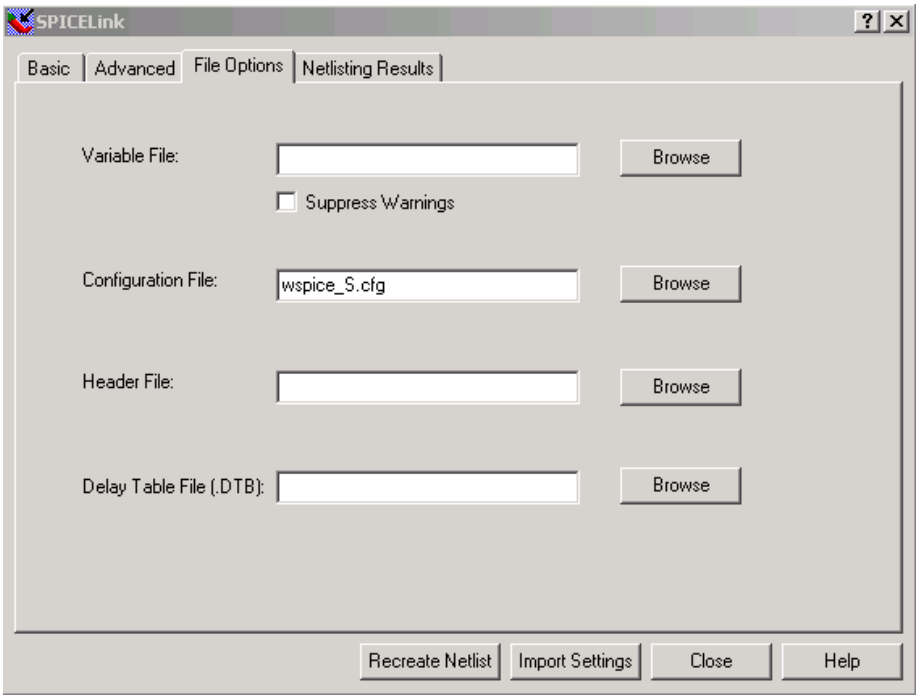


Figure 8: File options SPICELink settings. Make sure that the configuration file in the "`Smash`" directory is referenced.

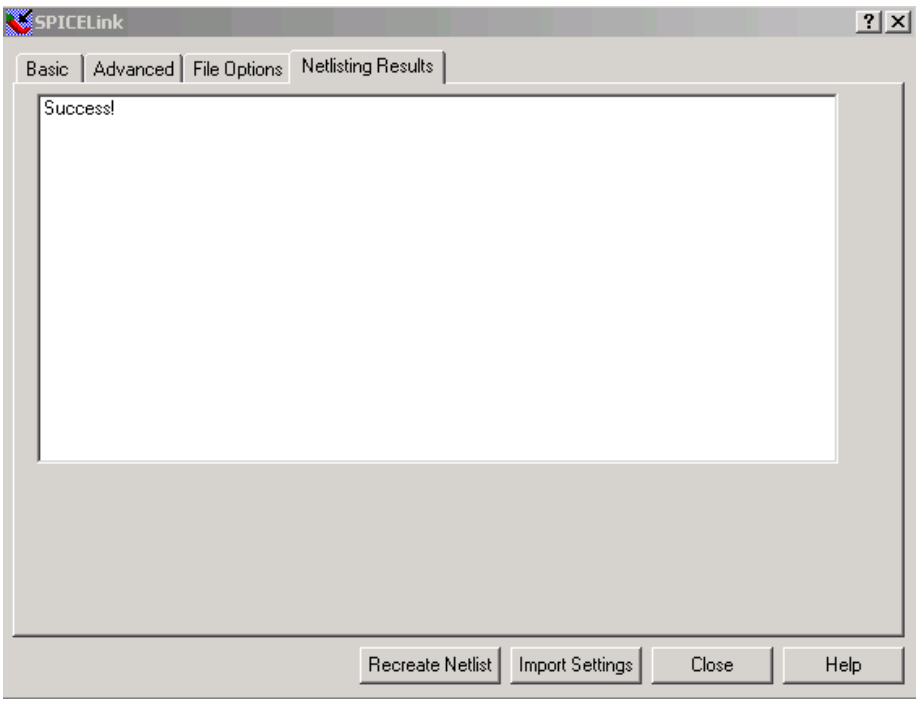


Figure 9: Netlisting results. Any error in the netlist will show up here. Correct the error(s) until the "Success!" message shows up.

5. SMASH Pattern and Netlist Files

The SPICELink netlist program produces a `*.cir` file that needs to be included in the `flash.nsx` file. In this design flow, the `*.nsx` file is the netlist file were all of the models, circuits, and Verilog files are included. The models and Verilog file can also be included in the `*.pat` file as library with the .lib directive as mention in § 9 Libraries of the SMASH™ Reference Manual.

The example `flash.nsx` file is showed below:

```
*ViewDraw Netlist Example For Smash

*Verilog File
`include "C:\simul\Viewnet\Schematic\S_Gates\logic.v"
.include "flash.cir"
.include "TSMC_0um35_N88Y_sim.mod"
```

Note:
You may need to modify the path of the `logic.v` file

For simplicity all of the files are included with `.include` and `` `include `` statements for all `*.cir`, `*.v`, and `*.mod` files. The model and Verilog files may be considered as (part of) the libraries (with `.lib` directive) for larger designs.

The `flash.pat` file contains all the input stimuli, output directives, analysis commands and global logic directives. The supply voltage was set to 3 volts. The logic directives were changed in this file to insure that the output logic was at the same level to drive the analog nodes correctly. For further modifications and clarifications of this file refer to the SMASH™ Manual.

The `flash.pat` file is showed below:

```
*flash.pat pattern file
*ViewDraw Netlist Example For Smash

*Analysis
.tran 1n 200n

*Clock generation
.param tp = '1/1e9' trf=1p pw = 'tp/2' dl1='pw-2*trf' tpn='tp-4*trf'
vclk1 CLK 0 pulse sup 0 0   trf trf pw tpn

*Input voltage
vinp VIN 0 sin 1 1 10e6 0 0 0

VDDA VDD 0 sup
.param sup = 3

*Logic Directives
.UNKZONE 1.4 1.6
.highlevel 'sup'
.lowlevel 0

.printall
.LPRINTall
.Trace  Tran   V(VIN) Min=-2.0000000E-001 Max=2.2000000E+000
.LTrace Tran   D1
.LTrace Tran   D0
.LTrace Tran   CLK
```

6. Mixed-Language Simulation

Now that all of the files have been created, the mixed-language simulation of the Flash ADC can be performed in SMASH™. Open the `flash.nsx` to allow SMASH™ to compile the files included in design. The `flash.pat` file is open together with `flash.nsx`. Once complied, run the transient simulation (from the menu: *Simulate* → *Transient* → *Run*).

Figure 10 shows the waveforms that should appear, showing the operation of the Flash ADC.

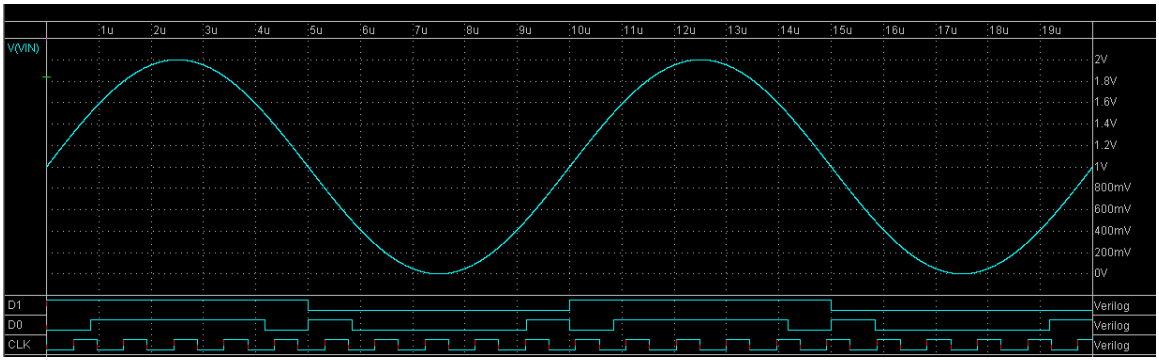


Figure 10: Results of FLASH ADC mixed-language simulation.

7. Cross-probing and Back-annotation

The flexibility in the presentation of the ViewDraw schematics allows for possibility of back-annotation and cross-probing with SMASH™ through Visual Basic Scripts (VBS) and other possible programs. Several of the interface programs in ViewDraw are VBS. The program SPICELink (described in section 4) is an example of such a script. Verlinet is a Verilog netlister included in the ViewDraw package that can be used to netlist a Verilog representation of logic blocks which could be included in a SMASH™ project. Verlinet is another example of a VBS. Several more programs are included and can be found in the Dashboard. A VBS in conjunction with a possible DLL could be designed to interface between SMASH™ and ViewDraw to allow back-annotation and cross-probing. First, a possible implementation concept is discussed for back-annotation. After which, the same is discussed for cross-probing.

Both ViewDraw and SMASH™ output text files for the schematic and operating-point (OP) information. In the text file for the ViewDraw schematic, changes in the file will update the schematic the next time it is opened. The proposed VBS might operate in the following way. When it is launched, the user would input the SMASH™ OP file name and the ViewDraw schematic file. The program would then search the schematic file for each node or device name listed in SMASH™ OP file. It would then add a line of text to the file that places text on the schematic below each node label with the voltage listed in the SMASH™ OP file. The same could be done for the device currents listed. The next time the file is opened, the new text of the operation point information would show up in the schematic. Special indicators could be used in the OP text information so that an erase common from the VBS dialog box could erase all of the OP information from the schematic. Also, the user could save the schematic as a new schematic to compare it to future simulations or document it. Hierarchy in the ViewDraw schematic would correspond to that given in the SMASH™ OP file which could be used to open the schematics lower in the hierarchy and continue the annotation processes.

Cross-probing might be somewhat more difficult in that is would require controlling the inner workings of SMASH™ and ViewDraw. The program might work as follows. Once the proposed program was enabled, it would add an extra attribute to the labeling function in ViewDraw that would allow cross probing for that node. When that function is selected, the program would then send a display request to SMASH™ for that node.

If the probing were to be done before simulation, a simple symbol could be added to schematic that would netlist a "`.Trace`" or "`.LTrace`" directive to the netlist which would cause the specified symbol to be included before the "`.Trace`" or "`.LTrace`" directives listed in the `*.pat` file. The SPICELink program requires that a prefix be added to the symbol in order to add it to the netlist. Therefore, a search and replace in the netlist would be needed to strip the netlist of the component value. An example symbol is given in the schematics directory called "`PROBE`". This symbol can be used to illustrate the concept. A screen capture of the flash schematic with the probe symbol is found in **Figure 11**. The required attributes are found in the dialog box. The netlist of this symbol appears as follows:

```
N1I38 .TRACE TRAN V(VIN)
N1I40 .LTRACE TRAN D0
```

The component names, "`N1I38`" and "`N1I40`" would have to be cleared in order for SMASH™ to understand the commands. However, a Simple script could be written to accomplish this and now the schematic is annotated.

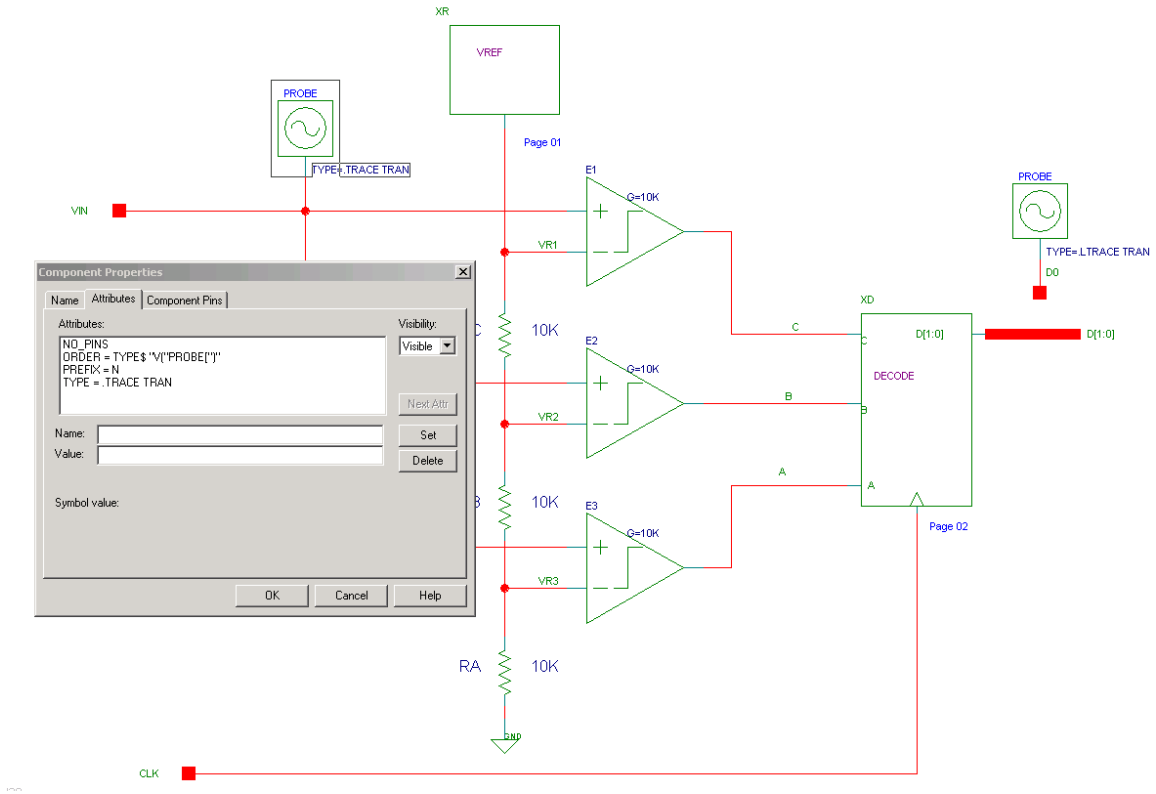


Figure 11: Probe symbol used to netlist .TRACE commands.

8. Conclusion

In this application note, a design flow was shown to use ViewDraw as a mixed-language schematic capture tool using an example of a trivial Flash ADC. The directories needed to generate the schematics were explained. Detailed instructions were given for generating schematics, symbols and netlists for simulations in SMASH™. The SMASH™ files were also explained and the simulation results shown. The example design flow in this document can easily be applied to more complicated mixed-signal and/or mixed-language designs.