Missing EDA Links

SLED 1.8

DOLPHIN INTEGRATION

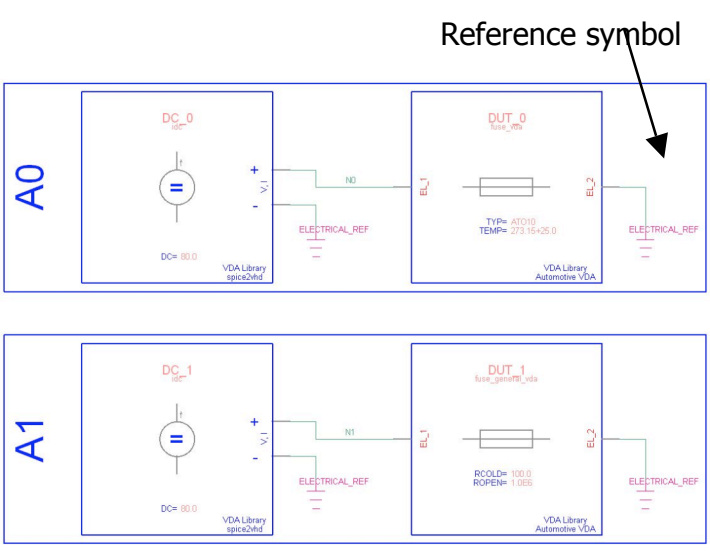
Multi-level and multi-domain modeling

Multi-level modeling is essential to be able to simulate complex subsystems in a reasonable amount of time and even to make simulations feasible. **SLED 1.8 focuses on delivering relevant features for smoother model management and browsing as well as for flexible mixed-signal and multi-domain netlisting.**

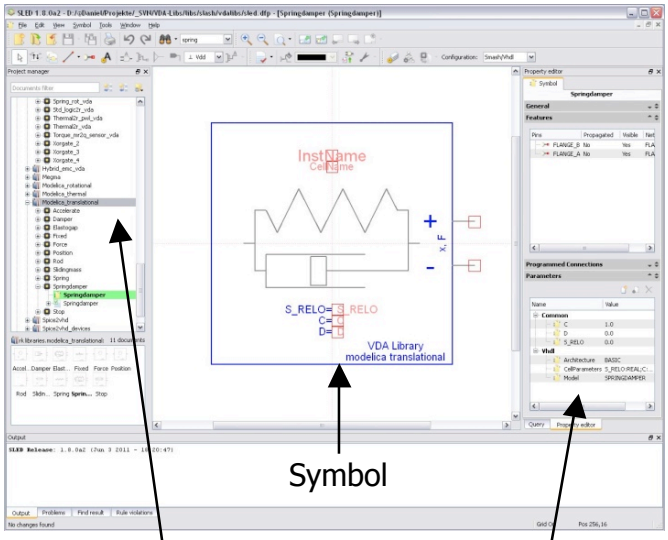
On top of that, this release of SLED also provides the means to set directives directly in the schematic in order to create more complete testbenches. The directives can be parameterized from the property editor and selectively enabled or ignored during netlisting depending on the design context.

KEY FEATURES

- ✓ Batch **scripting API** for automating design creation, check, migration and data search…
- ✓ **Management of directives** for simulation and layout from schematic
- ✓ Capability to check disciplines and bus expressions with the Design Rule Checker
- ✓ Automatic passing of parameters to instances when defined at schematic level
- ✓ Simplified pin order declaration
- ✓ **Improved capabilities and features of the Project Explorer**
  - Possibility to drag, copy, duplicate cells
  - Grouping of cells with tags
  - Renaming of designs, libraries, cells and views
- ✓ Improved bus creation with auto-incrementing of bus names
- ✓ Possibility to define schematic references locally using reference symbols
  - Electrical power supply / ground, mechanical reference…
- ✓ Improved Verilog netlisting through support of disciplines and management of references
- ✓ **Symbols for graphic structural assembly of models from the VDA Library**
- ✓ Increased interoperability with LayED (TexEDA Layout Editor) with XNDL netlist generation



Graphical assembly of a testbench using VDA library models



SLED is available identically under Linux and Windows.

Missing EDA Links

SLED 1.8

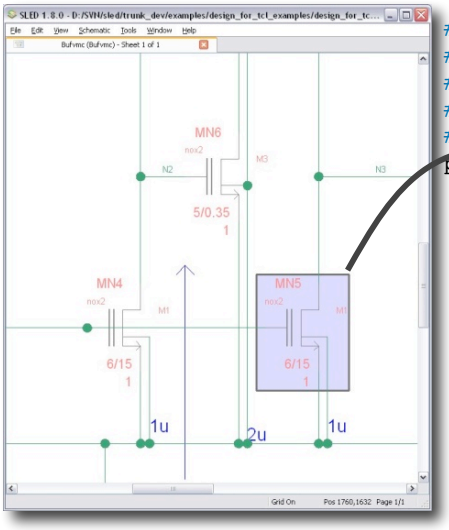
DOLPHIN INTEGRATION

Improved productivity with scripting API

SLED 1.8 provides a batch scripting API to automate repetitive tasks and increase productivity. Designers can save a lot of time by using the scripting API and can focus on creative tasks!

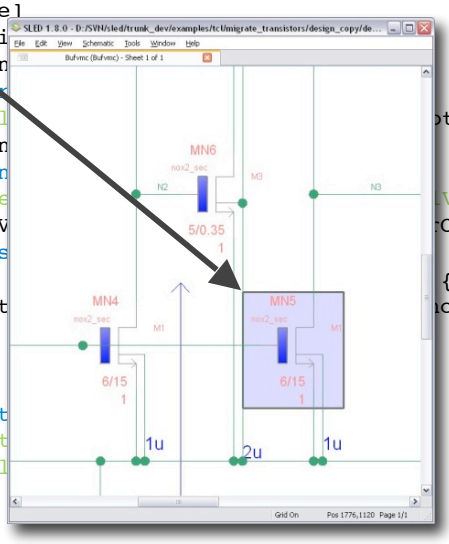
KEY BENEFITS OF THE SCRIPTING API

- ✓ **Speed**: Manual time-consuming tasks can be performed in a very short time with scripts
- ✓ **Reliability**: A well-written script reduces risks compared to manual changes.
- ✓ **Automation**: Scripts are the best ally for the automation of design flows.
- ✓ **Maintainability**: Scripts can be efficiently and explicitly commented. Even if comments are not written in the code, the user can easily understand what the script does by reading it; compared to manual actions that leave no trace.
- ✓ **Feasibility**: Some repetitive tasks, such as drawing matrices, can only be realistically performed by script.




```
## \brief Method to replace instances of the specified cell with the other spec
## @param inLibrary           the identifier of the library where to replace ins
## @param inCellNameToChange  the name of the cell which instances will be repla
## @param inNewLibrary        the identifier of the library where the replacing
## @param inNewCellName       the name of the cell replacing the previous
proc Migrate {inLibrary inCellNameToChange inNewLibrary inNewCellName} {

    # this pre-declaration is needed by tcl to use the enum type value SLED_Cel
    global SLED_CellViewType_SCHEMATIC
    # get the identifier of the new cell in the specified library
    set theNewCell [SLED_LibraryFindCell $inNewLibrary $inNewCellName]
    # define iterators
    set theIterCell [SLED_IterCellCreate]
    set theIterCellView [SLED_IterCellVi
    set theIterInstance [SLED_IterInstan
    # iteration on the cells in the lib
    for {SLED_IterCellBegin $theIterCell
        set theCell [SLED_IterCellConten
        # iteration on the schematics in
        for {SLED_IterCellViewBegin $the
            set theSchematic [SLED_CellV
            # iteration on the instances
            for {SLED_IterInstanceBegin
                set theInstance [SLED_It
                ...
        }
    }
    # do not forget to delete the iterat
    SLED_IterInstanceDelete $theIterInst
    SLED_IterCellViewDelete $theIterCell
    SLED_IterCellDelete $theIterCell
}
```



Example of a script which updates transistor instances to use symbols from a different library in order to migrate a design to another technology

Application examples of the scripting API are available in the example directory of SLED. They can serve as basis for your specific needs.
For more information, have a look at the User Manual!

SLED is available identically under Linux and Windows.

dolphin-integration.com/eda
solutions@dolphin.fr