A Guided debugging of EDA software with various components of Tcl/Tk GUI

Roshni Lalwani
roshni_lalwani@mentor.com

Amarpal Singh
amarpal_singh@mentor.com

Abstract

EDA software has various hardware design rule checks that can be debugged easily using schematic widget. The main objective of design rule checking (DRC) is to achieve a high overall yield and reliability for a hardware design. If design rules are violated the design may not be functional at all. This paper presents a flow of using some enhanced Tcl/Tk widgets in an innovate manner that can facilitate hardware designers in debugging various design issues of EDA tools.

1. Introduction

Our Tcl/Tk based GUI software provides a debugging environment to various EDA tools. It is built upon various widgets like schematic widget, dialog boxes, MTIwidgets etc. A schematic generator widget (Nlview) is a visualization software component that helps electronic design engineers to easily understand, debug, optimize and document electronic designs. A schematic window in a Tcl/Tk GUI is a simplified graphical representation of an electrical circuit .The schematic diagram consists of instances, pins and nets that are graphical representation of hardware design netlist. The schematic widget supports a number of features to navigate the user to **interesting parts** of the logic and to present engineering information in relation to the schematic. The Schematic Generator is not intended to extract any engineering data from the netlist - but is designed to generate a schematic as a "**skeleton"** for presenting these data. This implies the need of an engineering system that "feeds" Nlview with data. The data is provided by various EDA tools via our GUI interface. The interface between Schematic Generator and Our Tcl/Tk based GUI is defined by string based API (Application Programming Interface). These APIs provides a simple set of commands, callbacks and configuration properties and makes it easy to visualize and debug the EDA software backend data.

There are certain attributes associated with each HDL components/objects like instances, pins and nets. The idea here is to display this information in the callout box on various objects of the schematic window, in such a way that it will help the user in debugging the problematic design issues. A callout box in schematic window is sticky tag visually associated with each object in the schematic window. A callout box is a nothing but a pixmap formed by few rendering shape and text rendering APIs, so it is very fast and efficient. The call out box is a light weight object

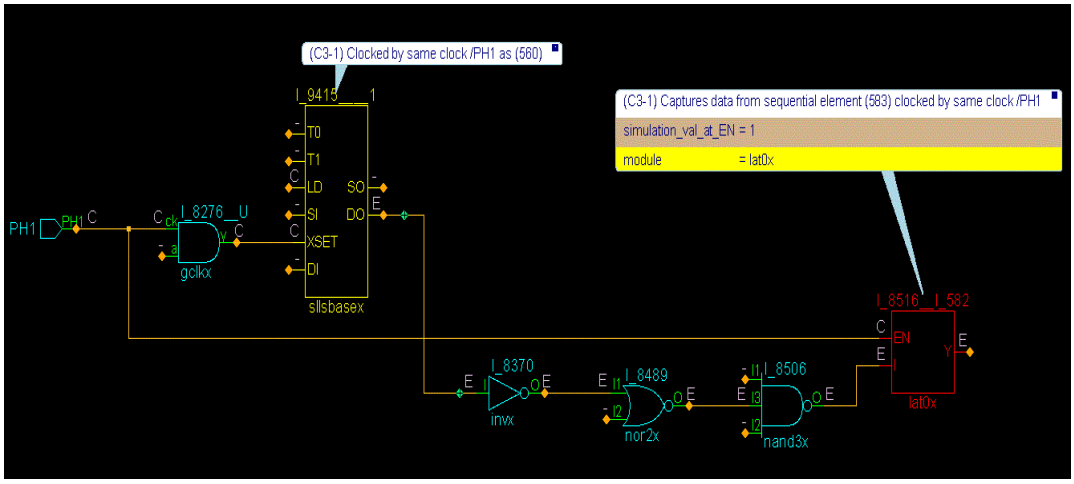
that can be displayed on any object in the schematic. It displays the text that gives a user a way to solve the DRCs and other design issues. The callout box is integrated in our Tcl/Tk GUI using Tcl/Tk interface provided by NLview widget.

Section 2 below covers callout box, highlighting how callout box helps in debugging design rules checks and **Section 3** outlines the enhanced dialogue box and its integration with the callout box functionality.

2. Guided debugging using Callout Box

A design rule error is often associated with one or two instances. There is an error instance where rule error occurs and there is source instance that is starting pointing for the error. For example, there is design rule error which reports a wrong simulation value at the input pin of error instance. The incorrect simulation value is because the source instance and error instance are clocked by same clock. The tool also displays callout box on source and error instance. The message displayed in callout box is an intuitive step to debug and resolve the design rule error. There are also attributes associated with source and error instances. This information is also displayed on the source and error instances in the same callout box.

For example the schematic view with callout box is as follows.



2.1. Callout box Integration in with GUI tool

The schematic generator component integrated with our GUI tool provides an API based mechanism to attach various kinds of attributes with Schematic objects like instances, pins nets. These attributes can be visual or only non-visual in nature. The various kind of visual attributes are like object's display name, object's border color, object's fill color

and object's line style (solid/dashed/thick/thin etc). Outbox is also one special kind of visual attribute attached to Nlview objects where application can along with specifying the text to be shown in an outbox, configure the outbox for its background color, foreground color and color of its various regions. Mostly, this whole configuration information about how an outbox should be rendered is provided via some options during setting outbox on an object.

Here is a simple API interface to demonstrate how an outbox is attached to a Schematic object and its configuration mechanism.

The add_outbox command allows user to add one or more outbox on objects.

add_outbox object_id -name n? ?-value value? ?-bgcolor n? ?-textcolor n? ?- -colorlist <string>? ?-separatorcolor n? ?-crosscolor n? ?-deltaX x? \
?-deltaY y?

The object_id addresses the data base object, one of:inst, net, netBundle,port, portBus,pin, pinBus, hierPin or hierPinBus.
Please note:

Option **-bgcolor** <number> option specifies the color of outbox region. (default value is 1)
For ex : -bgcolor 2 specifies that the color of outbox region will be taken from the property outboxcolor2.

Option **-textcolor** <number> option specifies the color of text for that particular outbox. (default value is 0)
For ex : -textcolor 1 specifies that the color of text for outbox will be taken from the property outboxcolor1.

Option **-crosscolor** <number> option specifies the color of cross for that particular outbox. (default value is 4)
For ex : -crosscolor 1 specifies that the color of cross for outbox will be taken from the property outboxcolor1.

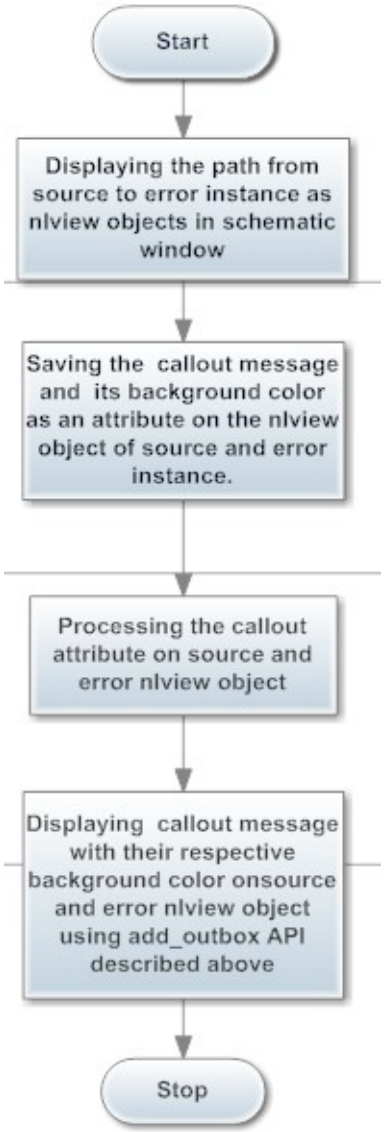
Option **-colorlist** <string> option specifies the in order list of colors of regions for that particular outbox. (by default colorlist is empty string)
For ex : -colorlist "3 5 4" specifies that the color of the first region of the outbox will be taken from the property outboxcolor3, color of second region from property outboxcolor5 and color of third region from property outboxcolor4.

Option **-deltaX** <number> specifies the horizontal shift.

Option **-deltaY** <number> specifies the vertical shift.

The text displayed in the callout box guides the user to resolve the problematic area of EDA design.

2.2. Algorithm for schematic view with callout box



The following pseudo code depicts the example usage of Nlview TCL API for displaying the instances in schematic window with callout box.

proc analyzeDrc { } {

- *Add objects to schematic window*
- *Add callout specific attributes to same objects*
- *Display the objects and its associated attributes in the schematic window.*

}

3. Section2 : Enhance TCL/TK dialogue box

There is an enhanced TCL/TK dialogue box that is used to modify the background color of the text displayed in the callout box. The user can add/delete the text from the callout box using this dialogue box. The dialog box and mainly consist of a combo box, two list boxes, an option menu and Add/Remove button.

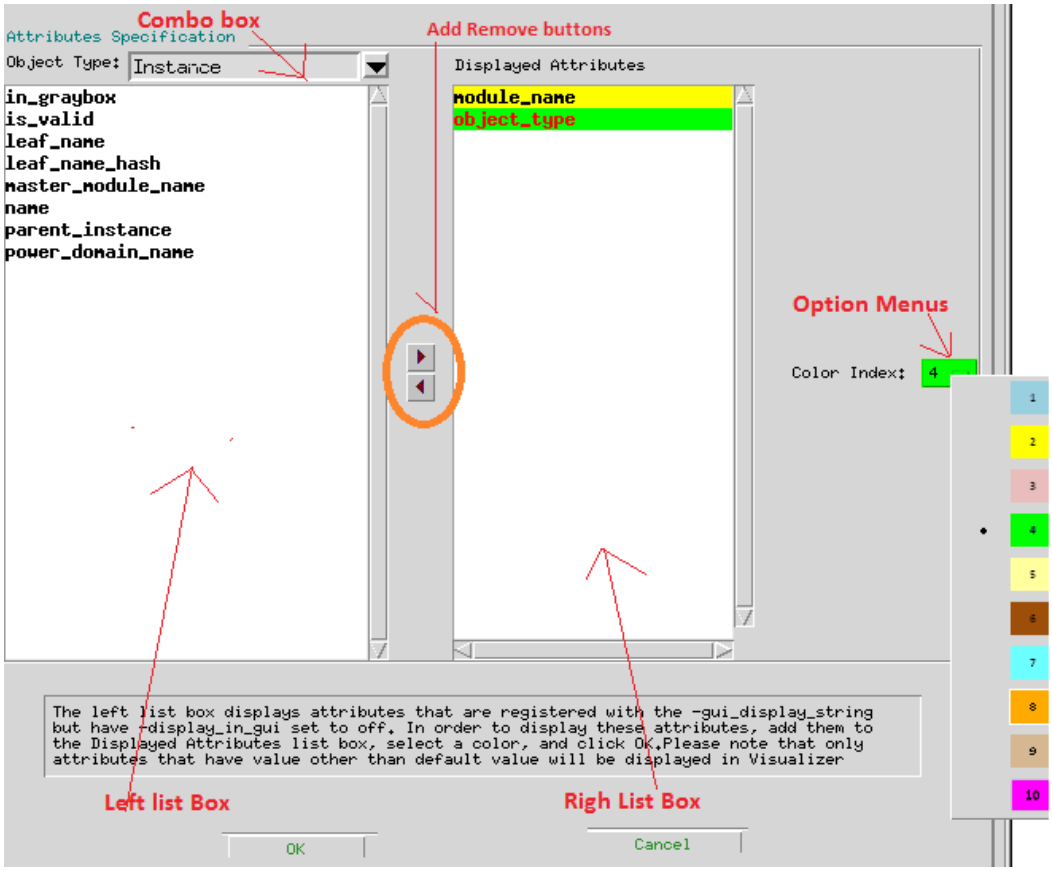


Figure2: An innovative dialogue box

3.1. Creation of Enhanced Dialogue box

The dialog box is also enhanced and mainly consists of a Combo-box, two list boxes, an option menu and Add/Remove buttons.

3.1.1. Combo box

The combo-box box is constructed using IWidgets combo-box .The user can select the category from Instances/Pins/Nets by using combo box and the respective attributes gets displayed in left /right list box.

3.1.2. Two list boxes

The two list boxes are scrolled list boxes and are constructed using list box and scroll bar of Tk widget.

3.1.3. Option menu

The Option menu is constructed using tk_Option Menu.

3.1.4. Text Message

The text message area is constructed using text widget of TK.

3.1.5. Add/Remove button

3.1.6. OK/Cancel button

The Add/Remove OK and Cancel buttons are constructed using button widgets of TK.

All the items are arranged in the grid using grid of TK.

3.2. Integration of Enhanced Dialogue box with Callout box functionality.

The user can select the category from Instances/Pins/Nets by using combo box and the respective attributes gets displayed in left /right list box. The attributes which are displayed in right list box gets displayed in callout box. The remaining attributes, displayed in the left list box are associated with the object type but are not displayed in the callout box .The user can add/remove attributes from left/right list using Add/Remove buttons. The respective changes will get applied to the callout box. The user can also modify the background color of the text displayed in the callout box by selecting a color in the option menu. These changes will also get applied to the callout box displayed in the schematic window.

3.3. Pseudo code for Integration of Dialogue Box with Callout Box

proc modifyCalloutBoxMessage { } {

1. *Creating individual widgets like combo box, scrolled listboxes and buttons and arranging them in grid.*
2. *The user can select an item from Pin/Instances/Nets from Combo-box , and the attributes of the same will be displayed in left and right list box*
3. *The user can also modify the background color of the attribute by selecting the appropriate color from the tk_option menu*

   OR

   The user can add/delete the attributes from left to right list box to display/remove the attributes from callout box.

   OR

   The user can execute both the steps.
4. *When the user press Ok button , these changes will be reflected in the callout box*

}

4. Glossary

GUI: Graphical User Interface.

HDL: Hardware description language.

DRC: Design Rule Checks.

5. Summary

Thus we display the DRC error information to user in an intuitive way. We can also modify the background color of the callout box using enhanced TCL/Tk dialogue. Thus we provide

a guided debugging environment to the user by implementing and using the enhanced TCL/ Tk widgets.

6. Bibliography

TCL wiki, http://wiki.tcl.tk