ShyHLS: a tiny framework for high-level synthesis and optimization

Antonio Cipolletta, Andrea Calimera

Last Revision March 31, 2020

Contents

1 Preface

The following document is the reference manual for the *ShyHLS CAD Tool* developed for the course of *Synthesis and Optimization of Digital Systems* (02LVNOV, 02LVNOQ) at Politecnico di Torino. The tool has been specifically designed for educational purposes. It recalls the internal structure and the naming convention of other commercial synthesis tools used during the course, yet with a simple and user-friendly interface. The aim is to help students learning High Level Synthesis (HLS) methods and algorithms, and hence to get confidence with Electronic Design Automation (EDA) tools in general. With ShyHLS, the students will be able to implement their own synthesis algorithms, quickly and easily, avoiding long set-up and learning phases.

ShyHLS is written in C but it offers a TCL interface. TCL stands for Tool Command Language, one of the most adopted scripting language in the field of EDA frameworks.

The next sections give a short yet exhaustive description of the main features of the tool, the key available commands and their synopsis.

2 Setup

ShyHLS has been tested on a workstation with Ubuntu 18.04 as operating system and tclsh8.6 as TCL shell.

The following command is used to load the shared library `libShyHLS_sods.so` into a TCL shell:

```
load <path-to-the-library>/libShyHLS_sods[info sharedlibextension]
```

Once the loaded all the commands become available to the users.

3 Reference Flow

The entry point for ShyHLS is a formal description of the dataflow graph (DFG) using DOT format[1]. The DOT file is read and translated into a standard graph datastructure inside the tool. The RTL library specification is provided in a plain text file. Each line of the file describes a functional unit composing the library by defining the operation performed and enumerating its figures of merit: delay, area and power consumption. This text file is read using a TCL command and translated into a datastructure inside the tool.

[1] Wikipedia Page of DOT language

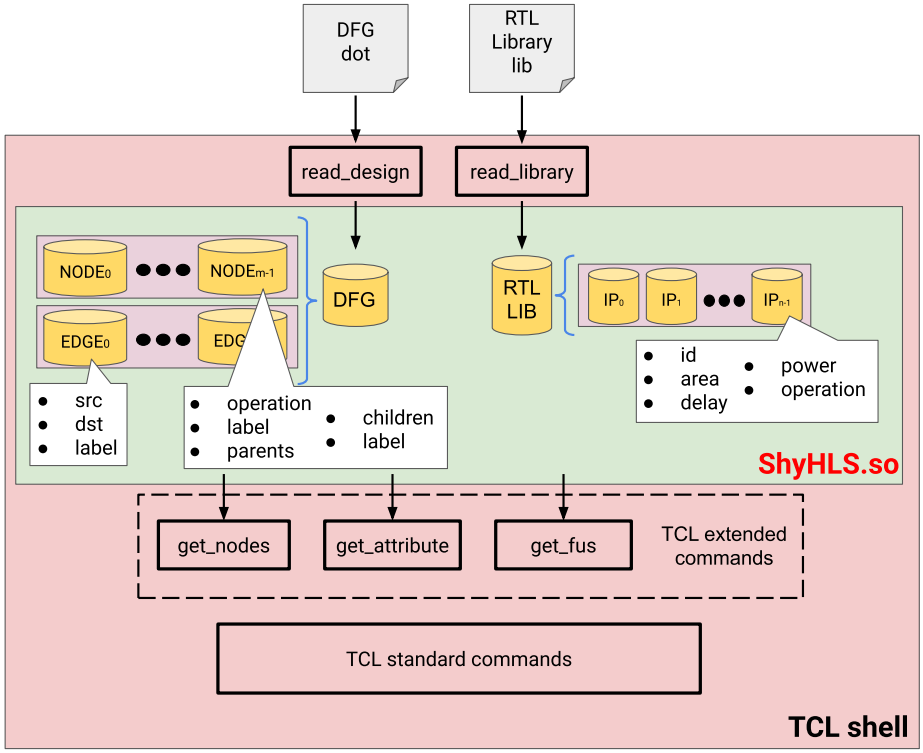

Figure 1: Abstract view of the framework.

4 Commands

4.1 DFG

4.1.1 read_design

```
read_design path-to-the-dot-file
```

Return Value: None
Description: this command is used to load a DOT file representing a DFG inside the tool. Only one design could be active at a time. Indeed, trying to load more than one design results into an error.

4.1.2 remove_design

```
remove_design
```

Return Value: None

Description: this command is used to unload the current DFG. Trying to remove a design when it has not been loaded results into an error.

4.1.3 get_nodes

```
get_nodes
```

Return Value: List of node ids.
Description: this command is used to retrieve all node Ids of the current DFG. The order in which the ids are returned is random.

4.1.4 get_sorted_nodes

```
get_sorted_nodes
```

Return Value: List of node ids in topological order.
Description: this command is used to retrieve all node Ids of the current DFG in topological order. A topological sort or topological ordering of a directed graph is a linear ordering of its vertices such that for every directed edge uv from vertex u to vertex v, u comes before v in the ordering. In particular, in case of a DFG this means that all the fan-in cone of a node Nx comes before Nx in the ordering.

4.1.5 get_edges

```
get_edges
```

Return Value: List of edge ids.
Description: this command is used to retrieve all edges of the current DFG. The order in which the edge ids are returned is random.

4.2 RTL Library

4.2.1 read_library

```
read_library path-to-the-library-file
```

Return Value: None
Description: this command is used to parse the textual description of a library of components and load it inside the tool. Only one library could be available inside the tool at a time. Indeed, trying to load more than one library results into an error.

4.2.2 remove_library

```
remove_library
```

Return Value: None
Description: this command is used to unload the current library. Trying to remove a library when it has not been loaded results into an error.

4.2.3 get_lib_fu_from_op

```
get_lib_fu_from_op operation
```

Return Value: the id of the library functional unit that can execute operation or null string if there is no functional unit in the loaded library that can perform the specified operation.
Description: this command is used to retrieve the id of the functional unit that is able to perform an operation.

4.2.4 get_lib_fus

```
get_lib_fus
```

Return Value: List of library functional unit ids.
Description: this command is used to retrieve all library functional unit ids of the current library. The order in which the ids are returned is random.

4.3 Object Attribute

```
get_attribute object attribute_name
```

Return Value: depends on the object and the attribute requested. See the following subsections for the description of the various attributes of each object type.
Description: Returns the attribute requested from the specified object.

4.3.1 list_attribute

```
list_attributes class_object
```

Return Value: List of attributes available for the specified object class.
Examples:

```
list_attributes node
list_attributes edge
list_attributes lib_fu
```

4.3.2 Node Attributes

Attribute Name	Description
label	label of the node
id	id
operation	operation
n_parents	number of inputs
n_children	number of nodes receiving as input the output of this node
parents	list of ids of the parent nodes
children	list of ids of the children nodes

Table 1: Attributes available for node object

4.3.3 Edge Attributes

Attribute Name	Description
label	label of the edge
id	id
src	id of the src node
dst	id of the dst node

Table 2: Attributes available for edge object

4.3.4 Library Functional Unit Attributes

Attribute Name	Description
id	id
operation	operation performed by the library functional unit
area	area
delay	delay
power	power
n_in_ports	number of input ports

Table 3: Attributes available for library functional unit object

4.4 Visualization Utilities

Description of commands used to plot graph in DOT language after scheduling.
The DOT linux cli utility can be used to render the .dot file.

```
dot -T{ps|pdf} <path-to-dot-file>/in_filename.dot > out_filename.{ps|pdf}
```

4.4.1 print_dfg

```
print_dfg filename
```

Return Value: None.
Description: Specify the filename.

4.4.2 print_scheduled_dfg

```
print_scheduled_dfg schedule filename
```

Return Value: None.
Description: Specify the filename and a list containing couples (node id, start time).