HoCL v1.1
User manual

J. Sérot

Chapter 1

Using the HoCL compiler

The compiler HoCL is invoked with a command like :

```
hoclc [options] file1 ... filen
```

where `file1`,...,`filen` are the names of the file(s) containing the source code (by convention, these files should be suffixed `.hcl`).

The complete set of options is described in Sec. **??**.

The set of generated files depends on the selected target. The output file `hocl.output` contains the list of the generated file.

1.1 Generating graphical representations

Example :

```
hoclc -dot main.hcl
```

The previous command generates a graphical representation of the graph(s) described in file `main.hcl` in `.dot` format[1]. Each toplevel graph (defined as `graph ... end`) and refined node (defined as `node ... struct ... end` or `node ... fun ... end`) gives a separate `.dot` file.

1.2 Generating XDF representations

Example :

```
hoclc -xdf main.hcl
```

The previous command generates a graphical representation of the graph(s) described in file `main.hcl` in `.xdf` format.

TODO: To be documented

1.3 Generating DIF representations

Example :

```
hoclc -dif main.hcl
```

The previous command generates a graphical representation of the graph(s) described in file `main.hcl` in `.dif` format.

TODO: To be documented

[1] `http://www.graphviz.org`.

1.4 Generating SystemC code

Example :

```
hoclc -systemc main.hcl
```

The previous command generates the SystemC code corresponding the dataflow graph described in file `main.hcl`. The following files are written :

- a pair of files `x_act.h`, `x_act.cpp` for each actor declared in the source file, containing respectively the interface and the implementation of the actor,
- a file `x_gph.h` for each defined graph (either as a toplevel graph or a refined node),
- a file `main.cpp` containing the toplevel description and driver for simulation.

The produced files can then compiled using the standard SystemC toolchain. When compiling (resp. linking) the HoCL-specific headers (resp. library) must be available[2]. Examples of `Makefiles` are provided in the `examples` sub-directories of the distribution.

[2]These headers and library are located in `$HOCL/lib/systemc` where `$HOCL` points to the installation directory of the HoCL toolset.

Chapter 2

Compiler options

General options

-stdlib	set location of the standard library file
-no_stdlib	do not use the standard library
-prefix	set prefix output file names (default is main source file basename)
-target_dir	set target directory for generated files (default is current directory)
-dump_tenv	dump builtin typing environment (for debug only)
-dump_typed	dump typed program (for debug only)
-dump_senv	dump builtin static environment (for debug only)
-dump_ir	dump intermediate representation (for debug only)
-dump_boxes	dump static representation of boxes
-insert_bcasts	insert broadcast boxes
-version	print version of the compiler
–v	print version of the compiler

DOT-specific options

-dot	generate .dot representation of the program
-dot_rank_dir	set rank direction for DOT output graph (default: LR)
-dot_unlabeled_edges	do not annotate graph edges
-dot_no_io_rates	do not annotate ports with resp. rates
-dot_show_indexes	print box and wire indexes
-dot_slotted_boxes	print boxes with i/o slots

SystemC-specific options

-systemc	activate the SystemC backend
-sc_stop_time	stop after n ns
-sc_clock_period	set clock period (ns) (default: 10)
-sc_default_fifo_capacity	set default fifo capacity (systemc only) (default: 256)
-sc_trace	set trace mode
-sc_dump_fifos	dump fifo contents
-sc_trace_fifos	trace fifo usage in .vcd file
-sc_dump_fifo_stats	dump fifo usage statistics after run
-sc_fifo_stats_file	set file for dumping fifo statistics (default: fifo_stats.dat)

XDF-specific options

-xdf	generate .xdf representation of the network
-xdf_package	set package name for the generated XDF code

DIF-specific options

-dif generate .dif representation of the program