Unleashing the Power of the Command Line Interface

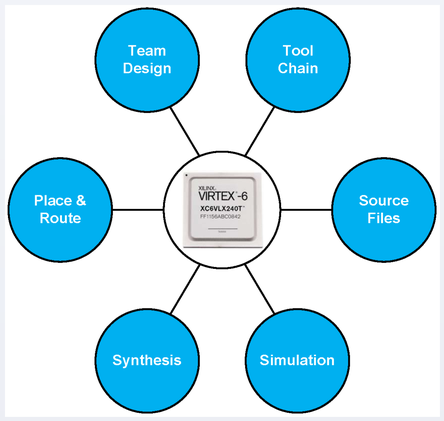
Jeremy W. Webb

UC Davis, VLSI Computation Laboratory

Abstract

The development of complex ASIC or FPGA designs involving multiple teams and loosely integrated tools is an arduous process. There is an inherent challenge in maintaining coherency and separation of source and generated files throughout the build process, particularly through different tool versions and vendors. These aspects of the development process make results hard to reproduce, reuse, and share. This paper highlights the benefits of a command-line-based build environment as an alternative to using graphical user interfaces (GUIs) for RTL development. A well-reasoned directory structure for projects is proposed, as well as a template for command-line integration of ASIC or FPGA development tools.

Motivation



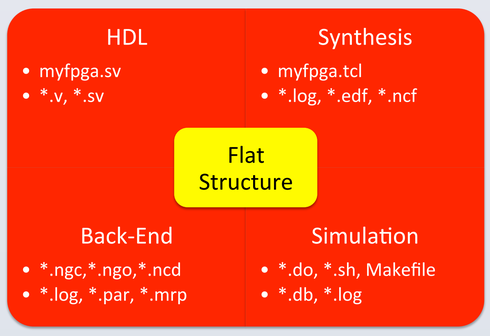
Team Design

- Directory Paths
  - Integrated Development Environments (IDEs) typically use absolute paths
  - Command-line design flow uses relative paths
    - Portable design environment for design team
- Revision Control Software
  - Distribute design throughout team
  - Some RCS tools provide a mechanism for ignoring intermediate files

Directory Structure: Design File Types

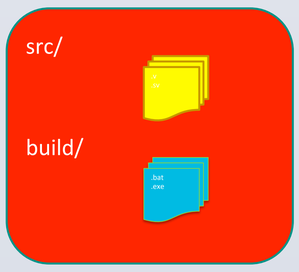
- HDL Modules
  - Multiple HDL files, IP Core Files
- Synthesis Files
  - Tcl Scripts, Project Files, Log Files, Netlists
- Back-End Tool Files
  - Project Files, Log Files, Netlists, Configuration Files
- Simulation Files
  - Project Files, Waveform Files, Database Files

Directory Structure: Flat Structure



Directory Structure: Semi-Hierarchical Structure

- Separate directories for:
  - Source Files
  - Build Files
- Scripts and generated files collocated in build directory making it difficult to distinguish between the two



Directroy Structure: Hierarchical Structure

- Defined location for files improve efficiency
  - source files
  - synthesis builds
  - place and route builds
  - simulation projects
  - other miscellaneous files
- Directory hierarchy generation can be automated with scripts

Directory Structure: Hierarchical Example

```
myfpga/  Top level directory of a design.
  cfg/  Configuration files.
  doc/  Specification and Design documents.
  misc/ Put datasheets, IP core generation, etc.
  src/  Put RTL source here.
  syn/  Synthesis directory.
  par/  Build directory.
```

```
src/  Put RTL source here.
  <modulename>/
    doc/  Module related documentation.
    sim/  Top level simulations directory.
    syn/  Synthesis directory.
    sw/   Software or script files.
```

```
syn/  Synthesis directory.
  bin/  Synthesis scripts.
  log/  Log files (including reports).
  out/  Generated EDIF files.
  run/  Running synthesis scripts.
  src/  Constraint files.
```

```
par/  Build directory.
  archive/  Build archives.
  bin/      Build scripts.
  log/      Log files (including reports).
  out/      Generated netlists and config files.
  run/      Running build scripts.
  src/      Constraint files.
```

Synthesis Flow

- Synplify Pro is executed in batch mode using a Tcl project file via a Makefile
- Tcl project file uses relative paths to source files located in ../src/ directory
- A script automates generation of source file list from Tcl project file for the Makefile
- Tcl callbacks (synhooks.tcl) copy EDIF and NCF files to the build directory ../par/run/

Synthesis Flow: Synthesis Makefile

```
PROJNAME  := myfpga                                  <- Set Project Name

# Source Code:                                       <- Create Source File List
SRCS      :=$(shell ../bin/parsetcl.sh $(PROJNAME))

# Environment Variables:                             <- Enable Callback Functions
export SYN_TCL_HOOKS=../bin/synhooks.tcl

synthesize : $(SRCS)                                 <- Perform Synthesis
 @echo "$(SRCS)"
 ../bin/outarch.sh $(PROJNAME) ../log ../out ../run
 synplify_pro -batch ../bin/$(PROJNAME).tcl
```

Synthesis Flow: Example Synthesis Build

```
[jwwebb@darthbane ~]
$ cd ~/snug/git/myfpga/par/bin/
[jwwebb@darthbane ../git/myfpga/par/bin]
$ make setup
Executing: make setup
[jwwebb@darthbane ../git/myfpga/par/bin]
$ make synthesize
Executing: make synthesize
../../src/myfpga/myfpga.sv ../../src/in_buf/
in_buf.sv ../../src/out_buf/out_buf.sv
Loading ../bin/synhooks.tcl

Running proj_1|log
TCL script complete: "../bin/myfpga.tcl"
```

FPGA Implementation Flow

- The FPGA implementation flow:
  - Netlist Translate
  - Mapping
  - Place and route
  - Timing Analysis
  - Configuration File Generation
- Makefile automatically places report and log files in a log directory and leaves intermediate files in the run directory
- Build is performed by three scripts located in the ../myfpga/par/bin/ directory:
  - Makefile
  - par.xilinx.mk
  - outarch.sh
- Build is performed in the ../myfpga/par/run/ directory in order to contain generated files

Summary

- The suggested directory structure, and use of command-line interface scripts and Makefiles, can improve the FPGA or ASIC design efficiency and promotes a team design flow.
- The design flow is controlled such that all files generated by both the design team and the tools are stored in a known location.
- The FPGA design flow parallels an ASIC design flow.