Cycle Model Studio

Version 9.0.0

Command Line Compilation Flow Application Note

Non-Confidential

ARM DUI 1042A

Cycle Model Studio
Command Line Compilation Flow Application Note

Release Information

The following changes have been made to this document.

Change History

Date	Issue	Confidentiality	Change
November 2016	B	Non-Confidential	Restamp release with 9.0.0

Non-Confidential Proprietary Notice

Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by ARM and the party that ARM delivered this document to.

Product Status

The information in this document is final, that is for a developed product.

Web Address

http://www.arm.com

Purpose

This guide supplements the *Cycle Model Studio (CMS) User Manual* (ARM DUI 0958), which describes the flow of creating a SoC Designer Component using the CMS GUI. Familiarity with Cycle Model Studio is assumed.

This document describes an alternate, command-line-based flow to creating a Cycle Model, which allows you to more efficiently manage your projects as they evolve. With a command-line-based flow in place, changes to the RTL can be executed via scripting without needing to launch the GUI.

Running the SoC Designer GUI initially sets up the externally-visible interface of the model. After this point, the procedure in this document allows you to save time by recompiling the RTL and putting the same interface (the wrapper) on the updated model.

***Note**: The flow in this document is intended for situations in which you are making minor updates to the RTL source files. If you make changes using the modelstudio GUI (after the initial creation of the SoC Designer model), or if your updates affect transactors, do not use this flow as there is a risk of over-writing your updates.*

***Note:** This Application Note is for Linux compilation only and does not address Windows or VHDL compilation.*

Introduction

References in this document to `$TARGET` refer to the name of your design, which must be specified at the start of the flow. Replace references to `$TARGET` with the name of your design (typically the top-level module name) as you proceed through the flow in this document.

At a high level, creating a Cycle Model from RTL is a two-step process:

- The first compilation produces the `$TARGET.a` file which is a basic C model of the RTL lacking any transactor interfaces.
- The second compilation produces the `$TARGET.so` file which can be loaded in SoC Designer via a `.conf` file.

The flow described in this document is as follows:

1. Produce the `$TARGET.a` file though a cbuild command.
2. Use the Cycle Model Studio GUI to produce Makefiles and the `lib$TARGET.ccfg` skeleton files.
3. Run the ARM carmgr program and `make` from the command line to generate the final `$TARGET.so` file.

Detailed Steps For Initial SoC Designer Component Creation

Running the SoC Designer GUI initially sets up the externally-visible interface of the model. The steps for initial SoC Designer component creation, described in this section, are:

1. Use GUI to create the Cycle Model Studio Project
2. Finalize SoC Designer Component from Cycle Model Studio GUI
3. Generate SoCDesigner Component Compilation Files
4. Final Compilation
5. Setup
6. Create “$TARGET.a”
7. Use GUI to Create ModelStudio Project
8. Finalize SoC Designer Component from Cycle Model Studio GUI
9. Generate SoC Designer Component Compilation Files
10. Final Compilation

Setup

The example tar ball, `CMS_cmdline_flow.tgz`, is set up for a simple module called "ahb_mem." If you do not already have `CMS_cmdline_flow.tgz`, request it from your ARM Application Engineer.

1. Extract the `CMS_cmdline_flow.tgz` to an empty directory.
2. Modify `cbuild.sh` to set `$TARGET` to the name of your design.
3. Specify Verilog compiler options in the `$OPT` in `cbuild.sh`. Potential options are:
   - `-sverilog` - Enables SystemVerilog compilation mode.
   - `-directive tieoffs.dir` - Specifies a cbuild separate directives file. This is NOT a general Verilog directives file.
4. Create a `filelist_$TARGET.f` file (for example, `filelist_ahb_mem.f`), which contains your typical Verilog simulator file list and options. Specify Include and Library directories before listing the individual RTL files; for example:

```
+incdir+./rtl
-y /home/library/shared
./rtl/ahb_mem.v
```

Create "$TARGET.a"

Run `cbuild.sh` to perform the initial cbuild compilation of your RTL and generate the initial `$TARGET.a`. The cbuild.sh script is performed with the following options:

```
cbuild –v2k –f $FILE_LIST –o $TARGET.a
```

The script does not check for cbuild errors, and it deletes intermediate files and logs which can be helpful for debug. If you need to see these files and logs then comment out the last `rm` command in the `cbuild.sh` script.

Use GUI to Create ModelStudio Project

1. `cd` to the directory `MODELS/$TARGET` and invoke `modelstudio`.
2. Select **File > Close Project** if Cycle Model Studio opens the last active project automatically.
3. Create a new project by selecting **File > New > Project**. The New Project dialog box opens.
4. Make sure "Project from existing database (`.symtab.db io.db`) is selected.
5. In the Name field, change "Project1" to the name of your design ($TARGET).
6. Ensure the Location field lists the `MODELS/$TARGET` directory.
7. Deselect "Create Directory for Project." Creating a directory is not required, because you have already created the necessary directories in the previous steps. Click **OK**.
8. Navigate to the directory `MODELS/$TARGET/Linux/Default` and click on the `lib$TARGET_.symtab.db` file. Click **OK**.
9. Select **Project > Create SoC Designer Component**…

Finalize SoC Designer Component from Cycle Model Studio GUI

Note: Using the Cycle Model Studio GUI to make changes results in Makefiles and other files being overwritten. You must save and restore them manually.

1. Using the Cycle Model Studio GUI, add transactors, parameters, and customize your SoC Designer component at this time. Refer to the *CMS User Manual* for more information.
2. Click **Save All** to save these setup files. This creates the following files:
   - `MODELS/$TARGET/$TARGET.carbon`. This is the modelstudio project file. You can open this project file in the modelstudio GUI in the future to make changes to how your SoC Designer component is created.
   - `MODELS/$TARGET/Linux/Default/Makefile.carbon`. Note that this overwrites any previous version, so be sure to save this file if you have modified it.
   - `MODELS/$TARGET/Linux/Default/SoCDesigner/lib$TARGET.ccfg`. Note that this overwrites any previous version so be sure to save this file if you have modified it. This

file holds the majority of the information about customizing the SoC Designer component, such as how transactors are attached to ports.

3. Exit the Cycle Model Studio GUI.

Generate SoCDesigner Component Compilation Files

1. cd to `MODELS/$TARGET/Linux/Default/SoCDesigner`.
2. Run `carmgr lib$TARGET.ccfg`. This adds many files to this directory; the most important are:
   - `Makefile.lib$TARGET.mx`
   - `mx.$TARGET.cpp`
   - `mx.$TARGET.h`

   Note that this command overwrites any previous versions of these files, so be sure to save any versions you would like to keep. Some of these files have pragmas (`//USER CODE`) and code that is within those pragmas is retained.

Final Compilation

From the directory `MODELS/$TARGET/Linux/Default/SoCDesigner` run:

```
make VHM_DIR=.. -f Makefile.$TARGET.mx
```

This compiles and creates the SoC Designer component. To use this component within SoC Designer (sdcanvas and sdsim), load the `maxlib.$TARGET.conf` with the `-b` option. In addition, the `.so` file must be present.

Re-running the Flow for ECOs

RTL Changes

1. Update `filelist_$TARGET.f` for any changes to paths and to add or remove RTL files. If the only RTL changes are within existing files, then you can skip this step. RTL-only changes using this flow should not result in any interesting intermediate files being overwritten.
2. Re-run `cbuild.sh` to create an updated `$TARGET.a`.
3. Skip Use GUI to Setup Final Compilation and Generate SoC Designer Component Compilation Files as these steps do not need to re-run.
4. If you are not making changes to the SoC Designer component, then run Final Compilation. Do not over-write any of the Makefiles. If changes to the SoCDesigner component are going to be made continue to the next section and don't Final Compilation now.

SoC Designer Component Changes

The flow described in this section is for implementing changes to a SoC Designer component, such as adding transactors to RTL ports or configuring parameters. This is typically done from the Cycle Model Studio GUI and ***WILL*** result in Makefiles and other files being overwritten.

1. Invoke `modelstudio` from the `MODELS/$TARGET` directory and close any project that automatically opens.
2. Select **File > Open > Project** and open the `MODELS/$TARGET/$TARGET.carbon` project.
3. Double-click on `lib$TARGET.ccfg` (on the left pane) to open SoC Designer component configuration.
4. Make your desired changes to the SoC Designer component.
5. Go to the section Finalize SoCDesigner Component from ModelStudio GUI and continue with those steps. **Note that this will overwrite the files mentioned in that step.**
6. Follow the remaining steps from the original flow, Generate SoC Designer Component Compilation Files and then Final Compilation.