3 Getting Started

3.1 Installation

RapidSmith2 is available on Github at: https://github.com/byuccl/RapidSmith2. You can either build RapidSmith2 into .class and .jar files for use in any Java environment, or configure RapidSmith2 to work in an IDE (recommended).

3.1.1 Requirements for Installation and Use

- Windows, Linux or Mac OS X all will work (see additional notes below for Mac OS X)
- Vivado 2016.2. Later versions of Vivado may work, have not been tested yet. Earlier versions will not work.
- JDK 1.8 or later
- Tincr

Tincr is a companion project (https://github.com/byuccl/tincr) which is used for importing/exporting designs between Vivado and RapidSmith2. Installation instructions for Tincr can be found at the repository linked above. For getting started (running the example programs on the provided sample designs) you will not need it installed. Later, as you actually start processing your own Vivado designs you will need to obtain and install it. There are additional dependencies beyond these required for installation, but they are either provided in the distribution itself or are automatically retrieved for you as a part of the installation process. Examples of these additional dependencies include QT Jambi and the BYU Edif Tools.

3.1.2 Steps for Installation

1. Clone the RapidSmith2 repository at https://github.com/byuccl/RapidSmith2. If you are not familiar with GitHub, you will need to install Git on your computer, and run the following command in an open terminal:

   ```
   git clone https://github.com/byuccl/RapidSmith2
   ```

   This will copy the RapidSmith2 repository into a local directory.

2. Create a new environment variable called RAPIDSMITH_PATH, and point it to your local repository of RapidSmith2 that you setup in step (1). This is needed so RapidSmith2 can find required device files and other items at runtime.

3. Build the RapidSmith2 project. RapidSmith2 is managed using a gradle build system. To build the project, navigate to your local repository of RapidSmith2 and execute one of the following scripts in a terminal:

   ```
   ./gradlew build (unix)
   gradlew.bat build (windows)
   ```

   The build process could take a few minutes.

4. At this point, you have two choices: set up RapidSmith2 for use in an IDE, or run RapidSmith2 from the command line. Both choices are detailed below.

   Running from an IDE The gradle scripts in RapidSmith2 currently support setup for both Eclipse and IDEA Java IDEs. This section will detail how to setup the Eclipse environment, but similar steps can be taken for IDEA. If using Eclipse, you must use Eclipse version Neon or later. To create a new eclipse project, execute one of the following in a terminal:

```
gradlew antlr eclipse (unix)
gradlew.bat antlr eclipse (windows)
```

Executing these will create an Eclipse *.project* file. After the project file has been created, you can import the project into Eclipse by opening Eclipse and selecting:

```
File->Open Projects From File System
```

and pointing it to your RapidSmith2 local repository. All Java source files will be found under *src/main/java*. **NOTE:** Your RapidSmith2 git repository should not be put inside your eclipse workspace. It is better to put it elsewhere, and then import it into your workspace.

Building on the Command Line After step (3) in the installation process, gradlew produces everything that you will need to run RapidSmith2 from the command line. The following directories are created:

- *build/classes/main*: This folder contains the RapidSmith2 class file directory tree.
- *build/libs*: This folder contains a Jar file of the RapidSmith2 class files.
- *build/distributions*: This folder has both .zip and .tar files with contains all Jars needed to run RapidSmith2 from the command line. This includes a full jar of the RapidSmith2 build alond with copies of dependency Jars (such as QT-Jambi).

After adding the appropriate .class files or Jars to your `CLASSPATH`, you should be able to run RapidSmith2 tools from the command line. If you make any changes to the RapidSmith2 code, you will have to rebuild before running the program again (Step 3). **CAUTION:** An obvious thing to try is to mix and match developing in Eclipse but then running the resulting apps from the command line. Just be aware that Eclipse puts its compiled .class files in very different places than where the command line gradle build process puts its .class and .jar files. Make sure you understand that before you try to combine these two build/execution methods. Our suggested approach is to choose one or the other, but not both.

3.1.3 Additional Notes for Mac OS X Installation

The instructions above require you to set the `RAPIDSMITH_PATH` environment variable. If running from the command line, the environment variables can be added to your *.bash_profile* file as in any other UNIX-like system. However, if using an IDE such as Eclipse you either need to define the environment variable for every Run Configuration you create, or you need to add the `RAPIDSMITH_PATH` definition system-wide in OS X. This can be done, but how to do so differs based on what OS X version you are running (and seems to have changed a number of times over the years). Search the web for instructions for how to do so if you desire. **Hint**: you will likely have to edit some *.plist* files.

3.1.4 Running RapidSmith2 Programs

Some points to keep in mind while configuring and running RapidSmith2 programs:

- The RapidSmith2 code base contains a number of assertions which may be helpful as you are developing code. These are not enabled by default in Java. To enable them, add *-ea* as a VM argument. This is highly recommended.
- If you are running on a Mac, when running RapidSmith2 programs that use Qt (any of the built-in programs like **DeviceBrowser**) that are GUI-based, you will need to supply an extra JVM switch, *-XstartOnFirstThread*.
- A common error when running RapidSmith2 programs is failing to have your `RAPIDSMITH_PATH` defined. If this is the case when you try to execute a program, an `EnvironmentException` will be thrown telling you that you forgot to set the variable.

- If you are running on Windows, only a 32-bit QT Jar file is included in the RapidSmith2 repository. This means that you will need to set your JRE to a 32-bit version when running the GUI programs.
- For Linux command line usage, the `CLASSPATH` environment variable must point to both the full (uncompressed) RapidSmith2 jar in the *build/distributions* folder as well as all the jar files in the */lib* subdirectory. An example `CLASSPATH` could look like this:

```
RAPIDSMITH2-SNAPSHOT/*:RAPIDSMITH2-SNAPSHOT/lib/*
```

3.1.5 Testing Your Installation

At this point you can test your installation by executing the java **DeviceBrowser** program:

```
java edu.byu.ece.rapidSmith.device.browser.DeviceBrowser
```

This can be done either from within Eclipse or from the command line, depending on how you are running RapidSmith2 (if running under OS X be sure to provide the *-XstartOnFirstThread JVM argument*. If all goes well you should see a graphical representation showing the details of a physical FPGA device as shown in Figure 3. You may initially be zoomed far in and might want to zoom out to see the entire chip layout.

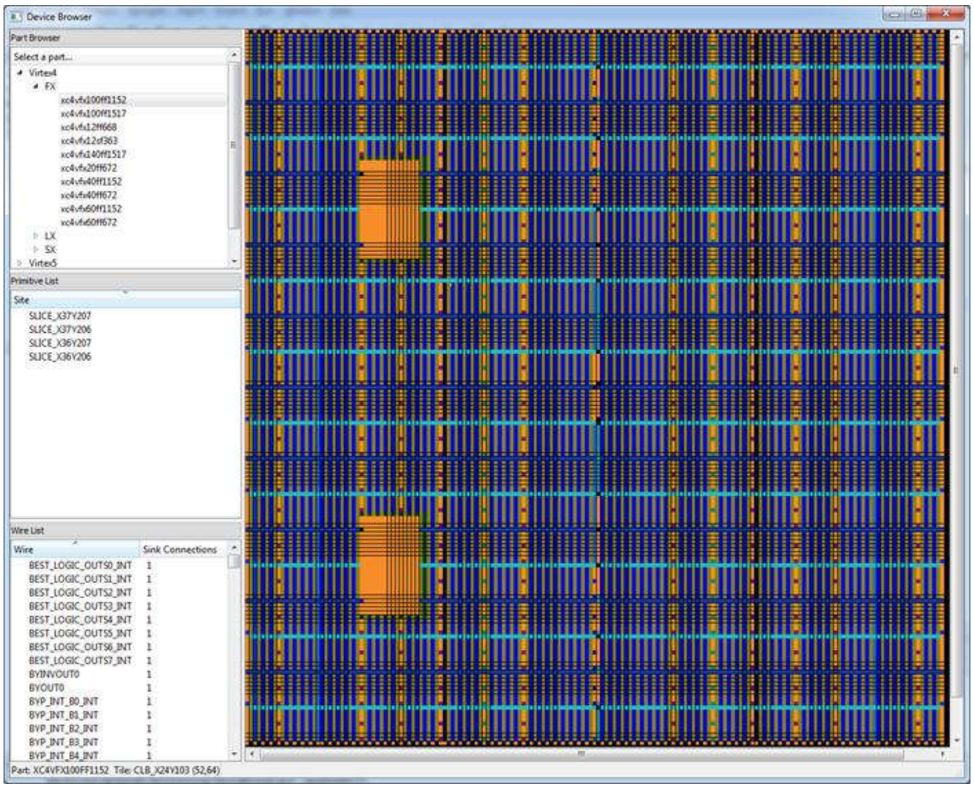

Figure 3: DeviceBrowser Sample Display

3.2 Running Real Designs - An Overview

This section will lead you through the required steps to export an implemented design from Vivado and load it into RapidSmith2. It will also demonstrate how to export a modified design from RapidSmith2 back into Vivado to complete implementation. Specifically, in this example you will fully synthesize, place, and route a simple Vivado FPGA