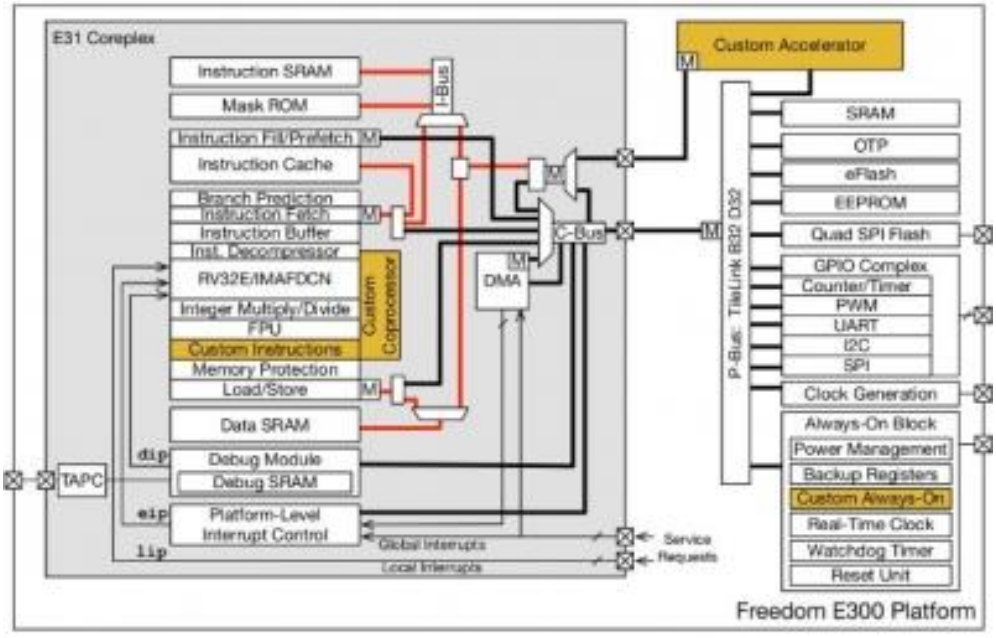
Running a RISC-V Processor on the Arty A7

The Arty A7-100T contains a Xilinx XC7A100T FPGA which is the largest FPGA available for the Arty A7 and is ideal for deployment of softcore processors. These processors can be either proprietary or open source. One of the most popular open source processors is the RISC-V. This tutorial covers building a RISC-V processor, specifically the SiFive Freedom E310. This guide steps through the process of loading the Freedom E310 onto an Arty A7, and programming it using the Arduino IDE.



Top-Level Block Diagram of the E300 platform, © 2017 SiFive Inc

---

Inventory

- Arty A7-100T
- Olimex ARM-USB-TINY-H USB Programmer
- Vivado 2017.1 – Webpack Edition
- Arduino Development Environment
- 10 Flying leads to connect between the programmer and one of the Arty's Pmod connectors, which is connected to the JTAG Test Access Port of the processor
- A Linux development or virtual machine is needed to compile the processor, generate the bitstream and upload applications to the processor.
- Install the following on the Linux development machine:

1. Git - the following command can be used:

   ```
   sudo apt-get install git
   ```

2. Device Tree Compiler - the following command can be used:

   ```
   sudo apt-get install device-tree-compiler
   ```

3. Java Run Time Environment - the following command can be used:

   ```
   sudo apt-get install default-jre
   ```

4. JAVAC installed - the following command can be used:

   ```
   sudo install openjdk-8-jdk
   ```

5. Ensure the JAVA_HOME environment variable is set

Getting Started

Download and install version 2017.1 of Vivado. Once downloaded, open a terminal window. Then for this tutorial, change directory (cd) to the home directory. This terminal window will be used to build the RISCV processor.

Set up Vivado by sourcing the settings script in the terminal command line, using the following command:

```
source /opt/Xilinx/Vivado/2017.1/settings64.sh
```

Note: *If using a 32-bit Linux environment, source the settings32.sh script instead.*

Download and include the Digilent's board files so that Vivado can use them. Use git to download the board definition files from the Digilent repository using the following command:

```
git clone https://github.com/Digilent/vivado-boards.git
```

Once downloaded, copy the 'new' board files into the appropriate Vivado directory. This can be achieved with the following command:

```
sudo cp -r vivado-boards/new/board_files/*
/opt/Xilinx/Vivado/2017.1/data/boards/board_files/
```

Now it is possible to generate the processor and the FPGA implementation.

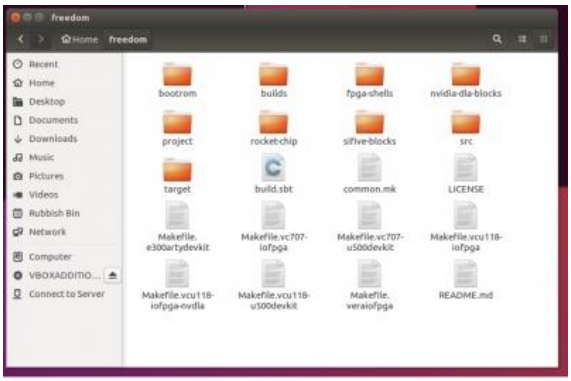
Building the RISC-V

Generation and implementation of the processor can be done by running Makefiles.

First, download the SiFive freedom processor by using git and cloning its repository:

```
git clone --recursive
https://github.com/sifive/freedom.git
```

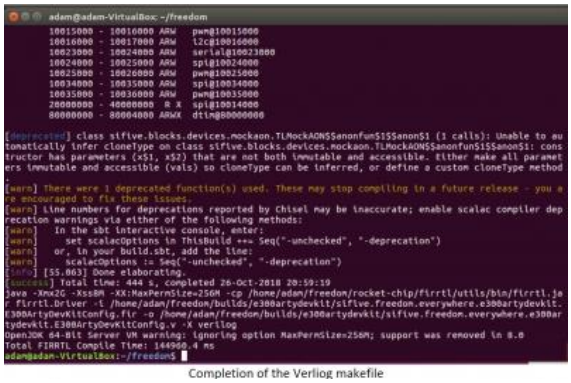
Cloning will take a little while due to a large number of files to download. Once downloaded, a new folder called “Freedom” can be seen in the working directory.



Freedom Directory contents once cloned

Within this directory, there are several Makefiles. This tutorial will be using the Makefile.e300artydevkit example. The example script will generate the RISC-V processor for both the Arty A7-35T and Arty A7-100T. The toolchain must be compiled first. In the terminal window, change the working directory to the toolchains directory within rocket-chip/riscv-tools and run the build.sh script. This will build the toolchain needed for generating the implementation files. To generate the Verilog instantiation of the processor from the Chisel HDL files, issue the following command:

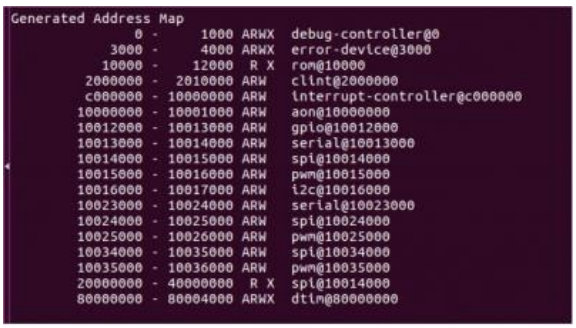
```
make -f Makefile.e300artydevkit Verilog
```



Completion of the Verliog makefile

Compiling the files will take a little time to complete. Generating generic Verilog files will allow either Arty A7-35T or Arty A7-100T to be targeted.

In the terminal window, the peripherals and their locations in the memory map can be seen.



Freedom RISC-V generated memory map for the example

With the Verilog description of the processor available, the next step is to decide which of the Arty boards will be targeted. No changes need to be made if targeting the Arty A7-35T. If targeting the Arty A7-100T, then some minor adjustments need to be made. To make these changes, open the makefile Makefile.e300artydevkit using a text editor and change board type. By default, the Makefile will generate the FPGA implementation for the Arty A7-35T.

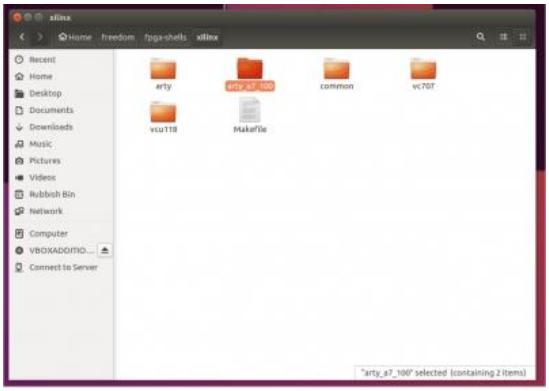
```
# See LICENSE for license details.
base_dir := $(patsubst %/,%,$(dir $(abspath $(lastword $(MAKEFILE_LIST)))))
BUILD_DIR := $(base_dir)/builds/e300artydevkit
FPGA_DIR := $(base_dir)/fpga-shells/xilinx
MODEL := E300ArtyDevKitFPGAChip
PROJECT := sifive.freedom.everywhere.e300artydevkit
export CONFIG_PROJECT := sifive.freedom.everywhere.e300artydevkit
export CONFIG := E300ArtyDevKitConfig
export BOARD := arty
export BOOTROM_DIR := $(base_dir)/bootrom/xip

rocketchip_dir := $(base_dir)/rocket-chip
sifiveblocks_dir := $(base_dir)/sifive-blocks
VSRCS := \
    $(rocketchip_dir)/src/main/resources/vsrc/AsyncResetReg.v \
    $(rocketchip_dir)/src/main/resources/vsrc/plusarg_reader.v \
    $(sifiveblocks_dir)/vsrc/SRLatch.v \
    $(FPGA_DIR)/common/vsrc/PowerOnResetFPGAOnly.v \
    $(BUILD_DIR)/$(CONFIG_PROJECT).$(CONFIG).rom.v \
    $(BUILD_DIR)/$(CONFIG_PROJECT).$(CONFIG).v

include common.mk
```

Default build script for the Arty 35T

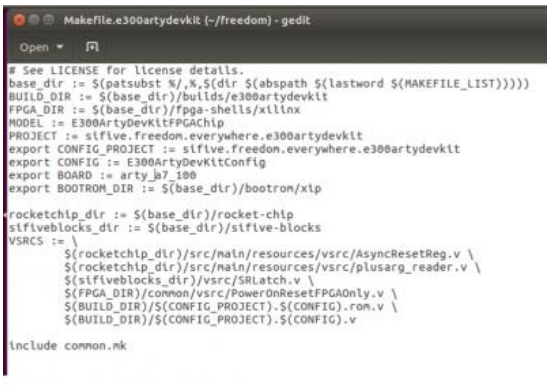
To generate an implementation for the Arty A7-100T, change the board type to arty_a7_100T. Open a file browser and navigate to the freedom/fpga-shells/Xilinx directory. Several folders, named for each supported development board, can be seen here. Note that the name of the arty_a7_100 folder matches the board type used.



Supported Xilinx development files

Back in the text editor with the Makefile, change the board name to the arty_a7_100.

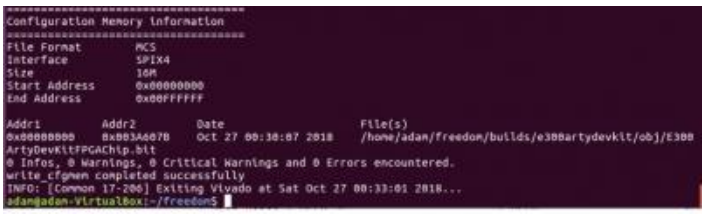
Save and close the modified file.



Updating the board definition for the Arty 100T

With that, the FPGA implementation can now be generated. In the terminal window issue the following command:

```
make -f Makefile.e300artydevkit mcs
```



Completion of the FPGA Implementation Script.

This will take time, as the script will use Vivado and the RISC-V toolchain to generate both a bitstream and an MCS file. The MCS file can be programmed into the flash memory on the Arty A7 100T and load the RISC-V processor when the board is powered on.

The output files will be available under the directory:

```
<workspace>/freedom/builds/e300artydevkit/obj
```

The BIT files, MCS files, and Vivado implementation reports can all be found here.

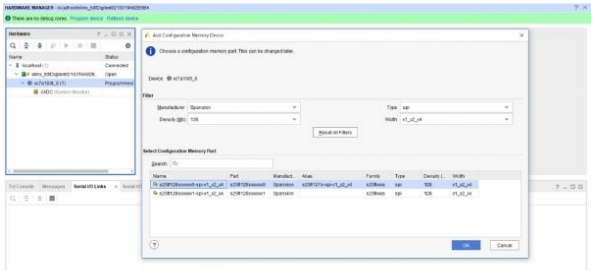
Vivado Utilisation Report

Programming the Hardware

With the programming file available, the next step is to program the flash memory using Vivado.

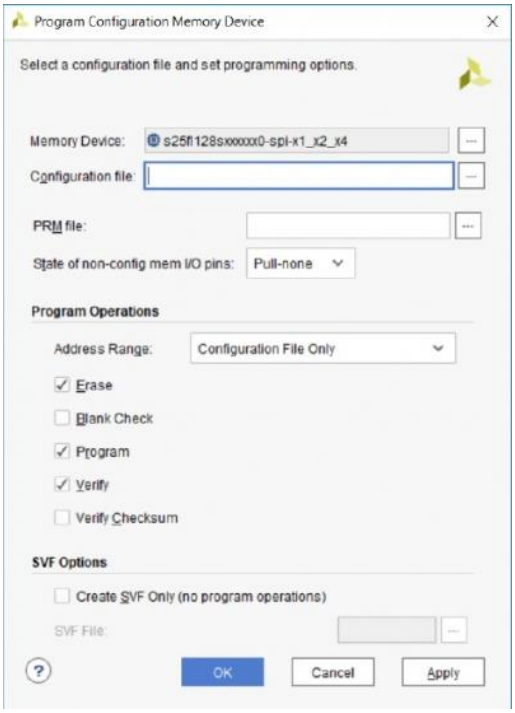
Open Vivado using the command: “Vivado”.

Once the GUI starts, the next step is to connect the Arty A7-100T to the development machine via the micro USB port (J10), and to open Vivado's Hardware Manager. Once the board has been connected, the hardware tab on the left of the screen will show the FPGA device. Select the device, then right click on it and select *Add Configuration Memory Device*. This will open a dialog box where the memory type can be selected. The Arty A7-100T has a Spansion S25FL128xxxxx device fitted.

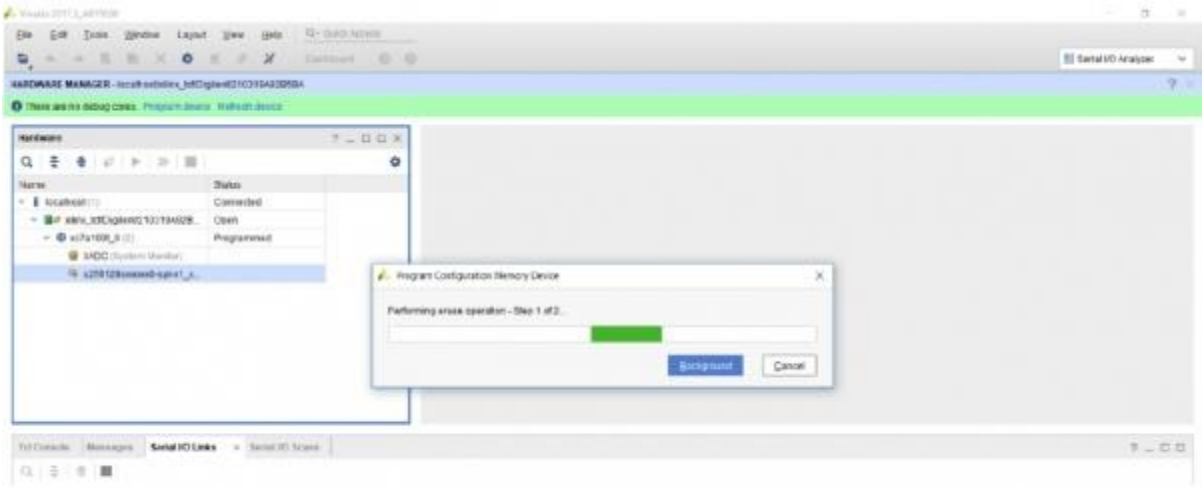


Selecting the flash for programming.

After selecting the device, a dialog prompt will ask whether the device should be programmed now. Click **Yes** and another dialog will open.



Select the generated MCS file, and the device will be programmed. Once this process is complete, the Arty A7-100T will be running the RISC-V processor. Check that the Arty A7's jumpers are configured so that it can be programmed from flash and press the reset button.



Confirm that the processor is running by doing the following:

1. Press button 3 and see LED 6 go out
2. Press the reset button and see LED 4 go out

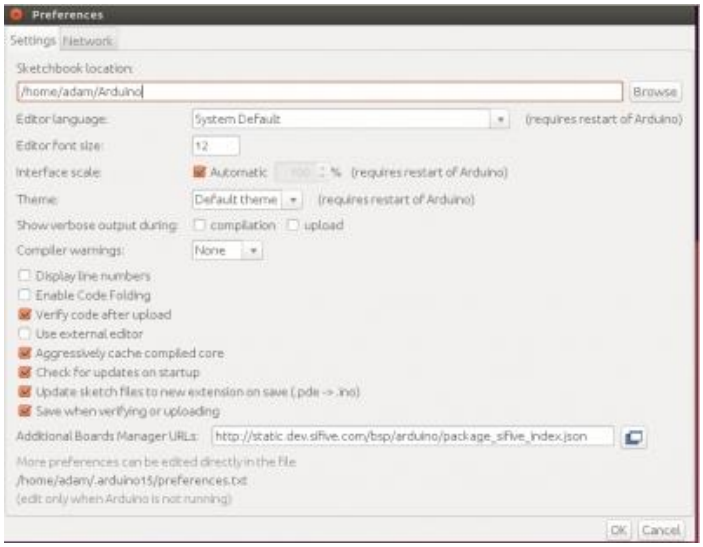
However, a custom application still needs to be created and uploaded. This can be achieved by using the Arduino development environment.

Generating Software

First though, the Olimex JTAG pod needs to be connected to the Arty A7's Pmod Port D, allowing programs to be downloaded and debugged using the JTAG interface. The pin out between for connecting the two can be found here.

With that, a custom application can now be developed. The best way to get started programming the RISC-V is to use the Arduino development environment.

Installing support for the SiFive Freedom processor is easy. Under “File → Preferences”, point the Additional Boards Manager URL to the following URL:




```
http://static.dev.sifive.com/bsp/arduino/package_sifive_index.json
```

The next step is to install the board using the Board Manager found through the “Tools → Boards” menu.

In the Board Manager dialog, select the **Contributed** type to find the SiFive Freedom Boards.

Once this is installed, applications can be developed, and examples applications loaded, just the same as for any other board in this environment. Just remember to select the Freedom Arty Dev Kit from the board selection list.

Troubleshooting

1. Ensure that the Xilinx cable drivers are installed. This can be achieved by first changing the directory to

```
<xilinx
Install>Vivado/2017.1/data/xicom/cable_drivers/lin64/install_script/install_d
rivers
```

Then running the command:

```
sudo ./install_drivers
```

2. Ensure that the Olimex USB has the required permissions. Edit the following file:

```
/etc/udev/rules.d/99-openocd.rules
```

By adding the following:

```
# These are for the Olimex Debugger for use with E310 Arty Dev Kit
SUBSYSTEM=="usb", ATTR{idVendor}=="15ba", ATTR{idProduct}=="002a",
MODE="664",
GROUP="plugdev"
SUBSYSTEM=="tty", ATTRS{idVendor}=="15ba", ATTRS{idProduct}=="002a",
MODE="664",
GROUP="plugdev"
```

Save and enter the command sudo udevadm control –reload-rules

3. If problems occur while uploading, check the connections between the Olimex and Pmod Port D, and ensure that the user is a member of the plugdev group.

Next Steps

For more guides on how to use the Arty A7, visit the device's Resource Center.

For more information on Vivado, visit Digilent's Vivado tutorials.

For technical support, please visit the FPGA section of the Digilent Forums.