

Integration of functional block to Shakti

Introduction to Shakti

Shakti is an Open Source processor development program based on RISC-V ISA. There are many classes of processors in the Shakti ecosystem with varying compute capabilities.

There are a couple of different classes of processors namely E-Class and C-Class processors.

Processors

E-Class Processor:

The E-Class is a 32/64 bit micro-controller capable of supporting all extensions of RISC-V ISA. The E-class is an In-order 3 stage pipeline having an operational frequency of less than 200MHz on silicon. It is positioned against ARM's M-class (CorTex-M series) cores.

The major anticipated use of the E-class of processors is in low-power compute environments, automotive and IoT applications such as smart-cards, motor-controls and home automation.

C-Class Processor:

The C-class is an in-order 6 stage 64-bit micro controller supporting the entire stable RISC-V ISA. It targets mid-range compute systems running over 200-800MHz. It can also be customized upto 2 Ghz. The C-class is highly customizable and there are variants for low-power and high-performance. It is positioned against ARM's Cortex A35/A55.

Hardware

The above mentioned classes of processors can be first loaded onto a FPGA to evaluate. [Digilent Arty A7](#) are development boards that are compatible with the processors.

They board support for different processors are given below

1. E-arty 35T
 - Shakti E-Class SoC on Arty A7 35T development board.
 - Also referred to as Pinaka
2. E-arty 100T

- Shakti E-Class SoC on Arty A7 100T development board.
 - Also referred to as Parashu
3. C-arty 100T
- Shakti C-Class SoC on Arty A7 100T development board.
 - Also referred to as Vajra.

Board Setup

User Manual: http://shakti.org.in/docs/user_manual.pdf

Shakti Official Website: <http://shakti.org.in/documentation.html>

All the following steps are from section 3 of the user manual.

Prerequisites

Ensure that Ubuntu 16.04 or above is used. (I had installed all the tools in Ubuntu 18.04 LTS)

Before starting to work on the development board we have to generate the RTL bitstream and program it to the FPGA.

Tooling

```
sudo apt install ghc libghc-regex-compat-dev libghc-syb-dev iverilog  
sudo apt install libghc-old-time-dev libfontconfig1-dev libx11-dev  
sudo apt install libghc-split-dev libxft-dev flex bison libxft-dev  
sudo apt install tcl-dev tk-dev libfontconfig1-dev libx11-dev gperf
```

Blue Spec Compiler

```
git clone --recursive https://github.com/B-Lang-org/bsc  
cd bsc
```

make PREFIX=/path/to/installation/folder

The /path/to/installation/folder must be replaced with where you git cloned your bsc repository;

Use **make release PREFIX=/path/to/installation/folder** to build and release dir with tools and docs.

For example :

```
swakath@LAPTOP-EJIR81GE:~/shakti/bsc$ pwd
/home/swakath/shakti/bsc
swakath@LAPTOP-EJIR81GE:~/shakti/bsc$ make PREFIX=/home/swakath/shakti/bsc
This Makefile will create an installation of the Bluespec Compiler tools,
in a directory named "inst". This directory can be moved anywhere, but
the contents should remain in the same relative locations. Intermediate
files are stored in a directory named "build". The "clean" target will
delete the "build" directory; the "full_clean" target will delete both
the "build" and "inst" directories.

make release      Build a release dir with the tools and docs

make install-src  Build and install just the tools
make install-doc  Build and install just the documentation

make check-smoke  Run a quick smoke test
make check-suite  Run the test suite (this will take time!)

make clean        Remove intermediate build-files unnecessary for execution
make full_clean   Restore to pristine state (pre-building anything)
swakath@LAPTOP-EJIR81GE:~/shakti/bsc$
```

Export the path of the bin folder in the bsc folder to .bashrc using the following command

```
export PATH=$PATH:/path/to/installation/folder/bin
```

Installing DTC:

The User guide calls for running these from the home directory.

```
sudo wget
https://git.kernel.org/pub/scm/utils/dtc/dtc.git/snapshot/dtc-1.4.7.t
ar.gz
sudo tar -xvzf dtc-1.4.7.tar.gz
cd dtc-1.4.7/
sudo make NO_PYTHON=1 PREFIX=/usr/
sudo make install NO_PYTHON=1 PREFIX=/usr/
```

Installing Vivado:

Ensure that you first register for an account with Xilinx. If you don't have a Xilinx account, create a free account, using below url

[Create Account \(xilinx.com\)](https://www.xilinx.com/registration)

Download the Vivado HLx 2018.3 Linux Self Extracting Web Installer, by clicking on the below link

[Xilinx Inc - Sign In](#)

****Prefer using Vivado HLx version 2018.3**

Make the downloaded Vivado installer executable and run it using the following command

```
chmod +x Xilinx_*.bin  
sudo ./Xilinx_*.bin
```

* - Indicate the downloaded file name.



If your installation gets stuck at the final processing stage like for example: -



Then the solution is: -

(On Ubuntu 20.04 this issue is usually observed due to missing libinfo.so.5 package.

Kindly delete/uninstall the previous installation and run the following command to install the libinfo package, once installed, re-install the Vivado/Vitis:)

sudo apt-get install libtinfo5

Install the Xilinx cable drivers:

```
cd
```

```
/tools/Xilinx/Vivado/2018.3/data/xicom/cable_drivers/lin64/install_script/install_drivers
```

```
sudo ./install_drivers
```

Install Digilent board files

Changing permissions

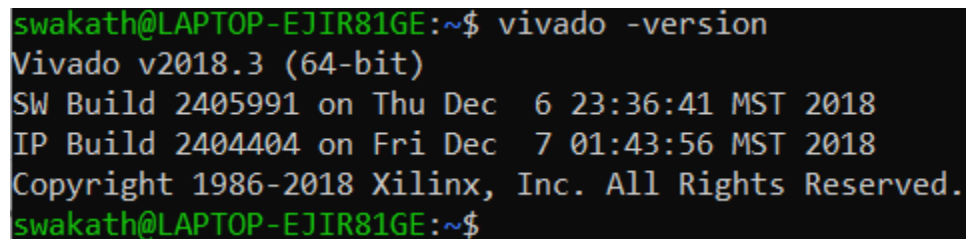
```
cd
cd .Xilinx/Vivado
sudo chown -R $USER *
sudo chmod -R 777 *
sudo chgrp -R $USER *
```

Add vivado path to environment variable PATH in .bashrc

```
export PATH=$PATH:/tools/Xilinx/Vivado/2018.3/bin
export PATH=$PATH:/tools/Xilinx/SDK/2018.3/bin
```

Test Vivado installation

```
vivado -version
```

A terminal window with a black background and green and white text. The prompt is 'swakath@LAPTOP-EJIR81GE:~\$'. The command 'vivado -version' has been executed, resulting in the following output: 'Vivado v2018.3 (64-bit)', 'SW Build 2405991 on Thu Dec 6 23:36:41 MST 2018', 'IP Build 2404404 on Fri Dec 7 01:43:56 MST 2018', 'Copyright 1986-2018 Xilinx, Inc. All Rights Reserved.', and the prompt 'swakath@LAPTOP-EJIR81GE:~\$' again.

```
swakath@LAPTOP-EJIR81GE:~$ vivado -version
Vivado v2018.3 (64-bit)
SW Build 2405991 on Thu Dec 6 23:36:41 MST 2018
IP Build 2404404 on Fri Dec 7 01:43:56 MST 2018
Copyright 1986-2018 Xilinx, Inc. All Rights Reserved.
swakath@LAPTOP-EJIR81GE:~$
```

Download Board Files

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

```
cd vivado-boards/new/board_files
```

```
sudo cp -r ./* /tools/Xilinx/Vivado/2018.3/data/boards/board_files
```

```
git clone https://github.com/Digilent/digilent-xdc.git
```

```
cd digilent-xdc
```

```
sudo cp -r ./* /tools/Xilinx/Vivado/2018.3/data/boards/board_files
```

Note: Make a dedicated directory to vivado download Board files and vivado bin file(Doubt)

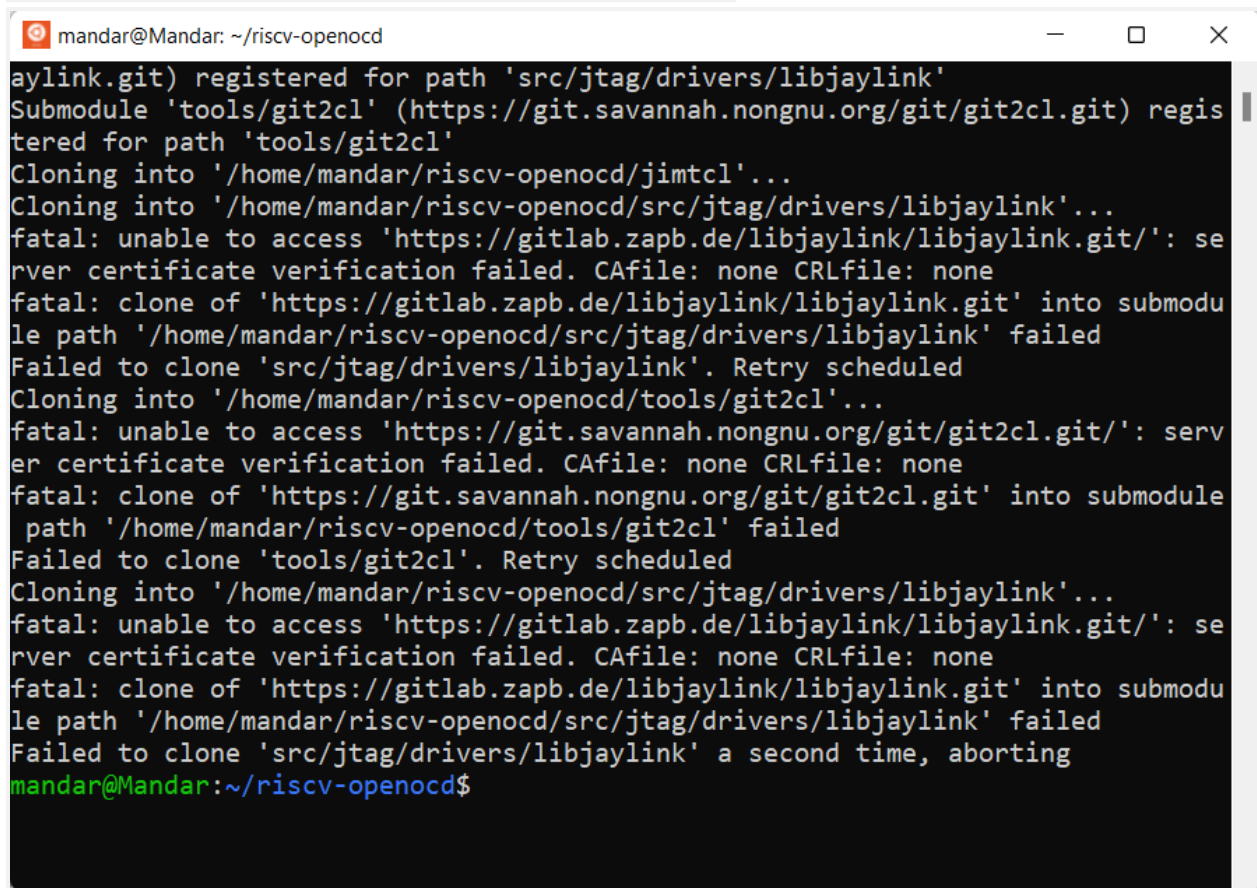
Installation of Miniterm

```
sudo apt-get install python3-serial
sudo apt-get install -y libtool
```

Installation of OpenOCD

```
git clone https://github.com/riscv/riscv-openocd.git
cd riscv-openocd
sudo ./bootstrap
```

If you encounter error like this then,

A terminal window titled 'mandar@Mandar: ~/riscv-openocd' showing a series of git submodule errors. The errors indicate that git is unable to access certain repositories (https://gitlab.zapb.de/libjaylink/libjaylink.git and https://git.savannah.nongnu.org/git/git2cl.git) due to server certificate verification failures. The terminal output shows the cloning process for several submodules, including 'src/jtag/drivers/libjaylink' and 'tools/git2cl', all of which failed with the same error message: 'fatal: unable to access 'https://gitlab.zapb.de/libjaylink/libjaylink.git/': server certificate verification failed. CAfile: none CRLfile: none'. The prompt 'mandar@Mandar:~/riscv-openocd\$' is visible at the bottom.

```
aylink.git) registered for path 'src/jtag/drivers/libjaylink'
Submodule 'tools/git2cl' (https://git.savannah.nongnu.org/git/git2cl.git) regis
tered for path 'tools/git2cl'
Cloning into '/home/mandar/riscv-openocd/jimtc1'...
Cloning into '/home/mandar/riscv-openocd/src/jtag/drivers/libjaylink'...
fatal: unable to access 'https://gitlab.zapb.de/libjaylink/libjaylink.git/': se
rver certificate verification failed. CAfile: none CRLfile: none
fatal: clone of 'https://gitlab.zapb.de/libjaylink/libjaylink.git' into submodu
le path '/home/mandar/riscv-openocd/src/jtag/drivers/libjaylink' failed
Failed to clone 'src/jtag/drivers/libjaylink'. Retry scheduled
Cloning into '/home/mandar/riscv-openocd/tools/git2cl'...
fatal: unable to access 'https://git.savannah.nongnu.org/git/git2cl.git/': serv
er certificate verification failed. CAfile: none CRLfile: none
fatal: clone of 'https://git.savannah.nongnu.org/git/git2cl.git' into submodule
path '/home/mandar/riscv-openocd/tools/git2cl' failed
Failed to clone 'tools/git2cl'. Retry scheduled
Cloning into '/home/mandar/riscv-openocd/src/jtag/drivers/libjaylink'...
fatal: unable to access 'https://gitlab.zapb.de/libjaylink/libjaylink.git/': se
rver certificate verification failed. CAfile: none CRLfile: none
fatal: clone of 'https://gitlab.zapb.de/libjaylink/libjaylink.git' into submodu
le path '/home/mandar/riscv-openocd/src/jtag/drivers/libjaylink' failed
Failed to clone 'src/jtag/drivers/libjaylink' a second time, aborting
mandar@Mandar:~/riscv-openocd$
```

The solution is: -

- 1) `sudo apt update`
- 2) `sudo apt-get install apt-transport-https ca-certificates -y`
- 3) `sudo update-ca-certificates`
- 4) `sudo apt-get install libusb-1.0-0-dev`

Now run again

```
sudo ./bootstrap(It should work this time)
```

```

cd ..
mkdir riscv-openocd-tool
cd riscv-openocd
export RISCVC=~/.shakti/riscv-openocd-tool/OpenOCD
sudo ./configure --prefix=$RISCVC --enable-remote-bitbang
--enable-ftdi --enable-jlink --enable-jtag_vpi --disable-werror

sudo make
sudo make install

```

Add path to environment variable PATH in .bashrc

```

export RISCVC=~/.shakti/riscv-openocd-tool/OpenOCD
export PATH=$PATH:$RISCVC/bin

```

Shakti-SOC

```

git clone https://gitlab.com/shaktiproject/cores/shakti-soc.git
cd shakti-soc/fpga/boards/artya7-35t/e-class
sudo make quick_build_xilinx (to program the fpga)

```

If the following error is encountered

```

calls to
fabric_xactors_to_slaves_7_f_rd_addr.enq vs. fabric_xactors_to_slaves_7_f_rd_addr.enq
fabric_v_f_rd_mis_7.enq vs. fabric_v_f_rd_mis_7.enq
Warning: "/Soc.bsv", line 148, column 10: (G0010)
Rule "fabric_rl_rd_xaction_master_to_slave_17" was treated as more urgent
than "fabric_rl_rd_xaction_master_to_slave_26". Conflicts:
"fabric_rl_rd_xaction_master_to_slave_17" cannot fire before "fabric_rl_rd_xaction_master_to_slave_26":
calls to
fabric_xactors_to_slaves_8_f_rd_addr.enq vs. fabric_xactors_to_slaves_8_f_rd_addr.enq
fabric_v_f_rd_mis_8.enq vs. fabric_v_f_rd_mis_8.enq
"fabric_rl_rd_xaction_master_to_slave_26" cannot fire before "fabric_rl_rd_xaction_master_to_slave_17":
calls to
fabric_xactors_to_slaves_8_f_rd_addr.enq vs. fabric_xactors_to_slaves_8_f_rd_addr.enq
fabric_v_f_rd_mis_8.enq vs. fabric_v_f_rd_mis_8.enq
Verilog file created: verilog//mkSoc.v
Elaborated module file created: ./bsv_build//mkSoc.ba
All packages are up to date.
Compilation finished
make[1]: Entering directory '/home/sreh/shakti/shakti-soc/fpga/boards/artya7-35t/e-class/boot-code'
'' Compiling BOOT Polling code ''
'' Caveat: BOOT Code starts at 0x2000. Configure RTL appropriately ''
/bin/bash: riscv32-unknown-elf-gcc: command not found
Makefile:18: recipe for target 'boot.riscv' failed
make[1]: *** [boot.riscv] Error 127
make[1]: Leaving directory '/home/sreh/shakti/shakti-soc/fpga/boards/artya7-35t/e-class/boot-code'
Makefile:151: recipe for target 'generate_boot_files' failed
make: *** [generate_boot_files] Error 2

```

Solution:

```
1) sudo chmod -r shakti-soc
```



```
2) sudo ln -s /usr/lib/x86_64-linux-gnu/libmpfr.so.6  
/usr/lib/x86_64-linux-gnu/libmpfr.so.4
```

Run “1” and try running the make command. If the error is persistent, run “2” and run the make command.

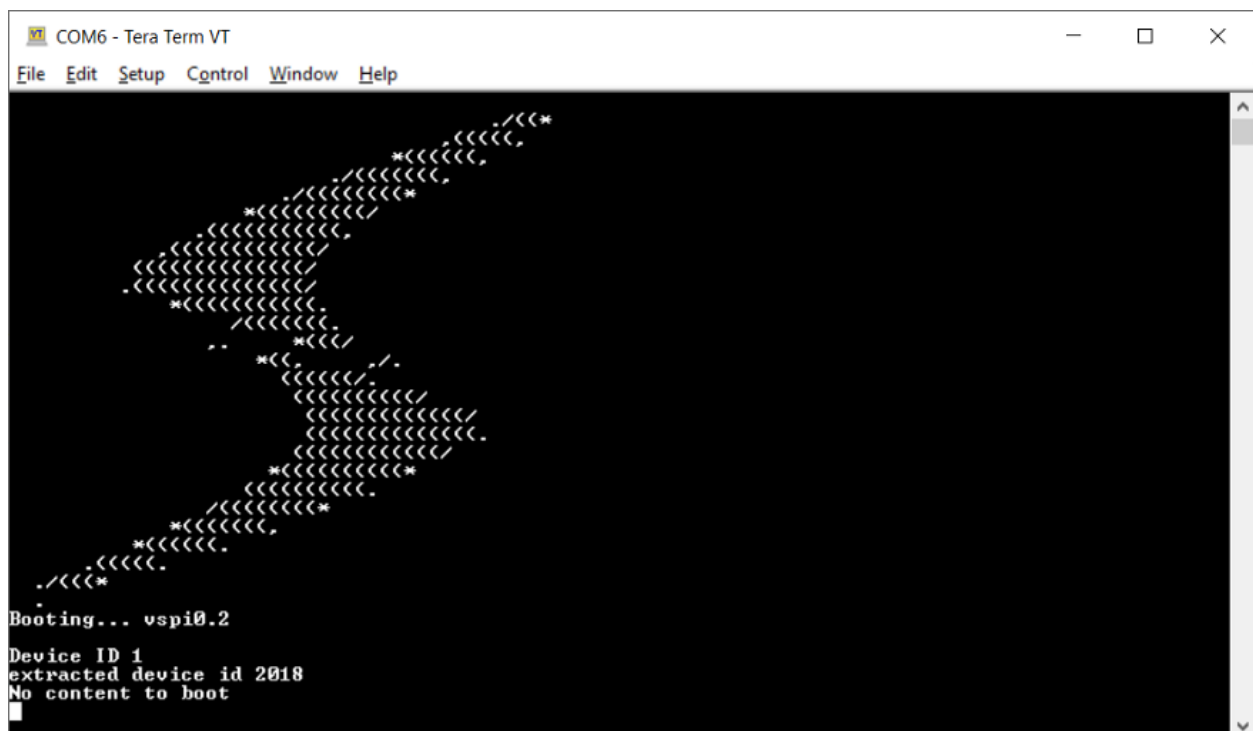
Once the programming is done. ehiDisconnect and reconnect the FPGA board

Run OpenOCD Command:

```
sudo $(which openocd) -f ./shakti-artty.cfg
```

The programming of the FPGA is completed. Y[ou can connect the board to any USB port of the computer and if you fire up a terminal program with baud rate 19200 to listen the COM port you will be able to get the following output

```
sudo miniterm.py /dev/ttyUSB0 19200
```



***** Follow the below procedure only as last measure *****

In case if the bit file is not generated in the local machine the following alternative. (Please note this will dump the default processor image only. Changes done in HDL code will not reflect in this case)

An alternative method to dump the class image was followed using the following youtube video:
<https://youtu.be/4EYoEWHGpHI>

The following git lab was used to access the mcs files :
<https://gitlab.com/shaktiproject/sp2020/-/tree/master/mcs>

The description in the video:

Steps for arty7 35t/100t:

1. Connect the board (arty7 35t or 100t) to the System.
2. Download the required mcs file.
3. Start Vivado 2018.3 and Open Hardware Manager.
4. Click on open target, Proceed to click on Auto connect. Now the FPGA is automatically connected
5. Right-click on the board name "xc7a100t" or "xc7a35t". Select " Add Configuration memory device"
6. Select:
 - *Manufacture as Spansion or Micron (based on the Micron or Spansion flash on the board)
 - *Density -128
 - *Type - spi
 - *Width -x1_x2_x4
7. Select Configuration memory part and click ok
8. Select the Required Configuration file and Click ok to program SHAKTI.