Building your own system based on the open and free 64-bit “Microwatt” OpenPOWER ISA processor.

H. Peter Hofstee

Paul Mackerras

Goal for Today's Session

- Provide you with an easy to follow set of instructions on how to build your own MicroWatt based linux system on the Arty A7-100T FPGA

- All instructions (and a recording of this session) are available at

https://github.com/hofstee-hp/

MicroWatt Summary

Tiny in-order single issue powerpc core

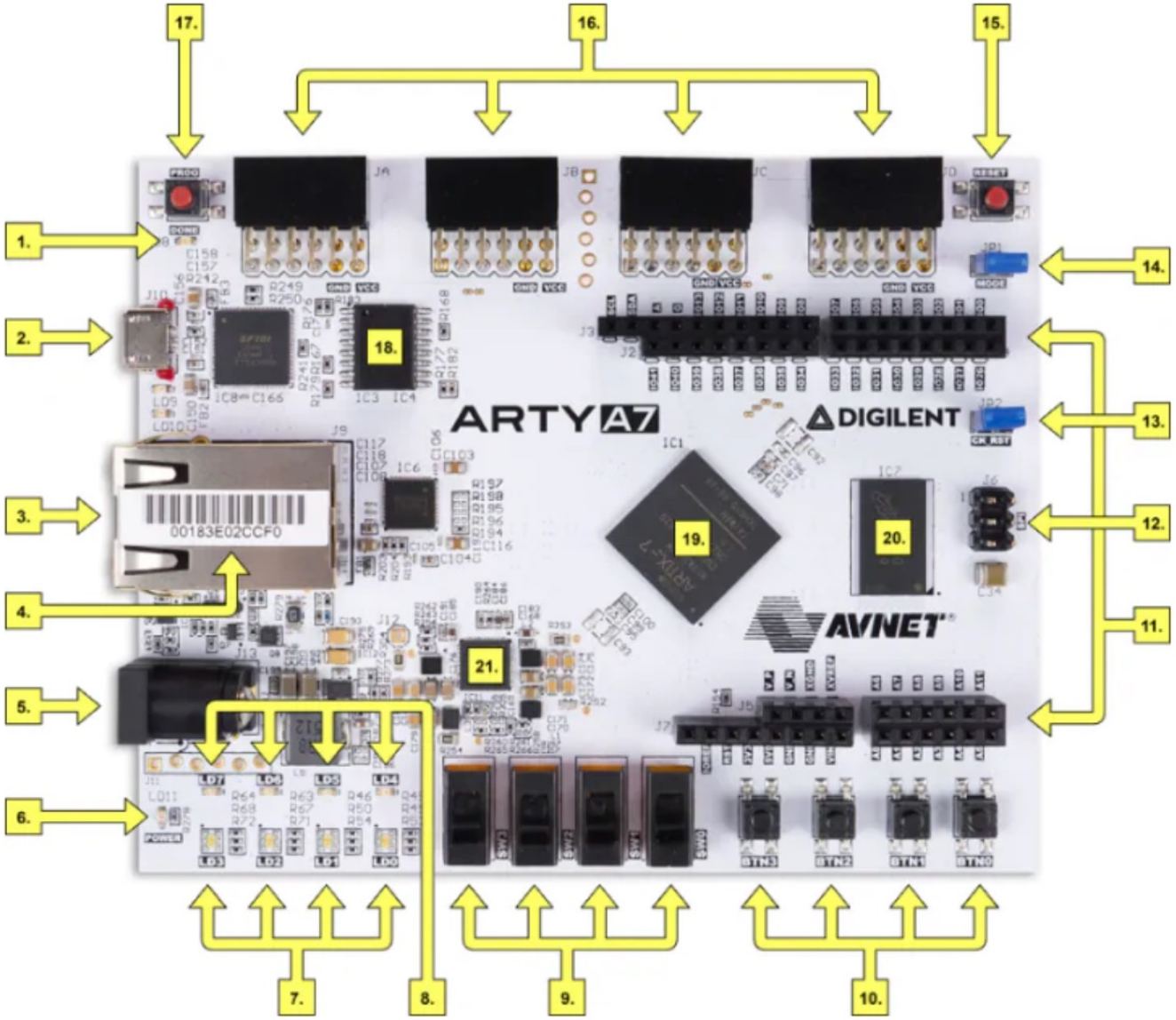
https://github.com/antonblanchard/microwatt
https://youtu.be/uEAoMCE6IKo

Power ISA compliant

https://openpowerfoundation.org/
https://youtu.be/CnMwCrtz6MA

Small enough to fit on a small FPGA and be used as a microcontroller, but capable enough to run linux!

Arty A7-100T



Callout	Description
1	FPGA programming DONE LED
2	Shared USB JTAG / UART port
3	Ethernet connector
4	MAC address sticker
5	Power jack for optional external supply
6	Power good LED
7	User LEDs
8	User RGB LEDs
9	User slide switches
10	User push buttons
11	Arduino/chipKIT shield connectors
12	Arduino/chipKIT shield SPI connector
13	chipKIT processor reset jumper
14	FPGA programming mode
15	chipKIT processor reset
16	Pmod connectors
17	FPGA programming reset button
18	SPI flash memory
19	Artix FPGA
20	Micron DDR3 memory
21	Dialog Semiconductor DA9062 power supply

What this project depends on

- Ubuntu 22.04.4 system (x86-64 connected to network)
  - Recommend at least 8GB RAM and 100GB storage
    - Tested w. 16GB / 1TB storage (but less than 100GB storage used)
  - Instructions assume only a minimal install
  - Only thing needed if you want to simulate MicroWatt

- Vivado 2024.1
  - Installed for CDAC VMs, install instructions included if you have your own Ubuntu 22.04.4 system

- Arty A7-100T

Dependencies

```
$ sudo apt-get update
$ sudo apt-get upgrade
$ sudo apt-get install build-essential git ghdl-common ghdl ghdl-llvm gnat
$ sudo apt-get install binutils-powerpc64le* gcc-powerpc64le-* g++-powerpc64le-*
```

The `build-essential` package typically includes:

1. ***GNU Compiler Collection (GCC)***: The essential compiler for C, C++, and other programming languages.
2. ***GNU Make***: A build automation tool that manages the build process of a project.
3. ***libc6-dev***: Development libraries and header files for the C library.
4. ***dpkg-dev***: Tools for building Debian source packages.
5. ***GNU Binutils***: A collection of binary tools, including the linker and assembler.

Source: medium.com

Dependencies

```
$ sudo apt-get update
$ sudo apt-get upgrade
$ sudo apt-get install build-essential git ghdl-common ghdl ghdl-llvm gnat
$ sudo apt-get install binutils-powerpc64le* gcc-powerpc64le-* g++-powerpc64le-*
```

GHDL is an open-source simulator for the VHDL language. GHDL allows you to compile and execute your VHDL code directly in your PC.

GHDL fully supports the 1987, 1993, 2002 versions of the IEEE 1076 VHDL standard, and partially the latest 2008 revision (well enough to support fixed_generic_pkg or float_generic_pkg).

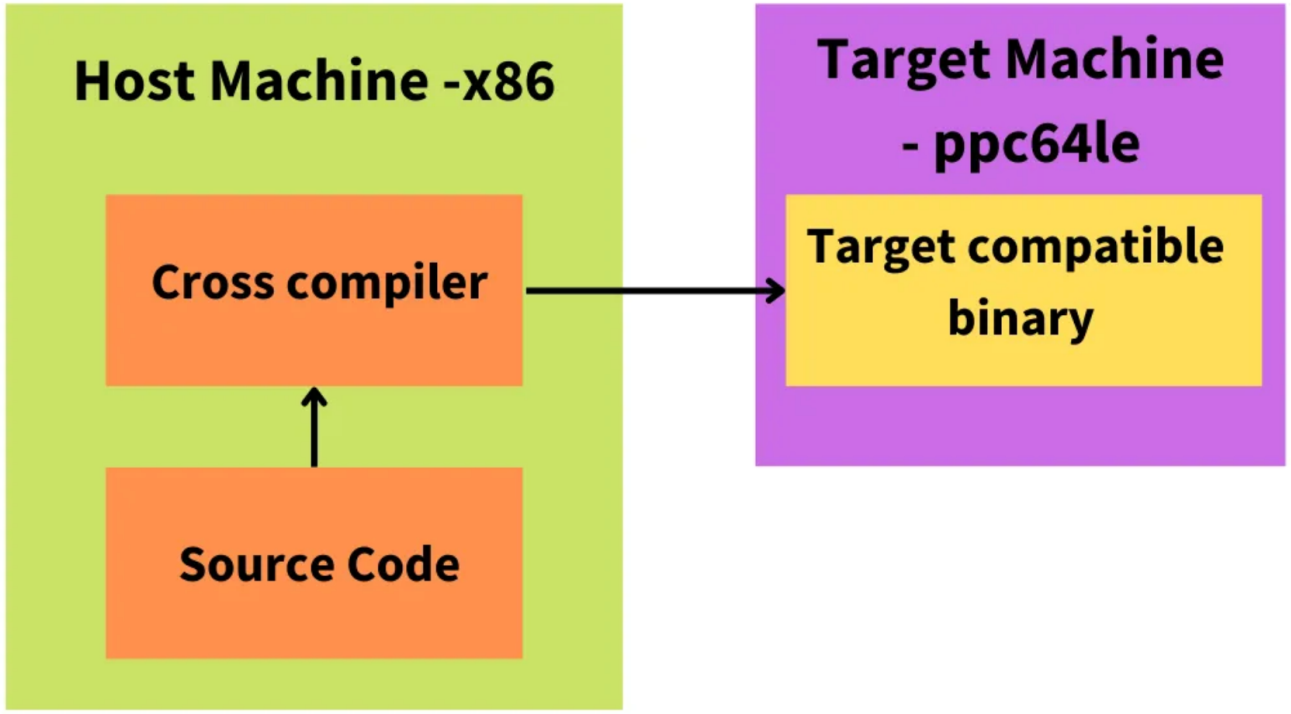
By using a code generator (llvm, GCC or a builtin one), GHDL is much faster than any interpreted simulator. GHDL runs on Linux, Windows and Apple OS X. You can freely download a binary distribution for your OS or try to compile GHDL on your own machine.

Go to GitHub to get releases or to get the sources!

gnat is the GNU Ada compiler

Dependencies

```
$ sudo apt-get update
$ sudo apt-get upgrade
$ sudo apt-get install build-essential git ghdl-common ghdl ghdl-llvm gnat
$ sudo apt-get install binutils-powerpc64le* gcc-powerpc64le-* g++-powerpc64le-*
```



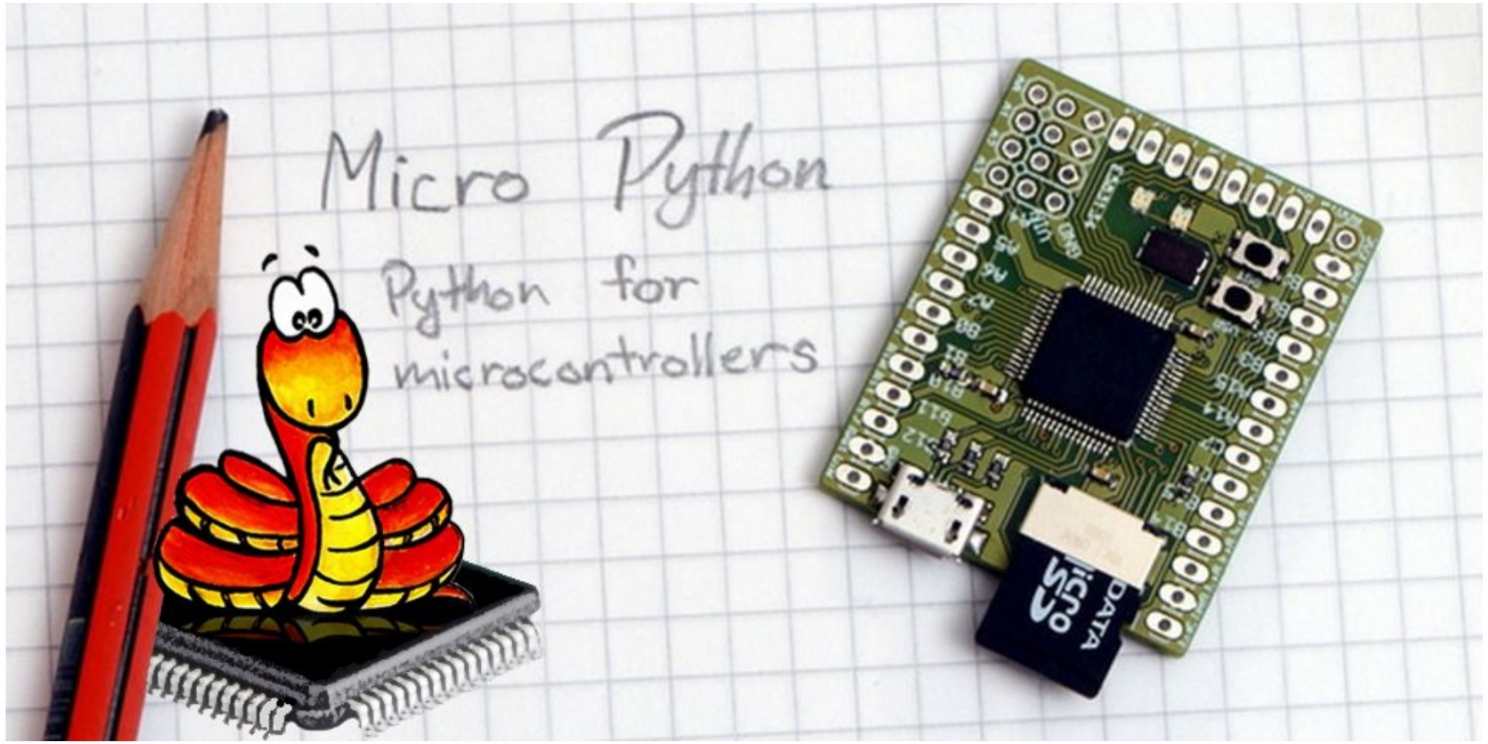
Cross compiler operation

https://medium.com/@mayurwaghmode/cross-compiling-c-c-code-for-ppc64le-on-x86-64-6226ff3a80b7

Building Micropython

```
$ cd ~
$ git clone https://github.com/micropython/micropython.git
$ cd ~/micropython
$ cd ports/powerpc
$ make
```

The MicroPython project



This is the MicroPython project, which aims to put an implementation of Python 3.x on microcontrollers and small embedded systems. You can find the official website at micropython.org.

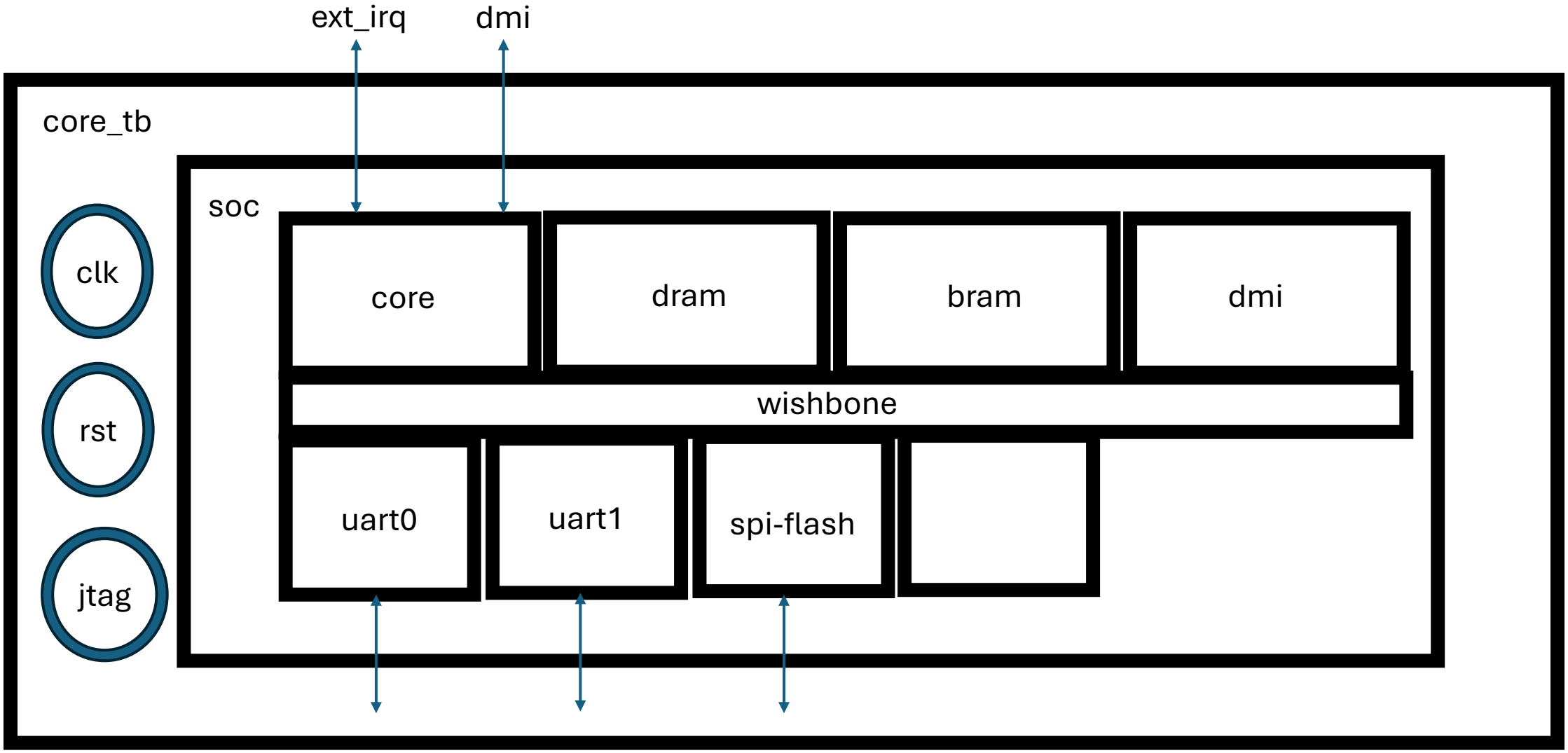
Building MicroWatt and Compiling/Running MicroPython w. GHDL

```
$ cd ~
$ git clone https://github.com/antonblanchard/microwatt
$ cd ~/microwatt
$ make
$ cd ~/microwatt
$ ln -s ../micropython/ports/powerpc/build/firmware.bin main_ram.bin
$ ./core_tb > /dev/null
```

core_tb source is **core_tb.vhd**

UNDER CONSTRUCTION

High-Level SoC Structure



Building BuildRoot / Linux Kernel

- The default linux kernel for ppc64le has a dependency on library functions that use instructions that operate on the 128b VMX registers, and these instructions are not included in the subset that MicroWatt (currently) implements, hence a special build is required
- Detailed steps on how to build the kernel are included on the github page
- It is not difficult to do, but it takes a long time
- Instead you can download a pre-built kernel from
  - ADD THIS TO THE GITHUB PAGE

Installing Vivado 24.01

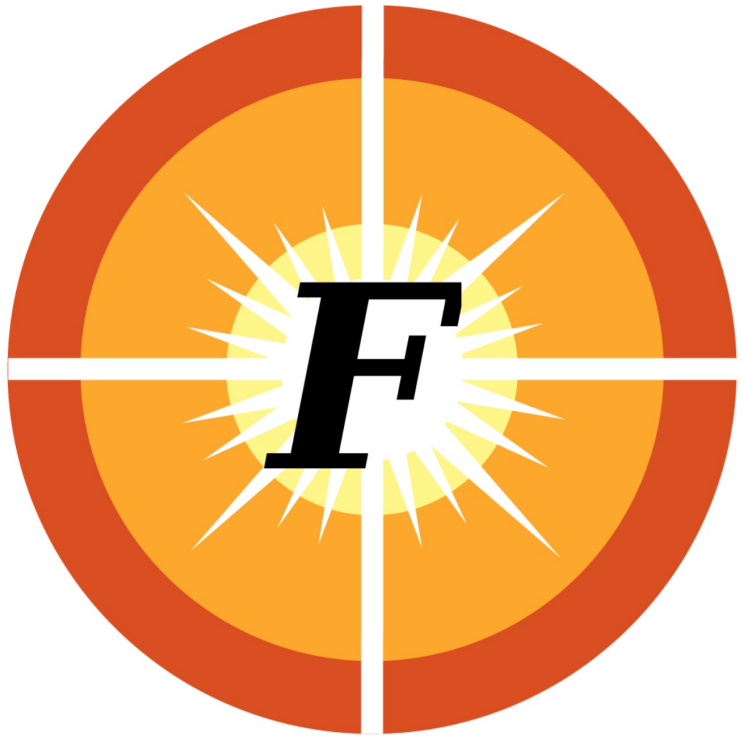
- This project has been tested with Vivado 24.01
  - Older versions may also work

- Vivado is pre-installed in the CDAC build environment
  - You do not have to install Vivado yourself
  - If you prefer to use your own system to run Vivado, detailed instructions can be found on the github page

FuseSoC

FuseSoC is an award-winning package manager and a set of build tools for HDL (Hardware Description Language) code.

Its main purpose is to increase reuse of IP (Intellectual Property) cores and be an aid for creating, building and simulating SoC solutions.



FuseSoC makes it easier to

- reuse existing cores
- create compile-time or run-time configurations
- run regression tests against multiple simulators
- Port designs to new targets
- let other projects use your code
- set up continuous integration

https://github.com/fusesoc/fusesoc.github.io

Install FuseSoC

```
----- install fusesoc
$ cd ~
$ sudo ln -s /usr/bin/python3 /usr/local/bin/python
$ sudo apt install pip3
$ pip3 install --user -U fusesoc

------ fusesoc init was not recognized so I had to do create a config file
$ cat - > ~/.config/fusesoc/fusesoc.conf
[main]
cores_root = ~/fuse/fusesoc-cores
cache_root = ~/fuse/fuse-cache
build_root = ~/fuse/fuse-builds
<ctrl-D>
$ mkdir ~/fuse
$ mkdir ~/fuse/fusesoc-cores
$ mkdir ~/fuse/fuse-cache
$ mkdir ~/fuse/fuse-builds

$ export PATH=$PATH:/home/<user>/.local/bin
```

Building the MicroWatt Bitfile

```
------ Update paths
$ cd <Xilinx install dir>/Xilinx/Vivado/2024.1
$ source settings64.sh
```

Import the required elements and build the bitfile for the MicroWatt on Arty

```
$ cd ~
$ fusesoc fetch uart16550
$ fusesoc library add microwatt microwatt
$ fusesoc run --build --target=arty_a7-100 microwatt --no_bram --memory_size=0
```

Tools Needed to Program and Run Microwatt

```
----- If you are installing on a system that has the board attached
$ sudo apt-get install openocd
$ sudo apt-get install putty
$ sudo apt-get install gtkterm
$ cd <Xilinx install dir>/Xilinx/Vivado/2024.1/data/xicom/cable_drivers/
$ cd lin64/install_scripts/install_drivers
$ ./install_drivers
$ sudo adduser $USER dialout
```

And Finally … Run Linux on MicroWatt

- Program Arty with the MicroWatt bitfile and Linux image

This operation will overwrite the contents of your flash.

```
$ microwatt/openocd/flash-arty -f a100 build/microwatt_0/arty_a7-100-vivado/microwatt_0.bit
$ microwatt/openocd/flash-arty -f a100 dtbImage.microwatt.elf -t bin -a 0x400000
```

The gateware has firmware that will look at FLASH_ADDRESS and attempt to parse an ELF there, loading it to the address specified in the ELF header and jumping to it. You may have to push the “program” button if you don’t see it starting automatically.

```
$ gtkterm -p /dev/ttyUSB1
```

Using Arty as an Edge Device

If you want to use Arty in an edge type environment, you will likely want to enable remote access over the Ethernet. To do this you'll need to connect your Arty to an Ethernet router.

```
In your gtkterm terminal after the system boots you should see

Welcome to Buildroot
microwatt login: ( enter "root" – without quotes )
# passwd root
# udhcpc -i eth0
```

If you look at the output while Linux is booting you'll see the IP address (lease obtained for ...)
Connect to Arty from a network connected device

ssh root@x.y.z.u

Still to come …

- Adding a MicroSD to Arty
  - Support should already be there, will update the github page with detailed instructions once it works
- Example of using MicroWatt as a microcontroller (for something on or attached to the board)
- Example of adding a custom instruction to MicroWatt
- **Instructions on how to prepare the MicroWatt SoC for tapeout to a semiconductor fab (or two)**