Purdue MicroBrewer: A Microcontroller Generator

Jacob R. Stevens John Skubic Eric Colter Dr. Matthew Swabey
Dr. Mark Johnson
{steven69, jskubic, coltere, maswabey, mcjohnso}@purdue.edu

March 1st, 2017

1 Abstract

Purdue MicroBrewer quickly and easily generates various RISCV-powered microcontrollers from an XML configuration file and a library of peripherals, providing the configured RTL, drivers, documentation, and scripts for FPGA synthesis. This generated microcontroller can then be flashed onto an FPGA. An open sourced board will be provided to support the FPGA. The end goal is to empower students, universities, makers, and other interested parties who may not even necessarily have digital design experience to be able to create fully customized microcontrollers, synthesized on FPGAs, for use in projects, classrooms, and research simply by defining an XML configuration file. In this whitepaper, we will first describe the motivation behind Purdue MicroBrewer, then describe the proposed architecture, and finally finish with an overview of the current progress of the ongoing project.

2 Motivation

With the silicon industry's increasing influence on other sectors and the success of the Arduino ecosystem, microcontrollers are becoming more and more accessible. Even governments and universities are recognizing the importance of access to microcontrollers, with a team from the Industrial University of Santander (UIS) of Columbia designing puzzle-piece shaped microcontrollers to appeal to children as young as five years old [1]. These forces have made microcontrollers quite accessible and usable, with a large presence in classrooms and maker spaces.

At the same time, people have been pushing for a democratization in ASIC design. OpenSoC System Architect [2], Rocket Chip [3], and Open-V are all RISCV-based projects that aim to enable chip designers to more quickly and cheaply produce high quality ASICs. These efforts will help spread the free and open RISCV ISA, as well as help new players enter the silicon industry. However, even with these advancements, the cost of a custom silicon run is a limiting factor, greatly increasing the cost of RISCV powered microcontrollers as compared to microcontrollers from established silicon vendors. This makes it difficult to use RISCV silicon in most microcontroller projects; educators, students, and makers simply cannot justify the large increase in the cost of commodity solutions.

Purdue MicroBrewer aims to bridge this gap, using a framework to quickly create custom microcontrollers ready to by synthesized to an FPGA with a support board. This will allow the use of custom RISCV-based microcontrollers in everyday projects. This means that microcontrollers can be tailored to the end user's exact needs: an educator can have the design reflect the desired curriculum or a maker can modify the design to reflect his or her prototype's constraints, without the large cost and time investment of custom silicon. Even better, with Purdue MicroBrewer, there is no need to understand a single line of HDL, worry about low level driver interfaces, or write extensive documentation.

3 Purdue MicroBrewer Architecture

Purdue MicroBrewer consists of two major blocks, as shaded in orange in Figure 1: the tool that generates the HDL, FPGA, and software files and the FPGA board. Both blocks will be as free and open as possible, allowing for the community to help improve Purdue MicroBrewer. The inputs to the system, supplied by the user, are shaded in grey and the outputs generated by the tool are shaded in green.

Since the HDL, scripts, and drivers are saved, a custom microcontroller can be configured once and then deployed on multiple boards.

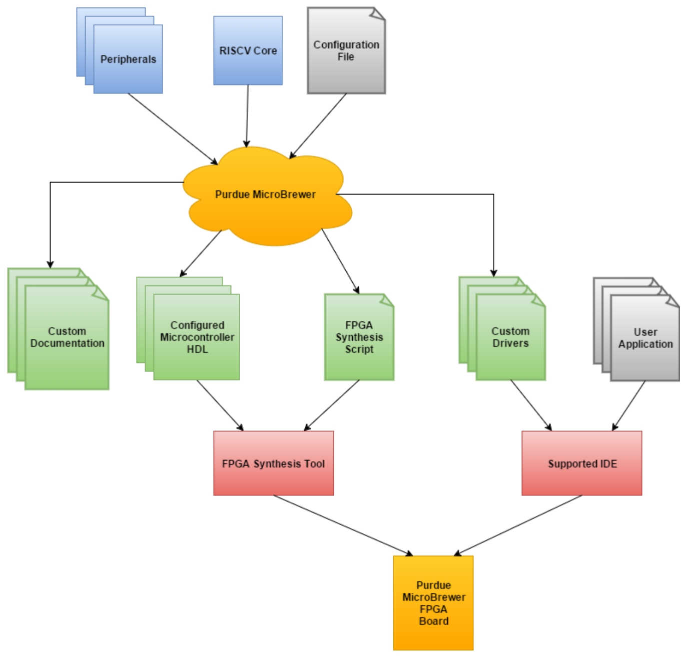


Figure 1: Purdue MicroBrewer Architecture

3.1 Purdue MicroBrewer Tool

The Purdue MicroBrewer tool operates on three types of input: a set of peripherals/buses, a RISCV core, and a configuration file. The configuration file is used to generate a custom microcontroller that uses a subset of the peripherals and cores. A custom set of drivers that reflect that layout described in the configuration file, as well as documentation for the peripherals included in the microcontroller, are also generated. The inputs and outputs to the Purdue MicroBrewer tool is explained in more depth in the following sections.

3.1.1 Peripherals

Peripherals, such as SPI, UART, timers, etc, are designed with a standardized interface. A standard interface is necessary to allow for the tool to be able to properly create a configured microcontroller. By following this standard, which will be freely available and easy to understand, any microcontroller peripheral written in SystemVerilog can easily be used with Purdue MicroBrewer.

There are two stakeholders who will be interested in the peripherals available in Purdue MicroBrewer: users taking advantage of existing IP and users wishing to develop new IP. Users such as hobbyists and educators leverage the existing IP to create custom microcontrollers. He or she can select which peripherals are included, as well as how many instances of those peripherals are used. Users such as researchers can develop new IP– a convolutional neural network accelerator, for example– and quickly integrate this IP into an existing SoC. Alternatively, users can also develop new peripherals that he or she thinks would be useful for others and integrate these new peripherals into Purdue MicroBrewer.

3.1.2 RISCV Core

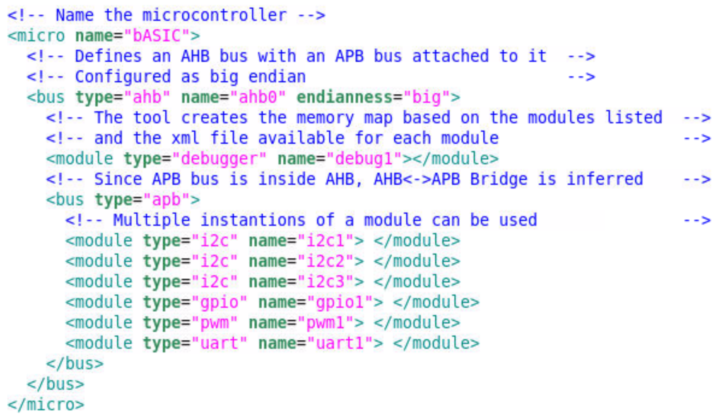
Purdue MicroBrewer uses RISCV-Business, another Purdue RISC-V project, as its core. RISCV-Business is a configurable and extensible processor, written in SystemVerilog. Similar to Purdue MicroBrewer, RISCV-Business uses a configuration file to quickly create new processors with different parameters, such as branch prediction scheme, cache size, or bus interface. Furthermore, with its RISC-MGMT (RISC Massively Generic Modification Tie-in) framework, different ISA extensions can be easily integrated into the base RV32I processor. This allows for users to easily design application specific instruction set processors, or ASIPs, on top of an open source design for which the source code is readily available and modifiable. Thus, the configuration available to Purdue MicroBrewer's end users is not just limited to peripherals, but also applicable to the RISCV core as well. More information about RISCV-Business can be found in the project's own whitepaper [4].

Although RISCV-Business's configurable and extensible two stage RV32I core is currently used as the base processor, any RISCV core written in SystemVerilog can be used, as long as the top level follows a standard interface. This allows for the community to develop cores to suit its needs.

3.1.3 Configuration File

A custom microcontroller is fully defined by an XML file. This XML file contains auxiliary information such as the name of the microcontroller, but more importantly it defines the bus structure of the microcontroller. Hierarchical bus structures are allowed, and the hierarchy may contains different bus types.

The XML file format chosen is flexible enough to be easy to use for basic configurations but powerful enough to accommodate advanced use cases; for example, optional XML parameters are available to allow for a single peripheral to be a master on one bus and a slave on another. An example of a basic microcontroller configuration is shown in Figure 2.



```xml
<!-- Name the microcontroller -->
<micro name="bASIC">
  <!-- Defines an AHB bus with an APB bus attached to it  -->
  <!-- Configured as big endian                           -->
  <bus type="ahb" name="ahb0" endianness="big">
    <!-- The tool creates the memory map based on the modules listed  -->
    <!-- and the xml file available for each module                   -->
    <module type="debugger" name="debug1"></module>
    <!-- Since APB bus is inside AHB, AHB<->APB Bridge is inferred    -->
    <bus type="apb">
      <!-- Multiple instantions of a module can be used               -->
      <module type="i2c" name="i2c1"> </module>
      <module type="i2c" name="i2c2"> </module>
      <module type="i2c" name="i2c3"> </module>
      <module type="gpio" name="gpio1"> </module>
      <module type="pwm" name="pwm1"> </module>
      <module type="uart" name="uart1"> </module>
    </bus>
  </bus>
</micro>
```

Figure 2: Example Microcontroller XML Configuration

3.1.4 Configured Microcontroller HDL

When the Purdue MicroBrewer tool is invoked, a new folder with the given name is created for the custom microcontroller. Inside this folder, various HDL files specific to the custom microcontroller are created: mainly, the top level that instantiates the desired peripherals and the bus interconnects. This is also where the FPGA synthesis script will be stored. The configured HDL is available as un-obfuscated SystemVerilog.

3.1.5 FPGA Synthesis Script

Purdue MicroBrewer uses a variation of the waf build system to generate a .tcl script targeting the chosen FPGA. This script includes all of the source and header files necessary to synthesis the custom microcontroller, as well as clocking and other constraints.

The variation of the waf build system used is also open source. Thus, if a new format of sythesis script needs to be added– for example, a .tcl script targeting the Quartus toolchain– the community using Purdue MicroBrewer can easily modify waf and the Purdue MicroBrewer tool to support this.

3.1.6 Drivers and Documentation

Purdue MicroBrewer allows the addition of 3rd party, community created peripherals to be used along with those provided by the Purdue MicroBrewer team. A potential problem with using a set of peripherals from different authors is that the documentation and firmware provided may be styled very differently, making it difficult for a new user to quickly understand. Rather than requiring the hardware designers to provide firmware and documentation, Purdue MicroBrewer enforces consistency between different modules by automatically generating documentation and firmware from a module description XML file.

Hardware designers include with their design a module description XML file that follows the module description format specified by the Purdue MicroBrewer team. This format allows the hardware designer to describe registers visible to the bus through memory mapped input/output. They may also declare arrays of register, break registers into fields of varying bit lengths, and associate names with certain values of a field or register.

When building a microcontroller, Purdue MicroBrewer parses the module definition XML of all the included peripherals and generates documentation, as well as firmware that provides macros to easily access the registers and fields of the included peripherals. This has the additional benefit of maintaining consistency between the firmware and documentation as both are generated from the same source. Furthermore, the documentation generator can be modified to generate new forms of documentation, such as webpages, markdown, or a pdf file, with no extra work required by the individual hardware designers.

This automated generation from XML written by designers supporting Purdue MicroBrewer allows for end users to quickly begin working with their custom microcontroller, with no need to spend time creating documentation or low level interfacing code for it. Instead, they can focus on creating the desired application for their use case.

3.2 Purdue MicroBrewer FPGA

To start, Purdue MicroBrewer targets a single, standardized FPGA. It also supplies the board schematics for a support board that contains everything needed, including but not limited to onboard flash, IO pins, programming circuitry, and debugging LEDs. By providing the FPGA scripts that target the FPGA as well as providing the board schematics and BOM, this ensures that the end users do not have to be familiar with FPGA toolchains or PCB design in order to use Purdue MicroBrewer; they can simply have the board fabricated and assembled. It is a long term goal of Purdue MicroBrewer to be able to produce these standard boards for sale as well, eliminating this step for the end user too. Furthermore, having onboard flash for the support board will allow for the custom microcontroller RTL to be persistently stored, remaining even after power cycling the board. This allows for the support board to be used exactly like a commercial, off the shelf microcontroller.

Since Purdue MicroBrewer will open sourced, it is our hope that the community will share their designs as they are produced, increasing the number of target FPGAs as well as the boards available to support them.

4 Current Progress

Purdue MicroBrewer is an ongoing project being developed by Purdue's System on Chip Extension Technologies (SoCET) team, a team of undergraduate and graduate students. There already exists a library of peripherals, most of which adhere to the Purdue MicroBrewer defined interface. These peripherals include: timer, PWM, GPIO, UART, I2C, and an SRAM controller. Other peripherals are also under active development, including SPI, a 2D drawing accelerator, and a flash controller. The RISC-V core, codenamed RISCV Business, has also been mostly verified and the team has successfully deployed a simple microcontroller consisting of this core, memory, and GPIO pins to an Altera DE-115 board, uploaded a simple program to it, and watched the core count up on seven segment displays using GPIO pins. Furthermore, the automatic generation of driver code and documentation for peripherals has been mostly completed, with testing currently ongoing using existing, non-configurable microcontrollers designed by the Purdue SoCET team. Finally, the configuration file format has been decided upon.

Work on the tool that will parse the configuration file and generate top level RTL has just begun. Additionally, a subteam of the Purdue SoCET team has begun work on the design of the FPGA board that custom microcontrollers can be flashed onto.

5 Conclusion

In summary, Purdue MicroBrewer provides a way to quickly generate custom RISC-V powered microcontrollers and deploy them to a support board that can be used exactly like a comparable commercial, off the shelf microcontroller. Doing so expands the reach of RISC-V to educators, students, and makers looking for unique microcontroller-based solutions. Purdue MicroBrewer can empower educators to easily deploy custom microcontrollers for learning in the classroom setting, students to experiment with digital design without the need for fab access, and makers to create a microcontroller that perfectly fits their project's needs.

6 Acknowledgements

We wish to acknowledge current and recent SoCet team members who have contributed to the design, verification, and testing of modules or software that were the precursors to Purdue MicroBrewer: Yunus Akhtar, Andrew Brito, David Castley, Gregory Chang, Noah Chesnut, Allen Chien, Yuchen Cui, Dwezil D'souza, Siddhant Ekale, Haoyue Gao, Travis Garza, Mingxuan He, Yaqin Huang, Jiangshan Jing, Ruchit Kank, Prashant Lalwani, Annan Ma, Arnav Mittal, Chandan , Muddamsetty, Trevor Odelberg, Erin Rasmussen, Sudharson Ravishankar, Kevin Rockwell, Manik Singhal, Sam Sowell, Erik Swan, Chuan Yean Tan, Xin Tze (Joyce) Tee, Todd Wild (ECE Industry Design Projects Manager), Kyle Woodworth, and Aliasger Zaidy.

References

[1] Renata Columbia. Chip colombiano para democratizar la educación con tecnología. `https://www.renata.edu.co/index.php/noticias/8736-chip-colombiano-para-democratizar-la-educacion-con-tecnologia`, 2016. Accessed: 2/18/2017.

[2] David Donofrio, Farzad Fatollahi-Fard, John D. Leidel, Xi Wang, and Yong Chen. Opensoc system architect, 2017.

[3] Krste Asanović, Rimas Avižienis, Jonathan Bachrach, Scott Beamer, David Biancolin, Christopher Celio, Henry Cook, Palmer Dabbelt, John Hauser, Adam Izraelevitz, Sagar Karandikar, Benjamin Keller, Donggyu Kim, John Koenig, Yunsup Lee, Eric Love, Martin Maas, Albert Magyar, Howard Mao, Miquel Moreto, Albert Ou, David Patterson, Brian Richards, Colin Schmidt, Stephen Twigg, Huy Vo, and Andrew Waterman. The rocket chip generator. Technical report, University of California, Berkeley, 2016.

[4] John Skubic, Jacob R. Stevens, Chuan Tan Yean, Dr. Mark Johnson, and Dr. Matthew Swabey. Riscv-business: A configurable, extensible core. Technical report, Purdue University-West Lafayette, 2017.