Zephyr and RISC-V: I Ain't Afraid Of No Ghosts

MOHAMMED BILLOO
FOSDEM 2024

CC BY-SA 4.0
THE SPEAKER

• Embedded Software Consultant

• Design Work
  • Medical devices
  • Scientific instruments
  • LIDAR
  • Custom ASIC
  • Consumer electronics

• Experience/Expertise
  • RTOS-based systems
  • Embedded Linux/The Yocto Project
  • Qt

Mohammed Billoo
(mab@mab-labs.com)
THE SPEAKER

• Training/Workshops
  • Virtual
  • On-site/In-person

BIOS FOOD Newsletter

www.mab-labs.com
Agenda

• Background and motivation

• Process

• Issues Encountered

• Resolutions

• Enhancements

• Next Steps

• Q&A
BACKGROUND AND MOTIVATION
Background

- **Embedded software**
  - Focus is on **software**
  - Assume CPU does what you expect it to do
    - Compiler is mature and well-defined
    - Hardware is mature and well-defined*
      - Mostly....
      - Errata
  - Gets boring after > 10 years of doing “just software”
    - Itch to understand how it works under the hood
ISA

- Instruction Set Architecture
- Language to command a CPU
- Well-documented
- Nothing (technically) stopping you from:
  - Building a CPU compliant to a particular ISA
  - **Costly + time consuming**
- Issue (as always) comes from **legal + financial**
  - Can get in trouble for violating IP
  - ISA is considered vendor IP
ISA

- How to (legally) use a proprietary ISA?
- Money ($$$)
- Not friendly for hobbyists/individual
  - Pay royalty to license ISA
  - Pay fee for validated implementation
  - Both
- More appropriate for (large) companies
  - Pass added cost to (ultimately) consumer
RISC-V

• A lot of buzz around an “open-source ISA”
• Free
  • No license necessary to use the ISA
  • No royalty or (significant) upfront cost
• Commercial advantage (for companies)
  • Massive reduction in per unit cost
  • (Hopefully) pass onto consumer
  • Another advantage ... later

WCH Launches a Sub-10¢ RISC-V Microcontroller, While a $6.90 Dev Board Gets You Started

Designed for less-computationally-demanding workloads, this 32-bit RISC-V chip is priced extremely aggressively.

Hackster.io
RISC-V

• Advantage for hobbyist?

• Yes!
  • Can “implement” own RISC-V compliant CPU
  • (Probably) easier to do than proprietary ISA
  • Don’t have to worry about paying royalty

• OK ...
  • But how do I get started?
NEORV32: A customizable RISC-V SoC (adafruit.com)
73 points by quarks_ on June 23, 2021 | hide | past | favorite | 10 comments

JUNE 23, 2021 AT 10:58 AM

NEORV32: a customizable RISC-V SoC #RISCV #FPGA

The NEORV32 Processor is a customizable microcontroller-like system on chip (SoC) that is based on the RISC-V NEORV32 CPU. The project is intended as auxiliary processor in larger SoC designs or as ready-to-go stand-alone custom / customizable microcontroller.
Support for FreeRTOS, Zephyr OS and LiteX SoC Builder Framework.

```
[zephyr]$ ls boards/riscv/
adp_xc7k_ea350  hifive1  index.rst  m2gl025_miv  opentitan_earlgrey  sparkfun_red_v_things_plus
esp32c3_devkitm  hifive1_revb  it82xx2_ev8b  mips_icicle  qemu_riscv32  stamp_c3
esp32c3_luatos_core  hifive_unleashed  it82xx2_ev8b  neorv32  qemu_riscv32e  titanium_ti60_f225
gd32vf103c_starter  hifive_unmatched  litex_vexriscv  niosv_g  qemu_riscv64  tlsr9518adk80d
gd32vf103v_eval  icev_wireless  longan纳米  niosv_m  rv32m1_vega  xiao_esp32c3
```

```
[zephyr]$ ls soc/riscv/riscv-privileged/
andes_v5  common  gd32vf103 Kconfig.deconfig  miv  neorv32  opentitan  starfive_jh71xx_virt
CMakeLists.txt  efine-sapphire  Kconfig  Kconfig.soc  mpfs  niosv  sifive-freedom  telink_b91
```

Interesting ....
Getting Started

• Hardware?

• Neorv32 can be flashed to an FPGA

  • 🍀 Exemplary setups and community projects targeting various FPGA boards and toolchains to get started.

• Have a list of FPGA boards that are supposed to work

• Decided to choose UPduino v3 board
  • Lattice FPGA
  • Inexpensive + small
  • Available (at the time)
Getting Started

• Personally have Xilinx, Altera FPGA boards
  • From past project work
  • Why not use those?

• Not exact match to supported boards
  • In (limited) experience, helps to have exact board
  • Definitions in RTL may be different
    ▪ Pin definitions
    ▪ Clocks
  • Result in weird/incorrect behavior
Getting Started

• Have FPGA, what next?

• Toolchain
  • Decided to go with commercial toolchain (initially)
  • Use vendor tools to get started
    ▪ Should* work with FPGA
  • Migrate to using open-source tools later
Getting Started

• First, had to jump through some hoops to get sources all aligned
  • FPGA tools require some hand holding to get project to build
    ▪ Usually
  • Needed to run some pre-build steps
    ▪ Pre-build library

• But still ran into errors …
Hunting Ghosts

- Ran into initial errors
  - Design not updated to support latest version of Radiant
  - Attempted to resolve on my own
Hunting Ghosts

• Maintainer responded with a fix in few days!
  • Advantage of a (well-maintained) open-source project

• Still ran into an issue
  • Neorv32 has bootloader in RTL that can be enabled
  • Flash application images to bootloader
  • LED should blink for 10 seconds + UART output
    - No blinky!
    - No UART!
Hunting Ghosts

- Toolchain issue!
  - Expected (work long enough with FPGA people)
  - Recently worked with an FPGA engineer to
    - Debug IDE Segfault
      » Cost $$,000 license/year
    - RE application binary
      » Located bug in how they used std library function
      » Worked around it …
Hello RISC-V World!

All that matters (usually...)

Need to know pins for bootloader output
Hello RISC-V World!

<< NEORV32 Bootloader >>

BLDV: Jul 28 2023
HWV: 0x01090004
CLK: 0x016ae3600
MISA: 0x40901104
XISA: 0x00000093
SOC: 0x00bf800d
IMEM: 0x00010000
DMEM: 0x00010000

Autoboot in 8s. Press any key to abort.
Aborted.

Available CMDs:
h: Help
r: Restart
u: Upload
s: Store to flash
l: Load from flash
x: Boot from flash (XIP)
e: Execute
CMD:>
Invalid CMD
CMD:>
Invalid CMD
CMD:>
Zephyr On RISC-V?

- west build -b neorv32 zephyr/samples/hello_world?

- Not sufficient
  - Bootloader expects a bin file
  - `CONFIG_BUILD_OUTPUT_BIN=y`
    - Add to prj.conf

```bash
-- The ASM compiler identification is GNU
-- Found assembler: /home/mab/mab_home/zephyr-toolchains/zephyr-sdk-0.16.1/riscv64-zephyr-elf,
-- The neorv32 image_gen utility was not found, neorv32 image files cannot be generated
-- Configuring done
```
Zephyr On RISC-V?

- image_gen tool
  - Compile from Neorv32 repo under sw/
  - Add to PATH

```
[zephyrproject]$ ls build/zephyr
arch drivers kconfig linker.cmd.dep snippets_generated.cmake zephyr.dts.d zephyr.hex zephyr.vhd
boards dts.cmake kernel linker_zephyr_pre0.cmd zephyr.dts.pre zephyr.map zephyr_pre0.elf
ckmake edt.pickle lib linker_zephyr_pre0.cmd.dep soc zephyr.elf zephyr_pre0.map
CMakeFiles include libzephyr.a misc zephyr.bin zephyr_exe.bin zephyr_pre0.map
ckmake_install.cmake isr_tables.c linker.cmd runners.yaml zephyr.dts
```
“zephyr.vhd”? 

1  -- The NEORV32 RISC-V Processor: https://github.com/stnolting/neorv32  
2  -- Auto-generated memory initialization file (for APPLICATION) from source file ~/media/mab/mab_home/fosdem/zephyrproject/build/zephyr/zephyr.bin  
3  -- Size: 17344 bytes  
4  MARCH: default  
5  Built: 03.02.2024 04:28:31  
6  
7  -- prototype defined in 'neorv32_package.vhd'  
8  package body neorv32_application_image is  
9  
10 constant application_init_image : mem32_t := (  
11  x"800002197",  
12  x"80818193",  
13  x"00000297",  
14  x"07c28293",  
15  x"30529073",  
16  x"00000011",  
17  x"00000037",  
18  x"30001073",  
19  x"30401073",  
20  x"907355f5",  
21  x"10732305",  
22  x"1073b000",  
23  x"1073b800",  

RTL !

Can embed our application inside FPGA memory!
Does it work? 😞

- No output to console
- Problem when working with FPGAs/hardware
  - If hardware changes underneath, software has to follow
  - Used a forked version of Zephyr with fixes
    - Actually compiles
    - Does it work?
Included in neorv32 repo to upload binary to bootloader

[fosdem]$ ./uart_upload.sh /dev/ttyUSB0 zephyrproject/build_neorv32/zephyr/zephyr.exe.bin
Uploading... Done.

invalid CMD
CMD:> h
Available CMDs:
  h: Help
  r: Restart
  u: Upload
  s: Store to flash
  l: Load from flash
  x: Boot from flash (XIP)
  e: Execute
CMD:> e
Booting from 0x00000000...

*** Booting Zephyr OS build 6f56a6a91e2c ***
Hello World! neorv32

Woohoo!
Next Steps: Custom Function Unit

- (Personally) most interesting feature in RISC-V
- ISA Extension to allow for operations optimized in hardware
- Call from SW using compiler intrinsic
Next Steps: Custom Function Unit

- Hardware multiply and add!
Next Steps: Custom Function Unit

- Doesn’t build for UPduino board
  - Logic doesn’t fit
  - (Another) typical problem with FPGA designs
  - Resource (and timing) constraints
Next Steps: Custom Function Unit

- From SW?
- Neorv32 "sw" directory has relevant functions
  - Detect if CFU is enabled
  - Call appropriate CFU

```
[sw]$ cd example/
atomic_test/       demo_cfu/       demo_mtime/
bus_explorer/     demo_crc/       demo_neopixel/
coremark/         demo_dma/       demo_newlib/
demo_blink_led/   demo_emulate_unaligned/ demo_onewire/
demo_blink_led_asm/ demo_gptmr/     demo_pwm/
demo_cfs/         demo_hpm/       demo_sdi/
```
Next Steps: Custom Function Unit

```c
96    // check if the CFU is implemented at all (the CFU is wrapped in the core's "Zxcfu" ISA extension)
97    if (neorv32_cpu_cfu_available() == 0) {
98        neorv32_uart0_printf("ERROR! CFU ('Zxcfu' ISA extensions) not implemented!\n");
99        return 1;
```

```c
109   /*
110      The CFU custom instructions can be used as plain C functions as they are simple "intrinsics".
111      
112      There are 4 "prototype primitives" for the CFU instructions (define in sw/lib/include/neorv32_cfu.h):
113      
114      > neorv32_cfu_r3_instr(funct7, funct3, rs1, rs2) - for r3-type instructions (custom-0 opcode)
115      > neorv32_cfu_r4_instr(funct3, rs1, rs2, rs3) - for r4-type instructions (custom-1 opcode)
116      > neorv32_cfu_r5_instr_a(rs1, rs2, rs3, rs4) - for r5-type instruction A (custom-2 opcode)
117      > neorv32_cfu_r5_instr_b(rs1, rs2, rs3, rs4) - for r5-type instruction B (custom-3 opcode)
```

```c
136    for (i=0; i<TESTCASES; i++) {
137        rs1 = xorshift32();
138        neorv32_uart0_printf("%w: neorv32_cfu_r3_instr( funct7=0b1111111, funct3=0b000, [rs1]=0x%x, [rs2]=0x%x ) = ", i, rs1, 0);
139        // here we are setting the funct7 bit-field to all-one; however, this is not used at all by the default CFU hardware module.
140        neorv32_uart0_printf("0x%x\n", neorv32_cfu_r3_instr(0b1111111, 0b000, rs1, 0));
141    }
```
Next Steps: CFU In Zephyr?

• Didn’t see existing support for using Neorv32 CFU in Zephyr
  • Maybe pull in is as external module via West
  • Add it to Zephyr?
Summary and Next Steps

• UPduino running Neorv32 works with Zephyr!
  • Need to use forked version

• Help get PR past the finish line
  • Some comments need addressing
  • Can help address those comments and get mainlined

• Get bigger FPGA for CFU support
  • Determine appropriate strategy to add CFU support in Zephyr

OrangeCrab
The OrangeCrab is a feather form factor electronics development board. For the Lattice ECP5 FPGA. The board follows the slim feather board specification from Adafruit. The FPGA is compatible with all open source toolchains and is perfect for experimenting with RISC-V cores. There aren’t many FPGA boards available that make use of the ECP5, but here are some distinct features that set this board apart:

- Small Compact size (Take it anywhere!)
- Direct USB connection to the FPGA (Operate as a DFU, MSC, CDC, or composite device!)
- On-board DDR3 Memory (1Gbit!)
- Pre-loaded DFU bootloader (No external programmer required!)
- It’s Orange!

The current hardware version that we are shipping is V0.2.1.
Thank you!

Questions?

MOHAMMED BILLOO

FOSDEM 2024

CC BY-SA 4.0