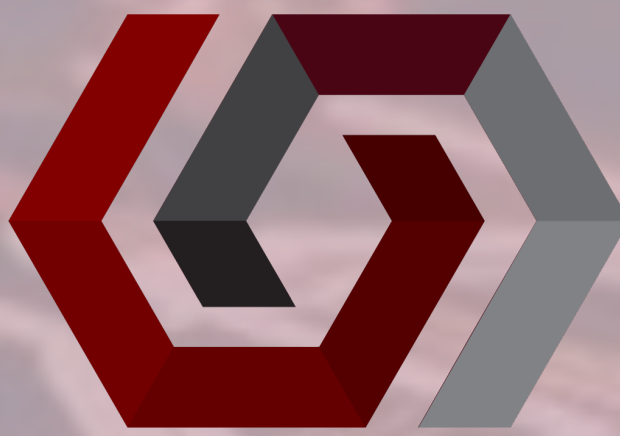




THE CFU: CUSTOM HARDWARE WITH RISC-V AND ZEPHYR

Mohammed Billoo

MAB Labs Embedded Solutions

Open Source Summit 2024

Mohammed Billoo
(mab@mab-labs.com)



/mab-embedded



@mabembedded



THE SPEAKER



- Embedded Software Consultant (NYC, USA)
- Design Work
 - Medical Devices
 - Scientific Instruments
 - LIDAR
 - Custom ASICs
- Experience/Expertise
 - Zephyr RTOS
 - Embedded Linux
 - GUI-based applications



BIOS FOOD NEWSLETTER

Training/Workshops



www.mab-labs.com



BACKGROUND



<https://www.youtube.com/watch?v=syA3xxKfB4s&t=28s>

- FOSDEM24
- Used neorv32 project to implement RISC-V ISA in an FPGA
 - Has support for built-in bootloader
- Used vendor tools to build and flash FPGA
- Loaded “Hello World” Zephyr application on to FPGA.
 - Zephyr has support for “neorv32” board
 - Uploaded application using bootloader

BACKGROUND



- After some fennagling, got output!
- See previous presentation for details
- Next step was to investigate CFU
- Roadblock
 - Design with CFU enabled didn't fit in FPGA

```
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
```

MOTIVATION



- **Custom Function Unit?**
- Allows us to offload functionality to hardware (FPGA)
- Specific for operations that are inefficient (in software):
 - Performance
 - Latency
 - Energy Consumption
 - Program Memory

MOTIVATION



- AI
- Crypto
- Communications
- Arithmetic
- Image Processing
- **Requires CPU dependency**



- Leverages “custom-0” and “custom-1” RISC-V opcodes
- CFU is implemented using the “**Zxcfu**” extension of the RISC-V ISA
 - **Specific to neorv32**
- Sample implementation of XTEA in RTL!
 - Simple block cipher
 - Embedded systems

inst[4:2] inst[6:5]	000	001	010	011	100	101	110	111 (> 32b)
00	LOAD	LOAD-FP	custom-0	MISC-MEM	OP-IMM	AUIPC	OP-IMM-32	48b
01	STORE	STORE-FP	custom-1	AMO	OP	LUI	OP-32	64b
10	MADD	MSUB	NMSUB	NMADD	OP-FP	reserved	custom-2/rv128	48b
11	BRANCH	JALR	reserved	JAL	SYSTEM	reserved	custom-3/rv128	≥ 80b

Table 19.1: RISC-V base opcode map, inst[1:0]=11

UNDER THE HOOD



rtl/core/neorv32_cpu_cp_cfu.vhd

```
-- instruction identifiers (funct3 bit-field) --
constant xtea_enc_v0_c : std_ulogic_vector(2 downto 0) := "000";
constant xtea_enc_v1_c : std_ulogic_vector(2 downto 0) := "001";
constant xtea_dec_v0_c : std_ulogic_vector(2 downto 0) := "010";
constant xtea_dec_v1_c : std_ulogic_vector(2 downto 0) := "011";
constant xtea_init_c   : std_ulogic_vector(2 downto 0) := "100";
```

Use “intrinsic” to “call” functions in CFU

sw/example/demo_cfu/main.c

```
#define xtea_hw_init(sum)          neorv32_cfu_r3_instr(0b0000000, 0b100, sum, 0 )
#define xtea_hw_enc_v0_step(v0, v1) neorv32_cfu_r3_instr(0b0000000, 0b000, v0, v1)
#define xtea_hw_enc_v1_step(v0, v1) neorv32_cfu_r3_instr(0b0000000, 0b001, v0, v1)
#define xtea_hw_dec_v0_step(v0, v1) neorv32_cfu_r3_instr(0b0000000, 0b010, v0, v1)
#define xtea_hw_dec_v1_step(v0, v1) neorv32_cfu_r3_instr(0b0000000, 0b011, v0, v1)
#define xtea_hw_illegal_inst()      neorv32_cfu_r3_instr(0b0000000, 0b111, 0, 0 )
```

UNDER THE HOOD



```
#define xtea_hw_init(sum)          neorv32_cfu_r3_instr(0b0000000, 0b100, sum, 0 )
#define xtea_hw_enc_v0_step(v0, v1) neorv32_cfu_r3_instr(0b0000000, 0b000, v0, v1)
#define xtea_hw_enc_v1_step(v0, v1) neorv32_cfu_r3_instr(0b0000000, 0b001, v0, v1)
#define xtea_hw_dec_v0_step(v0, v1) neorv32_cfu_r3_instr(0b0000000, 0b010, v0, v1)
#define xtea_hw_dec_v1_step(v0, v1) neorv32_cfu_r3_instr(0b0000000, 0b011, v0, v1)
#define xtea_hw_illegal_inst()      neorv32_cfu_r3_instr(0b0000000, 0b111, 0, 0 )
```

sw/lib/include/neorv32_cpu_cfu.h

```
#define neorv32_cfu_r3_instr(func7, func3, rs1, rs2) \
    CUSTOM_INSTR_R3_TYPE(func7, rs2, rs1, func3, 0b0001011)
```

UNDER THE HOOD



```
#define neorv32_cfu_r3_instr(func7, funct3, rs1, rs2) \  
    CUSTOM_INSTR_R3_TYPE(func7, rs2, rs1, funct3, 0b0001011)
```

sw/lib/include/neorv32_intrinsics.h

```
#define CUSTOM_INSTR_R3_TYPE(func7, rs2, rs1, funct3, opcode) \  
{  
    uint32_t __return;  
    asm volatile (  
        ".word (  
            (((\" #func7 \" ) & 0x7f) << 25) |  
            ((( reg_%2  ) & 0x1f) << 20) |  
            ((( reg_%1  ) & 0x1f) << 15) |  
            (((\" #funct3 \" ) & 0x07) << 12) |  
            ((( reg_%0  ) & 0x1f) <<  7) |  
            (((\" #opcode \" ) & 0x7f) <<  0)  
        );"  
        : [rd] "=r" (__return)  
        : "r" (rs1),  
          "r" (rs2)  
    );  
    __return;  
})
```

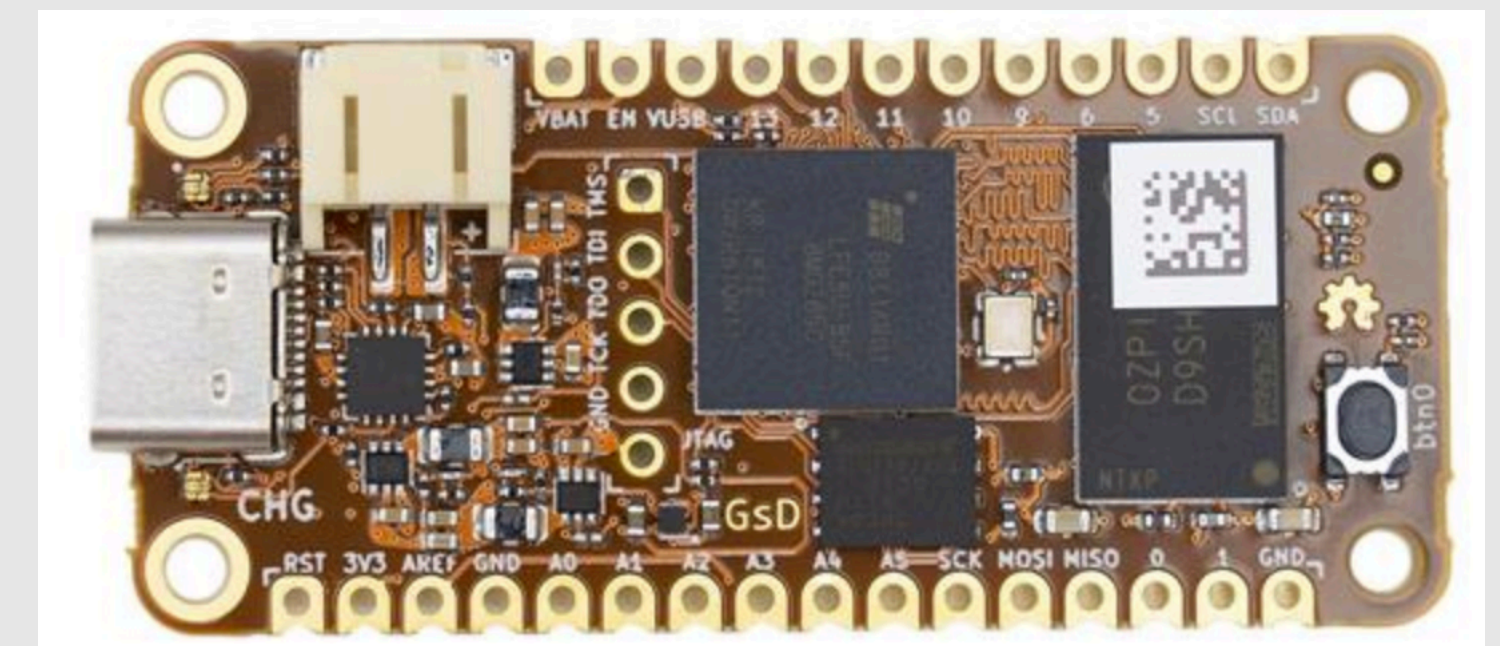
Assembler calls!

PUTTING IT ALL TOGETHER



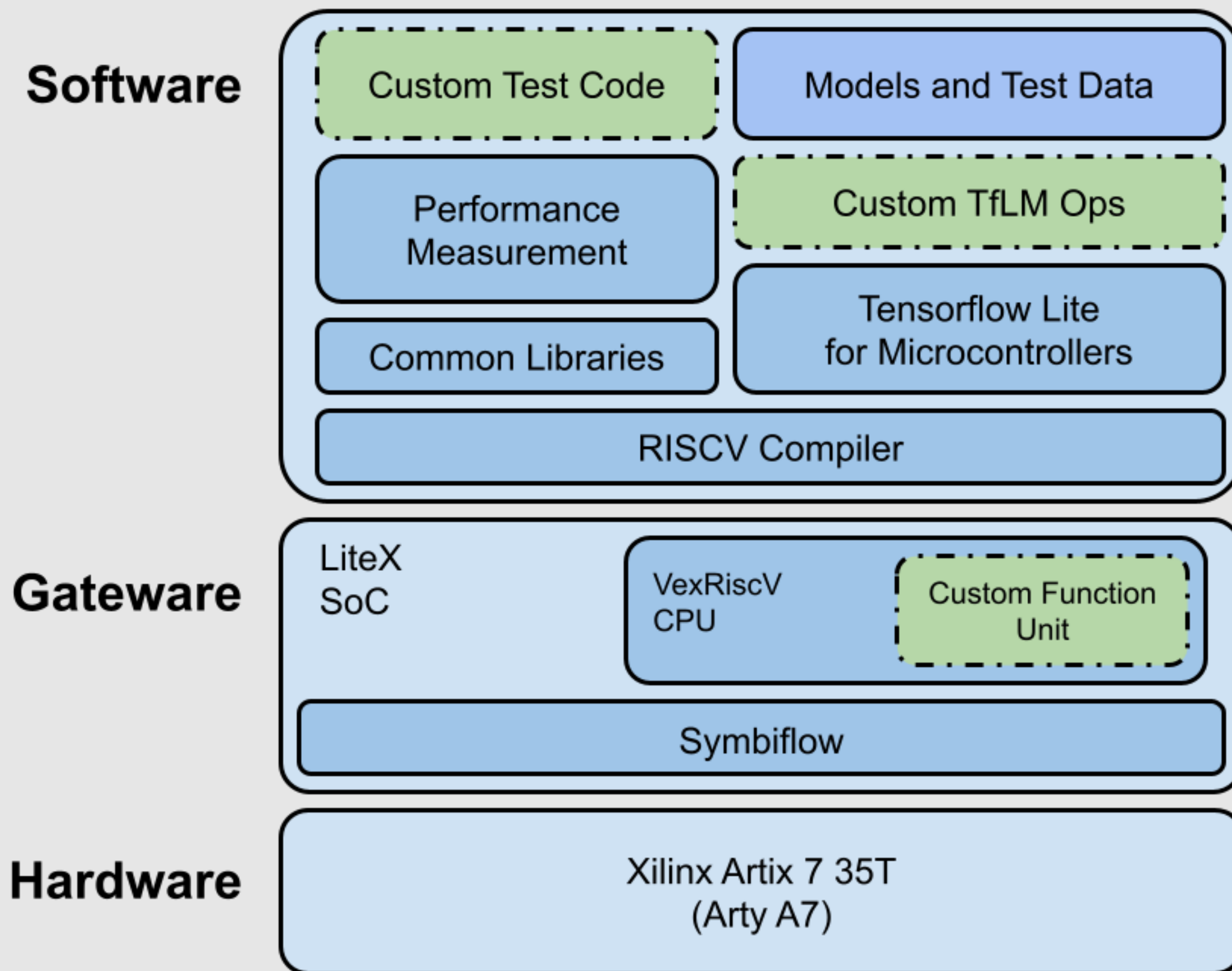
- Get board with FPGA that can fit design with CFU
- Enable CFU in board top level RTL file
- Build and flash design
- Add function calls to Zephyr from neorv32 library to call intrinsics
 - Need to pull in header/source files from neorv32 repo
- Confirm XTEA operation

<https://orangecrab-fpga.github.io/orangecrab-hardware/>



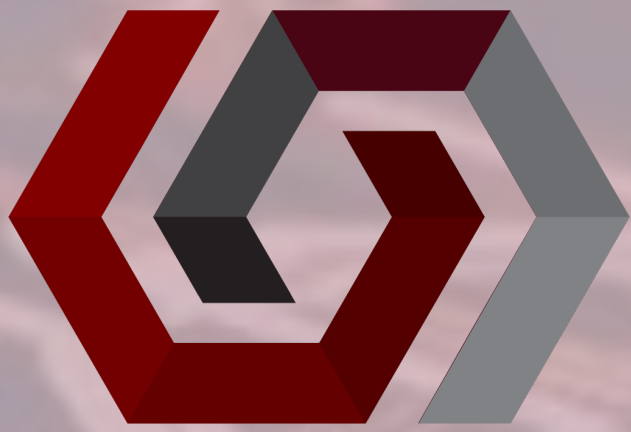


NEXT STEPS



CFU Playground Overview

- Try other hardware-accelerated functions
- CFU Playground
 - <https://cfu-playground.readthedocs.io/en/latest/>
 - Different design/implementation
 - Meant for machine learning
 - **Integrate with Zephyr and present results!**
- Custom Functions Subsystem
 - CPU independent
 - **Integrate with Zephyr and present results!**



THANK YOU!

Mohammed Billoo

MAB Labs Embedded Solutions

Open Source Summit 2024