The NEORV32 RISC-V Processor

AN INTRODUCTION

BY STEPHAN NOLTING

Project Overview

“A small, customizable and extensible MCU-class 32-bit RISC-V™ soft-core CPU and microcontroller-like SoC written in platform-independent VHDL.”

Permissive BSD-3-Clause license; free of charge

Available on GitHub™: github.com/stnolting/neorv32

Started in summer 2020, more than 7k commits since then

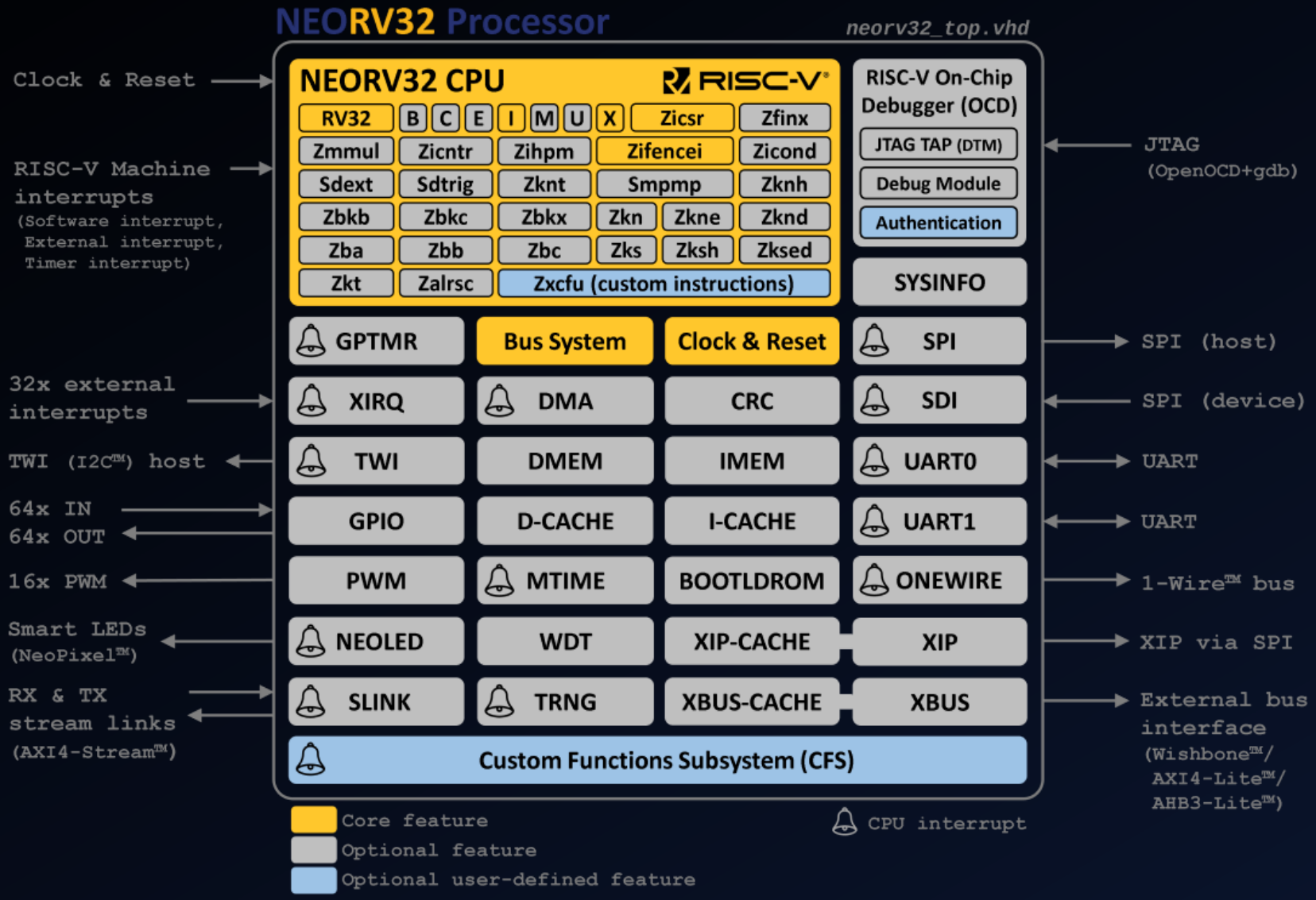
1.5k GitHub stars *(thanks to an awesome community!)*

Key Features

- all-in-one package: CPU + Processor + Software Framework
- completely described in behavioral, platform-independent VHDL
- extensive configuration options
- aims to be as small as possible while being as RISC-V-compliant as possible
- FPGA-friendly and -proven; ASIC-proven (*but proprietary so far* ☹)
- optimized for high clock frequencies to ease integration / timing closure
- from zero to `printf("hello world!");` - completely open-source and documented
- easy to use even for FPGA / RISC-V beginners – intended to *work out of the box*

Processor Overview

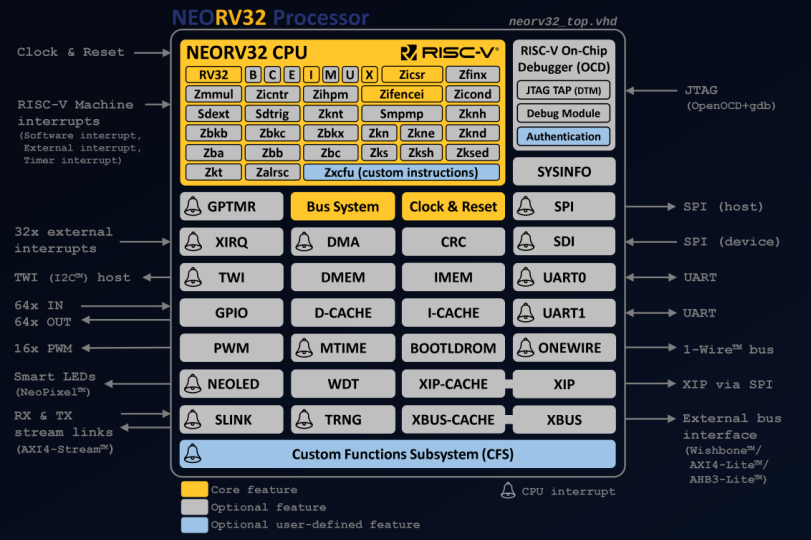
Datasheet: stnolting.github.io/neorv32/
User guide: stnolting.github.io/neorv32/ug/
PDFs: github.com/stnolting/neorv32/releases/tag/nightly



Processor version v1.10.5.6
See CHANGELOG.md

CPU Features

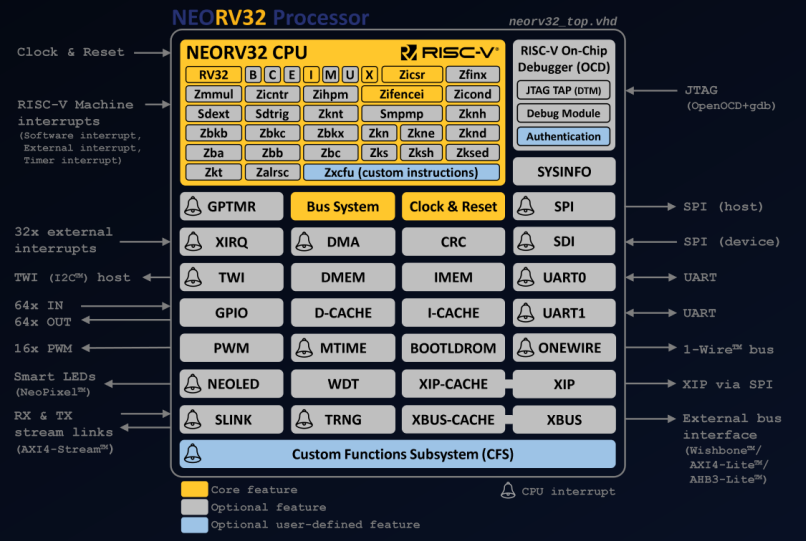
- Compatible to subsets of the RISC-V specs.
- Multi-cycle / pipelined architecture (~1 CoreMarks™/MHz)
- Modified Von-Neumann architecture
- “Execution safety” (e.g. trap on <u>every</u> illegal instruction)
- Minimal ISA configuration: `rv32e_zicsr_zifencei`
- Further <u>optional</u> RISC-V ISA extensions:
  - Compressed instructions `C`
  - Physical memory protection `Smpmp`
  - Scalar cryptography `Zk*`
  - User-mode `U`
  - Debug mode + HW breakpoints `Sd*`
  - Floating-point unit `Zfinx`
  - *and many more …*
- Tuning options (e.g. serial shifter or barrel shifter)

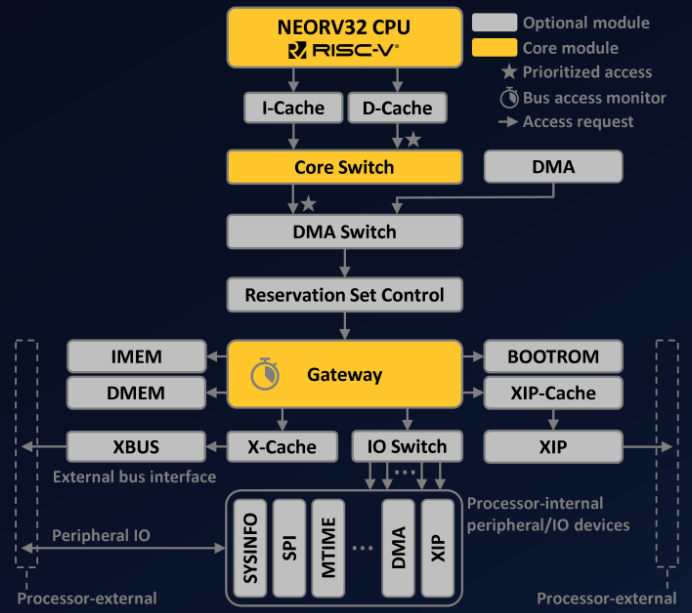


RISC-V Architecture ID 19

Processor Features

- Memories
  - Internal instruction (IMEM) & data memories (DMEM) + caches (*-CACHE)
  - Bootloader ROM (BOOTLDROM)
- Timers and counters
  - 64-bit machine timer (MTIME) + 32-bit general purpose timer (GPTMR)
  - Watchdog timer (WDT)
- Input / output
  - Standard serial interfaces (UART, SPI, TWI, ONEWIRE)
  - General-purpose IO (GPIO) and pulse-width modulation channels (PWM)
  - Native NeoPixel™ interface (NEOLED)
- Connectivity
  - External bus interface (Wishbone / AXI4-Lite™ / AHB3-Lite™) (XBUS)
  - Stream link (AXI4-Stream™) (SLINK)
  - External interrupts (32-channels) (XIRQ)
- Debugging
  - On-chip-debugger (OCD) via JTAG, compatible to openOCD & gdb
- Advanced
  - E.g. XIP, DMA, CRC, TRNG (github.com/stnolting/neoTRNG)

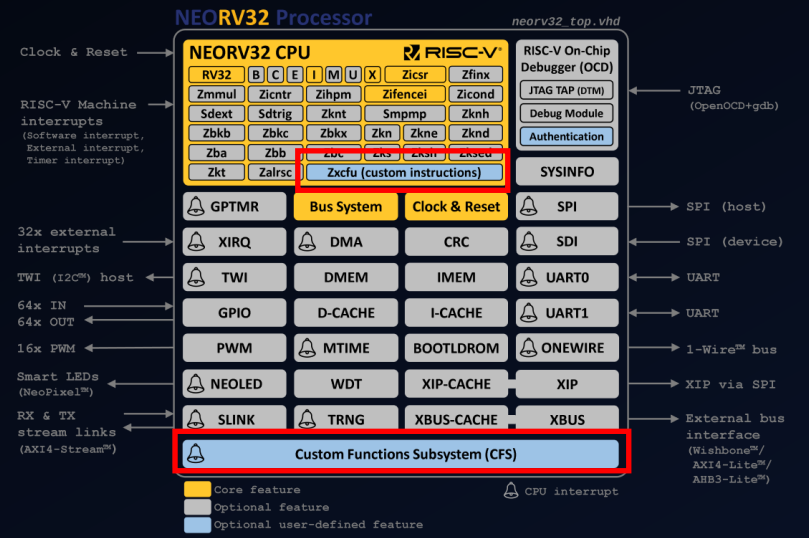




Simplified processor architecture

Custom Hardware Options

- External interfaces to attach custom IP
  - Wishbone, AXI4-Stream™, AXI4-Lite™, AHB3-Lite™
- Custom Functions Subsystem (CFS)
  - Tightly-coupled memory-mapped co-processor
  - Located inside the processor
  - HDL and software templates available
- Custom Functions Unit (CFU)
  - User-defined RISC-V instructions
  - Located inside the CPU
  - HDL and software templates available (*XTEA*)



R3- and R4-type custom instruction formats

Exemplary FPGA Implementations

Processor version v1.10.5.6
see CHANGELOG.md

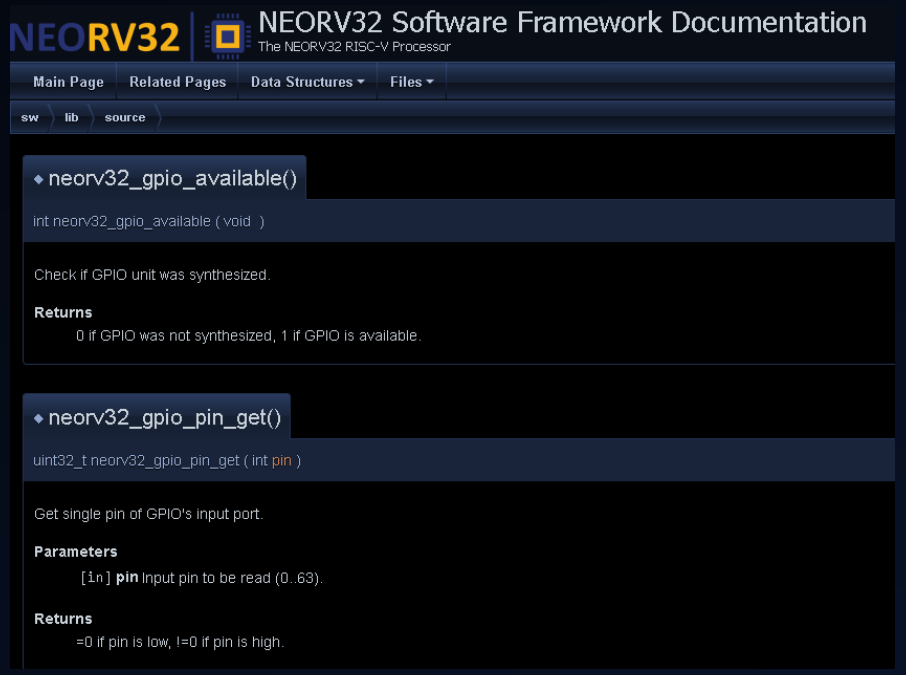
Platform	Configuration	LUTs	FFs	DSPs	Memory	*f**
Lattice™ iCE40UP5K	rv32i_zicsr_zifencei + IMEM + DMEM + BOOTLD + GPIO + MTIME + UART + SPI + TWI	3100 (58%)	2550 (48%)	0 (0%)	20x EBR (67%) 2x SPRAM (50%)	26 MHz
Intel™ Cyclone™ IV EP4CE22F17C6	All CPU extensions + all peripherals + *tuning options*	11396 (51%)	5193 (23%)	7 (5%)	369 kbits (61%)	130 MHz
AMD™ Artix™ xc7a35tics	Rv32icmbku_zicsr_zifencei + IMEM + DMEM + BOOTLD + GPIO + MTIME + UART + *tuning options*	7665 (37%)	7933 (19%)	4 (4%)	22x BRAM (43%)	150 MHz

* constrained

Preconfigured FPGA example setups (*AMD, Intel, Lattice, Gowin, Cologne Chip*)
github.com/stnolting/neorv32-setups

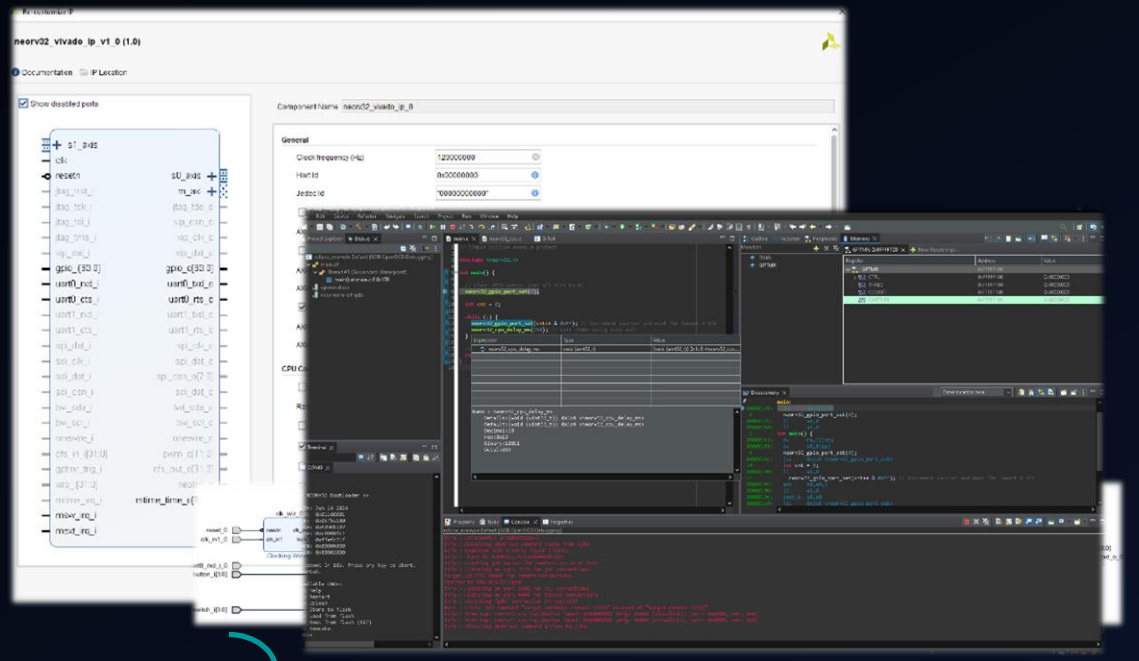
Software Framework

- Written in C (RISC-V GCC)
- HAL (hardware abstraction layer)
- Documentation based on Doxygen™
  - stnolting.github.io/neorv32/sw/files.html
- Newlib support
  - Including support for standard libraries (`malloc`, `printf`, `time`, ...)
- Example programs for all processor modules
  - github.com/stnolting/neorv32/tree/main/sw/example



Eco System

- LiteX SoC builder framework
- Vivado™ IP module
- Pre-configured Eclipse™ project
- FreeRTOS™ (and Zephyr™) port
  - github.com/stnolting/neorv32-freertos
- All-Verilog “version” (*GHDL-auto-generated*)
  - github.com/stnolting/neorv32-verilog
- RISCOF (official RISC-V compatibility checks)
  - github.com/stnolting/neorv32-riscof



Part of the project’s CI flow

Task / Subproject	Repository	CI Status
GitHub pages (docs)	neorv32	stnolting.github.io/neorv32 up
Build documentation	neorv32	Documentation passing
Processor verification	neorv32	Processor Check passing
RISCOF core verification	neorv32-riscof	neorv32-riscof passing
FPGA implementations	neorv32-setups	Implementation passing
All-Verilog version	neorv32-verilog	neorv32-verilog passing
FreeRTOS port	neorv32-freertos	neorv32-freertos sim passing
Prebuilt GCC toolchains	riscv-gcc-prebuilt	Prebuilt Toolchains passing

Thank you for your attention!

Questions?

Get in touch *(and get involved!)*

github.com/stnolting/neorv32


stnolting@gmail.com


stephan.nolting@ims.fraunhofer.de