FreeBSD/RISC-V project

Ruslan Bukin

University of Cambridge Computer Laboratory

August 25, 2016

Approved for public release; distribution is unlimited. This research is sponsored by the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory (AFRL), under contract FA8750-10-C-0237. The views, opinions, and/or findings contained in this article/presentation are those of the author(s)/presenter(s) and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

RISC-V, 2010

- David Patterson and Krste Asanović began searching for a common research ISA in 2010
- x86 and ARM: too complex, IP issues
- Decided to develop their own ISA (Summer 2010)

RISC-V, 2014

- Released frozen User Spec (v2.0) in May 2014
- Fifth RISC ISA from Berkeley, so RISC-V
- Modular ISA: Simple base instruction set plus extensions
- Less than 50 hardware instructions in the base ISA
- Designed for extension/customization

RISC-V privileged v1.9 ISA

- RV32, RV64, RV128 (32 registers)
- extensions
  - "A" atomic
  - "C" compressed, 2-byte instruction size
  - "E" embedded, 16 registers
  - "F" Single precision FPU
  - "G" General extensions, 'G' is combination IMAFD
  - "M" integer multiplication
  - ...

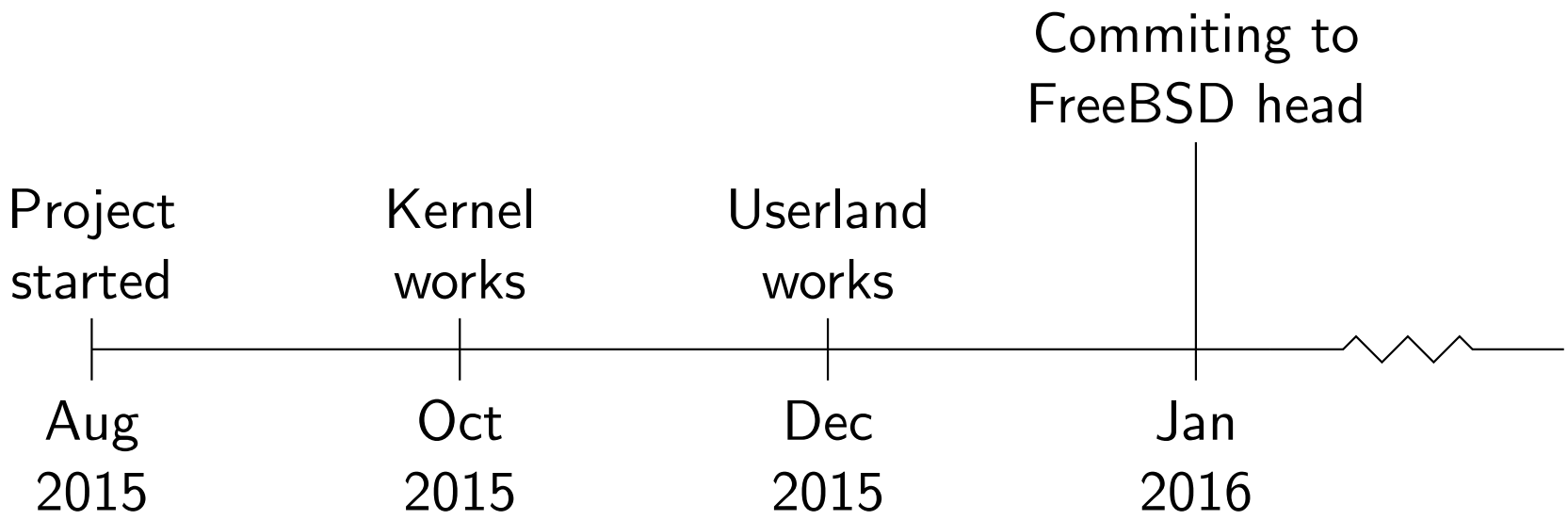
RISC-V assembly

- No conditional flags
- No branch delay slot
- Little-endian
- Branch offsets relative to PC
- 4-level pagetables
- 4 privilige levels
- 4-byte instruction length, 2-byte compressed

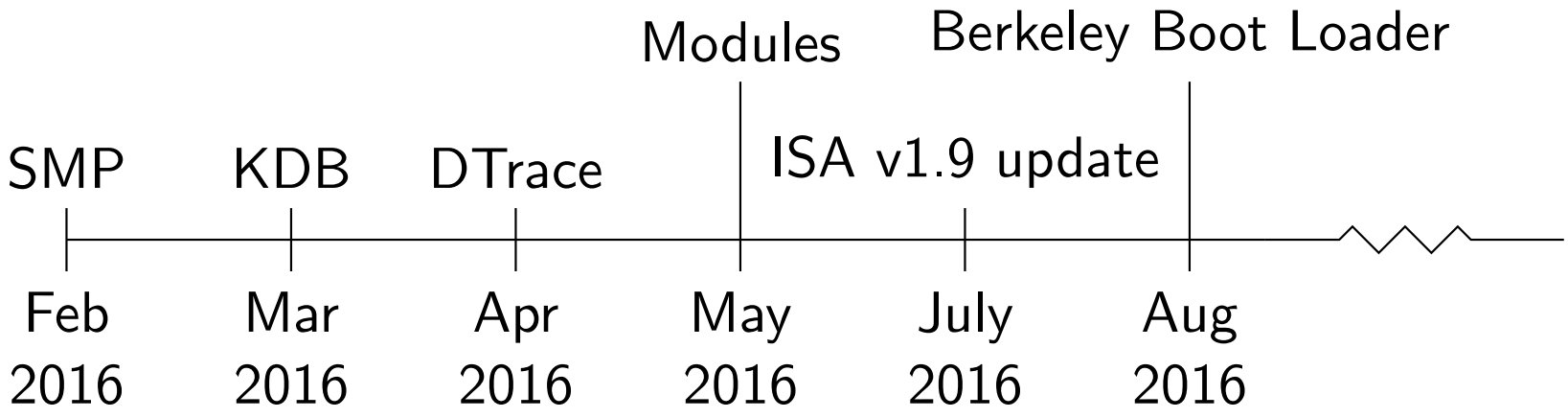
Some reasons to use FreeBSD/RISC-V

- Full stack BSD license (RISC-V, FreeBSD, LLVM/Clang)
- Technology transition
- Growing community, 42 Universities, 63 companies attended latest RISC-V workshop
- 90 RISC-V Foundation Members
  - Bluespec, Cortus, Draper, IBM, Mellanox, Google,
  - NVIDIA, HP, Microsemi, Microsoft, Oracle, Qualcomm,
  - Western Digital, AMD, etc

FreeBSD/RISC-V: first 6 months



FreeBSD/RISC-V: next 6 months



Early assembly code

1. Puts the hardware into a known state
2. Build a ring buffer for machine mode interrupts
3. Builds the initial pagetables
4. Enables the MMU
5. Branches to a virtual address (exiting from machine mode)
6. Calls into C code

Porting: kernel 1/2

- Early HTIF console device
  - EARLY_PRINTF
- Atomic inline functions
  - atomic_add(..)
  - atomic_cmpset(..)
  - atomic_readandclear(..)
  - ...
- Providing physical memory regions for VM subsystem
  - add_physmap_entry(0, 0x8000000, ..); /* 128 MB at 0x0 */
- VM (pmap)

Porting: kernel 2/2

- Exceptions, context-switching, fork_trampoline
- Timer, interrupt controller drivers
- HTIF block device driver (or use memory disk)
- copy data from/to userspace
  - fubyte, subyte, fuword, suword, sueword, fueword
  - copyin, copyout
- Trying to run /bin/sh (staticaly linked)
- Signals

FreeBSD/RISC-V: VM (pmap)

- Most sensitive machine-dependent part of VM subsystem
- Around 40 machine-dependent functions for managing page tables, address maps, TLBs
  - pmap_enter(pmap_t pmap, vm_offset_t va, vm_page_t m, vm_prot_t prot, u_int flags, int8_t psind)
  - pmap_extract(..)
  - pmap_remove(..)
  - pmap_invalidate_range(..)
  - pmap_remove_write(..)
  - pmap_protect(..)
  - pmap_activate(..)
  - pmap_release(..)
  - pmap_unwire(..)
  - ...

Porting: userspace

- jemalloc
- csu
  - crt1.S, crtn.S, crti.S
- libc
  - syscalls
  - setjmp, longjmp
  - _set_tp
- msun
- rtld-elf (runtime-linker)

Challenges

- Single page table base register
- Single tp (thread pointer) register
- Single sscratch register
- Single exceptions entry

FreeBSD/RISC-V: single page table base register problem

- Install kernel pagetables to each user pmap
- Update/delete on kernel pmap requires to update all the user pmaps

FreeBSD/RISC-V: single sscratch and tp registers

- Store PCPU on supervisor stack first
- Then store supervisor stack to sscratch register

FreeBSD/RISC-V: Exceptions entry problem

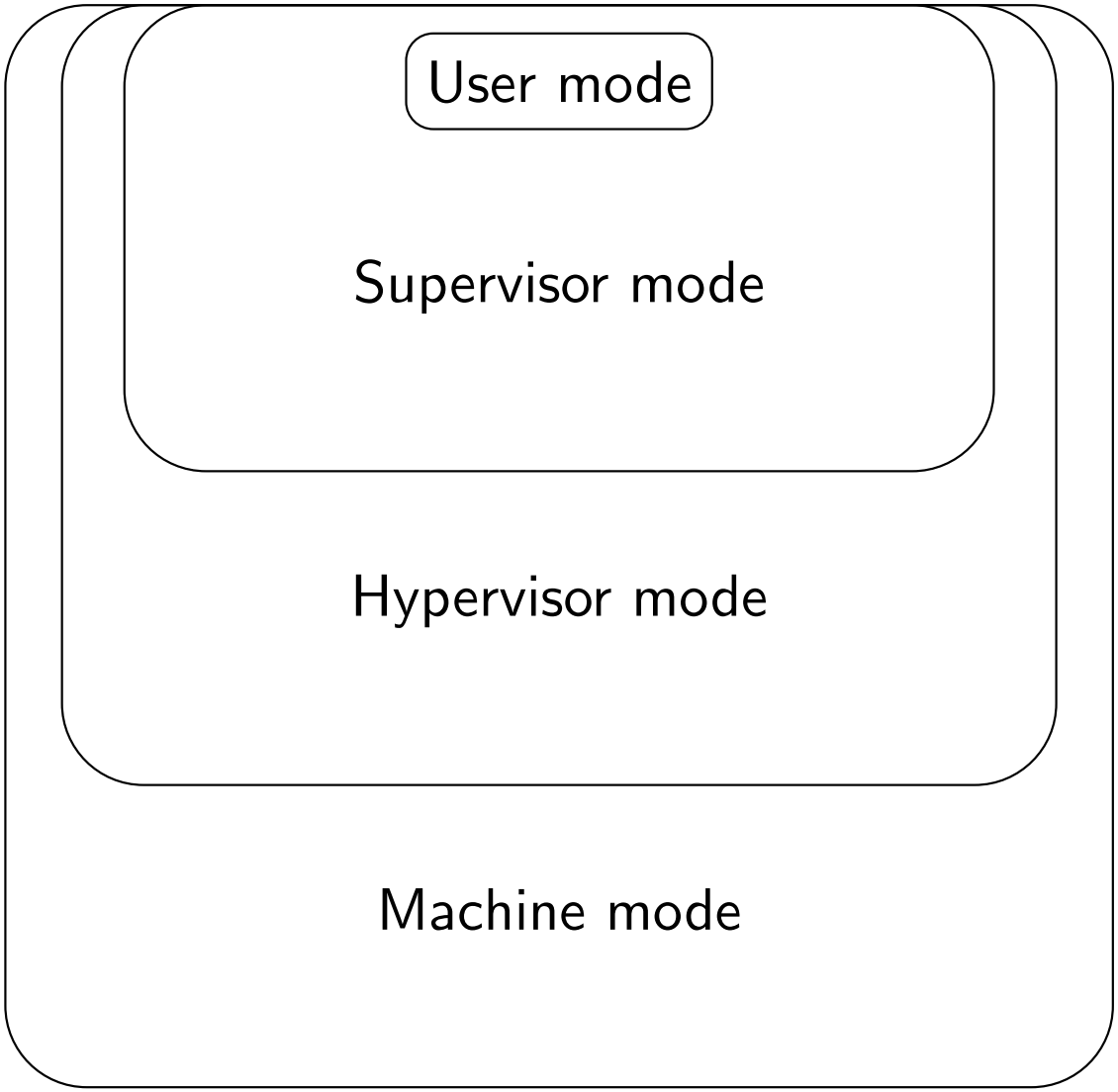
```
ENTRY(cpu_exception_supervisor)
        [..]
        call    _C_LABEL(do_trap_supervisor)
        [..]
END(cpu_exception_supervisor)

ENTRY(cpu_exception_user)
        [..]
        call    _C_LABEL(do_trap_user)
        [..]
END(cpu_exception_user)
```

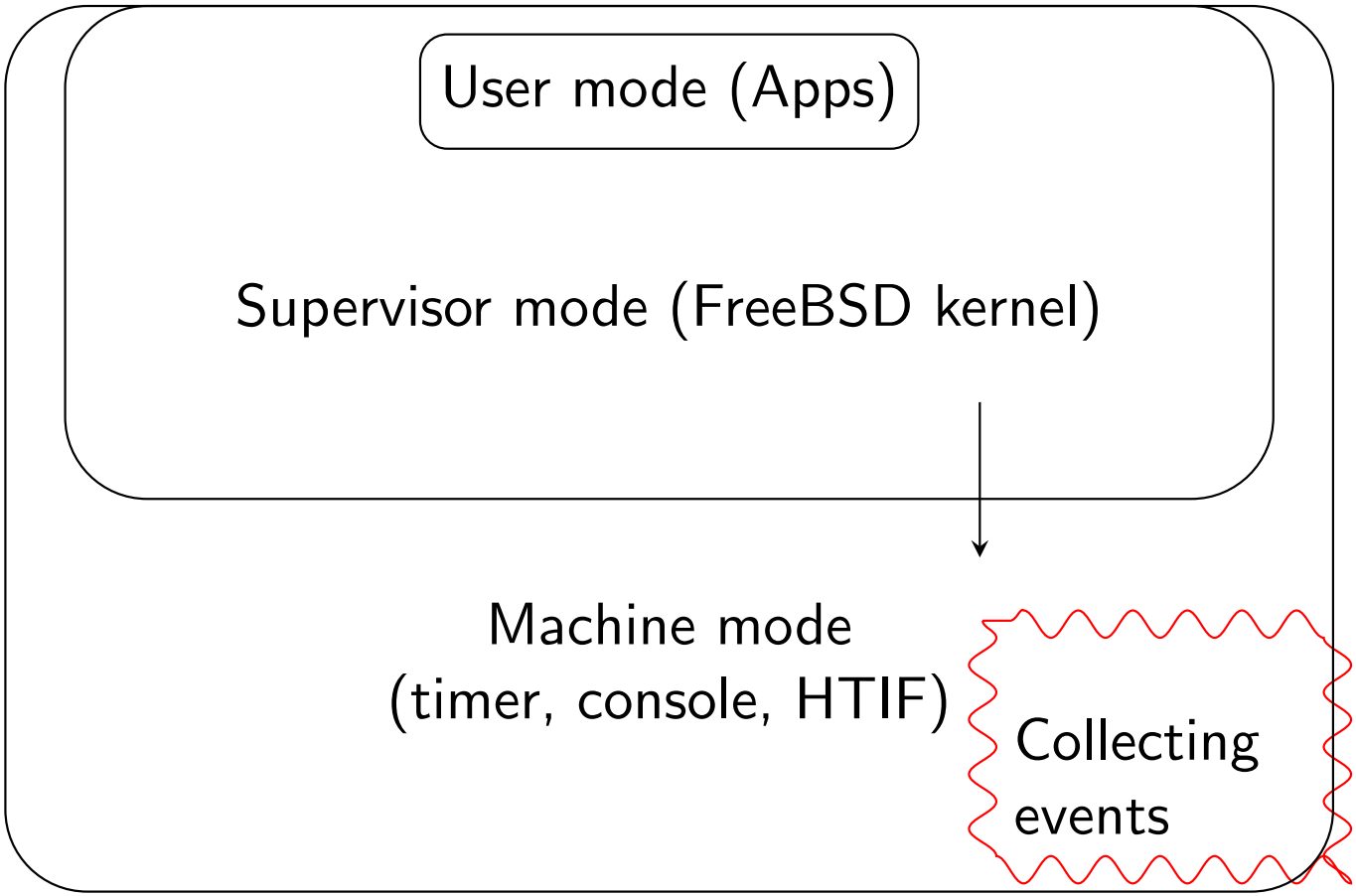
FreeBSD/RISC-V: Exceptions entry solution

```
ENTRY(cpu_exception_handler)
        csrrw   sp, sscratch, sp
        beqz    sp, 1f
        /* User mode detected */
        csrrw   sp, sscratch, sp
        j       cpu_exception_handler_user
1:
        /* Supervisor mode detected */
        csrrw   sp, sscratch, sp
        j       cpu_exception_handler_supervisor
END(cpu_exception_handler)
```

Privilege levels



Machine mode



FreeBSD/RISC-V: Berkeley Boot Loader

- Original firmware from RISC-V project

FreeBSD/RISC-V: facts

- Based on ARMv8 port
- Patch to head 25k lines (200 new files)
- External GNU toolchain used (GCC 6.1)
- 6 months from scratch

Thanks

People involved (Thanks!)

- Robert Watson (University of Cambridge)
- David Chisnall (University of Cambridge)
- Andrew Turner (ABT Systems)
- Arun Thomas (BAE Systems)
- Ed Maste (The FreeBSD Foundation)
- Yukishige Shibata (Sony Japan)
- Sean Bruno

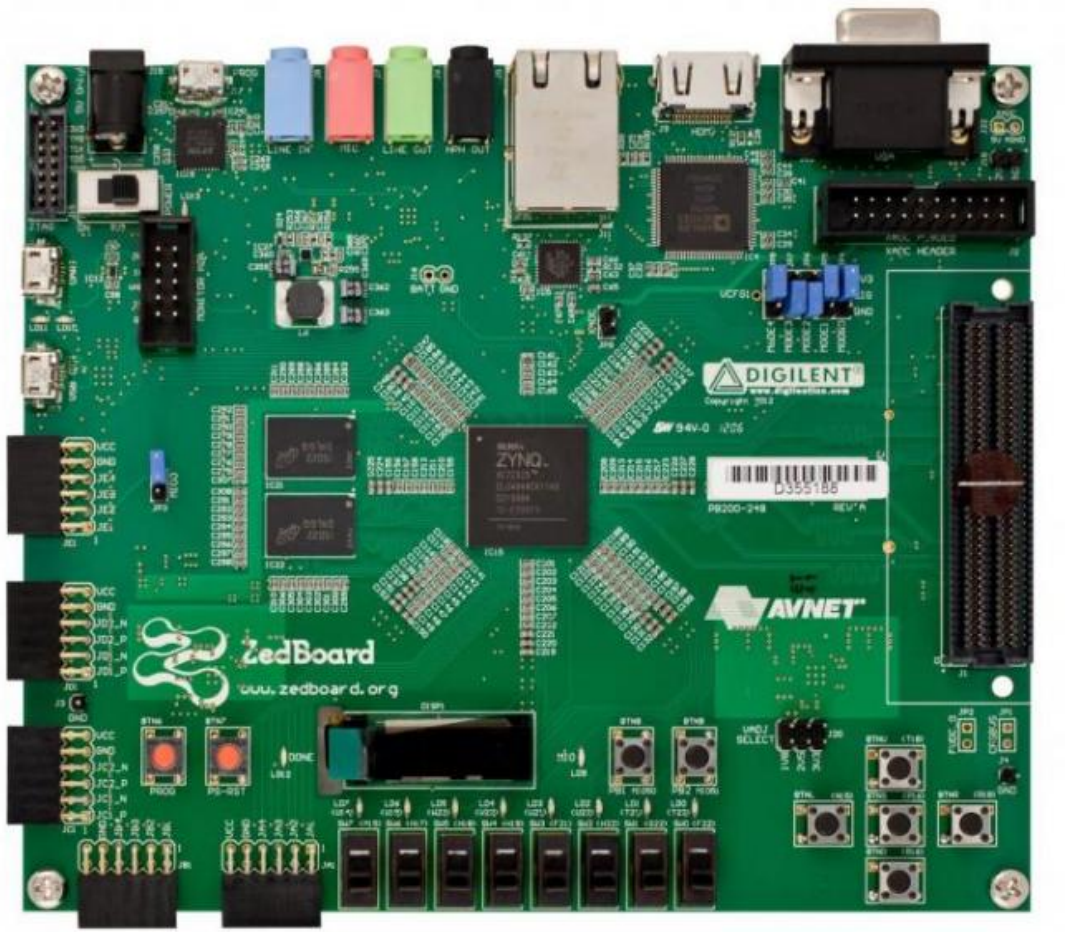
Tools

Table: Simulator/Emulator supported

	Wrotten in	FreeBSD support
Spike	C++	Yes
QEMU	C	Yes, v1.7 ISA
RocketCore FPGA	Chisel	Yes
lowRISC FPGA	Chisel	Not yet

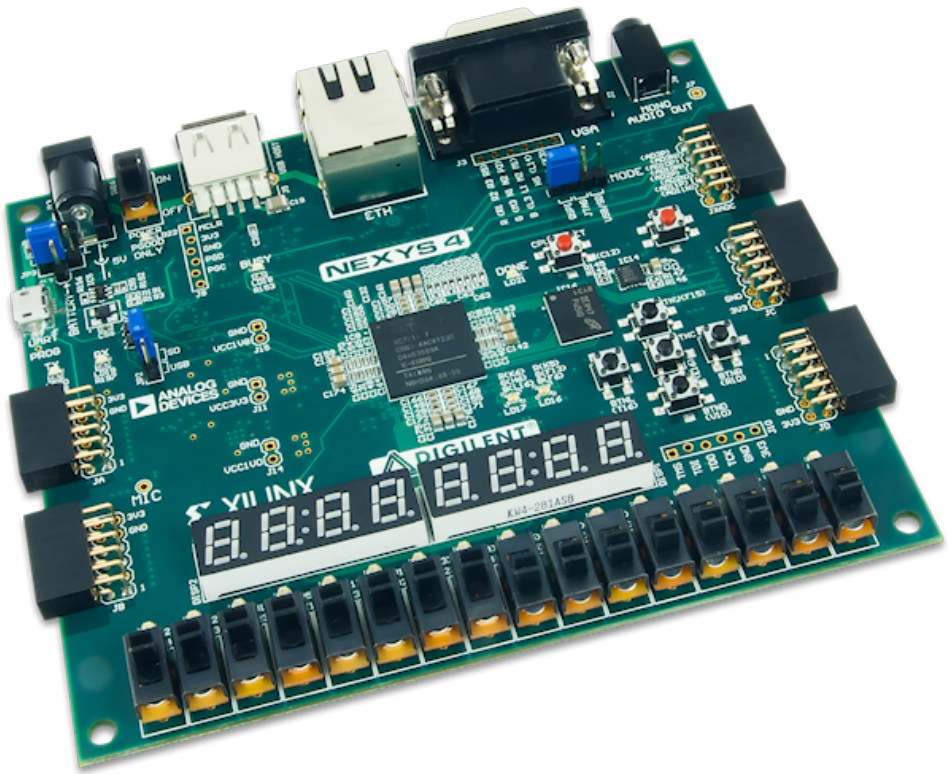
RocketChip

- 6-stage single-issue in-order pipeline
- Chisel $\rightarrow$ verilog $\rightarrow$ bitfile
- Synthesized on ZedBoard
- also SiFive Freedom U500 Platform



lowRISC

- RocketChip fork
- University of Cambridge
- Chisel $\rightarrow$ verilog $\rightarrow$ bitfile
- Synthesized on Nexys4



FreeBSD/RISC-V: Current port status

- Everything works (tm)

TODO

- Ports/packages for toolchain, for tools, for BBL
- QEMU integration for ports (usermode QEMU)
- FPU (Floating Point Unit)
- LLVM/Clang support
- FDT/non-FDT GENERIC kernel

Questions

Project home:
https://wiki.freebsd.org/riscv