Linux on RISC-V with Open Hardware

Drew Fustini (@pdp7)
<drew@beagleboard.org>
Open Source Hardware designer at OSH Park
  - PCB manufacturing service in the USA
  - <drew@oshpark.com> | twitter: @oshpark

Board of Directors, BeagleBoard.org Foundation
  - BeagleBone is a small open source hardware Linux computer
  - <drew@beagleboard.org>

Board of Directors, Open Source Hardware Association (OSHWA)
  - OSHW Certification Program: https://certification.oshwa.org/

RISC-V Ambassador for RISC-V International
  - https://riscv.org/risc-v-ambassadors/
RISC-V *(virtual)* meetups around the world

Find many more at: [https://riscv.org/local/](https://riscv.org/local/)
Upcoming Events

- **RISC-V *(Virtual)* Summit 2020**
  - December 8th to 10th
  - [https://tmt.knect365.com/risc-v-summit/](https://tmt.knect365.com/risc-v-summit/)
Berlin Embedded Linux Meetup

Berlin, Germany
140 members · Public group
Organized by Drew F. and 2 others

What we're about
Meetup for those interested in embedded Linux development and Linux kernel development.

Organizers
Drew F. and 2 others
Message
MNT Reform by MNT Research GmbH

The open source DIY laptop for hacking, customization, and privacy
Hardware whose **design** is made **publicly available** so that anyone can **study**, **modify**, **distribute**, **make**, **make**, and **sell** the design or hardware based on that design.

*(source: [Open Source Hardware (OSHW) Statement of Principles 1.0](https://www.oshw.org/principles)*)
Documentation required for electronics:

- **Schematics**
- **Board Layout**

Editable source files for CAD software such as KiCad or EAGLE

- **Bill of Materials (BoM)**

Not strict requirement, but best practice is for all components available from distributors in low quantity
CERN Open Hardware Licence

- Originally written for CERN designs hosted in the Open Hardware Repository
- Can be used by any designer wishing to share design information using a license compliant with the OSHW definition criteria.
- CERN OHL version 1.2
  Contains the license itself and a guide to its usage
Instruction Set Architecture (ISA)

- **Interface between hardware and software**
  - C++ program is compiled into instructions for a microprocessor (CPU) to execute.

- **How does compiler know what instructions the CPU understands?**
  - This is defined by the Instruction Set Architecture

- **ISA is a standard**
  - a set of rules that define the tasks the processor can perform
  - proprietary ISA’s like x86 and ARM require commercial licensing
RISC-V: a Free and Open ISA

- **History**
  - Started in 2010 by computer architecture researchers at UC Berkeley
  - Watch the [RISC-V State of the Union](https://example.com) by Krste Asanovic

- **Why “RISC”?**
  - RISC = Reduced Instruction Set Computer

- **Why “V”?**
  - 5th RISC instruction set to come of out UC Berkeley

- **Why is it “Free and Open”?**
  - Specifications licensed as Creative Commons Attribution 4.0 International
What is different about RISC-V?

- **Simple, clean-slate design**
  - Far smaller than other commercial ISAs
  - Clear separation between unprivileged and privileged ISA
  - Avoids micro-architecture or technology dependent features

- **Modular ISA designed for extensibility and specialization**
  - Small standard base, with multiple standard extensions
  - Suitable for everything from tiny microcontrollers to supercomputers

- **Stable**
  - Base and standard extensions are frozen
  - Additions via optional extensions, not new versions of base ISA

(source: *Instruction Sets Want to be Free* (Krste Asanović))
RISC-V Base Integer ISA

- **RV32I**: 32-bit
  - less than 50 instructions needed!

- **RV32E**: 32-bit embedded
  - reduces register count from 32 to 16 for tiny microcontrollers

- **RV64I**: 64-bit

- **RV128I**: 128-bit
  - Future-proof for nonvolatile RAM capacity; benefits security research

(source: RISC-V Summit 2019: State of the Union)
RISC-V base plus standard extensions

- Standard extensions
  - M: integer multiply/divide
  - A: atomic memory operations
  - F, D, Q: floating point, double-precision, quad-precision
  - G: “general purpose” ISA, short-hand for IMAFD
  - C: compressed instruction encoding to conserve memory and cache like ARM Thumb
  - Additions via optional extensions like Vector but not new versions of base ISA
  - Linux distros like Debian and Fedora target RV64GC

- Frozen in 2014, ratified 2019, will be supported forever

(source: RISC-V Summit 2019: State of the Union)
RV32I / RV64I / RV128I + M, A, F, D, Q, C
RISC-V “Green Card”

### RISC-V Reference Card

<table>
<thead>
<tr>
<th>Category Name</th>
<th>Fixed Word (F)</th>
<th>Optional Word (D, I)</th>
<th>Optional Compressed Word (Q)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 Optional FP Extensions: RV32(F)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 Optional FP Extensions: RV64(I)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 Optional FP Extensions: RV128(Q)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Arithmetic</td>
<td>ADD, OR, ADDI</td>
<td>ADD, ADDI, ADDI</td>
<td>ADD, ADDI</td>
</tr>
<tr>
<td>Logical</td>
<td>OR, AND, XOR</td>
<td>OR, AND, XOR</td>
<td>OR, AND, XOR</td>
</tr>
<tr>
<td>Configuration</td>
<td>Load, Load Res</td>
<td>Load, Load Res</td>
<td>Load, Load Res</td>
</tr>
<tr>
<td>System Status</td>
<td>Store Status</td>
<td>Store Status</td>
<td>Store Status</td>
</tr>
<tr>
<td>Jump &amp; Link</td>
<td>Jump, Branch</td>
<td>Jump, Branch</td>
<td>Jump, Branch</td>
</tr>
<tr>
<td>16-bit (RVC)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32-bit Instruction Formats</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(source: Hot Chips Tutorial, Part 1: RISC-V overview and ISA design, Krste Asanovic)
Learn more about RISC-V

- Get up-to-speed quick with the RISC-V Reader

riscvbook.com
RISC-V and Industry

- **RISC-V International now controls the specifications:** [riscv.org](http://riscv.org)
  - Non-profit organization with 690+ members from 50 countries including companies, universities and more
  - [Become a member](http://riscv.org) *(free of cost to individuals and non-profits)*
  - [YouTube channel has hundreds of talks!](http://youtube.com)

- **Companies plan to ship billions of devices with RISC-V cores**
  - Nvidia already shipping RISC-V cores for system management in its GPU products
  - Western Digital will be using RISC-V controllers in all of its storage products
RISC-V and Industry

- **Avoid ISA licensing and royalty fees**
  - including the legal costs and long delays due to complex licensing agreements

- **Freedom to choose micro-architecture implementation**
  - only a few companies like Apple, Samsung, and Qualcomm have ARM architecture licenses which allows them to do a custom implementation

- **Freedom to leverage existing open source implementations**
  - Berkeley’s Rocket and BOOM, ETH Zurich’s PULP cores, Western Digital SweRV

- **Already has a well supported software ecosystem**
  - Linux, BSD, gcc, glibc, LLVM/clang, FreeRTOS, Zephyr, QEMU
  - [The State of Software Development Tools for RISC-V](https://example.com) by Khem Raj
RISC-V around the world

- **RISC-V International** based in Switzerland
  - U.S.-based RISC-V Foundation reincorporated at the beginning of 2020 as RISC-V International in Switzerland to avoid being hampered by U.S. politics

- **EU, India and Pakistan have RISC-V processor design initiatives**
  - Desire for sovereign control of technology and avoid backdoors from other nations

- **Strong interest from chipmakers in China**
  - U.S. companies banned in 2019 from doing business with Huawei... who’s next?
  - ARM was deemed to be a UK-origin technology in 2019, so it is ok to do business with Huawei... but how long will that last? Will the Nvidia acquisition impact that?
Does RISC-V mean Open Source?

- RISC-V is a set of specifications under an open source license

- RISC-V implementations can be open source or proprietary

- Open specifications make open source implementations possible
  - It is not legal to design an open source processor for proprietary ISA like x86 and ARM
RISC-V Privileged Architecture

- **Three privilege modes**
  - User (U-Mode): applications
  - Supervisor (S-Mode): OS kernel
  - Machine (M-Mode): bootloader and firmware

- **Supported combinations of modes**
  - M (simple embedded systems)
  - M, U (embedded systems with memory protection)
  - M, S, U (Unix-style operating systems with virtual memory)

- **Hypervisors run in modified S mode (HS)**

(source: [Co-developing RISC-V Hypervisor Support](https://example.com), Anup Patel)
RISC-V Boot Flow

- Follows commonly used multiple boot stages model
  - ZSBL and FSBL are initial platform-specific bootloaders (*SiFive FU540 SoC in this example*)
  - U-Boot is the final stage bootloader that jumps into Linux kernel
  - NOTE: *hart is a hardware thread of execution*, which users may refer to as a “core”

(source: *RISC-V software ecosystem in 2020*, Atish Patra)
What is SBI?

- **SBI stands for Supervisor Binary Interface**
  - calling convention between Supervisor (S-mode OS) and Supervisor Execution Environment (SEE)
  - allows supervisor-mode software to be written that is portable to all RISC-V implementations

- **Unix-class Platform Spec working group**
  - Chaired by Al Stone
  - Transitioning to [RISC-V Profiles and Platform Spec WG](https://www.riscv.org/)

*(source: [OpenSBI Deep Dive](https://www.opensbi.org/), Anup Patel)*
What is OpenSBI?

- OpenSBI is an open source SBI implementation
  - avoid fragmentation of SBI implementations

- Layers of implementation
  - Platform specific reference firmware
  - Platform specific library
  - SBI library

- Provides run-time in M-mode
  - Typically used in boot stage following ROM/Loader
  - Provides support for reference platforms
  - Generic simple drivers included for M-mode to operate

(source: OpenSBI Deep Dive, Anup Patel)
UEFI Support

- **UEFI support for RISC-V coming in Linux 5.10** *(ETA December 2020)*

- **Grub2** and **U-Boot** support UEFI on RISC-V

- **RISC-V edk2 port** is upstream in TianoCore

(source: *Introduction to RISC-V Boot Flow*, Atish Patra and Anup Patel)
RISC-V emulation in QEMU

- **Support for** RISC-V in mainline QEMU
  - QEMU can boot 32-bit and 64-bit mainline Linux kernel
  - QEMU can run OpenSBI, U-Boot and Coreboot
  - Draft versions of Hypervisor and Vector extensions supported
  - QEMU sifive_u machine can boot same binaries as the physical board

- **Tutorial:** [Running 64- and 32-bit RISC-V Linux on QEMU](source: OpenSBI Deep Dive, Anup Patel)
RISC-V in the Linux kernel

- Initial port by Palmer Dabbelt landed in Linux 4.15
  - Mailing list: linux-riscv@lists.infradead.org (archive)

- “What's missing in RISC-V Linux, and how YOU can help!”
  - Björn Töpel at Munich RISC-V meetup (jump to 43:25)
  - “A great way to learn the nitty gritty details of the Linux kernel”
  - “It’s a fun, friendly, and still pretty small community”

(source: “What's missing in RISC-V Linux, and how YOU can help!”, Björn Töpel)
RISC-V in the Linux kernel

commits arm64 vs riscv

(source: “What’s missing in RISC-V Linux, and how YOU can help!”, Björn Töpel)
RISC-V in the Linux kernel

commits arm64 vs riscv (time shifted)

(source: “What’s missing in RISC-V Linux, and how YOU can help!”, Björn Töpel)
RISC-V in the Linux kernel

$ ./Documentation/features/list-arch.sh riscv | grep TODO

core/ cBPF-JIT : TODO | HAVE_CBPF_JIT # arch supports cBPF JIT optimizations
debug/ kprobes : TODO | HAVE_KPROBES # arch supports live patched kernel probe
debug/ kprobes-on-ftrace : TODO | HAVE_KPROBES_ON_FTRACE # arch supports combined kprobes and ftrace live patching
debug/ kretprobes : TODO | HAVE_KRETPROBES # arch supports kernel function-return probes
debug/ optprobes : TODO | HAVE_OPTPROBES # arch supports live patched optprobes
debug/ uprobes : TODO | ARCH_SUPPORTS_UPROBES # arch supports live patched user probes
debug/ user-ret-profiler : TODO | HAVE_USER_RETURN_NOTIFIER # arch supports user-space return from system call profiler
locking/ cmpxchg-local : TODO | HAVE_CMPXCHG_LOCAL # arch supports the this_cpu_cmpxchg() API
locking/ queued-rwlocks : TODO | ARCH_USE_QUEUE_RWLOCKS # arch supports queued rwlocks
locking/ queued-spinlocks : TODO | ARCH_USE_QUEUE_SPINLOCKS # arch supports queued spinlocks
perf/ kprobes-event : TODO | HAVE_REGS_AND_STACK_ACCESS_API # arch supports kprobes with perf events
sched/ membarrier-sync-core : TODO | ARCH_HAS_MEMBARRIER_SYNC_CORE # arch supports core serializing membarrier
sched/ numa-balancing : TODO | ARCH_SUPPORTS_NUMA_BALANCING # arch supports NUMA balancing
time/ arch-tick-broadcast : TODO | ARCH_HAS_TICK_BROADCAST # arch provides tick broadcast()
time/ irq-time-acct : TODO | HAVE_IRQ_TIME_ACCOUNTING # arch supports precise IRQ time accounting
time/ virt-cpuacct : TODO | HAVE_VIRT_CPU_ACCOUNTING # arch supports precise virtual CPU time accounting
vm/ ELF-ASLR : TODO | ARCH_HAS_ELFRANDOMIZE # arch randomizes the stack, heap and binary images of ELF binaries
vm/ huge-vmap : TODO | HAVE_ARCH_HUGE VMAP # arch supports the ioremap_pud_enabled() and ioremap_pmd_enabled() APIs
vm/ ioremap_prot : TODO | HAVE_IOREMAP PROT # arch has ioremap_prot()
vm/ PG_uncached : TODO | ARCH_USES_PG UNCACHED # arch supports the PG_uncached page flag
vm/ THP : TODO | HAVE_ARCH_TRANSPARENT_HUGEPAGE # arch supports transparent hugepages
vm/ batch-unmap-tlb-flush : TODO | ARCH_WANT_BATCHED_UNMAP_TLB_FLUSH # arch supports deferral of TLB flush until multiple pages are unmapped

(source: “What's missing in RISC-V Linux, and how YOU can help!”, Björn Töpel)
RISC-V in the Linux kernel

- Recent and ongoing work:
  - KVM (Anup Patel/Atish Patra)
    - waiting on ratification of Hypervisor spec
  - eBPF JIT (Björn Töpel)
  - KGDB support (Vincent Chen)
  - kexec/kdump support (Nick Kossifidis)
  - kprobes/kretprobes (Guo Ren)
  - generic vDSO support
  - syszcaller support
  - build with LLVM/clang

(source: “What's missing in RISC-V Linux, and how YOU can help!”, Björn Töpel)
Linux distro: Fedora

- "This project, informally called Fedora/RISC-V, aims to provide a complete Fedora experience on the RISC-V (RV64GC)"

(source: Fedora on RISC-V, Wei Fu)
Linux distro: Debian

- QEMU and libvirt/QEMU
  - Fedora Images can run on the QEMU with graphics parameters (VGA and bochs-display).

- SiFive Unleashed board
  - Fedora GNOME Image can run on SiFive Unleashed with Expansion Board, PCI-E graphic Card & SATA SSD

- Installation instructions

(source: Fedora on RISC-V, Wei Fu)
Port of Debian for the RISC-V architecture called riscv64

- “a port in Debian terminology means to provide the software normally available in the Debian archive (over 20,000 source packages) ready to install and run”

- 95% of packages are built for RISC-V

  - The Debian port uses RV64GC as the hardware baseline and the lp64d ABI (the default ABI for RV64G systems).
OpenEmbedded / Yocto

- **meta-risc-v**: general hardware-specific BSP overlay for the RISC-V
  - The core BSP part of meta-riscv should work with different OpenEmbedded/Yocto distributions and layer stacks
  - Supports QEMU and the SiFive HiFive Unleashed board
BuildRoot

- RISC-V port is now supported in the upstream BuildRoot project
- “Embedded Linux from scratch in 40 minutes (on RISC-V)”
  - Tutorial by Michael Opdenacker, Bootlin
  - Hardware emulator: QEMU
  - Cross-compiling toolchain: Buildroot
  - Bootloader: BBL Berkeley Boot Loader
  - Kernel: Linux 5.4-rc7
  - Root filesystem and application: BusyBox
  - That’s easy to compile and assemble in less than 40 minutes!
SiFive is a start-up founded by members of the Berkeley RISC-V team

FU540 debuted in 2018 as the first RISC-V SoC that could run Linux

- 4x U54 cores (up to 1.5 GHz) which implement RV64GC to run Linux
- 1x E51 low-power “minion” core for system management tasks
- 64-bit DDR4 with ECC
- Gigabit Ethernet, ChipLink, SPI, I2C, UART, GPIO, PWM (no USB)
SiFive Freedom Unleashed

- **The first **[Linux-capable RISC-V dev board](#) **dev board**
  - And the board design is Open Source Hardware!
- **Highest performance available yet**
  - FU540 SoC clocked over 10x faster than FPGA ‘soft’ cores
- **Too expensive for widespread adoption**
  - Sold for $999 on [CrowdSupply](#) and no longer available
  - FU540 SoC chip is not sold separately
  - SiFive core business is designing cores, not SoC’s or boards

*NOTE: ASIC is a term often used to indicate that an SoC (System-on-Chip) has a “hard” processor core constructed by silicon fabrication instead of “soft” core on FPGA where clock speeds are much lower*
SiFive Freedom Unleashed

- Fedora GNOME image running on Unleashed with PCIe graphics card
Microchip PolarFire SoC

- Microchip designed a SoC similar to SiFive U540 but adds a FPGA
  - 4x 667 MHz U54 cores, 1x E51 core
  - PolarFire FPGA fabric with 25k to 460k logic elements (LEs)
  - DDR3/4, LPDDR3/4
  - PCIe Gen2, USB 2.0 OTG, 2x GbE
- **Full commercial product family**
  - Available from distributors
  - Formerly branded as Microsemi before Microchip acquired it
Microchip Icicle board

- PolarFire SoC dev board
  - $499 on CrowdSuppy
  - Now shipping to backers
  - Available soon from distributors

- MPFS250T-FCVG484EES
  - 600 MHz clock RISC-V cores
  - 254K logic element FPGA

- Memory
  - 2 GB LPDDR4 x 32
  - 1 Gb SPI flash
  - 8 GB eMMC flash or SD card slot
Kendryte K210

- **400MHz dual core RV64GC**
  - 8MB SRAM but no DRAM interface

- **Affordable Sipeed dev boards**
  - Sipeed MAix BiT is only $13

- **Full support added in Linux 5.8**
  - “RISC-V NOMMU and M-mode Linux”
  - Damien Le Moal, Christoph Hellwig

- **2 boards supported by u-boot**
  - Sean Anderson
Kendryte K210

- **Buildroot with busybox** for rootfs
  - upstreaming in progress on the mailing list
  - tutorial from CNX Software

- **8MB runs out very quick!**
  - MMU based on draft spec not supported by Linux
  - userspace needs shared library support
  - "RISC-V FDPIC/NOMMU toolchain/runtime support"
    (Maciej W. Rozycki)
Kendryte K210
PicoRio

- Open source project from RIOS Lab
  - Goal is to create low-cost Linux-capable RISC-V platform

- Introduction by Zhangxi Tan
  - During RISC-V Global Summit back in September
  - Three phases of PicoRio planned
  - Samples of PicoRio 1.0 expected in Q4 2020
SiFive RISC-V PC

- **Powered by next-gen** [SiFive Freedom U740 SoC](#)
  - complete implementation of the latest RISC-V Vector (RVV) extension.

- **Details at the** [Linley Fall Processor Conference](#) **on October 29th**
  - “Extending AI SoC Design Possibilities Through Linux-Capable Vector Processors”, *Krstev Asanović, Cofounder & Chief Architect, SiFive*
  - “Creating a RISC-V PC Ecosystem for Linux Application Development”, *Yunsup Lee, CTO, SiFive*
Alibaba XuanTie 910

- **T-Head** is a subsidiary of Alibaba
- **16-core 2.5 GHz RISC-V processor**
- **implementation** of current draft RISC-V Vector (RVV) extension
- Expected to debut in 2021
Open source FPGA toolchains

Where do FPGAs Come In?

- Field Programmable Gate Array
- Change a chip's HARDWARE in a few minutes
- Make it act like a new chip!

“RISC-V and FPGAs: Open Source Hardware Hacking”
Keynote at Hackday Supercon 2019 by Dr. Megan Wachs
Open source FPGA toolchains

- Project IceStorm for Lattice iCE40 FPGA
  - “A Free and Open Source Verilog-to-Bitstream Flow for iCE40 FPGAs”
  - Claire Wolf (oe1cxw) at 32c3
Open source FPGA toolchains

- Project Trellis for the more capable Lattice ECP5 FPGA
  - “Project Trellis and nextpnr FOSS FPGA flow for the Lattice ECP5”
  - David Shah @fpga_dave at FOSDEM 19
Open source FPGA toolchains

- Project X-Ray & SymbiFlow for much more capable Xilinx Series 7
  - “Xilinx Series 7 FPGAs Now Have a Fully Open Source Toolchain!” [almost] Tim Ansell
  - “Open Source Verilog-to-Bitstream FPGA synthesis flow, currently targeting Xilinx 7-Series, Lattice iCE40 and Lattice ECP5 FPGAs. Think of it as the GCC of FPGAs”
Hackaday Supercon badge

- RISC-V “soft” core on ECP5 FPGA
- Gigantic FPGA In Game Boy Form Factor
“Team Linux on Badge”

- Michael Welling, Tim Ansell, Sean Cross, Jacob Creedon
- Attempt to use the built-in 16MB failed...
“Team Linux on Badge”

- Jacob Creedon designed a cartridge board that adds 32MB of SDRAM to the Hackaday Supercon badge... before the event!
This is how a Linux capable core looks like on an FPGA. #nextpnratwork

source: https://twitter.com/ico_TC/status/1205996588499210241
Why design an SoC in Python?

● Python has advantages over traditional HDL like VHDL and Verilog
  ○ Many people already are familiar with Python than HDL (hardware description languages)
  ○ There are currently more software developers than hardware designers
● **Migen** is a Python framework that can automate chip design
  ○ Leverages the object-oriented, modular nature of Python
  ○ Produces Verilog code so it can be used with existing chip design workflows
● “Using Python for creating hardware to record FOSS conferences!”
What is Migen?

An alternative HDL based on Python

Migen

source: https://github.com/litex-hub/fpga_101
LiteX

- Based on Migen, builds full SoC that can be loaded into an FPGA
LiteX

- "LiteX vs. Vivado: First Impressions"
- Collection of open cores for DRAM, Ethernet, PCIe, SATA and more...

<table>
<thead>
<tr>
<th>Name</th>
<th>Build Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LiteDRAM</td>
<td>build passing</td>
<td>DRAM</td>
</tr>
<tr>
<td>LiteEth</td>
<td>build passing</td>
<td>Ethernet</td>
</tr>
<tr>
<td>LitePCIe</td>
<td>build passing</td>
<td>PCIe</td>
</tr>
<tr>
<td>LiteSATA</td>
<td>build passing</td>
<td>SATA</td>
</tr>
<tr>
<td>LiteSDCard</td>
<td>build passing</td>
<td>SD card</td>
</tr>
<tr>
<td>LiteICLink</td>
<td>build passing</td>
<td>Inter-Chip communication</td>
</tr>
<tr>
<td>LiteJESD204B</td>
<td>build passing</td>
<td>JESD204B</td>
</tr>
<tr>
<td>LiteVideo</td>
<td>build unknown</td>
<td>VGA, DVI, HDMI</td>
</tr>
<tr>
<td>LiteScope</td>
<td>build passing</td>
<td>Logic analyzer</td>
</tr>
</tbody>
</table>
Linux on LiteX-VexRiscv

- **VexRiscv**: 32-bit Linux-capable RISC-V core
  - Designed to be FPGA friendly
  - Written in Spinal HDL (based on Scala)

- **Builds an SoC using VexRiscv core and LiteX modules**
  - Such as LiteDRAM, LiteEth, LiteSDCard, LitePCIe
  - “This project demonstrates how high level HDLs (Spinal HDL, Migen) enable new possibilities and complement each other. Results shown here are the results of a productive collaboration between open-source communities”

- **Supports large number of FPGA dev boards**
Add the Hackaday Supercon 2019 badge which has an ECP5 FPGA.

These changes are from a fork by Michael Welling (@mwelling).

During Supercon, we tried two approaches:

- use the built-in 16MB QSPI SRAM
- use add-on cartridge with 32MB SDRAM by Jacob Creedon

We were not able to get the QSPI SRAM working so I've removed those changes, and I have just added the changes that are needed to boot Linux with the 32MB SDRAM.

In addition to @mwelling, thank you to Jacob Creedon (@jcreedon), @gregdavill, Tim Ansell (@mithro), and Sean Cross (@xobs) who all helped get Linux working on this badge.

KiCad design files by @jcreedon for the SDRAM cartridge are available on GitHub.
add the Hackaday Supercon ECP5 badge #31
Changes from all commits ▼ File filter... ▼ Jump to... ▼

```
from litex.build.generic_platform import *
from litex.build.lattice import LatticePlatform

# IOs -----------------------------

_io = [
    ("clk8", 0, Pins("U18"), IOStandard("LVCMOS33")),
    ("programm", 0, Pins("R1"), IOStandard("LVCMOS33")),
    ("serial", 0,
      Subsignal("rx", Pins("U2"), IOStandard("LVCMOS33"), Misc("PULLMODE=UP")),
      Subsignal("tx", Pins("U1"), IOStandard("LVCMOS33")),
    ),
    ("led", 0, Pins("E3 D3 C3 C4 C2 B1 B20 B19 A18 K20 K19"), IOStandard("LVCMOS33"), # Anodes
     ("led", 1, Pins("P19 L18 K18"), IOStandard("LVCMOS33")), # Cathodes via FET
    ("usb", 0,
      Subsignal("d_p", Pins("F3")),
      Subsignal("d_n", Pins("G3")),
      Subsignal("pullup", Pins("E4")),
      Subsignal("ybustest", Pins("F4")),
      IOStandard("LVCMOS33")
    ),
    ("keypad", 0,
      Subsignal("left", Pins("G2"), Misc("PULLMODE=UP")),
      Subsignal("right", Pins("F2"), Misc("PULLMODE=UP")),
    ),
```

add 32MB SDRAM for hadbadge #97

Merged enjoy-digital merged 1 commit into enjoy-digital:master from pdp7:master 21 days ago

Conversation 2  Commits 1  Checks 0  Files changed 1

pdp7 commented 22 days ago

Add AS4C32M8SA-7TCN 32MB SDRAM used on cartridge PCB by Jacob Creedon (@jcreedon) for the Hackaday Supercon 2019 badge which has an ECP5 FPGA.

These changes are from a fork by Michael Welling (@mwelling)

In addition to @mwelling, thank you to Jacob Creedon (@jcreedon), @gregdavill, Tim Ansell (@mithro), and Sean Cross (@xobs) who all helped get Linux working on this badge.

KiCad design files by @jcreedon for the SDRAM cartridge are available on GitHub.

There is also a shared project to order the SDRAM cartridge PCB.
class AS4C32M16(SDRAMModule):

    technology_timings = _TechnologyTimings(tREFI=64e6/8192, tWTR=(2, None), tCCD=(1, None), tRRD=None)

    speedgrade_timings = {"default": _SpeedgradeTimings(tRP=18, tRCD=18, tWR=12, tRFC=(None, 60), tFAW=None, tRAS=None

+ class AS4C32M8(SDRAMModule):
+     memtype = "SDR"
+     # geometry
+     nbanks = 4
+     nrows = 8192
+     ncols = 1024
+     # timings
+     technology_timings = _TechnologyTimings(tREFI=64e6/8192, tWTR=(2, None), tCCD=(1, None), tRRD=(None, 15))
+     speedgrade_timings = {"default": _SpeedgradeTimings(tRP=20, tRCD=20, tWR=15, tRFC=(None, 66), tFAW=None, tRAS=44)

# DDR

ProTip! Use [n] and [p] to navigate between commits in a pull request.
@enjoy-digital WOW! much faster! It gets to login in 28 seconds (previous version was 258 seconds).

Recording:
https://asciinema.org/a/Pcm3vd1BEdEKY9srYX6MsNfCE

Text:

```
pdp7@x1:~/.dev/enjoy/linux-on-litex-vexriscv$ lxterm --images=images.json /dev/[LXTERM] Starting....
\[BIOS CRC passed (561ab1e2)

Migen git sha1: @63188e
LiteX git sha1: -------

---------------------- SoC ----------------------
CPU: VexRiscv @ 48MHz
ROM: 32KB
SRAM: 4KB
```
Showing 1 changed file with 5 additions and 3 deletions.

```python
@ -26,12 +26,13 @@ class SoCSDRAM(SoCCore):
}
csr_map.update(SoCCore.csr_map)
- def __init__(self, platform, clk_freq, l2_size=8192, **kwargs):
+ def __init__(self, platform, clk_freq, l2_size=8192, l2_data_width=128, **kwargs):
    SoCCore.__init__(self, platform, clk_freq, **kwargs)
    if not self.integrated_main_ram_size:
        if self.cpu_type is not None and self.csr_data_width > 32:
            raise NotImplementedError("BIOS supports SDRAM initialization only for cs
- self.l2_size = l2_size
+ self.l2_size  = l2_size
+ self.l2_data_width = l2_data_width

    self._sdram_phy  = []
    self._wb_sdram_ifs = []
```
Now you can enjoy watching Linux boot while outside!! 😊

No PC tether required.
Open Source ECP5 FPGA boards

- Radiona.org ULX3S
  - 32MB SDRAM; ESP32 on board for WiFi and Bluetooth
  - Sold for $115 on CrowdSupply and Mouser
Open Source ECP5 FPGA boards

- **OrangeCrab** by Greg Davill
  - 128MB DDR RAM; Adafruit Feather form factor
  - Sold on GroupGets for $129
Want to learn FPGAs? Try Fomu!

- Online workshop from Tim Ansell and Sean Cross
- $50 on CrowdSupply
- Fits inside USB port!
- Learn how to use:
  - MicroPython
  - Verilog
  - LiteX
No hardware? Try Renode!

- **Renode** can simulate physical hardware systems including CPU, peripherals, sensors, and wired or wireless network between nodes.
source: Renode supported boards
source: Renode supported boards
Trustworthy self-hosted computer

- "A Trustworthy, Free (Libre), Linux Capable, Self-Hosting 64bit RISC-V Computer" by Gabriel L. Somlo
  - My goal is to build a Free/OpenSource computer from the ground up, so I may completely trust that the entire hardware+software system's behavior is 100% attributable to its fully available HDL (Hardware Description Language) and Software sources"

- Talk: "Toward a Trustable, Self-Hosting Computer System"
  - Video: youtube.com/watch?v=5IhujGI-K0
Bootstrapping a Trustworthy RISC-V Cleanroom System

**Host** (x86/Linux):
- Use DDC to verify we have a clean C (cross-)compiler
- Build clean HDL compiler toolchain, for both x86 and rv64
- Cross-compile target rv64 OS (kernel, libraries, utilities)
- Build rv64 SoC FPGA bitstream, from HDL sources

**Target** (rv64/Linux):
- Boot up FPGA-based rv64 computer into cross-compiled OS
  - rv64/Linux system is *self-hosting* from this point forward!
- Natively rebuild FPGA bitstream, kernel, libraries, and applications
  - we now have a trustworthy cleanroom
  - guaranteed to “honestly” compile any imported sources (HDL and/or software)!

source: “Toward a Trustable, Self-Hosting Computer System”
LiteX + Rocket 64-bit FPGA-based Linux Computer

source: “Toward a Trustable, Self-Hosting Computer System”
Linux on RISC-V with Open Hardware

Drew Fustini (@pdp7)
<drew@beagleboard.org>