

Slides: <https://github.com/pdp7/talks/blob/master/xhain-tux.pdf>

# For the Love of Tux: Linux on RISC-V

*xHain Berlin hackerspace: Make 'n Tell (2020-02-14)*



**Drew Fustini**  
[drew@oshpark.com](mailto:drew@oshpark.com)  
Twitter: [@pdp7](https://twitter.com/pdp7)



- When you write a program in the Arduino IDE, it is compiled into instructions for the microcontroller to execute.
- How does the compiler know what instructions the chip understands?
  - defined by the **Instruction Set Architecture**
  - The **ISA** is a standard, a set of rules that define the tasks the processor can perform.
  - Examples: x86 (Intel/AMD) and ARM
    - Both are proprietary and need commercial licensing



- **RISC-V: Free and Open RISC Instruction Set Arch**
  - “new instruction set architecture (ISA) that was originally designed to support computer architecture research and education and is now set to become a standard open architecture for industry”



- Keynote at Hackday Supercon 2019 by Dr. Megan Wachs of SiFive
- **“RISC-V and FPGAs: Open Source Hardware Hacking”**
  - [https://www.youtube.com/watch?v=vCG5\\_nxm2G4](https://www.youtube.com/watch?v=vCG5_nxm2G4)

- My column in the latest Hackspace Magazine is an introduction to RISC-V and how it is enabling open source chip design:
  - [hackspace.raspberrypi.org/issues/27/](https://hackspace.raspberrypi.org/issues/27/)



Drew Fustini

COLUMN

## Open-source chips

Breaking free of chip design monopolies with RISC-V



Drew Fustini

**W**hen we think about what open-source hardware means, we usually think about the board design being freely available. But what about the processor? Is there a way to make hardware that is truly open source? This month's column is dedicated to an existing – and surprisingly political – development in chip design.

When you write a program in the Arduino IDE, it is compiled into instructions for the microcontroller to execute. How does the compiler know what instructions the chip understands? This is defined by the Instruction Set Architecture. The ISA is a standard, a set of rules that define the tasks the processor can perform.

Chances are that both your laptop and the data centre streaming your favourite movie are using an ISA owned by Intel or AMD. The processor in your smartphone is almost certainly using a proprietary ISA licensed from ARM. Proprietary standards can be overpriced, prevent innovation, or even disappear altogether when companies change strategy.

Enter RISC-V, a free and open ISA created by researchers at UC Berkeley, led by Krste Asanović and David Patterson. "We were always jealous that you could get industrial-strength software that was

open," Patterson explained to VentureBeat at the RISC-V Summit back in December. "But when it came to hardware, it was proprietary. Now, with RISC-V, we get the same kind of benefit. It helps education, and it helps competition."

This open standard proved to be useful outside of academia. Nvidia and Western Digital are now shipping millions of devices with RISC-V processors. These companies have the freedom to leverage open-source implementations while avoiding ARM licensing fees – which can really add up when shipping large volumes.

The ISA is a standard, a set of rules that define the tasks the processor can perform

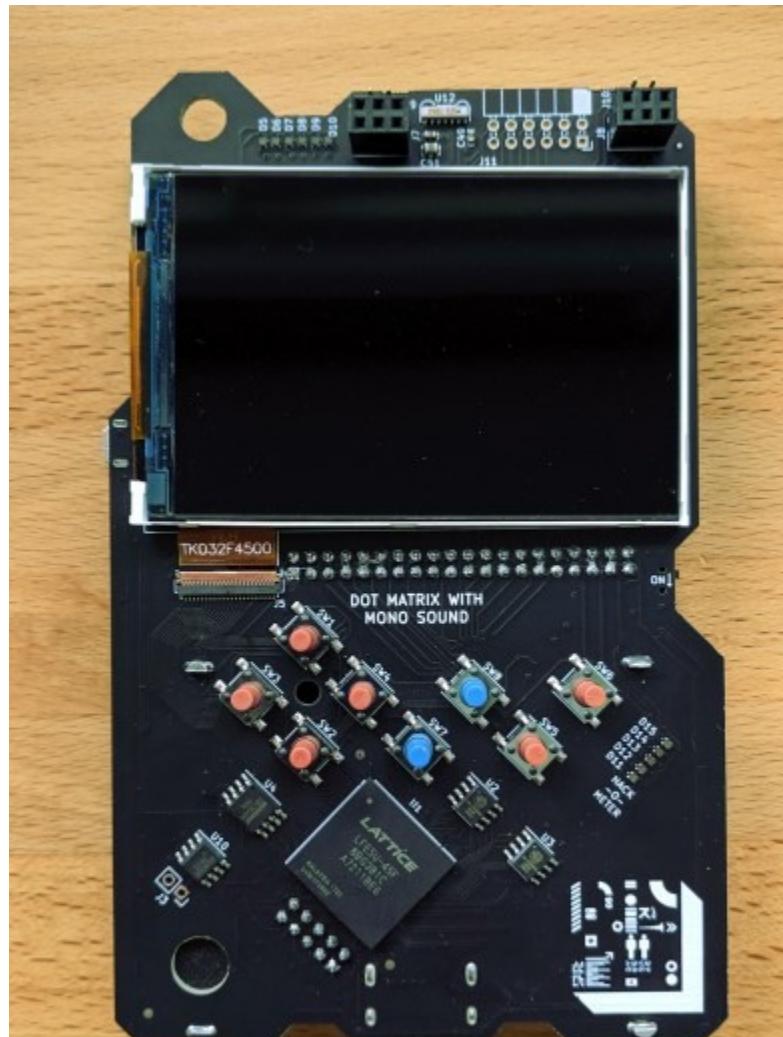
Nations such as India see the importance of being able to create processors that are not under the control of a foreign corporation, who may be forced to build in backdoors for their own government. There is also strong interest from chipmakers in China, especially now that US companies have been banned from doing business with Huawei.

With these financial and political motivations, plus an increasingly mature software ecosystem, including Linux support, it won't be long before you have a device with a RISC-V processor in your home or pocket.

You can learn more about the exciting possibilities that RISC-V unleashes from Dr. Megan Wachs by pointing your web browser to [hsmag.cc/qw-led](https://hsmag.cc/qw-led).

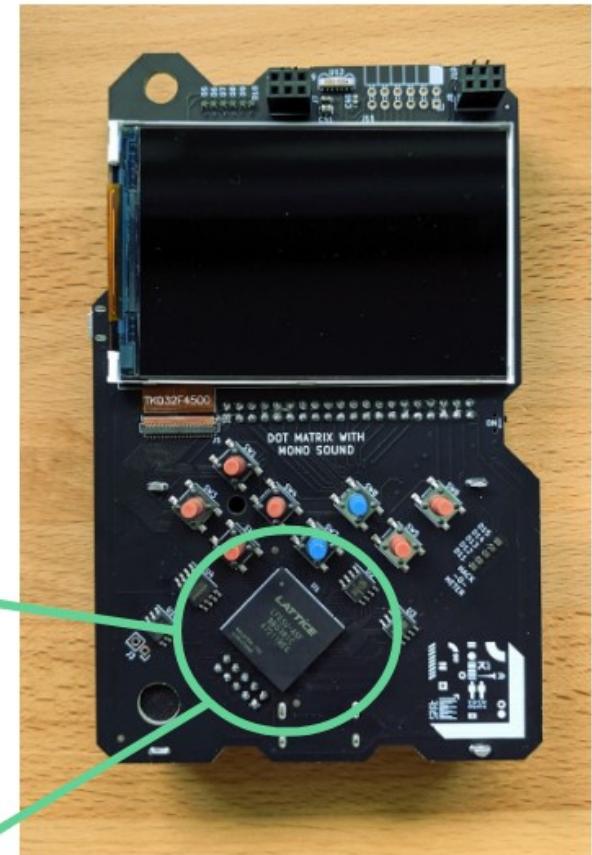
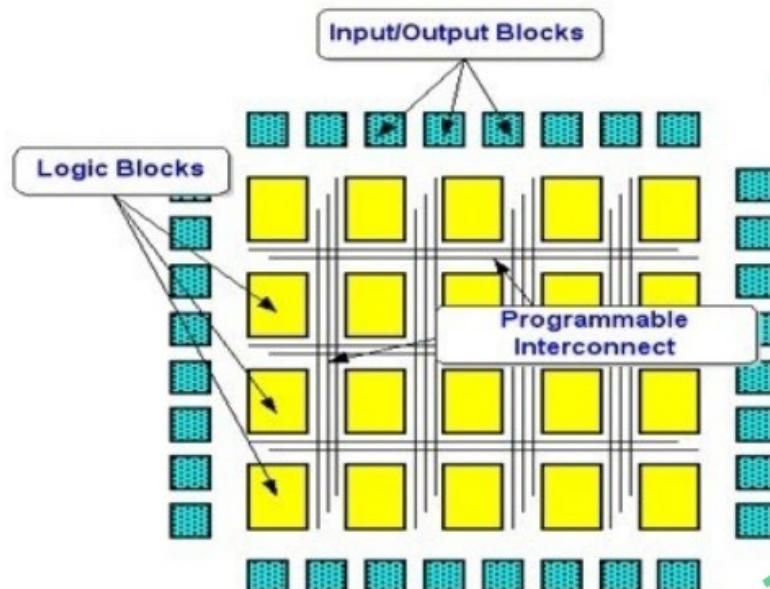
# Hackaday 2019 Supercon badge

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



# 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!



# Open Source and FPGAs

- Hackspace Magazine column about how open source FPGA tools developed by [Claire Wolf \(oe1cxw\)](#), [David Shah](#) and others have made FPGAs more accessible than ever before to makers and hackers:
  - [hackspace.raspberrypi.org/issues/26/](https://hackspace.raspberrypi.org/issues/26/)

MAKE | BUILD | HACK | CREATE **132 PAGES** OF MAKING

# HackSpace

TECHNOLOGY IN YOUR HANDS

hsmag.cc | January 2020 | Issue #26

WHAT 3D PRINTER?

Find the ultimate replicator for 2020

CIRCUIT PYTHON

SOLDERING WITH GAS

PICKING AN IMPACT DRIVER

SEWING MACHINES

Building a kiln

Melting glass with a Raspberry Pi

Drew Fustini

@pdpf

Drew Fustini is a hardware designer and embedded Linux developer. He is the Vice President of the Open Source Hardware Association, and a board member of the BeagleBoard Foundation. Drew designs circuit boards for OSH Park, a PCB manufacturing service, and maintains the Adafruit BeagleBone Python library.

FPGA

FPGAs have been the talk of the town at many of this year's hacker conferences. But what exactly is an FPGA, and why are they so hot right now?

FPGA stands for Field Programmable Gate Array, a digital logic chip that can be programmed to reconfigure the internal hardware. An FPGA does not run software – it physically changes the configuration of its gate arrays to adapt to the task at hand. Is an FPGA an incredibly versatile tool? Need 25 PWM pins for a project? No problem. Want to replicate the functionality of a vintage CPU? Your FPGA has you covered. Not only is an FPGA versatile, but it is also better at handling timing-critical tasks than a microcontroller. You can filter high-speed sensor data before it's read by your processor, or offload repetitive tasks like debouncing buttons from the burden on your microcontroller.

FPGAs are hot right now but they're not a new technology – they've been used in industry for decades

The Lattice ECP5 FPGA is capable of more advanced features than the iCE40, and it's easier to get started with too, thanks to Project Trellis led by David Shah. This enabled the ECP5-powered Supercon badge to have cool features like HDMI video, while still being open for anyone to hack on without requiring proprietary tools.

FPGAs are a fascinating technology with lots of awesome applications. If you want to find out more, start off by reading Luke Valenty's *The Hobbyists Guide to FPGAs on Hackaday.io*. ([hsmag.cc/GOAQnR](https://hsmag.cc/GOAQnR)), and watch Tim Ansell's Supercon talk to learn about the exciting future of open-source FPGA tools ([hsmag.cc/kY5IPD](https://hsmag.cc/kY5IPD)). □

The rise of the FPGA

Reconfigure your chips to suit your project

This opened the door for low-cost, open hardware boards such as mystorm Blackice, TinyFPGA, iCEBreaker, and Fomu, which are great tools for teaching workshops and building projects.

FPGAs are a fascinating technology with lots of awesome applications. If you want to find out more, start off by reading Luke Valenty's *The Hobbyists Guide to FPGAs on Hackaday.io*. ([hsmag.cc/GOAQnR](https://hsmag.cc/GOAQnR)), and watch Tim Ansell's Supercon talk to learn about the exciting future of open-source FPGA tools ([hsmag.cc/kY5IPD](https://hsmag.cc/kY5IPD)). □

- Open Source toolchains for FPGAs!
  - Project Trellis for Lattice ECP5
  - “Project Trellis and nextpnr FOSS FPGA flow for the Lattice ECP5”
    - David Shah (@fpga\_dave)
    - [youtube.com/watch?v=0se7kNes3EU](https://youtube.com/watch?v=0se7kNes3EU)

Project Trellis and nextpnr FOSS FPGA flow for the Lattice ECP5

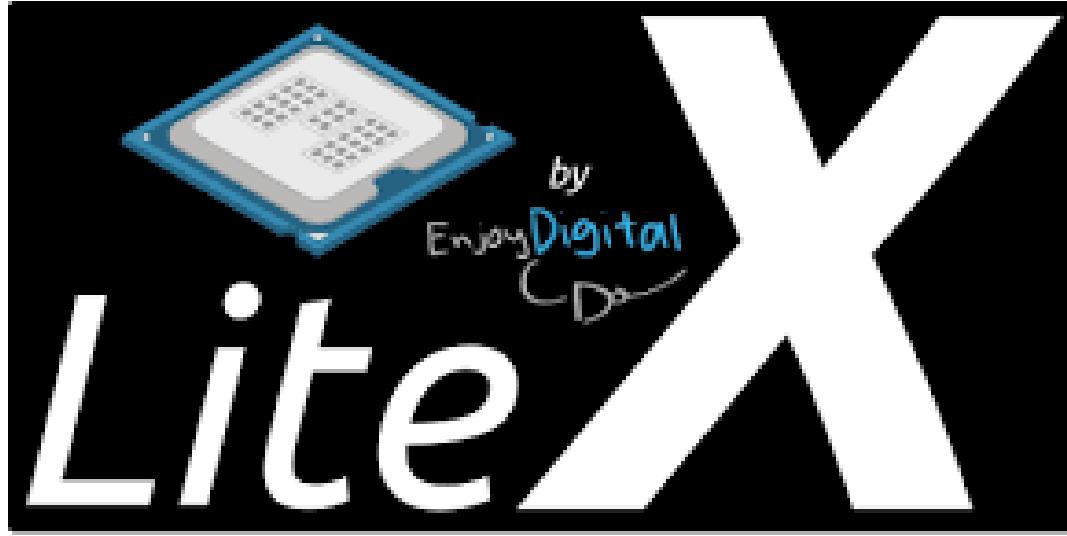
## **Project Trellis & nextpnr**

FOSS Tools for ECP5 FPGAs

David Shah  
@fpga\_dave  
Symbiotic EDA || Imperial College London

**FOSDEM** 19  
.org

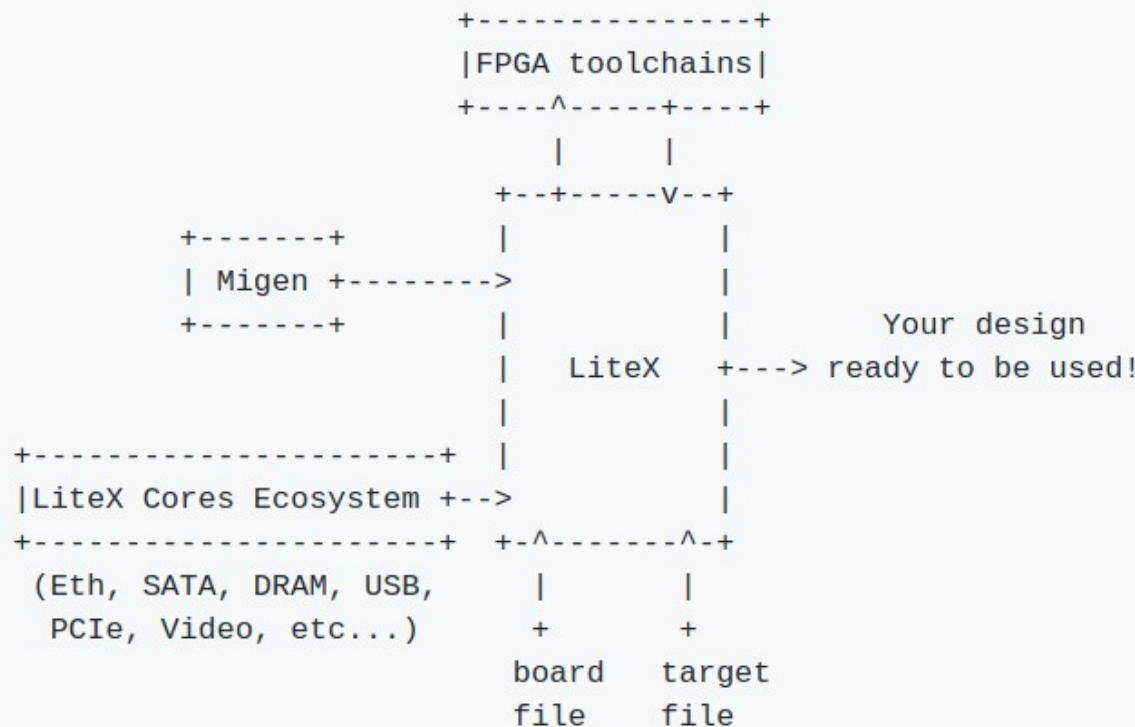




Build your hardware, easily!

- LiteX used to build cores, create SoCs and full FPGA designs.
- LiteX is based on Migen
- Migen lets you do FPGA design in Python!
- <https://github.com/enjoy-digital/litex>

# Typical LiteX design flow:



LiteX already supports various softcores CPUs: LM32, Mor1kx, PicoRV32, VexRiscv and is compatible with the LiteX's Cores Ecosystem:

Name	Build Status	Description
LiteDRAM	<span>build</span> <span>passing</span>	DRAM
LiteEth	<span>build</span> <span>passing</span>	Ethernet
LitePCIe	<span>build</span> <span>passing</span>	PCIe
LiteSATA	<span>build</span> <span>passing</span>	SATA

# Linux on LiteX-VexRiscv

- VexRiscv: 32-bit Linux Capable RISC-V CPU
  - SoC built using VexRiscv core and LiteX modules like LiteDRAM, LiteEth, LiteSDCard, ...
    - [github.com/litex-hub/linux-on-litex-vexriscv](https://github.com/litex-hub/linux-on-litex-vexriscv)

# “Team Linux on Badge”

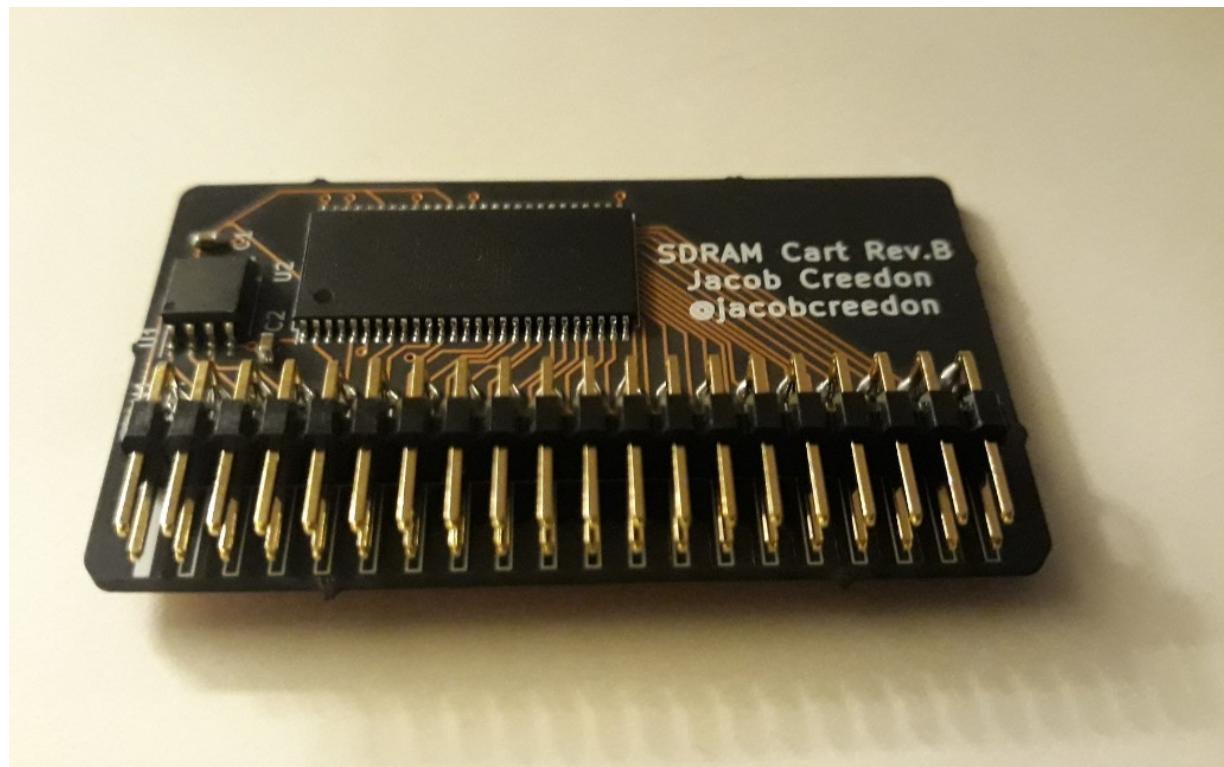


# “Team Linux on Badge”

- Blog post: Hackaday Supercon badge boots Linux using SDRAM cartridge
  - <https://blog.oshpark.com/2019/12/20/boot-linux-on-this-hackaday-supercon-badge-with-this-sdram-cartridge/>
- Michael Welling (@QwertyEmdedded), Tim Ansell (@mithro), Sean Cross (@xobs), Jacob Creedon (@jacobcreedon)
- First attempt: use the built-in 16MB SRAM... no luck :(
  - (*though xobs now might have a way to do it*)

# “Team Linux on Badge”

- Second attempt:
  - Jacob Creedon designed an a cartridge board that adds 32MB of SDRAM to the Hackaday Supercon badge... before the event!







- upstream support for Hackaday Supercon badge:
  - <https://github.com/litex-hub/litex-boards/pull/31>

[litex-hub / litex-boards](#)

Unwatch 7    Unstar 17    Fork 24

Code Issues 2 Pull requests 1 Actions Projects 0 Wiki Security Insights

## add the Hackaday Supercon ECP5 badge #31

**Merged** enjoy-digital merged 1 commit into `litex-hub:master` from `pdp7:master` 21 days ago

Conversation 18 Commits 1 Checks 1 Files changed 2 +461 -0

 pdp7 commented 22 days ago • edited

Contributor + ...

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.

**Reviewers**  
No reviews

**Assignees**  
No one assigned

**Labels**  
None yet

**Projects**  
None yet

**Milestone**

- upstream support for Hackaday Supercon badge:
  - <https://github.com/litex-hub/litex-boards/pull/31>

Merged add the Hackaday Supercon ECP5 badge #31 Changes from all commits ▾ File filter... ▾ Jump to... ▾ ⚙ Review changes ▾

215 litex\_boards/partner/platforms/hadbadge.py

```

@@ -0,0 +1,215 @@
+ from litex.build.generic_platform import *
+ from litex.build.lattice import LatticePlatform
+
+ # IOs -----
+
+ _io = [
+     ("clk8", 0, Pins("U18"), IOStandard("LVCMOS33")),
+     ("programn", 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("vbusdet", Pins("F4")),
+         IOStandard("LVCMOS33")
+     ),
+     ("keypad", 0,
+         Subsignal("left", Pins("G2"), Misc("PULLMODE=UP"))
+     )
+ ]

```

- upstream support for Hackaday Supercon badge:
  - <https://github.com/litex-hub/litex-boards/pull/31>

Merged add the Hackaday Supercon ECP5 badge #31 Changes from all commits ▾ File filter... ▾ Jump to... ▾ ⚙ ▾ Review changes ▾

246 litex\_boards/partner/targets/hadbadge.py

Viewed ...

```
@@ -0,0 +1,246 @@
+#!/usr/bin/env python3
+
+ # This file is Copyright (c) 2018-2019 Florent Kermarrec <florent@enjoy-digital.fr>
+ # This file is Copyright (c) 2018 David Shah <dave@ds0.me>
+ # License: BSD
+
+ import argparse
+ import sys
+
+ from migen import *
+ from migen.genlib.resetsync import AsyncResetSynchronizer
+
+ from litex_boards.platforms import hadbadge
+
+ from litex.soc.cores.clock import *
+ from litex.soc.integration.soc_sdram import *
+ #from litex.soc.integration.soc_core import *
+ from litex.soc.integration.builder import *
+
+ #from .spi_ram_dual import SpiRamDualQuad
+
+ from litedram import modules as litedram_modules
+ from litedram.phy import GENSDRPHY
```

[Code](#)[Issues 17](#)[Pull requests 2](#)[Actions](#)[Security](#)[Insights](#)

# add the Hackaday Supercon ECP5 badge #68

**Merged**enjoy-digital merged 1 commit into [litex-hub:master](#) from [pdp7:master](#) 21 days ago[Conversation 2](#)[Commits 1](#)[Checks 0](#)[Files changed 1](#)

pdp7 commented 22 days ago

Contributor

+ ...

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

Merged

## add the Hackaday Supercon ECP5 badge #68

Changes from all commits ▾ File filter... ▾ Jump to... ▾ ⚙

▼ 13 [make.py] ↗

```
@@ -160,6 +160,16 @@ def __init__(self):  
160      160          def load(self):  
161      161              os.system("ujprog build/ulx3s/gateware/top.svf")  
162      162  
163 + # HADBadge support -----  
164 +  
165 + class HADBadge(Board):  
166 +     def __init__(self):  
167 +         from litex_boards.targets import hadbadge  
168 +         Board.__init__(self, hadbadge.BaseSoC, {"serial"})  
169 +  
170 +     def load(self):  
171 +         os.system("dfu-util --alt 2 --download build/hadbadge/gateware/top.bit --reset")  
172 +  
163      173      # OrangeCrab support -----  
164      174  
165      175      class OrangeCrab(Board):  
@@ -209,6 +219,7 @@ def load(self):  
209      219          # Lattice
```

[Code](#)[Issues 21](#)[Pull requests 3](#)[Actions](#)[Projects 0](#)[Wiki](#)[Security](#)[In](#)

# 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

Contributor



...

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.

Refer to [my blog post](#) for more information.

Merged

## add 32MB SDRAM for hadbadge #97

Changes from all commits ▾

File filter... ▾

Jump to... ▾



9 litedram/modules.py



```
@@ -190,6 +190,15 @@ class AS4C32M16(SDRAMModule):
190      technology_timings = _TechnologyTimings(tREFI=64e6/8192, tWTR=(2, None), tCCD=(1,
191      speedgrade_timings = {"default": _SpeedgradeTimings(tRP=18, tRCD=18, tWR=12, tRFC=
192
193 +     class AS4C32M8(SDRAMModule):
194 +         memtype = "SDR"
195 +         # geometry
196 +         nbanks = 4
197 +         nrows  = 8192
198 +         ncols   = 1024
199 +         # timings
200 +         technology_timings = _TechnologyTimings(tREFI=64e6/8192, tWTR=(2, None), tCCD=(1,
201 +         speedgrade_timings = {"default": _SpeedgradeTimings(tRP=20, tRCD=20, tWR=15, tRFC=
193
194     # DDR -----
195
```



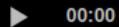
- Opened GitHub issue:
  - optimize performance on Hackaday Badge #35
    - <https://github.com/litex-hub/litex-boards/issues/35>
- Now 10x faster!
  - <https://asciinema.org/a/Pcm3vd1BEdEKY9srYX6MsNfCE>
  - Thanks to [enjoy-digital](#)



Asciicast updated.

```
[ 7.896093] Initramfs unpacking failed: junk in compressed archive
[ 7.953813] workingset: timestamp_bits=30 max_order=13 bucket_order=0
[ 9.236463] Block layer SCSI generic (bsg) driver version 0.4 loaded (major 253)
[ 9.239004] io scheduler mq-deadline registered
[ 9.240102] io scheduler kyber registered
[ 13.977920] f0001000.serial: MMIO 0xf0001000 (irq = 0, base_baud = 0) is a liteuart
[ 13.980290] printk: console [liteuart0] enabled
[ 13.980290] printk: console [liteuart0] enabled
[ 13.982965] printk: bootconsole [sbi0] disabled
[ 13.982965] printk: bootconsole [sbi0] disabled
[ 14.058778] libphy: Fixed MDIO Bus: probed
[ 14.074959] i2c /dev entries driver
[ 14.247461] NET: Registered protocol family 10
[ 14.307974] Segment Routing with IPv6
[ 14.315214] sit: IPv6, IPv4 and MPLS over IPv4 tunneling driver
[ 14.455698] Freeing unused kernel memory: 140K
[ 14.457905] This architecture does not have kernel memory protection.
[ 14.459170] Run /init as init process
mount: mounting tmpfs on /dev/shm failed: Invalid argument
mount: mounting tmpfs on /tmp failed: Invalid argument
mount: mounting tmpfs on /run failed: Invalid argument
Starting syslogd: OK
Starting klogd: OK
Running sysctl: OK
Initializing random number generator... [ 23.063050] random: dd: uninitialized urandom read (512 bytes read)
done.
Starting network: OK
Starting dropbear sshd: [ 27.336210] random: dropbear: uninitialized urandom read (32 bytes read)
OK
```

Welcome to Buildroot



00:00



Linux boots on Hackaday Supercon FPGA badge [10x faster!]

[Code](#)[Issues 2](#)[Pull requests 1](#)[Actions](#)[Projects 0](#)[Wiki](#)[Security](#)[Insights](#)

# optimize performance on Hackaday Badge #35

[Edit](#)[New issue](#)[Open](#)

pdp7 opened this issue 17 days ago · 7 comments



pdp7 commented 17 days ago

Contributor



...

**Assignees**

No one assigned

**Labels**

None yet

**Projects**

None yet

**Milestone**

No milestone

**Notifications**

Customize

[Unsubscribe](#)

You're receiving notifications because  
you're watching this repository.

[Related to comments in PR #31 \(comment\)](#)

The performance of the ECP5 Hackaday Badge with 32MB SDRAM is "painfully" slow.

@mithro suggested there could be some issue with the configuration.

@enjoy-digital has attempted some optimizations:

[#31 \(comment\)](#)

With [enjoy-digital/litedram@ 34e6c24](#) and [enjoy-digital/litex@ fa22d6a](#) we have a ~10% boot time speedup on designs using SDRAM:

- De0Nano: 94.6.s to 84.6s
- ULX3S: 75.9s to 68.2s

On Arty with DDR3 the gain is effect more limited: 8.7s to 8.4s. That would be interesting to test this on the badge.

I will measure if the boot time improve

Open

## optimize performance on Hackaday Badge #35

pdp7 opened this issue 17 days ago · 7 comments



pdp7 commented 15 days ago • edited

Contributor

Author

+ 😊 ...

@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....  
lBIOS CRC passed (561ab1e2)
```

```
Migen git sha1: 063188e  
LiteX git sha1: -----
```

```
-===== SoC =====  
CPU: VexRiscv @ 48MHz  
ROM: 32KB  
SRAM: 4KB  
Lo:
```



**enjoy-digital** committed 15 days ago

1 parent 2317519

commit 39ce39a298f5



Showing **1 changed file** with **5 additions** and **3 deletions**.

8 litex/soc/integration/soc\_sdram.py

@@ -26,12 +26,13 @@ class SoCSDRAM(SoCCore):

26 26 }

```
27      27      csr_map.update(SoCCore.csr_map)
```

28

```
29     -     def __init__(self, platform, clk_freq, l2_size=8192, **kwargs):
```

```
29 +     def __init__(self, platform, clk_freq, l2_size=8192, l2_data_width=128, **kwargs):
```

```
30      30          SoCCore.__init__(self, platform, clk_freq, **kwargs)
```

```
31      31          if not self.integrated_main_ram_size:
```

```
32      32          if self.cpu_type is not None and self.csr_data_width > 32:
```

```
raise NotImplementedException("BIOS supports SDRAM initialization only for c
```

34 - self.l2\_size = l2\_size

34 + self.l2\_size = l2\_size

```
35 +         self.l2_data_width = l2_data_width
```

35 36

```
36      37          self._sram_phy = []
```

```
37      38          self.wb_sdram_ifs = []
```

# Linux on Open Source Hardware with Open Source chip design

by Drew Fustini



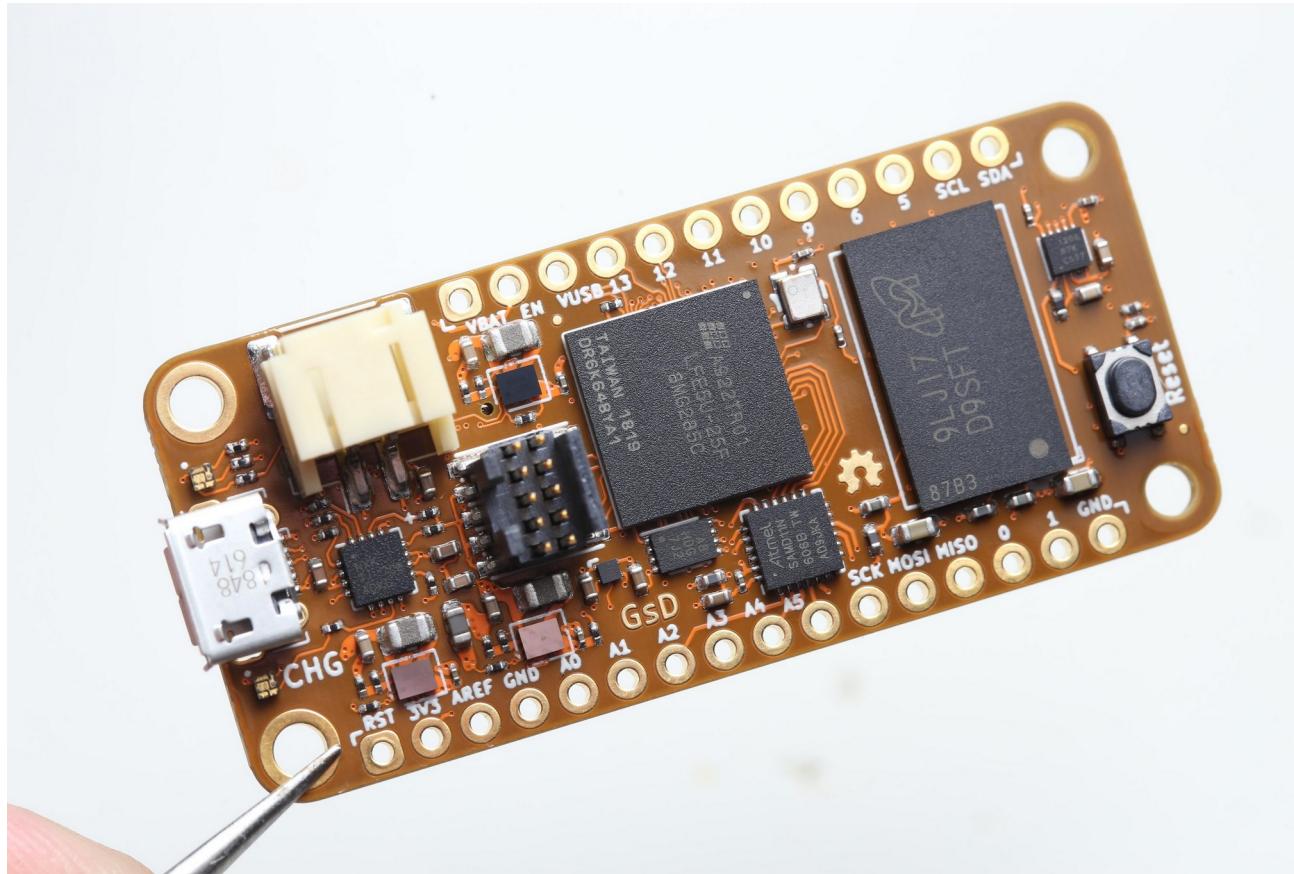
## CERN Open Hardware Licence

- 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



# Open Source ECP5 boards

- Lattice ECP5 FPGA in Adafruit Feather form factor and 128MB DDR RAM:
  - Orange Crab by Greg Davill
    - <https://github.com/gregdavill/OrangeCrab>





Greg @ #36c3  
@GregDavill



Replying to @mithro @pdp7 and 2 others

Done. 😊

```
File Edit View Terminal Tabs Help
SRAM:      4KB
L2:        8KB
MAIN-RAM: 131072KB

----- Initialization -----
Initializing SDRAM...
SDRAM now under software control
Read leveling:
m0, b0: |11100000| delays: 01+-01
best: m0, b0 delays: 01+-01
m1, b0: |11100000| delays: 01+-01
best: m1, b0 delays: 01+-01
SDRAM now under hardware control
Memtest OK

----- Boot -----
Booting from serial...
Press Q or ESC to abort boot completely.
sL5DdSMmkekro
[LXTERM] Received firmware download request from the device.
[LXTERM] Uploading buildroot/Image to 0xc0000000 (4545524 bytes)...
[LXTERM] Upload complete (85.6KB/s).
[LXTERM] Uploading buildroot/rootfs.cpio to 0xc0800000 (8029184 bytes)...
[ 0:43 ] 1.2K views => | 98%
```



# Open Source ECP5 boards

- Radiona.org ULX3S
    - <https://www.crowdsupply.com/radiona/ulx3s>
- 

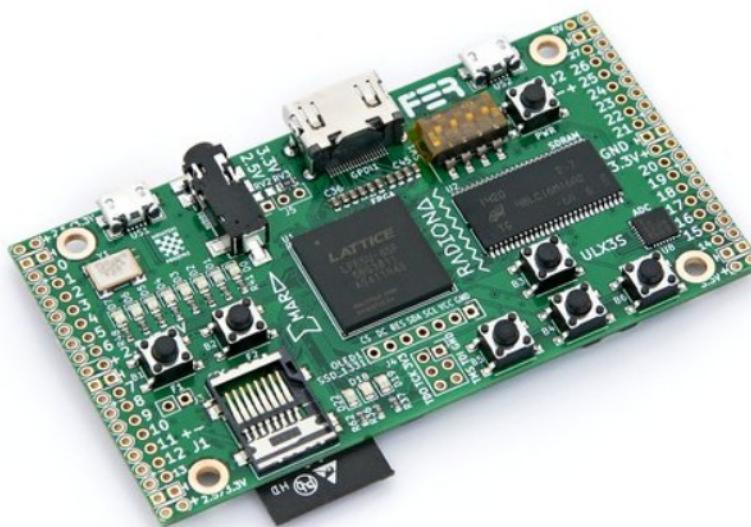
## ULX3S

A powerful, open hardware ECP5 FPGA dev board

This project is coming soon. Sign up to receive updates and be notified when this project launches.

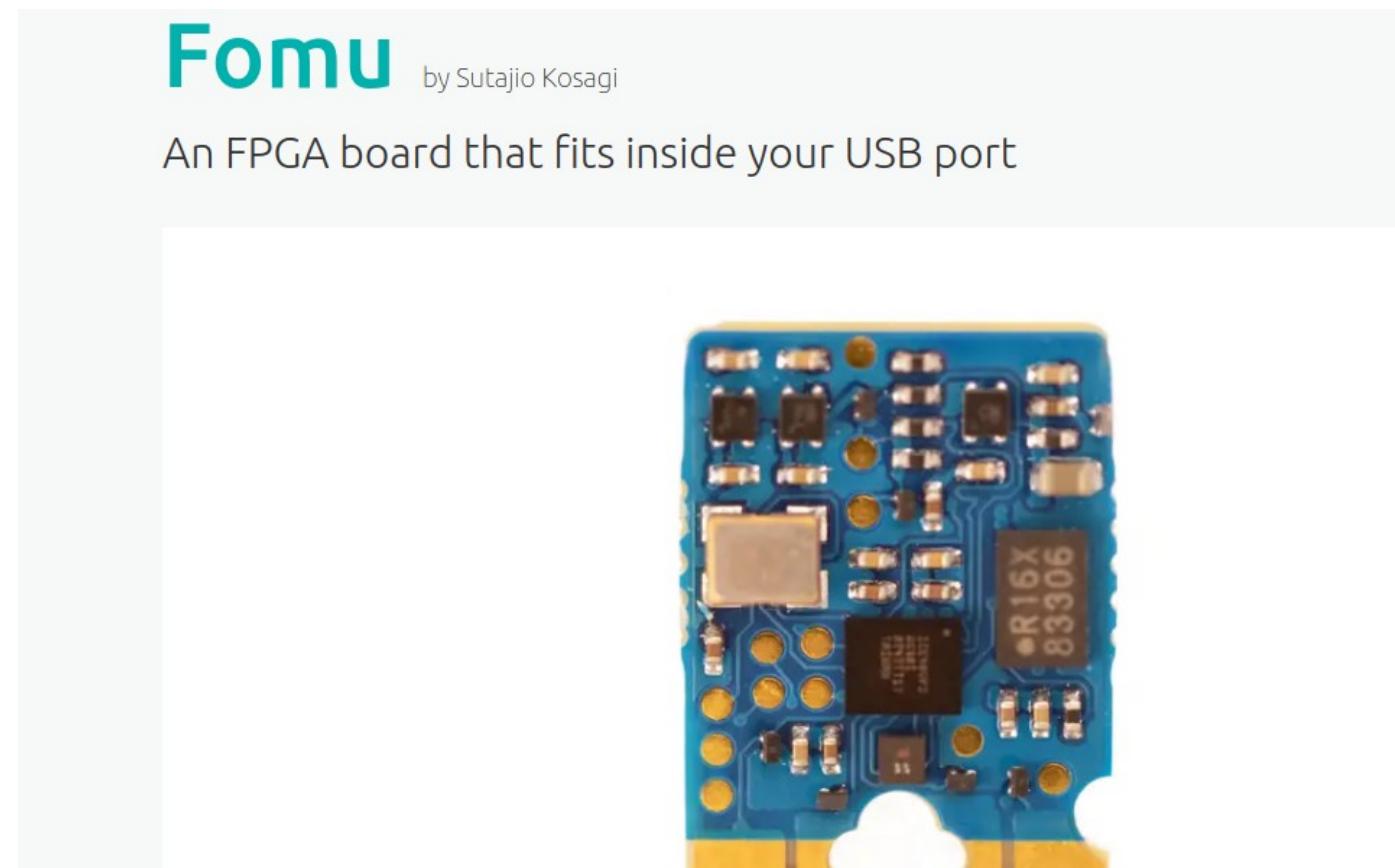
me@example.com

Subscribe



# Fomu – get way to learn FPGAs!

- [workshop.fomu.im](http://workshop.fomu.im)
- [crowdsupply.com/sutajio-kosagi/fomu](https://crowdsupply.com/sutajio-kosagi/fomu)
- Fits in USB port
- Learn:
  - MicroPython
  - Verilog
  - LiteX



# Designing Hardware in Python?

- Yes!
- “Using Python for creating hardware to record FOSS conferences!”
- Tim “mithro” Ansell
- [youtube.com/watch?v=MkVX\\_mh5dOU](https://youtube.com/watch?v=MkVX_mh5dOU)

# Why is this powerful?

- Python is a **powerful** language
- Python is a **productive** language
- Can generate exact Verilog as needed

hdmi2usb.tv

@mithro

j.mp/pyhw-lca2017



Using Python for creating hardware to record FOSS conferences!

1,041 views • Jan 19, 2017

16 0 SHARE ...

Up next



Visual Basic .Net : Search in Access Database ...  
iBasskung

AUTOPLAY

# Python HDL?

You can think of Migen as  
*"an easy way  
to generate Verilog"*

# Blink Led - Migen

```
class Blinker(Module):
    def __init__(self, led, cycles=15000000):
        counter = Signal(max=cycles)
        self.sync += counter.eq(counter - 1)
        self.comb += led.eq(counter[-1])
```

Most significant bit  
connected to LED signals