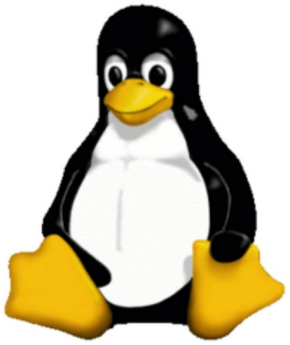
Slides: https://github.com/pdp7/talks/blob/master/xhain-tux.pdf

Linux on RISC-V

FOSS North 2020: Virtual Lightning Talk



Drew Fustini
drew@oshpark.com
Twitter: @pdp7

CC BY SA

- 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

- 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/


Drew Fustini

COLUMN SPARK

Open-source chips

Breaking free of chip design monopolies with RISC-V

Drew Fustini

@pdp7

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.org Foundation. Drew designs circuit boards for OSH Park, a PCB manufacturing service, and maintains the Adafruit BeagleBone Python library.

When 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 exciting – 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.

> **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.

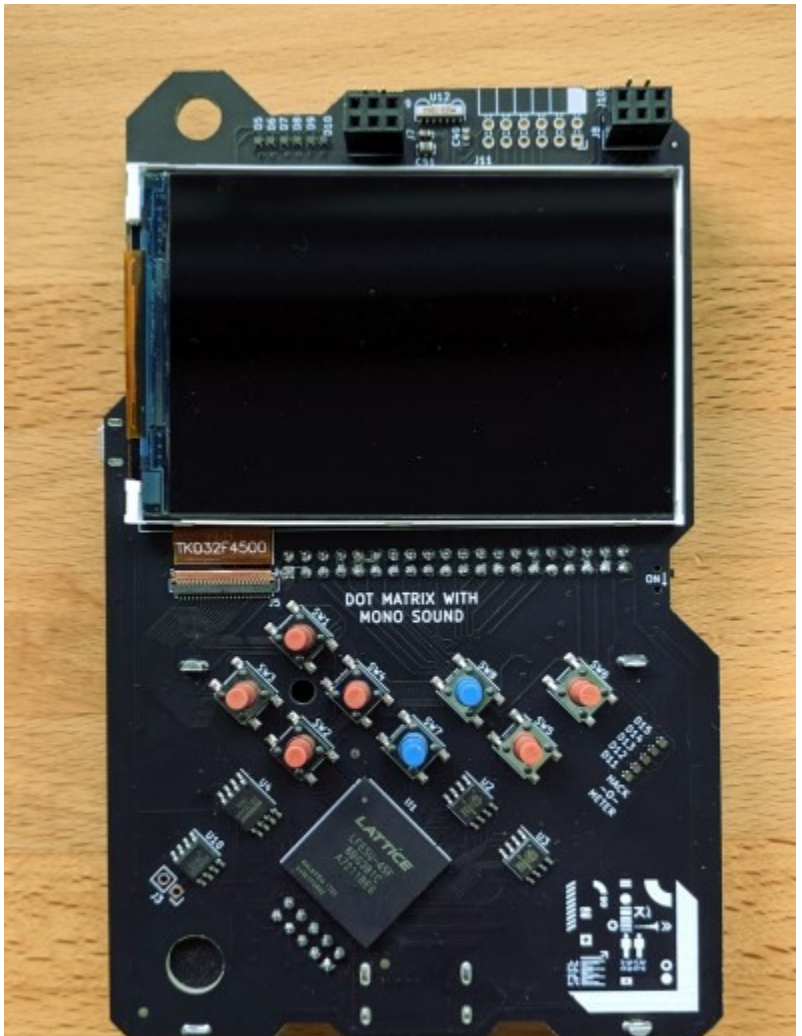
Nations such as India see the importance of being able to create processors that are not under the control of a foreign corporation, who may be forced to build in backdoors for their own government. There is also strong interest from chipmakers in China, especially now that US companies have been banned 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/qw4ed1**.

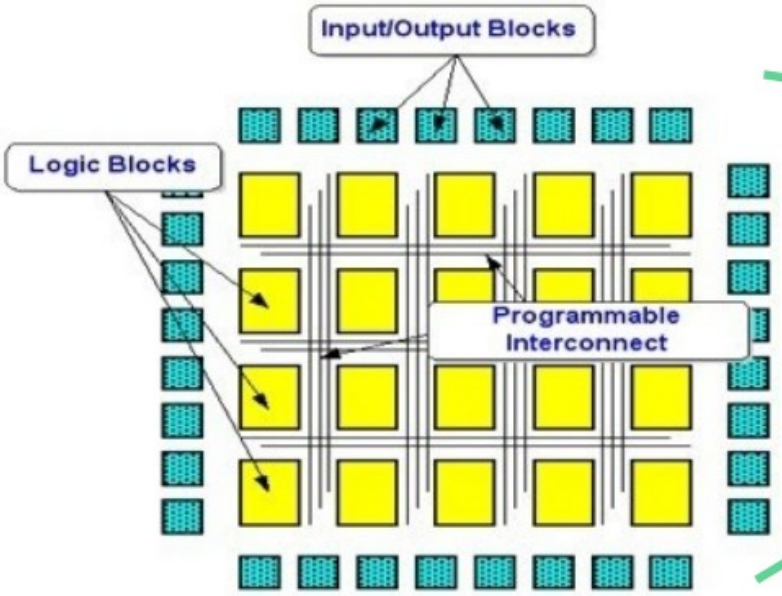
Hackaday 2019 Supercon badge

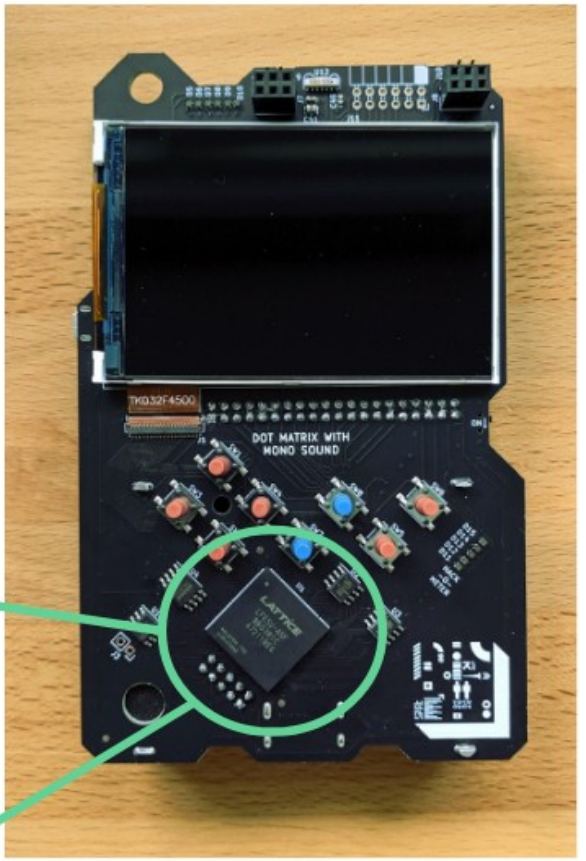
- RISC-V “soft” core on ECP5 FPGA
- Gigantic FPGA In A Game Boy Form Factor



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 about 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/

The rise of the FPGA

Reconfigure your chips to suit your project

Drew Fustini

@pdp7

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.org Foundation. Drew designs circuit boards for OSH Park, a PCB manufacturing service, and maintains the Adafruit BeagleBone Python library.

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, making 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, reducing 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. However, pricey FPGA development boards, and the massive proprietary software suites needed to develop FPGA designs, means that there has historically been limited adoption in the maker community.

When you write an Arduino program, you're using a language called Wiring, a variation of C++. When you press compile, this high-level code is used to generate a binary file that the processor on the Arduino board executes to achieve the purpose of your program. Similarly, we describe the circuits we want in an FPGA using a high-level, text-based format known as a hardware description language (HDL), such as Verilog. The HDL design is then transformed by a synthesis tool into the basic building blocks that exist in the FPGA.

This process is normally done with proprietary software from the FPGA vendor, but these tools can take ages to download and devour disk space. That is, until Project IceStorm, led by Clifford Wolf, enabled a complete open-source workflow for the Lattice iCE40 FPGA.

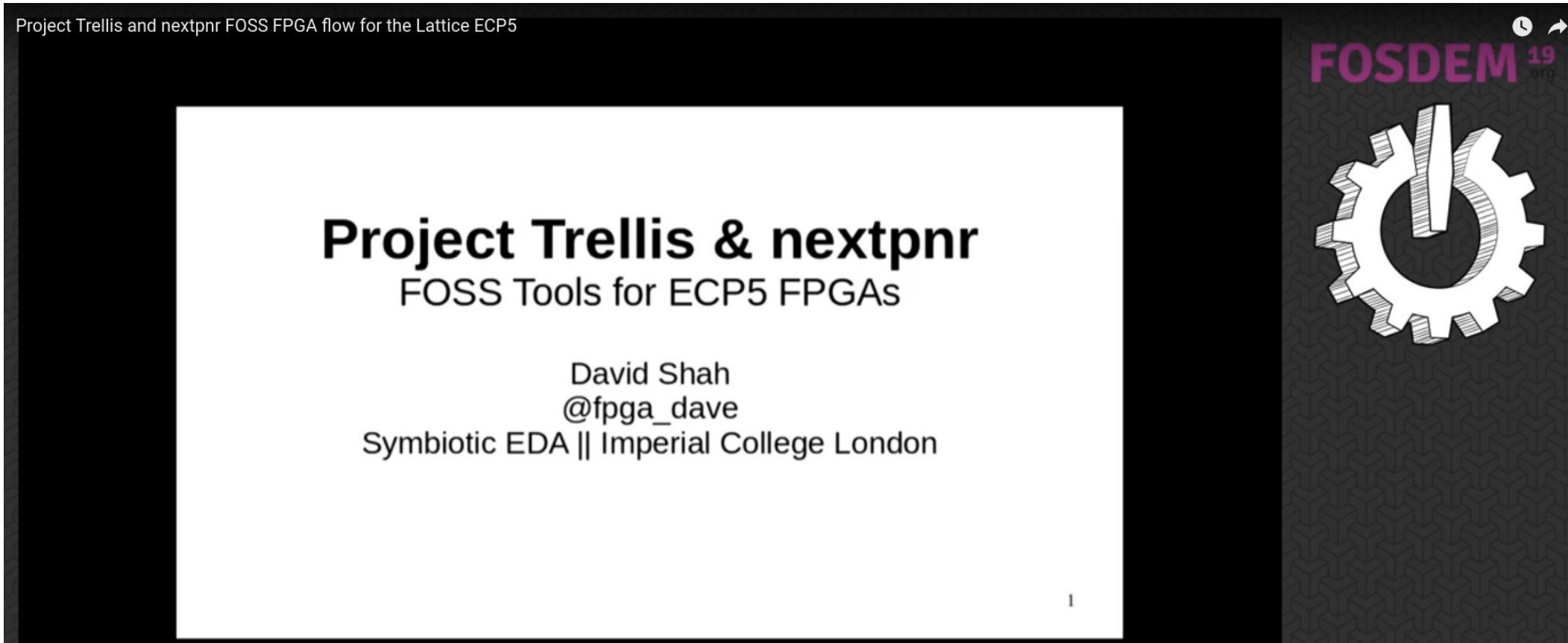
FPGAs are hot right now but they're not a new technology – they've been used in industry for decades

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.

The Lattice ECP5 FPGA is capable of more advanced features than the iCE40, and it now has an open-source workflow 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.

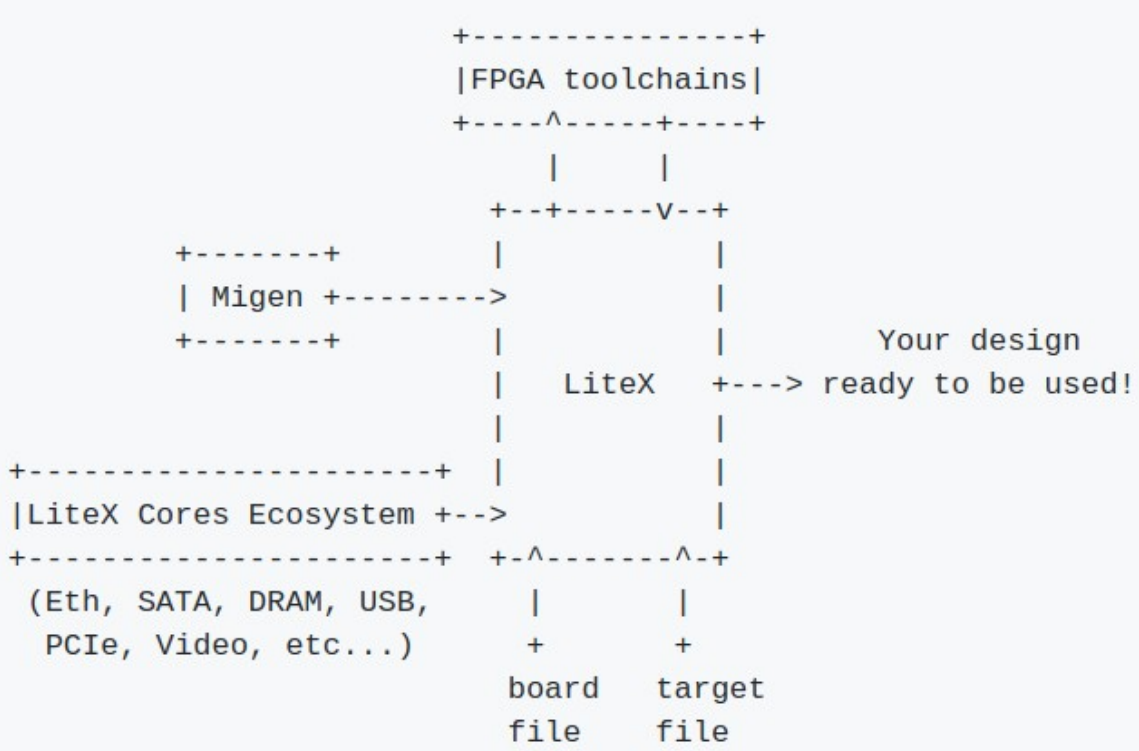
FPGAs are a fascinating technology with lots of awesome applications. If you want to find out more, start off by reading Luke Valenty's *The Hobbyists Guide to FPGAs* on **Hackaday.io** (**hsmag.ce/G0AQwR**), and watch Tim Ansell's Supercon talk to learn about the exciting future of open-source FPGA tools (**hsmag.cc/kyStPD**).

- 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



- 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:

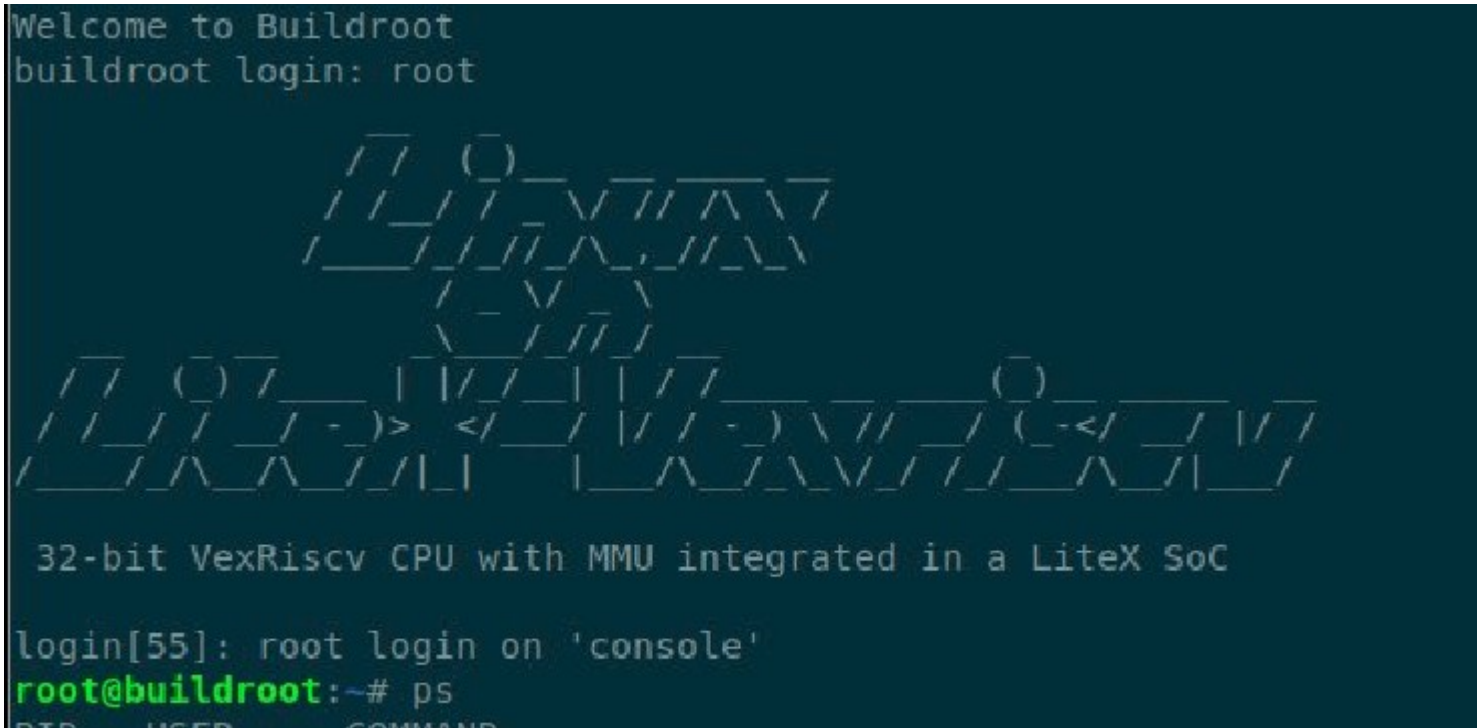


LiteX already supports various softcores CPUs: LM32, Mor1kx, PicoRV32, VexRiscv and is compatible with the LiteX's Cores Ecosystem:

Name	Build Status	Description
LiteDRAM	build passing	DRAM
LiteEth	build passing	Ethernet
LitePCIe	build passing	PCIe
LiteSATA	build passing	SATA

Linux on LiteX-VexRiscv

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



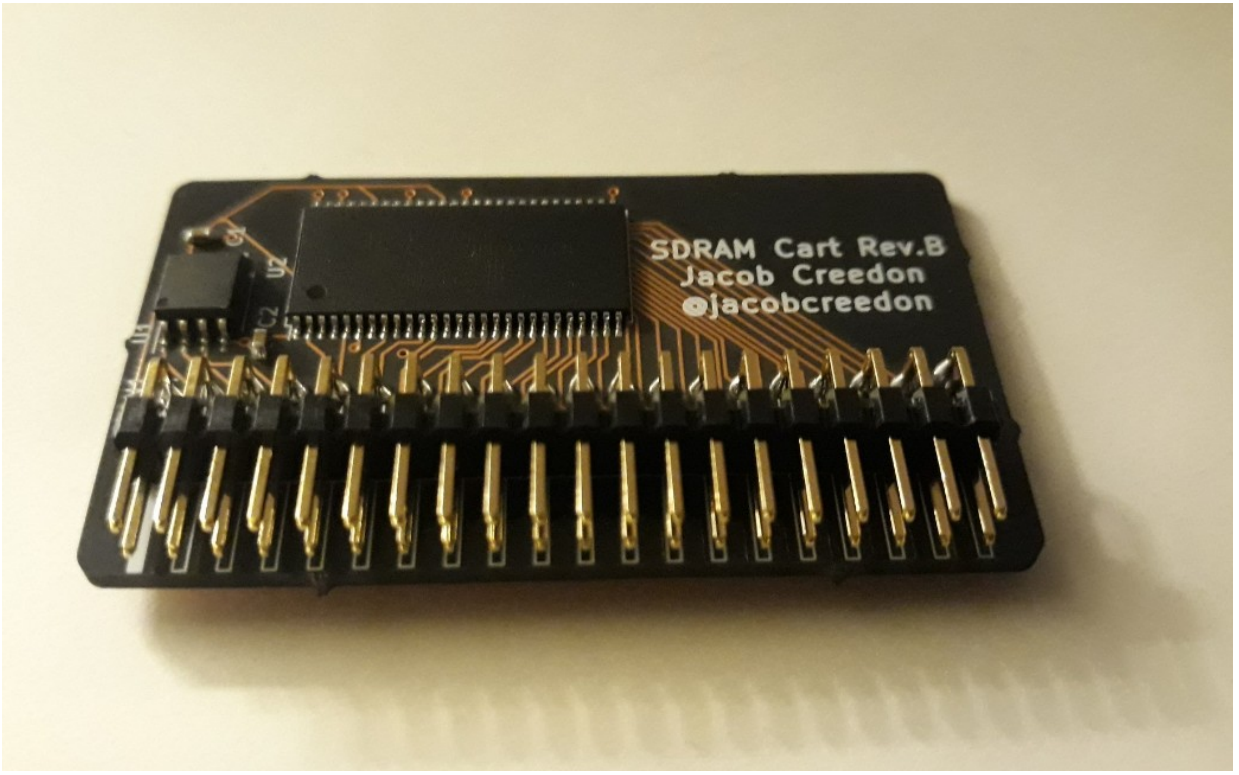
“Team Linux on Badge”

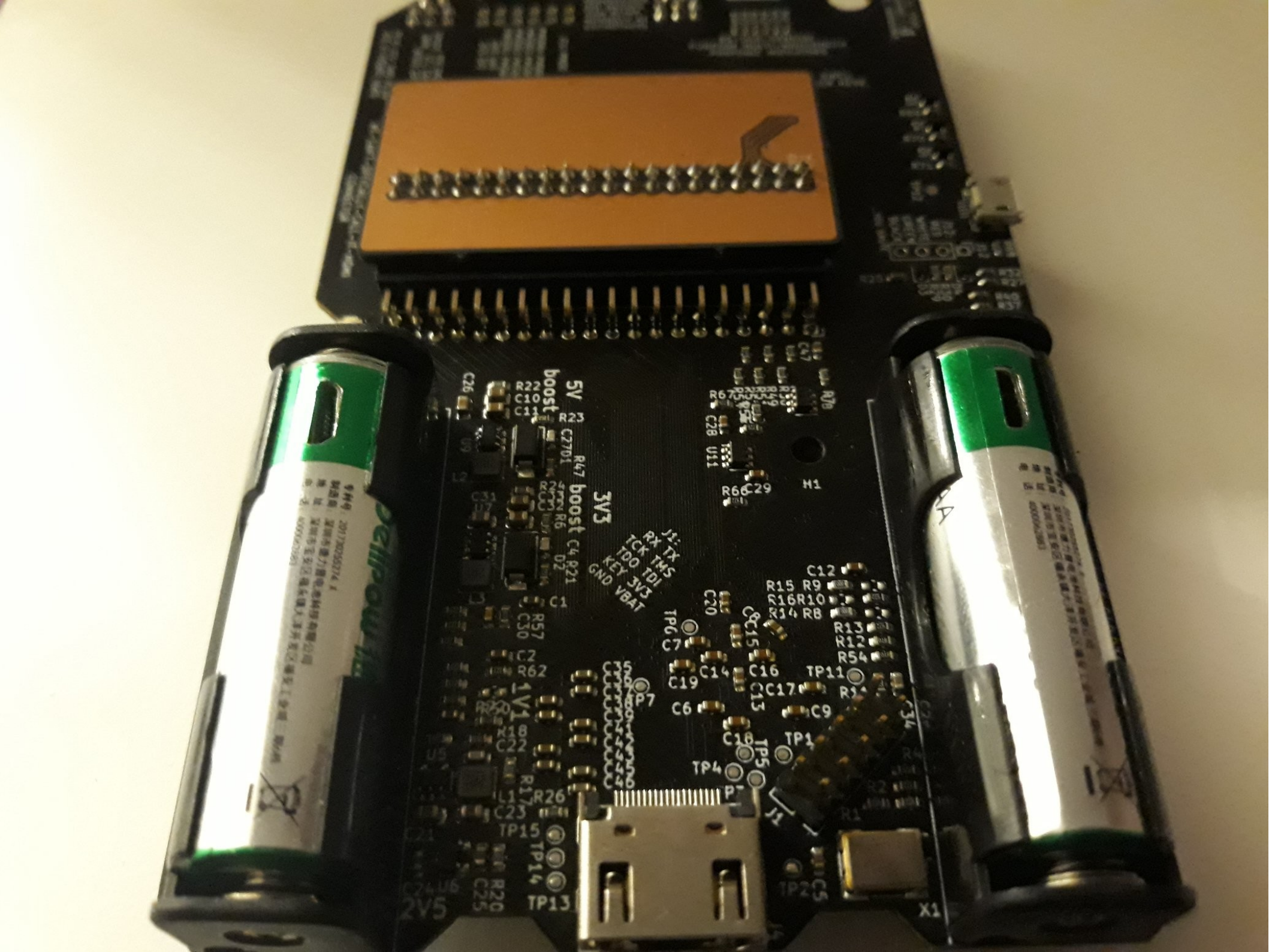
“Team Linux on Badge”

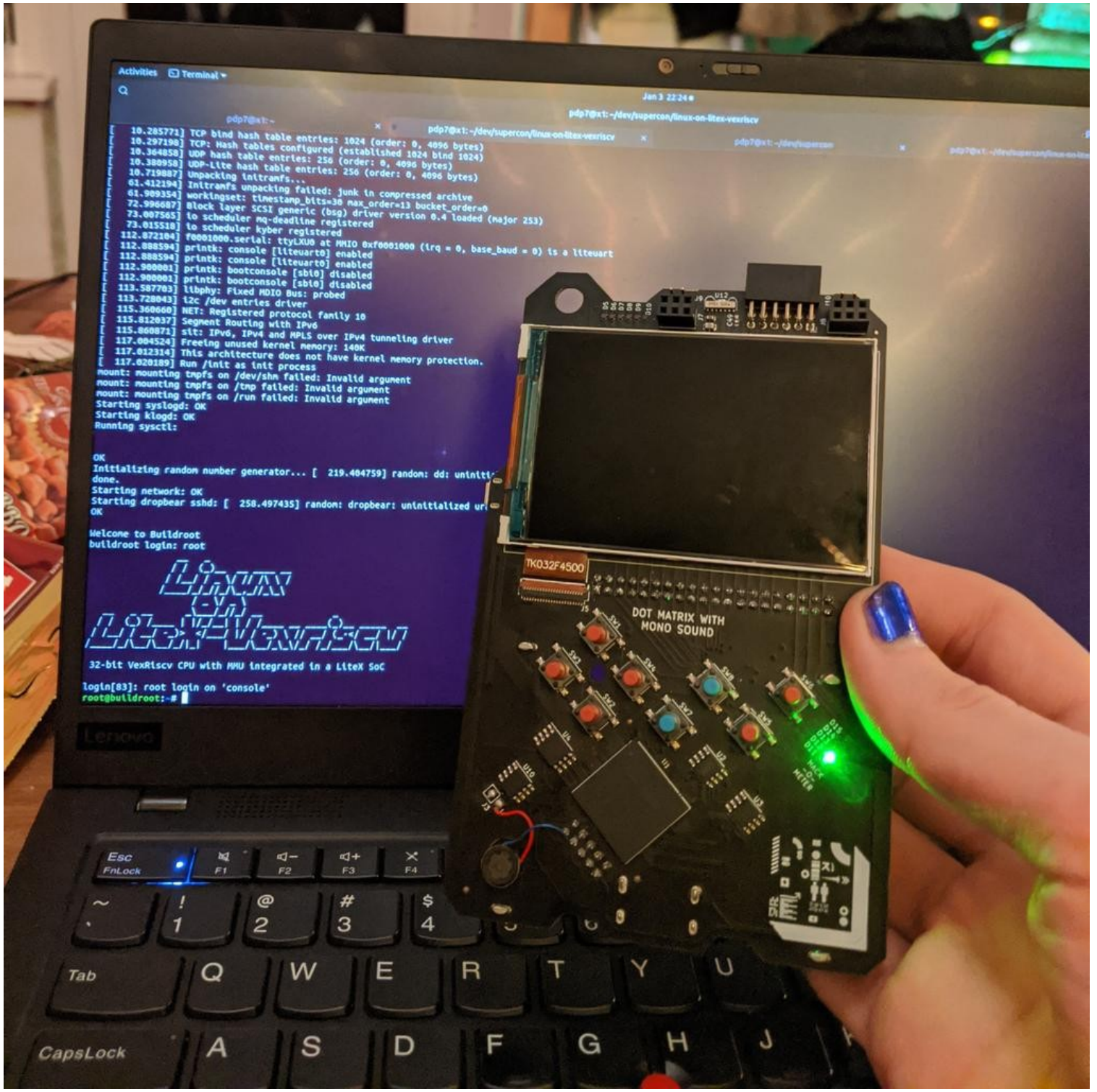
- Blog post: Hackaday Supercon badge boots Linux using SDRAM cartridge
  - https://blog.oshpark.com/2019/12/20/boot-linux-on-this-hackaday-supercon-badge-with-this-sdram-cartridge/
- Michael Welling (@QwertyEmedded), Tim Ansell (@mithro), Sean Cross (@xobs), Jacob Creedon (@jacobcreedon)
- First attempt: 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!




5V boost
3V3 boost
1V1
2V5
J1: TX RX TMS TCK TDI TDO KEY 3V3 GND VBAT
H1
X1
J1
TP1
TP4
TP5
TP11
TP13
TP14
TP15
TP25
R22
C10
C11
R23
C27
R47
R24
R6
C4
R21
C1
R57
C30
C2
R62
R18
C22
R17
R26
C23
L13
C21
C24
U6
R20
C25
C26
C31
U7
R67
R7
C28
U11
R66
C29
C12
R15
R9
R16
R10
R14
R8
R13
R12
R54
C20
C8
C15
C7
C14
C16
C19
C17
C13
C6
C9
C18
C34
R1
R2
C5


Activities
Terminal
Jan 3 22:24
pdp7@x1: ~/dev/supercon/linux-on-litex-vexriscv
[10.285771] TCP bind hash table entries: 1024 (order: 0, 4096 bytes)
[10.297198] TCP: Hash tables configured (established 1024 bind 1024)
[10.364858] UDP hash table entries: 256 (order: 0, 4096 bytes)
[10.380958] UDP-Lite hash table entries: 256 (order: 0, 4096 bytes)
[10.719887] Unpacking initramfs...
[61.412194] Initramfs unpacking failed: junk in compressed archive
[61.909354] workingset: timestamp_bits=30 max_order=13 bucket_order=0
[72.996687] Block layer SCSI generic (bsg) driver version 0.4 loaded (major 253)
[73.007565] io scheduler mq-deadline registered
[73.015518] io scheduler kyber registered
[112.872104] f0001000.serial: ttyLXU0 at MMIO 0xf0001000 (irq = 0, base_baud = 0) is a liteuart
[112.888594] printk: console [liteuart0] enabled
[112.888594] printk: console [liteuart0] enabled
[112.900001] printk: bootconsole [sbi0] disabled
[112.900001] printk: bootconsole [sbi0] disabled
[113.587703] libphy: Fixed MDIO Bus: probed
[113.728043] i2c /dev entries driver
[115.360660] NET: Registered protocol family 10
[115.812037] Segment Routing with IPv6
[115.860871] sit: IPv6, IPv4 and MPLS over IPv4 tunneling driver
[117.004524] Freeing unused kernel memory: 140K
[117.012314] This architecture does not have kernel memory protection.
[117.020189] 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... [219.404759] random: dd: uninitialized
done.
Starting network: OK
Starting dropbear sshd: [258.497435] random: dropbear: uninitialized
OK
Welcome to Buildroot
buildroot login: root
Linux on LiteX-VexRiscv
32-bit VexRiscv CPU with MMU integrated in a LiteX SoC
login[83]: root login on 'console'
root@buildroot:~#
TK032F4500
DOT MATRIX WITH
MONO SOUND
Lenovo
Esc
FnLock
F1
F2
F3
F4
Tab
CapsLock

- 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

Edit

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 cartiridge 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
```

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 cartiridge 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
```

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):
190 190         technology_timings = _TechnologyTimings(tREFI=64e6/8192, tWTR=(2, None), tCCD=(1,
191 191         speedgrade_timings = {"default": _SpeedgradeTimings(tRP=18, tRCD=18, tWR=12, tRFC=
192 192
    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=
193 202
194 203   # DDR ----------------------------------------------------------------------------
195 204
```

ProTip! Use [n] and [p] to navigate between commits in a pull request.

- 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

asciinema.org/a/293450

Watch Record Docs Blog About

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: ttyLXU0 at 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!]

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

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

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.

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
```

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):
        }
        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 c
-           self.l2_size = l2_size
+           self.l2_size       = l2_size
+           self.l2_data_width = l2_data_width

            self._sdram_phy    = []
            self._wb_sdram_ifs = []
```

Linux on Open Source Hardware with Open Source chip design

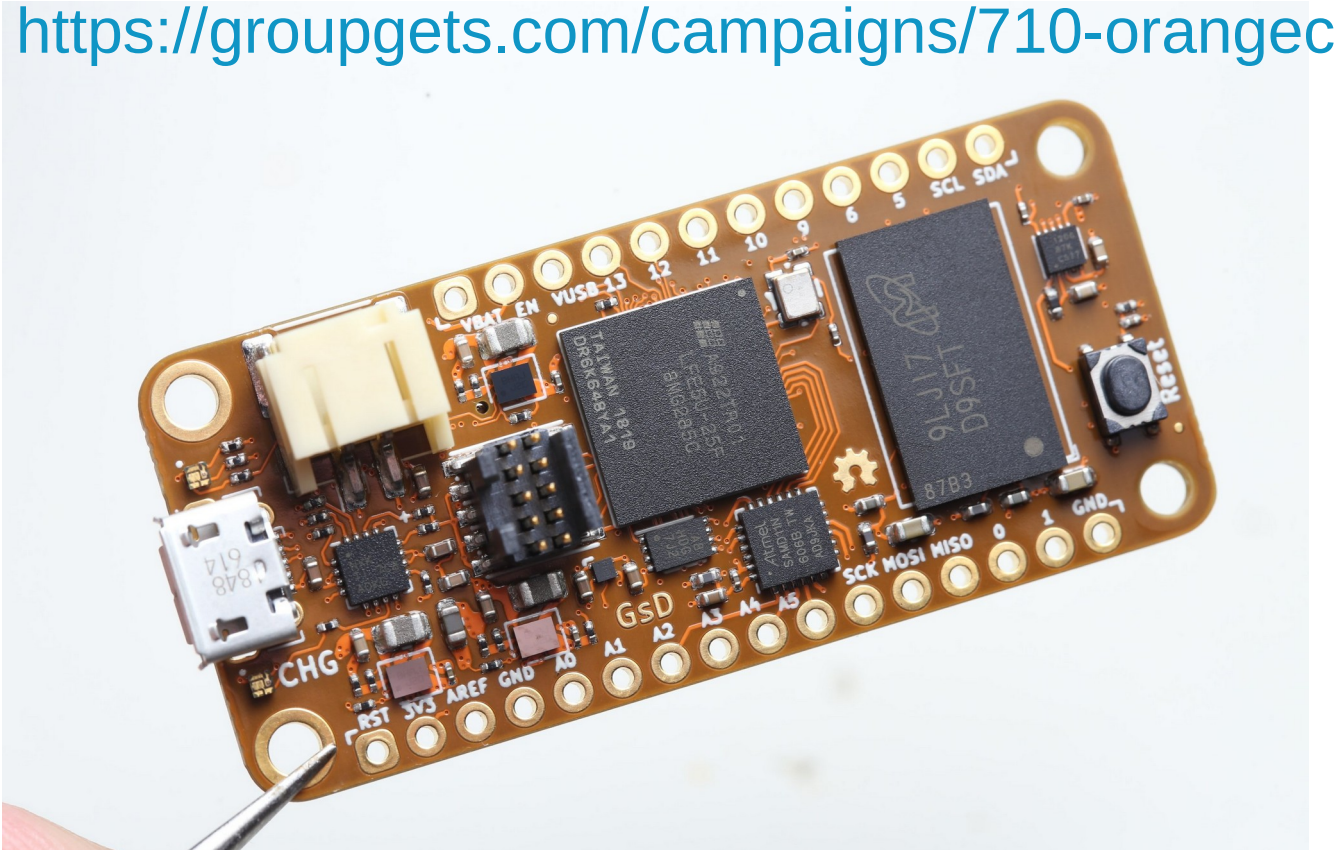
Drew Fustini

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

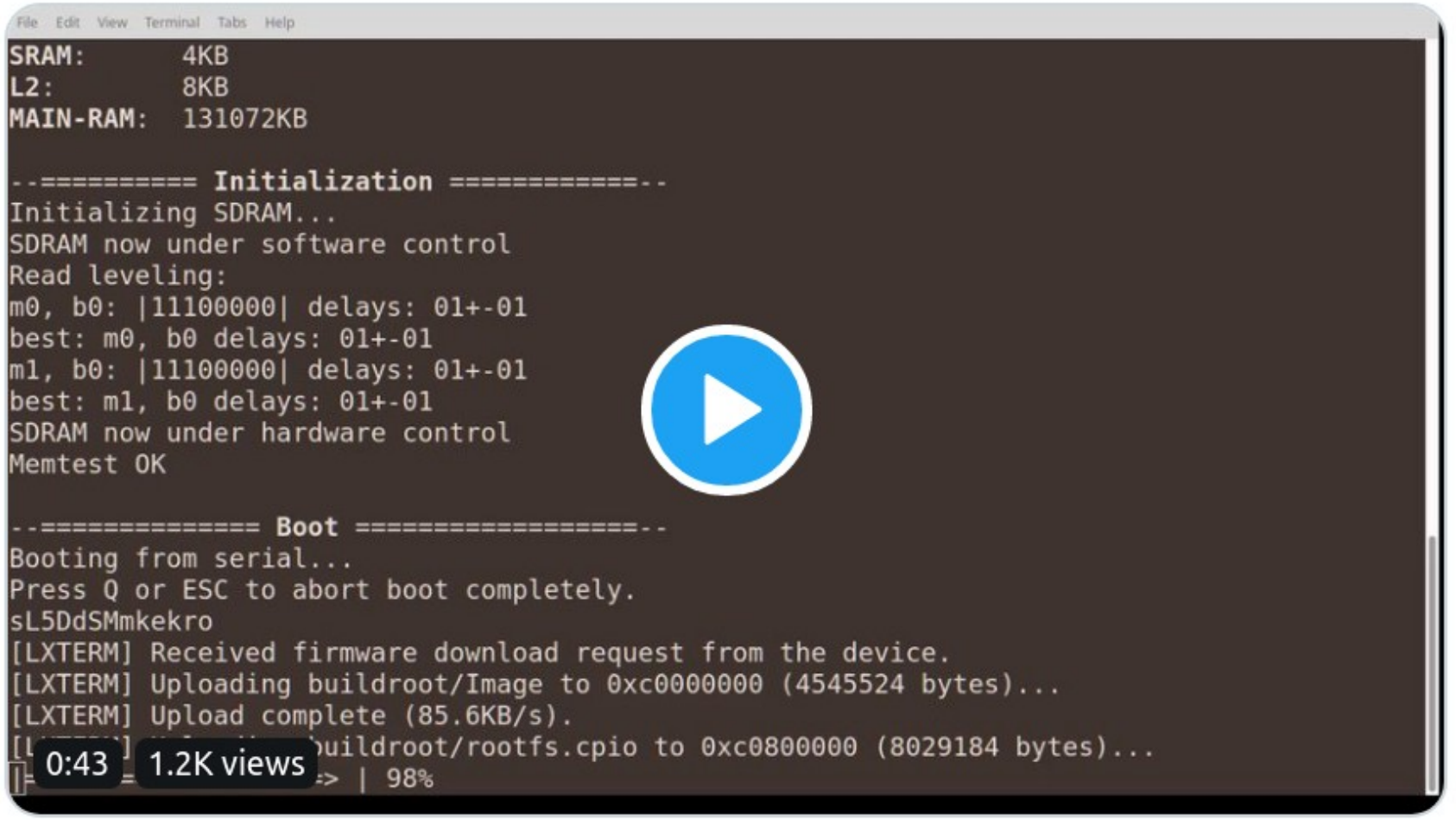
- Lattice ECP5 FPGA in Adafruit Feather form factor and 128MB DDR RAM:
  - Orange Crab by Greg Davill
    - https://github.com/gregdavill/OrangeCrab
    - https://groupgets.com/campaigns/710-orangecrab



Greg @ #36c3
@GregDavill

Replying to @mithro @pdp7 and 2 others

Done. 😃



4:55 PM · Dec 28, 2019 · Twitter Web App

Open Source ECP5 boards

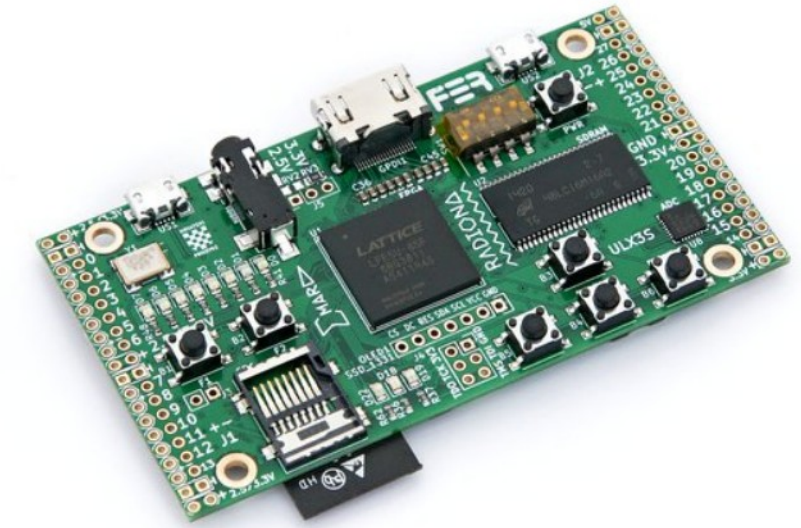
- Radiona.org ULX3S
  - https://www.crowdsupply.com/radiona/ulx3s

ULX3S

A powerful, open hardware ECP5 FPGA dev board

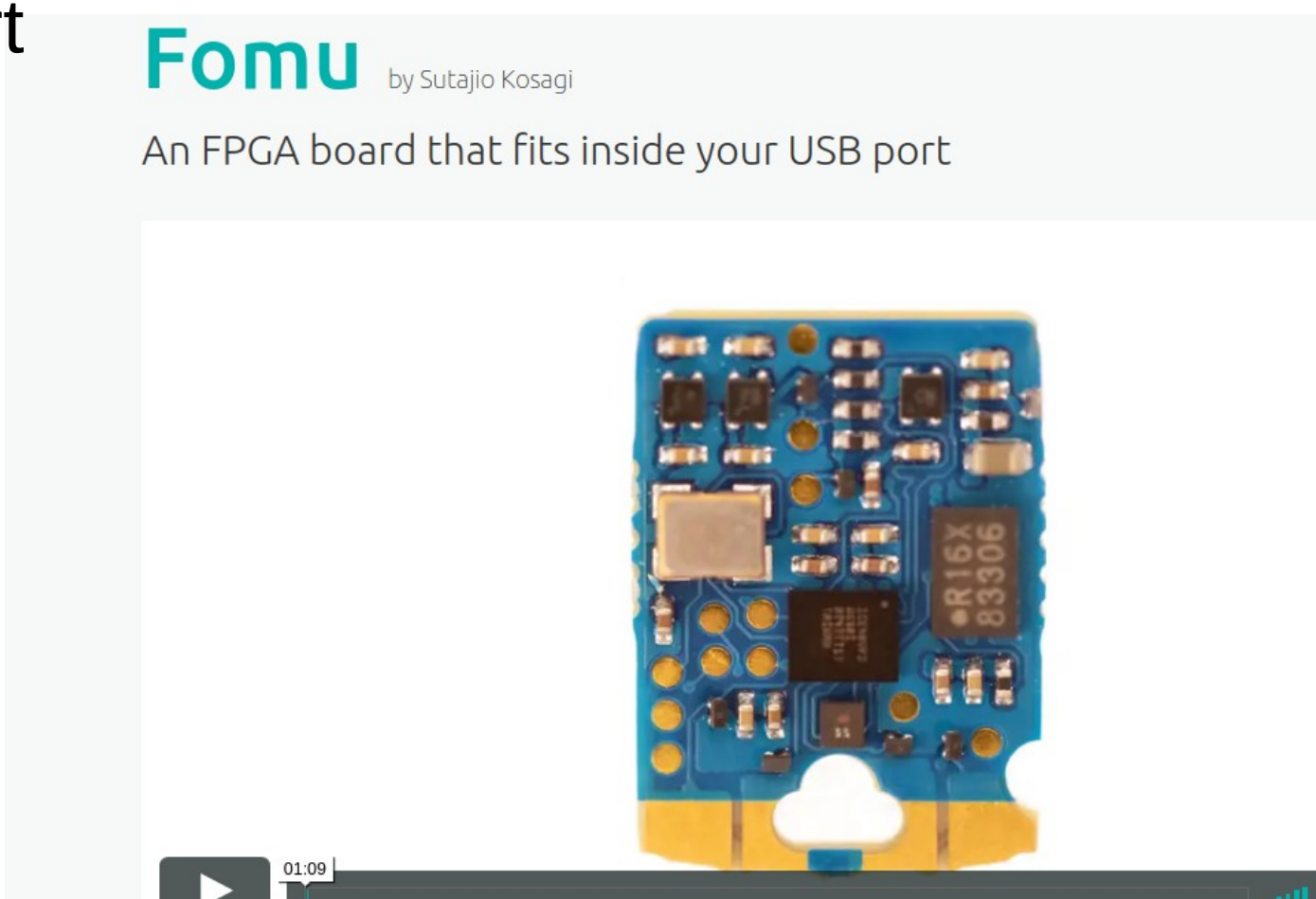
This project is coming soon. Sign up to receive updates and be notified when this project launches.

me@example.com Subscribe



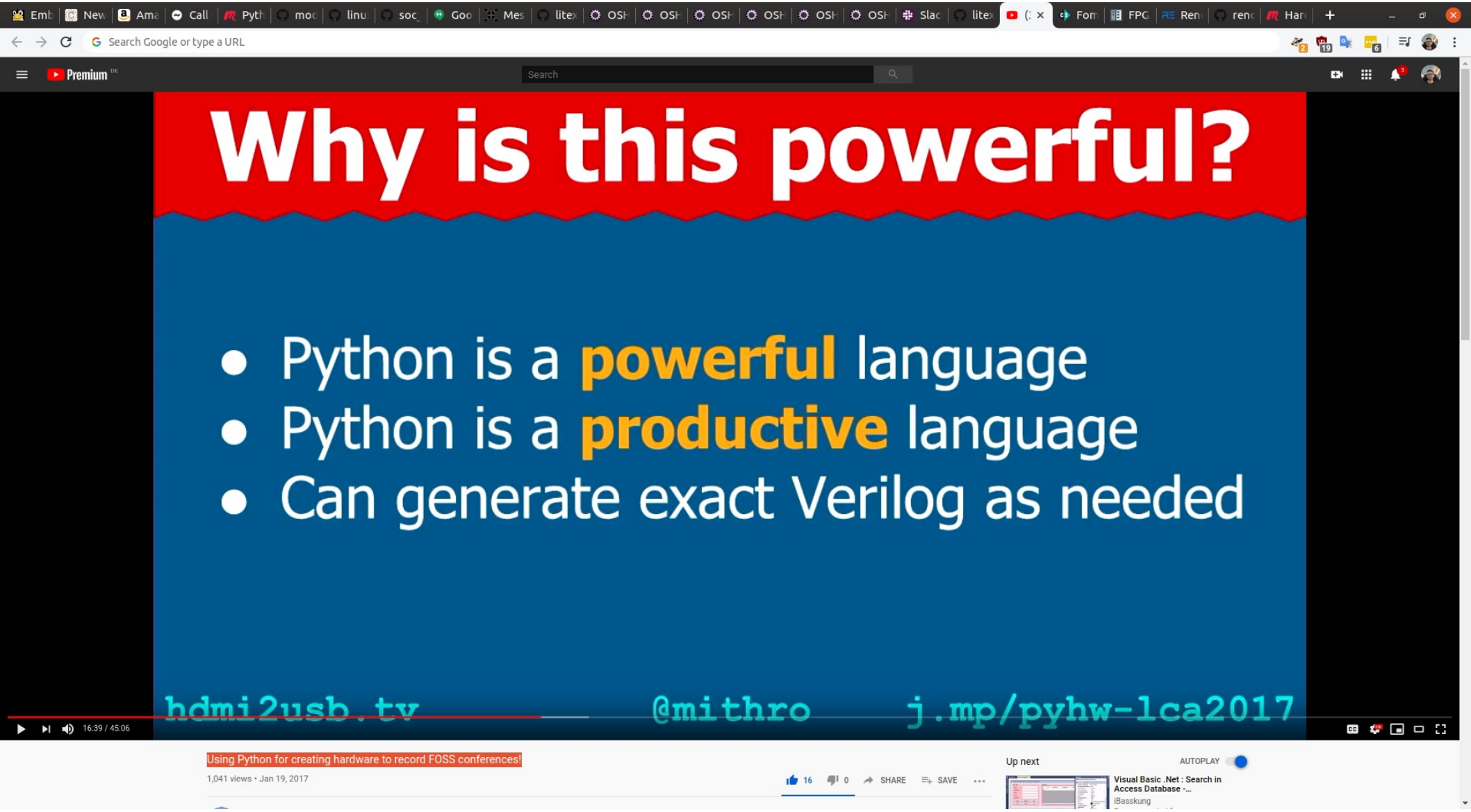
Fomu – great way to learn FPGAs!

- workshop.fomu.im
- crowdsupply.com/sutajio-kosagi/fomu
- Fits in USB port
- RGB LED
- 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



Using Python for creating hardware to record FOSS conferences!

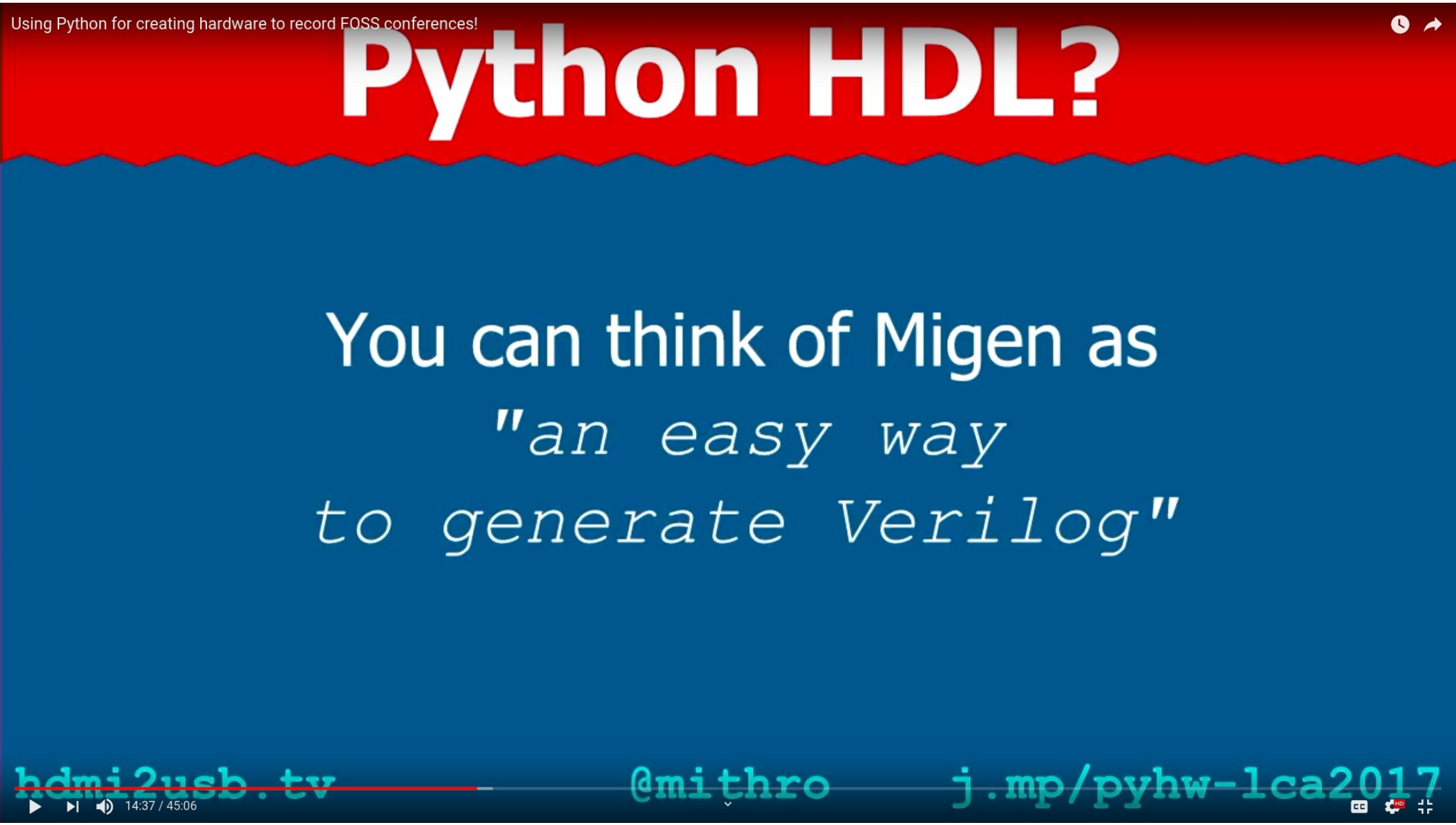
1,041 views • Jan 19, 2017

16 0 SHARE SAVE

Up next AUTOPLAY

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

iBasskung


Using Python for creating hardware to record FOSS conferences!
Python HDL?
You can think of Migen as
"an easy way
to generate Verilog"
hdmi2usb.tv
@mithro
j.mp/pyhw-lca2017
14:37 / 45:06

Kendryt

- Yes!
- “Using Python for creating hardware to record FOSS conferences!”
- Tim “mithro” Ansell
- youtube.com/watch?v=MkVX_mh5dOU


Using Python for creating hardware to record FOSS conferences!
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
hdmi2usb.tv
@mithro
j.mp/pyhw-lca2017
15:52 / 45:06