

Self-Hosting (Almost) All The Way Down

Gabriel L. Somlo, Ph.D.

CERT / SEI
Carnegie Mellon University
Pittsburgh, PA 15213

Copyright 2023 Carnegie Mellon University.

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

The views, opinions, and/or findings contained in this material are those of the author(s) and should not be construed as an official Government position, policy, or decision, unless designated by other documentation.

References herein to any specific commercial product, process, or service by trade name, trade mark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by Carnegie Mellon University or its Software Engineering Institute.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

This material may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other use. Requests for permission should be directed to the Software Engineering Institute at permission@sei.cmu.edu.

CERT® is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

DM23-0044

Executive Summary

- *Self-Hosting, Trusting-Trust, and Gateway*
 - ASIC & FPGA threat models, trustability trade-offs
- Slowest, most memory-constrained computer still able to boot **Fedora 37**: *50MHz* CPU, *512MB* RAM
 - **RocketChip**, **LiteX**, **yosys/nextpnr** FOSS HDL toolchain
 - Lattice ECP5 FPGA dev. board (**λC ECPIX-5 85F**)
 - Self-Hosting: Capable of building ECPIX-5 bitstream!

Self-Hosting (Compiler)

C compiler

- Written in its own language
- Compiles its own sources
- *Bootstrapping*
- *NOT* to be confused with *self-hosted Web services* (vs. hosted in the “cloud”)!

Compilers, Trusting Trust, and DDC

- Ken Thompson's **self-propagating C compiler hack**
 - malicious compiler inserts Trojan during compilation of *victim program*
 - clean sources → malicious binary (incl. *compiler's own sources!*)
 - compiler source hack *no longer needed* beyond 1st iteration!

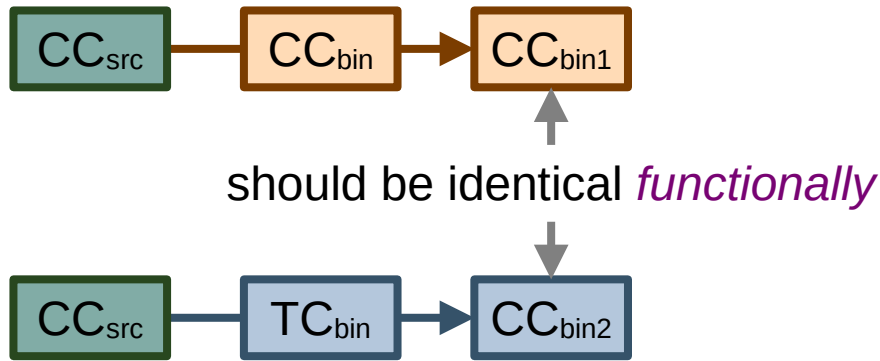
Compilers, Trusting Trust, and DDC

- Ken Thompson's [self-propagating C compiler hack](#)
 - malicious compiler inserts Trojan during compilation of *victim program*
 - clean sources → malicious binary (incl. *compiler's own sources!*)
 - compiler source hack *no longer needed* beyond 1st iteration!
- David A. Wheeler's mitigation: [Diverse Double Compilation \(DDC\)](#)
 - CC: "suspect" compiler; TC: "third party" compiler;

Compilers, Trusting Trust, and DDC

- Ken Thompson's [self-propagating C compiler hack](#)
 - malicious compiler inserts Trojan during compilation of *victim program*
 - clean sources → malicious binary (incl. *compiler's own sources!*)
 - compiler source hack *no longer needed* beyond 1st iteration!
- David A. Wheeler's mitigation: [Diverse Double Compilation \(DDC\)](#)

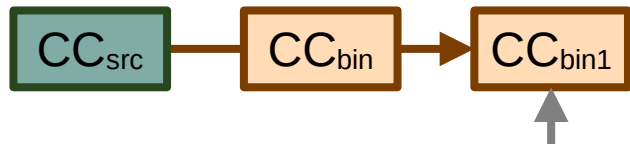
CC: "suspect" compiler; TC: "third party" compiler;



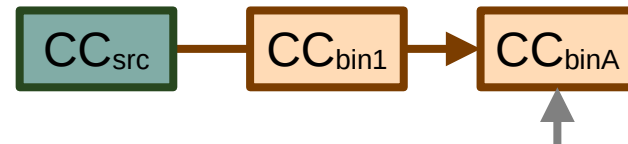
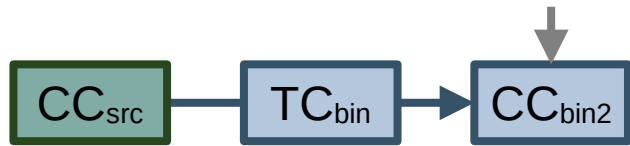
Compilers, Trusting Trust, and DDC

- Ken Thompson's [self-propagating C compiler hack](#)
 - malicious compiler inserts Trojan during compilation of *victim program*
 - clean sources → malicious binary (incl. *compiler's own sources!*)
 - compiler source hack *no longer needed* beyond 1st iteration!
- David A. Wheeler's mitigation: [Diverse Double Compilation \(DDC\)](#)

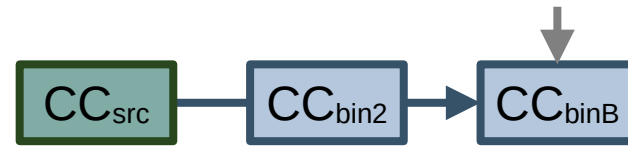
CC: "suspect" compiler; TC: "third party" compiler;



should be identical *functionally*



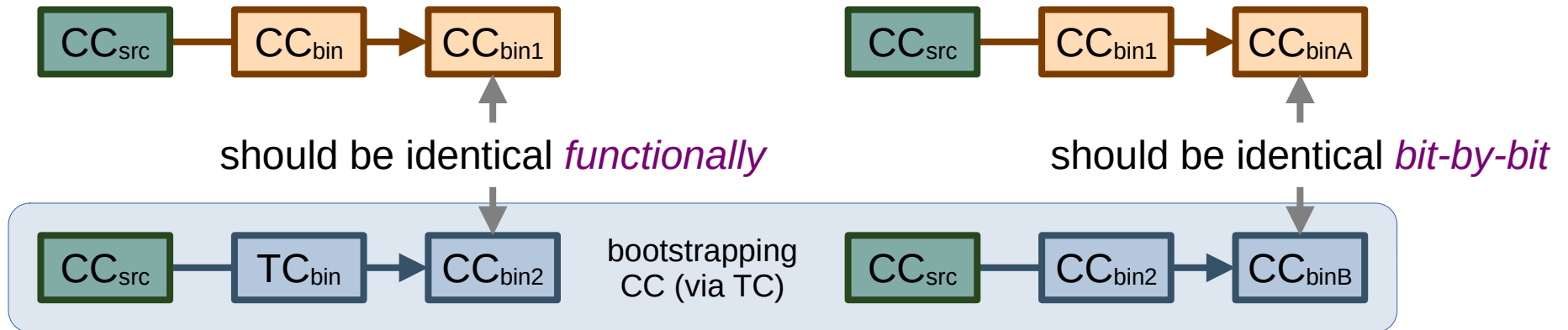
should be identical *bit-by-bit*



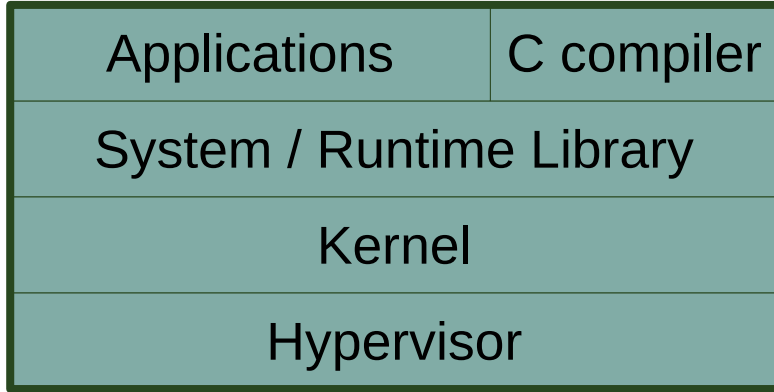
Compilers, Trusting Trust, and DDC

- Ken Thompson's [self-propagating C compiler hack](#)
 - malicious compiler inserts Trojan during compilation of *victim program*
 - clean sources → malicious binary (incl. *compiler's own sources!*)
 - compiler source hack *no longer needed* beyond 1st iteration!
- David A. Wheeler's mitigation: [Diverse Double Compilation \(DDC\)](#)

CC: "suspect" compiler; TC: "third party" compiler;

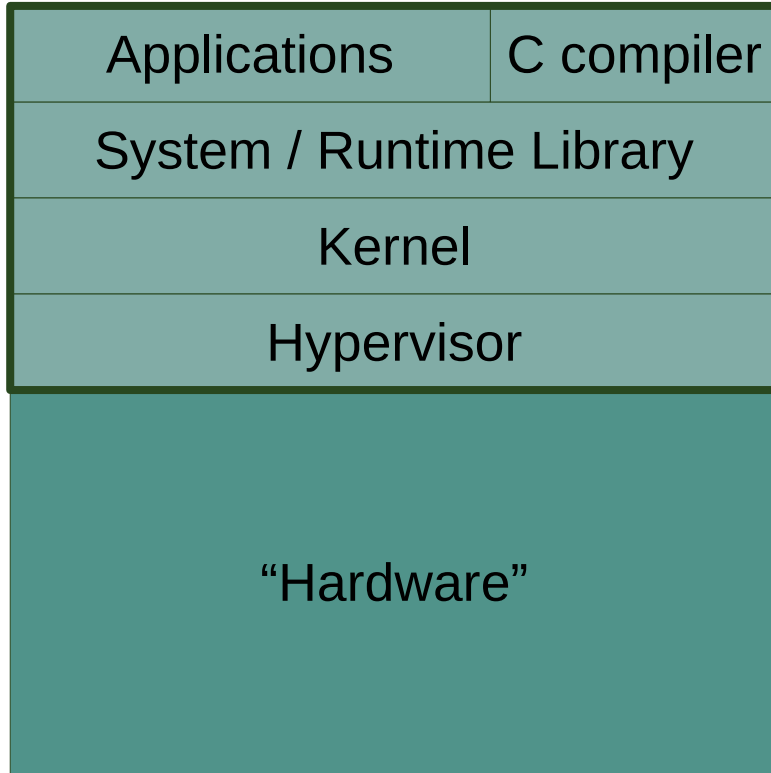


Self-Hosting Software Stack



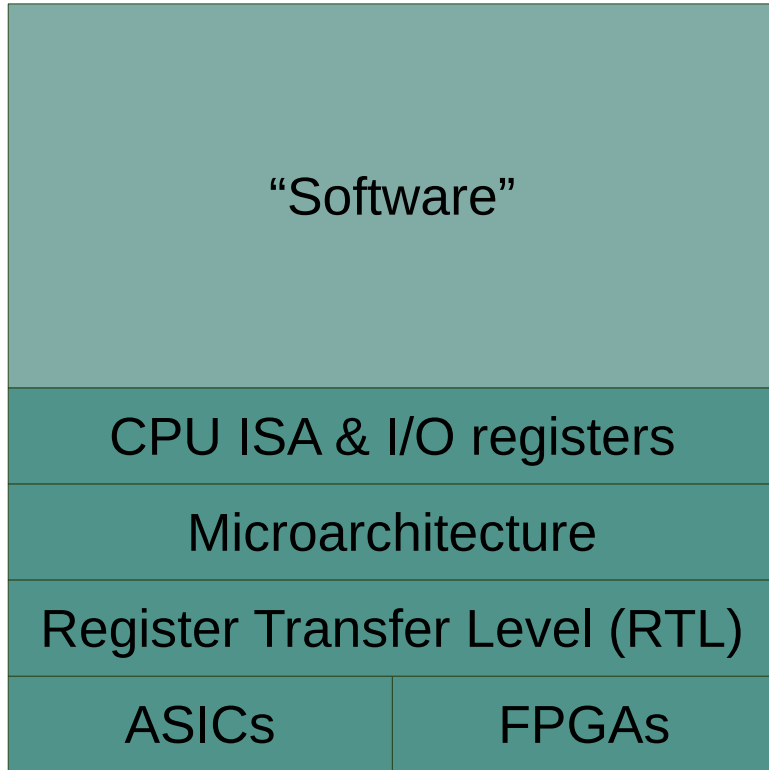
- Self-hosting compiler can build all software needed to support its own execution
- From FOSS sources to all components!

Self-Hosting Software Stack



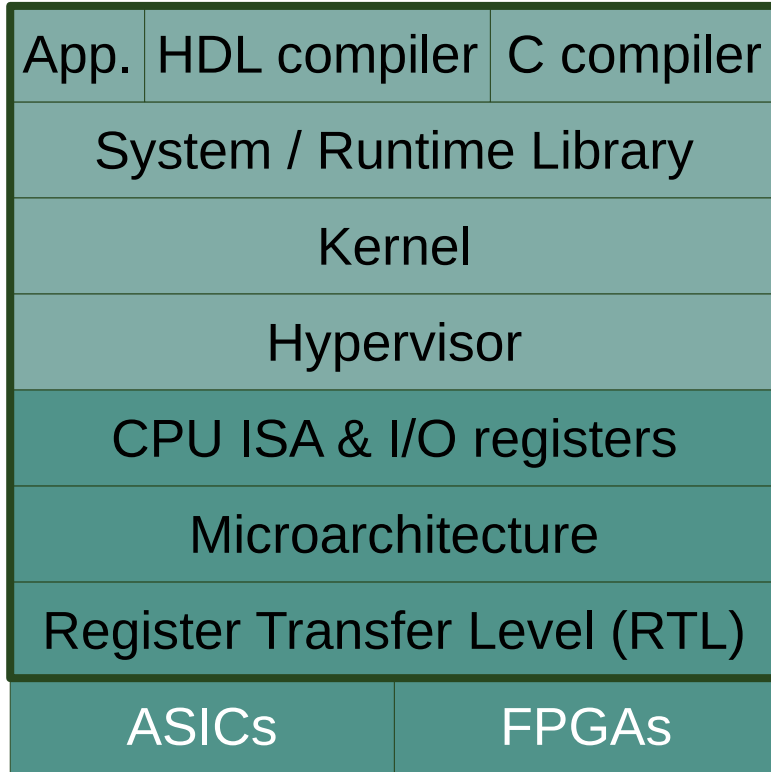
- Self-hosting compiler can build all software needed to support its own execution
- Relies on *Hardware*

More Details re. *Hardware*



- *Gateway* (written in HDL)
- *Physical* (i.e., silicon)

Self-Hosting Extended to Gateway



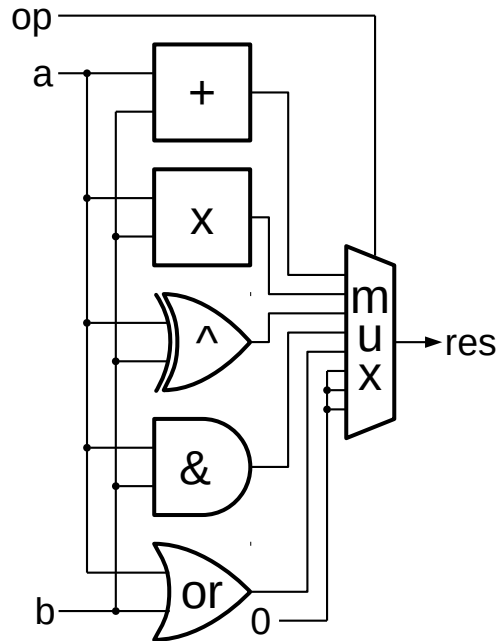
- C compiler → *Software*
- HDL compiler → *Gateway*
- Free / Libre sources to all software & gateway
- *Physical* layer (ASIC or FPGA) is out of scope!

Gateway Compilation Stages

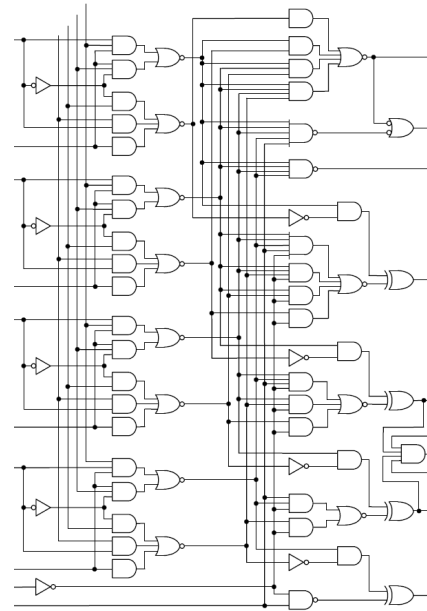
HDL Sources

```
module alu_mod (  
  // operator:  
  input  alu_op_t    op,  
  // operands:  
  input  logic [31:0] a, b,  
  // result:  
  output logic [31:0] res);  
  
always_comb begin  
  unique case (op)  
    ALU_ADD: res = a + b;  
    ALU_MUL: res = a * b;  
    ALU_XOR: res = a ^ b;  
    ALU_AND: res = a & b;  
    ALU_OR : res = a | b;  
    default: res = 32'b0;  
  endcase  
end  
endmodule: alu_mod
```

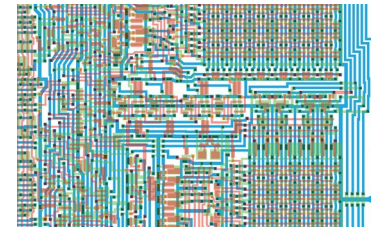
Elaboration



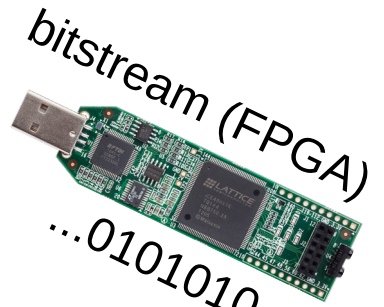
Synthesis,
Optimization



Tech. Mapping,
Place & Route

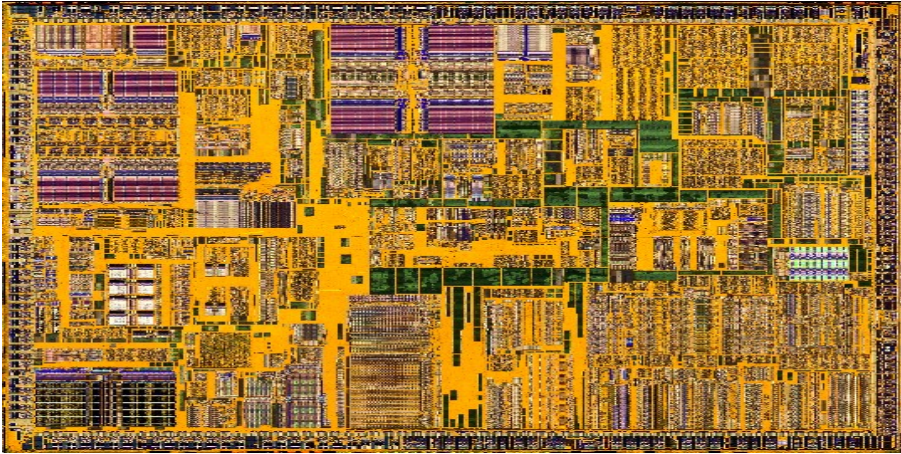


mask (ASIC)



...0101010...

ASICs vs. FPGAs



- Application Specific Integrated Circuit
- dedicated, optimized etched silicon
 - photolithographic masks
- *hard IP* cores



- Field Programmable Gate Array
- grid: programmable blocks, interconnect
 - bitstream
- *soft IP* cores

Hardware Attack Surface

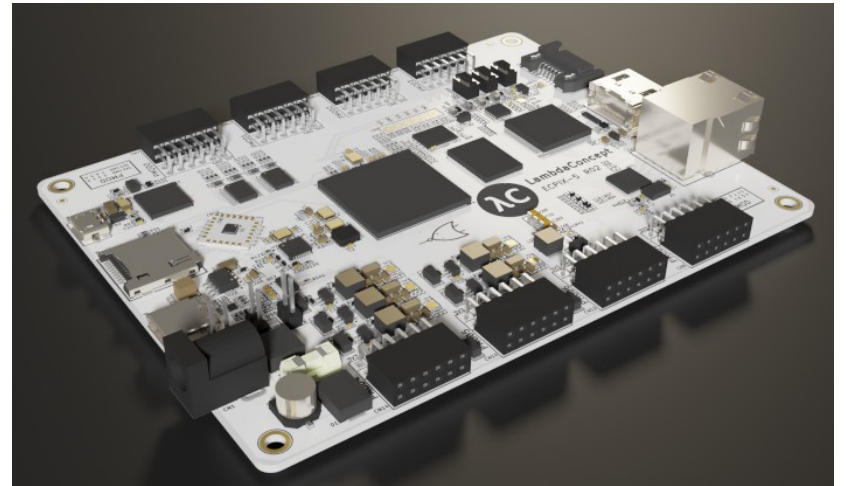
- Fabrication (Malicious ASIC Foundry)
 - masks reverse engineered, modified to insert malicious behavior into ASIC
 - privilege escalation CPU backdoor ([A2 Trojan](#))
 - tamper with silicon [doping polarity](#) (e.g., to weaken hardware-based crypto)
 - problematic to test / verify after the fact!
 - mitigated by using FPGAs: hard to predict where to add *useful* Trojan silicon!
- Compilation ([Malicious HDL Toolchain](#))
 - generate *malicious* design from *clean* HDL sources
- Design Defects (accidental or intentional HDL bugs)
 - [Spectre](#), [Meltdown](#), etc.

Why *Self-Hosting* ?

- *Freedom* and *Independence!*
 - From black-box, closed, non-Free dependencies
- Trust *deployed system* to a degree equal to its *comprehensive set of sources*:
 - Gateway HDL
 - Software (including C, HDL compilers)

Self-Hosting Fedora Demo

- LambdaConcept
ECPIX-5 85F board
- LiteX (Rocket CPU)
- yosys, trellis, nextpnr
- OpenSBI
- Fedora's riscv64 port



LiteX bitstream for the ECPIX-5

```
litex-boards/litex_boards/targets/lambdaconcept_ecpix5.py --build \  
  --cpu-type rocket --cpu-variant fulld --sys-clk-freq 50e6 \  
  --with-ethernet --with-sdcard \  
  --yosys-flow3 --nextpnr-timingstrict --nextpnr-seed $RANDOM \  
  --csr-csv ./csr.csv
```

- Use `.csv` and `.dts` from `pythondata-cpu-rocket` to build customized DT with `chosen` / `bootargs`:

```
console=liteuart earlycon=liteuart,0x12006800 swiotlb=noforce ro  
root=/dev/mmcblk0p2 enforcing=0 systemd.unit=multi-user.target  
systemd.default_timeout_start_sec=360s
```

OpenSBI firmware with built-in DTB

```
dtc -o dtb ecpix5_fedora.dts -o ecpix5_fedora.dtb  
  
make CROSS_COMPILE=riscv64-unknown-linux-gnu- \  
    PLATFORM=generic FW_FDT_PATH=./ecpix5_fedora.dtb
```

- Copy to first (vfat) partition of SDcard, along with kernel, initrd, and boot.json:

```
{  
    "initramfs.img": "0x83000000",  
    "Image":        "0x80200000",  
    "fw_jump.bin":  "0x80000000"  
}
```

Fedora with custom kernel, initramfs

- Download Fedora riscv64 pre-built [image](#)
 - Copy root partition to second (ext4) SDcard partition
- Still requires *custom* kernel at the moment
 - Upstream IRQ support for LiteUART (serial console) still [pending](#)
 - Several newer stock Fedora riscv64 `CONFIG_*` options cause kernel to crash during early boot
 - currently under investigation
 - Built on a Fedora riscv64 QEMU VM

https://www.contrib.andrew.cmu.edu/~somlo/BTCP/self_hosting_fedora.html

Demo

- Boot Fedora on ECPIX-5
 - Run `dnf install yosys trellis nextpnr`
 - Build blinky bitstream from Verilog, for ECPIX-5
- Load ECPIX-5 with resulting blinky bitstream

Future Work

Near term:

- Isolate and debug stock `CONFIG_*` kernel options
- Improve LiteX/Rocket SoC boot process
 - Programmatically generate DTS
 - Option to build DTB directly into LiteX BIOS
- Improve and upstream LiteSATA support

Future Work

Medium term:

- FOSS HDL toolchains targeting larger FPGAs
 - e.g., 8 Rocket cores at >100MHz, 8GB RAM on VC707
 - Dependency on Vivado currently precludes self-hosting!
- Fancier IP blocks: e.g., GPU, PCI
 - Or at least LiteVideo for 2-D GUI support

Future Work

Long Term:

- Better understanding of ASIC fabrication process

Sci-Fi Long Term:

- Making our own ASICs (from Free/Open sources)
 - Sam Zeloof's [home chip fab](#) experiments
 - Consumer-grade [molecular assemblers](#) ([Diamond Age](#))

Thank You!

Q&A