Improving Hardware Assurance through Self-Hosting

Gabriel L. Somlo, Ph.D.

CERT / SEI
Carnegie Mellon University
Pittsburgh, PA 15213

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.

DM21-0321

Self-Hosting (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!
- David A. Wheeler's mitigation: Diverse Double Compilation (DDC)
  - suspect compiler A: sources S_A, binary B_A
  - trusted compiler T: binary B_T

$S_A \rightarrow B_A \rightarrow X$ $\qquad$ $S_A \rightarrow B_T \rightarrow Y$

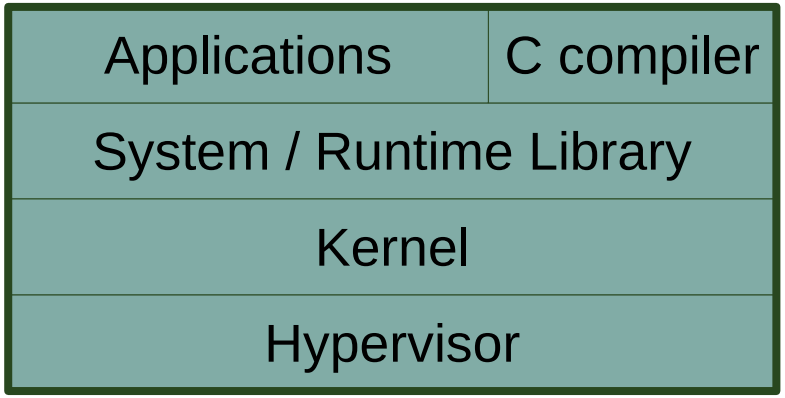
  - X and Y are *functionally identical*, but *different binaries*

$S_A \rightarrow X \rightarrow X_1$ $\qquad$ $S_A \rightarrow Y \rightarrow Y_1$

  - X_1 and Y_1 must be *identical binaries* (output of two *functionally identical* compilers)!

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.
DM21-0321

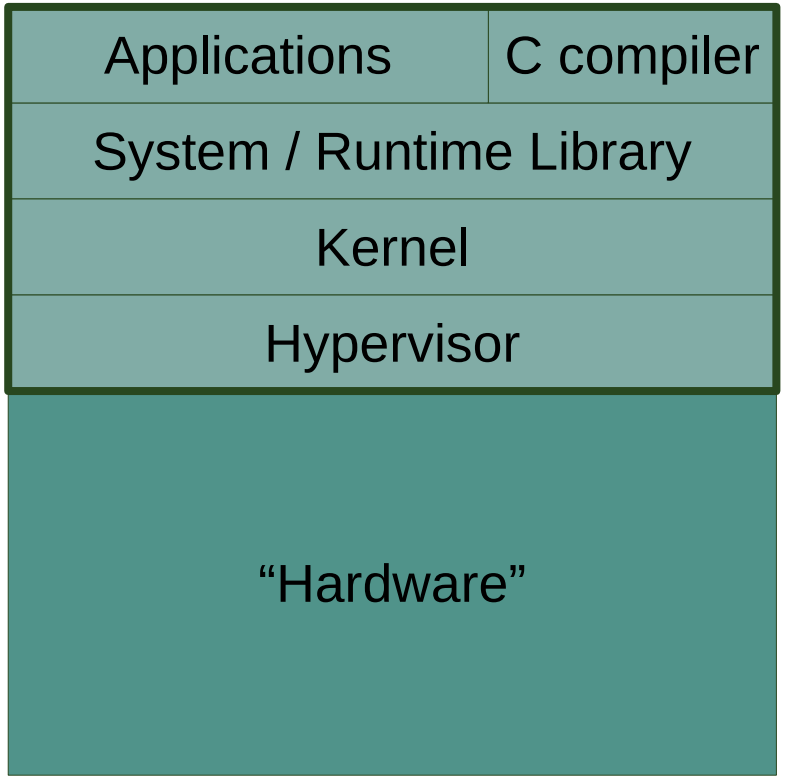
Self-Hosting Software Stack



- Self-hosting compiler can build all software needed to support its own execution

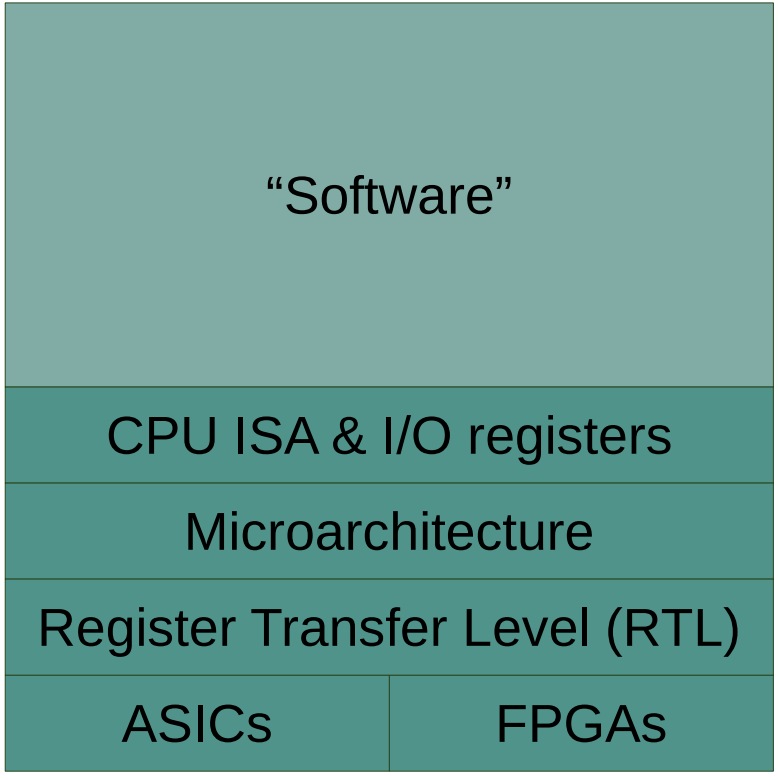
- From available sources to all components!

Self-Hosting Software Stack



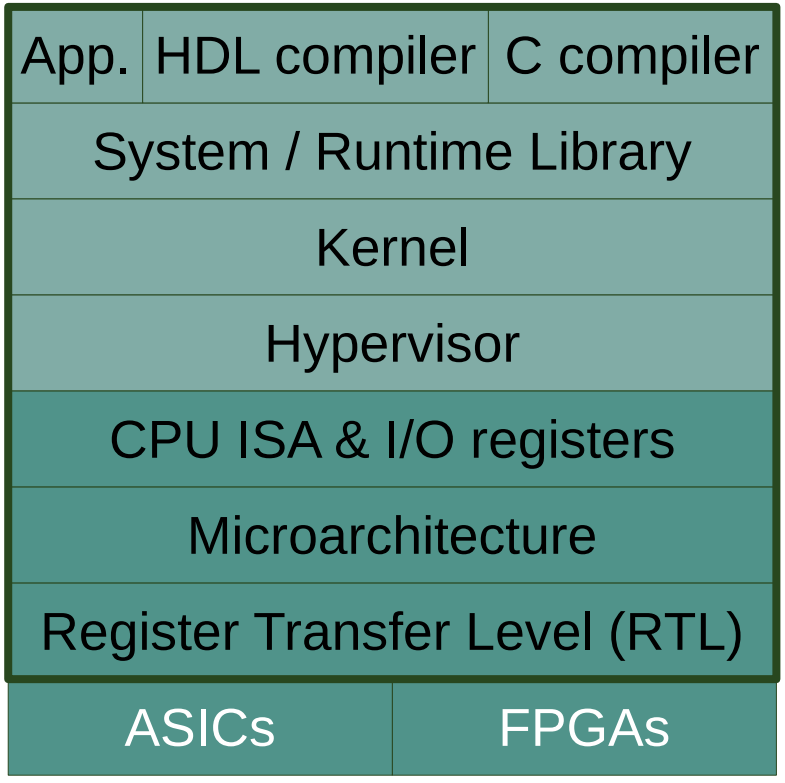
- Self-hosting compiler can build all software needed to support its own execution
- Relies on (deployed on top of) *Hardware*

More Details re. *Hardware*



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

Self-Hosting Extended to Gateware



- C compiler → *Software*
- HDL compiler → *Gateware*
- Free / Libre sources for all components!
- *Physical* (silicon, ASICs or FPGAs) out of scope!

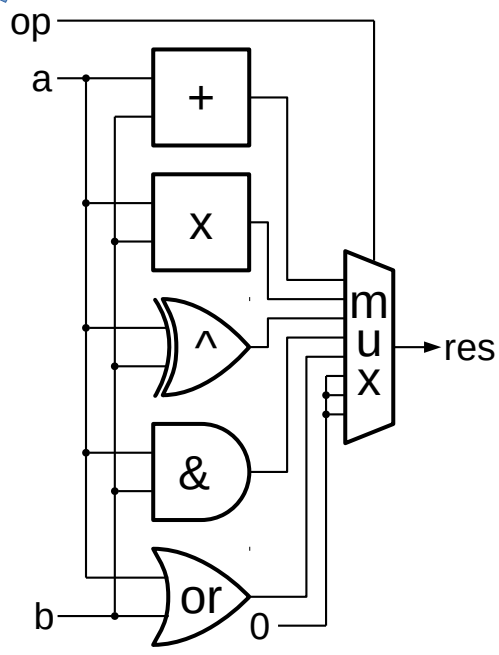
Gateware Compilation Stages

HDL Sources

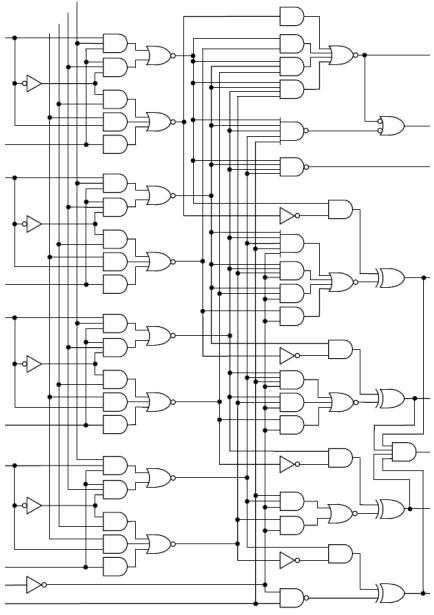
```
module alu_mod (
  // operator:
  input  alu_op_t     op,
  // operands:
  input  logic [31:0] a, b,
  // result:
  output logic [31:0] res);

  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
```

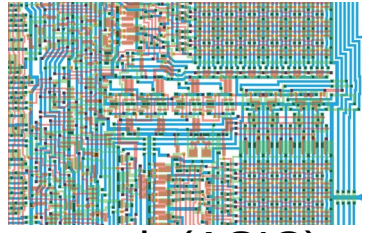
Elaboration



Synthesis, Optimization

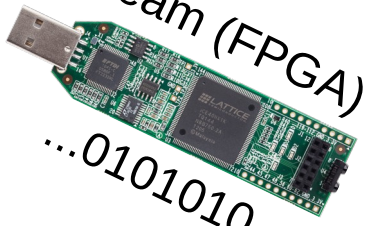


Tech. Mapping, Place & Route



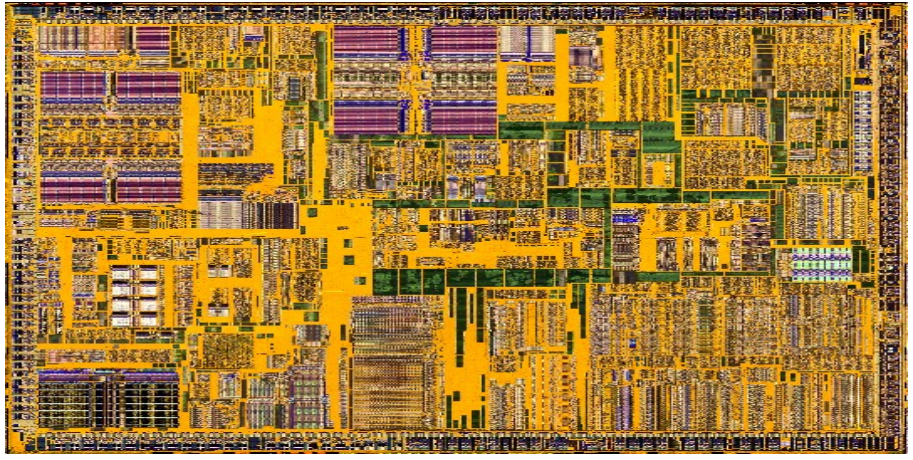
mask (ASIC)

bitstream (FPGA)

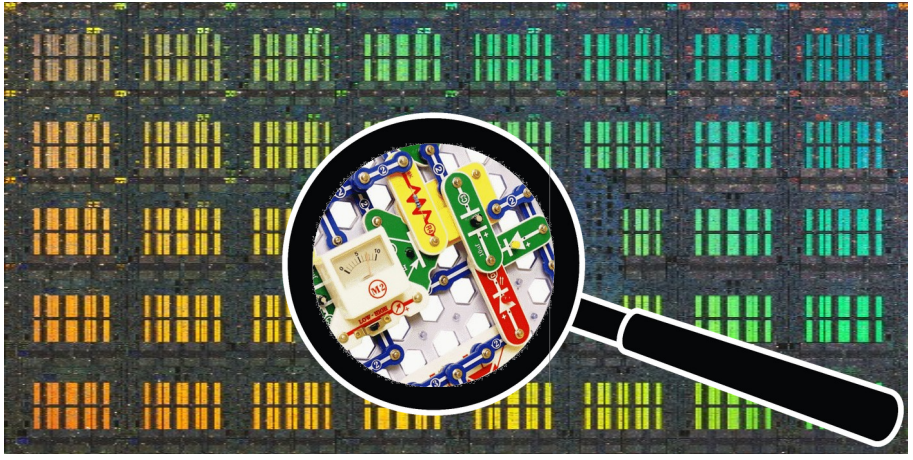


...0101010...

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.

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.
DM21-0321

Why *Self-Hosting* ?

- Freedom! Liberty! Independence! :)
  - From *black-box*, and / or *non-Free* dependencies
- Trust a running *software + gateware stack* to the same extent as its *cumulative sources*
  - Gateware HDL sources
  - Software sources (including C and HDL compilers)

Bootstrap Software+Gateware Stack

- *Host* (x86_64/Linux):
  - Build clean C (cross-)compiler (using DDC for addt'l assurance, if necessary)
  - Build clean HDL compiler (for both x86_64 and rv64gc)
  - Cross-compile target (rv64gc) software stack
  - Build gateware (FPGA bitstream) for target system
- *Target* (rv64gc/Linux):
  - Program FPGA board with gateware/bitstream
  - Boot into target software stack
    - Self-hosting from this point forward!
  - Natively rebuild gateware bitstream, software stack, from sources, as needed

LiteX + Rocket SoC Block Diagram

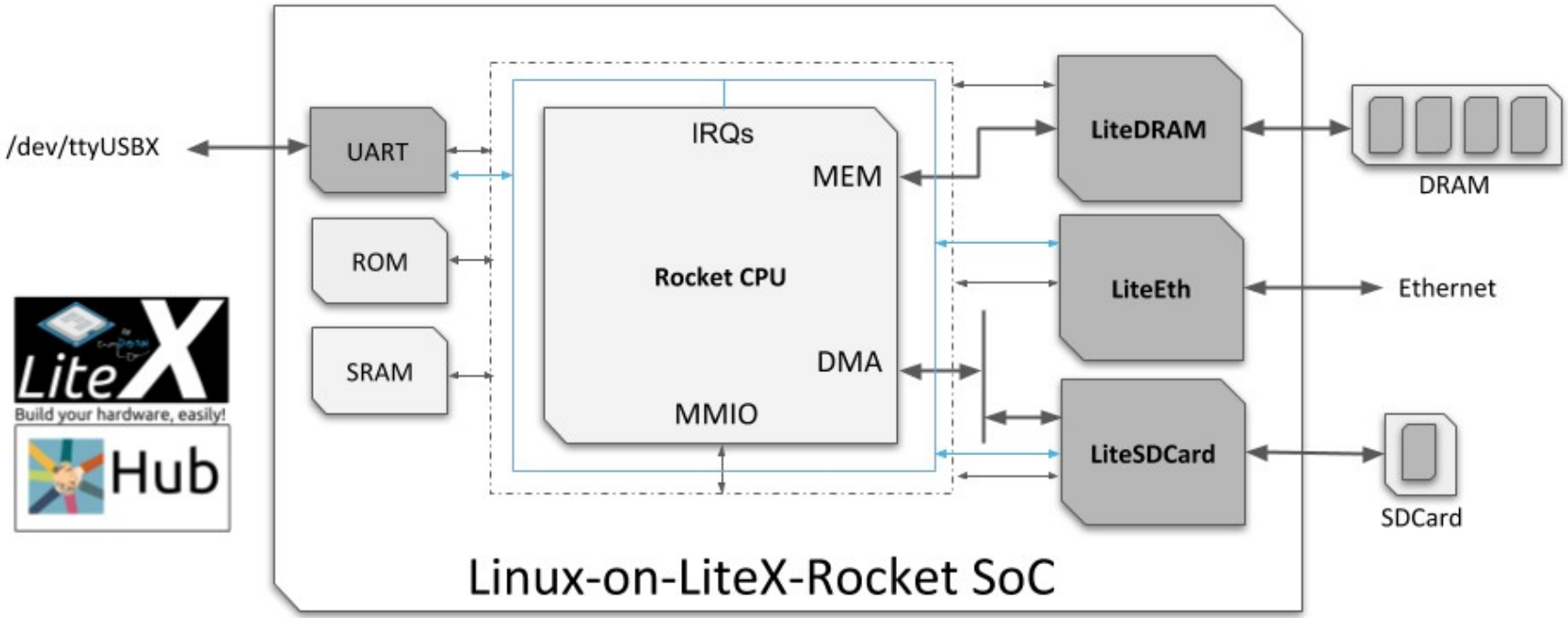
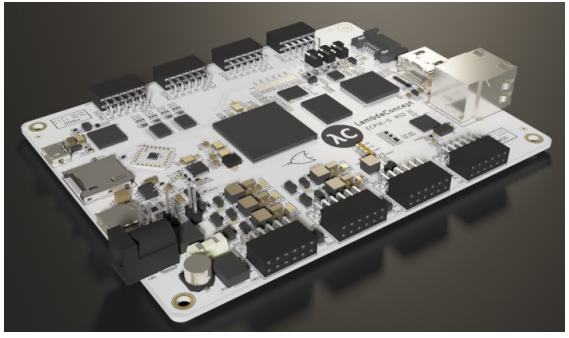


Image credit: Florent Kermarrec

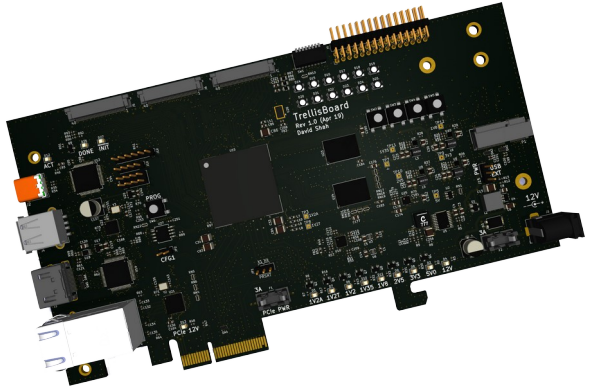
Ingredients

- Free / Open software *and* gateware sources:
  - Linux, BusyBox, BBL
  - LiteX, RocketChip
- Free / Open software *and* HDL toolchains:
  - gcc
  - yosys, trellis, nextpnr (on Lattice ECP5 FPGAs)

Try it on your own FPGA board!



ECPIX-5



trellisboard

ecp5-5g-versa

Build Instructions

- https://github.com/litex-hub/linux-on-litex-rocket
- Follow along on a pre-configured Fedora VM:
  - http://mirror.ini.cmu.edu/litex/litexdemo.f32.ova
    - Built for VMWare (Fusion / Workstation), for convenience
    - Link availability *not* guaranteed beyond April 2021!
    - Login: *user*
    - Password: *tartans*
    - Pre-installed with toolchains, sources

Demo, then Q&A

- Get in touch (on FreeNode IRC): *#litex*

- Thank you!