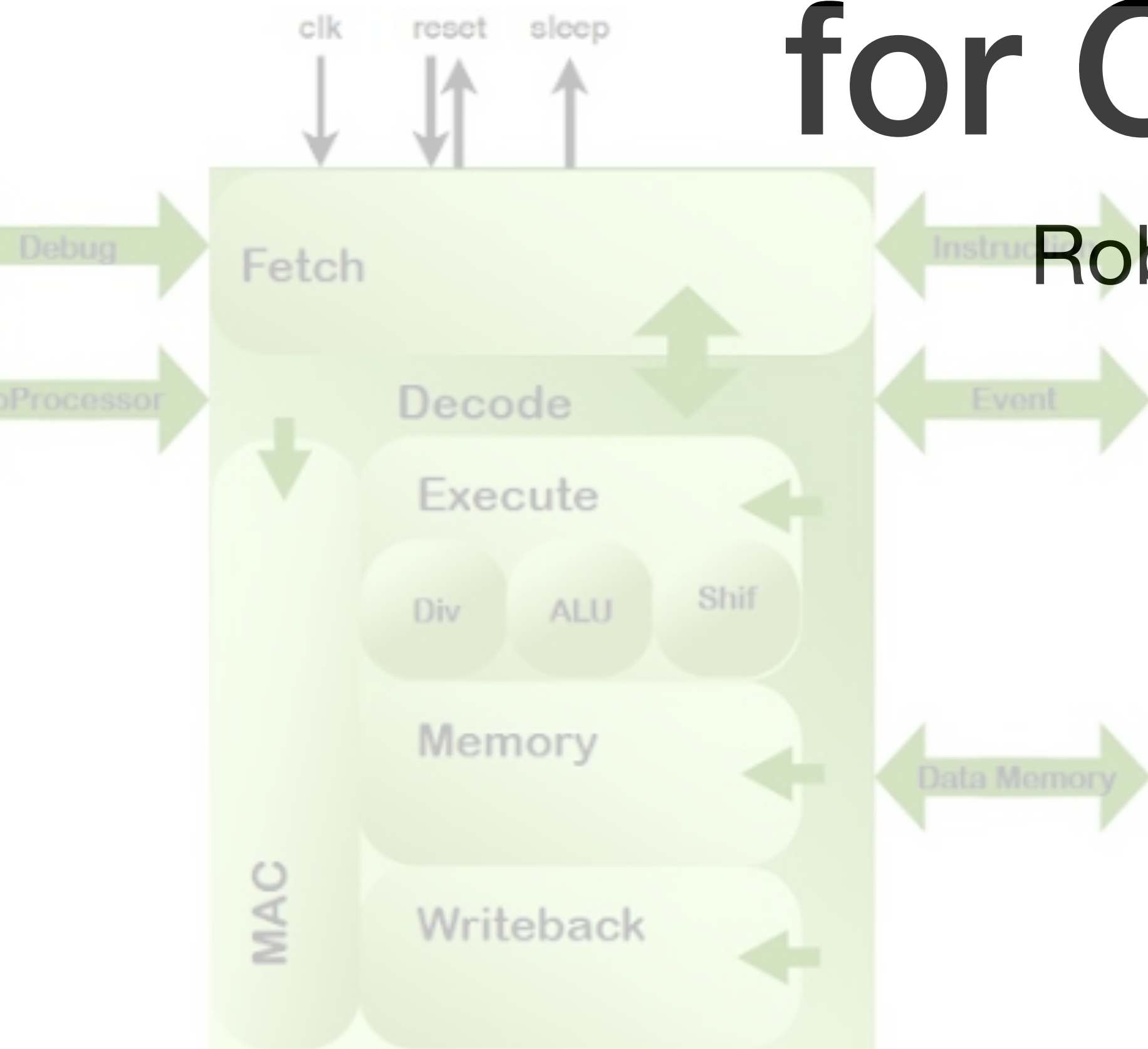


# J-Core Open Processor SoC for Open Hardware



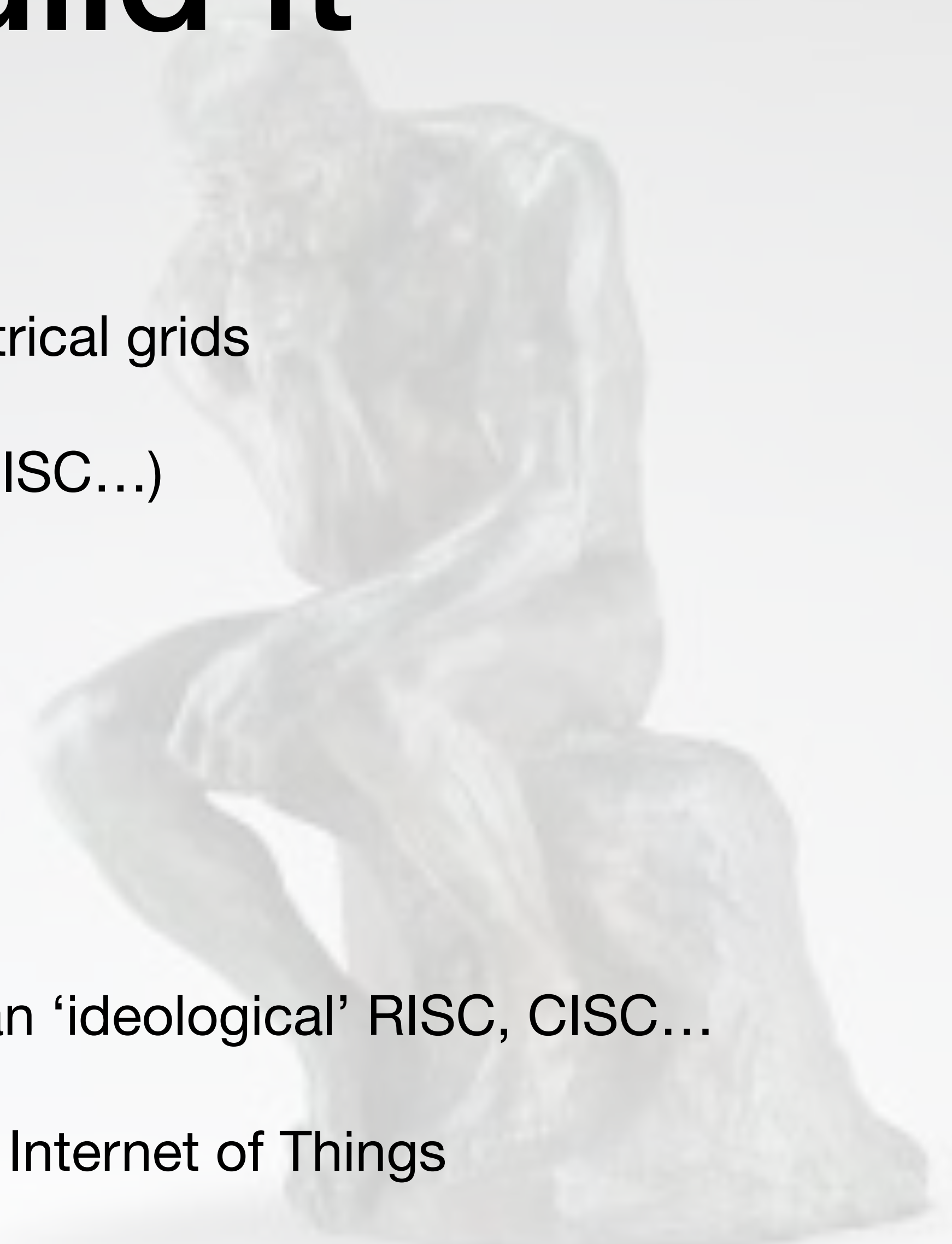
Rob W. Landley, D. Jeff Dionne  
Jan 2021

# 1. What is this

- J-Core is a patent free, Open Hardware implementation of the SuperH ISA
- SHCompact is a highly optimized ISA targeting high level languages
- Contemporary, cleaner, locally grown alternative to Motorola 68000
- The fundamental patents on the 16 bit ISA encoding lead to ARM Thumb
- Best selling CPU core in 1990s for 3 years.
- J-Core family is SHCompact+, with SMP extensions for FPGA and ASICs

# Why did we build it

- J-Core is purpose built: for connected embedded systems
  - Not an exercise: Deployed in 'Critical Infrastructure', electrical grids
- We surveyed available choices (SPARC, MIPS, ARM, OpenRISC...)
- Patents had expired, so provably safe to implement.
  
- SH is Cool: Local peak in ISA design thinking
  - ISA Design driven by e.g. compiler output statistics, not an 'ideological' RISC, CISC...
  - Embedded systems focus: Before things were called the Internet of Things
  - Instruction memory density and bandwidth was a large design consideration



# Historical Perspective

- Hitachi had a need to develop an in house ISA
  - At the time, Hitachi was licensing m68k, not a long term strategy
- The designers of the ISA were sought after by all the field leaders
- Design wins in Sega games, Bosch engine controllers drove volume
- Multiple deeply embedded SH processors powered Japanese cell phones
  
- Then, the market reorganized.

# Good Technology

- SHCompact is an excellent and solid technological foundation.
  - Well researched and documented design decisions. Unified ISA
- J-Core is a clean reimplementation, in high level dialect of VHDL
  - Designed with the ESA/Gaisler so called '2 Process Method'
  - Extensive use of code generators for the ISA and SoC bus structures
- Implemented from the ground up to do signal processing, networking, and run Linux

```
-- Interface Library for the HS-2J0 CPU core
library ieee;
use ieee.std_logic_1164.all;

package cpu2j0_pack is
  type cpu_instruction_o_t is record
    en      : std_logic;
    jp      : std_logic_vector(31 downto 1);
  end record;
  constant NULL_INST_0 : cpu_instruction_o_t := (en => '0',
);

  type cpu_instruction_i_t is record
    d      : std_logic_vector(15 downto 0);
    ack    : std_logic;
  end record;

  type cpu_data_o_t is record
    en      : std_logic;
    a      : std_logic_vector(31 downto 0);
    wr      : std_logic;
    we      : std_logic_vector(3 downto 0);
    d      : std_logic_vector(31 downto 0);
  end record;
  constant NULL_DATA_0 : cpu_data_o_t := (en => '0', a => (0
);

  type cpu_data_i_t is record
    a      : std_logic_vector(31 downto 0);
    ack    : std_logic;
  end record;

  type cpu_debug_o_t is record
    ack    : std_logic;
    d      : std_logic_vector(31 downto 0);
    rdy    : std_logic;
  end record;

  type cpu_debug_cmd_t is (BREAK, STEP, INSERT, CONTINUE);
```

# Roadmap

- Processor Family

➔ J0: a 16 bit datapath, 15k ASIC gates (target). 3 stage pipeline

✓ J1: Simple but full 32bit machine. 30k ASIC gates. 5 stage pipeline

✓ J2/J2SMP: Full Harvard, cache, SMP capable. 50k gates. Parallel MAC

✓ J32: Virtual Memory (MMU). 100k ASIC gates

➔ J32FM: Multi-issue, FPU. 200k ASIC gates (target).

✓ SoC Platform, Peripheral Library, SoC interconnect and top-level generators

# Application Specific...

- Each member designed for a specific application space
  - J0, J1: Deeply embedded systems
  - J2, J2smp: Network devices
  - J32: Compute devices

# Simple Application Example: VPN



- A hardware accelerated VPN device for remote telework.
  - J2SMP running Linux
  - Hardware cryptographic engine, coupled to a J1 processor
  - HSM functionality, and high speed.
- Prototype in FPGA: few weeks, new boards: a few months.
  - ASIC when the project reaches maturity.

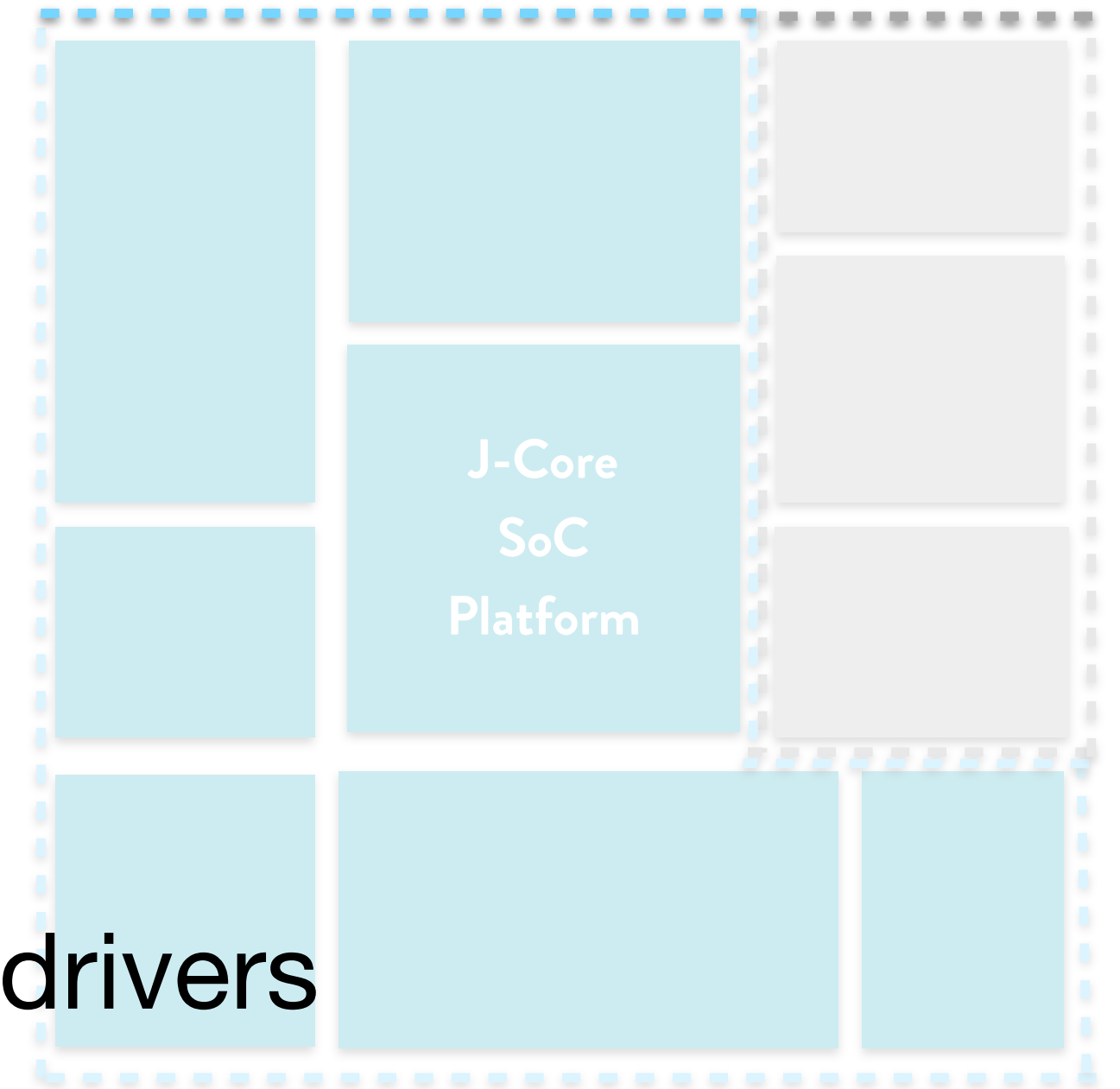


# FPGAs and Integrated Circuits

- Now cost on par in an FPGA with that of an equivalent microcontroller
  - J1 fits in a iCE40 UltraPlus 5k device -> 32bit CPU, 32k ROM + 128k RAM
- Performance equivalent to dedicated solutions
  - \$15 FPGA can host a dual core J2, plus e.g. a GPS baseband
  - Some applications need the FPGA anyway, whole SoC comes for free
- ASIC ready: J2 CPU is only  $0.45\text{mm}^2$  in 180nm, runs 125MHz.
  - J1 on a modern process is tiny, cents per die.

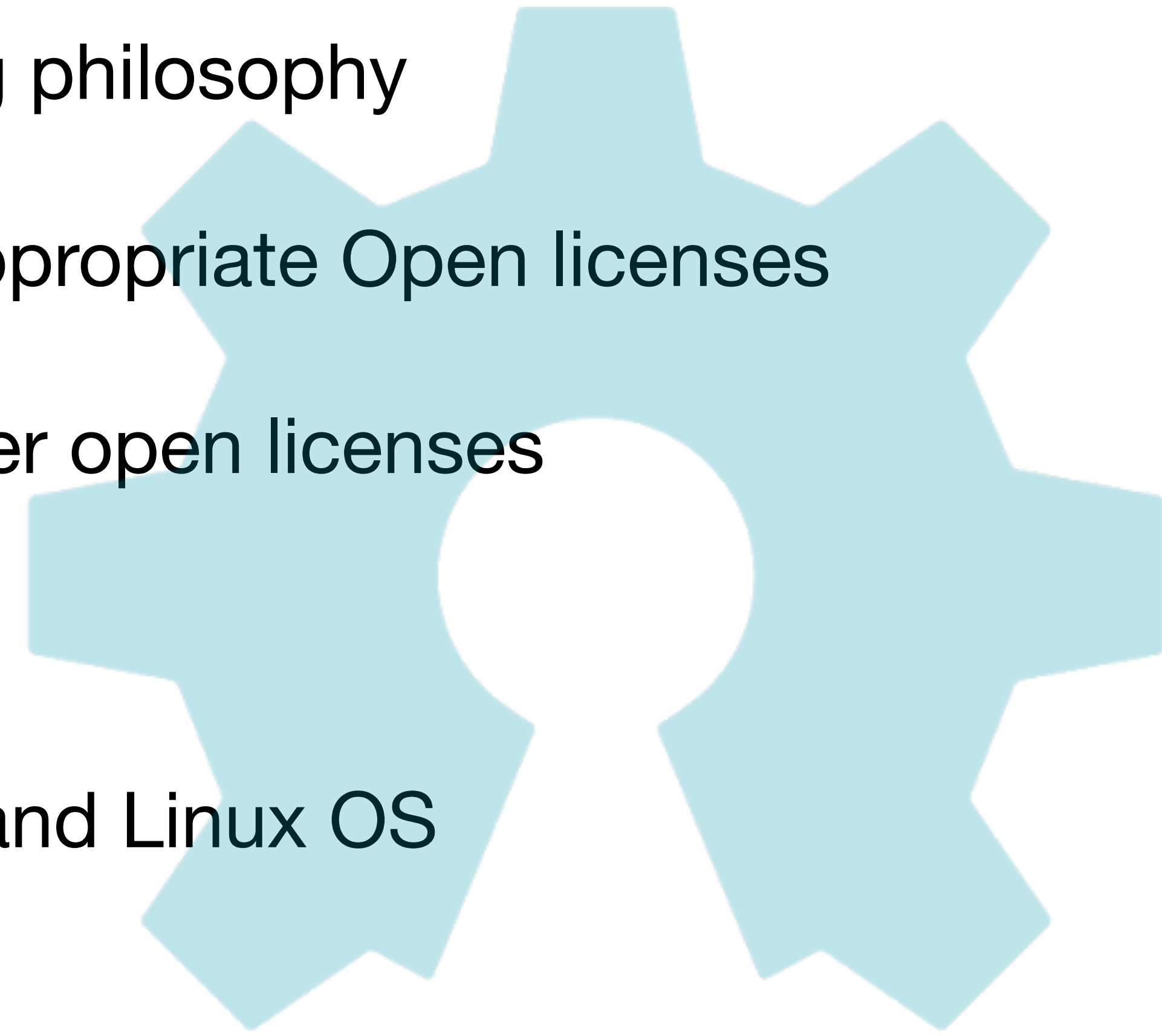
# Platform Approach

- A CPU is no use by itself.
- J-Core SoCs can be generated automatically
- Full set of standard peripherals
  - From memory and cache controllers to Ethernet to LCD drivers
- Supported by the toolchains and OS. Out of the box Linux capable devices

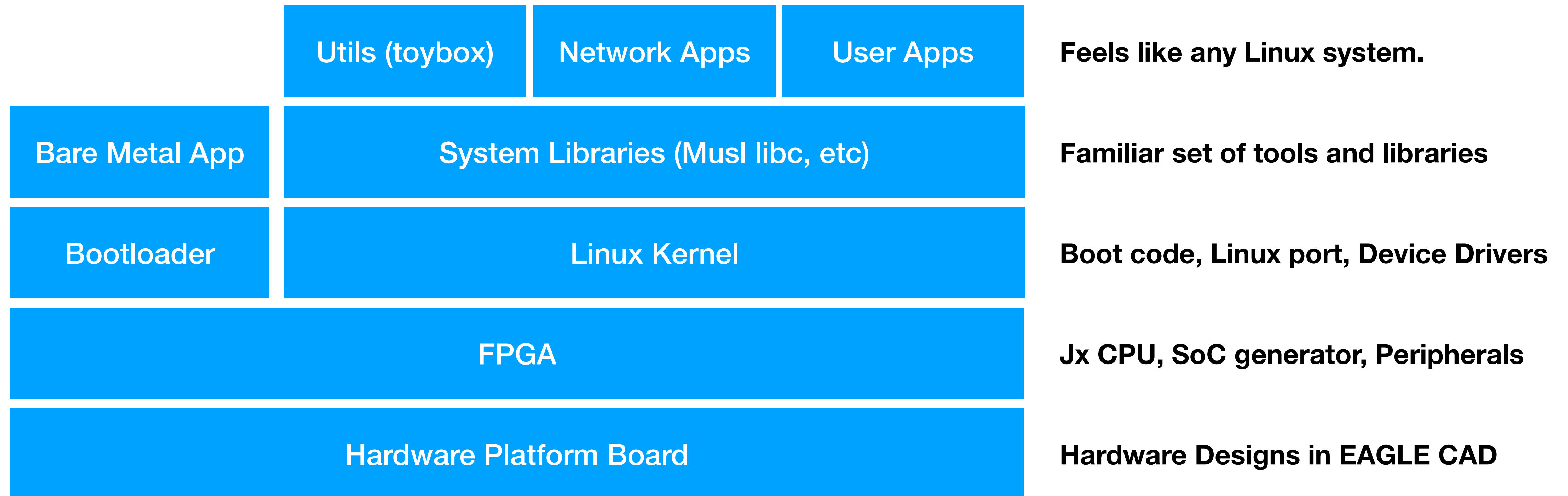


# Open Source and Open Hardware

- Completely Open implementations and driving philosophy
  - J-Core RTL and tools are available under appropriate Open licenses
- Hardware platforms to run it are available under open licenses
  - More than just a reference design
- Complete, mature toolchains for 'bare metal' and Linux OS
- Boot code, Linux OS, Application examples...



# RTL, HW and SW Complete

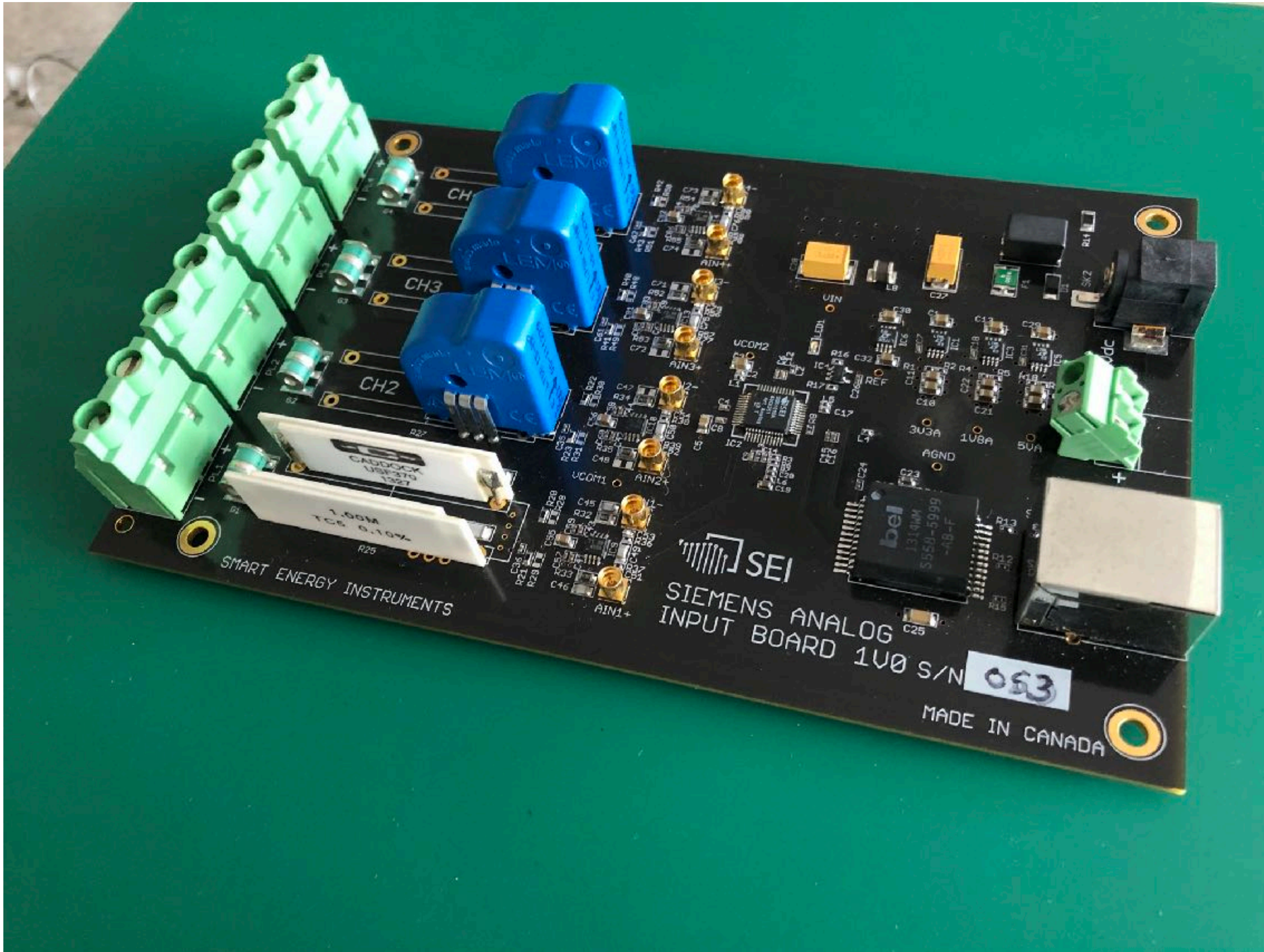




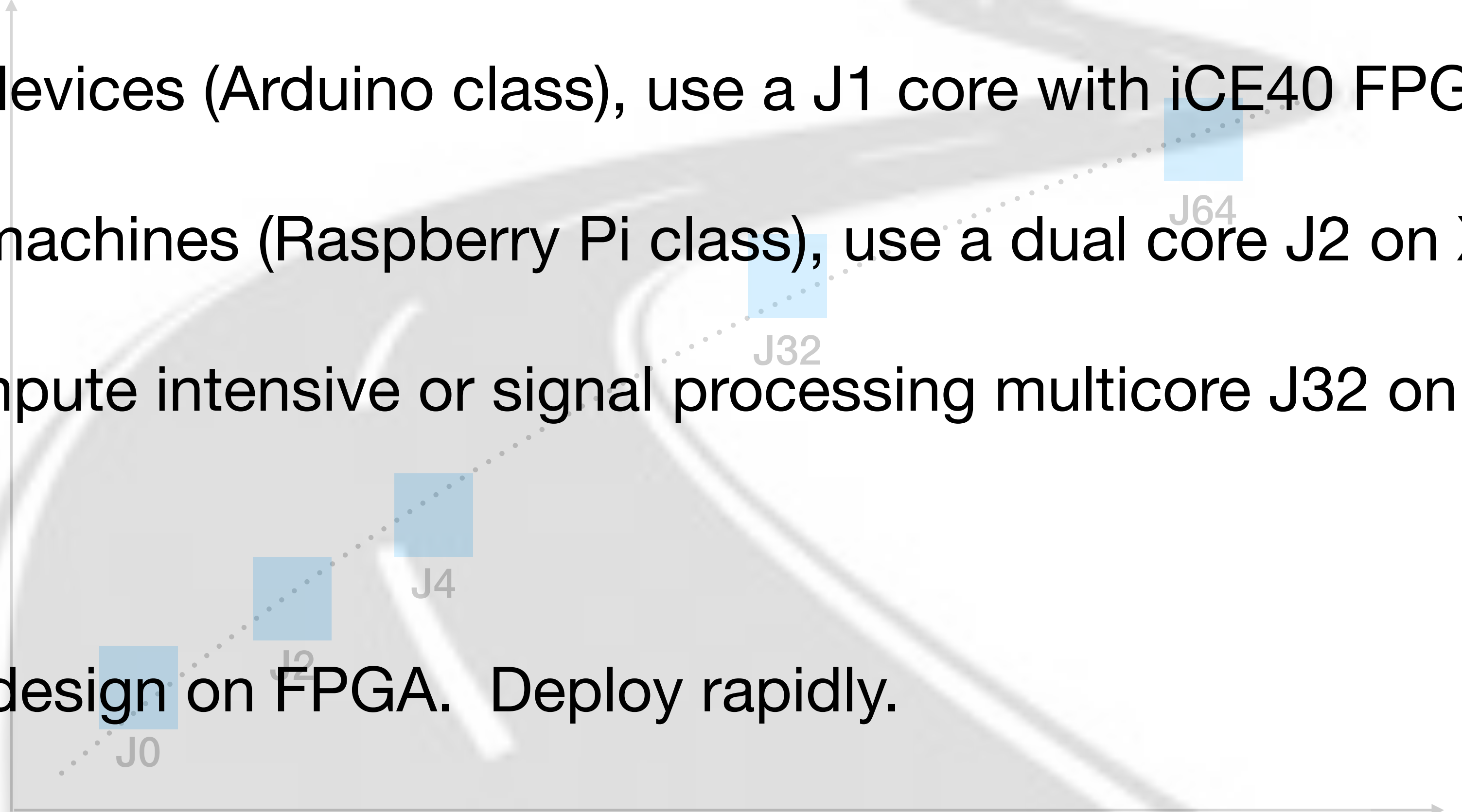
**Break for Questions**

dreamstime®

# 2. What Can You Do With This?



# Roadmap (again)

- For small devices (Arduino class), use a J1 core with iCE40 FPGA
  - For small machines (Raspberry Pi class), use a dual core J2 on Xilinx S6
  - Large, compute intensive or signal processing multicore J32 on Xilinx K7
  - Prove the design on FPGA. Deploy rapidly.
  - Scale the deployment in ASIC, anything 180nm or newer.
- 

# Application Spaces

- Small
  - On board multithreaded, DMA, etc. Compact cores
- Low Power
  - Tiny gate count.  $\sim 100\mu\text{A}$  standby current,  $3\text{mA}$  burst running
- Little Machine
  - Sub \$60US low quantity, including the FPGA. All peripherals
- Compute intensive / signal processing
  - Dedicated co-processors, Application specific accelerators...

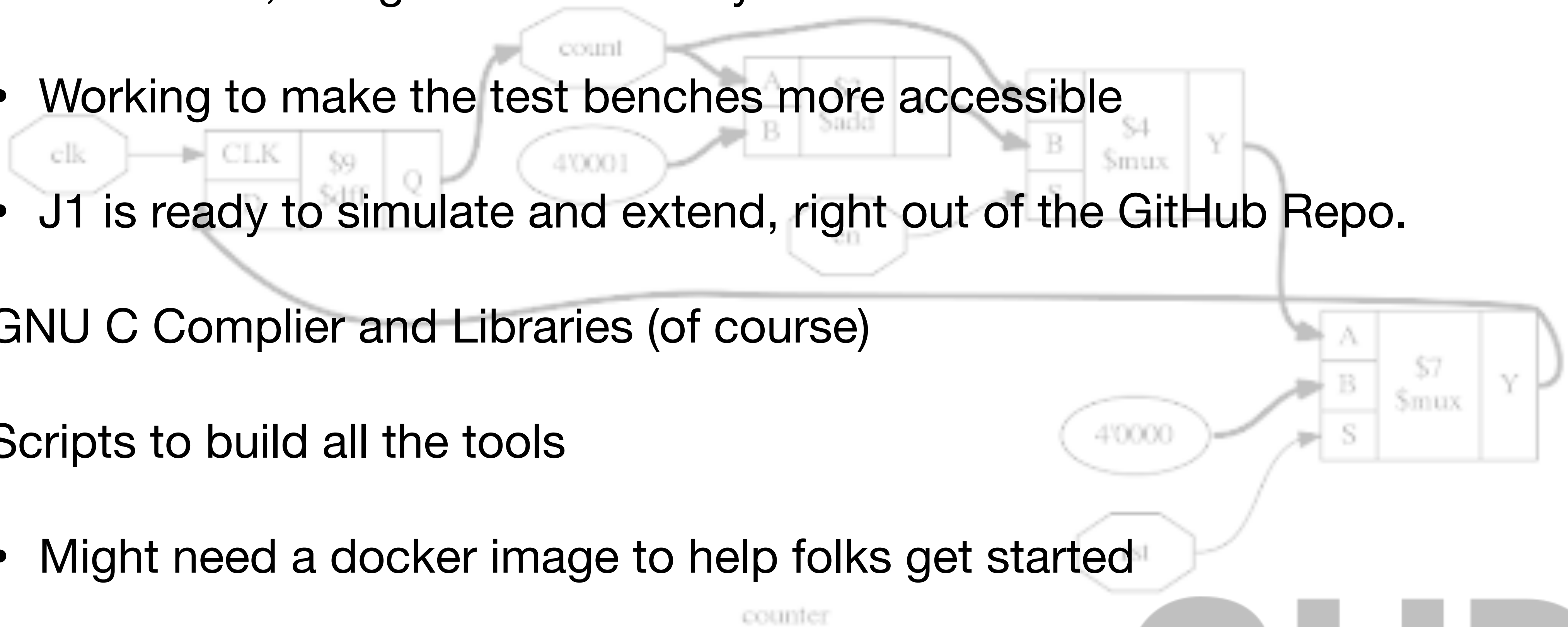


# Dev Boards and Platforms

- Working on making boards available for others to use.
- Modules for COTS use. Simple circuitry on the base board.
- Turtle Board represents a sort of 'ideal' J-Core Embedded device platform
- 3rd party small boards like the UPduino 2.0 useful also

# Tools and Components

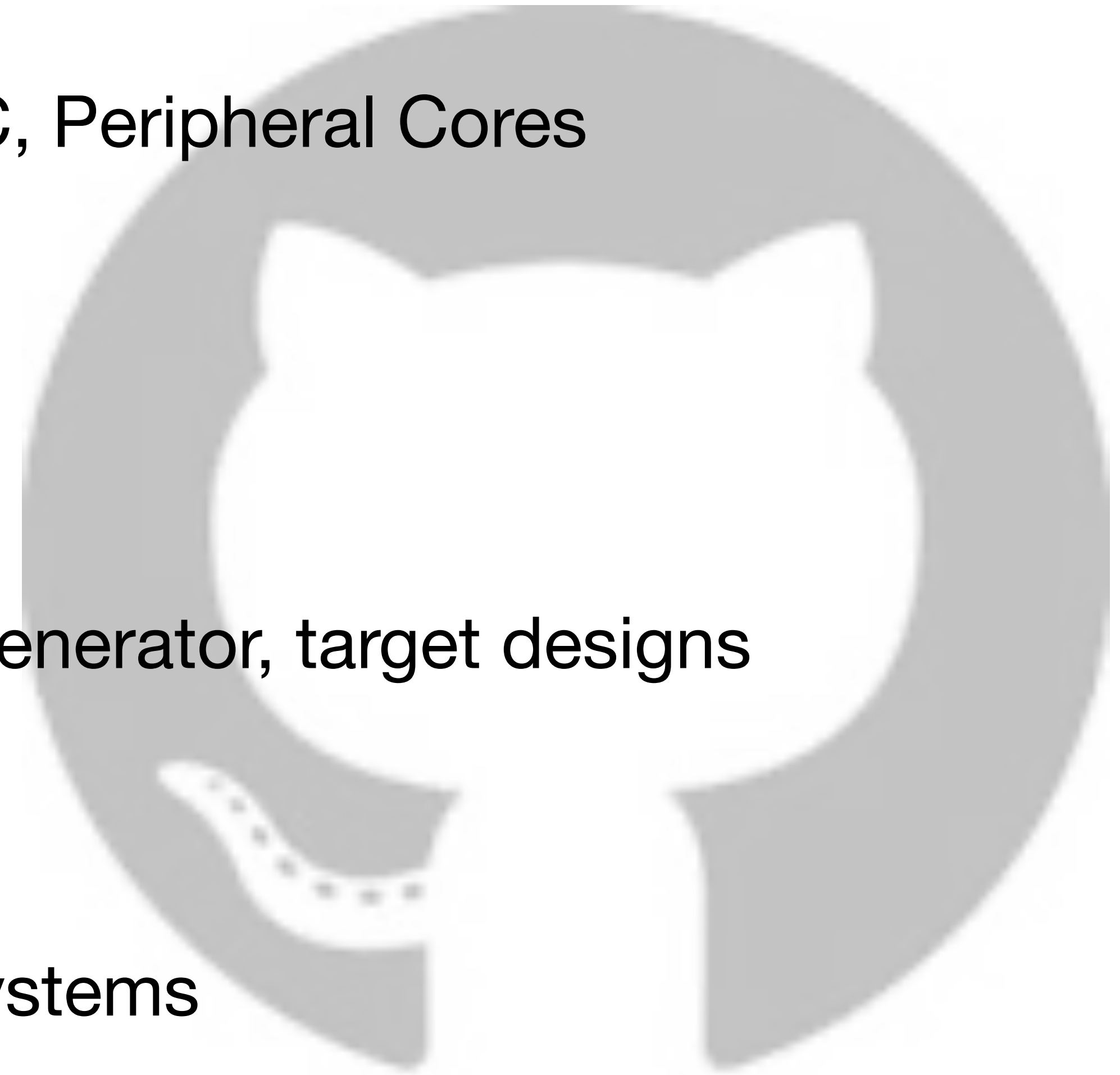
- Simulate first, using GHDL and Yosys.
- Working to make the test benches more accessible
- J1 is ready to simulate and extend, right out of the GitHub Repo.
- GNU C Compiler and Libraries (of course)
- Scripts to build all the tools
  - Might need a docker image to help folks get started
- GHDL+Yosys+NextPNR go all the way to FPGA for Lattice



GHDL

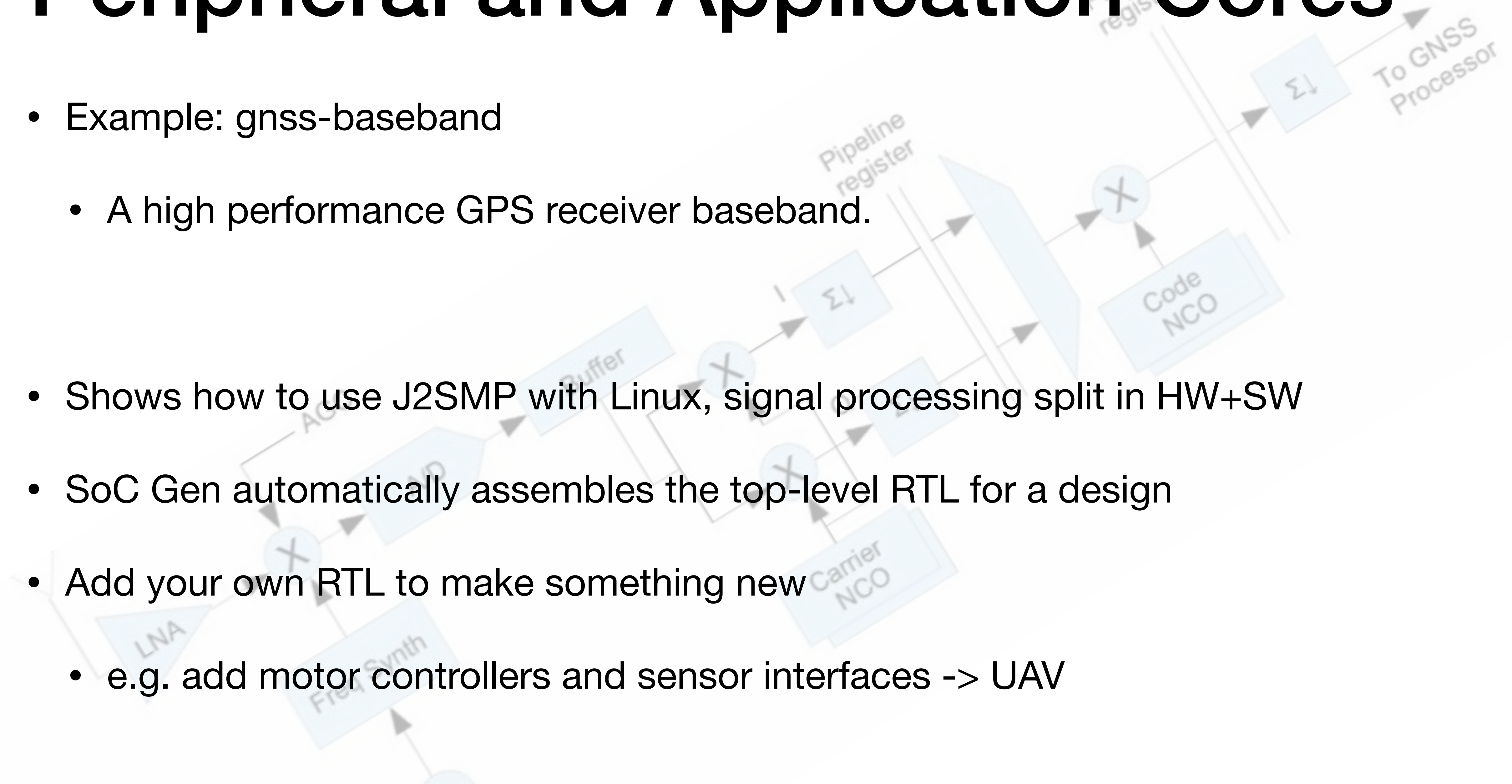
# GitHub Projects

- Organized into a hierarchy: Board, CPU, SoC, Peripheral Cores
- jcore-jx Turtle platform board
- jcore-cpu J2/J32 CPU core
- jcore-soc standard SoC components, SoC generator, target designs
- jcore-j1-ghdl tiny J1 design for single chip systems



# Peripheral and Application Cores

- Example: gnss-baseband
  - A high performance GPS receiver baseband.
- Shows how to use J2SMP with Linux, signal processing split in HW+SW
- SoC Gen automatically assembles the top-level RTL for a design
- Add your own RTL to make something new
  - e.g. add motor controllers and sensor interfaces -> UAV



# Scaling

- Choose the right platform to start with, migrate FPGAs as needed
- Working to provide SoC base design templates for each 'class'
- Off the shelf boards to support each class of device, ASIC path
- J-Core CPUs are upward code compatible
  - Working on a J1 single chip device similar to a full multicore J32

# Getting Started



- When Hitachi did this, only a huge company could (ARM notwithstanding)
- Small boards are available: UPduino2.0 is a good example
- Install the tools: This is where we need some help and beta testers
  - Another talk today is about the Yosys tool flow
- Template RTL can be modified to do cool things
  - More than just a 'soft core' Arduino, add your own logic
- Now, it's as simple as developing software

# Turtle Platform: Linux SoC

- Xilinx Spartan 6 LX45 FPGA.
  - Everything in a soft core. J2SMP CPU at 63MHz.
- Standard Raspberry Pi Peripherals
  - Some core a WIP, e.g. USB Host. Contribute!
- Mature design: 2017, with stable and standard SoC Gen port of jcore-soc
- OpenHardware. Use the circuit design or even layout for your own project

# Open 42s Board: Tiny system

- An example of a battery operated low power device.
- Runs a J1 with the computationally intensive multi precision Free42 app
- Just 2 chips: Lattice FPGA plus SPI FLASH for FPGA bitstream and code
  - Just 77uA standby current, 1-2mA on keystroke. 6mo battery life
- UI/UX example. CNC controller?



# GPS Hat: Signal processing

- Example signal processing application
  - 1.542GHz receiver for GPS signals. Produces digital IF at 16.368Ms/s
  - No on board processor! Uses the Turtle Board and J2SMP
- Baseband receiver RTL shows how to integrate complex functions on SoC
  - Hardware core does the high speed processing, J-Core solves fix
- Open design: Add it to your own board.



# Questions and Wrap Up



# Thanks.

Please check out our GitHub  
<http://github.com/j-core>