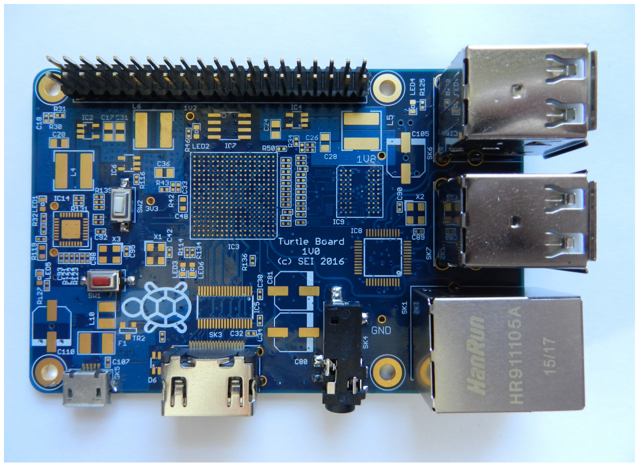
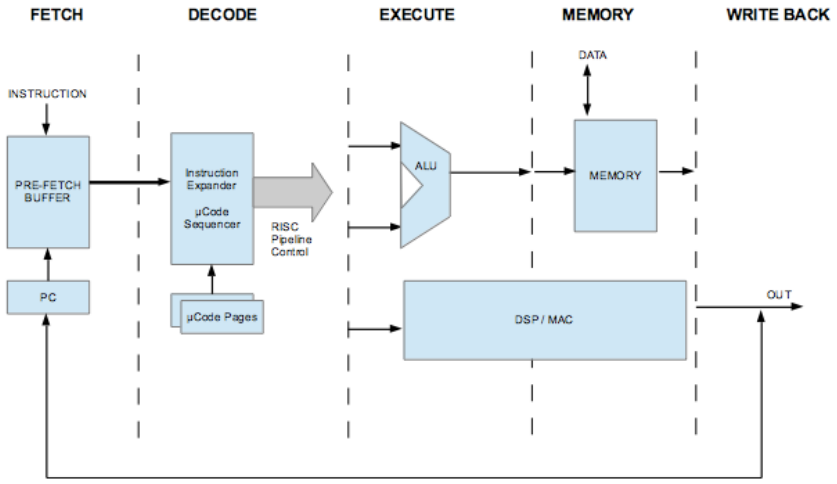
j-core Design Walkthrough





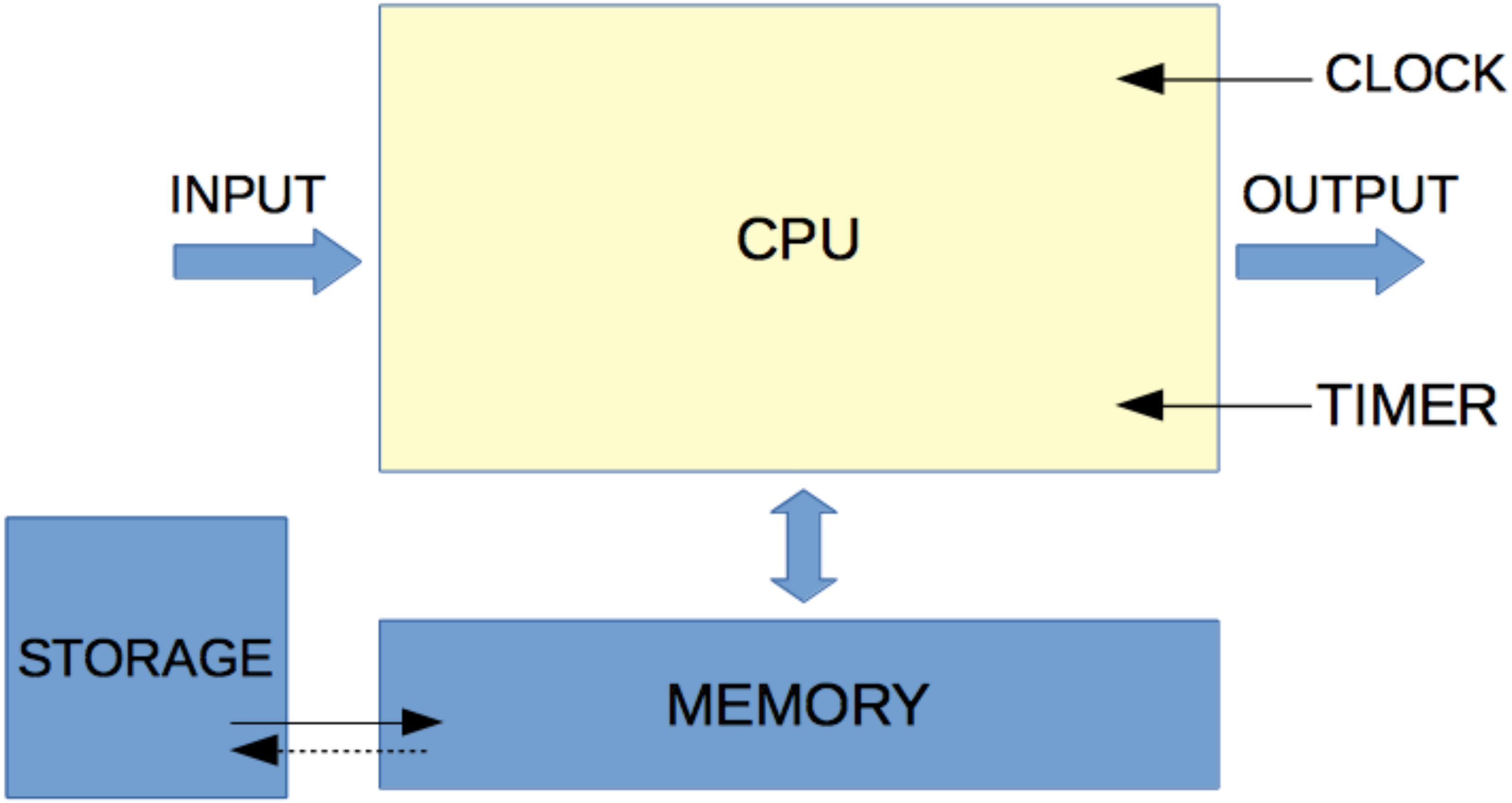
Q: What parts of Linux Systems are Open and Under your Control?

- Modern laptops have more arm/mips processors than x86
  - USB controller? Exploitable:
    http://arstechnica.com/security/2014/07/this-thumbdrive-hacks-computers-badusb-exploit-makes-devices-turn-evil/
  - Hard drive controller? Exploitable:
    http://hackaday.com/2013/08/02/sprite_tm-ohm2013-talk-hacking-hard-drive-controller-chips/
- System management mode, ACPI…
- Do you trust your <big vendor's> RNG ?

Taking Back Control of HW and SW

open source hardware

What is the minimum system that can run Linux?



The definition of a 'CPU' (for the purposes of this talk)

- Has a Flat 32bit memory space
  - All pointers 'just work' (no separate spaces)
- A port of GCC
  - Regular, efficient 32-16-8 bit int-short-char
- Executes instructions 'fast enough'
  - Servicing streaming Ethernet traffic: about 30MIPs

System Requirements

- CPU (of course)
- Memory
  - 8 megs RAM runs practical Linux workloads
  - less is possible, but awkward
- Storage (load kernel+initramfs into memory)
- Some form of I/O
  - Usually, and at first, just a UART
- A Timer (interrupt).

Things you don’t need

- Anything else, actually
  - Video, audio, keyboard, persistent storage
- An MMU, or any ‘fancy’ CPU features.
  - FPU, SMP, even cache is optional

A History Lesson

μClinux

The Linux/Microcontroller project is a port of Linux to systems without a Memory Management Unit (MMU).

uClinux first ported to the Motorola MC68328: DragonBall Integrated Microprocessor. The first target system to successfully boot is the PalmPilot using a TRG SuperPilot Board with a custom boot-loader created specifically for our Linux/PalmPilot port.

(January 1998)

What did we do?

- New SH2 instruction set compatible core design kit
  - called "j2" because trademarks haven't expired.
  - clean-room implementation, initially by Canadian engineers
    - built for design reuse
    - Then hired original SuperH team members to work on it afterwards
  - Source is VHDL using programming model developed by European Space Agency
    - Verilog is low level like assembly, VHDL is a HLL
    - Design Kit contains high level abstractions for future cores

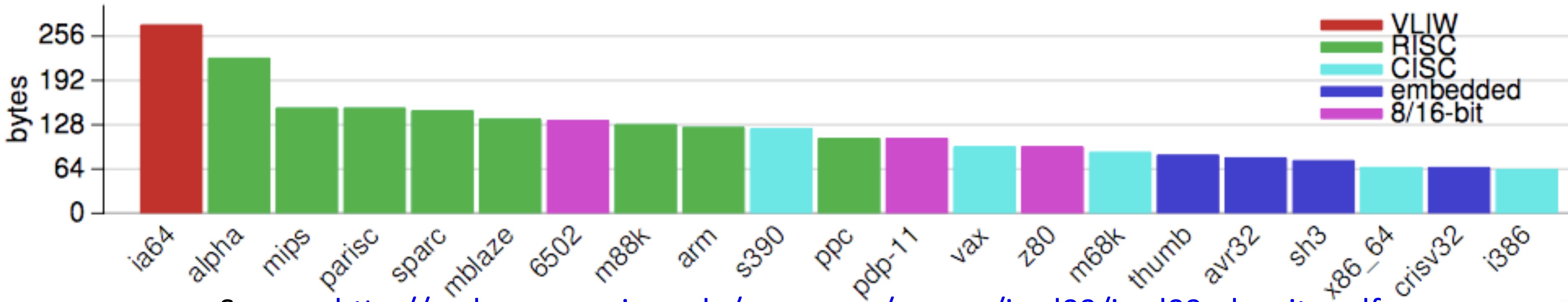
- SOC builder system
  - Links together peripherals automatically
    - E.g. Serial, mmc, Ethernet
    - Can produce FPGA bitstream, ASIC RTL, C emulator source
    - SMP capable but not finished yet

Why recreate existing architecture?

- Tools exist: compiler, kernel, debugger, strace...
- Leverage massive R&D outlay in SuperH
    - 5 stage RISC (full Harvard architecture)
    - instruction set density (16 bit fixed length)
    - simple highly optimized design
    - original designers/implementers still around
- Old chips are prior art vs. "breathing is patented”

Code Density : Efficiency

Fig. 2. Total size of benchmarks (includes some platform-specific code, so does not strictly reflect code density)



Source: http://web.eece.maine.edu/~vweaver/papers/iccd09/iccd09_density.pdf

- There *Are* other metrics, bit none actually matter more (unless something is broken)

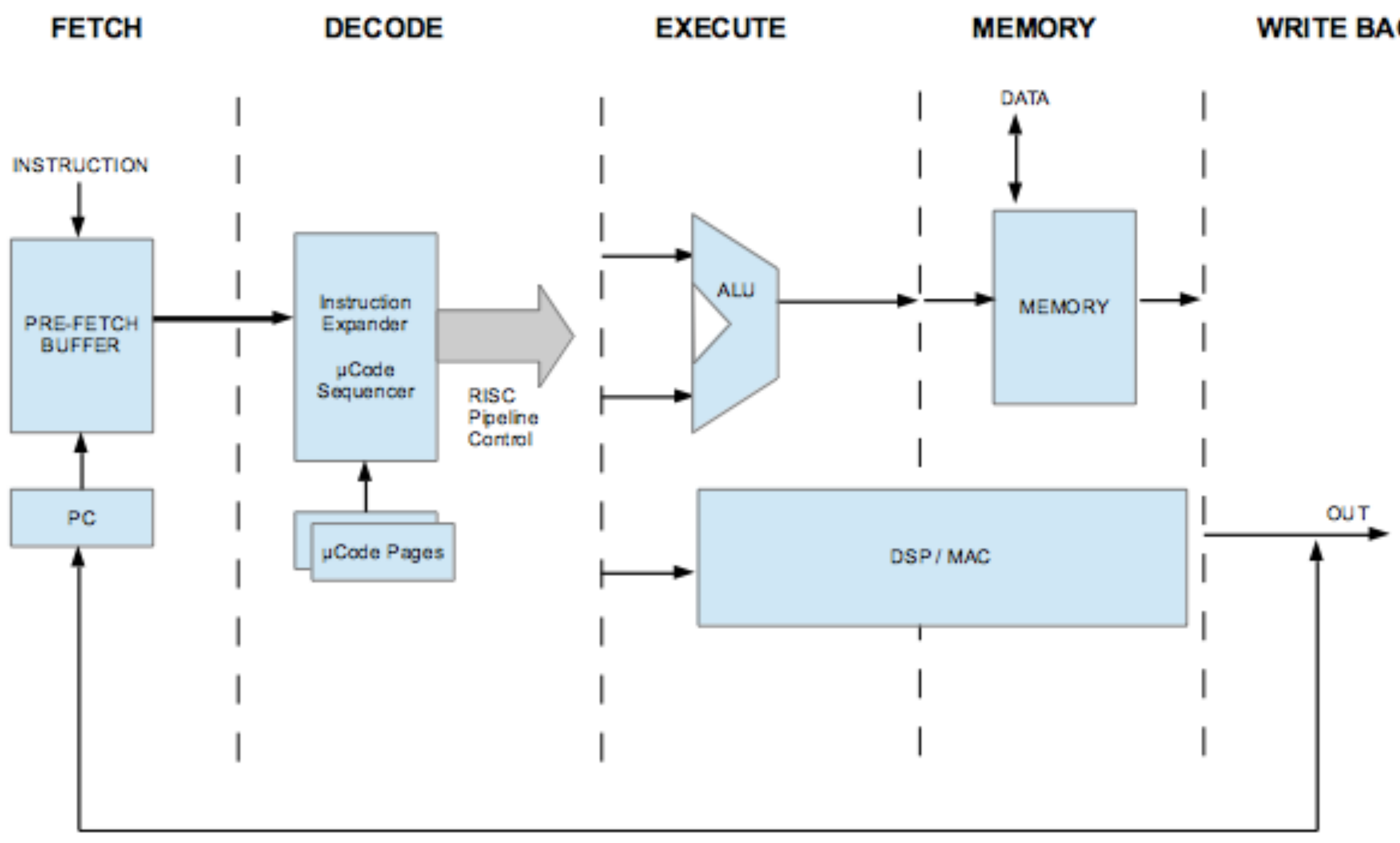
  - Note: ARM paid millions to Hitachi to use SuperH patents in Thumb instruction set… which just expired.

Patent Expiration

- SuperH ISA had a huge effort put into it
  - See above about efficient GCC.
- The SuperH architecture was developed by Hitachi a quarter century ago.
- Last patent on SH2 (Sega Saturn) expired in October 2014.
  - That's why we can release this now
  - last SH4 (Dreamcast) patent expires in 2016.
- SuperH ISA was the blueprint for ARM Thumb

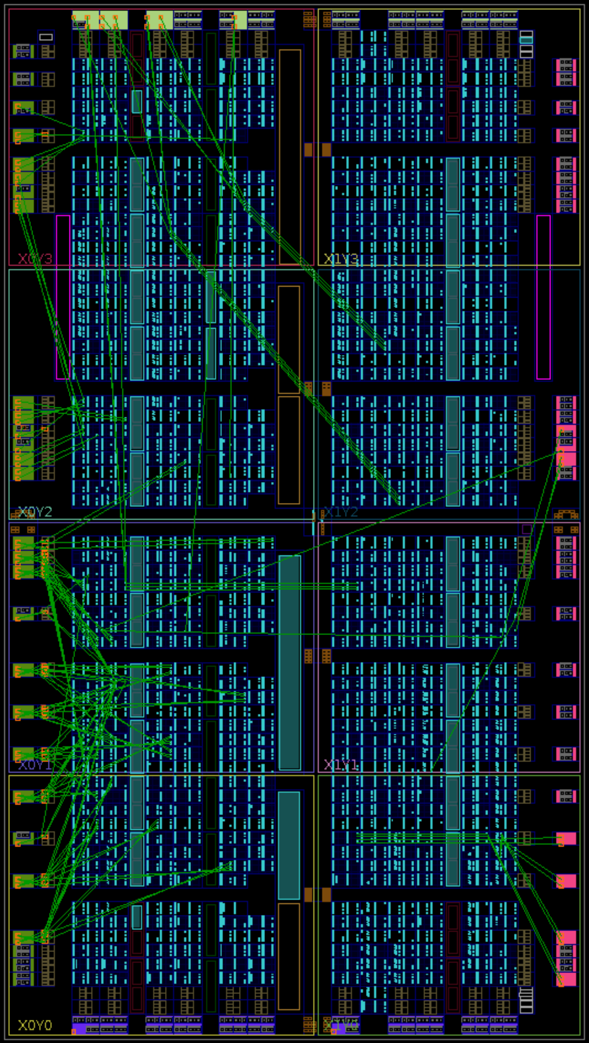
The basic SuperH design:

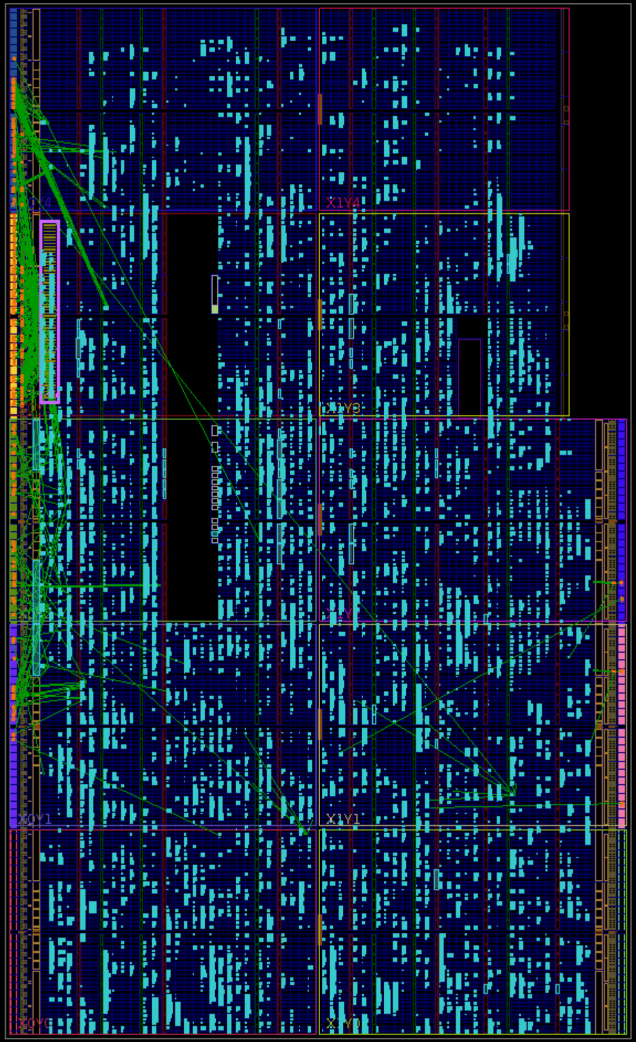
5 stage 'classical' RISC pipeline, with some additions



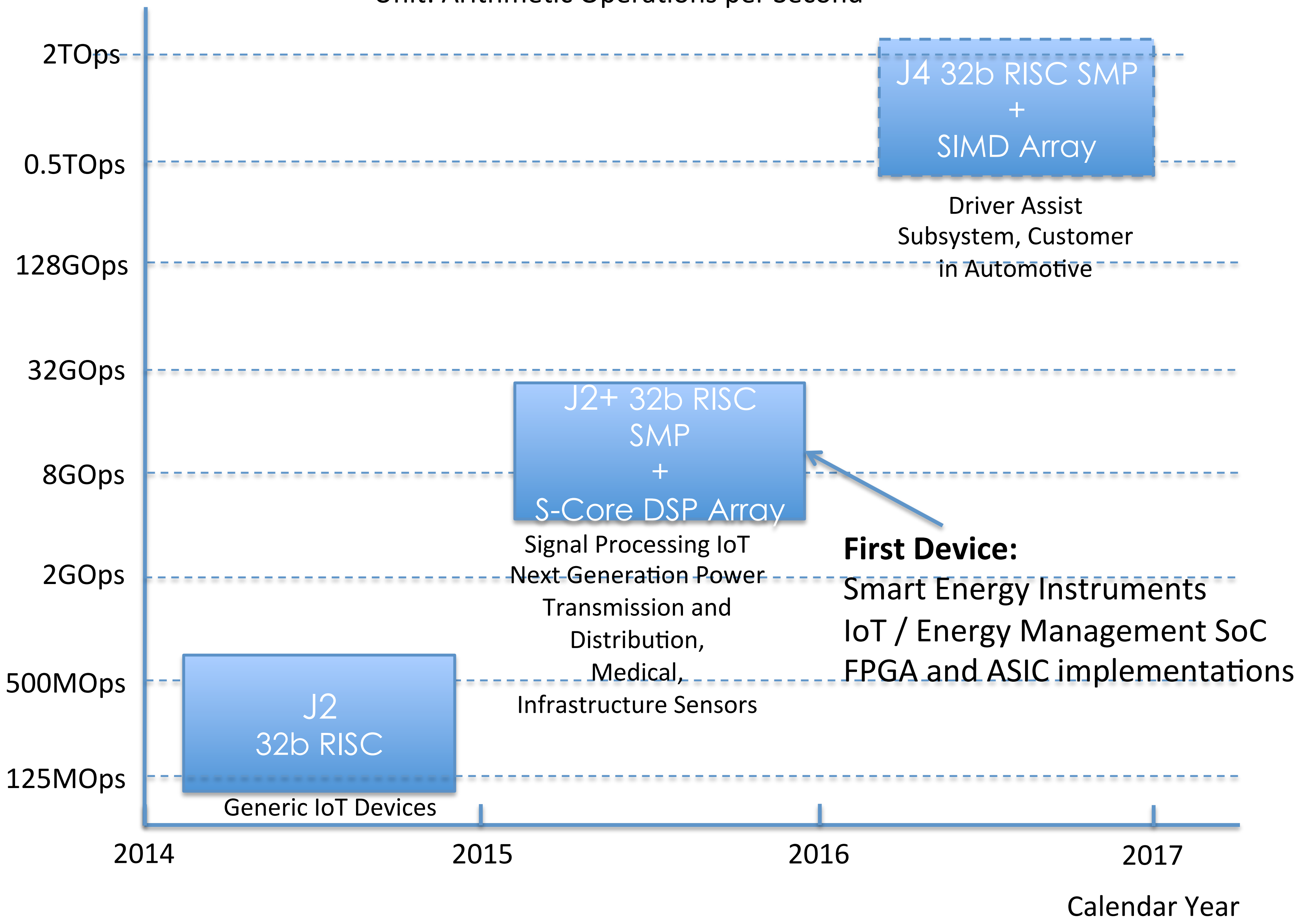
Pretty simple to implement, except the Instruction Decoder / Expander

FPGA floor plans: Spartan 6 & Kintex 7



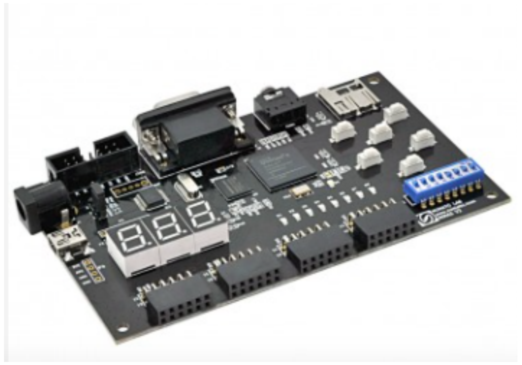


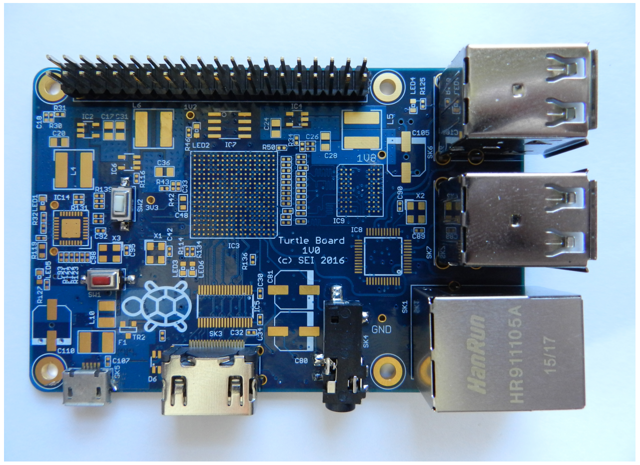
J Series Computation Core Cluster Roadmap



Demonstration Platforms

- Our j2 processor core can run linux on low-cost FPGA Spartan6 platforms such as Numato Labs' Mimas2



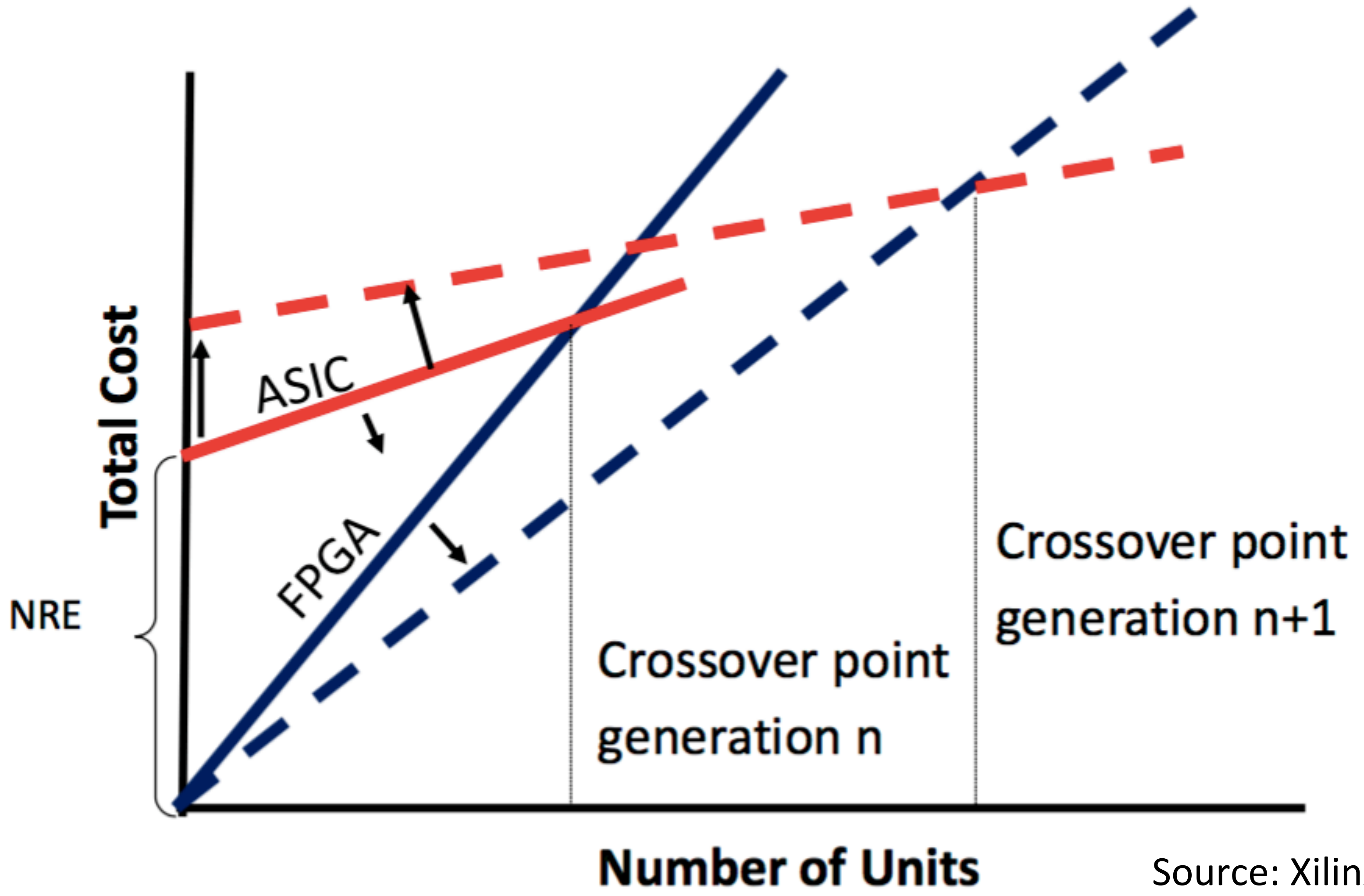


- We are launching a custom development board (Turtle Board) with the same form factor as Raspberry Pi
- (Please support out Kickstarter)

So, how do you use it for anything?

- VHDL and build system are under a BSD license
- Just use an FPGA: $15 gets you UP, $40 SMP + DSP
  - Plus any peripheral your application requires
- Make any chip you want
  - Royalty free
    - 180nm ASIC of SOC we're demoing costs <10 yen
      - Processor only, about 2 and a half cents
    - Disposable computing at "free toy inside" level
    - Think IoT : 'Trillion Sensor Network' economics, but running Linux
- j-core.org, nommu.org (uclinux-ng),
- Source, documentation, tutorials, mailing lists
    - We assume you've never done hardware before.
  - Still a bit sparse but filling sites out as we go
    - Patches welcome. No question too stupid.

Practical Hardware: FPGAs and ASICs



Source: Xilinx 2014

How: VHDL, not Verilog

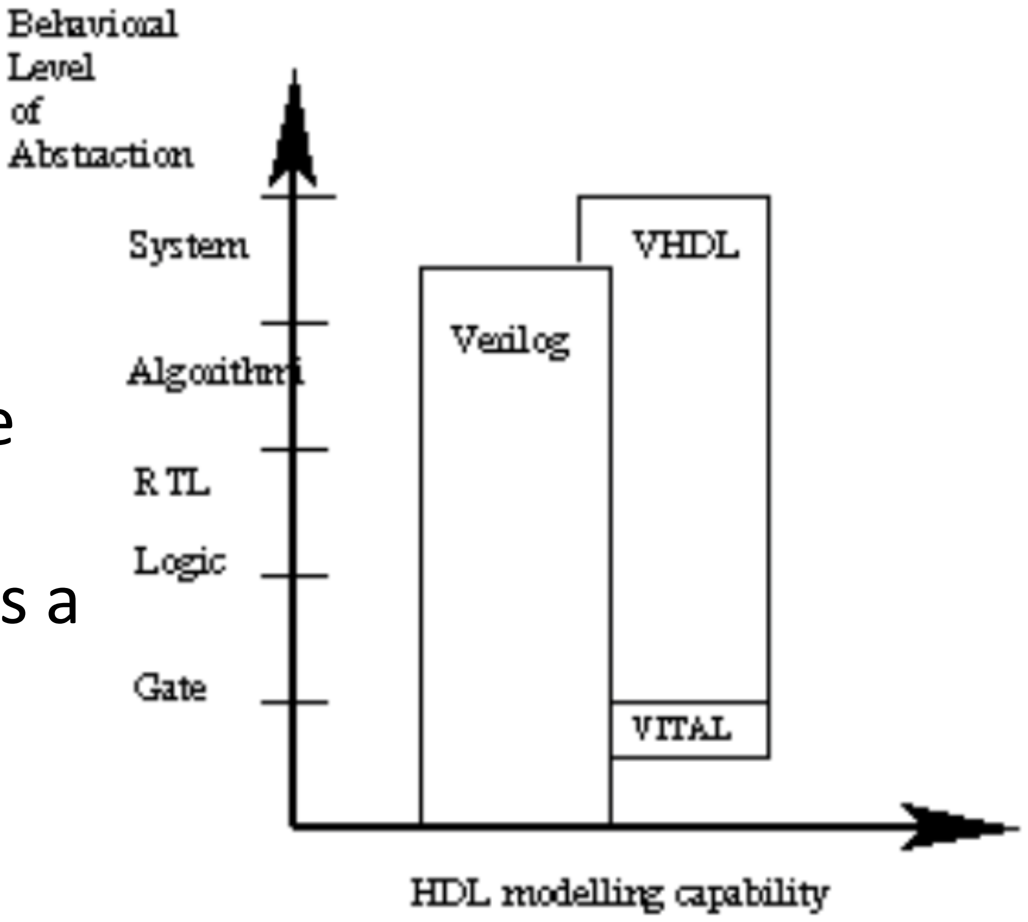
Although Verilog is more commonly used in certain geographies..

- VHDL is the preferred language in developments initiated or led by the European Space Agency.
- The VHSIC Hardware Description Language (VHDL) is a formal notation intended for use in all phases of the creation of electronic systems. Because it is both machine readable and human readable, it supports the development, verification, synthesis, and testing of hardware designs, the communication of hardware design data, and the maintenance, modification, and procurement of hardware.

These 2 things are not the same…

- Actually…

- It’s about the type system.
- Verilog don’t have one (to speak of)
- In VHDL, everything* is a derived type.
- Even + is ‘just’ and overloaded operator.



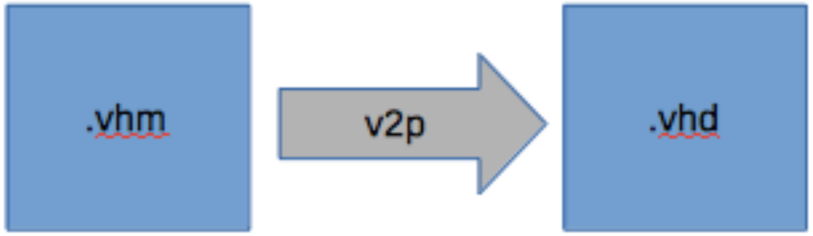
Going further : Structured VHDL Design Method

- In order to overcome the limitations of the classical 'dataflow' design style (large number of concurrent VHDL statements and processes, leading to bad readability and increased simulation time), a **'two-process'** coding method is proposed: one process contains all combinational logic, whereas the other process infers all (and only) the registers.
- The use of record types to increase readability and the safe use of variables to reduce simulation time. The method has been applied on several designs made by or for ESA.

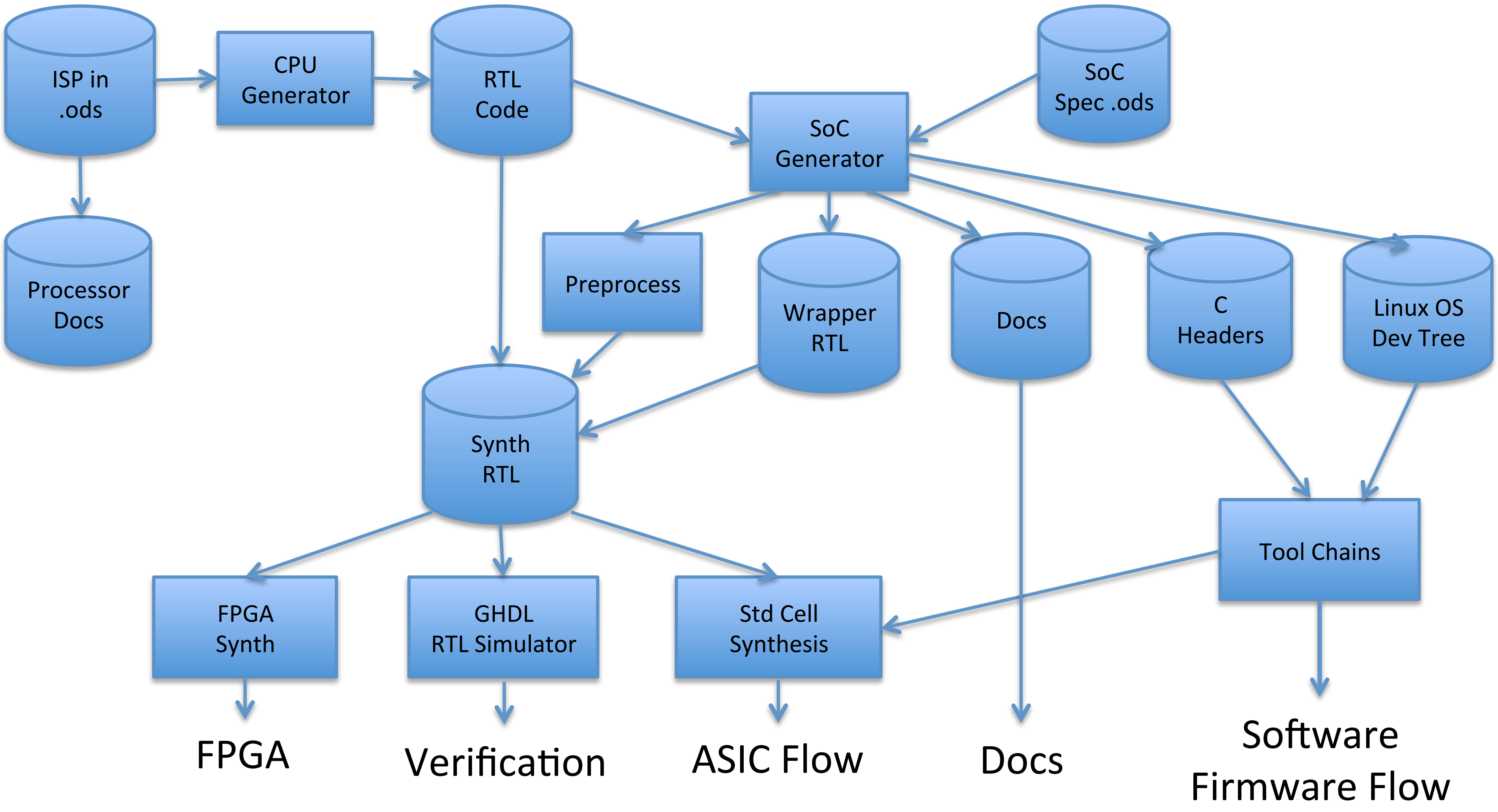
-Jiri Gaisler, **http://www.gaisler.com/doc/vhdl2proc.pdf**

Avoiding Common Errors

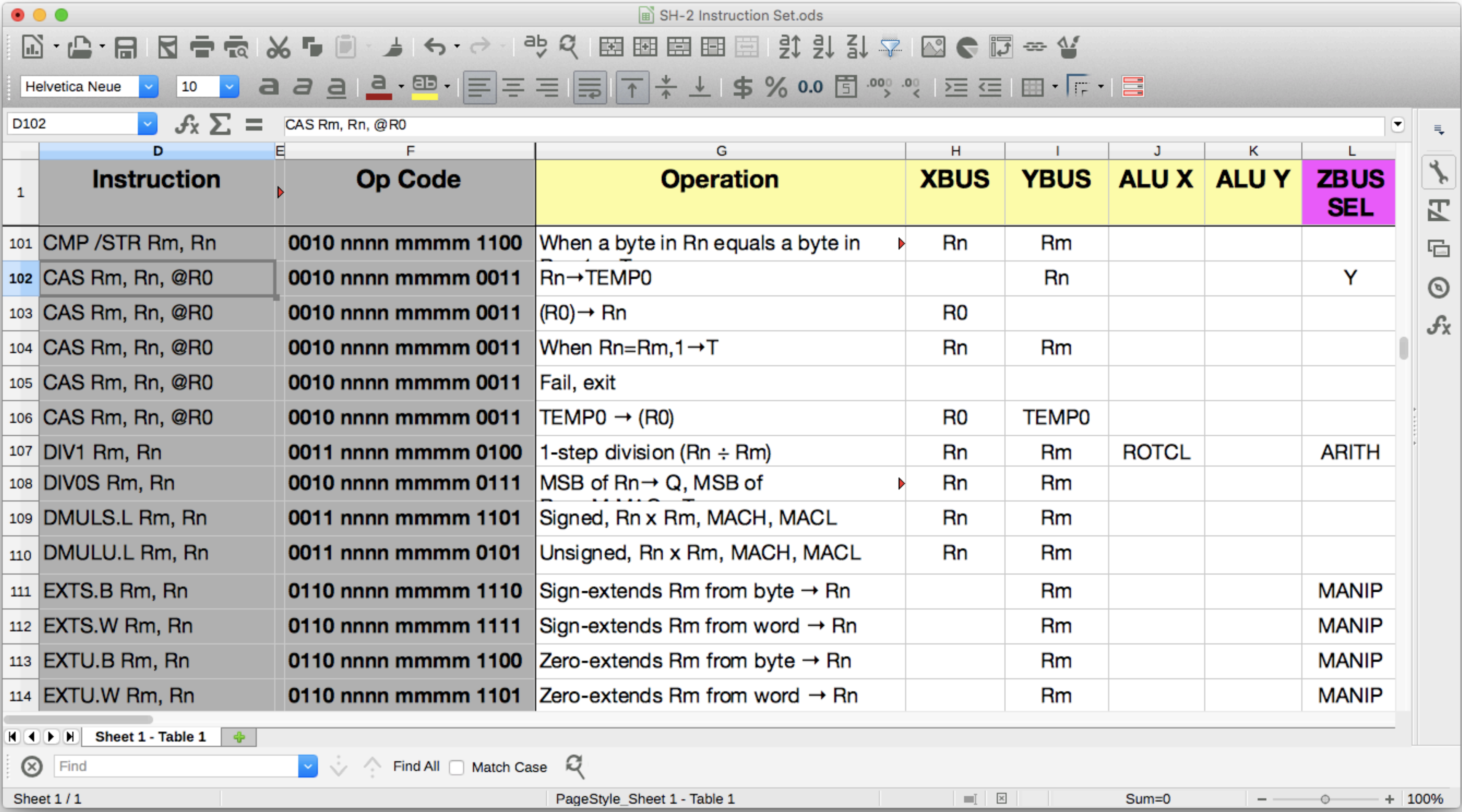
- We developed a pre-processor perl tool (v2p) to avoid latches, and other similar coding errors .
  - The `.vhm` file is a dialect of vhdl; sensitivity lists are generated by the perl script
- This has resulted in highly reliable RTL, and greatly increased the efficiency of our internal development process



Even Further : Automated ISA -> RTL Sim->FPGA->ASIC->SW Tools Flow



ISA Generator



SH-2 Instruction Set.ods

D102 CAS Rm, Rn, @R0

	Instruction	Op Code	Operation	XBUS	YBUS	ALU X	ALU Y	ZBUS SEL
1	Instruction	Op Code	Operation	XBUS	YBUS	ALU X	ALU Y	ZBUS SEL
101	CMP /STR Rm, Rn	0010 nnnn mmmm 1100	When a byte in Rn equals a byte in	Rn	Rm			
102	CAS Rm, Rn, @R0	0010 nnnn mmmm 0011	Rn→TEMP0		Rn			Y
103	CAS Rm, Rn, @R0	0010 nnnn mmmm 0011	(R0)→ Rn	R0				
104	CAS Rm, Rn, @R0	0010 nnnn mmmm 0011	When Rn=Rm,1→T	Rn	Rm			
105	CAS Rm, Rn, @R0	0010 nnnn mmmm 0011	Fail, exit					
106	CAS Rm, Rn, @R0	0010 nnnn mmmm 0011	TEMP0 → (R0)	R0	TEMP0			
107	DIV1 Rm, Rn	0011 nnnn mmmm 0100	1-step division (Rn ÷ Rm)	Rn	Rm	ROTCL		ARITH
108	DIV0S Rm, Rn	0010 nnnn mmmm 0111	MSB of Rn→ Q, MSB of	Rn	Rm			
109	DMULS.L Rm, Rn	0011 nnnn mmmm 1101	Signed, Rn x Rm, MACH, MACL	Rn	Rm			
110	DMULU.L Rm, Rn	0011 nnnn mmmm 0101	Unsigned, Rn x Rm, MACH, MACL	Rn	Rm			
111	EXTS.B Rm, Rn	0110 nnnn mmmm 1110	Sign-extends Rm from byte → Rn		Rm			MANIP
112	EXTS.W Rm, Rn	0110 nnnn mmmm 1111	Sign-extends Rm from word → Rn		Rm			MANIP
113	EXTU.B Rm, Rn	0110 nnnn mmmm 1100	Zero-extends Rm from byte → Rn		Rm			MANIP
114	EXTU.W Rm, Rn	0110 nnnn mmmm 1101	Zero-extends Rm from word → Rn		Rm			MANIP

Sheet 1 - Table 1

Sheet 1 / 1 PageStyle_Sheet 1 - Table 1 Sum=0 100%

What can you do with this now?

- Download bitstream + vmlinux/initramfs, install on fpga board, boot kernel to shell prompt
- HOWTOs with background info
  - Where to order FPGA board(s)
  - Download and install xilinx/digilent tools
    - Free download for linux/mac, but alas no open source bitstream compiler yet. OpenOCD installer is todo item.
  - Build new bitstream from source
  - Program nommu Linux (gcc/binutils/musl toolchain)

Codewalk Through

- Simple VHDL Example
  - Simulation
  - Synthesis example
- SoC Generator Input
  - Peripherals and busses
- CPU Core code organization
  - Pipeline and ISA decoder generator
- CPU RTL Simulation
  - Run the RTL without hardware

J-Core.org

- Watch for a Git Repository and a mailing list:

J-Core@J-Core.org

http://Lists.J-Core.org