The Libre-SOC Hybrid 3D CPU

Augmenting the OpenPOWER ISA
to provide 3D and Video instructions
(properly and officially) and make a GPU

FOSDEM2021

Sponsored by NLnet's PET Programme

February 3, 2021

Why another SoC?

- Intel Management Engine, Apple QA issues, Spectre
- Endless proprietary drivers, "simplest" solution:
  License proprietary hard macros (with proprietary firmware)
  Adversely affects product development cost
  due to opaque driver bugs (Samsung S3C6410 / S5P100)
- Alternative: Intel and Valve-Steam collaboration
  "Most productive business meeting ever!"
  https://tinyurl.com/valve-steam-intel
- Because for 30 years I Always Wanted To Design A CPU
- Ultimately it is a strategic *business* objective to develop
  entirely Libre hardware, firmware and drivers.

Why OpenPOWER?

- Good ecosystem essential
  linux kernel, u-boot, compilers, OSes,
  Reference Implementation(s)
- Supportive Foundation and Members
  need to be able to submit ISA augmentations
  (for proper peer review)
- No NDAs, full transparency must be acceptable
  due to being funded under NLnet's PET Programme
- OpenPOWER: established for decades, excellent Foundation,
  Microwatt as Reference, approachable and friendly.

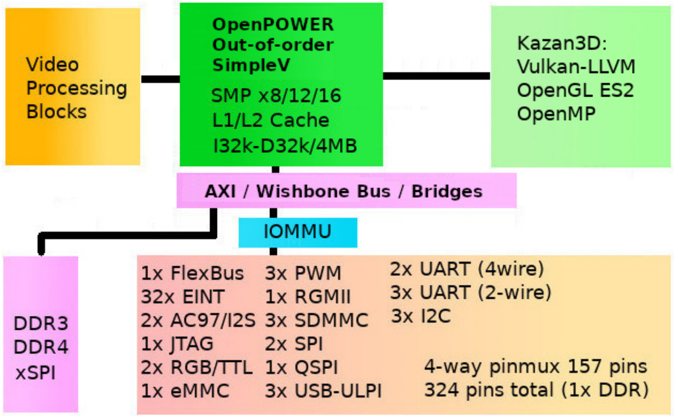
How can you help?

- Start here! https://libre-soc.org
  Mailing lists https://lists.libre-soc.org
  IRC Freenode libre-soc
  etc. etc. (it's a Libre project, go figure)
- Can I get paid? Yes! NLnet funded
  See https://libre-soc.org/nlnet/#faq
- Also profit-sharing in any commercial ventures
- How many opportunities to develop Libre SoCs exist, and actually get paid for it?
- I'm not a developer, how can I help?
  - Plenty of research needed, artwork, website
  - Help find customers and OEMs willing to commit (LOI)

What goes into a typical SoC?

- 15 to 20mm BGA package: 2.5 to 5 watt power consumption heat sink normally not required (simplifies overall design)
- Fully-integrated peripherals (not Northbridge/Southbridge) USB, HDMI, RGB/TTL, SD/MMC, I2C, UART, SPI, GPIO etc. etc.
- Built-in GPU (shared memory bus, 3rd party licensed)
- Built-in VPU (likewise, proprietary)
- Target price between $2.50 and $30 depending on market Radically different from IBM POWER9 Core (200 Watt)
- We're doing the same, just with a hybrid architecture. CPU == GPU == VPU

Simple SBC-style SoC



What's different about Libre-SOC?

- Hybrid - integrated. The CPU *is* the GPU.
  The GPU *is* the CPU. The VPU *is* the CPU.
  There is No Separate VPU/GPU Pipeline or Processor
- written in nmigen (a python-based HDL). Not VHDL
  not Verilog (definitely not Chisel3/Scala)
  This is an extremely important strategic decision.
- Simple-V Vector Extension. See 'SIMD Considered harmful'.
  https://tinyurl.com/simd-considered-harmful
  SV effectively a "hardware for-loop" on standard scalar ISA
  (conceptually similar to Zero-Overhead Loops in DSPs)
- Yes great, but what's different compared to Intel, AMD, NVIDIA, ARM and IBM?

OpenPOWER Cell Processor and upwards

- OpenPOWER ISA developed from PowerPC, with the RS6000 in the 90s.
- Sony, IBM and Toshiba began the Cell Processor in 2001 (Sony Playstation 3) - NUMA approach
- Raw brute-force performance pissed all over the competition at the time
- VSX later evolved out of this initiative.
- VSX, a SIMD extension, now showing its age.
  Fixed-width, no predication, limited pixel formats (15 bit)
- (Vulkan requires dozens of pixel formats)

Apple M1 (ARM) vs Intel / AMD (x86)

- Very interesting article: tinyurl.com/apple-m1-review
- Apple M1: uses ARM. Intel: implements x86
- Apple M1: RISC multi-issue. Intel: CISC multi-issue.
- Apple M1: uniform (easy) instruction decode
  Intel: *Cannot easily identify start of instruction*
- Result: multi-issue x86 decoder is so complex, it misses opportunities to keep back-end execution engines 100 percent occupied
- OpenPOWER happens to be RISC (easy decode), which is why POWER10 has 8-way multi-issue.
- Libre-SOC can do the same tricks that IBM POWER10 and Apple M1 can. Intel (x86) literally cannot keep up.

Hybrid Architecture: Augmented 6600

- CDC 6600 is a design from 1965. The *augmentations* are not. Help from Mitch Alsup includes *precise exceptions*, multi-issue and more. Academic literature on 6600 utterly misleading. 6600 Scoreboards completely underestimated (Seymour Cray and James Thornton solved problems they didn't realise existed elsewhere!)
- Front-end Vector ISA, back-end "Predicated (masked) SIMD" nmigen (python OO) strategically critical to achieving this.
- Out-of-order combined with Simple-V allows scalar operations at the developer end to be turned into SIMD at the back-end *without the developer needing to do SIMD*
- IEEE754 sin / cos / atan2, Texturisation opcodes, YUV2RGB all automatically vectorised.

Learning from these and putting it together

- Apple M1 and IBM POWER10 show that RISC plus superscalar multi-issue produces insane performance
- Intel AVX 512 and CISC in general is getting out of hand (what's next: 256-bit length instructions, AVX 1024?)
- RISC-V RVV shows Cray-style Vectors can save power. Simple-V has the same benefits with far less instructions (188 for RVV, 3 to 5 new instructions for Simple-V).
- CDC 6600 shows that intelligently-implemented designs can do the job, with far less resources.
- Libre-SOC combines the best of historical processor designs, co-opting and innovating on them (pissing in the back yard of every incumbent CPU and GPU company in the process).
- It's a Libre project: you get to help

Why nmigen?

- Uses python to build an AST (Abstract Syntax Tree). Actually hands that over to yosys (to create ILANG file) after which verilog can (if necessary) be created
- Deterministic synthesiseable behaviour (Signals are declared with their reset pattern: no more forgetting "if rst" block).
- python OO programming techniques can be deployed. classes and functions created which pass in parameters which change what HDL is created (IEEE754 FP16 / 32 / 64 for example)
- python-based for-loops can e.g. read CSV files then generate a hierarchical nested suite of HDL Switch / Case statements (this is how the Libre-soc PowerISA decoder is implemented)
- extreme OO abstraction can even be used to create "dynamic partitioned Signals" that have the same operator-overloaded "add", "subtract", "greater-than" operators

Why another Vector ISA? (or: not-exactly another)

- Simple-V is a 'register tag' system. *There are no opcodes* SV 'tags' scalar operations (scalar regfiles) as 'vectorised'
- (PowerISA SIMD is around 700 opcodes, making it unlikely to be able to fit a PowerISA decoder in only one clock cycle)
- Effectively a 'hardware sub-counter for-loop': pauses the PC then rolls incrementally through the operand register numbers issuing *multiple* scalar instructions into the pipelines (hence the reason for a multi-issue OoO microarchitecture)
- Current *and future* PowerISA scalar opcodes inherently *and automatically* become 'vectorised' by SV without needing an explicit new Vector opcode.
- Predication and element width polymorphism are also 'tags'. elwidth polymorphism allows for BF16 / FP16 / 80 / 128 to be added to the ISA *without modifying the ISA*

Quick refresher on SIMD

- SIMD very easy to implement (and very seductive)
- Parallelism is in the ALU
- Zero-to-Negligeable impact for rest of core

Where SIMD Goes Wrong:

- See "SIMD instructions considered harmful" https://sigarch.org/simd-instructions-considered-harmful
- Setup and corner-cases alone are extremely complex. Hardware is easy, but software is hell. strncpy VSX patch for POWER9: 250 hand-written asm lines! (RVV / SimpleV strncpy is 14 instructions)
- O(N^6) ISA opcode proliferation (1000s of instructions) opcode, elwidth, veclen, src1-src2-dest hi/lo

Simple-V ADD in a nutshell

```
function op_add(rd, rs1, rs2, predr) # add not VADD!
 int i, id=0, irs1=0, irs2=0;
 for (i = 0; i < VL; i++)
   if (ireg[predr] & 1<<i) # predication uses intregs
     ireg[rd+id] <= ireg[rs1+irs1] + ireg[rs2+irs2];
    if (reg_is_vectorised[rd] ) { id += 1; }
    if (reg_is_vectorised[rs1]) { irs1 += 1; }
    if (reg_is_vectorised[rs2]) { irs2 += 1; }
```

- Above is oversimplified: Reg. indirection left out (for clarity).
- SIMD slightly more complex (case above is elwidth = default)
- Scalar-scalar and scalar-vector and vector-vector now all in one
- OoO may choose to push ADDs into instr. queue (v. busy!)

Additional Simple-V features

- "fail-on-first" (POWER9 VSX strncpy segfaults on boundary!)
- "Twin Predication" (covers VSPLAT, VGATHER, VSCATTER, VINDEX etc.)
- SVP64: extensive "tag" (Vector context) augmentation
- "Context propagation": a VLIW-like context. Allows contexts to be repeatedly applied. Effectively a "hardware compression algorithm" for ISAs.
- Ultimate goal: cut down I-Cache usage, cuts down on power
- Typical GPU has its own I-Cache and small shaders. *We are a Hybrid CPU/GPU: I-Cache is not separate!*
- Needs to go through OpenPOWER Foundation 'approval'

Summary

- Goal is to create a mass-volume low-power embedded SoC suitable for use in netbooks, chromebooks, tablets, smartphones, IoT SBCs.
- No way we could implement a project of this magnitude without nmigen (being able to use python OO to HDL)
- Collaboration with OpenPOWER Foundation and Members absolutely essential. No short-cuts. Standards to be developed and ratified so that everyone benefits.
- Riding the wave of huge stability of OpenPOWER ecosystem
- Greatly simplified open 3D and Video drivers reduces product development costs for customers
- It also happens to be fascinating, deeply rewarding technically challenging, and funded by NLnet

The end

Thank you

Questions?

- Discussion: http://lists.libre-soc.org
- Freenode IRC #libre-soc
- http://libre-soc.org/
- http://nlnet.nl/PET
- https://libre-soc.org/nlnet/#faq