RISC-V Community needs Peripheral Cores

Elkim Roa

@onchipUIS
Good to have an Open ISA. What about Peripheral?

- IP vendors have IP based on previous customer. **Hard to get** a glue-and-play that works for your SoC. → $$$
- There are some std, such as PHYs: USB, LPDDR, PCIe, AMBA
  **BUT**
  no for clocking circuitry, biasing, GPIO
  For instance a simple Power-on-Reset can hit your pocket, just because!
- Buses IP are out there but expensive.
  Why: Similar to compilers decades ago

Can we do better?
We might
1. Why we need more open-std silicon-HW?

2. IC Community can build up peripheral. Our case.

3. Suggestion and take away.
Case 1: Receiving/Sending Data Bottleneck
Case 1.1: LowRISC Endeavour

Epiphany-V from Adapteva, 1024-core and 1024-IO running at 150MHz.
Case 2: Linux Drivers are Still a Pain in the Butt

Most of the open-source drivers exist because reverse-engineering.

- Sound card drivers - House Peripheral controller
- Graphic drivers - House Peripheral controller
  (worst)

There is no standard for hardware abstraction.

If standardization is possible, future code will be compatible with any device.

- RISC-V core as controller - Nvidia
- Register abstraction

Driver support with Open-hardware peripherals will work on any OS.

No additional reverse-engineering will be needed if everything follows a standard.

Open-hardware would translate into quality drivers

Article title: 11 Reasons Linux Sucks
There is no standard for GPIO.

Standard declarations using Open-hardware GPIO

Give standard features:
- Switching speed
- Current driving

Implementation driver for all users and companies

New ideas can be standardizable!
Developed IP
Peripheral tied to AMBA Buses

AXI4

RAM 4kB
ADC 10bit 10MS/s
DAC 12-bit
GPIO 8 I/O
SPI (Master)

APB

SPI (Master)

AXI4-light (master)
AXI4-light slave interface
AXI4-light-APB bridge
APB

Security Core
CNN Accelerator

RISC-V Core
Accelerator Core

AXI4 + APB + Peripheral
Pseudo-Synthesized ADC

Capacitive-DAC

Vref

Vinp

Vinn

SAR & Control Logic

reset

clk

Dout[9:0]

EoC

Std Cell-based High-Speed Strong-Arm Comparators
Synthesized CDR and PLL
USB 3.1 Gen 2
Analogish Front-End
ChipScope and Offset Correction
ChipScope and Offset Correction: Measurements
PRBS Gen and Checker: Handcrafted FF Std Cell
Fig. 2: LPDDR3 Writing process.

Fig. 3: LPDDR3 reading process.
Establishing a Common Key

Software: Large computational cost.

HW: Large footprint/Power
A Light-Weight Mechanism : TPM

Figure: TPM layout in 130nm. Final area consumption is 160\(\mu\)m\(\times\)160\(\mu\)m.
Fully Synthesized: True-Random Noise Generator

3-edge Ring Oscillator

Conventional Ring Oscillator
Going Beyond: NVRAM on CMOS
### Suggestion: RISC-V Recommends

<table>
<thead>
<tr>
<th>Open-IP Bus</th>
<th>RISC-V core used</th>
<th>Provider</th>
<th>Repository</th>
<th>Functional Model</th>
<th>Silicon proven</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPI</td>
<td>Rocket/Open-V/Epiphany-V</td>
<td>Adapteva</td>
<td>github.com/adapteva</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>I2C</td>
<td>Open-V/Pulp</td>
<td>Onchip/Pulp</td>
<td>github.com/onchipUIS</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>AXI-4.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>APB-4.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AHB-2.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>USB 2.0 PHY</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>USB 3.1 PHY</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RapidIO</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LPDDR3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Compliance test suite - Functional model → UVM
Take away

- Don’t get your SoC-idea stuck just because you don’t have regular IP.
- We need more standard Peripherals.
- Don’t reinvent the wheel. We need to join efforts.