Falcon: NVIDIA’s proprietary RISC

- Falcon = FAst Logic CONtroller
- General purpose embedded processor
- Design started in ~2005; production ~2007
Falcon’s history

- Embedded in 15+ designs
- Taped out in ~50 chips
- Shipped ~3 billion times
- No stop-ship bugs

<table>
<thead>
<tr>
<th>Falcons shipped estimate</th>
</tr>
</thead>
<tbody>
<tr>
<td>dGPU Volume /year</td>
</tr>
<tr>
<td>Years Falcon shipping</td>
</tr>
<tr>
<td>Avg. #Falcons / GPU</td>
</tr>
<tr>
<td>Avg. NVIDIA market share</td>
</tr>
<tr>
<td>Total shipped</td>
</tr>
</tbody>
</table>

Why replace such a successful design?

- Use cases getting more complex
  - Large complex SW
  - External SW
  - Threaded SW
  - Large virtual memory space

- Falcon limitations
  - Low performance
  - No caches (Icache added lately)
  - No thread protection
  - 32-bit address range
  - One size fits all
Selecting the next architecture

Technical criteria
- >2x performance of Falcon
- <2x area cost of Falcon
- Support for caches as well tightly coupled memories
- 64-bit addresses
- Suitable for modern OS

Considered architectures
- ARM
- Imagination Technologies MIPS
- Synopsys ARC
- Cadence Tensilica
- RISC-V
Why RISC-V for Falcon Next

RISC-V is the only architecture that meets all our criteria


<table>
<thead>
<tr>
<th>Item</th>
<th>Requirement</th>
<th>ARM A53</th>
<th>ARM A9</th>
<th>ARM R5</th>
<th>RISC-V Rocket</th>
<th>NV RISC-V</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core perf</td>
<td>&gt;2x falcon</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Area (16ff)</td>
<td>&lt;0.1mm^2</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Security</td>
<td>Yes</td>
<td>TZ</td>
<td>TZ</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>TCM</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>L1 I/D $</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Addressing</td>
<td>64bit</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Extensible ISA</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Safety (ECC/Parity)</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Functional Simulation model</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Scalability - beyond Falcon replacement

- Falcon ISA is monolithic
- RISC-V is flexible
  - 32, 64, and 128-bit versions
  - Cost/performance and supervisor options
  - Custom extensions
- NV-RISCV uses RV64IM_Sdef ISA
- New opportunities
  - Address lower cost and higher perf problems
  - Backward compatibility allows opening up to 3rd party programmers
  - Mix and match internally and externally developed cores
Open source architecture

Control
- Match NVIDIA interfaces and tools
- Original reason for Falcon

Quality
- Large community of contributors
- E.g. memory model tuning

Cost of ownership
- No license, royalty fees
- ISA, tools from community

<table>
<thead>
<tr>
<th></th>
<th>Licensed (ARM)</th>
<th>NVIDIA proprietary (Falcon)</th>
<th>Open source (RISC-V)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Quality</td>
<td>0</td>
<td>0</td>
<td>+</td>
</tr>
<tr>
<td>Cost</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
</tbody>
</table>
Why contribute to RISC-V?

- Benefit from thriving RISC-V community and architecture
- Influence the direction so RISC-V and (y)our interests align

Polder model: cooperation despite differences
Memory model workgroup

- Daniel Lustig chairing the workgroup
- Presentation on Wednesday
- Workgroup meeting on Thursday
Why is it important to NVIDIA?

Jetson TX2
- ~20 CPU cores
- 256 GPU cores
- Complex bus topology
- Coherent and non-coherent traffic
What is the problem?

Core0
{[a] == 1}
ld [a] → x
st [b], 0
{x == 0}

Core1
{[b] == 1}
ld [b] → y
st [a], 0
{y == 0}

That obviously cannot happen!
What is the problem?

Core0

{[a] == 1}

ld [a] → x

st [b], y

{x == 0}

Core1

{[b] == 1}

ld [b] → y

st [a], x

{y == 0}

Or can it?
Should HW or compiler prevent this?

Network

Video decode
Core0

Display controller
Core1

SDRAM

Not here

Maybe here

Core0
L1 cache

Core1
L1 cache
L2 cache
Memory model final remarks

- Need to balance between performance and ease of use
  - Weak memory model allows for more HW optimizations
  - Strong memory model allows for simpler SW
- May need flexibility to switch /combine depending on use case
- Different choices and vagueness exists in established architectures
Security architecture workgroup

- Joe Xie chairing the workgroup
- Helped organize the workshop
- Workgroup meeting on Thursday
Why is it important to NVIDIA?

the total revenue lost to pirated games was approximately $74.1 billion in 2014

http://gearnuke.com/video-game-piracy-rise-will-cost-industry-much-makes/

Chinese company hacks Tesla car remotely

What is the problem?

- Attack surface is growing
- Systems become more open
- Consequences are more severe

Add trust levels

Reduce attack surface of security sensitive software & provide isolation
Security Final Remarks

- Other proposals under discussion
  - Crypto ISA extensions to accelerate common algorithms (Microsemi)
  - Security metatags

- Excellent overview of all proposals in
  Richard Newell, Sr Principal Product Architect, Microsemi Corp.
  Escrypt Security Class, Embedded World Conference, Germany
  March 15, 2017

- Hardware attacks (e.g. differential power attacks) also a concern; not preventable by architecture
Final thoughts

- NVIDIA will use RISC-V processors in many of its products
- We are contributing because RISC-V and our interests align
- Contribute to the areas that you feel passionate about!