PROCESSORS FOR THE CONNECTED WORLD
Your application is unique, so why isn’t your processor?

AUTOMATED RISC-V VERIFICATION FLOW
Utilizing Simulation, Formal, and Emulation Technologies
FLEXIBLE RISC-V ISA

1 RISC-V ISA standard ⇒ many RISC-V HW architecture variants:
   Base is only integer instruction set ("I/E")
   Can be enhanced by standard extensions: integer multiplication and division ("M"), atomic instructions for handling real-time concurrency ("A"), IEEE floating point ("F") with double-precision ("D") and quad-precision ("Q")
   Can have compressed instructions ("C")
   Can have different number of the registers (16 or 32) of different sizes (32 or 64 bits).

We do not verify only 1 processor but many of its variants with enabled/disabled extensions!
OVERVIEW

Codix Berkelium processors from Codasip
Automated generation of SDK, tests, RTL, UVM by Codasip Studio
Mentor formal, simulation, VIP, emulation tools
Resulting productivity
CODIX CORES

CODIX HELIUM
Ideal for applications that demand performance and low power. Rich instruction set, variable-length instructions and compact code.

CODIX COBALT
Excellent general purpose embedded processor for ASIC and FPGA design with unique instruction-set features.

CODIX BERKELIUM
Delivers RISC-V instruction-set compatibility with the option of ZScale or Rocket-like architectures.

CODIX TITANIUM
Powerful multi-slot VLIW processor with a Von-Neumann architecture. It is ideal for applications that require a powerful compact core.
**CODIX BK3 – IP VARIANTS**

**Standard ISA:**
- **I** = integer ISA, 32 GPRs
- **E** = integer ISA, 16 GPRs
- **M** = multiplication extension
- **C** = compressed instructions

**Codasip HW extensions:**
- **U** = user mode with memory protection
- **P** = parallel multiplier/divider
- **j** = branch prediction
- **d** = JTAG debug
CODIX BK5 – IP VARIANTS

Standard ISA:
I = integer ISA, 32 GPRs
E = integer ISA, 16 GPRs
M = multiplication extension
F = floating-point ISA

Codasip HW extensions:
U = user mode with memory protection
d = JTAG debug
MORE VERIFICATION, BETTER PRODUCTS
RTL and Questa Autocheck
Verification Plan and Questa VRM
UVM and Questa Sim + Questa VIP
Regressions and Veloce Emulation
2.1.1 Functional Objects

These objects are mostly connected to processing of instructions and interrupts, synchronization of events.

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Block Name</th>
<th>Object Description</th>
<th>Bk3</th>
<th>Bk5</th>
</tr>
</thead>
<tbody>
<tr>
<td>FO_ISA_OBJ1</td>
<td>INSTRUCTION_SET</td>
<td>Every instruction from I (base integer) instruction set must be checked at least once.</td>
<td>✔️TH</td>
<td>✔️TH</td>
</tr>
<tr>
<td>FO_ISA_OBJ2</td>
<td>INSTRUCTION_SET</td>
<td>Every user instruction must be checked at least once. They must be running in all modes.</td>
<td>✔️TH</td>
<td>✔️TH</td>
</tr>
<tr>
<td>FO_ISA_OBJ3</td>
<td>INSTRUCTION_SET</td>
<td>Every master (privileged) instruction must be checked at least once. They must be running only in the machine mode (otherwise an exception is set, which denotes illegal instruction). RET, WFI, SFENCE</td>
<td>✔️TH</td>
<td>✔️TH</td>
</tr>
<tr>
<td>FO_ISA_OBJ4</td>
<td>INSTRUCTION_SET</td>
<td>Every instruction from M (multiplication/division) instruction set must be checked at least once.</td>
<td>✔️TH</td>
<td>✔️MB</td>
</tr>
<tr>
<td>FO_ISA_OBJ5</td>
<td>INSTRUCTION_SET</td>
<td>Division by zero in division instruction must be checked.*</td>
<td>✔️MB</td>
<td>✔️MB</td>
</tr>
<tr>
<td>FO_ISA_OBJ6</td>
<td>INSTRUCTION_SET</td>
<td>Division overflow must be checked. See table 5.1.</td>
<td>✔️TH</td>
<td>✔️TH</td>
</tr>
<tr>
<td>FO_ISA_OBJ7</td>
<td>INSTRUCTION_SET</td>
<td>Detection of finishing multi/div must be checked. Bk3: div_by_0 at least 3 cycles, div/mul/rem at least 35 cycles, fast_mul at least 4 cycles. If CSR instruction before div/rem/mul + 1 cycle latency. Hazards may also increase latency of these instructions.</td>
<td>✔️MB</td>
<td>✔️TH</td>
</tr>
<tr>
<td>FO_ISA_OBJ8</td>
<td>INSTRUCTION_SET</td>
<td>Every instruction from C (compression) instruction set must be checked at least once.</td>
<td>✔️MB</td>
<td>✔️MB</td>
</tr>
<tr>
<td>FO_ISA_OBJ9</td>
<td>INSTRUCTION_SET</td>
<td>Every instruction from F (single precision floating point) instruction set must be checked at least once.</td>
<td>✔️MB</td>
<td>✔️MB</td>
</tr>
</tbody>
</table>
RTL and Questa Autocheck

Verification Plan and Questa VRM

UVM and Questa Sim + Questa VIP

Regression and Veloce Emulation
Main differences between simulation and emulation

- Compilation of RTL files: 0:00:15 (Simulator), 0:00:17 (Emulator)
- Simulation/emulation of 1 random program (~100,000 instructions): 0:01:33 (Simulator), 0:01:16 (Emulator)

Comparison of total time needed for testing (different time scale!)

Total testing time for 1, 100 and 1000 programs

- Total testing time for 1 random program: Simulator: 2:01:31, Emulator: 2:06:55
- Total testing time for 100 random programs: Simulator: 0:33:13, Emulator: 5:17:35
- Total testing time for 1000 random programs: Simulator: 21:08:11, Emulator: 21:00:00
SHORTENING DEVELOPMENT TIME

- Sub-variant/Extension-variant of RISC-V core + accelerated verification (reuse)
- Initial development flow by Codasip + full verification
- Standard RISC-V processor development flow
SUMMARY

When we want to make RISC-V successful, strength is in maximizing automation in generating RTL and verification!

Codasip automates most processes from single high-level description, including RTL, SDK and UVM generation

Long-term partnership with Mentor increases quality of Berkelium cores every day 😊
Thank you for your attention!

zachariasova@codasip.com, Shakeel_Jeeawoody@mentor.com
+420 541 141 475 (CZE)  www.codasip.com, www.mentor.com