Open-source processor IP in the SCRx family of the RISC-V compatible cores by Syntacore

Ekaterina Berezina

2nd RISC-V Meeting, Paris, Oct 2019

scr1@syntacore.com

www.syntacore.com
Syntacore introduction

IP company, founding member of RISC-V foundation

Develops and licenses state-of-the-art RISC-V cores

- Silicon-proven and shipping to customers
- 4+ years of focused RISC-V development
- Core team comes from 10+ years of highly-relevant background
- SDKs, samples in silicon, full collateral

Full service to specialize CPU IP for customer needs

- One-stop workload-specific customization for 10x improvements
  - with tools/compiler support
- IP hardening at the required library node
- SoC integration and SW migration support
Company background

- Est 2015, 30+ EEs
- HQ at Cyprus (EU)
  - R&D offices in St. Petersburg and Moscow (Russia)
- Representatives in China/APAC, EMEA

Team background:

- 10+ years in the corporate R&D (major semi MNC)
- Developed cores and SoC are in the mass productions
- 15+ tapeouts, 180..14 nm

Expertise:

- Low-power and high-performance embedded cores and IP
- ASIP technologies and reconfigurable architectures
- Architectural exploration & workload characterization
- Compiler technologies
Open-source SCR1 core

Compact MCU core for deeply embedded applications and accelerator control

- **Open sourced under SHL-license** since May 2017
  - Industry-grade with unrestricted commercial use allowed
- **RV32l|E[MC] ISA**, M-mode only
- **<15 kGates** in basic RV32EC configuration
- Configurable 2 to 4 stages pipeline
- Optional IPIC with 16 IRQs
- Optional RISC-V Debug subsystem with JTAG interface
- Verification suite
- Extensive documentation
- Updated and maintained
  - Best-effort support provided, commercial available

https://github.com/syntacore/scr1

UPD

Open-source SCR1 core

Copyright © 2019 Syntacore. All trademarks, product, and brand names belong to their respective owners.
SCR1 codebase

Codebase:
- SystemVerilog, ~10000 LOC, ~500kB code size
- 33 configurable options
- 3 predefined/recommended configurations:
  - **Minimal** RV32EC w/o uncore
  - **Basic** RV32IC with debug/irq
  - **Max performance** RV32IMC

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Area, kgates</th>
<th>Artix-7 utilization*, LUT/FF</th>
<th>Coremark**, MHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimal RV32EC w/o uncore</td>
<td>11</td>
<td>2099 / 818</td>
<td>1.01</td>
</tr>
<tr>
<td>Basic RV32IC with debug/irq</td>
<td>26</td>
<td>4355 / 2267</td>
<td>1.27</td>
</tr>
<tr>
<td>Max performance RV32IMC</td>
<td>33</td>
<td>5753 / 2413</td>
<td>2.95</td>
</tr>
</tbody>
</table>

* utilization for Digilent Arty board (XC7A35T)

** Coremark 1.0, GCC 8.1 BM from TCM, -O3 -funroll-loops -fpeel-loops -fgcse-sm -fgcse-las -flto
SCR1 out-of-box simulation

Quick start from SCR1 repo root:

```
> make run_verilator_wf ARCH=IMC
```

All major simulators supported:

- Altera ModelSim
- Synopsys VCS
- Cadence NCSim
- Verilator + waveforms

Tests:

- RISC-V ISA tests
- RISC-V Compliance suite
- Dhrystone benchmark
- CoreMark benchmark
- Other tests/sample apps
## SCRx IP features at glance

### Baseline cores:
- Clean-slate designs in System Verilog
- Configurable and extensible
- 100% compatible with major EDA flows

<table>
<thead>
<tr>
<th>Features</th>
<th>SCR1</th>
<th>SCR3</th>
<th>SCR4</th>
<th>SCR5</th>
<th>SCR7</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Width</strong></td>
<td>32bit</td>
<td>64bit</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ISA</strong></td>
<td>RV32</td>
<td>RV32</td>
<td>RV32</td>
<td>RV32</td>
<td>RV64</td>
</tr>
<tr>
<td><strong>Pipeline type</strong></td>
<td>In-order</td>
<td>In-order</td>
<td>In-order</td>
<td>In-order</td>
<td>Superscalar</td>
</tr>
<tr>
<td><strong>Pipeline, stages</strong></td>
<td>2-4</td>
<td>3-5</td>
<td>3-5</td>
<td>Sttatic BP, BTB, BHT, RAS</td>
<td>7-9</td>
</tr>
<tr>
<td><strong>Branch prediction</strong></td>
<td>Static BP, RAS</td>
<td>Static BP, RAS</td>
<td>Static BP, RAS</td>
<td>User, Supervisor, Machine</td>
<td>Dynamic BP, BTB, BHT, RAS</td>
</tr>
<tr>
<td><strong>Execution priority levels</strong></td>
<td>Machine</td>
<td>User, Machine</td>
<td>User, Machine</td>
<td>User, Supervisor, Machine</td>
<td>User, Supervisor, Machine</td>
</tr>
<tr>
<td><strong>Extensibility/customization</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Execution units</strong></td>
<td>MUL/DIV</td>
<td>area-opt</td>
<td>hi-perf</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FPU</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Memory subsystem</strong></td>
<td>TCM</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L1$ w/ECC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Memory substrate</strong></td>
<td>MMU, virtual memory</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Debug</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Integrated T&amp;G debug</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HW BP</td>
<td>1-2</td>
<td>1-8 adv ctrl</td>
<td>1-8 adv ctrl</td>
<td>1-8 adv ctrl</td>
<td>1-8 adv ctrl</td>
</tr>
<tr>
<td>Performance counters</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Interrupt Controller</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IRQs</td>
<td>8-32</td>
<td>8-1024</td>
<td>8-1024</td>
<td>8-1024</td>
<td>8-1024</td>
</tr>
<tr>
<td><strong>Features</strong></td>
<td>basic</td>
<td>advanced</td>
<td>advanced</td>
<td>advanced</td>
<td>advanced+</td>
</tr>
<tr>
<td><strong>SMP support</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>up to 4 cores with coherency</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>IF options</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AHB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AXI</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ACE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Fully featured SW development suite

Stable IDE in production:
- GCC 8.1
- GNU Binutils 2.31.0
- Newlib 3.0
- GNU GDB 8.0.50
- Open On-Chip Debugger 0.10.0
- Eclipse 4.9.0

Hosts: Linux, Windows

Targets: BM, Linux (beta)

Also available:
- LLVM 5.0
- CompCert 3.1
- 3rd party vendors in 2019

Simulators:
- Qemu
- Spike
- 3rd party vendors

JTAG-based debug solutions:
Supports: Segger J-link, Olimex ARM-USB-OCD family, Digilink JTAG-HS2, Lauterbach trace32, more vendors in 2019
3rd party tools support

- Lauterbach Trace32
  https://www.lauterbach.com/frames.html?pro/pro__syntacore.html

- IAR Embedded Workbench
  NEW!
  https://www.iar.com/iar-embedded-workbench/#!?architecture=RISC-V

- Segger Embedded Studio
  https://wiki.segger.com/Syntacore_SCR1_SDK_Arty

...more in 2019
Stable Eclipse/gcc based toolchain with IDE:
- GCC 8.1
- GNU Binutils 2.31.0
- Newlib 3.0
- GNU GDB 8.0.50
- Open On-Chip Debugger 0.10.0
- Eclipse 4.9.0

HW platform based on standard FPGA dev.kits
- Multiple boards supported (Altera, Xilinx)
- Low-cost 3rd party JTAG tools
- Open design for easy start

Software:
- Bootloader
- OS: Zephyr/FreeRTOS/Linux
- Application samples, tests, benchmarks
Open SCR₁ SDKs

SDK platforms:
- Digilent Arty (Xilinx) @25MHz
- Terasic DE10-Lite (Intel) @20MHz
- Arria V GX Starter (Intel) @30MHz
- Digilent Nexys 4 DDR (Xilinx) @30MHz
- Lattice iCE40

NEW

Software:
- Bootloader
- Zephyr RTOS
- Tests/SW samples

https://github.com/syntacore/scr1-sdk

Open designs and pre-build images for a quick start, extensive guides
Extensibility/customization

- **Dynamic power**
  - Customized core
  - General-purpose core

- **Full energy**

- **Processing time**

Copyright © 2019 Syntacore. All trademarks, product, and brand names belong to their respective owners.
Workload-specific customization

Extensibility features:
- Computational capabilities
  - New functions using existing HW
  - New Functional Units
- Extended storage
  - Mems/RF, addressable or state
  - Custom AGU
- I/O ports
- Specialized system behavior
  - Standard events processing
  - Custom events

Domain examples:
- Computationally intensive algorithms acceleration
- Specialized processors (including DSP)
- High-throughput applications
  - Wire Speed Processing/DPI/Real-time/Comms

Copyright © 2019 Syntacore. All trademarks, product, and brand names belong to their respective owners.
## SCRx extensibility example

Custom ISA extension for AES & other crypto kernels acceleration for SCR5

### Data
- RV32G – FPGA-based devkit, g++ 5.2.0, Linux 4.6, optimized C++ implementation
- RV32G + custom – same + intrinsics
- Core i7 6800K @ 3.4GHz, g++ 5.4.0, Linux 64, optimized C++ implementation

- 60..575x speedup @ modest area increase: 11.7% core, 3.7% at the CPU cluster level

### Platform Performance

<table>
<thead>
<tr>
<th>Platform</th>
<th>Fmax, MHz</th>
<th>Encoding throughput, MB/s</th>
<th>Normalized per MHz, MB/s</th>
<th>RV32G + custom speed-up</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Crypto-1</td>
<td>Crypto-2</td>
<td>AES-128</td>
</tr>
<tr>
<td>RV32G</td>
<td>20</td>
<td>0.025</td>
<td>0.129</td>
<td>0.238</td>
</tr>
<tr>
<td>RV32G + custom</td>
<td>20</td>
<td>14.375</td>
<td>15.188</td>
<td>14.502</td>
</tr>
<tr>
<td>Core i7</td>
<td>3400</td>
<td>79.115</td>
<td>235.343</td>
<td>335.212</td>
</tr>
<tr>
<td>Core i7 + NI</td>
<td>3400</td>
<td>79.115</td>
<td>235.343</td>
<td>335.212</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3874.552</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Disclaimer: Authors are aware AES allows for more efficient dedicated accelerators designs, used as example algorithm.

---

Copyright © 2019 Syntacore. All trademarks, product, and brand names belong to their respective owners.
Getting access/evaluation

**SCR1**
- Is fully open: [https://github.com/syntacore/scr1](https://github.com/syntacore/scr1) and [https://github.com/syntacore/scr1-sdk](https://github.com/syntacore/scr1-sdk)
- SHL-licensed with unrestricted commercial use allowed
  - Commercial SLA-based support is available

**SCR 3|4|5|7**
- Full package* access is available after simple evaluation agreement

For more info: evaluation@syntacore.com

(*) sufficient for evaluation and tapeout
Summary

- Syntacore offers high-quality RISC-V compatible CPU IP
  - Founding member, fully focused on RISC-V since 2015
  - Silicon-proven and shipping in volume
- Open-source SCR1 core
  - Unrestricted commercial use allowed
  - In full wafer production
- Turnkey IP customization services
  - with full tools/compiler support
Thank you!

scr1@syntacore.com

www.syntacore.com