Syntacore introduction and SCRx IP family overview

RISC-V Day Shanghai
30 June 2018

info@syntacore.com
www.syntacore.com
Outline

- Company intro
- SCRx product line overview
- Extensibility/customization service
Syntacore introduction

IP company, founding member of RISC-V foundation

Develops and licenses state-of-the-art RISC-V cores
- Initial line is available and shipping to customers
- 3 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
Syntacore background

Company:
- Est 2015
- R&D offices in St.Petersburg and Moscow
  - Representatives in EMEA, APAC
- 25+ employees, hiring

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

Expertise:
- Low-power and high-performance embedded cores and IP
- ASIP technologies and reconfigurable architectures
- Architectural exploration & workload characterization
- Compiler technologies
SCRx baseline cores

- **SCR1:** Compact MCU-class open-source core
  - Minimal area configuration is ~11 kGates
  - [https://github.com/syntacore/scr1](https://github.com/syntacore/scr1)

- **SCR3:** High-performance 32-bit MCU with privilege modes
  - Competitive characteristics

- **SCR4:** 32-bit MCU core with high-performance FPU
  - IEEE 754-2008 compatible

- **SCR5:** Efficient mid-range APU/embedded core
  - 1GHz@28nm, virtual memory, 2-4 cores SMP, Linux

Stable designs, immediately available for evaluation
- SDKs, silicon samples, tools, documentation, support
- All cores are licensed
  - Lead customer’s SoC is deployed in the field in 2018

**Baseline** cores: extensible and customizable
Additions to the family in progress
## SCRx features at glance

<table>
<thead>
<tr>
<th>Features</th>
<th>SCR1</th>
<th>SCR3</th>
<th>SCR4</th>
<th>SCR5</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Width</strong></td>
<td>32bit</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>64bit</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Pipeline, stages</strong></td>
<td>2-4</td>
<td>3-5</td>
<td>3-5</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>Static 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>
</tr>
<tr>
<td><strong>Extensibility/customization</strong></td>
<td>V</td>
<td>V</td>
<td>V</td>
<td>V</td>
</tr>
<tr>
<td><strong>Execution units</strong></td>
<td>V</td>
<td>O</td>
<td>V</td>
<td>V</td>
</tr>
<tr>
<td><strong>FPU</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Memory subsystem</strong></td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td><strong>TCM</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>L1$</strong></td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td><strong>L2$</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>MMU, virtual memory</strong></td>
<td>V</td>
<td>V</td>
<td>V</td>
<td>V</td>
</tr>
<tr>
<td><strong>Debug</strong></td>
<td>V</td>
<td>V</td>
<td>V</td>
<td>V</td>
</tr>
<tr>
<td><strong>Integrated JTAG debug</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>HW BP</strong></td>
<td>1-2</td>
<td>1-8 adv ctrl</td>
<td>1-8 adv ctrl</td>
<td>1-8 adv ctrl</td>
</tr>
<tr>
<td><strong>Performance counters</strong></td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td><strong>IPIC</strong></td>
<td>V</td>
<td>V</td>
<td>V</td>
<td>V</td>
</tr>
<tr>
<td><strong>Features</strong></td>
<td>8-32</td>
<td>8-128</td>
<td>8-128</td>
<td>8-128</td>
</tr>
<tr>
<td><strong>SMP support</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>OCP</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>AHB</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>AXI4</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Download SCR1 free at [https://github.com/syntacore/scr1](https://github.com/syntacore/scr1)*
SCR1

Compact MCU core for deeply embedded applications
- RV32I[E][MC] ISA
- <15 kGates in basic untethered configuration (RV32EC)
- 2 to 4 stages pipeline
- M-mode only
- Optional configurable IPIC
  - 8..32 IRQs
- Optional integrated Debug Controller
  - OpenOCD compatible
- Choices of the optional MUL/DIV unit
  - Area- or performance- optimized
- Open sourced under SHL-license (Apache 2.0 derivative with HW specific)
  - Unrestricted commercial use allowed
- High quality free MCU IP
- In the top 3 System Verilog Github repos in the world
- Commercial support offered

Performance*, per MHz

<table>
<thead>
<tr>
<th></th>
<th>DMIPS -O2</th>
<th>Coremark -best**</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>-O2</td>
<td>-best**</td>
</tr>
<tr>
<td></td>
<td>1.30</td>
<td>1.73</td>
</tr>
<tr>
<td></td>
<td>2.78</td>
<td></td>
</tr>
</tbody>
</table>

* Dhrystone 2.1, Coremark 1.0, GCC 7.1 BM from TCM
** -O3 -funroll-loops -fpeel-loops -fgcse-sm -fgcse-las -flto

Synthesis data:
- Minimal RV32EC config: 11 kGates
- Default RV32IMC config: 32 kGates
- 250+ MHz @ tsmc90lp (typical, 1.0V, +25C)
SCR1 SDK

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

Repository contents:
- docs - SDK documentation
- fpga - SCR1 SDK FPGA projects
- images - precompiled binary files
- scr1 - SCR1 core source files
- sw – sample SW projects

Supported platforms:
- Digilent Arty (Xilinx)
- Terasic DE10-Lite (Intel)
- Arria V GX Starter (Intel)

Software:
- Bootloader
- Zephyr OS
- Tests/sample apps
- Pre-built GCC-based toolchain (Win/Linux)
SCR1 support

- Best-effort support provided
  - scr1@syntacore.com
  - academic/research use welcome

- SLA-based commercial support available
  - Optimal configuration
  - Integration into SoC
  - Integration at client-specific SDK boards
  - Tapeout support at the target node
  - Compiler/development tools
  - Customization service
**SCR3**

**High-performance MCU-class core**

- RV32I, optional M and C extensions
- Machine and User privilege modes
- Optional MPU (Memory Protection Unit)
- Tightly Coupled Memory (TCM) support
  - 4..1024KB
- 32bit AHB or AXI4 external interface
- Optional high-performance or area-optimized MUL/DIV unit
- Optimized for area and power
- Integrated IRQ controller
- Advanced debug with JTAG i/f

---

**Performance**, per MHz

<table>
<thead>
<tr>
<th></th>
<th>DMIPS</th>
<th>-O2</th>
<th>best**</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dhrystone 2.1</td>
<td>1.86</td>
<td>2.937</td>
<td></td>
</tr>
<tr>
<td>Coremark 1.0</td>
<td></td>
<td>3.30</td>
<td></td>
</tr>
</tbody>
</table>

* Dhrystone 2.1, Coremark 1.0, GCC 7.1 BM from TCM
** -O3 -funroll-loops -fpeel-loops -fgcse-sm -fgcse-las -flto
SCR4

MCU-class core with high-performance FPU

- RV32IMCF[D] ISA
- Configurable advanced BP, fast MUL/DIV
- Integrated IRQ controller
- U- and M-mode
- 32- or 64-bit AHB or AXI4 external interface
- Optional MPU
- Optional configurable TCM, L1 caches
- Advanced debug controller with JTAG i/f
- Configurable SP or DP FPU
  - IEEE 754-2008 compliant

**Performance**, per MHz

<table>
<thead>
<tr>
<th></th>
<th>DMIPS</th>
<th>Coremark</th>
<th>DP Whetstone</th>
</tr>
</thead>
<tbody>
<tr>
<td>-O2</td>
<td>1.86</td>
<td>2.96</td>
<td>3.30</td>
</tr>
<tr>
<td>-best**</td>
<td></td>
<td>**</td>
<td>**</td>
</tr>
<tr>
<td>GCC 7.1 BM from TCM</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>-O3 -funroll-loops -peel-loops -fgcse-sm -fgcse-las -fho</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
SCR5

Efficient mid-range APU/embedded core

- RV32IMC[AFD] ISA, 64bit option
- Single-, dual- and quad-core SMP configurations
- Advanced BP (BTB/BHT/RAS), IRQ controller, JTAG debug
- M-, S- and U-modes
- Virtual memory support, full MMU
- L1, L2 caches with coherency, atomics
- High performance double-precision FPU
- Linux and FreeBSD support
- 1GHz+ @28nm

Performance*, per MHz

<table>
<thead>
<tr>
<th></th>
<th>O2</th>
<th>-best**</th>
<th>Coremark</th>
<th>-best**</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMIPS</td>
<td></td>
<td>1,60</td>
<td></td>
<td>2,83</td>
</tr>
</tbody>
</table>

* Dhrystone 2.1, Coremark 1.0, GCC 7.1 BM from TCM
** O3-funroll-loops -fpeel-loops -fgcse-sm -fgcse-las -flto
SW development tools

IDE in production:
- GCC 7.1.1
- GNU Binutils 2.29.0
- Newlib 2.5.0
- GNU GDB 8.0.50
- Open On-Chip Debugger 0.10.0
- Eclipse 4.7.0

Hosts: Linux, Windows
Targets: BM, Linux (beta)

Also available:
- LLVM 5.0
- CompCert 2.6

Simulators:
- Qemu
- Spike

Debug solution:
Segger J-link, Olimex ARM-USB-OCD family, Digilink JTAG-HS2 supported, Lauterbach – H2’18

Copyright © 2018 Syntacore. All trademarks, product, and brand names belong to their respective owners.
SCR3 SDK Example

Integrated IDE/debug solution:
- GCC 7.1.1 (upstream)
- GNU Binutils 2.29.0 (upstream)
- Newlib 2.5.0
- GNU GDB 8.0.50
- Open On-Chip Debugger 0.10.0
- Eclipse IDE 4.7.0

Standard FPGA-based devkit board
- Number of boards supported (Arria V, Xilinx, others)
- Low-cost JTAG probe cables
- Open design

Software:
- First stage bootloader
- Zephyr/FreeRTOS, others – on request
- Sample applications, tests, benchmarks

Integrated IDE/debug solution:
- GCC 7.1.1 (upstream)
- GNU Binutils 2.29.0 (upstream)
- Glibc 2.22
- GNU GDB 8.0.50
- Open On-Chip Debugger 0.10.0
- Eclipse IDE 4.7.0

Standard FPGA-based devkit board
- Number of boards supported (Arria 10, Xilinx US+, others)
- Low-cost JTAG probe cables
- Open design

Software:
- First stage bootloader
- SMP Linux (4.x kernel)
- Sample applications, tests, benchmarks

Evaluation

SCR₁
- Is fully open

SCR₃-4-5
- Full package can be provided under simple evaluation agreement
Deliverables (“what is included?”)

**Standard core package (SCR3)**
- RISC-V compatible core
  - RV32IMC ISA
  - RTL (encrypted for evaluation stage), suitable for simulation and synthesis
  - Netlist for the required FPGA devices (Xilinx/Altera)
- Simulation environment
  - Testbench, verification environment
  - Architectural tests suite (pre- and post-si)
- Synthesis support harness
  - Sample scripts, SDC/timing constraints for the required flow
- Reference instantiation examples (for AHB and AXI sockets)
- Back-end support @ required process node (PDK access to be provided)
  - Full cycle: synthesis, floor-planning, netlist verification, PaR/CTS/timing closure, DRC, FEV, DFT)
- Support for 1 tapeout up to a year is included

**Tools (pre-built & sources)**
- GCC 7.1.0 based toolchain
  - Compiler, debugger, linker, functional simulator, binutils, newlib, openocd
- Eclipse-based IDE (Linux, Windows)

**FPGA-based SDK**
- Sample FPGA project (open design)
- Pre-build FPGA and SW images

**SW:**
- First stage bootloader (SC-BL)
- ZephyrOS /FreeRTOS for the SDK board, including BSP
- Application samples for BM env (tests)

**Documentation**
- SCRx quick-start guide
- SCRx EAS (External architecture specification)
- SCRx ISM (Instruction set manual)
- SCRx SDK guide
- Tools guide (IDE & CLI)
Extensibility/customization: how it works
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
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

<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></td>
<td></td>
<td></td>
<td></td>
<td></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>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.
Additional RISC-V resources

- Foundation: [http://riscv.org](http://riscv.org)
- GitHub: [https://github.com/riscv](https://github.com/riscv)
- Mailing Lists: [http://riscv.org/mailing-lists](http://riscv.org/mailing-lists)
- Open-source SCR1 core: [https://github.com/syntacore/scr1](https://github.com/syntacore/scr1)
- Stack Overflow: [http://stackoverflow.com/questions/tagged/riscv](http://stackoverflow.com/questions/tagged/riscv)
- “The Case for Open Instruction Sets”, *Microprocessor Report*
- “RISC-V Offers Simple, Modular ISA”, *Microprocessor Report*
- “An Agile Approach to Building RISC-V Microprocessors”, *IEEEMicro41*
Thank you!