High performance RISC-V embedded SweRV™ EH1 core: microarchitecture, performance and SSD controller implementations

Dr. Zvonimir Z. Bandic, Robert Golla, Joseph Rahmeh, Ofer Shinaar, Matthew Smittle and Paresh Pattel, Western Digital Corporation

RISC-V Instruction Set Architecture (ISA) has become a key driver of open source hardware projects. Most recently we have for example seen a lot of applications in the Internet of Things (IoT), microcontrollers for a variety of traditional embedded applications, and applications requiring capability for low power operation of Artificial Intelligence (AI) inference engines based on artificial neural networks. We have developed a super-scalar (2-way), 9-stage pipeline, mostly in-order, open-source core based on the RISC-V RV32IMC instructions set, named SweRV EH1. We present some of the architectural details of the core and implementation challenges, as well as discuss application of the core for the Flash controllers.

I. Introduction

In the last 3 years we have witnessed a flurry of activity in the domain of open source hardware implementations based on the RISC-V instruction set architecture (ISA) [1] [2][3][4]. The RISC-V architecture allows for a variety of power-performance optimized microarchitectural implementations for many compute problems spanning a gamut from an IoT controller to a datacenter processor. At the same time, all these applications can share an ecosystem of the most important common open source software components such as open source compilers, debuggers, embedded real time operating systems (RTOS), and ubiquitous operating systems such as Linux. In this paper, we report our results on the first RISC-V architecture based high performance embedded SweRV EH1 core, the microarchitectural details, 28nm TSMC implementation and performance benchmarks. Our SweRV EH1 core RTL implementation is open-sourced and available on GitHub at [3].

II. SweRV EH1 Core Microarchitecture

A. RISV-V – choice of ISA base and extensions

Our embedded application (storage controllers) does not require 64-bit memory address space, so we opt for 32-bit. However, we do require support for multiple/divide (M) and compressed (C) instructions. Therefore, we implement the RV32IMC instruction set variant.

B. SweRV EH1 core microarchitecture

The microarchitecture of the SweRV EH1 core is shown in Figure 1. The core is a superscalar, dual issue 9-stage pipeline supporting 4 arithmetic logic units (ALU) labeled EX1 and EX4, each in two pipelines I0 and I1. Furthermore it includes

one load/store pipeline, one multiplier pipeline and one out-of-pipeline 34-cycle latency divider. Compared to previous open source RISC-V cores, such as Rocket [6] or Pulpino [7] that employed classical 1-issue (scalar) pipeline, SweRV EH1 uses superscalar dual-issue micro architecture. Dual issue pipelines typically improve various performance benchmarks by 20-30% [8], at a relatively small expense of core gate count or implementation area. Based on our current understanding, SweRV EH1 core is the first RISC-V ISA in-order superscalar core, whose RTL is available as open source [5]. The pipeline shown in Figure has 9 stages, including writeback. There are total of four stall points in the pipeline: Fetch1, Align, Decode and Commit. Align will form instructions from 3 fetch buffers. Decode will decode up to 2 instructions from 4 instruction buffers. Commit will commit up to 2 instructions per cycle, depending on the workload.

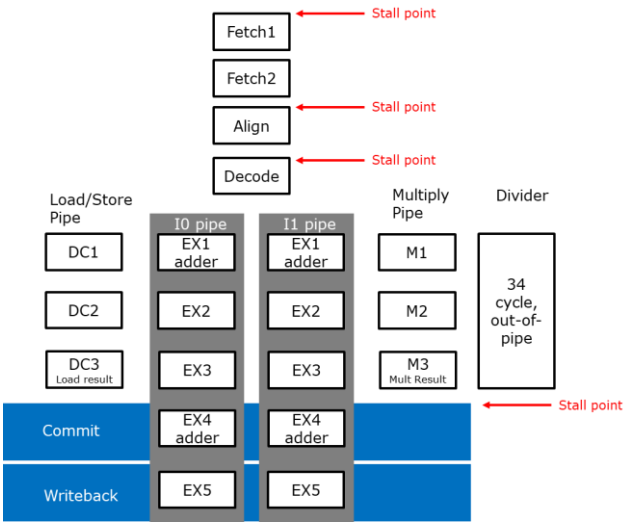

Figure 1 SweRV core microarchitecture showing dual-issue 9-stage pipeline with 4 execution units, load/store unit, 2-cycle multiplier and out-of-pipe 34-cycle divider unit.

In addition to dual instruction issue, one difference of the SweRV EH1 core compared to previous RISC-V open source cores [6],[7] is the presence of 4 symmetric ALUs statically assigned to pipelines I0 and I1. Load/store pipeline has load-to-use latency of 2 cycles for dependent load/store addresses (e.g. pointer-chasing). For dependent ALU instructions, the load-to-use varies between 0 and 3 cycles. This implies that arithmetic operations that miss the first opportunity for compute in the EX1 stage, may get second opportunity in the EX4 (Commit), leading to the high utilization of the pipeline. Let's consider a following snippet of RISC-V assembly code:

```
                          # depends on
L1: ld x11,8(x10)
L2: ld x13, 8(x12)
L3: ld x14, 8(x11)        # L1
A4: addi x15,x13,1        # L2
A5: add x16, x13, x14     # L2, L3
L6: ld x17,8(x16)         # A5
A7: addi x17,x17,1        # L6
```

The instruction dependencies are shown in the comments, for example, add immediate instruction A4: refers register x13, which is loaded in instruction L2. The execution of the SweRV EH1 core pipeline is shown in Figure 2 . When instruction A4 reaches stage EX1 in the pipeline, in clock step 4, register x13 is not ready, as load instruction L2 would take two clock cycles to load, and will load in clock step 5. However, A4 continues to propagate through the pipeline,

	1	2	3	4	5	6	7	8	9	10	11	12
DECODE	L1	L2	L3,A4			A5	L6,A7					
EX1/DC1		L1	L2	L3,A4			A5	L6,A7				
EX2/DC2			L1	L2	L3,A4			A5	L6,A7			
EX3/DC3				L1	L2	L3, A4			A5	L6,A7		
EX4/COM					L1	L2	L3, A4			A5	L6,A7	
EX5/WB						L1	L2	L3, A4			A5	L6,A7

Figure 2 SweRV pipeline diagram for the code sample shown above. Green color indicates completed load or arithmetic operation.

and once it reaches stage EX4, it will be able to execute (shown in green in the Figure), as L2 has loaded and x13 is ready. Additionally, we allow some minimal out-of-order execution capabilities. If load instruction misses, and requires access to slower memory in the hierarchy, we continue execution until the next instruction that depends on the data from this load instruction.

C. Memories, Debug and Platform Interrupt Controller

The SweRV EH1 core supports data closed couple memory (DCCM) and instruction closed couple memory (ICCM) in the traditional Harvard architecture, as well as configurable instruction cache (iCache). The iCache can be configured in sizes between 32KB and 256KB. The iCache is 4-way set associative cache with four banks of cache data, with line size of 64 Bytes. Cache accesses are pipelined, and can be accessed every clock cycle. The debug logic and debug port is implemented according to RISC-V debug specifications [9].

D. Branch prediction

The SweRV EH1 core supports GSHARE branch prediction, which represents a good compromise between complexity and branch prediction accuracy in embedded applications workloads [10]. The sizes of the branch target buffer (BTB) and the branch predictor table (BPT) are independently configurable with up to 512 and 2048 registers, respectively. Branches that hit in the BTB will result in one cycle branch penalty, while branches that mispredict in primary (EX1) or secondary (EX4) ALU will result in 4 or 7 cycles branch penalty, respectively.

III. SweRV EH1 core area and frequency

The SweRV EH1 core is targeted for System on Chip implementation for NAND Flash Controllers. The target technology is 28 nm TSMC (125C, 150ps clock skew), see Figure 3. The implementation area, without memories, in slow global corner (SSG) is 0.132 mm2 at 1 GHz, and 0.093 mm2 at 500 MHz. In the case of typical global (TT) corner, the implementation area is 0.092 mm2 at 1 GHz and 0.088 mm2 at 500 MHz.

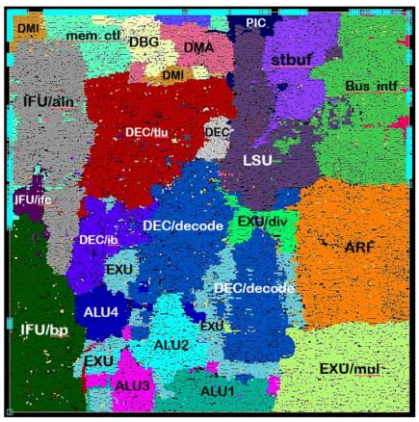

Figure 3 SweRV implementation in 28nm TSMC technology without memories.

IV. Performance benchmarking

Much can be said about performance benchmarks for embedded application use cases. It is difficult to choose the benchmark that realistically reflects a variety of use cases and applications, that today span everything from smallest IoT applications, to compute heavy neural network accelerators for industrial and automotive applications. We have evaluated the performance of the SweRV EH1 core using Coremark benchmark, that is frequently used for embedded CPU cores [12]. The results are shown in Figure 4. SweRV EH1 core achieves 4.9 on the Coremark scores, outperforming many commercial embedded cores, as well as many other RISC-V cores [13]. The overall performance improvement is a consequence of dual issue architecture and availability of 4 ALU units. Another benchmark that is frequently used in embedded applications is Dhrystone [14]. Using optimized string copy function, we were able to achieve 2.9 Dhrystone MIPS/MHz on this benchmark.

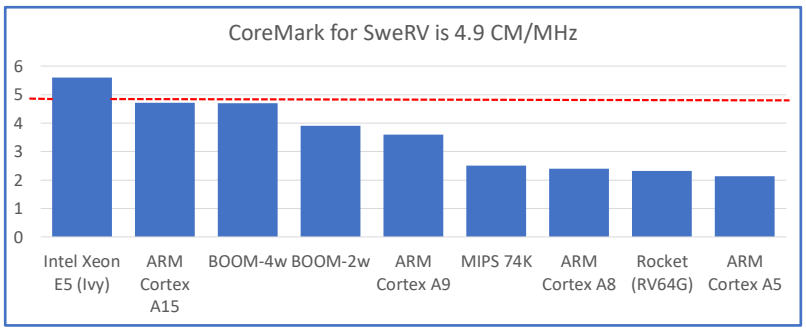

Figure 4 Coremark performance benchmark comparing Coremark benchmark, renormalized per single execution thread and for frequency.

V. SSD Controller Applications

Typically, Solid State Disk (SSD) drives implement Flash controller System on Chip (SoC) that manages NAND Flash array on board, and exposes error-free block I/O interface to the host computer. One typical flash controller SoC

architecture is shown in Figure 5. The main CPU core handles host controller interface, and exposes error-free blocks to the host. Datapath CPU coordinates actions of NAND sequences modules, which read and write from actual NAND chips, and ensures that data is encoded with error correction code (ECC) when written to the NAND, and after reading data, compares stored ECC code values with values read from NAND, to ensure that data is error-free. We have implemented SweRV EH1 cores for datapath CPU implementations.

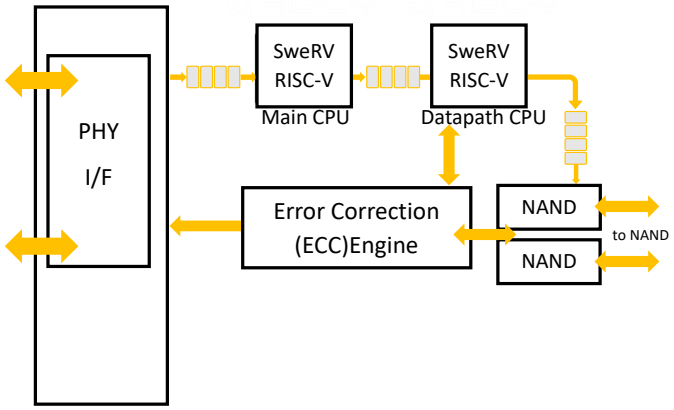

Figure 5 Flash controller SoC architecture for Solid State Disk (SSD) applications.

VI. SweRV cores roadmap and what is next

The technology roadmap for SweRV Cores is shown in Figure 6. The second generation of SweRV cores, SweRV EH2 and EL2 was introduced in December 2019. SweRV EH2 keeps the pipeline design of SweRV EH1, but introduces multi-threading. This allows exposing two RISC-V cores to the host, while keeping almost the same design area and power. For many workloads, that are very I/O intensive, this kind of configuration can be proven very effective in terms of performance to power and performance to area ratio. In addition to high performance upgrade of first generation core, we are also introducing SweRV EL2 – which is a small single-issue, 4-stage redux of SweRV EH1. SweRV EL2 is a fully functional RISC-V core optimized for SoC subsystem implementations (such as sequencers or finite state machines) where small size and high operational frequency are imperative. The source code and documentation are available at CHIPS Alliance GitHub [5].

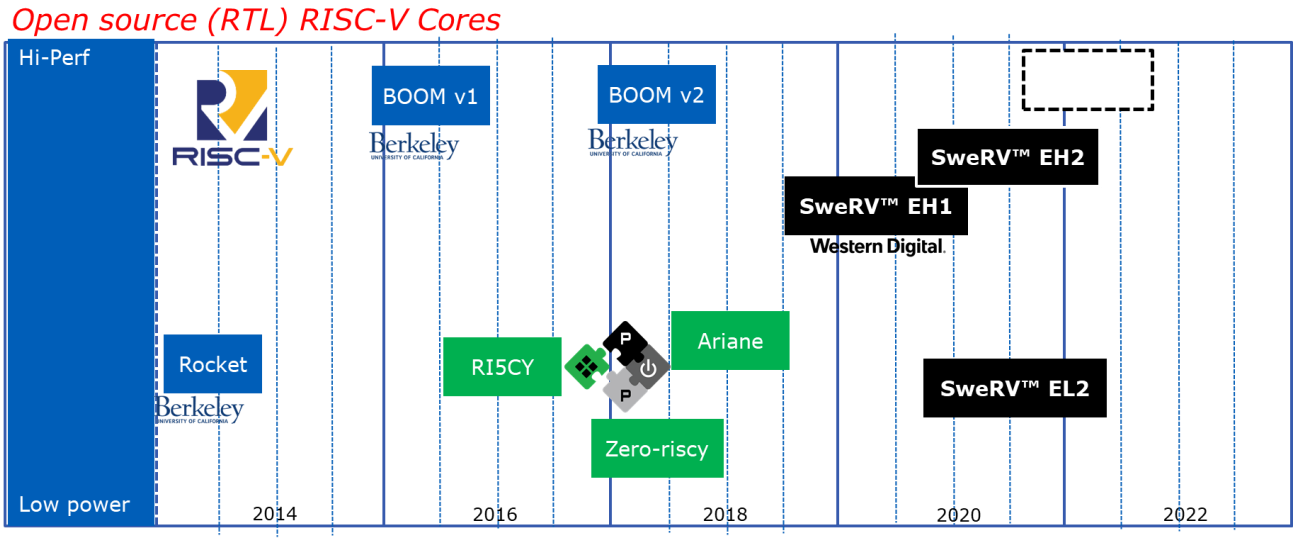

Figure 6 SweRV cores roadmap in CHIPS Alliance. SweRV EH1 core addresses high performance embedded requirements, increasing performance to 5 CM/MHz while keeping size in 0.1 mm2 range. SweRV EH2 introduces multi-threading, while EL2 represents ultra-low power core for variety of SoC applications

VII. CONCLUSION

In this paper, we presented the open sourced designs of SweRV EH1 core[5]. We introduce the 9-stage, superscalar (dual issue) pipeline with 4 ALUs, and point out advantages compared to traditional 5-stage classical RISC pipeline [1,6,7]. The dual issue superscalar design improves the performance by roughly 50 percent on practical benchmarks, by quadrupling the number of ALUs and tolerating load to use latency [1] with dual ALUs per pipeline. The SweRV EH1 core is compatible with RISC-V architecture [9], and is presently one of the highest performing 32-bit RISC-V Core. The size of the core is under 0.1 mm2 for most implementation use cases in 28 nm TSMC technology. This core is considered as very practical for RISC-V based embedded storage controllers. Finally, we have introduced SweRV cores roadmap in CHIPS Alliance, including two new cores, SweRV EH2 and EL2.

References

[1] D.A. Patterson, J.A. Hennessy, “Computer Organization and Design”, MK, [2018]
[2] https://www.sifive.com/
[3] https://www.westerndigital.com/company/newsroom/press-releases/2017/2017-11-28-western-digital-to-accelerate-the-future-of-next-generation-computing-architectures-for-big-data-and-fast-data-environments
[4] https://www.forbes.com/sites/tiriasresearch/2017/12/06/western-digital-gives-a-billion-unit-boost-to-open-source-risc-v-cpu/#a7c8ecf2266e
[5] https://github.com/chipsalliance/Cores-SweRV
[6] https://github.com/freechipsproject/rocket-chip

[7] https://github.com/pulp-platform
[8] M. Johnson, “Superscalar Microprocessor Design”, Chapter 3
[9] https://workspace.riscv.org/higherlogic/ws/public;
https://workspace.riscv.org/higherlogic/ws/groups/debug (RISC-V members) or https://github.com/riscv/riscv-debug-spec
[10] S. McFarling, “Combining branch predictors”, Technical Report TN-36, DEC Corp. (http://www.hpl.hp.com/techreports/Compaq-DEC/WRL-TN-36.pdf)
[11] https://github.com/riscv/riscv-angel
[12] https://www.eembc.org/coremark/;
[13] C.Celio,D.Patterson,K.Asanovic,https://www2.eecs.berkeley.edu/Pubs/TechRpts/2015/EECS-2015-167.pdf
[14] https://www.eembc.org/techlit/datasheets/ECLDhrystoneWhitePaper2.pdf