

ReonV: a RISC-V derivative of SPARC LEON3 processor

Lucas Castro, Rodolfo Azevedo

lcbc.lucascastro@gmail.com, rodolfo@ic.unicamp.br

Institute of Computing – State University of Campinas (UNICAMP)
Av. Albert Einstein, 1251 – 13083-852 – Campinas – SP – Brazil

***Abstract.** This paper endorses the importance of reuse and contribution for open source hardware, offering as example the development of a RISC-V soft-core processor, named ReonV, which was developed by reusing all IP cores from a well designed SPARC 32-bit processor changing only its ISA in order to obtain a fully operational RISC-V processor that inherits all other modules and Board Support Package (BSP) from the original SPARC core.*

1. Introduction

Computing community is very familiar with large scale software reuse, which is one of the advantages of developing open source software. However, when it comes to hardware, specially soft-core processors, reuse is often limited to small modules or extending the ISA with new instructions; normally, new cores development are made from the ground, creating the entire core and its Board Support Package (BSP). This approach raises unneeded obstacles such as recreating already existing modules, incompatibility problems when supporting new peripherals and adapting the BSP to new FPGA boards or running environments.

That motivated this research to build a new RISC-V [1] soft-core, named ReonV, reusing the IP cores from GRLIB, an IP Core Library containing a SPARC V8 processor named LEON3 [2, 3, 4]. The taken approach was to change only the pipeline of the processor in order to implement the RISC-V ISA instead of the original SPARC, inheriting all other already existing modules from LEON3, such as memory, memory controllers, peripherals support, debug support unit, BSP and synthesis scripts for a great number of FPGAs of different vendors. Thus, this work intended to show an example of how entire processors can be reused when developing new ones, adapting only the necessary to implement a different ISA. Even more, this study also aimed to make a contribution to the RISC-V community releasing a RISC-V version of a widely used and tested soft-core processor originally compliant with the SPARC ISA.

2. GRLIB and Leon3

The GRLIB IP Library is an integrated set of reusable IP cores, designed for system-on-chip (SOC) development released under GNU GPL license by Aeroflex Incorporated. The IP cores are centered around the common on-chip bus, and use a coherent method for simulation and synthesis. It also contains template designs for several FPGA boards and scripts in order to facilitate synthesis to supported devices [3, 4, 5].

GRLIB also contains LEON3, which is a synthesizable VHDL model of a 32-bit processor compliant with the SPARC V8 architecture. The model has a great number of configurable options and is specially aimed for system-on-a-chip (SOC) designs [3, 4, 6].

Figure 1 shows a representation of the LEON3 processor. It shows how flexible LEON3 is, since it can be synthesized with a range of at least 5 crucial modules with minimalist options to a total of 17 highly configurable modules [3, 6].

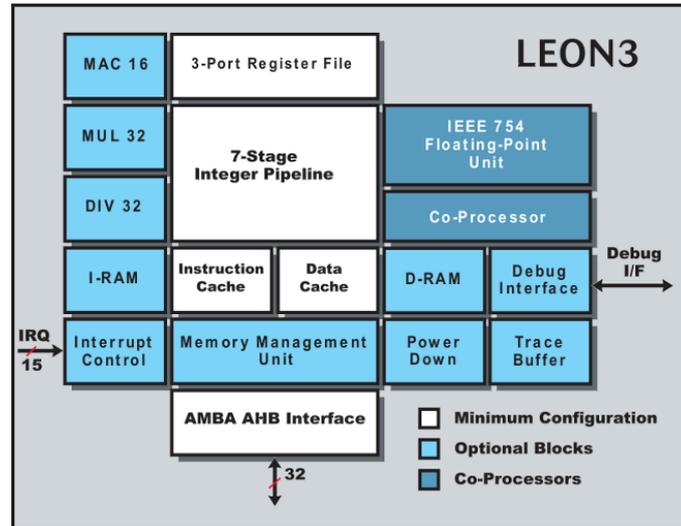


Figure 1. Representation of the LEON3 processor [7].

3. ReonV - A RISC-V version of the LEON3 processor

ReonV is the name given to the 32-bit RISC-V processor obtained after changing the LEON3's ISA from SPARC to RISC-V. It has an open repository¹ and is released under GNU GPL License [8].

ReonV was developed by reusing LEON3 processor and modifying its 7-stage integer pipeline (refer to Figure 1) in order to implement the ISA RV32I [1] without privileged instructions instead of the original SPARC V8, maintaining all other IP cores and resources provided by GRLIB untouched. With this, we aimed to obtain a RISC-V processor which provides all the support to synthesis, FPGAs and peripherals LEON3 has without the need to develop the whole core and its BSP. As a consequence, performance was left aside at this first moment, but there is space for future work in this regard.

4. Methodology and Results

As previously stated, the ReonV development was focused on reusing as much of the original SPARC LEON3 processor as possible. We managed to restrict changes to only modify the 7-stage Integer Pipeline in order to implement RISC-V ISA in the place of SPARC. Since there was no modification outside the pipeline, we intended to automatically gain the support already built for LEON3 to our RISC-V processor.

That allowed us to also use auxiliary software developed for LEON3 such as its synthesis tools, scripts and its debug monitor, named GRMON, to facilitate the communication with the processor and its development [9]. GRMON has an interface which interacts with the processor's Debug Support Unit (DSU), allowing to easily load and run programs, dump memory, read registers values and other useful debug operations [6, 9].

¹Repository available at: <https://github.com/lcbcFoo/ReonV>

While GRMON is a debug monitor for a SPARC processor which brings some expected incompatibility problems with the RISC-V ISA, it proved to be very useful to have such a tool on development stage for the new ISA.

4.1. Changing the ISA

The ISA change took advantage of the fact that the pipeline was already correctly implemented, thus any modifications could be easily tested, specially having the DSU and GRMON on hands. We modified the least necessary to have a working processor running the RV32I ISA, thus the pipeline continued with its original 7 stages (Fetch, Decode, Registers Access, Execute, Memory Access, Exception and Write Back). Most of the signals and modules were reused, with the proper modifications, but there were removed instructions and others completely included, accordingly to the new ISA.

We found problems with incompatible instructions or conventions, such asendianness and branches. ReonV development managed to deal with most of them, but branches required unwanted measures. Since branches on SPARC use already set status flags (Z, C, N, V) to only decide to take the branch or not [2] and, on the other hand, RISC-V branches read 2 registers, compute the condition and only then decide to take the branch or not [1], there was two main options to implement these instructions: make large changes on the pipeline structure and its stages or maintaining the current structure inserting unwanted stalls and without branch prediction. Since the project initially focused more on guaranteeing correctness than on performance, the second option was chosen, the processor’s branch prediction was disabled and stalls were inserted on the pipeline to implement RISC-V branches.

After these changes, all instructions except branches take the same amount of cycles as their equivalent do on SPARC, as shown on table 1. BP refers to Branch Prediction and the instructions abbreviations’ follow the RISC-V specifications [1]. Store instructions take 2 cycles on both models, because the pipeline design needs one cycle to load the memory position and another to send the data.

Type	Instructions	Cycles needed (ReonV / Leon3)
PC relative	AUIPC	1 / not implemented
Control Flow	JALR, JAL	1 / 1
Conditional Control Flow	BEQ, BNE, BLT, BGE, BLTU, BGEU	4 (without BP) / 1 (with BP)
Memory Manipulation	LB, LBU, LH, LHU, LW	1 / 1
Memory Manipulation	SB, SH, SW	2 / 2
Logic and Arithmetic	ADD(I), SUB, XOR(I), OR(I), AND(I), SLL(I), SRL(I), SRA(I), SLT(I), SLTU(I), LUI	1 / 1

Table 1. ReonV instructions and the comparison of cycles needed to equivalent SPARC instructions implemented by LEON3.

The project used the Xilinx Vivado Design Suite [10] and the Nexys4DDR FPGA from AVNET [11] to synthesize and run the processor design. Table 2 compares the resource utilization between LEON3 and ReonV on Nexys4DDR with the same synthesis configuration with clock frequency of 40MHz . We can note that while ReonV uses less LUT, it demands more flip-flops. However, since the difference is very small, we consider both designs to have near the same size.

Resource	ReonV utilization	LEON3 utilization	Available at Nexys4DDR	Utilization % (Reonv / LEON3)
LUT	8665	8820	63400	13.67 / 13.91
LUTRAM	97	98	19000	0.51 / 0.52
FF	5346	4977	126800	4.22 / 3.93
BRAM	23	23	135	17.04
I/O	81	81	210	38.57
BUFG	11	11	32	34.38
PLL	5	5	6	83.33

Table 2. Comparison of board resources utilization between ReonV and Leon3. Available collumn refers to the Nexys4ddr FPGA

4.2. Correctness Benchmarks

In order to certify the correctness of the processor, we set up an automated compilation and running environment to execute tests from the set of benchmarks from WCET [12], adapted to ReonV, since each one of them clearly states what structures it validates.

ReonV was tested and correctly executed each of the described programs of table 3. All tests were executed running the processor’s synthesised design as described on section 4.1. The source code of each test, as all other piece of software needed to run these benchmarks, are available at the project repository [8] and were compiled with the official RISC-V Toolchain [13].

Table 3, also shows which structures are tested on each program, accordingly to the following label:

- L - Has loops
- N - Has nested loops
- B - Bit Manipulation
- A - Uses arrays or matrices
- R - Has recursion

5. Discussion

All tests executed successfully and we consider that the pipeline change was successful and a RV32I processor was obtained reusing the LEON3 IP cores with very few modifications.

File	Description	L	N	A	R	B
bs.c	Binary search	✓		✓		
bsort.c	Bubble sort	✓	✓	✓		
cover.c	Distinct program flow	✓				
expint.c	Integer exponentiation	✓	✓			
fac.c	Factorial	✓			✓	
fibcall.c	Fibonacci	✓				
insertsort.c	Insertion Sort	✓	✓	✓		
complex.c	Nested loops	✓	✓			
matmult.c	Matrix Multiplication	✓	✓	✓		
ndes.c	Bit Manipulation	✓		✓		✓
prime.c	Prime check	✓				
qsort-exam.c	Quick Sort	✓	✓	✓		
recursion.c	Recursion				✓	

Table 3. Test programs used to validate ReonV. All of them are adaptations of the WCET benchmarks [12].

It's also important to note that the tested version of ReonV still does not implement a few important instructions such as multiplication, division and FP operations. However, LEON3 already have modules for executing these instructions [6], suggesting that extending the current implemented ISA will only be another step of hardware reuse and pipeline modification.

Moreover, reusing LEON3 IP core also allowed to use a debug monitor and a DSU since the beginning of the development what was a significant advantage to the project. Also, there was conviction that the other modules were correct since they were not modified and LEON3 is a well established and largely used processor, restricting eventual debug analysis to the changes on pipeline.

Although problems with incompatibility over SPARC and RISC-V ISAs existed and made the initial version of ReonV poorly efficient, specially with branches, it can be improved on future work. Furthermore those incompatibilities are ISA specific and may not occur on related development approaches for other architectures, for example MIPS and RISC-V would not face the same branches incompatibility since MIPS branches are similar to RISC-V ones [1, 14]. Thus, the approach proposed here did not bring any unavoidable or irreversible disadvantage for ReonV development.

6. Conclusion

This work concludes releasing an open source VHDL model of a operational RISC-V soft-core developed reusing another soft-core IP Core Library. This is a successful example of how open source hardware has potential for reuse and what advantages this approach may bring to development.

During development, incompatibility and efficiency problems existed, however they were expected since both ISAs have different instructions and conventions. Furthermore, none of them were irreversible, thus this kind of incompatibility can be fixed or at least minimized on future works. There is also room for forthcoming projects expanding

the current instructions supported by ReonV, such as implementing the multiplication, division and FP instructions.

Finally, this work offers to the RISC-V community the possibility for further development of a RISC-V soft-core processor which already brings with it all the support of a well established IP Core Library as a consequence of its development focus: Hardware Reuse.

References

- [1] *The RISC-V Instruction Set Manual - Volume I: User-Level ISA - Document Version 2.2*. RISC-V Foundation, 1300 Henley Court, Pullman, WA 99163, 509.334.6306. URL <https://riscv.org/specifications/>. Editors Andrew Waterman and Krste Asanovic. Accessed on 07/2018.
- [2] *The SPARC Architecture Manual, Version 8*. Sparc International Inc., 535 Middlefield Road - Suite 210 - Menlo Park - CA 94025.
- [3] *GRLIB IP Library User's Manual - Version 2018.1*. Cobham Gaisler AB, Kungsgatan 12, 411 19 Gothenburg, Sweden, . URL www.cobham.com/gaisler/products/grlib/grlib.pdf. Accessed on 07/2018.
- [4] Cobham Gaisler AB. Grlib ip library. URL www.gaisler.com/index.php/products/ipcores/soclibrary. Accessed on 07/2018.
- [5] *GRLIB IP Core User's Manual - Version 2018.1*. Cobham Gaisler AB, Kungsgatan 12, 411 19 Gothenburg, Sweden, . URL www.cobham.com/gaisler/products/grlib/grip.pdf. Accessed on 07/2018.
- [6] *Configuration and Development Guide - Version 2018.1*. Cobham Gaisler AB, Kungsgatan 12, 411 19 Gothenburg, Sweden, . URL www.cobham.com/gaisler/products/grlib/guide.pdf. Accessed on 07/2018.
- [7] Sven Åke Andersson. Leon3 32-bit processor core. URL www.rte.se/blogg/blogg-modesty-corex/leon3-32-bit-processor-core/1.5. Accessed on 07/2018.
- [8] Lucas Castro. ReonV - A RISC-V version of LEON3. URL www.github.com/lcbcFoo/ReonV. Accessed on 07/2018.
- [9] *GRMON2 User's Manual*. Cobham Gaisler AB, Kungsgatan 12, 411 19 Gothenburg, Sweden, . URL www.gaisler.com/doc/grmon2.pdf. Accessed on 07/2018.
- [10] *Vivado Design Suite User Guide - Release Notes, Installation, and Licensing*. Xilinx, Inc. URL www.xilinx.com.
- [11] *Nexys 4TM FPGA Board Reference Manual*. Digilent Inc, 1300 Henley Court, Pullman, WA 99163, 509.334.6306.
- [12] Mälardalen WCET research group. Wcet benchmark. URL <http://www.mrtc.mdh.se/projects/wcet/benchmarks.html>. Acessado em 30/04/2018.
- [13] RISC-V Foundation. Software Tools. URL www.riscv.org/software-tools. Accessed on 07/2018.
- [14] *MIPS® Architecture for Programmers Volume II-A: The MIPS32® Instruction Set Manual*. Imagination Technologies LTD.