DVINO: A RISC-V Vector Processor Implemented in 65nm Technology

Guillem Cabo*, Gerard Candón*, Xavier Carril*, Max Doblas*, Marc Domínguez*, Alberto González*, César Hernández†, Víctor Jiménez*, Vatistas Kostalampros*, Rubén Langarita*, Neiel Leyva†, Guillem López-Paradís*, Jonnatan Mendoza*, Francesco Minervini*, Julián Pavón*, Cristóbal Ramírez*, Narcís Rodas*, Enrico Reggiani*, Mario Rodríguez*, Carlos Rojas*, Abraham Ruiz*, Víctor Soria*, Alejandro Suanes‡, Iván Vargas*, Roger Figueras*, Pau Fontova*, Joan Marimon*, Víctor Montabes*, Adrián Cristal*, Carles Hernández*, Ricardo Martínez‡, Miquel Moretó*§, Francesc Moll*§, Oscar Palomar*§, Marco A. Ramírez†, Antonio Rubio§, Jordi Sacristán‡, Francesc Serra-Graells‡, Nehir Sonmez*, Lluís Terés‡, Osman Unsal*, Mateo Valero*§, Luís Villa†

*Barcelona Supercomputing Center (BSC), Barcelona, Spain. Email: name.surname@bsc.es
†Centro de Investigación en Computación, Instituto Politécnico Nacional (CIC-IPN), Mexico City, Mexico.
‡Institut de Microelectrònica de Barcelona, IMB-CNM (CSIC), Spain. Email: name.surname@imb-cnm.csic.es
§Universitat Politècnica de Catalunya (UPC), Barcelona, Spain. Email: name.surname@upc.edu

Abstract

This paper describes the design, verification, implementation and fabrication of the Drac Vector IN-Order (DVINO) processor, a RISC-V vector processor capable of booting Linux jointly developed by BSC, CIC-IPN, IMB-CNM (CSIC), and UPC. The DVINO processor includes an internally developed two-lane vector processor unit as well as a Phase Locked Loop (PLL) and an Analog-to-Digital Converter (ADC). The paper summarizes the design from architectural as well as logic synthesis and physical design in CMOS 65nm technology.

I. Introduction

Open-source hardware is increasingly gaining momentum, promising a revolution in hardware design similar to what happened in software since the advent of the Linux operating sytem. In the particular area of processor design, open Instruction Set Architectures (ISA) have been proposed, which enables the implementation of powerful processors that can be customized and extended without any cost.

RISC-V, originated in 2010 at the University of California at Berkeley [1] is one of the fastest growing open alternatives, currently supported by RISC-V International [2], a non-profit organization with hundreds of members in academia and industry. Several European and national projects are currently developing various processors and associated software stack based on RISC-V, which is explicitly targeted in different calls and mentioned in the Chips Act of the EU strategy for semiconductors in 2022 [3].

The existence of an open ISA opens the doors to the implementation of specialized processors with particular capabilities. This is the case of vector processors, in which a single instruction is executed concurrently on a large array of data and thus improve the computing performance on workloads typical of supercomputing applications, such as numerical simulations.

This paper describes the design, verification and physical implementation of DVINO, a RISC-V vector processor capable of booting Linux and executing vector operations. The chip, jointly developed in the framework of the DRAC project [4], contains an in-order processor and a two-lane vector processing unit. A 2-level cache hierarchy based on open-source hardware was also added. In addition, memory controllers, VGA, one ADC and an internal PLL for clock generation are integrated in the chip.

The structure of the paper is as follows: Section II describes the chip architecture and IP components. Section III explains the different verification techniques. Section IV displays the performance of the processor in comparison with similar designs. Next, Section V details the synthesis and physical design results. Section VI shows the measurements performed on the manufactured chip. Finally, Section VII concludes this work.

This is a Post-Print version of the paper published in DCIS 2022 proceedings by IEEE, DOI: 10.1109/DCIS51330.2020.9268664.
Copyright notice: 978-1-6654-5950-1/22/$31.00 ©2022 IEEE
IEEExplore link: https://ieeexplore.ieee.org/document/9970128

II. Chip Architecture

DVINO is a single core processor consisting of several components, each playing a different functionality that covers current processor necessities such as computation capabilities, access to storage, input/output communication. These components are directly integrated into a System-on-Chip (SoC) application-specific integrated circuit (ASIC) taped-out by TSMC in 65nm technology. We also designed and produced a custom Printed Circuit Board (PCB) to interconnect the SoC with other chips such as memory (SDRAM and HyperRAM), reference clock and connectors to external devices. While the PCB with the SoC is designed to work standalone, the SDRAM and HyperRAM for main memory access are new IPs that have been introduced in this tape-out. Hence, we decided to make use of the DDR3 controller and DDR3 Memory IPs available on a Xilinx KC705 Field-Programmable Gate Array (FPGA) board as a backup option to access main memory. This solution was used in the *pre*DRAC chip [5].

Figure 1 shows the final distribution of DVINO IP blocks between the ASIC, the PCB and the FPGA. In the following subsections we will break down the architecture of these components.

A. Processing unit. Scalar core

The main component developed within the project is the DVINO SoC. Figure 1 shows the block diagram of the DVINO single core processor. In the figure we can see all the SoC IPs and how are they connected. The main component of the SoC is the 5-stage single-issue in-order Lagarto Hun core[1]. This core implements the 64-bit RV64IMA scalar RISC-V ISA v2.2 and privileged ISA v1.11. The main function of the core is to fetch, interpret and execute RISC-V instructions from main memory. Thus, the core acts as the central processing unit.

B. Vector Processing Unit

The DVINO SoC includes a VPU codenamed *Hydra* that is based on the original design developed for the European Processor Initiative (EPI) project [6]. The proposal of the vector unit in EPI targets High Performance Computing by means of employing long vectors; an idea similar to the vector engine NEC SX-Aurora TSUBASA [7].

The DVINO SoC implements a subset of the Open Vector Interface (OVI) [8], that allows to have a decoupled design where all memory accesses are performed by a scalar core (Lagarto Hun in the DVINO design), while the VPU is in charge of the actual computation with vectors.

The vector unit for DVINO has a Maximum Vector Length of 4096 bits and supports 4 different vector element widths (8, 16, 32 and 64), therefore having up to 64 Double-Precision elements.

The RISC-V vector specification implemented on this design is version 0.7.1 [9].

C. Cache Hierarchy

The majority of the Cache Hierarchy used in DVINO is adopted from the cache design developed in the Untethered lowRISC SoC version 0.2 [10]. The lowRISC cache hierarchy has two levels of caches. The private caches of the core are located in the first level (L1), with separate instruction and data caches. In the second level (L2) there is a shared instruction and data cache.

Caches are inclusive, which means that the L2 cache contains an active copy of all the data in the L1 caches. This implies the need to invalidate all the copies in the L1 cache of each line evicted from the L2. The MESI cache coherence protocol [11] is the one used in DVINO. In this case, a directory located in L2 is used to maintain the coherence.

The communication protocol between L1 and L2 caches is based on a 128-bit wide TileLink, while AXI and AXI Lite protocols are used to access main memory and peripherals, respectively.

D. Access to Main Memory and Peripherals

There are three memory devices that can be used as main memory of the DVINO processor: HyperRAM, SDRAM (both located on the PCB), and a DDR3 (located on the FPGA). The on-chip SDRAM and HyperRAM controllers were designed for this project following the respective specifications for these type of memories. They provide an alternative way to access main memory without an auxiliary FPGA board. The communication between the processor and the HyperRAM and SDRAM controllers follows the AXI protocol, with a 128-bit data bus. The architecture of each controller is composed by an SDRAM or HyperRAM controller to manage the external memory chips, an AXI wrapper of 32-bit data bus and a FIFO to send or receive blocks of 32 bits from the AXI wrapper.

The SDRAM interface in DVINO has a 32 bit data bus that, combined with 4 chip selects, supports up to 256 MB of memory. The AXI-SDRAM controller is configured to work with SDRAMs at 200 MHz.

In the case of HyperRAM we sacrifice bandwidth compared to SDRAM in order to obtain a reduced number of bus signals (in particular 8 data pins). In total we have 4 HyperRAM chips of 8 MB for a total of 32 MB of memory. The HyperRAM controller is configured to work with the HyperRAM chips at 50 MHz.

[1] *Hun* stands for number one in Mayan and we make use of this term to represent the issue width of the design.

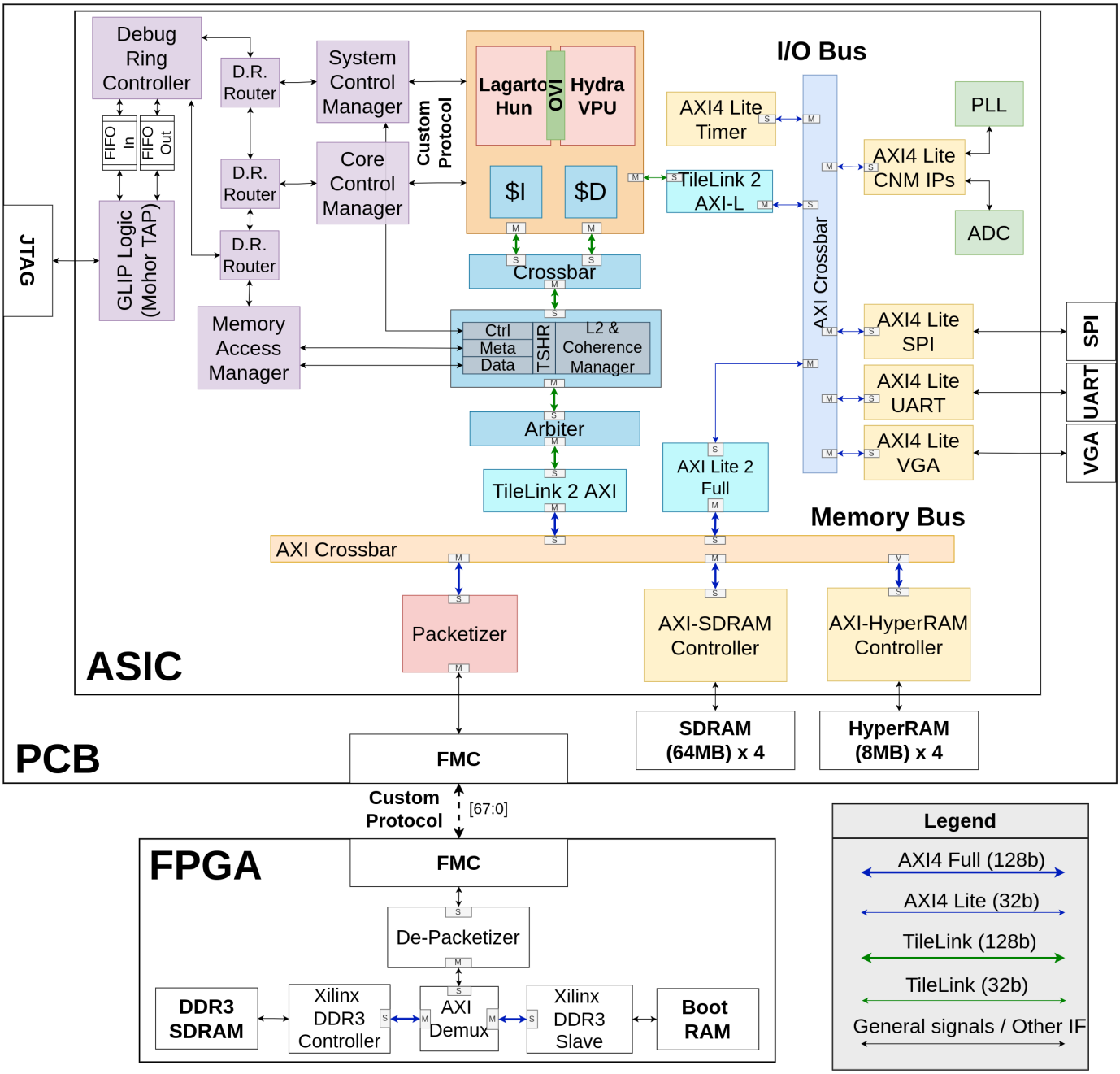

Fig. 1. DVINO processor block diagram detailing the ASIC, PCB and auxiliary FPGA board.

The lack of an on-chip physical interface (PHY) for a DDR3 memory controller motivated us to design a custom interface to communicate with a Xilinx Kintex KC705 board containing memory chips. This implied to split the design into two parts: one on the DVINO ASIC implementing a custom interface to the FPGA side, and another one on the FPGA that uses the integrated DDR3 PHY of the FPGA in the board. Both parts are connected with an FPGA Mezzanine Card (FMC) connector. This split is depicted in Figure 1. This external connection only supports up to 50 MHz transfer speed, and also the 128-bit bus must be split into 16 transactions of 8 bits each.

The custom interface (*Packetizer*) on the ASIC breaks down AXI transactions into packets and sends them through a link. On the FPGA side, a depacketizer reconstructs these packets back into AXI transactions on the FPGA side.

This kind of solutions can suffer Clock Domain Crossing issues because we are connecting two devices that have different clock sources. In order to reduce the possible issues related to CDC, we used a derived clock for the FMC in our ASIC. The FMC clock is simply a fixed 200 MHz reference clock divided by 4 as it is detailed in Section II-F, and is thus aligned with

the main clock of the system, reducing clock complexity.

E. Peripherals

The ASIC has standalone IO controllers connected to the internal AXI-Lite bus for UART, SD card and SPI. Finally, a VGA video output was added to the design. In addition to that, the Packetizer system has been designed to support the full AXI protocol of 128 bits data width as well as the AXI-Lite of 32 bits. In this way, we can use in the FPGA not only the DDR3 memory controllers connected to the full AXI interface, but also the FPGA IO controllers, such as UART and SPI, connected to the AXI-Lite interface.

F. Clocking

The DVINO SoC has different clock sources and different frequency configurations that are detailed in this section. Table I lists the main clock domains of the SoC IPs, explained below.

The clock for the core (and VPU) logic, the main clock, comes either from the external reference (*Ext.* in Table I) or the internal PLL (*PLL*) that can be configured to different frequencies up to 600 MHz. The external reference can be selected from an on-board 200 MHz oscillator or from an external clock source by an SMA connector, allowing to set an arbitrary frequency for the main clock.

Each peripheral requires a clock with a fixed frequency, independently of the main clock frequency being used. Ideally, such reference should be derived from the main clock to avoid synchronization issues. Therefore, since the frequency of the main clock is arbitrary, a Clock Selection Unit (CSU) is necessary that keeps the same reference frequency for the peripherals irrespective of the main clock configuration. In DVINO, the CSU provides a clock signal synchronous to the main clock, with a constant frequency of 200 MHz regardless the main clock is at 200, 400 or 600 MHz, which are the typical configurations. Each peripheral can then use the CSU clock and implement a divider to obtain the frequency they need:

- HyperRAM and Packetizer: 50 MHz (divide by 4.)
- VGA: 25 MHz (divide by 8.)
- UART and SPI: 10 MHz (divide by 20.)

The SDRAM does not use the CSU clock, but the external reference. Clock domain crossing synchronizers are added for this reason. It is important to note that, as mentioned, the reference clock may be set to an arbitrary frequency and thus, frequencies other than 200 MHz will disable the correct operation of the SDRAMs.

TABLE I
CLOCK DOMAINS.

Domain	Source	Frequency
Main clock	PLL or Ext.	up to 600 MHz
Packetizer	CSU	50 MHz
SDRAM	Ext.	200 MHz
HyperRAM	CSU	50 MHz
VGA	CSU	25 MHz
SPI & UART	CSU	10 MHz

G. Analog IPs

There are two analog IPs: a PLL for internal clock generation and an Analog to Digital Converter (ADC.) Both IPs were custom designed specifically for this project. Both IPs can be digitally configured using a dedicated SPI port.

The PLL is an Integer-N PLL with a current-controlled ring oscillator based on a nominal external reference of 200 MHz that is capable of generating a clock signal between 200 MHz and 1.2 GHz in steps of 50 MHz depending on its configuration parameters using the SPI port. The default configuration after a reset is a frequency of 400 MHz. The PLL can also operate with other reference clock frequencies, with its output frequencies being consequently scaled.

The ADC includes a 9-level second-order Delta-Sigma modulator frontend together with a 4th-order cascaded integrator-comb (CIC) decimator and an AXI-4 Lite slave backend for its connectivity to the processor. With 1.2 Vpp differential input full scale and 50-kHz bandwidth, the modulator runs at 12.8 MS/s, while the CIC decimator allows to reduce the oversampling ratio from 128 to 4. This data converter features internal flicker noise cancellation and it is designed to achieve a maximum Signal to Noise and Distortion Ratio (SNDR) of 99.8dB, which is equivalent to 16.3 bit.

H. Debug infrastructure

We developed a debug infrastructure to verify at run-time the state of the SoC, and to be able to inject internal tests without using the access to main memory. This system is referred in RISC-V literature as a Debug Ring [12]. We developed a custom infrastructure that allows to control the SoC initialization for Linux booting operations.

Our implementation is based on the Open SoC Debug Library (OSD) [13], which provides a plug and play communication interface with a base architecture, and the Generic Logic Interfacing Project (GLIP) [14], which gives a generic data exchange protocol based on FIFO queues.

III. Design Verification

A significant effort in the DVINO processor was devoted to verification at different stages of the design: RTL, post-synthesis and post-place and route.

A. RTL and Gate Level Simulations

Following standard industry practices, a Continuous Integration/Continuous Delivery (CI/CD) infrastructure using Gitlab are extensively used throughout the project to automate the periodic running of many jobs. The CI/CD environment is used for various types of automated testing, such as (i) Hardware simulations, testing on different snapshots of the IP (system-level, processor-level, RTL), linting tools such as Synopsys Spyglass, RTL coding errors detection tools, such as Questa Autocheck. (ii) Gate level simulations post-synthesis and post-place and route, connectivity checking, clock domain crossing. (iii) Verification with assertions and code/functional coverage enabled. (iv) Automated runs of benchmarks on the Software Development Vehicle (SDV) on FPGA.

The testing and verification strategy was based on running several types of code, such as (i) simple RISC-V ISA tests [15], which test each instruction, (ii) additional simple RISC-V binaries, (iii) Random Torture tests [16], as well as riscv-dv constrained-random binary generator [17], and (iv) microbenchmarks.

B. Tests on FPGA

To verify the design before moving to the ASIC design flow, multiple tests are performed with an implementation of the system in FPGA, first in one FPGA to test the standalone SoC, and then in two different FPGAs to mimic the final design.

The testing and verification strategy for the preDRAC design consisted of 4 incremental steps: (i) simple RISC-V ISA tests, which test each instruction, (ii) additional RISC-V lowRISC tests, (iii) random generation of torture tests from lowRISC, as well as (iv) booting the Linux kernel.

This verification phase allowed to start the ASIC physical design with confidence in the correct functionality of the system.

IV. Performance of the Lagarto Scalar Core

In this section, we evaluate the IPC performance of the Lagarto Hun design against a previous Lagarto design fabricated in the same technology node in 2019 [5]. Figure 2 shows the IPC improvement of DVINO's Lagarto Hun design over the preDRAC predecessor using the RISC-V benchmarks from lowRISC. The DVINO design achieves an average speed-up of $1.15\times$ with respect to the previous version of the core. Two main reasons introduce the IPC improvements of DVINO with respect to its predecessor: Firstly, DIVINO reduces the access latency of some edge cases of the instruction and data caches. Secondly, the branch prediction mechanism has been improved, reducing the branch misprediction penalty.

Figure 3 shows the IPC comparison of Lagarto Hun against the Ariane core [18] from ETH Zurich with the integer EEMBC Autobench suite. Lagarto Hun shows an IPC in the same range than Ariane, albeit 14% lower.

V. Synthesis and Physical Design

The DVINO SoC is designed using a digital-on-top design flow based on standard cells in TSMC 65nm technology. Standard cell libraries and SRAM macros were obtained via Europractice. SRAM memories are used for the register file, caches and associated tables. Analog IPs (PLL and ADC) were instantiated together with their corresponding pads as a single macro in one corner of the padring.

A. Synthesis

The synthesis tool is Cadence Genus. A multi-mode multi-corner flow was followed using 10-track, regular-Vt cells, and three corners: typical, best and worst cases with Non-Linear Delay Model (NLDM) characterization. Table II shows the corner conditions being considered.

Table III lists the memory block sizes used in the design. The same corners as with the Standard Cells were used for synthesis (Typical, Fast and Slow.)

Table IV shows a summary of the area report after synthesis, where it can be seen that of a total of 546,816 instances, of which 36 correspond to memory cells and 1 to the analog blocks macro.

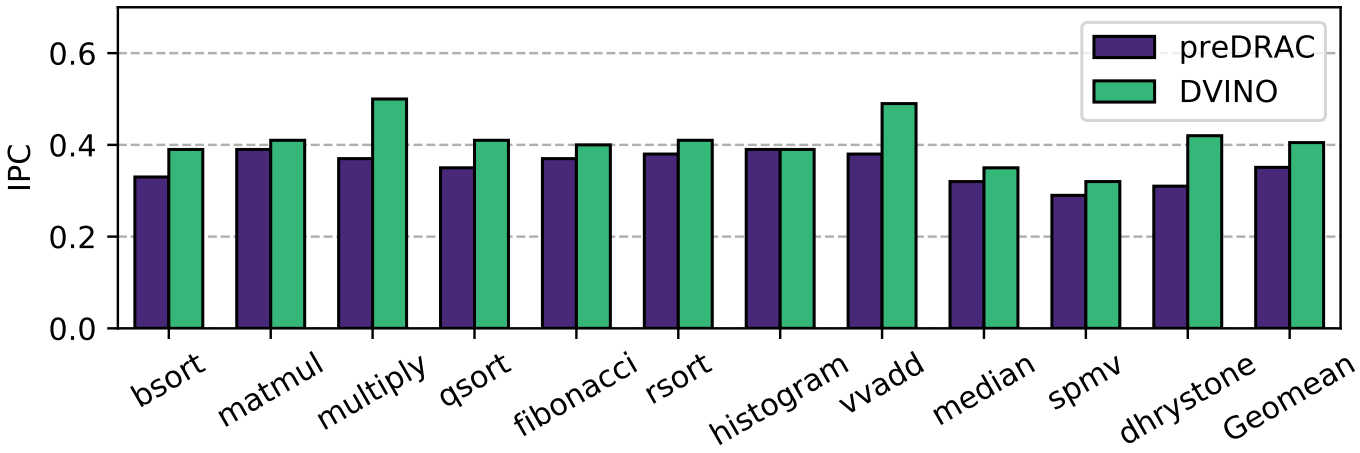

Fig. 2. RISC-V benchmarks IPC performance of DVINO and preDRAC Lagarto Hun designs.

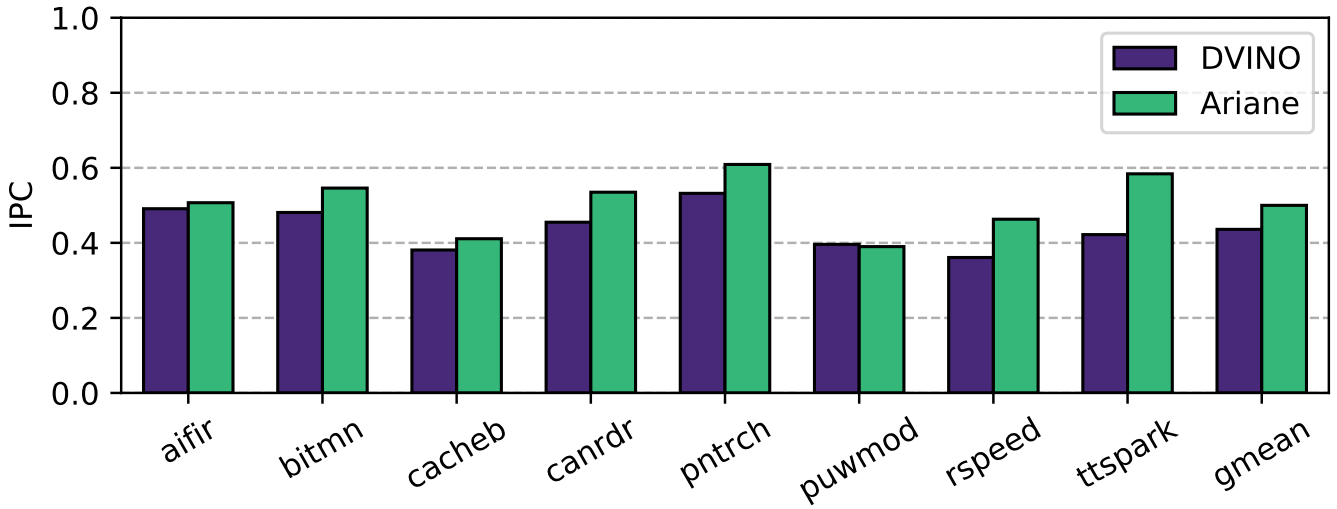

Fig. 3. EEMBC Autobench benchmark suite IPC performance of Lagarto Hun and Ariane.

B. Place and Route

Fig. 4 shows the obtained layout. Design size is $2940\,\mu\text{m}\times2940\,\mu\text{m}$ (8.6436 mm^2), with $2580\,\mu\text{m}\times2580\,\mu\text{m}$ (6.6564 mm^2) for the core. The design cells utilization in the core is around 71%. The custom IPs, located in the top-right corner, use less than $1000\,\mu\text{m}\times500\,\mu\text{m}$.

Most of the rest of the core area is occupied by the VPU (2.248 mm^2) while the SDRAM, Hyperram and VGA controllers take 0.006, 0.013 and 0.068 mm^2 respectively. Fig. 5 is included to emphasize the important percentage of area taken by the VPU. The full chip uses 192 pads and an estimation for the number of transistors is around 12.15M.

TABLE II
LIST OF STANDARD CELL CORNERS.

Corner	Conditions (V and T)
Typical	1.2 V and 25 C
Fast	1.32 V and 0 C
Slow	1.08 V and 125 C

TABLE III
SRAM SIZES. THE SECOND COLUMN INDICATES NUMBER OF INSTANCES.

DxW	Size (kb)	Description
256x128	4 x 32 kbits	I and D L1 Cache (256 kbits total)
512X128	8 x 512 kbits	L2 Cache (4096 kbits)
128x48	2 x 6 kbits	ICache tag(12 kbits))
128x48	2 x 6 kbits	MetadataArray(12 kbits)
128x128	1 x 16 kbits	Part of 128x176 MetadataArray_1
128x48	1 x 6 kbits	Part of 128x176 MetadataArray_1
608X28	1 x 16.625 kbits	VGA Buffer
2048x8	1 x 16 kbits	VGA font Mem
256x64	10 x 16 kbits	Vector Register File (160 kbits)

TABLE IV
INSTANCES SPLIT (POST-SYNTHESIS.)

Type	Instances	Area (μm^2)	Area %
Logic (comb+seq)	546,779	2,778,089.2	55.8
SRAM	36	1,612,974.6	32.4
Other IP	1	586,296.5	11.8
Total	546,816	4,977,360.3	100.0

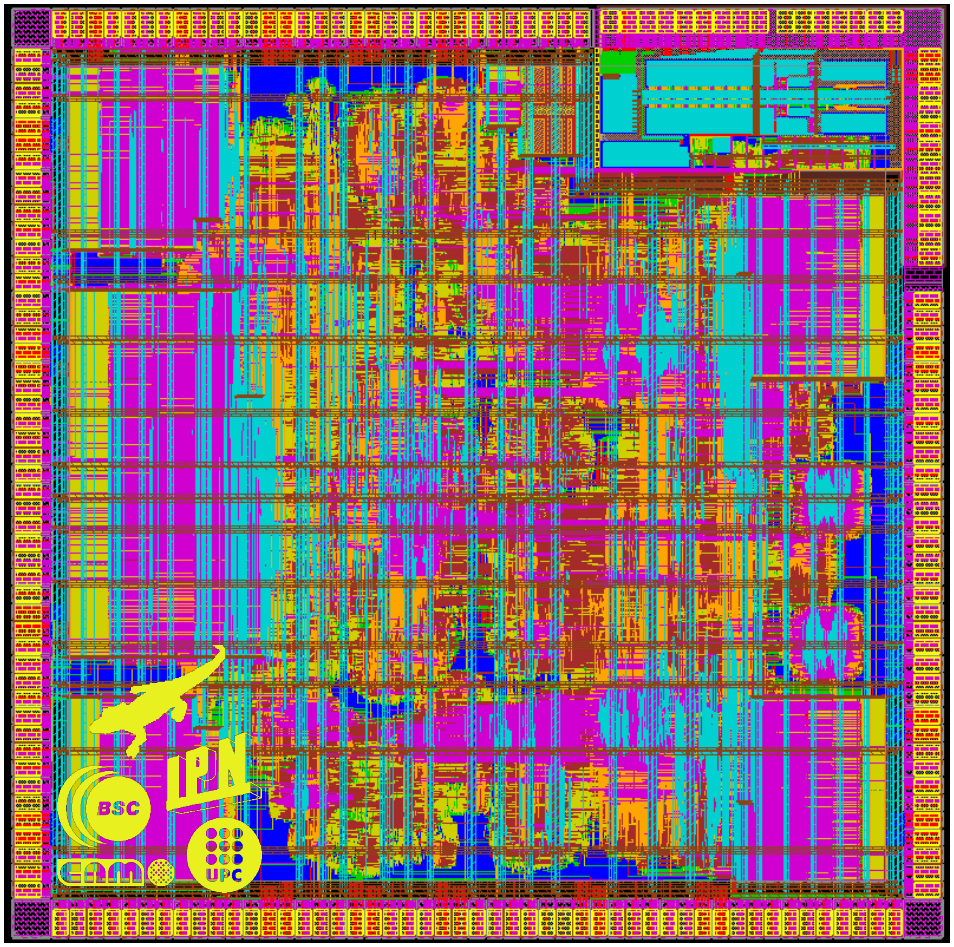

Fig. 4. Full layout of the *DVINO* ASIC. Logos can be identified in the lower-left corner. More details given in text.

C. Power analysis

The power planning is designed according to reported power consumption, technology parameters analysis, and rail analysis results. Of the 192 pads, 7 are for the IO supply (3.3 V), 16 for the core supply (1.2 V), 16 for GND connections, and 1 for

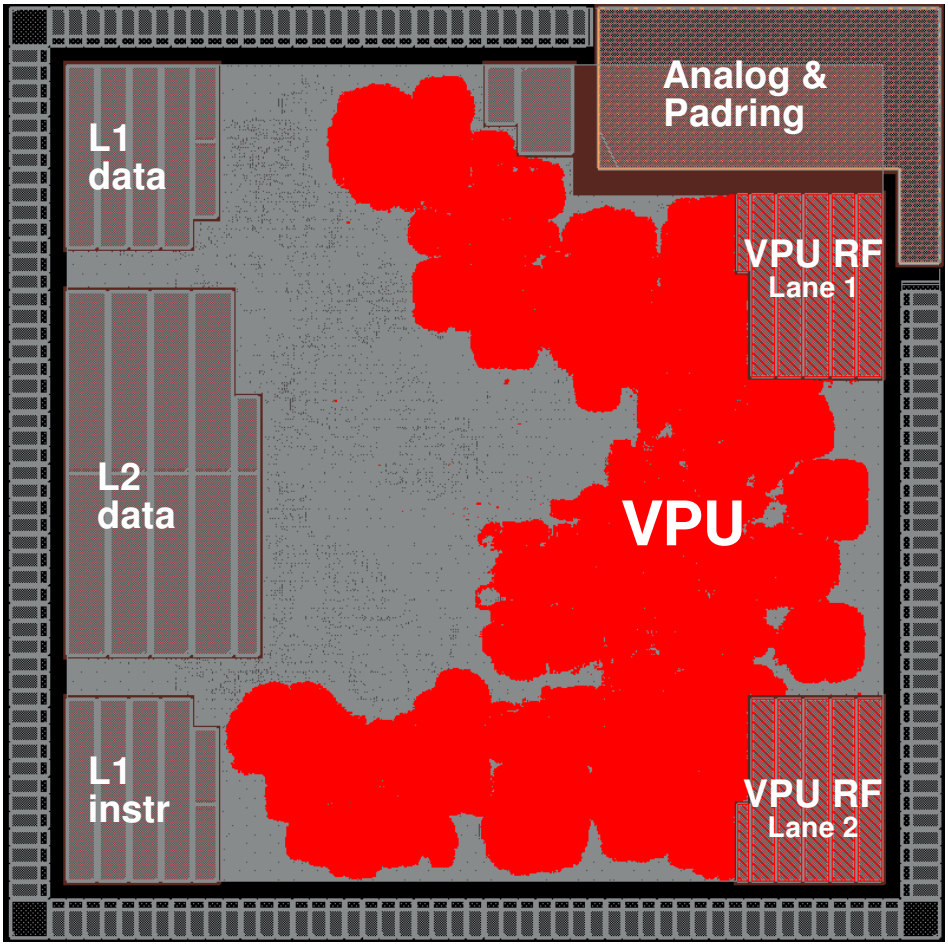

Fig. 5. Chip area distribution. In red, the design area occupied by the VPU, which is almost 50%.

the power-on-control (POC) pad. The analog IP supplies and padring are independent. Results for the IR drop analysis using VOLTUS are presented in Fig. 6. This analysis has been done for a core clock frequency of 600 MHz, which reports 3.2 W of power consumption (Table V.) These results are for the typical corner and vectorless, assuming an average activity of 20%. This represents a worst case analysis, as the actual activity is expected to be much lower. In addition, the default working frequency will be 400 MHz, so around only 2/3 of this power is expected.

TABLE V
POWER REPORT SUMMARY FOR A 600 MHZ MAIN CLOCK (UNITS: MILLI-W).

Group	Internal	Switching	Leakage	Total
Sequential	974.7	77.0	0.4	1,052.0
Macro	503.5	21.2	0.2	524.8
IO	213.6	129.8	0.0	343.4
Combinational	477.7	636.4	1.9	1,116.0
Clock (Comb)	12.1	164.2	0.0	176.3
Total	2,182.0	1,028.0	2.6	3,213.0

Even in this worst case scenario, the reported results show only 75 mV drop at the chip center with the implemented power planning, as shown in Fig. 6. This keeps the circuit still far from the slow corner voltage supply. On the other hand, EM analysis results reports no problems.

VI. POST-SILICON RESULTS

The DVINO design was sent to Europractice for fabrication in May 2021 and the packaged samples were received in November. To test it, we designed a PCB with all the required circuits (power supply, external 200 MHz clock, reset) and connectors (FMC for the FPGA, JTAG, UART, micro SD).

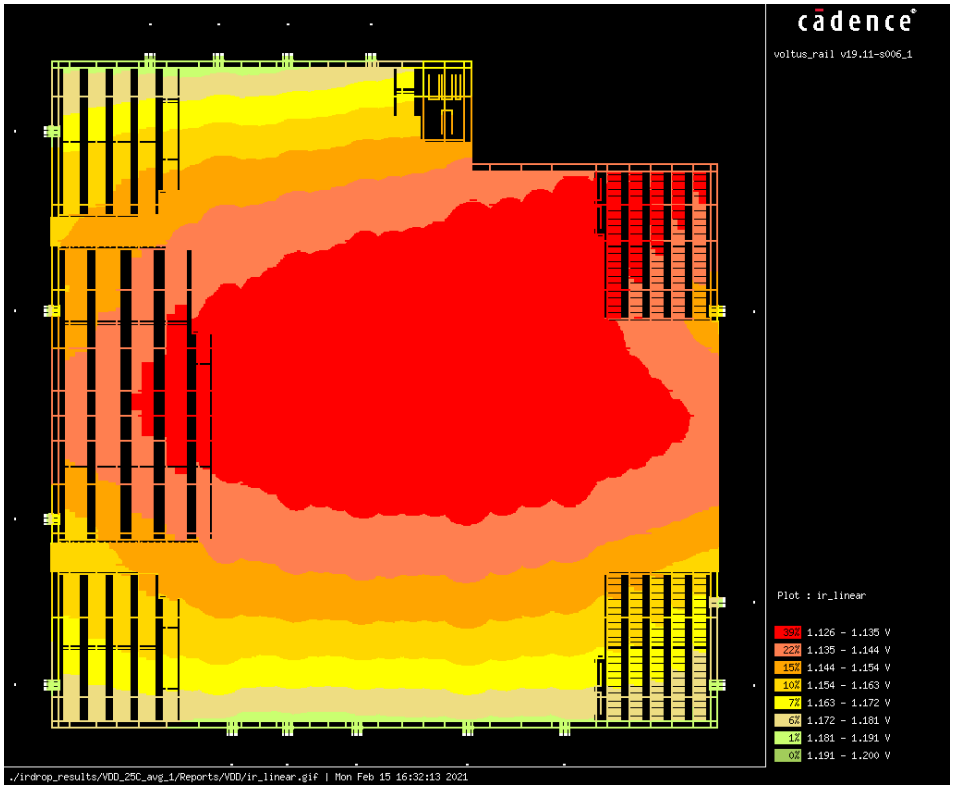

Fig. 6. IR drop analysis results for a core clock frequency of 600 MHz.

At this moment the bringup is still in process. The following functional tests have been applied:

- PCB initial test by checking power supplies and reset control.
- We tested the VPU standalone by inserting the Vector micro-operations through the JTAG debugger without using the scalar core, and it passed the test.
- Using a external clock of 50 MHz generated in the FPGA, UART and SPI in the FPGA were successfully accessed, running an SDCARD test. We can conclude that the ASIC and the packetizer interconnection are stable when the system works at 50 MHz.
- We used the PLL as the main clock and it was observed that the DRAM RW test is correctly performed up to up to 94 MHz PLL frequency. Simpler “hello world” tests also were performed, working up to 99.9 MHz.

VII. Conclusions and Future Work

The DVINO processor is an experimental vector processor based on RISC-V. The objective of this design was to provide high performance computing capabilities to a relatively simple processor. This is another example that shows how RISC-V open ISA opens great opportunities for educational, research and industry organizations to work on open processor based developments while opening many potential collaborations among these communities.

The scalar core presents some improvement with respect to its predecessor and a performance in line with comparable academic processors such as Ariane. Recent improvements in the architecture will allow an increase in performance in subsequent iterations.

The DVINO design includes analog IPs, which are critical in system-on-chip designs and very difficult to obtain as open-source. They were developed as part of the DRAC project effort in the TSMC 65nm with the PDK provided by Europractice. One of them is a PLL to be able to generate internal clock frequencies beyond what can be achieved by an external oscillator. In this design, it was decided to integrate the analog blocks as part of their own padring segment, separated from the digital padring. This block was custom designed. This offers the advantage of a tighter control of the interconnects to the pads. In future designs this approach may not be necessary.

Acknowledgments

The DRAC project is co-financed by the European Union Regional Development Fund within the framework of the ERDF Operational Program of Catalonia 2014-2020 with a grant of 50% of total eligible cost. The authors are part of Red-RISCV

which promotes activities around open hardware. The Lagarto Project is supported by the Research and Graduate Secretary (SIP) of the Instituto Politécnico Nacional (IPN) from Mexico, and by the CONACyT scholarship for Center for Research in Computing (CIC-IPN).

REFERENCES

[1] Andrew Waterman, Yunsup Lee, David A. Patterson, and Krste Asanovic. "The RISC-V Instruction Set Manual, Volume I: Base User-Level ISA". In *Tech. Report UCB/EECS-2011-62, EECS Dept., UC Berkeley*, pages 97–116, 2011.
[2] RISC-V International. https://riscv.org/. [Online; accessed 23-April-2020].
[3] "A European Chips Act Communication". https://digital-strategy.ec.europa.eu/news-redirect/734921, 8 February 2022.
[4] "Designing RISC-V-based Accelerators for next generation Computers". https://drac.bsc.es/, 2019-2022.
[5] Jaume Abella et al. "An Academic RISC-V Silicon Implementation Based on Open-Source Components". In *XXXV Conference on Design of Circuits and Integrated Systems (DCIS)*, pages 1–6, 2020.
[6] European Processor Initiative. https://www.european-processor-initiative.eu/. [Online; accessed 23-April-2020].
[7] NEC. SX-Aurora TSUBASA. https://www.nec.com/en/global/solutions/hpc/sx/architecture.html.
[8] Semidynamics. OVI: Open Vector Interface. https://github.com/semidynamics/OpenVectorInterface.
[9] RISC-V V-extension. https://github.com/riscv/riscv-v-spec.
[10] lowRISC. "Untethered lowRISC v0.2". https://www.lowrisc.org/docs/untether-v0.2/. [Online; accessed 27-August-2019].
[11] Mark S. Papamarcos and Janak H. Patel. "A Low-Overhead Coherence Solution for Multiprocessors with Private Cache Memories". In *Proceedings of the 11th Annual International Symposium on Computer Architecture*, ISCA '84, page 348–354, 1984.
[12] lowRISC. "Overview of the debug infrastructure". https://www.lowrisc.org/docs/debug-v0.3/overview/, 2018.
[13] The Open SoC Debug Contributors. "The Open SoC Debug Documentation Library". https://opensocdebug.readthedocs.io/en/latest/index.html#, 2018.
[14] Institute for Integrated Systems (LIS). "The Generic Logic Interfacing Project". https://www.glip.io, 2018.
[15] RISC-V International. RISC-V ISA tests. https://github.com/riscv-software-src/riscv-tests/tree/master/isa.
[16] UC Berkeley. UCB RISC-V Torture Test Generator. https://github.com/ucb-bar/riscv-torture.
[17] Google. RISC-V DV. https://github.com/google/riscv-dv.
[18] Florian Zaruba and Luca Benini. "The Cost of Application-Class Processing: Energy and Performance Analysis of a Linux-ready 1.7GHz 64bit RISC-V Core in 22nm FDSOI Technology". *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 27(11):2629–2640, 2019.