A Low-Power Low-Area SoC based in RISC-V Processor for IoT Applications

Ronaldo Serrano, Marco Sarmiento, Ckristian Duran, Khai-Duy Nguyen,
Trong-Thuc Hoang, Koichiro Ishibashi and Cong-Kha Pham
University of Electro-Communications (UEC), Tokyo, Japan
Email:{ronaldo, marco, duran, khaiduy, thuc}@vlsilab.ee.uec.ac.jp, ishibashi@ee.uec.ac.jp, phamck@uec.ac.jp

2021 18th International SoC Design Conference (ISOCC) | 978-1-6654-0174-6/21/$31.00 ©2021 IEEE | DOI: 10.1109/ISOCC53507.2021.9613880

***Abstract*—The IoT applications use embedded processors to execute lightweight tasks for sensing and management of communications, using different energy harvesting strategies. However, many IoT applications need a low-power consumption for the limitation of power supplies. This paper presents a low-power low-area System On a Chip (SoC) for IoT applications with a stable power supply. The SoC consists of a microprocessor, a 1-KB of Static Random Access Memory (SRAM), a debug module, a timer, a General-Purpose In-Outs (GPIO), and a Serial Peripheral Interface (SPI) programmer. The processor uses a RISC-V Instruction Set Architecture (ISA). The implementation is fabricated in 0.18μm CMOS General Purpose (GP) technology, occupies a 750μm x 536μm. The microprocessor represents only 7.6% of the area of all SoC. The measures denote a 2.17μW with a 1V of supply voltage and 32KHz operating clock frequency.**

***Index Terms*—RISC-V, SoC, Low-Area, Low-Power, IoT.**

I. Introduction

The sensor nodes in the Internet-of-Things (IoT) are using in several fields, like biomedical, healthcare, and agriculture applications. These applications often do not require a high-performance execution because the tasks are only limited to sense, store and sent data. However, these applications demand a low-power consumption by the limitation of accessing power supplies. Besides, the energy harvesting is not enough to supply the power in active modes. In this way, these IoT applications need a low-power System On a Chip (SoC) to manage the different sensors and the networking capabilities. In this work, we implement a low-power low-area SoC using a RISC-V ISA on a 0.18μm GP CMOS technology. The implementations present a 2.17μW of power consumption while operating in sub-MHz frequency for IoT applications. Besides, the SoC implemented occupy a 750μm x 536μm.

II. SoC

SERV is RISC-V Core development by O. Kindgren [1]. The unique feature of this processor is the use of a serial datapath, which means that the internal datapath is one bit wide. For each instruction, data is read from the register file or the immediate fields of the instruction word, and the result of the operation is stored back into the register file one bit at a time. This architecture significantly reduces the area and makes the SERV one of the smallest RISC-V core implementations in the world. The SERV uses 32 bits for addressing, and it includes the Integer extensions with some of the privilege specifications.

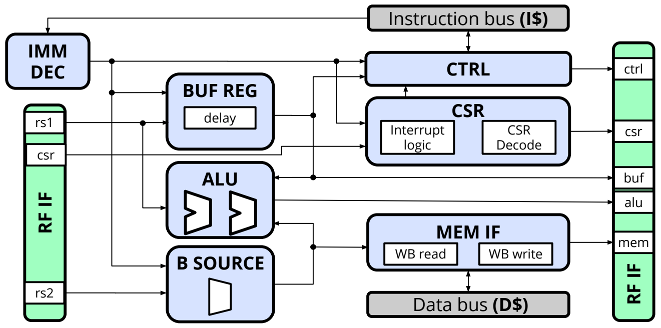

Fig. 1. Datapath of the SERV processor.

Fig. 1 illustrates the datapath of the SERV processor. The instruction life cycle in the SERV processor starts when the core is issuing a request for a new instruction on the Instruction Bus, waiting for the response of the memory to presents the new instruction. The opcode, funct3, and immediate value are saved in the *IMM_DEC*. The saved bits of the instruction are then decoded to create the internal control signals that correspond to the current instruction. A stage in SERV is 32 consecutive cycles, during which the core is active and processes the input, producing a bit-to-bit output, starting with the LSB. The processor distinguishes between two and one-stage instruction. The two-stage operation is all jump, shift, slt, and load-store instruction. The remaining instructions are one-state. In the new instruction, the processor checks the number of stages of the instruction. If the instruction is one-stage, the *RF_IF* highlighted in green is set in read-write mode. When the instruction is two-stage, the *RF_IF* is set only in read mode to check the alignment exception, and then put the *RF_IF* in write mode. In the case of the alignment exception exists, a trap is called. Finally, the PC has been updating with the following address of the next instruction.

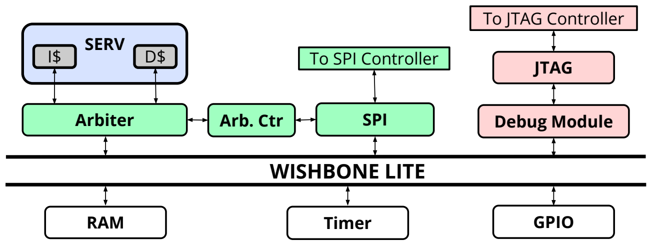

Fig. 2. Block diagram of the MCU implemented.

Fig. 2 illustrates the block diagram of the implemented MCU. The MCU consist of a bit-serial RISC-V processor (SERV) [1] and a 1-KB RAM, a GPIO, a Timer, a JTAG-based debug module, and SPI. An Arbiter is used to programming the MCU using the SPI interface. The bus implemented in the SoC is a lite version of the Wishbone bus [2].

III. Measurement Results

Fig. 3 shows the ROHM 0.18μm micrograph that includes the SoC. The SoC implementation occupies a 750μm x 536μm area. Table I illustrates the implementation results of the SoC. The microprocessor represents 7.59% of all SoC. The JTAG-based debug module and the RAM occupies the majority area of the SoC with 35.97% and 31.66%, respectively. The Wishbone bus shows a tiny portion, about 1.41% of the SoC.

Fig. 4 illustrates the percentage of the power consumption of the implemented SoC. The leakage highlighted in red represents the power means the system is without any signal and clock. The dynamic power highlighted in blue represents the power of the SoC when given signal and clock but running at an idle state. The active power highlighted in green is measured when the system process program. The SoC has a 2.17μW of power consumption with 1V of supply voltage and 32KHz operating clock frequency, representing 0.64% of leakage, 96.12% of dynamic, and 2.58% of active power. Table II compiles the comparison of the area with other SoC based on the RISC-V processor. The area occupied by the SoC with the debug module is less than the other implementations without debug module. The area of the processor is 5.6 times smaller than other works in RISC-V.

TABLE I
ASIC implementation results

	Area		Gate-count
	μm^2	%	(NAND2)
SoC	348,604.06	100	27,019
SERV processor	26,459.59	7.59	2,051
Debug module	125,392.87	35.97	9,718
Wisbone bus	4,950.17	1.42	384
RAM	110,368.04	31.66	8,554
SPI	51,314.51	14.72	3,977
GPIO	21,648.31	6.21	1,678
Timer	8,471.07	2.43	657

TABLE II
Area Comparison Results

		Area SoC [μm^2]	Area core [μm^2]	KGE core	Technology
This work		403,085	26,460	2.05	ROHM GP 0.18μm
[3]		672,146	—	—	XFAB GP 0.18μm
[4]		349,233	120,776	—	TSMC GP 0.13μm
[5]	Zero	—	27,216$^+$	18.9	UCM GP 65nm
	Micro	—	16,704$^+$	11.6	

$^+$ Area estimate. In UCM 65nm [5], gate area of NAND2 = 1.44 μm^2.
— Not reported.

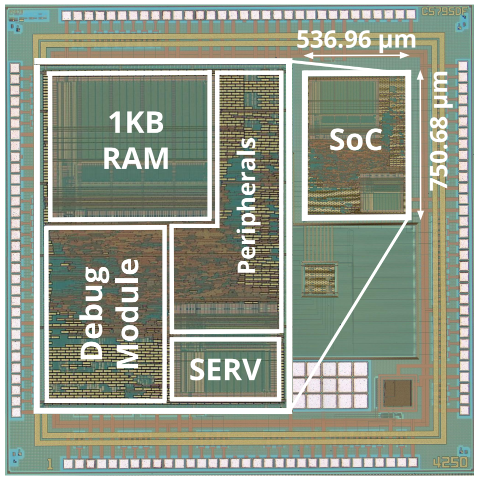

Fig. 3. ROHM 0.18μm SoC chip micrograph.

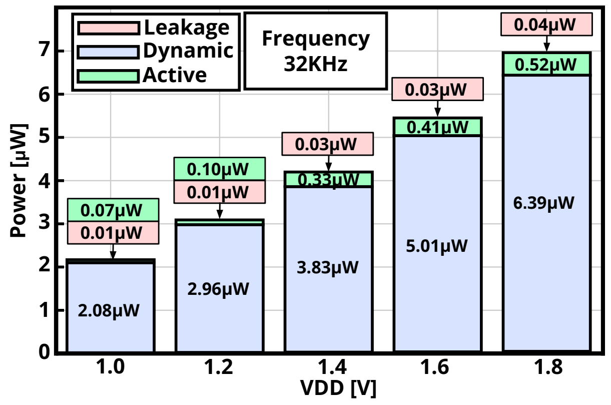

Fig. 4. Power consumption of the SoC implemented.

IV. Conclusion

A low-power low-area SoC for IoT applications is presented in this paper. The SoC contains a smaller RISC-V core, a 1-KB RAM, a GPIO, a Timer, a debug module, and SPI. The implementation report a 2.17μW of power consumption with 1-V of voltage supply and 32KHz operating clock frequency. The SoC occupies a 750μm x 536μm in 0.18μm GP CMOS technology. The microprocessor represents only 7.6% of the area of all SoC.

Acknowledgment

This work was supported by JST-CREST Grant Number JPMJCR16Q1.

References

[1] O. Kindgren, "SERV - The SErial RISC-V CPU," https://github.com/olofk/serv, 2018.
[2] R. Herveille *et al.*, "WISHBONE system-on-chip (SoC) interconnection architecture for portable IP cores," *OpenCores Organization*, 2002.
[3] R. Garcia-Ramirez *et al.*, "Siwa: a RISC-V RV32I based Micro-Controller for Implantable Medical Applications," in *2020 IEEE 11th Latin American Symposium on Circuits Systems (LASCAS)*, 2020, pp. 1–4.
[4] C. Duran *et al.*, "A 32-bit RISC-V AXI4-lite bus-based microcontroller with 10-bit SAR ADC," in *2016 IEEE 7th Latin American Symposium on Circuits Systems (LASCAS)*, 2016, pp. 315–318.
[5] P. Davide Schiavone *et al.*, "Slow and steady wins the race? A comparison of ultra-low-power RISC-V cores for Internet-of-Things applications," in *2017 27th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, 2017, pp. 1–8.