A 32-bit RISC-V AXI4-lite bus-based Microcontroller with 10-bit SAR ADC

Ckristian Duran, Luis Rueda D., Giovanny Castillo, Anderson Agudelo, Camilo Rojas,
Luis Chaparro, Harry Hurtado, Juan Romero, Wilmer Ramirez,
Hector Gomez, Javier Ardila, Luis Rueda, Hugo Hernandez, Jose Amaya and Elkim Roa
Design Group of Integrated Systems CIDIC, Universidad Industrial de Santander, Bucaramanga, Colombia
ckristian.duran@correo.uis.edu.co, efroa@uis.edu.co

***Abstract*—In this paper a complete implementation and design of a fully-synthesized 32-bit microcontroller in a 130nm CMOS technology is presented. This is the first microcontroller featuring the open source RISC-V instruction set all mounted through AXI4-Lite and APB buses for communication process. The microcontroller contains a 10-bit SAR ADC, a 12-bit DAC, an 8-bit GPIO module, a 4kB-RAM, an SPI AXI slave interface for output verification, and an SPI APB slave interface for checking the correct behavioral of the APB bridge. All peripherals are controlled by a RISC-V and an SPI AXI master interface that is used for programming the device and checking the data flowing through all the slaves. A total power density is reported as 167μW/MHz and the area for this RISC-V microcontroller has a reduced footprint of 798μm$\times$484μm.**

I. INTRODUCTION

Many commercial microcontrollers have been built to suit different requirements featuring many private-and-licensed instruction sets. Licensed instruction sets and microprocessor cores restrict the process of modifying the core for different purposes such as improving performance and adapting it to specific applications.

RISC-V is a new open ISA aimed to support architecture research and education [1]. It is fully available to the public and can replace ARM microprocessors because is comparable fast and has a small footprint size. RISC-V features an instruction set easy to migrate empowering a developer community to work on this new architecture. Recently, in [2] and [3] processors are reported capable of running linux using a RISC-V instruction set. However, up to date there has not been any reported work with of a small footprint RISC-V core for low-power microcontroller applications.

We are presenting the implementation of the first reported 32-bit RISC-V based microcontroller (mRISC-V). Our implementation and design is equivalent to commercial microcontrollers implemented with an ARM-M0 core. Features of this microcontroller are: AXI4-Lite and APB buses for interfacing communication between the core and all the peripherals attached to it, a 4kB-RAM, Serial Peripheral Interface (SPI) slaves for output, a GPIO module, a SAR Analog-to-Digital converter and a Digital-to-Analog converter. The circuit was designed using 130nm CMOS technology and tested using an efficient test algorithm in bus and using RISC-V toolchain. In addition, a master SPI has been implemented in order to explore all the status of the peripherals while the microprocessor is still executing its program.

II. RISC-V

RISC-V is a new open instruction set architecture (ISA) designed by the Berkeley Architecture Group with the aim to support architecture research and education [1]. RISC-V is fully available to public and has advantages such as a smaller footprint size, support for highly-parallel multi-core or many-core implementations [2], variable-length instructions to support an optional dense instruction, and energy efficient. Moreover, RISC-V presents improvements in different characteristics over another open ISAs as shown the comparison in the Table I.

TABLE I
COMPARISON OF THE OPEN ISA

PARAMETER/ISA	SPARCV8	OpenRISC	RISC-V
BASE+EXT	NO	NO	YES
Compact Code	NO	NO	YES
Quad FP	NO	NO	YES
32-bit	YES	YES	YES
64-bit	NO	YES	YES
128-bit	NO	NO	YES
GCC	YES	YES	YES
LLVM	YES	YES	YES
32-bit	YES	YES	YES

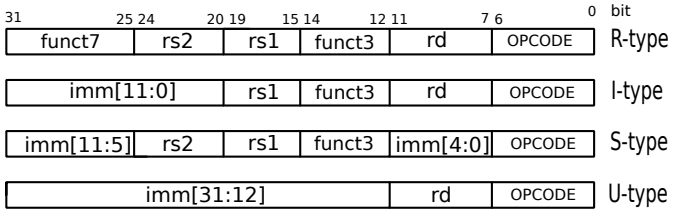

Fig. 1. RISC-V base instruction formats

The base ISA is clean and suitable for direct hardware implementation, the instructions of RISC-V are similar to other RISC instructions set such as OpenRISC. The Fig. 1 shows the four core instruction formats (R,I,S,U). R-type format is used for several arithmetic instructions with one or two source operands and for the atomic memory operation (AMO) instructions that perform read-modify-write operations for multiprocessor synchronization. In addition, R-type is used

ISBN 978-1-4673-7835-2/16/$31.00©2016 IEEE IEEE Catalog Number CFP16LAS-ART

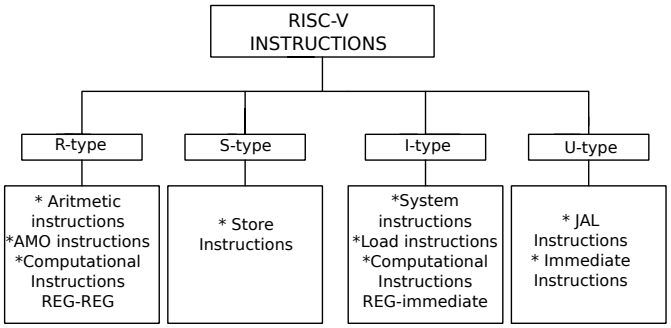

Fig. 2. RISC-V Instructions

for computational instruction register-register. I-type format is used for system instructions to access system functionality that might require privileged access and computational instructions register-immediate. Load and store instructions transfer a value between the registers and memory. Loads are encoded in the I-type format and stores are S-type. U-type format is used for JAL instructions or immediate instructions as LUI (load upper immediate) and AUIPC (add upper immediate to pc), see Fig. 2. All formats are fixed 32-bit in length, and keeps the source (rs1 and rs2) and destination (rd) registers at the same position to simply decoding, for all formats the bit 31 is the sign bit.

RISC-V architecture contains an Arithmetic Logic Unit (ALU), registers, past memory and future memory, an interrupt system, an instruction decoder and a Pico co-processor interface (PCPI) which is connected to a co-processor to do multiplications. We have implemented a reduced instruction set microcontroller mRISC-V, described in verilog, using a smaller core to promote research and development of the internet of things. According to [1], the implemented sub-set of instructions are RV32I Base Integer Instruction Set (all) and "M" Standard Extension for Integer Multiplication and Division (only MUL[H[SU—U]]).

III. mRISC-V ARCHITECTURE

Implementation of an efficient microcontroller requires a reliable and fast communication between masters and slaves blocks in the microcontroller. Nowadays, many bus-based communication architecture standards are found. In this work we are using the AMBA and APB protocols such that we can compare with microcontrollers based on ARM-M0 cores. The architecture of the mRISC-V is shown in Fig. 3.

Slave interfaces are interconnected to the AXI4-Lite or to the APB Bridge. Each interface is different but with several similarities for each protocol. Buses are composed of state machines, registers, multiplexers and Hi-Z buffers. Moreover, the master interface is implemented with a Serial Peripheral Interface (SPI). Following sections describe all these interfaces.

A. AXI4-Lite

The AMBA AXI4 is an ultra-high performance protocol bus standard [4] developed by ARM for easy application in small scale SoCs. AXI4 has different forms to be implemented and this work uses the AXI4-Lite protocol, which has two

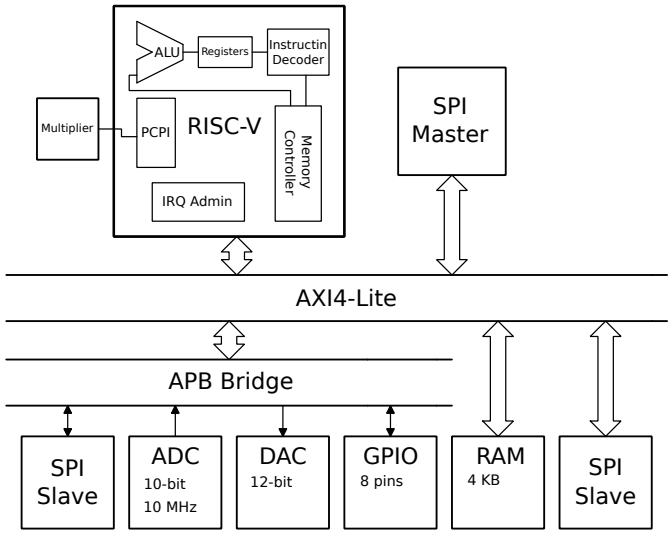

Fig. 3. mRISC-V block diagram.

data channels of 32-bit and other necessary control signals for communication between Masters (SPI_M, RISC-V) and slaves (RAM, SPI_S, APB) [5].

According to the AXI4 specification, communication between the masters and the slaves must be done through an Interconnect module. For this application the AXI4-Lite interconnect has been implemented following specification.

B. SPI Master

SPI is used as a master AXI4-Lite interface for controlling all slaves attached to the core. This interface has a 66-bit data instruction: 32-bit for data, 32-bit for address and 2-bit to define an action like write, read; to put the core reset; and check the last request. This is a fully custom design for debugging and programming purposes.

C. APB

Several types of protocols are available in SoC, which require a bridge to safely pass the information from one type of protocol to another without data loss. The Advance Peripheral Bus (APB) is part of the Advaced Microcontroller Bus Architecture (AMBA) protocol family. This protocol determines a low-cost interface that is optimized for minimal power expending and decreased interface complexity used to connect to low-bandwidth peripherals that do not demand the high performance of the AXI protocol.

The signal transaction between AXI master and APB slave are described by:

- AMBA AXI-Lite signals as described in the AMBA AXI-Lite 4.0 protocol specification. [5]
- AMBA APB signals as described in the AMBA APB 4.0 protocol specification. [6]

The APB bridge provides an interface between the high-speed AXI domain and the low-power APB domain. It's seen as slave on AXI but as a master on APB. To run a process of writing or reading transfers on the AXI bus

ISBN 978-1-4673-7835-2/16/$31.00©2016 IEEE

IEEE Catalog Number CFP16LAS-ART

are converted into corresponding transfers on the APB. This conversion is described in verilog applying the respective protocol specifications. Since its execution is not pipelined, wait states and response signals of protocol are added during transfers to and from the APB when the AXI must wait for the APB protocol.

The APB bridge is responsible for converting the APB signals corresponding AXI signals. The APB bridge acts as master in APB module and all transactions initiated by the AXI masters. Whenever AXI master tries to access a slave, it requires completing the handshaking process with the corresponding slave. When the AXI master wants to start a process of writing and reading simultaneously in any of the peripherals, reading takes priority and the writing process can take place after completion of the read transaction.

IV. PERIPHERALS

A. GPIO

General-purpose input/output (GPIO) are crucial for a variety of microcontroller applications and these pins are used as digital inputs or outputs. Figure 4 shows the block diagram of communication between the core and the pad, which perform the connection of the GPIO with the APB bridge protocol. This block is controlled by a digital control system designed for speed and low power consumption according to the APB Bridge protocol. The implemented GPIO have slew-rate control capability for output and Schmit-Trigger windows for input. The 8-port GPIO is able to drive up to 25mA per output pin.

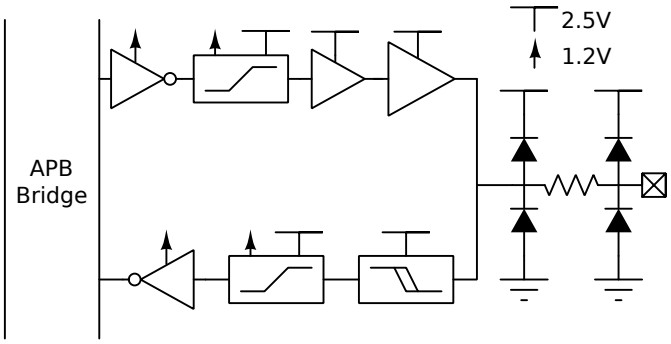

Fig. 4. GPIO block diagram.

B. ANALOG AND DIGITAL CONVERTERS

In order to perform analog and digital conversions, mRISC-V incorporates an analog-to-digital converter (ADC) and a digital-to-analog converter (DAC) which communicate with APB Bridge through the interface shown in Fig. 3. The ADC type that is implemented is a successive approximation register (SAR) and its structure is shown in Fig. 5. The SAR ADC implemented operates at a maximum sampling frequency of 10MHz and a resolution of 10-bit for a differential input.

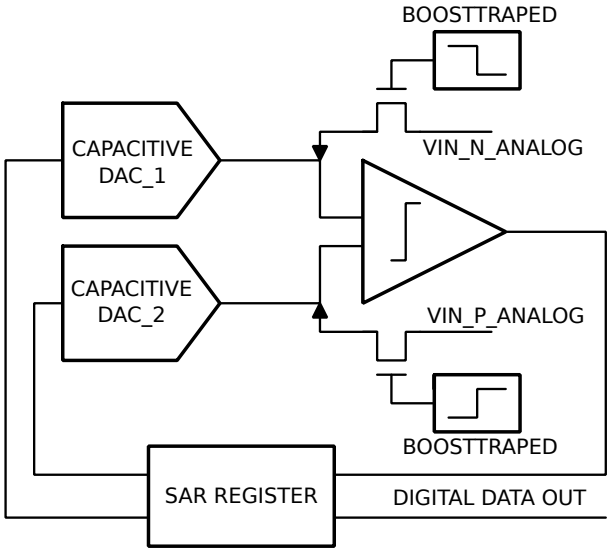

Fig. 5. SAR ADC block diagram.

On the other hand, the DAC implemented is based in a R2R structure with 12-bit resolution, rail to rail output voltage and a typical settling time of 100ns.

V. RESULTS

Verification of peripherals has been performed from the perspective of the RISC-V core and the SPI master. The SPI master is used to program the memory and also can read or write on any peripheral attached to the AXI-4 bus and the APB bridge. Fig. 6 shows the used initialization and verification tasks where signal generation and communications transactions are shown in Fig. 7.

A methodology is described in verilog which perfom an automatic verification of random handshake generation in the AXI master 1 and AXI master 2. As the first step, the AXI master choose a slave, then the transaction type is selected (read or write) depending of type transaction that can support the slave. The data is sent to the bus and the AXI scoreboard registers of the data flow between the master and the slave. The data is kept and compared to verify that the transaction is correct. In case of a violation in the protocol, the verification process is paused and the error is reported to the prompt.

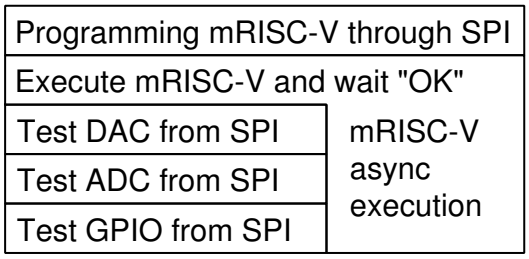

Fig. 6. Initialization and testing setup for mRISC-V.

Microcontroller is fully synthesized in 130nm CMOS technology. Synthesis results are shown in the Table II for each

ISBN 978-1-4673-7835-2/16/$31.00©2016 IEEE

IEEE Catalog Number CFP16LAS-ART

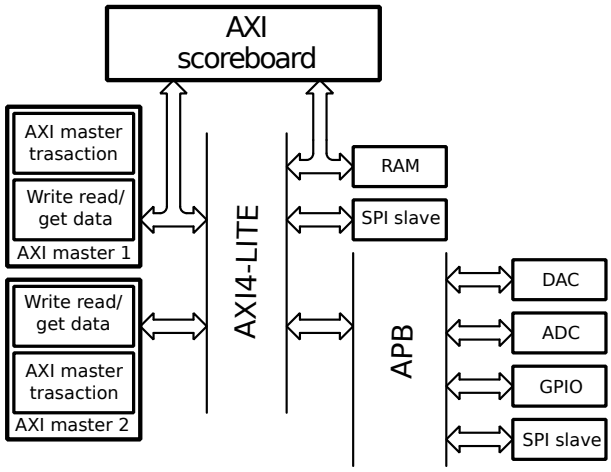

Fig. 7. Testbench architecture.

TABLE II
POWER, TIMING AND AREA BREAKOUT OF THE MRISC-V.

Core	Power [nW/MHz]	Time Slack [ps @ 100MHz]	Area [μm^2]
AXI-4 interconnect	5284.66	6093	11830
mRISC-V	96952.67	2143	120776
SPI AXI master	13532.69	3998	19627
AXI-RAM	2617.00	3602	2580
RAM	18997.73	N/A	168708
APB TOP	11703.14	3664	23794
SPI AXI slave	2176.93	8085	1899
All	166841.47	1185	349233

peripherals and for whole system with the AXI-APB implementation. The 4kB RAM module occupies almost the same area of the sum of the core and peripherals. The highest power consumption density comes from the RISV-V processor. Maximum operation frequency is determined by the RAM which operates at 100MHz despite the core been able to operate at higher frequency. Some cores, especially the AXI-4 interconnect, uses Hi-Z addressing instead of multiplexer to optimize area. The sum of the peripherals implemented on APB (ADC, DAC and GPIO) are on the APB TOP implementation.

The final layout is exposed in the Fig. 8. Each instance is highlighted to expose the area breakout. As expected, the RAM block occupies a significant area with a footprint close to 50% of the whole chip. A final area of 798μm×484μm and an energy consumption of about 167μW/Hz shows the feasibility of using the proposed mRISC-V -with additional sensor circuitry- in and low-cost and low-power applications.

VI. SUMMARY

A fully-synthesized microncontroller based on RISC-V architecture on 130nm CMOS technology has been presented. Many peripherals are included using a proposed methodology to verify the correct operation. The proposed architecture shows the interconnection between the RISC-V, the SPI AXI master, and all the peripherals attached to the AXI4-Lite and APB buses, explaining details of the implementation.

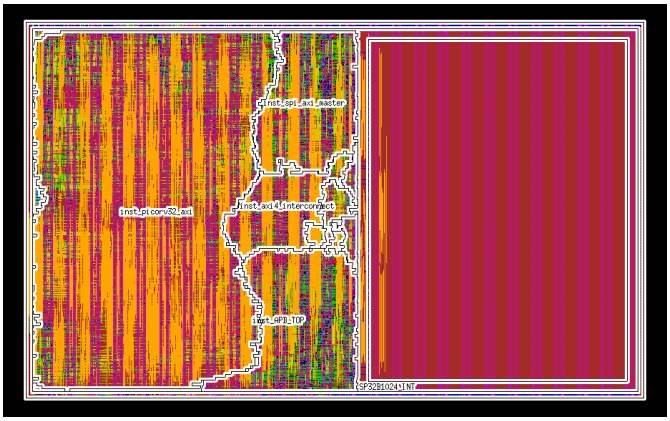

Fig. 8. Final layout for AXI-APB implementation. Area:798μm×484μm

The proposed mRISC-V is the first designed RISC-V microcontroller with enough peripheral to perform common microcontroller tasks. Power and area results show that a reduced RISC-V architecture can be used to replace ARM-MO based microcontrollers with similar peformance. Considering the advantage of the growing RISC-V community and the existing tool-chain and software around this new instruction set, the mRISC-V paves the way of future implementations for specific and general applications in the world of IoT with open source devices.

REFERENCES

[1] A. Waterman, Y. Lee, D. A. Patterson, and K. Asanovi, "The RISC-V instruction set manual, volume I: User-level ISA, version 2.0," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS- 2014-54, May 2014.

[2] Y. Lee, A. Waterman, R. Avizienis, H. Cook, C. Sun, V. Stojanovic, and K. Asanovic, "A 45nm 1.3GHz 16.7 double-precision GFLOPS/W RISC-V processor with vector accelerators," in *European Solid State Circuits Conference (ESSCIRC), ESSCIRC 2014 - 40th*, pp. 199–202, Sept 2014.

[3] B. Zimmer, Y. Lee, A. Puggelli, J. Kwak, R. Jevtic, B. Keller, S. Bailey, M. Blagojevic, P.-F. Chiu, H.-P. Le, P.-H. Chen, N. Sutardja, R. Avizienis, A. Waterman, B. Richards, P. Flatresse, E. Alon, K. Asanovic, and B. Nikolic, "A RISC-V vector processor with tightly-integrated switched-capacitor DC-DC converters in 28nm FDSOI," in *Proc. Symposium on in VLSI Circuits (VLSI Circuits),* pp. C316-C317, June *2015.*

[4] S. Pradeep and C. Laxmi, "Design and verification environment for AMBA AXI protocol for SoC integration."

[5] ARM, "AMBA AXI and ACE protocol specification," 2011, pp. 1–121.

[6] C. Ma, Z. Liu, and X. Ma, "Design and implementation of apb bridge based on amba 4.0," in *2011 International Conference on Consumer Electronics, Communications and Networks (CECNet),* pp. 193–196, April 2011.