Article

Journal Applied Computing
December, 2023 Vol.7 No.21 1-9

Debugging and programming a RISC-V processor, using the IEEE 1149.1 standard

Depuración y programación de un procesador RISC-V, usando el estándar IEEE 1149.1

RICO-GARCÍA, Paulina[1]†*, YAÑEZ-VARGAS, Juan Israel[1], PARRA-MICHEL, Ramón[2] and CONDE-ALMADA, Ernesto[2]

[1]*Universidad Politécnica de Juventino Rosas, Maestría en Ingeniería con Especialidad en Sistemas Inteligentes*
[2] *CINVESTAV Unidad Guadalajara*

ID 1st Author: *Paulina, Rico-García /* **ORC ID**: 0000-0003-2540-6309

ID 1st Co-author: *Juan Israel, Yañez Vargas* / **ORC ID**: 0000-0001-5749-8442

ID 2nd Co-author: *Ramón, Parra-Michel* / **ORC ID**: 0000-0003-2327-2482

ID 3rd Co-author: *Ernesto, Conde-Almada* / **ORC ID**: 0009-0008-4880-408X

DOI: 10.35429/JCA.2023.21.7.1.9 Received July 10, 2023; Accepted December 10, 2023

Abstract

RISC-V processors are an open source ISA (Instruction Set Architecture), which means that anyone can design and modify them to meet the specific objectives of any development. Thanks to the flexibility it brings with it, it is possible to add a debugging and programming logic unit to this processor to get a real sense of how the processor works, a unit that will be key to identifying errors during the design of the processor, as well as errors that may occur during the manufacturing process. This article presents the methodology used to incorporate hardware that allows the processor to be debugged, using the IEEE 1149.1 standard as a reference.

JTAG, IEEE 1149.1, RISC-V, Boundary Scan Chain, Processor, Register, Debugging, FTDI, FPGA, Hardware, RTL, Methodology, Driver, Development, Verification, Flexibility, Technology and Innovation, Incorporate, Modify

Resumen

Los procesadores RISC-V son una ISA (Instruction Set Architecture) de código abierto, lo que significa que cualquiera puede diseñarlos, así como modificarlo para cumplir con los objetivos particulares de cualquier desarrollo. Gracias a la flexibilidad que trae consigo es posible sumar a este procesador una unidad lógica de depuración y programación que permita tener una noción real del funcionamiento del procesador, unidad que será clave para la identificación de fallas durante el diseño del procesador, así como defectos que puedan surgir en el proceso de fabricación. En el presente artículo es presentada la metodología empleada para la inserción de hardware que permita la depuración del procesador usando como referencia el estándar IEEE 1149.1.

JTAG, IEEE 1149.1, RISC-V, Cadena de exploración de límites, Procesador, Registro, Depuración, FTDI, FPGA, Hardware, RTL, Metodología, Driver, Desarrollo, Verificación, Flexibilidad, Tecnología e Innovación, Incorporar, Modificar

Citation: RICO-GARCÍA, Paulina, YAÑEZ-VARGAS, Juan Israel, PARRA-MICHEL, Ramón and CONDE-ALMADA, Ernesto. Debugging and programming a RISC-V processor, using the IEEE 1149.1 standard. Journal Applied Computing. 2023. 7-21: 1-9

* Corresponding autor: (e-mail: m21030004@upjr.edu.mx)
† Researcher contributing as first author.

Introduction

Integrated circuits, present in countless devices that are now part of everyday life, are an area of development where there is much to be done. Beyond the fact that they are already everywhere, innovation and the creation of different solutions make this a niche of constant change.

A few years ago, technological development was reserved for powers with technology and knowledge so hermetic that it could only be accessed by being part of the industry. Today there are open source initiatives that make these development possibilities more accessible.

Thanks to the latter, technology design is possible, with well-established development processes that allow to go from requirements gathering to having the hardware assembled in an FPGA or as a finished ASIC.

Within the design and manufacturing flow of any technology it is important to keep in mind that identifying errors in early stages will represent less waste of any kind of resources. Hence the importance of having a debug and programming unit that allows greater control over the hardware at the simulation level during the functional tests that are performed at the design stage or physical tests that help to detect any defect in later stages of manufacturing where it is possible to identify in the silicon dice any physical defect, this helps to generate a certain level of certainty that the user will receive a complete product.

Standards such as JTAG are used to test the correct functioning of any circuit; thanks to the insertion of boundary scan chains, a TAP controller, as well as instruction registers, bypass registers, etc., it is possible to monitor the behaviour of a circuit with the help of different instructions such as SAMPLE, PRELOAD, EXTEST or BYPASS, etc., which makes it possible to identify faults or divergences from the initial functional objective of the design. This without the strict need to have access to the primary inputs and outputs of the circuit, which contributes to the intervention methodology being a more generic process.

Literature Review

Circuit design is an area with a lot of work to do, thanks to the presence of circuits in practically any electronic device, the contribution of new methodologies or solutions is constant. Some of the references related to this research and that contribute to the knowledge for its development are presented below. The reference (Shan Gao *et al* 2021.) proposes a fast on-chip debug design using the JTAG interface applied to a RISC-V processor. The novelty of this paper is the integration of a debug bus to reduce data input by switching data from serial to parallel. This is aimed at reducing the amount of input data. This makes it a less exhaustive operation.

In the article (Shan Gao *et al* 2022.) a debugging method adopting JTAG interface is presented to perform debugging functions for a RISC-V processor with IoT applications. This method was verified at RTL level by simulations and synthesised at gate level in 180nm libraries. The most important contribution of this methodology is that the most common debugging operations were simplified into instructions, which avoids a large data input through the shift register chain.

In this paper, an RTL-level verification methodology is proposed using the ARM debug interface (Ke, H. *et al* 2012.) where the ARM7TDMI SoC is taken as an example. An RTL simulation method using the ARM JTAG interface is proposed. Through EDA software (Synopsys VCS) the JTAG is tested to complete the code testing stage. To achieve this development, use is made of the ARM core, the EmbeddedICE logic controller circuit and the JTAG control circuit, which is composed of the TAP controller and scan chains.

Finally, a RED (Reused Embedded Debugger) design is presented where the JTAG interface is used. This model also has an additional hardware module (EICEM Embedded ICE Module) for more critical real-time debugging. The debugger proposed in the paper (Jung, D. *et al* 2003.) works as a stand-alone system through a PC interface, requiring no extra equipment cost which decreases the cost and time of testing the chip. The RED architecture is composed of three blocks; the boundary scan architecture, the boundary scan registers and EICEM.

Methodology

For the development of this research, a recursive methodology is proposed, working on three development blocks as shown in Figure 1; the starting block is the description of the hardware that will be added to the design to integrate the IEEE 1149.1 standard (IEEE Std 1149.1-2013) to the processor, in this sense, the requirements that the latter indicates as mandatory are contemplated as a minimum, in addition, certain elements developed specifically for the purpose of this research will be integrated. For example, dedicated registers, to monitor internal design registers that are of interest for analysis or debugging.

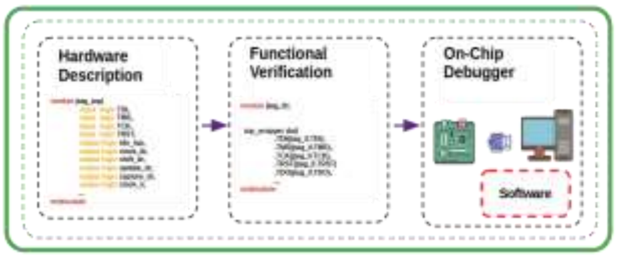

Figure 1 Methodology of research development
Source: Own work [Draw-LibreOffice]

Once the additional hardware is described, the correct functioning of the generated design will be verified. For this purpose, a test bed is developed to stress the design by sending different test vectors that, in addition to stimulating the design as such (without contemplating the JTAG hardware), stimulate the JTAG ports that allow verifying that this debugging mechanism complies with the requirements of the standard; in this development can be executed at least the instructions that the standard defines as mandatory that may or may not affect the operation of the design according to the instruction, as well as the instructions added for very specific purposes that were developed specifically to meet the objectives of this project.

All this to verify that at simulation level the design behaves as planned and with certainty to continue with the next stage of the proposed methodology. It is worth mentioning that we talk about recursion between the blocks of the methodology because the development is not a linear process (one stage is completely finished and the next one continues), because as certain modules and/or hardware components are described.

It is essential to verify that their operation is correct, since waiting to verify the final module could lead to many more connection problems or logical failures in the design. This would result in a more complex process to detect and solve them, which would increase the development time, as well as the consumption of other resources such as effort, money, etc.

For the simulations, a wide coverage is sought; where the design is stressed with typical scenarios and others that are not so typical so that this helps us to give certainty that under any circumstance the design will work as required or, failing that, to identify operating errors where the design does not comply with what is requested and thus make the relevant changes to solve the failure.

For the development of the third block of the methodology, once the design meets the requirements at simulation level, it is loaded into an FPGA card (as this is the most practical way to physically test a design before taking it to a silicon die or if this as such will not be implemented on a chip); once the FPGA is programmed with the design and this is stored in the memory of the latter is intended to communicate with it through the JTAG interface of an FTDI module.

Establishing this communication between devices allows data vectors to be inserted through the ports of the JTAG added to the design, so that it can be corroborated that the design works in the same way as it did with the simulation. This last phase, as well as serving as a functional verification, will be the communication methodology that will be used at the FPGA level for debugging and programming the processor, as sending and receiving data through the FTDI interface will finally allow the integration of the debugging and programming module in a practical way.

The proposed communication (FTDI-FPGA) was given by the characteristics of the board used (Figure 2).

The specifications of the devices mentioned are given below:

FPGA → *FPGA Lattice LFE5U-85*

FTDI → *FT2232H*

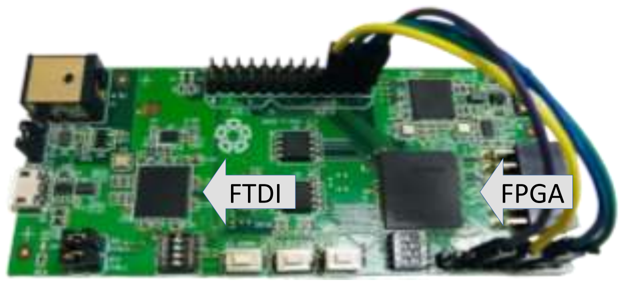

Figure 2 Card used for development
Source: Own work

Hardware Description

"JTAG insertion”

To add the JTAG interface to the core, we initially described and integrated the base hardware of the standard that encapsulates the processor as shown in Figure 3. Among the added elements are the boundary scan chain (cherry boxes), the state machine or TAP Controller, which is the core of JTAG operation, instruction decoder, instruction, bypass and device identification registers, as well as some multiplexers that allow the choice of the output according to the given instruction.

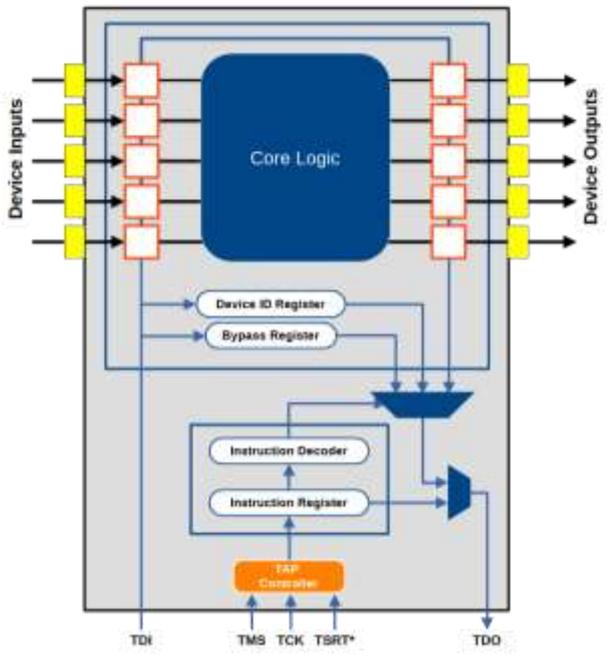

Figure 3 Base elements of JTAG
Source: Own work [Draw-LibreOffice]

Once we had this initial architecture and based on the requirements, we continued with the development of a second design stage, where the objective was to add the necessary logic to be able to scan the internal registers of the processor.

Based on the registers of interest, it was decided to take the elements of the different modules separately and generate an instruction for each one of them; with the aim of having as a result a serial output that concatenates the values of all the registers contained in each module, which allows us to have outputs that are not as long as if we did it with a single instruction for all the registers (which complicates the identification of any particular register) and also allows us to execute specific instructions that would avoid the use of the added hardware in sections that are not of interest at the moment. The architecture with these new registers and interconnections made to obtain the internal information of the processor is shown in Figure 4.

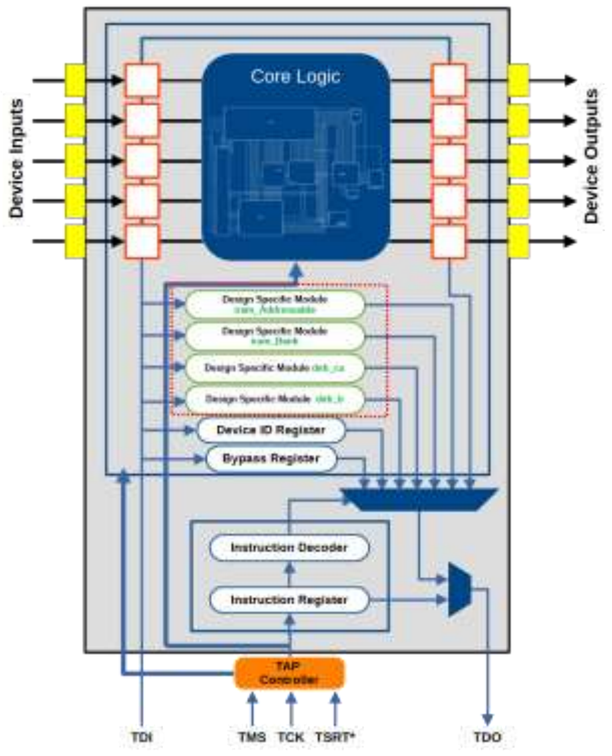

Figure 4 JTAG architecture added
Source: Own work [Draw-LibreOffice]

The red box shows the added registers, plus there are some interconnect changes that allow the results of the JTAG state machine to be taken to the processor to interact with the internal registers.

Functional Verification

In the functional verification phase, the aim is to create a simulation that stresses the design with stimuli that allow us to see how it behaves. In this sense, we worked with a base simulation where the processor is stimulated, with the objective that it is working or that it has information in the registers. Once the core has zeros and ones being processed, the internal registers are intervened to collect the information that is stored in each one of them at that moment.

This is for the aggregated instructions (red box - Figure 4). Similarly, instructions were tested where data is sent to the primary inputs through the boundary scan chain and relayed to the Logic Core for it to work with this information, or generic instructions such as bypass, for example.

On-Chip Debugger

For the communication between the Host computer and the design hosted in the FPGA, first the design is synthesised, routed and loaded to the FPGA, for this the tools indicated in Figure 5 were used.

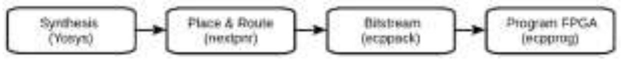

Figure 5 FPGA programming tools
Source: Own work [Draw-LibreOffice]

Once the design is loaded the objective is to establish communication between the computer and the latter, to stimulate it and monitor the results, the flow foreseen to establish this data exchange channel is as shown in Figure 6.

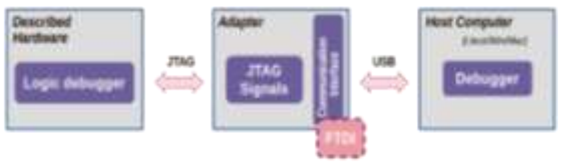

Figure 6 Information flow between computer and design
Source: Own work [Draw-LibreOffice]

The middle block in this diagram represents (Debug Adapter) the FTDI chip which has an integrated JTAG interface through which data will be sent to the interface integrated in the design, the connection made between these two devices is shown in Figure 7.

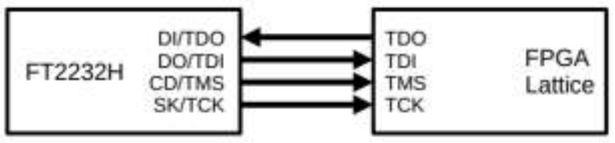

Figure 7 FTDI-FPGA interconnection
Source: Own work [Draw-LibreOffice]

The FTD2XX (D2XX FTDI) library and drivers from FTDI were used to complete the data transfer.

Tests and results

During the development of the hardware description, several simulations were carried out to corroborate that the functioning of the integrated elements was correct. For example, Figure 8 shows the scan execution of the registers of interest of one of the modules that are part of the processor.

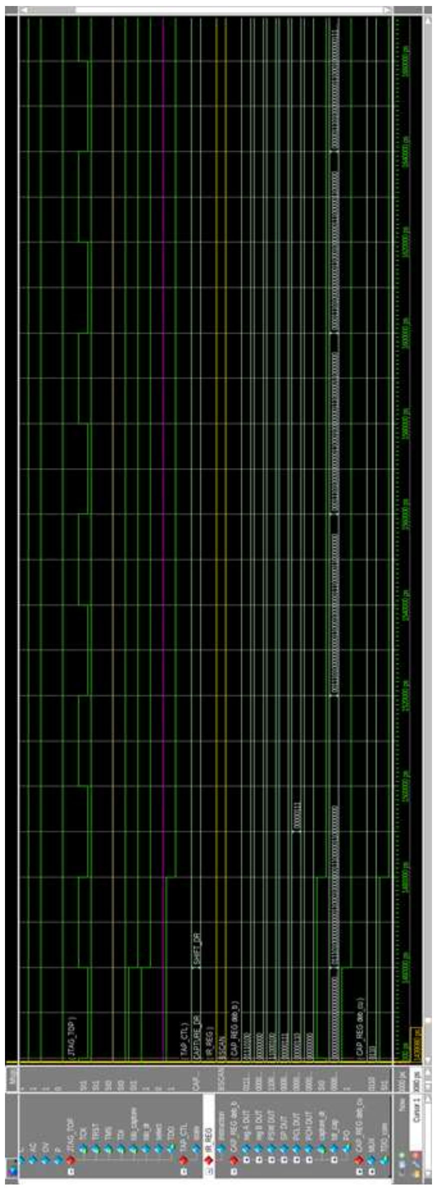

Figure 8 BSCAN instruction simulation
Source: Own work [ModelSim]

In the simulation we can see the registers of interest for the deb_b module in the group "CAP_REG deb_b", the content of these is concatenated and sent through "tdr_cap" to the TDO port of JTAG, in the image we can see how these bits are being displaced each time one of these is sent as output.

Figure 9 shows the same simulation time, but with a much more open view where we can see the output of TDO; which coincides with the bits that were captured and are being shifted to this port.

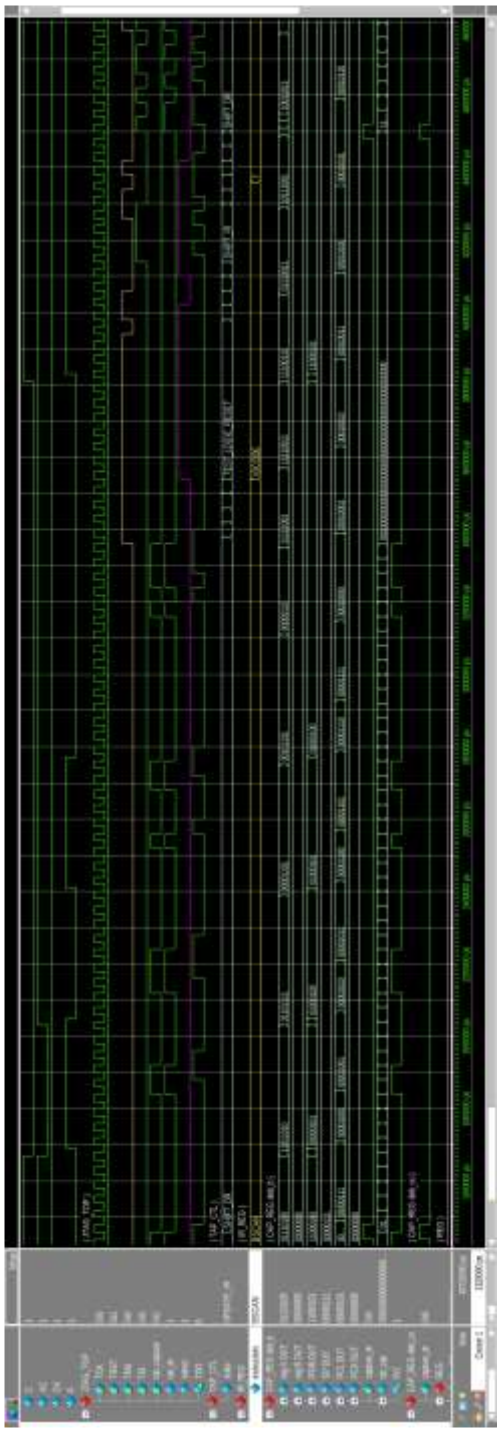

Figure 9 Simulation BSCAN instruction output data via TDO port
Source: Own work [ModelSim]

Other instructions such as INTEST or SAMPLE have so far not been tested on this processor as such, however, in previous stages of development this JTAG architecture was integrated into a RAM memory where these instructions were already tested.

NOTE: These instructions have not been tested on the processor because some modifications to the core RTL are still pending.

To test these instructions with the RAM memory, the following data vector was sent:

1	ram_dataInput_i[31]
1	ram_dataInput_i[30]
1	ram_dataInput_i[29]
1	ram_dataInput_i[28]
0	ram_dataInput_i[27]
1	ram_dataInput_i[26]
1	ram_dataInput_i[25]
0	ram_dataInput_i[24]
0	ram_dataInput_i[23]
1	ram_dataInput_i[22]
0	ram_dataInput_i[21]
1	ram_dataInput_i[20]
0	ram_dataInput_i[19]
1	ram_dataInput_i[18]
1	ram_dataInput_i[17]
0	ram_dataInput_i[16]
1	ram_dataInput_i[15]
0	ram_dataInput_i[14]
1	ram_dataInput_i[13]
0	ram_dataInput_i[12]
1	ram_dataInput_i[11]
1	ram_dataInput_i[10]
0	ram_dataInput_i[9]
0	ram_dataInput_i[8]
1	ram_dataInput_i[7]
0	ram_dataInput_i[6]
1	ram_dataInput_i[5]
0	ram_dataInput_i[4]
1	ram_dataInput_i[3]
1	ram_dataInput_i[2]
0	ram_dataInput_i[1]
1	ram_dataInput_i[0]
0	ram_writeAddress_i[2]
0	ram_writeAddress_i[1]
0	ram_writeAddress_i[0]
0	ram_readAddress_i[2]
1	ram_readAddress_i[1]
0	ram_readAddress_i[0]
0	ram_writeEnable_i
1	ram_writeClock_i

This vector is reflected in the design inputs (RAM for evidence of these results) at 1840ps time in the simulation shown in Figure 10.

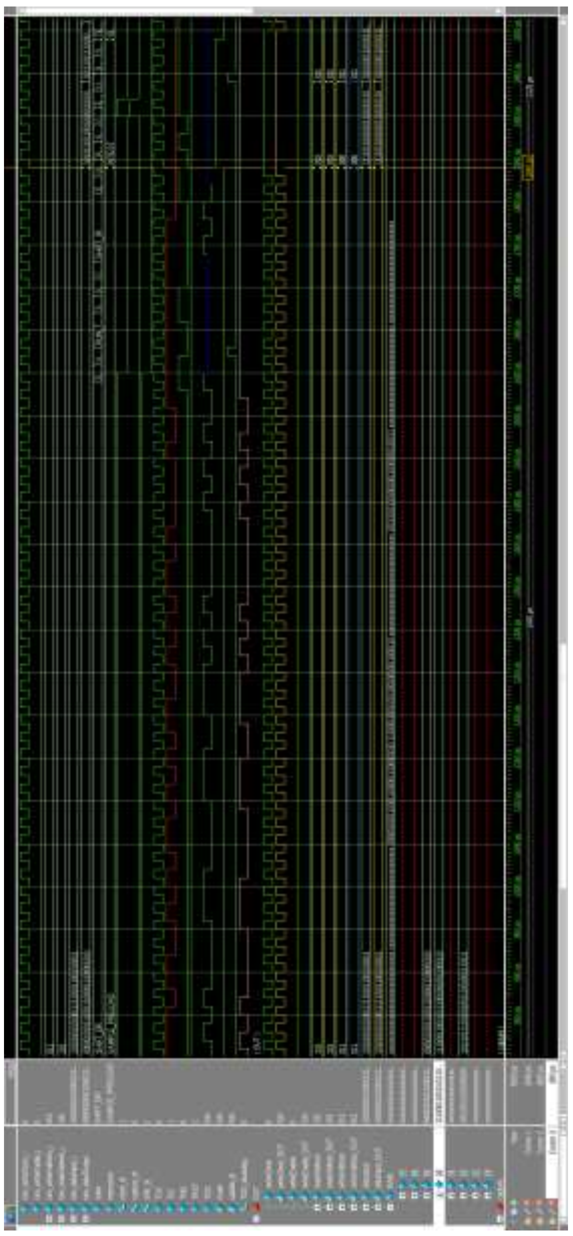

Figure 10 SAMPLE/PRELOAD simulation - INTEST
Source: *Own work [ModelSim]*

These simulations prove that the JTAG interface is working properly; it remains to test all the instructions with the processor, which, as mentioned, is awaiting some adjustments.

Talking about the physical part, tests were made by loading the design and sending data through the JTAG interface integrated in the FTDI chip. Where the main objective is to verify that the data is being sent to the FPGA and the design loaded on the board actually operates with these and gives us as output the expected result.

For these tests we again make use of the memory with the integrated JTAG, as an initial test only the BYPASS instruction was sent, since the objective up to this point is to verify that the communication between components is carried out.

In the simulation of Figure 11, the information vectors sent to each input pin are shown in a simple way, as well as the expected output in the TDO port of the standard.

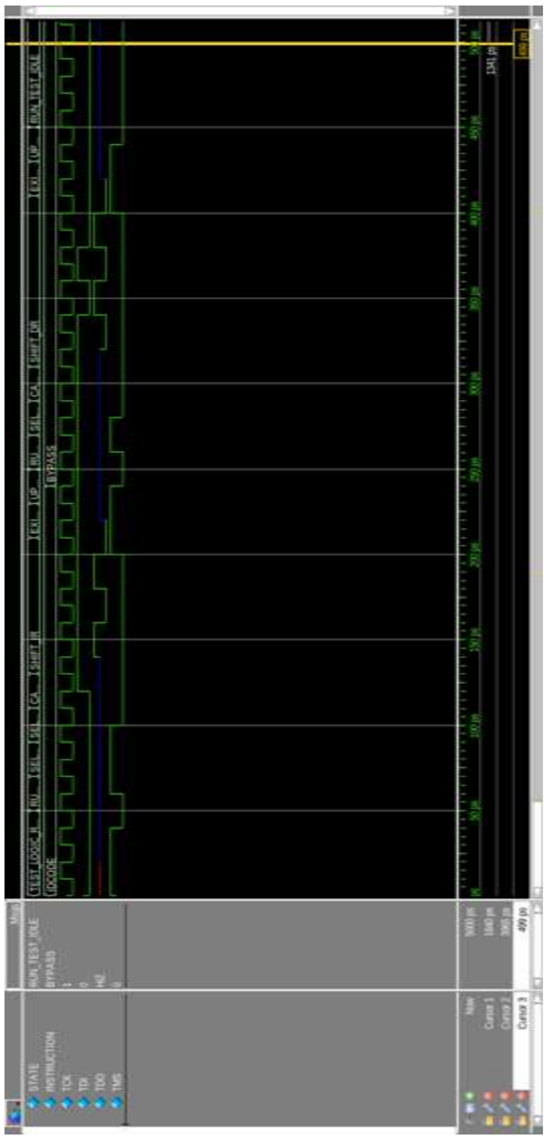

Figure 11 BYPASS simulation
Source: Own work [ModelSim]

To send the information, a C script was generated which, when compiled and sent to the FTDI chip, sends the data through its own ports (TDI, TMS, TCK) to the FPGA pins assigned to the similar ones described in the design.

With the help of an oscilloscope we could observe that the data are being sent correctly and also the output data expected by the TDO port are correct, in Figure 12 you can see the output that TDO (blue signal) has around 300ps which is the expected output of the BYPASS instruction, in addition to the data that are sent to the TDI port, which also match with what was presented in the simulation.

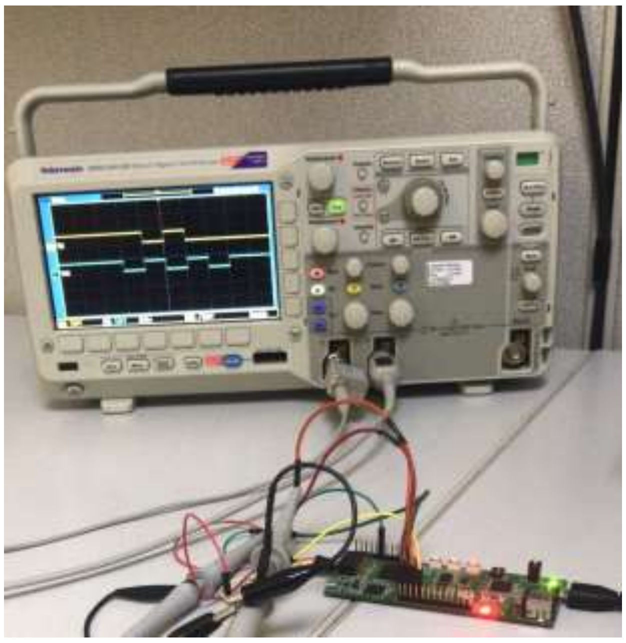

Figure 12 Physical results of BYPASS training
Source: Own work

Conclusions

The JTAG interface complies well with the IEEE 1149. 1 standard, as the instructions used to debug the design work as it indicates, and the added instructions fulfil the specific purpose of this project, which is the monitoring of the processor's internal registers.

With work to be done, but with a well established base and even put under test (with the RAM memory) that has given good results so far, it remains as future work to work in detail with each instruction using the processor as a Logic Unit within the JTAG encapsulation, to ensure that as already happened with the test design everything works correctly both in simulation and physical once the processor is loaded into the FPGA.

The present development where the JTAG interface is integrated to a processor gives us the awareness that this standard is a very flexible tool that is of great help for the performance testing of any design, because being able to stimulate it completely without having to do it necessarily through the primary inputs allows this activity to be less complex and less extra resources are needed to meet the objective.

In addition, hardware designers can seamlessly integrate new instructions that provide specific functions for a particular design without affecting the basis of the standard, which allows the integration of these devices with any other device that has the standard and can work together seamlessly using JTAG as the communication interface between them.

References

Gao, S., Xiao, W., Yang, Z., Wu, D., & Gao, W. (2021). A fast on-chip debugging design for RISC-V processor. *Journal of Physics: Conference Series, Vol 1976.* DOI: https://doi.org/10.1088/1742-6596/1976/1/012056

Gao, S., Wu, D., Xiao, W., Wang, Z., Yang, Z., & Gao, W. (2022). A novel method for on-chip debugging based on RISC-V processor. *MATEC Web of Conferences, 355.* DOI: https://doi.org/10.1051/matecconf/202235503055

Ke, H., Zhongliang, D., & Siyuan, L. (2012). RTL Verification of ARM SoC by JTAG. *International Journal of Digital Content Technology and Its Applications, 6, 77-84.* URL: https://www.researchgate.net/publication/269786095_RTL_verification_of_ARM_SoC_by_JTAG

Jung, D. S., Kwak, S., & Lee, M. H. (2003). Reusable embedded debugger for 32 bit RISC processor using the JTAG boundary scan architecture. *IEEE Asia-Pacific Conference on ASIC.* DOI: https://doi.org/10.1109/apasic.2002.1031569

"IEEE Standard for Test Access Port and Boundary-Scan Architecture," in IEEE Std 1149.1-2013 (Revision of IEEE Std 1149.1-2001), vol., no., pp.1-444, 13 May 2013, DOI: 10.1109/IEEESTD.2013.6515989.

D2XX Programmer's. *Guide Future Technology Devices International Limited (FTDI)* URL: https://ftdichip.com/wp-content/uploads/2023/09/D2XX_Programmers_Guide.pdf

.