Smart Carrier for Scan Chain Emulation of ASIC Prototypes under Test

Juan Manuel Galán, Ainhoa Cortés and Andoni Irizar *CEIT-Basque Research and Technology Alliance (BRTA)*
Manuel Lardizabal 15
20018 Donostia/San Sebastián
Spain
acortes@ceit.es Alejandro Arteaga and José Ignacio Gárate
and Armando Astarloa *Departamento de Tecnología Electrónica*
University of the Basque Country (UPV/EHU)
Bilbao, Spain
armando.astarloa@ehu.eus

Abstract

The low and medium complexity SoCs used for sensing and networking in Critical Sectors, like Energy, Industry, Transportation, and A&D, are typically built using mature 65-22nm technologies. The demand for more specialized, secure, and safe devices is growing due to the high specialization demanded by these strategic sectors. In this context, in the R&D project SoC4cris, we are working on a SoC subsystem based on a 32-bit RISC-V for 65-22nm technologies that could be easily adapted to new SoCs oriented to these sectors.

The testing stage of the ASIC, once it is manufactured and in prototype stages, is very important. Thus, the main aim of the presented work is to automate this testing stage by developing a flexible and cost-effective scan chain Design-for-Test (DfT) verification method that looks for flexibility and facilitates the testing of the ASIC. Furthermore, this method will allow us to test communication standards typically used in the industry.

As the level of integration in digital ICs increases and transistor size decreases, the post-silicon verification of the chips becomes critical. However, the usage of complex Automatic Test Equipment (ATE) is highly expensive, and sometimes very inflexible for little production volumes or multi-project wafers, which complicates its verification. In this paper we propose a low-cost, highly flexible ATE, capable of testing DUTs that integrate Scan Chain as Design-for-Test (DfT) architecture.

Our ATE can be used for little production volumes and prototypes where the verification time is not critical, but a exhaustive testing is needed. Together with a SW library based on Python, it is fast and easy to deploy, maintain and modify. Furthermore, it targets a wide range of DUTs as its working frequency can be dynamically modified by user.

Index Terms

ASIC, Automatic test Equipment (ATE), FPGA, Verification, testing, Scan Chain

I. Introduction

The demand for more specialized, secure, and safe devices in Critical Sectors, like Energy, Industry, Transportation, and A&D is growing. The high specialization demanded by these strategic sectors requires new and sovereign devices. In this context, in the R&D project SoC4cris, we are working on a SoC

This work has been supported by the Basque Government within the fund for research groups of the Basque university system IT1440-22, SOC4CRIS KK-2023/00015 and by 'Secretaría de Estado de Telecomunicaciones e Infraestructuras Digitales' through 'Plan de Recuperación, Transformación y Resiliencia-Financiado por la Unión Europea-NextGenerationEU', 'Cátedras Chip' program, SOC4SENSING TSI-069100-2023-0004.

subsystem based on a 32-bit RISC-V for 65-22nm technologies that could be easily adapted to new SoCs oriented to these sectors.

A. RISC-V for Edge Computing applications

RISC-V is an open-source instruction set architecture (ISA). Its modular design includes a compact base integer ISA and optional standard extensions. This design allows for hardware customization based on the system's requirements. Additionally, it allows Custom Extensions to tailor the ISA for specific applications, fostering innovation and adaptability. In SoC architecture, to this flexibility level, it is added the capability to accelerate or implement custom computation using on-chip bus-connected hardware IP cores. This adaptability of RISC-V makes it suitable for developing small or large applications and enables compilers to generate more reliable code. The flexibility of RISC-V makes it particularly attractive for customized accelerators and educational purposes. Consequently, research groups worldwide are showing a growing interest in developing processing architectures, System-on-Chips (SoCs), and instruction group extensions to be used in any embedded system design [1]. This growing interest is reflected in the expanding ecosystem of RISC-V cores and SoCs, which are well-documented on the initiative repository [2]

In the context of edge computing applications, RISC-V implementations include small 32-bit CPUs. Edge Implementations for Industry, Energy, and Aerospace are mainly based on this type of microcontroller.

The Parallel Ultra Low Power (PULP) Platform is a notable example of open-source academic research on processors using RISC-V. PULP is an advanced microcontroller system with a multi-core platform that can achieve excellent energy efficiency and performance based on the open-source RISC-V architecture [3]. The simplest PULP-based systems are microcontrollers that can be configured to use any supported 32-bit core from the PULP platform, along with adding memory and some peripherals. This RISC-V compatible cores are Ibex, CV32E40P and Micro-riscy.

Ibex is a small and simple core. The development of this core began in 2015 under the name "Zero-Riscy" as part of the PULP platform for energy-efficient computing. This compact 32-bit in-order core with a two-stage pipeline implements the RV32IMC ISA [4]. Ibex is designed to operate either independently in a single-core system or as a co-processor in a multi-core system. It fully adheres to the RISC-V ISA specification and has been verified using the RISC-V compliance test suite. Furthermore, Ibex is compatible with the RISC-V GNU Compiler Toolchain (GCC) and Low-Level Virtual Machine (LLVM) Compiler Infrastructure.

Micro-Riscy is a variation of Zero-Riscy with a minimal area, designed to create a smallest-possible RISC-V core. This core is implemented with a two-stage pipeline and has no hardware support for multiplication and division. To further reduce the area footprint, it implements the RVE RISC-V specification which allows to use only 16 general-purpose registers. Within the PULP ecosystem, this core is employed as a control core for PUPLPino and PULPissimo microcontrollers.

CV32E40P is a small and efficient in-order core based on the RISC-V architecture [5] with a four-stage pipeline. In 2016, under the name "RI5CY, it became a RISC-V core, and in 2020, it started being maintained by the PULP platform, contributing to the "Open Hardware Group". RI5CY was implemented to address energy efficiency in applications deployed on Digital Signal Processors (DSPs).

Several 32-bit microcontroller solutions are available in the RISC-V single-core context, including NEORV32 [6]. NEORV32 is designed to be user-friendly for both beginners and advanced users in FPGA/RISC-V architectures. The project has built-in execution safety measures to ensure predictable behavior. It can execute any C program, including CoreMark and FreeRTOS. Besides, it can be synthesized for any target technology, making it a versatile and powerful microcontroller solution suitable for several applications.

The NEORV32 CPU is an area-optimized RISC-V core that implements the `rv32i_zicsr_zifencei` base ISA and supports several additional/optional ISA extensions. This microarchitecture is based on a

Von-Neumann machine built upon a mixture of multi-cycle and two-stage pipelined execution schemes. The CPU provides independent interfaces for instruction fetch and data access. However, these two bus interfaces are merged into a single processor-internal bus via a prioritizing bus switch.

B. SoC4cris p1: 32-bit RISC-V SoC for Edge Computing

Figure 1 shows the Block Diagram of the ASIC prototype `SoC4cris p1` developed in the project [7]. The University of the Basque Country provides the `TxIKI CPU` and a White Rabbit [8] experimental IP in the front-end design. This RISC-V SoC subsystem is based on `neorv32` RISC-V CPU implementation [6]. CEIT research center contributes with `Peripheral IP`. This is a Sync1 IP to test the interfaces with RISC-V CPU and a digital blinking output.

`SoC4cris p1` is completed with the IPs `Convolution` and `Sync2` to experiment with the acceleration of convolutional arithmetics and the connection of time-aware peripherals.

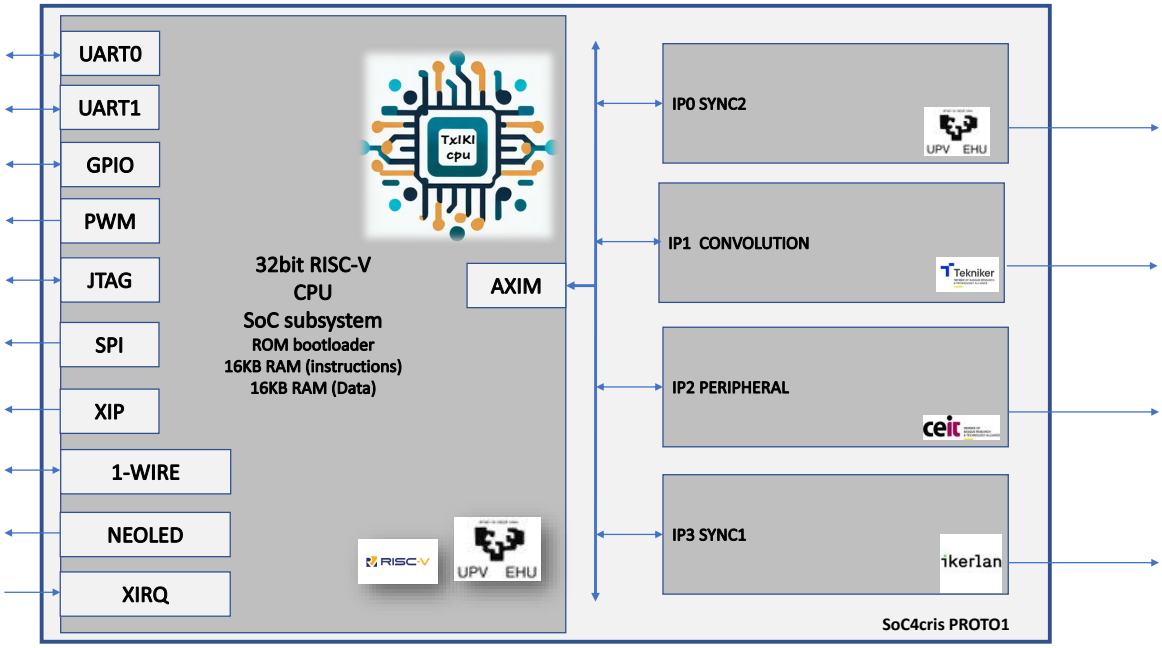


Figure 1: SOC4cris P1 architecture block diagram.

These IPs are used to verify the overall ASIC methodology validating the functionality of the IPs once the ASIC Back-end design is manufactured in UMC™ 65nm technology. On the other hand, the testing stage of the ASIC, once is manufactured, is very important. Thus, the main aim of the presented work is to automate this testing stage developing a low-cost scan chain Design-for-Test (DfT) verification method looking for flexibility and facilitating the testing of the ASIC. Furthermore, this method will allow us to test communication standards typically used in the industry.

This paper is organized as follows. Section II introduces previously developed work in scan chain emulation techniques. Section III presents the developed low-cost Automatic Testing Equipment (ATE) and its mode of operation. Section IV contains the verification procedure followed to test the ATE and the results of the implementation and verification. Finally, Section V concludes the paper and define the work to be performed on the future.

II. State-Of-The-Art

In [9], a FPGA-based ATE for ASIC verification is discussed. The authors describe an ATE to be used with ASICs where a scan chain has been added as Design-for-Test (DfT) architecture. This solution

showed good results for the verification of ICs whose production volume remain low. However, the ATE is focused on defect verification based on fault models detectable by the scan chain (e.g. stuck-at fault model), lacking the possibility of performing both functional and timing verification under real conditions.

In [10], another low-cost FPGA-based ATE is proposed. Authors emphasized the re-configurable capacity of the ATE taking advantage of the FPGA design flow. This design does not focus on an specific DfT architecture, rather on the possibility of inserting test vectors using every DUT GPIO and measuring DUT pinout values and timing behaviour.

Our design merges both scan chain DfT verification method into a low-cost solution easy to deploy and utilise, as it is presented in Section III-A. Besides, as described in Section V it will allow to test communication standards typically used in the industry, as well as to use those interfaces to control the DUT or charge an specific configuration or software (functional verification).

III. Scan Chain Emulation System via JTAG

A. Architecture

To reduce silicon area and increase yield and design flexibility, the Test Access Port is not implemented on the ASIC but in an external carrier. This approach, which maintains an external JTAG Test Access Port (TAP) and emulates the Scan Chain (SC) for Automatic Test Pattern Generation (ATPG), could be adopted as a future design strategy for low or medium-complexity SoCs based on Chiplets. As [11] states, it may help to overcome one of the critical drawbacks of Chiplet-based design, the need for standardized testing among different providers. Indeed, the support for the test provided by IEEE 1149.1 EXTEST instructions in each Chiplet is recommended, since replicating JTAG TAPs in each semiconductor chunk may be impractical.

In the current project, the ASIC implements a Design-for-Test structure, manifested as a single SC (`SDI-SDO`), which is driven by the Test Mode (`TM`) and Scan Chain Enable (`SCEN`) inputs. These two control signals enables two types of test points, the observation points (`OP`) and the control points (`CO`) that are distributed over the combinational logic of the ASIC. By instantiating SC as DfT, the defect detection centers on stuck-at and bridging fault models, that is, identification of shorts between a specific data path and VDD or VSS, and shorts between two nets. In contrast, transition or path delay fault models will be covered by DUT functional testing as SC utilization is limited to information capture on `TCK` edges.

As long as the sequential elements employed in the ASIC implementation process have their data and clock inputs multiplexed with the help of the aforementioned `SCEN` and `TM` signals, the `SCEN` signal either toggles the input between the scan input port (`SDI`) and the normal data input port (DataIn), thus routing the signal through the Design for Test (DFT) combinational test logic or through the normal data path, as shown in Figure 2.

Concurrently, the `TM` input signal sets up the ASIC into verification mode, consequently, the clock source changes to `TCK`, signal controlled by the Scan Chain Emulation System, and distributing the `TCK` signal through all scan chain sequential elements. Simultaneously, hard macros are bypassed to avoid shadowing logic around them.

The Scan Emulation System is based on a commercial TE0726-03M ZynqBerry™ board, being composed of 3 main modules: Test Access Port (TAP), TAP Controller (TAPC) and Zynq Processing System (PS) [12] [13]. The selection of ZynqBerry™ has been driven by its SoC capabilities (ARM Core + FPGA + peripherals) together with its competitive price (˜130€), which makes it a flexible and low-cost solution. The architecture of the Scan Emulation System is presented in Figure 3.

The TAP controls the `SDI` and `SCEN` outputs as well as acquires the `SDO` input, hence performing the link between the JTAG connection and the DUT Scan Chain Register (`SCR`). For this application, only the mandatory IEEE1149.1 functions are implemented into the TAP (Table I) [14]. Aiming to control the `SCR`, `SCEN` output is only asserted while data is being shifted in, that is, when the TAP Finite State

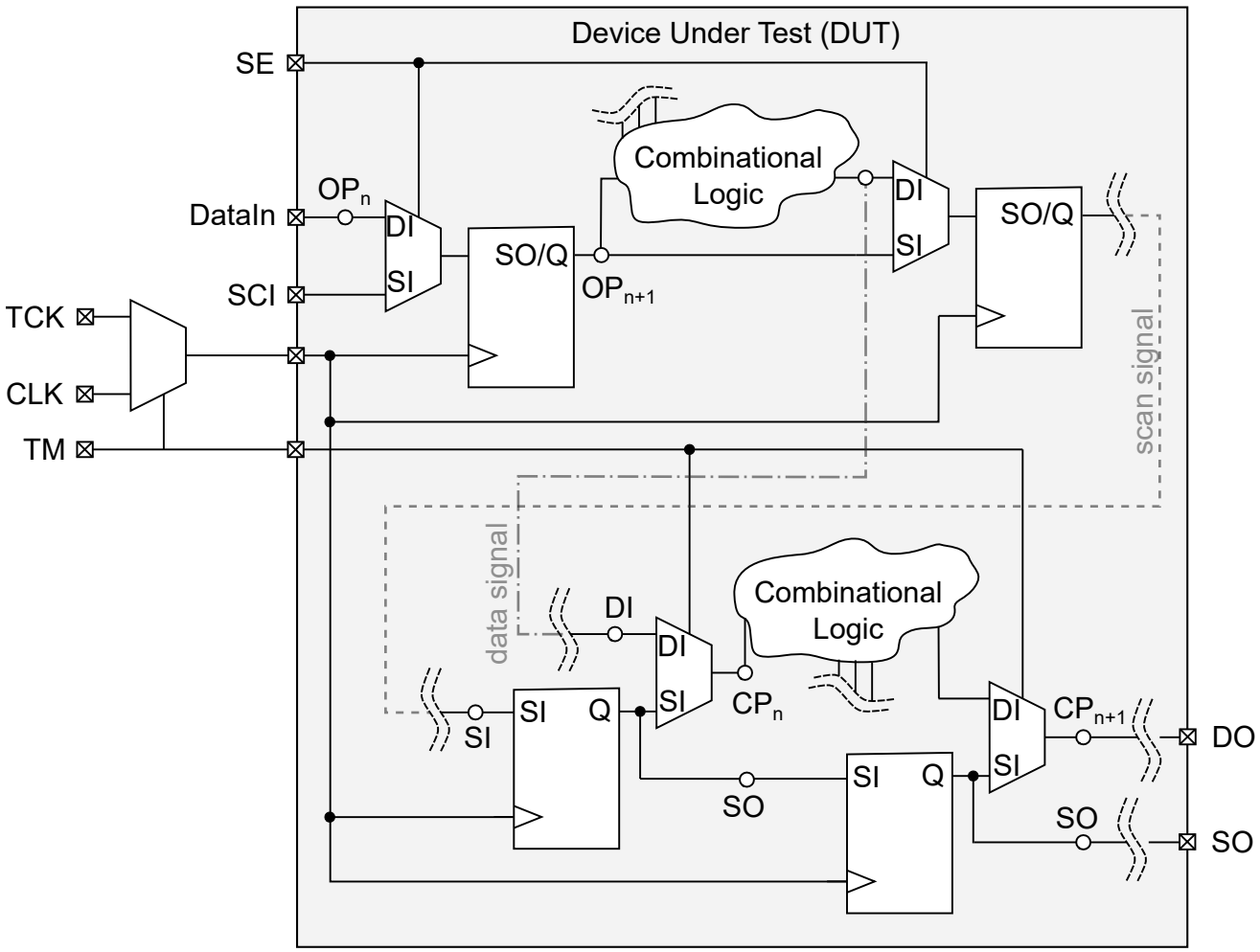


Figure 2: ASIC Scan Chain architecture

Machine is in *SHIFT-DR* state, as illustrated in Figure 4. Therefore, SAMPLE and PRELOAD instructions can be executed as one (SAMPLE/PRELOAD).

DUT register elements sampling is performed at `SCEN` rising edge, that is, once the register element multiplexer selection signal toggles from `DI` to `SI`, as depicted in Figure 2. On the contrary, a known state (`SCR` value) is loaded into the system when the register element multiplexer selection signal toggles from `SI` to `DI`, i.e., at `SCEN` falling edge.

In order to ensure a correct capture of `SDI` input by the DUT at `TCK` rising edge, `SDI` and `SCEN` values are updated at `TCK` falling edge following same behavior than JTAG `TDI` signal as specified by IEEE1149.1 standard [14]. Thus, half the `TCK` clock cycle is provided for `SDI` to become stable before being captured by the DUT. This strategy greatly reduces internal and board delays calculations as timing constrains can be relaxed just by reducing `TCK` frequency.

Table I: IEEE1149.1 implemented instructions on Scan Chain Emulation System.

Name	Code	Description
BYPASS	0b1111	It connects TDI input to TDO output through a 1b shift-register.
SAMPLE	0b1010	It allows a snapshot of the normal operation of the component to be taken and examined.
PRELOAD	0b1011	It allows data values to be loaded onto the latched parallel outputs of the boundary-scan register (BSR).
EXTEST	0b0010	It allows testing of off-chip circuitry and board-level interconnections.
IDCODE	0b0110	It provides the device identification register value.

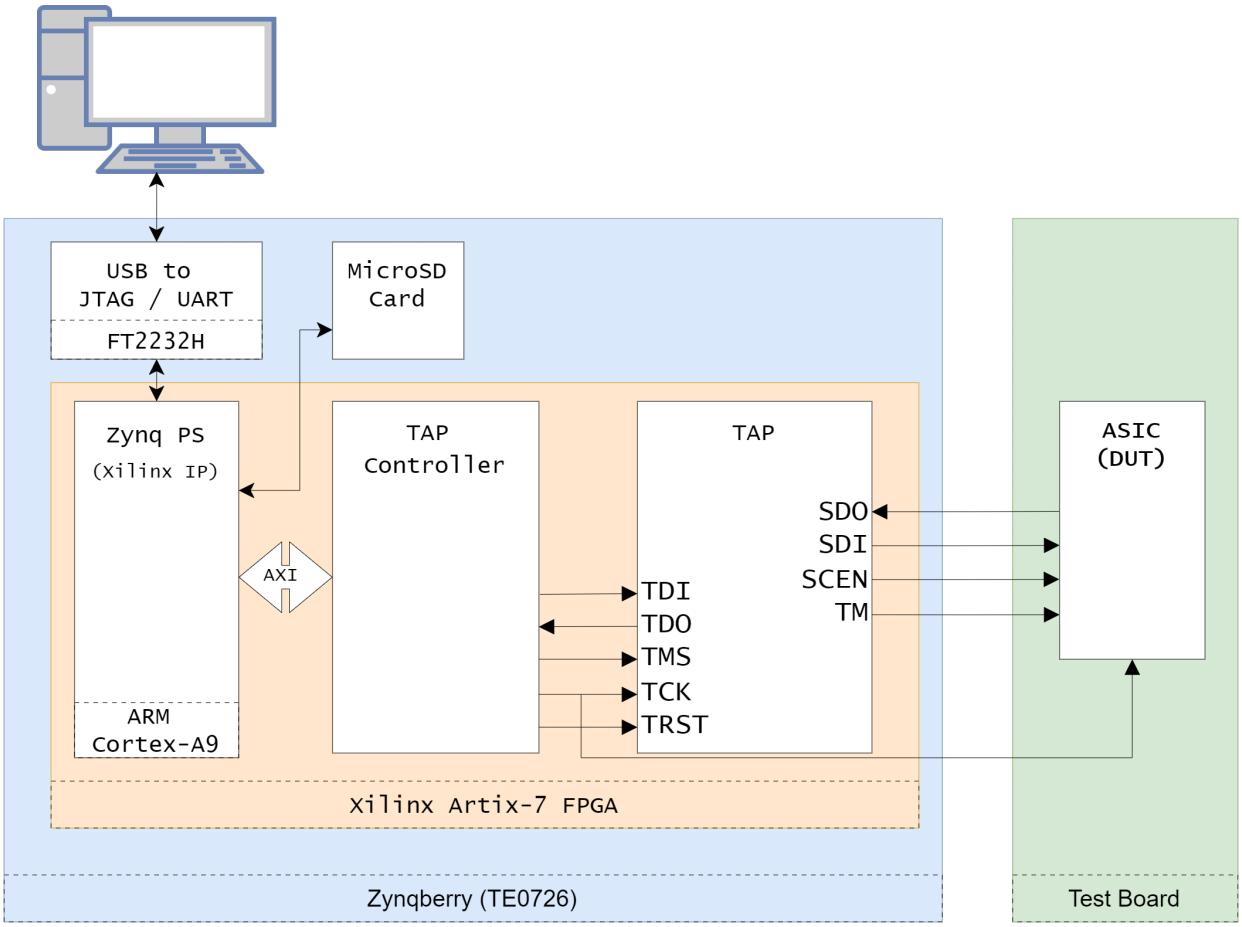


Figure 3: Scan Chain Emulation System architecture.

The TAP Controller (TAPC) acts as master in the JTAG FPGA internal connection to the TAP, generating `TCK`, `TMS` and `TDI` outputs and acquiring `TDO` input. Additionally, Test Mode (`TM`) output is controlled by TAPC and configurable by the user. This holds special significance for functional verification, where the system must be taken into a known state, sometimes unreachable by any other means, and then the system must be let free.

The TAP Controller has an AXI4 Lite interface controlled by the Zynq PS, whose local register map is presented in Table II [15]. This module is capable of generating the `TCK` output with configurable frequency between 12kHz and 25 MHz, targeting a wide range of DUTs. TAPC allows data with arbitrary length to be shifted in, prior division into 32b data vectors. Once the TAP Controller has registered the 32b vector to be shifted in, a write acknowledgement flag is raised and the user can provide the next 32b vector. If data were not provided fast enough, the TAP Controller would raise a underflow flag and the system (TAPC + TAP) would enter into pause state (*PAUSE-DR*, Figure 4). When dealing with shifted out data (coming from `SDO` input), a 32bx32 FIFO is utilized to minimize the chance of non-read data being overwritten, thereby providing users ample time to retrieve data and consequently freeing FIFO space.

Table II: TAP Controller AXI Register Interface (C: Control, S: Status, D: Data).

Address	Type	Name	Description
0x00	C	TAPC_CTRL	Control register
0x01	C	TAPC_TCK_DIV	CLK to TCK dividing factor
0x02	S	TAPC_STAT	Status register
0x03	S/D	TAPC_DATA_LEN	Data length (Write: to send; Read: received)
0x04	D	TAPC_DATA	Data (Write:to send; Read:received)

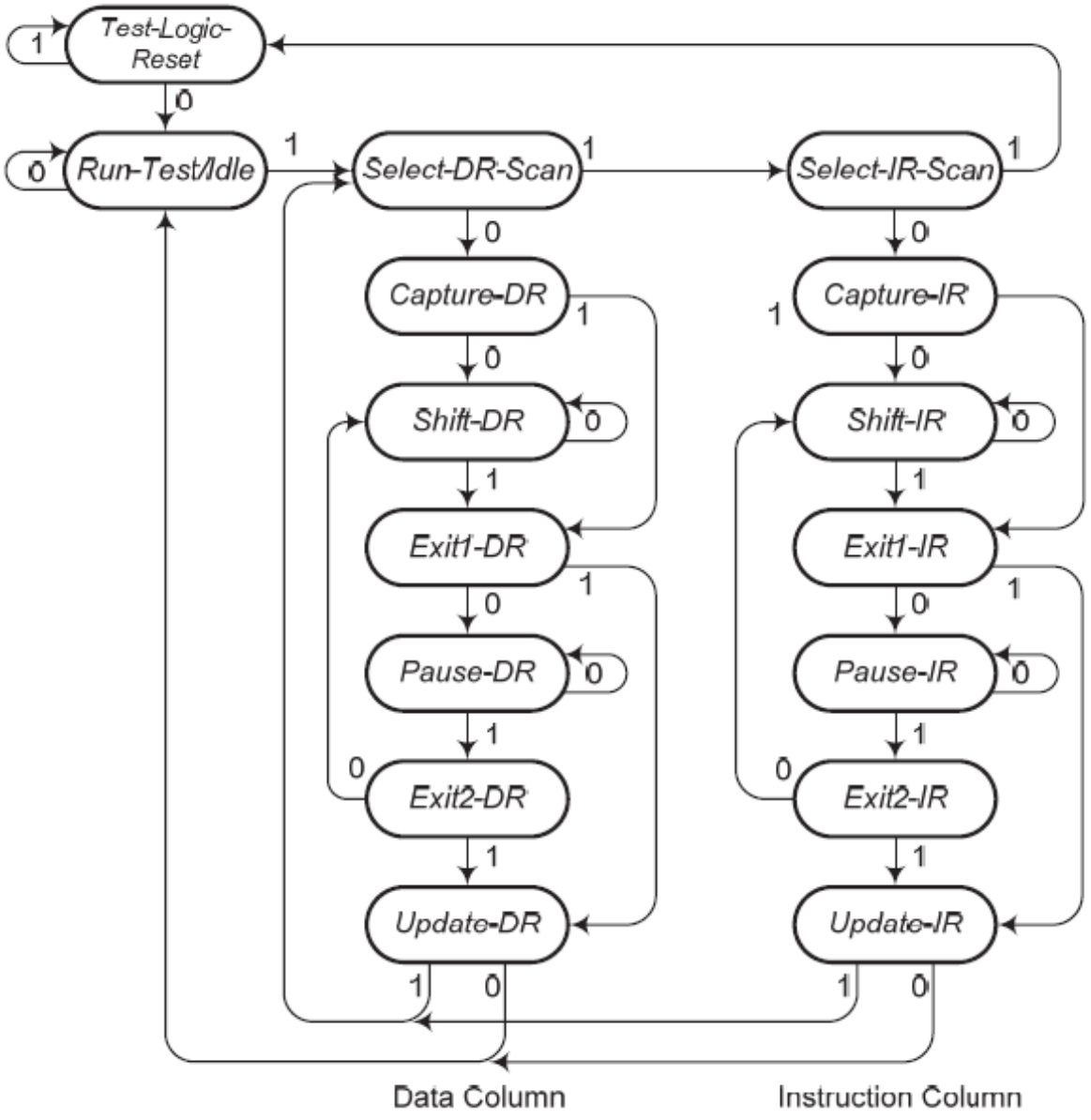


Figure 4: TAP Finite State Machine [14]

The Zynq Processing System is fed by a 33.33MHz external oscillator. ARM processor runs at 666.66MHz while a 50MHz clock is provided to the Zynq Programmable Logic (PL). The PS executes a Petalinux™ OS image, employing a two-stage boot process that begins with flash memory and then transitions to an external Micro-SD card. The Petalinux™ OS is commanded through an UART connection. Besides, the OS instantiates a SCP server available for file transfer through a 10/100Mbs Ethernet connection.

B. Mode of Operation

The Scan Chain Emulation System has two operational modes: directed and autonomous. On directed mode, the Zynq PS acts as bridge between the commands send by an external computer through UART connection and the TAP Controller. The SW link between these two modules is performed by an in-house developed Python library capable of reading and writing TAP Controller registers as well as executing JTAG instructions.

In autonomous mode, a Python script runs the test file (located into the Micro-SD card) and compares the generated output with the predefined expected value, recording the test log and result along the process. In order to ease parsing by the Python script, the test file is JSON formatted. For each test the instructions to be executed, the data input and the expected value are specified in the test file, as depicted in Figure 5.

```
{
"Mode" : ATP, test:
[
  {"instr": "IDCODE", "dataIn": "0x00000000", "expData": "0x000009DD"},
  {"instr": "SAMPLE", "dataIn": "0x00000000", "expData": "0xCE172024"}
]
}
```

Figure 5: Test file JSON format.

Additionally, for each test, test mode option must be selected between two possibilities: ATPG and Functional.

ATPG verification aims to detect defects on the DUT typically associated to the manufacturing process. In this verification mode, a test pattern is initially shifted into the DUT for its values to be propagated through the DUT's logic (see Figure 2). Test pattern propagation is limited to one clock cycle as depicted in Figure 7a (2). This way the logic function to be verified is reduced to a single data path between two register elements.

For this DUT in particular (RISCV SoC ASIC), the Scan Chain Emulation System (SCES) is not capable of setting all DUT inputs into a known state. However, the consequences of this drawback are limited as the SCES is capable of controlling the communication interfaces, leaving uncontrolled general purpose inputs which are registered after input, thus limiting the non-controllable logic to a negligible portion of the DUT's logic.

On the other hand, functional verification aims to take the DUT to a known state (through test vector and controllable I/Os) and let it run freely, being the expected behavior checked by other interfaces apart from the Scan Chain. This verification mode is specially relevant to test error modes not capable of being excited by any other means.

IV. Implementation Results

The Scan Chain Emulation System (SCES) Programmable Logic resources usage are presented in Table III.

Table III: Scan Chain Emulation System resources usage.

Resources	Used	Available	Percent
6-Input LUT (LUTRAM included)	574	17600	3.24 %
LUTRAM (512b)	58	6000	0.97 %
Flip-Flop	773	35200	2.20 %
Block RAMs (36Kb)	0.5	60	0.83 %

Verification of the system has been conducted using a ZedBoard to emulate the DUT. The complete test set-up is shown in Figure 6.

Verification activities focused on 2 main points: functional correctness and timing. ZedBoard scan chain register elements were pre-loaded with a known pattern in order to verify that the SCES is able to retrieve `SDO` value at the correct time over a wide range of frequencies. Additionally, the delay between `TCK` falling edge and, `SCEN`, `SDI` signals changes have been checked to ensured that there is enough time available for the signals to become stable at the DUT side. This parameter has been called fly-time (T_{fly},

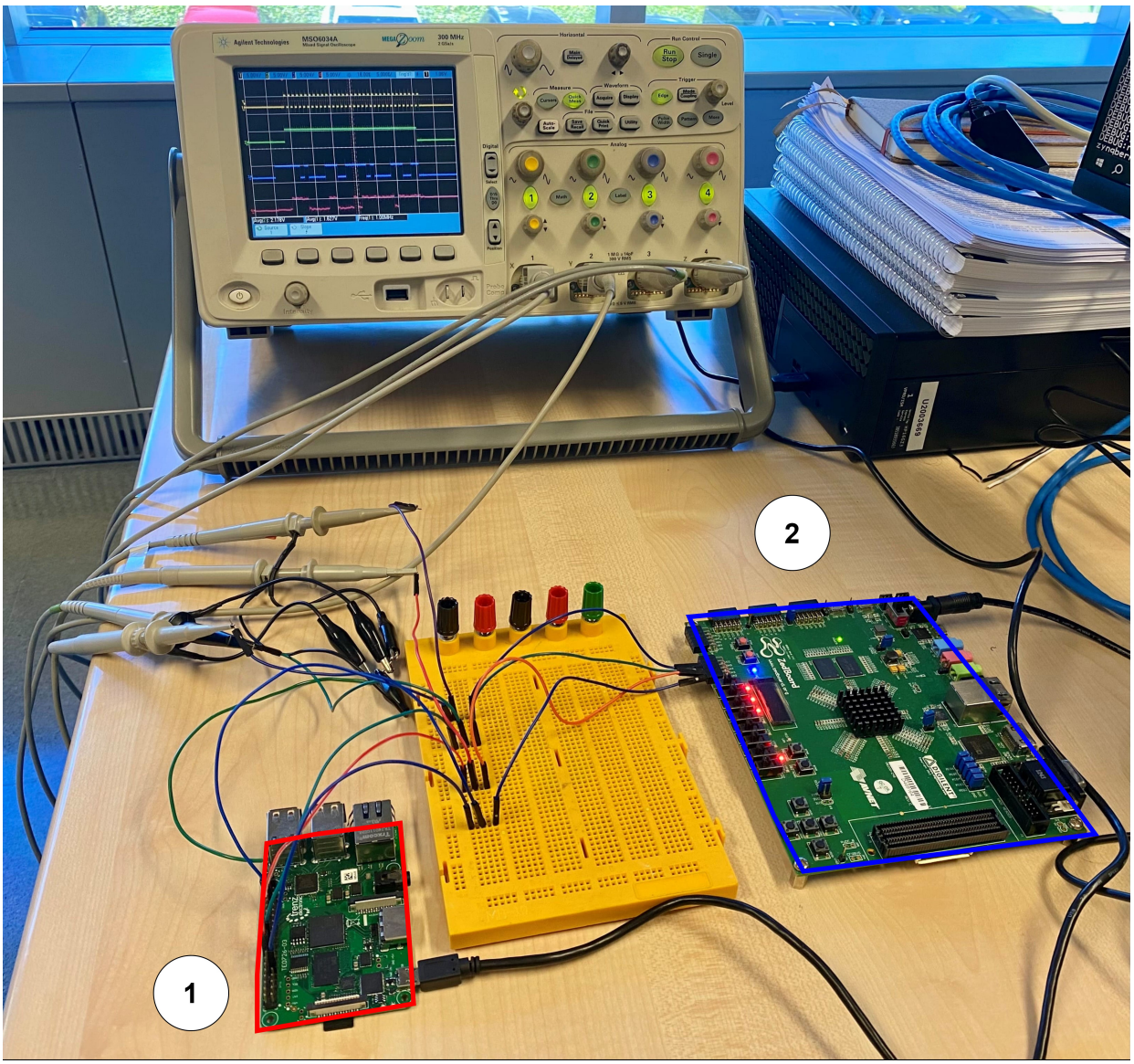


Figure 6: Scan Chain emulation System verification Set-up. (1) SCES (ZynqBerry) (2) DUT (ZedBoard)

presented in Eq 1) and it has great relevance as it limits the maximum frequency of the complete system (SCES + DUT). T_{fly} depends on 3 parameters:

- $T_{\text{TCK}}/2$. `TCK` half period, it represents the total available time in a ideal case. Half a period is taken because signals are generated-captured at `TCK` falling edge on TAP side, but at the rising edge on DUT side.
- $\Delta_{\text{TCK-SE}}, \Delta_{\text{TCK-SDI}}$. Output delay of `SCEN` or `SDI` signal with respect to `TCK` (this delay may be negative).
- T_{SU}^{DUT}. External Set-Up time of `SDI` with respect to the `TCK` input. This value must be respected in order to avoid metastability issues.

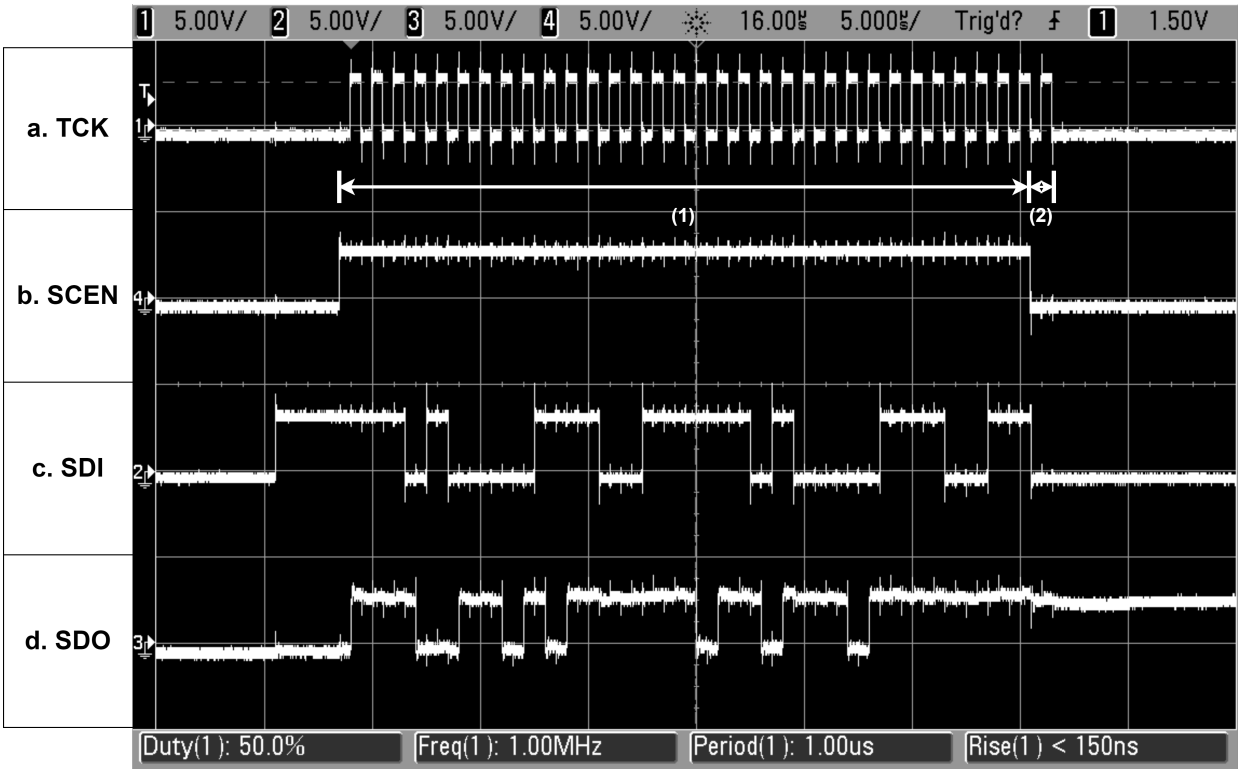
$$T_{\text{fly}} = T_{\text{TCK}}/2 - \max(\Delta_{\text{TCK-SE}}, \Delta_{\text{TCK-SDI}}) - T_{\text{SU}}^{DUT} \tag{1}$$

The following test procedure was conducted for the `TCK` frequencies shown in Table IV with a DUT scan chain length of 32b:

1) Set `TCK` frequency to the desired value by configuring TAPC_TCK_DIV register value.

2) Shift In scan chain value to be loaded. JTAG SAMPLE/PRELOAD instruction is used for this goal.
3) Check the previous/reset scan chain value has been correctly shifted out (through SDO output) by reading TAPC_DATA register.
4) Shift in the 32b wide 0x0 value and check that the previously shifted in value has been received by reading the TAPC_DATA register.

Figure 7 shows TCK, SCEN, SDI and SDO oscilloscope measurement and terminal capture for the test with TCK at 1MHz frequency.



(a) 1MHz Shift Data Test

```
zynqberryOS:~$ sudo python3 -O tap_lib.py --write 0x1 0x32
INFO:root:/dev/mem opened
INFO:root:Physical Memory was mapped to user space
INFO:root:Wrote 0x32 to address 0x1
zynqberryOS:~$ sudo python3 -O tap_lib.py --instr 0xA 0xCE17CE17 0x20
INFO:root:/dev/mem opened
INFO:root:Physical Memory was mapped to user space
INFO:root:Executing SAMPLE/PRELOAD instruction
zynqberryOS:~$ sudo python3 -O tap_lib.py --read 0x4
INFO:root:/dev/mem opened
INFO:root:Physical Memory was mapped to user space
INFO:root:Read address ['0x4']: 0XFEEDFACE
zynqberryOS:~$ sudo python3 -O tap_lib.py --instr 0xA 0x0 0x20
INFO:root:/dev/mem opened
INFO:root:Physical Memory was mapped to user space
INFO:root:Executing SAMPLE/PRELOAD instruction
zynqberryOS:~$ sudo python3 -O tap_lib.py --read 0x4
INFO:root:/dev/mem opened
INFO:root:Physical Memory was mapped to user space
INFO:root:Read address ['0x4']: 0XCE17CE17
zynqberryOS:~$
```

(b) Capture of Petalinux OS terminal

Figure 7: Test Measurements. (1) Shift In (SDI): 0xCE17CE17 Shift Out (SDO): 0xFACEFEED. (2) Update TCK cycle with SCEN de-asserted

Table IV shows the result of the tests performed on the Scan Chain Emulation System. As it can be

seen, the 25MHz test failed. T_{fly} was greater than the maximum available time, hence the capture value was not stable and reliable. It can be observed that working with the same set-up and reducing the working frequency solved the problem as previously expected.

Table IV: Scan Chain Emulation System verification results

TCK Freq	Shift In Value	SC Reset Value Reading	SC Shift Out Value Reading	32b Shift Time
25MHz	0xA5A5A5A5	FAIL	FAIL	N/A
5MHz	0xCAFE2024	PASS (0xFEEDFACE)	PASS	$\sim$690 µs
1MHz	0xCE17CE17	PASS (0xFEEDFACE)	PASS	$\sim$710 µs
100kHz	0xABCDEF01	PASS (0xFEEDFACE)	PASS	$\sim$1100 µs

V. Conclusion

This work introduced a new low-cost FPGA-based ATE to be used in the verification of ASICs where the scan chain is accessible via DUT pinout. The proposed solution avoids the need of implementing a Test Access Port (TAP) on the ASIC design, hence reducing its area and incrementing its flexibility. This system reduces the cost by an order of magnitude compared with other previously proposed solutions as [9]. Furthermore, it does not rely on proprietary SW or HW other than the carrier board, and it has been developed on well-known, highly used programming languages as Python (for SW) and VHDL (for HW).

On a future release of the Scan Chain Emulation System, it is intended to add a communication protocol test interface, capable of verifying the DUT compliance with the following commonly used protocols: UART, SPI, I2C and ONEWIRE. Moreover, it will be able to read/write GPIOs and external interrupts (XIRQ). These interfaces would be connected to the Scan Emulation System using the 2x20 pin header (same disposition as RaspBerry™ connector, Figure 8). An independent Python library will be developed for each interface, thus facilitating its use and simplifying its modification in case of adaptation need to new DUTs.

Furthermore, it is planned to adapt the test patterns input format to Standard Test Interface Language (STIL) to avoid the need of translation between ATPG output format and JSON.

References

[1] "Digital Design and Computer Architecture, RISC-V Edition - 1st Edition | Elsevier Shop." [Online]. Available: https://shop.elsevier.com/books/digital-design-and-computer-architecture-risc-v-edition/harris/978-0-12-820064-3

[2] "riscvarchive/riscv-cores-list," Feb. 2024, original-date: 2019-02-21T19:47:14Z. [Online]. Available: https://github.com/riscvarchive/riscv-cores-list

[3] "PULP Project Information." [Online]. Available: https://pulp-platform.org/projectinfo.html

[4] "Ibex: An embedded 32 bit RISC-V CPU core — Ibex Documentation 0.1.dev50+g5a8a1a9.d20240217 documentation." [Online]. Available: https://ibex-core.readthedocs.io/en/latest/

[5] D. Schiavone. Open source hardware the new reality CV32e40p project | OpenHW group. [Online]. Available: https://www.openhwgroup.org/resources/blog/open-source-hardware-the-new-reality-cv32e40p-project/

[6] S. Nolting and A. t. A. Contributors, "The NEORV32 RISC-V Processor," Aug. 2023. [Online]. Available: https://github.com/stnolting/neorv32

[7] APERT Research Team, "SOC4CRIS: Investigación y Capacitación del Ecosistema I+D+i en el Diseño, Fabricación y Testing de Semiconductores para Sistemas Críticos," 2024. [Online]. Available: https://www.ehu.eus/es/web/apert/soc4cris

[8] White rabbit official CERN website. [Online]. Available: https://white-rabbit.web.cern.ch/

[9] D. de Carvalho, B. Sanches, M. De Carvalho, and W. Van Noije, "A flexible stand-alone fpga-based ate for asic manufacturing tests," in *2018 IEEE 19th Latin-American Test Symposium (LATS)*, 2018, pp. 1–6.

[10] L. Mostardini, L. Bacciarelli, L. Fanucci, L. Bertini, M. Tonarelli, and M. De Marinis, "Fpga-based low-cost automatic test equipment for digital integrated circuits," in *2009 IEEE International Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications*, 2009, pp. 32–37.

[11] A. Detofsky, "Architecting chiplets for product manufacturing test resiliency," Apr. 2023. [Online]. Available: https://eps.ieee.org/publications/enews/april-2023/1000-architecting-chiplets-for-product-manufacturing-test-resiliency.html

[12] Zynqberry te0726 by trenz electronics. [Online]. Available: https://wiki.trenz-electronic.de/display/PD/TE0726+Resources

[13] Xilinx®, "Zynq-7000 soc data sheet: Overview (ds190)," Jul. 2018. [Online]. Available: https://docs.amd.com/v/u/en-US/ds190-Zynq-7000-Overview

[14] "Ieee standard for test access port and boundary-scan architecture," *IEEE Std 1149.1-2013 (Revision of IEEE Std 1149.1-2001)*, pp. 1–444, 2013.

[15] ARM™, "Amba® axi and ace protocol specification (h.c)," Jan. 2021. [Online]. Available: https://developer.arm.com/documentation/ihi0022/hc/?lang=en

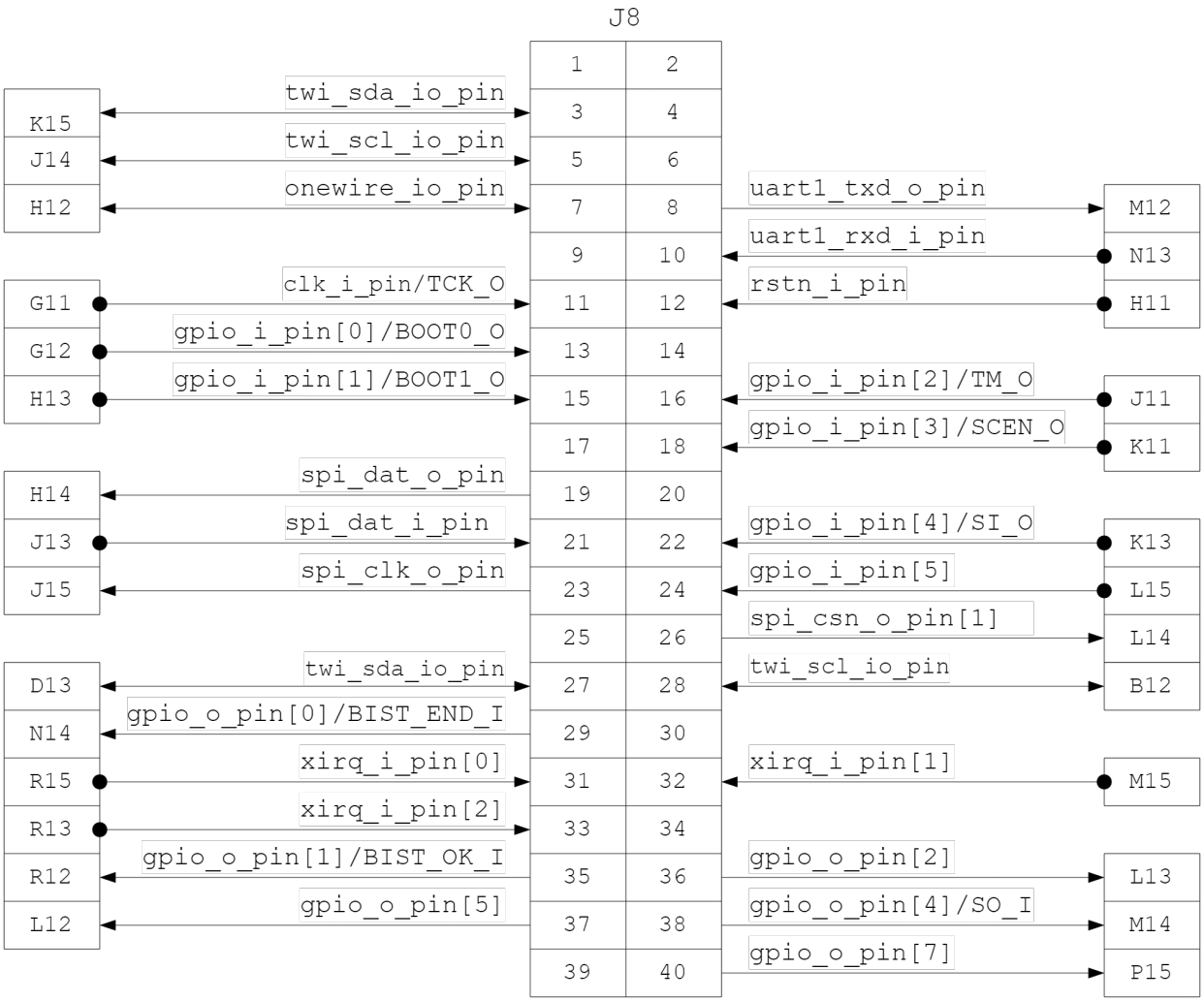


Figure 8: Zynqberry 2x20 connector disposition.