Design of Boundary-Scan Testing in CMOS Image Sensors for Industrial Applications

INTERNSHIP REPORT

Pedro Nuno Teixeira dos Santos
MASTERS IN TELECOMMUNICATIONS AND ENERGY NETWORKS ENGINEERING

ORIENTATION
Alberto de Jesus Nascimento

Master Thesis supervised by:

Alberto de Jesus Nascimento

Assistant Professor of the:

Competence Centre of Exact Sciences and Engineering

UNIVERSITY OF MADEIRA

Acknowledgement

I would like to sincerely thank all those who directly or indirectly contributed to accomplishing this work.

To Professor Alberto de Jesus Nascimento for the guidance during the execution of this work, his availability since the first moment, the straightforwardness and all the support given to improve and enrich the quality of this work.

To Mr Martin Wäny and to Mrs. Susana Barbezat Pita, on behalf of the Awaiba, Lda company, for the opportunity to perform this work with all its collaboration, for the support always provided and for their friendship, as well as to all other colleagues for their support and friendship.

To the University of Madeira and to all the Professors and Assistants from the *Competence Centre of Exact Sciences and Engineering*, for their support and academic education, referring in particular to Professors Dionísio Barros and José Manuel Baptista.

To the University of Aveiro, for my previous academic education and teaching quality, that allowed me to be well prepared for the work world.

To my wife in a very special way, for all the times and all the support, affection, attention, patience and friendship, without which it would be possible to go on this adventure.

To my parents, my in-laws, family and friends for all the understanding, friendship and support they have been giving me throughout all my life.

To everyone who participated in my life, that one way or another, has been important and permitted, in a constructive manner, to become the person I am.

This page was intentionally left blank

Preface

The master thesis presented in this document resulted from the study and implementation of a *Boundary-Scan* circuit used, in image sensors designed by the company Awaiba, Lda. This design arises from the need to improve and promote functional tests on sensors with a large number of inputs and outputs. Therefore an industrial Standard, widely used in other integrated circuits that are implemented on the same *Printed Circuit Board* as the sensor, was used.

This work was developed in a deep participation and development in Awaiba,Lda company, which provided all the technical and functional means to the development of the work.

The Awaiba,Lda company operates in the market of image sensors in CMOS technology, since 2004, holding proprietary technologies, know-how and many patents that make it one of the most innovative companies in Portugal. Not having a foundry (Fabless) it works in partnership with various foundries, using the manufacturing process most suited to the needs of each sensor. Without the use of this knowledge and these technologies it would not be possible to achieve the objectives of the work described in this document.

Currently the work is applied on the company's sensors, minimizing, this way, the test time of the produced sensors and quickly verifying those who do not meet the requirements at the connectivity level.

Abstract

Tests on printed circuit boards and integrated circuits are widely used in industry, resulting in reduced design time and cost of a project. The functional and connectivity tests in this type of circuits soon began to be a concern for the manufacturers, leading to research for solutions that would allow a reliable, quick, cheap and universal solution. Initially, using test schemes were based on a set of needles that was connected to inputs and outputs of the integrated circuit board (*bed-of-nails*), to which signals were applied, in order to verify whether the circuit was according to the specifications and could be assembled in the production line.

With the development of projects, circuit miniaturization, improvement of the production processes, improvement of the materials used, as well as the increase in the number of circuits, it was necessary to search for another solution. Thus *Boundary-Scan Testing* was developed which operates on the border of integrated circuits and allows testing the connectivity of the input and the output ports of a circuit.

The *Boundary-Scan Testing* method was converted into a standard, in 1990, by the IEEE organization, being known as the *IEEE 1149.1 Standard*. Since then a large number of manufacturers have adopted this standard in their products.

This master thesis has, as main objective: the design of *Boundary-Scan Testing* in an image sensor in CMOS technology, analyzing the standard requirements, the process used in the prototype production, developing the design and layout of *Boundary-Scan* and analyzing obtained results after production.

Chapter 1 presents briefly the evolution of testing procedures used in industry, developments and applications of image sensors and the motivation for the use of architecture *Boundary-Scan Testing.*

Chapter 2 explores the fundamentals of *Boundary-Scan Testing* and image sensors, starting with the *Boundary-Scan* architecture defined in the Standard, where functional blocks are analyzed. This understanding is necessary to implement the design on an image sensor. It also explains the architecture of image sensors currently used, focusing on sensors with a large number of inputs and outputs.

Chapter 3 describes the design of the *Boundary-Scan* implemented and starts to analyse the design and functions of the prototype, the used software, the designs and simulations of the functional blocks of the *Boundary-Scan* implemented.

Chapter 4 presents the layout process used based on the design developed on chapter 3, describing the software used for this purpose, the planning of the layout location (*floorplan*) and its dimensions, the layout of individual blocks, checks in terms of layout rules, the comparison with the final design and finally the simulation.

Chapter 5 describes how the functional tests were performed to verify the design compliancy with the specifications of Standard IEEE 1149.1. These tests were focused on the application of signals to input and output ports of the produced prototype.

Chapter 6 presents the conclusions that were taken throughout the execution of the work.

Key Words: *Boundary-Scan Testing, CMOS, Image Sensors, Standard IEEE 1149.1.*

Resumo

Os testes em placas de circuito impresso e circuitos integrados são largamente usados na indústria, resultando na redução do tempo e custos de um projeto. Os testes funcionais e de conectividade neste tipo de circuitos desde cedo começaram a ser uma preocupação dos fabricantes, levando à investigação de soluções que permitissem uma solução fiável, rápida, barata e universal. Inicialmente, foram usados esquemas de teste onde um conjunto de agulhas era conectado nas entradas e saídas da placa de circuito integrado ("cama de agulhas"), às quais eram aplicados sinais, para aferir se o circuito estava segundo a especificação e poderia ser assemblado na produção.

Com a evolução dos projetos, miniaturização dos circuitos, melhoria dos processos de produção e dos materiais utilizados, bem como, com o aumento do número de circuitos, foi necessário encontrar outra solução. Assim surgiu o *Boundary-Scan Testing* que atua na fronteira dos circuitos integrados e que permite testar a conectividade dos portos de entrada e saída.

O *Boundary-Scan Testing* foi convertido, em 1990, num *Standard* pelo organismo internacional IEEE, denominado como *Standard IEEE 1149.1.* Desde então um grande número de fabricantes de adotaram este standard nos seus produtos.

Esta dissertação tem como principal objetivo o projecto de *Boundary-Scan Testing* num sensor de imagem, em tecnologia CMOS, analisando para tal os requisitos do *Standard*, o processo utilizado na produção do protótipo, desenvolvendo o desenho e o *layout* do *Boundary-Scan* e analisando os resultados obtidos após a produção.

O capítulo 1 apresenta de forma introdutória a evolução dos processos de teste utilizados na indústria, a evolução da indústria e das aplicações de sensores de imagem e a motivação do uso da arquitetura *Boundary-Scan Testing*.

O capítulo 2 explora os fundamentos do *Boundary-Scan Testing* e dos sensores de imagem, desde a arquitetura *Boundary-Scan* definida no *Standard*, onde os blocos funcionais são analisados. Esta explicação é necessária para a aplicação do projecto no sensor de imagem. É ainda explicada a arquitetura dos sensores de imagem utilizados atualmente, focando-se os sensores com um grande número de entradas e saídas.

O capítulo 3 descreve o desenho do *Boundary-Scan* implementado, começando por analisar o desenho e as funções do protótipo, o *software* utilizado, os desenhos e as simulações dos blocos funcionais.

O capítulo 4 apresenta o processo de *layout* utilizado, partindo do desenho desenvolvido, descrevendo o *software* utilizado para o efeito, a planificação do local do *layout* (*floorplan*) e as suas dimensões, o *layout* dos blocos individuais, a verificação em termos de regras de *layout*, a comparação com o desenho e terminando com a simulação final.

No capítulo 5 é descrito como foram efetuados os testes funcionais para verificar se o desenho correspondia às especificações do *Standard* IEEE 1149.1. Estes testes incidiram sobre a aplicação de sinais aos portos de entrada e de saída do protótipo produzido.

O capítulo 6 apresenta as conclusões que foram retiradas ao longo da execução do trabalho desenvolvido.

Palavras Chave: *Testes de Boundary-Scan, CMOS, Sensores de Imagem, Standard IEEE 1149.1.*

Table of Contents

Acknowledgement ... *iii*

Preface ... *v*

Abstract ... *vi*

Resumo ... *viii*

Table of Contents ... *xi*

List of Figures ... *xiii*

List of Tables ... *xiv*

Glossary ... *xv*

1. Introduction ... *1*

1.1. Motivation of the Boundary-Scan Architecture ... *3*

2. Fundamentals of Boundary-Scan Chain and Image sensors ... *7*

2.1. The Boundary-Scan Chain Architecture ... *7*

2.1.1. The Test Access Port – TAP Controller ... *9*

2.1.2. The Instruction Register ... *15*

2.1.3. The Data Registers ... *18*

2.1.4. Non-Invasive Operational Modes ... *22*

2.1.5. Pin-Permission Operational Modes ... *23*

2.2. CMOS Image Sensors Architecture for Vision Applications ... *26*

2.2.1. CMOS Image Sensors with higher number of IOs ... *31*

3. Schematic and Simulation Model for the Design ... *33*

3.1. Sensor Prototype Architecture ... *33*

3.1.1. Power Supply of the Prototype ... *34*

3.1.2. Design For Test of the Prototype ... *35*

3.2. Prototype Design Rules ... *36*

3.3. The CAD Software used ... *37*

3.4. Boundary Scan-Chain Design ... *39*

3.4.1. TAP Controller State Machine Design ... *40*

3.4.2. TAP Controller Design ... *46*

3.4.3. The Instruction Register Design ... *48*

3.4.4. The Bypass Register Design ... *49*

3.4.5. The Boundary Scan Register Design ... *49*

3.4.6. Boundary Scan timing requirements ... *51*

3.4.7. Boundary Scan top simulation ... *52*

4. Layout implementation of the Design ... *63*

4.1. Software used for Layout – IC Station ... *63*

4.2. Boundary-Scan Layout – Floorplan ... *64*

4.3. *Layout of the TAP Controller State Machine* ... 66
4.4. *Layout of the TAP Controller Block* ... 66
4.5. *Layout of the Bypass Register Block* ... 67
4.6. *Layout of the Instruction Register Block* ... 67
4.7. *Layout of the Boundary-Scan Cell* ... 67
4.8. *Layout of the Boundary-Scan Controller Block* ... 68
4.9. *Boundary-Scan Register Layout dimensions* ... 68
4.10. *DRC and LVS checks* ... 69
4.11. *Post-Layout Simulation* ... 69
5. *Prototype verification and functional Boundary-Scan test* ... 71
5.1. *Validation of the Sensor Boundary-Scan Design* ... 73
5.2. *Tests and Results of the Boundary-Scan Register* ... 79
5.3. *Sensor Prototype BDSL file* ... 84
6. *Conclusions* ... 85
6.1. *Future Work* ... 86
References ... 89
Appendixes ... 91

List of Figures

Figure 1-1: IEEE Std. 1149.1 Boundary-Scan Testing [Std1149.1-Strx-AltCorp, 2007]. 2
Figure 1-2: Functional test using the "bed-of-nails". 4
Figure 2-1: Simplified Boundary-Scan Architecture of an 1149.1 compliant Integrated Circuit. 8
Figure 2-2: TAP Controller state machine diagram. 10
Figure 2-3: Instruction register architecture detail for boundary scan chain. 16
Figure 2-4: Example of a shift register used on the Instruction Register. 18
Figure 2-5: Shift Register Architecture with serial and parallel hold ranks. 18
Figure 2-6: The Bypass Register. 19
Figure 2-7: Boundary-Scan Register and Boundary-Scan Cell overview [BSH-PARKER-USA, 1998]. 20
Figure 2-8: A Bidirectional pin with separated boundary scan cells for input, output and enable [BSH-PARKER-USA, 1998]. 21
Figure 2-9: CMOS Image Sensor Pixel Organization. 26
Figure 2-10: CMOS APS pixel basic structure. 27
Figure 2-11: CMOS Image Sensor Architecture with digital readout. 28
Figure 2-12: CMOS Image sensors generic block diagram. 30
Figure 3-1: Prototype block diagram. 34
Figure 3-2: XFAB XC018 CMOS process layer stack example [XFAB-XC018]. 37
Figure 3-3: IC Design Workflow. 39
Figure 3-4: TAP State Machine Single Stage Logic. 40
Figure 3-5: Signal flow in simulation. 42
Figure 3-6: TAP State Machine Simulation results plot – 1. 42
Figure 3-7: TAP State Machine Simulation results plot – 2. 44
Figure 3-8: TAP State Machine Simulation results plot – 3. 45
Figure 3-9: Timing requirements for TCK signal, TDI and TDO. 51
Figure 3-10: Boundary-Scan Top Simulation - Main Signals. 53
Figure 3-11: Boundary-Scan Top Simulation - Instruction Register Control Signals. 54
Figure 3-12: Boundary-Scan Top Simulation - Data Registers Control Signals. 55
Figure 3-13: Boundary-Scan Top Simulation – Instruction Register Control Signals. 56
Figure 3-14: Boundary-Scan Top Simulation - Bypass Register Control Signals. 58
Figure 3-15: Boundary-Scan Top Simulation - Boundary-Scan Register Control Signals. 59
Figure 3-16: Boundary-Scan Top Simulation - Boundary-Scan Register Outputs. 60
Figure 4-1: Layout example performed on the IC Station tool 64
Figure 4-2: Sensor prototype layout floorplan for Boundary-Scan Block. 65
Figure 4-3: Boundary-Scan Control Block layout floorplan. 65

Figure 4-4: Simulation after layout, main IO signals. 70
Figure 5-1: PCB board with the sensor prototype. 71
Figure 5-2: Sensor Boundary-Scan signals separated from the FPGA Boundary-Scan signals. 71
Figure 5-3: Connection of Sensor Boundary-Scan in series with the FPGA Boundary-Scan. ... 72
Figure 5-4: iMPACT software module with Boundary-Scan Chain Debug mode enable. 73
Figure 5-5: Test Boundary-Scan Design flowchart. 74
Figure 5-6: Serial Vector Format Statement Specification flowchart. 78
Figure 5-7: JTAG signals captured using an oscilloscope 83

List of Tables

Table 2-1: Instruction Register operations during TAP Controller states. 17
Table 3-1: Pin name and description used by the Boundary-Scan circuitry. 40
Table 3-2: Instructions implemented in the Boundary-Scan Design. 48
Table 3-3: Timing Parameters for Boundary-Scan Design. 51
Table 4-1: TAP Controller State Machine Layout Dimensions. 66
Table 4-2: TAP Controller Block Layout Dimensions. 67
Table 4-3: Bypass Register Block Layout Dimensions. 67
Table 4-4: Instruction Register Block Layout Dimensions. 67
Table 4-5: Boundary-Scan Cell and Input Cell Layout Dimensions. 68
Table 4-6: Boundary-Scan Cell and Input Cell Layout Dimensions. 68
Table 4-7: Boundary-Scan Register Layout Dimensions. 68
Table 5-1: FPGA signals under test applied to the sensor and the measured values. 82
Table 5-2: Expected and measured values for each implemented instruction. 82

Glossary

ADC	-	Analogue to Digital Converter
APS	-	Active Pixel Sensor
ASIC	-	Application Specific Integrated Circuit
ATE	-	Automatic Test Equipment
BDSL	-	Boundary-Scan Description Language
BSC	-	Boundary-Scan Cell
BSR	-	Boundary-Scan Register
BST	-	Boundary-Scan Testing
CAD	-	Computer Aided Design
CCD	-	Charge Coupled Device
CDS	-	Correlated Double Sampling
CIS	-	CMOS Image Sensors
CMOS	-	Complementary Metal-Oxide Semiconductor
CVC	-	Charge to Voltage Converter
DRC	-	Design Rule Check
DSC	-	Digital Still Camera
DSLR	-	Digital Single Lens Reflex
EDA	-	Electronic Design Automation
ESD	-	Electrostatic Discharge
I2C	-	Inter-Integrated Circuit
IC	-	Integrated Circuit
JEDEC	-	Joint Electron Device Engineering Council
JETAG	-	Joint European Test Action Group
JTAG	-	Joint Test Action Group
LSI	-	Large System Integration
LVDS	-	Low-voltage differential signalling
LVS	-	Layout Versus Schematic
MOS	-	Metal Oxide Semiconductor
MOSFET	-	Metal Oxide Semiconductor Field Effect Transistor
PCB	-	Printed Circuit Board
SoC	-	System-on-Chip
SPI	-	Serial Peripheral Interface Bus
SPICE	-	Simulation Program with Integrated Circuit Emphasis
SVF	-	Serial Vector Format
TAP	-	Test Access Port

TCK - Test Clock

TDI - Test Data In

TDO - Test Data Out

TMS - Test Mode Select

TRST - Test Reset

VHDL - VHSIC Hardware Description Language

1. Introduction

Experiment and test are intrinsic to the Human being. Since the beginning of the existence, every time that a tool was developed, it was tested to verify if it fulfilled the expected functions. If not, the tool was changed and improved, until it fits perfectly on the desired functions.

The same idea is applied to Printed Circuits Boards or PCBs and Integrated Circuits or ICs. Since its first developments, PCBs and ICs have brought great progress to the industry, bringing high performance and complexity to systems. Over the years, many challenges were solved by testing the PCBs and ICs, analysing the results and improving new designs [BST-AST, 2007][BSH-PARKER-USA, 1998].

Presently, PCBs and ICs have become very complex and the need for thorough testing is more and more important. Together with accurate simulation models and testing techniques, IC designs can be implemented into smaller silicon areas, decreasing cost and increasing integrated the circuits complexity [BST-AST, 2007][BSH-PARKER-USA, 1998].

Early advances in surface mount packaging and PCB manufacturing have resulted in smaller PCB boards and ICs packages, which means that traditional test methods based on direct pin probing with oscilloscopes or digital analysers are difficult to implement. Besides, those methods rely on pin-by-pin testing, therefore, and considering that most ICs have hundreds of pins and thousands of ICs have to be tested, the test procedure becomes a lengthy process. As a result, cost savings due to PCB space reductions where spent on the increased cost of the traditional testing methods, bringing no benefit to the companies. In the 1980s, a work group was formed, named Joint Test Action Group – JTAG, to develop a specification for connectivity tests, in order to find a solution to testing problem. The reached solution is based on test the interface between the IC system core and the PCB connections. Thus, performed on the IC boundary and therefore are called *Boundary-Scan Testing* or BST. The *Boundary-Scan* was later standardized as the IEEE Std. 1149.1 specification [BST-AST, 2007]. This BST design offers the capability to test efficiently components on PCBs with tight lead spacing [BSH-PARKER-USA, 1998].

BST design tests pin connections without using physical test probes and captures functional data while a device is operating normally. This type of test had inherent mechanical problems when the probe was not correctly attached to the pin: if the tester

applied too little force on the probe, the connection could fail. If the tester applied too much force it could damage the bond connection. On the other hand, small pins required small probes, increasing the mechanical problems.

Boundary-Scan Testing uses small blocks, in between the pin and the system core logic, called *Boundary-Scan Cells,* to force or sense logic signals onto pins or the system logic. The sensed test data is then serially shifted out and externally compared to expected results. *Figure 1-1* illustrates the concept of BST [Std1149.1-Strx-AltCorp, 2007].

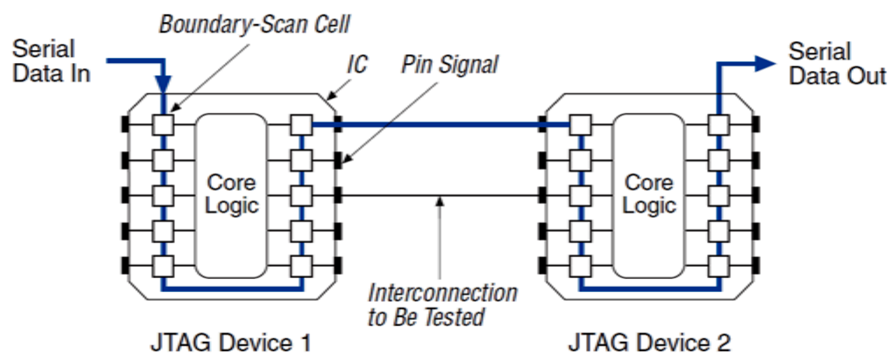


Figure 1-1: IEEE Std. 1149.1 Boundary-Scan Testing [Std1149.1-Strx-AltCorp, 2007].

Shift registers can be connected to several devices in, series to perform a chain test using the same control signals in all devices. For this reason this method is also called *Boundary-Scan Chain* when used to test a series of IC devices.

One technology used for constructing integrated circuits is the Complementary Metal Oxide Semiconductor or *CMOS*. This technology was developed and presented by C. T. Sah and F. Wanlass of the Fairchild R & D Laboratory. They showed that logic circuits combining p-channel and n-channel MOSFET transistors in a complementary symmetry circuit configuration drew close to zero power in standby mode.

CMOS technology is designed for standard digital and analogue designs, but, regardless of the specificity of photosensitive designs, it also permits the design of image sensors, and this category of sensors has been the subject of extensive development recently, leading to growth in terms of market share. Besides CMOS technology, image sensors can be designed and produced using another technology known as Charge Coupled Device or *CCD* technology. Presently the most common image sensing technologies are CCD and CMOS, however there are other technologies that are outside the scope of this work. CCD devices have been dominating the field of image sensors for a long time but, they are now losing their market share to CMOS technology devices [S-CIS-OHTA-USA, 2008].

Due to the lower manufacturing cost and the possibility to integrate processing functionalities on the chip, CMOS image sensors are currently widely used on consumer electronics, such as mobile phones cameras, digital still cameras (DSC), compact cameras, handy camcorders, digital single lens reflex cameras (DSLR), as well as in industrial applications, such as automobiles, surveillance, security and robotic machine vision. More recently, due to technologic advantages, these image sensors started to be used in fields such as medicine and biotechnology. Many of these applications require advanced performance, high speed, high dynamic range and high sensitivity. Other more specific applications, besides the performance and speed, require some special functions, such as real time tracking, laser triangulation and three-dimensional range finding. It is difficult and in some cases impossible to perform such tasks with conventional image sensors based on CCD technology. Besides, some signal processing devices directly connected to CCD image sensors, are insufficient for these purposes. CMOS image sensors with integrated smart functions on the chip, can accomplish the requirements for these applications [S-CIS-OHTA-USA,2008]. For the specific applications mentioned above, these sensors are called Application Specific Integrated Circuit – ASIC. In terms of testability, these integrated special functions must not compromise the basic functions of the image sensor. Interconnectivity tests should be the first tests to be performed, to avoid malfunctioning of some of these blocks or a complete failure. The evolution of standard image sensors to advanced System on Chip or SoC, devices requires more advanced and reliable test techniques.

1.1. Motivation of the Boundary-Scan Architecture

Digital logic testing is performed since the first digital systems where designed, because it was realized early that volume production of digital boards and systems would not be economically sustainable without some type of formalized testing. Furthermore, testing solutions should be fulfilled with relatively inexperienced labour, to free the designers for new projects [BSH-PARKER-USA, 1998].

In the middle of the 1970s, the common and wide spread testing method of loaded PCBs relied heavily on the use of the so-called in-circuit "bed-of-nails" technique [BSH-PARKER-USA,1998] (See *Figure 1-2*). In the picture it is possible to identify a Test Platform connected to the device using a group of nails. Those nails where aligned

with the IC pins to apply the stimulus signals and sense the respective results from the Device Under Test.

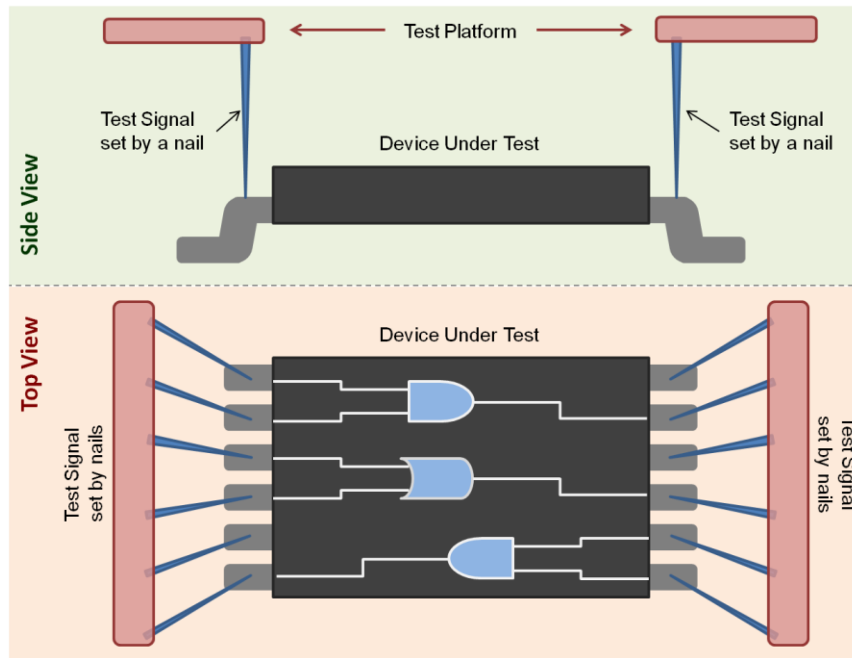


Figure 1-2: Functional test using the "bed-of-nails".

This testing method uses a test platform with a group of nails to access individual device pins on the PCB board, IC pads placed into the copper interconnect, or other convenient contact points. The tests are produced in two phases: *power-off* tests and *power-on* tests. The *power-off* tests check the integrity of the physical contacts, looking for short or open circuits between the nails and the board. The short and open test results are based on impedance measurements: if the impedance is low, the contact exists, and if the impedance is high, then there is no contact to the pin. The *power-on* tests apply stimulus to a specific device on the board, measure the device outputs and compare them with a pre-defined pattern settled on device design. Other devices electrically connected to the Device-Under-Test are usually placed on a protected state, to avoid damage, a process called "guarding". With the explained test process, the tester can evaluate the presence, the direction and bonding consistency of the Device Under Test present on the board.

Mainly, the in-circuit bed-of-nails technique relies on physical access to all pins and test-points of all devices present on the board. Typically, for these tests, the access to the device is made on the back side of the boards where the devices are placed. Usually called side "B". However, due increasing complexity and the need to reduce the occupied space, both sides of the board began to be used, obtaining higher density. Also

due to space concerns, the size of the copper tracks and the spacing in between the IC pins became significantly smaller. This had a major impact on the ability to place a nail accurately aligned with the test point to measure. Besides the test point size, also the space in between the test-points became smaller, forcing the nails to be thinner and closer. The problem increased even more with the development of multi-layer boards, were the test-points where even more inaccessible [BST-AST, 2007].

To find a solution to their problem, in the middle of 1980s, a group of concerned European electronic systems engineers, and their companies, explored together this problem. Primarily called Joint European Test Action Group – JETAG, the group changed its name to Joint Test Action Group or JTAG, after North American companies had joined the group. JTAG found a solution based on the concept of a serial shift register around the interface in-between the pads and the system or the boundary of the device – hence the name "*boundary scan*" [BSH-PARKER-USA, 1998]. In 1990, this organization converted its ideas into an international standard – the IEEE Standard 1149.1 [BST-AST, 2007]. The IEEE Standard 1149.1 is a testing standard. Actually, it is also a collection of *design rules*, applied to Integrated Circuits level.

The standard relies on two major operating modes: *Non-Invasive* and *Pin-Permission*. The *Non-Invasive* mode uses resources defined by the Standard and they are independent from the rest of the logic inside the IC. In this mode the resources are used to communicate asynchronously with the outside world to perform test or read test pattern results. These actions are invisible to the IC when it is in normal operation. Similarly to the *Non-Invasive* mode, the Standard has defined specific resources to *Pin-Permission* modes, however those resources can change states using instructions that can take control of the IC inputs and/or outputs, change the System Logic states defining its operation and disconnect it from the outside world. These operating modes allow the testing of the ICs System Logic or the IC connectivity to other ICs or pins on the PCB [BSH-PARKER-USA, 1998].

In addition, the Standard allows designers to add modes of operation. This extensibility is especially useful for several operations, such as loading firmware code to program the System Logic, running self-test modes or other specific operations that may help to test or operate the IC.

2. Fundamentals of Boundary-Scan Chain and Image sensors

This chapter describes the Boundary-Scan Chain architecture, focusing on the functional blocks and the features described on the IEEE Standard 1149.1. Section 2.1 analyses the details of the *Boundary-Scan* architecture, explaining the Test Access Controller block as the fundamental timing generator for all *Boundary-Scan* test process, the Instruction and Data Registers. There is, also, an explanation of the mandatory and optional Registers, the mandatory instructions to control the register in use, the optional instructions used for extended tests over the IC and the modes defined on the Standard IEEE 1149.1. Section 2.2 introduces the most recent Image Sensor Architecture, using CMOS technology, designed for high speed cameras. These sensors have digitalization and image processing on chip, that allows high speed and performance. However, with a high number of pins, the connectivity test of these ICs becomes, a problem if a scheme such as Boundary-Scan is not used.

2.1. The Boundary-Scan Chain Architecture

The basic architecture of IEEE 1149.1 Boundary-Scan describes a design that is fully integrated within the Integrated Circuit and should be interpreted as an extension of it and designed with it. As is illustrated on *Figure 2-1*, the architecture has four mandatory package pins named *Test Data Input – TDI*, *Test Data Output – TDO*, *Test Mode Select – TMS*, *Test Clock input – TCK* and an optional *Test Reset Input – TRST*. These additional pins, together with the *Boundary-Scan* controller and the implemented registers must be added to the IC design. The pins are connected internally to the *Test Access Port* or TAP controller and to the registers. The basic mandatory registers are the *Boundary-Scan Register*, the *Device ID Register*, the *Bypass Register* and the *Instruction Register*. Other registers can be added depending on the needs of the designer and the test functionalities implemented. Each register is selected using specific instructions.

The most important of the implemented blocks is the *Test Access Port* – TAP Controller. This block is responsible for the control of the mode of the *Boundary-Scan* and to select the test pattern path from input to output. This controller is connected to all implemented registers, being the principal register the *Boundary-Scan Register.* This register is a group of *Boundary-Scan Cells* implemented in each input or output pins to be tested by the *Boundary-Scan Chain*. The *Boundary-Scan Cell* is a multi-purpose

memory element that is responsible for the application of the *Boundary-Scan* modes to the signal, and is placed in between the signal pin and the system Logic. The other register is the *Device ID Register,* that is an optional register that identifies the IC and has a unique code. Every IC Manufacturer has to request an identification code to the JTAG Group.

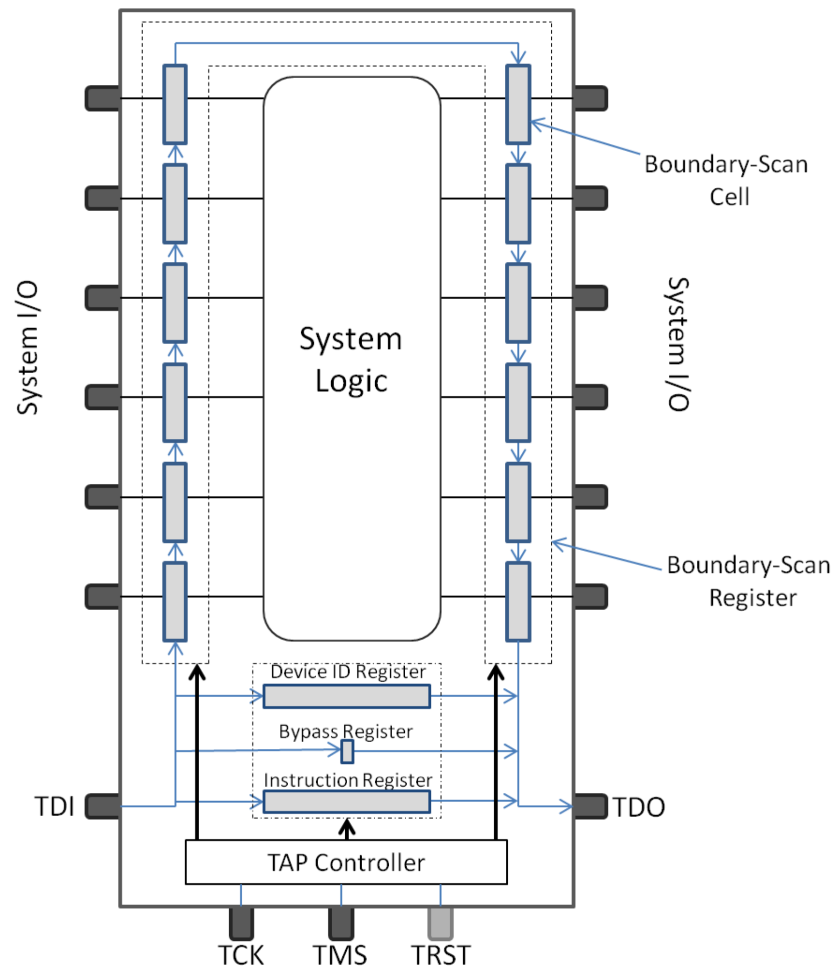


Figure 2-1: Simplified Boundary-Scan Architecture of an 1149.1 compliant Integrated Circuit.

The *Bypass Register* is a mandatory register and permits a bypass the *Boundary-Scan Register* when the IC is not in test mode. The *Instruction Register* is another mandatory register, which stores and decodes the instructions to be applied by the *Boundary-Scan Chain*. Its size is dependent on the amount of instructions implemented.

The pins *Test Data In – TDI*, *Test Data Out – TDO, Test Clock – TCK, Test Mode Select – TMS* and the optional *Test Reset – TRST* are assigned to the TAP controller and should not be shared with other functions on the IC. The *Boundary-Scan Logic* is accessed using a dedicated protocol of communication according to the Standard [BSH-PARKER-USA, 1998].

The cells that are connected to input signals are identified as *input cells* and the cells that are connected to output signals are identified as *output cells*. The designation of *input* and *output* is relative to the IC System Logic and is used in the definition of the direction whenever the IC is communicating with the neighbours [BST-AST, 2007].

Functionally, the *Boundary-Scan Architecture* is a system consisting of two paths: a serial shift that shifts test pattern data along all register cells and the parallel shift that loads in parallel to the shift register or writes in parallel from the shift register. The serial shift along the register is called *"shift"* operation, while the load in parallel is called *"capture"* operation and the write in parallel is the *"update"* operation. In the case of *input* cells, this operation, applies the signal values on the output scan cells to the system core. In the case of *output* cells, it applies the signal values to the output pin. During the analysis of the state machine these operations will be explained in detail.

The pattern data introduced into the *Test Data In* – TDI pin is shifted into the *Boundary-Scan Register* that is distributed around the sensor or along other specific registers. The *Test Data Out* or TDO pin is the output from the registers, where the data is shifted out. These two pins works together: while the data is shifted in through the TDI, the TDO shifts out the content of the *Boundary-Scan Register*. The dedicated pin *Test Clock* or TCK is the clock reference signal used by the shift operation and the reference clock to all the registers implemented and to the *Boundary-Scan* control logic or *TAP controller*. In order to control the mode of operation, a dedicated pin called *Test Mode Select* or TMS, consists of a serial control signal connected to the TAP controller [BST-AST, 2007].

2.1.1. The Test Access Port – TAP Controller

The *Test Access Port* – TAP controller is a finite state machine, with sixteen states, responsible for the control of Boundary-Scan Test operating modes. When the objective is to enter the state machine into the reset state, the jumps are synchronous with TCK or asynchronously with the assertion of the TRST, if it is present. The transition between states is defined by the logic level of the TMS signal at the rising edge of the TCK clock signal.

The TAP controller is responsible for the control of all registers on the design and is based on instructions to perform predefined actions. Thus the controller is connected to an *Instruction Register* and to the *Data Registers*. Besides the Instruction Register,

the *Data Registers* are all the registers, such as *Bypass, Boundary-Scan, Device ID,* optional registers and user defined registers.

Figure 2-2 shows the state diagram of the TAP controller. In the diagram, the logic values drawn in between the states represent the logic levels of the TMS signal at the rising edge of TCK. In the diagram, one identifies two similar state flows. The only difference relies on the state names, which differ from one column to the other in the suffix ‘DR’ or ‘IR’. ‘DR’ means Data Register, and ‘IR’ means Instruction Register. With this designation it is easy to see that the Instruction Register implemented on the Boundary-Scan has the same basic functionalities as all Data Registers implemented.

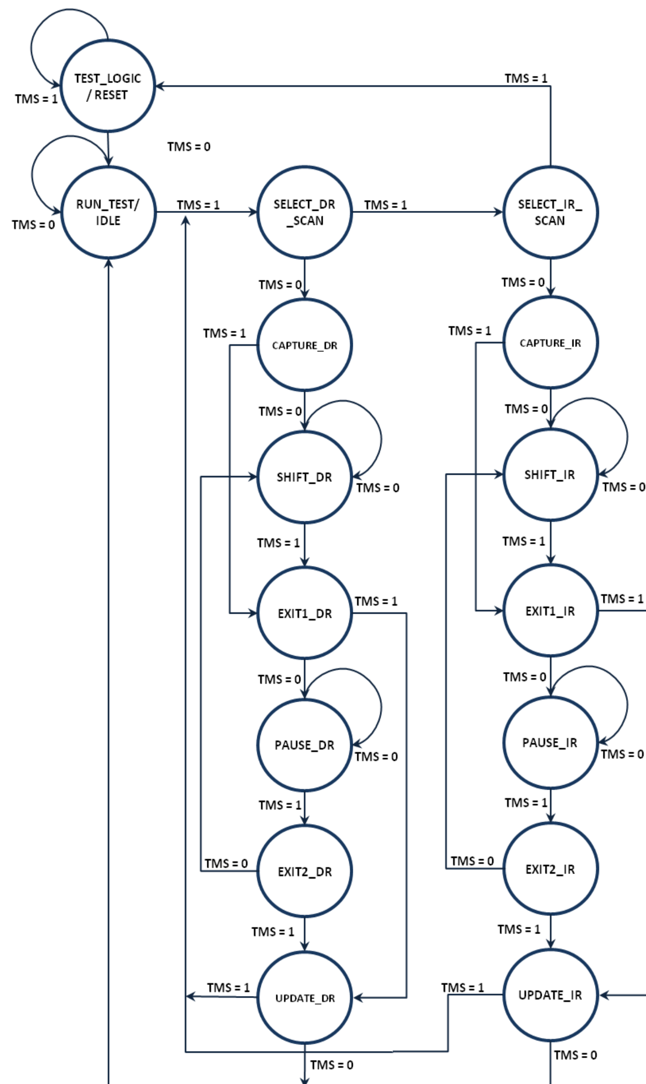


Figure 2-2: TAP Controller state machine diagram.

In the following, an overview of the states implemented on the state machine is presented.

TEST-LOGIC/RESET

This is the reset state. On this state the Boundary-Scan Logic is in reset state and is transparent to the IC's Logic System, which can operate normally. In the reset state, the Instruction Register contains the *ID CODE* instruction, if the Device Identification Register is present or the *Bypass* instruction if the Device Identification Register is not present. Regardless of the controller original state, the Test-Logic/Reset state is selected when the TMS is high during at least five rising edges of the TCK clock. This state is also selected when the TRST is asserted, if present. The controller remains in this state if the TMS is high at the rising edge of the TCK clock, otherwise it moves on to the Run_Test/Idle state [BSH-PARKER-USA, 1998].

RUN_TEST/IDLE

The TMS signal is set low to enter this state, and it remains on this state while it is held low. As soon as the TMS signal goes high, at the next rising edge of the TCK, the controller moves to the *Select_DR_Scan* state. In the *Run_Test/Idle* state the test logic becomes active if some self test modes are implemented. If the self-test modes are implemented, they are executed in this state and when the controller leaves the state, the result is stored on the associated registers [BSH-PARKER-USA, 1998].

SELECT_DR_SCAN

According to the Standard, this state is called a "temporary controller state". This means that it will exit this state on the next TMS evaluation. Thus, this state was designed to decide if the state machine enters in to the Data Register rank or moves on in to Instruction Register rank. Thus, if TMS is low at next evaluation, the controller moves to *Capture_DR* state, otherwise the controller moves to *Select_IR_Scan* state [BSH-PARKER-USA, 1998].

SELECT_IR_SCAN

This is another temporary controller state. Depending on the TMS state, the controller returns to the *Test_Logic/Reset* state or move to the Instruction Register rank. Thus, if the TMS remains high, the controller moves on to *Test_Logic/Reset* state and if it goes low, the controller moves on to the *Capture_IR* state [BSH-PARKER-USA, 1998].

CAPTURE_IR

As introduced before, the Register cells implement three main features: capture, shift and update. In a simplified view, these features capture the data that comes in to the cell, shift the captured data to the TDO output or update the data that is inserted on the TDI input. These features will be better explained on the section 2.1.3.3, where the Boundary scan cell is explained. The *Capture_IR* state permits the controller to parallel load a pre-defined pattern on the rising edge of TCK [BSH-PARKER-USA, 1998]. The *Instruction Register* size depends on the instruction set implemented. The smallest instruction set has only 3 instructions, thus two bits are needed to address each of them. If the Instruction Register is only 2-bit size, the pre-defined pattern is “01” otherwise the two least significant bits are assigned to “01”. The other bits should be fixed to known value or design specific value. This pattern is important, because it permits the execution of an integrity test of the Instruction Register circuitry. In this state, the controller enters either the *EXIT1_IR* state, if TMS is high, or the *SHIFT_IR* state, if TMS is low [BSH-PARKER-USA, 1998].

SHIFT_IR

When the controller enters in to this state, the Instruction Register is connected to TDI and TDO, starting to shift on the rising edge of the TCK. Once the TMS is held low, the data introduced on TDI is shifted into the Instruction Register and the generated fixed pattern set on the *Capture_IR* state is shifted towards TDO. If the TMS is high, on the next rising edge of TCK, then the controller moves to *Exit1_IR*, otherwise it remains on this state [BSH-PARKER-USA, 1998].

EXIT1_IR

This is another temporary controller state. Once on this state, if the TMS is held high then the controller moves to *Update_IR* state. If the TMS goes low, the controller moves to *Pause_IR* state [BSH-PARKER-USA, 1998].

PAUSE_IR

This controller state is used when it is needed to halt the Instruction Register temporarily. While the TMS signal is held low the controller remains on this state, and when it goes high, it enters the *Exit2_IR* state [BSH-PARKER-USA, 1998].

EXIT2_IR

This state is also a temporary controller state. If TMS is high, the controller enters on the *Update_IR* state. Otherwise, it returns to the *Shift_IR* state.

The flow *Shift_IR* → *Exit1_IR* → *Pause_IR* → *Exit2_IR* → *Shift_IR* is used if an external controller is loading instruction bits but does not have enough memory depth to complete the entire shift sequence in one burst. The shift sequence can be divided into manageable portions by passing to the *Pause_IR* state while the next portion of shift data is prepared. If the external controller has memory enough, when the controller arrives on the *Exit1_IR* state the state machine can move on to the *Update_IR* state [BSH-PARKER-USA, 1998].

UPDATE_IR

In this state, the instruction previously shifted into the *Instruction Register* is latched on the falling edge of TCK. After the latch, the instruction becomes the current instruction, leading the controller to a new operational mode. Once in this state, if TMS is held high, the controller moves to the *Select_DR_Scan* state. Otherwise, the controller moves to the *Run_Test/Idle* state [BSH-PARKER-USA, 1998].

CAPTURE_DR

This state performs the parallel loading of the data present at the input of the cell. If the TMS is high in this state, the controller enters in to the *Exit1_DR* state and if TMS goes low, the controller enters in to the *Shift_DR* state [BSH-PARKER-USA, 1998].

SHIFT_DR

Once in this controller state the TDI is connected to the TDO. The data is shifted on the rising edge of the TCK. The TDO shifts out the data from the active register and the TDI feeds the input data to the register. In this state, the controller enters either the *Exit1_DR* state, if TMS is held high, or remains in the *Shift_DR* state, if TMS is held low.

It is possible to return to *Shift_DR* by passing through the *Exit1_DR* → *Pause_DR* → *Exit2_DR* states. This is important if an external controller is loading data bits but does not have enough memory depth to complete the entire shift sequence in one burst. The shift can be separated into manageable data frames, by passing to the *Pause_DR* state while the next block is prepared [BSH-PARKER-USA, 1998].

EXIT1_DR

This is a temporary controller state. At this point, a decision must be made whether to enter the *Pause_DR* state, or the *Update_DR* state. If TMS is held high while in this state, the controller enters the *Update_DR* state. If the TMS is held low, the controller enters the *Pause_DR* state.[BSH-PARKER-USA,1998]

PAUSE_DR

This controller state allows temporarily halting the shifting of the test data register in the serial path between TDI and TDO. The controller remains in this state while TMS is low and when TMS goes high, the controller moves to the *Exit2_DR* state.[BSH-PARKER-USA,1998]

EXIT2_DR

This is another temporary state. Once again the decision must be made whether the controller moves to the *Update_DR* state, or returns to the *Shift_DR* state. If TMS is held high when evaluated, the scanning process terminates and the controller enters the *Update_DR* state. If TMS is held low, the controller enters the *Shift_DR* state.[BSH-PARKER-USA,1998]

UPDATE_DR

The *Update_DR* is the state that terminates the scanning process. Once on this state, some of the implemented registers can latch data that was shifted during the *Shift_DR* process. This state exists to permit that the output of those registers is latched in a specific moment in time. Thus, this latch is performed once in this state and at the falling edge of TCK. On this state, the controller enters either the *Select_DR_Scan* state, if TMS is high, or the *Run_Test/Idle* state if TMS is low [BSH-PARKER-USA, 1998].

The analysis of the TAP Controller transition states is concluded with the sequence of actions that the *Boundary-Scan* performs.

- Both shift states, *Shift-IR* and *Shift-DR,* activate the output driver for the TDO pin. This driver remains active until the falling edge of TCK, when the state controller is in the *Exit1-IR* or *Exit1-DR* states respectively. In all other states the TDO driver is turned off, forcing the TDO pin to be in the high impedance state.

- In either update state, *Update-IR* or *Update-DR*, the data transfer process from the shift register to a hold register occurs on the falling edge of TCK.
- In either capture state, *Capture-IR* or *Capture-DR*, the data is captured by the shift portion of the register between TDI and TDO, on the rising edge of TCK. Because this edge causes the TAP controller to leave the capture state, the data is captured on either state transition leaving the capture state.
- Data is shifted out from the register to the TDO on the TCK falling edge, in either of the two shift states. Note, however, that data is shifted in from the TDI on the rising edge. This yields two effects:
  - Data is shifted out when the state machine leaves a shift state. A common mistake is to associate shifting with the state and not the state transition. When one wants to shift one last bit into a register, it must take the arc that goes to *Exit1-IR* or *Exit1-DR.* No data is shifted by the rising edge of TCK, that first brings the TAP controller into a shift state from either *Capture-DR* or *Capture-IR.*
  - The data that might be present on the TDO when first entering a shift state will not be valid until after the first falling edge of TCK. Data is set up on the TDO a half TCK cycle before TDI is read for the first time [BSH-PARKER-USA, 1998].

2.1.2. The Instruction Register

The Instruction Register defines the mode in which *Boundary-Scan* data registers will operate. Like other registers in the IEEE 1149.1 Standard specification, the IR is a shift register. However, the *Instruction Register* can have a logic decoder and an hold register that stores the instruction until the next instruction scan. The scan register is loaded in parallel at the *Capture-IR* state, shifted between TDI and TDO at the *Shift-IR* state and the content of the scan register is transferred to the hold register at the *Update-IR* state.

Each Instruction Register cell comprises shift register flip-flops and a parallel output latch. The shift register holds the new instruction bits moving through the register flip-flops. The latch holds the current instruction in place while no shift is performed. This prevents “shift ripple” from being observed at the register parallel hold outputs during shifting. This ripple-free behaviour is important to many Boundary-Scan applications.

Mandatory and optional instructions are defined by the IEEE Standard 1149.1. Design-specific instructions can also be added to a component by the designer. The minimum size of the Instruction Register is two register cells, and the size of the register dictates the size of the instruction codes that can be used: the instruction number of bits must match the length of the Instruction Register [BSH-PARKER-USA, 1998].

As shown on *Figure 2-3*, the two least significant register cells must capture a fixed binary "01" pattern during controller state CAPTURE-IR. These bits are used to test the logic integrity. The higher-order bits of this register, if they exist, may capture fixed or variable bits, depending on design specific applications. The number of instructions decoded must be compatible with the number of register cells present on the Scan Register. For example, in case there were 10 register cells implemented, the instruction logic can decode up to 1024 different instructions.

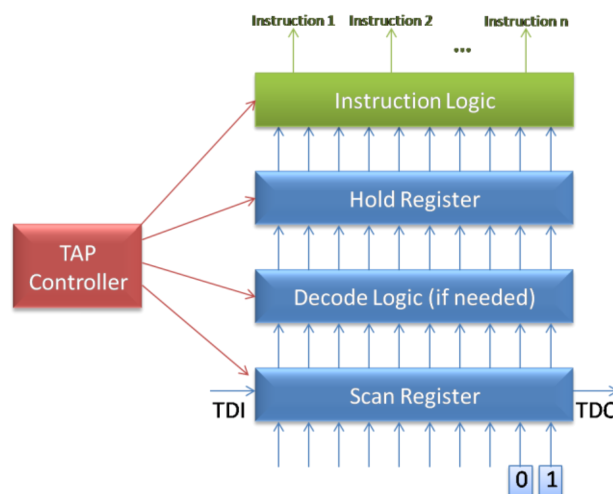


Figure 2-3: Instruction register architecture detail for boundary scan chain.

The instruction shifted to the shift register flip-flops is latched into the decode logic or, if no decode logic is implemented, directly into the hold register outputs, when the shifting process is completed. This operation occurs during the *Update-IR* state and ensures that the instruction changes only at the end of the Instruction Register (IR) shifting sequence. At the output of the *Update-IR* state, the Instruction Logic has the selected active instruction, defining the test mode and the test data register to be accessed [BSH-PARKER-USA, 1998].

When a reset action is performed, by operating the *Test Reset* pin – TRST or entering in to the *Test_Logic/Reset* controller state, one of two instructions must be latched onto the Instruction Register outputs in order to be the active instruction when

the State Machine leaves the reset state. Thus, if the IC has a Device Identification Register, then the instruction to be loaded is the *IDCODE* instruction, and its corresponding bit pattern must be loaded onto the Hold Register (please see *Figure 2-3*). Otherwise, the *Bypass* instruction is loaded. *Table 2-1* shows the behaviour of the *Instruction Register* during each TAP Controller state, when one of those instructions is being loaded [BSH-PARKER-USA, 1998].

The *Scan Register Architecture* shown on *Figure 2-3* is the typical architecture for a shift register where data is inserted serially in to the chain and shifted out at the Clock_IR frequency to the output. *Figure 2-4* shows a more detailed structure, representing a shift register used on the instruction register. In this example, it is possible to observe the TAP Controller signals used to control the shift of the data [BSH-PARKER-USA, 1998]. The *"Capture Data"* and the *"Instruction Bit"* are the parallel input and output signals, respectively. The signals *"Previous cell or TDI"* and *"To Next Cell or TDO"* are the serial input and output of the Instruction Register shift register. The Clock_IR is the dedicated clock signal from the TAP Controller to perform the shift and capture the data.

TAP Controller State	Shift Register Flip-Flops	Hold Register
TEST_LOGIC/RESET	Undefined	Set to the device IDCODE if implemented or to the BYPASS in case of no Device ID Register implemented.
CAPTURE_IR	Load '01' into the LSBs and other design specific bits into MSB's	Keep the last state.
SHIFT_IR	Shift the instruction bits towards the register	Keep the last state.
EXIT1_IR EXIT2_IR PAUSE_IR	Keep the last state	Keep the last state.
UPDATE_IR	Keep the last state	Loads the data from the shift register to be decoded. Hold the values at the outputs.
Other states	Undefined	Keep the last state

Table 2-1: Instruction Register operations during TAP Controller states.

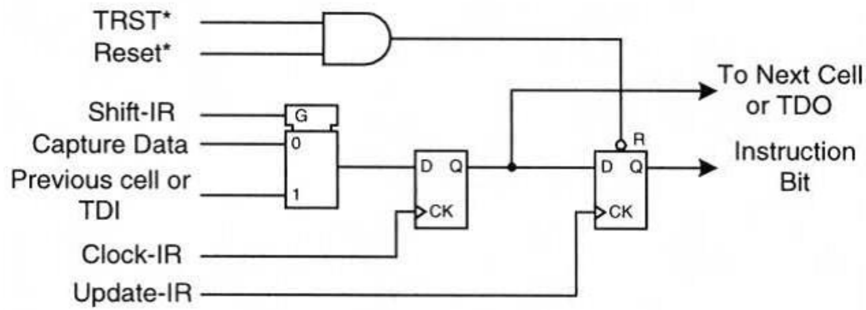


Figure 2-4: Example of a shift register used on the Instruction Register.

The signal *"Update_IR"* latches on the falling edge of the TCK and performs the update operation to hold the shifted instruction. The signal *"Shift_IR"* is high (as '1') when the TAP Controller is in the SHIFT_IR state. The signal *"Reset"* is high when the TAP Controller is in *Test_Logic/Reset* state. The pin TRST is asserted asynchronously and clears (or presets) immediately the state of the *Hold Register*. After this the initial instruction is loaded on the Hold Register (the *Bypass* or the *IDCODE*) [BSH-PARKER-USA, 1998].

2.1.3. The Data Registers

Instructions on the *Boundary-Scan* are designed to prepare the next test mode. The fundamental purpose of the Boundary-Scan is to set the operational mode that place the data in a shift register between the TDI and TDO[1]. The instruction determines the proper modification on the test logic and the *Instruction Register* is a shift register closing the path between the TDI and TDO pins. *Figure 2-5* shows the generic architecture of a shift register.

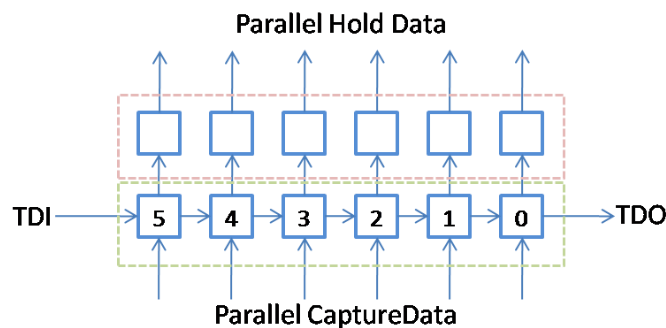


Figure 2-5: Shift Register Architecture with serial and parallel hold ranks.

Some of the registers used on the Boundary–Scan can be simplified because they don't need the *Hold Register* represented. The registers that are used only to store a bit

[1] Unless specific designs that use different architecture and do not violate the principle of the Boundary Scan Architecture.

pattern and do not control any other block belong to the group of simplified registers without an *Hold Register* [BSH-PARKER-USA, 1998].

2.1.3.1. Bypass Register

This register is mandatory in the *Boundary-Scan* architecture. This is a simple register that does not require the parallel hold rank. This register consists of only one scan cell, as show on *Figure 2-6*. When the register is selected by the *Bypass* instruction, the *Bypass Register* shortens the shift path within an IC to a single cell. This is useful for reducing shifting time when testing other boundary-scan components on a board. For example, if there are three *Integrated Circuits* on the PCB, connected by a *Boundary-Scan Chain,* and each one has 100 testable IO's on the chain, when only the third *Integrated Circuit* is selected to be tested, using the *Bypass* instruction on the first and second ICs, the tester only has to generate patterns of 102 bits. This is particularly useful when the number of devices in a chain is long and each device has hundreds o IO's. Another important feature of the *Bypass Register* is the default value when it is addressed. When the TAP controller passes through *Capture_DR*, it is placed on the register output a fixed binary "0" which is subsequently shifted out [BSH-PARKER-USA, 1998].

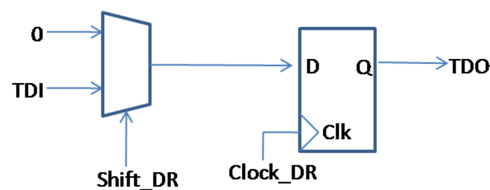


Figure 2-6: The Bypass Register.

2.1.3.2. Device Identification Register

Contrary to the Bypass Register, this is an optional 32-bit length register. The *Device Identification Register* contains component identification information. The register holds two functions: the *IDCODE* instruction and the *USERCODE* instruction. The *IDCODE* instruction is a unique code with different fields: the bits 31 to 28 (four bits) store the *Version Number* of the IC; bits 27 to 12 (sixteen bits) are reserved for the Manufacturer Part Number, assigned by the manufacturer; bits 11 to 1 (eleven bits) are the Manufacturer's Identify Number[2], finally bit 0 (one bit) is reserved, mandatory and fixed as '1' logic. The *USERCODE* has the same length as the *IDCODE*: 32 bits; but the captured data upon passing through the *Capture-IR* state is user-defined. The

[2] This identification number is registered on the JEDEC – the Joint Electron Device Engineering Council – and is unique for each Manufacturer.

USERCODE is an optional instruction and no specific pattern is specified for it. This register is also simple, without and *Hold Register* requirement.

When the TAP Controller passes through the *Capture-DR* state, the *Device Identification Register* will parallel load a fixed 32-bit identification code to be shifted out. This code is useful for chain integrity testing, as well as for simply identifying the IC [BSH-PARKER-USA, 1998].

2.1.3.3. Boundary-Scan Register

This register is the most important and has a boundary-scan cell adjacent to each digital system input and digital system output pin (but not the TAP Pins). This register is used to control and observe activities on the IC's input and output pins. It is a mandatory feature of the IEEE 1149.1 Standard. *Figure 2-7* shows an example of a single data register cell and an overview of the Boundary-Scan Register implementation. The cell design shown is flexible enough to permit the cell to be used as an input or output cell.

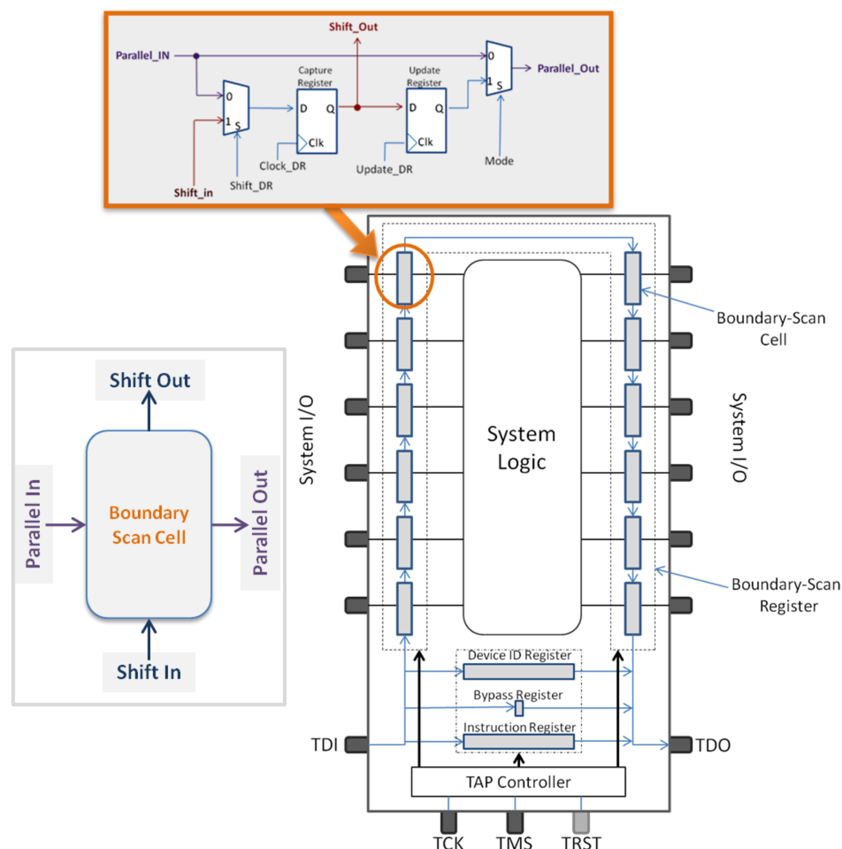


Figure 2-7: Boundary-Scan Register and Boundary-Scan Cell overview [BSH-PARKER-USA, 1998].

The "Parallel In" and "Parallel Out" labels in the signals are connected to the device pin or system circuitry, depending on the pin type and the role of the cell. For

example, if the cell serves an input pin, then the Parallel In signal is connected to the device pin and the Parallel Out signal is connected to the system circuitry. For a device output, these assignments are reversed. Regarding the Capture and Update Registers (represented as a D-flip-flop), these components are components of the shift and parallel hold ranks and are important in order to perform the connectivity tests and test functions of this register [BSH-PARKER-USA, 1998].

The signals named *"Shift In"* and "*Shift Out*" are the serial inputs and outputs of the Boundary Register forming the shift path. The *shift_DR, Clock_DR, Update_DR* and *Mode* are control signals from the TAP Controller into the cell. It is already known that the Boundary scan cell can be used together with an input pin and an output pin. Using 3 registers and this principle, it allows the support of bidirectional pins. *Figure 2-8* shows one implementation of a bidirectional pin with output enable support.

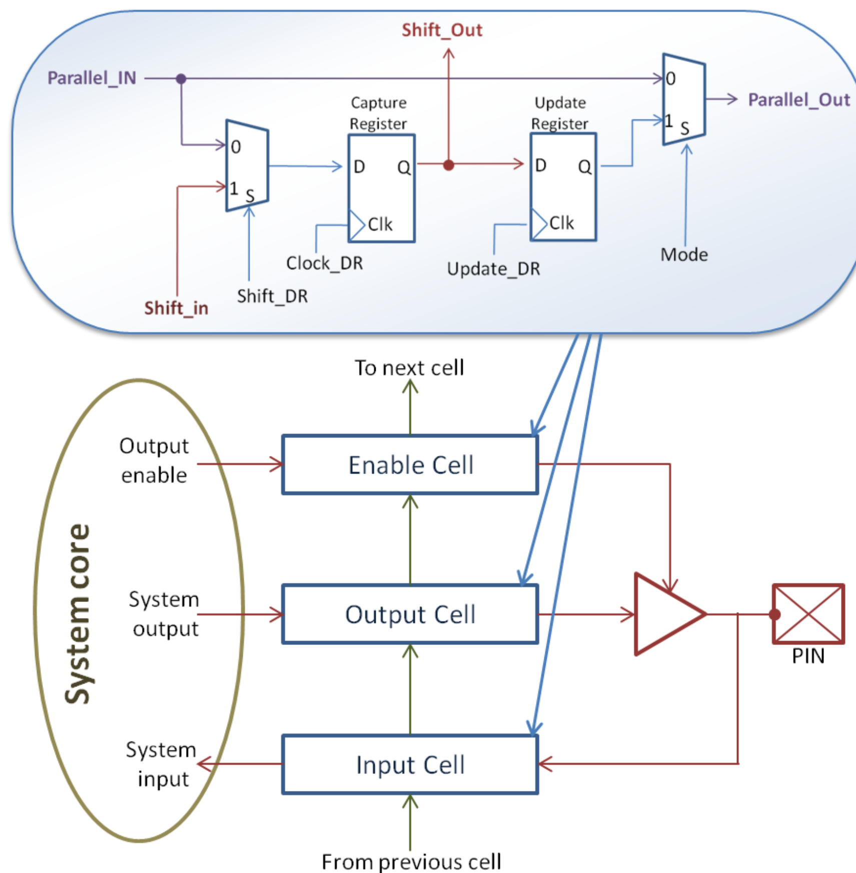


Figure 2-8: A Bidirectional pin with separated boundary scan cells for input, output and enable [BSH-PARKER-USA, 1998].

The same boundary scan cell is implemented on all cells and respects the data direction, if the signal is a system input or output. Besides the presented implementation for bidirectional pins, other implementations can be considered. One of these implementations consists in designing a single cell where the *output enable*, the *system*

output and the *system input* paths are kept. Each of these paths has one capture register and one update register. This is a complex single cell design, with the same functionality as the one above but it can save space for the designer [BSH-PARKER-USA, 1998].

2.1.3.4. User-Defined Registers

The IEEE 1149.1 Standard allows designers to implement user-defined registers. These registers are used in conjunction with user-defined TAP instructions for proprietary built-in self-tests, internal scan testing, or other functions. These registers must form a consistent shift path between TDI and TDO so that whenever selected, the path is not broken [BSH-PARKER-USA, 1998].

2.1.4. Non-Invasive Operational Modes

The TAP Controller and the four, or optionally five, TAP Pins can be operated asynchronously and independently of the System Logic. This allows the Boundary-Scan TAP to be used without disturbing the normal operation of a chip, board or system, as long as non-invasive modes of operation are used. Next the fundamental non-invasive modes for TAP instructions will be presented. Each of these modes corresponds to an instruction shifted and captured by the instruction register [BSH-PARKER-USA, 1998].

2.1.4.1. BYPASS

The *Bypass* mode is caused by the *Bypass* instruction when the all ‘1’s code is written on the *Instruction Register*. This is a mandatory instruction according to the Standard. The *Bypass* instruction causes the Bypass register to be placed between the TDI and TDO pins. By definition, the initialization state of the hold section of the IR should contain the Bypass instruction code, unless the optional *Device Identification Register* has been implemented, in which case, the *IDCODE* instruction code should be present in the hold section [BST-AST, 2007], [BSH-PARKER-USA, 1998].

2.1.4.2. SAMPLE

The *Sample* instruction is another mandatory instruction, but the corresponding code is not predefined on the Standard and it is the user responsibility to choose an appropriate code. This instruction is executed on the *Boundary-Scan Register* and closes the path between the TDI and TDO pins and it is done without disconnecting the System logic from the IC pins. The *Sample* instruction sets up the *Boundary-Scan* cells

to sample (capture) values applied to the device. It takes a snapshot of the cell input values and put them on the serial shift register to be shifted out, via the TDO pin [BST-AST, 2007], [BSH-PARKER-USA, 1998].

2.1.4.3. PRELOAD

The *Preload* instruction is a mandatory instruction and, likewise the SAMPLE instruction, the bit pattern is not predefined and it is executed on the *Boundary-Scan Register*, closing the path between the TDI and TDO pins. The *Preload* instruction is used to shift in, via the TDI, a pre-defined test pattern and to load it to the output *Boundary-Scan* cells prior to the update operation.

It is possible to integrate the *Sample* and the *Preload* instructions in one single instruction, as both operations are complementary. With this, the resultant instruction is the *Sample-And-Preload* operation, which starts by performing the capture of the cells input values, then these values are shifted out while the test pattern to preload is shifted in and finally transfers the preloaded pattern to the cells output. The capture operation is performed on the *Capture_DR* state, the shift is performed on the *Shift_DR* state, the TAP controller remains on this state the same number of clock cycles as the number of register cells, and finally the transfer operation is performed during the *Update_DR* state [BST-AST, 2007], [BSH-PARKER-USA, 1998].

2.1.4.4. IDCODE and USERCODE (Optional)

The *IDCODE* and *USERCODE* instructions are both optional instructions and are stored on the Device Identification Register. Those instructions, already introduced previously, have lengths of 32 bits and although the *IDCODE* format is predefined by the Standard, the *USERCODE* format is not. As a restriction, the *USERCODE* can only be implemented if the *IDCODE* is implemented. The purpose of those instructions is to identify the IC regarding the manufacturer and device version [BSH-PARKER-USA, 1998].

2.1.5. Pin-Permission Operational Modes

The pin-permission mode is a mode in which the pins under test cannot operate in normal mode while the test is being performed (also called as invasive mode). These instructions are characterized by the switching of the cell output multiplexers, so that the update flip-flop data bits are selected and passed on to the parallel outputs of the Boundary Register cells. This disconnects the component I/O pins from the System

Logic. It is important that no harm is done to the System Logic by this radical change of configuration. Thus, on the System Logic, where the signal is used, it may be necessary to design additional logic to force specific holding values, during the test phase, to prevent entering in forbidden states. These operational modes must be provided only with a specific pattern on the Instruction register and the non-invasive mode should have higher priority over the pin-permission modes. This would prevent the System Logic from entering in failure states and eventually permanent damage. Notice that both *Bypass* and *IDCODE* are non-invasive modes, so if a pin-permission instruction is currently active, passing to the *Test-Logic-Reset* state will remove this mode and put the IC's test logic back into non-invasive mode [BSH-PARKER-USA, 1998].

2.1.5.1. EXTEST

The *Extest* instruction is a mandatory instruction. Until the Standard version of 2001, the code for the *Extest* instruction was defined to be the all-'0's code. However that was changed, and since 2001 the code is no longer pre-defined and it is the designer's responsibility to use an adequate code. The *Extest* instruction is executed in the Boundary-Scan Register and connects the TDI to TDO pins. At the *Capture_DR* state, the inputs of the *Boundary Scan* cells are captured. Because the cell output multiplexers are reading the *Update Register* flip-flop, all outputs and output enables are under control of the Boundary Register. *Figure 2-7*, shows the multiplexer *Mode* signal that is set to "1". Thus, during *Extest*, it is possible to sample the IC inputs and control the IC outputs. During *Shift_DR* state, is shifted out through TDO pin a binary word, resultant from the capture of each cell input. At same time the word shifted in through TDI will set up a new cell output, which will become effective upon the TAP State Machine passes through *Update_DR. Extest* instruction is the workhorse of *Boundary-Scan* testing [BSH-PARKER-USA, 1998].

2.1.5.2. INTEST (Optional)

The INTEST instruction is an optional instruction and its bit pattern is not defined on the Standard. Likewise the EXTEST, it targets the Boundary-Scan Register. This instruction puts the System Logic inputs under control of the Update Register flip-flop. This mode applies tests to the core logic acting over its inputs, outputs and output enable pins. The test pattern is entered through the TDI pin and it is applied to the system core inputs, while the system outputs are shifted out through the TDO pin. This

may be a problem if the Boundary Scan Register is too long and the inputs require signals to be asserted within some time constrains [BSH-PARKER-USA, 1998].

2.1.5.3. RUNBIST (Optional)

The *RUNBIST* instruction is an optional instruction and the Standard does not specify an instruction bit pattern for it. *RUNBIST* has a designer-specified target register. The purpose of this instruction is to provide access to the IC's internal built-in self-test with a standardized access protocol. This mode takes effect over the System Logic, generating data stimulus and correcting the responses from those stimulus to shift them out. It is self-initialling and does not require any start-up or seed data. It should run when even the TAP Controller is in *Run_Test/Idle* state, once the result is captured on *Capture_DR* state and shifted out on the *Shift_DR* state [BSH-PARKER-USA, 1998].

2.1.5.4. HIGHZ (Optional)

The *HIGHZ* instruction is an optional instruction and no predefined bit pattern instruction is required. Its purpose is to enhance the ability of In-Circuit *Automatic Test Equipment* systems to test complex boards by reducing the potential for overdrive damage. *HIGHZ* targets the Bypass Register between TDI and TDO to shorten the shift path. It also causes all output and bidirectional pins to go into high-impedance states. It is the designer's responsibility to evaluate if it will be needed or not [BSH-PARKER-USA, 1998].

2.1.5.5. CLAMP (Optional)

The *CLAMP* instruction is also an optional instruction and no predefined bit pattern instruction is required. The *CLAMP* targets the *Bypass Register* between TDI and TDO, to shorten the shift path. It also places all output and bidirectional pins under control of the *Boundary-Scan Register,* which should be previously set up with a *Preload* sequence.

CLAMP is intended for "digital guarding". When testing a board, as it is often necessary to force static "0"s or "l"s on selected nodes in order to set up testable conditions or to block interfering signals [BSH-PARKER-USA, 1998].

2.2. CMOS Image Sensors Architecture for Vision Applications

CMOS technology allows the fabrication of Image Sensors with the same process or with minor changes regarding other CMOS IC devices designed with standard processes. The CMOS standard processes, which were developed for digital and mixed signal applications, are really attractive particularly because of their low power consumption, applicability for on-chip signal processing and large availability [CMOS-Pross-design-4-IS].

The access to small size process, together with the low manufacturing cost gives the possibility to implement pixel electronics to maximize the converted charge. The possibility to have analogue signal amplification and Analogue-to-Digital Conversion on the same chip as the photodiodes, higher noise immunity schemes and some digital signal processing allows camera manufacturers to reduce the overall production cost [AXYS-CCD-vs-CMOS, 2010].

A CMOS image sensor generally consists of an imaging area, comprising an array of pixels, vertical and horizontal access circuitry, and readout circuitry, as shown in *Figure 2-9* [S-CIS-OHTA-USA, 2008]:

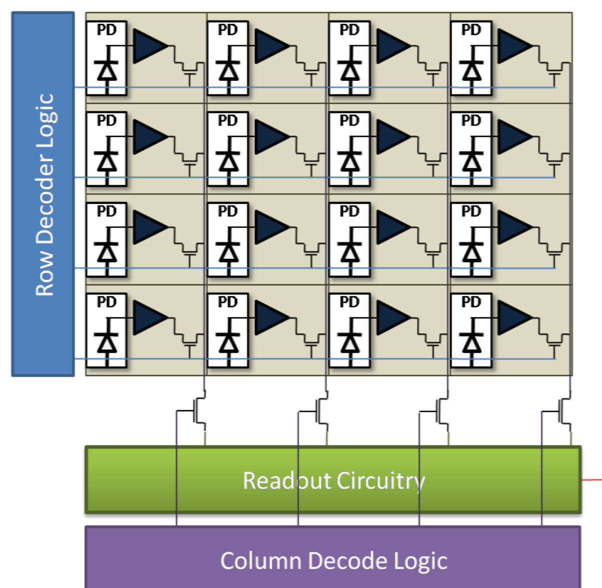


Figure 2-9: CMOS Image Sensor Pixel Organization.

Figure 2-9 shows a two-dimensional array of pixels, vertical and horizontal decoder logics for row and column access and readout circuitry connected to the pixel columns to perform the readout. The readout can be analogue or digital. The pixel consists of a photodiode, a charge conversion circuit and transistors working as switches to select the row and the column. The pixel array is itself the fundamental part of the image sensor.

The resulting image quality is mainly determined by the design quality and process size of this area. *Figure 2-10* shows the basic structure of a pixel:

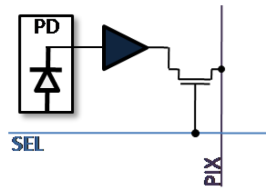


Figure 2-10: CMOS APS pixel basic structure.

The pixel structure is based on the photo-detector, also called photodiode, that is a light sensitive area, represented in the picture as PD, that collects the incident light. The collected light is quantified in number of *photons*[3] and it creates on the silicon semiconductor substrate a charge proportional to the incident light [S-CIS-OHTA-USA, 2008]. This charge is then converted to voltage using a *Charge to Voltage Converter* or CVC circuit, and it generates a voltage proportional to the collected light. This approach is applied on all pixels and each pixel has its own CVC circuitry. This is the base of the Active Pixel Sensor or APS sensor.

Regarding *Figure 2-9*, the readout operation of the pixels is performed by addressing the pixels in the array and selecting each line. The voltage is readout from each pixel through the vertical bit line and one row at a time is transfered to the *Readout* circuitry. Inside the *Readout* circuitry, analogue amplifiers receive the signal from each pixel. Two readout structures can be implemented: the sensor output signal is only analogue or the analogue signal is converted to digital and a digital bus is used as the sensor output. If the output is analogue, the signal from each pixel is amplified, filtered (to reduce the noise) and sent out to an external ADC or analogue processing device (same approach as the one used on CCD sensors). If the readout is digital, it has analogue amplifiers to amplify the signal, a filter block to remove the noise from the pixel signal, an ADC implemented on chip to digitalize the analogue signals and send them out on a digital bus. The digital readout is largely more complex than the analogue one, however it reduces lines delays and attenuations, resulting in a design for faster sensors. The camera complexity reduces considerably once the output of the sensor is already in the digital domain, giving the possibility to implement less or no analogue ICs such as ADC and reference voltage generators around the sensor. The use of pure digital ICs is desirable for the camera manufacturers [DALSA-Im-Sens-Arch].

[3] Photon – according to the theory of light, is a discrete bundle (or quantum) of electromagnetic energy. Photons are always vibrating and, in a vacuum, have a constant speed to all observers: $c = 2{,}998x10^8$ *m/s.*

Figure 2-11 shows the implementation of the readout with the digitalization feature integrated. The most recent CMOS image sensors follow this architecture. The charge captured by the photodiode is converted by the *Charge to Voltage Converter*, or CVC, block that generates a proportional voltage. This voltage is applied to the *Correlated Double Sampling*, or CDS, block when a row and a column are selected. The *Correlated Double Sampling* block is used to reduce the noise by sampling each pixel twice, once in reset condition and again after exposure. The dark signal is subtracted from the exposure signal, eliminating some noise sources. CDS is used widely in electronic imaging [DALSA-Im-Sens-Arch].

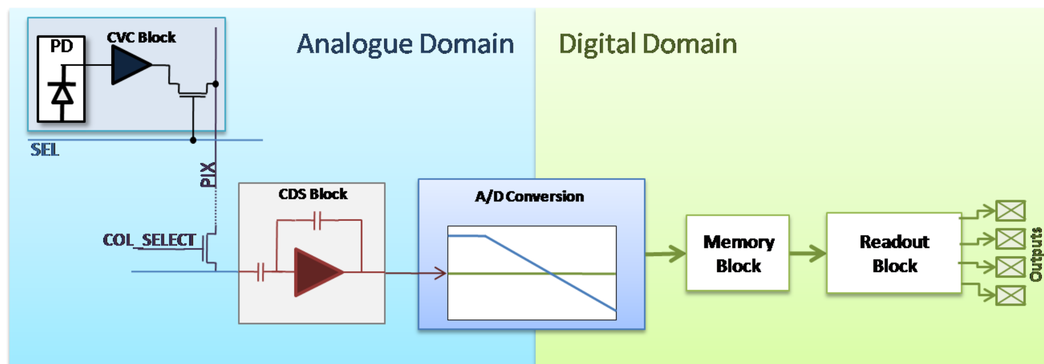


Figure 2-11: CMOS Image Sensor Architecture with digital readout.

This voltage is the applied to the *Correlated Double Sampling* – CDS when the row and the column are selected. The CDS stage reduces noise by sampling each pixel twice, once in dark and again after exposure. The dark signal is subtracted from the exposure signal, eliminating some noise sources. CDS circuitry is used widely in electronic imaging [DALSA-Im-Sens-Arch]. After the CDS stage, the filtered signal is applied to an ADC block that digitalizes the analogue signal delivered by the CDS. The ADC must have the minimum digital word tuned for the corresponding voltage in dark (no illumination) and the maximum digital word tuned for the corresponding voltage in saturation (no more charge can be collected by the photodiode). This permits the ADC to cover the full range of values given by the pixel. The resolution of the ADC will depend on the ADC architecture, sensor speed, and camera manufacturer requirements [008Mega-CIS-SPI-I2C], [AWA-XF768-2010], [KODAK-9618-CIS].

After the ADC, some fast sensors have a memory block that can store the data of a full line of pixels, but it is only required if speed is a concern. If the memory is not implemented, when the current *Analogue to Digital* conversion is finished, no other conversion can start before the read out to the outside of the sensor of all the converted pixels is complete. Also, the readout cannot output the pixel data while the *Analogue to*

Digital conversion is being executed. Thus, the presence of the memory block allows the *Analogue to Digital* conversion to be executed at the same time as the readout block is putting out the previously converted pixel data.

Finally, the readout block, puts in the Sensor pins the converted digital data by the ADC, which represents a specific level of pixel illumination. The readout can be a single block or multiples blocks acting in parallel and can read out the binary words in parallel, where more pins are required, or in serial, where the speed can be the concern.

This type of CMOS image sensors have introduced more logic processing than the first designs (without the digitalisation part), in the same die of the pixel array, reducing the camera complexity and transferring this complexity to inside the CMOS image sensor. This architecture where several features are implemented in the same die is called System-On-Chip architecture. The System-On-Chip architecture is widely in use today as it offers speed, cost reduction and less complexity in camera electronics. However, the added features are complex and depending on the mode of operation of the sensor it responds differently. Thus they have to be controlled internally using a state machine or externally using control signals connected to a programmable micro controller. If is implemented on an internal state machine there is much less control of possible failure, racing conditions or signal tuning, being limited in terms of flexibility, on the number of modes implemented on design and the internal complexity is considerably increased. If controlled externally, the signals must be connected to decoder logic and additional blocks to control the features inside limit the amount of control signals. This means that more blocks with control logic and pins have to be added to the IC.

Figure 2-12 presents a generic block diagram of a CMOS image sensor with some advanced features implemented on it. In this diagram it is possible to identify the Control Block used to control the already introduced blocks. These blocks receive the commands from the Communication block and signals from the outside of the IC, in order to control the states and timings of all other blocks. It is also in this block that it is possible to perform some tuning over other blocks, in order to have them operating in the optimum range. An example of this is the ADC block, where those levels have to fit between the black level (no collected charge on the pixel) and the saturation (maximum collected charge on the pixel).

Another block present in the diagram, is the Communication block. This block is responsible for communications with the outside of the sensor, in order to perform

configurations on the sensor. This block is mainly used to reduce the number of pins to the external world, implementing internal registers and a serial communications protocol. This block will take effect in all other blocks, in order to change some configurations, according to the needs of the user or camera manufacturer. The most common serial protocols implemented are SPI[4], I^2C[5], although there exists some other standard or proprietary protocols [008Mega-CIS-SPI-I2C], [AWA-XF768-2010], [KODAK-9618-CIS].

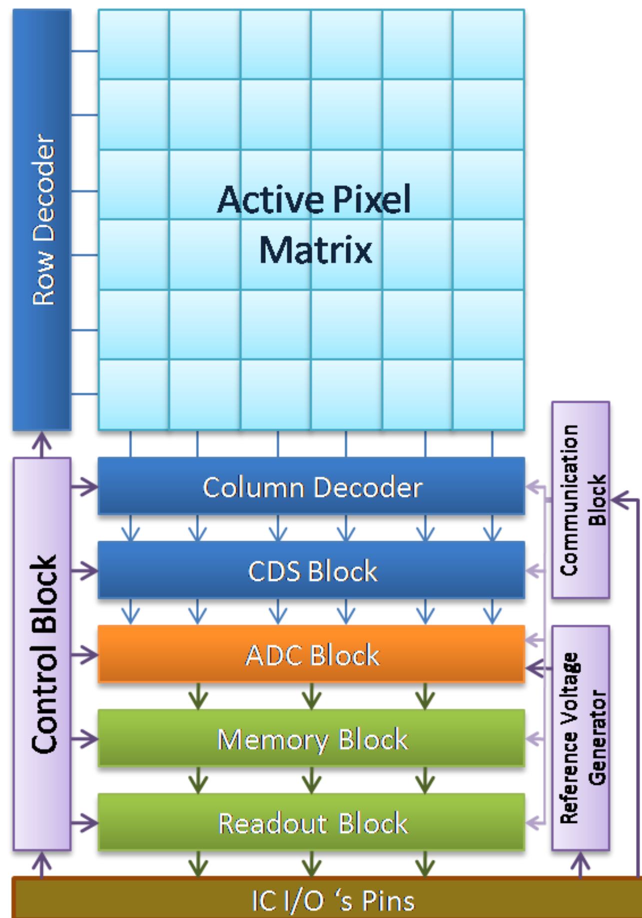


Figure 2-12: CMOS Image sensors generic block diagram.

The Reference Voltage Generator block represented on the diagram, is another important block that can be implemented on the chip or supplied externally via analogue pins. This block generates the proper voltages ranges to feed the ADC and the CDS block in order to have the maximum range available.

[4] SPI – The Serial Peripheral Interface Bus or SPI bus is a synchronous serial data link standard and operates in full duplex mode. Devices communicate in master/slave mode where the master device initiates the data frame. [SPI-I2C-Protocol]

[5] I^2C - Inter-Integrated Circuit bus is a multi-master protocol that uses 2 signal lines. The two I²C signals are called 'serial data' (SDA) and 'serial clock' (SCL). Philips© invented this protocol.

Besides the basic architecture presented in the diagram it is possible to increase the complexity if more speed and extra features are required. If a specific application needs higher speed, higher frame size and higher resolution, then the readout scheme must be redesigned in order to fulfil those requirements. This will increase the complexity and the number of IC pins.

2.2.1. CMOS Image Sensors with higher number of IOs

The level of complexity in the CMOS image sensors is quite high and it is a challenge to designers and manufacturers to include all the features on the same die. Thus, the need to control all the features inside the IC, allowing the sensor to change states or only tune it for a specific application, force the designer to introduce several control pins to the outside. For smart sensors, the amount of pins can be ten's or even hundred's and the connectivity is a concern since the design phase. Thus, the use of integrated connectivity testing processes such as *Boundary-Scan Testing* is desirable for such highly complex image sensors and this allows saving time and money for the companies.

3. Schematic and Simulation Model for the Design

This chapter describes the design process of a *Boundary-Scan Testing* block on an *Image Sensor* prototype. On section *3.1* are described the prototype specifications and characteristics, the power supply internally used on the logic system and on the *Boundary-Scan Testing* block and the design of the test specifications, where are covered the tests functionalities while the sensor is being normally operated. On sections *3.2* and *3.3* are presented the design process, the layer stack and rules, and the EDA CAD software used. Then, section *3.4* describes the *Boundary-Scan Testing* blocks according to the IEEE 1149.1 Standard, and explains the *Test Access Port Controller* and all the designed registers, finalizing with the simulation procedure and simulation results.

The proposed work is intended for a CMOS Image Sensor with a large number of digital inputs and outputs, required for specific and high speed applications.

3.1. Sensor Prototype Architecture

The prototype is a multiple line scan sensor with 512 lines of 4096 pixels each. Every line has active pixels and each pixel is designed with its own *Charge-to-Voltage Converter* block. The array of pixels is designed as a structure in which eight lines can be read at the same time: four lines read to one side of the matrix, named as *"bottom"* by convention and four lines read to the other side of the matrix, named as *"top"*. *Figure 3-1* shows the prototype block diagram where the *top* and the *bottom* sides are represented. The readout electronics are placed in the border of the chip, in the scan direction and in parallel with the array of lines.

On the sides of the array of pixels, also called *pixel matrix*, are placed the column decoders that can address two columns at the same time: one to top side and the other to bottom side. The row decoder has the ability to select up to 8 rows in the matrix of pixels. The selection can be made sequentially or randomly along the array of pixels. The number of selected rows and the side where the rows will be read out are selectable by the user.

The readout is placed after each column decoder. This block is located in two places on the sensor and can process and readout four lines at the same time: so the two blocks can process up to eight lines at the same time. Each readout block has four *Correlated Double Sampling* stages, four 9-bit *Analogue to Digital Converter* blocks, a

memory block (capable of storing the 4 lines) and some processing blocks that allow the user to read the lines unchanged, sampled over the columns and digital summing of the pixel value.

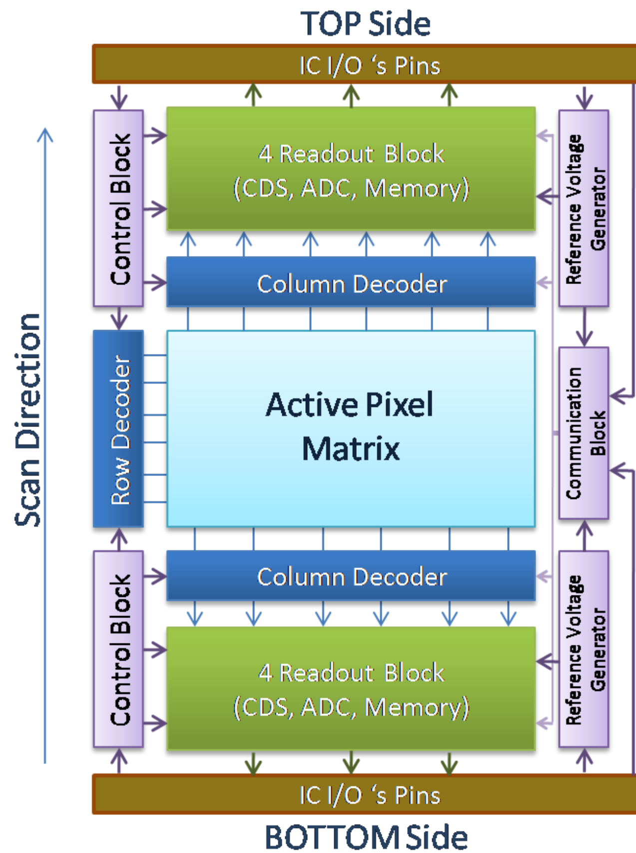


Figure 3-1: Prototype block diagram.

In order to achieve the desired readout speed, 16 Low-Voltage Differential Signalling, or LVDS[6], output TAPs are implemented on each side. Besides these outputs, two additional LVDS outputs are implemented with synchronous clocks. It means that, in total, the prototype has 32 LVDS for data and 4 LVDS for clock.

Due to the high complexity and the needed flexibility, this sensor has a total of 428 pads, where 164 are IO signals either analogue or digital, and 264 power pads. The complete list of pins is shown on *Appendix 1 – Prototype Pin List.*

3.1.1. Power Supply of the Prototype

The sensor has different power sources for the different blocks inside. The power sources are called *VDDD*, *VDDA*, *VDDIO* and *VDDLVDS*. The ground references are *VSSD*, *VSSA*, *VSSIO* and *VSSLVDS,* and are all connected outside the sensor to a

[6] LVDS - Low-voltage differential signalling is a standard published by ANSI/TIA/EIA-644-A in 2001 and uses two copper wires to transmit at high speed.

common reference on the PCB board. This separation keeps the power supply in the expected range, grants noise isolation and different power supply voltages.

The *VDDD* and *VSSD* are the power and ground sources to supply the digital blocks inside the sensor. The voltage difference should be 1,8 V. These power and ground signals are also connected to the digital substrate on all digital blocks. The *VDDA* should have 3,3 V power supply, to the analogue blocks inside the sensor, connecting also to the analogue substrate. In order to provide flexibility on the pad frame, where all the pins to the sensor are located, its dedicated power supply *VDDIO* and *VSSIO* ground are connected to all buffers on the pad and to the implemented ESD[7] protection. This is to provide the sensor with a global protection of the internal core in case of *Electrostatic Discharge*, or ESD, over the sensor. Also, it makes some power adjustment from outside giving the user the possibility to supply the pads with an external voltage different than the *VDDD* voltage. This power can be supplied up to 2,4 V and the typical voltage is 1,8 V.

Finally the *VDDLVDS* and *VSSLVDS* power supply allows keeping the data communication blocks with power and noise isolation from the rest of the sensor. It should be 1,8V and due to the constant current consumption and high speed data transmission, the power separation will grant that power drops or noise injected to the substrate do not reach the analogue and digital blocks inside the sensor.

3.1.2. Design For Test of the Prototype

In order to get the best reliability and the maximum information about the sensor status for operation, some specific test blocks are implemented on chip in order to test the connectivity and signal integrity. The blocks designed on the chip to grant the maximum operation reliability are: the Boundary Scan Chain or JTAG Chain, to test the connectivity of the pad frame to the PCB headboard; the Test Multiplexers, that can address several internal analogue and digital signals to verify and compare with simulations if they are conform with the amplitude and timing, as expected; the ADC Test Mode that allows testing of the ADC block and disconnects it from the CDS stage; the Memory Test Mode that tests the SRAM block; the Training Mode and the Counter Test Mode. The Counter is an ADC sub-block with specific particularities and it must be tested in order to be checked if it is a sequential, monotonous and ascending

[7] ESD – Electrostatic Discharge is the release of static electricity when two objects come into contact. [ESD-PROT-TI]

counting. The Training Mode is the mode used to test if the readout block is giving out the correct data, within the expected timing and without bit errors.

This particular case study will rely on the Boundary Scan Chain. It allows verifying if the connectivity between the sensor and the external world is being performed correctly. For that verification the recommendations of the IEEE 1149.1 Standard are followed, in order to get the Standard capability and it permits placing the sensor in a Standard compliant test system, in order to verify automatically if all signals are correctly connected to the sensor system.

3.2. Prototype Design Rules

The IEEE 1149.1 Standard design rules provide the basic architecture and must be used with the specific CMOS process design rules in order to have the implementation of the Standard for this particular design.

The CMOS process design used is the *XFAB XC018 CMOS design kit* [XFAB-XC018]. This process kit is a standard CMOS process design kit with 180 nanometres of minimum gate width. This process allows the design of *Standard CMOS 1,8V* devices that are the common devices for mixed signal design, RF or *Radio Frequency MOS Design*, which allows the design of high speed and radio frequency transmission MOS blocks, *Isolated MOS Design*, which allows the design of blocks with isolated substrates (useful for substrate noise isolation and voltage separation), and *Higher Voltage MOS Design*, which allows the design of devices for higher voltage supply (up to 3,3V and 5V).

The prototype design uses the *Standard Core Module*, named by the XFAB foundry as *MOSST 1,8V MOS Module*, to design all digital blocks. The analogue part was designed using the *Higher Voltage 3,3V MOS Module* that permits the use of higher voltages and a special stack of layers for the photodiode design. Its design is property of Awaiba, Lda and is beyond the scope of this thesis.

The Boundary Scan Chain design was made exclusively with the *Standard Core MOSST 1,8V MOS Module* and supplied with the VDDIO power supply and VSSIO ground reference. This design uses only the digital library of *Standard 1,8V MOS* provided by XFAB foundry to Awaiba, Lda.

3.3. The CAD Software used

The sensor prototype and the *Boundary Scan Chain* designs were created using the *Electronic Design Automation* software from Mentor Graphics Inc[8]. This software includes the *Design Architect*, also called *DA*, allowing the creation of logic schematics, symbols and an hierarchy of schematics following the process design kit [DA-IC-USR-1995]. It loads the specific process to help the designer to create the project with the correct device dimensions and properties.

The *Design Architect* software prepares the design schematics to perform simulations. Once the design is made, the user generates a SPICE[9] Netlist to start the simulation. The simulator used was the ELDO SPICE [ELDO-SIM-2005]. This simulator is the native simulator from Mentor Graphics, reliable and highly supported by the foundries. This was used to simulate all the schematics created on this prototype as the Boundary-Scan Chain design.

Once the design schematics and simulations were made, the IC layout was produced according to specification. *Mentor Graphics* also has a software in which it is possible to create the layout, using the *IC Station*. This software allows the drawing of different shapes with specific colour and pattern settings. These shapes are drawn one over the others, respecting some process rules, resulting in a specific layer stack. The layer stack must fulfil the specification from the foundry. *Figure 3-2* shows the stack layer for the XFAB XC018 CMOS process.

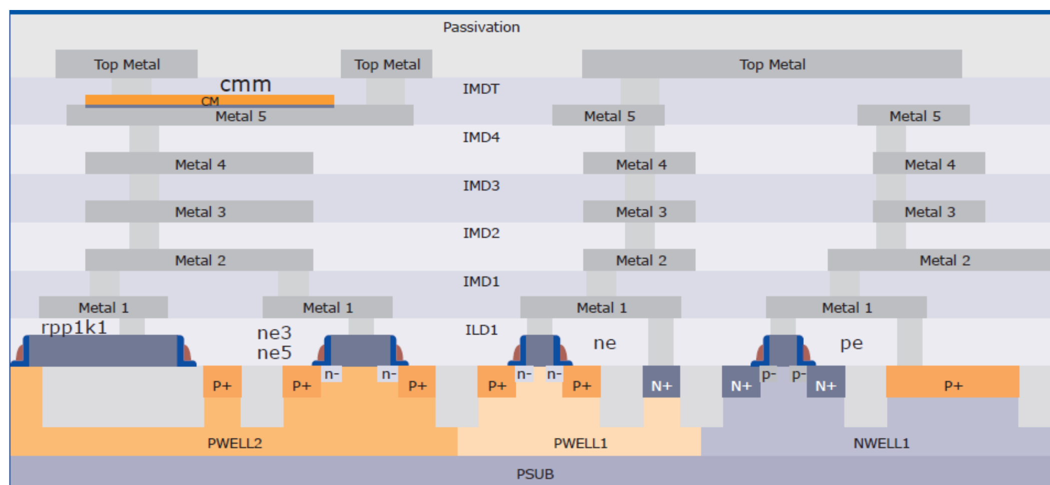


Figure 3-2: XFAB XC018 CMOS process layer stack example [XFAB-XC018].

[8] Mentor Graphics, Inc is a US-based multinational corporation dealing in electronic design automation (EDA) for electrical engineering and electronics.
[9] SPICE – Simulation Program with Integrated Circuit Emphasis is a integrated circuit simulator for integrity check and circuit behaviour.

The *IC Station* is a *Computer-Aided Design* or CAD software tool used to design the layers and devices according to their placement in the silicon and uses the adjusted scale. In order to verify if the foundry process rules are accomplished a design verification called *Design Rule Check* or DRC over the layout should be performed. The *Design Rule Check* is a very useful tool to confirm if the layout fulfils all the design rules.

When the Layout is verified in terms of foundry rules and passes the verification, it should be compared with the schematics used for simulation. For small designs, a visual comparison can be made, by designer responsibility. However, for higher complexity designs, with thousands of devices, the visual comparison is not practicable. Thus the Mentor Graphics tools permits to extract the layout, generating a SPICE netlist that can be compared with the schematics netlists. This process is called *Layout Extraction*. The comparison between the extracted netlist with the schematics netlists is called *Layout Versus Schematic Check* or LVS, and intents to confirm if the layout also meets the device number, sizes and connectivity of the simulated schematics. This guarantees that all the schematics simulated are comparable with the layout produced.

In some particular cases, such as analogue and mixed signal design (taking the example of ADCs), it is very important to verify and simulate more than connectivity and sizes. In these cases, it is very important to guarantee some specific aspects like stability, bandwidth, speed, and power consumption. The fact is that, in some sensitive blocks, the parasitic loads added by the covering routing tracks of metal can be critical for their performance and the designer must consider a *Post-Layout Simulation.* This simulation will take in to account the extracted netlist from the layout with the simulation parameters used to simulate the schematic. It is used to accomplish the sensor specifications.

Figure 3-3 is a flow diagram showing all the stages that the designer must follow and the steps to accomplish in the prototype specification. The *Design phase* is started with the schematic creation, symbols creation and finally simulation. The design phase is similar for one base block or for a top level block, where several base blocks are used. After the simulation, in the *Layout phase* the designed blocks are arranged in terms of process layer stack, the layout is checked in terms of layout design rules, compared with the proper schematic and extracted for a post layout simulation. All these phases should be accomplished by the designer in order to keep the simulation model as close as

possible to the real silicon. In case of failure the design should be reconsidered and re-simulated.

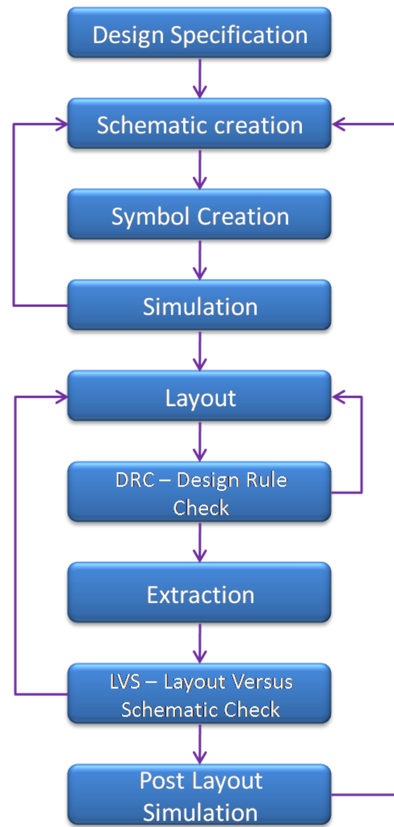


Figure 3-3: IC Design Workflow.

3.4. Boundary Scan-Chain Design

The design of the *Boundary Scan* blocks has to be done according to the description presented on chapter 2, in order to follow the IEEE 1149.1 Standard requirements. Thus, the blocks needed to design the *Boundary-Scan* are the TAP controller (with the *TAP state Machine)*, the *Instruction Register*, the *Bypass Register* and the *Boundary-Scan Register*. This prototype does not include any of the optional registers, such as the *Device ID Register* that requires Company registration in the *Joint Electron Device Engineering Council* group.

The design contains the Test pins required by the Standard. *Table 3-1* lists the pin names, their description and summarizes their functions. These are the only pins used by the *Boundary-Scan* to perform the test. These pins are used exclusively for *Boundary-Scan* test purposes and cannot be shared with the system logic signals. When the sensor is connected in a chain of IEEE 1149.1 Standard compliant devices, the

previous TDO signal is connected to the prototype TDI pin, and the TDO pin is connected to the TDI signal of the next device.

Pin	Description	Function
TDI	Test data input	Serial input pin for instructions as well as test pattern. Data is shifted in at the rising edge of TCK.
TDO	Test data output	Serial data output pin for instructions, test pattern and captured data. Data is shifted out at the falling edge of TCK. The pin is tri-state if data is not being shifted out.
TMS	Test mode select	Input pin that provides the control signal to determine the transitions of the Test Access Port (TAP) controller state machine. Transitions within the state machine occur at the rising edge of TCK. Therefore, TMS must be set up before the rising edge of TCK.
TCK	Test clock input	The clock reference to the boundary-scan circuit.
TRST	Test reset input	Active-low input to reset asynchronously the boundary-scan circuit. This pin should be driven low when not in boundary-scan operation and for non-JTAG users the pin should be permanently tied to ground reference.

Table 3-1: Pin name and description used by the Boundary-Scan circuitry.

3.4.1. TAP Controller State Machine Design

The TAP Controller State Machine block is designed using the principle presented in *Figure 2-2*, which presents the TAP State Machine of the Boundary Scan. Regarding one single state of this state machine, the logic representation of a generic state is represented on *Figure 3-4*.

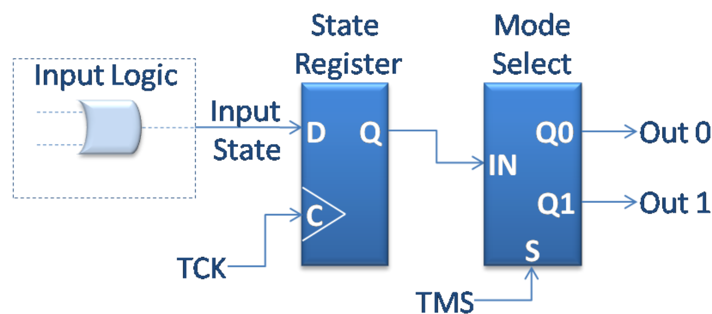


Figure 3-4: TAP State Machine Single Stage Logic.

The state logic has one or more inputs that are represented as *Input Logic*. Then the resulting signal is connected to a *State Register*. The *State Register* is a logic memory block that is sensitive to the rising edge of each TCK clock. At the rising edge of the clock the input signal is stored at the output of the logic cell. The stored logic value is connected to a decoder logic block that depends of TMS signal. If the TMS signal is at

logic '0', then the input of the decoder is connected to *Out0*, otherwise it is connected to *Out1*. These output signals will connect to the next *State-Machine* stages. The Schematic can be consulted on *Appendix 2 - TAP Controller State Machine Schematic,* where the nets are named according to the state it represents. As shown on chapter 2, the available states of the state machine are the *Test_Logic/Reset*, the *Run_Test/Idle* as reset and idle states respectively. From the *Run_Test/Idle* state the *TAP State Machine* moves to the *DR* rank and *IR* rank. The *DR* rank has the states *Select_DR_Scan, Capture_DR, Shift_DR, Exit1_DR, Pause_DR, Exit2_DR* and *Update_DR.* Regarding the *IR* rank, it has the states *Select_IR_Scan, Capture_IR, Shift_IR, Exit1_IR, Pause_IR, Exit2_IR* and *Update_IR.*

If the state machine enters in the *IR* rank it will control the reading process of the next Instruction. Thus the *IR* rank generates signals that will interact with the *Instruction Register.* On the other hand, if the state machine enters in the *DR* rank, then the present instruction will be executed, pointing to the respective Data Register and applying the operations defined by each state. The state transitions are latched in the rising edge of TCK clock.

3.4.1.1. TAP State Machine Simulation

By observing and analysing the schematic of the *TAP State Machine* it is difficult to verify if the *TAP State Machine* behaves as expected. To prove that the State Machine will work when integrated on the prototype is needed to simulate it. Thus, using the device models provided by XFAB, it is possible to create a simulation model close to the real behaviour of the IC. These models consider parasitic resistances, capacitances and inductances, also introducing mathematic models of the behaviour with low signal, high frequency, slew rate, bandwidth, phase and other factors. By this means the simulation can give the designer a level of confidence sufficiently high to validate the model.

The simulator used was the ELDO SPICE simulation tool, which reads the input, the stimulus signals, the *netlist* generated from a specific schematic and the device models for the process in use. The correct power supply must be supplied to the simulation. In the present simulation 1,8V was used. As a result, the output signals can be represented in text or graphical plots. If the text representation is chosen the results are limited to a specific moment in time, while the graphical representation presents the signal evolution with time. *Figure 3-5* has the simulation process flow representation.

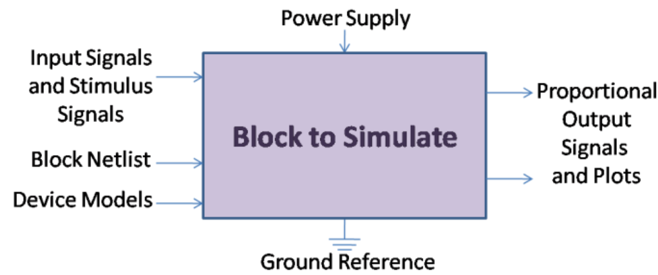


Figure 3-5: Signal flow in simulation.

To simulate the TAP State Machine the ELDO Simulator needs the netlist, shown in *Appendix 3 – TAP State Machine Netlist,* and it needs the simulation parameters with the proper instantiation of device models and stimulus signals, shown in *Appendix 4 – TAP State Machine Simulation Parameters*. The state machine simulation results are presented in a graphical view.

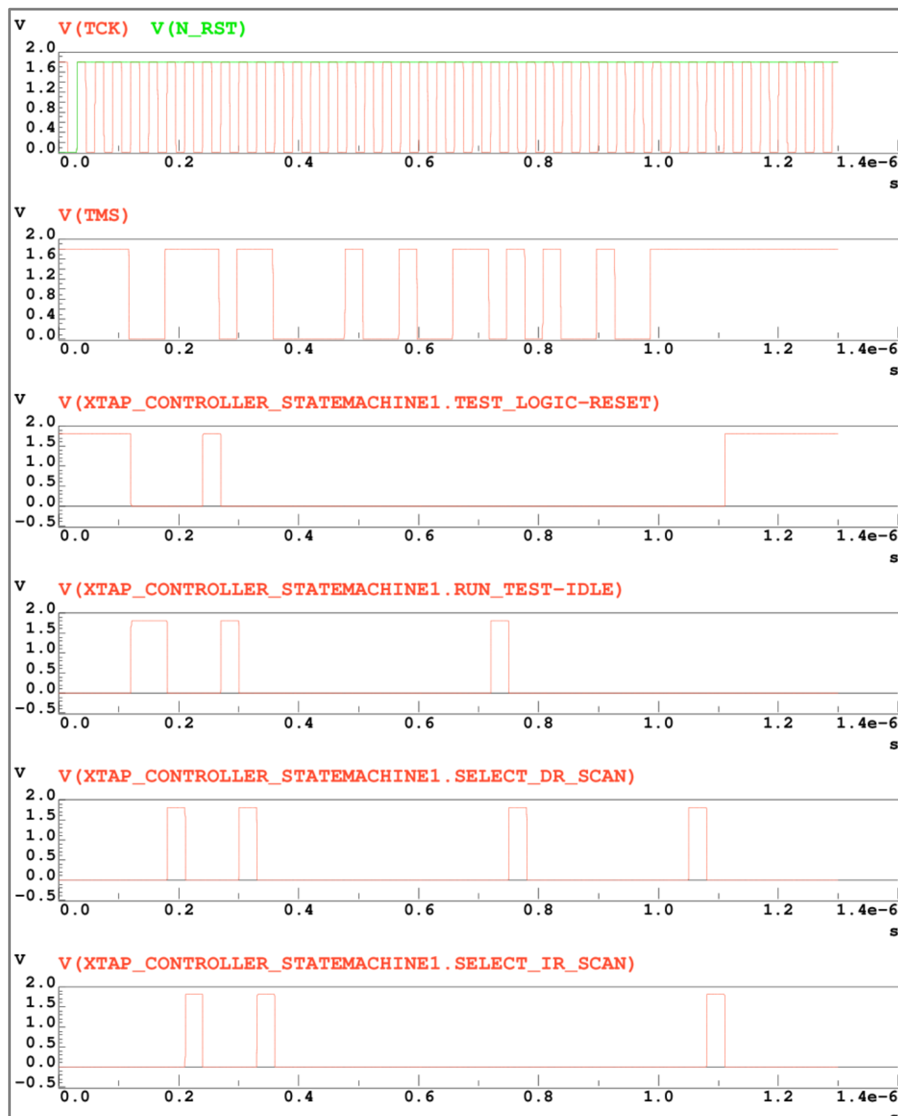


Figure 3-6: TAP State Machine Simulation results plot – 1.

Analysing the first simulation plots, in *Figure 3-6*, once the active low reset signal, *N_RST*, is released the state machine starts in the *Test_Logic/Reset* state as expected. While the *Test_Logic/Reset* signal is at logic high level represents the presence on this state. This assumption is valid for all states.

While the *TMS* signal remains high, the *Test_Logic/Reset* state remains active. The state transitions are made at the rising edge of *TCK* clock. Once *TMS* signal goes low, the state machine leaves the *Test_Logic/Reset* state and enters the *Run_Test/Idle* state. Once in the *Run_Test/Idle* state, the state machine remains in this state while the *TMS* signal is held low. When the TMS signal transits to high, the *Run_Test/Idle* state is left and the state machine enters in the *Select_DR_Scan* state. This is a transitory state used to enter in to the *Data Register* rank or follow to *Select_IR_Scan* state. As the *TMS* signal is held high, in the next rising edge of the *TCK* clock, the state machine moves to the *Select_IR_Scan* state. Likewise the previous state, the *Select_IR_Scan* state is a transitory state used to enter in to the *Instruction Register* rank or move to *Test_Logic/Reset* state. As the *TMS* signal is kept high the state machine moves back to the *Test_Logic/Reset* state.

The above explanation considers the test of the first closed loop states of the state machine defined by the states: *Test_Logic-Reset*, *Run_Test/Idle*, *Select_DR_Scan* and *Select_IR_Scan* (please see *Figure 2-2*). Regarding the second group of plots in *Figure 3-6* and considering the ninth *TCK* clock period, at this moment in time the state machine is again in the *Test_Logic/Reset* state. At the next rising edge of *TCK*, the *TMS* signal transits to low and the state machine moves to the *Run_Test/Idle* state. The *TMS* signal is then raised and the state machine will jump to *Select_DR_Scan* state. Once in this state, the TMS is kept high and the state machine moves to the *Select_IR_Scan* state. This process is described on *Figure 3-7*. To go into the *IR* rank, when the state machine is in the *Select_IR_Scan* state, the TMS signal must go low. At the rising edge of *TCK* clock, the state machine moves to *Capture_IR* state. The *Capture_IR* state is the first of states so select a new instruction to be loaded and executed by the *Boundary Scan Chain*. This state can move on to two different states: if the *TMS* is at '0', it moves to *Shift_IR* state and if the *TMS* is at '1', it moves to *Exit1_IR* state. In the simulation the *TMS* is at '0'and the state machine moves to *Shift_IR* state. The *TMS* signal is kept low for tree clocks, holding the state machine in the *Shift_IR* state. When the TMS signal goes to low, the state machine moves to the *Exit1_IR* state. This state can move

to the *Update_IR* if *TMS* is high or *Pause_IR* if *TMS* is low. As the *TMS* signal is changed to low, the actual state becomes *Pause_IR* at the next rising clock transition.

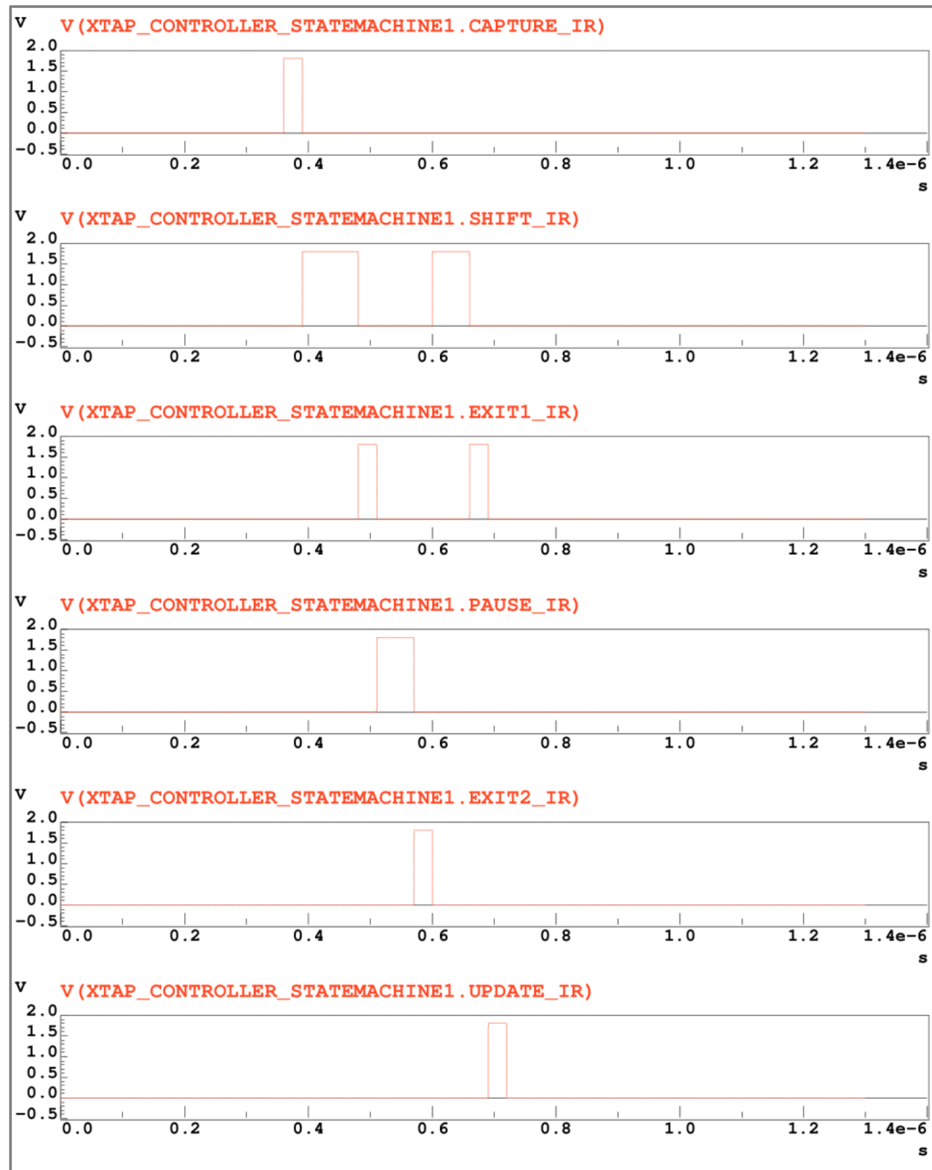


Figure 3-7: TAP State Machine Simulation results plot – 2.

The *Pause_IR* state permits to hold the state machine, thus, if the *TMS* is held low it remains in this state until the change of the *TMS* signal level. In the present clock of the simulation, the TMS is held one clock at low and then goes high. This keeps the state machine in this state one more clock and then moves to *Exit2_IR* state. This state has two exit states: the *Update_IR* state if the *TMS* is at '1' or the *Shift_IR* state if the TMS is at '0'. In this Simulation the *TMS* signal is changed to low and the state machine moves back to the *Shift_IR* state. The *TMS* signal remains low for one more clock and then goes high, leaving the *Shift_IR* state and entering in the *Exit1_IR* state. The *TMS* signal remains high and the state machine moves from the *Exit1_IR* state to the

Update_IR state. Once in the *Update_IR* state, the state machine can move to the *Run_Test/Idle* state, if *TMS* is low, or move to the *Select_DR_Scan* state, if the *TMS* signal is high. In the simulation, the *TMS* is kept high and the state machine goes to the *Select_DR_Scan* state.

As the Instruction related states on the state machine are already verified, then the *Data* related states must be also verified.

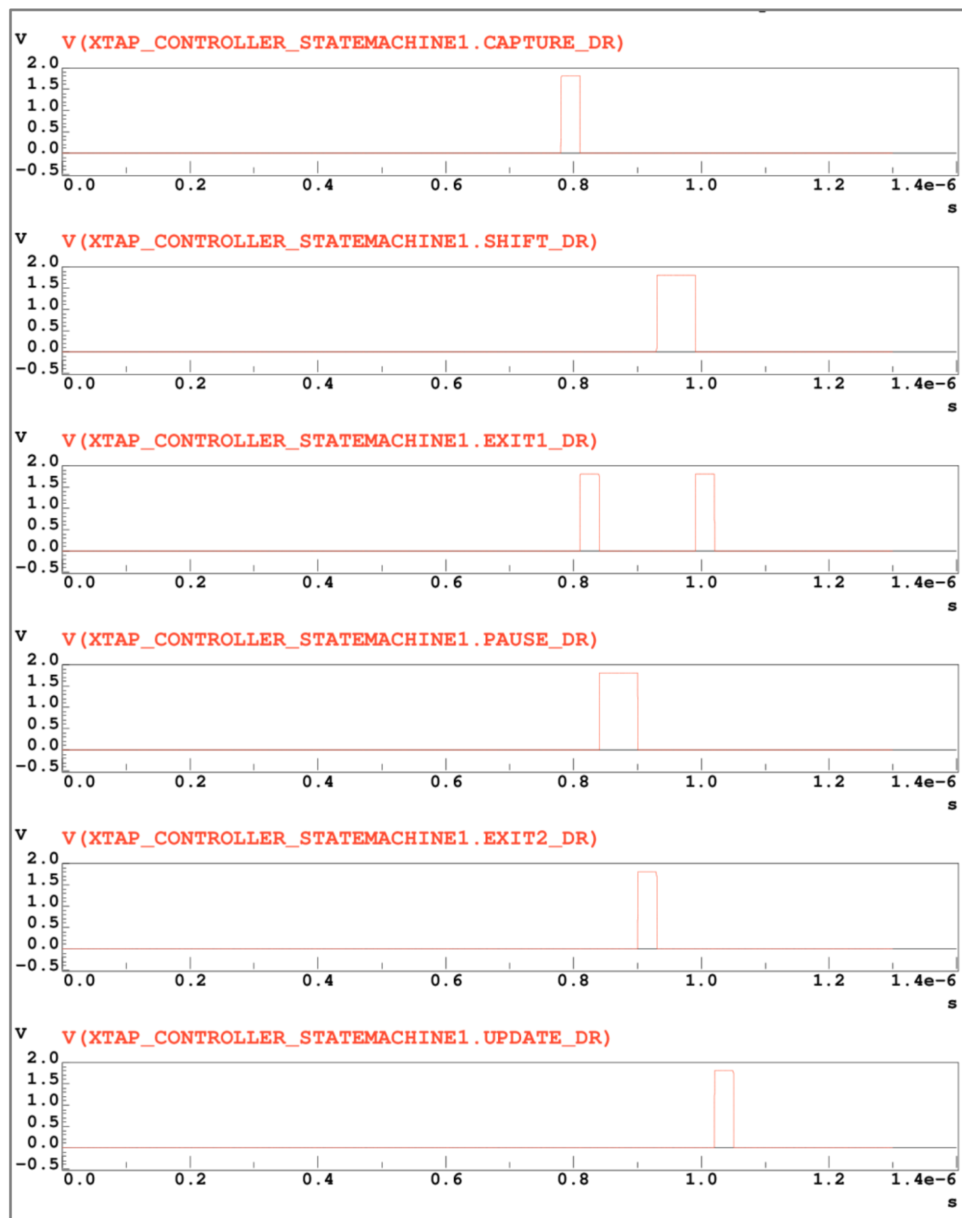


Figure 3-8: TAP State Machine Simulation results plot – 3.

Regarding *Figure 3-8*, starting on the *Select_DR_Scan* state, if the *TMS* goes low at the next rising transition of the clock, the state machine will move towards the *DR* rank, being the first state the *Capture_DR* state. The same conditions found on the *IR* rank can be considered here on the *DR* rank, because those ranks are equal in terms of states

and paths. Thus, if the *TMS is '0'*, and the next state will be the *Exit1_DR* state, then, as *TMS* is kept low, the state machine will move to the *Pause_DR* state. Then the *TMS* is raised and the state machine goes to the *Exit2_DR* state. In this state the *TMS* is changed to '0', moving to the *Shift_DR* state. Due to *TMS* being held low, it remains in this state for one clock, then the *TMS* is raised and moves again to the state *Exit1_DR*. Then the *TMS* goes low, moving the state machine to go to the *Pause_DR* state and remain there for one *TCK* clock period. Finally the *TMS* is raised, remaining high for ten clocks. This will make the state machine move to the following states, at *TCK* clock speed, in this order: *Exit2_DR, Update_DR, Select_DR_Scan, Select_IR_Scan* and *Test_Logic/Reset.* According to the Standard, whatever the state that the *State Machine* is in, if the *TMS* is kept high for five or more clock cycles it should move to the *Test_Logic/Reset* state and remain there. If we observe carefully the state machine has always a five clock cycle path to the *Test_Logic/Reset* state, fulfilling this Standard requirement. This result shows that it is possible to ensure that all states are reachable, controlled by the *TMS* and the *TCK,* and that all the state machine loops are closed to defined states.

3.4.2. TAP Controller Design

The TAP Controller is the main control block. It contains the state machine and the logic to control the *Bypass Register*, the *Instruction Register* and the *Boundary Scan Register*. The state machine is included in this block. Its schematic can be found in *Appendix 2 – TAP Controller Block Schematic.*

The signals used from the state machine are: the *Select_IR_Scan,* the *Capture_IR,* the *Shift_IR,* the *Exit1_IR,* the *Update_IR,* the *Select_DR_Scan,* the *Capture_DR,* the *Shift_DR,* the *Exit1_DR,* the *Update_DR.* The other states are not needed to control the registers implemented.

One of the controlled registers is the Instruction Register. This register will use three main control signals from this block: the *Capture_IR,* the *Shift_IR_CK*, the *Update_IR_CK* and the *Enable_IR.*

- The *Capture_IR* signal used on the Instruction Register is the pulse generated in the *Capture_IR* state, synchronized with the internal *TCK* clock.
- The *Shift_IR_CK* is a falling edge clock-triggered signal with half a clock period and generated when the state machine is in the *Shift_IR* state.

- The *Update_IR_CK* is a falling edge clock-triggered signal with half a clock period and generated when the state machine is in the *Update_IR* state.
- The *Enable_IR* is a signal raised when the state machine enters the *Shift_IR* state and lowered when the state machine enters the *Exit1_IR* state. It aims to enable the shift register inside the *Instruction Register*.

For the Bypass Register, the generated signals are the *Shift_DR_Bypass,* the *Clk_DR_Bypass* and the *Enable_DR_Bypass.*

The *Shift_DR_Bypass* signal is active when the Bypass instruction is active and the state machine enter in the *Shift_DR* state. The Bypass instruction is active when one of the signals, *Bypass* or *Not_Used,* are selected. These signals are generated on the Instruction Register and are explained in *Section 3.4.3.*

The *Clk_DR_Bypass* is the Bypass clock reference. It is the inverted input TCK clock to make effect on the falling edge reading the TCK clock, and is only a clock when the *Bypass Instruction* is selected. If *Bypass* is not selected it is constantly low. The other control signal is the *Enable_DR_Bypass* used to enable the Bypass shift register between the signals TDI and TDO. It is generated when the *Enable_DR* signal is active and when the *Bypass Instruction* is selected. The *Enable_DR* signal is a common signal used on the Data Registers such as the *Bypass Register* and the *Boundary-Scan Register* control logic. It leaves the reset state as '0' and has the rising edge when the state machine enters the *Capture_DR* state and has the falling edge when the state machine enters the *Exit1_DR* state.

The *Boundary-Scan Register* is controlled by the signals *Shift_DR_BSR, Clock_DR_BSR, Enable_DR_BSR, Update_DR_BSR* and *Mode_BSR*. These signals are routed along all cells of the *Boundary Scan Register* and that means, along all the digital pins of the sensor. The *Shift_DR_BSR* signal is high when the *Extest* or *Sample-Preload* instructions are active and when the state machine is in the *Shift_DR* state. The *Extest* or *Sample-Preload* instructions are introduced and explained in *Section 3.4.3.* As reference the *Clock_DR_BSR* signal is generated from the inverted *n_TCK* clock, when the *Extest* or *Sample-Preload* instructions are active and when the state machine enters the states *Capture_DR* and *Shift_DR*. In the remaining states and other instructions this clock is low. Like the *Bypass* control, the *Enable_DR_BSR* signal enables the *Boundary Scan Register* shift register between the signals *TDI* and *TDO*. It is generated when the *Enable_DR* signal is active and when the *Extest* or *Sample-Preload* instructions are selected. The *Update_DR_BSR* signal is generated when the *Update_DR* signal is active

and when the *Extest* or *Sample-Preload* instructions are selected. Finally, the *Mode_BSR* signal is used to select one of the two possible instructions to operate the *Boundary Scan Register*. This signal is '1' when the *Extest* instruction is selected and is '0' when *Sample-Preload* instruction is selected. By default the selected instruction when the reset is released, is the *Sample-Preload* instruction, because it is a non invasive instruction.

Internally some signals are buffered to avoid delays and excess of load where they are generated. This last condition is especially important to the state machine driven signals.

3.4.3. The Instruction Register Design

The *Instruction Register* was designed in accordance to the IEEE 1149.1 Standard and has the minimum size, with only two bits length. With this length it is possible to map up to four instructions.

The instructions implemented are the ones strictly necessary to perform *Boundary-Scan Testing*. Thus, the instruction set has only 3 instructions available. *Table 3-2* shows the implemented instructions and their functions.

Instruction	Code	Function
EXTEST	00	This instruction enables the boundary-scan EXTEST operation.
SAMPLE-PRELOAD	01	This instruction enables the boundary-scan SAMPLE/PRELOAD operation.
NOT_USED	10	This is a not valid code. If enabled, it forces the bypass instruction.
BYPASS	11	Enables the BYPASS operation.

Table 3-2: Instructions implemented in the Boundary-Scan Design.

It is important to notice the presence of an instruction that is not required by the Standard. The instruction *NOT_USED* was implemented to avoid that user typing mistakes or error transmissions would damage the prototype or lead to situations where the Boundary-Scan becomes uncontrollable. Thus, if the instruction code '10' is sent to the prototype, the bypass is forced. The use of the *Bypass* operation is desirable, as it is a non invasive instruction and only has one clock delay from the input *TDI* to the output *TDO*.

The Instruction Register Schematic used in this design is in *Appendix 2 – Instruction Register Block Schematic*.

Regarding the design, the Instruction Register is a Shift Register with two storage points: the *Inst_reg_middle* and the *Inst_reg_end.* During the *Capture_IR* state of the state machine these two points will have the code ‘01’ to be shifted to *TDO*. When the state machine is in the *Shift_IR* state, the instruction code is introduced by the *TDI*, shifted through the Instruction shift register, while the ‘01’ code is shifted out through the *TDO*. In the end of the *Shift_IR* state the lines *Hold_bit1* and *Hold_bit0* contain the new instruction to be decoded, and then these signals are connected to the Instruction decode logic in order to decode the next instruction. The result is used to enable the corresponding register. When the state machine enters the *Update_IR* state, the new instruction becomes effective.

3.4.4. The Bypass Register Design

The Bypass Register is a Data Register that is filled when the instructions *Bypass or Not_Used* are selected. This register is a 1-bit-long data register, that provides a minimum-length serial path between the TDI and TDO pins. It is normally used when the *Printed Circuit Board* has a chain of devices and there is no intension to test the *Boundary-Scan* on this IC. The input data comes from the TDI and it is shifted out through the TDO pins within one clock delay. By default this is the instruction that is loaded when the Boundary-Scan leaves the reset mode. The Bypass Register Schematic used in this design is in *Appendix 2 – Bypass Register Block Schematic.*

3.4.5. The Boundary Scan Register Design

The Boundary Scan Register is mandatory according to the Standard and it is the principal register of the *Boundary-Scan*. This register comprises a group of cells that perform as the interface between the output or inputs of the prototype and the system core. There is a *Boundary-Scan Cell* for each digital input or output. Besides the digital *Inputs/Outputs*, the LVDS outputs have a Boundary Scan cell before the LVDS Driver.

In *Appendix 1 - List of all prototype pins with Boundary Scan indication* there is a list of the sensor pinout, in which the pins associated to the Boundary Scan Cell are indicated with *BSC*. The output signals also have a tristate buffer and a Boundary Scan Cell is also implemented on the tristate enable signal.

The boundary scan cell schematic can be found in the *Appendix 2 – Boundary-Scan Cell Schematic.* This schematic follows the recommendations of the Standard

diagrams and includes the five control signals generated on the TAP Controller for the Boundary-Scan. The signals are *Shift_DR, Clock_DR, Update_DR, Mode* and *n_RST*. The *Shift_DR* signal is connected to an input selector which if '0' captures the *Data_in* signal, and if '1' captures the *Scan_in* signal. When the *BSR* is selected to execute one of the instructions, *Extest* or *Sample-Preload*, the cell starts to capture the *Data_in* signal to the shift register. When the shift process starts, the input selector toggles to the *Scan_in* signal and the data is shifted towards the TDO output. Signals *Data_in* and *Data_out* are the *Boundary-Scan Cell* parallel input and output signals respectively. They are connected to the system core and to the pad and close the system signal path from the system core to the pad or from the pad to the system core. *Scan_in* and *Scan_out* are the *Boundary-Scan Cell* serial input and output signals respectively. The *Scan_in* is connected to the *Scan_out* from the previous cell or to the TDI input. On the other hand, the *Scan_out* is connected to next cell *Scan_in* or to the TDO output buffer. These signals are the basis for the *Boundary-Scan Cell* shift register and its length depends only on the number of pads to be tested using *Boundary-Scan Testing*. The signal *Clock_DR* captures the *Data_in* signal and then shifts it out, through the shift register. This is a clock signal that triggers the shift register on the falling edge transitions. When the test purpose is to shift in a test pattern and then apply it to the system core or to the output pads, there is the need to use the signal *Update_DR*. This signal is used to transfer the digital value previously shifted to the output selector. Depending on the actual instruction it can be transferred or not to the *Data_out*, according to the level of the *Mode* signal. This signal can apply the updated value to the *Data_out* signal or closes the path between the *Data_in* and the *Data_out*. The first condition occurs if its level is '1' and the *Extest* instruction is selected. The second condition occurs if the *Mode* signal is '0' and the *Sample-Preload* instruction is selected at the same time. By default the *Mode* signal is '0' when the reset is released, because it is a non invasive instruction.

Besides the presented *Boundary-Scan Cell,* the design includes another design in which the Update is not possible. Therefore the *Extest* operation is not possible on this cell. Only the *Sample-Preload* operation is possible. This cell is only present in the *Main clock* reference for the system core, since this is one of the most sensitive input signals. The schematic is presented in *Appendix 2 – Boundary-Scan Input Cell Schematic*. This cell only has the control signals *Shift_DR, Clock_DR,* and *n_RST,* and they are the same ones as applied to the regular *Boundary-Scan Cell*.

The complete Schematic of the Boundary Scan Register is in *Appendix 2 – Boundary-Scan Register Block Schematic*. This schematic includes all the cells around the sensor and the pins which are tested according to the *Boundary-scan Testing*. This is the TOP level schematic used as a guideline for prototype layout.

At this point all the implemented blocks were already introduced and explained in terms of their functionality. It remains only to place the blocks together on the top level simulation and start the layout phase. The *Boundary-Scan Control* block schematic can be found in *Appendix 2 – Boundary-Scan Control Logic Schematic*. This schematic includes the symbols of the TAP Controller, the Bypass register, the Instruction Register, additional logic and buffers to the Boundary-Scan input and output signals.

3.4.6. Boundary Scan timing requirements

The *Boundary-Scan* design has to fulfil the timing requirements as requested on the IEEE 1149.1 Standard. For the present design, the maximum TCK clock speed is 33MHz. As shown in *Figure 3-9*, the TDI signal should change level on the clock rising edge transition and the TDO signal should change level on the falling edge transition.

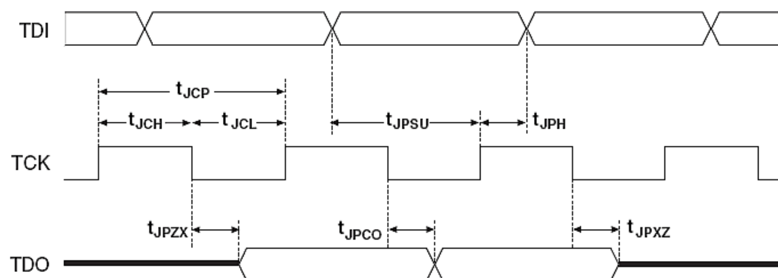


Figure 3-9: Timing requirements for TCK signal, TDI and TDO.

Symbol	Parameter	Min	Max	Units
t_{JCP}	TCK clock period	30		ns
t_{JCH}	TCK clock high time	13		ns
t_{JCL}	TCK clock low time	13		ns
t_{JPSU}	JTAG port setup time	3		ns
t_{JPH}	JTAG port hold time	5		ns
t_{JPCO}	JTAG port clock to output		11	ns
t_{JPZX}	JTAG port high impedance to valid output		14	ns
t_{JPXZ}	JTAG port valid output to high impedance		14	ns
t_{JSSU}	Capture register setup time	4		ns
t_{JSH}	Capture register hold time	5		ns

Table 3-3: Timing Parameters for Boundary-Scan Design.

Table 3-3 shows the transitions timings needed to fulfil the requirements on the *Boundary-Scan* design.

3.4.7. Boundary Scan top simulation

As already mentioned, simulations confirm if a specific design is in accordance to the primary specifications. The main purpose of the top level simulation is to simulate not only each block by itself, but also to evaluate its behaviour with the signals generated in between the connected blocks. With this simulation it is possible to validate the entire concept and analyze possible failures, parasitic states and eventual clock racing conditions.

The top level simulation was made using the schematic present in *Appendix 2 – Boundary-Scan Top Level Simulation Schematic.* This schematic includes the TAP controller block, the Bypass Register, the Instruction Register and six Boundary-Scan cells. The use of a reduced number of cells does validate the top level design and at the same time can reduce computation time for some hours. As the *Boundary-Scan Register* is scalable, i.e., the size of this register is only dependent on the number of cells needed to test each pad, all the top level functionalities are still testable, even for a small Register size. The simulation files can be found in *Appendix 3 – Boundary-Scan Top Level Simulation Netlist* and *Appendix 3 – Boundary-Scan Top Level Simulation File.* These files contain all the information needed by the *ELDO* simulator to perform the simulation. The described simulation on the files takes in to account only the process typical conditions regarding power supply, temperature and transistor models. Nevertheless tests with different power supplies, different temperatures and process variations were performed to ensure total design validation. Initially all signals are active high, (they are active at logic state '1'), with the exception of the *TRST,* that is active low (active when the logic state is '0') and the TDO can be high, low or tri-state.

Figure 3-10 shows the Boundary-Scan main signals. These signals where already introduced and are compiled in this figure to give a perspective of the IC external signals. The *TMS* and *TCK* control the state machine, all operations and instructions. According to these signals the *TMS* signal is evaluated at the rising *TCK* transition and its digital value determines the next state in the TAP State Machine. The *TDI* and *TDO* signals are the input and output signals of the selected shift register. The shift register in use depends on the actual instruction or if an instruction is being loaded. The *TDI* is used to send-in the Instructions and the test pattern to be transferred during the

Update_DR state. The *TDO* signal is tri-state when not in use, and was implemented on the simulation recurring to two 10 kΩ resistors connected between the *TDO* line, the power and the ground signals. By analysing the *TDO* plot, it is possible to confirm the signalling transition '10', when it left the tri-state condition as the new instruction was being loaded. In the case of Data Registers such as the *Bypass* and *Boundary-Scan*, the signalling transition did not occur.

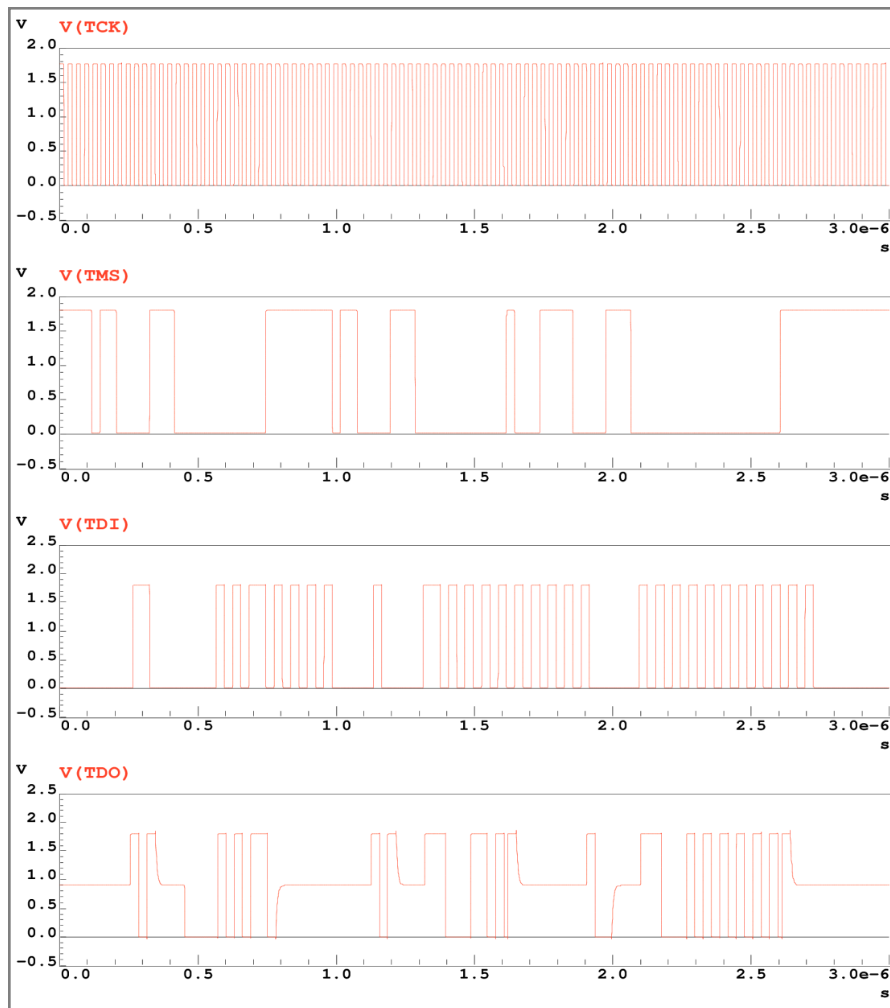


Figure 3-10: Boundary-Scan Top Simulation - Main Signals.

Figure 3-11 shows the top level simulation control signals regarding the *Instruction Register* block. The figure illustrates the signals generated on the TAP controller block that are connected to the *Instruction Register* block. The first signal is the *TCK* used as a reference. Then the *Select_IR_Scan* is related with the state entrance of the state machine and shows if the *Instruction Register* rank is selected or not. When it is selected the TAP controller generates the *Capture_IR* signal, which is generated from the state with the same name. With this signal, the *TDO* leaves the tri-state and sends out the signalling transition '10'. *Figure 3-10* shows this *TDO* transition.

After the *Capture_IR* signal, the *Shift_IR* signal was raised to logical one and enabled the *Instruction Register* shift register. At this moment, the new instruction was shifted in through the TDI. The *Shift_IR* was at logical one for three *TCK* clock cycles, where the first was used to load the signalling transition ‘10’ and the two others were used to load the instruction that has two bits length.

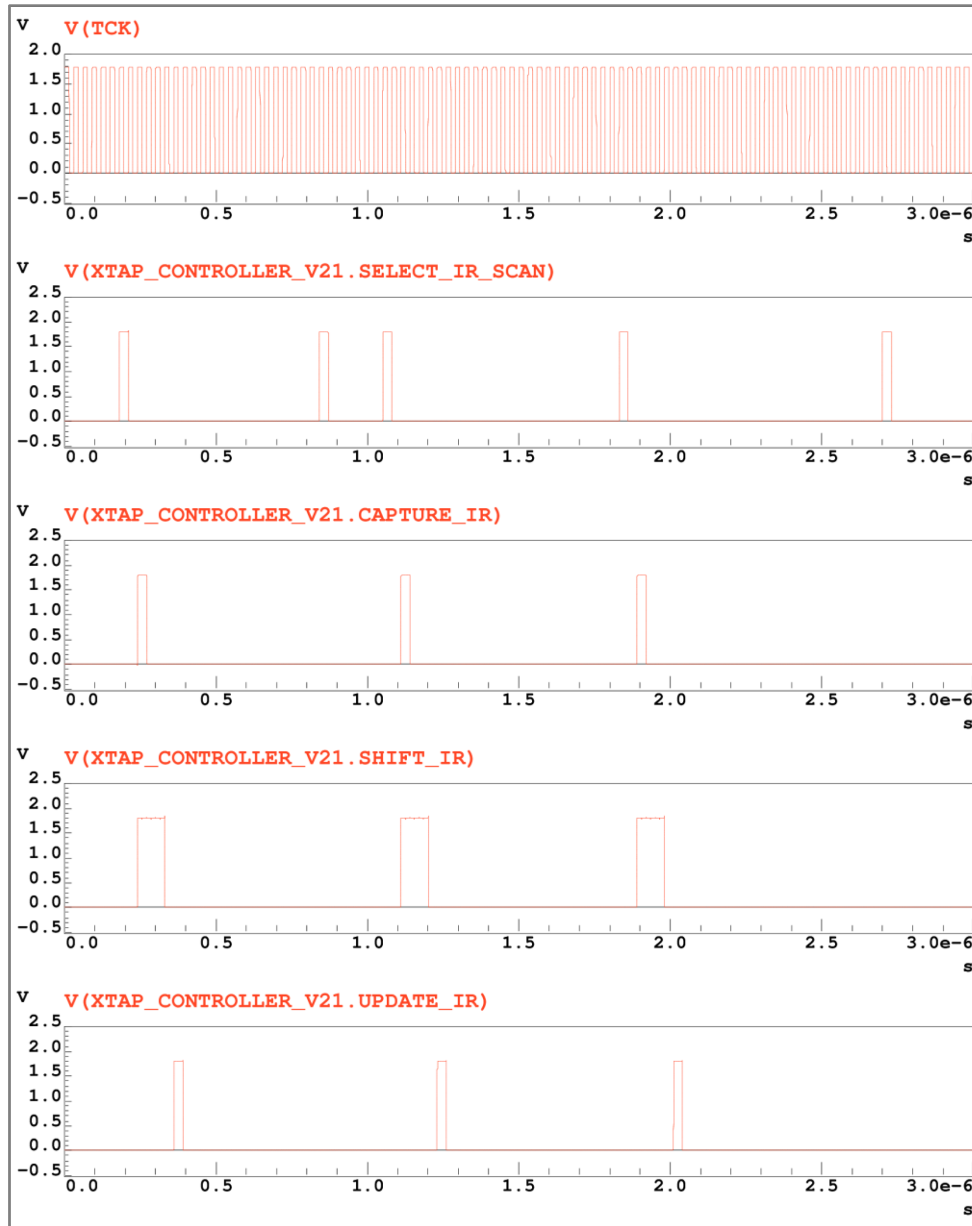


Figure 3-11: Boundary-Scan Top Simulation - Instruction Register Control Signals.

After the third clock cycles the *Shift_IR* falls to logical zero and one clock period later the *Update_IR* signal rises. This signal loads the new instruction and makes it effective. The three available instructions were loaded on this simulation: *Bypass, Sample-Preload* and *Extest.*

Figure 3-12 presents the Data registers related signals. The first signal *TCK,* was once again, used as a reference. The second signal, *Select_DR_Scan*, was related with the state from the TAP State Machine with the same name and decides, together with the *TMS* signal, if the state machine *DR* rank is accessed or not. Considering *Figure 3-11*, the *TMS* signal only allows entrance to the DR rank at three moments:

$T_1 \approx 0{,}4x\ 10^{-6}$, $T_2 \approx 1{,}3\ x\ 10^{-6}$ and $T_3 \approx 2{,}05\ x\ 10^{-6}$ seconds.

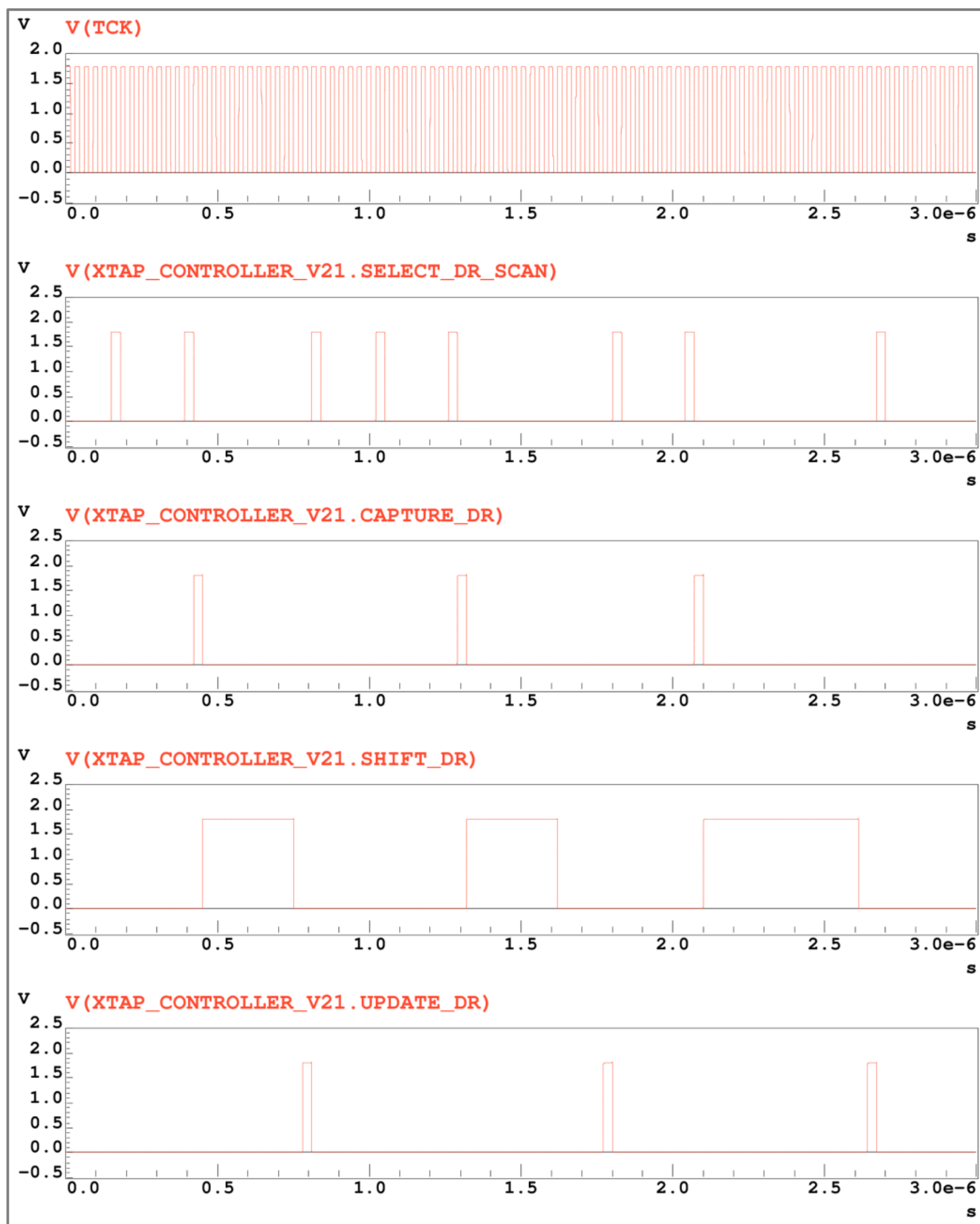


Figure 3-12: Boundary-Scan Top Simulation - Data Registers Control Signals.

Capture_DR is the signal generated in relation to the state with the same name. This signal starts the present instruction execution and closes the shift register path for a specific register: *Bypass* or *Boundary-Scan*. The next operation is to shift-in to the register the user data and to shift out the captured data, during the time that the *Shift_DR*

signal is at high level. This signal must be high for the amount of time needed to transmit all logical bits to the register or from the register, in opposition to the *Instruction Register* that only allows the size of the instruction to be loaded. After the transmission of all bits through the register is finished, the *Shift_DR* state changes to level “0” and one *TCK* cycle later the *Update_DR* signal finishes the instruction execution, updating the data to the register cells if the instruction demands it.

The above explanation is transversal to the two registers implemented: *Bypass* and *Boundary-Scan*, as well as to the instructions that can be executed on those registers: *Bypass, Sample-Preload* and *Extest.* The presented simulation reproduces the execution of all instructions and the basic signal generation and control are transversal to the executed instruction.

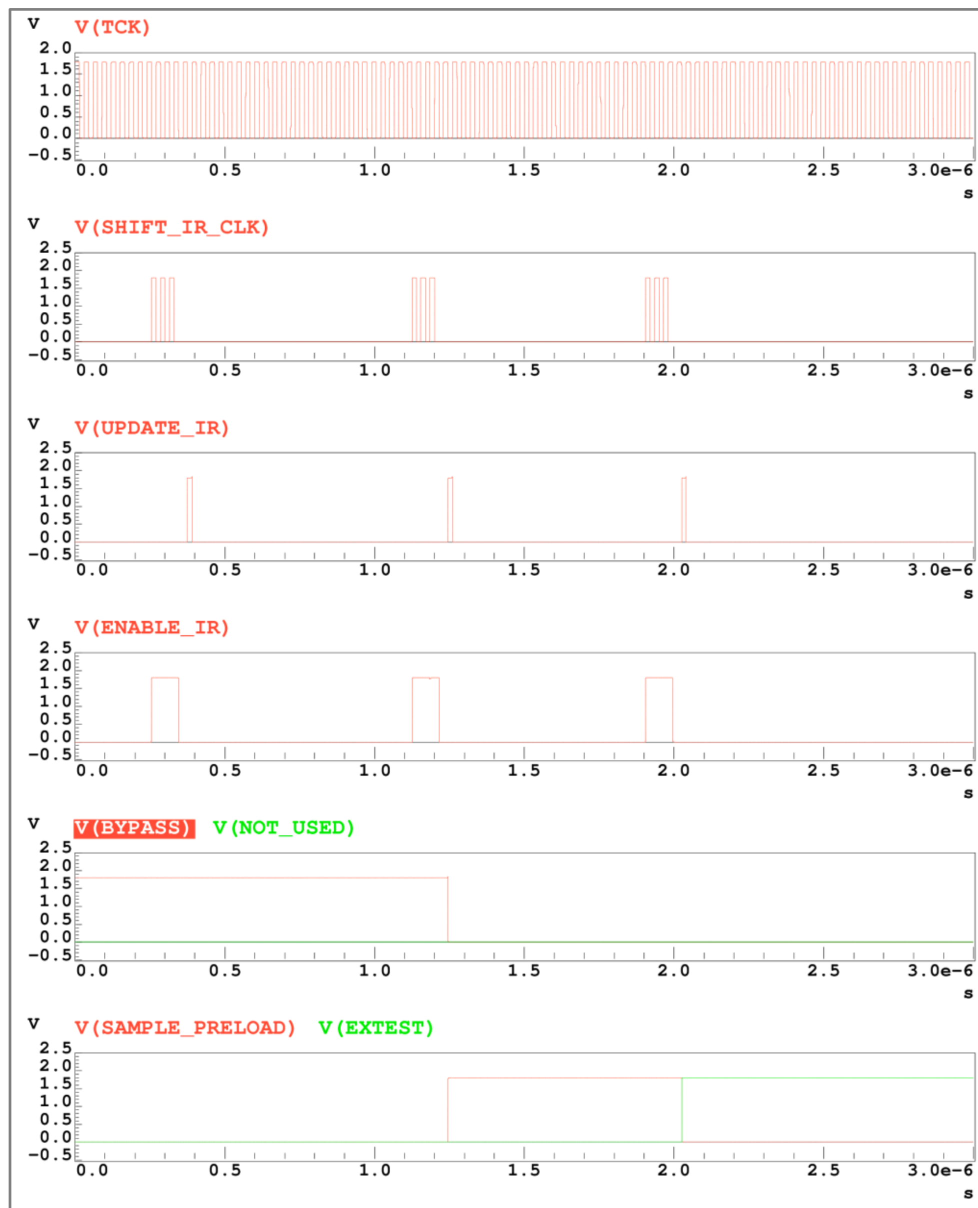


Figure 3-13: Boundary-Scan Top Simulation – Instruction Register Control Signals.

As it can be seen on *Figure 3-13*, the default instruction that the *TAP Controller* executes after leaving the reset state is the *Bypass* instruction. This is an IEEE 1149.1 Standard requirement. Thus, the first instruction to be loaded is precisely the *Bypass Instruction* and the corresponding signal does not change its state once the instruction remains active. The *Not_Used* instruction, implemented for failsafe purposes, was not simulated on the present simulation, also because it is a valid instruction that enables the *Bypass* instruction on the instruction set. The second tested instruction is the *Sample-Preload* that becomes active with the *Update_IR* signal. At the same time the *Bypass* becomes inactive because it is not allowed that two instructions be active at the same time. By the same token, the third instruction to become active is the *Extest*, and at the same time the *Sample-Preload* instruction turns inactive. The implemented instructions were, this way, verified.

While the *Shift_IR* signal is active (logical high), the *TDI* signal from *Figure 3-10*, has the two bit code being introduced to the *Instruction Register.* The *Shift_IR_Clk* is used to load the *TDI* bits to the instruction shift register. It is synchronous with the *TCK* clock and active during the *Shift_IR* state. The *Enable_IR* signal enables the output of the tri-state buffer that connects the instruction shift register to the *TDO* signal.

The plots represented on *Figure 3-13* shows the signals connected to the *Instruction Register* (which were generated on the TAP Controller block) and its output signals.

Analysing the *Bypass Register,* when the *Bypass Instruction* is being executed, the register has the signals according to the plots of *Figure 3-14*. To help to understand this plot, the *TCK* clock is once again presented as a reference. The *TMS* signal is also represented, since its logic state can help to determine whether the TAP controller is using the *Bypass Register* or not. The *Clock_DR_Bypass* is a synchronous clock signal and together with the *TCK* controls the bit shift. It is active if the *Bypass* instruction is active.

The *Enable_DR_Bypass* is only active during the time that the bit shift is being performed. As already explained, it enables the output tri-state buffer, which connects the *Bypass* shift register to the *TDO* signal. Together with this signal, the *Shift_DR_Bypass* performs the bit shift from the TDI to the TDO. The reason why the signals must be active at the same time relies on the fact that the bit shift cannot be done if the *TDI* to *TDO* path is not closed and, in the same way, if the buffer is enabled and no shift is being performed, then the bits coming out from the *TDO* have no relation with the shifted in the *TDI*. The last signal shows the *Bypass* signal related with the

instruction with the same name. All the operations over the *Bypass Register* must be performed while this signal is active high.

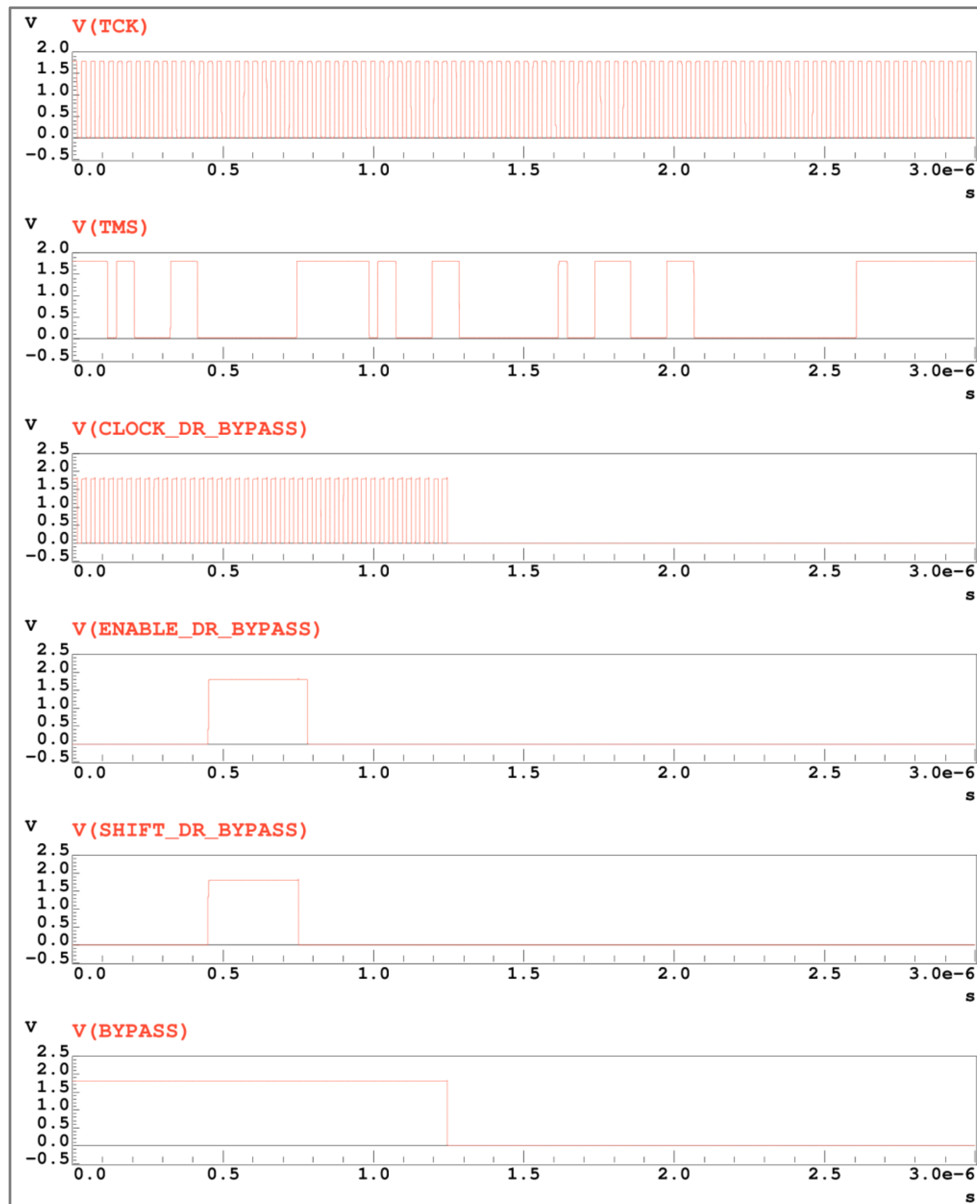


Figure 3-14: Boundary-Scan Top Simulation - Bypass Register Control Signals.

For comparison purposes, once the *Bypass Register* is analysed, it is important to consider the other Data Register: the *Boundary-Scan Register*. The control signal plots are shown in *Figure 3-15*. Likewise the *Bypass Register* description, the signals *TCK* and *TMS* are used as reference. The *Clock_DR_BSR*, the *Enable_DR_BSR*, the *Shift_DR_BSR*, the *Update_DR_BSR* and the *Mode_DR_BSR* are routed around the sensor connecting all the BSC present on the chip. As already mentioned, for simplicity and to reduce computation time, on this simulation those signals are only connected to

six *Boundary-Scan Cells*. The first of those signals is the *Clock_DR_BSR*, used to clock the serial bit shift used to connect the *TDI* to the *TDO*. This clock is operated when the *Boundary-Scan* instructions are being executed and together with the signals *Shift_DR_BSR* and *Enable_DR_BSR*.

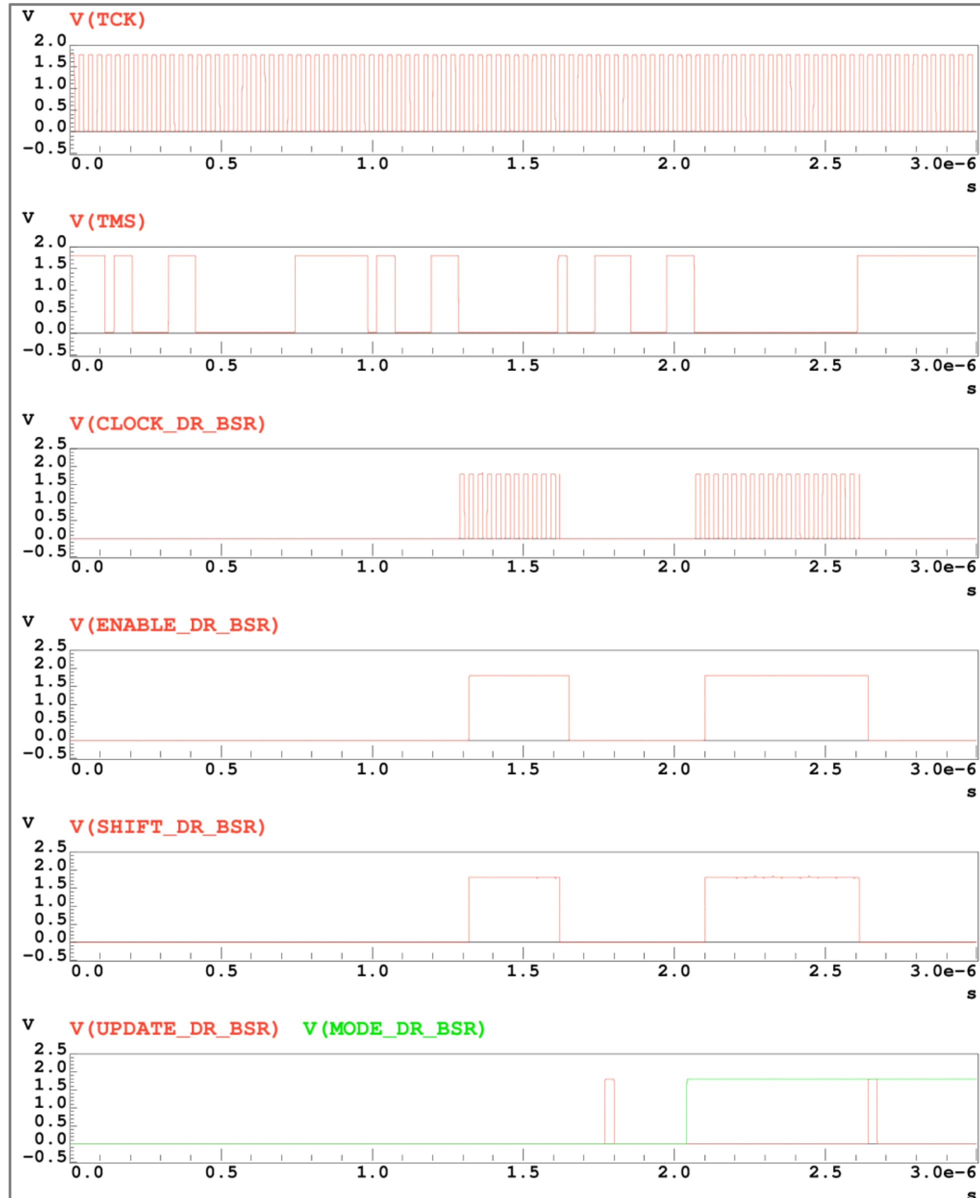


Figure 3-15: Boundary-Scan Top Simulation - Boundary-Scan Register Control Signals.

However it is operated one clock cycle earlier than the signals *Shift_DR_BSR* and *Enable_DR_BSR*, in order to perform the capture of the *Data_in* signal before the shift register starts to be operated. It is possible to identify from the plots the two moments where the *Clock_DR_BSR, Shift_DR_BSR* and *Enable_DR_BSR* signals are active. These moments represent the execution of the two implemented instructions: *Sample-Preload* and *Extest.* What distinguishes these two moments is the signal

Mode_DR_BSR: during the *Sample-Preload* instruction this signal is at logic low and the parallel output of the *Boundary-Scan Cell* is not changed. On the other hand, ifduring the *Extest* instruction the signal is at logic high, then the parallel output of the *Boundary-Scan Cell* is changed during the *Update_DR* state with the shifted test pattern.

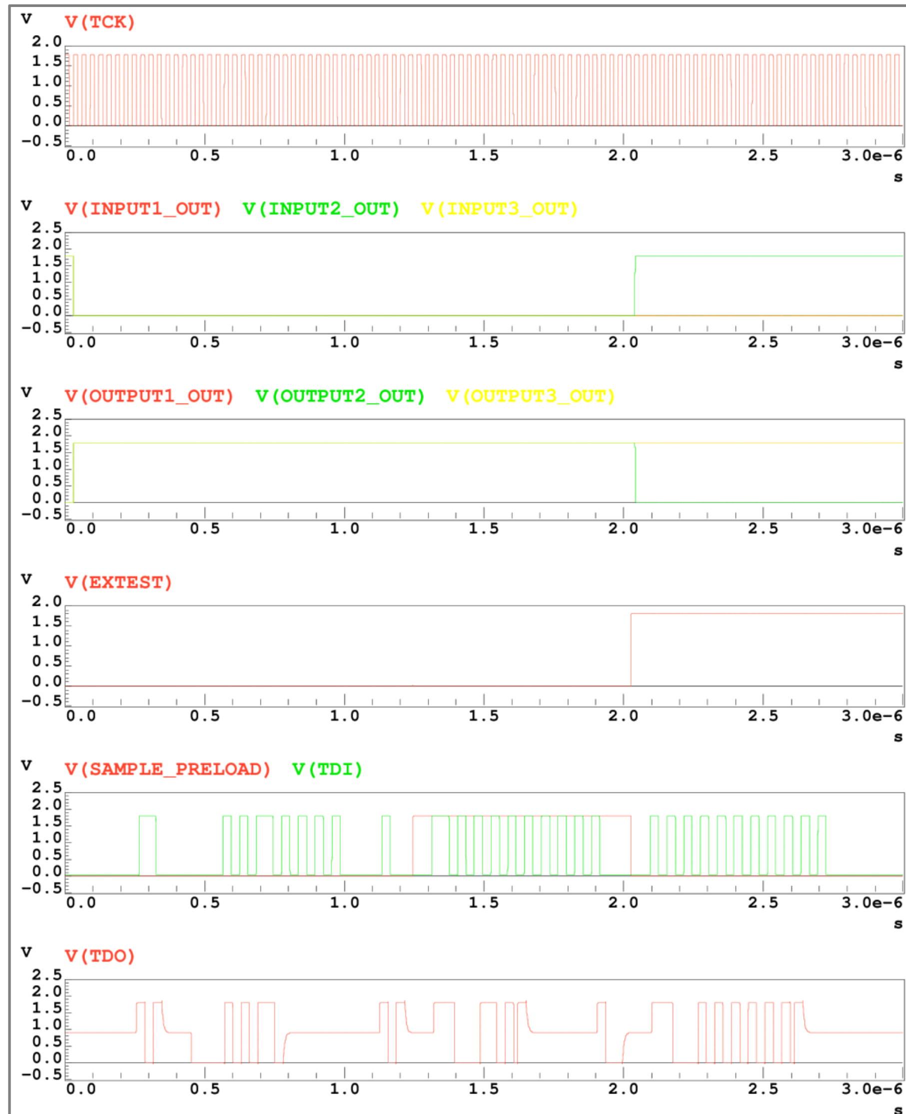


Figure 3-16: Boundary-Scan Top Simulation - Boundary-Scan Register Outputs.

The operations over the *Boundary-Scan Register* are the main purpose of the entire design. The simulation of the *Boundary-Scan Cell* should not be studied by only analysing the internal signals. The plots of *Figure 3-16* show the main signals used in the simulation as a stimulus to observe the *Boundary-Scan Cell* behaviour, these are the plots of the serial signals represented as *TDI* and *TDO*, together with the plots of the parallel outputs from each cell, represented as *Input1_out, Input2_out, Input3_out, Output1_out, Output2_out* and *Output3_out.* The instructions execution is illustrated

with the *Sample-Preload* and *Extest* plots. This figure should be analysed together with *Figure 3-15* for a better understanding of the cell operation mode.

The stimulus signals given to the *Data_in* signal of each *Boundary-Scan Cell* are included on the simulation file in the *Appendix 3 – Boundary-Scan Top Level Simulation File,* named as *Input1_in, Input2_in, Input3_in, Output1_in, Output2_in* and *Output3_in.* According to this file, the *Output1_in, Output2_in* and *Output3_in* stimulus signals are all with value '1' in the moment the instructions execution starts and at the same time the *Input1_in, Input2_in, Input3_in* stimulus signals are all at value '0'. As described before, the instruction *Sample-Preload* captures the parallel *Boundary-Scan Cell* inputs and shifts it out through the *TDO* signal. In addition, it shifts in, from the *TDI,* a preloaded pattern in order to store it on the update register inside the *Boundary-Scan Cell*. This data can be executed immediately after the *Extest* instruction. The *Data_out* from the *Boundary-Scan Cell* is kept unchanged for the entire *Sample-Preload* instruction, making the instruction non invasive, because the parallel output does not change with its execution.

The instruction *Extest*, captures the input value, shift out the captured data and, at the same time, shifts in the test pattern to be loaded on each *Boundary-Scan Cell.* Then the data is updated and the output of the *Boundary-Scan Cell* changes to the value of the test pattern. The number of values introduced on the *Boundary-Scan Register* is dependent of the number of cells implemented. If this restriction is not fulfilled, then the wrong pattern is updated to the system core or to the neighbour IC to be tested. This can either damage the system core of the sensor or fail the *Boundary-Scan Test*.

Observing the *TDO* signal at the moment that the instructions start their execution, and the signal leaves the high impedance state, the first three clock cycles are '1' and the following three clock cycles are '0', corresponding to the *Input1_in, Input2_in, Input3_in, Output1_in, Output2_in* and *Output3_in* signal values captured. As the shift goes on, the *TDO* signal starts to transmit the first bits introduced by the *TDI.*

4. Layout implementation of the Design

This chapter describes the layout process used to design the *Boundary-Scan Chain* after it was simulated. In section 4.1 a small introduction to the software used to produce the layout is made, then on section 4.2 are described the *floorplan* and a diagram that shows where the designed blocks will be placed in terms of physical location. On sections 4.3 to 4.9 are described the layout process, the choice of the metals used and their orientation, and the location of the logic elementary block for each layout. Next, on section 4.10 there is a small description of how DRC and LVS was performed, before the final tape-out or the full prototype layout file was sent for production. Finally, on section 4.11 is described how to setup the simulation after the layout, with the purpose of confirming all the values already obtained on the top level simulation.

4.1. Software used for Layout – IC Station

As mentioned on section 3.3, the Mentor Graphics® EDA design tool has incorporated the IC Station module that permits the design of layers, forming metal paths and semiconductor devices as they are placed on the silicon wafer and at same scale. To fulfil the foundry requirements, some files with the design rules and process characteristic, compatible with the Mentor Graphics® version, were made available by the foundry to be used on the project. One example of a design rule used to design MOSFETs in the *XC018* process from the *XFAB* foundry is precisely what gives the name to the process: the minimum gate length must be 0,18 micrometre or 180 nanometre, thus the number “018” on the process name.

To create the devices that will be compatible with the models used on the simulations, the rules and the process layer stack must be met. The process layer stack is a group of layers that that will represent the doping materials and the metal tracks to be staked on the wafer, in order to generate the desired semiconductor device. *Figure 3-2* shown the layer stack used on the XC018 process [XFAB-XC018].

Figure 4-1 shows a small diagram of a layout performed on the IC Station tool. It is possible to identify some shapes designed with different colours and patterns. The shape with the red colour is the *polysilicon* that is used as the gate for the transistor. The green shape is the diffusion area that defines the transistor boundaries and the gray squares

inside the *diffusion* area are *contact* pillars, to connect the *diffusion* area to the above metal layer. Finally, the solid blue shapes are the first metal layer, called *Metal1*.

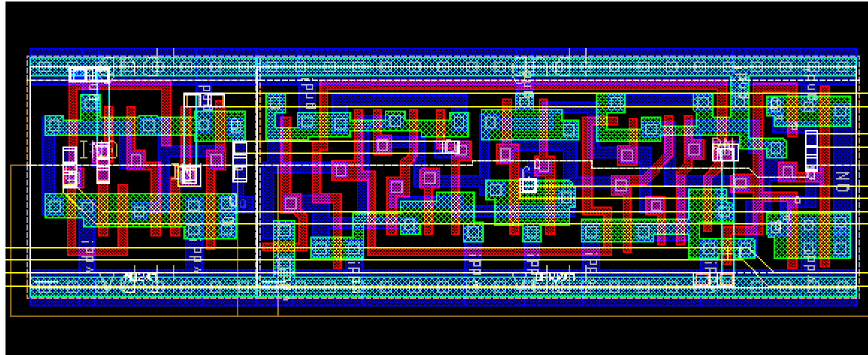


Figure 4-1: Layout example performed on the IC Station tool

Other layers can be identified and must be added, if other devices and structures are needed.

4.2. Boundary-Scan Layout – Floorplan

The floorplan is the first step to be done before the layout process. It is a drawing with the physical position of each block and must respect the requirements and dimensions of the design. In the case of the *Boundary-Scan* it is convenient that all cells be located as near as possible to the pads. Also, the *Boundary-Scan Controller* should be implemented as close as possible to the *TDI, TDO, TMS, TCK, and TRST* pads. Regarding these pads, they should be placed in a position which results in the smaller amount of interference to sensor operation and all pads should be together. The pad organization is not specified on the Standard, being the designers responsibility to organize them in the best way for testing.

The floorplan diagram in *Figure 4-2* shows the locations of the *Boundary-Scan Cells*, the *Boundary-Scan Control Block* position and the direction of the signals. It is important to mention that the number of pads and the number of *Boundary-Scan Cells* represented on *Figure 4-2* is not the same as is presented on the final prototype. Actually this diagram is a guideline for the places where the blocks shall be implemented. Note that, there are some pins that do not have the *Boundary-Scan Cell* represented. These pins are the power pins or analogue pins and they are not present on the *Boundary-Scan Register*.

On the side of the Control Block, in the sensor's right side, regarding the top view, were placed the five *Boundary-Scan* pads in the following order: (from top to bottom) TDO, TDI, TCK, TMS and TRST. The pads are connected to the Control Block.

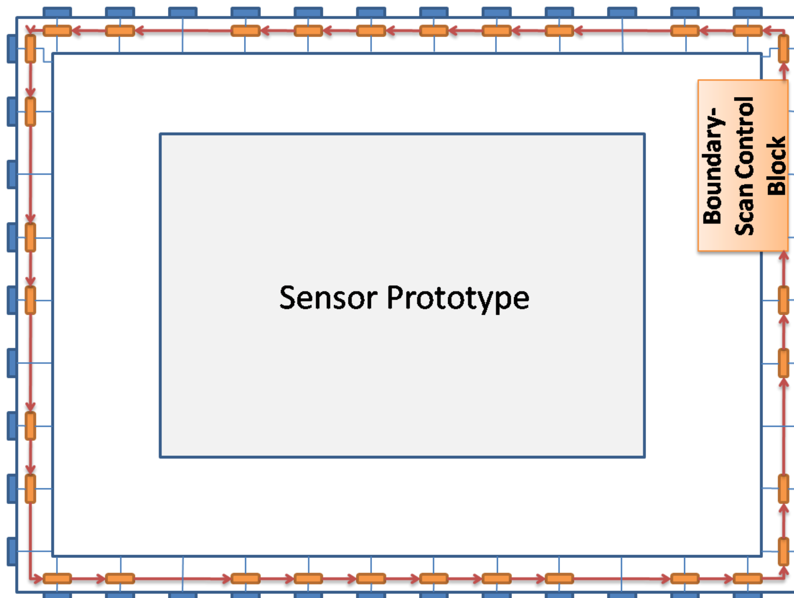


Figure 4-2: Sensor prototype layout floorplan for Boundary-Scan Block.

The layout organization of the *Boundary-Scan Control Block*, represented in *Figure 4-3*, shows the layout organization of all blocks and connections. Once the location of the blocks was defined then the layout phase started.

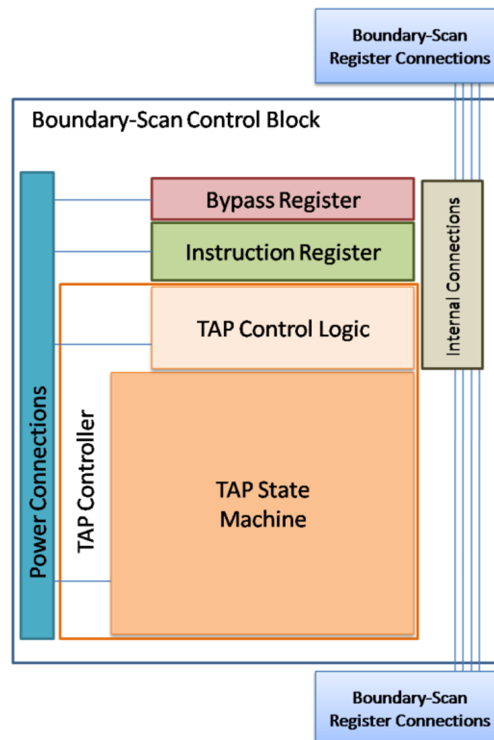


Figure 4-3: Boundary-Scan Control Block layout floorplan.

4.3. Layout of the TAP Controller State Machine

This layout was the first one to be made as it is the block with the highest level of complexity among all blocks. The main concern on the design of this block is the track load and routing that should be minimized, in order to avoid delays, which could constitute problems, or entering in to parasitic states. The stack layer of the XC018 process permits the choice of up to six metal layers, however it was decided to use just three metal layers for sensor design. This is a restriction for the sensor design and not the *Boundary-Scan Block*. The metal organization chosen for this block was the following: the first metal layer was used to interconnect devices and for lower level connectivity inside the elementary blocks; the second layer was used to make horizontal connections; and the third layer was used to make the vertical connections. The layout diagram can be found in *Appendix 4 - TAP Controller State Machine Layout*. Special attention was taken on designing the clock routing in order to minimise clock delays and avoid clock racing conditions. The two ranks, *Data Register* and *Instruction Register* were designed with symmetrical layout to improve the matching of the devices and equalize signal delays. This approach was used to maximize proximity between the state machine stages. The final block dimensions are shown in *Table 4-1*.

Block	X	Y	Units
TAP Controller State Machine	48,30	39,30	μm

Table 4-1: TAP Controller State Machine Layout Dimensions.

4.4. Layout of the TAP Controller Block

This layout was made considering the state machine layout, as the control logic is dependent on their states. The block was designed using the same metal organization chosen for the TAP Controller State Machine layout. The first metal layer is used for device connections and lower level connectivity. The second layer was used to make horizontal connections. The third layer was used to make the vertical connections. The layout diagram can be found in *Appendix 4 - TAP Controller Block Layout* and in *Appendix 4 - TAP Controller Block Layout – Blocks overview*, where the position of the designed blocks is visible. Still the TCK clock routing was considered as sensitive. The final block dimensions are shown in *Table 4-2*.

Block	X	Y	Units
TAP Controller Block	58,90	59,50	µm

Table 4-2: TAP Controller Block Layout Dimensions.

4.5. Layout of the Bypass Register Block

The Bypass block is a simple block with an input selector and a clocked register. In this layout the metal organization introduced on the TAP Controller State Machine layout was respected. The layout diagram can be found in *Appendix 4 – Bypass Register Block Layout*. The final block dimensions are shown in *Table 4-3*.

Block	X	Y	Units
Bypass Register	32,00	5,80	µm

Table 4-3: Bypass Register Block Layout Dimensions.

4.6. Layout of the Instruction Register Block

The Instruction Register block was designed taking in to account all the other designs already created, fitting together with the others in terms of connectivity. This way not only the metal organization, but also the connections fit together with the previous designs. The layout diagram can be found in *Appendix 4 – Instruction Register Block Layout*. The final block dimensions are shown in *Table 4-4*.

Block	X	Y	Units
Instruction Register	48,90	15,50	µm

Table 4-4: Instruction Register Block Layout Dimensions.

4.7. Layout of the Boundary-Scan Cell

The Boundary-Scan Cell is a design that is separated from the Boundary-Scan Controller. It was located aside each digital pad under test. This way the design does not have to be made taking in to account the other Boundary-Scan Blocks, but on the other hand the space it occupies has to be considered. The layout organization is a stretched rectangle and was placed in parallel with the system core, just as drawn in the floorplan diagram in *Figure 4-2*. Also there are two types of cells: the standard one and the input-only. The input-only is a smaller version of the standard one, used on the system main clock. Both layout diagrams can be found in *Appendix 4 – Boundary-Scan Cell Layout*

and *Appendix 4 – Boundary-Scan Input Cell Layout.* The final blocks dimensions are shown in *Table 4-5*.

Block	X	Y	Units
Boundary-Scan Cell	36,60	5,80	µm
Boundary-Scan Input Cell	5,80	18,80	µm

Table 4-5: Boundary-Scan Cell and Input Cell Layout Dimensions.

4.8. Layout of the Boundary-Scan Controller Block

The Boundary-Scan Controller Bock Layout is just the combination of the Bypass Register, the Instruction Register and the TAP Controller Block which were already made. The main concern on this design is to make all connections fit together, keeping all rules and fulfilling the space requirements. The final layout should fit according to the block organization shown on *Figure 4-3*. Regarding the layout diagrams found in *Appendix 4 – Boundary-Scan Controller Block Layout* and *Appendix 4 – Boundary-Scan Controller Block Layout – Blocks Overview,* it is possible to make a comparison, block by block. In the layout it is also possible to observe the buffers used and the TDO tri-state buffers. The final block dimensions are shown in *Table 4-6*.

Block	X	Y	Units
Boundary-Scan Controller	70,34	83,50	µm

Table 4-6: Boundary-Scan Cell and Input Cell Layout Dimensions.

4.9. Boundary-Scan Register Layout dimensions

Once this block was designed on each digital pin the sensor prototype, measurements were made of the Boundary-Scan Register in terms of occupied space. *Table 4-7* shows the total space occupied with routing and logic.

Boundary-Scan Controller location	X	Y	Units
Right side	6,27	18260,05	µm
Left side	6,27	18260,05	µm
Top side	25631,20	6,27	µm
Bottom side	25631,20	6,27	µm

Table 4-7: Boundary-Scan Register Layout Dimensions.

4.10. DRC and LVS checks

Once the layout is completed it is important to verify if the foundry rules are met according to the foundry specifications. Thus, a *Design Rule Check* or *DRC* must be performed over the layout. Once the *DRC* is done the software tool will create a report and a database with the discrepancies found. If all the design rules of the layout are met, the database will be empty and the report will give no errors. To verify the errors over the layout, the tool loads the non empty database and highlights the errors by user commands. In the report, the location of errors are indicated with the coordinates referenced to the cell referential origin. In order to produce the prototype with a similar behaviour to the one observed on the simulations, it is highly recommended that all the foundry rules are accomplished, and therefore no errors detected on the *DRC* check.

The primary checks performed by the *DRC* are: the space between two shapes in the same layer; the minimum and the maximum shape size; the grid alignment and design angle verification, the space in-between layers; the overlap in-between two layers, as well as other checks. The *DRC* strategy relies on the check of the elementary blocks with low complexity first. This way, once the elementary blocks are completed in terms of layout and *DRC* check, the design of the complex blocs was started. This approach helps to design the complex blocks based on simple ones that are in conformity with foundry rules.

Regarding the LVS or Layout-versus-Schematic check, the verification is made by comparing the layout extracted netlist with the corresponding schematic netlist. This check also generates a database and a report that contains all the discrepancies in between the two netlists. The database can be loaded and the errors that make the discrepancy highlighted. Some of the verifications performed are: the connectivity check; the device reconstruction check from a specific stack of layers; devices sizes check; matching check; short circuits check; as well as other checks.

The layout blocks presented before have the same characteristics like the original schematic, thus the *Layout-Versus-Schematic* was performed in all of them. After the *Layout-Versus-Schematic* verification, the post-layout simulation was performed.

4.11. Post-Layout Simulation

After the layout is complete, the top level simulation was conducted in order to compare the layout against the top level simulation.

The simulation setup was the same in order to obtain comparable results, however the layout netlist used was the *Boundary-Scan Controller*. The *Boundary-Scan Register* netlist used for this simulation was the same as top simulation with the six *Boundary-Scan Cells*. *Figure 4-4* shows the *TDI, TDO, TMS and TCK* signals that provided the answer about the layout behaviour. Observing the plots, all three instructions were executed correctly likewise on the top level simulation.

The fact is that, using only a 30MHz clock input and regarding the layout size, the bandwidth available on the designed *Boundary-Scan* is near to what was simulated on the top level simulation. That is, by using such a process as the XFAB, the timing requirements were accomplished without any problem. The resulting layout was then integrated with the prototype layout, a top level DRC and LVS was performed and then the design was sent to production.

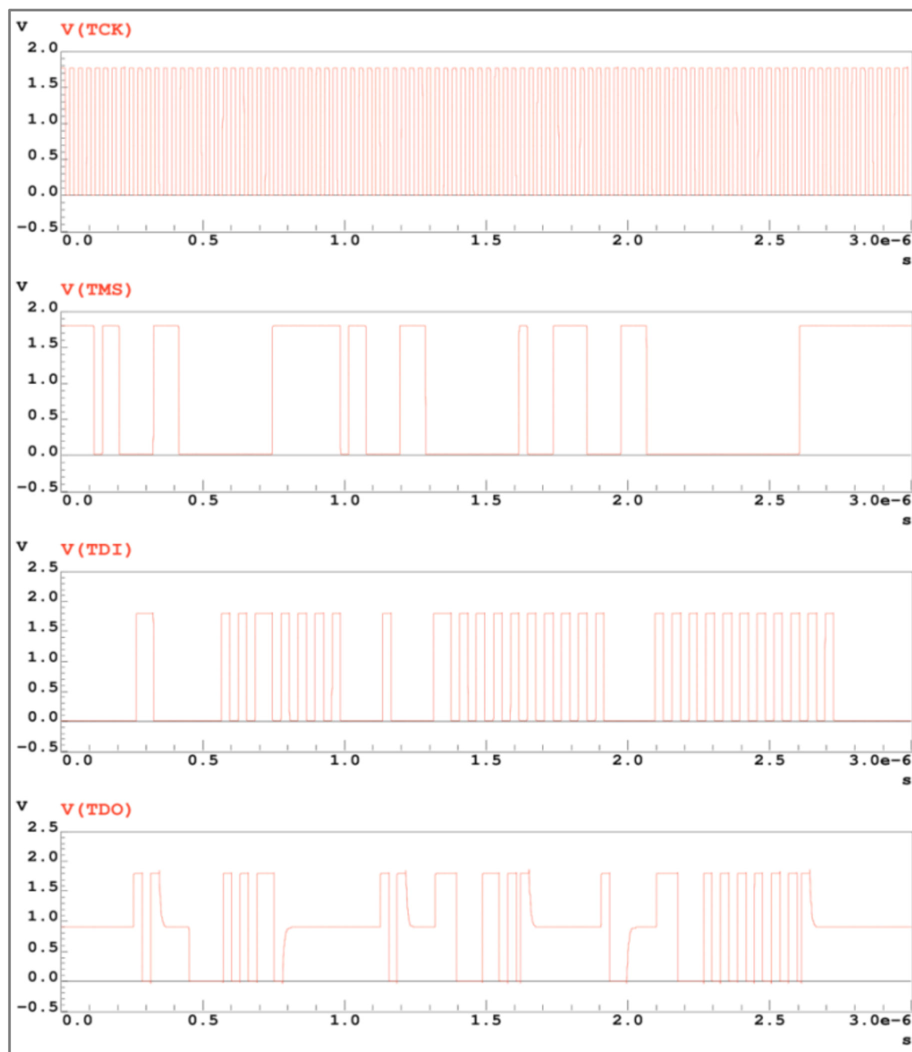


Figure 4-4: Simulation after layout, main IO signals.

5. Prototype verification and functional Boundary-Scan test

The sensor production procedure was made on the XFAB foundry, then it was sliced into individual ICs, picked from the production wafer, placed on a PCB board and connected using bond wires. The final board with the sensor ready to operate was delivered to Awaiba, Lda for test and characterization.

These tests are composed of: first, measure the overall power consumption and for each power supply separately; then the connectivity tests to verify if all pads were connected to the PCB; the next tests were made to verify the sensor's output signals integrity; afterwards the functional tests were performed and finally the sensor optical characterization was performed.

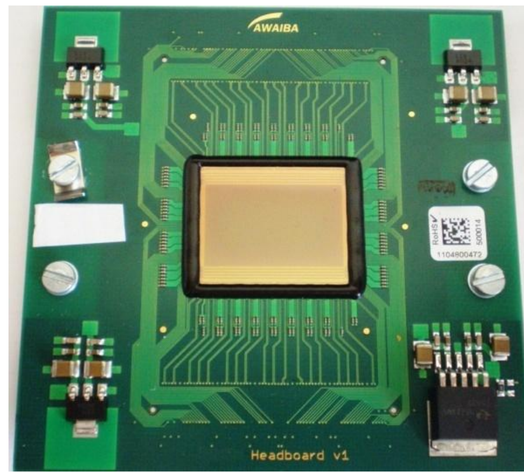


Figure 5-1: PCB board with the sensor prototype.

The Boundary-Scan Testing is part of the third group of tests: the connectivity verification tests. This test is a fundamental one that allows sorting the workable sensors from those that cannot be operated due to connectivity failure.

Figure 5-1 shows the sensor's photography. It is possible to identify the signal routings on PCB board, connected to the main board with FPGAs via plug-in connectors. Thus, the connectivity test is of great importance for this design level.

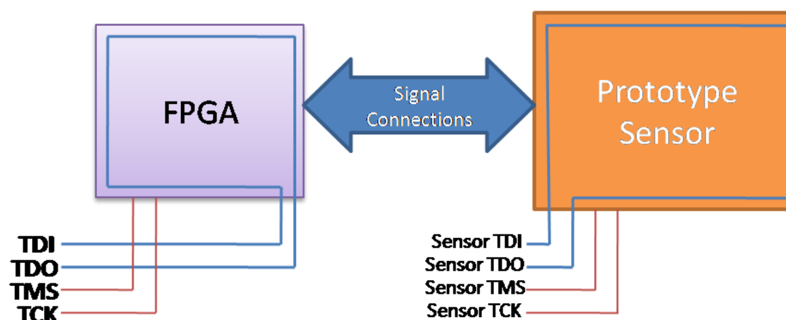


Figure 5-2: Sensor Boundary-Scan signals separated from the FPGA Boundary-Scan signals.

The test board with FPGAs was designed with Boundary Scan routing signals (TDI, TDO, TMS, TCK and TRST) that can be connected to the sensor prototype. The connection is selected on the test board and two configurations are possible for test purposes.

One of the configurations is shown in *Figure 5-2*. This configuration keeps separated the FPGA *Boundary-Scan* signals, also called JTAG signals, from the Sensor's *Boundary-Scan* signals. However, all the signals under test in between the FPGA and the sensor are kept connected to guarantee that all the sensor's IOs have the expected digital value. This configuration permits us to verify the Sensor *Boundary-Scan* design in terms of signals integrity, functionality and IEEE 1149.1Standard compliance. Thus, these signals are tested separated from the FPGA JTAG signals in order to program the FPGA without influencing the sensor and testing the sensor without damaging the FPGA.

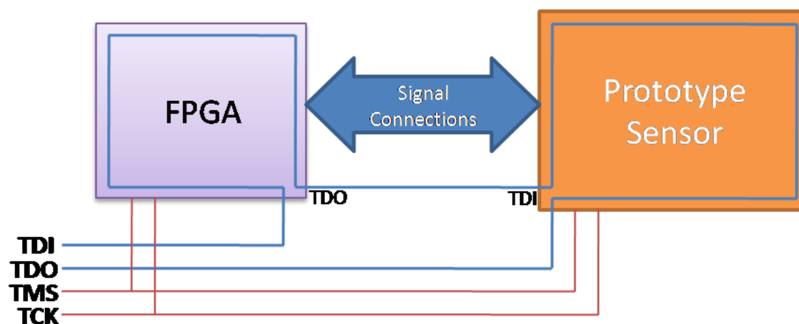


Figure 5-3: Connection of Sensor Boundary-Scan in series with the FPGA Boundary-Scan.

Once the sensor *Boundary-Scan* design is verified as compliant with the Standard, then the second configuration can be made. This configuration is shown in *Figure 5-3*. In this configuration the signals are routed creating a series connection between the *FPGA Boundary-Scan* and the *Sensor's Boundary-Scan* signals. The TDO from the FPGA is connected to the TDI from the sensor. The FPGA TDI is a global TDI, where all data is shifted in to the chain and the sensor's TDO is the global TDO, where all data from the chain is shifted out. To configure this chain, one possible operation in order to write a pattern to the sensor, is changing the FPGA to bypass mode, writing the bypass instruction and executing it. The data length to be shifted-in should be the number of Sensor Register Cells from the current register (if it is the *Bypass Register* then it is only one bit) and one bit for the FPGA *Bypass* register.

5.1. Validation of the Sensor Boundary-Scan Design

In order to verify the design and signals integrity, a boundary-scan test must be performed. So the configuration shown in *Figure 5-2*, was the first to be set-up, where the separated signals to be observed are named with *Sensor TDI, Sensor TDO, Sensor TCK* and *Sensor TMS*. The signals connected to the FPGA were used to program it. This program (firmware) is also defined to have specific output signals to test the *Extest* and *Sample-Preload* instructions on the sensor.

To generate the signals a pattern generator can be used or a FPGA with a state machine can be implemented. However to guarantee the IEEE 1149.1 Standard compliance the software XILINX© ISE, version 10.1.03, was used with a web pack license. Together with this software, a XILINX Platform Cable USB II programmer was used. This platform is used to load the firmware and program the XILINX FPGAs and Flash devices attached to it, using the IEEE 1149.1 Standard or JTAG [ISE_10-1-03_USER], [JTAG-PROG].

Once the ISE suite is installed, it is possible to open the iMPACT© module, to communicate with the JTAG device using the Platform Cable. The firmware loading and programming of the Xilinx FPGA and Flash devices are made using this module, however it also has an operating mode called *Boundary-Scan Chain Debug* mode. It allows us to define the logic state of each of the TDI, TDO and TMS signals, at every rising edge event of the TCK clock. The operation is executed by filling the fields and playing with the command buttons available on the software module. *Figure 5-4* shows the main interface of the iMPACT module with the *Boundary-Scan Chain Debug* mode enabled.

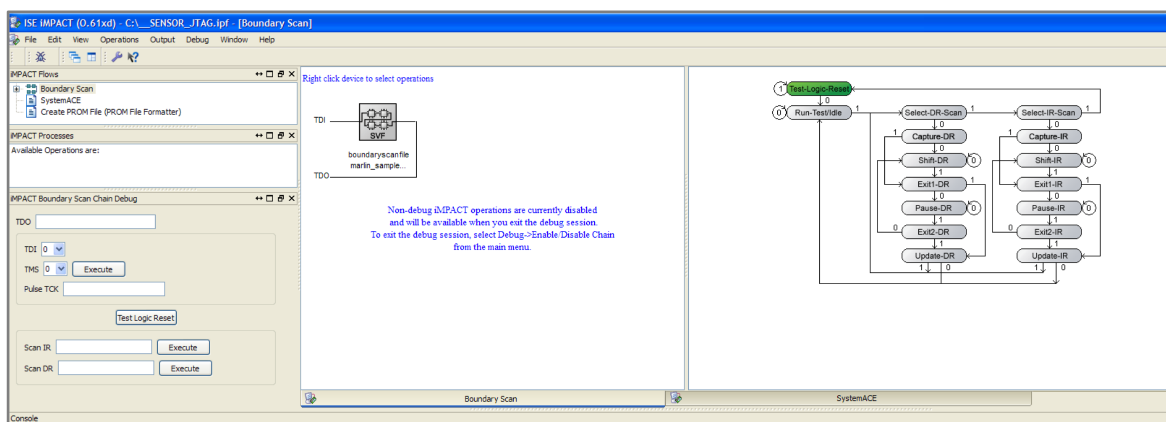


Figure 5-4: iMPACT software module with Boundary-Scan Chain Debug mode enable.

On the left end side of the iMPACT interface it is possible to observe the state machine diagram, according to the IEEE 1149.1 Standard. This diagram is used to enter the next logic value on the TMS signal.

The flowchart of *Figure 5-5*, describes the set of operations to be made over the sensor *Boundary-Scan* design, operating the JTAG signals *Sensor TDI, Sensor TDO, Sensor TCK* and *Sensor TMS*.

According to *Figure 5-5*, the operations flow start by releasing the reset signal, then load the instruction, make the instruction active with the update and then execute the instruction. This process sends to the TDO signal the resultant words, depending on the instruction implemented. Relied on the TDO output, it is possible to confirm if the instruction is correctly implemented or not. The state machine test can be confirmed with the load and execution of the three implemented instructions.

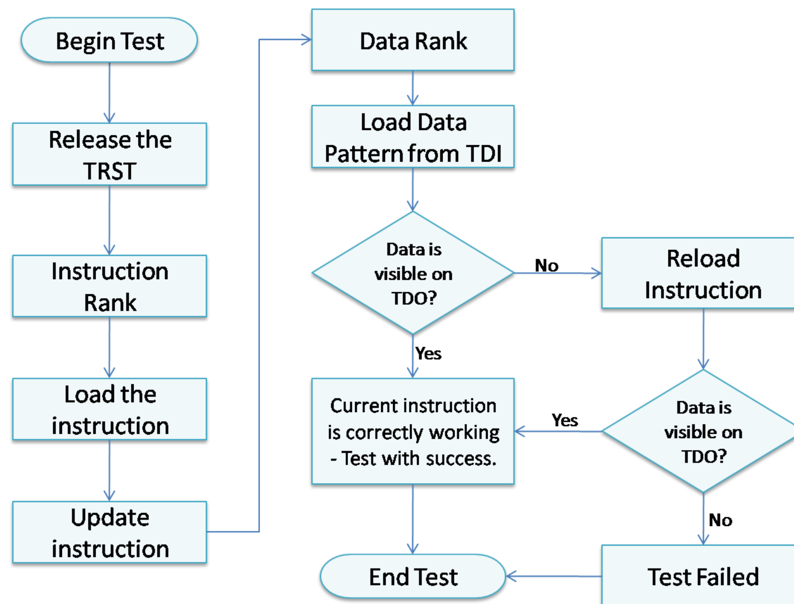


Figure 5-5: Test Boundary-Scan Design flowchart.

The first of the instructions to be tested should be the bypass, because it is also the default one. In case of fail on the instruction load, analysing the TDO results, it is possible to load the instruction one more time. If the second attempt fails, then the test on the *Boundary-Scan Chain* on sensor prototype fails.

To perform the test according to *Figure 5-5*, it was needed to operate the *Impact* software and observe the console log for results. The Xilinx Platform Cable is configured to operate at 6 MHz, a clock frequency that can be operated on the sensor's *Boundary-Scan*, according to the design specification. The Console field shows the results in each TCK cycle. The observed command responses are listed below:

```
// *** BATCH CMD : bsdebug -tms 1 -tdi 0 -tck 01          (1)
TDO Capture Data: 1
// *** BATCH CMD : bsdebug -tms 0 -tdi 0 -tck 01          (2)
TDO Capture Data: 1
// *** BATCH CMD : bsdebug -tms 1 -tdi 0 -tck 01          (3)
TDO Capture Data: 1
// *** BATCH CMD : bsdebug -tms 1 -tdi 0 -tck 01          (4)
TDO Capture Data: 1
// *** BATCH CMD : bsdebug -tms 0 -tdi 0 -tck 01          (5)
TDO Capture Data: 1
// *** BATCH CMD : bsdebug -tms 0 -tdi 0 -tck 01          (6)
TDO Capture Data: 1
// *** BATCH CMD : bsdebug -tms 0 -tdi 1 -tck 01          (7)
TDO Capture Data: 1
// *** BATCH CMD : bsdebug -tms 0 -tdi 1 -tck 01          (8)
TDO Capture Data: 0
// *** BATCH CMD : bsdebug -tms 1 -tdi 0 -tck 01          (9)
TDO Capture Data: 1
// *** BATCH CMD : bsdebug -tms 1 -tdi 0 -tck 01          (10)
TDO Capture Data: 1
// *** BATCH CMD : bsdebug -tms 1 -tdi 0 -tck 01          (11)
TDO Capture Data: 1
// *** BATCH CMD : bsdebug -tms 1 -tdi 0 -tck 01          (12)
TDO Capture Data: 1
// *** BATCH CMD : bsdebug -tms 1 -tdi 0 -tck 01          (13)
TDO Capture Data: 1
```

This is a sequence of command responses, representing the load of a new instruction. In this particular case the instruction loaded is the *Bypass* instruction, using the code '11', defined by the Standard and during the *Boundary-Scan* design. The command responses have two lines for iteration. The first line starts with the comment identifier "//" and has the logic values of the inputs TMS, TDI and TCK. The second line has the response of the TDO signal related with the present state and the input signals. Before starting the test, the signal TRST was released, moving the state machine to the first state: the *Test_Logic/Reset* state. Next the command responses are described:

(1) This response represents the *Test_Logic/Reset* control state and the TMS signal is kept at '1'. Maintaining the state machine in this state, the TDO is at high impedance, however the TDO pad has a pull-up circuit bringing the signal to '1' when it is not operated.

(2) The second command has the TMS signal at low and then the state machine has moved to the *Run_Test/Idle* state.

(3) This command represents the state transition from the *Run_Test/Idle* state to the *Select-DR-Scan* state.

(4) Once the TMS is kept at '1' on this command, the state machine has moved to the *Select-IR-Scan* state, the first state of the *Instruction* Rank.

(5) TMS signal is changed to low and the state machine has moved to the *Capture-IR* state.

(6) This command lead the state machine to enter in to the *Shift-IR* state.

(7) and (8) These two commands shift-in the instruction code. The TDI signal was changed to '1' (on each command), representing the Bypass instruction ('11'). The TDO also sends out the word '10', signalizing the instruction being shifted.

(9) – (13) Finally the TMS was changed to high during these commands, moving the state machine back to the *Test_Logic/Reset* state.

With this sequence of commands it was possible to verify the loading and the enabling of the Bypass instruction. However only with this it is not possible to confirm all the results. It is necessary to execute the instruction. Next is shown how the Bypass instruction was executed:

```
// *** BATCH CMD : bsdebug -tms 1 -tdi 0 -tck 01          (14)
TDO Capture Data: 1
// *** BATCH CMD : bsdebug -tms 0 -tdi 0 -tck 01          (15)
TDO Capture Data: 1
// *** BATCH CMD : bsdebug -tms 1 -tdi 0 -tck 01          (16)
TDO Capture Data: 1
// *** BATCH CMD : bsdebug -tms 0 -tdi 0 -tck 01          (17)
TDO Capture Data: 1
// *** BATCH CMD : bsdebug -tms 0 -tdi 0 -tck 01          (18)
TDO Capture Data: 1
// *** BATCH CMD : bsdebug -tms 0 -tdi 0 -tck 01          (19)
TDO Capture Data: 1
// *** BATCH CMD : bsdebug -tms 0 -tdi 1 -tck 01          (20)
TDO Capture Data: 0
// *** BATCH CMD : bsdebug -tms 0 -tdi 1 -tck 01          (21)
TDO Capture Data: 1
// *** BATCH CMD : bsdebug -tms 0 -tdi 0 -tck 01          (22)
TDO Capture Data: 1
// *** BATCH CMD : bsdebug -tms 1 -tdi 0 -tck 01          (23)
TDO Capture Data: 0
// *** BATCH CMD : bsdebug -tms 1 -tdi 0 -tck 01          (24)
TDO Capture Data: 1
// *** BATCH CMD : bsdebug -tms 1 -tdi 0 -tck 01          (25)
TDO Capture Data: 1
// *** BATCH CMD : bsdebug -tms 1 -tdi 0 -tck 01          (26)
TDO Capture Data: 1
// *** BATCH CMD : bsdebug -tms 1 -tdi 0 -tck 01          (27)
```

TDO Capture Data: 1

Regarding the command number (13), the instruction load ends up in the *Test_Logic/Reset* state. Observing the remaining command lines it is possible to identify:

(14) This command has maintained the state machine on the *Test_Logic/Reset* state while the TMS signal was kept at '1'.

(15) With this command, the TMS signal was changed to low and the state machine has moved to the *Run_Test/Idle* state.

(16) Then the TMS signal was raised to logic '1', the state machine has entered in the *Select-DR-Scan* state. As this is the first state of the *Data Rank*, the TMS was changed to '0' to enter this rank.

(17) This command has moved the state machine to the *Capture-DR* state and TMS is kept at '0' to move forward to the *Shift-DR* state. Once on this state, the TDI is connected to the TDO and, according to the *Bypass* instruction, the data introduced in the TDI is shifted to the TDO with one TCK clock delay.

(19) – (22) Observing these commands, the data shifts between the TDI and the TDO have occurred according to the implemented design for the *Bypass* instruction.

(23) In this command, the TMS signal has changed to '1', moving the state machine to the *Exit1-DR* state and the TDO signal is '0'. This value for TDO is related with the last transition on the *Shift-DR* state, that has the TDI as '0' and as the *Bypass* was latched at the falling transition of TCK, while the state transition was latched on the rising transition, the last TDI value has moved to the TDO.

(24) The TMS is held at '1' in this command and the state machine moved to the *Update-DR* state. This also brings the TDO back to the high impedance state that is represented as '1' due to the pull-up circuitry.

(25) – (27) The TMS is kept at '1', making the state machine be moved to the *Test_Logic/Reset* state.

With this command sequence it is proven that the state machine is working, and that with the correct TMS patterns it goes to the implemented states. However the instructions *SAMPLE-PRELOAD* and *EXTEXT* must be also tested. With the test solution presented here it is difficult to test all the 110 implemented

Boundary-Scan Cells that represents the *Boundary-Scan Register.* To solve this problem the *Serial Vector Format* or SVF file, was used.

The SVF file format is an industry standard file format designed to describe the high-level IEEE 1149.1 bus operations. In general IEEE 1149.1 bus operations consist of scan operations and actions between different stable states on the TAP state diagram presented on *Figure 2-2*. This file format was designed to be compact, portable, simple and independent of the target device. The SVF has the format of an ASCII file containing a set of statements. The maximum allowed length of each line is 256 characters, but a statement can be longer then the length of a single line. A statement is defined as a command, with specific parameters and a semicolon termination. The SVF is not case sensitive. It also allows the insertion of comments in between or at the end of statements. The comment is inserted after the exclamation point "!" or a pair of slashes "//". Both "!" and "//" comment out the remainder of the line [SVF-ASSERT], [SVF-XILINX].

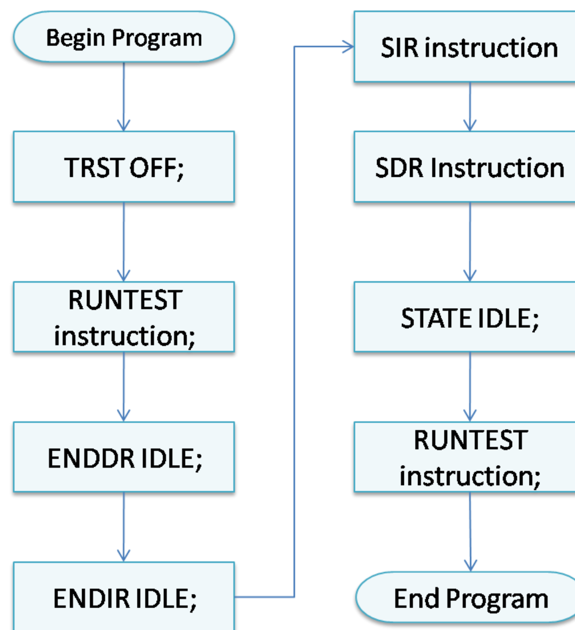


Figure 5-6: Serial Vector Format Statement Specification flowchart.

Figure 5-6 shows the flowchart of the statements sequence used on the sensor *Boundary-Scan* tests. The programs can be found on *Appendix 5 - SFV file used to test the SAPLE-PRELOAD instruction, SFV file used to test the BYPASS instruction* and *SFV file used to test the EXTEST instruction.* The first operation on the SVF was the release of the reset signal, using the statement *"TRST OFF;",* allowing access to the

Boundary-Scan logic. The following command, *RUNTEST,* defines the number of TCK clocks that the state machine was in *Run_Test/Idle* state.

This statement is not mandatory to test the *Boundary-Scan Register*, but it was used just to confirm that the state machine is working correctly. Then comes the *ENDDR* and *ENDIR* statements and those statements specify the stable state that the state machine was forced to be at the conclusion of a *Data Rank* or *Instruction Rank* scan respectively. The options were: *IDLE, RESET, DRPAUSE* or *IRPAUSE,* and the *IDLE* state is used by default. The next statements are the *SIR* and *SDR*, which specify the scan pattern to be applied to the *Instruction Register* and *Data Registers*, respectively. After the *SIR* or *SDR* word, it was defined the instruction length in number of bits to be scanned and followed with the optional fields: the TDI parameter, containing the pattern applied to the *TDI* signal, expressed in hexadecimal values; the TDO parameter, containing the pattern expected on the *TDO* signal, expressed in hexadecimal values; and the *SMASK* parameter containing hexadecimal values, in which each bit at '1' was transmitted data to TDI signal and each '0' represents don't cares. This filtering was applied to the *TDI* pattern defined with the TDI parameter. The *STATE* statement was used to move the state machine to a stable state, after the execution of the previous commands, provided that the *STATE* was in the *Run_Test/Idle* state. This means that, after the instruction command SIR or the data command SDR, the state machine will move on to the *Run_Test/Idle* state. The last code used was the repetition of *RUNTEST* command in order to keep the state machine on the *Run_Test/Idle* state. The use of the *RUNTEST* command was optional and just to stabilize the signals TDI, TDO and TMS, before the beginning and at the end of file execution.

5.2. Tests and Results of the Boundary-Scan Register

The file structure presented is generic and it was used for the three implemented instructions as an illustration of the testing procedure. Thus the only changed commands from one file to the next were the *SIR* and *SDR*. Regarding the FPGA signals that were under test, the logic values configured to be connected to the sensor during the SVF file execution, for the *Bypass* and *Sample-Preload* instructions are presented on *Table 5-1*. The execution of the *Extest* instruction requires a different signals pattern and all signals where changed to the '0' value unless the signals *N_CS_BTM* and *N_CS_TOP* that where '1'. *Table 5-1*shows the FPGA signals that were under test during the *Sample-*

Preload instruction execution, its values and the measured values through the *Boundary-Scan Chain.*

	SIGNAL	Digital Value	Measured Value
	TDI		
1	SERIAL_SYNC_TOP	1	1
2	TRANSFER_READOUT_TOP	1	1
3	LVAL_TOP	0	0
4	LVAL_TOP_ENA_tri	0	0
5	DATA_CLK_TOP	0	0
6	DATA_B_TOP_7	0	0
7	DATA_A_TOP_7	0	0
8	DATA_B_TOP_6	0	0
9	DATA_A_TOP_6	0	0
10	DATA_B_TOP_5	0	0
11	DATA_A_TOP_5	0	0
12	DATA_B_TOP_4	0	0
13	DATA_A_TOP_4	0	0
14	DATA_B_TOP_3	0	0
15	DATA_A_TOP_3	0	0
16	DATA_B_TOP_2	0	0
17	DATA_A_TOP_2	0	0
18	DATA_B_TOP_1	0	0
19	DATA_A_TOP_1	0	0
20	DATA_B_TOP_0	0	0
21	DATA_A_TOP_0	0	0
22	PIXEL_CLK_TOP	0	0
23	COLUMN_OUT_TRANSFER_EN_TOP[0]	1	1
24	COLUMN_OUT_TRANSFER_EN_TOP[1]	1	1
25	SUM_TOP	0	0
26	TDI_SEL_TOP[0]	0	0
27	TDI_SEL_TOP[1]	1	1
28	TDI_SEL_TOP[2]	1	1
29	TDI_SEL_TOP[3]	0	0
30	TDI_SEL_TOP[4]	0	0
31	WRITE_TOP	1	1
32	READ_TOP	1	1
33	BIT_SEL_TOP[0]	1	1
34	BIT_SEL_TOP[1]	0	0
35	BIT_SEL_TOP[2]	1	1
36	BIT_SEL_TOP[3]	1	1
37	OUTER_SEL_TOP[0]	1	1
38	OUTER_SEL_TOP[1]	0	0
39	OUTER_SEL_TOP[2]	0	0
40	TRANSFER_SHADOW_ADC	1	1
41	N_RST_PLL	1	1

	SIGNAL	Digital Value	Measured Value
42	MCLK	0	0
43	RST_CDS	0	0
44	INT_CDS	1	1
45	SAMPLE_CDS	1	1
46	OUTER_SEL_BTM[2]	0	0
47	OUTER_SEL_BTM[1]	1	1
48	OUTER_SEL_BTM[0]	0	0
49	BIT_SEL_BTM[3]	1	1
50	BIT_SEL_BTM[2]	1	1
51	BIT_SEL_BTM[1]	1	1
52	BIT_SEL_BTM[0]	1	1
53	READ_BTM	1	1
54	WRITE_BTM	1	1
55	TDI_SEL_BTM[4]	1	1
56	TDI_SEL_BTM[3]	1	1
57	TDI_SEL_BTM[2]	1	1
58	TDI_SEL_BTM[1]	1	1
59	TDI_SEL_BTM[0]	1	1
60	SUM_BTM	1	1
61	COLUMN_OUT_TRANSFER_EN_BTM[1]	1	1
62	COLUMN_OUT_TRANSFER_EN_BTM[0]	1	1
63	DATA_CLK_BTM	1	1
64	DATA_A_BTM_0	0	0
65	DATA_B_BTM_0	0	0
66	DATA_A_BTM_1	0	0
67	DATA_B_BTM_1	0	0
68	DATA_A_BTM_2	0	0
69	DATA_B_BTM_2	0	0
70	DATA_A_BTM_3	0	0
71	DATA_B_BTM_3	0	0
72	DATA_A_BTM_4	0	0
73	DATA_B_BTM_4	0	0
74	DATA_A_BTM_5	0	0
75	DATA_B_BTM_5	0	0
76	DATA_A_BTM_6	0	0
77	DATA_B_BTM_6	0	0
78	DATA_A_BTM_7	0	0
79	DATA_B_BTM_7	0	0
80	PIXEL_CLK_BTM	0	0
81	LVAL_BTM	0	0
82	LVAL_BTM_ENA_tri	0	0
83	TRANSFER_READOUT_BTM	0	0
84	SERIAL_SYNC_BTM	1	1
85	N_RST_LOGIC	1	1

	SIGNAL	Digital Value	Measured Value
86	ROW_TRANSFER	0	0
87	RST_CVC	1	1
88	ADD_VALID	1	1
89	ROW_ADD[11]	1	1
90	ROW_ADD[10]	1	1
91	ROW_ADD[9]	1	1
92	ROW_ADD[8]	1	1
93	ROW_ADD[7]	1	1
94	ROW_ADD[6]	1	1
95	ROW_ADD[5]	1	1
96	ROW_ADD[4]	1	1
97	ROW_ADD[3]	1	1
98	ROW_ADD[2]	1	1
99	ROW_ADD[1]	1	1
100	CLK_TEST	1	1
101	N_RST_SPI	1	1
102	FORCE_UPDATE	0	0
103	MISO_BTM	1	1
104	MISO_BTM_ENA_tri	0	0
105	N_CS_BTM	1	1
106	SCLK	1	1
107	MOSI	1	1
108	N_CS_TOP	1	1
109	MISO_TOP	1	1
110	MISO_TOP_ENA_tri	0	0
	TDO		

Table 5-1: FPGA signals under test applied to the sensor and the measured values.

The results for each instruction when the respective SVF was executed are presented on *Table 5-2* in hexadecimal representation.

Tested Instruction	Expected Values (hex)	Measured Value (hex)
Bypass	0abf	0abf
Sample-Preload	80001800007FFFF3FFFFC000037FFF3F	80001800007FFFF3FFFFC000037FFF3F
Extest	800080000000000000000000002000000e7	800080000000000000000000002000000e7

Table 5-2: Expected and measured values for each implemented instruction.

Besides the test to verify the signal integrity using the iMPACT software, the *TDI*, *TDO*, *TMS* and *TCK* signals were captured using one oscilloscope. *Figure 5-7* shows the JTAG signals captured. Regarding this capture, channel one (with colour yellow) is the TCK signal, channel two (with colour blue) is the TDI, channel three (with the colour purple) is the TMS signal and finally channel four (with the colour green) is the TDO. The capture shows the entire execution of a Bypass instruction, using the respective

SVF file. It is possible to identify the execution of the bypass instruction during the long time when the TMS signal is low. Regarding the TDO signal, is possible to observe single pulses with less than a clock period in length. This effect is related with asynchronous control signals from the TAP Controller and the pulsed clock given to the *Boundary-Scan Register*. For faster *Boundary-Scan* clocks up to 30MHz, can generate false transitions. Also, is possible to observe transitions on the falling edge of the TCK clock. The applied voltage to the sensor is 3,3V but the sensor is supplied with 1,8 IO voltage, thus the TDO is 1,8V.

Regarding the clock signal, is possible to verify that it is not a consecutive clock. This behaviour is related with the Platform Cable that sends bursts of clock cycles when configured for slower speeds. As the test speed was 6MHz and the maximum that the platform cable can perform is 10 MHz, the speed was reduced by the introduction of idle times in specific moments of the transmission instead of increasing the clock signal period.

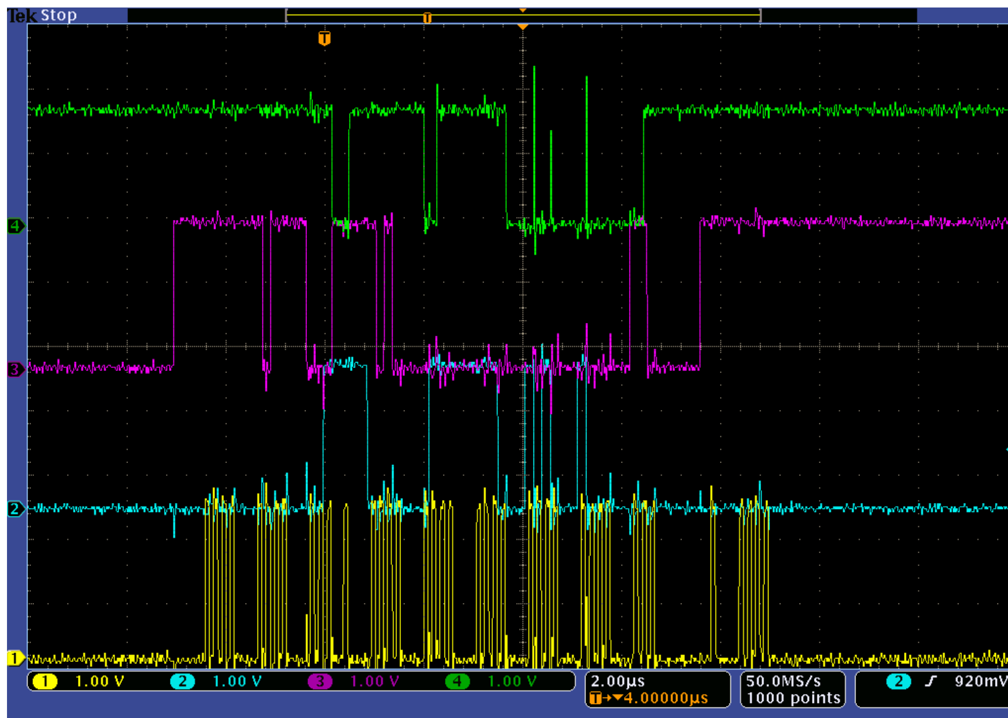


Figure 5-7: JTAG signals captured using an oscilloscope

In order to keep all signals integrity, the other signals are also stretched and during the idle time they remain in the same logic values.

The input signals TMS and TDI, changes on the rising edge of TCK signal.

5.3. Sensor Prototype BDSL file

The basic BDSL or Boundary-Scan Description Language file used with the sensor prototype is presented in *Appendix 5 - Sensor BDSL File.* The BDSL file is a VHDL or VHSIC Hardware Description Language file used to describe the boundary-scan design implemented in a IC device and its operation modes. IC devices must be provided with a BDSL to be implemented in industrial test boards. This file describes the input and output pins, the test pins, the implemented instructions, logical port description and Boundary-Scan test structures.

6. Conclusions

With this work it was possible to validate and use the *Boundary-Scan Testing* as a connectivity test verification protocol, widely used on the *Integrated Circuits* and *Printed Circuit Boards* industry.

On chapter 2 it was observed that for industrial applications the tests with the old fashion methods, such as the *Bed-of-nails* for probing each pin, are not practical and cost-efficient for small size designs, large number of connections and for large scale production of *Integrated Circuits* and *Printed Circuit Boards* boards. It was concluded that the use of the *Boundary-Scan* method can assure the requirements of large scale and small size testing, and also can be used for applications other than connectivity testing, such as on-chip block testing, firmware loading or IC identification. These features can increase even more the possibility of creating automated tests over the system, giving fast and reliable information about the system usability, saving time and reducing costs.

Likewise other *Integrated Circuits* with complex circuits and a large amount of inputs and outputs, image sensors based on CMOS technology are increasing in terms of functionality and complexity. It is common to find processing features integrated on the same die as the photosensitive area. Thus the need to control those feature forces the designers to increase the number of connections to outside the sensor. These System-on-chip become difficult to test with the common methods. The use of the *Boundary-Scan* method fulfils the requirements and permits the integration with other devices, like FPGAs or *Digital Signal Processing* devices that already have the *Boundary-Scan* implemented. It also increases the reliability and reduces the time and costs during the test phase of the development of *Image Sensors*. It is possible with this functionality to implement an automatic test architecture using, for example, a FPGA where some tests can be performed over the sensor and based on the results, the sensor can be used for the application or not.

On chapter 3 was observed that the design and simulation are compatible with the process in use for designing the sensor. All the *Boundary-Scan Testing* design was simulated successfully with the XFAB process used. The integration of the *Boundary-Scan Testing* design on the sensor design is independent of the system core. The only dependency is the pin type. In this particular case, only digital signals where prepared to be tested. The analogue ping require some more logic and analogue blocks. This is

specified in another Standard, the IEEE 1149.4 Standard. In normal operation, the sensor cannot be affected with the presence of the *Boundary-Scan Testing* design.

On chapter 4, it was verified that the implementation on chip is quite space efficient, only occupying a small space in comparison to the system core. This is an advantage not only in terms of space but also in terms of speed. Small blocks can be faster and the delay between blocks is reduced. The use of the XFAB process was very useful and practical, as the digital design blocks were available on a library obtainable from foundry, which has specified the timings, delays and signal strength for each block. Regarding the CAD tool used, the MENTOR GRAPHICS, it is a very useful and powerful tool, and can be used to design the schematic, run the simulation and create the layout in specific software modules and compare the schematic with the layout. This increases the reliability of the design.

On chapter 5 were described the functional tests made over the design. It was verified that the implemented design was working as expected, by comparing with the Standard and the simulations performed. All the implemented instructions were tested and the test results were according to the expected. Regarding the design, it was observed during the test phase that the clock distribution to the Bypass Register and Boundary-Scan Register should be improved in terms of glitch filtering. The observation of glitches is only possible with higher TCK clock speeds. For robustness, the clocks given to the Bypass Register, Instruction Register and Boundary-Scan Register should not be operated on specific states but using the TCK clock.

For test purposes, the use of an FPGA to test the sensor is very useful, because the pattern implemented can be configured on the fly or an automatic test can be implemented on it. On the present work automatic tests were not implemented and only the signals connected to the sensor where configured to perform the tests.

6.1. Future Work

As future work, the *Boundary-Scan testing* implemented can be improved, with the implementation of the *USERID* register where a 32-bit word with the device version, the device number and the manufacturer identification can be placed. This requires a registration on the *JEDEC* group.

Another future implementation could be the test of analogue pins. On the present design only digital pins are being tested. To extend the testability to all pins the

Standard IEEE 1149.4 must be implemented, as this standard specifies connectivity tests over analogue pins.

In order to increase the robustness of the actual design, the TAP controller should be redesigned to operate the Registers using synchronous control signals without interrupt the TCK clock and the TRST to them. The operation over the clock and asynchronous reset can generate glitches that could put the design in a forbidden state or, even worse, damage the system core.

In order to extend the usability of the *Boundary-Scan* signals and TAP controller, it is possible to implement test features on the system core. The goal is to verify if a specific block delivers a predefined test pattern once under test. This will test not only the connectivity but also nuclear functionalities, such as shift registers, adders, memories, and stages of internal sequential digital blocks.

References

[BST-AST,2007] *Boundary Scan Tutorial*, ASSET InterTech, Inc., 2007, PDF document.

[BSH-PARKER-USA,1998] *The Boundary-Scan Handboock – Analog and Digital*, PARKER Kenneth P., Kluwer Academic Publishers, United States of America, 1998.

[Std1149.1-Strx-AltCorp,2007] *IEEE 1149.1 (JTAG) Boundary-Scan Testing for Stratix II & Stratix II GX Devices,* Application Note SII52009-3.2, Altera Corporation, February 2007, PDF document.

[S-CIS-OHTA-USA,2008] *Smart CMOS Image Sensors and Applications*, OHTA Jun, CRC Press, United States of America, 2008.

[CMOS-Pross-design-4-IS] *Overview of CMOS process and design options for image sensor dedicated to space applications*, P. Martin-Gonthier, P.Magnan, F. Corbiere, SUPAERO – Integrated Image Sensors Laboratory, Toulouse, France. Document consulted on the web site: *http://oatao.univ-toulouse.fr/308/1/Martin-Gonthier_308.pdf, January of 2012*

[AXYS-CCD-vs-CMOS,2010] *CCD and CMOS sensor technology - Technical white paper,* AXYS Communications, 2010. Document consulted on the web site: *http://www.axis.com/files/whitepaper/wp_ccd_cmos_40722_en_1010_lo.pdf, January of 2012*

[DALSA-Im-Sens-Arch] *Image Sensor Architectures for Digital Cinematography*, DALSA Corp. Document consulted on the web site: *http://www.teledynedalsa.com/public/dc/documents/Image_Sensor_Architecture_Whitepaper_Digital_Cinema_00218-00_03-70.pdf, January of 2012*

[SPI-I2C-Protocol] *Introduction to I²C and SPI protocols,* consulted on the web site: *http://www.byteparadigm.com/kb/article/AA-00255/22/Introduction-to-SPI-and-IC-protocols.html, February of 2012*

[008Mega-CIS-SPI-I2C] 0.08 Mega CMOS Image Sensor SP0829, SuperPix Micro Technology Co., Beijing, China, 2011. Document consulted on the web site: *http://www.superpix.com.cn/cn/xiazai/SP0829/SP0829.pdf, February of 2012*

[AWA-XF768-2010] *XF-768 SHORT SPECIFICATION*, AWAIBA Lda, Madeira, Portugal, 2010. Document consulted on the web site: *http://www.awaiba.com/v2/wp-content/uploads/2010/10/XF-768_short_spec_v02.pdf, February of 2012*

[KODAK-9618-CIS] *KODAK KAC-9618CMOS IMAGE SENSOR*, Kodak Co, 2007. Document consulted on the web site:

http://www.1stvision.com/cameras/sensor_specs/KAC-9618LongSpec.pdf, February of 2012

[ESD-PROT-TI] *System-Level ESD/EMI Protection Guide,* Texas Instruments, 2010. Document consulted on the web site: *http://www.ti.com/lit/ml/sszb130a/sszb130a.pdf, February of 2012*

[XFAB-XC018] *0.18 µm CMOS Process Family*, XFAB, 2008. Document consulted on the web site: *http://www.xfab.com/fileadmin/X-FAB/Download_Center/Technology/CMOS/XC018_CMOS_Info_sheet.pdf, April of 2012*

[DA-IC-USR-1995] *Design Architect User's Manual*, Mentor Graphics Corporation, 1995. Document consulted on the web site: *http://pages.cs.wisc.edu/~sohi/cs552/Handouts/MentorDocs/design_architect_users_manual.pdf, April of 2012*

[ELDO-SIM-2005] *Eldo User's Manual*, Mentor Graphics Corporation, 2005. Document consulted on the web site: http://www.engr.uky.edu/~elias/tutorials/Eldo/eldo_ur.pdf, *April of 2012*

[ISE_10-1-03_USER] *ISE 10.1 Quick Start Tutorial*, Xilinx Inc., 2008. Document consulted on the web site: http://www.xilinx.com/itp/xilinx10/books/docs/qst/qst.pdf, *June of 2012*

[JTAG-PROG] *Platform Cable USB II*, Xilinx Inc., 2011, Document consulted on the web site: http://www.xilinx.com/support/documentation/data_sheets/ds593.pdf, *June of 2012*

[SVF-ASSET] *Serial Vector Format Specification*, ASSET InterTech, Inc. and Texas Instruments Inc., 1999. Document consulted on the web site: http://www.jtagtest.com/pdf/svf_specification.pdf, *June of 2012*

[SVF-XILINX] SVF and XSVF File Formats for Xilinx Devices, Xilinx Inc., XAPP503 (v2.1) August 17, 2009. Document consulted on the web site: http://www.xilinx.com/support/documentation/application_notes/xapp503.pdf, *June of 2012*

Appendixes

[Appendix 1] Pad frame list.

[Appendix 2] Design Schematics

[Appendix 3] Design Netlists and simulation files

[Appendix 4] Layout Drawings

[Appendix 5] SVF and BDSL files used to test the prototype

Appendix 1 – List of all prototype pins with Boundary Scan indication

Table of contents

1. Prototype Pin List..A2

1. Prototype Pin List

Pad Number	Pad Name	Type	Boundary Scan Cell
1	COLUMN_OUT_TRANSFER_EN_TOP[0]	DIG IN	BSC
2	COLUMN_OUT_TRANSFER_EN_TOP[1]	DIG IN	BSC
3	SUM_TOP	DIG IN	BSC
4	TDI_SEL_TOP[0]	DIG IN	BSC
5	TDI_SEL_TOP[1]	DIG IN	BSC
6	TDI_SEL_TOP[2]	DIG IN	BSC
7	TDI_SEL_TOP[3]	DIG IN	BSC
8	TDI_SEL_TOP[4]	DIG IN	BSC
9	VDDD	DIG PWS	
10	VDDD	DIG PWS	
11	VSSD	DIG GND	
12	VSSD	DIG GND	
13	VSSA	ANA GND	
14	VSSA	ANA GND	
15	VSSA	ANA GND	
16	VDDA	ANA 3V3	
17	VDDA	ANA 3V4	
18	VDDA	ANA 3V5	
19	VDDIO	IO PWS	
20	VDDIO	IO PWS	
21	VSSIO	IO GND	
22	VSSIO	IO GND	
23	WRITE_TOP	DIG IN	BSC
24	READ_TOP	DIG IN	BSC
25	BIT_SEL_TOP[0]	DIG IN	BSC
26	BIT_SEL_TOP[1]	DIG IN	BSC
27	BIT_SEL_TOP[2]	DIG IN	BSC
28	BIT_SEL_TOP[3]	DIG IN	BSC
29	OUTER_SEL_TOP[0]	DIG IN	BSC
30	OUTER_SEL_TOP[1]	DIG IN	BSC
31	OUTER_SEL_TOP[2]	DIG IN	BSC
32	VDDD	DIG PWS	
33	VDDD	DIG PWS	
34	VSSD	DIG GND	
35	VSSD	DIG GND	
36	VSSA	ANA GND	
37	VSSA	ANA GND	
38	VSSA	ANA GND	
39	VDDA	ANA 3V3	
40	VDDA	ANA 3V4	

41	VDDA	ANA 3V5	
42	VDDIO	IO PWS	
43	VDDIO	IO PWS	
44	VSSIO	IO GND	
45	VSSIO	IO GND	
46	TRANSFER_SHADOW_ADC	DIG IN	BSC
47	N_RST_PLL	DIG IN	BSC
48	MCLK	DIG IN	BSC Input
49	SPARE_DIG_IO_1	DIG BI-DIR	
50	SPARE_ANA_1	ANA	
51	RST_CDS	DIG IN	BSC
52	INT_CDS	DIG IN	BSC
53	SAMPLE_CDS	DIG IN	BSC
54	VSSIO	IO GND	
55	VSSIO	IO GND	
56	VDDIO	IO PWS	
57	VDDIO	IO PWS	
58	VDDA	ANA 3V3	
59	VDDA	ANA 3V4	
60	VDDA	ANA 3V5	
61	VSSA	ANA GND	
62	VSSA	ANA GND	
63	VSSA	ANA GND	
64	VSSD	DIG GND	
65	VSSD	DIG GND	
66	VDDD	DIG PWS	
67	VDDD	DIG PWS	
68	OUTER_SEL_BTM[2]	DIG IN	BSC
69	OUTER_SEL_BTM[1]	DIG IN	BSC
70	OUTER_SEL_BTM[0]	DIG IN	BSC
71	BIT_SEL_BTM[3]	DIG IN	BSC
72	BIT_SEL_BTM[2]	DIG IN	BSC
73	BIT_SEL_BTM[1]	DIG IN	BSC
74	BIT_SEL_BTM[0]	DIG IN	BSC
75	READ_BTM	DIG IN	BSC
76	WRITE_BTM	DIG IN	BSC
77	VSSIO	IO GND	
78	VSSIO	IO GND	
79	VDDIO	IO PWS	
80	VDDIO	IO PWS	
81	VDDA	ANA 3V3	
82	VDDA	ANA 3V4	
83	VDDA	ANA 3V5	
84	VSSA	ANA GND	
85	VSSA	ANA GND	

86	VSSA	ANA GND	
87	VSSD	DIG GND	
88	VSSD	DIG GND	
89	VDDD	DIG PWS	
90	VDDD	DIG PWS	
91	TDI_SEL_BTM[4]	DIG IN	BSC
92	TDI_SEL_BTM[3]	DIG IN	BSC
93	TDI_SEL_BTM[2]	DIG IN	BSC
94	TDI_SEL_BTM[1]	DIG IN	BSC
95	TDI_SEL_BTM[0]	DIG IN	BSC
96	SUM_BTM	DIG IN	BSC
97	COLUMN_OUT_TRANSFER_EN_BTM[1]	DIG IN	BSC
98	COLUMN_OUT_TRANSFER_EN_BTM[0]	DIG IN	BSC
99	SPARE_DIG_IO_2	DIG BI-DIR	
100	SPARE_ANA_2	ANA	
101	VSSD	DIG GND	
102	VSSD	DIG GND	
103	PIXEL_CLK BTM+	ANA	BSC
104	PIXEL_CLK BTM-	ANA	BSC
105	VDDD	DIG PWS	
106	VDDD	DIG PWS	
107	VDDD LVDS	LVD 1V8	
108	VSSD LVDS	LVD GND	
109	DATA_A BTM_0+	ANA	BSC
110	DATA_A BTM_0-	ANA	BSC
111	VSSD LVDS	LVD GND	
112	DATA_B BTM_0+	ANA	BSC
113	DATA_B BTM_0-	ANA	BSC
114	VSSD LVDS	LVD GND	
115	VDDD LVDS	LVD 1V8	
116	VSSA	ANA GND	
117	VSSA	ANA GND	
118	VDDA	ANA 3V3	
119	VDDA	ANA 3V4	
120	VDDD LVDS	LVD 1V8	
121	VSSD LVDS	LVD GND	
122	DATA_A BTM_1+	ANA	BSC
123	DATA_A BTM_1-	ANA	BSC
124	VSSD LVDS	LVD GND	
125	DATA_B BTM_1+	ANA	BSC
126	DATA_B BTM_1-	ANA	BSC
127	VSSD LVDS	LVD GND	
128	VDDD LVDS	LVD 1V8	
129	VSSD	DIG GND	
130	VSSD	DIG GND	

131	VDDD	DIG PWS	
132	VDDD	DIG PWS	
133	VDDD LVDS	LVD 1V8	
134	VSSD LVDS	LVD GND	
135	DATA_A BTM_2+	ANA	BSC
136	DATA_A BTM_2-	ANA	BSC
137	VSSD LVDS	LVD GND	
138	DATA_B BTM_2+	ANA	BSC
139	DATA_B BTM_2-	ANA	BSC
140	VSSD LVDS	LVD GND	
141	VDDD LVDS	LVD 1V8	
142	VSSA	ANA GND	
143	VSSA	ANA GND	
144	VDDA	ANA 3V3	
145	VDDA	ANA 3V4	
146	VDDD LVDS	LVD 1V8	
147	VSSD LVDS	LVD GND	
148	DATA_A BTM_3+	ANA	BSC
149	DATA_A BTM_3-	ANA	BSC
150	VSSD LVDS	LVD GND	
151	DATA_B BTM_3+	ANA	BSC
152	DATA_B BTM_3-	ANA	BSC
153	VSSD LVDS	LVD GND	
154	VDDD LVDS	LVD 1V8	
155	VSSD	DIG GND	
156	VSSD	DIG GND	
157	VDDD	DIG PWS	
158	VDDD	DIG PWS	
159	VDDD LVDS	LVD 1V8	
160	VSSD LVDS	LVD GND	
161	DATA_A BTM_4+	ANA	BSC
162	DATA_A BTM_4-	ANA	BSC
163	VSSD LVDS	LVD GND	
164	DATA_B BTM_4+	ANA	BSC
165	DATA_B BTM_4-	ANA	BSC
166	VSSD LVDS	LVD GND	
167	VDDD LVDS	LVD 1V8	
168	VSSA	ANA GND	
169	VSSA	ANA GND	
170	VDDA	ANA 3V3	
171	VDDA	ANA 3V4	
172	VDDD LVDS	LVD 1V8	
173	VSSD LVDS	LVD GND	
174	DATA_A BTM_5+	ANA	BSC
175	DATA_A BTM_5-	ANA	BSC
176	VSSD LVDS	LVD GND	

177	DATA_B BTM_5+	ANA	BSC
178	DATA_B BTM_5-	ANA	BSC
179	VSSD LVDS	LVD GND	
180	VDDD LVDS	LVD 1V8	
181	VSSD	DIG GND	
182	VSSD	DIG GND	
183	VDDD	DIG PWS	
184	VDDD	DIG PWS	
185	VDDD LVDS	LVD 1V8	
186	VSSD LVDS	LVD GND	
187	DATA_A BTM_6+	ANA	BSC
188	DATA_A BTM_6-	ANA	BSC
189	VSSD LVDS	LVD GND	
190	DATA_B BTM_6+	ANA	BSC
191	DATA_B BTM_6-	ANA	BSC
192	VSSD LVDS	LVD GND	
193	VDDD LVDS	LVD 1V8	
194	VSSA	ANA GND	
195	VSSA	ANA GND	
196	VDDA	ANA 3V3	
197	VDDA	ANA 3V4	
198	VDDD LVDS	LVD 1V8	
199	VSSD LVDS	LVD GND	
200	DATA_A BTM_7+	ANA	BSC
201	DATA_A BTM_7-	ANA	BSC
202	VSSD LVDS	LVD GND	
203	DATA_B BTM_7+	ANA	BSC
204	DATA_B BTM_7-	ANA	BSC
205	VSSD LVDS	LVD GND	
206	VDDD LVDS	LVD 1V8	
207	VSSD	DIG GND	
208	VSSD	DIG GND	
209	DATA_CLK BTM+	ANA	
210	DATA_CLK BTM-	ANA	
211	VDDD	DIG PWS	
212	VDDD	DIG PWS	
213	SPARE_ANA_3	ANA	
214	TEST_DIG_OUT_BTM	DIG OUT	
215	LVAL_BTM	DIG OUT	BSC + tristate BSC
216	TRANSFER_READOUT_BTM	DIG IN	BSC
217	SERIAL_SYNC_BTM	DIG IN	BSC
218	N_RST_LOGIC	DIG IN	BSC
219	SPARE_DIG_IO_4	DIG BI-DIR	
220	SPARE_ANA_4	ANA	
221	SPARE_ANA_5	ANA	

222	SPARE_ANA_6	ANA	
223	VDDD	DIG PWS	
224	VDDD	DIG PWS	
225	VSSD	DIG GND	
226	VSSD	DIG GND	
227	VSSA	ANA GND	
228	VSSA	ANA GND	
229	VSSA	ANA GND	
230	VDDA	ANA 3V3	
231	VDDA	ANA 3V4	
232	VDDA	ANA 3V5	
233	VDDIO	IO PWS	
234	VDDIO	IO PWS	
235	VSSIO	IO GND	
236	VSSIO	IO GND	
237	RST_CVC	DIG IN	BSC
238	ADD_VALID	DIG IN	BSC
239	ROW_ADD[11]	DIG IN	BSC
240	ROW_ADD[10]	DIG IN	BSC
241	ROW_ADD[9]	DIG IN	BSC
242	ROW_ADD[8]	DIG IN	BSC
243	ROW_ADD[7]	DIG IN	BSC
244	ROW_ADD[6]	DIG IN	BSC
245	ROW_ADD[5]	DIG IN	BSC
246	VDDD	DIG PWS	
247	VDDD	DIG PWS	
248	VSSD	DIG GND	
249	VSSD	DIG GND	
250	VSSA	ANA GND	
251	VSSA	ANA GND	
252	VSSA	ANA GND	
253	VDDA	ANA 3V3	
254	VDDA	ANA 3V4	
255	VDDA	ANA 3V5	
256	VDDIO	IO PWS	
257	VDDIO	IO PWS	
258	VSSIO	IO GND	
259	VSSIO	IO GND	
260	ROW_ADD[4]	DIG IN	BSC
261	ROW_ADD[3]	DIG IN	BSC
262	ROW_ADD[2]	DIG IN	BSC
263	ROW_ADD[1]	DIG IN	BSC
264	TEST_OUT_BTM	ANA	
265	CLK_TEST	DIG IN	
266	ADC_TEST	DIG IN	
267	TEST_OUT_TOP	ANA	

268	VSSIO	IO GND	
269	VSSIO	IO GND	
270	VDDIO	IO PWS	
271	VDDIO	IO PWS	
272	VDDA	ANA 3V3	
273	VDDA	ANA 3V4	
274	VDDA	ANA 3V5	
275	VSSA	ANA GND	
276	VSSA	ANA GND	
277	VSSA	ANA GND	
278	VSSD	DIG GND	
279	VSSD	DIG GND	
280	VDDD	DIG PWS	
281	VDDD	DIG PWS	
282	N_RST_SPI	DIG IN	BSC
283	FORCE_UPDATE	DIG IN	BSC
284	MISO_BTM	DIG OUT	BSC + tristate BSC
285	N_CS_BTM	DIG IN	BSC
286	SCLK	DIG IN	BSC
287	MOSI	DIG IN	BSC
288	N_CS_TOP	DIG IN	BSC
289	MISO_TOP	DIG OUT	BSC + tristate BSC
290	SPARE_DIG_IO_5	DIG BI-DIR	BSC
291	VSSIO	IO GND	
292	VSSIO	IO GND	
293	VDDIO	IO PWS	
294	VDDIO	IO PWS	
295	VDDA	ANA 3V3	
296	VDDA	ANA 3V4	
297	VDDA	ANA 3V5	
298	VSSA	ANA GND	
299	VSSA	ANA GND	
300	VSSA	ANA GND	
301	VSSD	DIG GND	
302	VSSD	DIG GND	
303	VDDD	DIG PWS	
304	VDDD	DIG PWS	
305	TRST	DIG IN	
306	TMS	DIG IN	
307	TCK	DIG IN	
308	TDI	DIG IN	
309	TDO	DIG OUT	
310	SERIAL_SYNC_TOP	DIG IN	BSC
311	TRANSFER_READOUT_TOP	DIG IN	BSC

312	LVAL_TOP	DIG OUT	BSC + tristate BSC
313	TEST_DIG_OUT_TOP	DIG OUT	
314	SPARE_ANA_7	ANA	
315	VDDD	DIG PWS	
316	VDDD	DIG PWS	
317	DATA_CLK TOP-	ANA	BSC
318	DATA_CLK TOP+	ANA	BSC
319	VSSD	DIG GND	
320	VSSD	DIG GND	
321	VDDD LVDS	LVD 1V8	
322	VSSD LVDS	LVD GND	
323	DATA_B TOP_7-	ANA	BSC
324	DATA_B TOP_7+	ANA	BSC
325	VSSD LVDS	LVD GND	
326	DATA_A_TOP_7-	ANA	BSC
327	DATA_A TOP_7+	ANA	BSC
328	VSSD LVDS	LVD GND	
329	VDDD LVDS	LVD 1V8	
330	VDDA	ANA 3V3	
331	VDDA	ANA 3V4	
332	VSSA	ANA GND	
333	VSSA	ANA GND	
334	VDDD LVDS	LVD 1V8	
335	VSSD LVDS	LVD GND	
336	DATA_B TOP_6-	ANA	BSC
337	DATA_B TOP_6+	ANA	BSC
338	VSSD LVDS	LVD GND	
339	DATA_A TOP_6-	ANA	BSC
340	DATA_A TOP_6+	ANA	BSC
341	VSSD LVDS	LVD GND	
342	VDDD LVDS	LVD 1V8	
343	VDDD	DIG PWS	
344	VDDD	DIG PWS	
345	VSSD	DIG GND	
346	VSSD	DIG GND	
347	VDDD LVDS	LVD 1V8	
348	VSSD LVDS	LVD GND	
349	DATA_B TOP_5-	ANA	BSC
350	DATA_B TOP_5+	ANA	BSC
351	VSSD LVDS	LVD GND	
352	DATA_A TOP_5-	ANA	BSC
353	DATA_A TOP_5+	ANA	BSC
354	VSSD LVDS	LVD GND	
355	VDDD LVDS	LVD 1V8	
356	VDDA	ANA 3V3	

357	VDDA	ANA 3V4	
358	VSSA	ANA GND	
359	VSSA	ANA GND	
360	VDDD LVDS	LVD 1V8	
361	VSSD LVDS	LVD GND	
362	DATA_B TOP_4-	ANA	BSC
363	DATA_B TOP_4+	ANA	BSC
364	VSSD LVDS	LVD GND	
365	DATA_A TOP_4-	ANA	BSC
366	DATA_A TOP_4+	ANA	BSC
367	VSSD LVDS	LVD GND	
368	VDDD LVDS	LVD 1V8	
369	VDDD	DIG PWS	
370	VDDD	DIG PWS	
371	VSSD	DIG GND	
372	VSSD	DIG GND	
373	VDDD LVDS	LVD 1V8	
374	VSSD LVDS	LVD GND	
375	DATA_B TOP_3-	ANA	BSC
376	DATA_B TOP_3+	ANA	BSC
377	VSSD LVDS	LVD GND	
378	DATA_A TOP_3-	ANA	BSC
379	DATA_A TOP_3+	ANA	BSC
380	VSSD LVDS	LVD GND	
381	VDDD LVDS	LVD 1V8	
382	VDDA	ANA 3V3	
383	VDDA	ANA 3V4	
384	VSSA	ANA GND	
385	VSSA	ANA GND	
386	VDDD LVDS	LVD 1V8	
387	VSSD LVDS	LVD GND	
388	DATA_B TOP_2-	ANA	BSC
389	DATA_B TOP_2+	ANA	BSC
390	VSSD LVDS	LVD GND	
391	DATA_A TOP_2+	ANA	BSC
392	DATA_A_TOP_2-	ANA	BSC
393	VSSD LVDS	LVD GND	
394	VDDD LVDS	LVD 1V8	
395	VDDD	DIG PWS	
396	VDDD	DIG PWS	
397	VSSD	DIG GND	
398	VSSD	DIG GND	
399	VDDD LVDS	LVD 1V8	
400	VSSD LVDS	LVD GND	
401	DATA_B TOP_1-	ANA	BSC
402	DATA_B TOP_1+	ANA	BSC

403	VSSD LVDS	LVD GND	
404	DATA_A TOP_1-	ANA	BSC
405	DATA_A TOP_1+	ANA	BSC
406	VSSD LVDS	LVD GND	
407	VDDD LVDS	LVD 1V8	
408	VDDA	ANA 3V3	
409	VDDA	ANA 3V4	
410	VSSA	ANA GND	
411	VSSA	ANA GND	
412	VDDD LVDS	LVD 1V8	
413	VSSD LVDS	LVD GND	
414	DATA_B TOP_0-	ANA	BSC
415	DATA_B TOP_0+	ANA	BSC
416	VSSD LVDS	LVD GND	
417	DATA_A TOP_0-	ANA	BSC
418	DATA_A TOP_0+	ANA	BSC
419	VSSD LVDS	LVD GND	
420	VDDD LVDS	LVD 1V8	
421	VDDD	DIG PWS	
422	VDDD	DIG PWS	
423	PIXEL_CLK TOP-	ANA	BSC
424	PIXEL_CLK TOP+	ANA	BSC
425	VSSD	DIG GND	
426	VSSD	DIG GND	
427	SPARE_ANA_8	ANA	
428	SPARE_DIG_IO_4	DIG BI-DIR	

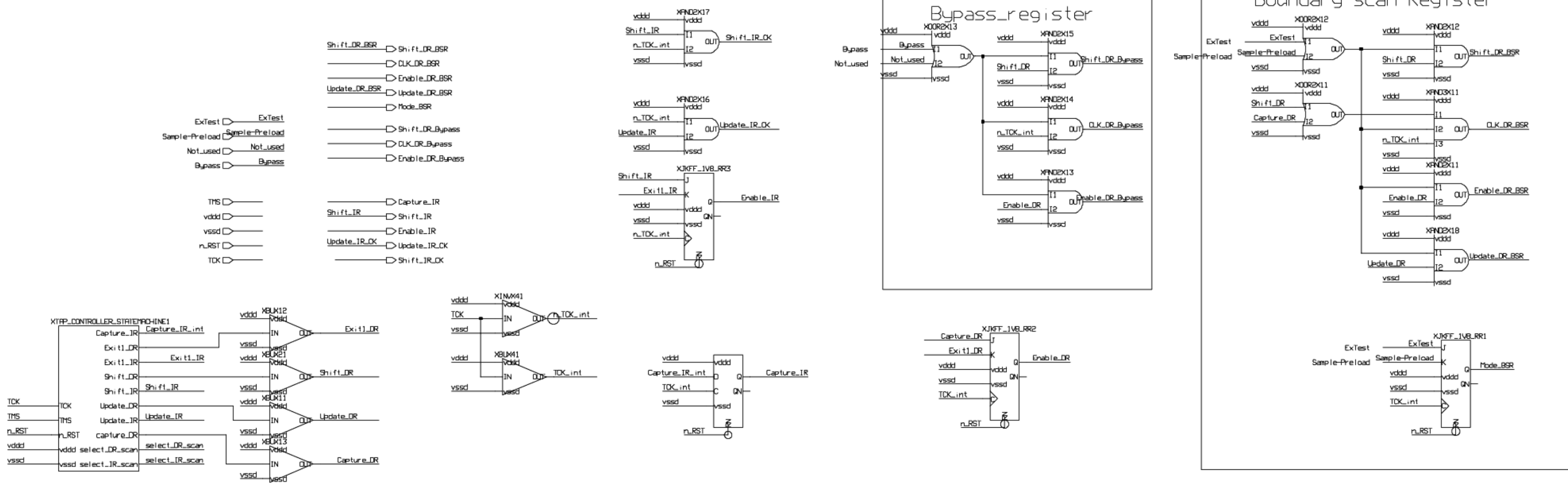
Appendix 2 - Schematics generated to the Boundary Scan Chain Blocks

Table of contents

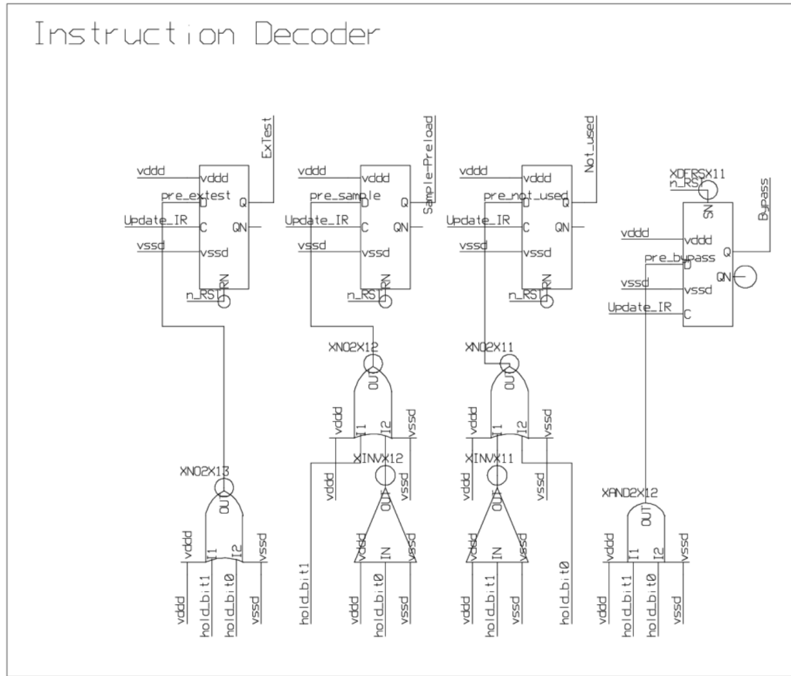
1. *TAP Controller State Machine Schematic* ---------- B2
2. *TAP Controller Block Schematic* ---------- B3
3. *Instruction Register Block Schematic* ---------- B4
4. *Bypass Register Block Schematic* ---------- B5
5. *Boundary-Scan Cell Schematic* ---------- B5
6. *Boundary-Scan Input Cell Schematic* ---------- B6
7. *Boundary-Scan Register Block Schematic* ---------- B7
8. *Boundary-Scan Control Logic Schematic* ---------- B8
9. *Boundary-Scan Top Level Simulation Schematic* ---------- B9
10. *Boundary-Scan Top Level Schematic* ---------- B10

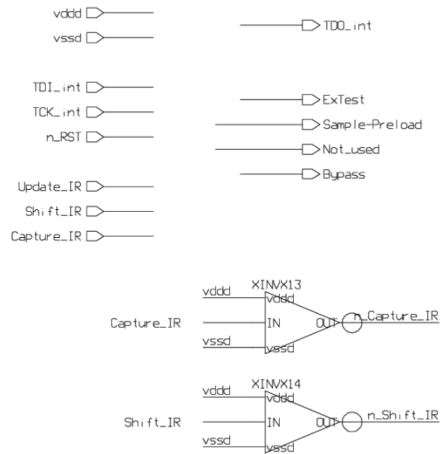
1. TAP Controller State Machine Schematic

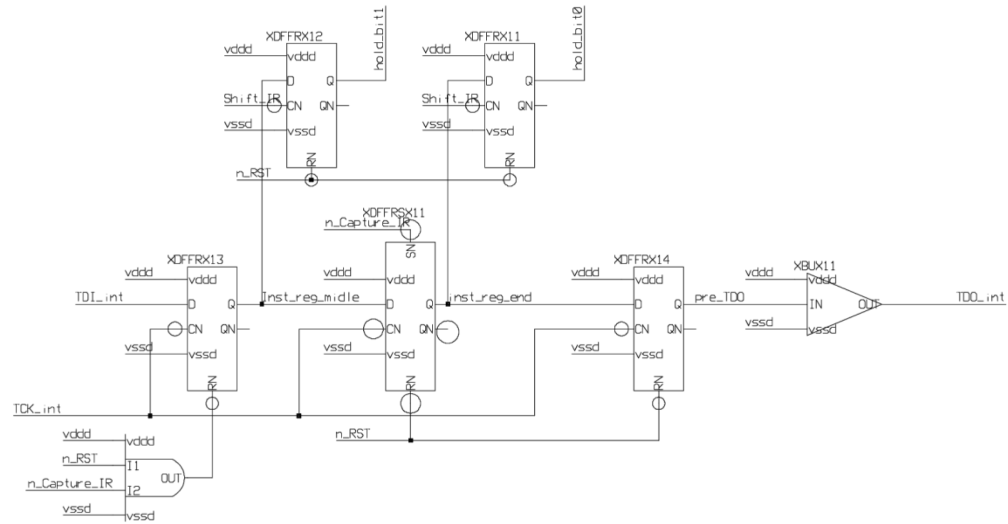
2. TAP Controller Block Schematic



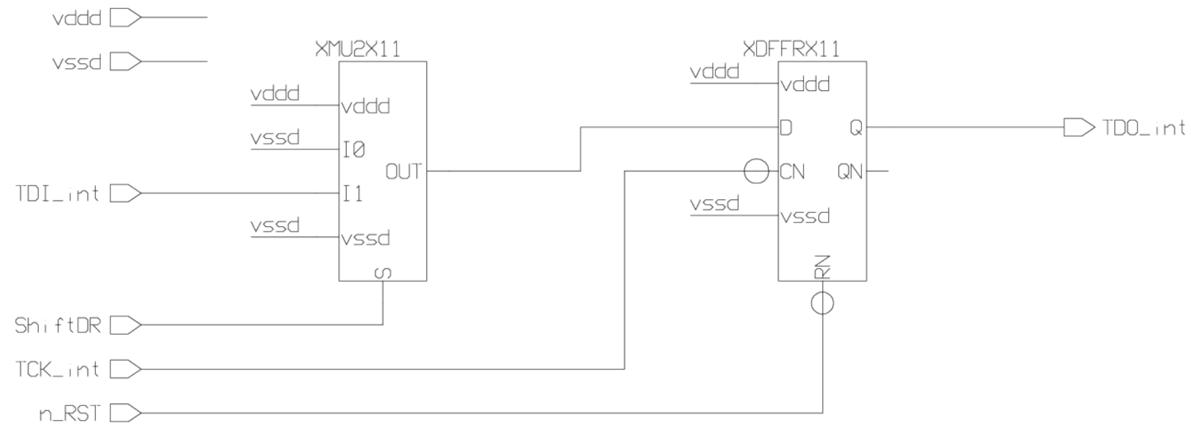
3. Instruction Register Block Schematic



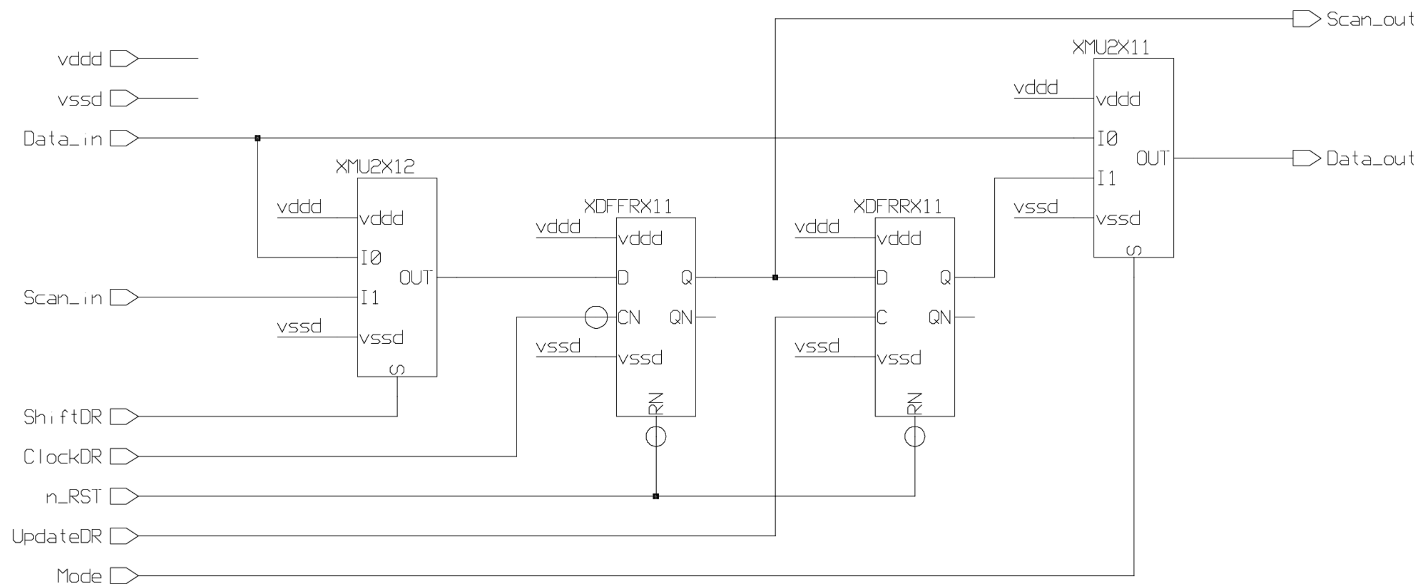




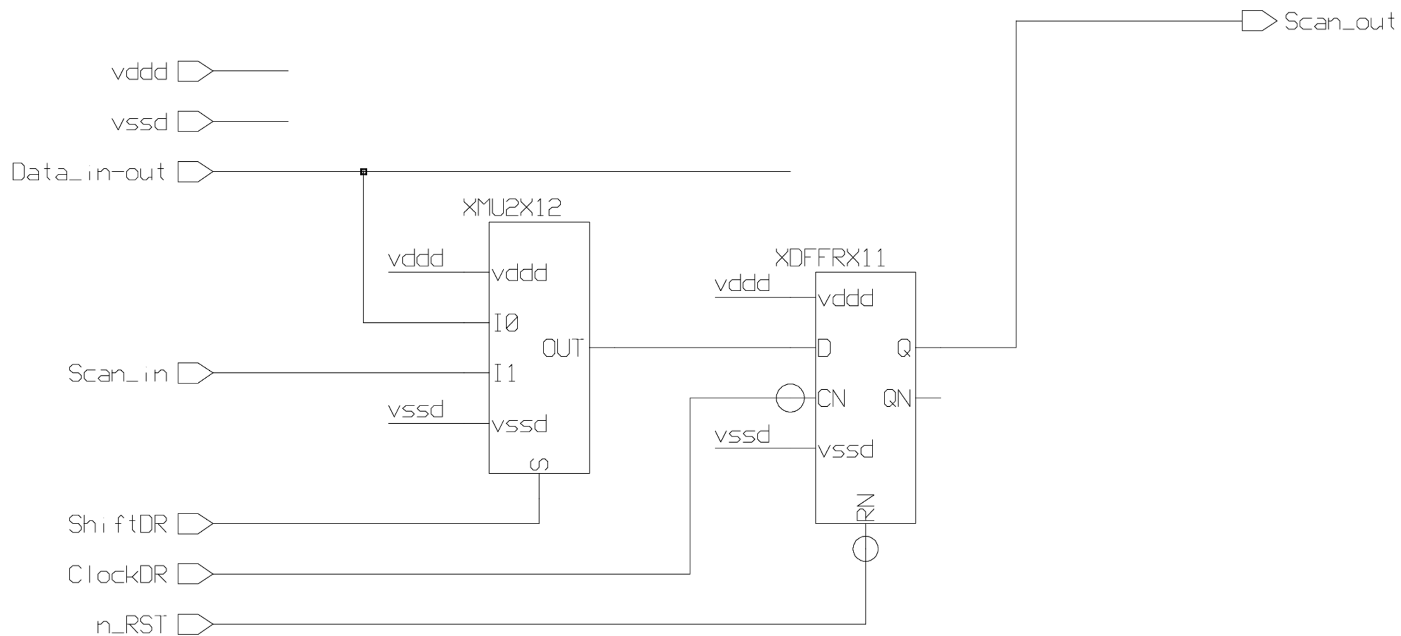
4. Bypass Register Block Schematic



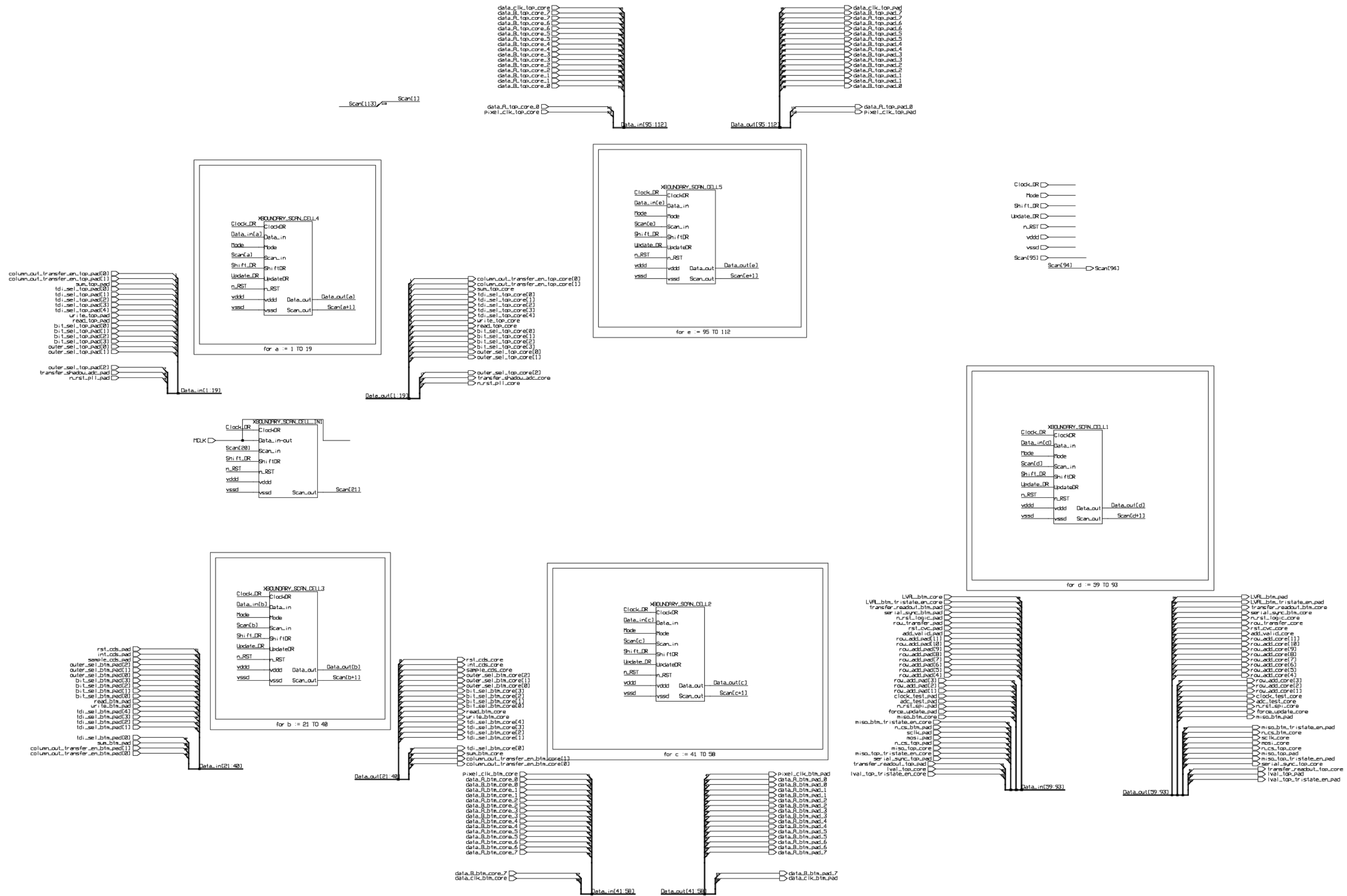
5. Boundary-Scan Cell Schematic



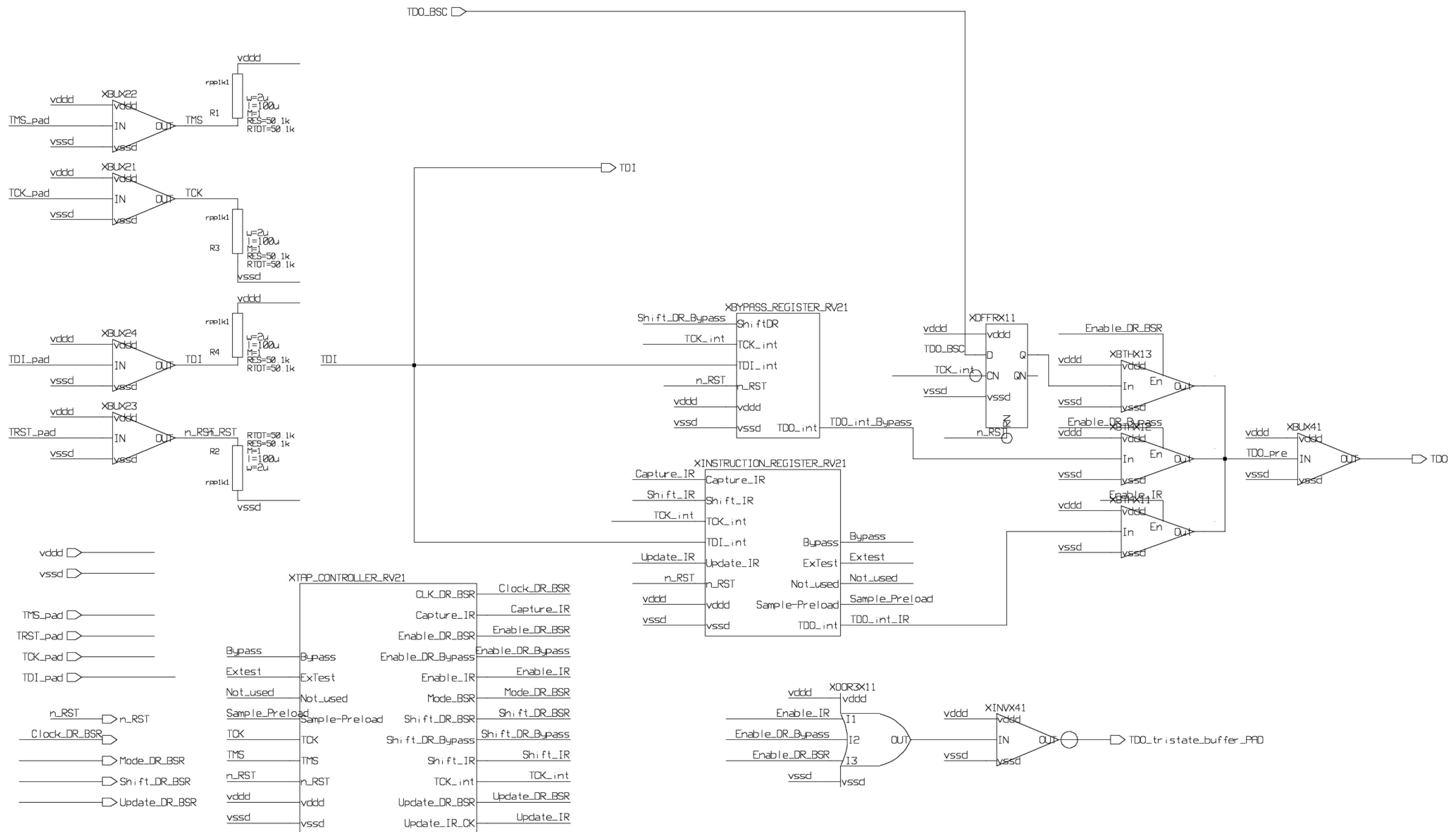
6. Boundary-Scan Input Cell Schematic



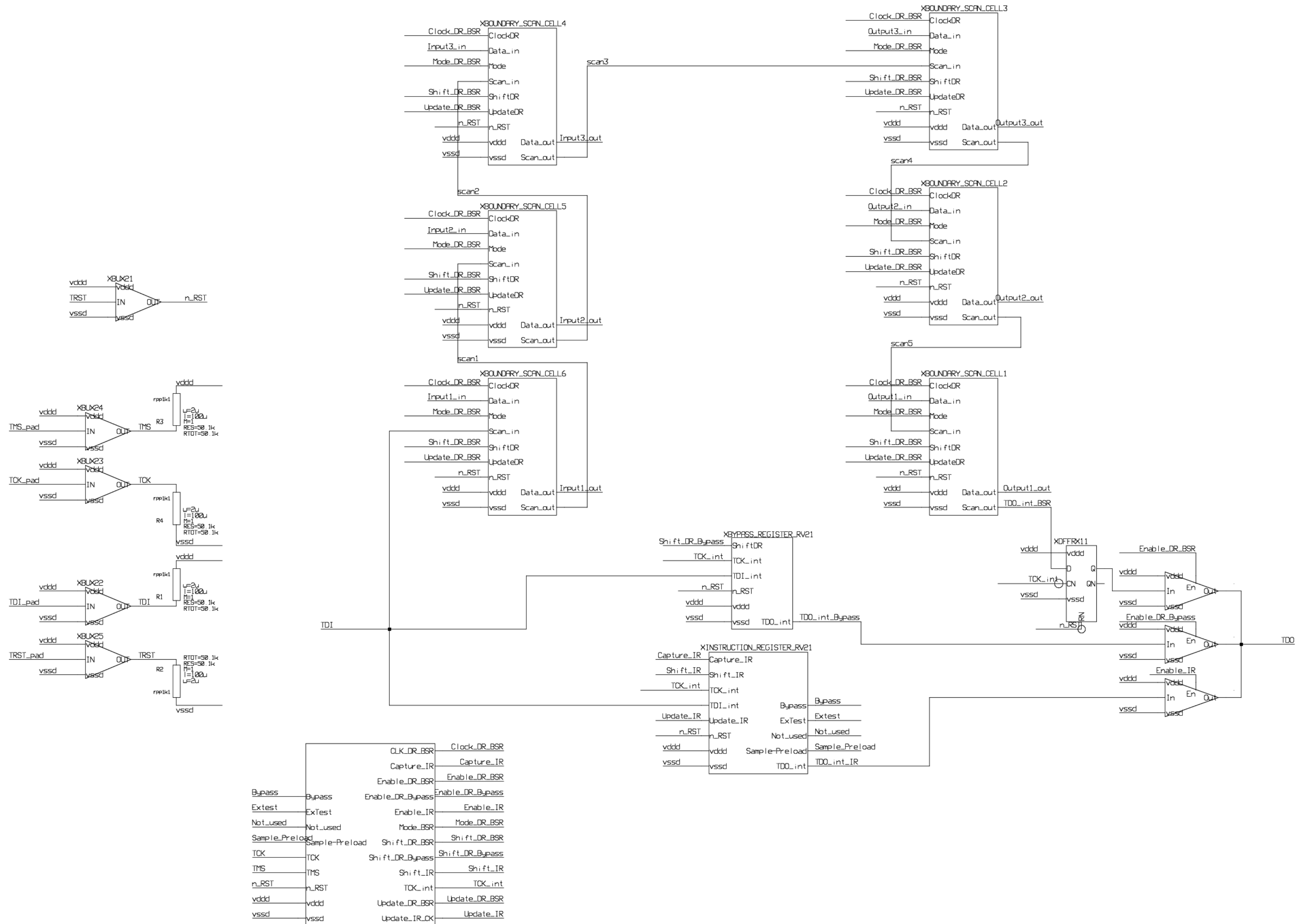
7. Boundary-Scan Register Block Schematic



8. Boundary-Scan Control Logic Schematic



9. Boundary-Scan Top Level Simulation Schematic



10.Boundary-Scan Top Level Schematic

Appendix 3 – Simulation files and netlists to the Boundary Scan Chain Blocks

Table of contents

1. *TAP Controller State Machine Netlist*.......... C2
2. *TAP Controller State Machine Simulation file*.......... C6
3. *Boundary-Scan Top Level Simulation Netlist* C8
4. *Boundary-Scan Top Level Simulation file*.......... C20

1. TAP Controller State Machine Netlist

```
*
* .CONNECT statements
*
.CONNECT vssd 0

* ELDO netlist on Tue Out 14 2011 at 17:15:12

*
* component pathname : $MGC_DESIGN_KIT/PRIMLIB.group/logic.views/ne [ELDOSPICE]
*
*       .include $TECHNO_LIB/PRIMLIB.group/logic.views/ne/ne.sub

*
* component pathname : $MGC_DESIGN_KIT/PRIMLIB.group/logic.views/pe [ELDOSPICE]
*
*       .include $TECHNO_LIB/PRIMLIB.group/logic.views/pe/pe.sub
*
* component pathname : $MASTERS/da/LOGIC/DFRRSX1
*
.subckt DFRRSX1  Q QN C D RN SN VDDD VSSD

        X_M4  Q SQIB VSSD VSSD ne w=0.66u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M3  Q SQIB VDDD VDDD pe w=0.99u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M2  QN SQI VSSD VSSD ne w=0.66u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M1  QN SQI VDDD VDDD pe w=0.99u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M32  CIB C VSSD VSSD ne w=0.66u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M31  CIB C VDDD VDDD pe w=0.99u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M29  CI CIB VSSD VSSD ne w=0.66u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M30  CI CIB VDDD VDDD pe w=0.99u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M34  MQI SN VDDD VDDD pe w=0.5u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M33  N$1 SN VSSD VSSD ne w=0.26u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M36  SQI SN VDDD VDDD pe w=0.42u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M35  N$15 SN VSSD VSSD ne w=0.22u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M27  MQIB CIB N$4 VSSD ne w=0.22u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M28  MQIB CI N$6 VDDD pe w=0.49u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M26  N$6 D VDDD VDDD pe w=0.7u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M25  N$4 D N$8 VSSD ne w=0.35u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M24  MQIB CI N$10 VSSD ne w=0.22u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M23  MQIB CIB N$12 VDDD pe w=0.22u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M22  N$12 MQI VDDD VDDD pe w=0.22u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M21  N$10 MQI N$8 VSSD ne w=0.22u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M20  MQIB RN VDDD VDDD pe w=0.44u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M19  N$8 RN VSSD VSSD ne w=0.54u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M18  MQI MQIB N$1 VSSD ne w=0.26u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M17  MQI MQIB VDDD VDDD pe w=0.44u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M16  SQIB CI N$18 VSSD ne w=0.26u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M15  SQIB CIB N$20 VDDD pe w=0.48u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
```

```
+  nrs=0.135 nrd=0.135 par1=1
        X_M14  N$20 MQI VDDD VDDD pe w=0.48u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M13  N$18 MQI N$21 VSSD ne w=0.26u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M12  SQIB CIB N$23 VSSD ne w=0.22u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M11  SQIB CI N$25 VDDD pe w=0.33u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M10  N$25 SQI VDDD VDDD pe w=0.33u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M9  N$23 SQI N$21 VSSD ne w=0.22u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M8  SQIB RN VDDD VDDD pe w=0.72u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M7  N$21 RN VSSD VSSD ne w=0.4u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M6  SQI SQIB N$15 VSSD ne w=0.22u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M5  SQI SQIB VDDD VDDD pe w=0.33u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
.ends DFRRSX1
*
* component pathname : $MASTERS/da/LOGIC/DFRRX1
*
.subckt DFRRX1  Q QN C D RN VDDD VSSD

        X_M14  N$20 MQI VDDD VDDD pe w=0.48u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M13  N$18 MQI N$21 VSSD ne w=0.26u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M12  SQIB CIB N$22 VSSD ne w=0.22u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M11  SQIB CI N$23 VDDD pe w=0.33u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M10  N$23 SQI VDDD VDDD pe w=0.33u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M9  N$22 SQI N$21 VSSD ne w=0.22u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M8  SQIB RN VDDD VDDD pe w=0.72u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M7  N$21 RN VSSD VSSD ne w=0.4u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M6  SQI SQIB VSSD VSSD ne w=0.22u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M5  SQI SQIB VDDD VDDD pe w=0.33u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M4  Q SQIB VSSD VSSD ne w=0.66u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M3  Q SQIB VDDD VDDD pe w=0.99u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M2  QN SQI VSSD VSSD ne w=0.66u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M1  QN SQI VDDD VDDD pe w=0.99u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M32  CIB C VSSD VSSD ne w=0.66u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M31  CIB C VDDD VDDD pe w=0.99u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M30  CI CIB VDDD VDDD pe w=0.99u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M29  CI CIB VSSD VSSD ne w=0.66u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M28  MQIB CI N$4 VDDD pe w=0.49u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M27  MQIB CIB N$2 VSSD ne w=0.22u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M26  N$4 D VDDD VDDD pe w=0.7u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M25  N$2 D N$11 VSSD ne w=0.35u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M24  MQIB CI N$6 VSSD ne w=0.22u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M23  MQIB CIB N$7 VDDD pe w=0.22u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M22  N$7 MQI VDDD VDDD pe w=0.22u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
```

```
        X_M21  N$6 MQI N$11 VSSD ne w=0.22u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M20  MQIB RN VDDD VDDD pe w=0.44u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M19  N$11 RN VSSD VSSD ne w=0.54u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M18  MQI MQIB VSSD VSSD ne w=0.26u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M17  MQI MQIB VDDD VDDD pe w=0.44u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M16  SQIB CI N$18 VSSD ne w=0.26u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M15  SQIB CIB N$20 VDDD pe w=0.48u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
.ends DFRRX1
*
* component pathname : $MASTERS/da/LOGIC/OOR4X0
*
.subckt OOR4X0  Q I1 I2 I3 I4 VDDD VSSD

        X_M12  N$450 I4 VDDD VDDD pe w=0.33u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M11  N$237 I1 VSSD VSSD ne w=0.22u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M10  N$446 I2 VDDD VDDD pe w=0.33u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M9  N$237 I1 N$446 VDDD pe w=0.33u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M8  N$237 I2 VSSD VSSD ne w=0.22u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M7  N$234 I3 N$450 VDDD pe w=0.33u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M6  N$234 I4 VSSD VSSD ne w=0.22u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M5  N$234 I3 VSSD VSSD ne w=0.22u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M4  N$444 N$234 VSSD VSSD ne w=0.26u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u
pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M3  Q N$237 N$444 VSSD ne w=0.26u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M2  Q N$237 VDDD VDDD pe w=0.3u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M1  Q N$234 VDDD VDDD pe w=0.3u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
.ends OOR4X0
*
* component pathname : $MASTERS/da/LOGIC/AND2X1
*
.subckt AND2X1  OUT I1 I2 VDDD VSSD

        X_M4  N$4 I2 VDDD VDDD pe w=0.66u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M2  N$4 I1 N$3 VSSD ne w=0.66u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M3  N$4 I1 VDDD VDDD pe w=0.66u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M1  N$3 I2 VSSD VSSD ne w=0.66u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M5  OUT N$4 VSSD VSSD ne w=0.66u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M6  OUT N$4 VDDD VDDD pe w=0.99u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
.ends AND2X1
*
* component pathname : $MASTERS/da/LOGIC/INVX1
*
.subckt INVX1  OUT IN VDDD VSSD

        X_M2  OUT IN VDDD VDDD pe w=1.62u l=0.185u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M1  OUT IN VSSD VSSD ne w=0.66u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
.ends INVX1
*
* component pathname : $MASTERS/da/JTAG/TDEC
*
```

```
.subckt TDEC  Q0 Q1 IN S VDDD VSSD

        X_I$1  Q1 S IN VDDD VSSD AND2X1
        X_I$2  Q0 N$7 IN VDDD VSSD AND2X1
        X_I$3  N$7 S VDDD VSSD INVX1
.ends TDEC
*
* component pathname : $MASTERS/da/LOGIC/OOR3X0
*
.subckt OOR3X0  OUT I1 I2 I3 VDDD VSSD

        X_M1  N$8 I1 VSSD VSSD ne w=0.22u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M4  N$8 I1 N$213 VDDD pe w=0.33u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M3  N$8 I3 VSSD VSSD ne w=0.22u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M2  N$8 I2 VSSD VSSD ne w=0.22u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M5  N$213 I2 N$211 VDDD pe w=0.33u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M6  N$211 I3 VDDD VDDD pe w=0.33u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M8  OUT N$8 VDDD VDDD pe w=0.33u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M7  OUT N$8 VSSD VSSD ne w=0.22u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
.ends OOR3X0
*
* component pathname : $MASTERS/da/LOGIC/OOR2X1
*
.subckt OOR2X1  OUT I1 I2 VDDD VSSD

        X_M4  N$1 I2 VDDD VDDD pe w=0.7u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M3  N$4 I1 N$1 VDDD pe w=0.7u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
nrs=0.135
+  nrd=0.135 par1=1
        X_M2  N$4 I2 VSSD VSSD ne w=0.33u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M1  N$4 I1 VSSD VSSD ne w=0.33u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M6  OUT N$4 VDDD VDDD pe w=1u l=0.184u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
        X_M5  OUT N$4 VSSD VSSD ne w=0.66u l=0.18u m=1 as=0.96p ad=0.96p ps=4.96u pd=4.96u
+  nrs=0.135 nrd=0.135 par1=1
.ends OOR2X1
*
* component pathname : $MASTERS/da/JTAG/TAP_Controller_StateMachine
*
.subckt TAP_CONTROLLER_STATEMACHINE  CAPTURE_DR CAPTURE_IR SELECT_DR_SCAN SELECT_IR_SCAN
+ SHIFT_DR SHIFT_IR UPDATE_DR UPDATE_IR N_RST TCK TMS VDDD VSSD

        X_I$881  TEST_LOGIC-RESET N$1346 TCK N$1354 VDDD N_RST VDDD VSSD DFRRSX1
        XDFRRX16  CAPTURE_IR N$183 TCK N$1394 N_RST VDDD VSSD DFRRX1
        XDFRRX15  SHIFT_IR N$186 TCK N$1370 N_RST VDDD VSSD DFRRX1
        XDFRRX11  UPDATE_IR N$206 TCK N$1400 N_RST VDDD VSSD DFRRX1
        XDFRRX14  EXIT1_IR N$191 TCK N$1404 N_RST VDDD VSSD DFRRX1
        XDFRRX13  PAUSE_IR N$197 TCK N$1374 N_RST VDDD VSSD DFRRX1
        XDFRRX12  EXIT2_IR N$202 TCK N$1375 N_RST VDDD VSSD DFRRX1
        XOOR4X01  N$1355 N$22 N$1056 N$1305 N$1342 VDDD VSSD OOR4X0
        XDFRRX19  EXIT1_DR N$73 TCK N$1362 N_RST VDDD VSSD DFRRX1
        XDFRRX115  RUN_TEST-IDLE N$15 TCK N$1355 N_RST VDDD VSSD DFRRX1
        XDFRRX114  UPDATE_DR N$111 TCK N$1368 N_RST VDDD VSSD DFRRX1
        XDFRRX113  SELECT_DR_SCAN N$25 TCK N$1391 N_RST VDDD VSSD DFRRX1
        XDFRRX112  SELECT_IR_SCAN N$36 TCK N$1390 N_RST VDDD VSSD DFRRX1
        XDFRRX111  CAPTURE_DR N$49 TCK N$1359 N_RST VDDD VSSD DFRRX1
        XDFRRX18  PAUSE_DR N$86 TCK N$1364 N_RST VDDD VSSD DFRRX1
        XDFRRX110  SHIFT_DR N$58 TCK N$1361 N_RST VDDD VSSD DFRRX1
        XDFRRX17  EXIT2_DR N$102 TCK N$1386 N_RST VDDD VSSD DFRRX1
        X_I$667  N$1300 N$1301 EXIT2_DR TMS VDDD VSSD TDEC
        X_I$674  N$1305 N$1306 UPDATE_DR TMS VDDD VSSD TDEC
        XOOR3X01  N$1370 N$1333 N$1316 N$1311 VDDD VSSD OOR3X0
        X_I$675  N$1316 N$1317 SHIFT_IR TMS VDDD VSSD TDEC
        X_I$676  N$1322 N$1323 EXIT1_IR TMS VDDD VSSD TDEC
        X_I$677  N$1327 N$1375 PAUSE_IR TMS VDDD VSSD TDEC
        X_I$678  N$1333 N$1334 EXIT2_IR TMS VDDD VSSD TDEC
```

```
        X_I$462  N$1056 N$9 TEST_LOGIC-RESET TMS VDDD VSSD TDEC
        XOOR3X03  N$1391 N$1343 N$1306 N$130 VDDD VSSD OOR3X0
        X_I$669  N$1276 N$1277 CAPTURE_DR TMS VDDD VSSD TDEC
        X_I$463  N$22 N$130 RUN_TEST-IDLE TMS VDDD VSSD TDEC
        X_I$464  N$1359 N$1390 SELECT_DR_SCAN TMS VDDD VSSD TDEC
        XOOR3X02  N$1361 N$1300 N$1282 N$1276 VDDD VSSD OOR3X0
        X_I$668  N$1394 N$1271 SELECT_IR_SCAN TMS VDDD VSSD TDEC
        X_I$670  N$1282 N$1283 SHIFT_DR TMS VDDD VSSD TDEC
        X_I$671  N$1288 N$1289 EXIT1_DR TMS VDDD VSSD TDEC
        X_I$672  N$1294 N$1386 PAUSE_DR TMS VDDD VSSD TDEC
        X_I$673  N$1311 N$1344 CAPTURE_IR TMS VDDD VSSD TDEC
        X_I$679  N$1342 N$1343 UPDATE_IR TMS VDDD VSSD TDEC
        XOOR2X11  N$1400 N$1334 N$1323 VDDD VSSD OOR2X1
        XOOR2X12  N$1374 N$1322 N$1327 VDDD VSSD OOR2X1
        XOOR2X13  N$1404 N$1317 N$1344 VDDD VSSD OOR2X1
        XOOR2X15  N$1362 N$1283 N$1277 VDDD VSSD OOR2X1
        XOOR2X17  N$1368 N$1289 N$1301 VDDD VSSD OOR2X1
        XOOR2X16  N$1354 N$9 N$1271 VDDD VSSD OOR2X1
        XOOR2X14  N$1364 N$1288 N$1294 VDDD VSSD OOR2X1
.ends TAP_CONTROLLER_STATEMACHINE
*
* MAIN CELL: component pathname :
$MASTERS/da/JTAG/TEST_and_SIMULATIONS/Test_TAP_controller_stateMachine
*
        XTAP_CONTROLLER_STATEMACHINE1  N$214 N$213 SELECT_DR_SCAN SELECT_IR_SCAN SHIFT_DR
+  SHIFT_IR UPDATE_DR UPDATE_IR N_RST TCK TMS VDDD VSSD TAP_CONTROLLER_STATEMACHINE
*
* eldo include file.
*
*
.end
```

2. TAP Controller State Machine Simulation file

```
* Component: $MASTERS/da/JTAG/TEST_and_SIMULATIONS/Test_TAP_controller_stateMachine
Viewpoint: eldonet
*** TEST JTAG TAP controller for Boundary-scan chain tests - MASTERS PROJECT ***

** Netlist include
.INCLUDE Test_TAP_controller_stateMachine_eldonet.spi

** Device Models include
.LIB $MGC_DESIGN_KIT/eldo/xc018_R3.0/mosst/bsim3v3.lib TM
.LIB $MGC_DESIGN_KIT/eldo/xc018_R3.0/mosst/param.lib 3S

**** SIMULATOR OPTIONS ****
.OPTION NOASCII
.OPTION MODWL
.OPTION SPICEDC
.OPTION AEX
.OPTION ENGNOT
.OPTION PROBOP2
.OP

**** SIMULATOR OPTIONS ****
.temp 55
.option gmin = 10e-15
.option hmax = 0.1n

**** POWER SUPPLY ****
.param supply_voltage = 1.8
vpower vddd vssd dc supply_voltage
*********************************************************************************************
*******************

*** Type of Simulation ***
.tran 0 1.3u

*** Set input values for state Machine ***

vrst n_RST vssd pattern supply_voltage 0 0 0.5n 0.5n 30n 01
vclk   TCK vssd pattern supply_voltage 0 0 0.5n 0.5n 15n 10 R
vtms   TMS vssd pattern supply_voltage 0 27n 0.5n 0.5n 30n 111 0011 1011 0000 1001 0011 0101
0010 0111 1111
```

```
**** PLOT DEFINITIONS ****

*** Plot inputs ***
.plot tran v(n_RST)
.plot tran v(TCK)
.plot tran v(TMS)

*** Plot oututs and test points ***

.plot tran v(XTAP_CONTROLLER_STATEMACHINE1.TMS)
.plot tran v(XTAP_CONTROLLER_STATEMACHINE1.test_logic-reset)
.plot tran v(XTAP_CONTROLLER_STATEMACHINE1.run_test-idle)
.plot tran v(XTAP_CONTROLLER_STATEMACHINE1.select_DR_scan)
.plot tran v(XTAP_CONTROLLER_STATEMACHINE1.select_IR_scan)

.plot tran v(XTAP_CONTROLLER_STATEMACHINE1.capture_DR)
.plot tran v(XTAP_CONTROLLER_STATEMACHINE1.Shift_DR)
.plot tran v(XTAP_CONTROLLER_STATEMACHINE1.Exit1_DR)
.plot tran v(XTAP_CONTROLLER_STATEMACHINE1.Pause_DR)
.plot tran v(XTAP_CONTROLLER_STATEMACHINE1.Exit2_DR)
.plot tran v(XTAP_CONTROLLER_STATEMACHINE1.Update_DR)

.plot tran v(XTAP_CONTROLLER_STATEMACHINE1.capture_IR)
.plot tran v(XTAP_CONTROLLER_STATEMACHINE1.Shift_IR)
.plot tran v(XTAP_CONTROLLER_STATEMACHINE1.Exit1_IR)
.plot tran v(XTAP_CONTROLLER_STATEMACHINE1.Pause_IR)
.plot tran v(XTAP_CONTROLLER_STATEMACHINE1.Exit2_IR)
.plot tran v(XTAP_CONTROLLER_STATEMACHINE1.Update_IR)
```

3. Boundary-Scan Top Level Simulation Netlist

```
*
* .CONNECT statements
*
.CONNECT vssd 0


* ELDO netlist on Thu Out 21 2011 at 10:42:13

*
* component pathname : $MGC_DESIGN_KIT/PRIMLIB.group/logic.views/rpp1k1 [ELDOSPICE]
*
*       .include $TECHNO_LIB/PRIMLIB.group/logic.views/rpp1k1/rpp1k1.sub

*
* component pathname : $MGC_DESIGN_KIT/PRIMLIB.group/logic.views/ne [ELDOSPICE]
*
*       .include $TECHNO_LIB/PRIMLIB.group/logic.views/ne/ne.sub

*
* component pathname : $MGC_DESIGN_KIT/PRIMLIB.group/logic.views/pe [ELDOSPICE]
*
*       .include $TECHNO_LIB/PRIMLIB.group/logic.views/pe/pe.sub
*
* component pathname : $MASTERS/da/LOGIC/DFRRSX1
*
.subckt DFRRSX1  Q QN C D RN SN VDDD VSSD

        X_M4  Q SQIB VSSD VSSD ne w=0.66u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M3  Q SQIB VDDD VDDD pe w=0.99u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M2  QN SQI VSSD VSSD ne w=0.66u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M1  QN SQI VDDD VDDD pe w=0.99u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M32  CIB C VSSD VSSD ne w=0.66u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M31  CIB C VDDD VDDD pe w=0.99u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M29  CI CIB VSSD VSSD ne w=0.66u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M30  CI CIB VDDD VDDD pe w=0.99u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M34  MQI SN VDDD VDDD pe w=0.5u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M33  N$1 SN VSSD VSSD ne w=0.26u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M36  SQI SN VDDD VDDD pe w=0.42u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M35  N$15 SN VSSD VSSD ne w=0.22u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M27  MQIB CIB N$4 VSSD ne w=0.22u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M28  MQIB CI N$6 VDDD pe w=0.49u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M26  N$6 D VDDD VDDD pe w=0.7u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M25  N$4 D N$8 VSSD ne w=0.35u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M24  MQIB CI N$10 VSSD ne w=0.22u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M23  MQIB CIB N$12 VDDD pe w=0.22u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
```

```
        X_M22  N$12 MQI VDDD VDDD pe w=0.22u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M21  N$10 MQI N$8 VSSD ne w=0.22u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M20  MQIB RN VDDD VDDD pe w=0.44u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M19  N$8 RN VSSD VSSD ne w=0.54u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M18  MQI MQIB N$1 VSSD ne w=0.26u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M17  MQI MQIB VDDD VDDD pe w=0.44u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M16  SQIB CI N$18 VSSD ne w=0.26u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M15  SQIB CIB N$20 VDDD pe w=0.48u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M14  N$20 MQI VDDD VDDD pe w=0.48u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M13  N$18 MQI N$21 VSSD ne w=0.26u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M12  SQIB CIB N$23 VSSD ne w=0.22u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M11  SQIB CI N$25 VDDD pe w=0.33u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M10  N$25 SQI VDDD VDDD pe w=0.33u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M9  N$23 SQI N$21 VSSD ne w=0.22u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M8  SQIB RN VDDD VDDD pe w=0.72u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M7  N$21 RN VSSD VSSD ne w=0.4u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M6  SQI SQIB N$15 VSSD ne w=0.22u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M5  SQI SQIB VDDD VDDD pe w=0.33u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
.ends DFRRSX1
*
* component pathname : $MASTERS/da/LOGIC/OOR4X1
*
.subckt OOR4X1  Q I1 I2 I3 I4 VDDD VSSD

        X_M12  N$7 I4 VDDD VDDD pe w=0.7u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M11  N$11 I1 VSSD VSSD ne w=0.33u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M10  N$3 I2 VDDD VDDD pe w=0.7u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M9  N$11 I1 N$3 VDDD pe w=0.7u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M8  N$11 I2 VSSD VSSD ne w=0.33u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M7  N$9 I3 N$7 VDDD pe w=0.7u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M6  N$9 I4 VSSD VSSD ne w=0.33u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M5  N$9 I3 VSSD VSSD ne w=0.33u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
```

```
        X_M4  N$1 N$9 VSSD VSSD ne w=0.80u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M3  Q N$11 N$1 VSSD ne w=0.80u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M2  Q N$11 VDDD VDDD pe w=1.0u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M1  Q N$9 VDDD VDDD pe w=1.0u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
.ends OOR4X1
*
* component pathname : $MASTERS/da/LOGIC/DFRRX1
*
.subckt DFRRX1  Q QN C D RN VDDD VSSD

        X_M8  SQIB RN VDDD VDDD pe w=0.72u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M7  N$21 RN VSSD VSSD ne w=0.4u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M6  SQI SQIB VSSD VSSD ne w=0.22u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M5  SQI SQIB VDDD VDDD pe w=0.33u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M4  Q SQIB VSSD VSSD ne w=0.66u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M3  Q SQIB VDDD VDDD pe w=0.99u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M2  QN SQI VSSD VSSD ne w=0.66u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M1  QN SQI VDDD VDDD pe w=0.99u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M32  CIB C VSSD VSSD ne w=0.66u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M31  CIB C VDDD VDDD pe w=0.99u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M30  CI CIB VDDD VDDD pe w=0.99u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M29  CI CIB VSSD VSSD ne w=0.66u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M28  MQIB CI N$4 VDDD pe w=0.49u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M27  MQIB CIB N$2 VSSD ne w=0.22u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M26  N$4 D VDDD VDDD pe w=0.7u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M25  N$2 D N$11 VSSD ne w=0.35u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M24  MQIB CI N$6 VSSD ne w=0.22u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M23  MQIB CIB N$7 VDDD pe w=0.22u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M22  N$7 MQI VDDD VDDD pe w=0.22u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M21  N$6 MQI N$11 VSSD ne w=0.22u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M20  MQIB RN VDDD VDDD pe w=0.44u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M19  N$11 RN VSSD VSSD ne w=0.54u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M18  MQI MQIB VSSD VSSD ne w=0.26u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M17  MQI MQIB VDDD VDDD pe w=0.44u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
```

```
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M16  SQIB CI N$18 VSSD ne w=0.26u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M15  SQIB CIB N$20 VDDD pe w=0.48u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M14  N$20 MQI VDDD VDDD pe w=0.48u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M13  N$18 MQI N$21 VSSD ne w=0.26u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M12  SQIB CIB N$22 VSSD ne w=0.22u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M11  SQIB CI N$23 VDDD pe w=0.33u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M10  N$23 SQI VDDD VDDD pe w=0.33u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M9  N$22 SQI N$21 VSSD ne w=0.22u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
.ends DFRRX1
*
* component pathname : $MASTERS/da/LOGIC/OOR3X1
*
.subckt OOR3X1  OUT I1 I2 I3 VDDD VSSD

        X_M1  N$7 I1 VSSD VSSD ne w=0.335u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M4  N$7 I1 N$3 VDDD pe w=0.84u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M3  N$7 I3 VSSD VSSD ne w=0.335u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M2  N$7 I2 VSSD VSSD ne w=0.335u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M5  N$3 I2 N$1 VDDD pe w=0.84u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M6  N$1 I3 VDDD VDDD pe w=0.84u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M8  OUT N$7 VDDD VDDD pe w=0.99u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M7  OUT N$7 VSSD VSSD ne w=0.66u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
.ends OOR3X1
*
* component pathname : $MASTERS/da/LOGIC/AND2X1
*
.subckt AND2X1  OUT I1 I2 VDDD VSSD

        X_M4  N$4 I2 VDDD VDDD pe w=0.66u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M2  N$4 I1 N$3 VSSD ne w=0.66u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M3  N$4 I1 VDDD VDDD pe w=0.66u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M1  N$3 I2 VSSD VSSD ne w=0.66u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M5  OUT N$4 VSSD VSSD ne w=0.66u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M6  OUT N$4 VDDD VDDD pe w=0.99u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
.ends AND2X1
*
* component pathname : $MASTERS/da/LOGIC/INVX1
*
.subckt INVX1  OUT IN VDDD VSSD
```

```
        X_M2  OUT IN VDDD VDDD pe w=1.62u l=0.185u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M1  OUT IN VSSD VSSD ne w=0.66u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
.ends INVX1
*
* component pathname : $MASTERS/da/JTAG/TDEC
*
.subckt TDEC  Q0 Q1 IN S VDDD VSSD

        X_I$1  Q1 S IN VDDD VSSD AND2X1
        X_I$2  Q0 N$7 IN VDDD VSSD AND2X1
        X_I$3  N$7 S VDDD VSSD INVX1
.ends TDEC
*
* component pathname : $MASTERS/da/LOGIC/OOR2X1
*
.subckt OOR2X1  OUT I1 I2 VDDD VSSD

        X_M4  N$1 I2 VDDD VDDD pe w=0.7u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M3  N$4 I1 N$1 VDDD pe w=0.7u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M2  N$4 I2 VSSD VSSD ne w=0.33u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M1  N$4 I1 VSSD VSSD ne w=0.33u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M6  OUT N$4 VDDD VDDD pe w=1u l=0.184u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M5  OUT N$4 VSSD VSSD ne w=0.66u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
.ends OOR2X1
*
* component pathname : $MASTERS/da/JTAG/TAP_Controller_StateMachine
*
.subckt TAP_CONTROLLER_STATEMACHINE  CAPTURE_DR CAPTURE_IR EXIT1_DR EXIT1_IR SELECT_DR_SCAN
+ SELECT_IR_SCAN SHIFT_DR SHIFT_IR UPDATE_DR UPDATE_IR N_RST TCK TMS VDDD VSSD

        XDFRRSX11  TEST_LOGIC-RESET N$1346 TCK N$1354 VDDD N_RST VDDD VSSD DFRRSX1
        XOOR4X11  N$1355 N$22 N$1056 N$1305 N$1342 VDDD VSSD OOR4X1
        XDFRRX19  EXIT1_DR N$73 TCK N$1362 N_RST VDDD VSSD DFRRX1
        XDFRRX115  RUN_TEST-IDLE N$15 TCK N$1355 N_RST VDDD VSSD DFRRX1
        XDFRRX114  UPDATE_DR N$111 TCK N$1368 N_RST VDDD VSSD DFRRX1
        XDFRRX113  SELECT_DR_SCAN N$25 TCK N$1391 N_RST VDDD VSSD DFRRX1
        XDFRRX112  SELECT_IR_SCAN N$36 TCK N$1390 N_RST VDDD VSSD DFRRX1
        XDFRRX111  CAPTURE_DR N$49 TCK N$1359 N_RST VDDD VSSD DFRRX1
        XDFRRX18  PAUSE_DR N$86 TCK N$1364 N_RST VDDD VSSD DFRRX1
        XDFRRX110  SHIFT_DR N$58 TCK N$1361 N_RST VDDD VSSD DFRRX1
        XDFRRX17  EXIT2_DR N$102 TCK N$1386 N_RST VDDD VSSD DFRRX1
        XDFRRX16  CAPTURE_IR N$183 TCK N$1394 N_RST VDDD VSSD DFRRX1
        XDFRRX15  SHIFT_IR N$186 TCK N$1370 N_RST VDDD VSSD DFRRX1
        XDFRRX11  UPDATE_IR N$206 TCK N$1400 N_RST VDDD VSSD DFRRX1
        XDFRRX14  EXIT1_IR N$191 TCK N$1404 N_RST VDDD VSSD DFRRX1
        XDFRRX13  PAUSE_IR N$197 TCK N$1374 N_RST VDDD VSSD DFRRX1
        XDFRRX12  EXIT2_IR N$202 TCK N$1375 N_RST VDDD VSSD DFRRX1
        XOOR3X12#3  N$1426#3 VSSD VSSD VSSD VDDD VSSD OOR3X1
        XOOR3X12#2  N$1426#2 VSSD VSSD VSSD VDDD VSSD OOR3X1
        XOOR3X12#1  N$1426#1 VSSD VSSD VSSD VDDD VSSD OOR3X1
        X_I$678  N$1333 N$1334 EXIT2_IR TMS VDDD VSSD TDEC
        XTDEC1  N$1056 N$9 TEST_LOGIC-RESET TMS VDDD VSSD TDEC
        XOOR3X11  N$1391 N$1343 N$1306 N$130 VDDD VSSD OOR3X1
        X_I$669  N$1276 N$1277 CAPTURE_DR TMS VDDD VSSD TDEC
        X_I$463  N$22 N$130 RUN_TEST-IDLE TMS VDDD VSSD TDEC
        X_I$464  N$1359 N$1390 SELECT_DR_SCAN TMS VDDD VSSD TDEC
        X_I$18  N$1361 N$1300 N$1282 N$1276 VDDD VSSD OOR3X1
        X_I$668  N$1394 N$1271 SELECT_IR_SCAN TMS VDDD VSSD TDEC
        X_I$670  N$1282 N$1283 SHIFT_DR TMS VDDD VSSD TDEC
        X_I$671  N$1288 N$1289 EXIT1_DR TMS VDDD VSSD TDEC
        X_I$672  N$1294 N$1386 PAUSE_DR TMS VDDD VSSD TDEC
        X_I$673  N$1311 N$1344 CAPTURE_IR TMS VDDD VSSD TDEC
        X_I$679  N$1342 N$1343 UPDATE_IR TMS VDDD VSSD TDEC
        X_I$667  N$1300 N$1301 EXIT2_DR TMS VDDD VSSD TDEC
        X_I$674  N$1305 N$1306 UPDATE_DR TMS VDDD VSSD TDEC
        X_I$36  N$1370 N$1333 N$1316 N$1311 VDDD VSSD OOR3X1
```

```
        X_I$675  N$1316 N$1317 SHIFT_IR TMS VDDD VSSD TDEC
        X_I$676  N$1322 N$1323 EXIT1_IR TMS VDDD VSSD TDEC
        X_I$677  N$1327 N$1375 PAUSE_IR TMS VDDD VSSD TDEC
        XOOR2X18#5  N$1425#5 VSSD VSSD VDDD VSSD OOR2X1
        XOOR2X18#4  N$1425#4 VSSD VSSD VDDD VSSD OOR2X1
        XOOR2X18#3  N$1425#3 VSSD VSSD VDDD VSSD OOR2X1
        XOOR2X18#2  N$1425#2 VSSD VSSD VDDD VSSD OOR2X1
        XOOR2X18#1  N$1425#1 VSSD VSSD VDDD VSSD OOR2X1
        XOOR2X15  N$1362 N$1283 N$1277 VDDD VSSD OOR2X1
        XOOR2X17  N$1368 N$1289 N$1301 VDDD VSSD OOR2X1
        XOOR2X16  N$1354 N$9 N$1271 VDDD VSSD OOR2X1
        XOOR2X14  N$1364 N$1288 N$1294 VDDD VSSD OOR2X1
        XOOR2X11  N$1400 N$1334 N$1323 VDDD VSSD OOR2X1
        XOOR2X12  N$1374 N$1322 N$1327 VDDD VSSD OOR2X1
        XOOR2X13  N$1404 N$1317 N$1344 VDDD VSSD OOR2X1
        XAND2X11  N$1420 VSSD VSSD VDDD VSSD AND2X1
        XINVX11#7  N$1416#7 VSSD VDDD VSSD INVX1
        XINVX11#6  N$1416#6 VSSD VDDD VSSD INVX1
        XINVX11#5  N$1416#5 VSSD VDDD VSSD INVX1
        XINVX11#4  N$1416#4 VSSD VDDD VSSD INVX1
        XINVX11#3  N$1416#3 VSSD VDDD VSSD INVX1
        XINVX11#2  N$1416#2 VSSD VDDD VSSD INVX1
        XINVX11#1  N$1416#1 VSSD VDDD VSSD INVX1
.ends TAP_CONTROLLER_STATEMACHINE
*
* component pathname : $MASTERS/da/LOGIC/JKFF_1V8_RR/JK_LOGIC/A22NO_1V8_2
*
.subckt A22NO_1V8_2  OUT A B C D VDDD VDDD_BULK VSSD VSSD_BULK

        X_M8  N$223 B VSSD VSSD_BULK ne w=0.78u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u
+  pd=-4.96u nrs=-0.135 nrd=-0.135 par1=1
        X_M7  N$217 A VDDD VDDD_BULK pe w=1.45u l=0.184u m=1 as=-0.96p ad=-0.96p ps=-4.96u
+  pd=-4.96u nrs=-0.135 nrd=-0.135 par1=1
        X_M6  OUT A N$223 VSSD_BULK ne w=0.78u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u
+  pd=-4.96u nrs=-0.135 nrd=-0.135 par1=1
        X_M5  OUT C N$217 VDDD_BULK pe w=1.45u l=0.184u m=1 as=-0.96p ad=-0.96p ps=-4.96u
+  pd=-4.96u nrs=-0.135 nrd=-0.135 par1=1
        X_M4  N$225 D VSSD VSSD_BULK ne w=0.78u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u
+  pd=-4.96u nrs=-0.135 nrd=-0.135 par1=1
        X_M3  OUT C N$225 VSSD_BULK ne w=0.78u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u
+  pd=-4.96u nrs=-0.135 nrd=-0.135 par1=1
        X_M2  N$217 B VDDD VDDD_BULK pe w=1.45u l=0.184u m=1 as=-0.96p ad=-0.96p ps=-4.96u
+  pd=-4.96u nrs=-0.135 nrd=-0.135 par1=1
        X_M1  OUT D N$217 VDDD_BULK pe w=1.45u l=0.184u m=1 as=-0.96p ad=-0.96p ps=-4.96u
+  pd=-4.96u nrs=-0.135 nrd=-0.135 par1=1
.ends A22NO_1V8_2
*
* component pathname : $MASTERS/da/LOGIC/JKFF_1V8_RR
*
.subckt JKFF_1V8_RR  Q QN C J K RN VDDD VSSD

        XA22NO_1V8_21  D SQI K N$960 SQIB VDDD VDDD VSSD VSSD A22NO_1V8_2
        X_M16  SQIB CI N$729 VSSD ne w=0.26u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M27  MQIB CIB N$715 VSSD ne w=0.22u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M28  MQIB CI N$718 VDDD pe w=0.49u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M26  N$718 D VDDD VDDD pe w=0.7u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M25  N$715 D N$719 VSSD ne w=0.35u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M24  MQIB CI N$721 VSSD ne w=0.22u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M23  MQIB CIB N$723 VDDD pe w=0.22u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M22  N$723 MQI VDDD VDDD pe w=0.22u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M21  N$721 MQI N$719 VSSD ne w=0.22u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u
```

```
+  pd=-4.96u nrs=-0.135 nrd=-0.135 par1=1
        X_M20  MQIB RN VDDD VDDD pe w=0.44u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M19  N$719 RN VSSD VSSD ne w=0.54u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M18  MQI MQIB VSSD VSSD ne w=0.26u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M17  MQI MQIB VDDD VDDD pe w=0.44u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M8  SQIB RN VDDD VDDD pe w=0.72u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M7  N$733 RN VSSD VSSD ne w=0.4u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M6  SQI SQIB VSSD VSSD ne w=0.22u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M5  SQI SQIB VDDD VDDD pe w=0.33u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M34  N$960 J VSSD VSSD ne w=0.22u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M3  Q SQIB VDDD VDDD pe w=0.99u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M2  QN SQI VSSD VSSD ne w=0.66u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M1  QN SQI VDDD VDDD pe w=0.99u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M32  CIB C VSSD VSSD ne w=0.66u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M31  CIB C VDDD VDDD pe w=0.99u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M29  CI CIB VSSD VSSD ne w=0.66u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M30  CI CIB VDDD VDDD pe w=0.99u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M33  N$960 J VDDD VDDD pe w=0.62u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M10  N$737 SQI VDDD VDDD pe w=0.33u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M15  SQIB CIB N$732 VDDD pe w=0.48u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M14  N$732 MQI VDDD VDDD pe w=0.48u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M4  Q SQIB VSSD VSSD ne w=0.66u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M9  N$735 SQI N$733 VSSD ne w=0.22u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M12  SQIB CIB N$735 VSSD ne w=0.22u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M13  N$729 MQI N$733 VSSD ne w=0.26u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u
+  pd=-4.96u nrs=-0.135 nrd=-0.135 par1=1
        X_M11  SQIB CI N$737 VDDD pe w=0.33u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
.ends JKFF_1V8_RR
*
* component pathname : $MASTERS/da/LOGIC/AND3X1
*
.subckt AND3X1  OUT I1 I2 I3 VDDD VSSD

        X_M1  N$1 I3 VSSD VSSD ne w=0.66u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
```

```
        X_M5  N$7 I2 VDDD VDDD pe w=0.66u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M2  N$5 I2 N$1 VSSD ne w=0.66u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M4  N$7 I1 VDDD VDDD pe w=0.66u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M7  OUT N$7 VSSD VSSD ne w=0.66u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M8  OUT N$7 VDDD VDDD pe w=0.99u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M3  N$7 I1 N$5 VSSD ne w=0.66u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M6  N$7 I3 VDDD VDDD pe w=0.66u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
.ends AND3X1
*
* component pathname : $MASTERS/da/LOGIC/BUX4
*
.subckt BUX4  OUT IN VDDD VSSD

        X_M2  N$1 IN VDDD VDDD pe w=3.2u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M1  N$1 IN VSSD VSSD ne w=1.5u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M4  OUT N$1 VDDD VDDD pe w=6.4u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M3  OUT N$1 VSSD VSSD ne w=2.64u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
.ends BUX4
*
* component pathname : $MASTERS/da/LOGIC/BUX2
*
.subckt BUX2  OUT IN VDDD VSSD

        X_M2  N$1 IN VDDD VDDD pe w=1.6u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M1  N$1 IN VSSD VSSD ne w=0.75u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M4  OUT N$1 VDDD VDDD pe w=3.2u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M3  OUT N$1 VSSD VSSD ne w=1.32u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
.ends BUX2
*
* component pathname : $MASTERS/da/LOGIC/BUX1
*
.subckt BUX1  OUT IN VDDD VSSD

        X_M2  N$1 IN VDDD VDDD pe w=0.8u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M1  N$1 IN VSSD VSSD ne w=0.38u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M4  OUT N$1 VDDD VDDD pe w=1.6u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M3  OUT N$1 VSSD VSSD ne w=0.66u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
.ends BUX1
*
* component pathname : $MASTERS/da/LOGIC/INVX4
*
.subckt INVX4  OUT IN VDDD VSSD

        X_M2  OUT IN VDDD VDDD pe w=6.4u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M1  OUT IN VSSD VSSD ne w=2.64u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
.ends INVX4
*
* component pathname : $MASTERS/da/JTAG/TAP_Controller
*
.subckt TAP_CONTROLLER_V2  CAPTURE_IR CLK_DR_BSR CLK_DR_BYPASS ENABLE_DR_BSR ENABLE_DR_BYPASS
```

```
+ ENABLE_IR MODE_BSR SHIFT_DR_BSR SHIFT_DR_BYPASS SHIFT_IR SHIFT_IR_CK UPDATE_DR_BSR
UPDATE_IR_CK
+ BYPASS EXTEST N_RST NOT_USED SAMPLE-PRELOAD TCK TMS VDDD VSSD

        XTAP_CONTROLLER_STATEMACHINE1  N$136 CAPTURE_IR_INT N$115 EXIT1_IR SELECT_DR_SCAN
+  SELECT_IR_SCAN N$126 SHIFT_IR N$131 UPDATE_IR N_RST TCK TMS VDDD VSSD
TAP_CONTROLLER_STATEMACHINE
        XJKFF_1V8_RR3  ENABLE_IR N$27 N_TCK_INT SHIFT_IR EXIT1_IR N_RST VDDD VSSD JKFF_1V8_RR
        XJKFF_1V8_RR2  ENABLE_DR N$39 TCK_INT CAPTURE_DR EXIT1_DR N_RST VDDD VSSD JKFF_1V8_RR
        XJKFF_1V8_RR1  MODE_BSR N$52 TCK_INT EXTEST SAMPLE-PRELOAD N_RST VDDD VSSD
JKFF_1V8_RR
        X_I$25  CAPTURE_IR N$106 TCK_INT CAPTURE_IR_INT N_RST VDDD VSSD DFRRX1
        XAND3X11  CLK_DR_BSR OR_SHIFT_CAPTURE OR_EXTEST_SAMPLE TCK_INT VDDD VSSD AND3X1
        XAND2X16  UPDATE_IR_CK N_TCK_INT UPDATE_IR VDDD VSSD AND2X1
        XAND2X15  SHIFT_DR_BYPASS N$43 SHIFT_DR VDDD VSSD AND2X1
        XAND2X14  CLK_DR_BYPASS N$43 TCK_INT VDDD VSSD AND2X1
        XAND2X13  ENABLE_DR_BYPASS N$43 ENABLE_DR VDDD VSSD AND2X1
        XOOR2X12  OR_EXTEST_SAMPLE EXTEST SAMPLE-PRELOAD VDDD VSSD OOR2X1
        XAND2X12  SHIFT_DR_BSR OR_EXTEST_SAMPLE SHIFT_DR VDDD VSSD AND2X1
        XAND2X18  UPDATE_DR_BSR OR_EXTEST_SAMPLE UPDATE_DR VDDD VSSD AND2X1
        XAND2X17  SHIFT_IR_CK SHIFT_IR N_TCK_INT VDDD VSSD AND2X1
        XAND2X11  ENABLE_DR_BSR OR_EXTEST_SAMPLE ENABLE_DR VDDD VSSD AND2X1
        XOOR2X11  OR_SHIFT_CAPTURE SHIFT_DR CAPTURE_DR VDDD VSSD OOR2X1
        XOOR2X13  N$43 BYPASS NOT_USED VDDD VSSD OOR2X1
        XBUX41  TCK_INT TCK VDDD VSSD BUX4
        XBUX21  SHIFT_DR N$126 VDDD VSSD BUX2
        XBUX11  UPDATE_DR N$131 VDDD VSSD BUX1
        XBUX13  CAPTURE_DR N$136 VDDD VSSD BUX1
        XBUX12  EXIT1_DR N$115 VDDD VSSD BUX1
        XINVX41  N_TCK_INT TCK VDDD VSSD INVX4
.ends TAP_CONTROLLER
*
* component pathname : $MASTERS/da/LOGIC/DFRSX1
*
.subckt DFRSX1  Q QN C D SN VDDD VSSD

        X_M23  NETZ193 D VDDD VDDD pe w=0.700000u l=0.180000u m=1 as=3.36e-13 ad=3.36e-13
+  ps=2.36u pd=2.36u nrs=0.385714 nrd=0.385714 par1=1
        X_M5  QN SQIB VDDD VDDD pe w=0.99u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M49  SQIB SQI NETZ241 VSSD ne w=0.220000u l=0.180000u m=1 as=1.056e-13 ad=1.056e-13
+  ps=1.4u pd=1.4u nrs=1.22727 nrd=1.22727 par1=1
        X_M61  NETZ213 MQI VSSD VSSD ne w=0.220000u l=0.180000u m=1 as=1.056e-13 ad=1.056e-13
+  ps=1.4u pd=1.4u nrs=1.22727 nrd=1.22727 par1=1
        X_M30  NETZ147 MQI VDDD VDDD pe w=0.480000u l=0.180000u m=1 as=2.304e-13 ad=2.304e-13
+  ps=1.92u pd=1.92u nrs=0.5625 nrd=0.5625 par1=1
        X_M4  CIB C VSSD VSSD ne w=0.66u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M3  CIB C VDDD VDDD pe w=0.99u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M2  CI CIB VSSD VSSD ne w=0.66u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M1  CI CIB VDDD VDDD pe w=0.99u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M47  NETZ221 MQI VSSD VSSD ne w=0.260000u l=0.180000u m=1 as=1.248e-13 ad=1.248e-13
+  ps=1.48u pd=1.48u nrs=1.03846 nrd=1.03846 par1=1
        X_M8  QN SQIB VSSD VSSD ne w=0.66u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M35  SQI CI NETZ155 VDDD pe w=0.330000u l=0.180000u m=1 as=1.584e-13 ad=1.584e-13
+  ps=1.62u pd=1.62u nrs=0.818182 nrd=0.818182 par1=1
        X_M46  SQI CI NETZ221 VSSD ne w=0.260000u l=0.180000u m=1 as=1.248e-13 ad=1.248e-13
+  ps=1.48u pd=1.48u nrs=1.03846 nrd=1.03846 par1=1
        X_M50  NETZ233 SQIB VSSD VSSD ne w=0.220000u l=0.180000u m=1 as=1.056e-13 ad=1.056e-
13
+  ps=1.4u pd=1.4u nrs=1.22727 nrd=1.22727 par1=1
        X_M17  MQI MQIB NETZ209 VSSD ne w=0.260000u l=0.180000u m=1 as=1.248e-13 ad=1.248e-13
+  ps=1.48u pd=1.48u nrs=1.03846 nrd=1.03846 par1=1
        X_M34  MQIB CIB NETZ151 VDDD pe w=0.220000u l=0.180000u m=1 as=1.056e-13 ad=1.056e-13
+  ps=1.4u pd=1.4u nrs=1.22727 nrd=1.22727 par1=1
        X_M51  SQI CIB NETZ233 VSSD ne w=0.220000u l=0.180000u m=1 as=1.056e-13 ad=1.056e-13
+  ps=1.4u pd=1.4u nrs=1.22727 nrd=1.22727 par1=1
        X_M18  MQIB CIB NETZ205 VSSD ne w=0.220000u l=0.180000u m=1 as=1.056e-13 ad=1.056e-13
+  ps=1.4u pd=1.4u nrs=1.22727 nrd=1.22727 par1=1
        X_M7  Q SQI VDDD VDDD pe w=0.99u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
```

```
        X_M6  Q SQI VSSD VSSD ne w=0.66u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M36  NETZ155 SQIB VDDD VDDD pe w=0.330000u l=0.180000u m=1 as=1.584e-13 ad=1.584e-
13
+  ps=1.62u pd=1.62u nrs=0.818182 nrd=0.818182 par1=1
        X_M24  MQIB CI NETZ193 VDDD pe w=0.490000u l=0.180000u m=1 as=2.352e-13 ad=2.352e-13
+  ps=1.94u pd=1.94u nrs=0.55102 nrd=0.55102 par1=1
        X_M60  SQIB SN VDDD VDDD pe w=0.420000u l=0.180000u m=1 as=2.016e-13 ad=2.016e-13
+  ps=1.8u pd=1.8u nrs=0.642857 nrd=0.642857 par1=1
        X_M57  MQI SN VDDD VDDD pe w=0.500000u l=0.180000u m=1 as=2.4e-13 ad=2.4e-13 ps=1.96u
+  pd=1.96u nrs=0.54 nrd=0.54 par1=1
        X_M15  NETZ205 D VSSD VSSD ne w=0.350000u l=0.180000u m=1 as=1.68e-13 ad=1.68e-13
+  ps=1.66u pd=1.66u nrs=0.771429 nrd=0.771429 par1=1
        X_M59  SQIB SQI VDDD VDDD pe w=0.330000u l=0.180000u m=1 as=1.584e-13 ad=1.584e-13
+  ps=1.62u pd=1.62u nrs=0.818182 nrd=0.818182 par1=1
        X_M56  MQI MQIB VDDD VDDD pe w=0.440000u l=0.180000u m=1 as=2.112e-13 ad=2.112e-13
+  ps=1.84u pd=1.84u nrs=0.613636 nrd=0.613636 par1=1
        X_M19  MQIB CI NETZ213 VSSD ne w=0.220000u l=0.180000u m=1 as=1.056e-13 ad=1.056e-13
+  ps=1.4u pd=1.4u nrs=1.22727 nrd=1.22727 par1=1
        X_M20  NETZ209 SN VSSD VSSD ne w=0.260000u l=0.180000u m=1 as=1.248e-13 ad=1.248e-13
+  ps=1.48u pd=1.48u nrs=1.03846 nrd=1.03846 par1=1
        X_M28  NETZ151 MQI VDDD VDDD pe w=0.220000u l=0.180000u m=1 as=1.056e-13 ad=1.056e-13
+  ps=1.4u pd=1.4u nrs=1.22727 nrd=1.22727 par1=1
        X_M33  SQI CIB NETZ147 VDDD pe w=0.480000u l=0.180000u m=1 as=2.304e-13 ad=2.304e-13
+  ps=1.92u pd=1.92u nrs=0.5625 nrd=0.5625 par1=1
        X_M52  NETZ241 SN VSSD VSSD ne w=0.220000u l=0.180000u m=1 as=1.056e-13 ad=1.056e-13
+  ps=1.4u pd=1.4u nrs=1.22727 nrd=1.22727 par1=1
.ends DFRSX1
*
* component pathname : $MASTERS/da/LOGIC/DFFRX1
*
.subckt DFFRX1  Q QN CN D RN VDDD VSSD

        X_M29  CIB CI VSSD VSSD ne w=0.66u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M28  MQIB CI N$4 VDDD pe w=0.49u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M27  MQIB CIB N$2 VSSD ne w=0.22u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M26  N$4 D VDDD VDDD pe w=0.7u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M25  N$2 D N$11 VSSD ne w=0.35u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M24  MQIB CI N$6 VSSD ne w=0.22u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M23  MQIB CIB N$7 VDDD pe w=0.22u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M22  N$7 MQI VDDD VDDD pe w=0.22u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M21  N$6 MQI N$11 VSSD ne w=0.22u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M20  MQIB RN VDDD VDDD pe w=0.44u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M19  N$11 RN VSSD VSSD ne w=0.54u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M18  MQI MQIB VSSD VSSD ne w=0.26u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M17  MQI MQIB VDDD VDDD pe w=0.44u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M16  SQIB CI N$18 VSSD ne w=0.26u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M15  SQIB CIB N$20 VDDD pe w=0.48u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
```

```
        X_M14  N$20 MQI VDDD VDDD pe w=0.48u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M13  N$18 MQI N$21 VSSD ne w=0.26u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M12  SQIB CIB N$22 VSSD ne w=0.22u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M11  SQIB CI N$23 VDDD pe w=0.33u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M10  N$23 SQI VDDD VDDD pe w=0.33u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M9  N$22 SQI N$21 VSSD ne w=0.22u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M8  SQIB RN VDDD VDDD pe w=0.72u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M7  N$21 RN VSSD VSSD ne w=0.4u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M6  SQI SQIB VSSD VSSD ne w=0.22u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M5  SQI SQIB VDDD VDDD pe w=0.33u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M4  Q SQIB VSSD VSSD ne w=0.66u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M3  Q SQIB VDDD VDDD pe w=0.99u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M2  QN SQI VSSD VSSD ne w=0.66u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M1  QN SQI VDDD VDDD pe w=0.99u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M32  CI CN VSSD VSSD ne w=0.66u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M31  CI CN VDDD VDDD pe w=0.99u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M30  CIB CI VDDD VDDD pe w=0.99u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
.ends DFFRX1
*
* component pathname : $MASTERS/da/LOGIC/NO2X1
*
.subckt NO2X1  OUT I1 I2 VDDD VSSD

        X_M4  N$2 I2 VDDD VDDD pe w=1.4u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M3  OUT I1 N$2 VDDD pe w=1.4u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M2  OUT I2 VSSD VSSD ne w=0.66u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M1  OUT I1 VSSD VSSD ne w=0.66u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
.ends NO2X1
*
* component pathname : $MASTERS/da/JTAG/Instruction_register
*
.subckt INSTRUCTION_REGISTER  BYPASS EXTEST NOT_USED SAMPLE-PRELOAD TDO_INT CAPTURE_IR N_RST
+ SHIFT_IR SHIFT_IR_CK TDI_INT UPDATE_IR VDDD VSSD

        XDFRRSX12  INST_REG_MIDLE N$15 SHIFT_IR_CK TDI_INT N$2743 VDDD VDDD VSSD DFRRSX1
        XDFRRSX11  INST_REG_END N$2127 SHIFT_IR_CK INST_REG_MIDLE N_RST N_CAPTURE_IR VDDD
+  VSSD DFRRSX1
        XDFRSX12  INST_BIT0 N$18 UPDATE_IR HOLD2 N_RST VDDD VSSD DFRSX1
        XDFRSX11  INST_BIT1 N$17 UPDATE_IR HOLD1 N_RST VDDD VSSD DFRSX1
        XDFFRX12  HOLD1 N$1698 SHIFT_IR INST_REG_MIDLE N_RST VDDD VSSD DFFRX1
        XDFFRX11  HOLD2 N$1700 SHIFT_IR INST_REG_END N_RST VDDD VSSD DFFRX1
        XNO2X13  EXTEST INST_BIT1 INST_BIT0 VDDD VSSD NO2X1
        XAND2X12  BYPASS INST_BIT1 INST_BIT0 VDDD VSSD AND2X1
        XNO2X12  SAMPLE-PRELOAD INST_BIT1 N$33 VDDD VSSD NO2X1
        XNO2X11  NOT_USED N$42 INST_BIT0 VDDD VSSD NO2X1
        XAND2X11  N$2743 N_CAPTURE_IR N_RST VDDD VSSD AND2X1
```

```
        XINVX13  N_CAPTURE_IR CAPTURE_IR VDDD VSSD INVX1
        XBUX11  TDO_INT INST_REG_END VDDD VSSD BUX1
        XINVX12  N$33 INST_BIT0 VDDD VSSD INVX1
        XINVX11  N$42 INST_BIT1 VDDD VSSD INVX1
.ends INSTRUCTION_REGISTER
*
* component pathname : $MASTERS/da/LOGIC/MU2X1
*
.subckt MU2X1  OUT I0 I1 S VDDD VSSD

        X_M8  N$1 N$21 N$3 VSSD ne w=0.4u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M9  N$1 S N$6 VDDD pe w=0.7u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M10  N$6 I0 VDDD VDDD pe w=0.7u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M7  N$3 I0 VSSD VSSD ne w=0.4u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M12  OUT N$1 VDDD VDDD pe w=1u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M1  N$21 S VSSD VSSD ne w=0.33u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M4  N$1 S N$23 VSSD ne w=0.4u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M5  N$1 N$21 N$24 VDDD pe w=0.7u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M6  N$24 I1 VDDD VDDD pe w=0.7u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M3  N$23 I1 VSSD VSSD ne w=0.4u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M2  N$21 S VDDD VDDD pe w=0.5u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M11  OUT N$1 VSSD VSSD ne w=0.66u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
.ends MU2X1
*
* component pathname : $MASTERS/da/JTAG/Boundary_scan_cell
*
.subckt BOUNDARY_SCAN_CELL  DATA_OUT SCAN_OUT CLOCKDR DATA_IN MODE N_RST SCAN_IN SHIFTDR
+ UPDATEDR VDDD VSSD

        XDFFRX11  SCAN_OUT N$14 CLOCKDR N$13 N_RST VDDD VSSD DFFRX1
        XDFRRX11  N$3 N$15 UPDATEDR SCAN_OUT N_RST VDDD VSSD DFRRX1
        XMU2X12  N$13 DATA_IN SCAN_IN SHIFTDR VDDD VSSD MU2X1
        XMU2X11  DATA_OUT DATA_IN N$3 MODE VDDD VSSD MU2X1
.ends BOUNDARY_SCAN_CELL
*
* component pathname : $MASTERS/da/JTAG/Bypass_register
*
.subckt BYPASS_REGISTER  TDO_INT CLOCKDR N_RST SHIFTDR TDI_INT VDDD VSSD

        XDFRRX11  TDO_INT N$10 CLOCKDR N$9 N_RST VDDD VSSD DFRRX1
        XMU2X11  N$9 VSSD TDI_INT SHIFTDR VDDD VSSD MU2X1
.ends BYPASS_REGISTER
*
* component pathname : $MASTERS/da/LOGIC/BTHX1
*
.subckt BTHX1  OUT EN IN VDDD VSSD

        X_M10  N$10 EN VSSD VSSD ne w=0.33u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M9  N$10 EN VDDD VDDD pe w=0.5u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M8  N$11 IN VDDD VDDD pe w=0.65u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M7  N$11 EN VDDD VDDD pe w=0.5u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M6  OUT N$11 VDDD VDDD pe w=1.4u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M5  N$13 N$10 N$11 VDDD pe w=0.5u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
```

```
        X_M4  N$11 EN N$13 VSSD ne w=0.33u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M3  N$13 IN VSSD VSSD ne w=0.42u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M2  N$13 N$10 VSSD VSSD ne w=0.33u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
        X_M1  OUT N$13 VSSD VSSD ne w=0.8u l=0.18u m=1 as=-0.96p ad=-0.96p ps=-4.96u pd=-
4.96u
+  nrs=-0.135 nrd=-0.135 par1=1
.ends BTHX1
*
* MAIN CELL: component pathname : $MASTERS/da/JTAG/TEST_and_SIMULATIONS/Test_JTAG_circuit
*
        XTAP_CONTROLLER_V21  CAPTURE_IR CLOCK_DR_BSR CLOCK_DR_BYPASS ENABLE_DR_BSR
ENABLE_DR_BYPASS
+  ENABLE_IR MODE_DR_BSR SHIFT_DR_BSR SHIFT_DR_BYPASS SHIFT_IR SHIFT_IR_CLK UPDATE_DR_BSR
+  UPDATE_IR BYPASS EXTEST N_RST NOT_USED SAMPLE_PRELOAD TCK TMS VDDD VSSD TAP_CONTROLLER_V2
        XINSTRUCTION_REGISTER1  BYPASS EXTEST NOT_USED SAMPLE_PRELOAD N$90 CAPTURE_IR N_RST
+  SHIFT_IR SHIFT_IR_CLK N$115 UPDATE_IR VDDD VSSD INSTRUCTION_REGISTER
        XBOUNDARY_SCAN_CELL6  INPUT1_OUT N$5 CLOCK_DR_BSR INPUT1_IN MODE_DR_BSR N_RST N$126
+  SHIFT_DR_BSR UPDATE_DR_BSR VDDD VSSD BOUNDARY_SCAN_CELL
        XBOUNDARY_SCAN_CELL5  INPUT2_OUT N$6 CLOCK_DR_BSR INPUT2_IN MODE_DR_BSR N_RST N$5
+  SHIFT_DR_BSR UPDATE_DR_BSR VDDD VSSD BOUNDARY_SCAN_CELL
        XBOUNDARY_SCAN_CELL4  INPUT3_OUT N$7 CLOCK_DR_BSR INPUT3_IN MODE_DR_BSR N_RST N$6
+  SHIFT_DR_BSR UPDATE_DR_BSR VDDD VSSD BOUNDARY_SCAN_CELL
        XBOUNDARY_SCAN_CELL3  OUTPUT3_OUT N$8 CLOCK_DR_BSR OUTPUT3_IN MODE_DR_BSR N_RST N$7
+  SHIFT_DR_BSR UPDATE_DR_BSR VDDD VSSD BOUNDARY_SCAN_CELL
        XBOUNDARY_SCAN_CELL2  OUTPUT2_OUT N$9 CLOCK_DR_BSR OUTPUT2_IN MODE_DR_BSR N_RST N$8
+  SHIFT_DR_BSR UPDATE_DR_BSR VDDD VSSD BOUNDARY_SCAN_CELL
        XBOUNDARY_SCAN_CELL1  OUTPUT1_OUT N$89 CLOCK_DR_BSR OUTPUT1_IN MODE_DR_BSR N_RST
+  N$9 SHIFT_DR_BSR UPDATE_DR_BSR VDDD VSSD BOUNDARY_SCAN_CELL
        XBYPASS_REGISTER1  N$1 CLOCK_DR_BYPASS N_RST SHIFT_DR_BYPASS N$125 VDDD VSSD
BYPASS_REGISTER
        X_I$11  TDO ENABLE_DR_BSR N$89 VDDD VSSD BTHX1
        X_I$12  TDO ENABLE_DR_BYPASS N$1 VDDD VSSD BTHX1
        X_I$13  TDO ENABLE_IR N$90 VDDD VSSD BTHX1
        X_I$14  N$126 ENABLE_DR_BSR TDI VDDD VSSD BTHX1
        X_I$15  N$125 ENABLE_DR_BYPASS TDI VDDD VSSD BTHX1
        X_I$16  N$115 ENABLE_IR TDI VDDD VSSD BTHX1
        XBUX21  N_RST TRST VDDD VSSD BUX2
        XBUX25  TRST TRST_PAD VDDD VSSD BUX2
        XBUX24  TMS TMS_PAD VDDD VSSD BUX2
        XBUX23  TCK TCK_PAD VDDD VSSD BUX2
        XBUX22  TDI TDI_PAD VDDD VSSD BUX2
        X_R4  TCK VSSD rpp1k1 W=2u L=100u m=1 par1=1
        X_R3  VDDD TMS rpp1k1 W=2u L=100u m=1 par1=1
        X_R2  VDDD TRST rpp1k1 W=2u L=100u m=1 par1=1
        X_R1  VDDD TDI rpp1k1 W=2u L=100u m=1 par1=1
*
* eldo include file.
*
*
.end
```

4. Boundary-Scan Top Level Simulation file

```
* Component: $MASTERS/da/JTAG/TEST_and_SIMULATIONS/Test_JTAG_circuit  Viewpoint: eldonet

*** TEST JTAG TAP controller for Boundary-scan chain tests ***

** Netlist include
.INCLUDE Test_JTAG_circuit_eldonet.spi

** Device Models include
.LIB $MGC_DESIGN_KIT/eldo/xc018_R3.0/mosst/xc018.lib TM
.LIB $MGC_DESIGN_KIT/eldo/xc018_R3.0/mosst/param.lib 3s

**** SIMULATOR OPTIONS ****
.OPTION NOASCII
.OPTION MODWL
```

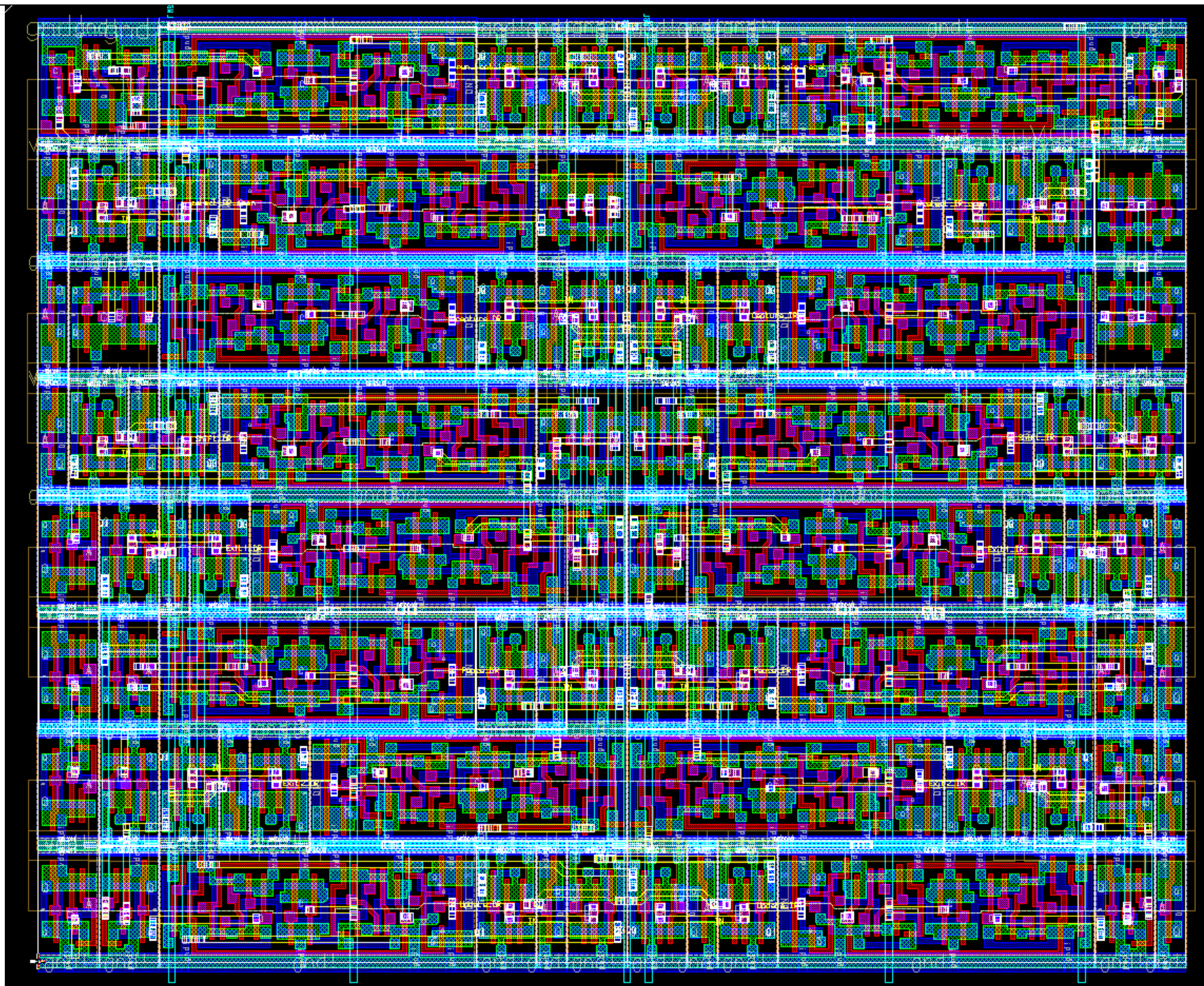
```
.OPTION SPICEDC
.OPTION AEX
.OPTION ENGNOT
.OPTION PROBOP2
.OP

**** OPTIONS ****
.temp 55
.option gmin = 10e-15

**** POWER SUPPLY ****
.param supply_voltage = 1.8
vpower vddd vssd dc supply_voltage

*** Type of Simulation ***
.tran 0 3u

*** Set input values for state Machine ***

vrst TRST_pad vssd pattern supply_voltage 0 0   0.5n 0.5n 30n 01
vclk TCK_pad  vssd pattern supply_voltage 0 0   0.5n 0.5n 15n 10 R
vtms TMS_pad  vssd pattern supply_voltage 0 25n 0.5n 0.5n 30n 111 01100 0011 1000 0000 0000
1111 1111 01100 0011 100 000 000 000 1000 1111 000 01 1100 0000 0000 0000 0000 11111
vtdi TDI_pad  vssd pattern supply_voltage 0 25n 0.5n 0.5n 30n 000 00000 1100 0000 0010 1011
0101 0101 00000 1000 001 101 010 101 0101 0101 010 00 0001 0101 0101 0101 0101 01010

**** Pull up resistors ****
rload TDO vssd 1MEG
rload2 TDO vddd 1MEG

*** Set input PADs values ***

vinput1_in input1_in vssd pattern supply_voltage 0 0 0.1n 0.1n 30n 10
vinput2_in input2_in vssd pattern supply_voltage 0 0 0.1n 0.1n 30n 10
vinput3_in input3_in vssd pattern supply_voltage 0 0 0.1n 0.1n 30n 10

voutput1_in output1_in vssd pattern supply_voltage 0 0 0.1n 0.1n 30n 01
voutput2_in output2_in vssd pattern supply_voltage 0 0 0.1n 0.1n 30n 01
voutput3_in output3_in vssd pattern supply_voltage 0 0 0.1n 0.1n 30n 01


*** Plot inputs ***
.plot tran v(n_RST) v(TCK) v(TMS) v(TDI) v(TDO)

*** Plot ututs and test points ***
.plot tran v(Extest) v(Not_Used) v(Bypass) v(Sample_Preload)

.PLOT TRAN V(CLOCK_DR_BSR) V(SHIFT_DR_BSR)
.PLOT TRAN V(EXTEST) V(SAMPLE_PRELOAD)
.PLOT TRAN V(XTAP_CONTROLLER_V21.OR_EXTEST_SAMPLE) V(XTAP_CONTROLLER_V21.OR_SHIFT_CAPTURE)
V(XTAP_CONTROLLER_V21.TCK_INT)
.PLOT TRAN V(SHIFT_DR_BSR) V(CLOCK_DR_BSR)

.PLOT TRAN V(XTAP_CONTROLLER_V21.SHIFT_DR) V(XTAP_CONTROLLER_V21.SHIFT_IR)
V(XTAP_CONTROLLER_V21.UPDATE_DR) V(XTAP_CONTROLLER_V21.UPDATE_IR)
V(XTAP_CONTROLLER_V21.CAPTURE_DR) V(XTAP_CONTROLLER_V21.SELECT_DR_SCAN)
V(XTAP_CONTROLLER_V21.SELECT_IR_SCAN)
.PLOT TRAN V(SHIFT_IR_CLK) V(UPDATE_IR) V(ENABLE_IR)
.PLOT TRAN V(SHIFT_DR_BYPASS) V(CLOCK_DR_BYPASS) V(ENABLE_DR_BYPASS)
.PLOT TRAN V(XTAP_CONTROLLER_V21.ENABLE_DR) V(XTAP_CONTROLLER_V21.SELECT_DR_SCAN)
V(XTAP_CONTROLLER_V21.UPDATE_DR)
.PLOT TRAN V(XTAP_CONTROLLER_V21.CAPTURE_IR)

.PLOT TRAN V(XINSTRUCTION_REGISTER1.INST_REG_MIDLE) V(XINSTRUCTION_REGISTER1.INST_REG_END)
V(XINSTRUCTION_REGISTER1.INST_BIT0) V(XINSTRUCTION_REGISTER1.INST_BIT1)
.PLOT TRAN V(CAPTURE_IR) V(SHIFT_IR)
.PLOT TRAN V(XINSTRUCTION_REGISTER1.INST_REG_MIDLE) V(XINSTRUCTION_REGISTER1.INST_REG_END)
V(XINSTRUCTION_REGISTER1.hold1) V(XINSTRUCTION_REGISTER1.hold2)
V(XINSTRUCTION_REGISTER1.INST_BIT1) V(XINSTRUCTION_REGISTER1.INST_BIT0)
.PLOT TRAN V(CLOCK_DR_BSR) V(MODE_DR_BSR) V(SHIFT_DR_BSR) V(UPDATE_DR_BSR) V(ENABLE_DR_BSR)
.PLOT TRAN V(INPUT3_OUT) V(INPUT2_OUT) V(INPUT1_OUT) V(OUTPUT3_OUT) V(OUTPUT2_OUT)
V(OUTPUT1_OUT)
```

Appendix 4 – Layout drawings from the Boundary Scan Chain Blocks

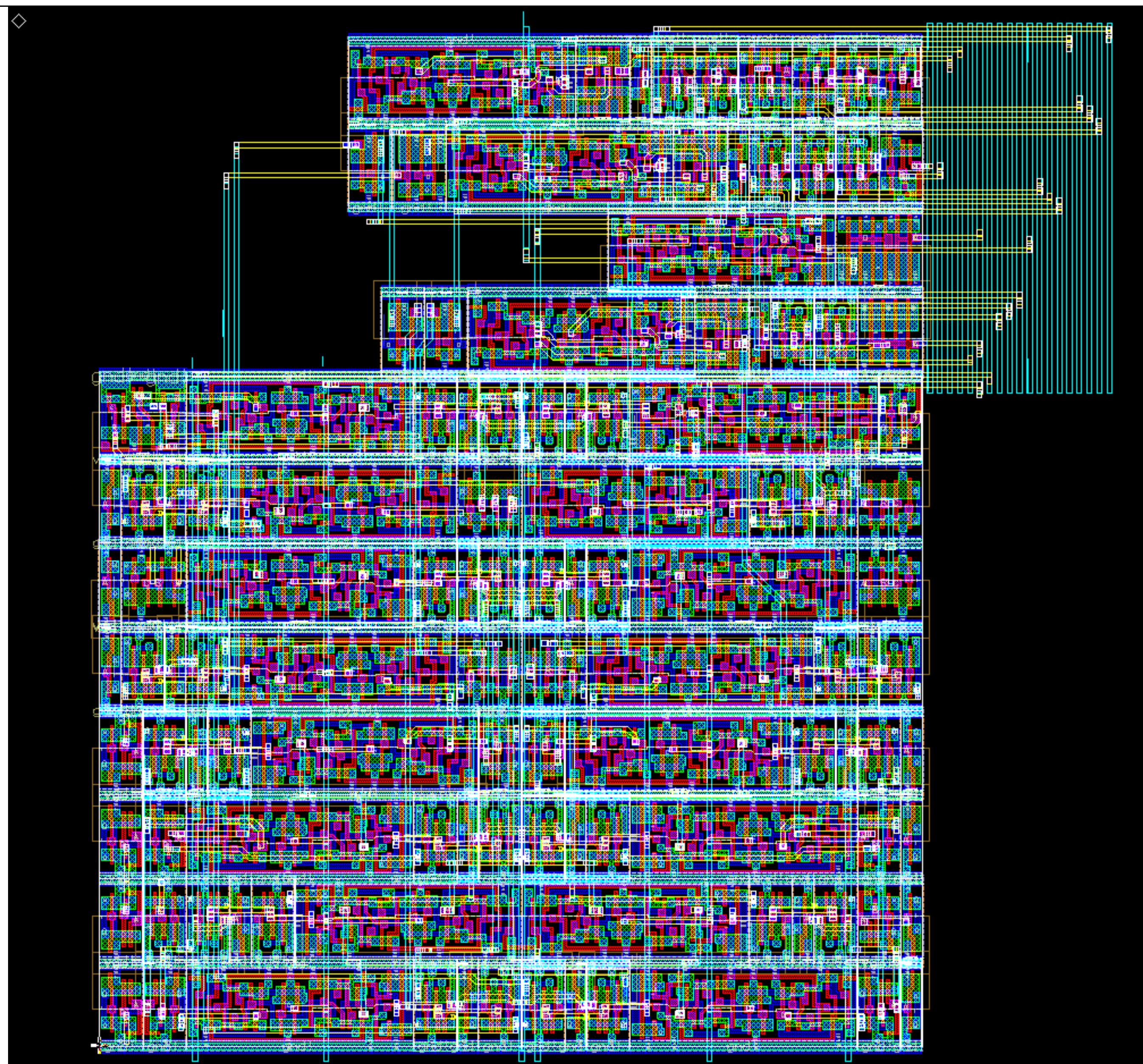
Table of contents

1. *TAP Controller State Machine Layout*--D2
2. *TAP Controller Block Layout*---D3
3. *TAP Controller Block Layout – Blocks overview* ---D4
4. *Bypass Register Block Layout* --D5
5. *Instruction Register Block Layout* ---D5
6. *Boundary-Scan Cell Layout* --D6
7. *Boundary-Scan Input Cell Layout* --D6
8. *Boundary-Scan Controller Block Layout* ---D7
9. *Boundary- Scan Controller Block Layout – Blocks overview* -------------------------------------D8

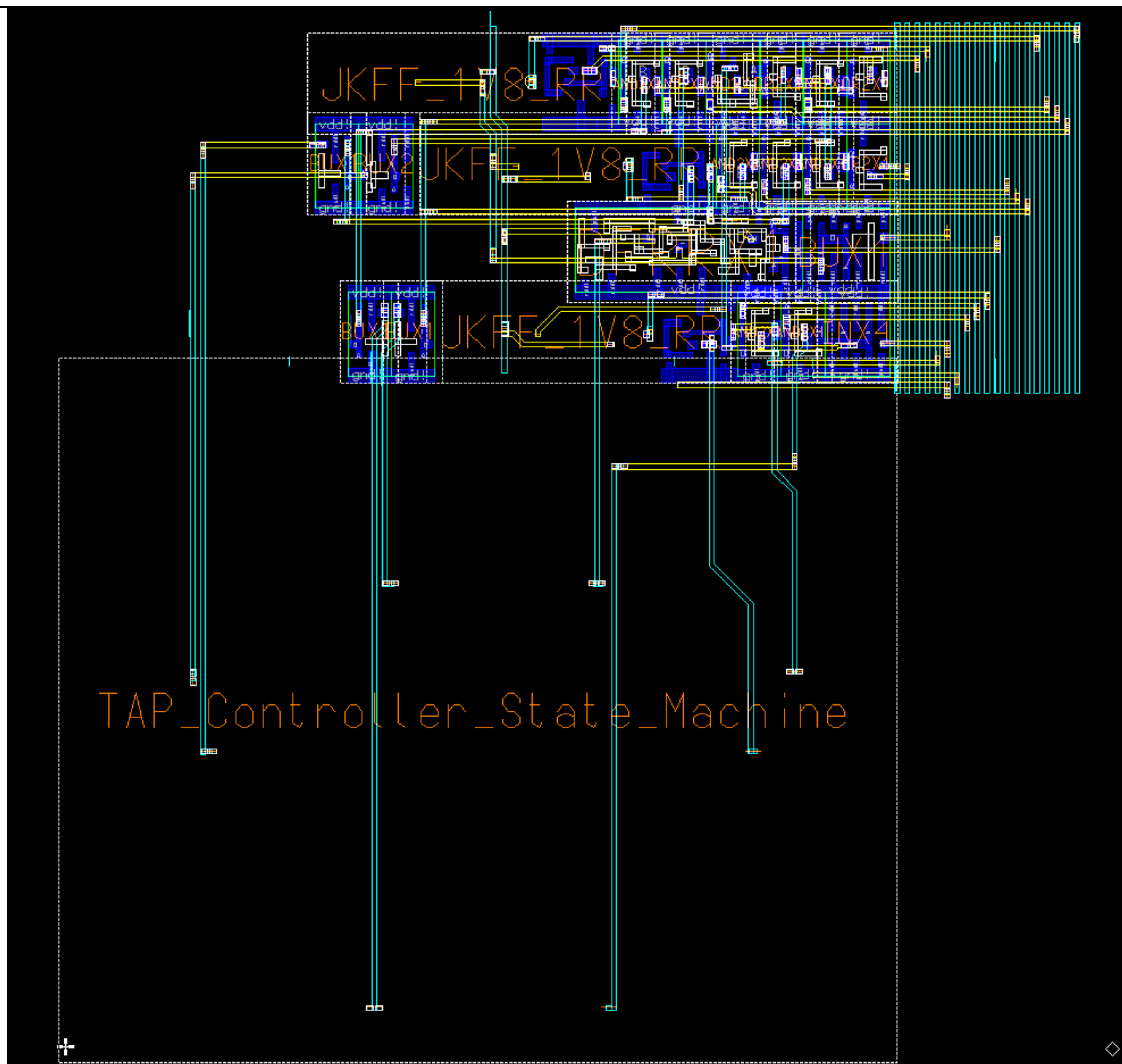
1. TAP Controller State Machine Layout



2. TAP Controller Block Layout



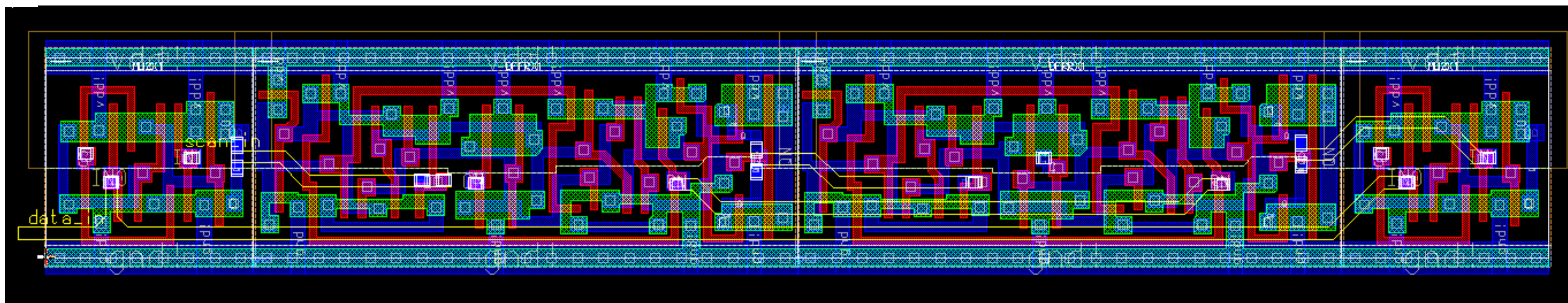
3. TAP Controller Block Layout – Blocks overview



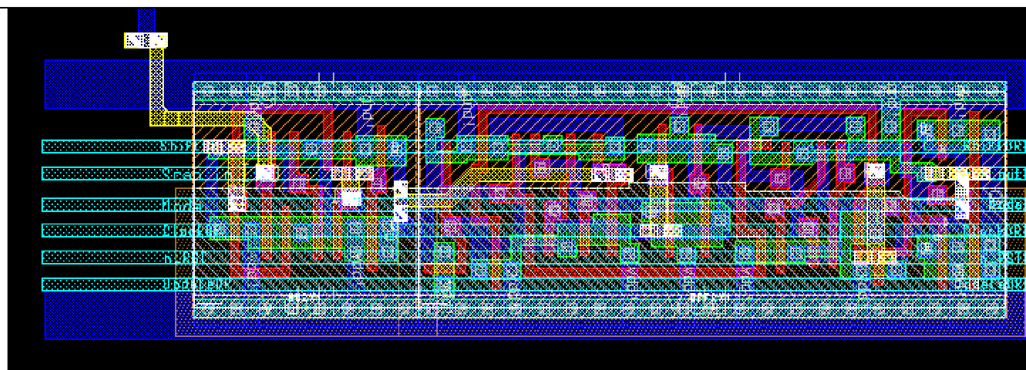
4. Bypass Register Block Layout

5. Instruction Register Block Layout

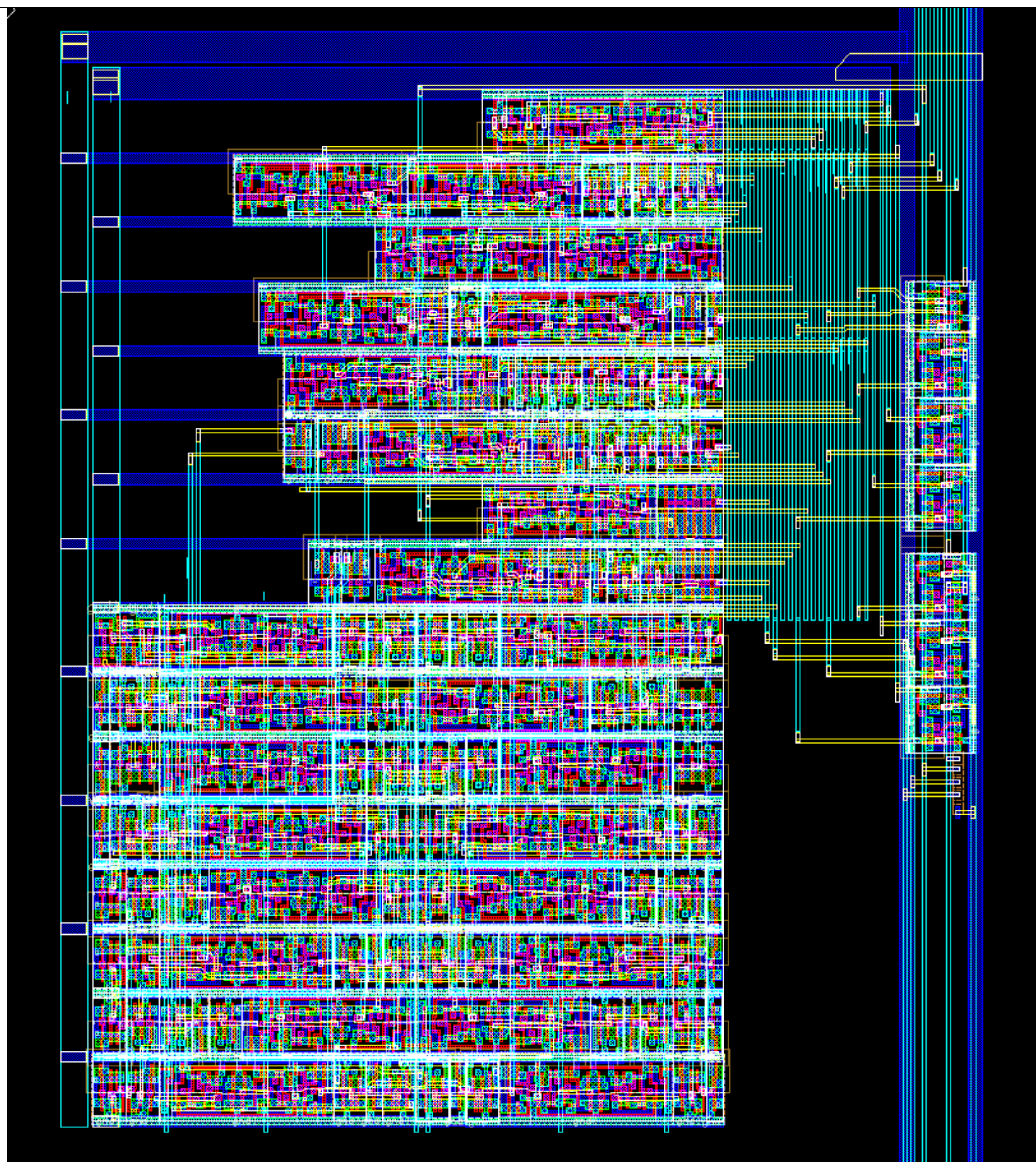
6. Boundary-Scan Cell Layout



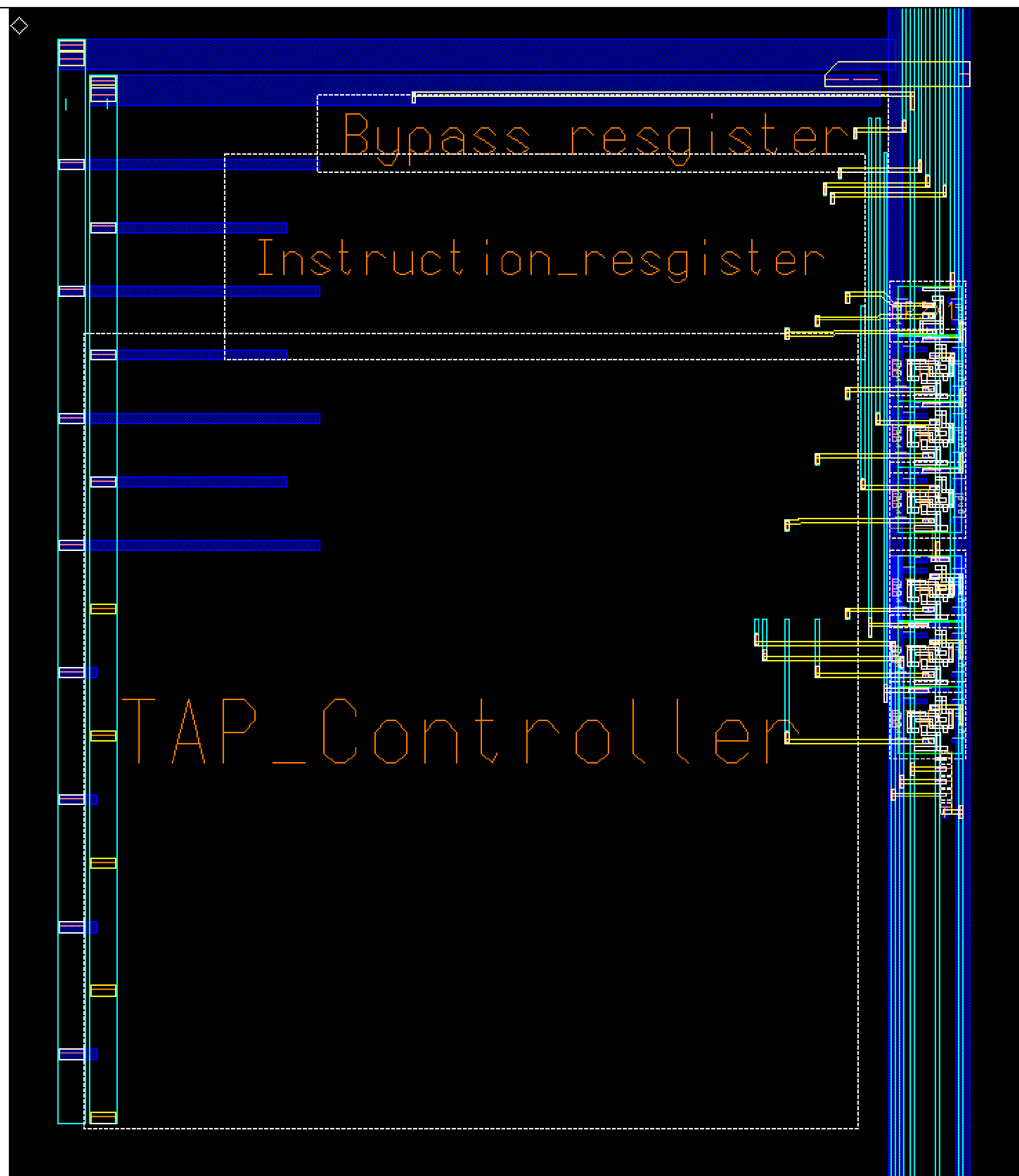
7. Boundary-Scan Input Cell Layout



8. Boundary-Scan Controller Block Layout



9. Boundary- Scan Controller Block Layout – Blocks overview



Appendix 5 –SVF and BDSL files used to test the prototype

Table of contents

1. *SFV file used to test the SAPLE-PRELOAD instruction* -- *E2*
2. *SFV file used to test the BYPASS instruction* -- *E2*
3. *SFV file used to test the EXTEST instruction* --- *E3*
4. *Sensor BDSL File* --- *E3*

1. SFV file used to test the SAPLE-PRELOAD instruction

```
// Created using Xilinx iMPACT Software [ISE - 10.1.03]
// Updated to be used on SENSOR PROTOTYPE with Boundary Scan Testing.
//
// Updated by:  Pedro Santos
//
// Content: SVF file that test the sample functionality of SENSOR PROTOTYPE.
//           It takes a snapshot of the device I/Os and shift it out to the TDO signal.
//
// REVISION HISTORY:
// Functionality created by Pedro Santos
//
// Begin of sample program
TRST OFF;                                       !Disable test reset line
//
RUNTEST 100 TCK;                                    !RUNBIST for 100 TCK clocks
ENDIR IDLE;                                            !End IR scans in IDLE
ENDDR IDLE;                                            !End DR scans in IDLE
//
SIR 2 TDI (0) SMASK (3);                  !Define a 2 bit IR scan
SDR  128  TDI  (00000000000000000000000000000000)  TDO  (80001800007FFFF3FFFFC000037FFF3F)
SMASK (ffffffffffffffffffffffffffffffff);       !Define  a  128  bit  DR  scan  for  sample
functionality
//
STATE IDLE;                                            !Go to stable state IDLE
RUNTEST 100 TCK;                                    !RUNBIST for 100 TCK clocks
// End of sample program
```

2. SFV file used to test the BYPASS instruction

```
// Created using Xilinx iMPACT Software [ISE - 10.1.03]
// Updated to be used on SENSOR PROTOTYPE with Boundary Scan Testing.
//
// Updated by:  Pedro Santos
//
// Content: SVF file that test the bypass functionality of SENSOR PROTOTYPE.
//           Permits to bypass the actual device with one TCK clock delay - what goes in
on TDI signal, goes out on TDO signal.
//
// REVISION HISTORY:
// Functionality created by Pedro Santos
//
// Begin of bypass program
TRST OFF;                                       !Disable test reset line
//
RUNTEST 100 TCK;                                !RUNBIST for 100 TCK clocks
ENDIR IDLE;                                            !End IR scans in IDLE
ENDDR IDLE;                                            !End DR scans in IDLE
//
SIR 2 TDI (3) SMASK (3);                  !Define a 2 bit IR scan
SDR 16 TDI (0abf) SMASK (ffff);           !Define a 8 bit DR scan for bypass functinality
//
STATE IDLE;                                            !Go to stable state IDLE
RUNTEST 100 TCK;                                    !RUNBIST for 100 TCK clocks
// End of bypass program
```

3. SFV file used to test the EXTEST instruction

```
// Created using Xilinx iMPACT Software [ISE - 10.1.03]
// Updated to be used on SENSOR PROTOTYPE with Boundary Scan Testing.
//
// Updated by:  Pedro Santos
//
// Content: SVF file that test the sample functionality of SENSOR PROTOTYPE.
//          It takes loads a external test pattern and applies it to the sensor
prototype.
//
// REVISION HISTORY:
// Functionality created by Pedro Santos
//
// Begin of sample program
TRST OFF;                                         !Disable test reset line
//
RUNTEST 100 TCK;                                      !RUNBIST for 100 TCK clocks
ENDIR IDLE;                                             !End IR scans in IDLE
ENDDR IDLE;                                             !End DR scans in IDLE
//
SIR 2 TDI (2) SMASK (3);                                !Define a 2 bit IR scan
// ALL FPGA SIGNALS AT 0 UNLESS THE N_CS_BTM AND N_CS_TOP SIGNALS
SDR 128 TDI (ffffffffffffffffffffffffffffffff) TDO (800008000000000000000000200000e7)
SMASK (ffffffffffffffffffffffffffffffff);        !Define a 128 bit DR scan for sample
functionality
//
STATE IDLE;                                             !Go to stable state IDLE
RUNTEST 100 TCK;                                      !RUNBIST for 100 TCK clocks
// End of sample program
```

4. Sensor BDSL File

```
----------------------------------------------------------------------------
-- M A S T E R S - S E N S O R  -  B O U N D A R Y   S C A N   T E S T    --
----------------------------------------------------------------------------
-- BSDL file
-- File Name:     MASTERS.BSD
-- File Revision: 0.1
-- Date created:  2012-01-23
-- Created by:    Awaiba, lda
-- Support:       pedro@awaiba.com
--
-- Device:        MASTERS sensor
--                rev I
-- Package:       428 pin FR4 on COB board
--                110 pins on the Boundary scan chain
--
--
--
--
-- Notes:
-- 1. The sensor is compliant with the basic JTAG Boundary scan chain. Only
--    the digital I/o's and the LVDS signals are on the Boundary scan chain.
-- 2. The implemented commands are BYPASS, EXTEST and SAMPLE-PRELOAD.
--

entity MASTERS_Sensor is

   generic (PHYSICAL_PIN_MAP : string := "FR4_on_COB_PACKAGE");
```

```
port (  --  LEFT SIDE
        --
        COL_OUT_TRNASF_EN_TOP0: in bit;
        COL_OUT_TRNASF_EN_TOP1: in bit;
        SUM_TOP: in bit;
        TDI_SEL_TOP0: in bit;
        TDI_SEL_TOP1: in bit;
        TDI_SEL_TOP2: in bit;
        TDI_SEL_TOP3: in bit;
        TDI_SEL_TOP4: in bit;
        --
        VDDD1: linkage bit ;
        VSSD1: linkage bit ;
        VSSA1: linkage bit ;
        VDDA1: linkage bit ;
        VDDIO1: linkage bit ;
        VSSIO1: linkage bit ;
        --
        WRITE_TOP: in bit;
        READ_TOP: in bit;
        BIT_SEL_TOP0: in bit;
        BIT_SEL_TOP1: in bit;
        BIT_SEL_TOP2: in bit;
        BIT_SEL_TOP3: in bit;
        OUTER_SEL_TOP0: in bit;
        OUTER_SEL_TOP1: in bit;
        OUTER_SEL_TOP2: in bit;
        --
        VDDD2: linkage bit ;
        VSSD2: linkage bit ;
        VSSA2: linkage bit ;
        VDDA2: linkage bit ;
        VDDIO2: linkage bit ;
        VSSIO2: linkage bit ;
        --
        TRANSF_SHADOW_ADC: in bit;
        N_RST_PLL: in bit;
        MCLK: in bit;
        SPARE_DIG_IO1: linkage bit ;
        SPARE_ANA1: linkage bit ;
        RST_CDS: in bit;
        INT_CDS: in bit;
        SAMPLE_CDS: in bit;
        --
        VSSIO3: linkage bit ;
        VDDIO3: linkage bit ;
        VDDA3: linkage bit ;
        VSSA3: linkage bit ;
        VSSD3: linkage bit ;
        VDDD3: linkage bit ;
        --
        OUTER_SEL_BTM2: in bit;
        OUTER_SEL_BTM1: in bit;
        OUTER_SEL_BTM0: in bit;
        BIT_SEL_BTM3: in bit;
        BIT_SEL_BTM2: in bit;
        BIT_SEL_BTM1: in bit;
        BIT_SEL_BTM0: in bit;
        READ_BTM: in bit;
        WRITE_BTM: in bit;
        --
        VSSIO4: linkage bit ;
        VDDIO4: linkage bit ;
        VDDA4: linkage bit ;
        VSSA4: linkage bit ;
        VSSD4: linkage bit ;
        VDDD4: linkage bit ;
        --
```

```
TDI_SEL_BTM4: in bit;
TDI_SEL_BTM3: in bit;
TDI_SEL_BTM2: in bit;
TDI_SEL_BTM1: in bit;
TDI_SEL_BTM0: in bit;
SUM_BTM: in bit;
COL_OUT_TRNASF_EN_BTM1: in bit;
COL_OUT_TRNASF_EN_BTM0: in bit;
--
-- BOTTOM SIDE
--
SPARE_DIG_IO2: linkage bit;
SPARE_ANA2: linkage bit;
VSSD01: linkage bit ;
DATA_CLK_BTM: out bit;
VDDD01: linkage bit ;
--
VDDD_LVDS01: linkage bit ;
VSSD_LVDS01: linkage bit ;
DATA_A_BTM0: out bit;
VSSD_LVDS02: linkage bit ;
DATA_B_BTM0: out bit;
VSSD_LVDS03: linkage bit ;
VDDD_LVDS03: linkage bit ;
--
VSSA01: linkage bit ;
VDDA01: linkage bit ;
--
VDDD_LVDS04: linkage bit ;
VSSD_LVDS04: linkage bit ;
DATA_A_BTM1: out bit;
VSSD_LVDS05: linkage bit ;
DATA_B_BTM1: out bit;
VSSD_LVDS06: linkage bit ;
VDDD_LVDS06: linkage bit ;
--
VSSD02: linkage bit ;
VDDD02: linkage bit ;
--
VDDD_LVDS07: linkage bit ;
VSSD_LVDS07: linkage bit ;
DATA_A_BTM2: out bit;
VSSD_LVDS08: linkage bit ;
DATA_B_BTM2: out bit;
VSSD_LVDS09: linkage bit ;
VDDD_LVDS09: linkage bit ;
--
VSSA02: linkage bit ;
VDDA02: linkage bit ;
--
VDDD_LVDS010: linkage bit ;
VSSD_LVDS010: linkage bit ;
DATA_A_BTM3: out bit;
VSSD_LVDS011: linkage bit ;
DATA_B_BTM3: out bit;
VSSD_LVDS012: linkage bit ;
VDDD_LVDS012: linkage bit ;
--
VSSD03: linkage bit ;
VDDD03: linkage bit ;
--
VDDD_LVDS013: linkage bit ;
VSSD_LVDS013: linkage bit ;
DATA_A_BTM4: out bit;
VSSD_LVDS014: linkage bit ;
DATA_B_BTM4: out bit;
VSSD_LVDS015: linkage bit ;
VDDD_LVDS015: linkage bit ;
```

```
--
VSSA03: linkage bit ;
VDDA03: linkage bit ;
--
VDDD_LVDS016: linkage bit ;
VSSD_LVDS016: linkage bit ;
DATA_A_BTM5: out bit;
VSSD_LVDS017: linkage bit ;
DATA_B_BTM5: out bit;
VSSD_LVDS018: linkage bit ;
VDDD_LVDS018: linkage bit ;
--
VSSD04: linkage bit ;
VDDD04: linkage bit ;
--
VDDD_LVDS019: linkage bit ;
VSSD_LVDS019: linkage bit ;
DATA_A_BTM6: out bit;
VSSD_LVDS020: linkage bit ;
DATA_B_BTM6: out bit;
VSSD_LVDS021: linkage bit ;
VDDD_LVDS021: linkage bit ;
--
VSSA04: linkage bit ;
VDDA04: linkage bit ;
--
VDDD_LVDS022: linkage bit ;
VSSD_LVDS022: linkage bit ;
DATA_A_BTM7: out bit;
VSSD_LVDS023: linkage bit ;
DATA_B_BTM7: out bit;
VSSD_LVDS024: linkage bit ;
VDDD_LVDS024: linkage bit ;
--
VSSD05: linkage bit ;
PIXEL_CLK_BTM: out bit;
VDDD05: linkage bit ;
SPARE_ANA3: linkage bit;
TEST_DIG_OUT_BTM: linkage bit;
--
-- RIGHT SIDE
--
LVAL_BTM: out bit;
LVAL_BTM_TRISTATE: out bit;
TRANSF_READOUT_BTM: in bit;
SERIAL_SYNC_BTM: in bit;
N_RST_LOGIC: in bit;
ROW_TRANSFER: in bit;
SPARE_ANA4: linkage bit;
SPARE_ANA5: linkage bit;
SPARE_ANA6: linkage bit;
--
VDDD5: linkage bit ;
VSSD5: linkage bit ;
VSSA5: linkage bit ;
VDDA5: linkage bit ;
VDDIO5: linkage bit ;
VSSIO5: linkage bit ;
--
RST_CVC: in bit;
ADD_VALID: in bit;
ROW_ADD11: in bit;
ROW_ADD10: in bit;
ROW_ADD9: in bit;
ROW_ADD8: in bit;
ROW_ADD7: in bit;
ROW_ADD6: in bit;
ROW_ADD5: in bit;
```

```
--
VDDD6: linkage bit ;
VSSD6: linkage bit ;
VSSA6: linkage bit ;
VDDA6: linkage bit ;
VDDIO6: linkage bit ;
VSSIO6: linkage bit ;
--
ROW_ADD4: in bit;
ROW_ADD3: in bit;
ROW_ADD2: in bit;
ROW_ADD1: in bit;
TEST_OUT_BTM: linkage bit;
CLK_TEST: in bit;
ADC_TEST: linkage bit;
TEST_OUT_TOP: linkage bit;
--
VSSIO7: linkage bit ;
VDDIO7: linkage bit ;
VDDA7: linkage bit ;
VSSA7: linkage bit ;
VSSD7: linkage bit ;
VDDD7: linkage bit ;
--
N_RST_SPI: in bit;
FORCE_UPDATE: in bit;
MISO_BTM: out bit;
MISO_BTM_TRISTATE: out bit;
N_CS_BTM: in bit;
SCLK: in bit;
MOSI: in bit;
N_CS_TOP: in bit;
MISO_TOP: out bit;
MISO_TOP_TRISTATE: out bit;
SPARE_DIG_IO3: linkage bit;
--
VSSIO8: linkage bit ;
VDDIO8: linkage bit ;
VDDA8: linkage bit ;
VSSA8: linkage bit ;
VSSD8: linkage bit ;
VDDD8: linkage bit ;
--
TRST: in bit ;
TMS: in bit;
TCK: in bit;
TDI: in bit;
TDO: out bit;
SERIAL_SYNC_TOP: in bit;
TRANSF_READOUT_TOP: in bit;
LVAL_TOP: out bit;
LVAL_TOP_TRISTATE: out bit;
--
-- TOP SIDE
--
TEST_DIG_OUT_TOP: linkage bit;
SPARE_ANA7: linkage bit;
VDDD06: linkage bit ;
PIXEL_CLK_TOP: out bit;
VSSD06: linkage bit ;
--
VDDD_LVDS023: linkage bit ;
VSSD_LVDS023: linkage bit ;
DATA_A_TOP7: out bit;
VSSD_LVDS024: linkage bit ;
DATA_B_TOP7: out bit;
VSSD_LVDS025: linkage bit ;
VDDD_LVDS025: linkage bit ;
```

```
--
VDDA06: linkage bit ;
VSSA06: linkage bit ;
--
VDDD_LVDS026: linkage bit ;
VSSD_LVDS026: linkage bit ;
DATA_A_TOP6: out bit;
VSSD_LVDS027: linkage bit ;
DATA_B_TOP6: out bit;
VSSD_LVDS028: linkage bit ;
VDDD_LVDS028: linkage bit ;
--
VDDD07: linkage bit ;
VSSD07: linkage bit ;
--
VDDD_LVDS029: linkage bit ;
VSSD_LVDS029: linkage bit ;
DATA_A_TOP5: out bit;
VSSD_LVDS030: linkage bit ;
DATA_B_TOP5: out bit;
VSSD_LVDS031: linkage bit ;
VDDD_LVDS031: linkage bit ;
--
VDDA07: linkage bit ;
VSSA07: linkage bit ;
--
VDDD_LVDS032: linkage bit ;
VSSD_LVDS032: linkage bit ;
DATA_A_TOP4: out bit;
VSSD_LVDS033: linkage bit ;
DATA_B_TOP4: out bit;
VSSD_LVDS034: linkage bit ;
VDDD_LVDS034: linkage bit ;
--
VDDD08: linkage bit ;
VSSD08: linkage bit ;
--
VDDD_LVDS035: linkage bit ;
VSSD_LVDS035: linkage bit ;
DATA_A_TOP3: out bit;
VSSD_LVDS036: linkage bit ;
DATA_B_TOP3: out bit;
VSSD_LVDS037: linkage bit ;
VDDD_LVDS037: linkage bit ;
--
VDDA08: linkage bit ;
VSSA08: linkage bit ;
--
VDDD_LVDS038: linkage bit ;
VSSD_LVDS038: linkage bit ;
DATA_A_TOP2: out bit;
VSSD_LVDS039: linkage bit ;
DATA_B_TOP2: out bit;
VSSD_LVDS040: linkage bit ;
VDDD_LVDS040: linkage bit ;
--
VDDD09: linkage bit ;
VSSD09: linkage bit ;
--
VDDD_LVDS041: linkage bit ;
VSSD_LVDS041: linkage bit ;
DATA_A_TOP1: out bit;
VSSD_LVDS042: linkage bit ;
DATA_B_TOP1: out bit;
VSSD_LVDS043: linkage bit ;
VDDD_LVDS043: linkage bit ;
--
VDDA09: linkage bit ;
```

```
        VSSA09: linkage bit ;
        --
        VDDD_LVDS044: linkage bit ;
        VSSD_LVDS044: linkage bit ;
        DATA_A_TOP0: out bit;
        VSSD_LVDS045: linkage bit ;
        DATA_B_TOP0: out bit;
        VSSD_LVDS046: linkage bit ;
        VDDD_LVDS046: linkage bit ;
        --
        VDDD10: linkage bit ;
        DATA_CLK_TOP: out bit;
        VSSD10: linkage bit ;
        SPARE_ANA8: linkage bit;
        SPARE_DIG_IO4: linkage bit);

use STD_1149_1_2001.all;

attribute Component_Conformance of MASTERS_Sensor : entity is "STD_1149_1_2001";

attribute PIN_MAP of MASTERS_Sensor : entity is PHYSICAL_PIN_MAP;

constant FR4_on_COB_PACKAGE : PIN_MAP_STRING :=
"TCK : 1," & "TDI : 2," & "TDO : 3," & "TMS : 4";
  "COL_OUT_TRNASF_EN_TOP0: 1, " &
  "COL_OUT_TRNASF_EN_TOP1: 2, " &
  "SUM_TOP: 3, " &
  "TDI_SEL_TOP0: 4, " &
  "TDI_SEL_TOP1: 5, " &
  "TDI_SEL_TOP2: 6, " &
  "TDI_SEL_TOP3: 7, " &
  "TDI_SEL_TOP4: 8, " &
        --
  "VDDD1: 9, " &
  "VSSD1: 10, " &
  "VSSA1: 11, " &
  "VDDA1: 12, " &
  "VDDIO1: 13, " &
  "VSSIO1: 14, " &
        --
  "WRITE_TOP: 15, " &
  "READ_TOP: 16, " &
  "BIT_SEL_TOP0: 17, " &
  "BIT_SEL_TOP1: 18, " &
  "BIT_SEL_TOP2: 19, " &
  "BIT_SEL_TOP3: 20, " &
  "OUTER_SEL_TOP0: 21, " &
  "OUTER_SEL_TOP1: 22, " &
  "OUTER_SEL_TOP2: 23, " &
        --
  "VDDD2: 24, " &
  "VSSD2: 25, " &
  "VSSA2: 26, " &
  "VDDA2: 27, " &
  "VDDIO2: 28, " &
  "VSSIO2: 29, " &
        --
  "TRANSF_SHADOW_ADC: 30, " &
  "N_RST_PLL: 31, " &
  "MCLK: 33, " &
  "SPARE_DIG_IO1: 34, " &
  "SPARE_ANA1: 35, " &
  "RST_CDS: 36, " &
  "INT_CDS: 37, " &
  "SAMPLE_CDS: 38, " &
        --
  "VSSIO3: 39, " &
  "VDDIO3: 40, " &
```

```
"VDDA3: 41, " &
"VSSA3: 42, " &
"VSSD3: 43, " &
"VDDD3: 44, " &
       --
"OUTER_SEL_BTM2: 45, " &
"OUTER_SEL_BTM1: 46, " &
"OUTER_SEL_BTM0: 47, " &
"BIT_SEL_BTM3: 48, " &
"BIT_SEL_BTM2: 49, " &
"BIT_SEL_BTM1: 50, " &
"BIT_SEL_BTM0: 51, " &
"READ_BTM: 52, " &
"WRITE_BTM: 53, " &
       --
"VSSIO4: 54, " &
"VDDIO4: 55, " &
"VDDA4: 56, " &
"VSSA4: 57, " &
"VSSD4: 58, " &
"VDDD4: 59, " &
       --
"TDI_SEL_BTM4: 60, " &
"TDI_SEL_BTM3: 61, " &
"TDI_SEL_BTM2: 62, " &
"TDI_SEL_BTM1: 63, " &
"TDI_SEL_BTM0: 64, " &
"SUM_BTM: 65, " &
"COL_OUT_TRNASF_EN_BTM1: 66, " &
"COL_OUT_TRNASF_EN_BTM0: 67, " &
       --
       -- BOTTOM SIDE
       --
"SPARE_DIG_IO2: 68, " &
"SPARE_ANA2: 69, " &
"VSSD01: 70, " &
"DATA_CLK_BTM: 71, " &
"VDDD01: 72, " &
       --
"VDDD_LVDS01: 73, " &
"VSSD_LVDS01: 74, " &
"DATA_A_BTM0: 75, " &
"VSSD_LVDS02: 76, " &
"DATA_B_BTM0: 77, " &
"VSSD_LVDS03: 78, " &
"VDDD_LVDS03: 79, " &
       --
"VSSA01: 80, " &
"VDDA01: 81, " &
       --
"VDDD_LVDS04: 82, " &
"VSSD_LVDS04: 83, " &
"DATA_A_BTM1: 84, " &
"VSSD_LVDS05: 85, " &
"DATA_B_BTM1: 86, " &
"VSSD_LVDS06: 87, " &
"VDDD_LVDS06: 88, " &
       --
"VSSD02: 89, " &
"VDDD02: 90, " &
       --
"VDDD_LVDS07: 91, " &
"VSSD_LVDS07: 92, " &
"DATA_A_BTM2: 93, " &
"VSSD_LVDS08: 94, " &
"DATA_B_BTM2: 95, " &
"VSSD_LVDS09: 96, " &
"VDDD_LVDS09: 97, " &
```

```
        --
"VSSA02: 98, " &
"VDDA02: 99, " &
        --
"VDDD_LVDS010: 100, " &
"VSSD_LVDS010: 101, " &
"DATA_A_BTM3: 102, " &
"VSSD_LVDS011: 103, " &
"DATA_B_BTM3: 104, " &
"VSSD_LVDS012: 105, " &
"VDDD_LVDS012: 106, " &
        --
"VSSD03: 107, " &
"VDDD03: 108, " &
        --
"VDDD_LVDS013: 109, " &
"VSSD_LVDS013: 110, " &
"DATA_A_BTM4: 111, " &
"VSSD_LVDS014: 112, " &
"DATA_B_BTM4: 113, " &
"VSSD_LVDS015: 114, " &
"VDDD_LVDS015: 115, " &
        --
"VSSA03: 116, " &
"VDDA03: 117, " &
        --
"VDDD_LVDS016: 118, " &
"VSSD_LVDS016: 119, " &
"DATA_A_BTM5: 120, " &
"VSSD_LVDS017: 121, " &
"DATA_B_BTM5: 122, " &
"VSSD_LVDS018: 123, " &
"VDDD_LVDS018: 124, " &
        --
"VSSD04: 125, " &
"VDDD04: 126, " &
        --
"VDDD_LVDS019: 127, " &
"VSSD_LVDS019: 128, " &
"DATA_A_BTM6: 129, " &
"VSSD_LVDS020: 130, " &
"DATA_B_BTM6: 131, " &
"VSSD_LVDS021: 132, " &
"VDDD_LVDS021: 133, " &
        --
"VSSA04: 134, " &
"VDDA04: 135, " &
        --
"VDDD_LVDS022: 136, " &
"VSSD_LVDS022: 137, " &
"DATA_A_BTM7: 138, " &
"VSSD_LVDS023: 139, " &
"DATA_B_BTM7: 140, " &
"VSSD_LVDS024: 141, " &
"VDDD_LVDS024: 142, " &
        --
"VSSD05: 143, " &
"PIXEL_CLK_BTM: 144, " &
"VDDD05: 145, " &
"SPARE_ANA3: 146, " &
"TEST_DIG_OUT_BTM: 147, " &
        --
        -- RIGHT SIDE
        --
"LVAL_BTM: 148, " &
"LVAL_BTM_TRISTATE: 149, " &
"TRANSF_READOUT_BTM: 150, " &
"SERIAL_SYNC_BTM: 151, " &
```

```
"N_RST_LOGIC: 152, " &
"ROW_TRANSFER: 153, " &
"SPARE_ANA4: 154, " &
"SPARE_ANA5: 155, " &
"SPARE_ANA6: 156, " &
       --
"VDDD5: 8, " &
"VSSD5: 8, " &
"VSSA5: 8, " &
"VDDA5: 8, " &
"VDDIO5: 8, " &
"VSSIO5: 8, " &
       --
"RST_CVC: 8, " &
"ADD_VALID: 8, " &
"ROW_ADD11: 8, " &
"ROW_ADD10: 8, " &
"ROW_ADD9: 8, " &
"ROW_ADD8: 8, " &
"ROW_ADD7: 8, " &
"ROW_ADD6: 8, " &
"ROW_ADD5: 8, " &
       --
"VDDD6: 8, " &
"VSSD6: 8, " &
"VSSA6: 8, " &
"VDDA6: 8, " &
"VDDIO6: 8, " &
"VSSIO6: 8, " &
       --
"ROW_ADD4: 8, " &
"ROW_ADD3: 8, " &
"ROW_ADD2: 8, " &
"ROW_ADD1: 8, " &
"TEST_OUT_BTM: 8, " &
"CLK_TEST: 8, " &
"ADC_TEST: 8, " &
"TEST_OUT_TOP: 8, " &
       --
"VSSIO7: 8, " &
"VDDIO7: 8, " &
"VDDA7: 8, " &
"VSSA7: 8, " &
"VSSD7: 8, " &
"VDDD7: 8, " &
       --
"N_RST_SPI: 8, " &
"FORCE_UPDATE: 8, " &
"MISO_BTM: 8, " &
"MISO_BTM_TRISTATE: 8, " &
"N_CS_BTM: 8, " &
"SCLK: 8, " &
"MOSI: 8, " &
"N_CS_TOP: 8, " &
"MISO_TOP: 8, " &
"MISO_TOP_TRISTATE: 8, " &
"SPARE_DIG_IO3: 8, " &
       --
"VSSIO8: 8, " &
"VDDIO8: 8, " &
"VDDA8: 8, " &
"VSSA8: 8, " &
"VSSD8: 8, " &
"VDDD8: 8, " &
       --
"TRST: in bit ;
"TMS: 8, " &
"TCK: 8, " &
```

```
"TDI: 8, " &
"TDO: 8, " &
"SERIAL_SYNC_TOP: 8, " &
"TRANSF_READOUT_TOP: 8, " &
"LVAL_TOP: 8, " &
"LVAL_TOP_TRISTATE: 8, " &
       --
       -- TOP SIDE
       --
"TEST_DIG_OUT_TOP: 8, " &
"SPARE_ANA7: 8, " &
"VDDD06: 8, " &
"PIXEL_CLK_TOP: 8, " &
"VSSD06: 8, " &
       --
"VDDD_LVDS023: 8, " &
"VSSD_LVDS023: 8, " &
"DATA_A_TOP7: 8, " &
"VSSD_LVDS024: 8, " &
"DATA_B_TOP7: 8, " &
"VSSD_LVDS025: 8, " &
"VDDD_LVDS025: 8, " &
       --
"VDDA06: 8, " &
"VSSA06: 8, " &
       --
"VDDD_LVDS026: 8, " &
"VSSD_LVDS026: 8, " &
"DATA_A_TOP6: 8, " &
"VSSD_LVDS027: 8, " &
"DATA_B_TOP6: 8, " &
"VSSD_LVDS028: 8, " &
"VDDD_LVDS028: 8, " &
       --
"VDDD07: 8, " &
"VSSD07: 8, " &
       --
"VDDD_LVDS029: 8, " &
"VSSD_LVDS029: 8, " &
"DATA_A_TOP5: 8, " &
"VSSD_LVDS030: 8, " &
"DATA_B_TOP5: 8, " &
"VSSD_LVDS031: 8, " &
"VDDD_LVDS031: 8, " &
       --
"VDDA07: 8, " &
"VSSA07: 8, " &
       --
"VDDD_LVDS032: 8, " &
"VSSD_LVDS032: 8, " &
"DATA_A_TOP4: 8, " &
"VSSD_LVDS033: 8, " &
"DATA_B_TOP4: 8, " &
"VSSD_LVDS034: 8, " &
"VDDD_LVDS034: 8, " &
       --
"VDDD08: 8, " &
"VSSD08: 8, " &
       --
"VDDD_LVDS035: 8, " &
"VSSD_LVDS035: 8, " &
"DATA_A_TOP3: 8, " &
"VSSD_LVDS036: 8, " &
"DATA_B_TOP3: 8, " &
"VSSD_LVDS037: 8, " &
"VDDD_LVDS037: 8, " &
       --
"VDDA08: 8, " &
```

```
   "VSSA08: 8, " &
          --
   "VDDD_LVDS038: 8, " &
   "VSSD_LVDS038: 8, " &
   "DATA_A_TOP2: 8, " &
   "VSSD_LVDS039: 8, " &
   "DATA_B_TOP2: 8, " &
   "VSSD_LVDS040: 8, " &
   "VDDD_LVDS040: 8, " &
          --
   "VDDD09: 8, " &
   "VSSD09: 8, " &
          --
   "VDDD_LVDS041: 8, " &
   "VSSD_LVDS041: 8, " &
   "DATA_A_TOP1: 8, " &
   "VSSD_LVDS042: 8, " &
   "DATA_B_TOP1: 8, " &
   "VSSD_LVDS043: 8, " &
   "VDDD_LVDS043: 8, " &
          --
   "VDDA09: 8, " &
   "VSSA09: 8, " &
          --
   "VDDD_LVDS044: 8, " &
   "VSSD_LVDS044: 8, " &
   "DATA_A_TOP0: 8, " &
   "VSSD_LVDS045: 8, " &
   "DATA_B_TOP0: 8, " &
   "VSSD_LVDS046: 8, " &
   "VDDD_LVDS046: 8, " &
          --
   "VDDD10: 8, " &
   "DATA_CLK_TOP: 8, " &
   "VSSD10: 8, " &
   "SPARE_ANA8: 8, " &
   "SPARE_DIG_IO4: 8, ";

attribute Tap_Scan_In of TDI: signal is true;
attribute Tap_Scan_Mode of TMS: signal is true;
attribute Tap_Scan_Out of TDO: signal is true;
attribute Tap_Scan_Clock of TCK: signal is (1.0e06, BOTH);

attribute Instruction_Length of MASTERS_Sensor: entity is 2;
attribute Instruction_Opcode of MASTERS_Sensor: entity is "BYPASS (11)";
attribute Instruction_Capture of MASTERS_Sensor: entity is "01";
attribute Boundary_Length of MASTERS_Sensor: entity is 1;
attribute Boundary_Register of MASTERS_Sensor: entity is "0 (BC_1, *, control, 0)";

end MASTERS_Sensor;
```