VERIFICATION METHODOLOGY FOR A 3D STACKED MULTI-PROCESSOR CHIP

MASTER’S THESIS
Lausanne, August 19, 2011

Candidate: Paolo Giovannini
Supervisors: Prof. Yusuf Leblebici
Dr. Giulia Beanato
Dr. Alessandro Cevrero
Prof. Fabrizio Pirri
Prof. Fabrizio Bonani

MICROELECTRONIC SYSTEM LABORATORY:
Abstract

Three dimensional integration is emerging as a promising solution to continue the Moore’s law, despite of the limitation of the CMOS scaling is approaching. A performance increase joined to an extreme miniaturization can be accomplished by the design of innovative systems, constituted by multiple layers, bonded together and interconnected in the vertical direction. Indeed, reducing the global wires length, it is possible to obtain relevant improvements with respect to planar technologies, in terms of speed, bandwidth and integration density. The target can be achieved exploiting a new technology process still in development: the TSVs, able to bring signals across the different stacked layers of the device.

In this report the realization of a 3D multi-core processor, designed by the LSM Laboratory of the EPFL, is presented. The chip, called Miracle, is formed by the superimposition of at least two layers, each of them constituted by four RISC 32-bit units.

The core of this Thesis consists in finding the most flexible and efficient solution to fully validate the device functionalities. The chosen solution exploits an FPGA board that communicated with the JTAG interface on the Miracle. The setup is conceived to allow a multi-phase test procedure, starting from the verification of the architecture functional behavior and moving then to the estimation of the performances with ad hoc benchmarks; the test has been applied first of all to the chip emulation on FPGA, then to the single Miracle layers and finally to the ASIC device, after the stacking fabrication in clean-room.

All the design choices made during the realization of the testing infrastructure are presented in the central chapters of the document, with the highlighting of the main advantages, problems and configurations attempts. A hardware solution, exploiting re-timed input vectors extracted from the pre-prototype simulations, succeed in testing the basic functionalities. In order to reach more advanced objectives, a software solution is needed: with this higher level approach it is possible execute a complete benchmark on the multi-core system.

The experimental validation of the testing apparatus and of the functionalities of the chip model on FPGA is set up and described in details, with the contribution of the correct results obtained. The outcomes are promising, but some incompatibilities in the performances analysis are still unsolved and have to be polished in order to conclude the procedure positively.
Acknowledgments

The presentation of this report is the occasion to thank Professor Yusuf Leblebici for the opportunity to work at an interesting and innovative project. I am particularly grateful to Giulia Beanato for the time spent to follow my project and for the constant encouragement. I would also thank Alessandro Cevrero for all the precious advices that he gave me in these months and Hossein Afshari for helping me to keep up the good work. All the LSM people have been kind and friendly, both inside and outside the laboratory.

I can not forget to thank my family for demonstrating everyday their incomparable affection towards me and for the trust in my capabilities: without their support I would not be able to reach any of my objectives. A particular message of gratitude is for who make me always happy, despite the distance: Laura, for her tender support, Marta, for being my point of reference, Andrea and Leonardo, brothers of life.

Paolo Giovannini
Lausanne, Suisse
19-08-2011
“There are two ways to live your life. One is as though nothing is a MIRACLE. The other is as though everything is a MIRACLE.”
—Albert Einstein
## List of Figures

2.1 From 2D to 3D interconnected structures [1] ...................................... 5
2.2 RoadMap from the naked chip to the 3D staking [3] .......................... 6
2.3 a) RC delays predictions [1] b) comparison 2D - 3D approaches [4] . 7
2.4 SiP suspended wires [1] ............................................................... 8
2.5 3D model of inter-layer connections with TSV [1] .......................... 9
2.6 Sections and technological process flow of a SuperVia by Tezzaron [1] 11
2.7 Sections and process flow of a SuperContact by Tezzaron [1] .......... 12
2.8 Graphical comparison between W2W and D2W bonding techniques 13
2.9 Section of the 3D-MAPS device [6] .......................................... 14
2.10 Boundary Scan cell [8] ............................................................... 17
2.11 Logic of a Boundary Scan chain for one chip [8] ................. 18
2.12 State diagram of the TAP controller Finite State Machine [9] ... 19
2.13 Set of data register in the JTAG logic [8] ................................. 20

3.1 Layout of a Miracle layer [10] .............................................................. 24
3.2 Scheme of a Core Tile structure [10] ................................................ 25
3.3 Scheme of a Peripheral Sub-system [10] ........................................ 26
3.4 SEM top view of a TSV [11] ............................................................. 29
3.5 Electroplating @EPFL [11] ............................................................... 30
3.6 Experimental setup with the Timing Block ("hardware solution") ... 31
3.7 Experimental setup with OpenOCD ................................................ 32
3.8 Experimental setup with Probe Card ............................................. 32

4.1 Basic block diagram of the JTAG interface ............................ 34
4.2 Muxes for the selection of the input parameters .................... 36
4.3 RTL schematics of the designed reset synchronizer .......... 37
4.4 State diagram of the system input Finite State Machine ....... 39
4.5 Basic block diagram of the JTAG Synchronizer ..................... 39
4.6 Timing specification diagram for the JTAG communication .... 42
4.7 State Diagram of the Timing block and formation of the re-timed TCK 43
4.8 State Diagram of the Checking block ........................................... 44
4.9 EDK flux scheme [18] ................................................................. 45
4.10 Boundary scan chain on the board [18] ................................. 46
4.11 Simulation extracted from not Real Time Mode (sx) and Real Time (dx) 47
4.12 Connection of ICON and ILA ...................................................... 48
4.13 Timing specification of one TCK cycle ........................................ 49
4.14 Timing specification of TDI .......................................................... 51

5.1 Schematic view of the pin-out assignment of the JTAG connector [23] 54
5.2 Control (up) and Break Single Step (down) registers of the DSU [24] 58
5.3 Schematic of the epfl load command behaviour on the RAM .... 60
5.4 Possible output of the bypass status function .................... 61
5.5 OpenOCD output of the TAP detection in a scan chain ........ 61
5.6 Simple diagram of the scan chain adaptation .......................... 62
List of Tables

2.1 Description of fundamental JTAG instruction .................................. 21
3.1 LEON III architecture ....................................................................... 28
4.1 Properties of the ROM memory (Inputs example) ............................... 38
4.2 Properties of the FIFO (Inputs example) ........................................... 40
4.3 Working frequencies ......................................................................... 50
4.4 TDI delays ....................................................................................... 50
6.1 Wrap up of the basic functionalities tested ......................................... 66
6.2 Preliminary PCB testing flow ............................................................. 71
6.3 Test results on packaged chip ............................................................ 72
Introduction

1.1 Objectives

The purpose of this master thesis project consists in the design of a complete testing procedure capable of verifying the behavior of a multi processor chip, called Miracle, realized with a 3D technology. The work done is inserted as conclusion of a first level of development that lead to the presentation of a prototype.

The importance of this project resides in the possibility to prove the stability, reliability and efficiency of a new approach to a multi-core system. Devices exploiting the vertical integration are already present in the market, as the System in Package, but they are just constituted by the superimposition of chips at package level. Instead, to the best of the actual knowledge, the Miracle can be considered the first fabricated prototype of true 3D multi-core processor, realized starting from the wafer-level and using Trough Silicon Via for the vertical communication.

Testing the correct behavior and the performances of the entire system will lead to an evaluation of the real advantages introduced by the innovative design approach.

1.2 Preliminary studies

The definition of a test model has been created ad hoc for the system under analysis, still maintaining enough flexibility and reliability for future development and prototypes. A good knowledge of the chip under test became unavoidable for the realization of an efficient verification; hence a deep study of the technology and the architecture of the chip, with particular attention to the structure of each core and its JTAG module used for the external access, has been the first step.

The technology process for 3D integration is not directly involved in the testing operation, but nonetheless it represents an essential background to justify all the choices made at the RTL level. In this context, the various strategies for the stacking of different layers have been analyzed, making comparisons with attempts made by other groups and understanding advantages and limitations.

A large amount of time has been dedicated to a very detailed study of the testing approach through the JTAG communication, in particular, with the aid of a software able to generate commands in this format for the processor debug. An open source software can be used as basic platform but it needs to be adapted for the tested system, requiring a meticulous analysis of all the codes in order to allow the customization of the tool function.

1.3 Testing policy and experimental setup

The choice of the testing policy has been influenced by the will of flexibility and reproducibility; for this reason, an FPGA board is the most advantageous choice,
allowing the realization of different verification methods that share the same resources set.

In fact, the board constitutes a key element for both the main phases in which the testing is organized: an evaluation of the performances and the validation of the ASIC prototype.

The first step has been achieved downloading on the FPGA, a Xilinx Virtex5, the RTL model of the chip and sending to it JTAG instructions in order to verify its behavior. The communication with the chip can be realized following different strategies: the first consists in the setup of a hardware solution, with the creation of input vectors and the analysis of the output streams, while the second one operates at a higher software level, in order to use a more complete and versatile debug system.

Both the techniques are finalized to validate the chip functionality and move to the main objective: execute benchmarks able to test the functionalities and the performances of the chip in order to demonstrate the validity of the design and of the 3D integration approach for the processors.

In the second testing phase, the one realized on the ASIC chips, the board can also be used as an interface between the external world, meaning the source of inputs (memory storage or host computer) and the Miracle placed on specific PCB and Probe Cards, designed internally to the laboratory.

The validation is made first with the single layers before the stacking operation, assuring that the TSV fabrication and the bonding process are realized on perfectly working dies. This strategy introduces a large simplification on the test of the final device, because the causes of possible issue are limited to the superimposition process made in clean-room.

1.4 Outline of the project

The purpose of this section is to give a brief preview on the topics organization inside the report, arranged following the carrying out of the research activities during these seven months.

The Chapter 2 is focused on the description of the 3D integration process and its applicability to the stacked processor concept. A particular attention is given to the mode in which signals are passed among the different superimposed layers and so to the TSV (Through Via Silicon) technology. A section is dedicated to the revision of the JTAG standard format of communication: additional logic, instruction set and detailed behavior.

The Chapter 3 describes the Unit Under Test, the Miracle chip, from different point of view: design, general architecture and core structure. This section is fundamental to fix the constraints of the project and to give a precise collocation to the necessity of an ad hoc system of testing.

Chapter 4 concerns the implementation on FPGA of a specific digital design able to take input vectors stored in memory and to send them to the Miracle chip after a correct re-timing operation. This first solution is perfect for a preliminary validation of the general behavior without a huge amount of data, but presents limitations in the realization of a complete benchmark.

Chapter 5 describes a second solution, not completely alternative to the previous
one but extremely competitive in front of the need of controlling easily and in a limited amount of time a large quantity of data and operations for all the cores. The key element is the customization of an open source software able to send commands to the chip.

**Chapter 6** contains description and results of the experimental application of the designed infrastructures to the different testing phases: model on FPGA for performances evaluation and functionality validation of the ASIC Miracle single layers. **Chapter 7** concludes the project; further developments required to improve the verification methodology are also discussed.
The chapter plays the role of a propaedeutic description of needs and purposes of 3D integration technology. The analysis shows the major challenges of this new approach in terms of design, manufacturing and test, with the support of theoretical overviews and a couple of case studies.

2.1 3D integration concept

From almost two decades, the microelectronic domain concentrates the majority of the efforts in the miniaturization of the new systems by scaling down CMOS devices. In the last few years, the need of reducing the sizes has become strictly connected to the research of different solutions, exploiting the use of alternatives materials and innovative approaches both for the design and manufacturing process. This change is due to the prediction described in the Moore’s Law, that put in evidence how the proportional classical scaling method for semiconductor devices is now next to the saturation, slowing down the increase in speed and device density in the future generations.

At the same time, the trends shown by the ITRS (International Technology Roadmap for Semiconductor [2]) organization reveals that the demand for System on-a Chip (SoC), already high, it will continues to increase. Hence, present and future microelectronic applications require significantly more complex devices with enhanced functionality and performances. In particular, smaller multi-chip modules will become even more indispensable for the market of mobile phones, hand-held computers and chip cards. These kind of systems are often characterized by mixed signals and mixed technology systems, including DRAM, flash, logic, analog, RF and MEMS. Integration on a single structure of elements with different purpose and properties, can dramatically increases the chip area if they are arranged in a 2D manner, one near the other. A large space is not fitting with the concept of SoC and its specifications in terms
of sizes (Figure 2.1). The real bottleneck, when moving to deep sub-micron technology, rely on the signal transmission delays, that increase as a consequences of the larger area required: if many sub-systems are disposed in parallel, they have to be connected with longer wires, that are radically affecting the performances and reliability of the entire device.

Moreover, a two dimensional arrangement become not feasible when trying to integrate elements that cannot be realized on the same wafer because of the need of completely different fabrication phases. A possible solution is achievable with a three dimensional integration, meaning with the stacking of integrated circuits, connected together in a vertical manner through the Silicon so that they behave as a single device. In this way, from a theoretical point of view, many problems will be overcome:

- the length of the vertical interconnections is minimal and leads to the elimination of speed limiting long inter-chip wires; the reduction of the RC delays in a 3D structure is predicted in the Figure 2.3a [1];

- the die area can be drastically reduced exploiting the possibility to add more layers, hence cells, without changing the form factor;

- a natural method is given for partitioning the different sectors of the multi-purpose device, enabling optimizations and reducing the fabrication effort for the integration of heterogeneous technology. An example can be the possibility to integrate in an easier manner analog and digital circuits, generally characterized by two different technology nodes.
volume, weight and overall power consumption are potentially reduced, allowing a perfect compatibility with the requirements of every portable application. In fact, shorter wires will decrease the average load capacitance and resistance, lowering the number of repeaters along the line and so the power needed to ensure the signals propagation (Figure 2.3b [4]);

- the reduction of load and wire to wire capacitances leads to a decrease of the noise due to simultaneous switching events and of the noise coupling between the lines.

- the bandwidth of the system is increased with respect to a 2D organization of the same chips;

- all components can be integrated together starting from the wafer level, avoiding increasing packaging size and expensive single element assembling (as in Package on Package devices);

- in the long run, when the 3D technique will be fully developed the whole costs can be drastically reduced, comparing to the ones of implementing each successively generation of shrinked planar CMOS.

2.2 3D technology

The purpose of the 3D integration is very challenging: building up systems with high complexity just by combining devices, achieving potentially all the "hot buttons" of the microelectronics field: small area, low power, high performances and contained costs. So can be 3D integration be reduced to a simple and easy superimposition of layers connected vertically? Obviously, the answer is negative: 3D integration is a very promising idea but it has several issue that has to be solved under many points of view, from the fabrication process, to the thermal and power dissipation models, passing through the lack of tools that allow an efficient design in three dimensions. An "efficient design" is exactly the key expression for the success of 3D: just with
prudent choices and careful engineering, the high number of vertical interconnections needed can be limited and, consequently, the increase of costs, area and technological processes required for the realization of the system. The described approach is not a new concept but is still incomplete and explored just superficially, as it is noticeable from the good results achieved in some particular field and product. An example is the realization of stacked sensors, Package on Package (PoP) and System in Package (SiP) devices. In PoP structures, different chips already packaged are stacked in a vertical direction achieving a sensible reduction of the occupied area. All digital cameras and cell phone applications are currently use this technology, generally in order to stack logic and memory architectures. That solutions represents a 3D conception of the space but not an integration in three dimensions; a step further is made with the development of the SiP devices, now used for some portable purpose. In fact, these structures are constituted by the vertical stacking of multiple naked components, bounded to a wrapper package and internally connected by fine off-chip wire bonds or by solder bumps. The SiP architectures are already in the market, but they still do not represent a complete solution in the 3D field. They allow the possibility to stack a very high number of chips (also more than nine), but they are 2D finished layer piled up just for area reasons and separated from each other by a dummy structure in silicon: there is no real direct vertical communication among them. The connections are fabricated as suspended wires between the package and each layer, as shown in Figure 2.4. These elements require complex technological process, able to realize a reliable and efficient structures also from the mechanical point of view.

The new purpose, is the realization of structures exploiting the 3D dimension starting from the design of layers directly communicating. The development of this approach depends on a large number of limiting factors, but is actually slowed down by the research of a completely reliable and efficient solution for the fabrication [5]. The issues are summarized in the following sequence:

- realization of an internal connection between the chips, with the TSV (Through Silicon Via) formation inside the substrate;

- wafer thinning of the strata;

- alignment and bonding of the stacked structures.
2.3 Through Silicon Via

A TSV is a conductive connection realized between both sides of a silicon wafer, because it needs to penetrate not only the various materials composing the overlying circuitry but also the whole depth of the substrate. This reflects on the larger size (1 − 100 µm of diameter and 10 − 400 µm of depth) with respect to any other on-chip vias. TSV are electrically isolated from the silicon, thanks to the deposition of a silicon oxide layer, called generally TSV liner. It is also responsible of the presence of parasitic capacitances in the vias, that have to remain under acceptable threshold in order to exploit at maximum the reduction of the interconnections in the 3D structure.

For this reason, the material chosen and the methods applied in the realization of the TSV are very important; they can be categorized in several groups, according to the different technological process. In particular, TSV may are created during the IC fabrication, distinguishing between:

- FEOL TSV (Front-End-Of-Line), meaning realized before the Silicon front-end device processing, before the metal interconnection are laid down. The conducting material for the filling must be doped polysilicon in order to allow thermal and material compatibility with the subsequent step of the technological flow. That choice of materials entails the drawback of an higher resistivity respect to metal vias, too large for some applications. They are used for very advanced integration thanks to the smallest dimensions, that allows having thousands of vias per die, generally packed in closely spaced local arrays.
• BEOL TSV (Back-End-Of-Line), fabricated during the metal wiring procedure, generally at the beginning. In this case the structures can consist of either tungsten (W) or copper (Cu). Their sizes are larger than the previous kind, because they require the elimination of the dielectrics layers already deposited on the surface.

The Through Silicon wires can be realized also after the complete IC flow (post-BEOL). With this solution, a particular attention has to be given for the design of 3D structures; in fact, there is the necessity to leave an exclusion zone in the IC wiring level on the front side of the wafers in order to have a space completely free from other circuit for the TSV. These process create the largest vias, with a $30-50\mu m$ of diameter, limiting the density of interconnects to lower values, in the order of an hundred per die.

The vias in this case have to penetrate also the full stack of the BEOL insulators and conductors in addition to the silicon substrate; so it is generally performed a double etching phase. First a steady state process targeting the BEOL section is applied, while just after the standard Bosh Deep Reactive Ion Etching is used for the deepest part of the hole.

On the other hand the post-BEOL approach has a large advantage: allows to realize IC layer in "normal" foundries not yet specifically ready for this kind of vias formation.

In terms of the 3D integration, it is possible to distinguish between:

• VIA FIRST, fabricated before the bonding of the different layers, with a FEOL, BEOL or post-BEOL strategy.

• VIA LAST, fabricated after the bonding of the chips in the three dimensional structure, forcedly with a post-BEOL approach. Vias created in that conditions are quite large (between 5 and 50$\mu m$) and the order of density is more or less the same of the post-BEOL before bonding. They are etched form the backside of an already thinned wafer, unlike most other solutions in which are etched blind into the substrate and then revealed by thinning, with the definition of a stop layer during the process.

### 2.3.1 Case study: TEZZARON TSV

The TSV technology is actually still underdevelopment: several foundries and academic laboratories have realized their own model of inter-layer via, exploiting different design, fabrication techniques and materials. Analyzing them, the Tezzaron Semiconductor project has been pointed out as one of the most complete methods and also as a possible reference for any other solutions.

This foundry has conceived a complete process of three dimensional stacking, called FaStack and planned specifically for the superimposition of DRAM layers, but still conceptually valid for many kinds of applications.

All the Tezzaron work is based on the idea to use just existing foundry process, standard commercial equipments, and classical materials in order to keep low the costs. For the same purpose, they are limiting only to wafer-to-wafer structure,
2.3 Through Silicon Via

with a metal binding, that allows to forms both electrical and mechanical bond-
points in one process step.

The choice allows to keep the benefit of a very common fabrication flow from the
Via last technique and to apply it to the Via first category, more flexible and ad-
vanced. In particular, realizing TSV before bonding operation entails many others
advantages:

- generally the sizes are in average lower than the ones realized at the end of
  the IC fabrication;
- vias can be used for thinning control, thanks to the function of in situ polish
  stop made by the material present inside them (copper plus a tantalum layer
  or tungsten wrapped by nitride) and able to ensure an enhances substrate
  thickness control;
- vias can be used to align the masks for additional backside process step, be-
  cause they are clearly visible after the thinning.

Tezzaron has developed two different techniques: the first is a BEOL, giving a Su-
perVias structure, and the other is a FEOL, producing the so called SuperContacts.
The first possibility employs a connection between the via and a local metal wiring
(as it is possible to see in the Figure 2.6), gaining in flexibility but loosing in size.
In fact, it requires a large open field area with no transistor and no interconnection.
On the other hand, thanks to the BEOL type of process, it is not necessary to intro-
duce any change during the fabrication flow and it improves the thermal transfer
with the use of copper as filling material in the via.

Figure 2.6: Sections and technological process flow of a SuperVia by Tezzaron [1]

The second method is provided during the wafer fabrication, so it requires specific
and particular adaptation in the technology flow in order to perform all operations
in a single process step. When it is done, all the subsequents stacking phases be-
come greatly simpler. With this approach, the diameter of the vias can be extremely
decreased, also thanks to the covering properties of the tungsten filling the TSV, that
allows an higher aspect ratio.
The choice of this material has some other benefits, like reduction of the capacitances, the heat expansion and the stress (allows to put TSV close to other structures, in particular transistor), but also some drawbacks, like higher resistance, less heat transfer and difficulties in the power distribution.

The following figure (2.7) presents the section of a SuperContact and a list of the steps needed for the fabrication; a more detailed scheme of the technology flow is reported in the Appendix A at the end of this report. Finally, one of the major strong point of the technology can be the possibility to scale down these structures, improving the performances of vias in a 3D integrated design: Tezzaron has ensured the possibility to reach nano-TSV in the next five years, but actually has faced issues in the production of vias in external prototypes.

![Figure 2.7: Sections and process flow of a SuperContact by Tezzaron][1]

### 2.4 Thinning process

All 3D structures exploit a thinning operation in order to reduce the depth of the silicon substrate of the upper layers of the stack. The thinning process constitute the only way to open an access to the fabricated vias: they have a small depth inside the silicon, because of the limitation on the diameter size and on the fixed aspect ratio, still lower than 10:1.

It is performed in two subsequent phase: a grinding (including a coarse and a following fine process) and an etching or Chemical Mechanical Polishing (CMP). The first step, the back grinding, eliminate the majority of the unwanted depth, but causes physical damage to the wafer, including scratches, crystal mismatches, surface roughness and stress. All the defects are removed with one or more etching phases, but the thinned object become really weak and there is the risk of an irreparable crack.

For this reason the thinning step became crucial in the fabrication flow and it has to be done only after the bonding of two layers or attaching it to an handle wafer, in order to increase the mechanical resistance of the tiny element (obviously the handle layer is removed after the bonding).
2.5 Alignment and bonding process

One of the critical points in the 3D integration process consist in the alignment and bonding operations, because of the natural difficulties in reaching an high precision in these steps. There are, above all, three possibilities in the way in which the chips are stacked:

- Wafer-to-Wafer (W2W);
- Die-to-Wafer (D2W);
- Die-to-Die (D2D).

The first technique is the most practical for individual wafers with a very high yield, like memories, which contain dies with the same size die. With this approach it is ensured a sufficient probability that the majority of the dies are correctly working: for example, two wafers with an individual yield of 90%, give a good stacked device in the 81% of the case, assuming null the losses in the bonding operation. Generally, the problem is even more serious than a low yield and it results useful to switch to D2W methodology; in this case, known good dice of the top wafer, after testing and separation steps, are aligned and bonded to the known good dice of a bottom wafer. Obviously, this processes possible just if the chip are designed in order to have access to fully test directly the die. The image 2.8 explain clearly the difference between the two described approach. The D2D methods is instead preferred especially during the realization of a prototype, in order to save a large amount of costs, assuming negligible the loss in terms of assembly time and the difficulties to obtain a very precise alignment between the two small elements.

The bonding operation depends also on the side of the layers that constitute the contact surface between them: Face-to-Face or Back-to-Face bonding are possible in a 3D stacking fabrication.
F2F operation consist in the bonding of the top chip flipped on the bottom one and it can be performed either before or after the realization of the TSV. After the bonding, if the vias are present, the wafer is thinned down in order to expose the TSV. In the opposite case, the thinning process is propaedeutic to the etching of the TSV done from the back side of the top flipped chip. That kind of stacking is used just for two layers; in fact, after the first bonding there are no more top surface available for a F2F attachment between the second and third wafer and it is necessary to continue the stacking with a face to back solution.

B2F process, instead, allows to stack with the same orientation a high number of layers. Also in this case the presence of the TSV before the bonding is not so limiting. In fact, if they are already present, a backside process is performed to thin the top wafer and to create the necessary pads.

Finally the bonding is also characterized by the kind of technology and materials used in the process: generally for 3D stacking are exploited metal, direct or adhesive bonding. The analysis of the different methods is not a purpose of this thesis and consequently it is just useful to indicate the copper bonding as one of most used techniques in the applications already realized.

2.6 Case study: 3D-MAPS project

The market of 3D applications actually counts many devices exploiting the vertical directions (as seen in the previous sections: PoP, SiP and different types of stacked sensors) and is open to a new generation of prototypes realized in the academic field and based on a real 3D architecture. Many applications are still not considered or too much difficult to be implemented, despite their idea can be very promising for the future. An extremely challenging issue is the realization of a processor or multi-core system using a complete 3D structure with vertical interconnections between stacked layers. In fact, the largest memory bandwidth demanded by many-core applications candidates a structure in three dimension as the most efficient solution to solve that requirement. In this field, the first very interesting result has been achieved with the 3D-MAPS device, realized at the beginning of 2011 by a large team of engineer from the Georgia Institute of Technology [6]. It has been possible to discover deeply the system thanks to a direct meeting with its responsible, Professor Lee.
during a conference about the 3D integration, sustained at EPFL in July. This oppor-
tunity was also very meaningful for a confirmation of the good road map un-
dertaken with the Miracle project, thanks to the comparison of common experience 
with the vertical integration issue.

The project consists in a 3D processor composed by 64 5-stages pipelined cores, all 
arranged in a layer of 8x8 dies. Each of them has its private SRAM module stacked 
above it, in a second layer, and communicating through TSV.

The approach allows to reduce the required area and to eliminate long wires de-
pendencies, obtaining the purpose of a very huge bandwidth. In order to reach the 
objectives encountering the minimum number of possible obstacles, the Georgia-
Tech team has made a set of assumptions and design preliminary choices.

- The processor functionalities are minimized with the elimination of all the 
  not fundamental complex components, especially the area and wire domi-
nated ones, like decoder stages, buffers and blocks for the data disambigua-
tion mechanism. The consequence is a saving of area, power consumption 
  and risk of faults.

- The intra-core communication is designed in such a way that each unit is 
in contact just with its neighbors along the directions of the well-known four 
cardinal points (north, south, est, west). In this way the 2D routing is strongly 
reduced, leading to a decrease of power consumption and an increase of 
the performances. In addition, the whole layout is divided in four macro-
partitions of 16 cores each, in which units at the boundaries are not directly 
communicating each others.

- The two layers are bonded with a Face-to-Face strategy, allowing a very low 
resistivity links for the power/ground rings and extremely short connections 
for all the nets that cross the boundary surface.

- As a consequence of the previous choice, the TSVs are used only for off-chip 
I/O and power/ground purposes, limiting the area assigned to the vertical 
connections. In reality there is the necessity to leave more space for a set of 
dummy TSVs in the middle of the structure in order to respect the design 
rules about the pitch distance, imposed by the foundry (Tezzaron Semicon-
ductor).

In the project, a big effort has been spent to overcome the lack of efficient tools for 
the 3D design: the majority of the placement and routing operations is realized 
following a standard of 2D approach, but taking care manually of special and 
particular elements in the layout, in order to form the interaction with the second 
stacked layer.

One striking aspects of the 3D-MAPS is the analysis and the testings setup [7]. 
Verification of timing constraints, signals integrity, power noise and thermal 
dissipation have furnished a wide and exhaustive spectrum of data, completed by 
software-dedicated simulations.

The device is designed but not yet fabricated and so the Georgia-Tech group has 
concentrate its efforts on pre-prototype simulations, proving the functionality 
of the 3D model with a complete and satisfactory set of benchmarks. The test
operation is possible applying the Boundary Scan Cells method, communicating through JTAG format to the cores arranged in one or multiple scan chains; a necessary overview on this theoretical topic is provided in the next section. Respect to the standard behavior of 2D devices, changes in the architecture has to be applied to the stacked chip in order to create a unique interface with the testing system taking care of the correct signals repartition among the two or more layers. *Ad hoc* digital modules have to be added during the design phase, according to the specific requirements of the structure. In the case of the Georgia-Tech processor, the designers have decided to create from scratch a complete system able to manage the JTAG communication through all the layers (in the first idea, there where three additional layers of stacked DRAM). For the project, a new Test Access Port has been modeled with global and sector control units and a completely custom Test Control State Machine (TSCM), based on a large and complex FSM, that plays the role of "heart of the entire chip".

### 2.7 Testing of SoC and 3D devices

The core of the master thesis project is the realization of a complete procedure of testing of a 3D stacked device through the use of the Boundary Scan and consequently of the JTAG format as communication instrument. A brief revision of these two preliminary aspect of the work could result useful to make more complete the vision of the objectives.

#### 2.7.1 Introduction to testing

The testing operation on electronic components and devices can be defined significantly with two concepts: control and observation. In fact, the standard testing procedure is realized putting the investigated system into a known state, supplying defined and determinate input data (or vectors) and observing the response in order to see if the unit performs as designed and manufactured. If control or observation cannot be carried out, there is no way to know empirically if the system performs as it should. Testing operations are fundamental in the realization of any electronic device and for this reason during the years several types of solutions have been found out. Nevertheless, traditional testing methods are not able to fit the exigences of the micro-electronics components that have become the base of all the new technologies. In fact, it would require great deal of time and the realization of special hardware and complex automatic test equipment (ATE) adapted for each type of device, resulting in a huge increase of the costs and the development time. In addition, extensive testing is necessary for the evermore stringent reliability and performance standards. An external probing in many case is definitely impossible from the physical point of view, because of the difficulties in access to the test point in systems characterized by an high integration, a multi layer structure or fine-pitch pins. The turning point was the idea to incorporate design-for-test techniques into the system, allowing embedded verification as in the case of Boundary Scan Chain.
The new methods allow several advantages, as the reduction of the total costs of the testing procedure, the saving of time and an easier isolation of the faults at component level.

2.7.2 Boundary scan test

The Boundary Scan method relies on the application of a scan path at every Input/Output pin (namely, the boundary of the system) of ICs to provide controllability and observability access via scan operations [8]. A device that conforms to the JTAG standard contains one instruction register (IR), a number of test data registers (DR) and a test access port (TAP) controller which handles all test operations (par. 2.7.3).

In the Figure 2.10, it is represented an essential scheme of an IC with the logic needed to the application of a boundary scan path consisting of a series of boundary-scan cells (BSCs), connected to a component’s inputs and outputs (or in other words, one for each functional I/O pin).

They are interconnected to form a scan path (or better a shift register) between the host IC’s test data input (TDI) pin and test data output (TDO) pin. During normal operation, input and output signals pass freely through each BSC, from the normal data input (NDI), to the normal data output (NDO). However, when the test phase is enabled, the IC’s boundary is controlled in such a way that the TAP bus master shifts in the test patterns into the chain and apply known values to the scan cells. The scan cells previous content is shifted out of the component and can be captured by the TAP bus master.

The instruction and data registers are included in separate scan paths, arranged between the primary test data input pin and the output one. This architecture allows the TAP to select and shift bits through one of the two path, without accessing the other one.

All the details presented about the scan chain logical architecture are visible in the example of one core test in Figure 2.11, reported in the next page.

From the scratched model, it is possible to discover that the standard format for JTAG is composed by five different signals; a part of the already described TDI and TDO, there are also:

- TCK, this signal allows data to be scanned into multiple elements of the chain, independently from components specific system clock. It is not a common clock signal because it may be stopped at logic 0 or 1 for an indefinite time, in which test components are guaranteed to retain their current state.
• TMS, that is the signal in charge of selecting the behavior of the JTAG state machine (see Figure 2.11). The standard requires the application of a logic 1 to TMS when the signal is undriven, ensuring normal operation when no test equipment is connected.

• TRST, or better nTRST, an active low reset used to asynchronously initialize test equipment, if it is necessary (optional operation).

These signals are controlling all the JTAG operations through a Finite State Machine implemented in the TAP logic, represented in Figure 2.12: in particular, it is driven by the TMS signal clocked on the rising edge of the TCK.

Figure 2.11: Logic of a Boundary Scan chain for one chip [8]
The starting point of a test session is the communication initialization, obtained putting the TAP controller in the first of the six steady state implemented in the flow: the Test-Logic-Reset. It can be reached setting high the TRST signal (put low the nTRST for many configurations) or keeping the TMS setted for five TCK cycle (executing the loop in the upper part of the flow). During the reset phase there is the selection between the identification register or the bypass one (see the related paragraph 2.7.3). Then a protocol is applied (TCK edge and TMS unset), causing the TAP to exit the Test-Logic-Reset state and move through the appropriate states. The first action of a test mode is the reaching of a capture operation, that can act on the data register or on the instruction register according to the selection made between the two path inside the flow, perfectly mirrored but separate.

Figure 2.12: State diagram of the TAP controller Finite State Machine [9]

In regime test mode, generally, from the CAPTURE condition the FSM is entering in the SHIFT one, allowing the flux of bits between TDI and TDO. In fact, until the TAP controller keep low the TMS signal, the device under test is sampling TDI on
the rising edge of the JTAG clock, while on the falling edge the desired values start to be outputted on the TDO.

The procedure can directly jump to the exit of the path or entering in a PAUSE state, in which the situation is idled for an indefinitely duration, blocking any logic operations of the TAP. In any case, at the end of both path there is the UPDATE state, in which at the falling edge of the TCK, the current value of the serial shift register is latched in a shadow parallel one. For example, in the IR case, latching a new value on the instruction parallel output means making it the new current instruction.

### 2.7.3 Registers of Boundary logic

The instruction register is accessed when the TAP receive the correspondent scan protocol, providing the address and control signals required to work with a particular data register in the chain path.

The instruction register has a double structure, composed by a shift register (forming a single scan line between the TDI and TDO pins) and by a shadow latch register (structured with a series of one bit registers correspondent to previous ones). The behaviour of the instruction register during the TAP working flow have already been described in the previous paragraph.

The JTAG communication need the use of two fundamental data registers (the Boundary Scan and the Bypass ones), but it can use also more optional registers, like the Device Identification one. They are arranged in parallel from the TDI to the TDO and they are selected with multiplexers regulated by control signals internal to the TAP, as it is possible to see in the Figure 2.13.

![Figure 2.13: Set of data register in the JTAG logic](image_url)
the chip and providing the controllability and observability features required to perform the testing operation.

The Bypass register it is a single bit that, if selected, can provide a one bit path between the TDI and the TDO, in order to reduce the total length of the scan chain excluding the devices not involved in the test.

The Device Identification register is used to verify the device’s serial number, composed by a tag for the version, one of the part number and one for the manufacturer code. When this register is selected, the bit scanned out from the scan chain correspond to the TAP IDCODE contained in it.

### 2.7.4 JTAG instructions

The standard format for the JTAG defined by the IEEE provides nine test instruction, useful to create a test mode for a general chip; in the case of a processor as the LEON III instantiated inside the Miracle chip, are used just four more important instruction among the available ones, described briefly in the following table:

| Table 2.1: Description of fundamental JTAG instruction |
|--------------------------|---------------------------------|
| Name        | Description                                                                 |
| bypass      | When the BYPASS instruction is selected on a device, the 1-bit wide bypass register is connected as the current test data register. This allows the scan chain to be shortened, making accesses faster. A device in BYPASS mode should not perform any test operation. One binary code for BYPASS shall be all ones (e.g. b1111 or 0xf for a device with a 4-bit wide instruction register). |
| extest      | The mandatory EXTEST instruction selects the boundary-scan register as the current test data register. Signals that are driven from outside of the component are loaded into the boundary-scan register, during the falling edge of TCK in Capture-DR state. Signals that are driven from the component are loaded from the boundary-scan register on the falling edge of TCK in Update-DR state. This allows signals from the system to the component to be captured, and known values to be applied to signals driven from the component to the system. |
| intest      | The optional INTEST instruction also selects the boundary-scan register, but is used to capture signals driven out of the component, and known values to be applied to signals driven into the component. |
| idcode      | This is an optional instruction that selects a device identification register as the current test data register. While IDCODE is selected, no other test data register shall be selected. |
Miracle is a project born to demonstrate the feasibility of the vertical interconnections concept to a multi-core processor. The introduction of the TSV, fabricated in EPFL, leads to the first known solution in this field; main goals of the design are the obtainment of a highly flexibility and reconfigurability of the device. The chapter describe the main aspects of this challenging study.

3.1 Miracle description

Research study, modelizations and prototypes realized in the 3D integration field have assured that it can exist an alternative promising roadmap to the standard approach.
The Miracle chip was done to create a flexible, homogeneous and complex architecture, built up from the LEON III from Gaisler, an open-source processor characterized by a complete and powerful architecture compacted in a very small area. The property allows to arrange many complete cores, with all their private modules and common peripherals, in a custom 2D layer. This complete structure become the “central stacked unit” of the device; in fact, the Miracle is composed by the superimposition of more layers, creating a unique multi-core system extended both in 2D and in 3D. Each layer is composed by 4 cores and a specific additional circuitry, external to the cores, in order to manage the TSV connections for the signal transmission among the different “floor” and the possibility to auto-configure itself. The striking and powerful aspects of the Miracle reside in the possibility to:

- stack an high number of identical layers (for a maximum of four, according to the 2-bit layer signal of selection), increase the number of cores and so the processor’s performances without lowering the form factor;
- allow the scalability of the chip, scaling down all the components, without altering the overall behavior;
- automatic layer configuration at the start-up;
- limit the cost of the structure, thanks to the use of just one mask;
- limit the number of TSVs by serializing/deserializing signals;
- test each single layer before the stacking operation.

A comparison can be made with the 3D-MAPS structure (section 2.6); the last one is based on the stacked unit core-private memory made at wafer level; on top of that is just possible to add other layers of memories, able to increase the storage capability but not the processor ones.
24 3D multi-core: MIRACLE

The Miracle project has been realized around three main aspects: the analysis of the open-source processor architecture, the careful design of a layer and the fabrication of efficient vias. In the following sections are presented briefly these three parts of the work.

3.2 Miracle design

The layer entity, key element of the Miracle device, has to be designed with a particular attention in a 2D configuration, but considering the 3D purpose of the system and so building a set of digital modules able to manage the signals that have to pass from/to the upper and lower floor of the structure.

![Figure 3.1: Layout of a Miracle layer](image)

The Figure 3.1 represents a top view of the layout of a chip layer, in which are recognizable the main blocks:

- 4 Gaisler LEON III processors, with 32KB private ROM and cached RAM (each of them, called Core Tile);
- a Peripheral sub-system, with a 32KB shared memory;
- a Network On Chip interface (NOC), for the communications between sub-structures (mainly cores-peripherals);
- a JTAG interface;
- the TSV arrays for the intra layer synchronization;
- a PLL in order to generate an alternative clock.
In Figure 3.2, it is shown a schematic view of the Core Tile, in which are highlighted the LEON processor and its own private blocks of memory accessible through the AMBA bus. The access from the external side, especially for debugging operations, is ensured by the JTAG module, that is directly linked with the core thanks to the bus.

The communication between the core and the other parts of the system is achieved thanks to the use of a complex network interface, that exploits the shared memory. In fact, all the LEON units are equal but completely separated from each other: the only possible interaction is made with the exchange of data and information thorough the Peripheral Sub-System in which the common resources are implemented (Figure 3.3).

A fundamental role is played by the JTAG module, able to redistribute the signal for the testing in all the cores of the system, according to the layer identification number: they arrived all to the upper floor pads and then the correct ones are transmitted downwards through the TSVs path. The bottom, accessible just with the vias, is configured itself with a scan chain arrangement of the cores. For example, a layer in the middle of the stack receive JTAG bits from the vias derived from the upper part and takes care of sending data to the lower section.

For this version of the Miracle chip, the design follows a very conservative line in order to have a very high security and confidence on the behavior of the TSV.
instantiated in the structure. To achieve the better functionality there is an high redundancy applied to all the data signals: for each of them are present two vias. At the start-up of the system, a procedure in the ROM test the correct functionality of the TSV and set proper multiplexers in order to select the signals arriving just from the working wire. With this approach it is possible to ensure that at least one of them keep the data with the correct value and integrity. Clock and reset signals have also an higher form of safety, with a vertical transmission in three different TSV and using a glitch-free majority voter circuit: in the receiving layer, the assigned value is the one send correctly at least by two of the wires. For the same reason, a PLL is inserted in each entity of the stacked structure, in order to allow the rigenaration of the clock signal if the relative TSV are not able to furnish a signal with a sufficient integrity. The redundancy can be reduced in further prototype if the technology applied will show good results in the test phase, allowing a reduction of the dimension: in fact, as noticeable in the Figure 3.1, almost half of the 16 mm² of the area is occupied by the TSV.

The correct transmission of the intra layer signals allows also the increase of the number of "floors" stacked in the 3D structure and consequently the increase of a factor of 4 on the overall number of processors in the package, speeding up parallel work loads. Increasing the number of cores enhances the performances of the chip, thanks to possibility to distribute the computing and to enlarge the bandwidth exploited for processor to processor communication; the chip is synthesized to reach up to 300 MHz.

3.3 LEON III overview

Each core of the Miracle chip is constituted by an open source processor, the LEON III, realized by the Aeroflex Gaisler; it is a synthesizable VHDL model of a
32 bit processor RISC compliant with the SPARC V8 architecture and instruction set.
The idea was born inside the European Space Agency (ESA) in order to create a portable and non proprietary processor for space applications, but now it is used in several domains: automotive, multimedia, GPS, SoC platform and mobile phones. The LEON processor software is suitable for the Miracle structure, because it combines several different properties and reaching an optimum trade off between them:

- open-source with respect to high-price comparable IP cores;
- high performances;
- robustness and fault tolerance;
- advanced on-chip debug support unit;
- flexible and extensively configurable;
- energy efficient.

All the highlighted characteristics contribute to make the LEON III suitable for the realization of System on Chip (SoC), because it allows an easy integration with complex design, with several peripheral blocks and multiple core resources, as in the case of the Miracle.

The striking aspect of the LEON, under the research point of view, remains the high reconfigurability: the GRLIB library furnished by the processor provides reusable IP cores with general functional and logistic interfaces, that can be directly used but also modified to fill the requirements of any specific application. It takes mainly advantages from the "plug&play" capabilities of the libraries, peculiarly conceived to allow extensions from the final user. Leon has an high degree of freedom in the definition of modules and their properties, as for example in the instantiation of caches, Floating Point Unit (FPU) and Memory Management Unit (MMU).

In other words, this property gives the possibility to detect the system hardware through software, meanings that the software applications configure themselves to the underlying hardware structure, avoiding the necessity to be customized to each specific architecture. The LEON III can be easily the base of a multi processor system, because it is possible to instantiate several core in the same design, allowing also different configurations among them. Both for single or many-core processors, the user can rely on a unique hardware-debug interface, that allows a non intrusive access to all on-chip registers and memory. That flexibility is enhanced by the possibility to implement the system on several types of FPGA and ASIC technology, with a limited use of resources: a basic LEON III (with pipeline, caches and AMBA bus) consumes approximatively less than 25’000 gates of the total (more than 300’000) present in a Xilinx Virtex 5 chip.

The Master thesis research has involved the detailed study of the entire LEON III architecture, but a description of all the modules behavior is outside the purposes of this report. It is useful to just have a short list of the common generic internal modules (Table 3.1); instead, in the next section, is present only a brief overview of the central unit, "engine" of the processor.
Table 3.1: LEON III architecture

<table>
<thead>
<tr>
<th>Units present in the LEON III</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer Unit (7 stage)</td>
</tr>
<tr>
<td>Hardware multiply, divide and MAC units</td>
</tr>
<tr>
<td>High performances Floating Point Unit</td>
</tr>
<tr>
<td>Multi-way configurable data and instruction caches</td>
</tr>
<tr>
<td>PROM and SRAM memory private blocks</td>
</tr>
<tr>
<td>Memory controller and Memory Management Unit</td>
</tr>
<tr>
<td>Debug Support Unit (DSU)</td>
</tr>
<tr>
<td>TAP controller for JTAG test</td>
</tr>
<tr>
<td>AMBA AHB bus with arbiter and bridges</td>
</tr>
<tr>
<td>Uart, timers, interrupt controllers and GPREG port</td>
</tr>
</tbody>
</table>

3.3.1 Integer Unit

The LEON III Integer unit uses a single instruction pipeline formed by 7 stages:

- Fetch (FE): the instruction is fetched from the instruction cache, previously enabled;
- Decode (DE): the instruction is decoded and the relative target addresses are generated;
- Register Access (RA): operands are read from the registers;
- Execute (EX): are performed the necessary arithmetic, logical or shift operations;
- Memory (ME): data cache is read or written;
- Exception (EX) are resolved traps and interrupts;
- Write (WR) the result of any ALU, logical, shift operations are written back to the register file.

The instructions executed by the processor are stored in a circular trace buffer, controlled by the debug support unit without affecting the normal behaviour of the processor. In detail, this buffer is able to store a complete set of data for each operation: instruction address, instruction result, load or store data and address, trap information and time tags.

A striking aspect of the LEON III is the feature for the minimization of the power consumption during idle periods, with a system of power down. During the saving cycles, the pipeline is halted until the next interrupts occurs and so all internal signals and caches are static, reducing the dynamic switching and consequently the power waste.
3.4 TSV design and fabrication

The Thought Silicon Vias take the useful signal across the different layers, in order to assure an homogeneous and cohesive behavior among the floors of the stacked structure as it is only one unit on a single planar substrate.

As described in the previous chapter of this report, the formation of TSV is a complex and not already defined process, in which is still missing an optimum flow of steps able to achieve all the technological objectives: simple and standard operations, small sizes, high aspect ratio and controlled costs. For the Miracle chip, the project of the vias has been realized internally to the laboratory in all the aspects, from the layout interface to the fabrication in clean-room. Many different solutions and prototypes have been developed in order to achieve the better performances. Nevertheless, is still difficult to estimate which technique can be the most efficient in the context of the this multi-core structure: an interesting aspect can be the test of chip fabricated with various methods in order to have an helpful and complete response.

The realization of the single layers in an external foundry impose to drill TSV after the CMOS processing steps, using a Via Last approach. Furthermore, results of the same vias but applied to other purpose devices suggest as most promising solution the application of a chip embedding phase and a copper bottom-up electroplating.

The first session of the manufacturing process consist in the partial clamping of chips, rigorously known-good-dies (KGD), into a carrier wafer, in order to work easily on a set of diced chips, saving time and costs respect to operation on whole wafer with a not perfect yield. This technique has been finely improved with some expedient:

- the placement in the carrier can be done in a environment of isopropyl alcohol (IPA), letting the chips align themselves into the openings with the evaporation of the liquid substance;
- the embedded dies are sealed with a conformal parylene deposition at room temperature, that allows to reuse the handler wafer many times thanks to simple etching of the coating.

After the fixing operation, the TSV openings can be realized through the stencil carrier wafer: a Reactive Ion Etching (DRIE) is used to drill the aluminum pads in...
which vias are required and the silicon dioxide layer present under them. Then, a Deep RIE with the Bosch method manage to produce profound holes in the silicon substrate of the dies. The high aspect ratio is the major limit of the filling step, because a super conformal operation requires a continuous seed-layer film on via sidewalls, difficult to be achieved with conventional tools. Hence, it has been set up a low-cost TSV metallization technique exploiting the previous innovative embedding step. In fact, the bottom layer of the carrier structure is preemptively sputtered on both side with a Titanium/Platinum layer in order to have electrical connection for the electroplating and also an optimum adhesion layer. Then the carrier wafer and the deposited parylene are acting respective as mask and isolator on the sidewalls in order to confine the growth of the copper inside the TSV.

Finally, the electroplating process is made on all the structure (inclusive of bottom layer, carrier and sealed chips) enabling the uniform filling with Cu. The correct results of the fabrication is verified until 380µm of depth and in a range between 40 and 100µm. In details, the obtained structures are characterized by a diameter of 60µm and a 200µm pitch between neighbor TSVs. Consequently, nevertheless the good results obtained, the bottom openings are quite large in order to allow the filling operation; this factor weigh negatively on the total area of the chip, already enlarged by the choice of redundancy in the inter-layer transmissions. A possible improvement for future Miracle prototypes reside in the fabrication of vias during the CMOS processing (Via First), but knowing that in this case all the chip manufacturing steps have to be done simultaneous and so made externally to the laboratory. In fact, drilling the TSV before the BEOL process, could lead to a relevant reduction of their diameter and a consequently space saving, as promised by the TEZZARON foundry.

For the Miracle chip, is mandatory to use a F2B bonding in order to achieve a same modular stacking of the layers, that can be exactly equal among each other, laying down design and fabrication costs. It is impossible to apply a face to face approach as in the (only) two layer of the 3D-MAPS device, abandoning the attractive idea of having direct connections between the two floors and using TSV just for taking power, ground and I/O signals across the structure.

3.5 Miracle verification methodology

The architecture of the Miracle is innovative in many aspects and so become fundamental to submit it to a complete testing session, allowing the validation of
the design and the use of the TSVs as communication among a multi-layer processors system.

The verification approach has to involve the JTAG format for the debug of system on chip, as explained in details in the section 2.7. The choice requires supplementary circuitry in the Miracle chip for the control and the managing of the JTAG signals among the layers and the cores. Respect to the 3D-MAPS project of the Georgia-Tech it has been decided to entrust in the standard logic of the Boundary Scan Chain and not customize the whole TAP controller. However, aside from the different purpose and architecture of the two project, the meeting with Professor Lee demonstrates that, separately and in parallel, have been chosen two very similar procedure of testing, with the same role played by the FPGA board.

As anticipated in the introduction, the testing setup has to be characterized by an high degree of flexibility and by different kind of approaches, but considering that it has to be interfaced with a very specific design. Indeed, many phases of testing should be used as intermediate steps to achieve one unique purpose: control the system execution of a specific set of instructions (a program routine) in order to verify the performances of the device. They should be analyzed in absolute value and compared with same level solutions, that are not exploiting 3D integration.

Before the extraction of the performances, it is fundamental to validate the behavior of the basic functionalities of all the digital blocks, present in the design. The strategy has to be simple and efficient: send inputs to the cores, processing and verifying the obtained outputs using the JTAG ports, as already mentioned in the previous chapters. A technique to reach the objective is to store bit vectors on FPGA memory slots and then send them to the chip, after a correct re-timing operation according to the communication standard. The infrastructure required by this approach is visualized in the Figure 3.6 and described in details in the Chapter 4.

![Figure 3.6: Experimental setup with the Timing Block ("harware solution")](image)

The good results obtained with this method can be not enough efficient to reach the final objective. Afterwards, it has been designed a more flexible strategy based on the use of an on-line system host-terminal/FPGA/chip, in order to manage more complex commands, enabling the possibility to direct write routine codes in the core memories. Creating an accurate net of instructions, it is possible to involve all the functionality of the multi-core device, allowing therefore a performance analysis. The structure obtained is represented in the Figure 3.7, in which are put in
evidence all the possible different configurations:

- with FPGA as holder, to extract easily data and performances from the emulated Miracle (MiracleFPGA), as see in the schematic 1;

- with a tester-PCB, to validate the functionality of the single layers in the packaged chip (case 2);

- with the FPGA as interface, to allow a test on packaged chip but with a unique complete scan chain (all of the cores in parallel), as shown by the third portion of the image.

![Figure 3.7: Experimental setup with OpenOCD](image)

To reach this target, a debug software, OpenOCD, works as infrastructure after an adaptation to the device requirements. The testing setup is designed for the progressive validation of the Miracle version for FPGA, of the packaged layer of the real Miracle ASIC and finally of the total Miracle structure after the vertical integration made in the EPFL clean-room. The procedure on the naked chip exploits the same kind of approach of the other two phases, but with the introduction of a probe card to order to handle unpackaged dies (Figure 3.8).

![Figure 3.8: Experimental setup with Probe Card](image)
One solution for the testing operation, the closest to the "hardware level", is obtained working with external input vectors. In this chapter is put in evidence the purpose of that methods and are described in details all the digital modules involved in this application.

4.1 General purpose of the interface

The first issue of the project is to create an interface in order to test one (or more) specific core contained in one of the layers of the multi-stacked processor. This target is achievable with the design of a digital unit implementable on an FPGA that is connected to the PCB in which the chip under test is located. In order to create this block for primary analysis, the FPGA is not already connected to the USB_JTAG cable (and consequently to the software on the computer) but it do reference to previously stored data. In other words, it is possible to exploit the timing simulations of the chip digital architecture in order to obtain a set of inputs vectors in a JTAG format, that can be stored in memories on the same FPGA. The specific use of this interface relies on the type of the chosen input source: the vectors could correspond just to a particular test on one core or to a complete test on more units. The size of these operations are just limited by the area of the FPGA devoted to the data storage.

Instead, the design does not have specific constraints in term of area or maximum delays, but it must respect carefully and correctly the timing of the communication between the interface and the chip: this is the key point for the realization of the digital block. The analysis and different simulations of the structure, especially under the timing point of view, have lead to realize the design presented in the Figure 4.1. Each part and all the characterizing aspects are described in details in the next sections.

4.2 Interface Design

4.2.1 Clock generation

The first issue that has to be solved in the design of this digital circuit is the generation of a clock signal. The aim is to produce a very general scheme in order to have the possibility to change a wide range of parameters with a restricted effort, but at the beginning of the design process is however a good idea to use a very common value for the operative frequency, as for instance 50 MHz. The creation of this signal is made with a PLL circuit instantiated inside the FPGA in order to transform a clock signal already present on the Xilinx ML board in one with the wanted frequency. Consulting the electronic and electric schematics of
the board [14] and in particular the outputs of the System Clock Generation, the choice falls on the 33 MHz pin, that is directly connected to the FPGA. The Clock Generator module instantiated on the FPGA receive the external signal
at 33 MHz and produce the system clock at 50 MHz that will distributed across the digital block of the designed circuit.

The phase-locked loops circuit (PLL) is a dynamical system used to maintain the synchronization between the external and internal clock domains in spite of the delays introduced by the additional elements of the distribution network, compensating them in response to the frequency and phase of the input clock [15]. The peculiarity of the PLL is the feedback control, that allows a monitoring activity of the produced clock in order to correct possible misalignment with the external signal, adjusting its phase with the oscillator present inside the loop. The control signal that arrives to the oscillator is generated by a loop filter (LP) that converts the error signal produced by the phase detector (PD) as the difference of the input and output clocks. The variation in frequency is obtained multiplying and dividing the two signal, to produce an output equivalent to

\[ F_{OUT} = \frac{N}{M} \times F_{IN}. \]

In the design with Xilinx tool, it is possible to take advantage of the Core Generator with the IP models in order to design easily a PLL. The device has two input pins (the external clock and reset signals) and two outputs (the generated clock and a valid flag signal). The properties of the PLL are set up to obtain the better performance possible according to the application:

- optimized bandwidth, reaching best solution without the constraints of a specific bandwidth;
- CLK with BUFG, using a buffer at the output of the system for the phase alignment;
- multiplication factor of 15;
- division factor of 1 (general feedback division) * 10 (division for the specific output channel).

The PLL, with this settings, produces a system clock of 49.5 MHz; the difference from the planned frequency is absolutely not relevant because it has no influence on the behavior of the circuit but it is important to notice the high precision put in evidence by the quite small value of the jitter on the clock signal (259 ps).

### 4.2.2 Parameter check

The JTAG interface provide the possibility to completely shape the timing of the signals send to the chip with different parameters: F_DIV express the frequency of the new TCL, DEL_I and DEL_O can modify the duty cycle according to the need setup time before JTAG signal are valid. On the FPGA the number of input switches is limited at 8 pins and so it is necessary to introduce in the design a block for the multiplexing of this signal. On the board, it is possible to choose as inputs the select signals of three multiplexers shaped as in the figure below (4.2).

There is no need to have an higher number of choice inside the quite large range of frequency; so it contains four specific values, turned out as more important than the others. In fact considering the system clock at 50 MHz, the first three selection of
frequency division are able to produce a TCK correspondent to values commonly used for the JTAG communication (as seen in many specific sources of JTAG working frequency [16]): they respectively generate a clock at 6.25 MHz, 3 MHz and 1.5 MHz. The last selection of the multiplexers is inserted in order to reproduce the unitarian frequency of 1 MHz, but also because it correspond to the lower value reachable with the 6 bits counter used internally; in the design phase it has been considered sufficient, but it can be easily enlarged to allow a slower communication between the interface and the chip under test.

Also the delays (expressed in frequency divisions) are condensed in order to have just a useful range of values, instead of a complete spectrum of possibilities, too large and counter-productive. The selectable values are chosen in order to allow for each of the four frequency a duty cycle correspondent (more or less) to the 25%, 50% and 75% of the TCK period. The values of the delay before the reading of the TDO from the chip has been designed always greater than one division respect to the other one in order to compensate the presence of the fixed STATE_A in the Finite State Machine of the Timing Block. However, the two values are left more or less equal for the maintenance of a certain equilibrium inside the TCK shape respect to the transition of the states in the machine.

This block also checks the validity of the duty cycle value, calculated from the chosen parameter for the period, the delay before the rising edge and after the falling one:

\[ DC = F\_DIV - [DEL_I + DEL_O + 1]. \]

The condition is \( DC >= 2 \) and \( DC <= F\_DIV - 3 \), where 2 and 3 are the minimum number of states necessary respectively for the high and low part of the TCK. If it is not verified, an enable signal remain low and never activate all other parts of the system.

The behavior of this block is sensible to the system clock edge in order to make a design completely synchronized in each part.

Paolo Giovannini

Master Micro&Nano Technology
4.2 Interface Design

4.2.3 Reset synchronizer

The behavior of the system reset is another important and preliminary issue in the design of this interface, because it is needed to avoid the use of the external asynchronous reset. [17] In fact, this solution would entail many disadvantages for the entire system; timing analysis are more complicated, but the real problem is the risk of metastability condition for the output of flip-flops when the reset is asserted or de-asserted at (or near) a clock edge, causing the loss of the reset state. Another problem in asynchronous circuit is that there is an higher possibility of having spurious reset due to noise or glitches occurrence.

A synchronous system reset, instead, is able to extend this property to all the circuit and to solve the previously described problems. In fact, it is insured that reset can only occur at an active clock edge, that is acting also as a filter for small glitches. Particular attention as to be provided in the design of the reset synchronizer in order to avoid the metastability issue and this target is reached with the use of more flip-flop in series. For this digital block it has been decided that 2 master flip-flop are sufficient to obtain a good design result, avoiding any metastability that might be caused by the reset signal being removed asynchronously and too close to the rising clock edge.

Focusing deeper on this simple circuit, the first flip-flop could lead to potential metastability problems because the reset may go high too close to the rising edge of the clock input of the same flip-flop. So the second one is required because there is no logic difference between the input and output of it and so no chance that the output would oscillate between two different logic values.

The signals that are able to activate the synchronous signal (active on high level) are the external reset or the valid flag coming from the PLL, when the internal clock is not generated properly. Synchronous resets may need a larger pulse, width enough to guarantee that the reset is present during an active edge of the clock, but is not an effective problem because generally the machine is in a reset condition just at the beginning and at the end of a test operation.

---

**Figure 4.3: RTL schematics of the designed reset synchronizer**

---

Master Micro&Nano Technology
Paolo Giovannini
4.2.4 Data storage and system input block

The basic circuit of the interface is designed in the way that is more general and versatile possible; the basilar function is to extract data (function of system input block) from memories in which the values are stored in JTAG format from an input file.

The design constraints leave a lot of degree of freedom in terms of dimension, number of blocks, type and properties of the memories; in order to obtain good performances according to necessity, two different ROM block of 4096 lines of depth are chosen for containing inputs data: one with a word length of three bits (for TRST, TMS, TDI values) and the other one of just one bit (for expected TDO values).

In this design, a ROM type is sufficient because there is just the need of read, line by line, the values with which the memory is initialized from a COE file; the core of the VHDL code for a ROM memory is very simple but it has been chosen to use the CORE GENERATOR of Xilinx in order to make easier the initialization and more optimized the area.

The tool allows to create two type of ROM memories, block or distributed, that are different just from the reading process of the words, respectively synchronous and asynchronous; the system has need to have an enable signal that control the read operation for allowing changing in the frequency of operation and so the first type matches very well this requirement.

<table>
<thead>
<tr>
<th>Table 4.1: Properties of the ROM memory (Inputs example)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input ROM ( TRST, TMS, TDI )</strong></td>
</tr>
<tr>
<td>Memory type</td>
</tr>
<tr>
<td>Size (read width)</td>
</tr>
<tr>
<td>Size (read depth)</td>
</tr>
<tr>
<td>Address width</td>
</tr>
<tr>
<td>Latency</td>
</tr>
<tr>
<td>Reading operation</td>
</tr>
</tbody>
</table>

The ROM memory filled with the expected TDO values has a similar structure except for the read width equal to 1. The system input block, charged of the extraction of data from the memories, is constituted by a Finite State Machine that is able to provide input values to the circuit with the searched working frequency. This target is obtained with the use of two states and one counter, that is incremented at each system clock cycle until it has reached the threshold correspondent to the right timing.

The first state is the idle one, in which the machine remains during the reset operation and when the counter is running; one clock cycle before the threshold, the reading enable of the two ROM memories is activated and the machine goes to the second state, in which data are received and sent to the rest of the circuit together with a valid signal. The threshold for the counter is expressed by the formula:

\[ C_F = \frac{((1.25 \times F_{DIV}))}{2} \]
The expression allows to obtain a frequency of extraction a little bit lower than the working frequency of the timing block (device that send JTAG signals to the miracle chip), equals to the ratio between the system clock and the frequency division (F_DIV). It is possible to notice that the frequency of extraction is divided by two, because in the memory one clock cycle occupy two lines, one for the high portion of the input TCK present in file, one for the low segment.

**Figure 4.4: State diagram of the system input Finite State Machine**

### 4.2.5 Input JTAG synchronizer

The first use of the system is to take from a memory block the data extracted from the simulated behavior of the chip and presented in a JTAG format.

**Figure 4.5: Basic block diagram of the JTAG Synchronizer**

There is the need to synchronize the wanted input signals (TRST, TMS, TDI and also expected TDO) with the clock of the JTAG; this signal is not a continuous with
a perfect constant period and it could remain at low or high logic level for more than the expected half-cycle. For this reason, it necessary to make a synchronization on the rising edge of the TCK, using a circuit similar to the one used for the reset synchronizer, with a series of flip flops in order to avoid any possibility of metastability.

The three flip flops are working as an edge detector, giving an high level enable that allows the synchronization of the others input. Each of them is sampled when it has surely a valid value and there is no more the risk of finding it during an edge. The valid signal is also depending on the value of the JTAG reset (TRST), that is able to invalidate the data before the starting of the testing process, assuring space and time saving.

4.2.6 FIFO data storage

During the test, a few “current” data are stored in FIFOs in order to make more fluent the process and, overall, to allow a quasi-continuous input flux when the device is operating.

Also the FIFOs design allows several degree of freedom, concerning the type of structure (shift registers, block ram based...) and different properties (use of flags, way of reading and writing...). In the Table 4.2 are reminded the characteristics of the designed modules.

<table>
<thead>
<tr>
<th>Input FIFO ( TRST, TMS, TDI )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory type</td>
</tr>
<tr>
<td>Clock</td>
</tr>
<tr>
<td>Size ( width )</td>
</tr>
<tr>
<td>Size ( depth )</td>
</tr>
<tr>
<td>Programmable threshold</td>
</tr>
<tr>
<td>Latency</td>
</tr>
<tr>
<td>Reset</td>
</tr>
<tr>
<td>Writing operation</td>
</tr>
<tr>
<td>Reading operation</td>
</tr>
<tr>
<td>Flags</td>
</tr>
</tbody>
</table>

The other two FIFOs, the ones for the expected and real TDO, have exactly the same property excepts for the width, equal to 1.

In order to increase the flexibility of the design, the three FIFOs present in the system can be used in two different behavior, just with the addition of a switch and some double different enable condition.

The first operating mode is a Real Time Test, in which the result GO-NOGO is obtained directly after the sending of the JTAG timed signal to the miracle chip, according to an obviously latency of at least FDIV + 3 system clock cycle. This latency is expressed as the sum of one cycle of the re-timed TCK (equal to F_DIV) and three system clock cycle for writing the real TDO on the relative FIFO, reading that value and check it with the expected one.

In this configuration, with a normal work at regime, the FIFOs have always zero
or one element inside themselves; so the timing block has just to check if there are valid and readable values in the inputs FIFO and the checking block make the same verification on the other two FIFOs expected and real TDO.

The main disadvantage of this solution is that each few cycle of the timing block, there is a stop for at least one cycle in order to allow the re-charge of one value in the FIFO, because of the reading operation is done faster than the writing one.

The second operating mode is a not Real Time Test, in which the result GO-NOGO is provided with a very large latency from the moment in which the first input signal is stored in the relative fifos. So this type of test, differently from the previous case, not allow to have directly the input vector that has caused an error or a mismatch on the output.

In this case, the re-timing of the JTAG signal start just when the number of elements in the inputs FIFO has reached the programmable full flag (for instance, 250) and the checking block works just when also the TDO FIFOs have reached their threshold; so it is possible to conclude that there are more than 500 re-timed-clock cycle (500 * F_DIV) of delay. When the inputs FIFO is empty, all the process has to be stopped for all the time need to re-fill it until the threshold.

This second solution allows a complete continuous process in the interval between two re-filling operation, but in the whole test is slower and also need a set of condition to enable the final emptying of the elements at the end of the test operation. On the other hands it can have more advantages in follow complete implementation of this design.

4.2.7 Timing block

The timing circuit is the core of this design because it is the digital circuit in charge of the correct re-timing of the JTAG signals that are sent to the Miracle chip. In other words, its function is to generate a new clock for the JTAG data, with a frequency decided by the external, providing the number of division to be applied to the system clock. The re-timed TCK is obtained through a Finite State Machine, in which the flow among the different states establish the value of the clock (high or low) and the correct time of sending input vectors to the Miracle chip and of receiving output test bit from it. The aim is to respect the timing of access of a JTAG port and so have signal with the form presented in the Figure 4.6, between the interface and the Unit Under Test.

The timing block has one of its striking points in the large flexibility of the design, according to three external parameter that could be chosen with high degree of freedom in order to have the more adapt timing for the JTAG communication with the miracle chip. The FSM structure, shown in Figure 4.7, remain unchanged (behavior, conditions, number of states...), but it is possible to vary the frequency division of the new TCK signal and its duty cycle, thanks to two parameter: DEL_I, DEL_O.

These values correspond respectively to T_Tsetup and to the T_TDO out in the graph above; they can assume each integer value but obviously not zero, selecting an upper limit for the frequency of the new TCK to 10MHz: in order to have higher working frequency, it is necessary to increase the system clock above the 50 MHz
The described flexibility is reached with the use of a 6 bit counter (assuming to not work below 1 MHz for the new TCK, meaning around 50 division), that manage all the passage within the states and consequently all the timing of the constructed FSM. The number of states in the FSM could be reduced to just three steps but paying an unacceptable increase of the number of case conditions and so also of the logic present in this block. The used structure is considered the better trade off between the circuit area and the length of the states chain.

The behavior of this system can be more clear with a brief description of the functionality, starting from the Real Time Mode description and explained after only the differences that occur with the other working configuration (the switch is “static” and so the two modes are exclusive).

An occurrence of the system reset lead the machine in the STATE_C and cause a re-initialization of all the signals, maintaining the TCK low until one cycle after the end of the reset pulse. In this state, the counter starting to increment its value and it is also asserted an internal signal (idle), that results high when the reset is de-asserted and when it is possible to read values of TRST-TMS-TDI from the FIFO (in other word, when the inputs FIFO is not empty). When it is high, the STATE_C is responsible of the rising edge on the TCK.

The STATE_D has the task to apply the chosen delays, respecting the correct timing of the duty cycle and maintaining high the TCK signal: when the counter reaches the value correspondent to the decided duty cycle, there is the passage to the STATE_E. The latest one provide the reading enable to the Miracle chip, just if in the previous clock cycle an input vector (TCK-TRST-TMS-TDI) has been sent to it. If the DEL_O parameter has a value large that one, the machine remain in this state until the counter has reached the threshold established by the expression DC (duty cycle) + DEL_O + 1; this situation can be very useful to assure that the TDO value from the Miracle chip is already valid in the moment in which it is analyzed.

Figure 4.6: Timing specification diagram for the JTAG communication
and after stored in the relative FIFO. These latest operations are made in the next STATE_A, in which the value extracted from the Miracle are sent to the related FIFO together with the writing enable. In this state there is also the enable of the next input vector reading from the related FIFO, if that is not empty. In the case of Not Real Time Mode, these operations are effectively done just if the testing process is not finished (the address of initial memory is the bigger one and the input FIFOs is definitively empty, meanings that no more JTAG communication has to be done between the interface and the Miracle).

The machine can always remain just one system clock cycle in the STATE_A before passing to the STATE_B, in which the input vector is sent to Miracle if it is already valid after the reading from the relative FIFO. The machine could remain for more cycle in this state, according to the value of the external parameter DEL_I, waiting that the counter has reached the threshold imposed by the frequency division. Than the counting is re-initialized in order to restart a new cycle from the next state, STATE_C.

Figure 4.7: State Diagram of the Timing block and formation of the re-timed TCK
4.2.8 Checking block

The value of the real TDO, obtained from the Miracle chip and stored in a FIFO, has to be compared with the expected one in order to complete the test operation. This function is provided by the checking module, constituted by a small Finite State Machine of two states: in the first one, the machine is waiting an enable signal, activating the reading of the two TDO FIFOs and the transition to the second state; in that one, if data extracted are valid, there is just the simple assertion of a “result test” signal in case the two outputs are equals.

The important part is constituted by the enable signals because they are different according to the operating mode (Real Time Mode or not): in the first case, the check block works when the two TDO FIFOs are not empty, instead there is a more complex scenario for the second case. In fact, in normal regime of non Real Time, the enable is given by the reach of the threshold in the number of element present in the real TDO FIFO. Just when it is full at threshold, a comparison of the outputs is made.

Figure 4.8: State Diagram of the Checking block
4.3 Debug and test of the interface

The design of the interface is very complex on the timing point of view, because of the difficulties to integrate and synchronize the three FSM with the signal supply from the Miracle chip. For this reason, the structure has to be verified with simulation and measurement made after the physical implementation on the FPGA board using the EDK tools; all the details of these operation are furnished in the next sections, with the addition of some theoretical reminder about Xilinx support tools.

4.3.1 EDK overview

The Xilinx Embedded Development Kit is a complete tool able to manage all the implemented modules at an high level of abstraction, integrating the ISE VHDL models and handle both the hardware and software elements of the chip.

![EDK flux scheme](image)

Figure 4.9: EDK flux scheme [18]

If the design is not involving the internal Xilinx Micro-controller, it is just sufficient to follow the ISE branch of the flux (the green section in the Figure 4.9) and directly download the VHDL design (synthesized, mapped and routed) on the board with the cable connection.

The high performance and flexibility of the board allows to store the design in a non volatile memory of the board, downloading automatically on the FPGA when the system is powered up. If the dimension of the program is not exactly known in advance, the reserved space can be enlarged creating a cascade of two Platform Flash PROM in the chain formed for the Boundary Scan operation (represented in the Figure 4.10), where the highlighted sections are the ones used for this application.

The entrance point is constituted by the PC4 JTAG connection with an host computer for the bit-streams download. After in the chain, are connected in series the two PROM memories (erased before any new programmation) and a System ACE.
controller (System Advanced Configuration Environment) that is able to write the VHDL code on the FPGA according to the selection of one of eight possible configurations.

Figure 4.10: Boundary scan chain on the board [18]

4.3.2 Experimental validation - Simulations

Before the effective download on the FPGA, the functionality of the JTAG interface can be verified by software exploiting the potentialities of the Xilinx and ModelSim to realize complete simulation of the system. First of all, it has been realized a register transfer level (RTL) description of the circuit with the purpose of checking the correct behavior of the structures before the implementation of the design, formed by the synthesis, mapping and Place&Route processes [19]. At the end of the process, with the routing, are implemented the different components of the design and the wires for the interconnection between them, computing also the relative delays originated by the obtained topology. With a well constructed test bench it is possible to simulate in details the behavior of the system in any working condition and with any of the possible set of input parameter, discovering the total time needed for the test operation and in which configuration are obtainable the better performances. The simulation shows that there is not a big difference in terms of total time needed for the test between the two mode of operation, if all the RAM memories are filled with useful data, meanings if there is an input TCK changing along all the 4096 trigger point of the ModelSim list. In other word they are comparable as long as the number of vectors is equal to the maximum size of the memories and the clock remains constant just for small interval of the test. In these cases, the Real Time Mode became strongly advantageous when the number of input vectors stored is lower than the maximum size of the memories, because in this case the testing operation is stopped when the last edge on the input TCK is caught from the machine. The set of vectors used for the one core test is constituted by almost 2000 lines and so in Real Time it is possible to halve the needed time: 234μs instead of the 455μs requested for the non real time mode with the same inputs (as seen in figure 4.11).
Figure 4.11: Simulation extracted from not Real Time Mode (sx) and Real Time (dx)
Obviously, dividing by two the frequency, the total duration of the test is almost doubled: for example, maintaining the set of vectors with 16 frequency division, the timing is almost $465\mu s$ for the Real Time and $900\mu s$ in the other case. The simulations on ModelSim allows also to check in detail the correct behavior of the digital circuit, in particular the timing of the JTAG communication, the interaction among the three Finite State Machine, the absence of glitches from all the output signals. A perfect functionality on the software, lead to have a verification on FPGA with an high degree of certainty at the first attempt.

### 4.3.3 Experimental validation - Debug System Unit

The functionality of the real system on the FPGA can be tested with a digital Debug System Unit, called ChipScope, that can be instantiated directly on the chip, allowing an internal verification of all the important signals as in a real Logic Analyzer. Use this debug tools involves some specific knowledge, reported in the following section.

The digital debug inside the FPGA is possible thanks to the instantiation of two particular IP modules, present in the Xilinx Core Generator library:

- the Integrated CONtroller (ICON) [20];
- the Integrated Logic Analyzer (ILA) [21].

![Figure 4.12: Connection of ICON and ILA](image)  

The controller provides an interface between the ChipScope software tool and the Analyzer core, via the JTAG Boundary Scan port of the target FPGA, bridging the gap between the FPGA TAP controller and the target cores. This last connection is made by a specific bi-directional control port (4.12), including JTAG clock, input and output data, and control signals necessary to configure and communicate with the target core. In order to test the system is enough one control channel of 36 bits of length [22]. The Logic Analyzer is a module able to monitor any internal signal of the design, acting in a complete synchronous behavior with all others components of the system; the core can capture the signals connected to it at the design speed and according to the trigger conditions. In fact, it is necessary to define one or multiple trigger ports of the ILA in order to monitor different kind of signals or groups of them; in this way, it is also possible to associate to each of them a different match logic unit ($=, \neq, >, <, \geq, \leq$) for the comparison and the creation of the trigger event. After the design is loaded into the FPGA, the ChipScope software can be used to assign a condition to each trigger signal (1 - 0) and among the various group (and - or), defining when and how generate the events.

Paolo Giovannini  
Master Micro&Nano Technology
After the trigger occurs and the sample buffer is filled, the data buffer is uploaded into the Analyzer software and the result can be displayed in waveforms or list format.

For a digital debug of the JTAG interface, it has been triggered on the rising edge of the re-timed TCK and on all the other signals that is useful to display: TRST, TMS, TDI, real TDO, expected TDO, pll_locked enable and test result. It is also possible to trigger on the signals representing the current state in order to verify the transitions of the different Finite State Machine presents in the design; finally, this is the real aim of the ChipScope Analyzer: check if every internal signals are correct at each state shift.

Among all the data taken, it is useful to put in evidence the timing of the JTAG communication through the Miracle_FPGA dummy ROM (4.13), with the respect of the setup and hold time of all the signals in relation with the edges of the TCK, as seen from a theoretical point of view in the Figure 4.6. The analysis with theChipScope are limited by the maximum number of samples that can be captured and stored in the buffer for waveform generation; this parameter can be enlarged during the instantiation of the ILA module, but however depends on the resources free space remained on the FPGA.

With low working frequency (below 3 MHz) and with the not Real Time Mode (slower in an average case), the number of samples needed to trigger all the JTAG vectors for the duration of the test operation is too high and it is not possible to have a complete waveform displayed in one time on the screen.

The problem is overcome exploiting an analog analysis with the oscilloscope.

4.4 Experimental validation - Analog Measurement

The analog analysis completes the scenario of the JTAG interface experimental validation, because allows to make more interesting measurements. The first step is to charge the design on the PROM memory, as seen in section 4.3.1, in order to automatically restore it on the FPGA at each power up of the board. Then it is just necessary to probe the output signals on the respective pins of the PCI connector, exploiting the multiple channel function of the oscilloscope to verify the timing specification on the JTAG communication. This target is reached probing all the JTAG signals and triggering on the TCK rising edge; the result is a confirmation of the ILA waveform and, as an example, is reported in the Figure 4.14 the relation-
ship between TDI and TCK. On the waveforms is possible to make some measurements in order to have an evaluation of the precision of the signals timing. For example, it is interesting to evaluate the frequency of the re-timed TCK, according to the possible variations introduced by the instantiated PLL and the Timing block; from the table 4.3 (realized on a quite small set of samples) it is possible to notice that the deviation from the expected values is not too large and correspond in average to a value lower than 1%.

Table 4.3: Working frequencies

<table>
<thead>
<tr>
<th>Nominal (MHz)</th>
<th>PLL (MHz)</th>
<th>Std dev on effective (MHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.25</td>
<td>6.1875</td>
<td>0.0271</td>
</tr>
<tr>
<td>3.125</td>
<td>3.0938</td>
<td>0.0057</td>
</tr>
<tr>
<td>1.5625</td>
<td>1.5469</td>
<td>0.0001</td>
</tr>
<tr>
<td>1</td>
<td>0.99</td>
<td>0.0040</td>
</tr>
</tbody>
</table>

Similar results of standard deviation are visible during the measurements of the timing interval present between a change on the TDI and the rising edge of the TCK. The uncertainty shown in the period of the clock, is multiplied (in a quasi proportional way) by the number of cycles present between the two events. Also in this case, the variation is negligible, because from the data analyzed and reported in the following table (with two different configuration of frequency and duty cycle), the standard deviation result lower than the 2% in any case.

Table 4.4: TDI delays

<table>
<thead>
<tr>
<th>Frequency div.</th>
<th>expected delay (ns)</th>
<th>std dev rising (ns)</th>
<th>std dev falling (ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>32(5)</td>
<td>101.0101</td>
<td>1.1709</td>
<td>1.3656</td>
</tr>
<tr>
<td>50(18)</td>
<td>363.6364</td>
<td>4.7945</td>
<td>5.5668</td>
</tr>
</tbody>
</table>

In the Figure 4.14, the upper waveform is the TCK and the bottom one represent the TDI; they are delayed by nearly 20 ns, correspondent to one system clock cycle or one state transition in this configuration with 8 frequency division and duty cycle of 50%

The test result signal corresponds perfectly to the expected shape (predicted with ModelSim simulation), meanings that the values of the real and expected TDO are aligned at the moment of the comparison. On the screen of the oscilloscope, are measurable some delays between the two signals, that analyzed statistically result in average equal to 1.4929 ns.

It is interesting to put in evidence that these misalignments are not affecting the
behavior of the internal check, because it is a block completely synchronized and enabled just when the valid signals are active. The delay is due to possible errors in the measurement instrument and overall to the way in which the two signal are routed from their position inside the FPGA to the respective pin on the PCI connector. In fact, they are surely mapped in different manner and so there is the probability that one path is longer than the other, introducing a small delays on the line.
The testing operation of the Miracle chip achieves high efficiency and potentiality with the use of a custom debug software as direct source of JTAG commands for the device. In the chapter is described this techniques and the steps done to reach a valid test policy.

5.1 Introduction

The testing phase with the inputs vectors is very useful in order to have a first direct verification of the functionality of each single processor through an FPGA, reproducing the EDA simulation on the real chip. A complete test of the entire multi-core chip is possible just with an interaction among the different processors. To reach this target, the first needed step consists in writing an executable program in the private RAM memory of the cores; the vector approach do not fit well the operation, first of all because of the limit on data that we can store in the FPGA. The objective is to use an host computer in order to send directly instructions and data to the chip, using an USB to JTAG interface. In other words there is the necessity of a software able to send valid commands to the processor (halt, reading and writing register, debug...) through the USB port. It has to work joined to a hardware debug adapter used to convert them in a JTAG format that can be interpreted by the correspondent module present in each layer of the Miracle. The software OpenOCD ( and the connector Amontec JTAGKey2) used for the test is one product actually in the market with a behavior correspondent to the requests explained in this paragraph.

5.2 OpenOCD description

OpenOCD is an open source software specific for the testing of System On Chip structures and allows fundamental operations on them:

- On-Chip Debugging;
- In-System Programming;
- Boundary-Scan Testing.

Embedded system testing not requires just a specific interface on chip for the access to all its components (obtained with the set up of a Boundary Scan methods), but also a new approach from the software point of view. In fact, it is inconvenient to run a debugger together with the one present on the same chip or even completely impossible if there is any software (or any debugger) support in the UUT. In this kind of verification, it is necessary to keep in account that the hardware
itself could be affected by errors; a faulty or untested memory element instantiated in the system compromise the correct behavior of any code, even a debugger.

The better solution for the embedded processors is to perform a remote checking: the debugger is running on an host computer and control the target both from the hardware side and from the software one, passing just part of instruction inside it, without run a full featured program of verification on the core.

All the functionality needed to exploit this method are supported by OpenOCD and the hardware adapter, from the interfacing in JTAG format to the internal debug. This software has been selected as the better technique to reach one of the first objectives of the Miracle testing: being able to download program code to the target system while this is at regime in order to create a reliable performances evaluation.

5.2.1 The Amontec JTAGKEY2

The device is a debug adapter or, more specifically, it is a small hardware module providing the right kind of electrical signaling to the target being debugged. These are required since the debug host (on which OpenOCD runs) won’t usually have native support for such signaling but needs, a connector from USB port to JTAG one, able to make the conversion between these two formats.

The core of the element is constituted by a multi-purpose chip produced by the Future Technology (FT2232) and designed in order to allow the communication between the USB and different serial and parallel ports. The functionality is possible thanks to the internal Multi-Protocol Synchronous Serial Engine (MPSSE), a module designed to interface efficiently with synchronous serial protocols such as JTAG and SPI Bus.

Both for the use of the Miracle_FPGA and of the ASIC prototype, it is necessary to bring one by one the needed signal from the converter to the correspondent pins of connector, according to the assignment made in the chip and the specifications of the pin-out configuration of the debug adapter, reported in the Figure 5.1.

![Figure 5.1: Schematic view of the pin-out assignment of the JTAG connector][23]
5.2 OpenOCD description

It is possible to notice the presence of these signals:

- the four fundamental JTAG wires: TCK, TMS, TDI and TDO;
- two optional reset signals: S(system)RST and TRST, used respectively to reset the interface from the board and to block the communication from the computer;
- a main V ref pin, with a voltage range of 1.35 V / 5 V, plus a second pin used as co-supply, in order to reach easily the fixed level of voltage;
- a main Ground pin, plus other eight used for the maintenance of the signal integrity.

5.2.2 Configuration overview

The OpenOCD software has a flexible and general structure and so has to be correctly configured according to the used interface, target boards and device under test. The setup is made with a configuration file in which are described all the details of the debug operation, following a pre-defined common scheme. The first step is the definition of the debug adapter used for the communication between the computer and the board:

```plaintext
interface ft2232
   ft2232_device_desc Amontec JTAGkey2
   ft2232_layout jtagkey
   ft2232_vid_pid 0x0403 0xcff8
```

The second step is the pre-configuration of the target, constituted by the so called "Default Value Boiler Plate Code", in which are defined a series of variable fundamental for the establishment of the communication: the name of the chip, the type of the endianness and the identification code of the TAP controller of each device under test. An example of code is present in the Appendix C

The system could present more than one TAP inside the Boundary Scan chain in the case of complex chips, with micro-controller, peripherals or multi-processors, as in the case of the Miracle. Configuration lines have to be established for each of them, following the design specifications or the response of the Auto-probe communication of OpenOCD. This last operative mode is a limited ability to look at the scan chain, doing a blind interrogation and then reporting the found TAPs.

In the system setup is also useful to add some specification about the speed of the JTAG communication, according to the physical limit of the debug adapter (30 MHz in the case of Amontec JTAGkey2) and the system clock of the system under test, that has to be several time larger. A particular importance has to be given to the reset configuration, because the communication depends in large part on it and the way in which it has to be set is very specific to the device under test. The whole system (OpenOCD, USB adapter, board, JTAG module and devices on chip) can support three type of reset:
- **System Reset (SRST)**: this hardware signal resets all chips connected to the JTAG adapter and normally behave exactly like pressing a reset button;

- **TAP controller Reset (TRST)**: this hardware signal resets just the TAPs connected to the JTAG adapter, remaining not visible to the rest of the system;

- **Emulation Reset**: is a software reset made through JTAG commands; these reset is often un-distinguishable from system resets, either explicitly or implicitly.

Generally speaking, the presence of a specific reset is completely optional in a JTAG communication, assuming that the target circuit is correctly ready to operate; in many case is however advisable to exploit one of the methodology presented, specifying it in the configuration file, as for example:

```
reset_config srst_only
reset_config trst_and_srst separate.
```

The reset present many other specifications: one of the more important consists in the manual setup of the timing properties of the reset events, as the delay of the signal de-assertation. With these commands is completed the basilar initialization of the communication between the adapter and the digital system; there is the need to define the TAP, allowing the identification from the OpenOCD software, with the use of a command as the following:

```
jtag newtap $_CHIPNAME cpu -irlen 4 -expected-id $_CPUTAPID
```

where the parameters:

- "cpu" defines the role inside the system as a part of a processor (alternatively it could be a TAP of a flash memory, a route controller, a trace buffer or a separate one);

- "irlen", representing the length in bits of the instruction register, such as 4 in the Leon processor case;

- "expected-id variable" allowing the check of the TAP identification number, defined as a variable at the beginning of the procedure.

The last part of the configuration involves the definition of the target, meaning the processor (or processors) related to the detected TAP. For this close relationship, it is a common convention to use the same name for both the controllers and their CPUs. There is the possibility to configure specific aspect of the target according to the properties of the device, but generally (as in the Miracle case) it is just sufficient to define its private area of work in the memory:

```
set_TNAME $_CHIPNAME.cpu
target create $_TNAME epfl -chain-position $_TNAME
$_TNAME configure -work-area-phys 0x60000000 -work-area-size 0x8000
```
5.3 Adaptation of OpenOCD commands

The address of the memory starting point can be the physical one or the virtual one, as done in the example above; the first case it is used when no MMU (Management Memory Unit) is present in the device and requires generally a direct specification about the size of the private memory. Is not actually the case of the Miracle chip, but in this space of configuration it is also fundamental to define the external locations, for example of a Flash memory, present in a lot of processor and in particular in micro-controllers.

5.3 Adaptation of OpenOCD commands

The software OpenOCD is designed specifically for ARM unit and other few types of devices and so is not fitting with the properties of the LEON III processor, central core of the Miracle project. The reason is that each family of CPU is characterized by a particular architecture, with its own instruction set (Sparc V8 in the case of the Gaisler unit), its specific TAP controller and way of communicating with JTAG modules.

It has been necessary to modify all the main important commands of the software, making adaptation to the LEON structure. The chosen design rule is to exploit the OpenOCD function already implemented, creating an external wrapper in substitution of the top level associated with the commands: the low level structure of the software is maintained not altered as much as possible.

Unfortunately, in order to use OpenOCD as a powerful, flexible and automatic testing tool, some functionalities have been completely re-designed, exploiting low JTAG function and the instruction set manual. For this reason, the approach of software testing requested a large preparatory work, based on the detailed study of the LEON processor architecture and of the software codes.

In the following paragraphs, the functions more used and that implied a larger effort of adaptation are described.

Epfl halt

The command is able to put the processor in Halt State, condition where it is not executing instructions, but can return to an active state essentially instantaneously. The "stand by" situation is preparatory to the execution of other OpenOCD commands, as the writing or reading from the private RAM of the processor.

In particular, with this specific function the processor can be put in Debug Mode, meaning that it can execute instructions one by one (after specific command from the user), in spite of the "hard" halt condition.

The target is halted controlling the value of some flag bits in the Debug System Unit (DSU) of each LEON III core present in the Miracle; in details, the steps needed to reach this objective are the following:

- set the Break on IU Watchpoint (BW) bit of the DSU Control register, in order to force a watchpoint trap;
- set the Break Now (BN0) bit in the DSU Break and Single Step register (BSSR): it force the processor into the debug mode if the watchpoint trap is already selected;
• verify the set of two read-only flags of the Control register, DSU enable (EE) and Debug Mode (DM).

If there is the necessity to produce an hard halt (not just Debug Mode), in the DSU Control register also the Processor halt (HL) bit has to be forced at 1.

Figure 5.2: Control (up) and Break Single Step (down) registers of the DSU

**Epfl step**
The processor can execute one instruction at time in Debug Mode in order to allow a specific check of its behavior. At each command "step" the CPU makes the operation present in the Program Counter, updating it with the content of the Next Program Counter register. The situation of all the processor’s registers is changed according to the executed instruction and so can be monitored in detail from the user.
The activation of the Single Step Mode is obtained with the configuration of the BSSR register in the DSU: the Break bit (BN0) has to be cleaned, instead the Single Step bit (SS0) has to be set to 1. This working mode is created just to activate the core for one clock cycle and so the command return automatically to the stand-by condition, re-selecting the BN flag.

**Epfl resume**
After the operations made in halt mode, as for example the writing of a binary in the memory, the processor has to return to work normally, executing continuously its instructions. The resume command exploits this purpose, using opposite procedures respect to the halt and Single Step starting point: the BW, BN0 and SS0 bits inside the Debug System Unit are unselected. The core is now in "running" mode.

**Epfl write memory**
The command is an adaptation of the correspondent default OpenOCD operation to the LEON III architecture. It is able to write in a memory address data composed of 8, 16, 32 bit or also more than one words, thanks to the count flag that allow the extension of the operation to the following location in the memory block. The structure of the function follows directly the behavior of the TAP controller Finite State Machine diagram (Figure 2.12), with the two separate and consecutive flows for instructions and data:

• write in the instruction register the base TAP code correspondent to the address selection (ASEL 0x08), in order to inform the logic that the next data present in the scan chain will be the address;
• shift in the scan chain a 3 bits sequence, special for the LEON TAP architecture, in which are defined the type of the operation (read - 0 / write - 1) and a code for the size;

• send to the chain buffer the location in which the writing has to be made;

• write the instruction register with the code related to the data selection (DSEL 0x09);

• insert in the chain the content that has to be written in the addressed memory location.

At the end of this set of operations, the scan chain buffer has to be executed with a call of a simple JTAG command (execute_queue), correspondent to the UPDATE state in the JTAG FSM.

Epfl read memory
The reading of a memory location works with the same base scheme of the writing operation for what concern the address selection. For the second part, no instructions or specific data have to be scanned into the chip: it is just sufficient to pump into the scan chain a set of zeros equal to the length of the content of the memory in order to obtain the requested value at the TDO of the system.

Epfl load
During the running mode, the processor is executing the instruction present in the RAM memory block and pointed by the Program Counter. Especially for a testing procedure, results very useful to verify the correct execution of a known program downloaded on the memory from an external source, overwriting the previous content.

The new executable code can be inserted from any position of the RAM, but it is more useful to start from the first address in order to have the maximum free space possible to write a complete test procedure.

This target is obtained with a quite complex sequence of steps and operations, that can be resumed in the following list:

• halt the processor, putting it in Debug Mode with the halt command, described at the beginning of the paragraph, allowing the access to the RAM private memory;

• open the source file of the executable code and copy the first instruction in the defined location (for the LEON in Miracle, it is useful to choose the starting point of the RAM, 0x60000000 - Appendix B);

• save the address of this location in a windowed register, as for example out1 or global1 with a SETHI instruction (Sparc V8); the instruction has to be repeated two times in order to be really executed at the moment of the next operation, solving a timing discrepancy between the management of the LEON III and the C code of the OpenOCD software;

• execute the two operations with a step command, moving the Program Counter further of a position;
• substitute the content of the location pointed by new Program Counter with the call of a jump (JMPL) command. In this call, the processor is invited to move to the position written in the general purpose register with the previous SETHI operations;

• substitute the Next Program Counter pointed location, with a NOP instruction, according to the default Sparc standard about the need of a null instruction following each call or jump instruction;

• execute the two operations with one "step" command, moving the Program Counter to the first instruction of the binary;

• copy all the code from the source file in the following lines, without any movement of the Program Counter (cycle of writings until the end of the binary);

• start to execute the program copied in the RAM and exit from the processor halt mode, with a resume command.

![Figure 5.3: Schematic of the epfl load command behaviour on the RAM](image)

**Epfl chain status**
The test of the multi-processor gains in efficiency if the cores are arranged in series, as happened in the bottom layer of the first Miracle prototype. In fact, in this case all the inputs (TDI) are sent through the first TAP controller of the chain and the output (TDO) is received from the last one; the commands as to reach each time specific different cores, thanks to the control of the TMS, TCK and TRST. For this reason these three signals are connected in parallel with all the elements of the chain.
In particular, in order to address one specific core for the binary load on the memory, it is necessary that all the other TAP controllers are put in bypass. As described in the section 2.7 about the theoretical aspects of the Boundary Scan Chain, when the bypass instruction is received, it is selected a one bit register instead of passing through all the flip-flop of the core (internal data register).

Especially in many-targets layouts, during the debug phase it is convenient to have an updated situation of the status of all the taps in the chain: this is the purpose of the "Epfl chain status". Combining low level functions of OpenOCD it is possible to detect the number of elements in the path and the properties of these. This can be achieved thanks to the fact that the software organize the TAPs in chain as a list of structs: passing from the current to the next one and reading the fields, it is possible to show the bypass flag of each of them (1 if bypassed).

**Epfl automated chain**

The use of targets arranged in a chain implies the need of a large effort in order to make a new adapting operation to all the functions previously described. Also in this case, in fact, the OpenOCD software supports this configuration just for a series of standard unit, but not the Leon. Moreover, the Gaisler processor has a classical architecture and it is not equipped with a JTAG Route controller, able to exclude for a defined time interval one or more TAP from the chain, avoiding the use of the bypass instructions.

So, it is necessary to control "manually" any operation of the debug, because the set of bits inserted into the chain changes its length according to the number of processors detected. For example, considering a one-core configuration, an address can be simply step into the chain, because its number of bits (32) is exactly equal to
the number of flip-flops units composing the internal data register. Instead, in a five-TAPs situation that length is increased to 160 bits, counting all the 32 flip-flops for each unit, and so the address in example has to be inserted in the right location (the 32 bits correspondent to the desired core) inside this large set of bits, filling all other positions with dummy values.

The idea exploited for the Miracle debug, as seen in the previous paragraph, is to address one targets at time, putting all the others in a bypass state: the length of the chain is theoretically equal to the internal register of the active TAP plus one bit for each of the inactive ones.

![Figure 5.6: Simple diagram of the scan chain adaptation](image)

Going back to the five-units arrangement example, any system operation is based on the definition of an instruction of 20 bits made by the concatenation of the standard 4 bits code each. The slot correspondent to the desired active core is filled with the operative instruction (extest, address selection, data selection...) and all the other bits are set with logic 1 in order to enabling the bypass (in fact, in SPARC the 0x01 instruction is related to the bypass).

The bits are shifted one by one inside the chain but they are considered just when the desired sequence is inserted and the TAP Finite State Machine passes to the UPDATE_IR state.

After checking the actual configuration with the home made function "EPFL chain status", it is necessary to provide the data code. Dummy values are added to the sequence in order to fill the bypass registers that precede or follow the active target. The special architecture of the Leon, not considered in the OpenOCD commands, impose to add three status bits before any data (see paragraph about custom read function), for a final total length of 39 bits for each set scanned into the chain. The use of large case statements allows to automatize the access to different cores, using the same commands and just specifying the name of the current unit targeted by the software. The efficiency of debug is also extremely enhanced by a scan chain approach that involves both the layers of the Miracle prototype, but it require the addition of an external digital module to manage the chain of TDIs and TDOs inside the FPGA.
Experimental validation

In the previous part of the report, are presented the studies, the researches and the hard practical attempts made in order to design a complete testing setup for a three dimensional structure, able to analyze the Miracle chip under several points of view and with different approaches. The huge obstacles found during this months have required a ri-scaling of the project timing line, but nevertheless it has been possible to start the validation of some aspects of the chip: all the details are shown in this chapter, organized according the various methods conceived.

6.1 Performances analysis and results on FPGA

Exploiting the particular adaptability of the LEON core to the implementation on the FPGA, it is possible to analyze the behavior of the chip and its performances before the testing phase on the ASIC prototype. In fact, a set of information and data obtained from the FPGA model of Miracle results fundamental in order to have a validation on the digital architecture of the design, independently from possible faults introduced during the technological fabrication. The definition of the implemented Miracle_FPGA is obtained after a series of structural modifications allowing the adaptation of the architecture to the specification of a design on a Xilinx board. The changes involve in particular the instantiation of the storage modules. On the used FPGA, the memory is organized in 148 blocks of predefined size (36 KB) and they have to fit with the requirements of a two-stacked-layer Miracle: 32 KB of shared memory per layer, plus 72 KB for each core, counting ROMs, RAMs and caches. The straightforward solution is the use memories from the UniSim

Figure 6.1: Testing instrumentation
library from Xilinx, directly prepared for the Virtex5 architecture, but a problem of resources obstructs that choice. In fact, they exploit a particular construction made with separated columns, that during the synthesis process are assigned to different blocks of 36 KB. In this way, it is required a huge number of slots, but the majority of them remains almost empty, occupied by just one column.

In order to use in a smarter way the FPGA resources, all the memories are re-builted using the standard Core Generator and the synthesizer is forced to use just one complete block for each of them. In addition, to save area, the ROMs size is reduced to just 2 KB, because it results enough to store a basic initialization code.

After the various preliminary tests and analysis realized implementing just some portions of the whole design (a core, a layer, a two layer structure without all peripherals...), the model described in the previous paragraph is chosen as the better solution in order to have the most efficient version of the chip on the board. Some parts of the FPGA model are already designed from the first session of the project, as an Interswitch module, in charge of blocking the TSVs according to the position of the switches on the board, in order to exclude the bottom layer and make a test just on the upper one.

The testing of the Miracle_FPGA is organized in order to exploit a first phase of generic functionality verification (realized with two different methods), followed by a second session in which are designed deeper and more focused benchmarks for the extraction of performances parameters. These two main domain of the validation are analyzes in the next sections.

6.1.1 Testing of basic functionality

The ultimate objective of the testing operation consist in the demonstration of the advantages in the adoption of a 3D design for a multi-core processor and in the detection of the weaker aspect of the structure in order to solve them in future prototypes. Nevertheless, this target is subordinated to the correct behavior of the processors arranged in the multi-layer architecture: therefore the functionality validation of each core plays a fundamental role and has to be defined carefully, in an exhaustive way.

In particular, it has been decided that this verification can not prescind from the analysis of the peculiar aspects:

- receiving an identification code from the TAP controller of the core, unavoidable step for the opening of a JTAG debug communication;
- reading back and checking of the IDCODE correctness;
- bypassing a core from a possible JTAG scan chain;
- reading and writing on any locations of the private RAM and of the shared memories;
- using the GPREG functionality, showing outputs on the board LEDs.

This goal is achieved with both the approaches described in the previous chapters: the hardware Timing Block, implemented on a second FPGA of support, and the
debug software OpenOCD, with its straight access to the digital architecture. The first solution represents the contact point with the Miracle design phase, because it consist in the direct transportation to a real level (FPGA board) of the device simulations realized in ModelSim on the post P&R design. The input vectors stored on the memories are the translation in JTAG format of a VHDL test-bench, in which are defined the bit-streams that has to be injected into the scan chain of one or more TAP controllers. The bypass verification is obtained scanning into the flip-flops the relative instruction (0xF for the Sparc V8 instruction set), followed by a random sequence of bits: if the JTAG module is working correctly, on the TDO of the system has to appear the same series of logic 0 and 1, just delayed of one TCK cycle correspondent to the presence of the one-bit bypass register in the chain. The access to the shared memories can be direct and easy also for an external debugger, though the NOC that interact with the Peripheral Sub-System, in which the shared RAM is contained, ensuring the exchange of data with the cores. Each LEON processor see the shared memory as an extension of its addressable space and can access directly to it, just subordinate to the semaphores control of the others cores in parallel. The memory map diagram is shown by the schematic representation in the Appendix B.

The verification of the functionality with the use of the Timing Block as interface is not immediate and quick; in fact, a very fast answer can be obtained looking at the 1-bit test output, result of the bit-a-bit comparison between the expected TDO and the one received by the Miracle instance. This represent a first marker of the good behavior, but to reach a deeper level of debug is necessary to use an oscilloscope with multiple channels and check directly the waveforms of the five JTAG signal, in order to discover the cohesion with the ones extracted from ModelSim. This disadvantage can be overcome choosing the second test approach with the use of the OpenOCD software, paying at the end just a larger theoretical distance from the ModelSim simulations: the results it has been possible after a very long work of adaptation of the debug tool to the LEON requirements, with the realization of several new functions (Chapter 5).

With this approach, all the desired parameters can be validated with a quick sequence of commands from the host computer and with an easy cross check on the screen. The software approach enables a step further respect to other methods: the download of a benchmark directly on the private RAM of the core under test, allowing to verify the work at regime of the processor in front of different types of situations. Any example of benchmark is compiled with a specific tool for LEON and Sparc architecture, translating it in the correspondent assembly and op-code commands that can be interpreted by the processor. The file allows a quite exhaustive verification of each core, showing the correctness of the results on the LEDs present on the board and connected to the GPREG register of the cores. After the charge of the binary in the RAM memory, an additional and efficient test could be the reading of all the written location, made with just one commands in which specify the number of words of content that have to be displayed by the software. Also the proof of the bypass condition can be demonstrated with the call of the custom function "epfl_status_chain", completely customized during the
Experimental validation

The following table shows a brief visual resume of the functional tests realized on the digital architecture of the Miracle:

<table>
<thead>
<tr>
<th>Entity</th>
<th>Verifications</th>
<th>Results Met.1</th>
<th>Results Met.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core0</td>
<td>read own IDcode</td>
<td>Passed</td>
<td>Passed</td>
</tr>
<tr>
<td></td>
<td>TAP bypass</td>
<td>Passed</td>
<td>Passed</td>
</tr>
<tr>
<td></td>
<td>R/W from private RAM</td>
<td>Passed</td>
<td>Passed</td>
</tr>
<tr>
<td></td>
<td>R/W from shared RAM</td>
<td>Passed</td>
<td>Passed</td>
</tr>
<tr>
<td>Core1</td>
<td>&quot; &quot;</td>
<td>Passed</td>
<td>Passed</td>
</tr>
<tr>
<td>Core2</td>
<td>&quot; &quot;</td>
<td>Passed</td>
<td>Passed</td>
</tr>
<tr>
<td>Core3</td>
<td>&quot; &quot;</td>
<td>Passed</td>
<td>Passed</td>
</tr>
<tr>
<td>CoreS</td>
<td>&quot; &quot;</td>
<td>Passed</td>
<td>Passed</td>
</tr>
<tr>
<td>CoreB</td>
<td>scan chain bottom layer</td>
<td>Passed</td>
<td>Passed</td>
</tr>
<tr>
<td></td>
<td>Binary download and execution</td>
<td>Not feasible</td>
<td>Passed</td>
</tr>
</tbody>
</table>

6.1.2 Testing of multi-core potentialities

A correct behavior of all the LEON processors during the first phase establishes a solid base from which creating more complex validations, involving the communication among the cores and between the two layers. In the Miracle design, in fact, the exchange of data among the different active unity can be managed through the shared memory blocks, in which all cores have access and the same rights of reading and writing. The usage of the common addressable space should not be completely open, but must be regulated by a complex system of semaphores. This strategy consists in the implementation of digital modules able to manage the requests from the processor to the memory: if two or more core want to simultaneously read or write the same location, the semaphores assign a round robin priority to them, avoiding collisions and consequently the forwarding of faulty bits on the bus of the device. Exploiting the safety of that hardware protection, it is removed a significant possible source of errors in the behavior of the multi-core system, but it remains the necessity to control the timing under the software point of views. Operations involving the participation of more processors have to fit perfectly together and be synchronized intrinsically.

In fact, an additional difficulty in the Miracle architecture comes from its nature of mega-processor composed by single instance of cores, completely equal but separated and invisible to each other, except for the only common memory resource. This property is reflected in the impossibility to create a software benchmarks in which there are groups of specific instructions for each core and visible just to the relative one.

The only possible solution is to synchronize the software execution internally to each core, with the support of the shared memory, in which is written a set of flags indicating the correct conclusion of any relevant operation made. Any core is
polling the common resource in order to evaluate if the threshold values are already reached. This approach allows to solve two main problems of parallel execution:

- the starting "alignment": for example, each processor increment a common flag when they are ready to begin the execution, waiting for the assertion of all the others, before any computation;

- the enhancement of the performances: for example, each output operation on the shared area could be linked with its own flag, that can be "busy", if it is not yet concluded, or "done", if it is finished.Receiving these informations during the polling, all the other cores skip the operation and calculate the next one, speeding up the computational time.

The strategy described in the previous paragraph is the only one that allows to reach the final objective of the FPGA testing setup: the realization of a complete benchmark in which all cores of the Miracle execute the same procedure in order to exploit the parallelism and the increase of the bandwidth. In this way, it is possible to realize custom architecture based on the LEON, allowing interesting and fundamental estimation and balancing between advantages and disadvantages. In particular, the validation methodology prepared during the months of project has been conceived for comparing the response of the same procedure in the case of one, four and eight Gaisler cores, placing side by side the performances of a single processor, a Miracle layer and the entire device.

The analysis, indeed, has been designed in order to evaluate the following parameters:

- execution time $T$ of a benchmark on a single core, expressed in milliseconds or number of cycles required;

- factor $N_{\text{core}}$ correspondent to the speeding up of the time with a $N$ cores configuration respect to previous case;

- factor $N_{\text{layer}}$ representative of the increase of the performances with a more layers architecture;

- factor $R = N_{\text{layer}} / N_{\text{core}}$, in order to have an estimation of the change due to the intra-layer logic.

A perfect response from the Miracle on FPGA is reached if the $N_{\text{core}}$ parameter is more or less equal to $N$, meanings maximum advantage from the parallel configuration, and if the $N_{\text{layer}}$ is not so lower than $N_{\text{core}}$ in the complete chip architecture, representing the benefits of a vertical integration of the layers and consecrate the validity of the 3D processor prototype.

An algorithm of any type could play the role of the benchmark on which the execution performance of the Miracle are evaluated; many attempts are possible, but during the testing setup design has been chosen a matrix approach for the calculation made by the processor. Starting from local values, the cores compute a matrix of defined size and write it on the shared memory; after the end of this phase, they execute operations on the input matrix, producing an output matrix of the same
Experimental validation

dimension with the final results. Both the two data structures are duplicate in order to create a dual memory space for the relative flags necessary for the parallel execution. Unfortunately, the testing with this kind of algorithm encounters problems in the interaction with the LEON processor: the impossibility to use an unique and equal code for all the cores has stranded the application of the validation model, forcing to investigate in many domains to find a possible solution. This issue has been demonstrated really hard to solve because of a partial lack of coherence in the appearance of the problem in different type of code. Debugging the procedure with step by step operations, it seems that the LEON falls into a trap exception during the execution of some particular "store" instructions present in the assembler code downloaded in the private memory. Any times this problem come out, the program counter jumps into the ROM searching a procedure able to manage the traps (not present in this arrangement of the LEON) and the DSU flag signal of the error state of the processor is set). A really interesting point is that the system could execute many times a possible incriminate line of code (for example, if it is in a loop) but it is blocked just at a fixed moment (always the same for one specific binary). Several attempts of adapting the C source and deep researches on many different fields have lead to the formulation of some hypothesis on the possible causes:

- **Compiler**: the benchmark in C language is compiled with a tool specific for the LEON architecture because it has to create an assembler code and then an opcode sequence, that can be interpreted by the Sparc V8 model. Some settings in the compiler can be wrong, considering the fact that modification of some parameters can change the presence and the frequency of the errors. For example, with a optimized compilation the number of store operation is dramatically reduced, obtaining a code not sensible to those traps. Unfortunately, the extremely high degree of compaction causes also the loss of some important instructions that the compiler considers negligible because it is not aware that the LEON is inserted in a multi-processors dynamic system.

- **Linker**: the linker that effectively produce the binary file for the RAM is adapted to the custom architecture of the LEON used inside the Miracle and so could be a possible source of errors; instead of the hypothesis it seems to originate the files according to the right address space and with the correct disposition of the different section (text, data, comments, initialized variables...).

- **Sparc V8**: there could be an incompatibility with the normal execution of the instructions and the sequence of op-codes, especially with the order and the frequency of the storing operations on some particular registers of the Integer Unit; in particular, it can be that the benchmark is saving some local variables in positions, randomly and automatically shared also with the stack pointer.

- **LEON**: it could exist some kind of mismatch with the reference type of memory content that a Gaisler processor is expecting. A possible confirmation for this cause or the previous one is due to the apparently similar behavior of the Miracle during ModelSim simulations, done with the exploitation of the
created binary as ROM or RAM content. It is difficult to impute the cause to the use of the shared memory, because the same structure of instructions is faulty also if it is addressed to the private resources.

- **OpenOCD**: the adaptation of the software involves a lot of functionalities but some low level instruction could be not perfectly customized to the processor in use: for example, the volatile registers of the Integer Unit are not easy to be read. However, the presence of undetermined behavior in the RTL model of the cores, lead to exclude OpenOCD from the cause, including it rather in the faulty consequences of a different problem.

Solving this problem is a condition of primary importance for the prosecution of the testing process: without the application of a benchmark of this type, the only one theoretically efficient for the Miracle architecture, became impossible to extract a complete performances analysis, able to confirm the improvements of a multi-layers 3D processor. Finding the cause of the trap exception in this LEON configuration constitute the first fundamental further development in the finalization of the testing.

**6.2 Test on packaged Miracle layer**

The first step made on the real chip is the functional verification of the single die in the relative package (Figure 6.2), using a PCB for the interface between the Miracle and the external world (FPGA or directly the software on the computer): this validation is fundamental in order to exclude the faulty dies from the successive operations. When this phase is accomplished, the test can move to the naked dies, that need the use of a specific Probe Card and a second PCB board in order to fit completely inside the test apparatus for this step in the testing procedure.

These two validations are fundamental in order to apply on known good dies the last processes in clean-room: TSV formation, grinding of the top wafer and bonding of the two layer. Unfortunately, the huge obstacles encountered during the adaptation of the OpenOCD software have expanded the expected timing line, preventing the possibility yo insert interesting results about this final procedures in this report.

**6.2.1 PCB design and verification**

The flexibility of test environment and the various alternatives that it intend to offer have to influence also the design of the interfaces with the ASIC Miracle prototype. So, particular attention has been devoted to the realization of the board,
Experimental validation

in order to enable a complete verification of the functionalities in different context (directly on board or exploiting the FPGA potentiality) and conditions (frequency sweep, type of test).

The PCB in Figure 6.3 allows to test the packaged chip establishing a direct connection between the OpenOCD software, running on a computer through the JTAG connectors; this method directly tests the basic functionality of each core, as the possibility to read and write correctly its own private memory, to control output peripherals and to execute programs downloaded in the RAMs.

![Figure 6.3: PCB for testing packaged Miracle layer](image)

As already seen in various points of this thesis, the testing operation could be extremely more efficient if all the processors inside both Miracle layers are arranged forming one unique scan chain, gaining in time and automation; that objective can be reached with the instantiation on the FPGA of a digital module able to manage the TDO signals arrived from the chip and to re-direct them to the input of the next core in the chain. The FPGA could be used also for the Miracle start-up validation with the implemented hardware system for the re-timing of stored input vectors: the digital design takes care of the sending of the JTAG signals (TCK, TRST, TMS, TDI) and of the comparison between the received TDO and the expected values used as references.

Both these approaches require to double the input pins, PCI connectors joined to the JTAG ones. In addition, it is mandatory to have a set of switches enabling the swapping of the input between the two sources: in particular, control signals of
Miracle (as the reset, the layer identification number, the PLL enable...) can be generated on the board during a direct computer-chip test or on the Xilinx Virtex5, if an indirect mode has been chosen. According to the various phases of verification, before and after the bonding process, it has been required the design of two twins board, different just for a connection between each other: one allows the verification of the packaged devices and the other is able to connect the FPGA to the probe card necessary to handle the naked chips.

The testing policy is based on the assumption that all the infrastructures designed and prepared for the checking are correctly working and are not introducing any source of faults to the system: in order to assert this situation, it is fundamental to establish a full and comprehensive verification of the functionality of the PCB. In particular, it is very interesting to define a sequence of preliminary steps, that can be applied to the board without the chip and also repeated at the beginning of the test of any Miracle die.

The first check is the power supply analysis to assure that the correct voltages and currents arrive to the chip handler on the board, the socket. The system is supplied with 5V and some hundreds of milliAmpere, but two DC/DC converters are used to obtain the necessary voltage level for the Miracle: a 2.5 V reference for the I/O rings and a 1.2 V for the processor’s logic.

<table>
<thead>
<tr>
<th>Domain</th>
<th>Verification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power supply</td>
<td>external source: 5 V</td>
</tr>
<tr>
<td></td>
<td>I/O voltage : 2.5 V</td>
</tr>
<tr>
<td></td>
<td>chip reference : 1.2 V</td>
</tr>
<tr>
<td>Clock source</td>
<td>external signal</td>
</tr>
<tr>
<td></td>
<td>quartz oscillator : 100 MHz</td>
</tr>
<tr>
<td></td>
<td>FPGA clock (oscillator or PLL)</td>
</tr>
<tr>
<td>Input signals</td>
<td>JTAG and PCI connectors</td>
</tr>
<tr>
<td></td>
<td>Switches</td>
</tr>
<tr>
<td></td>
<td>PCB lines</td>
</tr>
<tr>
<td>Socket layout</td>
<td>signaling board-socket</td>
</tr>
<tr>
<td></td>
<td>signaling socket-Miracle</td>
</tr>
</tbody>
</table>

Providing a coherent clock source is the second unavoidable step in the testing setup; in this case, it is necessary to check the integrity of three possible signals among which a multiplexer can choose according to the requirements:

- a fixed clock signal derived from a quartz oscillator at 100 MHz;
- a clock arriving from the FPGA unit, received directly from an oscillator or transformed in frequency with an implemented PLL;
- an external signal provided with a coaxial cable and very useful in order to tune the test according to a frequency sweep.
Experimental validation

To prove the correct behavior of the chip, it is also fundamental that it receive the right inputs signals and so during the testing flow it is necessary to check if they arrive to the sockets from the connectors and the switches. The last preliminary elements that has to be verified is the pin layout of the socket on the board, ensuring not just the integrity of the signals provided to the chip but also the perfect correspondence of roles among socket, package and Miracle layer pins. This step requires surely a particular care in order to avoid the presence of some mismatch: a simple error in the design phase can compromise completely the behavior of a chip, if it is not detected in the preliminary phase.

The board checking flow is also summarized in the Table 6.2.

6.2.2 Chip testing results

The validation of the PCB behavior allows to move to a step further in the expected procedure: the test on the packaged chip has been started following the software OpenOCD defined approach. Hence, the same set of commands used for the Miracle_FPGA verification is introduced for the functional validation of the real Miracle, obtaining correct and satisfying outputs. In fact, it has been possible to establish the communication with all the cores present in the layer, checking the IDCODE scanned out from the TDO. Also the reading and writing of any memory locations gives reliable positive answers.

The LEON unit can be put in Halt State for the downloading of a binary routine on its own private RAM and then correctly resumed in order to execute the new program: expected outputs can be obtained reading specific locations written by the processor as flags.

The interaction with the shared resources has not answer positively to the first trials and so its verification is still under development.

Table 6.3: Test results on packaged chip

<table>
<thead>
<tr>
<th>Entity</th>
<th>Verifications</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single core</td>
<td>read own IDcode</td>
<td>Passed</td>
</tr>
<tr>
<td></td>
<td>TAP bypass</td>
<td>Passed</td>
</tr>
<tr>
<td></td>
<td>R/W from private RAM</td>
<td>Passed</td>
</tr>
<tr>
<td></td>
<td>R/W from shared RAM</td>
<td>Still in development</td>
</tr>
<tr>
<td></td>
<td>Binary download and execution</td>
<td>Passed</td>
</tr>
</tbody>
</table>
The thesis work tried to embrace all the aspects concerning the design of an infrastructure for the testing of a 3D integrated processor, in order to propose a fully compatible solution that maintains also a high flexibility. This goal is achievable thanks to the use of the chip as integrating part of its own testing system, implying the necessity to understand deeply its architecture and the tools chosen for its debugging step. In the software approach, in particular, an extensive adaptation of the open-source OpenOCD tool to the Miracle requirements have been necessary, obtaining relevant results. This method, however, faces with the lack of information and documentation of the debug tool and of the Miracle prototype. Regarding the software, it has been often mandatory to directly study at C-code level in order to be aware of the real behavior and be able to apply modifications, implementing or changing the desired functions. Also a sufficient control on the LEON III architecture has implied a large preliminary examination, due to the fact that this master thesis project was the first deep approach with the structure of a processor (especially in a multi-core layout). The other approach is the "hardware solution", realized using ModelSim test vectors as input for the verification; it is able to offer good basic results, but paying an high effort in the design flow and a not negligible limitation in the sizes of the test procedure: it is significantly less efficient than the software technique. Nevertheless, both methods lead with different performances to the realization of a complete testing procedure, that results designed carefully and following precise and coherent rules. In fact, the validation of all the testing infrastructure and in particular the verification of the functionality of the chip model on FPGA have turned out successfully. The results are promising and encouraging, but still incomplete because the performance extraction, made with a designed benchmark, presents some unsolved incompatibilities with the device. Hypothetical sources of error have been highlighted by debugging the system in several different configurations, but no one of them seems to be the real cause of the issue. The time spent in avoiding possible mismatch between the test infrastructure and the unit have delayed the successive planned phases and for this reason it is not yet possible to furnish data and results on the final stacked chip. In any case, the development of this project evidences the importance of the testing phase in the realization of a new device; in some cases, it requires efforts equal to the ones spent for the design of the chip itself.
7.1 Future work

The future work is based on precise demand of overcoming the issues that have delayed the testing of the prototype. First of all, further experiments must be carried out to characterize the performances of the device and to demonstrate the real potentiality of a vertically stacked multi-processor, with respect to existent planar solutions. It would be interesting to finally fabricate the TSVs on the Miracle dies and testing them to discovering the yield of the effective clean-room process and the changes on the performances of the system. The physical realization and testing of the last process can lead to the presentation of the first prototype and to a further development of the device design. Improvements can be done also in the testing infrastructure that is already working properly; concentrating all the efforts on the software approach it could be possible to reach a more automated, efficient and complete procedure of validation.
Nomenclature

ALU  Arithmetic Logic Unit
ASIC Application Specific Integrated Circuit
ATE  Automatic Test Equipment
B2B  Back to Back bonding
BEOL Back-End-Of-Line
BSC  Boundary Scan Cell
BSSR Break Signle Step Register
CMOS Complementary Metal Oxide Semiconductor
CPU  Central Processing Unit
CR   Control Register
D2D  Die to Die bonding
D2W  Die to Wafer bonding
DRIE Deep Re-active Ion Etching
DSU  Debug System Unit
EDK  Embedded Development Kit
EPFL École polytechnique fédérale de Lausanne
F2B  Face to Back bonding
FEOL Front-End-Of-Line
FPGA Field Programmable Gate Array
FPU  Floating Point Unit
FSM  Finite State Machine
GRLIB Gaisler Research LIBrary
IC   Integrated Circuit
ID   Instruction Decode
IO   Input/Output
IP   Intellectual Property
JTAG Joint Test Action Group
MMU  Memory Management Unit
MSB  Most Significant Byte
NDI  Normal Data Input
NDO  Normal Data Output
NOC  Network On Chip
PC   Program Counter
PCB  Printed circuit board
PLL  Phase Locked Loop
PoP  Package on package
PROM Programmable Read Only Memory
RAM  Random Access Memory
RISC Reduced Instruction Set Computer
ROM  Read Only Memory
RST  Reset
RTL  Register Transfer Level
SIP  System on package
SOC  System On Chip
SRAM Static Random Access Memory
UART Universal Asynchronous Receiver/Transmitter
USB  Universal Serial Bus
TAP  Test Access Port
TCK  Test Clock
TDI  Test Data Input
TDO  Test Data Output
TMS  Test Mode Selection
TSV  Through Silicon Vias
VHDL Very High Speed Integrated Circuit Hardware Description Language
W2W Wafer to Wafer bonding
XPS  Xilinx Platform Studio
Bibliography


http://www.itrs.net/.


*Asynchronous & Synchronous Reset Design Techniques - Part Deux.*

[18] Xilinx Inc.  


[22] Prof. Atieza. *Co-design of Systems-on-Chip on Reconfigurable Hardware. - Lab.3.*  
EPFL, 2011.


[24] Aeroflex Gaisler. ..  
Table of the fabrication process of a SuperContact (Chapter 2).

Figure A.1: First phases of the process
Figure A.2: Second phase of the process
Memory Map

Schematic of the Miracle memory map for each core.

Figure B.1: Memory Map
Configuration file for OpenOCD (complete scan chain)

```c
# configure adapter
interface ft2232
ft2232_device_desc "Amontec_JTAGkey-2"
ft2232_layout jtagkey
ft2232_vid_pid 0x0403 0xcff8

# set the name of the complete target chip
set _CHIPNAME epfl

# this defaults to a big END
set _END little

# TAP expected IDCODE
set _CPUTAPIDS 0x14c611a1
set _CPUTAPID3 0x14c591a1
set _CPUTAPID2 0x14c511a1
set _CPUTAPID1 0x14c491a1
set _CPUTAPID0 0x14c411a1
set _CPUTAPIDUP0 0x14c011a1
set _CPUTAPIDUP1 0x14c091a1
set _CPUTAPIDUP2 0x14c111a1
set _CPUTAPIDUP3 0x14c191a1
set _CPUTAPIDUPS 0x14c211a1

# jtag speed
jtag_khz 2000

# has only system reset
reset_config srst_only

# scan_chain of TAPs
jtag newtap $_CHIPNAME cpuS -irlen 4 -expected-id $_CPUTAPIDS
jtag newtap $_CHIPNAME cpu3 -irlen 4 -expected-id $_CPUTAPID3
jtag newtap $_CHIPNAME cpu2 -irlen 4 -expected-id $_CPUTAPID2
jtag newtap $_CHIPNAME cpu1 -irlen 4 -expected-id $_CPUTAPID1
jtag newtap $_CHIPNAME cpu0 -irlen 4 -expected-id $_CPUTAPID0
jtag newtap $_CHIPNAME cpupS -irlen 4 -expected-id $_CPUTAPIDUPS
jtag newtap $_CHIPNAME cpup3 -irlen 4 -expected-id $_CPUTAPIDUP3
jtag newtap $_CHIPNAME cpup2 -irlen 4 -expected-id $_CPUTAPIDUP2
jtag newtap $_CHIPNAME cpup1 -irlen 4 -expected-id $_CPUTAPIDUP1
jtag newtap $_CHIPNAME cpu5 -irlen 4 -expected-id $_CPUTAPIDUP0
```
#target S
set $_TN_S $_CHIPNAME.cpuS
target create $_TN_S epfl -END $_END -chain-position $_TN_S
$_TN_S configure -work-area-phys 0x41000000 -work-area-size 0x8000

#target 3
set $_TN_3 $_CHIPNAME.cpu3
target create $_TN_3 epfl -END $_END -chain-position $_TN_3
$_TN_3 configure -work-area-phys 0x60000000 -work-area-size 0x8000

#target 2
set $_TN_2 $_CHIPNAME.cpu2
target create $_TN_2 epfl -END $_END -chain-position $_TN_2
$_TN_2 configure -work-area-phys 0x60000000 -work-area-size 0x8000

#target 1
set $_TN_1 $_CHIPNAME.cpu1
target create $_TN_1 epfl -END $_END -chain-position $_TN_1
$_TN_1 configure -work-area-phys 0x60000000 -work-area-size 0x8000

#target 0
set $_TN_0 $_CHIPNAME.cpu0
target create $_TN_0 epfl -END $_END -chain-position $_TN_0
$_TN_0 configure -work-area-phys 0x60000000 -work-area-size 0x8000

#target UPS
set $_TN_UPS $_CHIPNAME.cpuS
target create $_TN_UPS epfl -END $_END -chain-position $_TN_UPS
$_TN_UPS configure -work-area-phys 0x41000000 -work-area-size 0x8000

#target UP3
set $_TN_UP3 $_CHIPNAME.cpuS
target create $_TN_UP3 epfl -END $_END -chain-position $_TN_UP3
$_TN_UP3 configure -work-area-phys 0x60000000 -work-area-size 0x8000

#target UP2
set $_TN_UP2 $_CHIPNAME.cpuS
target create $_TN_UP2 epfl -END $_END -chain-position $_TN_UP2
$_TN_UP2 configure -work-area-phys 0x60000000 -work-area-size 0x8000

#target UP1
set $_TN_UP1 $_CHIPNAME.cpuS
target create $_TN_UP1 epfl -END $_END -chain-position $_TN_UP1
$_TN_UP1 configure -work-area-phys 0x60000000 -work-area-size 0x8000

#target UP0
set $_TN_UP0 $_CHIPNAME.cpuS
target create $_TN_UP0 epfl -END $_END -chain-position $_TN_UP0
$_TN_UP0 configure -work-area-phys 0x60000000 -work-area-size 0x8000