FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Study and Implementation of an Optimized Modular Architecture for an ATE Test Environment

Diogo Ferreira Oliveira Santos

Master in Electrical and Computers Engineering

Supervisor: Dr. João Paulo de Castro Canas Ferreira

External Supervisor: Eng. Mara Carvalho

External Co-Supervisor: Eng. Domingos Terra

November 26, 2020

Abstract

The semiconductor business has been growing at an enormous speed, which drives the need to reduce chip sizes and power consumption, while increasing their performance. But more important, the reduction of every cost associated with it.
The most time consuming step of the development is surely the verification of the chip, before and after its fabrication, in order to ensure that all the chips delivered to the customer are functioning correctly.

This work was done in partnership with Synopsys, with the objective of optimizing its SerDes ATE environment. After the study of the digital design and the current ATE environment, a solution was proposed, and so, in this master thesis, a modular architecture capable of integrating different Synopsys products in its SerDes family is presented.

By today design standards, it is common for an IC (integrated circuit) to have external pins meant for serial communication, which is the case of Synopsys SerDes, since pin count is limited. Considering that the testing of the chip is the most time consuming phase of the development of an IC, the proposed architecture takes advantage of the internal parallel interface of the SerDes to reduce chip access configuration procedures by having the chip self configure after the external interface commands it to do so, instead of using the serial interface to configure, register by register.

The architecture is modular taking in account that it can be adapted to different products in the same family with relative ease.

Resumo

O mercado de semicondutores tem crescido a uma velocidade enorme, o que leva à necessidade de reduzir tamanhos e consumo de energia dos chips, aumentando o desempenho dos mesmos. Mas mais importante, a redução de todos os custos associados ao seu fabrico.
A etapa mais demorada do desenvolvimento é certamente a verificação do chip, antes e depois de sua fabricação, para garantir que todos os chips entregues ao cliente estão em perfeito funcionamento.

Este trabalho foi realizado em parceria com a Synopsys, com o objetivo de otimizar o ambiente ATE para o seu SerDes. Após o estudo do design digital e do ambiente ATE atual, foi proposta uma solução, e sendo assim, nesta dissertação de mestrado, é apresentada uma arquitetura modular capaz de integrar diferentes produtos da Synopsys na família SerDes.

Pelos padrões atuais de design, é comum que um CI (circuito integrado) tenha pinos externos destinados à comunicação série, que é o caso do SerDes da Synopsys, já que o núemro de pinos é limitado. Considerando que a verificação é a fase mais demorada do desenvolvimento de um CI , a arquitetura proposta aproveita a interface paralela interna do SerDes para reduzir os procedimentos de configuração de acesso ao chip, fazendo com que o chip tenha capacidade de se configurar depois da interface externa o comandar, em vez de usar a interface série para o fazer, registo a registo.

A arquitetura é modular tendo em consideração que pode ser adaptada a diferentes produtos da mesma família com relativa facilidade.

Acknowledgements

Foremost, I cannot express enough grattitude to Eng. Mara Carvalho and Eng. Domingos Terra, my supervisors at Synopsys, for all the help and support throughout this work. I'd also like to thank all of Eng. Luís Cruz team and Prof. João Canas Ferreira, my supervisor at FEUP.
Secondly, the friends I have made along the way, thank you for the motivation and knowledge we've shared.
Lastly, but not less important, my family. My parents José and Paula and my little sister Carolina. Thank you for your irreplaceable support.

"Peace can only come as a natural consequence of universal enlightenment and merging of races, and we are still far from this blissful realization."

Nikola Tesla

Contents

List of Figures

Abbreviations

ATE	Automatic Test Equipment
BIST	Built in Self Test
CPU	Central Processing Unit
DUT	Device Under Test
DR	Data Register
FSM	Finite State Machine
HDL	Hardware Description Language
HDVL	Hardware Description and Verification Language
IC	Integrated Circuit
IR	Instruction Register
I/O	Input/Output
PHY	Physical Layer
RTL	Register Transfer Level
RX	Receiver
SerDes	Serializer/Deserializer
SV	SystemVerilog
TAP	Test Access Port
TX	Transmitter

Chapter 1

Introduction

1.1 Context

In today's world, ICs (Integrated Circuits) are everywhere. There is currently an immeasurable number of ICs and each year are made billions of them , with more and more being manufactured daily. This manufacturing is confronted with many challenges, but one common objective, which is to make sure we do not deliver a faulty IC to the customer. It is therefore extremely important that these ICs be tested before being handled to the consumer. Since time equates to money, each microsecond counts and reducing the time it takes to verify the correct behavior of a single IC is crucial.

The fact that we can manufacture integrated circuits with an enormous number of transistors exchanging at gigahertz clock rates and containing numerous I/O (Input/Output) terminals running at multigigabit rates is fascinating. It is even more impressive that we can ensure the correct behavior of nearly all of them. As such, validation and testing must be number one priority, which creates a demand for time, since we need to ensure the product is delivered without problems to the client, carrying out the deadlines.

1.2 Objectives

This dissertation presents as its main objective the optimization of an ATE test environment for Synopsys Ser/Des (Serializer/Deserializer) - DesignWare® Multi-Protocol 10G PHY. A detailed explanation of this product is done in chapter 3. For this, an HDL (Hardware Description Language) has to be studied (Verilog/SystemVerilog will be used, since it's the main HDL used at Synopsys), as well as the SerDes itself, in order to understand the internal design of the product and the current ATE environment. The main goals of this thesis is to reduce the time it takes for each chip to be tested (by reducing the overall test time), and the optimization of its test sequences. This will allow a faster verification of the chips in the simulation phase and also in the silicon and post production phase.

Chapter 2

Technical Background Review

2.1 The IC - From Design to Packaging

This section briefly summarizes an IC manufacturing process from design to Packaging, since it deviates from the main focus of the thesis. This thesis mainly focuses in the sixth - *Physical Verification* and ninth - *Die Test* steps of the following list of design steps.

Design Steps

1. System Specification
2. System Level Design
3. Logic Design
4. Circuit Design
5. Physical Design
6. Physical Verification and Signoff
7. Wafer Fabrication
8. Packaging
9. Chip Deployment

2.1.1 System specification

This is the inital step where the customer enumerates the desired functionalities of the IC. It is then translated to a more detailed view of the problems the IC has to deal with, which leads to a feasibility and die area study.

2.1.2 System Level Design

After consolidating the specification, it's now time for the System Design. The specification is converted to a modular architecture composed by blocks in order to simplify the architecture.

2.1.3 Logic Design

In this design step, the functional part of the circuit is implemented by the digital and the analog team. The digital team is responsible for the digital design and mainly codes the circuit in a Hardware Description Language, such as Verilog or VHDL, whereas the analog team for the analog design and corresponding layout. These two parts are simulated individually and if everything is working correctly, a system simulation and verification is performed.

2.1.4 Circuit Design

After the functional simulation and verification, the next step is to synthesize the digital design which means that the digital blocks are translated to logic cells. It's also in this step where the engineers add testability techniques in the design. This is called Design for Testability and its objective is to make the tests faster and easier and/or allow the IC to test itself.

2.1.5 Physical Design

This is the last step of the design itself. It is composed by Floor Planning, Place and Route and Parasitic Extraction. The first sub step is essentially the study of the available area and some components are placed, such as bonding pads which connect the IC to the outside. Place and Route, as the name implies, is the step where the individual blocks which compose the IC are placed in the previously defined positions and, at the same time, the interconnects between all the blocks are made. In a final step, it is performed a Parasitic Extraction, which is the calculation of all parasitic effects in the design and its interconnections. This extraction is important to the next step analysis.

2.1.6 Physical Verification and Signoff

Before sending the IC to be manufactured in the foundry, a physical verification must be performed in order to check if the chip will actually work in a real world situation. With all the physical design done and parameters extracted, an almost ideal simulation will be executed. This simulation is really close to the real world scenario, where it accounts for the electrical and logical functionality of the IC. After the correct behaviour is guaranteed, the photomask (essentially a mold for the photolithography process) of the IC is sent to the manufacturer to initiate production.

2.1.7 Wafer Fabrication

Wafer Fabrication or Semiconductor Device Fabrication is the actual physical process where the circuits are created on a wafer, also called substrate, which is a slice of a semiconductor material

(usually silicon). This is a very long process with innumerous steps which can take up to 15 weeks with modern technology nodes. Before finalizing this process and individualize each die (process called Wafer Dicing), there's a special equipment - a Wafer Prober - that will perform a primary test.

2.1.7.1 Wafer Level Testing

Wafer level testing occurs near the end of the Wafer Fabrication process and before the Wafer Dicing. It is done right in the wafer and checks for functional defects of the device using a Wafer Prober (a type of ATE, explained in the next section). This step is one of the most expensive, since a Wafer Prober is a really expensive machine and, sometimes, wafer level testing is ignored if the testing at the package level compensates this step cost.

2.1.8 Packaging

After the dies are ready, they need to be packaged. A die can't be handled without a package, not only because it's very fragile, but also because it needs contact pins in order to be soldered or simply fitted onto a PCB. There are lots of package types and they all have their own advantages and disadvantages, but the package type is conditioned by different factors, such as pin count and/or how it will be mounted on the PCB (some are not even used on a PCB, they are simply designed for developing purposes on breadboards). After all this is finished, the IC is put through a last verification test.

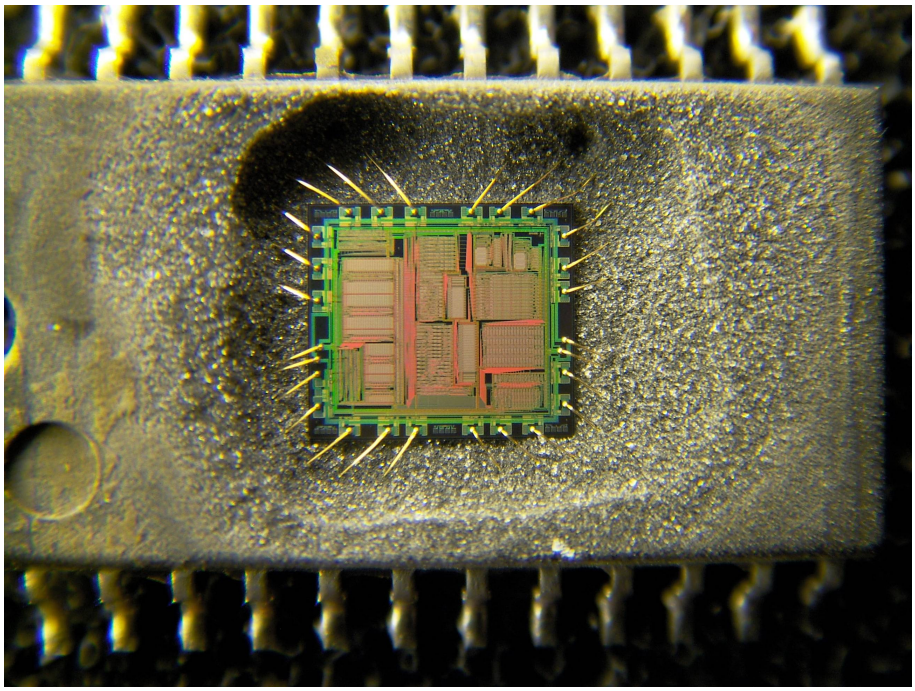

Figure 2.1: Die inside a DIP (dual in-line package)

2.1.8.1 Package Level Testing

After packaging, the IC is tested one last time before being sent to the client. Again, this phase of the process performs a complete test in order to ensure the client does not receive a faulty IC. Such test is done by an ATE.

2.1.9 Chip Deployment

Chip Deployment phase is used to analyze the aftermath of the production. An IC characterization procedure is done in order to improve the design for a next generation chip. The failures and the yield is also studied. In this phase, an IC Characterization is also done; Every time time a new IC, a modification of an existing one, a new wafer fabrication process or any new technology is developed, a characterization procedure is fundamental. This is an extremely important step of the design, since it analyzes the electrical performance of the ICs and might provide valuable information such as yield and minimum and maximum operating voltages/frequencies. IC characterization shows that devices are (or not) able to perform their targeted functions according to the specifications [2]. For an IC characterization to be performed, a batch of ICs, randomly selected from different wafers after the fabrication process, is subject to a series of tests which are expected to come across potential problems during early development , reducing both time and cost [2].

2.2 Yield

Yield is an important measure of the quality of a semiconductor process. Although the perfect semiconductor manufacturing process would produce only working dies, there are always defects during manufacturing which results in ICs which do not work or operate within specs.

Yield is therefore the fraction of the yielding wafers that are not discarded during the manufacturing process.

$$Y = \frac{Ngood}{Ntotal}$$

where:

Y = Yield
$Ngood$ = Number of good dies
$Ntotal$ = Total number of dies (partial dies do not count)

The yield is affected by [3]:

- Die area 2.2
- Defects statistical distribution
- Manufacturing process defects density

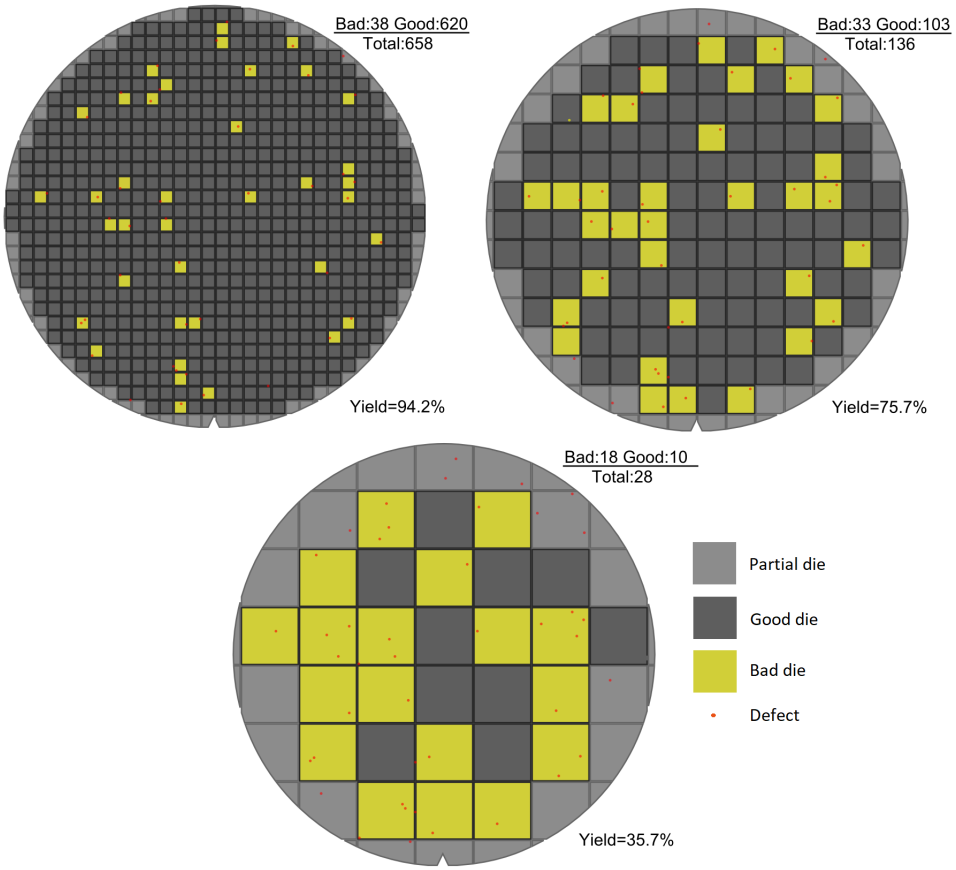

Figure 2.2: Yield variation with die size. Source: Wikimedia Commons

With this information, we can conclude that chip cost is not linear with its die size, since the larger the area, the more chips are discarded, which makes manufacturing larger chips more costly. Some companies design its chips to work with some manufacturing defects, since IC characterization allows them to know which part of the chips fails more often, disabling for example a part of the circuit and selling them as a mid-range product, instead of a top of the line. This happens for example in CPUs (Central Processing Units), where a manufacturing problem in a core leads to the company disabling that and other core and some memory to match a lower range product in their line.

2.3 The Automatic Testing Equipment

2.3.1 What's an ATE

While some authors used to not precisely define what an ATE is [1], nowadays, in the semiconductor business, we can define the ATE as a computer controlled machine which tests the ICs for any type of faults [6]. The ATE supplies the stored test vectors (which were previously created) to

the DUT (Device Under Test) and compares the output with the expected values, confirming the correct or incorrect operation of the DUT.

These tests are performed still in the wafer, just before the preparation of the *die*, using Wafer Probers (a type of ATE), and after the IC packaging process.

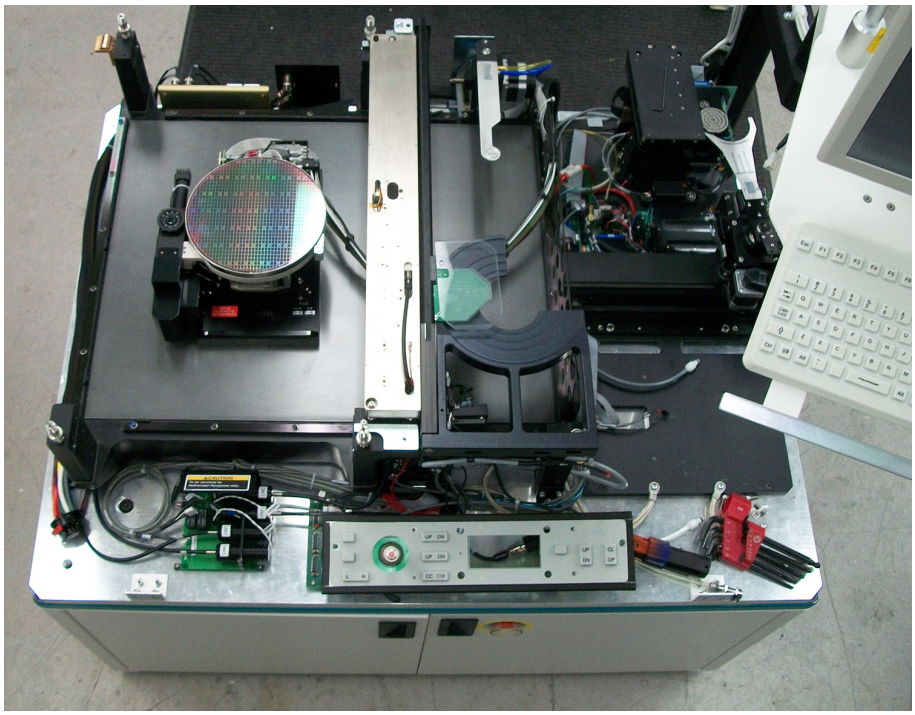

Figure 2.3: Wafer Prober. Cover panels, tester and probe card elements removed **Source:** MarTek, Inc.

2.4 Chip Access

2.4.1 JTAG (IEEE Std-1149.1)

This standard defines a serial communication protocol and a state machine accessible via a TAP (Test Access Port) interface. It allows test instructions and data to be injected into the IC, further improving its external testing capabilities, such as testing the DUT interconnections or the DUT itself, besides the possible BIST (Built-in Self Test) features already present in the IC [4]. This test logic is accessed through a TAP, a 4-pin (optionally 5) interface and is controlled by a trivial FSM (Finite State Machine) as we can see in figure 2.4, which makes the hardware level very simple.

JTAG interface pins

- TDI (Test Data In)
- TDO (Test Data Out)
- TCK (Test Clock)

- TMS (Test Mode Select)
- TRST (Test Reset) - optional

The state machine transitions are controlled by the TSM signal at every clock (TCK) positive edge. The main purpose of FSM are to control the IR (Instruction Register) and the DR (Data Register). The IR has 3 mandatory instructions:

1. EXTEST - for external testing, such as using pins to probe board-level behaviors
2. PRELOAD - to load pin output values before EXTEST
3. SAMPLE - to read pin values into the boundary scan register

Besides these, many more instructions can be implemented, depending on what functionalities are required and needed. After loading the desired instruction, it's now possible to load values into the DR or shift values out of it, depending on the use case.

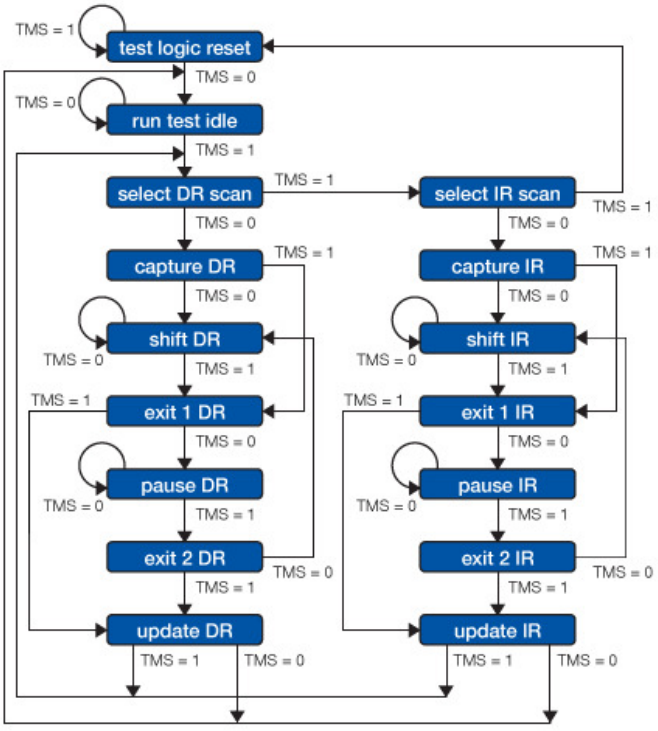

Figure 2.4: TAP Controller Finite State Machine

Since JTAG is a serial interface (bits are shifted into TDI each TCK clock cycle), it is very slow compared to a parallel interface. However, using just 4 pins is a major advantage, since pin count is a limitation. Another advantage of this interface is the possibility of connecting two or more devices in a chain. In this case, the number of clocks needed to shift the IR or the DR increase, since we now have multiple IRs and DRs. This eliminates the need to directly connect a circuit inside a SoC, for example, which can consist of multiple IPs from different vendors. This greatly

increases test efficiency and cost as there are always only 4 (or 5) pins needed to be connected to the DUT.

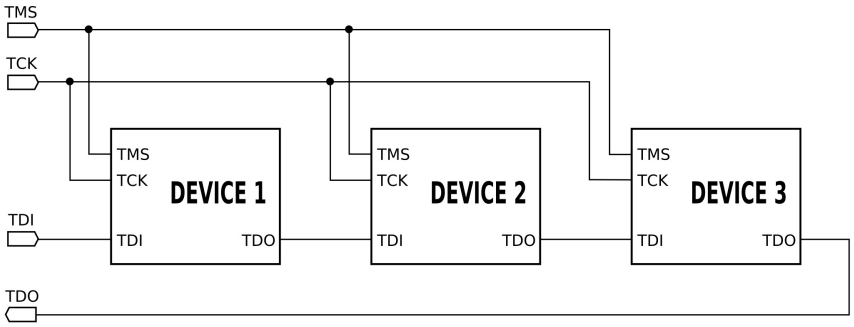

Figure 2.5: TAP interface controlling 3 devices in a chain

2.5 Hardware Description Languages

This section slightly deviates from the main topic of the thesis, but it's important to explain what an HDL (Hardware Description language) is and what its capabilities are, since in practice, most of the work will be reading an HDL and later, the implementation of a block.
As the name implies, an HDL is merely a language used to describe hardware, more precisely, the structure and behaviour of a circuit, usually digital logic ones. It is important to mention that despite looking very similar to a programming language, it is not one and cannot be used for such purpose. An HDL explicitly includes the notion of time, whereas a programming language does not.
There are many HDLs, such as Verilog, VHDL and System C, which are open source and the most used, but there are also other proprietary HDLs used by their companies. At Synopsys, the main HDL is Verilog/SystemVerilog and it will be the only HDL explained here, since it will be the only one used throughout this work.

2.5.1 Verilog/SystemVerilog

As previously mentioned, Verilog is an HDL, a general purpose one used in the design and verification of digital circuits. SV (SystemVerilog) came after, as a superset of Verilog, adding more functionalities to the base language and making it not just a language to describe hardware but also a language purposely made to verify it (SV is considered an HDVL, hardware description and verification language). In other words, SV features allow it to be a more robust language to design testbenches.
Because SV is a superset of Verilog, for simple circuits we don't really see much difference between both language. Below we can find a simple combinational two to one multiplexer written in both Verilog and SV, since the code is the exact same, due to the fact that SV is a superset of Verilog.

```
module  mux_2to1(
    input wire   a   ,
    input wire   b   ,
    input wire   sel ,
    output wire  out
);

assign out = (sel) ? a : b;

endmodule
```

Chapter 3

Synopsys SerDes Study and Overview

All of the work developed in this thesis is based on Synopsys Ser/Des - DesignWare® Multi-Protocol 10G PHY - which is a high-performance, multi protocol, multi rate PHY transceiver. Because of the nature of this thesis, none of Synopsys intellectual property can be revealed, and therefore the depth of the information described from here on will be limited and will be replaced by general concepts.

3.1 The SerDes

In order to transfer data from an IC to another, the datapath from the source IC must be connected to the destination IC. We could simply connect the I/O of each IC but, as of modern techonology, external pin count is limited, and dedicating the same number of I/O on both source and destination ICs would be costly. There are also the synchronization problems that would happen if the clocks were not exactly the same. Since this synchronization is affected by the chip process, voltage or even temperature conditions, synchronization can not be ensured [5]. Therefore, we need to serialize the data and send it through a channel (normally a single channel to minimize I/O but, if high bandwidth is required and one channel is not enough, multiple channels can be used). At the other end, we will deserialize, recovering the data and clock, and reconstruct the data. Since it's not in in the scope of this thesis to explore the complex structure of a modern SerDes, a generic schematic is shown in Figure 3.1, where it's possible to identify the output driver after the serializer, which is an example of an analog block that is not designed by the digital team, but by the analog team. The internal clock is often generated by a local PLL driven by an external reference clock. The opposite side, the first stage is the input stage, where the analog signal is converted back do digital, followed by the deserializer. In parallel, we have a CDR (Clock and Data Recovery) block which extracts the timing information from the serial stream.

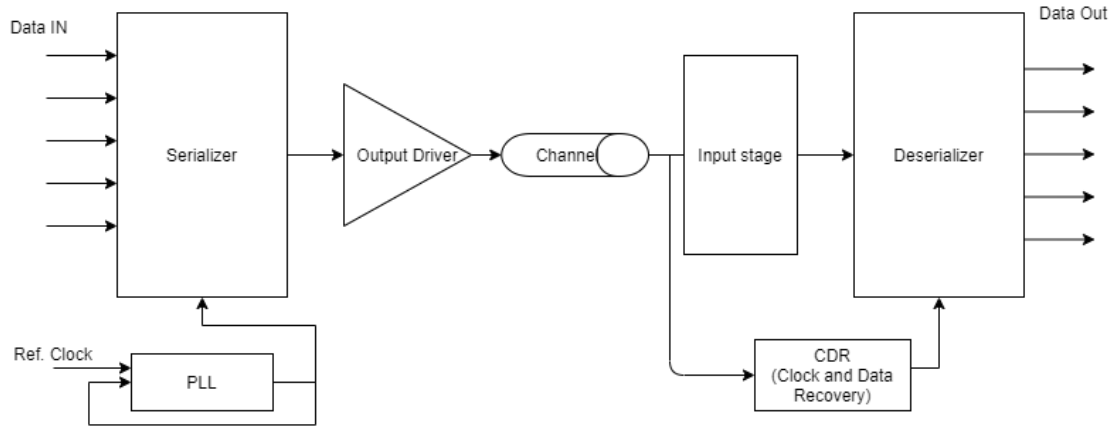

Figure 3.1: Generic schematic of a SerDes

3.1.1 PHY Registers

The PHY has a wide range of registers which have the purpose of configuring its multiple parameters, allowing different modes of operation. Some of these registers correspond to the PHY inputs, which means that the PHY can be configured without externally stimulating them. For testing, this allows for a more individualized test, eliminating the need of an external circuit.

3.1.1.1 Register Access

Parallel Access

The exact name of the interface can't be disclosed, and therefore this interface will, from now on, be referenced as Parallel interface.
The parallel interface is the PHY internal way of controlling its multiple blocks. All of the registers just mentioned can only be accessed by the internal register controller, which is, as the name suggests, an internal parallel interface used to read/write from/to the PHY registers.

JTAG

JTAG is also an option to access these registers and, in the case of an ATE test in the silicon or post production phase, it's the interface used. JTAG does not directly control any register of the PHY, since all of the JTAG commands are interpreted and converted to its internal interface instructions. As previously mentioned in JTAG (IEEE Std-1149.1), JTAG is a slow interface which will be the starting point of the optimization process.

3.1.2 ATE environment

Current ATE test environment consists of multiple individual tests which stimulate the PHY externally using JTAG, or, if still on the development phase, its parallel interface so as to increase simulation speed, since the functional stimulus is the same. It is important to note that after wafer

fabrication, the tests are made through JTAG. These tests set the PHY for different modes of operation and verify if its behaviour is expected. Due to the limitations of this thesis, these tests can't be disclosed. In any case, a generic test that is common to all SerDes is the loopback test, which can be internal or external, can be explained.
It is important to note that running the ATE environment through JTAG (which is translated to the Parallel interface) or through the Parallel interface produces the same results, but the simulation time is much longer in the first case.

3.1.2.1 Loopback Test

In this test, a signal is internally generated and then, if doing an external test, sent out of the PHY and returned (hence the name loop back) to it as a way to determine the correct behaviour of the device. This test needs an auxiliary external circuit to connect PHY TX (transmitter) and RX (receiver), which makes it hard to do in the wafer fabrication phase 2.1.7. To be able to perform this test still in the wafer, the PHY can internally connect RX and TX, but, as an advantage of this test, not all circuitry is tested, since the analog drivers won't be sending any signal to the outside of the PHY. Even though the internal loopback is not as complete as the external loopback, it allows a test already in the wafer testing phase 2.1.7.1. As mentioned in the beginning of the chapter, the SerDes supports more than one protocol. These protocols have to be told to the PHY before performing the test, as different protocols have different rates, and so the PHY must be configured to operate in that specific state. In the current state of the test, these configurations are all sent to the PHY each time we want to change protocol, which is not optimal in terms of efficiency.

3.1.3 Tests Study

During the study of the ATE environment, some ideas regarding the optimization process were discussed and discarded along the way, such as test procedures restructuring or firmware integration with the ATE environment, which would require more experience with the full ATE environment and the PHY itself.
In the end, it was concluded that the bottleneck was definitely the injection of the procedures through the external serial interface JTAG, which lead to the idea of the PHY having the procedures already inside to maximize testing efficiency. In order to do this, all the tests were studied again, this time to search for common procedures which would be chosen in the final solution.
The final procedures chosen were protocol rate changes, present in a task which was frequently called during the Loopback Test. As mentioned in the beginning of this chapter, Synopsys SerDes is a multi protocol PHY, and that makes the optimization of these rate changes a good solution, since the test sequences are used more than once, which allows the solution to be reused. For the optimization, two major routines, which configure different parts of the PHY for the new rate change, were chosen for analysis. We will call the longer sequence "routine type 1" and the faster one "routine type 2".

3.1.4 Optimization Analysis

Analysing the log files after running the environment in both JTAG and Parallel access modes, we could conclude the following:

- A single register write operation access through JTAG (*i.e.* JTAG + instruction translation to Parallel) takes 26 times longer than a Parallel access, as we can see, after zooming, in figure 3.2, which closely resembles the actual signals in the PHY. JTAG interface needs to send the address and then the data. We can observe this situation in the transitions the signals Parallel Address and Parallel Data. Signal Parallel Write Enable only asserts after given the total JTAG instruction starting from the first positive edge in the waveform of the signal JTAG Update. After receiving the JTAG instruction,it only takes one JTAG clock cycle for the Parallel interface to receive the Parallel ACK acknowledge (it takes 2 clock cycles to conclude the write operation).
- Routine 1 takes 20.8 times longer to run through JTAG than Parallel.
- Routine 2 takes 22.5 times longer to run through JTAG than Parallel.
- A single call of the routine 1 takes about 1.8% of the whole Internal Loopback with rate change test. In total, the routine takes aproximately 21.6% of the whole test.
- A single call of the routine 2 takes about 0.93% of the whole Internal Loopback with rate change test. In total, the routine takes aproximately 11.2% of the whole test.
- These two routines make up to 32.8% of the whole test.
- The maximum optimization possible with just these two routines is, after substituting the JTAG time with the Parallel time, a time decrease to $100-32.8+(21.6/20.8+11.2/22.5)=68.74\%$ of the original test time.

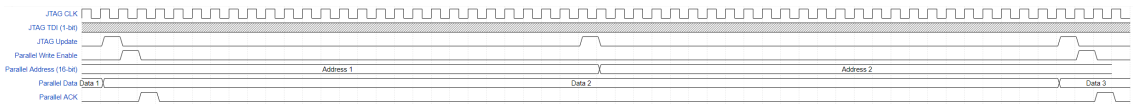

Figure 3.2: JTAG and Parallel interface signals comparison waveform. Write operation

Chapter 4

Proposed Solution

The previous chapter discussed how current SerDes ATE tests are performed, which helps us understand how we can optimize them. The proposed solution is a module that will be integrated in the PHY, which aims to reduce JTAG access by saving the instructions in the PHY itself, eliminating the external procedure of a rate or protocol change through JTAG.

4.1 Planning

The implementation was divided into 4 phases:

- Design Phase. The structure of the module, methods of iteration and storage of the sequences are defined here.

- Sequences extraction. These were extracted from the simulation logs and compared with a master configuration file for the rates of the product. A script was created in order to automate this process.

- Integration Phase. The module was integrated in the PHY. Specific location is chosen near the Parallel Interface controller.

- Simulation Phase I. Module testing with one of the sequences.

- Simulation Phase II. Module testing with multiple sequences.

- Synthesis Phase. Last phase of the planning which translates in a area increase discussion to add to the conclusions.

4.2 Architecture Overview

4.2.1 Module Flow

Instead of programming the PHY through JTAG, the external testbench will just execute a write operation in the register created specifically for this situation. The data written specifies the sequence number which the testbench would write through JTAG otherwise. The module can be described as an iterator, since after being activated through a write operation in the register, it iterates over its corresponding sequence, which is hard coded in the module itself. After the iteration is finished, the module will revert the register to its default value so the testbench can read it and confirm that the sequence was successfully loaded. This flow can be better understood when analyzing figure 4.1.

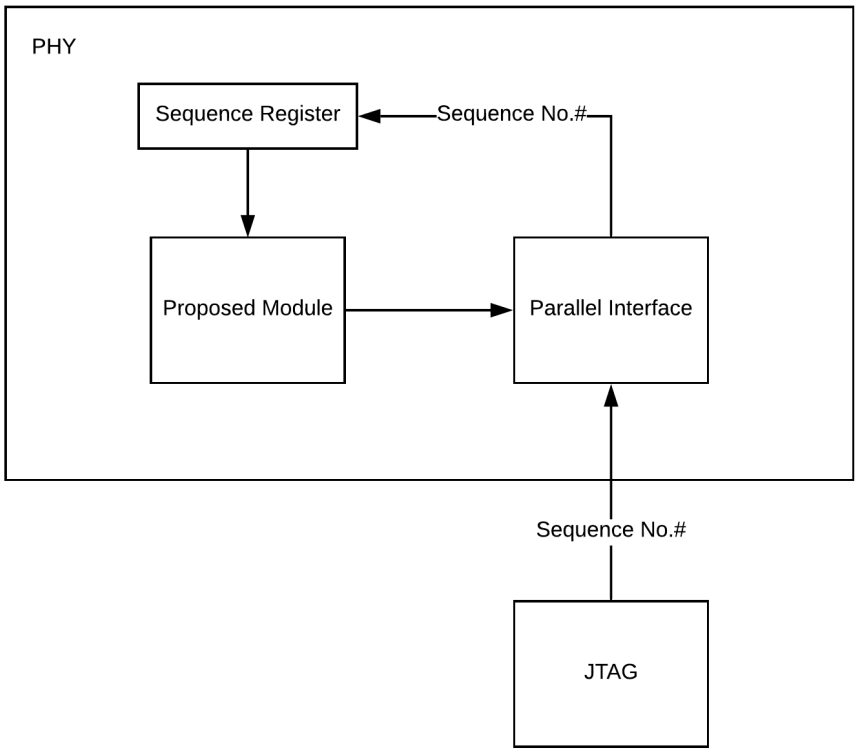

Figure 4.1: Black box schematic of the module integrated in the PHY

4.2.2 Module architecture

Decoder

The first stage of the module is a 5 to 32 Decoder, since for an initial stage of the work, a maximum number of 32 sequences was defined. The decoder converts the 5 bit sequence bus coming from

the register into the enable of the respective sequence module, signaling it to start.

Sequence Blocks

The sequence submodules are the second stage of the module. They are instantiated for each sequence, since they hold a unique sequence. These are activated by the decoder and, upon activation, their output cycles through the addresses and corresponding data, always waiting for an acknowledge from the parallel interface.

Iterator

This sub module is the controller of the sequences. Its job is to provide the clock and position for the sequence cycling, stop the sequence modules when they finish an operation and revert the sequence register to its default value.

4.2.3 Sequences Script

Since all the sequences had to be extracted from the current environment, there was the necessity of writing a script which would do exactly that. Therefore, using Python, the script inspected the simulation log files and grouped the routines 1 and routine 2 sequences, creating 24 different sequences to be implemented in the module.

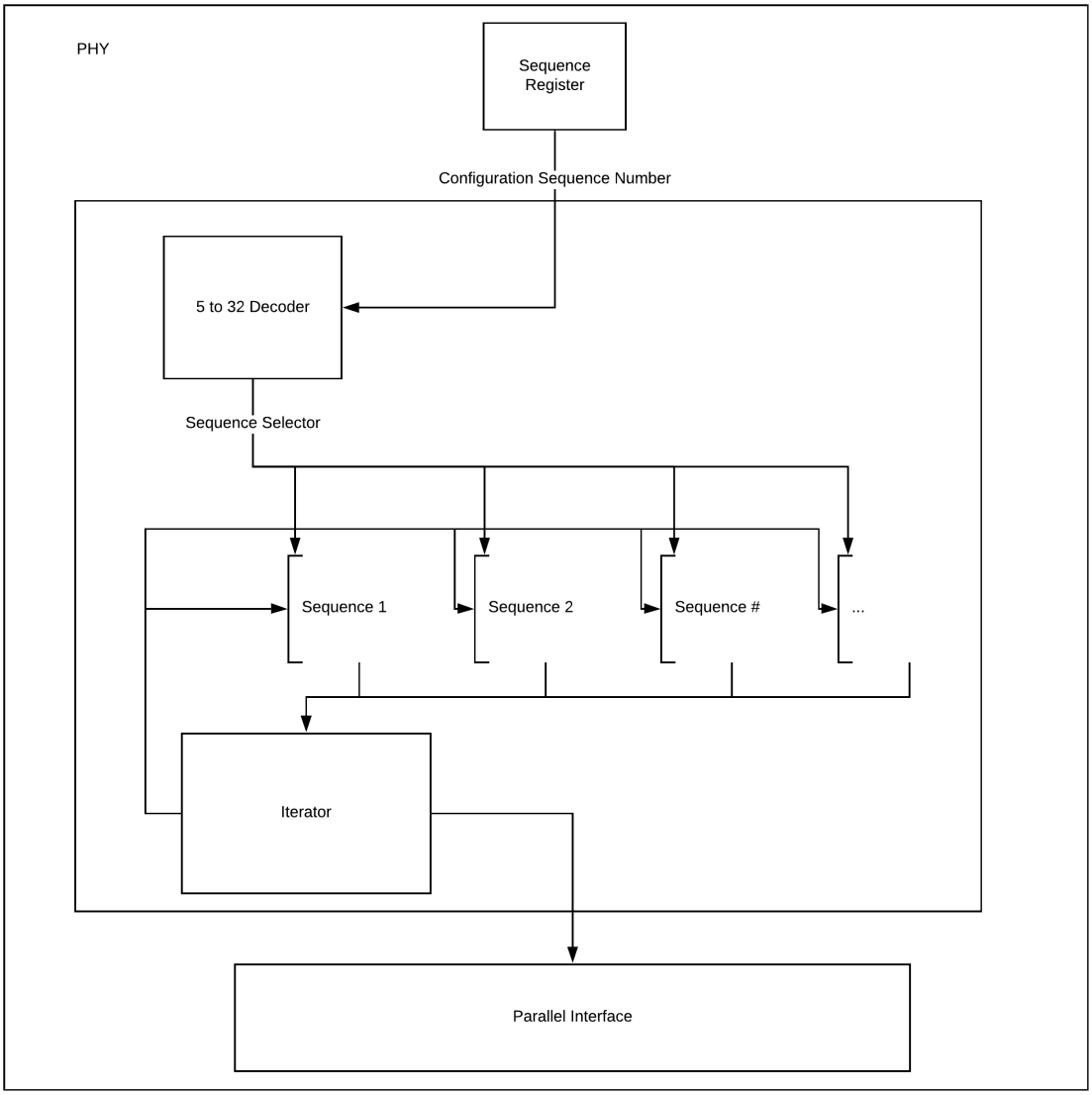

Figure 4.2: Internal high level schematic of the module

4.3 Results

In this section only the Simulation Phase I results are presented since it was the only phase successfully concluded due to time limitations, since the debug, the learning process of the software and tools and the study of the current ATE scripts took longer than expected. Regarding the debug, the simulation would run for hours, which made the debug a problem.
Despite these problems, at the final stage of Phase I, the module successfully loaded the sequences into the Parallel interface after detecting its register had been written by the JTAG interface. Comparing the sequence written through the module to the one written through JTAG, this architecture lowered a routine type 1 sequence to 10.5% of its time, resulting in a total of 1.52% reduction in the total time of the test.

Since the time was running out, an attempt was made to include the area increase in the results by running the last Phase, the Synthesis, which was unsuccessful. The module had been written with non synthesizable constructs, due to lack of experience with Synopsys synthesis tool, and needed a different structure, which couldn't be done in time. It was also after this phase that another problem was identified, which was the fact that Phase II couldn't be concluded either, as the simulation did not pass for multiple sequences.
The following results are then extrapolated from the single sequence optimization and are an approximation of the expected from the multiple sequence optimization. The reduction of the twelve type 1 sequences to 10.5% would mean a total of 18.24% reduction. Type 2 sequences would have a similar reduction, in the order of 10%. Extrapolating the results, a type 2 sequence would reduce the total time in approximately 0.5% to 0.8%. Multiplying by 12 sequences, this results in a decrease of 6 to 9.6% of the total test time. Finally, adding these $2 * 12 = 24$ test sequences reductions, the total time reduction would be between 24.24% and 27.84%.

Chapter 5

Conclusion

5.1 General Conclusions

Overall, the main objective of the thesis was accomplished, there was an optimization in the simulation time of the ATE environment. Considering that not all of the expected timing reductions were applied to the environment, we can call this work a Proof of Concept. The feasibility was demonstrated by reducing the partial test time for a single sequence and a more optimized environment can be achieved by increasing the number of sequences in the module.
There's also the fact that the architecture can be moved to a different product of the SerDes family, which also validates the modular requirement of the architecture. This possible future integration should be easy and relatively fast, since the scripts to extract the sequences from the simulation logs were created. It's also important to note what didn't go so well in this work, such as leaving the synthesis for last and finding non synthesizeable constructs in the code, due to the basic Verilog knowledge I had at the start of the thesis, or taking too much time to learn how to use Synopsys software and scripts, which didn't allow for a total completion of the proposed solution.

5.2 Future Work

First of all, the module should be restructured a rewritten according to standard practices in order to procceed with a synthesis in its Simulation Phase I state. Secondly, the module integration with the PHY for multiple sequences should be reviewed, since the tests with multiple sequences have failed.
Then, a new synthesis should be performed to compare the PHY area before and after the module integration to understand if the increase in area pays off the lower test time.
Thirdly, as an optional step, more repetitive sequences can be found in other tests of the ATE environment if time is allocated for such. Although not easy to identify at first glance, the identification of such sequences are crucial for the reduction of the ATE environment verification time.

References

[1] Keith. Brindley. *Automatic test equipment*. Newnes, 1991.

[2] Tim Haifley and Sandra Healy. Automotive Electronics Council GUIDELINE FOR CHARACTERIZATION OF INTEGRATED CIRCUITS Automotive Electronics Council Acknowledgment. Technical report, Automotive Electronics Council, 2013.

[3] W Hansch, J Biba Admos, and Josef Biba. Advanced MOSFETs and Novel Devices Advanced MOSFETs and Novel Devices Yield Calculation. Technical report, Bundeswehr University Munich, 2017.

[4] IEEE Computer Society. Test Technology Standards Committee., Institute of Electrical and Electronics Engineers., and IEEE-SA Standards Board. *IEEE standard test access port and boundary-scan architecture*. Institute of Electrical and Electronics Engineers, 2001.

[5] David R. Stauffer, Jeanne Mechler, Michael Sorna, Kent Dramstad, Clarence R. Ogilvie, Amanullah Mohammad, and James Rockrohr. *High Speed Serdes Devices and Applications*. Springer, 2009.

[6] Laung-Terng Wang, Cheng-Wen Wu, and Xiaoqing Wen. *VLSI Test Principles and Architectures: Design for Testability (The Morgan Kaufmann Series in Systems on Silicon)*. Morgan Kaufmann, 2006.