Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Institut für Integrierte Systeme
Integrated Systems Laboratory

Department of Information Technology and Electrical Engineering

VLSI III: Test and Fabrication of VLSI Circuits

227-0148-00L

Exercise 7

Speed Testing

F. K. Gürkaynak
Prof. Dr. H. Kaeslin

SVN Rev.: 1628
Last Changed: 18-12-2015

1 Introduction

The goal of this exercise is to find the maximum frequency of a chip or of a submodule. As we will see, the maximum frequency can depend on several properties, as reset, IO-timing, supply voltage, test vectors, temperature etc. To find the maximum frequency, we utilize a spec-search test, and a shmoo plot, both are explained on the wiki.[1]

Before we can start with the exercise, everything has to be set up properly.

1.1 Perparations

This exercise is using the *Pony* chips, a 65nm technology crypto chip with multiple modules of different frequencies. You can get the chips in the office J60.1. Before you start with the exercises, you have to do a couple of preparation steps.

- Log in on *hava* and copy the files hp93000 (`./home/vlsi3/ex_07/install_ex07`)
- Start the tester
- Start the eclipse environment
- Contact an assistant

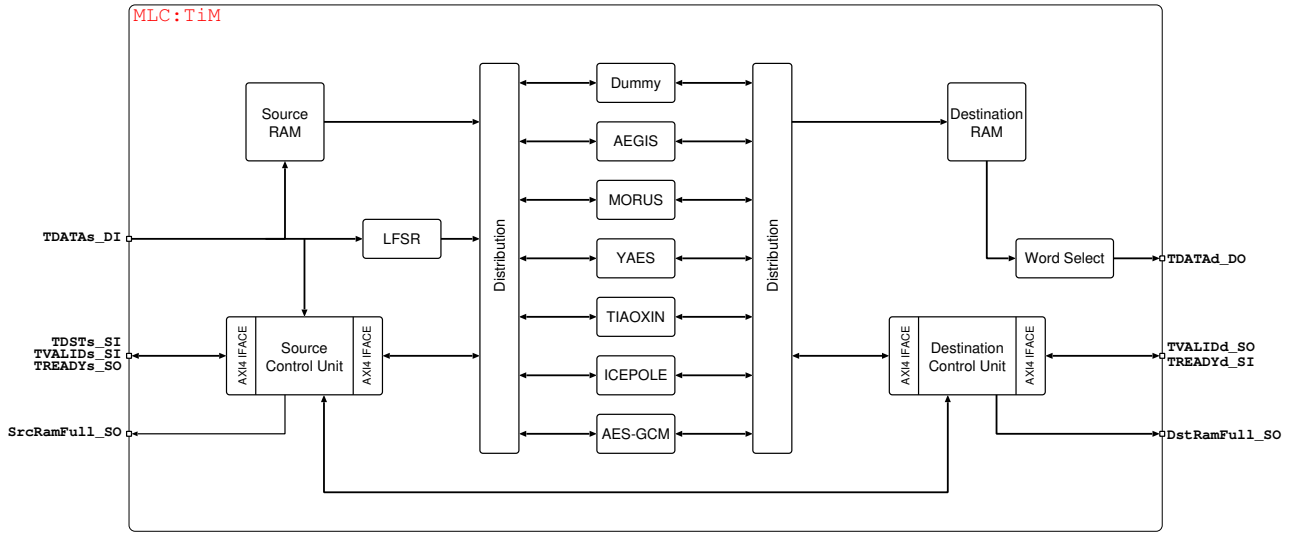

Figure 1: Top level block diagram of the *Pony* chip

2 Basic Tests

In this section we will load all the setups and perform some basic functional tests in order to see if the chips are responding. But first we will give some information about the *Pony* chips.

[1] `http://eda.ee.ethz.ch/index.php/Maximum_Frequency_Tests`
`http://eda.ee.ethz.ch/index.php/Shmoo_Plots`

2.1 Pony chips

A simplified architecture of the *Pony* chips is shown in figure 1. As you can see, the *Pony* chip is a container for multiple cryptographic modules. The chips are tested by loading input data to the source RAM, then performing the cryptographic tasks, and verifing the data in the destination RAM. Loading and unloading is done over a simple AXI stream interface. Throughout this exercise we will only work with the Morus module.

2.2 First Steps

You can open the already existing testflow (*vlsi_ex07*) and load all the setups (*pins, timing, levels*). The testflow contains some initial ATPG and functional tests, and some place holders (*Task1-6*) which you have to extend with specific tests. Before we start with finding the maximum frequency, we load the board with one chip (choose any *Pony* chip).

Check sequentially if the continuity tests, ATPG, BIST, basic IO and basic functional tests pass. If all tests pass, you can proceed with the exercise.

3 Finding the maximum frequency

3.1 First frequency sweep

Student Task 1: Read the wiki page on how to configure a frequency sweep and determine the maximum frequency of the morus module, by using a spec-search test. For the start, we use the simple timing equation 1. You can use the following parameters:

- Timing: Equation 1, Specification 5, functional set
- Level: Equation 1, Specification 2, ideal-terminated set
- patterns: MorusAll
- spec: MHz
- fmin: 10 MHz
- fend: 400 MHz
- fstep: 1 MHz

What maximum frequency did you measure with the given set? You can use the *linbin* method to get a higher resolution of the maximum frequency.

3.2 I/O - Timing

As you probably remember from VLSI I&II, the timing depends on the input, and output delay. Hence, it is important to know the setup and hold times of the chips. In order to be able to control the setup time and propagation delay, we need a timing equation with more freedom as for example timing equation 2, which is parametrized. Copy the frequency search tests and use the parametrized set this time. The setup times and the propagation delay of the chip can be configured in the specification.

Student Task 2: Change the setup times and propagation delay values and see how the maximum frequency varies (don't forget to download the specs each time you change something).

In the io-timing search block of the testflow, you will already find a couple of tests which help you to determine the setup time and propagation delay.

Student Task 3: Utilizing these tests, determine the setup time, and the propagation delay, and update the timing specifications with the found values.
What is the maximum frequency you measure now?
Note: remember to use the "ideal-terminated" levels.

3.3 Dependency on Vectors

So far we have been using the pattern MorussAll.By using this pattern you should have measured a maximum frequency a bit over 300MHz. The reason why we cant get higher frequencies, is that the source RAM is limiting us in speed and it is therefore not possible to determine the maximum frequency of the Morus block with this pattern. Luckily the designers were aware of this, and added a linear feedback shift register (LFSR) to be able to generate random data directly on chip, hence, not utilizing the timing critical RAM. The pattern Morus_lfsr can therefore be used to determine maximum speed of the Morus block on chip.

Student Task 4: What is the maximum frequency when using the LFSR?

3.4 Frequencies above 400MHz

Read the Wiki page on Tester Xmode[2]. If you want to use frequencies above 400 MHz, this Xmode has to be used. We will use the x2 mode, in which 4 events are defined per tester cycle. Hence, we can achieve frequencies up to 800MHz, more than enough for every module on the chip.

3.5 Changing Timing Sets

If the pads or other parts of your chip interface are limiting your speed, you have the possibility of changing timing sets during a test. Data can be read in and out using a slow speed, while the chip processes the data at a higher speed. This is of course not possible for a streaming processor type of chip.
Since the interfaces of this chip are not designed to work above 400MHz, all vectors with the x2 mode will also have changing timing sets, where the interface speed is 8x slower than the chip itself. Open one of the patterns, and try to find the changing timing sets. (e.g. @line 4622 of pattern Aegis_lfsr_x2).
In order to use the x2 mode, use the timing equation 10 which already implements this mode. Timing equation 10 defines the interface speed, and then internally computes the fast speed for the chip. Hence, if you specify a frequency of 50 MHz, the chip will actually run at 400 MHz.

[2] `http://eda.ee.ethz.ch/index.php/Tester_Xmode`

Student Task 5: Define a new frequency search test and utilize the timing equation 10, which is using the x2 mode. Don't forget to also update this specification with the previously found setup time and propagation delay.

- Find the maximum frequency of the pattern Morus_speed_x2
- Find the maximum frequency of the pattern Morus_lfsr_x2

3.6 Frequency vs. Supply Voltage

We have been able to determine the maximum frequency of the Morus module. However, we have only determined the maximum frequency at one operating point. How does the maximum frequency depend on the supply voltage? One good way to see this impact is to perform a shmoo plot. Read the wiki pages on how to setup such a shmoo plot. In this exercise we are interested in the behaviour of the maximum frequency versus the supply voltage.

Student Task 6:

Perform a shmoo plot with the following paramters:

- $f_{low} = 10MHz : \#100 : 100MHz$
- $Vdd = 0.75V : \#100 : 1.4V$

Did you get what you expected? Save the result somewhere, or show it to one of the assistants.