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 5

Power Testing

F. K. Gürkaynak
Prof. Dr. H. Kaeslin

SVN Rev.: 1915
Last Changed: 2018-10-17

Reminder:
With the execution of this training you declare that you understand and accept the regulations about using CAE/CAD software installations at the ETH Zurich. These regulations can be read anytime at `http://eda.ee.ethz.ch/index.php/Regulations`.

1 Introduction

The goal of this exercise is to properly take measurements of the power dissipation of a chip. You will learn how to use specifications and level sets and to setup the dynamic and leakage power measurements.

In particular, you will learn how to

- Understand level setup of the testing environment
- Setup a dynamic power measurement and deal with loop patterns
- Do leakage power measurements and compare results with estimated values
- Vary measurement parameters in order to verify the correctness of measurements

You can find detailed manuals for the tester on our eda-wiki.[1] If you feel unsure about the basic handling you might want to brush up your memory by having a look at the previous exercises or *Tester: Basic Handling* and *Tester Software* on the wiki.

2 Preparations

This section first refreshes some theoretical background and gives information on the *Pony* chips. Then, you will learn how to perfom the level setup of the tester setup.

2.1 CMOS Power Dissipation

As a reminder, power dissipation in digital CMOS circuits is given as the sum of static and dynamic power dissipation. While static power is dependent on supply voltage and leakage current, dynamic power depends on several factors.

Student Task 1: Write down the formula for the power dissipation in a CMOS circuit:

2.2 Pony Chips

This exercise is using the *Pony* chips, a 65nm technology crypto chip with multiple modules of different frequencies.[2]

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. Alternatively, pseudo-random data can be applied from an LFSR. Loading and unloading is done over a simple AXI stream interface. Each module and the LFSR can be disabled through clock gates for power savings. Throughout this exercise we will only work with the AEGIS module.

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

[2] The datasheet and a user guide can be found in the *doc* folder of your exercise.

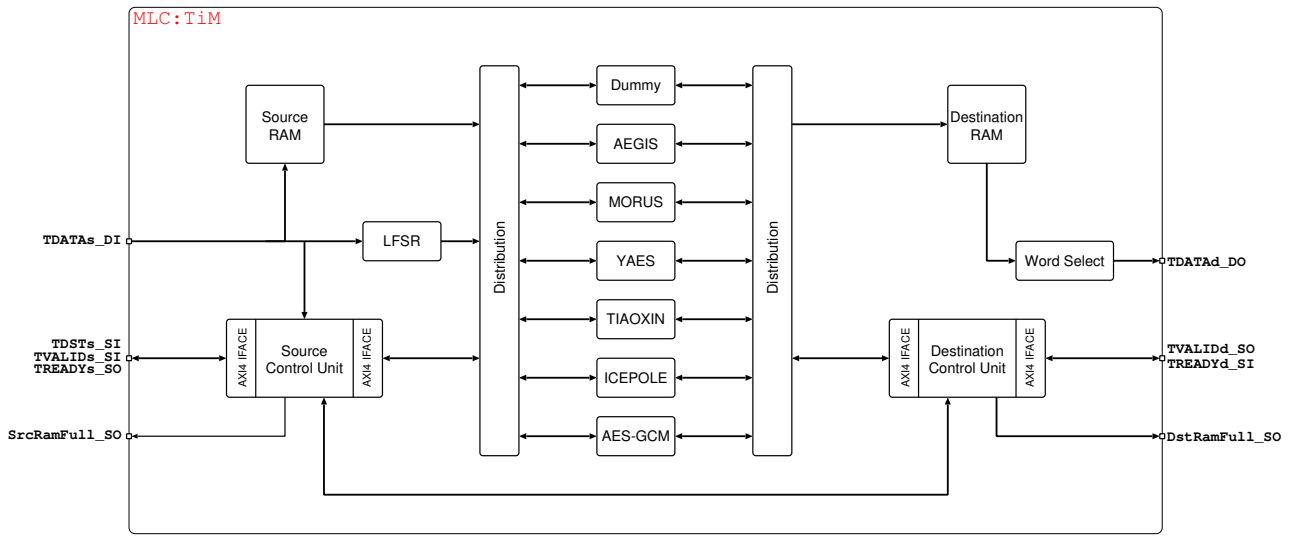

Figure 1: Top level block diagram of the *Pony* chip

2.3 First Steps

Prepare the exercise as follows:

Student Task 2:

- Take ESD protection measures. Set up the tester as usual. Pony uses a QFN56 package, thus mount tester board *Student_QFN56.std_v1_rev0*.
- Log in on *hava* and copy files to your exercise folder:

```
sh> cd YOUR_EXERCISE_FOLDER
sh> ~vlsi3/ex_05/install_ex_05
```

- Start the SmarTest Eclipse Workcenter

```
sh> ~hp93000/bin/start_93000
```

- Select a temporary folder for the current session.
- Load the device. (`93000→Device→Change Device`).
  Select the *hp93000* directory from the *ex_05* folder.
- Load the preconfigured settings to the current test program (right-click and `Load` on the according items in the *Test Program Explorer*)
  - Pin configuration *io_vlsi3ex05*
  - Levels *levels_vlsi3ex05*

  There will be some error messages when loading the level configuration which we are going to fix soon.

3 Level Setup

The level setup on one hand defines the power supply levels (DPS) to your chip, and on the other hand specifies the I/O drive levels for your chip's inputs and outputs. The level setup consists of the two main components:

Equations that assign voltage levels and current thresholds to power and I/O pins according to parameters they declare.

Specifications that set the actual parameter values to be used for the equations.

3.1 Equations

Student Task 3:

- Open the *Level Setup* window by selecting the levels item in the *Test Program Explorer* and pressing `F3`. Alternatively right-click on the item and select `Open` or double click.
- Open the equations by selecting `Select→Edit Equations`.

In this file, multiple different equation sets (EQNSET) can be defined. Each equation set consists of:

SPECS that declare parameters which can be changed manually or in the test flow.

EQUATIONS that define actual values to apply to pins which are calculated from the SPECS.

DPSPINS that assign voltage levels and current limits to tester power supply channels.

LEVELSETs that assign the values from the EQUATIONS to pins or pin groups.

Student Task 4: Have a look at the equations file and familiarize yourself with it.

- What is the difference between the first two level sets defined? Discuss with the assistant:

  _____

  _____

- Complete the level set "Standard" so that the inputs are driven with a 25% drop and the outputs tolerate up to 10% drop from nominal values.
- Download the file into the tester configuration by pressing `F8` or through `Shell→Download` and close the file.

  Attention! You have to download the changed confuguration to the tester as described above to save your changes. Saving the equation file (through `Ctrl+S`) has no effect since it's only temporary on your workstation.

3.2 Level Specifications

Student Task 5: Finally, we need to set the specific values for the SPECs from the equation files. This is done through *Specification Sets*.

- Open the *SpecTool* by selecting `Select→Edit Specifications`. You will see the (only) spec set currently defined. All values have intentionally been set to 0.
- Where can you find the appropriate values to for your specification set?

- Change the existing specification set to reflect the typical operating supply voltages of this chip. Set *curr_limit* $= 200\,\mathrm{mA}$. Also rename it to reflect the values (`Change→Change \ Name`.
- Add two more specification sets with the minimum and maximum supply voltages, respectively. You can add new specification sets through `Change→Create Specification`.
- Download your changes by pressing the big down arrow button and close the SpecTool.
- Verify your changes in the *Level Setup* window by switching through the specification sets and level sets. Show your setup to the assistant.
- After closing the *Level Setup* window, make sure to save your new level setup by right clicking on the levels item *Test Program Explorer* and selecting `Save`.

4 Testflow

The level setup is now complete and we can start testing. For this exercise, the additional setups as well as patterns are provided.

Student Task 6:

- Load the testflow *testflow_vlsi3ex05*.
- Load the additional missing setups (Timing, Pattern) one-by-one or by right clicking your loaded testflow in the *Test Program Explorer* and selecting `Load All Setups`.
- Mount a pony (*badum tshh*) and run continuity, ATPG and functional tests.

4.1 Setting up a Power Measurement

In this exercise we would like to determine the power dissipated by the core of the *Pony* chip while running the AEGIS module to encrypt data from the LFSR (cf. Figure 1). The pattern that accomplishes this is called ”aegis_lfsr_only”. The testflow item ”func_runAEGISlfsr” will run this pattern as a functional test.

Student Task 7: We will now create a test in the testflow to measure the power dissipated by the chip during the "aegis_lfsr_only" pattern.

- First, run the functional test for this pattern to verify it's working properly.
- Add a new test inside the "Power Measurements" group in the testflow:
  1. Right-click on the node inside the group and select `Insert→Run Test` or select the note and press `Ctrl+N`.
  2. Give a sensible name to the new node.
  3. Select `dc_tml→DcTest→OperatingCurrent` as the test method.
  4. Select the "simple_10MHz" timing spec and the typical level set you created earlier.
  5. Finally, select the test pattern.
- Run the test. What happens? Why? _____
- Fix the issue and rerun the test. What results do you receive? Do they make sense?

_____

The tester's built-in digital multi-meter (DMM) actually measures current. Knowing which level set we use - and hence what voltage is applied to the chip - arriving at the power dissipation is trivial.

Looking at the *Parameters* section of your test method, you will notice that the *testName* of our test is specified as "passCurrLimit_uA". This test works by comparing the measured current to a preset limit, specified in the *Limits* section of the test method. If there are no limits, the test will pass anyways.

As with a conventional multi-meter, the magnitude of your limits impacts the values you can measure. Don't expect to measure individual $\mu\mathrm{A}$ when you set the limit in the range of A.

Student Task 8:

- Set the current limits for the test to lie between $1\,\mathrm{mA}$ and $200\,\mathrm{mA}$. Watch the units!
- As we're only interested in the power dissipated in the core region of the chip, select only this supply in the *dpsPins* section of your test *Parameters*.
- Rerun the test. How much current is drawn by the core of the chip? _____
- Change the test *Parameters* such that it only takes one single sample. Also, vary the delay before the sample is taken for values between $45\,\mathrm{ms}$ and $55\,\mathrm{ms}$. List some of the values you measure at various points in time:

  _____
- What is the chip doing during these measurements? How long does the current pattern run for? Open the current pattern and try to familiarize yourself with what it is doing.

  _____

As we have seen, it's nigh impossible to take a measurement sample at a precise point in the test pattern.[3] In fact, one cannot be sure that the measurement on the DMM and the begin of pattern

[3] Even though it's possible to trigger measurements through the test pattern itself, the problems with the synchronisation of delays and difficult to plan flow of pattern reloads is still not mitigated.

executions are perfectly synchronized. Also, depending on the installed tester memory, patterns need to be reloaded which causes stalls that make timing the appication of a specific vector very difficult. Lastly, the testing environment will restart the pattern if it runs out before the delay has passed.

It should be obvious now that in order to reliably measure the current drawn during a specific operation of the chip, we need to make sure it surely remains in this state during the measurement. This can actually be done in a variety of ways and greatly depends on the functionality of the chip under test as well.

Student Task 9: We are going to explore ways to achieve a steady continuous operation in the region of interest.

What could, in general, be done to allow for easy power measurements during specific operations of a design? Discuss with the assistant:

The controller of the *Pony* chip blocks any further crypto module execution once the destination RAM is full to prevent overwriting output data (c.f. Figure 1). As the depth of the on-chip memory is very limited, continuous execution of the module would be frequently interrupted because of the need to reload and clear the RAM contents. While this lock on the destination RAM can be disabled, you would still need to reload data in the source RAM or issue a command to reexecute the same data.

The chip is equipped with an LFSR exactly for the reason of easier power measurements[4]. A command can be issued to the controller in the chip that causes data from the LFSR to be applied to the module on every clock cycle without limit. Together with a disabled destination RAM, this can be used to run a module for as long as you wish.

Student Task 10:

- First, clone the current test pattern and save it as ”aegis_lfsr_loop”.
- Open the new pattern and check out vector number 7. Currently the *pony_opSel* field is set to `0000`, signifying a `NOOP` (no operation) instruction. Set this field according to the user guide in order to disable the destination RAM.

In order to continuously stay in the LFSR mode, we will instruct the tester to repeat vectors in our pattern:

- Identify the vector entry that starts the continuous LFSR mode to apply plain text (PT) blocks to the module, using the comments in the pattern.
- Insert a `LOOP` instruction into a vector somewhere after that point.
- Close the loop with a `LOOPEND` before the vectors break this mode (again, look at the comments in the pattern).
- Run the measurement again using our new pattern, a delay of $500\,\mathrm{ms}$ and averaging over 1000 samples. What current do you measure?

[4] Also, the LFSR is significantly faster than the RAM which is important for speed tests.

- Repeat the measurement with a different number of samples and/or delays to verify its consistency.

4.2 Impact of Frequency on Active Power

Now that we have a working power pattern, it makes sense to start varying variables contributing to total power in order to make different measurements and verify whether our measurements make sense.

Student Task 11: We will start by making measurements at different frequencies.

- First, run a measurement with our current setup, using a delay of $500\,\mathrm{ms}$ and averaging over 1000 samples. Mark the corresponding power in Figure 2.
- Run two more measurements, this time using the following timing specification sets: *simple_25MHz* and *simple_50MHz*. Mark the corresponding power values in Figure 2 as well.
- Interpolate the power between the points you measured. What property do you expect from this graph?

Figure 2: Gridlines for power plots

4.3 Impact of other Factors on Active Power (Optional)

After having varied frequency, we can also explore the impact of other factors on active power.

Student Task 12:

- What other factors impact active power and how can we vary them?

- Choose one of the three variations and perform your measurements. Mark the results in Figure 2.

4.4 Determining Static Power

Another number of interest is the static power consumption of the circuit. There's two ways of determining leakage power: by interpolation or by actual measurement.

Student Task 13:

- How can you determine static power from your results in the previous section? Provide an estimate for the leakage power of the chip.

- In order to measure static current, set up a new test node in the testflow with the following attributes:
  - Test method: `dc_tml→DcTest→StandbyCurrent`
  - Timing Set Spec: *simple_10MHz*
  - Level Set: typical values, terminated
  - Pattern: "aegis_lfsr_only"
  - 1000 samples after $500\,\mathrm{ms}$.
- Run the test. What do you obtain and does the value correspond to your estimate?

- Change the pattern to "mixAllCand" and repeat the measurement. Does it change?

- Run the functional test "*mixallcand_func*", then run the measurement. What changed?

- Run the measurement with the $50\,\mathrm{MHz}$ timing set spec. Does the value change? Why?

- Discuss with the assistant what factors could impact the static power of a design:

5 Discussion

Student Task 14: If you have time, go back to Student Task 12 and perform another exploration.

Discuss any questions you might have with the assistant.

Student Task 15: Cleaning up:

- Close all programs and properly turn off the tester.
- Log off from the workstation and clean up the workspace.
- Do not forget to remove the ESD strip from your feet!