FOBOS: Flexible Opensource BOard for Side-Channel Analysis
"DPA in a Box"

Jérémy Barthélemy, Aaron Hunter, Anne-Marie Cressin

TABLE OF CONTENTS

1 Introduction

1.1 Problem

FOBOS is the Flexible Opensource BOard for Side-channel analysis. Side-channel analysis is a type of passive attack where a device is being monitored while it performs a cryptographic function, with the ultimate the goal of recovering cryptographic secrets such as the key or plaintext. Unfortunately, a typical DPA (Differential Power Analysis) Setup requires approximately $6000 to $12000 as expensive equipment such as a Sasebo Board and a digital oscilloscope are typically used.

1.2 Motivation

Due to the high costs of current Side-Channel Analysis boards, we wish to develop a tool which will perform Power Analysis and Fault Injection on a commercial off the shelf Spartan-6 FPGA board using a custom PCB to explore a cheaper alternative to performing these kind of attacks, while keeping cost to a minimum, and to release this as an open-source project.

1.3 Proposed Solution

The current implementation of FOBOS uses the Spartan 3-E based Nexys 2. We are migrating the current FOBOS system to use the Spartan-6 based Nexys 3. FOBOS will also be expanded to include the ability to perform Fault Injection attacks as well as the currently implemented Power Analysis attacks.

The Nexys 3 will be used to control the complete operation of the target, or victim, board. FOBOS will have control over the power supply, clock generation, and input/output of the victim board. In order to perform Fault Injection, a custom PCB will be created that allows clock and power glitches to be injected to the victim board. These glitches should cause erroneous operation for components that are on the critical path. Our board will be modeled on circuits presented by David Oswald for Fault Injection. [1] This clock/power PCB board will plug into the two PMOD connectors on the Nexys 3, JC and JD. [2] The clock signal to the victim will be controlled by the Spartan-6 FPGA. The power supply will be a Digital to Analog converter connected to an operational amplifier. This will allow the FPGA to control the voltage level the victim board receives.

In order to perform Power Analysis attacks, an OpenADC board [3] will be connected to the other two PMOD connectors on the Nexys3, JA and JB. The OpenADC allows for the supply voltage and current consumption of the victim to be monitored. There is no current monitoring option on the OpenADC, so a current shunt resistor will be added on to the clock/power board and a cable connected to the OpenADC.

1.4 Background

There are two main approaches to attacking devices that implement cryptographic functions: Side Channel Analysis (SCA) and Fault Injection (FI). SCA is considered a passive attack and not an active attack because it does not require any actual physical disturbances to the victim circuit to perform the attack. FI, on the other hand, attempts to disturb the victim circuit in order to erroneous operation to occur that allows the security of the circuit to be compromised.

SCA can be broken down by the side channels that are analyzed. In a power analysis attack, the current or change in current consumed by the device is recorded. With a timing attack, variations in execution time are recorded then analyzed to ultimately determine the key.

1.4.1 Simple Power Analysis

Simple Power Analysis (SPA) relies upon the interpretation of power traces in order to possibly obtain the key (or to offer some insights into the inner workings of the victim device). Sometimes this will work when the power consumption of the circuit depends heavily upon the bits of the key. If this is the case, it could be easy to see that for a key bit of 0 we have operation a, and for a key bit of 1 we have operation b. When we see that these operations a and b have distinctly different power consumptions, we can extract the key input to the circuit.

1.4.2 Clock Fault Injection

We are feeding a normal clock from our attacking device to the victim device. We then perform injection of clock faults by altering the clock quickly for the victim device such that this causes “glitches” in the victim circuit, which then in turn allows us to further analyze the device.

2 Design

We based our system off of work done by Oswald on the GIAnT system. We also build our work off of previous work on the FOBOS system. Our work focuses on building a PCB/perf board that allows for the measurement of the victim power consumption and injection of clock and voltage faults.

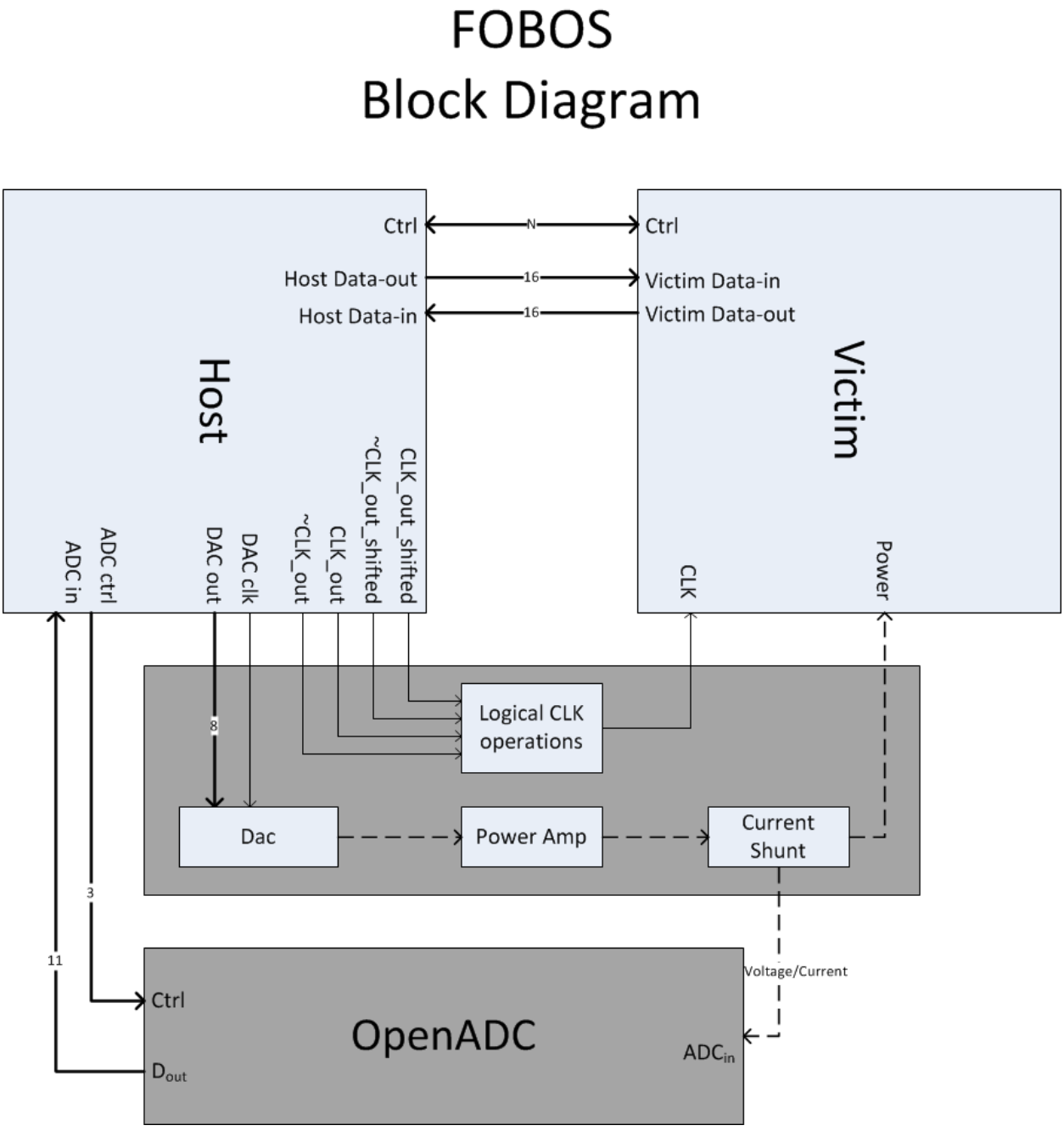

Figure 1: System Architecture

2.1 Clock Fault Injection

The victim clock fault injection module (CFIM) allows for single cycle glitches to be injected on the victims system clock input. The injection module allows for clock cycle lengthening or shortening. To inject a fault, two clocks signals that are phase shifted with respect to each other are generated and then combined using a logical operation (AND, OR, etc). For normal victim clock operation, one of the clock signals is disabled. Figure 2 shows the operation of this fault.

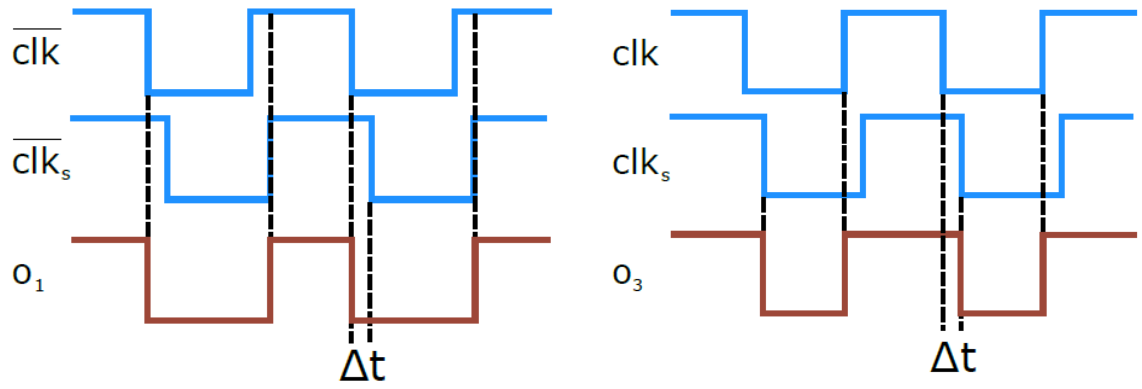

Figure 2: Clock Skew Examples [*a]

We constructed a system which uses two Xilinx Digital Clock Managers (DCM) to generate the victim clock. The logical negation of these clocks is also created. These four signals are the outputs of the FPGA to the fault injection board. The board contains a 74HCT08 and 74HCT32 for the AND and OR operations respectively. Since only one clock setup can go to the victim at a time, a manual pin jumper selection system is used. The output of the pin selector goes to a AD8045 unity gain operational amplifier (Op Amp) buffer. With this Op Amp, the victim clock can source 70 mA of current. Figure 3 below shows how the module functions.

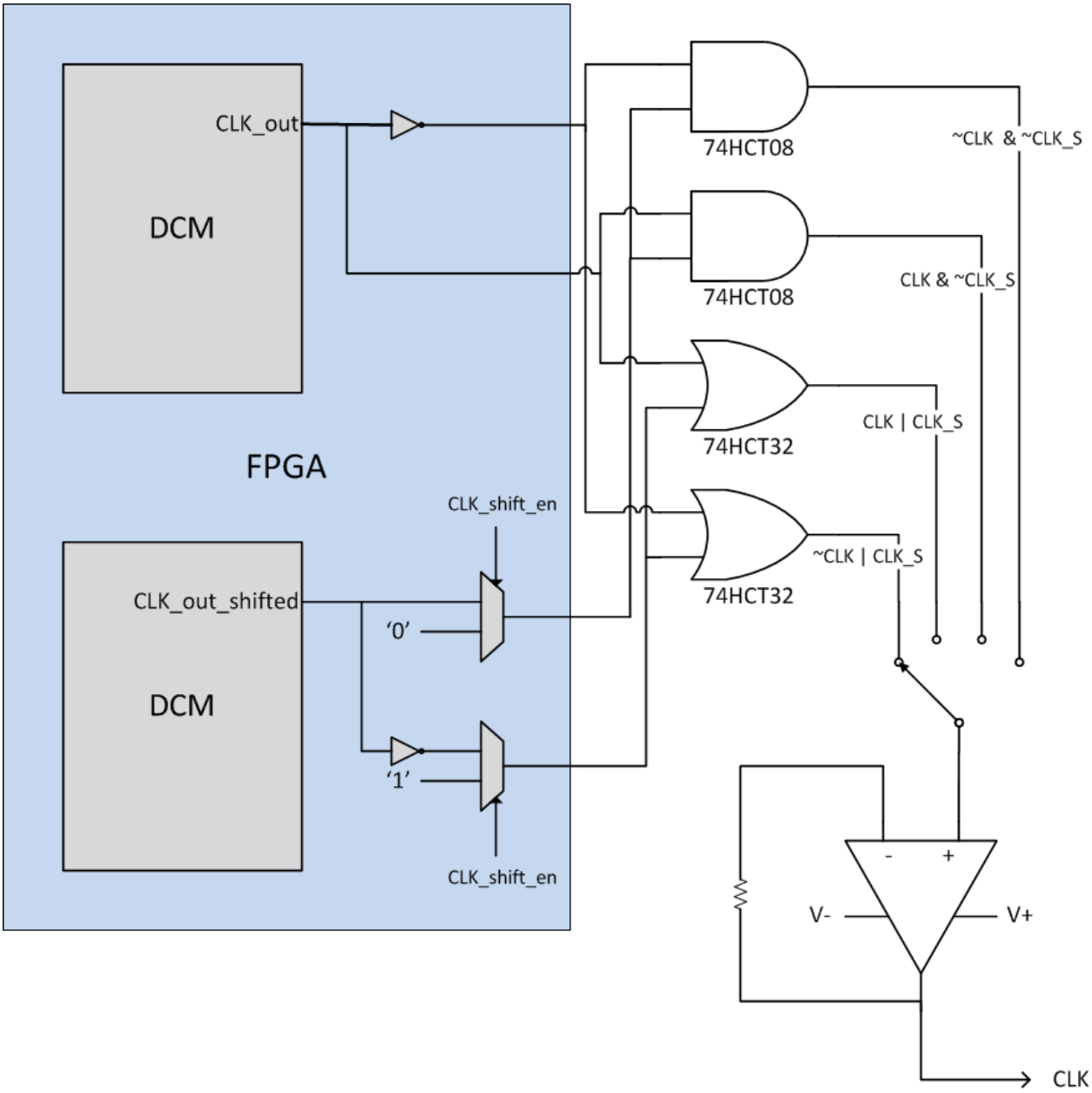

Figure 3: Functional diagram of CFIM

2.2 Power Fault Injection

We used the AD9708 chip as the D/A converter, and the AD8058 as the high frequency operational amplifier. The DAC AD9708 has a maximum update rate of 125 MSPS while the op amp AD8058 has a maximum frequency of 325 MHz. The circuit diagram for the power fault is shown below:

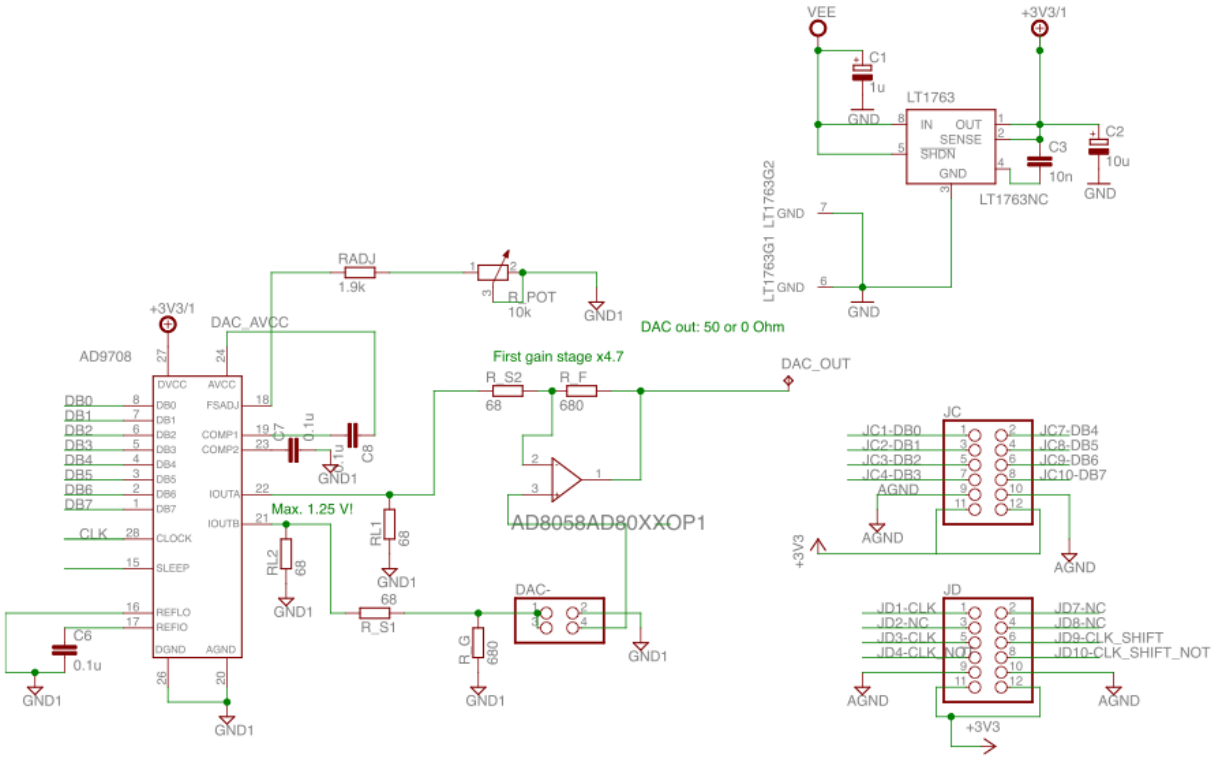

Figure 4: Power Fault Injection to Nexys 3 board

2.3 Data Output

A method needed to be developed in order to conserve space on the FPGA, as well as to have a minimum amount of delay while reading out the values from the ADC to the FPGA board and through the USB to the PC.
There were different approaches to doing this, one of them being to use BRAMs such as alternating FIFO blocks to pass the data out.

Another approach was to write ADC output values to the Micron Cellular RAM for the purpose of storing data off of the FPGA, while also attaining the read/write of 80MHz. This would free up some space on the host FPGA and could potentially reduce a bottleneck in the passing of data from the ADC output to the PC. Despite seeming a menial task, we encountered many difficulties, far worse than expected.

One of the main issues was finding a sample memory interface for communicating with the component. Digilent supports an example memory interface controller with EppCtrl for talking with the USB to a Nexys 2 (for use with Adept 1.10 and the TransPort software, an application for data transfers), which seemed simple to convert to Nexys 3 through modifications of the UCF file. Unfortunately, this memory interface only supported asynchronous communication to the Micron Cellular RAM. Below is the example FPGA design for use with Adept 1.10 and TransPort software.

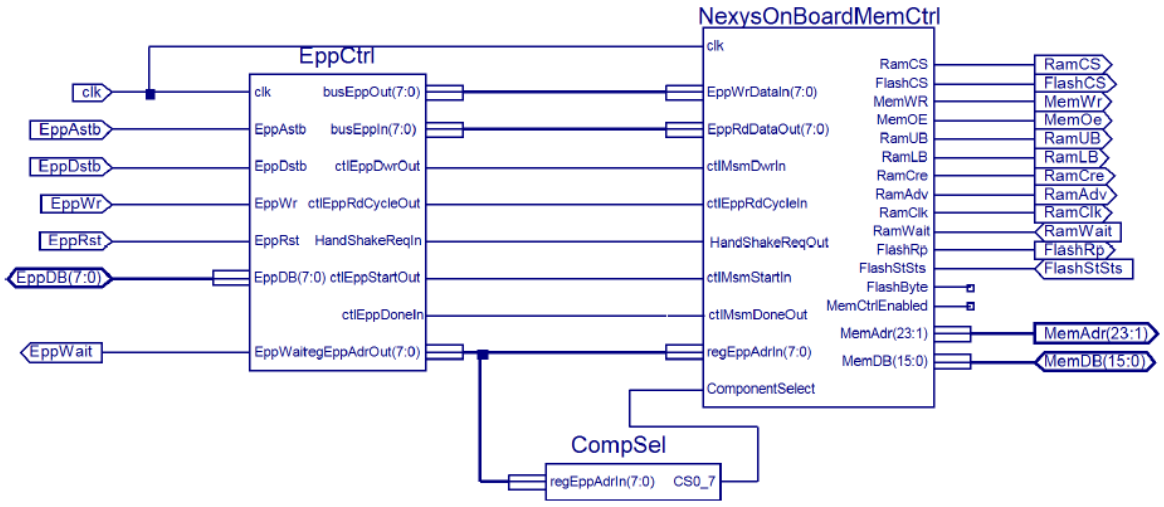

Figure 5: Nexys2 Asynchronous Memory Module [*b]

Due to not being able to find a sample memory interface for the “Burst” (synchronous 80MHz) mode of operation (no Burst mode seemed to be available or documented), we developed a means by which we could swap out memory modules depending upon our application as we developed and improved the speed of communication.

We decided to do a Hardware Abstraction Layer, where we could use different HAL and Memory Interface implementations as we developed them, and so a user could select whatever they needed for which scenario. In the process of developing the HAL, we used a modified NexysOnBoardMemCtrl module and added control and Hardware Abstraction Layers between the EppCtrl module, along with a stimulus module.

2.3.1 Dual FIFO Method

Our initial approach was to split the data between two FIFO blocks. With each clock cycle, we would allocate a matching data entry. This approach was found to require too much space on an attacking FPGA. If we were to have another FPGA used solely for taking ADC outputs, this would not have been an issue (although to do so would increase costs and one of the main goals of this project was to keep costs as low as possible). This option was scrapped due to its infeasibility as a result of memory usage. It is still documented and included in the project in case another FPGA is ever integrated into the system.

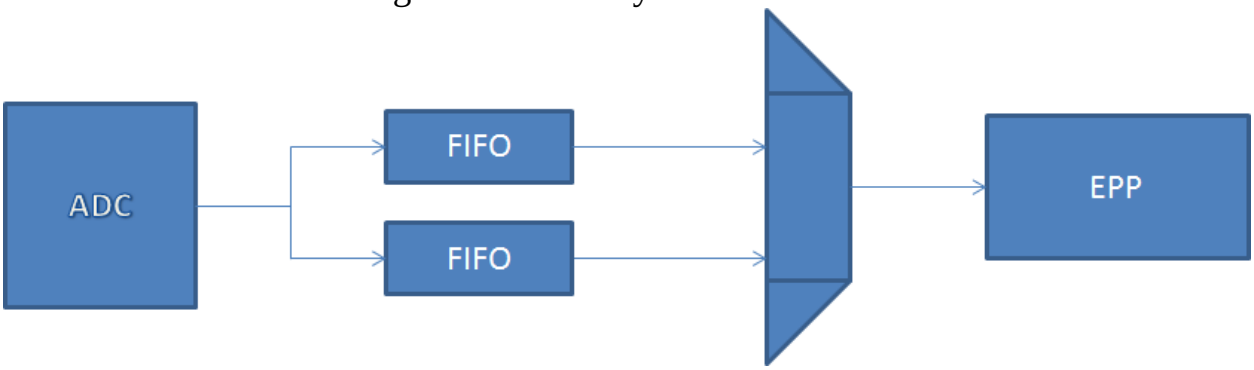

Figure 6: FIFO Scheme Logic

Below is the active version of this memory output interface, which takes fewer inputs from external logic in order to simplify host control logic outside of the combined memory architecture.

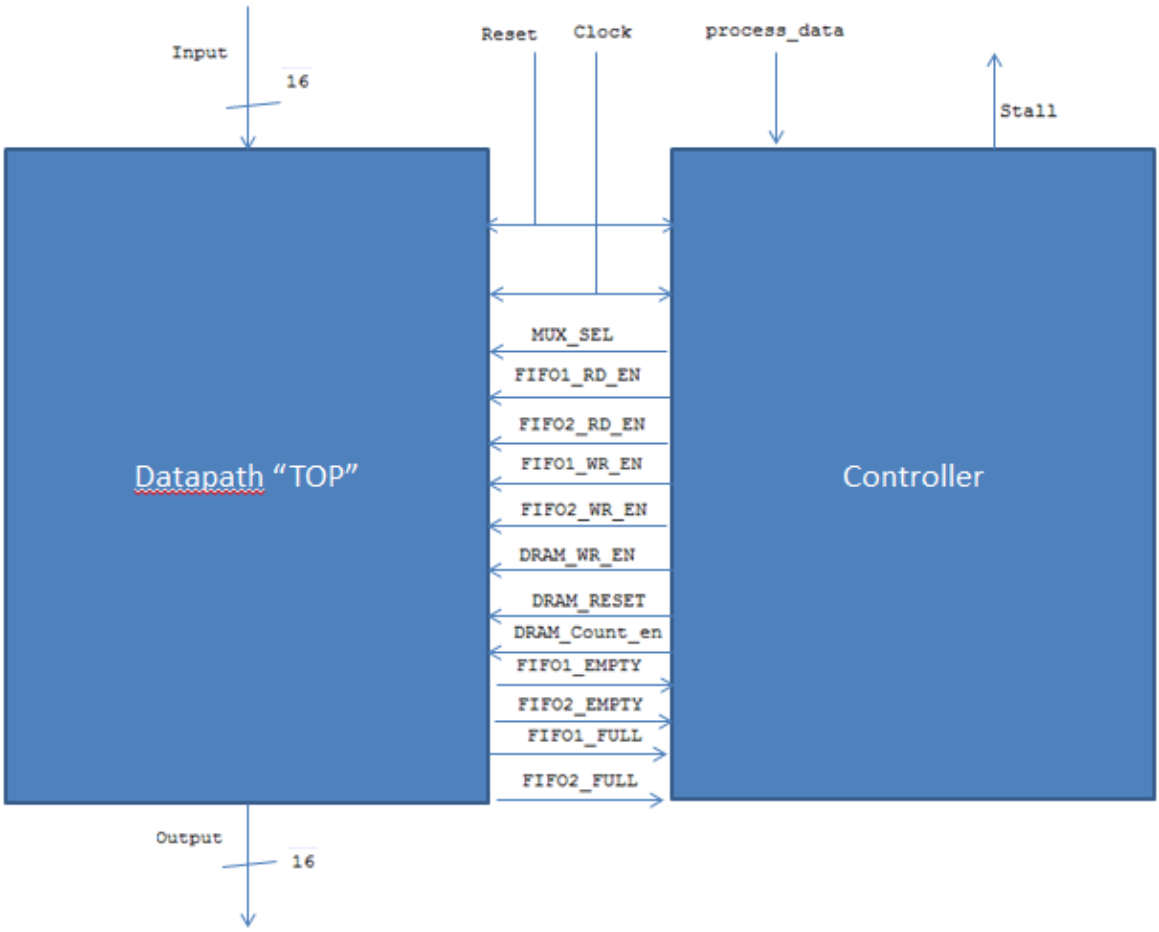

Figure 7: FIFO Scheme Datapath and Controller

2.3.2 Slow Memory Interface

Due to the lack of an actual memory interface implemented in the burst mode of operation, we decided to start with an asynchronous memory interface, with a much simpler design than that of the Digilent Nexys2 implementation. This interface unfortunately allows so far only for a maximum data rate of 10MHz.

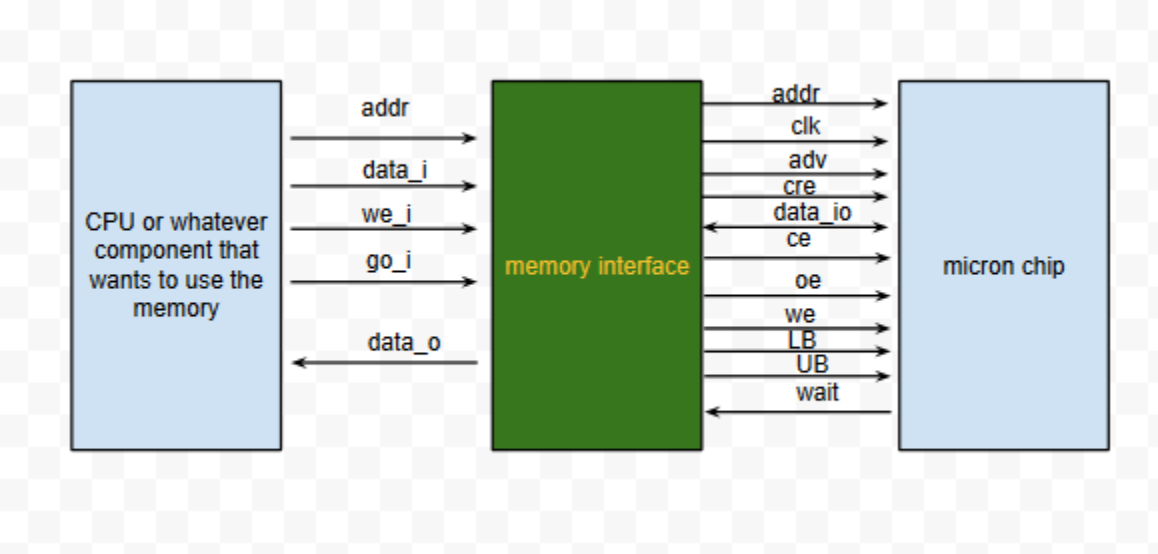

Figure 8: HAL Slow Memory Scheme [*c]

2.3.3 Hardware Abstraction Layer Method

With this approach, the memory interface and HAL are coupled for the purpose of supporting the architecture with different speeds of memory output control. This allows the for the decoupling of sample storage from reading data from the ADC/transferring samples to the computer. With this method, the storage medium interface is adaptable and easier to change in the future for different scenarios. This solution would allow us to integrate the slow memory Cellular RAM communication scheme into our final design integration for testing, and then to add the final burst mode after it was complete (i.e. just swap out the HAL and Memory Interface modules, which would be included in one block per scheme).

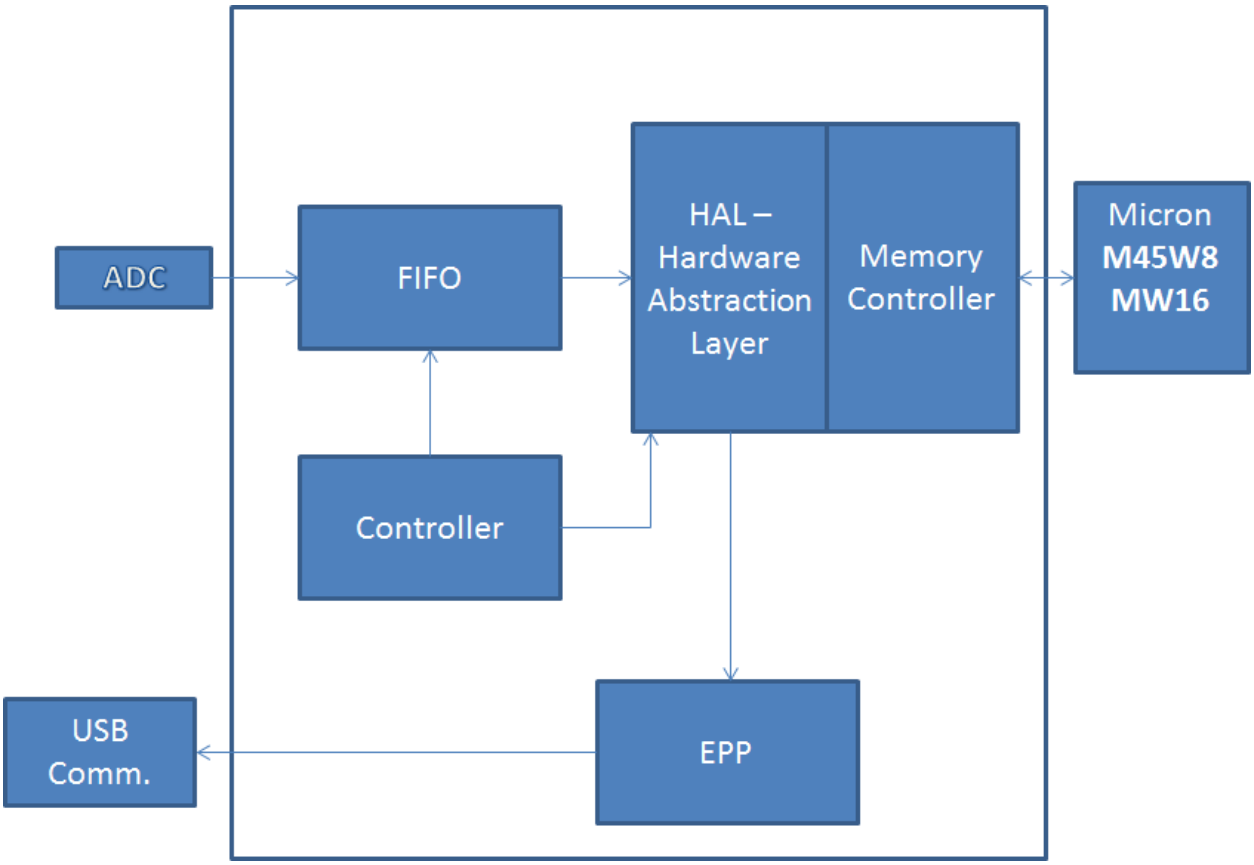

Figure 9: ADC Output HAL System Architecture

3. Discussion of Results

Integration remains a challenge for different parts of this project. However we have made significant progress at each level and have gained a solid understanding of all functionalities and components. We have also made improvements to the design of the code and simplified the analog design. Verification still remains an issue as we were unable to fully integrate and verify the functionality of our project.

4. Conclusion

Many aspects of electrical engineering were covered in this project, such as analog design, digital design, VHDL programming, understanding of FPGA architecture, review of cryptographic protocols such as RSA, AES, etc… Integrating all those technologies and continuing the progress of this project was a great learning experience. We hope that this project will make it easier and cheaper in the future for students and engineers to experiment with power analysis and fault injection, so that cryptographic functions will be carefully designed and tested. Ultimately, there will be more and better security measures implemented on embedded systems to preserve cryptographic secrets.

5. References

[1] Development of an Integrated Environment for Side Channel Analysis and Fault Injection, David Oswald
[2] Nexys3™ Board Reference Manual
[3] OPENADC Product Datasheet:
http://www.newae.com/files/openadc-datasheet.pdf
[4] XST User Guide for Virtex-6 and Spartan-6 Devices
[5] Micron MT45W8MW16BGX-701 Datasheet:
http://www.micron.com/parts/psram/cellularram/mt45w8mw16bgx-701-it?pc={BD8A72EA-2DC2-4B88-846E-7B59997A2D97
[6] Nexys 3 Spartan-6 FPGA Board Product Details:
http://www.digilentinc.com/Products/Detail.cfm?NavPath=2,400,897&Prod=NEXYS3
[7]Development of an Integrated Environment for Side Channel Analysis and Fault Injection, David Oswald, September 11, 2009:
http://www.emsec.rub.de/media/crypto/attachments/files/2010/04/da_oswald.pdf

6. Figure Citations

[*a] Development of an Integrated Environment for Side Channel Analysis and Fault Injection, David Oswald September 11, 2009

[*b] OnBoard MemCffg Reference Project

[*c] How to use cellular ram from Micron M45M8W16:
http://embsi.blogspot.com/2013/01/how-to-use-cellular-ram-from-micron.html