ECE-493 Final Paper

Differential Power Analysis Testbed

Differential power analysis is a type of side channel attack used to compromise a cryptographically secure system by obtaining the secret key the device uses to encrypt data. This method is proven effective on many different algorithms, including the advanced encryption standard, AES one of the most secure encryptions commonly available. Of the numerous different platforms capable of running such cryptographical algorithms, one of the most notable types is the field programmable gate array, or FPGA. FPGAs are used widely in industry as they allow one to inexpensively create different circuits. Currently, in order to conduct differential power analysis on an FPGA based system, a singular unit can be used which requires a user to take numerous measurements with a digital oscilloscope. This method is tedious, repetitive and inefficient. An existing automated solution exists but it is limited in its ability to test different chips as well as requiring the user to purchase a new board with each chip. The purpose of this project is to build a generic automated controller circuit to conduct these tests based upon a set of user defined parameters. The control software will be run on a computer which will have a graphical interface with which the user can control the FPGA system.

May 2nd, 2011

Dr. Kaps

Patrick Adams
Lindsay Walton

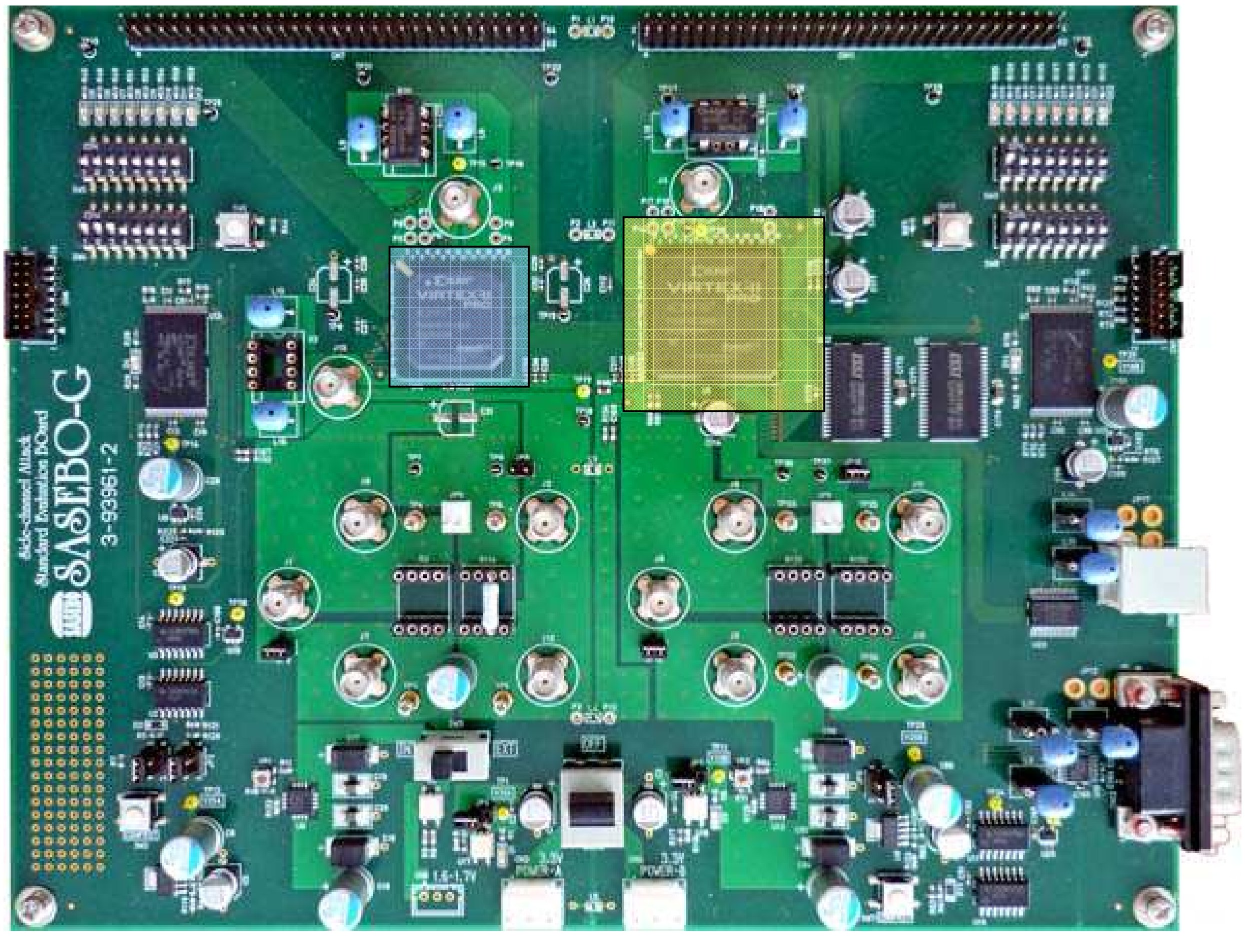
Contents

1 Introduction

With increasingly sensitive information being stored electronically, more focus has been placed on keeping data secure. To that end, encryption algorithms have become increasingly complex, rendering attacks difficult or, in some cases impossible. However less has been done to improve the platforms on which these algorithms are implemented. Common platforms such as Field Programmable Gate Arrays (FPGAs) have been found to be susceptible to power analysis attacks.

Differential power analysis (DPA) attacks are particularly effective against FPGA based cryptographic implementations. By examining the power consumption of a device implementing an encryption system, the attacker is able to use this side-channel attack to “look inside” the device without having to physically examine the hardware. This ability provides the attacker with newer, more subtle exploits that often enable him or her to comprise seemingly impenetrable cryptographic systems. This poses the cryptographic community with the challenge of developing implementations less susceptible to DPA attacks.

In order to harden FPGAs against DPA attacks, one must first have a manner by which to test a chip's susceptibility to the attack. The best platform currently available for such testing is the Side-channel Attack Standard Evaluation Board (SASEBO).



As shown above, the SASEBO features two FPGAs, one to act as a “controller” (highlighted in blue) which will drive the attack, and one to act as the “victim” (shown in yellow). This setup makes an excellent platform for testing side-channel attacks, except in the case of DPA.
In any DPA attack the physical characteristics of the victim plays a large role in effectiveness of the attack. As such, it is necessary for the developer to be able to run an attack on numerous different victim FPGAs, a feature that is not possible with SASEBO's victim-on-board setup. These, along with other issues to be discussed later, preclude SASEBO from being used as a viable testbed. Thus it is apparent that a new testing platform is needed in order to aid development of cryptographic implementations that resist DPA attacks. To that end, it is our intent to expand upon the SASEBO design, and develop a new testbed capable of evaluating an FPGA's vulnerability to DPA attacks.

2 Statement of Need

2a Basic Attack

For this project, we are to build an attack board that conducts differential power analysis (DPA) with the goal of attacking an FPGA running the Advanced Encryption Standard (AES-128) with a key size of 128 bits. The method as described by William Hnath and Jordan Pettengill in their paper: *Differential Power Analysis: Side Channel Attacks in Cryptography* focuses on obtaining the cipher key used in the encryption process by using statistical analysis and correlation on power traces recorded while the victim board conducts the encryption. The attack that we will be implementing will focus on the final step of the AES encryption algorithm which makes the computation for the attack significantly simpler as a longer component of AES is not implemented in the final phase. In AES, data is operated on one byte array[1] of 16 bytes at a time with a key array of equal size. The key that is used is generated by AES is based on the key schedule and each round of encryption includes part of the original key. Recording the final round for each subsequent encryption allows an attacker to get one byte of this key at a time instead of attacking the whole key at once. Once power consumption of the cryptographic chip is recorded by obtaining several thousand[2] traces, statistical correlation can be performed on the data. Using previous power measurements taken on operations with known keys, an attacker is able to make an accurate guess as to what the key is.

2b Problem Statement

While this method of attack is generally effective, it has several notable flaws. The first issue of this approach is that currently available attack platforms require manual operation. A user must manually obtain more than 10,000 power traces to obtain only 8 bits of the encryption key. For AES-128, this means the user must obtain a minimum of 4,096 separate power traces in order to obtain a full key. Hnath and Pettengill have been shown that an average of 10,000 traces is necessary to produce an accurate key. The user's task is further complicated by limitations of the digital oscilloscope used to obtain these power traces. Generally, the oscilloscope is incapable of storing the needed 4096 traces for a key. This forces the user to periodically dump the memory of the oscilloscope to the PC during the attack. Due to the extreme number of operations required to definitively obtain a key via this method, it is apparent that automation is necessary. Development of an automated testbed would allow one to attack a desired device quickly and efficiently.

In order to efficiently develop such a testbed, the use of open source tools has been chosen to expedite development time. Several aspects of the setup, as will be discussed later, would require communication between different elements. While proprietary drivers such as Digilent's Adept program are available, most are only compatible with a Windows operating system, and can be limited in functionality. Open source drivers, while often intended for use with Linux operating systems, could potentially be modified in order to remove any platform dependence.

[1] 4x4 array

[2] On average, more than 10,000 traces are needed to accurately get the key

This would not only allow the testbed to operate with numerous operating systems, it would also allow one to efficiently design a testbed by allowing one to optimize the drivers to work with their equipment, while reducing overall costs.

With these considerations in mind, it is apparent that a platform independent, automated testbed would enable more efficient DPA attacks on all encryption algorithms, if they are susceptible to DPA. Our initial focus will be on the AES encryption algorithm. This increased efficiency will lead to a better understanding of vulnerabilities in the implementation of a given algorithm which will help produce more secure implementation techniques for cryptographic modules in the future.

3 Approach and Requirements

3a Oscilloscope Control

Many thousands of measurements are required to successfully conduct a differential power analysis attack. Current oscilloscope memory limitations require a human operator to download the data once the memory is full and then restart the attack with a new set of data points. An ideal system would be able to pause while the data is uploaded to a data file on the computer. A requirement for our system is that the controller board be able to pause the cryptological processing unit while the data from the oscilloscope is saved on the computer. Additionally, most DPA attacks focus on a particular step in the encryption process, in our case, the substitution round of the final phase. As such, one does not require power measurements for the entire operation of the encryption system, just traces for the operations of interest. These areas of interest are not necessarily periodic and only occur for small periods of time. To increase efficiency, our system will have to detect when such an area of interest is occurring, at which point it would enable the oscilloscope’s recording mechanism. To control the oscilloscope, the control FPGA board we are designing must then have an interface connected to the oscilloscope which allows for more precise measurement and removal of erroneous data.

3b FPGA/PC Interface

Today’s standard for peripheral communication is the Universal Serial Bus standard or USB. This connection has a variety of uses, for an FPGA data transmission and the ability to program the FPGA as well as the system FLASH are the primary uses. A derived requirement is that the control FPGA board must have a USB interface, which will support programming as well as supporting data transmission.   While USB is universal, the driver software to program and communicate with the FPGA board is not. Many systems today use open source software which may not have an official supported interface driver. In order to be applicable to as many current researchers as possible our system must support the USB interface through Linux as well as Windows in addition to other open source development tools.

3c Control FPGA/Victim FPGA Interface

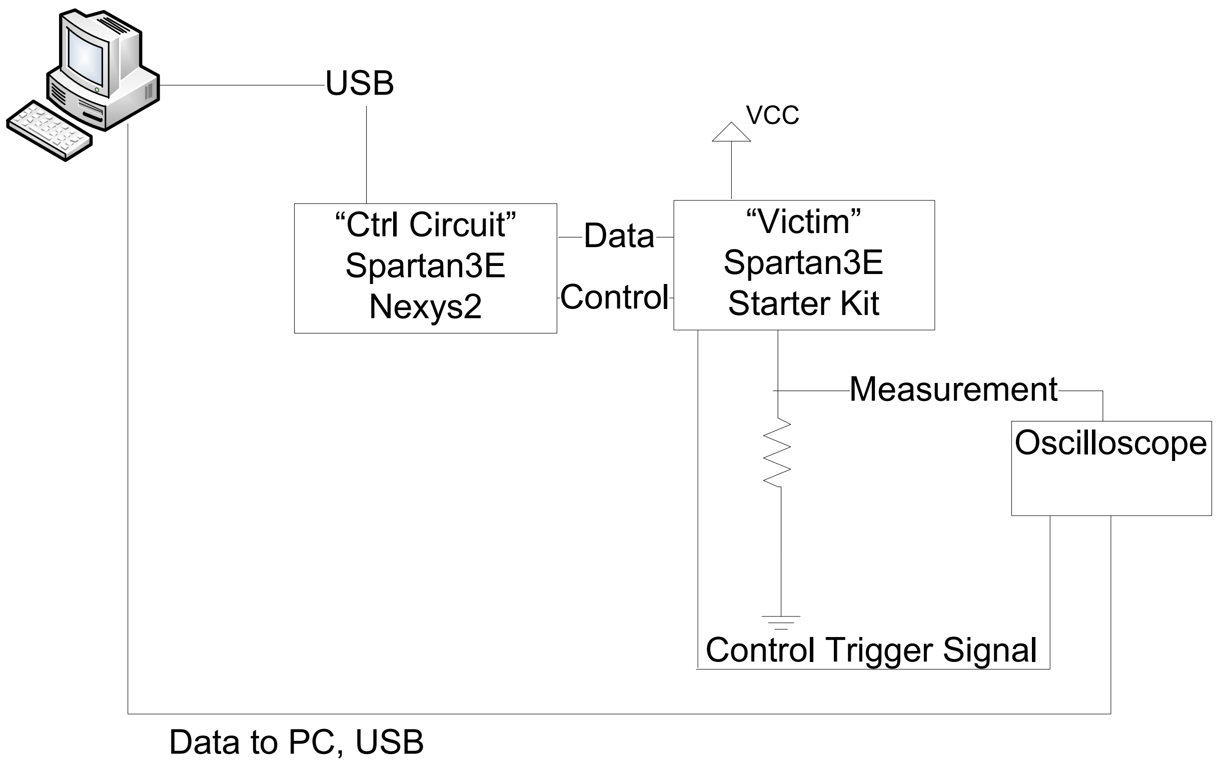
As part of being able to perform a DPA attack, our system must interface with many different cryptologic boards and must have a bridge which has hirose connections as well as optional pin out connections to allow for the optional cases that the victim board does not have a hirose connection to interface with our controller board. Our specific test system will interface with a Spartan 3E Starter Kit as the victim board.

3d Control Program (Software)

In order to automate the attack, a software program will be needed on a PC, which will interface with the FPGA boards. For greatest user flexibility a GUI software program would be preferred. To simplify the development process, our design incorporates a modified version of the SASEBO control software.

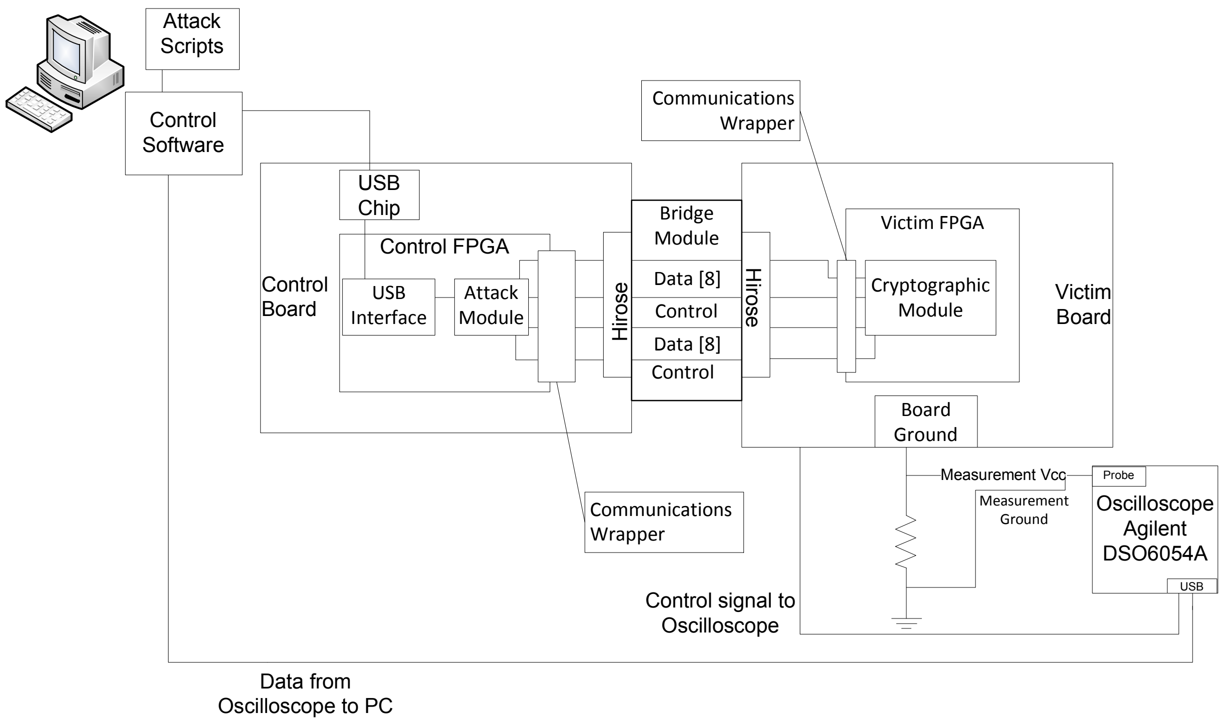
4 Design

4a Black Box



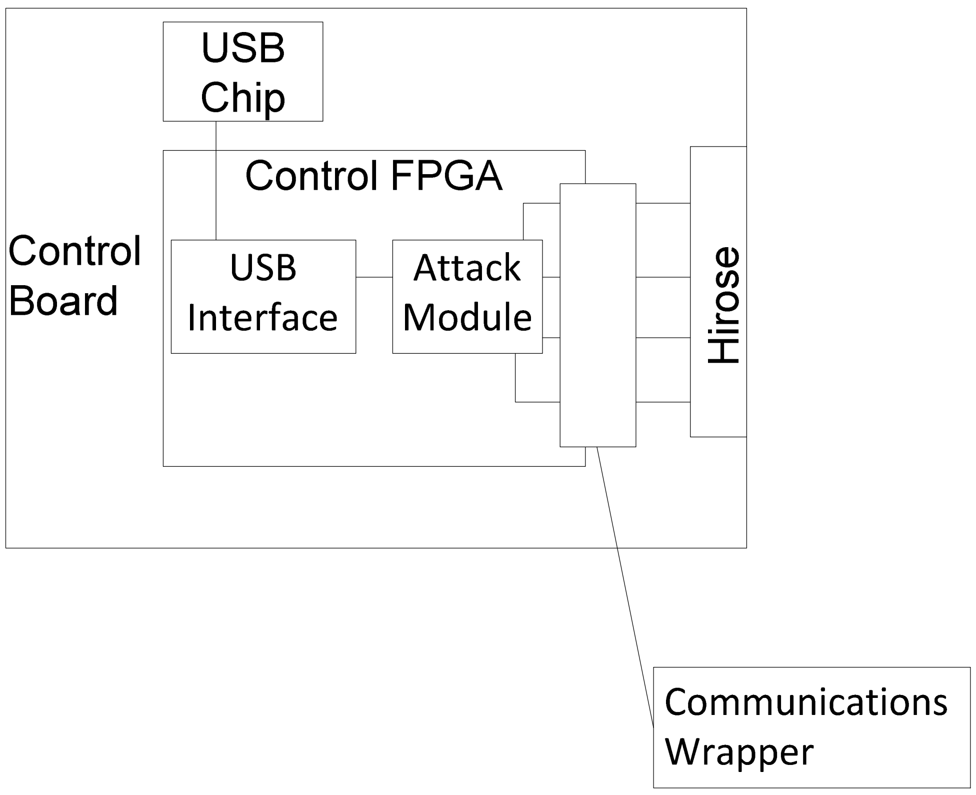
Our first step in designing the basic setup was to determine what components we will use. We decided to use the Nexys2 board as the controller because open source, Linux compatible drivers already exist, which will help us satisfy the requirement that the system be able to work with a Linux PC. The Spartan 3e Starter Board was selected for the victim due to its availability in the cryptography lab, thereby reducing overall cost.

4b White Box

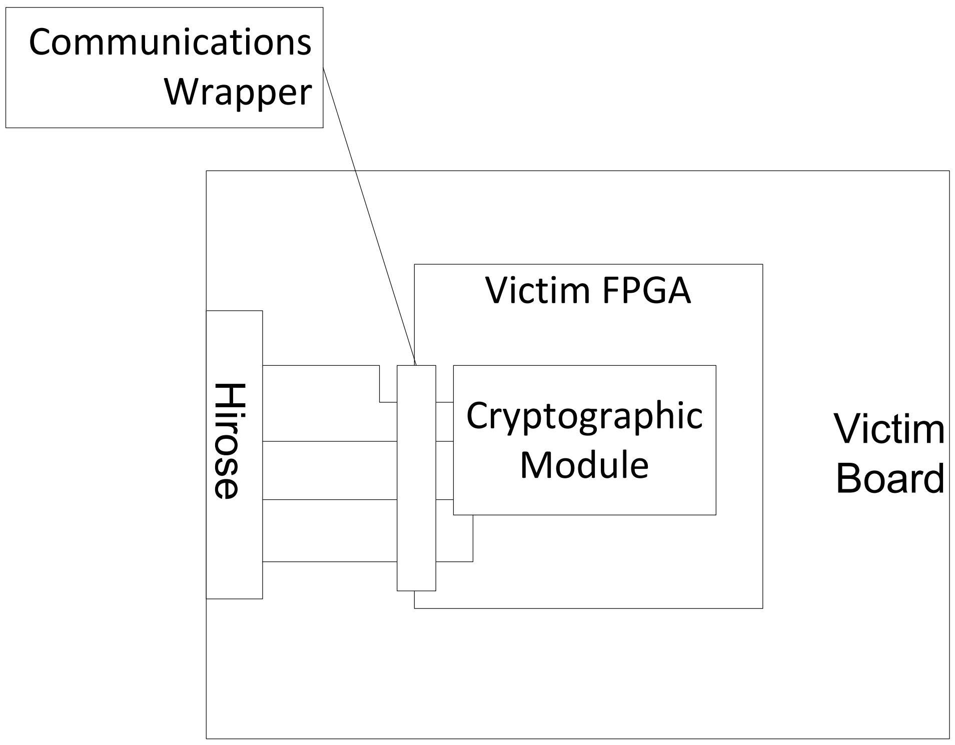


Our second preliminary design focused on identifying necessary interfaces between devices at run time. A USB interface will need to be setup to pass data to the controller FPGA. The controller and victim will communicate via their respective Hirose FX-2 connectors. The trigger signal will be determined by the PC's control software. The oscilloscope will send data to the PC via a USB connection. The PC control software will interface with preexisting post-analysis scripts in Matlab. The control software is based on an existing control module that was designed for SASEBO.

4c Unit Designs

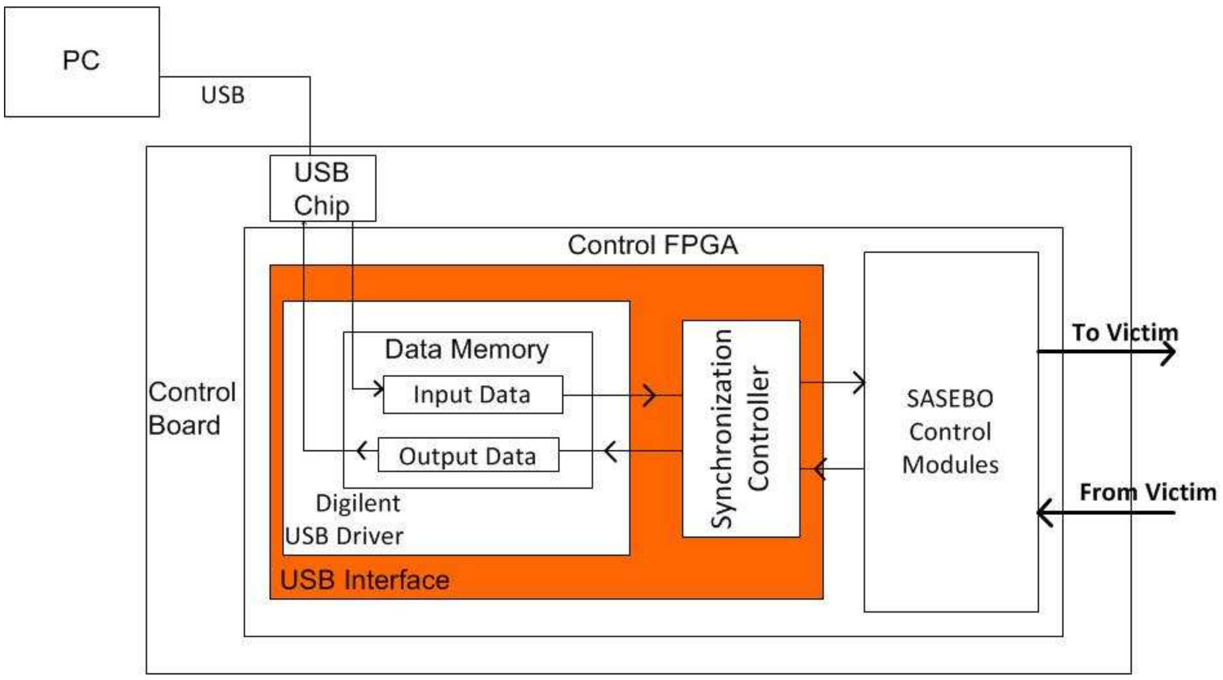


After designing the overall layout of the test bed, the individual FPGA's were considered. To meet our requirements, the controller will drive the victim. The controller will send control signals as well as data. Once the encryption operation is complete on the victim board, the controller will receive back the next byte of ciphertext. To simplify communications we decided to pursue a master/slave communications protocol, in which the controller would send its system clock across the bridge module to drive the victim. There is an optional clock connection on the bridge module which would allow a user to drive one or both boards at a desired frequency with an external clock. Through examination of established methods of sending data from a computer to an FPGA via USB, it became apparent that an adapter module would be needed to convert the raw USB signals into data usable by the controller. This adapter module sits between the USB chip and an asynchronous FIFO used to send data to the attack module on the control FPGA. All subsequent control signals are then handled by this attack module.



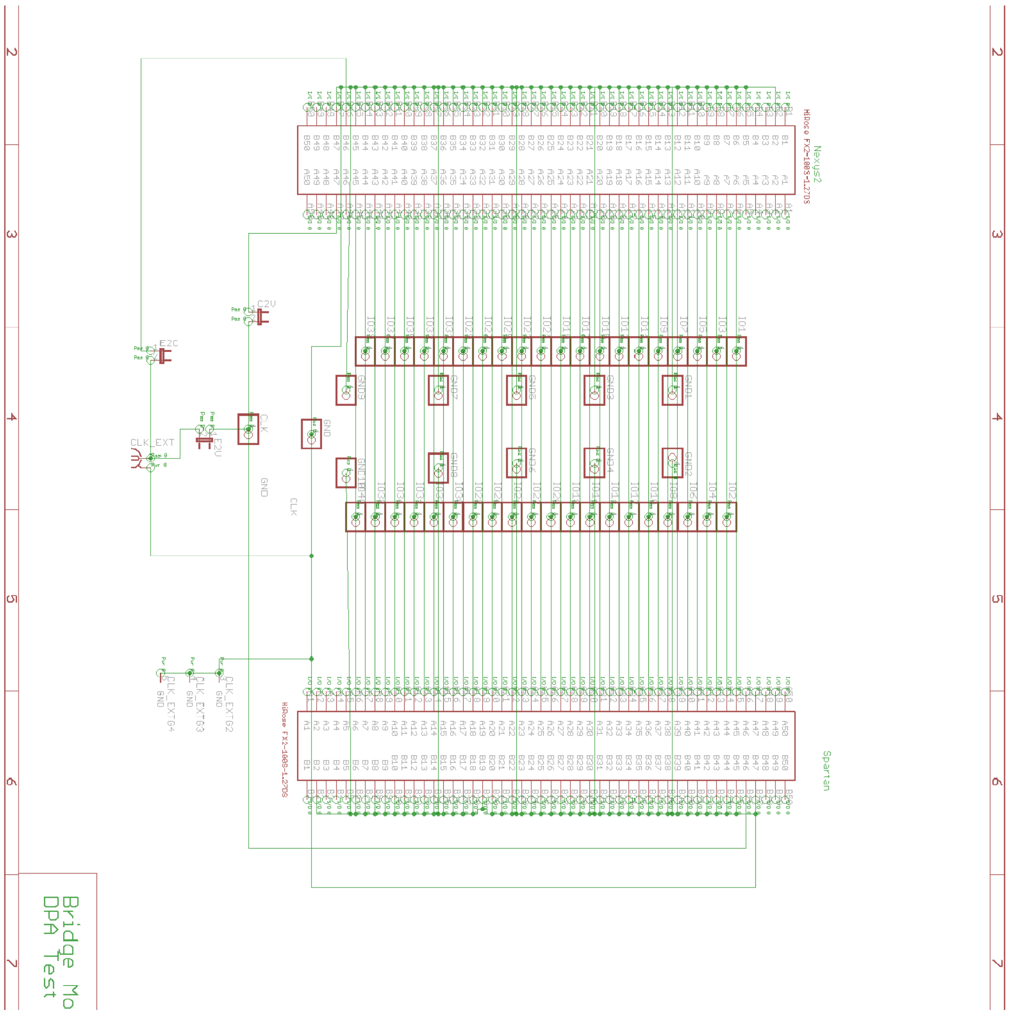
The victim board will house the cryptographic module that is being attacked. This module will be modeled after SASEBO hardware which includes synchronous FIFOs for signals over the Hirose connection.

4d USB Module



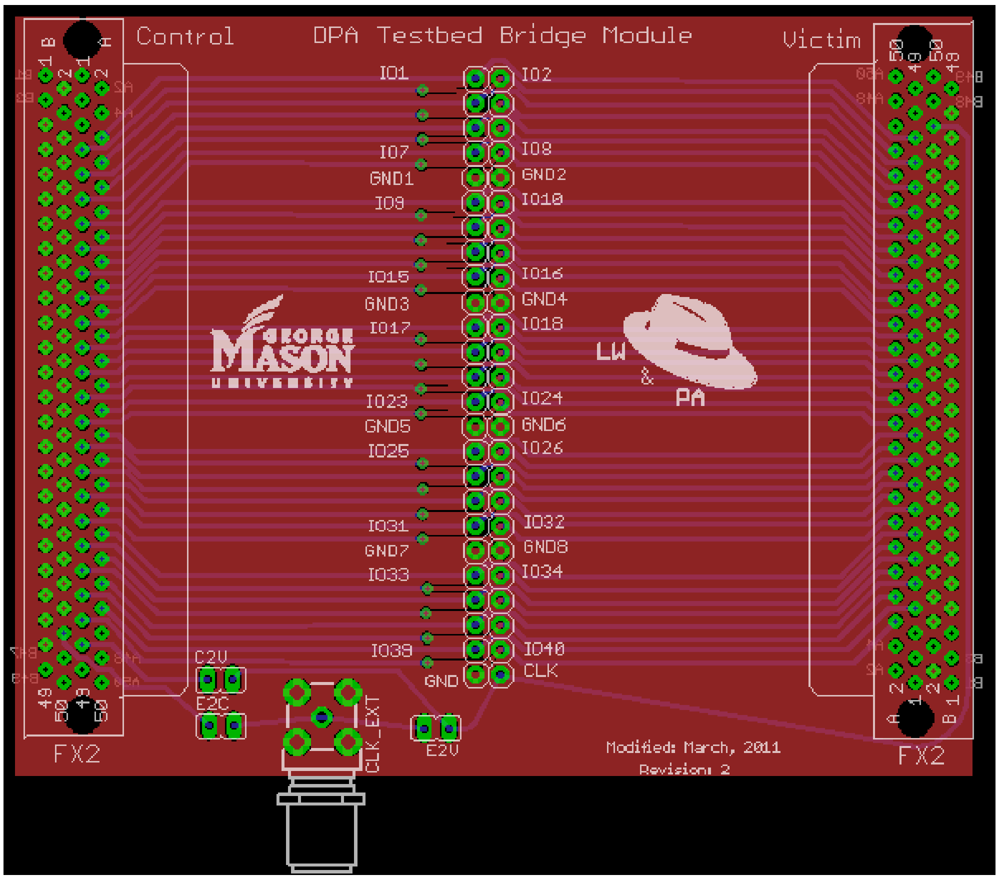
This is a more in-depth view of our USB interface. The Digilent communications driver was designed to allow the PC to directly read to, and write from a RAM on the FPGA. SASEBO, however, was designed to communicate to the PC via byte streams that had to pass through multiple FIFO's. In order to connect these two protocols, we reduced the RAM size down to two addresses: one for data sent by the PC that would be streamed to the victim, and one that would hold data from the victim that the PC could read. Various synchronization signals used by the SASEBO FIFO's, such as write enable and read enable, had to be controlled by the USB interface. To accommodate this, two synchronization controllers were connected to both the incoming, and outgoing FIFO's. Our testing confirmed that this USB interface was able to send data through the SASEBO control module correctly, using the Digilent protocol.

4e Bridge Schematic



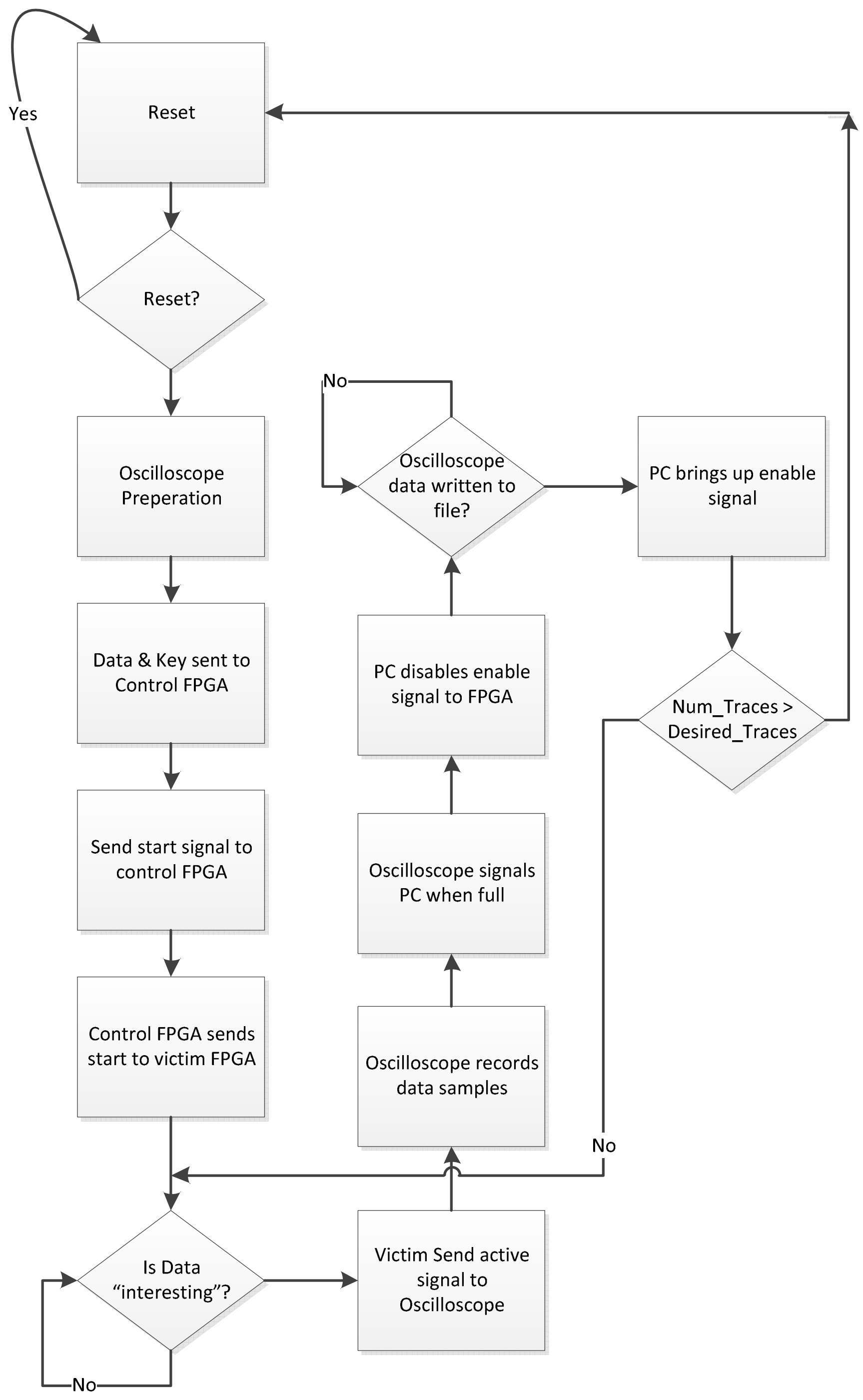
The schematic on the previous page is a logical layout of our bridge module. The two hirose connections on either end connect to one another as well as a generic pin out in the middle of the board. This pin out is designed to be connected to by a standard 50x2 pin ribbon cable. This pin out is included to interface with boards which do not feature hirose connections.

4f PCB Layout



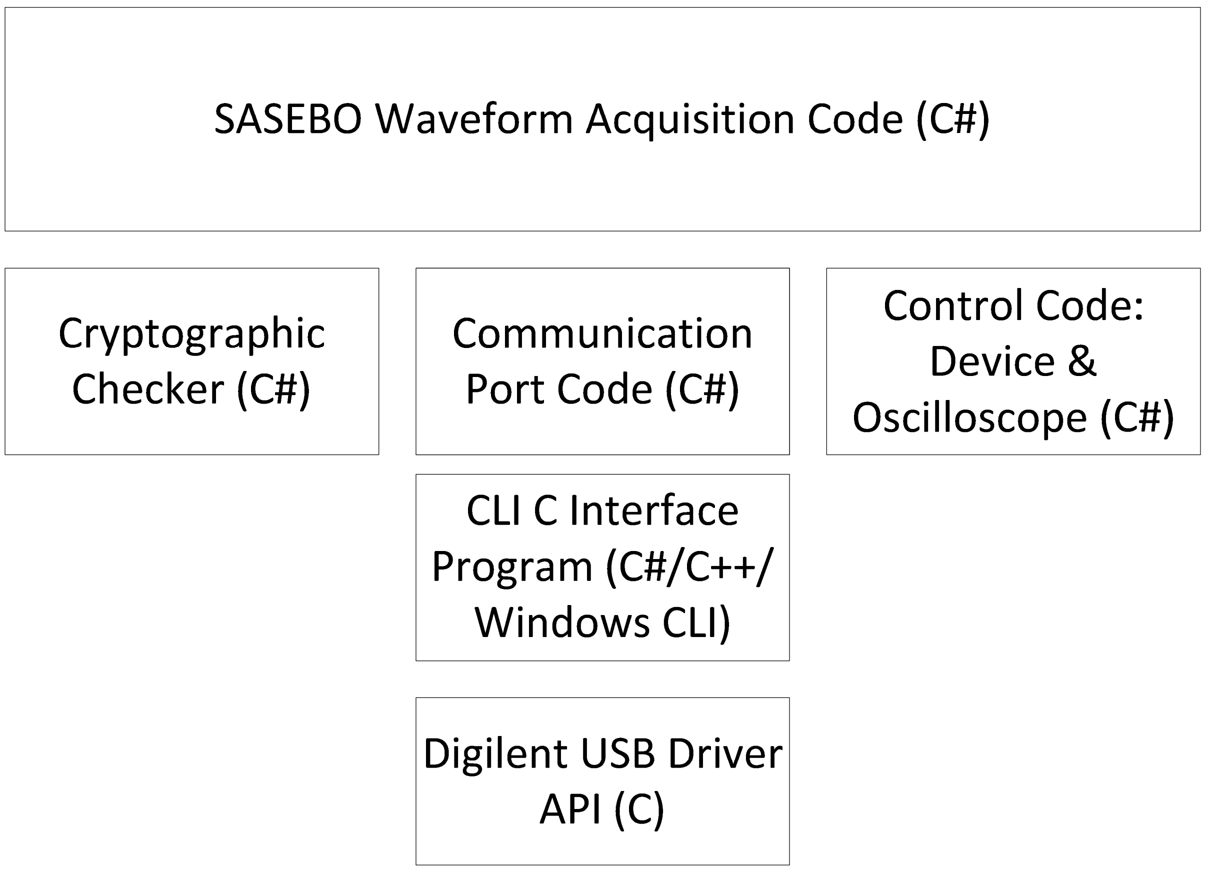
This is the physical layout of our bridge module. The top trace featured in red primarily consists of ground connections with a few communication signals. The design feature of having the ground connections flood the top layer is intended to reduce signal noise on the communication pins. The communication signals featured in blue are all located on the bottom layer of the layout. An external clock connection was added by our faculty supervisor's request which would allow an operator to drive one or both boards off of an external clock. Jumpers are used to control how this external clock is connected to the boards.

4f Flowchart


Reset
Yes
Reset?
Oscilloscope Preperation
Data & Key sent to Control FPGA
Send start signal to control FPGA
Control FPGA sends start to victim FPGA
Is Data "interesting"?
No
Victim Send active signal to Oscilloscope
Oscilloscope records data samples
Oscilloscope signals PC when full
PC disables enable signal to FPGA
Oscilloscope data written to file?
No
PC brings up enable signal
Num_Traces > Desired_Traces
No

The flow chart above diagrams our current protocol for the operating the attack. After all elements are placed in reset, data and control information is passed to the control board from the PC. The data is then passed to the victim, which encrypts the data according to control signals received from the controller. At the appropriate time, the PC set a trigger signal, causing the oscilloscope to record power traces until the trigger signal falls. In the event that the oscilloscope's memory is exhausted, execution of the rest of the testbed is suspended while the power readings transfer from the oscilloscope's internal memory to a data file on the PC. When the full attack has completed, back-end analysis scripts are invoked to process the data.

4h Software Design


SASEBO Waveform Acquisition Code (C#)
Cryptographic Checker (C#)
Communication Port Code (C#)
Control Code: Device & Oscilloscope (C#)
CLI C Interface Program (C#/C++/ Windows CLI)
Digilent USB Driver API (C)

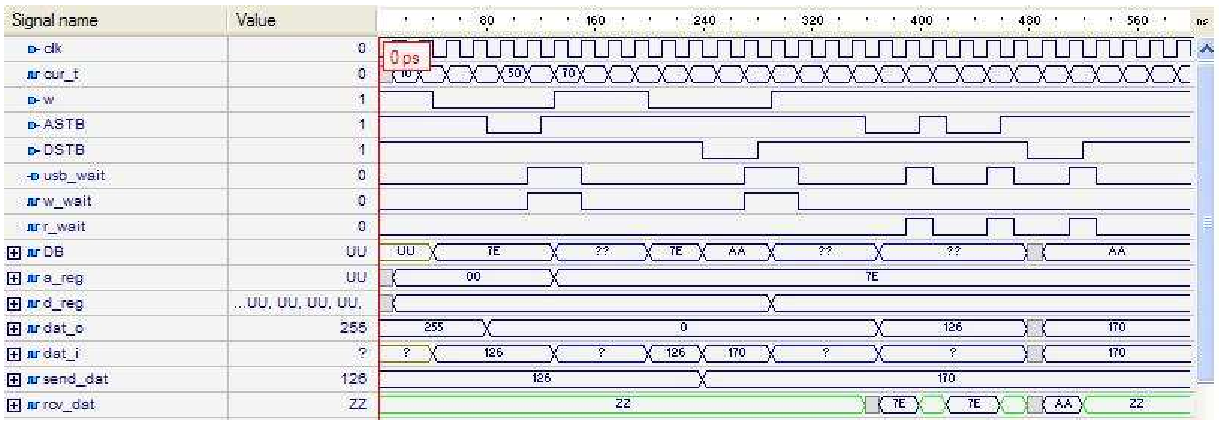
The overall design of the software for the PC control was to maintain SASEBO’s core functionality and components. As such, only the communications port section of code needed to be modified. Due to C#’s limitation on pointer handling, an interface program was written in C++ which was able to be called from the C# SASEBO software. This C++ program interfaced directly with the Digilent USB driver API with headers which were provided in the C programming language.

5 Cost Figures

Unit	Cost	Quantity	Subtotal
Hirose Connector	**$12.00**	**4**	**$48.00**
Nexys2 Board (Donated)	**$0.00**	**1**	**$0.00**
Spartan 3E Starter Board (Donated)	**$0.00**	**1**	**$0.00**
PCB Fabrication	**$50.00**	**1**	**$50.00**
Code Time (C#)	**$43.00**	**80**	**$3,440.00**
Schematic Design	**$31.00**	**42**	**$1,302.00**
Code Time (VHDL)	**$36.00**	**100**	**$3,600.00**
Debugging Time (System)	**$30.00**	**45**	**$1,350.00**
Total			**$9,790.00**

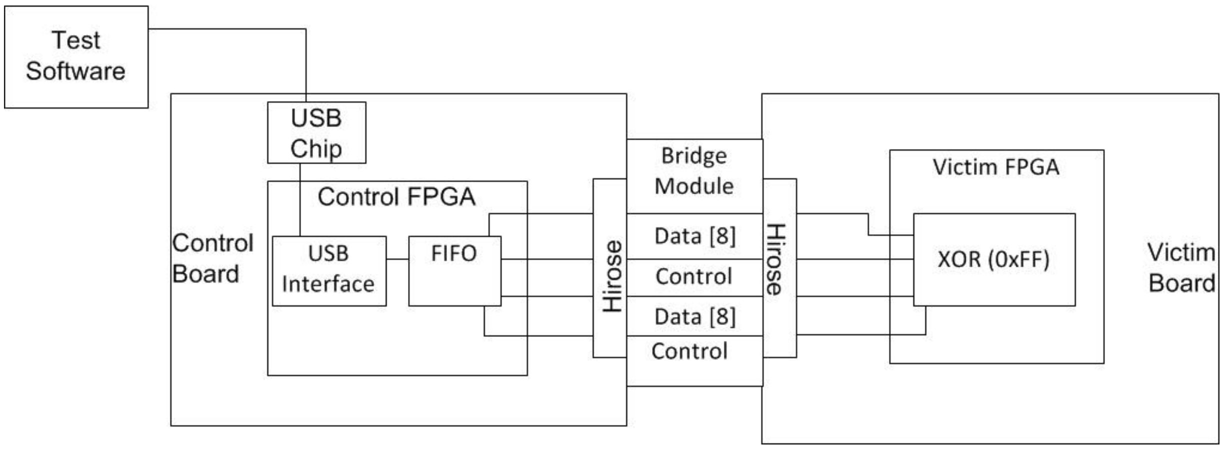
6 Results

6a USB



This is a simulation of the USB interface and it’s connection to the PC. The w, ASTB and DSTB signals are used by the PC to tell the board what kind of operation to perform, and DB is the bidirectional bus used to transfer data. The usb_wait signal is used by the board to synchronize communication with the PC. In the case of a write request from the PC, the board will bring usb_wait to acknowlage the request, and will keep this signal high until the data has been passed to the SASEBO control module. In the case of a read request from the PC, the board will delay raising the usb_wait signal until a byte from SASEBO has been loaded onto DB.

6b Communications Testing



For prototyping and final evaluation of success, we determined that a testing circuit would be beneficial. The overall design is shown above the necessary components pictured. The test software communicated with a SASEBO like control unit via USB which then passed data to our victim. Once on the victim, to ensure that data was in fact being processed, we inverted all of the bits of the data and sent it back. The control unit then passed the information back to the test software via the USB device.

6c SASEBO Integration

The final goal of our project was to have our solution be capable of running a victim FPGA through a series of encryptions which would allow us to run a DPA attack. As much of the code for the hardware and software was already implemented by SASEBO, we just had to integrate our new or redesigned components with theirs and ensure functionality. What we found was that while we could send information through the SASEBO waveform software, we would not get any information back. In order to debug this, we wrote an additional piece of software in which we controlled the information being sent. This software also did not get any data back from the AES core on our victim board. After using a logic analyzer to trace the signals, we found that the data was in fact getting passed by our control module to the victim, however once the victim received it no operation was done. We unfortunately ran out of time to find the error; however, we believe that it may be involved with the way that their AES core is initialized, or that the synchronous FIFO on the victim board was not coded properly, it appeared that data would inconsistently appear on the victim. We found this error by putting one byte of data on the LEDs on the victim board. Should improvements be made to our design, they must first overcome the communication problems with the SASEBO AES core or replace it with a different cryptographic module.

7 Maintainability Maintenance and Retirement

7a Potential Use

As mentioned before this platform is intended for anyone who wishes to test any cryptographic implementation’s susceptibility to a DPA attack. When completed, our test bed will first be used by Dr. Kaps of the ECE department to aid him and the Cryptographic Engineering Research Group (CERG) with their research on DPA attacks. Our final product could potentially be turned into a commercial product but with limited distribution to academia and government. Since our goal is to make a system that can attack any hardware, one would only need the code we create in order to set up one’s own test bed using their hardware implementations for both the control and victim.

7b Design Alternatives

Alternatives for our current design that could impact costs are definitely the various implementations of the bridge connector. There are several low cost implementations or wires that could be used including making our own printed circuit board. All of these implementations have a significantly different cost as well as man hour requirement. Currently our project uses only ROHS certified components and is designed to be as environmentally friendly as possible. It is very difficult for our project to find anything that is not pre-designed to have minimal environmental impact and thus any changes we make will likely have little to no impact on the environment.

An alternative would be to use a control board other than the Nexys2. This board was chosen for our project because of its low cost as well as Linux compatibility. The control FPGA on this board is significantly more powerful than required so a cheaper board could be used if the appropriate external connections were present. An additional benefit of the Nexys2 is the presence of the hirose connector for interfacing with other Digilent boards such as our chosen victim for testing, the Spartan 3E starter kit.

7c Maintenance Considerations

Maintainability for our project is designed to be low and is expected to be able to be used for a longer period of time without updates. Should a designer want to update or add a new module to our project, our choice of an FPGA based solution means that the designer just needs the original source code and they can compile their module into our design. As the design is an automated testing solution that can be changed based upon inputs from a computer for each test little updating is foreseen.

7d End of Life Considerations

Should one of the boards in this platform reach the end of its lifetime, it can easily be replaced by configuring a new FPGA board with our platform code. In the case that the bridge module should need replacing, we intend to include the PCB layout for the module in our final documentation, so that one may recreate this connector easily. Various companies offer PCB recycling services that would allow for more environmentally friendly disposal of retired components.

Works Cited

Hnath, William and Jordan Pettengill. Differential Power Analysis Side-Channel Attacks in Cryptography. Senior Design Project. Worcester: Worcester Polytechnic Institute, 2010.

Mangard, Stefan, Elisabeth Oswald and Thomas Popp. Power Analysis Attacks. New York: Springer Science+Business Media, 2007.

Velegalti, Rajesh and Panasayya Yalla. Differential Power Analysis Attacks. Fairfax: George Mason University, 2009.

—. "DPA Attacks on FPGA Implementation of AES." Fairfax: George Mason University, 2010.

Appendix A: Proposal

ECE-492 Proposal

Differential Power Analysis Testbed

Differential power analysis is a method of side channel attack which allows anyone capable of monitoring a cryptological system to use a set of known data points to obtain the secret key used to encrypt data on that specific device. Differential Power Analysis is proven effective on many different algorithms, including AES-128. There are many different systems capable of running such cryptological algorithms, one notable type being the field programmable gate array, or FPGA. FPGAs are used widely in the academic community as research tools as they allow one to inexpensively prototype many different circuits on one chip. Currently in order to conduct differential power analysis on an FPGA based system, a singular unit is used, requiring a user to take numerous measurements using an oscilloscope. This method is tedious and requires a user to repeat many times in order to ascertain the cryptological key with any certainty. The purpose of this project is to build an automated controller circuit to conduct these numerous tests based upon a set of user defined parameters. The control system will be run on a computer which will have a command line interface with which to control the FPGA system.

October 15th, 2010

Dr. Kaps

Patrick Adams
Lindsay Walton

Introduction

Differential Power Analysis (DPA) is a type of side channel attack, an attack that gains information based upon the physical implementation of the cryptological system. DPA is a method of attack that measures power consumption and can be successfully conducted on various symmetric cryptological systems as a method of being able to illicitly procure the encryption key. A symmetric cryptological device means one in which the key used for encryption and decryption is identical. DPA attacks are successful against many different encryption algorithms including RSA, DPA[3], and AES-128[4]. This project focuses on AES-128 DPA attacks on FPGA based systems which require a lot of human operator time as there is a set of specific power points of interest after which the oscilloscope memory is full and the test must be restarted. The board that is currently used for this type of research is the side-channel attack and evaluation board (SASAEBO). This board integrates two FPGA modules, one controller FPGA and one crypto FPGA which allows for an operator to set operational parameters and then allow the control module to conduct the attack.

Statement of Need

Basic Attack

For this project, we are focusing on using DPA to attack an FPGA running AES-128, using the method described by William Hnath and Jordan Pettengill in their paper: “Differential Power Analysis: Side Channel Attacks in Cryptography.” This method focuses on obtaining the cipher key used in the final step of the AES encryption process. Focusing on this final step is advantageous because the Mixed Columns step is omitted, which makes the the attack computationally faster. After sending a 16 bit message to the test chip, the resulting 128 bit ciphertext is broken into individual bytes. Since AES encrypts each byte individually, one can analyze a given byte of the ciphertext to obtain the corresponding byte in the encryption key. For a given byte, one attempts to decrypt each byte using an arbitrary, “guessed,” value for the key. This attempted decryption must then be repeated 256 times, each with a unique key guess, in order to obtain power readings for all possible values of the key. Each of these 256 traces are then compared against the actual AES power trace for the same byte to determine which of the key guesses is actually correct.

Problem Statement

While the afore mentioned method is generally effective, it has several notable flaws. The first issue of this approach is that the attack must be manually operated. A user must manually run 256 decryption attempts to obtain only 8 bits of the encryption key. For AES-128, this means the user must obtain a minimum of 4,096 separate power traces in order to obtain a full key.

[3] (Mangard, Oswald, & Popp, 2007)
[4] (Velegalti & Yalla, DPA Attacks on FPGA Implementation of AES, 2010)

Hnath and Pettengill have been shown that an average of 10,000 traces are necessary to produce an accurate key. The user's task is further complicated by limitations of the digital oscilloscope used to obtain these power traces. Generally, the oscilloscope is incapable of storing the needed 4096 traces for a key. This forces the user to periodically dump the memory of the oscilloscope to the PC during the attack. Due to the extreme number of operations required to definitively obtain a key via this method, it is apparent that automation is necessary. Development of an automated testbed would allow one to attack a desired device quickly and efficiently.

In order to efficiently develop such a testbed, one would require the use of open source tools. Several aspects of the setup, as will be discussed later, would require communication between different elements. While proprietary drivers such as Digilent's Adept program are available, most are only compatible with a Windows operating system, and can be limited in functionality. Open source drivers, while often intended for use with Linux operating systems, could potentially be modified in order to remove any platform dependence. This would not only allow the testbed to operate with numerous operating systems, it would also allow one to efficiently design a testbed by allowing one to optimize the drivers to work with their equipment, while reducing overall costs.

With these considerations in mind, it is apparent that a platform independent, automated testbed would enable more efficient DPA attacks on the AES encryption algorithm. This increased efficiency will lead to a better understanding of vulnerabilities in the algorithm which will help produce stronger encryption algorithms in the future.

Approach and Requirements

Oscilloscope Control

Many thousands of measurements are required to successfully conduct a differential power analysis attack. Current oscilloscope memory limitations require a human operator to download the data once the memory is full and then restart the attack with a new set of data points. An ideal system would be able to pause while the data is saved off to a file on the computer. A requirement for our system is that the controller board would be able to pause the cryptological processing unit while the data from the oscilloscope is saved on the computer. The method of which a DPA attack is conducted requires taking measurements of specific areas of interest. These areas of interest are not a periodic occurrence and only occur for small periods of time. Our system will have to detect when such an area of interest is occurring, at which point it would enable the oscilloscope’s recording mechanism. To control the oscilloscope, the control FPGA board we are designing must then have an interface connected to the oscilloscope which allows for more precise measurement and removal of erroneous data.

FPGA/PC Interface

Today’s standard for peripheral communication is the Universal Serial Bus standard or USB. This connection has a variety of uses, for an FPGA data transmission and the ability to program the FPGA as well as the system FLASH are the primary uses. A derived requirement is that the control FPGA board must have a USB interface, which will support programming as well as supporting data transmission. While USB is universal, the driver software to program and communicate with the FPGA board is not. Many systems today use open source software which may not have an official supported interface driver. In order to be applicable to as many current researchers as possible our system must support the USB interface through Linux as well as Windows in addition to other open source development tools.

Control FPGA/Victim FPGA Interface

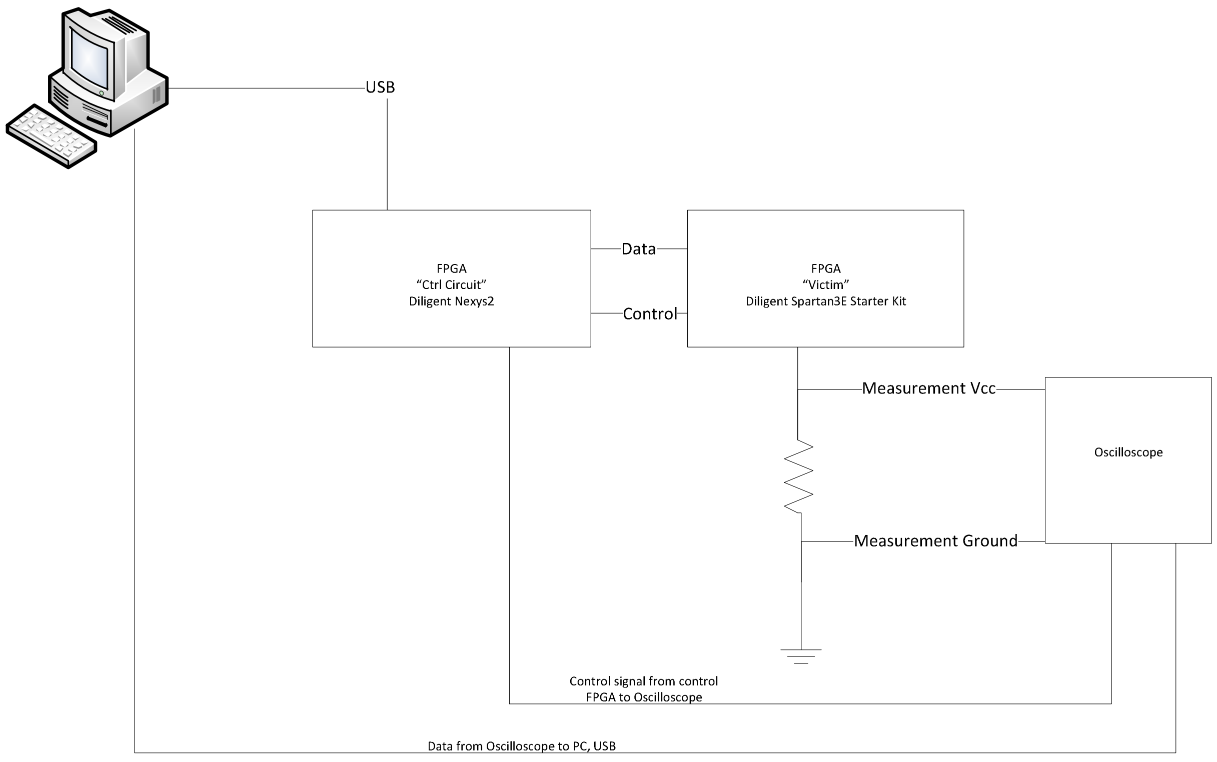
As part of being able to perform a DPA attack, our system must interface with many different cryptologic boards and must have a bridge which has hirose connections as well as optional pin out connections to allow for the optional cases that the victim board does not have a hirose connection to interface with our controller board. Our specific test system will interface with a Spartan 3E Starter Kit as the victim board.

Command Line Program

In addition to supporting the controller board on Linux, a control program must be supplied that works on all operating systems. This program must read configuration data about the test from a file, and if no file is provided, conduct the attack based on a specified set of default values. The program must also be able to automatically download the data from the oscilloscope while pausing the operation of the measurement circuit.

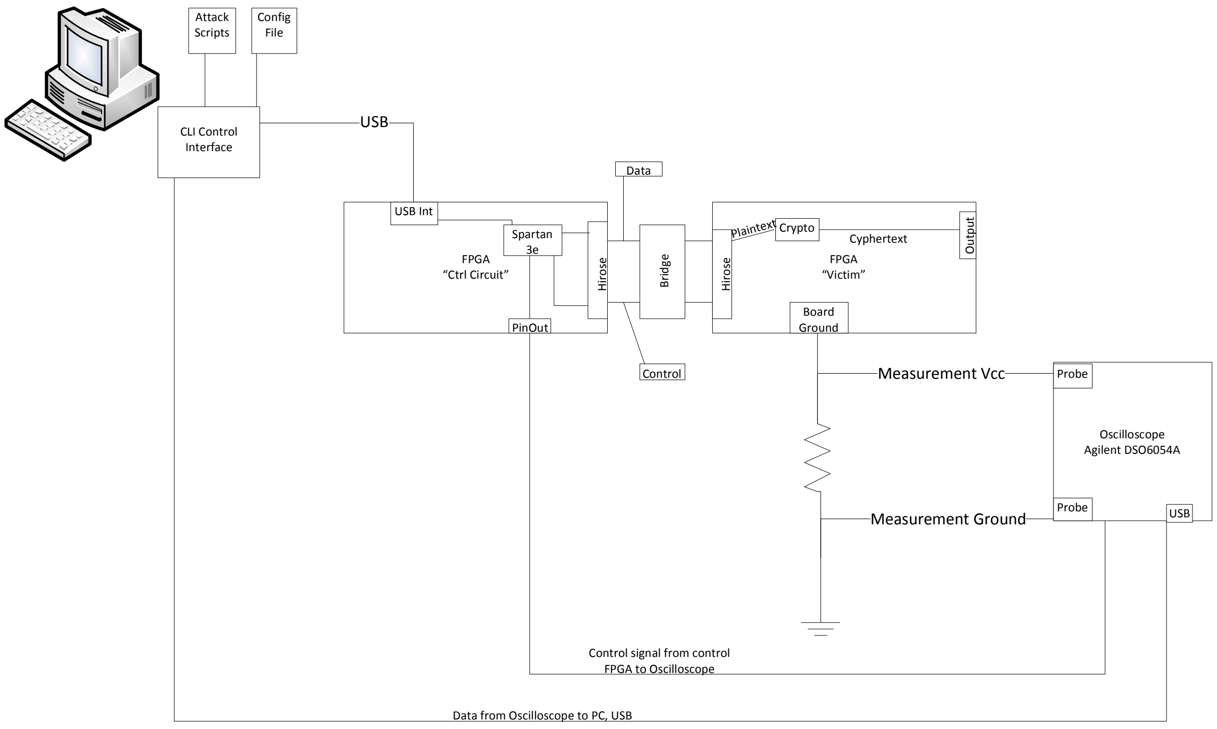
Preliminary Design

Black Box


USB
FPGA
“Ctrl Circuit”
Diligent Nexys2
Data
Control
FPGA
“Victim”
Diligent Spartan3E Starter Kit
Measurement Vcc
Oscilloscope
Measurement Ground
Control signal from control FPGA to Oscilloscope
Data from Oscilloscope to PC, USB

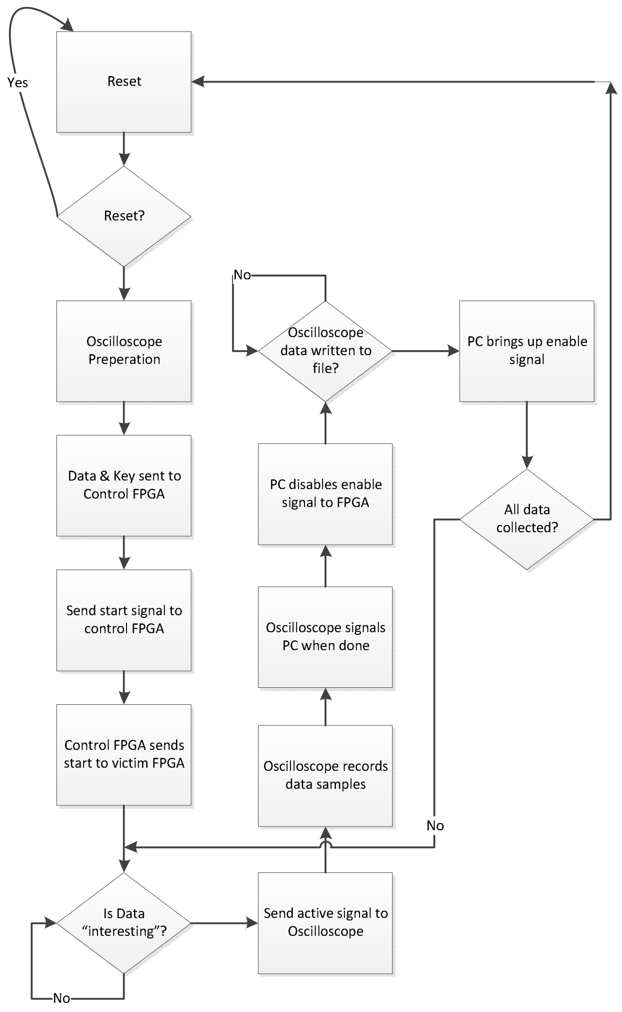
Our first step in designing the basic setup was to determine what components we will use. We decided to use the Nexys2 board as the controller because open source, Linux compatible drivers already exist, which will help us satisfy the requirement that the system be able to work with a Linux PC. The Spartan 3e Starter Board was selected for the victim due to its availability in the cryptography lab, thereby reducing overall cost.

White Box



Our second preliminary design focused on identifying necessary interfaces between devices at run time. A USB interface will need to be setup to pass data to the controller FPGA. The controller and victim will communicate via their respective Hirose FX-2 connectors. The trigger signal sent to the oscilloscope via a pin in one of the controller board's serial interface ports. The oscilloscope will send data to the PC via a USB connection. The PC software will interface with preexisting attack scripts. For ease of use, we have elected to use a configuration file that will dictate how the attack will be run.

Flowchart


Yes
Reset
Reset?
Oscilloscope Preperation
Data & Key sent to Control FPGA
Send start signal to control FPGA
Control FPGA sends start to victim FPGA
Is Data "interesting"?
No
Send active signal to Oscilloscope
Oscilloscope records data samples
Oscilloscope signals PC when done
PC disables enable signal to FPGA
Oscilloscope data written to file?
No
PC brings up enable signal
All data collected?
No

The flow chart above diagrams our current protocol for the operating the testbed. After all elements are placed in reset, data and control information is passed to the controller from the PC. The data is passed to the victim, which encrypts the data according to control signals received from the controller. At the appropriate time, the controller set the trigger signal, causing the oscilloscope to record power traces until the trigger signal falls. In the event that the oscilloscope's memory is exhausted, execution of the rest of the testbed is suspended while the power readings transfer from the oscilloscope's internal memory to a data file on the PC. When the full attack has completed, back-end analysis scripts are invoked to process the data.

Appendix B Design Document

ECE-492 Design Review

Differential Power Analysis Testbed

Differential power analysis is a type of side channel attack used to compromise a cryptographically secure system by obtaining the secret key the device uses to encrypt data. This method is proven effective on many different algorithms, including AES-128. Of the numerous different platforms capable of running such cryptographical algorithms, one of the most notable types is the field programmable gate array, or FPGA. FPGAs are used widely in industry as they allow one to inexpensively create different circuits Currently, in order to conduct differential power analysis on an FPGA based system, a singular unit is used which requires a user to take numerous measurements with a digital oscilloscope. This method is tedious, repetitive and inefficient. The purpose of this project is to build an automated controller circuit to conduct these numerous tests based upon a set of user defined parameters. The control system will be run on a computer which will have a command line interface with which to control the FPGA system.

December 3rd, 2010

Dr. Kaps

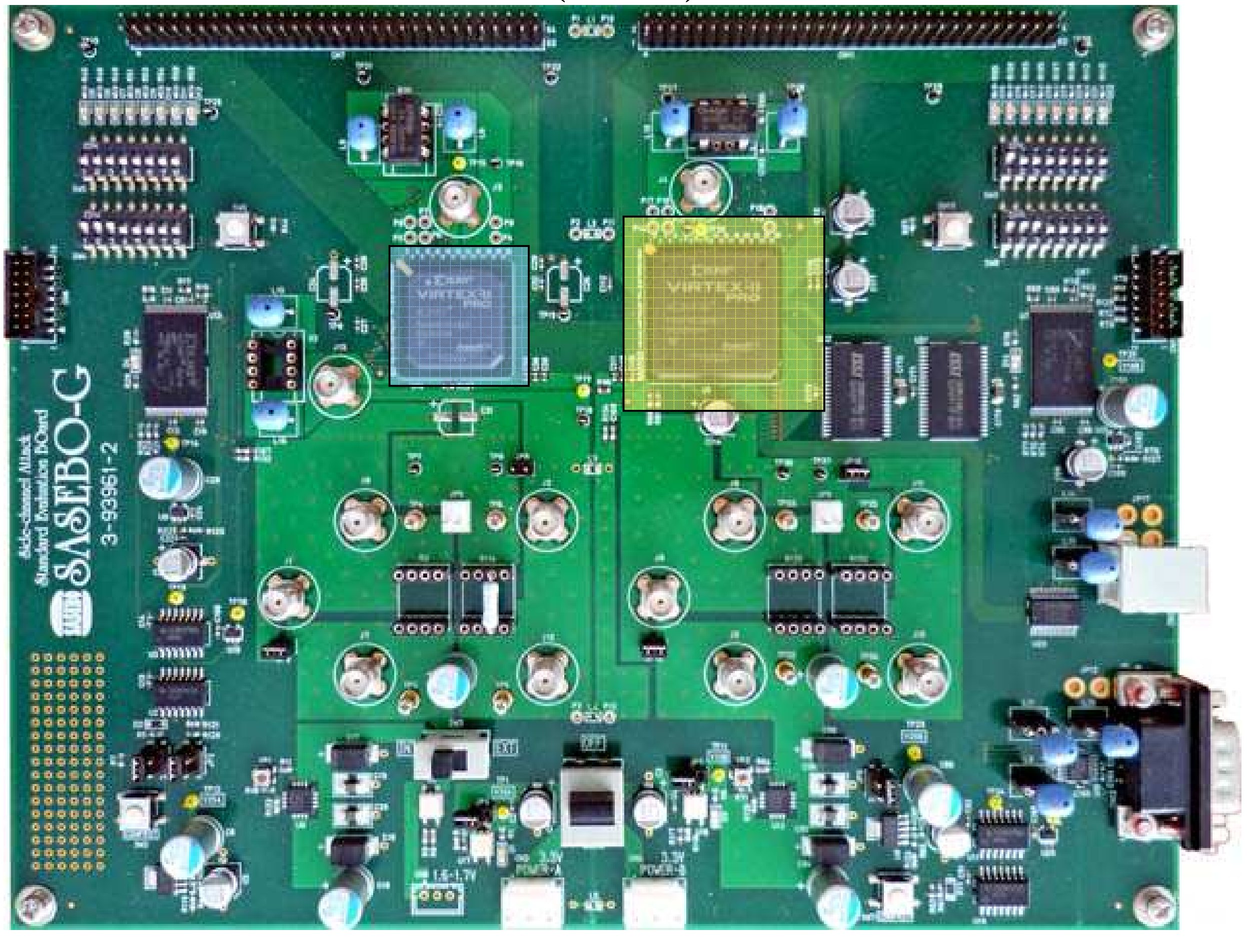
Patrick Adams
Lindsay Walton

Introduction

With increasingly sensitive information being stored electronically, more focus has been placed on keeping data secure. To that end, encryption algorithms have become increasingly complex, rendering attacks difficult or, in some cases impossible. However less has been done to improve the platforms on which these algorithms are implemented. Common platforms such as Field Programmable Gate Arrays (FPGAs) have been found to be susceptible to power analysis attacks.

Differential power analysis (DPA) attacks are particularly effective against FPGA based cryptographic implementations. By examining the power consumption of a device implementing an encryption system, the attacker is able to use this side-channel attack to “look inside” the device without having to physically examine the hardware. This ability provides the attacker with newer, more subtle exploits that often enable him or her to comprise seemingly impenetrable cryptographic systems. This poses the cryptographic community with the challenge of developing implementations less susceptible to DPA attacks.

In order to harden FPGAs against DPA attacks, one must first have a manner by which to test a chip's susceptibility to the attack. The best platform currently available for such testing is the Side-channel Attack Standard Evaluation Board (SASEBO).



As shown above, the SASEBO features two FPGAs, one to act as a “controller”

(highlighted in blue) which will drive the attack, and one to act as the “victim” (shown in yellow). This setup makes an excellent platform for testing side-channel attacks, except in the case of DPA.
In any DPA attack the physical characteristics of the victim plays a large role in effectiveness of the attack. As such, it is necessary for the developer to be able to run an attack on numerous different victim FPGAs, a feature that is not possible with SASEBO's victim-on-board setup. These, along with other issues to be discussed later, preclude SASEBO from being used as a viable testbed. Thus it is apparent that a new testing platform is needed in order to aid development of cryptographic implementations that resist DPA attacks. To that end, it is our intent to expand upon the SASEBO design, and develop a new testbed capable of evaluating an FPGA's vulnerability to DPA attacks.

Statement of Need

Basic Attack

For this project, we are to build an attack board that conducts differential power analysis (DPA) with the goal of attacking an FPGA running the Advanced Encryption Standard (AES-128) with a key size of 128 bits. The method as described by William Hnath and Jordan Pettengill in their paper: *Differential Power Analysis: Side Channel Attacks in Cryptography*, focuses on obtaining the cipher key used in the encryption process by using statistical analysis and correlation on power traces recorded while the victim board conducts the encryption. The attack that we will be implementing will focus on the final step of the AES encryption algorithm which makes the computation for the attack significantly simpler as a longer component of AES is not implemented in the final phase. In AES, data is operated on one byte array[5] of 16 bytes at a time with a key array of equal size. The key that is used is generated by AES is based on the key schedule and each round of encryption includes part of the original key. Recording the final round for each subsequent encryption allows an attacker to get one byte of this key at a time instead of attacking the whole key at once. Once power consumption of the cryptographic chip is recorded by obtaining several thousand[6] traces, statistical correlation can be performed on the data. Using previous power measurements taken on operations with known keys, an attacker is able to make an accurate guess as to what the key is.

Problem Statement

While this method of attack is generally effective, it has several notable flaws. The first issue of this approach is that currently available attack platforms require manual operation. A user must manually obtain more than 10,000 power traces to obtain only 8 bits of the encryption key. For AES-128, this means the user must obtain a minimum of 4,096 separate power traces in order to obtain a full key. Hnath and Pettengill have been shown that an average of 10,000 traces is

[5] 4x4 array
[6] On average, more than 10,000 traces are needed to accurately get the key

necessary to produce an accurate key. The user's task is further complicated by limitations of the digital oscilloscope used to obtain these power traces. Generally, the oscilloscope is incapable of storing the needed 4096 traces for a key. This forces the user to periodically dump the memory of the oscilloscope to the PC during the attack. Due to the extreme number of operations required to definitively obtain a key via this method, it is apparent that automation is necessary. Development of an automated testbed would allow one to attack a desired device quickly and efficiently.

In order to efficiently develop such a testbed, the use of open source tools has been chosen to expedite development time. Several aspects of the setup, as will be discussed later, would require communication between different elements. While proprietary drivers such as Digilent's Adept program are available, most are only compatible with a Windows operating system, and can be limited in functionality. Open source drivers, while often intended for use with Linux operating systems, could potentially be modified in order to remove any platform dependence. This would not only allow the testbed to operate with numerous operating systems, it would also allow one to efficiently design a testbed by allowing one to optimize the drivers to work with their equipment, while reducing overall costs.

With these considerations in mind, it is apparent that a platform independent, automated testbed would enable more efficient DPA attacks on all encryption algorithms, if they are susceptible to DPA. Our initial focus will be on the AES encryption algorithm. This increased efficiency will lead to a better understanding of vulnerabilities in the implementation of a given algorithm which will help produce more secure implementation techniques for cryptographic modules in the future.

Approach and Requirements

Oscilloscope Control

Many thousands of measurements are required to successfully conduct a differential power analysis attack. Current oscilloscope memory limitations require a human operator to download the data once the memory is full and then restart the attack with a new set of data points. An ideal system would be able to pause while the data is uploaded to a data file on the computer. A requirement for our system is that the controller board be able to pause the cryptological processing unit while the data from the oscilloscope is saved on the computer. Additionally, most DPA attacks focus on a particular step in the encryption process, in our case, the substitution round of the final phase. As such, one does not require power measurements for the entire operation of the encryption system, just traces for the operations of interest. These areas of interest are not necessarily periodic and only occur for small periods of time. To increase efficiency, our system will have to detect when such an area of interest is occurring, at which point it would enable the oscilloscope's recording mechanism. To control the oscilloscope, the control FPGA board we are designing must then have an interface connected to the oscilloscope which allows for more precise measurement and removal of erroneous data.

FPGA/PC Interface

Today's standard for peripheral communication is the Universal Serial Bus standard or USB. This connection has a variety of uses, for an FPGA data transmission and the ability to program the FPGA as well as the system FLASH are the primary uses. A derived requirement is that the control FPGA board must have a USB interface, which will support programming as well as supporting data transmission. While USB is universal, the driver software to program and communicate with the FPGA board is not. Many systems today use open source software which may not have an official supported interface driver. In order to be applicable to as many current researchers as possible our system must support the USB interface through Linux as well as Windows in addition to other open source development tools.

Control FPGA/Victim FPGA Interface

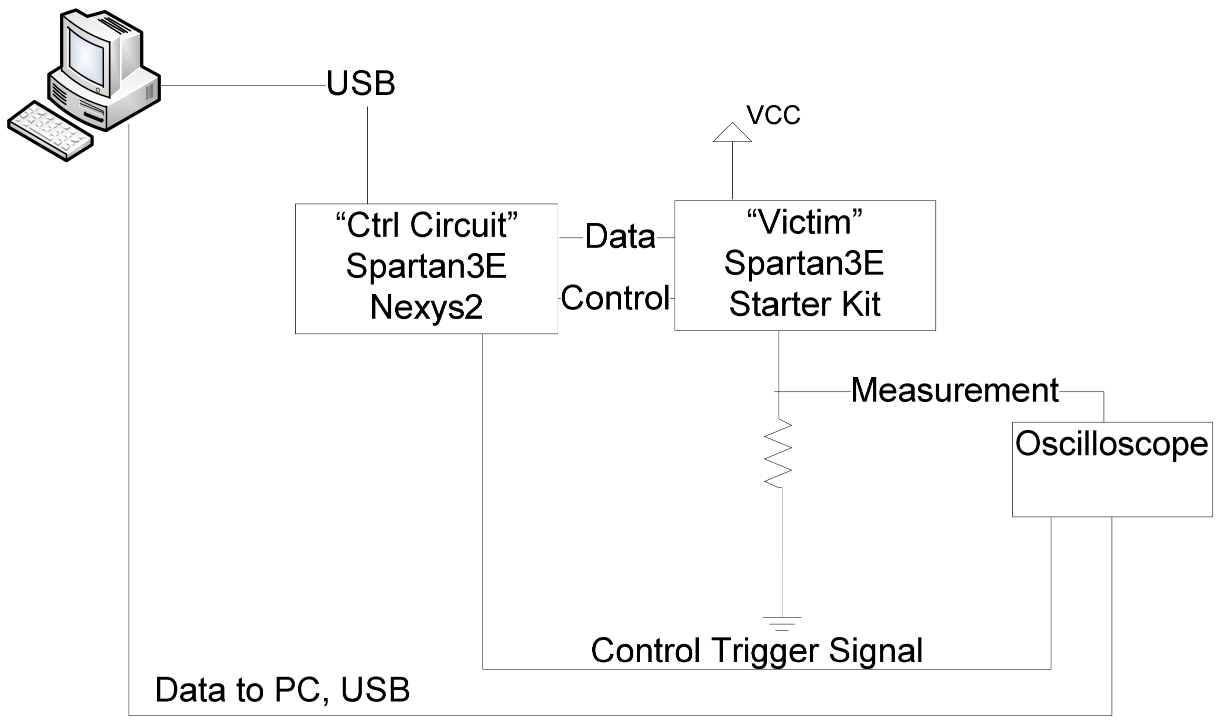
As part of being able to perform a DPA attack, our system must interface with many different cryptologic boards and must have a bridge which has hirose connections as well as optional pin out connections to allow for the optional cases that the victim board does not have a hirose connection to interface with our controller board. Our specific test system will interface with a Spartan 3E Starter Kit as the victim board.

Command Line Program

In addition to supporting the controller board on Linux, a control program must be supplied that works on all operating systems. This program must read configuration data about the test from a file, and if no file is provided, conduct the attack based on a specified set of default values. The program must also be able to automatically download the data from the oscilloscope while pausing the operation of the measurement circuit.

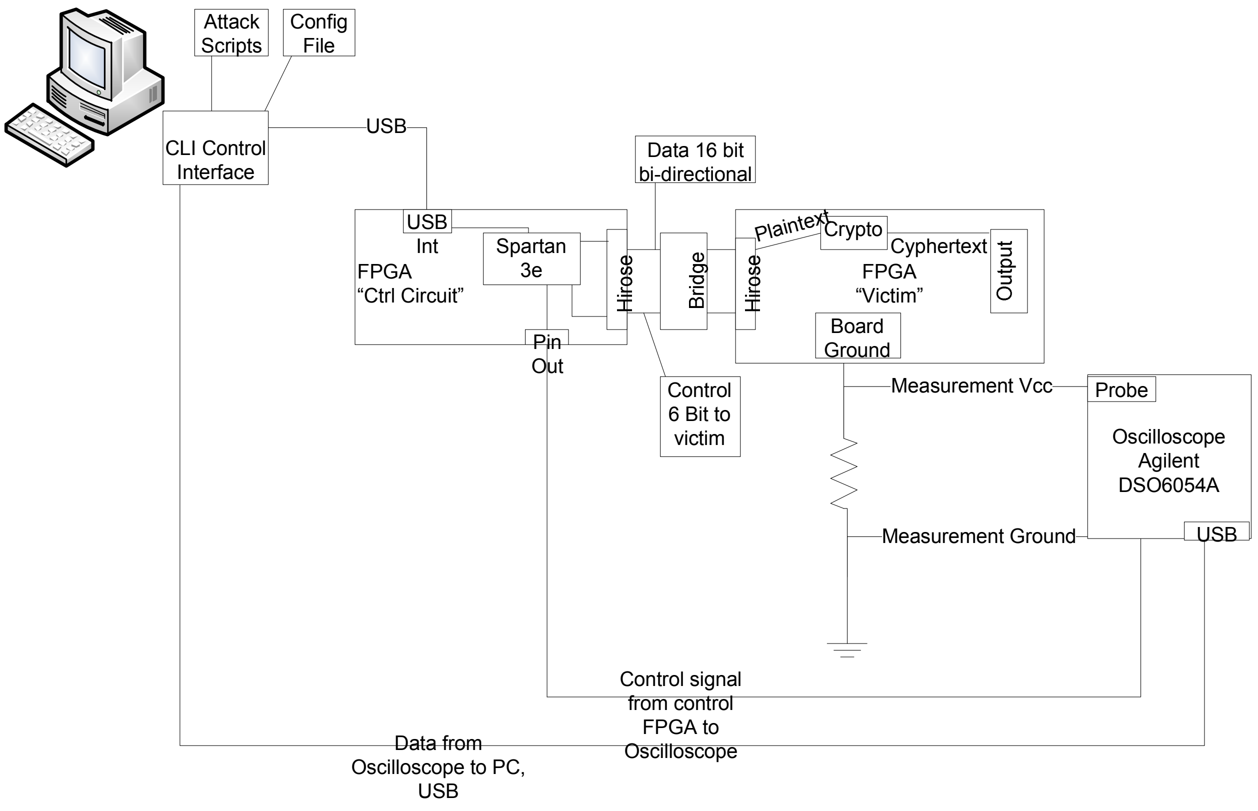
Preliminary Design

Black Box



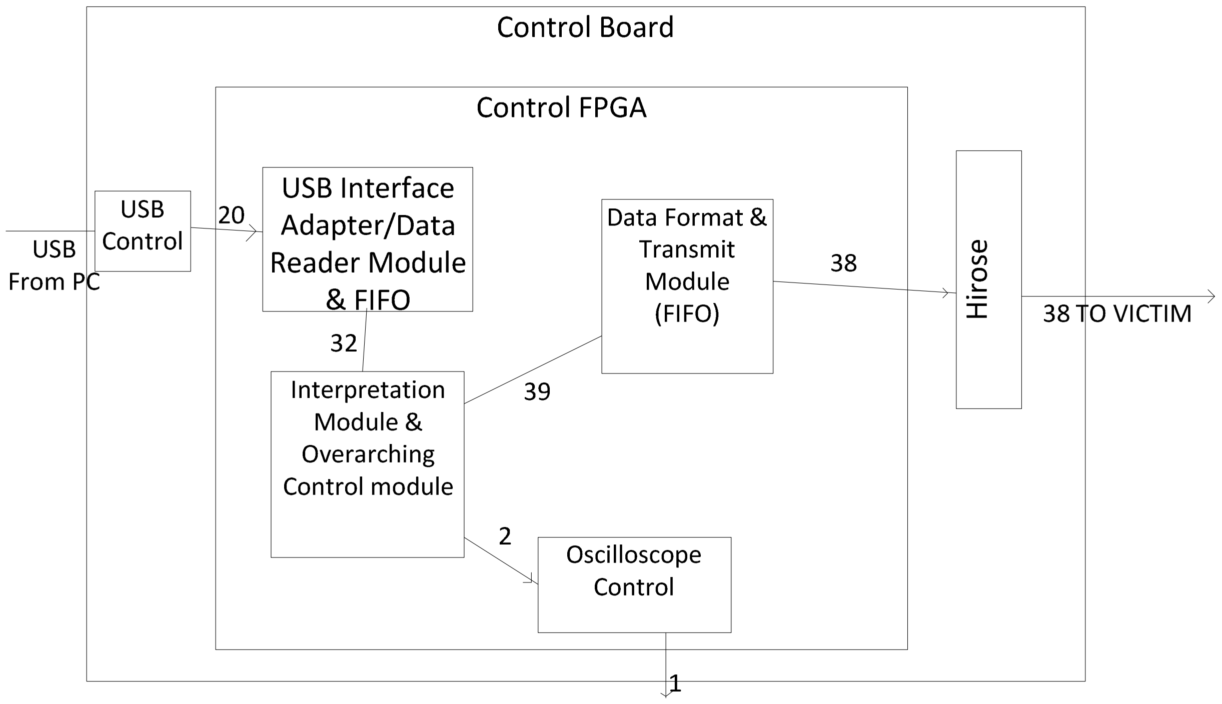
Our first step in designing the basic setup was to determine what components we will use. We decided to use the Nexys2 board as the controller because open source, Linux compatible drivers already exist, which will help us satisfy the requirement that the system be able to work with a Linux PC. The Spartan 3e Starter Board was selected for the victim due to its availability in the cryptography lab, thereby reducing overall cost.

White Box

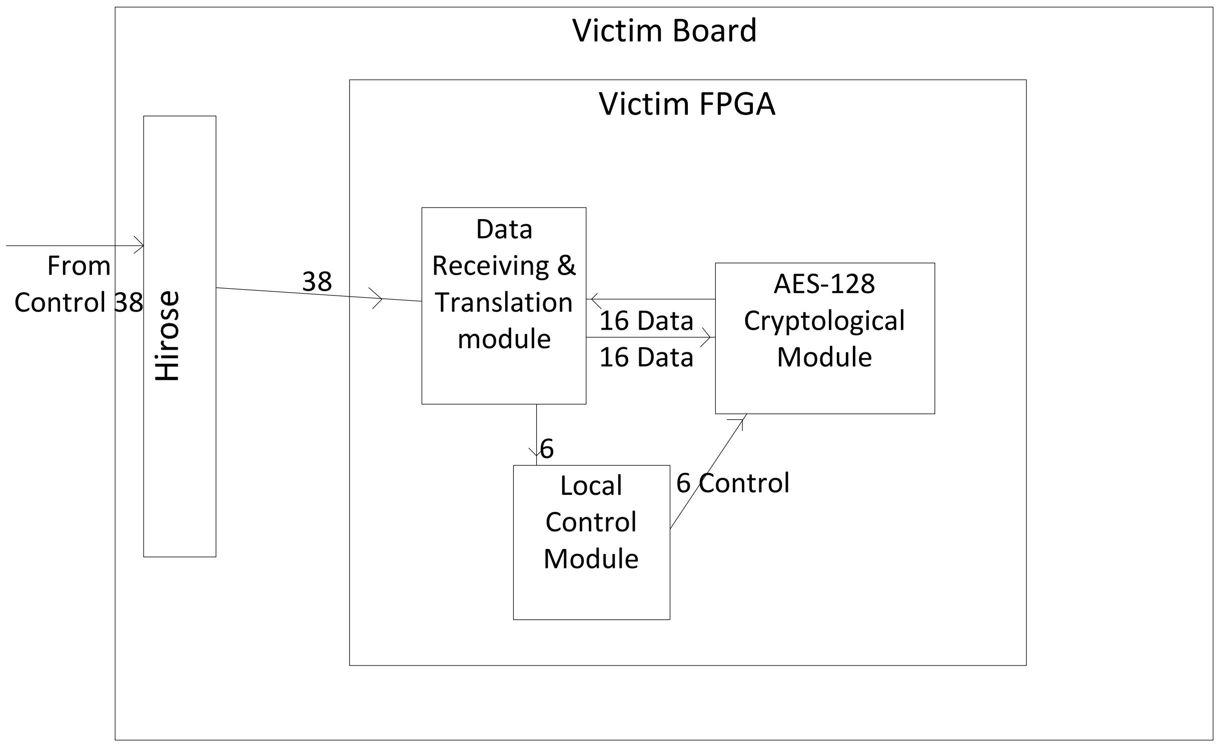


Our second preliminary design focused on identifying necessary interfaces between devices at run time. A USB interface will need to be setup to pass data to the controller FPGA. The controller and victim will communicate via their respective Hirose FX-2 connectors. The trigger signal sent to the oscilloscope via a pin in one of the controller board's serial interface ports. The oscilloscope will send data to the PC via a USB connection. The PC software will interface with preexisting attack scripts. For ease of use, we have elected to use a configuration file that will dictate how the attack will be run.

Unit Designs

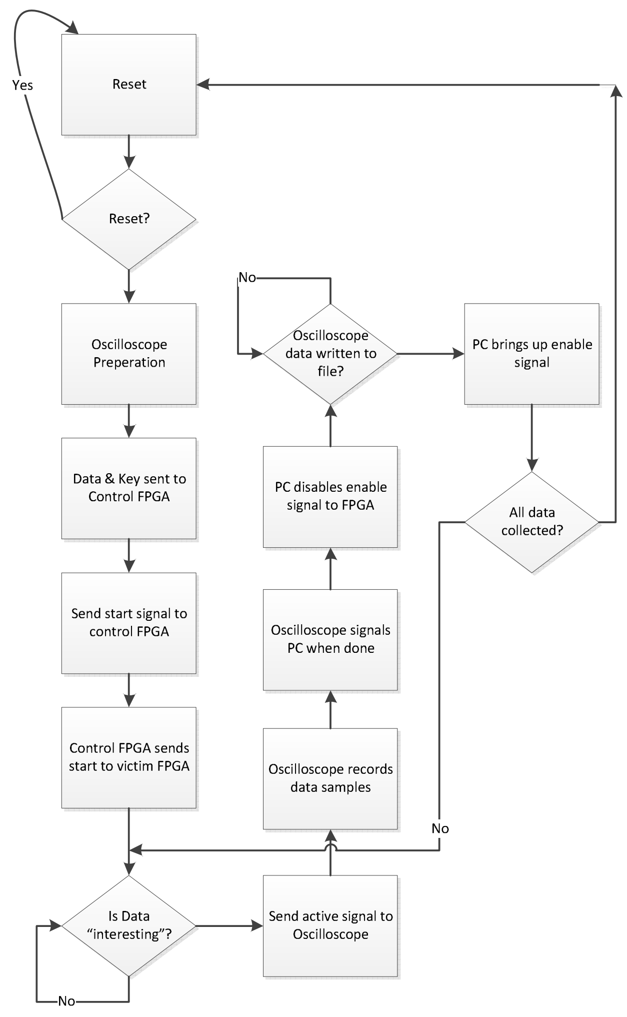

Control Board
Control FPGA
USB From PC
USB Control
20
USB Interface Adapter/Data Reader Module & FIFO
32
Interpretation Module & Overarching Control module
39
Data Format & Transmit Module (FIFO)
38
Hirose
38 TO VICTIM
2
Oscilloscope Control
1

After designing the overall layout of the test bed, the individual FPGA's were considered. To meet our requirements, the controller will need to control the oscilloscope, dictating when the scope will record data, and when it will send its readings to the computer. The controller will also drive the victim, by sending it control data and cipher text. To simplify communications we decided to pursue a master/slave communications protocol, in which the controller would send its system clock across the bridge module to drive the victim. Through examination of established methods of sending data from a computer to an FPGA via USB, it became apparent that an adapter module would be needed to convert the raw USB signals into data usable by the controller. All of these modules would be controlled by an overarching control module.



The victim FPGA will be far simpler, as it will only require a communications and control wrapper for the encryption module. Both of these wrapper modules will be driven by the control board via the bridge device.

Flowchart


Reset
Yes
Reset?
Oscilloscope Preperation
Data & Key sent to Control FPGA
Send start signal to control FPGA
Control FPGA sends start to victim FPGA
Is Data "interesting"?
No
Send active signal to Oscilloscope
Oscilloscope records data samples
Oscilloscope signals PC when done
PC disables enable signal to FPGA
Oscilloscope data written to file?
No
PC brings up enable signal
All data collected?
No

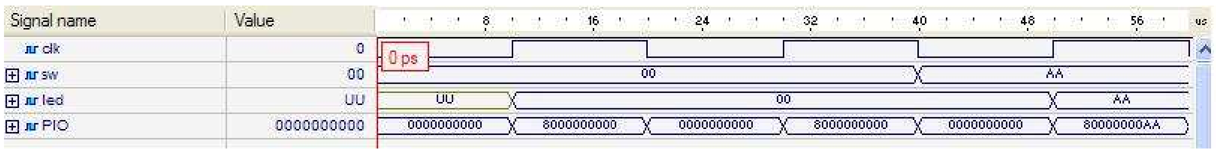
The flow chart above diagrams our current protocol for the operating the testbed. After all elements are placed in reset, data and control information is passed to the controller from the PC. The data is passed to the victim, which encrypts the data according to control signals received from the controller. At the appropriate time, the controller set the trigger signal, causing the oscilloscope to record power traces until the trigger signal falls. In the event that the oscilloscope's memory is exhausted, execution of the rest of the testbed is suspended while the power readings transfer from the oscilloscope's internal memory to a data file on the PC. When the full attack has completed, back-end analysis scripts are invoked to process the data.

Prototyping

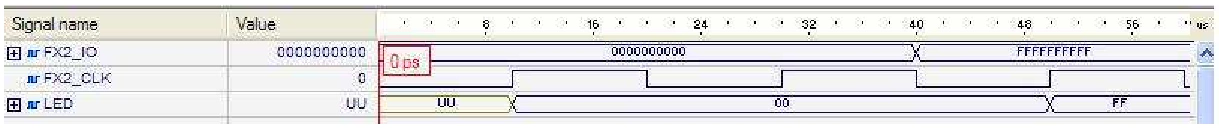
For our differential power analysis testbed we felt that we needed a tested and robust controller platform as the centerpiece of our system. To this end, we researched several different boards and found that Digilent’s Nexys2 board met our specifications, and offered a suitable educational discount price. This board meets the requirement of the specialized 100 pin hirose connection, as well as having the ability to run off of an external power source, separate from the USB interface. Communication with the device is supported over the USB channel which was also required. The onboard FPGA chip is a Xilinx Spartan 3E with 500 gates which is identical to the FPGA on the victim board.
We have also created as a prototype bridge connection using wire wraps around two male hirose connectors. We only wrapped the pins that we needed, and have tested connectivity between them. As part of our prototyping efforts, we created a test circuit which sent 8 bits, half of the end goal of 16 bits of data across the hirose connection. This data was simply passed around on the victim FPGA and no change was made. Next steps include modifying this circuit to make a change to the data and sending it back to the control FPGA and ensuring that the data is correct. Once this is verified, a simple change taking the data bus width from 8 to 16 bits can be made which would allow for the full testbed functionality. Along the way we are prototyping USB communication with the control Nexys2 board. We currently can send data and receive data from the PC. The FIFOs have not yet been implemented so they all operate on one global clock, driven by the USB. Once FIFOs are in place and the Spartan3 prototyping board is fully programmable, the communication wrappers will be complete.

Current Simulation Results

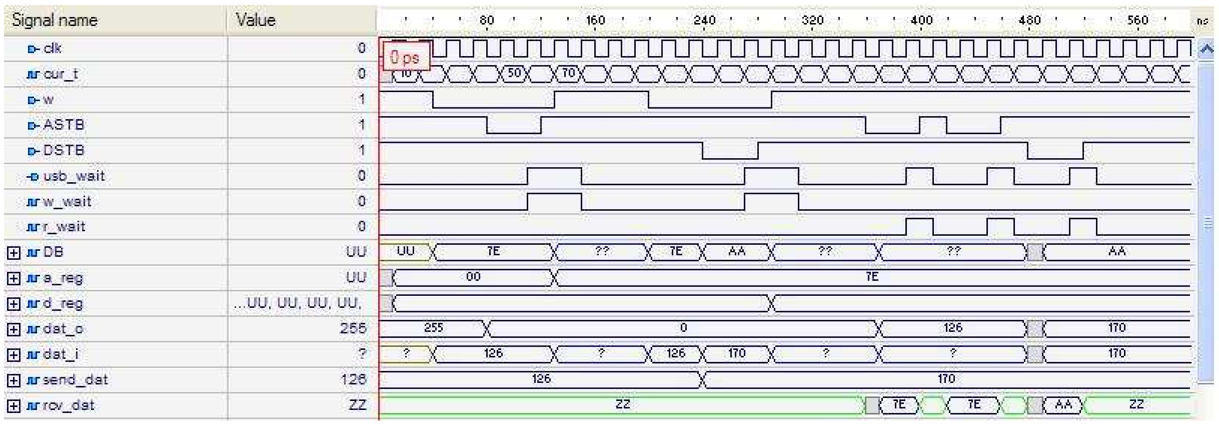
Transmitter



Receiver



USB



Signal name	Value
clk	0
cur_t	0
w	1
ASTB	1
DSTB	1
usb_wait	0
w_wait	0
r_wait	0
DB	UU
a_reg	UU
d_reg	...UU, UU, UU, UU,
dat_o	255
dat_i	?
send_dat	126
rcv_dat	ZZ