A Reconfigurable Arduino Crypto FPGA Shield

Ryan Bornhorst
Gomathy Venkata Krishnan
Meiqi Zhao
Dustin Schnelle

Version 1.2 1/24/18

Table of Contents

Purpose of Product Design Specification

This product design specification fully defines the high-assurance security provided by the reconfigurable FPGA crypto shield designed to be compatible with the Arduino. The document is a detail reference guide for the team members, faculty advisor, and project sponsor associated with the Capstone at Portland State University.

Background

Inexpensive open hardware platforms, such as the popular Arduino, have had a huge impact on the embedded systems industry. Especially in the world of the Internet-of-Things (IoT) [4]. Arduino's open-source microcontroller platform has become popular to both hobbyists and educators because of its easy-to-use hardware and software interface. The low cost of the Arduino platform also helps. As Arduino has become more popular in academics, the amount of documentation for the Arduino platform has greatly increased. There are already two course at the University of California that teach Arduino [1], its programming environment, and interfacing principles for an IoT specialization.

With the massive growth of the IoT the world is changing, allowing for the innovation of new designs and embedded devices within the home. This growth of devices depending on a connection to the internet means that we need to focus on the security as a first class component when communicating sensitive data. It's becoming increasingly necessary to ensure the security of embedded systems connected to the network, so adding cryptography to the functionality is important.

One solution to the problem is using an FPGA shield to perform the encryption and add to the functionality of the Arduino. The current Arduino FPGA shields on the market however don't guarantee high-assurance of the software, firmware, and hardware. High-assurance is needed for the embedded system to provide objective evidence of the system's correctness and security. Usually this evidence comes from a hand-written test bench, but we are looking to obtain this evidence from formal assurance or what is known as formal methods.

<Talk about the market>

Problem Statement

In current internet market, the embedded system architects rarely pay attention to security as a first class component, which could cause many vitally important information leaked. The sponsor's primary goal of the project is to develop a high-assurance crypto FPGA shield for Arduino. This vision system should be able to guarantee performancing high-assurance crypto on inexpensive Arduino devices.

[1] http://online-journals.org/index.php/i-jep/article/view/6845/4454

Behavioral Requirement

- The system must be affordable and be built for under $100.
- The system's API must interface the FPGA via microcontroller.
- The system must exhibit high assurance to provide system correctness.
- The system must be high performance by being 10x faster than OpenSSL current benchmarks.

Non-Behavioral Requirement

- The system should use model checking as a formal assurance technique.
- The system should use implementation tools to synthesis, validate, and verify cryptographic systems.
- The system should be built on an Arduino R3 format compatible shield.
- The system should be built from scratch.

Objective

- Gain expertise in Arduino programming.
- Gain expertise in FPGA programming especially for a specific FPGA shield.
- Design a custom Arduino FPGA board.
- Learn to use cryptographic libraries to implement high-assurance algorithms.
- Learn how to create APIs.
- The Arduino crypto FPGA shield should be dynamically reconfigurable.
- Learn to use Cryptol, SAW, and NaCl libraries.
- Learn how to create a PCB and interface the Arduino board with an FPGA embedded shield.

Deliverables

Delivered by Capstone Deadline

- *Proposal*: Document describing the background, motivation, and initial design requirements for implementing a Crypto FPGA Shield for the Arduino.
- *Test Plan*: This document will be used to verify that all of our system requirements can be met by identifying required testing instrumentation as well as test results.
- *Code Repository and Wiki*: All project documentation including schematics, source code, and general information will be located here.

- *Crypto FPGA Shield*: System Verilog code will be generated from Cryptol software provided to us from Galois. That code will then be uploaded to an FPGA shield. An Arduino will then be able to interact with the FPGA shield through an API.
- *Wrapper Cryptographic Software Library*: A variant of an AES library, such as Sodium, will be used within the Arduino development environment to configure the FPGA shield.
- *Demonstration*: Benchmarks and test applications will be used to demonstrate the use of the API as well as validate our requirements for "formal assurance."

Time Permitting or Future Deliverables

- *Stretch Goal 1*: Our team would like to contribute our software library to the Arduino website and gain visibility to it by publishing periodic updates and announcements on related forums.
- *Stretch Goal 2*: Design a daughterboard containing an FPGA shield from scratch that can interface with an Arduino, use our software library, and meet all of the design requirements within this document.

Marketing Requirements

- The shield should design should be compact.
- The system should be plug and play.
- The system should cost less than $100 while maintaining high functionality.
- There will be an online forum that supports the consumers and gives them constant updates about the device and features.
- The system will cater to the requirements of the cybersecurity market and will have features that will be constantly updated.
- Future software updates will remain compatible with the hardware and the firmware used on the system.
- The system will be durable within its intended operating environment.

Engineering Requirements

Performance

- The system should guarantee "formal assurance."

Functionality

- The Arduino environment and FPGA will be able to interface with each other.
- The system should have a reset (software or hardware) incase it hangs.
- The system API will be able to use at least 4 crypto algorithms.
- The FPGA should be chosen in a way that it will be dynamically reconfigurable.

Economic

- The total cost and manufacturing cost should not exceed $100.

Energy

- The system will operate for at least 5V or 3.3V logic.
- The system will function using the embedded system power input.
- The system will operate indefinitely while plugged in.

Legal

- The system will be 100% open source.

Maintainability

- Future software updates will remain compatible with the hardware and the firmware used on the system.

Manufacturability

- The custom board should be able to easily interface with the arduino headers.
- The custom board should be manufactured with readily available parts.

Reliability and Availability

- The product will have frequent software updates to ensure that it works as expected.
- The product will be operational 100% of the time.
- Future hardware or firmware updates will be communicated to the consumers months in advance.

Social and Culture

- The development team should contribute a library on the Arduino website.
- The team will publish announcements on related blogs and forums to gain visibility for the software library.

Usability

- Users of the system should be able to learn 80% of its functionality within 2 hours.
- The system API will be well documented.
- The API will be user friendly and can be subject to refactoring to enhance functionality at any time.
- Documentation for the hardware, software, and firmware will be provided.

System Component Block Diagram

The AtMega microcontroller will be powered from a PC source using a USB connector. This system will use the Arduino IDE to configure the AtMega MCU on the board. For the initial project design, a development board containing both the MCU and the FPGA will be used. The boards used will be the Mojo v3 and the Papilio One 250k. Both of these boards contain a variation of the AtMega controller as well as the Spartan 6 and Spartan 3E FPGAs respectively.

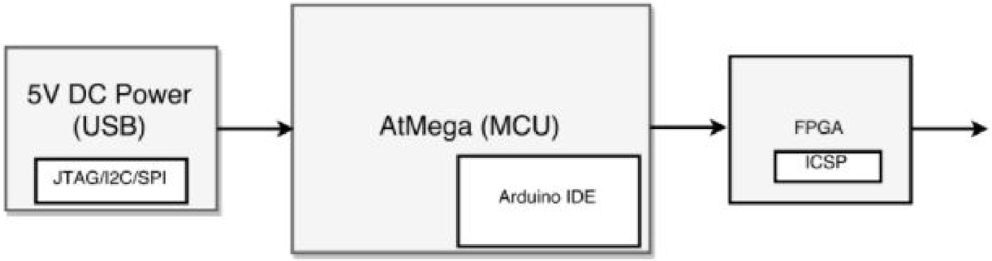

Figure 1. System Component Block Diagram

Software Component Block Diagram

AES Wrapper Library - Wrapper libraries take in a current software library's interface and turns it into a more compatible one. This is usually done to make the original library less complicated or to make the data between the library and the interface you are calling it from more compatible. The wrapper library that we are implementing will use the pre-existing cryptography library Sodium and make it compatible with the Arduino interface as well as the synthesis and formal assurance tools we will be using.

Cryptol - Cryptol's synthesis tools perform extensive and often very complicated transformations to turn Cryptol programs into hardware primitives available on target FPGA platforms. The formal verification framework of Cryptol allows equivalence checking between Cryptol and netlist representations that are generated by various parts of the compiler [2].

SAW - SAW is another tool designed by Galois that is used to formally verify properties of code written in C, Java, or Cryptol. SAW is closely connected with Cryptol and is commonly used to prove equivalence between a Cryptol specification of an algorithm and one that was written in C or Java [3].

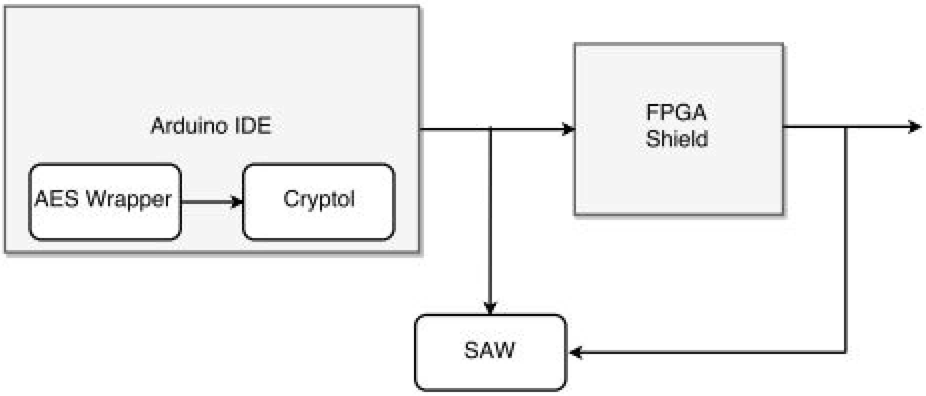

Figure 2. Software Component Block Diagram

Testing/Debugging

This part of the document is expected to change dynamically during testing as constraints and new measures to check for assurance are identified.

Test Writer:						
Test Name:	Software Compatibility Test					
Description:	This test will be used to test if the Cryptol and SAW software is able to compile and function properly. (generating expected outputs)					
Tester Information:						
Name:			Date:		Time:	
Setup:	Load Cryptol and SAW software onto the development boards.					
Test	Program Compiles	Test Hash	Test Key Encryption	Test Stream Cipher		
1 (Cryptol: Papilio One)						
2 (Cryptol: Mojo v3)						
3 (SAW: Papilio One)						
4 (SAW: Mojo v3)						

Test Writer:						
Test Name:	AES Library Functionality Test					
Description:	This test will be used to test if the Sodium library is able to compile as well as check if the benchmark tests produce valid results. (expected function output)					
Tester Information:						
Name:			Date:		Time:	
Setup:	Load Sodium library onto the development boards and test benchmark files if they exist.					
Test	Program Compiles	Test Hash Benchmark	Test Key Encryption Benchmark	Test Stream Cipher Benchmark		
1 (Papilio One)						
2 (Mojo v3)						

Test Writer:						
Test Name:	Model Checker					
Description:	This test will be used to test all possible inputs for our crypto algorithms as well as test that the program does not stall during testing. (exceed 30 min program time out)					
Tester Information:						
Name:			Date:		Time:	
Setup:	Use SAW to exhaustively test all inputs into our generated crypto functions.					
Test	Program Compiles	Model Check Hash	Model Check Key Encryption	Model Check Stream Cipher		
1 (Papilio One)						
2 (Mojo v3)						

Test Writer:						
Test Name:	Benchmark Comparison Test					
Description:	This test will be used to test speed benchmarks for x amount of inputs using original AES library and comparing it to the Cryptol generated library after loading it onto the FPGA. Test will determine % speed increase using an FPGA shield loaded with Cryptol generated HDL.					
Tester Information:						
Name:			Date:		Time:	
Setup:	Load Cryptol benchmark files and test them against the original Sodium library benchmarks.					
Test (Mojo v3 and Papilio One)	Program Compiles	Test Hash Benchmark	Test Key Encryption Benchmark	Test Stream Cipher Benchmark		
1 (10%)						
2 (20%)						
3 (30%)						
4 (40%)						
5 (50%)						
6 (60%)						
7 (70%)						
8 (80%)						
9 (90%)						
10 (100%)						

Licensing Information

Proposal Approval

The undersigned acknowledge they have read/examined the project proposal document and agree with the material contained within. Any changes to this document require the discussion and approval of the undersigned.

Dustin Schnelle - Team Member
Date

Gomathy Ventaka Krishnan - Team Member
Date

Ryan Bornhorst - Team Member
Date

Meiqi Zhao - Team Member
Date

Dr. Christof Teuscher - Faculty Advisor
Date

Joe - Project Sponsor
Date

Dan - Project Sponsor
Date

Revision History

Version Number	Implemented By	Reversion Date	Reason
1.0	Team	1/29/2018	Initial PDS didn't have enough information
1.1	Team		Revised per Advisor and sponsors comments

Appendix A - Bill of Materials

Sno	**Product**	**Quantity**	**Cost/Piece**	**Total Cost**
1	Papilio One 250k	2	$37.99	$75.98
2	Mojo v3	2	$74.99	$149.98
3	PulseRain M10	2	$85.00	$170.00
4	Spartan 3E	1	$13.86	$13.86
5	Spartan 6	1	$23.73	$23.73
6	Intel MAX10	1	$7.10	$7.10
6	Arduino Mega	2	$45.95	$91.90
7	PCB	2	$10	$20.00
8	Soldering tools: Liquid solder, solder wick, reflux	1	$13	$13.00
9	M/M, F/M, F/F header pins or wires	1	$8	$8.00
10	Heat sink	2	$3	$6.00
Total:				$579.55

Appendix B - Gantt Chart

Appendix C - Mnemonics

Appendix D - Markdown for Hyperlinks(Oxygen)

SAW: https://galois.com/project/software-analysis-workbench/

Appendix E - Code References

[1] A. Deshpande, M. Deshpande and D. Kayatanavar, "FPGA implementation of AES encryption and decryption", IEEE, 2009. Available: http://ieeexplore.ieee.org/document/5204421/
[2] A. Gielata, P. Russek and K. Wiatr, "AES hardware implementation in FPGA for algorithm acceleration purpose", 2008 International Conference on Signals and Electronic Systems, 2008. Available: http://ieeexplore.ieee.org/document/4673377/
[3] S. Varhade and N. Kasat, "Implementation of AES Algorithm Using FPGA & Its Performance Analysis", International Journal of Science and Research, vol. 4, no. 5, pp. 2484-2492, 2015. Available: https://pdfs.semanticscholar.org/3dcd/de6d1e3329d69083c10714866023da2ae068.pdf
[4] A. Borkar, R. Kshirsagar and M. Vyawahare, "FPGA implementation of AES algorithm", 2011 3rd International Conference on Electronics Computer Technology, 2011. Available: http://ieeexplore.ieee.org/document/5941780/.
[5] W. Fischer and N. Homma, "Cryptographic hardware and embedded systems" -- CHES 2017, 1st ed. Springer International Publishing, 2017. Available: http://www.springer.com/us/book/9783319667867
[6] Wang W., Szefer J., Niederhagen R. (2017) FPGA-based Key Generator for the Niederreiter Cryptosystem Using Binary Goppa Codes. In: Fischer W., Homma N. (eds) Cryptographic Hardware and Embedded Systems – CHES 2017. CHES 2017. Available: https://eprint.iacr.org/2017/595.pdf
[7] Joe Kiniry, "Galois Ultra Low Power High Assurance Asynchronous Crypto", Galois, 2016. Available: https://www.nist.gov/sites/default/files/documents/2016/10/19/kiniry-presentation-lwc2016.pdf
[8] Bernstein D.J., Chou T., Schwabe P. (2013) McBits: Fast Constant-Time Code-Based Cryptography. In: Bertoni G., Coron JS. (eds) Cryptographic Hardware and Embedded Systems - CHES 2013. CHES 2013. Available: https://link.springer.com/chapter/10.1007/978-3-642-40349-1_15
[9] Cherkaoui A., Fischer V., Fesquet L., Aubert A. (2013) A Very High Speed True Random Number Generator with Entropy Assessment. In: Bertoni G., Coron JS. (eds) Cryptographic Hardware and Embedded Systems - CHES 2013. CHES 2013. Available: https://link.springer.com/chapter/10.1007/978-3-642-40349-1_11

[10] Hardin D., Hiratzka T.D., Johnson D.R., Wagner L., Whalen M. (2009) Development of Security Software: A High Assurance Methodology. In: Breitman K., Cavalcanti A. (eds) Formal Methods and Software Engineering. ICFEM 2009. Available: https://link.springer.com/chapter/10.1007/978-3-642-10373-5_14

Appendix F - Competitor Research

Sno	Product	Attributes	Cost	Availability
1	Sparkfun CryptoShield (Daughter Board)	● Encrypted and Decrypted Hardware installed ● Encrypted EEPROM ● RSA encryption/decryption ● SHA-256 ● Uses elliptic Digital Signature Algorithm	Retired Product	Retired Product
2	Arduino Compatible FPGA shield	● Spartan 6 FPGA shield includes SPI Configuration Flash, programmable from Arduino	Hackaday Project	Can be manufactured with the .sch, .brd and .gbr files
3	PicoEVB (Embedded Soc)	● Xilinx Artix XC7A50T ● PCIe Gen 2 ● PCIe Design prototyping(plug into PCIe slot in Laptop and program)	$249	Available on crowd supply currently
4	Snicker Doodle (ARM based SoC)	● Xilinx Zync ● Dual Core ARM Cortex A9 ● Wifi and BLE module ● Crypto EEPROM	$95	Available on crowd supply currently

Appendix G - Glossary

High Security - Using cryptography to prevent unauthorized access to digital information. Data integrity and authenticity.
Formal Assurance/Formal Methods - Formal assurance requires reliability, which translates to 99.999% functional over a legitimate timespan. The software library should be functioning for all possible inputs and there should be no loss of data from memory. A model checking tool

(Formal Methods technique) like SAW can be used for exhaustive logic based testing. Cryptol can also be used for exhaustive testing and can generate a mathematical proof of the algorithm used.
Crypto Algorithms - Mathematical algorithms, usually implemented in software, that are able to encrypt or decrypt data as a measure of security.
SAW (Software Analysis Workbench) - Formal verification software that is primarily used to verify cryptographic algorithms.
AES (Advanced Encryption Standard) - Software standard used implement reasoning, performance and accuracy.
Threat Modeling - Optimizing security by identifying vulnerabilities and defining countermeasures to prevent threats to the system.
Model Checker - Technique for automatically verifying correctness of all possible states within a system.
High Performance - Take reference benchmark, small program or case study, check before and after loading using the Sodium wrapper (Sodium wrapper dictates benchmark specs). Motivation for this is to exceed the processing speed of the original software benchmarks.
Random Number Generator (RNG) - An algorithm that generates a random number between some specified minimum and maximum value.
Hash Functions - A function that verifies that input data maps to a given hash value. This value is usually stored in a hash table that links the input to the corresponding hash value.
Ciphers - It is an algorithm in cryptography for performing encryption and decryption. To encipher or encode is to convert information into cipher or code.
Symmetric Ciphers - In a symmetric cipher(also known as secret key), the key that deciphers the cipher text is the same of can be derived from the key enciphers that clear text. Ex. AES and DES
Asymmetric Ciphers - The asymmetric cipher uses two keys : private and public that define who can view the content. These two keys cannot be derived from each other. Ex. RSA and DSA.
API (Application Programming Interface) - A set of clearly defined methods and/or specifications for communicating between various software components.

Bibliography

[1] M. El-Abd, “A Review of Embedded Systems Education in the Arduino Age: Lessons Learned and Future Directions,” *International Journal of Engineering Pedagogy*, vol. 7, no. 2, pp. 79–93, Apr. 2017.

[2] S. A. Browning, M. Carlsson, L. Erkok, J. Matthews, B. Martin, and S. Weaver, *“Empowering the Experts: High-Assurance, High-Performance, High-Level Design with Cryptol.”* [Online] Available: https://cryptol.net/files/empowering_the_experts.pdf [Accessed February 14th 2018]

[3] *“SAW: The Software Analysis Workbench”* [Online] Available: https://saw.galois.com [Accessed February 14th 2018]

[4] Daniel Burris, *“The Internet of Things is Far Bigger than Anyone Realizes”* [Online] Available: https://www.wired.com/insights/2014/11/the-internet-of-things-bigger/ [Accessed February 16th 2018]