XTEA Implementation on the Nexys A7

Adrian Alarcon, Isaac Elizarraz, Jacob Swenke, Mohamed El-Hadedy
ECE-Department, College of Engineering, California State Polytechnic University, Pomona
Email: {ielizarraz, jcswenke, aalarcon1, mealy}@cpp.edu

***Abstract* – In the modern world, data security has become one of the most widely discussed and researched topics, and for good reason. As human lives become more dependent on technology, society desires peace of mind when working with information on the web, and in the hardware processing that information. In this report, we discuss the implementation of the extended tiny encryption algorithm (XTEA) on the Nexys Artix-7 100T FPGA development board using VHDL code, written and synthesized in Xilinx Vivado.**
Keywords* – *Nexys Artix-7, FPGA, XTEA

I. INTRODUCTION

Data security is a main concern for many people nowadays; user data is exchanged on many social platforms and leave a trace for anyone hacking, snooping, sniffing, cloning, counterfeiting on a network. Not only social platforms, but also RFID, smart meters, smart thermometers, and many of the smart appliances seen in households. Even with being connected to a trusted destination it can still be exposed.

This is where cryptographic algorithms step in as they are specialized for implementation in constrained environments. With resources found in a variety of application different algorithm protocols have emerged. They can be categorized as block encryptions, extended tiny encryption algorithm (XTEA) is one of these block encryptions. David Wheeler and Roger Needham of the Cambridge Computer Laboratory developed TEA, XTEA, and XXTEA. XTEA is a 64-bit Feistel block network cryptographic algorithm, it relies on thirty-two rounds and uses a 128-bit secret key. Each evolution of the tiny encryption algorithm line corrects a weakness of its predecessor. For example, TEA suffers from equivalent keys, meaning each key is equivalent to three others, so even though the key is supposed to be 128-bits, TEA only effectively uses only 126 bits. XTEA is presented along with the variable-width block cipher term known as Block TEA, which uses XTEA round function. XXTEA was developed to correct the weaknesses from Block TEA. The differences between the TEA and XTEA algorithms are stark, where the XTEA contains a more complex key-schedule as well as rearrangement of the shift, XORs, and additions in the traditional combinational logic instantiation. [Figure 1 below shows the two Feistel rounds used in the XTEA algorithm.

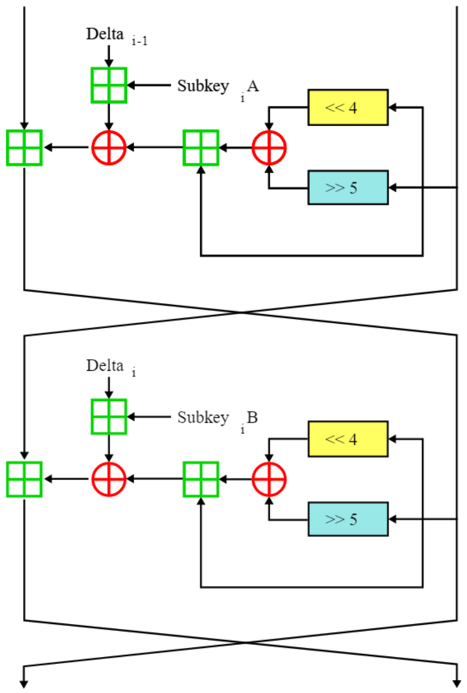


Figure 1: XTEA Algorithm

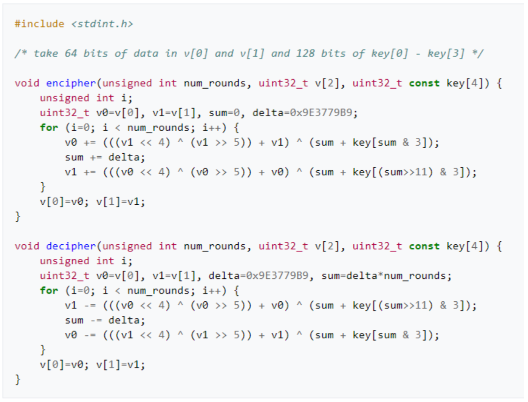
II. BACKGROUND

Hardware

The board selected for this project was the Nexys Artix-7 100T development board. In addition to the board, a VGA cable and a suitable monitor are required for the display of encrypted/decrypted data. To enable UART communication between the board and the host computer, the USBUART (USB to UART interface) Pmod from Diligent was also used in conjunction with Realterm to send serial data.

Software

Shown below is a C implementation of the XTEA algorithm. We used this and implementations in other languages as a model for implementing it in VHDL.



```c
#include <stdint.h>

/* take 64 bits of data in v[0] and v[1] and 128 bits of key[0] - key[3] */

void encipher(unsigned int num_rounds, uint32_t v[2], uint32_t const key[4]) {
    unsigned int i;
    uint32_t v0=v[0], v1=v[1], sum=0, delta=0x9E3779B9;
    for (i=0; i < num_rounds; i++) {
        v0 += (((v1 << 4) ^ (v1 >> 5)) + v1) ^ (sum + key[sum & 3]);
        sum += delta;
        v1 += (((v0 << 4) ^ (v0 >> 5)) + v0) ^ (sum + key[(sum>>11) & 3]);
    }
    v[0]=v0; v[1]=v1;
}

void decipher(unsigned int num_rounds, uint32_t v[2], uint32_t const key[4]) {
    unsigned int i;
    uint32_t v0=v[0], v1=v[1], delta=0x9E3779B9, sum=delta*num_rounds;
    for (i=0; i < num_rounds; i++) {
        v1 -= (((v0 << 4) ^ (v0 >> 5)) + v0) ^ (sum + key[(sum>>11) & 3]);
        sum -= delta;
        v0 -= (((v1 << 4) ^ (v1 >> 5)) + v1) ^ (sum + key[sum & 3]);
    }
    v[0]=v0; v[1]=v1;
}
```

Figure 2: C Implementation of XTEA Algorithm

This specific screenshot of the C implementation was taken from the XTEA Wikipedia page but was released into the public domain by David Wheeler and Roger Needham, the designers behind XTEA. We will summarize the background of the XTEA algorithm in the rest of this section.

III. Design & Implementation

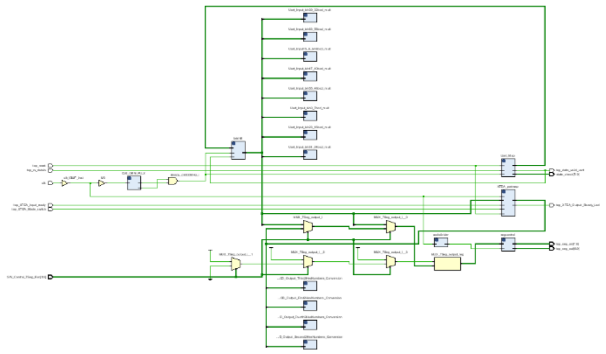


Figure 3: Elaborated Design on Vivado

Shown above is the elaborated design of the group's most updated code for the project. Our design combines Derek Wang's VHDL implementation of the XTEA algorithm with elements of labs that we have completed during class. If everything were to be working as intended, we would be using the UART pmod to send the keys (four different ones) where two 32-bit portions are encrypted and decrypted. In our most updated version, we had these two 32-bit numbers being displayed on the seven segment displays, where the bottom 32 bits are displayed initially, then a switch flip causes the top 32 bits to be displayed. The displays would show the bottom and top half of the 64-bit encrypted string and after pushing a button to trigger encryption, would then show the encrypted string. Another button push would cause the encrypted string to then become decrypted, showing the XTEA algorithm at work.

The group originally intended to integrate the project with VGA but ran into issues displaying strings on the screen that are being updated, since we only have experience showing a static image and changing color as well as position. If the group were to go back to the project later, integrating the display would be at the top of our to-do list as trying to display long numbers on the seven segments was inconvenient for the user. Ideally, we could display the keys on the seven segments and the encrypted/decrypted data using VGA on a separate screen or monitor. Since the code is quite lengthy, specifics regarding the design of each entity will not be discussed and the reader should refer to the attached project for the exact methods.

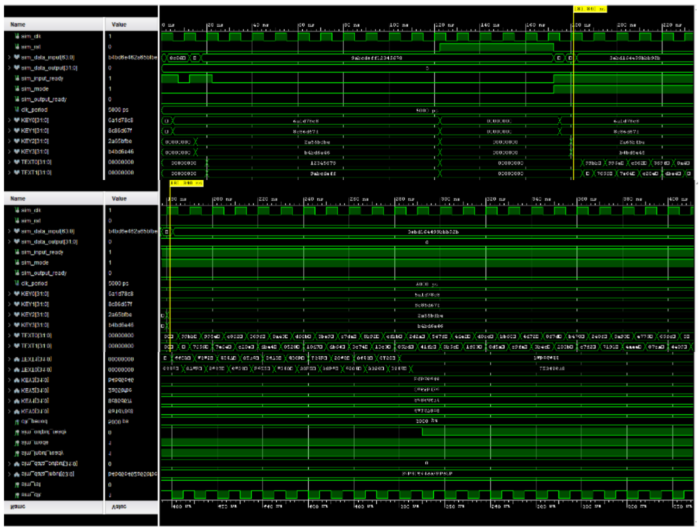
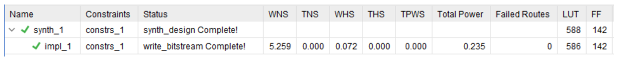


Figure 4: Simulation Screenshots

The group was able to simulate the XTEA module by itself (as shown above) with results that proved it was working, the only issues came in our attempts to get the integrated project running on the board – mainly when including the UART and the VGA.

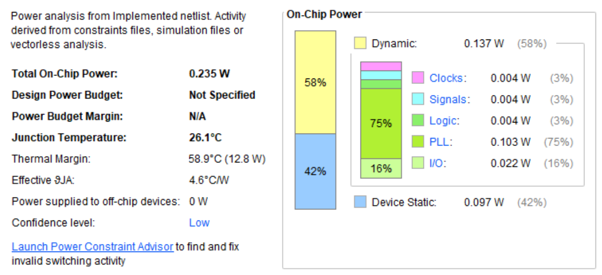
IV. Results & Discussion

Unfortunately, we ran into a few walls when trying to integrate everything together and we would need more time to debug the issues we are facing with our current knowledge of VHDL. Shown below are the pictures of the power usage as well as flip flop and look up table count. One of the things that sticks out is how much power PLL consumes, and as we used it for the UART we were wondering about better methods of implementation that would help reduce this section.



Name	Constraints	Status	WNS	TNS	WHS	THS	TPWS	Total Power	Failed Routes	LUT	FF
✓ synth_1	constrs_1	synth_design Complete!								588	142
✓ impl_1	constrs_1	write_bitstream Complete!	5.259	0.000	0.072	0.000	0.000	0.235	0	586	142

Figure 5: Design Run



Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

Total On-Chip Power:	**0.235 W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**26.1°C**
Thermal Margin:	58.9°C (12.8 W)
Effective ϑJA:	4.6°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Launch Power Constraint Advisor to find and fix invalid switching activity

On-Chip Power

🟨	Dynamic:	0.137 W	(58%)
🟪	Clocks:	0.004 W	(3%)
🟦	Signals:	0.004 W	(3%)
🟩	Logic:	0.004 W	(3%)
🟩	PLL:	0.103 W	(75%)
🟩	I/O:	0.022 W	(16%)
🟦	Device Static:	0.097 W	(42%)

Figure 6: Power Usage

V. Conclusion

The purpose of this project was to use the knowledge we have gained over the semester along with modules we have written in previous labs. Utilizing the Nexys A7 FPGA development board along with the required items for UART and VGA implementation, we came very close to being able to integrate everything we have learned thus far. Learning about XTEA was an interesting experience for the group and we look forward to applying our practice in VHDL from this semester in similar projects in the future.

VI. References

[1] M. H. AlMeer, "FPGA implementation of a hardware XTEA light encryption engine in co-design computing systems," Luton, UK, 2017.

[2] S. Sopna and R. Muthaiah, "Implementation of High Throughput Extended TIny Encryption Algorithm Block Cipher in Field Programmable Gate Array," *Indian Journal of Science and Technology*, vol. 9, 2016.

[3] Jens-Peter Kaps, Chai-Tea, Cryptographic Hardware Implementation of XTEA, 2008 https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.178.6719&rep=rep1&type=pdf