The DEATHSTAR

Data Encryption Accelerator That Handles Security Tasks Anytime Readily

By Lynn Asiimwe, Daniel Galasko and Victor Mushabe of P15A
YODA Phase 4 • Design Review Final Report • 18 May 2012

Table of Contents

Introduction

In today′s world the security of information sent over a network is a top priority. Many organisations take pride in their ability to intercept sensitive data with malicious intent. One of the only means of combatting this interception of sensitive data sent over a network (such as credit card details) is a process called Encryption.

Encryption is simply a means of preventing unauthorised access to information by converting that information into an unreadable format known as cipher-text. Cipher-text can only be converted into readable plaintext with a unique key, this key is known only to the intended recipient of the information and this way no one else can decrypt the data without this key.

As computers use encryption more and more the bottleneck of the speed of the encryption process performed by the CPU becomes increasingly evident. The solution is to design a Data Encryption Accelerator (DEA) that can perform data encryption for a computer instead of relying on the CPU, thus improving the speed of the encryption process and removing the added strain on the CPU in performing encryption.

We propose the DEATHSTAR (Data Encryption Accelerator That Handles Security Tasks Anytime Readily) as a means of solving this processing problem.

OVERVIEW OF TOPIC AND REPORT STRUCTURE

The purpose of the DEATHSTAR is to accelerate the process of encryption. As a result, it alleviates the computer from wasting resources encrypting data and thus allowing it to focus on other tasks more efficiently.

This report will begin by identifying the important skills required to develop the DEATHSTAR as well as the methodology for developing the DEATHSTAR. There are two possible approaches for development namely plan A and plan B. A draft model of the DEATHSTAR will then be presented in the means of a block diagram providing a detailed description of how the system works. After that a simple strategy for evaluating the effectiveness of the DEATHSTAR will be presented as well as paper-based calculations and estimations of performance criteria. Sample code will also be provided as a means of showing the golden based measure for Data Encryption using XOR'ing. Finally some results will be discussed and conclusions based on those results will be made.

Skills Identification and Proposed Methodology

This section of the report will begin by addressing the fundamental skills that will be required in development of the DEATHSTAR FPGA application. After that the proposed methodology that will be used in developing the DEATHSTAR will be described in detail. Two approaches will be identified in this section namely plan A and plan B. plan B is a simplified version of a DEA using only XOR encryption and one key whereas plan A will detail asymmetric encryption and prime number key encryption including the use of both public and private keys.

IMPORTANT SKILLS NEEDED

Development of the DEATHSTAR requires a team that has knowledge over a vast amount of information. First the team needs to understand encryption. This includes the understanding of the mathematics behind asymmetric encryption so it is expected that team members be very proficient in mathematics. This understanding in mathematics further comes in use through the application of algorithm analysis which is another fundamental skill required. It is pivotal that the team members be able to identify means of making algorithms more efficient and reducing algorithm complexity in order to improve overall processing time which forms part of the fundamental aim of the DEATHSTAR. Since the implementation is to be done on the Nexys 2 team members are to have a significant understanding of the Nexys 2 layout and architecture. HDL Programming and schematic design are therefore fundamental aspects of FPGA utilisation and thus crucial skills needed for development. Since a golden measure is also to be designed it is expected that the team members be familiar with coding in C so that a competent golden measure can be constructed for comparison between the serial method run on a computer and the parallel method run on the Nexys 2.

METHODOLOGY

Plan A

Project Objective

Plan A represents the ideal approach in developing the DEATHSTAR. In this approach the objective is asymmetric encryption involving the use of both public and private keys for enhancing security. To strengthen the encryption process the industry norm is to use a unique, large prime number to generate the key. This system will need to accelerate the process of generating both public and private keys (performed at receiver) as well as accelerating the process of encrypting plaintext to cipher-text (performed by sender)

Identified Design Problems

- Time is an important factor in the completion of the YODA project and thus directly affects the feasibility of achieving plan A. It has been realised that given the strict time constraints in the construction of the DEATHSTAR that plan A might not be realised.

- Plan A also places a stronger emphasis on the understanding of encryption as well as the ability to speed up the generation of prime numbers typically used in asymmetric encryption. This means a steeper learning curve and skill set required for development.

- The Nexys 2 boards available might not be able to provide a solution that is viable given that there are newer and more powerful FPGA boards currently on the market

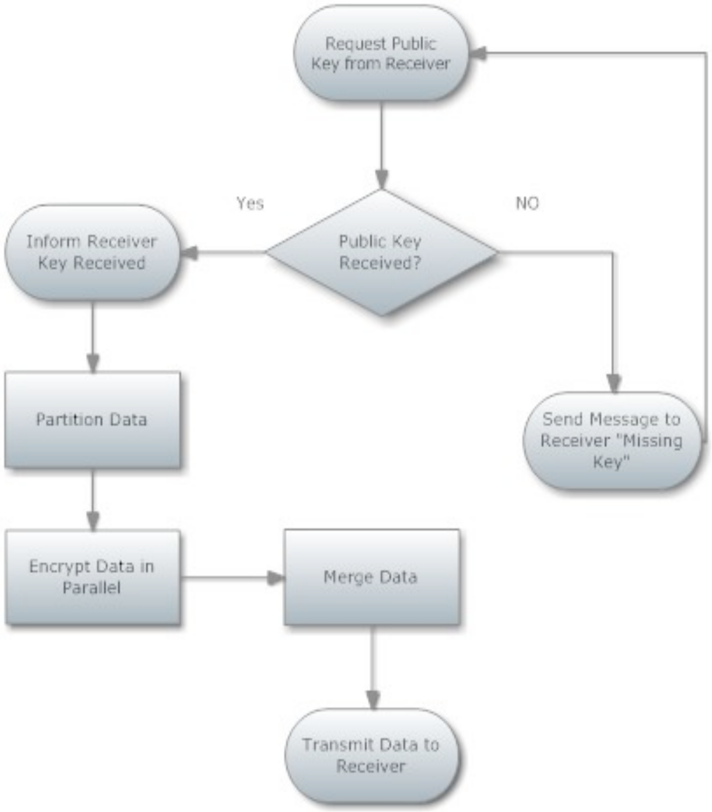
- The small clock rate of 50 MHz for the Nexys 2 is a potential design problem in that it might inhibit the FPGA from outperforming a modern computer

Proposed Solution

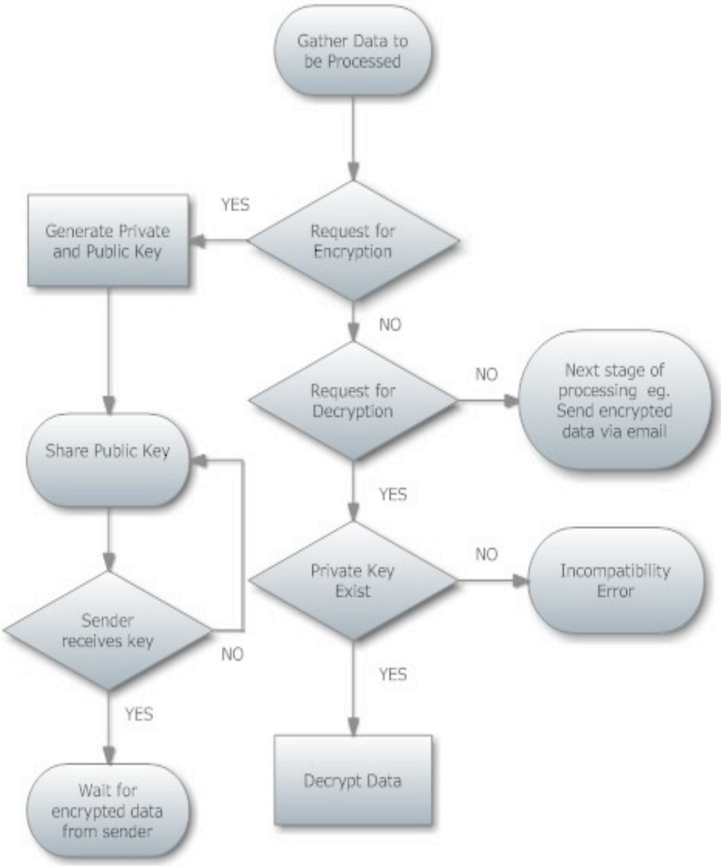
The proposed solution is an ultra fast asymmetric encryption key generator and plain-text to cipher-text encryptor built into an FPGA.

Evaluation Process

Evaluation will be performed comparing the FPGA performance with the performance of a standard Intel core 2 duo processor


Request Public Key from Receiver
Public Key Received?
Yes
NO
Inform Receiver Key Received
Send Message to Receiver "Missing Key"
Partition Data
Encrypt Data in Parallel
Merge Data
Transmit Data to Receiver

Sender End Process Flow


Gather Data to be Processed
Request for Encryption
YES
Generate Private and Public Key
NO
Request for Decryption
NO
Next stage of processing eg. Send encrypted data via email
Share Public Key
YES
Private Key Exist
NO
Incompatibility Error
Sender receives key
NO
YES
YES
Decrypt Data
Wait for encrypted data from sender

Receiver End Process Flow

Plan B

Project Objective

Plan B is the risk-reduced approach to develop the YODA product. The objective of Plan B is to design a basic DEA that uses a simple encryption key that is XOR'ed with the original text to create the cipher-text. In this methodology the XOR key is generated by the receiver and transmitted to the sender who then encrypts the data with the key and transmits to the receiver. The information consists of a 5 byte data packet with the first byte being the header information and the latter 4 bytes the data to be encrypted. This 5 byte value is random and can easily be made to be any number of bytes however for demonstration purposes five was chosen for simplicity.

Identified Design Problems

- The generation of XOR gates on an FPGA to be used in the encryption process
- The attempt at making encryption of a single data stream relatively fast
- Breaking up a plaintext data stream into chunks so that each chunk can be encrypted in parallel

Proposed Solution

The obvious solution to accelerating encryption is through parallelization. By dividing a message into chunks and encrypting each chunk in parallel a significant speedup can be performed. This can only occur if different parts of the message can be operated upon independently.

Evaluation Process

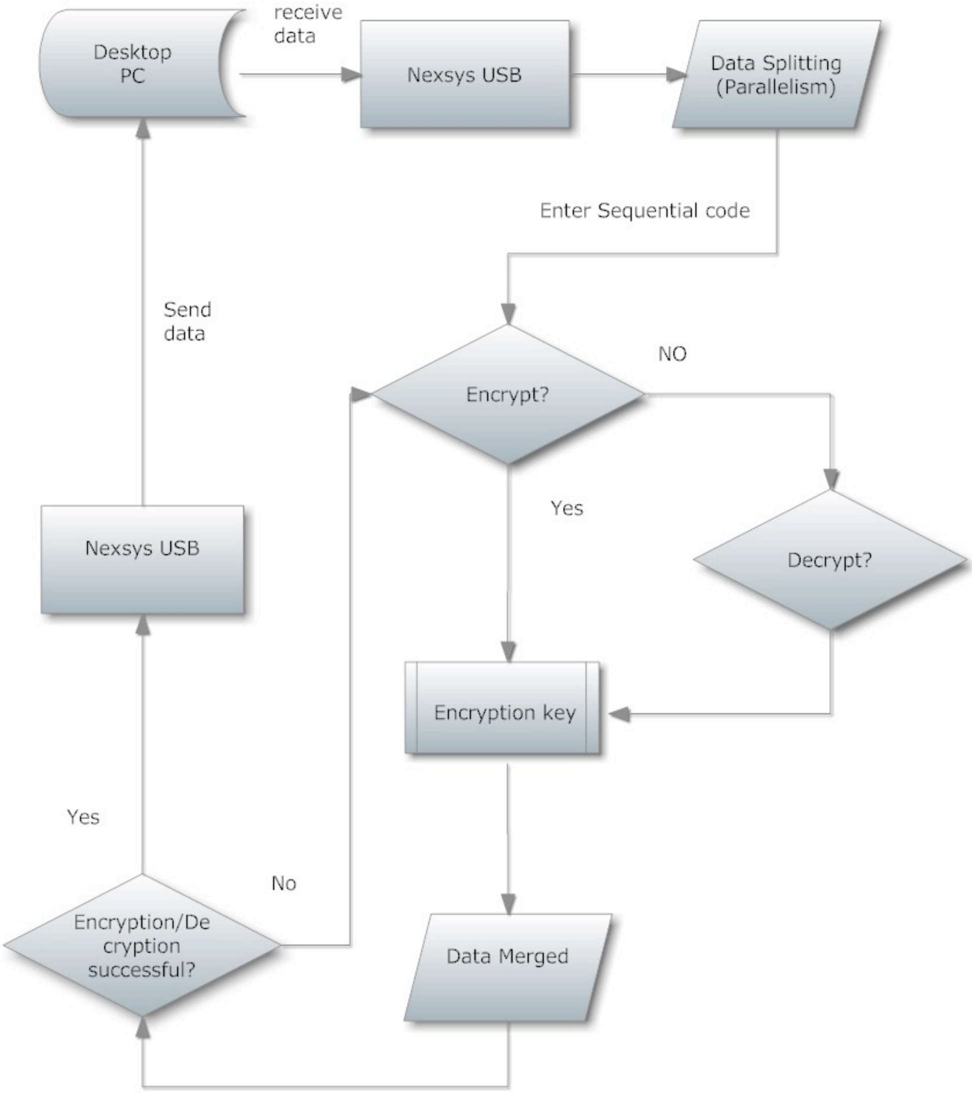
Evaluation will be performed comparing the FPGA performance with the performance of a standard Intel core 2 duo processor

Modelling and Analysis

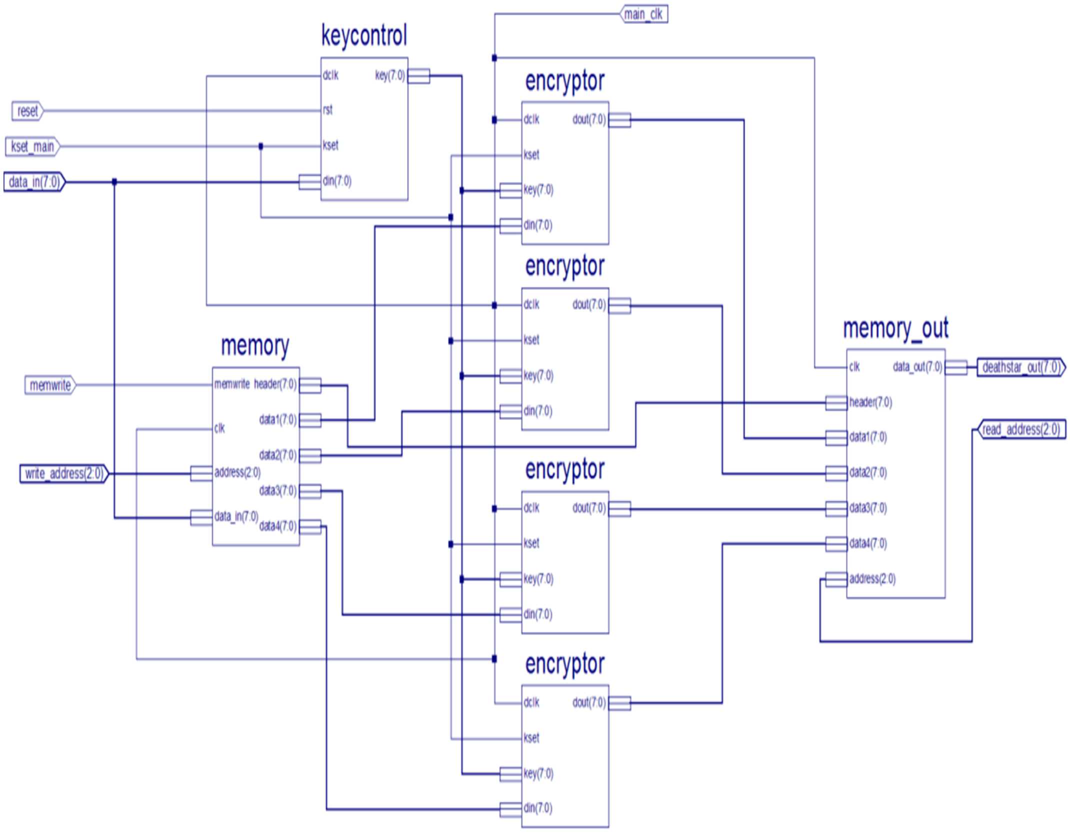
DEATHSTAR DRAFT

The DEATHSTAR would retrieve data from the system it is connected to, there after it would split the data into streams such that the encryption code (sequential code) can be run on each stream in parallel.

After each stream has been processed the data is recombined and ready for transmission to the network. Below is a flowchart of the encryption method:



The DEATHSTAR was then designed in VHDL under the assumption that each data packet was five bytes in length and contained a one byte header which, for obvious reasons, is not encrypted. The following schematic was produced for plan B:



The DEATHSTAR Schematic

Note: the code for the DEATHSTAR is not included in this document but is attached as code

STRATEGY FOR DESIGN EVALUATION

Exclusive OR encryption is a symmetric encryption method (The same encryption key is used for both the encryptor and decryptor). Exclusive OR works by applying the Exclusive OR Boolean algebra operator $\oplus$ to every bit in the stream with a symmetric key .The key generation is done by a random number generator.The truth table below for an XOR operation:

A	B	A XOR B
1	1	0
1	0	1
0	1	1
0	0	0

This type of encryption can be applied to any type of file for example music files,documentation file,image files etc. since it is done at a binary level.It's almost unbreakable by brute force.

Limitations:

- It is very liable to patterns however this can be fixed by first compressing the file.
- Since the key is shared by the encryptor and decryptor,it can only be used by small institutions i.e people within the same community.
- The clock rate of the FPGA board being significantly less than most modern day computers

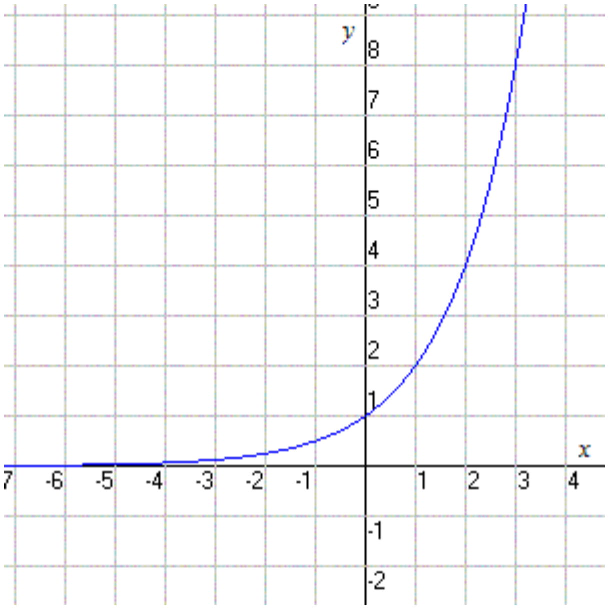
In order to evaluate our design we will be timing the implementation of the DEATHSTAR using the Xilinx ISE Project Navigator. The time for successful encryption in the simulation will be compared against the timing of the golden measure results obtained on the computers in the blue lab.

Sample Code for the golden measure, including timing:

```
#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <sys/time.h>
#include <math.h>
#include <fstream>

using namespace std;
/
*============================================================================
 ENCRYPTION FUNCTION
===========================================================================*/
void encrypt(string original)
{
    string encrypted = "";
    string key="bluelabs";

    int x=0;

     for (int temp = 0; temp < original.size(); temp++)
     {
        if(x==7)
             {
                    x=0;
             }
             encrypted += original[temp] ^ (int(key[x]) + temp) % 255;
             x++;
     }



}
/
*============================================================================
 DECRYPTION FUNCTION
===========================================================================*/

void decrypt(string enfile)
{
     string unencrypt;
     string key="bluelabs";
     int x=0;
     for (int temp = 0; temp < enfile.size(); temp++)
     {
             if(x==7)
             {
                    x=0;
             }
     unencrypt += enfile[temp] ^ (int(key[x]) + temp) % 255;
             x++;
     }
}

/
*============================================================================
 MAIN CODE
===========================================================================*/
int main()
{
    char x;
    string line;
    struct timeval start_time, end_time;
    double run_time, total_time;
    ifstream myfile ("text.file");
    if (myfile.is_open())
    {
        while ( myfile.good() )
        {
            getline (myfile,line);
            if(line!="")
            {
                gettimeofday(&start_time, NULL);

                /* Step 4: Call a user function! */
                encrypt(line);

                /* Step 5: Check & print elapsed time */
                gettimeofday(&end_time, NULL);
                run_time = ((double)(end_time.tv_sec) + (double)
                           (end_time.tv_usec)/1000000.0)
                           - ((double)(start_time.tv_sec) + (double)
                           - (start_time.tv_usec)/1000000.0);
                total_time+=run_time;
                //cout << run_time << " RUN TIME";

            }
        }
        myfile.close();
    }

    cout << total_time << " RUN TIME \n";

}
//============================================================================
```

ESTIMATION OF PERFORMANCE

Given a key space of $n=2^k$ with H characters, the running time can be represented by $O(2^{Hk})$. The graph below shows the estimated performance:



From the graph it shows that estimated running time will double every time an input element is added.This shows that this kind of algorithm is meant for small datasets. For example, an $O(2n)$ algorithm which takes an input of 30 elements may need to perform as many as 1 billion steps. If the input has 40 elements then the 1 trillion steps may be necessary. Few computers in the world can do this in a reasonable amount of time.

With the FPGA solution the complexity remains $O(2n)$ however there will be multiprocessors working in parallel therefore a small datasets and this will reduce the overall computation time.

At the outcome of the experiment we would expect the DEATHSTAR to outperform the computer.

Results and Conclusion

The golden measure was able to encrypt a 5 byte file in approximately $5.6\mu s$. The DEATHSTAR was able to encrypt the same 5 byte file (including copying the data to memory and the assumption of a 50 MHz clock rate) in $130ps$. Thats $0.00013\mu s$ and a significant improvement. This means that we have succeeded in being able to accelerate the process of encryption on an FPGA if only in simulation.

We are very confident in the ability of FPGA's to outperform computers in encryption based tasks and feel that given more time we could easily have come up with a more robust and practical DEA.

INFORMATION SOURCES

PCWorld Communications, Inc,"How it works encryption" [online] Available at http://www.pcworld.com/article/15230/how_it_works_encryption.html [Accessed 10 May. 2012]

Simon Winberg ,[lecture notes] "Lecture 18 on Amdahl's law"[online] Available at https://vula.uct.ac.za/portal/site/a1a608c2-e9e3-4c2c-844b-d4cf5bc878c0 [Accessed 10 May. 2012]

CPlusPlus.com,2011. "Simple XOR Encryption. [online] (Mar 13. 2011) Available at: http://www.cplusplus.com/articles/Ly86b7Xj/ [Accessed 10 May. 2012]

Area72,"Simple XOR algorithm"[online] Available at http://area72.ro/programming/simple-xor-encryption.html [Accessed 2 May. 2012]

Rob Bell,"A beginners guide to Big O" [online]Available at http://rob-bell.net/2009/06/a-beginners-guide-to-big-o-notation/ [Accessed 7 May. 2012]