The Iby and Aladar Fleischman Faculty of Engineering
Tel Aviv University

OCB

Project Number: 22-1-1-2494
Names: Basel Mansour, Adan Khaled
Advisor: Oren Ganon

Introduction

OCB is a block cipher-based mode of operation that simultaneously provides both privacy and authenticity for a user-supplied plaintext. Such a method is called an *authenticated-encryption scheme.* What makes OCB remarkable is that it achieves authenticated encryption in almost the same amount of time as the fastest conventional mode, CTR mode, achieves privacy alone.

Using OCB one buys privacy and authenticity about as cheaply (or more cheaply—CBC was once the norm) as one used to pay for achieving privacy alone. Despite this, OCB is simple and clean, and easy to implement in either hardware or software. **OCB accomplishes its work without bringing in the machinery of universal hashing**, a technique that does not seem to lend itself to implementations that are simple and fast in both hardware and software.

Motivation/Objectives

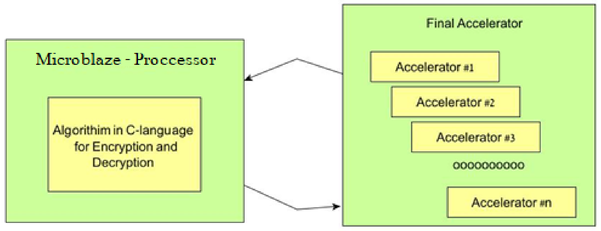
Why even run an encryption algorithm? And why should you speed up? Important questions whose answer is simple ... security. Encryption is the best way to keep **data secure** so that it protects the contents of the files and no one who doesn't have the "tool" to do the proper decryption will be able to use them, and it's always better to use hardware accelerators for an even simpler reason... **speed,** we've come to 2023 and use file encryption that will take a week, **a few days** or even a few hours sometimes **will harm the company's position** in the market and lose against competitors.

Using software or hardware alone to perform the encryption gives us **either** "good" encryption but with a very long time that can reach a week, or **bad** encryption but with an excellent encryption time, hence the need for an intermediate solution that combines a hardware solution with a software solution.

our main goal after we finish the job is to improve **runtime** as much as we can and in the worst case we get an **improvement** of **10%,** without increasing the on chip **power** by **more than 25%,** or increasing the overall **size** by **more than 40%** of the size of the processor (Microblaze) it self.

Methods/Implementation

One way of achieving the goal is **combining** both **software** and **hardware** by writing the algorithm using **C-Code** and replace parts of it with **accelerator** that is written using **Verilog** like so:



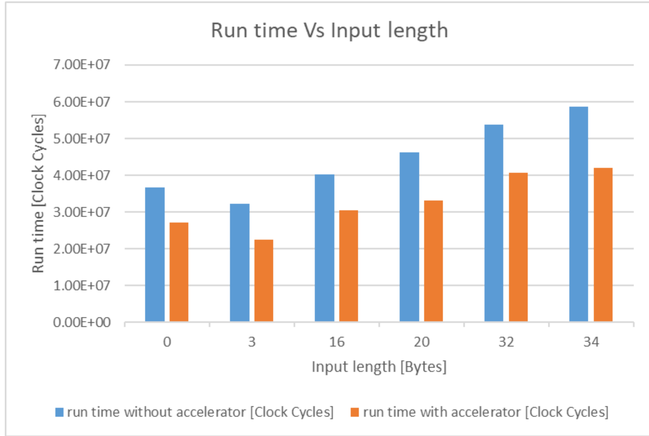
Results

As a result of adding the accelerator that we have designed ,we have achieved even better results that our goal like we can see in the table below:

In average	
without accelerator	run time [Clock Cycles]
	1.02E+08
	power consumption [Watt]
	2.085
	size [Slices]
	21747
with accelerator	run time [Clock Cycles]
	7.64E+07
	power consumption [Watt]
	2.119
	size [Slices]
	21925
run time after Vs before [Clock Cycles]	
25.43%	
power on chip after VS before [Watt]	
1.63%	
size of new IP VS size of MicroBlaze[Slices]	
22.25%	

Conclusions

After analyzing all of the results we got we found that after we add the accelerator we improve the runtime by two orders of magnitude as we can see in the plot below:



Beside that as we can see from the table in the results section we can conclude that it is a good idea to replace parts of the software with RTL in order to get better runtime since we did not increase the size of the hardware and the on chip power by a large number but with a minor amount, which means that we can easily combine it with the existed hardware without leading to unnecessary power consumption or spending a lot of money on fabrication.

Bibliography

[1] OCB: Background by Prof.Philip Rogaway, University of California ,21, Feb 2023.
https://www.cs.ucdavis.edu/~rogaway/ocb/ocb-faq.htm