System-on-Chip for Lightweight Cryptography

Mohd Saufy Rohmad,Emilia Noorsal, Azilah Saparon, Habibah Hashim
Universiti Teknologi MARA,Malaysia
saufy@salam.uitm.edu.my

INTRODUCTION

Seamless Security is very crucial for future Internet of Things. The design of SoC and platform that embed this feature is a promise for fully secure stack of hardware and software. This research present the general architecture of SoC for Present lightweight cryptography. We are designing this SoC to investigate how the Present block cipher could fit inside the GCM (galois counter mode) modes of operation and we want to design full cryptography accelerator with the capabilities to securely exchange keys, store the keys and perform encryption and authentication function. The research will integrate the cryptography accelerator with MIPS microprocessor, UART and ADC to function as a working SoC (we named SoCLW) that can be loaded into FPGA chip. How far is Present better than AES in this kind of design?Will the internal operation influence the performance when integrated in the cryptographic accelerator system? These are the research questions that we are asking ourselves to challenge and prove that lightweight cryptography is the future of hardware cryptography and will become the basis of future security protocols on chip.

Present Ultra-Lightweigt Block Cipher

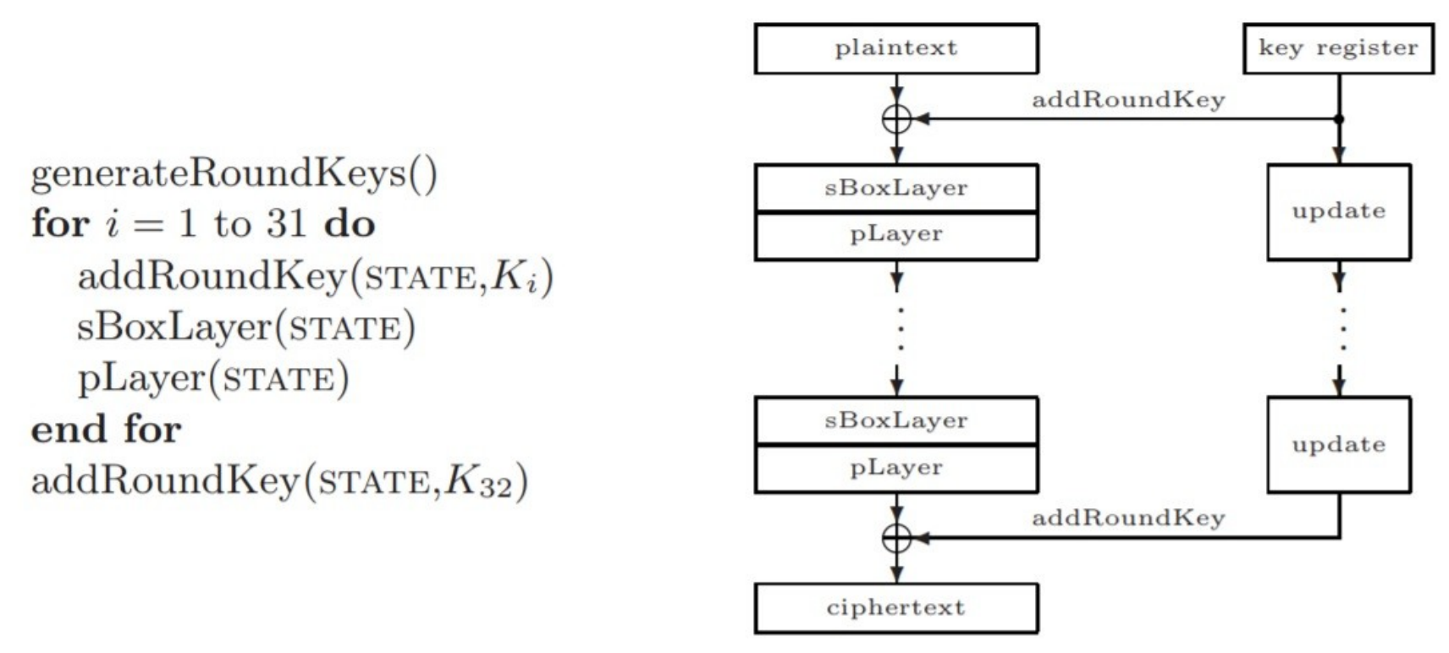

```
generateRoundKeys()
for $i = 1$ to 31 do
    addRoundKey(STATE,$K_i$)
    sBoxLayer(STATE)
    pLayer(STATE)
end for
addRoundKey(STATE,$K_{32}$)
```

Figure 14 : Present Internals

Present is chosen due to its adoption among the research community. The availability of its sample code in C and VHDL also make it easier to be tested and ported to any platform. Compare to AES, present only consist of two native operation while AES comes with 4 operations in each round. Gate equivalent- GE result also shows that present can be implemented with less than 3000 GE and considered as ultra-lightweight block cipher.

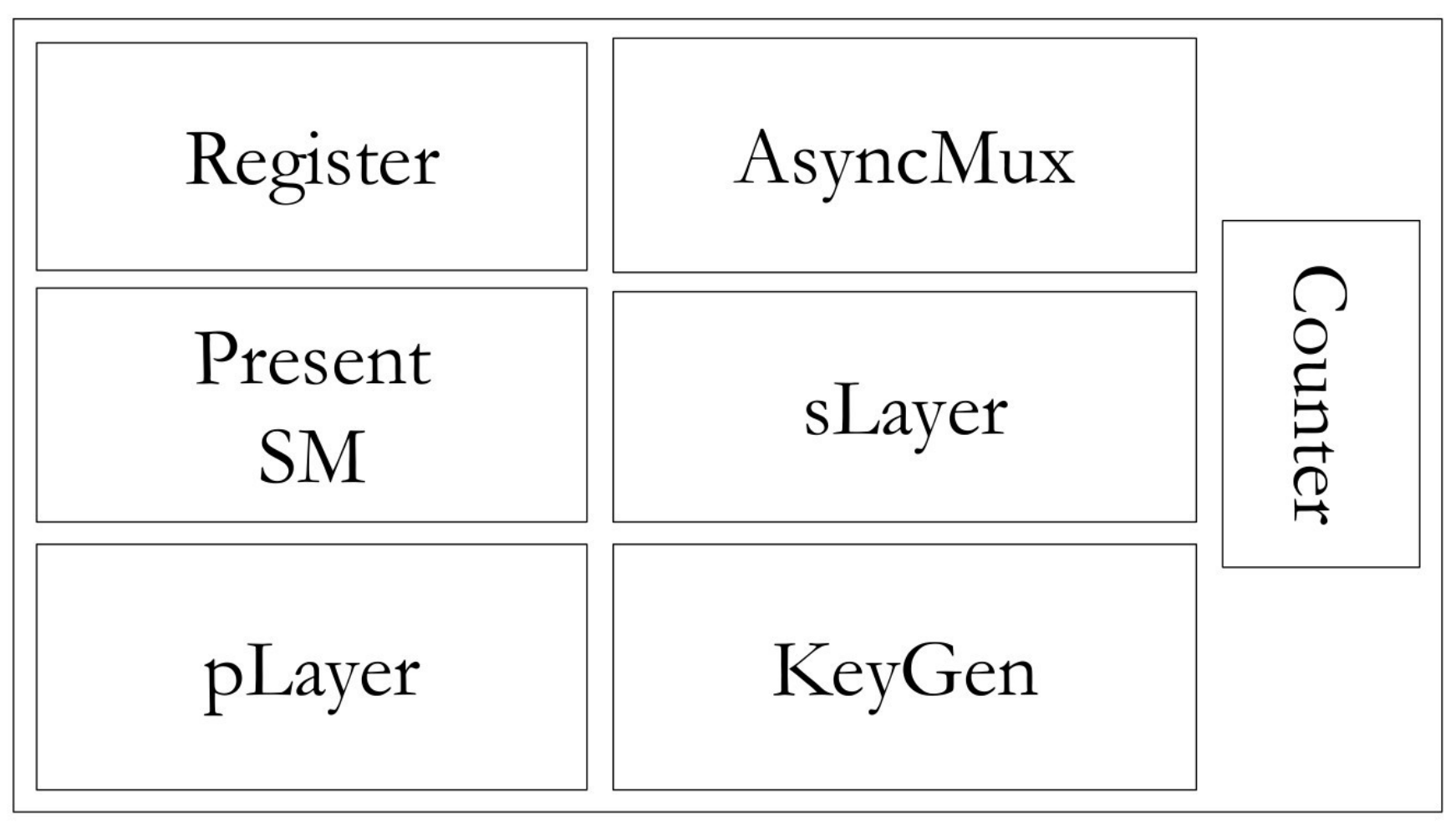


Figure 15: Present VHDL Modules

NOVELTY

This research aim to produce first implementation of present-gcm hardware block that fully function inside a complete working SoC. The technique use to parallelized dual 64 bit presents inside 128 bit GCM modes of operation will be further explore and new implementation techniques will be achieved. This will be a contribution to the body of knowledge and will be further develop to become new IP that can be used in any applications.

PROBLEM STATEMENT & OBJECTIVES

Lightweight cryptography has been proven as a light version of the normal security primitives used in current security systems. A lot of lightweight cryptography algorithms are designed either as a block cipher,or as symmetric ciphers of hash functions. Present is one of the prominent lightweight block cipher that is being studied by the research community worldwide. Papers are published to explore the design space of the Present block cipher but there are still no implementation attempt to really put Present in the heart of crypto-accelerators. This research is an attempt to design a complete working SoC that put Present as the core algorithm for providing various security services. Its security architecture is being studied and analyzed to ensure that the design of the crypto-accelerator is security proof. The expected contribution of this work is the architecture of lightweight cryptography accelerator (or controller) designed to be main hardware IP for lightweight security services. Other secondary aspects such as performance, power analysis on implementation technology, design efficiency also being studied.

SoC DESIGN

The complete SoC is design to enable the fully working environment of end to end Internet of Things systems. There are 4 main sub system in this SoCLW, Present Crypto controller (or accelerator), ADC controller to take input from analog source, MIPS as the microprocessor that interact with the software and UART controller to enable the communication with outside parties. Finally, the SoC is targetted to be program inside Xilinx SP605 FPGA board.

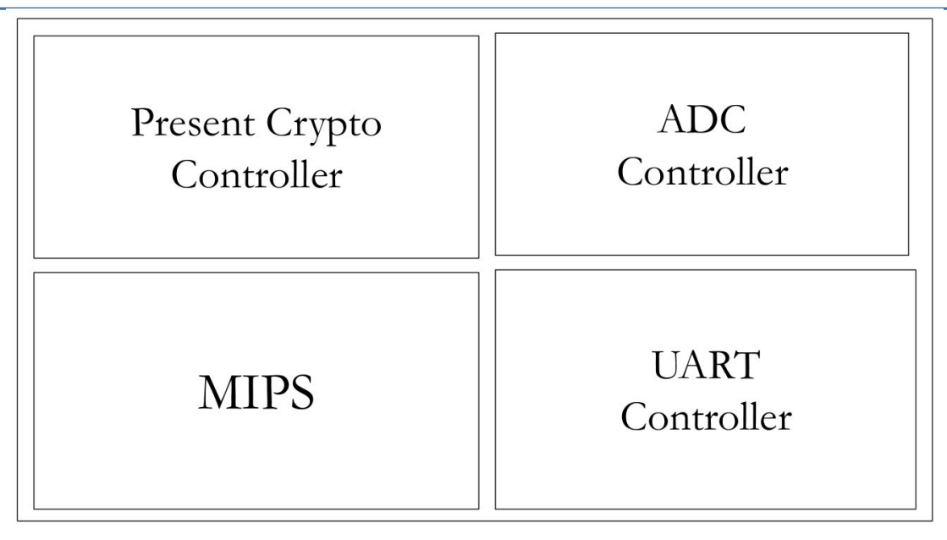


Figure 1 : SoCLW SubSystem

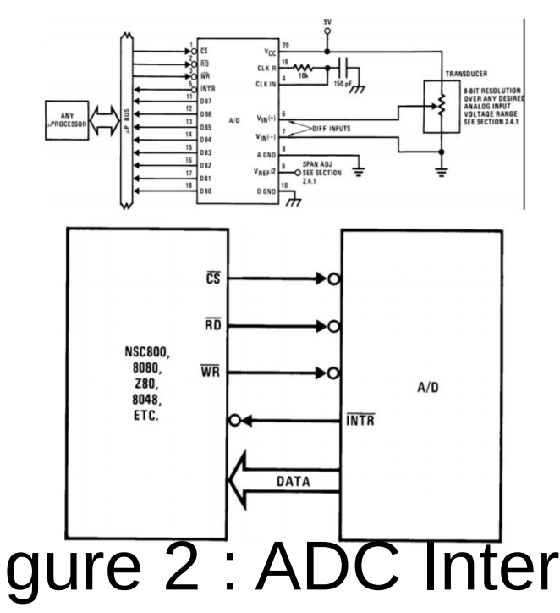


Figure 2 : ADC Interfacing

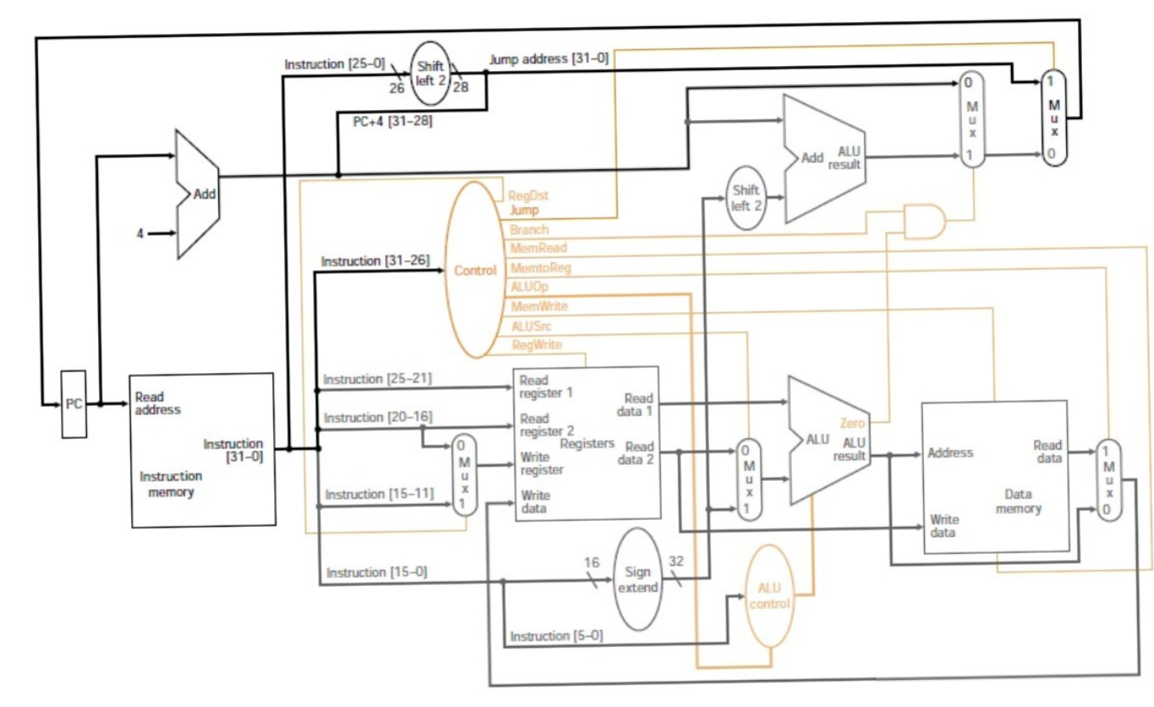


Figure 3 : MIPS

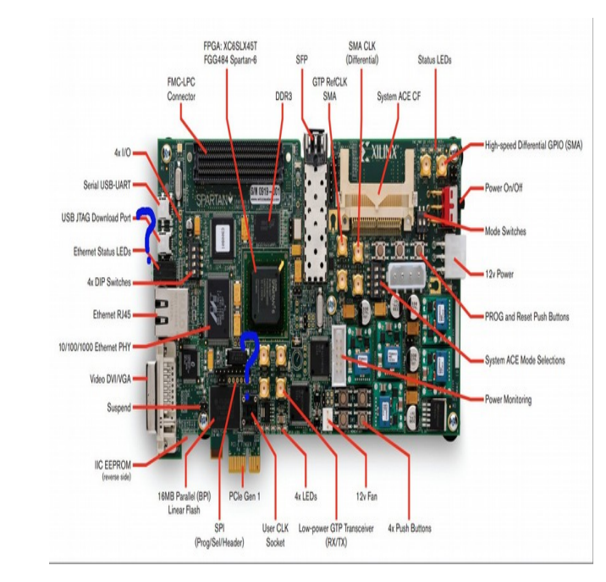


Figure 4 : Xilinx SP605 FPGA

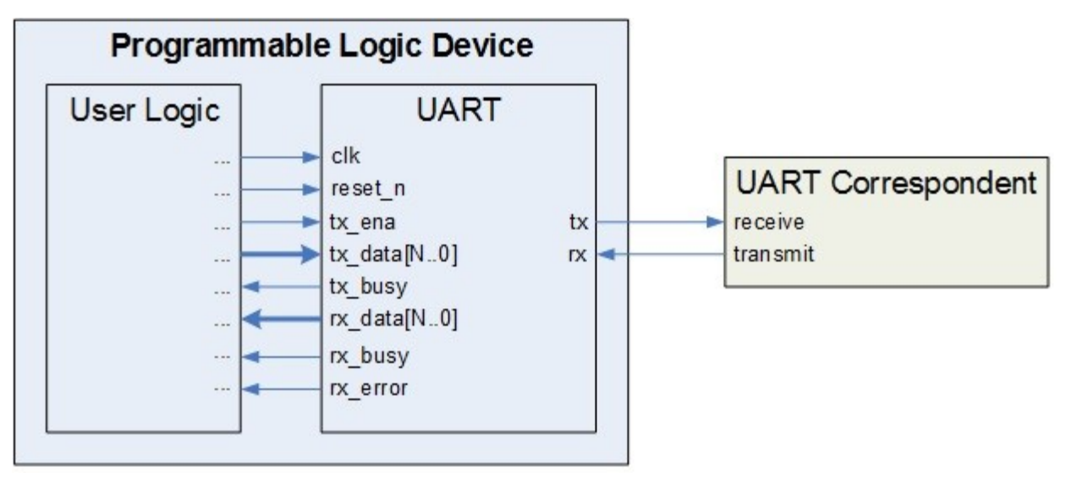


Figure 5 : UART

Lightweight Crypto-Accelerator

This is where the design of the accelerator is. It consist of 6 modules as presented in Figure 6. It then Follow with figure of other module in the accelerators that finally build up the whole accelerator.

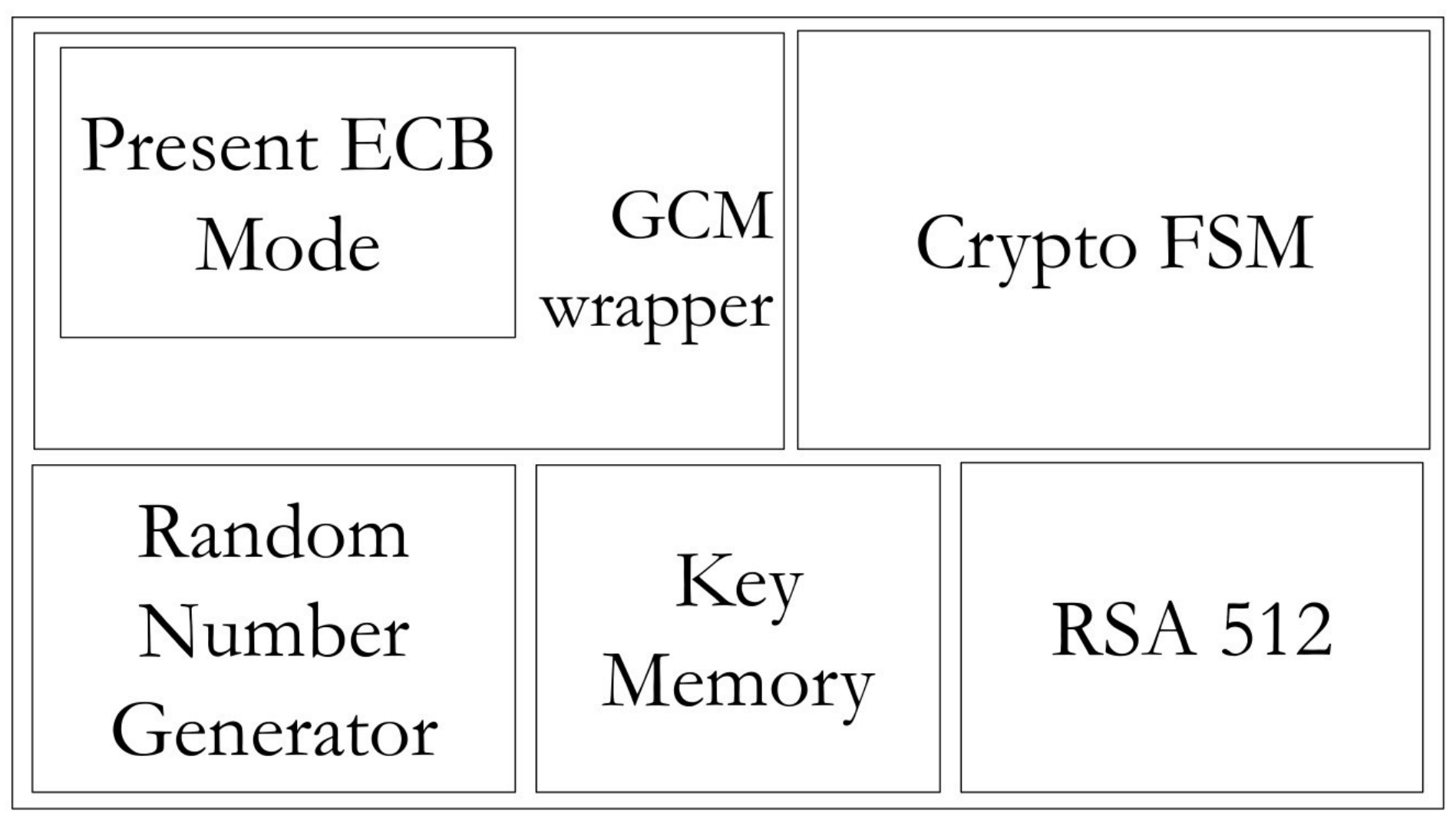
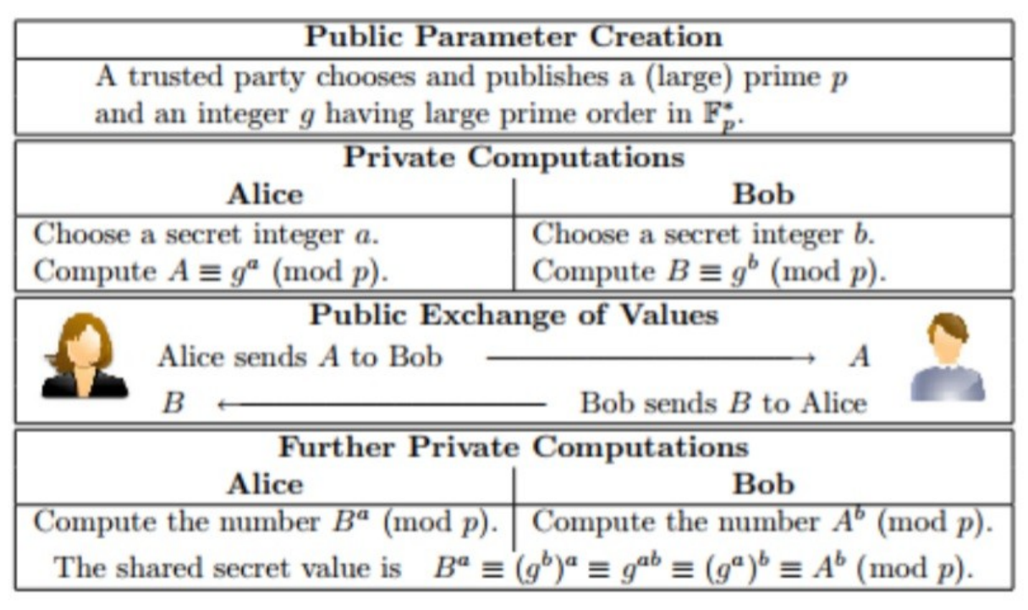


Figure 6 : Crypto Accelerator Modules



Public Parameter Creation

A trusted party chooses and publishes a (large) prime p and an integer g having large prime order in $\mathbb{F}_p^*$.

Private Computations

Alice	Bob
Choose a secret integer a. Compute $A \equiv g^a \pmod{p}$.	Choose a secret integer b. Compute $B \equiv g^b \pmod{p}$.

Public Exchange of Values

Alice sends A to Bob $\longrightarrow$ A

B $\longleftarrow$ Bob sends B to Alice

Further Private Computations

Alice	Bob
Compute the number $B^a \pmod{p}$.	Compute the number $A^b \pmod{p}$.

The shared secret value is $B^a \equiv (g^b)^a \equiv g^{ab} \equiv (g^a)^b \equiv A^b \pmod{p}$.

Figure 7 Diffie Hellman Key Exchange

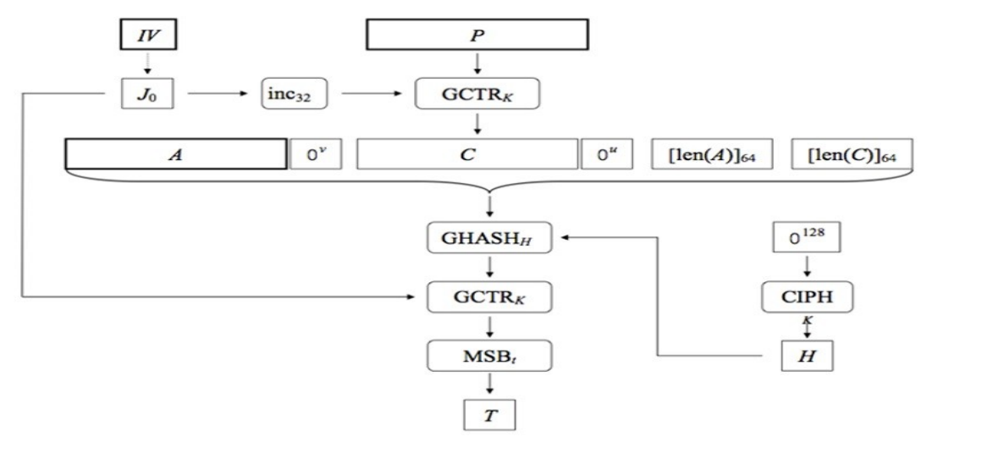


Figure 8 Galois/Counter Mode

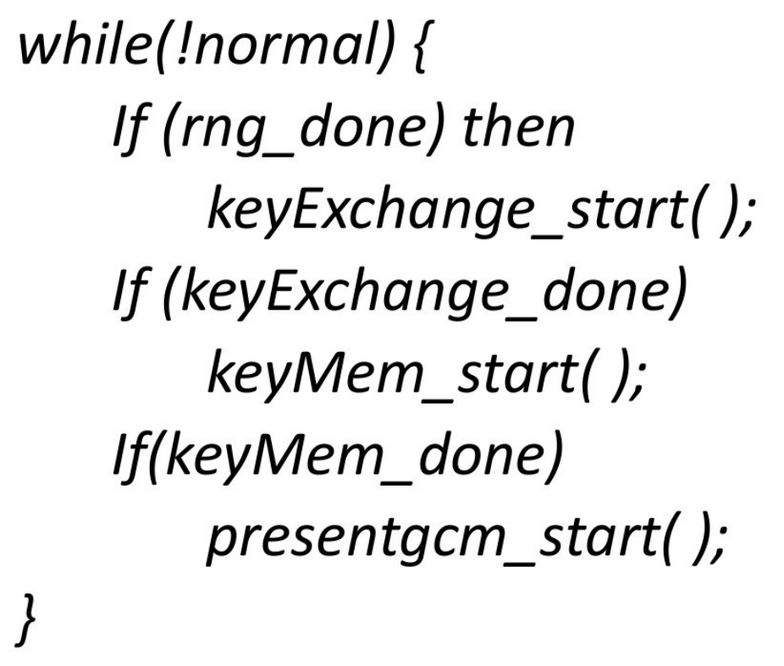

```
while(!normal) {
    If (rng_done) then
        keyExchange_start( );
    If (keyExchange_done)
        keyMem_start( );
    If(keyMem_done)
        presentgcm_start( );
}
```

Figure 11 FSM Logic of Crypto-Accelerator

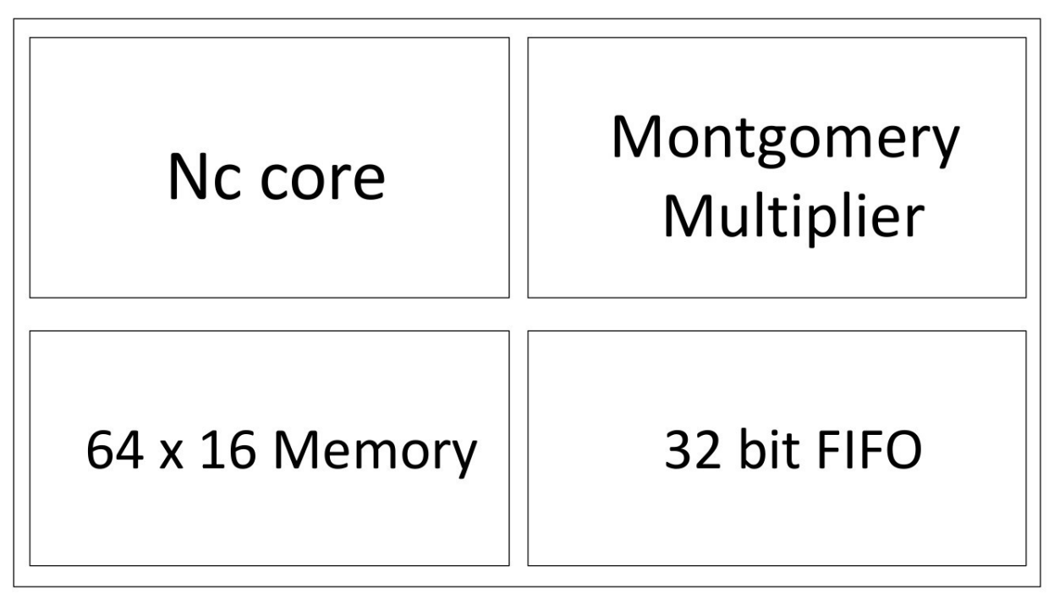


Figure 9 VHDL Module of RSA 512

Figure 7 is the operation of DHKE, DHKE can be considered as the most fundamentals way to exchange keys. Other than that, RSA512 also considered as the algorithm for key exchange. Figure 9 is the internal hardware code block in building RSA512 on VHDL. Figure 8 is the process of generating tag and message in GCM modes of operation. Here present will be use as the algorithm that wrap with GCM. Figure 12 is the process of real random number generator that commonly use. Figure 13 shows the key memory that will be use to store the exchange keys and Figure 11 is the logic that used in the FSM to control the accelerator operation.

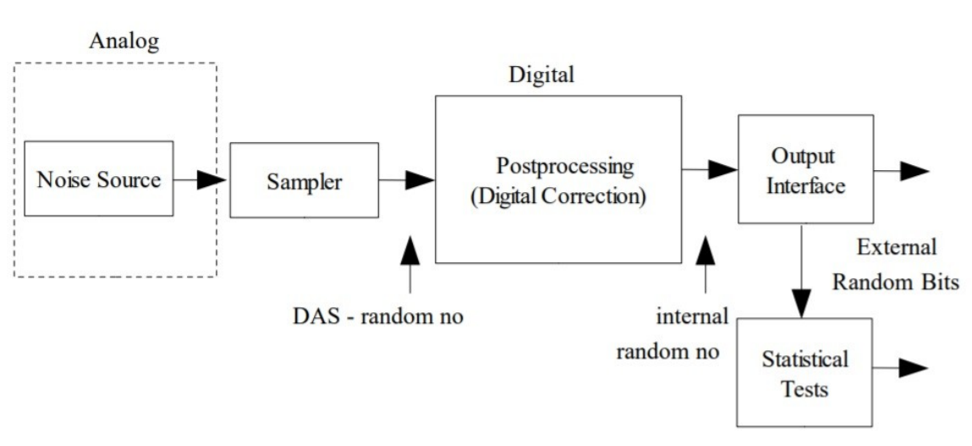


Figure 12 Random Number Generator

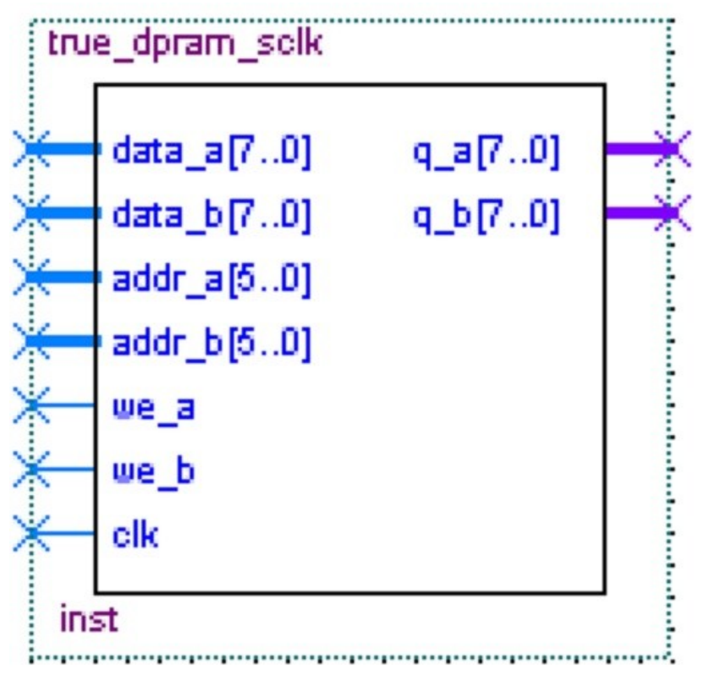


Figure 13 Key Memory

CONTACTS

Prof Dr. Habibah Hashim, Mohd Saufy Rohmad
Head, Information Security and Trusted Infrastructure Laboratory – InstiL
Faculty of Electrical Engineering
Universiti Teknologi MARA
Shah Alam Selangor MALAYSIA
ibahhashim@gmail.com, msaufy@gmail.com