SOC Based Cryptosystem Implementation Using RC4 and IDEA Algorithm

Surekha Pujar[1], Sleeba Mathew[2]

[1]M. Tech DCN, Department of E&C Guide/Asst.Professor

[2]PACE, Mangalore Department.of PACE, Mangalore

Abstract: ***In this paper we present a RC4 & IDEA cryptographic technique implemented on System on chip, IDEA in the sense that we are going to implement both stream cipher and block cipher in a SOC. The encryption of data will be performed by different cipher base on the application.. Here we are integrating IDEA and RC4 cryptographic algorithm for block cipher and stream respectively, so that we can achieve both encryption of data for transmitting through the communication link (i.e. stream cipher) and encryption data like files and images (i.e. Block cipher) can be implemented in single SOC. Finally we are computing the efficiency for both ultra lightweight cryptography (IDEA) and lightweight cryptography (RC4) implemented on single SOC.***

Keywords: IDEA, RC4, System on chip (SOC), Cyclone IV E, Block cipher, Stream Cipher.

1. Introduction

The explosion in the Internet based Electronic Commerce; there is an increasing need to secure the data and commercial transactions. The intent to transmit messages securely is not new. For centuries, people want to keep their communications secret. Hence, a mechanism is required to guarantee the security and privacy of information that is transmitted over the electronic communications media and in the modern era of inexpensive internet connections, data computing and global data communications there is high demand for energy efficiency, computational speedup and data security. There are many effort has been taken in response to ever-increasing need for data security, to protect the data from unwanted action, unauthorized user and destructive force the cryptography plays an important role. Clearly there are two ways to implement any algorithm, i.e. either hardware or software, the choice of platform depends on algorithm performances, cost and flexibility However in software level implementation of this algorithm leads to high energy consumption of the system, increase in complex algorithm, computational overhead and long execution time. Successful studies have been made for implement of cryptography algorithm to reduce the before said cons. The computational speed can be increased and power consumption can be reduced by increasing the number of core in single chip. High impact computing problem on more capable hardware is emergence realization for many-core system1. The two basic scheme of encryption are one-key and two-key ciphers. If the encryption of the plain text and decryption of the corresponding cipher text are performed using same key then it is one-key cipher in contrast if it is performed with different key then it is called is two-key cipher. Further one-key cipher is divided in to block cipher and stream cipher based on their process of computing the cipher text. The process of dividing the plain text in to fixed length block and then performing the encryption on each block to produce cipher text using same key is called as Block cipher and in relation to this if short string of key is used to generate the long sequence of bits and it is added bitwise modulo 2 to the plain text to produce the cipher text means then it will become Stream cipher.

2. Cryptographic Algorithm: Lightweight RC4 Algorithm

RC4 is a, variable key size and stream cipher with byte-oriented operations4,5. Algorithm is based on the use of a random permutation. The analysis shows that the period of the cipher is overwhelmingly likely to be greater than 10100 Eight to sixteen machine operations are required per output byte and cipher can be expected to run very quickly in software.RC4 was kept as a trade secret by RSA Security. The RC4 algorithm is remarkably simple and quite easy to explain, shown in figure 3. A variable-length key from 1 to 256 bytes (8 to 2048 bits) is used to initialize a 256-byte state vector S, with elements S[0], S[1], …, S[255]. At all times, S contains a permutation of all 8-bit numbers from 0 through 255. For encryption and decryption byte k (figure 1) is generated from S by selecting one of the 255 entries in a systematic fashion. Each value of k is generated the entries in S are once again permuted.

1) Initialization of S

To begin, the entries of S are set equal to the values from 0 to 255 in ascending order; S[0] = 0, S[1] = 1, …,S[255] = 255. The temporary vector T also created. The lengthof the key K is 256 bytes, then K is transferred to T[3].Otherwise, a key length of keylen bytes, the first keylen elements of T copied from K and then K is repeated as many times as necessary to fill out T. Because the only operation of S is a swap and the only effect is a permutation. S is still contains all the numbers from 0 to 255.

2) Stream Generation

The input key is no longer used when the S vector is initialized. Stream generation involves starting with S[0] and going through to S[255] and each S[i], swapping S[i] with another byte in S. according to the scheme dictated by the current configuration of S. After the S[255] is reached then the process continues starting over again at S[0]:

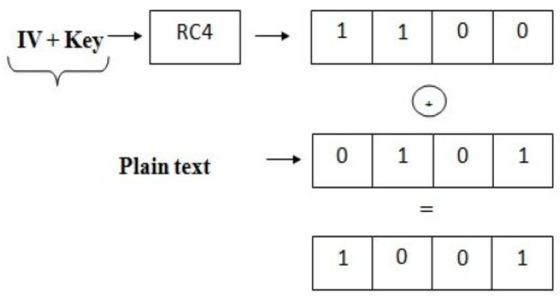


Figure 1: RC4 Algorithm

The HIGHT algorithm consists of 32-rounds with initial and final transformations before the first and after the last rounds respectively. The plaintext P and cipher text C are split into eight 8-bit blocks P7, … , P0 and C7, … , C0 and the original key K into sixteen 8-bit blocks K15 , … , K0.

3. Objectives

- Develop the cryptography modules compatible with APB bus protocol in Verilog
- Use those modules as peripherals to a SoC processor
- Incorporate techniques to reduce the cycles required for completing the process

4. Methodology

The project will be carried out in the following stages:

Further literature survey has to carried out to extract more ideas A design plan should be created with an application RTL Design of the proposed design should made RTL verification should be done to make sure the design is working as decided First level FPGA implementation should be done. This will complete one full cycle. Next, speed, area, power and other parameters should be observed and tabulated.Comparison of the parameters should be done. A demonstration should be and this can be done using the application decided Thesis/dissertation should be written.

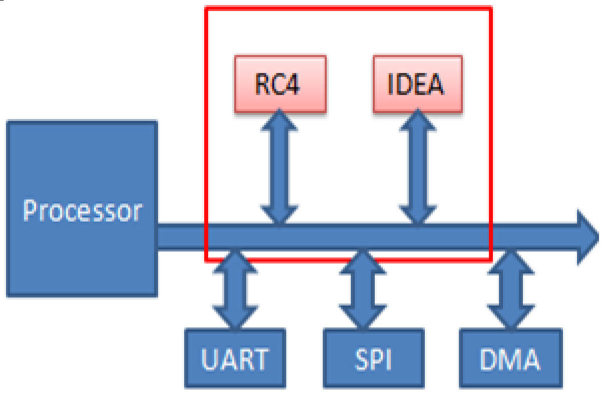


Figure 2: Typical SoC Implementation

What's new?

1) Advantage of SoC Implementation:

The added advantage of SOC implementation is that applicability on high volume production circuit, modification and reprogramming of the design can be done at lower unit cost and power consumption is low. In this project, we will explore various strategies to improvise the algorithms mentioned. Our objective is to reduce the number of software cycles which was previously implemented in the base paper. However, we will propose a new technique on the highly efficient Xilinx FPGAs. This will be based on the highly efficient AMBA protocols suite's APB bus.

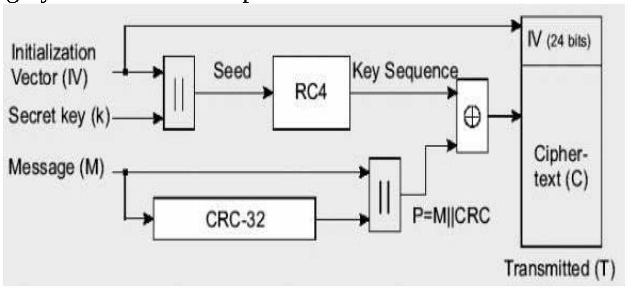


Figure 2: RC4 Implementation

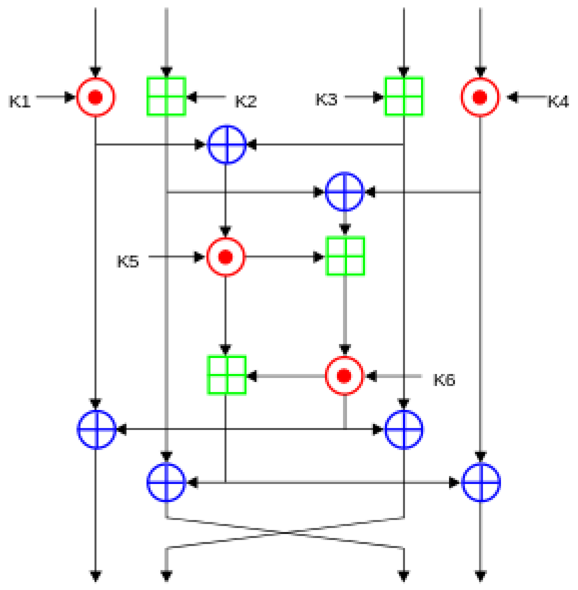


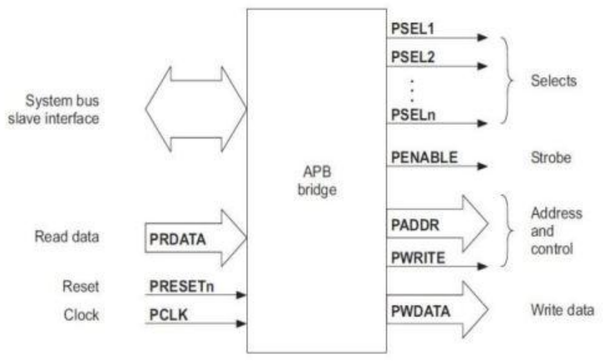
Figure 3: International Data Encryption Algorithm (IDEA)

2) AMBA Advanced Peripheral Bus

The Advanced Peripheral Bus (APB) is part of the Advanced Microcontroller Bus Architecture (AMBA) hierarchy of buses and is optimized for minimal power consumption and reduced interface complexity. The AMBA APB should be used to interface to any peripherals which are low bandwidth and do not require the high performance of a pipelined bus interface.

Components of APB

APB bridge: The APB bridge is the only bus master on the AMBA APB. In addition, the APB bridge is also a slave on the higher-level system bus (for example AHB). Figure below shows the APB signal interface of an APB

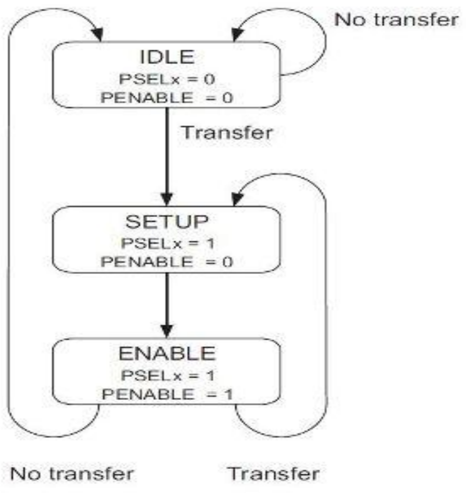


- Drives the APB data onto the system bus for a read transfer.
- Generates a timing strobe, PENABLE, for the transfer.

1. APB slave: APB slaves have a simple, yet flexible, interface. The select signal PSELx, the address PADDR and the write signal PWRITE are combined to determine which register should be updated by the write operation. For read transfers the data is driven on to the data bus when PWRITE is LOW and both PSELx and PENABLE are HIGH. While PADDR is used to determine which register should be read.

3) APB Operation

Transfer on APB bus takes place by transitions on a 3-state state machine. At any point, the APB bus is in one of the 3 states. The state diagram, shown in Figure below, can be used to represent the activity of the peripheral bus:

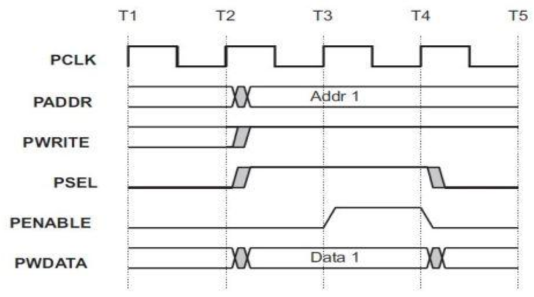


Operation of the state machine is through the three states described below:

1. **Idle :** The default state for the peripheral bus.
2. **Setup** : When a transfer is required the bus moves into the SETUP state, where the appropriate select signal, PSELx, is asserted. The bus only remains in the SETUP state for one clock cycle and will always move to the ENABLE state on the next rising edge of the clock.
3. **Enable** : In the ENABLE state the enable signal, PENABLE is asserted. The address, write and select signals all remain stable during the transition from the SETUP to ENABLE state. The ENABLE state also only lasts for a single clock cycle and after this state the bus will return to the IDLE state if no further transfers are required. Alternatively, if another transfer is to follow then the bus will move directly to the SETUP state.

Write transfer

The basic write transfer is shown in Figure below:



The write transfer starts with the address, write data, write signal and select signal all changing after the rising edge of the clock. The first clock cycle of the transfer is called the SETUP cycle. After the following clock edge the enable signal PENABLE is asserted, and this indicates that the ENABLE cycle is taking place. The address, data and control signals all remain valid throughout the ENABLE cycle. The transfer completes at the end of this cycle. The enable signal, PENABLE, will be deasserted at the end of the transfer. The select signal will also go LOW, unless the transfer is to be immediately followed by another transfer to the same peripheral. In order to reduce power consumption the address signal and the write signal will not change after a transfer until the next access occurs.

1) Applications

- Telecom
- Military
- Wireless Senson Networks
- Internet-of-Things

2) Expected Results

Successful implementation of RC4 and IDEA on SoC platform with improved performance parameters

5. Conclusions

To reduce the number of software cycles which was previously implemented in the base paper. However, we will propose a new technique on the highly efficient Xilinx FPGAs. This will be based on the highly efficient AMBA protocols suite's APB bus.

References

[1] Diamos, Gregory and Sudhakar Yalamanchili, An Execution Model and Runtime For Heterogeneous Many-Core Systems-2008
[2] Yalla Panasayya and Kaps J., Lightweight cryptography for FPGAs, International Conference on Reconfigurable Computing and FPGAs (2009)
[3] Poschmann Axel, Amir Moradi, Khoongming Khoo, Chu-Wee Lim, Huaxiong Wang, and San Ling, Side-channel resistant crypto(2011)

[4] Paul Rourab, et al., A simple 1-byte 1-clock RC4 design and its efficient implementation in FPGA coprocessor for secured ethernet communication,(2012)
[5] Sasidharan, Sapna, and Deepu Sleeba Philip. "A fast partial image encryption scheme with wavelet transform and RC4, (2011)
[6] William Stallings, Cryptography – RC4 Algorithm, October (2011)
[7] Mousa, Allam, and Ahmad Hamad, Evaluation of the RC4 Algorithm for Data Encryption, IJCSA (2006)
[8] Fluhrer, Scott, Itsik Mantin, and Adi Shamir, Weaknesses in the key scheduling algorithm of RC4, Selected areas in cryptography, Springer Berlin Heidelberg, (2001)
[9] Saarinen M-JO, The BlueJay Ultra-Lightweight Hybrid Cryptosystem, IEEE Symposium on Security and Privacy Workshops (SPW), (2012)
[10] Yuan X-C., et al., Hybrid encryption and decryption technique using microfabricated diffractive optical elements.