POLISH MARITIME RESEARCH Special Issue 2017 S2 (94) 2017 Vol. 24; pp. 60-65
10.1515/pomr-2017-0065

RESEARCH ON THE MARITIME COMMUNICATION CRYPTOGRAPHIC CHIP'S COMPILER OPTIMIZATION

Li Sheng Ph.D
Department of Electrical Engineering,Zhengzhou institute of information science and technology, Zhengzhou, Henan, China
Railway Police College, Zhengzhou, Henan, China
Z.B.Dai Professor
Zhengzhou institute of information science and technology, Zhengzhou, Henan, China

ABSTRACT

In the process of ocean development, the technology for maritime communication system is a hot research field, of which information security is vital for the normal operation of the whole system, and that is also one of the difficulties in the research of maritime communication system. In this paper, a kind of maritime communication cryptographic SOC(system on chip) is introduced, and its compiler framework is put forward through analysis of working mode and problems faced by compiler front end. Then, a loop unrolling factor calculating algorithm based on queue theory, named UFBOQ (unrolling factor based on queue), is proposed to make parallel optimization in the compiler frontend with consideration of the instruction memory capacity limit. Finally, the scalar replacement method is used to optimize unrolled code to solve the memory access latency on the parallel computing efficiency, for continuous data storage characteristics of cryptographic algorithm. The UFBOQ algorithm and scalar replacement prove effective and appropriate, of which the effect achieves the linear speedup.

Keywords: Maritime Communication Encryption System, Cryptographic SOC, Compiler Frontend, Loop Unrolling

INTRODUCTION

As the attractiveness of marine resources and the awareness of marine sovereignty, countries in the world are continuously strengthening the development and utilization of sear, which has led to the development of marine environmental monitoring, deep ocean voyage and marine military. Any ocean-related technology is inseparable from the data communication, in this area a wealth of researches has been implemented, such as underwater local communication network[1][2], underwater laser communication system[3], watercraft communication and navigation system[4].When it comes to the field of data communication research, the security transmission of data is one of the most important research point[5], which is to protect the data transmission process from being falsified or interception, and the end of information security between transmitting terminals and receiving terminals is the basic requirements, especially in marine military communications. However, this is often neglected in the construction of maritime communication system, because the maritime communication system research is more complex, which is characteristic of wireless and remote transmission with mutable natural environment, and it is always a cross of different research fields, so most of the researches are focused on the stability and feasibility of the maritime communication system, ignoring the information security in the original design. Thus, it is necessary for us to study it.

Encryption is the main way to realize information security, and the maritime communication encryption system often consists of encryption devices and transmission devices, which is shown in Fig. 1.

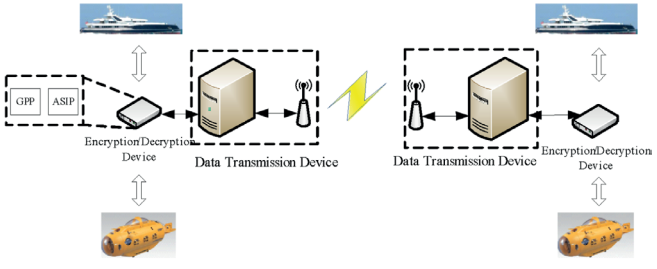

Fig. 1. Structure of maritime communication encryption system

With the development of integrated circuit, the encryption device is able to be implemented by SOC technology, which integrates GPP(general purpose processor) with ASIC(application specific integrated circuit), FPGA(field programmable gate array), ASIP(application specific instruction processor) into one chip[6-8]. As for the diversity of encryption algorithm in the maritime communication, encryption device should be designed with GPP and ASIP of SOC. The GPP takes the function of tasks control and the ASIP is attributed to computationally intensive operation such as encryption/decryption under the specific instructions. This heterogeneous system indicates that programs coded in the serial mode are not able to be executed, so a compiler structure for this heterogeneous system is needed to partition the source programs and map the codes to GPP and ASIP correspondingly.

In this paper, we introduce framework and workflow of one kind of maritime communication cryptographic SOC, then a compilation framework is designed for that, and a loop unrolling factor calculating algorithm based on queue theory is proposed according to the encryption algorithm features, which is exploited in the compiler front end to enhance parallel. The scalar replacement method is utilized to optimize the unrolled encryption codes to tackle the latency of memory access.

THE MARITIME COMMUNICATION CRYPTOGRAPHIC SOC

THE FRAMEWORK

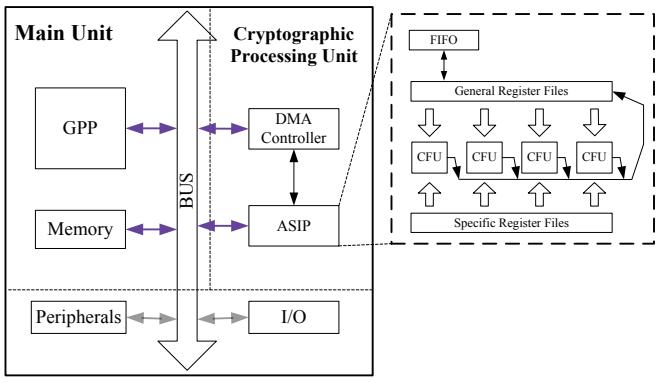

Fig. 2 The framework of maritime communication cryptographic SOC

The framework of maritime communication cryptographic SOC is in Fig. 2, it includes the main unit, the cryptographic processing unit and peripherals. The main unit is mainly composed of GPP and Memory, and the cryptographic processing unit is mainly composed of the cryptographic ASIP with the DMA(direct memory access) controller. The main unit, the cryptographic processing unit and peripherals send/receive data by data bus, and the cryptographic ASIP also can transmit data through DMA.

The cryptographic ASIP is comprised of general register files, specific register files and cryptographic CFUs(custom function unit) which is designed according to the encryption algorithm's computing features, and each cryptographic CFU operate after configurations in order to implement various encryption algorithms. Multiple CFUs operate together to do encryption in different bits length[9].

OPERATION PROCESS

The framework of the cryptographic SOC is a master-slave structure, which requires two parts of the program, that is, the executable program in the main unit and the executable program of the cryptographic processing unit, and the cryptographic processing unit needs conFig. the internal function unit, then perform the cryptographic algorithm. The operation process is illustrated in Fig. 3 including four steps:

1) The GPP reads the memory;
2) The GPP initiates the DMA, sending the configuration information, the operation instruction and the operation data are sent into the input FIFO of the cryptographic ASIP;
3) The cryptographic ASIP disassemble the data in the FIFO, extracting the configuration information, the operation instruction and the plaintext data, then conFig.s cryptographic CFU and executes the operation instruction;
4) The ciphertext data are sent into the output FIFO and written back to the memory by the DMA controller via the bus.

As can be seen from the framework and workflow of the processor, there are two control flows in the whole process, so the framework of the SOC and the situation about control flow must taken into account in the design of the compiler.

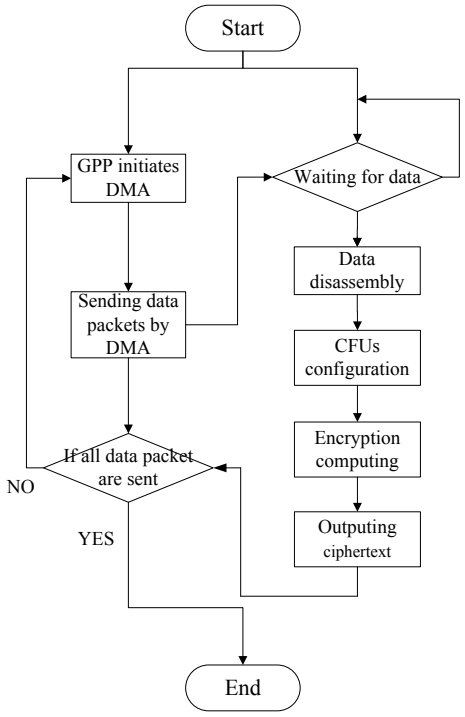

Fig. 3. The workflow of maritime communication cryptographic SOC

THE FRAMEWORK OF COMPILER FOR THE MARITIME COMMUNICATION CRYPTOGRAPHIC SOC

Because most encryption programs are coded 由 in serial mode based on the traditional processor framework, these programs are not able to be implemented in the heterogeneous framework SOC directly, and programming codes in parallel mode is difficult for software engineer. Thus a new compilation framework has to be designed to parallel the serial programs.

The compiler consists of the frontend and backend, and the frontend handles works which are not connected with the target machine, such as lexical analysis, syntax analysis; the backend is mainly responsible for handling the intermediate codes into executable code of the target machine According to the framework and the control flow analysis of the maritime communication cryptographic SOC, the overall framework of the compiler for it is shown in Fig. 4.

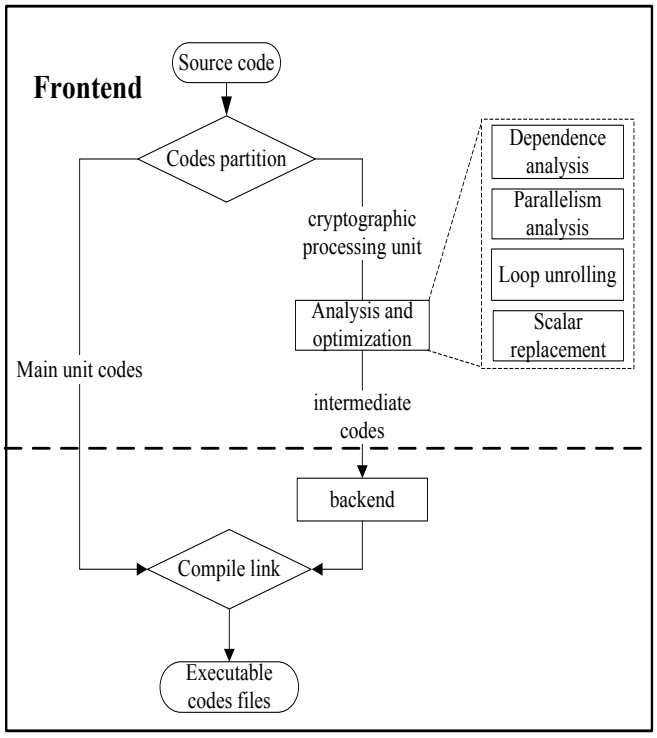

Fig. 4. The framework of compiler

The frontend of the compiler separate the program source code into the main processing unit codes and the cryptographic processing unit codes, which realize the program source code distributed into different control flows, and then maps and links codes by the backend. The main unit processes the entire system function control, so its detailed analysis is not discussed in this paper. The code executed by the cryptographic processing unit needs to be analyzed and optimized by the frontend of the compiler to explore the parallelism of the code, which is important for achieving the purpose of making full use of the multi CFUs in the cryptographic processing unit and improving the efficiency of the operation. After analysis and optimization, The cryptographic processing unit codes are generated into intermediate code, and then processed by the backend to generate configuration information and operation instructions, finally the compile and link procedure generate executable codes files.

COMPILER FRONTEND OPTIMIZATION

ANALYSIS OF ENCRYPTION ALGORITHM FEATURES AND THE FRONTEND TASK

In terms of the structural and application characteristics of the encryption algorithm, the obvious feature is iterative operations in both symmetric cipher and asymmetric cipher, which is to achieve the purpose of Confusion and diffusion. Especially in the block cipher, the block length and structure is closely related to the iteration. This feature is based on the idea of Shannon product password, so the iterative feature is one of the important features of the encryption algorithm application. Thus, there are the following problems in the compiler frontend design:

1) There is not a widely applicable parallel programming model, so the compiler frontend need to search the serial codes parallelism. In terms of encryption algorithms codes, there exist a large number of loops and the number of iterations is big, which causes the complexity and difficulties for loops optimization of the compiler frontend. In order to search the parallelism of the serial program codes, it is necessary to unroll loops, which can effectively improve the instruction level parallelism of the program, the locality of the register and the hierarchical structure of the storage structure[10][11]. However, the degree of loop unrolling is significant. The fully loop unrolling could make instruction parallelism to the maximum extent, but it will increase the pressure of the instruction RAM(random access memory), and the parallelism of the program is determined by the proportion of the serial part of the code in the program according to Amdahl's law, which means insufficient loop unrolling will result in a decrease in overall program performance. All considerations above indicate that the determination of loop unrolling factor is important and need overall consideration about the target machine.
2) As for encryption algorithm application codes, there is amount of memory access instructions in the loop, which will consume more clock cycles than other instructions and cause the computing unit, such as the CFUs, to use some clock cycles waiting for data, interrupting calculation temporarily and reducing the parallelism of the whole program. So, the frontend is required to tackle the problems.

THE LOOP UNROLLING FACTOR CALCULATING ALGORITHM BASED ON QUEUE THEORY

As for the key point of loop unrolling factor, if programs are coded in serial mode, the instructions in the loop will be executed line by line, which could be summarized as below:

i. every instructions of the codes will hold the computing function unit when it is executed;
ii. different function instruction will consume certain amount of clock cycle when it is executed;
iii. the unexecuted are waiting in line for executing

The three steps could be described form of formal specification, define:

$aei(t) \overset{DEF}{=}$ the amount of executed instructions in time segment $(0,t)$;

$sei(t) \overset{DEF}{=}$ the speed of executing instructions in time segment $(0,t)$;

$T \overset{DEF}{=}$ the time that all instructions are executed;

$aet(t) \overset{DEF}{=}$ the average executing time for each instructions;

$\overline{N(t)} \overset{DEF}{=}$ the amount of all the instructions(including the instructions being executed and the other instructions waiting in line);

So formula (1) and (2) are as below:

$$sei(t) \overset{DEF}{=} \frac{aei(t)}{t} \tag{1}$$

$$T = \sum_{i=0}^{t} aei(i) \bullet d(i) \tag{2}$$

$d(i)$ is the $aei(t)$ instructions' time consuming at the timing i, formula (3) and (4) are as below:

$$aet(t) = \frac{T}{aei(t)} \tag{3}$$

$$\overline{N(t)} = \frac{T}{t} \tag{4}$$

It can be concluded: $\overline{N(t)} = sei(t) \bullet aet(t)$, and on the situation that the amount of instructions is limited, there is $sei = \lim_{t \to \infty} sei(t)$ and $aet = \lim_{t \to \infty} aet(t)$, $t \in [0, +\infty]$, so formula (5) can be gotten:

$$\overline{N} = sei \bullet aet \tag{5}$$

formula (5) indicates that the amount of instructions which could be executed in the processor is equal to product of the speed of executing instructions and the average executing time for each instructions, if the program is able to provide $\overline{N}$ parallelism, the processor could achieve the best efficiency[15]. For the consideration that the capacity of the instruction memory is limited and related with the unrolling factor, the loop expansion factor must be constrained by the of instructions' number and volume.

Based on the analysis above, the UFBOQ (unrolling factor based on queue) is proposed as formula(6), and the following notations are adopted:

NL numbers of instruction in the loop
NI numbers of iteration
N the instruction throughput
M the delay/average executing time
R the restriction of total instructions number define by the programmer
UF the unrolling factor

$$UF = Min(NI, R/NL, N*M) \tag{6}$$

If $UF = NI$, the loop is fully unroll, and if $UF = N*M$, $N*M \neq NI$, it means the loop is unrolled according to the degree of parallelism.

After the loop unrolling, the original loop boundary is broken which could enlarge the program basic block. The same operation code in a loop can be vectorized or parallelized to realize the parallel execution of the small bit width operation to achieve the large bit width operation effect. For example, many block cipher operations have a non-linear S-box operation, which determines the security of the cryptographic algorithm. It can be regarded as a mapping of fixed-length data to another fixed-length data, which is often less than the packet length. Because there is no data dependency between the S-box operations within the packet length range, it can be executed in parallel after the loop unrolling to improve the operation bit width.

SCALAR REPLACEMENT

There is usually a certain number of array reading and writing operations in the encryption algorithm codes, and the cryptographic ASIP is the slave unit in the system, which means the ASIP need to get plaintext data from the bus. As for the ASIP, the bus can be regarded as an external storage device, so the memory access instructions can cause communication cost to reduce the computational efficiency. And there is data reuse in the round operation, so the best way is put the last round operation data into the register, but data reading and writing operation code is usually compiled as access to the external memory. For above considerations, it is necessary to optimize the memory access codes in unrolled loop.

The scalar replacement technology is exploited to tackle the problem. After the loop unrolling, the array reading

instructions are moved out then put in front of the loop body, and the array writing back instructions are moved out then put after the loop body. Finally, array elements are replaced by scalars. This scalar replacement technology enables the compiler to allocate registers to data which need to be reused, instead of reading/writing data to memory by bus repeatly, and this optimization achieves the separation of calculation process and data reading/writing process. After the scalar replacement optimization, when the ASIP read the external plaintext data from the bus, the data are able to be input to the buffer, and then input to the general register files. When the encryption are finished and ciphertext data are to be written back to the memory, the data are written to the output buffer, and then to the external memory. In this mode, the plaintext data and the encryption data are able to be transferred to correspond FIFO by DMA mode. The codes in Fig. 5 is the example about S-box operation.

```
for(c=0; c<N; c++)
{
    state[c] = Sbox(state[c]);
}
```

(a) S-box operation codes

```
for(c=0; c<N; c=c+UFL)
{
  Parallel do
    state[c] = Sbox(state[c]);
    state[c+1] = Sbox(state[c+1]);
                    ⋮
    state[c+UFL] = Sbox(state[c+UFL]);
   Parallel
}
```

(b) Codes after loop unrolling

```
X0 = state[0]
      ⋮
XN = state[N]
for(c=0; c<N; c=c+UFL)
{
  Parallel do
      XC = Sbox(XC);
      XC+1 = Sbox(XC);
                ⋮
      XC+UFL = Sbox(XC+UFL);
  Parallel
}
state[0] = X0;
      ⋮
state[N] = XN ;
```

(c) Codes after scalar replacement

Fig. 5. Optimization for S-box operation codes

In Fig.5(a), the original S-box operation codes in the loop contain once external data memory reading, once table handling and once data writing to external memory, and the loop body will repeat N times.

After the loop unrolling in Fig. 4 (b), the number of table handling operation is still N times, but can be executed in parallel by UF threads in N/UF iterations which is less time consuming. In Fig.4 (c), the 2N times the external storage device reading/writing operations can be done to the DMA controller, and the ASIP only runs table handling operation, in this way it solves the latency of memory access to external storage device time consumption which reduces parallel computing effect. Because the plaintext data and the ciphertext data of the cryptographic operation are stored serially in the memory, it is very suitable for the DMA operation. In the optimization of scalar replacement, the volume of the buffer unit and register need to be taken in consideration to avoid data overflow.

EXPERIMENTS AND PERFORMANCE

The experiment is implemented based on the SOC which is manufactured in 65nm, and the ASIP's throughput is 1 instruction/clock, the latency is 3 clock for each instruction. The encryption algorithm DES、AES-128、IDEA、SMS4 are utilized in the experiment, and the performance comparison between non-optimized compilation and optimized compilation is shown in table 1.

Tab. 1. Performance comparison

Encryption algorithm	Clocks under non-optimized compilation	Clocks under optimized compilation	speedup
AES-128	268	105	2.55
DES	695	241	2.88
IDEA	168	62	2.7
SMS4	5632	2063	2.73

According to the experimental results, it can be seen that the UFBOQ algorithm and scalar replacement prove effective and appropriate, of which the effect nearly achieves the linear speedup.

CONCLUSIONS

In this paper, a kind of cryptographic SOC for maritime communication encryption system is introduced. For solving the problems of the heterogeneous SOC, a compiler framework for it is proposed. What is more, for the problem of parallelism and latency of memory access operation, the optimization of UFBOQ algorithm and scalar replacement is presented based on the compiler framework and the encryption algorithm codes feature. The optimization searches the parallelism of the serial codes and tackle the latency caused by memory access.

The experiment is implemented with 4 classic encryption algorithms, and the results verifies the validity of the frontend optimization function of the compilation system.

BIBLIOGRAPHY

1. RWL Coutinho, A Boukerche， LFM Vieira, AAF Loureiro. A novel void node recovery paradigm for long-term underwater sensor networks. Ad Hoc Networks, Vol. 34, No.C , pp.144-156, 2015.

2. MR Bharamagoudra, SKS Manvi. Deployment Scheme for Enhancing Coverage and Connectivity in Underwater Acoustic Sensor Networks. Wireless Personal Communications, Vol.89, No.4, pp.1265-1293,2016.

3. J Xu, A Lin, X Yu et al. Underwater Laser Communication Using an OFDM-Modulated 520-nm Laser Diode. IEEE Photonics Technology Letters,28 (20),pp.2133 - 2136, 2016.

4. C Specht. Accuracy and coverage of the modernized Polish Maritime differential GPS system.Advances in Space Research, Vol.47,No.2, pp.221-228,2011 .

5. G Dini, AL Duca. A Secure Communication Suite for Underwater Acoustic Sensor Networks.Sensors, Vol.12,No.11,pp. 15133-15158,2012.

6. K. akdemir, M. Dixon, et al. Breakthrough AES performance with Intel AES new Instructions. Whilte paper, June, 2010

7. J. burk, J. Mcdonald, et al. Architecture support for fast symmetric-key cryptography. Acm Sigplan Notices, Vol.35, No.11:178-189, 2000.

8. Wang Y, Ha Y. FPGA based 40.9-Gbit/s Masked AES with Area optimization for storage area network. Circuits & Systems II Express Briefs IEEE Tranaction on , Vol. 60, No.1, pp.36-40, 2013.

9. LI Wei, ZENG xiaoyang, NAN longmei. A Reconfigurable Block Cryptographic Processor Based on VLIW Architecture[J]. China Communication, Vol.13, No.1, pp. 91-98, 2016

10. Gao Fei，Li hongyan, Zhang Yongfu. Research on cipher coprocessor instruction level parallelism compiler, Application research of computers, Vol. 27, No.5, pp. 1633-1637,2010.

11. DAVID F. BACON, SUSAN L. GRAHAM, AND OLIVER J. SHARP. Compiler transformations for high-performance computing, Acm Computing Surveys, Vol. 26, No.4, pp. 345-420, 1994.

CONTACT WITH AUTHOR

Li Sheng

e-mail: linirvana@126.com
Zhengzhou Institute of Information
Science and Technology
Zhengzhou
China