Proceedings of the International MultiConference of Engineers and Computer Scientists 2015 Vol I,
IMECS 2015, March 18 - 20, 2015, Hong Kong

Low-cost Bit Permutation Circuit with Concise Configuration Rule

Tianyong Ao, Zhangqing He, Kui Dai, Xuecheng Zou

***Abstract*—Bit-level permutation is widely used in cryptography systems. However, it is not well-supported in simple and low-cost way. Here, a low-cost construction of n-bit permutation circuit capability of $n!$ permutations is proposed. This construction just needs one n-to-1 multiplexer and $(2n + n * \lceil log_2 n \rceil)$ bits registers. Its configuration rule is very concise. Users only straightforward input the numbers that indicate which bit of source data will go to i-th bit of destination data. The Synopsys' DC synthesis results show that approximate 31% of the area can be saved by this architecture compared with Benes networks of identical size. This design provides a new perspective on implementation of bit permutation and may be helpful for the design of secure resource-constrained systems.**

***Index Terms*—bit permutation, ASIC, low-cost,concise configuration rule,block cipher.**

I. Introduction

BIT-level permutations are important primitives in cryptography systems to achieve security. They are widely used in block ciphers such as DES, TWOFISH, Sperent and Present [1], and even in fully homomorphic encryption [2]. Different ciphers usually use different bit permutations. Therefore, the designs supporting arbitrary bit permutations with low-cost and convenient operations are required for implementing various ciphers in cryptography systems.

A bit permutation only moves the positions of data's bits without affecting their values. For a bit permutation function which converts data S into data D, we have:

$$D[i] = S[\pi(i)] \tag{1}$$

where $\pi(i)$ is a permutation over the index space and $D[i]$ indicating the value of i-th bit of D, $S[\pi(i)]$ denoting the value of $\pi(i)$-th bit of S. S and D are called source data and destination data, respectively.

Both software and hardware can be used to implement bit permutations. Almost all the hardware designs are based on the principle that a data path between the i-th bit of D and the $\pi(i)$-th bit of S should be established. To support arbitrary permutation, the networks with the capability of $n!$-permutation such as Waksman and Benes networks [3] are usually used. Those designs have the advantage of high performance, however, when n is the larger, the cost is higher and the configuration rules are more complex. The main principle with software is based on choosing the aimed bits from source data, moving into the aimed positions and then placing them in the destination data. A n-bit permutation usually needs $4 * n$ instructions by repeating Load, Mask, Shift and Or operations. It is low performance, especially in the processors without barrel shift instructions. To compute arbitrary bit permutations efficiently on general purpose microprocessors, several fast bit permutation instructions have been proposed based on permutation networks [4] [5] [6].

Previous designs are focused on the performance and their configuration rules are usually complex. However, they may be difficult to meet the cost of resource-constrained systems. Here, we present an arbitrary n-bit permutation circuit with very concise configuration rules and low cost. Its principle is based on choosing and cyclic shift operations. This design is inspirited by the fact that cyclic shift registers are widely used in security systems such as hash functions and key scheduling of Present cipher.

II. Proposed architecture

A n-bit sequence can be considered as an array. Each element of the array is structure data which has two fields. One is address field and the other is value field. Assuming the elements from right to left are numbered $0, 1, \cdots$,n-1. Considering the data D as such an array, where the address field of the i-th element is used to store the number $\pi(i)$, and its value field is used to receive the corresponding bit from the source data. The source data S also can be seen as the such array with variable value field and fixed address field, where the value of the address field of the i-th element is the number i. Then, the following process can implement the bit permutation function.

```
$For\ i = 0 \quad to \quad n-1$
    $D[i] = S[\pi(i)]$
$EndFor;$
```

We present a bit permutation circuit which can perform any one of $n!$ permutations for n-bit data, as the structure is shown in Fig.1. It only consists of n-bit source data registers, n-bit destination data registers, n m-bit address registers and one n-to-1 multiplexer. The i-th element of the array consists of the i-th data register, denoted Db_i, and the i-th address register, denoted A_i. The destination data registers and address registers can work in a serial shift way. The

This work is supported by the National Natural Science Foundation of China (No.61376031) and Natural Science Foundation of Hubei Province of China (No.ZRZ0051).

Tianyong Ao is with the School of Optical and Electronic Information, Huazhong University of Science and Technology, Wuhan, China and is also with the School of Physics and Electronics, Henan University, Kaifeng, China (e-mail: tyaohust@gmail.com).

Zhangqing He is with the School of Optical and Electronic Information, Huazhong University of Science and Technology and is also with Hubei Collaborative Innovation Center for High-efficient Utilization of Solar Energy, Hubei University of Technology, Wuhan, China (e-mail: ivan_hee@126.com)*.

Kui Dai is with the School of Optical and Electronic Information, Huazhong University of Science and Technology, Wuhan, China (e-mail:daikui@mail.hust.edu.cn).

Xuecheng Zou is with the School of Optical and Electronic Information, Huazhong University of Science and Technology, Wuhan, China (e-mail:estxczou@gmail.com).

All the authors are also with Innovation Center for MicroNanoelectronics and Integrated System, Wuhan, China.

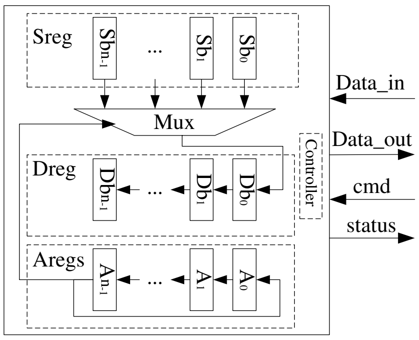

Fig. 1. n-bit permutation circuit

inputs of the multiplexer are from the bits of source data registers, and the channel control bits come from register A_{n-1}. There are four operation steps to do an arbitrary n-bit permutation.

Step 1. Configure permutation rules. It is very simple to configure permutation rules. It only needs to write the number $\pi(i)$ to the address register A_i according to the permutation rules, indicating that which bit of the source data will go to the i-th bit of destination data.

Step 2. Write source data to the source registers.

Step 3. cyclic shift operations. In each clock cycle, the address registers do one step cyclic shift to the left. Each bit value in destination register is shifted one position to the left. The value of Db_0 is updated by the output of Mux. Here, n clock cycles are needed.

Step 4. Read the result. After step 3, the data in destination register is just permutation result. In addition, the values of address registers are the same with what were configured in the first step.

The bit permutation rules of a block cipher usually are fit, therefore the first step for configuring permutation rules is only needed to do one time in the initial state for this scenario.

This construction is easily extended to sub-block permutations when the data width of Sb_i, Db_i and the multiplexer are a number greater than 1. Sub-block permutations are widely used in generic Feistel ciphers, such as SM4. For sub-block permutation, the number of address registers is only the number of sub-blocks.

III. Implement results and Comparisons

We have implemented several bit permutation hardware designs with n in different sizes in Verilog HDL, where n is equal to 32, 64, or 128, since those sizes are usually used in block ciphers. To implement arbitrary n-bit permutation with this method, only $(n*\lceil log_2 n\rceil)$ bits registers for configuration information, $2n$ bits registers for source and destination data, and one n-to-1 multiplexer are required, where $\lceil x\rceil$ denoted the smallest integer which is greater than or equal to x. To compare the cost, three Benes networks of the same size are also implemented in RTL , since Benes network is one of the most famous nonblocking networks which can perform any of the $n!$ permutations of n bits. For n-bit Benes networks, there are $(n*\lceil log_2 n\rceil - n/2)$ bit registers for its configuration information, and the identical number 2-2 switches which either swap or pass through the bits based on whether their

TABLE I
Implementation results and Comparisons

n	Area (GEs)		
	This	Benes	GRP [6]
32	1372	1983	68.6K/19.7K
64	3552	5165	
128	7756	11275	

control bits are 1 or 0. The size of Benes networks should be the power of 2, but that of this circuit can be any number.

The designs were synthesized by Synopsys' DC with the TSMC 0.13um CMOS standard cell library, and the results are shown in table I. The saved areas by this method are above 31% and 60% compared with Benes networks with the identical size and the special bit permutation instruction GRP [6],respectively. The design of [6] should be embedded as a function units in a CPU. It is good for speedup bit permutation in CPU, but not suitable for the low cost implementations. Approximately 4*n instructions ROM spaces have to be reserved to store program codes if n-bit arbitrary permutations will be implemented by a CPU.The area of those ROM may be greater than that of this method.

The total cycles for n-bit arbitrary permutation is the sum of writing and reading data time (T_{data}), cyclic shift time (T_{cyc}), configuration time (T_{cfg}) in this designs. T_{cyc}, which is the cycles spent on cyclic shift state, is always equal to n. T_{data} and T_{cfg} are the cycles spent on writing/reading data and configuration information, respectively. They depend on the data width. If the data width is m, T_{data} is equal to $\lceil n/m\rceil$, and T_{cfg} is equal to $\lceil \frac{n}{m/\lceil log_2 n\rceil}\rceil$. The throughput of 128-bit permutation circuit is equal to 421 Mbps with the maximum frequency of 500MHz. Since the data width of each address register is less than eight, eight address registers can be written with 64 bits data in parallel. When the permutation rule is fixed and the permutation is performed many times, for example, in SPN block ciphers, only one time is asked to set the configuration data to the address registers in the initial stage. In this case, T_{cfg} will be equal to zero in the following permutations.

This design has been used in our lightweight customized encryption system which makes users customize a block cipher quickly and easily.

IV. Conclusions

A bit permutation circuit which can perform any one of the $n!$ permutations for n bit data is proposed in this letter. The proposed bit permutation circuit has the advantages of very concise configuration rules and low cost. It does not need any routing algorithm and its size n is no longer limited to the power of 2 like Benes networks. This design method can attain a balance between performance and cost, and could give a chance that can share the shift registers with other modules. This design provides a new view of implementation of bit permutation and may be helpful for the design of secure resource-constrained systems.

References

[1] A. Bogdanov, L.R. Knudsen, G. Leander, C. Paar, A. Poschmann,M.J.B. Robshaw, Y. Seurin , and C. Vikkelsoe: PRESENT: An ultra-lightweight block cipher, *Cryptographic Hardware and Embedded Systems-CHES*, 2007, pp. 450-466.

[2] Gentry, Craig, Shai Halevi, and Nigel P. Smart,Fully homomorphic encryption with polylog overhead, *Advances in CryptologyCEURO-CRYPT*,2012. pp.465-482.

[3] R. A. Spanke and V. E. Benes, N-stage planar optical permutation network, *Applied Optics*,vol.26, pp.1226-1229,1987.

[4] R.B. Lee, Z. Shi, and X. Yang, Efficient permutation instructions for fast software cryptography, *Micro, IEEE*, vol.21, pp.56-69,2001.

[5] X. Yang, M.Vachharajani, and R. B. Lee, Fast subword permutation instructions based on butterfly network, *International Society for Optics and PhotonicsIn Electronic Imaging*, vol.3,pp. 80-86, 1999.

[6] Y. Hilewitz, Z. J. Shi and R. B. Lee, Comparing fast implementations of bit permutation instructions, *Conference Record of the Thirty-Eighth Asilomar Conference on Signals, Systems and Computers*, vol.2,pp. 1856-1863, 2004.