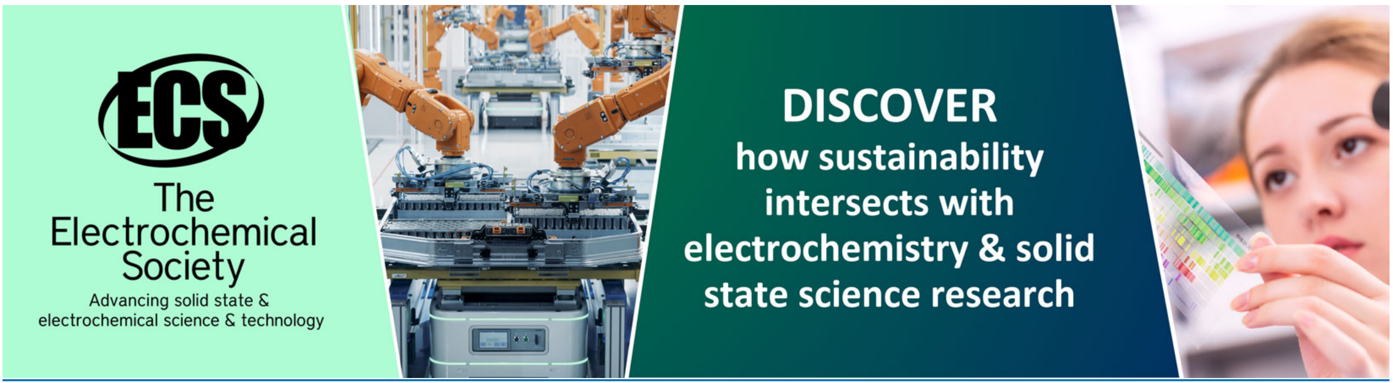
Research and Design of a Novel Reconfigurable Cyclic Shift Unit

To cite this article: Z X Chang *et al* 2018 *IOP Conf. Ser.: Mater. Sci. Eng.* **439** 042074


The Electrochemical Society
Advancing solid state & electrochemical science & technology
DISCOVER how sustainability intersects with electrochemistry & solid state science research

Research and Design of a Novel Reconfigurable Cyclic Shift Unit

Z X Chang, M F Chen, Y X Sun

College of Information and Communication, National University of Defense Technology, Wuhan, China

Corresponding author e-mail: changzhongxiang0@126.com

Abstract. In this paper, a high speed reconfigurable control bits generation algorithm for cyclic shift operations is proposed. The algorithm utilizes the characteristics of routing information when Inverse Butterfly multi level network supports cyclic shift. Moreover, it also integrates a kind of cyclic shift operations together, including cyclic shift and sub-word cyclic shift based on a bit width of 2^i(i = 1, 2, …). Based on this, the shifter is implemented, and the logic synthesis is realized in the SMIC 65nm process. The results show that when the proposed unit only supports cyclic shift operations, enhances the frequency by 3.6% ~ 8.6% and reduces the area by1.4% ~ 32%, compared with previously proposed solutions. When achieving a variety of cyclic shift operations, the frequency of our unit enhances the frequency by 5% and reduces the area by 6%.

1. Introduction

The shifter is a basic arithmetic component that is most commonly used in general-purpose processors. In recent years, with the diversification of general-purpose processor applications, conventional barrel shifters, logarithmic shifters [1], funnel shifters [2] and other implementations have been unable to meet cryptography [3], digital image processing, multimedia processing and so on. More and more parallel shift operations and more complex bit-level replacement operations [4] are required,, it is urgent to find a new implementation to meet the possible future complex bit-level replacement operations.

At present, a new type of shifter implementation based on interconnection networks has emerged, based on Inverse Butterfly (IButterfly) proposed by Hilewitz in Princeton Architecture Laboratory for Multimedia and Security and Ma Chao in University of Information Engineering. The implementation of the IButterfly network is not only capable of supporting traditional cyclic shift operations, but also supports functions such as parallel filtering and parallel insertion, and integrates them into a unified hardware architecture, providing new ideas of implementations for shifters..

However, the method of Hilewitz proposed that the routing information of the network be serially iterative [7]. The method of Ma Chao proposed that the network routing information utilizes the number of shifting steps to parallelize the initial routing information of the network [8]. As the network progression increases, the routing information generation time is too long, and the hardware resources are too large.

Therefore, this paper analyzes the characteristics of routing information when the network supports cyclic shift, and proposes an algorithm that can generate routing information at all levels at the same time. It cannot only supports the displacement types such as cyclic shift and short word cyclic shift, and can effectively reduce the routing information generation time and reduce resource consumption.

2. Related work

IButterfly is a dynamic multi-level interconnection network. By changing the switch state, it can realize the connection between different nodes, so that the network has the ability of reconstruction, thus completing the data rearrangement.

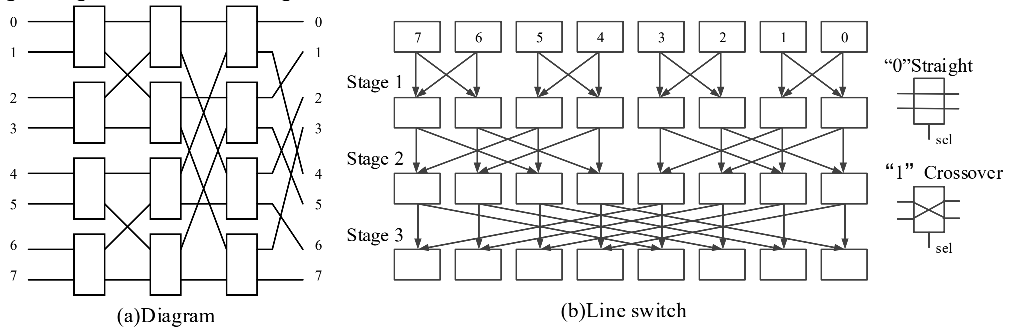


Figure1. Two representations of IButterfly

As shown in Fig. 1, this is an IButterfly dynamic multilevel interconnect network with an m=8-bit width. The network has a $\log_2 m$ level, Figure 1(a) is a diagram, and Figure 1(b) is a line switch. There are m/2 2-input crossbars at the first level, and the input of the crossbar switch is fixed. There are a total of $\log_2 m \times m/2$ crossbar switches. Each switch has two states of "crossover" and "straight-through", which can achieve cyclic shift, P sequence replacement, parallel insertion and other kinds of replacement operations [4].

Currently, Lee and Hilewitz of Princeton University and others studied the routing information generation algorithm of Inverse Butterfly/Butterfly supporting cyclic shifting, insertion, and extraction operations based on the characteristics of routing information, and presented complex permutation operations such as cyclic shift, parallel insertion, and parallel extraction, then design GRP, PEX / PEDP, ROT and other special instructions [5,6], which PEX and PDEP instructions to complete the data parallel extraction and parallel insertion operations. Intel integrated PEX and PDEP in the instruction set of the HASWELL processor which released in 2013 [10].

For cyclic shifting routing information, Hilewitz proposed a control information generation algorithm in the literature [7]. Traditional shift operations and complex bit replacements are implemented in a unified hardware unit. The level control information must be serially generated. Therefore, the degree of parallelism of the algorithm is low, not only the resource consumption is large, but also the delay is long. The complexity of the algorithm rises sharply with the number of network levels increases, and the hardware implementation becomes very difficult.

In the previous research, I also proposed a parallelized cyclic shift control information generation algorithm based on IButterfly network. This algorithm generates parallel control information by shifting the number of steps, effectively improving the cyclic shift processing efficiency [9]. However, the algorithm takes advantage of the uniqueness of different permutation routing information, simplifying the number of shifting steps and the truth table, so the hardware resource consumption is large, and it is only applicable to networks with smaller bit widths.

Dr. Ma Chao also proposed a parallelized cyclic shift control information generation algorithm based on this. The algorithm inverts and shifts the initial control information of the network by shifting the number of steps, and optimizes the process in terms of hardware implementation. The degree of parallelism and the efficiency of calculation are greatly improved, but the routing information is calculated. A large number of inverse shift operations need to be calculated, and the performance improvement is limited.

In this paper, a new type of cyclic shift control information generation algorithm is proposed by studying the relationship between the initial position encoding of data, the number of shift steps and the control information at each level, and it can support bidirectional cyclic shift, 2^i (i=1,2, ..., $\log_2 m$) is a bit-width short word cyclic shift operation. At the same time, it can greatly reduce the complexity of the algorithm and effectively improve the processing speed of the shift.

3. The features of IButterfly network implements cyclic shift routing information

According to the IButterfly network topology and interconnection function, the routing information required for each type of replacement is unique. However, the type of cyclic shift is very limited, and the required state set of routing information is also small. Therefore, this paper focuses on the characteristics of routing information needed for cyclic shift.

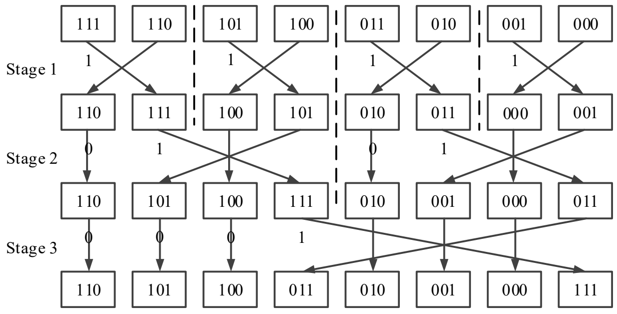


Figure 2.Left shift one step

As shown in Fig. 2, IButterfly networks supporting four 2-bit cyclic shift operations because the first level of the network can be seen as four independent 2-bit. IButterfly network supporting two 4-bit cyclic shift operations because the second level can be seen as two independent 4-bit. IButterfly network supporting a set of 8-bit cyclic shift operations because the entire network can be viewed as an independent 8-bit; 8-bit cyclic left shifting in one step, four independent 2-bit cycles must be left in the first stage, and two independent 4-bit cycles in the second stage. In each level, the interconnection function between each input and output is exactly the same, and the routing information corresponding to the same function is also the same. Therefore, it can be seen that the IButterfly network supports the characteristics of cyclic routing corresponding to routing information.

Feature 1 When the IButterfly network with m-bit implements cyclic shift, its i-th level control information, $0<i\leq\log_2 m$, i is a positive integer, and is divided into $m/2^i$ groups, each group has a total of 2^i-1-bit control information, and The control information is exactly the same.

Proof: To achieve an m-bit cyclic shift, you must complete the independent 2^i-1-bit cyclic shift of the $m/2^i$ group at i-th level, and the number of shift steps in each group is the same as the number of m-bit shift steps. The group input and output interconnection functions are also identical, so each group of control information is identical and feature 1 holds.

Feature 2 When the IButterfly network with m-bit implements cyclic shift, the control information of each group in the i ($0<i\leq\log_2 m$, i is a positive integer) stage is equal to, where $(d_{i-1},\ldots, d_0)$ correspond to the purpose. The binary code of the corresponding initial data at the position, and the superscript in $(d_{i-1})^0$ corresponds to the destination.

Proof:

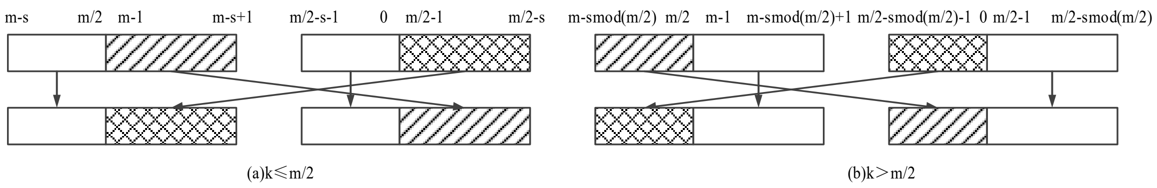


Figure 3 Two situations

1） Shift steps (k) is less than or equal to m/2

As shown in Fig. 3(a), when $k \leq m/2$, {m/2-1, ..., m/2-s} is set to 1, {m/2-s-1,...,0} is set to 0 at $\log_2 m$ level. Then, the number of 1 is equal to s, and s is added to the initial coding. It can be seen that the routing information of this level is equivalent to the i-bit when {m/2-1,..., m/2-s, m/2-s-1, ..., 0} change to binary code.

2） Shift steps (k) is greater than m/2

As shown in Fig. 3(b), when k>m/2, {m/2-1,…,m/2-smod(m/2)} is set to 0, {m/2-s mod(m/2)-1,…,0} is set to 1 at log2m level. Then, the number of 0 is equal to s, and s is added to the initial coding. It can be seen that the routing information of this level is equivalent to the i-bit when {m/2-1,…,m/2-smod(m/2), m/2-s mod(m/2)-1,…,0} change to binary code.

Feature 3 When the IButterfly network implements a cyclic shift, the corresponding routing information of each level of the left cyclic shift and the right cyclic shift is negated.

Proof: It is known from the characteristics of the cyclic shift that the left-shift k-step of the m-bit cycle is the same as the result of the right-shift m-k of the cycle. According to the feature 2 proof process, when the loop is left-shifted by k steps and the loop is right-shifted by k steps, the $\log_2 m$ level corresponding routing information is negated.

4. Research on parallel cyclic shift routing algorithm

4.1. Parallel left cyclic shift routing algorithm

From feature 1 and feature 2, the parallel looping left routing information generation process can be summarized as follows:

Step1: the initial position information and the number of shift steps are binary coded and added;

Step2: the corresponding level of left cyclic control information in the added results is extracted and copied to correspond to the control information of all the switches in the network.

Step3:, when multiple sets of cyclic shifts are operated in parallel, the routing information at each level after the $\log_2 n$ level is set to 0 according to the bit width n of each group, n≤m, where n is a power exponent of two.

According to the parallel looping left routing information generation process, the circular leftward routing information generation algorithm is as follows:

```
Input  $S^u$  Initial position code;  $Shamt$  Number of steps;  $n$  width of shift;
Output  $Crt[i]$          i-th control

for u=0,1,…,m          m is width of network
    $D^u = S^u + Shamt$
 for i=1,2,…,$\log_2 n$
    $Crt[i] = n/2^i\{(d_{i-1})^{2^{i-1}}, ..., (d_{i-1})^0\}$        $m/2^i\{\}$means several $\{\}$concatenation
```

4.2. Parallel bidirectional cyclic shift routing information generation algorithm

From feature 3 and the parallel left-shifted routing information generation algorithm, parallel bidirectional cyclic shift routing information generation algorithm is as follows:

```
Input  Input  $S^u$  Initial position code; $Shamt$  Number of steps;  $n$  width of shift; P  direction
 Output  $Crt[i]$          i-th control
 for u=0,1,…,m          m is width of network
      $D^u = S^u + Shamt$
 for i=1,2,…,$\log_2 m$
    if P=left
      $Crt[i] = m/2^i\{(d_{i-1})^{2^{i-1}}, ..., (d_{i-1})^0\}$
      else
      $Crt[i] = m/2^i\overline{\{(d_{i-1})^{2^{i-1}}, ..., (d_{i-1})^0\}}$     $m/2^i\{\}$means several $\{\}$concatenation
```

4.3. Uniformity of short word cyclic shift algorithm

Short word cyclic shift is essentially a special case of cyclic shift. The m-bit initial input data is grouped. Each group has a bit width of 2^k-bit, a total of $m/2^k$ groups, $0<k\leqslant\log_2 m$, and k is a positive integer. When the short word cyclic shift is completed, it is equivalent to the previous $\log_2(m/2^k)$ level

routing information is set to pass through to ensure that the data between the groups does not cross, and the actual number of shifting steps in each group is equal to 2^{sk} steps, and then the actual number of shifting steps is passed. The proposed two-way shift algorithm can calculate routing information at all levels.

5. Functional and performance evaluation

In order to objectively evaluate the circular shift routing information generation algorithm proposed in this paper, bidirectional cyclic shift function is implemented on the 64-bit IButterfly network, and Comprehensive analysis and evaluation in this paper is processed from two aspects: processing function and performance occupation.

5.1. Uniformity of short word cyclic shift algorithm

As shown in Table 1, the current main shift unit supports shift operation types. The results show that the logarithmic shifter generally only supports one-way shift operations due to structural limitations. Both [7] and [9] are based on the IButterfly network, but only bi-directional cyclic shifts and parallel shifts are supported. Short word shifts are not supported. Based on the literature [7] and literature [9], this design and literature [8] can implement a variety of short word cyclic shift operations under the same algorithm. Therefore, the design has higher flexibility.

Table 1. Functions of Different shift unit

unit	Bidirectional shift	Short words	Parallel shift
Log shift	√		
literature [7]	√		√
literature [9]	√		√
literature [8]	√	√	√
My design	√	√	√

Note：The table only compares the shift function with other documents

5.2. Uniformity of short word cyclic shift algorithm

In order to make objective comparisons with existing designs, the same general environment is used as literature [8]. As shown in Table 2, since the literature [7] uses the TSMIC 90nm process for comprehensive optimization, this paper uses area and delay parameters to normalize and compare the relative area and relative delay. The results show that the routing information generation algorithm proposed in this paper adopts the method of generating the routing information at all levels in parallel with the initial position coding and the number of shift steps. When the bidirectional cyclic shift function is implemented, the relative area is compared with the logarithmic shift method. It is 1.06x for the (Relative Area) logarithmic shifter and 1x relative latitude. Compared with literature [7] and literature [8], The frequency is increased by 8.6% and 3.6%, the area is reduced by 32% and 1.4%. At the same time, when the short word cyclic shift is also supported, the frequency in the literature [11] is increased by about 5%, and the area is reduced by about 6%. Although the relative delay is larger than that in [7], the shift operation supported by [8] is twice that of the former two units, and it is very flexible.

Table2. Performance comparison

unit	Area (NAND gates)	Total Area (um^2)	Relative Area	Buffer Area	Latency (ns)	Relative Latency
Log shift	1.30k	1875.32	1	162.64	0.53	1
literature [7]	-	-	1.38	-	-	1.18
literature [9]	2.02k	2906.75	1.55	608.52	0.58	1.11
literature [8]*	1.40k	2017.88	1.08	245.76	0.55	1.04

literature [8]	2.11k	3038.40	1.62	435.96	0.60	1.13
My design *	1.28k	1988,34	1.06	187.78	0.53	1
My design	1.99k	2857.90	1.52	376,98	0.57	1.07

*Denotes a design that supports the same function as literature [9]

6. Conclusion

This paper analyzes the three characteristics of cyclic shift corresponding routing information when IButterfly supporting cyclic shift. Based on this, this paper proposes an algorithm that can quickly generate routing information at all levels. Compared with existing algorithms, when bidirectional cyclic shift is only supported, resource can be effectively reduced and processing speed can be improved. At the same time, when the short word cyclic shift is supported, the resource occupancy is only slightly larger than that of [7], but it can support twice the shift types of the former. At the same time, the routing information generation algorithm proposed in this paper lays the foundation for the routing information generation algorithm of the IButterfly/Butterfly network to support the extraction of new complex bit replacements such as shift and shift insertion, and has a strong practical value and theoretical guidance significance.

References

[1] Khan L. 2013 *International Journal of Advanced Research in Computer Science*, 4 108.
[2] Hosseininia N, Boroumand S, Haghparast M. 2015, *Journal of Circuits System and Computers,* 24 1
[3] Sayilar G, Chiou D. 2014 *IEEE International Conference on Computer-Aided Design (ICCAD). San Jose: IEEE*, 2014. p 155.
[4] Hilewitz Yedidya, 2008 Advance Bit Manipulation Instructions: Architecture，Implementation and Applications. NewJersey: Princeton University chapter 4 pp 95-145
[5] Hilewitz Y, Shi Z J, Lee R B. 2004 *38th IEEE Annual Asilomar Conference on Signals, Systems, and Computers.* vol 2 p 1856
[6] Hilewitz Y, Lee R B. 2009 *Journal of Signal Processing Systems*, 53 145.
[7] Hilewitz Y, Lee R B. 2009 *IEEE Transactions on Computers*, 58 1035.
[8] Ma Chao, Dai Zibin, Li Wei, 2017 *Electronic Journal*, 45 1025
[9] Z X Chang, J S Hu, C Ma. 2013.*Computer Engineering and Technology*, 92.
[10] Intel Corporation. 2014 *Haswell Intel Architecture Software Developer's Manua*l.