基于 Inverse Butterfly/Butterfly 网络的置换-逆序与置换-移位选路算法

马 超[1],戴紫彬[1],李 伟[2],南龙梅[2]
(1. 解放军信息工程大学,河南郑州 450000; 2. 复旦大学集成电路国家重点实验室,上海 200433)

摘 要: 本文利用 Inverse Butterfly/Butterfly 多级动态互连网络的自路由和可重排特性,提出了基于该网络的置换-逆序和置换-移位选路算法. 它们都能够对所有一次通过该网络的任意置换结果动态地完成逆序和移位操作,且算法复杂度低,硬件实现简洁. 进一步,将本文提出的算法对基于该网络设计的置换操作进行了功能扩展,分别构建了置换-逆序、置换-移位以及置换-逆序-移位硬件单元. 并将它们在 SMIC 65nm 工艺下进行了综合,结果表明:当在以往研究成果上扩展逆序操作时,硬件电路面积仅增加约 6% 且几乎不影响原架构的延迟;当在以往研究成果上扩展移位和逆序-移位操作时,原架构以 18% 和 21% 的面积增加值和 30% 的延迟增加值,实现了功能性 2 倍的提升.

关键词: Inverse Butterfly/Butterfly 网络; 自路由; 置换-逆序算法; 置换-移位算法
中图分类号: TP331.2 **文献标识码:** A **文章编号:** 0372-2112 (2017)11-2685-10
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2017.11.016

Permutation-Reverse and Permutation-Rotation Routing Algorithms Based on Inverse Butterfly/Butterfly Network

MA Chao[1], DAI Zi-bin[1], LI Wei[2], NAN Long-mei[2]
(1. *The PLA Information Engineering University, Zhengzhou, Henan* 450000, *China*;
2. *State Key Lab of ASIC and System, Fudan University, Shanghai* 200433, *China*)

Abstract: In this paper, we propose routing algorithms for permutation-reverse and permutation-rotation operations based on Inverse Butterfly and Butterfly Networks respectively. The algorithms utilize self-routing and self-reconfigurable characteristics of the networks, and are capable of completing all reverse and rotation operations of arbitrary permutation-P. Their computational complexities are low and the hardware implementations are simple. Following this, we extend the functions of the previous permutation operations based on the networks with our proposed algorithms. And then, permutation-reverse, permutation-rotation and permutation-reverse-rotation hardware units are developed and synthesized in SMIC 65-nm process. The results show that when the reverse operations are extended in previous designs, the area of the original circuits is only increased by 6%, and the latency of original circuits is almost not affected. In addition, when the rotation and reverse-rotation operations are extended in previous designs, the area of the original circuits is increased by 18% and 21%, and the latency of them is increased by 30%, while the functions of original circuits are expanded 2 times.

Key words: Inverse Butterfly/Butterfly Network; self-routing; permutation-reverse algorithms; permutation-rotation algorithm

1 引言

复杂比特置换类操作在密码学、数字信号处理、图像处理等领域有着广泛地应用[1~3]. 例如,在密码学中,扩散和混乱特性是密码算法设计的首要原则[4,5],比特置换类操作又是实现扩散的主要手段[6],因此被应用于如 PRESENT、HIGHT、AES 等众多密码算法的设计中[7]. 然而面向字位宽优化的通用处理器对于这些细粒度比特级操作处理效率很低,需要将其转化成多条指令组合实现,这极大地制约了整个系统的处理性能. 因此,如何提高比特级置换类操作在处理器中的执行效率,成为了人们研究的热点. 一个 N = 16 比特的输入

收稿日期:2016-06-23;修回日期:2016-09-28;责任编辑:蓝红杰
基金项目:国家自然科学基金(No. 61404175)

序列其置换结果有 N! 约 20 万亿种,若采用全 Crossbar 互连结构实现,其硬件复杂度很高为 O(N^2),VLSI 实现非常困难. 多级动态互连网络-Inverse Butterfly/ Butterfly,能够通过改变内部交叉开关的状态,实现输入、输出结点间的不同连接从而灵活地完成数据的重新排列,使系统具有自路由、可重排特性[8]. 且拓扑结构规整,硬件复杂度较低为 $O(N\times \lg N)$,能够实现速度、面积和功耗的有效平衡,符合 VLSI 设计的基本条件[9,10],是实现置换操作的良好载体,因此引起了广泛关注.

近年来,基于 Inverse Butterfly/Butterfly 网络实现比特置换类操作的研究主要集中于两个方面:一方面,研究某一特定置换操作在其架构中实现时,各级开关状态的专用选路算法,如基于 Inverse Butterfly 网络的 PEX 和 GRP 指令操作、基于 Butterfly 网络的 PDEP 和 ROT 指令操作等[11],它们的选路算法比较简洁,易于硬件电路实现,因此具有动态实时改变的特性;另一方面,针对任意置换 P,研究普适性的选路算法来控制各级开关状态,如基于 Inverse Butterfly 级联 Butterfly 网络的 CROSS 指令操作选路算法[12]、基于双 Inverse Butterfly 多次通过的全置换选路算法[13]等. 它们虽然以较小的硬件消耗实现了任意置换操作,但选路算法却十分复杂,因此一般采用软件预计算选路信息,而后由硬件配置的方式来实现任意置换. 这消耗了一定的存储资源,且在电路正常工作时无法动态调整已配置完成的置换操作.

本文的研究重点是基于 Inverse Butterfly/ Butterfly 网络,提出了在任意 P 置换下实现逆序、移位操作的实时选路算法. 相比于特定置换下的专用选路算法,本文研究的范围更广,能够实现一切 P 置换下的逆序和移位操作;相比于选路信息需要预配置的任意置换选路算法,本文提出的算法能够以较低的资源消耗,动态地完成置换结果的逆序、子逆序和移位等操作,使一次配置信息能够实现多种常用置换,从而达到原架构功能倍增的目的.

2 相关工作及与本文的区别与联系

2.1 相关工作

Inverse Butterfly/Butterfly 网络作为多级动态互连网络的一种拓扑形式,通常应用于处理器与处理器、处理器与存储器之间的互连通信中,其结构特性及选路算法一直是学术界研究的热点[14]. 图 1 描述了一个 $N=8$-bit 的 Inverse Butterfly 网络拓扑结构,该网络有 $n=\log_2 N=3$ ($N=2^n$)级从上到下依次为第一级、第二级和第三级,每一级有 $N/2=4$ 个 2 输入交叉开关,每一个交叉开关在 1-比特选路控制信号的作用下完成一对输入信号的“交叉”或“直通”. 该网络共有 $\log_2(N)\times N/2=12$ 个交叉开关,每个交叉开关有两种配置状态,因此能够实现 $2^{\log_2 N\times N/2}=4096$ 种置换操作.

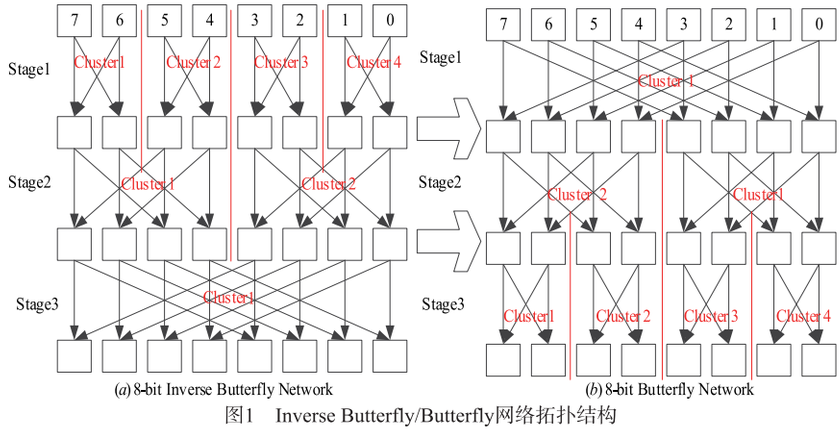


图1 Inverse Butterfly/Butterfly网络拓扑结构

从图 1(*a*)的数据流图中还可以看出,网络第一级有 4 个子蝶网络(簇)从左至右依次为 Cluster1 ~ Cluster4,簇内位宽为 2-bit,且各簇之间不会发生数据交互. 网络第二级有 2 个子蝶网络(簇)从左至右依次为 Cluster1 和 Cluster2,每个簇位宽为 4-bit,且簇间数据不发生交互. 网络最后一级仅有一个子蝶网络(簇)Cluster1,位宽为 8-bit. 若把该网络最后一级舍弃,剩下的部分可以看成两个独立的以 *N*/2-bit 为位宽的 Inverse Butterfly,它与 *N*-bit 位宽的 Inverse Butterfly 结构相似,只是规模较小,因此该网络具有迭代和可拆分特性. Butterfly 网络是 Inverse Butterfly 网络的逆结构,数据的分组间距为 $2^{\log_2(N)-i}$,在此不再详述.

改变该网络中的开关状态,能够实现不同结点之间的连接,进而完成一个输入数据的置换操作. 因此,Lee 等人率先利用这种自路由、可重排特性,将应用于通信交换领域的 Inverse Butterfly/Butterfly 引入到通用处理器内部硬件单元设计中,构造了一种新型移位-置换单元[15]. 该单元将传统循环移位指令(ROT)与复杂比特置换类指令如比特归类(GRP)、并行抽取和并行插入(PEX, PDEP)、块抽取与插入(EXTRR,DEP)统一到了一个架构

下. 并针对移位及每一种特定置换操作都提出了各自专用的实时选路算法. 其中 GRP 指令属于复杂线性变换类指令,因其具备较强的抗线性和差分攻击能力,而被用于新一代密码算法的设计中[16]. PEX 和 PDEP 指令由于在实际应用中的广泛性,已被纳入 Intel 在 2013 年发布的 HASWELL 处理器指令集中[17]. ROT 指令是一种常用的循环移位操作指令,被多数处理器指令集所支持[18]. 随后,Chang 等人在 Lee 的基础上,针对循环移位操作选路算法串行执行的问题进行了研究,提出了一种并行度更高的循环移位选路信息生成算法,大幅提升了循环移位操作的硬件实现性能[19]. 对于实现任意 $N!$ 种全置换操作,由于输入数据与输出数据之间没有特定的置换规律,且结果空间巨大. 因此 Lee 等人又进一步将 Inverse Butterfly 与 Butterfly 网络前后级联,构成 CROSS 可重排无阻塞网络. 输入数据一次通过该网络即可以完成任意置换,但选路算法十分复杂,往往采用软件提前计算选路信息,然后静态配置到网络各开关中以完成置换操作. 该过程消耗了额外的存储资源,并给系统带来了一定的重构开销,不利于通用处理器的集成,因此往往被应用于专用指令处理器和可重构处理器架构中[20].

2.2　本文的研究动机

基于 Inverse Butterfly/Butterfly 网络,针对特定置换的专用选路算法,硬件实现简洁,实时性较好,但功能单一,算法本身不具有可扩展性. 而针对任意置换的普适性选路算法,虽然功能强大,但算法过于复杂,硬件开销巨大,不具有实时性. 因此,本文结合工程中的实际应用,通过调整网络各级开关的初始置换选路信息,使其能够对任意置换结果动态地完成逆序和移位操作. 从理论上讲,本文研究的任意置换下动态逆序和移位操作选路算法,使一切基于该网络实现特定置换的专用选路算法具备一定的功能多样性,同时使通过静态配置实现任意置换的通用选路算法具备一定的动态实时性,从而有效地平衡专用选路算法功能单一和通用选路算法实时性不强之间的差异. 从工程实际出发,本文提出的逆序操作可被应用于存储器中大小端的数据对齐[21]、密码算法的逆序排列[22]等领域. 而置换-移位操作,若置换限定为逆序时,它将完成一个序列的逆序后循环移位操作,该操作可被应用于数字信号处理中[23]. 下面,本文将重点研究基于 Inverse Butterfly、Butterfly 网络的逆序和移位选路算法实现原理,以进一步拓展该架构的置换能力.

3　置换-逆序选路原理及算法实现

3.1　基于 Inverse Butterfly/Butterfly 网络的置换-逆序原理

性质 1　若 N-bit 数据 $M=\{a_{N-1},a_{N-2},\cdots,a_0\}$ 一次通过 Butterfly 网络已经实现任意置换 P,记 $P=\begin{Bmatrix}a_{N-1} & a_{N-2} & \cdots & a_0\\ a_0 & a_{N-1} & \cdots & a_{N-2}\end{Bmatrix}$,那么通过调整初始置换各级控制信息,该网络还能够实现其结果的逆序置换 $\sim P$,记 $\sim P=\begin{Bmatrix}a_{N-1} & a_{N-2} & \cdots & a_0\\ a_{N-2} & \cdots & a_{N-1} & a_0\end{Bmatrix}$.

证明:在 Butterfly 网络的第 i 级($i=1,2,\cdots lg(N)$),输入数据被分成 2^{i-1} 个簇,各簇内数据位宽为 $w=N/2^{i-1}$,通过将初始控制信息取反,每一比特初始输出数据将移位 $w/2$-bit,移位后的输出数据与原输出数据相比,实现了原网络中左、右两个部分(位宽为 $w/2$-bit)位置的互换. 如图 2(a)所示,当将初始控制序列(Initial control bits)按位取反后,图中右半部分输出数据将整体移位 $w/2$-bit,与左半部分数据互换位置,实现两个位宽为 $w/2$-bit 数据间的逆序,其结果如图 2(b)所示. 由于 Butterfly 网络的递归、迭代特性,网络第一级完成了 N-bit 初始输出数据左 data(L)、右 data(R)两个部分的逆序. 网络第二级,将第一级逆序后的两部分数据独立地作为输入,再将这两部分数据对应的初始控制信息分别取反,则该级网络的输出数据将以 $N/4$-bit 为位宽形成四个独立的簇,且簇间数据是逆序的. 依此类推,当数据到达第 lg(N)级时,将形成以 2-bit 为位宽的 $N/2$ 个簇,且簇间与初始输入数据相比为它的逆序. 这时,这时仅需将每个簇内的 2-bit 数据对应的初始控制信息取反,即可完成基于任意置换 P 的逆序操作,如图 3 所示.

性质 2　在 Inverse Butterfly 网络的第 i 级,若分别将左、右两个部分的初始输入数据逆序(位宽均为 $w/2$-bit),那么通过调整初始控制信息,其相应的初始输出数据(位宽为 w-bit)也能够完成逆序.

证明:在 Inverse Butterfly 网络的第 i 级,输入数据被分成 $N/2^{i-1}$ 个簇,簇内数据位宽为 $w/2=2^{i-1}$. 通过将初始控制信息取反,能够实现原网络中左、右两个部分数据位置的互换. 如图 4(a)所示,若将初始控制信息序列按位取反:Control bits = "01…0"→"10…1",并作用于初始网络,其输出端左、右两部分数据将互换位置,如图 4(b)所示. 然后再将初始左、右两个部分的输入数据各自逆序,同时也将取反后的控制序列跟随输入数据一起逆序,那么初始输出数据(位宽为 w-bit)也就完成了逆序操作如图 4(c)所示. 图 4(c)中分别将左、右两个部分的初始输入数据逆序,同时再将取反后的控制序列逆序从"10…1"到"1…01",那么输出的数据与初始输出数据相比,完成了一个逆序过程. 因此性质 2 成立.

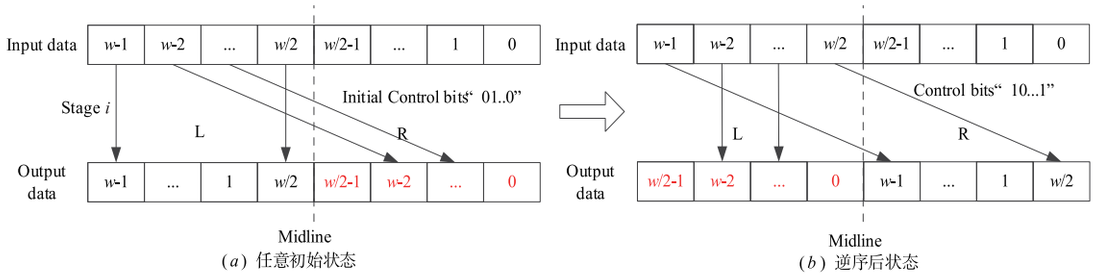


图2 网络第i级逆序原理

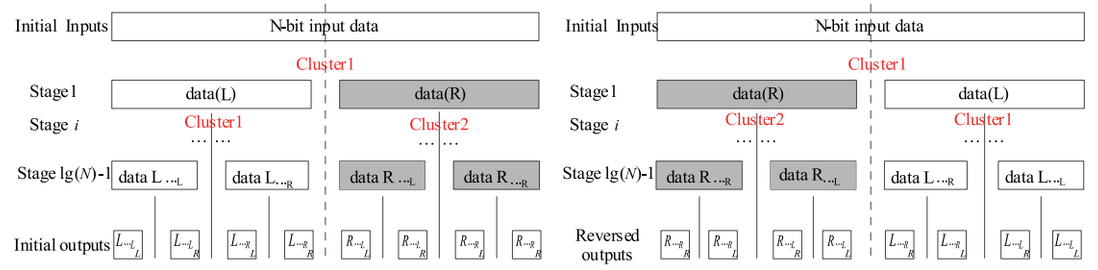


图3 Butterfly网络逆序原理

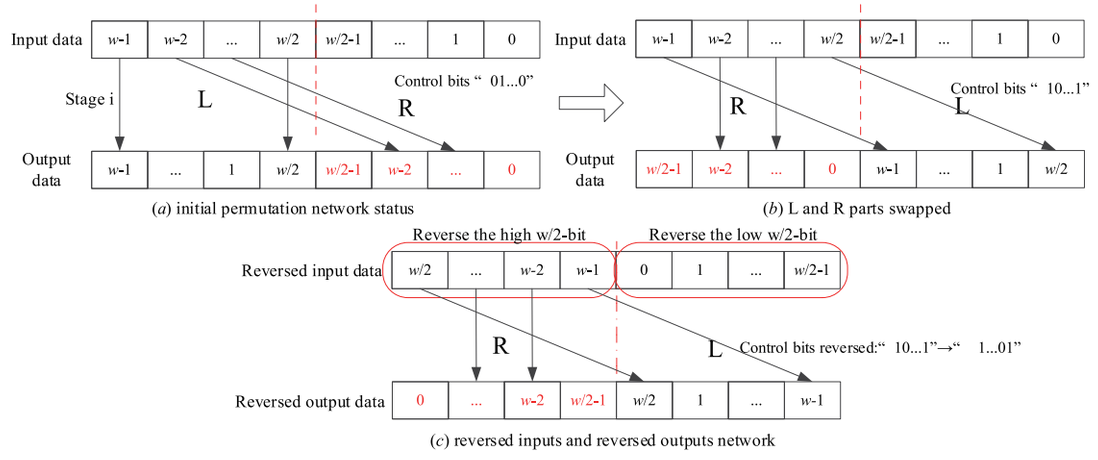
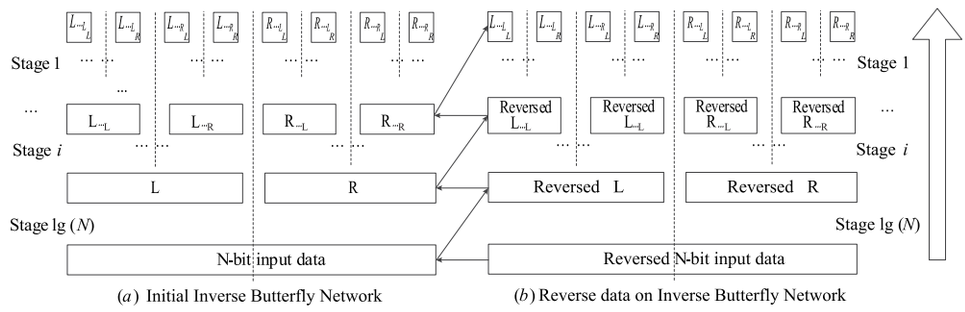


图4 Inverse Butterfly网络逆序特征

性质 3 若 N-bit 数据 $M=\{a_{N-1},a_{N-2},\cdots,a_0\}$ 一次通过 Inverse Butterfly 网络实现置换 P，记 $P=\begin{Bmatrix} a_{N-1} & a_{N-2} & \cdots & a_0 \\ a_0 & a_{N-1} & \cdots & a_{N-2} \end{Bmatrix}$，通过调整初始置换各级的控制信息，该网络还能够实现其结果的逆序置换 $\sim P$，记 $\sim P=\begin{Bmatrix} a_{N-1} & a_{N-2} & \cdots & a_0 \\ a_{N-2} & \cdots & a_{N-1} & a_0 \end{Bmatrix}$.

证明：根据性质 2 知，若需完成网络最后一级初始输出数据的逆序，可以通过将最后一级初始输入数据按照左、右两部分分别逆序，同时调整初始控制信息实现. 如图 5(a)所示，通过将网络最后一级初始输入端左 L、右 R 两部分数据(位宽为 $N/2$-bit)逆序，同时调整初始控制信息，那么就可以完成最后一级数据的逆序(位宽为 N-bit). 对比图 5(a)、(b)中倒数第二级输出(即最后一级的输入)数据可以发现，图 5(b)中倒数第二级的输出数据(Reversed L、Reversed R)是图 5(a)初始网络中该级输出数据(L、R)的逆序. 然后，将图 5(a)中 L、R 两部分数据分别逆序，根据性质 2 可知，这样的操作可以通过将该级各簇内输入数据逆序，同时调整簇内初始控制信息实现. 以此方式向上递推，当数据到达该网络第一级时，初始输出数据的逆序，也可以通过将该级初始输入数据逆序，同时调整初始控制信息实现. 而第一级数据的初始输入位宽为 1-bit，它逆序与否，其结果均与初始第一级初始输入数据序列相同. 因此，从图 5(b)中初始输入端出发，整个网络就相当于完成了任意置换 P 下的逆序置换 $\sim P$.



(a) Initial Inverse Butterfly Network　　(b) Reverse data on Inverse Butterfly Network

图5　Inverse Butterfly网络逆序原理

3.2　基于 Inverse Butterfly/Butterfly 网络的置换-逆序选路算法

根据性质 1 和性质 3 及在任意 P 置换下扩展逆序操作时,各级控制信息改变的规律,将基于 Butterfly 网络和 Inverse Butterfly 网络的逆序操作选路算法分别总结如下:

(1)基于 Butterfly 网络的逆序算法

算法 1　基于 Butterfly 网络的从初始置换 P 到 ~P 的逆序操作选路算法

```
Input:Original Control Bits,Or_cb;
//total of lg(N) × N/2 control bits
Output: Reverse Control Bits,Reverse_cb;
// total of lg(N) ×N/2 control bits
For (i=1,i<=lg(N),i++)      // i stand for each stage
  {
    If (i==1)
      {
        Reverse_cb1[N/2-1:0] = flip each bit of the Or_cb1[N/2-1:0];
      }
    Else
      {
        J=2^(i-1);
// J stand for the amount of sub-networks(cluster) in stage i
        For (l=1,l<=J,l++)
        {
        Flip_clusterl = flip each bit of the clusterl[N/2^i-1:0];
        }
// clusterl stand for the Or_cbi at each sub-butterfly network in stage i, and the cluster number 'l' is increased from left most to right most,the total bits of clusterl(l=1,…,J) are equal to Or_cbi.
        Reverse_cbi[N/2-1:0] = {Flip_clusterJ[N/2^i-1:0]:
        Flip_clusterJ-1[N/2^i-1:0],…, Flip_cluster1[N/2^i-1:0]}
      }
  }
```

(2)基于 Inverse Butterfly 网络的逆序算法

算法 2　基于 Inverse Butterfly 网络的从初始置换 P 到 ~P 的逆序操作选路算法

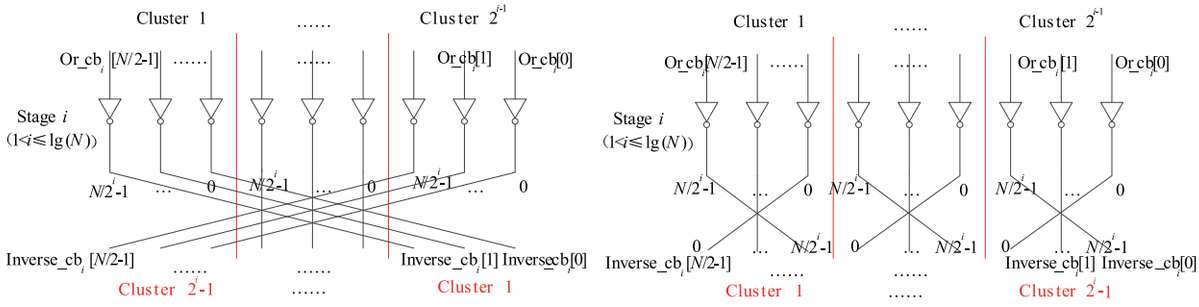
```
Input:Original Control Bits,Or_cb;
//total of lg(N) ×N/2 control bits
Output: Reverse Control Bits,Reverse_cb;
// total of lg(N) ×N/2 control bits
For (i=1,i<=lg(N),i++)   // i stand for each stage
  {
    If (i==1)
      {
      Reverse_cb1[N/2-1:0] = flip each bit of the Or_cb1[N/2-1:0];
      }
    Else
      {
      J=N/2^i;
      For (l=1,l<=J,l++)
        {
        Flip_clusterl = flip each bit of the clusterl[2^(i-1)-1:0];
// clusterl stand for the Or_cbi at each sub-interse butterfly network in stage i, and the cluster number 'l' is increased from left most to right most.
        Reverse_clusterl = reverse each bit in each of the flip_clusterl;
        }
      Reverse_cbi[N/2-1:0] = {Reverse_cluster1[2^(i-1)-1:0]:
      Reverse_cluster2[2^(i-1)-1:0],…,
      Reverse_clusterJ[2^(i-1)-1:0]}
  }
}
```

3.3　置换-逆序算法关键模块设计

算法 1 中核心操作是 Flip_cluster$_l$ 和 Reverse_cb$_i$ 这两个函数,它们均处于两个循环嵌套体内,被执行 N-1 次,因此算法 1 的时间复杂度为:$O(N)$. 其中第一个函数实现的是将所有第 i 级的初始选路信息 Or_cb$_i$[$N/2-1$]进行比特取反,然后再以簇为最小单位(clus-

$\text{ter}_1,\cdots,\text{cluster}_l, 1<=l<=2^{i-1}$)进行簇间逆序操作,其结果则为网络该级最终的选路控制信息,如图 6(a)所示.

算法 2 与算法 1 的核心操作相似,其算法复杂度仍为:$O(N)$. 它需先将所有第 i 级初始控制信息取反,而后再将取反后的初始控制信息在各个簇内进行逆序操作,而簇间无需进行任何操作,如图 6(b)所示. 图 6(b)中取反后的控制序列在簇内进行了逆序,从而得出最终的控制信息 $\text{Reverse\_cb}_i[N/2-1]=(\text{reverse\_cluster}_1,\cdots,\text{reverse\_cluster}_l, 1<=l<=2^{i-1})$. 算法 1 和 2 硬件仅由反向器和互连线构成,因此资源消耗极低且关键路径延迟很小. 这为算法的 VLSI 高效集成奠定了良好的理论基础.



(a) 算法1关键模块设计 (b) 算法2关键模块设计

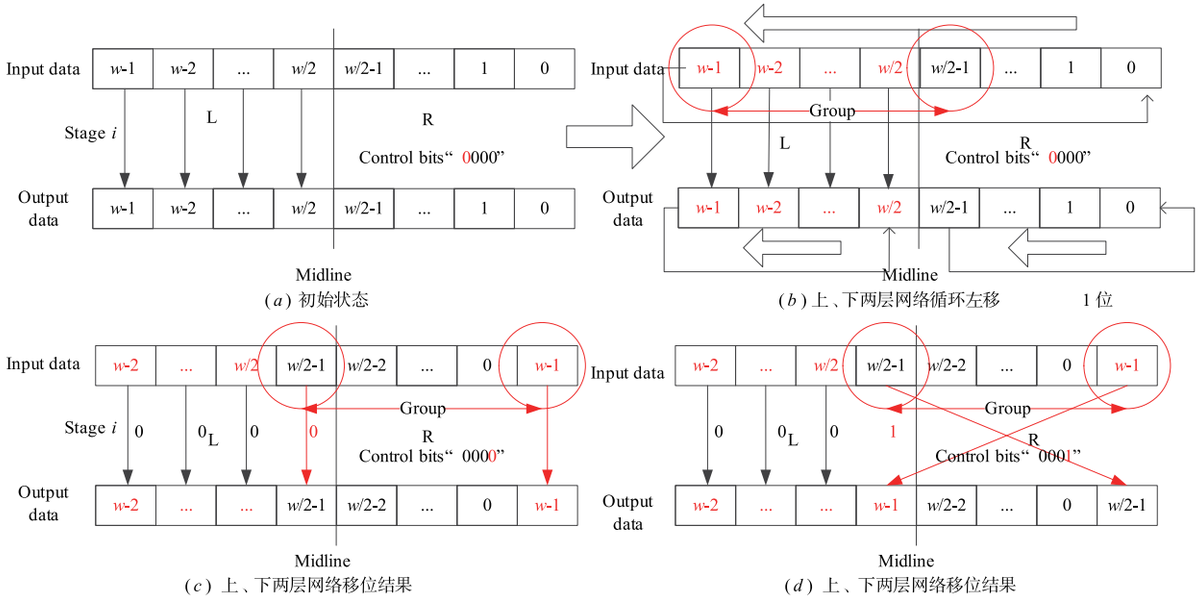
图6 算法1和算法2关键部分硬件实现

4 置换-移位选路原理及算法实现

4.1 基于 Inverse Butterfly 网络的置换-移位原理

性质 4 在 Butterfly 网络的第 i 级,若将初始输入数据循环左移 S-bit(位宽为 $w=N/2^{i-1}$),那么通过调整初始控制信息,其相应的初始输出数据则能够以左、右两个部分(位宽为 $w/2$-bit)分别完成 S-bit 循环移位.

证明:图 7(a)中,假设 w-bit 数据在第 i 级初始均为直通状态,其初始控制信息序列对应为全“0”. 当输入数据以 w 为位宽循环左移 1 位时,图 7(b)中左半部分最高位数据($w-1$)将被移位到右半部分最低位,同时右半部分中的最高位数据($w/2-1$)将越过网络中线到左半部分的最低位,其余数据仅左移 1 位并没有改变其初始所属的部分. 本文将左、右两部分中最高位的 2-bit 数据($w-1$)和($w/2-1$)称为特殊对,它们被同一个控制信号控制,如图 7(b)中虚线圆框所示. 若初始控制信息序列也随着循环左移 1 位,那么仅特殊对的输出数据会对调原属部分,其它输出数据左移 1 位,其所属部分保持不变,如图 7(c)所示. 由 Butterfly 网络的拓扑特性可知,处于 i 级的输入数据能够完成间隔距离为 $w/2$-bit 的移位与否. 那么,若将图 7(b)中特殊对所对应的控制信息取反即为



(a) 初始状态 (b) 上、下两层网络循环左移 1 位

(c) 上、下两层网络移位结果 (d) 上、下两层网络移位结果

图7 直通状态下特殊对移位规律

“0001”,那么,该特殊对将重新回到原来所属的部分中,这时该级的输出序列相当于将初始输出序列从中间分成左右两个部分,各自部分内数据循环左移 1 位的结果,如图 7(*d*)所示. 当特殊对初始状态为“1”交叉时,与直通“0”情况相似,将其取反即可.

若该网络再完成一次循环左移 1-bit 操作,初始输入的下一对数据将成为特殊对即($w-2$)与($w/2-2$). 当输入数据序列循环左移 *S*-bit 时,特殊对依次向后传递. 相应地需要将初始控制序列循环移并末位取反 *S* 次,那么网络最终的输出结果相当于初始输出数据分别以左、右两个部分各自循环左移 *S* 位的结果. 综上所述,性质 4 成立.

性质 5　若 *N*-bit 数据 $M=\{a_{N-1},a_{N-2},\cdots,a_0\}$ 一次通过 Butterfly 网络实现任意置换 $P:M\rightarrow F$,其中 $F=\{a_0,a_{N-1},\cdots,a_{N-2}\}$ 是 *M* 中各元素的一种任意排列,记 $P=\begin{Bmatrix}a_{N-1} & a_{N-2} & \cdots & a_0\\ a_0 & a_{N-1} & \cdots & a_{N-2}\end{Bmatrix}$,那么通过调整该网络的初始选路控制信息,还能够实现置换 $P':M'\rightarrow F$,其中 M' 是数据序列 *M* 循环移位 *S*-bit($0\leqslant S\leqslant N$)的结果.

证明:由性质 4 知,若将图 8(*a*)中 *N*-bit 初始置换网络输入数据(Initial Inputs)循环左移 *S* 位,那么通过调整网络第一级初始选路信息,其初始输出数据能够分别以左、右两个部分(位宽为 *N*/2-bit)分别完成 *S*-bit 循环移位,如图 8(*b*)中第一级输出数据所示. Butterfly 网络是一种迭代递归网络,若分别将初始置换网络中第二级两个以 *N*/2-bit 为位宽的子蝶(簇)网络中初始输入数据(相当于第一级的输出数据)循环移位 *S*,其相应的初始输出数据则能够以左、右两个部分(位宽为 *N*/4-bit)分别完成 *S*-bit 循环移位,如图 8(*b*)中第二级所示. 以此方式向下递归处理,数据在第 lg(*N*)级有 *N*/2 个簇,簇内数据位宽为 2-bit,当它循环移位 *S*-bit 时,初始输出数据将以 1-bit 为位宽循环移位,那么移位后的输出数据则与最后一级初始输出数据相同,如图 8(*b*)中最后一级输出所示. 因此输入数据序列 *M* 循环移位 *S* 后得到序列 *M*',一次通过该网络能够完成从 $M'\rightarrow F$ 的置换 *P*',性质 5 成立.

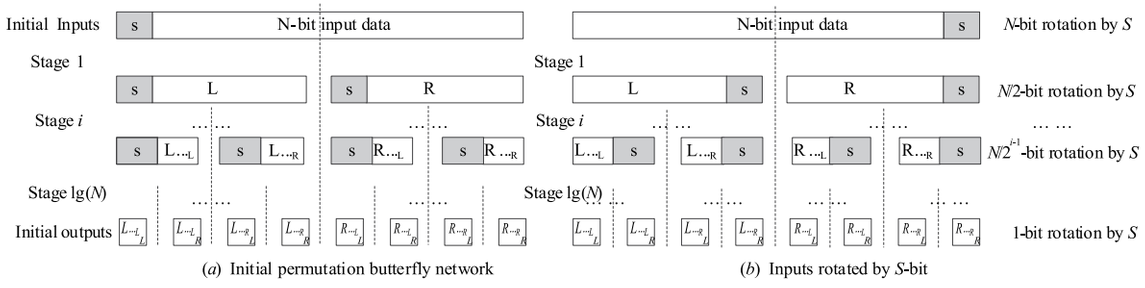


图8　置换-移位实现原理

Inverse Butterfly 网络是 Butterfly 网络的逆结构,若数据序列从图 8(*b*)中的 Butterfly 网络最后一级向第一级流动,很容易得出如下推论:

推论 1　若 *N*-bit 数据 $M=\{a_{N-1},a_{N-2},\cdots,a_0\}$ 一次通过 Inverse Butterfly 网络实现置换 $P:M\rightarrow F$,其中 $F=\{a_0,a_{N-1},\cdots,a_{N-2}\}$,记 $P=\begin{Bmatrix}a_{N-1} & a_{N-2} & \cdots & a_0\\ a_0 & a_{N-1} & \cdots & a_{N-2}\end{Bmatrix}$,那么通过调整初始选路控制信息该网络还能够实现置换结果的移位操作记 $P':M\rightarrow F'$,其中 *F*' 是数据序列 *F* 循环移位 *S*-bit($0\leqslant S\leqslant N$)的结果.

4.2　基于 Inverse Butterfly 网络的置换-移位算法

根据推论 1 以及各级初始选路信息的改变规律,将基于 Inverse Butterfly 网络的置换-移位操作算法提取如下:

算法 3　**基于 Inverse Butterfly 网络的从初始置换 P 到 P' 置换-移位操作的选路算法**

```
Input: S;
//rotation bit
Input: Original Control Bits, Or_cb;
//total of lg(N) × N/2 control bits
Output: Rotational Control bits, Ro_cb;
// total of lg(N) × N/2 control bits
(1) For (i=1, i<=lg(N), i++)
    {
      J = N/2^i;
      For (l=1, l<=J, l++)
      Ro_cb_i = RLTR(cluster_l[2^(i-1) -1:0], S);
    }
//cluster_l stand for the Or_cb_i at each sub-butterfly network in stage i,
and the cluster number increased from left most to right most.
(2) RLTR(cluster_l[2^(i-1) -1:0], S)
    For (a=0, a<=S, a++)
      {
        cluster_l[2^(i-1) -1:0] <<<1
//M zero string left rotation one time
        cluster_l[2^(i-1) -1:0] ~^L
//   "~^L" Filp the least significant bit of the rotated string cluster_l
      }
```

4.3 置换-移位算法关键模块设计

算法 3 实现了 Inverse Butterfly 网络架构下任意置换 P 结果的移位置换 P',其中函数 RLTR(cluster$_l$[$2^{i-1}-1:0$])是该算法的关键路径,在最坏情况下将被执行 N^2 次. 因此,该算法的复杂度为: O(N^2),属于 2 阶复杂度. 本节将重点对该函数的硬件高速实现进行研究. 函数 RLTR(cluster$_l$[$2^{i-1}-1:0$])以 2^i 为周期. 例如,当 $i=3$ 时,若初始序列为"0110",且 S 从 0-7 变化,RLTR 函数运算结果如表 1 所示.

表 1 $i=3$,RLTR 函数变化规律

S	0	1	2	3	4	5	6	7
RLTR(cluster$_l$)[$2^{i-1}-1:0$]	0110	1101	1010	0100	1001	0010	0101	1011

从表 1 中 RLTR 函数随 S 变化的结果可以发现,它与 8-bit 序列"0110_1001"循环移位后取高四位输出值的结果相同,而该 8-bit 序列中低 4 位输入是高 4 位初始输入值的按位取反序列. 因此本文首先将 RLTR 函数中初始序列{d3,d2,d1,d0}按位取反后{~d3,~d2,~d1,~d0}与原序列并置,输入到 8-bit 对数移位器中,如图 9 所示. 对数移位器根据移位位数的二进制数($S=s_2s_1s_0$),以 2 的幂指数进行移位. 该结构优点是移位位数的二进制数可直接作为各级开关的控制信号,移位速率高且硬件实现简洁. RLTR 函数仅需要 8-bit 输出序列的高 4 位结果,因此还可以将最后一级的低 4 位输出舍弃,从而降低电路资源消耗.

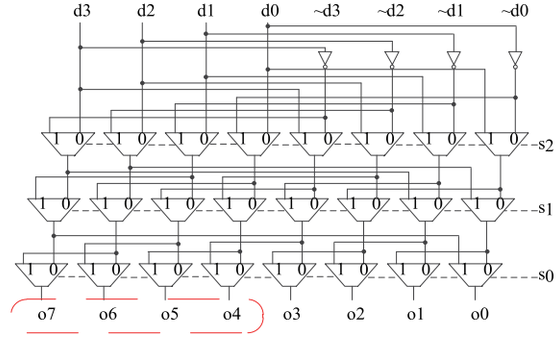


图9 RLTR(cluster$_3$[3:0],S)函数硬件实现

5 功能与性能分析

(1)功能分析

本节将提出的置换-逆序和置换-移位选路算法实例化,分别对现有基于 Inverse Butterfly /Butterfly 网络的动态专用置换类指令和全置换静态配置类指令进行了功能扩展,如表 2 所示.

表 2 基于 Inverse Butterfly/Butterfly 网络的初始置换指令及其扩展置换

扩展指令 / 置换指令	Reverse P	Sub-Reverse P 2^i(i=0,1,2,…,lg(N))	Rotation P	Reverse-rotation P	K
GRP (ibfly)	√	–	√	√	3
PEX (ibfly)	√	–	√	√	3
PDEP(ibfly)	√	√	–	–	2
ROT(bfly ibfly)	√	√(ibfly)	√	√	4
CROSS(bfly & ibfly)	√	–	√	√	3

表 2 中列举了基于 Inverse Butterfly/Butterfly 网络的 4 种特定置换指令和一种全置换指令(均为 32-bit 位宽),以及它们针对逆序、子字逆序、循环移位、逆序-循环移位等置换的可扩展性. 其中第一列置换指令后括号中的内容是该指令的硬件实现架构,最后一列参数 K 为扩展因子,表示该指令能够被扩展置换的个数. 从表 2 可知,这些指令均可以实现其结果的逆序操作. 但是,扩展子字逆序操作时,GRP 指令、PEX 指令和 CROSS 全置换指令并不能支持该置换,这是因为它们均基于 Inverse Butterfly 网络实现,其拓扑结构决定了它的输入数据只能在各级簇内进行移位,而簇间数据无法交互,因而无法完成子字逆序操作. 当扩展循环移位操作时,仅有 PDEP 指令无法完成扩展,这是因为该指令基于 Butterfly 网络实现,该网络拓扑结构中最后一级的数据是以 2-bit 为一个簇,簇间数据无法进行移位. 逆序后循环移位操作的可扩展能力与循环移位操作可扩展性相同,仅有 PDEP 指令无法完成该功能扩展. 需要说明的是,CROSS 指令虽然能够实现任意置换操作,但是却不具有实时性,其选路信息需由软件提前计算,而后硬件配置的方式来完成任意置换. 因此若要完成一次置换结果的逆序、循环移位或者逆序-循环移位等常用置换操作,该电路必须停止工作,待重新计算选路信息并完成配置完后,电路才能正常工作,这将带来极大的重构开销,严重影响电路性能. 若在该架构上扩展本文提出的 3 种选路算法硬件电路后,则能够实现配置一次选路信息,可动态实现多种置换操作的效果. 总之,从功能角度出发,本文研究的置换-逆序和置换-移位选路算法,不仅是对现有指令操作的一种功能扩展($K\geqslant 2$),同时对未来基于该网络实现的一切特定置换也具有普适性的扩展作用.

(2)性能分析

本文首先对表 2 中的 5 种置换以及本文提出的 3 种算法进行了硬件代码编写,然后使用 NC-Verilog 对其功能进行了覆盖性测试,在功能正确的基础上,将它们在 SMIC 65-nm 工艺[24]下进行了逻辑综合与优化,综合

时环境参数设置为:最慢工艺角(ss)、最低温度(-40℃)和最低电压(1.08v). 采用 flatten 的优化策略并设置时序优先,其结果如表 3 所示.

表 3　置换指令功能扩展前后硬件面积、延迟对比

指令	Original Area (um^2)	Expended Area (um^2)			Original Latency (ns)	Expended Latency (ns)		
		Reverse P	Rotation P	Re-rotation P		Reverse P	Rotation P	Re-rotation P
ROT	1650.72	1800.34 (9.1%)	2168.13 (31.3%)	2263.78 (37.1%)	0.48	0.49 (1.02x)	0.68 (1.42x)	0.69 (1.41x)
PDEP	2745.68	2932.39 (6.8%)	–	–	0.56	0.57 (1.02x)	–	–
PEX	2857.83	3038.71 (6.3%)	3394.59 (18.8%)	3484.26 (21.9%)	0.57	0.58 (1.02x)	0.76 (1.33x)	0.77 (1.35x)
GRP	5429.13	5709.13 (5.2%)	6341.64 (16.8%)	6611.78 (21.8%)	0.63	0.64 (1.02x)	0.83 (1.32x)	0.84 (1.33x)
CROSS	3218.96	3379.28 (5.9%)	3795.15 (17.9%)	3943.23 (22.5%)	0.91	0.91 (1.00x)	0.91 (1.00x)	0.91 (1.00x)

表 3 中列举了 ROT、PEX、PDEP、GRP 和 CROSS 等五种置换指令对应硬件单元的面积(Original Area)和延迟(Original Latency). 其中 CROSS 指令对应的硬件电路由 Butterfly 网络后接 Inverse Butterfly 网络组成,因此该单元的关键路径由两个网络拼接后所有级数的总和组成,其延迟为 0.91ns. 当该指令扩展逆序和移位操作时,由于逆序和移位算法本身硬件电路的延迟较小,不会大于 0.91ns,因此该指令在扩展功能后对原有架构的延迟不产生任何影响.

表 3 中五种指令的功能单元虽然面积差异较大,但当扩展逆序功能(Reverse Permutation Function)后,它们单元面积增加比例都很小,约为 6% 左右. 而且随着指令硬件复杂度的提升,扩展功能后增加的面积占总面积的比例呈下降趋势. 且它们在扩展逆序操作后,对原硬件单元的延迟影响甚微仅增加约 2%. 因此置换-逆序算法,不仅硬件实现简洁,而且几乎不影响原架构的处理性能. 置换-移位(Rotation Permutation Function)算法本身复杂度较高为 O(N^2). 当四种置换指令(不包括 PDEP 指令)扩展移位操作后面积增加明显,其中对 ROT 指令对应的硬件单元的面积响最大,面积增加约 30%,这是因为 ROT 指令本身路由算法比较简洁,硬件资源占用较少. 其余 3 种指令硬件单元面积增加了约为 18%. 但由于算法内在并行度较高,硬件上可以并发执行,因此对原有硬件单元延迟影响不大,约为原有架构延迟的 1.3 倍. 最后在实现上述两种操作的基础上,还进一步设计了基于四种指令(不包括 PDEP 指令)的逆序-移位单元(Re-rotation Permutation Function),该单元的面积和延迟增加比例与扩展置换-移位单元相近,分别增加 21% 和 30% 左右. 这也印证了扩展逆序操作算法简洁且利于硬件实现的结论.

6　结束语

本文分别提出了基于 Inverse Butterfly /Butterfly 网络的置换-逆序算法和基于 Inverse Butterfly 网络的置换-移位算法. 并将三种算法进行了硬件实现,结果表明:若在已有置换指令上扩展逆序操作,其硬件单元的面积增加很小约 6%,且几乎不影响原单元的延迟;若在已有置换指令上(不包括 PDEP 指令)扩展移位操作,其硬件单元面积增加适中约为 18%,且对于原电路延迟影响较小约为原电路延迟的 1.3 倍. 三种算法即具有特定置换操作专用选路算法的实时性,又具有任意置换操作复杂选路算法的普适性,是对 Inverse Butterfly/Butterfly 网络功能的一次完善,也为未来该架构下集成更多的功能提供有益的参考.

参考文献

[1] Shan W, Fu X, Xu Z. A secure reconfigurable crypto IC with countermeasures against SPA, DPA and EMA[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2015, 34(7): 1201 – 1205.

[2] Yin S Y, Ouyang P, Chen T, et al. A configurable parallel hardware architecture for efficient integral histogram image computing[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2016, 24(4): 1305 – 1318.

[3] Liu R. Chaos-based fingerprint images encryption using symmetric cryptography[A]. Proc of 9th IEEE International Conference on Fuzzy Systems and Knowledge Discovery[C]. Sichuan, China: IEEE, 2012. 2153 – 2156.

[4] Shan W, Zhang X, Fu X Y, et al. VLSI design of a reconfigurable S-box based on memory sharing method[J]. IEICE Electronics Express, 2014, 11(1): 1 – 6.

[5] Shan W, Chen X, Li B, Cao P, et al. Evaluation of correla-

tion power analysis resistance and its application on asymmetric mask protected data encryption standard hardware [J]. IEEE Transaction on Instrumentation and Measurement, 2013, 62(10): 2716 – 2724.

[6] Shan W, Chen X, Lu Y C, et al. A novel combinatorics-based reconfigurable bit permutation network and its circuit implementation[J]. Chinese Journal of Electronics, 2015, 24(3): 513 – 517.

[7] Bansod G, Raval N, Pisharoty N. Implementation of a new lightweight encryption design for embedded security[J]. IEEE Transactions on Information Forensics and Security, 2015, 10(1): 142 – 151.

[8] Nassimi D, Sahni S A. Self_routing benes network and parallel permutation algorithm[J]. IEEE Transaction on Computers, 1981, 30(5): 332 – 340.

[9] Yin S Y, Ouyang P, Liu L B, et al. A fast and power-efficient memory-centric architecture for affine computation [J]. IEEE Transactions on Circuits and Systems-II: Express Brief, 2016, 63(7): . 668 – 672.

[10] Yin S Y, Gu J Y, Liu D J, et al. Joint modulo scheduling and vdd assignment for loop mapping on dual-vdd CGRAs [J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2016, 35(9): 1475 – 1488.

[11] Hilewitz Y, Lee R B. Fast bit gather, bit scatter and bit permutation instructions for commodity microprocessors [J]. Journal of Signal Processing Systems, 2008, 53(1 – 2): 145 – 169.

[12] Shi Z, Yang X, Lee R B. Arbitrary bit permutations in one or two cycles[A]. 14th IEEE International Conference on Application-Specific Systems, Architectures and Processors[C]. Netherlands: IEEE, 2003. 237 – 237.

[13] Hilewitz Y, Shi Z J, Lee R B. Comparing fast implementations of bit permutation instructions[A]. 38th IEEE Annual Asilomar Conference on Signals, Systems, and Computers[C]. United State: IEEE, 2004. 1856 – 1863.

[14] Bansod G, Gupta A, Ghosh A, et al. Experimental analysis and implementation of bit level permutation instructions for embedded security[J]. Wseas Transactions on Information Science & Applications, 2013, 10(9): 303 – 312.

[15] Hilewitz Y, Lee R B. A new basis for shifters in general-purpose processors for existing and advanced bit manipulations[J]. IEEE Transactions on Computers, 2009, 58(8): 1035 – 1048.

[16] Vidhya S P, Venkatesulu M. A block cipher based on boolean matrices using bit level operations[A]. 13th IEEE International Conference on Computer and Information Science (ICIS)[C]. Tai Yuan: IEEE, 2014. 59 – 63.

[17] Intel Corporation, Intel® 64 and IA-32 Architectures Software Developer's Manual[S]. 2015.

[18] 杭州中天微系统有限公司, CK-Core 用户手册[S]. 2007.

[19] Chang Zhongxiang, Hu Jinshan, Ma Chao. Research on shifter based on ibutterfly network[A]. 17th China Computer Federation[C]. Xi Ning, China: Springer-Verlag, 2013. 92 – 100.

[20] Hilewitz Y. Advanced bit manipulation instructions: Architecture, implementation and applications[D]. Princeton: Princeton University, 2008.

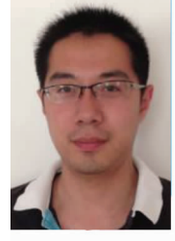
[21] Filer E P, Getzinger T W. Dynamic endian switching[P]. US Paten: 7139905, 2006.

[22] Chen J, Wang Q, Guo Z, et al. A Circuit Design of SMS4 against Chosen Plaintext Attack[A]. 17th International Conference on Computational Intelligence and Security [C]. Paris, France: IEEE, 2015. 371 – 374.

[23] Neki K. Digital data processing circuit equipped with full bit string reverse control circuit and shifter to perform full or partial bit string reverse operation and data shift operation[P]. US Patent: 4984189, 1991.

[24] Semiconductor Manufacturing International Corporation. SMIC 65nm Logic Process Standard Cell Library Databook[S]. 2012.

作者简介



马 超 男,1988 年生于陕西西安. 解放军信息工程大学博士研究生. 研究方向为专用处理器设计,多级动态互连网络路由算法,ASIC 专用芯片设计.
E-mail: wenlu_ma@163.com

戴紫彬 男,1966 年出生于河南商丘. 解放军信息工程大学专用芯片设计教研室主任,博士生导师. 研究方向为密码处理器设计, VLSI 设计.

李 伟(通信作者) 男,1983 年生于天津. 解放军信息工程大学副教授,现为复旦大学国家集成电路重点实验室博士研究生. 研究方向为密码处理器设计,ASIC 专用芯片设计.
E-mail: liwei12@fudan.edu.cn