新型可重构移位-置换单元研究与设计

马 超[1],李 伟[2],戴紫彬[1],冯 晓[1]
(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多级动态网络的自重构特性,提出了一种针对循环移位操作的高速可重构控制信息生成算法,该算法不仅具有极高的并行性和较小的资源消耗,还首次将循环移位、双向循环移位和以$2^i(i=0,1,2\cdots)$为位宽的短字循环移位等10余种不同类型的移位操作统一在一个算法中.并在此基础上,设计了一种新型可重构移位-置换单元.该单元在SMIC 65nm工艺完成了逻辑综合.实验结果表明,当该单元只实现循环移位时,与以往研究成果相比,频率提升了6.4%~12%,面积减小了22%~30%;当该单元实现多种移位操作时,频率下降约8.4%,但能够支持的移位操作类型是以往研究成果的2倍.

关键词: Inverse Butterfly/Butterfly网络; 循环移位算法; 可重构; 短字循环移位
中图分类号: TP331.2 **文献标识码:** A **文章编号:** 0372-2112 (2017)05-1025-10
电子学报URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2017.05.001

A Novel Reconfigurable Rotation-Permutation Unit Research and Implementation

MA Chao[1],LI Wei[2],DAI Zi-bin[1],FENG Xiao[1]
(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, a high speed reconfigurable control bits generation algorithm for rotation shift operations is proposed. The algorithm utilizes self-reconfigurable characteristics of the Inverse Butterfly/Butterfly network, and it is highly parallelized and low-cost. Moreover, it also integrates more than 10 kinds of rotation shift operations together, including rotation shift, bidirectional rotation shift, and sub-word rotation shift based on a bit width of $2^i(i=1,2,\cdots)$. Following this, a new high-speed reconfigurable rotation-permutation unit (HRRU) is developed and synthesized in SMIC 65-nm process. The results show that the proposed unit only containing rotation shift operations, enhances the frequency by 6.4%~12% and reduces the area by 22%~30%, compared with previously proposed solutions. When achieving a variety of rotation shift operations, the frequency of our unit decreases by 8.4%. But it can support twice operations as many as previous designs.

Key words: Inverse Butterfly/Butterfly Network; rotation algorithm; reconfigurable; sub-word rotation operations

1 引言

移位器与算术逻辑单元(Arithmetic Logical Unit)一样作为一种基本功能运算部件被广泛应用于通用处理器中[1].移位器实现的方式很多,例如基于反馈移位寄存器、桶型移位器[2]、对数移位器[3]等,它们在实现固定位宽且单一方向的移位操作时具有较高的性能.然而在密码学[4]、数字图像处理,多媒体处理等众多领域中除了简单的移位操作外还存在着子字并行移位操作以及形式更复杂的比特置换类操作,面向字优化的通用处理器对于这些细粒度比特级置换类操作处理效率很低,需要将其转化成多条指令组合实现[5].因此,亟需找到一种新的架构来适应未来更为复杂的比特移位-置换类操作.

目前,关于提高比特置换在通用处理器中的执行效率有着广泛的研究[6].然而移位操作实质也是一种特殊的比特置换操作,若将这两种操作彼此孤立并各自设计专用硬件单元,必将造成资源的重复开销[7,8].因此,普林斯顿大学的Lee和Hilewitz在文献[9]中提出了一种基于动态多级互连网络-Inverse Butterfly/But-

收稿日期:2016-01-14;修回日期:2016-03-07;责任编辑:蓝红杰
基金项目:国家自然科学基金(No.61404175)

terfly 的移位-置换单元,首次将循环移位(ROTR,ROTL)、逻辑移位(SHL,SHR)等传统移位操作与并行抽取和并行插入(PEX,PDEP)[10]、比特归类(GRP)[11]、块抽取与插入(EXTRR,DEP)[12]、BFLY 和 IBFLY[13]等 10 余种复杂比特级置换操作统一到了一个架构下,并针对循环移位操作提出了专用的控制信息生成算法. 该架构不仅突破了传统移位器功能单一的限制,更将以往移位、置换硬件单元分离的设计方法统一在一起,为可重构移位-置换单元的设计提供了一种新思路.

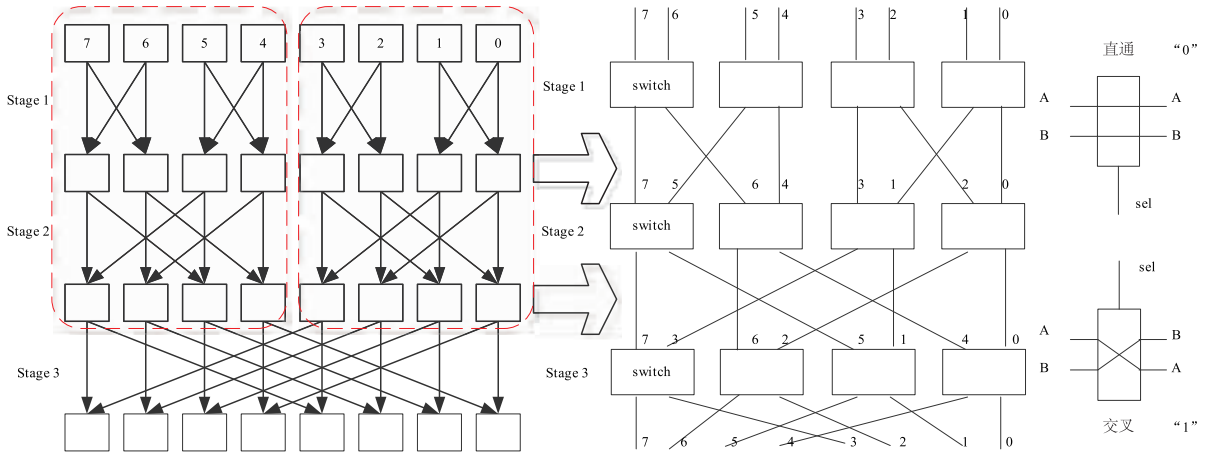
然而,Hilewitz 提出的基于 Inverse Butterfly/ Butterfly 网络的循环移位算法是一种串行迭代式的,硬件实现起来存在关键路径较长、资源消耗过大、处理性能不高等问题. 本实验的 Chang 在文献[14]中也提出了一种基于该网络的循环移位控制信息生成算法,虽然在一定程度上提高了算法并行性,但硬件实现起来资源消耗巨大. 因此,本文通过对该网络拓扑结构的深入分析,提出了一种更为高效的可重构循环移位控制信息生成算法,它不仅具有极高的并行性,而且还将原有该网络能够实现的置换种类扩展了近一倍.

2 Inverse Butterfly/Butterfly 网络结构及应用

动态多级互连网络可以通过改变开关状态实现不同结点之间的连接,使系统具有自重构能力,从而灵活地完成数据的重新排列[15,16]. 图 1 描述了一个 $n=8$ - bit 位宽的 Inverse Butterfly 动态多级互连网络,该网络有 $\log_2(n)$ 级从上到下依次为第一级、第二级和第三级,每一级有 $n/2$ 个 2 输入交叉开关,每一个交叉开关在 1-比特控制信号(Sel)的作用下完成一对输入信号的"交叉"或者"直通". 该网络共有 $\log_2(n)\times n/2$ 个交叉开关,每个交叉开关有两种配置状态,因此能够实现 $2^{\log_2(n)\times n/2}$ 种置换操作. 从图 1(a)的数据流图中还可以看出,该网络具有迭代性和可拆分性. 若把该网络的最后一级舍弃,那么剩下的部分可以看成两个独立的以 $n/2$ 为输入宽度的子蝶网络;Butterfly 网络是 Inverse butterfly 网络的逆结构,数据的分组间距为 $2^{\log_2(n)-i}$,它与 Inverse Butterfly 网络规律相似,在此不再详述.

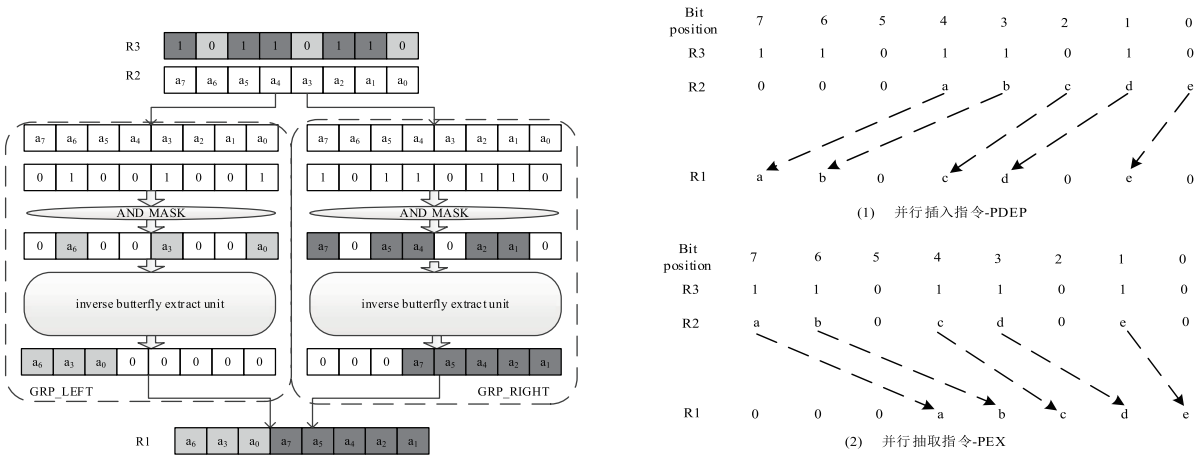
Lee 和 Hilewitz 等人通过研究不同置换操作在动态互连网络实现时各级开关状态的控制信息生成算法,提出了一系列基于 Inverse Butterfly/Butterfly 网络的复杂比特置换指令如 GRP、PEX/PEDP、ROT 等. GRP 指令能够完成比特归类操作,它将控制序列 R3 中为"1"的控制位对应在序列 R2 中的数据并行地抽取出来依次排在目的序列 R1 的最右侧,同时将控制序列 R3 中为"0"的控制位对应在序列 R2 中的数据并行地抽取出来依次排在目的序列 R1 的最左侧,如图 2(a)所示. 由于该指令能够较好地满足密码学中数据的扩散效应,且同时具备较强的抗线性和差分攻击能力,因此被广泛应用于新一代密码算法的设计中[17,18]. PEX 和 PDEP 指令分别完成数据并行抽取与并行插入操作如图 2(b)所示,其中 PEX 指令基于 Inverse Butterfly 网络实现,它是 GRP 操作的一种简化形式,其硬件电路仅包含 GRP_right 这部分. PDEP 指令基于 Butterfly 网络实现,是 PEX 指令的反操作. 由于这两条指令在实际应用中的广泛性,因此 Intel 在 2013 年发布的 HASWELL 处理器的指令集中已将它们集成[19].

ROT 指令是一种常用的循环移位操作指令,其硬件结构一直未有太大突破. Hilewitz 在文献[9]中首次提出了基于 Inverse Butterfly 的循环移位控制信息生成算法,将传统移位操作与复杂比特级操作的硬件功能单元统一在了一个架构下,极大地扩展了 Inverse Butterfly 网络的灵活性. 然而,该算法是一种递归算法,各级控制信息必须依次串行生成,因此算法并行度很低,硬件执行较为耗时. 且当网络级数增加时算法复杂度会急剧上升,硬件实现将变得十分困难. 本实验室的



(a) Inverse Butterfly 数据流结构 (b) Inverse Butterfly 拓扑结构

图1 Inverse Butterfly网络数据与结构流图



(*a*) GRP指令操作示意　　(*b*) PDEP/PEX指令操作

图2　Inverse Butterfly/Butterfly网络应用

Dr. Chang 基于该网络提出了一种具有并行化特征的循环移位控制信息生成算法. 该算法根据移位位数 S 和网络各级中的初始控制信息这两参数,并行地计算各级开关中的控制信息,有效提升了循环移位的处理性能,但算法的核心运算本身就是一种循环移位操作,因此硬件资源消耗较大.

本文通过研究数据在 Inverse Butterfly 网络中各级的流向规律,提出了一种可重构循环移位控制信息生成算法. 它不仅支持双向循环移位操作,还能够支持以 $2^i(i=1,2,\cdots,\log_2(n))$ 为位宽的短字循环移位操作. 同时,与以往文献相比该算法还兼具低复杂度和高并行度两方面的优势.

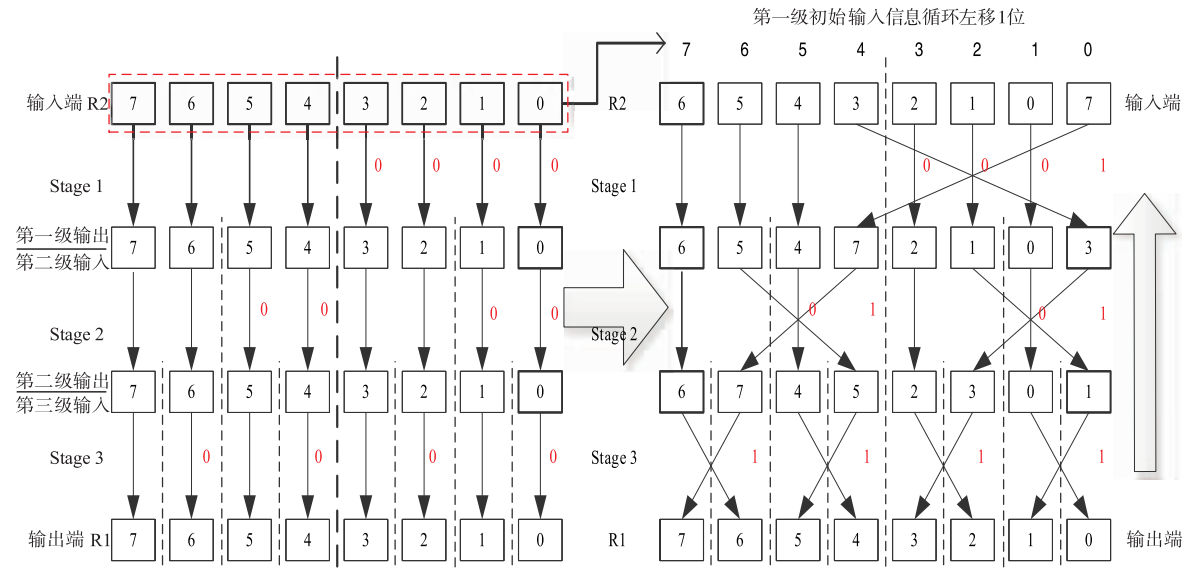
3　可重构循环移位算法研究及电路设计

3.1　Inverse Butterfly/Butterfly 网络的基本性质

快速高效地生成网络各级开关所需的控制信息以完成循环移位操作是本文的研究重点. 下面从一个实例出发阐述循环移位操作在 Inverse Butterfly/Butterfly 网络中的实现机理. 如图 3(*a*)所示,当网络各级初始控制信息均为"0"时,各级输出数据与初始输入数据位置相同,没有进行移位操作. 当初始输入数据循环左移 1 位且适当调整各级初始控制信息后,该网络便重构成为 1-bit 循环右移电路,图 3(*b*)所示;若数据从图 3(*b*)的输出端往输入端流动,该网络便成为一个能够实现 1-bit 循环左移的 Inverse Butterfly 电路.

性质 1　Butterfly 网络各级数据不移位或者以 $2^{\log_2(n)-i}$ 为间隔移位,其中 n 为网络数据位宽,i 为级数 $(1\leqslant i\leqslant\log_2(n))$.

根据性质 1 可以得出数据在网络各级位置变动的函数 C_i:



(*a*) 未进行循环移位电路　　(*b*) 循环右移1位电路

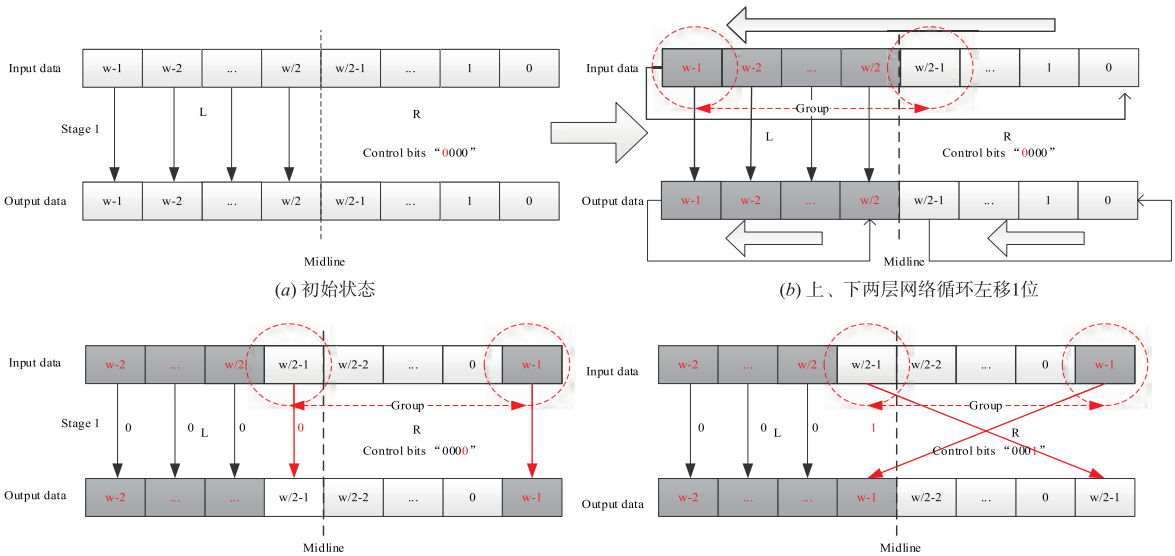
图3　循环右移1比特控制信息适配

$$C_i(x_{\log_2(n)-1}x_{\log_2(n)-2}\cdots x_1x_0)=\\ x_{\log_2(n)-1}x_{\log_2(n)-2}\cdots\overline{x_{\log_2(n)-i}}\cdots x_1x_0 \text{ or}\\ x_{\log_2(n)-1}x_{\log_2(n)-2}\cdots\overline{x_{\log_2(n)-i}}\cdots x_1x_0 \quad (1)$$

其中 C_i 为位置改变函数，$x_{\log_2(n)-1}\cdots x_1x_0$ 为输入数据的二进制地址，这样的函数和文献[20]中的方体置换表达式类似，输入数据的二进制地址序列有且只有一级网络与其对应，那么当某一数据通过 $\log_2(n)$ 级网络后便可以路由到任何位置且路径唯一.

性质 2 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4(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4(b)中左半部分的初始输入数据(上层灰色方块部分)中最高位($w-1$)将被移位到右边最低位，同时右半部分中的最高位数据($w/2-1$)将越过网络的中间线到左边的最低位，其余数据仅左移一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数据对($w-1$)和($w/2-1$)称为一个特殊对，如图4(b)中虚线圆框所示. 它们不仅被同一个控制信号控制且在循环移位时它们都将跨越网络边界，到达对方初始所属的部分中. 若初始控制信息序列也随着循环左移1位从“00…00”→“00…00”，那么除特殊对以外的初始输出数据将继续停留在原本的部分中，而特殊对的输出数据将对调位置，如图4(c)所示. 与图4(b)的初始输出数据相比，仅有特殊对彼此调换了原有所属的部分，其余的初始输出数据仅仅是左移1位. 由性质1可知，处于 i 级的输入数据能够完成间隔距离为 $w/2=2^{\log_2(n)-i}$ 的移位或者不移位. 那么若将图4(b)中特殊对所对应的控制信息取反即“0001”，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后，各自部分内循环左移1位的结果如图4(d)所示. 当特殊对初始状态为“1”交叉时，与直通“0”情况相似，不在单独解释. 若该网络再完成一次1-bit循环左移时，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 s 位，同时保证每1次移位后末位控制信息取反，那么其输出数据的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2成立.



(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4 直通状态下特殊对移位规律

3.2 并行循环移位算法研究

(1)并行循环移位算法原理

一个 n-bit 位宽的 Inverse Butterfly 网络将最后一级网络去掉或者设置为“直通”，即可重构为左、右两个由 $\log_2(n)-1$ 级组成的子蝶网络，左、右两边 $n/2$-bit 的初始输入数据在 $\log_2(n)-1$ 级以前相互独立，不存在交互通路(见图1(a)所示). 若在位宽为 n-bit 的网络中实现循环移 S 位，左、右两个以 $n/2$ 为移位宽度的子蝶网络必须在最后一级($\log_2(n)-1$)各自完成其输入数据的 $S'=S \bmod n/2$ 的循环移位. 由于 Inverse Butterfly 网络的迭代型结构，因此可将这个规律可以推广到网络各级中. 如在第 i 级共有 $n/2^i$ 个子蝶网络，每个子蝶网络的位宽为 2^i，那么 n-bit 初始数据在第 i 级将被分割为 $n/2^i$ 个独立的部分，每个部分完成 $S'=S \bmod 2^i$ 的循环移位. 那么，初始输入数据在 Inverse Butterfly 网络中完成 S 位循环移位时有如下性质：

性质 3　n-bit Inverse Butterfly 网络在完成 S 位循环移位操作时,第 i 级各子蝶网络将分别实现输入数据的 $S' = S \bmod 2^i$ 位循环移位.

性质 3 中无论 S 取何值,在位宽为 2^i 的子蝶网络中数据循环移位 S 与循环移位 $S' = S \bmod 2^i$ 的结果是一样的. 因此可将性质 3 改写如下:

性质 4　n-bit Inverse Butterfly 网络在完成 S 位循环移位操作时,各级子蝶网络(位宽为 2^i)都将实现其内部数据的 S 位循环移位.

若 Inverse Butterfly 网络实现数据循环左移 S 位,那么在各级控制信息不变的情况下 Butterfly 网络能够将左移的数据还原,实现数据右移 S 位如图 5 所示. 图中 Butterfly 网络各级数据相对于初始各级数据(initial input data)均进行了各自位宽的 S 位循环移位. 当 n-bit Butterfly 网络中各级数据均与初始输入数据相同时(各级控制信息全为"0"),我们称该网络为本源 Butterfly 网络(Original Butterfly Network),如图 1(a)所示. 根据性质 4 可知,若完成基于 Butterfly 网络的循环右移 S 位操作,只需将本源网络中各级的输入数据(他们均等于 initial input data)按照各自的位宽并行循环左移 S 位即可. 同时根据性质 2 知,只需适当地调整本源网络中各级的初始控制信息,那么该网络即可完成输入数据循环右移 S 位的操作.

以 8-bit 位宽的本源 Butterfly 网络循环右移 1 位为例,对该过程进行详细解释. 图 6 中(a)、(b)、(c)三个子图中的上半部分为本源网络拆分后的各级输入、输出以及控制信息的初始状态(全为"0"),其中灰底方块表示该级所在的特殊对. 当完成循环右移 $S = 1$ 时,被拆分成 3 级子网络的初始输入数据将并行地按照各自子蝶网络的位宽进行循环左移,整个循环移位过程如图 6 中三个子图下半部分所示. 图 6(a)中第一级 8-bit 初始输入数据循环左移 1 位后为(65432107),根据性质 2 将初始控制序列"0000"循环左移后最低位取反"0001",那么该级的输出数据则为(6547)(2103);该输出数据正好为第二级子蝶网络输入数据循环左移 1 移位的结果. 依次递推,网络第二级输出数据则正好为第三级子蝶网络输入数据循环左移移位的结果. 因此将图 6(a),(b),(c)下半部分网络提取出来并首尾相接即可完成基于 Butterfly 网络的循环右移 1 操作,如图 6(d)所示.

本文将上述过程提炼并归纳成一种基于 Butterfly 网络循环右移操作并行控制信息生成算法,如算法 1 所示.

算法 1　**To generate the parallel control bits from S for right rotation operations**

Input:S;　//rotation bit
Output:Control bits;　//the lg(n) * n/2 control bits
(1)for($i = 1; i <= lg(n), i++$)　//parallelism
{
$J = n/2^{i-1}$;//the input with of sub-butterfly network at stage i
$M = J/2$;　//the number of control bits at stage i for each sub-network
$K = 2^{i-1}$;　//the number of sub-butterfly network at stage i
Control_bits(i) = RLTR(0^M,S)^k//RLAR(0^M,S) indicate a bit-string of M zeros will be rotation left than reverse the last significant bit. RLTR(0^M,S)^k means the bit-string RLAR(0^M,S) will be duplicated k times
}
(2)RLTR(0^M,S)
For($a = 0; a <= s, a++$)
{
C = 00…00 < < <1　//M zero string left rotation one time
$C \sim^L$//　"$\sim^L$" Reverse the least significant bit of the rotated string C
}

(2)并行循环移位算法优化

(a)算法第一部分优化

算法第一个部分的循环体中,J、M、K 这三个参数与网络拓扑结构相关,都能够根据不同的级数 $i(1 \leqslant i \leqslant \log_2(n))$提前预计算出来,唯一变换是各级的控制信息生成函数即Control_bits(i) = RLTR(0^M,S)^k,其中k表

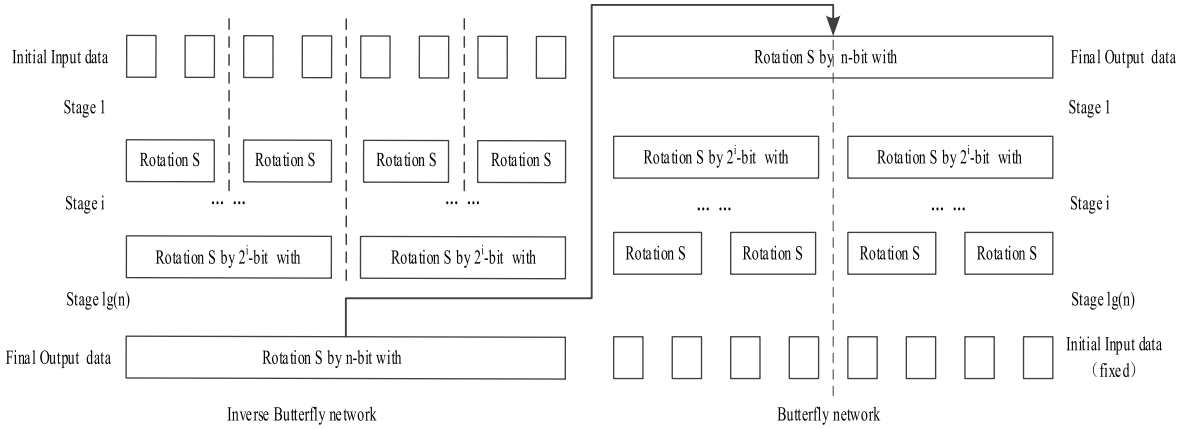


图5　循环移S位数据流模型

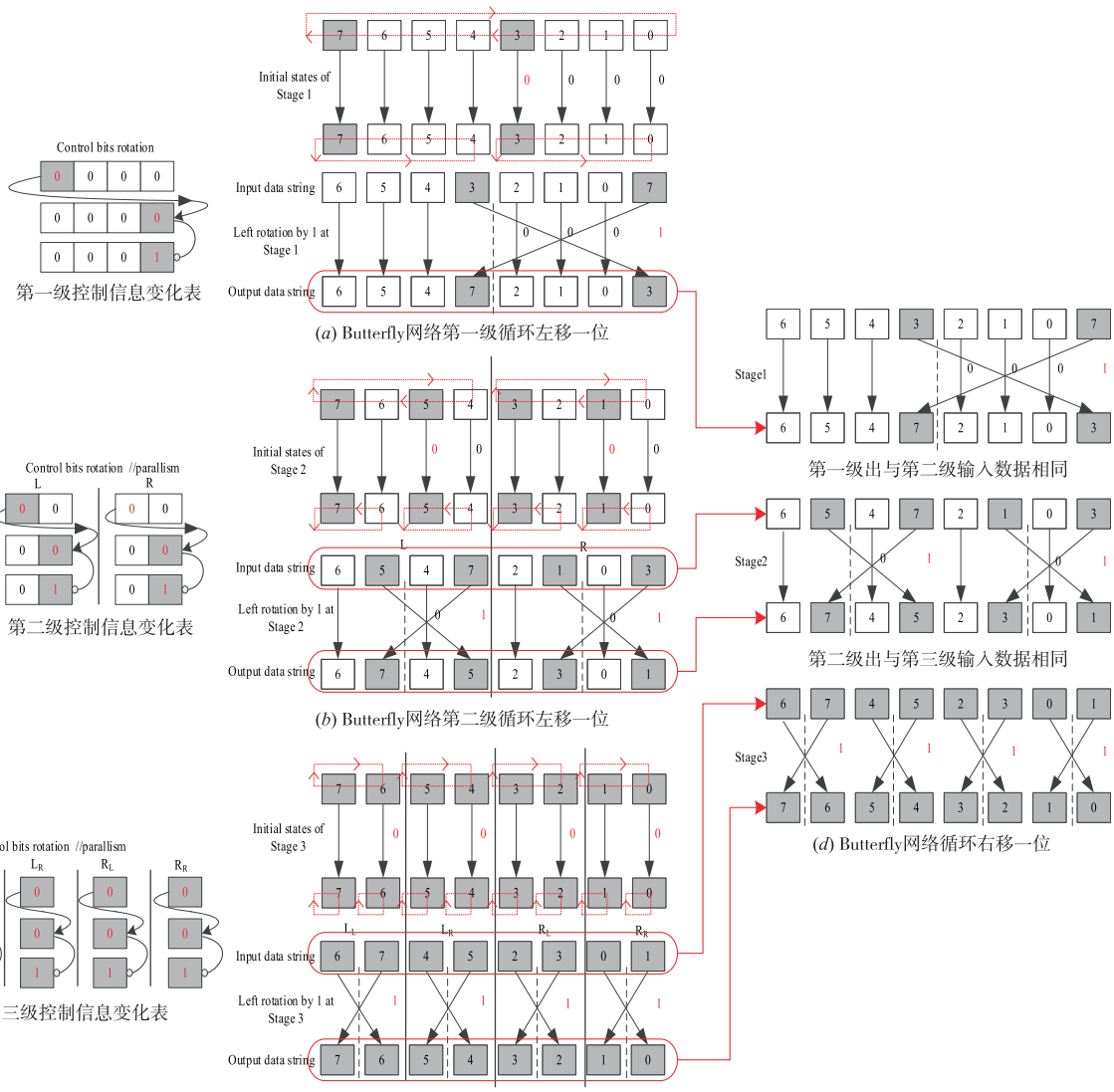


图6 Butterfly网络循环右移1位各级控制信息状态

示 RLTR 函数将要被并行地执行的次数. 如 $n=8$-bit Butterfly 网络的第 $i=3$ 级有:$M=1$,$K=4$. 那么,该级 RLTR(0^1,S)将被并行地执行 4 次以生成其所需要的 4-bit 控制信息,这也表明硬件层面就必须有 4 套相对应的控制信息生成电路作为算法并行实现的基础. 由性质 4 可知,数据在各级子 Butterfly 网络中移位的形式是相同的,那么同级中各个子蝶网络控制信息也必然相同. 因此同一级的所有子蝶网络可以共享一套控制信息生成电路. n-bit Butterfly 电路第一级子蝶网络有 1 个,需要 $n/2$ 比特控制信息;第 i 级有 2^{i-1} 个子蝶网络,需要 $n/2^i$ 比特控制信息;当 $i=\log_2(n)$ 时,整个网络所需的控制信息量为:$T_{control-bits}=n-1$. 若 n = 64-bit 时,其生成的控制信息只有 63-bit,这与算法 1 中生成的 196-bit 控制信息相比,减少近 68%.

(*b*)算法第二部分优化

算法第二个部分仅由 RLTR(0^M,S)函数构成. 它将一个全零的初始序列进行循环移位后最低位取反 S 次以生成最终各级的控制信息. 通过研究发现,一个 M 比特的全零序列进行 RLTR 操作周期是 $2M=n/2^{i-1}$,这与该级输入端子蝶网络的位宽 J 相等. 因此在生成各级控制信息时,初始控制信息不必都进行 S 次 RLTR 操作,它们根据不同的级数仅需进行 $S'=S \bmod J=S \bmod n/2^{i-1}$次操作即可. 例如当 $i=\log_2(n)$ 时,$S'=S \bmod 2$,那么 RLTR 的次数仅由 S 的二进制数 $S_{\log_2(n)-1}\cdots S_1S_0$ 最低位 S_0 直接决定,这样便极大地简化了 RLTR 在各级操作的次数,有效提升了算法的硬件实现速度.

3.3 循环移位算法功能扩展

(1)双向循环移位算法统一

循环移位包括循环左移和循环右移操作,上文仅对基于 Butterfly 网络的循环右移操作控制信息的生成

算法进行了研究. 我们可以采用同样的分析方法得出循环左移操作控制信息在各级的生成规律:

性质 5　循环左移 s 位的适配于 Butterfly 网络各级的控制信息是循环右移 s 位控制信息的逆序

结合性质 5 以及 3.2 节中对算法的优化,将算法 1 改进如算法 2 所示:

算法 2　(算法 1 的改进):**To generate the parallel control bits from S for bi-rotation operations**

```
Input:S;  //rotation bit
Input:enable_right  //to define the right rotations or left rotations1
Output:Control bits;  //the log2(n) * n/2 control bits
(1)for(i=1,i<=lg(n),i++)  //parallism
{
J=n/2^(i-1);  //the input with of sub-butterfly network at stage i
M=J/2;  //the number of control bits at stage i for each sub-network
K=2^(i-1);  //the number of sub-butterfly network at stage i
If(enable_right==1)
Control_bits(i)=RLTR(0^M,S');
Else
Control_bits(i)=~RLTR(0^M,S');  //~means the inversion string of RLTR
}
(2)RLTR(0^M,S')
S'=S mod n/2^(i-1)
For(a=0,a<=S',a++)
{
C=00…00<<<1  //M zero string left rotation one times
C^-1//"~^1" reverse the least significant bit of the rotated string C
}
```

分别对上述两种算法的时间复杂度进行分析有:算法 1 循环体中最为耗时的操作为 $\mathrm{RLTR}(0^M,S)^k$,该函数在算法第一部分中被执行 $n-1$ 次,在第二部分中最坏情况下(移位位数 $S=n$ 时),被执行 n 次. 因此整个算法的时间复杂度为:$T1(n)=O(n^2)$,属于 2 阶时间复杂度. 改进型算法 1 中,由于没有了指数 k 这个参数,同时对移位位数 S 根据级数 i 的不同又做了相应地"模"处理,因此整个算法的时间复杂度下降为:$T2(n)=O(n\log_2 n)$,属于线性对数复杂度. 因此,改进后算法的时间复杂度大幅下降,执行效率更高.

若将 Butterfly 网络中循环右移 S 位的控制信息直接应用到 Inverse Butterfly 网络中即可实现输入数据向左循环移位 S. 又根据性质 5 知,Butterfly 网络中循环左移 S 位的各级控制信息是循环右移的逆序,若将该逆序后的控制信息作用于 Inverse Butterfly 网络即可实现循环右移 S 位操作. 因此,将 Inverse Butterfly 和 Butterfly 网络关于循环左、右移位操作各级控制信息的规律总结如下:

表 1　Butterfly 与 Inverse Butterfly 循环移位控制信息规律

Network \ Operations	Left Rotation S	Right Rotation S
Butterfly	~RLTR(0^M,S')	RLTR(0^M,S')
Inverse Butterfly	RLTR(0^M,S')	~RLTR(0^M,S')

(2)短字循环移位算法统一

短字循环移位实际上是循环移位操作的一种分裂操作模式. 它将输入的 n 比特数据分为 $k=n/m$ 个短字(n、m 为 2 的幂指数且 $n>m$),可以并行执行 k 个短字内部的循环移位操作. 这样的操作非常适合在 Inverse Butterfly/Butterfly 网络中实现,这与该网络迭代性和可拆分性是分不开的. 输入数据分成了 k 个组,每个组有 m-bit 数据,要完成 m-bit 数据的循环移位只需要将前 $\log_2(m)$ 级的 Inverse Butterfly 网络中的开关状态按照改进型算法 1 适配即可,而剩余 $\log_2(n)$-$\log_2(m)$ 级的开关只需保持"直通"即所有剩余的开关信息直接配置成"0",以保证各子蝶网络中的数据不会置换到其他子蝶网络中,从而完成短字循环移位操作.

3.4　可重构移位架构设计

本节根据改进型算法 1,设计了基于 Inverse Butterfly 网络的高速可重构循环移位-置换单元(High-Speed Reconfigurable Rotation Unit,HRRU),该单元支持双向循环移位和以 2^i($i=1,2,\cdots,5$)的短字循环移位操作,还可以通过增加新的控制信息生成电路完成复杂比特置换操作. 它由可重构循环移位控制信息生成电路和 Inverse Butterfly 互连网络电路组成,如图 7 所示. 其中可重构控制信息生成电路是整个单元的核心模块,它根据移位位数(shamt)和执行模式的不同实时地生成用于 Inverse Butterfly 网络中各级开关的控制信息. 其端口功能如表 2 所示.

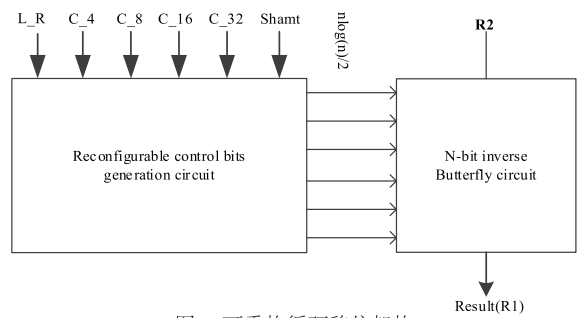


图7　可重构循环移位架构

该电路负责完成 $\mathrm{RLTR}(0^M,S')$ 函数操作. 当 $M=4$ 时,初始序列为"0000",若 S' 从 0 - 7 变化,RLTR 函数运算结果如表 3 所示.

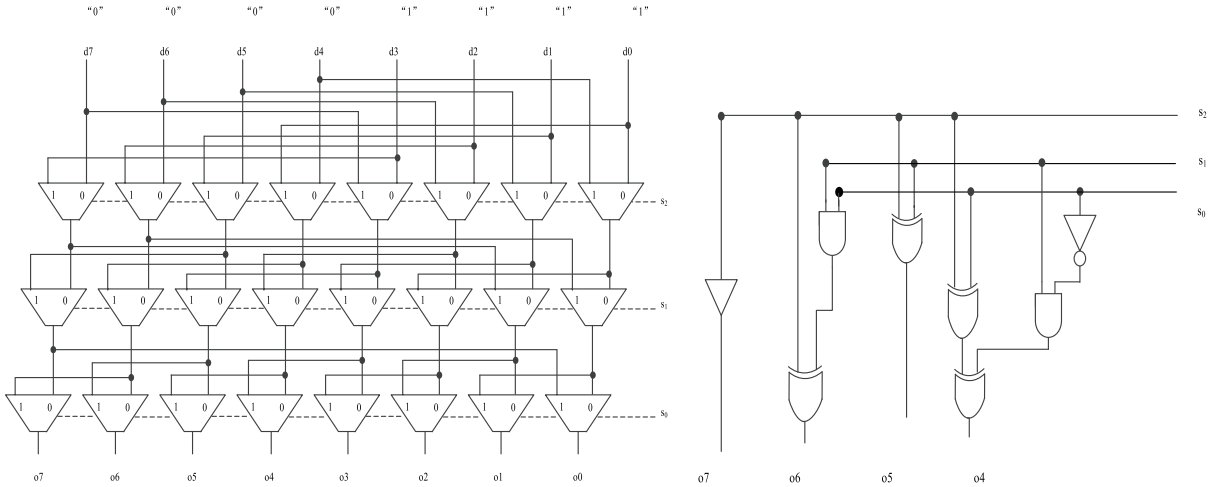
表 2 可重构循环移位单元端口列表

MODE FUNC {C_4,C_8,C_16,C_32}	L_R	
	0	1
0000	64bit Rotation Right	64bit Rotation Left
1xxx	4bit Rotation Right	4bit Rotation Left
01xx	8bit Rotation Right	8bit Rotation Left
001x	16bit Rotation Right	16bit Rotation Left
0001	32bit Rotation Right	32bit Rotation Left

表 3 $M=4$,RLTR 变化规律

S'	0	1	2	3	4	5	6	7
RLTR	0000	0001	0011	0111	1111	1110	1100	1000

从表 3 中 RLTR 随 s' 变化的结果可以发现,它与 8-bit 序列“0000_1111”循环移位后取高四位输出值的结果相同,因此我们将 RLTR 操作首先用对数移位器初步实现,然后再进行结构布尔优化. 对数循环移位器结构如图 8(a)所示,它基于二选一数据选择器实现,一个 N-bit 的输入数据能够在每一级按照 2 的幂指数进行移位. 该结构优点是电路无需译码,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. 而 RLTR 操作初始输入值为恒定值,因此可以对恒定值输入的对数循环移位器进行布尔逻辑优化以精简冗余逻辑,其优化后结构如图 8(b)所示. 与传统结构相比,经过优化后的电路面积大幅减少且速度有所提升.



(a) RLTR 优化前电路 (b) RLTR 优化后电路

图8 8-bit RLTR优化前(左)后(右)电路结构

4 功能与性能评估

为了客观的与以往文献进行对比,本文将 HRRU 单元位宽选定为 64-bit,并分别从功能和性能角度出发对提出的单元进行全面的分析和评估.

(1)HRRU 功能性分析

表 4 中列举了五种不同移位单元所能够支持的移位操作种类.

结果表明:对数移位器由于结构的限制,只能实现单一方向的移位操作. Hilewitz 与 Chang 设计的移位器和本文设计的 HRRU 移位器由于都基于具有自重构特性的 Inverse Butterfly/Butterfly 网络,因此不仅能够实现循环移位、双向循环移位操作,而且还可以通过增加额外的控制信息生成电路实现包括 GRP、PEX/PDEP 等在内的多种复杂比特置换操作,具有较强的灵活性. 此外,本文设计的 HRRU 还能够利用同一个算法(改进型算法 1)实现多种短字循环移位类操作. 因此从功能性角度出发,HRRU 单元相比于其他设计有着更强的重构能力.

表 4 不同移位单元支持的功能

移位操作 / 功能单元	GRP 64-bit	PEX/PDEP 64-bit	IBFLY/BFLY 64-bit	Bi-Rotation 64-bit	Bi-Subword Rotation (32/16/8/4)-bit
Log Rotation Shifter				√ –	
Hilewitz's Shifter[8]	√ *	√ *	√ *	√	
Chang' Shifter[14]	√ *	√ *	√ *	√	
Our Basic HRRU	√ *	√ *	√ *	√	
Our HRRU	√ *	√ *	√ *	√	√

– 代表该单元只能支持单一方向的移位; * 代表通过增加新的控制逻辑,移位器也能够支持的操作;Our Basic HRRU 是 HRRU 的功能裁减版,它与 Hilewitz's 和 Chang 设计的移位单元功能完全相同

(2)HRRU 性能分析

本文首先对提出的 HRRU 使用 NC-Verilog 对其功能正确性进行了仿真验证. 然后,基于 SMIC 65-nm 工艺[21]在最慢工艺角(ss)、最低温度(-40℃)和最低电压(1.08v)下,进行了逻辑综合与优化,综合时采用 flatten 的优化策略并设置时序优先,其结果如表 5 所示:

表 5 中需要特别说明的是 Hilewitz's shifter 采用 TSMC 90nm 工艺综合优化,那么通过比较绝对面积值和绝对延迟值意义并不大. 他在文献[9]中将设计的单元与通用结构的对数移位器进行了性能对比,并将面积和延迟两个参数归一化处理,得出了相对面积和相对延迟值. 这些相对值突破了传统工艺的限制,能够在不同工艺下实现数据的有效对比. 因此本文将提出的可重构移位-置换单元 HRRU 与本实验室 Chang 设计的移位单元和通用结构的对数移位器一起进行了代码编写. 然后在相同环境约束下进行了代码综合,并将综合结果同样做了归一化处理. 实验结果表明:本文设计的 Basic HRRU 虽然功能与 Hilewitz's Shifter 和 Chang's Shifter 相同,但是由于采用了并行化的控制信息生成算法,并在硬件层面对算法关键模块进行了布尔优化,因此相对面积(Relative Area)为对数移位器的 1.08x,相对延迟(Relative Latency)为它的 1.04x. 与文献[9]中的相对面积 1.38x 和相对延迟 1.18x 相比,本文提出的 Basic HRRU 面积相对值减小了 22%,延迟相对值缩小了 12%. 与文献[14]相比,其面积相对值减小了 30%,延迟相对值减小了 6.4%. 通过进一步分析还可以发现:当除去 Basic HRRU 和对数移位器中的 Buffer 面积后,剩余部分的面积仅比对数循环移位器的面积略大为 59.44um^2,这说明本文提出的控制信息生成算法的硬件实现简洁,资源占用较少. HRRU 单元虽然相对面积值略大为 1.62x,但相对速度介于 Hilewitz's shifter 和 Chang's Shifter 之间为 1.13x,更为重要的是它所支持的移位操作是前两种单元的 2 倍,具有很强的可重构能力.

表 5　可重构移位操作单元性能比较

功能单元	Area (NAND gates)	Total Area (um^2)	Relative Area	Buffer Area	Latency (ns)	Relative Latency	f (GHz)
Log Rotation	1.30k	1875.32	1	162.64	0.53	1	1.89
Hilewitz's[9]	–	–	1.38	–	–	1.18	–
Chang's[14]	2.02k	2906.75	1.55	608.52	0.58	1.11	1.72
Our Basic HRRU	1.40k	2017.88	1.08	245.76	0.55	1.04	1.82
Our HRRU	2.11k	3038.40	1.62	435.96	0.60	1.13	1.67

* Hilewitz's Shifter[9] 基于 Inverse Butterfly 网络设计实现,综合时使用 TSMC 90nm 工艺库

* Chang's Shifter[14] 基于 Inverse Butterfly 网络设计实现,采用与本文相同环境进行综合

5　结束语

本文首先提出了一种基于 Inverse Butterfly/Butterfly 网络的具有高并行度的循环移位控制信息生成算法,然后设计了一种新型可重构移位-置换单元 HRRU,当该单元仅包含循环移位操作时,与以往文献相比其速度提升了 6.4% ~12%,而面积减小了 22% ~30%. 当该单元包含多种移位操作时,其频率略有下降约 8.4%,但能够支持的移位操作种类是以往研究成果的 2 倍. 同时,还可以通过在电路中增加新的算法生成电路来支持如 GPR,PEX,BFLY/IBFLY 等复杂比特置换操作,因此该单元具很强的重构能力和广阔的拓展空间,能够为传统单一模式移位器的发展提供了一种新的思路. 我们希望未来能将此单元的控制信息生成电路进行全定制设计,并与 Intel Haswell 处理器中的 PEX 单元充分融合,开辟循环移位-置换架构的新领域.

参考文献

[1] Trived P, Tripathi R P. Design and analysis of 16 bit RISC processor using low power pipelining[A]. IEEE International Conference on Computing, Communication & Automation(ICCCA)[C]. India: IEEE, 2015. 1294 – 1297.

[2] Khan L. Designing of low power high speed 32-bit barrel shifter for ARM7 processor[J]. International Journal of Advanced Research in Computer Science, 2013, 4(3): 108 – 113.

[3] Hosseininia N, Boroumand S, Haghparast M. Novel nanometric reversible low power bidirectional universal logarithmic barrel shifter with overflow and zero flags[J]. Journal of Circuits System and Computers, 2015, 24(04): 1 – 14.

[4] 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.

[5] Ao T, He Z, Dai K. Low-cost bit permutation circuit with concise configuration rule[A]. Proceedings of the International MultiConference of Engineers and Computer Scien-

tists[C]. Hong Kong: International Association of Engineers, 2015. 158 – 160.
[6] Kolay S, Khurana S. PERMS: a bit permutation instruction for accelerating software cryptography[A]. 16th Euromicro Conference on Digital System Design[C]. Spain: IEEE, 2013: 963 – 968.
[7] Sayilar G, Chiou D. Cryptoraptor: high throughput reconfigurable cryptographic processor[A]. IEEE International Conference on Computer-Aided Design(ICCAD)[C]. San Jose: IEEE, 2014. 155 – 161.
[8] 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.
[9] 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.
[10] 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.
[11] 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.
[12] Intel Corporation, Intel Itanium Architecture Software Developer's Manual[S]. 2006.
[13] 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.
[14] 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.
[15] Nassimi D, Sahni S A. Self_routing benes network and parallel permutation algorithm[J]. IEEE Transaction on Computers, 1981, 30(5): 332 – 340.
[16] 戴浩, 沈孝钧. 在 7 级混洗交换网络中实现 16 × 16 的可重排性[J]. 电子学报, 2007, 35(10): 1875 – 1885.
Dai Hao, Shen Xiaojun. Rearrageability of the 7-stage 16 × 16 shuffe exchange network[J]. Acta Electronica Sinica, 2007, 35(10): 1875 – 1885. (in Chinese)
[17] 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.
[18] 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.
[19] Intel Corporation. Haswell Intel Architecture Software Developer's Manual[S]. 2014.
[20] Rajkumar S, Goyal N K. Design of 4-disjoint gamma interconnection network layouts and reliability analysis of gamma interconnection Networks[J]. Journal of Supercomputing, 2014, 69(1): 468 – 491.
[21] Semiconductor Manufacturing International Corporation. SMIC 55nm 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, ASIC 专用芯片设计.
E-mail: wenlu_ma@163.com

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

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

冯 晓 女, 1987 年出生于河北衡水. 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, ASIC 专用芯片设计.