面向密码算法的大位宽比特置换操作高速实现方案

戴紫彬[①] 马 超*[①] 李 伟[②] 南龙梅[②]
[①](解放军信息工程大学 郑州 450001)
[②](复旦大学专用集成电路与系统国家重点实验室 上海 200433)

摘 要：针对面向字级优化的通用处理器，在应对密码算法中大位宽比特置换操作时效率较低的问题，该文提出 $2N$-$2N$ 和 kN-kN($k>2$)的大位宽比特置换操作高速实现方案。并针对方案中涉及的比特提取和比特提取-移位两种操作，分别提出专用扩展指令 BEX, BEX-ROT。在此基础上，对专用指令硬件架构的高效设计进行研究，提出一种基于 Inverse Butterfly 网络统一硬件架构-RERS(Reconfigurable Extract and Rotation Shifter)及相应可重构路由算法，以最大限度地共享硬件资源，减小电路面积。实验结果表明，所提方案能够将处理器架构执行大位宽比特置换操作的指令条数缩减约 10 倍，大幅提升其处理效率。同时，由专用指令所带来的硬件资源开销和延迟开销均较低，不会影响到原架构正常工作频率。

关键词：比特置换；$2N$-$2N$ 置换；kN-kN 置换；专用指令；路由算法；硬件架构
中图分类号： TP393 **文献标识码**： A **文章编号**：1009-5896(2017)09-2119-08
DOI: 10.11999/JEIT161285

Wide-width Bit Permutation Instructions for Accelerating Cryptographic Algorithms

DAI Zibin[①] MA Chao[①] LI Wei[②] NAN Longmei[②]
[①](*The PLA Information Engineering University*, *Zhengzhou* 450001, *China*)
[②](*State Key Laboratory of ASIC and System*, *Fudan University*, *Shanghai* 200433, *China*)

Abstract: Wide-width bit permutation is a very commonly used operation in symmetric cryptographic algorithms. However, current word-oriented general microprocessors are inefficient to cope with the complex bit-level permutation operations. To solve this problem, two schemes for $2N$-$2N$ and kN-kN permutations are proposed respectively, including two extended instructions BEX and BEX-ROT. Furthermore, the efficient hardware implementation of the instructions are studied, and then a unified hardware circuit named RERS (Reconfigurable Extract and Rotation Shifter) is proposed with a corresponding reconfigurable routing algorithm. The RERS can share hardware resources to achieve the purpose of reducing area. The experimental results show that the proposed schemes can truly decrease the number of instructions for accomplishing an arbitrary wide-width bit permutation (instructions reduced by 10 times), which greatly accelerate the performance of microprocessors. At the same time, the overhead of hardware resources and delay caused by the two extended instructions is very low, which will not affect the normal operating frequency of the original microprocessors.

Key words: Bit permutation; $2N$-$2N$ permutation; kN-kN permutation; Extended instructions; Routing algorithm; Hardware circuit

1 引言

比特置换操作在密码学、图像处理、数字信号处理等领域有着广泛的应用[1–4]。特别地，在密码学中，比特置换操作作为实现“扩散(diffusion)”密码特性的有效途径，被用于如 DES, Serpent 等多数密码算法中[5]。然而，面向字级优化的通用处理器，对于这种细粒度比特级操作处理效率很低，需要将其转化成多条基本指令组合实现，这极大地影响了密码算法的处理性能，成为了制约数据安全高速传送的瓶颈[6]。针对上述问题，近年来人们提出了一系列加速比特置换操作在通用处理器执行效率的专用指令。它们均能够在 $\lg N$ 条指令内完成一次 N bit 任意置换操作，其中 N 为通用处理器处理位宽，常为 32 bit 或者 64 bit。

收稿日期：2016-11-25；改回日期：2017-06-05；网络出版：2017-06-19
*通信作者：马超 wenlu_ma@163.com
基金项目：国家自然科学基金(61404175)
Foundation Item: The National Natural Science Foundation of China (61404175)

然而，在密码领域的实际应用中还存在着很多大位宽比特置换操作，如一些密码算法为了提升自身的密码强度，使用到了 128-128，甚至 256-256 的比特置换操作。因此，如何在 32 bit 或者 64 bit 通用处理器中高效地执行这些大位宽比特置换操作，成为了一个不可忽略的问题。基于此，本文针对大位宽比特置换操作，在通用处理器中高效实现的问题进行了详细的研究。提出了两条专用置换指令 BEX 和 BEX-ROT，并设计了对应的统一硬件架构，将 kN-kN($k \geq 2$)的大位宽比特置换指令条数从 $4Nk$ 降低为 $(2k+\lg N-1)\times k$ 。同时，实验结果也表明，本文提出的方案能够以较小的硬件资源开销，实现大位宽比特置换操作指令条数约 10 倍的下降，从而大幅提升其处理性能。

2 需求分析及研究现状

比特置换操作能够有效地隐藏明文与密文之间的相互关系，具有良好的位级“扩散”特性。因此，被广泛应用于密码算法的设计中。表 1 中总结了一些常用密码算法所使用到的置换操作及其处理位宽。

表 1 典型密码算法中的置换操作

算法	置换位宽(bit)
DES/3DES(P)*	32-32
PICCOLO[7]	64-64
PRESENT[8]	64-64
CRYPTON[9]	128-128
SERPENT-128/192/256[10]	128-128
PUFFIN[11]	128-128

*指 DES 中轮运算所使用的置换 P 位宽为 32 bit

对表 1 分析可知，密码算法所使用的置换操作位宽灵活多变从 32 bit 到 128 bit 不等，且以 128 bit 的大位宽比特置换为主。未来，随着密码技术和计算机处理能力的发展，比特置换操作还很有可能向着更大位宽的方向发展。目前国内外针对如何提高比特置换操作在通用处理器中的执行效率，提出了一系列专用置换指令如 PPERM3R, CROSS, OMFLIP, BFLY & IBFLY, PERMS 以及 GRP[12−14]。但它们仅适合 N-N 及 $2N$-$2N$ 类型置换的高效实现，表 2 中总结了上述指令的功能及其硬件实现代价。

表 2 中，PPERM3R 指令基于 Crossbar 全选择器实现，Nbit 被置换数据中的每 1 bit 由 $\lg N$ bit 索引确定，一共需要($N\times\lg N$) bit 索引。由于处理器指令结构本身的限制，仅能提供两个源寄存器，一个用于存储被置换数据，另一个作为索引值存储寄存器。因此若要完成一个 N bit 的任意置换操作一共需要 Num_ISA = $\lceil N/A \rceil$ 条指令，其中 $A=\lfloor N/\lg N \rfloor$。CROSS 和 OMFILP 指令均基于 $2\lg N$ 级 Benes 可重排无阻塞多级动态互连网络构成，它们之间虽然互连函数不同，但拓扑结构等价。因此完成一次置换操作都需要 $\lg N$ 条指令。BFLY & IBFLY 指令基于 Inverse Butterfly 和 Butterfly 多级动态互连网络首尾相接而成，它能够在两条指令内完成一次 N bit 任意置换操作，是目前实现置换最快的指令。但该指令需要的源操作数较多，必须改变通用处理器的指令架构及硬件结构才能实现，因此设计复杂度高，实际可行性不强。PERMS 指令利用比特交换的方式完成 N bit 数据的置换，能够在 $\lg N$ 条指令内完成一次置换，但一条 PERMS 指令需要执行$\lfloor N/\lg N \rfloor$个时钟周期。虽然该指令的硬件设计复杂度较低，但其置换性能也较低。GRP 指令基于两个并行的 Inverse Butterfly 网络构成，它将控制寄存器 R3 中为“1”的控制位对应在寄存器 R2 中的数据抽取出来依次排在目的寄存器 R1 的最右侧，并将为“0”的控制位对应的数据抽取出来依次排在目的寄存器 R1 的最左侧。常忠祥等人[15]结合双调归并排序的思想，提出了一种能够在 $\lg N$ 条 GRP 指令内完成任意置换的方案。该指令的硬件结构复杂度适中，且无需改变原处理器结构及指令编码规则，具有较高的实用价值。

针对 $2N$-$2N$ 比特置换在 N 位通用处理器上高效实现的问题，文献[16]提出了一种基于比特数据先置

表 2 常用比特置换指令功能及性能

专用指令	N-N (32bit)	$2N$-$2N$	kN-kN	硬件实现复杂度	处理性能
PPERM3R	$\approx \lg N$	$3\lg N+6$	–	高	适中
CROSS	$\lg N$	$3\lg N+6$	–	低	适中
OMFLIP	$\lg N$	$3\lg N+6$	–	低	适中
BFLY&IBFLY	2	2+6	–	高	高
PERMS	$\leq \lg N$	$3\lg N+6$	–	低	低
GRP	$\lg N$	$2\lg N+10$	–	适中	适中

换后组装的快速实现方法。该方法能够在 $3Q+6$ 条指令内完成一次任意 $2N$-$2N$ 的大位宽置换操作，Q 为完成一次 N bit 置换所需的指令条数。然而对于 kN-kN($k>2$)更大位宽的比特置换高效实现技术几乎没有涉及，这仍然是一个亟待解决的问题。若在 N bit 位宽处理器上实现 kN-kN 的置换操作，首先需要将 kN bit 的数据按照 N bit 位宽进行数据分组归位，确保分组归位后的 k 个 N bit 数据组间不会发生交互，而组内数据乱序。当分组完成后，它们则能够利用已有的研究成果在 $\lg N$ 条指令内完成各组内的任意置换。因此每一比特数据分组归位的执行效率直接决定了大位宽比特置换的效率。基于此，本文重点针对大位宽比特置换操作中数据的分组归位环节进行了详细分析，提出了一种利用比特提取和比特提取-移位专用指令的高速分组归位实现方案。然后，对方案中涉及的两条指令，构造了一种基于 Inverse Butterfly 网络的统一硬件单元-RERS，并设计了配套路由算法。

3　大位宽置换高速实现原理及专用指令设计

3.1 $2N$-$2N$ 比特置换实现原理

当在 N bit 通用处理器中完成 $2N$ bit 任意置换操作时，被置换的数据将被存储于两个 N bit 寄存器中，假设为 Rs1 和 Rs2。其中 Rs1 中一定存在 X bit 数据($0\le X\le N$)属于 Rs2，同时 Rs2 中也相应的有 X bit 数据属于 Rs1，且两个寄存器中 X bit 的数据位置随机。因此，为了完成一次 $2N$ bit 的任意置换，首先需要将这 X bit 数据置换到各自所属的寄存器中，这样处理器就能够在其规定的操作位宽下(N bit)进行任意置换操作。图 1 中，分 3 步完成了 Rs2 中属于 Rs1 的 X bit 数据归位：步骤 1，分别提取出 Rs1, Rs2 中属于 Rs1 中的数据，同时将无关数据置零；步骤 2，将 Rs2 中提取的数据左移 $N-X$ 位，以确保两个寄存器中属于 Rs1 部分在“或”操作时的数据不会产生交叠，这需要一条 SHIFT 指令；步骤 3，将 Rs1 与 Rs2 进行“或”(OR)操作，从而完成 $2N$ bit 数据中属于 Rs1 部分数据的归位操作，需要一条“或”指令。然后，就可以在 $\lg N$ 条指令内完成 Rs1 中乱序数据的置换操作。那么，一个寄存器中数据的归位及再置换过程一共需要进行 2 次数据提取操作(结果序列中未被提取的数据需置为“0”)，外加 $\lg N+2$ 条基本指令。两个寄存器数据的归位及再置换操作，则共需 4 次数据提取操作，再加 $(2+\lg N)\times 2$ 条基本指令。

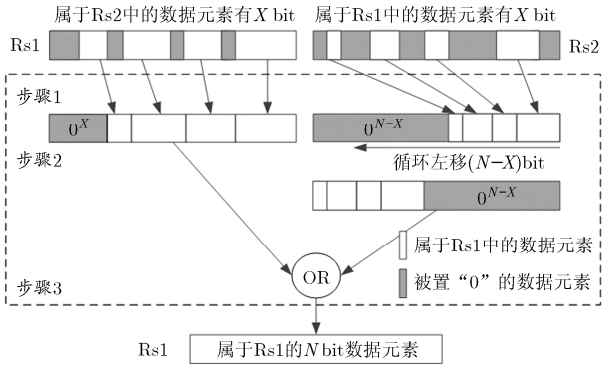


图 1　$2N$-$2N$ 比特置换的实现原理

3.2 kN-kN 比特置换实现原理

当被置换数据从 $2N$ bit 扩展到 kN bit 时，设它们被存储于 k 个 N bit 寄存器中，依次为 Rs1 ~ Rsk。若要完成 kN-kN 的置换操作，其原理与 $2N$-$2N$ 相似，都需要先将属于各个寄存器内部的数据归位。但是，每次进行归位操作时，各个寄存器中存储的属于该寄存器的数据个数不一定相同，因此需要对 $2N$-$2N$ 的置换过程进行调整，以适应更大位宽的比特置换操作。假设需要对所有属于 Rs1 中的 N bit 数据进行归位，那么步骤 1 需要从每个寄存器(Rs1 ~ Rsk)提取出属于 Rs1 部分的数据，置于该寄存器的最右边，并将无关数据置零，该过程需要 k 条数据提取操作，如图 2 中步骤 1 所示。接着，为了使各寄存器中被提取的数据在进行“或”操作时不产生交叠，还需将除第 1 个寄存器 Rs1 外，每个寄存器中提取结果进行移位操作，移位位数根据前面寄存器提取数据的个数确定，该过程需要 $k-1$ 条 SHIFT 指令，如图 2 中步骤 2 所示。那么，最后一个寄存器 Rsk 中被提取到最右边的属于 Rs1 中的数据需要移位 $(\mathrm{S1}+\mathrm{S2}+\cdots \mathrm{S}i+\cdots+\mathrm{S}k-1)bit, Si$ 为第 i 个寄存器中属于 Rs1 中的数据个数，且 $1\le i\le k-1$。最后，将 k 个寄存器中的数据依次进行“或”操作，即可完成 Rs1 中数据的归位，这需要 $k-1$ 条“或”指令，如图 2 中步骤 3 所示。因此，完成一个寄存器数据的归位操作需要 k 次数据提取操作，外加 $2k+\lg N-2$ 条基本指令。当完成 k 个寄存器数据的任意置换操作时，一共需要 k^2 次数据提取操作，再加 $(2k+\lg N-2)\times k$ 条基本指令组合完成。

3.3 专用扩展指令设计

无论是 $2N$-$2N$ 还是 kN-kN 的置换都用到了数据的提取操作。此外，除了被分组归位的那一个寄存器外，剩余的 $k-1$ 个寄存器中被提取的数据还需要进行移位操作。因此为了进一步缩减指令条数，增强单个指令功能，本文基于 MIPS32 指令架构，提出了比特提取指令(Bit EXtract operation, BEX)和

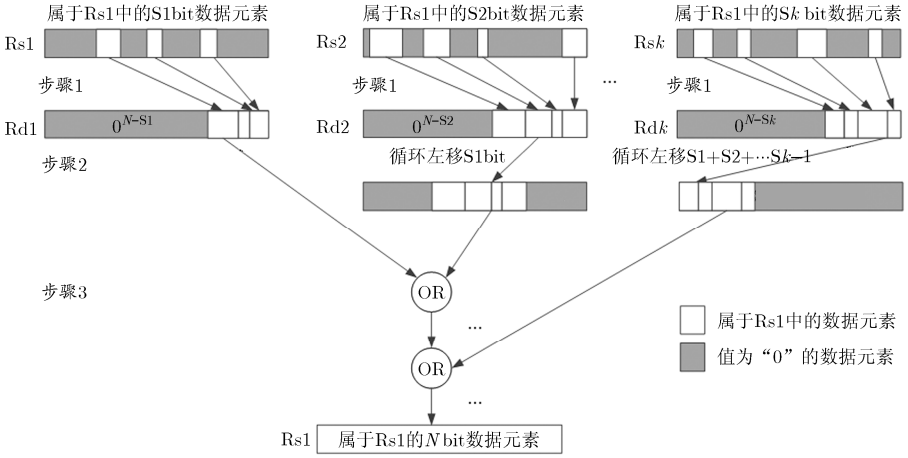


图 2 *kN*-*kN* 比特置换的实现原理

比特提取后循环移位指令(Bit EXtraction and ROTation operation, BEX-ROT)。使用这两条专用指令后，2*N*-2*N* 置换指令条数缩减为 $2\lg N+6$ 条，*kN*-*kN* 置换指令条数缩减为 $(2k+\lg N-1)\times k$ 条。MIPS32 指令架构中所有指令都是 32 位如图 3(a)所示，其中 op 是指令操作码，func 是功能码，sa 仅在移位时使用，rs 和 rt 为两个源寄存器且 rs 与前文中 Rs 寄存器功能相同为数据寄存器，rd 与前文中 Rd 寄存器功能相同为结果寄存器。

图 3(a)中 R 型指令能够提供较多的源寄存器，主要完成复杂算数、逻辑运算指令，因此本文也基于该类型指令进行专用指令扩展。其中 BEX 指令格式为

BEX rd rs rt

指令中 6 位 op 操作码和 5 位 func 功能码一起决定该指令所完成的功能，sa 为恒定值 0，rs 寄存器中存放需要被提取的数据序列，rt 寄存器存放控制序列，rd 寄存器存放 BEX 指令输出结果，如图 3(b)所示。

BEX-ROT 指令格式为

BEX-ROT rd rs rt sa

指令中 6 位 op 操作码和 5 位 func 功能码一起决定该指令所完成的功能，sa 为 5 bit 移位位数(0 ~ 31)，rs 寄存器中存放需要被提取的数据序列，rt 寄存器存放控制序列，rd 寄存器存放该指令输出结果，如图 3(c)。

4 可重构路由算法及统一硬件架构设计

BEX-ROT 指令能够完成 N bit 数据的并行提取后循环移位操作，且当 sa 为 0 时，即可退化为 BEX 指令。已知 BEX 指令能够通过 Inverse Butterfly 网络实现[17]。因此，本文通过深入分析该网络的拓扑结构特性，提出了一种可重构提取-移位操作路由算法，算法能够根据不同的操作模式(Opcond_mode)，在同一个架构下生成用于并行提取(BEX)或并行提取+循环移位(BEX-ROT)操作的路由信息，如表 3 所示。

算法第(1)部分中，Popcnt 函数的作用是统计 *N*bit 序列 C_bit 从第 0 比特位到第 l 比特位中“1”的个数，记为 sum[l]，其中 $0\le l\le N-1$。图 4 中计算出了控制序列 C_bit=1001_0101 中最低 5 bit 和最低 3 bit 中“1”的个数，分别记为 sum[5] = 3，sum[3] = 2。

算法第(2)部分中，函数 R_L($1^m, x$) 的功能是根据参数 x 的数值，对连续 m 个“1”序列进行循环左移后最低位取反操作 x 次。如图 5 所示，当 $m=4$ 时，它以 $2m$ 为周期，初始序列为“1111”经过 8

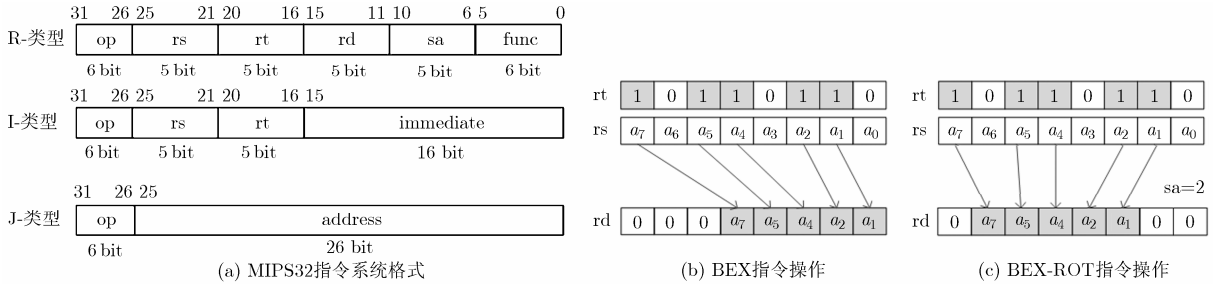


图 3 MIPS32 架构中指令格式及扩展指令功能

表 3 可重构提取-移位路由算法

Input: C_bit; //Initial control bit for parallel bit-extraction C_bit
Input: Shamt; // Rotation amount Shamt=sa
Input: Opcond_mode;
Output: Control_bit; // Total of $\lg N \times N/2$ control bit
(1) **for** $l = 1,2,\cdots,N-2$
　Sum[l] = Popcnt(C_bit{$l:0$}) //Popcnt(a) is the sum of '1's in the Control string C_bit from l to 0
end for
(2) **for** $(i=1, i<=\lg N, i++)$ //parallel i stand for each stage
　$m = 2^{i-1}$; // the length of control bit in stage i
　for $(j=1, j<=N/2^{i-1}-1, j=j+2)$ //the number of sub_networks in stage i
　$q = j \times m - 1$;
　if (Opcond_mode==1)
　Control_bit(i) = R_L(1^m, Sum[q] + Shamt);
　else
　Control_bit(i) = R_L(1^m, Sum[q]); //only implementation parallsim bit-extraction operations
end for
end for

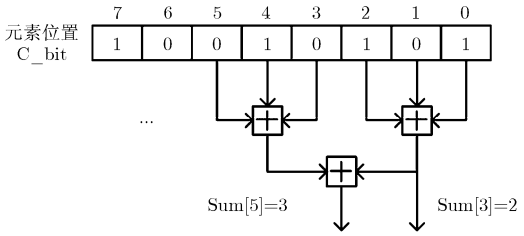


图 4　Popcnt 函数功能

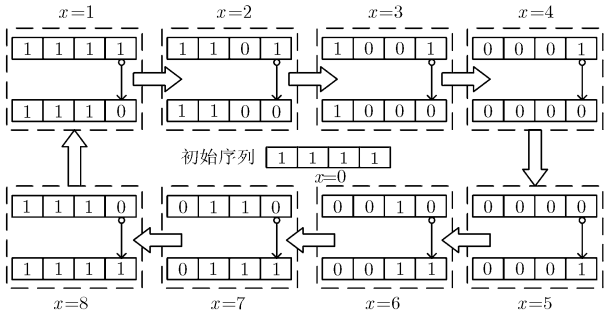


图 5　R_L($1^m = 4, x$) 函数功能

次变换后，其结果回归初始输入序列。

从该算法中还可以看出，比特提取和比特提取-移位两种操作路由控制信息的生成，都依赖于 Popcnt 函数和 R_L($1^m, x$) 函数，不同之处仅在于参数 x 的取值不同，前者为 $x = \text{Sum}[q]$，后者为 $x = \text{Sum}[q] + \text{Shamt}$。因此，这两种操作在硬件功能单元映射时，不仅能够共用 Inverse Butterfly 网络电路，而且路由算法电路也能够最大程度地共享，这有效提高了硬件资源的复用率并节省电路面积。

在两种操作路由算法高度融合的基础上，本文构建了面向比特提取和比特提取-移位两种指令的统一硬件加速单元 RERS(Reconfigurable Extract and Rotation Shifter)，如图 6 所示。它有 4 个输入端口和一个输出端口，其中输入端口 op+func 负责 RERS 单元工作与否，由译码电路产生。rs 为 32 bit 数据寄存器数据输入端口，rt 为 32 bit 控制序列输入端口，sa 为 5 bit 移位位数。当执行比特提取指令(BEX)时，sa 恒为 0，输出端口 rd 为 32 bit 结果序列。

图 6 中主要包括 Inverse Butterfly 网络电路和可重构提取-移位路由算法生成电路，其中左边 Inverse Butterfly 网络电路由 5 级二输入交叉开关组成，负责数据的路由传递工作。图 7 描述了一个 $N = 8$ bit 位宽的 Inverse Butterfly 动态互连网络。该网络有 $\lg N = 3$ 级从上到下依次为第 1 级、第 2 级和第 3 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 bit 控制信号(Sel)的作用下完成一对输入信号的“交叉”(Sel = 1)或者“直通”(Sel = 0)。

右边为路由算法生成电路，由 Popcnt 加法电路和 R_L 函数电路两部分构成。其中 Popcnt 电路用于统计控制序列 C_bit 中从最低位开始到每一位为止的“1”的个数，并与移位位数(sa)相加。该部分电路仅完成加法运算，设计原理简单，不再详述。R_L 函数根据 Popcnt 加法电路的结果 x，对连续 m 个 1 序列进行循环移位后最低位取反 x 次，每次移位的输出序列如图 5 所示。由于该函数使用到了循环移位操作，因此本文首先利用对数移位器并稍加改进以实现其函数功能。如图 8(a)所示，假设 d_j 为 4 bit 输入数据，y_j 为函数结果序列($0 \le j \le 3$)，$x_2x_1x_0$ 为移位位数 sa 的二进制表示形式。由于其初始输入值为恒定值全“1”，因此可以对该函数每一个输出进行卡诺图化简，以精简冗余逻辑。化简过程及化简结果分别如图 8(b)和图 8(c)所示。与传统结构相比图 8(a)，优化后的电路面积减少约 85%，延迟减少约 40%。

5　性能评估与分析

通用处理器(RISC 结构)一般处理位宽为 32 bit，因此本文将 N 选定为 32 bit[18,19]。同时，根据密码算法中置换操作的实际需求，将扩展参数 k 选定为 2, 3, 4, 8。然后分别用本文提出的 2N-2N 和

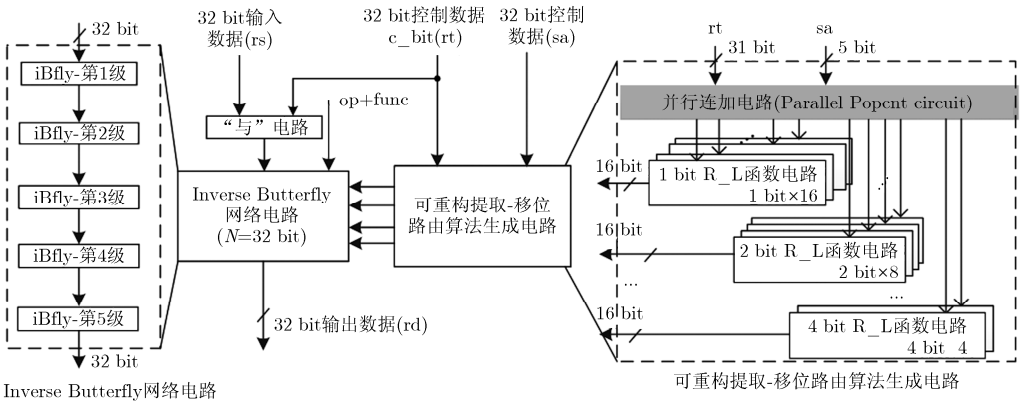


图 6 RERS 整体架构

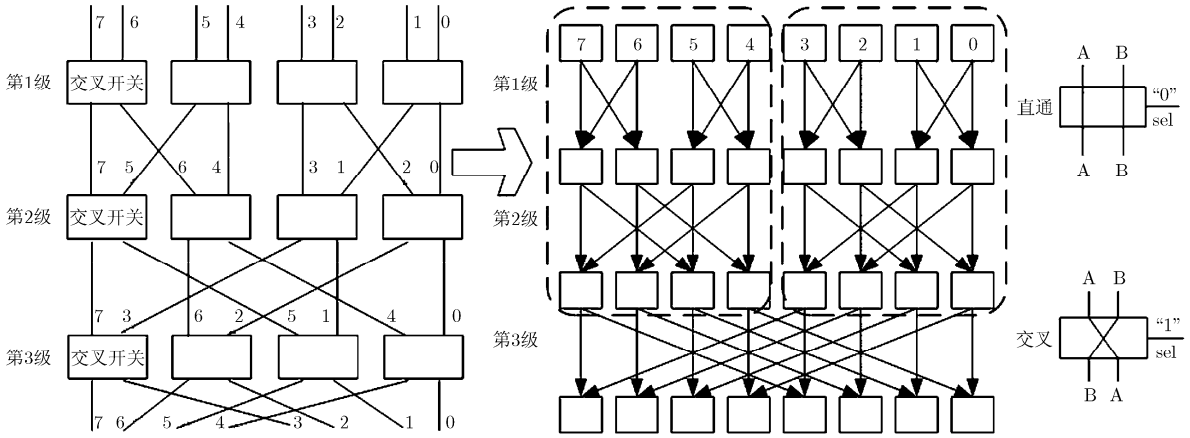
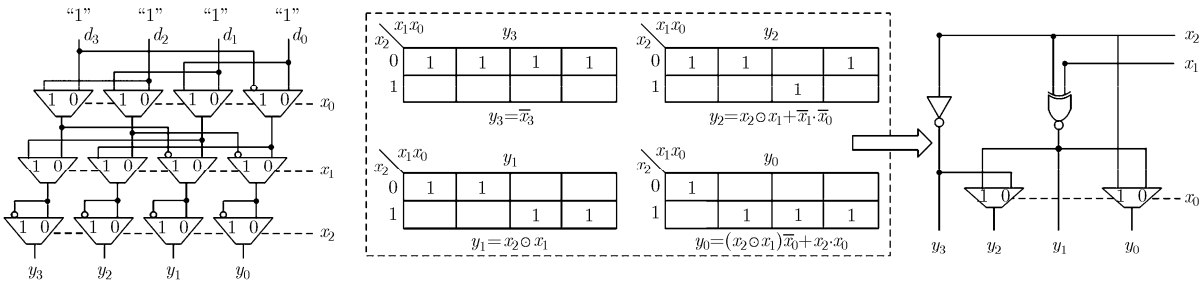


图 7 Inverse Butterfly 网络拓扑结构与数据流图



(a) R_L电路优化前电路 (b) R_L电路布尔化间过程 (c) R_L电路布尔优化后电路

图 8 $m=4$ 时 R_L 函数电路布尔逻辑优化

kN-*kN* 大位宽置换实现方案，对不同参数 k 下的置换操作以统计指令条数的方式进行量化，并与文献[12]和文献[13]提出的置换实现方案进行对比，结果如图 9 所示。

图 9 中横坐标为扩展参数 k 的取值，纵坐标为指令条数，用以 2 为底的对数刻度表示。当完成 2*N*-2*N* 的置换操作时，利用本文提出的方案仅需要 16 条指令，比原指令架构所需的 256 条减少约 16 倍。与同类设计相比，较文献[12]减少 4 条指令，指令条数降低约 20%；较文献[13]的 15 条指令仅差一条指令。当完成 $k=3,4,8$ 等大位宽置换时，文献[12]和文献[13]由于不能有效地完成分散到各个寄存器中数据的归位操作。因此，无法支持这些类型的置换，图 9 中将其对应的指令条数标注为“NA”，表示不支持该操作。利用本文提出的方案，能够分别在 30, 48 和 160 条指令内完成这些大位宽置换，且分别比原指令架构实现对应置换操作所需的指令条数减少 12.8 倍、10.7 倍和 6.4 倍。因此，本文提出

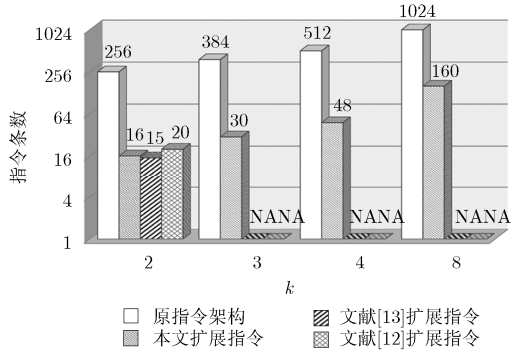


图 9 不同参数 k 的置换指令条数对比

的置换方案及专用指令能够有效降低大位宽置换操作在通用处理器中的指令条数，提升其处理性能。

进一步，为了评估由扩展专用指令所带来的硬件资源消耗对通用处理器硬件架构的影响，本文将设计的 RERS 硬件单元在 COMS 65 nm 工艺下进行了逻辑综合与优化。同时，为了实现与文献[12]和文献[13]的客观对比，本文将 RERS 的位宽 N 扩展到 64 bit，命名为 RERS-64，并在相同工艺下完成了综合与优化。综合时环境设置为 $P=\text{Slow}$, $V=1.08$ V, $T=125$℃，并采用 flatten 的优化策略，结果如表 4 所示。

表 4 中文献[12]和文献[13]方案中的置换单元位宽均为 64 bit，且基于 TSMC 90 nm 工艺综合完成。一般地，对不同工艺下器件的性能进行对比并不科学。因此，为了尽可能地降低由于工艺偏差而导致的性能差异，本节根据文献[22]中针对不同工艺下的延迟换算关系：$T_{\text{new}}=l\times T_{\text{old}}$，其中 T_{new} 和 T_{old} 分别表示新、旧工艺器件的延迟，l 表示旧工艺与新工艺特征尺寸的比值，将表 4 中第 3 列数据进行了等效延迟换算，结果如表 4 中最后一列所示。

对表 4 中前 3 行分析可知：与通用处理器架构相比，本文设计的 32 位 RERS 电路面积分为 CK807 处理器(杭州中天微电子公司设计的 32 位 RISC 处理器)面积的 0.44%，为 Cortex-A8 处理器(ARM 公司设计的 32 位 RISC 处理器)面积的 0.13%，且 RERS 的延迟值为 0.87 ns，小于表 4 中两个处理器的延迟。因此，若将该指令扩展到不同 RISC 处理器中，其硬件资源开销极低，且不会影响到原架构的正常工作频率。

对表 4 中的后 3 行分析可知：与同类设计相比，本文设计的 RERS-64 电路面积为 9.82 k，约为文献[12]面积的一半，但比文献[13]的面积大 3 倍多。等效延迟为 1.22 ns 与文献[12]相当，是文献[13]延迟的 2.5 倍。虽然，文献[13]硬件单元的面积和延迟均较小，但是它完成一条其对应的扩展指令需要 16 个时钟周期。而完成一条本文提出的专用指令，仅需一个时钟周期。从该角度分析，本文提出的一条专用指令耗时为 $1.22\times1=1.22$ ns，文献[13]的一条专用指令则耗时为 $0.51\times16=8.16$ ns。因此，本文设计 RERS-64 单元与文献[12]相比，无论从面积还是等效延迟均占有一定优势；与文献[13]相比，虽然面积为它的 3.6 倍，但是完成一次对应指令所消耗的时间仅为后者的 15%。

6　结束语

本文详细分析了 $2N$-$2N$ 和 kN-kN 大位宽比特置换操作在通用处理器中的高速实现原理。提出了一种基于比特提取和比特提取-移位操作的高效数据分组归位方案。进一步，针对方案中的比特提取和比特提取-移位操作，设计了相应的专用扩展指令 BEX 和 BEX-ROT。并构造了基于 Inverse Butterfly 动态互连网络的统一硬件加速单元-RERS。实验结果表明，本文提出的两条专用置换指令，以较小的硬件资源消耗(面积增加不超过原处理器面积的 0.5%)，在不影响原通用处理器正常工作频率的情况下，将大位宽比特置换操作的执行效率提高了 10 余倍。下一步，还需研究如何将本文提出的针对大位宽置换操作的高效实现方案，移植到更多处理器架构中，以评估其带来的实际性能提升。

表 4 面积、延迟性能对比

功能单元	面积(nand gates)	延迟(ns)	位宽(bit)	工艺(nm)	等效延迟(ns)
本文 RERS	5.27k	0.87	32	COMS 65	0.87
CK807 处理器[20]	1200k	1.50	32	COMS 65	1.50
ARM Cortex-A8[21]	4000k	1.00	32	COMS 65	1.00
本文 RERS-64	9.82k	1.22	64	COMS 65	1.22
文献[12]	19.7k	1.74	64	TSMC 90	1.26
文献 [13]	2.7k	0.70	64	TSMC 90	0.51

参 考 文 献

[1] SHAN Weiwei, CHEN Xin, LU Yinchao, *et al*. A novel combinatorics-based reconfigurable bit permutation network and its circuit implementation[J]. *Chinese Journal of Electronics*, 2015, 24(3): 513–517. doi: 10.1049/cje.2015.07.013.

[2] AO T, HE Z, and DAI K. Low-cost bit permutation circuit with concise configuration rule[C]. Proceedings of the International MultiConference of Engineers and Computer Scientists, Hong Kong, 2015: 158–160.

[3] JOLFAEI A, WU X, and MUTHUKKUMARASAMY V. On the security of permutation-only image encryption schemes[J]. *IEEE Transactions on Information Forensics and Security*, 2015, 11(2): 235–246. doi:10.1109/TIFS.2015.2489178.

[4] LI W, YU F, and MA Z. Efficient circuit for parallel bit reversal[J]. *IEEE Transactions on Circuits & Systems II Express Briefs*, 2016, 63(4): 381–385. doi: 10.1109/TCSII.2015.2504943.

[5] RAVAL N, BANSOD G, PISHAROTY D N, *et al*. Implementation of efficient bit permutation box for embedded security[J]. *WSEAS Transactions on Computers*, 2014(13): 442–451.

[6] 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.

[7] SHIBUTANI K, ISOBE T, HIWATARI H, *et al*. PICCOLO: An ultra-lightweight blockcipher[C]. Cryptographic Hardware and Embedded Systems-CHES 2011, Nara, 2011: 342–357. doi: 10.1007/978-3-642-23951-9_23.

[8] BOGDANOV A, KNUDSEN L R, LEANDER G, *et al*. PRESENT: An ultra-lightweight block cipher[J]. *Lecture Notes in Computer Science*, 2007, 4727: 450–466. doi: 10.1007/978-3-540-74735-2_31.

[9] MINIER M and GILBERT H. Stochastic cryptanalysis of crypton[C]. FAST Software Encryption, International WorkShop, FSE 2000, New York, 2000: 121–133. doi: 10.1007/3-540-44706-7_9.

[10] BIHAM E, ANDERSON R, and KNUDSEN L. SERPENT: a new block cipher proposal[J]. *Lecture Notes in Computer Science*, 1998, 1372: 222–238. doi: 10.1007/3-540-69710-1_15.

[11] CHENG H, HEYS H M, and WANG C. PUFFIN: A novel compact block cipher targeted to embedded digital systems[C]. Euromicro Conference on Digital System Design Architectures Methods and Tools, Parma, 2008: 383–390. doi: 10.1109/DSD.2008.34.

[12] HILEWITZ Y and LEE R B. Fast bit gather, bit scatter and bit permutation instructions for commodity microprocessors [J]. *Journal of Signal Processing Systems*, 2008, 53(1):145–169. doi: 10.1007/s11265-008-0212-8.

[13] KOLAY S, KHURANA S, SADHUKHAN A, *et al*. PERMS: A bit permutation instruction for accelerating software cryptography[C]. Euromicro Conference on Digital System Design, Los Alamitos, 2013: 963–968. doi: 10.1109/DSD.2013.109.

[14] SANGEETHA M and JAGADEESWARI M. Design and implementation of new lightweight encryption technique[J]. *International Journal of Innovative Research in Science Engineering and Technology*, 2016, 5(3): 8610–8617.

[15] 常忠祥, 戴紫彬, 李伟, 等. 基于互连网络的比特置换实现技术[J]. 计算机工程与设计, 2014(8): 2640–2644. doi: 10.3969/j.issn.1000 -7024.2014.08.004.

CHANG Zhongxiang, DAI Zibin, LI Wei, *et al*. Bit permutation based on interconnection network[J]. *Computer Engineering and Design*, 2014(8): 2640–2644. doi: 10.3969/j.issn.1000-7024.2014.08.004.

[16] SHI Z J. Bit permutation instructions: Architecture, implementation, and cryptographic properties[D]. [Doctoral dissertation]. Princeton University, 2004.

[17] HILEWITZ Y and 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. doi: 10.1109/TC.2008.219.

[18] SAYILAR G and CHIOU D. CRYPTORAPTOR: High throughput reconfigurable cryptographic processor[C]. IEEE /ACM International Conference on Computer-Aided Design, San Jose, 2014: 155–161. doi: 10.1109/ICCAD.2014.7001346.

[19] BENHADJYOUSSEF N, ELHADJYOUSSEF W, MACHHOUT M, *et al*. Enhancing a 32-bit processor core with efficient cryptographic instructions[J]. *Journal of Circuits, Systems & Computers*, 2015, 24(10): 1550158–1550178. doi: 10.1142/S0218126615501583.

[20] 胡敏, 卢永江, 刘兵. 基于 CK810 处理器的汇编链接时优化[J]. 计算机工程, 2014, 40(11): 250–254. doi: 10.3969/j.issn.1000-3428.2014.11.050.

HU Min, LU Yongjiang, and LIU Bing. Assembly and link time optimization based on CK810 processor[J]. *Computer Engineering*, 2014, 40(11): 250–254. doi: 10.3969/j.issn.1000-3428.2014.11.050.

[21] ARM corporation. Cortex®-A8 processor [OL]. http://www.arm.com/zh/products/processors/cortex-a/cortex-a8.php, 2016.10.

[22] LIU B and BAAS B M. Parallel AES encryption engines for many-core processor arrays[J]. *IEEE Transactions on Computers*, 2013, 62(3): 536–547. doi: 10.1109/TC.2011.251.

戴紫彬： 男，1966 年生，教授，博士生导师，研究方向为专用集成电路设计、多核安全处理器设计、信息安全芯片技术研究等.

马 超： 男，1988 年生，博士生，研究方向为动态互连网络设计、专用集成电路设计、安全芯片设计等.

李 伟： 男，1983 年生，副教授，研究方向为安全专用芯片设计、阵列处理器设计、集成电路技术研究等.

南龙梅： 女，1981 年生，讲师，研究方向为体系结构、安全芯片设计、集成电路技术研究等.