IBPU:一种面向通用处理器架构的比特置换功能单元

马 超[1],南龙梅[2],潘达杉[1],李 伟[3],戴紫彬[3]
(1. 国家高性能集成电路(上海)设计中心,上海 201204;2. 复旦大学集成电路国家重点实验室,上海 200433;
3. 信息工程大学,河南郑州 450000)

摘 要: 本文利用 Inverse Butterfly 网络拓扑结构的自路由特性,并结合分治策略,提出了一种能够硬件高速实现任意比特置的换选路算法. 利用该算法能够在 $O(\lg N)$ 条指令内完成 N-bit 任意静态置换操作,在 $O(\lg^2 N)$ 条指令内完成 N-bit 任意动态置换操作. 在此基础上,本文构造了一种新型比特置换单元-Permutation Unit based on Inverse Butterfly,IBPU. 并将它在 SMIC 65nm 工艺下进行了逻辑综合,结果表明:与以往研究成果相比,本文提出的 IBPU 资源消耗降低了约 32%,延迟降低了近 30%. 当完成静态置换操作时,其功能单元所消耗的代价最小,不超过以往设计的 60%;当完成动态置换操作时,虽然消耗的代价较大,但其随置换位宽 N 的增加涨幅较小,因此具有较高的稳定性,其综合性能优势明显.

关键词: Inverse Butterfly 网络; 分治策略; 置换选路算法; 硬件实现
中图分类号: TP393.3 **文献标识码:** A **文章编号:** 0372-2112 (2018)08-1960-09
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2018.08.022

IBPU:A Bit Permutation Functional Unit for General-Purpose Processors

MA Chao[1], NAN Long-mei[2], PAN Da-shan, LI Wei[3], DAI Zi-bin[3]
(1. *National High Performance Integrated Circuit Design Center, Shanghai* 201204, *China*;
2. *State Key Lab of ASIC and System, Fudan University, Shanghai* 200433, *China*;
3. *Information Engineering University, Zhengzhou, Henan* 450000, *China*)

Abstract: In this paper, a new routing algorithm for arbitrary bit permutation operations is proposed combining with the divide and conquer strategy. The algorithm utilizes self-routing characteristics of the Inverse Butterfly Network. It can complete any N-bit fixed permutation in no more than $O(\lg N)$ instructions, and also can complete any N-bit dynamic permutation in no more than $O(\lg^2 N)$ instructions. On this basis, a new bit-permutation unit based on Inverse Butterfly, IBPU is developed and synthesized in SMIC 65-nm process. The results show that our IBPU has less resource consumption which decreased by about 32%, and lower latency which reduced by nearly 30% compared with the similar designs. Moreover, when it performs fixed permutation, the cost of the functional unit is minimal, which is not more than 60% of what was previously designed. When it performs dynamic permutation, though its cost is greater, the cost has smaller increase accompanying with the increase of permutation width N, so it has higher stability and its comprehensive performance advantages are obvious.

Key words: Inverse Butterfly Network; divide and conquer; permutation routing algorithm; hardware implementation

1 引言

比特置换操作在密码学[1]、图像处理、数字信号处理[2]等领域有着广泛地应用,它能够完成 N-bit 序列的任意排列,其结果空间有 N! 种. 然而,无论是通用处理器还是专用指令处理器,对于这种细粒度比特级操作处理效率很低,往往需要将其转化成多条基本指令组合实现,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操作在处理器中的执行效率,成为了人们研究的热点[3,4].

目前,比特置换操作在处理器中加速的主要实现方式是基于多级动态互连网络. 根据置换能力的不同,可将多级动态互连网络分为非阻塞型和阻塞型两种结构[5,6]. 前者的典型代表是 Benes 可重排无阻塞网络,该结构类

收稿日期:2016-07-21;修回日期:2017-11-14;责任编辑:蓝红杰
基金项目:国家自然科学基金(No.61404175)

型网络一般由 2lg$N$-1 级交叉开关组成,理论上数据一次通过该网络就能够完成 $N!$ 种任意置换操作. 但由于其各级开关选路算法复杂度较高,硬件实现十分困难. 因此,在实际处理器应用中,往往先由软件执行置换选路算法,再采用指令配置的方式注入到相应寄存器堆中供 Benes 网络使用[7]. 该方式的缺点在于消耗了额外的存储资源,需要对处理器架构进行一定的改动,不利于快速集成. 而且它应用范围有限,仅适用于置换操作已知的情况,即静态置换操作. 当所需置换的映射关系根据先前指令运算结果产生不可提前预知时,该网络电路必须停止工作,将先前指令的运算结果进行软件预计算选路信息后,再注入到相应配置寄存器来完成置换,这将消耗大量的重构配置时间,严重影响电路性能. 动态多级阻塞网络,如 Inverse Butterfly、Omega、Baseline 等,它由 lg$N$ 级组成,是 Benes 网络级数的一半. 虽然数据一次通过该网络不能完成任意置换操作,但由于其网络级数较短,拓扑结构规则,具有良好的拓扑迭代特性,因此广泛应用于通用处理器架构的设计中. 其中,最为典型的是 Inverse Butterfly. 目前,基于该网络已经开发出了能够硬件高效适配的多种特定置换的路由算法及相应指令,如并行抽取指令(parallel bit extraction,PEX)、并行插入指令(parallel bit deposition,PDEP)[8]、比特归类指令(bit group,GRP)[9]和循环移位指令(bit rotation,ROT)等[10]. 进一步,为了增强该架构的置换能力,扩展其应用范围,Chang 等人又提出了通过多次调用 GRP 指令完成任意置换的实现方案[11]. 相比于无阻塞互连网络的实现方式,它不需要额外的选路信息配置存储电路,其初始选路信息直接来自于指令提供的源操作数,对处理器架构改动较小,适于快速集成. 但初始选路信息的计算复杂度较高,需要由软件承担,该方式虽然降低了硬件资源开销,但从本质上讲它仍仅适于静态置换操作. 因此,基于 Inverse Butterfly 网络设计一种既能兼容已有特定置换操作,又能高效完成任意静态、动态比特置换操作的硬件功能单元,突破比特置换操作在处理器中的计算瓶颈,就显得十分迫切.

本文的创新点是基于多级动态阻塞网络-Inverse Butterfly,提出了一种能够针对任意静态、动态置换快速生成各级开关所需控制信息的实时选路算法. 与传统 Benes 选路算法相比,它复杂度更低,关键路径更短,算法可由硬件电路直接实现. 与文献[9]和文献[11]的同类设计相比,本文提出的选路算法硬件实现时,在面积和延迟两个方面均有提升,其综合性能优势明显.

2　背景知识

Inverse Butterfly 网络作为多级动态阻塞网络的一种,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,其结构性质及选路算法一直是学术界研究的热点问题[12,13]. 图 1 描述了一个 N = 8-bit 的 Inverse Butterfly 网络,它由 lg$N$ = 3 级组成,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(Switch),每个交叉开关在 1-bit 控制信号 Sel 的作用下,能够实现 2-bit 输入数据的交叉或直通,网络共有 $\lg N \times N/2$ 个交叉开关,能够实现 $2^{\lg N \times N/2}$ 种置换. 输入数据在网络各级以 2^{i-1}-bit(i 为级数,$1 \leqslant i \leqslant \lg N$)为间隔进行两两分组后进入同一个开关中,若将所有开关设置为直通状态即“0”,那么该网络的输出数据等于输入数据,完成恒等置换. 若将最后一级开关舍弃或设置为直通,那么剩余的 lg$N$-1 级网络则能够看成是两个相互独立且以 $N/2$-bit 为位宽的子蝶网络,左边记为 sub-ibfly$_2$,右边记为 sub-ibfly$_1$,子蝶网络与原网络的功能完全相同,只是规模较小.

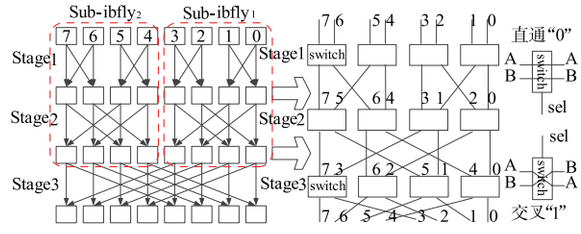


图1　Inverse Butterfly网络拓扑结构与数据流图

2009 年,Hilewitz 等人率先将 Inverse Butterfly 网络引入到通用处理器内部单元设计中,构造了一种新型移位-置换单元[14]. 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(PEX)、并行插入(PEDP)统一到了一个架构下,并成功应用于 Intel 在 2013 年发布的 Haswell 处理器中[15]. 2014 年,Chang 等人结合归并排序算法,提出了一种利用多条 GRP 指令实现任意置换的方案,使其对任意置换操作具有了普适性,大幅提升了该网络的置换性能[11]. 但方案中,GRP 指令所需的初始选路信息的生成算法仍然较为复杂,只能采用软件预计算的方式生成. 且该指令对应的硬件架构由两个 Inverse Butterfly 网络和两套路由算法电路构成. 因此,该方式硬件实现起来代价较高,且只适于置换已知的情况,对于动态置换操作仍不能较好的支持.

基于上述原因,本文以高效实现置换操作为研究目标,结合分而治之的策略,从基于 Inverse Butterfly 网络的置换实现原理入手,详细分析整个置换过程,进而提出一种仅利用一个 Inverse Butterfly 网络就能够完成任意置换,且适于硬件实现的选路算法.

3　任意置换操作选路算法研究

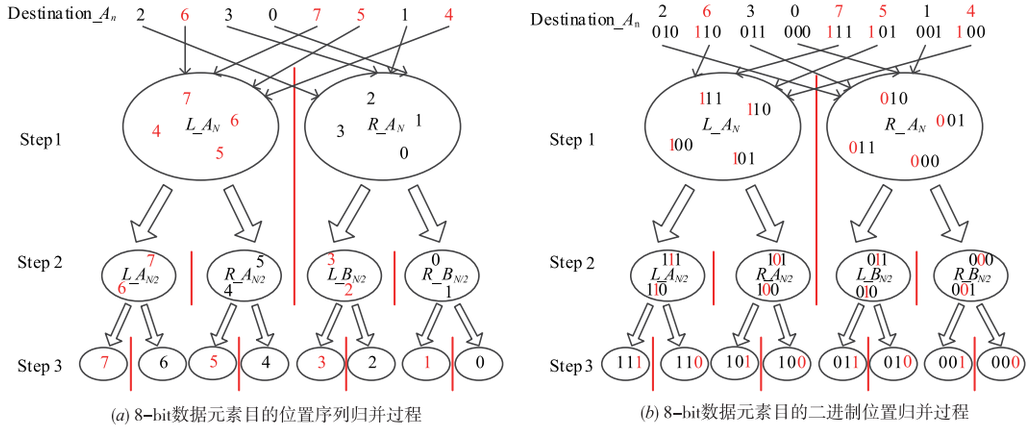
3.1　任意置换在 Inverse Butterfly 网络的实现原理

本节将应用分治策略首先将一个 N-bit 的置换划分成两个独立的 $N/2$-bit 置换,然后再递归迭代处理两个

并行的 $N/2$-bit 置换,直到完成整个置换.

设一个 N-bit 序列为 $A_N=(a_{N-1},a_{N-2},\cdots,a_0)$,记 A_N 中从左至右每个元素置换后的目的位置为 Destiniation _ position _ a_i,$(0\leqslant i\leqslant N-1)$ 则有置换表 $\begin{matrix}\text{Orignial\_position\_a}_i=\\ \text{Destination\_position\_a}_i=\end{matrix}\begin{pmatrix}N-1 & N-2 & \cdots & N/2 & N/2-1 & \cdots & 1 & 0\\ 2 & 6 & N-1 & \cdots & \cdots & N-2 & 0 & 1\end{pmatrix}$,其中上层数据代表被置换元素 a_i 的源地址,下层数据代表该元素被置换后的目的地址. 对序列 A_N 的置换过程可以归结为对 Destiniation_position_a_i 集合中元素的排序过程. 现将序列 A_N 从中间分开形成左、右两个位宽相等的子序列,分别记为 $L\_A_N=(a_{N-1},a_{N-2},\cdots,a_{N/2})$ 和 $R\_A_N=(a_{N/2-1},a_{N/2-2},\cdots,a_0)$. 假设 L_A_N 中有 M 个元素($0\leqslant M\leqslant N/2$)需要被置换到 R_A_n 中,同时 R_A_n 中也一定有 M 个元素需被置换到 L_A_N. 若将这两部分中不属于自己的 M 个元素相互换位,那么初始序列 A_N 中左、右两个子序列将完成 $N/2$-bit 位宽下元素的“归位”. 然后,再将两个子序列看待成独立需要被置换的序列 $A_{N/2}=L\_A_N$ 和 $B_{N/2}=R\_A_N$,并以相同的方式迭代处理. 当第 lg$N$ 次时,A_N 将形成 $N/2$ 个彼此独立的子序列,且子序列间已经完成了元素的归位. 这时仅需将各子序列内部的 2-bit 数据元素进行有条件的交换即可.

如图 2(a)所示,8-bit 目的序列完成第一次归并后(Step 1),左边需要被置换到右边的 3-*bit* 元素(2,3,0)和右边需要被置换到左边的元素(7,5,4)完成了位置互换,结果为(7,4,6,5)、(2,3,1,0). 第二次归并时(Step 2),将上述 2 部分结果独立看待,再完成各自内部数据的归位,结果为(6,7)、(4,5)、(3,2)、(0,1). 最后一次的归并(Step 3)是将 4 个 2-bit 子序列内部元素进行位置调整,以完成最终的置换. 从图 2(b)中可知每一次的归并,其实质是对各元素目的二进制地址序列从高位到低位的一个归类. 例如,第一次归并时,将 8 个元素二进制目的地址序列中最高位为“1”的划到了左边,为“0”的划到了右边. 每一次归并操作的执行均涉及两个方面的问题:第一,分别提取出左、右两个部分中被交换的若干元素;第二,将提取的元素互换位置,以完成各自部分内数据的归位. 那么,如何将上述归并过程与 Inverse Butterfly 网络拓扑结构相结合,使数据一次通过该网络,即可同时解决这两个问题,进而完成从置换理论分析到具体硬件实现是本文的研究重点.



(a) 8-bit数据元素目的位置序列归并过程 (b) 8-bit数据元素目的二进制位置归并过程

图2 任意8-比特数据置换分解过程

已知 N-bit Inverse Butterfly 网络能够实现数据的并行抽取操作(PEX),若不考虑网络最后一级开关,剩余 lg$N$-1 级则构成了两个独立的 $N/2$-bit 子蝶网络,它们能够并行地完成两个 $N/2$-bit 数据的并行抽取操作. 并行抽取操作的功能是将控制序列 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,其它数据以原相对位置的反序置于左侧,如图 3(a)所示.

第一步归并时,如图 3(b)中两个红框所示,左、右两个独立的 4-bit 子蝶网络,根据各输入数据最终被置换的目的位置,将左边子蝶网络中的{2,3,0}和右边子蝶网络中的{7,5,4}在初始控制序列(1011_1101)的作用下分别抽取到了第 2 级网络的最右边(提取元素的个数 $M=3$),从而完成各子蝶网络中不属于自己部分数据的提取操作. 然后,根据网络在最后一级的拓扑结构知,各元素以 4-bit 为间距进行两两分组,因此只需将最低位连续的 M 个初始选路信息(默认为“0”)置“1”,即可完成提取数据的交换操作,如图 3(c)中第 3 级网络所示. 数据以此方式第一次通过网络后,便完成了左、右两个部分的归位操作,对应的目的位置序列从(2 6 3 0 7 5 1 4)变为(6 7 5 4 1 2 3 0),相应地各元素目的二进制地址序列的最高位完成了归位.

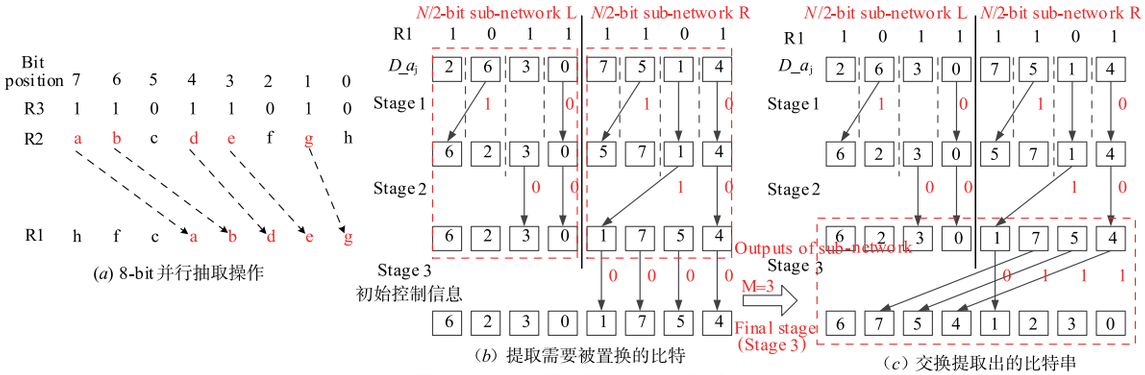


图3　2个4-bit子蝶网络并行抽取操作

第二步归并时,是对两个 4-bit 数据分别进行归并操作,因此只需要前 lg(4) =2 级网络,剩余部分(最后一级)控制信息保持直通,如图 4(*a*)所示. Inverse Butterfly 网络第一级最右边的两个子蝶网络在初始控制信息(10_10)的作用下分别抽取了不属于自己部分的 1 比特(M =1)元素“1”和“3”. 然后根据 M 的个数在第二级右边网络中,将从最低位开始的 M =1 个初始控制信息置“1”,从而完成被抽取元素的交换操作,其最终结果为(2 3 0 1). 相应地,左边 4-bit 数据(6 7 5 4)按照同样的方法,经过第二次归并操作后结果为(7 6 4 5). 当数据第二次通过该网络后各元素目的位置从(6 7 5 4 1 2 3 0)变为(7 6 4 5 2 3 0 1). 同时,各元素二进制地址序列的次高位(第 2 位)也完成了归位.

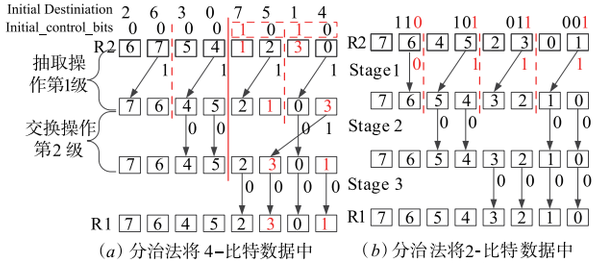


图4　8-比特序列置换硬件实现步骤

第三步归并时,有 4 个独立的 2-bit 数据组,因此只需要前 lg(2) =1 级网络,剩余部分(最后二级)控制信息保持直通如图 4(*b*)所示. 这时,若组内最右边 1-bit 数据目的二进制位置的最低位为“1”则开关状态位为交叉,否则为直通. 图 4(*b*)中各组内最右边元素目的位置二进制最低位为“0”“1”“1”“1”,那么对应网络第一级的控制信息就为“0”“1”“1”“1”. 因此,对于任何一种 N-bit 的置换,按照此方式执行,在最差情况下,仅需要 lg$N$ 步即可完成.

3.2　基于 Inverse Butterfly 网络的置换选路算法

根据上述分析,本文提出了一种基于 Inverse Butterfly 网络的可重构抽取-交换操作选路算法,其算法流程图如图 5 所示. 该算法能够生成并行抽取和交叉换位两个步骤所需的网络各级开关选路控制信息,进而实现数据元素的归位操作. 图 5 中,参数 T 为归并所处的阶段,参数 h($0 \leqslant h \leqslant 2^T - 1$)将初始 N-bit 控制信息 R3 平均分成 2^T 段,各段对应位宽为 $N/2^T$ 的子蝶网络,从左至右依次记为 sub-ibfly$_{2^T}$ ~ sub-ibfly$_1$. 然后根据不同的连加参数 l 值($0 \leqslant l \leqslant 2^T - 1$)执行不属于自己内部数据的并行抽取操作。当 h 为奇数时,还需要统计对应子蝶网络中“1”的个数,以完成交叉换位操作. 其中并行抽取操作如算法 1 所示,POPCNT 函数用于统计 R3 中从第($h \times N/2^T + l$)位到第($h \times N/2^T$)位的“1”的个数,并记为 $Sum\_pc[l]$,RLTR(1^k,m)函数则根据参数 m 的值,将一个由 k 个“1”组成的序列进行循环左移 1 位后末位取反,这样的操作重复 m 次(算法 1 中第二个部分). 同时,根据 3.1 节的置换原理可知,每进行一次归并操作,Inverse Butterfly 网络输入数据位宽减半,同时并行度提升一倍. 因此,算法 1 不仅能够实现 2 个 N/2-bit 位宽下并行抽取操作,还能够实现多组小位宽 2^i(i =1,2,…,lg$N$ -1)并行抽取操作,其可重构意义也在于此.

接下来,就需要对抽取后的元素进行换位操作,该操作将在网络第 lg($N/2^{T-1}$)级进行,它根据参数 T,统计该级所有右边子蝶网络中对应 R3 中的“1”的个数,然后将该级初始开关控制信息进行 RLTR 函数后将结果取反,如式(1):

$$\begin{aligned} &\text{Control\_bits}(\lg(N/2^{T-1})) = \sim \text{RLTR}(1^{N/(2^T)},\\ &M_T = \text{POPCNT}(\text{R3}[Odd\_h \times N/2^T - 1:\\ &(Odd\_h - 1) \times N/2^T])) \end{aligned} \tag{1}$$

其中“Odd_h”代表参数 h 从 0 到 $2^T - 1$ 范围中为奇数的值,“M_T”表示在 T 阶段被交换元素的个数,“ ~ ”表示按位取反. 当 T >1 时,除了完成数据元素的抽取和交换操作外,还需要将 Inverse Butterfly 网络中不要的级数设置为直通即“0”,如式(2):

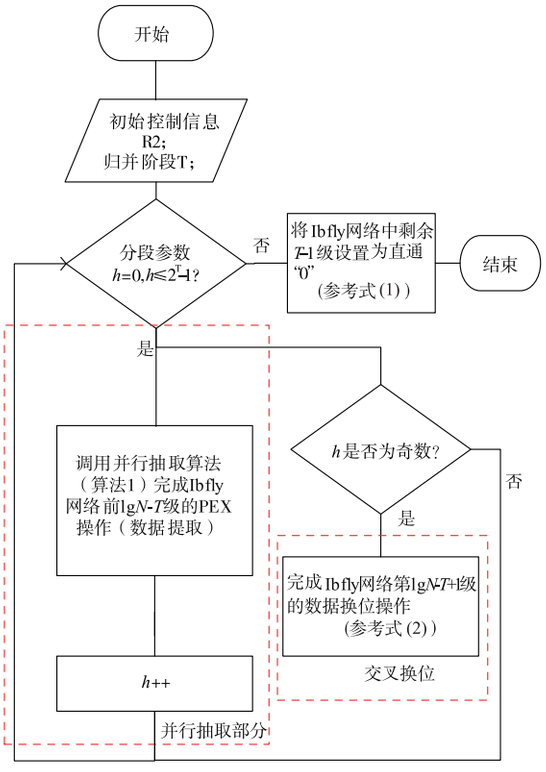


图5 可重构抽取–交换操作流程图

$$\text{Control\_bits}(\text{from } \lg(N/2^{T-1})+1 \text{ to } \lg N)=0^{N/2} \quad (1<T\leqslant \lg N) \tag{2}$$

它直接将 Inverse Butterfly 网络中从第 $\lg(N/2^{T-1})+1$ 级到第 $\lg N$ 级的所有开关设置为“0”即直通.

算法 1:generate the control bits for parallel extraction

```
Input:R3;          //R3 :N-bit initial control bits in register R3
Input:T;.....//Num_step T=1,2,3…lgN-1;
output:C_bits;  //the( lgN-T)×N/2 control bits;
(1) For (l=0,l<=N/2^T-1,l++)
{
  Sum_pc[l]=POPCNT(R3[h×N/2^T+l:h×N/2^T]);
  For i=1,2,…,lg(N/2^T)   //i stand for each stage
    {
      k=2^(i-1)   // the sub-inverse butterfly networks width in stage i
      For (j=1,j<=N/2^(T+i-1)-1,j=j+2)
        {
        q=j × k -1
        C_bits(i)=RLTR(1^k,Sum_pc[q])
        }
    }
}
(2)RLTR(1^k,m)   //1^k indicates a bit-string of k ones
  For (a=0,a<=m,a++)
    {
    C=1^k<<<1   //the k ones bit-string left rotation one time
    C^reverse//reverse the least significant bit of the rotated string C
    }
```

算法 1 的作用是针对 Inverse Butterfly 网络在完成抽取-交换操作时,各级开关所需选路信息的一种生成算法,它本身并不能实现数据的任意置换. 接下来,将讨论如何生成每次归并操作所需的初始控制信息即 R3(Initial Control Bits). 假设初始各元素对应的目的位置序列仍为(2 6 3 0 7 5 1 4),其二进制地址序列对应为(010 110 011 000 || 111 101 001 100). 第一次归并操作完成的是 8-bit 数据中高 4-bit(左半部分)和低 4-bit(右半部分)数据的归位. 归位后高 4-bit 元素对应二进制目的序列最高位为“1”,低 4-bit 元素对应为“0”. 这说明初始二进制序列中左半部分最高位为“0”的元素,和右半部分最高位为“1”的元素需要被提取后交换. 根据并行抽取操作的原理知,需要被提取元素上方的控制信息必须为“1”. 因此,第一次归并操作($T=1$)的初始控制信息可以通过对左边二进制地址序列最高位按位取反后,与右边二进制地址序列最高位一起并置后取得,为:Initial Control Bits = (1 0 1 1 ||1 1 0 1),如图 6(a)所示.

第二次归并操作($T=2$)是将第一次操作的结果作为两个独立的 4-*bit* 输入数据,再完成 4-bit 序列内 2-bit 元素的归位. 由于第一次归并操作完成后,初始数据的位置已经发了变化,从(2 6 3 0 7 5 1 4)变为(6 7 5 4 1 2 3 0). 那么要完成第二次数据的归并操作,初始数据对应的二进制目的地址序列也必须跟随数据位置的改变而改变,这样才能正确提取出第二次归并所需的初始控制信息. 因此,在图 6 双竖线右边的控制路径中,初始各数据元素对应的二进制目的地址序列除最高位外(已经归并完成),都必须按照第一次归并操作所构建的网络路由通过一次,如图 6(a_1)和图 6(a_2)所示. 这样,在进行第二次归并操作时,就能够像第一次归并操作一样,将左、右各自部分内二进制目的地址序列次高位的左边 2-bit 数据按位取反后,与右边序列并置,以生成第二次归并操作的初始控制序列. 如图 6(b)所示,Initial Control Bits = (0000)和(1010).

第三次归并前,各数据元素最低位二进制目的地址序列首先经过第二次形成的归并网络,完成二进制地址最低位的联动,如图 6(b_1)所示. 然后将联动后独立的 4 个 2-bit 序列中左边 1-bit 二进制地址取反后并置为一个 4-bit 的路由控制信息,直接用于 Inverse Butterfly 网络第一级交叉开关为:(1 0 0 0)→(0 1 1 1).

上述初始控制信息生成的流程图如图 7 所示,其中 $R_{\lg N-1}$ ~ R_0 为 N-bit 寄存器,依次存储了从高位到低位各元素二进制目的地址序列,$Cq(0<q\leqslant 2^T)$ 为对应

子蝶网络的初始控制信息,C_{2^r} ~ C_1 从左至右依次对应 sub-ibfly$_{2^r}$ ~ sub-ibfly$_1$. 图 7 中在计算第 T 次归并所需的初始控制信息 $C_{\lg(N)-T}$时,需要对参数 q 进行了奇偶判定,当 q 为偶数时,各元素二进制目的地址序列 $R_{\lg(N)-T}[q\times N/2^T-1:0]$按位取反记为 ~$R_{\lg(N)-T}[q\times N/2^T-1:0]$;$q$ 为奇数时,则保持不变. 当 $T=\lg N$ 时,无需调用算法 1,第一级路由控制信息可直接由 R_0 中所有奇数位置比特数据并置后按位取反而成,而剩余各级所有开关路由控制信息全部设置为“0”.

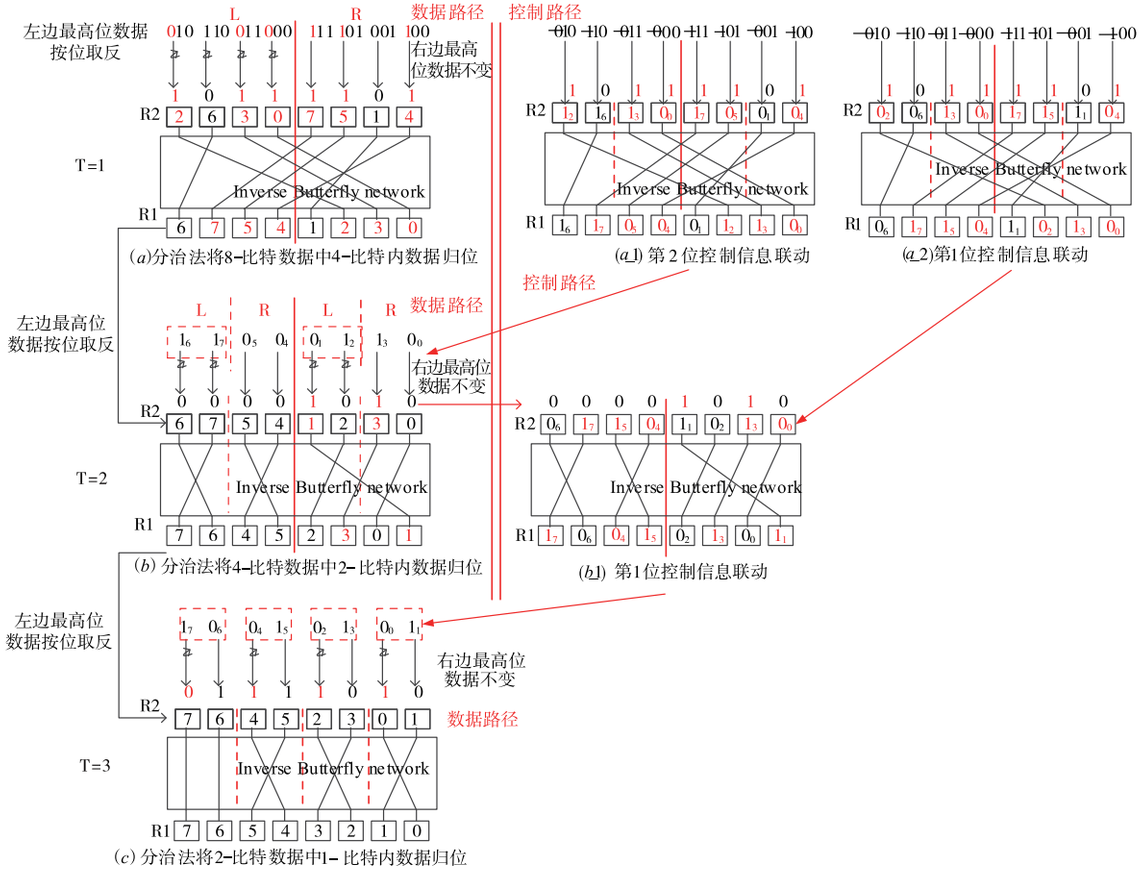


图6　Inverse Butterfly网络初始控制信息生成方案

4　关键模块硬件设计

由第 3 节分析知,实现任意置换操作需要两个步骤:第一是生成抽取-交换操作所需的初始控制信息如图 7 所示;第二是在初始控制信息的作用下生成抽取-交换操作所需的适配于 Inverse Butterfly 网络各级开关的选路信息,如图 5 所示. 其中初始控制信息的实现,仅需根据不同阶段 T 将各数据对应的二进制目的地址序列进行有条件取反,它的硬件实现电路由一级反向器和选择控制器构成,设计原理简单,因此不再讨论. 本节重点根据算法 1,以 N = 16-bit 为例,设计了一种可重构抽取-交换功能单元,如图 8(a)所示. 它由 4 级 Inverse Butterfly 网络和可重构选路信息生成电路组成,其中可重构选路信息生成电路是整个单元的核心模块,它能够根据归并阶段 T 和初始控制信息 R3,实时地生成用于完成抽取-交换操作网络各级开关所需的选路控制信息. 该模块可再细分为可重构并行比特连加电路(Reconfigurable Parallel Popcnt,RPP)、循环移位后末尾取反电路(Rotation left then reverse the least significant bit,RLTR)和调整电路 Adjust.

RPP 电路用来确定 RLTR 电路循环左移后末位取反的次数,内部结构如图 8(b)所示,其中每个圆点都是一个加法电路,用于计算 R3 中“1”的数目. 例如,当 T =1 时,RPP 电路中的 16-bit 控制序列 R3 将被分成 2 个独立的 8-bit 序列. 根据算法 1,网络第 1 级需要分别计算 R3[14:8]、R3[12:8]、R3[10:8]、R3[8:8]、R3[6:0]、R3[4:0]、R3[2:0]、R3[0:0]的加和值. 从图 8(b)中的红色方框部分可知,Sum_pc[14] = Popcnt(R3[14:8]) = Popcnt(R3[14:12]) + Popcnt(R3[11:8])正好对应了第一级最左边交叉开关所需的加和值;当T

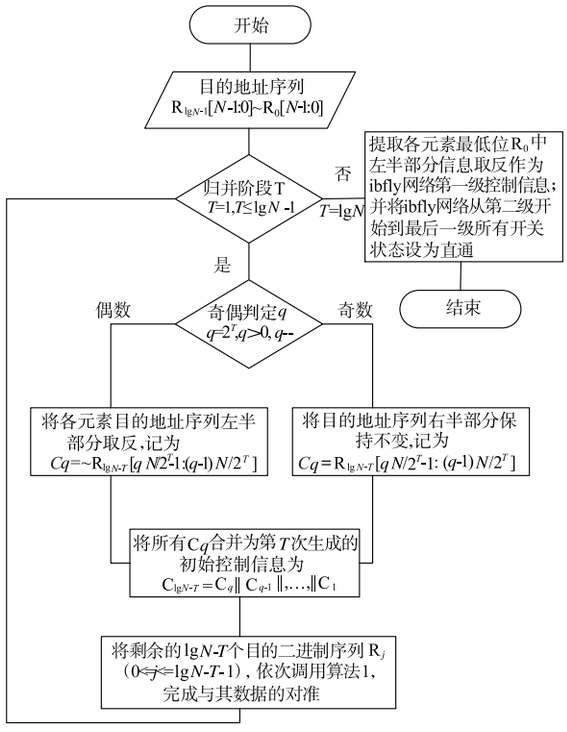


图7 初始控制信息生成方案流程图

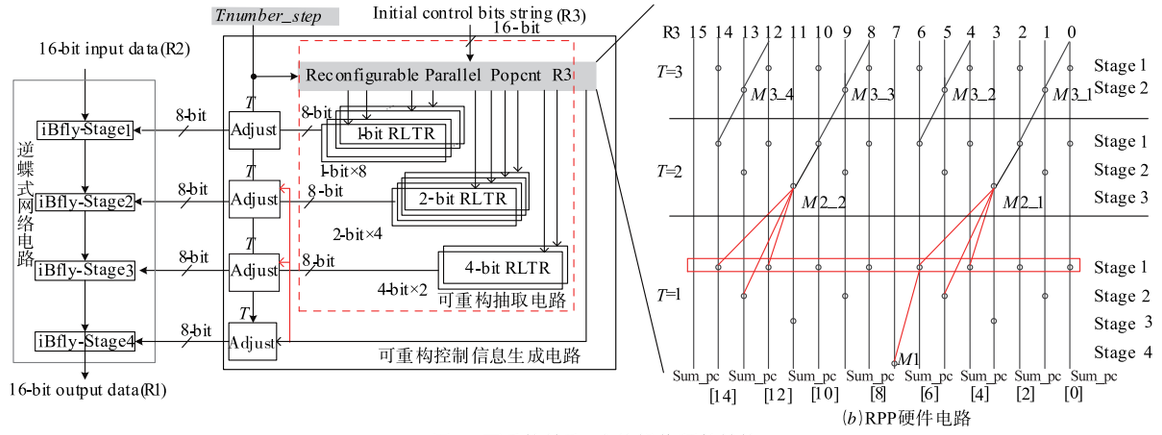
分别为 2 和 3 时,16-bit 控制序列 R3,将被相应地分成 4 个 4-bit 和 8 个 2-bit 序列进行并行抽取-交换操作,各级 Sum_pc 的计算过程与 $T=1$ 时相似,不同点仅在于加和的起始位置不同.

RLTR(1^k,m)电路完成的功能是将一个有 k 个“1”的序列,进行循环左移后末尾取反 m 次. 该电路的运算结果以 $2k$ 为周期. 当 $k=4$ 时,初始序列为“1111”,若 m 从 0 – 7 变化,RLTR 函数运算结果如表 1 所示.

表 1 $M=4$,RLTR 变化规律

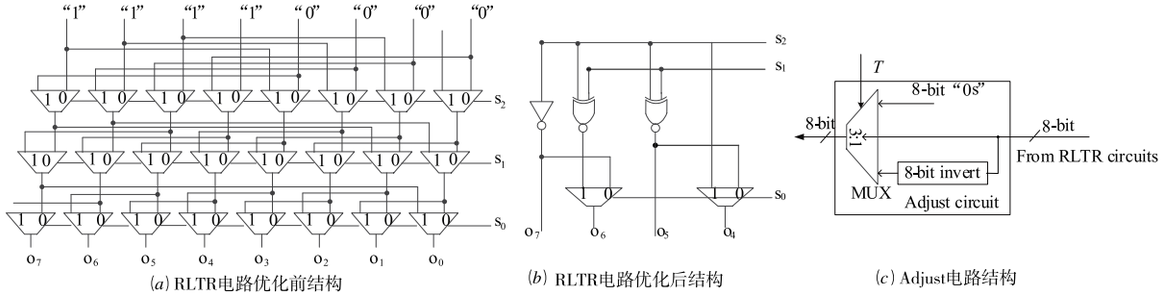
m mod 8	0	1	2	3	4	5	6	7
RLTR	1111	1110	1100	1000	0000	0001	0011	0111

它与 8-bit 序列“1111_0000”循环移位后取高四位输出值的结果相同,因此将 RLTR 操作首先用对数移位器初步实现如图 9(a)所示. 由于其端口输入为固定值,那么可以进一步进行布尔逻辑优化以精简冗余,优化后结构如图 9(b)所示. 与初始结构相比,优化后的电路面积减少约 85%,延迟减少约 20%. Adjust 电路将根据归并所处的阶段 T 和 RLTR 函数电路计算出的 8-*bit* 序列,生成最终用于抽取 + 交换操作的 Inverse Butterfly 路



(*b*)RPP硬件电路

图8 可重构抽取–交换操作硬件结构



(*a*) RLTR电路优化前结构 (*b*) RLTR电路优化后结构 (*c*) Adjust电路结构

图9 8–bit RLTR及Adjust电路结构

由控制信息,其结构如图 9(*c*)所示. 8-bit RLTR 函数电路输出序列进入 Adjust 电路后分成两个部分,一部分直接进入 3 选 1 MUX 中,另一部分进入由 8 个反向器(inverter)组成的反向电路中,完成序列的按比特反序即 ~RLTR,之后再输入到 MUX 中. MUX 最后一个输入端口为 8-bit 常数值"0",用于屏蔽置换时不需要级数. 它们在归并次数 T 的作用下,选出 Inverse Butterfly 网络各级开关所需的最终路由控制信息.

本节将 N 选定为 64-bit,构造了基于 Inverse Butterfly 网络的比特置换单元(bit-permutation unit based on Inverse Butterfly,IBPU). 然后将它在 SMIC 65nm [16] 工艺下进行了逻辑综合,综合时参数为最慢工艺角、最低温度(-40 摄氏)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 2 所示:

表 2　16-bit 任意置换单元硬件性能对比

置换网络类型	网络延迟(ns)	资源消耗(NAND gates)	静态/动态置换指令条数	工艺	等效延迟(ns)	代价(静态/动态)
Hilewitz' GRP[9]	1.74	19.7K	6/*NA*	TSMC 90	1.34	158.39/*NA*
Change's network[11]	1.21	15.76K	6/*NA*	SMIC 65	1.21	114.42/*NA*
Kolay's network[4]	0.75	2.7K	64/64	TSMC 90	0.54	93.31/93.31
Our IBPU	0.85	10.7K	6/21	SMIC 65	0.85	54.57/191

"*NA*"表示不支持该操作

表 2 中,文献[4]和文献[9]设计的比特置换单元位宽均为 64-bit,且基于 TSMC 90nm 工艺综合完成. 为了尽可能地降低由于工艺偏差而导致的性能评估偏差,本节根据文献[17]中针对不同工艺下的延迟换算关系:$T_{new}=l\times T_{old}$,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 2 中第 2 列网络延迟数据进行了等效换算,结果如第 6 列所示.

由表 2 中资源消耗可知:本文设计的比特置换单元 IBPU 资源消耗为 10.7 千门,与具有相似结构的文献[9]和文献[11]相比,资源消耗仅为它们的 54.3% 和 67.9%. 与具有不同结构,但能够完成相同功能的文献[4]相比,本文 IBPU 的资源消耗是其 4 倍. 由表 2 中等效延迟可知:本文设计的比特置换单元 IBPU 等效延迟为 0.85ns,分别为文献[9]和文献[11]等效延迟的 63.4% 和 70%. 但比文献[4]的等效延迟大 1.57 倍.

从功能角度出发,当完成任意静态置换操作时,本文设计的 IBPU 与文献[9]、文献[11]一样,均能够在 $\lg N=6$ 条($N=64$)指令内完成,而文献[4]则需要 $N=64$ 条指令才能够完成. 当完成动态置换操作时,由于文献[9]和文献[11]的初始控制信息选路算法较为复杂,一般采用软件预计算的方式,因此它们并不支持这样的操作,表 2 中标注为"*NA*". 而文献[4]利用各数据元素目的二进制地址的相互关系,采用比特交换的方式依次完成各数据位置的相互交换,因此无论是静态还是动态置换,它并不加以区分均能够支持. 为了更加客观地评价各实现方式的综合性能,本文提出了代价 = (资源消耗 × 等效延迟 × 静态/动态指令条数)这一参数. 由表 2 最后一列所示的代价可知,当 4 种置换单元完成静态置换操作时,本文设计的 IBPU 代价值最低,仅为 54.57,分别是文献[9]、文献[11]和文献[4]的 34.45%,47.7% 和 58.48%,具有较为明显的优势. 当完成动态置换操作时,只有文献[4]和本文设计的 IBPU 支持该类型的操作,但 IBPU 的代价是文献[4]的一倍. 通过进一步分析发现,本文 IBPU 在实现动态置换操作时,指令条数与位宽 N 的关系为:$(\lg N\times(\lg N+1))/2$,复杂度为 $O(\lg^2 N)$ 低于文献[4]所需的 N 条指令 $O(N)$. 因此,当置换位宽 N 增大时,IBPU 的代价涨幅要低于文献[4],那么当 $N=256$ 或者更大时,IBPU 的代价最终会优于文献[4].

5　结束语

本文基于 Inverse Butterfly 网络结合分治的思想,提出了一种能够适于硬件实现任意置换的选路算法. 该算法能够对各输入数据对应二进制目的地址序列,按照从高位到低位的顺序依次进行归并,并在 $\lg N$ 条指令内完成任意静态置换操作,在 $O(\lg^2 N)$ 条指令内完成任意动态置换操作. 实验结果表明,与同类设计相比,本文构造的 IBPU 单元在完成静态置换时代价很低,其综合性能具有明显优势. 在完成动态置换操作时的代价虽然较高,但随着置换位宽 N 的增加,其代价的增长幅度较低,因此稳定性较好. 总之,本文基于 Inverse Butterfly 网络提出的置换选路算法以及构造的比特置换单元 IBPU,不仅具有较好的性能,而且还有效扩展了该网络架构所支持的功能,是对该架构下扩展任意置换操作的一次有益探索.

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] 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.
[3] Ao T,He Z,Dai K. Low-cost bit permutation circuit with concise configuration rule[A]. Proceedings of the International Multi Conference of Engineers and Computer Scientists[C]. Hong Kong:International Association of Engineers,2015. 158 – 160.
[4] 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.
[5] Dimitrakopoulos G ,Mavrokefalidis C,Galanopoulos K,et al. Fast bit permutation unit for media enhanced microprocessors [A]. 2006 IEEE International Symposium on Circuits and Systems(ISCAS 2006)[C]. Greece:IEEE,2006. 49 – 52.
[6] 陈国良,韩雅华. Benes 网络的半自动选路法[J]. 计算机学报,1990,3(3):161 – 173.
Chen Guo Liang,Han Ya Hua. The semi-self-routing algorithms for Benes network[J]. Chinese Journal of Computers,1990,3(3):161 – 173. (in Chinese)
[7] Shan W,Chen X,et al. A novel combinatorics-based reconfigurable bit permutation network and its circuit implementation [J]. Chinese Journal of Electronics,2015,24(3):513 – 523.
[8] 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.
[9] 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):145 – 169.
[10] Chang Zhongxiang,Hu Jinshan,MA Chao. Research on shifter based on ibutterfly network[A]. 17th China Computer Federation[C]. Xi Ning,China:2013. 92 – 100.
[11] 常忠祥,戴紫彬,李伟,等. 高速比特置换实现技术研究[J]. 小型微型计算机系统,2015,36(3):627 – 630.
Chang Zhong Xiang,Dai Zi Bin,Li Wei,et al. High-speed bit permutation implementation technology[J]. Journal of Chinese Computer Systems,2015,36(3):627 – 630. (in Chinese)
[12] 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.
[13] Bistouni F,Jahanshahi M. Pars network:A multistage interconnection network with fault-tolerance capability[J]. Journal of Parallel & Distributed Computing,2015,75:168 – 183.
[14] 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.
[15] Intel Corporation,Intel® 64 and IA-32 Architectures Software Developer's Manual[S]. 2015
[16] Semiconductor Manufacturing International Corporation. SMIC 65nm Logic Process Standard Cell Library Databook[S]. 2012.
[17] LIU B,BAAS BM. Parallel AES encryption engines for many-core processor arrays[J]. IEEE Transactions on Computers,2013,62(3):536 – 547.

作者简介

马 超 男,1988 年生于陕西西安. 国家高性能集成电路(上海)设计中心,博士. 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.

南龙梅(通信作者) 女,1981 年生于陕西韩城. 信息工程大学教员,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,高性能互连网络设计.
E-mail:lnan13@ fudan. edu. com