可重构分组密码指令集处理器自动映射方法研究

李 盛*①② 戴紫彬①
①(解放军信息工程大学 郑州 450001)
②(铁道警察学院 郑州 450053)

摘 要：计算资源与寄存器资源分配是可重构处理器自动并行映射的重要问题，该文针对可重构分组密码指令集处理器的资源分配问题，建立算子调度参数模型和处理器资源参数模型，研究了分组密码并行调度与资源消耗之间的约束关系；在此基础上提出基于贪婪思维、列表调度和线性扫描的自动映射算法，实现了分组密码在可重构分组密码指令集处理器上的自动映射。通过可用资源变化实验验证算法并行映射的有效性，并对AES-128算法的映射效果做了横向对比验证算法的先进性，所提自动映射算法对分组密码在可重构处理中的并行计算研究有一定的指导意义。

关键词：可重构分组密码指令集处理器；自动映射；资源分配；列表调度；线性扫描
中图分类号：TN918.2; TP314 **文献标识码**：A **文章编号**：1009-5896(2021)09-2526-08
DOI: 10.11999/JEIT200372

Research on Automatic Mapping Method of Reconfigurable Block Cipher Instruction Set Processor

LI Sheng①② DAI Zibin①
①(*The PLA Information Engineering University*, *Zhengzhou* 450001, *China*)
②(*Railway Police College*, *Zhengzhou* 450053, *China*)

Abstract: The allocation of computing resources and register resources is an important issue for automatic parallel mapping of reconfigurable processors. In this paper, for the resource allocation of reconfigurable block cipher instruction set processors, an operator scheduling parameter model and processor resource parameter model are established, and the constraint relationship between the parallel scheduling of block ciphers and resource consumption is studied; Consequently, an automatic mapping algorithm based on greedy thinking, list scheduling and linear scanning is proposed to realize the block cipher atuomatic mapping on the reconfigurable block cipher instruction set processor. The experiment verifies the effect of the algorithm's parallel scheduling under different resource constraints, and the contrast of AES-128 algorithm's mapping effect is made to verify the progress of the algorithm, which obtains certain significance for the parallel computing research of block ciphers in reconfigurable processing.

Key words: Reconfigurable block cipher instruction set processor; Automatic mapping; Resource allocation; List scheduling; Linear scan

1 引言

密码算法的实现是高性能计算处理器设计与应用的重要目标之一，可重构分组密码处理器将可重构计算技术用于实现分组密码算法，其功能单元可配置与并行计算资源丰富的特点有效地提升了分组密码运算速度，该研究领域产生了可重构分组密码指令集处理器和多核可重构分组密码指令集处理器[1–5]等研究成果。这些研究均采用指令配置和指令控制方式实现了可重构资源在时间和空间上的分配，提高了分组密码计算效率。由于可重构处理器的计算资源和存储资源在时间和空间上需要结合应用程序的特征，增加了算法映射的难度[6,7]。在这样的背景下，分组密码算法在可重构处理器上的映射过程常采用手动方式完成。如何结合可重构处理器的结构特点和资源特点，自动为分组密码分配可重构资源成为重要的研究点。

可重构处理器自动映射所要实现的目标是利用编译器后端自动为计算过程分配可重构资源，最大限度地利用并行计算资源，研究的要点是计算可重

收稿日期：2020-05-12；改回日期：2021-01-12；网络出版：2021-02-02
*通信作者： 李盛 lisheng@rpc.edu.cn

构处理器中的资源约束条件和应用程序的依赖性约束条件。为具体的应用程序分配可重构资源属于NP问题，目前关于该问题有较多的切入点和研究成果，如文献[8]将性能和功耗作为资源分配的衡量指标，利用加权度量模型完成了资源的分配；文献[9]通过对可重构密码阵列的结构进行分析并建立参数化模型，建立互连资源与资源分配的关系并利用蚁群算法进行求解；文献[10]提出了利用深度强化学习对数据流图的映射问题进行求解，从整体上解决资源分配问题。上述成果在不同的假设条件下，通过启发式算法或智能优化算法求解资源分配问题，但该类型研究成果针对的可重构处理器均是阵列结构，阵列结构有丰富的互联资源和功能单元本地存储资源，因此针对阵列结构的映射研究成果不能直接应用于可重构指令集处理器的资源分配问题。目前有一定数量关于多发射型可重构分组密码指令集处理器资源分配问题的研究[11,12]，实现了分组密码算法并行性能的提升，但是研究过程中以分组密码算法的运算特点与计算资源的结合为主，没有考虑数据存储资源约束对映射的约束。文献[13]提出了MCGA算法，着重研究了指令扩展与可重构计算资源分配问题，在传统编译后端构造方法基础上对指令生成算法进行了改进，将可重构指令集处理器的资源分配问题归纳为多图着色问题，如果在此研究基础上考虑寄存器资源的分配与计算资源分配之间的关系，则会有更广的适用性。

综合对上述研究的分析和借鉴，本文分析了分组密码算法的特点、可重构分组密码指令集处理器的结构与资源特点，建立了算子调度参数模型和处理器的计算资源、寄存器资源关系模型，量化调度与映射过程中计算资源、寄存器资源之间的约束关系；在此基础上整体考虑计算资源和寄存器资源的分配问题，结合贪婪思维、列表调度方法和线性扫描方法完成了调度与映射过程中计算资源和寄存器资源的自动计算与分配，实现分组密码算法在可重构指令集处理器中的自动映射。

2　分析与参数建模

2.1　分组密码算子调度参数模型

分组密码算法的核心是轮运算，其设计思想是利用“混乱”和“扩散”运算在SP, FEISTEL, LM, MISTY 4种典型的网络结构中通过轮运算反复执行隐藏明文与密文之间的统计信息，影响分组密码算子映射的两个因素是：

(1) 算子类型和排列顺序多样，增加了映射过程中的并行分配计算资源的复杂性；

(2) 算子的数据依赖关系复杂，需要综合考虑算子输入和输出变量的生命周期，增大了映射过程中的寄存器资源分配的难度。

考虑上述两个因素的影响，如果想要最大限度地发挥分组密码可重构处理器的并行计算能力和可重构单元的灵活性，需要建立适合分组密码算法的调度分析参数模型。常用的调度分析参数模型为数据流图DFG(Data Flow Graph), $G=(V,E)$, V为算子结点集，E为数据传输有向边集，$\forall v_i,v_j\in V$, $e(v_i,v_j)\in E$表示结点v_i到v_j的有向边。考虑分组密码算法特点对调度与映射过程的影响，在DFG图中增加每个算子结点的属性描述，即$G=(V,E,P)$，定义P为结点属性集合，则$\forall v_i\in V$, p_i表示结点v_i的属性如式(1)所示

$$p_i=(\mathrm{IW}_i,\mathrm{OW}_i,\mathrm{Id}_i,\mathrm{Od}_i,\mathrm{TP}_i,\mathrm{Or}_i,\mathrm{Sd}_i,\mathrm{Ss}_i,\mathrm{Sp}_i)\quad(1)$$

其中，IW_i表示结点输入操作数的数据位宽，OW_i表示结点输出数据位宽，Id_i表示该点的前驱结点数目，Od_i表示该结点的后继结点数目，TP_i表示结点的密码运算类型，Or_i表示结点的编号，Sd_i表示结点在调度过程中所处的控制步，Ss_i为结点在初始调度后的调整自由度，表示调度过程中结点所处的最大深度与最小深度之间的差值，Sp_i表示结点在调度过程中资源分配优先级。

2.2　可重构分组密码指令集处理器资源参数模型

本文研究的映射对象为此前项目已开发的可重构分组密码指令集处理器[4]，其执行分组密码加密计算的精简体系结构如图1所示。处理器采用了分簇式多发射VLIW结构以适应分组密码算法长位宽运算特点，与数据运算相关的资源主要为功能单元RFU(Reconfigurable Function Unit)和寄存器。每个簇中有一个被设计为可配置电路结构的功能单元RFU，处理器通过调用配置文件实现RFU的计算功能和计算位宽重构。满足了不改变体系结构也能实现各种常用类型、各种常用位宽的分组密码算子要求。RFU具有如下特征：

(1) RFU通过配置指令写配置文件实现运算功能重构，可重构为常用的8种分组密码运算功能单元；RFU的运算位宽在配置时通过配置参数设定，可实现亚字(4, 6, 8 bit)、字(32 bit)长度的计算；无论运算位宽大小，每个控制步中一个功能单元只能被配置为一种分组密码运算功能单元；

(2)完成配置的功能单元在运算指令的控制下对数据进行加密计算，一条VLIW运算指令的执行为一个控制步，RFU的计算都在一个控制步内完成，RFU之间没有直通互连结构，只能通过读写寄存器互相通信；由于RFU在每个控制步的功能

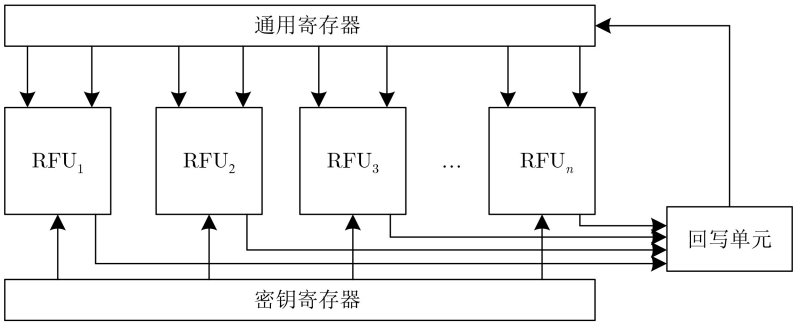


图 1 可重构分组密码指令集处理器资源精简模型

和位宽可以发生改变，处理器的计算资源被视为局部资源。

处理器的寄存器资源分为通用寄存器和密钥寄存器，前者存储明文数据、运行时的中间变量数据和运算后的密文数据，后者存储运行时所需要的密钥数据。寄存器资源有如下特征：

(1) 通用寄存器允许RFU任意读取两个操作数，并通过回写单元向通用寄存器中写入一个操作数；密钥寄存器只允许RFU读取数据，其数据由外部电路写入。RFU在每个控制步从通用寄存器和密钥寄存器读取数据完成运算，并通过回写功能单元回写单元将数据写回寄存器中；

(2) 由于计算资源之间的数据传递通过读写寄存器操作实现，其运算结果对寄存器资源的占用时长不同，因此对于一个分组密码算法运算过程，寄存器资源被视为全局资源。

2.3 资源消耗与资源约束关系分析

根据上述分析，从资源消耗方式角度对可重构分组密码指令集处理器进行计算资源和寄存器资源参数化建模。一个加密操作至少需要一个RFU实现，如果同一个控制步中有多个相同类型的操作，则可以根据运算位宽大小进行算子合并，如在同一个控制步中的两个同类型的亚字加密运算，可以通过配置一个RFU进行亚字并行计算。定义式(2)表示处理器的资源总量参数描述

$$\mathrm{RES}=(\mathrm{RFUN},\mathrm{REGN},\mathrm{TS}) \tag{2}$$

其中，RFUN表示处理器中可重构功能单元集合，REGN表示处理器的寄存器的数据总容量集合，TS为总的控制步集合。

根据2.1节对RFU的分析，在控制步$t,t\in\mathrm{TS}$中功能单元的占用量取决于两个因素：(1)同类型算子的运算位宽总量；(2)非同类型算子的运算位宽总量。定义$\mathrm{ORFU}(t)$为控制步$t,t\in\mathrm{TS}$中算子对可重构功能单元的占用量，$\{V(t),W(t)\subseteq V|\forall v_i\in V(t),\forall w_j\in W(t),\mathrm{Sd}_i=\mathrm{Sd}_j=t,\mathrm{TP}_i\neq\mathrm{TP}_j\}$，$\mathrm{ORFU}(t)$如式(3)

$$\mathrm{ORFU}(t)=\left\lceil\frac{\sum\limits_{V(t)}\max(\mathrm{IW}_i,\mathrm{OW}_i)}{32}\right\rceil+\left\lceil\frac{\sum\limits_{W(t)}\max(\mathrm{IW}_j,\mathrm{OW}_j)}{32}\right\rceil \tag{3}$$

由于寄存器资源为全局变量，所以其具体的占用量需要考虑两个因素：(1)当前控制步中各个算子输出变量和数据位宽；(2)之前控制步中算子输出变量生命周期和数据位宽。定义$\mathrm{OGREG}(t)$为控制步$t,t\in\mathrm{TS}$中寄存器资源的占用量，定义集合$\{V(t)\subseteq V|\forall v_i\in V(t),\mathrm{Sd}_i=t\}$为控制步$t$中的算子结点集合，则寄存器资源占用量如式(4)所示。在式(4)中$\mathrm{PA}_i(t)$表示生命周期系数，用来衡量控制步t之前的所有控制步中算子的输出变量在控制步t中的具体数值，如果其值不为零，则表示某算子的输出变量在控制步t需要占用寄存器。

$$\left.\begin{aligned}\mathrm{OGREG}(t)&=\sum_{V(1)}^{V(t-1)}\left\lceil\frac{\mathrm{OW}_i\times\mathrm{PA}_i(t)}{32}\right\rceil+\sum_{V(t)}\left\lceil\frac{\mathrm{OW}_i}{32}\right\rceil\\ \mathrm{PA}_i(t)&=\left\lfloor\frac{\mathrm{Sd}_i+\mathrm{Ss}_i+1}{t}\right\rfloor\end{aligned}\right\} \tag{4}$$

在加密运算过程中，如式(5)所示任意控制步中的密码算子对资源的消耗均不得超过资源总量。

$$\mathrm{ORFU}(t)\le|\mathrm{RFUN}|\wedge\mathrm{OGREG}(t)\le|\mathrm{REGN}| \tag{5}$$

3 可重构指令集处理器自动映射算法

可重构处理器的自动映射过程关键的研究点在于资源的分配方法，其主要思想为在寄存器资源消耗不超过处理器资源限制和不破坏分组密码算子之间数据依赖性的前提下，在程序的每一控制步尽可能将更多的计算资源分配给密码算子，提升并行计算资源利用率。为了实现上述目标，本文利用贪心策略，基于列表调度与线性扫描算法思想设计了自动映射算法如图2所示。

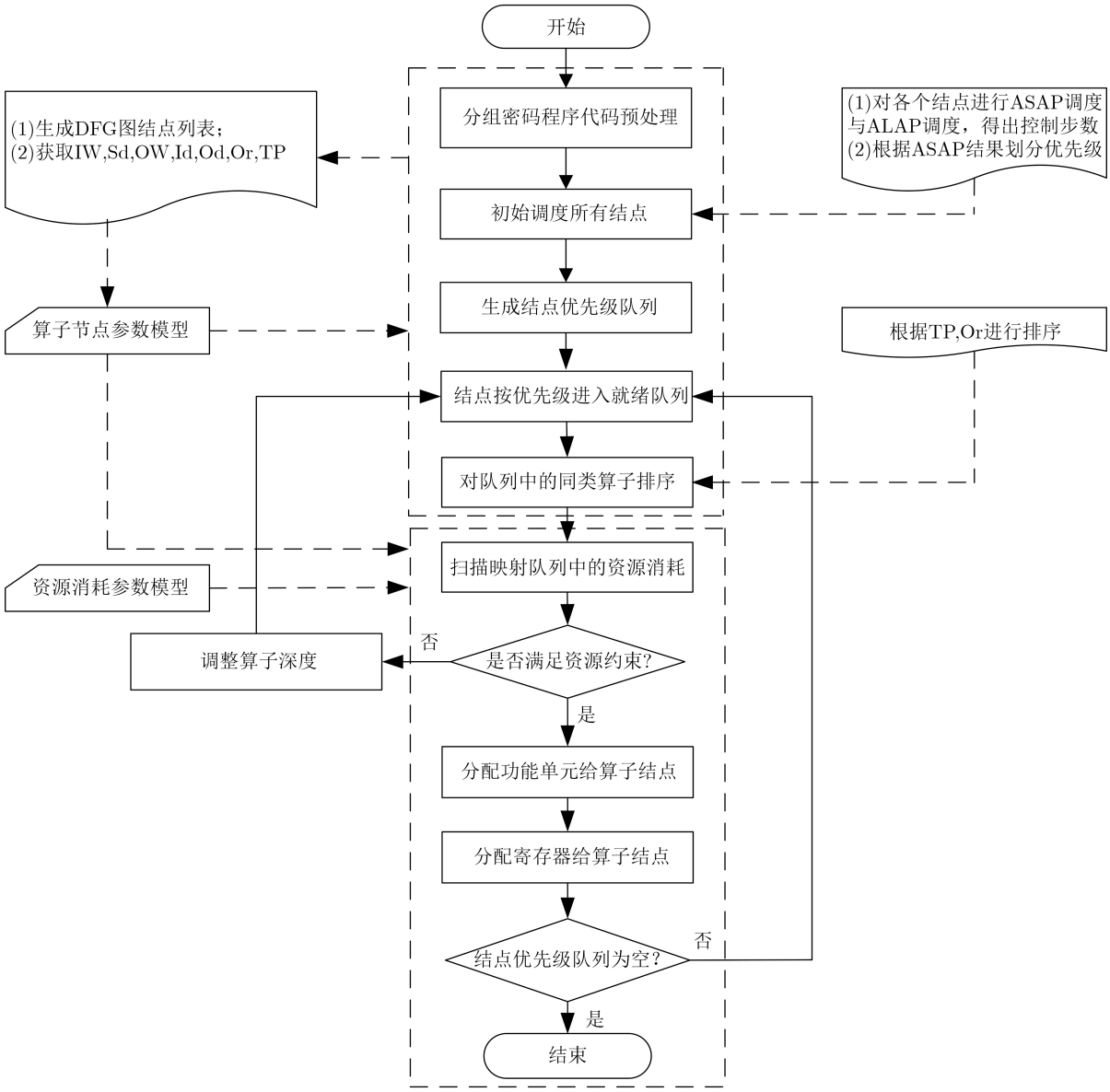


图 2 自动映射算法流程图

3.1 初始化调度

本部分首先对分组密码算子结点进行预处理和初始调度，获取结点的各种属性数值，生成结点就绪队列。初步调度中，对DFG图中的每个结点首先进行了ASAP调度和ALAP调度[14]，计算出了每个结点的Sd和Ss，生成结点的优先级队列。其中Ss为0表示关键路径结点，如果调整则整个运算过程的控制步会变长，不为0表示非关键路径结点，在Ss范围内调整不影响总控制长度。对调度后的结点进行排序，按照最早的发射时间生成就绪结点队列。ASAP调度用来计算结点在理想条件下的最早启动时间，ALAP用来计算不增加控制步情况下结点的最迟启动时间，这两种调度算法在计算时不考虑资源限制，仅仅是在确保结点依赖关系下进行的理想化调度，也不考虑运算类型，所以就绪队列并不能直接用来作为资源分配优先级的依据，需要对调就绪结点队列进行按优先级排序。基于以下几点考虑，算子结点的映射优先级，应先由就绪队列中的算子类型、编号顺序决定：

(1)每个控制步中结点数量应该尽可能多，以便占用更多的功能单元。所以在生成就绪队列时，应该以结点最早的启动时间为准；

(2)每个控制步中一个功能单元至少被一个算子结点占用，所以应该尽可能地将同类型的亚字算子由一个功能单元完成计算，所以应尽可能将相同类型的算子结点排列在一起；因此在对就绪队列进行排序时，优先级应由结点类型和结点在程序中的初始排序决定，如式(6)所示

$$\mathrm{PR} = \mathrm{TP} - \mathrm{Or} \tag{6}$$

按照这样的优先级确定方式，在映射队列中，同算子按照最早的发射时间、相同的运算类型排列，生成了结点映射队列。

3.2 资源分配与结点调度调整

在对映射队列进行资源分配时，采用线性扫描算法思想对队列中结点逐个扫描，利用之前章节所设计的计算资源与寄存器资源参数化模型进行计算，满足资源分配约束条件的点则进行计算资源和寄存器资源的分配，不满足约束条件的结点则放回就绪队列，并调整深度属性数值，重新计算该结点的优先级，对原结点优先级排序队列进行插入排序，然后再继续后续步骤。自动映射算法描述如表1所示。

对于满足资源分配条件的结点，进行计算资源和寄存器资源的分配，计算资源按照同类算子占用相同功能单元的原则进行分配；寄存器资源的分配则首先需要更新结点的生命周期形成记录表，充分利用结点的life hole，使得结点优先获得其前驱结点变量所占用的寄存器。

表 1 自动映射算法

```
输入：serial_code                                   //分组密码串行程序代码
输出：映射结果
Initial_schdule(list)                               //对结点进行调度
{
    for each node v_i in list;
        Sd_i = ASAP(v_i);
        Ss_i = Sd_i – ALAP(v_i);
    Endfor
}
Select_ReadyQ(list)
{
    for each node v_i in list;
        merge_sort(list,Ss_i);                      //进行归并排序
    endfor
    select the v_i with smallest Sd_i;
    v_i → ready_queue;                              //将启动时间最早的结点放入就绪队列
}
Mapping_sequencing (list)                           //对就绪队列中的点按照type和order排序，将同类型算子结
{                                                   点排列在一起
    for each node v_j in list
    Sp_j = TP_j–O_j;
    endfor
    merge_sort(list, Sp_j);                         // 进行归并排序
    return list;
}
BEGIN
compiler_infrastructure(serial_code)→nodelist;      //利用编译基础设施预处理串行代码
S_Q = Initial_S(nodelist);                          //对结点进行初始调度
while (S_Q≠∅)
{
  R_Q = Select_ReadyQ(S_Q);                         //生成就绪队列
  M_Q = Mapping_sequencing (R_Q);                   //将同类型结点放一起
  for each v_k in M_Q
  if(ORFU(v_k)> RFUN or OREG(v_k)>REGN);
        Sd_k = Sd_k –1;
        v_k→S_Q;
    else
        update lifetime_table;
        Allocate_RFU(v_k);
        Allocate_Reg (v_k,lifetime_table);
        delete v_k in M_Q;
      endif
  endfor
}
END
```

在自动映射算法中，算法的时间复杂度主要和算子的数量有关，如果算子数量为n，则最坏情况下初始调度的时间复杂度为$O(n)$，就绪队列生成的时间复杂度为$O(n\log_2 n)$，映射队列生成的时间复杂度为$O(n\log_2 n)$，资源分配的时间复杂度为$O(n)$，所以自动映射算法的时间复杂度为$O(n+n\log_2 n+n\log_2 n+n)=O(2n+2n\log_2 n)$，约等于$O(n\log_2 n)$。

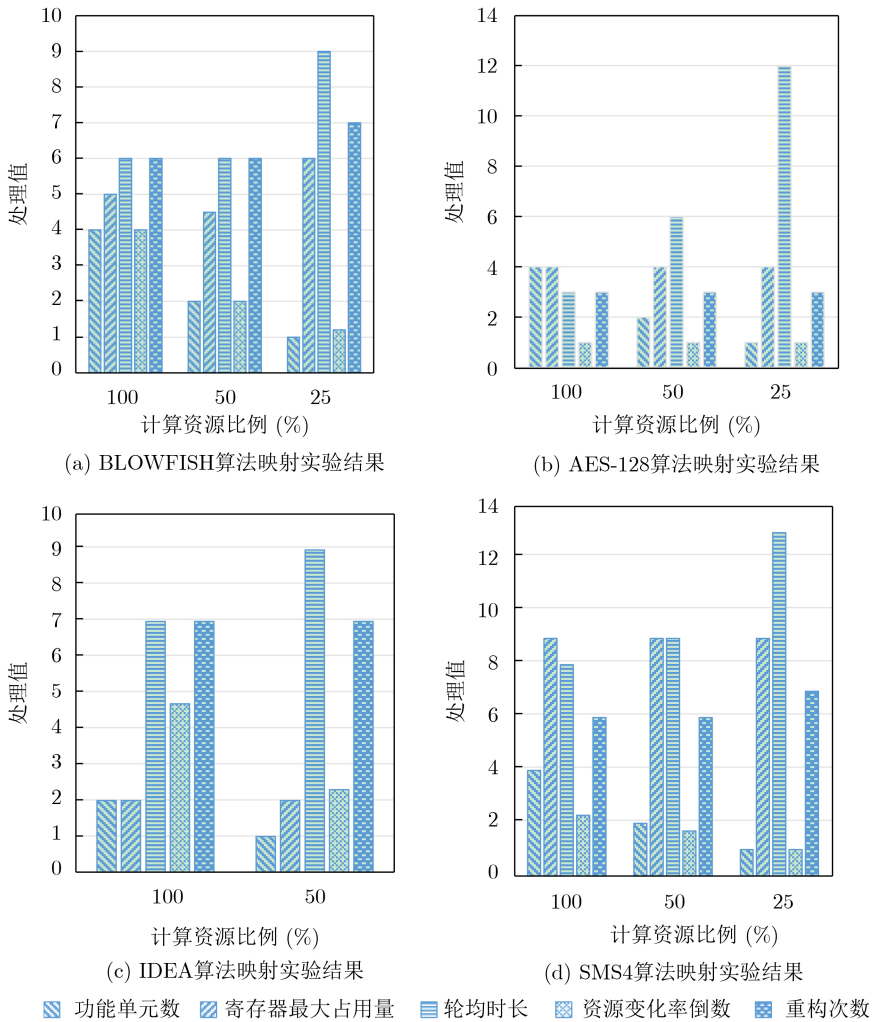
4　实验分析

4.1　自动映射算法有效性实验

本文实验利用C++编写自动映射算法仿真程序，在2.4 GHz主频的CORE(TM)通用计算机进行实验。编写BLOWFISH, AES-128, SMS4, IDEA算法串行程序代码，利用项目组已开发的编译基础设施将分组密码算法应用程序代码编译为算子级代码，由本文研究自动映射算法仿真实验程序对算子级代码进行调度和资源分配，模拟器可根据处理器资源分布情况，对整个处理器的运行过程进行模拟并输出运行结果报告。然后采用项目组已开发的模拟器观察实验数据。实验过程首先计算RFU资源和寄存器资源不受限制条件下，即理想条件所需要的两类资源数量，然后将计算资源降为理想条件的50%和25%进行调度和资源分配，记录实验中寄存器资源的最大占用量、程序运行的时钟周期数、每个时钟周期分配的计算资源类型及数量、重构次数等数据。

程序在计算资源数量变化条件下运行的速度反映了自动映射算法的并行效果；重构次数反映了读配置文件次数，每次重构都可以视为可重构资源数增加了1倍；计算资源变化率反映了计算资源的利用效果，其数值计算等于功能单元配置次数除以重构次数与功能单元的乘积；最大寄存器资源占用量反映了程序运行过程中对寄存器所造成的最大压力，同时也是保证最大并行效果的最低要求。由于各个实验数据的量纲不同，为了方便显示，程序运行的快慢用总时钟周期除以相应分组密码的轮运算轮数所得的轮均时长表示，资源变化率则使用其倒数表示。各分组加密算法的实验结果如图3所示。

通过实验观察，在所测试的分组密码算法中，



(a) BLOWFISH算法映射实验结果　(b) AES-128算法映射实验结果

(c) IDEA算法映射实验结果　(d) SMS4算法映射实验结果

图 3 4种分组密码实验结果

随着计算资源的减少，轮均时长也逐渐增加，尤其是对于并行特征较为明显的调度AES-128算法，轮均时长与计算资源呈反比例增长，体现了自动映射算法的并行调度有效性。寄存器资源主要用于存储多轮加密运算的中间结果，在加密算法运算位宽固定不变的前提下，实验数据显示AES-128, SMS4, IDEA算法的寄存器最大占用量保持不变，BLOWFISH算法在计算资源不同的条件下最大寄存器占用量出现了一定程度的改变，这主要是由于BLOWFISH算法中存在8位输入数据扩展为32位输出数据的“混淆”变换，在不同并行度下出现了变量的生命周期与数据位宽的变化所致。除AES-128算法外，其他分组密码算法计算资源变化率随着计算资源数量的减少逐渐增大，说明当功能单元数量减少时，分组密码算法在执行时的并行度逐渐降低，导致程序在执行时对配置生成的功能单元的利用率在降低。AES-128算法由于结构和位宽比较“整齐”，各种类型算子在整个程序的各个控制步中分布较为统一，所以保持了较为稳定的计算资源变化率。

4.2 自动映射算法执行效果对比

为了更准确地测试和对比本文所设计的自动映射算法的执行效果，在65 nm工艺流片的可重构分组密码指令集处理器中进行实验。AES-128算法在各类密码处理器研究文献中被较为广泛进行测试，故将AES-128算法的映射结果对比，为了尽量避免对比过程不同处理器工艺和结构的差异，采用AES-128映射后处理器的吞吐率与时钟频率的比值作为参考指标，衡量单位时钟频率所产生的性能，实验结果如表2和图4所示。通过实验可以观察到，AES-128算法经过本文算法自动映射后，单位时钟频率所产生的性能要优于所对比文献中的AES-128算法执行效果，最大优势比例为258.82%，最小优势比例为6.75%，体现自动映射算法的先进性。

表 2 AES-128算法映射后执行效果对比

对比处理器	时钟频率 (MHz)	η (Mbps/MHz)	本文相比其他算法的优势比例(%)
CryptoManic[1]	360	1.29	231.01
RCBCP[2]	180	2.20	94.09
Crptonite[3]	360	1.62	163.58
PVHArray[15]	110	4.00	6.75
MCPA[16]	1210	1.19	258.82
SRPC[17]	100	3.04	40.46
本文	250	4.27	/

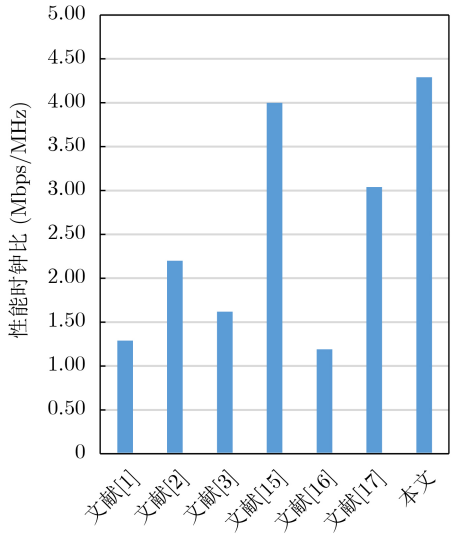


图 4 AES-128算法映射后执行效果对比

5 结束语

为了解分组密码在多发射型可重构指令集处理器中的自动映射问题，本文建立了分组密码算子调度参数模型、多发射型可重构指令集处理器的资源参数模型，研究了映射过程中处理器的资源消耗与资源约束关系，在此基础上利用贪婪思维、列表调度和线性扫描的思想研究了自动映射算法，实现了分组密码在多发射型可重构指令集处理器上的自动并行映射。通过可用资源变化实验验证了自动映射算法在100%, 50%和25%资源条件下的有效性，并横向对比了AES-128的映射效果，实验显示自动映射算法的先进性。本文所做研究丰富了可重构指令集处理器编译研究工作，也可供其他结构可重构处理器的自动映射研究借鉴和改进。

参 考 文 献

[1] WU L, WEAVER C, and AUSTIN T. CryptoManiac: A fast flexible architecture for secure communication[C]. The 28th Annual International Symposium on Computer Architecture, Goteborg, Sweden, 2001: 110–119. doi: 10.1109/ISCA.2001.937439.

[2] 孟涛, 戴紫彬. 分组密码处理器的可重构分簇式架构[J]. 电子与信息学报, 2009, 31(2): 453–456. doi: 10.3724/SP.J.1146.2007.01586.

MENG Tao and DAI Zibin. Reconfigurable clustered architecture of block cipher processor[J]. *Journal of Electronics & Information Technology*, 2009, 31(2): 453–456. doi: 10.3724/SP.J.1146.2007.01586.

[3] BUCHTY R, HEINTZE N, and OLIVA D. Cryptonite–a programmable crypto processor architecture for high-bandwidth applications[C]. The International Conference on Architecture of Computing Systems, Augsburg,

Germany, 2004: 184–198. doi: 10.1007/978-3-540-24714-2_15.

[4] LI Wei, ZENG Xiaoyang, NAN Longmei, *et al*. A reconfigurable block cryptographic processor based on VLIW architecture[J]. *China Communication*, 2016, 13(1): 91–99. doi: 10.1109/CC.2016.7405707.

[5] 严迎建, 王寿成, 徐进辉, 等. 基于Amdahl定律的分组密码并行处理模型研究[J]. 北京理工大学学报, 2018, 38(9): 977–984. doi: 10.15918/j.tbit1001-0645.2018.09.017.
YAN Yingjian, WANG Shoucheng, XU Jinhui, *et al*. Research on parallel processing model for block cipher based on Amdahl's law[J]. *Transactions of Beijing Institute of Technology*, 2018, 38(9): 977–984. doi: 10.15918/j.tbit1001-0645.2018.09.017.

[6] 刘雷波, 王博, 魏少军. 可重构计算密码处理器[M]. 北京: 科学出版社, 2019: 167–172.
LIU Leibo, WANG Bo, and WEI Shaojun. Reconfigurable Cryptographic Processor[M]. Beijing: Scientific Publisher, 2019: 167–172.

[7] 戴紫彬, 曲彤洲. 基于预配置和配置重用的粗粒度动态可重构系统任务调度技术[J]. 电子与信息学报, 2019, 41(6): 1458–1465. doi: 10.11999/JEIT180831.
DAI Zibin and QU Tongzhou. Task scheduling technology for coarse-grained dynamic reconfigurable system based on configuration prefetching and reuse[J]. *Journal of Electronics & Information Technology*, 2019, 41(6): 1458–1465. doi: 10.11999/JEIT180831.

[8] 杜怡然, 南龙梅, 戴紫彬, 等. 可重构分组密码逻辑阵列加权度量模型及高能效映射算法[J]. 电子学报, 2019, 47(1): 82–91. doi: 10.3969/j.issn.0372-2112.2019.01.011.
DU Yiran, NAN Longmei, DAI Zibin, *et al*. Reconfigurable block cryptographic logic array weighted metric model and high energy-efficient mapping algorithm[J]. *Acta Electronica Sinica*, 2019, 47(1): 82–91. doi: 10.3969/j.issn.0372-2112.2019.01.011.

[9] 杜怡然, 杨萱, 戴紫彬, 等. 粗粒度可重构密码逻辑阵列智能映射算法研究[J]. 电子学报, 2020, 48(1): 101–109. doi: 10.3969/j.issn.0372-2112.2020.01.012.
DU Yiran, YANG Xuan, DAI Zibin, *et al*. Research on coarse-grained reconfigurable cryptographic logic array intelligent mapping algorithm[J]. *Acta Electronica Sinica*, 2020, 48(1): 101–109. doi: 10.3969/j.issn.0372-2112.2020.01.012.

[10] LIU Dajiang, YIN Shouyi, LUO Guojie, *et al*. Data-flow graph mapping optimization for CGRA with deep reinforcement learning[J]. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2019, 38(12): 2271–2283. doi: 10.1109/TCAD.2018.2878183.

[11] LI Wei, ZENG Xiaoyang, DAI Zibin, *et al*. A high energy-efficient reconfigurable VLIW symmetric cryptographic processor with loop buffer structure and chain processing mechanism[J]. *Chinese Journal of Electronics*, 2017, 26(6): 1161–1167. doi: 10.1049/cje.2017.06.010.

[12] 王寿成, 严迎建, 徐进辉. 基于流体系结构的高效能分组密码处理器研究[J]. 电子学报, 2017, 45(4): 937–943. doi: 10.3969/j.issn.0372-2112.
WANG Shoucheng, YAN Yingjian, and XU Jinhui. Research of high-efficient block cipher processor based on stream architecture[J]. *Acta Electronica Sinica*, 2017, 45(4): 937–943. doi: 10.3969/j.issn.0372-2112.

[13] 张惠臻, 王超, 李曦, 等. 可重构指令集处理器的代码优化生成算法研究[J]. 计算机研究与发展, 2012, 49(9): 2018–2026.
ZHANG Huizhen, WANG Chao, LI Xi, *et al*. An optimized code-generating algorithm for reconfigurable instruction set processors[J]. *Journal of Computer Research and Development*, 2012, 49(9): 2018–2026.

[14] KWOK Y K and AHMAD I. Static scheduling algorithms for allocating directed task graphs to multiprocessors[J]. *ACM Computing Surveys*, 1999, 31(4): 406–471. doi: 10.1145/344588.344618.

[15] DU Yiran, LI Wei, DAI Zibin, *et al*. PVHArray: An energy-efficient reconfigurable cryptographic logic array with intelligent mapping[J]. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2020, 28(5): 1302–1315. doi: 10.1109/tvlsi.2020.2972392.

[16] LIU Bin 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.

[17] SHAN Weiwei, FU Xingyuan, and XU Zhipeng. 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. doi: 10.1109/TCAD.2015.2419621.

李　盛：男，1984年生，副教授，博士，研究方向为可重构计算与安全专用新芯片编译技术.

戴紫彬：男，1966年生，教授，博士，研究方向为可重构计算与安全专用新芯片设计.

责任编辑：马秀强