计算机科学
COMPUTER SCIENCE

向量 DSP 的混合资源启发式循环展开因子选择方法研究

陆浩松，胡勇华，王书盈，周新莲，李慧祥

引用本文

陆浩松，胡勇华，王书盈，周新莲，李慧祥. 向量 DSP 的混合资源启发式循环展开因子选择方法研究[J]. 计算机科学，2022, 49(6A): 777-783.

LU Hao-song, HU Yong-hua, WANG Shu-ying, ZHOU Xin-lian, LI Hui-xiang. Study on Hybrid Resource Heuristic Loop Unrolling Factor Selection Method Based on Vector DSP[J]. Computer Science, 2022, 49(6A): 777-783.

相似文章推荐（请使用火狐或 IE 浏览器查看文章）

Similar articles recommended (Please use Firefox or IE to view the article)

基于弱约束指派的 DSP 寄存器偶对分配算法研究

Research on DSP Register Pairs Allocation Algorithm with Weak Assigning Constraints

计算机科学, 2021, 48(6A): 587-595. https://doi.org/10.11896/jsjkx.200600061

基于数据重用分析的多面体循环合并策略

Loop Fusion Strategy Based on Data Reuse Analysis in Polyhedral Compilation

计算机科学, 2021, 48(12): 49-58. https://doi.org/10.11896/jsjkx.210200071

基于神经网络的循环分块大小预测

Prediction of Loop Tiling Size Based on Neural Network

计算机科学, 2020, 47(8): 62-70. https://doi.org/10.11896/jsjkx.191200180

循环展开技术在向量程序中的应用

Loop Unrolling in Vectorized Programs

计算机科学, 2016, 43(1): 226-231. https://doi.org/10.11896/j.issn.1002-137X.2016.01.049

一种面向 SIMD 扩展部件的向量化统一架构

Unified Vectorization Framework for SIMD Extensions

计算机科学, 2014, 41(9): 28-31. https://doi.org/10.11896/j.issn.1002-137X.2014.09.004

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.210400146

向量 DSP 的混合资源启发式循环展开因子选择方法研究

陆浩松　胡勇华　王书盈　周新莲　李慧祥
湖南科技大学计算机科学与工程学院　湖南 湘潭 411201
(webdfbxg@163.com)

摘　要　在现代处理器中,具有向量处理单元的 VLIW 体系结构逐渐成为高性能 DSP 体系结构的典型代表。基于这类体系结构的寄存器资源丰富、执行单元多等特点,研究了相应的循环展开因子选择问题,提出了一种循环展开因子选择方法来提升循环展开这种重要优化的效果。该方法考虑了循环体代码的向量标量属性、基址寄存器和索引寄存器资源使用规则等因素,并且在展开因子选择算法中增加了执行单元使用占比和展开因子按幂次对齐这两种启发式因素。针对 3 种常用数字信息处理算法开展了实验研究,实验结果表明了该方法的有效性。对于这三种 DSP 算法,用所提方法获得的循环展开因子进行循环展开处理后,它们的平均性能相比已有方法提升了 10%以上。

关键词:循环展开;展开因子;超长指令字;向量 DSP;编译优化
中图法分类号　TP314

Study on Hybrid Resource Heuristic Loop Unrolling Factor Selection Method Based on Vector DSP

LU Hao-song, HU Yong-hua, WANG Shu-ying, ZHOU Xin-lian and LI Hui-xiang
School of Computer Science and Engineering, Hunan University of Science and Technology, Xiangtan, Hunan 411201, China

Abstract　For modern microprocessors, the very long instruction word (VLIW) architecture integrating vector processing units has gradually become a typical representative of high-performance digital signal processor (DSP) architectures. This architecture is mainly characterized by rich register resources and many instruction execution units. Based on these characteristics, a selection method for the corresponding loop unrolling factor is proposed to improve the effect of loop unrolling optimization. This method takes into account the vector or scalar attribute of the code in a loop body, and the usage rules of base address registers and index registers. Moreover, another two heuristics, i. e. , the proportion of the times that the execution units are used and the power alignment of unrolling factor, are used in the loop unrolling factor selection algorithm. The ability of this method in developing more instruction level parallelism is proved by experiments performed on three commonly used digital signal processing algorithms. Experiment results show that the average performance of the algorithms improves by more than 10% compared with the existing methods. In particular, experiments on FFT algorithm show that the proposed method can analyze the usage of related hardware resources more accurately through the hybrid resource heuristics, and makes the judgment of unrolling and obtains the corresponding value of loop unrolling factor.

Keywords　Loop unrolling, Unrolling factor, VLIW, Vector DSP, Compiler optimization

1　引言

DSP 产业经历了从理论基础到广泛应用,再到现今的多元化发展,DSP 处理器的硬件性能有了多个改进方向,如 Vector-SIMD 结构[1-2]、多核技术[3-4]等。目前许多公司和机构开始结合 Vector-SIMD 结构和多核技术研发新型高性能 DSP,如密歇根大学的 AnySP 处理器[5]、Tensilica 公司的 BBE-128 处理器[6]、国防科大研发的 YHFT 处理器[7]等。Vector-SIMD 结构一般包含标量单元(Scalar Unit,SU)和 SIMD 单元(SIMD Unit,SIMDU),其中 SU 主要负责流控、标量计算等,SIMDU 包含多个并行的 Lane,在运算加速中起主导作用[7]。此外,针对多核处理器的核间通信技术也得到了发展[8],DSP 运算能力不断提高。不过,尽管高性能 DSP 的硬件资源更加丰富,但其硬件性能的发挥还取决于编译器对其代码的优化能力。因此,随着高性能 DSP 的发展,相应的编译优化理论与技术变得越来越重要。

一般 DSP 算法程序的运行时间主要花费在循环上。施加在循环上的代码优化手段或方法主要有循环嵌套优化[9-10]、循环变换[11]、循环展开[12]等。在这些方法中,循环展开是一种循环变换技术[12],其基本方法是将循环体展开多次,消除循环转移指令,从而可以减少循环次数,较大程度地扩大指令调试域。对于 VLIW 体系结构来说,循环展开可以有效地提升代码的指令级并行度,因此是一种非常重要的优化。循环展开的核心问题在于如何确定循环展开因子(即

基金项目:湖南省教育厅科研项目(20B242,19A169);湖南省自然科学基金(2017JJ3087);国家自然科学基金(61872138)
This work was supported by the Research Projects of Hunan Provincial Department of Education(20B242,19A169), Natural Science Foundation of Hunan Province, China(2017JJ3087) and National Natural Science Foundation of China(61872138).
通信作者:胡勇华(yonghuahu@hnust.edu.cn)

展开后的循环体中原循环体代码的复本数量)。迄今,确定循环展开因子的方法主要是通过构建代价模型的循环展开因子计算方法[13-16]。例如:针对多重完美嵌套循环,Sarkar 通过代价函数评估循环展开后带来的性能提升来确定展开因子[13];Carr 等提出一种低代价(Low Cost)方法,考虑标量替换、常规优化、软件流水和寄存器分配共同作用的影响,先预测寄存器压力,再利用预测结果来确定循环展开因子[14];Li 等提出了基于程序特性的展开因子算法,通过评估程序的资源限制和延迟限制下循环展开的性能来获得循环展开因子[16]。

需要指出的是,这类方法虽然有效,但目前的研究仅限于对标量资源的使用,没有综合考虑各种硬件资源的影响。因此,在向量 DSP 情况下还不能用它们获得合适的循环展开因子。本文针对典型的向量 VLIW 体系结构提出了一种求 DSP 算法程序的循环展开因子的通用方法。本文第 2 节对本文方法所面临的问题进行了讨论分析;第 3 节给出了本文方法的处理模型及其循环展开因子求解方法;第 4 节给出了具体的算法,并结合实例展示了算法求解过程;第 5 节将本文方法施加到 3 种 DSP 算法上,研究了本文方法的有效性并进行了分析;最后总结全文。

2 问题及分析

DSP 算法中的循环部分代码性能的主要衡量标准是程序代码的运行时间。如果循环展开因子过小,结果可能会导致循环展开得不充分,有些硬件资源仍被闲置;相反,循环展开因子过大,结果可能会导致硬件资源过度使用(如导致寄存器数量不够,进而导致额外的符号寄存器溢出处理),展开后的循环的性能可能反而下降。因此,一个恰当的循环展开因子对适合进行循环展开优化的 DSP 算法来说至关重要。

根据 VLIW 体系结构的特点,考虑循环代码对硬件资源的需求可以得知,影响其循环展开的考虑因素主要有:1)各类寄存器的实际数量与循环对这些寄存器的需求量;2)执行单元的需求情况。

循环展开优化中对硬件资源的考虑主要集中在寄存器方面。在向量 DSP 体系结构下,寄存器资源包括标量寄存器、向量寄存器等多种类别的异构寄存器。因此,需要根据上述影响因素 1 建立专门的模型。对于因素 2,当循环中一个执行单元占比过高,可能导致指令调度时没有足够多的执行单元来并行使用,这时就需要插入空节拍,等待执行单元使用结束才能给它派发一条新的指令,降低了代码的密度。当较大的循环体的展开次数较多时,过长的基本块可能导致代码 Cache 取指失效的概率增大,从而影响循环的总体性能。

在传统方法中,循环展开因子选择算法主要是将标量资源作为确定循环展开因子的依据,并且主要考虑的是标量寄存器。然而,在向量 DSP 体系结构中还设有用于 SIMD 处理的向量资源,该体系结构主要靠向量资源来提升处理器的计算能力。因此,向量资源也应像标量资源一样成为循环展开因子选择算法的依据。此外,作为高性能结构,向量 DSP 往往有多个专用寄存器组来加速,如用于存储基地址或偏移量的寄存器组等。对它们的使用合理与否也会影响循环展开的效果,但现有文献中没有将它们纳入考虑范围。

对于向量 DSP 结构,在已有算法的基础上,本文提出的循环展开因子选择算法将进一步考虑以下几个影响因素。

(1)循环体代码的标量/向量属性(以下简称 SV 属性)。之所以考虑这个因素,是因为标量资源和向量资源数量不一定对等,向量资源中的寄存器和执行单元均可能超过相应的标量资源。

(2)基址寄存器和索引寄存器资源数量。在已有的算法中,主要考虑通用寄存器对循环展开因子的限制。但需要指出的是,在高性能 DSP 体系结构中,为了提升访存处理的性能,寄存器包括通用寄存器、基址寄存器和索引寄存器三大类。任何一类寄存器被过度使用都会影响代码的性能。

(3)执行单元使用占比。当循环中一个执行单元占比过高,可能导致指令调度时没有足够多的执行单元来并行使用,这时就需要插入空节拍,等待执行单元使用结束,才能给它派发一条新的指令,降低了代码的密度。当较大的循环体的展开次数较多时,过长的基本块可能导致代码 Cache 取指失效的概率增大,从而影响循环的总体性能。

3 处理模型

3.1 基本定义

为了建立本文算法所需的处理模型,对向量 DSP 处理器核心的基本资源定义如下。

定义 1(通用寄存器) 用 CR 表示,且可细分为标量寄存器和向量寄存器。

定义 2(基址寄存器) 其中保存的地址用来作为访存指令的基地址,用 AR 表示,可以分为用于标量访存的标量基址寄存器和向量访存的向量基址寄存器。

定义 3(索引寄存器) 指与 AR 匹配使用,用来作为地址的位置索引的寄存器,用 OR 表示,可以相应地分为标量索引寄存器和向量索引寄存器。

定义 4(执行单元) 指能够执行某条指令的功能单元,用 FN 表示。

定义 5(不展开指令) 这类指令在展开算法中不用创建复本,如跳转指令。

与循环中的代码有关的几个概念为:1)用 IS 表示;2)循环变量,指在循环中其值可能发生变化的变量,用 VV 表示;3)循环不变量,指在循环中其值不被改变的变量,用 CV 表示。

3.2 模型

本文提出的向量 DSP 结构循环展开因子选择模型中,其最终的循环展开因子 UF_{final} 可以通过式(1)来计算:

$$UF_{\text{Final}} = 2^{\text{floor}(\log_2(UF_{\text{Best}}))} \tag{1}$$

其中,UF_{Best} 代表多类寄存器限制下的最优循环展开因子。式(1)表示将各类寄存器限制下的最优循环展开因子按幂次对齐。若用 R_x 代表某类寄存器,则 UF_{Best} 的表达式为:

$$UF_{\text{Best}} = \min(UF_{Rx}), Rx \in \{CR, AR, OR\} \tag{2}$$

其中,UF_{Rx} 是一个对应寄存器资源的量,其表达式为:

$$UF_{Rx} = floor((VS_{Rx} - CVN_{Rx})/VVN_{Rx}) \tag{3}$$

其中,VS_{Rx} 为 Rx 类寄存器的总数量,CVN_{Rx} 为 Rx 类寄存器的循环不变量的数量,VVN_{Rx} 为 Rx 类寄存器的循环变量的数量。式(2)的作用是在各类寄存器限制下的循环展开因子中取最小值,获得一个多类寄存器限制下的最优循环展开因子。式(3)表示通过使用寄存器资源对应的循环变量和循环不变量的个数以及能够使用的寄存器资源来计算某一类寄存

器资源限制下的循环展开因子。

在这个模型中，与已有算法相比，本文算法考虑了 VLIW 体系结构中存在的向量资源情况以及不同指令对循环展开后效果的影响。

3.3 模型求解

图 1 给出了本文算法的数据流模型。循环展开因子 UF_{final} 的具体计算过程如下。

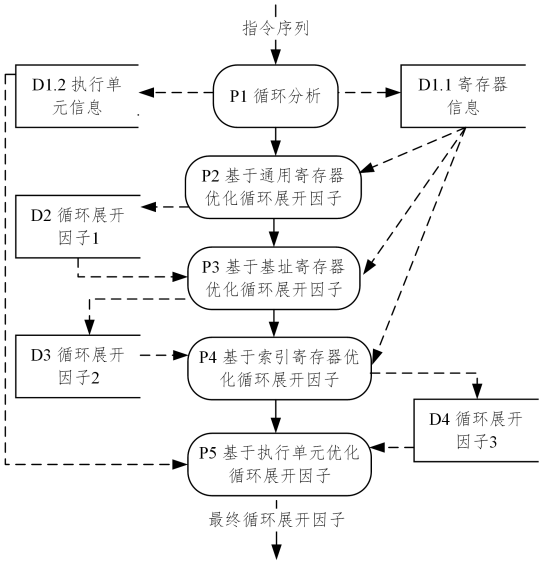

图 1 向量 DSP 结构下循环展开因子选择算法数据流模型

Fig. 1 Data flow model of loop expansion factor selection algorithm based on vector DSP

Step 1 分析循环中的变量寄存器类别、指令数量、特殊指令数量等信息，同时判断循环体内各指令的 SV 属性，获取所需的硬件资源使用情况信息，将得到的数据存储于 D1.1 和 D1.2。这一步对应于图 1 的子过程 P1。

Step 2 根据 Step 1 获得的信息，分析通用寄存器资源对应的循环变量和循环不变量的个数，用式(4)计算基于某类通用寄存器的循环展开因子作为当前循环展开因子：

$$UF_{Best}=floor((CRVS-CRCVN)/CRVVN) \tag{4}$$

其中，$CRVS$ 表示能够使用的通用寄存器的数量，$CRCVN$ 表示对应通用寄存器的循环不变量的个数，$CRVVN$ 表示对应通用寄存器的循环变量的个数。$floor(X)$ 函数表示对 X 向下取整。UF_{Best} 表示此时得到的循环展开因子。这一步对应于图 1 中的子过程 P2，将获得的数据存储于 D2。

Step 3 根据 Step 1 获得的信息，分析基址寄存器资源对应的循环变量和循环不变量的个数，用式(5)计算基址寄存器资源限制下的循环展开因子。

$$UF_{AR}=floor((ARVS-ARCVN)/ARVVN) \tag{5}$$

其中，$ARVS$ 表示能够使用基址寄存器的数量，$ARCVN$ 表示对应基址寄存器的循环不变量的个数，$ARVVN$ 表示对应基址寄存器的循环变量的个数。

Step 4 用式(5)比较基址寄存器资源限制下的循环展开因子与当前循环展开因子，用两者中的较小值更新当前循环展开因子。

$$UF_{Best}=\min(UF_{Best},UF_{AR}) \tag{6}$$

其中，上述 Step 3 和 Step 4 对应于图 1 中的数据处理过程 P3，将获得的数据存储于 D3。

Step 5 根据 Step 1 获得的信息，分析索引寄存器资源对应的循环变量的数量和循环不变量的数量，计算索引寄存器资源限制下的循环展开因子 UF_{OR}：

$$UF_{OR}=floor((ORVS-ORCVN)/ORVVN) \tag{7}$$

其中，$ORVS$ 表示能够使用的索引寄存器的数量，$ORCVN$ 表示对应索引寄存器的循环不变量的个数，$ORVVN$ 表示对应索引寄存器的循环变量的个数。

Step 6 比较 UF_{OR} 与当前循环展开因子，用两者中的较小值更新当前循环展开因子。

$$UF_{Best}=\min(UF_{Best},UF_{OR}) \tag{8}$$

其中，上述 Step 5 和 Step 6 对应于图 1 中的子过程 P4，将获得的数据存储于 D4。

Step 7 将 UF_{Best} 按幂次对齐，计算由各类寄存器决定的展开因子 UF_{Final} 作为当前循环展开因子。

$$UF_{Final}=2^{floor(\log_2(UF_{Best}))} \tag{9}$$

Step 8 根据 Step 1 中分析的数据获得的信息计算各执行单元使用占比。

$$FR_X=\frac{FI_X*UF_{Final}}{(IS-SIN)*UF_{Final}+SIN},X=\{1,2,3,\cdots,N\} \tag{10}$$

其中，X 代表某个执行单元，FR_X 指第 X 个执行单元的占比，FI_X 为第 X 个执行单元的使用次数，IS 为指令总数，SIN 为不展开指令的数量。这一步对应于图 1 中的子过程 P5。

Step 9 比较各指令单元使用占比是否超过上限值 FR。如果各执行单元占比都未超过上限值，那么 UF_{Final} 即为最终循环展开因子，处理过程结束。如果某个执行单元占比超过上限值，那么令：

$$UF_{Best}=UF_{Final}-1 \tag{11}$$

然后转 Step 7 继续处理。

4 算法及实例分析

4.1 算法

上述求解过程如算法 1 所示。

算法 1 向量 DSP 的混合资源启发式循环展开因子选择算法

```
Input:循环代码
output:循环展开因子
begin
1. LoopAanalysis()
2. JudgeCodeType();
3. BestFactor=CalcCRRegisterUnrollFactor();
4. MaxFactor=CalcARRegisterUnrollFactor();
5. BestFactor=min(BestFactor,MaxFactor);
6. MaxFactor=CalcORRegisterUnrollFactor();
7. BestFactor=min(BestFactor,MaxFactor);
8. while 1
9. X=1
10. while X*2<BestFactor
11. X=X*2;
12. end while
13. BestFactor=X;
14. if FactorMeetFun(BestFactor) || BestFactor==1
15. break
16. end if
17. BestFactor-=1;
18. end while
end
```

此算法主要包含以下几个部分：

(1)LoopAanalysis()，分析循环，获得循环展开因子选择算法中所需的各类数据。主要包括各类寄存器的需求情况，指令总数、不展开指令数量、各执行单元使用次数等。这个函数通过分析程序代码的每一行来获得当前行的指令类别、需要使用的寄存器类别(区分用于循环不变量和循环变量)。在分析中对每一行的数据进行统计。

(2)JudgeCodeType()，通过循环体内的指令类型来判定判断代码类型为向量处理代码还是标量处理代码。

(3)CalcCRRegisterUnrollFactor()，计算通用寄存器限制下的循环展开因子，计算式见式(4)。把所获得的循环展开因子存储在整型变量 BestFactor 中，后续将用来与其他循环展开因子进行比较。

(4)CalcARRegisterUnrollFactor()，计算基址寄存器限制下的循环展开因子，计算式见式(5)。把所获得的循环展开因子存储在整型变量 MaxFactor 中，后续将用来与其他循环展开因子进行比较。

(5)CalcORRegisterUnrollFactor()，计算索引寄存器限制下的循环展开因子，计算公式见式(7)。把所获得的循环展开因子存储在整型变量 MaxFactor 中，后续将用来与其他循环展开因子进行比较。

(6)FactorMeetFun()，用于判断当前循环展开因子的限制下，是否会存在某一个执行单元使用占比达到上限的情况，这个函数的返回值为 True 或者 False，用于标识该条件是否成立。

(7)代码第 11－21 行，对应于模型求解方法中的 Step 7－Step 9。将当前循环展开因子 BestFactor 按幂次对齐，通过式(10)计算当前循环展开因子 BestFactor 下每个执行单元使用占比是否超过上限值。如果未超过(或当前循环展开因子为 1)，则结束循环并返回最终循环展开因子。

4.2 实例分析

本节通过复数求模算法实例来阐述本文模型选择循环展开因子的具体步骤。我们设原始数据的实部和虚部分分别存放在 ARvRe 和 ARvIm 对应的地址空间中。在结构方面，我们假设处理器有专用的求平方根的指令 V_float_Sq，并假设通用寄存器资源个数为 NR，基址寄存器资源个数为 NAR，索引寄存器资源个数为 NOR，执行单元使用占比上限为 P。该算法的计算循环片断如算法 2 所示。

算法 2

```
1. A_Vector_module_unroll_f1s:
2.    LT Riloop,RloopTimes,Rcond;
3.    [! Rcond]BR A_Vector_module_unroll_f1e;
4.    S_int_mul ROneC,Riloop,RH:RL;
5.    S_RegPair_ARF RH:RL,ORVamArrayIndex_1;
6.    v_ld *+ARvRe[ORVamArrayIndex_1],VRre1;
7.    v_ld *+ARvIm[ORVamArrayIndex_1],VRim1;
8.    V_float_Mul VRre1,VRre1,VRre2;
9.    V_float_Mul VRim1,VRim1,VRim2;
10.   V_float_add VRre2,VRim2,VRSqSum;
11.   V_mov VRSqSum,VRsg1;
12.   V_mov VRSqSum,VRsx1;
13.   V_float_Sq VRsg1,VRL;
14.   V_float_Mul VRsx1,VRL,VRL1;
15.   V_float_add VRsg1,VRL1,VRsg1;
16.   V_float_Mul VRsg1,VRhalf,VRsg1;
17.…
18.   v_stw VRsg1,*+ARvRe[ORVamArrayIndex_1];
19.   S_int_add 1,Riloop,Riloop;
20.   sbr A_Vector_module_unroll_f1s;
21. A_Vector_module_unroll_f1e.
```

由于在部分体系结构中求浮点开方运算需要进行多次迭代才能达到所需的精度，上述代码中第 17 行的…代表重复第 13－16 行代码 N 次(N 由体系结构决定)。

分析这个循环中的向量指令可以得知这是一个向量代码程序，因此需要获得总体向量资源的情况。对于这个实例的具体处理过程以及其中涉及的数据变化分析如下。

(1)使用通用寄存器的循环不变量(所有寄存器中未出现在目的操作数上的寄存器作为循环不变量)为 VRhalf，因此这个实例中循环不变量个数为 1，$CRCV$ 的值为 1；使用通用寄存器的循环变量个数为 10，因此 $CRVN$ 的值为 10。根据式(4)计算通用寄存器资源限制下的循环展开因子为：

$$UF_{\text{Best}}=floor((NR-1)/10)$$

(2)使用基址寄存器的循环不变量为 2 个，分别是 ARvRe，ARvIm，因此 $ARCV$ 的值为 2；使用基址寄存器的循环变量个数为 0，所以 $ARVN$ 的值为 0。因此这个循环中基址寄存器不会限制循环展开因子。

(3)使用索引寄存器的循环不变量为 0 个，因此 $ORCV$ 的值为 0；使用索引寄存器的循环变量个数为 1，因此 $ORVN$ 的值为 1。根据式(7)计算索引寄存器资源限制下的循环展开因子为：

$$UF_{\text{OR}}=floor((NOR-0)/1)$$

(4)根据式(8)将 UF_{OR} 与当前循环展开因子进行比较，取最小值为：

$$UF_{\text{Best}}=\min(UF_{\text{Best}},UF_{\text{OR}})$$

(5)将当前循环展开因子按幂次对齐后的最终展开因子为 UF_{Final}，并分析最终展开因子 UF_{Final} 是否会导致某个执行单元使用占比超过使用占比上限值 P。如果不会，其即为最终循环展开因子。

5 实验

5.1 实验设计

本节通过比较使用不同循环展开因子选择算法对程序性能的影响来反应循环展开因子选择算法的效果。程序性能由指令节拍数来衡量，节拍数越少，代表程序性能越优。

本节中所有实验是基于国防科大自主研发的 YHFT 系列芯片搭建的 FT-M7002DSK 板卡。此外，实验中设定执行单元部件占比为 75%。

本文实验对算法 3 和算法 4 进行了对比。

算法 3 文献[16]中提出的循环展开因子选择算法，其主要处理模型如下：

首先通过编译选项(由代码编写者预估)来确定循环展开的下限值 min_uf 和上限值 max_uf。

在上限值与下限值之间综合考虑每个展开因子 $k(min\_uf\leqslant k\leqslant max\_uf)$ 下的寄存器资源利用率：

$$P=\frac{k*\text{局部变量数}+\text{循环不变量数}}{\text{寄存器总数}}$$

然后选择资源利用率 P 最大的循环展开因子作为最终的循环展开因子(注：最终循环展开因子的资源利用率不宜超过 1)。最后判定是否含有循环分支指令，如果含有，那么最终循环展开因子设置为 2(注：文中直接设定)。

算法 4 文献[17]中提出的循环展开因子选择算法，其主要处理模型如下：

首先获得平台寄存器数目，分析循环中的循环不变量和循环变量的个数后再通过下列公式计算：

$$UF=\frac{PowerOf2Floor(TNR\text{-}LIR)}{LUR}$$

其中，TNR 为平台寄存器的数量，LIR 为循环不变量的数目，LUR 为局部变量的数目。$PowerOf2Floor(n)$ 表示不大于 n 的最大 2 的整数次幂。

本文的每个对比实验都将通过比较代码的指令节拍数的方式来分析循环展开因子选择算法在当前程序代码下的性能。由于 FT-M7002DSK 板卡上的 DSP 处理器带有定时器，在测试程序中对定时器初始化之后就可以读取它的计时值。因此，我们在测试程序中先对定时器进行初始化，然后在调用算法函数前后分别读取该定时器的值，两个值相减就得到了算法函数所需的执行时间 Tr。之后，Tr 再乘以一个与系统的时钟频率有关的系数，就得到了对应的节拍数。

5.2 实验结果与分析

下面给出 3 个 DSP 算法的对比实验来验证本文提出的循环展开因子选择方法。

对比实验 1 中使用的实例是 4.2 节中提到的复数求模算法。分析后所获得各信息的数据如表 1 所列。

表 1 复数求模算法程序中的各信息数据表

Table 1 Information data in the program of complex modulus algorithm

信息名称	数值
使用通用寄存器的循环不变量个数	1
使用通用寄存器的循环变量个数	10
使用基址寄存器的循环不变量个数	2
使用基址寄存器的循环变量个数	0
使用索引寄存器的循环不变量个数	0
使用索引寄存器的循环变量个数	1

结合模型可以获得通用寄存器限制下的循环展开因子为 6，索引寄存器限制下的循环展开因子为 8，基址寄存器不限制循环展开因子。最终循环展开因子取各类寄存器限制下的循环展开因子最小值为 6，将最终循环展开因子按幂次对齐，获得最终循环展开因子为 4。通过对执行单元占比的计算分析可知，最终循环展开因子不会导致执行单元占比超过上限值，所以本文提出的 DSP 体系结构代码下循环展开因子选择算法(后续用本文算法代替)选择的循环展开因子为 4。

在进行对比算法 1 计算时，我们先预设了下限展开因子为 1，上限展开因子为 8。分别计算每个循环展开因子对应的资源使用率后获得的资源使用率不超过 1 的最大值为 0.90625(循环展开因子为 5 时达到)。因此，这个实验中算法 1 的循环展开因子为 5。

在进行对比算法 2 计算时，我们将各类数据代入算法模型中计算获得：

$$UF=\frac{PowerOf2Floor(64-1)}{10}$$

因此，这个实验中算法 2 的循环展开因子为 3。

具体实验结果如图 2 所示。

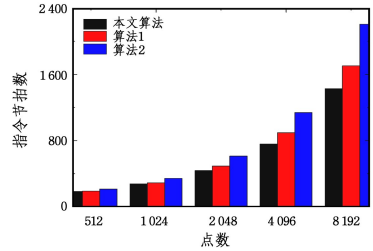

图 2 循环展开因子在复数求模的程序代码中不同点数的节拍数对比

Fig. 2 Code beat number comparison of different points of cycle expansion factor incomplex modulus program

图 2 中纵坐标表示运行完对应数据点数所需要的节拍数，横坐标表示数据点数。实验结果表明，相比其他两种算法，在不同点数下本文算法选择的循环展开因子下的循环运行节拍数最少，性能最优。与已有的循环展开因子选择算法相比，本文算法在这个实例上的平均性能提升了 10%。

在选择实验对比的标准上，我们选择节拍数作为对比标准，而不是选择静态指令数来对比。这是因为对于一个循环次数为 10 的循环来说，在展开 3 次和展开 2 次两种情况下，展开后的循环体内静态指令数前者会大于后者，但前者的循环次数需要 3 次而后者循环次数需要 5 次。所以循环展开因子选择算法的优劣不能直接通过静态代码的指令数来比较，只有通过动态运行程序来比较程序运行所需的节拍数才能反应出循环展开因子对程序性能提升的优劣。

对比实验 2 中使用快速傅里叶变换(FFT)基 2 计算中的一个内层循环，下面给出分析后获得该实例中的各信息数据，如表 2 所列。

表 2 FFT 基 2 程序中的各信息数据

Table 2 Information data of FFT Radix-2 program

信息名称	数值
使用通用寄存器的循环不变量个数	1
使用通用寄存器的循环变量个数	28
使用基址寄存器的循环不变量个数	4
使用基址寄存器的循环变量个数	0
使用索引寄存器的循环不变量个数	0
使用索引寄存器的循环变量个数	4

结合模型可以获得通用寄存器限制下的循环展开因子为 2，索引寄存器限制下的循环展开因子为 2，基址寄存器不限制循环展开因子。最终循环展开因子取上述限制下的循环展开因子最小值为 2，将最终循环展开因子按幂次对齐获得最终循环展开因子为 2，所以本文算法选择的循环展开因子为 2。

在进行对比算法 1 计算时，我们先预设了下限展开因子为 1，上限展开因子为 3。通过分别计算每个循环展开因子对应的资源使用率后获得的资源使用率不超过 1 的最大值为 0.578125(循环展开因子为 1 时达到)。其他资源使用率超过资源上限值 1。因此，这个实验中算法 1 的循环展开因子为 1。

在对比算法 2 计算时，我们将各类数据代入算法模型中计算获得：

$$UF=\frac{PowerOf2Floor(64-1)}{28}$$

因此，这个实验中算法 2 的循环展开因子为 1。

具体实验结果如图 3 所示。

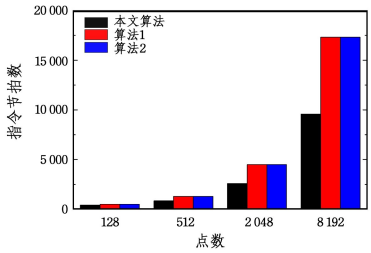

图 3 循环展开因子在 FFT 基 2 的程序代码中不同点数的节拍数对比

Fig. 3 Code beat number comparison of different points of cycle expansion factor in FFT based-2 program

从实验结果可以看出，相比其他两种算法，在不同点数下本文算法选择的循环展开因子展开后的循环运行节拍数最少，性能最优。

在这个 FFT 基 2 的程序代码中，当不展开时 OR 寄存器使用了 4 个。本文算法考虑了 OR 寄存器的限制使用了 8 个，且通用寄存器数量不能足以支撑展开 3 次，所以选择展开 2 次。而算法 1 和算法 2 通过对循环的分析计算后获得循环展开因子均为 1，对该循环不进行展开。通过对比可以看出，该循环适合进行展开且展开后的性能提升较为明显。因此我们使用混合资源的启发式方法对循环展开因子的选择更为准确。与已有的循环展开因子选择算法相比，本文算法在这个实例上的平均性能提升了 34%。

对比实验 3 中使用的实例为向量求负数函数中的一个计算循环，下面给出分析循环后获得该实例中的各信息数据，如表 3 所列。

表 3 向量求负数程序中的各信息数据表

Table 3 Information data in vector negative program

信息名称	数值
使用通用寄存器的循环不变量个数	1
使用通用寄存器的循环变量个数	2
使用基址寄存器的循环不变量个数	1
使用基址寄存器的循环变量个数	0
使用索引寄存器的循环不变量个数	0
使用索引寄存器的循环变量个数	1

结合模型可以获得通用寄存器限制下的循环展开因子为 31，索引寄存器限制下的循环展开因子为 8，基址寄存器不限制循环展开因子。最终循环展开因子取上述限制下的循环展开因子的最小值 8，将最终循环展开因子按幂次对齐获得最终循环展开因子为 8，因此本文算法选择的循环展开因子为 8。

在进行对比算法 1 计算时，我们先预设了下限展开因子为 1，上限展开因子为 16。分别计算每个循环展开因子对应的资源使用率后获得的资源使用率不超过 1 的最大值为 0.9218(循环展开因子为 7 时达到)。因此，这个实验中算法 1 的循环展开因子为 7。

在进行对比算法 2 计算时，我们将各类数据代入算法模型中计算获得：

$$UF=\frac{PowerOf2Floor(64-1)}{2}$$

因此，这个实验中算法 2 的循环展开因子为 16。具体实验结果如图 4 所示。

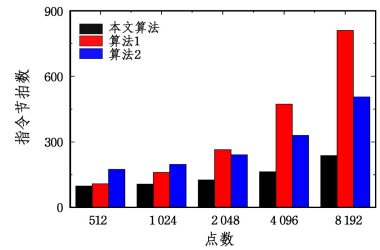

图 4 循环展开因子在向量求负数的程序代码中不同点数的节拍数对比

Fig. 4 Code beat number comparison of different points of cycle expansion factor invector negative program

从实验结果可以看出，相比其他两种算法，在不同点数下本文算法选择的循环展开因子节拍最少，性能最优。与已有的循环展开因子选择算法相比，本文算法在这个实例上的平均性能提升了 46%。

在这个向量求负数的程序代码中，算法 1 未考虑向量资源和区分寄存器的类别，因此根据算法 1 计算获得的循环展开因子为 7。算法 2 只考虑通用寄存器的限制因素，因此根据算法 2 计算获得的循环展开因子为 16。本文算法在考虑索引寄存器的限制条件下计算获得循环展开因子为 8。由实验结果可以看出，由于算法 2 存在过度使用索引寄存器的情况，导致了在展开因子选择过大时程序性能下降。因此本文将索引寄存器作为限制因素之一是合理的。通过以上 3 个对比实验可以看出，在考虑索引寄存器资源限制以及按幂次对齐的情况下对算法性能有较大提升。

结束语 本文根据向量 DSP 的 VLIW 体系结构寄存器资源丰富、执行单元多等特点研究了循环展开优化问题，提出了一种循环展开因子选择方法来提升循环展开的效果。我们通过构建代价模型的方式来计算循环展开因子，并阐述了计算模型及其求解方法。通过实验我们发现，与已有方法相比，本文方法对循环代码的性能提升较明显，硬件使用率也较高。尤其在 FFT 算法上的实验表明，对于已有方法判定为不能展开的循环，本文方法能通过混合资源启发式方法来更加精准地分析相关资源的使用情况，并作出循环展开判断和获得循环展开因子值。对于具备本文所述体系结构所涵盖的各种 DSP 体系结构，本文方法可应用于解决相应代码的循环展开因子选择问题。此外，本文方法也可以用于事先了解各类寄存器和执行单元等处理器核心资源的各种配置数量带来的性能提升效果变化，从而帮助 DSP 设计者做出更好的决策。

参 考 文 献

[1] LEE Y, AVIZIENIS R, BISHARA A, et al. Exploring the tradeoffs between programmability and efficiency in data-parallel accelerators[J]. ACM Sigarch Computer Architecture News, 2011, 39(3): 129-140.

[2] LIN C C, GU N J, LEI Y M, et al. SIMD compiler optimization of clustered VLIW DSP [J]. Journal of University of Science and Technology of China, 2011(8): 53-59.

[3] HE G Q, CHEN Y. Research on Huarui DSP software architecture [J]. Modern Radar, 2016(9): 17-22.

[4] BLAKE G, DRESLINSKI R G, MUDGE T. A survey of multicore processors[J]. IEEE Signal Processing Magazine, 2009, 26(6): 26-37.

[5] WOH M，SEO S，MAHLKE S A，et al. AnySP：Anytime Anywhere Anyway Signal Processing[C]// 36th International Symposium on Computer Architecture（ISCA 2009）. Austin，TX，USA. ACM，2009：20-24.

[6] ROWEN C，DAN N，RAVINDRAN R，et al. The world's fastest DSP core：Breaking the 100 GMAC/s barrier[C]// 2011 IEEE Hot Chips 23 Symposium（HCS）. IEEE，2011.

[7] CHEN S M，LIU S，WAN J H，et al. Architecture and implementation of collaborative multi-core DSP YHFT qmbase [J]. Chinese Science：Information Science，2015，45(4)：560-573.

[8] ZHOU N，WANG R，QIN Y Y，et al. Design and implementation of inter core communication mechanism based on heterogeneous multi-core environment [J]. Computer Engineering and Design，2019，40(3)：294-300，308.

[9] PADUA D A. Advanced compiler optimizations for supercomputers[J]. Communications of the ACM，1986，29(12)：1184-1201.

[10] SHIVAM A，WATKINSON N，NICOLAU A，et al. Towards an Achievable Performance for the Loop Nests[J]. arXiv：1902.00603，2019.

[11] CUI Y Z，LIU S，WANG Q，et al. Study on cyclic optimization technique of lattice Boltzmann method [J]. Acta Computa Sinica，2020，450(6)：116-132.

[12] WEISS S，SMITH J E. A study of scalar compilation techniques for pipelined supercomputers[J]. ACM Sigarch Computer Architecture News，1990，15(5)：105-109.

[13] SARKAR V. Optimized Unrolling of Nested Loops[J]. International Journal of Parallel Programming，2001，29(5)：545-581.

[14] CARR S，GUAN Y. Unroll-and-jam using uniformly generated sets[C]// Proceedings of 30th Annual International Symposium on Microarchitecture. IEEE，1997：349-357.

[15] CARR S M，KENNEDY K W. Improving the ratio of memory operations to floating-point operations in loops[J]. ACM Transactions on Programming Languages & Systems，1994，16(6)：1768-1810.

[16] LI W L，LIU L，TANG Z Z. Optimization of loop unfolding in software pipeline [J]. Journal of Beijing University of Aeronautics and Astronautics，2004，30(11)：1111-1115.

[17] HUANG Y B，LI C J. Implementation of tail loop Vectorization Based on llvm[C]// Proceedings of the 20th Annual Conference of Computer Engineering and Technology and the 6th Microprocessor Technology Forum. Computer Society of China，2016.

LU Hao-song，born in 1995，postgraduate. His main research interests include DSP compilation and code optimization technology.

HU Yong-hua，born in 1981，Ph.D，professor，Ph.D supervisor. His main research interests include DSP compilation and code optimization technology.

（上接第 679 页）

[10] CAI K Y，WANG H. Cloud Classification of Satellite Image Based on Convolutional Neural Networks[C]// IEEE International Conference on Software Engineering and Service Science（ICSESS）. 2017：894-897.

[11] GAO Y L，ZHAN X G，CHI C Y. Research on sentiment analysis based on CNN-LSTM model[J]. Journal of University of Science and Technology Liaoning，2018，41(6)：469-474.

[12] LI T，HUA M，WU X. A Hybrid CNN-LSTM Model for Forecasting Particulate Matter(PM2.5)[J]. IEEE Access，2020(8)：26933-26940.

[13] REHMAN A U，MALIK A K，RAZA B，et al. A Hybrid CNN-LSTM Model for Improving Accuracy of Movie Reviews Sentiment Analysis[J]. Multimedia Tools and Applications，2019，78(18)：26597-26613.

[14] KBESSHO，DATE K，HAYASHI M，et al. An Introduction to Himawari-8/9—Japan's New-Generation Geostationary Meteorological Satellites[J]. Journal of the Meteorological Society of Japan，2016，94(2)：151-183.

[15] PURBANTORO B，AMINUDDIN J，MANAGO N，et al. Comparison of Cloud Type Classification with Split Window Algorithm Based on Different Infrared Band Combinations of Himawari-8 Satellite[J]. Advances in Remote Sensing，2018，7(3)：218-234.

[16] ZHANG C W. The Cloud-type Classification Research and its Application for the New Generation Geostationary Satellite Himawari-8[D]. Nanjing：Nanjing University，2019.

[17] LI Y Y，FANG L Z，KOU X W. Principle and Standard of Auto-Observation Cloud Classification for Satellite，Ground Measurements and model[J]. Chinese Journal of Geophysics，2014，57(8)：2433-2441.

[18] LIU Y. Remote Sensing Precipitation Forecasting Model Using Radar and Geostationary Meteorological Satellite and Analysis on Precipitation Predictability[D]. Beijing：Chinese Academy of Sciences，2014.

[19] FENG F，WANG S，WANG C，et al. Learning Deep Hierarchical Spatial-Spectral Features for Hyperspectral Image Classification Based on Residual 3D-2D CNN[J]. Sensors，2019，19(23)：5276.

[20] ROY S K，KRISHNA G，DUBEY S R，et al. HybridSN：Exploring 3D-2D CNN Feature Hierarchy for Hyperspectral Image Classification[J]. IEEE Geoscience and Remote Sensing Letters，2020，17(2)：277-281.

WANG Shan，born in 1981，Ph.D，professor. His main research interests include image processing and artificial intelligence.

XU Chu-yi，born in 1995，postgraduate. Her main research interests include meteorological data processing and so on.