文章编号：2095-4980(2024)09-0959-08

基于 RISC-V 架构的行人定位 SoC 系统设计

喻　胜[1]，史超凡[2]

(1.成都斯达康科技有限公司 光网络系统部，四川 成都 610041；2.电子科技大学 信息与通信工程学院，四川 成都 611731)

摘　要：行人定位方法中，捷联式惯导定位系统需要处理惯性测量单元(IMU)传感器的数据，通过算法处理后得到行人的位置，因此对于芯片实时性以及低功耗有很高的要求。由于行人定位算法大多基于浮点传感器数据开发，一般要求终端设备能够处理浮点数据。第五代精简指令集(RISC-V)架构作为一种开源架构，能节约架构授权费，在物联网领域有着广泛应用，并且其浮点(F)和向量(V)等高性能扩展指令能够很好地满足行人定位算法对实时性的要求。针对行人定位系统的特定性能要求，提出了一种基于浮点内核向量处理器优化RISC-V架构的行人定位片上系统(SoC)，并在实际系统中进行验证。与多个准32位架构RISC-V处理器以及高层次综合组件(HLS)生成的算法专用IP(locate_IP)的标准处理器方案的性能对比分析表明，该设计实现了34倍的性能提升以及5.6倍的能效提升，满足了微终端的要求。

关键词：行人定位系统；第五代精简指令集计算；现场可编程逻辑阵列；片上系统

中图分类号：TN47；TP332　　**文献标志码：**A　　**doi：**10.11805/TKYDA2023410

Design of pedestrian positioning SoC based on RISC-V architecture

YU Sheng[1]，SHI Chaofan[2]

(1.Department of Optical Network System，Chengdu Starcom Technology Limited，Chengdu Sichuan 610041，China；2.School of Communication and Information Engineering，University of Electronic Science and Technology of China，Chengdu Sichuan 611731，China)

Abstract：In pedestrian positioning methods, the strapdown inertial navigation system requires processing data from the Inertial Measurement Unit(IMU) sensors, and the position of the pedestrian is obtained after algorithmic processing, thus placing high demands on the real-time performance and low power consumption of the chip. Since most pedestrian positioning algorithms are developed based on floating-point sensor data, the terminal device is generally required to handle floating-point data. The fifth-generation Reduced Instruction Set Computer(RISC-V) architecture, as an open-source architecture, can save on architectural licensing fees and has a wide range of applications in the field of the Internet of Things. Moreover, its floating-point(F) and vector(V) high-performance extension instructions can well meet the real-time requirements of pedestrian positioning algorithms. In response to the specific performance requirements of the pedestrian positioning system, a System on Chip (SoC) for pedestrian positioning based on a floating-point core vector processor optimized RISC-V architecture is proposed and verified in actual systems. A performance comparison analysis with several quasi-32-bit architecture RISC-V processors and standard processor schemes of algorithm-specific IPs (locate_IP) generated by High-Level Synthesis(HLS) components shows that the design has achieved a 34-fold improvement in performance and a 5.6-fold improvement in energy efficiency, meeting the requirements of micro terminals.

Keywords：Pedestrian Navigation System；Reduced Instruction Set Computing V(RISC-V)；Field Programmable Gate Array(FPGA)；System on Chip(SoC)

行人定位是指专门针对行人的定位，通过处理传感器数据得到行人实时的位置[1]。目前常见的行人定位方式

收稿日期：2023-12-16；**修回日期：**2024-03-06

基金项目：国家自然科学基金资助项目(61973056)

有：基于全球定位系统(Global Positioning System，GPS)的定位、基于蓝牙WIFI等信号的定位、基于惯性传感器的定位等。基于GPS和蓝牙等传感器需要外部设备支持，无法实现自主定位，在没有预装载基站等设备的地方难以使用；使用手机中的惯性传感器进行定位虽然方便，但很难有较高精确度，在一些场景下，比如火场救援和野外探险等领域无法使用。在这些场景下，使用捷联式惯性导航定位方案，通过将高确精度惯性传感器固定到人身上，得到的加速度计和陀螺仪数据更加稳定，并且通过特定的算法可以实现高精确度定位，相比于使用GPS等方案，其可实现的定位场景更多，并且不受外界信号干扰，能够实现自主定位。现有的行人定位处理器大多基于安谋精简指令机器(Acorn RISC Machines，ARM)或者x86架构，或者基于这2种架构的异构系统[2−3]。ARM和x86均为闭源架构，而基于第五代精简指令集(RISC−V)架构的处理器，拥有开源的指令架构，可以节省架构授权的费用，并且RISC−V具有良好的生态，目前已经在物联网领域有着广泛应用[4]。由于其模块化的指令设计，系统设计者可以根据需要选择适合自己的指令架构，从而取得更好的能效比。

1 捷联式惯导行人定位系统

采用RISC−V架构实现一种基于捷联式惯导定位方案的行人定位系统(Pedestrian Navigation System，PNS)。RISC−V由于其开源指令架构，有多种开源处理器的具体实现。典型的32位RISC−V内核处理器实现包括ibex, e203以及CV32E40P等。其中ibex是一款两级流水线处理器[5]，使用system verilog语言编写，内核可高度参数化，可根据配置选项支持整数/嵌入式(Integer/Embedded，I/E)、乘除法(Multiply，M)、压缩(Compress，C)、位操作(Bit operation，B)扩展指令，并且经过多次流片验证。e203也是一款两级流水线处理器，支持整数乘除法原子压缩(Integer Multiply Atomic Compress，IMAC)扩展指令。CV32E40P是一款四级流水线处理器[6]，支持整数乘除法压缩(Integer Multiply Compress，IMC)扩展指令并且可以选择支持浮点指令，进而更好地支持一些传感器的数据处理。本文基于e203内核进行优化设计，并且基于e203开源片上系统(SoC)进行后续验证测试。

在特定领域以RISC−V基本指令作为控制部分，扩展或者自定义指令进行任务卸载能够有效提升能耗比。文献[7]基于RV32E_Zfinx指令架构设计了一款低功耗内核，在处理二氧化碳传感器任务上，相比于RV32IMF架构的CV32E40P内核，减少了53.3%的能耗；相比于RV32E架构的micro−ibex内核，减少了94.8%的能耗。文献[8]基于RISC−V向量扩展指令，以mircro blaze软核加向量协处理器的方式进行微架构实现，在向量和矩阵基准测试中，相比于标量处理器，能减少20%~90%的能量消耗，并且有2~78倍的性能提升。文献[9]基于RV32EV架构，设计了一款向量处理器，在一些基准测试下，性能有5.8倍提升，而资源消耗优化2.6倍。文献[10]将RISC−V处理器用于CCSDS401.0−B−32标准的纠错码算法运算中，在Forward−Backward MM decoder运算中能取得8.48 kbps的吞吐率。文献[11]通过支持自定义乘积累加运算(Multiply ACcumulate，MAC)以及二维卷积运算(2−dimension Convolution，CONV2)指令，在神经网络应用上实现了1.35~1.7倍的性能提升。相应地针对PNS系统，本文研究了基于RISC−V架构的能耗提升方案，设计了基于FPGA的实际系统并对比分析其功耗表现。同时，本文还使用高层次综合组件(Vivado High Level Synthesis，Vivado HLS)将定位算法打包为独立的IP，同时对比该IP与处理器的性能表现。

2 捷联式惯性导航算法实现

捷联式惯性导航算法将惯性导航传感器连接到行人身上，通过积分运算得到行人位置，并通过行人的位姿解算对得到的行人位置进行修正，将行人的运动状态作为观测量，修正积分运算的状态过程。典型的定位方式包括“零速更新”算法，通过判定行人是否处于行走状态作为观测量，对运动状态进行更新，从而对传感器误差进行修正。见图1，该算法计算量主要包括3个部分：a) 位姿解算，即通过坐标系变换将传感器坐标系变换为绝对坐标系；b) 零速点的判定；c) 滤波算法，一般使用扩展卡尔曼滤波或者粒子滤波对状态向量进行处理。由于不同的定位算法所使用的传感器数据维度不同，采样频率也不同，并且使用的滤波算法以及零速点判定方式均不同，因此如果使用传统的专用集成电路(Application−Specific Integrated Circuit，ASIC)方式进行加速，对于某一个算法适用，而稍微更换一下算法就无法运算。而通过支持扩展指令，由于粒度较小，能够较好地适应算法更新。因此作为一种微终端应用，PNS适宜采用微处理器实现。本文在RISC−V处理器上实现优化的捷联式惯性导航算法。对于并行度较低的部分，依靠编译器自动编译出浮点指令进行加速运行，而对于并行度较高的部分，则以使用内联汇编的方式调用向量指令加

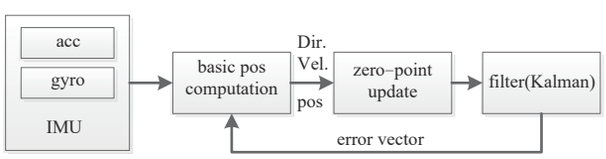


Fig.1 PNS strapdown inertial navigation algorithm
图1 PNS捷联式惯性导航算法

速运行。

2.1 SoC架构系统设计

当前存在多种计算架构可以实现捷联式惯性导航系统，包括：DSP、ARM或单片机作为核心处理器；DSP或ARM与单片机的双机组合系统；DSP和ARM组合、DSP和FPGA的组合以及SoC和可编程片上系统(System on Programmable Chip，SOPC)结构等[12]。其中SoC架构能够在一颗芯片上集成加速器与处理器，从而减少数据搬运的时间，进而提高处理速度，因此本文采用SoC架构作为行人定位的处理器。见图2，本文的SoC系统采用32位RV32IMAFC核构建，整数乘除法浮点原子压缩(Integer Multiply Atomic Float Compress，IMAFC)分别表示该核支持整数、乘法、原子操作、浮点和压缩指令。本文设计包含3个时钟域：常开域使用32.768 kHz低速时钟，主要用于实时时钟、看门狗以及电源控制器；调试域包括一个调试(Debug)单元，通过联合测试工作组(Joint Test Action Group，JTAG)协议对处理器进行调试；主域使用33 MHz时钟，处理器核、总线、内存以及协处理器均在这个时钟频率下。主域部分包含64 KB的指令紧耦合内存(Instruction Tightly–Coupled Memory，ITCM)以及数据紧耦合内存(Data Tightly–Coupled Memory，DTCM)，通过总线连接通用异步收发器(Universal Asynchronous Receiver/Transmitter，UART)SPI和I2C等多种外设，并且通过蜂鸟协处理器指令扩展(Nuclei Instruction Co–processor Extension，NICE)、接口外挂向量处理器，对算法中的矩阵运算进行加速。

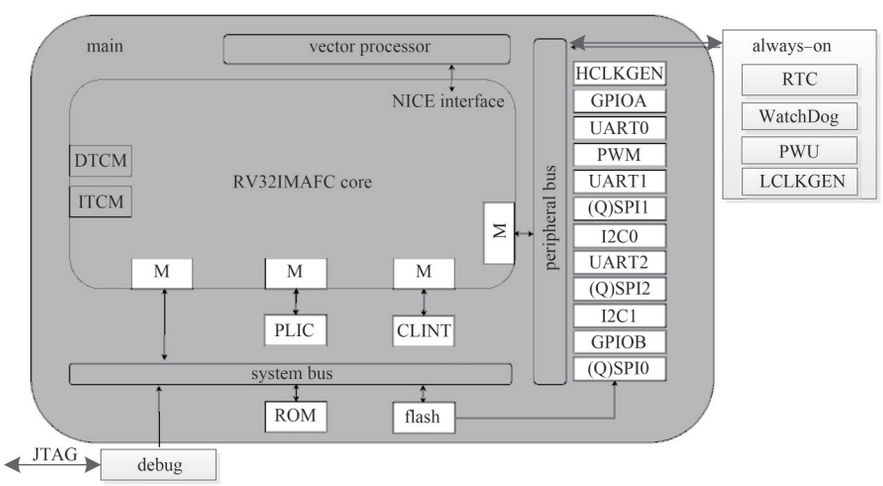


Fig.2 PNS SoC system architecture
图2 PNS SoC系统架构设计

2.2 RISC–V浮点内核设计

针对本文PNS系统的算法需求，在e203 RV32IMAC架构的内核基础上进行浮点内核的优化设计，见图3。在不改变流水线架构的情况下支持RISC–V的“F”扩展指令：a) 加入32个浮点寄存器。RISC–V的F扩展规定了32个单独的浮点寄存器，防止与标量寄存器冲突。本文在SoC系统进行相应设计。b) 修改解码单元，将浮点指令的opcode加入解码单元中，并且在派遣单元加入从浮点寄存器组中读出的浮点操作数，将打包好的数据发送给ALU。c) 设计浮点数据通路，本文采用浮点处理器FPnew的架构实现浮点计算的运算器[13]。

2.3 向量处理器优化

浮点处理单元能够处理算法中的小批量运算，但是对于算法中存在的矩阵运算，即使将循环展开，也很难取得较高的性能。这是由于矩阵运算需要不断进行数据搬运，在搬运数据后进行运算，而标量处理器无法同时执行多条指令，也无法同时执行多个数据运算，导致矩阵运算无法流水化运行，运算效率很低。本文设计的SoC平台为向量处理器提供了单独访问数据内存的接口，因此向量处理器可以独立访问内存，从而实现流水化数据处理。基于RISC–V向量指令子集，本文对向量协处理器进行了优化设计：a) 对向量指令进行分层，针对行人定位算法的最小架构进行设计实现，实现功耗与性能的最优系统。b) 设计多条流水线，支持数据搬运单元、整数运算单元以及浮点运算单元3条流水线并行运行。设计多口向量寄存器模块支持多流水线同时访问以及同时写回。c) 设计指令状态记录单元，记录指令运行状态，防止数据冲突。d) 设计单独的访存单元，通过芯片内部总线(Internal Chip Bus，ICB)协议进行独立内存访问，支持stride–load stride–store对内存数据进行搬运。

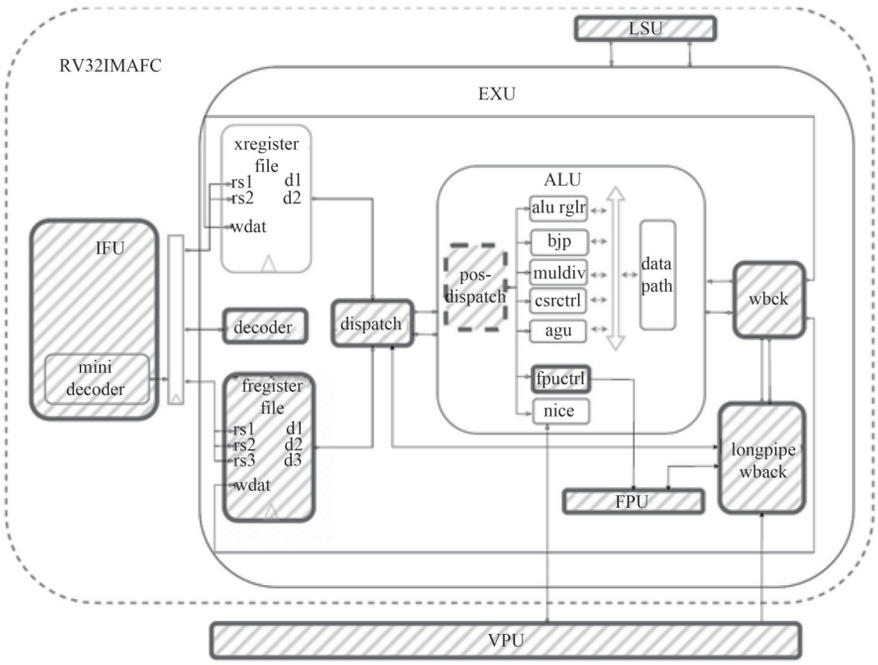


Fig.3 Design of RV32IMAFCV floating point kernel

图3 RV32IMAFCV浮点内核设计

RISC-V系统包含的向量指令数量较多，大约有400多条，指令全部实现需要消耗大量硬件资源。针对本文PNS系统，采用指令分层的模式[14]，选择了算法实现中用到的部分指令。本文实现的指令子集可以分为4组，见表1，其中向量设置(Vector Set，VSET)指令用于设置向量长度以及寄存器分组，向量加载/存储(Vector Load/Store，VLd/VSt)指令负责向量寄存器与内存进行数据交换，向量浮点(Vector Float，VFLOAT)指令包括浮点向量乘法、除法、加法以及乘累加等操作，向量整数(Vector Integer，VINT)指令包括整数乘法和除法等计算操作。

表1 PNS向量指令分层子集

Table1 Hierarchical subset of PNS vector instruction

group	VSET	VLd/VSt	VINT			VFLOAT	
Instruction Name(rvv1.0)	Vsetvli Vsetivli Vsetvl	Vle.v Vlse.v Vse.v Vsse.v	Vadd.vv Vsub.vv Vmv.v.v Vdiv.vv Vmul.vv Vmadd.vv Vmacc.vv Vredsum.vs	Vadd.vx Vsub.vx Vmv.v.x Vdiv.vx Vmul.vx Vmadd.vx Vmacc.vx	Vadd.vi Vmv.v.i	Vfadd.vv Vfsub.vv Vfdiv.vv Vfmul.vv Vfmadd.vv Vfmacc.vv Vfredusum.vs Vfsqrt.v	Vfadd.vf Vfsub.vf Vfdiv.vf Vfmul.vf Vfmadd.vs Vfmacc.vf

针对PNS系统优化目标，本文SoC系统向量处理器(Vector Process Unit，VPU)采用优化的超标量处理器架构[15]，使用3条独立流水线，分别可以单独运算整数、浮点以及数据搬运指令，见图4。与普通超标量处理器不同的是，由于向量处理器每条指令需要存取多个数据，而数据通路带宽受限于SoC带宽，且向量长度不固定，很难实现单周期取出指令所需的所有操作数，因此取数及运算后的数据写回一般需要多个周期，这就导致需要一个单独的模块处理操作数的取出与结果写回。此外，向量寄存器位宽一般较大，使用寄存器实现的资源利用效率较低[16]。同时，在FPGA系统实现时大位宽的寄存器综合容易造成布线拥堵。为此，本文提出在FPGA中使用lut ram实现向量寄存器组，而采用multi-bank ram实现多口向量寄存器。

在解决读写操作数方面，本文系统使用仲裁结构保证3条流水线能够同时读写向量寄存器组[17]。在指令冒险方面，则通过阻止存在写后读(Read After Write，RAW)、先后读写(Write After Write，WAW)、读后写(Write After Read，WAR)冒险的指令下发，防止产生数据冲突。

3 系统验证和分析

3.1 验证方案

为验证本文设计PNS系统的性能，分别从仿真和FPGA实现系统进行了验证，见图5。首先对寄存器转换级

(Register Transfer Level，RTL)代码进行系统级仿真验证，然后再进行FPGA系统实现，通过行人定位系统平台对数据解算进行实时验证。主要步骤如下：1) 首先将算法进行向量化。在RISC-V仿真器上将仿真后的软件读取到ITCM中，整个SoC主域部分的RTL代码都可以加入验证平台中，从而进行系统级的仿真。加入时钟信号启动SoC后就可以对CPU进行调试。2) 在仿真通过后，将RTL代码编译成bitstream，下载到Xilinx FPGA芯片xc7a200tfbg484-2上进行FPGA验证。3) 最后构造完整的实现系统，将传感器数据发送到FPGA中，将定位后的数据实时显示出来，从而对整个系统设计进行验证。

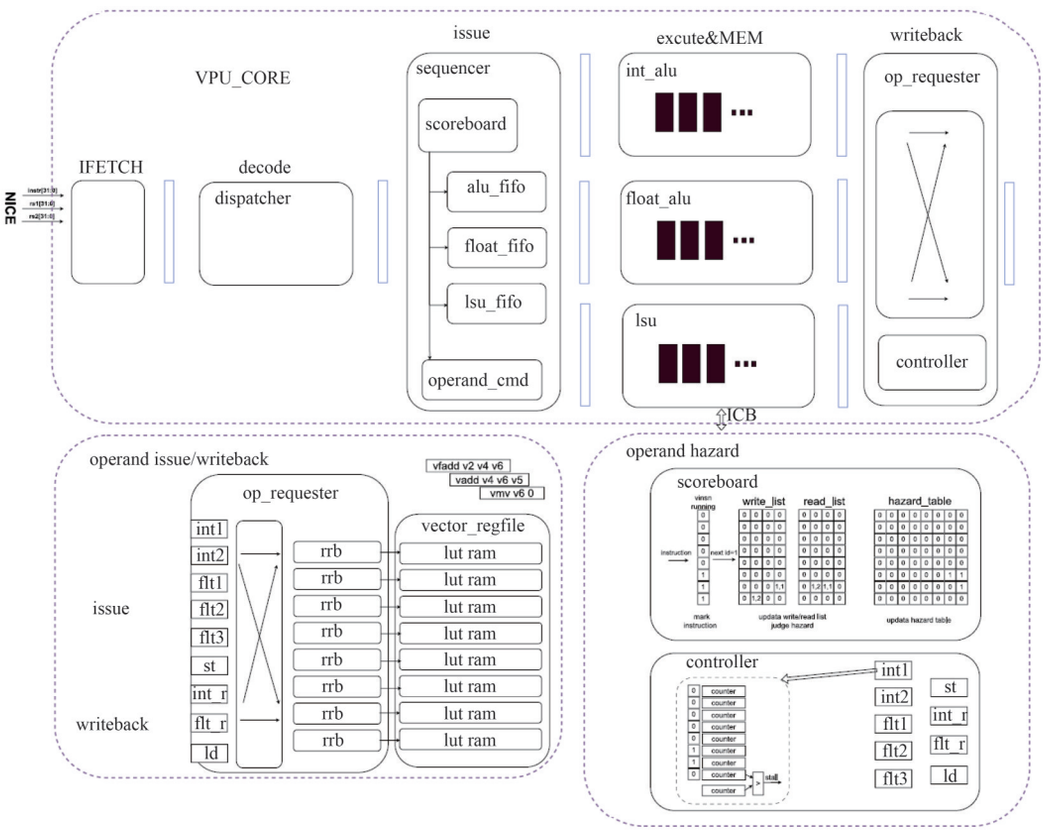


Fig.4 Architecture of VPU system design
图4 VPU系统设计架构

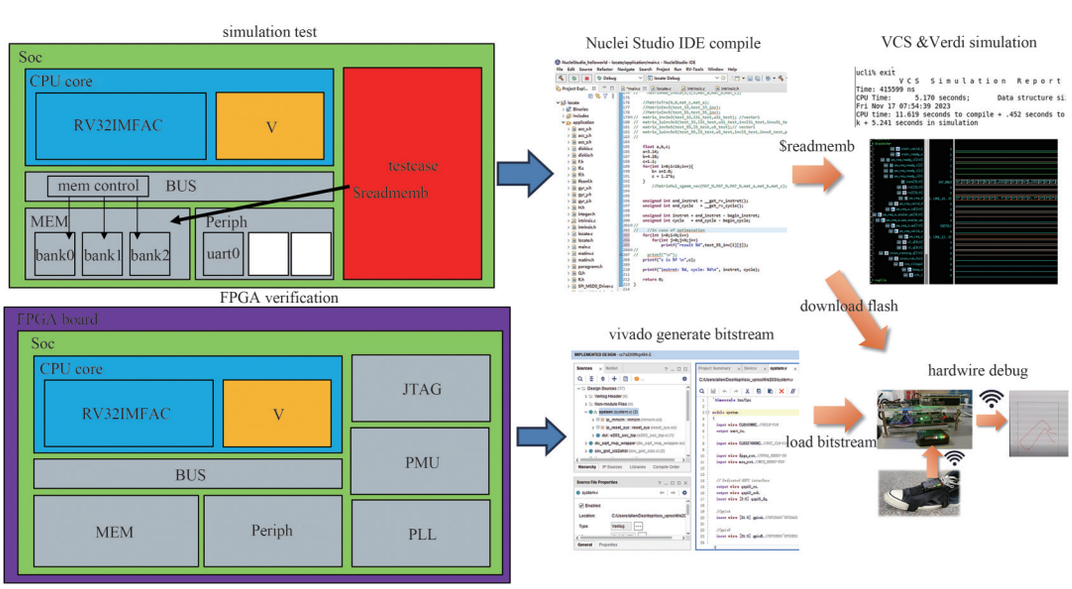


Fig.5 System verification method
图5 系统验证方案

3.2 指令分析

本文使用传感器收集真实数据，在仿真阶段启动处理器对这些数据进行运算，并通过脚本导出指令进行分析。首先将指令分为整数、浮点以及向量3组，其中整数指令包含I,M,A,C扩展指令，Arithmetic是加减以及逻辑运算指令，MUL是乘法指令，DIV是除法指令，Ld/St是访存指令，Branch/Jump是跳转指令，Misc指令为剩余的其他指令，比如fence指令。向量组中的VLd/VSt代表向量load/store指令，VFMAC代表向量浮点计算指令，VINT代表向量整数计算指令，VSET代表向量配置指令，浮点指令组中Fld/FSt代表浮点load/store指令，FMAC代表浮点加减乘指令，FDiv代表浮点除法指令，FMisc代表浮点其余非计算非数据搬运等指令，比如符号注入等。

见图6，在使用向量和浮点扩展后，指令数量下降了接近20倍，这是因为不使用硬浮点的情况下，软件将跳转到软浮点函数部分，利用整数指令进行IEEE754格式的浮点运算。在开启硬浮点后，只需要一条指令即可完成一次浮点运算，因此可以节约大量时间和指令。

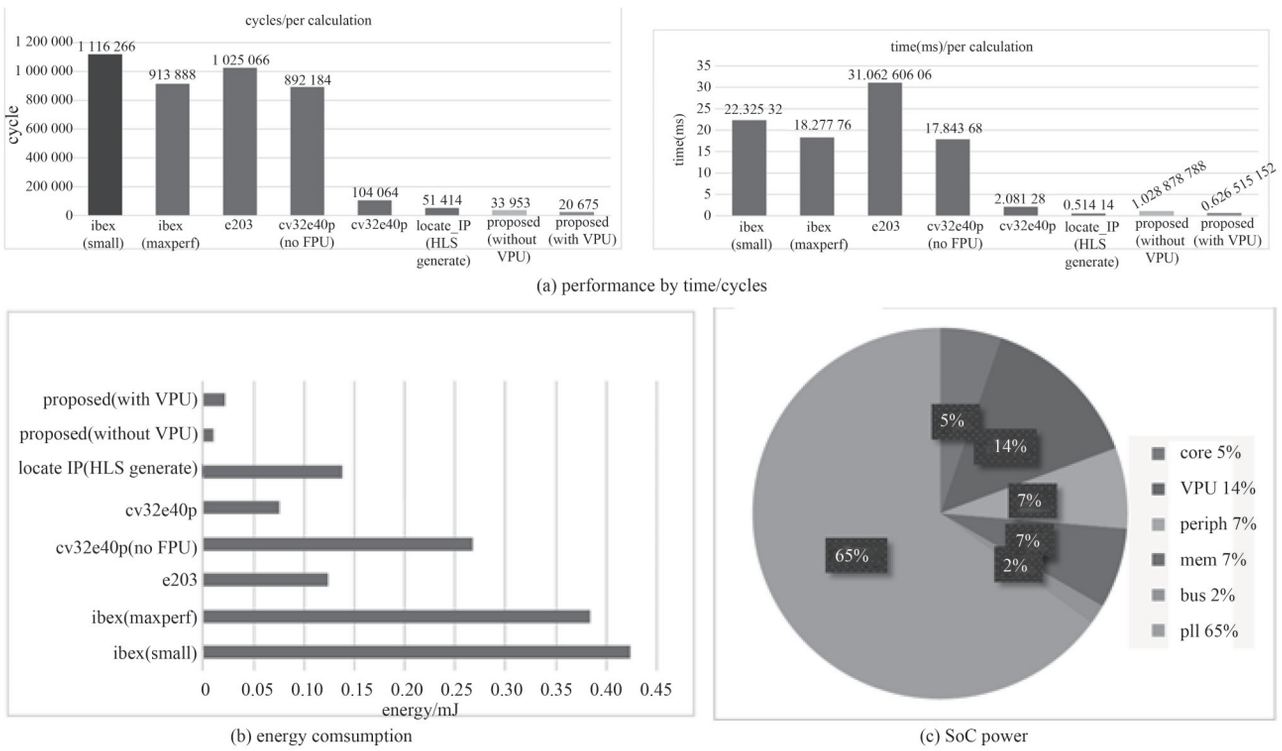


Fig.6 Performance and power analysis

图6 性能和功耗分析

指令占比最多的仍然是数据搬运指令，几乎占到一半甚至以上，因为任何操作都需要先使用搬运指令将数据从内存中搬入到寄存器中运算，运算后再搬出，频繁使用搬运指令是不可避免的。

在同时启用浮点和向量扩展后二者的占比仍然是比较低的，因为不可避免地会使用到各种整数标量指令，即使算法运算均使用float变量，其中部分原因是由于各种函数调用导致经常进行函数入栈出栈保存操作。

开启向量扩展后，指令数量相比只使用浮点扩展减少了37%，这主要是由于浮点指令减少了很多，但同时标量指令也增加了25%。这是因为在向量指令的架构中，需要标量处理器计算保存指针地址以及向量长度，而在设计矩阵运算的相关应用程序接口(Application Programming Interface，API)函数时，由于需要保持函数的可配置性，又需要增加很多标量指令进行指针运算，这导致部分性能损失。完全循环展开能够减轻这个问题造成的性能损失，但由于目前编译器仍然不能很好地支持自动化向量代码，并且循环展开也会导致更大的代码体积，因此在本文中大部分矩阵运算仍然使用可配置的API函数对调用向量指令对矩阵运算进行加速。

根据运行时间以及指令分析，可以得到如下结论：

1) 使用扩展指令，并对算法进行向量化处理能有效减少指令数量，在并行度较高的运算中能够借助专用的向量协处理器对算法进行有效加速。

2) VPU在小型矩阵运算中的效果不一定比浮点运算单元(Float-point Processing Unit，FPU)好，并且需要手动循环展开才能获得较好性能。由于需要标量处理器保存指针等信息，VPU指令也需要通过标量处理器发送过来，

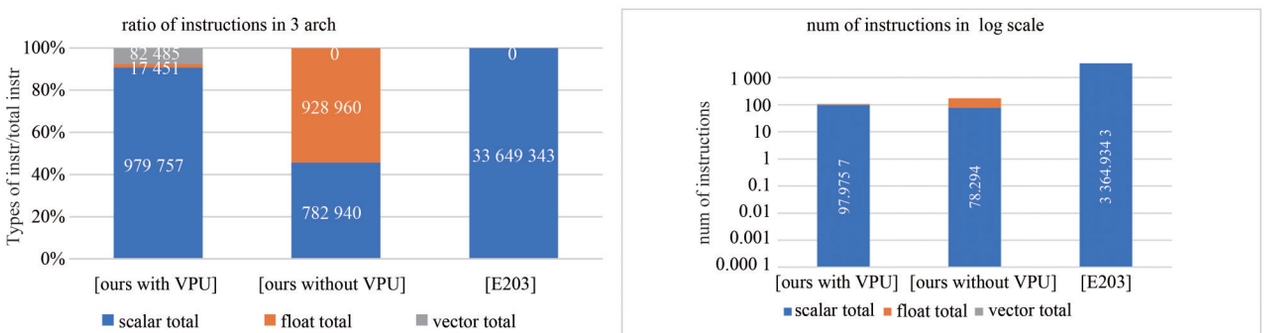
因此其性能受限于标量处理器。

3) load/store 占所有指令一半以上，且一般访存指令所需时间都超过一个周期，因此提升访存指令执行速度能有效提升行人定位算法执行速度。

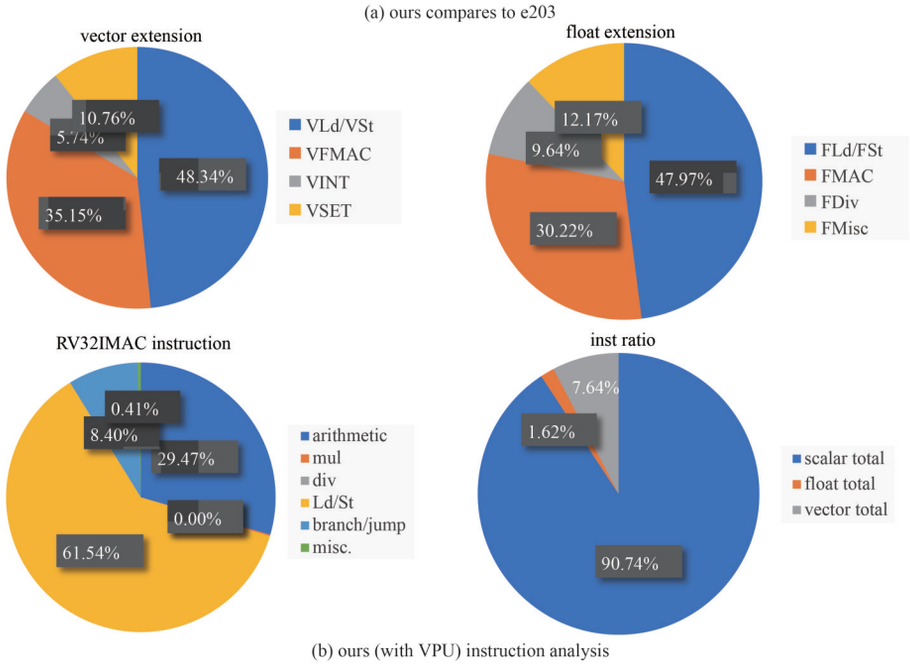
4) 即使是全部使用浮点进行计算的代码，标量指令仍然占大部分，因此有针对性地对标量指令进行加速也可以起到不错的效果。

3.3 性能对比

见图7，本文对比了多个32位架构RISC-V处理器以及HLS生成的算法专用IP(locate_IP)，在不考虑频率的情况下，本文设计的RV32IMAFC架构的内核实现了周期数最少，比第二的cv32e40p(with fpu)快5倍，相比e203速度提升了34倍。同时可以看到具有浮点扩展的cv32e40p处理器也能够实现较高性能，但其动态功率较高。



(a) ours compares to e203



(b) ours (with VPU) instruction analysis

Fig.7 Instruction analysis

图7 指令分析

当考虑时钟频率后，用时最短的是locate_IP模块，但其消耗的资源也是最多的。具有浮点扩展的cv32e40p执行时间也较短，且该内核能够综合在一个较高频率。由于组合逻辑较长的问题，本文设计的内核和协处理器在所使用的FPGA上均无法跑到一个较高频率，因此本文统一使用浮点内核的最高频率为33 MHz。

在能耗比方面，不使用向量协处理器的浮点内核的能效比最高，每次执行算法只需要消耗0.008 mJ。使用向量协处理器后，虽然提升了性能，但是同样执行一次算法所需能量也变多了。e203由于使用了一些低功耗设计，导致其在FPGA综合实现时动态功耗非常低，执行一次算法虽然所需时间较长，但所需能量只有0.085 mJ。ibex虽然执行时间与e203接近，但由于在FPGA上实现后其动态功耗较高，因此消耗能量比e203多很多。运算最快的locate IP虽然在时间上最短，但是其消耗能量也较多，平均执行一次算法需要消耗0.138 mJ能量，甚至高于e203处理器。

4 结论

RISC-V架构在物联网领域有着广泛应用。本文基于RISC-V架构的行人定位SoC系统针对捷联式惯导定位系统应用充分优化了RISC-V的指令架构，利用其F、V等高性能扩展指令满足了行人定位算法对实时性的要求。在实际系统中的验证结果表明，与已有方案相比，本文系统实现了34倍的性能提升以及5.6倍的能效提升。基于优化RISC-V架构的行人定位系统不仅可以提高行人定位系统的性能和能效，降低系统成本，而且为物联网领域的特定指令架构系统提供了设计指导。

参考文献：

[1] WANG Qu, LUO Haiyong, WANG Jianquan, et al. Recent advances in pedestrian navigation activity recognition: a review[J]. IEEE Sensors Journal, 2022,22(8):7499-7518. doi:10.1109/JSEN.2022.3153610.

[2] LI Hongpeng, LIU Haichao, LI Zhen, et al. Adaptive threshold-based ZUPT for single imu-enabled wearable pedestrian localization[J]. IEEE Internet of Things Journal, 2023,10(13):11749-11760. doi:10.1109/JIOT.2023.3243296 .

[3] CHEN Tianyu, YANG Gongliu, CAI Qingzhong, et al. A novel calibration method for gyro-accelerometer asynchronous time in foot-mounted pedestrian navigation system[J]. Sensors, 2021,22(1):209. doi:10.3390/s22010209.

[4] 刘畅,武延军,吴敬征,等. RISC-V指令集架构研究综述[J]. 软件学报, 2021,32(12):3992-4024. (LIU Chang, WU Yanjun, WU Jingzheng, et al. Survey on RISC-V system architecture research[J]. Journal of Software, 2021,32(12):3992-4024.)

[5] PULP P. Ibex.GitHub repository[DB/OL]. [2023-11-01]. https://github.com/pulp-platform/ibex .

[6] OPENHWGROUP. cv32e40p.GitHub repository[DB/OL]. [2023-11-01]. https://github.com/openhwgroup/cv32e40p.

[7] NUNEZ P R, CASTELLS R D, TERES L. RisCO_2: implementation and performance evaluation of RISC-V processors for low-power CO_2 concentration sensing[J]. Micromachines, 2023,14(7):1371. doi:10.3390/mi14071371.

[8] ASSIR I A, ISKANDARANI M E, SAGHIR H R A, et al. Arrow: a RISC-V vector accelerator for machine learning inference[DB/OL]. (2021-07-15)[2023-11-01]. https://arxiv.org/abs/2107.07169.

[9] JOHNS M, KAZMIERSKI T J. A minimal RISC-V vector processor for embedded systems[C]//2020 Forum for Specification and Design Languages(FDL). Kiel: IEEE, 2020:1-4. doi:10.1109/FDL50818.2020.9232940.

[10] KUO Yaoming, FLANAGAN M F, GARCIA H F, et al. Integration of a real-time CCSDS 410.0-B-32 error-correction decoder on FPGA-Based RISC-V SoCs using RISC-V vector extension[J]. IEEE Transactions on Aerospace and Electronic Systems, 2023, 59(5):5835-5846. doi:10.1109/TAES.2023.3266314.

[11] MOUSAVIKIA S K, GHOLIZADEHAZARI E, MOUSAZADEH M, et al. Instruction set extension of a RISC-V based SoC for driver drowsiness detection[J]. IEEE Access, 2022(10):58151-58162. doi:10.1109/ACCESS.2022.3177743.

[12] 王猛,吴一,张宇,等. 捷联惯性导航计算机系统架构发展综述[J]. 电子产品世界, 2023,30(2):33-36. (WANG Meng, WU Yi, ZHANG Yu, et al. Development of strap down inertial navigation computer system architecture[J]. Qutlook of Electronic Technology, 2023,30(2):33-36.)

[13] MACH S, SCHUIKI F, ZARUBA F, et al. FPNEW: an open-source multiformat floating-point unit architecture for Energy-Proportional transprecision computing[J]. IEEE Transactions on Very Large Scale Integration(VLSI) Systems, 2021,29(4):774-787. doi:10.1109/TVLSI.2020.3044752.

[14] WHITE C. Design and implementation of RVV-Lite: a layered approach to the official RISC-V vector ISA[D]. Vancouver, University of British Columbia, 2023. doi:10.14288/1.0431080.

[15] 姚永斌. 超标量处理器设计[M]. 北京:清华大学出版社, 2022. (YAO Yongbin. Super scalar RISC processor design[M]. Beijing, China: Tsinghua University Press, 2022.)

[16] LEE Yunsup. Decoupled vector-fetch architecture with a scalarizing compiler: UCB/EECS-2016-117[R]. (2016-05-24). https://www2.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-117.html.

[17] CAVALCANTE M, SCHUIKI F, ZARUBA F, et al. Ara: a 1-GHz+ scalable and energy-efficient RISC-V vector processor with multiprecision floating-point support in 22-nm FD-SOI[J]. IEEE Transactions on Very Large Scale Integration(VLSI) Systems, 2020,28(2):530-543. doi:10.1109/TVLSI.2019.2950087.

作者简介：

喻 胜(1973-)，男，博士，高级工程师，主要研究方向为通信系统设计、计算机算法设计. email:alex.yu2005@163.com.

史超凡(2000-)，男，在读硕士研究生，主要研究方向为物联网技术.