基于矢量处理器的可编程并行MIMO检测器设计

范阿冬，秦晓卫，戴旭初

(中国科学技术大学电子工程与信息科学系 合肥 230027)

【摘要】针对部分层间并行球形译码(PIPSD)算法的特点，基于超长指令字(VLIW)和单指令多数据(SIMD)混合结构的矢量处理器原型，合理安排处理器结构参数，选择合适的寄存器数目和长度。根据算法和结构的相互作用特点，挖掘算法内在的数据并行性和指令并行性，设计高效的指令集和相应的功能单元，软硬件协同优化VLIW分布，在支持多种配置的基础上尽可能减小执行周期数，提高译码吞吐率，从而设计出高效的基于专用指令集矢量处理器的可编程并行MIMO检测器。

关 键 词 多天线检测； 单指令多数据； 矢量处理器； 超长指令字

中图分类号 TN929.5 **文献标志码** A doi:10.3969/j.issn.1001-0548.2016.02.004

Programmable Parallel MIMO Detector Design Based on Vector Processor

FAN A-dong, QIN Xiao-wei, and DAI Xu-chu

(Department of Electronic Engineering and Information Science, University of Science and Technology of China Hefei 230027)

Abstract Aimed at the features of PIPSD (Partial Inter-layer Parallel Sphere Detector) algorithm and based on the prototype of a hybrid architecture of VLIW (Very Long Instruction Word) and SIMD (Single Instruction Multiple Data) vector processor, a highly efficient parallel MIMO detector is designed by tuning the architecture parameters. The detector can be used to explore the intrinsic parallelism of the algorithm, customize the function units, instruction set, hardware/software co-optimizing VLIW distribution, and support multi-mode detection while minimizing execution cycles and maximizing decoding throughput.

Key words MIMO detection; SIMD; vector processor; VLIW

多输入多输出(multiple input multiple output，MIMO)被广泛应于移动通信系统中，然而MIMO技术在成倍提高系统数据速率的同时，也使得接收端复杂度呈指数增长[1]。另一方面，无线通信的标准也越来越要求MIMO具有支持不同配置模式的能力，如LTE-A中要求能够支持最多8天线及64QAM调制的配置模式[2]。因此，能够支持多种配置的灵活可编程MIMO检测器将成为未来无线通信终端设备必不可少的组成部分。

数字通信飞速发展的同时也对硬件实现平台的灵活性、可配置和可重用性提出了更高要求。通用标量处理器或DSP的处理能力不足，专用IC电路(application specific integrated circuit, ASIC)以及FPGA的灵活性又不够[3]，而针对特定目标应用设计的专用指令集处理器(application specific instruction-set processor, ASIP)，既能够针对该应用相应地优化结构与设计特殊指令以提供较强的处理能力，又能够以可编程和可扩展指令集的特点提供很好的灵活性，因此可以很好地适应新型应用的要求[4]。文献[5]针对MMSE-IC检测算法设计的EquiASIP，可以支持2*2和4*4天线配置，以及QPSK，16QAM和64QAM等调制阶数配置下的MIMO检测；文献[6]则针对MMSE算法设计了可编程的多模MIMO检测器FLEXDET，这些都是使用专用指令集处理器设计MIMO检测器的典型例子。

本文基于支持相邻层之间同时计算的部分层间并行球形译码(partial inter-layer parallel sphere decoder, PIPSD)算法[7]，分析算法中数据并行性和指令并行性，基于矢量处理器架构原型，设计寄存器堆，功能单元和相应的指令集，并分析程序特点以优化VLIW分布，从而完成高效的可编程并行MIMO检测器的设计。

收稿日期：2015－03－12；修回日期：2015－04－17
基金项目：国家973项目(2013CB329004)；国家科技重大专项(2013ZX03003013-004)
作者简介：范阿冬(1991－)，男，主要从事无线通信多天线检测技术方面的研究.

1 PIPSD算法简介

1.1 系统模型

考虑发射和接收天线数目分别为 N_t 和 N_r (设 $N_{\mathrm{t}}=N_{\mathrm{r}}=N$)的MIMO系统，其系统模型可表示为：

$$\boldsymbol{y}_c=\boldsymbol{H}_c\boldsymbol{s}_c+\boldsymbol{n}_c \tag{1}$$

式中，$\boldsymbol{y}_c$ 和 $\boldsymbol{s}_c$ 分别表示复数形式的接收信号和发射信号向量；$\boldsymbol{n}_c$ 为接收机噪声向量；$\boldsymbol{H}_c$ 是 $N_r\times N_t$ 的信道增益矩阵。

在各种MIMO检测算法[8]中，以K-best算法[9]为代表的广度优先球形译码算法以其优良的性能、较低的复杂度以及便于硬件实现的特点获得广泛应用。而尽管广度优先球形译码算法很容易挖掘出每一层内的数据并行性，从而加快处理速率，然而“层间搜索为串行”的特点成为了限制译码速率进一步提高的瓶颈。

1.2 PIPSD算法简介

为了进一步发掘层间的并行性，PIPSD算法引入一种新的实数检测模型[10]，即式(1)中信道矩阵 $\boldsymbol{H}_c$ 的一种新的实数形式，为：

$$\boldsymbol{H}=\begin{bmatrix}\mathrm{Re}(H_{c11}) & -\mathrm{Im}(H_{c11}) & \cdots & \mathrm{Re}(H_{c1N}) & -\mathrm{Im}(H_{c1N})\\ \mathrm{Im}(H_{c11}) & \mathrm{Re}(H_{c11}) & \cdots & \mathrm{Im}(H_{c1N}) & \mathrm{Re}(H_{c1N})\\ \vdots & \vdots & \ddots & \vdots & \vdots\\ \mathrm{Re}(H_{cN1}) & -\mathrm{Im}(H_{cN1}) & \cdots & \mathrm{Re}(H_{cNN}) & -\mathrm{Im}(H_{cNN})\\ \mathrm{Im}(H_{cN1}) & \mathrm{Re}(H_{cN1}) & \cdots & \mathrm{Im}(H_{cNN}) & \mathrm{Re}(H_{cNN})\end{bmatrix} \tag{2}$$

该矩阵经过QR分解之后得到的上三角矩阵$\boldsymbol{R}$的奇数行的对角线元素下一个元素总是零，即：

$$R_{2k-1,2k}=0\quad k=1,2,\cdots,N \tag{3}$$

这意味着在对每一层计算分支度量值并进行节点选择的过程中，从第一层开始每相邻两层之间是相互独立的，可以同时进行计算，而不同的两层与两层之间仍为串行，因此称为“部分层间并行”。

对第 $2k+1$ 层的每一个候选节点，在 $2k$ 层和 $2k-1$ 层的子节点的度量值时进行局部排序，并分别选择出 m_{2k} 和 m_{2k-1} 个候选节点(m_{2k} 和 m_{2k-1} 称为对应层的扩展系数)。这个过程称为“局部排序”。第 $2k+1$ 层的不同节点并行地执行该过程。

对第 $2k+1$ 层的每一个候选节点，将第 $2k$ 层和第 $2k-1$ 层局部排序得到的候选节点直接排列组合得到候选的部分节点路径，这个过程称为“快速组合”。

应用层间并行计算，同时采用局部排序与快速组合等手段，便可以得到完整的PIPSD检测算法。图1给出了PIPSD搜索过程的一个示例，其中天线配置为2×2，调制方式为16QAM，扩展系数向量为 $\boldsymbol{m}=[1,1,2,2]$。图中实线代表当前层的幸存节点，虚线代表被剪除的节点。

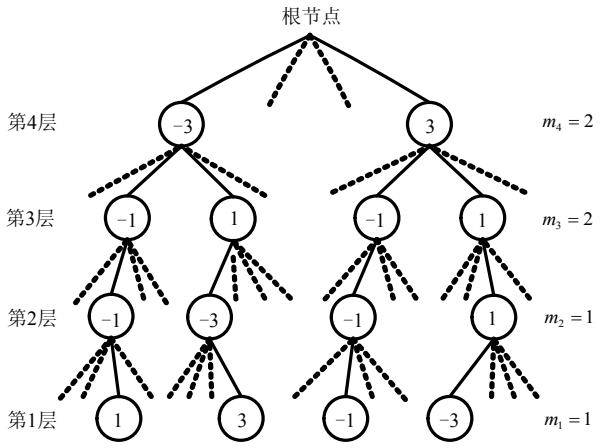

图1 PIPSD算法搜索树结构示例

1.3 PIPSD算法实现分析

从具体实现的角度对目标算法进行分析，PIPSD算法的代码结构分析如图2所示，图中小三角形代表顺序执行，圆圈箭头代表循环(后面括号中的是循环次数)，重叠框代表并行操作(对应指令级并行)，阴影框代表矢量化操作(对应数据级并行)。

从图2中可以看出PIPSD算法包含了丰富的并行性，计算路径度量的过程中可以将各个路径组合成矢量并行计算，局部排序和快速组合过程中各个路径之间也具有独立性，树搜索过程中奇数层和偶数层并行计算更使得并行性得以成倍提高。

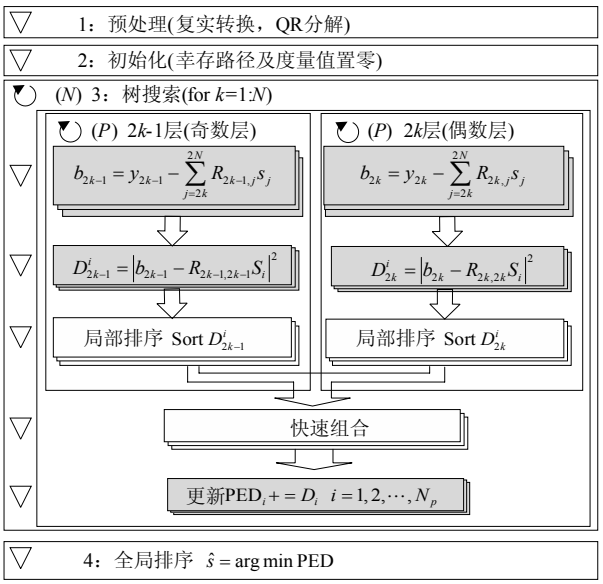

图2 PIPSD算法代码结构分析

2　基于矢量处理器的设计流程

基于矢量处理器的总体设计流程如图3所示，首先分析算法结构，找出算法中的数据并行性，分析算法的指令需求，初步确定处理器的总体结构和功能单元，实现之后再进行结构参数调整和性能优化。

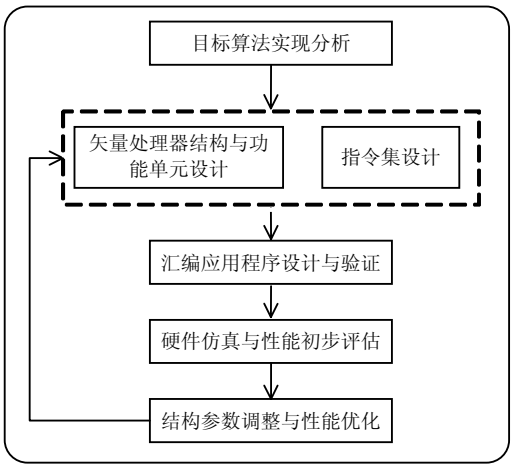

图3　基于矢量处理器的总体设计流程

图4描述了整个基于矢量处理器的开发流程中涉及到的工具链的设计流[11]。开发过程分为两个抽象层：LISA (language of instruction-set architecture)抽象层主要是使用LISA语言设计处理器结构和指令集，编写汇编程序并完成逻辑功能的仿真验证，开发环境是Synopsys Processor Designer开发套件；HDL(hardware description language)抽象层则是生成真实的硬件语言描述，并在实际的FPGA硬件环境中运行验证。

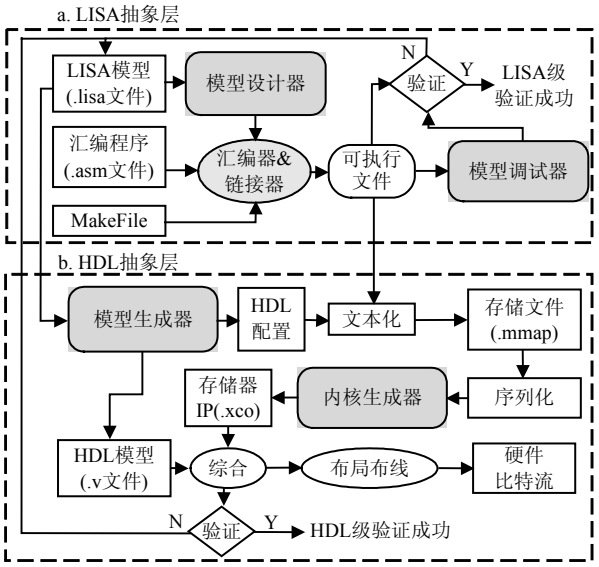

图4　基于矢量处理器的工具链设计

3　可编程MIMO并行检测器设计

PIPSD算法在层内具有数据并行性和指令并行性，在层间具有两层同时计算的并行性，充分发挥该并行性有赖于相应硬件平台的支撑。为此设计以矢量操作为特点，支持两层同时计算，并针对PIPSD算法特点进行优化的专用指令集矢量处理器。

3.1　矢量处理器总体架构

矢量处理器总体架构如图5所示，处理器采用基于精简指令集(reduced instruction set computer, RISC)的VLIW+SIMD架构原型[12]。VLIW对应控制通路中的并行性(指令级并行)，SIMD对应数据通路中的并行性(数据级并行)。

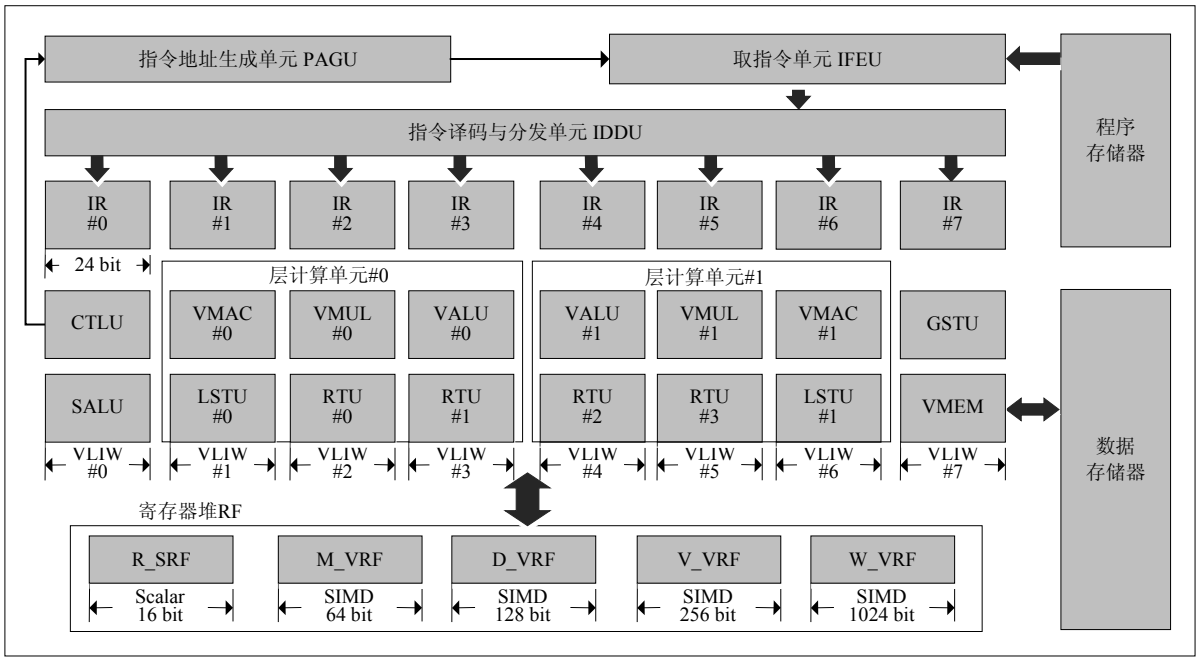

图5　基于矢量处理器的可编程并行MIMO检测器总体结构

所设计处理器的最基本结构可分为系统单元、功能单元、寄存器堆和存储系统。其中系统单元包括指令地址生成单元PAGU、取指令单元IFEU、指令译码与分发单元IDDU等。PAGU和IFEU用于生成正确的下一条指令地址并从程序寄存器中取出对应的VLIW指令；IDDU将上述指令译码并分发到对应的IR(指令寄存器)。存储系统用于存放代码和数据。图5中共有8个VLIW指令通路(VLIW#0~VLIW#7)，每个指令通路由对应的指令寄存器(IR#0~IR#7)与图中位于其正下方的功能单元组成，如VLIW#0由IR#0和控制单元CTLU以及标量运算单元SALU组成，以此类推。

3.2 SIMD优化及寄存器设计

采用矢量寄存器进行数据运算是矢量处理器的主要特点，同时也是SIMD特性的主要体现。合理的设计矢量寄存器的长度(即SIMD长度, 单位为数据单元的个数)关系到矢量处理器的资源和执行效率，而SIMD长度主要由目标应用中数据并行性决定。

表1统计了不同系统配置(天线数目和调制阶数)下使用PIPSD算法涉及到的主要矢量长度(输入矢量长度、星座矢量长度、幸存路径数和扩展路径数)。表中扩展系数矢量是在该配置下的典型选择，表格中采用了简化记法，省略号表示后面全为1，如第5行4×4 16QAM配置下的扩展系数向量实际上是[4,4,1,1,1,1,1,1]。

表2是对表1中出现的所有数据矢量长度进行统计，由统计结果可以看出长度分别为4,8,16,64的矢量的需要频次较高。因此本文在处理器中设计4种长度的矢量寄存器就可以精确满足绝大部分需求，其余长度则可以通过补零或者分段的方式实现。注意由于每个数用16 bit表示，寄存器堆中的实际比特长度要将SIMD长度再乘以16。表3是对不同的寄存器堆(包括标量寄存器堆)进行命名，图5中的寄存器堆中的命名是相对应的。

表1 不同配置下需要的SIMD长度

天线数目	调制阶数	输入矢量长度	星座矢量长度	扩展系数矢量	幸存路径数	扩展路径数
2×2	2/4	4	2	2,2,…	4	8
2×2	16	4	4	4,4,…	16	64
2×2	64	4	8	4,2,…	8	64
4×4	2/4	8	2	2,2,…	4	16
4×4	16	8	4	4,4,…	16	64
4×4	64	8	8	8,8,…	64	512
8×8	2/4	16	2	4,4,…	16	32
8×8	16	16	4	8,8,…	64	256
8×8	64	16	8	8,8,…	64	512

表2 SIMD长度统计结果

长度/个	2	4	8	16	32	64	>64
频次	3	8	8	7	1	6	3

表3 不同寄存器堆命名及对应长度

寄存器堆命名	R	M	D	V	W
SIMD长度/个	1	4	8	16	64
实际长度/bit	16	64	128	256	1 024

3.3 功能单元设计

功能单元是处理器进行运算和控制的功能实体，如图5所示分布在各个VLIW通路中，不同VLIW通路中的功能单元可以同时执行，从而实现指令级并行。

表4列举了主要功能单元的功能描述以及对应的指令。SALU(标量运算单元)主要完成标量加减、移位、逻辑运算等；CTLU(控制单元)主要完成循环控制及条件跳转等操作；VALU(矢量运算单元)、VMUL(矢量乘法单元)和VMAC(矢量乘累加单元)矢量的加减和乘法运算，对于每种长度的矢量的运算都有对应的指令；RTU(寄存器转换单元)主要是在5种不同长度的寄存器中进行转换，如将短矢量扩展为长矢量，或者从矢量中提取一个标量元素等；LSTU(局部排序单元)用于完成PIPSD算法中的层搜索过程中的局部排序；GSTU(全局排序单元)用于全局排序。

表4 主要功能单元描述

功能单元	功能描述	对应指令示例
SALU	标量运算	ADD_RR,SUB_RR等
CTLU	控制及跳转	RPT,JNE等
VALU	矢量加减	ADD_MM,SUB_RV等
VMUL	矢量乘法	MUL_RM,MUL_VV等
VMAC	矢量乘累加	MAC_RV,MAC_WW等
RTU	寄存器长度转换	EXP_MV, DI_to_R等
LSTU	局部排序	SORT_VM,SORT_WV等
GSTU	全局排序	SORT_MR,SORT_VR等
VMEM	矢量读写	LD,SD等

3.4 专用指令集设计

指令集的设计与功能单元的设计是相对应的，指令在硬件上是通过功能单元实现的，功能单元的使用在汇编程序中也就表现为对应的指令。根据图2算法实现结构的需求，结合图5的硬件结构(特别是寄存器堆结构)设计相应的指令。表5中列出了部分指令作为示例。

表5　专用指令集部分指令示例

指令名	指令语法	指令功能	单元
LI	LI R[dst], imm	标量赋值	SALU
JNE	JNE R[src]	条件跳转	CTLU
ADD_MM	ADD_MM M[src1],M[src2],M[dst]	M矢量相加	VALU
MUL_RV	MUL_R R[src1],V[src2],V[dst]	标量乘M矢量	VMUL
MAC_WW	MAC_WW W[src1],W[src2],W[dst]	W矢量乘累加	VMAC
EXP_VW	EXP_VW V[src],W[dst],flag	M扩成V矢量	RTU
SORT_WV	SORT W[src],V[dst1],V[dst2]	局部排序	LSTU
SORT_VR	SORT_VR V[src],R[dst1],R[dst2]	全局排序	GSTU
LD	LD R[src],imm4,D[dst]	读取D矢量	VMEM

3.5　软硬件协同VLIW优化

VLIW指令级并行是该处理器的重要特征，同时也是发挥PIPSD算法两层并行计算优势的重要保证。研究最优的VLIW配置方案，需要协同考虑硬件结构与目标算法的汇编应用程序。在设计VLIW多发射架构时，主要是将功能单元合理地分布于各个VLIW通路。为了进一步优化功能单元的分布，减少执行周期数，同时提高资源利用率，本文将从层间并行性支持、功能单元的使用频率以及功能单元间数据依赖关系几个方面综合考虑。

3.5.1　层间并行性支持

从图2的算法实现分析中可以看出，PIPSD算法的最主要计算量集中于树搜索部分，树搜索部分的最大特点就是相邻两层同时计算，为了充分发挥该并行特征，将层计算涉及到的主要功能模块(即VALU，VMUL，VMAC和LSTU)从逻辑上组成层计算单元，并在不同的VLIW通路中设置两份。其余的功能单元按照分类将其分布到不同的VLIW通路中，得到VLIW初步分布如图6所示。

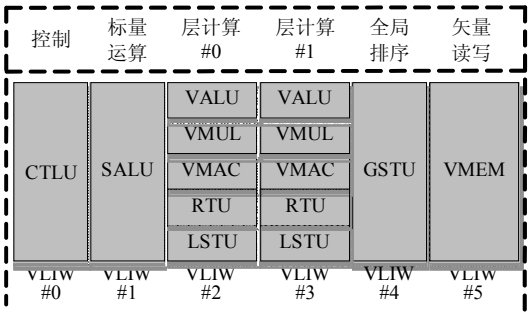

图6　VLIW初步分布

3.5.2　功能单元使用频率

使用频次高的功能单元可以在不同的VLIW通路中多设置，以便尽可能并行执行；而使用频次较低的功能单元则可以经适当组合放到一个VLIW通路中，以便提高资源利用效率。

各种配置模式下完整PIPSD汇编程序中各功能单元的使用频率的统计结果如图7所示。

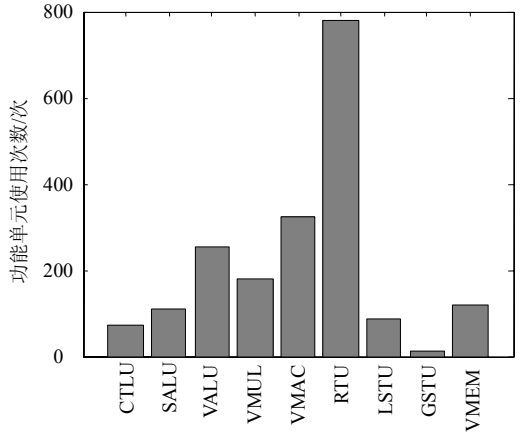

图7　各个功能单元的使用频率比较

从图7中可以看出使用频率最高的是寄存器长度转换单元RTU，这是由于5种不同长度的寄存器会频繁进行切换；此外矢量加减运算VALU、矢量乘法运算VMUL和矢量乘累加运算VMAC使用频率也较高，而其余各功能单元使用频率则相对较低。

汇编程序	功能模块	
;计算第6层		
LD R10,2,D2	VMEM	
DI_to_R D2[7],R2	RTU	D
MUL_RV R2,V15,V1	VMUL	
DI_to_R D2[8],R3	RTU	
…	…	
SUB_RV R2,V3,V4	VALU	D
EXP_VW V4,W1,0	RTU	
MUL_RV R3,V15,W1	VMUL	
…	…	
MAC_WW W1,W1,W2,W3	VMAC	D
SORT_WV W1,V1,V2	LSTU	

D：后一条指令的源操作数依赖于前指令目的操作数
ND：反之，即后源操作数不依赖于前目的操作数

图8　数据依赖关系分析示例

3.5.3　功能单元间数据依赖关系

数据依赖关系指的是在程序执行的过程中，后一条指令的源操作数是否依赖于前一条指令的目的操作数，分别记为D(依赖)关系和ND(不依赖)关系。如果是ND关系，则这两条指令可以并行执行，反之则后一条指令必须等待前一条指令执行完毕才能执行。尽量将不具有数据依赖关系的操作分布在不同的VLIW通路，以便并行执行；尽量将具有前后数据依赖关系的操作分布在相同的指令槽，以便减少因等待数据而造成该指令通路填充NOP(空指令)的情形。

图8是以4×4天线16QAM调制为例，从该配置下

PIPSD算法的汇编程序中摘取一个典型代码片段(第6层的计算和排序)分析相邻指令之间的数据依赖性。指令之间的数据依赖性实际上也就是功能单元之间的数据依赖性，如LSTU是依赖于VMAC的，因为程序中SORT_WV指令紧挨着VMAC指令并且对其运算结果进行局部排序。

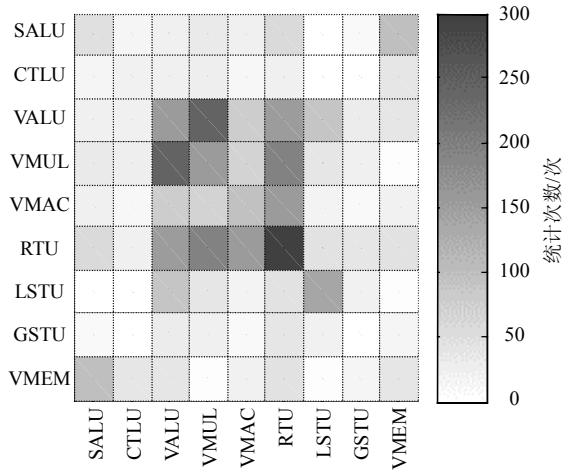

图9 统计ND(不依赖)关系

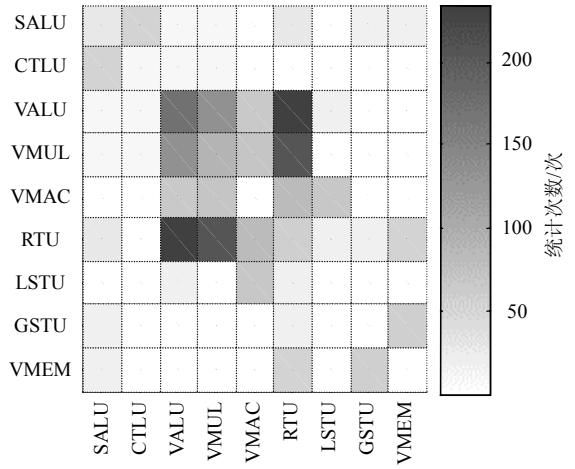

图10 统计D(依赖)关系

统计各种配置模式下完整PIPSD汇编程序中各个功能单元之间的ND关系和D关系，将得出的结果绘制成热图，分别如图9和图10所示。综合图7、图9和图10的统计结果，在图6的基础上进一步优化VLIW的分布：

1) 从图9可以发现RTU自身之间的ND关系是最强的(即并行性最强)，从图10看出RTU之间的D关系相对较弱(即数据依赖性较弱)，再结合图7中RTU使用频率最高的结论，将每个层计算单元中的RTU拆分成两份并分布到两个不同的VLIW通路中；

2) 从图9可以看出VALU、VMUL和VMAC之间的ND关系较强，从图10中看出VALU与RTU之间，以及VMUL与RTU之间D关系较强，因此将VALU、VMUL和VMAC分布在不同的VLIW通路，同时将VMUL与RTU，以及VALU与RTU分布在同一个VLIW通路。

3) 其余单元因为使用频率较低，在图9和图10中统计次数也较少，相对来看，GSTU与VMEM，以及SALU与CTLU的D关系较强，SALU与VMEM的ND关系较强，因此将SALU与CTLU安排在同一VLIW通路，将GSTU与VMEM安排在另一通路中。

综合上述分析，最终可得出优化之后的各VLIW通路中功能单元的分布情况如图5所示。

4 性能比较

首先将不同定点方案下设计的MIMO检测器的误码率与算法的浮点仿真结果进行比较，以验证该实现方案的正确性；然后从吞吐率和硬件资源消耗等角度，与近年来其他可编程MIMO检测器方案进行比较，以证明该实现方案的有效性。

4.1 定点误差比较

图11是以4×4天线16QAM调制为例，给出了不同定点长度(均为“1位符号位+小数部分”方案)下所设计检测器的误码率性能与浮点算法的比较结果。

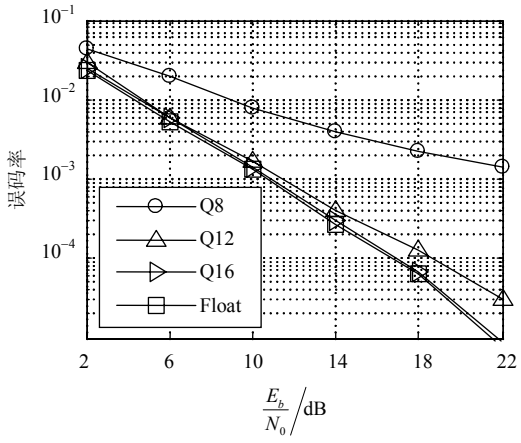

图11 定点误差比较

从图11中可以看出，采用8 bit以及12 bit定点时，均有较大的性能误差，采用16 bit定点方案时，与浮点情形的性能误差已经几乎可忽略不计，因此在本设计中采用16 bit定点方案能够保证译码的正确性。

4.2 实现性能比较

与传统的专用电路实现不同，本文设计的MIMO检测器是可编程的，并且能够适应多种不同的配置模式，因此在进行性能比较时可与多模可编

程的解决方案进行比较。表6为本文与EquiASIP[5]和FLEXDET[6]的各项性能比较，它们均采用ASIC工艺综合实现。

表6　多模可编程MIMO检测器性能比较

性能	EquiASIP[5]	FLEXDET[6]	本文
	MMSE-IC	MMSE	PIPSD
执行周期(2×2)	70	25	18
执行周期(4×4)	185	77	39
时钟频率/MHz	546	263	302
符号吞吐率(2×2)/ MSymb·s^{-1}	15.6	21.04	33.55
符号吞吐率(4×4)	11.8	13.66	30.97
电路面积/kGE	470	597	933
面积延时积(2×2)	30.13	28.37	27.81
面积延时积(4×4)	39.83	43.70	30.13

表6中符号吞吐率的计算如式(4)所示；电路面积是ASIC门级综合的等效门数。

$$\text{吞吐率}=\frac{\text{天线数目}\times\text{时钟频率}}{\text{执行周期数目}} \tag{4}$$

同时，为了将电路面积和符号吞吐率合成一个综合指标，本文引入面积A和延时T(符号吞吐率的倒数)的乘积(AT-Product)作为度量，其计算如式(5)所示，AT值越小，代表电路资源使用的效率越高。

$$AT=\frac{\text{电路面积}}{\text{符号吞吐率}} \tag{5}$$

观察表6可以发现，本文方案的符号吞吐率是其余方案的2～3倍，占用电路面积同时也大于后者，几种方案的面积时延积相近，相比之下本文方案仍略有优势。因此本文方案成功实现了以电路资源换取处理速率，达到了并行化的初衷。

综上所述，本文所设计实现方案不仅能够达到算法所设计的性能，并且能在不降低电路设计效率的前提下，以合理的硬件资源消耗为代价将吞吐率提高2～3倍。

5　结 束 语

本文从实现角度深入分析PIPSD检测算法的基础上，基于矢量处理器架构原型设计了可编程的并行MIMO检测器，该检测器不仅具有高效支持指令并行与数据并行的VLIW+SIMD结构，而且还根据算法特点设计了寄存器堆、功能单元以及相应指令，并根据程序特点优化VLIW分布，以最大限度减小运行时间增大译码吞吐率。性能比较结果表明所设计可编程并行MIMO检测器相比其他多模可编程MIMO检测器设计方案，能够在不同系统配置条件下保证算法译码性能的同时，以合理的硬件资源消耗为代价将译码吞吐率提高2～3倍。

参 考 文 献

[1] PAULRAJ A J, GORE D A. An overview of MIMO communications-a key to gigabit wireless[J]. Proc IEEE, 2004, 92(2):198-218.

[2] LI Qing-hua, LI Guang-jie, LEE W, et al. MIMO techniques in WiMAX and LTE: a feature overview[J]. IEEE Commun Mag, 2010, 48(5): 86-92.

[3] BURG A, BORGMANN M, WENK M, et al. VLSI implementation of MIMO detection using the sphere decoding algorithm[J]. IEEE J Solid-State Circuits, 2005, 40(7): 1566-1577.

[4] FASTHUBER R, CATTHOOR F, RAGHAVAN P, et al. Energy-efficient communication processors[M]. Berlin: Springer, 2013.

[5] JAFRI A R, KARAKOLAH D, BAGHDADI A, et al. ASIP-based flexible MMSE-IC linear equalizer for MIMO turbo-equalization applications[C]//Proceedings of the Conference on Design, Automation and Test in Europe. Nice: IEEE, 2009.

[6] CHEN Xiao-lin, MINWEGEN A, HASSAN Y, et al. FLEXDET: Flexible, efficient multi-mode MIMO detection using reconfigurable ASIP[C]//The 20th Annual International Symposium on Field-Programmable Custom Computing Machines. Toronto: IEEE, 2012.

[7] FAN A-dong, QIN Xiao-wei, DAI Xu-chu. A partial inter-layer parallel sphere decoder for multiple-input multiple-output systems[C]//The 6th Sixth International Conference on Wireless Communications and Signal Processing. Hefei: IEEE, 2014.

[8] LARSSON E G. MIMO detection methods: How they work [J]. IEEE Signal Process Mag, 2009, 26(3): 91-95.

[9] GUO Z, NILSSON P. Algorithm and implementation of the k-best sphere decoding for MIMO detection[J]. IEEE J Sel Areas Commun, 2006, 24(3): 491-503.

[10] AZZAM L, AYANOGLU E. Reduced complexity sphere decoding via a reordered lattice representation[J]. IEEE Trans Commun, 2009, 57(9): 2564-2569.

[11] JAFRI A R, BAGHDADI A, JEZEQUEL M. ASIP design and prototyping for wireless communication applications [M]. Rijeka: INTECH Open Access Publisher, 2011.

[12] 张建正，秦晓卫，周武旸. 面向LTE-A终端软基带的矢量处理器设计[J]. 无线通信技术, 2014, 03: 15-20.
ZHANG Jian-zheng, QIN Xiao-wei, ZHOU Wu-yang. Vector processor for soft baseband terminals[J]. Wireless Communication Technology, 2014, 03: 15-20.

编　辑　税　红