一种应用定制指令集可重构结构及FFT算法映射优化*

刘 磊,杨子煜,沈剑良,李思昆
(国防科技大学 计算机学院,湖南 长沙 410073)

摘 要:现代无线通信应用对FFT计算吞吐率与灵活性需求越来越高,针对传统方案实现FFT计算时难以兼顾性能与灵活性的问题,提出一种应用定制指令集可重构结构ASRA,实现了FFT算法在该结构上的映射优化。ASRA在静态多发射处理器内紧耦合应用定制的混合粒度可重构硬件作为扩展功能单元簇,通过运行时重构动态切换扩展指令集。ASRA采用多体便笺存储器、多端口便笺管理单元及可重构互连构成片上缓存系统,结合多体并行访问、循环级乒乓交替、读/写流水化等技术有效提高了访存带宽;静态多发射和运行时语境管理机制支持核心循环的硬件自动流水执行和软流水执行,开发了指令级、数据级和循环级等多层次并行性。实验结果表明,ASRA大幅提升了FFT计算吞吐率,且支持的FFT计算参数更加灵活,而增加的面积开销相对较小。

关键词:应用定制;运行时重构;傅里叶变换
中图分类号:TP368 **文献标志码**:A **文章编号**:1001-2486(2012)06-0039-07

An application specific instruction set reconfigurable architecture and the mapping of FFT on it

LIU Lei, *YANG Ziyu*, *SHEN Jianliang*, *LI Sikun*
(College of Computer, National University of Defense Technology, Changsha 410073, China)

Abstract: To meet the stringent requirements on both high-throughput and high-flexibility for FFT computation in modern wireless communication applications, an application specific reconfigurable architecture called ASRA is presented. ASRA is a VLIW-like static multi-issue processor with custom multi-grained reconfigurable fabric as extensible function units. The run-time context manager in ASRA offers multi-grained Custom Instructions selection and is bound to the appropriate reconfigurable fabric while considering run-time changing scenarios. Mapping of the FFT algorithm on ASRA is studied and optimized. ASRA employs on-chip scratchpad as fast local memory. The read/write operation and butterfly operation is pipelined to support hardware pipeline execution of a loop kernel. Experiment results show that ASRA achieves a high performance improvement and a good trade-off between area and performance.

Key words: application specific instruction-set; run-time reconfigurable; fast Fourier transform

快速傅里叶变换(FFT, Fast Fourier Transform)是正交频分复用(OFDM, Orthogonal Frequency Division Multiplexing)技术的核心算法。OFDM广泛用于通信领域,例如数字视频广播、移动通信、无线局域网、无线城域网等。迅速发展的通信应用需要更高吞吐率和更好灵活性的FFT计算,以适应多通信协议和多操作模式融合的应用环境。

现有FFT实现算法可归纳为三类:一类基于传统Cooley-Turkey FFT算法,在数据通路上放置大量计算资源,利用流水线并行性提高计算性能。该类算法中每级蝶形运算都依赖前级的结果数据,需要频繁访问主存,访存带宽限制了性能的提升。第二类采用非2基FFT分解方法,例如高基FFT和混合基FFT,减少了乘法和加法操作数量,但仍然受限于访存带宽。第三类致力于缓解访存带宽限制,Baas[1]提出了基于缓存的Cached FFT算法,通过重新组织蝶形运算的顺序,缓存中的数据可以供给多级蝶形运算,减少访问主存的次数。Guan[2-3]进一步改进了Cached FFT蝶组内的蝶形运算组织结构。Baas[1]和Guan[2-3]都使用寄存器文件作为缓存,但寄存器文件大小限制了可计算的FFT点数,例如32个寄存器支持至多1024点Cached FFT。

本文提出一种应用定制指令集可重构结构ASRA(Application Specific Reconfigurable Architecture)。面向FFT计算特征,研究实现了高吞吐率、连续

* **收稿日期**:2012-07-02
基金项目:国家自然科学基金资助项目(61076020,61133007)
作者简介:刘磊(1984—),男,山西襄汾人,博士研究生,E-mail: alwater.liulei@gmail.com;
李思昆(通信作者),男,教授,博士生导师,E-mail: lisikun@263.net.cn

流、多种长度 FFT 计算在 ASRA 上的映射与优化。基于 ASRA 实现的 FFT 计算方案主要有如下特点:(1)针对现有研究中遇到的访存瓶颈问题,采用便笺存储器、专用便笺管理单元和可重构互连构成片上缓存系统,通过多体并行访问、循环级乒乓交替、读/写流水化等技术有效地提高数据带宽;(2)数据读/写操作与蝶形运算操作组成流水线,访问数据与蝶形计算重叠执行;(3)运行时语境管理机制,支持核心循环的硬件自动流水执行;(4)利用静态多发射和指令调度技术,实现循环的软流水执行。

1 应用定制的可重构结构:ASRA

专用指令集处理器 ASIP(Application Specific Instruction set Processor)兼具软件可编程的灵活性和专用硬件加速计算的高效性,适于实现高吞吐率、高灵活性 FFT 计算。ASIP 在基本处理器中加入定制功能单元,对目标应用的计算核心进行加速。根据定制功能单元的硬件实现方式,ASIP 分两类:一类采用硬连线,例如 Tensilica 公司的 Xtensa 处理器[2-3];另一类采用可重构结构,例如本文的 ASRA。

1.1 ASRA 结构框图

ASRA 结构如图 1 所示,在类似超长指令字(VLIW)的静态多发射指令流水线中紧密耦合了一个可重构阵列作为扩展功能部件,实现定制指令功能。ASRA 结构中有两种数据通路:基本数据通路(basic data path)和定制数据通路(custom data path)。应用中的计算核心编译映射到定制数据通路上执行;其他部分映射到基本数据通路上执行。两种数据通路有各自的寄存器文件,执行数据传输指令在寄存器文件之间交换数据,实现数据通信。

1.2 应用定制的数据通路

定制功能单元 CFUs(Custom Function Units)、可重构互连、C 组寄存器文件(C register file)、便笺管理单元(scratchpad manage unit)、数据存储器(data memory)、配置存储器(configuration memory)、语境管理器(context manager)组成了应用定制的数据通路。

数据存储器和配置存储器都是片上便笺式存储器,由软件管理访存操作,通过直接存储访问方式(DMA, Direct Memory Access)或者执行 load/store 指令与主存交换数据。便笺管理单元具有多套读/写端口,对组成数据存储器的多个存储体使用各自独立的读/写地址及控制信号,实现多路访存操作并行执行。它可支持循环流水中的自动读/写操作数功能。

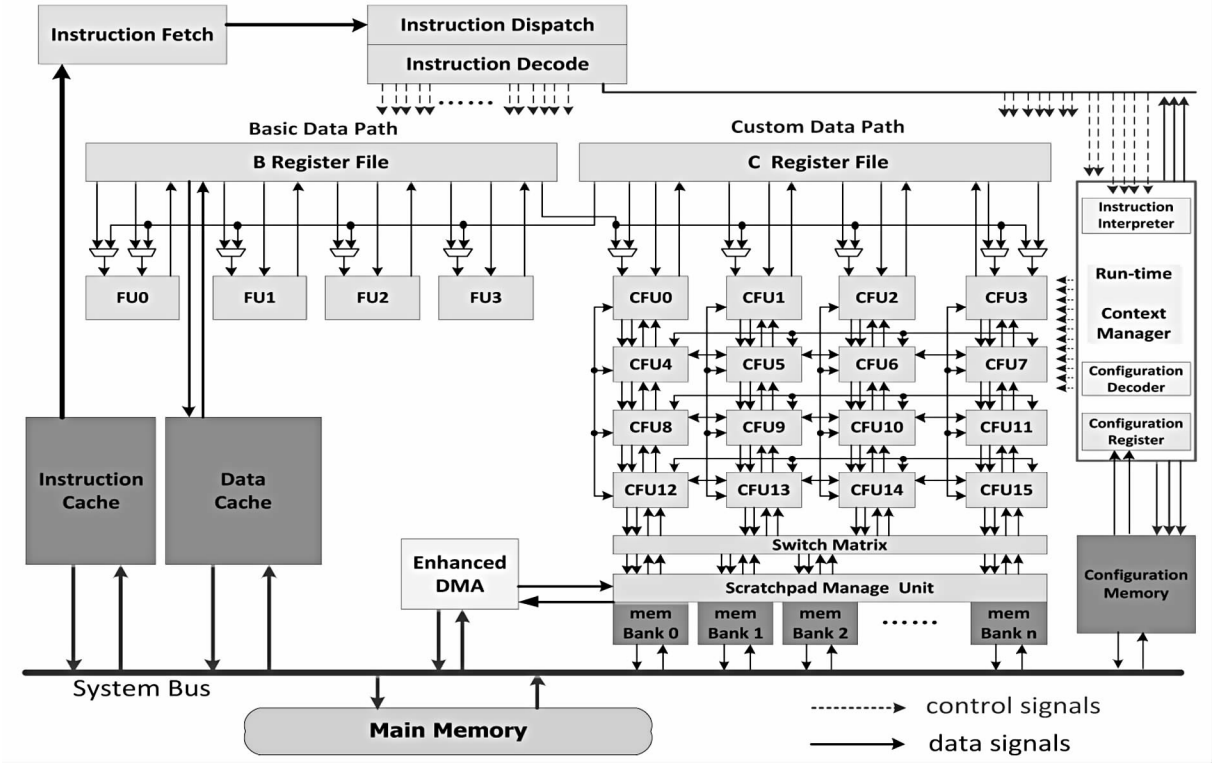

图 1 ASRA 结构框架

Fig. 1 ASRA Architecture

CFU 是根据应用特征定制的可重构单元。功能单元之间有多种可重构互连方式,例如二维 MESH,相邻连接,层次化互连等。

传统可重构阵列一般采取细粒度或粗粒度的同构组织。ASRA 中功能单元并不局限于传统的单一粒度和同构组织,而是根据应用的计算特征定制设计,支持混合粒度和异构组织,有利于提高硬件资源利用效率,减少重构过程的时间和存储开销。CFU 支持由多个基本操作组合而成的宏操作,其中基本操作划分到多级流水段,形成多周期操作流水线。

通过运行时动态切换配置,CFU 可组织成多种执行方式,充分利用指令级、数据级和循环级等多层次并行性。将多个 CFU 并联可组成单指令流多数据流(SIMD)或者超长指令字 (VLIW)方式;将多个 CFU 串联可组成多操作流水线。

1.3　运行时语境管理

可重构阵列运行时所需控制信息构成了定制数据通路的运行时语境(run-time context),主要包括:功能单元的功能与互连、指令执行状态等。对于一个定制功能,如果所需控制信息较少,则控制信息全部编码在指令中;如果所需控制信息超出指令编码长度,则将部分控制信息编码在配置中。由指令(instruction)与配置(configuration)共同提供控制信息,便于实现固定长度指令编码,简化指令译码逻辑;同时可解决指令编码空间受限的问题,支持更大规模的定制功能。

定制指令执行过程中,语境管理器负责发出控制信号,并跟踪记录指令执行状态。语境管理器(context manager)内部含有配置寄存器、配置译码器、指令解释器和有限状态机。首先由指令译码器依据指令类型,把发射槽内的指令送给指令解释器,把需要的配置从配置存储器加载到配置寄存器。然后,在语境管理器内部,指令解释器对指令进行译码(指令语义),配置译码器对配置进行译码(配置语义),两者发出的控制信号共同组成完整的控制信息。

配置寄存器中保存的配置信息将会长时间有效,直到下一次修改寄存器。配置预取指令用于加快配置加载过程。

1.4　ASRA 模板的设计空间

ASRA 是一个灵活的应用定制指令集可重构体系结构模板。机器描述语言 HMDES[5] 用于描述各项体系结构参数,基于 Trimaran[5] 可重定向编译框架开发的工具用于辅助探索设计空间。可探索的体系结构参数包括:数据 Cache 大小、寄存器文件大小、功能单元的功能与数量、可重构互连方式、便笺存储器大小、指令与配置编码等。

ASRA 结构模板与 ADRES[4] 相似,都是在静态多发射处理器内紧密耦合可重构硬件作为扩展的计算加速部件。不同之处主要有两方面。首先,在基本处理器与可重构硬件的接口方面:(1) ADRES 中 VLIW 模式与阵列模式只能串行执行;ASRA 中基本数据通路与定制数据通路可并行执行,便于利用更多并行性。(2) ADRES 采用集中式寄存器文件;ASRA 采用分簇式寄存器文件,减少了寄存器文件端口数量约束。其次,在可重构硬件的组织控制方面:(1) ADRES 使用 VLIW 控制器控制可重构阵列;ASRA 使用语境管理器控制可重构硬件,除支持循环流水执行外,还可支持其他类型并行操作;(2) ADRES 可重构阵列中功能单元属于传统粗粒度类型;ASRA 的可重构功能单元根据应用特征定制设计,既可支持由多个基本操作组合而成的宏操作,也可支持字节级/位级的细操作,属于混合粒度,有利于提高硬件资源利用率。

2　FFT 计算特征分析

FFT 是实现离散傅里叶变换(DFT)的快速算法。N 点复数序列$\{x(n)\}$的 DFT 定义为:

$$Y(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk},\ k,\ n \in [0,\ N-1] \quad (1)$$

其中 $W_N^{nk} = (W_N)^{nk} = e^{-j\frac{2pnk}{N}}$ 是旋转因子。FFT 有多种实现算法,对于给定点数,高基数 FFT 比低基数 FFT 所需运算操作少。但高基数算法可实现的 FFT 点数范围较窄,基 r 算法只能实现 r^n 点 FFT。基 2 FFT 由于算法结构规整且支持的 FFT 点数范围最广而大量应用。Cooley-Turkey FFT 是时分 FFT 算法,N 点基 r FFT 分成 $\log_r N$ 级,每级包含 $N/2$ 次蝶形运算。每级蝶形运算都依赖前级结果数据,级数较大时需要频繁访问主存。Cached FFT 算法通过优化蝶形运算顺序,开发缓存利用率,减少访问主存。

蝶形运算是 FFT 的核心。图 2 显示了基 2 时分蝶形运算公式,A,C,X,Y,W 都是复数,相应的加、减和乘法都是复数运算。图 3 显示了基 2 FFT 的两种组织结构。32 点 Cooley-Turkey FFT 计算过程分成 5 级(stage0 ~ stage4),每级包含 16 个蝶形运算。Cached FFT 计算过程分成 2 期(epoch),每期分出 4 组(group),每组分成 3 遍(pass)。每组蝶形运算所需数据全部放在缓存。

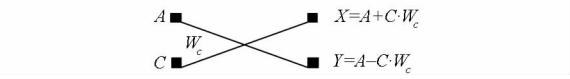

图 2　基 2 时分蝶形单元

Fig. 2　Radix-2 divided in time butterfly computation

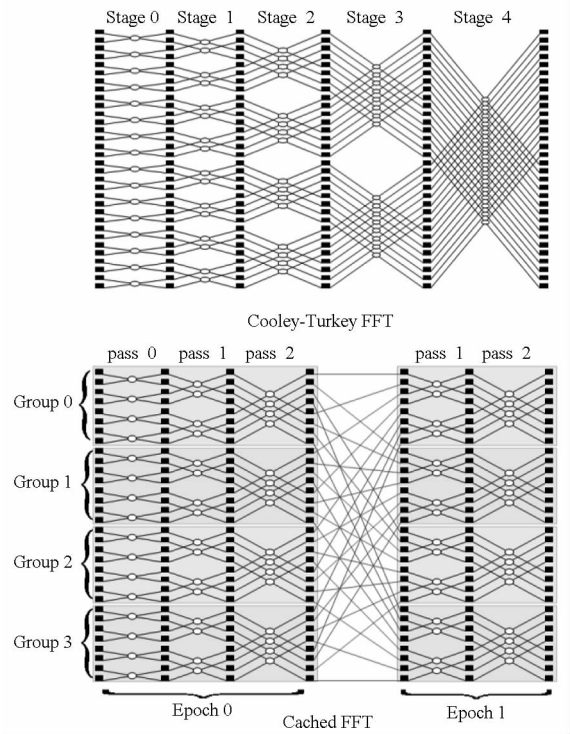

图 3　32 点基 2 FFT 的两种组织结构

Fig. 3　Structure for 32-point radix-2 FFT

3　数据通路设计与算法映射优化

FFT 计算性能受算法、软件和硬件结构等多方面因素影响，不同实现方法所获得的性能、面积、功耗以及灵活性不同。由于基 2 FFT 算法应用最广泛，所以本文针对基 2 FFT 算法进行定制。

3.1　功能单元

为了对 FFT 计算的关键核心进行加速，CFU 的功能设计为实现一个基 2 蝶形运算。

图 4 表示一个蝶形运算在 CFU 上的映射。该 CFU 有 2 个 32 位输入端口，2 个 32 位输出端口。32 位数据的高 16 位表示复数实部，低 16 位表示复数虚部。两个 16 位数乘法运算得到 32 位数据，舍掉低 16 位后作为有效结果。CFU 内包含 4 个实数乘法器和 8 个实数加/减法器，基本算术操作分布在 3 段流水线中，运算的中间结果保存在流水线寄存器中。完成一次蝶形运算需要 3 个时钟周期。CFU 的输入操作数和结果数据都存放在片上数据存储器中。便笺管理单元从数据存储器读取数据或写回数据需要 1 个时钟周期。便

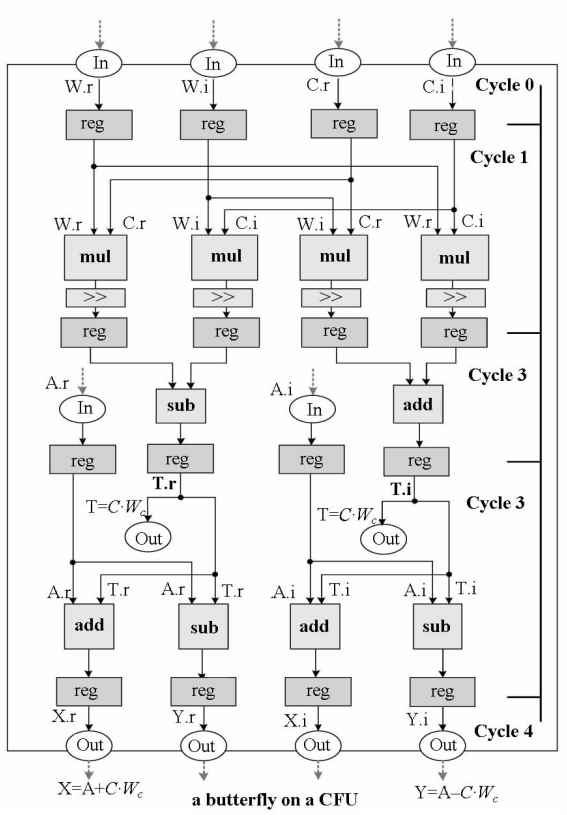

图 4　基 2 蝶形运算在 CFU 上的映射

Fig. 4　Mapping of Radix-2 butterfly on a CFU

笺管理单元的读操作和写操作与 CFU 内部运算操作可连接组成 5 段流水线。

3.2　定制指令

为了支持 FFT 计算中循环流水执行，引入 4 条定制指令：DMA_DO，BUT2_CONF，BUT2_DO，BUT2_LOOP。

DMA_DO 指令用于启动 DMA 单元，在主存和数据存储器之间传输数据。该指令所需信息有：数据在主存中的起始地址、在数据存储器中的起始地址，传输数据块的大小等。BUT2_CONF 指令用于加载配置信息，配置 CFU 的功能与互连、操作数在数据存储器中的访问地址生成逻辑等。BUT2_DO 指令用于执行一次蝶形运算。BUT2_LOOP 指令用于启动一组蝶形运算的循环计算过程。执行 BUT2_LOOP 指令后，定制数据通路处于循环自动流水运行状态，运行过程中所需控制信号由运行时语境管理器依据配置信息和当前状态发出。

3.3　片上存储器管理与优化

FFT 计算过程的操作数与中间结果都存放在片上数据存储器。读/写数据时的带宽是影响蝶形运算流水线性能的关键因素。图 5 表示基于 1 个 CFU 的蝶形运算流水线。当流水线填满后，

CFU 每周期输入 3 个数据并且写回 2 个数据。为此需要便笺管理单元每周期并行读出 3 个 32 位数据,写入 2 个 32 位数据。基于多个 CFU 的并行运算流水线,所需数据带宽成相应倍数增加。

Cycles:	Cycle 0	Cycle 1	Cycle 2	Cycle 3	Cycle 4	Cycle 5	Cycle6
bfly 0	R:W0,C0	mul	R:A0	add/sub	W:X0,Y0		
bfly 1		R:W1,C1	mul	R:A1	add/sub	W:X1,Y1	
bfly 2			R:W2,C2	mul	R:A2	add/sub	W:X2,Y2
bfly 3				R:W3,C3	mul	R:A3	add/sub
bfly 4					R:W4,C4	mul	R:A4
bfly 5	R=read,W=write					R:W5,C5	mul
bfly 6							R:W6,C6

图 5　定制数据通路上的蝶形运算流水线

Fig. 5　Butterfly operation pipeline on custom data path

数据带宽需求决定了片上存储器的设计决策。由于双端口 SRAM 开销过大,本文选用单端口 SRAM 作存储体。对于一条 CFU 运算流水线,每次蝶形运算中的 5 个数据分别存放在不同的存储体(记作 memA,memC,memX,memY,memW),每个时钟周期对各个存储体的访问操作至多有一次,并且对多存储体使用各自独立的数据总线、地址总线和控制信号完成读/写操作,从而实现多操作数并行读/写功能。

利用多个存储体实现多操作数并行读/写,需要保证每次并行访问的数据位于不同的存储体。针对 FFT 计算过程的蝶形运算分级特征,通过合理分配数据在存储体间的存放顺序可满足上述要求。例如,通过 DMA 加载初始数据时按照地址奇偶性组织数据存放顺序。相邻两级蝶形运算过程之间采用乒乓交替方式访问数据。第 i 级蝶形运算的源操作数取自{memA,memC},结果写入{memX,memY};则第 i + 1 级的源操作数从{memX,memY}取出,结果写入{memA,memC}。

蝶形运算具有原位操作特征,但若按原位地址写回结果,后级运算所需操作数将可能位于相同存储体内,并行访问时将会发生冲突。采用合理的软件调度方法避免冲突:首先,组织蝶形运算顺序时,采用蝶群间交叉调度方式;其次,写回结果时对地址重排序,依据当前蝶形运算所处的蝶级编号、蝶群编号、群内编号来计算新地址。

3.4　算法实现

各种基 2 时分 FFT 算法都可映射到定制数据通路上执行。Cooley-Turkey FFT 的循环体执行遍数较多,适于 BUT2_LOOP 指令硬件流水执行。Cached FFT 的数据局部性较好,适于 BUT2_DO 指令结合静态多发射和软件调度技术实现软流水执行。

```
//Cached FFT 在 ASRA 上的计算流程伪码
//输入:N 个 32 位复数点存放在主存中
1. DMA_DO:load coefficients;
2. BUT2_CONF: configure the datapath
3. for( e =0; e < E; e + + ) //epoch
4. {
5.      DMA_DO:transfer data to scratchpad;
6.      for( g =0; g < G; g + + ) //group
7.      {
8.          for( p =0; p < P; p + + )//pass
9.          {
10.             BUT2_DO: perform a butterfly
11.         }
12.     }
13.     DMA_DO:transfer data to main memory;
14. }
// 输出:运算结果保存在主存中
```

4　实验与分析

4.1　实验配置

基于 SystemC 语言实现了时钟周期精确的 ASRA 原型系统,进行指令集模拟验证,快速评估系统性能。手工转换为 Verilog 硬件模型后,使用 Synopsys Design Compiler 工具在 TMSC 130nm 典型工艺库上做逻辑综合,获取各项硬件参数。

表 1　ASRA 实现方法对比

Tab. 1　Various implementation of FFT on ASRA

实现方案	CFU 数	存储体数量	软件中主要使用的定制指令	性能加速比
C1N	1	5	–	1
C1H	1	5	BUT2_DO	1.43
C1S	1	5	BUT2_LOOP	1.41
C2H	2	10	BUT2_DO	2.13
C2S	2	10	BUT2_LOOP	2.05
C4S	4	10	BUT2_LOOP	2.76
C4HS	4	10	BUT2_LOOP&BUT2_DO	3.84

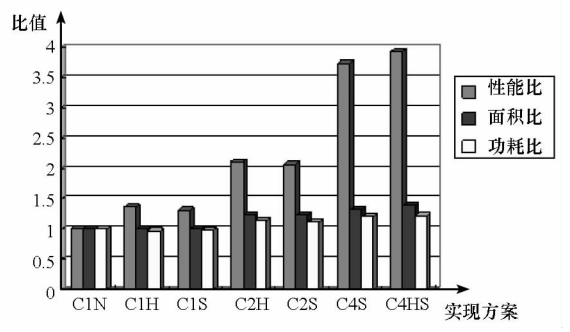

图 6　ASRA 实现方法对比

Fig. 6　Various implementation of FFT on ASRA

在功能单元数量、便笺存储器大小、软件实现方法等方面进行了设计空间探索，主要结果如表 1 和图 6 所示。CFU 数量范围取[1,4]；每个存储体大小为 1KB(256 × 32B)，数量范围取[1,10]；软件映射方法有三种：基于 BUT2_DO 指令的多发射软流水实现、基于 BUT2_LOOP 指令的硬件自动流水实现、混合使用 BUT2_DO 和 BUT2_LOOP 指令并行实现。

以 1 个 CFU 和 5 个存储体且仅使用基本指令完成 1024 点 FFT 计算时(C1N 方案)的实验数据作为参照系，对各种实现方法的实验数据进行归一化处理。从表 1 看到，C1H 和 C1S 方案分别获得了 43% 和 41% 加速比，表明定制指令有效提升了计算性能。C2H 和 C2S 方案将两条 CFU 流水线并行计算，获得了很好性能提升，加速比分别为 113% 和 105%。当 CFU 增加至 4 个后，由于便笺存储器带宽不足以支持 4 路流水线并行执行，导致 C4S 方案未能随着 CFU 数量增加而相应大幅提升性能。C4HS 方案中 2 个 CFU 以便笺作缓存，以硬件流水方式运行；另 2 个 CFU 以寄存器文件作缓存，以多发射软流水方式运行；利用 BUT2_LOOP 和 BUT2_DO 指令并行执行，进一步提升了性能，加速比达到 284%。与参考方案 C1N 相比，C4HS 面积增加约 39.7%，功耗增加约 30.1%，以较少的开销换取了大幅性能提升。

4.2 结果分析

C4HS 方案计算各种长度 FFT 时的时钟周期数和吞吐率(频率为 106MHz)列于表 2。

表 2 C4HS 方案实现各种长度 FFT 计算

Tab. 2 C4HS implementation for various-length FFT

FFT 点数	周期数(Cycles)	吞吐率(MSample/s)
128	284	47.7
256	568	47.7
512	1188	45.6
1024	2496	43.4
2048	6192	35.1
4096	25474	17.1
8192	53762	16.2

从表 2 看到，系统运行于 106MHz 时实现 1024 点 FFT 计算达到 43.4MS/s 吞吐率，满足 UWB-OFDM 无线通信规范要求。

以 C4HS 方案的实验数据与相关研究进行比较，数据列于表 3。相关的 FFT 实现方案主要有四种：FPGA，ASIC，DSP，ASIP。

FPGA 或 ASIC 实现 FFT 处理的方案，不考虑可编程性，一般采用延时反馈结构(例如 SDF，DDF，MDF)，利用多个蝶形单元、存储器及延时控制部件实现全流水化运行，目标是追求吞吐率最大化。与陆波[11] FPGA 方案相比，在 106MHz 频率计算 256 点 FFT，本文的吞吐率仅差 10.17%，但本文方案灵活性很好，可支持多种长度 FFT 计算。

DSP 和 ASIP 都属于存储程序体系结构，在可编程的灵活性基础上，追求 FFT 计算的高吞吐率和高性能。本文与 Guan[3] 相似，都是定制 4 个基 2 蝶形单元用于 FFT 计算加速，但本文的可重构定制数据通路支持更多并行性，片上便笺缓存系统支持更大访存带宽，显著提升了 FFT 计算吞吐率和灵活性。

在数据带宽方面，Guan[3] 采用寄存器文件作 Cached FFT 算法缓存，每次循环开始前和结束后都要执行 Load/Store 指令与主存交换数据，访存操作花费了大量时间，且寄存器大小限制了 FFT 点数范围。ASRA 仅在任务初始时用 DMA 加载

表 3 各种 FFT 处理器实现方案的比较

Tab. 3 Various implementations of FFT processing

比较对象	实现类型	工艺(nm)	计算范围(点数)	字宽	性能(1024 点)	面积	频率(MHz)	电压,功耗
TI[7]	DSP,R^2 – CFFT	–	–	–	77.55μs	–	320	–
Tang[9]	ASIC,R^XMDF	180	64 – 1024	10	512:2.4GS/s	3.2mm^2	300	1.8V,507mW
Chen[10]	ASIC,$R^{2/4}$DDF	180	128 – 1024	13	61μs	1.47mm^2	51	33.3mW
陆波[11]	FPGA,R^4SDF	–	256	16	256:53.1MS/s	3416slices	106	–
Jacobson[8]	ASIP,$R^{2/4}$ – CFFT	65	– 4096	–	3.1μs	–	866	1.3V,35mW
Hassan[6]	ASIP,R^2 – CFFT	130	– 4096	13	42.2μs	–	100	1.08V,25mW
Guan[3]	ASIP,R^2 – CFFT	130	16 – 1024	16	14.1μs	147 KGate	320	60.7mW
本文	ASIP,R^2 – CFFT	130	16 – 8192	16	7.3μs	183 KGate	320	1.8V,65.2mW
备注	R^2:Radix – 2; CFFT: Cached FFT; MDF:Multipath-Delay-Feedback; DDF: Dual-Delay-Feedback;SDF: Single-Delay-Feedback;							

部分数据,当蝶形运算流水线填满之后,读/写数据与蝶形计算并行执行,访存时间被隐藏;且循环核心在可重构通路上以自动流水方式执行,提高了计算吞吐率,性能提升了约 48.2%。

在重构开销方面,与 ADRES[4] 等通用可重构单元相比,ASRA 的功能单元是基于资源共享策略而定制的有限重构单元,所需配置信息大幅减少,重构的存储开销和时间开销都很低,切换一条定制指令所花费的重构时间约有几个到十几个时钟周期。实现重构功能的底层电路是多路选择器,其延时相当于 2 ~ 4 级门延时。有限重构所需多路选择器数量较少,相应的延时和面积开销也较少。与 Guan[3] 相比,本文的面积增加约 24.5%,主要原因是增加了片上存储器。本文工作以少量面积开销换取了大幅提升性能,在性能与面积权衡中取得了较好效果。

5　结论

本文提出应用定制指令集可重构结构 ASRA,针对 FFT 算法特征,定制了高效数据通路,支持高吞吐率、连续流、多种长度 FFT 处理。ASRA 采用片上便笺存储器作缓存,由专用便笺管理单元负责访问数据,采用多体并行访问、循环级乒乓交替、读/写流水化等技术提高了数据访问带宽。ASRA 的静态多发射和运行时语境管理机制,实现了 FFT 循环核心的硬件自动流水和软流水两种方式执行。实验结果表明该方案以增加少量面积开销的代价,取得了良好的 FFT 算法加速性能和高度灵活性。

参考文献(References)

[1] Baas B M. An approach to low power, high performance, fast fourier transform processor design [D]. Stanford University, 1999.

[2] Guan X, Lin H, Fei Y S. Design of an application-specific instruction set processor for high-throughput and scalable FFT[C]// IEEE International Symposium on Circuits and Systems, Taipei, 2009: 2513 - 2516.

[3] Guan X, Fei Y S, Lin H. Hierarchical design of an application-specific instruction set processor for high-throughput and scalable FFT processing[J]. IEEE Transactions on VLSI Systems, 2012, 20(3): 551 - 563.

[4] Bouwens F, Berekovic M, Kanstein A, et al. Architecture exploration of the ADRES coarse-grained reconfigurable array[J]. Springer Reconfigurable Computing: Architectures, Tools and Applications, 2007(1): 1 - 13.

[5] Chakrapani L, Gyllenhaal J, Scott A, et al. Trimaran: an infrastructure for research in instruction-level parallelism[J]. Languages and Compilers for High Performance Computing, 2005(1): 922 - 932.

[6] Hassan H M, Mohammed F, Shalash A. Implementation of a reconfigurable ASIP for high throughput low power DFT/DCT/FIR engine[J]. Journal on Embedded Systems, 2012(1): 1 - 18.

[7] Texas Instruments. TMS320C6713 floating-point digital signal processor[M]. Dallas: 2005.

[8] Jacobson A T, Truong D N, Baas B M. The design of a reconfigurable continuous-flow mixed-radix FFT processor[C]// IEEE International Symposium on Circuits and Systems. Taipei, 2009: 1133 - 1136.

[9] Tang S N, Liao C H, Chang T Y. An area-and energy-efficient multimode FFT processor for WPAN/WLAN/WMAN systems[J]. IEEE Journal of Solid-State Circuits, 2012(7): 1419 - 1435.

[10] Chen C M, Hung C C, Huang Y H. An energy-efficient partial FFT processor for the OFDMA communication system [J]. IEEE Trans. Circuits Syst. II: Express Briefs, 2010, 57(2): 136 - 140.

[11] 陆波,许炜阳,胡星波. 高吞吐率可配置 FFT 处理器 IP 核的设计与 VLSI 实现[J]. 复旦学报, 2010(2): 150 - 157.
LU Bo, XU Weiyang, HU Xingbo. Design and VLSI implementation of IP core of high-throughput configurable FFT processor[J]. Journal of Fudan University, 2010(2): 150 - 157. (in Chinese)