

UWB_OFDM 系统中 128 点 FFT/IFFT 的 IP 核的设计

施隆照, 高峰

(福州大学 物理与信息工程学院, 福建 福州 350002)

摘 要: 提出了两种针对超宽带(UWB)无线通信系统的 128 点快速傅里叶变换(FFT) IP 核的设计, 分别采用了 288 和 448 两种混合基结构. 从算法、结构上对其优点和缺点作了分析. 经过研究发现, 448 结构以增加面积为代价, 降低了工作频率, 在功耗方面有优势. 而 288 结构在面积方面有优势. 两个算法都可以在 312 ns 内完成运算, 都符合超宽带无线通信系统中对 FFT 的要求. 两种算法都有规则的数据流, 且控制器结构简单.

关键词: 超宽带; 快速傅里叶变换; 面积; 功耗

中图分类号:

文献标识码: A

文章编号: 1000-7180(2013)08-0040-04

Design of the 128-pt FFT/IFFT IP Core for UWB_OFDM System

SHI Long-zhao, GAO Feng

(College of Physics and Information Engineering, Fuzhou University, Fuzhou 350002, China)

Abstract: Two 128-pt FFT/IFFT IP core architectures for UWB_OFDM system are proposed in the paper, where the mixed-radix structures of 288 and 448 are used. The relative merits and demerits of the architectures have been analyzed in arithmetic and structure. Through the research, the 448 structure is better from the power perspective by giving a reduction to the working frequency, however the area is increased. On the other hand, the other one is better in the perspective of area. Both architectures are completed in 312 ns, that meet the requirement for fft in the UWB_OFDM system. They both have regular data flow, and the structures of their controllers are simple.

Key words: UWB; FFT; area; power

1 引言

UWB(超宽带)通信已成为无线通信领域关注的热点之一. 这种通信技术的发展能够满足未来无线通信的要求, 具有高空间频谱效率、高测距精度、低截获概率、抗多径衰落、不干扰现有通信系统、低功耗、低成本等诸多优点, 已成为中短距无线网络理想的传输/接入技术之一^[1]. 而基于 OFDM 技术的 UWB 系统具有各子载波之间相互正交, 不存在小区内干扰, 系统容量大等特点, 已成为其中的研究热点. 混合基是最常用的设计方法. 文献[2]中采用了 844 混合基结构, 数据读取采用 8 通道的形式, 用了 75 个时钟完成 128 点 FFT 计算. 该结构需要较高的功耗, 因为它时钟数多, 必须跑在较高的频率来满

足要求. 文献[3]中对比了不同数据通道数下的功耗情况, 并给出了两种 16 通道的设计结果. 一种采用 844 混合基结构, 完成计算需要 37 时钟, 功耗为 198.9 mW. 另一种采用 828 混合基结构, 所需功耗为 121 mW, 时钟数为 22. 这两种结构都需要缓冲器来存储计算过程的中间数据, 并存在整序问题, 不利于用流水线方式进行处理.

本文主要是研究应用于 UWB 系统中的 128 点 FFT 的实现方案. 提出 288 与 448 两种混合基结构, 分别采用 8 通道和 16 通道的设计方法. 采用串、并行联合处理的设计方案, 在满足系统要求的同时, 拥有流畅的数据流, 能够充分发挥流水线运算单元的吞吐能力, 并且在功耗和面积上都有一定的改良.

2 算法推导

128 点离散傅叶变换(DFT)的公式为: $X(k) = \sum_{n=0}^{127} x(n) * W_{128}^{n*k}$, 其中, W_{128}^{n*k} 是旋转因子, $n=0 \dots 127, k=0 \dots 127$.

首先对其进行 $16 * 8$ 分解, 即令 $k=16 * k_1 + k_0, n=8 * n_1 + n_0$, 其中 $n_0=0 \dots 7, n_1=0 \dots 15, k_1=0 \dots 7, k_0=0 \dots 15$. 则

$$X(k) = X(16 * k_1 + k_0) = \sum_{n_0=0}^7 \sum_{n_1=0}^{15} x(8 * n_1 + n_0) * W_{128}^{(8 * n_1 + n_0) * (16 * k_1 + k_0)} \quad (1)$$

16 点 FFT 采用两种方式进行分解, Arch1 采用 $2 * 8$ 的结构, Arch2 采用 $4 * 4$ 的结构. 16 点 FFT 的公式:

$$U(k_0) = \sum_{n_1=0}^{15} x(n_1, n_0) * W_{16}^{n_1 * k_0}$$

令 $n_1=8 * n_{11} + n_{10}, k_0=2 * k_{01} + k_{00}$, 其中 $n_{11}=0 \dots 1, n_{10}=0 \dots 7, k_{01}=0 \dots 7, k_{00}=0 \dots 1$, 整理后得到:

$$U(k_0) = U(k_{01}, k_{00}) = \sum_{n_{10}=0}^7 \sum_{n_{11}=0}^1 x(n_{11}, n_{10}, n_0) * W_{16}^{(8 * n_{11} + n_{10}) * (2 * k_{01} + k_{00})} = \sum_{n_{10}=0}^7 \left(\sum_{n_{11}=0}^1 x(n_{11}, n_{10}, n_0) * W_{2^{11}}^{n_{11} * k_{00}} * W_{16}^{n_{10} * k_{00}} * W_{8^{10}}^{n_{10} * k_{01}} \right) * W_{16}^{n_{10} * k_{01}} \quad (2)$$

将式(2)代入式(1)并整理得

$$X(k) = X(16 * k_1 + 2 * k_{01} + k_{00}) = \sum_{n_0=0}^7 \left\{ \sum_{n_{10}=0}^7 \left[\left(\sum_{n_{11}=0}^1 x(n_{11}, n_{10}, n_0) * W_{2^{11}}^{n_{11} * k_{00}} \right) * W_{16}^{n_{10} * k_{00}} \right] * W_{8^{10}}^{n_{10} * k_{01}} \right\} * W_{128}^{n_0 * (2 * k_{01} + k_{00})} * W_{8^{10}}^{n_0 * k_1}.$$

类似地, 448 结构的表达式:

$$X(k) = X(16 * t_2 + 4 * t_1 + t_0) = \sum_{s_0=0}^7 \left\{ \sum_{s_1=0}^3 \left[\left(\sum_{s_2=0}^3 x(s_2, s_1, s_0) * W_4^{s_2 * t_0} \right) * W_{16}^{s_1 * t_0} \right] * W_{4^1}^{s_1 * t_1} \right\} * W_{128}^{s_0 * (4 * t_1 + t_0)} * W_{80}^{s_0 * t_2}.$$

其中 $s_0=0 \dots 7, s_1=0 \dots 3, s_2=0 \dots 3, t_2=0 \dots 7, t_1=0 \dots 3, t_0=0 \dots 3$.

IFFT 能用相同的结构来运算, 只需在输入/输出端分别做一次共轭运算, 在运算过程中再除一个系数 128.

3 结构设计

不同的算法应采用不同的拓扑结构. Arch1 中, 为了采用流水线的处理方式与简化的硬件结构, 将蝶 2 部分放在后面计算, 如图 1 所示. 整个结构采用

8 条流水线并行处理. 首先, 数据按 16 抽取, 先进行并行蝶 8 运算, 计算结果乘以旋转因子后送入到 8 个串行蝶 8 运算单元. 前 8 组数据经串行蝶 8 运算后, 其结果送到 FIFO 缓冲区进行缓存. 等到 8 组 FIFO 缓冲区填满以后, 该区输出与串行蝶八模块的输出(需乘旋转因子)并行送入蝶 2 模块, 进行蝶 2 运算. 这样便可以得到 128 点 FFT 的运算结果. 8 个蝶 2 运算单元一个时钟可以输出 16 点数据, 其中 8 点数据是顺序的, 可以直接输出; 另外 8 点需要用 FIFO 缓冲器存储, 等前 64 点数据输出之后再输出.

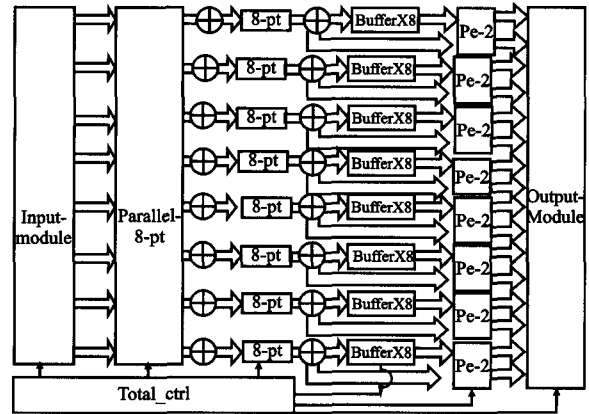


图 1 288 结构框图

而 Arch2 为 16 点并行流水线处理结构. 该结构包括并行处理的 16 点 FFT 运算模块和 16 个串行蝶 8 运算模块, 如图 2 所示. 16 点 FFT 模块采用两级蝶 4 的运算模块组成, 每级 4 个. 16 个串行通道在运算过程中无需使用 FIFO 缓冲器, 无堵塞与

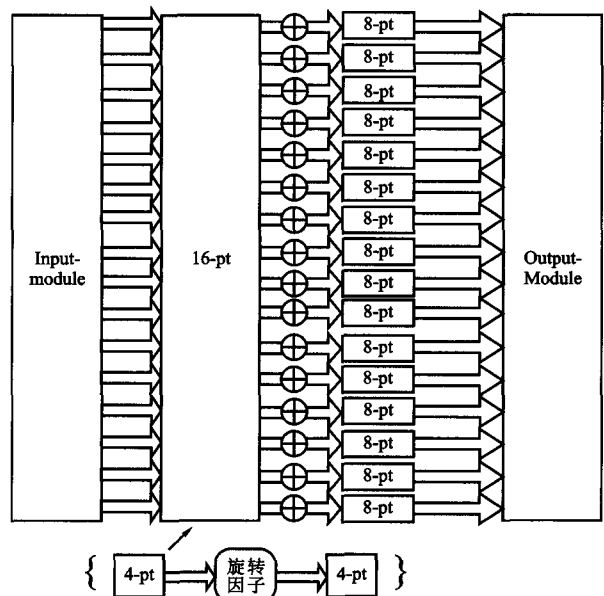


图 2 448 结构框图

等待现象,硬件结构上简单明了.在 Arch2 结构中,16 点 FFT 计算完成后,输出结果乘以对应的旋转因子,送入串行蝶 8 模块进行计算.串行蝶 8 模块的输出既是 128 点 FFT 的运算结果.输入是按 8 抽取的(隔 8 个数抽取一次),其输出结果是顺序的.

Arch1 中,从并行读出数据到并行蝶 8 运算再到乘以旋转因子,可控制在 2 个时钟周期内完成;串行蝶 8 模块需要 8 个时钟后才有数据输出;输出 8 点数据还需 8 个时钟,蝶 2 运算与旋转因子的乘法需 1 个时钟周期.所以,一组数据在读入后,需要经过至少 19 个时钟周期才能输出结果.而 Arch1 中,128 点 FFT 的运算共有 16 组数据依次输入.所以,128 点 FFT 全部运算完至少需要 35 个时钟. Arch2 中,蝶 4 运算需一个时钟周期完成,串行蝶 8 过程与 Arch1 相同.一组数据从输入到有数据输出需至少 11 个时钟周期,而 128 点全部运算完只需 19 个时钟.所以 Arch2 在所需时钟数上有优势,它对应的功耗也会降低.

输入模块: Arch1 与 Arch2 的输入模块都采用了乒乓存储器存储法,即存储区由两块相同的寄存器组成.当其中一块存储区处于读出状态时,另一块则处于写入状态.两块 RAM 交替的进行数据的写入与读出.这种存储方式有利于保证数据流的通畅,且时钟利用率提高了.

两个结构在输入数据的时候,均由信号 mode 和 d_in_en 控制,如图 3、图 4 所示. Mode 控制整个 IP 核的工作模式(FFT/IFFT),当 mode 为 1 的时候,该芯片进行 FFT 变换,反之,做 IFFT 的变换. d_in_en 用于输入数据的使能,高电平有效.在存储区资源的分配方面,两个结构略有不同. Arch1 的存储区如图 3 所示,整个存储区分成 8 块,每块 16 个 16 bit 的寄存器.数据输入端采用 16 位的数据线,128 点数据串行存入. 16 bit 的数据由 8 bit 实数和 8 bit 虚数组成,高 8 位是实数.而 Arch2 的存储区则分成 16 块,每块 8 个 16 bit 的存储单元,如图 4 所示.读出时, Arch1 按 16 抽取(隔 16 个数抽取一个),每次抽读取 8 点 16 bit 的数据.而 Arch2 按 8 抽取每次读取 16 点 16 bit 的数据.即 Arch1 分 16 次读取整组 128 点的数据,而 Arch2 只需 8 次.

16 点 FFT 单元: Arch1 采用 2×8 结构对 16 点 FFT 单元进行设计.同时,为了数据流的顺畅,将蝶 2 运算部分放在后面处理.输入模块的数据按 16 抽取,先进行并行蝶 8 的运算,后进行串行蝶 8 运算,最后进行蝶 2 运算,16 点 FFT 由串行蝶 8 和蝶 2 模

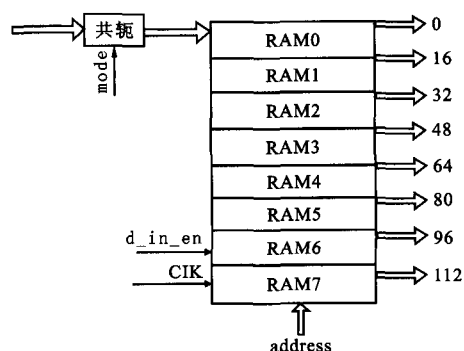


图 3 288 输入模块

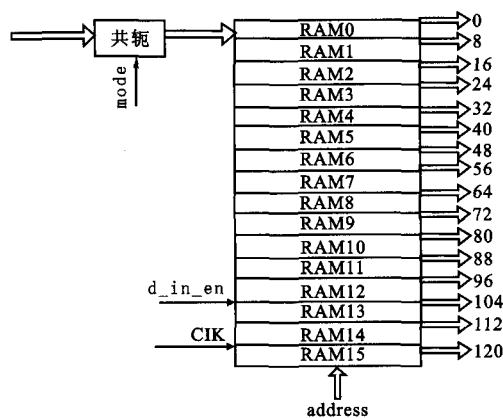


图 4 448 输入模块

块组成;串行蝶 8 单元与并行蝶 2 单元之间还需要一组 FIFO 缓冲器.每组缓冲器由 8 个寄存器组成,用于存储串行蝶 8 输出的前 8 组数据.待缓冲区填满后,与之后产生的 8 组数据进行蝶 2 运算.蝶 2 运算的输出结果一半可以直接输出,另一半需用 FIFO 存储.在 Arch2 中,16 点 FFT 采用两级蝶形单元实现,每级 4 组蝶 4 的运算模块.该单元以 16 通道读取 16 点数据,送到蝶 4 模块进行运算.对于所采用的两级蝶 4 模块,前级运算的 16 个结果乘以旋转因子后,可直接作为下级 4 个蝶 4 模块的输入,中间不需要存储.

8 点 FFT 单元: Arch1 中涉及到两种 8 点 FFT 的实现方式,即并行蝶 8 方式和串行蝶 8 方式.并行蝶 8 方式采用图 5 的蝶算过程实现,即采用三级蝶 2 单元进行运算,该过程共使用了 12 个 2 点的蝶形模块.而串行蝶 8 的实现方式采用如图 6 所示结构^[4],其工作原理以 stage1 为例进行说明. Stage1 共用了 2 个寄存器,用 r1、r2 表示,其位置关系如图所示.当第一个数据 $b(0)$ 输入时,先经过 add1_din 直接存储于 r1 寄存器.接下来当 $b(1)$ 输入时,一方面该输入经过 add1_din 存储到 r1 中,另一方面通

过多路选择器与 r1 输出的内容做加法运算;r1 的内容在第 2 个数据来临时,移位到 r2 中,同时参与了与 $b(1)$ 的运算,这时计算的结果就是 $X(0)=b(0)+b(1)$,从 add1_out 输出.在接下来的第 3 个时钟到来时,输入数据 $b(2)$ 通过 add1_din 直接送到存储 r1,多路选择器选择 r2 的输出,此时 r2 的输出内容是 $b(0)$,r1 存储的输出内容是 $b(1)$,加减运算器完成 $b(0)$ 与 $b(1)$ 的减法运算 $X(1)=b(0)-b(1)$. 同样,在后面的时钟里重复以上过程.其它两个 stage 完成类似的功能.另外,在 stage1 与 stage2 的加减法完成之后,需要乘以旋转因子.不同的序号要乘以不同的旋转因子,这些旋转因子分别是 $1,-1,-j,W_8^1,W_8^3$.具体乘哪一个因子,由 mul1_ctrl 与 mul2_ctrl 信号控制.

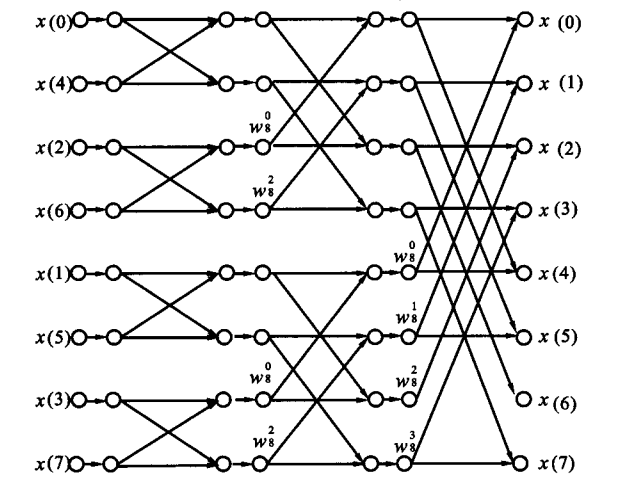


图 5 蝶 8 运算^[5]

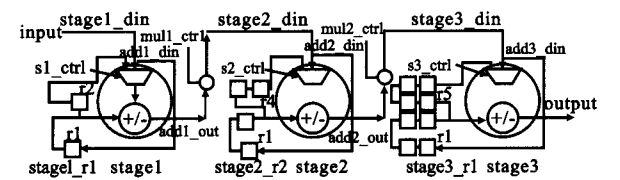


图 6 串行蝶 8 运算

输出单元:Arch1 输出模块中有 8 组 8 个 16 bit 的 FIFO 寄存器来存储结果.而 Arch2 中没有使用缓存器,16 个串行蝶 8 运算单元可以直接输出结果.

4 实现与功能评价

前面讲的算法用 verilog HDL 实现,并且在 ModelSim™里通过仿真.仿真输出也已经跟 MATLAB 里计算的结果进行对比印证.将两个结构的 128 点 FFT IP 核用 synopsys 工具 DC compiler 进行综合,得到了它们的面积和功耗等性能参数.综合过程中使用了 SMIC0.18 μm 的库.综合结果如表 1

所示.从表中可以发现,在功耗方面,第 2 个结构更小,只用了 93.9 mW,而 Arch1 却用了 141 mW. Arch1 的高功耗主要来源于缓冲器模块的应用与较高的工作频率等.面积上 Arch1 要比 Arch2 小.具体的面积占比如图 7、图 8 所示.

表 1 功耗/面积对比

参数	Arch1	Arch2	Arch3	Arch4
面积/ μm^2	3 117 727	3 528 437	3 444 842	3 644 674
功耗/mW	141.56	93.91	198	121

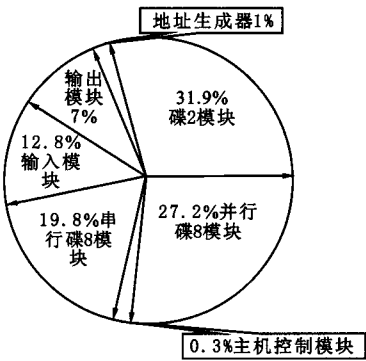


图 7 Arch1 面积占比

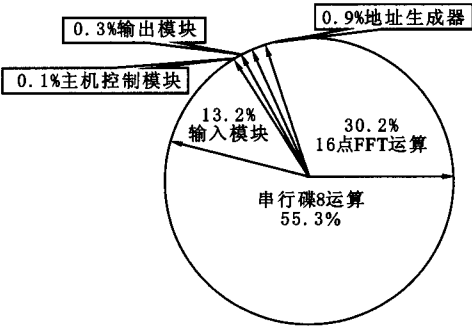


图 8 Arch2 面积分布

FFT 处理器主要是用来在算法上对数据进行时域到频域的变换,其核心部分是用来做数据处理的算法运算模块.而芯片面积也主要来源于运算模块. Arch1 的运算模块包括并行、串行的 8 点 FFT 模块以及蝶 2 模块,该部分占用了 78.8% 的面积,如图 7 所示. Arch2 的运算模块占了总体面积的 85.5%,如图 8 所示.运算模块之所以占如此多的面积,主要原因是因为其中使用了大量的复数乘法器.而 Arch1 面积比 Arch2 小的原因,主要是因为 Arch1 所消耗的运算资源比 Arch2 更少.据统计,完成 Arch1 的运算,总共使用了 21 个复数乘法器,而 Arch2 中却用了 55 个.

(下转第 48 页)

转引入了左右邻居表,使得节点间查找时间大大减少,有效的提升了整个网络拓扑的性能.

参考文献:

- [1] Stoica I, Morris R, Karger D, et al. Chord: a scalable peer-to-peer lookup service for internet applications [C] //Proc. of ACM SIGCOMM 2001. San Diego, CA. 2001:149-160.
- [2] Deshpande H, Bawa M, Garcia-molina H. Streaming live media over a peer to peer network[J]. Technical Report Stanford database group technical report, 2001, 3(2):15-23.
- [3] 刘金岭. 基于 P2P 网络的 AVL 索引树范围查询研究[J]. 微电子学与计算机, 2011, 28(2):11-12.
- [4] 毛瑞芳, 李清霞. 一种基于二叉平衡树的 P2P 覆盖网络的研究[J]. 应用科技, 2009, 36(4):40-44.
- [5] Suyong E, Arakawa Shin'ichi A, Masayuki M. Self organizing topology transformation for peer-to-peer networks[J]. IEICE Transactions on Communications, 2010(3):516-524.
- [6] Jonathan S, Omer L, Raphael G. An optimal topology for a static P2P live streaming network with limited resources [C] //Proc. of International Conference on Parallel and Distributed Systems. Tainan, Taiwan, 2011(27):729-734.

作者简介:



倪朔东 男, (1981-), 硕士, 实验师. 研究方向为 P2P 网络、流媒体技术.

袁蔚芳 女, (1982-), 硕士, 讲师. 研究方向为网络与数据库.

施 佳 男, (1973-), 博士, 副教授, 硕士生导师. 研究方向为智能信息处理.

(上接第 43 页)

5 结束语

文中提出了两种 128 点 FFT 结构, 从算法、结构以及功能方面, 对它们各自的优点和缺点进行分析. 我们发现, 在功耗方面 Arch2 要比 Arch1 更有优势, 而在面积方面, Arch1 更好. 两种算法都有规则的数据流, 控制器结构都很简单, 都有利于充分发挥流水线运算单元的处理能力. 两种结构的 FFT IP 核都能满足 UWB-OFDM 系统的要求.

参考文献:

- [1] Aiello G A, Rogerson G D. Ultra-wideband wireless systems[J]. IEEE Microwave Mag June, 2003, 4(2): 36-47.
- [2] Huggett C, Maharatna K. On the implementation of 128-pt fft/fft for high performance WPAN[C]//IEEE International Symposium on Circuits and systems, ISCAS2005. UK: Bristol University, 2005.
- [3] Mathew J, Maharatna K, Jose B R, et al. Pseudo-parallel datapath structure for power optimal implementation of 128-pt FFT/IFFT for WPA-N[J]. Circuits Syst Signal Process, 2011, 30(4):871-882.
- [4] Shen-fu Hsiao, Wei ren Shiue. Design of low cost and high throughput linear arrays for DFT computations: algorithms, architectures, and implementations [J]. Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on, 2000, 47(11):1188-1203.
- [5] 程佩青. 数字信号处理教程[M]. 2 版. 北京: 清华大学出版社, 2001.
- [6] Mathew J, Maharatna K, Pradhan D K. A low power 128-Pt implementation of FFT/IFFT for high performance wireless personal area networks[C]//IEEE PRIME Conference. Italy: Otranto, 2006:377-380.

作者简介:

施隆照 男, (1968-), 副教授. 研究方向是数字集成电路设计与系统集成.

高峰 男, (1988-), 硕士研究生. 研究方向为数字集成电路设计.