DOI:10.3969/j.issn.1003-5060.2023.07.008

一种高速2-D滑动FFT的设计实现

许丁鸿[1,2], 张多利[1,2], 陶相颖[1,2], 韩帅鹏[1,2], 宋宇鲲[1,2]
(1.合肥工业大学 微电子学院,安徽 合肥 230601; 2.教育部IC设计网上合作研究中心,安徽 合肥 230601)

摘 要:文章介绍了采用2-D快速傅里叶变换(fast Fourier transform,FFT)算法的滑动窗FFT的基本特性原理和硬件实现过程,完成了窗长256点、步长16点的2-D滑动窗FFT的专用集成电路(application specific integrated circuit,ASIC)设计。传统FFT算法受序列完整性的制约,时滞较大,无法满足某些高实时性信号分析领域的处理速度要求。该文采用滑动FFT算法,克服了传统FFT对序列完整性的依赖,设计的滑动FFT处理器使用2-D FFT压缩新序列计算时间,以基16蝶形运算器为核心,采用系数复用和高基Booth方法优化系数编码技术压缩乘法器的数量,减少电路面积。所设计的2-D滑动FFT完成单次滑动窗长的计算时间比传统算法节约了16.1%,变换结果与MATLAB的运算结果相比,信噪比(signal-to-noise ratio,SNR)大于130 dB。在TSMC 28 nm的工艺下,工作主频为600 MHz,面积为1 980 μm×2 060 μm。

关键词:快速傅里叶变换(FFT);滑动FFT;2-D FFT算法;高基Booth编码
中图分类号:TN47 **文献标志码**:A **文章编号**:1003-5060(2023)07-0912-07

A design and implementation of a high speed 2-D sliding FFT

XU Dinghong[1,2], ZHANG Duoli[1,2], TAO Xiangying[1,2], HAN Shuaipeng[1,2], SONG Yukun[1,2]
(1. School of Microelectronics, Hefei University of Technology, Hefei 230601, China; 2. IC Design Web-cooperation Research Center of Ministry of Education, Hefei 230601, China)

Abstract: This paper introduces the basic characteristic principles and hardware implementation of sliding fast Fourier transform(FFT) using 2-D FFT algorithm, and completes the application specific integrated circuit(ASIC) design of 2-D FFT with 256 window length and 16 step size. With the restriction of sequence integrity, the traditional FFT algorithm cannot meet the processing speed requirements of some high real-time signal analysis fields due to its large time delay. In this paper, sliding FFT algorithm is used to overcome the dependence of traditional FFT on sequence integrity. The sliding FFT processor uses 2-D FFT to compress the computation time of the new sequence, takes radix-16 butterfly arithmetic unit as the core, and adopts coefficient multiplexing and high radix Booth method to optimize coefficient coding technology, so as to compress the number of multipliers and reduce the circuit area. Compared with the traditional algorithm, the calculation time of single sliding window length of the designed 2-D sliding FFT is reduced by 16.1%. Compared with the result of MATLAB, the signal-to-noise ratio(SNR) of the 2-D sliding FFT is more than 130 dB. In the TSMC 28 nm process, the main frequency is 600 MHz and the area is 1 980 μm×2 060 μm.

Key words: fast Fourier transform(FFT); sliding FFT; 2-D FFT algorithm; high radix Booth coding

在波形分解[1]、医学成像[2]、雷达[3]以及通信系统[4]等诸多应用中,都采用快速傅里叶变换(fast Fourier transform,FFT)算法完成信号的时域与频域间变换。在远程医疗、口语互译等实时

收稿日期:2021-05-11;**修回日期**:2021-07-04
基金项目:国家自然科学基金资助项目(61874156);安徽省高校协同创新资助项目(GXXT-2019-030)
作者简介:许丁鸿(1996—),男,浙江台州人,合肥工业大学硕士生;
张多利(1976—),男,黑龙江七台河人,博士,合肥工业大学研究员,博士生导师,通信作者:E-mail:zhangduoli@hfut.edu.cn.

性应用领域，受传统 FFT 算法对序列完整性依赖的约束，无法在序列采样过程中开展 FFT 变换，变换的实时性难以达到目标应用要求。此外，在处理时域数据更新周期较短的长序列信号时，受运算复杂度和计算实时性约束，传统 FFT 只能依靠大量计算器资源达到设计目标，导致整个硬件结构异常复杂[5]。

针对此类问题，文献[6]使用滑动 FFT 算法。滑动 FFT[7]的执行过程为迭代累加累乘计算，可在前一次滑动 FFT 结果基础上递推得到当前滑动 FFT 结果，复用前次运算结果，节约运算时间，提高信号分析实时性。利用滑动 FFT 运算递推特征，可以根据信号分析需要选择计算相应的频谱区段，增加信号分析效果。

本文设计的滑动 FFT 变换器采用 2-D FFT[8]硬件架构，即将 1-D 长序列 FFT 分解成由若干条短序列组成的矩阵，再分别对行/列向量进行 1-D FFT 运算，利用行/列向量间无关性实现多路并行。FFT 处理器通常分为顺序处理结构、级联处理结构[9]、并行迭代结构[10]、阵列结构 4 类，本文根据应用需求选择阵列结构。

1　算法原理

1.1　滑动 FFT 原理

设时域信号波形的采样值为 $x(n)$，傅里叶变换截取的滑窗的长度为 N，滑窗每次滑动的步长为 $p=2^L$。记 i 点起连续采样 N 点数据的傅立叶变换为 $X_i(k)$，则 $i+p$ 点至 $i+p+N$ 点的 N 点傅立叶变换记为 $X_{i+p}(k)$。

根据离散傅里叶变换（discrete Fourier transform，DFT）的公式假设，即

$$X_i(k)=\sum_{m=0}^{N-1}x_i(m)\exp\left(-\mathrm{j}\frac{2\pi}{N}mk\right),\quad k=0,1,2,\cdots,N-1 \tag{1}$$

$i+p$ 点的 FFT 公式可以表示为：

$$X_{(i+p)}(k)=\sum_{m=p}^{N-1+p}x_i(m)\exp\left[-\mathrm{j}\frac{2\pi}{N}(m-p)k\right] \tag{2}$$

由式(1)及 $\exp\left(-\mathrm{j}\frac{2\pi}{N}pk\right)$ 与 n 无关性，式(2)化简为：

$$X_{(i+p)}(k)=\left[X_i(k)+\sum_{n=0}^{p-1}[x_i(n+N)-x_i(n)]\exp\left(-\mathrm{j}\frac{2\pi}{N}nk\right)\right]\exp\left(\mathrm{j}\frac{2\pi}{N}pk\right) \tag{3}$$

令

$$E(k)=\sum_{n=0}^{p-1}[x_i(n+N)-x_i(n)]\times\exp\left(-\mathrm{j}\frac{2\pi}{N}nk\right)=\sum_{n=0}^{p-1}e(n)\exp\left(-\mathrm{j}\frac{2\pi}{N}nk\right) \tag{4}$$

式(4)中 $e(n)$ 的前 $p=2^L$ 点为 $x_i(n+N)-x_i(n)$，后 $N-p$ 点为 0。

文献[11]提出采用 Pruning 算法解决上述特征序列 FFT 计算问题。由式(3)可得，当窗长 $N=2^M$，步长 $p=2^L$ 时，仅需 L 级的 FFT 运算即可得出 N 点的 FFT 值。前 $M-L$ 级的运算都是输入的 $p=2^L$ 序列与 0 进行蝶形运算，也就是前 $p=2^L$ 点数据的复制，并不需要用到加法和乘法运算。$N=16$、$L=2$ 的 Pruning 算法图如图 1 所示。图 1 中：虚线表示 0 值；实线表示输入序列值。

从图 1 可以看出：第 1 级与第 2 级是对输入序列与 0 值做蝶形运算，可视为对输入值的复制；第 3 级与第 4 级是 2 个输入值为非零的复数蝶形运算。

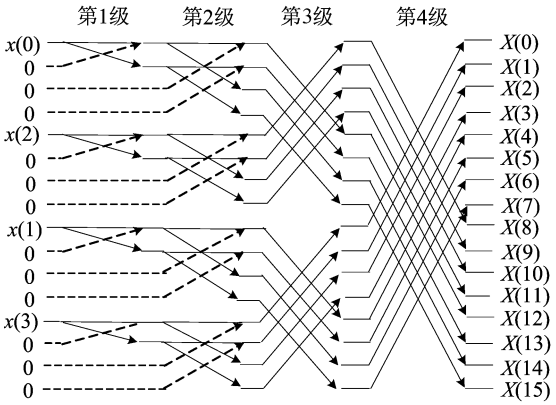

图 1　输入值为 4 点的 16 点 FFT Pruning 算法图

假设已知 $X_i(k)$ 的 N 点 FFT 值，滑动步长为 $p=2^L$，计算 $X_{i+p}(k)$ 的 FFT 值，采用滑动 FFT 算法和使用传统 FFT 算法节约的时间比值 t_τ 计算公式为：

$$t_\tau=\frac{M-L}{M}=1-\frac{L}{m} \tag{5}$$

滑动 FFT 算法需要进行 $NL/2+N$ 次复数乘法、$NL+N$ 次复数加法，而传统的 FFT 算法需要进行 $(N\mathrm{lb}\ N)/2$ 次复数乘法、$N\mathrm{lb}\ N$ 次复数加法。滑动窗长度为 64～256 点，滑动步长为4～16 点，滑动窗每滑动一次产生的复数乘法计算量与复数加法计算量的关系见表 1 所列。由表 1 可知，滑动 FFT 所需复数乘法数量和复数加法数量都小于传统 FFT 算法。

表 1 滑动窗长、滑动步长与运算量的关系

窗长度	计算类型	滑动步长	复乘法次数	复加法次数
256	传统 FFT		1 024	2 048
	滑动 FFT	16	768	1 280
		8	640	1 024
		4	512	768
128	传统 FFT		448	896
	滑动 FFT	16	384	640
		8	320	512
		4	256	384

1.2 2-D FFT 算法原理

2-D FFT 算法[12]是一种优化 1-D 长序列 FFT 的运算方法,该算法将 N 点长序列 FFT 运算分解为两级短序列 FFT 级联运算。

对 N 点数据进行 FFT 运算,令 $N=LC$,即将输入的 N 点分解为 L 行、C 列的矩阵形式。根据 DFT 算法公式,有

$$X(k)=X(Lk_1+k_0)=X(k_1,k_0)=\mathrm{DFT}[x(n)]=\mathrm{DFT}[x(Cn_1+n_0)]=\mathrm{DFT}[x(n_1,n_0)] \tag{6}$$

时间域变量 n 可以表示为:

$$n=Cn_1+n_0 \tag{7}$$

其中,$n_1=0,1,\cdots,L-1$;$n_0=0,1,\cdots,C-1$。

频率域变量 k 可以表示为:

$$k=Lk_1+k_0 \tag{8}$$

其中:$k_1=0,1,\cdots,C-1$;$k_0=0,1,\cdots,L-1$。

将式(7)、式(8)代入离散傅里叶变换式(1)中,并利用旋转因子周期性 $W_N^{LCn_1k_1}=1$ 和可约性,得到等价式:

$$X(k_1,k_0)=\sum_{n_0=0}^{C-1}W_C^{n_0k_1}\left[\sum_{n_1=0}^{L-1}x(n_1,n_0)\times W_L^{n_1k_0}\right]W_N^{n_0k_0} \tag{9}$$

令

$$G(k_0,n_0)=\sum_{n_1=0}^{L-1}x(n_1,n_0)W_L^{n_1k_0} \tag{10}$$

$$H(k_0,n_0)=G(k_0,n_0)W_N^{n_0k_0} \tag{11}$$

得:

$$X(k_1,k_0)=\sum_{n_0=0}^{C-1}H(k_0,n_0)W_C^{n_0k_1}=X_1(k_0,k_1) \tag{12}$$

其中:$G(k_0,n_0)$为 L 点的列 FFT 变换;$X_1(k_0,n_0)$为 C 点行 FFT 变换。因为输出序列 $X(k)$的序列顺序是满足先填写行方向再填写列方向,和输入数据相逆,所以还需要一次整序才是正确的 FFT 序列。

对一个 N 点的大点数进行 FFT 运算可以分解为 C 组 L 点一维列变换与 L 组 C 点的一维行变换的级联,其实现步骤如下:

1) 将 N 点表示为 L 行、C 列的矩阵。

2) 对矩阵的所有列进行 L 点 FFT 列变换得到 $G(k_0,n_0)$。

3) 将 $G(k_0,n_0)$乘以中间的旋转因子 $W_N^{n_0k_0}$ 得到 $H(k_0,n_0)$。

4) 对 $H(k_0,n_0)$矩阵进行 C 点的 FFT 行变换得到 $X_1(k_0,k_1)$。

5) 将 $X_1(k_0,k_1)$整序为 $X(k_1,k_0)$,然后输出。

1.3 硬件架构选择

本文采用 2-D 滑动 FFT 硬件结构,从以下方面进行考虑。

1) 算法性能。由式(3)可知,滑动窗由 p 个有效值与 $N-p$ 个 0 值构成,以本文设计的参数滑动窗长为 16、滑动步长为 4 举例,使用 2-D FFT 结构可以构成第 1 行为滑动步长的有效值,其余 3 行都为 0 的二维矩阵,如图 2 所示。

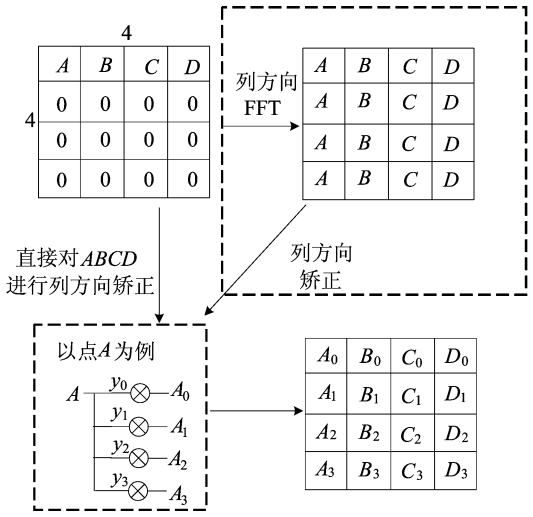

图 2 2-D 滑动 FFT 的 2-D 运算图

根据二维矩阵的运算顺序,先执行列方向矫正计算,因为列构成为 1 个有效值与 3 个 0 值,该 FFT 结果等于有效值的列方向复制,所以可以忽略列方向 FFT,直接对输入数据进行列矫正。

为达到计算速度要求,本文设计行方向采用全展开 FFT 结构,对±1 和±i 的旋转因子进行操作,采用交换虚实部的方法压缩运算,对其他旋转因子系数使用优化编码方法压缩运算。

2) 压缩资源。首先计算过程中间数据存储资源的消耗，1-D 架构为全序列长度，2-D 架构为短序列中最长的序列长度。在设计中短序列采用全展开结构，无需缓冲中间结果。

其次是压缩浮点数乘法资源。滑动 FFT 运算中的矫正运算是影响设计速度的关键，以滑动窗长为 256 点、滑动步长为 16 为例，滑动 FFT 运算的矫正因子系数为 ±0.707 1、±0.382 7、±0.923 9。对上述因子使用高基 Booth 编码操作将浮点数乘法转变为常数乘法，优化逻辑结构，减少资源。

最后是公用因子优化复用。式(3)中滑动 FFT 运算的矫正因子系数为 ±0.707 1、±0.382 7、±0.923 9，也是 16 点 FFT 运算的旋转因子系数。对滑动 FFT 矫正运算的优化方法可以在 16 点FFT 运算中得到复用。

综上所述，本文设计采用 2-D 滑动 FFT 结构使其可多路并行并减少电路复杂度及 FFT 运算时间，通过对基 16 核的优化和一部分定点数乘法的优化来减少资源消耗。

2　系统结构

本文设计的 2-D FFT 处理器结构如图 3 所示。

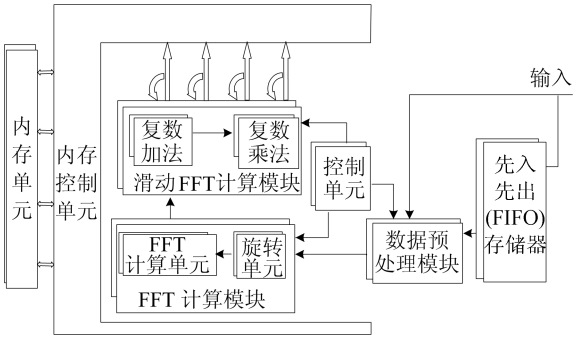

图 3　2-D FFT 处理器结构

处理器结构包括以下部分：① 控制单元模块(control unit)，用来调节各个模块的工作内容以及状态，对模块置零的控制；② 数据预处理模块(data preproccessing unit)，完成式(3)中 $x_i(n+N)-x_i(n)$ 的步骤以及对数据的倒序处理；③ FFT运算模块(FFT calculate module)，其中包含了输入数据与 2-D FFT 的列方向矫正模块(revolve unit)和基 16 的 FFT 运算单元(FFT computing unit)；④ 滑动 FFT 的运算模块(sliding FFT calculate)，完成 FFT 运算模块的结果与上一次滑动 FFT 运算结果进行相加的复数加法模块(complex add)，以及对该结果进行复乘调整操作的复数乘法模块(complex mul)；⑤ 存储控制单元模块(memory control unit)，实现地址无冲突规则，管理存储单元的访存；⑥ 存储单元组(memory unit)，由多块 memory 组成，负责存储当前时刻滑动 FFT 的结果。

滑动 FFT 处理器的工作流程如图 4 所示。

1) 滑动窗 FFT 接收到启动信号后开始工作，单次滑动窗计算结束后判断当前序列处理是否完成，否则读取源数据并写入先入先出(first input first output，FIFO)和预处理模块中。

2) 当累计输入数据的长度等于滑动窗长度时，在 INPUT 端口读取 $x_i(n+N)$，从 FIFO 读取 $x_i(n)$，在数据预处理模块中完成 $x_i(n+N)-x_i(n)$ 操作以及倒序操作，每 16 个周期从预处理模块输出 16 个浮点数。

3) 数据预处理模块输出的 16 个结果送入 FFT 列方向矫正模块(revolve unit)，每个周期完成 16 次与矫正因子复乘运算，得到16 个结果；再将其结果分输入 FFT 单元(FFT computing unit)。

4) FFT 单元完成 16 点 FFT 计算。

5) FFT 单元结果输入滑动 FFT 运算模块(sliding FFT calculate)进行矫正运算，即与前一次滑动 FFT 结果进行复数加法与复数乘法。

6) 上述结果分别写入结果 memory 并输出，同时作为前一次的 FFT 结果暂存在滑动 FFT 运算模块，参与下一次的运算。

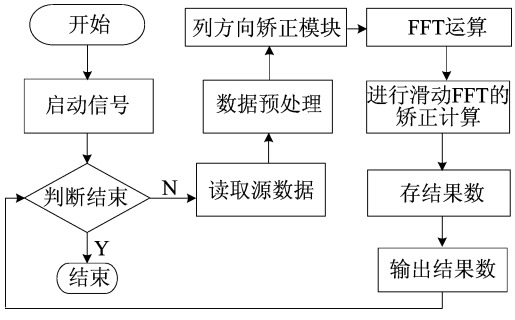

图 4　2-D 滑动 FFT 处理器工作流程

2.1　基 16 蝶形运算器设计

设计的滑动 FFT 矫正模块需要在每个周期内完成 16 点的复数加法与复数乘法，相应地，本文设计必须在每个周期输出 16 点的 FFT 结果，因此采用基 16 核的全展开结构。

当旋转因子系数为±1 和±i 时，可以通过交换符号位以及实部或虚部的方法规避复数乘法。

剩余部分运算使用基 2 来构建基 16 核需要 10 次复乘与 64 次复加;使用基 4 来构成基 16 核需要 8 次复乘与 96 次复加,因此本文设计使用基 2 来构建基 16 的核,如图 5 所示[13]。

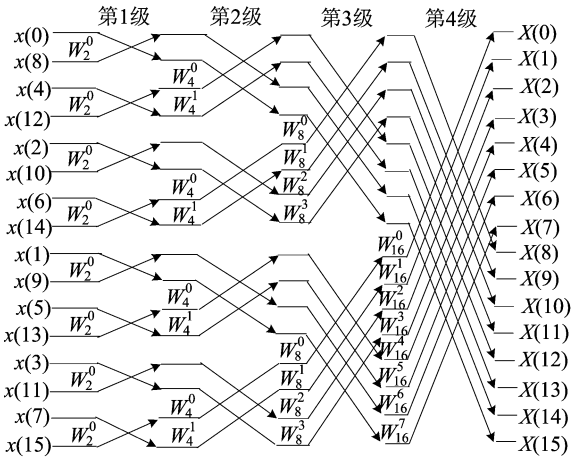

图 5 16 点蝶形单元结构

2.2 使用高基 Booth 编码优化定点数乘法

本文设计的滑动 FFT 运算矫正模块需要在 16 个周期内完成 256 点的复加与复乘,因此必须压缩浮点数乘法器的关键路径长度。

IEEE-754 浮点数由 1 位符号位、8 位阶码、23 位尾码构成,2 个浮点数相乘是阶码相加,尾码相乘。在本文设计中,16 点 FFT 运算的旋转因子系数与滑动 FFT 运算的矫正因子系数相同,仅有±0.707 1、±0.382 7、±0.923 9。因此可对系数 0.707 1、0.382 7、0.923 9 进行 Booth 编码,使用常数与变量乘法替代通用浮点乘法,压缩计算时间[14]。

本文设计使用高基 Booth 编码进行系数优化[15],编码的公式为:

$$x_{i+k+1} = -2^{k+1},\quad x_{i+k} = 2^{k},\quad x_{(i-1)} = 2^{0},\quad k = 0,1,\cdots,a-2 \tag{13}$$

其中,2^a 为 Booth 的基。0.707 1、0.382 9、0.923 9 的高基编码情况如图 6 所示。

图 6 中 2 的幂次方乘法可以通过移位来获得,其余通过相加获得(如 3、6 等),因为每个系数都可找到一个基获得最小加法执行次数,所以本文设计中 0.707 1、0.382 7 使用基 16 编码,0.923 9 使用基 32 编码,即上述 3 个系数的乘法操作分别转换为 8 次加法操作,达到了资源最小化目标。

在 TSMC 28 nm 工艺下,高基 Booth 运算器模块的 DC 综合频率在 700 MHz 以上,资源的消耗情况见表 2 所列。

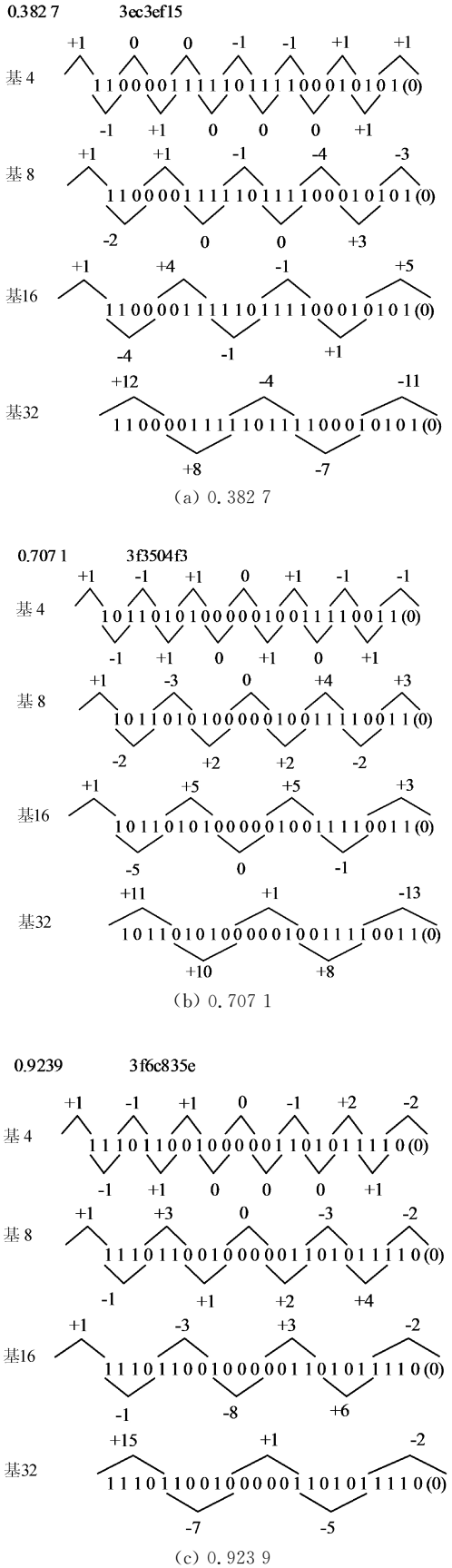

图 6 0.707 1、0.382 7、0.923 9 的高基编码情况

表 2　Booth 编码乘法器与 DW 乘法器的资源对比

乘法器类型	单元数量/个	资源优化率/%	流水级数	频率/MHz
Design Ware(DW)	1 852		5	600
0.382 7	664	64.1	1	600
0.707 1	702	62.1	1	600
0.923 9	651	64.8	1	600

2.3　滑动 FFT 的矫正模块旋转因子结构设计

本文设计的参数为窗长度 256，滑动窗步长为 16。式由(3)可知：

$$\exp(\mathrm{j}\frac{2\pi}{N}pk)=\exp(\mathrm{j}\frac{2\pi}{256}16k)=\exp(\mathrm{j}\frac{\pi}{8}k)。$$

每 16 个值一次循环，对应着 2-D FFT 运算结果，矫正因子的排列方式如图 7 所示。

$\exp(\mathrm{j}\frac{\pi}{8}k_0)$	$\exp(\mathrm{j}\frac{\pi}{8}k_{16})$	$\exp(\mathrm{j}\frac{\pi}{8}k_{32})$	$\exp(\mathrm{j}\frac{\pi}{8}k_{48})$	…	$\exp(\mathrm{j}\frac{\pi}{8}k_{240})$
$\exp(\mathrm{j}\frac{\pi}{8}k_1)$	$\exp(\mathrm{j}\frac{\pi}{8}k_{17})$	$\exp(\mathrm{j}\frac{\pi}{8}k_{33})$	$\exp(\mathrm{j}\frac{\pi}{8}k_{49})$	…	$\exp(\mathrm{j}\frac{\pi}{8}k_{241})$
$\exp(\mathrm{j}\frac{\pi}{8}k_2)$	$\exp(\mathrm{j}\frac{\pi}{8}k_{18})$	$\exp(\mathrm{j}\frac{\pi}{8}k_{34})$	$\exp(\mathrm{j}\frac{\pi}{8}k_{50})$	…	$\exp(\mathrm{j}\frac{\pi}{8}k_{242})$
$\exp(\mathrm{j}\frac{\pi}{8}k_3)$	$\exp(\mathrm{j}\frac{\pi}{8}k_{19})$	$\exp(\mathrm{j}\frac{\pi}{8}k_{35})$	$\exp(\mathrm{j}\frac{\pi}{8}k_{51})$	…	$\exp(\mathrm{j}\frac{\pi}{8}k_{243})$
⋮	⋮	⋮	⋮		⋮
$\exp(\mathrm{j}\frac{\pi}{8}k_{15})$	$\exp(\mathrm{j}\frac{\pi}{8}k_{31})$	$\exp(\mathrm{j}\frac{\pi}{8}k_{47})$	$\exp(\mathrm{j}\frac{\pi}{8}k_{63})$	…	$\exp(\mathrm{j}\frac{\pi}{8}k_{255})$

图 7　滑动 FFT 矫正因子排列

由图 7 可知：1、5、9、13 行的矫正因子的实部与虚部由±1 构成；2、6、10、14 行的矫正因子的实部与虚部由±0.382 7、±0.923 9 构成；3、7、11、15 行的矫正因子的实部与虚部由±0.707 1 构成；4、8、12、16 行的矫正因子的实部与虚部由±0.382 7、±0.923 9 构成。

设输入的复数为 $a+b\mathrm{i}$，使 a 和 b 分别与定点数的常数 0.707 1、0.382 7、0.923 9 进行乘法计算，然后根据状态机来进行带有正负的实部与虚部的组合排列。

3　设计实现与性能分析

SMIC 28 nm 工艺下，本文设计使用 Synopsys 公司 Design Compiler 和 IC Compiler 完成后端设计工作，设计的滑动窗 FFT 变换器工作主频高于 700 MHz，核心面积为 1 980 μm×2 060 μm，版图如图 8 所示。

因为本文设计采用的滑动窗算法是一个乘法累加的情况，所以计算误差随滑动窗滑动次数增多而累加。运算周期性地清零[16]可以有效防止误差累积，使设计的输出稳定在一个误差范围内。

本文设计在 Xilinx Virtex-7 FPGA 芯片上进行功能和性能测试，输入序列生成采用不同频率的单频正弦信号叠加随机噪声。

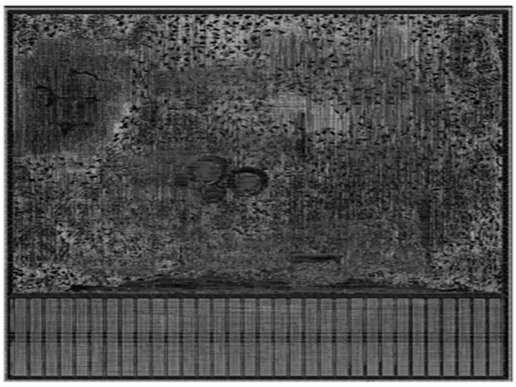

图 8　滑动窗 FFT 版图

参考结果为 MATLAB 中 FFT 函数的输出。

$$y_1(t)=\sin(2\pi\times 50\times t\times 1/n)+5\mathrm{randn}(\mathrm{size}(t)),$$
$$y_2(t)=\sin(2\pi\times 100\times t\times 1/n)+5\mathrm{randn}(\mathrm{size}(t)),$$
$$y_3(t)=\sin(2\pi\times 150\times t\times 1/n)+5\mathrm{randn}(\mathrm{size}(t)),$$
$$y_4(t)=\sin(2\pi\times 200\times t\times 1/n)+5\mathrm{randn}(\mathrm{size}(t))。$$

对以上 4 段信号进行 $n=1\ 024$ 的采样，采样间隔为 $1/n$。由以上 4 段正弦随机信号拼接成一个4 096的不同频率的序列数据。滑动窗长度 256，滑动步长 16，一共产生 256 段数据。在滑动了不同长度之后进行数据复位，保证误差精度后的误差分析，结果见表 3 所列。

表 3　滑动 FFT 复位前 10 段结果数据的误差分析

复位长度	实部		
	平均误差	均方差	信噪比/dB
1 024	2.800 8E−06	2.013 9E−09	137.922
2 048	5.037 2E−06	6.518 3E−09	135.541
3 072	7.798 7E−06	1.562 8E−08	133.122
4 096	1.086 0E−05	3.032 2E−08	131.592
复位长度	虚部		
	平均误差	均方差	信噪比/dB
1 024	2.789 7E−06	2.000 8E−09	138.095
2 048	5.166 0E−06	6.850 1E−09	135.266
3 072	7.695 7E−06	1.524 7E−08	133.182
4 096	1.076 2E−05	2.974 2E−08	131.044

设计要求在 600 MHz 以上，使用 Booth 编码对旋转/矫正因子系数的常数化操作不仅减少了

消耗资源,也提高了工作速度。600 MHz的工作频率,Design Ware乘法器需要5级流水,而使用Booth编码设计在单周期即可达到频率要求。Booth编码的优化系数也用于FFT计算模块。2种实现工程的运算周期数对比见表4所列。

表4 2-D滑动FFT不同乘法器类型运算周期数

乘法器类型	滑动序列长度	运算周期
DW	4 096	4 220
Booth编码	4 096	4 210

在参数滑动窗长为256、滑动步长为16的案例下,2-D滑动FFT处理器与1-D滑动FFT处理器的单次滑动后,计算滑动窗长所需周期见表5所列。

表5 不同滑动FFT处理器单次滑动后滑动窗计算所需周期

滑动FFT处理器类型	运算周期	优化率/%
1-D滑动FFT(理论值)	136	
2-D滑动FFT(本文方法)	114	16.1

4 结 论

本文给出了一种高速高精度的2-D滑动FFT处理器设计,采用IEEE-754单精度浮点数数据格式,支持256点步长16点的FFT计算,并完成了TSMC 28 nm工艺下后端实现。实验结果表明,本文所设计的FFT处理器的工作频率≥600 MHz,计算精度达到10^{-4},SNR大于130 dB。

[参 考 文 献]

[1] 王小军,周学科,宫世子.连续滑动窗的递推-Pruning-FFT算法[J].西安邮电学院学报,2004,9(2):99-102.

[2] HAN B,SCHOTTEN H D. A cost efficient and flexible cyclostationary feature detector based on sliding discrete Fourier transform for cognitive spectrum sensing[C]//2017 24th International Conference on Telecommunications (ICT).[S.l.]:IEEE,2017:1-5.

[3] LI Z, ZHANG X, MA Y, et al. Noncontact detection of heartbeat and respiratory rate via 77 GHz radar based on adaptive double sliding-time window algorithm[C]//2019 IEEE International Conference on Signal, Information and Data Processing (ICSIDP).[S.l.]:IEEE,2019:1-5.

[4] ZHANG Y,ZHOU R,RHEE W,et al. A 6.1 mW 5 Mb/s 2.4 GHz transceiver with F-OOK modulation for high bandwidth and energy efficiencies[C]//2017 IEEE Custom Integrated Circuits Conference (CICC).[S.l.]:IEEE,2017:1-4.

[5] HU S,XIE Y,LIAN J,et al. The verification system of sliding spotlight mode SAR imaging based on SoPC[C]//2019 IEEE International Conference on Signal, Information and Data Processing (ICSIDP).[S.l.]:IEEE,2019:1-5.

[6] CSUKA B,KOLLÁR I,KOLLÁR Z,et al. Comparison of signal processing methods for calculating point-by-point Discrete Fourier Transforms[C]//2016 26th International Conference Radioelektronika (RADIOELEKTRONIKA).[S.l.]:IEEE,2016:217-221.

[7] 李永忠,安文森.一种改进的滑动窗FFT递推算法[J].信号处理,2007(1):141-143.

[8] ZHU Y S,JIN L W,ZHANG Z S. Architecture of array processors of 1-D,2-D complex and real DFT[C]//1991 International Conference on Circuits and Systems.[S.l.]:IEEE,1991:843-846.

[9] WANGZ K,LIU X,HEB S. A combined SDC-SDF architecture for normal I/O pipelined radix-2 FFT[J]. IEEE Transactions on Very Large Scale Integration Systems, 2015, 23(5):973-977.

[10] LIU S,LIU D. A high-flexible low-latency memory-based FFT processor for 4G,WLAN,and Future 5G[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems,2019,2018:1-13.

[11] SKINNER D. Pruning the decimation in-time FFT algorithm[J]. IEEE Transactions on Acoustics, Speech, and Signal Processing,1976,24(2):193-194.

[12] TANNO K,TAKEDA T,HORIGUCHI S. Parallel 2-D FFT algorithms on an eight-neighbor processor array[C]//Singapore ICCS/ISITA'92.[S.l.]:IEEE,1992:621-627.

[13] 刘小明,吴曼青,洪一.基于FPGA的基-16FFT模块的实现[J].合肥工业大学学报(自然科学版),2006,29(5):536-539.

[14] KUMAR U C S P,GOUD A S,RADHIKA A. FPGA implementation of high speed 8-bit vedic multiplier using barrel shifter[C]//2013 International Conference on Energy Efficient Technologies for Sustainability.[S.l.]:IEEE,2013:14-17.

[15] CHEN P H,ZHAO J. High-speed parallel 32×32-b multiplier using a radix-16 Booth encoder[C]//2009 Third International Symposium on Intelligent Information Technology Application Workshops.[S.l.]:IEEE,2009:406-409.

[16] 王宏伟.滑动离散傅里叶算法输出稳定性研究[J].电波科学学报,2012,27(4):773-779,796.

(责任编辑 张 镅)