ISSN 1004-9037,CODEN SCYCE4
Journal of Data Acquisition and Processing Vol. 34,No. 2,Mar. 2019,pp. 358－366
DOI:10. 16337/j. 1004-9037. 2019. 02. 019

http://sjcj. nuaa. edu. cn
E-mail:sjcj@nuaa. edu. cn
Tel/Fax:＋86-025-84892742

应用于UWB系统的低硬件开销128点FFT处理器设计

于　建[1]　赵晃柱[2]

(1. 河北民族师范学院物理与电子工程系，承德，067000;2. 韩国圆光大学,益山，54538)

摘　要：快速傅里叶变换(Fast Fourier transform，FFT)处理器是数字信号处理领域的核心单元。本文针对超宽带(Ultra wideband，UWB)系统提出了一种低硬件开销的128点FFT处理器设计方案。此方案在算法上采用了混合基－2^4－2^3算法,硬件实现上采用了单路延迟负反馈(Single delay feedback，SDF)流水线架构,在处理复数乘法运算上,提出一种新型串接正则有符号数(Canonical signed digit，CSD)常数乘法器替代常用布斯乘法器对旋转因子W_{128}^{i}的复数乘法运算进行实现,大幅降低了FFT处理器消耗的硬件资源。本文设计基于QUARTUS PRIME平台进行开发,并搭配Cyclone 10 LP系列器件,编译报告显示本文方案对比于其他已存在的方案,具有最低的硬件开销和功耗。

关键词：傅里叶变换;混合基算法;CSD常数乘法器;布斯乘法器;流水线架构

中图分类号：TN47　　**文献标志码：**A

Design of Low Hardware-Cost 128-Point Fast Fourier Transform Processor for UWB System

Yu Jian[1], Cho Kyungju[2]

(1. Department of Physics and Electronic Engineering, Hebei Normal University for Nationalities, Chengde, 067000, China; 2. Wonkwang University, Iksan, 54538, South Korea)

Abstract: Fast Fourier transform (FFT) is a key block in the field of digital signal processing (DSP). A low hardware-cost 128-point FFT for UWB system is presented in this paper. Mixed radix－2^4－2^3 algorithm is adopted, and single-path delay feedback (SDF) architecture is used for hardware implementation. A novel cascade canonical signed digit (CSD) multiplier is proposed for the complex multiplication of W_{128}^{i} instead of the common booth multiplier, which can significantly reduce the hardware-cost. Based on QUARTUS PRIME tool with Cyclone 10 LP, the proposed scheme is developed, and the compilation report shows that the proposed scheme has the least hardware-cost and power consumption compared with the existing schemes.

Key words: fast Fourier transform (FFT); mixed radix algorithm; canonical signed digit (CSD) constant multiplier; Booth multiplier; pipelined architecture

引　　言

超宽带技术广泛应用于短距离高速的数据传输。本文介绍了一种低硬件开销的128点FFT处理

基金项目：河北民族师范学院科研课题(PT201507)资助项目。
收稿日期：2018-03-17;**修订日期：**2018-12-24

器方案用于超宽带系统。FFT处理器作为超宽带系统中关键单元模块,消耗着相当大的硬件资源。因此,如何降低FFT单元模块所占用的硬件资源成为近年来的研究热点。

基－2算法作为最著名的FFT算法,由于其简单的蝶形架构应用于FFT模块的硬件实现,可以降低硬件资源的开销,但是随着FFT点数的增加,它所需要的复杂乘法运算量会变得相当巨大[1]。随后,基－2^2,基－2^3,和基－2^4相继被提出,以上这些算法被统称为基－2^k算法[2-4]。基－2^k算法具有与基－2算法一样简单的蝶形架构,同时又能够大大降低旋转因子(Twiddle factor, TF)的计算量,因此,非常适合FFT的硬件实现。不过,随着k值的增加基－2^k算法的优势变得越来越小,这是由于所能利用的对称子项变得越来越少[5]。而且,高k值的基－2^k算法可以用低k值的混合基－2^k算法替代,表1为512点混合基－2^4－2^3算法和改良基－2^5算法[6]的详细旋转因子分布。仔细观察表1的旋转因子序列,容易发现如果以W_{512}为轴,将两边的旋转因子互换位置,就会得到同样旋转因子序列的分布。因此,本文的设计方案只考虑当$k \leqslant 4$时的基－2^k算法。

表1　512点混合基－2^4－2^3算法和改良基－2^5算法旋转因子序列分布

Tab. 1　Sequence distribution of 512-point FFT twiddle factor for mixed radix －2^4－2^3 and modified radix －2^5

算法	旋转因子序列							
	1	2	3	4	5	6	7	8
基－2^5[6]	$-j$	W_8	W_{32}	$-j$	W_{512}	$-j$	W_{16}	$-j$
基－2^4－2^3	$-j$	W_{16}	$-j$	W_{512}	$-j$	W_8	W_{32}	$-j$

在以往的研究过程中,不同的FFT架构被提出。在这些架构中,流水线架构由于其较高的吞吐量以及适中的硬件成本得到了广泛的应用。流水线FFT处理器架构一般分为两类:多路径延迟转换(Multi-path delay commutator, MDC)和单路径延迟反馈(Single-path delay feedback, SDF)[7]。MDC架构同时支持M路并行输入数据,数据吞吐量是SDF架构的M倍,但是其对数据路径、FFT点数以及基－2^k算法都有限制。另外,MDC架构中所需的存储器和复杂乘法器都比SDF架构多。所以,MDC架构能完成较高数据吞吐率,而SDF架构需要较少的存储器和硬件成本。为了获取更低的硬件开销,本文的设计方案使用SDF架构。

一般来说,对于N点FFT($N > 64$)都会采用布斯乘法器来处理序列与旋转因子W_N^i的复数乘法运算。本文提出了一种新型串接CSD常数乘法器来实现序列与W_{128}^i的运算,一方面能够进一步降低硬件资源的开销,另一方面无需任何只读存储器(Read only memory, ROM)对旋转因子常数值进行存储。

1　算法选择

利用基－2^k算法实现128点FFT包括6个旋转因子序列,表2给出了针对基－2^k算法的详细旋转因子序列分布以及复数乘法的运算量。其中$-j$的运算为普通运算,只需对输入序列实部和虚部的位置进行交换,再对虚部求反即可。

复数乘法运算量可由下列公式进行计算

$$2^{s-k}\left(2^k-1\right)\left(N/2^s-1\right)-2^{s-k} \tag{1}$$

$$3N/2^n\left(2^{n-2}-1\right)-N/2^n \tag{2}$$

式中:N代表FFT点数;s代表旋转因子所在的阶段;k代表基2^k算法的指数;n代表基2^k算法所合并出来常数旋转因子W_{constant}的指数,例如,基2^3算法合并出来的W_8,其n值为3($2^n=8 \rightarrow n=3$)。

表2　128点基－2^k算法旋转因子序列分布

Tab. 2　Sequence distribution of 128-point FFT twiddle factor for radix －2^k

算法	旋转因子序列						运算量
	1	2	3	4	5	6	
基－2	W_{128}	W_{64}	W_{32}	W_{16}	W_8	$-j$	270
基－2^2	$-j$	W_{128}	$-j$	W_{32}	$-j$	W_8	206
基－2^3	$-j$	W_8	W_{128}	$-j$	W_8	W_{16}	216
基－2^4－2^3	$-j$	W_{16}	$-j$	W_{128}	$-j$	W_8	200

以计算基2^3算法128点FFT为例，其中W_8为算法合并出来的W_{constant}，需要利用式(2)计算它的复数乘法运算量，而W_{128}和W_{16}为普通旋转因子，其复数乘法运算量用式(1)来计算。因此W_8的复数乘法运算量为32，W_{128}的复数乘法运算量为104，W_{16}的复数乘法运算量48。从表2可以看出，混合基-2^4-2^3算法拥有最简单的旋转因子序列，同时复数乘法的运算量也是最少的。因此，本文的设计方案采用混合基-2^4-2^3算法。

2 设计方案

图1所示为混合基-2^4-2^3算法的128点SDF流水线结构图。图中⊗代表复数乘法运算；BFI和BFII代表两种类型的蝶形运算单元；CLK是整个架构的控制时钟，它由7位计数器产生。在整个设计过程中，复数乘法运算都采用CSD常数乘法器来实现。

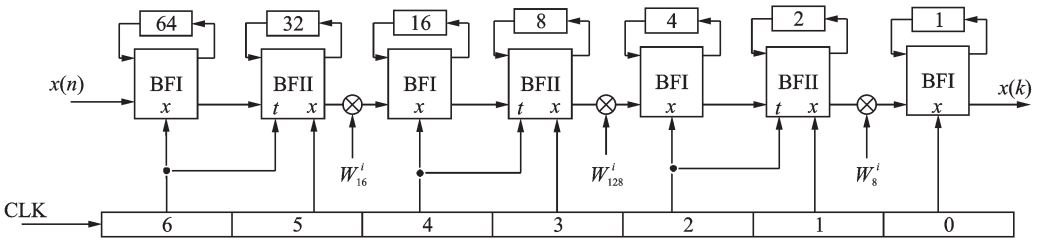


图1 混合基-2^4-2^3算法128点SDF结构图

Fig.1 Block diagram of 128-point FFT with mixed radix -2^4-2^3 algorithm

2.1 蝶形单元

图2所示为两种类型的蝶形单元详细架构，作为FFT处理器中必不可少的部分，主要用来执行复杂的加法和减法操作，图2中的$x_r(n)$和$x_r(n+N/2)$对应与输入复数序列实部，而$x_i(n)$和$x_i(n+N/2)$则对应输入复数序列的虚部。$Z_r(n)$，$Z_r(n+N/2)$，$Z_i(n)$和$Z_i(n+N/2)$对应于输出复数序列的实部和虚部。值得注意的是，I型蝶形单元对输入序列只进行简单的复数加减法运算，而II型蝶形单元除了需要进行必要的复数加减法运算以外，还需要额外的“$-j$”运算。因此，在硬件实现上比I型蝶形单元多出了选择单元与相应的控制逻辑电路。

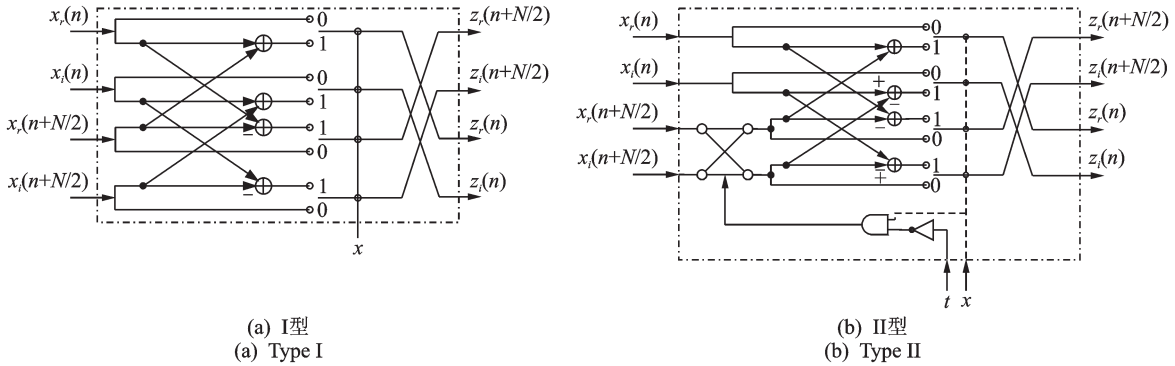


图2 蝶形单元架构

Fig.2 Block diagram of butterfly structure

2.2 W_8^i-CSD常数乘法器单元

实现输入序列与旋转因子W_8^i的复数乘法运算，只需要考虑3个旋转因子常数值W_8^1，W_8^2，W_8^3，由

于 $W_8^2=-j$,它的复数乘法运算由蝶形单元即可完成,而 W_8^3 等于 $-jW_8^1$。因此,只需知道常数值 W_8^1 即可。基于最小化硬件资源消耗的考虑,采用CSD表示法以及子模块共享方案来降低硬件成本。为了保持一定的量化信噪比,将旋转因子参数设置为12位字长。如果继续提高旋转因子参数的字长,量化信噪比并无显著的提高,反而会大大增加硬件开销[1]。图3为计算旋转因子 W_8^i 的12位字长的CSD表示,椭圆圈起的部分为子表达式共享模块"101"。图4所示为 W_8^i-CSD常数乘法器架构图,映射电路只需2个2选1数据选择器就可获得相应正确的结果。编译报告显示 W_8^i-CSD常数乘法器资源消耗只有布斯乘法器的11%。

$\mathrm{Re}\{W_8^i\}$	1	0	$\bar{1}$	0	$\bar{1}$	0	1	0	1	0	0	0
$\mathrm{Im}\{W_8^i\}$	1	0	$\bar{1}$	0	$\bar{1}$	0	1	0	1	0	0	0

图3　12位字长 W_8^i 的CSD表示

Fig. 3　CSD representation for W_8^i with 12-bit word-length

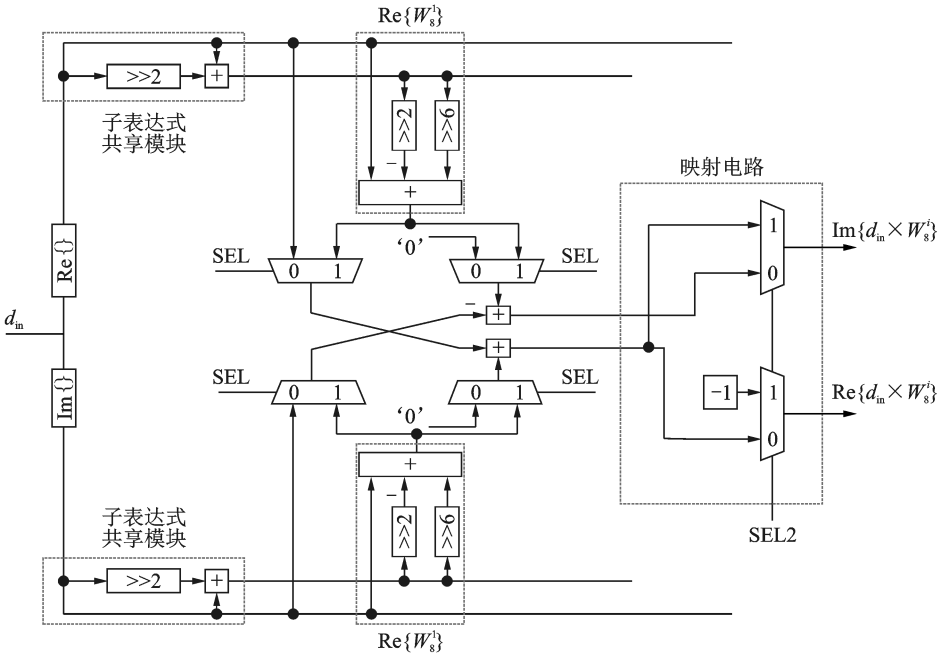


图4　W_8^i-CSD常数乘法器架构

Fig.4　Structure of W_8^i-CSD constant multiplier

2.3　W_{16}^i-CSD常数乘法器单元

对于输入序列与旋转因子 W_{16}^i 的乘法运算,总共需要考虑7个旋转因子常数值:W_{16}^0, W_{16}^1, W_{16}^2, W_{16}^3, W_{16}^4, W_{16}^6, 和 W_{16}^9。其中 W_{16}^0 和 W_{16}^4 分别等于1和 $-j$,它们的复数乘法运算通过蝶形单元就可以完成。而对于剩下的旋转因子而言,只需知道3个相关数值:0.923 9, 0.382 7和0.707 1即可。这3个数值正好对应于 W_{16}^1, W_{16}^2, 和 W_{16}^3 3个复数的实部数值 $\mathrm{Re}\{W_{16}^1\}$, $\mathrm{Re}\{W_{16}^2\}$ 和 $\mathrm{Re}\{W_{16}^3\}$。表3所示为所有7个旋转因子常数值与 $\mathrm{Re}\{W_{16}^1\}$, $\mathrm{Re}\{W_{16}^2\}$ 和 $\mathrm{Re}\{W_{16}^3\}$ 关系式。

表3　旋转因子 W_{16}^i 的7组常数值

Tab. 3　Sven constant values of twiddle factor W_{16}^i

常数值	关系式
W_{16}^0	1
W_{16}^1	$\mathrm{Re}\{W_{16}^1\}-j\mathrm{Re}\{W_{16}^3\}$
W_{16}^2	$\mathrm{Re}\{W_{16}^2\}-j\mathrm{Re}\{W_{16}^2\}$
W_{16}^3	$\mathrm{Re}\{W_{16}^3\}-j\mathrm{Re}\{W_{16}^1\}$
W_{16}^4	$-j(=-j\times W_{16}^0)$
W_{16}^6	$-\mathrm{Re}\{W_{16}^2\}-j\mathrm{Re}\{W_{16}^2\}(=-j\times W_{16}^2)$
W_{16}^9	$-\mathrm{Re}\{W_{16}^1\}+j\mathrm{Re}\{W_{16}^3\}(=-1\times W_{16}^1)$

图5所示为旋转因子计算 W_{16}^{i} 的12位字长的CSD表示，其中椭圆圈起的“101”和“$\bar{1}0\bar{1}$”部分为子表达式共享模块。图6所示为 W_{16}^{i} 常数乘法器详细的架构图，从图中可知，选择恰当的控制信号“SEL”和“SEL2”，便可以得到输入序列与旋转因子 W_{16}^{i} 计算后的结果。编译报告显示，W_{16}^{i}-CSD常数乘法器只有布斯乘法器的28%。由此可见，随着CSD常数乘法器所需旋转因子常数个数的增加，乘法器所要消耗的资源也会不断提高。

$\mathrm{Re}\{W_{16}^{1}\}$	1	0	0	0	1	0	$\bar{1}$	0	0	1	0	0
$\mathrm{Re}\{W_{16}^{1}\}$	1	0	$\bar{1}$	0	1	0	1	0	$\bar{1}$	0	0	0
$\mathrm{Re}\{W_{16}^{1}\}$	0	1	0	$\bar{1}$	0	0	0	1	0	0	0	$\bar{1}$

图5 12位字长 W_{16}^{i} 的CSD表示

Fig. 5 CSD representation for W_{16}^{i} with 12-bit word-length

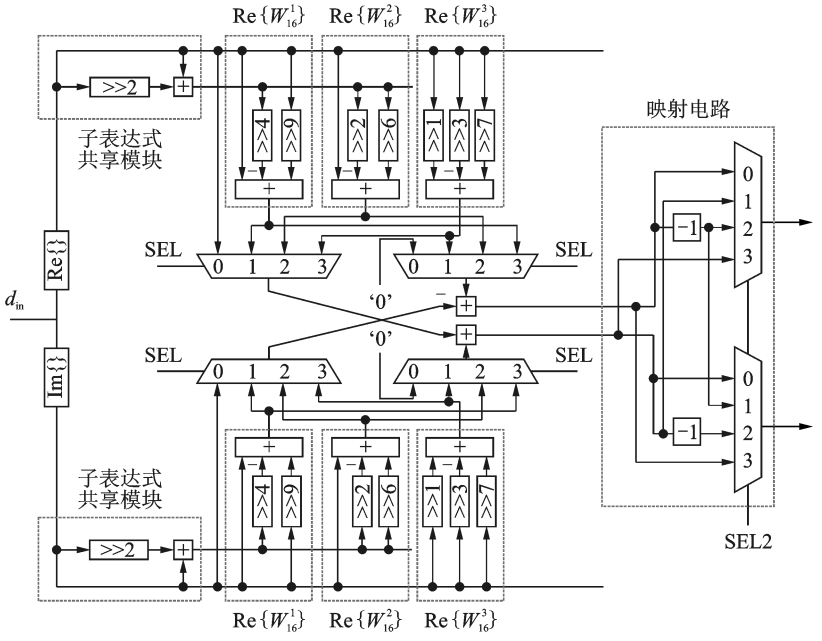
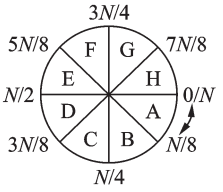


图6 W_{16}^{i}-CSD常数乘法器架构

Fig.6 Structure of W_{16}^{i}-CSD constant multiplier

2.4 新型 W_{128}^{i} 串接CSD常数乘法器单元

图7所示为旋转因子的1/8特性，根据此特性可将旋转因子的常数值的个数降低为原来的1/8。对于 W_{N}^{i} 来说，利用此特性，所需旋转因子常数值的个数仅仅为 $N/8$。



Region	Real	Imaginary	Expression
A	x_p	$-y_p$	x_p+jy_p
B	$-y_p$	$-x_p$	$-y_p-jx_p$
C	y_p	$-x_p$	y_p-jx_p
D	$-x_p$	y_p	$-x_p+jy_p$
E	$-x_p$	$-y_p$	$-x_p-jy_p$
F	y_p	x_p	y_p+jx_p
G	$-y_p$	x_p	$-y_p+jx_p$
H	x_p	$-y_p$	x_p-jy_p

图7 旋转因子对称性映射图

Fig.7 Symmetric region mapping of twiddle factor

一般来说，当旋转因子常数值的个数过多，直接利用CSD常数乘法器处理输入序列与旋转因子的复数乘法运算往往比普通布斯乘法器在硬件资源消耗上更高。因此，为了减少旋转因子常数值个数，提出了新型串接CSD常数乘法器的方案。串接CSD常数乘法器将复数乘法运算拆解成两阶段复数乘法运算，以达到降低旋转因子常数值个数的目的，从而降低乘法

器的资源消耗。虽然串接CSD常数乘法器能够减少旋转因子常数值的个数,但是仅仅局限于完成输入序列、旋转因子与 W_{128}^{i}, W_{256}^{i} 和 W_{512}^{i} 的复数乘法运算。

以本文设计对象128点串接CSD乘法器为例,首先,利用1/8对称特性,将旋转因子常数值的个数降低到16个。然后,对旋转因子的指数 i 进行分解,分解方案为 $i=4i_1+i_2$, $i_1=1\sim4$, $i_2=0\sim3$,旋转因子常数值的个数由16个减少为8个。如果要实现输入序列与 W_{128}^{13} 进行复数乘法运算,令 $i_1=3$, $i_2=1$($W_{128}^{13}=W_{128}^{4\times3}W_{128}^{1}$),输入序列先与 W_{128}^{12} 进行复数乘法运算,再与 W_{128}^{1} 进行复数乘法运算,得到最终的输出结果。图8所示为计算旋转因子 W_{128}^{i} 12位字长的CSD表示,其中椭圆圈起的部分是3组子表达式共享模块:椭圆圈起的"101",椭圆圈起的"$10\bar{1}$"和椭圆圈起的"$1000\bar{1}$"。

i_1	$\mathrm{Re}\{W_{128}^{4i_1}\}$												$-\mathrm{Im}\{W_{128}^{4i_1}\}$											
1	1	0	0	0	0	0	$\bar{1}$	0	1	0	0	0	0	0	1	0	$\bar{1}$	0	0	1	0	0	0	1
2	1	0	0	0	$\bar{1}$	0	1	0	0	1	0	0	0	1	0	$\bar{1}$	0	0	0	1	0	0	0	1
3	1	0	$\bar{1}$	0	1	0	1	0	1	0	1	0	0	1	0	0	1	0	0	1	0	0	0	1
4	1	0	$\bar{1}$	0	$\bar{1}$	0	1	0	1	0	0	0	1	0	$\bar{1}$	0	$\bar{1}$	0	1	0	1	0	0	0
i_2	$\mathrm{Re}\{W_{128}^{i_2}\}$												$-\mathrm{Im}\{W_{128}^{i_2}\}$											
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	1	0	$\bar{1}$	0	0	1	0	0
2	1	0	0	0	0	0	0	0	$\bar{1}$	0	$\bar{1}$	0	0	0	0	1	0	1	0	0	1	0	0	0
3	1	0	0	0	0	0	$\bar{1}$	0	1	0	0	1	0	0	0	1	0	$\bar{1}$	0	$\bar{1}$	0	0	0	0

图8　12位字长 W_{128}^{i} 的CSD表示

Fig.8　CSD representation for W_{128}^{i} with 12-bit word-length

图9(a)所示为 W_{128}^{i} 串接CSD乘法器的详细架构图,其中子表达式共享模块由加法单元和位移单元实现,长方形的箱体模块为右移单元,通过简单的硬件连接被放置在乘法器单元恰当的位置,10个4选1数据选择器为选择正确的输出结果。为了减少关键路径(Critical path, CP),在两阶段CSD乘法器单元中间插入了流水线寄存器,如图9(b)所示。因此,FFT处理器的计算时间大大缩短,能够满足实时处理的需求。利用映射电路模块获得其他区域(如图7所示B~H区域)的旋转因子与输入序列相乘的结果。编译报告显示,本文提出的新型串接128点CSD乘法器硬件资源消耗只有普通布斯乘法器的59%,而且无需ROM对旋转因子的常数值进行存储,进一步降低了硬件资源的消耗。

3　结果与比较

本文方案基于QUARTUS PRIME工具软件进行开发,使用Verilog语言进行设计。表4所示为不同方案128点FFT计算量的比较,包括了布斯乘法器使用量,CSD常数乘法器使用量,复数加法器使用量,计算时间以及延迟的比较。本文的设计方案计算时间为 $6T_{\mathrm{A}}+3T_{\mathrm{MUX}}+2T_{\mathrm{N}}$,表4中 T_{A} 代表加法单元运算所需时间, T_{MUX} 代表选择器单元运算所需要时间, T_{N} 代表取补码运算所需时间, T_{M} 代表乘法单元运算所需时间。由于本文的设计方案只利用CSD常数乘法器来完成复数乘法运算,因此只需考虑加法单元,取补码单元以及选择单元的计算时间。由于乘法单元计算所需时间要高于加法单元计算所需时间。因此,本文方案的计算时间同其他方案一样能够满足128点FFT处理器实时处理的需求。表5所示为不同设计方案逻辑单元使用量,寄存器使用量,记忆体单元使用量和动态功耗的比较。设计的

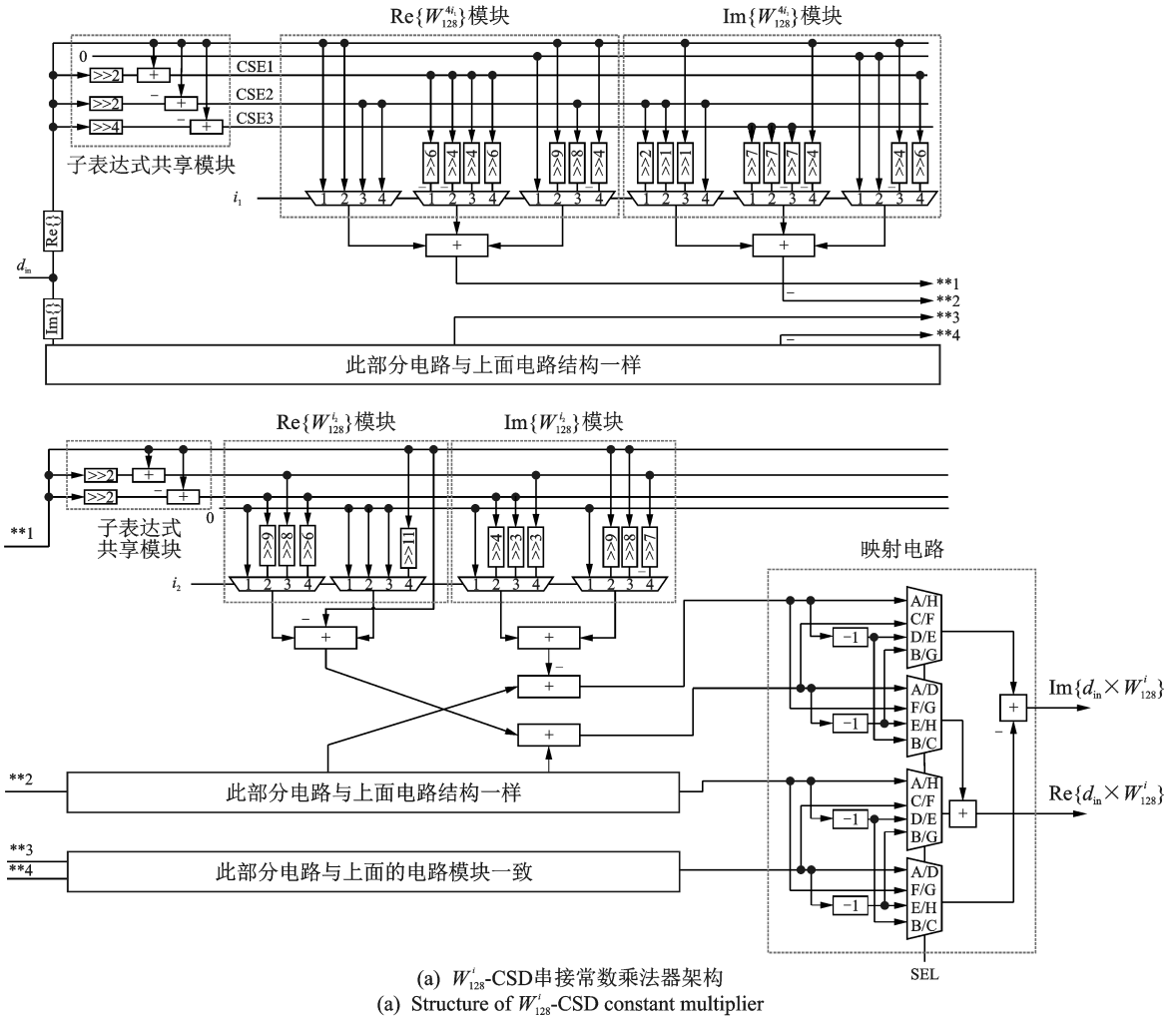


图9 128点串接CSD常数乘法器详细架构

Fig.9 Detailed structure of 128-point cascade CSD constant multiplier

输入字长设置为12位,输出字长设置为22位,器件家族选择的是Cyclone 10 LP,功率评估等级(Power estimation confidence,PEC)为高。编译报告显示,对比于其他方案[8-11],本文的设计方案至少可节省逻辑单元使用量40%,寄存器使用量3%,记忆体单元的使用量14%。同时,基于本文设计方案的128点FFT处理器的动态功耗仅仅为5.59 mW。图10所示为基于MODELSIM RTL级仿真结果(输入序列的实部和虚部设定为1~128),与MATLAB计算的结果一致,证实了本设计方案的有效性。

表4　不同方案128点FFT计算量比较

Tab. 4　Comparison of various schemes computation for 128-point FFT

算法与架构	布斯乘法器	CSD常数乘法器	复数加法器	计算时间	延迟
$R8$MDF[8]	4	8	16	$T_M+5T_A+T_{MUX}+2T_N$	127
$R2^4$SDF[9]	1	2	14	$T_M+3T_A+3T_{MUX}+2T_N$	255
$R2^4$MDF[10]	12	0	168	$T_M+3T_A+T_{MUX}$	255
$R2^2$SDF[11]	1	2	14	$T_M+3T_A+3T_{MUX}+2T_N$	255
本文方案	0	3	14	$6T_A+3T_{MUX}+2T_N$	255

表5　不同方案综合结果比较

Tab. 5　Performance of the proposed scheme compared with previous implementation

算法与架构	$R8$MDF[8]	$R2^4$SDF[9]	$R2^4$MDF[10]	$R2^2$SDF[11]	本文方案($R2^42^3$SDF)
乘法器	布斯&CSD	布斯&CSD	布斯	布斯&CSD	CSD
逻辑单元	9876(1)	6859(0.69)	12368(1.25)	7047(0.71)	2983(0.30)
寄存器	789(1)	440(0.56)	879(1.11)	447(0.57)	429(0.54)
记忆体单元	7623(1)	3756(0.49)	7786(1.02)	3986(0.52)	2926(0.38)
动态功耗/mW	11.26	7.13	13.76	7.72	5.59

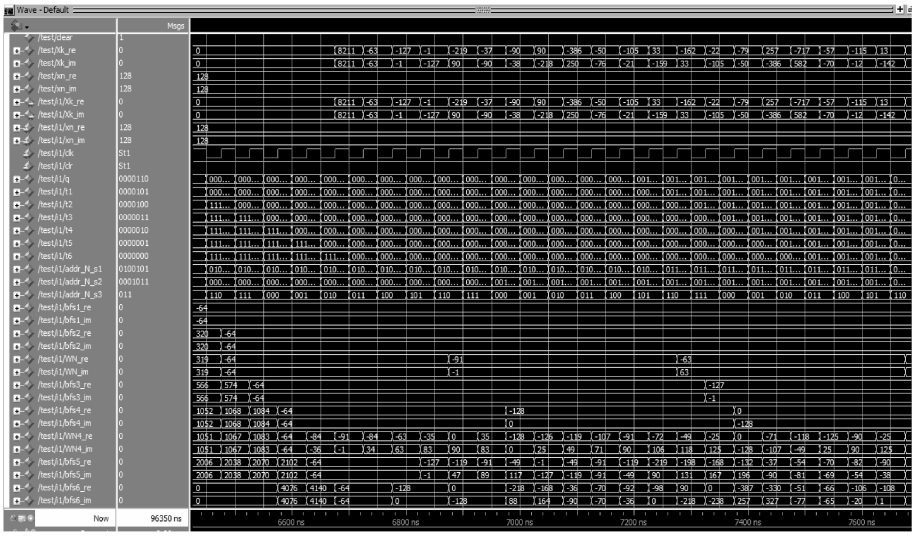


图10　基于本文方案的128点FFT处理器RTL级仿真结果

Fig.10　RTL level simulation result of 128-point FFT processor based on proposed scheme

4　结束语

本文针对于128点FFT处理器的设计,算法上采用了混合基-2^4-2^3算法,硬件实现上采用了SDF架构,新型串接CSD常数乘法器替代传统的布斯乘法器完成与旋转因子W_{128}^{i}的复数乘法运算,使得整个FFT处理器的复数乘法运算均由CSD常数乘法器完成,大幅降低了硬件资源消耗和功耗。同时,给出了复数乘法运算量的计算公式,让设计者可以更容易地选择适合的基-2^k算法对FFT处理器

进行设计。编译报告的结果显示本文的设计方案对比于已存在的设计方案在实现128点FFT上有较大优势。

参考文献：

[1] Ganjikunta G K, Sahoo S K. An area-efficient and low-power 64-point pipeline fast Fourier transform for OFDM applications [J]. Integration the Vlsi Journal, 2017, 57: 125-131.

[2] He S, Torkelson M. A new approach to pipeline FFT processor[C]//Proceedings of IPPS′96. [S.l.]: IEEE, 1996: 766-770.

[3] Jung Y, Yoon H, Kim J. New efficient FFT algorithm and pipeline implementation results for OFDM/DMT applications[J]. IEEE Transactions on Consumer Electronics, 2003, 49(1): 14-20.

[4] Oh J Y, Lim M S. New radix-2 to the 4th power pipeline FFT processor[J]. IEICE Transactions on Electronics, 2005, 88(8): 1740-1746.

[5] Yang C H, Yu T H, Markovic D. Power and area minimization of reconfigurable FFT processors: A 3GPP-LTE example[J]. IEEE Journal of Solid-State Circuits, 2012, 47(3): 757-768.

[6] Cho T, Lee H. A high-speed low-complexity modified radix-2^5 FFT processor for high rate WPAN applications[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2013, 21(1): 187-191.

[7] Yang K J, Tsai S H, Chuang G C H. MDC FFT/IFFT processor with variable length for MIMO-OFDM systems[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2013, 21(4): 720-731.

[8] Lin Y W, Liu H Y, Lee C Y. A 1-GS/s FFT/IFFT processor for UWB applications[J]. IEEE Journal of Solid-State Circuits, 2005, 40(8): 1726-1735.

[9] Yang C H, Yu T H, Markovic D. Power and area minimization of reconfigurable FFT processors: A 3GPP-LTE example[J]. IEEE Journal of Solid-State Circuits, 2012, 47(3): 757-768.

[10] Liu H, Lee H. A high performance four-parallel 128/64-point radix-24 FFT/IFFT processor for MIMO-OFDM systems[C]//APCCAS 2008. [S.l.]: IEEE, 2009: 834-837.

[11] Yu C. A128/512/1024/2048-point pipeline FFT/IFFT architecture for mobile WiMAX[C]//Proc 2nd IEEE Global Conf Consumer Electron. [S.l.]: IEEE, 2013: 243-244.

作者简介：

于建(1979-)，男，博士研究生，研究方向：超大规模集成电路设计，E-mail：mjyujian1979@hotmail.com。

赵炅柱，男，博士生导师，韩国益山人，研究方向：超大规模集成电路的架构与算法，E-mail：kjcho@wku.ac.kr。

（编辑：夏道家）