An Inexact Multiplier based Area-Efficient Fast Fourier Transform Processor

Xiang Yin[1,a], Hao Xiao[2,b,*], Weiwei Cao[3] and Shu Chen[3]

[1]College of Electronic and Information Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China

[2]School of Microelectronics, HeFei University of Technology, Hefei 230009, China [3]Shanghai Institute of Spaceflight Control Technology, Shanghai 201109, China

[a]manxia101@163.com, [b]xiaohao@hfut.edu.cn

Keywords: Truncated multiplier, Inexact 4-2 Compressor, Fast Fourier Transform.

Abstract. Fast Fourier Transform (FFT) and its inverse transform (IFFT) are commonly used tools for digital signal processing. They are widely used in various communication systems, especially the communication system based on orthogonal frequency division multiplexing (OFDM) technology. Reducing the cost of multiplier area can effectively spare the area of FFT because multipliers take up the mainly cost of the hardware. To solve the above problems, inexact multiplier for FFT, with lower area cost and low SQNR loss compared to a conventional exact FFT processor, is proposed. Comparison results show that the proposed FFT processor has good property for applications like DVB-T, LTE.

一种基于非精确乘法器的低成本快速傅里叶变换处理器

印象[1, a]，肖昊[2,b,*]，曹卫卫[3]，陈纾[3]

[1]南京航空航天大学电子信息工程学院，南京，江苏，中国

[2]合肥工业大学微电子学院，合肥，安徽，中国

[3]上海航天控制技术研究所，上海，中国

[a]manxia101@163.com, [b]xiaohao@hfut.edu.cn

关键词: 截尾乘法器；非精确4-2压缩；快速傅里叶变换

中文摘要. 快速傅里叶变换(FFT)及其反变换(IFFT)是数字信号处理中常用的工具，广泛应用于各种通信系统，特别是基于正交频分复用(OFDM)技术的通信系统。乘法器是FFT处理器中占用硬件成本较大的部分，因此乘法器的设计对减少FFT处理的面积和功耗至关重要。本文提出了一种使用非精确乘法器的低成本FFT处理器，能够有效减少FFT处理器的面积，并且可以保持较高精度，满足DVB-T、LTE等应用的性能要求。

1. 引言

传统的电路设计大多运用精确计算算法，但实际应用中如图像、信号处理等，具有一定的容错性，这些应用并不需要绝对的精确性。所以将非精确运算运用到这些系统中可以减少能量损耗及电路面积，同时也可提高系统的性能[1]。非精确的算法的主要思想是通过减小硬件的复杂度来提高系统的性能及效率。离散傅里叶变换(DFT)是数字信号处理领域最常用的处

理方法之一。库利(Cooley)和图基(Tukey)在1965年提出了一种实现DFT的非常有效的算法——快速傅里叶变换(FFT)的算法[2]。乘法器是FFT处理器中占用面积和功耗较大的部分，因此减少FFT处理器中乘法器的面积与数量可以有效降低FFT处理器的面积。

文献[3]设计了一种低功耗非精确FFT处理器，该设计综合使用物理层与设计层的非精确设计实现非精确FFT处理器；文献[4]设计了一种非精确的浮点数加法器，该设计通过将低位部分精确加法替代为或门算法实现非精确运算。文献[5]设计了一种基于非精确压缩的乘法器，该设计通过修改部分积压缩真值表实现低成本的非精确乘法器。

但是以上非精确设计应用在FFT处理器中时，或者损失的精度较多，或者节省的面积较少，因此本文设计了一种截尾部分积，并综合使用多种非精确设计的非精确乘法器，基于此非精确乘法器的低成本FFT处理器设计，能够有效减少FFT处理器的成本，并且保持较高精度，其性能满足DVB-T、LTE等以OFDM技术为核心的通信应用的性能要求。

本文结构如下：第2节介绍一种非精确的4-2压缩器，第3节则介绍了一种截尾乘法器，第4节介绍了本文的非精确FFT设计与其他同类型设计的性能比较，第5节总结了该设计方法。

2．非精确4-2压缩器

如图1所示为13位乘13位的基-2Booth编码乘法器。由图1可知，Booth编码为重叠的3 bit一组，Booth编码与部分积一一对应，每个部分积与相邻上一行部分积相比都左移两位。一般的，部分积的个数由n减少到(n + 1)/ 2，n为乘数N的长度，例如图1中13位乘13位乘法器的部分积数量由13减少到7。如图1中虚线处所

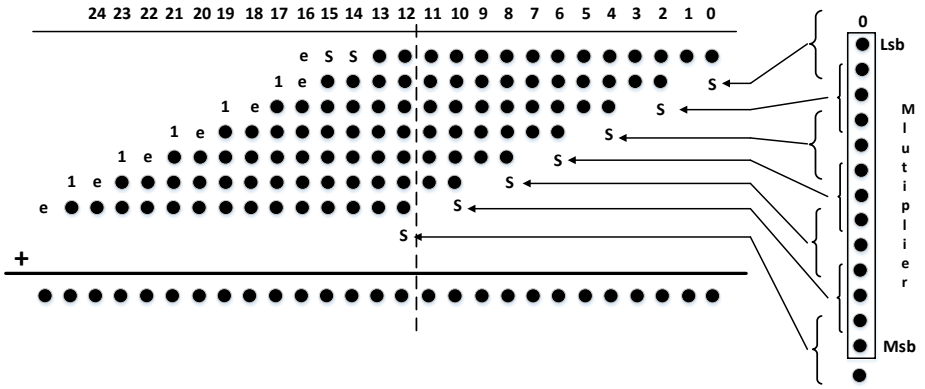
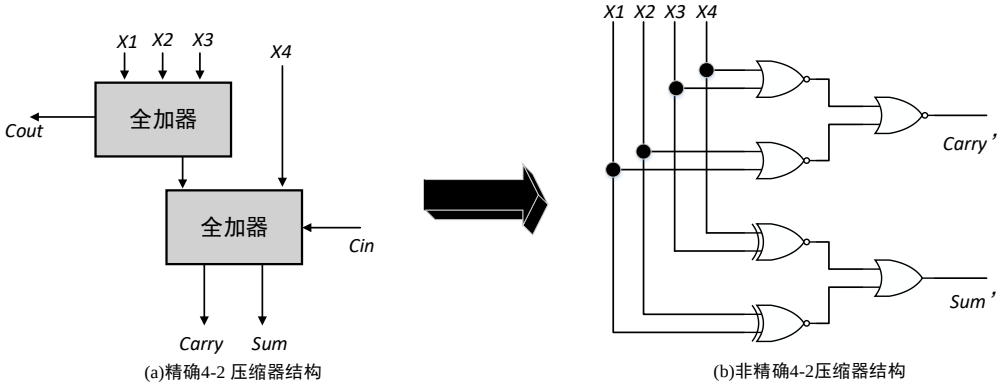


图1　13位乘13位基-2Booth编码乘法器

示，最终标准化的乘积将截去低(n-1)位，以节省硬件面积，例如图1中的13位乘13位乘法器需要截去低12位（即第0位至第11位）。

在Daada树[9]与Wallace树中，精确4-2压缩器和全加器、半加器一样都是常见的部分积压缩方法，如图2(a)所示，精确4-2压缩器实际上是2个全加器的级联。除了压缩和Sum与进位位Cout，4-2压缩会产生另一种进位——Carry，Carry与Cout的权重相同。



(a)精确4-2 压缩器结构　　(b)非精确4-2压缩器结构

图2 精确4-2压缩与非精确4-2压缩

公式(1)、(2)、(3)为4-2压缩器的输出逻辑表达式：

$$Sum = x1 \oplus x2 \oplus x3 \oplus x4 \oplus Cin \tag{1}$$

$$Cout = (x1 \oplus x2)x3 + \overline{(x1 \oplus x2)}x1 \tag{2}$$

$$Carry = (x1 \oplus x2 \oplus x3 \oplus x4)Cin + (x1 \oplus x2 \oplus x3 \oplus x4)x4 \tag{3}$$

2.1 非精确4-2压缩方式1

如表1所示，精确4-2压缩真值表中共有的32个状态，其中24个状态输出carry的值与输入Cin相同。利用此特性，在本设计中输出carry的值简化为Cin，即式(4)：

$$Carry' = Cin \tag{4}$$

因为输出carry拥有更高的权重，例如，如果输入为“01001”（表1第10行），正确的输出应为“010”，值为2。但因简化carry等于Cin，非精确4-2压缩得到的
输出为“000”，值为0。因为此差值较大，需要通过简化Cout和Sum信号来补偿损失的精度。通过对比，将表1中精确4-2压缩下半部分的输出Sum全部简化为0值会减小非精确压缩与精确压缩之间的误差，同时也降低了压缩器的硬件复杂度，即式(5)：

$$Sum' = \overline{Cin}(\overline{x1 \oplus x2} + \overline{x3 \oplus x4}) \tag{5}$$

另外，输出Sum的简化也会减少对应的延迟以及本设计的整体延时（因为关键路径缩短）[5]。最后，改变部分Cout的值减少非精确carry与Sum计算产生的误差，同时设计也更简化，即式(6)：

$$Cout' = \overline{\overline{x1 + x2} + \overline{x3 + x4}} \tag{6}$$

表1也给出了非精确4-2压缩方式1的真值表以及非精确压缩方式1与精确压缩的总体输出的差值，非精确4-2压缩方式1共有12个错误输出（正确率为62.5%）。式(4)-(6)即为非精确4-2压缩方式1的逻辑表达式[6]。

表1 精确4-2压缩与非精确4-2压缩方式1真值表

Cin	*X4*	*X3*	*X2*	*X1*	精确			非精方式1				非精确方式2		
					Cout	*carry*	*sum*	*Cout*	*carry*	*sum*	误差	*carry*	*sum*	误差
0	0	0	0	0	0	0	0	0	0	1	1	0	1	1
0	0	0	0	1	0	0	1	0	0	1	0	0	1	1
0	0	0	1	0	0	0	1	0	0	1	0	0	1	0
0	0	0	1	1	1	0	0	0	0	1	-1	0	1	0
0	0	1	0	0	0	0	1	1	0	1	0	0	1	0
0	0	1	0	1	1	0	0	1	0	0	0	0	1	0
0	0	1	1	0	1	0	0	1	0	0	0	0	1	-1
0	0	1	1	1	1	0	1	0	0	1	0	0	1	-1
0	1	0	0	0	0	0	1	1	0	1	0	0	1	0
0	1	0	0	1	0	1	0	1	0	0	0	0	1	0
0	1	0	1	0	0	1	0	1	0	0	0	1	0	0
0	1	0	1	1	1	0	1	0	0	1	0	1	0	0
0	1	1	0	0	0	1	0	1	0	1	-1	1	0	0
0	1	1	0	1	1	0	1	1	0	1	0	1	0	0
0	1	1	1	0	1	0	1	1	0	1	0	1	1	0
0	1	1	1	1	1	1	0	0	0	1	-1	1	1	0
1	0	0	0	0	0	0	1	0	1	0	1	0	1	0
1	0	0	0	1	0	1	0	0	1	0	0	0	1	0
1	0	0	1	0	0	1	0	0	1	0	0	1	0	0
1	0	0	1	1	1	0	1	0	1	0	-1	1	0	0
1	0	1	0	0	0	1	0	1	1	0	0	1	0	0

1	0	1	0	1	1	0	1	1	1	0	1	1	0	0
1	0	1	1	0	1	0	1	1	1	0	1	1	1	0
1	0	1	1	1	1	1	0	0	1	0	0	1	1	0
1	1	0	0	0	0	1	0	0	1	0	0	0	1	-1
1	1	0	0	1	0	1	1	1	1	0	1	0	1	-1
1	1	0	1	0	0	1	1	1	1	0	1	1	1	0
1	1	0	1	1	1	1	0	1	1	0	0	1	1	0
1	1	1	0	0	0	1	1	0	1	0	-1	1	1	0
1	1	1	0	1	1	1	0	1	1	0	0	1	1	0
1	1	1	1	0	1	1	0	1	1	0	0	1	1	-1
1	1	1	1	1	1	1	1	1	1	0	-1	1	1	-1

2.2 非精确4-2压缩方式2

输出carry与输出Cout具有相同的权重，因此非精确4-2压缩中二者的输出表达式可互换,所以carry等于表达式(6)等号右边，Cout等于Cin；又因为在第一级进行4-2压缩时Cin值为0，因此Cout与Cin恒为0，在硬件设计中二者可以忽略。将这种非精确4-2压缩命名为非精确4-2压缩方式2。如图2(b)所示即为此非精确4-2压缩方式2的流程图，而非精确4-2压缩方式2的表达式为：

$$Sum' = (\overline{x1 \oplus x2} + \overline{x3 \oplus x4}) \tag{7}$$

$$Carry' = \overline{\overline{x1 + x2} + \overline{x3 + x4}} \tag{8}$$

在Cin=0时，式(8)与式(6)相同，式(7)与式(5)相同。如表2所示为精确4-2压缩和两种非精确4-2压缩的晶体管数量比较，非精确4-2压缩方式2需要的晶体管最少，精确4-2压缩需要的晶体管最多[5]。因为非精确4-2压缩方式2只有4个输入，2个输出，而精确4-2压缩有5个输入，3个输出，与预期相符。

表2 不同4-2压缩晶体管数量比较

	晶体管数量
精确4-2压缩	52
非精确4-2压缩方式1	44
非精确4-2压缩方式2	42

表1中也给出了非精确4-2压缩方式2的真值表以及出非精确压缩方式2与精确压缩的总体输出的差值。例如，当所有输入(X1-X4)为1时，正确的十进制输出应为4，但是此非精确4-2压缩方式2的输出Carry和Sum分别为1，所有对应的十进制输出为3，误差为1。

由表1可知，非精确4-2压缩方式2共有4个错误输出（正确率为75%）；且非精确4-2压缩方式2比非精确4-2压缩方式1使用的晶体管更少，因此在进行非精确4-2压缩时，非精确4-2压缩方式2性能更好。将使用此压缩方法的乘法器命名为非精确乘法器1。在13位乘13位乘法器中，非精确压缩结束后为保持位宽不变，把乘法器分为2级，第一级压缩所有部分积，第二级截去低12位。

3．非精确乘法器

3.1 截尾乘法器

如图3所示为一个截尾低8位的13位乘13位乘法器；因为截尾低八位部分积节省了原本压缩低8位所需的4-2压缩器、半加器和全加器，所以截尾乘法器能够减

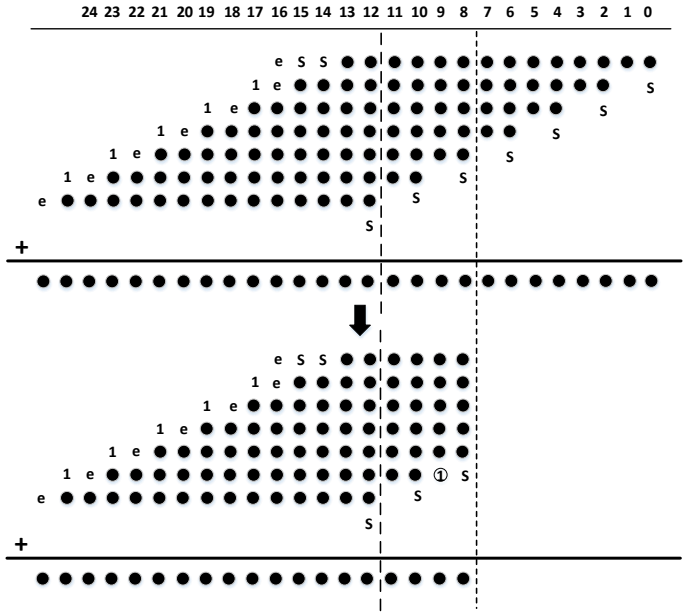
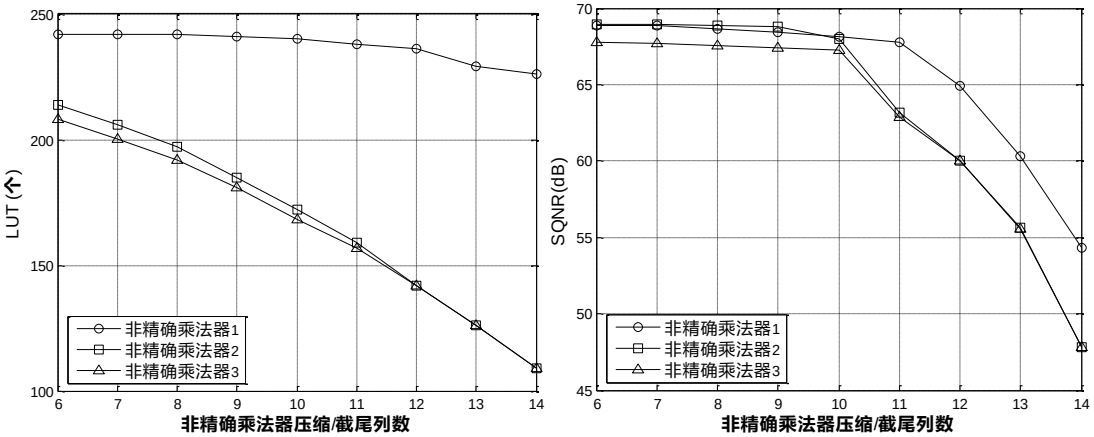


图3 截尾低8位13位乘13位乘法器

少乘法器的面积及功耗。为补偿在第一级截去低8位产生的精度损失，在第九列加“1”。这个“1” 是低八位的部分积值全为0与全为1的均值，以补偿因为截尾操作造成的精度损失。截尾的位数不同，补偿值的大小与位置不同。在第二级，为保持位宽不变截去低12位。将此截尾乘法器命名为非精确乘法器2。

使用Xilinx Zynq XC7Z020进行乘法器的面积测试，因乘法器为纯组合逻辑，所以LUT (Look-Up-Table)个数比即表示面积比。使用相同的随机数计算非精确乘法器1、2的精度，精度由SQNR表示，其表达式为式(9)。如图4(a)所示为非精确乘法器1、2、3的面积比较，图4(b)为非精确乘法器1、2、3的精度比较。

$$SNR = 10log_{10}\left(\frac{\sum\left\|sigal_{exact}\right\|^2}{\sum\left\|sigal_{exact} - sigal_{inexact}\right\|^2}\right) \tag{9}$$



(a) (b)

图4 非精确乘法器1、2、3的面积、精度比较

经实际测试可知，Xilinx Zynq XC7Z020上精确乘法器需要243个LUT；在第二级截去低12位后，平均SQNR为68.94。由图4(a)可知在第一级处理（非精确4-2压缩或截尾）相同位数的部分积时，非精确乘法器2比非精确乘法器1减小的面积更多，且处理位数越多，乘法器的面积越小；由图4(b)可知在第一级处理相同位数的部分积时，非精确乘法器1的精度略高于非精确乘法器2，且处理位数越多，这两种乘法器的精度降低得越明显。

3.2 非精确压缩截尾乘法器

由图4(a)(b)可知，非精确乘法器1在第一级压缩低12列时，减少了乘法器的面积，并且精度损失较小。因此，当n小于12时，在非精确乘法器2的第n列至第11列使用非精确乘法器1中的非精确压缩，当n大于等于12时非精确乘法器2不变，n为截尾列数；将此乘法器命名为非精确乘法器3。以图3非精确乘法器2为例，第0列至第7列使用截尾操作，第8列至第11列使用非精确4-2压缩。

图4(a)(b)也比较了非精确乘法器2、3的面积和精度。通过综合比较乘法器面积与精度，非精确乘法器3在第一级截尾低10列时表现最好。

4．低成本快速傅里叶变换的实现

由于算法复杂与硬件复杂的关系，FFT通常采用基-2^2与基-2^3结构。本文探讨在基-2^3的512点FFT处理器中，将精确的乘法器替换为第3节中介绍的非精确乘法器3。本文用Design Complier 进行逻辑综合，使用SMIC 130nm COMS工艺，时钟频率为300MHz。如表3所示为使用非精确乘法器3的基-2^3的512点FFT处理器与同类型设计的参数比较。由表3可知本文设计的使用非精确乘法器3的基-23的512点FFT处理器与精确的512点FFT处理器相比，平均SQNR仅损失1.25dB。

此外，本设计与精确512点FFT、文献[7]、文献[8]相比，核面积分别降低了

表3 使用非精确乘法器3的512点FFT处理器与同类设计的参数比较

	非精确FFT	精确FFT	[7]	[8]
FFT点数	512	512	512	256
工艺	130nm	130nm	90nm	90nm
结构	SDF	SDF	Memory-based	SDF
平均SQNR/dB	49.51	50.76	-	-
时钟/MHz	300	300	324	300
功耗/mW	128.2	132.6	103.5	119.7
核面积/mm^2	1.50	1.69	2.46	3.53

11.24%、39.02%、57.51%。结果表明，本文设计的使用非精确乘法器3的基-23的512点FFT处理器有效的减少了FFT处理器的成本，并保持了较高精度。

5．结论

本文设计了一种基于非精确乘法器的低成本FFT处理器设计，能够有效减少FFT处理器的成本，并且保持较高精度，其性能满足DVB-T、LTE等以OFDM技术为核心的通信应用的性能要求。

References

[1] K. Palem and A. Lingamneni, Ten years of building broken chips: the physics and engineering of inexact computing, *ACMTrans. Embedded Computing Systems,* vol.87, pp. 1-23, 2013.

[2] J. Cooley and J. Tukey, An algorithm for the machine calculation of complex Fourier series, *Mathematics of Computation,* vol.19, pp. 297-301, 1965.

[3] A. Lingamneni, C. Enz, et al., Highly energy-efficient and quality-tunable inexact FFT accelerators, *Custom Integrated Circuits Conference ,* vol.9, pp.141-153, 2014.

[4] W. Q. Liu, L. B. Chen, et al. Inexact Floating-Point Adder for Dynamic Image Processing, *Computers IEEE Transactions on,* vol. 65, pp. 308-314, 2016.

[5] A. Momeni , J. Han , et al. Design and Analysis of Approximate Compressors for Multiplication, *IEEE Transactions on Computers ,* vol.64, pp. 984-994, 2015.

[6] V. Gupta, D. Mohapatra, S. Park, et al. IMPACT: IMPrecise adders for low-power approximate computing, *Int. Symp. on Low Power Electron. And Design,* pp. 409–414, 2011.

[7] S.J. Huang and S.G. Chen, A green FFT processor with 2.5-GS/s for IEEE 802.15.3c (WPANs), *Proc. Int. Conf. Green Circuits Syst.,* pp. 9–13, 2010

[8] Y. Chen, Y. W. Lin, et al. A 2.4-G sample/s DVFS FFT processor for MIMO OFDM communication systems, *IEEE J.Solid-State Circuits,* vol.43, pp.1260–1273, 2008.

[9] L. Dadda. Multiple addition of binary serial numbers, *1978 IEEE 4th Symposium onomputer Arithmetic (ARITH),* pp.140-148, 1978.