引用格式：徐礼晗，景佳，许丁鸿，等. 基于无冲突访存规则的高性能FFT处理器的设计与实现[J]. 微电子学与计算机，2021，38(1)：57-63. [XU L H, JING J, XU D H, et al. Design and implementation of high performance FFT processor based on conflict-free access rules[J]. Microelectronics & Computer, 2021, 38(1): 57-63.]

基于无冲突访存规则的高性能FFT处理器的设计与实现

徐礼晗[1]，景 佳[2]，许丁鸿[1]，宋宇鲲[1]

(1 合肥工业大学 微电子设计研究所 教育部IC设计网上合作研究中心，安徽 合肥 230009；
2 合肥工业大学 电子科学与应用物理学院，安徽 合肥 230009)

摘 要：高性能快速傅里叶变换(FFT)处理器在雷达与通信等实时信号处理系统中具有广泛的应用场景. 本文通过优化无冲突访存规则，结合基－2和基－8时域抽取的FFT算法，设计了一种高性能的混合基FFT处理器. 该处理器采用基于存储的架构，主要组成单元包括蝶形运算单元，存储单元和控制单元. 通过优化无冲突访存规则以及旋转因子生成方案，提高计算速度和硬件效率。在SMIC 40nm标准CMOS工艺下，FFT处理器工作主频超过500 MHz，核心面积为1.76×0.85 mm^2，且计算结果信噪比超过136 dB. 在32K点FFT计算任务下，计算速度相比同类型FFT处理器提高约4倍.

关键词：快速傅里叶变换；基-8无冲突访存规则；混合基；ASIC

中图分类号：TN47 **文献标识码**：A **文章编号**：1000－7180(2021)01－0057－07

DOI:10.19304/j.cnki.issn1000-7180.2021.01.011

Design and implementation of high performance FFT processor based on conflict-free access rules

XU Li-han[1], JING Jia[2], XU Ding-hong[1], SONG Yu-kun[1]

(1 Institute of VLSI Design IC Design Web-cooperation Research Center of MOE, Hefei University of Technology, Hefei 230009, China;
2 School of Electronic Science & Applied Physics, Hefei University of Technology, Hefei 230009, China)

Abstract: The high-performance Fast Fourier Transform (FFT) processor has a wide range of application scenarios in real-time signal processingsystems such as radar and communication. In this paper, a high performance FFT processor is designed by optimizing the conflict-free access rules and combining the radix-2 and radix-8 DIT-FFT. The processor employs a Memory-Based architecture, including a butterfly operator unit, a memory unit, and a control unit. The calculation speed and hardware efficiency are improvedImprove calculation speed and hardware efficiency by optimizing Conflict-Free access rules and twiddle factor generation schemes. The proposed processor is implemented on SMIC 40 nm CMOS technology. Simulation results show that the processor can work at over 500 MHz, the core area is1.76×0.85mm^2, and the SNR is over136dB or more. Under the 32K-point FFT calculation task, the calculation speed is about 4 times higher than that of the same type of FFT processor.

Key words: FFT; Radix-8 conflict-free access rules; mixed radix; ASIC

收稿日期：2020－04－19；**修回日期**：2020－05－10

基金项目：国家自然科学基金(61874156)；国家重点研发计划(2018YFB2202604)

1 引言

离散傅里叶变换(Discrete Fourier Transform，DFT)是信号处理中最基本的运算，许多相关算法都可以由 DFT 实现. 1965 年提出的快速傅里叶变换(Fast Fourier Transform，FFT)能够显著加速 DFT 计算过程，使其更加适合计算机运算. 随信号处理精度和速度要求不断提高，新的 FFT 算法和实现方法层出不穷. 常见算法有基-2、基-4 算法、分裂基算法、WFTA 算法等[1-2]，前两者已经满足不了现代雷达、通信等系统对于长序列计算任务的速度要求[3]；而 WFTA 等算法存在编程困难的缺陷；高基算法不但在长序列任务上有速度优势，而且运算过程规则，所以高基算法已是长序列 FFT 处理器的主流算法. FFT 处理器通常分为流水线结构[4]和存储结构[5]两类，后者硬件效率更高. 理论上增加蝶形运算器的数量可以提高存储结构 FFT 处理器的性能，但是多蝶形运算器多数据通道并行访问存储单元的效率将成为制约性能提高的瓶颈[6]. 因此，为保证整机的实时性能，研究并设计出高效的多蝶形运算器并行的 FFT 处理器成为关键.

本文设计的 FFT 处理器复用可重构的多蝶形运算器实现基-2 FFT 和基-8 FFT 算法，通过优化的无冲突访存规则确保整机性能达到对应蝶形运算器数量的理论值，其中基-2 FFT 算法用于实现对更多序列的覆盖，基-8 FFT 算法用于长序列的计算. 最后，本文在 SMIC 40 nm 工艺下完成了所设计 FFT 处理器的综合和后端实现工作.

2 算法原理

由文献[1]知，基-2 FFT 蝶形运算公式如下：

$$\begin{cases} X_0(k) = X'_0(k) + X'_1(k) \times Wk_N \\ X_1(k) = X'_0(k) - X'_1(k) \times Wk_N \end{cases} \tag{1}$$

式中：$Wk_N = e^{-j\frac{2\pi k}{N}}$，$(k = 0,1,2,\cdots\cdots,\frac{N}{2}-1)$. 单路基-2 蝶形单元有 2 输入 2 输出，本文设计中采用 8 路并行基-2 蝶形单元结构，即为 16 输入 16 输出. 基-2 FFT 算法比其它算法具有更宽的序列覆盖率.

由文献[7-8]可知，基-8 FFT 运算公式如公式(2)所示，单路基 8 蝶形单元为 8 输入 8 输出. 基-8 FFT 算法因其在长序列计算上迭代次数少，所以在长序列计算上具有显著的速度优势.

$$\begin{cases} X_0(k) = A + BW_N^k + CW_N^{2k} + DW_N^{3k} + EW_N^{4k} + FW_N^{5k} + GW_N^{6k} + HW_N^{7k} \\ X_1(k) = A + BW_N^k W_8^1 - jCW_N^{2k} + DW_N^{3k} W_8^3 - EW_N^{4k} + FW_N^{5k} W_8^3 + jGW_N^{6k} + HW_N^{7k} W_8^7 \\ X_2(k) = A - BW_N^k - CW_N^{2k} + jDW_N^{3k} + EW_N^{4k} - jFW_N^{5k} - GW_N^{6k} + HW_N^{7k} \\ X_3(k) = A + BW_N^k W_8^3 + jCW_N^{2k} + jDW_N^{3k} W_8^1 - EW_N^{4k} + FW_N^{5k} W_8^7 - jGW_N^{6k} + HW_N^{7k} W_8^5 \\ X_4(k) = A - BW_N^k + CW_N^{2k} - DW_N^{3k} + EW_N^{4k} - FW_N^{5k} + GW_N^{6k} - HW_N^{7k} \\ X_5(k) = A + BW_N^k W_8^5 - jCW_N^{2k} + DW_N^{3k} W_8^7 - EW_N^{4k} + FW_N^{5k} W_8^1 + GW_N^{6k} + HW_N^{7k} W_8^7 \\ X_6(k) = A + jBW_N^k - CW_N^{2k} - DW_N^{3k} + EW_N^{4k} + jFW_N^{5k} - GW_N^{6k} - HW_N^{7k} \\ X_7(k) = A + BW_N^k W_8^7 + CW_N^{2k} + DW_N^{3k} W_8^5 - EW_N^{4k} + FW_N^{5k} W_8^3 - GW_N^{6k} + HW_N^{7k} W_8^1 \end{cases} \tag{2}$$

式(2)中：$W_8^1 = \frac{\sqrt{2}}{2}(1-j)$，$W_8^3 = -\frac{\sqrt{2}}{2}(1+j)$，$W_8^5 = -\frac{\sqrt{2}}{2}(1-j)$，$W_8^7 = \frac{\sqrt{2}}{2}(1+j)$，$W_N^k = e^{-\frac{2\pi k}{N}}$，$\left(k = 0,1,2,\cdots\cdots,\frac{N}{8}-1\right)$. $A = X'_0(k)$，$B = X'_1(k)$，$C = X'_2(k)$，$D = X'_3(k)$，$E = X'_4(k)$，$F = X'_5(k)$，$G = X'_6(k)$，$H = X'_7(k)$. $X'_i(k)$ 和 $X_i(k)(i = 0,1,2,3,4,5,6,7)$ 分别为蝶形单元输入操作数和输出结果数.

由式(2)推导出时域抽取的基-8 FFT 运算的蝶形运算流程，如图 1 所示. 单路基-8 蝶形运算器需 7 组复数乘法器，本文在虚线框①内增加一组复数乘法器，使复数乘法器资源同时满足构建 8 路基-2 和单路基-8 蝶形运算器的要求，且虚线框②③④内的复数加法器可以全部复用，达到了资源利用率最大化. 复数乘法器复用部分如图 1 中虚线框①，复数加法器复用部分如图 1 中虚线框②③④.

3 系统结构

本文设计的 FFT 处理器结构图如图 2 所示，包括：(1)控制单元(control unit)是整个系统的工作核心，解析配置信息，调节各模块工作状态；(2)接口单元(Interface Unit)负责 FFT 处理器与外部配置信息和数据的交互；(3)可重构运算单元(Reconfigurable Operator，Re-Operator)实现在基-2 模式和基-

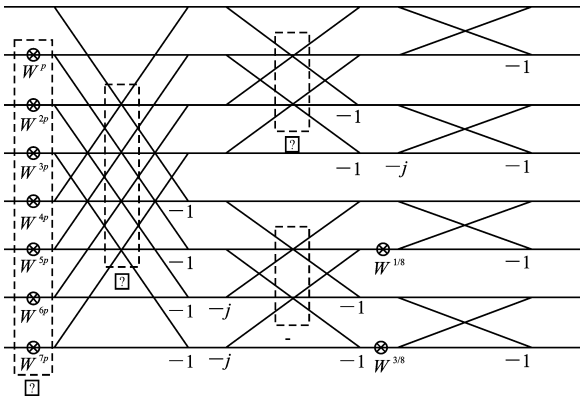

图 1　基 8 DIT-FFT 蝶形运算图

8 模式下相对应的蝶形运算；(4)旋转因子生成单元(Twiddle Factor Generator)生成每次蝶形运算所需的旋转因子；(5)操作数地址生成单元(AGU_operand)生成运算器所需操作数的读地址；(6)结果数地址生成单元(AGU_result)负责生成运算器输出结果数的写地址；(7)存储管理单元(Memory Management Unit)实现地址无冲突规则，管理存储单元的访存方向；(8)存储单元(Mem Unit)由多块存储块组成，负责存储源数据，中间过程数据和结果数据.

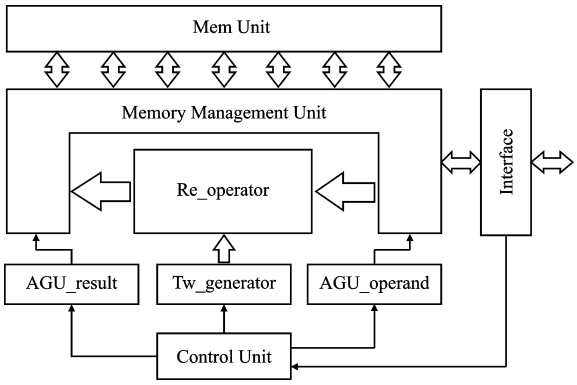

图 2　FFT 处理器结构框图

FFT 处理器的工作流程如下：配置信息和源数据由接口单元分别送入控制单元和存储管理单元. 控制单元根据配置信息生成相关信号下发到各个模块中. 操作数从 Mem Unit 中取出送入 Re_operator，旋转因子由 Tw_generator 生成送入 Re_operator 中参与计算，结果数送回 Mem Unit，此时完成一级 FFT 计算. 当 1 k≤序列长度≤16 k 时，Re_operator 被配置成 8 路基-2 蝶形运算器；当 16 k＜序列长度≤32 k 时，Re_operator 被配置成单路基-8 蝶形运算器.

本文设计可重构蝶形运算器满足基-2 FFT 和基-8 FFT 计算，优化基-8 无冲突地址规则与旋转因子生成方法，提升计算性能.

3.1　蝶形运算器设计

本文设计蝶形运算单元可以配置为 8 路并行基 2 蝶形运算单元，或配置为单路基-8 蝶形运算单元. 由公式(1)(2)可发现，基-8 蝶形运算单元可以视为 3 级 8 路基 2 蝶形运算单元级联而成，所以可在基-8 蝶形运算单元中构建基-2 蝶形运算单元. 通过复用复数乘法器和复数加法器，以提高硬件效率.

图 3 为部分蝶形运算器结构，此处可配置为 2 路基-2 蝶形运算，如图 3 中虚线路径，或者配置为 1/8 的单路基-8 蝶形运算器. 对于 ± 1，$\pm j$ 等系数由系数处理部分通过变换符号位和交换实部虚部方法实现，从而减少硬件资源消耗.

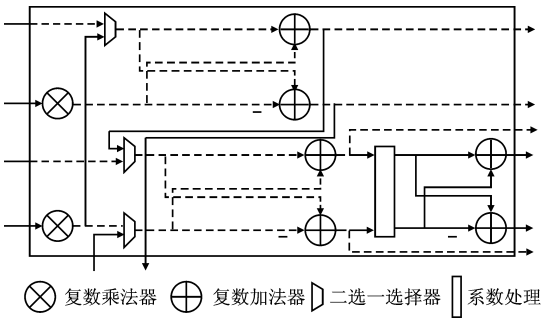

图 3　蝶形单元结构

3.2　基-8 无冲突访存规则

对于基-8 DIT-FFT 算法，蝶形运算器支持 8 输入 8 输出. 为了实现预期性能，必须确保计算过程中蝶形器内数据流水线不被破坏，就需要设计一套高效率的基-8 访存规则，实现每个时钟周期内并行读写 8 组数据，达到存储器的访存效率最大化. 本文设计基于以往的设计与研究[9-10]，优化了基-8 FFT 算法的无冲突访存规则.

访存效率最大的关键是将每级计算所需数据分布在不同的存储块中. 基-8 DIT-FFT 算法数据抽取间隔为 8^{M-L}，其中 M 为总级数，L 为当前计算级数. 因为每次数据抽取的间隔基底为 8，所以在每次数据读取之前，所需的计算数据须写入 8 块并行的存储器中，每级计算数据可以从 8 块不同的存储器中读出，即可完成无冲突访存. 通过变换第 n 级结果数据的写规律和第 $n+1$ 级操作数据的读规律，简化命中存储块和获得存储块内地址的方法，实现对无冲突访存规则的优化.

将存储模块分为 8 组存储器，源数据通过一条数据通道存入存储器中. 存储块循环系统根据源数据的输入长度进行每组存储块的移动. 如图 4 所示，当数据长度＝$N/8$ 时，N 为 FFT 总点数，存储块循环系统循环一次，即存储块 7 移动到存储块 0 的位置，存储块 0 移动到存储块 1 的位置，以此类推. 本次操作将第一级计算抽取的数据分布在不同的 8 块存储块中，如 32 k 点 FFT 计算，第一级抽取间隔为 4 096，所以相邻源数据存储块之间的数据间隔为 4 096.

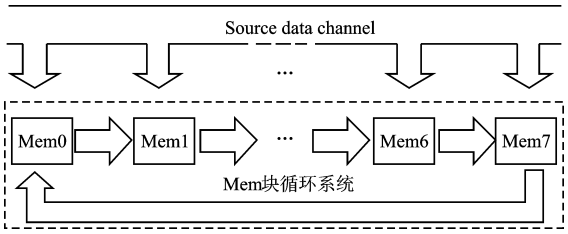

图 4 源数据输入存储块循环

当蝶形运算器读取数据时，存储块循环系统每输出 1 组数据就循环一次，即存储块 1 移动到存储块 0 的位置，存储块 2 移动到存储块 1 的位置，以此类推. 类似的，当蝶形运算器计算完毕写回存储单元时，存储块循环系统每输入 $N/16$ 组数据就循环一次，即存储块 1 移动到存储块 0 的位置，存储块 2 移动到存储块 1 的位置，以此类推. 此时的存储块循环系统的循环方向相与源数据输入时的循环方向相反，如图 5 所示. 如 32 k 点 FFT 计算，每级抽取的间隔为(4 096，512，64，8，1)，即每级相邻的存储块中数据间隔为(4 096，512，64，8，1).

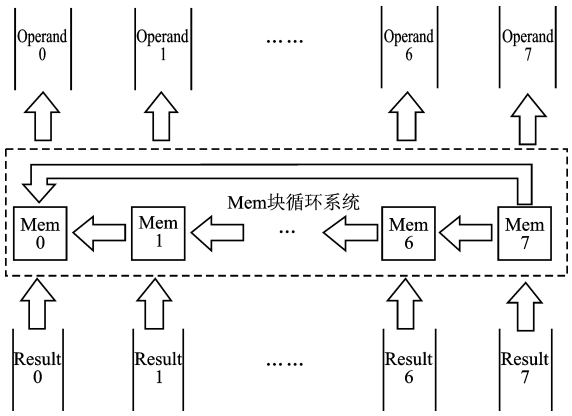

图 5 计算过程中存储块循环

在本文设计中，8 块存储块对应 8 组识别信号，每组识别信号的位宽为 3 位，由 8 组识别信号组成 24 位宽的寄存器 F，其初始值 F＝{3'b000，3'b001，…，3'b111}. 初始情况下，8 条数据通道对应存储块 0 到存储块 7. 每次循环后对应的存储块由 F 寄存器的值来确定. 数据存储地址根据计数器 cnt 确定，计数器 cnt 位宽 $m=\log_2 N$，并截掉低 3 位得到存储块内地址 addr，相关伪码如图 6 所示.

```
for (i=1,i<=N,i++)
    if (i=N/8*n) // n=0,1,2,...
        F=(F>>3)|(F<<21);
    end if
    cnt=cnt+1;
    addr=cnt[m-1:3];
end for
```

(a) 源数据输入阶段伪码

```
for (i=1, i<=N/8, i++)
    F=(F<<3)|(F>>21);
    cnt=cnt+1;
    addr 0=cnt[m-1:3];
    addr 1=(cnt+N/8)[m-1:3];
    addr 2=(cnt+N/4)[m-1:3];
    II II
    addr 2=(cnt+7N/8)[m-1:3];
end for
```

(b) 读操作数阶段伪码

```
for (i=1, i<=N/8, i++)
    if (i=N/16*n)/n=0,1,2,...
        F=(F<<3)|(F>>21);
    end if
    cnt=cnt+8;
    addr 0=cnt[m-1:3];
    addr 1=(cnt+1)[m-1:3];
    II II
    addr 7=(cnt+7)[m-1:3];
end for
```

(c) 写结果数阶段伪码

图 6 伪码

3.3 旋转因子生成

理论上 FFT 计算所需旋转因子个数与序列长度成正比. 对于长序列的 FFT 计算不仅要考虑数据和旋转因子存储问题，还要考虑旋转因子对计算精度和计算速度的影响. 使用 CORDIC 等算法实时生成旋转因子的方法降低了旋转因子对存储空间的需求，但是过长的旋转因子生成流水线将严重影响 FFT 计算的速度[11]. 存储完整旋转因子的方法有着精度高的优点，但是该方法所需的大量旋转因子存储空间将影响整个系统的面积和功耗，所以并不适合长序列 FFT 算法. 为了减少旋转因子的存储空间，并且保持可接受的计算精度. 本文采用旋转因子

压缩方法[12]. 该方法利用旋转因子的可约性和对称性，将旋转因子拆分成两子式相乘的结构. 如下式所述：

$$W_N^k = W_N^{(\alpha+\beta)} = W_N^\alpha \times W_N^\beta \tag{3}$$

将 $32K$ 点的旋转因子进行分解，令：

$$K = 256\alpha + \beta \tag{4}$$

变换之前 k 的范围为[0，327 67]，变换后令 $\alpha \in [0,127]$，$\beta \in [0,255]$. 将式(4) 带入式(3) 中：

$$\begin{aligned} W_{32\,768}^k &= W_{32\,768}^{256\alpha+\beta} \\ &= W_{128\times256}^{256\alpha} \times W_{128\times256}^{\beta} \\ &= W_{128}^{\alpha} \times W_{128\times256}^{\beta} \end{aligned} \tag{5}$$

根据基-2 FFT 算法可知 16 k 点 FFT 计算所需旋转因子应覆盖 $W_{16\,\mathrm{k}}^{8\mathrm{K}-1}$ 以下系数，32 k 点 FFT 计算所需旋转因子应覆盖 $W_{32\,\mathrm{k}}^{4095\times7}$ 以下系数. 为了同时满足基-2 FFT 和基-8 FFT 所需旋转因子，本模块将 32 k 点旋转因子进行压缩，既可以满足 32 k 点的基-8 FFT 计算，利用旋转因子的周期性和对称性也可以满足 1 k 点到 16 k 点的基-2 FFT 计算. 本文设计采用的 IEEE-754 标准单精度浮点数[14]，根据式(5)，两类旋转因子子式所需的存储空间分别为 256×64 bit 和 128×64 bit，该方法将 32 k×64 bit 的旋转因子存储空间压缩到 384×64 bit，相比存储所有旋转因子的方法节省 98.8%的存储空间. 本文设计中采用双端口 ROM 存储旋转因子子式，部署了 4 组 8 块双端口 ROM. 每次计算生成 8 路基 2 FFT 计算所需的 8 组旋转因子，或者生成单路基 8 FFT 计算所需的 8 组旋转因子.

本文设计使用初值和步进值累加的方式生成无冲突规则的旋转因子地址，再根据旋转因子压缩方法拆分得到两子式在 ROM 中地址. 通过多组计数器将生成过程拆分成 M 个阶段，如 $(C_{M-1}C_{M-2}\cdots C_1 C_0)$，其中 C 为模 8 的计数器，M 为基-8 FFT 计算的总级数. 接收到启动信号后，计数器 C_0 开始计数. 每完成一级 FFT 计算，所有计数器清零. 根据计数器组选择不同的初值，而每级 FFT 计算过程中的步进值是固定的. 根据上述旋转因子压缩方法，将 T_{addr}^w 按照高低位拆分成两部分. 如 32 k 点旋转因子的存储地址 T_{addr}^w，得到旋转因子子式存储地址 $T_{\mathrm{addr}}^w[14:7]$ 和 $T_{\mathrm{addr}}^w[6:0]$. 将从 ROM 中得到的两组子式送入复数乘法器中生成所需旋转因子. 图 7 为旋转因子生成单元结构图. 将使用上述方法生成的旋转因子与 MATLAB 生成的旋转因子作误差分析得到表 1. 由表 1 可知，此方法可以满足实时计算的精度要求.

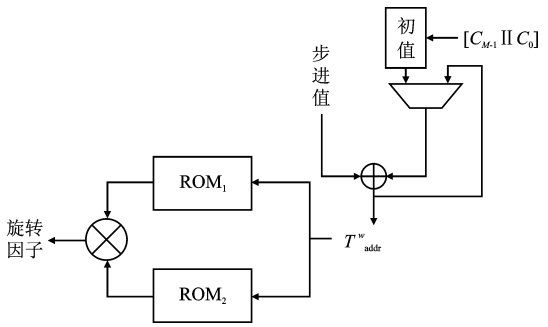

图 7　旋转因子生成单元

表 1　旋转因子误差分析

SNR	平均误差	最大误差
147.210 7	1.714 3e-08	1.192 1e-07

4　设计实现与性能分析

本文设计由 Verilog HDL 完成 RTL 级代码，按照标准 ASIC 流程，使用 Synopsys 公司 Design Compiler 和 IC Compiler 在 SMIC 40 nm 工艺下完成综合与实现，PVT 参数为(SS，0.99V，125℃). 该设计可以工作在 500 MHz 以上，核心面积为 1.76×0.85 mm^2，版图如图 8 所示.

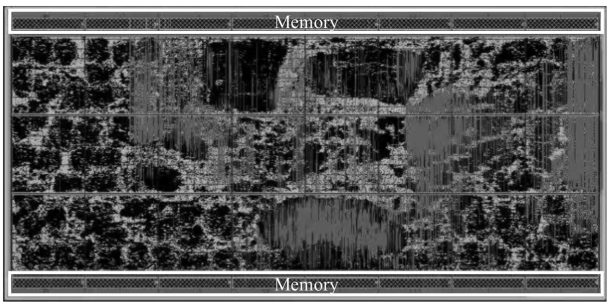

图 8　Layout 图

本文设计通过 Xilinx Virtex-7 FPGA 进行验证，将 FPGA 验证结果与 MATLAB 中 FFT 函数的计算结果进行比较，输入序列生成采用单频正弦信号叠加随机噪声，$x(t) = \sin(2\pi \times 100 \times t \times 1/n) + 5 \times \mathrm{randn}(\mathrm{size}(t))$，采样间隔为 $1/n$，n 为总采样点数，总采样时间为 1 秒. 非 2 的整数次幂计算点数，采用尾部补零的方法将计算点数补充为 2 的整数次幂. 部分点数的误差分析如表 2 所示.

表 2 误差分析

序列长度		SNR	平均误差	最大误差
1 k	实部	138.13	1.0110e-5	4.5776e-5
	虚部	137.94	1.0750e-5	45776e-5
2 k	实部	137.35	1.5911e-5	6.8665e-5
	虚部	137.27	1.6534e-5	1.2207e-4
8 k	实部	136.27	3.7476e-5	2.4414e-4
	虚部	136.56	3.7565e-5	2.4414e-4
16 k	实部	136.09	5.6936e-5	3.6621e-4
	虚部	136.00	5.6542e-5	4.8828e-4
32k	实部	136.25	7.5934e-5	4.8828e-4
	虚部	136.33	7.6528e-5	9.7656e-4

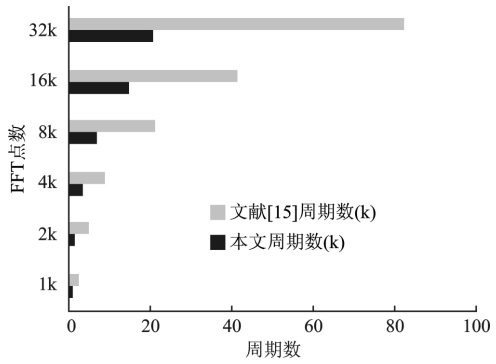

图 9 FFT 计算时间

根据验证结果给出运算时间，如图 9 所示. 与文献[14]进行对比，同为 40 nm 工艺下，工作频率为 500 MHz，本文设计在计算速度上提升了 4 倍. 在 32 k 计算任务上，与文献[15]对比，计算速度提升 2.4 倍.

5 结束语

本文给出了一种高速高精度的基-2/8 DIT-FFT 混合结构 FFT 处理器设计与实现，可以完成 IEEE-754 标准单精度浮点数的 32k 点以下 FFT/IFFT 计算，并在 SMIC 40 nm 标准 CMOS 工艺下进行设计与实现. 实验结果表明，本文所设计的 FFT 处理器可以工作频率高于 500 MHz，SNR 在 136 dB 以上，计算速度优于同类型的 FFT 处理器[14-15]. 本文的设计通过优化无冲突访存规则、旋转因子的生成规则，压缩旋转因子存储空间，提高了计算速度和硬件效率，完全满足实时数字信号处理系统的要求.

参考文献：

[1] COOLEY J W, TUKEY J W. An algorithm for the machine calculation of complex Fourier series[J]. Mathematics of Computation, 1965, 19(90): 297-301. DOI:10.1090/S0025-5718-1965-0178586-1.

[2] HASAN M, ARSLAN T, THOMPSON J S. A novel coefficient ordering based low power pipelined radix-4 FFT processor for wireless LAN applications[J]. IEEE Transactions on Consumer Electronics, 2003, 49(1): 128-134. DOI:10.1109/TCE.2003.1205465.

[3] 宋玮，李如玮，代栋敏. 基于 FPGA 的基 8-FFT 处理器设计[J]. 科技导报，2010，28(16)：67-70.
SONG W, LI R W, DAI D M. Radix-8 FFT processor design based on FPGA[J]. Science & Technology Review, 2010, 28(16): 67-70.

[4] WANG Z K, LIU X, HE B S, et al. A combined SDC-SDF architecture for normal I/O pipelined radix-2 FFT[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2015, 23(5): 973-977. DOI:10.1109/TVLSI.2014.2319335.

[5] LIU S H, LIU D K. 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, 27(3): 511-523. DOI:10.1109/TVLSI.2018.2879675.

[6] XIA K F, WU B, XIONG T, et al. A memory-based FFT processor design with generalized efficient conflict-free address schemes[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2017, 25 (6): 1919-1929. DOI: 10.1109/TVLSI.2017.2666820.

[7] SUN M X, TIAN L Y, DAI D M. Radix-8 FFT processor design based on FPGA[C]//Proceedings of the 2012 5th International Congress on Image and Signal Processing. Chongqing: IEEE, 2012: 1453-1457. DOI:10.1109/CISP.2012.6469786.

[8] JIANG R M. An area-efficient FFT architecture for OFDM digital video broadcasting[J]. IEEE Transactions on Consumer Electronics, 2007, 53(4): 1322-1326. DOI:10.1109/TCE.2007.4429219.

[9] 王江，黑勇，郑晓燕，等. 基于无冲突地址生成的高性能 FFT 处理器设计[J]. 微电子学与计算机，2007，24(3)：15-19. DOI:10.3969/j.issn.1000-7180.2007.03.004.
WANG J, HEI Y, ZHENG X Y, et al. Design of high performance FFT processor with conflict free memory access[J]. Microelectronics & Computer, 2007, 24(3): 15-19. DOI:10.3969/j.issn.1000-7180.

2007.03.004.

[10] YU J Y, HUANG D, LI X, etal. Conflict-free architecture for multi-butterfly parallel processing in-place Radix-r FFT[C]//Proceedings of the 2016 IEEE 13th International Conference on Signal Processing (ICSP). Chengdu: IEEE, 2016. DOI: 10.1109/ICSP.2016.7877884.

[11] 蔚接锁. 基于 FPGA 与流水线 CORDIC 算法的 FFT 处理器的实现[D]. 天津：天津大学, 2009.
WEI J S. The implementation of a FFT processor-based on FPGA and CORDIC algorithm[D]. Tianjin: Tianjin University, 2009.

[12] ZHANGD L, HUANG L, SONG Y K, et al. Design and implementation of 1-D and 2-D mixed architecture FFT processor in heterogeneous multi-core SoC based on FPGA[J]. International Journal of Control and Automation, 2014,7(6): 177-188. DOI:10.14257/ijca.2014.7.6.18.

[13] STEVENSON D. 754-1985-IEEE standard for binary floating-point arithmetic[S]. New York: IEEE, 1985. DOI:10.1109/IEEESTD.1985.82928.

[14] HAN F, LI L, WANG K, et al. An ultra-long FFT architecture implemented in a reconfigurable application specified processor[J]. IEICE Electronics Express, 2016, 13(13): 20160504. DOI:10.1587/elex.13.20160504.

[15] 周益超. 数字广播通信系统中 FFT 的算法仿真与 FPGA 实现[D]. 南京：东南大学, 2016.
ZHOU Y C. A simulation of FFT algorithm and FPGA implementation in digital broadcasting communication system [D]. Nanjing: Southeast University, 2016.

作者简介：

徐礼晗　男，(1994-)，硕士研究生. 研究方向为 SoC 设计.

景　佳　男，(1977-)，博士，副教授. 研究方向为大学物理教学和医学物理计算.

许丁鸿　男，(1996-)，硕士研究生. 研究方向为 SoC 设计.

宋宇鲲(通讯作者)　男，(1975-)，博士，副研究员. 研究方向为面向数据密集与计算机密集应用的 SoC/MPSoC 体系结构与实现. E-mail:songyukun@hfut.edu.cn.