华为第一届领航杯应用数学大赛

低功耗自适应 FIR 滤波硬件算法

柴昊、李明昊、王熙元

复旦大学
南京大学

2023.9.20

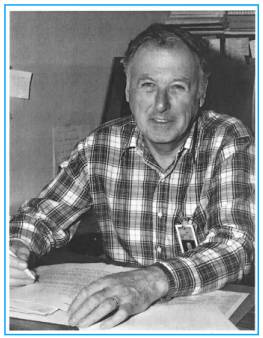
目录

理论回顾

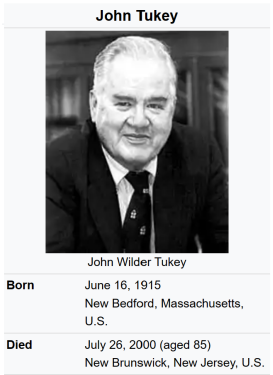
快速 Fourier 变换 (FFT) 处理器算法

一种浮点 FFT 流处理器：基于冗余计算和折叠架构的浮点运算蝶形阵列

> Cooley 和 Tukey 是 Princeton 的两位数学家和计算机科学家，他们首先设计出离散傅立叶变换算法 (Discrete Fourier Transformation/ Fast Fourier Transformation, DFT/FFT)。时间复杂度可以达到 $\mathrm{O}(n \ln n)$。此方式适用于 $n = 2^m$。对于 $n \neq 2^m$ 的多项式乘法或卷积，总可以通过引入先导零/占位的方式约化到 $n = 2^m$ 的情形，由于 k 和 $2k$ 之间必然有一个 2 的次幂，理论上快速傅立叶变换的时间复杂度不会超过 $\mathrm{O}(2n \ln n)$。另外 Cooley-Tukey 的变形中会处理 $N_1 N_2$ 阶数 FFT，通常分解为 N_1 个 N_2 阶 FFT 处理 (N1 $\ll$ N2)。



(a) James Cooley



(b) John Tukey

图: 发明第一个 FFT 算法的数学家及计算科学家

冗余计算加速算法

基四最小冗余混合加法 (mrHY4A)

算法架构简介

本项目选用了基四最小冗余 (Minimal Redundant Radix-4,mr4) 方案，对于任意一个 $2n$ 二进制位的整数 $X \in \mathbb{Z}$，其 mr4 冗余表达式为：

$$X = \sum_{i=0}^{i=n-1} [x_i]4^i, \quad [x_i] \in \{-2, -1, 0, 1, 2\}$$

而数位 $[x_i]$ 用三个 bit $(x_i^{-2}, x_i^{+}, x_i^{++}) \in \{0, 1\}^3$ 表示为

$$[x_i] = -2x_i^{-2} + x_i^{+} + x_i^{++}$$

其优势至少体现在以下三点。

- 冗余示数法具有多映射的特点实现无**进位传播**的冗余加法
- 大端和小端先入结构的硬件优势
- mr4 冗余表示不需要符号位

与无优化算法对比

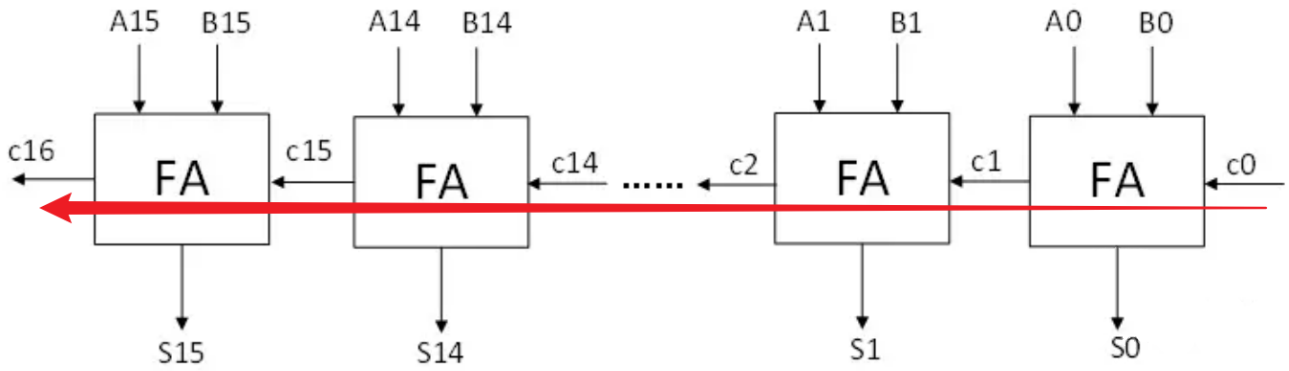


图: 非冗余计算形式

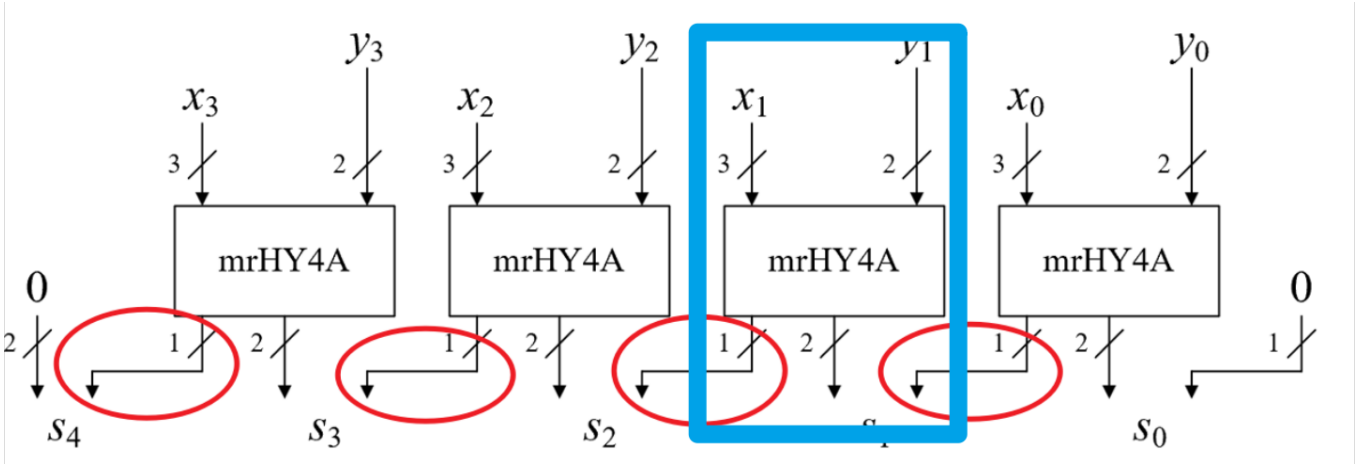
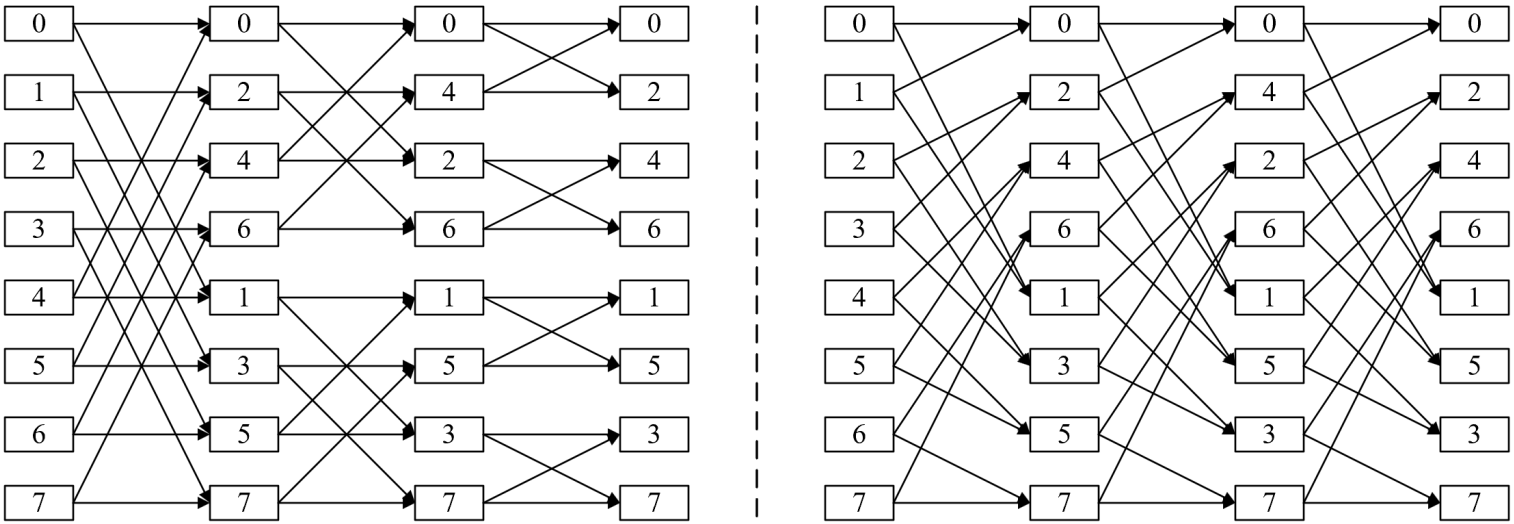


图: 冗余 4 计算形式

算法实现与创新

优化思路一

采用各层同形的 FFT 拓扑结构优化绕线



Cooley-Tukey 传统NTT蝶形拓扑　　各层同形NTT蝶形拓扑

图: 不同绕线方式比较

并行 FFT 设计中，传统的 Cooley-Tukey 蝶形拓扑的各层绕线复杂度不同。

> 具体记各层布线复杂度 $O_1 > O_2 > \cdots > O_{\log_2 N}$ (一般后层布线更迅速)。则硬件实现拓扑复杂度是关于 $O_1, .., O_{\log_2 N}$ 的带权函数。

优化思路二

通过蝶形运算 Folding 架构

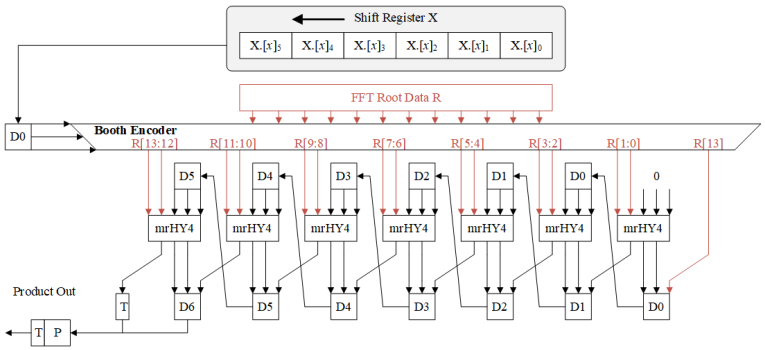


图: mr4HY 串行乘法器

> 好处：降低绕线复杂度；降低浮点运算复杂度；减小并行压力。

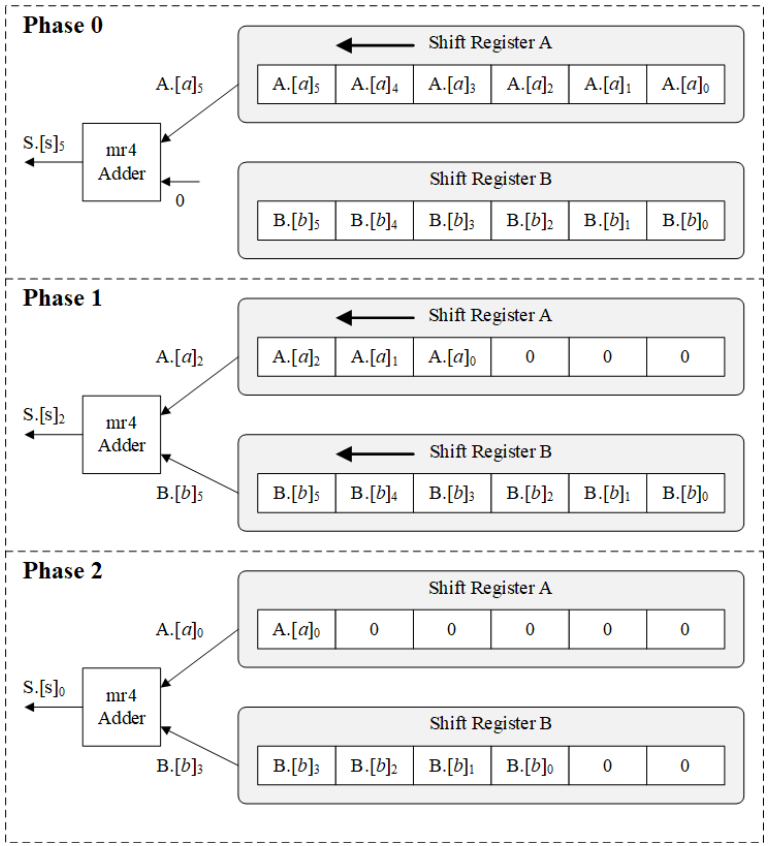


图: mr4HY 串行浮点加法器

流处理器工作管道

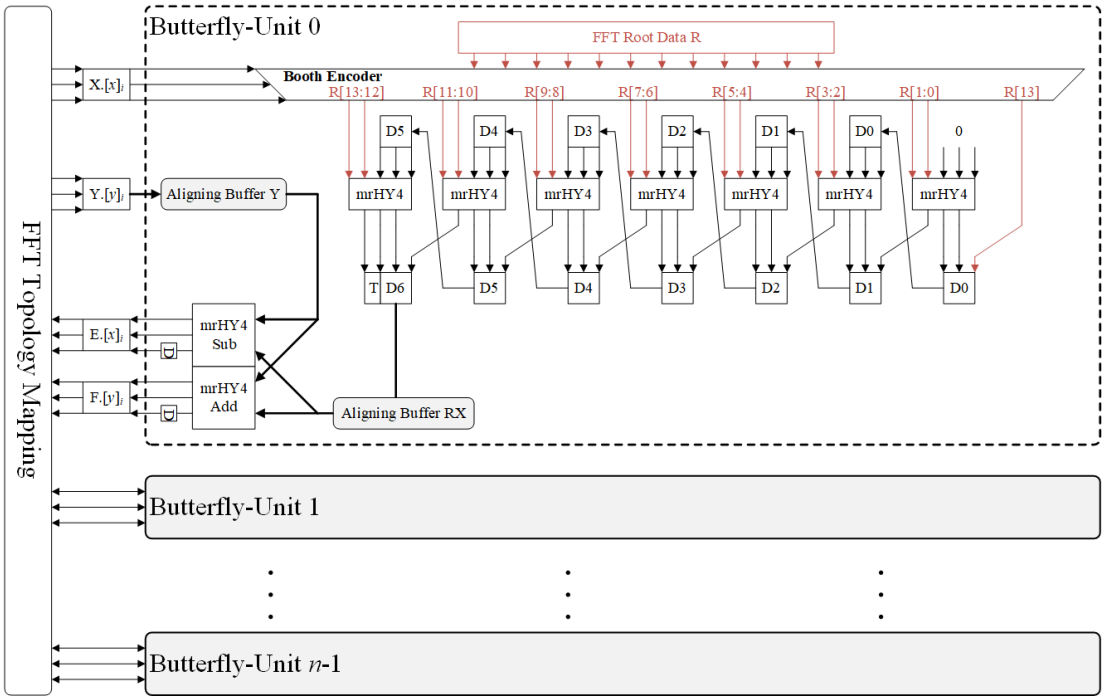


图: 蝶形运算阵列

a) 第一级流水线为冗余乘法：FFT 上一级蝶形运算输出，经过 Mapping 拓扑映射单元送往当前蝶形运算输入端，随输入数据串入，乘法执行结束，与此同时，乘法器输出逐个数位地输入 Aligning Buffer 中，便于后续冗余加减法操作；

b) 第二级流水线为冗余浮点加减法：从 Aligning Buffer 先后输出数据被加减归约，其输出至 Mapping 单元，以便送达 FFT 下一级蝶形运算对应蝶形结输入端。

技术优势

本方案使用**折叠展开架构**，本质上是在 **蝶形运算并行度**和 **FFT 每级内蝶形并行个数**之间做权衡，通过内部折叠蝶形，外部展开计算阵列，完成拓扑结构的化简与计算流程的化简。

降低绕线复杂度的方法

1. 采用各层同形拓扑
2. 采用 Digital Serial 处理结构

其他设计收益

1. 冗余串行算法降低浮点运算复杂度
2. 大端输入减小系统延迟
3. 流处理结构的控制逻辑非常简单

进一步的优化思路：可以利用 Twiddle factor 常数乘法的特点，对串行乘法器做进一步简化。

性能比较

基于 Xilinx Artix-7 仿真系统数据

表 2: FFT-based FIR or Polynomial Multiplication Implementation Results and Comparison

Work[1]	# LUTs	# FFs	# DSPs	# BRAMs[2]	# Slices[3]	Frequency (MHz)	Latency (cc)	Time (μs)	ATP (# SEC[4] $\times$ μs)
2021, Chen	533	514	1	1.5	198 *	246	1030	4.18	2,499
2021, B, N	360	145	3	2	187	115	940	9.32	8,267
2021, B, N	737	290	6	4	371	115	474	4.68	8,288
2021, Guo	1,549	788	4	2 [5]	635	159	228	1.43	2,052
2021, Ma	5,181	4,833	16	0	1,468	227	143	0.63	1,933
2021, Yarman-1	948	352	1	2.5	281 *	190	904	4.76	4,194
2021-4	2,543	792	4	9	735 *	182	232	1.27	3,727
2021-16	9,508	2,684	16	35	2,713 *	172	69	0.40	4,525
Ours $N = 64$	3,120	3.224	0	0	910	200	56	0.28	254.8
Ours $N = 128$	6,250	6,718	0	0	1,975	177	64	0.36	714.1
Ours $N = 256$	13,447	14,002	0	0	4,122	162	72	0.44	1813.7

[1] All of the related works evaluate their designs on Xilinx Artix-7.
[2] 36Kb BRAM slices.
[3] For works that do not provide #Slices (marked by *), #Slices is approximated by #LUTs $\times$ 0.25 + #FFs $\times$ 0.125 xilinx7series.
[4] #SEC = #BRAMs $\times$ 200 + #DSPs $\times$ 100 + #Slices.
[5] Number of BRAMs used during the polynomial multiplication.

- "**Ours**" stands for our team project performance.

图: 本团队工作和其他已知算法的性能比较

后续计划

基于量化感知进行信号预测和自适应处理

通过 OpenNN C++ 神经网络框架

使用 OpenNN 神经网络框架搭建量化感知算法

回顾上部分的 Twiddle Factor 近似技术加速硬件运算

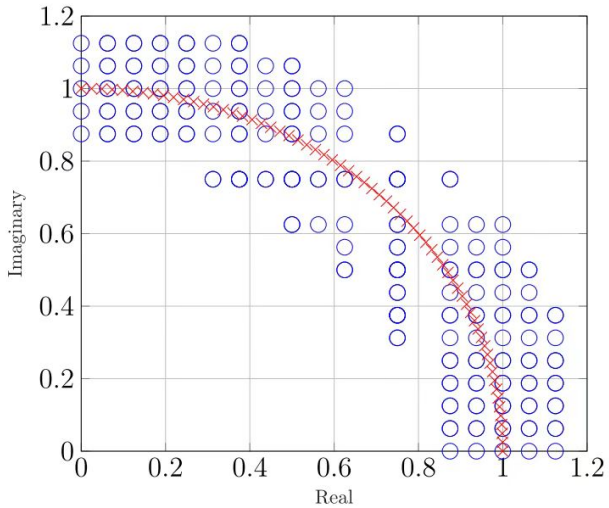


Fig. 2. Quantized twiddle factors: red cross markers are the exact twiddle factors, blue circles are the alphabet of complex numbers with a maximum of two nonzero digits in CSD representation of each real and imaginary part.

图: 近似 FFT 与 Twiddle Factor

基于此高效 FFT 流处理器的降噪/变频算法

基于不同应用场景

- 射频波段的频率俘获/提纯硬件算法:

  具体理论来自数字信号处理/小波分析

- 全频带的自适应信号预测:

  将主要基于此次竞赛实现的流信号处理器，使用贝叶斯网络或智能 Pid 控制策略设计算法

- 声学波段的主动降噪算法:

  希望可以开发出能使用在蓝牙耳机/通话系统等硬件的降噪功能模块上

最后，此项工作也有更深层次的理论意义，即开发更快的浮点乘法和 Fourier 变换运算器。这在图像处理，数值计算等领域都是重要的。

致谢

报告结束 感谢聆听