HUAWEI APPLIED MATHEMATICAL CONTEST

Contest Thesis

Low Dissipation and Adaptive FIR Filter Algorithms

Author:
Chai Hao,
Li Minghao,
Wang Xiyuan

Team:
REK3000

A thesis submitted in fulfillment of the requirements
for the HUAWEI Applied Mathematics ContestApplied Mathematics Contest

Question Number 7 in August 2023

Fudan University, Nanjing University
Chai Hao in SMCS,Fudan University,
Li Minghao in EIE,Nanjiing University,
Wang Xiyuan in SMS,Fudan University

华为领航杯应用数学大赛论文

低功耗自适应有限脉冲滤波器算法

2023.8

摘要

本文主要实现一种低功耗 (Low Dissipation) 自适应 (Adaptive) 的有限长单位脉冲响应滤波器 (Finite Impulse Response Filter) 算法。主要参考文献是 [Winograd 1968; Gholami et al. 2021; Jiang et al. 2020; Cormen et al. 2022; Nussbaumer 1982]。

低功耗算法的主要思想是两类：一是等价改变滤波器架构最小化乘法器个数；二是降低滤波器的乘法器的运算复杂度。其中架构的不同实现包含 Cooley-Tukey 算法、内积加速算法[Winograd 1968]、Karatsuba 快乘[Karatsuba 1995] (主要是通过减少乘法而加速离散傅立叶变换，通论可见快速[Nussbaumer 1982, Chap1, 2])；加速乘法器本身的算法诸如 tooth 加速乘法[C. Li et al. 2010]、近似乘法和量化感知优化[Gholami et al. 2021](中文写成的数值计算神书 [C. Li et al. 2010]有详尽说明)。

自适应算法采用机器学习预测算法，主要保证快速校正和模型无损 (题目的诉求三)。通过预测信号量进行自适应滤波能够输出指定的或期望的样式信号，以达到降噪、降低功耗、快速响应等实时场景的不同要求。

本文是第一届华为领航杯应用数学大赛参数论文。本文之格式内容均依照该竞赛要求完成。

论文目录

1 问题背景阐述 **5**

1.1 有限脉脉冲响应滤波器 (Finite Impulse Response Filter, FIR Filter) . . . 5

1.2 离散傅立叶变换及快速实现 (Discrete/Fast Fourier Transformation, DFT/FFT) . . . 9

1.3 信号处理算法中运算器性能 . . . 10

1.4 技术诉求 . . . 11

2 经典方法复现 **12**

2.1 基于 Cooley-Tukey FFT 算法实现的 FIR 滤波器 . . . 12

2.2 冗余计算和 Booth 乘法 . . . 12

2.3 基于量化感知的自适应加速 . . . 12

3 理论及算法 **16**

3.1 离散傅立叶变换 (Discrete Fourier Transformatioin, FFT) 的快速实现 . . . 16

3.2 Booth 无进位乘法和 Karatsuba 快乘 (Booth Multiplication Algorithms & Karatsuba Multiplication) . . . 18

3.3 量化感知中的混合精度量化 (Mixed Precision Quantization) . . . 21

4 实验数据及软硬件介绍 **24**

5 实验结果及主要结论 **25**

5.1 主要测试结果 . . . 25

5.2 结论 . . . 25

6 总结及展望 **26**

6.1 源程序文件树 . . . 28

6.1.1 滤波器代码注解 . . . 28

6.1.2 机器学习框架代码注解 . . . 29

6.1.3 可视化框架注解 . . . 29

6.2 使用指南 . . . 29

6.2.1 操作系统和软件要求 . . . 29

6.2.2 执行步骤 . . . 29

附录 **29**

参考文献 **30**

插图

1 两种基本 FIR 架构 6
2 层级形式 FIR 滤波器 (实数化乘法) 7
4 减少系数乘法个数 8
5 内存管理优化 9
6 系统识别滤波器工作过程 13
7 噪声抵消滤波器工作过程 13
8 信号矫正滤波器工作过程 14
9 连续型参数的量化离散映射 15
10 蝶形运算单元 17
11 项目流水线示意图 17
12 基四最小冗余混合加法 (mrHY4A) 18
13 Serial mrHY4A 19
14 冗余浮点加法运算过程 19
15 冗余浮点乘法运算过程 20
16 一种混合精度神经网络架构 22
17 量化神经网络的模型训练方法 22

1 问题背景阐述

论文的第一部分是问题背景阐述。主要说明研究问题的背景来源、目前的主流解决策略、本项目采用的解决方式以及后续可能出现的前景和困难等。这一部分本论文将详细解释 FIR 滤波器的原理、离散傅立叶变换的原理和计算机实现、信号处理算法中的乘法器性能等内容。最后我们将简要说明本项目如何分别解决此次比赛中题目的三个技术诉求。

1.1 有限脉脉冲响应滤波器 (Finite Impulse Response Filter, FIR Filter)

工程上进行 FIR 滤波器 (FIR Filter, 以下简称 FIR 或 FIR 滤波器) 是非递归型滤波器的简称，全称是有限长单位脉冲响应滤波器 (Finite Response Filter)。

滤波器是对波动信号进行处理的基础器件，工程上需要对信号 (诸如电磁波、声波、辐射光谱等随时间波动变化信号) 进行采样，得到随时间变换的采样信号序列。数学上抽象成一族点或自然数集上的函数 $\{z_{nT}\}_{n\in\mathbb{N}}$。这里 T 是采样周期，在一个采样周期中通常认为波形在基频 ω 的倍频上的能量分布是不随时间变化的，不同采样周期的频谱能量分布可能有变化。

数学上看 FIR 滤波器是一些重要信号处理变换 (诸如 Laplace 变换/ Fourier 变换等) 的近似情况下推广，其数学核心是如下的 Z-变换。

Z-变换 (Z-Transformation) 和卷积 (Convolution) 有限长单位脉冲响应滤波器 (以下简记为 FIR 滤波器) 数学上通常抽象成是一系列 (有限长度) Z-变换如下

$$
\begin{aligned}
&H(z)=\sum_{n=0}^{n=N} h(n)z^{-n}, \quad z\in\mathbb{C}\\
&(\text{或}H(z)=\sum_{n=N_1}^{n=N_2} h(n)z^{-n}, \qquad N_1,N_2\in\mathbb{N}),
\end{aligned}
\tag{A1}
$$

其中 $h(0),..,h(n)$ 是滤波器的脉冲响应序列 (Impulse Response Sequence)。在 FIR 滤波器的设定中这是一个有限的序列 (一般而言从零到一个正整数 n，称为 FIR 滤波器的阶数)，从而 FIR 中的有限长单位因此得名。而乘数 $z\in\mathbb{C}$ 是一个标准的复数，用来表示处理信号采样的频率信息。z 通常可以随时间变化但是在一般是采样周期内假设是基频 ω 的常系数幂级数 (含时 Fourier 分析或构建语谱图 (Spectrogram) 的工程要求)，即 $z_{nT}=\sum_{k\in\mathbb{Z}} a_{n,k}e^{-\sqrt{-1}k\omega}, \quad \forall t\in\mathbb{R}^+, n\in\mathbb{N}^+$

特别地如果 $z=e^{\sqrt{-1}\omega}$ 是 Dirac 脉冲函数的离散 Fourier 变换 (DFT)，即一些特殊波形的频谱时无截断的 Z-变换时，有

$$
\begin{aligned}
H(z) &:= \textstyle\sum_{n>-\infty}^{n<+\infty} h(n)z^{-n}\\
\Rightarrow H(e^{\sqrt{-1}\omega}) &= \textstyle\sum_{n>-\infty}^{n<+\infty} h(n)e^{-\sqrt{-1}n\omega} = \widehat{h}(\omega)
\end{aligned}
$$

恰好是无截断的 DFT。(对 h 进行有限长度截断从而得到有限长度的 DFT)

进一步考虑 $(A1)$。如果考虑一段频率信号 $X(z)=\sum_{n\in\mathbb{Z}} x(n)z^{-n}$，那么其在一个 $(A1)$ 形式的 FIR 滤波

器作用下得到频率响应 $Y(z)$ (或者等价地称为脉冲响应，Impulse Response/Frequency Response) 应该是

$$Y(z) = H(z)X(z), \qquad z = e^{-\sqrt{-1}\omega}(\text{或其他形式的频谱})$$
$$Y(z) = \sum_{n\in\mathbb{Z}} y(n)z^{-n}, \qquad y(n) = \sum_{k=0}^{k=N} h(k)x(n-k), \quad \forall n \in \mathbb{Z}$$

于是计算滤波器输出的频率响应 $Y(z)$ 的各分量 (各倍频的能量分配) 是

$$y(n) = \sum_{k=0}^{k=N} h(k)x(n-k), \quad \forall n \in \mathbb{Z}, \tag{A2}$$

因为这代表了滤波器的理论输出，从而通过式 $(A2)$ 可以计算每一个倍频上分配的多少能量。抽象到数学上这是给定输入序列 $\{x(n)\}_{n\in\mathbb{Z}}$ 和有限卷积序列 $\{h(k)\}_{k\in\{0,..,N\}}$ 进行卷积运算 (Convolution Operation)。

从上述的数学表示可知，一个 FIR 滤波器的硬件算法性能，最底层运算速度上是受以下两方面制约

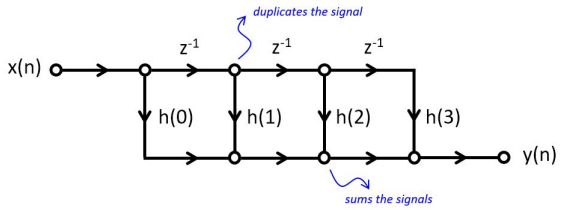
- **第一个公式 $(A1)$ 中的 z^{-1}-乘法器的性能**。这部分对应 FIR 滤波器的真实输出。
- **第二个公式 $(A2)$ 中的卷积操作的性能**。这部分对应滤波器的理论输出。

通常而言卷积运算中涉及分量错位乘法时也会加入硬件实现的高精度-高精度乘法的性能，由于 $(A1), (A2)$ 完全表示相同的频谱信息，本项目主要大力优化卷积操作的性能来实现华为 ‘‘低功耗自适应 FIR 滤波器’’ 难题的技术诉求。

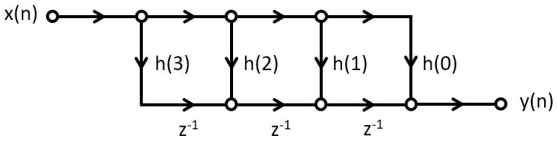
除此之外，滤波器本身在硬件上的实现要适应硬件自身的条件，例如不同硬件的基础运算性能不尽相同、不同硬件的内存管理资源调度的性能也不尽相同。以下简要通过 网络资料 简单解释这两点。

滤波器的架构优化 FIR 滤波器大概有如下四种不同的架构实现。本项目通过综合如下四种架构的优劣开发出自行的架构。这将在论文的第三部分说明。

第一种是直接形式 (Direct-form)。其带有常数系数的滤波器数据流程图如下图1a所示。



(a) 直接形式 FIR 滤波器 (阶数 $n = 4$)



(b) 反向形式 FIR 滤波器 (阶数 $n = 4$)

图 1: 两种基本 FIR 架构

从图中可以看出直接形式滤波没有对架构作任何优化，将会进行 $n+1$ 次常系数 z^{-1}-乘法和 n 次加法。其中 n 是滤波器阶数，那么进行完整的 FIR 滤波信号处理的时间复杂度至少是 $\mathcal{O}(n^2)$ (假设乘法复杂度和加法相同)。

和直接形式相反是反向形式 (Transpose-form) 即上述1b。注意此时的乘法器的系数正好和直接形式实现相反。相比于直接形式反向形式没有复生信号的操作，而是并行处理信号后累加。于是对于并行计算性能好的硬件，反向形式架构的滤波器性能更好。

而事实上硬件上高精度乘法要比加法慢很多，如果能够在架构设计上减少乘法次数将有效地提升滤波器性能。后续的层级形式 (Cascade-form) 和频采形式 (Frequency Sampling) 均是基于这种思想。首先考虑层级形式，考虑 Z-变换的 Laurent 多项式以 z^{-1} 变量因式分解得到

$$H(z)=\sum_{n=0}^{n=N}h(n)z^{-n}=h(0)\prod_{k=1}^{k=N}(1-a_kz^{-1})=\prod_{k=1}^{k=N}G_k(z)$$

这里 $a_k\in\mathbb{C},k=1,..,N$ 是一些和 $H(z^{-1})$ 的零点的倒数。那么安装滤波原理可以逐次实现 $G_1(z),..,G_k(z)$ 的滤波器再将它们串行处理。特别地如果 $\{h(n)\}_{n\in\mathbb{N}}$ 是实数序列那么根据复根成对原理可以将 H 分解为一些至多二次因子的乘积。从而在串行处理的滤波器中只包含实数乘法。这样作降低了复数乘法的次数，提高了性能。下图是一个按上面论证实现的 FIR Cascade-form Filter。

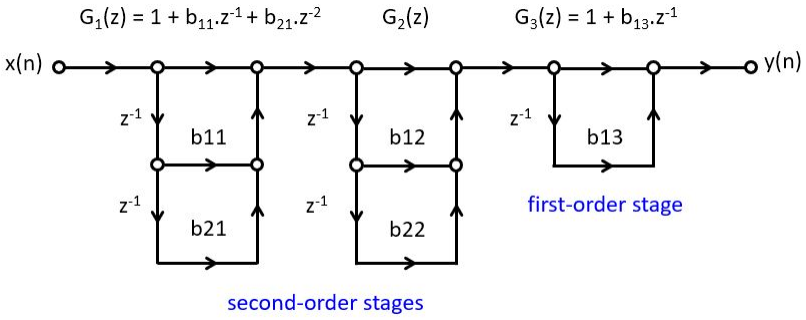


图 2: 层级形式 FIR 滤波器 (实数化乘法)

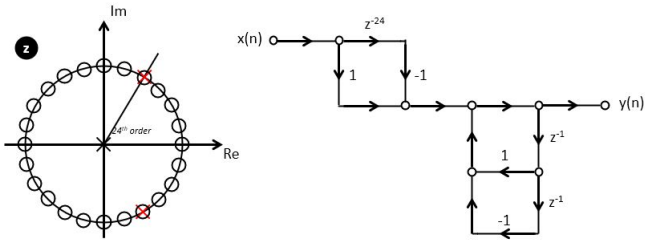
层级形式将阶数较高的 FIR 滤波器分解成小阶数的子滤波器，此设计可以模块化架构从而增大数据吞吐量。可以说有一定功能。最后一种架构是功能性架构即频采架构 (Frequency Sampling)。其来自于物理学中的共振器/共鸣器 (Resonator)，作用是在特定频率上放大其幅度/振幅。Z-变换结构是

$$H(z)=\frac{z^2}{z^2-2\cos\theta z+1}=\frac{1}{1-2\cos\theta z^{-1}+z^{-2}}=\frac{1}{X(z)}\tag{A3}$$

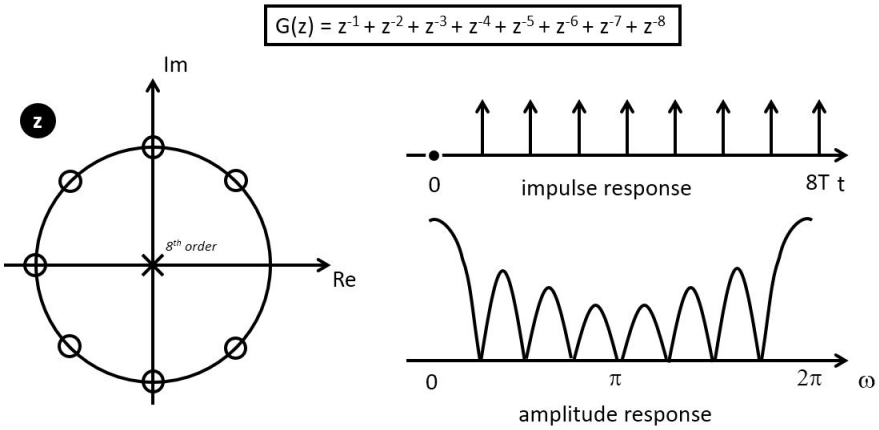
这里 θ 通常和共振频率 ω 相关，而其反方向是实现是一种频率梳 (Comb)。滤波器结构是形如

$$H(z)=1-z^{-m},\tag{A4}$$

$(A4)$ 的 Z-变换通常会将频率为 m 间隔的局部响应消除从而呈现如下的梳状结构 (3b)。事实上 $(A3)$ 的 $H(z)$ 可以写成频率梳的某种逆变换 (按式 $(A3)$)。



(a) 频率采集 $H(z) = 1 - z^{-24}$



(b) 频率梳的效果

另外，如果响应序列 $\{h(n)\}_n$ 具有一定的对称性，通过更聪明的办法可以设计乘法器更少的架构以提升性能。例如 4 中 $h(n-k) = h(k), \forall k \leq n$ 是中心对称的，那么通常的直接实现可以改成系数乘法次数减半的对称架构如

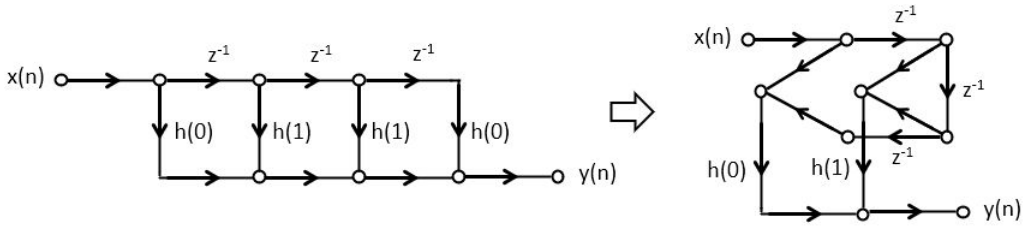
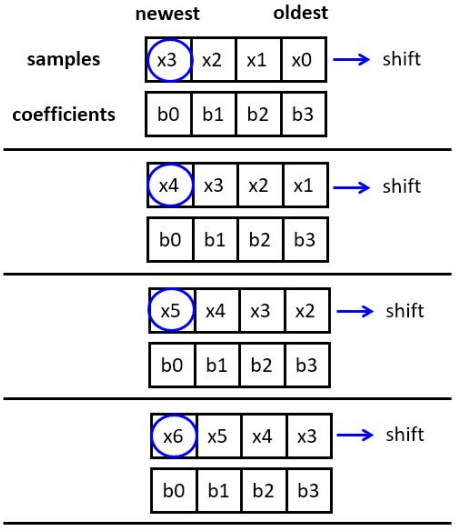


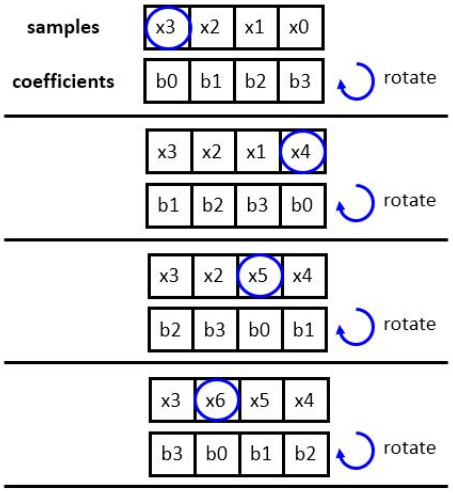
图 4: 减少系数乘法个数

另外如果 $h(n)$ 中有公共间距数字例如 $3 = 1 + 2, 5 = 1 + 2 + 2$ 时可以通过只计算 $\times 2$ 乘法然后累加的方式降低乘法器算力负担。

滤波器中的内存管理 经过上百次系数乘法计算后 FIR 设计可能变得相对复杂很多。因此在某种意义上对 FIR 滤波器硬件进行内存管理尤为重要。详细的字典级资料此处建议参考[Parhi 2007]。本处只指出关于缓存管理的本项目优化方式。即将信号延迟处理的缓存移位操作优化成指标计算的旋转操作。示意图如5。



(a) 缓存 Shift 操作



(b) 缓存旋转操作

图 5: 内存管理优化

其效果参考如下表格。

表 1: 性能比较

FIR Design	**16 taps**	**32 taps**	**64 taps**
tap-shift	383	703	1343
circular buffer	306	530	913
throughput (circular)	1.25x	1.32x	1.47x

最后对于更加详细的信号处理介绍，尤其是滤波器的简要介绍本文采用了此课程页面为主要参考资料。其为伍斯特理工学院的 “实时信号处理 B” (Real Time Digital Signal Processing B Term) 课程。

1.2 离散傅立叶变换及快速实现 (Discrete/Fast Fourier Transformation, DFT/FFT)

在上一节中 $(A2)$ 涉及到卷积算法的快速实现。实际上考虑最简单情形的两个同次数多项式乘法

$$A(x) := \sum_{n=0}^{n=N} a(n)x^n, \qquad B(x) := \sum_{n=0}^{n=N} b(n)x^n, \qquad N \in \mathbb{N}^+$$

$$C(x) := A(x)B(x) = \sum_{n=0}^{n=2N} c(n)x^n$$

乘法结果中 $C(x)$ 的各项系数 $c(n)$ 不难计算出是 $\{a(n)\}_{n=0,..,N}$ 和 $\{b(n)\}_{n=0,..,N}$ 的卷积和 (或卷积的某处值) 形如

$$c(n) = \sum_{k=0}^{k=N} a(k)b(n-k), \qquad n = 0, .., 2N$$

由于多项式可以被其各项系数唯一决定，离散卷积算法等价于多项式乘法系数计算算法，也就是说求序列的离散卷积和求两个多项式乘积的系数一样困难 (在算法理论上看这两个问题是等价的)。

根据上面计算如果用多项式系数存储多项式，计算多项式乘法的复杂度大概是 $\mathcal{O}(n^2)$ (因为计算 n 个乘积系数，每一个系数平均要进行 $\frac{1}{2}n$ 次乘法和加法)。那么这是可以在时间复杂度上优化的吗？考虑著名的 Lagrange 插值公式，一个 n 次多项式在 $n+1$ 个点的取值唯一确定了这个多项式。具体表达式是

$$F(x)=\sum_{k=0}^{k=n}F_k\prod_{i=0,i\neq k}^{i=N}\frac{(x-x_i)}{(x_k-x_i)},\qquad \text{where } F(x_k)=F_k,\ k=0,..,N$$

如果考虑 $(F(x_0),...,F(x_n))$ 表示多项式 F 则另一个多项式 $G\sim(G(x_0),..,G(x_n))$ 同样表示。此时计算 F,G 乘积只需要 n 次乘法即

$$FG\sim(F(x_0)G(x_0),...,F(x_n)G(x_n))$$

如果能以时间复杂度小于 $\mathcal{O}(n^2)$ 的算法从多项式系数计算出其在选定的 $n+1$ 个点的数值及此过程的逆运算，那么定点表示的多项式乘法算法就在性能上 (至少运算速度) 超过普通乘法。记 $N=\deg F$。从离散傅立叶分析的观点如果取 $x_k=e^{-\frac{2\pi\sqrt{-1}}{N+1}k}$，那么可知

$$F(x_k)=\sum_{n=0}^{n=N}f(n)x_k^n=\sum_{n=0}^{n=N}f(n)e^{-\frac{2\pi\sqrt{-1}}{N+1}kn},k=0,..,N$$

于是 $F(x_k)=\widehat{f}(k)$。即在上述选定点下，多项式的定点序列是系数序列的离散傅立叶变换 (DFT)。那么系数序列此时是定点序列的离散傅立叶逆变换 (Discrete Inverse Fourier Transformation,DIFT)。于是如何快速计算 DFT 称为快速卷积、多项式快乘甚至随机变量分布计算或数论算法等领域的核心问题。而第一个快速实现 DFT 的算法是基于分治思想的 Cooley-Tukey Scheme。

Cooley 和 Tukey 是 Princeton 大学的两位数学家和计算机科学家，他们在二十世纪利用快速实现的离散傅立叶变换算法 (Discrete Fourier Transformation/ Fast Fourier Transformation, Discrete Fourier Transformation with Fast ImplementationDFT/FFT)。时间复杂度可以达到 $\mathcal{O}(n\ln n)$。此方式适用于 $n=2^m$ 时，本项目实现方式主要参考了算法导论一书的 FFT 章节见 [Cormen et al. 2022, Polynomials and FFT] 和 Wikipedia。

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 $2k$ 中必然有一个 2 的次幂，理论上快速傅立叶变换的时间复杂度不会超过 $\mathcal{O}(2n\ln n)$。另外 Cooley-Tukey 的变形中会处理 N_1N_2 阶数 FFT，通常分解为 N_1 个 N_2 阶 FFT 处理 ($N_1\ll N_2$)。参看本文第二、第三部分。

1.3 信号处理算法中运算器性能

实时信号处理中提取的时间/频率信号 (如 u_t 或者 z_t) 通常而言取值在一个高精度的浮点数集合上，例如 C/C++ 中的 float_32 或 float_64 等不同位数的浮点数。工程原理上来说，计算机 (“朴素” 的计算机，不包括例如 Wolfgang Mathematica 此种存储实现任意精度的实数计算算法和诸多几乎 “完全精度” 的计算数学全套算法的超级计算机代数系统) 无法实现任意精度的数值计算，因为内存的有限性和很多计算过程实现时的限制。事实上 64 位的高精度浮点数在信号处理上已经非常够用了，而且为了集成到硬件，综合考虑性能、稳定性等因素，滤波器中的乘法器构件的实现，往往还需要舍弃一定精度作近似计算。另外，一些更复杂的运算器

例如卷积、内积、张量乘等运算，其优先级一般低于常数乘法或浮点数乘法，但是不同实现方式的性能不尽相同。通常用执行一次运算器使用的基础乘法次数和基础加法次数带上权重 (表示单次乘法的平均耗时或资源开销) 来衡量运算器的性能。如

$$P_{\text{op}} := N_{\text{mul}} w_{\text{mul}} + N_{\text{add}} w_{\text{add}} + P_{\text{sche}}$$

上面的 P_{sche} 是硬件的调度性能指标，本项目尽可能将这个指标拟合为和缓存量、I/O 吞吐量相关的近真实指标。不仅是硬件、算法层面上度量运算器的性能也尤为重要。在本论文的第四和第五部分，我们团队采用了上述指标对算法进行了测试，并记录描述了结果。

1.4 技术诉求

针对华为领航杯应用数学大赛的难题七 ‘‘低功耗自适应 FIR 滤波器’’ 有如下三个主要技术诉求。

1. **诉求一**：通过改变滤波器架构设计等办法，使得其硬件算法整体性能提升 40%。
2. **诉求二**：指定位宽下引入近似乘法，实现乘法资源降低 50%，误差或信噪比 (SNR) 大于 75dBc ；或者使用机器学习模型训练，使得其训练性能基本无损 (< 0.1dB)
3. **诉求三**：‘‘低位宽’’ FIR 滤波器算法实现 (< 6bit)，并能够实现模型稳定校正且性能基本无损 (< 0.1dB)

诉求一 就本团队理解衡量 FIR 滤波器的整体性能指标可以使用功耗面积度量。

诉求二 本项目通过基准测试方案在仿真机上对 FIR 滤波算法进行测试。主要指标为 (参考刘伟强等人综述[Liu et al. 2019]) 信噪比

$$\text{SNR} := 10 \log_{10} \left[\frac{\max_i |\text{AR(i)}|^2}{\frac{1}{N} \sum_{i=0}^{N-1} |\text{AR(i)} - \text{ER(i)}|^2} \right] \tag{A5}$$

其中 RR(i) 是真实 (近似) 结果，ER(i) 是全精度结果。后续部分将着重测试此指标。

诉求三 通过理论计算确定实现位宽的理论值应该为 12bit。通过更大量的 ‘‘代价权衡’’ 将滤波器的位宽降低到 6bit，本项目后续将采用自适应机器学习预测的办法完成此诉求。

在本文第二部分经典方法复现中，三个子节将会依次详细论述本项目对上述三个诉求的经典解决方式。

在第三部分 ‘‘理论和算法’’ 中本文将尽可能清晰地叙述项目中使用的关键算法理论，具体创新和优化的细节部分。

在第四部分 ‘‘实验数据及软硬件介绍’’ 和第五部分 ‘‘实验结果及主要结论’’ 中本文将展示根据前三部分的算法理论实现的工程在仿真环境的测试数据以及指标比较。

最后关于项目的复现以及移植，完整的源代码解释和运行环境信息全面呈现在附录一节。

2 经典方法复现

这里复现几个可以显著提高滤波器性能的硬件算法。分别是著名的快速傅立叶变换 (Fast Fourier Transformation)、Booth 无进位乘法 (Booth Multiplication Algorithm) 和 Karatsuba 快乘 (Karatsuba Multiplication)。

2.1 基于 Cooley-Tukey FFT 算法实现的 FIR 滤波器

本项目实现了 Cooley-Tukey 的快速傅立叶变换算法 (Fast Fourier Transformatioin, FFT)。

实现细节可见算法理论的经典教材算法导论[Cormen et al. 2022, Chap30] 一书。

由第二部分如果采用无优化的离散 Fourier 变换算法在硬件上进行运算，那么完成一次完整的 DFT 需要进行 $\mathcal{O}(n^2)$ 次乘法和 $\mathcal{O}(n^2)$ 次加法。其中 n 代表 DFT 变换器的阶数 (或者滤波器的抽头数)。这个速度一般而言是硬件运算能承受的最大时间复杂度。

在二十世纪六十年代，来自 Princeton 的计算机科学家 Cooley 和数学家 Tukey 发明了一种基于快速幂算法和 Gauss 和算术性质的快速离散 Fourier 算法。其可以通过数量级地减少 DFT 算法中的乘法次数大大加快原有离散 Fourier 变换的运算速度。本质上这一算法的理论框架已经被德国数学家 Carl Friedrich Gauss 于 19 世纪初发现并证明，参见Wikipedia。

此方式适用于 $n = 2^m$ 时，这里的实现方式主要参考了算法导论一书的 FFT 章节见 [Cormen et al. 2022, Polynomials and FFT]。

另外我们团队也考虑到了古老的 Wingoard 加速算法 (Wingoard Fourier Transformation,WFT)。其主要处理内积加速。

2.2 冗余计算和 Booth 乘法

此处是简单介绍冗余计算的概念。

本质上所有的大整数/高精度浮点数的计算都涉及到进位问题。原因很简单，在整数的范畴中 a^2 往往要大于 a。一个十分聪明的调整办法就是在表示数的时候不是采用 $[0, N]$ 区间而是 $[-\lfloor \frac{N}{2} \rfloor, \lfloor \frac{N}{2} \rfloor]$。

逆元的产生催生出一种硬件快速乘法：无进位乘法。

Booth 乘法器的原生算法可以在保持设定精度的前提下显著地提升滤波器中乘法器的性能。后续项目采用的 Booth 乘法是 “基四最小冗余 (Minimal Redundant Radix-4,mr4)” 方案的实现。具体方式本文放在第三部分阐述。

2.3 基于量化感知的自适应加速

这一节本文关注的是此问题的发展前景：利用机器学习方法实现滤波器的各种自适应功能。

自适应滤波器的常见场景 自适应滤波器是一种参数即时更新滤波技术。

自适应滤波器通过模型训练不断更新参数，使得滤波器的输出更加接近预期的输出。

一般而言，整个系统提供输入波形与预期波形。输入波形传入滤波器，通过相应计算得到输出波形。自适应滤波器旨在通过最小化输出波形与预期波形的某种差距以提高滤波能力。

自适应滤波器大体上可以完成四种任务。

- 系统识别

  在实际场景应用中, 我们需要识别一些未知的系统，这些系统会影响波形的传递, 使得传递波形与原始波形不同。自适应滤波器可用于识别一个未知系统。我们将输入波形分别通过滤波器以及未知系统, 将未知系统的输出波形作为预期波形，与通过了输入波形的自适应滤波器所输出的波形进行对比，并通过机器学习等方式训练滤波器参数，该滤波器最终可以代表未知系统的近似模型。

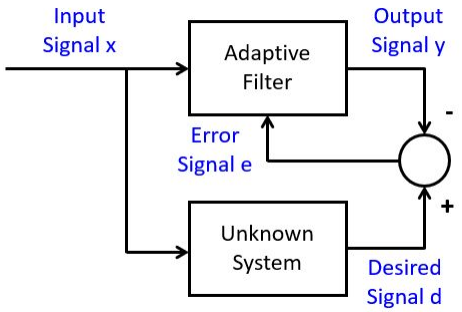


图 6: 系统识别滤波器工作过程

- 噪声抵消

  噪声抵消出现于传递波形包含噪声的情形。某一噪声在未知系统影响下被叠加到另外一段波形中，该波形被非线性地叠加在另一端输入波形中。我们希望通过滤波器还原出输入波形的初始形式。实际场景中，自适应滤波器将利用初始波形，噪声以及含噪声的最终波形进行参数训练。滤波器在噪声抵消的过程中将实际噪声作为输入，通过滤波器后将输出波形叠加到最终波形上，以逼近初始波形。

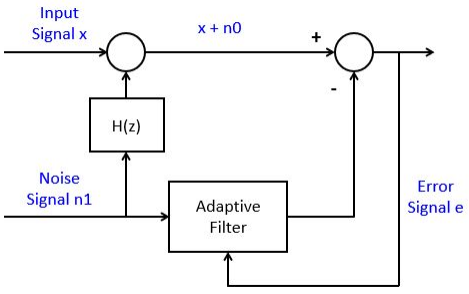


图 7: 噪声抵消滤波器工作过程

- 信号矫正

  在滤波过程中会出现一些信号扭曲的情形，导致收到的信号会与实际信号产生偏离。自适应信号平衡任务即是通过滤波器将发生偏离的信号进行矫正，以使得其逼近实际的无偏差信号。

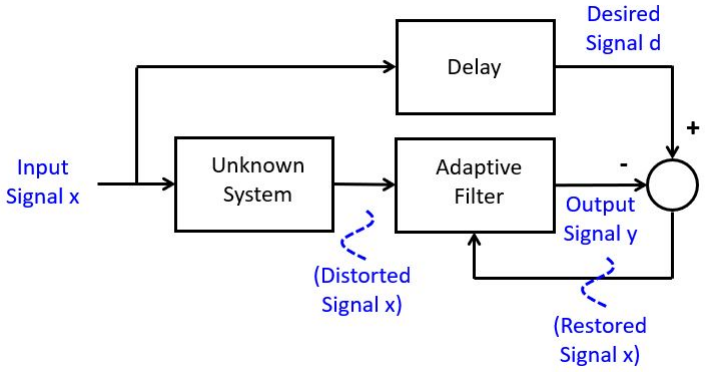


图 8: 信号矫正滤波器工作过程

- 自适应预测

  滤波器还可以用于自适应预测一段波形的未来形状。一般而言，自适应滤波器的系数可以用于计算未来波形，在信号编码中可以被使用。

自适应滤波器的数学模型如下.

对于第 n 次波形过滤过程，滤波器的滤波参数被定义为 $\omega(n) = [\omega_k(n)]_0 k = 1^L \in \mathbb{R}^L$. 初始波形为 $x(n) = [x_k(n)]_{k=1}^L \in \mathbb{R}^L$, 预期波形为 $d(n) = [d_k(n)]_{k=1}^L \in \mathbb{R}$.

滤波器的输出波形为

$$y(n) = x^T(n)\omega(n)$$

定义滤波器的单次误差为

$$e(n) = y(n) - d(n)$$

并采用最小二乘误差的优化算法进行模型的参数更新训练，即最小化单次误差的 l_2 范数:

$$e^2(n) = d^2(n) - 2d(n)x^T(n)\omega(n) + \omega^T x(n)x^T(n)\omega(n)$$

由优化知识结合梯度下降算法我们可以得到最终的参数更新公式如

$$\omega(n+1) = \omega(n) + 2\beta e(n)x(n), \tag{B1}$$

量化感知技术 由于滤波器工作场景硬件设备限制，量化感知技术被用于节约模型训练中的时间空间成本。量化感知技术主要实现模型实际参数的低精度近似储存。计算机中的参数储存通常由浮点数等相对较高精度的数据格式完成，而相对大型的模型在数据存储时消耗的资源较多。量化感知技术旨在将原本通过高精度数据

格式存储的方式改为用低精度存储参数。与此同时，量化感知技术希望在训练过程中也采用针对低精度格式的计算方法，相应技术构成了量化感知技术的核心。

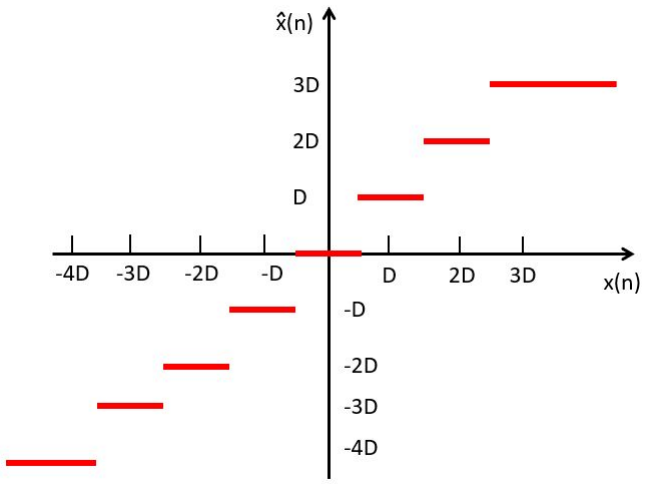


图 9: 连续型参数的量化离散映射

当前深度学习框架也增加了许多量化感知方法，这些方法可以在训练过程中直接将参数量化为低精度形式，并用低精度方法对模型参数进行直接训练。相对传统的训练方法在训练后将最终参数量化存储而言，前者更为直接地在训练过程中引入量化感知技术，减少了近似步骤，在精度方面也有提高。

自适应量化滤波器 自适应滤波器结合量化感知技术是一种提高自适应滤波器模型参数训练效率的方式之一。将自适应滤波器的模型参数放置于量化感知技术的框架下进行训练，例如搭建量化神经网络还原滤波器结构并搭建相应量化参数的优化算法，最终进行训练。量化感知技术由于其存储格式上的低精度处理，可以有效地降低存储结构的空间消耗。同时由于量化感知计算的计算复杂度降低，模型进行训练更为快速，即时修正模型参数效率得到提升。

量化感知技术 (Quantization Perception) 可以减轻神经网络训练上浮点数计算的时间/空间复杂度，从而释放算力进而加速神经网络的训练。

3 理论及算法

3.1 离散傅立叶变换 (Discrete Fourier Transformatioin, FFT) 的快速实现

基于第二部分经典理论，本项目实现高精度 FFT 流处理器的主要方式如下所述。

高性能 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 为实现小面积、高并行、高灵活度的 FFT 处理单元，本文创新性地提出了 FFT 流处理器，通过冗余计算和架构折叠来取得面积和运算效率的最佳权衡。

该部分创新点可总结为：

- 基于冗余计算提出了具有极短关键路径的 ‘‘冗余串行乘法’’ 单元和 ‘‘冗余浮点加法单元’’，他们的关键路径仅有 2 到 3 个全加器 (Full Adder,FA)。
- 通过乘加运算折叠 (Folding) 使得浮点数运算的复杂度极限接近定点数运算。
- 通过蝶形单元展开 (Unfolding) 使得选用各层同性 FFT 拓扑成为可能，从而充分简化系统控制逻辑，减少硬件运算单元在处理数据中的等待和空拍，提高硬件资源利用率。
- 依赖于数据的冗余表示法和各层同性的 FFT 拓扑，实现了紧凑的数据流映射方案。
- 使得系统的面积效率获得较大提升。
- 具体而言：(1) 通过折叠乘加运算；(2) 通过扩大蝶形单元并行度。

总结而言，在设计策略上，本文通过冗余计算使得浮点数乘、加运算取得较好的架构折叠方案；进一步通过灵活应用折叠和展开技术，实现了在相同面积约束下，乘、加子运算并行度与蝶形单元个数 (FFT 处理点数并行度) 之间的权衡，从而取得较为理想的面积效率。

此外更加高效的办法是参考 [C. Li et al. 2010]。或者关于快速傅立叶变换的综述性质书籍[Nussbaumer 1982]。

基于冗余计算的折叠蝶形计算阵列 图10 所示 Butterfly Unit 完成 FFT 所需的蝶形运算即

$$\begin{cases} E = Y + R \cdot X \\ F = Y - R \cdot X \end{cases}$$

而完成一个蝶形运算共需要两级流水线：

- **第一级流水线**为冗余乘法：FFT 上一级蝶形运算的输出，经过 Mapping 拓扑映射单元送往当前蝶形运算的输入端，随着输入数据的串入，乘法执行结束，与次同时，乘法器的输出也逐个数位地输入到 Aligning Buffer 中，便于后续冗余加、减法操作；
- **第二级流水线**为冗余浮点加、减法：从 Aligning Buffer 先后输出的数据被加、减归约，其输出被送往 Mapping 单元，以便输送给 FFT 下一级蝶形运算对应蝶形结的输入端。

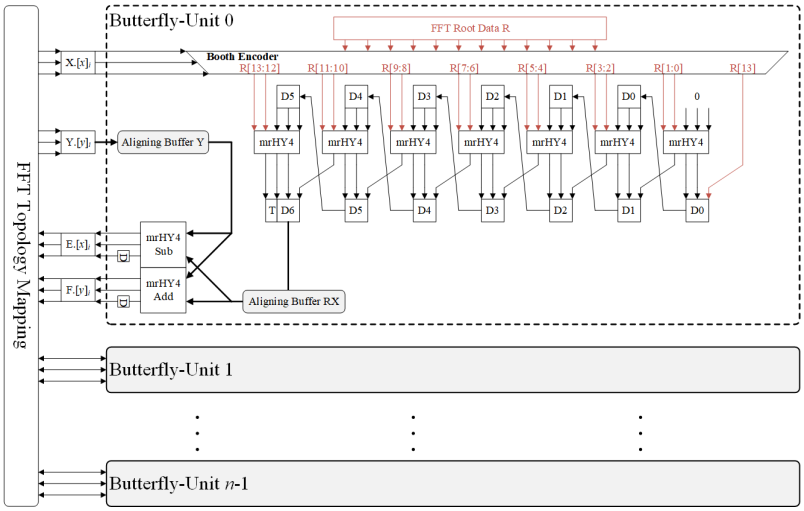


图 10: 蝶形运算单元

在这一过程中，两级运算流水执行，并无空闲等待时间，如下图任务流程图所示。每级流水的执行时间与量化精度 (即冗余量化位 LEN) 有关，且乘法与加、减法所需执行周期均等于量化位数 LEN。对于一个 $N = 2^n$ 点的 FFT，其共包含 $\log_2 N$ 级别蝶形运算，故而系统工作的总周期数为：

$$(\log_2 N + 1) \cdot \mathrm{LEN}, \tag{C1}$$

Mapping 模块实现了如下图所示拓扑。得益于选用折叠的乘法和加法结构，每个蝶形计算单元的面积已经尽可能地减小了，故而降低了多点数全并行蝶形单元的部署压力。

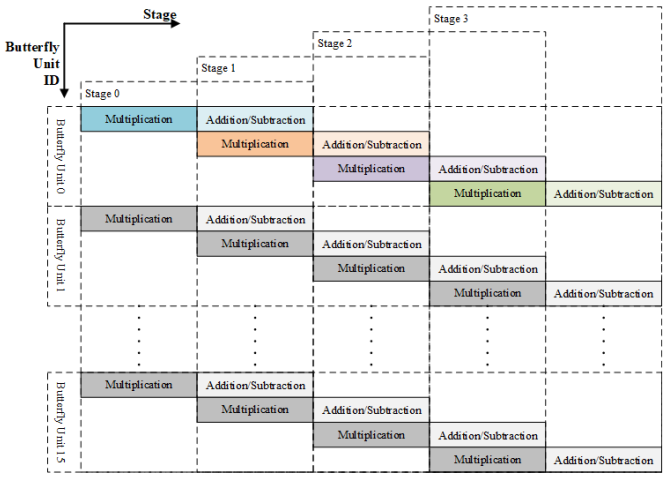


图 11: 项目流水线示意图

在我们的处理模块中，N 个蝶形单元并行处理数据，它们的输出经过 Mapping 模块映射到其他蝶形单元的输入端，结合流水化的运算流程，全自动地完成 FFT 的多级计算，除了原根输入和 Reset 操作外，整个系

统不包含状态机和控制逻辑，实现了自动化的流处理操作。

3.2 Booth 无进位乘法和 Karatsuba 快乘 (Booth Multiplication Algorithms & Karatsuba Multiplication)

本文选用了 ‘‘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：

$$X = \sum_{i=0}^{n-1} [x]_i \cdot 4^i$$

其中数位 $[x]_i \in \{-2, -1, 0, 1, 2\}$，在计算机中 $[x]_i$ 由三个比特 $\{x_i^{-2}, x_i^{+}, x_i^{++}\} \in \{0, 1\}^3$ 表示，即

$$[x]_i = -2 \cdot x_i^{-2} + x_i^{+} + x_i^{++}$$

相较于传统的二进制补码表示法，mr4 冗余表示具有以下三点优势：

- 传统的存在进位传播的二进制加法在硬件实现时往往要考虑进位传播；而冗余示数法具有多映射的特点，即相同的 X 具有多个不同的冗余表达式，利用这一特点，我们可以实现无 ‘‘进位传播’’ 的冗余加法，如图 ‘‘基四最小冗余混合加法 (mrHY4A)’’，如下图12。

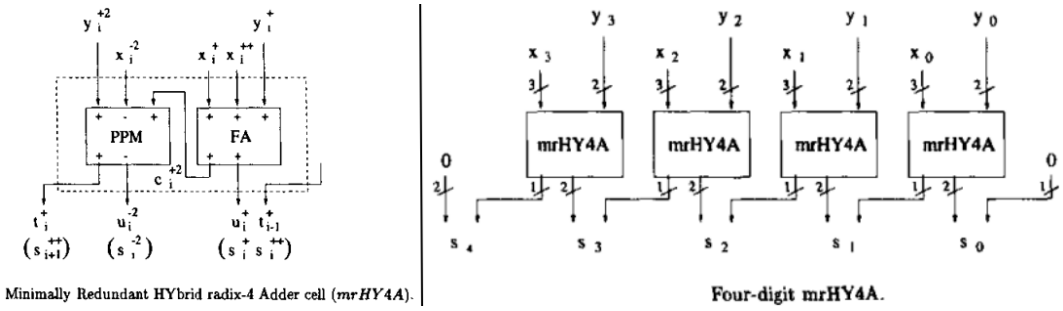
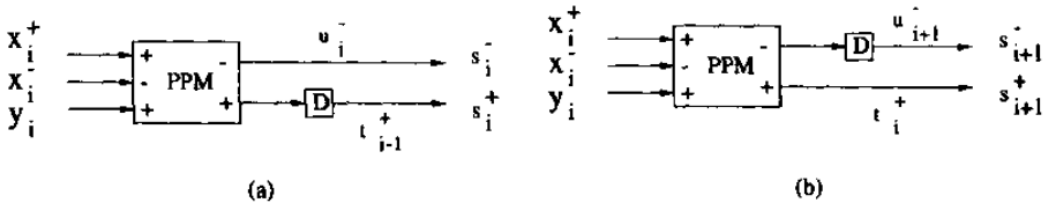


图 12: 基四最小冗余混合加法 (mrHY4A)

- 对于存在进位传播的传统二进制加法，其大端先入 (MSB-first-in) 折叠架构相较于小端先入 (LSB-first-in) 在硬件复杂度上具有天然的劣势；然而对于冗余表示，其大端和小端先入结构在硬件复杂度上几乎没有差别，如下图所示。

- mr4 冗余表示不需要符号位，其所表示数字 X 的符号蕴含在 $\{x_i^{-2}, x_i^{+}, x_i^{++}\}$ 的大小关系之中，这让我们避免了运算过程中的符号位扩展问题。

在浮点运算中，浮点加法的设计难度在于尾数的对齐。因而相较于定点数加法，浮点数加法具有更高的硬件复杂度。本文基于冗余 mr4 表示，针对浮点加法提出了一个大端先入的折叠计算模块。如下图所示为量化



Digit-serial hybrid radix-2 redundant adders. (a) Lsd-first adder; (b) msd-first adder.

图 13: Serial mrHY4A

精度为 6 个数位 (12 个二进制位) 的冗余浮点加法运算过程，为保持输出仍为 6 位的量化精度，该串行加法执行 6 次，串行输出 $S = A + B$ 的每一个冗余数位。其中，B 的阶码比 A 小 3 阶，故而在大端加法的前 3 个周期，只有数字 A 所属移位寄存器移位输出。在浮点数的串行加法中，大端先入是避免复杂对齐操作的关键，而冗余计算的大端先入串行加法具有和小端先入同等的复杂程度。

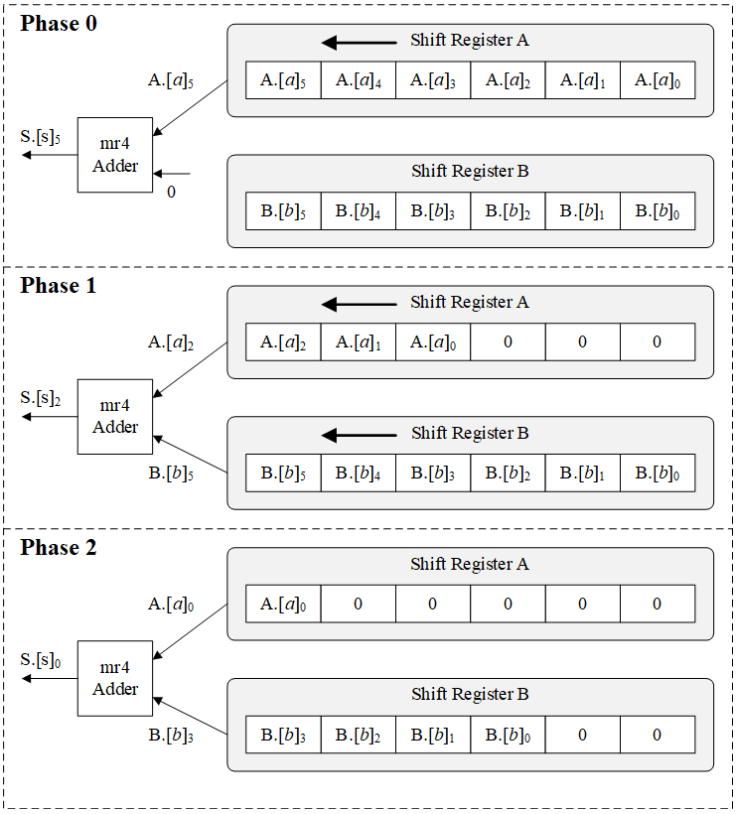


图 14: 冗余浮点加法运算过程

为配合大端先入的冗余浮点加法模块，本文继续设计了一款大端先入先出的串行冗余乘法单元，其主要电路结构入下图所示。图中乘法器输入为 FIR 数据点 X 和 FFT 单位根 R，其中在单次乘法过程中，R 为固定输入，而 X 所在的串行移位寄存器连接到 Booth 编码器 (Booth Encoder) 的输入端，完成对单位根 R 的 Booth 编码，编码输出作为当前输入与部分积 D 累加。因为冗余加法不存在进位传播，故而 D 的最高位 $D6$

可以直接作为当前乘积的有效位而输出。该结构迭代 6 次即可完成一次 6 数位 (12 bit 精度) 的乘法。

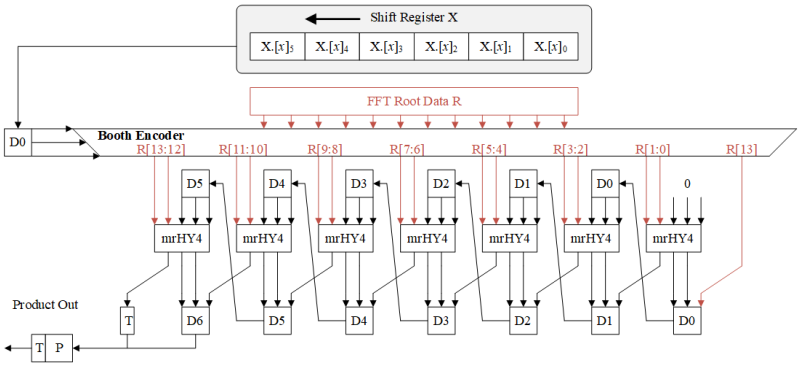


图 15: 冗余浮点乘法运算过程

Karatsuba 快乘　以下对于高精度大整数乘法的 Karatsuba 算法简介摘录自 OI Wiki:Karatsuba Algorithm。

如果取高精度数字 (必要时同时约化为大整数) 的位数为 n，那么高精度—高精度竖式乘法需要花费 $\mathcal{O}(n^2)$ 的时间。本节介绍一个时间复杂度更为优秀的算法，由前苏联（俄罗斯）数学家 Anatoly Karatsuba 提出，是一种分治算法。

考虑两个十进制大整数 x 和 y，均包含 n 个数码并可以有前导零。任取 $0 < m < n$，记

$$
\begin{aligned}
x &= x_1 \cdot 10^m + x_0, \\
y &= y_1 \cdot 10^m + y_0, \\
x \cdot y &= z_2 \cdot 10^{2m} + z_1 \cdot 10^m + z_0,
\end{aligned}
$$

其中 $x_0, y_0, z_0, z_1 < 10^m$。简单四则运算可得

$$
\begin{aligned}
z_2 &= x_1 \cdot y_1, \\
z_1 &= x_1 \cdot y_0 + x_0 \cdot y_1, \\
z_0 &= x_0 \cdot y_0.
\end{aligned}
$$

继续观察 (本质上使用分配律或 Wingoard 加速算法) 将

$$
z_1 = (x_1 + x_0)(y_1 + y_0) - z_2 - z_0
$$

从而只要计算出 $(x_1 + x_0), (y_1 + y_0)$ 再与 z_2, z_0 相减即可求出 z_1。

因为普遍讲计算机上乘法的耗时要大于甚至远大于加法 (尤其是数位越大的时候)。从而将乘法替换为加法会大大降低算法的渐进时间复杂度。

上式实际上是 Karatsuba 算法的核心，它将长度为 n 的乘法问题转化为了 3 个长度更小的子问题。若令

$$
m = \left\lceil \frac{n}{2} \right\rceil
$$

记 Karatsuba 算法计算两个 n 位整数乘法的耗时为 $T(n)$，则有

$$T(n) = 3 \cdot T\left(\left\lceil \frac{n}{2} \right\rceil\right) + \mathcal{O}(n)$$

由主定理可得

$$T(n) = \mathcal{O}(n^{\log_2 3}) \approx \mathcal{O}(n^{1.585})$$

这相比于通常的大整数乘法的 $\mathcal{O}(n^2)$ 有数量级的减少。也回答了 20 世纪 60 年代 Kolmogorov 关于大整数乘法渐进时间复杂度是否为 $\Omega(n^2) \approx \mathcal{O}(n^2)$ 的重大问题。

而 Anatoly Karatsuba 发明这个算法时年仅 23 岁，仅仅是在 Kolmogorov 陈述他关于任何大整数乘法算法的时间复杂度趋于 $\mathcal{O}(n^2)$ 这一猜测一周后 Karatsuba 就完成了上述更快算法的构造。具体的历史信息可参考 Wikipedia:Karatsuba Algorithm。其中一段摘录如下

> - In 1960, Kolmogorov organized a seminar on mathematical problems in cybernetics at the Moscow State University, where he stated the $\Omega(n^2)$ conjecture and other problems in the complexity of computation. Within a week, Karatsuba, then a 23-year-old student, found an algorithm that multiplies two n-digit numbers in $\mathcal{O}(n^{\log_2 3})$ elementary steps, thus disproving the conjecture. Kolmogorov was very excited about the discovery; he communicated it at the next meeting of the seminar, which was then terminated. Kolmogorov gave some lectures on the Karatsuba result at conferences all over the world (see, for example, "Proceedings of the International Congress of Mathematicians 1962", pp. 351–356, and also "6 Lectures delivered at the International Congress of Mathematicians in Stockholm, 1962") and published the method in 1962, in the Proceedings of the USSR Academy of Sciences. The article had been written by Kolmogorov and contained two results on multiplication, Karatsuba's algorithm and a separate result by Yuri Ofman; it listed "A. Karatsuba and Yu. Ofman"as the authors. Karatsuba only became aware of the paper when he received the reprints from the publisher.

更完整的第一手算法理论参考可见 Karatsuba 本人的综述 [Karatsuba 1995]。本项目使用其思想实现了针对滤波器乘法的版本如下，主要集成了 Karatsuba 算法实现了多项式乘法和卷积，最后再处理所有的进位问题。具体信息参考源代码或附录。

3.3 量化感知中的混合精度量化 (Mixed Precision Quantization)

此部分主要参考 "A Survey of Quantization Methods for Efficient Neural Network Inference" 的大量参考文献中实现的不同的量化感知技术。

混合精度量化 选取最优的数据混合精度, 硬件可以追求准确度与计算成本之间的平衡。

量化感知技术为硬件带来了效率提升。我们能够发现, 在低精度计算模式下, 量化感知技术可以让硬件计算速度完成提升。但我们发现, 对于一个元件而言, 全部计算参数的低精度处理常常会带来元件工作准确度的大幅下降。因此, 为了保证元件工作的准确性, 同时维持元件的高效运行, 不同硬件采用不同的计算精度成为了合适的选择。混合精度量化因此被提出, 旨在研究对于某一固定硬件架构, 应该选择什么样的低精度计算策略。

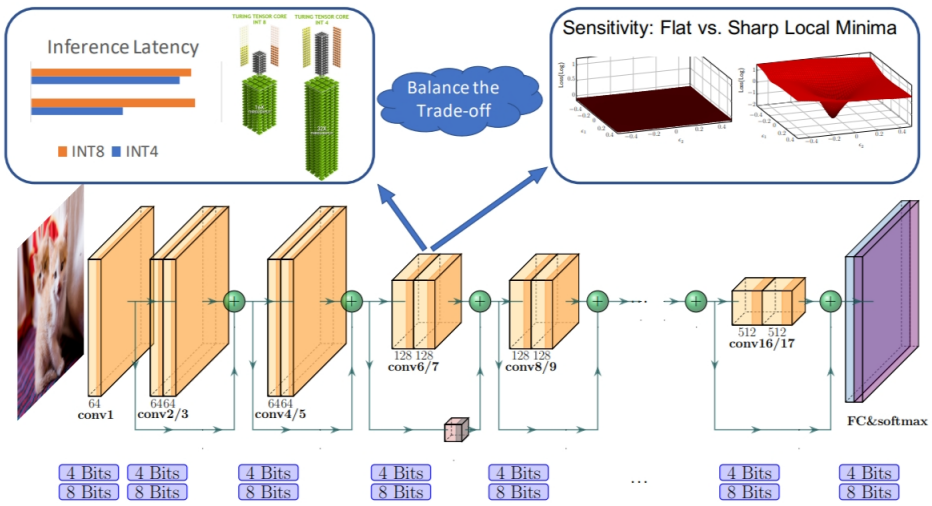


图 16: 一种混合精度神经网络架构

如图16所示, 整个神经网络架构的各个卷积层分别选取了不同的参数精度, 在不同参数精度的条件下分别进行训练。在训练过程中挑选效果最优的精度组合, 最终构成硬件结构的实际参数。

上述过程是混合精度量化的大体过程。对于混合精度量化模型的训练而言, 通常有不同的训练方式。

在量化模型训练过程中, 通常采用与全精度浮点模型不同的方式。如图17左表示的是浮点模型的训练方式。浮点模型直接通过浮点数运算进行网络训练。而计算机在处理量化模型时一般采用模拟量化训练与完全量化训练两种方式, 即图17剩余两部分的量化训练方法。

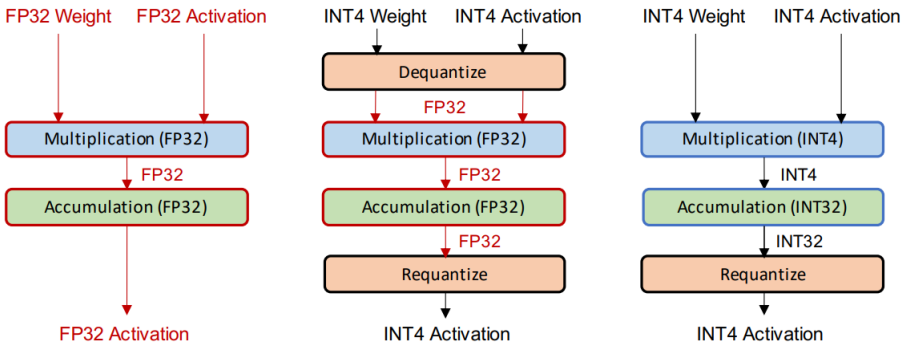


图 17: 量化神经网络的模型训练方法

模拟训练量化的全部参数被储存在不同精度格式下, 但在训练时首先通过计算将参数转化为浮点数, 再利用浮点数的训练模式进行参数训练。浮点数参数训练的优势在于其训练时非线性部分可以被有效近似, 训练比较稳定。

完全训练量化则是根据低精度的数据格式的计算来进行模型训练。在低精度下, 求导等运算的准确度会有所降低, 可能会引入较大的计算误差。因此, 寻求更为有效的低精度近似计算方法是提高量化模型训练效率的重要方向之一。

上述两种量化训练方式各有优势与劣势, 在不同的训练任务中需要根据侧重进行合适地选取。量化感知的混合精度量化技术仍然具有提升潜力, 更为有效的低精度计算方法将为混合量化模型的训练提供更加有效的提升。

4 实验数据及软硬件介绍

为了进行公平的比较，我们在 Xilinx Artix-7 FPGA 平台上，分别对 64、128 和 256 点的 FFT 硬件设计进行了实验，以探索速度和面积效率之间的不同权衡。为了考虑 DSP 和 BRAM，我们使用 Slice 等效成本 (SEC) 作为面积评估的度量。通过 SEC = #BRAMx200 + #DSPsx100 + #Slice 来计算 SEC，即一个 DSP 块和一个 36 Kb BRAM 分别等于 102.4 和 196.2 片。面积效率指标则由面积时间积 (ATP) 评估，计算为 # SEC × FFT 算法执行时间。我们将所提出的高效 FFT 流处理器性能与近期内和 “FIR’’、‘‘多项式乘法’’、‘‘傅里叶变换’’ 和 ‘‘蝶形运算’’ 等主题相关的硬件工作做对比。受益于我们所采用的冗余计算方案，即使在高并行度的执行环境下，我们的设计仍然可以跑到 200 Mhz、177 Mhz 和 162 MHz。为与所引工作进行公平比较，我们针对 $N = 256$ 点的 FFT 做了实现，其 ATP 评估结果为 1813.7，位列相关工作最优。此外，为充分论证设计在不同并行度下的执行性能和面积、速度权衡，我们也针对 $N = 64$ 以及 128 做了实现，以供参考。基于实现结果，我们推荐使用 $N = 32 \sim 128$ 的参数来加速 FIR 应用问题，以取得最佳的加速效果。

5 实验结果及主要结论

本部分主要展示我们团队的 FIR 滤波器算法测试数据。以下是和其他 IEEE 研究相比较的图表和结论。

5.1 主要测试结果

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.

5.2 结论

基本实现了第一部分的前两个技术诉求。目前本团队正努力实现第三个技术诉求。

6 总结及展望

有限长单位脉冲响应滤波算法是一种广泛应用在通信工程、音视频处理、信号处理等工程方向的硬件算法。其作为信号处理技术中的核心主流算法，已经被多系列主流基带芯片通信芯片所集成。如何高效执行有限长单位脉冲响应滤波算法，无论是从算法理论上还是硬件兼容软件集成上都是学界和工业界的主攻难题。

低功耗滤波器的算法实现主要依靠两部分性能的提升，一个是底层算法诸如乘法器和卷积运算器的性能提升，另一个是系统架构升级产生的运算次数减少/内存调度优化等方式的整体性能提升。数学上抽象即在保持一些例如稳定性、准确性的条件下优化

$$\min S_{\text{Integral}} + C_{\text{num+res}}, \qquad \text{s.t.} \qquad \text{Constraints}(\mathrm{r}_1, .., \mathrm{r}_\mathrm{k}) = 0$$

S_{Integral} 硬件架构运算器数量等整体功耗 (用功耗面积度量)

$C_{\text{num+res}}$ 底层算法功耗 (用单次运算消耗内存和时间的综合指标度量)

$r_1, ..., r_k$ 度量稳定性准确性和其他要求的指标

自适应滤波器原理是通过预测/经验方法实现诸如确定模型无损的方式是期望最小化如下的近似/预测/补偿误差

$$\text{Find} \qquad \widehat{y} \quad \text{s.t.} \, \|y - \widehat{y}\|_2 < \epsilon, \qquad \forall y \in \Omega_D$$

$y, \widehat{y},$ 分别表示测试信号数据和补偿信号

$\epsilon \to F_\epsilon = \widehat{(-)},$ 表示针对不同的精度要求能够开发不同的自适应补偿算子F_ϵ

量化感知是近二三十年来发展迅速的加速神经网络训练的主流技术之一。从提出至今已经有不同研究者提出了种类繁多的量化策略其想要解决的主要问题是在一些性能限制下最小化如下的量化后数据和原始数据的偏差

$$\min \sum_r \|Q(r) - r\|_2, \qquad \text{s.t.} \quad \text{Constraints}(\mathrm{r}) = 0$$

高效的量化感知算法可以大大加快神经网络训练的底层运算速度，是一个加速神经网络训练的可行方向。但是另一方面，神经网络自身的结构优劣和误差传播算法的设计好坏也是制约其训练质量和速度的重要指标。从这一意义上说，设计更好的神经网络结构和与之配套的误差传播/激励传播算法可能会从另一个方向优化神经网络的训练。这种优化可能达到量化感知技术无法实现的程度。

综观工程技术发展历史，最底层的算法和技术迭代往往艰深险难，每进步一毫可能都要耗费巨大的时间人力成本，更甚者算法层面的进步可能需要数学理论的艰难创新或是天才般的灵光乍现。本文集成了快速傅立叶变换算法、低功耗约束的高效规划算法、量化感知加速技术等诸多经典或者前沿的工程算法，其中不妨极具影响力的 '世界级' 算法。经过软硬件整合实现一套可以在华为系硬件芯片上运行的完整的自适应低功耗有限长单位脉冲响应滤波算法 (Self-Adjusting Low-Dissipation Finite Impulse Response Filter)

计划中附录中包含本项目工程的源代码实现以及注释；量化感知技术实现的神经网络框架 (基于 PyTorch/Tensorflow) 源代码实现和接口注释；以及详细的操作方式。(API Source Code;Description;Manual)

如果有后续的经费支持，我们团队认为我们可能继续维护该项目的开发，并且生产出更加高效的算法。

特别地，我们团队将集中处理利用神经网络进行特定场景的自适应滤波器算法设计的研究和实现上。因为在我们近一个月的调研和测试中，我们发现底层算法的性能突破很大程度依赖于硬件自身，另一大部分是要求设计者对离散调和分析理论有极深的造诣。这才有可能设计出超过本文第二三部分所述的经典或部分改良的底层理论算法。

未来本团队可能的进一步研究方向包括如下几个方面：

1. 完善 FIR 滤波算法的不同场景的适应性，以及去耦合、模块化整个 ACFIR 滤波程序项目。
2. 接口专家提供数据集、构建神经网络对 FIR 滤波器自适应功能进行训练。
3. 更完善的前端呈现、包括基于 Qt 的网络训练可视化和接口化的 FIR Filter 硬件架构/测试可视化工程。

后续的难题攻克需要更多的技术支持和经费支持，也希望本文回答了此题目的技术诉求。期待能够得到华为技术有限公司的后续资助。

本团队的全部项目代码均按照 MIT License 开源。

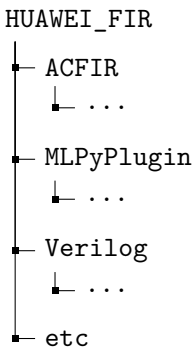
项目仓库地址是 https://github.com/YOTALTEAM/HUAWEI_FIR。欢迎有关人士对项目提出宝贵意见建议。

附录

这是附录文件。有待于后续更新。

6.1 源程序文件树

全工程文件树如下图所示



其中 ACFIR 文件夹是可运行在硬件或仿真系统上的 FIR 滤波算法源代码；MLPyPlugin 文件夹是优化滤波器性能的机器学习 Python 代码文件；VDHL 是用于仿真模拟或硬件测试的硬件描述语言源代码。“etc”表示其他诸如测试数据，日志文件，编译软件生成文件等附加的文件。“...”表示文件夹中存在的没有列举出的文件。

关于本论文和后续汇报的幻灯片文件将在 GitHub 团队 地址 上的 Contest-Piece 仓库中的 “华为领航杯论文” 文件夹中发布。由于自身嵌套问题此文件恕不在附录的源程序文件解释部分解释，也于文件树上省略此文件夹。

同时此比赛的一切源代码均发布到指定邮箱。

>>> 此处添加 ACFIR 文件树

transformation.h 此为 FIR 使用的变换函数的声明文件。为 C++ 头文件 (Header File)。

6.1.1 滤波器代码注解

下面的 C++ 函数实现了滤波器的基准测试功能。

使用信号处理的名字空间

convolution.cpp

```
#include "convolution.h"

namespace dsplib{
    namespace conv{
        vfp fir_regular(const vfp& seq, const vfp& tap) {
            int slen = seq.size();
            int tlen = tap.size();
            int olen = slen + tlen - 1;
```

```
                    vfp vo(olen);
                    for(int i = 0; i < olen; i++) {
                        vo[i] = 0;
                        for(int t = 0; t <= i; t++) {
                            if((t < tlen) && ((i - t) < slen))
                                vo[i] += tap[t] * seq[i - t];
                        }
                    }
                    return vo;
                };
            }
        }
```

6.1.2 机器学习框架代码注解

本项目使用 PyTorch 包对滤波器参数选择进行神经网络优化。

6.1.3 可视化框架注解

待更新

6.2 使用指南

6.2.1 操作系统和软件要求

参考源代码或 GitHub 仓库。

6.2.2 执行步骤

参考源代码或 GitHub 仓库。

参考文献

BishehNiasar2021 Mojtaba Bisheh Niasar, Reza Azarderakhsh, and Mehran Mozaffari Kermani. "Instruction Set Accelerated Implementation of CRYSTALS-Kyber". In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 68.11 (2021), pp. 4648–4659 (cit. on p. 25).

Chen2021 Zhaohui Chen et al. "High-performance area-efficient polynomial ring processor for CRYSTALS-Kyber on FPGAs". In: *Integration* 78 (2021), pp. 25–35 (cit. on p. 25).

Cormen2022 Thomas H Cormen et al. *Introduction to algorithms*. 4th ed. MIT press, 2022 (cit. on pp. 2, 10, 12).

Gholami2021 Amir Gholami et al. "A Survey of Quantization Methods for Efficient Neural Network Inference". In: (Mar. 2021). arXiv: 2103.13630 [cs.CV] (cit. on pp. 2, 21).

Guo2021 Wenbo Guo, Shuguo Li, and Liang Kong. "An Efficient Implementation of Kyber". In: *IEEE Transactions on Circuits and Systems II: Express Briefs* (2021) (cit. on p. 25).

Jiang2020 Honglan Jiang et al. "Approximate Arithmetic Circuits: A Survey, Characterization, and Recent Applications". In: *Proceedings of the IEEE* 108.12 (2020), pp. 2108–2135. DOI: 10.1109/JPROC.2020.3006451 (cit. on p. 2).

Karatsuba1995 Anatolii Alexeevich Karatsuba. "The complexity of computations". In: *Proceedings of the Steklov Institute of Mathematics-Interperiodica Translation* 211 (1995), pp. 169–183 (cit. on pp. 2, 21).

Li2010 Chao Li et al. *Mathematical Principle of Computer Algebra System*. TsingHua University Press, 2010. ISBN: 9787302230106. URL: https://mathmu.github.io/MTCAS/mtcas.pdf (cit. on pp. 2, 16).

Liu2019 Weiqiang Liu et al. "Approximate designs for fast Fourier transform (FFT) with application to speech recognition". In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 66.12 (2019), pp. 4727–4739 (cit. on p. 11).

Ma2021 Liejun Ma, Xingjun Wu, and Guoqiang Bai. "Parallel polynomial multiplication optimized scheme for CRYSTALS-Kyber Post-Quantum Cryptosystem based on FPGA". In: *2021 International Conference on Communications, Information System and Computer Engineering (CISCE)*. IEEE. 2021, pp. 361–365 (cit. on p. 25).

Nussbaumer1982 Henri J. Nussbaumer. *Fast Fourier Transform and Convolution Algorithms*. Springer Berlin Heidelberg, 1982. DOI: 10.1007/978-3-642-81897-4 (cit. on pp. 2, 16).

Parhi2007 Keshab K Parhi. *VLSI digital signal processing systems: design and implementation*. John Wiley & Sons, 2007 (cit. on p. 8).

Winograd1968 S. Winograd. "A New Algorithm for Inner Product". In: *IEEE Trans. Comput.* 17.7 (June 1968), pp. 693–694. ISSN: 0018-9340. DOI: 10.1109/TC.1968.227420. URL: https://doi.org/10.1109/TC.1968.227420 (cit. on p. 2).

Yarman2021 Ferhat Yarman et al. "A hardware accelerator for polynomial multiplication operation of CRYSTALS-Kyber PQC scheme". In: *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE. 2021, pp. 1020–1025 (cit. on p. 25).