目录

1. 概述

ACM32F4/FP4/F3 系列芯片的内核基于 ARMv8-M 架构，支持 Cortex-M33 和 Cortex-M4F 指令集。内核支持一整套 DSP 指令用于数字信号处理，支持单精度 FPU 处理浮点数据，同时还支持 Memory Protection Unit（MPU）用于提升应用的安全性。

内核处理器基于 ARMv8-M 架构。处理器包括两个总线接口分别称为 C-AHB 总线、S-AHB 总线：

C-AHB 总线：用于访问 ARMv8-M 存储架构下代码区的指令或数据。

S-AHB 总线：用于访问 ARMv8-M 存储架构下 SRAM 区、外部 RAM 区、外设区或厂商自定义系统区的指令或数据。

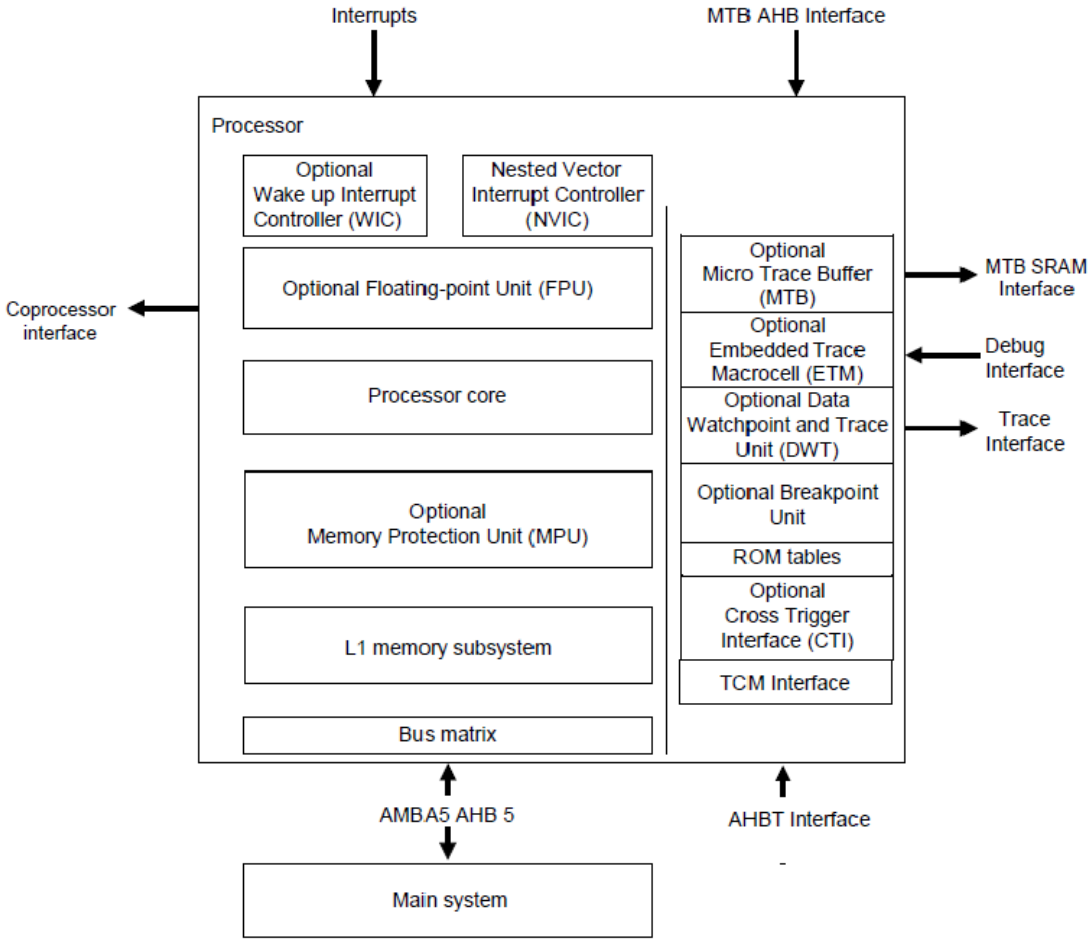

图 1-1 处理器结构

2. CMSIS 的 DSP 库介绍

2.1. CMSIS DSP 库功能说明

CMSIS-DSP 库包含一组常用的信号处理和数学运算函数，该库位于 ARM 的 CMSIS 包中。CMSIS DSP 库大部分函数都是支持 f32，Q31，Q15 和 Q7 四种格式的。该库分为以下几个功能：

- 基本数学运算函数：BasicMathFunctions
- 快速数学运算函数：FastMathFunctions
- 复杂数学运算函数：ComplexMathFunctions
- 滤波器函数：FilteringFunctions
- 矩阵函数：MatrixFunctions
- 数学变换型函数：TransformFunctions
- 马达控制函数：ControllerFunctions
- 统计型函数：StatisticsFunctions
- 支持型函数：SupportFunctions

2.2. DSP 库文件

ARM 官方已经封装好了针对不同内核的 Lib，位于 CMSIS/Lib 目录下，且分别针对不同的平台。我们这里只讨论 ARM 平台。

对于这些库的解释，可以参考官方的说明文档：CMSIS DSP Software Library (keil.com)。我们在使用的时候，可以根据不同的内核选取对应的 lib 文件。

ACM32F4/F3/FP4 使用的 DSP 库文件为 arm_ARMv8MMLldfsp_math.lib。

- arm_cortexM7lfdp_math.lib (Cortex-M7, Little endian, Double Precision Floating Point Unit)
- arm_cortexM7bfdp_math.lib (Cortex-M7, Big endian, Double Precision Floating Point Unit)
- arm_cortexM7lfsp_math.lib (Cortex-M7, Little endian, Single Precision Floating Point Unit)
- arm_cortexM7bfsp_math.lib (Cortex-M7, Big endian and Single Precision Floating Point Unit on)
- arm_cortexM7l_math.lib (Cortex-M7, Little endian)
- arm_cortexM7b_math.lib (Cortex-M7, Big endian)
- arm_cortexM4lf_math.lib (Cortex-M4, Little endian, Floating Point Unit)
- arm_cortexM4bf_math.lib (Cortex-M4, Big endian, Floating Point Unit)
- arm_cortexM4l_math.lib (Cortex-M4, Little endian)
- arm_cortexM4b_math.lib (Cortex-M4, Big endian)
- arm_cortexM3l_math.lib (Cortex-M3, Little endian)
- arm_cortexM3b_math.lib (Cortex-M3, Big endian)
- arm_cortexM0l_math.lib (Cortex-M0 / Cortex-M0+, Little endian)
- arm_cortexM0b_math.lib (Cortex-M0 / Cortex-M0+, Big endian)
- arm_ARMv8MBLl_math.lib (Armv8-M Baseline, Little endian)
- arm_ARMv8MMLl_math.lib (Armv8-M Mainline, Little endian)
- arm_ARMv8MMLlfsp_math.lib (Armv8-M Mainline, Little endian, Single Precision Floating Point Unit)
- arm_ARMv8MMLld_math.lib (Armv8-M Mainline, Little endian, DSP instructions)
- arm_ARMv8MMLldfsp_math.lib (Armv8-M Mainline, Little endian, DSP instructions, Single Precision Floating Point Unit)

图 2-1 ARM 官方库文件

3. DSP 库移植到 ACM32F4/FP4/F3

3.1. 硬件环境

硬件测试环境基于航芯的 ACM32F4 系列开发板。如下图:

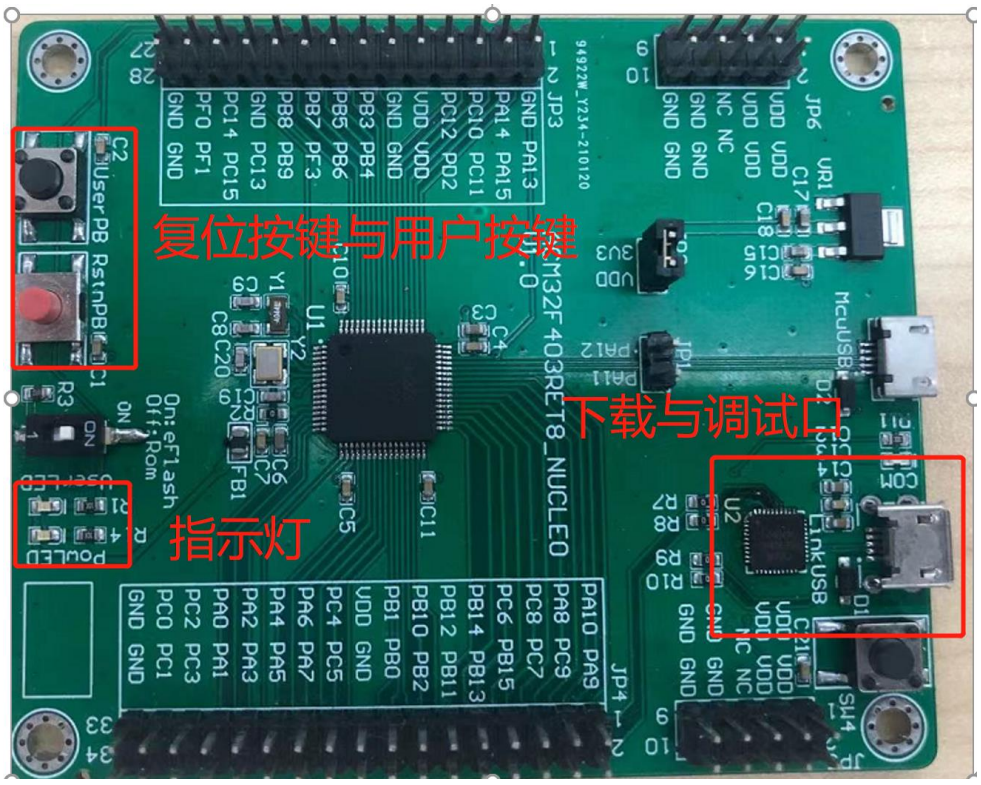

图 3-1 开发板

3.2. 使用 lib 库的形式

由于 ACM32F4 使用的是 ARMV8 架构，支持单精度（Single Precision）运算，支持 DSP 算法。所以这里我们选取 arm_ARMv8MMLldfsp_math.lib，对应的头文件为 arm_math.h。

如 DSP->demo_arm_class_marks 中的工程所示，将库文件 arm_ARMv8MMLldfsp_math.lib 增加到工程中，在 app.c 中增加 arm_math.h 头文件，就可以调用库中的函数了。

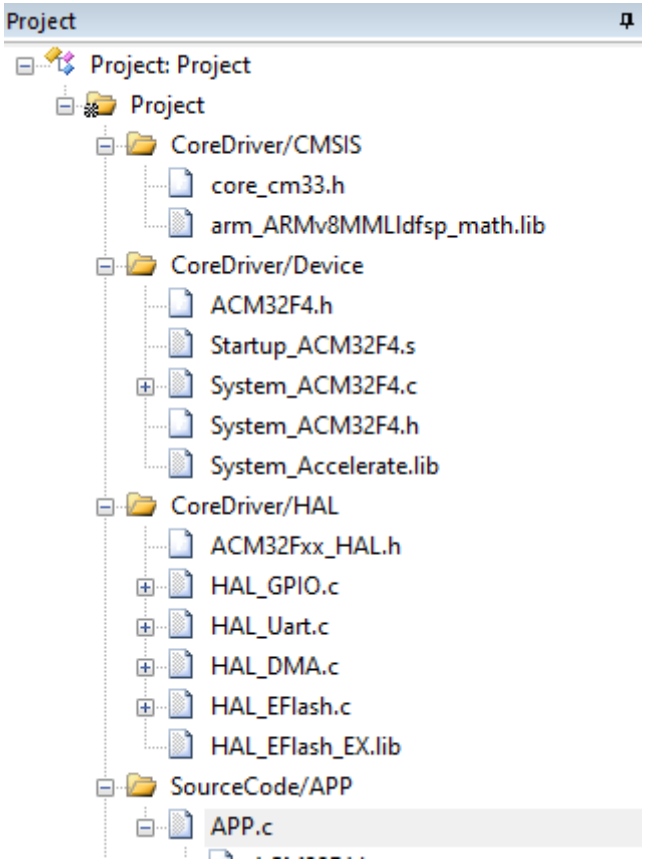
Project
Project: Project
Project
CoreDriver/CMSIS
core_cm33.h
arm_ARMv8MMLldfsp_math.lib
CoreDriver/Device
ACM32F4.h
Startup_ACM32F4.s
System_ACM32F4.c
System_ACM32F4.h
System_Accelerate.lib
CoreDriver/HAL
ACM32Fxx_HAL.h
HAL_GPIO.c
HAL_Uart.c
HAL_DMA.c
HAL_EFlash.c
HAL_EFlash_EX.lib
SourceCode/APP
APP.c

图 3-2 示例工程

4. 基本示例展示

4.1. 点积运算示例

本例程主要展示如何使用相乘和相加来实现点积。两个向量的点积是通过将对应元素相乘并相加来获得的。

这里以一维向量的点积为例：

若 A 和 B 均为一维向量，且均包含有 n 个元素，则 A 与 B 的点积为： A[0]*B[0]+A[1]*B[1]+...+A[n]*B[n]。

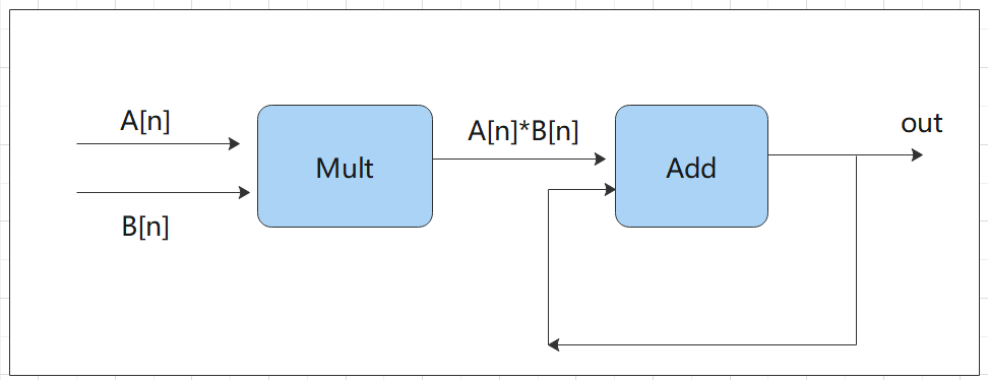

图 4-1 点积运算

对应的参考代码为：demo_arm_dotproduct，测试函数为 APP_ARM_Dotproduct_Test()，其中各参数说明如下：

- srcA_buf_f32 相当于 A[n]:指向第一个输入变量
- srcB_buf_f32 相当于 B[n]:指向第二个输入变量
- testOutput：指向点积的结果

使用到的 DSP 库函数有：

- arm_mult_f32()
- arm_add_f32()

4.2. FIR 低通滤波器

FIR 滤波器应用于多种音频、视频和数据分析中。通常使用 FIR 低通滤波器去除输入信号的高频部分，本示例展示了如何配置 FIR 滤波，然后以块方式传递数据。

示例用法：

输入信号为两个正弦波 1KHz 和 15KHz 的叠加。输入信号被截至频率为 6KHz 的进行低通滤波，滤掉 15KHz 的信号，只留下 1KHz 信号。

本例程的低通滤波器采用 MATLAB 设计，对应的代码为：h = fir1(28, 6/24)，采样率为 48kHz,长度为 29 个点。

- 第一个参数是过滤器的“顺序”，并且总是比所需长度小 1；
- 第二个参数是归一化截止频率。范围是 0（DC）到 1.0（Nyquist）。24 kHz 奈奎斯特频率的 6kHz 截止频率为 6/24=0.25 归一化频率。

CMSIS FIR 滤波器函数要求系数按时间倒序排列。所得滤波器系数如下图所示。需要注意的是，该滤波器是对称的（线性相位 FIR 滤波器的属性）。对称点是样本 14，对于所有频率，该滤波器具有 14 个样本的延迟。

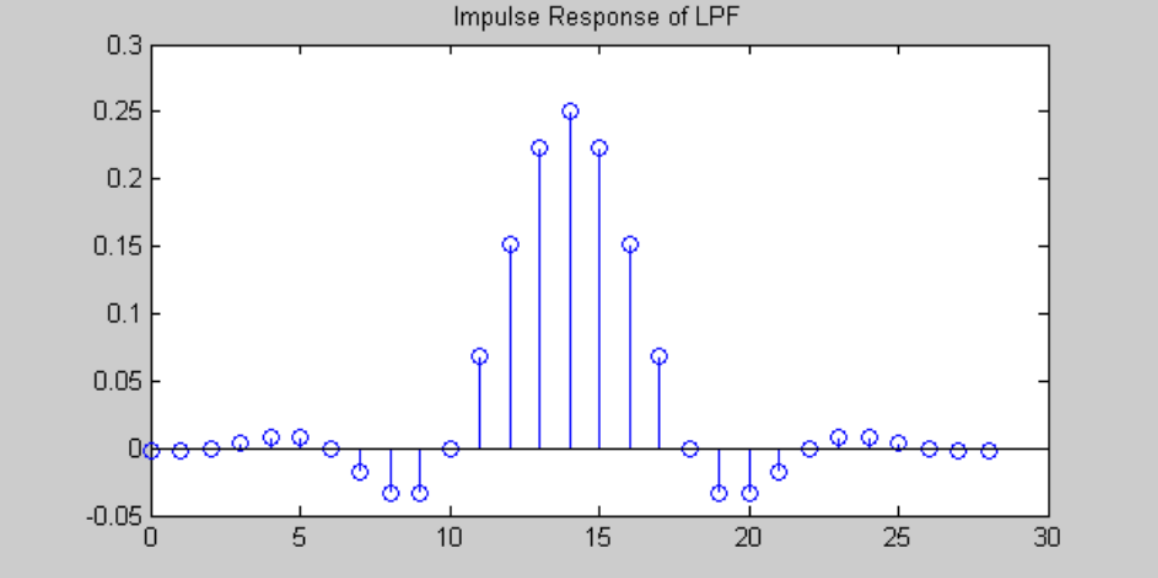

图 4-2 低通滤波

然后是滤波器的频域响应，滤波器的带通增益为 1.0，截止频率为 6kHz 时达到 0.5。

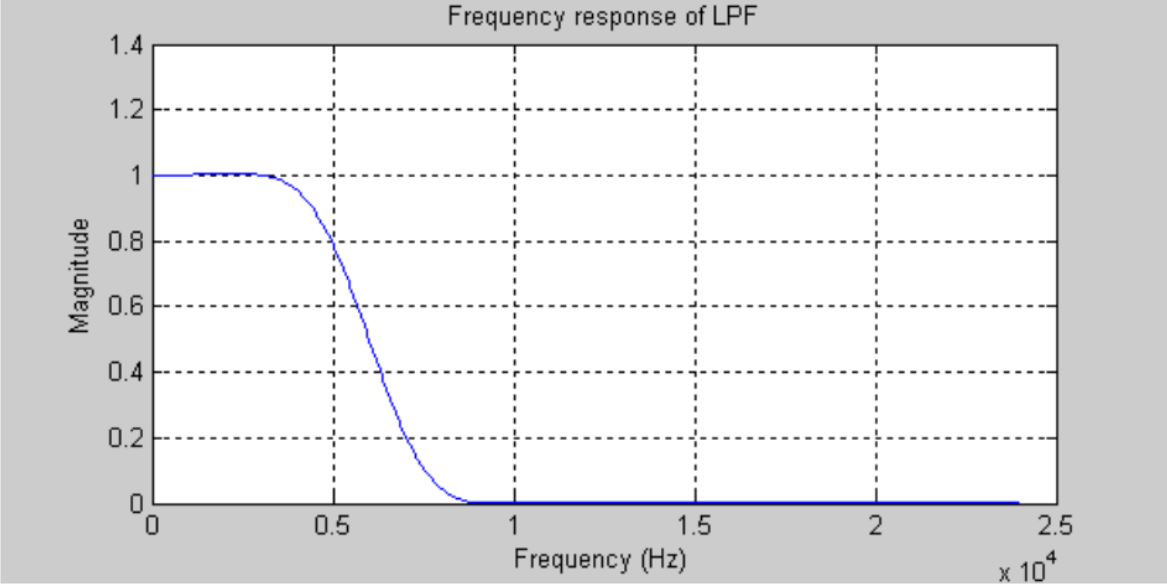

图 4-3 低通滤波频域

输入信号如下所示。左侧显示时域信号，右侧显示频域。可以清楚的看到两个正弦波分量。

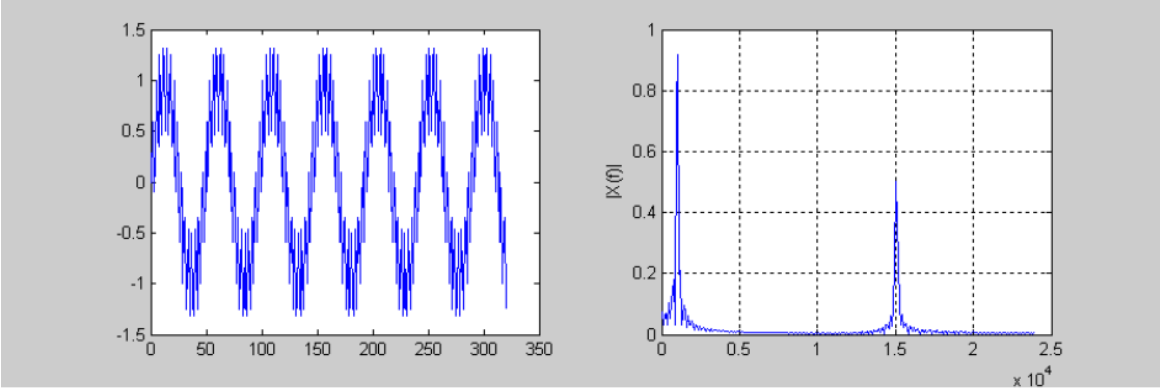

图 4-4 输入信号的时域和频域

这样可以得到 15KHz 被滤除的输出波形。

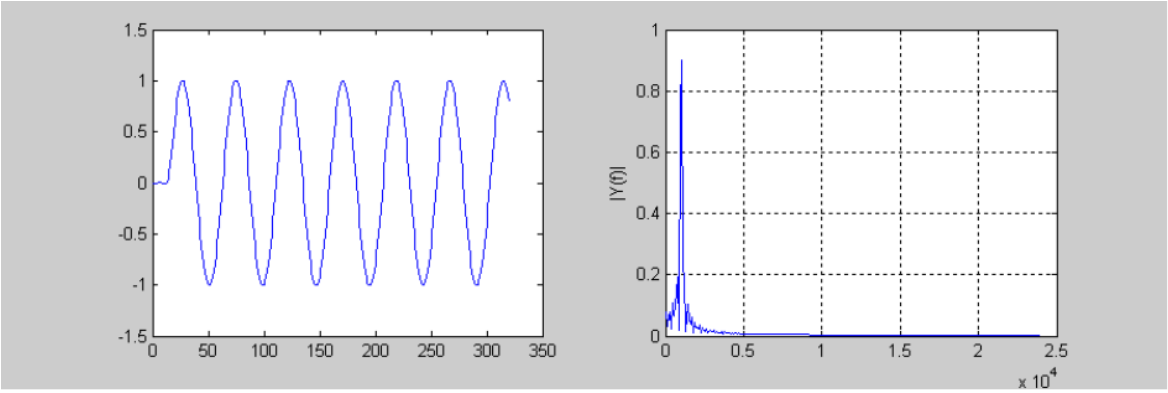

图 4-5 输出信号的时域和频域

对应的参考代码为：demo_arm_fir,测试函数为 APP_ARM_Fir_Test()。其中各参数说明如下：

- testInput_f32_1kHz_15kHz：输入的 1KHz 和 15KHz 的叠加采样信号
- refOutput：参考输出数据
- testOutput：测试输出数据
- firStateF32：状态缓冲区
- firCoeffs32：系数缓冲区
- blockSize：一次处理的样本数
- numBlocks：测试的帧数

使用到的 DSP 库函数有：

- arm_fir_init_f32()
- arm_fir_f32()

4.3. 卷积运算示例

卷积是信号处理中的一种重要的运算。一个输入信号，经过一个线性系统（可以理解为卷积运算）以后，得到输出信号。

卷积理论指出，时域中的卷积对应频域中的乘法。因此，两个信号的卷积后的傅里叶变换等于他们各自的傅里叶变换的乘积。使用快速傅里叶变换（FFT）可以有效的评估信号的傅里叶变换。

本示例主要展示基于复数 FFT、复数乘法与支持函数的卷积理论：两个输入信号 a[n]和 b[n]填充为零，n1 和 n2 分别对应其信号长度。因此他们的长度将变为 N，N 大 于或等于 n1+n2-1。由于采用基 4 变换，因此基数为 4。a[n]和 b[n]的卷积是通过对输入信号进行 FFT 变换，对联更新好进行傅里叶变换。并对相乘后的结果进行逆 FFT 变换来获得的。相关计算公式如下：

A[k] = FFT(a[n],N)

B[k] = FFT(b[n],N)

conv(a[n], b[n]) = IFFT(A[k] * B[k], N)

其中 A[k]和 B[k]分别是信号 a[n]和 b[n]的 N 点 FFT。卷积长度为 n1+n2-1。

示例运算流程框图：

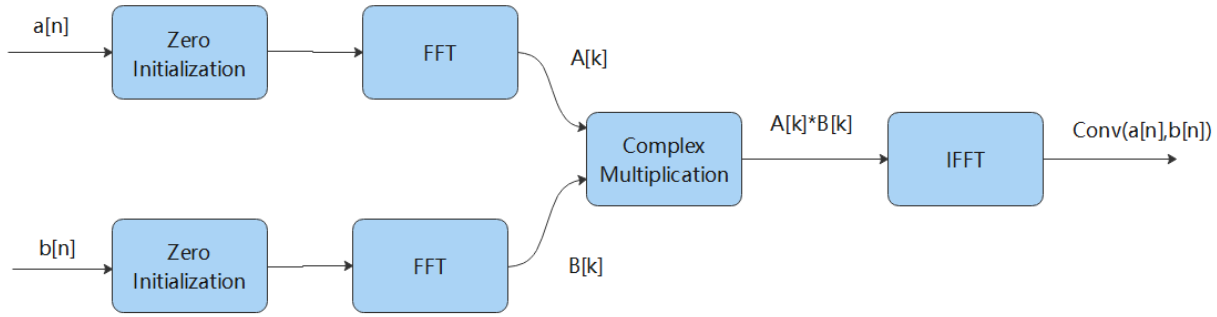

图 4-6 卷积示例流程

对应的参考代码为：demo_arm_convolution,测试流程函数 APP_ARM_Convolution_Test()。其中各参数说明如下：

- testInputA_f32 ： 第一个输入序列
- srcALen：第一个输入时序的长度
- testInputB_f32：第二个输入序列
- srcBLen：第二个输入序列的长度
- outLen：卷积输出序列的长度(srcALen + srcBLen - 1)
- AxB：指向 FFT 乘积后输出数组地址

使用到的 DSP 库函数有：

- arm_fill_f32()
- arm_copy_f32()
- arm_cfft_radix4_init_f32()
- arm_cfft_radix4_f32()
- arm_cmplx_mult_cmplx_f32()

4.4. 正弦和余弦运算示例

ARM DSP 库支持正弦和余弦等三角运算。本例程通过使用正弦，余弦，向量乘法和向量加法函数演示三角学的勾股定理。公式如下：

sin(x) * sin(x) + cos(x) * cos(x) = 1，其中 x 为弧度值。

示例运算流程框图：

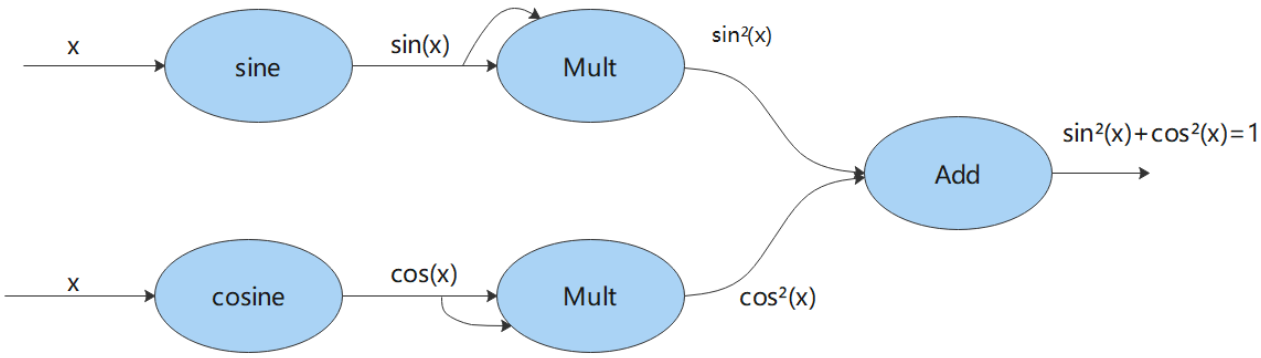

图 4-7 正余弦

对应的参考代码为：demo_arm_sin_cos,测试流程函数 APP_ARM_Sin_Cos_Test()。其中各参数说明如下：

- testInput_f32:角度输入数组
- testOutput：正弦和余弦的平方和

使用到的 DSP 库函数有：

- arm_cos_f32()
- arm_sin_f32()
- arm_mult_f32()
- arm_add_f32()

4.5. 方差运算示例

方差是在概率论和统计方差衡量随机变量或一组数据时离散程度的度量，方差等于各个数据与其算术平均数的离差平方和的平均数。

本示例演示如何使用基本函数和支持函数来计算 N 个样本的输入序列的方差，将均匀分布的白噪声作为输入。相关公式如下：

variance = ((x[0] - x') * (x[0] - x') + (x[1] - x') * (x[1] - x') + ... + * (x[n-1] - x') * (x[n-1] - x')) / (N-1)

其中，x[n]是输入序列，N 输入样本数，x 是输入序列 x[n]的平均值。 平均值 x 的定义如下：

x' = (x[0] + x[1] + ... + x[n-1]) / N

示例运算流程框图：

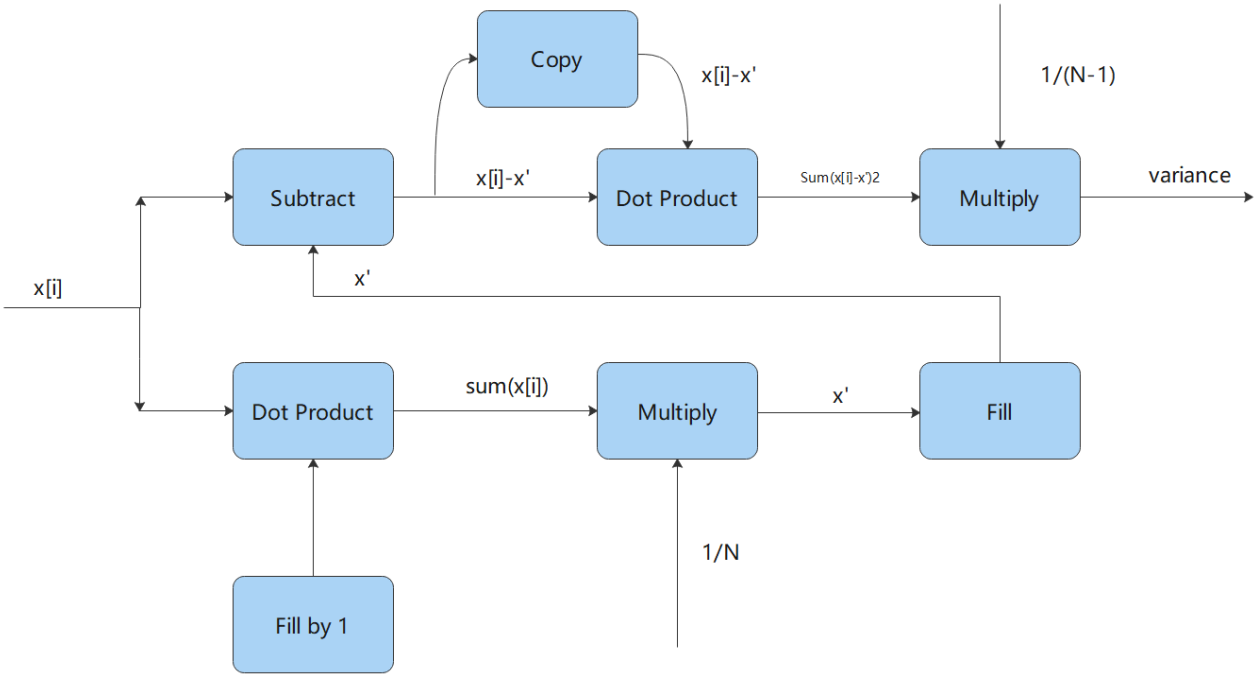

图 4-8 方差流程示例

对应的参考代码为：demo_arm_variance，测试流程函数 APP_ARM_Variance_Test()。其中各参数说明如下：

- testInput_f32:输入数据
- wire1, wir2, wire3：临时数据缓冲区
- blockSize：单次处理的样本数
- refVarianceOut：参考方差值

使用到的 DSP 库函数有：

- arm_dot_prod_f32()
- arm_mult_f32()
- arm_sub_f32()
- arm_fill_f32()
- arm_copy_f32()

4.6. 班级成绩统计

本例程演示使用最大，最小，均值，标准差，方差和矩阵函数来统计一个班级的成绩，还演示了静态初始化的用法。

对应的参考代码为：demo_arm_class_marks，测试流程函数 APP_ARM_Class_Marks_Test()。其中各参数

说明如下：

- testMarks_f32: 指向 20 名学生在 4 门学科中获得的分数
- max_marks:最高分成绩
- min_marks：最低分成绩
- mean：所有成绩的平均分
- var：所有成绩方差
- std：标准差
- numStudents：学生总成绩
- testOutput：存放每个同学四门课的总分数

使用到的 DSP 库函数有：

- arm_mat_init_f32()
- arm_mat_mult_f32()
- arm_max_f32()
- arm_min_f32()
- arm_mean_f32()
- arm_std_f32()
- arm_var_f32()

该例子在最前面展示了用内在函数运行饱和 SIMD 指令（单指令多数据）的方法：

simd_test = __UQADD16(0x1122AABB, 0x33448899);

该内在函数会调用汇编指令 UQADD16, UQADD16 是无符号饱和 16 位加法，低 16 位饱和相加的结果和高 16 位饱和相加的结果相或产生最终的结果。低 16 位数据相加的值已经超出 16 位无符号数的最大值，但因为是饱和运算，因此值为 0xFFFF;高 16 位的数据相加为 0x4466。

联系我们

公司：上海爱信诺航芯电子科技有限公司
地址：上海市闵行区合川路 2570 号科技绿洲三期 2 号楼 702 室
邮编：200241
电话：+86-21-6125 9080
传真：+86-21-6125 9080-830
Email： Service@AisinoChip.com
Website： www.aisinochip.com

版本维护

版本	日期	作者	描述
V1.0	2021-07-08	Aisinochip	初始版