基于串行数据的全并行FFT

1. 基于串行数据的全并行FFT蝶形运算电路

对于基的FFT而言，可以利用流水的蝶形运算电路实现计算，比如一个8点的FFT蝶形运算电路如图所示。蝶形运算电路的好处是运算较快，结构简单，但随着的加大，硬件消耗增长非常快。

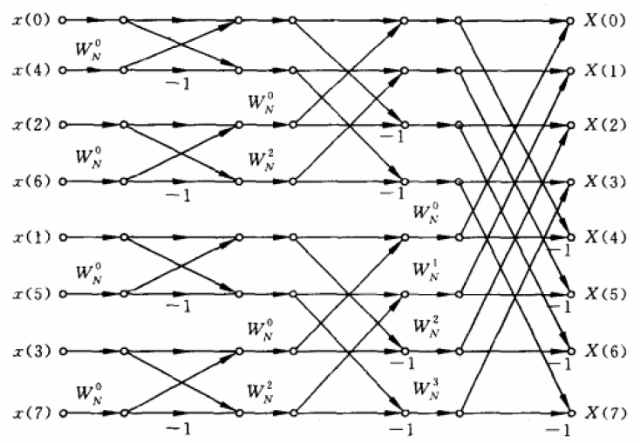


图1. 8点FFT蝶形运算电路

一般的电路设计之后数据位宽是一定的，比如固定处理16位二进制补码。为了实现的快速的FFT硬件计算，同时支持不同数据位宽长度，可以基于蝶形运算电路做出一定改进：

1. 数据输入为串行输入。这样电路中的加法器可以是单比特全加器，由于蝶形运算中FFT的旋转因子都是常数，可以利用移位加来实现串行输入的常数乘法器。这样输入数据的长度改变不会改变电路硬件，只需要更改电路控制逻辑。
2. 可以对乘法器进行改进以减少移位加的数量，从而进一步减少电路硬件开销
3. 利用正则符号数CSD改进乘法器

由于FFT中的乘法都是涉及到常数乘法，则可以利用移位加的办法来实现乘法。常系数乘法所需要的加法次数等于该常系数中非零位数减一，所以可以通过使得常系数非零位最少来减少硬件消耗。

正则有符号数（canonic signed digit, CSD）与二进制补码相似，是一种权重码，不同则是二进制补码每个位上是0或者1，而CSD数扩充为{-1,0,1}。CSD数有一系列性质[1]：

1. CSD数不存在连续的两个非零位。
2. 一个数的CSD数表示中含有的非零位是最少的
3. CSD数表示唯一
4. 在区间[-1,1)中CSD数的非零位平均比二进制补码少33%

例如在一个8点FFT运算中，涉及到对0.7071的常系数乘法：



0.707的二进制数为0.101101010000010，CSD数为1.0-0-01010000000（-表示-1），-0.707的CSD数为-.01010-0-00000-0。所以移位加的式子为：



电路实现上，信号进入之后分为原码和补码，取补码采用取反加1。假设第一拍数据到达，7拍到达时两个单比特全加器同时开始计算和，第8拍计算这两路的和，第9时刻开始计算与两路和的结果输出。



对每个CSD数统一截9位，其中一位是整数位，则乘法器的处理时延为9拍，以32点FFT为例，如果以二进制补码来表示，则有290个非零位，而CSD数表示时有208个非零位。

2015.8.28更新

关于旋转因子的常数乘法器的运算规则

1. 旋转因子的CSD数取值为16位（暂定，其中一位常数位），乘法为输入序列对应CSD数的非零位来移位相加。
2. 从CSD数最高的非零位对应的移位序列开始，每有一个非零位对应的移位序列第一个数字到达，就做一次加法。比如CSD数为0.010-001，最高非零位对应的序列是原序列移位两位的，第二个非零位对应序列是，所以当第五个数字到达的时候进行第一次加法，到达时进行。
3. 加法器为一个全加器：当进行加法时，串行加法器的Cout反馈到Cin参与下一拍的高位计算。进行减法时，减数那一路先取反，然后进位Cin初始值为‘1’。



