**基2-N点FFT的FPGA实现**

## 1.1 前言

FFT一般有多种计算方式，常见的为基2-DIT-FFT与基2-DIF-FFT。两种不同计算方式的最大区别是数据输入和输出的位序，在DIT-FFT中，输入序列为“码位倒读”顺序，输出序列为自然顺序，在DIF-FFT中序列顺序则相反。

|  |  |  |
| --- | --- | --- |
|  | |  |
| 图1.1 DIT-FFT流程图 | 图1.2 DIF-FFT流程图 | |

当然，通过改变运算顺序和增加中间存储器，也可以实现输入和输出序列均为自然顺序，但是这样会额外增加FPGA资源与时钟周期。在基2DIT-FFT方法中，蝶形算子的两个输入与输出的地址相同，即同址运算。因此使用一个深度为N、位宽为2M的RAM作为数据存储空间即可，N为计算FFT的点数，也是信号的长度；M为数据实部与虚部的位宽大小；高M位存储数据实部，低M位存储数据虚部。

虽然使用同址运算会使得程序更加复杂，但为尽可能减少FFT完成的时间与消耗资源，以给后续的网口TCP通信实现提供足够的资源与时钟盈余，这里采用DIT-FFT实现计算。

## 1.2 蝶形运算模块Butterfly

在图1.1的DIT-FFT运算中，最小的运算单元为一次蝶形运算。蝶形运算结构如图1.3所示：

|  |
| --- |
|  |
| 图1.3 蝶形算子结构 |

图1.3中的A、B均为复数形式的输入，C为本次蝶形运算旋转因子.设

由于在DIT-FFT中，蝶形运算为同址运算，因此数据A、B计算得到的数据A+BC、A—BC仍然是保存在原始A、B存储的地址。

设。在一次蝶形运算中，需要完成四次乘法，即B、C的实、虚部之间互相相乘；待乘法结束后，数据需要进行两次加减法运算。蝶形算子计算大致过程如图1.4所示：

|  |
| --- |
|  |
| 图1.4 蝶形算子完成一次运算过程 |

在计算时，数据Data A、B的位宽均为24位，旋转因子Data C位宽为16位，由于。在每一次蝶形运算中，旋转因子均扩大了倍，并提前计算好，保存在ROM中，供每次蝶形运算进行选择。相应地，Data A同样需要扩大倍。由图1.4可知，从RAM读到地址输出数据到最后蝶形运算计算完成，需要3个Cycle.具体如下：

1）控制模块RAM\_Ctrl输出的地址位与时钟上升沿保持同步输入至RAM模块中；RAM在时钟下降沿进行读取数据，送至Butterfly中。

2）送至Butterfly中的数据与时钟下降沿保持同步。在Butterfly中，在时钟上升沿对Data A完成移位操作，同时并发事件有Data B、C完成相乘操作。

3）在下一个时钟上升沿来临时，对移位完成的Data A进行寄存一拍，同时并发事件为对Data B、C相乘得到的数据进行加减操作。

4）计算输出。

上述过程并保持了流水线结构，最大限度地节省了时钟周期。流水线时序示意图如图1.5所示。

|  |
| --- |
|  |
| 图1.5 蝶形运算流水线 |

由图1.5可知，从读取地址到输出最后结果，共需要3个Cycle。因此计算n次蝶形运算共需要消耗n+3个Cycle。

## 1.3 控制模块RAM\_Ctrl

由图1.1可知，蝶形运算的两个输入节点与蝶形算子的选择存在一定数学规律。假设N为2的整数次幂，如果要完成N点FFT，则需要完成级计算。此时FFT按照L=1,2…,的顺序完成各级运算。

在各级完成运算时，每级运算对应有N/2个蝶形运算，并且存在种不同的旋转因子。假设在L级中，采用相同旋转因子完成计算的索引为第J层。如在第2级、第1层的所有蝶形运算使用的旋转因子相同。

在第L级运算时，对应有种旋转因子。因此在L级中按照J=0,1,…,的次序完成运算。在第L级、第J层中，蝶形运算节点A的取值满足J:2^L:N。

上述过程由RAM\_Ctrl控制模块实现，并且伪代码如下所示：

|  |
| --- |
| L=1:1:L\_MAX  J=0:1:2^(L-1)-1  factor\_re\_r <= rom[(2^(L\_max-L))\*J][31:16]  factor\_im\_r <= rom[(2^(L\_max-L))\*J][15:0]  k=J:2^L:N  rd\_add1 = k;  rd\_add2 = k+2^(L-1); |

其中；rd\_add1、rd\_add2为取出数据Data A、B的地址位；factor\_re\_r、factor\_im\_r分别为旋转因子的实部和虚部。程序流图如图1.6所示。

|  |
| --- |
|  |
| 图1.6 RAM\_Ctrl程序流图 |

由图1.6可知，在每次FFT开始前，要等待RAM中数据初试化完毕，也就是对初始数据按照“码位倒读”顺序进行存储完成后，才可以开始第一级的运算；否则需要一直等待初试化结束。

并且在计算完每一级运算之后，需要等待3个Cycle作为数据写入缓冲期，以避免在RAM中同时对一个地址进行读写操作。如图1.7所示。

|  |
| --- |
|  |
| 图1.7 RAM\_Ctrl控制读写操作 |

待所有FFT操作完成之后，输出wd\_finish信号，代表本次FFT结束，此时进入等待期，等待下一次FFT操作开始。

## 1.4 N点FFT的FPGA结构及分析

根据1.2、1.3的分析，本次FPGA实现的整体框图如图1.8所示。

|  |
| --- |
|  |
| 图1.8 系统结构 |

据图1.8可知，本系统使用了一个RAM作为存储空间，相对于传统的FFT使用“乒乓结构”的两个RAM，可以节省50%的资源。并且每一级计算结束后，RAM仅需要3个Cycle作为写入缓冲期，并没有因为少使用一个RAM而大大增加时钟消耗。

ROM中存储的是旋转因子。第L级旋转因子有种，每级旋转因子有N/2个。按照传统运算，是将每级所有的旋转因子存储下来，再寻址。

本结构利用数学规律，将每级运算又分为J层，J。每层运算的旋转因子相同，因此在同一层运算时，取旋转因子的地址位不变即可。此时我们仅需存储个不同的旋转因子，相对于传统需要存储个旋转因子，可以节省较多的ROM资源。

除此之外，本结构避免了设计出来的FFT点数固定，我们可以通过修改FPGA中的参数进行FFT点数选择。修改参数如下：

|  |
| --- |
| module top  #(  parameter N ,  parameter L\_max  ) |

通过对顶层模块top.v中的参数N、L\_max进行修改，即可得到所需要的FFT点数。N为FFT点数，L\_max为完成一次N点FFT所需要的级数。L\_max=.

## 1.5 FPGA与Matlab的FFT结果对比

对FPGA实现的FFT与Matlab自带的FFT输入同样的信号，分别做512点FFT。Modelsim仿真结果如图1.9所示，对比结果如图1.10所示。

|  |
| --- |
|  |
| 图1.9 9级运算过程示意图 |
|  |
| 图1.10 FPGA与Matlab的FFT实现对比 |

从图1.10中可以发现FPGA与Matlab实现的FFT结果几乎一致，由于FFT中数据精度取决于位宽，因此会存在一定的量化误差，通过图1.10中可以发现，该量化误差可以忽略不记。