

# 基于 FPGA 的成像声纳 FFT 波束形成器设计

作者：杨威，赵极远

导师：刘鑫

(哈尔滨工程大学水声技术重点实验室, 哈尔滨, 150001)

摘要：本文针对成像声纳波束形成器的特点，设计了一种基于 FPGA 的 FFT 波束形成器。整个系统采用 Altera 公司的 DSP Builder 构建。FFT 波束形成器采用基 2-512 点 DIT-FFT 算法，并使用流水线技术、乒乓操作。在 Altera StratixII FPGA EP2S90F784I4 硬件平台上测试，30MHz 系统时钟，17.07 $\mu$ s 得到 512 点 FFT 运算结果，满足成像声纳系统波束形成器要求。

关键词：FPGA；成像声纳；FFT 波束形成；DSP Builder

## Design of the FFT Beam Former of Imaging Sonar Based on FPGA

Author: YANG Wei, ZHAO Jiyuan,

Tutor: LIU Xin

(College of Underwater Acoustic Engineering, Harbin Engineering University, Harbin 150001, China)

**Abstract:** In terms of the characteristics of the FFT beam forming of imaging sonar, the FFT beam forming is designed by FPGA. The whole system is builded by DSP Builder of Altera. The FFT beam forming uses 512 Points DIT-FFT radix 2 algorithm, pipeline technology and Ping-Pong operation. The system is tested in the hardware of Altera StratixII EP2S90F784I4. The system can get the 512 results of FFT in 17.07 $\mu$ s under the condition of system clock is 30MHz, which can meet the requirement of the beam forming of imaging sonar easily.

**Key words:** FPGA; Imaging Sonar ;FFT Beam Forming; DSP Builder

## 1 引言

海洋面积占地球表面积的 71%，海底蕴藏的石油、天然气等矿产资源量也远远超过陆地。由于能源危机和资源短缺日益严重，世界各国对海洋的开发利用愈发重视。水声成像技术不仅能够探测海底结构，而且水声成像技术相比于传统视频设备的优点是呈现的图像几乎不受水文条件的影响，无论在军用或民用领域，声成像技术都是未来船舶与海洋工程研究的主要技术之一<sup>[1]</sup>。我国拥有丰富的海洋资源，成像声纳的发展对海洋资源开发、海洋生物观测、海洋物理研究等相关科学的发展有着重要而深远的意义。

对要求实时成像的成像声纳来说，成像速度是衡量其性能优劣的非常重要的一个标准。波束形成器是成像声纳数字系统的重要组成部分，其运算速度影响到整个系统的成像速度，因此提高波束形成运算速度是提升成像声纳成像速度的关键。常见的波束形成技术有：时延、相移波束形成等，相移波束形成中的 FFT 波束形成由于具有最成熟的算法实现结构和快速

的运算速度，成为成像声纳波束形成器首选。

## 2 FFT 波束形成器原理

### 2.1 FFT 波束形成

波束形成技术是指将按一定几何形状排列的多元基阵的各阵元输出经过处理形成空间指向性的方法。波束形成的目的是使多阵元构成的基阵经过适当的处理得到在预定方向的指向性，它可以滤去空间某些方位的信号，只让特定方位的信号通过，也可以说波束形成器起到空间滤波器的作用<sup>[2]</sup>。

本文采用等间隔直线阵 FFT 波束形成。一个  $N$  元等间隔直线阵阵元间隔为  $d$ ，当接收信号为单频或窄带信号时，基阵第  $i$  号阵元的输出信号可用复数表示为：

$$x_i(t) = w_i e^{j(\omega t + i\varphi)} \quad (-\pi \leq \varphi \leq \pi), (i = 0, 1, \dots, N-1) \quad (2.1)$$

其中  $\varphi = \frac{2\pi d}{\lambda} \sin \theta$  为相邻阵元接收信号间的相位差， $\theta$  为信号入射方向与基阵法线之间的夹角， $w_i$  为基阵各阵元的幅度加权系数， $\omega$  为信号角频率， $\lambda$  为信号波长。忽略时间因子，记阵元输出为：

$$x_i = w_i e^{ji\varphi} \quad (2.2)$$

如果在相邻阵元间插入相移  $\beta_r$ ，则基阵第  $i$  号阵元的输出信号变为：

$$x'_i(t) = w_i e^{j\omega t} e^{j(i\varphi - i\beta_r)} \quad (2.3)$$

阵输出为：

$$Br(t) = \sum_{i=0}^{N-1} w_i e^{j\omega t} e^{j(i\varphi - i\beta_r)} = \sum_{i=0}^{N-1} x_i(t) e^{-ji\beta_r} \quad (2.4)$$

忽略时间因子，阵输出为：

$$Br = \sum_{i=0}^{N-1} x_i e^{-ji\beta_r} \quad (2.5)$$

如果取  $\beta_r = \frac{2\pi}{N} r$ ，(2.5) 式就可以化为：

$$Br = \sum_{i=0}^{N-1} x_i e^{-j\frac{2\pi}{N} i r} \quad (2.6)$$

式(2.6)实际上是离散傅立叶变换的形式,因此计算一个等间隔直线阵各波束输出值就等价于计算各阵元的输出信号  $x_i$  的离散傅立叶变换。 $x_i$  的物理意义表示基阵的不同阵元在同一时刻对信号在空间上的等间隔采样。对  $x_i$  作 FFT 就是为了提取信号的离散空间谱线,这个空间谱线就表示信号在空间上的能量分布,这和波束形成要做的工作恰恰是一致的。所以可以利用这一特点对基阵输出信号作快速波束形成。

## 2.2 FFT 基本原理

FFT 算法的基本原理是把长序列的 DFT 逐次分解为较短序列的 DFT。按照抽取方式的不同可分为 DIT-FFT(Decimation In Time, 按时间抽取)和 DIF-FFT(Decimation In Frequency, 按频率抽取)算法。按照蝶形运算的构成不同可分为基 2、基 4、基 8 以及任意因子 ( $2n, n$  为大于 1 的整数),其中基 2、基 4 算法较为常用。本文使用基 2 DIT-FFT 算法<sup>[3]</sup>。

设序列  $X(n)$  的长度为  $N$ ;且满足式 (2.7)

$$N = 2^M, \quad M \text{ 为自然数} \quad (2.7)$$

按  $n$  的奇偶把  $X(n)$  分解为两个  $N/2$  点的子序列, 如式 (2.8)

$$\begin{aligned} x_1(r) &= x(2r), \quad r = 0, 1, \dots, \frac{N}{2} \\ x_2(r) &= x(2r + 1), \quad r = 0, 1, \dots, \frac{N}{2} \end{aligned} \quad (2.8)$$

则  $X(n)$  的 DFT 为式 (2.9)

$$\begin{aligned} X(k) &= \sum_{n=\text{偶数}} x(n)W_N^{kn} + \sum_{n=\text{奇数}} x(n)W_N^{kn} \\ &= \sum_{r=0}^{N/2-1} x(2r)W_N^{2kr} + \sum_{r=0}^{N/2-1} x(2r+1)W_N^{(2r+1)k} \\ &= \sum_{r=0}^{N/2-1} x_1(r)W_N^{2kr} + W_N^k \sum_{r=0}^{N/2-1} x_2(r)W_N^{2kr} \end{aligned} \quad (2.9)$$

由于

$$W_N^{2kr} = e^{-j\frac{2\pi}{N}2kr} = e^{-j\frac{2\pi}{N}kr} = W_{N/2}^{kr} \quad (2.10)$$

所以

$$\begin{aligned}
&= \sum_{r=0}^{N/2-1} x_1(r)W_{N/2}^{kr} + W_N^k \sum_{r=0}^{N/2-1} x_2(r)W_{N/2}^{kr} \\
&= X_1(k) + W_N^k X_2(k) \quad k = 0, 1, \dots, N-1
\end{aligned} \tag{2.11}$$

其中  $X_1(k)$  和  $X_2(k)$  分别为  $x_1(r)$  和  $x_2(r)$  的  $N/2$  点 DFT，即式 (2.12)

$$\begin{aligned}
X_1(k) &= \sum_{r=0}^{N/2-1} x_1(r)W_{N/2}^{kr} = \text{DFT}[x_1(r)] \\
X_2(k) &= \sum_{r=0}^{N/2-1} x_2(r)W_{N/2}^{kr} = \text{DFT}[x_2(r)]
\end{aligned} \tag{2.12}$$

这样，就将  $N$  点 DFT 分解为两个  $N/2$  点的 DFT 的运算。并且由于  $N/2$  仍然是偶数，故可以对  $N/2$  点 DFT 再作进一步分解，推导思路类似，不再赘述，详细推倒过程可参看参考文献[3]。

### 3 FFT 波束形成器的 DSP Builder 实现与验证

#### 3.1 DSP Builder 设计流程

当使用 DSP Builder 建立一个设计时，首先使用 MATLAB Simulink 软件设计模型，在生成了模型后，能输出用于 Quartus II 编译的 VHDL 文件或生成 VHDL、VerilogHDL 仿真的文件。FFT 波束器设计流程包括下列步骤<sup>[4]</sup>：

1. 使用 MATLAB/Simulink 软件生成由 Simulink 与 DSP Builder 组合的 FFT 波束形成器模型。
2. 使用 SignalCompiler 模块分析设计。
3. 在 Simulink 中，使用监视结果的 Scope 模块仿真模型，观察运算结果。
4. 运行 SignalCompiler 设置 RTL 仿真和综合设计。
5. 执行 RTL 仿真。利用 DSP Builder 生成的 Tcl 脚本，使用 ModelSim 软件仿真。
6. 使用由 DSP Builder SignalCompiler 模块生成的输出文件来执行 RTL 综合。本设计采用 Quartus II 9.0 执行综合工作。
7. 在 Quartus II 软件中编译 FFT 波束形成器模型。
8. 下载到成像声纳数字板 StratixII EP2S90F78014 上并测试。

图 3.1 展示了使用 DSP Builder 设计的系统级设计流程。

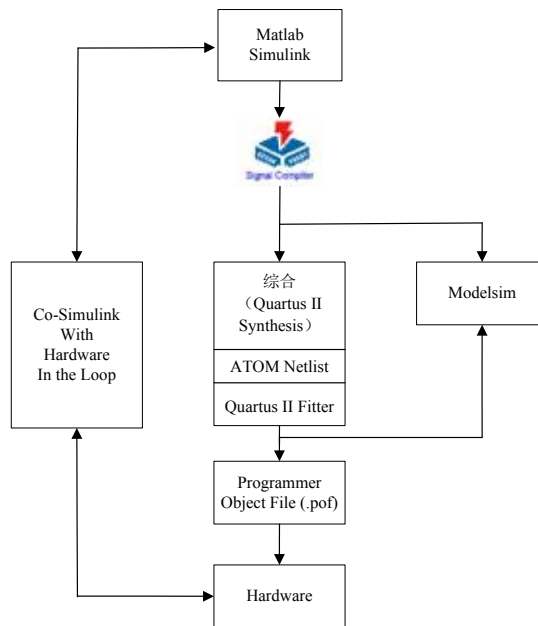


图 3.1 DSP Builder 系统级设计流程

### 3.2 FFT 波束形成器的 DSP Builder 实现

#### 3.2.1 总体设计框架

为了让成像声纳达到较高的分辨率，需要对更多的数据进行实时性处理，因此本设计要求在  $20\mu\text{s}$  内得到 512 点 FFT 运算结果，并且运算结果误差在 1% 以内。考虑参数要求，FFT 波束形成器设计包括如下三部分：数据预处理部分（加权、聚焦）；512 点基 2 DIT-FFT 算法模块部分；数据整理部分（ABS 计算）。总体框架如图 3.2 所示。

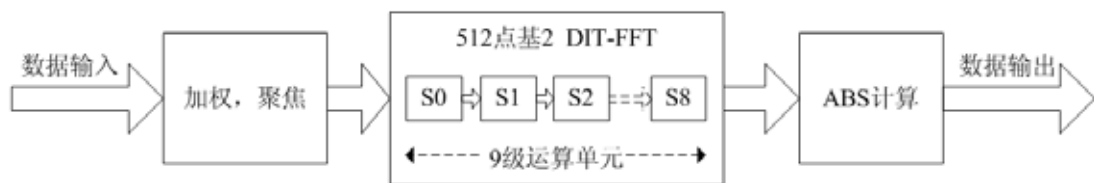


图 3.2 FFT 波束形成器总体框架图

#### 1、流水线技术

为了提高数据处理能力，采用流水线设计方法提高系统的工作频率。FFT 模块 9 级运算单元 (State0~State8) 并行运行，这样 9 级数据运算时间仅为 1 级的运算时间 (512 个时钟周期)。图 3.3 显示了流水线设计时序示意图，某一时刻 T1 数据 A 送入 State0，下一时刻 T2 数据 A 送入 State1，B 送入 State0...这样 B 数据不需要等待 A 数据经过 9 级运算得出结果后，再进行运算。使用流水线技术后的计算时间相当于未使用流水线的  $1/9$ ，即在相同的时钟频率下，数据处理能力提高了 9 倍。图 3.4 给出了 9 级处理单元 (State0~State8) 的 DSP Builder 实现结构图。

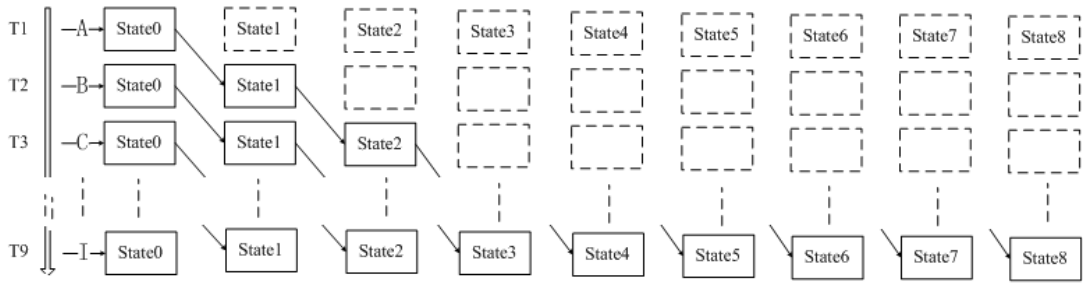


图 3.3 流水线设计时序示意图

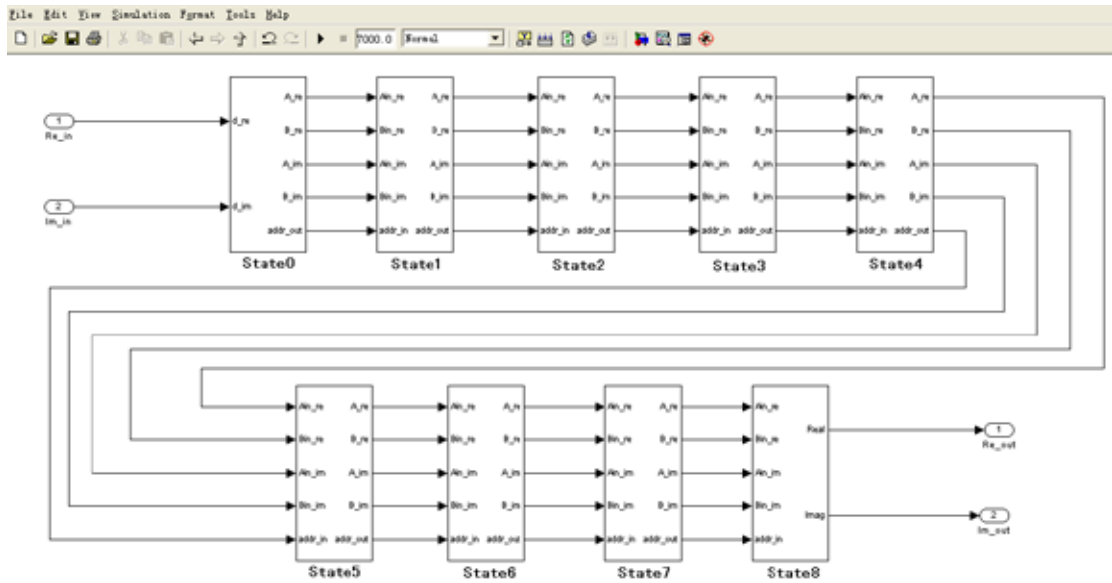


图 3.4 9 级处理单元 DSP Builder 实现结构图

## 2、乒乓操作

为了不间断处理数据，本设计采用乒乓操作控制数据流。FFT 模块每级运算单元之间都有两块 DPRAM（双口 RAM，1024 点、16 位）来缓冲数据，一块 RAM 缓冲实部运算结果，另一个 RAM 缓冲虚部运算结果。图 3.5 显示了一块双口 RAM 乒乓操作的示意图。RAM 划分为两部分 PART A (addr:0~511) 和 PART B (addr:512~1023)，某一时刻 T1，向 A 中写入数据，从 B 中读取数据；下一时刻 T2，向 B 中写入数据，从 A 中读取数据，按照上述次序循环写入读取数据。这样，在完成一次 512 点数据 FFT 运算后，不需要等待即可开始下一次 512 点的运算，实现不间断处理数据。

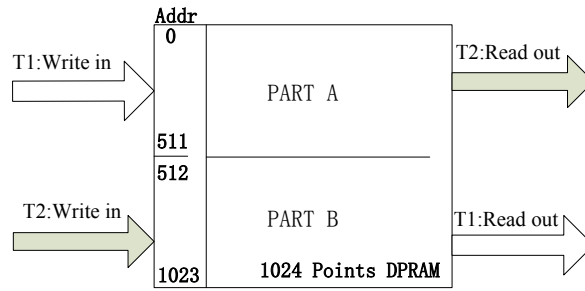


图 3.5 乒乓操作示意图

### 3.2.2 FFT 组成单元设计

FFT 每级运算包括三单元：地址产生单元(addr\_sx)；数据读取单元(rd\_data)；基 2 蝶形运算单元(butterfly\_radix2)。图 3.6 表示第 0 级 (State0) 运算顶层结构图，其它级运算结构与此级类似。

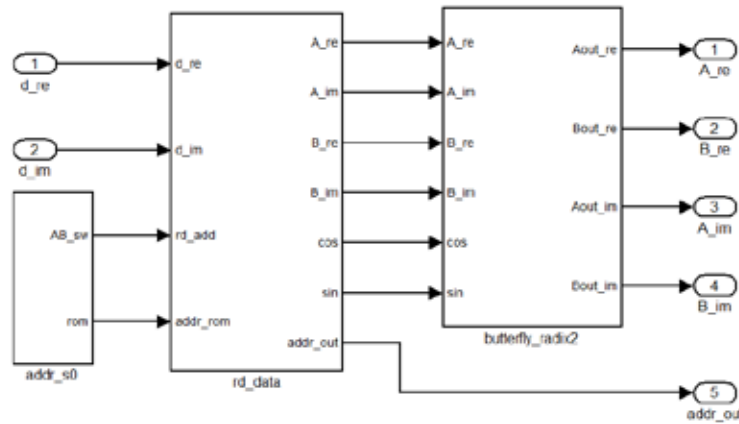


图 3.6 第 0 级 (State0) 顶层结构图

下面简要介绍每级的三个组成单元：旋转因子产生单元，地址产生单元，蝶形运算单元。

#### 1、旋转因子产生

本设计由 Matlab 产生需要的旋转因子数据。Matlab 代码如图 3.7。

```
function [w]=fftw
clear all;
clc;
n=512; %fft点数
for i=1:n/2
    w(2*i-1)=(cos(-2*pi*(i-1)/n)*(2^15-1));%产生正变换的选择因子 *32767
    w(2*i)=(sin(-2*pi*(i-1)/n)*(2^15-1));
end

fid = fopen('twiddle_factors512.txt','wt');
fprintf(fid,'%10.0f,%10.0f,\n',w);
fclose(fid);
```

图 3.7 Matlab 产生旋转因子代码

由于 RAM 中只能存储整数，所以对得到数据乘 32767，将所有小数位移动至整数位，再与数据相乘。将得到的 512 点数据分别存入 Quartus II 新建的两个 16 位 256 点 input\_sin.hex 和 input\_cos.hex 文件中，并在 ROM 模块中设置从 input\_sin.hex、input\_cos.hex 初始化。

## 2、地址产生

地址产生分两部分，包括运算数据 (A, B) 与旋转因子地址产生。

### (1)运算数据 (A, B) 地址产生

对于点数  $N = 512$  的 FFT,  $m$  表示运算级数,  $m \in [0, 1 \dots 7, 8]$ , 分析其地址产生规律, 发现在第  $m$  级运算数据 A、B 的地址可以由如下操作得到: 9 位 Bus Splitter 的第 0 位与 Bus Builder 的第  $m$  位相连, Bus Splitter 的第  $[m:1]$  位依次与 Bus Builder 的第  $[m-1:0]$  相连。此操作产生新的数据即为数据 A、B 的地址。DSP Builder 中实现第 1 级运算数据地址产生, 如图 3.8 所示。

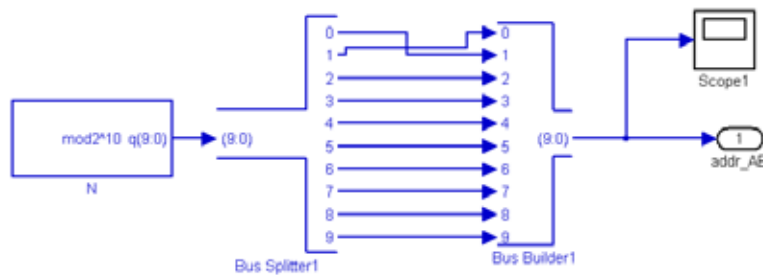


图 3.8 DSP Builder 中实现第 1 级运算数据地址产生

在示波器 Scope1 中观察运算数据地址产生如图 3.9 所示。

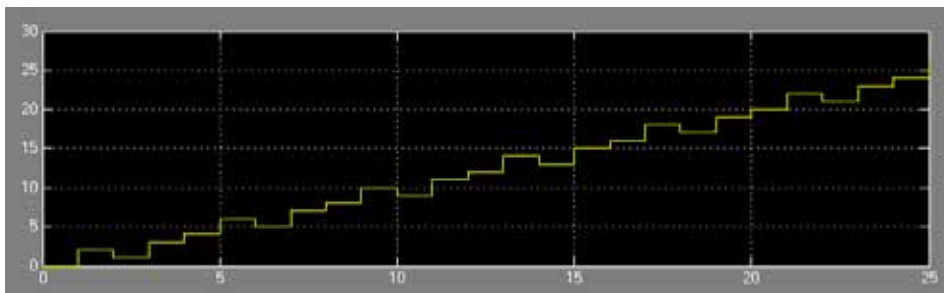


图 3.9 示波器观察第 1 级运算数据地址产生

256 组数据 A、B 地址值分别是: (0,2;1,3;4,6...508,509;510,511)。与理论值相符。

### (2)旋转因子地址产生

$N$  点 FFT 共需要旋转因子  $N/2$  个,  $r=(0 \dots N/2-1)$ 。设  $addr\_w$  为当前蝶型运算所对应的旋转因子查找表入口地址。需要移位和求余两种简单的运算。

比如: 第 1 级  $addr\_w = (1 * 2^{1 \circ 9_2 \cdot 512-1}) \% 512 / 2 = (0, 127, 0, 127 \dots 0, 127, 0, 127)$

逻辑实现:第  $m$  级, 第  $n$  个蝶型运算的旋转因子地址可由逻辑移位的方法快速得到。将  $n(8\text{bits})$ 左移位 $(8-m)$ , 低位补 0, 就得到了第  $m$  级的  $\text{addr}_w$ 。DSP Builder 实现第 1 级旋转因子地址产生, 如图 3.10 所示。示波器 Scope1 观察结果如图 3.11 所示。示波器的结果与理论值相一致。

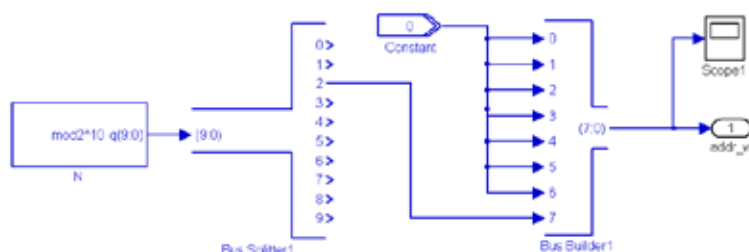


图 3.10 DSP Builder 中实现第 1 级旋转因子地址产生

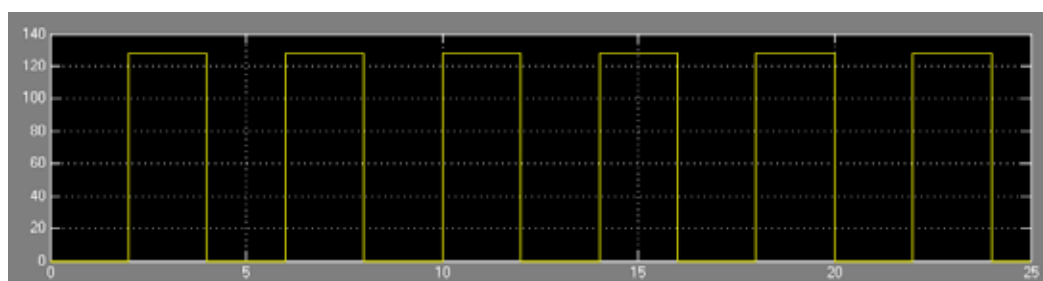


图 3.11 示波器中观察第 1 级旋转因子地址产生

256 组  $\text{addr}_w$  的值分别是:  $(0,128; 0,128 \dots 0,128; 0,128)$ 。与理论相符。

### 3、蝶形运算单元

蝶形运算器完成数据  $A$ 、 $B$  的蝶形运算, 设输入数据为  $A = a_r + aj$ ,  $B = b_r + bj$ ,

旋转因子  $W_N^k = \cos w + \text{si n } wj$ 。则蝶形运算输出为<sup>[5]</sup>

$$A_{out} = A + W_N^k B = a_r + b_r \cos w - b_i \text{si n } w + [a_i + b_r \text{si n } w + b_i \cos w]j$$

$$B_{out} = A - W_N^k B = a_p - b_p \cos w + b_i \text{si n } w + [a_i - b_r \text{si n } w - b_i \cos w]j$$

从上两式子可以看出共有 2 个乘累加项 ( $b_r \cos w - b_i \text{si n } w$ ) 和 ( $b_r \text{si n } w + b_i \cos w$ )。输入数据为 16 位, 乘累加单元运算结果需截断低 15 位 (蝶形运算因子低 15 位为小数), 保留整数部分再输入到加法器、减法器。同时为了使数据对齐, 数据通过寄存器组需延时 4 个时钟周期。

本设计采用定点逐位扩展算法。浮点算法有较大的动态范围和较高的精度, 但使用浮点运算不仅会消耗更多的资源还会使蝶形运算单元的运算速度降低。从设计应用的角度分析,

本设计对每次蝶形运算扩展一位。

下面简要介绍 DSP Builder 实现第 1 级蝶形运算：数据 B 的实部、虚部先与蝶形因子进行乘加运算，并对得到的运算结果截断低 15 位（如图 3.12-I 所示）；再与延时了 4 个时钟周期的数据 A 的实部、虚部进行加减运算，最后扩展数据位宽 1 位，得到蝶形运算的输出（如图 3.12-II 所示）。

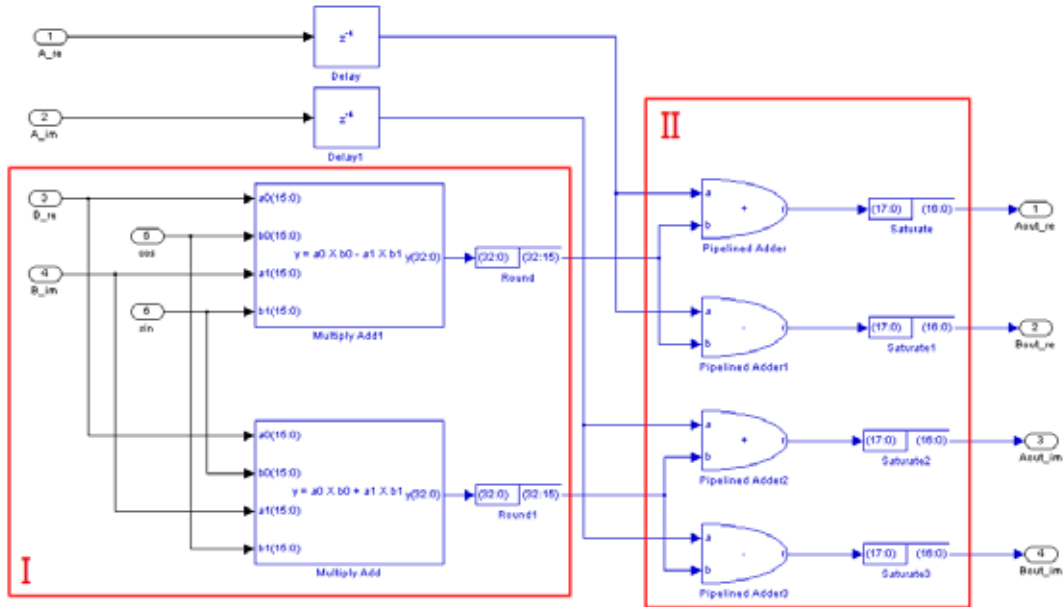


图 3.12 第 1 级蝶形运算结构图

### 3.3 FFT 波束形成器的 DSP Builder 验证

首先，使用 Matlab 中 fft()函数分析在给定激励（实部、虚部均输入[0:1:511]）时 512 点 FFT 运算的理论输出，输入如图 3.13 所示，Matlab 中运行得到的运算结果如图 3.14 所示。

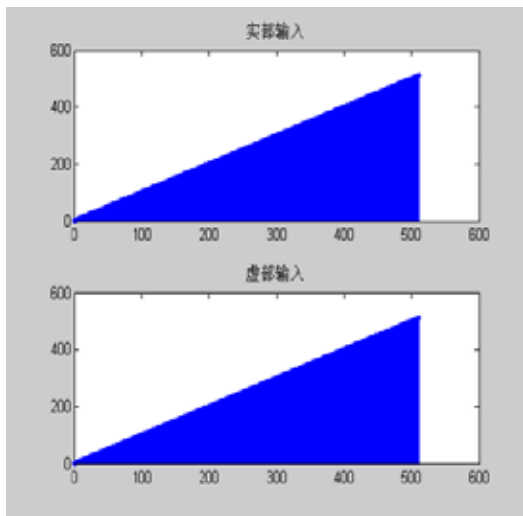


图 3.13 输入实部、虚部

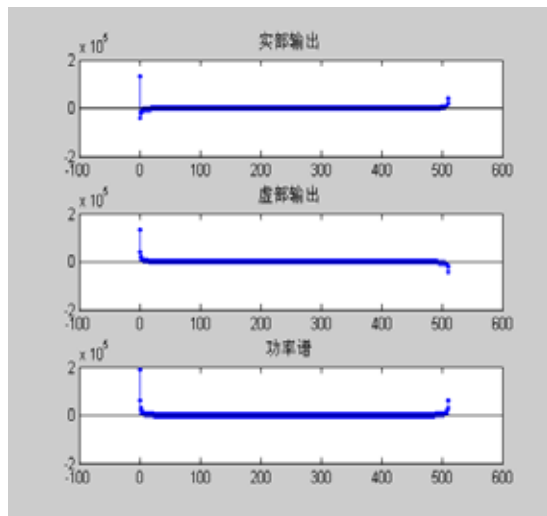


图 3.14 输出实部、虚部、功率谱

### 3.3.1 软件验证

本小节分析 DSP Builder 设计的 FFT 运算模型与理论之间的误差,使用 Simulink 中函数 Repeating Sequence Stair 作为设计激励输入,实部、虚部循环输入数据[0:1:511]。激励如图 3.15 所示。运行设计,得到软件仿真 512 点 FFT 功率谱,如图 3.16 所示。

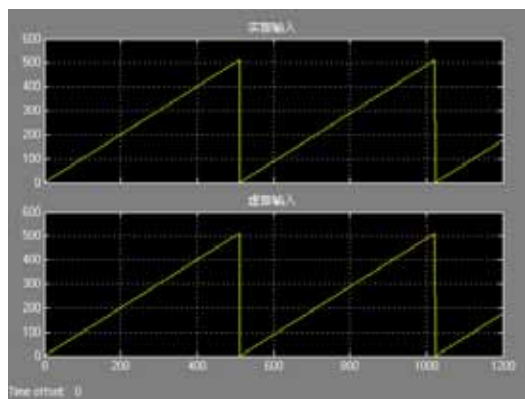


图 3.15 激励输入

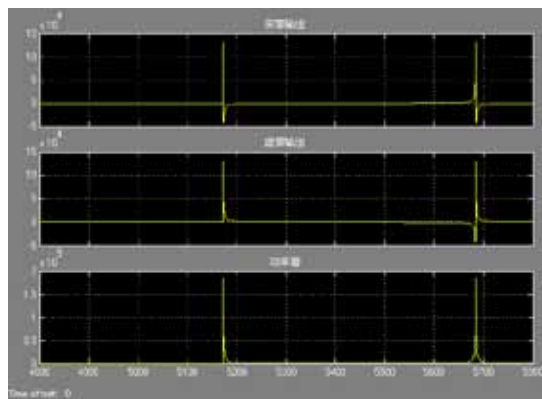


图 3.16 软件仿真 512 点 FFT 运行结果

DSP Builder 运行结果与 Matlab 得到理论功率谱值求误差,并作图,得到如图 3.17 所示。

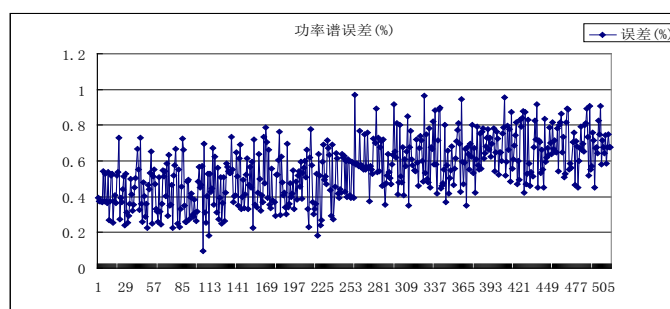


图 3.17 功率谱误差曲线

分析发现 512 点运算结果精度保持在 1%以内,满足设计要求。同样的激励,在 Modelsim 中仿真运算模型,如 3.18 所示上两个信号分别是实部、虚部,下两个信号分别是运算输出实部、虚部,与 Matlab 理论运算结果误差在 1%以内。

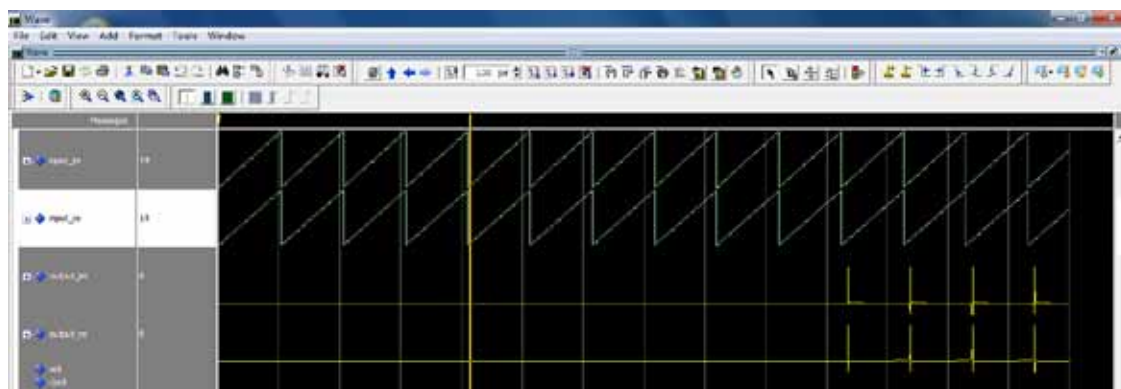


图 3.18 Modelsim 仿真结果

### 3.3.2 硬件验证

使用 DSP Builder HIL(Hardware In Loop)模块将设计包裹在一套接口中间，编译然后下载到板子的 FPGA 中。Simulink 通过下载电缆把测试数据不断的灌入，然后在输出端不断的获得硬件运行结果<sup>[6]</sup>。HIL 设计视图如图 3.19 所示。

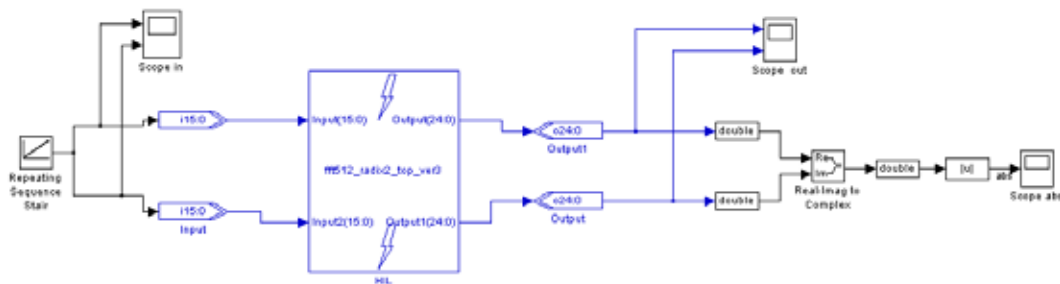


图 3.19 512 点 FFT HIL 视图

将设计下载到 Stratix II FPGA 芯片 EP2S90F780I4 进行测试，下载并运行工程，得到结果与图 3.16 一致，即与软件仿真结果相同，硬件仿真正确。下载调试实物图如图 3.20、3.21 所示。



图 3.20 下载调试实物图 1

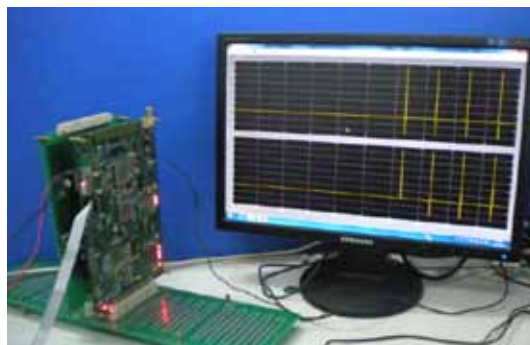


图 3.21 下载调试实物图 2

## 4 FFT 波束形成器资源消耗与性能分析

### 4.1 资源消耗分析

为满足成像声纳实时性处理大量数据的要求，本设计采用 Altera 公司高性能 Stratix II FPGA EP2S90F780I4 作为硬件实现平台。

本设计中采用流水线技术与乒乓操作作为提高 FFT 波束形成器速度的主要途径。根据运算特点采用了 9 级流水线，使得整个序列的计算时间仅为 1 级蝶形运算所需的时间；乒乓操作的使用则可以使 FPGA 不间断处理数据。使用 Altera 综合与布线工具 Quartus II 9.0 编

译设计，得到编译报告如图 4.1 所示。

```

Flow Status                Successful - Mon Apr 18 18:15:34 2011
Quartus II Version        9.0 Build 132 02/25/2009 SJ Full Version
Revision Name             fft512_radix2_top_ver3_HLL
Top-level Entity Name     fft512_radix2_top_ver3_HLL
Family                    Stratix II
Device                    EP2S90F780I4
Timing Models             Final
Met timing requirements   No
Logic utilization         12 %
  Combinational ALUTs     3,222 / 72,768 ( 4 % )
  Dedicated logic registers 8,232 / 72,768 ( 11 % )
Total registers           8232
Total pins                0 / 535 ( 0 % )
Total virtual pins        0
Total block memory bits   762,019 / 4,520,448 ( 17 % )
DSP block 9-bit elements  216 / 384 ( 56 % )
Total PLLs                0 / 6 ( 0 % )
Total DLLs                0 / 2 ( 0 % )
    
```

图 4.1 Quartus II 编译结果

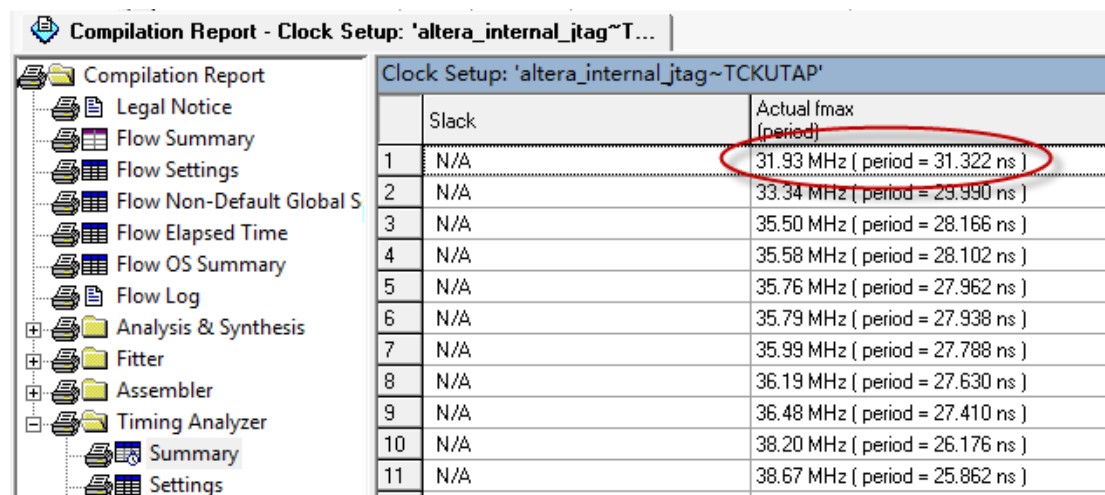
由图 4.1 可知，逻辑资源使用 12%，存储资源使用 17%，DSP 资源使用 56%。本设计中一共使用了 20 块 1024 点 RAM（位数分别从 16bit 到 25bit），以及 18 块 256 点 ROM（16 位），所以消耗了 762019 bits 存储资源。另外本设计流水线结构采用了 9 级运算单元，而且由于每级蝶形运算位数扩展，势必要消耗大量的 DSP 资源，FFT 运算模型共消耗了 216 个 9×9 的乘法器，主要原因是采用了定点算法，每级蝶形运算单元的每一级都有 1bit 的扩展，导致乘法器输入宽度增加，而 FPGA 中的 DSP 块的乘法器部分是由 4 个 18×18 的乘法器构成的，或组成 1 个 36×36 的乘法器。当数据宽度大于 19 位后，就会调用一个 36×36 的乘法器，乘法器资源被很大的浪费了。表 4.1 展示的是 DSP 资源详细消耗情况。

表 4.1 DSP 资源消耗情况

级数\项目	乘因子位数	DSP18x18	DSP36x36	DSP Elements
0	16	4	0	8
1	17	4	0	8
2	18	4	0	8
3	19	0	4	32
4	20	0	4	32
5	21	0	4	32
6	22	0	4	32
7	23	0	4	32
8	24	0	4	32
总数		12	24	216

## 4.2 性能分析

在编译报告时序分析结果中，查看关键路径的  $f_{\max}$ ，如图 4.2 所示。从图中可知，最低  $f_{\max}$  为 31.93MHz，也就是系统最高运行速度可达到 31.93MHz，本设计采用 30MHz 系统时钟，因此系统可在 17.07 $\mu$ s 内得到 512 点 FFT 运算结果，满足本设计成像系统波束形成 20 $\mu$ s 以内要求。



	Slack	Actual fmax (period)
1	N/A	31.93 MHz ( period = 31.322 ns )
2	N/A	33.34 MHz ( period = 29.990 ns )
3	N/A	35.50 MHz ( period = 28.166 ns )
4	N/A	35.58 MHz ( period = 28.102 ns )
5	N/A	35.76 MHz ( period = 27.962 ns )
6	N/A	35.79 MHz ( period = 27.938 ns )
7	N/A	35.99 MHz ( period = 27.788 ns )
8	N/A	36.19 MHz ( period = 27.630 ns )
9	N/A	36.48 MHz ( period = 27.410 ns )
10	N/A	38.20 MHz ( period = 26.176 ns )
11	N/A	38.67 MHz ( period = 25.862 ns )

图 4.2 运算模型时序分析报告

## 5 结论

本论文以成像声纳数字系统的设计为背景，采用 DSP Builder 完成了 FFT 波束形成器设计与验证。在 Stratix II EP2S90F780I4 FPGA 上测试设计，30MHz 系统时钟，17.07  $\mu$ s 内得到 512 点 FFT 运算结果、误差在 1% 以下，满足设计要求。

与传统的 FPGA 设计方法(Matlab 编写代码设计算法,HDL 实现)相比,使用 DSP Builder 开发成像声纳 FFT 波束形成器极大的提高了设计的效率,节省了宝贵的开发时间,并且当算法变动时,修改参数也更为方便、快捷。同时自主设计的 FFT 运算模型代替了使用 FFT IP 核,节省了开发成本。下一步可使用 DSP Builder 完成成像声纳数字系统其他部分,实现数字系统全部的功能。

## 参考文献

- [1] 杨长根. 基于 FPGA 的成像算法研究与实现[D]. 哈尔滨工程大学硕士学位论文, 2009 年 3 月: 1-5.

- [2] 田坦, 刘国枝, 孙大军. 声呐技术[M]. 哈尔滨: 哈尔滨工程大学出版社, 2000.3: 1-25.
- [3] 姜进东. 高速 FFT 的 FPGA 实现[D].西安电子科技大学硕士学位论文, 2009.7: 8-11.
- [4] DSP Design Flow User Guide[Z]. <http://www.altera.com>, 2008: 5-15.
- [5] 田丰等. FFT 算法的一种 FPGA 实现[D].现代电子技术, 2005. 8 .
- [6] DSP Builder Reference Manual[Z]. <http://www.altera.com>, 2008: 5-10.

## 原创性声明

郑重声明: 本论文《基于 FPGA 的成像声纳 FFT 波束形成器设计》是作者在导师的指导下进行的研究工作及取得的研究成果。论文中除注明引用的内容外, 不包含任何他人已经发表或撰写过的研究成果。

作者签名: 杨威, 赵极远

导师签名: 刘鑫

撰写日期: 二零一一年八月十二日