第六届

全国大学生集成电路创新创业大赛

报告类型*：技术文档

参赛杯赛*：航天微电子杯

作品名称*：基于国产亿门级 FPGA 的软件无线电射频信号校准

目录

一、 项目概述

当前无线电通信领域，由于传输环境等因素影响，不同组的射频信号的传输会造成一定的相位延时，通过软硬件技术对高速射频信号进行相位校正有着诸多领域的应用潜力。

本设计是基于北京微电子技术研究所研制的 BQR7VX690T FPGA 开发平台进行实现， **此设计实现功能如下：**

1. **利用 GTH 进行 2Gbps 速率收发两路射频信号；**
2. **使用 DDR3 对高速信号存储并读写；**
3. **校正后两路信号相位差在 1° 以内输出；**
4. **使用 RS232 串口进行上位机监测，可视化校正前两路信号相位差值，及校正后是否对齐；**

开发板架构图及实物图如下：

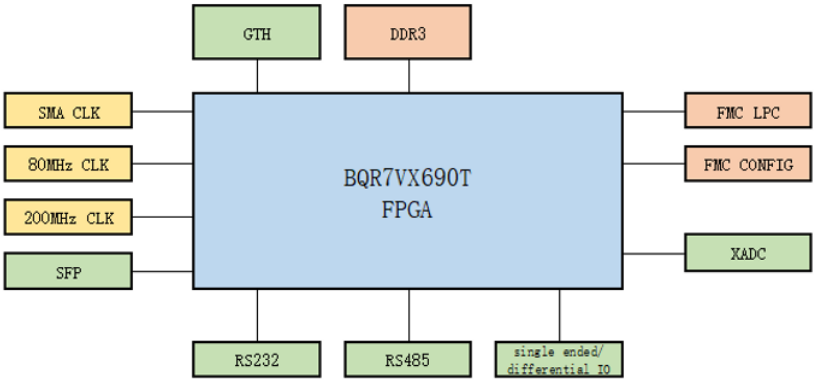


图 1-1　BQR7VX690T FPGA 开发板架构图

图 1-2　BQR7VX690T FPGA 开发板实物图

二、 相位校准算法

2.1 基于快速傅里叶变换(FFT)的相位校准算法原理

所接收的两路信号可表示为： $S1=\cos(2\pi f\bullet t+\varphi_1)$， $S2=\cos(2\pi f\bullet t+\varphi_2)$；那么离散余弦信号 $S1(n)$，$S2(n)$ 通过快速傅里叶变换技术(FFT)处理可分别得到其对应频谱 $S1(k)$， $S2(k)$。快速傅里叶变换(FFT)是基于离散傅里叶变换(DFT)的高效算法，信号经 DFT 处理如下：

$$\begin{aligned} S(k) &= DFT[S(n))] \\ &= \sum_{n=0}^{N-1} S(n)e^{\left(-j\frac{2\pi}{N}n\right)} \\ &= \sum_{n=0}^{N-1} S(n)[\cos(\frac{2\pi k}{N}n) - j\sin(\frac{2\pi k}{N}n)] \\ &= \mathrm{Re}[S(k)] + \mathrm{Im}[S(k)] \end{aligned} \tag{1}$$

其中：$k=0,1,2,3.....N-1$。余弦信号频率谱对应的实部虚部分别包含其相位的余弦量与正弦量。对于该信号的频率谱 $S(k)$，其幅度最大的点 k_0 对应频率谱线即为该余弦信号的频率谱线，通过获取该点的实部值 $\mathrm{Re}[S(k_0)]$ 与虚部值 $\mathrm{Im}[S(k_0)]$ 可计算出该信号的初相位：

$$\varphi = \arctan\left(\frac{\mathrm{Im}[S(k_0)]}{\mathrm{Re}[S(k_0)]}\right) \tag{2}$$

两组信号相位差可表示为：

$$\begin{aligned} \varphi_m &= \varphi_2 - \varphi_1 \\ &= \arctan\left(\frac{\mathrm{Im}[S_2(k_0)]}{\mathrm{Re}[S_2(k_0)]}\right) - \arctan\left(\frac{\mathrm{Im}[S_1(k_0)]}{\mathrm{Re}[S_1(k_0)]}\right) \end{aligned} \tag{3}$$

得到相位差 φ_m，结合我们所取余弦信号每点之间的相位分辨率 φ_0 为：

$$\varphi_0 = 2\pi/N \tag{4}$$

其中2π为余弦信号一个周期弧度，N为离散余弦信号采样一个周期点数。通过两信号之间的相位差φ_m与相位分辨率φ_0可得到其对应相差的相位点数N_m：

$$N_m = \frac{\varphi_m}{\varphi_0} \tag{5}$$

通过对信号$S2$进行N_m点的时延，即可得到与$S1$同相信号。整个算法流程如下：

① 通过 FFT 分别求出 S1、S2 的频谱，即实部虚部值；
② 由实部虚部值平方相加分别求出 S1、S2 各频谱点的模值；
③ 通过比较模值找出最大模值，即对应该信号的频率点；
④ 获取该频率点的实部虚部值，通过反正切运算求出该信号的初相位，相减得到两信号相位差；
⑤ 通过相位差与相位点分辨率度数相除得到两信号相位点差值；
⑥ 通过相位点差值进行相位补偿以达到两信号相位校正输出；
整个算法流程示意图如图 2-1 所示：

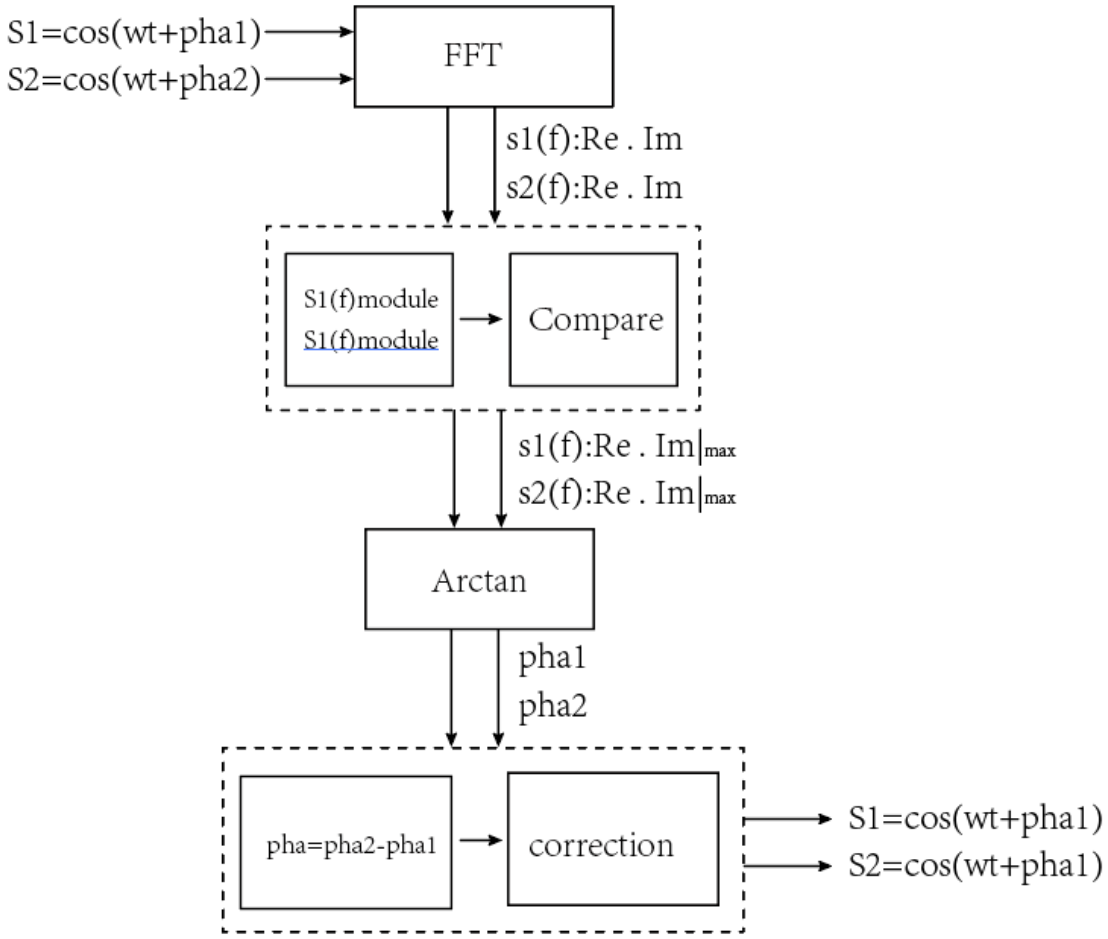


图 2-1：相位矫正算法流程示意图

2.2 MATLAB 仿真实验

2.2.1 仿真代码截图

```
function y=S1(x,F)
pha1 = (2*pi)*10/512;
y=cos(2*pi*F*x+pha1);
end
```

```
function y=S2(x,F)
pha2 = (2*pi)*50/512 ;
y=cos(2*pi*F*x+pha2);
end
```

```
clf;
% 余弦信号及FFT参数设计
fs=512;
N=512;
n=0:N-1;
num = 0:2*N-1;
t=n/fs;
t_1 = num/fs;
f = n*fs/N;
F = 1;

% 分别对两组信号做快速傅里叶变换
S1_fft=fft(S1(t,F),N);
S1_fft_re = real(S1_fft);
S1_fft_im = imag(S1_fft);
S2_fft=fft(S2(t,F),N);
S2_fft_re = real(S2_fft);
S2_fft_im = imag(S2_fft);

% 计算模值，并比较出最大值，求出两信号初相位,并得到相位差
S1_fft_module = abs(S1_fft) ;
S2_fft_module = abs(S2_fft) ;
S1_fft_module_max = max(S1_fft_module(1:256));
S2_fft_module_max = max(S2_fft_module(1:256));

for k =1:256
    if(S1_fft_module_max==S2_fft_module(k))
        Location_s1 = k ;
    end
end
for z =1:256
    if(S2_fft_module_max==S2_fft_module(z))
        Location_s2 = z ;
    end
end

phase1  = atan2(S1_fft_im(Location_s1),S1_fft_re(Location_s1));
phase2  = atan2(S2_fft_im(Location_s2),S2_fft_re(Location_s2));
phase2_1 = phase2-phase1 ;

```

```
% 通过相位差求相差相位点数，并进行相位补偿
N0 = 2*pi/N ;                %相位点分辨率；
N_phase = phase2_1/N0;       % 相差的点数 ；

if(N_phase>=0)
    N_phase_S2 = N_phase;
elseif(N_phase<0)
    N_phase_S2 = 512 + N_phase ;
%     N_phase = abs(N_phase);
%     phase2_1 = abs(phase2_1);
end

n_phase=0-N_phase_S2:2*N-1-N_phase_S2;
t1=n_phase/fs;

subplot(1,2,1)
plot(num,S1(t_1,F),'r-',num,S2(t_1,F),'b-');
legend('S1','S2');
xlabel('时间t','fontsize',12);
ylabel('幅值','fontsize',12);
title('校正前','fontsize',12)
grid on;

subplot(1,2,2)
S3=S2(t1,F); %+0.1;   % 补偿 S2，使其与S1同相
plot(num,S1(t_1,F),'r-',num,S3,'b-');
legend('S1','S2');
xlabel('时间t','fontsize',12);
ylabel('幅值','fontsize',12);
title('校正后','fontsize',12)
grid on;
```

2.2.2 仿真实验结果

① S1 初相点为 10，S2 初相点为 50 （一个周期对应 512 个点）

MATLAB 仿真结果如下：

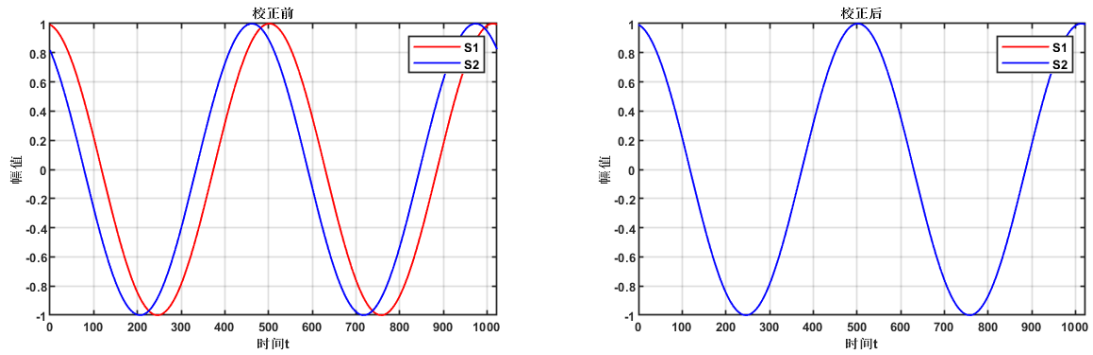


图 2-1：MATLAB 仿真实验结果 1

图 2-2 为校正前后的图像，校正后 S1、S2 输出的图像完全重合，从实验结果图可知矫正后 S1、S2 相位相同，相位矫正成功。

仿真求得其相位差点数及相位差度数（度°）如下：

```
>> N_phase

N_phase =

    40.0000
```

```
>> phase2_1 * 180/pi

ans =

    28.1250
```

② S1 初相点为 20，S2 初相点为 21 （一个周期对应 512 个点）

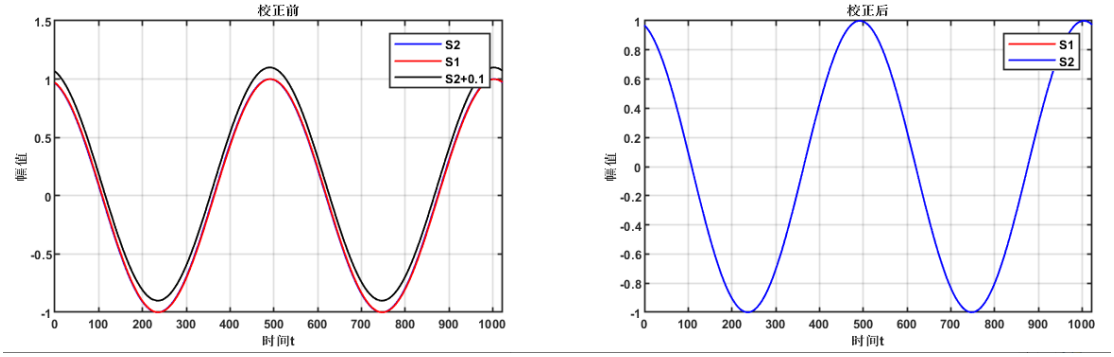


图 2-2：MATLAB 仿真实验结果 2

由于相位点数相差为 1 无法良好的观察校正前 S1 与 S2 的波形，故添加第三条 S2+0.1 幅值的波形便于更好地观察。

如上所示，可正确校正相位点数相差为 1，即 $360^{\circ}/512=0.703125^{\circ}$，仿真实验满足相位校正能够实现 1°以内。

仿真求得其相位差点数及相位差度数（度°）如下：

```
>> N_phase

N_phase =

    1.0000
```

```
>> phase2_1 * 180/pi

ans =

    0.7031
```

③ S1 初相点为 30，S2 初相点为 286 （一个周期 512 个点，S2 与 S1 相位差 180°）

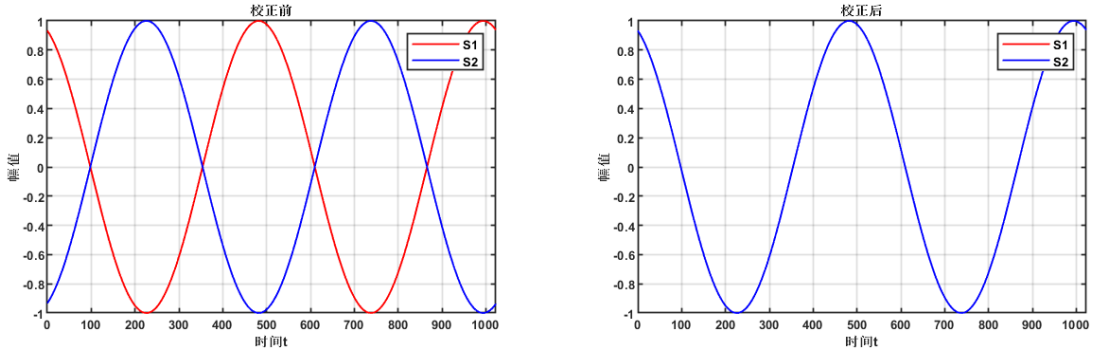


图 2-3：MATLAB 仿真实验结果 3

如上所示，可正确校正相位点数相差 256，即 180°，仿真实验满足相位校正能够实现 180° 差值的校正能力。

仿真求得其相位差点数及相位差度数（度°）如下：

```
>> N_phase

N_phase =

   256
```

```
>> phase2_1 * 180/pi

ans =

   180
```

④ S1 初相点为 46，S2 初相点为 87 （一个周期对应 512 个点）

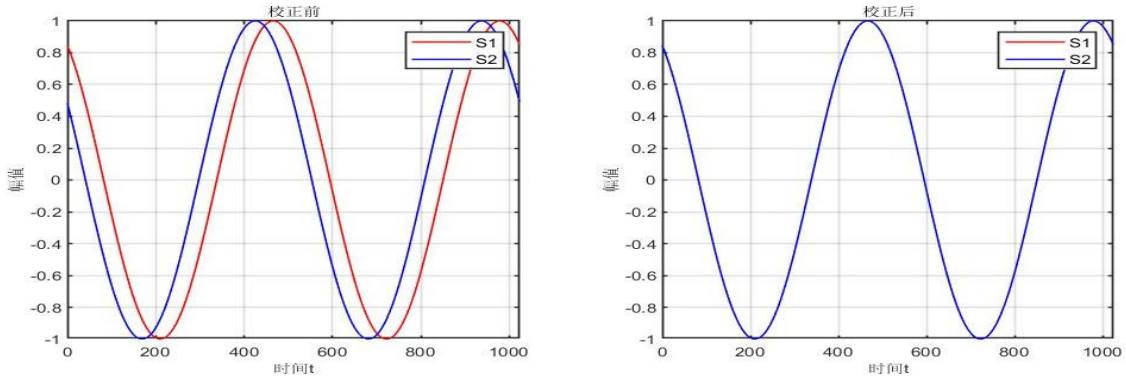


图 2-4: MATLAB 仿真实验结果 4

如上所示，可正确校正相位差点数相差 41，及 41/512 * 360° =28.82° 。

仿真求得其相位差点数及相位差度数（度°）如下：

```
>> N_phase

N_phase =

   41.0000
```

```
>> phase2_1 * 180/pi

ans =

   28.8281
```

通过上述 MATLAB 仿真实验，我们验证了所设计的相位校正算法，进行不同初始相位差条件下的校正能力测试，得到了如下结论：

① 能够实现 1° 以下相位差的两组信号校正输出，最小为 0.703125°；

② 能够实现余弦信号 S2 与余弦信号 S1 相位差 180° 范围内的校正输出。

2.3 坐标旋转数字计算(CORDIC)算法

前面我们提到的相位校正算法中需要通过反正切运算对频域信息的实部虚部进行处理得到相位值，而在硬件 FPGA 中实现这 样的复杂函数较为困难，那么坐标旋转数字计算方法(CORDIC)能够通过简单运算实现反正切计算，即通过基本的加减和移位运算代替乘法运算，无限逼近所需值，从而得到对应的数值解。

CORDIC 算法的基本原理是通过旋转角度θ分成多个连续的小偏转角，通过逐次摇摆逼近目标旋转角度来完成旋转过程。以圆坐标系旋转为例来分析旋转过程。如下图所示，在xy坐标平面内将点(x_1, y_1)旋转θ角度到点(x_2, y_2)，其数学逻辑可如下表示：

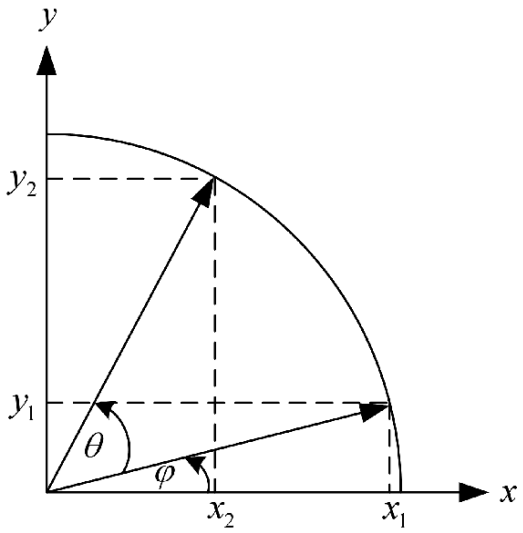


图 2-5 CORDIC 坐标旋转示意图

$$x_2 = \sqrt{{x_1}^2 + {y_1}^2} cos(\theta + \varphi) = x_1 cos\theta - y_1 sin\theta \tag{6-1}$$

$$y_2 = \sqrt{{x_1}^2 + {y_1}^2} sin(\theta + \varphi) = x_1 sin\theta + y_1 cos\theta \tag{6-2}$$

换算成矩阵形式如下：

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} \tag{7}$$

对公式 6-1、6-2 进行变形可得如下形式：

$$x_2 = x_1 cos\theta - y_1 sin\theta = cos\theta(x_1 - y_1 tan\theta) \tag{8-1}$$

$$y_2 = x_1 sin\theta + y_1 cos\theta = cos\theta(y_1 + x_1 tan\theta) \tag{8-2}$$

两边同时除以$cos\theta$，即可得到伪旋转方程：

$$\hat{x}_2 = x_1 - y_1 tan\theta \tag{9-1}$$

$$\hat{y}_2 = y_1 + x_1 tan\theta \tag{9-2}$$

通过省略$cos\theta$可以简化旋转过程中的计算，但余弦项不应该被忽略，通过伪旋转角度来进行计算，又由于$tan\theta = 2^{-i}$，那么上式可化简为：

$$\hat{x}_2 = x_1 - y_1 tan\theta = x_1 - y_1 2^{-i} \tag{10-1}$$

$$\hat{y}_2 = y_1 + x_1 tan\theta = y_1 + x_1 2^{-i} \tag{10-2}$$

在变换成 2 的次方迭代运算后，将正切乘法变换成移位操作，可以很好的通过硬件实现，并且对旋转角度变换进行一些特定处理，使用固定的一些角度来进行无限逼近，使得对任意旋转角度都能够通过连续的小角度的迭代来完成角度值的求解，那么 FPGA 内部的大量逻辑资源能够通过移位、加法及减法运算实现 CORDIC 算法的计算功能，使得反正切值的求解简单化。

CORDIC 算法中每次迭代的特定角度值如下表所示，迭代上限 13 次。

i	$tan\theta$	θ^i	$cos\theta$
1	1	45.0	0.707106781
2	0.5	26.5550511771	0.894427191
3	0.25	14.0362434679	0.9701425
4	0.125	7.1250163489	0.992277877
5	0.0625	3.5763343750	0.998052578
6	0.03125	1.7899106082	0.999512078
7	0.015625	0.8951737102	0.999877952
8	0.0078125	0.4476141709	0.999969484
9	0.00390625	0.2238105004	0.999992371
10	0.001953125	0.1119056771	0.999998093
11	0.000976563	0.0559528919	0.999999523
12	0.000488281	0.0276764526	0.999999881
13	0.000244141	0.0139882271	0.999999997

那么对应迭代 13 次后，$cos\theta$ 的乘积为：
$cos45^\circ \times cos26.5^\circ \times cos14.03^\circ \times \cdots \times cos0.028^\circ \times cos0.014^\circ \approx 0.607252941$ ，
其倒数为$\frac{1}{\sum cos\theta_i} = 1.64676024187$，即在迭代完成后需要乘以这样的一个系数，
对于每一次迭代，其伪旋转都可以表示为：

$$x^{i+1} = x^i - d_i \cdot \left(2^{-i} y^i\right) \tag{11-1}$$

$$y^{i+1} = y^i + d_i \cdot \left(2^{-i} x^i\right) \tag{11-2}$$

其中d_i为角度累加器，只有±1 两个值，用来确定角度旋转的方向，在迭代过程中旋转角度的关系式如下：

$$z^{i+1} = x^i - d_i \cdot \theta^i \tag{12}$$

那么通过每次迭代的d_i及其对应的迭代角度，我们最终可以得到其对应的角度值。

三、 基于 FPGA 的射频信号相位校准实现

本设计采用 BQR7VX690T FPGA 开发板进行射频正弦信号校准，通过在线逻辑分析仪 ILA 进行信号抓取观察，并通过上位机监测两组信号相位差值并显示两组信号校正后是否对齐。系统架构图如图 3-1 所示，由于条件限制，我们没有 AD9361 等器件，只能用开发板的射频 SMA 口模拟射频信号的收发 ，以验证程序设计实际上板的功能效果。

其中开发板通过接收射频信号进行相位校准，也是本文档介绍的核心内容，它负责将输入的存在相位差的信号通过我们的算法进行处理和恢复，并输出校准后的信号，最后通过上位机来检验校准的结果，（本系统采用 Vivado 的内部逻辑分析仪（ILA）来检验校准的结果）两组发射信号参数如下：

a) 均为余弦信号： $y=\cos\left(2\pi f \bullet t+\varphi\right)$；

b) 线速率（Line rate）2Gbps；

c) 外部接口及余弦信号幅值位宽：16 bits；（采用 8B/10B 编码）

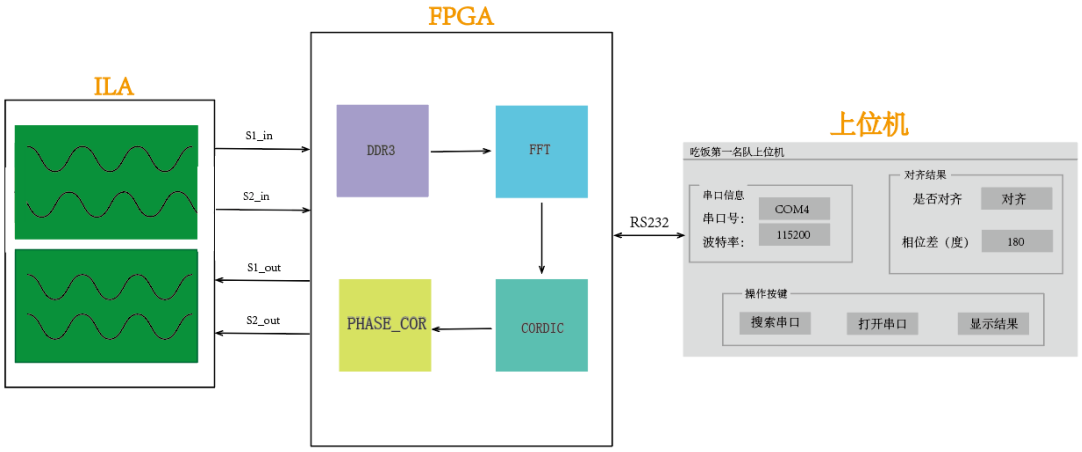


图 3-1 系统架构图

下面将从 Verilog 代码算法实现、上位机及下位机设计、FPGA 仿真实验、FPGA 上板测试实验等四个方面详细介绍 FPGA 的射频信号相位校正实现内容。

3.1 Verilog 代码算法实现

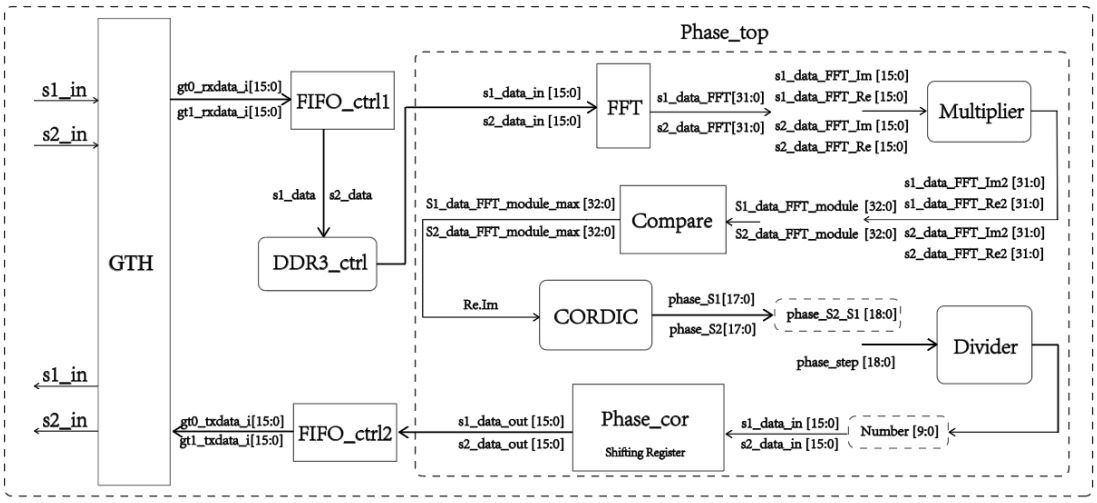


图 3-2 算法硬件实现模块框图

图 3-2 FPGA 实现的射频信号校准系统模块组成和校准算法模块框图

图 3-2 展示了我们设计的完整的射频信号校准模块的的层次以及校准算法模块的组成。其中：

3.1.1 GTH 模块

负责进行数据收发，接收外部输入串行数据，通过串并转换得到并行数据给后面的模块进行处理，同时通过其中的 CDR 模块恢复出有关时钟给后面的模块使用，此外还负责将经过校准恢复后的数据进行并串转换成后发送给下游器件。

3.1.2 FIFO_ctrl 模块

分两块，接收数据 FIFO_ctrl1 负责完成接收数据帧头处理，通过跨时钟域去掉因数据传输所添加的帧头，发送数据 FIFO_ctrl2 负责完成发送数据帧头的添加以实现下一个接收端的同步。

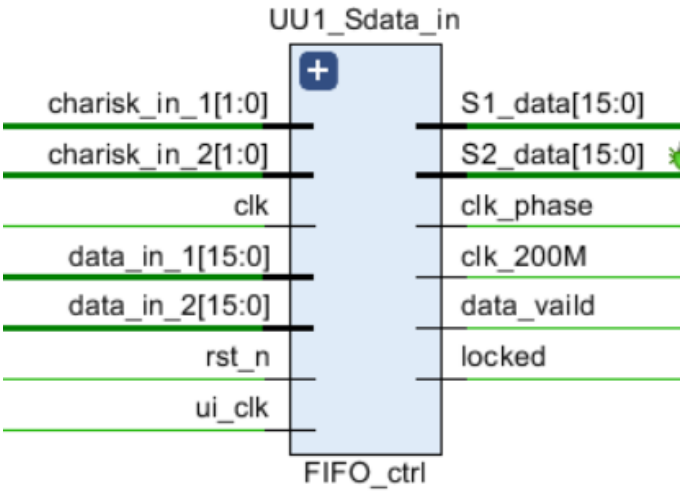


图 3-3 FIFO 控制模块图

3.1.3 DDR3_crtl 模块

负责完成高速数据的缓存并提供信号数据给后续 Phase_top 模块进行处理。DDR3 控制模块分为三个模块，第一个模块为数据缓存模块，主要负责 DDR3 进入数据的缓存，第二个模块为 DDR3 读写控制模块，主要负责 DDR3 的读写，最后一个模块为数据缓冲输出模块，负责将 DDR3 的读出的信号连续输出。其模块示意图如下所示：

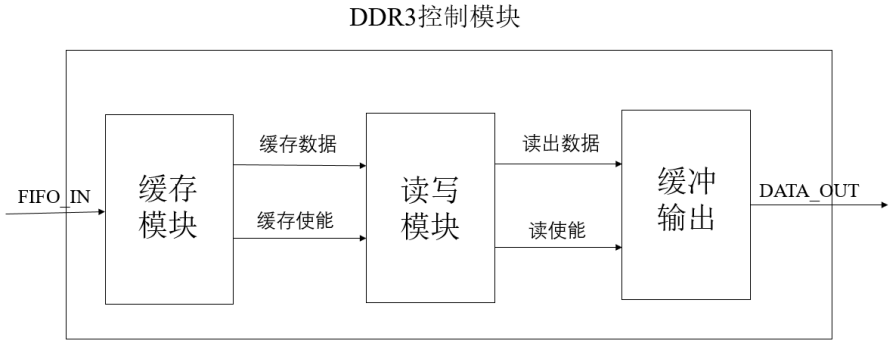


图 3-4 DDR3 控制模块示意图

① 缓存模块

由于 DDR3 写入的过程是一个随机过程，而 FIFO 输出是一个连续过程，所以需要先将 FIFO 输出的数据缓存下来，然后交给 DDR3 进行读写操作，缓存的过程就是利用计数器，将一个周期（512 个点）的正弦信号记录下来，保存到存储器中。记录完成后输出缓存使能信号。

② DDR3 读写控制模块

DDR3 的读写逻辑由读写状态机来控制，其读写控制状态机示意图如下：

首先复位进入空闲状态，当 DDR3 初始化完成后进入写模式，将缓存模块的数据写入到 DDR3 的固定地址中，当计数器记录到写入最后一个数据时，状态机进入等待状态，等待状态将地址位复原成最开始的状态，等待一个时钟周期后，进入读状态，读状态按照地址将写入到 DDR3 的数据读出来传给缓冲输出模块。

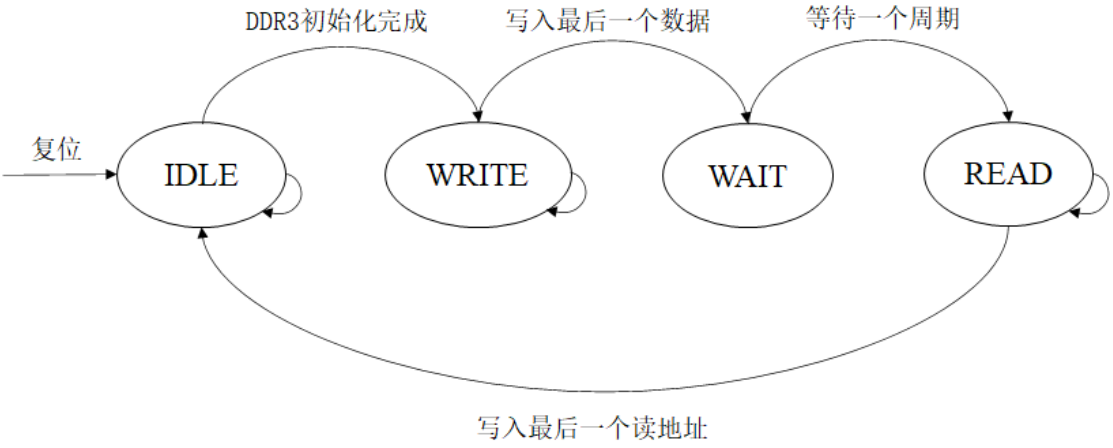


图 3-5 DDR3 读写控制状态机流程示意图

DDR3 的读写时序如下：

写使能：即写指令,命令接收和数据接收都准备好并且状态机处于写状态，此时拉高写使能，也就是将 app_cmd 置为 1。

DDR3 读写地址：随机选取一部分 DDR3 地址（本次选用 OX4000），然后每写入或者读出一个数据，该地址递增 8。

读使能：当状态机出去读状态，且 DDR3 准备好，拉高读使能。将数据读出

③ 缓存输出模块

由于 DDR3 读出也是一个随机状态，所以我们需要将 DDR3 读出的信息缓存，保证信号的连续性，当读出信号有效的时候收集读出的数据并缓存。然后将缓存信号输出传给下一个模块。

3.1.4 Phase_top 模块

是相位差校准算法模块，负责实现射频信号的相位校准，也是本设计中最为核心的处理部分。下面将对 Phase_top 模块中算法的实现进行详细介绍。

4_1 FFT 模块

FFT 模块主要实现将两路 16 比特位宽的余弦信号数据进行快速傅里叶变换（FFT），从而得到对应的频谱，分别以实部虚部值体现，频谱点模值最大的点包含了其对应的相位信息，对应实现算法原理公式（1）的功能，如下图所示，两路 16 比特位宽的信号及其有效信号输入，通过此模块计算得到其对应的实部虚部值。

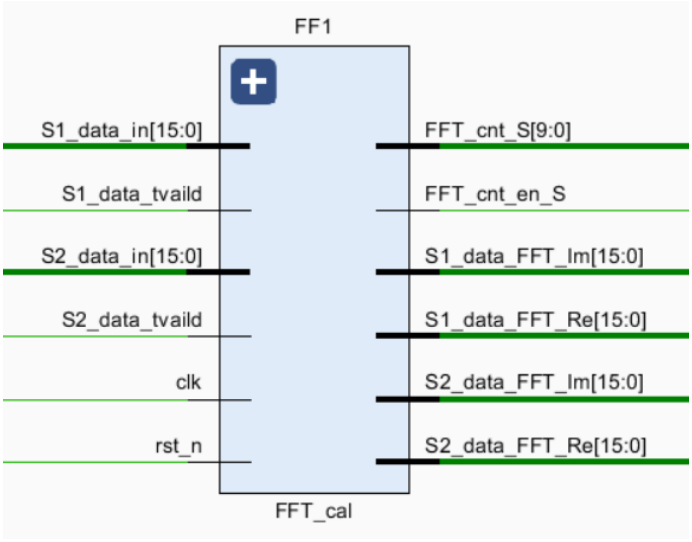


图 3-6 FFT 运算模块图

4_2 模值运算模块

由于需要找到频谱值中信号频率点对应的实部虚部值，不能通过简单的比较其频谱点的实部或虚部以确定中心频率点，此处通过对实部虚部值乘方后的加法运算得到其模值，模值运算模块由 1 级 pipeline 的乘法器和加法器来实现，模块图如下所示。

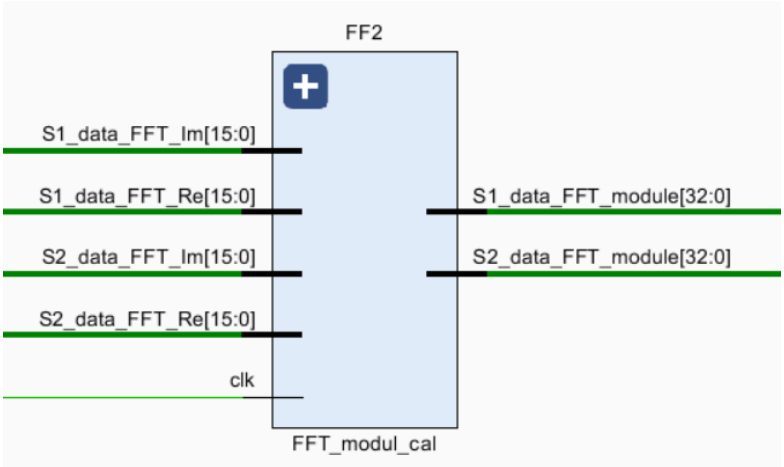


图 3-7 模值运算模块图

先将经过 FFT 模块得到的 S1_data_FFT_Re，S1_data_FFT_Im，，S2_data_FFT_Re，S2_data_FFT_Im 两对虚实信号经过乘法器运算后将将对应的平方信号，然后将虚部和实部的平方信号经过加法器后得到两路 FFT 输出信号的模值信号给比较器模块求最值。

4_3 比较器模块

通过流水线的模式分别对信号频域模值进行比较以求出其最大值，并获取其对应的频率点的位置，从而获得其对应的实部虚部值，以供后续求解信号相位值。比较器实现流程示意图如下所示：

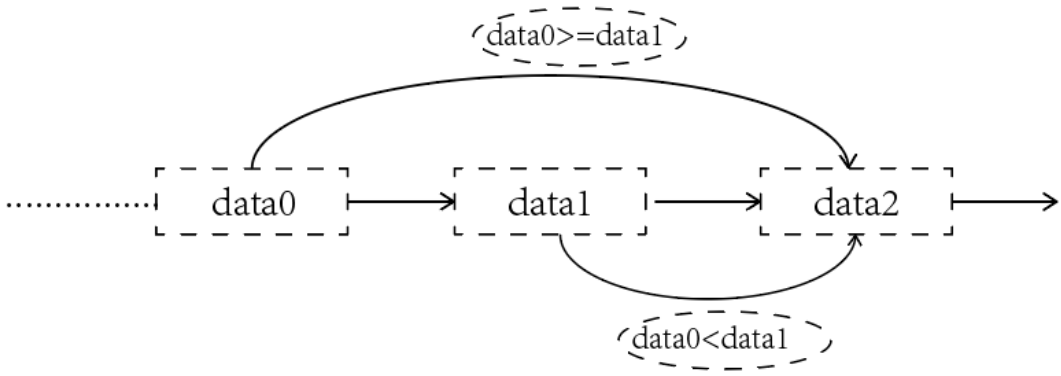


图 3-8 比较器流程示意图

比较器模块图如下所示，两组模值信号输入，并结合使能信号，得到最大值对应的频域值的位置信息。

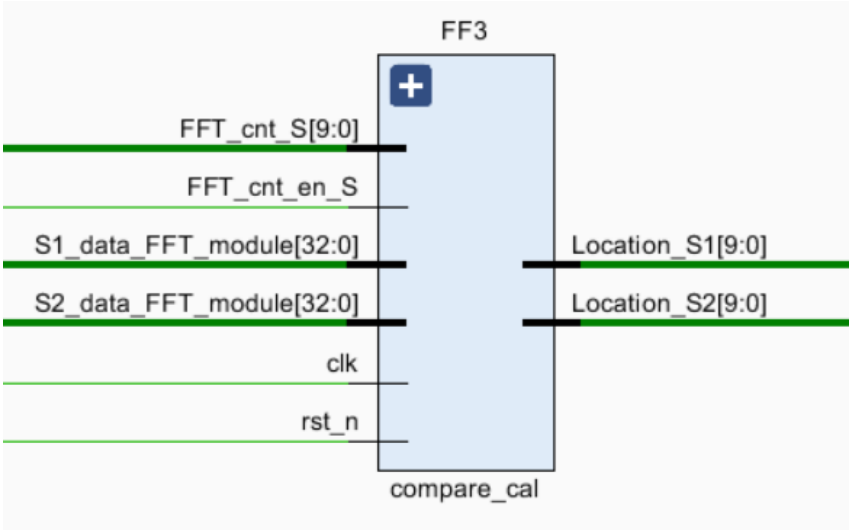


图 3-9 比较器模块图

4_4 CORDIC 模块（反正切计算）

在得到信号频率点对应的频谱实部虚部值后，该模块输入比较器模块获得的最大模值对应的 S1_data_FFT_Re，S1_data_FFT_Im，，S2_data_FFT_Re，S2_data_FFT_Im 信号，经过 arctan 运算后得到两路信号的相位信息 phase_S1 和 phase_S2。具体通过坐标旋转数字计算法（CORDIC）以实现，即实现原理中公式（2）的功能。具体模块图如下所示。

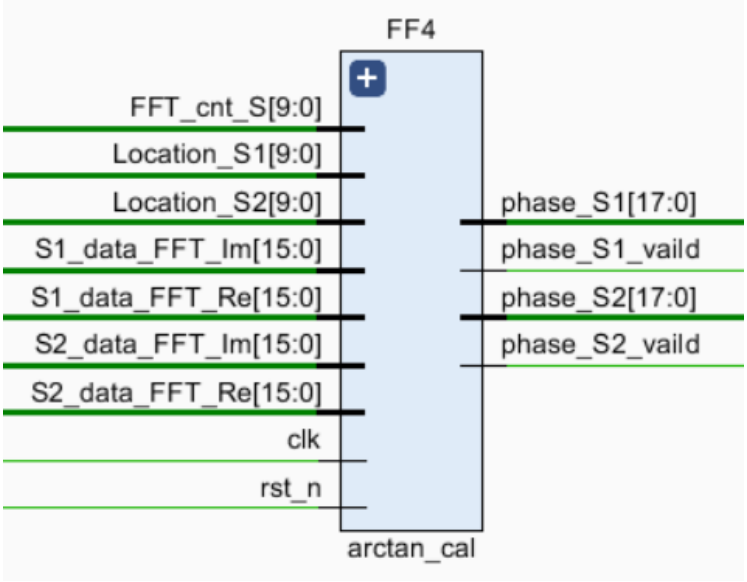


图 3-10 反正切计算模块图

4_5 相位差及相差点数计算模块

对输入的两路信号初相值结合其有效信号进行减法运算即可得到其对应的相位差值 phase_S2_S1，并结合获取的信号频率点位置去确定我们的相位分辨率的值，再结合相位分辨率 phase_step 通过除法器即可求得相位差对应的相差点数 Number，并输出两路信号的相位差 phase，如下是相位差及相差点数计算模块图

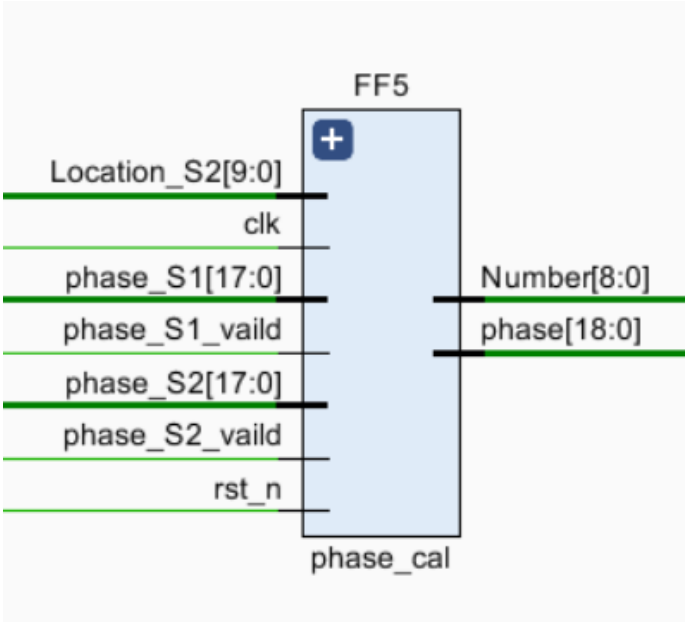


图 3-11 相差及相差点计算模块图

4_6 信号相位补偿模块

再得到相位差及相差点数后，通过对信号 S1，S2 的处理即可补偿其相位差使得同相输出，具体方法如下：保持 S1 的原输出，通过相差点数对 S2 进行相应的滞后输出即可恢复其与信号 S1 的相位相同。具体的滞后实现我们使用移位寄存器（Shifting Register）实现：首先通过较大位宽的寄存器来实现对 S2 数据的缓存，每一个时钟 S2 数据输入寄存器的[15:0]位，并同时在每一个时钟寄存器数据向前移位 16bit，通过对应的相位差点数 Number，我们输出移位寄存器中对应位置的 S2 数据，以完成相位差值的补偿，输出两路信号数据到下一个模块，其中移位寄存实现补偿原理图如下图所示。

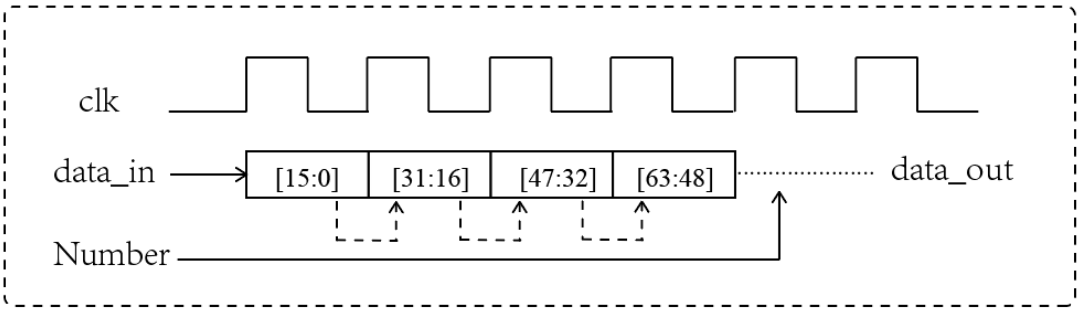


图 3-12 移位寄存器实现相位补偿示意图

如下为信号相位补偿模块图。

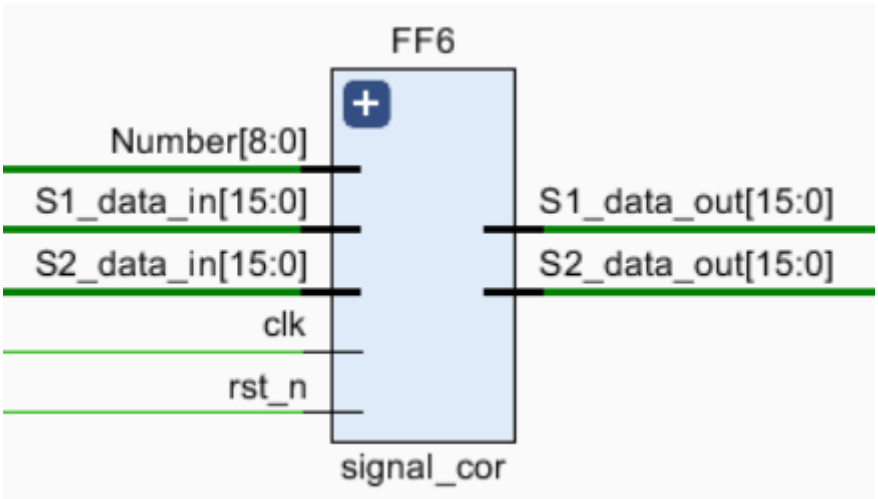


图 3-13 信号相位补偿模块图

3.1.5 uart_top 模块

该模块是与上位机通信的下位机 FPGA 程序，通过 RS232 串口交互数据与上位机进行通信，将计算出的两路信号初始相位值发送到上位机，在校正后，将判决两路信号是否对齐的信息发送到上位机，完成上位机对两路信号初始情况与校正后情况的监测，下图是此功能实现的模块图。

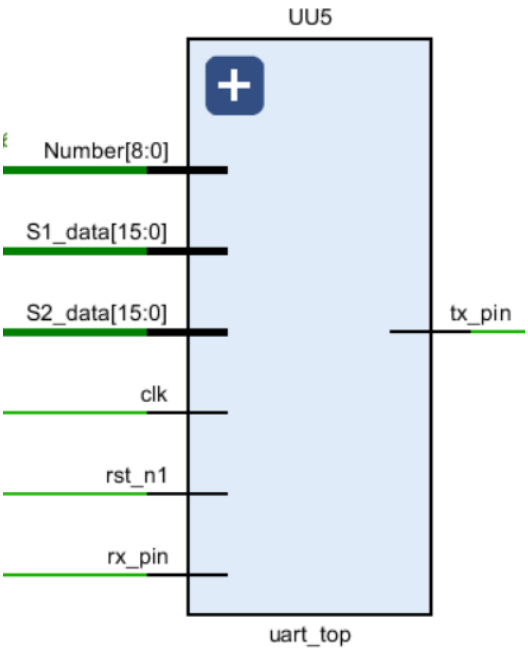


图 3-14 uart_top 模块图

3.2 上位机及下位机设计

3.2.1 通信协议

上下位机的通信协议：上位机与 FPGA 之间通过 RS232 进行通信，由于通信速率限制，没办法实时显示出波形，所以我们将对准信号和两信号间的相位差作为传输数据传输到上位机，在上位机上面进行显示。

RS232 是以字节来传输数据，每次只能传输一个字节。其传输数据的形式如下图所示:

图 3-15 传输数据格式图

其中起始位是低电平，中间是 8bit 是数据，最后是高电平的停止位。

发送协议如下图所示：

图 3-16 发送协议图

整个协议分为 6 个字节，其中下位机第一个字节发送 EE 作为对准信息的起始位，方便上位机检测对准信息，然后发送 FFFF 作为相位信息的起始位，上位机通过检测 FFFF 来分选出相关的相位信息。

接收协议：上位机通过操作按键来传输字符 AB 到下位机，当下位机检测到该字符，下位机发送相关的对准与相位信息。

3.2.2 上位机说明

上位机采用 QT 进行编写，封装成 exe 程序，具有较强的独立性和通用性。其 UI 设置如下图所示：

图 3-17 上位机 UI 图

首先在左上方展示串口信息，包括串口号和波特率。右上方展示对准结果，包括两个信号是否对其，两个信号相差的相位信息。下面是按键操作，包括搜索串口、打开串口、显示对准结果三个功能。其中搜索串口可以搜索串口号和波特率。打开串口的作用是开始接收区域。由于两个信号对准后，其相位差和是否对准是固定的，为了系统的功耗考虑，使用按键显示对其结果，而不是让串口一直发送数据。当我们需要的查看对准结果的时候，点击显示结果按钮就可以显示对准信息。

3.2.3 下位机说明

下位机由三个模块组成，分别是顶层的控制模块，下面的发送模块与接收模块。其中发送与接收模块实现的功能分别是发送一字节数据与接收一字节数据。

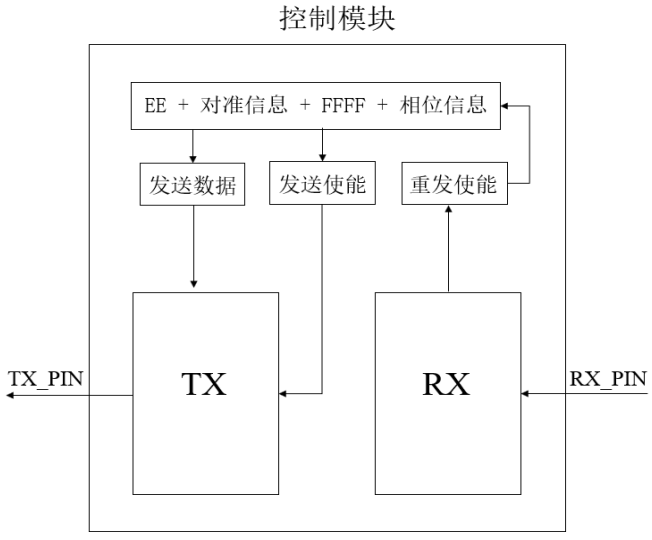


图 3-18 下位机模块控制图

控制模块利用计数器控制发送的信息，使发送模块按照约定好的协议形式发送数据，当接收到重新发送信号的时候，控制模块控制发送端重新发送相关的数据信息。

3.2.4 上下位机通信结果

打开上位机的串口，当下位机对准的时候，下位机按照约定的协议发送相关数据信息，上位机通过显示结果的按键操作来显示对准结果，对准结果包括两个信号是否对准以及这两个信号相差的相位度数。其结果图如下图所示。

图 3-19 上下位机通信结果图

3.3 vivado 仿真实验

通过调用 DDS IP 核产生两组同频同振不同相的余弦信号作为相位校正程序的输入，其在 vivado 中几种不同情况下的仿真结果如下：

① 仿真实验一

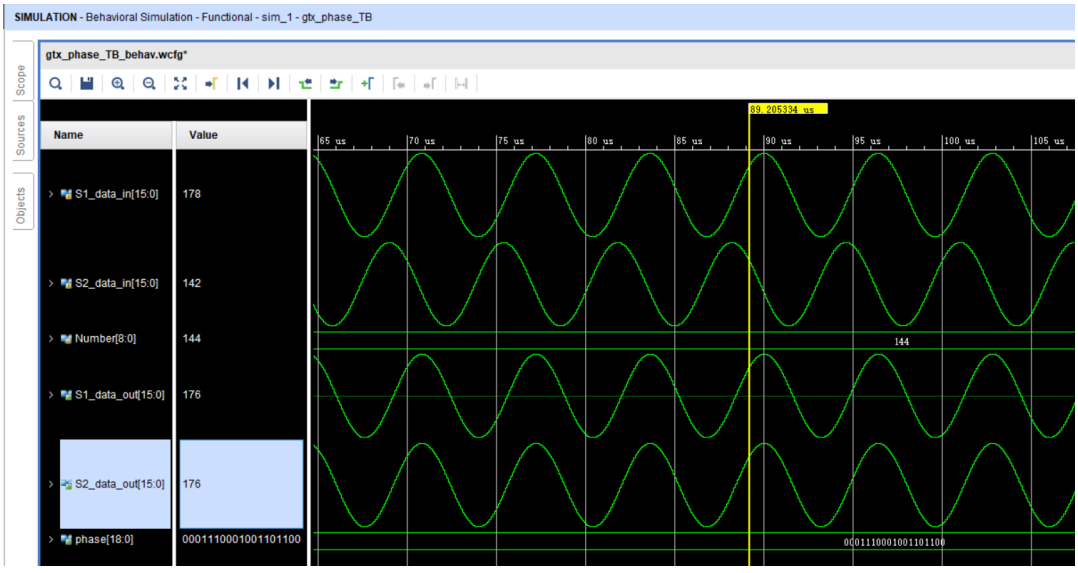


图 3-20：vivado 仿真实验一

S1_data_in、S2_data_in 为我们的两组余弦输入信号；S1_data_out、S2_data_out 是校正后的输出信号；Number 是计算出的 S2 与 S1 相位相差点数，phase 是两路信号的相位差，采用定点数 fixed 19_15 格式，换算值约为 1.768 rad，结果正确。如上图所示，校正后的输出信号同相。

② 仿真实验二

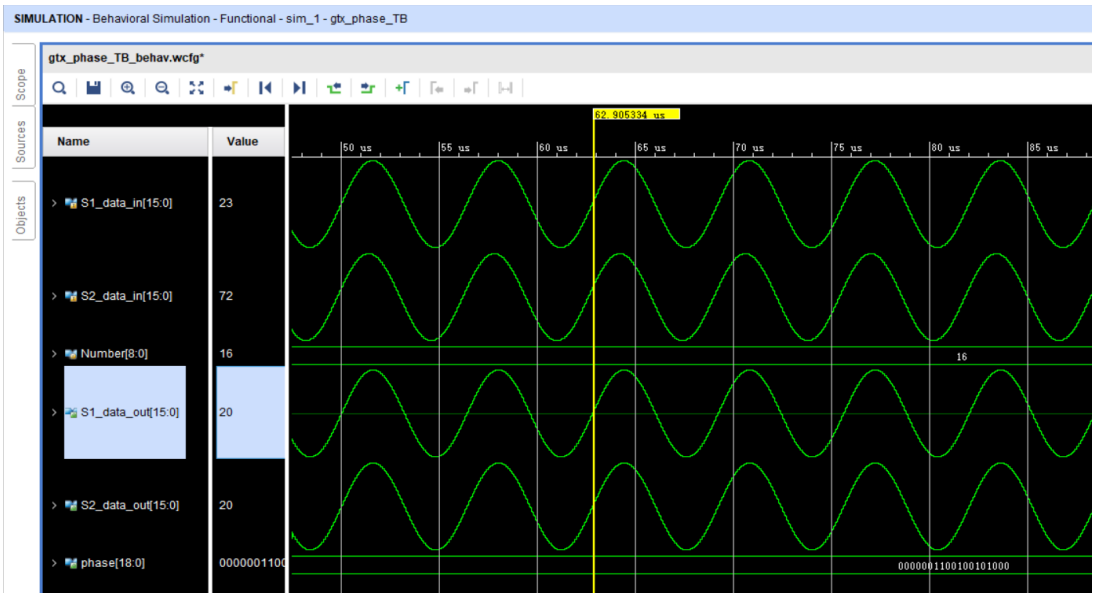
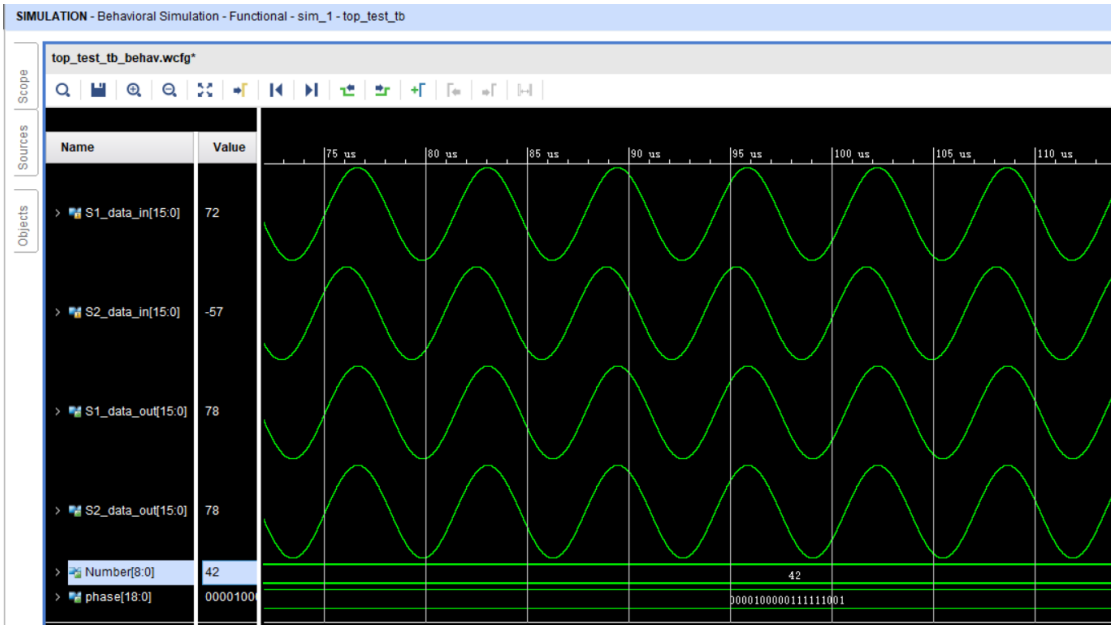


图 3-21：vivado 仿真实验二

如上所示，相位差 phase 换算后值约为：0.196 rad，同时输出信号同相。

③ 仿真实验三



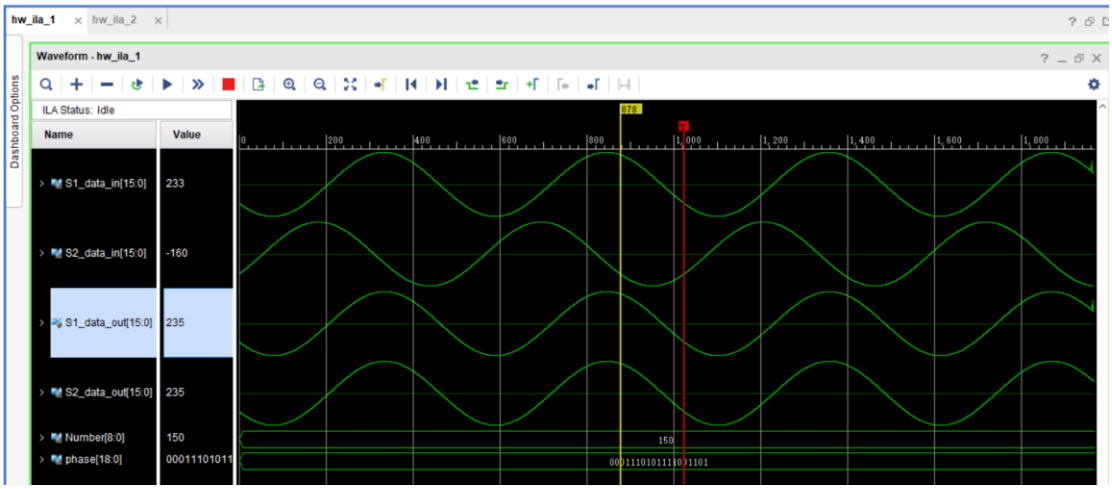
如上所示，相差点为 42 的两路信号进行校正后实现同相输出。

通过上述 vivado 仿真实验，验证了设计能够实现两组余弦信号相位校准，并同相输出。同时经理论分析，由于数据精度设置不够的原因所导致极少部分情况会校正后相差一个点，差值在 1° 以内，满足误差要求。

3.4 FPGA 上板测试实验

将完整程序进行上板测试程序可行性，通过 ILA 抓取相位校正模块的输入输出信号、相位点差值等，具体实验结果如下：

① 上板实验结果一



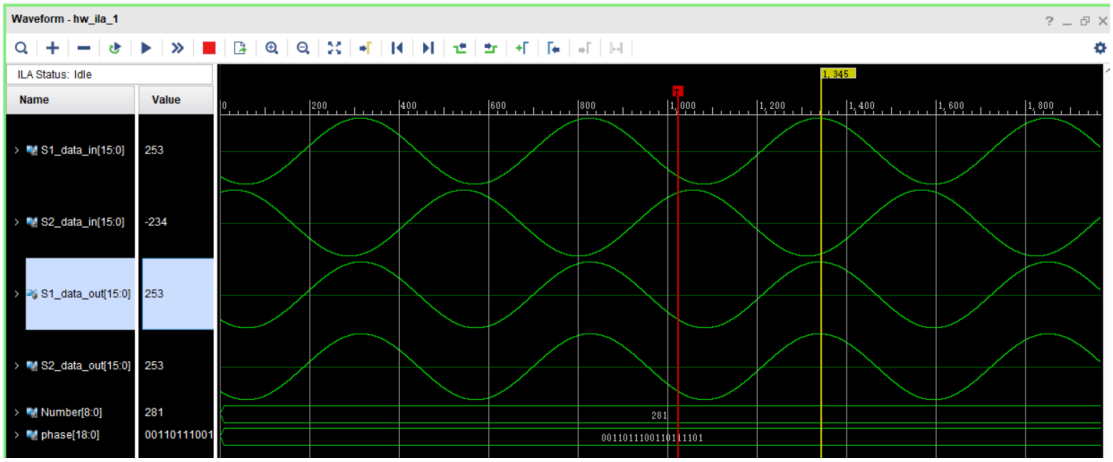
a. ILA 信号显示图

b. 上位机显示图

图 3-22 FPGA 上板测试实验一

如上所示，此次实验测试准确校准，同相输出，上位机显示相位差正确。

② 上板实验结果二



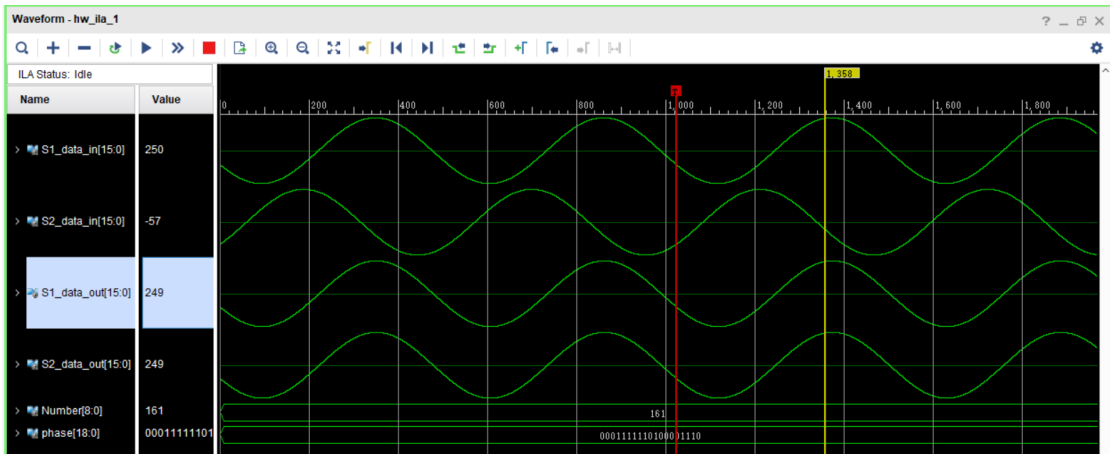
a. ILA 信号显示图

b. 上位机显示图

图 3-23 FPGA 上板测试实验二

如上所示，此次实验测试准确校准，同相输出，上位机显示相位差正确。

③ 上板实验三



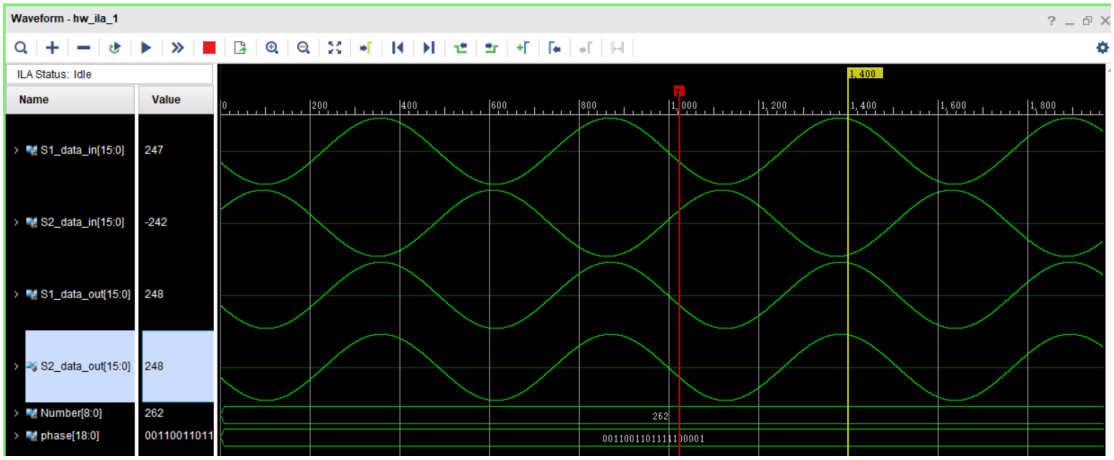
a. ILA 信号显示图

b. 上位机显示图

图 3-10 FPGA 上板测试实验三

如上所示，此次实验测试准确校准，同相输出，上位机显示相位差正确。

④ 仿真实验结果四

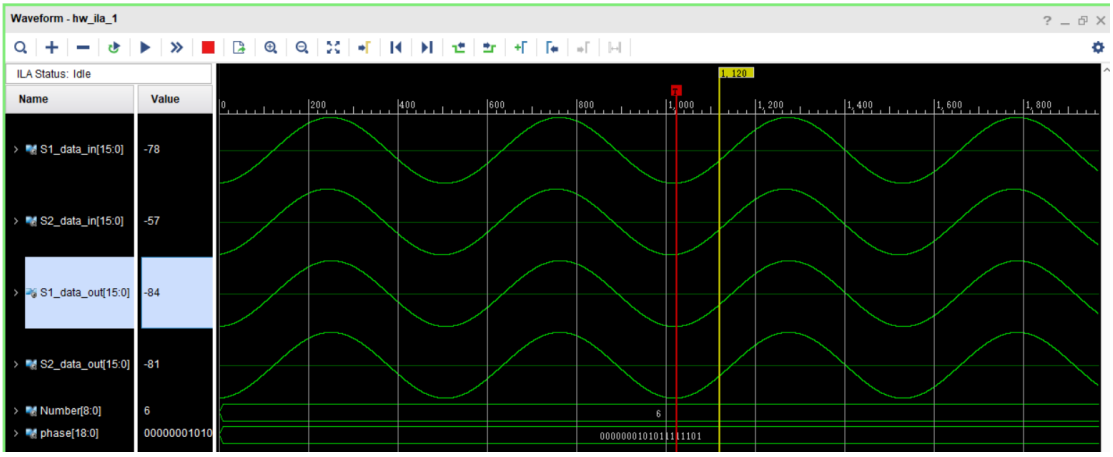


a. ILA 信号显示图

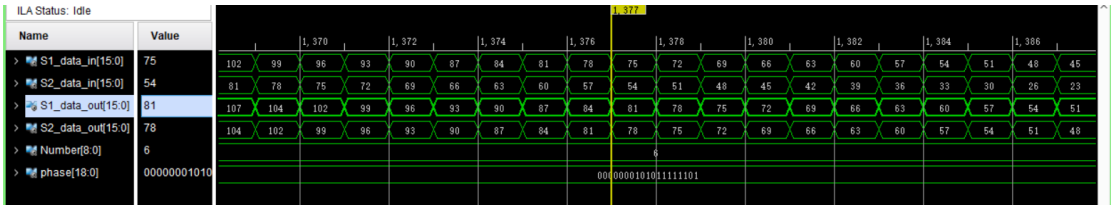
b. 上位机显示图

如上所示，此次实验测试准确校准，同相输出，上位机显示相位差正确。

⑤ 仿真实验结果



a. ILA 波形模拟显示图



b. ILA 波形数字显示图

c. 上位机显示图

如上所示，通过模拟波形及数字波形对比，可知此次实验校正后相差一个相位点，差值为 0.703125°，在 1°误差范围内，符合要求。

如上所示，实验一二三四均实验了完全同相的校正输出，实验无校准有一个点的误差，在 1°误差范围内，由于数据精度问题。

实验结论：与 vivado 仿真实验结果一致，能够实现两信号相位校正，少量情况下由于数据处理过程中，数据精度不够的问题造成一个相位点的误差，即能够实现输出信号相位差在 1°范围内的校正。

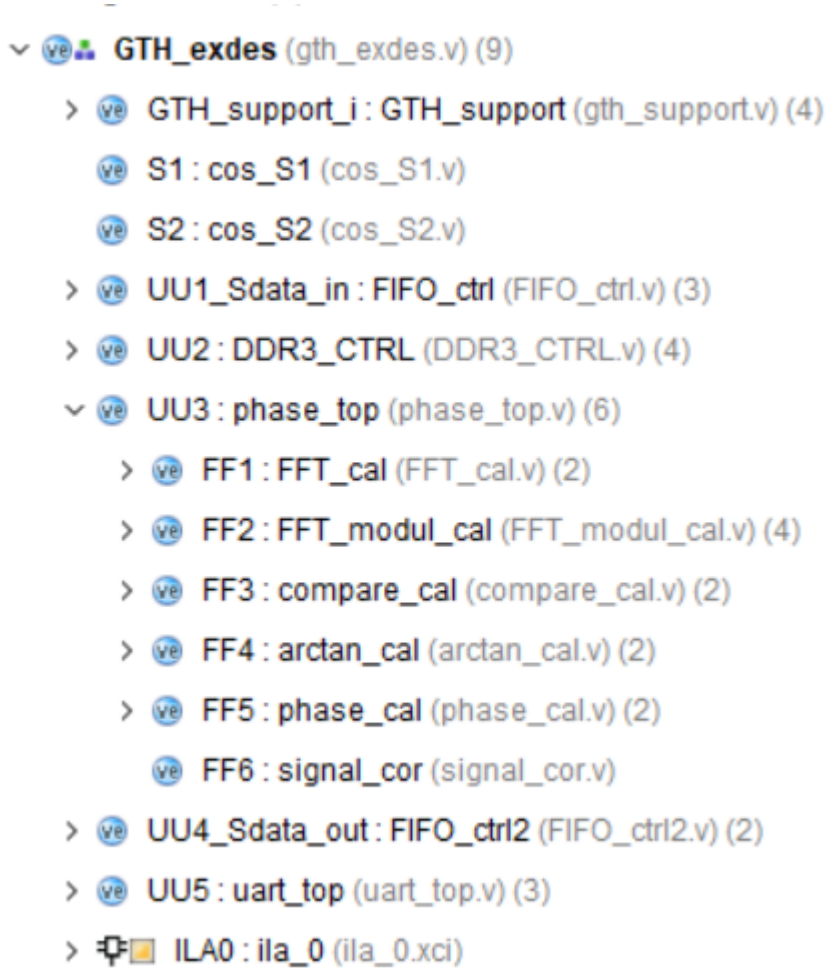
四、 总结与展望

针对两组相位不同的余弦信号的校准，在硬件实现相位校正算法上，我们采取 CORDIC 算法实现反正切函数求取相位差，通过移位寄存器实现相位差点的相位补偿（即实现对应的延时输出）。

大部分相位差情况能够实现准确校正并同相输出，同时由于硬件实现过程中信号位宽的限制，部分情况下会与准确值出现一定的误差，使得此情况下校正后的输出信号会差一个相位点，均在实验结果中有所体现，输出误差在 1° 范围内。

综上所述，此设计能够实现两组余弦信号能识别相位差在 1° 以上的输入信号，并实现输出信号的相位差在 1rad 以内，并能够通过上位机程序监测两路信号初始相位差值，及校正后两路信号是否对齐。

附：整个程序的框架图



各模块介绍：

GTH_exdes ：顶层模块，连接 GTH、DDR3、相位处理、上位机、ILA 等模块；

GTH_support_i: GTH_support ：GTH 有关的模块，用作信号收发；

S1:cos_S1 : 余弦信号 S1 的产生模块，连接到 GTH 的 TX 端 1；

S2:cos_S2：余弦信号 S2 的产生模块，连接到 GTH 的 TX 端 2；

UU1_Sdata_in：FIFO_ctrl： 异步 FIFO 模块去除数据中的通信帧头；

UU2:DDR3_CTRL ：DDR3 读写控制；

UU3:phase_top ：相位处理顶层模块，进行子模块的连接控制；

FF1:FFT_cal：FFT 傅里叶变换频域计算模块；

FF2:FFT_modul_cal：频域模值计算模块；

FF3:compare_cal：模值比较模块，得到最大模值及其对应的位置；

FF4:arctan_cal：反正切计算模块（CORDIC），计算相位；

FF5:phase_cal：将相位差结合相位分辨率用除法器计算相差点数；

FF6:signal_cor: 通过相差点数对信号进行补偿实现相位校正；

UU4_Sdata_out: FIFO_ctrl2：校正后的两路信号添加通信帧头以供发出同步；

UU5：uart_top：与上位机通信的下位机程序；

ILA0:ila_0： ILA 抓取信号进行观测。