（课程设计）： Cordic ASIC 设计

设计任务

基于 SMIC 180nm 标准数字工艺设计一个 Cordic 的 ASIC 芯片。 输入输出数据是 13 位定点符号数(10 位小数部分)，芯片的输入输出如表 1。

表 1. 芯片接口说明

引脚名称	位宽	方向	说明
CLK1	1	输入	接口模块时钟信号
CLK2	1	输入	Cordic 模块时钟信号 (速率是 CLK1 的一半)
Rst_n	1	输入	Rst_n 是异步复位信号， 低电平有效
Data	13	输入输出	输入输出数据 作为输入时，上升沿采样 作为输出时，在上升沿给出输出
Data_ready	1	输出	电平敏感，高电平有效
Enable	1	输入	电平敏感，高电平有效
IN_N_OUT	1	输入	I/O 口输入输出控制 电平敏感信号，高电平为输入，低电平为输出

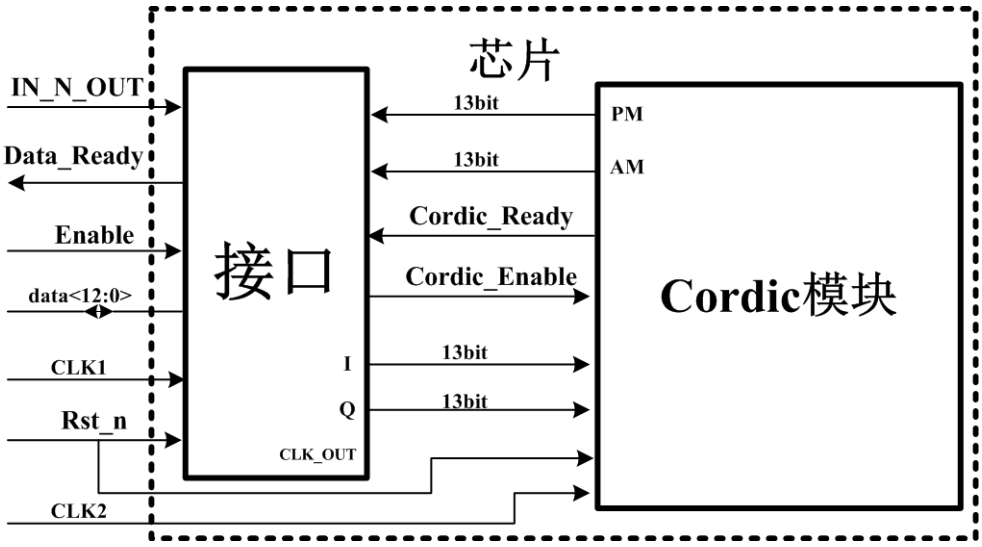

图 1. 芯片接口示意图

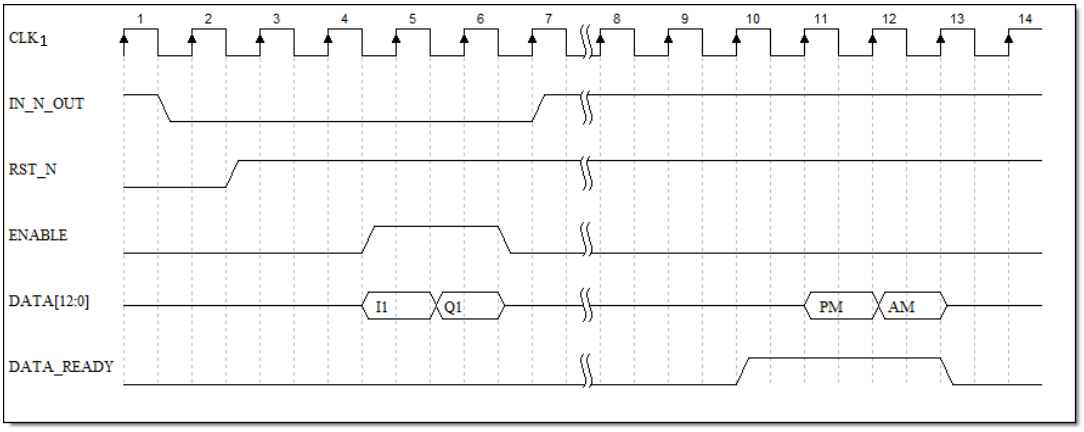

图 2. 芯片接口时序图

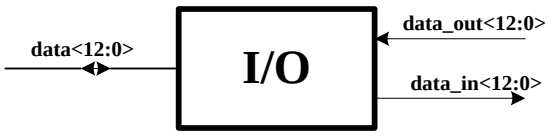

图 3. 双向 IO 口示意图

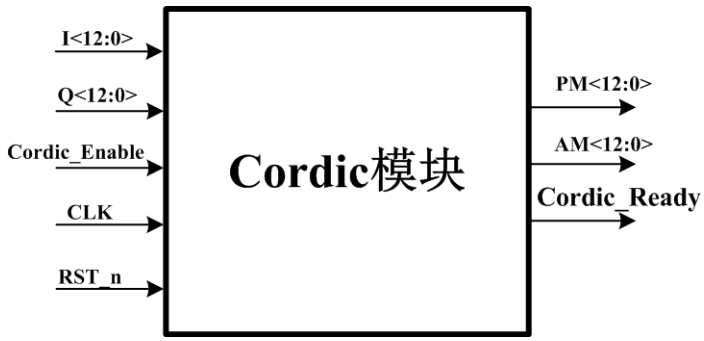

图 4. Cordic 模块接口示意图

芯片工作方式说明

芯片接口的时序如图 2 所示，在接收数据时，Enable 先拉高，在 CLK1 的下一个时钟周期的上升沿，接口把时钟为 CLK1 的串行数据转换为时钟为 CLK2 的 I/Q 两路数据(串并转换)，并且把这个数据传给 cordic 模块进行运算。在输出数据时， cordic 模块把时钟为 CLK2 的 PM/AM 两路数据传给接口，接口对其完成并串转换后，Data_ready 拉高，在 CLK1 下一个周期的上升沿把数据串行输出。需要注意的是，IN_N_OUT 实际上是 IO 的收发控制端，如图 3 所示，在 IO 口内侧有 data<12:0>的输入输出口，由 IN_N_OUT 选择输入还是输出，这种 IO 口由 smic 180nm 工艺库提供，具体说明见设计库说明文档。**需要注意的是这里的 CLK1 的时钟速率是 CLK2 的 2 倍。**

Cordic 模块实现了笛卡尔坐标向极坐标转变。

输入数据为二维平面中的点在笛卡尔坐标下的两个正交分量 I，Q，输出数据为该点在极坐标系中的 PM（相位），AM（幅度）。

即：

$$AM = \sqrt{I^2 + Q^2}$$

$$PM = \arctan\left(\frac{Q}{I}\right)$$

如图 4 所示，输入信号包括由接口模块给出的 I、Q 两路数据、Cordic_Enable 信号以及外部输入的 480KHz 时钟、Rst_n 复位信号；输出信号包括坐标转换后的 AM、PM 信号(幅度和相位)以及 Cordic_Ready 信号。当接口准备好并行的 I、Q 数据之后，将 Cordic_Enable 信号拉高，Cordic 模块开始接收 I、Q 数据。Cordic 模块完成坐标转换之后，Cordic_ready 信号拉高，接口模块开始接收 Cordic 模块输出的 AM、PM 数据，当接口模块接收完成之后 Cordic_Enable、Cordic_ready 信号拉低。

定点数的量化和溢出处理

由于设计要求输入输出都采用 13 位定点有符号数，输入 IQ 的格式为 1 位符号位，2 位整数、10 位小数，输出幅度的格式为：1 位符号位，3 位整数、9 位小数，输出相位的格式为：1 位符号位，2 位整数、10 位小数。

1．计算过程中的精度规定：

为了保证计算精度，要求 CORDIC 的中间计算过程中精度更高。计算过程中每次得到的迭代结果 In,Qn（n 表示第 n 次迭代）都用 18 位表示（1 位符号位，3 位整数位，15 位小数位）。迭代计算过程中的相位用 18 位表示（1 符号位，2 个整数位，15 个小数位），相位采用的是弧度单位，即相位的范围是[-pi,pi].由于输入输出都采用 13 位定点数，即输入进来的 13 位 I/Q 要先进行位数扩展，最后一次迭代得到的结果要进行截断。

2. 采用删除量化模式：即当小数位精度超过 10 位时，将多余的“尾巴”比特（LSB）直接舍弃。

设计要求

1. 给出芯片详细的划分方式。（ Cordic 模块由哪几个基本的模块组成，每个模块的输入输出如何定义，每个模块的功能是什么）。

2. 给出 Cordic 单元的详细的实现方式。推荐采用流水线的方式。

3. Verilog 代码要有详细的注释。

4. 采用 SMIC 0.18um 工艺标准单元库实现设计，完成整个电路的 RTL 设计、逻辑综合、布局布线，并通过 DRC 和 LVS 检查。

假设：所有输入信号的 transition time 为 0.5ns， clock jitter<0.1ns，所有输出信号的负载小于 10pF。除时钟外的所有输入信号的 input_delay=2ns，输出信号的 output_delay=2ns。

指标说明

1　为了保证计算精度，要求迭代 13 次。

2　内核总面积（不包括 pad）< 800um x 800um。

3　时钟 CLK1 和 CLK2 都是由外部给入，但是 CLK1 的速率是 CLK2 的两倍。(例如：CLK1 是 960KHz，CLK2 是 480KHz)

评价与要求

1. 分组进行，每组 2~3 人；

2. 需要提交的文档：

a) RTL verilog 源代码

b) ICC 后的门级网表和 SDF 文件

c) ICC 和 DC 的约束报告(report_constraints –v)

d) 版图的 DRC 和 LVS 报告(calibre 的.summary 文件)

e)设计报告（包括电路结构、数据通路模块的实现方式、设计流程完成情况、主要指标、总结，不超过 4 页 word 或 8 页 ppt）

3. 评分标准： 根据设计流程完成的情况，电路性能和报告情况三个方面综合评分。

Cordic 算法简要说明(具体说明见参考文献[1-3])

Cordic 算法，即坐标旋转数字计算方法（Coordinate Rotation Digital Computer），是实现笛卡尔坐标体系向极坐标体系转变的一种数值逼近方法，其基本原理为笛卡尔坐标的平面变换。将笛卡尔坐标体系中任意一个点(x_1, y_1)旋转 θ 角度到点(x_2, y_2)，如图 5（a）所示，可以利用如下的数学方程表示

$$\begin{cases} x_2 = x_1 \cdot cos\theta - y_1 \cdot sin\theta = cos\theta \cdot (x_1 - y_1 \cdot tan\theta) \\ y_2 = x_1 \cdot sin\theta + y_1 \cdot cos\theta = cos\theta \cdot (y_1 + x_1 \cdot tan\theta) \end{cases}$$

若去除cosθ项，则得到伪旋转方程式

$$\begin{cases} \widehat{x_2} = x_1 - y_1 \cdot tan\theta \\ \widehat{y_2} = y_1 + x_1 \cdot tan\theta \end{cases}$$

即旋转的角度是正确的，但是 x 与 y 的值均增大了$\cos^{-1}\theta$倍，模变大。图 5(b)伪旋转的示意图。

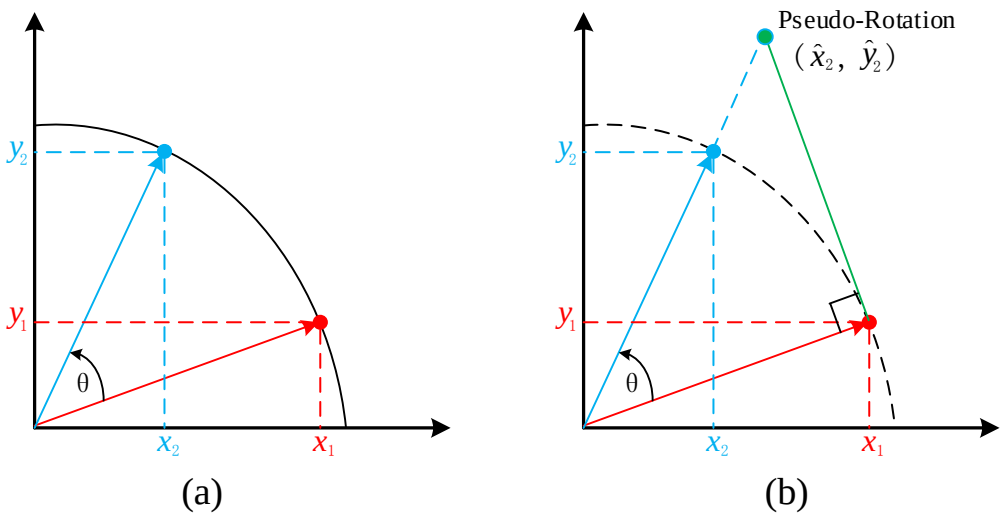

图 5 笛卡尔坐标系中的坐标旋转

(a) 旋转 (b) 伪旋转

Cordic 算法利用确定的角度 θ 实现（伪）旋转，即

$$tan\theta = 2^{-i} \quad i = 0,1,2,\cdots$$

从而有

$$\begin{cases} \widehat{x_{i+1}} = x_i - y_i \cdot 2^{-i} \\ \widehat{y_{i+1}} = y_i + x_i \cdot 2^{-i} \end{cases}$$

显然2^{-i}的乘法运算可以用移位来实现，这样借助伪旋转公式利用加法和移位操作就可以实现角度的旋转。表 2 给出每次迭代(i)的角度变化。

很显然，每次旋转的方向都影响到最终要旋转的累积角度。由于

$$\sum_i \theta = \sum_i tan^{-1}2^{-i} = 99.7°$$

因此在$-99.7° \le \theta \le 99.7°$的范围内的任意角度都可以旋转。对于该范围之外的角度，可使用三角恒等式转化成该范围内的角度。这样，笛卡尔坐标系中任意一个坐标点都可以利用该方法实现旋转。

表 2 Cordic 算法迭代角度值

迭代次数	$tan\theta = 2^{-i}$	角度 ϑ	模系数($cos\theta$)
1	1	45.000000	0.707107
2	0.5	26.565052	0.894427
3	0.25	14.036243	0.970143
4	0.125	7.125016	0.992278
5	0.0625	3.576334	0.998053
6	0.03125	1.789106	0.999512
7	0.15625	0.895174	0.999878
8	0.0078125	0.447614	0.999969
9	0.00390625	0.223811	0.999992

10	0.001953125	0.111906	0.999998
11	0.000976563	0.055953	0.9999995
12	0.000488281	0.027976	0.9999999
13	0.000244141	0.013988	≈1

伪旋转方程的实现中，坐标点的模（幅度）发生了变化，因此需要对模进行校正。定义伸缩因子

$$\prod_{i}^{13} cos\theta_i \approx \prod_{i=1}^{13} cos\theta_i = cos45° \cdot cos26.565° \cdot \cdots \cdot cos0.0002° = 0.607252941$$

因此只需要将最后旋转结束后的结果乘以伸缩因子就可以得到实际的模值。

利用 Cordic 算法实现笛卡尔坐标向极坐标的转化的基本思想是以笛卡尔坐标给定的位置为初始点，利用 Cordic 算法将其不断的进行旋转，以逼近 X 轴，同时修正坐标点的模，当旋转到 X 轴时（y 分量=0），累计的旋转角度就是对应的极坐标相角，X 分量就是极坐标系中的幅度。

参考文献

[1] R. Andraka. A survey of CORDIC algorithms for FPGA based computers.

[2]http://www.uio.no/studier/emner/matnat/ifi/INF5430/v12/undervisningsmateriale/dirk/Lecture_cordic.pdf

[3] http://web.cs.ucla.edu/digital_arithmetic/files/ch11.pdf