# 环形振荡器工程描述

环振工程由两部分组成，其中硬件编程部分（PL）分为环阵矩阵及其控制电路和片上频率测量电路，软件部分（PS）接收测量数据并发送给PC端。

## 硬件部分：

### 环阵部分：

环形振荡器由奇数个非门首尾相连而成，即前一非门的输出为后一非门的输入，最后一个非门的输出为第一个非门的输入。由图一所示，由于环路的闭回路增益为负一，这种结构本身会输出周期性方波，其频率由环路延迟决定（公式一）。因此我们可以用这种结构测量fpga延迟。

（1）

****

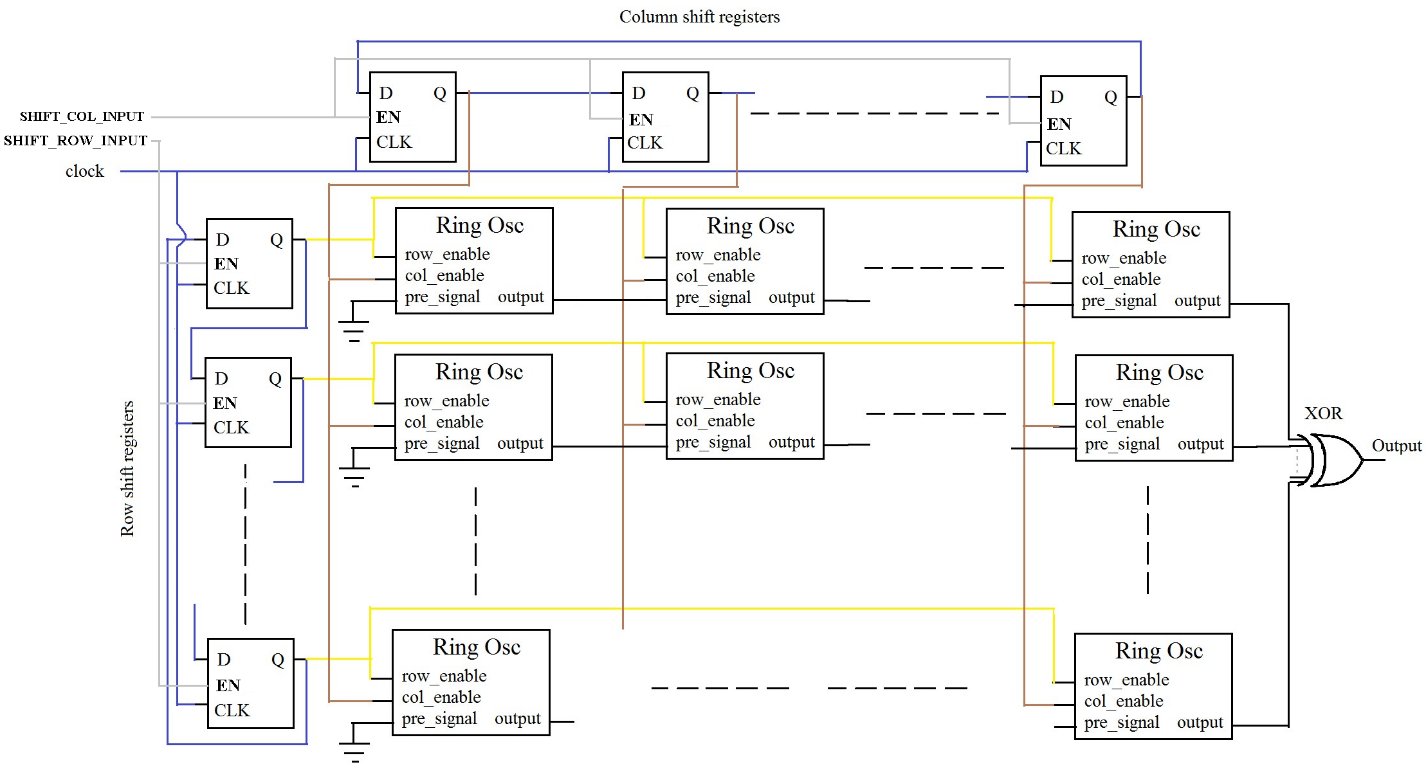
图一 环阵结构



图二 设计的单个环阵单元（九非门环阵）

我们将完全一致的环阵尽可能多的铺满整个电路。每个环阵为三输入（行使能、列使能和前一电路输出）一输出结构。其中为使得环阵的连线完全一致，我们需要在约束文件(xdc文件)约束每一条线路（见附录A）；为减少测量时电路的功率，我们每次仅使能一个环阵，即第九个非门事实上为三输入一输出结构:当控制电路控制行使能和列使能为有效时此结构等效为非门，否则则输出低电平；为使电路只有一个输出，每个环阵的输出与同一行前一环阵的输出异或，同时每行的末端输出进行异或，由此终端输出即为被使能的环阵的输出。

环阵的使能由移位寄存器控制。行、列移位寄存器初始化后有且仅有一个‘1’在最低位，在外部输入时钟上升沿时，若外部控制行列信号（SHIFT\_ROW\_INPUT,SHIFT\_COL\_INPUT）为‘1’则移位‘1’到临位。整体结构如图三所示。

图三 环阵矩阵及移位寄存器整体结构

### 频率测量、储存及输出部分：

频率测量电路的设计思路是：在一定已知时间内计环阵电路输出方波上升沿的次数（M计次），由次数和已知的时间计算其频率。其中已知时间为预设定个数（N周期）个系统周期（f系统）。其计算公式为：

（2）

整体测量系统由状态机实现，整体流程如下图所示。即在收到开始信号后，从第一个环阵开始顺次使能并计数、储存数据、停止当前环阵并使能下一环阵。在全部测量完成后，发送指定信号告知arm内核可以开始进行数据读取。



图四 片上频率测量系统流程图

### 工程硬件部分的整体描述

整体工程输入输出端口如下表所示。

表格 1 整个环阵及其测量系统的输入输出端口

|  |  |  |
| --- | --- | --- |
| 端口 | 性质 | 功能/备注 |
| CLK | 输入 | 50MH，由zynq7 processing system PLL输出 |
| Rst | 输入 | 复位所有测试信号，低电平有效 |
| Test\_en[7:0] | 输入 | 控制启动、终止信号：连续接收到0x53、0x54 时启动全局环振测 |
| P\_r[31:] | 输入 | 设定测试计数周期个数，即单个环振计时最大值 |
| R\_ADDR[13：0] | 输入 | 读计数结果地址 |
| T\_finish | 输出 | 全局测试结束：输出‘1’收到0x55后变为 ‘0’ |
| R\_DATA[31：0] | 输出 | 计数结果 |
| 备注：封装在AXI4内，与zynq7 processing system连接构成硬件系统 | | |

整体工程包含六个HDL文件，具体功能描述如下表所示

表格 2 环阵工程所用文件及文件功能

|  |  |
| --- | --- |
| 文件名 | 功能 |
| cmcs\_ro.vhd | 头文件，连接各模块 |
| counter.vhd | 计数器，在被使能时遇输入时钟上升沿计数个数加一，输出计数值 |
| ro\_test.vhd/  mixed\_ro.vhd | 九非门环形振荡器，用RLOC和BEL将单个环阵固定在五个相邻的CLB上其中前一文件将环阵仅固定在M或L slice上，后一文件为混M和L slice。 |
| main\_ro.vhd | 环阵矩阵及移位器，用RLOC将每个环阵固定在相应的位置上 |
| control\_test.vhd | 状态机，进行整体流程控制，复位信号有效则在下一周期复位所有信号 |
| bram\_sp.vhd | 数据存贮在片上BRAM中，在一个周期内不能同时进行写和读的操作 |
| ring\_osc\_v1\_0\_  S00\_AXI.v | 环阵工程以AXI4封装，修改的AXI4文件，配置相应端口的输入信号 |
| fixed\_route.xdc | 固定环阵及移位器，与lock\_pins(m).xdc一起固定所有环阵。 |
| lock\_pins.xdc/  lock\_pinsm.xdc | 固定每个环阵所用的LUT及所用LUT管脚，与fixed\_route.xdc 一起固定所有环阵。其中含m的文件用来配置混M和L slice的环阵。 |
| bramloc.xdc | 大体上限制其他器件的摆放位置 |
| system\_loc.xdc |
| 备注：1. 如遇到工程自动删除某些器件，应使用DONT\_TOUCH使相应的signal和component不被优化  2. 环阵的位置由RLOC确定相对位置，在xdc文件中用LOC声明第一个环阵的确定位置，即固定了所有环阵所在SLICE。将所有环阵声明在同一个u\_set内，即可以用RLOC配置他们的相对位置  3. lock\_pins(m).xdc 需要配置到相应文件，即：scoped\_to\_ref 值为ring\_osc(m)，fixed\_route.xdc的相应值为ro\_array，bramloc.xdc的相应值为cmcs\_ro，system\_loc.xdc为整个工程所以无此值。详细配置方法见附录C。  4. main\_ro.vhd中有生成环阵矩阵用了多个generate，即分成了多块生成，因为不同块RLOC值不同。  5. 在生成比特流文件时需要添加loop\_warn.tcl文件将环阵的loop error变为warning。  6. 其中现有已生成的环阵矩阵为一27行26列环阵矩阵，每个环阵为九非门环阵并占据相邻的五个CLB。其中第5、7、13、15、17、19、24、25列环阵完全在L slice上，第1-4，21-23列为混合M和L slice的环阵，其余列环阵完全在M slice上。在xc7z030fbg484-2上vivado对M和L slice给出的线路延迟基本一致，所以仅在M或L slice上的环阵应该有相似的频率。同时，第5、14、23行由于跨过相对较长的布线，应当比其他行相对频率慢一些。 | |

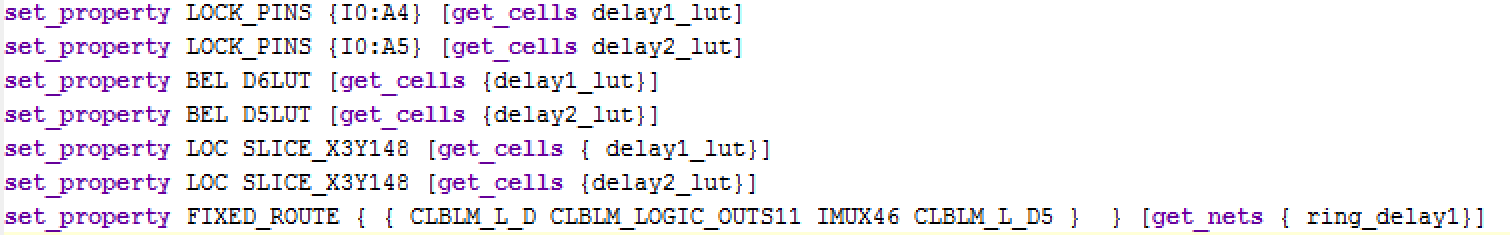
## 软件部分：

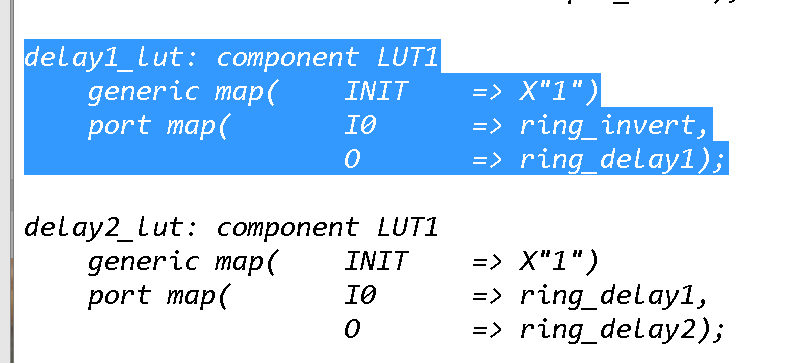
编写C语言程序在ARM内核内启动测试、读取测试数据并传输到电脑COM口。在电脑端用MATLAB读取并处理数据。所用文件如下表

|  |  |
| --- | --- |
| 文件名 | 功能 |
| testperiph.c | 写入ARM核的C语言程序。向硬件部分发送开始信号。当接到完成信号后，发送地址以读回存在bram里的数据，最后通过串口传输数据回电脑。重复多次以上过程以获得同一位置更多的数据 |
| zynq.m | 电脑端Matlab程序。打开相应COM口，设置COM口的速度、缓存、超时时间和停止符，接收数据到string变量中，最后处理数据。（公式二） |

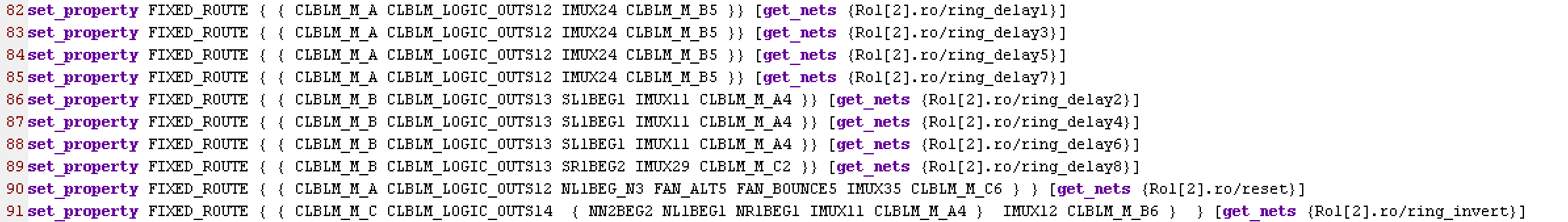
## 附录A 详细固定环阵方法解释：

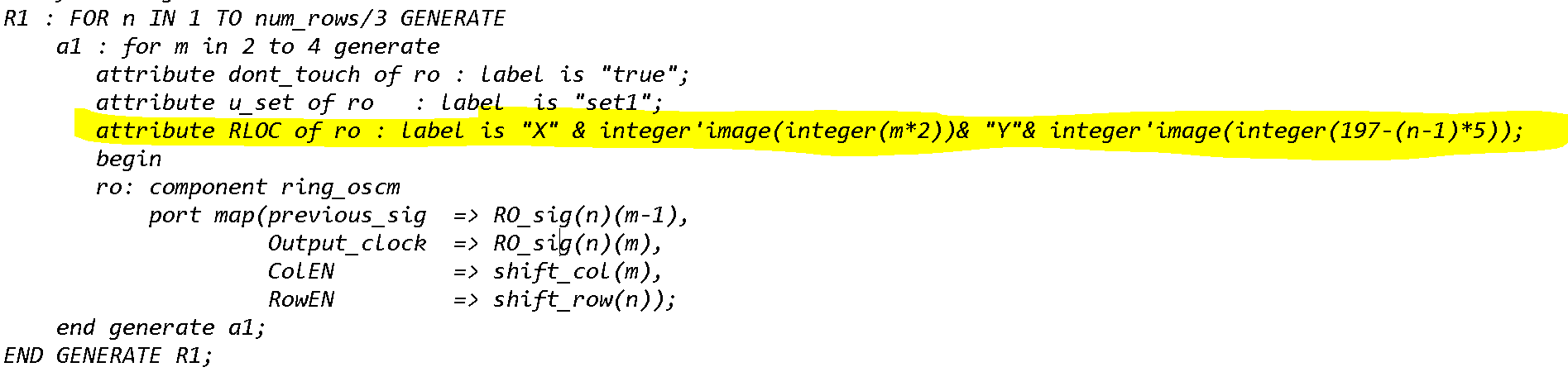
为使得环阵线路固定，以固定第一非门与第二非门间连线为例，需要声明需连线的两结构所在的具体SLICE(即图中LOC性质)、具体当片SLICE上的指定LUT（即图中BEL性质）、连线中后一结构使用LUT的哪一输入管脚（即图中LOCK\_PINS性质）以及这条连线具体经过哪些节点（即图中FIXED\_ROUTE性质）。下图示例为固定单一环阵所有需要的约束。



左图为生成环阵文件中第一第二非门的生成方式及名称，对应上图get\_cells中的名称。

实际上，lock\_pins 和BEL 用lock\_pins(m).xdc文件固定下来，约束文件直接应用于所有环阵。环阵的具体位置由main\_ro.vhd中RLOC和fixed\_route.xdc中LOC固定，其中前一文件生成所有相对位置，而后一文件声明第一个环阵的具体位置。最后，所有的环阵连线都在fixed\_route.xdc文件中用fixed\_route固定。下图中示例固定单个环阵的所有连线。



左图为在生成环阵时（main\_ro.vhd文件）固定其相对位置。

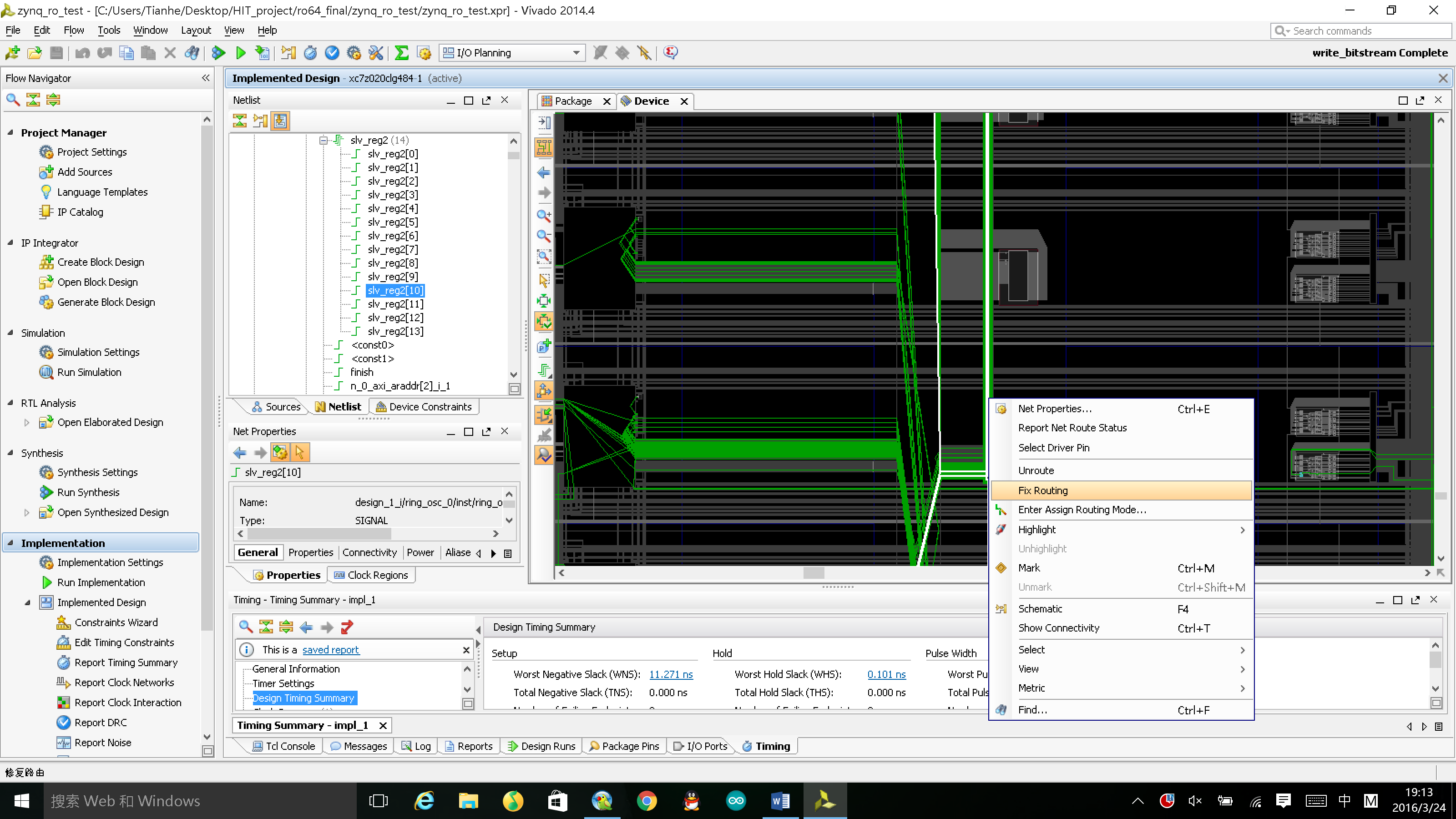
## 附录B 约束文件生成的操作过程：

由于每条连线都需要约束并保持相同，我们需要找到一组可用的固定连线然后使用在所有环阵上。具体方法如下：

首先生成一个2\*3的小环阵矩阵，其中需要改环阵所用的具体非门个数、非门位置（即ro\_test.vhd文件），还需要改 main\_ro.vhd文件减少生成环阵的个数到2\*3个，其中环阵的相对位置在这里约束（即RLOC）。

然后在xdc文件中配置第一个环阵所在具体位置，让vivado在仅有位置约束下自动布线。

在implementation界面看到合适的约束后将线固定方法如下图所示：

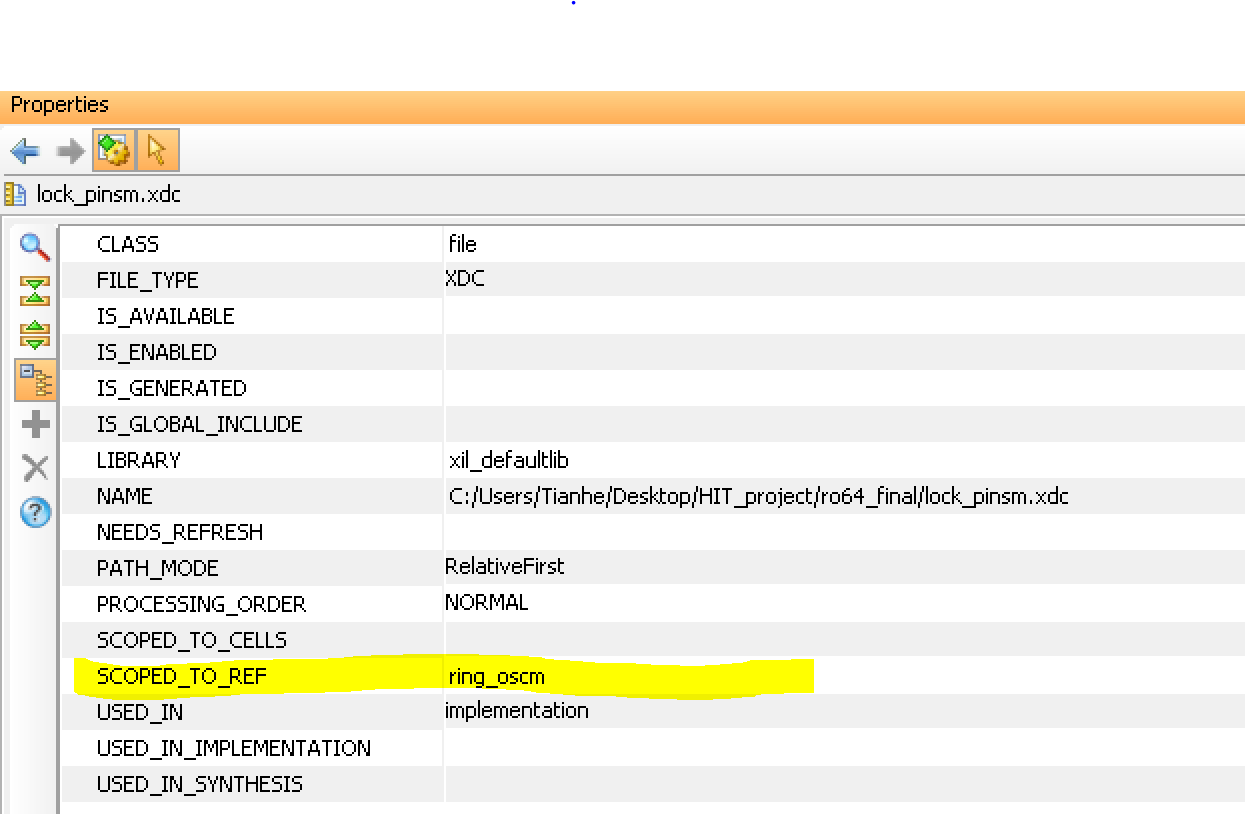


固定所有环阵必须的连线后（在现有工程下为ring\_delay1到ring\_delay8及ring\_invert）ctrl+s保存到xdc文件内，用python程序复制多次这组连线给所有环阵（即fixed\_route性质）。同时需要记录下每个器件所用管脚（即lock\_pins性质）。将管脚信息和使用的LUT信息写入lock\_pins.xdc文件（固定连线必须在xdc文件中写BEL、LOCK\_PINS和LOC性质，虽然我们已经在ro\_test.vhd文件中声明过一次BEL信息）。

这里繁琐的地方在于生成所有连线。由于FPGA有M和L slice，且连线时相同节点有两种不同的名称（如IMUX\_L24和IMUX24）所以在python生成所有连线时需要对不同的名称分组生成，详情参考python文件mixed\_ro.py。

## 附录C：

其中将xdc文件使用在指定的模块上（scoped\_to\_ref性质配置）如下图所示：



其中图中为配置lock\_pinsm.xdc文件到ring\_oscm模块上