嵌入式系统结构与设计课程实验

实验报告 1

二输入逻辑门的设计与实现

邱迪聪 / 11331262

2013/10/6

目录

一、实验目的

这个实验将指导实验者通过使用 ISE 软件进行简单的 2 输入逻辑门的设计与实现。

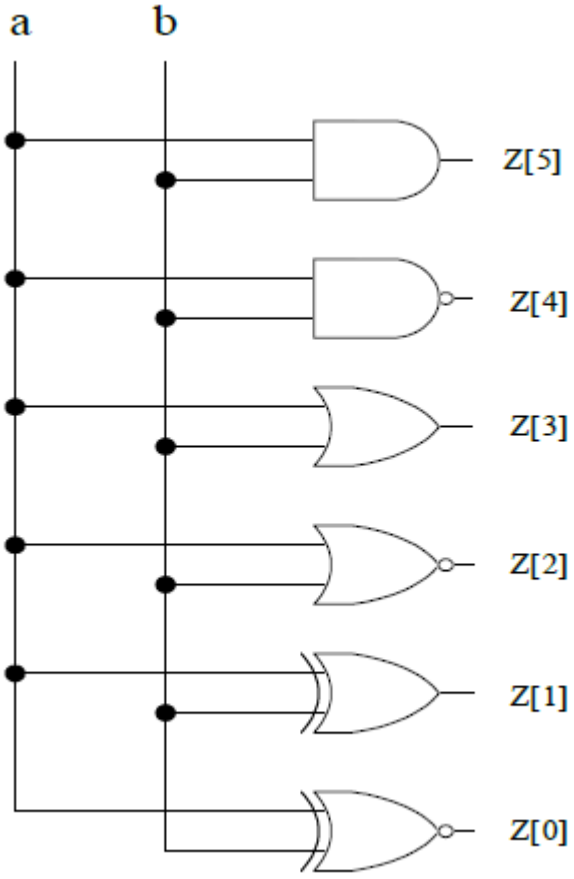
实验目的有如下两个：

- 使用 ISE 软件设计并仿真。
- 学会程序下载。

二、实验内容及结果

1、使用 Verilog 语句描述逻辑电路

如下电路图为二输入逻辑门电路：



在本实验中，需要使用 Verilog 语句对该电路进行描述。

新建好项目后，首先要对模块的输入和输出进行定义。从该电路图可见，输入为 a 和 b 两个输入端，

而输出则为 z[5] – z[0] 六个输出端。故模块的参数定义如下：

```
module gates2( input wire a, b, output wire[5:0] z );
```

表示输入有两个端口 a 和 b，而输出则为一个数组 z[5:0]。

根据逻辑电路的知识，以及 Verilog 语言的定义，可以把电路转化为如下表达语句：

```
assign z[5] = a & b;
assign z[4] = ~(a & b);
assign z[3] = a | b;
assign z[2] = ~(a | b);
assign z[1] = (~a & b) | (a & ~b);
assign z[0] = (a & b) | (~a & ~b);
```

到了这里，主模块的定义已经完成。完整代码见附录 3。

2、电路运行仿真及验证

对上一步得到的模块进行仿真模块的建立，ISE 系统会自动生成对应的接口代码，而实验者只须要添加仿真的具体运行代码即可。在

```
// Add stimulus here
```

这句注释之后加入如下仿真测试代码：

```
#200
a <= 0;
b <= 0;

#200
a <= 1;
b <= 0;

#200
a <= 0;
b <= 1;

#200
a <= 1;
b <= 1;
```

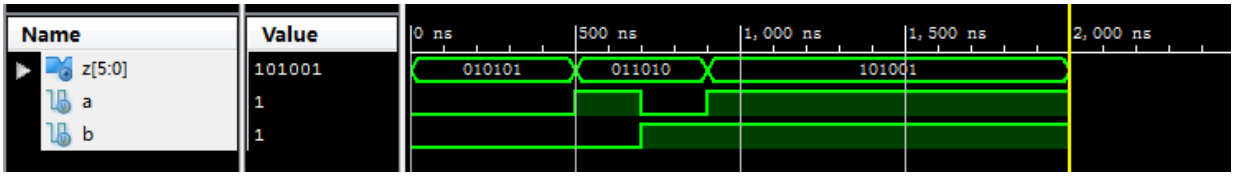
该段仿真代码在每个测试样例之前都有 200 毫秒的延迟，以显示运行结果。该测试对 a 与 b 端口的所有输入情况都进行了测试，故是一个有充分完整性的测试。完整代码见附录 3。

预期得到的测试结果如下：

a	b	z[5]－z[0]
0	0	010101
1	0	011010
0	1	011010
1	1	101001

在编写好测试代码后，配置仿真程序的运行时间。由仿真代码可见，仿真过程起码需要 900 毫秒的时间，为了显示更美观，故配置仿真的时间长度为 2000 毫秒。

运行方针程序后，波形如下：



仿真结果正如预期，故对应的主模块代码编写正确。测试通过。

3、综合、实现和程序生成

在项目管理中新建一个 Implementation Constraints File 进行管脚约束。

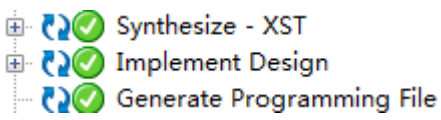
为了操作与观察方便，故本次实验使用了开发板的 T5 与 U8 开关作为输入，使用 T11、R11、N11、M11、V15 和 U15 作为输出，且分别对应逻辑电路图的 a、b 及 z[5]－z[0]。

管脚约束代码如下：

```
NET "a" LOC = "T5";
NET "b" LOC = "V8";
NET "z[5]" LOC = "T11";
NET "z[4]" LOC = "R11";
NET "z[3]" LOC = "N11";
NET "z[2]" LOC = "M11";
NET "z[1]" LOC = "V15";
NET "z[0]" LOC = "U15";
```

以上代码亦可见附录 3。

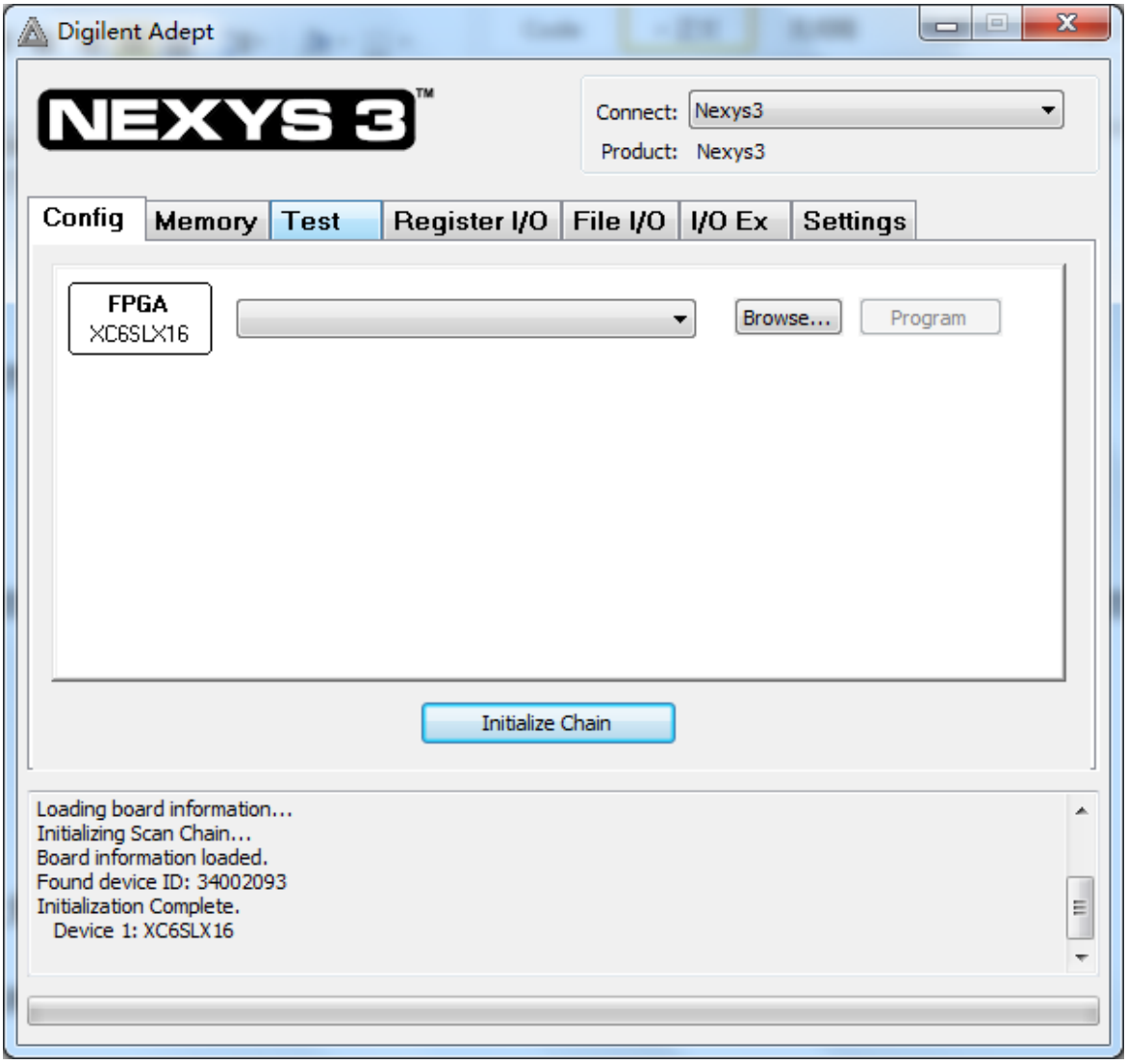
在编辑好管脚约束后，依次进行综合、实现和生成程序这三个步骤。这三个步骤执行的结果均成功，运行结果截图如下：



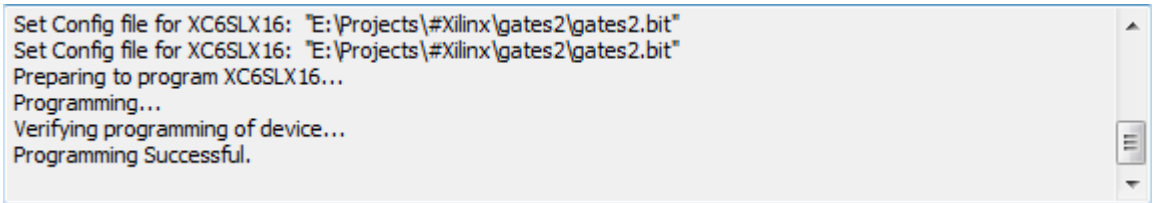
则在项目目录下生成了一个 .bit 文件，可提供程序上载。创建的 RTL 电路图见附录 2。

4、程序上载及真机测试

将开发板连接好电脑后打开开关，运行 Adept 程序，并选择对应的开发板选项，如下图：

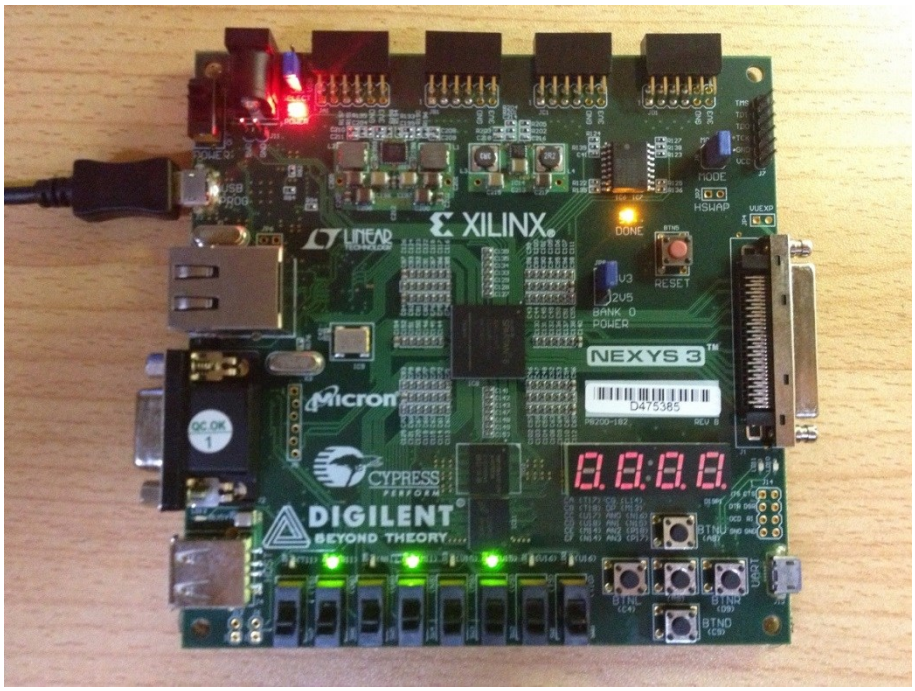


打开上一步生成的 .bit 程序文件，并进行上载。结果现实上载成功：



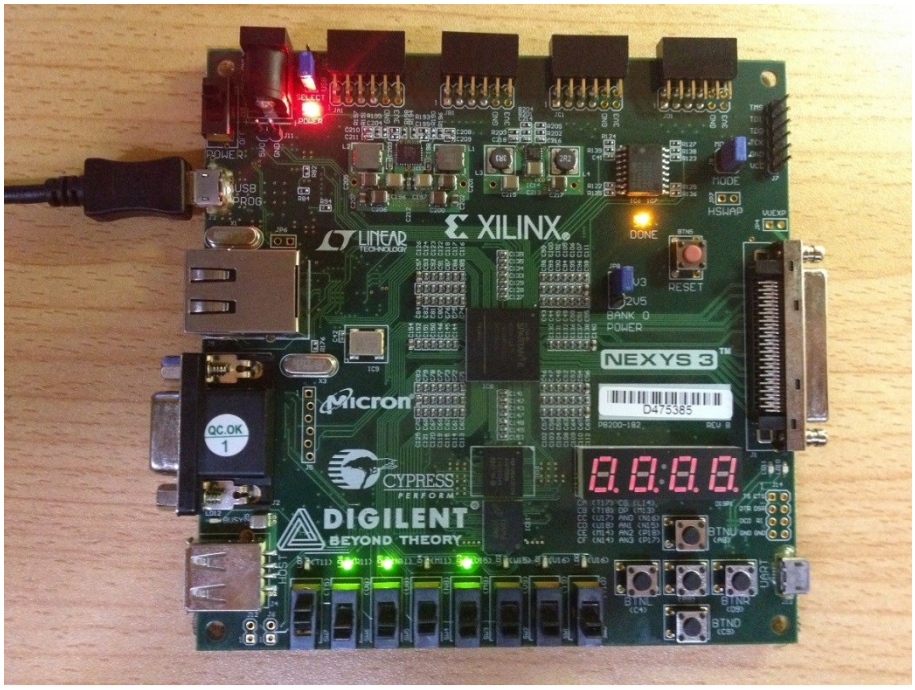
此时在开发板上分别对对应 a 与 b 输入端口的两个开关进行拨动，并观察开发板上六个作为输出的 LED 灯的变化情况。

测试 1：a = 0, b = 0



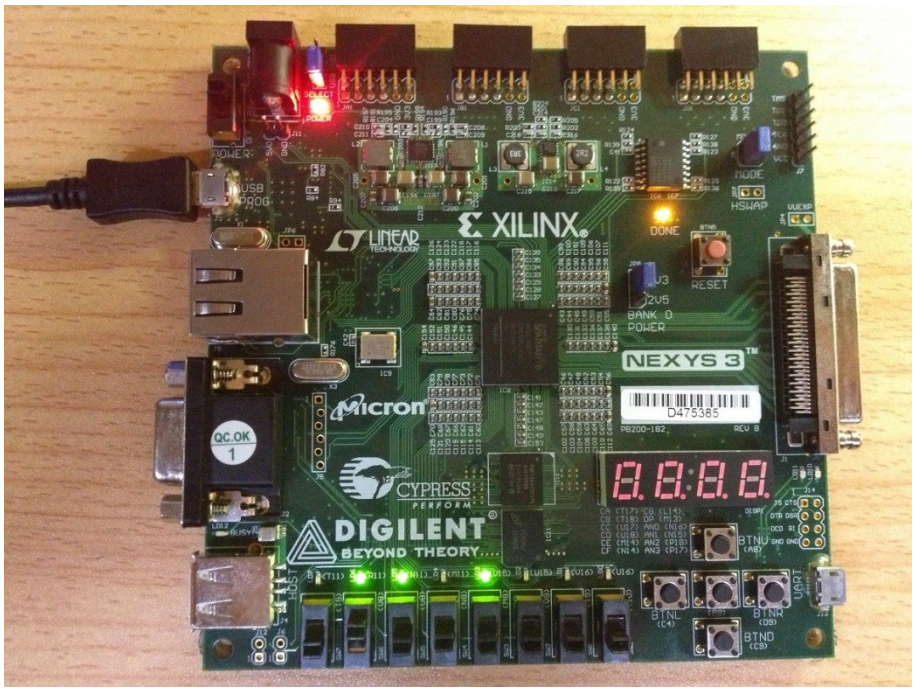
符合预期结果（010101）。

测试 1：a = 1, b = 0



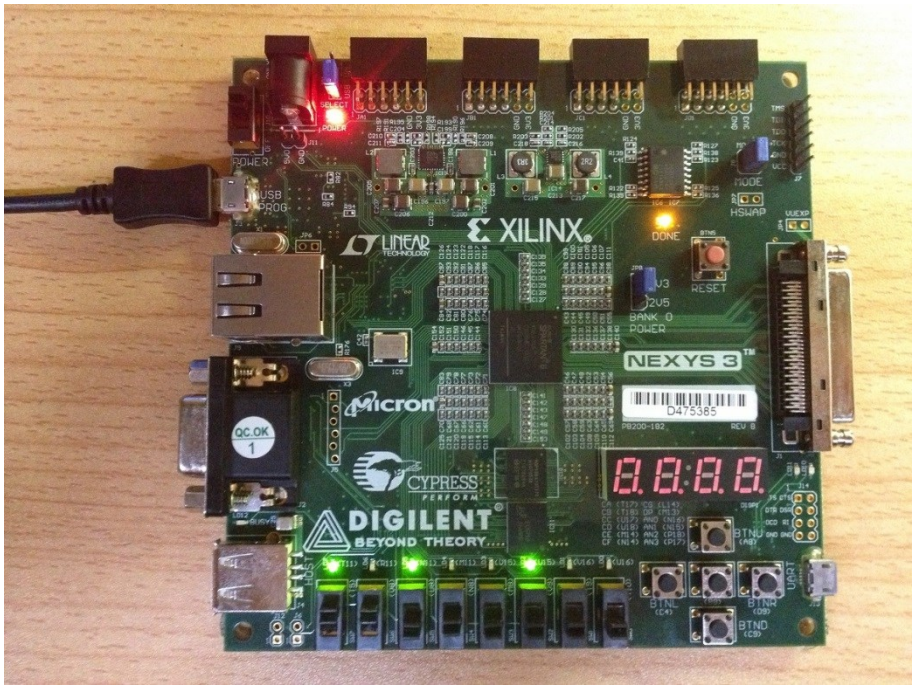
符合预期结果（011010）。

测试 1：a = 0, b = 1



符合预期结果（011010）。

测试 4：a = 1, b = 1



符合预期结果（101001）。

三、实验感想

很早之前就听说过，写在软件上的程序其实是不太可靠而且运行效率低下，因此有很多人把程序直接写到硬件里面。虽然知道这个方法可行，可是一想到如果要自己来编辑逻辑电路的话，估计是这辈子就这么没有了。然后也想过写个软件去化简逻辑表达式，并进行逻辑优化，可是之前做了个类似的软件，可是运行的效率非常低下，而且一想到这个是一个 NP 完全的问题，就萎缩了。

虽然知道必然有什么软件可以帮我们做这些事情，但是却一直不知道从何入手。之前在做一些简单的电路的时候，都是手动去布线的，而且即使电路再简单，布线也得花好几个小时，而且还没太多地考虑高频状态下的电磁干扰等情况。

在上了这次课后，发现有如此一个神器以及如此完善的系统，还居然只要简单地写些跟 C 语言非常像的语句之后点几个小按钮，就把整个电路的逻辑门路都自动化简并生成出来，实在是要赞叹前人付出的巨大努力。

有了这些工具，以后我们要做一些非常高效的、基于硬件实现的程序就变得异常简单了，而且在与这些底层硬件接触的过程，本身就是非常有趣以及好玩的一件事。实在要感谢我们的老师，给我们带来了这么好的一个乐趣之物，而且还是直接借给我们玩一个学期，如此良好的教学理念，绝对值得我们去认真学习一番！

邱迪聪

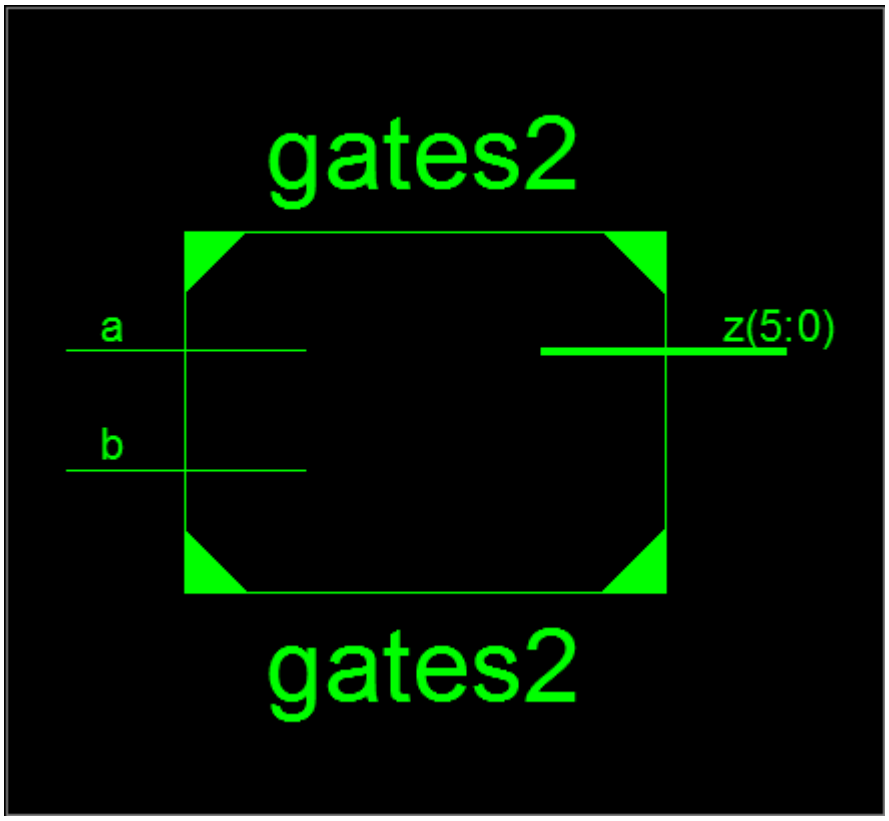
2013 年 10 月 6 日

附录 1：附件列表

本次实验包含三个附件：

- gates2.v （主模块代码文件）
- gates2_sim.v （仿真测试代码文件）
- gates2_ctr.ucf （管脚约束定义文件）

附录 2：RTL 电路图



附录 3：实验代码

1、二输入逻辑门 Verilog 表达代码

```
module gates2( input wire a, b,
                output wire[5:0] z );
   assign z[5] = a & b;
   assign z[4] = ~(a & b);
   assign z[3] = a | b;
   assign z[2] = ~(a | b);
   assign z[1] = (~a & b) | (a & ~b);
   assign z[0] = (a & b) | (~a & ~b);
endmodule
```

2、仿真测试代码

```
module gates2_test;

   // Inputs
   reg a;
   reg b;

   // Outputs
   wire [5:0] z;

   // Instantiate the Unit Under Test (UUT)
   gates2 uut ( .a(a), .b(b), .z(z) );

   initial begin
      // Initialize Inputs
      a = 0;
      b = 0;

      // Wait 100 ns for global reset to finish
      #100;

      // Add stimulus here
      #200
      a <= 0;
      b <= 0;
```

```
        #200
        a <= 1;
        b <= 0;

        #200
        a <= 0;
        b <= 1;

        #200
        a <= 1;
        b <= 1;

    end

endmodule
```

3、管脚约束代码

```
NET "a" LOC = "T5";
NET "b" LOC = "V8";
NET "z[5]" LOC = "T11";
NET "z[4]" LOC = "R11";
NET "z[3]" LOC = "N11";
NET "z[2]" LOC = "M11";
NET "z[1]" LOC = "V15";
NET "z[0]" LOC = "U15";
```