(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 103365976 A
(43)申请公布日 2013.10.23

(21)申请号 201310268650.X

(22)申请日 2013.06.28

(71)申请人 哈尔滨工业大学
地址 150001 黑龙江省哈尔滨市南岗区西大直街92号

(72)发明人 俞洋 刘旺 陈诚

(74)专利代理机构 哈尔滨市松花江专利商标事务所 23109
代理人 张宏威

(51)Int.Cl.
G06F 17/30(2006.01)
G06F 17/50(2006.01)

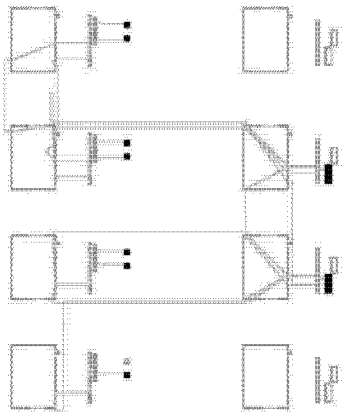
权利要求书2页 说明书8页 附图7页

(54)发明名称

使用Perl语言对电路XDL级网表描述进行面向应用的测试修改方法及测试方法

(57)摘要

使用Perl语言对电路XDL级网表描述进行面向应用的测试修改方法及测试方法，涉及一种对电路XDL级网表描述进行面向应用的测试修改方法及测试方法。它是为了解决现有对电路XDL级网表描述进行面向应用的测试修改的正确性和有效性差的问题。本发明使用Xilinx提供的XDL工具将NCD文件转换为XDL文件，然后使用适用于文本处理的Perl语言修改XDL文件，最后通过XDL工具将修改后的XDL文件转成NCD文件，获得最后的配置文件，完成对电路XDL级网表描述进行面向应用的测试修改。本发明适用于对电路XDL级网表描述进行面向应用的测试修改及测试。



CN 103365976 A

1. 使用 Perl 语言对电路 XDL 级网表描述进行面向应用的测试修改方法，其特征是：它的方法为：

使用 Xilinx 提供的 XDL 工具将 NCD 文件转换为 XDL 文件，然后使用适用于文本处理的 Perl 语言修改 XDL 文件，最后通过 XDL 工具将修改后的 XDL 文件转成 NCD 文件，获得最后的配置文件，完成对电路 XDL 级网表描述进行面向应用的测试修改。

2. 根据权利要求 1 所述的使用 Perl 语言对电路 XDL 级网表描述进行面向应用的测试修改方法，其特征在于使用 Perl 语言对电路 XDL 级网表描述进行面向应用的测试修改方法具体为：

步骤一、使用 ISE 将综合后网表 NCD 文件转换为可读的 XDL 文件；

步骤二、分析 XDL 文件并借助 FPGA Editor 工具提取出布局布线、CLB 配置的信息；

步骤三、根据布局布线的信息制作用户约束文件，即：UCF 文件，指定 CLB 的位置；

步骤四、对电路 XDL 级网表描述进行修改，获得最终用来测试的 NCD 文件。

3. 基于权利要求 1 所述的使用 Perl 语言对电路 XDL 级网表描述进行面向应用的测试修改方法的互连测试方法，其特征是：

将待测试电路中的 LUT 的配置分别修改成全与逻辑或全或逻辑，进而实现互连测试。

4. 根据权利要求 4 所述的使用 Perl 语言对电路 XDL 级网表描述进行面向应用的测试修改方法的互连测试方法，其特征在于所述互连测试方法具体为：

将待测试电路中的 LUT 的配置分别修改成全与逻辑，

使用 XDL 工具将修改后的 XDL 文件转换为 NCD 文件，作为互连测试中第一次测试配置的配置文件；

再将待测试电路中的 LUT 的配置分别修改成全或逻辑，

修改完所有的 LUT 后，转换成 NCD 文件，作为第二次测试配置的配置文件，进而对电路 XDL 级网表描述进行面向应用的互连测试。

5. 基于权利要求 1 所述的使用 Perl 语言对电路 XDL 级网表描述进行面向应用的测试修改方法的逻辑测试方法，其特征是：

在未使用的 CLB 中添加测试向量生成器和响应分析器，并将测试向量生成器的输出连接到待测 LUT 的输入端，将 LUT 的输出端连接到相应分析器上，进而对电路 XDL 级网表描述进行面向应用的测试修改方法的可编程逻辑资源测试。

6. 根据权利要求 5 所述的使用 Perl 语言对电路 XDL 级网表描述进行面向应用的测试修改方法的逻辑测试方法，其特征在于 LUT 为具有 n 个输入端，n 为正整数，对电路 XDL 级网表描述进行面向应用的测试修改方法的可编程逻辑资源测试的方法具体为：

采用一个 n 位的计数器作为测试向量生成器，使用 Verilog 硬件设计语言实现；

计数器的添加为在源代码中添加，具体为：

将用 Verilog 代码构建的模块添加到源代码中，并在原设计代码中增加一个时钟信号和一个复位信号供计数器使用；

然后进行编译综合，将计数器放置到未使用的 slice 中；将综合后的 NCD 文件转化为 XDL 文件，删除待测 LUT 的输入端的布线；

再然后通过创建连接关系，将计数器信号连接到 LUT 上，并修改 XDL 文件，创建计数器输出端与待测 LUT 输入端之间的连接；进而实现对电路 XDL 级网表描述进行面向应用的测

试修改方法的可编程逻辑资源测试。

使用Perl语言对电路XDL级网表描述进行面向应用的测试修改方法及测试方法

技术领域

[0001] 本发明涉及一种对电路XDL级网表描述进行面向应用的测试修改方法及测试方法。

背景技术

[0002] 一般来说，一个FPGA设计的基本流程分为如下几步，如图1所示：在上面几个步骤中，通过综合工具将HDL语言，原理图等设计输入翻译成由与、或、非等逻辑门和RAM、触发器等基本逻辑单元组成的逻辑连接（网表），并根据目标和要求优化所生成的逻辑连接，从而生成EDF文件（EDA工业标准文件）。然后进行功能仿真，验证设计的功能，在满足功能之后使用实现（Implement）工具，将综合输出的逻辑网表翻译成所选器件的底层模块的硬件原语，将设计映射到器件结构上，进行布局布线，达到在选定器件上实现设计的目的。此后通过时序验证使设计达到要求的时序约束，最后生成相应的可以下载到FPGA中的下载编程文件，实现芯片编程。

发明内容

[0003] 本发明是为了解决现有对电路XDL级网表描述进行面向应用的测试修改的正确性和有效性差的问题，从而提供一种使用Perl语言对电路XDL级网表描述进行面向应用的测试修改方法。

[0004] 使用Perl语言对电路XDL级网表描述进行面向应用的测试修改方法，其特征是：它的方法为：

[0005] 使用Xilinx提供的XDL工具将NCD文件转换为XDL文件，然后使用适用于文本处理的Perl语言修改XDL文件，最后通过XDL工具将修改后的XDL文件转成NCD文件，获得最后的配置文件，完成对电路XDL级网表描述进行面向应用的测试修改。

[0006] XDL文件是文本格式的可读文件。

[0007] 使用Perl语言对电路XDL级网表描述进行面向应用的测试修改方法具体为：

[0008] 步骤一、使用ISE将综合后网表NCD文件转换为可读的XDL文件；

[0009] 步骤二、分析XDL文件并借助FPGAEditor工具提取出布局布线、CLB配置的信息；

[0010] 步骤三、根据布局布线的信息制作用户约束文件，即：UCF文件，指定CLB的位置；

[0011] 步骤四、对电路XDL级网表描述进行修改，获得最终用来测试的NCD文件。

[0012] 基于上述方法所述的使用Perl语言对电路XDL级网表描述进行面向应用的测试修改方法的互连测试方法，

[0013] 将待测试电路中的LUT的配置分别修改成全与逻辑或全或逻辑，进而实现互连测试。

[0014] 所述互连测试方法具体为：

[0015] 将待测试电路中的LUT的配置分别修改成全与逻辑，

[0016] 使用 XDL 工具将修改后的 XDL 文件转换为 NCD 文件，作为互连测试中第一次测试配置的配置文件；

[0017] 再将待测试电路中的 LUT 的配置分别修改成全或逻辑，

[0018] 修改完所有的 LUT 后，转换成 NCD 文件，作为第二次测试配置的配置文件，进而对电路 XDL 级网表描述进行面向应用的互连测试。

[0019] 基于使用 Perl 语言对电路 XDL 级网表描述进行面向应用的测试修改方法的逻辑测试方法，

[0020] 在未使用的 CLB 中添加测试向量生成器和响应分析器，并将测试向量生成器的输出连接到待测 LUT 的输入端，将 LUT 的输出端连接到相应分析器上，进而对电路 XDL 级网表描述进行面向应用的测试修改方法的可编程逻辑资源测试。

[0021] LUT 为具有 n 个输入端，n 为正整数，对电路 XDL 级网表描述进行面向应用的测试修改方法的可编程逻辑资源测试的方法具体为：

[0022] 采用一个 n 位的计数器作为测试向量生成器，使用 Verilog 硬件设计语言实现；

[0023] 计数器的添加为在源代码中添加，具体为：

[0024] 将用 Verilog 代码构建的模块添加到源代码中，并在原设计代码中增加一个时钟信号和一个复位信号供计数器使用；

[0025] 然后进行编译综合，将计数器放置到未使用的 slice 中；将综合后的 NCD 文件转化为 XDL 文件，删除待测 LUT 的输入端的布线；

[0026] 再然后通过创建连接关系，将计数器信号连接到 LUT 上，并修改 XDL 文件，创建计数器输出端与待测 LUT 输入端之间的连接；进而实现对电路 XDL 级网表描述进行面向应用的测试修改方法的可编程逻辑资源测试。

[0027] 本发明提高了现有对电路 XDL 级网表描述进行面向应用的测试修改的正确性和有效性。

附图说明

[0028] 图 1 是本发明背景技术中所述的 FPGA 设计的基本流程图；

[0029] 图 2 是本发明所述的自动 PAR 的原理示意图；

[0030] 图 3 是使用 UCF 指定 slice 位置的原理示意图；

[0031] 图 4 是添加计数器的原理示意图；

[0032] 图 5 是删除 LUT 输入端布线的原理示意图；

[0033] 图 6 是计数器输出与 LUT 输入之间创建连接原理示意图；

[0034] 图 7 是计数器输出与 LUT 输入之间布线原理示意图；

[0035] 图 8 是仿真电路示意图；

[0036] 图 9 是故障注入仿真中的原电路输出结果；

[0037] 图 10 是故障注入仿真中的 net4 注入固定 0 型故障输出结果；

[0038] 图 11 是故障注入仿真中的 net4 注入固定 1 型故障输出结果；

[0039] 图 12 互连测试配置的仿真中的全与配置后的输出结果；

[0040] 图 13 互连测试配置的仿真中的全或配置后的输出结果；

[0041] 图 14 互连线上故障检测中的检测固定“0”型故障输出结果；

[0042] 图 15 互连线上故障检测中的检测固定“1”型故障输出结果；

[0043] 图 16 可编程逻辑测试中的故障注入前输出结果；

[0044] 图 17 可编程逻辑测试中的故障注入后输出结果。

具体实施方式

[0045] 具体实施方式一、使用 Perl 语言对电路 XDL 级网表描述进行面向应用的测试修改方法，它的方法为：

[0046] 使用 Xilinx 提供的 XDL 工具将 NCD 文件转换为 XDL 文件，然后使用适用于文本处理的 Perl 语言修改 XDL 文件，最后通过 XDL 工具将修改后的 XDL 文件转成 NCD 文件，获得最后的配置文件，完成对电路 XDL 级网表描述进行面向应用的测试修改。

[0047] 具体实施方式二、本具体实施方式与具体实施方式一所述的使用 Perl 语言对电路 XDL 级网表描述进行面向应用的测试修改方法的区别在于，使用 Perl 语言对电路 XDL 级网表描述进行面向应用的测试修改方法具体为：

[0048] 步骤一、使用 ISE 将综合后网表 NCD 文件转换为可读的 XDL 文件；

[0049] 步骤二、分析 XDL 文件并借助 FPGA Editor 工具提取出布局布线、CLB 配置的信息；

[0050] 步骤三、根据布局布线的信息制作用户约束文件，即：UCF 文件，指定 CLB 的位置；

[0051] 步骤四、对电路 XDL 级网表描述进行修改，获得最终用来测试的 NCD 文件。

[0052] 由于测试不修改原设计文件，不改变原设计功能，所以为了实现面向应用的测试方法，所以只能对以上步骤中的实现步骤进行操作。实现主要分为三个步骤，翻译 (Translate) 逻辑网表、映射 (Map) 到器件单元和布局布线 (Place & Route)。通过比较目前两个主流 FPGA 厂商 Xilinx 公司和 Altera 公司的 FPGA 及其提供的工具，本发明选择 Xilinx 公司的 FPGA 作为研究对象，原因如下：

[0053] (1)Xilinx 的 FPGA 结构比 Altera 的 FPGA 结构更开放；

[0054] (2)Xilinx 的 FPGA 开发软件 ISE 可以提供更多的 FPGA 信息；

[0055] (3)Xilinx 的 ISE 软件允许使用者按照需要修改配置文件并提供相应工具；

[0056] (4) 大多数 FPGA 测试方法的研究平台都是 Xilinx 的 FPGA，具有可比性。

[0057] 在基于 ISE 的实现过程中，翻译的主要作用是将综合输出的逻辑网表翻译为 Xilinx 特定器件的底层结构和硬件原语，在翻译过程中，设计文件和约束文件将合并成 NGD（原始类型数据库）输出文件等文件，其中 NGD 文件包含了当前设计的全部逻辑描述。映射的主要作用是将设计映射到具体型号的器件上 (LUT、FF、Carry 等)，在映射过程中，由转换流程生成的 NGD 文件将被映射为目标器件的特定物理逻辑单元，并保持在 NCD 文件中，NCD 文件中包含了当前设计的物理映射信息，此外还会输出包含所有物理约束信息的 PCF（物理约束文件）等文件。而布局布线步骤会调用 Xilinx 布局布线器，根据用户约束和物理约束，对设计模块进行实际的布局，并根据设计连接，对布局后的模块进行布线，产生 FPGA 配置文件，布局布线过程通过读取当前设计的 NCD 文件，将映射后生成的物理逻辑单元在目标系统中放置和连线，并提取成相应的时间参数，输出文件包括 NCD 和 DLY（延时文件）等相关文件。

[0058] 在发明使用的测试方法实现中，主要就是通过修改 NCD 文件获得相应的测试配

置。使用 Xilinx 提供的 XDL 工具将 NCD 文件转换为 XDL 文件，此文件是文本格式的可读文件，使用的语法是 Xilinx design language。然后使用适用于文本处理的 Perl 语言按要求修改 XDL 文件，再通过 XDL 工具将修改后的 XDL 文件转成 NCD 文件，得到最后的配置文件。

[0059] 以一个简单的例子电路为每个步骤做说明，该电路为 5 个输入 (i1，i2，i3，i4，i5)，2 输出 (o1，o2) 的组合电路，实现的逻辑为

[0060] o1 = i1 & i2|i3

[0061] o2 = o1 & i4 &(! i5)

[0062] 电路使用 Verilog 语言描述，目标芯片型号为 XC5VFX100T，封装 FF1738。

[0063] 具体实施方式三、基于具体实施方式一所述的使用 Perl 语言对电路 XDL 级网表描述进行面向应用的测试修改方法的互连测试方法，

[0064] 将待测试电路中的 LUT 的配置分别修改成全与逻辑或全或逻辑，进而实现互连测试。

[0065] 本实施方式所述的内容是互连测试的实现。

[0066] 具体实施方式四、本具体实施方式与具体实施方式四所述的使用 Perl 语言对电路 XDL 级网表描述进行面向应用的测试修改方法的互连测试方法的区别在于，所述互连测试方法具体为：

[0067] 将待测试电路中的 LUT 的配置分别修改成全与逻辑，

[0068] 使用 XDL 工具将修改后的 XDL 文件转换为 NCD 文件，作为互连测试中第一次测试配置的配置文件；

[0069] 再将待测试电路中的 LUT 的配置分别修改成全或逻辑，

[0070] 修改完所有的 LUT 后，转换成 NCD 文件，作为第二次测试配置的配置文件，进而对电路 XDL 级网表描述进行面向应用的互连测试。

[0071] 对于例子电路，使用 ISE 综合并自动布局布线后的结果如图 2 所示。图中从左至右不同方框对应的 FPGA 中的实际资源分别为全局交叉开关、局部交叉开关、缓冲器、IO 口以及 slice(Xilinx 中的可编程逻辑资源)。其中，深色涂黑的方框表示的资源是设计中实际使用的硬件 FPGA 资源。图中的细线为实际 FPGA 中连接设计中使用的各功能器件的互连线，也就是此时需要测试的那些互连线。

[0072] 从图 2 中可以看出此电路使用了两个 LUT，分别存在于两个 slice 中，为了证明用户约束文件可以有效的约束 slice 的位置，使用 UCF 文件将这两个 slice 指定到现在位置的右上角的 slice，如图 3 所示。

[0073] 接着修改 LUT 的配置逻辑，实现互连的测试。在这里将对经过 UCF 文件指定后的电路进行操作，因为在实际应用设计中为了满足某些时序要求都会使用 UCF 文件对设计添加约束。使用 ISE 中的 XDL 工具，将布局布线 (PAR) 后的 NCD 文件转换成文本格式的 XDL 文件，在 XDL 文件中找到对应的 slice。例如在这个电路中使用的一个 slice，其在 XDL 文件中的描述为：

[0074]

```
inst "oLgc1_OBUF" "SLICEL", placed CLBLM_X41Y98 SLICE_X65Y98  ,
cfg " A5LUT::#OFF A6LUT::#OFF ACY0::#OFF AFF::#OFF AFFINIT::#OFF
      AFFMUX::#OFF AFFSR::#OFF AOUTMUX::#OFF AUSED::#OFF B5LUT::#OFF
      B6LUT::#OFF BCY0::#OFF BFF::#OFF BFFINIT::#OFF BFFMUX::#OFF
      BFFSR::#OFF BOUTMUX::#OFF BUSED::#OFF C5LUT::#OFF C6LUT::#OFF
      CCY0::#OFF CEUSED::#OFF CFF::#OFF CFFINIT::#OFF CFFMUX::#OFF
[0075]
      CFFSR::#OFF CLKINV::#OFF COUTMUX::#OFF COUTUSED::#OFF
      CUSED::#OFF D5LUT::#OFF D6LUT:oLgc11:#LUT:O6=((A2*A4)+A6)
      DCY0::#OFF DFF::#OFF DFFINIT::#OFF DFFMUX::#OFF DFFSR::#OFF
      DOUTMUX::#OFF DUSED::0 PRECYINIT::#OFF REVUSED::#OFF
      SRUSED::#OFF SYNC_ATTR::#OFF "
;
```

[0076] 为了将 LUT 的配置改为全与逻辑，将其中的语句

[0077] D6LUT:oLgc11:#LUT:O6 = ((A2*A4)+A6)

[0078] 修改为：

[0079] D6LUT:oLgc11:#LUT:O6 = ((A2*A4)*A6)

[0080] 同理将其他 LUT 的配置修改为全与逻辑。再次使用 XDL 工具将修改后的 XDL 文件转换为 NCD 文件，作为互连测试中第一次测试配置的配置文件。对于互连测试的第二次配置，需要将设计中使用的 LUT 的配置修改为全或逻辑，方法类似，对于上例，将语句

[0081] D6LUT:oLgc11:#LUT:O6 = ((A2*A4)+A6)

[0082] 修改为：

[0083] D6LUT:oLgc11:#LUT:O6 = ((A2+A4)+A6)

[0084] 修改完所有的 LUT 后，转换成 NCD 文件，作为第二次测试配置的配置文件。

[0085] 为了验证此修改方法的有效性，可以在 FPGA Editor 中打开修改后的 NCD 文件，通过属性观察窗来查看。通过以上操作，就可以完成互连的测试配置。

[0086] 具体实施方式五、基于具体实施方式一所述的使用 Perl 语言对电路 XDL 级网表描述进行面向应用的测试修改方法的逻辑测试方法，

[0087] 在未使用的 CLB 中添加测试向量生成器和响应分析器，并将测试向量生成器的输出连接到待测 LUT 的输入端，将 LUT 的输出端连接到相应分析器上，进而对电路 XDL 级网表描述进行面向应用的测试修改方法的可编程逻辑资源测试。

[0088] 本实施方式所述的可编程逻辑测试的实现。

[0089] 具体实施方式六、本具体实施方式与具体实施方式五所述的使用 Perl 语言对电路 XDL 级网表描述进行面向应用的测试修改方法的逻辑测试方法的区别在于，对电路 XDL 级网表描述进行面向应用的测试修改方法的可编程逻辑资源测试的方法具体为：

[0090] 采用一个 6 位的计数器作为测试向量生成器，使用 Verilog 硬件设计语言实现；

[0091] 计数器的添加为在源代码中添加，具体为：

[0092] 将用 Verilog 代码构建的模块添加到源代码中，并在原设计代码中增加一个时钟信号和一个复位信号供计数器使用；

[0093] 然后进行编译综合，将计数器放置到未使用的 slice 中；将综合后的 NCD 文件转化为 XDL 文件，删除待测 LUT 的输入端的布线；

[0094] 再然后通过创建连接关系，将计数器信号连接到 LUT 上，并修改 XDL 文件，创建计数器输出端与待测 LUT 输入端之间的连接；进而实现对电路 XDL 级网表描述进行面向应用的测试修改方法的可编程逻辑资源测试。

[0095] 为了测试可编程逻辑资源，在不改变原设计使用的 CLB 内部配置逻辑和位置的前提下，需要在设计中未使用的 CLB 中添加测试向量生成器和响应分析器，并将测试向量生成器的输出连接到待测 LUT 的输入端，将 LUT 的输出端连接到相应分析器上。这里只描述测试向量生成器的添加方法，响应分析器的添加方法类似。

[0096] 由于目标芯片的 LUT 有 6 个输入，所以使用一个 6 位的计数器作为测试向量生成器，使用 Verilog 硬件设计语言实现。计数器的添加有两种方法，在设计的源代码中添加计数器模块的形式添加和在布局布线后以 IP 核的形式添加。这里使用在源代码中添加的方法。将用 Verilog 代码构建的模块添加到源代码中，并在原设计代码中增加一个时钟信号和一个复位信号供计数器使用。注意在添加时需要给计数器模块添加 KEEP 约束，以保证在综合是不被优化掉。因为计数器模块只是在源代码中调用但并没有使用。

[0097] 然后编译综合，将计数器放置到未使用的 slice 中，如图 4 所示，图中左下角的两个 slice 为计数器的主体部分，时钟和复位信号在图中没有显示。

[0098] 然后将此综合后的 NCD 文件转为 XDL 文件，修改删除待测 LUT 的输入端的布线，结果如图 5 所示，与图 4 相比，右侧两个待测 LUT 的输入端与芯片输入引脚之间的布线均被删除，右边上侧的 LUT 输入端与右边下侧 LUT 输出端的布线也被删除。两个 LUT 都只剩下各自输出端与芯片输出引脚之间的连线。断开连接之后就具备了将计数器的输出信号添加到 LUT 输入端的前提条件。

[0099] 接着，为了实现将计数器信号连接到 LUT 上，首先得创建连接关系，然后实现布线。修改 XDL 文件，创建计数器输出端与待测 LUT 输入端之间的连接。图 6 展示了这一步的结果，图中的飞线表示两点之间有连接关系但还未布线。

[0100] 在 FPGA Editor 工具中使用自动布线功能完成上一步创建的连接之间的布线，得到用于可编程逻辑测试的配置文件，图 7 为布线之后的结果，从图中可以清楚的看到计数器的输出端已经通过互连资源连接到待测 LUT 的输入端。

[0101] 至此，用于整个测试的三个配置文件制作完成，包括用于互连第一次配置的全与逻辑配置文件，第二次配置的全或逻辑配置文件，用于可编程逻辑资源测试的配置文件。

[0102] 综上，通过上述步骤可以实现对 FPGA 中互连资源和可编程逻辑资源的面向应用的测试。在测试方法的实现过程中，当设计非常大时会使用相当大的 FPGA 资源，那么手动按要求修改 XDL 文件的工作量将相当巨大，因此在发明中，使用了 Perl 程序完成对 XDL 文件的修改。

[0103] 下面用另外一个例子电路，通过仿真的方式验证测试方法的正确性：

[0104] 在仿真中，使用的电路为 6 输入 2 输出，使用了 4 个 LUT 的组合电路。每个 LUT 实现的逻辑功能如下：

[0105] net1 = (iLgc1|iLgc4)^iLgc3

[0106] net2 = (iLgc4 & iLgc5)|iLgc6

[0107] net3 = net4^net2

[0108] net4 =！ net1

[0109] 其中 iLgc1 ～ iLgc6 为电路的输入，net1 ～ net4 为每个 LUT 的输出，net2 和 net3 分别作为整个电路的输出，电路结构如图 8 所示。

[0110] 互连测试的仿真验证及分析：

[0111] 本发明中使用的测试方法是针对互连线上的固定 0 型故障和固定 1 型故障进行测试的，那么为了验证测试方法的有效性，必须在电路中人为注入故障。在这里通过人为配置 LUT4 的逻辑，使 net4 分别一直保持“0”或“1”，即可实现故障的注入。

[0112] 使 net4 上出现固定 0 型故障时，将 LUT4 的配置逻辑人为修改为

[0113] net4 = net1 &(！ net1)

[0114] 这样无论 LUT4 的输入为何值，net4 的值都被固定在“0”上。同理在 net4 上注入固定 1 型故障的方法是人为将 LUT4 的逻辑配置为

[0115] net4 = net1|(！ net1)

[0116] 这样即可实现无论 LUT4 的输入为何值，net4 上的值都为“1”。

[0117] 图 9 中是给实验中所使用电路的 6 个输入端施加计数器信号后得到的仿真结果，以作为参考。图 10 是在 net4 注入固定 0 型故障后的电路输出图，电路的输入依旧是计数器信号，从图中可以看出 oLgc1 的结果和原电路相同，而输出端 oLgc2 的结果已和原电路不同，从原电路结构分析，输出 oLgc1 是从 net2 引出，输出 oLgc2 从 net3 引出，为 LUT3 的输出，而 LUT3 实现的逻辑功能即为 net2 和 net4 的异或逻辑，从图 10 中可以明显看出 oLgc1 与“0”的异或结果正是 oLgc2 的值，由此可以说明电路中 net4 上注入了固定 0 型故障。同理图 11 中在输入端依旧施加计数器信号，oLgc2 的值正为 oLgc1 与“1”的异或结果，由此可以说明电路中 net4 上注入了固定 1 型故障。

[0118] 图 12 至 13 给出的是将电路进行测试配置后在计数器信号下的输出结果。图 12 为测试互连线上的固定 0 型故障的全与逻辑配置，LUT 中实现的逻辑都为 LUT 输入信号的与。从原电路结构可以看出，输出 oLgc1 为输入 iLgc4 ～ iLgc6 的与，输出 oLgc2 为输入 iLgc1 ～ iLgc6 的与。图 12 的结果刚好与输入相对应，oLgc1 只在 iLgc4 ～ iLgc6 都为“1”的时候才为“1”，oLgc2 在 iLgc1 ～ iLgc6 都为“1”的时候才为“1”。图 13 的结果也与输入相对应，oLgc1 只在 iLgc4 ～ iLgc6 都为“0”的时候才为“0”，oLgc2 在 iLgc1 ～ iLgc6 都为“0”的时候才为“0”。图 12 至 13 验证了配置成全与和全或逻辑后结果的正确性。

[0119] 在 net4 上分别注入固定 0 型和固定 1 型故障，并分别使用相应的测试方法进行测试，图 14 中给出的是电路注入固定 0 型故障，使用全与配置，输入端施加 all-1 测试向量，从图中可以看出，在 200ns 处施加 all-1 测试向量后，输出端 oLgc1 输出为“1”，说明 oLgc1 的前端电路上没有发生故障，oLgc2 的输出为“0”，说明检测到电路存在固定 0 型故障，且此故障发生在与 oLgc2 有关的线路上。在图 15 中，电路注入固定 1 型故障，在电路的输入端施加 all-0 测试向量，从图中可以看出，200ns 处测试向量施加后，oLgc1 输出为“0”，说明

oLgc1 的前端电路正常没有固定 0 型故障，而 oLgc2 的输出为“1”，说明 oLgc2 的前端电路上发生了固定 0 型故障。在此电路中 net4 的位置位与 oLgc2 的前端，而与 oLgc1 没有影响到 oLgc1。

[0120] 可编程逻辑测试的仿真验证及分析：

[0121] 图 16 给出的是注入逻辑故障前的仿真结果，每个 LUT 的输入端加入的信号由计数器及仿真中使用的测试向量生成器提供。每个 LUT 的输出分别引到 FPGA 的 IO 上，对应关系为 LUT1 的输出为 oLgc3，LUT2 的输出为 oLgc1，LUT3 的输出为 oLgc2，LUT4 的输出为 oLgc4，以观察结果。仿真中，将原电路 LUT 的输出结果 (oLgc1 ～ oLgc4) 与仿真得到的 LUT 输出结果 (oLgc1_R ～ oLgc4_R) 进行比较，故障显示信号分别为 (fault1 ～ fault4)。由从图 16 中可以看出，没有注入故障的原电路 LUT 输出与仿真的 LUT 输出完全相同，所有的故障显示信号一直保持为“0”，即没有发生故障。

[0122] 在原电路使用的可编程逻辑资源上注入故障用于测试，仿真中，LUT3 中实现的逻辑原为异或逻辑，现在人为修改为或逻辑，即实现了可编程逻辑资源上的故障注入。每个 LUT 的输入端接上相应的激励生成器的输出信号，从图 17 中可以看出，oLgc2 和 oLgc2_R 在大约 2us 后出现不同，故障显示信号 fault2 同时也变成“1”，表明相应的 LUT3 有逻辑故障。由此可以仿真结果可以说明，此测试方法可以检测出 LUT 的逻辑故障。

[0123] 在此仿真中，直接将输出信号直接引到 FPGA 的 IO 上，这样构造响应分析器的好处是简单易实现，可以非常方便的与仿真结果进行比较并直接获得发生故障的 LUT 的位置。但是在实际中 IO 属于稀缺资源，所以需要构造内部的响应分析器，将正确的结果存储在 FPGA 内部，在 FPGA 内部进行比较，将比较的结果压缩后使用少量的 IO 输出。虽然在使用 IO 方面的代价降低，但是响应分析器的结构复杂，实现起来困难，而且会占用大量的存储器资源。综合权衡，在本次验证性仿真中，仿真的重点是验证第二章的面向应用的测试方法的正确性和有效性，响应分析器不是仿真的重点，而且仿真采用的电路简单，使用的 LUT 较少，IO 的数量可以满足，所以在仿真中采用直接将 LUT 的输出引出到 FPGA 的 IO 上，以方便实验的进行。

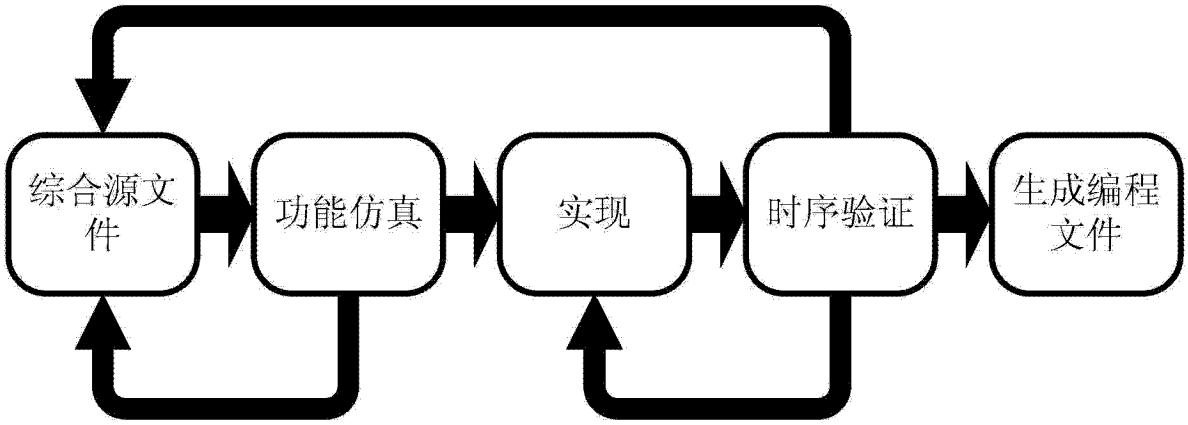


图 1

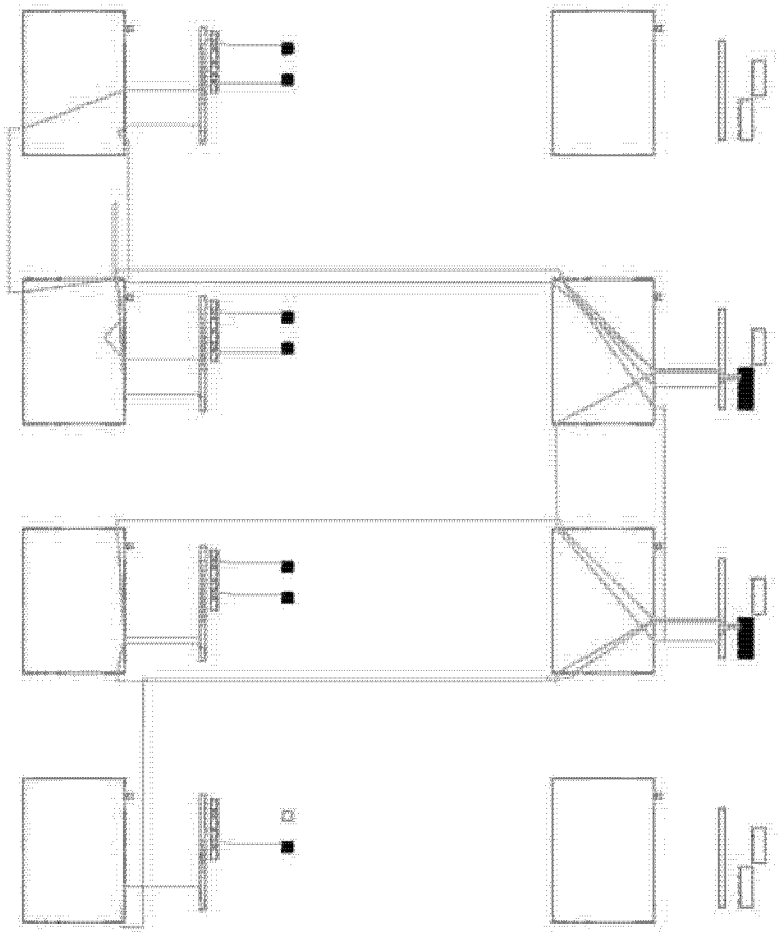


图 2

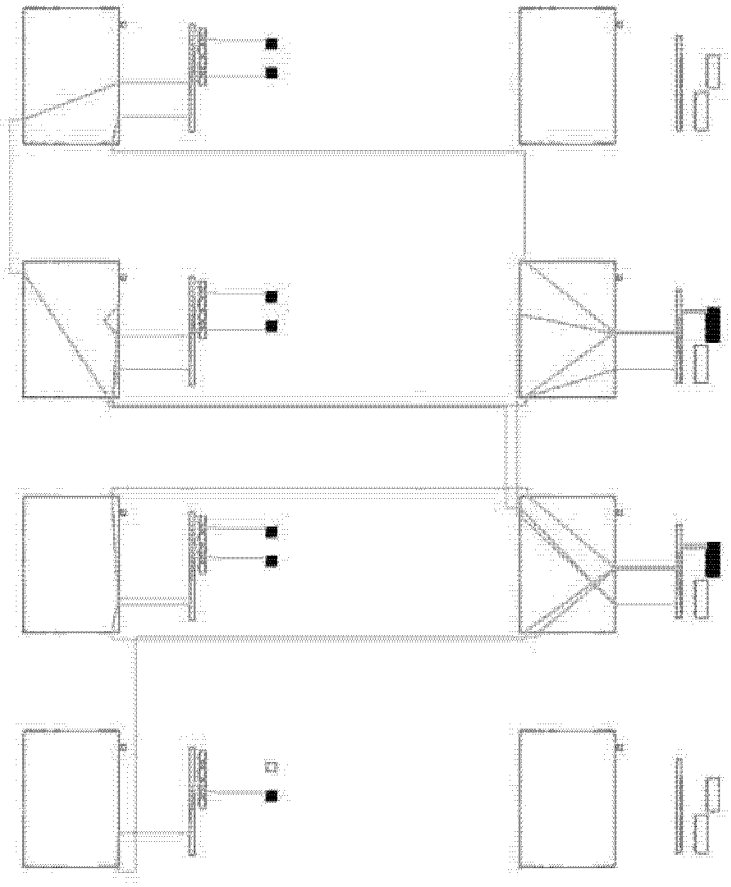


图 3

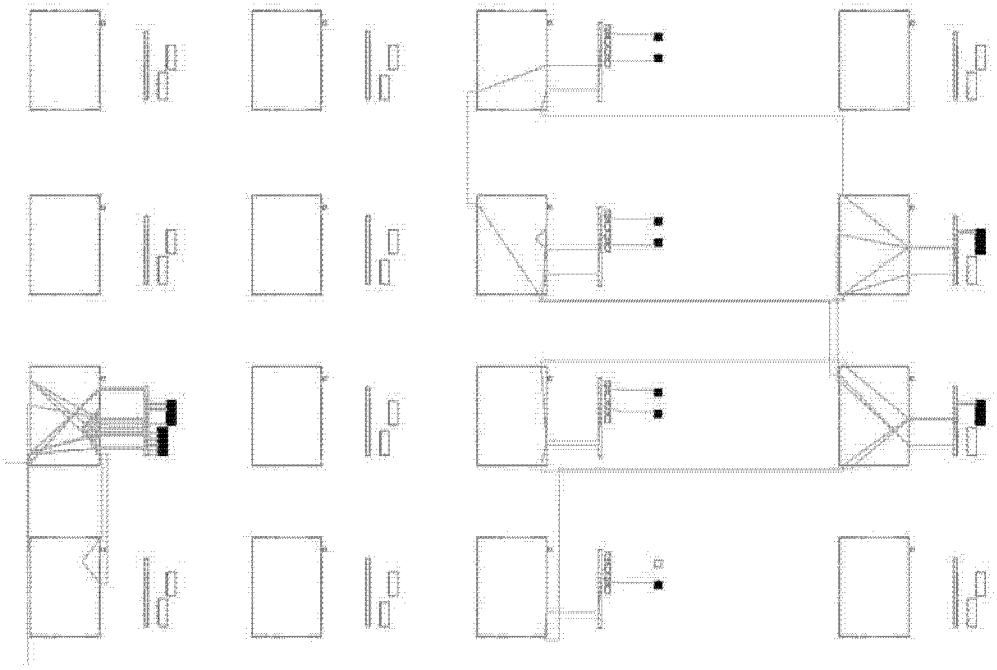


图 4

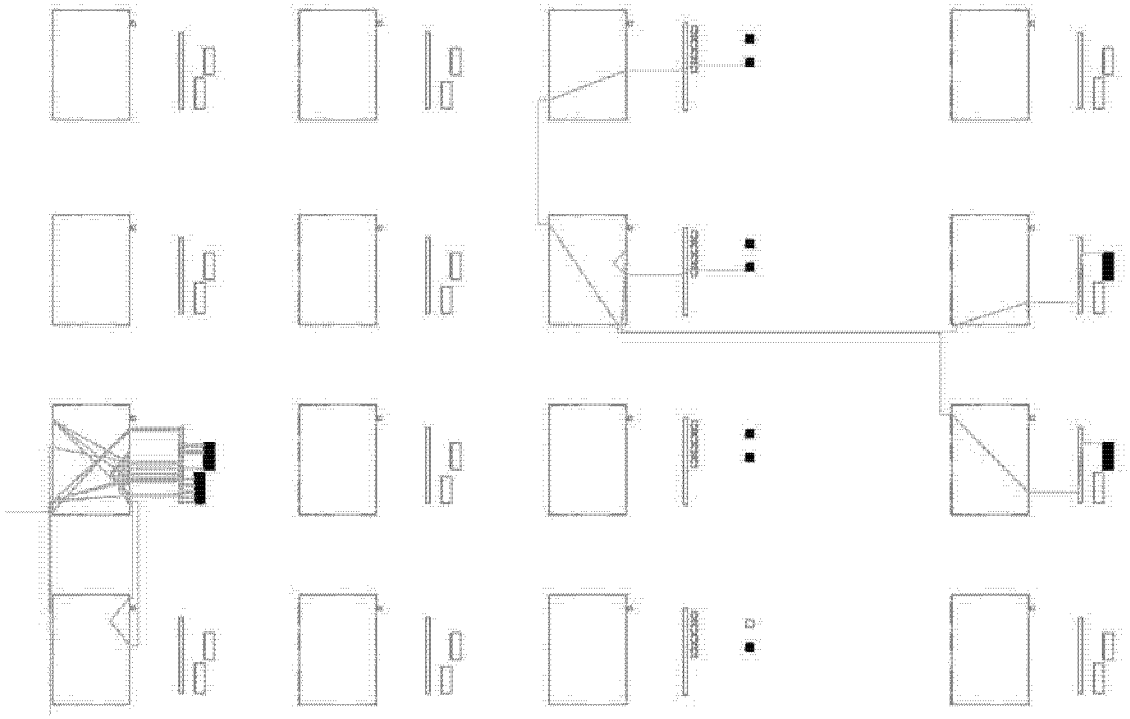


图 5

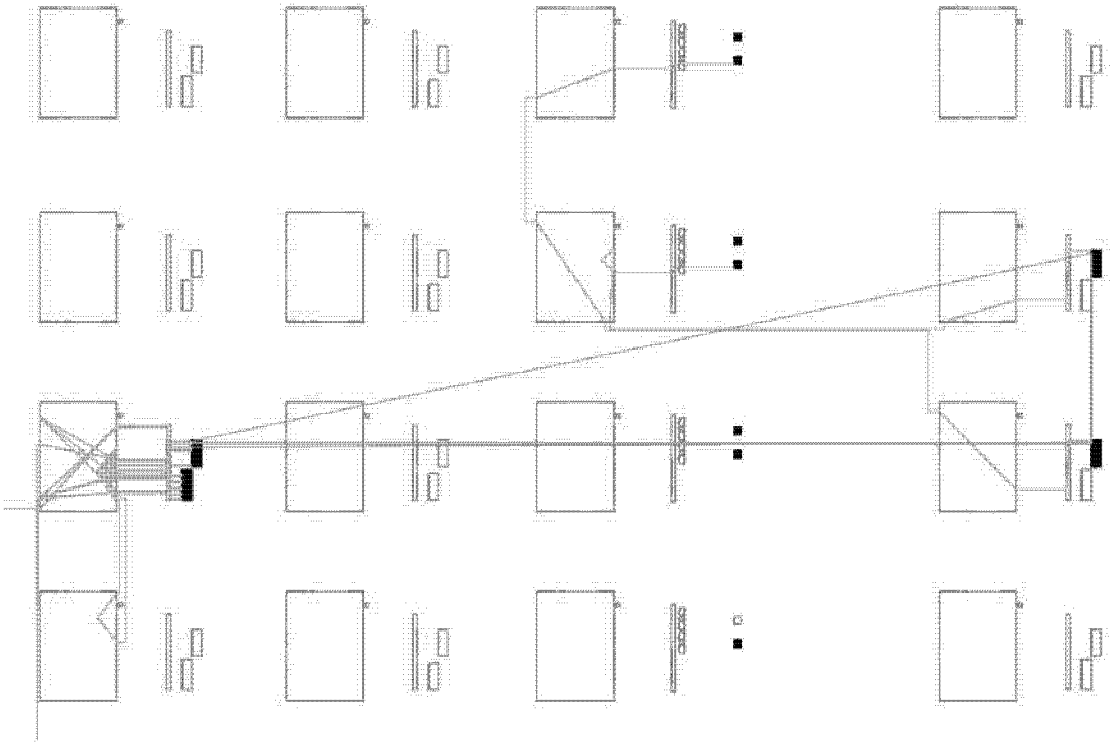


图 6

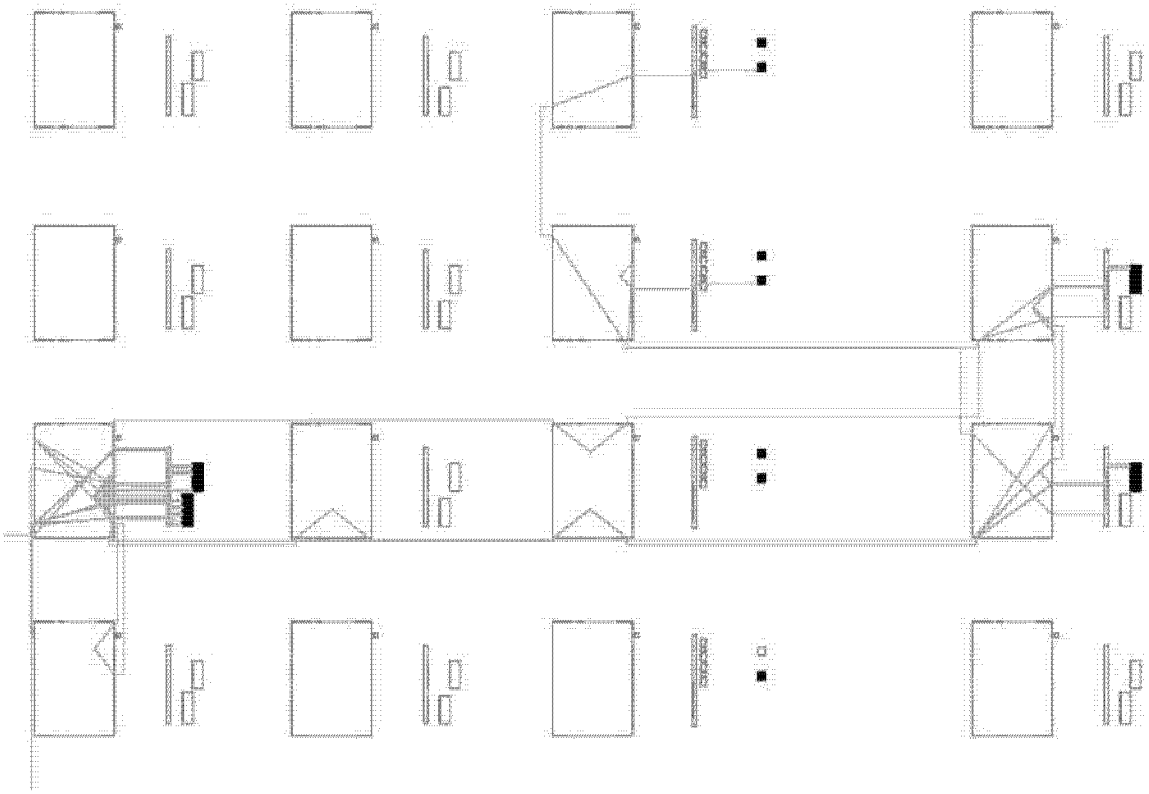


图 7

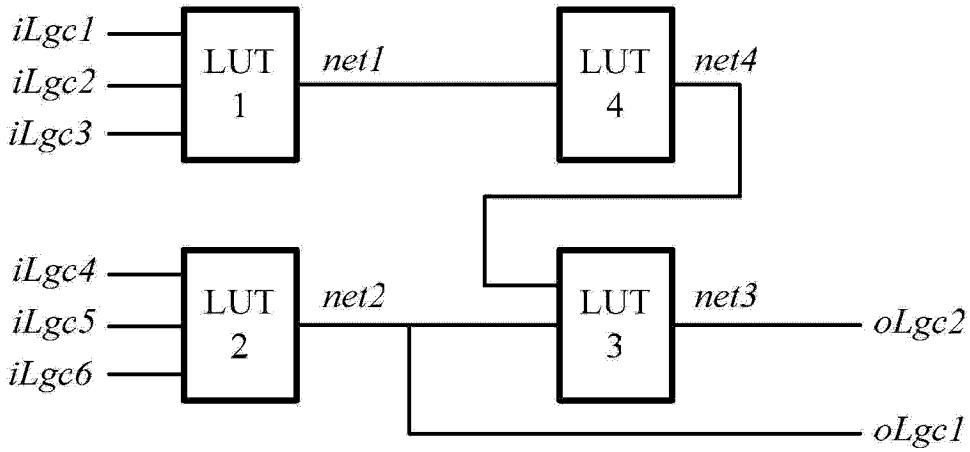

iLgc1
iLgc2
iLgc3
LUT 1
net1
LUT 4
net4
iLgc4
iLgc5
iLgc6
LUT 2
net2
LUT 3
net3
oLgc2
oLgc1

图 8

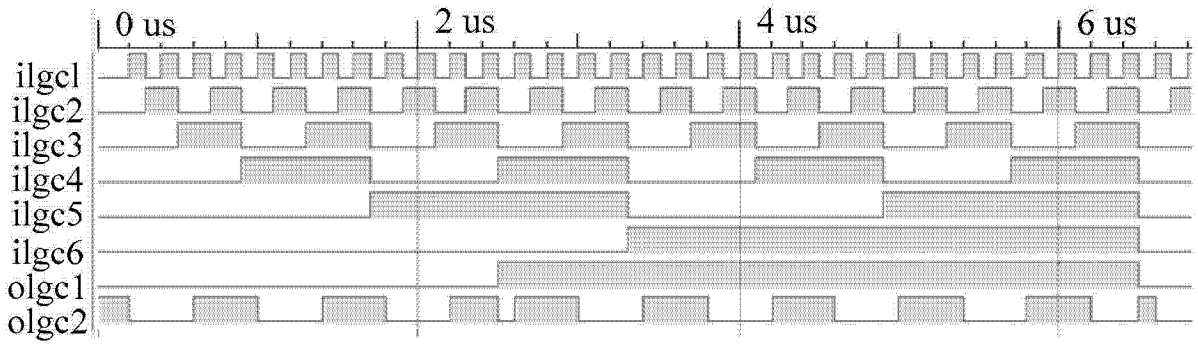

0 us
2 us
4 us
6 us
ilgc1
ilgc2
ilgc3
ilgc4
ilgc5
ilgc6
olgc1
olgc2

图 9

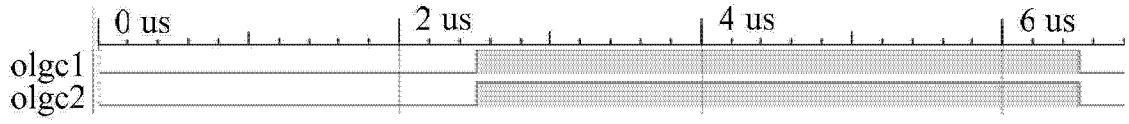

0 us
2 us
4 us
6 us
olgc1
olgc2

图 10

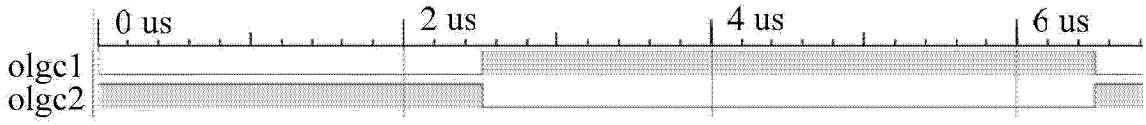

0 us
2 us
4 us
6 us
olgc1
olgc2

图 11

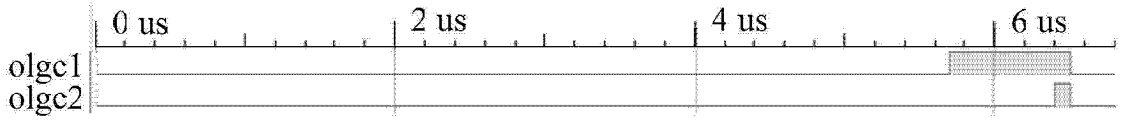


图 12

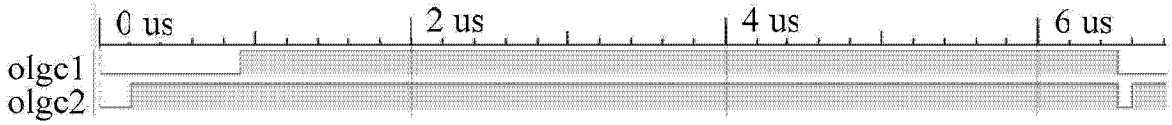


图 13

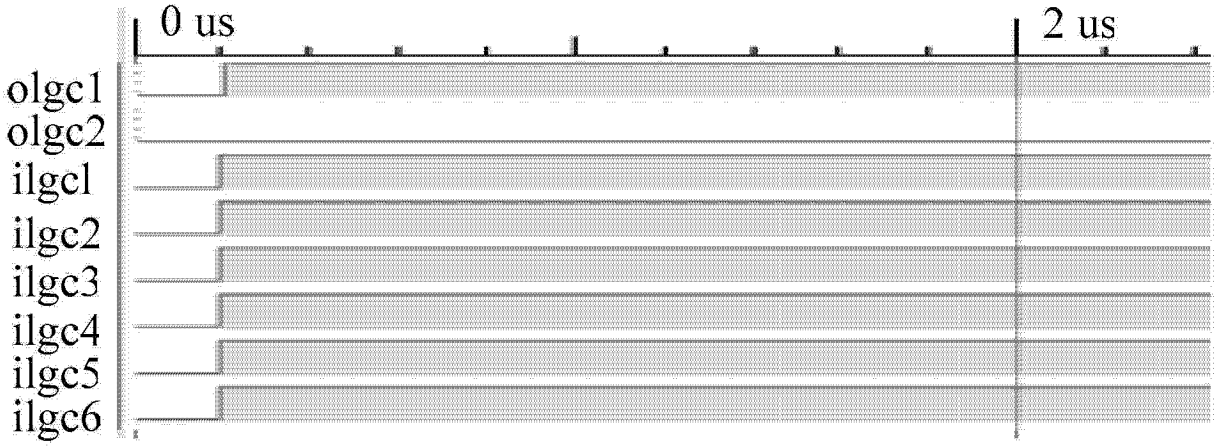


图 14

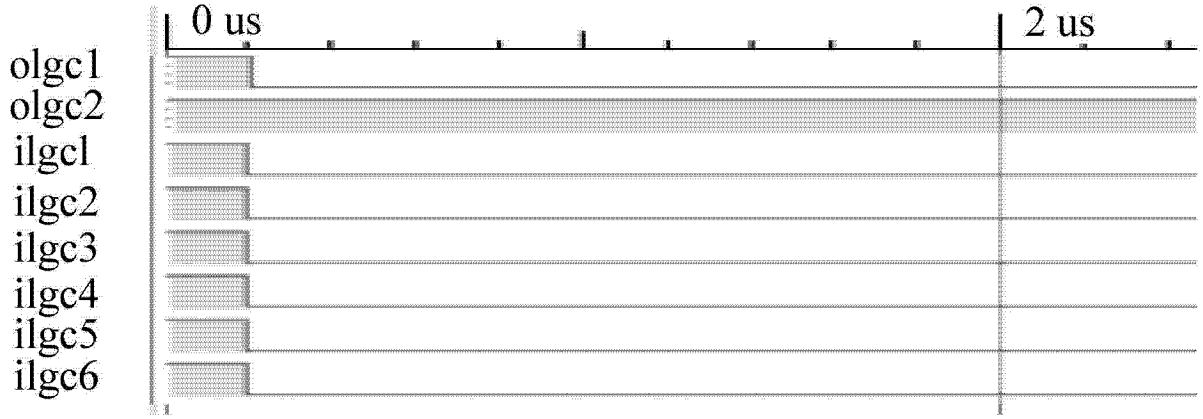


图 15

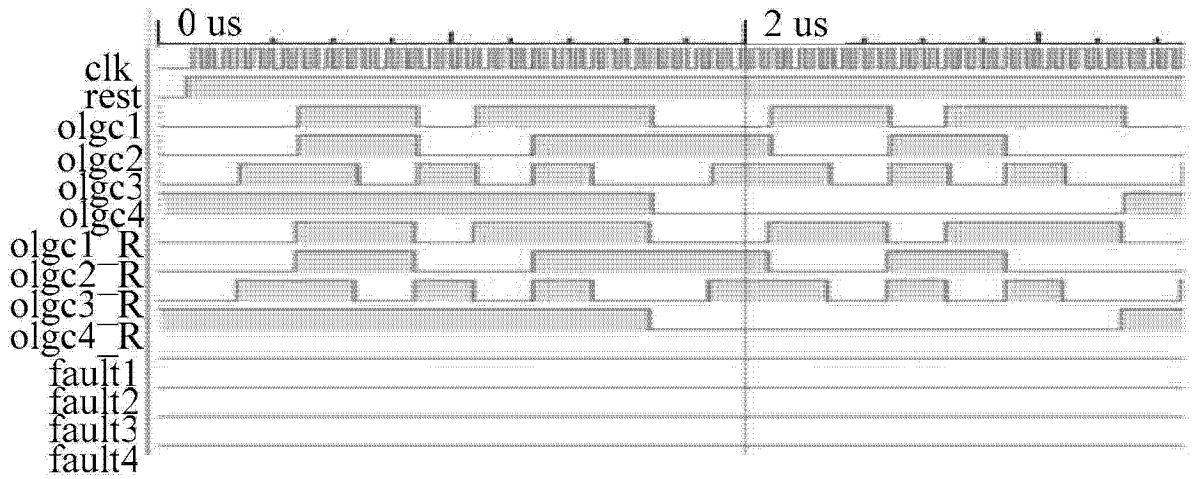

0 us
2 us
clk
rest
olgc1
olgc2
olgc3
olgc4
olgc1_R
olgc2_R
olgc3_R
olgc4_R
fault1
fault2
fault3
fault4

图 16

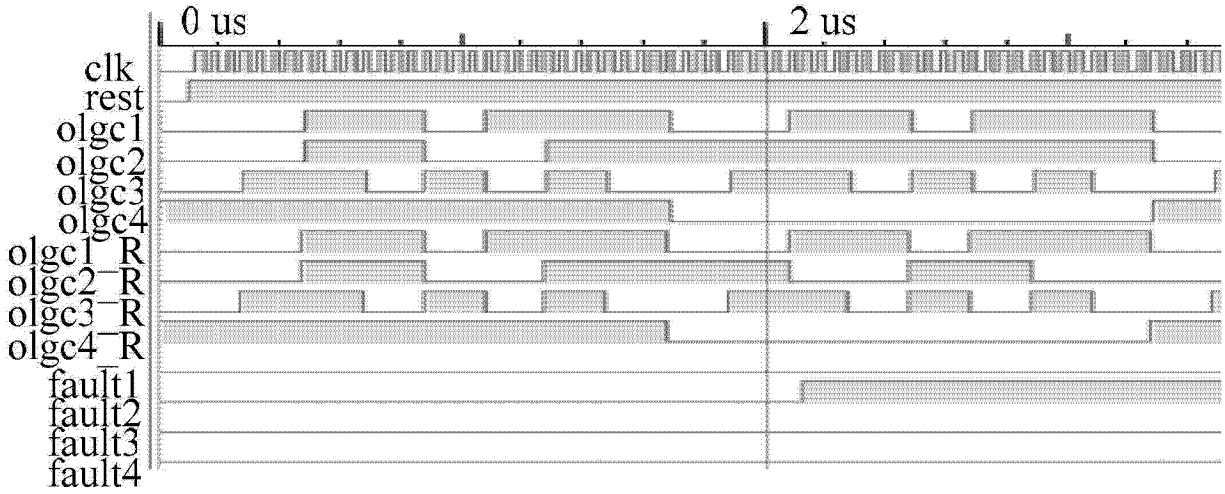

0 us
2 us
clk
rest
olgc1
olgc2
olgc3
olgc4
olgc1_R
olgc2_R
olgc3_R
olgc4_R
fault1
fault2
fault3
fault4

图 17