(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 104515947 A
(43)申请公布日 2015.04.15

(21)申请号 201410763028.0

(22)申请日 2014.12.12

(71)申请人 中国电子科技集团公司第五十八研究所
地址 214035 江苏省无锡市惠河路5号

(72)发明人 解维坤 章慧彬 张秋丽

(74)专利代理机构 北京中誉威圣知识产权代理有限公司 11279
代理人 蒋常雪

(51)Int.Cl.
G01R 31/28(2006.01)
G06F 11/36(2006.01)

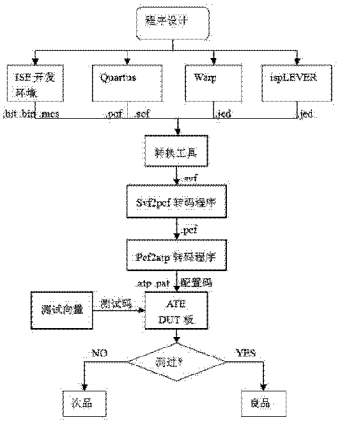
权利要求书1页 说明书5页 附图7页

(54)发明名称

可编程逻辑器件在系统编程的快速配置与测试方法

(57)摘要

本发明公开了一种可编程逻辑器件在系统编程的快速配置与测试方法，通过一次编程四次转码过程获得ISP状态机配置码；所述快速配置与测试方法的步骤为：在可编程逻辑器件相应的开发环境下进行测试配置程序开发，获得原始配置码；将原始配置码通过转换工具转换成串行向量格式的SVF文件；将SVF格式文件转换成PCF格式文件；利用C语言转码程序生成ATP格式文件；将ATP格式转成Pattern文件，利用ATE自动测试系统进行快速配置与测试。本发明方法能够自动生成ISP状态机配置码，可以进行多次配置与测试操作，大大提高了测试故障覆盖率，解决了可编程逻辑器件测试的问题；并且本发明方法具有通用性。



CN 104515947 A

1.可编程逻辑器件在系统编程的快速配置与测试方法,其特征在于,所述快速配置与测试方法通过一次编程四次转码过程获得 ISP 状态机配置码 ;

所述快速配置与测试方法的步骤为 :

(1)在可编程逻辑器件相应的相应开发环境下进行测试配置程序开发,获得原始配置码 ;

(2)将原始配置码通过转换工具转换成串行向量格式的 SVF 文件 ;

(3)将 SVF 格式文件转换成 PCF 格式文件 ;

(4)利用 C 语言转码程序将 PCF 格式文件中的有效数据提取出来直接生成 ATP 格式文件 ;

(5)将 ATP 格式转成测试系统所需 Pattern 文件,利用 ATE 自动测试系统进行可编程逻辑器件在系统编程的快速配置与测试。

2.如权利要求 1 所述的可编程逻辑器件在系统编程的快速配置与测试方法,其特征在于,不同公司的可编程逻辑器件采用不同的开发环境进行测试配置程序开发。

3.如权利要求 2 所述的可编程逻辑器件在系统编程的快速配置与测试方法,其特征在于,Xilinx 公司的可编程逻辑器件采用 ISE 开发环境进行测试配置程序开发获得 .bit、.bin、.mcs 格式的测试配置程序 ;Altera 公司的可编程逻辑器件采用 Quartus 开发环境进行测试配置程序开发获得 .pof、.sof 格式的测试配置程序 ;Lattice 公司的可编程逻辑器件采用 ispLEVER 开发环境进行测试配置程序开发获得 .jed 格式的测试配置程序 ;Cypress 公司的可编程逻辑器件采用 Warp 开发环境进行测试配置程序开发获得 .jed 格式的测试配置程序。

4.如权利要求 1 所述的可编程逻辑器件在系统编程的快速配置与测试方法,其特征在于,采用测试配置程序自动转码生成 ISP 配置码,ATE 自动测试系统直接在系统快速配置 ;不采用人工采集制作配置码和利用下载线以及编程器配置的方法。

5.如权利要求 1 所述的可编程逻辑器件在系统编程的快速配置与测试方法,其特征在于,可编程逻辑器件包括 Xilinx、Altera、Lattice、Cypress 公司的 PROM、PAL、GAL、PLA、CPLD、FPGA 可编程逻辑器件。

6.如权利要求 1 所述的可编程逻辑器件在系统编程的快速配置与测试方法,其特征在于,步骤(2)中的转换工具为 :Xilinx 公司的 iMPACT 工具,Altera 公司的“Creat JAM、SVF、ISC”工具,Cypress 公司的 ISR 工具,Lattice 公司的 ispVM 工具。

7.如权利要求 1 所述的可编程逻辑器件在系统编程的快速配置与测试方法,其特征在于,步骤(3)中使用的转换工具为 Svf2pcf 转码程序。

8.如权利要求 1 所述的可编程逻辑器件在系统编程的快速配置与测试方法,其特征在于,步骤(4)中使用的转换工具为 Pcf2atp 转码程序。

9.如权利要求 1-8 任一所述的可编程逻辑器件在系统编程的快速配置与测试方法,其特征在于,ATE 自动测试系统是美国泰瑞达公司的 J750EX 或 UltraFLEX 测试系统。

10.如权利要求 9 所述的可编程逻辑器件在系统编程的快速配置与测试方法,其特征在于,测试配置程序是依据被检测的可编程逻辑器件产品的数据手册、测试方法、测试规范、测试大纲的要求进行测试配置程序开发。

可编程逻辑器件在系统编程的快速配置与测试方法

[0001]

技术领域

[0002] 本发明属于集成电路技术领域，涉及一种可编程逻辑器件的配置与测试方法，具体涉及一种可编程逻辑器件在系统编程的快速配置与测试方法。

背景技术

[0003] 可编程逻辑器件包括 PROM、PAL、GAL、PLA、CPLD、FPGA 等，它们主要有可编程的与阵列、或阵列、门阵列等组成，可通过编程实现一定的逻辑功能。

[0004] 对可编程逻辑器件进行测试要对其内部可能包含的资源进行结构分析，经过一个测试配置（TC）和向量实施（TS）的过程，将其配置为具有特定功能的电路，再从应用级别上对电路进行测试，完成电路的功能及参数测试。

[0005] 在系统可编程 (In－System Programmable，简称 ISP) 技术是 20 世纪 80 年代末 Lattice 公司首先提出的一种先进的编程技术。所谓“在系统编程”是指对器件、电路板或整个电子系统的逻辑功能可随时进行修改或重构的能力。这种重构或修改可以在产品设计、制造过程中的每个环节，甚至在交付用户之后进行。支持 ISP 技术的可编程逻辑器件称为在系统可编程逻辑器件 (ISP－PLD)。

[0006] 传统的对可编程逻辑器件的编程一般都是通过下载线或编程器连接电脑对器件进行编程，对 CPLD 器件来说是将 JED 文件“下载 (Down Load)”到 CPLD 器件中去，对 FPGA 来说是将位流数据 Bit 文件“配置”到 FPGA 中去。对 CPLD 器件的测试往往是先用下载线编程后再到测试系统上去进行测试，只进行一次配置——测试过程，这种方法测试覆盖率低下，或者在用下载线编程过程中，利用逻辑分析仪去采集配置数据，再经过人工手动处理获得 ISP 配置码，这样非常费时费力而且很容易出错；对 FPGA 器件测试虽然可以通过从串或从并等配置方法实现在系统配置与测试，但对 JTAG 的测试仍然需要利用状态机方法进行配置与验证。

发明内容

[0007] 本发明需要解决的技术问题就在于克服现有技术的缺陷，提供一种可编程逻辑器件在系统编程的快速配置与测试方法，本发明方法能够自动生成 ISP 状态机配置码，能够利用 ATE 对 PROM、CPLD、FPGA 等可编程逻辑器件实现在系统快速配置，可以进行多次配置与测试操作，大大提高了测试故障覆盖率，解决了可编程逻辑器件测试的问题；并且本发明方法具有通用性，可广泛应用于 Xilinx、Altera、Lattice、Cypress 等公司的 PROM、PLD、CPLD、FPGA 等可编程逻辑器件的快速配置与测试之中。

[0008] 为解决上述问题，本发明采用技术方案为：

可编程逻辑器件在系统编程的快速配置与测试方法，所述快速配置与测试方法通过一次编程四次转码过程获得 ISP 状态机配置码；

所述快速配置与测试方法的步骤为：

（1）在可编程逻辑器件相应的相应开发环境下进行测试配置程序开发，获得原始配置码；

（2）将原始配置码通过转换工具转换成串行向量格式的 SVF 文件；

（3）将 SVF 格式文件转换成 PCF 格式文件；

（4）利用 C 语言转码程序将 PCF 格式文件中的有效数据提取出来直接生成 ATP 格式文件；

（5）将 ATP 格式转成测试系统所需 Pattern 文件，利用 ATE 自动测试系统进行可编程逻辑器件在系统编程的快速配置与测试。

[0009] 优选的，不同公司的可编程逻辑器件采用不同的开发环境进行测试配置程序开发。

[0010] 优选的，Xilinx 公司的可编程逻辑器件采用 ISE 开发环境进行测试配置程序开发获得 .bit、.bin、.mcs 格式的测试配置程序；Altera 公司的可编程逻辑器件采用 Quartus 开发环境进行测试配置程序开发获得 .pof、.sof 格式的测试配置程序；Lattice 公司的可编程逻辑器件采用 ispLEVER 开发环境进行测试配置程序开发获得 .jed 格式的测试配置程序；Cypress 公司的可编程逻辑器件采用 Warp 开发环境进行测试配置程序开发获得 .jed 格式的测试配置程序。

[0011] 优选的，采用测试配置程序自动转码生成 ISP 配置码，ATE 自动测试系统直接在系统快速配置；不采用人工采集制作配置码和利用下载线以及编程器配置的方法。传统的人工采集制作配置码和利用下载线以及编程器配置的方法，测试覆盖率低下，非常费时费力而且很容易出错，不利于提高可编程逻辑器件的快速配置与测试操作。

[0012] 优选的，可编程逻辑器件包括 Xilinx、Altera、Lattice、Cypress 等公司的 PROM、PAL、GAL、PLA、CPLD、FPGA 等可编程逻辑器件。

[0013] 优选的，步骤（2）中的转换工具为：Xilinx 公司的 iMPACT 工具，Altera 公司的“Creat JAM、SVF、ISC”工具，Cypress 公司的 ISR 工具，Lattice 公司的 ispVM 工具，针对不同公司的产品选择使用相对应的转换工具。

[0014] 优选的，步骤（3）中使用的转换工具为 Svf2pcf 转码程序。

[0015] 优选的，步骤（4）中使用的转换工具为 Pcf2atp 转码程序。

[0016] 优选的，ATE 自动测试系统是美国泰瑞达公司的 J750EX 或 Ultra-FLEX 测试系统。根据具体产品的性能和测试要求选择上述两个测试系统中的一个作为 ATE 自动检测系统。

[0017] 优选的，测试配置程序是依据被检测的可编程逻辑器件产品的数据手册、测试方法、测试规范、测试大纲的要求进行测试配置程序开发。

[0018] 本发明的优点和有益效果为：

本发明可编程逻辑器件在系统编程的快速配置与测试方法，能够自动生成 ISP 状态机配置码，能够利用 ATE 对 PROM、CPLD、FPGA 等可编程逻辑器件实现在系统快速配置，可以进行多次配置与测试操作，大大提高了测试故障覆盖率，解决了可编程逻辑器件测试的问题；

本发明可编程逻辑器件在系统编程的快速配置与测试方法，具有极大通用性，可广泛应用于 Xilinx、Altera、Lattice、Cypress 等公司的 PROM、PLD、CPLD、FPGA 等可编程逻辑器

件的快速配置与测试之中；

本发明可编程逻辑器件在系统编程的快速配置与测试方法，测试覆盖率高并且不容易出错，能够极大提高在系统编程的快速配置与测试的效率和质量。

附图说明

[0019] 图 1 为本发明可编程逻辑器件在系统编程的快速配置与测试方法流程图。

[0020] 图 2 为本发明使用的三状态 ISP 状态机转移图。

[0021] 图 3 为本发明使用的 IEEE1149.1 标准 ISP 状态机转移图。

[0022] 图 4 本发明创建 SVF 文件操作界面。

[0023] 图 5 本发明生成的 SVF 格式文件示意图。

[0024] 图 6 本发明 svf2pcf 转码操作界面。

[0025] 图 7 本发明生成的 PCF 格式文件示意图。

[0026] 图 8 本发明生成的 ATP 格式文件示意图。

具体实施方式

[0027] 下列实施例将进一步说明本发明。

[0028] 实施例 1

本发明采用技术方案为可编程逻辑器件在系统编程的快速配置与测试方法，所述快速配置与测试方法通过一次编程四次转码过程获得 ISP 状态机配置码；

快速配置与测试方法的步骤为：

(1) 在可编程逻辑器件相应的相应开发环境下进行测试配置程序开发，获得原始配置码；

(2)将原始配置码通过转换工具转换成串行向量格式的 SVF 文件；转换工具为：Xilinx 公司的 iMPACT 工具，Altera 公司的“Creat JAM、SVF、ISC”工具，Cypress 公司的 ISR 工具，Lattice 公司的 ispVM 工具。

[0029] (3) 将 SVF 格式文件转换成 PCF 格式文件；转换工具为 Svf2pcf 转码程序；

(4) 利用 C 语言转码程序将 PCF 格式文件中的有效数据提取出来直接生成 ATP 格式文件；转换工具为 Pcf2atp 转码程序；

(5)将 ATP 格式转成测试系统所需 Pattern 文件，利用 ATE 自动测试系统进行可编程逻辑器件在系统编程的快速配置与测试。

[0030] 本方法中不同公司的可编程逻辑器件采用不同的开发环境进行测试配置程序开发。Xilinx 公司的可编程逻辑器件采用 ISE 开发环境进行测试配置程序开发获得 .bit、.bin、.mcs 格式的测试配置程序；Altera 公司的可编程逻辑器件采用 Quartus 开发环境进行测试配置程序开发获得 .pof、.sof 格式的测试配置程序；Lattice 公司的可编程逻辑器件采用 ispLEVER 开发环境进行测试配置程序开发获得 .jed 格式的测试配置程序；Cypress 公司的可编程逻辑器件采用 Warp 开发环境进行测试配置程序开发获得 .jed 格式的测试配置程序。

[0031] 本方法采用测试配置程序自动转码生成 ISP 配置码，ATE 自动测试系统直接在系统快速配置；不采用人工采集制作配置码和利用下载线以及编程器配置的方法。传统的人

工采集制作配置码和利用下载线以及编程器配置的方法，测试覆盖率低下，非常费时费力而且很容易出错，不利于提高可编程逻辑器件的快速配置与测试操作。

[0032] 可编程逻辑器件包括 Xilinx、Altera、Lattice、Cypress 等公司的 PROM、PAL、GAL、PLA、CPLD、FPGA 等可编程逻辑器件。

[0033] ATE 自动测试系统包括：美国泰瑞达公司的 J750EX 和 / 或 Ultra-FLEX 测试系统。根据被检测的可编程逻辑器件的产品特性选择上述两个自动检测系统中的一个作为 ATE 自动测试系统；

测试配置程序是依据被检测的可编程逻辑器件产品的数据手册、测试方法、测试规范、测试大纲的要求进行测试配置程序开发。

[0034] 本实施例中使用的测试配置程序是本领域的普通技术人员能够根据检测需求进行编写，属于现有的公知的成熟技术，在本实施例中不再叙述测试配置程序的相关要求与方法，也不属于本发明的保护范围。

[0035] 结合图 1 至图 8 说明本实施例的快速配置与测试方法，

本方法的实施对象是所有支持 ISP(在系统可编程)的可编程逻辑器件，包括 PROM、PAL、GAL、PLA、CPLD、FPGA 等。根据 ISP 状态机原理，通过一些转码工具以及利用 C 语言编写的转码程序经过四次转码过程，能够快速自动的生成在系统配置所需配置码，再利用 ATE 直接加载配置码将器件配置成具有一定功能的电路，然后再进行全面的功能和参数测试，整个配置与测试流程如图 1 所示。

[0036] ISP 状态机有两种：三状态 ISP 状态机和 IEEE1149.1 标准的 ISP 状态机。最早由 Lattice 公司提出的 ISP 状态机只有三个状态：闲置态 (IDLE)、移位态 (SHIFT) 和执行态 (EXECUTE)，其状态转移如图 2 所示。后来由联合测试活动组织 (JTAG) 提出来了边界扫描技术，IEEE 对此制定了测试标准，称为 IEEE1194.1 标准，符合该标准的 ISP 状态机共有十六个状态，如图 3 所示。两种状态机都有四个信号端口：模式选择端 MODE（TMS）、数据输入端 SDI（TDI）、数据输出端 SDO（TDO）、时钟输入端 SCLK（TCK），根据状态机控制这些端口状态即可将配置数据串行的写入器件，实现 ISP 器件的在系统编程。

[0037] 串行向量 (SVF) 格式是一种描述状态机编程的指令格式，可以记录整个 ISP 编程过程信息。大部分可编程逻辑器件的开发环境都支持将配置数据文件转换成 SVF 格式。用不同的开发环境完成配置程序编写后，能自动生成 bit、bin、mcs、jed 、pof 等格式的配置文件，首先需要将其转换成串行向量格式 (SVF) 文件。以 Xilinx 公司的 ISE 开发环境为例，生成配置程序文件后进入 Configure Device(iMPACT) 界面 (如图 4 所示)，然后用 Boundary Scan，加入目标配置器件和配置文件，然后选择菜单栏——Output——SVF File——Create SVF File，开始录制 SVF 文件，现在对目标器件的任何操作 (Program、Verify、Erase、Get Device ID 等) 都可以记录在 SVF 文件中，录制完成后，选择菜单栏——Output——SVF File——Stop Writing to SVF File，即可生成 SVF 格式文件，如图 5 所示。

[0038] 生成 SVF 格式文件后，接着需要利用转码软件 (其操作界面如图 6 所示) 将 SVF 格式转换成 PCF 格式文件，如图 7 所示。

[0039] PCF 格式文件中双引号中的数据便是需要配置到目标器件中的配置数据，另外，还有比如“RUNTEST IDLE 100000 TCK”或“wait 100000s”之类信息，意思是需要在状态机的“Run-Test-Idle”状态等待 100000 个时钟周期，这些有用数据需要提取出来。可以编写 C

语言程序将这些有效数据提取出来，并转换成 ATP 格式的二进制向量文件，如图 8 所示。

[0040] 最后，再利用测试系统配套软件转成与测试系统相应的测试码格式文件，比如 Teradyne 公司的 IG-XL 中的 Pattern compiler 工具将 ATP 格式转换成 Pattern 文件，就可以利用 ATE 加载配置码对目标器件进行在系统配置与测试。

[0041] 因而，本发明的快速配置与测试方法实现了在系统快速配置，可以进行多次配置与测试操作，大大提高了测试故障覆盖率，解决了可编程逻辑器件测试的问题；具有极大的通用性，可广泛应用于 Xilinx、Altera、Lattice、Cypress 等公司的 PROM、PLD、CPLD、FPGA 等可编程逻辑器件的快速配置与测试之中；并且，测试覆盖率高并且不容易出错，能够极大提高在系统编程的快速配置与测试的效率和质量。

[0042] 最后应说明的是：显然，上述实施例仅仅是为清楚地说明本发明所作的举例，而并非对实施方式的限定。对于所属领域的普通技术人员来说，在上述说明的基础上还可以做出其它不同形式的变化或变动。这里无需也无法对所有的实施方式予以穷举。而由此所引申出的显而易见的变化或变动仍处于本发明的保护范围之中。

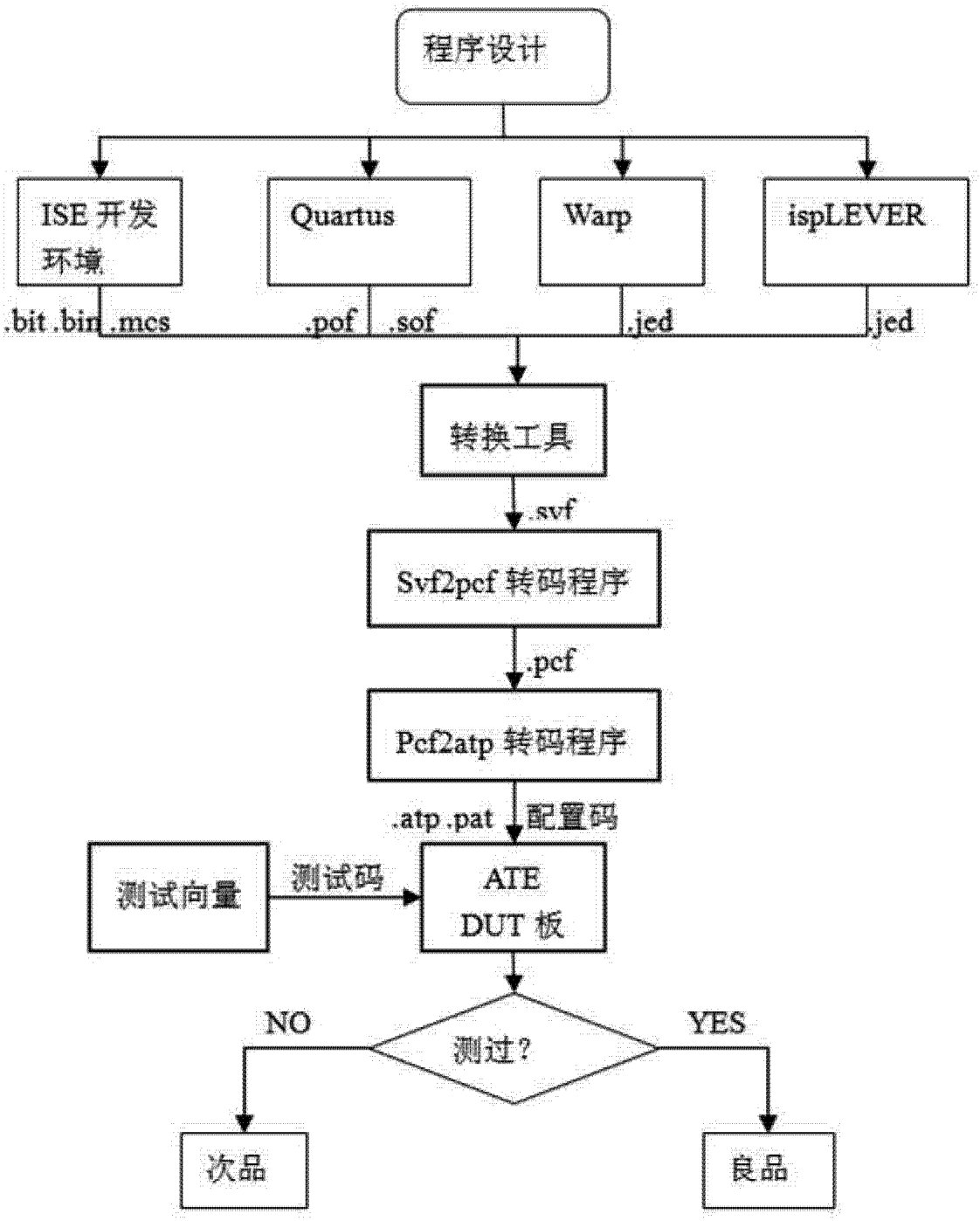

程序设计
ISE 开发环境
Quartus
Warp
ispLEVER
.bit .bin .mcs
.pof
.sof
.jed
.jed
转换工具
.svf
Svf2pcf 转码程序
.pcf
Pcf2atp 转码程序
.atp .pat
配置码
测试向量
测试码
ATE
DUT 板
NO
测过?
YES
次品
良品

图 1

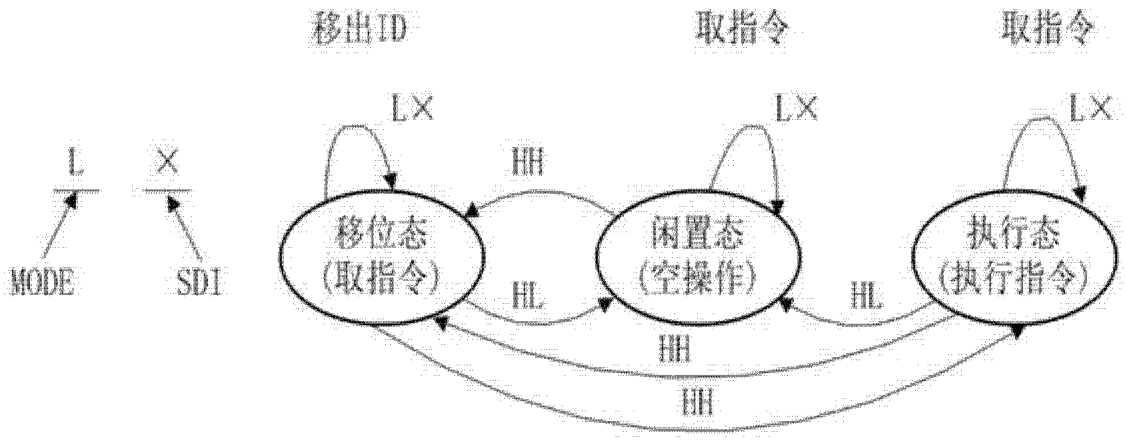

移出ID
取指令
取指令
L
X
MODE
SDI
LX
LX
LX
HH
移位态
(取指令)
闲置态
(空操作)
执行态
(执行指令)
HL
HL
HH
HH

图 2

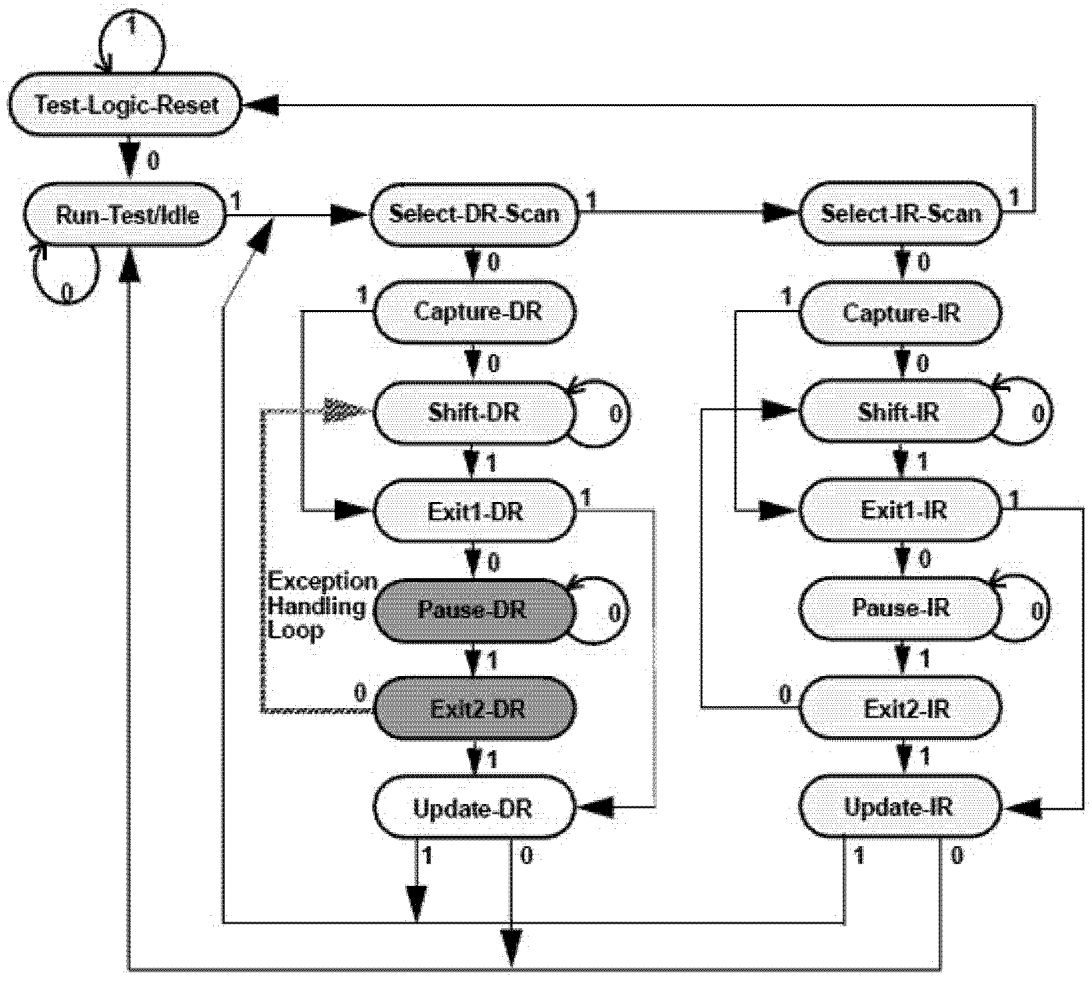

Test-Logic-Reset
Run-Test/Idle
Select-DR-Scan
Capture-DR
Shift-DR
Exit1-DR
Pause-DR
Exit2-DR
Update-DR
Select-IR-Scan
Capture-IR
Shift-IR
Exit1-IR
Pause-IR
Exit2-IR
Update-IR
Exception
Handling
Loop

图 3

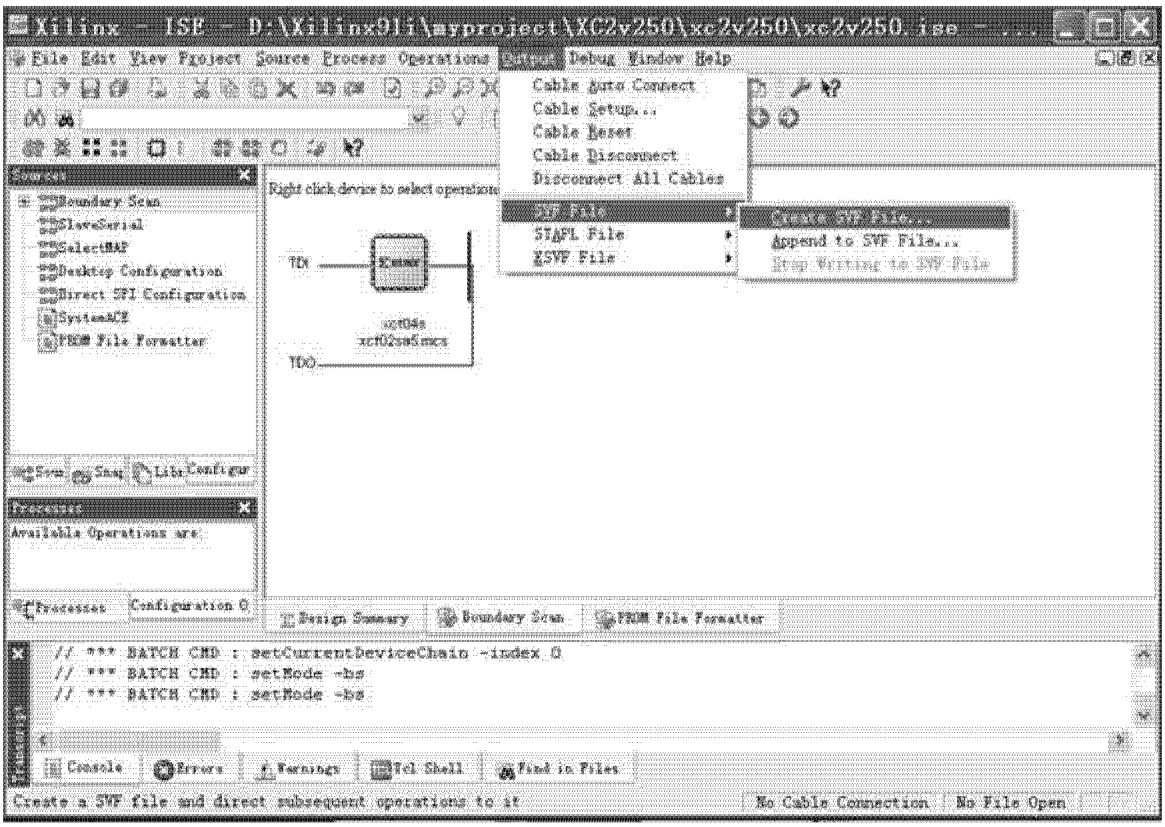

Xilinx - ISE - D:\Xilinx91i\myproject\XC2v250\xc2v250\xc2v250.ise - ...
File Edit View Project Source Process Operations Debug Window Help
Cable Auto Connect
Cable Setup...
Cable Reset
Cable Disconnect
Disconnect All Cables
SVF File
STAPL File
XSVF File
Create SVF File...
Append to SVF File...
Stop Writing to SVF File
Right click device to select operations
TDI
TDO
Design Summary
Boundary Scan
PROM File Formatter
// *** BATCH CMD : setCurrentDeviceChain -index 0
// *** BATCH CMD : setMode -bs
// *** BATCH CMD : setMode -bs
Console
Errors
Warnings
Tcl Shell
Find in Files
Create a SVF file and direct subsequent operations to it
No Cable Connection
No File Open

图 4

```
// Created using Xilinx iMPACT Software [ISE - 9.1i]
TRST OFF;
ENDIR IDLE;
ENDDR IDLE;
STATE RESET;
STATE IDLE;
FREQUENCY 1E6 HZ;
TIR 0 ;
HIR 0 ;
TDR 0 ;
HDR 0 ;
TIR 0 ;
HIR 0 ;
HDR 0 ;
TDR 0 ;
//Loading device with 'idcode' instruction.
SIR 5 TDI (01) SMASK (1f) ;
SDR 32 TDI (00000000) SMASK (ffffffff) TDO (f484ffff) MASK (0fff8001) ;
TIR 0 ;
HIR 0 ;
TDR 0 ;
HDR 0 ;
TIR 0 ;
HIR 0 ;
TDR 0 ;
HDR 0 ;
// Loading devices with 'enable' or 'bypass' instruction.
SIR 5 TDI (09) ;
// Loading devices with 'erase' or 'bypass' instruction.
ENDIR IRPAUSE;
SIR 5 TDI (0a) SMASK (1f) ;
ENDIR IDLE;
STATE  IREXIT2 IRUPDATE DRSELECT DRCAPTURE DREXIT1 DRPAUSE;
RUNTEST DRPAUSE 20 TCK;
STATE  IDLE;
RUNTEST IDLE 100000 TCK;
STATE  DRPAUSE;
```

图 5

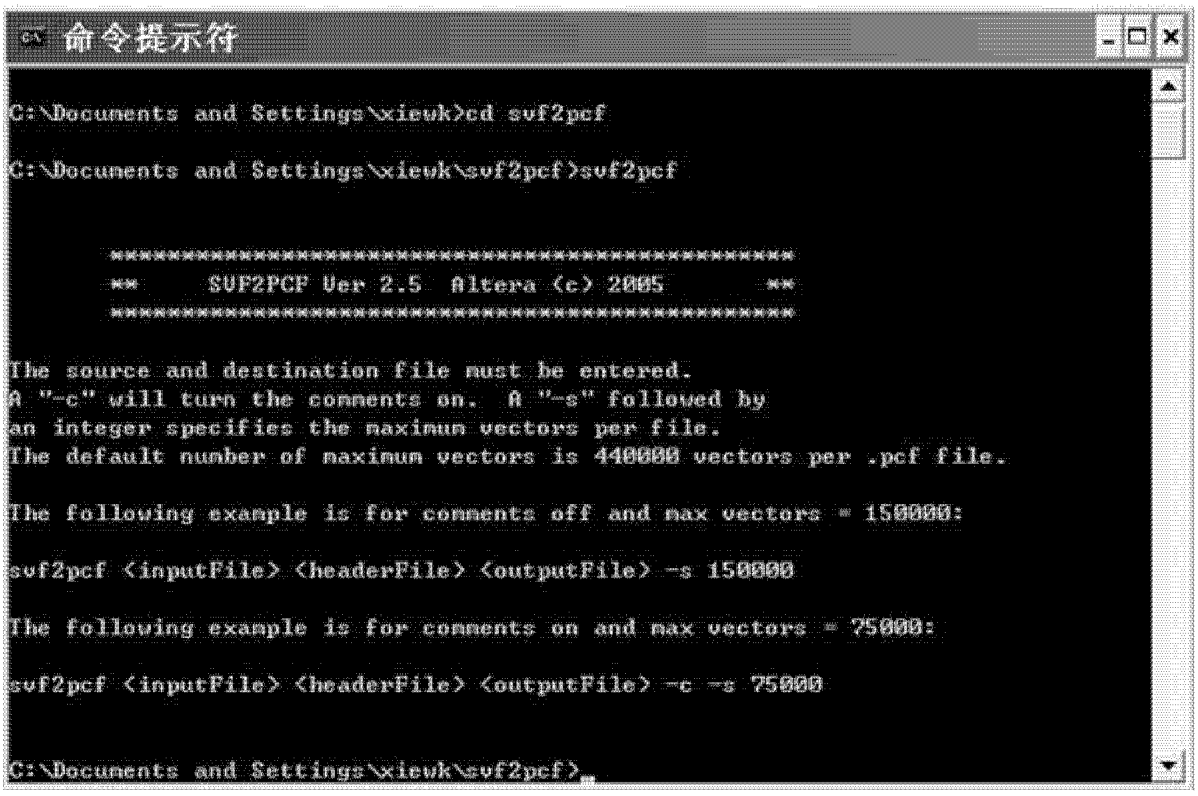

命令提示符
C:\Documents and Settings\xiewk>cd svf2pcf
C:\Documents and Settings\xiewk\svf2pcf>svf2pcf
****
** SVF2PCF Ver 2.5 Altera (c) 2005 **
****
The source and destination file must be entered.
A "-c" will turn the comments on. A "-s" followed by
an integer specifies the maximum vectors per file.
The default number of maximum vectors is 440000 vectors per .pcf file.
The following example is for comments off and max vectors = 150000:
svf2pcf <inputFile> <headerFile> <outputFile> -s 150000
The following example is for comments on and max vectors = 75000:
svf2pcf <inputFile> <headerFile> <outputFile> -c -s 75000
C:\Documents and Settings\xiewk\svf2pcf>

图 6

```
pcf order default Scan is TCK, TMS, TDI, TDO

unit "Test"
pcf
use pcf order Scan
"01ZX"
"11ZX" !  first reset clock
"01ZX"
"11ZX" !  second reset clock
"01ZX"
"11ZX" !  third reset clock
"01ZX"
"11ZX" !  fourth reset clock
"01ZX"
"11ZX" !  fifth reset clock, Test-Logic-Reset
! STATE IDLE;
"01ZX"
"11ZX" !Test-Logic-Reset
"00ZX"
"10ZX" !Run-Test-Idle
! FREQUENCY 1E6 HZ;
! TIR 0 ;
! HIR 0 ;
! TDR 0 ;
! HDR 0 ;
! TIR 0 ;
! HIR 0 ;
! HDR 0 ;
! TDR 0 ;
! //Loading device with 'idcode' instruction.
! SIR 5 TDI (01) SMASK (1f) ;
"01ZX"
"11ZX" !Select-DR-Scan
"01ZX"
"11ZX" !Select-IR-Scan
"00ZX"
"10ZX" !Capture-IR
```

图 7

```
import tset tsb1;
vector        ( $tset , TCK, TMS, TDI, TDO)
{
              > T_1M     1 X X X ;
              > -        1 X X X ;
repeat 200    > -        1 X X X ;
              > -        1 X X X ;
              > -        0 1 x X ;
              > -        1 1 x X ; //  first reset clock
              > -        0 1 x X ;
              > -        1 1 x X ; //  second reset clock
              > -        0 1 x X ;
              > -        1 1 x X ; //  third reset clock
              > -        0 1 x X ;
              > -        1 1 x X ; //  fourth reset clock
              > -        0 1 x X ;
              > -        1 1 x X ; //  fifth reset clock, Test-Logic-Reset
              > -        0 0 x X ;
              > -        1 0 x X ; //Run-Test-Idle
              > -        0 1 x X ;
              > -        1 1 x X ; //Select-DR-Scan
              > -        0 1 x X ;
              > -        1 1 x X ; //Select-IR-Scan
              > -        0 0 x X ;
              > -        1 0 x X ; //Capture-IR
              > -        0 0 x X ;
              > -        1 0 x X ; //Shift-IR
              > -        0 0 0 X ;
              > -        1 0 0 X ; //Shift-IR SIR bit 0
              > -        0 0 0 X ;
              > -        1 0 0 X ; //Shift-IR SIR bit 1
              > -        0 0 1 X ;
              > -        1 0 1 X ; //Shift-IR SIR bit 2
              > -        0 0 0 X ;
              > -        1 0 0 X ; //Shift-IR SIR bit 3
              > -        0 0 0 X ;
              > -        1 0 0 X ; //Shift-IR SIR bit 4
```

图 8