文章编号：2095-2163(2019)03-0324-04 中图分类号：TP311 文献标志码：A

基于 Eclipse CDT 的 FPGA 工程过程管理优化设计

胡启罡，曲明成，吴翔虎
（哈尔滨工业大学 计算机科学与技术学院，哈尔滨 150001）

摘 要： 如果使用 Xilinx 公司的 Vavido 软件直接用于 FPGA 开发，则需要进行过程繁琐的图形化配置，容易出错且不方便新手使用。为了解决这一问题，本文详细研究了 Vavido 软件中有关 FPGA 工程编译部分的实现原理，并在 Eclipse CDT 提供的基础扩展点上针对 FPGA 工程编写插件，通过检索工程目录，自动生成编译 FPGA bit 文件的 tcl 脚本，并以命令行的方式调用 Vavido 的相关功能，完成 bit 文件的生成。实现了 FPGA 工程的建立、编译和烧写功能。

关键词： FPGA；Vavido；Eclipse CDT；tcl 脚本生成；bit 文件生成

Optimization design of FPGA engineering process management based on Eclipse CDT

HU Qigang, QU Mingcheng, WU Xianghu
(School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China)

【Abstract】 If Xilinx´s Vavido software is used for direct FPGA development, it is required to process complex graphical configuration, which is easy to cause the problem and inconvenient for beginners. In order to solve this problem, this paper studies the realization principle of the FPGA project compilation in the Vavido software, and based on the expansion of development provided by Eclipse CDT, according to the FPGA project, plug-ins are written. By retrieving the engineering directory, the tcl script compiling FPGA bit files is automatically generated, the related functions of the Vavido are invoked by the command line method, and the generation of the bit file is completed. The FPGA project establishment and the functions of compiling and writing are really achieved.

【Key words】 FPGA; Vavido; Eclipse CDT; tcl script generation; bit file generation

0 引 言

本文是实验室项目《通用可配置嵌入式软件集成开发环境》中的 FPGA 部分。该项目涉及到对 Xilinx 公司 ZC706 开发板的调试工作，其中包含 FPGA 部分。FPGA，即现场可编程门阵列，过程中，主要使用硬件描述语言（Verilog 或 VHDL）来完成电路设计。

项目的初期，通过使用 Xilinx 公司的 Vavido 软件实现相关的操作。但是在设计过程中发现，使用 Vavido 图形化界面进行 FPGA 开发，需要提供繁琐的配置，过程复杂，容易出错且对新手不友好。同时，如果已有外部硬件描述文件，无法方便快捷地生成 bit 文件。

目前，尚未有一款良好的开源项目能够支持 FPGA 的编译工作，故本文仍在 Vavido 的基础上研究其实现机制。

其实，Vivado 软件的核心是一个脚本解释器，所有操作都附有对应的 tcl 脚本可以调取执行，GUI 界面也是将各种脚本命令封装为图形化界面。对此，本文研究了 Vivado 中关于 FPGA 部分的 tcl 脚本，并对其进行了抽取集成。

而 Eclipse 作为一类优秀的开源框架，设计融入了大量的扩展节点供开发者集成自定义的功能。本文 FPGA 部分的功能就是在 Eclipse CDT 增配的扩展点基础上对 Xilinx 公司的 Vivado 设计工具进行了集成。

在本文中，后台采用 tcl 交互式命令行的模式启动 Vivado，调用 Vivado 的相关功能，即调用 Java 提供的 *exe*( ) 函数，在 cmd 命令行中执行“vivado -mode tcl tcl_path”指令。tcl_path 为该指令需要的参数，即为编译 bit 文件所需的 tcl 脚本的绝对路径。

基金项目： 国家自然科学基金(61402131)。

作者简介： 胡启罡(1993-)，男，硕士研究生，主要研究方向：嵌入式软件集成开发环境；曲明成(1980-)，男，博士，讲师，主要研究方向：自动化软件工程、嵌入式计算；吴翔虎(1968-)，男，博士，教授，CCF 高级会员，主要研究方向：嵌入式计算、操作系统、高可靠软件工程。

收稿日期： 2017-06-07

1　FPGA 工程的建立

在 Eclipse CDT 的基础上集成 FPGA 开发功能，首先需要能够建立 FPGA 工程。FPGA 工程是不同于 C/Java 工程的自定义新工程类型。研究可知，FPGA 工程建立的函数流程则如图 1 所示。

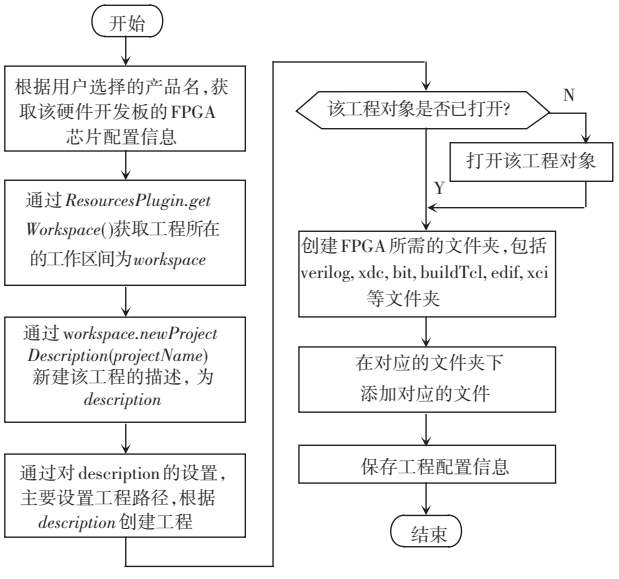

图 1　FPGA 工程建立函数流程图

Fig. 1　Function flow chart of FPGA engineering

由图 1 可知，FPGA 工程建立的步骤流程可详述如下。

(1)根据用户选择的产品名，从已集成的硬件配置数据中获取该硬件开发板的 FPGA 芯片配置信息。调用 *ResourcesPlugin.getWorkspace*( ).*getRoot*( ).*getProject*(*projectName*) 函数新建工程对象 *project*。

(2) 通过 *ResourcesPlugin.getWorkspace*( ) 函数获取待建工程所在的工作区间 *workspace*。

(3) 通过 *workspace.newProjectDescription* (*projectName*) 函数新建待创立工程的描述信息 *description*，参数为工程名。

(4) 通过对 *description* 的设置，主要通过 *description.setLocation*(*projectLocation*) 设置工程路径，参数为创建后工程所在的绝对路径。

(5) 通过调用函数 *project.create*(*description*, *null*) 创建工程。判断该工程对象是否打开，如果没有打开，则通过函数 *project.open*(*monitor*) 打开该工程。

(6) 创建 FPGA 所需的文件夹，包括 verilog、xdc、bit、buildTcl、edif、xci 等文件夹，并在指定的文件夹下添入相应的源代码文件。

(7) 保存工程配置信息。

研究中，经由上述的操作，即可新建出 FPGA 工程。

2　tcl 脚本及 bit 文件的自动生成

2.1　tcl 脚本及 bit 文件的自动生成分析

tcl 脚本的自动生成首先需要添加一个功能按钮来提供相关操作。

在 org.eclipse.ui.actionSets 扩展节点上，设计实现了一个生成 FPGA bit 文件的操作功能按钮。该按钮能够对 FPGA 工程进行编译，包括生成 tcl 脚本以及根据该脚本自动生成 bit 文件。需要在 plugin.xml 中进行配置。

研究可得，在扩展节点上创建新按钮的配置测试代码可表述如下。

```
-  -
<extenstion
      point = "org.eclipse.ui.actionSets">
   <actionSet
        id = "Embedded.actionSet1"
        label = "build"
        visible = "true">
       <action
          class = "test.function.FpgaBuildAction"
          disabledIcon = "icons/fpgaBuild.png">
```

分析可知，在该 xml 中，action class 属性指向了该按钮的功能 Java 文件，在该文件中，编写了该按钮的点击事件。重写 *run* 函数，定制形成了自己的业务操作。

在该按钮的点击事件中，设计了如下处理内容：

(1)检索工程目录，如果已有 bit 文件，提示用户是否重新生成，重新生成，则执行第(2)步操作。

(2)检索工程目录，如果已有生成 bit 文件的 tcl 脚本，则执行“vivado -mode batch -source build.tcl”，根据此脚本生成 bit 文件。如果没有 tcl 脚本，则递归遍历整个工程，根据工程所包含的文件，采用模板填充技术自动生成 tcl 脚本并执行。

首先需要规定出 tcl 脚本的标准模板，在此基础上根据工程包含的文件以及用户选择的产品类型做出相应修改，自动生成 tcl 脚本文件，tcl 脚本文件模板如图 2 所示。

(3)执行 tcl 脚本出现错误，则将报错信息展示给用户，用户可以根据报错信息，修改自己的工程文件，也可直接修改 tcl 脚本，该部分的函数流程可如图 3 所示。在此过程中，工程目录检索采用深度优先搜索算法。

```
# 定义工程文件的存放路径
set outputDir ./obj
read_vhdl -library bftLib [ glob ./Sources/hdl/bftLib/*.vhdl ]#指定需要添加的VHDL库文件，g
read_vhdl ./Sources/hdl/bfc.vhdl#指定需要添加的VHDL文件
read_verilog  [ glob ./SRC/*.v ]#指定需要添加的Verilog文件，glob是扫描某个路径下的全部文件
read_xdc [ glob ./CONSTRS/*.xdc ]#指定需要添加的xdc文件，glob是扫描某个路径下的全部文件（这
read_edif ./test.edif#指定需要添加的网表文件
read_ip ./CORE/DMCM/DMCM.xci#指定需要添加的xci IP文件
# 运行综合设计，同时设定相关综合参数
synth_design -tcl SCRIPT_TEST                \
             -part xc7a100tfgg900-1          \
             -fanout_limit 1000              \
             -shreg_min_size 5               \
             -flatten_hierarchy full
opt_design #优化设计
place_design #布局
route_design #布线
write_bitstream -force                    /SCRIPT_TEST/bbc  #生成bit文件
```

图 2　tcl 脚本文件模板

Fig. 2　tcl script file template

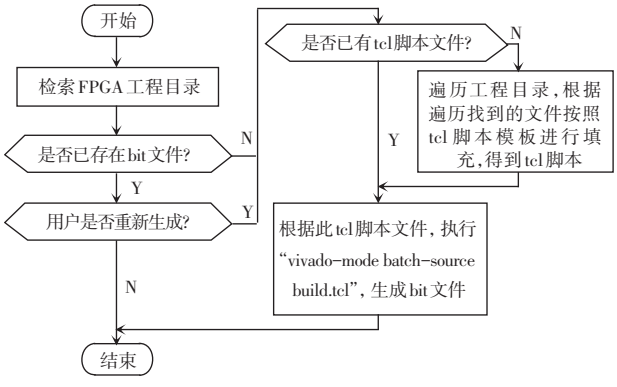

图 3　tcl 脚本及 bit 文件生成函数流程图

Fig. 3　tcl script and bit file generation function flow chart

2.2　tcl 脚本的自动生成设计

本次研究中，相对重要的是 tcl 脚本的自动生成。图 2 已经给出了 tcl 脚本的模板。在此基础上，关于 tcl 脚本的自动生成算法流程可推演论述如下。

(1)采用深度优先搜索算法遍历工程的工程目录。

(2)遍历到文件时，根据文件的后缀名，判断文件的类型，填充至模板的相应位置。*read_vhdl -library* 是添加 vhdl 库文件；*read_vhdl* 是添加 vhdl 文件；*read_verilog* 是添加 verilog 文件；*read_xdc* 是添加 xdc 文件；*read_edif* 是添加网表文件；*read_ip* 是添加 xci IP 文件。这些指令后面续接的参数都是文件所在的绝对路径。而还有一种 *glob* 指令是添加指定目录下的所有指定后缀名的文件。

(3)文件遍历完成后，根据该 FPGA 工程创建时所选择的芯片种类，设置综合设计中的有关指标参数。之后就是填充优化设计、布局、布线等指令。在 tcl 脚本的末尾再加上 bit 文件的指令。

至此，研究进一步给出 tcl 脚本自动生成的函数运行流程可如图 4 所示。

该部分设计的按钮只对 FPGA 工程有用，点击该按钮时，需要判断当前选中工程的类型。但是综观本次集成开发环境中，除了 FPGA 的工程类型是自定义的新的工程类型，其它工程类型都是基于 C/C++工程或者 Java 工程的，因此只是根据工程类型即可判断是否为 FPGA 工程。

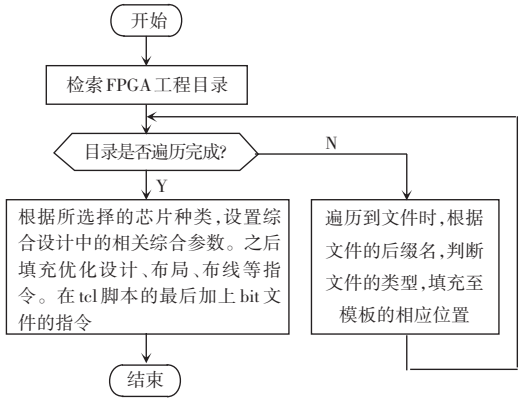

图 4　tcl 脚本生成函数流程图

Fig. 4　tcl script generation function flow chart

2.3　tcl 脚本的执行设计

在执行 tcl 脚本时，通过 Java 的 *Process* 类实现。具体执行步骤如下。

(1)通过 *ProcessBuilder*( ) 函数新建一个 *builder* 对象，参数为要执行的第三方可执行程序的绝对路径。

(2)通过 *builder.start*( ) 执行该第三方程序，获得一个 *Process* 进程对象 *process*，*process* 即为打开该第三方程序后的进程。

(3)通过 *process.getOutputStream*( ) 获得该进程的输入流，通过该输入流执行相应的指令。指令代码可参见如下。

new *BufferedWriter* (new *OutputStreamWriter* (*process.getOutputStream*( )));

// 初始化一个 *BufferedWriter* 对象 *br*

之后通过执行函数 *br.write*(*cmd*)，参数为要执行的程序指令，即可通过该输入流向第三方可执行

程序发送指令。

(4)使用 *WorkspaceJob* 创建一个新的线程,用来监听 *process* 进程的输出流。

(5)当监听线程开始执行之后,会监听第三方可执行程序的输出流。这里使用了一个 while 循环,循环的终止条件就是标准输出流和错误输出流均为空。

(6)当标准输出流和错误输出流都为空时,关闭这 2 个输出流,同时刷新工程所在的工作区间。

刷新工程所在的工作区间的原因在于,在调用第三方可执行程序时,有时会产生一个文件结果,例如 FPGA 工程编译结束后会形成一个 bit 文件。如果不执行刷新操作,则 bit 文件不会直接显示在 Eclipse 工程目录下,需要用户手动刷新。因此为了用户友好性的研究需要,这里就在每次调用了第三方可执行程序后都会针对工程工作区间采取自动刷新的解决办法。

Process 类调用第三方程序执行,获得执行结果的通用函数设计流程则如图 5 所示。

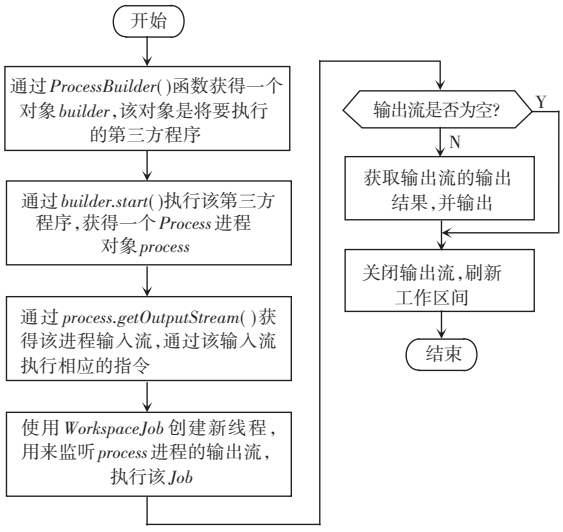

图 5　执行 tcl 脚本函数流程图

Fig. 5　Function flow chart executed by tcl script

3　结果测试

综合前述的研究设计可得,最终的测试效果将如图 6 所示。

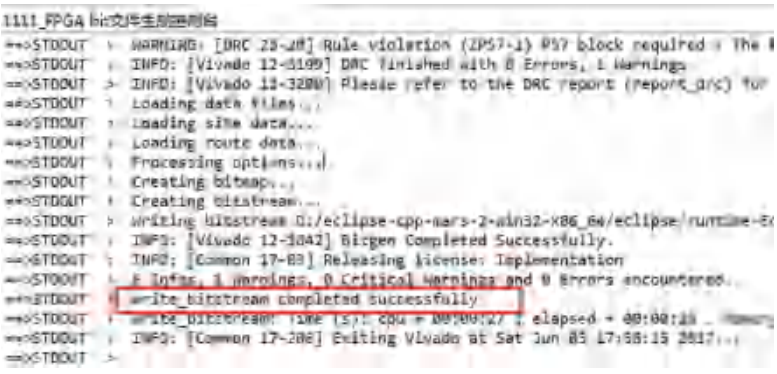

图 6　测试效果图

Fig. 6　Test result chart

从控制台的输出可以看出 bit 文件已被成功生成。

4　结束语

本文在 Eclipse 的扩展节点上,研究建立了 FPGA 工程,同时提供了一个功能按钮,通过深度优先搜索算法对工程目录进行遍历,按照模板填充生成 tcl 脚本文件,最终通过命令行方式调用 Vivado 软件中的相关功能,实现了 FPGA bit 文件的生成工作。

参考文献

[1] 田丹, 林卓, 卫进. 基于 Eclipse 的嵌入式集成开发环境工程管理[J]. 微处理机, 2015(2):29-31,34.

[2] 朱娜. 基于 Eclipse 的嵌入式软件开发管理平台的研究与实现[D]. 成都:电子科技大学, 2011.

[3] 南方. 基于 Eclipse 的嵌入式集成开发环境分析与设计[D]. 西安:西安电子科技大学, 2009.

[4] YANG Penghao, WANG Rongliang, FAN Ziguo. Study on debugging method based on embedded system development platform[J]. Advanced Materials Research, 2013, 2385(694): 2646-2650.

[5] HE Huiqin. Application research of JTAG standard based on ARM debugging system[J]. Applied Mechanics and Materials, 2015, 3749(719):522-526.

[6] STOLLON N. Multicore Debug[M]// MOYER B. Real world multicore embedded systems. USA: Elsevier Inc,2013:561-602.

[7] DAN U C. Software development tools for embedded systems [M]//OSHANA R, KRAELING M. Software engineering for embedded systems - Methods, practical techniques, and applications. USA:Elsevier Inc,2013:511-562.