一、引言

本文共有六个章节，各章节概述如下。

第一章引言即本章，主要描述相关的背景内容，包括FPGA测试的概念，国内外各类测试平台概述，。

第二章整体框架介绍，主要描述本文所在的BR0101测试平台的整体框图，采用的，包括个。

第三章后端设计实现，主要描述本文使用的服务器后端运行的相关软件系统组件。

第四章前端设计实现，主要描述本文使用的浏览器前端运行的相关软件系统组件。

第五章实验结果与分析，主要描述波形可视化系统与BR0101测试平台整体软件系统的整合，对若干个测试用例进行的波形绘制、控制与测量过程，以及相应的性能指标的测定与提高。

第六章总结与展望，总结本文工作的贡献与不足，以及未来的改进方向。

二、整体框架介绍

三、后端设计实现

本章是对本文设计的波形处理与分析系统的后端设计部分的描述。BR0101测试平台的服务器后端软件系统是一个基于Django后端框架构建的系统，包括用户模型管理、测试任务调度、硬件驱动、测试结果生成等。本文的工作是在已有的软件系统基础上，设计波形绘制与分析所必需的解析器代码模块，以及相关的服务器接口模块，并利用Python语言的模块化特性，整合进BR0101测试平台的服务器后端软件系统。

1波形数据格式概述

1VCD格式的波形数据

BR0101测试平台产生的波形数据以VCD（Value Change Dump，值变转储）格式保存在服务器中。VCD格式是根据1996年[IEEE](https://en.wikipedia.org/wiki/IEEE)制定的1364标准中规定的一种转储文本格式，它将波形的变化状况以ASCII文本形式记录下来。一般的，一个完整的VCD文件包含三个模块：头模块，变量定义模块和值变模块，各个模块之间的划分是根据特定的关键字实现的。头模块包含timescale等与测试相关的背景信息，变量定义模块为每个信号定义了一个代表符号，值变模块则记录了各个信号具体的变化历史，其中模块首部的$dumpvars关键字提供了各个信号的初始值，而总线信号的值含有前缀字符b。

一个完整的VCD文件示例如图所示。左中右三图分别对应头模块，变量定义模块和值变模块。根据实际波形绘制的需求，将该VCD文件所包含的波形信息整理如表。

表

|  |  |
| --- | --- |
| 模块名 | 信息 |
| 头模块 | timescale=1ns |
| 变量定义模块 | 定义有clk和en两个信号 |
|  | clk信号：初值0，10ns为1，20ns为0，30ns为1，40ns为0 |
| 值变模块 | en 信号：初值0， 30ns为1 |
| 测试向量总长度50ns |

将该表格中的信息适当整理，即可形成适合于浏览器前端绘制流程的波形数据。解析VCD文件采用的算法流程将在节中叙述；算法的输出是一个JSON数据包，将对此格式做进一步的说明。

2JSON格式的波形数据

经过VCD文件解析算法处理的数据是以JSON数据包的形式发送回用户端浏览器。JSON格式是一种轻量级的存储和交换文本信息的格式。在语法上，JSON使用了与创建 JavaScript 对象相同的键值对语法结构，使得JSON具有良好的可读性。目前包括Python、JavaScript在内的多种语言均允许JSON作为一种输入输出的格式，使得JSON具有良好的跨语言移植性。

根据本文的整体框架，解析出的JSON数据包应当包含以下两方面的内容：一是VCD文件包含的信号，二是各个信号的变化历史，即对应于VCD文件中的变量定义模块和值变模块。VCD文件中其他的信息与波形绘制和分析的关系不密切，在解析时可以丢弃。使用本文提出的算法将节中的示例VCD文件进行解析，得到的目标JSON数据包示例如图。该图中的波形信息与表的波形信息等价。

2VCD文件解析算法概述

根据VCD文件的格式特点，可以编写一个解析器提取其中的波形变化数据。在的讨论中，已经知道VCD文件是由三个模块组成，而每个模块的格式略有差异，因此解析器可以分解成三个相对独立的子解析器，由子解析器分别完成对应模块的解析工作。本节即围绕三个子解析器的设计展开讨论。

1头模块子解析器设计

头模块无论是其规模还是信息量都相对较小，故对其进行解析的流程较为直接。由图可知，头模块包含若干个关键字，不同关键字对应了相应的信息，其中只有$timescale关键字下含有有效信息。故设计如图所示的解析的流程。

此流程中，解析器逐行读取VCD文件，直到遇见$timescale关键字；$timescale关键字的下一行保存有timescale数据，可以提取出timescale的时间值与单位；最后，解析器继续读取VCD文件，直到进入变量定义模块，头模块子解析器退出。

2变量定义模块子解析器设计

变量定义模块是介于$scope关键字和$enddefinitions关键字之间的内容，其中被定义的变量均通过$var关键字引导的变量定义行完成。因此该模块子解析器的要点是正确提取变量定义行包含的内容。通过观察，可以看出变量定义行具有固定的格式，出于简便起见，本节使用BNF范式描述变量定义行的语法如下。

<变量定义行> ::="$var" <信号类型> <位宽> <信号符号> <信号名> "$end"

<信号类型> ::= "reg"|"wire"|"parameter"

<位宽>::= 数字|[数字]

<信号符号> ::= ASCII字符|[ ASCII字符]

<信号名> ::= Verilog标识符| ["["总线宽度"]"]

<总线宽度> ::= (数字":"数字)

根据以上语法描述，变量定义行是规定了信号名与符号之间的一一对应。在Python语言中，一一对应关系可以通过字典这样数据结构表示，因此只需要建立一个字典，以信号名作为键，符号等其他信息作为值。解析的流程如图所示。

对每一个变量定义行的分析步骤都是统一的。由于变量定义行的各个组成部分是通过空白符分隔，使用Python字符串处理函数split()可以直接将变量定义行划分成相互独立的数据片段，再根据变量定义行的语法将数据片段对应到各个组成部分。最后，向字典中添加新的键值对，用于值变模块子解析器的数据查询。

3值变模块子解析器设计

值变模块是VCD文件的主体，其解析器的设计也较为复杂。同样使用BNF范式描述其语法如下。

<值变模块> ::= <值变单元> |[<值变单元>]

<值变单元> ::= <时间行> |[<值变行>]

<时间行>::= "#" 数字|[数字]

<值变行> ::= [<总线标记>] <信号值> [空白符] <符号>

<信号值> ::= ("0" | "1" | "x" | "z") | [<信号值>]

<总线标记> ::= "b"

<符号> ::= ASCII字符|[ ASCII字符]

根据以上语法描述，值变模块是若干个值变单元组成，每个值变单元包含一个时间行和若干个值变行，表示在时间行所示时刻，值变行所涉及到的信号已发生了跳变。值变行则保存了信号跳变后的新值，总线与非总线信号的值变行的格式略有差异，体现在总线标记的有无，但不影响对值变行的解析。解析的流程如图所示。

解析值变模块实质上是解析每个值变单元。

在值变模块解析完成后，可以根据解析得到的字典对象，生成最终的JSON数据包。由于字典对象中含有冗余信息，在JSON生成步骤之前，尚需去除冗余，以提高传输效率。显然，信号名与信号符号在VCD文件中是等价的，但是波形绘制与分析等前端功能不需要信号符号；非总线信号的状态分类值与具体信号值也是等价的，两者取其一即可，但为了减少前端设计的代码冗余度，保留了状态分类值。将上述冗余通过del语句从字典对象中去除

接口实现

本章叙述的解析器代码模块并不是Django框架所必需的模块。换言之，对Django框架而言，解析器代码模块是一个独立的第三方模块，它要与Django框架产生联系就需要通过Django框架内的标准调用此模块完成。因此需要设计一个封装有解析器代码模块的接口模块，由此接口完成与Django框架的关联。

在Django框架规范中，views.py这一文件集中了各类后端业务逻辑，每个具体的处理函数应当定义在或导入到该文件中，且必须接受一个名为request的参数。与此同时，这些处理函数还需要在urls.py中绑定到一个url，当浏览器前端通过某个url向后端发送请求时，Django框架就可以调用对应的处理函数。请求的内容则通过request参数传递，该参数是一个Python字典，包含请求的类型和请求参数。

本节实现的接口模块的处理流程如图所示。该流程中从浏览器前端获取的数据是待解析的VCD文件的文件名，为了传递该文件名，采用环回手段，即在Django框架中将当前文件名渲染入前端页面，前端页面校验后通过AJAX请求发回后端Django框架，其原因将在节中说明。

四、前端设计实现

本章是对本文设计的波形处理与分析系统的前端设计部分的描述。实现波形可视化功能并不一定需要复杂的前端设计，但是为了使波形可视化系统具有一定的交互性，需要引入前端行为；波形可视化系统还需要具有较为整洁美观的外观，这可以通过适当的样式设计完成。本文的工作是设计波形绘制与分析所必需的前端行为与样式，以及相关的服务器数据传输模块，并利用Django框架的模板渲染机制，整合进BR0101测试平台的服务器前端软件系统。

1前端设计工具简介

前端设计工具主要包括HTML（），JavaScript，CSS（），它们分别控制前端页面的结构，行为与样式。为，可以以，因此，

1HTML

是通过标签

2 JavaScript

3 CSS

2设计实现

1波形绘制功能实现

波形绘制是波形可视化功能的核心。在HTML前端中，实现绘图的方式丰富，其中较为成熟的方法是利用canvas元素提供的绘图功能实现。

canvas元素是在HTML页面上插入一个动态的绘图区域，通过JavaScript脚本控制区域内的图案生成。就本文实现的测试波形绘制而言，思路是较为直接的：根据欲显示的时间范围，查找待绘制的波形状态，通过描点连线完成绘制。相应的处理流程如图所示。

1波形控制功能实现

实现了波形绘制功能，但是仅有绘制功能并不能体现出Web可视化的优点，需要引入具有交互的波形控制功能以充分利用前端的灵活性。，为了波形控制功能要，

于使用

五、实验结果与分析

本章是对本文设计的波形处理与分析系统的实际测试。进行的测试包括两方面：一是通过与BR0101测试平台的测试流程整合联动，检验系统功能的正确性与完备性；二是通过处理若干个标准的测试样例，获取系统的性能指标，并进行一定的优化。

1. 系统与BR0101测试平台的整合

本文设计的系统包含前端和后端两部分，需要分别整合进入BR0101测试平台。

进行后端整合时，由于Django后端框架是使用Python语言编写，而Python语言是模块化的，只需要将新的后端处理代码保存在相应文件夹下，使用import语句将其作为一个子模块导入即可完成整合，对既有流程的影响降到最低。

进行前端整合时，根据Django后端框架的规范，HTML文件应当保存在template文件夹中，而JavaScript脚本和CSS样式表应当保存在static文件夹中，按此规范提交相关前端文件后才能被Django框架识别。由于Django框架是利用模板渲染机制来获取保存在static文件夹中的文件，相应的需要在HTML文件添加模板渲染的占位符。

图为整合后的测试APP的界面。。

BR0101测试平台内部集成有一个波形处理绘制模块，其处理流程与本文的工作有相似之处，整理如图。该模块与本文提出的系统的比较如表。

图

|  |  |  |
| --- | --- | --- |
|  | BR0101原有模块 | 本文系统 |
| 波形结果生成场合 | 进行测试时 | 用户打开相应页面时 |
| 信号数据表示方式 | 每个时刻的信号值均保存 | 仅保存信号跳变时刻 |
| 后端向前端发送的数据 | 图片文件本身 | JSON数据包 |
| 波形结果的呈现形式 | JPG格式图片 | canvas元素 |
| 波形结果的交互性 | 无 | 有 |
| 波形结果能否测量 | 否 | 能 |

可见，原有的流程具有冗余数据多、数据传输量大、灵活性差的缺点，本文提出的流程则以延长一定的前端加载时间为代价，部分克服了以上的缺点。表是对个参数样例分别按照新旧两种流程运行时，波形结果显示完毕所需的时间，可以验证以上的分析。

|  |  |  |
| --- | --- | --- |
| 波形结果显示耗时 | BR0101原有模块 | 本文系统 |
| AccuShift |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

2后端性能分析

3前端性能分析

本文的波形处理系统具备了基本的交互性，很自然的，需要对其进行性能分析与优化。

六、总结与展望

本文

参考文献

致谢