“贴心云衣柜——基于机器学习和社交网络的服装搭配推荐”结题报告

计算机科学与技术系，2016级，李鑫烨，曹佳涵，金亦凡，李光耀

指导老师：卜磊，副教授

**摘要：**本项目4名成员均为计算机系2016级本科生。指导老师为卜磊副教授。当下穿衣搭配是一个很热门的话题，本项目开发了一款基于机器学习和社交网络的服装搭配推荐APP——贴心云衣柜，它可以为用户生成个性化的穿衣搭配，还可以通过对相似风格的用户间进行相互推荐。

**关键词：**Android；服装搭配推荐；机器学习；协同过滤；社交网络

# 选题背景

## 背景

在集成电路设计的电子设计自动化领域中，Verilog是一种用于描述、设计电子系统（特别是数字电路）的硬件描述语言。Verilog代码能够被用于逻辑仿真，逻辑综合等等，被广泛应用于硬件系统设计、生产流程中。在依据设计生产具体硬件电路之前，首先要确定相应设计的正确性。否则，在设计不正确的情况下进行流片，会造成人力物力的巨大浪费与损失。针对相应问题，传统的检测错误的方法是进行枚举测试，尽可能将所有情况枚举出来。然而，相关手段通常不能覆盖所有情况。

与传统测试不同，本项目基于模型检验的手段去对Verilog代码的正确性进行自动化证明，在用户完成设计之后验证其行为是否与预期相符，并在不正确时给出反例帮助用户进行修正。在完成相关技术的基础上，本项目将开发面向Verilog的一体化验证工具平台，该平台将可直接读入Verilog代码及用户规约，自动验证并在发现错误的情况下给出反例路径。结合本系数字电路课程所学内容，本项目也将为同学们理解并掌握Verilog提供有力支撑。

## 知识储备

项目开始前，我们已经系统地学习了《数字电路与数字系统》课程。能够使用卡诺图化简逻辑表达式，用Logisim绘制和仿真电路图，用面包板做实验对该电路图进行布线和实验测试。然而这一过程太过底层且非常麻烦。甚至对于简单的电路都需要重复进行多次实验才能得出正确的结果。而一次布线的过程必须全神贯注做几个小时，稍有不慎就需要重做。

我们注意到，做这件事情太容易出错，如果工业界直接使用门电路和面包板布线的方式进行生产，其生产效率一定很低且不能保证正确性。通过搜索相关资料，我们了解到硬件描述语言解决了上述问题。如Verilog主要用于在集成电路设计，特别是超大规模集成电路的计算机辅助设计中。

另一个问题随之到来，Verilog虽然大大提升了生产效率，但Verilog代码的正确性又如何验证？我们对这个问题产生了兴趣。大二上学期（项目初期）的《数字电路与数字系统实验》课程上学习了Verilog语言。检验Verilog的正确性通常使用仿真测试的手段，这种验证方式需要再写一段Verilog代码，穷举所有可能情况，观察各个变量的电位变化图，分析某时刻的电位是否是“正确”的。即，通过比对和逻辑分析来检验自己的设计是否正确。

针对这一问题，我们与导师进行了讨论。在软硬件的验证领域，形式验证是检查设计是否满足某些要求（属性）的过程。[[1]](#_[1][2]_IEEE_P1364-2005:)形式化验证能够直接对用户规约进行验证，输出正确或者错误。形式化验证已经广泛应用在许多对正确性要求高的领域。老师给我们指明了一条解决此问题的技术路线：将Verilog代码建模为有穷状态机（Finite-state Machine，以下简称FSM），将FSM转换为现有的模型检验工具输入语言（如NuSMV[[2]](#_[2]_NuSMV_home)），使用工具验证并得到结果，分析结果并反馈给用户有用信息。

这一技术路线需要掌握的知识和理论有：编译原理、有穷状态机、模型检验等等。这些也是项目中需要学习的知识。

# 项目情况

## 成员组成

项目成员均为计算机科学与技术系2016级本科生。

李鑫烨：项目负责人，动手能力较强，擅长做实验和协调各成员的工作。主要负责学习算法决策树的实现和各个模块的连接、督促成员的工作和联系导师等。

曹佳涵：执行力强，注重理论学习和动手实践相结合。主要负责推荐算法协同过滤的实现，以及后端数据库的搭建。

金亦凡：具有很强的创新思维。主要负责客户端架构，前端界面及各功能模块的实现。

李光耀：思维清晰，知识面广。主要负责前端功能实现和客户端与服务器端的socket通信。

## 导师指导情况

项目初期，卜磊老师给我们介绍了当前APP Store的内容在穿衣个性化推荐上的缺失的现状，在此基础上，他提出了关于这个APP的构想和最终功能的预期，并提供给我们完成该项目所需要自主学习的相关知识，给我们初步定下了项目的目标。

项目进展过程中，老师时常关注着项目进展，并安排了一位研究生学姐为我们提供帮助。大二上学期末，在老师的指导下，我们进行了模块划分和分工。中期答辩前，老师对我们的答辩材料给出了非常有用的建议。中期答辩后，老师为项目接下来的进展指明了方向。

结项答辩前，老师对项目成果进行了评估，并就结项事宜进行了详细的说明和指导。

整个项目过程，老师非常认真和仔细地提供帮助和关心，使得项目能够顺利地进展。

# 创新特色

## 创新方法

### 模块分解

本项目的是一个需要用代码实现的项目。项目涉及到众多的模块，如果不进行分解，会造成实现复杂度很高，代码可维护性低等缺点。遵循模块分解的原则，我们分解了几大相互独立的模块：如抽象语法树生成、FSM生成、NuSMV输入语言生成、输出结果处理、图形化界面等等。这些模块之间的使用接口或某种数据结构进行耦合。模块分解的使用使得项目化繁为简，分工明确，目标清晰。

### 功能再分解

模块分解使得模块间高内聚、低耦合，但项目如果迟迟没有成果会打击成员们的自信心。我们不能够让每个模块都分别实现了才进行模块连接测试，也不可能这样做，因为有的模块的输入数据需要前一个模块提供，手动编写输入非常麻烦且容易使得项目成员联系松散。我们把模块内功能再分解，标出哪些功能是易实现的，那些功能需要仔细斟酌。编写恰当的测试用例，使得程序能够尽快地跑起来。这样使得大家有足够的信心加入更多的功能。

### 转换关系研究

项目需要研究许多转换关系：Verilog -> 抽象语法树 -> FSM -> NuSMV输入语言。除了Verilog到抽象语法树有编译原理等理论支持外（但数据结构也需要自己设计），其他的转换关系都需要探索，没有直接可用的资料。这时候正确性显得尤为重要。如何设计转换关系才是正确的？首先，需要阅读相关手册和文献，了解其工作流程和原理。从测试用例的角度讲，测试用例通过且符合预期就可以初步判断其为正确。然而过程中我们也遇到了很多困扰。例如使用NuSMV的INT CONSTANT作为相应FSM中变量的转换关系，造成的问题有溢出时无法自动进行模运算（超出寄存器的位数限制舍弃高位）、溢出时会被判定为死锁等。使用WORD则更加符合寄存器传输级的转换逻辑。又比如NuSMV提供的状态转移语句有ASSIGN和TRANS，初期我们使用TRANS，能对组合逻辑有很好的支持，而后来发现它对时序逻辑、多always语句支持很差，几乎不能扩展，后改为ASSIGN。这些转换关系的研究需要创新思维，也锻炼了创新思维，提升了解决问题的能力。

### FSM的中间层表示

上述转换关系中，可以发现从抽象语法树到NuSMV输入语言有中间过程FSM。FSM是否可以去掉？项目开始时，我们就此问题进行了讨论。如果没有FSM中间表示，直接遍历语法树也可以生成NuSMV输入文件。然而这样表示方式缺乏普适性且增大了模块耦合度，如果使用其他检验工具，其转化规则是另外的一套，不具有可移植性。在编译原理中，编译器不会将某种语言直接转化为机器码，使得编译器可以进行更多的分析和优化，做出更多的产品（比如clang可以用来做多样化的代码分析，而不仅仅是编译）。初期我们直接将抽象语法树转化为NuSMV输入语言，获得了程序的一个原型，再次证明了项目的可行性。后来全面转换到FSM模型，产生中间产物，如可用于数据流分析的状态转移图。

## 科学意义

本项目解决的问题和解决问题的思路可以推广到多个领域。我们研究了Verilog的正确性检验工具。同理，相似的硬件描述语言如VHDL也可以通过这个方法进行建模。甚至可以对软件代码也进行类似的检验。

目前，Verilog代码的分析流程大多为仿真测试。我们提供了一种新的思路，编写了相应的转换规则。为形式化检验Verilog代码的提供支持。

项目不是基础课上我们做的实验，只要按着老师讲的步骤做就行了，其本身具有很强的创新性，过程中锻炼了人的创新能力、阅读能力、研究能力、思维能力、动手能力。

## 应用价值

我们对Verilog代码进行自动建模，生成等价有穷状态机模型，并基于相关模型进行自动验证。这个过程中的许多中间成果都有其应用价值，如FSM模型，可以用来演示一个Verilog代码的状态转移图。又如正确或者错误的提示，简单明了地告诉用户这样做是否可行，或者提示修改，可以提升数字电路课程同学们对Verilog的掌握和理解。

# 进展成果

## 进展情况

项目进展可分为：可行性分析、制定计划、执行计划和收尾整理阶段。

可行性分析：2016年9月到11月，这个阶段收集了大量相关资料和文档，学习相关理论和技术。最终分析出项目可行方案：建模Verilog代码生成FSM模型，转换该模型为NuSMV模型检验工具输入语言，输出用户规约是否得到满足。在这个阶段中，卜磊老师给了我们很多指导和关心，让我们有解决问题的勇气和信心。

制定计划：2016年11月到12月，此阶段对项目进行了模块划分，制定各个模块对外接口、可用的数据结构并确定分工。主要模块有：提取抽象语法树、生成FSM、转化FSM、模型检验、反向路径追踪和功能整合与呈现。我们对未来的时间进行了具体安排：在2017年2月完成常见的词法语法解析和转换，3月到6月，基本实现完整功能并能正确运行，7月至9月实现图形化界面方便用户使用，10月至11月整理资料。

执行计划：2017年1月，完成抽象语法树的提取工作。1～3月，研究FSM到NuSMV转换规则，这个过程比想象中复杂艰难，使得进展比计划的缓慢。我们实现了基本组合逻辑的转换，比如单模块、if、case、输入输出、表达式等。将模块封装成命令行窗口的验证平台。4月，成功通过了阶段性考核（等同于中期考核）。5月～9月，研究了许多之前难以解决的问题，如时序逻辑、模块实例化、多always语句、数组引用等。设计了合理的解决方法，优化了FSM决策树模型，修改代码框架，至此，基本功能已经完成。9～10月实现了一体化验证平台图形化界面。10～11月，提供反向路径给用户提示可能出错的地方。

收尾整理：11～12月，整理文档，准备结项答辩。

## 创新成果

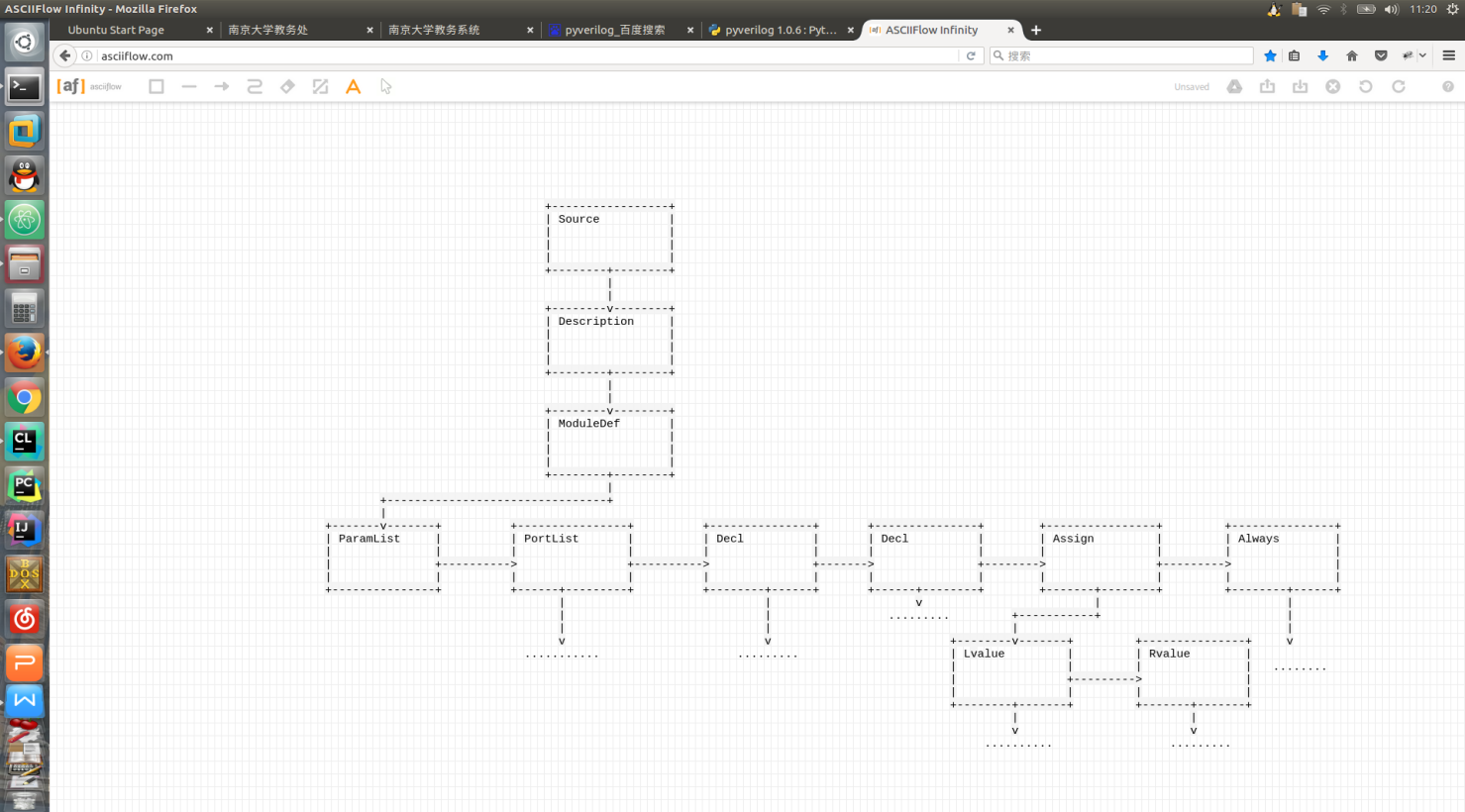
总结：我们实现了Verilog代码形式化检验的一体化平台。用户输入Verilog代码和规约文件，程序对输入代码进行建模，生成FSM模型。将该FSM模型转换为模型检验工具NuSMV输入文件，进行模型检验。程序输出Verilog代码是否满足规约，若不满足，给出一条反例路径，并反向追踪到Verilog相关代码，给出相应的推断和提示。

在项目研究过程中，我们也取得了一些阶段成果，现列举如下并作简要说明：

1) 形式化验证理论和工具的学习。形式化验证可以保证软硬件设计的正确性，在对错误容忍度比较低的领域有广泛应用。我们学习了NuSMV模型检验工具，为了更深入理解NuSMV的工作原理，我们仔细地阅读手册并将NuSMV的输入语言部分进行了翻译，供内部参考使用，也当做学习笔记。



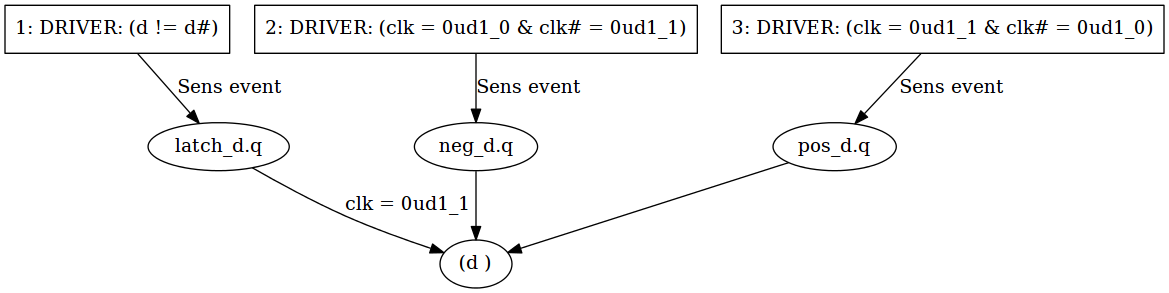
2) 抽象语法树的生成与表示。语法树是程序中最重要的数据结构之一，后续代码的编写依赖于该语法树，如果语法树存在二义性、不完整性、结构不规范性都会导致后续步骤的错误或处理上的麻烦。我们将Verilog代码转换为更易于处理的格式（借助工具PyVerilog[[3]](#_[3])），使用了子女-兄弟链的表示方式，同层的语句块在兄弟链中，子语句块在子女链中。根据Verilog语法规则，划分语句块主要使用begin和end标记，另外module、always、initial也会产生语句块，导致规则非常复杂，数据结构设计并不简单。下图为一颗Verilog源代码的抽象语法树结构示意图：



3) 有穷自动机的建模与生成。有穷自动机是一种表示有限个状态以及在这些状态之间的转移和动作等行为的数学模型。Verilog语言可以用有穷状态机表示。我们通过对Verilog代码建模，生成其有穷状态机模型。该模型是下一步研究的基础。这个过程也是艰巨的，因为直接面对Verilog的语法语义规则，生成等价的FSM模型，需要直接处理各种语法树结点。

4) NuSMV输入文件的生成。NuSMV是一种模型检验工具，其输入语言具有特定的语法。我们研究了NuSMV语法规则及其对应的FSM表示。将FSM模型转换为NuSMV输入文件，并验证用户规约能否得到满足。这个转换过程涉及到NuSMV的语义规则，也是困难的。比如，上升沿和下降沿的敏感事件，不能使用单个变量在NuSMV中表达出来，我们使用两个变量来表示对应Verilog中该变量的状态变化情况。

5) 生成Verilog状态转移图。FSM的中间表示使得程序更具有可扩展性。我们学习了Graphviz语法，利用FSM生成Graphviz输入文件并最终转换为状态转移图。如下图是锁存器、下降沿D触发器和上升沿D触发器的状态转移图。



6) 反向路径追踪。反例可以让用户知道程序存在错误，但需要进行一定分析才能得知错误的地方。通过反向路径追踪，直接定位在Verilog代码上，提示该行语句将会造成规约不能满足，使得程序的验证结果更加直观。

7) 图形化界面。中期时使用的命令行，界面不太友好。我们设计了一个可选的简洁的图形化窗口，可直接输入Verilog代码和规约，一键验证程序正确性，默认查看验证结果，可查看9个中间处理过程。



# 收获体会

## 总结

项目给我们许多共同的体会，比如要善于勤于思考、动手动脑、多沟通协作、多阅读文献和资料。也让我们学到很多东西，除了项目相关的理论和技术外，我们还学到了方法，比如模块划分、确定分工等等。

## 各成员体会

### 唐瑞泽

作为一名负责人，我感觉到压力是很大的。开始项目时，我们对该领域一无所知，对未知事物的恐惧充斥在整个团队中。但我认为创新项目的创新就是要不断地对未知领域进行探索，在这个过程中知识和能力上都能得到提升。

项目进展中也遇到了很多困难。比如答辩，站在讲台上面对专家和听众，不可能不紧张。又比如沟通，我们有各自的分工，整合起来时发现别人的代码很难和自己的连接起来。以及对未知领域的探索，可能没有绝对正确的答案，也没有人告诉你该怎么做。

非常感谢卜磊老师对我们的关怀和帮助，使得项目进展的同时，我们也收获成长。

### 米佳玮

做项目的时候，最重要的不是自己如何快速地将自己分配的任务做完，而是要注重团队合作。一开始必须对这个项目的进行统筹规划，完成充分的理论建模。在技术实现上要定义好公共的接口和数据类型，如果一开始不将这些步骤统一，会给后面的实现带来一系列的问题。遇到问题需要与组内的成员进行交流，大家辛苦努力一起解决问题。

### 孙旭东

在整个项目实现过程中我负责的是最靠前的语法树的生成工作。整个工作过程中一个深刻的体会就是自己单人写代码和团队作业完全不是一个概念，看上去人多了，压力小了，但是事实上每个人稍微不注意自己的代码风格就可能严重影响他人的工作进展，拖慢整体进程，而且由于大家是分成一个个模块来实现，所以究竟什么函数什么变量该隐藏起来什么该暴露给队友使用都是值得深思熟虑的问题。再有就是动手实现之前一定要仔细思考清楚，不然很可能会因为自己先前的不成熟的想方法造成重复劳动。

### 万子文

创新项目是一个很大的工程，一开始感到很迷茫。在指导老师的帮助下，我们把创新项目拆分成了几个独立的模块，细分了每个模块的输入输出以及功能，模块化的划分使整个任务目标变得明朗很多，团队之间的分工也变得清晰。这让我深刻体会到在面对复杂问题时候，这种分解问题并逐个解决的想法是非常有效的。

在进行项目的时候，需要用到很多之前没接触过的工具和知识。在这个过程里，我们不得不借助网络来获取相应的知识。这个过程让我有了解决未知问题的信心。学习生活中，碰到的大量问题都是陌生的甚至没有头绪的，而闭门造车是很难解决问题的，更多的应该从别人的解决方案里借鉴别人的经验来解决自己的问题。

# 参考文献

### [1] Introduction to Formal Verification: Berkeley University of California, Retrieved November 9, 2017 <https://embedded.eecs.berkeley.edu/research/vis/doc/VisUser/vis_user/node4.html>

### [2] NuSMV home page: FBK-irst, CMU, The University of Genova. Retrieved November 9, 2017 <http://nusmv.fbk.eu/>

### [3] Pyverilog: Shinya Takamaeda-Yamazaki. A Python-based Hardware Design Processing Toolkit for Verilog HDL, 11th International Symposium on Applied Reconfigurable Computing (ARC 2015) (Poster), Lecture Notes in Computer Science, Vol.9040/2015, pp.451-460, April 2015. <https://pypi.python.org/pypi/pyverilog>