# 特征提取

对于机器学习项目，决定预测准确度的上限并不取决于模型的算法的高深或模型算法的先进性，而决定模型预测上限的是特征的全民性、准确性和样本。所以特征的选择和提取尤为重要。

本章主要阐述使用Verific的Parse-tree来对输入的RTL设计进行二次解析并提取所需要的信息作为后续机器学习的输入特征。

## Verilog HDL 语法特征分析

基于前一步骤实现的基于Verific的快速资源估算建模所获取的资源之间的映射关系，优化技巧等理论及实践基础，我们发现对于综合后技术映射的基本逻辑就是将Verilog HDL预计合理的使用FPGA上的器件来实现Verilog代码中所需要的数字逻辑，所以我们可以分析将Verilog HDL设计语言和器件间的典型关系，提取VerilogHDL中的语法特征，通过这些特征来构建机器学习模型，从而实现更加快速精确的大型FPGA设计所需要的FPGA片上资源数量，首先我们基于IEEE所给出的Verilog的技术文档，对VerilogHDL进行全面特征提取；

通过对Verilog HDL基本语法分析结合IEEE Verilog 标准、RTL样例设计，拟提取以下信息作为Verlog HDL语法特征：



Verilog语法类型与关键字

## 特征提取方法

首先我们通过第二章的综合及技术映射原理已知Verilog HDL语句的综合原则及对应的FPGA片上器件的映射规则，所以我们基于以上原理，借由Verific的Parse-Tree对当前读入的Verilog设计进行二次解析,**具体流程图如下：**

****

特征提取器结构图

通过解析Verific读入设计时构建的Parse-tree对所有Verific进行提取主要包含以下几类：

### 端口信息（input、output、）的提取：

首先对于FPGA上的IO资源是通过RTL中Module的端口映射而来，所以我们首先提取Module的端口信息，对于一个Module的端口信息在Verific的Parse-Tree中存储在VeriModule类下的VeriDataDeclar类下面，所以我们通过重载Verific提供的Visitor虚函数来获取端口数据类型指针，通过解析VeriDataDeclar下的数据解析出端口数量包括Input、Output、Inout，同时通过解析VeriDataDeclar下的VeriDatatype数据类型，获取对应的端口位宽，并存储为InputWidth、OutputWidth、InoutWidth，从而完成对Module端口特征信息的提取；

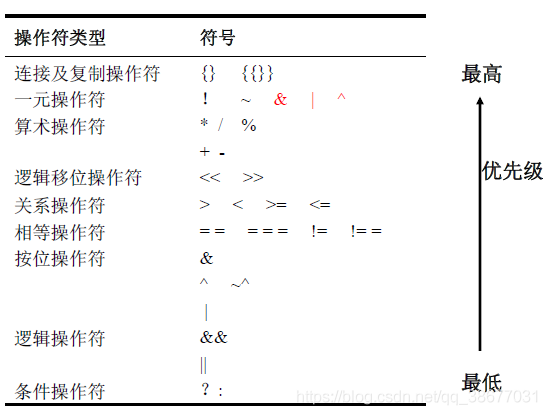
### 线网信息（reg、wire）的提取：

Verilog VHDL线网类型语法在RTL级描述中也是很重要的存在，基于第二章中描述的综合及技术映射理论可知，线网通常用于赋值语句左右，或被映射为寄存器结构，所以获取Verilog设计的线网类数据信息对于机器学习模型的训练而言应相对重要；

对于线网结构在Verilog HDL语法中主要体现为Wire和Reg，在Verific的Parse-Tree中线网结构被构造为VeriNetDecl类进行存储，我们通过重载Visitor基类来获取Module下的全部的VeriNetDecl类，通过解析可获得Reg和Wire的数量以及对应的位宽，分别存储为特征Reg、RegWidth、Wire、WireWidth，从而完成对Module线网信息的提取；

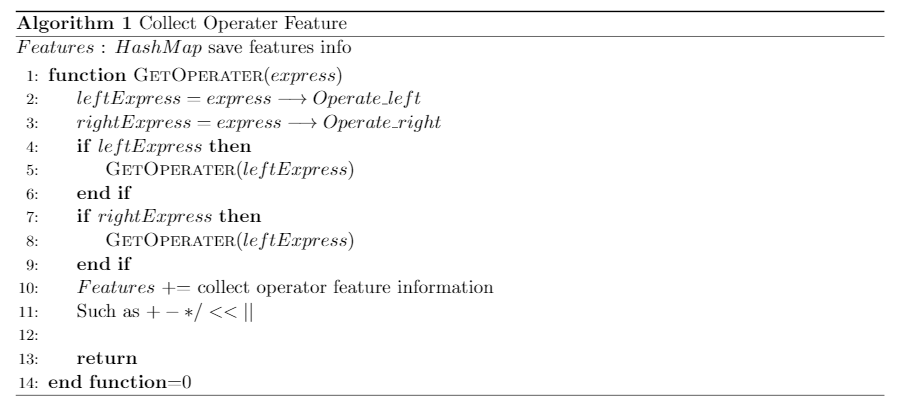
### 逻辑运算符、算术运算符、关系运算符表达式

在Verilog HDL语法中，表达式是将操作符和操作数联合起来都是用的一种Verilog HDL语言结构，通过运算得到一个结果，从语义上说，这个结果值也可以看作操作数的函数值；而操作运算符的类型与C语言类似，**如图所示：**



图

首先我们明确在逻辑运算符、算术运算符和关系运算符在进行逻辑综合后会被综合为相应的逻辑运算，所以符号表达式的特征信息也是关键特征，所以我们通过如下方式对表达式进行解析从而获得，首先整个表达式在Verific的Parse-tree中被存储在VeriExpression类下，通过解析VeriExpression类获得表达式存储类VeriOpenbinery类，通过判断VeriOpenbinery类下的Type类型获取运算符类型，如OR、AND、PLUS、AlterShift等，并存储至特征Map中，通过获取运算符左右两侧表达式，通过递归算法递归解析剩余运算符两次表达式，从而获取运算符两次表达式位宽，并存储至特征Map中，递归完毕后既可以完成对运算符表达式的特征提取。算法伪代码如下：

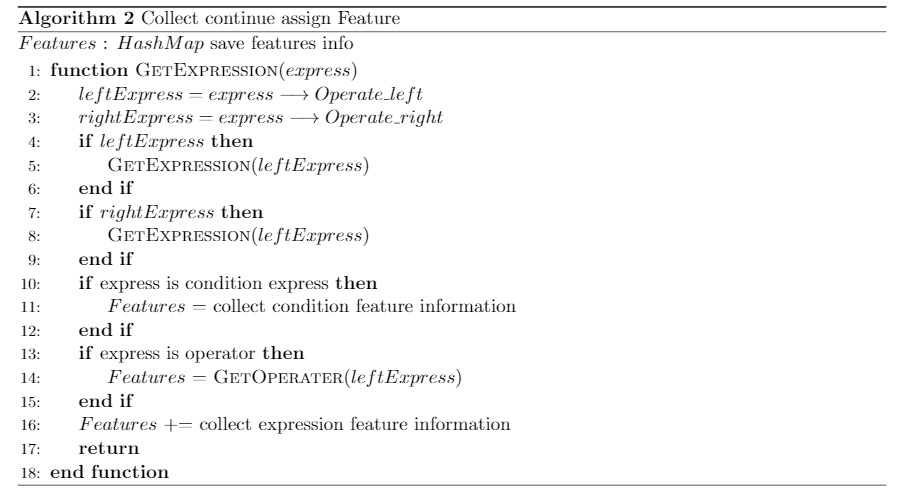


图

### 连续赋值语句（assign）的特征提取

连续赋值语句为Verilog HDL语法中比较重要的语句，常被综合为FF及LUT等关键器件，所以连续赋值语句的特征提取尤为重要，由于连续赋值语句常包含组合或嵌套式语法，如**下图所示，**通过IEEE标准中对Verilog连续赋值语句的定义可知，连续赋值语句左侧只能为线网结构即wire类型，而右侧可以包含逻辑运算符（&、|、~）、算术运算符（+、-、\*、/）、选择表达式（A?B:C）等多种语法嵌套组合；

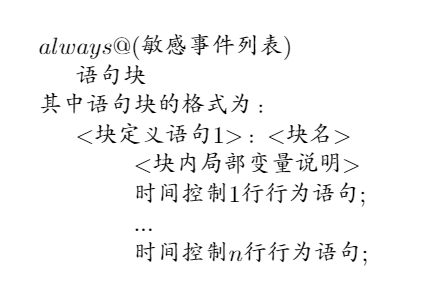
由于需要解析嵌套语法且种类较多，所以采用递归算法对表达式进行解析，首先根据=号左右获取AssignStatement，由于左侧只能为线网结构，所以将表达式左侧存储为AssignRightWidth用以存储assign语句左侧位宽，右侧进一步根据符号进行拆分，若存在逻辑运算符则存储逻辑运算符数量如AND（&）、or（&）等符号特征，并存储数量，则先以符号进行左右两侧拆分，若存在括号，则先进行括号匹配，对提取出的表达式进行进一步递归解析，同时计算符号两侧位宽，并保存为ANDRIGHTWIDTH、ANDLEFTWIDTH等；若表达式右侧为选择表达式，则存储选择表达式数量为QUESTIONCOLON特征，进一步解析该选择表达式分别提取if条件并存储为QUESTIONCOLONIF特征，若if条件存在逻辑或算术运算同理递归调用进行解析并存储至对应特征，else则存储至QUESTIONCOLONELSE特征，同时解析else的逻辑与算术运算，同时存储至对应特征中，then则存储至QUESTIONCOLONTHEN特征中，同时递归解析Then语句中的逻辑与算术运算并存储至对应的特征统计中；算法伪代码**如图所示：**



图

### Always过程块（always）

Verilog HDL语法中每个过程块是由过程语句initial或always和其内容语句块组成的，由于initial语句是不可综合语句，所以在该特征提取方案中不考虑initial过程块。其中语句块主要由过程性赋值语句（包括过程赋值语句和过程连续赋值语句两种）和高级程序语句（包括条件分支语句和循环控制语句）这两种行为语句构成，其语法格式如下，**如图**



图

由于Always过程块主要受敏感事件列表控制，敏感事件列表是由一个或多个表达式构成，当存在多个表达式时用or将它们组合起来。

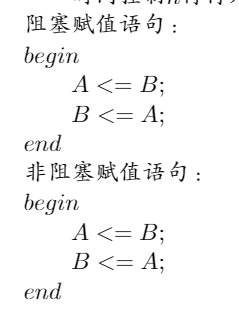
所以对于Always块的特征信息的提取，主要分为三部分：

1. 获取Always语法块数量
2. 获取Always语法块敏感事件列表、并解析宽度
3. 获取Always语法块内容（由于语法块中包含语法种类较多，将在后续着重叙述）

在Verific Parse-tree中always块构造存储在VeriAlwaysSatement类中，通过重载Visitor函数，来获取Parse-Tree中的Always结构体的数量，通过统计数量可以获得RTL设计中Always块的数量，通过解析获取到的VeriAlwaysStatement结构中的VeriEventExpress类获取Always的敏感事件列表，由于敏感事件列表由表达式构成，所以通过使用前述3）、4）所提及的递归方法来获取敏感事件表达式的宽度，并存储至特征AlwaysEventWidth中；从而完成对Always结构体数量与Always结构敏感事件列表的特征信息提取；

### 过程赋值语句

过程赋值语句的语法格式为**如图**：



图

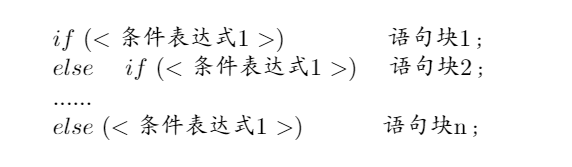
其中赋值操作符可以是“=”或“<=”之一，它们分别代表了阻塞型和非阻塞型赋值类型。过程赋值语句只能用于对变量型数据进行赋值操作，结合第二章的逻辑综合原理可知，赋值语句右侧的赋值表达式可以用以逻辑表达，从而综合为逻辑资源，所以过程赋值语句的特征信息提取也相对重要，由于过程赋值语句中的赋值表达式可以使任何合法的表达式，存在符号表达式等，所以我们在3）、4）、5）的递归提取方法的基础上对过程赋值表达式进行特征提取，提取特征类别分别为，阻塞型赋值语句数量BlockAssign，阻塞型赋值语句左右位宽BlockAssignRIghtWidth、BlockAssignLeftWidth、阻塞型赋值语句左右线网数量BlockAssignRightNum、BlockAssignLeftNum；非阻塞型赋值赋值语句NonBlockAssign、非阻塞型赋值语句左右位宽NonBlockAssignRightWidth、NonBlockAssignLeftWidth，其中符号表达式部分，将被存储在对应的符号表达式特征中提取过程与方法思路与连续赋值语句一致，可参见上图xxx。由于过程赋值表达式主要被用于过程块中，所以5中对语法块内容的提取将在该部分体现。

### If-else条件分支语句（if、else、then、case）

条件分支语句对于硬件的行为描述及其重要，且常被综合为FF和LUT器件，是一种高级语言的抽象表达，故在特征信息提取时，也是重点关注的语法；

在VerilogHDL中的高级程序语句，常用语过程块中，5）中已经提及过程块中常由两种普通的过程赋值语句及高级程序语句，高级程序语句就包含分支语句和循环控制语句，分支语句包含条件分支语句及case条件控制语句，该小节主要分析if-else条件分支语句的特征提取，case条件控制语句将在下小节进行分析。

通过Verilog HDL语法可知，if-else条件分支语句的作用是根据指定的判断条件是否满足来确定下一步要执行的操作。其主要语法形式如图：

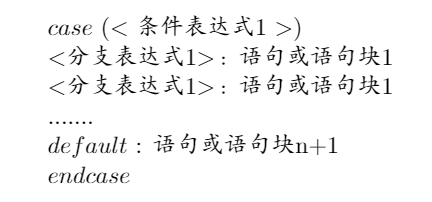


图

由于if-else条件分支语句常用语Always过程块中，所以我们在5）模块的基础上对if-else条件分支语句进行解析，在Verific-Parse-tree中，条件分支语句被存储为VeriQuestionColon类中，通过解析该类可获得if-else条件语句块数量特征Questioncolon，，由于if-else条件语句存在多条件或多if-else条件嵌套组成，所以采用递归方式分别解析if条件、if条件内容、else分支内容、then条件内容，由于if条件内容为条件表达式，所以可以采用3）模块进行解析，并存储特征QUESTIONCOLONIF；if条件内容为赋值表达式，所以可以使用6）中对过程赋值语句的提取方法来提取，并存储至对应的过程赋值特征中，同理可对Else内容块及Then内容块解析，并提取特征QUESTIONCOLONELSE、QUESTIONCOLONTHEN中；从而完成对if-else条件分支语句的特征提取；

### Case条件分支语句 （case）

基于7）中对Verilog HLD分支控制语句的分析可知，case分支控制语句与if-else语句都是用来实现分支控制选择，但是case分支控制语句是用来实现多路分支选择控制的分支语句，其在实现多路选择控制上更加简便和直观，其通常用于描述译码或状态机，其语法格式如图：

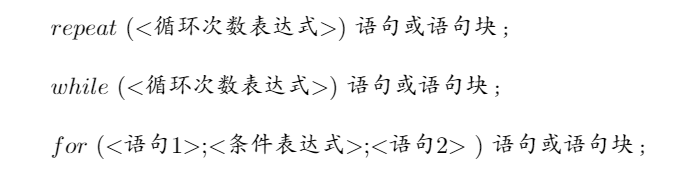


图

在Verific-Parse-tree中Case语句块被解析构造存储至VeriCase类中，通过统计该类数量可获得Case数量存储至CaseNum特征，VeriCase类中包括Case控制表达式及分支表到式及分支语句块，模块需要对该类及表达式进行解析，解析控制表达式可用3）中所述方法进行递归提取，从而获得控制表达式数量CASECONDITIONNUM、控制表达式位宽CASECONDITIONWIDTH、以及分支数量CASTITEM，由于分支语句块内容多位赋值语句所以可以使用5）中模块进行对应特征提取；

### 循环语句 （for、while）

与条件分支语句一样，循环控制语句也是一种高级Verilog HDL程序语句，Verilog提供的循环控制语句有forever、repeat、while、for四种循环语句，其中除去forever语句是一种无限的循环语句外，其余的三种循环语句都是根据指定循环条件或循环次数进行循环；循环控制语句语法格式如下如图：



图

对于循环语句其逻辑综合后为将循环体按循环次数展开，所以也需要对其提取特定特征，主要分为以下两部分即循环次数，循环体内容；

对于For循环体，由于其循环次数在for语句条件表达式部分，需要解析出for循环初始值，判断条件，补偿，从而计算出循环次数，for循环语句块在Verific-Parse-tree中被构造为VeriFor结构，其中循环次数部分需要二次解析，通过解析后通过计算初始值-终止值/步长来获得循环次数，并存储于特征foritem，同时将for结构体数量存储于特征forblock，对于循环体而言，其内容可嵌套if-else、赋值表达式等多种语法，需要使用递归算法对循环体进行解析，使用3）、4）、6）所提及的算法进行统计，同时需要对统计的数量再乘以循环体个数，并存储至特征内；

对于while循环体，由于其判决条件存在与其条件表达式中，所以需要我们通过二次解析条件表达式来获取终止值，同时需要通过判断终止值的变量表达式来计算循环体次数；

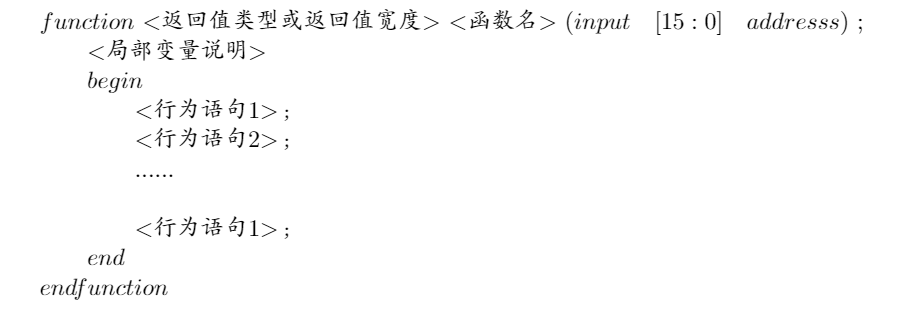
对于repeat循环体，由于repeat循环体的循环次数在其循环次数表达式中，只需对循环次数表达式进行二次解析获得循环次数即可，同理使用上述方法解析循环体特征信息，并存储至对应特征中；

对于forever循环语句，只需要将Forever语句中的forever循环体内容进行解析即可完成对forever特征的提取；

通过以上方法完成对循环语句的特征提取；

### 函数语句（function）

Verilog HDL语言中也包含类似于其他编程语言中的函数概念一样的语法，函数（Function）的语法形式如下，如图：



图

函数定义结构只能出现在模块中，而不能出现在过程块内且函数必须至少有输入输出端口和双向端口。

在Verilog HDL语句中，函数代表了纯组合逻辑。函数在综合的时候，被展开成了内联（in-line）的代码。任何在函数中定义的局部变量都被当做纯粹的临时变量，仅被综合成导线。由于Function内部代表了纯组合逻辑，所以对于Function内部的逻辑表达及Function的端口及调用数量进行特征分析及提取；

在Verific的Parse-tree中Function被构造存储于VeriFunction类中，通过重载Visitor函数获取该RTL设计下的VeriFunction类，并统计其数量并存储至FunctionNum特征中，同时进一步获取Function的内部逻辑结构，同时对内部逻辑使用3）、4）、5）模块中提到的方法对组合逻辑进行提取，并存储至对应的类中；从而完成对Funtcon的特征提取；

## 特征器构建

通过5.2小节所述的特征提取方法构造对应的模块，从而完成特征提取，获取的特征数量为：xx，特征列表如下：

涵盖VerilogHDL的所有关键语法及在逻辑综合和技术映射中重要的语法信息。

## 特征工程流水线构造

本研究提取了XX项特征，这些特征的数据取值范围差别很大，量纲也不同，如端口数量为个，而位宽可大可小等，没有办法进行比较。所以我们需要对特征进行进一步的预处理，使其能够更加适合在机器学习环境下进行预测。

我们需要构造特征工程流水线来帮助我们减少复用操作，从而在样本数据集发生更新时能够快速处理。

### 特征预处理

1. 数据清理

大部分的机器学习算法无法在确实的特征上工作，所以我们要创建一些函数来辅助其完成，首先就是对于数据存在缺失的情况，为解决它我们选用三种解决办法：

1. 放弃这些数据缺失的区域；
2. 放弃整个属性；
3. 将缺失值设置为某个值，比如0或者平均数、中位数等

通过对预测目标和特征属性的分析，由于该特征目标为预测准确设计所需FPGA资源数值，所以对于数据存在缺失的数据类型，我们选择放弃这些数据缺失的数据，从而不对预测结果产生影响。

1. 处理文本和分类属性

由于特征中可能存在除数值属性外的其余数据表达形式的特征，由于机器学习模型是数学模型，所以我们需要将文本数据属性和具有分类含义的数据属性进行抽象处理：

1. 使用OrdinalEncoder顺序编码对文本属性进行编码；
2. 使用OneHotEncoder独热编码对分类属性进行编码；
3. 特征缩放

最重要也是最需要应用到数据上的转换就是特征缩放。如果输入的数值属性具有非常大的比例差异，往往会导致机器学习算法的性能表现不佳，当然也有极少数特例，比如提取特征中的端口数量和端口位宽就很有可能存在数量级上的差异。

在机器学习领域最常用的同比例缩放所有属性特征的两种常用方法：

1、最小-最大缩放也可称之为归一化，归一化的操作较为简单，即将值重新缩放使其最终范围归于0~1之间，实现方法是将值减去最小值并除以最大值和最小值的差。

2、标准化方法，与归一化方法相比，标准化则完全不一样，其实现是首先减去平均值，所以标准化值得均值总是零，然后除以方差，从而使得结果的分布具备单位方差。与最小-最大缩放的区别是，标准化不将值绑定到特定范围，但是标准化的方法受异常值的影响更小。

### 转换流水线

由于上述特征预处理需要使用正确的顺序来执行，所以我们采用scikit-learn提供的pipeline类来支持我们实现上述对数据的转换，构造流水线转换器如图：

****

图

当数据集经过流水线处理后便形成了进入机器学习模型进行训练的基础.

## 本章小结

本章叙述了本文中最为重要的一部分，即对Verilog HDL语言进行特征提取，并完成特征清洗、特征处理和基本特征工程；为下一步构架机器学习模型打好基础；