摘要随着后摩尔时代的到来，在超大规模集成电路设计阶段验证已经逐渐成为困扰各大芯片设计人员的关键问题，虽然可以使用软件仿真、硬件加速仿真等验证方法来加速验证流程，但是随着集成电路设计规模逐渐增大，原先的验证方法在时间成本上已经无法满足当前快速设计迭代的需求，使用FPGA进行芯片设计原型验证已逐渐成为验证阶段主流。但随着设计的规模剧增，单片FPGA已无法满足超大型集成电路设计的验证需求，从而衍生出高密度FPGA原型验证系统与配套EDA软件。而如何将超大型集成电路设计在满足资源约束的前提下合理的划分至高密度FPGA原型验证系统中的每个FPGA中，将成为集成电路验证划分阶段的关键问题，而超大型集成电路设计所需要的FPGA片上资源数量将作为合理划分电路的重要依据。同时随着机器学习的快速发展，将EDA软件与机器学习进行结合从而提升时间性能也逐步成为EDA行业的发展趋势。论文主要对高密度FPGA原型验证系统中的划分阶段如何对集成电路进行快速资源估算进行了研究，基于C++语言并结合相关FPGA验证原理，设计了一款RTL级快速资源估算模型，并实际应用于商业软件中，解决了FPGA原型验证RTL划分阶段快速资源估计这一工程问题，为我国自主研发的高密度FPGA原型验证系统及配套EDA工具做出了一定贡献。同时为进一步研究快速的RTL级资源估算方法，将RTL级设计资源预测与机器学习相结合，设计并实现了面向RTL级资源估算机器学习模型，为EDA软件与机器学习的结合论证了可行性。本论文的主要工作与成果如下：（1）对原型验证流程中综合及技术映射阶段进行基本研究，并举例简述在逻辑综合的基本原则及技术映射阶段的基本算法，以此为依据使用Parser作为底层软件基础，并使用C++设计了基于传统集成电路流程的快速资源估算模型工具，之后使用来自多个行业的超大规模集成电路进行验证，在未进行优化时，时间性能提升超过100%，对LUT与FF资源平均估算准确度达60%以上。在上述传统快速资源估算模型的基础上设计了为满足于高密度FPGA原型验证系统划分阶段需求的BottomToUp算法，使得对RTL级设计资源估算时间较优化前时间利用率提升达220%以上，同时为进一步提升估算精度，设计了LUT合并、面积约束等优化与约束项算法，使得不考虑特殊器件及工艺的情况下LUT及FF等资源的估算准确率最高可达90.11%，更加完善了快速资源估算模型的性能及易用性。（2）研究人工智能与传统EDA工具相结合的可能性，并使用Verific为基础，通过进一步分析逻辑综合与技术映射的原理，分析集成电路RTL级描述语言特征，构建RTL级描述特征提取器，实现提取RTL级描述特征67种。对特征相关性、贡献度进行进一步分析后进行特征工程，并结合线性回归、随机森林、SVM支持向量机、人工神经网络构造适用于RTL级设计FPGA片上资源估算的机器学习模型，并完成超参数调校及训练，实现平均LUT、FF、DSP等资源估算模型确定系数为73.4%的神经网络模型。最后通过对其数据进行分析，进一步验证了机器学习在原型验证划分阶段集成电路快速资源估算领域的可行性。关键词：FPGA原型验证，资源估算，特征工程，机器学习ABSTRACT插图索引图1.1高密度FPGA原型验证系统2图1.2高密度FPGA原型验证流程3图1.32008年Xilinx公司快速资源估算模型[17]5图2.1FPGA基本组成图示9图2.2LUT结构图示10图2.3LUT基本工作原理10图2.4D触发器11图2.5FPGA设计基本流程11图2.6FPGA逻辑综合流程13图2.7连续赋值语句综合图示14图2.8阻塞赋值语句综合图例15图2.9非阻塞赋值语句综合图例15图2.10非阻塞赋值语句综合图例16图2.11算数运算符综合图例16图2.12if语句综合图例16图2.13case语句综合图例17图2.14循环语句综合图例17图2.15函数语句综合图例18图2.16工艺映射流程图19图2.17电设计转化为DAG图20图2.18逻辑划分20图2.19器件覆盖21图2.20Mapping算法伪代码22图2.21机器学习实现RTL级设计资源估算流程23图2.22Bagging算法伪代码24图2.23决策树学习基本算法25图2.24随机森林模型算法伪代码27图2.25神经元计算图示28图2.26感知器结构图示29图2.27多层感知机结构图示30图3.1VeriTreeNode类视图34图3.2VeriModule类视图35图3.3Verific-Netlist-DataBase数据结构图36图3.4快速资源估算模型文件流37图3.5快速资源估算模型流程图示38图3.6快速资源估算模型Parse-Design流程39图3.7RTLElaborate操作图示39图3.8Flatten操作电路图示40图3.9Flatten操作树结构图示40图3.10RTL级基本器件库41图3.11快速资源估算模型计算结果41图3.12RTL代码中的GlueLogic和Instance42图3.13BottomToUp算法数据结构43图3.14RTL设计树结构示意图43图3.15BottomToUp算法伪代码44图3.16优化后快速资源估算模型输出45图3.17LUT-5BaseFPGA结构图示46图3.184输入查找表真值表47图4.1特征提取器结构图52图4.2Verilog操作符图示53图4.3运算符特征提取伪代码54图4.4连续赋值语句特征提取伪代码55图4.5VerilogAlways过程块语法格式55图4.6Always特征提取伪代码|56图4.7Verilog过程赋值语句语法格式56图4.8VerilogIf条件语句语法格式57图4.9If条件语句特征提取伪代码58图4.10VerilogCase语句语法格式58图4.11Case条件分支语句特征提取伪代码59图4.12Verilog循环语句语法格式59图4.13循环语句特征提取伪代码60图4.14VerilogHDLFunction语法结构61图4.15RTL语法特征提取器流程图62图4.16特征数据集直方图64图4.17特征热力图65图4.18特征预处理流水线69图4.19模型训练开发流程70图4.20各模型尝试预测LUT资源结果71图4.21机器学习模型训练及调优流程71图4.22随机森林预测LUT模型参数搜索学习曲线72图4.23随机森林LUT预测模型学习曲线72图4.24随机森林预测FF特征搜索学习曲线73图4.25随机森林FF预测模型学习曲线73图4.26神经网络学习曲线，左图为超参数调优前学习曲线74图4.27神经网络模型结构图75图4.28神经网络与LUT模型学习曲线75图4.29神经网络模型结构图76图4.30神经网络模型预测FF资源学习曲线76图4.31机器学习模型验证结果77表格索引表2.1典型多层感知机架构31表3.1快速资源估算模型无优化结果对比48表3.2快速资源估算模型增加优化结果对比48表4.1Verilog主要语法类型与关键字51表4.2部分基本数据集数据图示63表4.3数据集基本概要63表4.4RTL级特征与LUT资源的相关性数据66表4.5RTL级特征与FF资源的相关性数据66表4.6RTL级特征与BUFG资源的相关性数据67表4.7RTL级特征与IO资源的相关性数据67表4.8模型评估77符号对照表符号符号名称F激活函数L损失函数I(X,Y)特征权值分数H(Y)熵D特征个数\nablaf梯度G二维高斯函数d\prime保留的特征个数M正交基向量Q协方差矩阵Z加权和缩略语对照表缩略语英文全称中文对照ASICApplicationSpecificIntegratedCircuit专用集成电路SoCSystemonChip片上系统FPGAFieldProgrammableGateArray现场可编程逻辑门阵列RTLRegisterTransferLevel寄存器传输级别LUTLookUpTable显示查找表DSPDigitalSignalProcessing数字信号处理单元BRAMBlockRAM块状存储HLSHighLevelSythesis高层次综合LBLogicBlock逻辑块FFFlip-Flop触发器CLBConfigurableLogicBlock可配置逻辑块RAMRandomAccessMemory随机存取存储器SRAMStaticRandomAccessMemory静态随机存取存储器HDLHardwareDescriptionLanguage硬件描述语言APIApplicationProgrammingInterface应用程序接口CADComputerAidedDesign计算机辅助设计EDAElectronicDesignAutomation电子设计自动化RFRandomForest随机森林LRLinearRegression线性回归DTDecisionTree决策树SVMSupportVectorMachine支持向量机MLPMultilayerPerceptron多层感知机ANNArtificialNeuralNetwork人工神经网络目录摘要IABSTRACTIII插图索引V表格索引IX符号对照表XI缩略语对照表XII第一章绪论151.1研究背景151.1.1论文研究背景151.1.2高密度FPGA原型验证系统及EDA161.2国内外研究现状181.3研究的意义191.4主要内容及组织结构20第二章FPGA资源估计方法232.1FPGA基本结构与资源组成232.2FPGA逻辑综合与技术映射262.2.1行为级硬件描述语言262.2.2FPGA逻辑综合272.2.1FPGA技术映射322.3机器学习模型与资源估计方法362.3.1机器学习与EDA结合的可行性362.3.2特征提取方法372.3.3随机森林372.3.4人工神经网络412.4机器学习性能评估指标45第三章快速RTL级资源估算模型的设计与实现473.1Verific及相关数据结构473.1.1Verific平台介绍473.1.2Parse-Tree数据结构473.1.3Netlist数据结构493.2资源估算模型流程与构造513.2.1资源估算模型流程设计513.2.2资源估算模型构造实现523.3快速资源估算模型优化563.3.1BottomToUP算法的设计与实现563.3.2冗余逻辑优化及合并603.3.3最小面积优化方法613.4结果分析633.5本章小结64第四章基于机器学习的RTL级资源估算654.1VerilogHDL语法特征分析654.2RTL级设计特征提取方法设计664.3自动化特征提取器的实现754.4数据集分析764.4.1数据集构建764.4.2数据集分析774.5特征工程824.5.1特征预处理824.5.2转换流水线834.6模型框架选择844.7机器学习模型训练854.7.1随机森林864.7.2人工神经网络884.8结果与分析914.9本章小结92第五章总结与展望935.1工作总结935.2工作展望94参考文献95致谢99作者简介101绪论研究背景论文研究背景随着市场环境的逐步升级，市场需求对与IC的功能愈加严格，IC的复杂度在工艺的进步下也逐步增加。目前的ASIC（ApplicationSpecificIntegratedCircuit，专用集成电路）设计，这项又称为片上系统SoC（SystemonChip，片上系统）的集成电路技术在业界已经得到广泛的认可与应用，且当代的SoC设计也不在局限于一块IC上一般情况下，由于采用ASIC设计具有功耗低、投产成本低、迭代周期快等优势，所以ASIC可容纳更为复杂和灵活的逻辑设计。同样正是由于现代ASIC设计的高复杂度，使得其研发设计的风险也在大幅增加。一旦芯片设计失败，将会导致大量研发本的损失，所以，IC正确性验证逐步变为超高复杂度IC研发过程中必备且关键的环节。最新一项统计数据表明在一些项目中芯片的设计团队规模甚至不如验证团队规模，验证所花费的时间也占到产品研发周期的一半以上，找到有效可行的验证方法成为验证的关键[4]。目前，最常见的验证方法是软件仿真技术，但是软件仿真虽然简单易行，在面对复杂的集成电路设计时，其验证效率过低，最终可能影响到产品的及时发布[6]。而基于FPGA的原型验证方法，依靠其可反复擦除、可真实的模拟设计中的硬件行为以及能够系统全面的反映设计功能等优势，被越来越多的验证团队所采用[6]。目前随着高性能ASIC芯片设计复杂度的提升以及体系结构的不断升级换代及创新，SoC设计的规模在快速膨胀，研发中的验证阶段已成为高性能芯片研发过程中的关键环节，因此为保证验证的高效性和准确性,许多EDA或集成电路设计公司相继推出了许多相对高级的验证方法。在具有超高复杂与高性能IC验证上，主流验证方法分为以下三类：软件模拟验证、硬件加速器仿真验证、FPGA原型验证。软件模拟验证是验证集成电路设计功能正确性上最常采用的方法，其具有模拟精确、方便灵活等优点，因此对于量级较小的IC模块的正确性验证通常会采用EMU的方法。但是，当芯片整体复杂度提升，软件模拟的效率也在不断下降，同时也将难以覆盖整体IC功能的验证。实践表明，不论使用哪种算法去设计EDA工具的仿真方式，当IC设计所包含的电路门数超过一定数目时，EDA工具对于系统信号的反应将很难保证，验证的效率也将会被极大制约，从而很难高效的实现验证结果[3]。针对软件模拟仿真的局限性，各大EDA厂商与集成电路厂商进而提出通过硬件加速器来实现仿真验证，其主要思想是通过专门的硬件加速器，将软件仿真中的一些复杂运算通过使用告诉硬件加速器来计算，最后硬件加速器将运算结果传回仿真软件，最终提升整体上的验证效率。而且硬件加速器验证的仿真环境也更加真实，相比于完全使用软件来实现仿真，硬件加速器使用了真实的硬件仿真环境，可以更加真实的反映集成电路设计系统的各种问题，从而可以辅助开发人员有效的发现软件仿真无法捕获的异常，这对于验证工程师的调试很有帮助，典型的硬件仿真器有Cadence公司的Palladium和Mentor公司的Veloce系列[35]。但是硬件仿真器也存在固有的缺陷，因为其需要针对特定的电路构造特点的硬件加速设备，同时还需要EDA软件与硬件加速设备间的通信接口，而一旦所验证的设计改变，原有的硬件加速固件将可能无法复用，从而需要重新设计硬件固件及相关接口。从理论上讲，本文的所提及的高密度原型验证系统亦可基于硬件仿真器实现，但硬件仿真器的成本高昂，而且由于当代先进FPGA的技术发展及配套软件已经较为完备，所以通过基于FPGA可反复擦除的特性，使用先进FPGA来进行高密度原型验证成为了当下集成电路设计验证阶段较优的选择。随着FPGA性能的不断提升，基于FPGA的原型验证逐渐成为大规模复杂芯片验证的选择。高密度FPGA原型验证系统及EDA高密度FPGA原型验证系统近年来，随着人工智能、GPU、CPU等高性能模块的快速发展与运算需求的增加，仅靠单个FPGA的片上资源已难以ASIC芯片设计的需要，因此基于多颗FPGA的高密度原型验证技术将成为最优的选择。对于能够使用多FPGA进行正确的验证，则必然需要进行逻辑划分，而为实现正确且合理的逻辑划分，首先要解决的就是设计逻辑综合时间长的问题[4]。一个设计规模达到几亿门甚至数十亿门的大规模SoC设计，若使用整个设计先进行综合，再进行逻辑分割的方式，设计的综合时间会非常的漫长，一次综合流程就有可能花费的时间长达数天甚至数周，若设计的规模特别大，甚至会出现编译软件或服务器崩溃的情况[5]。显然，面对超大规模的SoC设计采用RTL（RegisterTransferLevel，寄存器传输级别）级分割，将会成为最优的选择，RTL-Partition流程如图1.2所示。高密度FPGA原型验证流程使用RTL划分通过在RTL阶段将设计划分为个子设计后，便可以有效解决设计逻辑综合时间长的问题，同时实现并行编译、综合、布局布线的流程，从而有效提升FPGA原型验证的效率。但为正确实现RTL级的逻辑划分，则需要解决以下几个关键问题：（1）首先，在对超大型SoC设计不进行逻辑综合前提下进行RTL级别分割时，需确定最合理的逻辑划分的边界，满足SoC设计的全局时钟树，同时使得进行逻辑划分后的SoC设计满足整个大型FPGA原型验证系统的互联资源和每个划分后的子设计不超出每个FPGA可综合的资源的数量（为增加FPGA综合及布局布线的成功率，常使用FPGA资源总量的60%-70%）；（2）其次，由于无法对设计进行整体综合，则无法获得设计在FPGA上所需的准确片上资源数量，则如何对设计在RTL级进行快速资源估算，在不进行逻辑综合为前提下获得设计中每个模块的资源，从而为确认划分的边界提供参考依据将成为关键问题；（3）最后，如何通过已经获得的RTL级设计的资源与大型原型FPGA设计平台相结合，能同时满足FPGA间的互联结构、互联资源、片上逻辑资源等条件实现合理的逻辑划分，并且划分结果与原始设计逻辑等价；本文将主要讨论上述问题中的（1）（2），即如何对RTL设计进行快速且满足RTL级逻辑划分所需精度的快速逻辑资源估算方法并尝试结合机器学习探究解决该问题的方法。国内外研究现状当设计者通过硬件描述语言完成对集成电路寄存器传输级（RegisterTransferLevel，RTL）的软件编码设计后，RTL级设计将通过几个基本步骤，最终转化为用于配置FPGA内部结构的比特流文件。RTL需要经过的基本设计步骤，包括逻辑综合、逻辑优化、技术映射、逻辑打包、布局、布线等。FPGA的硬件组成结构中包含许多不同类别的片上资源，主要有实现逻辑电路所需要的逻辑资源，用于提供外部输入/输出接口的资源，以及连接并导通前两种资源的布线资源。具体来讲，一般包括触发器（F-F）、数字处理单元（DSP）、查找表（LUT）、块状存储（BlockRAM，BRAM）等多种片上起源。其中，LUT和FF等逻辑资源一般存在于逻辑块LB（LogicBlock或slice）中。由于高密度FPGA原型验证整个流程如图1.2包含多个步骤，即RTL级设计组网布局、RTL级设计分割、逻辑综合、工艺映射、单个FPGA的布局和布线等，由于高密度FPGA原型验证流程步骤繁多且耗时较长，设计者在编写和修改RTL描述时，无法快速预估待划分电路在FPGA上实现时需要的片上硬件资源数量，这样就可能导致无法完成综合流程或者无法完成FPGA布局布线，或互联关系复杂，无法满足FPGA间组网关系，从而需要多次设计迭代来解决上述问题。另外，在逻辑综合流程中，每一部分RTL级描述所需要的实际FPGA片上资源的数量，也将对RTL到门级网表的转化与优化过程有指导意义。综上所述，在RTL级分割阶段如何能够快速且准确的得到RTL级描述所需要的FPGA片上资源使用量，是一个很关键且重要的问题。虽然在国内也有部分针对大型设计提供EDA及FPGA原型验证厂家，但是对于在划分阶段涉及的资源估算问题在我国国内暂时没有可商业使用的较成熟的解决方案。随着FPGA设计规模的越来越大，对于片上资源估算这一环节的重要性越发明显，也是必不可少的步骤，使得FPGA资源估算这一问题也逐渐成为学者和EDA厂商的研究重点，随之而来的也是多种多样的估算方法也都各有优劣。2008年Xilinx公司PaulSchumacher和PradipJha提出由于用户在对FPGA设计的开发使用过程中，无法规避对设计进行逻辑综合这一步骤，从而可以通过使用快速逻辑综合生成的网表结构，通过在设计的宏观层面对现有的器件匹配库对原设计进行匹配替换，从而得到初步的资源估算结果，使用该方法可以较快速的得到不计精度的资源粗略估计值[17]。2008年Xilinx公司快速资源估算模型[17]2011年代尔夫特工业大学RoelMeeuws等人提出可以通过深度学习建立基于统计方法的高级定量硬件预测模型设想，并做出了初步的验证，为人工智能在资源估计这一问题上的实现建立可行性分析[11]。2017年法国瓦伦西亚大学MariemMakniLAMIH发布的论文中提出一种面向异构FPGA的SoC片上资源估算方法，其出发点是通过结合HSL（HighLevelSythesis，高层次综合）报告构建模型对布局后FPGA真实面积进行估算[13]。2018年SteveDai，YuanZhou等人在IEEE会议上发表的论文中提出了一种通过提取HLS报告相关信息特征对FPGA布局布线后资源及时序进行预测的方法，并且达到了一个较好的预测结果，由于HLS是比RTL级在设计流程上更高层级、更抽象的表述方法，所以该方法的提出进一步为通过RTL级特征预测实际所需FPGA片上资源提供了可行性[12]。通过对国内外研究现状的分析，我们可以发现在面向高密度FPGA原型验证系统的RTL级资源估算方面我国还存在较大的待填补的空白。研究的意义对于高密度FPGA原型验证系统而言，面向的设计都是规模上千万门或设计规模超过亿门，更有甚者规模超过百亿门，需要上百片当代先进FPGA才可容纳下整个设计，但是对于给定的RTL描述，它所需的FPGA片上硬件资源数量一般需要在完成逻辑打包和布线后，才可以全部得到。但对于规模较大的电路设计，从RTL描述到布线的设计综合流程往往需要花费几个小时乃至数十小时。当出现片上资源不足的问题时，设计者需要修改RTL描述并进行耗时较长的设计综合流程迭代。若用户的设计超过百亿门时，进行一次迭代所耗费的时间成本，人力成本都是不可预估的，所以若可以在不进行综合、布局布线等流程时快速获得所需要的所需要资源的估计值，将会为缩短开发人员的迭代周期提供帮助。同时由于高密度FPGA原型验证划分阶段需要以设计资源使用情况来划分合理的分割边界，合理的分割边界将会对设计在满足硬件FPGA组网前提下为系统的运行，规划，验证，调试提供良好的帮助，而分割边界的确认需要以RTL设计中的每个层级中每个节点的Instance的各项资源作为依据。综上所述不论对于开发人员还是对于高密度FPGA原型系统而言，能够快速且精确的获取到大型设计所需的FPGA片上资源及对应层级所需的资源的估计值将具有巨大的意义。主要内容及组织结构针对我国需要开发具有自主知识产权的大型FPGA原型验证流程和相应的支持EDA软件，本文首先介绍了FPGA的基本结构及大型FPGA原型验证流程中的逻辑综合、技术映射与机器学习模型的一般方法作了详细描述，最后分别以上述基本流程为基础实现了以传统流程为基础的快速资源估算模型及以机器学习为基础的RTL级资源估算模型。本文的组织结构如下：绪论主要介绍了当前集成电路设计阶段中验证阶段的重要性及相关的验证手段，并介绍了高密度FPGA原型验证系统与快速资源估算在RTL划分阶段的重要性、意义与待解决的关键问题，并对论文中的主要工作内容进行简要叙述。FPGA资源估算方法介绍了FPGA的基本结构与资源组成，并进一步介绍了FPGA设计与验证流程中的快速逻辑综合与技术映射的相关原理，为第三章资源估算模型的实现打下基础，同时分析了EDA工具与机器学习的可能性，并着重介绍了本文第四章实现所用的特征提取方法与机器学习模型。快速RTL级资源估算模型的设计与实习本章主要介绍了基于C++与第三方Parser，结合第二章所述的相关原理与方法，如何设计及实现RTL级设计进行快速资源估算的模型，并再次基础上为进一步提升运算速度与估算准确度并满足高密度FPGA原型验证系统RTLPartition的需求，设计并实现了BottomToUp算法，并论述了模型中增加的优化方法与约束项。基于机器学习的RTL级资源估算本章主要结合第二章所述的机器学习模型方法，通过分析进一步RTL语法与综合、技术映射的关系，基于C++实现了面向RTL设计的自动化特征提取器，并通过尝试多种机器学习模型，基于Python构建了面向RTL级设计的机器学习资源估算模型，并使用多个Case进行验证，并对模型的预测结果进行分析。总结与展望总体概括了论文的工作，并总结了论文中可改进的方式方法与方向。FPGA资源估计方法FPGA是一种现代研发场景下常见的可编程逻辑器件，是一种可以用来实现任意逻辑电路的集成电路[43]。FPGA的出现是当代集成电路快速发展的一个缩影。上世纪，由于研发、设计集成电路的成本急剧增加，集成电路研发企业都想要找到一种即满足功能需求又价格低廉的芯片。在这种时代背景下，现场可编程门阵列FPGA应运而生。相对于其他相同种类的芯片，由于其可反复擦除的特性，使得基于FPGA进行研发的容错率更好，成本降低。在目前的多数集成电路研发团队的研发过程中，研发团队会先基于一定数量的FPGA芯片对集成电路功能、算法进行实现，并根据实际情况进行修改直到设计达到预期的效果，最后再将达到要求的设计用ASIC实现并最终投入生产。FPGA的本质结构是在现场可编程的门阵列芯片。但FPGA实际上并不是仅依靠简单的门级结构所组成。本章将主要介绍FPGA的基本构成资源及高密度FPGA原型验证的基本流程。FPGA基本结构与资源组成FPGA大致由三部分构成：第一部分是实现逻辑电路的逻辑要素即逻辑块（LogicBlock，LB）也就是我们常说的基本逻辑单元，可编程逻辑中逻辑块的实现方式有多重，比如最早的乘积项和现在的查找表、数据选择器等，但是无论哪种方式都是由可以实现任意逻辑电路的可编程部分触发器即Flip-Flop，FF等数据存储电路和数据选择器组成[43]。逻辑要素是在FPGA设计开发流程中最为紧俏的资源种类之一，各大FPGA厂商对各自的逻辑块结构有不同的定义，但基本原理都是类似的，如Xilinx公司设计研发的FPGA其最小基本逻辑单元为CLB(ConfigurableLogicBlock)，是以LUT来构成的。第二部分则是用以组成信号输入和输出的重要部分即IO块，主要用于与FPGA外部信号进行交互，第三部分是连接FPGA内部互联资源的内部布线资源。FPGA基本组成图示上述FPGA三部分主要组成具体包括以下内容：可编程输入输出及内嵌RAM、可编程逻辑单元、布线资源等。可编程逻辑单元负责逻辑功能的实现，布线资源则能够将各个单元进行链接，而内嵌RAM可配置为各种存储形式，输入输出负责FPGA与外部模块的通信，在最新及更高的FPGA上还会集成DSP、LUTRAM、CARRY等高级资源，如Xilinx最新的Virtex-7系列FPGA。而在以上FPGA的结构中，设计者及其支持软件中最关系、最重要的是可编程逻辑单元，它主要包括触发器和查找表两部分。查找表在功能实际上等效为一种特殊的静态存储器SRAM。可编程逻辑单元中，LUT表示的是一张内存表，其存储的字数或容量实际取决于存储地址的位数。FPGA上LUT的基本存储单元大多使用静态随机存储器实现。LUT单元的具体结构如图2.2所示LUT结构图示如图2.2中所示的查找表，它拥有三个输入，则说明其可实现任何逻辑运算函数为三输入。一般输入的LUT由个静态随机存储单元和一个输入的数据复选器组成。LUT基本工作原理LUT基本工作原理如图2.3所示，上图左侧为电路的实现逻辑及真值表，上图右侧则对应的等效查找表。由图2.3可以知，查找表中的实际内容是将逻辑的输入等效转化为对应的存储地址，对应的输出的则是存储地址中的具体内容。最终的结果是查找表将对应的复杂逻辑运算对等的映射为存储器中的数值，这也是使用FPGA进行验证效率远远高于使用软件模拟的方法的主要因素。所以LUT资源是在对大型集成电路设计使用FPGA进行原型验证时的紧俏资源。此外，当面对的是时序逻辑电路，则FPGA会通过在每个查找表的后侧通过触发器的配置来完成。D触发器可编程逻辑单元中的另一种重要资源FF（FlipFlop）触发器,其基本结构如图2.4所示，其基本原理同数字电路设计使用的触发器原理相同，也称为双稳态门，或称双稳态触发器。触发器（flip-flops）电路之间是相互关联的，从而形成数字集成电路的逻辑门。在FPGA中存在有多种类型的FF，如切换触发器、重置触发器、JK触发器和D信号触发器等。常见的触发器结构通常包含最多2个输入信号、时钟信号及输出信号。触发器资源通常在大型RTL设计进行高密度FPGA原型验证时也是一种常常面临资源紧俏的情况。FPGA逻辑综合与技术映射在本文中提到的FPGA原型验证方法流程中的起点如图2.5均为RTL级硬件描述语言代码，故本章我们先对硬件描述语言verilogHDL语法及逻辑进行简略的介绍，同时进一步介绍基于VerilogHDL的逻辑综合及工艺映射基本原理及基本算法，上述内容将为后续我们进一步建立可工程化的快速资源估算模型提供基础。行为级硬件描述语言FPGA设计基本流程VerilogHDL是用以描述硬件行为级的一种高级硬件描述语言，其通常以文本或字符形式来描述数字逻辑系统的逻辑行为与系统结构。通过合理的使用该语言可以表示正确表示逻辑电路图、逻辑表达式等，还可以表示数字逻辑系统所完成的逻辑功能。VerilogHDL是当前世界上最流行的两种硬件描述语言之一，都是在20世纪80年代中期开发出来的。根据IEEE标准所说明的，VerilogHDL是一种用来描述和创建电路系统所有阶段的表示形式，由于VerilogHDL是机器可读的、人类可读的，所以该语言能够支持对硬件设计进行开发、正确性验证、合成和测试、硬件设计数据的通信和维护、修改[16]。在VerilogHDL中有两种基本语句：一种是连续型赋值语句，通过使用连续性赋值表达式，其中使用的寄存器和线网都用于连续驱动FPGA中的线网结构，从而实现基本建模；另一种是过程型赋值语句，在过程性赋值语句的行为级描述中，将使用寄存器来存储寄存器与线网间的运算结果，从而实现行为级的描述建模。一个完整的RTL级行为描述设计通常会包含多个模块，每个模块都将用于描述该模块对应的功能描述逻辑[16]。这些模块将被开发者进行抽象，然后通过线网连接。而在典型FPGA设计流程中，或大型高密度FPGA原型验证流程中，首选通过使用VerilogHDL等硬件描述语言编写寄存器传输级（RTL）代码来描述电路设计的行为级逻辑即RTL级描述。RTL级描述通过逻辑综合得到电路网表（Netlist），通过对网表进行优化、技术映射、FPGA布局布线等操作后，最后生成FPGA配置数据（bitstream）。其中对于大型原型验证系统最为关注的前端工程就是逻辑综合和技术映射，这个过程是验证VerilogHDL语法正确性和计算该设计所需FPGA片上资源总数的重要过程。其中逻辑综合和技术映射的过程就是将VerilogHDL语言描述的寄存器级代码转换映射至具有工艺概念的逻辑器件的过程，也是本文中结合机器学习人工智能进行FPGA资源估算设计过程中重点关注的过程。FPGA逻辑综合对于FPGA数字电路设计而言综合简而言之就是从采用硬件描述语言构造的寄存器传输级(RTL)电路抽象级模型构造出对应的门级电路网表的过程。综合的过程就是将具有较高层级的对集成电路行为或逻辑的描述经由自动化工具转化为较低层次描述的过程，综合细分行为综合与逻辑综合，行为综合是从算法表示、行为描述转换到RTL级，如HLS综合；而逻辑综合是从RTL级描述转换到包含有多级触发的逻辑门级的过程，也是在FPGA片上资源估算过程中最关注的步骤，后续将主要介绍逻辑综合这一对高密度FPGA原型验证系统较为重要的流程。综合整个ASIC设计流程的中间步骤，综合后生成的网表文件，其本质是基于导线来进行连接的RTL级功能块（如FF、LB和MUX等）所组成。而这里一般就需要使用RTL模块构造器来通过用户所注明或进一步指定的目标工艺元件库中构造或从已有库中获取必须的RTL级功能模块。综合产生网表文件后，逻辑优化器会读入网表，并且会按照用户配置的FPGA片上资源要求和时序约束为目标来对网表进行优化。并且，这些资源约束或称为面积约束和时序约束也能够指定模块构造器合理的选取或生成RTL级功能块。既然综合器的目标是能将硬件描述语言转换成为底层的门级网表，那么综合器必定有一种映射的机制，能够实现Verilog代码中的某些变量、组件映射成对应的硬件元件，这种映射机制将是后续我们基于机器学习进行资源映射方法研究所重点关注的对象。FPGA逻辑综合流程①综合首先需要使用RTL代码构成的电路行为级描述进行编译使其转换为GATE级描述；②RTL模块构造器从用户指定或综合器所携带的工艺库中里提取并生成对应的模块；③逻辑优化器读取未优化的门级网表，并按照用户所期望的布局后面积和系统运行各个信号的时序约束，进行进一步的优化；由于本文中主要涉及计算RTL模块构造器从工艺库中提取对应的语法及模块进行TechnologyMapping映射后所得到的资源，所以当我们聚焦于本文所述的基于机器学习资源估算模型，更加关注于VerilogHDL语句是如何转换成实际硬件器件的，如：数据类型是如何映射成硬件的，常量是如何转换成逻辑数值的，语句是如何转换成硬件的。下面我们将详细分析下综合中各个VerilogHDL语句及器件的综合原则。1、储值单元的综合原则对于FPGA上的实际硬件主要有三种基本的存储数值的单元分别是：线网、锁存器、触发器，分别对应于Verilog语言中的net和reg类型关键字，其综合的映射原则是对net类型的关键字在综合后将映射为FPGA片上的布线资源，而reg类型关键字综合后将根据行为级描述的逻辑可能被综合为片上布线资源、触发器资源或锁存器资源即存储器资源，当reg在always模块中即时序电路逻辑中以临时变量被使用时其综合结果取决于设计的具体逻辑实现[44]。2、连续赋值语句的综合原则连续赋值语句综合图示连续赋值语句（ContinuousAssignment）语句对应的硬件实现如图2.7所示，在EDA工具中综合器会将连续赋值语句右侧的表达式逻辑进行提取，用于驱动该语句中左侧的net类型资源，连续赋值语句的右侧表达式中逻辑功能实现后，赋值表达式中信号的变化随机会被传输至赋值线网型数据的取值上[16]。3、过程赋值语句的综合原则过程过程赋值语句（ProcejuralAssignment）语句对应的FPGA的硬件实现是：从赋值语句也就是‘=’或‘<=’的右边提取出逻辑，用于驱动赋值语句左边的变量，但是有个语法前提，就是用于驱动赋值语句左边的变量是reg类型，且只有Always语句块中的过程赋值语句才可以被综合器综合，若出现在initial语句块中将被仅用于仿真。过程赋值语句有以下两种分别为：阻塞赋值语句（BlockingAssignmentStatement）、非阻塞赋值语句(Non-blockingAssignmentStatement)阻塞赋值语句综合阻塞赋值语句综合图例阻塞赋值语句简而言之就是在一个always语句块中，语句是从上到下顺序执行，综合时将会在依据赋值语句逻辑原理结合always块的触发器来实现逻辑综合，阻塞赋值语句RTL级描述与电路映射如图2.8所示。非阻塞赋值语句综合非阻塞赋值语句综合图例非阻塞赋值语句简而言之是在一个always语法块中，语句是并行执行的，非阻塞赋值语句RTL级描述与电路映射图2.9所示，对比上图可以得到对于一个Always块中的阻塞或非阻塞语句的综合最大的区别在于是并行还是顺序执行，但是对于综合后器件的数量，也就是我们所关心的资源数量并没有明显的影响，主要影响使用资源数量的关键在于赋值语句右边的驱动逻辑的组成。4、逻辑运算的综合原则进行逻辑综合时VerilogHDL语法中的逻辑运算符将会被直接转换成对应的基本逻辑门,如与或非、异或门等[43]。非阻塞赋值语句综合图例5、算术运算符的综合原则：在VerilogHDL语句中支持各种算术运算符如、、、等，在进行综合时RTL构造器会将其构造为加法器、减法器、除法器等源语器件。如图2.11所示为加法运算综合后器件映射图。算数运算符综合图例always语句块的综合原则在VerilogHDL设计中用的最多的就是always语句块，always关键字用于描述组合逻辑和时序逻辑，两种逻辑的主要区别在于敏感触发信号是否为时序信号，alway语句块中多包裹其余语法使用，单独使用不具有任何意义。if及条件语句的综合原则在VerilogHDL语句中的单条if-then语句常表达条件判断行为，故常被综合为MUX数据复选器，若为嵌套组合预计，常会被逻辑综合为收到触发条件控制的纯组合逻辑电路，如图2.12所示。if语句综合图例case语句的综合原则在VerilogHDL语句种，case包含两类条件表达式和选择项表达式，case语句可以看作是嵌套的if语句，所以其综合原则与嵌套if条件语句类似，如图2.13所示。case语句综合图例10、循环语句的综合对于循环语句，在逻辑综合时依据行为级的逻辑描述若其循环体内部逻辑不存在依赖性或关联性，则循环体内部语句将被展开为并行语句后进行逻辑综合，且语句执行先后顺序不会影响到最终的结果，如：For循环语句在逻辑综合后会依据循环体内部的逻辑构成级联的链式组合逻辑电路。循环语句综合图例11、函数语句的综合在verilogHDL语句中。function如C++的内联函数一样，在逻辑综合时，将会被直接展开成对应代码放在对应的模块中。任何在函数中定义的局部变量都被当做纯粹的临时变量，且由于其所表述的行为级逻辑并没有时序控制，因此function逻辑综合结果是组合逻辑，如图2.15所示。函数语句综合图例FPGA技术映射在综合优化过程中，技术映射是将技术映射作为综合优化的重要组成部分，也是FPGA支持软件工具EDA中最为关键的部分，技术映射是面向FPGA的EDA软件中至关重要的一步，其映射和优化结果对整个FPGA设计流程中下一阶段的布局布线等步骤有着决定性的作用，进而最终影响芯片的性能成本，故而已形成了以面积优化、时间延迟优化、功率优化以及面积和延迟同时优化为目标的一系列研究成果。工艺映射是指将不依赖于任何工艺的门级网表转换为由特定FPGA逻辑单元所表示的网表的过程[43]。本文此处所提及的逻辑单元均为使用LUT、MUX、FF作为基本逻辑单元所构成的FPGA的最小逻辑单位。技术映射是从RTL级出发对所描述的硬件行为进行逻辑转换的重要步骤，因此对FPGA片上最终实现的电路的面积、功耗、运行速度相当重要。工艺映射流程图技术映射的本质是将电路所描述的行为逻辑所对应的布尔网络，通过算法与特定FPGA的工艺器件库，将布尔网络转换为对应FPGA最小逻辑单元的过程。其目的是对用户输入的RTL级电路经过逻辑综合后构成的布尔网络通过先进的算法来进行划分，使得原设计中的每一小块电路的逻辑功能可以借由FPGA的一部分CLB来实现。我们可以参考图2.16来进一步对技术映射的流程进行分析。由图2.16可以发现工艺映射的流程是完成门级网表的输入至实际工艺下可实现的工艺网表输出这一个过程，门级网表我们可以理解为是由一组组逻辑表达式构成的的布尔网络，该布尔网络图上的个节点均为任意的逻辑功能组合单元。由图2.16可以看出划分是工艺映射流程的第一个步骤，划分的本质是对已经获得的用以表示电路逻辑的布尔网络依据其所实现的功能逻辑进行分割，划分步骤会将将由多个输出组成的布尔网络等价转化为一组仅有一个输出的布尔子网，从而将布尔图进行简化，进而提升技术映射的效率及简化其过程。当划分完毕，原始布尔网络所代表的逻辑电路便转换为一个个仅具有单一输出的布尔子网络。根据工艺映射算法的不同，原电路均可被构造成由一张DAG图或一组树来表示[15]。电设计转化为DAG图逻辑分解是将电路进行逻辑等价的转换，目的是将复杂的逻辑单元用基本的逻辑形式表达，以便于以后的匹配和映射[45]。布尔网络在经过逻辑分解操作后，其原本的原本的布尔网络的节点之间将只会保留基本的函数逻辑关系。逻辑划分经过划分和逻辑分解之后下一步就是匹配。而匹配则是在划分与逻辑分解后，将与工艺无关的布尔网络与工艺库相结合。其本质就是使用FPGA厂商所提供的实际工艺下可实现的逻辑单元或更高级的片上器件对布尔图进行等效匹配，是把理想的布尔网络用库中的实际单元实现的过程[45]。目的是使得FPGA片上所对应的实际工艺器件最终可以实现布尔图所表述的电路逻辑。在EDA流程中技术映射的最后一步是对匹配后的树或图结构进行器件的覆盖，其作用是根据实际电路的要求，在所有节点的多种可行匹配集中选出足够的匹配来覆盖整个网络的所有节点[47]。既依据一定的规则分离出部分子图进行匹配，并从匹配集中选出库单元，完成布尔网图到工艺网表的实际映射。要保证每完成的一组匹配单元，其每个单元的栅入和其他组的匹配后的单元的栅出进行相连，且每组单元间不存在重叠关系。工艺映射的核心优化算法是如何分割布尔网表、如何选取匹配集，从而满足用户所期望的对面积、功耗、速度的需求。最后以实际的门级网表的形式输出，所构成的网表也是在高密度FPGA原型验证系统中基于网表进行FPGA划分的基本输入。器件覆盖下面我们通过分析具有代表性的FlowMap的LUT工艺映射原理，借由以下实例对工艺映射流程进行进一步说明。无论多么先进的FPGA芯片中其基本资源中其查找表的输入数是既定的，如LUT1、LUT2、LUT3等，基于图2.2可知，查找表可以实现所有输入不大于其本身输入数量的任意逻辑。而FPGA的工艺映射的目的就是要从目标电路的布尔逻辑图中不断划分出输入数在组成该FPGA的最小查找表之内的等效逻辑。首先将图2.17中的门级网表转换为图中的DAG有向图的方式来表示。最上层的节点成为主输入，最下层的节点称为主输出，从而可以得到如图2.18所示的映射范围，图2.19所示为在映射范围下进行切分和标注求解的过程及将映射范围的有向图依据主输入及主输出进行逻辑分层，从而标注拓扑序列，进而完成覆盖。其具体步骤伪代码如图2.20所示。Mapping算法伪代码通过上述计算可以得到每一层的标注为依据上层计算所得出的最小值，从而可以保证使用最浅的逻辑层深度。之后对电路的主输出开始进行器件映射，每映射一个后主输出将发生变更，对所有的主输出依次进行算法映射后，便可得到相对应的查找表及网表图，如21所示。对目标电路进行切分和映射方法的历史发展中，有很多根据不同工艺、不同划分或覆盖思路形成的优秀技术映射算法，如：哥伦比亚大学SteveWilton团队所开发的更倾向于改善逻辑电路映射后功耗的EMAP算法、多伦多大学StephenBrown团队所开发的应用比较广泛的IMAP算法等[2]。机器学习模型与资源估计方法本章前两节主要叙述了关于面向FPGA的逻辑综合与技术映射的基本流程，为本文后续建立基于传统意义上的快速资源估算模型建立基本方法论，而本节将主要介绍再上述两节所述的传统框架基础下，结合机器学习来构建的FPGA设计资源估算的可行性与模型的方法论，包括机器学习与EDA相结合的可行性、随机森林模型，人工神经网络（ANN）及学习模型评估指标。机器学习与EDA结合的可行性机器学习是一门多学科交叉领域，其涵盖概率论，统计学，近似理论和复杂算法等知识，使用计算机作为工具并致力于真实实时的模拟人类学习方式，并将现有内容进行对应知识体系划分来有效提高学习效率的方法[3]。通过机器学习的定义同时结合FPGA设计逻辑综合流程及技术映射原理，我们可以得出对于大型FPGA设计在综合阶段存在的资源估算的时间长的弊端，我们可以使用机器学习方式以VerilogHDL的各种语法及参数的数量及值作为特征，Vivado综合的资源数值为标签，来训练监督学习的回归模型，从而实现快速的资源估算。基于机器学习实现资源估计方法的流程如图机器学习实现RTL级设计资源估算流程特征提取方法对于机器学习或人工智能而言，模型的选择与好坏并不能最终决定预测精度的上限与下限，最终模型的上限取决于特征与样本的优劣，所以特征对于机器学习模型的学习与预测是相当重要的。特征的优化和清洗将对机器学习最终的结果产生较大的影响，所以在选择学习模型与训练器前需要先对样本进行特征提取，对于在RTL级进行资源预测这个命题，我们需要在RTL级提取尽可能多的语法特征，从而确保不会丢失重要特征，虽然会获得很多无关特征或冗余特征，但在进行特征工程及训练模型迭代时可以进一步对特征进行优化，从而保留相关特征和重要特征。由于RTL级设计是基于VerilogHDL语言开发的编程语言，所以我们采用尽可能细致的特征提取方法，即将VerilogHDL抽象为数据结构，并结合IEEE1800所示Verilog语法标准，通过进一步解析数据结构来尽可能全面的提取VerilogHDL语言的语法特征，从而构建在基于2.2节原理的基础上的合理特征集。具体的特征提取算法与特征提取的构建将在错误!未找到引用源。进行详尽的阐述。随机森林在机器学习中，随机森林是通过包含多个基于决策树弱学习器进而构成的强学习器。随机森林是LeoBreiman和AdeleCutler发展出推论出随机森林的算法。这个算法是多学习集成学习的典型模型。BaggingBagging是并行式集成学习中比较具有代表性的机器学习方法，它基于自主采样法，即给定一个数据集拥有固定m个数据样本，学习算法先随机拿出一个样本放入采样集中，再通过将该出去的样本放回数据集，使得在每一次采样时均有可能取到之前的样本，这样经过覆盖所有样本的次数的随机采样，我们可以获得包含m个数据的采样机，初始训练集中有的样本在采样集中多次出现，有的则从未出现。由式（2-1）可知采样集中将会出现训练集中约三分之二的数据。(2-1)通过上述算法可以采样出个含个训练样本的采样数据集，随后基于每个采样集训练出对应的弱学习器，再通过将这些基学习器的学习结果进行进一步的结合。从而实现Bagging算法的流程。我们进一步分析其算法复杂度来估计其学习效率，若基学习器的计算复杂度为,则通过上述方法可知该算法的复杂度约为,考虑到采样与决策时的时间复杂度很小，而通常是较大样本数据而言是一个较小的常数，因此，训练使用Bagging集成的学习器与使用一个弱学习算法训练所需要的时间复杂度同阶，从而证明Bagging是具有较低时间复杂度、较高学习效率的集成学习算法。从偏差-方差分解的角度看，由于Bagging学习算法的主要关注点在于如何降低方差，因此它在受样本的干扰较大的学习器上的优秀表现更为明显。Bagging的算法伪代码如图2.22所示。Bagging算法伪代码决策树决策树是一种常见的机器学习算法，决策树学习的目的是为了构造具有泛化能力强，处理未见示例能力强的二叉树，该树能够对泛化样本进行正确且合理的决策，其基本流程遵循简单且直观的“分而治之”策略，决策树学习基本算法伪代码实现如图2.23所示。决策树学习基本算法显然基于图2.23中的伪代码可知决策树是依靠递归来生成的算法过程。在决策树的决策过程中我们更关心的肯定是如何划分，或者说如何选择最优划分属性，即结点的“纯度”，决策树中衡量一个结点的纯度的高低有以下几类指标：信息熵公式：(2-2)增益率公式：(2-3)基尼指数：(2-4)随机森林随机森林是Bagging的变体，是基于Bagging的算法思路扩展而来，其本质也是一种由多个弱学习器（决策树）所集成的强学习器。RF在通过在训练过程中引入随机属性选择，并以Bagging使用决策树作为基本弱学习器的基础上构建得来。具体来说，传统决策树在选择划分属性时是在当前树节点的特征集合（假定有个特征）中选择一个最优属性；而在RF中，对基决策树的每个结点，先从该结点的属性的集合中随机选择一个包含个属性的子集，然后再从这个子集中选择一个最优属性用于划分[30]。随机森林在多种多样的决策与回归任务中展现了超乎预期的强大的性能，随机森林通过增加随机属性的选择实现了对Bagging做微小的改动，但继承了Bagging中所具有的学习多样性，随机森林中基学习器的多样性不仅来自样本扰动，还来自属性扰动，这就使得最终集成后的强学习器的泛化性能上的提升，其可通过弱学习器之间差异度的互补来增强。通过对随机森林算法进行研究，发现随机森林相较于传统的机器学习模型具有训练速度快、预测效果好、泛化能力强、鲁棒性强等优势，可有效避免模型过拟合的现象出现，适用于高维数据的处理。随机森林的回归预测结果是棵决策树的预测结果取均值而得到的，它满足以下定理：定理1当时，(2-5)式(2-5)右侧部分表示随机森林的泛化误差，将其记为。则表示一课决策树的平均泛化误差即(2-6)定理2对所有随机生成的训练集都有,(2-7)式(2-7)中是在与相互独立的情况下，残差和的加权相关系数。上述定理1和定理2给定了使随机森林精确的前提是：残差间的相关系数低以及错误决策树数目较少。为降低决策树的平均误差，随机森林回归选择对相关系数进行加权处理。随机森林的具体算法步骤伪代码可概括为下图2.24所示。随机森林模型算法伪代码人工神经网络人工神经网络是深度学习的核心。其在当代用途广泛、功能强大且可扩展，使其非常适合处理大型和高度复杂的机器学习任务，例如对数十亿张图像进行分类，例如Googal的Images，或者为语音识别服务提供支持[44]。下面将主要介绍人工神经网络中的感知器计算原理及多层感知机。感知器神经元计算图示感知器架构是最为简单的神经网络基础架构之一。结构如图2.25所示，图中所示神经元结构称为阈值逻辑单元（TLU），也称为阈值逻辑值单元（LTU）。该神经元的IO均为数字信号，而不是二进制的0/1值，并且每个输入连接都有对应的相关联的权重值[44]。TLU来计算输入的加权和，其中阶跃函数将使用加权和来输出学习结果,在感知器中公式（2-8,9）所示为最常用的阶跃函数，Heaviside阶跃与sng函数。(2-8)sgn\left(z\right)=-1,&z<00,&z=0+1,&z>0(2-9)感知器是一层阈值逻辑单元单元所构成的，每个阈值逻辑单元单元与全部的Input信号相连接。当一层中的所有神经元都连接到上一层中的每个神经元时，该层通常称为全连接层。所有输入神经元形成输入层，用以与感知器的输入数据互联。同时，每一个神经元会特别增加一个额外的偏置特征，并通过偏置神经元（一种特殊类型的神经元）来保证其输出始终为1。具有，作为输入的感知器如图2.26所示。该感知器可以将输入的数据同时借由输入层来分为三个不同的二进制类，并作为输出。感知器结构图示借助线性代数的数学关系，通过公式（2-10）可以计算出当存在多个输入时计算全连接层感知机的输出结果，且计算是高效的。(2-10)关于感知器的训练算法，其很大程度上受到规则的启发，当两个神经元在某种情况下经常进行交互时，这两个神经元之间的互联关系便会得到加强，也就是说，当两个神经元被同时激活时，期间的权重将会增大。更具体的说，感知器每一次训练被送入一个训练样本，同时每个训练样本进行预测。从而对产生错误预测的神经元的权重进行调整，如增大，从而增强来自输入的连接权重，这些神经元权重将有助于正确的预测，具体如公式（2-11）所示。(2-11)多层感知机多层感知器也称深度人工神经网络，则是在上述感知器的基础上，由一层输入层、一层或多层TLU即隐藏层和一个TLU的最后一层即输出层组成。靠近输入层级常称为较低层，靠近输出的层通常为较高层。每一层都拥有一个偏置神经元并全连接下一层神经元，除输出层外。多层感知机结构图示常见的人工神经网络结构是如图2.27所示的多层级结构，每层的单个神经元与下一层神经元之间实现全连接，同层的神经元间也不会出现相互之间的连接，跨层的神将元之间也不存在连接关系。这种网络结构通常称为“多层前馈神经网络”，即多层感知器，其中输入层神经元用以输入外部输入，隐藏层和输出层用以处理神经信号，最终结果由输出层输出预测结果；换句话说，输入层只接受输入而不对输入样本进行任何处理，隐藏层和输出层包含函数神经元。神经网络的学习过程是根据训练样本的输入来调整神经元将之间的权重值和各个功能神经元的阈值，神经网络所学习的主要内容包含在连接权重和阈值中。当它进入ANN内部的隐藏层时，它处于深度神经模式（DNN）中，分为前向传播和后向传播。反向传播是目前使用的学习算法。使用有效的技术自动计算此级别。只穿越电平两次，即一前一后传播算法可以计算每个模型参数的网络误差梯度。换句话说，可以用以学习如何修复每个连接权重和每个偏移项错误。当找到这些梯度时，常规梯度会执行下降步骤，直到梯度难以拟合，然后它会更新整个过程。常见的DNN网络超参数与典型值如表2.1所示，后续基于机器学习建立快速资源估算模型会进一步对超参数进行调整及介绍。典型多层感知机架构机器学习性能评估指标评价模型性能的指标有很多种,需要我们根据特定的任务选择合适的评价指标。不恰当的评价指标不仅无法模型进行评估,反而会造成误导,得出错误的结论。一个评价指标可以展现模型在某一方面的能力,所以一般情况下,需要构造多个评价指标,来对模型进行全面的分析。针对大型RTL设计的FPGA资源估算模型的需求分析，其为回归问题模型，对于回归问题模型的特点有如下几种模型性能评估指标：均方误差均方误差是在回归问题的机器学习模型中常见的衡量泛化误差的学习指标，均方误差是反映样本真实值与实际预测值之间差异程度的度量[20]。评价点估计的常用标准即均方误差均方误差公式如(2-12)所示。(2-12)式(2-11)说明，均方误差由点估计的方差与偏差的和的平方两部分组成。均方根误差均方根误差，也称做标准误差，均方根误差是预测值与真实值偏差的平方与采样次数的比值的平方根，在实际测量中，观测次数总是有限的，真值只能用最可信赖（最佳）值来代替。标准误差对一组测量中的特大或特小误差反映非常敏感，所以，标准误差能够很好地反映出机器学习的精密度。这也是均方根误差在工程领域被广泛使用作为评估指标的原因。因此，均方根误差是常用来度量预测值同真实值之间的偏差，其计算过程公式如(2-13)所示。(2-13)3、R-Squared是统计学里常用的统计量，的名字是coefficientofdetermination，另一个名字是Nash–Sutcliffemodelefficiencycoefficient。给定一系列真值和对应的预测值，的定义如公式(2-13)所示。(2-14)的含义是，预测值解释了变量的方差的比例差距，其度量的是预测值对于实际值的拟合程度。进一步理解，假定的方差为1，则使用表示该模型之后，的残差的方差减少了多少。比如等于0.8，则使用该模型之后残差的方差为原始值方差的20%。：最理想情况，所有的预测值等于真值。：一种可能情况是"简单预测所有值等于平均值"，即所有都等于真实值的平均数。：模型预测能力差，比"简单预测所有值等于平均值"的效果还差。这表示可能用了错误模型，或者模型假设不合理。的最小值没有下限，因为预测可以任意程度的差。因此，其范围是。快速RTL级资源估算模型的设计与实现本章将结合对第二章所述的FPGA逻辑综合原理与技术映射原理的研究，使用C++语言设计并构建基于传统框架下的快速资源估算模型，并基于该模型结合第一章所述高密度FPGA原型验证系统所需要解决的关键问题及基本需求增加算法优化与约束项优化，实现符合基本原理与能够解决实际工程问题的快速RTL级设计资源估算模型。Verific及相关数据结构Verific平台介绍VerificDesignAutomation是Verific公司开发的一款用于解析与构建SystemVerilog、VHDL和UPF的解析器平台，使其客户能够以低成本快速开发高级EDA产品。Verific分析创建解析树并执行类型推断以解析标识符的含义。解析器/分析器模块支持整个系统VerilogIEEE1800（包括VerilogIEEE1364）、VHDL-IEEE-1076和UPF-IEEE-1801语言。生成的解析树/数据模型附带了大量成熟的C++API。Verific的解析器平台分发为C++源代码，并在所有32位和64位UNIX、Linux、MAC和Windows操作系统上进行编译。Verific的解析器平台目前在全球许多公司的生产和开发中使用，从EDA初创公司到《财富》500强半导体供应商等。我们将使用Verific，并基于Verific生成的语法树来完成后续的模型构建操作，由于Verific具有当前行业认可的Parse和Analyze功能，并基于C++平台实现开发，提供广泛的语法树API作为二次开发的基础，故我们选择在Verific的基础上实现可商业化使用的快速资源估算模型。首先需先对Verific所构造的基本数据结构进行相关了解及分析，从而选择构造合理的数据机构及API进行模型建模实现；Verific主要数据结构分为以下两个主要部分：基于RTL的语法树结构和基于Netlist的Database结构,下文将主要介绍构建资源估算模型使用的主要数据结构。Parse-Tree数据结构Verific在RTL级别VerilogVHDL的语法树Parser上有丰富的经验，通过对语法进行文本级别的流解析在内存中构建层次树（Hierarchy-Tree）结构用以表述VerilogVHDL语法信息，层次结构树（hierTree）是设计层次结构的独立于语言的表示形式。设计的层次结构由Verilog语言中使用的模块实例化和生成语句定义。随后在本文档中，我们将此数据结构称为hier树或Parse-Tree。这个层次结构树结构是一个轻量级的静态详细层，位于分析的解析树之上，保存着关于各种设计对象的关键信息。树的每个节点都存储有关它所表示的设计对象的信息，以及该层次结构路径中特定于该设计对象的参数和泛型等常量的值。语法树结构主要由以下几种类结构：①VeriTreeNode是所有Verilog树节点元素的基类。因此，它包含行文件信息和在所有解析树节点上的数据信息。其中VeriExpression、VeriIdDef、VeriStatement等中非常常用的派生类，用于对VerilogDesign作进一步数据构造，下图为VeriTreeNode的继承示图：VeriTreeNode类视图②VeriIdDef是标识符定义树节点，是标识符节点的基类。Verilog中声明的任何标识符都将成为VeriIdDef结构类。下图为VeriIdDef的类关系视图，其中VeriidDef的继承类VeriModuleID、VeriOperatorID、VeriInstID等为进一步对Verilog语言语法的描述类。③VeriExpression是所有Verilog表达式解析树节点的基类。其继承类用于描述包括常量、名称引用、运算符以及出现在实例化端口关联列表和模块端口列表中的端口关联信息等。下图为VeriExpression的类关系视图。④VeriStatement是所有Verilogstatement节点的基类，包括所有Always语句块等语法⑤VeriModule位于任何Verilog解析树的顶部，是模块节点的基类。它以解析树的形式表示一个完整的Verilog模块。VeriModules存储在静态哈希表中，该哈希表表示已分析的所有Verilog模块。VeriModule类视图上述5种数据结构为Verific中最基本的基类，所有语法树的语法表述及存储均可由其中一种进行表述，在本文中所有RTL代码的语法将会使用上述类进行语法树处理及构造，从而获取正确的设计层级及内容。Netlist数据结构由于传统技术映射是在网表级的基础上通过器件库使用图划分算法，将网表节点映射至对应的器件，所以为构建基于传统方法的快速资源估算模型，也需要使用Verific的Netlist数据结构来实现，VerificNetlistDatabase数据结构如下：Verific-Netlist-DataBase数据结构图①Libset是一个单例对象（只实例化一次的对象），它包含（char\*）lib\_name到Library\*映射的映射。Libset没有所有者，因此被视为全局对象；②Library是Libset拥有的对象，它包含所有Module模块命名到cell指针映射的映射。库对应于它的VHDL或Verilog等价物。例如，在VHDL中，将用户实体/体系结构对分析到一个名为“user”的库中，然后详细描述解析树，那么将创建一个名为“user”的库对象并将其添加到全局库集中。③Cell是库所拥有的对象，它包含Netlist\_name到netlist\*映射的映射。当处理Verilog时，单元是包含单个网络列表的模块。就VHDL而言，单元是一个可以有多种体系结构（网络表）的实体。④网络列表是数据库层次结构中的关键组件。它是一个由单元拥有的对象，包含其他网络列表实例化形式的功能（通过称为网络、端口和端口引用的对象连接在一起）。它还包含一组自身正在实例化的引用（从其他网络列表中）。如上图所示，基本数据结构为LibSet用以存储所有的Library库，Library库为设计网表存储基本数据结构，Cell结构对应为VHDL代码中Module模块，每个Cell尤其仅有一个对应的Netlist结构，Netlist结构用以存储该Module所有互联信息及实例化信息，如：Port、Net、Instance，包括Net位宽、方向等。基于Verific的上述数据结构以及相关的API，使得我们可以在后续基于C++语言实现传统快速资源估算模型提供基本数据基础。资源估算模型流程与构造本节将主要介绍使用C++构建基于传统框架下的快速资源估算模型的算法分析及具体实现步骤。资源估算模型流程设计基于第二章节所述逻辑综合及资源映射原理，我们对模型的构建做出进一步的分析，模型输入为RTL级设计文件即VerilogVHDL文件，模型输出为该设计所需FPGA资源数目，其中对于面向大型RTL设计Partition中所需的最重要的关键的资源为FF及LUT，文件流如下图所示。快速资源估算模型文件流基于TechnologyMapping的原理进一步分析，为实现传统框架下较为准确的资源估算，则需要将输入的原始设计从RTL级代码转换至Netlist级别才能实现进一步的TechnologyMapping，但是由于获取最直接的网表解构就需要使用综合器对RTL设计进行综合，通过前面对综合原理的分析，可知道综合主要分为以下几步，即RTL设计细化，Flatten，综合优化，综合约束，技术映射等几个步骤，但由于需要面向大型FPGA设计，综合器耗时太长，所以我们选择通过使用Verific来对设计进行静态细化及初步细化，使RTL级设计在内存中转变为伪网表，通过在内存中对伪网表进行Flatten操作，将伪网表所有的层级结构UnGroup至顶层，从而实现与基本综合器相对应的操作的快速版本，对Flatten至顶层的设计依据TechMapping原理使用已经获得的器件库进行器件匹配，基于本节分析及初步设计结果，着手使用Verific对传统资源估算进行进一步分析及模型构建，具体流程如图所示，关键操作及相关原理将在下文着重叙述。快速资源估算模型流程图示资源估算模型构造实现我们以上一节中对快速资源估算模型分析的流程进行进一步的分析，同时进行实现：1、Parse-Design操作通过将原始设计使用Verific的Parser接口进行设计读入，由于需要商业使用，需要考虑多种用户输入情况，首先考虑常见的通过Filelist进行读入，并支持对于文件正确性校验及报错，并支持输入文件包括.sv\.v\.vh\.svh等多种文件输入，同时支持通过路径进行设计输入，并对设计进行正确性校验，如图3.6所示。快速资源估算模型Parse-Design流程2、Elaborate操作通过对上一步骤获得的完整输入设计的Parse-Tree进行Elaborate，从而获得对应设计的由VerificGTecCell构成的伪Netlist的设计文件，Elaborate后的设计语法依旧满足VerilogHDL的语法信息，RTLElaborate会使用门级的网表来表示之的RTL级代码。Elaborate会将RTL级代码进行逻辑展开，获得具有符合电路逻辑的门级视图网络和门级逻辑电路的VerilogHDL表述，该步骤为FPGA设计流程中综合的前置步骤，为下一步进行优化的技术映射做好基础工作，Elaborate操作如图3.10所示。RTLElaborate操作图示3、Flatten操作在步骤2的基础上，我们将对设计进行Flatten操作，Flatten操作顾名思义是将现有包含层级关系的层级设计进行展平操作，也可以称为ungroup操作，参考下图样例可以更好的说明Flatten操作对于设计的操作，如图3.8所示。Flatten操作电路图示Flatten操作树结构图示在FPGA设计综合流程中Flatten操作会将一些层次树底层的Module、逻辑提升至顶层，将可以合并的逻辑进行合并，同时将一些冗余的互联Net进行优化，为进步一的综合优化和器件映射做好准备，因为对于FPGA或者数字集成电路而言，它是没有层次的概念的，所以Flatten操作可以更好的让寄存器级设计与物理器件之间建立更进一步的联系。4、基本器件库由于我们中所设计的快速资源估算模型是面向Xilinx的FPGA的模型，所使用的基本器件库是Xilinx公司Virtex系列器件库，其中主要包含的为基本逻辑单元即CLB，具体为MUX、LUT1~6、各形式的触发器即FF、FDR等，一些IP器件由于知识产权等问题无法直接使用，如图3.10以LUT、FF、MUX为例进行说明。RTL级基本器件库5、TechnologyMapping对步骤4中完成Flatten后的网表文件进行TechnologyMapping，具体原理如2.2.1节中所述，具体操作为：将Flatten后的Netlist网表文件依据设计逻辑对将步骤4中的器件库的器件进行匹配和覆盖使用算法为EMAP，匹配后的网表将不存在基本的门级逻辑（布尔逻辑），而将会被器件库中的基本逻辑器件所替代，如MUX、D-FLIP-FLOP触发器、4-输入LUT器件等。6、估算结果展示下图为以上述思路构建的快速资源估算模型的初步估算结果，如图图3.11可以获得输入的RTL设计所使用的FPGA片上LUT1-6、BUFG、FF等器件资源数量。快速资源估算模型计算结果快速资源估算模型优化通过图3.11与Vivado综合后结果对比，可以发现，在无优化的情况下仅对设计进行细化及基本技术映射虽然可以得到基本的资源，且在速度上得到了有效的提升，但是存在较大的误差，通过分析Xilinx的综合器Vivado细节及TechMapping，我们可以知道在与FPGA板载器件进行基本逻辑单元及触发器映射时存在优化项，可以理解为约束项，如：面积约束、功耗约束、时序约束、逻辑门约束等多种约束；结合面向大型RTL设计分割流程的实际需求，增加以下优化方案。BottomToUP算法的设计与实现由于使用Verific进行Elaborate处理及Flatten后无法得到每个层级对应的资源数量，且对整个设计进行Elaborate及Flatten所花费的时间较高，同时由于该资源估算模型面向RTLPartition进行使用，所以需要每个层级每个Inst的资源用于确认RTLPartition时正确的分割边界，所以我们从VerilogHDL与FPGA片上资源映射关系角度出发，在不考虑优化的前提下，每个Module的资源是由其所例化的Instance对应的Module的资源与该Module本身的胶水逻辑所组成，如下图样例所示，红框标注分别为例化Instance与胶水逻辑。RTL代码中的GlueLogic和Instance以上述分析为出发点，我们希望可以结合VerilogHDL设计的结构化建模方式，以该方式为参考，能够更加准确，更加快速的计算每个Module和Instance节点的资源，我们以RTL设计及计算资源所需数据为出发，设计以下数据机构用于存储每个节点的资源，如图3.13所示。BottomToUp算法数据结构RTL设计树结构示意图通过以上述数据结构，用以实现Bottomtoup的计算方式，通过对RTL设计树层次如图3.14所示，进行树遍历算法及递归算法构造存储当前Parse\_tree对应的树结构及查找表，该查找表用以存储已被计算过的Module的资源，同时对多个不同参数例化的Module分别计算资源同时通过匹配算法将树结构上的节点与已计算资源节点进行匹配，本算法可以最大程度的减少重复计算，同时为减少网表结构带来的peakmemory过大问题，每次计算都将使用不同的库进行技术映射，并在计算完毕后进行释放。另外由于本算法设计数据结构从资源复用角度出发，在用户的设计发生迭代时只用重新计算资源变更层级以上部分层级的资源，从而减少了当设计变更引起的重复运算。并且为支持用户在不同的设计层级对设计进行运算的需求，数据结构中的level变量用以存储当前设计层级，若输入指定不同层级时，可以根据指定层级分层次进行计算，从而加快计算速度。算法伪代码如图3.15所示。BottomToUp算法伪代码通过上图所示算法，在3.2节实现的快速资源估算模型的基础实现快速层次资源估算，输入为RTL级VerilogHDL设计，输出为该设计每个层级所使用的资源数量及相对应的树结构层次，若在输入时提供FPGA型号和资源，程序也会评估该设计所需要的FPGA的数量。如图3.16所示。快速资源估算执行图示可复用估算结果的Json文件优化后快速资源估算模型输出报告图3.18中为开源设计openrisc经过该快速资源估算模型新增的Bottom-to-Up算法计算所获得资源数量与每个Hierarchy对应的资源数量，从图中我们可以看到经过资源计算后设计的具体层级，每个层级前标注序号为该Instance节点名称及对应的树层级结构层次，节点后中括号内分别为RTLPartition流程所关注的十种资源数目，同时可以获得在给定的FPGA\_Type、FPGA资源阈值下该设计所需的FPGA数量并且会输出如图3.17所示的一份Json文件用以存储每个Module对应的资源，便于复用。经过同Vivado进行对比后发现，虽然该算法已经很好的解决了资源估算的速度问题，但是在准确度上还存在可优化空间及约束项，经过对逻辑综合与技术映射技术的进一步分析我们将在Elaborate后增加优化策略与约束策略，从而使估计结果更加符合实际应用场景。冗余逻辑优化及合并在进行逻辑综合和技术映射时，EDA综合器会对集成电路的规模、运行速度进行优化。综合的优化过程中会去掉常数输入、悬空输出等信号，还会针对源文件中未连接的输入或具有多个驱动的输出等问题报告错误或者警告[16]。若EDA工具发现设计存在部分缺陷如非关键输出悬空能，通常EDA工具都将依据自己的方式进行优化修复，但多所带的结果是可能优化结果并不满足设计者的初衷。综合常见的优化过程还对逻辑进行组合置换，从而进一步减少集成电路所需FPGA资源、功耗、并尽可能提升系统运行速度与减少阻塞。通过前述步骤得到的快速资源估算模型，虽然已经可以得到资源的结果，同时若使用Bottom-to-Up的算法可以得到每个层级的资源结果，但是我们通过第二章的综合和技术映射的原理可以得知，在实际的综合和技术映射过程中，是对设计进行了大量的优化和处理的操作，所以如果我们不对当前的快速资源估算模型进行优化和约束，那么该模型的结果是不可用，没有价值和意义的。所以我们为使我们资源估算的精确度更加高，所以需要符合逻辑综合和优化的过程，通过分析逻辑综合流程，在逻辑综合的工程中会对冗余逻辑进行优化，比如一些复选器或单纯的数值传递等非有效逻辑进行合并。快速资源模型将会查找并发现没有负载或被禁用的逻辑（例如，当选择线保持高位时，驱动多路复用器上第一个选择的逻辑），并假设消耗了零资源。一旦找到一个这样的元件，所有驱动逻辑也会被移除，同时采取预防措施确保其他逻辑也不会被该元件驱动。最小面积优化方法通过本文第二章中对逻辑综合过程中优化步骤的分析，我们知道在综合开发环境会对RTL代码进行很多优化，从而提高设计的性能、减少硬件资源消耗等，所以在快速资源估算模型中也需加入相关优化项，来对RTL设计进行进一步的优化，从而使得资源估算模型的结果更加的精确。首先我们考虑资源共享，即在互斥条件下共享算术逻辑单元，主要有关系运算、加法运算、减法运算、除法运算等。LUT-5BaseFPGA结构图示根据FPGA的结构，可以知道不同系列、不同世代、不同型号的FPGA在逻辑单元的组成上是有很大区别的，比如对于赛灵思的VU440，其基本逻辑单元即CLB单元是以LUT5为基础，同理也有以LUT1、LUT2、LUT3、LUT4、LUT6为基础，支持至最高为LUT8的FPGA，由于LUT1~8之间存在换算关系，LUT的本质就是如图3.18的2进制查找表，故而2个LUT1可以合并组成为一个LUT2，同理以LUT5为Base的FPGA，可以使用多个LUT5来组成LUT6等高维度资源，从而有效的是缩小FPGA布线后使用的面积即器件使用的数量。4输入查找表真值表在进行资源共享优化时我们遵循以下原则：运算符和操作数均相同，则一定可以贡献逻辑单元。对于具有相同运算符的表达式，且操作数或操作表达式相同时，为实现逻辑共享我们可以加入一个多路复用开关来完成。相同运算符，完全不同的操作数，可以共享逻辑单元[16]。不同的运算符，相同的操作数，可以共享逻辑单元。对于运算符不相同的运算表达式，若其有且仅有一个公共的操作数，则为实现逻辑共享可以引入MUX来实现。不同运算符、不同操作数，可以通过引入两个多路开关，从而实现逻辑单元的共享。除去资源共享优化外，本文基于对RTL代码的分析，通过对公共子表达式增加逻辑优化合并，使并借由数据结构来保存公共部分的运算结果数值，这样在模块内部后续使用相同表达式的地方则可以直接调用该值。所以基于以上原理，在进行冗余逻辑优化和合并的基础上，可以增加最小面积优化方法，在资源估算时，以LUT4为base进行初步资源估算，在当前基础上，以LUT4之间的优化方法，我们可以以高维度LUT为Base的FPGA作为方向，将多个LUT在合理的范围内进行合并，合理的范围即在一个Module内的组合逻辑或连续赋值、阻塞赋值、非阻塞赋值等LUT合并至LUT6，从而实现以最小面积为目标的优化策略。资源估算模型将找到具有共同输入的共同元素类型并假设共享资源。例如，如果找到两个加法器，均为A+B，则假定加法器逻辑是共享的。对合并的元素数量设置限制，以限制扇出。同时模型将查找将使用LUT实现的二进制逻辑函数序列，并根据树的唯一输入数和输出数对其进行优化。结果分析快速资源估算模型无优化结果对比快速资源估算模型增加优化结果对比表3.1、表3.2分别为快速资源估算模型在5个基本Case下，未增加优化项与优化算法的估算结果与增加优化项及优化算法后的准确值与估算时间的对比，通过分析我们可以发现在不增加优化项及优化算法的情况下在使用资源量即电路逻辑复杂度较低的情况下准确度较好，但随着设计规模的增加电路复杂度的上升其预测资源准确度越来越差，由于对电路的过度精细化导致在面对设计规模较大的电路时由于内存原因导致无法得到估算结果，但是在时间性能上已经较综合布局得到较高的提升最小的设计时间提升约15倍，随着设计负责度的增加，在时间性能提升越明显，同时对比优化后结果可以得出首先由于设计规模较大无法计算的资源已经可以得到结果，且各个Case的估算结果准确度较好，LUT资源估算平均准确度达90.11%，FF资源估算平均准确度达88%，并且在时间性能上进一步提升，较未优化模型提升2倍，已满足在高密度FPGA原型验证系统划分阶段中对超大型设计进行快速资源估算并借以确定划分边界的实际需求。本章小结本章详细叙述了，在以第二章理论为基础的前提下，使用Verific基础数据结构基于C++语言构建面向高密度FPGA原型验证系统RTLPartition的FPGA设计的快速资源估算模型，同时提供多种优化及约束方案，对模型进行约束及二次优化，使得对设计在FPGA上的资源估算精确度达90%以上，并得到了较好的时间性能，在超过百亿门的电路估算时间仅需几十分钟，时间较完整的综合流程缩短200%以上，同时快速资源估算模型为支持高密度FPGA原型验证系统，为用户提供多种额外功能，如用户指定资源输入、对应不同FPGA的使用数量评估等作为商业软件对用户的友好性功能。同时也为后续基于人工智能的FPGA资源估算模型提供思路及开发基础。基于机器学习的RTL级资源估算对于机器学习项目，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面性、准确性和样本。所以特征的选择和提取尤为重要。本章主要阐述基于第二章中提到特征提取方法，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。构建基于C++的自动化特征提取器并对样本数据构建数据集，并对数据集样本进行分析，洞见特征数据间的数学联系及统计学关系，并尝试多种机器学习回归模型，选择表现较好模型进行进一步训练。VerilogHDL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VerilogHDL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VerilogHDL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IEEE1801所给出的Verilog的技术文档，对VerilogHDL进行全面特征提取。通过对VerilogHDL基本语法分析结合IEEEVerilog标准、RTL样例设计，拟提取VerlogHDL语法特征对应关键字如表4.1所示。Verilog主要语法类型与关键字语法类型语法关键字端口InputOutputInout线网WireReg逻辑运算符、算术运算符、关系运算符+-\*/^|&连续赋值语句AssignAlways过程块Always过程赋值语句=<=条件表达式Ifelsethen条件分支语句case循环语句ForWhileRepeat函数语句FunctionRTL级设计特征提取方法设计首先我们通过第二章的综合及技术映射原理已知VerilogHDL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所述特征。（1）端口信息特征提取首先对于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数据结构中存储在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RTL\_InputWidth、RTL\_OutputWidth、RTL\_InoutWidth特征，实现对Module端口特征信息的提取。（2）线网信息特征提取VerilogVHDL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VerilogHDL语法中主要体现为Wire和Reg关键字，在Verific的Parse-Tree数据结构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关键字的数量以及对应的信号位宽，分别存储为特征RTL\_Reg、RTL\_RegWidth、RTL\_Wire、RTL\_WireWidth特征，从而实现对Module线网信息的特征提取。（3）操作符及运算表达式特征提取在VerilogHDL语法中，表达式的本质就是通过将操作数值或操作变量、通过运算符或操作符进行联结运算的表达式，所以在VerilogHDL语言结构中可理解为一致，即通过运算符运算得到一个结果，从语义上说，这个结果即为运算后得到的的函数值；而操作运算符的类型与C语言类似，如图4.2所示。Verilog操作符图示首先已知逻辑运算符、算术运算符和关系运算符等操作符在进行逻辑综合后会被综合器转换为对应的逻辑运算门，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Hash表中，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Hash表中，递归完毕后既可以完成对运算符表达式的特征提取，算法伪代码如图4.3所示。运算符特征提取伪代码（4）连续赋值语句特征提取连续赋值语句是在VerilogHDL语法中比较重要且常用的语句，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，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语句则存储至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4.4所示。连续赋值语句特征提取伪代码（5）过程结构体特征提取VerilogHDL语法中的过程机构体，都是由关键字initial或always和其块内内容语句所组成的，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。通过对Always语法分析，always语句块中的内容块主要由赋值语句和高级程序语句（包括条件分支语句和循环控制语句）这两种行为语句构成[16]，其语法格式如下，如图4.6所示。VerilogAlways过程块语法格式由于Always过程块的触发是根据敏感事件列表中的变量是否可以被触发来控制，敏感事件列表通常可有1个或多个表达式来组成。所以对于Always块的特征信息的提取，主要分为三部分：获取Always语法块数量；获取Always语法块敏感事件列表、并解析宽度；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VerificParse-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，由于敏感事件列表中多由表达式构成，所以通过使用前述0、0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。Always特征提取伪代码|（6）过程赋值语句特征提取Verilog过程赋值语句语法格式过程赋值语句的语法格式如图4.7所示，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0、0、0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，可参见图4.4。由于过程赋值表达式主要被用于过程块中，所以0中对语法块内容的提取将在该部分体现。（7）If-else条件分支语句特征提取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；If-else语句为Verilog语法中高级语句，多用于AlwaysBlcok中，图4.5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VerilogHDL语法可知，if-else条件分支语句的作用是复选，即通过是否满足给定的判断条件来判断下一步电路需要执行的操作或逻辑。其主要语法形式如图4.8所示。VerilogIf条件语句语法格式由于if-else条件分支语句常用语Always过程块中，所以我们在0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0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0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。从而完成对if-else条件分支语句的特征提取。If条件语句特征提取伪代码（8）Case条件分支语句特征提取基于0中对VerilogHLD分支控制语句的分析可知，case分支控制语句与if-else语句在功能上大同小异，均是实现分支控制选择，但Case语句可以用较少的语法表达提供较多的分支选择，在实际开发中多用于表述状态机的行为级描述，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4.10所示。VerilogCase语句语法格式在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0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为赋值语句，所以可以使用0中模块进行对应特征提取。Case条件分支语句特征提取伪代码如图4.11所示。Case条件分支语句特征提取伪代码（9）循环语句特征提取与条件分支语句表达含义一致，循环控制语句也是高级VerilogHDL程序语句的一种，Verilog提供的循环控制语句关键字有forever、repeat、while、for四种，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图4.12所示。Verilog循环语句语法格式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、从而计算出整个循环语句块的循环总次数，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RTL\_ForItem，同时将for结构体数量存储于特征RTL\_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0、0中所提及的算法进行特征抽取，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，通过以上方式完成对Verilog循环语句的特征提取。循环语句特征提取伪代码（10）函数语句特征提取VerilogHDL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4.14所示。VerilogHDLFunction语法结构function函数定义由于只能被定义在Module中，不被允许在Always过程块内调用且函数模块必须具备相应的IO端口用以提供外部互联。在VerilogHDL语句中，函数代表了纯组合逻辑。在函数语句块中定义的全部的变量在综合时都会被定为临时变量，仅被综合映射为成片上导线[16]。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0、0、0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。自动化特征提取器的实现本节将叙述基于4.2节所述的特征提取方法构造对应的模块，使用C++设计并开发特征提取器程序，特征提取器流程框图如图4.15所示，输入为包含待所有待提取特征设计文件绝对或相对路径的Filelist，输出为包含特征的Fileli。使用C++14语言进行开发测试，从而完成自动多Case特征提取器的实现。RTL语法特征提取器流程图该自动化特征提取器，基本涵盖VerilogHDL的所有关键语法及在逻辑综合和技术映射中常用的重要语法信息。包括逻辑运算如（or、shift）的位宽信息、算术运算符的位宽信息、连续赋值运算的位宽信息、阻塞及非阻塞赋值运算的位宽信息、高级运算语法的解析如循环体、函数体、判断决策语句等。数据集分析数据集构建由于机器学习本质上是使用大量的样本数据，通过计算机实现的数学模型，来探索并计算输入特征之间的数学关系，从而实现预测，所以需要使用大量的样本数据来构建数据集，所以我们使用300个VerilogHDL设计及其在XilinxVivado综合工具上综合后输出的资源利用率结果来作为基础数据集及预测标签。VerilogHDL数据集来自GitHub、Opencore等开源网站，基本数据集涉及CPU、加法器、乘法器、GPU、机器学习等多种类型。部分数据集展示如表4.2所示，数据集主要描述描述数据如表4.3所示。部分基本数据集数据图示数据集基本概要数据集分析数据可视化我们对数据集使用第四章构造的特征提取器进行特征提取后对数据进行可视化，将可以便于我们对数据进行进一步的分析，使用直方图来展示数据，如图4.16所示。特征数据集直方图相关性分析数据是对事实进行观察后产生的结果的描述，是对客观事物一种数学逻辑归纳，是用于表示客观事物的最好的原始素材[3]。数据相关性是指数据之间存在某种关系，如正相关，负相关。所以我们需要通过对提取的特征在样本数据下进行相关性分析，来进一步分析特征的对预测值的重要程度，其指标为标准相关系数也称为皮尔逊系数r。使用错误!未找到引用源。所构建的特征提取器所提取的特征进行整体特征分析可以得到如图4.17所示的特征热力图与表4.4、表4.5、表4.6、表4.7所示的各个资源与RTL级特征间的皮尔逊相关系数。特征热力图RTL级特征与LUT资源的相关性数据RTL级特征与FF资源的相关性数据RTL级特征与BUFG资源的相关性数据RTL级特征与IO资源的相关性数据通过分析我们可以得到FPGA片上资源LUT对应的RTL级代码使用4.2节所述的特征提取器所提取的特征之间的相关性关系，我们可以发现较多特征与LUT、FF、BUFG、IO资源具有典型相关性，进一步验证了通过4.2节构造的特征提取器提取的特征基本符合第二章中逻辑综合原理与技术映射原理，证明该特征提取存在合理性，同时相关性分析也为我们下一步特征工程特征降维删除不相关体征提供参考及依据。特征工程本研究共计提取了67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，而位宽大小为位且数量级不同等，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。则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。特征预处理数据清理机器学习算法无法在存在缺失值的特征上来进行学习，但每次进行手动数据填补或清洗将带来巨大的人力损耗，所以我们可以基于流水线的方式，通过构造些自动化方法其完成对数据的清洗处理，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：放弃这些数据缺失的区域；放弃整个属性；将确实的特征值或实际值以书记进行填充，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。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：使用OrdinalEncoder顺序编码对文本属性进行编码；使用OneHotEncoder独热编码对分类属性进行编码；特征缩放由于样本存在较大的差异性，且不同的模型对于数据差异的敏感度不同，所以需要对特征数据进行的一项重要操作就是特征缩放。若输入的样本的特征数据存在较大的数值差异，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：最小-最大缩放（归一化）归一化的操作较为简单，即将样本数值重新处理后使其使其最终范围归于区间[0,1]之间，具体的实现方法是将该样本值减去样本数据集中的最小值比上样本最大值与样本最小值的差。标准化方法与归一化方法相比，标准化则完全不同，其本质将特征数据值先减去平均值，再与方差相除，进而使得特征数据具备满足单位方差的分布。标准化方法与最小-最大缩放的典型区别与其优势便是标准化处理不会将输入的特征数值绑定到特定范围内，但是标准化方法对于异常值的敏感程度或受影响程度将会减小。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4.18所示。特征预处理流水线当数据集经过特征预处理流水线处理后，完成基本特征工程便具备了进入模型进行训练的基础。经特征工程处理后最终输入特征张量为61\*218，标签为使用Xilinx公司Vivado软件进行综合及布局布线后获得的准确资源利用率数值。模型框架选择模型训练开发流程机器学习模型训练开发流程如图4.19所示，在完成数据分析，特征分析，特征工程后，我们需要先数据集进行划分选择合适的机器学习模型来进行预测，首先我们不知道选择哪种模型的时候，先对各种回归模型进行初步尝试，指标采用R2和RMSE进行评估，然后选择在初始参数及该特征下表现较好的模型框架进一步对模型参数进行配置。我们分别使用线性回归、决策树、SVM向量机、随机森林、深度神经网络来对数据集进行初步的训练，从而选择出两种有前途的特种进行进一步的训练和微调从而实现我们的预测目标。首先我们对数据集进行划分，我们随机抽样20%的数据来构成验证集，数据集中剩余80%的样本作为训练集，为保证训练及验证的全面覆盖，我们在验证时使用5折交叉验证来对我们的模型进行预测验证[29]。以下结果均采用标准参数对LUT资源进行尝试预测，预测结果如图4.20所示。各模型尝试预测LUT资源结果通过上图所示对多种机器学习模型进行原始参数初步尝试后，我们可以发现在该特征提取及样本集上线性回归、决策树的表现的初始模型预测表现较差，但是随机森林、MLP这两种模型在标准参数下存在较好的的预测性能，所以本文下面将着重以随机森林及深度神经网络来进行进一步的预测模型训练及参数调整。机器学习模型训练Keras是具有高度模块化且提供易用API的库的神经网络库。基于Python语言，能以TensorFlow或Theano等为后端运行。Keras能够非常快速的搭建网络模型，以便迅速的将使用者的想法进行实现。基于上述目的Keras具有以下特征，首先，其可以在CPU与GPU上无缝的切换运行；第二，拥有非常简洁且友好的调用接口，减小了研究人员的工作量；第三，模块化设计，使用者可以根据需求非常方便的配置自己的模型；第四，支持支持任意的网络架构。基于上述的优势，本文选择Keras作为机器学习的基本框架。机器学习模型训练及调优流程随机森林本节在错误!未找到引用源。所尝试的随机森林模型的基础上对随机森林参数进行调优，并通过学习率曲线结合随机搜索与网格搜索对参数进行进一步优化，我们将主要调整影响度较大的n\_estimators与max\_depth参数用于寻找最优模型。RF模型进行训练、调优的流程如图4.21所示。LUT资源模型训练随机森林预测LUT模型参数搜索学习曲线随机森林LUT预测模型学习曲线依据图4.21所示对随机森林预测LUT模型进行训练，首先通过多次迭代依据借由随机森林模型提供的特征贡献度对特征进行二次调优，通过放弃贡献度较低的模型来提升模型准确度，最终输入张量为59\*220。在完成特征二次调整的基础上对模型进行超参调整，主要参数n\_estimators与max\_depth的的学习曲线如图4.22所示，验证方法使用10折交叉验证，得到在n\_estimators取33与max\_depth取12时评价指标RMSE最优。FF资源模型训练随机森林预测FF特征搜索学习曲线随机森林FF预测模型学习曲线依据图4.21所示对随机森林预测FF模型进行训练，训练流程与4.7.2.2方法一致，通过迭代训练与参数调优，主要参数n\_estimators与max\_depth的的学习曲线如图4.24所示，验证方法使用10折交叉验证，得到在n\_estimators取89与max\_depth取7时评价指标RMSE最优。人工神经网络预测LUT资源模型训练神经网络学习曲线，左图为超参数调优前学习曲线依据图4.21所示流程图构建神经网络模型，为探究初始数据集在模型上的表现特征，通过对未进行超参数调整的模型进行1000训练可得如图4.26所示学习曲线，通过分析可发现初步训练模型存在过拟合情况，为解决过拟合情况，通过使用随机森林迭代构建的特征，并调整模型深度与宽度来解决该问题，分别使用随机搜索与网格搜索对模型隐藏层层数、神经元数、学习率及采样数进行调整，均采用5折交叉验证及提前终止回调用于终止搜索裁判，通过参数搜索可得神经网络模型如图4.27所示，神经网络包括输入及输出层、隐藏层共计6层，输入层宽度为59，由于为典型回归模型故输出层宽度为1，隐藏层共4层，对应神经元数分别为59、96、110、58，学习率使用搜索最佳学习率为0.000107，对模型进行200轮次训练。神经网络训练学习曲线如图4.28所示，通过观察学习曲线可以发现通过特征迭代与超参数调优已经解决过拟合问题，且拟合情况较好。神经网络模型结构图神经网络与LUT模型学习曲线预测FF资源模型训练依据图4.21所示流程图构建神经网络模型，使用神经网络模型预测FF资源的训练方法与4.7.2.1基本一致，分别通过随机搜索与网格搜索对神经网络模型进行空间搜索，得到如图4.29所示的神经网络结构，隐藏层共3层，对应神经元数分别为53、162、27，学习率使用搜索最佳学习率为0.00013，对模型进行235轮次训练。神经网络学习曲线如图4.30所示，我们可以发现模型训练曲线拟合度较好并未出现明显的过或欠拟合情况。神经网络模型结构图神经网络模型预测FF资源学习曲线结果与分析机器学习模型验证结果模型评估图4.31为机器学习模型在验证集上预测值与基准值的验证结果，X轴为验证集的准确值，Y轴为预测值。图（a）、图（b）分别为使用随机森林模型对LUT资源、FF资源进行预测的验证结果，图（c）、图（d）为人工神经网络模型对LUT资源、FF资源进行预测的验证结果。各模型验证集评估指标如表4.8所示，结合R方评估指标与图4.31我们可以发现机器学习模型在4.3节中构造的特征提取器的基础上得到了较好的拟合结果，验证了对VerilogHDL语言特取的特征对LUT与FF资源的预测有较好的数学描述。但同时也能发现在LUT、FF资源的预测上人工神经网络这种对非线性特征的学习能力较高的模型表现结果优于随机森林，对LUT预测和FF分别的模型确定性系数达73.4%、85.06%，证明模型已经可以较好的描述特征所代表的数学意义，同时均方根误差对LUT、FF资源分别为和。我们在测试数据集的基础上选取了模型评价指标较高的模型的结果进行分析，结果表明，在使用随机森林对RTL级设计RTL特征进行资源估算后与基准值进行对比，达到了31.3391、13.2000误差的精度，得到了一个较为满意的结果，同时由于该模型预测结果面向为高密度原型验证系统，主要应用于RTL-Partition阶段，基于其对FPGA片上资源利用率存在阈值，即常规情况下划分结果平均占用FPGA资源的50%~70%，且在划分系统程序中可以通过设定资源阈值控制来调整误差，所以当前模型输出的结果基本符合高密度FPGA原型验证系统的需求，同时除去精度外其更加聚焦于时间性能上的提升，通过统计RF、MLP的模型预测时间，我们可以明显看到机器学习在时间性能方面的巨大优势。本章小结本章叙述了本文中面向机器学习进行资源估算中最为重要的一部分，本章通过结合第二章所述的特征提取方法，通过对VerilogHDL语言分析，结合其语法在逻辑综合、技术映射上的特征关系，完成特征提取器的设计，通过C++完成开发，实现对RTL级代码的特征提取，为下一步建立机器学习模型的建立打好基础。并提出了使用机器学习来进行RTL设计在RTL级对其所需FPGA片上资源估算的方法，在对原始数据集进行特征缩放、特征标准化、特征降维等特征工程处理后分别使用RF、MLP进行回归预测，叙述了对多种机器学习模型在标准参数下对使用文本方法所构建的数据集的预测性能表现及对随机森林、深度神经网络进行进一步预测、参数微调等方法及结果的叙述，最终我们得到了对RTL级设计FPGA片上资源利用率预测符合预期且合理的模型。总结与展望工作总结本文的主要研究方向集中在高密度FPGA原型验证系统软件设计CAD流程中的划分阶段的资源估算领域，研究主要面向的是超大型的SoC设计的RTL级描述，研究主要分为三个方面：首先介绍了原型验证流程中FPGA的设计综合流程，包括逻辑综合原理，逻辑综合中VerilogHDL语法对应的逻辑综合原则，技术映射原理，现阶段工业界常用技术映射算法的描述，其次基于该原理结合高密度FPGA原型验证系统划分阶段需求设计并实现了快速资源估算模型，最后提出在EDA软件中结合机器学习方法，将资源估算与机器学习相结合，最终实现以下内容：1、实现以传统思路框架为基础的快速资源估算模型，用户输入RTL级设计即可以快速得到设计所需要得FPGA片上资源数量如LUT、FF、BUFG等资源，有效提升了在超大型集成电路设计原型验证RTL划分阶段的迭代速度。2、并在此基础上加入了合理的LUT优化算法，Bottom-To-Up算法，使该快速资源估算模型更加匹配实际划分需求及面向不同设计种类的优化方法及面向不同需求的优化方法，并增加可复用资源结构定义，并支持用户自定义功能user-define，使得原型验证划分阶段的迭代速度和实用性得到显著提升，有效提升约2.2倍，由于自主开发故拥有完善的C++输入输出接口，可集成在商业软件中，且实际应用至Player-ProRTLPartition工业商用软件中，并在实际客户应用场景中得到检验。3、以EDA与机器学习相结合的思路，结合逻辑综合及技术映射原理，构建面向RTL级集成电路行为描述的特征提取器，以XilinxEDA设计软件为标签，尝试训练了线性回归、决策树、SVM、随机森林、人工神经网络等多种机器学习模型，使运行速率提升15倍，平均准确度达75%，同时验证了EDA软件与人工智能相结合的可行性，进一步提升了在超大规模集成电路设计FPGA原型验证的划分阶段的迭代速度，为EDA与原型验证流程相结合的研究打下基础。通过以上三点，本文论述的工作为我国自主研发EDA与高密度FPGA原型验证系统的RTLPartition提供了可使用的工程软件，并在使用机器学习对从RTL级设计出发进行FPGA资源估算领域填补了空白。同时也为EDA软件与人工智能的结合验证了可行性。 工作展望面向RTL级设计的FPGA快速资源估算问题结合了图论、可行性问题、逻辑分解、逻辑优化等众多方面特性，具有一定挑战性，随着设计规模的逐渐增大，在验证方面的需求特性也在增加，如何能有效的快速且准确的得到RTL设计所需的FPGA片上资源，并能有效合理的支持原型验证系统的RTL-Partition划分是未来EDA流程中将面临的一大问题。本文虽在这方面做了一些研究和实现，但仍需在一下方面做出进一步的改进：1.目前虽然实现快速资源估算模型，并增加了符合实际映射原理的优化及约束选项，但是在特殊的FPGA板上器件映射计算方面依然存在较大误差，如BRAM、LUTRAM等先进器件，所以可以考虑后续在这方面增加进一步的研究，使得资源估算更加精确。2.虽然在本文中验证了机器学习方法与资源估算相结合的可行性，但是还处于初步阶段，精确度及估算器件类型上还有待增加，如可参照文章【】中提到的多级估算方法，将初步得到的一般器件估算结果进行二级抽象，得到更高级别的资源类型。3.在面向资源估算的机器学习模型中增加更高层级抽象特征，增加实际模型泛化率，可参照文章【】利用HLS高层次综合方法快速构建样本设计从而增加模型训练样本数量，使得机器学习模型的泛化程度更高更有效。4.神经网络资源估算模型还可以进一步增加不同设计在不同约束下进行综合的样本，使得机器学习模型可以支撑不同约束下的资源估算结果，如面积优化、功率优化等约束选项。