**基于深度学习的FPGA快速布局算法**

刘 伟，王伶俐，周 灏

（复旦大学 专用集成与系统电路国家重点实验室，上海201203）

摘 要：本文提出一种新型的基于深度学习的FPGA快速布局算法，将FPGA布局转化为动态的进行逻辑单元块的选择和逻辑单元块位置确定的过程，从而实现电路网表在FPGA上的逐步布局。其中每一个逻辑单元块的位置确定由训练好的深度学习网络预测实现，所有逻辑单元块位置确定之后采用基于交换的快速详细布局算法进行优化。实验中使用MCNC基准电路进行测试，将测试结果与VPR中基于模拟退火的布局算法进行对比，结果表明在关键路径延时平均9.8%布线后的损失代价下，整个布局过程的运行速度平均提升了24.54倍，其中处理十万量级大规模电路实现64.9倍的速度提升。

关键字：现场可编程门阵列；深度学习；布局算法

中图分类号：TN791 文献标志码：A

现今，现场可编程门阵列（Field Programmable Gate Array, FPGA）作为一个可重构平台获得越来越广泛的应用。为满足实际的需要，Xilinx和Intel公司最新的芯片包含近百万个逻辑单元块。随着电路规模和复杂度的提升，在FPGA的整个计算机辅助设计流程中，布局占据近49%[1]的运行时间，这直接影响到FPGA的使用效率。

FPGA布局是NP（Non-Deterministic Polynomial）完全的组合优化问题[2]，对于一块给定的FPGA芯片，对打包流程之后的电路网表中的每一个逻辑单元块进行位置选择，以实现最终关键路径延时最小和线长的最优。总体来说，布局算法有三类：第一类为启发式算法，典型的代表就是VPR[3]中使用的模拟退火算法，它基于交换的方式实现对线长和关键路径的优化。虽然这一类方法能够得到比较理想的布局结果，但是处理大规模电路布局时，需要非常长的运行时间；第二类是基于划分的布局算法，如PPFF[4]，它将一个大的电路布局问题分解为更多的子问题，从而降低了电路计算的复杂度。但是该算法在性能上会带来非常大的关键路径延时损失；第三类是解析法，如QPF[5]，CAPRI[6]，StarPlace[7]，这也是目前Xilinx商业计算机辅助设计软件中使用的布局算法。解析法是目前综合表现最优的算法，在接近于VPR布局结果的情况下通常能够实现布局速度3～8倍的提升。

针对基于模拟退火的VPR布局算法的优化主要分为三类：第一类是提供更精确的目标函数，如采用基于路径的时序分析模型或者引入拥挤度矩阵为每一次交换提供更高质量的结果。但是在大规模电路指数级增长的交换次数前提下，VPR布局阶段使用的半周长线网框模型和最小延时时序矩阵是相对理想的选择；第二类则是寻找更高效的策略如初始化布局，自适应退火表，交换逻辑单元的选择以及交换的目标区域选择等。模拟退火算法交换过程中随着温度降低逐渐将交换范围逐渐缩小，RBSA[8]采用线网中心而不是逻辑单元块的中心指定交换范围产生交换策略，在不损失关键路径延时的情况下运行速度提升一倍；第三类是采用并行策略对模拟退火算法进行加速，文献[9]中在小规模电路中实现64线程下运行加速提升34倍，线长损失平均为1.7%，关键路径延时损失为0.6%，但是该并行策略并没有在大规模电路有效的实现。所以目前对于基于模拟退火的FPGA布局算法，在处理大规模电路时难以得到一个快速而且比较理想的布局结果。

收稿日期：2019-01-05

作者简介：刘伟（1993—），硕士研究生；王伶俐，男，教授，通信联系人，Email：llwang@fudan.edu.cn

深度学习[10]与传统的机器学习方法不同，在模型深度上扩展，分层地进行学习并将上层学习到的特征作为下层的输入，实现由低阶到高阶的特征提取。深度学习已经广泛应用于视觉，文本识别，机器人，人工驾驶，游戏，医疗等诸多领域。文献[11]使用机器学习算法对FPGA布局结果进行布线拥挤度的预测，相较于传统的VPR中使用的拥挤度预测方法，在近乎准确的拥挤度预测情况下实现了291倍的运行速度提升。本文基于深度学习处理FPGA布局这样的NP完全组合优化问题，实现了布局速度的极大提升。

1 模型思想

本文模型有别于基于传统的三类布局算法，由底层到顶层的方式逐步进行布局。在每一次逻辑单元块

的选择和逻辑单元块位置确定的过程中，需要动态的提取当前布局阶段的状态信息，为下一次决策做准备。采用训练好的深度学习网络，将对应的状态信息作为网络输入，实现对下一个逻辑单元块的位置确定，最后使用基于交换的快速详细布局算法进行优化。状态信息的提取和网络模型的训练在模型实现中详细介绍。

2 模型实现

**2.1** FPGA**结构**

深度学习的模型训练针对VPR中一种通用的FPGA芯片描述k4\_n4\_v7\_bidir.xml，芯片的核心部分由可配置逻辑块阵列（Configurable Logic Block, CLB）和全局布线资源组成。如图1所示，每个可配置逻辑单元由四个基础逻辑单元（Basic Logic Element, BLE）组成，每个BLE包含一个四输入查找表（Look-Up Table, LUT)和可配置D触发器。FPGA全局布线资源中线长为4，CLB通过连接盒与全局布线资源连接。

**2.2电路网表解析**

电路网表的解析用于逻辑单元块的选择和逻辑单元块的位置确定。打包之后的电路网表中的结点不仅包括可配置逻辑块等逻辑单元块的输入输出端口的信号结点（外部结点），也包含可配置逻辑块内部如LUT4和D触发器输出端口的信号结点（内部结点）。初始化时，不考虑结点之间的延时，对电路网表中的所有结点进行拓扑排序，对所有结点进行遍历之后，可以确定决定每一个结点最长路径的输入结点。在对网表进行解析的过程中保留确定所有结点最长路径的路径，超图的连接由此解析为所有结点组成的森林，每个结点都在森林中的一棵树中。布局过程中，采用VPR中的布局阶段延时模型对已布的电路的结点延时和路径延时进行分析，树中父子结点的关系由延时模型计算得到的最大延时确定，所以在周期性的更新过程中，动态的布局过程也伴随电路网表解析森林中父子结点连接关系的变化。 图1可配置逻辑块结构

Fig.1 Architecture of CLB

**图2所示即是经过拓扑排序之后的一个电路网表。图中实线和虚线都是网表中不同结点的连接，实线为确定该线输出端点最长路径的路径，从图中可以看出拓扑排序之后的网表形成了由三棵树组成的森林，每棵树由椭圆内的结点和实线连接组成。在算法的动态的布局过程中，动态的网表解析过程如表1所示，首先在布局开始之前进行拓扑排序，排序之后保留确定每个网表中结点最长路径的路径从而将电路解析为森林。随后布局的实现过程中，在每次布局单个逻辑单元块之后对当前电路中已布的部分网表进行时序更新，如图3描述的是布局的中间过程，此时已经处理到网表中的A，B，C，D，E，G，J结点，与之前分析不同的是每条路径对应有相应延时计算值，此时结点J的最长路径延时由CJ之间的路径确定。受运行时间的影响，时序更新之后，网表的动态解析条件由第十行确定，在本文具体实现中*Every\_Iteration\_times*取值0.01，*Threshold\_time\_update*取值1.10。**

 

**图2 初始化网表解析 图3 动态网表解析**

**Fig.2 Initialization of netlist parser Fig.3 Dynamic netlist parser**

**表1 网表解析过程伪代码**

**Tab.1 Pseudocode of netlist parser**

|  |
| --- |
| **输入：电路网表*Netlist，P*为网表逻辑单元块数量**  **输出：电路网表解析数据结构**  **对网表*Netlist*中结点进行拓扑排序** |
| **保留确定每个网表中结点最长路径的路径** |
| **//布局过程中动态更新网表解析过程，*Last\_Update*用于记录上次动态更新网表解析到当前阶段布置逻辑单元块的数目**  **//*Last\_Critical\_Time*用于记录上次动态更新网表解析时最长路径的延时, *Critical\_Time*为当前阶段最长路径延时** |
| **For j 🡨 P to 0 do**  **调用训练好的深度模型进行布局**  **更新已布部分网表中结点和路径延时，结点中最大延时值为*Critical\_Time***  **if (*Last\_Update* > *Every\_Iteration\_times*×*P*) or(*Critical\_Time / Last\_Critical\_Time*) > *Threshold\_time\_update***  **更新确定每个网表中结点最长路径的路径**  ***Last\_Update* = 0，*Last\_Critical\_Time = Critical\_Time***  **else**  ***Last\_Update* = *Last\_Update* +1**  **End if**  **End For** |

**2.3 深度学习模型**

训练模型的建立主要包括数据的提取，深度学习网络结构的确定，特征的选取三部分。

2.3.1数据的提取

数据由两类电路网表产生，一部分数据由基于PEKO[13]的最优线长电路生成器产生，该电路生成器中每个逻辑单元块的位置都是线长最优的，另一部分数据是由随机化电路生成器产生。由于VPR采用的模拟退火算法通常能够得到较好的布局结果，所以对于随机化电路网表将模拟退火算法布局得到的逻辑单元块的位置作为深度学习模型训练的理想标签。

最优线长电路是普通电路的一种特例，也是一种理想情况。根据VPR中线网框计算方法，对于有N个端点的线网，将其所连接的N个逻辑单元块限定在包含该逻辑单元块的最小的矩阵框范围内能够实现周长最小的线网框，从而得到最小线长。对于所有的线网都找到最小的线网框从而得到最优线长的布局结果[13]。虽然与通常的基准电路不同，最优线长电路网表中逻辑单元块之间简单的连接，以及最优位置的确定更有利于深度学习模型的训练。PEKO算法确定的是逻辑单元块之间外部线网，以达到线长最优。最优线长电路网表生成器在PEKO基础上产生适用于的FPGA芯片k4\_n4\_v7\_bidir.xml的电路网表。表2为最优线长电路网表生成器的伪代码，*P*为网表逻辑单元数量，*Arch*clb为逻辑单元块结构。如表中所示，为产生不同的电路网表，一方面需要根据结构产生不同的电路网表参数如*In*ave,*Out*ave,*Dff*ave，*Distri*以及对应于每个逻辑单元块的*Lut*num和*Dff*num，另一方面需要在合法连接的前提下实现全局互连以及逻辑单元内部互连的随机性。随机化电路网表生成器不同之处是在选择每一条线网所连接的逻辑单元块时，不是根据半周长线网框模型选择包含该线网所连接逻辑单元块的最小矩阵，而是随机性的选择任意大小和任意位置的线网框，并最终在该线网框选择确定数量的任意逻辑单元块。

表2 最优线长电路网表生成器伪代码

Tab.2 Pseudocode of optimal wirelength circuit netlist generation

|  |
| --- |
| 输入：*P*为网表逻辑单元块数量，*Arch*clb 为逻辑单元块结构。  输出：电路网表*Netlist*以及最优布局结果*R*  //确定线网分布向量，表示*n*端线网的数量  //为*Arch*clb输入端口数量，为电路网表中平均每个逻辑单元块输入端线网数量  //为*Archclb*输出端口数量，为电路网表中平均每个逻辑单元块输出端线网数量  确定 ，  随机产生使得  //确定FPGA芯片的规模[*X,Y*]  确定***X*,*Y***  //将*P*逻辑单元块布局到FPGA阵列中，并且确定外部结点线网连接以及布局结果  For i 🡨 n to 2 do  While > 0 do  随机选择目前具有最少连接的逻辑单元块*m*  随机选择一个大小为或者的包含*m*的线网框*b*  在线网框*b*中产生*m*与其他逻辑单元块的线网连接  将外部结点线网连接存入*Netlist*  End While  End For  存入最优布局结果*R*  //产生逻辑单元内部结点连接，保证时序电路合法并存入  //为*Arch*clb中D触发器总数量，为电路网表中平均每个逻辑单元块使用的D触发器数量  确定逻辑单元块内部  For j 🡨 P to 0 do  初始化用来存储和更新结点之间的合法化信息  While success == 0 //产生逻辑单元块内部连接不成功  确定该逻辑单元块中LUT4使用数量 ，根据 确定，为使用的D触发器数量  For k 🡨 to 0 do  随机选择LUT4的输入信号，判断是否需要使触发器保证电路合法性  End For  确定数量的触发器的分配  For k 🡨 to 0 do  产生内部结点连接并保存入Netlist  End For  更新  End While  End For |

2.3.2 深度学习的网络结构

FPGA芯片阵列映射到深度学习网络后的阵列称为FPGA资源阵列，本文使用的资源阵列中的行列数为[100,100]。映射之后的FPGA资源阵列中每个坐标位置可能包含零个，一个或者多个逻辑单元块，包含非零个逻辑单元块的坐标位置的集合称为FPGA有效资源阵列。**由于FPGA芯片阵列的规模[*n,m*]可能变化，对应的映射规则如下：对于FPGA芯片的行列数[*n*,*m*]（假设*n*大于或者等于*m*），当*n*小于100时，可以在FPGA资源阵列中任意某个连续的[*n*,*m*]区间的每一个坐标位置中，一一对应的从FPGA芯片阵列进行映射到FPGA有效资源阵列，而[100,100]规模的FPGA资源阵列中除该[*n*,*m*]外为每个坐标位置包含零个逻辑单元块。而*n*大于100时，要映射到FPGA资源阵列，则需要有效资源阵列中坐标位置包含多个逻辑单元块。本文设定映射后的FPGA有效资源阵列除边界可能不同外，每个有效的坐标位置包含相同数目的逻辑单元块，其中FPGA有效资源阵列中的每个坐标位置同样对应于FPGA芯片阵列范围中连续的[]区间，有效的阵列范围为[]。**

深度学习采用的为残差网络[14]，实验中证明，在深度网络的训练中，残差块能够加速收敛并提高训练效果。整个网络结构由输入层，20个中间层（残差块），输出层组成。中间层（残差块）及参数如图4所示，残差块的第一层的输入被加入到最后一层的输出中，其内部包含由批归一化和激活函数隔离的卷积层。输入层同样使用类似残差块结构，如图5所示，卷积核的大小分别为[5,5]与[1,1]，卷积层之后的输出进行相加。输出层卷积层中卷积核大小为[3,3]，输出层的大小为[100,100，1]，输出层经过softmax层后得到的是逻辑单元块位于资源阵列中各个位置的概率分布。

 

图4 残差块 图5 残差输入层

Fig.4 Residual block Fig.5 Resudual input layer

2.3.3 特征提取

布局过程中已经被选择并且确定位置的逻辑单元块称为已布逻辑单元块，被选出来并且即将由训练好的深度学习模型进行位置预测的逻辑单元块称为待布逻辑单元块，目前未被选择的逻辑单元块称为未布逻辑单元块。特征提取的思想是在动态布局过程中，统计FPGA资源阵列中每个坐标位置可供使用的逻辑单元块资源（总的逻辑单元块资源与已布逻辑单元块资源之差）和每个坐标位置附近需求的逻辑单元块资源，并根据提取的时序和线长信息对待布逻辑单元块进行位置的确定。

特征一：FPGA资源阵列。该特征记录FPGA芯片的总逻辑资源，包含FPGA芯片映射的FPGA资源阵列以及每个坐标位置所包含的逻辑单元块数量。

特征二：已布逻辑单元块占用资源。该特征动态记录FPGA资源阵列中每个坐标位置上已布的逻辑单元块数目。最大值为特征一中对应坐标位置的值。特征二与特征一的差值也即是每个坐标位置可供使用的逻辑单元块资源数目。

特征三：线网连接数量。线网连接记录坐标位置已布逻辑单元块与待布逻辑单元块线网连接的数量，线网的统计包含了直接相连和跨一级相连（在一条路径中，隔一个中间结点实现相连）。

特征四：线长代价。根据半周长线网框的线长计算方法，对于待布逻辑单元块中输入输出端线网（线网已出现在已布逻辑单元块或该待布逻辑单元块中），提取待布逻辑单元块位于不同位置时对线长的影响。

特征五：需求的逻辑单元块数量。该特征以解析的电路网表为基础，用于提取已布逻辑单元块坐标位置附近需求的逻辑单元块数量。当一个逻辑单元块确定位置之后，则可以分别对该逻辑单元块每个输出端线网源结点的逻辑单元块需求数量统计，并累计求和得到每个逻辑单元块的该特征数据。计算方法如图6所示，假设动态电路网表解析之后，图**1**中逻辑单元块输出端源结点Node C在以Node A为根结点的树中，为统计所在位置Node C源结点对逻辑单元块的需求数量*Demand\_C*，分别记录以Node C为根结点的子树中深度在[0，*Nlen*]，[Nlen，2*Nlen*],[ 2*Nlen*，3*Nlen*]的子结点数量*Num*1,*Num*2,*Num*3，其中*Nlen*等于，最后由式（1）求得（注意到Node G由于出现在已布逻辑单元块中，则该结点以及其子结点不统计在内）。

 (1)



图6 源结点需求的逻辑单元块数量

Fig.6 Source node’s demand of logic elements

特征六：时序信息特征。时序部分特征包含多个子特征。时序优化作为VPR中时序驱动模拟退火算法的主要目标，是决定布局结果的关键。本文中时序部分信息的提取采用了VPR布局中的延时模型。

1） 结点时序最大值。结点时序最大值表示FPGA资源阵列的坐标位置上已布逻辑单元块的所有结点中最长延时的最大值。

2） 输入端线网源端时序值。该特征提取待布逻辑单元块输入端线网源端结点的最长延时，与结点时序最大值特征考虑全局时序信息不同的是，该特征考虑局部与待布逻辑单元块输入端直接相连的时序信息。

3） 输出端线网漏端时序值。该特征提取待布逻辑单元块输出端线网漏端结点的最长延时。

在FPGA资源阵列每个特征中无效或者没有实际值的坐标位置都取值为0，当FPGA资源阵列的坐标位置包含多个逻辑单元块时，各个特征在该坐标位置的取值为与各个逻辑单元块对应特征值的平均值。

**2.4** **待布逻辑单元块的选择**

|  |  |  |
| --- | --- | --- |
| 表3 FPGA的簇映射 | | |
| Tab.3 Cluster mapping of FPGA | | |
| 有效资源  阵列大小 | 逻辑单元簇大小 | 簇阵  列 |
| [0 ,30] | [2,2] | [99,99] |
| [30,50] | [3,3] | [98,98] |
| [50,70] | [4,4] | [97,97] |
| [70,100] | [5,5] | [96,96] |

随着布局过程的进行需要动态选择下一个待布的逻辑单元块。根据PEKO思想，一个好的布局结果，倾向于将线网连接的逻辑单元块布局紧凑[9],[13]，所以将在FPGA资源阵列中坐标位置靠近的多组逻辑单元块集合为一个逻辑单元簇。如表**3**所示为FPGA有效资源阵列参数*n*大小与选取的逻辑单元簇大小的对应关系，确定了逻辑单元簇大小之后，FPGA资源阵列映射后得到的FPGA簇阵列的阵列规模也如表中所示。对FPGA簇阵列中每个坐标位置计算未布逻辑单元块中与对应逻辑单元簇范围内已布逻辑单元块亲密度加权和最大的逻辑单元块。每次选择所有簇中与簇内逻辑单元块亲密度最大的逻辑单元块作为待布逻辑单元块。

待布逻辑单元块的选择分为两次筛选过程：首先选出与逻辑单元簇中逻辑单元块线网连接的数量之和最大的未布逻辑单元块。然后在其基础上选出与逻辑单元簇中逻辑单元块亲密度加权和最大的未布逻辑单元块。两个逻辑单元块*clb*1，*clb*2之间的亲密度*Correlation(clb*1*,clb*2*)*由式（2），（3）可得，*Basic*val称为逻辑单元块的基础价值，在动态电路网表解析后，如图1输出端线网源结点Node C，*Tot*node为解析后网表中Node C所在树的总结点数目，*Rm*node为Node C为根结点的子树的总结点数目，*Cur*len为Node C结点的深度，*Tot*len为该结点所在树的最大深度。α为经验参数，这里取值为0.25。*Connection(clb*1*,clb*2*)*即为两个逻辑单元块线网连接数量特征的值。由式（5）中*Correlation(clb*1*,cluster)*为clb1与逻辑单元簇*cluster*中逻辑单元块的亲密度加权和。其中*X*ave和*Y*ave认为是*clb*1与逻辑单元簇亲密度计算下的重心位置，*X*ave由式（4）计算得到，*Xclb*2为*clb*2在该逻辑单元簇中的相对位置，*Y*ave同理。

 **（2）**

 （3）

 （4）

 （5）

**2.5 快速详细布局**

**FPGA资源阵列中可能包含一个或者多个逻辑单元块，在处理其布局流程时略有不同，主要为：一，处理各个特征时，对应各个坐标位置的取值为与各个逻辑单元块对应特征值的平均值；二，由于预测位置得到的是在FPGA资源阵列中的位置，处理大规模电路布局过程中位置坐标中包含多个逻辑单元块的情况，预测的位置不是精确映射回FPGA芯片阵列。**为了处理上述问题，并且进一步优化时序，可以对模型训练之后的布局结果进行详细布局。采用基于交换的详细布局方法，为实现更有效交换从而加快详细布局速度，对需要交换的逻辑单元块和其进行交换的区间进行约束。待交换的逻辑单元块是与延时裕量（slack）最低15%路径关联的逻辑单元块[15]，交换区间的选择基于可行区域[16]进行优化，经过多次迭代得到最终的布局结果。

 

图7 基于线网Net1的可行区域 图8 CLB1可行区域的分级

Figure.7 Feasible range of Net1 Figure.8 Classfication of feasible range of CLB1

优化策略上，首先使用最小延时时序矩阵对可行区域进行进一步提取，其中最小延时时序矩阵*F*(*x*,*y*)是VPR布局阶段使用全局布线资源的延时分析矩阵，记录位置坐标相差(*x*,*y*)的两个结点之间的全局布线之间的最小延时，虽然不能准确估算布线之后的路径延时，但是可以在布局阶段给出合理的参考。如图7所示，假设CLB1中*Net*1，*Net*2，*Net*3分别是网表中slack最低15%的线网路径，以*Net*1为例，（*x*1,*y*1）和（*x*2,*y*2）是决定线网输入端(*xs*,*ys*)最大延时的结点，可行区域如图中ABCD所示。经过最小延时时序矩阵计算，保留可行区域中使得(*x*1,*y*1),(*x*2,*y*2)到（*xt*,*yt*）延时更小的部分区间。其次，对于连接到多条时序上高关键度的路径的逻辑单元块，分别对每条路径的可行区域进行评估，并优先使用区间重叠区域进行交换。假设图8为图7中线网*Net*1,*Net*2,*Net*3分别对应的三个可行区域的交叠情况，区域1为三个可行区域重叠的区域，在进行位置选择时具有最高的优先级，其次是区域2，再是区域3。

3 模型训练与特征工程

为了训练用于确定待布逻辑单元块位置的深度学习模型，采用随机电路网表生成器和最优线长电路网表生成器生成的网表用于产生模型训练数据，分别为270万和30万数据。训练环境为Linux系统下Keras 2.1.0（基于Tensorflow 1.4.0），Xeon E5-2620处理器，4块TITAN-V和3块TITAN-X GPU，在初始学习率0.003下经过20 epoches训练得到最终的模型。在过滤掉无效和不合法数据之后提取top5模型预测准确度（标签位置在预测位置概率分布中前五的精确度）。并针对多组特征进行分别训练，分析不同特征对模型训练的影响。

表4为不同特征组合进行模型训练得到的结果，f**1**-f**5**分别为特征一到特征五，f**6-1**,f**6-2**,f**6-3**分别为特征六的三个特征，总共进行了DL1-DL9九组对照分析。可以看到不同特征对模型的影响差别各异，同时采用六个完整特征时得到最高的模型预测准确度，预测达到41.3%的top5准确度。

表4 不同特征组合和对应的预测精确度

Tab.4 Different features and corresponding predict accuracy

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 模型序号 | f**1** | f**2** | f**3** | f**4** | f**5** | f**6-1** | f**6-2,3** | 精确度/% |
| DL**1** | \* | \* | - | - | - | - | - | 3.06 |
| DL**2** | \* | \* | - | - | \* | - | - | 11.6 |
| DL**3** | \* | \* | \* | \* | - | - | - | 14.3 |
| DL**4** | \* | \* | \* | \* | \* | - | - | 25.2 |
| DL**5** | \* | \* | \* | - | \* | \* | \* | 38.3 |
| DL**6** | \* | \* | \* | \* | \* | \* | - | 36.9 |
| DL**7** | \* | \* | \* | \* | \* | - | \* | 36.7 |
| DL**8** | - | - | \* | \* | \* | \* | \* | 14.8 |
| DL**9** | \* | \* | \* | \* | \* | \* | \* | 41.3 |

注：“\*”表示为模型序号中包含的特征，“-”则表示该模型序号中没有使用的特征。

4 实验结果

实验采用深度学习特征组合DL9训练得到的模型进行测试，调用模型进行位置预测后进行快速详细布局优化。在与模型训练的相同运行环境下使用MCNC基准电路[12]与VPR 8.0中使用的模拟退火布局算法进行对比。使用的FPGA为通用岛形结构k4\_n4\_v7\_bidir.xml，实验结果如表5所示：

表5 基于深度学习的布局算法与VPR在MCNC下对比

Tab.5 Comparison of DL-based placement with VPR in MCNC

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 电路  名称 | 逻辑单元块数量/个 | 关键路径延时比 | 运行速  度比 | 电路  名称 | 逻辑单元块数量/个 | 关键路径  延时比 | 运行速  度比 |
| or1200 | 3 627 | 1.077 | 1.59 | stereovision0 | 4 385 | 1.056 | 2.44 |
| raygentop | 5 437 | 1.073 | 2.49 | stereovision1 | 9 630 | 1.102 | 6.83 |
| bgm | 16 916 | 1.093 | 9.35 | stereovision2 | 17 121 | 1.093 | 9.21 |
| LU8PEEng | 30 894 | 1.114 | 18.7 | LU32PEEng | 89 262 | 1.123 | 53.3 |
| mcml | 106 865 | 1.126 | 63.0 | arm\_core | 137 154 | 1.129 | 78.5 |
| - | - | 1.098 | 24.54 |  |  |  |  |

表5可以看出，尽管在布线后关键路径延时上，基于深度学习的布局算法与VPR相比关键路径延时相差大致为9.8%，但是从运行时间上看，基于深度学习的布局算法能够明显提升运行速度，提升在1.59到78.5倍，平均上看运行速度提升在24.54倍，其中接近十万量级的大规模电路实现近似64.9倍的运行速度提升。分析数据可以看出，布局的加速效果受电路规模影响很大，这是由于基于交换的模拟退火算法交换空间随着电路规模的增大成指数级增长，因而模拟退火算法在处理类似mcml,arm\_core等规模大的电路交换次数达到数亿次，而基于深度学习的布局算法布局的次数与可交换逻辑单元块数量相同，快速详细布局部分只有少量的时序优化交换次数，总的来说，在处理大规模电路时该算法的运行速度能够得到非常大的提升，这也适应于实际应用。

5 总结

本文基于FPGA芯片描述k4\_n4\_v7\_bidir.xml将布局过程转化为逐步进行待布逻辑单元块的选择和逻辑单元块位置确定的过程。利用残差网络对数据进行训练，得到在六组特征下41.3%的top5预测精确度。该算法对MCNC基准电路与VPR相比在平均9.8%的关键路径延时损失下实现了平均运行速度24.54倍的提升，在处理大规模电路时有明显的时间优势。实际上本文中的算法可以支持复杂的可配置逻辑块的结构，也体现深度学习工具在处理FPGA这样的NP问题有实际的意义。

**参考文献：**

[1] Murray K E, Whitty S, Liu S, et al. Titan: Enabling large and complex benchmarks in academic CAD[C]// International Conference on Field Programmable Logic & Applications. Porto, Portugal: IEEE 2013: 1-8.

[2] Sait S M, Youssef H. VLSI Physical Design Automation[J]. *McGRAW-Hill Book Company*, 1995.

[3] Betz V , Rose J . VPR: a new packing, placement and routing tool for FPGA research[C]// International Workshop on Field Programmable Logic and Applications. Springer, Berlin, Heidelberg, 1997: 213-222.

[4] Maidee P , Ababei C , Bazargan K . Timing-driven partitioning-based placement for island style FPGAs[J]. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2005, 24(3):395-406.

[5] Xu N Y, Khalid M A S. QPF: efficient quadratic placement for FPGAs[C]// International Conference on Field Programmable Logic & Applications. Tampere, Finland: IEEE, 2005: 555-558.

[6] Gopalakrishnan P, Li X, Pileggi L. Architecture-aware FPGA placement using metric embedding[C]// IEEE Design Automation Conference. San Francisco, CA, USA: IEEE, 2006: 460-465.

[7] Xu M , Grewal G , Areibi S . StarPlace: A new analytic method for FPGA placement[J]. *Integration the Vlsi Journal*, 2011, 44(3):192-204.

[8] Yuan J , Wang L , Zhou X , et al. RBSA: Range-based simulated annealing for FPGA placement[C]// International Conference on Field Programmable Technology.Melbourne, Australia: IEEE, 2017: 1-8.

[9] An M , Steffan J G , Betz V . Speeding Up FPGA Placement: Parallel Algorithms and Methods[C]// 2014 IEEE 22nd Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM). Boston, Massachusetts, USA: IEEE, 2014: 178-185.

[10] Schmidhuber, Jürgen. Deep learning in neural networks: An overview[J]. *Neural Networks*, 2015, 61:85-117.

[11] D. Maarouf, A. Alhyari, Z. Abuowaimer . Machine-Learning Based Congestion Estimation for Modern FPGAs[C]// International Conference on Field Programmable Logic & Applications. Dublin, Ireland: IEEE, 2018: 427-434.

[12] Rose J , Luu J , Yu C W , et al. The VTR project:architecture and CAD for FPGAs from verilog to routing[J]. *Proceedings of the ACM/SIGDA international symposium on Field Programmable Gate Arrays*, 2012: 77-86.

[13] Chang C C , Cong J , Xie M . Optimality and scalability study of existing placement algorithms[J]. *Proceedings of the ASP-DAC Asia and South Pacific Design Automation Conference,* 2003: 621-627.

[14] He K, Zhang X, Ren S, et al. Identity Mappings in Deep Residual Networks[J]. *European Conference on Computer Vision*, 2016: 630-645.

[15] Vorwerk K , Kennings A , Greene J W . Improving Simulated Annealing-Based FPGA Placement With Directed Moves[J]. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2009, 28(2):179-192.

[16] Chen G , Cong J . Simultaneous timing-driven placement and duplication[J]. *Proceedings of the 2005 ACM/SIGDA 13th international symposium on Field-programmable gate arrays*,2005: 51-59.

**Deep Learning Based FPGA Fast Placement Algorithm**

LIU Wei, WANG Lingli, ZHOU Hao

(*State Key Laborlatory of ASIC & Systems, Fudan University, Shanghai 201203, China*)

**Abstract**：This paper proposes a new deep learning based FPGA placement algorithm, which implements the FPGA placement flow through a dynamic sequential process of selecting the logical element and determining the position of the logic element, thus achieving FPGA placement result of the netlist step-by-step. The choosen logic element block every time is placed on where the well-trained deep learning network predicts, and after the above process is completed, a fast detailed placement is used to optimize the previous placement result. Experiment is carried out to test the performance of the algorithm in MCNC benchmarks. For comparison, we use the simulated annealing based VPR to run these circuits. The result shows that under 9.8% increase of the critical path delay after routing, the entire runtime speed up is 24.54 times on average and for circuit netlist with about a hundred thousand exchangeable logical element blocks it rises to 64.9 times.

**Keywords**：field programmable gate array; deep learning; placement algorithm