文章编号：2095-4980(2020)03-0491-06

面向 eFPGA 的拼接式布线资源建模方法

涂开辉[1,2]，王鑫楠[2]，黄志洪[1]，杨海钢[*1,2]

(1.中国科学院大学，北京 100049；2.中国科学院 电子学研究所，北京 100190)

摘　要：嵌入式可编程门阵列核(eFPGA)在定制过程中的每一次迭代，都需要在新生成的布线资源图(RRG)上进行布线，进而完成该次迭代对面积/时序等参数的评估。传统的eFPGA RRG建图方法，在每次评估迭代时都需要重新生成全芯片的结构描述并在其基础上建立布线边和布线点，建图问题复杂度随芯片规模线性增大，很容易达到性能瓶颈。为了应对上述挑战，首先针对复用单元类型建立其RRG模型以及互连关系模型，然后采用一种根据资源排布关系，以动态拼接方式即时生成不同待评估阵列规模RRG的方法。实验证明，其相较于传统方法，在复用单元类型库不变的eFPGA评估过程中，依赖更小且近乎不变的数据库，建图总时间降低了约84%，内存峰值占用平均降低了约64%，从而提高了eFPGA的评估效率。

关键词：嵌入式可编程门阵列核(eFPGA)；Pathfinder算法；布线；布线资源图(RRG)；拼接

中图分类号：TN402　　**文献标志码：**A　　**doi：**10.11805/TKYDA2019053

Module-based routing resource graph modeling of eFPGA

TU Kaihui[1,2]，WANG Xinnan[2]，HUANG Zhihong[1]，YANG Haigang[*1,2]

(1.University of Chinese Academy of Sciences，Beijing 100049，China；
2.Institute of Electronics, Chinese Academy of Sciences，Beijing 100190，China)

Abstract：During each iteration of embedded Field Programmable Gate Array(eFPGA) customization process, router has to be run on a new Route Resource Graph(RRG) for the new architecture in order to meet design constraints in many aspects, such as area and timing. Conventional eFPGA RRG modeling method regenerates the whole chip's architecture description and builds routing nodes and edges based on it in every evaluation iteration. However, the efficiency of this method suffers from the rising scale of the chip being evaluated. A module-based RRG modeling method is proposed to address this problem. It firstly builds RRG for every module type and models interconnect relations among them, after that, stitches them together according to the device resource arrangement. It depends on a relatively small database, reduces the modeling runtime and memory peak footprint by around 84% and 64% respectively and thereby improves the eFPGA evaluation efficiency.

Keywords：embedded Field Programmable Gate Array(eFPGA)；pathfinder；routing；Route Resource Graph(RRG)；stitch

嵌入式可编程门阵列核(eFPGA)的出现，构建了一种新的商业模式，eFPGA 厂商不直接向终端用户提供 FPGA 芯片，而是提供给片上系统(System on Chip，SoC)集成商定制评估软件，由 SoC 集成商使用该软件定制评估出符合其需求的 eFPGA[1]。在评估过程中，需要动态地生成不同规模、资源的 eFPGA 结构，并使用评估电路在其上进行布局布线等电子设计自动化(Electronic Design Automation，EDA)流程，进而判断每一次定制的 eFPGA 面积和时序是否满足 SoC 应用电路的要求[2-3]。目前 FPGA 布线的主流算法为 PathFinder[4-5]，算法的运行需要在一个预先建立的布线资源图(RRG)上进行路径搜索。在 eFPGA 的评估过程中，一般依然采用 PathFinder 算法，但由于 eFPGA 的资源随每一次生成而不断变化，每一次布线流程所依赖的 RRG 也会随之变化。RRG 的建立基于对

收稿日期：2019-02-24；**修回日期：**2019-03-22
基金项目：国家自然科学基金资助项目(61876172；61704173)；北京市科技重大专项课题资助项目(Z171100000117019)
作者简介：涂开辉(1984–)，男，在读博士研究生，主要研究方向为大规模集成电路设计自动化。email:kelv@mail.ie.ac.cn
[*]**通信作者：**杨海钢　email:yanghg@mail.ie.ac.cn

芯片资源的建模，传统的建模方法中，复旦大学和深圳同创国芯公司申请的专利[6-7]，以及学术界的多功能布局布线工具(Versatile Place and Route，VPR)[8-10]均可针对 FPGA 成片进行建模，并从中提取 RRG，其方法是采用各自定义的格式进行芯片结构描述，但在 eFPGA 的应用场景中，类似的建模方式存在以下问题：a) 由于全芯片结构描述文件需要提前生成，适用于芯片结构已经固定的 FPGA 成片，但无法适应评估过程中芯片资源和结构根据需要可变的 eFPGA；b) 自定义的芯片结构描述需要人工定义参数，对于一些复杂模块，甚至需要全部手写，过程复杂且容易出错。针对 eFPGA 建模的文献[11-13]，通过参数化的模型自动产生评估器件的 RTL 作为后续 EDA 软件的输入，这种方法在芯片规模增大时，每次评估迭代都需要重新生成全芯片的 RTL 并在其基础上建立布线边和布线点，建图问题复杂度随芯片规模线性增大，很容易达到性能瓶颈。

为了解决上述问题，提出了一种在 eFPGA 评估迭代时动态拼接式构建 RRG 的方法，相比于上述方法具有以下特性：a) 只预先在复用单元(tile)级进行 RRG 建图，然后在 eFPGA 的评估迭代过程中使用动态拼接的方式，来满足可变大小 eFPGA 的评估需要；b) 由于对复用单元种类建立 RRG 所耗费的时间不随芯片规模的变化而变化，而建图时只需要考虑拼接处的布线点和布线边的连接关系，因此建图速度得以提高，峰值内存占用得以降低；c) 在复用单元级采用标记后的 Verilog 网表直接读入并解析的方法，精简了数据库，并且可以覆盖任意结构。

1 拼接式 RRG 建图方法

1.1 建图思想

eFPGA 和普通 FPGA 的芯核部分类似，由复用单元，例如可编程逻辑模块(Configurable Logic Block，CLB)、块存储器模块(Memory Block，MEM)、数字信号处理模块(Digital Signal Processor，DSP)等和它们之间的互连组成[14]，在定制过程中，这些复用单元以列为单位可进行一定程度上的自由组合。由于这些复用单元类型在一款芯片定制的过程中是固定不变的，因此可将其作为共享信息，提前将其 RRG 建立好。基于上述思想，图 1 中首先对共享信息进行建模，主要包括各复用单元种类以及它们之间的连接关系(connexion)，作为共享信息库(shared info library)。在 eFPGA 评估布线时，再根据当前迭代的资源排布信息，从共享信息库中获取相应复用单元级布线资源模型和其相关的连接关系信息进行拼接，形成布线算法在当前评估迭代中所使用的 RRG。

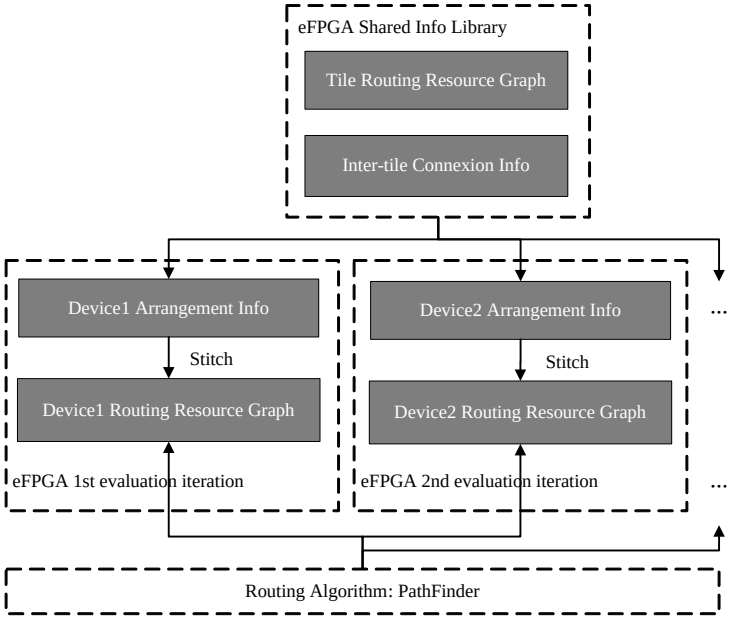


Fig.1 Flow of stitched RRG modeling
图 1 拼接式布线资源建图总体流程

1.2 eFPGA 共享信息库建模

1.2.1 复用单元布线资源(tile_graph)建模

选择复用单元的 Verilog 网表文件(tile_ netlist)作为输入，借助 iVerilog[15]开源程序进行解析，从而获得该复用单元的顶层模块(top-module)、子模块(sub-module)、端口(port)以及线网(wire)信息。对复用单元的布线资源建模，即对解析出的上述资源进行布线点和布线边的抽象化。在建模之前，为了让 PathFinder 布线算法可以识别布线路径的起始节点(source)、通道节点(channel)以及终结节点(sink)，首先对相关资源进行如下标记：

1) 布线模块(route module)标签：对复用单元网表文件中的每个模块进行分类标记，该标记被称为“布线模块标签”。布线模块标签依据功能划分为以下几类：“TOP”用于标记顶层模块；“CONNECTION”用于标记可编程连接性质模块，例如开关盒(SwitchBox)、连接盒(ConnectionBox)等；“BRIDGE”用于标记表示直接连接作用模块，例如电阻、反相器、缓冲器等；“THIGH”用于标记恒定高电平模块，例如恒高(TieHigh)模块；“TLOW”用于标记恒定低电平模块，例如恒低(TieLow)模块；其余逻辑功能子模块标记为“FUNCTION”类型，例如查找表(LUT)、触发器(FF)等。图 2 以可配置 CLB 类型的复用单元为例说明以上标记规则。

2) 布线节点(route node)标签：对复用单元网表文件中的端口和连接按照其所在模块的布线模块标签进行分类标记，该标记被称为“布线节点标签”。布线节点标签依据功能划分为 5 种：分别是 SOURCE 类型、SINK 类型、CONSTHIGH 类型、CONSTLOW 类型以及 CHAN 类型。其中，SOURCE 类型节点为一般布线路径的起点，FUNCTION 类型模块的输出端口定义为此类型；SINK 类型节点为一般布线路径的终点，FUNCTION 类型模块的输入端口定义为此类型；CONSTHIGH 类型节点，专指固定高电平，为 VCC 信号布线路径的起点，THIGH 类型模块的输出端口定义成此类型；CONSTLOW 类型节点，专指固定低电平，为 GND 信号布线路径的起点，TLOW 类型模块的输出端口定义成此类型；CHAN 类型节点为布线路径上的中间节点，CONNECTION 类型模块的一组连接定义为一个 CHAN 类型有效点，CHAN 类型有效点相对特殊，并不是对应某一个端口，而是对应一组连接关系，所以涉及的端口同时包含 input 和 output。值得注意的是，由于 BRIDGE 模块类型只起桥接作用，并不需要布线做出选择，因此不需在此建立节点，但若此模块桥接了两个布线节点，则后续会为其建立一条布线边。布线模块以及布线节点标签定义见表 1。

```
1  module BLK_TILE ( blk_cout, blk_rcascout,......);
2    output blk_cout, blk_rcascout;
3    ......
4    wire   [9:0] Icb_lab_lea;
5    wire   [9:0] Icb_lab_leb;
6    wire   [9:0] Icb_lab_cout0;
7    wire   [9:0] Icb_lab_cout1;
8    ......
9  LE_CELL  Icb_lab_Ilab_Ile_S0(.cout0(Icb_lab_cout0[0]),.cout1(Icb_lab_cout1[0]),
10 .lea(Icb_lab_lea[0]),.leb (Icb_lab_leb[0]),......);
11 LE_CELL  Icb_lab_Ilab_Ile_S9(.cout0(Icb_lab_cout0[9]),.cout1(Icb_lab_cout1[9]),
12 .lea(Icb_lab_lea[9]),.leb (Icb_lab_leb[9]),......);
13 ......
14 SWITCH_CELL  Icb_Isw_N10(.y(so_18[0],.din0(channel_leftlink[0]),.din1(ei_14[0])),
15 ......);
16 ......
17 TIEHI_CELL Iclab_Itiehi_0__Itiehi (.y(Iclab_Itiehi_hi[0]));
18 TIELO_CELL Iclab_Itielo_0__Itielo (.y(Iclab_Itielo_lo[0]));
19
20 RES_CELL  Iswb_Isw_N13_U0_9_ (.in(ei_14[19]),.out(wo_14[29]));
21 RES_CELL  Iswb_Isw_N13_U0_8_ (.in(ei_14[18]),.out(wo_14[28]));
22 ......
23 endmodule
```

TOP　FUNCTION　CONNECTION　THIGH　TLOW　BRIDGE

Fig.2 Route module label in a tile
图 2 复用单元中的布线模块标签

表 1 布线模块以及布线节点标签定义
Table1 Definition of route module and route node labels

route module label	port type	route node label
TOP	/	none
FUNCTION	output	SOURCE
FUNCTION	input	SINK
THIGH	output	CONSTHIGH
TLOW	output	CONSTLOW
CONNECTION	input and output	CHAN
BRIDGE	/	none

在标记完成之后，只需遍历复用模块中所有的子模块，根据布线节点标签提取全部布线节点集合，再依据网表文件中描述的各个子模块之间通过端口建立的连接关系创建对应布线节点的连接，即布线边，从而得到完整的有向图，即该复用单元的 RRG。由于有布线标签的存在，PathFinder 布线算法在读取 RRG 时即可清楚地知道哪些点可作为布线起点(source)，哪些点可作为布线终点(sink)等，从而选择相应的节点完成布线。这一点和传统建模将上述信息写入自定义描述文件的方法有所不同，由于绝大部分信息来源于复用单元的 Verilog，不但可以覆盖任意结构，且省去了大量人工定义复用单元结构描述文件的时间。

1.2.2 复用单元连接关系(tile_connexion)建模

以复用单元种类为单位建立其输出端口与其他复用单元的输入端口之间的连接关系。图 3 以可配置 CLB 类型的复用单元为例，使用 XML 语言描述着色 CLB 的连接关系：OUTPORT 语句依次列出该复用单元所有输出端口的连接情况，包含端口名、端口宽度属性信息，CONNEXION 语句则描述了输出端口与可连接的其他复用单元的输入端口的连接关系，属性依次描述可连接的其他复用单元的相对坐标信息、输入端口名。

```
1 <TILE name="CLB">
2     <OUTPORT name="channel_link" width="3">
3         <CONNEXION delta_x="1" delta_y="0" port_name="CLB.in"/>
4         <CONNEXION delta_x="-1" delta_y="0" port_name="IOB.rlink"/>
5         <CONNEXION delta_x="0" delta_y="1" port_name="IOB.blink"/>
6         ......
7     </OUTPORT>
8     ......
9 </TILE>
```

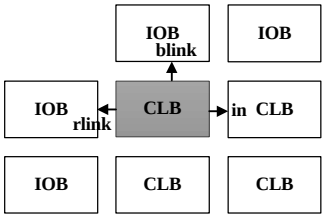


Fig.3 Tile connexion modeling
图 3 复用单元连接关系建模

1.3 eFPGA 资源排布信息建模

每次 eFPGA 的评估迭代都会生成一个体现当前资源排布情况的排布信息文件(device_arrangement)。以图 4 为例，该文件标明了当前定制芯片中每个坐标位置下放置何种复用单元。

TILE 语句描述 FPGA 器件所使用的复用单元类型，TILE_INS 语句描述该类型实例分布的坐标位置。

1.4 eFPGA 评估器件布线资源(device_graph)建图

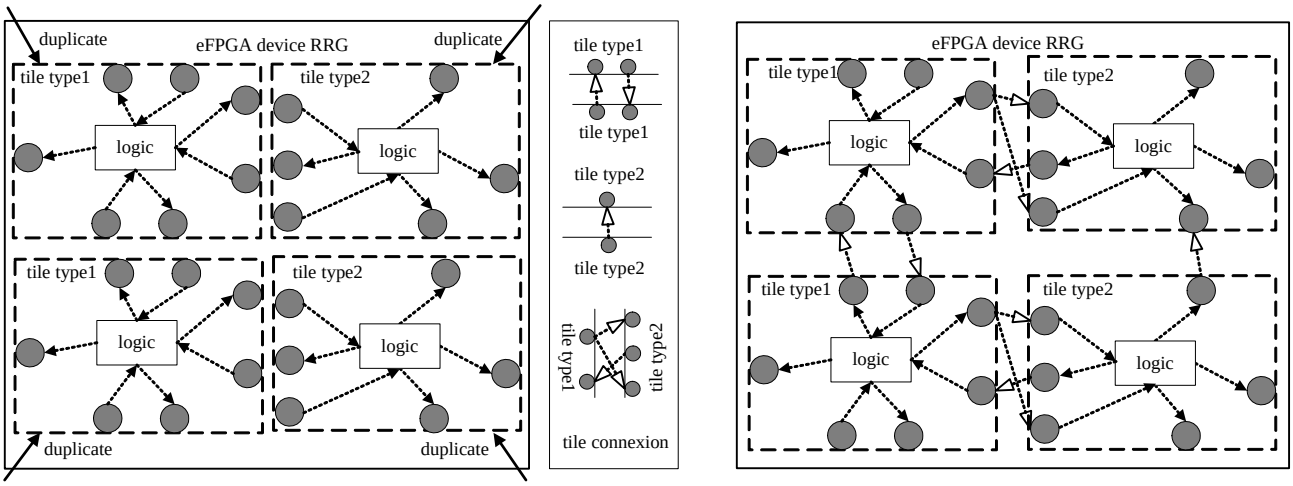
基于 1.1 节描述的建图思想，根据复用单元的共享信息库以及当前迭代下的排布关系，建立当前评估迭代下 eFPGA 器件的 RRG 流程如下：

1) 遍历器件排布信息中每个坐标下的复用单元，根据其类型从芯片共享信息库中获得该复用单元的 RRG(tile_graph)，并将其中的布线节点、布线边以及标签信息全部复制到器件 RRG 中去，示意见图 5(a)。

```
1  <DEVICE series="series1" name="device1" size_x="8" size_y="8">
2      <TILE name="BLK" size_x="1" size_y="1">
3          <TILE_INS  loc_x="1:8" loc_y="1:8"></TILE_INS>
4      </TILE>
5      <TILE name="IOB_T" size_x="1" size_y="1">
6          <TILE_INS  loc_x="1,3,5,7" loc_y="9"></TILE_INS>
7      </TILE>
8      <TILE name="IOB_B"  size_x="1" size_y="1">
9          <TILE_INS  loc_x="1,3,5,7" loc_y="0"></TILE_INS>
10     </TILE>
11     ......
12 </DEVICE>
```

Fig.4 Device arrangement modeling
图 4 器件排布关系建模

2) 再次遍历每个坐标下的复用单元，根据其类型从芯片共享信息库中获得该复用单元的连接关系(tile_connexion)，并在其中遍历该复用单元每个对外连接端口的连接信息(通常由一个输出端口和多个输入端口组成)，接下来由输出端口和其所在的坐标获取器件布线资源图 device_graph 中相应的输出布线节点(out_nodes)，并根据其连向的输入端口使用同样的方法获得相应的输入布线节点(in_nodes)，最后为每一个输出和输入布线节点建立布线边并写入器件 RRG，示意见图 5(b)。



(a) duplicate tile RRGs into the eFPGA device RRG (b) stitched eFPGA device RRG

Fig.5 Stitching of eFPGA device routing resources
图 5 eFPGA 器件布线资源拼接示意图

该过程的伪代码为：

```
new device_graph //current device's RRG
for every tile of the device do//duplicate tile_graph's nodes and edges
    tile_type = device_arrangement(i,j);}//i,j is the coordinate of current tile
    tile_graph = GetTileRRG(series_library,tile_type);
    for all nodes, n, in tile_graph do
        Add n to device_graph
        for all edges, e, in tile_graph do
            Add e to device_graph
        end for
    end for
end for
for every tile of the device //build inter-tile connections
    tile_type = device_arrangement(i,j); //i,j is the coordinate of current tile
    tile_conn = GetTileConn(series_library,tile_type);
    for every port connection, port_conn, in tile_conn do
        out_nodes = tile_graph(port_conn.outport,i,j); //get out_nodes using outport
        list of in_nodes = tile_graph(port_conn.inports,k,l)} k,l is the possible connected inport's tile coordinate
        for each out_nodes do
            for each in_nodes do
                Build edges from out_nodes to in_nodes in device_graph
            end for
        end for
```

```
    end for
end for
```

2　实验结果

针对传统结构建图和拼接式 RRG 建图方法的依赖数据库、执行效率和计算资源占用情况进行了实验比较并完成统计分析。实验条件如下：a) 使用 3 GHz 主频 CPU，8GB 内存的 PC。b) 评估的 eFPGA 仅含 1×1 单位坐标大小的 CLB 以及互连资源(CLB 是最具代表性的复用单元,且仅使用 CLB 在简化问题的同时不影响对比结果)。c) 以 7 款待评估 eFPGA 作为建图对象，资源规模从 24×24 个 CLB(30 万门左右)到 128×128 个 CLB(超过 1 000 万门)不等，这个规模范围覆盖了目前主流的 eFPGA 大小。d) 统一采用 Verilog 模型描述芯片结构以消除不同结构描述格式所带来的偏差。

2 种进行对比的 RRG 建图方法如下：

方法 1：依照传统方法[11-13]的思想，对 7 次评估迭代下不同阵列规模的 eFPGA 器件(24×24~128×128)分别先建立 Verilog 模型，再对其进行提取并建图。

方法 2：依照本方法，对 7 次评估迭代下不同阵列规模的 eFPGA 器件(24×24~128×128)进行建图。

实验主要针对以下方面进行评估：

1) 相关数据库大小：方法 2(本方法)所依赖的数据库主要包括所有复用单元的网表文件、复用单元的连接关系文件以及器件排布关系文件。对于复用单元类型库不变的 eFPGA 评估过程来说，仅在器件排布关系上有微小变化。而方法 1 所依赖的数据库由于包含全芯片的网表信息，会随着芯片规模的增大而线性增长。2 种方法所依赖的数据库文件大小对比见表 2。

表 2 两种方法所依赖的数据库文件大小对比

Table2 Depending database file size comparison of the two methods

Eval. iteration	device scale	method 1/MB	method 2(this paper)/MB
1	24×24	4.8	0.5
2	40×40	6.4	0.5
3	64×64	8.7	0.5
4	72×72	9.5	0.5
5	80×80	10.1	0.5
6	96×96	11.7	0.5
7	128×128	16.0	0.6

2) 建图时间：通过表 3 可以看出，方法 2(本方法)由于在芯片规模增大时，在建立布线边和布线点时避免了对全器件资源进行搜索，而是只对拼接边缘处的布线边和布线点进行搜索，7 次迭代的总运行时间也因此降低了约 84%。

表 3 2 种方法的布线资源图建立时间对比

Table3 RRG building time comparison of the two methods

Eval. iteration	device scale	method 1/s	method 2(this paper)/s	method 2/method 1/%
1	24×24	29	10	34
2	40×40	56	12	21
3	64×64	90	15	17
4	72×72	115	18	16
5	80×80	142	21	15
6	96×96	210	29	14
7	128×128	366	60	16
total		1 008	165	16

3) 内存峰值开销：对于建图来说，过大的内存峰值开销会导致程序内存不够而崩溃(实验使用了较大的 8 GB 内存故没有出现这种情况，但如果内存为 2 GB，则方法 1 的第 6、7 次迭代就会因此而崩溃)。方法 2(本方法)由于避免了直接读入全芯片 Verilog 网表而带来冗余的信息(例如无需建立布线点和布线边的资源信息以及重复的 tile_graph/connexion 信息等)，在内存峰值开销方面，也得到了很大改善，见表 4，7 次迭代中平均约降低了 64%。

表 4 两种方法的内存峰值开销对比

Table4 Memory peak footprint comparison of the two methods

Eval. iteration	device scale	method 1/MB	method 2(this paper)/MB	method 2/method 1/%
1	24×24	410	82	20
2	40×40	822	306	37
3	64×64	1 385	528	38
4	72×72	1 664	631	38
5	80×80	1 974	706	36
6	96×96	2 575	1 022	40
7	128×128	4 021	1 617	40
average		1 836	699	36

3　结论

本文提出了一种高效的 eFPGA RRG 构建方法。和传统方法相比，仅依赖很小的数据库，且信息直接来源于设计网表，省去了产生复用单元级结构描述文件的时间。在评估过程中使用了动态拼接的方式在每次迭代中即时生成 RRG。实验结果表明，与传统的读入全芯片 RTL 模型的建图方法相比，依赖更小且近乎不变的数据库，且

在建图时间上减少了约84%，内存峰值开销约降低了64%，从而可以充分降低eFPGA多次迭代评估过程中的计算资源需求，提高迭代评估的效率，以达到快速定制贴合用户需求的eFPGA结构的目的。

参考文献：

[1] NOLL T G. Application specific eFPGAs for SoC platforms[C]// IEEE VLSI-TSA International Symposium on VLSI Design Automation and Test. Hsinchu,Taiwan,China:IEEE, 2005:348.

[2] ACHRONIX Corp. Achronix product brief[EB/OL]. (2019-01)[2019-04]. https://www.achronix.com/wp-content/uploads/2017/05/Speedcore_ eFPGA_Product_BriefPB028.pdf.

[3] FLEXLOGIX Corp. Embedded FPGA basics[EB/OL]. (2018-06)[2019-04]. http://www.flex-logix.com/fpga-tutorial.html.

[4] VENUGOPAL N,MANIMEGALAI R. Survey on FPGA routing techniques[J]. International Journal on Computer Science and Engineering(IJCSE). 2012,4(7):1304-1309.

[5] MCMURCHIE L,EBELING C. PathFinder:a negotiation-based performance-driven router for FPGAs[C]// Third International ACM Symposium on Field-Programmable Gate Arrays. Napa Valley,USA:[s.n.], 1995:111-117.

[6] 王健,来金梅,余建德,等. 可编程逻辑器件硬件结构通用建模方法:中国,CN101246510A[P]. 2010-12-29. (WANG Jian,LAI Jinmei,YU Jiande,et al. Programmable logic device hard structure universal modeling method:China. CN101246510A[P]. 2010-12-29.)

[7] 许明亮. FPGA建模方法及装置:中国,CN105740520A[P]. 2016-07-06. (XU Mingliang. FPGA modeling method and device:China. CN105740520A[P]. 2016-07-06.)

[8] ROSE J,LUU J,YU C W,et al. The VTR project: architecture and CAD for FPGAs from Verilog to routing[C]// 20th ACM/SIGDA International Symposium on Field Programmable Gate Arrays(FPGA). Monterey,USA:[s.n.], 2012:77-86.

[9] BETZ V,ROSE J. Automatic generation of FPGA routing architectures from high-level descriptions[C]// ACM/SIGDA International Symposium on Field Programmable Gate Arrays. ACM,New York,NY:[s.n.], 2008:175-184.

[10] LI Zhihua,YANG Haigang,YANG Liqun,et al. Architecture model and resource graph building algorithm for detailed FPGA architecture design[J]. Journal of Electronics(China), 2014,31(6):505-512.

[11] COENEN T,SCHLEIFER J,WEIB O,et al. Interconnect routing of embedded FPGAs using standard VLSI routing tools[C]//International Symposium on System on Chip. Tampere,Finland:IEEE, 2010:121-124.

[12] NEUMANN B,SYDOM T,BLUME H,et al. Design flow for embedded FPGAs based on a flexible architecture template[C]// Design,Automation & Test in Europe. Munich,Germany:IEEE, 2008:56-61.

[13] CHEN Xiaolin,LI Shuai,SCHLEIFER J,et al. High-level modeling and synthesis for embedded FPGAs[C]// Design, Automation & Test in Europe Conference & Exhibition. Grenoble,France:IEEE, 2013:1565-1570.

[14] KUON I,TESSIER R,ROSE J. FPGA architecture:survey and challenges[J]. Foundations and Trends in Econometrics, 2008,2(2):135-253.

[15] WILLIAMS S. Icarus verilog[EB/OL]. (2016-08)[2019-04]. http://iverilog.icarus.com/index.html.