UPRFloor:一种动态可重构 FPGA 建模方法与布局策略

王今雨[1,2],伍卫国[1,2],秦朝楠[1],赵东方[1],聂世强[1]
(1. 西安交通大学电子与信息工程学院,陕西西安 710049;2. 西安交通大学国家数据广播工程中心,陕西西安 710049)

摘 要: 针对现场可编程门阵列(Field Programmable Gate Array,FPGA)布局过程中片上可重构资源利用率低与通信开销过高问题,本文提出了一种支持多描述模型的布局策略 Union Partial Reconfiguration Floorplans (UPRFloor). 首先,该策略根据逻辑功能客观形状,定义了矩形、非矩形多描述模型,然后利用混合整数线性规划方法,从可重构资源利用率、逻辑功能间通信开销与逻辑功能内部通信开销三个方面进行多目标优化,实现了三者之间相互影响与共同作用下的最优布局方案. 该策略已在 FPGA 芯片上进行了仿真布局,结果表明:与基于矩形模型的布局方法相比,UPRFloor 布局策略在资源利用率方面最高有 25.59% 的提升. 在 Microelectronics Center of North Carolina (MCNC)标准测试集上的对比实验表明:在耗时几乎相同的情况下,UPRFloor 较其它算法的布线长度最多减少了 22.49%;在 Software Defined Radio (SDR)测试数据中,UPRFloor 在节约 29.41% 可重构资源的同时,布线长度节省了 13.41%,从而有效降低了资源浪费与通信开销.

关键词: 现场可编程门阵列; 重构计算; 布局策略; 混合整数线性规划
中图分类号: TP301.6 **文献标识码:** A **文章编号:** 0372-2112 (2018)12-2862-08
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2018.12.007

UPRFloor:A Modeling and Floorplanner for Partially Reconfigurable FPGA Systems

WANG Jin-yu[1,2],WU Wei-guo[1,2],QIN Zhao-nan[1],ZHAO Dong-fang[1],NIE Shi-qiang[1]
(1. *The School of Electronic and Information Engineering,Xi'an Jiaotong University,Xi'an,Shaanxi* 710049,*China*;
2. *National Data Broadcast Engineering Center,Xi'an Jiaotong University,Xi'an,Shaanxi* 710049,*China*)

Abstract: In this paper,a multi-model based floorplanner named UPRFloor is proposed to save reconfigurable resources and reduce communication overhead during the floorplanning of FPGAs. According to the shape of logical functions,a description model which can depict both rectangular and non-rectangular shapes is firstly defined. Then,the Mixed-Integer Linear Programming (MILP) ideology is used to optimize an objective function which takes the waste of resources and all kinds of communication costs into account to obtain a desirable floorplan. Finally,the UPRFloor strategy has been simulated on FPGA chips and the results show that the strategy saves reconfigurable resources as much as 25.59%. The proposed method is validated on the data set from MCNC standard benchmark. The UPRFloor reduces 22.49% global wire length at most with almost the same time-consuming as other algorithms. Experiments conducted on the SDR design also demonstrate that the strategy saves reconfigurable resources as much as 29.41% and reduces 13.41% global wire length at most,which effectively reduces the wasting of resources the communication cost.

Key words: field programmable gate array;reconfigurable computing;floorplan;mixed-integer linear programming

1 引言

近年来,随着可重构技术的迅猛发展,现场可编程门阵列(FPGA)被广泛地应用于医疗、航空航天、高性能计算等需要硬件资源重复利用的领域. 动态部分可重构(Partial Reconfiguration,PR)[1]技术使 FPGA 芯片在不影响静态区域逻辑功能正常工作的前提下,可以实时在芯片(in chip)改变动态可重构区域的逻辑功能,通过这种分时复用的方式,使用少量可重构资源实现了更多的逻辑功能,能够在有效降低 FPGA 芯片资源浪费的同时,增强了系统灵活性. 同时,由于 FPGA 的并行性与扩展性强等特点,基于 FPGA 的硬件可编程加速器

收稿日期:2017-09-04;修回日期:2018-02-13;责任编辑:梅志强
基金项目:国家重点研发计划(No.2017YFB1001701);国家自然科学基金项目(No.61672423,No.61628210)

(Programmable Accelerators, PA)[2]在异构数据中心设计[3]、人工智能[4]等领域得到了广泛的应用,也越来越多的得到了研究者的关注.

相比超大规模集成电路(VLSI)只包含单一种类资源而言,主流 FPGA 芯片包含多种可重构资源[5],如图 1 所示为 FPGA 芯片可重构资源抽象图. 其中,蓝色代表可配置逻辑块(Configuration Logic Block,CLB),红色为块随机存储单元(Block RAM,BRAM),绿色为数字信号处理单元(Digital Signal Processing,DSP). FPGA 芯片中的资源数量有限,并且在芯片设计时各种资源数量与位置均已固定,不可变更,各类资源采用非均匀分布方式按列分布,如图 1 所示,BRAM 和 DSP 分别位于第 5 列,第 14 列和第 17 列,非均匀的分布于 FPGA 芯片中. 资源非均匀分布式 FPGA 的布局算法更加复杂,也更耗时. 图中红色阴影部分为一个逻辑功能在芯片上的布局结果,该逻辑功能包含 8 个 CLB 单元和 6 个 BRAM 单元. 每个逻辑功能所需资源种类与数量各不相同,布局算法在满足其所需要的资源类型与数量的前提下,还需要考虑各个待布局逻辑功能之间的通信开销. 如何将多个逻辑功能在有限的 FPGA 芯片可重构资源中进行合理布局,是学术界广泛关注的热点问题,也是困难所在.

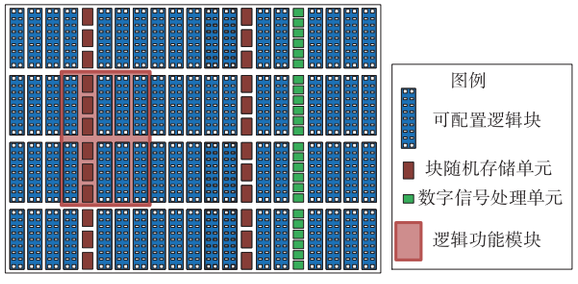
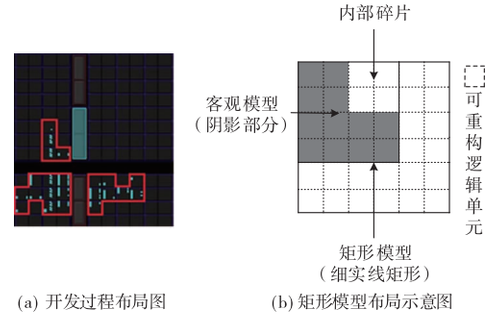


图1　FPGA可重构资源抽象图

FPGA 的布局是 NP-Hard 问题,主要实现方式分为以下两类:基于贪心算法、模拟退火等启发式算法的实现方法;基于混合整数线性规划(Mixed-Integer Linear Programming,MILP)的线性数学优化实现方法. 其中,多伦多大学的 V. Betz 等提出的 FPGA 芯片结构辅助开发工具(Verilog to Routing,VTR)[6,7]是学术 FPGA 研究中较为常用的辅助设计工具,该工具允许研究人员自由设计 FPGA 芯片结构、实现更高效的综合、布局布线与配置算法. VTR 采用的是基于模拟退火算法的布局策略,其优点是布局耗时较短,不足之处是容易陷入局部最优;文献[8]中使用启发式算法实现 FPGA 布局策略,该方法使用冲突图(conflict graph)表示所有逻辑功能的可行布局结果,使用贪心算法求解冲突图并且使用局部搜索算法来降低解空间.

相比启发式算法存在精度不高的问题,采用线性数学优化方法可以寻找到全局最优解或近优解,以 Marco 等[9]为代表,将 FPGA 布局问题转化为 MILP 求优问题,文献[10]中首先对芯片资源和待布局逻辑功能进行建模,然后设立芯片和逻辑功能需要满足的各项约束,最后确定优化目标函数. 采用该方法可以有效提高布局精度,但是由于求解时间过长,通常在待布局逻辑功能数量过大时需要设置求解器的运行时限,同时,随着 FPGA 芯片规模不断扩大,对其进行准确建模的难度也急剧增加.

上述各种 FPGA 的布局算法中,均以矩形模型作为待布局逻辑功能的描述模型,其具有易于表达,建模简单的优点. 但是在真实的开发过程中,逻辑功能的客观形状一般均不是矩形[11],如图 2(a)所示为基于 Xilinx ISE14. 6 开发环境下的部分逻辑功能布局图,其中红色实线框为实际占用的可重构资源. 采用矩形模型容易产生过多的内部碎片而导致资源利用率降低,图 2(b)为矩形模型示意图,可见因内部碎片而造成的资源浪费达 25%.



(a) 开发过程布局图　　(b) 矩形模型布局示意图

图2　真实逻辑功能布局与矩形抽象模型对比

布线长度是影响电路通信开销和能耗的重要因素[1],文献[12]将布局过程分为两个部分,首先将芯片按区域进行划分,目的在于降低资源浪费;然后在不改变占用资源总数量的前提下,通过调整布线长度来降低通信开销. 该方法虽然考虑了资源占用与通信开销两个方面,但是仅以单一因素作为约束进行优化,并未考虑两者之间的相互影响与共同作用.

针对上述问题,本文提出了一种逻辑功能多描述模型,该模型能够遵循逻辑功能的客观形状,选择矩形或非矩形的方式进行建模,有效减少内部碎片,从而提高了芯片可重构资源的利用率;提出了一种基于混合整数线性规划的 FPGA 可重构资源布局策略-UPRFloor(Union Partial Reconfiguration Floorplans),该策略支持逻辑功能多描述模型,综合考虑了资源占用、布线长度与任务纵横比等因素,从可重构资源利用率、逻辑功能间通信开销与逻辑功能内部通信开销三个方面进行多目标优化,实现了三者之间相互影响与共同作用下的最优布局方案.

2 逻辑功能多描述模型

使用集合 $T=\{T_1,T_2,T_3,\cdots,T_n\}$ 表示需要部署于FPGA芯片上的全部待布局逻辑功能,如图3(*a*)所示将某个待布局逻辑功 T_k 进行分块,纵向分为两个部分 $T_k=\{\text{part}_0,\text{part}_1\}$,每个部分用 $\text{part}_i\mid_{i=0,1}=\{x_i,y_i,w_i,h_i\}$ 表示,其中 x_i 表示该部分最左侧水平(横轴)坐标值,y_i 表示最左侧垂直(纵轴)坐标值,w_i 为宽度,h_i 为高度. 为确保同一逻辑功能 T_k 内两个部分 part_0,part_1 在垂直方向上连续,模型需满足约束式(1);式(2)将 part_0,part_1 在水平方向上约束为左对齐或右对齐,如图3所示. 进行该项约束的目的在于确保非矩形模型呈现类似L型形状,避免类似T型形状,降低了待布局逻辑功能模型的复杂度.

$$\forall p\in\text{part}_i,\ \forall\ \text{part}_i\in T:$$

$$y_0=y_1+h_1 \tag{1}$$

$$x_0=x_1\ \text{or}\ x_0+w_0=x_1+w_1 \tag{2}$$

使用上述建模方法,在定义了非矩形模型的同时,也可用于表示如图3(*e*)所示的矩形模型($w_0=w_1$),在实际布局过程中,根据各个逻辑功能所需资源种类与数量特点,灵活采用矩形、非矩形的方式进行建模.

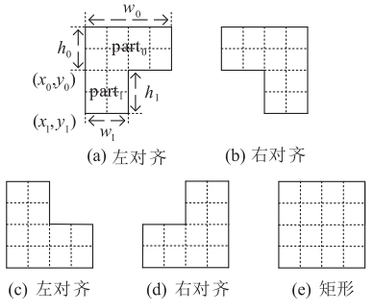


图3 逻辑功能多描述模型

3 UPRFloor布局策略

FPGA布局过程是寻找全局最优解或近优解的过程,UPRFloor布局策略采用MILP方法,将FPGA布局问题转化为线性数学优化问题. 图4为使用该策略进行布局的基本流程图:首先,UPRFloor将布局过程中不会发生变化的属性定义为常量;其次,通过一系列变量表示逻辑功能位置、资源占用等信息;然后,从语义与问题两个方面将上述常量与变量进行约束,确保布局结果的正确性;最后,通过优化由资源占用、布线长度与任务纵横比三项代价函数线性组合的目标函数,实现三者之间相互影响与共同作用下的最优布局方案.

3.1 常量定义

FPGA芯片的可重构资源分区数量与位置信息固定,待布局逻辑功能个数、所需资源数量与种类等信息在布局前也已确定,并且在布局过程中不会发生变化,

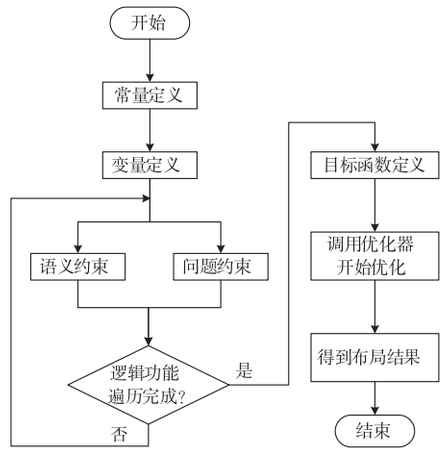


图4 UPRFloor布局策略基本流程图

所以在UPRFloor布局策略中使用常量表示上述信息,具体定义如表1所示.

表1 布局策略常量定义

名称	定义
F	FPGA芯片分区数量
H	FPGA芯片高度
W	FPGA芯片宽度
N	待布局逻辑功能数量
P	逻辑功能分块数量,取值为2
T	FPGA芯片资源种类集合,包括(CLB,BRAM,DSP等)
$r_{n,t}$	逻辑功能 n 所需 t 种资源数量
x_f^{l}	芯片分区 f 的左侧(起始)横坐标
x_f^{r}	芯片分区 f 的右侧(终止)横坐标
$d_{f,t}$	芯片分区 f 中一个最小可重构单元包含 t 种资源的数量

3.2 变量定义

逻辑功能的位置信息、逻辑功能占用FPGA可重构资源的情况、逻辑功能间有无位置重叠等信息在布局过程中会发生变化,因此使用如表2所示的一系列变量对其进行定义.

表2 布局策略变量定义

名称	定义
$z_{n,p,h}$	二值变量,1代表逻辑功能 n 的 p 分块占用了芯片的第 h 行
$x_{n,p}$	逻辑功能 n 的 p 分块左侧(起始)位置横坐标,范围$[0,W-1]$
$w_{n,p}$	逻辑功能 n 的 p 分块宽度,范围$[1,W-1]$
$y_{n,p}^{\text{l}}$	逻辑功能 n 的 p 分块最低(起始)位置纵坐标,范围$[0,H-1]$
$y_{n,p}^{\text{h}}$	逻辑功能 n 的 p 分块所占最高行,范围$[0,H-1]$
$h_{n,p}$	逻辑功能 n 的 p 分块高度(所占行数),范围$[1,H-1]$
$g_{n,p,f,h}$	逻辑功能 n 的 p 分块与芯片 f 分区在 h 行的交集,数值以最小可重构单元(一个时钟域)为单位,范围$[0,\infty]$
$k_{n,p,f}$	二值变量,0代表逻辑功能 n 的 p 分块与芯片 f 分区没有交集
r_{n_1,p_1,n_2,p_2}	二值变量,0代表两个逻辑功能 n_1,n_2 的各分块间没有位置重叠

3.3 约束定义

为了确保获得最优的布局,需要将上述常量与变

量进行规约,约束可分为语义约束和问题约束两类. 其中,语义约束用于保证布局策略正常实施与模型正确建模,如式(3)~(7),具体包括:各个变量的约束,确保其正确的取值范围、统一描述模型形状等. 问题约束关注布局结果,如式(8)~(14),具体包括:保证各个逻辑功能间布局不存在重叠、逻辑功能所需各类资源得到满足、满足各项通信开销等.

待布局逻辑功能横坐标位置约束,式(3)确保每一个分块横坐标在芯片范围内.

$$\forall n\in N,p\in P:\quad x_{n,p}+w_{n,p}-W\leq 0 \tag{3}$$

待布局逻辑功能纵向连续性约束,式(4)表示:如果待布局逻辑功能 n 的每一个分块 p 占用 h_{l}、h_{h}行,且有 $h_{\mathrm{l}}>h_{\mathrm{m}}>h_{\mathrm{h}}$,则该分块一定占用 h_{m}行.

$$\forall n\in N,p\in P,h_{\mathrm{l}},h_{\mathrm{m}},h_{\mathrm{h}}\in H\mid h_{\mathrm{l}}>h_{\mathrm{m}}>h_{\mathrm{h}}:\quad z_{n,p,h_{\mathrm{m}}}\geq z_{n,p,h_{\mathrm{l}}}+z_{n,p,h_{\mathrm{h}}}-1 \tag{4}$$

高度约束,式(5)确定了每一个分块高度,式(6)、(7)定义了其纵向坐标位置.

$$\forall n\in N,p\in P:\quad h_{n,p}=\sum_{h\in H}z_{n,p,h} \tag{5}$$

$$\forall n\in N,p\in P,h\in H:\quad y^{\mathrm{h}}_{n,p}-y^{\mathrm{l}}_{n,p}\geq(z_{n,p,h}-1)\cdot H \tag{6}$$

$$y^{\mathrm{h}}_{n,p}-y^{\mathrm{l}}_{n,p}+1=h_{n,p} \tag{7}$$

式(8)和式(9)通过逻辑功能分块 p 与芯片 f 分区横坐标的位置,定义二者的重叠关系.

$$\forall n\in N,p\in P,f\in F:\quad x^{\mathrm{l}}_{f}\cdot k_{n,p,f}\leq x_{n,p}+w_{n,p}-1 \tag{8}$$

$$x_{n,p}\leq x^{\mathrm{r}}_{f}+(1-k_{n,p,f})\cdot(W-x^{\mathrm{r}}_{f}) \tag{9}$$

式(10)表示根据 $z_{n,p,h}$,$k_{n,p,f}$,$w_{n,p}$计算每一个分块 p 与芯片 f 分区交集数值上限,式(11)规约分块 p 在左右两侧与 f 分区部分相交时重叠区域的范围.

$$\forall n\in N,p\in P,f\in F,h\in H:\quad g_{n,p,f,h}\leq\min(z_{n,p,h},k_{n,p,f},w_{n,p})\cdot(x^{\mathrm{r}}_{f}-x^{\mathrm{l}}_{f}+1) \tag{10}$$

$$g_{n,p,f,h}\leq x_{n,p}+w_{n,p}-k_{n,p,f}\cdot x^{\mathrm{l}}_{f}$$
$$g_{n,p,f,h}\leq W-x_{n,p}+1+z_{n,p,h}\cdot(x^{\mathrm{r}}_{f}-W) \tag{11}$$

考虑到完全重叠,即分块 p 在某一行 h 与 f 分区全部相交时,设定式(12)计算交集数值下限,通过式(13)和式(14)确定了 $g_{n,p,f,h}$的取值范围,规范了逻辑功能在芯片中的位置与大小.

$$\forall n\in N,p\in P,f\in F,h\in H:\quad \sum_{f\in F}g_{n,p,f,h}\geq w_{n,p}-(1-z_{n,p,h})\cdot W \tag{12}$$

通过式(13)对 $r_{n_1,p,n_2,p}$进行约束,确保逻辑功能之间布局不存在位置重叠.

$$\forall n_1,n_2\in N,p_1,p_2\in P,h\in H,\mid n_1<n_2:$$
$$x_{n_1,p_1}+w_{n_1,p_1}\leq x_{n_2,p_2}+r_{n_1,p_1,n_2,p_2}\cdot W$$
$$x_{n_1,p_1}\geq x_{n_2,p_2}+w_{n_2,p_2}-(3-r_{n_1,p_1,n_2,p_2}-z_{n_1,p_1,h}-z_{n_2,p_2,h})\cdot W \tag{13}$$

确保每一个逻辑功能分配的各类资源不少于其所需资源数量,定义式(14).

$$\forall n\in N,p\in P,t\in T:\quad \sum_{f\in F,h\in H}\Big(\sum_{p\in P}g_{n,p,f,h}\Big)\cdot d_{f,t}\geq r_{n,t} \tag{14}$$

3.4　目标函数定义

UPRFloor 布局策略从芯片资源占用、布线长度以及任务长宽比三个方面进行多目标优化,各项指标采用线性组合的方式进行关联,目标函数如式(15)所示:

$$\min\left(w_1\cdot\frac{\mathrm{Res}_{\mathrm{cost}}}{\mathrm{Res}_{\max}}+w_2\cdot\frac{C_{\mathrm{cost}}}{C_{\max}}+w_3\cdot\frac{P_{\mathrm{cost}}}{P_{\max}}\right) \tag{15}$$

其中,芯片资源占用情况($\mathrm{Res}_{\mathrm{cost}}$)是布局策略首要考虑的因素. 通过实际占用资源数量与所需资源数量之差来表示,式(16)为具体的计算方法,其中 λ_t 为惩罚项,用于计算浪费一个单位的 t 类资源所需代价,默认设置为 $\lambda_t=1/(d_{f,t}\cdot F)$.

$$\mathrm{Res}_{\mathrm{cost}}=\sum_{n\in N,t\in T}\Big(\sum_{f\in F,h\in H}\Big(\Big(\sum_{p\in P}g_{n,p,f,h}\Big)\cdot d_{f,t}\Big)-r_{n,t}\Big)\cdot\lambda_t \tag{16}$$

布线长度(C_{cost})是影响通信开销与能耗的重要指标. 本文采用半周长连线长度(Half-perimeter Wire Length,HPWL)方法进行计算[13],如式(17)所示,首先求得每一个逻辑功能重心 cx_n,cy_n,然后计算逻辑功能模块 n_1,n_2间曼哈顿距离 $x^{\mathrm{dis}}_{n1,n2}$与 $y^{\mathrm{dis}}_{n1,n2}$,进而计算通信开销 C_{cost},其中 $l_{n1,n2}$为连线宽度.

$$\forall n_1,n_2\in N,p\in P$$
$$cx_n=\max(x_{n,p}+(w_{n,p}-1)/2)$$
$$cy_n=\max((y^{\mathrm{l}}_{n,p}+y^{\mathrm{h}}_{n,p})/2)$$
$$x^{\mathrm{dis}}_{n_1,n_2}=\mid cx_{n_1}-cx_{n_2}\mid$$
$$y^{\mathrm{dis}}_{n_1,n_2}=\mid cy_{n_1}-cy_{n_2}\mid$$
$$C_{\mathrm{cost}}=\sum_{n_1,n_2}(x^{\mathrm{dis}}_{n_1,n_2}+y^{\mathrm{dis}}_{n_1,n_2})\cdot l_{n_1,n_2} \tag{17}$$

周长最短(P_{cost})的引入是为了考虑逻辑功能块内通信开销,对于由两个矩形块(part_0,part_1)堆叠形成的逻辑功能非矩形模型,在面积相同的情况下,非矩形周长越短,块内任一点到达各边的距离越短,从而块内通信距离越短,通信延迟越小,定义如公式(18)所示.

$$P_{\mathrm{cost}}=2\cdot\sum_{n\in N,p\in P}(\max(w_{n,p})+h_{n,p}) \tag{18}$$

式(15)中 $\mathrm{Res}_{\max}$,$C_{\max}$,$P_{\max}$为目标芯片可提供的相关代价函数的最大数值,用于归一化各项代价函数取值,参数 w_1,w_2,w_3 为各项代价函数权值,可以根据实际需要进行赋值.

算法 1 为伪代码实现,其中第 1 行包括各类常量

定义、变量定义与赋初值操作,第 2 ~ 28 行遍历各个待布局逻辑功能,将各项约束写入模型,第 29 行定义目标函数,最后调用求解器对模型进行求优. 由表 1 可知,$P=2$ 为常数,同时对于目前 FPGA 芯片规模来说 F,W,H 通常也很小[8],因此建立布局模型的时间复杂度为 $O(N^2)$. 由于 MILP 求优的时间复杂度为指数阶,因而 UPRFloor 模型的求优时间复杂度和文献[10]均为 $O(2^N)$.

算法 1 UPRFloor 布局过程伪代码

输入:待布局逻辑功能 N、逻辑功能分块数量 P、目标 FPGA 分区 F,宽度 W,高度 H
输出:各个逻辑功能的位置 $x_{n,p}$,$y^1_{n,p}$和大小 $w_{n,p}$,$h_{n,p}$(布局结果)
1. model. init( );//初始化模型信息
2. for each n in N do
3. for each p in P do
4. 根据式(3)对 n 的分块 p 横向位置($x_{n,p}$,$w_{n,p}$)进行约束
5. for each h in H do
6. 根据式(4) ~ (7)对逻辑功能 n 的高度 $h_{n,p}$进行约束
7. end for
8. for each f in F do
9. 根据式(8)、(9)对 n 的分块 p 与芯片分区 f 位置约束
10. for each h in H do
11. 根据式(10) ~ (12)对 n 与 f 分区在 h 行的重叠数量进行约束
12. end for
13. end for
14. end for
15. model. addconstrain( );//将上述各项约束写入模型文件
16. end for
17. for each n_1 in N do
18. for each p_1 in P do
19. for each n_2 in N do
20. for each p_2 in P do
21. if $n_1 < n_2$ then
22. 根据式(13)确保 n_1,n_2之间布局不重叠
23. model. addconstraint( );
24. end if
25. end for
26. end for
27. end for
28. end for
29. model. setObjective( );//目标函数定义
30. return model. optimize( );//模型优化并返回布局结果

4 实验结果与性能分析

本文采用主流数学规划优化器 Gurobi Optimizer[14] 作为优化引擎,使用 Java 语言实现 UPRFloor 布局策略. 具体实验环境配置:CPU 为 Intel(R) Core(TM) i7 - 4790 @ 3. 60GHz,内存为 12GB DDR4.

4.1 不同参数配置对布局性能影响分析

本节以目标函数中各项权重为变量,分析不同参数配置对布局性能的影响. 选取如表 3 所示的 6 组参数配置对 UPRFloor 进行测试. 其中,参数配置 1 为资源占用、布线长度和任务长宽比三项指标平均占比,可作为基准参数与其它参数配置进行对比,参数配置 2 中资源占用指标的比例更高,侧重于节约珍贵的可重构资源,参数配置 3 降低资源指标,更强调逻辑功能块内与块间通信开销,参数配置 4 ~ 6 分别为各项指标的极限占比情况.

表 3 各项指标的参数配置表

Config	w_1	w_2	w_3
1	$0.33\mathrm{Res}_{\max}$	$0.33C_{\max}$	$0.33P_{\max}$
2	$0.60\mathrm{Res}_{\max}$	$0.20C_{\max}$	$0.20P_{\max}$
3	$0.20\mathrm{Res}_{\max}$	$0.40C_{\max}$	$0.40P_{\max}$
4	$\mathrm{Res}_{\max}$	1	1
5	1	$C_{\max}$	1
6	1	1	$P_{\max}$

选择 Virtex-5 系列 XQ5FX70T 芯片作为测试芯片,实验所用测试数据集为仿真数据集:待布局逻辑功能数量规模范围[5,20],抽取 4 组测试数据,每组测试数据所包含的各类资源数量通过随机方式产生,为验证算法准确性,取 10 次运行时间的平均值作为测试结果. 并且,每组测试数据保证至少 3 到 10 个逻辑功能包含 BRAM 资源,至少 1 到 6 个逻辑功能包含 DSP 资源.

由表 4 可得,在每一种逻辑功能数量规模下,采用 UPRFloor 布局策略均比采用矩形模型布局策略占用的可重构资源数量更少,同时耗时均有所增长,这是因为在待布局逻辑功能数量相同的情况下,UPRFloor 方法要处理的数据比传统布局方法多 $N*(P-1)$ 个,因此所需时间增多. 在配置方式 1 中,UPRFloor 布局策略相比矩形模型布局策略算法耗时平均增加 21. 53% 的同时,平均节省了 25. 59% 的可重构资源,. 可见 UPRFloor 布局策略在牺牲一定算法时间复杂度的情况下能够节省更多的可重构资源.

图 5 以参数配置 1 为基准,对比 UPRFloor 布局策略的其它五种参数配置的布局性能,随着 w_1 取值的增大,节省的可重构资源随之增多,同时算法耗时也有所增长,在参数配置 4($w_1=\mathrm{Res}_{\max}$)时达到峰值. 相比参数配置 1($w_1=0.3\mathrm{Res}_{\max}$),UPRFloor 布局策略在参数配置 2($w_1=0.6\mathrm{Res}_{\max}$)时所占用的资源数量平均减少了 10. 21%,同时算法耗时平均增长了 4. 61%,在参数配置 4($w_1=\mathrm{Res}_{\max}$)时减少了 21. 63% 的资源占用,耗时增加 18. 6%.

值得注意的是,在 w_1 值较小的配置方式 3 中,算法耗时有所降低的同时所占用的资源数量与配置 1 相似,甚至优于配置 1.

表 4　UPRFloor 与矩形模型策略的性能对比

Config	Resource Wastage					Time				
	5	10	15	20	Avg	5	10	15	20	Avg
1	-34.03%	-40.67%	-8.31%	-19.34%	**-25.59%**	33.06%	30.08%	6.94%	16.02%	**21.53%**
2	-51.05%	-63.52%	-10.22%	-28.97%	**-38.44%**	25.60%	42.78%	9.22%	41.58%	**29.80%**
3	-42.52%	-33.52%	-9.22%	-18.97%	**-26.06%**	23.15%	36.52%	10.12%	31.80%	**25.40%**
4	-45.85%	-39.38%	-36.98%	-21.14%	**-35.84%**	36.90%	44.30%	29.60%	18.98%	**32.45%**
5	-47.71%	-39.38%	-13.53%	-6.68%	**-26.83%**	32.10%	26.89%	47.46%	10.53%	**29.25%**
6	-46.64%	-37.70%	-3.83%	-1.76%	**-22.48%**	40.29%	47.85%	9.99%	1.63%	**24.94%**

进一步的,与参数配置 1 ~4 中资源利用率增速高于算法耗时增速不同,参数配置 5,6 在绝大多数情况下浪费更多可重构资源的同时,算法耗时未有明显降低,如配置 5 中,相比配置 1,浪费了 6.35% 的可重构资源,算法耗时只减少了 5.52%,配置 6 与配置 1 几乎使用了几乎相同的布局时间,却多占用了 4.35% 的资源.

因此可以得到如下结论:(1)相比矩形布局策略,UPRFloor 布局策略在牺牲一定算法时间复杂度的情况下能够节省更多的可重构资源;(2)随着 w_1 取值的增大,节省的可重构资源随之增多,同时算法耗时也有所增长,在可重构资源稀缺且布局时限宽松的应用场景中,可选择如参数配置 4 的参数方案以便获得最高的资源利用率;(3)当 w_1 的取值小于 w_2 与 w_3 时,该布局策略仍能具有较好的布局性能;(4)因为算法耗时与资源利用率增速不匹配的原因,不建议采用类似配置 5、6 的极限参数配置方式.

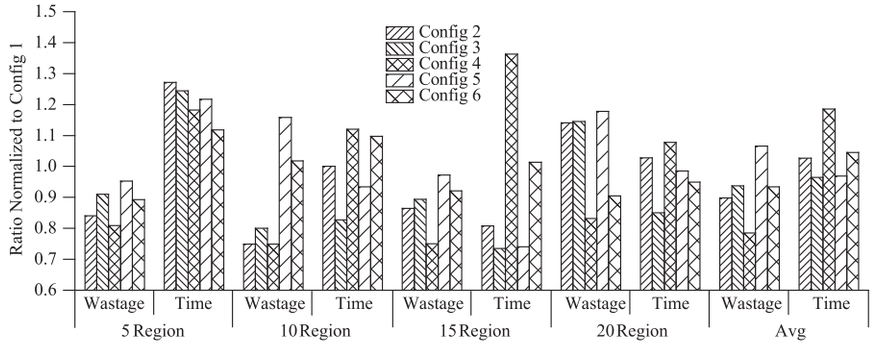


图5　UPRFloor布局策略在不同参数配置下的布局性能对比

4.2　布线长度与布算法耗时分析

本节选取 FPGA 设计领域常用的 Microelectronics Center of North Carolina (MCNC) benchmark 基准电路[13]中的 5 种标准测试数据集对 UPRFloor 布局策略和文献[8]、文献[10]与文献[13]的布局方法在布线长度与算法耗时两个方面进行对比,文献[13]为早期的考虑资源非均匀分布式 FPGA 布局的算法,文献[8]和文献[10]均采用 MILP 方法,待布局功能为矩形模型,其中文献[8]使用冲突图(Conflict Graph)对逻辑功能进行约束,避免位置冲突的同时寻找可行解. 目标芯片选取与文献[8]一致的 Virtex-7 XC7K160T 芯片,参数配置选取平均占比的 $w_1 = 0.33\mathrm{Res}_{\max}$, $w_2 = 0.33C_{\max}$, $w_3 = 0.33P_{\max}$ 方案,文献[8]将布局求解器的运行时限设置为 1800s,所以本文也采用类似设置.

表 5　布线性能分析

Circuit	No. of Regions	Wire length				Execution time(sec)			
		文献[13]	文献[10]	文献[8]	UPRFloor	文献[13]	文献[10]	文献[8]	UPRFloor
apte	9	12789	9682	5206	4035	21.75	1947.93	1801.88	1800.43
xerox	10	26589	22974	12813	11562	25.53	2107.98	1802.37	1804.09
hp	11	12403	11036	5298	4676	29.42	2001.05	1802.45	1801.14
ami33	33	172332	130121	104414	99383	164.61	2124.57	1803.98	1802.72
ami49	49	55819	36930	18583	18835	180.54	1992.26	1809.47	1800.21

对比实验结果如表 5 所示,其中文献[13]耗时最短,布线长度也最长;UPRFloor 方法在前 4 个测试集中布线性能最优,相比文献[8],在算法耗时几乎相同的情况下,UPRFloor 方法的布线长度最多减少了

22.49%,平均减少了 10.58%;随着逻辑功能数量的增多,UPRFloor 的性能优势逐渐降低,值得注意的是,在包含 49 个逻辑功能的 ami49 基准电路中,UPRFloor 耗费了与文献[8]几乎相同的时间,但是并未得到更优的布线性能,虽然在进一步的实验中验证了解除求解器运行时限,在耗时更长的情况下 UPRFloor 的布线长度可以更短,但是分析造成上述问题的原因是因为非矩形布局方式需要多处理 $N*(P-1)$ 个逻辑功能,所以随着数据规模的增大,其性能优势也逐渐降低.

4.3 案例分析

软件定义的无线电(Software Defined Radio,SDR)是一种无线电广播通信技术,它采用基于软件定义的无线通信协议来实现包括工作频段、加密模式、通信协议在内的各项功能,并且可通过软件下载和更新进行升级,而不用更换硬件,因此具有很强的灵活性与开放性. 以 FPGA 为载体实现 SDR 共包括:Matched Filter,Carrier Recovery,Demodulator,Signal Decoder 和 Video Decoder 这五个逻辑功能模块[10],所有模块之间以 64bit 总线相连.

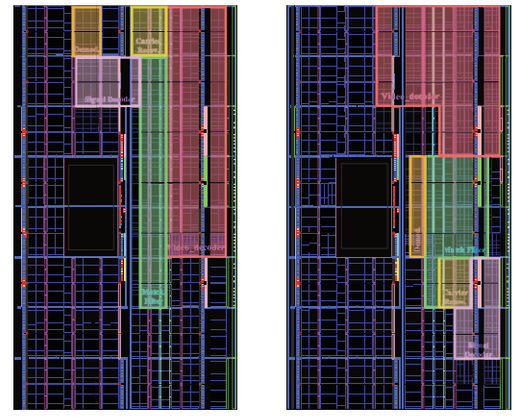
表 8 SDR 所需各项可重构资源数量

Region	CLB tiles	BRAM tiles	DSP tiles	No. of Frames
Matched Filter	25	0	5	1040
Carrier Recovery	7	0	1	280
Demodulator	5	8	0	240
Signal Decoder	12	1	0	462
Video Decoder	55	2	5	2180
Total	104	5	11	4202

本节以 SDR 作为测试数据来源,将 UPRFloor 布局策略与文献[10]的布局结果进行对比,采用与文献[10]相同的 Virtex-5 FX70T 芯片作为目标芯片,该芯片包括 CLB,BRAM,DSP 三种可重构资源,每种可重构资源的一个最小重构单位(tile)中包含 36,30 和 28 个可重构帧(configurable frame),SDR 各模块所需资源数量见表 8.

因为文献[10]将资源浪费作为首要优化条件,然后在不影响主目标的前提下尽量降低布线长度,所以本文采用同样的方式,选取 $w_1=\mathrm{Res}_{\max}, w_2=1, w_3=1$ 的参数配置,按照这样的目标函数进行优化,意味着在找到最少资源浪费的布局中挑选一个布线长度最短的作为最终布局结果.

图 6 为两种策略的布局结果对比图,其中,文献[10]方法共浪费 306 帧(frame)可重构资源,而 UPRFloor 方法共浪费 216 帧可重构资源,资源利用率提升了 29.41%;布线长度方面文献[10]为 2624,UPRFloor 方法为 2272,节省了 13.41%;算法耗时方面,文献[10]的耗时为一个多小时,而 UPRFloor 方法的耗时总共为 6572.62 秒,与文献[10]中的方法耗时大体相当.



(a) 文献 [10] 的布局结果 (b) UPRFloor的布局结果

图6 UPRFloor方法与文献[10]方法的布局结果对比

5 结论

基于 FPGA 的动态可重构技术在许多领域应用越来越广泛,如何提高 FPGA 可重构资源利用和降低电路通信开销受到了越来越多的关注. 本文针对 FPGA 布局优化问题,综合考虑待布局逻辑功能客观形状,建立多描述模型,并设计实现了一种以减少芯片可重构资源浪费与降低逻辑功能的各项通信开销为目的的多目标优化布局策略 UPRFloor,并将其在三款不同系列的 FPGA 芯片上进行仿真实验. 结果表明:使用 UPRFloor 布局策略,相比基于矩形模型的布局方法在资源利用率方面最高有 25.59% 的提升;在 MCNC 标准测试集中进行了验证测试,在算法耗时几乎相同的情况下,较其它算法的布线长度最多减少了 22.49%;以 FPGA 实现 SDR 作为测试数据,UPRFloor 在节约 29.41% 可重构资源的同时,布线长度节省了 13.41% 从而有效降低了资源浪费与通信开销.

参考文献

[1] Xilinx Inc. Vivado Design Suite User Guide:Partial Reconfiguration[DB/OL]. https://china.xilinx.com/support/documentation/sw_manuals/xilinx2016_4/ug909-vivado-partial-reconfiguration.pdf,2016-11-30.

[2] Cong J,Huang H,Ghodrat M A. AScalable communication-aware compilation flow for programmable accelerators [A]. Proc of the 21st Asia and South Pacific Design Automation Conference[C]. Piscataway,NJ:IEEE,2016. 503-510.

[3] Cong J,Huang M,Wu D,et al. Invited-heterogeneous datacenters:options and opportunities[A]. Proc of the 53rd Annual Design Automation Conference,ser. DAC'16[C].

New York:ACM,2016. 1 – 6.
[4] Zhang C,Li P,Sun G,et al. Optimizing FPGA-basedaccelerator design for deep convolutional neural networks[A]. Proc of the 23th ACM/SIGDA International Symposium on Field Programmable Gate Arrays[C]. New York:ACM,2015. 161 – 170.
[5] Nguyen T D A,Kumar A. PRFloor:An automatic floorplanner for partially reconfigurable FPGA systems[A]. Proc of the 24th ACM/SIGDA International Symposium on Field Programmable Gate Arrays[C]. New York:ACM,2016. 149 – 158.
[6] Luu J,Goeders J,Wainberg M,et al. VTR 7.0:next generation architecture and CAD system for FPGAs[J]. ACM Transactions on Reconfigurable Technology & Systems,2014,7(2):1 – 30.
[7] University of Toronto. Verilog-to-Routing Documentation[DB/OL]. https://docs. verilogtorouting. org/en/latest/,2017-12-20.
[8] Rabozzi M,Durelli G C,Miele A,et al. Floorplanningautomation for partial-reconfigurable FPGAs via feasible placements generation[J]. IEEE Transactions on Very Large Scale Integration Systems,2017,25(1):151 – 164.
[9] Ma Y,Liu J,Zhang C,et al. HW/SWpartitioning for region-based dynamic partial reconfigurable FPGAs[A]. Proc of the 32th International Conference on Computer Design[C]. Piscataway,NJ:IEEE,2014. 470 – 476.
[10] Rabozzi M,Lillis J,Santambrogio M D. Floorplanning for partially-reconfigurable FPGA systems via mixed integer linear programming[A]. Proc of the 22th Annual International Symposium on Field-Programmable Custom Computing Machines[C]. Piscataway,NJ:IEEE,2014. 186 – 193.
[11] Wang C,Wu W,Nie S,et al. BFT:a placement algorithm for non-rectangle task model in reconfigurable computing system[J]. IET Computers & Digital Techniques,2016,10(3):128 – 137.
[12] Vipin K,Fahmy S A. Architecture-aware reconfiguration-centric floorplanning for partial reconfiguration[A]. 8th International Symposium on Applied Reconfigurable Computing[C]. Berlin:Springer-Verlag,2012. 13 – 25.
[13] Bolchini C,Miele A,Sandionigi C. Automated resource-aware floorplanning of reconfigurable areas in partially reconfigurable FPGA systems[A]. Proc of 21th International Conference on Field Programmable Logic and Applications[C]. Piscataway,NJ:IEEE,2011. 532 – 538.
[14] Gurobi Optimization,Inc. Gurobi Optimizer Reference Manual[DB/OL]. http://www. gurobi. com/documentation/7. 0/refman/index. html,2017-02-21.

作者简介

王今雨　男,1989 年生于陕西富平,现为西安交通大学计算机系博士生. 主要研究方向为可重构计算.
E-mail:wjyokok@ gmail. com

伍卫国　男,1963 年生于江西,现为西安交通大学高性能计算机研究所博士生导师. 主要研究方向为高性能计算机体系结构,云计算与嵌入式系统.
E-mail:wgwu@ mail. xjtu. edu. cn

秦朝楠　男,1990 年生于陕西渭南,现为西安交通大学计算机系硕士研究生. 主要研究方向为云计算与大数据.
E-mail:413208357@ qq. com

赵东方　男,1995 年生于云南昆明,现为西安交通大学计算机系硕士研究生. 主要研究方向为 FPGA 性能优化.
E-mail:476337455@ qq. com