文章编号:1671-4598(2023)08-0205-06 DOI:10.16526/j.cnki.11—4762/tp.2023.08.031 中图分类号:TP301 文献标识码:A

FPGA 中基于空间连续性的碎片度量及任务放置

饶 广[1],饶云波[2]
(1. 内江职业技术学院,四川 内江 641000;2. 电子科技大学 信息与软件工程学院,成都 610054)

摘要:针对部分可重构现场可编程门阵列允许在运行时对芯片的各个部分进行配置导致的区域碎片,提出了一种新的基于被占用(或空闲)空间的连续性的碎片度量及在线任务放置方法;首先从一维结构出发,得到一个单元序列对一个单元流 S 的碎片度量 F_S 的贡献值,进而得到一维碎片度量值,它不依赖于到达任务的大小;然后将一维结构得到的碎片度量值结果推广到二维及高维结构;最后在 FPGA 上的在线任务放置过程中采用这种碎片度量方法,从而减少芯片碎片;在二维结构的 FPGA 上的仿真实验结果表明,与通常采用的左下角、第一匹配和最佳匹配放置策略相比,采用提出的碎片度量及放置方法不仅在等待时间、分配时间和响应时间方面有所改善,而且提高了芯片的利用率,降低了失配率。

关键词:FPGA;部分可重构;区域碎片;在线任务放置;时间;芯片利用率;失配率

Fragmentation Metric and Task Placement Based on Spatial Continuity in FPGA

RAO Guang[1], RAO Yunbo[2]
(1. Neijiang Vocational & Technical College, Neijiang 641000, China; 2. School of Information and Software Engineering, University of Electronic Science and Technology of China, Chengdu 610054, China)

Abstract: Aimed at area fragment resulted in partially reconfigurable field-programmable gate arrays (FPGA), each part of the chip is required to be configured at run-time, a new fragment metric and online task placement method based on the continuity of occupied or free space is proposed. Firstly, from the one-dimensional structure, the contribution of one cell sequence to the fragment metric F_S of one cell stream S is obtained, and then the one-dimensional fragment metric is obtained, it is independent of the size of the incoming tasks. Then, the obtained result of fragment metric in the one-dimensional structure is extended to two-dimensional and high-dimensional structures. Finally, this fragment metric method is used to reduce chip fragments during the online task placement on FPGA. The simulation results on the two-dimensional FPGA show that, compared with the usual the bottom left, first fit and best fit placement strategies, the proposed fragment metric and placement method not only results in improvement in terms of the waiting time, allocation time and response time, but also increases the chip utilization and reduces the miss ratio.

Keywords: FPGA; Partially reconfigurable; area fragment; online task placement; time; chip utilization; miss ratio

0 引言

现场可编程门阵列(FPGA, field-programmable gate array)通常是部分可重构的,允许在运行时对芯片的各个部分进行配置,其中每部分可以执行一个独立的任务[1-2]。具有这种特性的设备可以同时容纳多个应用,其中每个应用可以由多个任务构成。要充分利用部分可重构结构的优势,就必须充分利用芯片资源[3-6]。不合理放置到达任务会导致芯片区域的一些部分浪费,因为这些部分尽管是空闲的,但它们太小而无法容纳另一个到达的任务。通常把这些空闲的较小部分称为碎片,类似于传统存储系统中的碎片,这些碎片可能占芯片区域的很大比例[7]。因此,区域碎片是获得良好的芯片资源利用的最大障碍之一。一方面,高区域碎片化表示芯片上占用部分的分散,因此可能导致低的芯片利用率;另一方面,低区域碎片化使得到达任务能够被放置,从而提高芯片利用率。

在一个在线放置系统中,由于矩形任务的动态添加和删除,使得 FPGA 的空区域变得高度碎片化,因而无法有效地利用 FPGA 区域。一个高度碎片化的芯片是指其中包含了大量的由一些被占用单元所分割形成的孔洞[8],即任务矩形边界以外的区域资源的碎片。如果一个任务的矩形区域没有可用的足够的相邻资源,则该任务可能被拒绝放置。图 1 所示为这样的一个例子,其中任务 T_1 和 T_2 导致 FPGA 区域碎片化,因此,即使 FPGA 上的总的空闲区域大于任务 T_3 的区域,但任务 T_3 也不能放置在 FPGA 上。

目前,有 2 个主要的方向来提高芯片的区域利用。一是允许任务抢占,即停止活跃任务,保存它们或将它们移动到其他部分,并重新配置新的部分。文献[9]提出了一种任务压缩的启发式算法,这是一种一维保序算法。如果无法在芯片区域上分配(放置)到达任务,则该技术尝试

收稿日期:2022-11-27; **修回日期**:2022-12-23。
基金项目:内江市科技孵化和成果转化专项资金项目(2016-3)。
作者简介:饶 广(1981-),男,四川资中人,硕士,副教授,主要从事计算机应用方向的研究。
饶云波(1978-),男,四川泸州人,博士,副教授,主要从事图像/视频处理、三维重建、虚拟现实、人工智能方向的研究。
引用格式:饶 广,饶云波.FPGA 中基于空间连续性的碎片度量及任务放置[J].计算机测量与控制,2023,31(8):205-210.

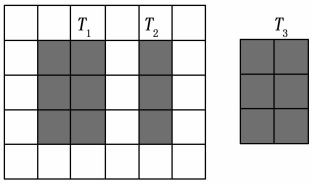

图 1 FPGA 区域的碎片化

将运行的任务压缩到特定的方向，从而能够得到更好的芯片利用率；文献［10］针对碎片资源干扰下云计算资源的优化调度，提出了采用基于膜计算和蝙蝠算法的方法，对云计算环境下不同的资源调度策略形成的资源碎片进行量化处理，重新定量划分，使其可重组成整块资源，最大化接收后续任务。仿真结果表明，所提出的方法有效提高了资源调度能力；二是采用碎片感知放置技术，在这种技术中，仔细地进行放置，以减少区域碎片，为将来的任务分配提供更多的空间。这种技术需要一个可靠的碎片测量来决定把新的任务放在哪里。所有空位置都必须经过测试，才能选择一个能降低碎片的地方。因此，这种放置策略在选择位置时考虑到了碎片度量；文献［11］采用可重构区域中空矩形的较短边的方格（称为基准尺寸）作为碎片度量。他们计算基准尺寸的分布，并将这种分布的均值作为碎片度量。显然，不同的分布（碎片状态）可能有相同的均值，换句话说，分布的平均值并不能唯一地识别芯片的碎片数量；文献［12］针对 FPGA 可重构设计过程中会产生大量空闲碎片的问题，提出了一种碎片合并算法，测试结果表明，对于 1 000 个连续产生的随机任务进行碎片合并之后，所有任务一次申请资源成功率达到 87.4%，等待一个任务结束之后资源申请成功率达到 95.4%；文献［13］采用碎片等级来对碎片进行量化，在 FPGA 上寻找覆盖空区域所需的所有空矩形。如果 a_i 为第 i 个空矩形的面积，则碎片等级计算为 $F=1-\sqrt{\sum a_i^2}/\sum a_i$。计算式中所使用的空矩形可能重叠，所以空区域可能不止一次参与碎片等级的计算，这样就会导致结果不准确。图 2（a）所示为使用同一区域 2 次的碎片等级；文献［14］采用碎片度量将整个芯片上的空洞的分散程度量化为芯片碎片，即 $f_i=a_i/A$，$F=1-\prod_i f_i$，其中 a_i 为第 i 个空矩形的面积，A 为芯片面积。他们在 FPGA 上寻找覆盖空区域所需的最大空矩形，并计算了碎片因子。这里仍存在着空域重叠的问题，而且两种不同的碎片状态可能得到相同的碎片因子，如图 2（b）所示。

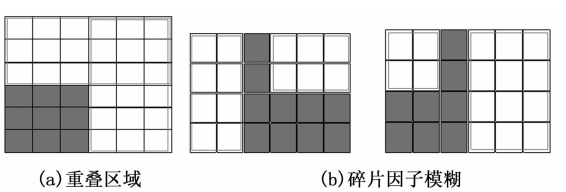

图 2 碎片计算示例

上述这些方法都给出的是绝对的碎片度量，仅考虑了可重构单元的分布。

另一种发展趋势是相对碎片度量，它基于到达任务的大小考虑可重构单元的状态。作为相对碎片度量的一个例子，文献［15］根据任务的平均大小来进行计算。他们在一个空单元的垂直和水平附近使用空单元数量，如果这个数量大于或等于到达任务平均大小的 2 倍，则该单元的碎片贡献为 0。总的碎片由可重构芯片区域内所有空单元的垂直碎片和水平碎片之和来计算；文献［16］基于多 FPGA 的动态可重配置的多任务调度和放置方法，通过任务级和子任务级的两阶段调度和放置方法，实现了在多 FPGA 系统的多任务调度和放置。在任务调度阶段，考虑任务相似性和资源需求相似性，为每个任务选择合适的计算单元，以减少重新配置和资源争用的可能性，在子任务调度阶段，综合考虑子任务的调度顺序和放置位置，以充分利用 FPGA 的硬件资源，利用 FPGA 可重配置能力使任务高度并行化，从而减小多任务的最小完工时间；文献［17］提出了一种 HRCA 系统的可重构单元的二维任务放置方法。方法结合 3 种影响 HRCA 系统的可重构单元的碎片产生因素，包括当前任务与其邻接任务在时间上的重合度，当前任务与其邻接任务的边长的重合度以及当前任务对其它空闲块的影响程度，依次计算当前任务的长、宽分别沿每个空闲块的每两条相邻边放置时的合适度，选出所有空闲块的所有位置中合适度最大的位置作为当前任务的最终放置位置。提出的可重构单元的二维任务放置方法，可以使任务放置更为紧凑合理，减少可重构单元中的碎片，提高可重构单元的空间利用率；文献［18］针对最佳匹配（BF，best fit）算法存在的缺陷，在目标处理器结点的局部调度下和避免最大入侵原则下，提出了一种避免入侵最佳匹配算法 AIBFA。AIBFA 算法分别降低了全局系统调度的平均时间负载率和目标处理器结点局部调度的任务拒绝率。针对寻找空闲资源全集的问题提出了一种基于单向栈的算法来寻找最大空闲矩形，利用可重构计算单元的不同 M 值进出单向栈来搜索到所有最大空闲矩形。实验表明，算法通过使用单向栈与算法优化，有效提高了查找空闲资源全集时的性能，减少了 FPGA 资源的碎片率；文献［19］针对目前可重构系统任务在线调度方法的不足，提出了一种基于放置代价的可重构系统软/硬件任务统一调度方法。方法考虑了 3 种代价，分别为硬件任务在 FPGA 上的执行时间、占用的 FPGA 面积以及 FPGA 的碎片情况，还考虑了软/硬件任务的统一调度方法。在调度过程中，当硬件任务的代价超过设定的阈值时，就拒绝其在 FPGA 上运行，并由 CPU 执行其相应软件任务实现. 通过合理地拒绝一些代价较大的任务，从而从整体上提高任务调度成功率。实验表明，提出的方法能够获得更高的任务截止保证率。

相对碎片度量对特定时间到达任务的大小不敏感，所以对于相同的芯片碎片状态，可能在不同时间得到 2 种不同的碎片度量，而到达任务的大小也可能是高度变化的。因此，绝对度量更准确，它不需要给出芯片是满的或空的任何指示，相反，仅给出整个芯片区域的孔（或空区域）

的分散程度。

对此，本文提出了一种测量区域碎片的新方法和利用这种碎片度量的在线任务放置。提出的碎片度量方法考虑的是可重构芯片上被占用（或空闲）空间的连续性，而不是被占用（或空闲）空间的数量。这样，可用于监测芯片区域和选择最佳的空区域来放置新任务，从而减少总的芯片区域碎片。基于一个二维结构的 FPGA 的仿真实验结果表明，相比于一般的左下角、第一匹配和最佳匹配放置策略，本文提出的碎片度量及放置方法不仅在等待时间、分配时间和响应时间方面有所改善，而且提高了芯片的利用率，降低了失配率。

1　系统模型

这部分给出本文所采用的部分可重构系统的任务和系统模型。

一个 $H\times W$ 的部分可重构 FPGA 芯片由 H 行和 W 列构成，左下角是芯片原点，在不影响芯片其余部分的情况下对芯片的一部分进行配置。由于单元按顺序配置，故 FPGA 是部分可重构的，重构时间与重新配置的单元数量成正比，配置延迟是配置一个单元及其相关路由资源所需要的时间。

系统假设任务在线到达，并按到达顺序排队和放置。只要在 FPGA 区域中有可用的空闲单元，服务器就会继续为到达的任务提供服务，方法是将每个任务放置在 FPGA 芯片的一个未被占用的区域上。如果由于没有可用的相邻资源（空闲单元），队列顶部的任务被拒绝放置，则放置队列将处于停顿状态，直至存在足够的相邻空空间。

任务是非抢占的[20]。一旦任务被放置在可重构芯片区域的任何空闲区域，就停留在这个地方直至完成执行；任务参数事先是未知的，这些任务参数定义为：对于一个任务 t_i，$t_i=(h_i, w_i, a_i, s_i, d_i, x_i, y_i)$，$h_i$ 和 w_i 分别代表其高度和宽度，是用单元数量来度量的，a_i、s_i 和 d_i 分别为任务到达时间、任务服务时间和任务截止时间。分配给任务的矩形区域由其左下角 (x_i, y_i) 给出，其中 x_i 表示行号，y_i 表示列号。这些特征能够反映一个通用计算系统。任务的大小、到达时间、服务时间和截止时间在一个预定义的区域都是均匀分布的，且事先未知的。

从形式上说，任务 t_i 在时刻 a_i 到达，在时刻 s_i 开始执行，在时刻 f_i 完成执行，因此，任务 t_i 的等待时间为 $w(t_i)=s_i-a_i$ 给出，响应时间为 $r(t_i)=f_i-a_i$。任务的分配时间 $al(t_i)$ 是任务在等待队列的顶部等待直到它在可重构区域中找到一个位置的时间。

2　碎片度量

本节提出一种新的多维结构的碎片度量方法。假设单元是区域资源的最小单位，已使用的单元是指已占用的单元，未使用的单元称为空单元。我们从一维结构的度量开始，然后将它扩展到二维结构和更高维的结构。

2.1　一维碎片度量

首先给出一些定义。

考虑一个一维结构 S，由 L 个单元构成，每个单元可以是占用的或空闲的两种状态之一，称这种结构为单元流。

定义 1：长度为 L 的单元流 S 是一组 L 个连续单元。

备注：一个单元流表示一组连续的单元，而不考虑每个单元的状态。

定义 2：长度为 K 的单元序列 Q 是一组 K 个连续的空单元。

在图 3（a）中，存在长度分别为 2、3 的两个单元序列，而在图 3（b）中，存在长度分别为 1、4、5 的 3 个单元序列。令 $c(S)$ 为单元流 S 中单元序列的数量，$Q_S(x)$ 为单元流 S 中第 x 个单元序列，$l(Q_S(x))$ 为其长度（$0\leqslant x\leqslant c(S)$）。

引理 1：在长度为 L 的单元流 S 中，单元序列数量 $c\leqslant S\lceil L/2\rceil$。

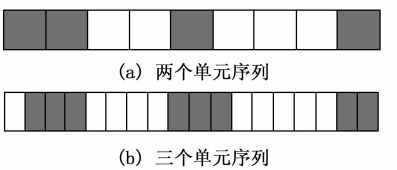

图 3　一维碎片度量说明

证明：当每个单元流的长度为 1 时，就会出现最大的单元流数量，即当单元序列中有一个空单元，然后是一个已占用的单元时，等等，就会发生这种情况。空单元的数量表示单元序列的数量，对于一个长度为 L 的单元流来说，它为 $L/2$。

令 $Q'_S=\{Q_S(x): 0\leqslant x\leqslant c(S)\}$ 表示长度为 L 的单元流 S 中的单元序列的集合，显然，单元序列 $|Q'_S|$ 的最大数量$\leqslant\lceil L/2\rceil$。

我们的度量指标测量在单元流内被占用区域的连续性，而不是测量有多少个单元是空的。因此，并不与具体的问题相关联，而是一个通用的度量指标，可以在不同的情况下使用。测量单元流中被占用单元的连续性，可转化为测量在单元流中有多少单元序列。具有小长度的许多单元序列的存在是高度碎片化的一个标志，我们建立在这一观察的基础上。

令 F_S 表示单元流 S 的碎片度量，随着单元序列数量的增加，S 的碎片也增加。每个单元序列的长度是计算 S 的碎片化的另一个重要因素，随着每个单元序列长度的减小，单元序列对碎片度量的贡献就更大。一个单元序列对 F_S 贡献的值为 $1/l(Q_S(x))$，则一维碎片度量为：

$$F_{1-d}=\sum_{x=0}^{|Q'_S|-1}\frac{1}{l(Q_S(x))} \tag{1}$$

在图 3（a）和（b）中，分别有 $F_{1-d}=1/2+1/3=5/6$，$F_{1-d}=1/1+1/4+1/5=29/20$。

2.2　二维碎片度量

将上一节得到的一维碎片度量扩展到二维结构。考虑一个二维结构（阵列）A，由 $H\times W$ 个单元构成，每个单

元可以是被占用的或空闲的两种状态之一。

令 $Q'_{R(i)}=\{Q_{R(i)}(x):0\leqslant x\leqslant c(R(i))\}$ 为第 i 行中的单元序列的集合，$Q'_{C(j)}=\{Q_{C(j)}(y):0\leqslant y\leqslant c(C(j))\}$ 为第 j 列中的单元序列的集合，其中 $0\leqslant i\leqslant H-1$，$0\leqslant j\leqslant W-1$。显然，单元序列 $|Q'_{R(i)}|$ 的最大数量≤ $[W/2]$，$0\leqslant i\leqslant H-1$，同样，单元序列 $|Q'_{C(j)}|$ 的最大数量≤ $[H/2]$，$0\leqslant j\leqslant W-1$，每行和每列都是一个一维结构，可以应用 3.1 的结果。每行（列）都对区域碎片度量有贡献。同样，这里测量阵列内被占用区域的连续性，而不测量有多少区域是空的。

令 $F_{R(i)}$ 和 $F_{C(j)}$ 分别表示第 i 行和第 j 列对芯片的碎片度量 F_{2-d} 的贡献，单元序列 $Q_{R(i)}$ 对 $F_{R(i)}$ 贡献的值为 $1/l(Q_{R(i)}(x))$，则：

$$F_{R(i)}=\sum_{x=0}^{|Q'_{R(i)}|-1}1/l(Q_{R(i)}(x)) \tag{2}$$

在 FPGA 环境下，每行 i 对总的区域碎片度量贡献的值为 $F_{R(i)}$。令 F_R 表示芯片中全部行的贡献值，则：

$$F_R=\sum_{i=0}^{H-1}F_{R(i)} \tag{3}$$

同样，每列对总碎片度量的贡献值为：

$$F_{C(j)}=\sum_{y=0}^{|Q'_{C(j)}|-1}1/l(Q_{C(j)}(y)) \tag{4}$$

令 F_C 表示芯片中全部列的贡献值，则：

$$F_C=\sum_{j=0}^{W-1}F_{C(j)} \tag{5}$$

最后，二维碎片度量 F_{2-d} 可以计算为 $F_{2-d}=F_R+F_C$，即：

$$F_{2-d}=\sum_{i=0}^{H-1}\sum_{x=0}^{|Q'_{R(i)}|-1}\frac{1}{l(Q_{R(i)}(x))}+\sum_{j=0}^{W-1}\sum_{y=0}^{|Q'_{C(j)}|-1}\frac{1}{l(Q_{C(j)}(y))} \tag{6}$$

2.3 高维碎片度量

现在将前面的结果推广到 n 维结构，如超立方体。考虑一个 n 维单元阵列 $m_1\times m_2\times\ldots\times m_n$，令 d_1，d_2，…，d_n 表示 n 维。这种结构中的基本单元是基本逻辑块（单元）。每个单元要么被占用，要么是空的。我们将考虑每个维度，并计算这个维度对碎片度量的贡献，将 2.2 节中的做法扩展到 n 维。可以通过确定单元所在的每个维度中的值来定义阵列中的单元。例如，单元 $C(x_1,x_2,\ldots,x_n)$ 是位于 d_1 维的 x_1，d_2 维的 x_2 等等的一个单元，其中 $0\leqslant x_i\leqslant m_i-1$，$1\leqslant i\leqslant n$。令 F_{d_1}，F_{d_2}，…，F_{d_n} 分别表示 d_1，d_2，…，d_n 维对碎片度量的贡献，令 $Q_{d_r(x_1,x_2,\ldots,x_{r-1},x_{r+1},\ldots,x_n)}(k)$ 表示维 d_r 中第 k 个单元序列，$1\leqslant r\leqslant n$，且沿维 d_1，d_2，…，d_{r-1}，d_{r+1}，…，d_n 的值分别为 x_1，x_2，…，x_{r-1}，x_{r+1}，…，x_n，令

$$Q'_{d_r(x_1,x_2,\ldots,x_{r-1},x_{r+1},\ldots,x_n)}(k)=\{Q_{d_r(x_1,x_2,\ldots,x_{r-1},x_{r+1},\ldots,x_n)}(k):0\leqslant k\leqslant c(d_r)\} \tag{7}$$

为在维 d_r 中单元序列的集合，同时令 $l(Q_{d_r(x_1,x_2,\ldots,x_{r-1},x_{r+1},\ldots,x_n)}(k))$ 为该单元序列的长度，把 4.2 节的结果进行扩展（这里 $1\leqslant r\leqslant n$），得到：

$$F_{d_r}=\sum_{x_1=0}^{m_1-1}\sum_{x_2=0}^{m_2-1}\sum_{x_{r-1}=0}^{m_{r-1}-1}\sum_{x_{r+1}=0}^{m_{r+1}-1}\cdots\sum_{x_n=0}^{m_n-1}\sum_{k=0}^{|Q'_{d_r}(k)|-1}\frac{1}{l(Q_{d_r(x_1,x_2,\ldots,x_{r-1},x_{r+1},\ldots,x_n)}(k))} \tag{8}$$

总的碎片度量是每维贡献的总和：

$$F_{n-d}=\sum_{r=1}^{n}F_{d_r} \tag{9}$$

当 $n=3$ 时，这个度量的一个直接应用是测量超立方体中的活跃节点有多分散。

2.4 碎片度量计算及实例

本节提出一种运行时间高效的算法来计算在给定时间的碎片度量。我们用一维阵列 R（或 C）的形式来保存行（或列）的碎片信息，碎片信息的更新是在每次添加和删除任务到 FPGA 之后。最初，当 FPGA 为空时，R 和 C 中每个元素的值为零。

图 4 所示为有一些正在运行任务的一个芯片，代表芯片区域的阵列 R 和 C 在图中是沿列和行显示的，R（或 C）的每个元素保存每行（或列）中相邻空单元的数量。元素 $R(i)$（或 $C(j)$）包含行 i（或列 j）的碎片信息。R（或 C）中的每个元素可能包含多个值，因为一行（或一列）可能有多个空单元序列。图 4 给出了一个 6 行 6 列的 FPGA 设备的示例。在图 4 中，$R(0)=4$ 表示第一行中相邻空单元的数量。C 的第一个元素 $C(0)=[2,1]$ 分别表示长度为 2 和 1 的两个不同的空序列。

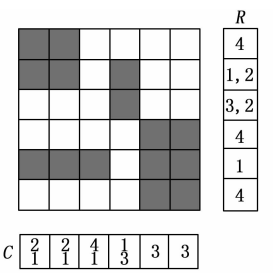

图 4 计算碎片度量

芯片的碎片度量 $F_{2-d}=F_R+F_C$，其中，$F_R=1/4+1/1+1/2+1/3+1/2+1/4+1/1+1/4$，$F_C=1/2+1/1+1/2+1/1+1/4+1/1+1/1+1/3+1/3+1/3$。

3 基于碎片度量的任务在线放置

3.1 任务的添加和删除

在添加或删除任务之后，可以非常高效地更新碎片阵列 R 和 C。当一个任务被添加或删除时，它只影响 R 和 C 的相应元素。如果将任务添加到芯片或从芯片中删除，已占用的行 r_i 到 r_j 和列 c_i 到 c_j，则元素 $R(r_i)$ 到 $R(r_j)$ 将相应改变，以反映这些行的新的状态。对于元素 $C(c_i)$ 到 $C(c_j)$ 也是同样的分析。

3.2 任务的在线放置

根据其前面的讨论，碎片度量测量的是被占用（空闲）区域的连续性，也就是说，具有较低碎片度量值的芯片状态表示空闲空间的更多连续性，我们据此来选择到达任务的位置。

一旦一个任务位于队列的顶部，放置引擎就检查该任务的全部可能位置。放置引擎选择能够得到新状态较低碎

片度量的位置，这将得到更多连续的空间，将有助于为下一组到达任务腾出空间。

碎片度量采用前述的阵列 R 和 C 快速计算，主要是找到最佳候选位置来放置任务，以减少新状态的碎片度量。可以根据任务大小、任务间到达时间和任务执行时间的不同分布，通过多次运行来观察这些参数对系统性能的影响。

4　实验结果

本节在二维结构上，通过对一个综合任务集的一系列实验来测试算法的性能。对于每个实验，生成 1 000 个任务的集合，任务由 4 个独立选择的均匀分布随机变量来描述，其中 2 个变量表示随机选择（均匀分布）的到达任务的长度和宽度，允许从 1～32 个单元不等；一个变量表示任务的服务时间，在 1～1 000 个时间单位之间随机生成（均匀分布）；任务的终止时间是通过在任务的服务时间中添加一个随机变量（在 1～50 之间均匀分布）来形成的。对均匀分布在 1～（10，20，30，40，50，60，70，80，90，100）时间单位之间不同的任务间到达时间间隔重复实验；对这些任务进行排队，并按到达顺序放置到一个大小为 64（64 的仿真 FPGA 上。加载一个任务所需要的时间由芯片上空位置的可用性以及用于配置任务所需单元的时间确定，因此，每个单元的配置延迟也是一个参数，这里选择固定值 1/1 000 时间单位。

首先，在不采用实时截止时间测试的情况下，对于相同的数据集，将本文提出的碎片度量和放置方法与传统的放置策略左下角（BL，bottom-left）、第一匹配（FF，first first）和最佳匹配（BF，best fit）放置方法的性能进行比较，其次，在实时环境下对失配率进行测试。

图 5 和图 6 比较了传统的放置策略 BL、FF 和 BF 与本文提出的碎片度量及任务放置方法的不同性能。采用的一组输入数据任务侧可达 32 个单元，服务时间最大为 500 个时间单位，得到的不同任务间到达时间对各种性能的影响。

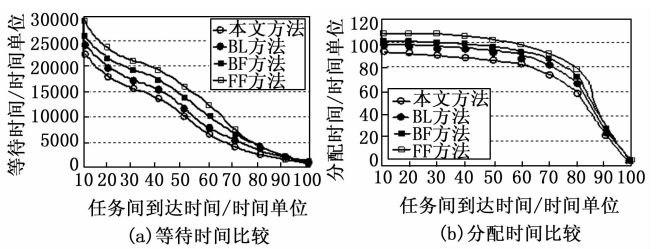

图 5　性能测试 1

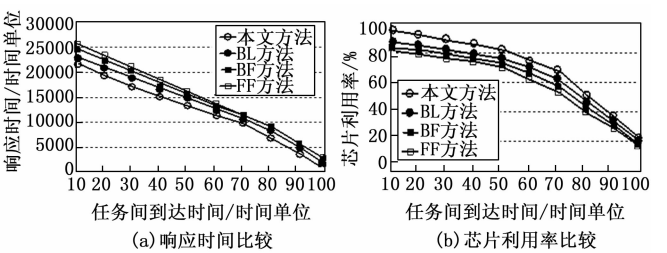

图 6　性能测试 2

图 5（a）所示为不同放置放法的等待时间的比较。可见，本文提出的碎片度量及任务放置放法比 BL 平均降低了约 10%，比 FF 平均降低了约 25%，比 BF 平均降低了约 13%；图 5（b）所示为不同放置放法的分配时间的比较。可见，本文提出的碎片度量及任务放置方法比 BL 平均减少了约 5%，比 FF 平均减少了约 9%，比 BF 平均减少了约 6%；对于响应时间，从图 6（a）可以看到，本文提出的碎片度量及任务放置方法比 BL 平均降低了约 10%，比 FF 平均降低了约 16%，比 BF 平均降低了约 12%；对于芯片利用率，从图 6（b）可以看到，本文提出的碎片度量及任务放置方法比 BL 提高了约 5%，比 FF 提高了约 17%，比 BF 提高了约 13%。

表 1 所示在实时环境下，本文提出的碎片度量及任务放置方法与 BL、FF 和 BF 相比在失配率方面的改进。第一列的任务大小从 1×1 到 32×32、第二列从 8×8 到 32×32，等等。从表 1 可见，在第一列的任务大小中，任务间到达时间为 90 个时间单位时有最佳的改进。随着最小任务边长的增加，失配率有更大的改进，这是因为小任务可以放在芯片中的一些碎片中；当最小任务侧为 24 个单元时，本文提出的碎片度量及任务放置方法的失配率改进效果较好。

表 1　本文方法相比于 BL、FF 和 BF 的失配率改进　%

任务间到达时间	任务大小			
	1～32	8～32	16～32	24～32
	失配率改进	失配率改进	失配率改进	失配率改进
	BL FF BF	BL FF BF	BL FF BF	BL FF BF
10	0.2 0.4 0.6	0.8 0.9 0.9	1.1 1.3 1.4	1.5 1.7 1.9
20	0.2 0.3 0.5	0.9 0.9 1.0	1.2 1.3 1.5	1.7 1.9 2.0
30	0.5 0.6 0.8	1.2 1.9 2.0	1.8 2.3 2.5	2.7 2.9 2.8
40	1.2 1.3 1.5	0.9 1.9 1.8	1.6 1.8 2.7	2.1 3.0 3.1
50	0.9 1.5 1.8	1.9 2.1 2.4	2.2 1.3 2.5	2.7 2.9 3.0
60	1.9 2.1 2.3	2.5 2.9 3.1	3.2 3.3 4.5	3.7 3.9 4.8
70	2.3 2.9 3.5	2.9 3.0 3.6	3.4 3.3 3.5	3.9 4.1 4.4
80	2.6 2.6 3.8	3.3 3.9 4.0	4.2 4.3 4.7	4.7 4.9 4.0
90	2.9 3.3 3.5	3.9 4.9 5.0	5.2 6.1 7.5	6.7 5.9 7.9
100	3.2 4.3 4.5	3.9 4.8 6.0	6.2 6.3 6.5	7.7 7.9 9.1

5　结束语

本文针对部分可重构 FPGA 提出了一种多维碎片度量及任务的在线放置，并在二维结构中采用这个度量来提高 FPGA 的区域利用率。实验结果表明，本文提出的碎片度量及任务在线放置方法与常用的放置策略如左下角、第一匹配和最佳匹配相比较，在芯片利用率方面提高了约 5%～17%，在响应时间方面降低了约 10%～16%；在实时系统中的测试表明，失配率最大可改进约 9%。

将本文提出的放置方法应用于异构可重构系统将是我们未来一段时间的研究方向。

参考文献：

[1] HUANG J，XU X，YAO L，et al. Reconfigurable topology synthesis for application-specific noc on partially dynamically reconfigurable FPGAs [C] //Proceedings of 2017 ACM/IEEE International Workshop on System Level Interconnect Prediction (SLIP)，Austin，TX，USA，2017：1－8.

[2] LIU N，CHEN S，YOSHIMURA T. Resource－aware multi－layer floorplanning for partially reconfigurable FPGAs [J]. IEICE Transactions on Electronics，2013，96 (4)：501－510.

[3] MARIANI G，SIMA V M，PALERMO G，et al. Run－time optimization of a dynamically reconfigurable embedded system through performance prediction [C] //Proceedings of 2013 23rd International Conference on Field Programmable Logic and Applications (FPL)，Porto，Portugal，2013：1－8.

[4] 李智华，黄 娟，李 威，等．一种 SRAM 型 FPGA 互连资源的位流码配置方法 [J]．太赫兹科学与电子信息学报，2016，14 (1)：136－142.

[5] 苏 阳，赵英潇，黄 睿，等．FPGA 的多路数据并行录取和时序资源优化 [J]．单片机与嵌入式系统应用，2017，17 (7)：19－22.

[6] 王子龙，郑美松，涂 吉，等．SRAM 型 FPGA 的基于可观性度量的选择性三模冗余方法 [J]．计算机辅助设计与图形学学报，2015，27 (11)：2184－2191.

[7] LEE J. Improved schedulability analysis using carry－in limitation for non－preemptive fixed－priority multi－processor scheduling [J]. IEEE Transactions on Computers，2017 (99)：1816－1823.

[8] CARLO S D，GAMBARDELLA G，PRINETTO P，et al. SA－FEMIP：a self－adaptive features extractor and matcher IP－core based on partially reconfigurable FPGAs for space applications [J]. IEEE Transactions on Very Large Scale Integration Systems，2015，23 (10)：2198－2208.

[9] SAHA S，SARKAR A，CHAKRABARTI A. Scheduling dynamic hard real－time task sets on fully and partially reconfigurable platforms [J]. IEEE Embedded Systems Letters，2015，7 (1)：23－26.

[10] 王文彬．碎片资源干扰下的云计算资源优化调度仿真 [J]．计算机仿真，2016，33 (7)：359－362.

[11] AGNE A，HAPPE M，KELLER A，et al. ReconOS：an operating system approach for reconfigurable computing [J]. IEEE Micro，2014，34 (1)：60－71.

[12] 何乃味．基于任务预测的可重构资源合并算法研究 [J]．计算机应用与软件，2013，30 (5)：197－199.

[13] MORALES V A，KUMAR R，GORDON R A. Configuration prefetching and reuse for preemptive hardware multitasking on partially reconfigurable FPGAs [C] //Proceedings of the 2016 Conference on Design，Automation & Test in Europe，Dresden，Germany，2016：1505－1508.

[14] BLANCHARDON A，CHOTIN A，MEHREZ H，et al. Impact of defect tolerance techniques on the criticality of a SRAM－based mesh of cluster FPGA [C] //Proceedings of 2014 International Conference on ReConfigurable Computing and FPGAs (ReConFig)，Cancun，Mexico，2014：1－6.

[15] Al-WATTAR A，AREIBI S，GREWAL G. An efficient evolutionary task scheduling/binding framework for reconfigurable systems [J]. International Journal of Reconfigurable Computing，2016 (2)：1－24.

[16] 张海涛，孙滋唱，张泽晗，等．基于多 FPGA 的动态可重配置的多任务调度和放置方法 [P]．中国：CN110231986A. 2019－09－13.

[17] 陈 雪，张隽丰．HRCA 系统的可重构单元的二维任务放置方法 [P]．中国：CN102999435B. 2017－02－22.

[18] 张胜辉．基于 FPGA 加速的异构计算多结点系统实时硬件任务调度与管理 [D]．南昌：华东交通大学，2014.

[19] 郭 兵，沈 艳，蔡富强，等．一种基于放置代价的可重构系统软/硬件任务统一调度方法 [P]．中国：CN101944049B. 2014－04－02.

[20] 李智翔，李 赟，贺 亮．一类特殊的非抢占式周期任务的调度方法 [J]．计算机工程与应用，2018，54 (9)：22－27.

(上接第 204 页)

[21] PAN C，ZHOU Y，WANG J. CFD study of heat transfer for oscillating flow in helically coiled tube heat－exchanger [J]. Computers & Chemical Engineering，2014，69：59－65.

[22] 张周卫，薛佳幸，汪雅红．双股流低温缠绕管式换热器设计计算方法 [J]．低温工程．2014，6：17－27.

[23] 梁 琳，袁宇阳，张 庆．螺旋缠绕管换热器相变流动工艺计算软件开发及应用 [J]．广东化工．2019，2 (46)：172－174.

[24] 薛佳幸．缠绕管式换热器换热工艺研究 [D]．兰州：兰州交通大学，2015.

[25] 卞庆飞．离心泵动静叶栅内流模拟及优化分析 [D]．徐州：中国矿业大学，2015.

[26] 买买提明·艾尼，热合买提江·依明．现代数值模拟方法与工程实际应用 [J]．工程力学，2014，31 (4)：11－18.

[27] 焦 兰．三叶孔整圆形支撑板换热器的实验及数值模拟研究 [D]．武汉：华中科技大学，2011.

[28] WEBB R L. Performance evalution criteria for use of enhanced heat transfer srufaces in heat exchanger design [J]. International Journal of Heat and Mass Transfer，1981，24 (4)：715－726.

[29] 余建祖．最新热交换器设计计算与传热强化及质量检验标准规范实用手册 [M]．北京：机械工业出版社，2006.

[30] NAPHON P. Study on the heat transfer and flow characteristics in a sprial－coil tube [J]. Mass Transfer，2011，38 (1)：69－74.

[31] 梁 新，李 浩．管壳式换热器设计软件的开发 [J]．计算机与应用化学，2008，25 (5)：19－21.

[32] 马 飞．螺旋缠绕管换热器传热数值模拟 [D]．郑州：郑州大学，2014.