



面向SCA的DPR软件架构设计与调度技术

郭彪^{1,2}, 唐麒², 文智敏³, 傅娟⁴, 王玲¹, 魏急波²

(1. 湖南大学 电气与信息工程学院, 长沙 410082; 2. 国防科技大学 电子科学学院, 长沙 410073;

3. 长沙轨道交通运营有限公司, 长沙 410000; 4. 军事科学院 系统工程研究院, 北京 100101)

摘要: 为提高软件无线电(SDR)系统波形应用部署的灵活性和FPGA资源利用率, 基于SDR系统的软件通信体系架构(SCA), 设计一种支持FPGA的动态部分可重构(DPR)软件架构。针对DPR FPGA与CPU组成的异构计算平台, 提出一种蚁群优化调度算法, 以提高波形应用部署效率。实验结果表明, 与MILP算法和ILP算法相比, 所提算法的求解性能平均提升了约30%, 且随着任务规模的增大, 优势更加明显。

关键词: 软件通信体系架构; 动态部分可重构; 可扩展标记语言; 有向无环图; 蚁群优化

开放科学(资源服务)标志码(OSID):



中文引用格式: 郭彪, 唐麒, 文智敏, 等. 面向SCA的DPR软件架构设计与调度技术[J]. 计算机工程, 2021, 47(12): 221-229.

英文引用格式: GUO B, TANG Q, WEN Z M, et al. DPR software architecture design and scheduling technology for SCA[J]. Computer Engineering, 2021, 47(12): 221-229.

DPR Software Architecture Design and Scheduling Technology for SCA

GUO Biao^{1,2}, TANG Qi², WEN Zhimin³, FU Juan⁴, WANG Ling¹, WEI Jibo²

(1. College of Electrical and Information Engineering, Hunan University, Changsha 410082, China;

2. College of Electronic Science and Technology, National University of Defense Technology, Changsha 410073, China;

3. Changsha Rail Transit Operation Co., Ltd, Changsha 410000, China;

4. Academy of System Engineering, Academy of Military Science, Beijing 100101, China)

[Abstract] In order to improve the flexibility of waveform application deployment and FPGA resource utilization in Software Defined Radio(SDR) systems, a software architecture that supports Dynamic Partial Reconfiguration(DPR) of FPGA is designed based on the Software Communication Architecture(SCA) in SDR systems. At the same time, in order to improve the efficiency of waveform application deployment, an Ant Colony Optimization(ACO) scheduling algorithm is designed for the heterogeneous computing platform composed of DPR FPGA and CPU. Experimental results show that compared with the MILP algorithm and the ILP algorithm, the proposed algorithm improves the solution performance by about 30% on average, and the solution performance grows along with the task size.

[Key words] software communication architecture; dynamic partial reconfiguration; eXtensible Markup Language(XML); directed acyclic graph; ant colony optimization

DOI: 10.19678/j.issn.1000-3428.0060642

0 概述

目前, 不同用户需求和迭代衍生出WIFI、蓝牙等短距无线通信以及卫星通信、GSM等长距无线通信。硬件平台趋于多样化, 设备之间存在兼容性差、软件开发和维护升级周期长、部署成本高等问题。软件无线电(Software Defined Radio, SDR)技术

提供了一种有效的、成本相对低的解决方案, 通过软件更新升级即可实现多模式、多频段、多功能的无线通信^[1]。

SDR在移动终端、通信基站等民用领域应用广泛。在军事上, SDR作为新一代军事无线通信领域的关键技术, 已成为全球军事无线通信系统的技术体制和发展方向。以美军为例, 其各军种的无线通信频率

基金项目: 国家自然科学基金(62001483, U19B2024); 湖南省科技创新计划(2020RC2045)。

作者简介: 郭彪(1995—), 男, 硕士, 主研方向为软硬件划分算法、软件无线电; 唐麒, 副研究员、博士; 文智敏, 工程师; 傅娟, 工程师、博士; 王玲、魏急波, 教授、博士。

收稿日期: 2021-01-19 **修回日期:** 2021-03-01 **E-mail:** guobiao@hnu.edu.cn

范围为2 MHz~3 GHz,频谱覆盖范围广,相互之间通信协同性要求高。美军主导制定了联合战术通信系统和联合战术网络中心计划,在2012年全面实现软件无线电电台装备体制,截止到2017年底装备约50万部各型软件无线电电台。这种电台改变了传统以硬件平台为主的设计模式,通过将接收、发送、调制、解调等功能进行模块化设计,从而装载不同功能的波形软件,实现同一平台的多种通信方式切换。

SDR系统包括硬件支撑平台和软件体系架构。软件体系架构是支撑整个系统的核心,其中以美军JTRS/JTNC项目中发布的软件通信体系结构(Software Communication Architecture, SCA)最具代表性。目前有诸多针对SDR的相关研究^[2-4],在硬件支撑平台方面,文献[5]搭建了通用一体化SDR平台,该平台可实现基本的SDR射频数据收发功能。文献[6]设计了基于ZEDBOARD的嵌入式软件无线电软件平台,该平台拥有较高的集成度和可拓展性。

基于SCA的软件架构是SDR系统的核心,相关研究也十分广泛^[7]。在SCA的标准符合性测试方面,文献[8]基于扩展有限状态机进行SCA符合性测试,并使用了贪心算法以快速验证测试波形应用是否符合SCA标准规范。在SCA的框架设计方面,文献[9]设计了轻量化的SCA核心框架,解决了接口冗余与可调整性不足的问题。在SDR系统兼容性发展方面,SCA已从被设计时部署在通用处理器(General Purpose Processor, GPP)上面,到现在可支持多种类型设备的扩展。文献[10]解决了SCA在数字信号处理器(Digital Signal Processor, DSP)上的部署问题,通过其框架设计与调制解调器硬件抽象层

(Modem Hardware Abstraction Layer, MHAL)的使用,使波形组件可以在兼容SCA规范的前提下运行在DSP上。文献[11]设计了基于SCA的新型软硬件架构,通过GPP+DSP+FPGA的数字基带处理单元在硬件平台构建符合SCA的规范,将该架构应用在雷达、数据链、电子战等多种领域。

随着目前以赛灵思Zynq-7000等为代表的新型DPR FPGA计算架构的出现,如何在SCA中对FPGA动态部分可重构能力提供支撑的问题尚未得到有效解决。传统的SCA将FPGA虚拟化为单一资源,导致资源不能得到有效的空时复用,且SCA存在资源粒度大、部署方式单一的缺点。

本文将SCA和FPGA DPR技术相结合,设计基于SCA的动态部分可重构软件架构。通过扩展SCA功能,提升波形应用在动态部分可重构FPGA下的实时部署能力。在此基础上,设计ACO调度算法增强组件的调度和管理方式,提高硬件平台的使用效率。

1 SCA与可重构技术

1.1 SCA软件体系架构

SCA是美军为建立联合战术通信电台开发定义的与设计实现无关的框架。SCA通过公共对象请求代理体系结构定义统一的开发标准框架和通用结构,使通信系统开发的软件和硬件可以分离,其宗旨是通过在同一硬件平台上加载不同的波形组件以实现不同的功能。此外,也可根据特定的需要对系统进行部分升级,从而最大化地利用系统资源^[12]。基于SCA的通信系统架构如图1所示。

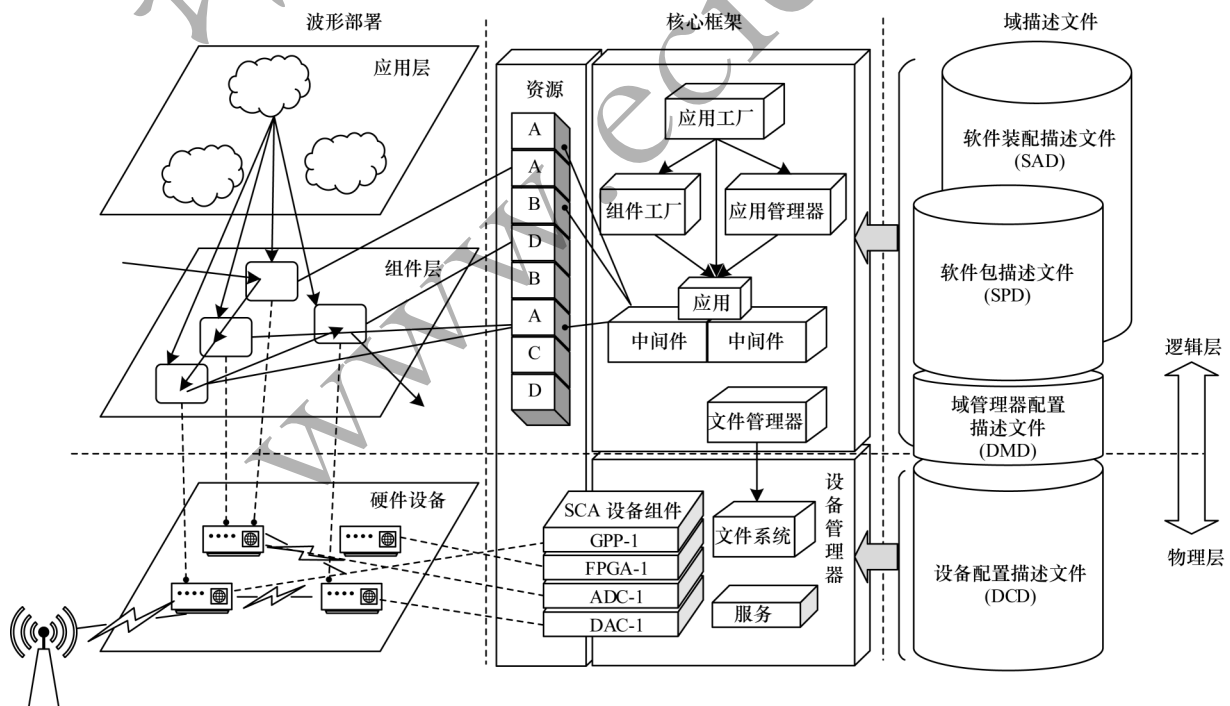


图1 基于SCA的通信系统架构

Fig.1 Communication system architecture based on SCA

如图1所示,该系统包括上层的波形应用、中间的核心框架及对应的域描述文件3个部分。其中波形应用由不同功能的组件组成,用来实现FSK、MSK、扩频等通信功能。核心框架是一系列应用编程接口与组件集合,在操作系统上对软件和底层硬件进行更高层次封装,为波形应用提供标准化接口和服务,并对波形应用的动态加卸载、部署和域内资源进行统一管理。域描述文件采用可拓展标记语言(eXtensible Markup Language, XML)配置SCA域内的所有组件。根据配置功能的不同又分为软件包描述文件(Software Package Descriptor, SPD)、设备配置描述文件、软件装配描述文件(Software Assembly Descriptor, SAD)等^[13]。这些文件配置了域内每个组件的接口、容量、属性、内部依赖、互联、逻辑位置等信息^[14]。核心框架通过解析组件的

域描述文件,完成组件的部署、启动、配置、查询等操作。

SCA将FPGA、DSP、GPP等物理硬件设备虚拟化为域内的设备组件,设备组件通过提供统一的加载、卸载、部署等接口满足硬件平台的可复用性需求。

1.2 FPGA的DPR技术

基于FPGA的动态可重构技术能够在FPGA运行过程中将其内部的全部或部分逻辑资源重新配置,进而实现FPGA逻辑功能的动态切换和时分复用,而不终止器件的运行^[15]。此外,FPGA的DPR技术重构粒度更小,允许每个重构区域根据功能需要动态加载和切换配置文件。FPGA通过内部配置访问端口下载配置文件,并对指定区域进行逻辑功能实时更改而不影响其他区域的任务执行。FPGA DPR功能示意图如图2所示。

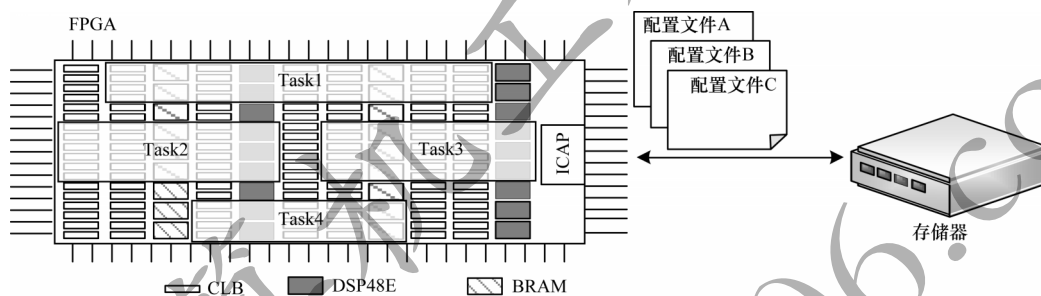


图2 FPGA动态部分可重构示意图

Fig.2 Schematic diagram of dynamic partial reconfiguration of FPGA

图2中的DPR FPGA划分了4个重构子区域,并分别执行Task1、Task2、Task3、Task4任务。每个重构区域之间独立执行,可根据任务需求实时更改重构子区域的数量和大小,不影响其他正在执行的重构区域。

在现有的SCA软件架构体系下,只有面向FPGA全局重构方式的资源虚拟化支撑,没有针对DPR FPGA的相关虚拟化架构。本文第2节在基于SCA的规范下设计DPR FPGA的硬件资源虚拟化软件架构,提出波形组件DPR部署的流程机制。

2 基于SCA的DPR软件架构设计

为满足SDR系统的异构性和可移植性需求,SCA域内将FPGA硬件设备虚拟化为FPGA设备组件,核心框架通过管理设备组件来间接管理FPGA硬件。设备组件的域描述文件利用XML和统一的软件架构完成硬件资源虚拟化。其主要配置两部分信息:一是该设备组件自身运行所需的软硬件环境,主要包括操作系统、设备组件编程代码语言及设备组件代码在系统内的存放路径等;二是该设备组件所代表的硬件资源,包括硬件设备名称、资源量(内存、计算单元)、接口等信息。

然而,现有SCA规范下的软件架构将FPGA设备虚拟化为单一的硬件资源,导致FPGA的重构粒

度大,不能进行空时复用。

2.1 DPR FPGA设备组件域描述文件架构设计

将DPR FPGA划分的子区域数量和子区域大小不同称为不同的划分方式。将DPR FPGA硬件设备设计并抽象封装成支持DPR能力的逻辑设备组件时,该逻辑设备组件应支持不同的划分方式、不同区域约束和不同资源配置。同时,区域描述文件的软件架构要符合SCA规范要求。图3所示为本文设计提出的DPR FPGA设备组件域描述文件的软件架构示意图。其中,DPR FPGA逻辑设备组件的域描述文件(Domain Profile)主要由软件包描述文件(SPD)和属性配置文件(PRF)组成。SPD文件的根元素(softpkg)配置了组件开发者(author)、组件实现(implementation)、属性配置文件(PRF)等信息。其中author元素配置了开发者姓名(name)、组件开发公司(company)等信息;implementation元素配置了组件代码(code)、组件运行环境(os)、组件执行处理器(processor)等信息;code元素配置了组件接口信息(entrypoint)、组件代码位置(localfile)等信息。PRF文件中的结构元素(struct)用来配置多个不同类别组件属性,simple元素用来配置单一属性。

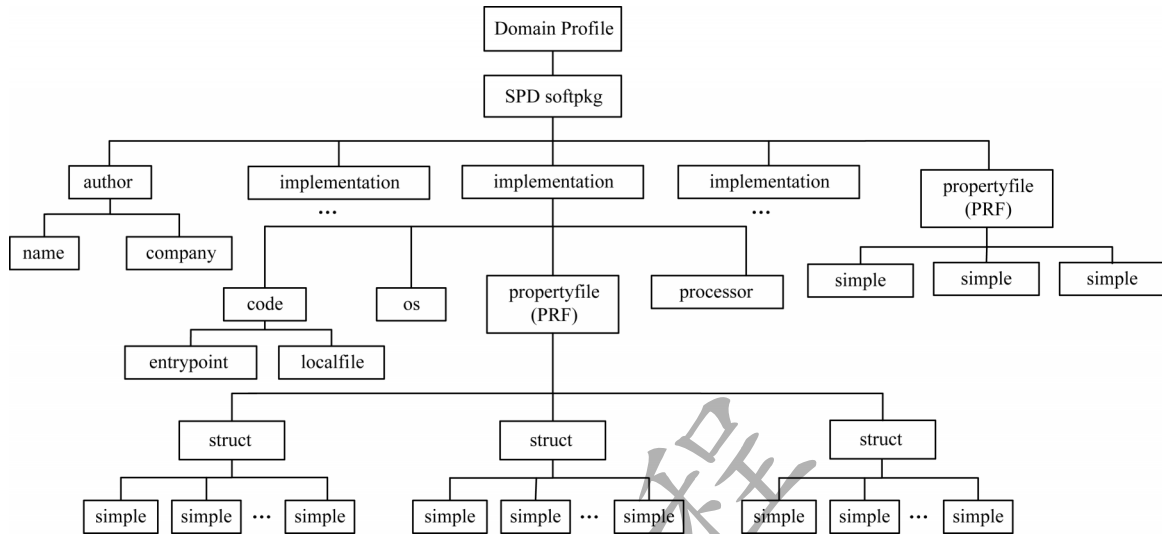


图3 DPR FPGA设备组件域描述文件的软件架构

Fig.3 Software architecture of DPR FPGA device component domain description file

上述架构在SCA的规范下提供了对DPR FPGA的资源虚拟化,满足了如下3个约束:

1) DPR FPGA支持多种区域划分方式。通过配置图3中SPD文件的softpkg根元素,使其包含一个或多个implementation子元素,部分代码实现如下:

```
<softpkg id="DCE: 03974D2E-FBB0-11E9-A2B2-0235D2B38928" name="FPGA_Device">
  <implementation id="DCE: 03974D2E-FBB0-11E9-A2B2-0235D2B38928">
    <description>Partition Kind1
    </description>
  </implementation>
  <implementation id="DCE: 03974D2D-FBB0-11E9-A2B2-0235D2B38928">
    <description>Partition Kind2
    </description>
  </implementation>
</softpkg>
```

softpkg元素下的2个implementation子元素分别表示一种FPGA动态部分可重构区域划分方式的资源描述,每一种划分方式包含不同的FPGA动态部分可重构区域的数量和大小。

2) 每种划分方式支持不同的重构子区域资源描述。配置implementation元素包含的PRF属性文件,该文件由多个struct元素组成,每个struct元素代表一个重构区域,部分实现代码如下:

```
<properties>
  <!--重构区域1-->
  <struct id="DCE: 03974D2A-FBB0-11E9-A2B2-0235D2B38928" name="Static_Region" mode="readwrite">
    <description>Static Reconfigurable Region
    </description>
  </struct>
```

```
<!--重构区域2-->
```

```
<struct id="DCE: 03974D2B-FBB0-11E9-A2B2-0235D2B38928" name="Dynamic_Region 1" mode="readwrite">
  <description>Dynamic Reconfigurable Region 1
  </description>
</struct>
```

```
<!--重构区域3-->
```

```
<struct id="DCE: 03974D2C-FBB0-11E9-A2B2-0235D2B38928" name="Dynamic_Region 2" mode="readwrite">
  <description>Dynamic Reconfigurable Region 2
  </description>
</struct>
</properties>
```

上述代码中配置了3个重构区域,分别是静态重构区、动态重构区1和动态重构区2。

3) 每个重构区域支持细粒度的资源配置。图3中的每个simple元素表示重构区域的一个资源配置信息。资源配置信息包括重构区域标识、重构区域坐标位置、重构区域CLB Logic Cells数量、重构区域CLB Slices数量、重构区域Digital Clock Managers数量、重构区域DSP数量、重构区域RAM数量等。以重构区域CLB Slices数量为例,部分实现代码如下:

```
<simple id="DCE: 03974D2A-FBB0-11E9-A2B2-0235D2B38928" type="long" name="CLB_Logic_Cells" mode="readonly">
  <value>2132</value>
  <description>CLB Logic Cells
  </description>
  <kind kindtype="configure"/>
</simple>
```

上述代码中的simple元素配置区域中CLB资源的数量为2 132。

上述针对DPR FPGA设备组件域描述文件的软件架构设计,支持了FPGA的细粒度资源虚拟化,拓展了FPGA DPR功能,建立了FPGA硬件可重构区域到SCA域内的映射关系。软件架构设计使用的所有域描述文件配置元素如simple、struct等均符合SCA规范,保证了SCA整体架构的一致性和统一性,减小了SCA软件的升级和维护难度。

2.2 基于SCA的波形组件DPR部署机制

波形应用是完成某种通信功能的软件应用,由多个组件组成,图4所示为波形应用示例。

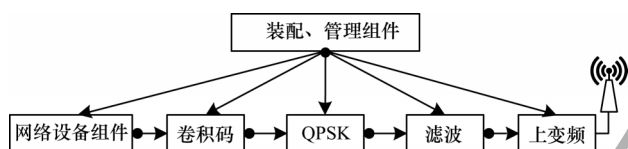


图4 波形应用示例

Fig.4 Sample of waveform application

该波形应用由网络设备、滤波、变频等组件组成。基于SCA的SDR系统,域管理器通过解析波形应用的XML域配置文件启动、初始化、运行、配置、重构相应的波形应用程序^[16]。

在2.1节中完成了FPGA DPR的资源虚拟化,使SCA域内有了FPGA动态部分可重构区域的资源信息。为了将波形组件部署到FPGA动态部分可重构区域,本节提出了在SCA域内波形组件DPR部署的流程和机制,如图5所示。

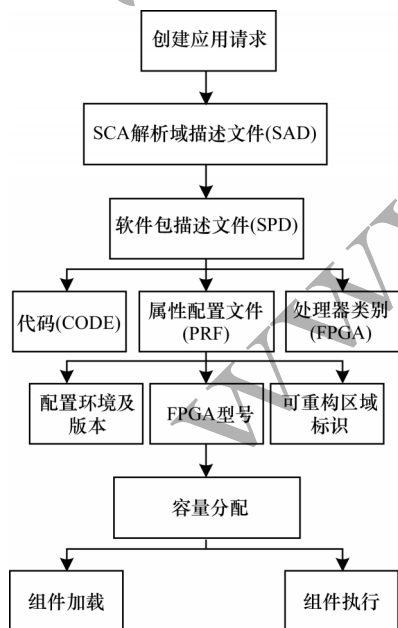


图5 波形组件DPR部署机制

Fig.5 DPR deployment mechanism of waveform components

在图5中,客户端首先调用核心框架控制接口中ApplicationFactory接口的create操作。create操作在SCA域内创建应用,从而在客户端请求的设备上创建应用提供客户端接口。SCA解析波形应用的域描述文件,包括应用的SAD文件、各个组件的SPD文件等,从而获得应用的组成、各个组件之间的端口连接、组件部署所需的处理器/运行环境等配置信息。逻辑设备通过执行allocateCapacity操作,分配组件运行所需的内存、处理器,从而更新Device(s)的内存和处理器使用状态。逻辑设备还可以通过LoadableInterface和ExecutableInterface接口完成组件的加载和执行。如此,波形组件DPR部署机制建立了波形应用创建、依赖性检查、容量分配、可重构部署完整的映射,提高了组件部署和管理的灵活性。

3 基于蚁群优化算法的调度技术

本文第2节提供了SCA下对于DPR FPGA的架构支撑,包含DPR FPGA的SDR硬件设备系统,使组件部署更具灵活性。DPR FPGA与CPU计算设备构成了可重构的异构计算平台。为进一步提高波形应用在该平台的部署效率,提升硬件资源利用率,高效的调度算法支撑尤为关键。

3.1 波形应用与计算平台建模

目前针对应用的建模方法有多种,如有向无环图(DAG)^[17-19]、同步数据流图(SDFG)^[20-22]等。本文运用目前研究较多的DAG图建模波形应用。图6所示为DAG应用示例。

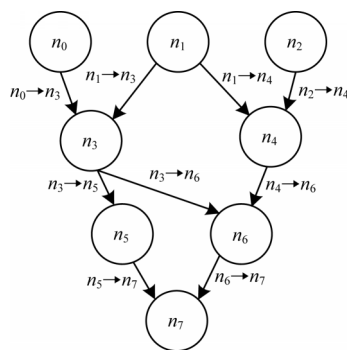


图6 DAG应用示例

Fig.6 Sample of DAG application

在图6中,该应用由8个任务和9条边组成。任务之间的箭头表示相互依赖关系。将任务在DPR FPGA上的执行时间称为硬件执行时间,在CPU上的执行时间称为软件执行时间。表1为具体任务参数信息,包括任务名、软件执行时间、硬件执行时间以及在FPGA上执行所需的CLB资源数量。

表1 波形应用的任务参数

Table 1 Task parameters of the waveform application

任务名	硬件执行时间	软件执行时间	所需 CLB 数量
n_0	26	13	9
n_1	9	4	4
n_2	10	4	2
n_3	23	6	2
n_4	6	1	1
n_5	28	11	3
n_6	24	2	1
n_7	2	1	1

DPR FPGA 与 CPU 组成的 SDR 系统中的异构计算平台模型如图 7 所示。

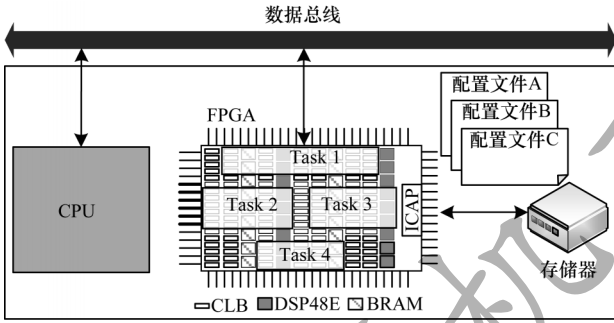


图7 异构计算平台

Fig.7 Heterogeneous computing platform

图 7 中的异构计算系统由一个 CPU 和支持二维重构的 DPR FPGA 组成,其中 FPGA 被划分为 4 个重构子区域,通过加载不同的配置文件来更改其逻辑功能。

3.2 调度问题

高效的调度算法可以减小波形应用的执行时间,从而提高系统计算资源的利用率。将确定 DAG 图中各个任务的执行方式(软件执行或硬件执行)和执行时间(任务的开始时间和结束时间)称为调度。调度结果示意图如图 8 所示。

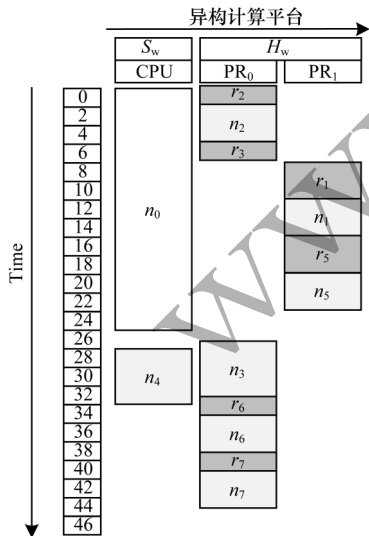


图8 波形应用的调度结果示意图

Fig.8 Schematic diagram of scheduling results of the waveform application

在图 8 中,计算单元包括一个 CPU 和 2 个 DPR FPGA 重构子区域,深灰色方块代表任务在 FPGA 上的重构时间,浅灰色方块代表任务的执行时间。

对于 3.1 节异构计算平台,在对波形应用进行调度时需要满足一定的约束条件,具体如下:

1) DAG 图中的任务依赖关系。子任务需要等其所有父任务执行和通信结束后才能开始执行。

2) 任务只能执行一次且只能选择一个计算单元(CPU 或 FPGA 的一个重构区域)。

3) FPGA 上执行的任务需要先重构再执行。

4) 限制 FPGA 的资源数量,需满足任务执行所需的资源且同一时间不能超过 FPGA 的资源总量。

5) 重构端口不能复用,且同一时间只能重构一个任务。

3.3 算法设计

针对 3.1 节的应用和计算平台模型,本文提出一种基于蚁群优化的调度算法,通过设置循环次数和控制参数进行迭代求解。每次迭代计算主要分为 3 步:确定任务的调度顺序;映射任务到计算单元;信息素更新。该算法能够解决 SDR 系统中的波形应用调度问题,有效支撑波形应用的部署。

下面将对算法细节进行介绍。

步骤 1 确定任务的调度顺序。算法实现时,通过计算组成 DAG 各个任务的调度概率,选择调度概率最大的任务作为当前步骤需要调度的任务。任务 i 在第 j 步的调度概率如下:

$$p_{ij}^s = \frac{[\tau_{ij}^s]^\alpha \times [\eta_{ij}^s]^\beta}{\sum_{l \in N_j} [\tau_{il}^s]^\alpha \times [\eta_{il}^s]^\beta} \quad (1)$$

其中: α^s 为全局调度信息素挥发控制因子; β^s 为局部调度信息素挥发控制因子; τ_{ij}^s 为全局调度信息素; η_{ij}^s 局部调度信息素;集合 $N = \{1, 2, \dots, M\}$ 为调度 DAG 所需要的总步数,其中 M 的值等于组成 DAG 任务的数量。

步骤 2 映射任务到计算单元。确定好每一步待调度的任务后,将任务映射到计算单元上并执行,其中计算单元集合由 CPU 和 DPR FPGA 的重构区域组成。计算任务的“映射概率”,并选取计算值最大的映射单元为计算单元映射。任务 i 选择计算单元 k 的映射概率如下:

$$p_{ik}^m = \frac{[\tau_{ik}^m]^\alpha \times [\eta_{ik}^m]^\beta}{\sum_{l \in N} [\tau_{kl}^m]^\alpha \times [\eta_{kl}^m]^\beta} \quad (2)$$

其中: α^m 为全局映射信息素挥发控制因子; β^m 为局部映射信息素挥发控制因子; τ_{ik}^m 为全局映射信息素; η_{ik}^m 局部映射信息素; N 是所有计算单元的集合。

步骤 3 信息素更新。所有蚂蚁完成一次求解时,即更新对应的信息素。信息素的更新公式如下:

$$\tau_{ij}^s = (1 - \rho^s) \times \tau_{ij}^s + \rho^s \times \frac{1}{S_{SL}} \quad (3)$$

$$\tau_{ik}^m = (1 - \rho^m) \times \tau_{ik}^m + \rho^m \times \frac{1}{S_{SL}} \quad (4)$$

$$S_{SL} = \min(s_i), i \in \text{Antcount} \quad (5)$$

其中: ρ^s 为调度信息素挥发因子; ρ^m 为映射信息素挥发因子; S_{SL} 为本次迭代所有蚂蚁求解的应用完成时间最小值。其中 s_i 为第 i 只蚂蚁求解的调度长度,其大小等于本次迭代所有任务所完成时间的最大值。Antcount

的大小等于蚂蚁数量。上述 3 个步骤为单次迭代求解过程,算法执行时不断进行迭代求解。当达到所设置迭代次数的上限时返回已经求得的最优解,该最优解中包含了波形应用的调度结果和波形应用的总完成时间。

4 仿真实验

4.1 实验参数设置

为验证本文所提算法的性能,波形应用调度的 DAG 图使用文献[23]中的方式随机生成。表 2 所示为随机生成 DAG 的详细信息,包括任务数 n 、边数 m 和问题规模 $2n+3m$ ^[24]。由于软件无线电的硬件计算平台大多执行计算密集型的波形应用,通信计算比(CCR)较小,故本文测试基准 DAG 的 CCR 设置为 0.1。所提算法中的迭代次数为 1 000 次,蚂蚁数量为 5,全局调度信息素挥发控制因子 α^s 设置为 1,局部调度信息素挥发控制因子 β^s 、全局映射信息素挥发控制因子 α^m 及局部映射信息素挥发控制因子 β^m 均设置为 1,调度信息素挥发因子 ρ^s 设置为 0.9,映射信息素挥发因子 ρ^m 设置为 0.98。为了验证所提调度算法的有效性,本文仿真了没有调度的 DAG 随机算法部署(Random Deployment Without Scheduling,RDWS)、MILP 算法^[25]和 ILP 算法^[26]的调度结果。调度性能的评价指标为 DAG 的调度长度(SL)和算法求解时间。其中 MILP 算法的求解器为 LINDO API12.0,ILP 算法的求解器为 Gurobi 9.0.1,求解时间上限值(timeout)均设置为 1 800 s,若求解器达到时间上限值时仍没有求出最优解,则返回当前已求得的最优解。

表 2 随机生成的 DAG 信息

Table 2 Randomly generated DAG information

DAG 名称	任务数	边数	问题规模
random1	10	13	59
random2	15	22	96
random3	20	32	136
random4	25	39	167
random5	30	49	207
random6	35	59	247
random7	40	72	296
random8	45	101	393
random9	50	101	403
random10	55	117	461
random11	60	146	558
random12	65	156	598
random13	70	174	662
random14	75	161	633
random15	80	208	784
random16	85	247	911
random17	90	298	1 074
random18	95	223	859
random19	100	336	1 208

4.2 实验结果与分析

表 3 为仿真实验的结果数据,包含 DAG 的调度长度和算法求解时间。

表 3 本文算法仿真实验结果

Table 3 Simulation results of the algorithm in this paper

DAG 名称	任务数	ACO		MILP		ILP		RDWS	
		SL	时间/s	SL	时间/s	SL	时间/s	SL	时间/s
random1	10	46	19.398	42	4.94	42	3.12	104	—
random2	15	79	31.239	67	290.20	67	245.56	178	—
random3	20	109	51.872	107	timeout	100	timeout	231	—
random4	25	122	62.621	143	timeout	140	timeout	318	—
random5	30	162	81.358	184	timeout	176	timeout	391	—
random6	35	199	101.422	281	timeout	243	timeout	490	—
random7	40	182	114.871	293	timeout	261	timeout	429	—
random8	45	205	147.817	311	timeout	285	timeout	540	—
random9	50	234	164.386	498	timeout	437	timeout	559	—
random10	55	243	190.679	413	timeout	352	timeout	623	—
random11	60	325	227.711	768	timeout	568	timeout	797	—
random12	65	297	258.945	592	timeout	491	timeout	621	—
random13	70	322	293.221	745	timeout	638	timeout	760	—
random14	75	392	309.315	860	timeout	694	timeout	877	—
random15	80	334	359.419	790	timeout	687	timeout	792	—
random16	85	390	408.442	916	timeout	812	timeout	924	—
random17	90	439	484.949	983	timeout	836	timeout	1 006	—
random18	95	521	484.919	990	timeout	875	timeout	1 042	—
random19	100	477	666.651	1 006	timeout	913	timeout	1 057	—

为评估所提算法在波形应用调度中的性能,引入调度长度提升程度(S_{SLI})来计算求解性能提升程度。

$$S_{SLI}^{contrast} = \left(1 - \frac{S_{SL}^{ACO}}{S_{SL}^{contrast}}\right) \times 100\% \quad (6)$$

其中: S_{SL}^{ACO} 为本文算法所求出的DAG调度长度; $S_{SL}^{contrast}$ 为对比算法所求DAG调度长度。

图9分析了在面对不同DAG规模时4种算法的调度结果变化关系。由图9可知,任务规模越大,本文所提算法求解性能越好,可以有效提升部署效率,减少应用的总执行时间。图10分析了算法的求解时间变化关系。从图10中可以看出,当任务规模大于15时,MILP和ILP算法在1800s内已经求不出最优解,时间复杂度远大于本文所提算法复杂度。

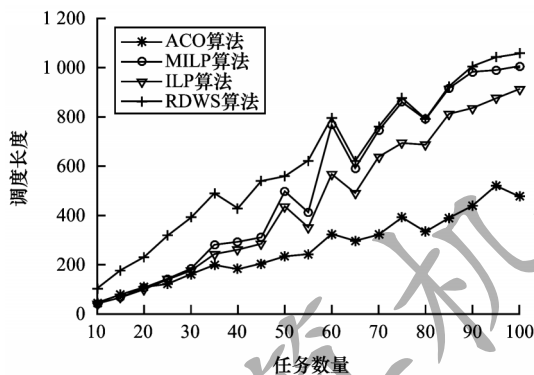


图9 调度长度与任务规模的关系

Fig.9 The relationship between scheduling length and task size

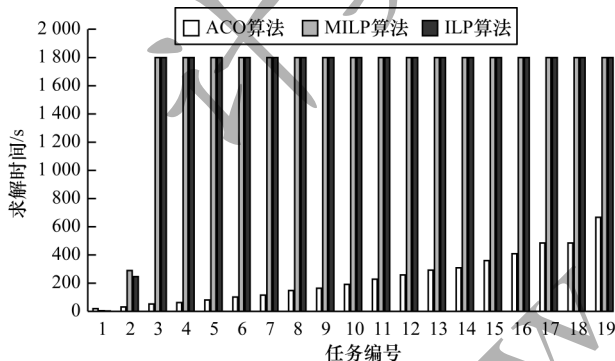


图10 3种算法求解时间对比

Fig.10 Comparison of solution time of three algorithms

表4所示为测试DAG应用的SLI数据。由表4数据可知,与没有调度的随机算法相比,本文所提算法的平均性能提升了57.04%,可有效减少SDR系统中的软件部署和执行时间,提升硬件的资源利用率。与MILP算法相比,当任务规模小于20时,本文算法性能略差于MILP算法,但随着任务规模的增大总体性能呈递增的趋势,最终平均性能提升了35.89%。当任务规模较小时,ILP算法可在时间上限值内求出问题的最优解,求解时间也较短。但随着任务规模的增大,算法的时间复杂度呈指数级增长,任务数量大于20时,在时间上限值内已求不出问题的最优解,求解结果差于本文算法。实验结果表明,本文算

法与ILP算法相比,平均性能提升了29.09%。综合分析,在面对大规模应用时,本文算法在求解性能和求解时间上更有优势。

表4 3种算法SLI分析

DAG名称	任务数	MILP	ILP	RDWS
random1	10	-9.52	-9.52	55.77
random2	15	-17.91	-17.91	55.62
random3	20	-1.87	-9.00	52.81
random4	25	14.69	12.86	61.64
random5	30	11.96	7.95	58.57
random6	35	29.18	18.11	59.39
random7	40	37.88	30.27	57.58
random8	45	34.08	28.07	62.04
random9	50	53.01	46.45	58.14
random10	55	41.16	30.97	61.00
random11	60	57.68	42.78	59.22
random12	65	49.83	39.51	52.17
random13	70	56.78	49.53	57.63
random14	75	54.42	43.52	55.30
random15	80	57.72	51.38	57.83
random16	85	57.42	51.97	57.79
random17	90	55.34	47.49	56.36
random18	95	47.37	40.46	50.00
random19	100	52.58	47.75	54.87

5 结束语

本文将软件通信体系架构与FPGA的DPR技术相结合,提出在软件通信体系架构规范下的DPR FPGA设备资源虚拟化及波形组件的部署机制。介绍基于XML域描述文件的DPR软件架构设计,并从软件架构层面建立从波形组件到FPGA可重构区域的映射关系。在此基础上,针对SDR系统中支持DPR FPGA的异构计算平台和波形应用,提出基于蚁群优化的调度技术算法。实验结果表明,与MILP算法和ILP算法相比,该算法能有效提升调度性能,减少求解时间。下一步将对列表启发式和搜索元启发式算法进行系统设计,以支持波形应用的实时调度。

参考文献

- [1] 张俊妹. 基于SDR平台的SCA波形组件设计[D]. 哈尔滨:哈尔滨工业大学,2009.
ZHANG J S. Design of SCA waveform based on SDR platform[D]. Harbin: Harbin Institute of Technology, 2009. (in Chinese)
- [2] 刘鹏飞,杜欣军. 基于加权反馈三维混沌系统的调制跳变通信系统[J]. 计算机工程,2021,47(3):183-189,195.
LIU P F, DU X J. Modulation hopping communication system based on weighted feedback three-dimensional chaotic system[J]. Computer Engineering, 2021, 47(3): 183-189, 195. (in Chinese)

- [3] MANOUFALI M, BIALKOWSKI K, MOBASHSHER A T, et al. Insitu nearfield path loss and data communication link for brain implantable medical devices using software-defined radio[J]. IEEE Transactions on Antennas and Propagation, 2020, 13(3): 6787-6799.
- [4] SADAT M N, VARGAS A E, DAI R, et al. QoE-VS: a cross-layer qoe-aware video streaming platform using software-defined radio[C]//Proceedings of 2020 IEEE vehicular technology conference. Washington D. C., USA: IEEE Press, 2020: 1-6.
- [5] 胡婉如,王竹刚,张仁良,等. 通用一体化测试平台设计[J]. 电讯技术, 2020, 60(11): 1368-1372.
- HU W R, WANG Z G, ZHANG R L, et al. Design of a universal integrated test platform[J]. Telecommunication Engineering, 2020, 60(11): 1368-1372. (in Chinese)
- [6] 暴臻涛. 基于ZedBoard的嵌入式软件无线电软件平台设计研究[J]. 数码世界, 2020(11): 68-69.
- BAO Z T. Design and research of embedded software radio software platform based on ZedBoard[J]. The Digital World, 2020(11): 68-69. (in Chinese)
- [7] PUTTHAPIPAT P, ANDRIAN J H, LIU C. Studies on inter-component communication latency based on variation number of components and packet size in SDR-SCA waveform application[J]. International Journal of Computational Science and Engineering, 2016, 12(1): 65-72.
- [8] 伍旭东,唐麒,张伟,等. 基于扩展有限状态机的SCA符合性测试方法研究[J]. 计算机工程与应用, 2021, 57(16): 263-268.
- WU X D, TANG Q, ZHANG W, et al. Research on SCA compliance testing method based on extended finite state machine[J]. Computer Engineering and Applications, 2021, 57(16): 263-268. (in Chinese)
- [9] 刘铮,施峻武,唐麒. 面向小型设备的SCA核心框架优化设计[J]. 中国新通信, 2014, 16(18): 96-98.
- LIU Z, SHI J W, TANG Q. Optimized design of sca core framework for small devices[J]. China New Communications, 2014, 16(18): 96-98. (in Chinese)
- [10] 唐麒,施峻武,吴宇. 一种基于SCA的DSP软件无线电框架实现[J]. 现代电子技术, 2011, 34(17): 85-89.
- TANG Q, SHI J W, WU Y. Implementation of DSP software defined radio architecture based on software communication architecture[J]. Modern Electronics Technique, 2011, 34(17): 85-89. (in Chinese)
- [11] 赵宾华,黄伟,邓炜. 面向多领域应用的软件无线电架构设计[J]. 河北省科学院学报, 2020, 37(3): 15-19.
- ZHAO B H, HUANG W, DENG W. Research on software defined radio system for multi-field applications[J]. Journal of the Hebei Academy of Sciences, 2020, 37(3): 15-19. (in Chinese)
- [12] 李欣宇. DDS在SCA系统中的应用[J]. 计算机工程与设计, 2013, 34(6): 1931-1935.
- LI X Y. Application of DDS for SCA system[J]. Computer Engineering and Design, 2013, 34(6): 1931-1935. (in Chinese)
- [13] JPEO J. Software communications architecture specification[EB/OL]. [2020-12-03]. <http://www.doc88.com/p-9502563599238.html>.
- [14] 唐麒. 小型化软件通信体系结构的研究与实现[D]. 长沙:国防科学技术大学, 2011.
- TANG Q. Research and implementation of lightweight software communication architecture[D]. Changsha: National University of Defense Technology, 2011. (in Chinese)
- [15] 李昆吉. FPGA动态可重构技术及其应用研究[D]. 哈尔滨:哈尔滨工业大学, 2012.
- LI K J. Research on dynamically reconfigurable technology and its application based on FPGA[D]. Harbin: Harbin Institute of Technology, 2012. (in Chinese)
- [16] 王烟青. 基于SCA可重构的微波通信系统设计与实现[C]//2017年全国微波毫米波会议论文集. 杭州:电子工业出版社, 2017: 4-12.
- WANG Y Q. Design and implementation of reconfigurable microwave communication system based on SCA[C]//Proceedings of 2017 National Conference on Microwave Millimeter Waves. Hangzhou, China: Publishing House of Electronics Industry, 2017: 4-12. (in Chinese)
- [17] BIONDI A, BALSINI A, PAGANI M, et al. A framework for supporting real-time applications on dynamic reconfigurable FPGAs[C]//Proceedings of 2016 IEEE Real-Time Systems Symposium. Washington D. C., USA: IEEE Press, 2016: 36-49.
- [18] KIZHEPPATT V, FAHMY S A. FPGA dynamic and partial reconfiguration: a survey of architectures, methods, and applications[J]. ACM Computing Surveys, 2018, 51(4): 1-39.
- [19] XIAO X, LI Z. Chemical reaction multi-objective optimization for cloud task DAG scheduling[J]. IEEE Access, 2019, 7: 102598-102605.
- [20] STUIJK S, GEILEN M, BASTEN T. SDF3: SDF for free[EB/OL]. [2020-12-03]. <http://www.doc88.com/p-7816259352419.html>.
- [21] XUE Y Z, MARC G, TWAN B, et al. Multiconstraint static scheduling of synchronous dataflow graphs via retiming and unfolding[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2016, 35(6): 905-918.
- [22] SINGH A, EKBERG P, BARUAH S. Uniprocessor scheduling of real-time synchronous dataflow tasks[J]. Real-Time Systems, 2019, 55(1): 1-31.
- [23] TANG Q, BASTEN T, GEILEN M, et al. Mapping of synchronous dataflow graphs on MPSoCs based on parallelism enhancement[J]. Journal of Parallel and Distributed Computing, 2017, 101(3): 79-91.
- [24] 侯能,何发智. 混合并行两步调整遗传策略的软硬件划分算法[J]. 华中科技大学学报(自然科学版), 2017, 45(12): 39-45.
- HOU N, HE F Z. Hybrid parallel genetic strategy with two-step adjustment for HW/SW partitioning[J]. Journal of Huazhong University of Science and Technology (Natural Science Edition), 2017, 45(12): 39-45. (in Chinese)
- [25] 朱丽花,王玲,唐麒,等. 一种针对动态部分可重构SoC软硬件划分的高效MILP模型[J]. 计算机科学, 2020, 47(4): 18-24.
- ZHU L H, WANG L, TANG Q, et al. Efficient MILP model for HW/SW partitioning of dynamic partial reconfigurable SoC[J]. Computer Science, 2020, 47(4): 18-24. (in Chinese)
- [26] TANG QI, WANG ZHE, GUO BIAO, et al. Partitioning and scheduling with module merging on dynamic partial reconfigurable FPGAs[J]. ACM Transactions on Reconfigurable Technology and Systems, 2020, 13(3): 1-24.