面向可重构系统的 eCos 拓展

张亮忠，熊选东，付建丹，王松锋
（解放军信息工程大学 电子技术学院，郑州 450004）

摘　要：针对可重构系统中任务模型灵活性差、硬件任务重构延时长、FPGA 资源利用率低等问题，提出了将应用程序划分为软件任务和混合任务的划分模式，并在 eCos 的基础上，通过重构控制机制、混合任务管理机制、通信机制三方面的拓展，设计了支持可重构系统的嵌入式操作系统框架 eCos4RC。仿真结果表明，eCos4RC 实现了对混合任务的有效管理，在兼容 eCos 多线程机制的同时提高了应用程序执行速度和可重构资源利用率，为可重构计算平台提供了良好的运行环境支持。

关键词：可重构系统；混合任务；嵌入式操作系统；eCos；eCos4RC
中图分类号：TP368　　**文献标志码**：A　　**文章编号**：1001-3695(2012)05-1768-04
doi:10.3969/j.issn.1001-3695.2012.05.044

Extension of eCos for supporting reconfigurable system

ZHANG Liang-zhong, XIONG Xuan-dong, FU Jian-dan, WANG Song-feng
(*Electronic Technology Institute, PLA Information Engineering University, Zhengzhou* 450004, *China*)

Abstract: For solving the traditional deficiencies of reconfigurable system such as the poor flexibility of tasks model, the long time of hardware tasks configuration, the low resource utilization of FPGA, this paper proposed to partition a software application into software tasks and hybrid tasks. Secondly through extension of eCos in three aspects including reconfigurable control mechanisms, hybrid tasks management and inter-task communication mechanisms, this paper presented an embedded operating system prototype called eCos4RC which was compatible with the multi-thread mechanism of eCos. The experiments results indicate eCos4RC could manage the hybrid tasks and reconfigurable computing resources effectively, and provide a runtime support for reconfigurable computing platforms while reducing the execution time of application.

Key words: reconfigurable system; hybrid task; embedded operation system; eCos; eCos4RC

由 CPU 和运行时可重构 FPGA 两种计算资源耦合成的具备动态部分重构(dynamic partial reconfiguration, DPR)能力的可重构系统被大量应用于计算密集型的嵌入式应用中，成为业界的研究热点[1,2]。DPR 技术为硬件任务提供了运行时重构的灵活性，实现了可重构计算资源的分时复用，在系统的灵活性和高效性间取得了折中。当采用可重构 FPGA 实现一个计算逻辑时，其整个运行时间开销包括配置和计算时间。目前制约可重构计算广泛应用的一个重要因素就是配置时间相对于计算时间仍然太长，如何加速和隐藏配置过程已经成为可重构系统的重要研究课题。然而，大量的研究资料都将用户应用划分为软件和硬件任务两类[3]，利用硬件任务的并行性来隐藏配置时间，而没有考虑当任务只能顺序执行时如何隐藏配置时间。另外，由于在操作系统层面缺乏对可重构硬件的运行时服务支持[4,5]，若沿用传统的设备管理方式，仅对可重构器件提供设备驱动，设计人员在设计应用程序时需要关注底层平台实现细节显式地对硬件任务和可重构计算资源进行管理，存在设计难度大、不能充分利用系统的可重构计算资源等问题。

本文从可重构系统的任务划分模式出发，通过对 eCos(embedded configurable operating system)的拓展，设计支持可重构计算的操作系统框架 eCos4RC(embedded configurable operating system for reconfigurable computer)。在可重构资源管理方面，采用硬件任务预配置策略，通过操作系统拓展为其提供统一的重构控制机制；在混合任务管理方面，由操作系统根据可重构资源上任务配置情况，灵活地选择混合任务的实现方式。

1　动态可重构系统任务划分

1.1　可重构计算平台

本文以总线连接的可重构计算平台为研究对象，系统主要由 CPU、可重构器件、系统存储器、重构控制器和通信控制器组成(图 1)。操作系统运行在 CPU 上，负责管理整个可重构计算系统以及处理软件任务。可重构器件作为 CPU 之外的附加计算单元，包含一定数量的可重构处理单元(reconfigurable processing unit, RPU)，用于执行粗粒度的计算密集型任务。重构控制器负责对 RPU 进行配置和释放，根据 CPU 的指令选择硬件任务的配置位流文件在 RPU 上进行配置；通信控制器负责处理底层通信细节，为任务间通信提供支持。各 RPU 具备动态重构能力，在某一时刻重构控制器可对某一个 RPU 重构，以加载新的硬件任务，而对其他正在运行的 RPU 没有影响。

1.2　软件任务/混合任务划分模式

一般地，在可重构计算平台上，多采用软/硬件任务划分模式[6,7]，即将用户任务划分为软件任务(software tasks, SwT)和硬件任务(hardware tasks, HwT)两类。应用中计算密集或具有较高计算并行度的模块划分为硬件任务在 RPU 上实现，分支控制或具有状态机特征的模块划分为软件任务在 CPU 上执行。这种划分模式的缺点显而易见：

收稿日期：2011-10-29；**修回日期**：2011-12-14

作者简介：张亮忠(1985-)，男，硕士研究生，主要研究方向为嵌入式系统分析与设计(370690956@qq.com)；熊选东(1965-)，男，研究员，主要研究方向为系统工程与计算机应用；付建丹(1986-)，男，硕士研究生，主要研究方向为系统工程；王松锋(1987-)，男，硕士研究生，主要研究方向为 Petri 网.

a)设计人员需要预先确定每个软件任务和硬件任务的执行时间,还需要确定每个硬件任务的配置时间和配置位置。

b)软/硬件划分模式中,任务调度算法往往非常复杂[8],需要应用各种启发式调度算法,如遗传算法、模拟退火算法、微粒群优化算法等,而这些调度算法复杂度高,占用了系统大量的时间,却不能保证取得最好的调度效果。

c)当系统中某个任务需要修改时,即便是很小的改动都需要对所有任务重新进行调度。

为此,本文提出将用户应用划分为软件任务和混合任务(hybrid tasks,HbT)两类,其定义如下。

定义 1　软件任务是指系统中的一段可执行代码,由操作系统以软件线程的方式进行调度。

定义 2　混合任务是指同一个任务同时实现了软件和硬件两个版本,操作系统在调度该任务时,首先判断该任务的硬件版本是否在可重构硬件上实现了预配置。若没有预配置,则创建该任务的软件线程,在 CPU 上执行该任务;若已进行了配置,则采用硬件版本由可重构器件来实现该任务。

HbT 既能以二进制代码形式在 CPU 上执行,也能以硬件逻辑电路的形式在 FPGA 上实现。HbT 提供的这种实现灵活性,使得操作系统可以根据任务到达、任务执行、资源占用、系统负载等运行时状态在 CPU 和 FPGA 之间动态地调度任务,与事先静态的任务划分相比,更易于达到系统计算资源的优化利用,并且能提高系统的容错能力。

2　eCos4RC 框架设计

2.1　eCos 的优势分析

由于 eCos 具有众多的优势[9~11],目前国内外许多公司已推出或正在开发基于 eCos 的产品,包括网络和通信产品、工业自动化产品、消费电子、多媒体、办公设备、信息处理、卫星系统、导弹、天文望远镜、地震监测、飞机、机器人等。其优势如下:

a)eCos 提供了普通嵌入式应用中所需要的全部功能,包括设备驱动程序、内存管理、例外处理、标准 C、数学库等。这些功能采用模块化设计,将不同功能的软件分成不同的组件,组件具有可重用性,分别位于系统的不同层次。这种层次结构实现了 eCos 的可配置性、可移植性、兼容性和可扩展性。

b)eCos 系统及其应用程序以特权方式运行,没有用户方式和内核方式之分,在运行时使用了多任务抢占机制,具有最小的中断延迟,支持嵌入式系统所需的所有同步原语,并拥有灵活的调度策略和中断处理机制;eCos 具有非常快捷的引导时间、相对较小的内存要求、非常强的实时特性,是一个适合深度嵌入式应用的开放源代码实时操作系统。

鉴于以上优势,本文选择 eCos 作为可重构操作系统的拓展基础。

2.2　eCos4RC 拓展功能概览

eCos 内核包含了中断和例外处理机制、多线程机制、同步机制、可供选择的多种调度机制、定时机制、计数器等。为实现对可重构系统及以上混合任务模型的支持,eCos4RC 框架如图 2 所示,图中加粗部分为在 eCos 实时内核的基础上进行的拓展。其拓展功能如下:

a)重构控制机制。为实现对可重构计算资源的有效管理,在内核中增加了重构控制机制,该机制采用了预配置策略,将在下一时刻最有可能被调度到的混合任务的硬件配置位流文件提前配置在 RPU 上。

b)混合任务管理机制。为实现对混合任务的管理,增加了一个代理线程创建函数,并对 eCos 的线程数据结构进行了扩充。

c)通信机制。为实现软件任务与混合任务之间的高效通信,对 eCos 的通信机制作了拓展。

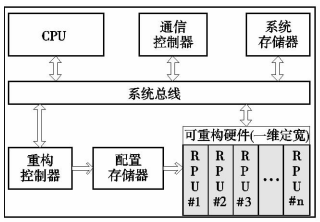

图1　可重构计算平台

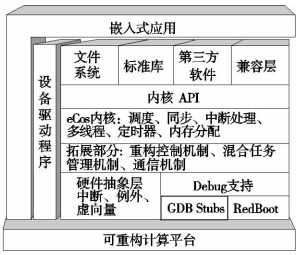

图2　eCos4RC框架

3　重构控制机制

可重构系统在执行硬件任务时,首先要为硬件任务在可重构器件上找到合适的配置位置,进行配置后才能运行。可重构器件的资源抽象模型[12]主要有一维定宽模型、一维变宽模型、二维固定模型、二维可变模型四种。目前,以 Xilinx 公司的 FPGA 为代表的商用动态可重构器件主要支持前三种,一维定宽和二维固定模型本质上是一样的,都是将 FPGA 划分为多个大小固定的 RPU;一维变宽和二维可变模型则根据硬件任务配置文件的大小分配相应大小的 RPU。可并行执行的硬件任务数量取决于可重构计算资源的多少。

eCos 并不支持 RPU 的管理,为了提高可重构资源利用率,eCos4RC 设计了重构控制机制进行 RPU 的统一管理。系统运行时,内核需要对每个混合任务进行引用计数,再按照最近最频繁使用策略或任务相关策略对 RPU 中的配置位流文件进行替换。操作系统需要根据 RPU 上的任务配置情况维护当前配置任务列表(current configuration list,CCL),还需要维护一个循环移位缓存(history_buffer),用于记录系统在最近调用过的混合任务。重构控制机制的具体步骤如下:

a)当用户主线程调用到某个混合任务 HbT_name1 时,首先更新 history_buffer,使该混合任务的引用计数加 1。

b)系统查询 CCL 列表,判断 HbT_name1 是否已配置在 RPU 上,若已配置,则不进行重构操作,若没有配置,则转 c)。

c)系统查询 history_buffer,统计 CCL 中每个已配置的任务的引用次数,选择其中引用次数最少的任务 HbT_name2。

d)若 HbT_name1 的引用次数大于 HbT_name2 的引用次数,则 CPU 向重构控制器发送重构命令,用 HbT_name1 替换 HbT_name2 的配置位流文件。

4　混合任务管理机制

4.1　代理线程

如图 3 所示。在 eCos4RC 中,混合任务有两种实现形式,即软件线程和硬件任务。当用户主线程需要调度某个混合任务时,首先通过可重构资源管理器查询该任务的硬件版本是否已布局在可重构器件上,若没有布局,则创建该任务的软件线程;若已布局,则创建该任务的代理线程(agent thread,AT),每个硬件任务都存在一个 AT 与之对应。AT 作为硬件任务的软件代理,由操作系统内核进行调度和管理,它与硬件任务动态绑定,用于完成对应的硬件任务控制(包括参数初始化、启动、停止等)和实现软件任务与混合任务间的通信,并且记录硬件任务的属性和状态。从内核线程调度器的角度看,AT 本质上是软件线程,因而无须修改内核调度模块,只需要通过如下扩展的线程创建函数、创建 AT。

```
void cyg_at_thread_create
(    cyg_addword_t sched_info, /* scheduling info (priority) */
     cyg_thread_entry_t  * entry, /* thread entry point */
     cyg_addrword_t entry_data, /* entry point argument */
     char    * name, /* name of thread */
```

```
void    * stack_base,/ * pointer to stack base * /
cyg_ucount32 stack_size,/ * size of stack in bytes * /
cyg_handle_t  * handle,/ * returned thread handle * /
cyg_thread  * thread/ * space to store thread data * /
unsigned int bitstream_address/ * bitstream address * /
unsigned int bitstream_length/ * bitstream length * /
unsigned char hardware_task_state/ * state of hardware task * /
)
```

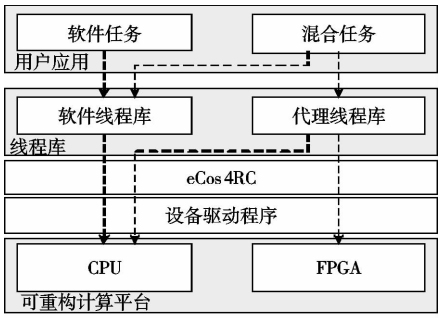

图3 混合任务的实现形式

新产生的 AT 线程分配一个唯一的句柄(即线程 ID),它通过参数 handle 带回。此后所有对该线程的操作都使用该句柄来表示该线程。参数 name 是该线程的名字,对于每一个 AT 线程,内核都为其提供一个小的内存空间,该空间具有 cyg_thread 数据结构的形式,用于存放线程的相关信息(如线程的当前状态),参数 thread 提供了该空间。与软件任务不同,硬件任务是经过综合、布局布线的逻辑电路,以配置位流文件形式被预先存储在 EEPROM(或 flash)中,能够在系统运行时被配置控制器配置到分配的 PRR 上。可并行执行的硬件任务数量取决于可重构计算资源的多少。AT 是一个普通的 eCos 线程,由 eCos 实时内核进行管理和调度。AT 的主要功能是控制对应的硬件任务(包括参数初始化、启动、停止等)和实现任务间的通信。由于 AT 就是一个软件任务,任务间的通信直接采用 eCos 提供的通信机制来实现,对硬件任务的控制是通过通信接口来完成的。在 eCos 的实时调度器看来,AT 与软件线程没有区别,因而不必修改 eCos 的调度程序。

4.2 混合任务的启动、执行与终止

混合任务的配置位流文件由重构控制器根据重构控制机制预先配置到可重构器件上,eCos4RC 在调度混合任务时,若查询到该任务的硬件版本已配置到可重构器件上,则创建其相应的 AT。AT 可以设置硬件任务的参数,并启动硬件任务运行。在硬件任务完成后,AT 重新设置参数,再次启动硬件任务。混合任务的启动、执行与终止如图 4 所示,具体步骤如下:

a)调用重构控制器,判断需要调用的混合任务的硬件配置位流文件是否已配置到可重构器件上。若已配置,转步骤 d)。

b)调用 eCos 的线程创建 API 函数 cyg_thread_create()创建该混合任务的软件线程。

c)操作系统内核的位图调度器或多级队列调度器根据内核提供的标准 API 函数对新创建的线程进行控制,从而完成混合任务的启动、执行和终止。

d)拓展的操作系统内核调用 cyg_at_thread_create()函数创建该混合任务的代理线程 AT,若成功,返回 AT 创建成功标志转步骤 e),若失败,返回 AT 创建失败标志并转步骤 b),改用软件线程执行该混合任务。

e)操作系统内核对 AT 进行调度,通过线程入口函数 thread_entry_function 启动 AT 执行,AT 设置硬件任务的参数,并启动硬件任务运行,设置硬件任务的状态为 ACTIVE。

f)硬件任务处理完毕后,产生中断,通信控制器将结果返回给 AT 或传递给下一个硬件任务,设置硬件任务状态为 WAITING。

g)反复执行步骤 e)f)直到所有数据都处理完毕,设置硬件任务状态为 IDLE。

h)重构控制器根据预配置策略,用其他硬件任务替换该硬件任务;

i)当硬件任务被替换时,产生中断通知操作系统内核调用线程终止函数 cyg_thread_exit()终止该硬件任务相应的代理线程 AT。

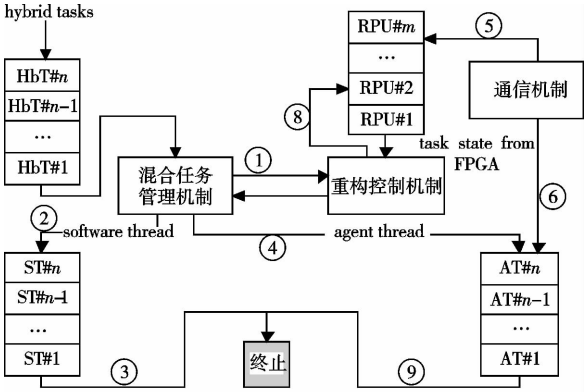

图4 混合任务的启动、执行与终止

上述代理线程 AT 创建过程在扩展的操作系统内核内完成,对设计人员完全透明。在应用开发过程中,与软件线程创建类似,设计人员可通过 cyg_at_thread_create()API 创建 AT。可选择在 AT 创建失败时创建其软件线程,从而提高了系统的容错能力和系统运行时的灵活性。

5 通信机制

统一的通信和同步机制有利于简化可重构操作系统的线程间通信,有效地支持任务的并行。如何在底层通过体系结构支持实现高效数据访问机制,使得开销最小化而不至于失去硬件加速的优势将是任务间通信需要仔细考虑的。在 eCos4RC 中,用户应用由多个软件任务和混合任务组成。在某一时间段内,系统中可以存在多个活动的软件线程和硬件任务的代理线程,多个线程的运行通过可配置的两种调度策略进行调度。与传统的将硬件任务抽象为设备,在设备驱动中提供共享缓冲的通信方式不同。为了在动态可重构环境下提供灵活的软件线程和硬件任务间通信机制,eCos4RC 在调用可重构器件上的硬件任务时,首先创建该硬件任务的代理线程 AT。AT 作为硬件任务的代理,与其他软件线程共享内存地址空间,无须使用繁琐的进程间通信 IPC 和其他复杂的通信机制,只需按照数据生产者——消费者关系在标准 eCos 线程同步机制协调下对共享缓冲区进行访问。设计人员可通过 AT 对硬件任务发送命令,控制硬件任务的运行,也可调用扩展的数据传输 API,由通信控制器实现共享缓冲区与硬件任务之间的数据交互。这样的设计,一方面使硬件任务专注于计算任务处理;另一方面,代理线程的存在使得硬件任务和其他软件线程之间的通信转换为代理线程与其他软件线程间的通信,与 eCos 通信与同步机制兼容。

AT 与硬件任务之间的交互步骤如下:

a)eCos4RC 调用函数 cyg_thread_resume()启动代理线程 AT。

b)AT 通过函数 cyg_data_read()向通信控制器发出读数据命令,该函数同时将数据在内存中基地址 * base 和数据长度 length 传递给通信控制器。

c)通信控制器在接到命令后,通过中断信号取得总线控制权,并在内存中读取数据至硬件任务的输入 FIFO。

d)硬件任务从本地输入 FIFO 读入数据,进入运行态;同时,AT 向通信控制器发出指令 cyg_data_write()或 cyg_data_

transfer( )。前者指示通信控制器将数据处理结果写回到内存;后者指示通信控制器将数据处理结果传递给另一个硬件任务。

e)数据处理完成后,硬件任务将处理结果存入本地输出 FIFO 中。

f)通信控制器根据步骤 d)接收到的指令,将数据处理结果写回内存或传递给另一个硬件任务。

这样的设计兼容 eCos 的通信与同步机制,同时为混合任务模型提供良好的支持,有益于在可重构系统中实现灵活的任务间通信。

6　仿真分析

采用 SystemC 对可重构平台进行仿真,以 AES 数据加密应用为例来评估 eCos4RC 的性能,并与传统的采用软/硬件任务划分模式和设备管理方式的系统(简称为 Sw/Hw System)进行了对比。图 5 所示为建立仿真所用的 SystemC 时间/功能模型。

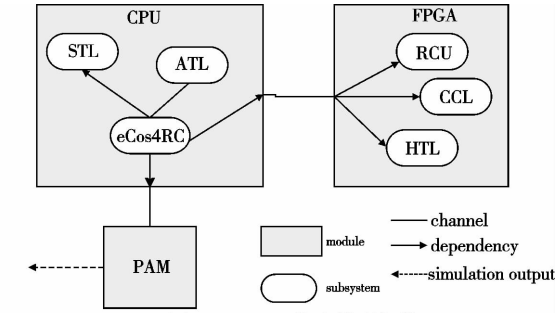

图5　SystemC仿真模型架构

1)CPU 模块　对可重构计算平台上 CPU 的功能进行仿真,它包括 eCos4RC 子系统、软件线程库(software thread library,STL)和硬件任务代理线程库(agent thread library,ATL)。前者主要是对系统中的软件任务和混合任务之间的依赖关系进行模拟,判定混合任务的执行方式,确定任务对应线程的创建时间;后两者用于仿真各个软件线程和代理线程的创建和执行过程。

2)FPGA 模块　主要对 FPGA 的功能进行仿真,它包括重构控制器(reconfiguration control unit,RCU)、当前配置列表(current configuration list,CCL)、硬件任务库(hardware tasks library,HTL)三个子系统。RCU 负责运行预配置调度算法,以确定需要配置的硬件任务,更新 CCL;CCL 则负责记录当前实现配置的硬件任务,CPU 模块在调度混合任务时,首先查询 CCL 检测混合任务是否实现了预配置,从而选择混合任务的执行方式;HTL 用于模拟硬件任务的执行过程。

3)PAM 模块(performance analyzing module)　性能分析模块根据本文提出的线程管理和通信机制,对系统性能进行分析。

AES 加密应用采用了文献[13]的模块划分方法,AES 加密应用中的混合任务如表 1 所示,其中所用到的时间参数都是根据对实际运行在 FPGA 上的信号处理应用分析所得。软件线程和代理线程的创建时间均为 19 μs,软件线程之间的通信时间为 24 μs,软件线程与硬件任务的通信时间为 32 μs,硬件任务之间的通信时间为 14 μs。

表 1　混合任务的时间参数

混合任务编号	软件实现时的执行时间/μs	硬件实现时的执行时间/μs	任务说明
1	3 945	748	S 盒变换
2	1 567	392	行变换
3	1 880	360	列混合
4	1 216	205	Key Expansion
5	2 639	522	ADD Round Key

图 6 为在可重构资源分区数量为 1 的情况下,AES 的一轮加密所花费的执行时间(execution time,ET)与硬件任务在可重构器件上的重构时间(reconfiguration time,RT)间的关系。从图中可以看到,在 RT 较小时,Sw/Hw System 具有一定的优势,但随着 RT 的增加,Sw/Hw System 所花费的执行时间增长较快。而本文通过 eCos4RC 对混合任务的有效管理,应用程序的执行时间受重构时间的影响很小,由此可见,eCos4RC 通过任务执行方式的灵活选择能很好地隐藏硬件任务的配置时间。图 6 还展示了采用纯软件(Sw System)实现时的执行时间,其不受 RT 影响。图 7 为在硬件任务配置时间为 500 μs 的情况下,AES 的一轮加密所花费的执行时间与 RPU 数量的关系。从图中可以看到,在 RPU 数量越少时,eCos4RC 的优势越明显,当 RPU 数量为 5 时,所有硬件任务都实现了预配置,eCos4RC 与 Sw/Hw System 的执行时间趋于相等。这表明,采用 eCos4RC 可以有效地提高可重构资源的利用率。

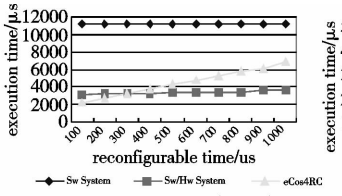

图6　RT对ET的影响

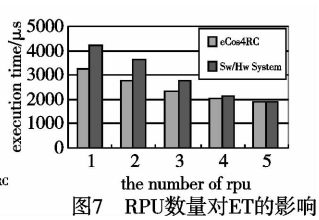

图7　RPU数量对ET的影响

7　结束语

本文的仿真结果表明,eCos4RC 可显著改善硬件任务配置时间较长的状况;线程间通信机制简单有效;在探索编程灵活性的同时兼顾了硬件的效率,能够较好地支持可重构应用的开发。

参考文献:

[1] LAGGER A, UPEGUI A, SANEHEZ E, *et al*. Self-reconfigurable pervasive platform for cryptographic application[C]//Proc of International Conference on Field Programmable Logic and Applications. 2006:1-4.

[2] De BOLE M. Configurable accelerators for video analytics[D]. Pennsylvania:Pennsylvania State University,2011.

[3] 马宏星,周学海,高妍妍. 可重构计算平台上软/硬件任务划分与调度算法[J]. 系统工程与电子技术,2010,32(11):2459-2464.

[4] LUBBERS E,PLATZNER M. A portable abstraction layer for hardware threads[C]//Proc of International Conference on Field Programmable Logic and Applications. 2008:17-22.

[5] 周学海,罗赛,王峰,等. 一种数据驱动的可重构计算统一编程模型[J]. 电子学报,2007,35(11):2123-2128.

[6] 李涛,杨恩鲁,马平,等. 基于遗传算法的可重构系统软硬件划分[J]. 计算机工程与应用,2007,43(26):56-58.

[7] 邵岁锋,张英杰. 基于免疫粒子群的嵌入式系统软硬件划分方法[J]. 计算机应用,2010,30(2):479-481.

[8] TELLER J S. Scheduling tasks on heterogeneous chip multiprocessors with reconfigurable hardware[D]. Ohio:Ohio State University, 2008.

[9] 蒋句平. 嵌入式可配置实时操作系统 eCos 开发与应用[M]. 北京:机械工业出版社,2008.

[10] MASSA A. Embedded software development with eCos[M]. [S. l.]: Prentice Hall PTR,2002.

[11] eCos discuss[EB/OL]. http://ecos. sourceware. org/ml/ecos-discuss.

[12] WANG Fei. A field programmable gate array architecture for two-dimensional partial recon-figuration[D]. Dayton:Wright State University, 2006.

[13] SOWRIRAJAN S. FPGA based hardware implementation of advanced encryption standard[D]. Dayton:Wright State University, 2007.