

异构可重构计算系统的 Petri 网模型

张妮妮¹, 郭 军²

(1. 西安思源学院, 陕西 西安 710038;

2. 西北大学, 陕西 西安 710069)

摘要: 构建系统描述模型是设计可重构计算系统的重要环节。现有的系统建模方法主要分为形式化方法和非形式化方法两种, 其中, 非形式化模型缺乏严格的数学定义, 给模型的分析 and 验证带来困难。采用形式化方法建立的系统模型无歧义, 更适合分析和验证。Petri 网作为一种常用的形式化建模方法, 有严格的数学定义和建模理论, 但是, 基本 Petri 网在对可重构计算系统建模时存在数据流描述能力不足的问题。为此, 对基本 Petri 网进行扩展, 提出了一种数据流 Petri 网。首先给出了数据流 Petri 网的结构定义和动态行为规则, 并定义了模型的图形符号表示方法。然后, 分析了模型对可重构计算配置任务和计算任务的描述方法, 在考虑硬件资源约束条件下, 讨论了可重构计算系统的数据流 Petri 网建模技术。最后, 通过一个典型的乘加运算器模型分析, 表明该方法易于实现, 所建模型结构简洁, 便于分析验证系统功能。

关键词: 可重构计算; 异构系统; Petri 网; 形式化模型

中图分类号: TP311

文献标识码: A

文章编号: 1673-629X(2018)09-0112-06

doi:10.3969/j.issn.1673-629X.2018.09.023

A Petri Net Model for Heterogeneous Reconfigurable Computing Systems

ZHANG Wei-wei¹, GUO Jun²

(1. Xi'an Siyuan University, Xi'an 710038, China;

2. Northwest University, Xi'an 710069, China)

Abstract: The system description model plays an important role in the design of reconfigurable computing systems. The existing models can be divided into two categories: formal and informal methods. As informal models lack rigorous mathematical definitions, it is difficult to analyze and verify the functions of a system by informal models. On the contrary, formal models are unambiguous and suitable for functional verification. Petri nets is a commonly used formal method with rigorous mathematical definitions and modeling theory. But the basic Petri net could not meet the requirement of modeling reconfigurable computing systems without the ability of describing data flow. Thus we extend the basic Petri net and propose a kind of data flow Petri net (DPN). Firstly, we define the structure and firing rules of DPN mathematically. The graphic symbols of DPN are also defined. And then we introduce the DPN models of configurable tasks and computing tasks for reconfigurable computing systems. At the same time, the techniques for modeling reconfigurable computing systems by DPN are discussed with the consideration of hardware resource constrain. Finally, as an instance, a multiply-adder is analyzed by DPN models. The result indicates that the DPN model is simple to construct, easy to understand and convenient for functional verification.

Key words: reconfigurable computing; heterogeneous system; Petri net; formal model

0 引言

随着 IC 芯片制造工艺越来越接近物理极限, 传统的冯·诺依曼结构计算机的性能提高遇到了瓶颈。可重构计算(reconfigurable computing, RC)作为一种新的高性能计算模式, 是解决性能瓶颈问题的有效方法。可重构计算系统将密集运算(intensive computation)任

务交给可重构硬件实现, 利用硬件运算的高速性和并行性, 大幅度提高系统的运算能力。目前, 可重构计算技术在图像处理、生物计算、密码算法、多媒体等高性能计算领域获得了成功应用^[1-6]。

由于可重构计算系统的功能是通过软件和硬件共同实现的, 其设计方法不同于传统计算系统, 需要采用

收稿日期: 2017-09-09

修回日期: 2018-01-11

网络出版时间: 2018-04-28

基金项目: 陕西省教育自然科学基金(17JK1073); 陕西省自然科学基金基础研究计划资助项目(2017JM6056)

作者简介: 张妮妮(1978-), 女, 硕士, 副教授, CCF 会员, 研究方向为计算机应用技术。

网络出版地址: <http://kns.cnki.net/kcms/detail/61.1450.TP.20180427.1630.038.html>

软硬件协同的设计方法。设计过程首先是根据系统规格说明建立系统功能在可重构平台上的描述模型,确定计算任务和相应算法,然后将计算任务有效地映射到计算平台的软硬件组件上。因此,选择合适的系统描述模型是可重构系统设计过程至关重要的一步。

目前,计算系统描述模型主要分为形式化模型和非形式化模型。非形式化模型一般无需严格定义,采用比较简洁的符号表达,易于理解和实现。常用的非形式化模型主要有数据流/控制流图、流程图和时序状态机等。文献[7-8]采用有向无环图(DAG)描述可重构系统任务,简洁易懂,是一种常用的非形式化模型。文献[9-11]采用非形式化模型来解决系统设计过程中的问题。但是,由于非形式化模型缺乏严谨的数学定义和理论支持,容易产生歧义,给模型分析验证带来了困难。与此相反,形式化模型通常有严格的数学定义和理论支持,建模方法具有无歧义、便于分析验证等特点,在可重构计算系统的描述中比非形式化方法更有优势。常用的形式化模型主要有图灵机、Petri网、共代数(co-algebras)、 π 演算(π -calculus)、时间自动机和UML模型等。文献[12]研究了共代数方法描述可重构计算系统,给出了相关定义,归纳出了基本的重构操作,但模型比较复杂,需要较好的代数知识才能理解应用。文献[13]提出 π 演算的变形 φ 演算(φ -calculus)方法描述可重构计算系统,利用了 π 演算描述动态变化的优势,但给出的模型定义比较抽象,也需要相关数学知识才能理解。 π 演算和 φ 演算模型涉及的数学知识比较复杂,实际应用不够方便。文献[14]在 π 演算的基础上提出了一种 π 语言,能够对可重构系统进行形式化描述,并产生实用代码,但是该方法需要专门的编译器,应用的便利性还需要实践检验。文献[15]采用自动机与UML结合的模式验证方法,虽然UML模型能够提供系统的多视图描述模型,模型结构简洁易懂,便于实现,但是UML模型本身是半形式化模型,其描述符号定义不够严谨,模型不支持动态运行,难以验证系统的动态行为。

Petri网作为一种传统的形式化建模工具,理论完善,尤其适合描述并发、异步、资源冲突的系统^[16-17],符合可重构计算系统的特点;Petri网模型又是一种可运行模型,具有动态分析系统资源分配、验证系统功能的能力,这也是可重构计算系统设计关心的问题。尽管Petri网模型在软硬件设计中已经得到成功应用,但是,Petri网模型应用于可重构计算系统建模分析时还存在数据流描述能力不足的问题。因此,针对传统Petri网模型的不足进行扩展,提出一种数据流Petri网模型,使之满足可重构计算系统建模需求,并讨论相关的建模技术和动态分析技术。

1 可重构计算系统

实际应用的可重构计算系统多为异构系统,通常是由CPU(中央处理器)、GPU(图形处理器)、FPGA(可编程门阵列)、DSP(数字信号处理器)等部件通过高速网络或接口以一定的耦合方式组成的并行计算系统。针对不同应用,系统结构也不相同,典型的可重构计算系统如图1所示,由CPU+FPGA等组成^[1]。其中处理器可以是通用CPU,如Intel、AMD的处理器,或者是ARM、POWER PC等嵌入式处理器。处理器主要执行控制任务和一些非密集计算任务,FPGA主要完成密集计算任务,其中FPGA的作用更像一个协处理器。著名的Cray XD1、SRC-6和SGI Altix等可重构计算机都采用类似的体系结构。

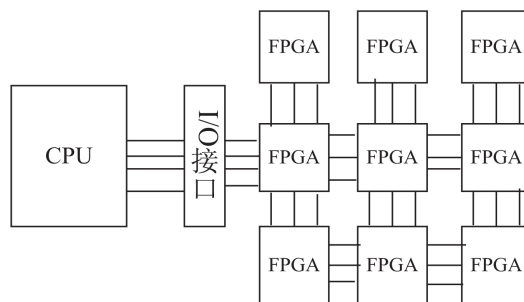


图1 可重构计算系统结构

一个具体的应用在可重构平台上的运行过程是配置硬件和运算操作的交替。即首先根据具体应用,将FPGA配置成特定功能的逻辑电路,通常是IP核,建立数据通路;其次是将数据输入逻辑电路接口,由逻辑电路完成计算;最后,把运算结果保留到寄存器/缓存。如果经过一次配置,应用任务尚未完成,就需要再一次配置硬件进行运算,直到得到最终结果。对于复杂的应用,往往需要经过多次重复,如图2所示。由图可见,在可重构计算平台上,应用是在时间和空间进行的。在时间方向,完成一次一次的运算,在 $x-y$ 空间方向,完成一次一次的配置。因此,在可重构系统中,计算任务不仅在时间延续,而且在空间展开,由于空间展开,并行任务可以实现真正意义上的时间并行,从而大大提高了系统性能。

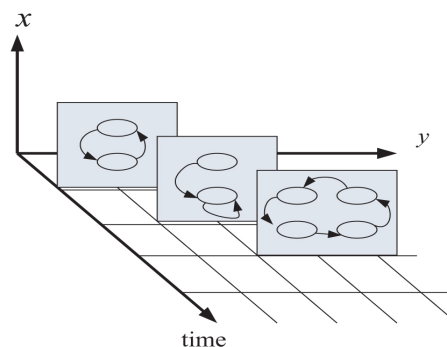


图2 任务执行过程

从以上分析可以看出,不同的应用在可重构系统上的执行过程是配置操作和运算操作的交替往复。通过不同的硬件配置,到达优化的系统结构,在优化的配置下,进行高性能的运算。如果将每次配置定义为配置件 (configure ware), 将每个配置下的数据运算称为数据流件 (dataflow ware), 那么,一个应用可以抽象为若干配置件和数据流件组成的有序序列,这个序列应该能够满足系统的硬件资源、运行时间等约束条件。对于相互独立的并行的应用任务,可以同时提供硬件配置,实现硬件意义上的真并行。因此,系统的描述模型,应该能够描述可重构系统这种资源约束、并行和有序的应用特征, Petri 网模型就具备这些描述能力。

2 数据流 Petri 网

Petri 网适合描述并发、异步、资源冲突的系统,可以图形表示,直观易懂,便于分析模拟。但是,一般 Petri 网以描述控制流见长,描述数据流的能力较弱,为此,提出了着色网和双变迁网等扩展形式,以增强 Petri 网的数据处理能力。文中将根据可重构系统的特点,借鉴着色网和双变迁网的思想,扩展定义一种简明易用的数据流 Petri 网。

2.1 数据流 Petri 网基本定义

定义 1: 数据流 Petri 网定义为一个 7 元组 $DPN = (P, T, S, Q, F, W, M^0)$, 其中 $T \cap P \cap S \cap Q = \emptyset$ 。 P 是有限的配置库所集合,在可重构系统中用来表示占用资源的配置件; T 是有限配置变迁集合,在可重构系统中用来表示配置操作; S 是数据库所集合,在可重构系统中用来表示数据存储单元,代表存储器或寄存器等; Q 是数据变迁集合,在可重构系统中用来表示配置件所实现的运算功能; $F = F_c \cup F_d$ 是流关系集合, F_c 是配置流集合, $F_c \subseteq (P \times T) \times (T \times P) \times (Q \times T) \times (T \times Q)$, 在可重构系统中用来表示资源的流向, F_d 是数据流集合, $F_d \subseteq (S \times Q) \cup (Q \times S)$, 在可重构系统中用来表示数据流; $W: F_c \rightarrow \{1, 2, \dots\}$ 是定义在配置流弧上的权值,在可重构系统中用来表示资源的使用数量,省缺值为 1; 标识 $M = M_c \cup M_d$, 其中 $M_c: P \rightarrow \{0, 1, 2, \dots\}$ 称为配置标识,通常称为令牌,反映了系统资源分配的状态,在模块化可重构系统中表示配置件需要的模块数量,初始标识为 M_c^0 , $M_d: S \rightarrow \{0, 1\}$ 称为数据标识,表示存储单元是否有等待处理的数据, 0 表示无数据, 1 表示有一批数据,初始标识为 M_d^0 。

DPN 可以用图形符号直观表示,图 3 给出了图形符号的基本表示形式。图中,细圆圈表示配置库所,圈中非负整数表示配置标识;粗圆圈表示数据库所,其中的数字是数据标识;短线条表示配置变迁,矩形表示数

据变迁;带箭头的弧线表示流关系,细弧表示配置流,粗弧表示数据流,权值标注在弧上。图中初始标识为 $M_c(p_1) = 10, M_d(s_1) = 1, M_d(s_2) = 1$, 其余为 0。

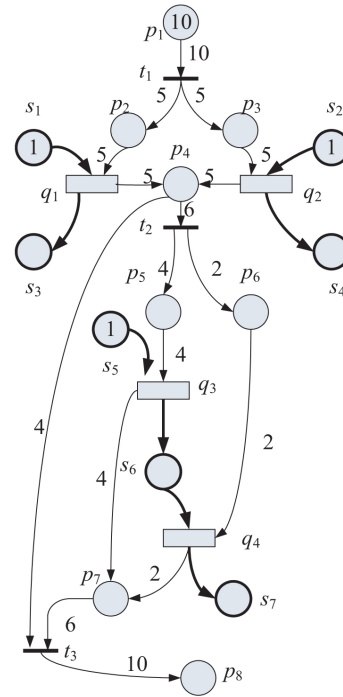


图 3 DPN 图形表示

以上给出的是 Petri 网的静态结构,通过定义变迁激发条件,可以得到 Petri 网的动态执行特性,从而研究系统状态的变化过程。

定义 2: 对于 DPN, $\forall x \in P \cup T \cup Q$ 则

$$\bullet x = \{y \mid y \in P \cup T \cup Q \wedge (y, x) \in F_c\}$$

$$x \bullet = \{y \mid y \in P \cup T \cup Q \wedge (x, y) \in F_c\}$$

称 $\bullet x$ 为 x 的配置前集, $x \bullet$ 为 x 的配置后集。

定义 3: 对于 DPN, $\forall x \in Q \cup S$ 则

$${}^\circ x = \{y \mid y \in S \cup Q \wedge (y, x) \in F_d\}$$

$$x^\circ = \{y \mid y \in S \cup Q \wedge (x, y) \in F_d\}$$

称 ${}^\circ x$ 为 x 的数据前集, x° 为 x 的数据后集。

定义 4: 数据变迁函数 $f(q): {}^\circ q \rightarrow q^\circ, q \in Q$ 描述数据变迁完成的操作。

定义 5: 如果 $t \in T, \forall p \in \bullet t$, 有 $M_c(p) \geq W(p, t)$, 则配置变迁 t 使能或有发生权; 如果 $q \in Q, \forall p \in \bullet q$, 有 $M_c(p) \geq W(p, q)$, 且 $\forall s \in {}^\circ q, M_d(s) = 1$, 则数据变迁 q 使能或有发生权。

定义 6: 在标识 M_c, M_d 下, 使能的配置变迁 t 激发后 M_d 不变, 产生新的标识 M'_c 。

$$M'_c(p) = \begin{cases} M_c(p) - W(p, t) & p \in \bullet t - t \bullet \\ M_c(p) + W(t, p) & p \in t \bullet - \bullet t \\ M_c(p) & \text{other cases} \end{cases}$$

定义 7: 在标识 M_c, M_d 下, 使能的数据变迁 q 激发后, $\forall s \in {}^\circ q$, 则 $M_d(s) = 0, \forall s \in q^\circ$, 则 $M_d(s) = 1$;

并产生新的配置标识 M'_c 。

$$M'_c(p) = \begin{cases} M_c(p) - W(p, q) & p \in \bullet q - q \bullet \\ M_c(p) + W(q, p) & p \in q \bullet - \bullet q \\ M_c(p) & \text{other cases} \end{cases}$$

数据流 Petri 网扩展定义了数据库所,数据可以有不同的类型,这一点类似于着色 Petri 网。但是,由于进一步定义了数据变迁,其数据处理能力强于着色网。而且,着色 Petri 网控制流与数据流是不加以区分的,而数据流 Petri 网将数据流与控制流明显区分开,既是为了描述可重构计算系统的需要,也是不同于着色网的显著特征。

2.2 数据流 Petri 网运行过程分析

对于图 3 描述的系统,初始标识为 $M_c(p_1) = 10$, 这时,根据定义 5,只有 t_1 满足使能条件, t_1 激发后,引起系统状态变化, p_2 、 p_3 各得到 5 个配置资源,即完成了一次配置,如图 4(a);在该配置下,数据变迁 q_1 、 q_2 满足使能条件,且数据库所前集不为空, q_1 、 q_2 可以并发执行,完成数据操作,如图 4(b),需要注意的是, q_1 、 q_2 不一定同时执行,可以不同步;激发后,配置资源全部释放给 p_4 , p_4 得到 10 个配置资源,如图 4(c),从而完成了一次配置/计算任务;接下来,配置变迁 t_2 满足使能条件,激发后 p_5 、 p_6 分别得到 4 个和 2 个令牌资

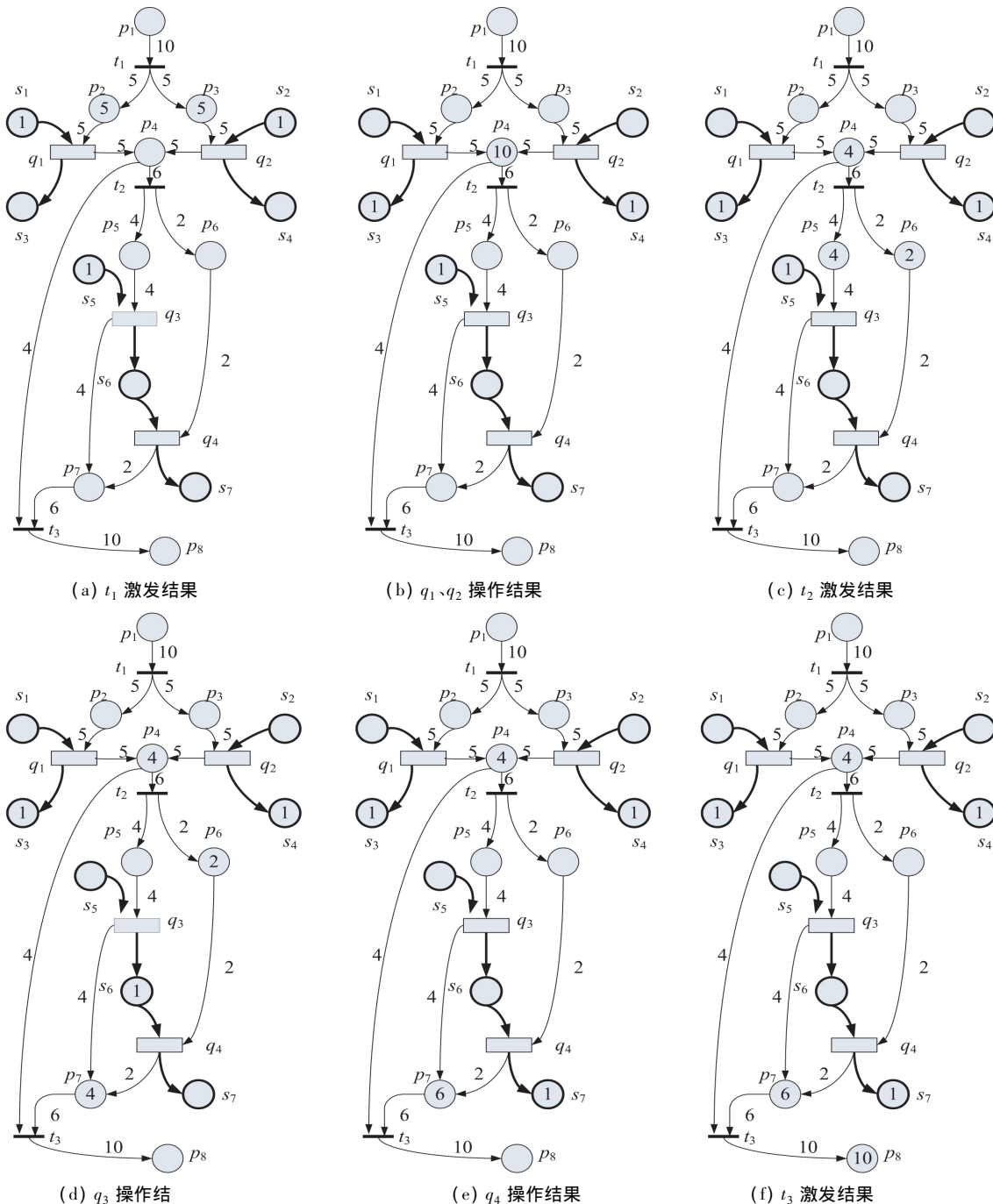


图 4 模型运行过程

源,完成第 2 次配置,如图 4(d),在该配置下, q_3 首先满足使能条件,激发后释放 4 个令牌资源给 p_7 ,并传递结果数据到 s_6 ,从而 q_4 满足使能条件激发,释放 2 个令牌资源到 p_7 ,运算结果保留到 s_7 ,如图 4(e);由于第二次配置资源并没有使用完,剩余的资源通过变迁 t_3 ,传递给 p_8 ,变化结果如图 4(f)。

可见,图 3 的模型描述了 2 次配置操作 t_1 、 t_2 ,在完成 t_1 配置后,进行 2 个可并行数据操作 q_1 、 q_2 ;在完成 t_2 配置后,进行 2 个串行数据操作 q_3 、 q_4 。不同配置下输入的数据可以不相关,也可以相关。

模型对于数据操作采用批处理的概念,一次数据变迁处理一批数据,这样可以避免因多次数据操作造成的复杂控制过程,而且,现有的硬件堆栈和 DMA 技术都能够支持这种操作。

如果需要分析系统时间约束,可以进一步定义变迁激发时间,扩展成时间 Petri 网,但是,变迁激发条件需要重新定义;通过定义 DPN 的标识向量和关联矩阵,分析可达路径和标识变化,可以形式化表示和分析系统。鉴于篇幅所限,下面仅对模型的有界性、可达性和活性进行简要说明。

有界性是指对于 $\forall p \in P$,配置标识 $M_c(p) \leq M_{\max}$, M_{\max} 是系统可提供的最大资源数量;对于 $\forall s \in S$,数据标识 $M_d(s) \leq 1$;故 DPN 模型是有界的。对于模块化可重构计算系统,可重构资源模块是一定的,配置过程实际上就是把资源分配给不同的配置件,因此,无论何时,系统总的配置标识数量是不变的,资源数量是守恒的,这样才能保障系统的安全性。利用 DPN 模型,可以很好地分析系统的有界性。

可达性描述了模型从状态标识 M_i 经过一个变迁系列,能够到达新状态标识 M_j 。就可重构系统应用而言,模型的可达性是至关重要的,决定了系统是否能够实现某个配置,达到相应的系统状态。由于大模型的状态空间很大,分析可达状态空间是一个困难的问题。通过模型运行,可以分析所关心的状态是否可达,例如在图 4 的模型分析中,两次配置状态都是可达的。

活性是指变迁 t 在某一标识下是否总能够使能激发,也就是说,系统不会出现死锁。这一点对可重构系统来说也是十分重要的,因为配置资源时,要避免资源竞争可能造成的死锁。通过模型运行,可以分析发现系统可能出现的死锁现象,图 4 的分析表明示例模型不存在死锁。

3 设计实例

乘加运算是矩阵运算的主要操作,而矩阵运算是一种典型的密集计算任务,在图形图像、信号处理中应用广泛^[18]。 n 阶方阵乘法包含的乘法操作和加法操作

数都在 n^3 量级,而在图像处理应用中,上百阶的矩阵普遍存在,需要完成的乘法操作和加法操作数量十分巨大。这些运算操作大部分是可并行的,但是,在非并行体系结构上只能采用软件技术实现串行操作,在可重构计算平台上则可以实现并行运算。下面就以矩阵乘法为例,采用 DPN 模型描述任务的实现过程。

矩阵乘法运算的核心是乘法和加法操作,因此,配置任务主要是完成乘法器和加法器的配置。为了说明问题,这里仅以简单的 3×3 矩阵为例,建立 DPN 描述模型。

矩阵乘法可以采用并行或串行算法。这里以并行乘、串行加为例,研究 DPN 描述不同结构的技术方法。如图 5 所示,初始标识: $M_c(p_1) = 6$, $M_d(s_1) = 1$, $M_c(s_2) = 1$, $M_c(s_3) = 1$, $M_c(s_4) = 1$, $M_c(s_5) = 1$, $M_c(s_6) = 1$,其他库所标识均为 0。通过分析模型的运行过程,可以得到系统状态的变化情况,从而验证是否满足功能需求以及资源约束条件。模型运行时, t_1 变迁首先使能激发,分配给库所 p_2 、 p_3 、 p_4 各 2 个资源,配置 3 个并行乘法器,由数据变迁 q_1 、 q_2 、 q_3 表示;第 2 次配置由 t_2 实现,配置两个串行加法器 q_4 和 q_5 ,第 2 次配置与第 1 次配置存在数据相关,即 s_8 和 s_9 中上次配置运算结果先行计算,中间结果再与 s_7 中数据相加,最终结果存入 s_{11} ,资源全部释放到 p_8 。

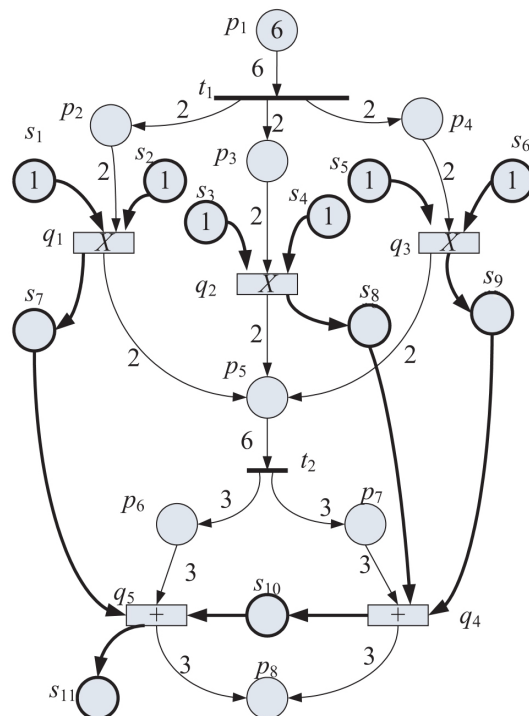


图 5 乘加运算的 DPN 描述模型

4 结束语

采用 Petri 网等形式化方法描述可重构计算系统,有助于设计者较早地分析系统的功能和发现设计存在

的问题,同时,也为各个设计阶段提供了一个统一的系统级抽象模型,保证模型的一致性。通过定义数据流 Petri 网,文中提出一种针对可重构计算系统的建模方法,描述可重构计算中存在的配置流和数据流;实际建模表明,该方法所建模型结构简单,分析过程直观易懂,理论方法容易掌握,能够描述资源约束下可重构计算系统中的主要特征参数,是一种实用性的可重构计算系统建模方法。

参考文献:

- [1] TODMAN T J, CONSTANTINIDES G A, WILTON S J E, et al. Reconfigurable computing: architectures and design methods [J]. IEE Proceedings: Computers and Digital Techniques, 2005, 152(2): 193-207.
- [2] GALANIS M D, THEODORIDIS G, TRAGOUDAS S, et al. Mapping computational intensive applications to a new coarse-grained reconfigurable data-path [C]//14th international PATMOS workshop. Greece: [s.n.], 2004: 652-661.
- [3] 王峰, 周学海, 陈艾, 等. 基于部分重构技术的加密算法实现研究[J]. 电子学报, 2007, 35(5): 959-963.
- [4] WANG Xu, ZHU Yongxin, HUANG Linan. A comprehensive reconfigurable computing approach to memory wall problem of large graph computation [J]. Journal of Systems Architecture, 2016, 70: 59-69.
- [5] 张宇, 范建华, 吕遵明, 等. FPGA 动态部分可重构技术概述[J]. 计算机与现代化, 2014(3): 210-214.
- [6] 刘杰, 吴强, 赵全伟. 面向可重构计算系统的模块映射算法[J]. 计算机工程, 2012, 38(3): 276-279.
- [7] 郝水侠, 曾国荪, 谭一鸣. 一种基于 DAG 图的异构可重构任务划分方法[J]. 同济大学学报: 自然科学版, 2011, 39(11): 1693-1698.
- [8] RAMEZANI R, SEADAGHAT Y, NAGHIBZADEH M, et al. Reliability and make span optimization of hardware task graphs in partially reconfigurable platforms [J]. IEEE Transactions on Aerospace and Electronic Systems, 2017, 53(2): 983-994.
- [9] MISHRA A, AGARWAL M, ASATI A R, et al. Using graph isomorphism for mapping of data flow applications on reconfigurable computing systems [J]. Microprocessors and Microsystems, 2017, 51(6): 343-355.
- [10] LIU Tawei, LIU Yen-Fang, CHEN Yashu. Energy-aware run-time task partition and allocation in dynamic partial reconfigurable systems [J]. Journal of Systems Architecture, 2017, 78: 55-67.
- [11] MEHRI H, ALIZADEH B. Analytical performance model for FPGA-based reconfigurable computing [J]. Microprocessors and Microsystems, 2015, 39(8): 796-806.
- [12] CONG-VINH P, BOWEN J P. A formal approach to aspect-oriented modular reconfigurable computing [C]//IEEE/IFIP symposium on theoretical aspects of software engineering. Shanghai, China: IEEE, 2007: 369-378.
- [13] ROUNDSW C, SONG H. The ϕ -calculus: a language for distributed control of reconfigurable embedded systems [C]//International workshop on hybrid systems: computation and control. Prague, Czech Republic: Springer-Verlag, 2003: 435-449.
- [14] UL-ABDIN Z, SVENSSON B. Retargetable compilation framework for heterogeneous reconfigurable computing [J]. ACM Transactions on Reconfigurable Technology & Systems, 2016, 9(4): 2401-2422.
- [15] 陈卫涛. 动态可重构系统形式化验证工具与原型平台的设计与实现[D]. 沈阳: 东北大学, 2010.
- [16] 蒋昌俊. Petri 网理论与方法研究综述[J]. 控制与决策, 1997, 12(6): 631-636.
- [17] ABELLARD A. Architectural Petri nets: basic concepts, methodology and examples of applications [C]//IEEE international conference on systems, man and cybernetics. Waikoloa, HI, USA: IEEE, 2005: 2037-2042.
- [18] 张禾, 陈客松. 基于 FPGA 的稀疏矩阵向量乘的设计研究[J]. 计算机应用研究, 2014, 31(6): 1756-1759.