

# 一种基于离散 Hopfield 神经网络的 RTOS 功耗优化方法

郭 兵<sup>1)</sup> 沈 艳<sup>2)</sup> 王殿辉<sup>3)</sup> 李志蜀<sup>1)</sup> 陈向东<sup>4)</sup>

<sup>1)</sup>(四川大学计算机学院 成都 610065)

<sup>2)</sup>(电子科技大学机械电子工程学院 成都 610054)

<sup>3)</sup>(拉筹伯大学计算机科学与工程系 墨尔本 VIC 3086 澳大利亚)

<sup>4)</sup>(西南交通大学信息科学与技术学院 成都 610031)

**摘 要** RTOS(Real-Time Operating System, 实时操作系统)是 SoC(System-on-a-Chip, 系统芯片或片上系统)的一个重要组成部分, 其功耗一般约占整个系统功耗 30~40% 的比例, 而基于软/硬件划分的 RTOS 功耗优化方法(简称 RTOS-Power 划分)能够明显地减少 SoC 的功耗. 因此, 文中首先引入了 RTOS-Power 划分问题的一个新模型, 这有助于理解 RTOS-Power 划分的本质. 然后, 提出了一种基于离散 Hopfield 神经网络的 RTOS-Power 划分方法, 重新定义了神经网络的神经元表示、能量函数、运行方程和系数. 最后, 对该方法进行了仿真实验, 并同遗传算法和蚂蚁算法进行了性能比较. 实验结果表明: 该文提出的方法能够以相对较小的代价(FPGA 开销小于 4K 个可编程逻辑块)取得高达 60% 的功耗节省, 同时, 与纯软件实现的 RTOS 相比, 系统性能也得到了相应的提高.

**关键词** Hopfield 神经网络; 功耗优化; RTOS; 软/硬件划分; SoC

中图法分类号 TP316

## A Power Optimization Approach to Real-Time Operating Systems Based on Discrete Hopfield Neural Networks

GUO Bing<sup>1)</sup> SHEN Yan<sup>2)</sup> WANG Dian-Hui<sup>3)</sup> LI Zhi-Shu<sup>1)</sup> CHEN Xiang-Dong<sup>4)</sup>

<sup>1)</sup>(School of Computer Science & Engineering, Sichuan University, Chengdu 610065)

<sup>2)</sup>(School of Mechatronics Engineering, University of Electronic Science & Technology of China, Chengdu 610054)

<sup>3)</sup>(Department of Computer Science & Engineering, La Trobe University, Melbourne, VIC 3086 Australia)

<sup>4)</sup>(School of Information Science & Technology, Southwest Jiaotong University, Chengdu 610031)

**Abstract** The RTOS(Real-Time Operating System) is a critical component in the SoC(System-on-a-Chip), which consumes the 30~40% of total system energy in average. Power optimization based on hardware-software partitioning of a RTOS(RTOS-Power partitioning) can significantly reduce the energy consumption of a SoC. This paper presents a new model for RTOS-Power partitioning, which helps in understanding the essence of the RTOS-Power partitioning techniques. A discrete Hopfield neural network approach for implementing the RTOS-Power partitioning is proposed, where a novel neuron expression, energy function, operating equation and coefficients of the neural network are redefined. Simulations are carried out with comparison to generic algorithm and ant algorithm. Experimental results demonstrate that the proposed method can achieve higher energy savings up to 60% at relatively low costs of less than 4K PLBs while increasing the performance compared to the SoC-RTOS realized purely in software.

**Keywords** Hopfield neural network; power optimization; RTOS; hardware-software partitioning; SoC

收稿日期: 2006-04-04; 最终修改稿收到日期: 2007-06-15. 本课题得到国家自然科学基金(60572026)资助. 郭 兵, 男, 1970 年生, 博士后, 副教授, 主要研究方向为嵌入式实时系统、SoC 和中间件. E-mail: guobing@sohu.com; guobing@scu.edu.cn. 沈 艳, 女, 1973 年生, 博士, 副教授, 主要研究方向为分布式测量系统、嵌入式系统开发、无线传感器网络和机器人. 王殿辉, 男, 1963 年生, 博士, 副教授, 主要研究方向为计算智能、生物医学的数据挖掘和多媒体信息处理. 李志蜀, 男, 1946 年生, 教授, 博士生导师, 主要研究领域为智能系统和医药信息处理. 陈向东, 男, 1967 年生, 博士, 教授, 博士生导师, 主要研究兴趣为 SoC、无线传感器网络.

## 1 引言

随着全球能源危机的不断发展,数量庞大的嵌入式系统的功耗问题日益引起人们的关注.作为嵌入式系统的一种新形式,SoC(System-on-a-Chip,片上系统或系统芯片)在单个 IC(Integrated Chip,集成电路芯片)里基本上实现了一个完整计算机系统的软/硬件功能.同传统的嵌入式系统相比,SoC 具有许多明显的优点,如体积小、功耗低、可靠性高以及更高的性价比等,缺点是复杂性上升、设计成本高、开发周期长,完全改变了先前整机系统的总体设计架构. SoC 通常是一个定制的 IC,一般由通用的微处理器核、可逻辑编程的硬件单元、应用相关的 I/O 接口电路和相应的嵌入式软件组成. SoC 设计结合了传统的 IC(包括数字、模拟和射频集成电路)设计和嵌入式软件开发两方面的内容,是学术界的研究热点和工业界大力推广应用的一项新技术.

在 SoC 中,嵌入式软件一般由 RTOS(Real-Time Operating System,实时操作系统)和嵌入式应用程序组成,其中 SoC 中的 RTOS 简称 SoC-RTOS. 嵌入式软件的执行驱动了低层微处理器、存储器和 I/O 接口等硬件的电路活动,导致了系统功耗的产生. 先前的许多研究表明,软件的使用方式对系统功耗有着直接而重要的影响,不同的汇编指令、软件算法以及软件的高层体系结构对系统功耗能够产生不同的作用,并且能够采用各种测量工具对影响程度进行量化. 可见,嵌入式硬件电路的低功耗设计是“系统省电的基础”,而嵌入式软件的低功耗设计是“系统省电的源头”<sup>[1-2]</sup>.

SoC-RTOS 是嵌入式软件的核心组成部件,为上层应用程序提供了硬件抽象和资源管理等功能,是满足嵌入式系统并发需求、提高嵌入式应用程序开发效率和可移植性的重要手段,也是嵌入式应用程序必不可少的运行平台. SoC-RTOS 一般采用微内核结构,基于优先权抢占的调度策略,具有任务管理、任务间同步和通信(如信号量、消息队列、异步信号、共享内存、管道)、内存管理和中断管理等功能,其运行方式有两种:嵌入式应用程序通过 API(Application Programming Interface,应用程序编程接口)显式地调用其服务例程,或者硬件产生的中断隐式地引起其服务例程的执行. RTOS 频繁执行不但占用了微处理器机器运行周期相当大的部分,而且消耗了 30~40% 的系统功耗. 因此,为了进一步推动嵌入式系统和 SoC 功耗优化技术的发展,

大力开展 SoC-RTOS 功耗优化方法研究是十分必要的<sup>[3-4]</sup>.

软/硬件自动划分(hardware-software automated partitioning)是嵌入式系统软/硬件协同设计(Hardware-software Co-design)方法的一个重要步骤,即在一定成本约束下提高系统的性能. 针对嵌入式实时系统,Gupta 和 De Micheli 提出了在协处理器和通用处理器上进行软/硬件划分的迭代算法<sup>[5]</sup>. Eles 采用模拟退火算法(Simulated Annealing Algorithm, SAA)和禁忌搜索算法(Tabu Search Algorithm, TSA)实现软/硬件划分<sup>[6]</sup>. Saha 开发了一种基于遗传算法(Genetic Algorithm, GA)的软/硬件划分算法<sup>[7]</sup>. Filho 等采用 Petri Nets 进行软/硬件划分<sup>[8]</sup>. Xiong 等实现了一种遗传算法和蚂蚁算法动态融合的混合算法进行嵌入式系统的软/硬件划分<sup>[9]</sup>. Arató 采用整数线性规划解决软/硬件划分问题,适合于相对较为复杂的系统<sup>[10]</sup>. Stitt 考虑了一种软/硬件动态划分的方案<sup>[11]</sup>. Stitt 和 Vahid 提出了软件二进制程序的软/硬件划分方法<sup>[12]</sup>. 在文献[13]中,我们提出了一种基于 Hopfield 神经网络的 SoC-RTOS 软/硬件划分方法,在硬件面积约束条件下优化 SoC-RTOS 的运行时间,明显地提高了多任务 SoC-RTOS 的运行性能. 但是,目前大多数软/硬件划分方法是在保持系统成本(如硬件面积)尽可能小的条件下,使得系统满足性能目标要求(如速度),几乎没有提供相关的功耗优化与估计策略.

在文献[14]中,针对基于微处理器结构的嵌入式系统,Henkel 提出了一种系统级的功耗优化方法,在细粒度(指令/操作级)的功耗估计与分析基础上,采用软/硬件划分方法对嵌入式系统的功耗进行优化. 同文献[1-2]中一些局部功耗优化技术相比,这种基于软/硬件划分的系统级功耗优化方法能够产生更高的功耗节省,大约为 35~95%. 但是,该文只从硬件角度,考虑了处理器核和 ASIC 核的功耗问题,由于在指令级或操作级进行精确的功耗估计是非常困难的,实际上无法保证软/硬件划分结果能够达到规定的功耗目标要求. 同时,采用简单的表调度(list scheduling)将划分对象与目标模块进行对比、排序,完成对软/硬件的划分,也难以保证划分结果是功耗最优的,或功耗次优的.

因此,SoC-RTOS 的功耗相关软/硬件划分(简称 RTOS-Power 划分)对于 SoC 设计是至关重要的,它决定了 SoC-RTOS 的哪些功能应该由硬件实现,哪些功能应该由软件实现,其划分结果直接影响

到 SoC 的系统功耗. 和一般的 RTOS 相比, 由于 SoC-RTOS 具有不同的特点和应用需求, 使得其具有不同的功能集合和软/硬件实现要求, 从而造成了 RTOS-Power 划分与一般的嵌入式系统和 SoC 划分存在比较大的区别, 过去的软/硬件划分方法在许多方面对于 RTOS-Power 划分是不充分的, 如 RTOS-Power 功耗建模与估计、功耗约束条件和目标参数的提炼、功耗优化求解算法的设计、划分结果的评价以及系统结构问题等. 在本文中, 我们将主要进行 RTOS-Power 划分优化算法的开发与设计<sup>[1]</sup>.

## 2 RTOS-Power 划分问题的描述

RTOS-Power 划分是一个 NP 完全问题, 其主要目标是将 RTOS 的功能行为在一定约束条件下优化地分配到 SoC 的软/硬件系统结构上, 以实现系统功耗的最小化. 在有些研究中, 将 RTOS-Power 划分作为 SoC-RTOS 软/硬件综合的一部分. SoC-RTOS 的功能行为一般采用任务图(task graph)建模, 每个任务具有相应的功耗属性, 我们采用基于任务级的系统功耗建模与估计策略. 对软件而言, 一个任务是具有明确接口的、粗粒度的一系列运算操作的集合, 通常表现为一个功能模块、函数、算法过程、对象或构件; 对硬件而言, 一个任务是一个特定的 IP(Intellectual Property, 知识产权)模块, 具有清晰的功能、接口和约束<sup>[11,13]</sup>.

为形式化地描述 SoC-RTOS 划分问题, 在本文中我们将使用下列符号:

$G$ : 一个有向无环图, 亦即一个 SoC-RTOS 功能行为的任务图,  $G=(V, E)$ ;

$V$ : 将要划分的任务节点集合,  $V=\{v_1, v_2, \dots, v_n\}$ ;

$E$ : 表示两个节点间控制或数据依赖/通信关系的有向边,  $E=\{e_{ij} \mid v_i, v_j \in V, i \neq j\}$ ;

$N$ :  $G$  的任务节点的数量,  $N=|V|$ ;

$P$ :  $G$  的一个功耗相关软/硬件划分;

$V_H$ : 划分为硬件的节点子集,  $V_H \subseteq V$ ;

$V_S$ : 划分为软件的节点子集,  $V_S \subseteq V$ ;

$s(v_i)$ (或  $s_i$ ):  $v_i$  软件实现的成本;

$h(v_i)$ (或  $h_i$ ):  $v_i$  硬件实现的成本;

$c(v_i, v_j)$ (或  $c_{ij}$ ):  $v_i$  和  $v_j$  的通信成本, 假如它们在不同的子集内(硬件子集或软件子集), 而同一个节点子集内节点间的通信成本忽略不计;

$c_i$ :  $c_{ji}$  之和,  $c_i = \sum_{j=1, j \neq i}^N c_{ji}$ ;

$H_P$ : 划分  $P$  的硬件成本之和,  $H_P = \sum_{v_i \in V_H} h_i$ ;

$S_P$ : 划分  $P$  的软件成本之和,  $S_P = \sum_{v_i \in V_S} s_i$ ;

$C_P$ : 划分  $P$  的通信成本之和,

$$C_P = \sum_{v_i \in V_S, v_j \in V_H \text{ 或 } v_i \in V_H, v_j \in V_S} c_{ij};$$

$g_P(V_H, V_S)$ : 划分  $P$  的系统整体成本;

$f_P(V_H, V_S)$ : 划分  $P$  的系统整体功耗.

**定义 1**( $k$  路划分). 对于给定的  $G=(V, E)$ ,  $k$  路划分就是寻找簇的集合  $P=\{p_1, p_2, \dots, p_k\}$ , 满足

$$\begin{cases} p_i \subseteq V, 1 \leq i \leq k \\ \bigcup_{i=1}^k p_i = V \\ p_i \cap p_j = \emptyset, 1 \leq i, j \leq k, i \neq j \end{cases} \quad (1)$$

当  $k=2$  时,  $P$  被称为双路划分, 它意味着在目标系统中, 考虑只有一个软件子集(如一个通用的微处理器)和一个硬件子集(如一个 ASIC 或 FPGA)的情况; 当  $k>2$  时,  $P$  被称为多路划分, 它意味着在目标系统中, 考虑多个软件子集(如多个通用的微处理器)和多个硬件子集(如多个 ASIC 或 FPGA)的情况. 因此, 根据目标系统的结构, RTOS-Power 划分可以分为双路划分和多路划分. 由于双路划分是多路划分的基础, 并广泛应用于相关的研究与开发中, 因此, 本文中, 在没有特别声明的情况下, 划分一般指的是双路划分.

**定义 2**(RTOS-Power 划分). 对于给定的  $P=(V_H, V_S)$ ,  $V_H \cup V_S = V$  和  $V_H \cap V_S = \emptyset$ , RTOS-Power 划分可以表示为下列组合优化问题的数学模型:

$$\begin{cases} \min & f_P(V_H, V_S) \\ \text{s. t.} & C_{\min} \leq g_P(V_H, V_S) = H_P + S_P + C_P \leq C_{\max} \\ & v_i \in V, e_{ij} \in E, 1 \leq i, j \leq n, i \neq j \end{cases} \quad (2)$$

其中,  $C_{\min}>0$  和  $C_{\max}>0$  分别是 SoC-RTOS 给定成本的最小值与最大值.

## 3 一种离散 Hopfield 神经网络方法

离散 Hopfield 神经网络方法(Discrete Hopfield Neural Network Approach, DHNNA)已经成功地应用于信号与图像处理、模式识别和组合优化等问题, 本文中, 将尝试采用该神经网络方法求解 RTOS-Power 划分组合优化问题.

### 3.1 神经元的表示

一个具有  $N$  个神经元的神经网络对应于任务图  $G$  的  $N$  个节点(如图 1 所示), 第  $i$  个神经元与节点  $i$

存在一种一一对应关系,即具有一个输入  $U_i$  和输出  $V_i$  间的函数关系. 神经元的激励函数由下式确定:

$$\begin{aligned} V_i &= f\left(\sum_{j=1, j \neq i}^N W_{ji} V_j - \theta_i\right) \\ &= f(U_i) \\ &= \begin{cases} 0, & U_i > 0 \\ 1, & U_i \leq 0 \end{cases} \end{aligned} \quad (3)$$

其中,  $W_{ji} V_j = h_j (1 - V_j) + s_j V_j + c_{ji} ((1 - V_j) + V_j)$ ,  $\theta_i = \alpha_i (1 - V_j) + \beta_i V_j + \varphi_{ij} ((1 - V_j) + V_j)$ , 权重  $W_{ji}$  是  $v_j$  的成本 (即  $s_j$  或  $h_j$ ) 和通信成本  $c_{ji}$  的和,  $\theta_i$  是第  $i$  个神经元的阈值 ( $\alpha_i, \beta_i$  和  $\varphi_{ij}$  将在 3.2 小节中做进一步的解释). 同时, 神经元输出值  $V_i = 0$ , 表示  $v_i \in V_H$ ;  $V_i = 1$ , 表示  $v_i \in V_S$ .

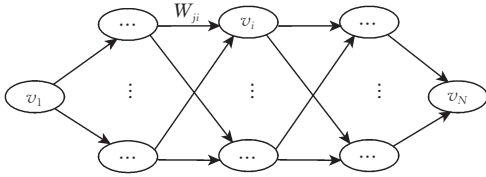


图 1 离散 Hopfield 神经网络的结构

为避免初始条件引起的局部优化问题,神经元的输入值应当限制在某一范围内.  $c_{ij}$  的平均值  $U_{avg}$  采用所有相连的任务节点进行计算,因此,神经元输入的上限  $U_{max}$  和下限  $U_{min}$  按如下设置:

$$U_{avg} = \frac{2}{N} C_P, \quad U_{min} = -\frac{U_{avg}}{2}, \quad U_{max} = \frac{U_{avg}}{2} \quad (4)$$

$$U_i = \begin{cases} U_{min}, & U_i < U_{min} \\ U_{max}, & U_i > U_{max} \end{cases} \quad (5)$$

### 3.2 能量函数

对应于 RTOS-Power 划分的约束条件和目标函数, Hopfield 神经网络的能量函数包括下列两部分:

$$E = \frac{A}{2} E_1 + \frac{B}{2} E_2 \quad (6)$$

$$E_1 = \sum_{i=1}^N \sigma_i^2 \left( \sum_{j=1}^N (h_j V_i (1 - V_j) + s_j (1 - V_i) V_j + c_{ij} (V_i (1 - V_j) + (1 - V_i) V_j)) - C_{min} \right) \quad (7)$$

$$\begin{aligned} E_2 &= f_P(V_H, V_S) \\ &= \sum_{i=1}^N \left( \sum_{j=1, j \neq i}^N (\alpha_i V_i (1 - V_j) + \beta_i (1 - V_i) V_j + \varphi_{ij} (V_i (1 - V_j) + (1 - V_i) V_j)) \right) \end{aligned} \quad (8)$$

其中,  $A$  和  $B$  是两个正系数,在 3.4 小节将进一步确定它们的值.  $\alpha_i$  表示一个任务节点硬件实现的功耗,不同的任务节点具有不同的值;  $\beta_i$  表示一个任务节点软件实现的功耗,不同的任务节点具有不同的

值;  $\varphi_{ij}$  表示一个相应的任务节点间通信的功耗,不同的通信具有不同的值<sup>[13-14,16]</sup>.

在方程(7)中函数  $\sigma_i(x)$  由下式确定:

$$\sigma_i(x) = \begin{cases} x + C_{min} - (h_i + s_i + c_i), & x < -C_{min} + (h_i + s_i + c_i) \\ 0, & 0 \leq x \leq C_{max} - C_{min} \\ x - C_{max} + (h_i + s_i + c_i), & x > C_{max} - (h_i + s_i + c_i) \end{cases} \quad (9)$$

$E_1$  是能量函数的一个约束条件,同  $g_P(V_H, V_S) = \sum_{i=1}^N \left( \sum_{j=1, j \neq i}^N (h_j V_i (1 - V_j) + s_j (1 - V_i) V_j + c_{ij} (V_i (1 - V_j) + (1 - V_i) V_j)) \right)$  相关;  $E_2$  是能量函数的一个目标函数  $f_P(V_H, V_S)$ , 表示 SoC-RTOS 的功耗值<sup>[4,13]</sup>.

### 3.3 运行方程

第  $i$  个神经元的运行方程为

$$\begin{aligned} \frac{dU_i}{dt} &= -\frac{\partial E}{\partial V_i} \\ &= -A \sigma_i \left( \sum_{j=1}^N (h_j V_i (1 - V_j) + s_j (1 - V_i) V_j + c_{ij} (V_i + V_j - 2V_i V_j)) - C_{min} \right) \times \\ &\quad \left( \sum_{j=1}^N (h_j (1 - V_j) - s_j V_j + c_{ij} (1 - 2V_j)) \right) - \\ &\quad \frac{B}{2} \sum_{j=1, j \neq i}^N ((\alpha_i + \varphi_{ij}) - (\alpha_i + \beta_i + 2\varphi_{ij}) V_j) \end{aligned} \quad (10)$$

为进一步避免出现局部优化现象和在有限的时间获得一个高质量的解,一个“噪声”条件  $D$  将加到运行方程(10)上,即

$$D = \eta(1 - 2V_j) = \begin{cases} +\eta, & V_j = 0 \\ -\eta, & V_j = 1 \end{cases} \quad (11)$$

但是,如果“噪声”条件  $D$  一直加在状态更新规则中,神经网络的状态可能会变化过度,以至于连一个局部最优解都无法获得. 因此,当条件  $[t/T_0] \geq \lambda$  满足时,在运行方程中条件  $D$  将被丢弃,其中  $[\cdot]$  是一个取整操作符,  $\lambda = T_0 - (t \times T_0) / T_{max} - 1$ ,  $T_0$  是一个正系数,  $T_{max}$  是迭代的最大步数.

### 3.4 运行方程系数的确定

系数  $A$  由任务节点成本的平均值  $\bar{\omega}$  确定,即

$$\bar{\omega} = \frac{H_P + S_P}{N} \quad (12)$$

系数  $B$  由  $U_{avg}$  确定,设定  $A \bar{\omega} = K B U_{avg}$ , 其中  $K$  是一个调节常数. 实验中,取  $K = 1/3$ ,  $B = 1$ ,  $T_0 = 20$ ,  $T_{max} = 300$ ,  $\eta = U_{avg} / (3 + 10 \times \rho)$ . 这里,  $\rho = M / (N(N - 1) / 2)$  表示图  $G$  的边产生率 ( $0 < \rho \leq 1$ )<sup>[15]</sup>.

$\alpha_i, \beta_i$  和  $\varphi_{ij}$  的值可通过一个低层的功耗仿真

器 EMSIM 获得,该仿真器模拟了一个包含 Intel StrongARM 处理器和 RTOS  $\mu\text{C}/\text{OS}$  的 SoC 平台.另外,不同软/硬件实现条件下任务节点的执行时间也可以通过该仿真器获得<sup>[4,13]</sup>.

当 SoC-RTOS 完全由软件实现时,  $C_{\min} = \sum_{i=1}^N s_i$  是最小成本;当 SoC-RTOS 完全由硬件实现时,  $C_{\max} = \sum_{i=1}^N h_i$  是最大成本.为使方案更加实用、合理,将最大成本修订为  $C_{\max} = \frac{1}{2} \sum_{i=1}^N h_i$ <sup>[16-17]</sup>.

## 4 仿真实验

为验证本文提出的 DHNNA 的可行性和有效性,采用与文献[4,9]和文献[13]相似的仿真模型和方法,同时,与采用遗传算法 (Genetic Algorithm, GA) 和蚂蚁算法 (Ant Algorithm, AA) 的 SoC-RTOS 划分结果进行比较.其中,遗传算法和蚂蚁算法的参数设置如文献[9]所示.

### 4.1 目标系统结构

由于本文面向双路划分问题,因此,在目标系统中只有 1 个处理器和 1 个 FPGA 模块(如图 2 所示).使用 Xilinx 公司的 Spartan-3 S1000 芯片作为 FPGA 模型,它最多可包含 4 个处理器核和 17280 个可编程逻辑块 (Programming Logic Blocks, PLB).实验中,只使用了 1 个处理器核和最多 3284 个可编程逻辑块.由于软件保存在存储器中,由 StrongARM 处理器执行,因此,实验中 StrongARM 处理器和存储器代表软件,FPGA 模块代表硬件.

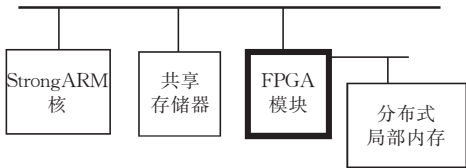


图 2 目标系统结构的抽象模型

### 4.2 实验条件

目前,在 SoC-RTOS 及嵌入式系统软/硬件划分领域,国际上还没有公认的测试基准和测试集.常用方法是随机产生有向无环图,并对节点和边赋以相关属性.

为简化实验,做出如下合理的假设:  
(1) 任务在处理器和 FPGA 可编程逻辑块上实现所需的成本(如运行时间和占用的硬件面积)是静态的,可预先计算出来.

(2) 不同软/硬件划分节点间的通信成本在运

行时间内固定不变.

(3) 为在相同条件下与软件实现进行比较,不考虑硬件实现的并行性.

在实验中,实验条件设置如下:  
(1) 用 GVF (Graph Visualization Framework) 软件包随机生成 5 组有向无环图作为任务图,每组的节点数  $N$  分别为 50, 300, 800, 1500 和 2500. 每组有 30 个样本图形,每个图形有不同的边生成率  $\rho$ ,以 30 个样本图形的功耗平均值作为该组的最终功耗值.

(2) 任务节点的成本和边的通信成本:每个任务节点有两个成本关联函数,一个是硬件成本函数,另一个是软件成本函数,而每个边只有一个成本关联函数.成本函数的开销(如运行时间、硬件面积、通信代价等)作为任务节点和边的成本,从 MediaBench 基准程序包中选择与任务节点和边相关联的成本函数<sup>[9]</sup>.

(3) 作为 RTOS-Power 划分的一个初始条件,软件子集和硬件子集各分配  $N/2$  个任务节点.

(4) 仿真环境为 Intel Celeron 2.6GHz 处理器, 512MB SDRAM, Linux 9.0 操作系统, KDevelop 3.2 集成开发环境.

### 4.3 实验结果与分析

表 1 给出了 DHNNA、GA 和 AA 3 种算法进行 RTOS-Power 划分得到的  $f_P(V_H, V_S)$  (单位: mJ) 和与纯软件实现的 SoC-RTOS 相比的功耗节省比例的实验数据,表 2 给出了 DHNNA、GA 和 AA 3 种算法进行 RTOS-Power 划分得到的测试程序执行时间和与纯软件实现的 SoC-RTOS 执行时间相比的降低比率的实验数据,图 3 给出了 3 种算法的  $f_P(V_H, V_S)$  与任务节点数的关系,图 4 给出了 3 种算法的执行时间与任务节点数的关系.

在图 2 所示的目标系统结构下,本实验的目标是在芯片面积约束条件下优化 SoC-RTOS 的功耗,并在相同条件下验证 SoC-RTOS 的运行时间等其它性能.可以看出,DHNNA 获得的功耗值明显地小于 GA 的获得值,略微小于 AA 的获得值.同时,DHNNA 获得的执行时间小于 GA 的获得值,略微大于 AA 的获得值.由于 SoC-RTOS 的一部分功能采用硬件实现,使得其不但消耗了较少的能量,而且运行速度更快,这是本文提出的 DHNNA 方法非常重要的一个优点.实际上,经过适当的修改,DHNNA 能够用于一般嵌入式系统和 SoC 的软/硬件划分,同时,在考虑其它约束条件和优化目标的情况下(如硬实时和多处理器等)也是适用的.

表 1 DHNNA、GA 和 AA 的  $f_P(V_H, V_S)$  与功耗节省比例的结果

总 DAG 节点数	DHNNA		GA		AA	
	$f_P(V_H, V_S)$	节省比例 / %	$f_P(V_H, V_S)$	节省比例 / %	$f_P(V_H, V_S)$	节省比例 / %
50	221.73	-42.41	281.29	-26.94	234.26	-39.22
300	616.32	-49.74	721.47	-35.38	587.21	-46.92
800	1451.22	-53.65	1867.55	-37.63	1787.39	-50.26
1500	6412.37	-60.08	9605.82	-40.20	7216.95	-55.07
2500	8128.92	-65.08	11212.25	-41.75	8736.58	-56.62

表 2 DHNNA、GA 和 AA 的执行时间与降低比率的结果

总 DAG 节点数	DHNNA		GA		AA	
	执行时间/cycles	降低比率/%	执行时间/cycles	降低比率/%	执行时间/cycles	降低比率/%
50	1134028	-11.35	1212060	-5.25	1079532	-15.61
300	3062381	-16.87	3629257	-10.27	2912461	-20.35
800	5652693	-23.92	6249128	-13.45	5383657	-25.86
1500	8642516	-28.75	10064134	-17.03	8458142	-30.27
2500	11739274	-30.68	14592472	-20.28	10542935	-32.68

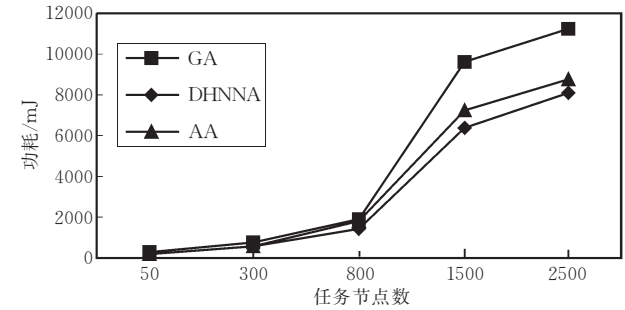


图 3 功耗与任务节点数的关系

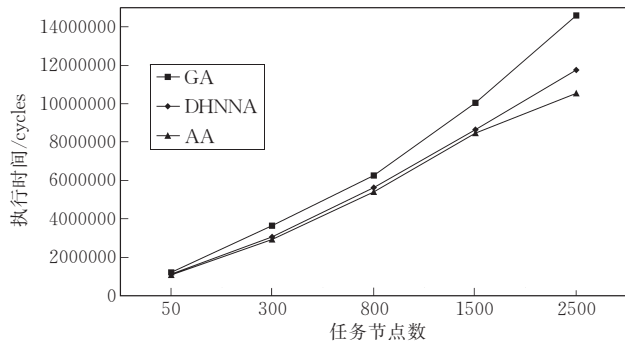


图 4 执行时间与任务节点数的关系

随着任务节点数的增加,  $f_P(V_H, V_S)$  的值也在不断地增长, 这是 DHNNA 的一个主要缺点. 在工程实践中, 期望  $f_P(V_H, V_S)$  是一个更加精确和稳定的值, 导致出现这种情况的一个原因是初始条件和神经网络参数的设置, 另一个原因是 SoC-RTOS 的功耗建模和估计不够精确. 因此, SoC-RTOS 的功耗宏建模 (energy macromodeling) 是下一步值得研究的问题, 它是一种基于操作系统服务例程级的更加精确的功耗建模方法, 能够有效地改善 RTOS-Power 划分的结果.

5 结 论

为解决 RTOS-Power 划分的优化求解问题, 本文提出了一种离散 Hopfield 神经网络方法. 根据 SoC-RTOS 划分的特点, 重新定义了 Hopfield 神经网络的能量函数、状态更新规则和参数设置. 仿真实验结果表明, 本文提出的 DHNNA 优于其它一些传统的优化算法, 如遗传算法和蚂蚁算法, 能够以相对较小的代价 (FPGA 开销小于 4K 个可编程逻辑块) 取得高达 60% 的功耗节省, 同时, 与纯软件实现的 SoC-RTOS 相比, 系统性能也得到了相应的提高. 下一步需要研究的主要问题是基于 SoC-RTOS 功耗宏建模的 RTOS-Power 划分.

参 考 文 献

[1] Jerraya A A, Yoo S, Verest D, When N. Embedded Software for SoC. Netherlands: Kluwer Academic Publishers, 2003

[2] Dick R P. Multi-objective synthesis of low-power real-time distributed embedded systems [Ph. D. dissertation]. Department of Electrical Engineering, Princeton University, New Jersey, USA, 2002

[3] Li T, John L K. Run-time modeling and estimation of operating system energy consumption//Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems. San Diego, California, USA, 2003: 160-171

[4] Tan T K, Raghunathan A, Jha N K. Energy macromodeling of embedded operating systems. ACM Transactions on Embedded Computing Systems, 2005, 4(1): 231-254

[5] Gupta R K, De Micheli G. Hardware-software co-synthesis for digital systems. IEEE Design and Test of Computers, 1993, 10(3): 29-41



- [6] Eles P, Peng Z, Kuchcinski K, Doboli A. System level hardware/software partitioning based on simulated annealing and tabu search. *Design Automation for Embedded Systems*, 1997, 2(1): 5-32
- [7] Saha D, Mitra R S, Basu A. Hardware/software partitioning using genetic algorithm//*Proceedings of the International Conference on VLSI Design*. Hyderabad, India, 1998: 155-159
- [8] Filho F C, Maciel P, Barros E. A petri nets based approach for hardware/software partitioning. *Integrated Circuits and System Design*, 2001, 8(6): 72-77
- [9] Xiong Zhi-Hui, Li Si-Kun, Chen Ji-Hua. Hardware/software partitioning based on dynamic combination of genetic algorithm and ant algorithm. *Journal of Software*, 2005, 16(4): 503-512(in Chinese)  
(熊志辉,李思昆,陈吉华. 遗传算法与蚂蚁算法动态融合的软/硬件划分. *软件学报*, 2005, 16(4): 503-512)
- [10] Arató P, Juhász S, Mann Z Á, Papp D. Hardware-software partitioning in embedded system design//*Proceedings of the IEEE International Symposium on Intelligent Signal Processing*. Budapest, Hungary, 2003: 63-69
- [11] Stitt G, Lysecky R, Vahid F. Dynamic hardware/software partitioning: A first approach//*Proceedings of Design Automation Conference*. Anaheim, California, USA, 2003: 74-81
- [12] Stitt G, Vahid F. Hardware/software partitioning of software binaries//*Proceedings of the IEEE/ACM International Conference on Computer Aided Design*. San Jose, California, USA, 2002: 164-170
- [13] Guo B, Wang D H, Shen Y, Li Z S, Liu Z. Hardware-software partitioning of real-time operating systems using Hopfield neural networks. *NeuroComputing*, 2006, 69(16-18): 2379-2384
- [14] Henkel J. A low power hardware/software partitioning approach for core-based embedded systems//*Proceedings of the Design Automation Conference*. New Orleans, Louisiana, USA, 2002: 122-127
- [15] Tamaki Y, Funabiki N, Nishikawa S. A binary neural network algorithm for the graph partitioning problem. *Electronics and Communications in Japan, Part 3*, 1999, 82(12): 34-42
- [16] Pereira C, Raghunathan V, Gupta S, Gupta R, Srivastava M. A software architecture power aware real time operating systems. University of California, Irvine, California, USA: CECS Technical Report: 02-02, 2002
- [17] He Z T, Mok A, Peng C. Timed RTOS modeling for embedded system design//*Proceedings of the IEEE Real-Time and Embedded Technology and Applications Symposium*. San Francisco, California, USA, 2005: 54-63



**GUO Bing**, born in 1970, Ph. D., associate professor. His current research interests include embedded real-time system, SoC and middleware.

**SHEN Yan**, born in 1973, Ph. D., associate professor. Her main research interests include distributed measurement systems, embedded system development, wireless sensor networks, robotics.

## Background

This research is partly supported by the National Natural Science Foundation of China (No. 60572026), and undertaken in cooperation with the computation intelligence Laboratory of Department of Computer Science, La Trobe University, Australia.

Along with the development of global energy crisis, power issues of software in the vast number of embedded systems have been increasingly concerned. The research group has done much research work in RTOS, software/hardware automatic partitioning algorithms and software energy consumption models. However, RTOS plays an important role in the running of embedded software. Many existed researches mainly concern the dynamic behavior of RTOS,

**WANG Dian-Hui**, born in 1963, Ph. D., associate professor. His current research interests include computational intelligence and data mining for bioinformatics and multimedia information processing.

**LI Zhi-Shu**, born in 1946, professor, Ph. D. supervisor. His current research interests include intelligence system and medical data processing.

**CHEN Xiang-Dong**, born in 1967, Ph. D., professor and Ph. D. supervisor. His main research interests include SoC, wireless sensor networks.

such as power-aware scheduling algorithm. This project mainly focuses on the energy optimization of RTOS (Real-time Operating System) in SoC (System-on-a-Chip) based on software/hardware automatic partitioning method. The authors attempt to employ Hopfield neural network to resolve this combined optimization problem, and achieve some initial results. As a system-level power optimization approach, it can yield high energy savings compared to other techniques, such as Genetic algorithm and Ant algorithm. Simulations experimental results demonstrate that the proposed method can acquire higher energy savings up to 60% at relatively low costs of less than 4K PLBs while increasing the performance compared to the SoC-RTOS realized purely in software.