第34卷第6期　　中国科学技术大学学报　　Vol. 34, No. 6
2004年12月　JOURNAL OF UNIVERSITY OF SCIENCE AND TECHNOLOGY OF CHINA　Dec. 2004

文章编号:0253-2778(2004)06-0724-08

基于遗传算法的嵌入式系统软硬件划分算法*

邹　谊,庄镇泉,杨俊安

(中国科学技术大学电子科学与技术系,合肥 230026)

摘 要:针对嵌入式系统软硬件协同设计中的软硬件划分问题,提出了一个基于基本调度块图的软硬件划分模型,并给出了一个基于遗传算法的软硬件划分算法.通过采用自适应的适应度函数和演化策略,提高了算法的稳定性、搜索效率和求解质量.实验结果说明了该算法对解决软硬件划分问题是有效的.

关键词:软硬件协同设计;软硬件划分;遗传算法;自适应演化策略

中图分类号:TP302　　**文献标识码**:A

0　引言

随着现代嵌入式系统规模的增加及系统功能的日益复杂,传统设计方法已经无法满足现代嵌入式系统的设计要求,取而代之的是以软硬件协同设计为主要特征的系统设计方法[1,2,3].软硬件协同的设计方法,通过系统资源配置(allocation)、软硬件分配(assignment)以及调度(scheduling)三个步骤用自动、优化的系统体系结构开发代替人工的软硬件子系统分割,在系统设计的高层阶段实现系统原型的快速开发并预估系统实现的性能,优化系统的性能、成本、功耗等.

根据对系统功能要求分析的粒度不同,目前的软硬件协同设计方法主要分为两类:一类采用任务图(task graph)描述系统功能,分析粒度为任务级,称为软硬件协同综合(cosynthesis);另一类采用控制数据流图(CDFG)描述系统功能,分析粒度为基本调度块(BSB:basic scheduling block),称为软硬件划分(partitioning).本文重点讨论软硬件划分方法.

目前,见于文献的软硬件划分算法主要有:Ernst 等提出了基于软件的方法[4],利用模拟退火算法选择软硬件划分,但算法中模拟退火的运算时间较长,退火起始温度以及退火速度选择对算法质量影响很大,选择较困难.Gupta&Michela 等提出了基于硬件的方法[5],由于系统功能的复杂性,寻找系统功能的全部硬件实现过程本身就是非常复杂甚至不太现实,算法存在实现上的困难.Knudsen&Madsen 等的 PACE 方法[6],采用动态规划的方法完成软硬件的划分,算法对系统硬件的成本约束限定为整数,算法的复杂度与系统中基本调度块数目的平方以及成本约束成正比,算法运行过程中需要维持一个很大的规划表,算法的内存

* **收稿日期**:2003-11-13
基金项目:国家自然科学基金资助项目(60171029).
作者简介:邹谊,男,1971 年生,博士生.主要研究方向:软硬件协同设计,量子计算.E-mail:rock@szonline.net.

开销和运算开销较大. 此外,算法中将基本调度块的划分局限于相邻 BSB 构成的 BSB 序列,降低了软硬件划分的优化潜力. Kalavade 等提出的 MIBS 算法[7],通过启发式的构造方法,动态调整软硬件划分的阈值,但容易因为先考虑的功能块划分结果对资源的过大占用,使得后续的功能块的划分无法按阈值调整结果选择合适的实现,使得整体划分的最终结果与最优划分产生偏差. 同时算法忽略了软硬件实现之间数据传递开销的影响.

软硬件划分问题本质上是一类组合优化问题,问题模型和求解算法选择是解决软硬件划分问题的关键,因此同时划分模型中必须考虑数据通讯的影响. 本文提出了一个基于系统基本调度块图的软硬件划分模型,并给出了一个基于遗传算法的软硬件划分算法.

1　系统功能描述

软硬件划分算法的目标是在满足一定系统约束(通常是成本约束)的前提下实现系统性能(通常是系统运行时间)的最优化,选择系统功能的全软件实现作为算法优化的出发点.

系统功能的软件实现通常采用 C/C++语言等高级程序设计语言. 为方便划分,首先将系统功能的高级语言描述转为控制数据流图(CDFG)的形式. 控制数据流图(CDFG)由节点集 N 和有向边集 E 构成,对于 $n_i \in N$, 有向边 $e_{i,j}=(n_i,n_j) \in E$ 表示节点间的控制关系或数据流向.

对于 $n_i \in N$ 的节点,可以表示为如下的 5 种形式之一:

n_i =DFG | cond | loop | FU | wait

cond=(branch1,branch2)

loop=(test,body)

branch1=CDFG

branch2=CDFG

test=CDFG

body=CDFG

FU=CDFG

其中,DFG 表示单纯数据流图,不包含控制结构,cond 和 loop 分别表示条件分支和循环控制结构,branch1 和 branch2 代表不同条件分支,test 和 body 代表循环终止判断和循环体. FU 表示子过程或函数调用,wait 用于和运行环境同步. 利用系统功能的 CDFG 描述,可以得到相应的所谓基本调度块(BSB). 最基本的 BSB 块对应于单纯的数据流图(DFG),通过坍缩(Collapse)操作可以将相邻的 BSB 合并,从而控制 BSB 的数目,压缩划分算法的搜索空间.

下面通过一个简单的实例说明系统功能的高级语言(C 语言)实现,控制数据流图(CDFG)以及基本调度块 BSB 描述之间的对应关系. 图 1 为对 100 以内奇数和偶数分别求和的程序,图 2 为相应的 CDFG 描述,图 3 为对应的 BSB 块,其中每个圆点对应一个 BSB 块. 系统的 BSB 块构成 BSB 描述图.

```
i=1; j=0; k=0;
while (i <100)
{ if ((i %2)==0)   j+=i;
  else    k+=i;
   i++;}
```

图 1　系统功能的 C 语言描述

Fig. 1　C description of system function

系统的 BSB 描述图同样由节点集 B 和有向边集 E 构成. 在 BSB 图中,每个 BSB 块节点

Bi 的序号对应于系统执行各 BSB 块的顺序. 对于 $B_i \in B$, 有向边 $e_{i,j} = (B_i, B_j) \in E$ 表示 BSB 块之间存在的数据交换关系,有向边属性值 $v(e_{i,j})$ 为 BSB 块之间的数据交换量或变量集. 每个 BSB 节点可以从它的前驱节点得到数据,也可以向它的后续节点输出数据. 一个 6 BSB 节点示范系统的 BSB 描述图如图 4.

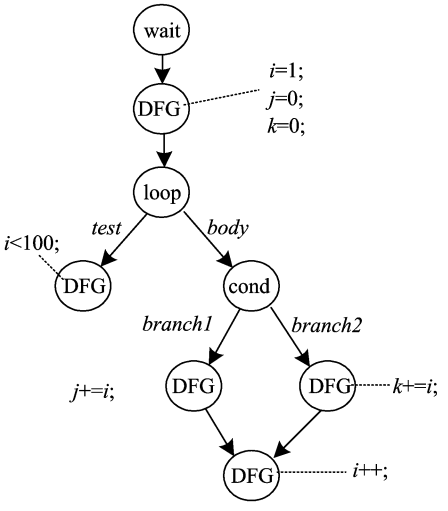

图 2 CDFG 描述

Fig. 2 CDFG description

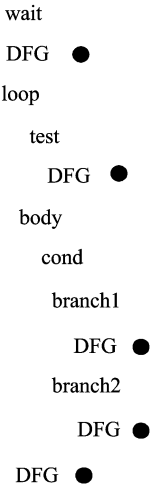

图 3 对应的 BSB 块

Fig. 2 Corresponding BSB blocks

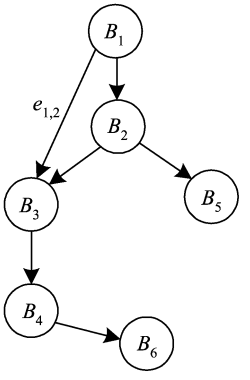

图 4 系统的 BSB 描述图

Fig. 4 BSB graph

每个 BSB 节点 B_i 属性定义为如下的一个 7 元组:

$$B_i = < as_i, ts_i, ah_i, th_i, r_i, w_i, pc_i >$$

其中, as_i 表示 B_i 的软件实现面积(代价),可以用 B_i 需要的指令代码数和(或)需要的存储器数定义,软硬划分算法中,通常可以不考虑该属性,将该属性置为 0; ts_i 表示 B_i 由软件实现时的执行时间, ah_i 表示 B_i 的硬件实现面积(代价),可以用 B_i 综合后的基本可编程逻辑块数或逻辑门数定义; th_i 表示 B_i 由硬件实现时的执行时间, r_i, w_i 表示 B_i 运行时需要输入和输出的变量集,是 BSB 之间通讯开销的度量; pc_i 表示系统运行时 B_i 被调用的次数. B_i 属性数据的获取主要采用统计(Profiling)的方法,如输入输出变量集可以利用编译工具得到,调用次数可以通过运行计数统计. 值得注意的是 B_i 属性中的 ah_i 和 th_i 属性,由于采用硬件实现的各 BSB 块之间存在资源共享的可能, B_i 的硬件综合采用带有数据路径的有限状态机模型(FSMD),数据路径资源由系统预留,供各硬件实现的 BSB 块共享, ah_i 对应 B_i 综合后的有限状态控制机部分的硬件代价,预留的数据路径资源影响各 B_i 的 th_i 属性值.

图 5 体系结构模板

Fig. 5 System architecture template

2 划分模型

软硬件划分算法采用主处理器-从处理器的体系结构模板,主处理器通常是通用的处理器,从处理器通常是专用硬件 ASIC 或可编程逻辑器件 FPGA,主-从处理器之间通过共享存储耦合的方式实现数据通讯,如图 5 所示. 划分算法的目标定义为在满足系统硬件成本约束 CostReq 的

条件下,最小化系统运行时间 T,系统硬件成本 CostReq 由设计者事先给定.

本文提出了一个基于系统功能的基本调度块 BSB 描述的划分模型. 通过将适当的 BSB 块由软件实现划分为硬件实现就能在满足成本约束的条件下,实现系统运行时间的最小化,达到划分目标. 划分模型必须同时考虑 BSB 之间数据通讯对划分结果的影响.

划分后的系统包括硬件部分 HW 和软件部分 SW,每个 BSB 块分别采用硬件或软件方式实现. BSB 之间的通讯开销为 BSB 之间的变量传递,同属软件部分的 BSB 之间的通讯开销以及同属硬件部分的 BSB 之间的通讯开销可忽略为零,系统通讯开销记为分属软硬件部分的 BSB 之间的通讯开销之和. BSB 节点 B_i 的输入输出变量集 r_i 和 w_i 中的有效输入输出集定义为 r^{*i} 和 w^{*i}.

对于硬件方式实现的节点 B_i,有:

$$r_i^* = \bigcup_{B_k \in \mathrm{SW} \cap \exists e_{k,j}=(B_k,B_i)} \{v(e_{k,j})\} \qquad w_i^* = \bigcup_{B_k \in \mathrm{SW} \cap \exists e_{i,k}=(B_i,B_k)} \{v(e_{i,k})\}$$

对于软件方式实现的节点 B_i,有:

$$r_i^* = \bigcup_{B_k \in \mathrm{HW} \cap \exists e_{k,j}=(B_k,B_i)} \{v(e_{k,j})\} \qquad w_i^* = \bigcup_{B_k \in \mathrm{HW} \cap \exists e_{i,k}=(B_i,B_k)} \{v(e_{i,k})\}$$

r^{*i} 和 w^{*i} 分别代表了 BSB 节点 B_i 来自不同实现方式的 BSB 的输入变量集和传递到不同实现方式的 BSB 的输出变量集.

划分后系统的整体运行时间表示为 T,有

$$T = \mathrm{SW}_p - \sum_{\forall B_i \in \mathrm{HW}} pc_i(ts_i - th_i) + \Big(\sum_{v \in r_i^* \cap r_i^* \in B_i \cap B_i \in \mathrm{SW}} pc_i(v) \cdot t_{m \to s} + \sum_{v \in w_i^* \cap w_i^* \in B_i \cap B_i \in \mathrm{SW}} pc_i(v) \cdot t_{s \to m} + \sum_{v \in r_i^* \cap r_i^* \in B_i \cap B_i \in \mathrm{HW}} pc_i(v) \cdot t_{m \to h} + \sum_{v \in w_i^* \cap w_i^* \in B_i \cap B_i \in \mathrm{HW}} pc_i(v) \cdot t_{h \to m} \Big)$$

式中,SW_P 表示系统全部由软件实现的时间,中间项表示由 BSB 块硬件化带来的加速量,后面项表示软硬件部分之间的通讯开销. $t_{s \to m}$ 表示变量从软件写入共享存储器的时间,$t_{m \to s}$ 表示变量从共享存储器读入软件的时间,$t_{h \to m}$ 表示变量从硬件写入共享存储器的时间,$t_{m \to h}$ 表示变量从共享存储器读入硬件的时间,$pc_i(v)$ 表示基本调度块 B_i 中变量 v 被调用次数,等于 B_i 被调用次数 pc_i.

系统的硬件成本 Cost 表示为所有的硬件实现 BSB 的成本之和, $\mathrm{Cost} = \sum_{\forall B_i \in \mathrm{HW}}$

划分目标可以描述为

$$\begin{aligned} &\text{Minimize} \quad T \\ &\text{subject to } \mathrm{Cost} \leqslant \mathrm{Cost\ Req} \end{aligned}$$

3　基于遗传算法的划分算法

搜索效率、解的质量、算法的鲁棒性是软硬件划分算法设计主要考虑的问题. 据此本文提出了一个基于遗传算法的划分算法,并针对软硬件划分问题的特点,具体设计了自适应的适应度函数和演化策略.

3.1　编码、初始群体生成、群体规模及演化终止条件

算法的目的在于确定各 BSB 块的软硬件实现方式,因此,直接采用二值符号串编码,染

色体定义为 $(k_1、k_2、k_3\cdots k_n)k_i \in \{1,0\}\ i=\{1\cdots n\}$，其中 n 是 BSB 节点数，$k=1$ 表示节点由硬件实现，0 表示由软件实现. 初始群体随机产生，但产生的个体必须满足系统的成本约束条件，舍弃不满足成本约束条件的个体. 兼顾群体多样性和算法执行效率的要求，群体规模取为系统 BSB 数目 N 的 2 倍. 进化过程中，保持群体规模不变. 经过 $3N$ 代演化，演化过程终止.

3.2 适应度函数

适应度函数的选择，可以首先根据算法优化目标即系统整体运行时间 T 以及系统成本约束 CostReq 构造一个广义目标函数，然后通过对广义目标函数定标得到. 系统成本约束转为广义目标函数中的惩罚项. 需要注意的是，惩罚项的选择应保证划分结果满足约束，同时也要保证划分结果对系统成本的充分利用，系统成本过小往往不对应最优结果. 此外，成本约束和运行时间往往存在数值范围和量级上的差异，需要做归一化处理.

引入归一化因子 σ_c 和 σ_t.

$\sigma_c = \max((\mathrm{CostHW}-\mathrm{CostReq}),\mathrm{CostReq})$，CostHW 为系统全硬件实现成本.

$\sigma_t = \mathrm{SW}-\mathrm{HW}_p$，$\mathrm{SW}_p$ 和 HW_p 分别为系统的全软件实现时间和全硬件实现时间.

广义目标函数定义为：

$$\mathrm{COBJ} = \beta^{*} e^{\frac{\mathrm{Cost}-\mathrm{CostReq}}{\sigma_c \cdot M^i}} * \frac{|\mathrm{Cost}-\mathrm{CostReq}|}{\sigma_c} + \lambda * \frac{T-\mathrm{HW}_p}{\sigma_t}$$

算法寻优对应于广义目标函数的最小化，适应度函数定义为

$$\mathrm{Fitness} = \frac{1}{1+\mathrm{COBJ}}$$

广义目标函数的前部分增加了一个指数惩罚项，用于对违反约束个体的惩罚. M_i 是类似模拟退火算法中的退火系数，演化过程中取 $M_{i+1} = 0.98 * M_i$，演化初期对违反约束个体的惩罚较小，以保持种群多样性，随着演化过程的进行，逐渐加大对违反约束个体的惩罚. 系数 β,λ 用来调整 Cost 和 T 的相对比重.

3.3 选择机制、演化策略

算法采用基于赌轮选择和精英保留策略的混合选择机制. 算法中既保留当代中的适应度值最大的个体(此个体可能违反系统的成本约束)，也保留符合系统约束的个体(称为有效个体)中具有最大适应度值的个体.

交叉和变异算子的设计对算法的效率和解质量影响很大. 我们认为，交叉操作相当于算法在相对较大的范围内寻优，而变异操作相当于算法在个体邻域内寻优. 因此，在演化的初期应充分发挥交叉操作的作用，加快算法的搜索速度，同时保持群体的多样性，在演化的后期应充分发挥变异操作的作用，变异操作有意识地针对优良个体进行以提高解的质量.

算法设计中，子代群体的组成按比例 $P\%$和$(1-P)\%$设定为通过交叉和变异两种途径得到. 在演化初期，确定子代个体中有 $P_0\%$的个体通过交叉操作产生，交叉的父代个体按赌轮机制选择，采用简单单点交叉，交叉概率 P_c(实验中取 0.9)；剩余$(1-P_0)\%$的子代个体通过变异操作产生，变异操作选择父代中适应度取值前$(1-P_0)\%$的个体进行，变异操作改变的染色体基因位数为 m(实验中取 $m=2$). 随着演化过程的进行，调整子代群体的组成比例，逐渐减少子代中通过交叉操作产生的个体数目而增加通过变异操作产生的个体数目，即

$P = P_0 + i * \Delta P$，ΔP 为每代演化的比例调整因子. 实验表明，设计的交叉和变异算子有较好的效果.

4　实验结果及分析

由于在前述的文献中没有列出可供比较的原始数据，因此，本文采用随机产生的系统 BSB 图数据的方法进行算法实验，同时，根据文献设计了相应的模拟退火算法(SA)供比较. 算法采用 C++在 Pentium III 500MHz 128M Ram 环境下实现. 随机产生的 BSB 数据各属性值之间考虑了数值的合理性. 表 1 列出了一个 10 节点示范系统的产生数据，表 2 列出了节点间的数据传递关系，表 3 列出了部分划分实验结果，从表中数据可以看出，在得到同样甚至更优化结果的情况下，本文提出的遗传算法的搜索效率要明显高于模拟退火算法. 图 6 为 50 节点示范系统运行 20 次，划分过程中平均每代最佳有效个体广义目标函数曲线，从图中可以看出算法有很好的收敛性. 图 7 为 50 节点示范系统运行 20 次每次终止演化后最佳有效个体代表的系统性能(运行时间)，从图中可以看出算法有很好的稳定性.

表 1　10 节点系统数据表

Tab. 1　Attribute data of a 10-node system

BSB 序号	软件代价	软件时间/ms	硬件代价	硬件时间/ms	调用次数
1	24	526	166	318	65
2	45	209	158	133	39
3	58	511	201	255	77
4	23	366	152	242	62
5	80	21	351	6	30
6	55	178	194	92	59
7	24	92	162	57	98
8	237	204	2342	8	14
9	67	277	239	117	68
10	59	317	261	122	77

表 2　10 节点间的数据传递

Tab. 2　Communication data of a 10-node system

	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10
B1	0	5	20	19	0	0	3	16	0	14
B2	0	0	0	7	14	18	0	13	0	2
B3	0	0	0	0	0	0	0	4	19	0
B4	0	0	0	0	0	0	0	0	0	7
B5	0	0	0	0	0	5	0	0	0	0
B6	0	0	0	0	0	0	0	0	11	0
B7	0	0	0	0	0	0	0	0	10	0
B8	0	0	0	0	0	0	0	0	0	19
B9	0	0	0	0	0	0	0	0	0	13
B10	0	0	0	0	0	0	0	0	0	0

表 3　模拟退火算法与本文算法的比较实验结果

Tab. 3　Results comparison of SA and GA algorithm

节点数	成本约束	算　法	系统代价	系统总时间/ms	优化加速比	算法耗时/s
20	2000	GA(本文)	1925	390844	36.69%	0.084
		SA	1925	390844	36.69%	0.73
30	3000	GA(本文)	2977	725484	35.36%	0.21
		SA	2977	725484	35.36%	1.72
50	5000	GA(本文)	4997	1088002	35.73%	1.67
		SA	4997	1088002	35.73%	29.54
100	10000	GA(本文)	9982	2193015	34.01%	16.87
		SA	9936	2201721	33.75%	79.69
200	30000	GA(本文)	29982	3768331	43.23%	256.78
		SA	29963	3797538	42.79%	789.32

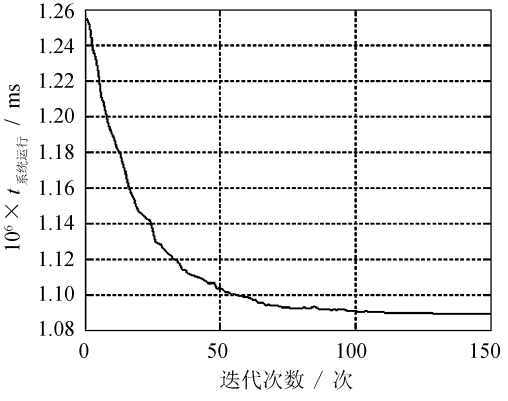

图 6 广义目标函数收敛曲线

Fig. 6 Convergence of generalized object function

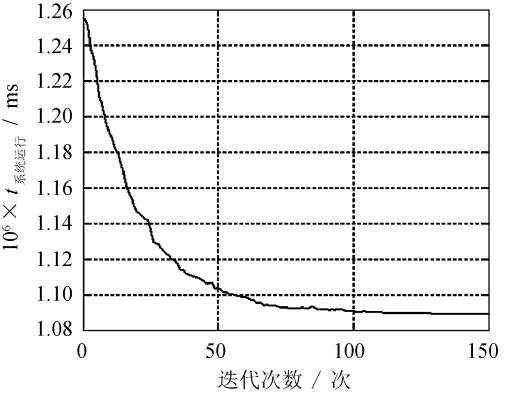

图 7 随机运行 20 次的每次结果曲线

Fig. 7 Results of 20 stochastic runnings

通过调整 $t_{s\to m}$、$t_{m\to s}$、$t_{h\to m}$ 和 $t_{m\to h}$ 的数值,可以考察系统在不同数据通讯开销下的划分结果变化. 轻度数据通讯开销下的划分结果,硬件实现的 BSB 分布较离散,算法尽可能选择具有最大加速贡献的 BSB 块硬件化,随着数据通讯开销的加大,划分结果中硬件化的 BSB 块分布的离散性将逐渐降低. 表 4 列出了 20 节点示范系统在不同数据通讯开销下的划分结果.

表 4 不同通信开销下的划分结果

Tab. 4 Partitioning results under various communication cost

类型	系统代价	系统时间/ms	算法时间/ms	加速比/%	划分结果
轻度	1964	266643	600	44.53	0 1 0 0 0 1 0 0 0 0 1 0 0 1 1 0 1 0 0 1
中度	1925	390844	720	36.69	0 0 0 0 1 1 0 0 0 1 0 1 0 0 1 0 0 1 1 0
重度	1982	460838	1650	34.95	0 0 0 0 0 0 1 1 1 0 0 0 0 1 1 1 1 1 0 0

5 总结

本文在分析嵌入式系统软硬件划分问题的基础上,提出了一个基于基本调度块图的软硬件划分模型,给出了一个基于遗传算法的软硬件划分算法并进行了算法实验. 实验结果表明本文提出的算法能有效地完成软硬件划分,并具有较好的计算稳定性. 不同数据通讯开销条件下的算法实验,验证了通讯开销对软硬件划分结果的影响. 今后的工作主要在以下几个方面:对于大规模的系统,应重点提高划分算法的执行效率,提高系统功能描述从程序设计语言到控制数据流图以及基本调度块描述的转化效率以及提高基本调度块属性数据获取的效率. 由于功能的硬件实现着眼于速度、成本、功耗等不同方面往往有不同的实现方案,后续工作将探索在划分算法中实现划分过程对多种硬件实现方式的选择.

参 考 文 献

[1] 庄镇泉,胡庆生编著. 电子设计自动化[M]. 北京:科学出版社,2000.

[2] 王志华,邓仰东编著. 数字集成系统的结构化设计与高层次综合[M]. 北京:清华大学出版社,2000.

[3] Ernst R. Codesign of embedded systems: sta-

tus and trends[J]. IEEE Design&Test of Computers,1998,45-54.

[4] Ernst R, Henkel J, Benner T. Hardware-Software cosynthesis for microcontrollers[J]. IEEE Design & Test of Computers,1993:64-75.

[5] Gupta R K, De Micheli G. System synthesis via hardware-software codesign[R]. Technical Report, Computer Systems Laboratory, Stanford University, October 1992: CSL-TR-92-548

[6] Peter Voigt Knudsen, Jan Madsen. PACE: A dynamic programming algorithm for hardware software partitioning[A]. 4th International Workshop on Hardware/Software Co-Design (Codes/CASHE'96) March[C]. Pittsburgh, Pennsylvania. 1996, 18-20.

[7] Kalavade A, Lee E A. The extended partitioning problem: hardware/software mapping and implementation-bin selection[A]. Rapid System Prototyping, Proceedings Sixth IEEE International Workshop[C]. 1995,7-9.

[8] 陈国良,王煦法,庄镇泉等编著. 遗传算法及其应用[M]. 北京:人民邮电出版社,1999.

[9] Sri Parameswaran, *et al*. Profiling in the ASP codesign environment[J]. Journal of Systems Architecture,2000,46:1263-1274.

[10] Harkin J, McGinnity T M. Partitioning methodology for dynamically reconfigurable embedded systems[J]. IEE Proc.-Comput. Digit. Tech,2000,147(6):391-396.

[11] Edwards M D, Forrest J. Software acceleration using programmable hardware devices [J]. IEE Proc.-Comput. Digit. Tech, 1996, 143(1).

HW-SW Partitioning Based on Genetic Algorithms

ZOU Yi, ZHUANG Zhen-quan, YANG Jun-an

(*Dept. of Electronic Sci. & Tech., USTC, Hefei* 230026, *China*)

Abstract: HW-SW partitioning is an important problem in HW-SW codesign of embedded systems. We established an HW-SW partitioning model based on the system's Basic Scheduling Block (BSB) graph and proposed an genetic partitioning algorithm. By adopting an adaptive fitness function definition and a novel evolving strategy, we enhanced the stability, efficiency and result quality of our partitioning algorithm. Experiment results show that the algorithm's effectiveness in solving the HW-SW partitioning problem.

Key words: HW-SW codesign; HW-SW partitioning; genetic algorithm; adaptive evolution