文章编号：1001-2486（2002）02-0064-05

遗传算法在软硬件划分中的应用*

刘功杰，张鲁峰，李思昆
（国防科技大学计算机学院，湖南 长沙 410073）

摘 要：软硬件划分是软硬协同设计中的一个关键问题。针对单处理器嵌入式系统，给出了基于遗传算法的解决方案，并引入了模拟退火和按概率选择两种技术。结果表明，算法有效地解决了软硬件划分问题，稳定性好、效率高，模拟退火和按概率选择的引入，进一步提高了算法效率，保证了算法的自适应性及结果的全局最优性。

关键词：遗传算法；软硬件划分；软硬协同设计
中图分类号：TP302 **文献标识码**：A

Research on the Use of Genetic Algorithm in HW-SW Partition

LIU Gong-jie，ZHANG Lu-feng，LI Si-kun
（College of Computer，National Univ. of Defense Technology，Changsha 410073，China）

Abstract：Hardware-software partitioning is a key problem in Hardware-software co-design of embedded system. This paper describes an approach based on genetic algorithm. Two methods，simulated annealing and selecting an individual in light of the probability，are used to enhance the algorithm's self-adaptability，efficiency and global optimization. It is validated via the partitioning results that the approach is stable，effective and efficient.

Key words：genetic algorithm；hardware-software partition；hardware-software co-design

嵌入式系统一般有软件和硬件两种基本实现方式，同硬件相比，软件容易修改、成本较低，而硬件可以提供更高的性能。软硬件划分问题，就是在嵌入式系统的高层次设计阶段，根据设计约束及系统各个部分的特点，确定各部分的软件或者硬件实现方式，以获得高性能、低成本的优化设计方案。软硬件划分是嵌入式系统的软硬协同设计中的一个关键问题。

常用于软硬件划分的算法有整数规划[1]、混合线性规划[7]、启发式算法[6]等。对于规划类算法，难以给出明确的目标函数，约束条件太多，不易使用，而且当目标系统的结点很多时，算法的计算时间会很长。启发式算法对启发式规则的要求很高，结果很容易受其影响。这几种算法还有一个共同的弊端：容易陷入局部最优。

1 软硬件划分模型

一个系统使用任务流图（task graph）（参见图 1）来描述，任务流图是有向无环图，只有一个起始节点和一个终止节点，两个节点之间最多只有一条边。图中的一个节点表示系统的一个任务，边表示任务之间的控制关系或数据流向，每条边的终点任务必须在此边的始点任务完成之后才可以开始执行。每个节点包含其软件、硬件代价信息，边的权重代表两个节点之间的通讯开销。

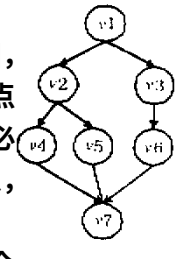

图 1 任务流图
Fig.1 Task graph

任务流图是系统的行为级描述，主要描述系统中任务间的控制、数据关系及每个任务的代价信息，而与系统实现时采用什么样的体系结构无关。

定义 1 目标系统的体系结构[1]：包括一个处理器 P，一个 ASIC $H=\{h_1,\cdots,$

* 收稿日期：2001－10－09
基本项目：国家 863 计划资助项目（863－SOC－Y－3－2－1）
作者简介：刘功杰（1977—），男，硕士生。

$h_k\}$，存储器和总线，目标系统的组件集合定义为 $A = H \cup \{P\}$。其中，h_1，…，h_k 是 ASIC 中具有不同功能的硬件电路，参见图 2。

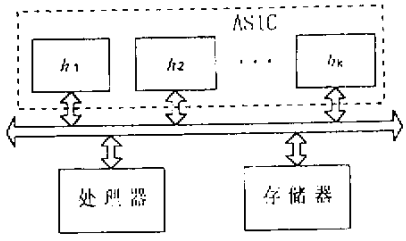

图 2　目标系统结构简图

Fig.2　The architecture of the target system

定义 2　一个系统定义为一个三元组 $S = \{V, R, M\}$：$V = \{v_1, \cdots, v_N\}$ 是系统中的节点集合，N 为系统的节点总数；$R = \{r_1, \cdots, r_N\}$，$r_i \subset A = H \cup \{P\}$ 代表节点 i 的实现方式集合，P 表示用软件实现，h_i 表示用硬件 i 实现；二维矩阵 $M \subset V \times V$ 代表节点之间的连接关系。

节点 i 应该包含以下信息：用方式 k（$0 \leqslant k < | r_i |$）实现时的代价（软件代价主要由所占内存决定，硬件代价主要由硬件面积决定）和用方式 k（$0 \leqslant k < | r_i |$）实现时的处理时间。需要注意的是，硬件代价与软件代价需要使用统一的度量，而每个节点的硬件代价应该比其软件代价大得多。

一个设计 ΓA 是指系统 S 在目标系统的体系结构 A 上的实现。检验一个设计好坏的参数包括系统实现的总代价 $Cost$（S）和系统的总处理时间 $Time$（S），设计的约束条件是系统的最大处理时间 $\mathrm{MAX^T}$（S）。系统实现的总代价 $Cost$（S）定义为所有节点代价之和。系统的总处理时间 $Time$（S）定义为系统的最大可能完成时间。计算 $Time$（S）时，本文采用如下假设：

·处理器处理软件任务时采用先来先服务的策略；

·不同的硬件单元可以并行处理任务；

·只计算采用不同实现方式的相邻节点之间的通讯开销，如果相邻节点都使用硬件或都使用软件实现，则忽略两者之间的通讯开销。

定义 3　软硬件划分问题：在满足性能约束的条件下，寻找一个从节点集合 V 到设计 ΓA 的映射：$V \rightarrow A$，使得系统的代价 $Cost$（S）最小。整个问题转化为一个规划问题：

目标函数　$\mathrm{Min}\ Cost\ (S)$　　(1)

约束条件　$Time\ (S) \leqslant \mathrm{MAX^T}\ (S)$　　(2)

2　软硬件划分算法

设计的遗传算法，主要包括以下几个步骤：(1) 种群初始化，种群规模为 $2N$；(2) 使用设定的选择策略选择两个父个体，进行杂交、变异生成两个子个体，重复此过程直到生成 N 个子个体；(3) 用 N 个子个体替换父代中最差的 N 个父个体；(4) 重复 (2)、(3) 两步，直到满足终止条件。

2.1　目标函数和约束条件

软硬件划分是一个约束极小化问题，对于约束优化问题，遗传算法常采用惩罚函数法，即在目标函数中加上一个反映解是否位于约束集内的惩罚项，来构成一个广义目标函数，从而使得算法在惩罚项的作用下找到问题的最优解。对约束极小化问题，惩罚函数应对非可行解产生一个正的惩罚，而对可行解则不产生惩罚或惩罚尽量小[4]。

定义的广义目标函数为：

$$\mathrm{Min}\quad Cost\ (S) = a \times e^{\frac{Time(S) - \mathrm{MAX^T}(S)}{T_E} \cdot | Time\ (S) - \mathrm{MAX^T}\ (S) |} + b \times Cost\ (S) \qquad (3)$$

该目标函数包含了公式 (1) 和公式 (2) 两部分的信息。前半部分是惩罚函数，惩罚不满足性能约束的不可行解，绝对值项是考虑到设计应该恰好满足时间约束，而不是速度越快越好（速度快往往意味着代价高）；后半部分包含了原目标函数的信息。参数 a 和 b 决定了两部分的比重，由于系统性能决定设计是否可行，所以应保证 a 大于 b [3]。T_E 是控制惩罚项变化的退火温度参数，它随着演化过程的进行逐渐变小，种群每经过一代演化后，对 T_E 进行修正，使 $T_E = 0.95 \times T_E$，这样，随着演化的进行，目标函数对不满足约束的非可行解的惩罚越来越大，而对于可行解的惩罚越来越小。

2.2 编码问题

遗传算法求解问题不能直接作用在问题的解空间上，首先需要对问题的解进行编码。本文采用整数向量编码，整数向量 $B = (b_1, \cdots, b_N)$ 表示系统的一种设计，同时也是遗传操作的对象，其中 N 为任务流图的节点数。

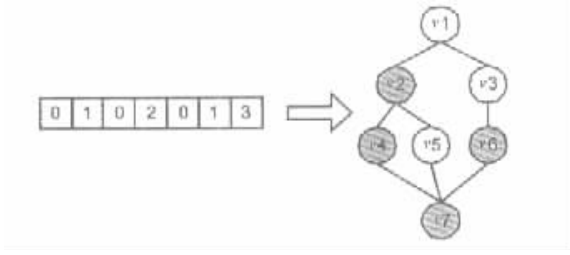

图 3 编码与系统实现的关系

Fig.3 The relation of coding and system's implementation

$$b_i = \begin{cases} 0 & \text{节点 } i \text{ 用软件实现} \\ k & \text{节点 } i \text{ 用其实现集合的第 } k \text{ 种硬件实现} \end{cases} \quad 0 < k < |r_i| \tag{4}$$

例如，对图 1 所示系统的任务流图，如果编码向量是（0，1，0，2，0，1，3），则它所表示的含意如图 3 所示（白色节点表示用软件实现，灰色节点表示用硬件实现）。

2.3 种群的初始化及算法过程中控制参数的设定

（1）随机生成初始种群；

（2）种群规模设定为系统节点数 N 的 2 倍，变异概率 p_m 为 0.001；子代规模为 N（每一代进化生成 N 个子个体）；

（3）替换策略：每次用生成的 N 个子个体替代父代中最差的 N 个父个体；

（4）终止条件：当种群中绝大多数个体所代表的解的代价和执行时间都基本相同时，则终止演化。具体操作是：进行完一代演化之后，随机选择 k 个个体对（$S_{1,1}$ $S_{1,2}$），（$S_{2,1}$ $S_{2,2}$），…，（$S_{k,1}$ $S_{k,2}$），设每个个体所代表的解的系统总处理时间为 $time$（$S_{i,j}$），系统代价为 $cost$（$S_{i,j}$），演化的终止条件为：

$$\sum_{i=1}^{k} \left| time(S_{i,1}) - time(S_{i,2}) \right| / k \leqslant x_t \text{ 且 } \sum_{i=1}^{k} \left| \cos t(S_{i,1}) - \cos t(S_{i,2}) \right| / k \leqslant x_c \tag{5}$$

其中，k，x_t，x_c 是设计时确定的常量参数，当 $N \leqslant 150$ 时，可以取 $k = 15$，当 $N > 150$ 时，可取 $k = \lceil N/10 \rceil$，x_t，x_c 需要根据系统的具体时间、代价参数确定，本文取 $x_t = 0.5$，$x_c = 0.5$。这种终止策略具有很强的自适应性，既可以避免迭代次数过多而进行无用搜索，又可以防止迭代次数太少而达不到最优解。x_t，x_c 可以控制目标种群中个体之间的差异大小和算法的搜索时间。

2.4 父个体选择策略

采用基于局部竞争机制的选择策略：随机在种群中选择 k 个个体进行比较，“生存能力”最强（目标函数值最小）的个体被选择作为生成下一代的父个体。本文取 $k = 2$。

在进化的初始阶段，种群中个体的多样性是很重要的，不然的话，容易导致过早收敛和停滞。为了保证这种多样性，从两个个体中选择父个体时，不是绝对按其目标函数值决定取舍，而是按照一定的概率选择：假设选择了两个个体 X，Y，定义 $g(X,Y) = \dfrac{Cost'(X) - Cost'(Y)}{Cost'(X) + Cost'(Y)}$，$g(X,Y)$ 表示个体 X 相对于个体 Y 的优劣程度，如果 $g(X, Y) > 0$，说明 X 比 Y 差，如果 $g(X, Y) < 0$，说明 X 比 Y 好；定义 X 被选择的概率为

$$Select(X)=\begin{cases}0 & g(X,Y)>0.1\\ 0.5-10\times g(X,Y)\\ 1 & g(X,Y)<-0.1\end{cases} \tag{6}$$

当 X 比 Y 好时，$Select(X)>0.5$，X 优势越明显，被选择的概率越大；反之如果 X 比 Y 差，则 $Select(X)<0.5$，X 越差，被选择的概率越小；如果 $Cost'(X)=Cost'(Y)$，则 X、Y 的选择概率都为 0.5。

按概率选择的策略，在演化的开始阶段，有助于保证解的多样性，但当演化进行到一定阶段的时候，反过来又会阻碍解的收敛，所以随着演化的进行，应该逐渐削弱这种按概率选择的作用。因此，引入了模拟退火中温度控制的技术。

假设演化目前进行到第 m 代，则概率选择函数修改为

$$Select(X)=\begin{cases}0 & g(X,Y)>0.1/m\\ 0.5-10\cdot m\cdot g(X,Y)\\ 1 & g(X,Y)<-0.1/m\end{cases} \tag{7}$$

这样，如果 $Select(X)<0.5$，则使其逐渐趋向于 0；如果 $Select(X)>0.5$，使其逐渐趋向于 1。

上述概率选择函数的曲线及其在进化过程中的变化如图 4 所示。引入概率选择和模拟退火两个概念后，算法的解更趋近于最优，算法变得更加稳定，而且收敛速度明显提高。通过多次实验统计表明，使用 2.3 节所述的终止条件，并引入模拟退火（在目标函数和选择父个体两个问题都用到了模拟退火的有关技术）和概率选择后，算法的速度提高了一倍左右。

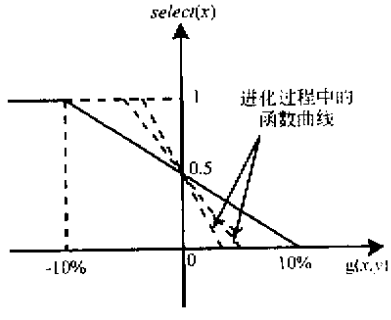

图 4 概率选择函数曲线

Fig.4 The curve of selection function by probability

2.5 演化策略

2.5.1 杂交算子

采用多点杂交，杂交点的个数与系统节点数成正比，设系统的节点数为 N，杂交点数取「N/K」即在两个父向量上随机选择「N/K」个点，然后交换这两个向量的对应点的元素产生两个子向量（见图 5.）。其中 k 是需要设定的参数，k 取在 10 ~ 15 之间比较合适，本文取 $k=12$。

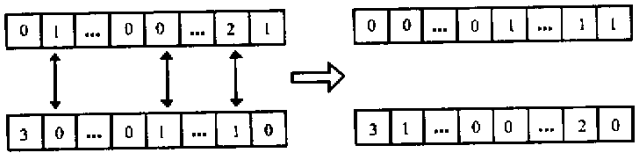

图 5 多点杂交示意图

Fig.5 Multiple-point crossover

2.5.2 变异算子

算法的变异算子比较简单，对杂交产生的子代个体向量的每一位以变异概率 $p_m=0.001$ 加上一个干扰量。设向量第 i 位的值为 k，相应节点有 $|r_i|$ 种实现方式。如果 $k=0$，则以概率 p_m 将该位的值加 1；如果 $k=|r_i|-1$，则以概率 p_m 将该位的值减 1；如果 $0<k<|r_i|-1$，则以概率 p_m 将该位的值 ± 1（+1 还是 −1 也是随机的）。

3 算法结果及总结

使用本文的算法针对随机生成的任务流图进行了划分。任务流图的结点数、时间约束、每个节点的信息、结点之间的连接关系都是随机生成的。本文与文献[3]的划分结果的比较见表 1，这些数据是针对与文献[3]中完全相同的任务流图进行划分获得的。

表1 本文与文献[3]中的结果比较

Tab.1 Comparing the results of this paper and the Ref [3]

例	节点数	时间约束	方 法	总处理时间	系统总代价	算法执行时间(ms)
1	42	450	SA [3]	448	572	2457
			GA [3]	449	580	1313
			GA (本文)	450	567	557
2	50	500	SA [3]	498	630	2896
			GA [3]	495	637	1289
			GA (本文)	497	630	672
3	65	500	SA [3]	497	613	3102
			GA [3]	500	604	1678
			GA (本文)	500	600	934

以其中的例1为例，任务流图共有42个节点，时间约束为450，SA[3]和GA[3]表示文献[3]中分别使用模拟退火（SA）和遗传算法（GA）的划分结果，GA（本文）表示使用本文的遗传算法的划分结果。由于文献[3]没有考虑通讯开销，为了使实验数据具有可比性，我们在实现时也忽略了通讯开销。表2列出了对另外几个随机实例的划分结果。

通过比较，并经过多次实验验证，得出结论：(1) 设计的算法可以很好地解决软硬件划分问题；(2) 终止条件、目标函数的改进，按概率选择和模拟退火技术的引入，使得算法效率有了明显提高，也使得算法更加稳定、结果更优。

表2 随机实例划分结果

Tab.2 Partitioning results of the random instances

例	节点数	时间约束	总处理时间	系统总代价	算法执行时间(ms)
4	89	890	890	1020	3222
5	109	1090	1090	1400	5003
6	178	1780	1779	2267	19359

需要说明的是，本文采用的单处理器目标模型比较简单，但这种结构仍然是目前使用最多的结构，而且本文的算法对于有关分布式嵌入式系统的研究工作很有意义。

参考文献：

[1] Ranf Niemann, Peter Marwedel. Hardware-Software Partitioning using Integer Programming [C]. Proceedings of European Design and Test Conference, 1996.

[2] Robert P Dick, Niraj K Jha. MOGAC: A Multiobjective Genetic Algorithm for Hardware-Software Co-Synthesis of Distributed Embedded Systems [J]. IEEE Trans. on Computer-Aided Design of integrated circuits and systems, 1998, 17 (10).

[3] 张鲁峰，李思昆，刘功杰. 嵌入式系统软硬件划分方法研究 [J]. 计算机应用，2000 增刊.

[4] 潘正君，康立山 等. 演化计算 [M]. 北京：清华大学出版社，南宁：广西科学技术出版社，1998.

[5] Yen T-Y. Hardware-software cosynthesis of distributed embedded systems [D]. PhD dissertation, Dept. Electrical Engineering, Princeton University, Princeton, NJ, June 1996.

[6] Kalavade A. System-level Co-design of mixed hardware – software systems [D]. PhD dissertation, University of California, Berkeley, CA, September 1995.

[7] Prakash S, Parker AC. Synthesis of Application Specific Heterogeneous Multiprocessor Systems [J]. Journal of Parallel and Distributed Computing, 1992 (10).