基于路径的软硬件划分算法*)

朱智林[1,3]　韩俊刚[2]　陈　平[3]

(山东工商学院计算机系　烟台 264005)[1]　(西安邮电学院计算机系　西安 710065)[2]

(西安电子科技大学软件工程研究所　西安 710071)[3]

摘　要　软硬件划分是嵌入式系统中的一个关键问题。本文给出了一种贪心算法来搜索问题的最优解。本算法未考虑相邻任务之间的通讯开销。实验结果表明，任务数目的多少对加速比影响不大，影响加速比的关键因素就是硬件的有效面积。

关键词　软硬件划分，嵌入式系统，Hot路径，算法

Hardware/Software Partitioning Algorithm Based on Path

ZHU Zhi-Lin[1,3]　HAN Jun-Gang[2]　CHEN Ping[3]

(Department of Computer, Business Technology College of Shandong, Yantai 264005)[1]

(Department of Computer, Xi'an Institute of Post and Telecoms, Xi'an 710065)[2]

(Institute of Software Engineering, Xidian University, Xi'an 710071)[3]

Abstract　Hardware/software partitioning is a key problem in embedded systems. The problem is modeled into a 0-1 knapsack, and an greedy algorithm is given for approximation optimally sloving the problem , the communication overhead is not taken into account in this algorithm. Experimental results show that the algorithm efficient is effected by the available hardware space not by the number of task block.

Keywords　Hardware/Software partitioning, Embedded system, Hot path , Algorithm

1　引言

在嵌入式应用系统中，软硬件划分对于将应用需求规范转换为嵌入式的软件和硬件实现，并且能够满足应用的需求和系统性能的约束，有着十分重要的作用。一般而言软件实现相对比较灵活和廉价，硬件实现可提高系统速度，但成本比较高。因此，软硬件划分对于改善系统的性能和功耗有着重要的影响[1,2]，高效的软硬件划分技术可以获得较高的性价比。

常用于软硬件划分的算法有面向硬件和面向软件的启发式方法。文[3~5]是面向硬件的方法。其基本思想就是，最初系统中的所有任务全部以硬件实现，然后逐步将系统中的部分硬件实现的任务用软件实现方式来取代，以降低功耗和面积，经过反复的叠代替换，直到不能满足系统性能约束为止；文[2,6,7]是面向软件的方法，其基本思想就是，系统最初所有的任务均以软件实现，然后将部分软件任务以硬件方式来替代，以改善系统的速度，直到能够满足系统的时间约束为止。另外，还有注重于算法本身的许多方法，如进化算法，整数规划算法，模拟退火算法，动态规划算法以及蚁群算法。在不同应用系统中，其相应的体系结构以及代价函数构造方法度有所区别，各种算法适用于不同特征的应用系统，以获得具体应用执行的最小时间的优化解。软硬件划分问题是一个NP完全问题[8]。

随着程序分析技术的日益成熟，许多强大的分析工具[9]已经成型，可以记录在运行过程中数据流图上的每个路径的具体状况。一般而言，一个访问频率高的路径，如循环，在给定应用的整个执行过程中是起主导作用的。因此在软硬件划分中，可以优先考虑高频率访问路径。本文在基于文[9]分析工具获得高频率访问路径的基础上，对高频访问路径进行软硬件划分。与启发式方法不同，本文给出了一种贪心算法来搜索问题的最优解，本算法未考虑相邻任务之间的通讯开销。

2　软硬件划分模型

对于一个系统，采用任务执行流图描述，如图1所示，图中每个节点就是一个任务，每个任务都是粒度大小确定的基本调度模块，简称块，这些块可用硬件方式实现也可用软件方式实现。高频访问路径 P 是由块 $B_1, B_2, \cdots, B_n$ 组成，即：$P=\{B_1, B_2, \cdots, B_n\}$，对于任意的 $i\in[1, n-1]$，B_{i+1} 为 B_i 的后继。Ts_i 表示用软件实现 B_i 时运行所需要的时间，Th_i 表示用硬件实现 B_i 时运行所需要的时间，a_i 表示 B_i 用硬件实现后，所占用相应的面积。所有的 Ts_i，Th_i 和 a_i 可以通过某个具体的工具获得，如文[10]。H 和 S 分别表示由硬件和软件实现任务的两个集合，对于一个确定的 P，寻找一个满足 $P=H\cup S, H\cap S=\Phi$ 的软硬件划分，以获得良好的系统加速性能，同时实现硬件面积的最小化。

软硬件划分问题可以转换为在一个有向图上寻找一条最短路径的问题，该有向图具有这样的特征，即只有唯一的入口和唯一的出口，这条最短路径应该满足相应的硬件面积约束

*)本课题受到国家自然科学基金项目"用于芯片系统验证的定理引擎研究"(No. 90207015)和"十五"国防预研课题"嵌入式系统综合设计技术"(No. 417010401)的支持。朱智林　博士研究生，主要研究方向为面向对象技术，嵌入式系统设计；韩俊刚　博士生导师，主要研究方向为IC设计的形式化验证；陈　平　博士生导师，主要研究方向为面向对象技术，软件工程。

条件。图2是通过程序分析工具获得的具有4个任务块的模型，如果相邻的块以不同的方式实现，则它们之间存在一定的通信开销，设 $B_i'(B_i)$ 表示块 B_i 用硬件(软件)实现，若 B_i 和 B_{i+1} 以不同的方式实现，则假定以 $c_{i,i+1}$ 表示 B_i 和 B_{i+1} 之间的通信开销；若相邻块实现方式相同，则假定它们之间的通信开销为0。本文假设，对于 $1\leqslant i<n$，有 $s_i>>h_i+c_{i,i+1}$，以保证硬件实现相对于软件实现可以加快运行速度。

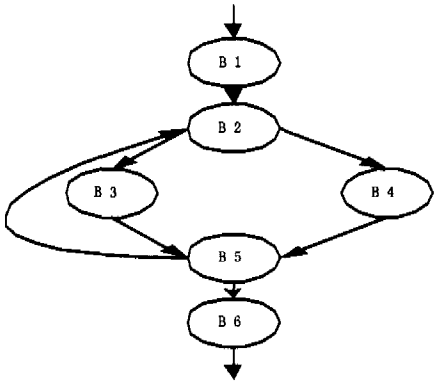
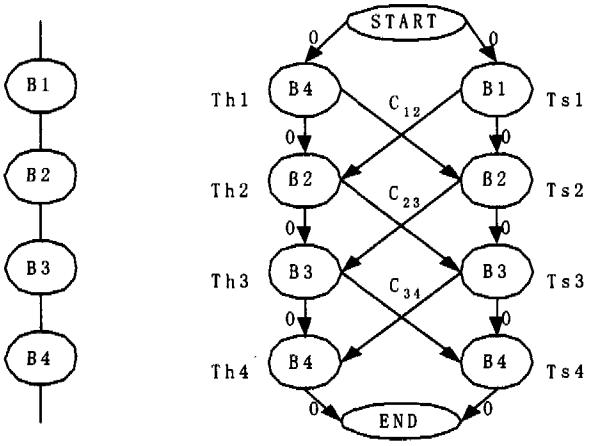


图1 任务执行流图



(a) (b)

图2 (a) 4个任务块构成的高频路径
(b) 高频路径的软硬件划分模型

设 $(x_1,x_2,\cdots,x_n)$ 是软硬件划问题的一个可行解，其中 $x_i\in\{0,1\}$，$x_i=1(x_i=0)$ 表示 B_i 属于硬件(软件)集合。对于 $1\leqslant i\leqslant n$，设 $T(x_1,x_2,\cdots,x_n)$ 为划分 $(x_1,x_2,\cdots,x_n)$ 相应的运行时间，可以形式化描述为：

$$T(X_1,X_2,\cdots,X_n)=\sum_{i=1}^{n}(X_i\cdot Th_i+(1-X_i)\cdot Ts_i)+\sum_{i=1}^{n}|X_i-X_{i+1}|\cdot C_{i,i+1}$$

已知硬件的有效面积为A，软硬件划分问题则可以描述为如下最小化问题模型：

ρ：使得 $T(x_1,x_2,\cdots,x_n)$ 最小值，并且满足 $\sum_{i=1}^{n}a_i\cdot x_i\leqslant A$

若通信开销忽略不计，即对于 $1\leqslant i\leqslant n-1$，$c_{i,i+1}=0$，则有：

$$T(x_1,x_2,\cdots,x_n)=\sum_{i=1}^{n}(x_i\cdot Th_i+(1-x_i)\cdot Ts_i)+\sum_{i=1}^{n}|x_i-x_{i+1}|\cdot c_{i,i+1}=\sum_{i=1}^{n}(x_i\cdot Th_i+(1-x_i)\cdot Ts_i)=\sum_{i=1}^{n}Ts_i-\sum_{i=1}^{n}(Ts_i-Th_i)\cdot x_i=TS-\sum_{i=1}^{n}(Ts_i-Th_i)\cdot x_i$$

对于一个确定的应用系统而言，所有任务模块都用软件实现所需要的执行时间 Ts 是一个确定的常数，从而上述问题 ρ 就转换为下列这样一个最大化问题 ρ' 模型，即：

使得 $\sum_{i=1}^{n}(Ts_i-Th_i)\cdot x_i$ 最大，并且满足 $\sum_{i=1}^{n}a_i\cdot x_i<A$

上述的最大化问题 ρ' 显然是一个标准的NP完全的0-1背包问题[11]，其中A对应于背包的容量，块 $B_i(1\leqslant i\leqslant n)$ 就是相应的0-1背包问题中要装背包的物品 i。因此问题 ρ' 也是NP完全问题。不过，对问题 ρ 的解，完全可以增加通信开销就可以推导出问题 ρ 的解来。所以问题 ρ' 的解就是问题 ρ 近似解，特别地，对于通信开销比较小的 $c_{i,i+1}(1\leqslant i\leqslant n-1)$，求解问题 ρ' 的算法通常可以用来作为求解问题 ρ 的算法。

3 软硬件划分算法

3.1 划分问题描述

已知背包的容量为 C，物品的集合 $S=\{1,2,3,\cdots,n\}$，每个物品的重量为 w_i，价值为 b_i，该问题就是寻找一个子集 S' 包含于 S 中，使得在满足 $\sum w_i\leqslant C$ 的条件下，$\sum b_i$ 最大，其中属于 S'，这就是标准的0-1背包问题，形式化描述如下：

0-1背包问题就是：使得 $\sum_{i=1}^{n}b_i\cdot x_i$ 最大，并且满足 $\sum_{i=1}^{n}w_i\cdot x_i<C$

其中：$x_i\in\{1,0\}$，$i=1,2,3,\cdots,n$

$x_i=1$ 表示物品 i 装入了背包，而 $x_i=0$ 则相反。

设物品按照它们价值重量比的降序进行排序，价值重量比，$e_i=\frac{b_i}{w_i}$，则：

$$\overline{B_j}=\sum_{i=1}^{j}b_i,\overline{W_j}=\sum_{i=1}^{j}W_i$$

记背包剩余的容量 r 为：$r=C-\overline{W_{k-1}}$

文[2]给出了0-1背包问题解的上界 u 和下界 l 分别为：

$$u=\lceil\overline{B_{k-1}}+r\cdot\frac{b_k}{w_k}\rfloor$$

$$l=\overline{B_{k-1}}$$

对于问题 ρ'，我们定义块 B_i 的效益为 $e_i=(s_i-h_i)/a_i$。我们假设所有块的排列是以 e_i 非递增序的，即对于所有的 $1\leqslant i\leqslant n-1$，均存在 $e_i\geqslant e_{i+1}$，则问题 P' 的解 $(\underline{x_1},\underline{x_2},\cdots,\underline{x_n})$ 可以采用贪心算法获得0-1背包的解的方式获得，即以块的效益的递减序依次装入背包，直到没有合适大小的块可以装入背包为止，从而可以得到问题 ρ 的近似解。

3.2 贪心算法

```
Algorithm Greedy
{
   按照块效益的降序对将所有块进行排序;
   area_used=0;
   for (i=1;i<=n;i++)
      if (area_used+ai)<=A
        { x_i=1;area_used+=ai}
      else x_i=0;
}
```

在任务块效益无序的情况下，算法Greedy的时间复杂度是 $O(n\log n)$。

4 实验结果

一般划分问题目前尚不存在获得广泛认同的判定标准[12]，与文[13]的实验方法类似，s_i 取[100,1000]之间的随机整数，h_i 取[10,500]之间的随机整数，a_i 取[1,50]之间的随机整数，算法对同一组数据分别取硬件总面积 A 的0.1,0.2,0.3,…,0.9倍进行比较，硬件总面积为 $A=\sum_{i=1}^{n}a_i$。

为了说明实验仿真的结果，定义加速比指标 λ 如下：

$$\lambda=(1-\frac{TH+TH_{\max}}{TS})$$

其中：$TS=\sum_{i=1}^{n}Ts_i$ 是全部任务均由软件实现所用的执行时间；$TH_{\max}=\max_{i=1}^{n}\{Th_i\cdot x_i\}$ 是问题 ρ' 解中由硬件实现的任务，它们所需要的最长的硬件执行时间，假设硬件实现的任务区不是并行执行的；$TH=\sum_{i=1}^{n}Th_i\cdot x_i$ 是问题 ρ' 解中由硬件实现的全部任务执行所需要的时间之和。

算法分别取任务数 $n=30,50,70$ 三组进行比较，仿真结果如图 3 所示，仿真结果表明任务数目的多少对加速比影响不大，影响加速比的关键因素就是硬件的有效面积。

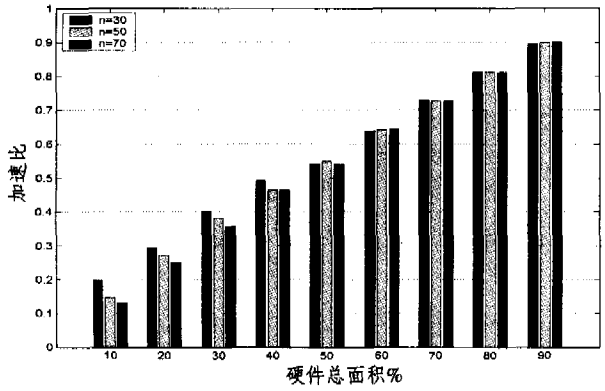


图 3　算法仿真结果比较

小结　对于嵌入式系统设计而言，软硬件的划分对系统性能有着重要的影响。本文给出了在高频访问路径的基础上，采用贪心算法中经典的 0-1 背包模型进行求解软硬件划分问题是一种新的思路，在忽略软硬件任务间通信开销的基础上，可以在时间复杂度为 $O(n\log n)$ 的情况下获得问题的近似最优解。将通信开销纳入求解的空间内是下一步工作需要考虑的内容。

参 考 文 献

1　Ernst R, Henkel J, Benner T. Hardware-Software Cosynthesis for Micro-controllers. IEEE Design and Test of Computer, 1993, 10(4): 64～75

2　Harkin J, McGinnity T M, Maguire L P. Partitioning methodology for dynamically reconfigurable embedded systems. IEE Proceedings Computers and Digital Techniques, 2000, 147(6): 391～396

3　Niemann R, Marwedel P. Hardware/software partitioning using integer programming. In: Proc. of IEEE/ACM European Design Automation Conf. (EDAC), 1996. 473～479

4　Gupta R, Micheli G D. Hardware-software cosynthesis for digital systems. IEEE Design and Test of Computers, 1993, 10(3): 29～41

5　Gupta R K, Coelho C, Micheli G D. Synthesis and Simulation of Digital Systems Containing Interacting Hardware and Software Components. In: Proc. of 29th ACM, IEEE Design Automation Conf. 1992. 225～230

6　Vahid F, Gajski D D, Gong J. A binary-constraint search algorithm for minimizing hardware during hardware/software partitioning. In: Proc. of IEEE/ACM European Design Automation Conf. (EDAC), 1994. 214～219

7　Vahid F, Gajski D D. Clustering for improved system-level functional partitioning. In: Proc. 8th IEEE/ACM Int. Symp. System Synthesis, 1995. 28～33

8　Jinwoo S, Dong-In K, Crago S P. A communication scheduling algorithm for multi-FPGA systems. In: Proc. IEEE Symp. Field-Programmable Custom Computing Machines, 2000. 299～300

9　Melski D. Interprocedural path profiling and the interprocedural express-lane transformation: [PhD thesis]. University of Wisconsin, 2002

10　Madsen J, Grode J, Knudsen P V, Petersen M E, Haxthausen A. LYCOS: The Lyngby co-synthesis system. Design Automation for Embedded Systems, 1997, I: 195～235

11　Pisinger D. Algorithms for knapsack problems: [Ph. D. Thesis]. University of Copenhagen, 1995

12　Edwards S A, Lavagno L, Lee E A, Sangiovanni-Vincentelli A. Design of Embedded Systems: Formal Models Validation, and Synthesis. Proceedings of the IEEE, 1997, 85(3): 366～390

13　Knudsen P V, Madsen J. PACE: A dynamic programming algorithm for hardware/software partitioning. In: Proc. of 4th IEEE/ACM Int. Workshop Hardware/Software Codesign, 1996. 85～92

(上接第 152 页)

少覆盖程度高的测试用例，能够部分实现自动化而不过多增加用户额外的工作量，这样的测试方法容易被已经使用 UML 的工业界采用，但可测试性要求较多，下一步的工作是使方法和工具能处理利用更多的模型信息，往实用化方向发展。

参 考 文 献

1　UML Specification 1.5. Available at: http://www.uml.org Jan 2005

2　Booch G, Rumbaugh J, Jacobson I. The Unified Modeling Language User Guide. Addison-Wesley, 2001

3　Booch G, Rumbaugh J, Jacobson I. The Unified Software Development Process. Addison-Wesley, 2001

4　Kruchten P. The Rational Unified Process -An Introduction, 2nd edition. Addison-Wesley, Reading, MA, 2000

5　Binder R V. Testing Object-Oriented System: Models, Patterns, and Tools. Addison-Wesley, 2000

6　Beck K. Extreme Programming Explained: Embrace Change. Addison-Wesley, Reading, MA, 1999

7　Booch G, Rumbaugh J, Jacobson I. The Unified Modeling Language Reference Manual. Addison-Wesley, 2001

8　Bertolino A. Software Testing research and practice. In: 10th Intl. Workshop on Abstract State Machines, 2003

9　Fraikin F, Leonhardt T. Sequence Diagram Based Test Automation. In: Proc. of the 17th IEEE Intl. Conf. on Automated Software Engineering, Edinburgh, Sep. 2002

10　Rational Rose@, Using the Rose Extensibility Interface. http://www.cs.rhul.ac.uk/CompSci/Computers/rational/pdf/rose_REI_guide/Rose_REI_guide.pdf, Jan. 2005

11　Wang Lin- Zhang, Li Xuan-Dong, Zhen Guo-Liang. Generating Test Cases from UML Activity Diagram Based on Gray-Box Method. In: 11th Asia-Pacific Software Engineering Conf. (APSEC'04)

12　Wittevrongel J. The Scentor Web Site. http://sern.ucalgary.ca/~milos/etesting/scentor, Jan. 2005

13　The AGEDIS project. http://www.agedis.de, Jan. 2005

14　Abdurazik A, Offutt J. Using UML Collaboration Diagrams for Static Checking and Test Generation. In: Proc. 3rd Intl. Conf. on the Unified Modeling Language (UML'00), York, UK, Oct. 2000. 383～395

15　Basanieri F, Bertolino A. A Practical Approach to UML-based Derivation of Integration Tests. In: Proc. of QWE2000, Bruxelles, Nov. 3T

16　Basanieri F, Bertolino A, Marchetti E. CoWTeSTt: A Cost Weighted Test Strategy. In: Proc. of ESCOM-SCOPE 2001, London, England, April 2001

17　Josuttis N M. C++标准程序库. 武汉：华中科技大学出版社，2002

18　王林章，李宣东，郑国梁. 基于 UML 协作图生成测试用例. 电子学报，2004，32(8)