一种有效的面向多目标软硬件划分的遗传算法

罗 莉[1] 夏 军[1] 何鸿君[1] 刘 瀚[2]
(国防科技大学计算机学院 长沙 410073)[1] (电子工程学院 合肥 230037)[2]

摘 要 软硬件划分是软硬件协同设计的关键技术之一,划分结果对最终的设计方案有非常重要的影响。软硬件划分根据优化目标的数量,可分为单目标划分和多目标划分。多目标划分问题是一个NP-hard问题,一般不存在传统意义上的"最优解",而是存在一组互不支配的Pareto最优解。遗传算法因其具有并行、群体搜索的特点而非常适于求解多目标优化问题。通过抽象描述将一个实际SOC设计问题转化为多目标软硬件划分问题,采用遗传算法便可获得最优设计方案。为克服过早收敛及加快搜索速度,改进了适应度函数的定义,通过自适应参数调整,加入惩罚函数的适应度定义,提高了进化速度,从而有效地获得了Pareto最优解集。在实际问题的应用中,多目标软硬件划分遗传算法是能有效求取平衡系统成本、硬件面积、功耗和时间特性的最优化方案。

关键词 软硬件划分,遗传算法,多目标划分,适应度函数
中图法分类号 TP302 **文献标识码** A

Effective Multi-objective Genetic Algorithm for Hardware-software Partitioning

LUO Li[1] XIA Jun[1] HE Hong-jun[1] LIU Han[2]
(School of Computer Science, National University of Defense Technology, Changsha 410073, China)[1]
(Electronic Engineering Institute, Hefei 230037, China)[2]

Abstract Hardware/Software partitioning is one of the critical step in Hardware/Software Codesign flow, and has very important influence on the final design. In terms of the number of optimizing objects, it can be classified as single object partitioning and multi objects partitioning. Multi objects partitioning is NP-hard problem, a multi objective partitioning algorithm usually gets irrelevant Pareto results, no traditional optimum results. Genetic algorithm benfits to solve multi objective partitioning for its parallel colony research. Fitness function was investigated and a redefined fitness function was proposed, which adopts self-adaptive parameter and penalization function to escape from the premature convergence and improve evolution speed. The practical experiment results demonstrate that this algorithm is more efficient to balance all the parameters to optimize multi system objects under some constraints, for instance, executing time, cost, hardware area power, etc.

Keywords Hardware/software partitioning, Genetic algorithm, Multi objective partitioning, Fitness function

1 引言

系统级芯片(system on chip, SoC)以其集成度高、体积小、功耗低、可靠性高、产品面市周期短等优点得到了越来越广泛的应用,并且逐渐成为当前ASIC系统设计的主流。为了提高设计效率,软硬件协同设计方法学应运而生。软硬件划分是软硬件协同设计的关键技术之一,主要任务是在满足各项设计约束的条件下,把系统功能划分到目标结构中的软件和硬件部分上,并为系统提供最佳的软硬件折衷方案。软硬件划分根据优化目标的数量,可分为单目标划分[12-19]和多目标划分[4-6,8]。单目标仅考虑了时间或成本,随着系统规模扩大、功能增强,软硬件划分不仅仅关注系统性能,还需综合考虑时间、成本、功耗等因素,在满足系统功能需求和多项性能指标约束的情况下确定一个最优的SoC软硬件组合方案,这是一个NP-hard问题。

目前已有许多软硬件划分方法,如贪婪算法、爬山算法、整数规划[1]、混合线性规划[2]等。贪婪算法是一种启发性算法,不同的问题有不同的解决模式;爬山算法很容易陷入局部最优解,而不能搜索到全局最优解;整数规划、混合线性规划适合中小规模问题,速度慢,并且可能得到局部最优解;模拟退火法[3]、禁忌搜索法[3,4]、遗传算法[5-8]是全局算法,运行速度快,是目前最常用的方法。

多目标优化问题的最优解与单目标优化问题的最优解有着本质的不同。多目标优化问题一般不存在传统意义上的"最优解",而是存在一组互不支配的Pareto最优解,即相对最优解。多目标划分求解的关键是求出问题的Pareto解集,然后由设计者选择合适的解。

多目标优化问题一般可以描述为下面的数学模型:

$$\begin{cases} \min f(x)=[f_1(x),f_2(x),\cdots,f_p(x)]^T \\ s.t.\ g_i(x)\leqslant 0, i=1,2,\cdots,m \end{cases}$$

到稿日期:2010-01-08 返修日期:2010-03-18 本文受863国家专项基金项目(2008AA01A202)资助。

罗 莉(1971—),女,博士,副研究员,主要研究方向为体系结构等,E-mail:luoli_hi@sina.com;夏 军(1976—),男,博士,副研究员,主要研究方向为体系结构等;何鸿君(1968—),男,博士,副教授,主要研究方向为计算机应用等;刘 瀚(1970—),女,讲师,主要研究方向为计算机应用等。

式中，$x=(x_1,x_2,\cdots,x_n)^T$ 是 R^n 空间中带有 n 个决策变量的向量；$f(x)$是目标函数，包含 p 个子目标函数；$g_i(x)$是 m 个不等式的约束函数，构成多目标优化的约束集。在多目标优化问题中，各个子目标往往是互相冲突的，一个子目标的改善有可能引起另一个子目标性能的下降。也就是说，通常情况下要同时使多个子目标都一起达到最优值是不可能的，而只能是在它们中间进行协调，做折衷处理，使各个子目标函数都尽可能地达到最优。

在问题的解空间中，一个可行候选解如果至少在某些目标函数上是最优的，则被称为相对最优解或 Pareto 最优解。但是，在实际的多目标最优化问题中，相对最优解往往是不能令人满意的，主要原因是：相对最优解通常是一个集合，由于这些解只具有部分目标的最优性，使得决策者难以从中选择一个解作为该多目标优化问题的最终结果。因此，为了要从相对最优解集中去选择“最好的”解，必须根据决策人的偏好，即决策人对各种目标的偏好程度，指定决策规则，并用该决策规则来排列方案的优劣次序，从而找出最满意的决策方案。

近年来，随着遗传算法的发展，越来越多的研究者采用遗传算法解决多目标优化问题，并已被证明是一种行之有效的方法[1]。遗传算法的本质特征是通过在代与代之间维持由潜在解组成的种群来实现多向性和全局搜索。这种从种群到种群的优化方法在搜索 Pareto 解时是非常有用的，在一次优化中可以得到多个 Pareto 解。

本文应用遗传算法解决文献[9]的多目标划分问题。文献[9]采用基于图论的遍历算法，它虽然能够保证找到一个最优解，但得到多个 Pareto 解时出现了虚假解。由于搜索了许多与最短路径无关的节点，因此其搜索速度较慢，搜索效率非常低，时间花费较多。遗传算法不用遍历整个搜索空间，而是根据所选择的适应度函数朝着最有希望的方向前进。它的搜索速度虽然较快，理论上也能找到最优解，但在实际应用过程中往往不收敛。本文通过抽象描述将实际 SoC 设计问题转化为多目标软硬件划分的遗传算法，深入研究自适应函数的定义，选择自适应参数调整的适应度函数，在增强有效解产生的同时也能避免遗传算法过早收敛；同时，算法定义了惩罚函数，既保证了 Pareto 个体有最大的机会遗传到下一代，又维持了种群的多样性，充分发挥了遗传算法群体进化和隐含并行性的优势。实例证明，此算法不仅克服了过早收敛，加快了搜索速度，还有效获得了 Pareto 最优解集。

2 实际问题描述

我们的实际用例是一个 PDA 手机平台的视频、音频发送系统，由音频数据采集模块、视频数据采集模块、用户界面、MP3 编码模块、MPEG4 编码模块和同步模块组成[9]。音频数据采集模块和视频数据采集模块负责将声音和图像转换为音频和视频信息；用户界面负责接收来自用户的输入信息并将其发送给 MP3 编码模块和 MPEG4 编码模块；MP3 编码模块和 MPEG4 编码模块负责压缩、编码音频和视频信息；同步模块负责解决音频和视频的同步问题，并将这些信息通过无线网络发送给收方。

我们采用 SoC 方法进行设计。系统中的每一个功能模块都有相应的 IP 核和软件构件，可以实现其功能。这些 IP 核和软件构件的性能参数如表 1 所列。除了相应的功能要求，PDA 手机平台视频、音频发送系统还有相应的性能要求，即要求系统成本不能超过 1100 元人民币，系统的硬件面积不能超过 3900mm^2，系统功耗不能超过 1.9W，时间特性不能超过 600ms。

表 1 各功能模块对应软件构件及 IP 核性能参数表

功能单元	编号	类型	成本（元）	硬件面积（mm^2）	功耗（W）	时间特性（ms）
音频数据采集模块	Core11	软件构件	80	0	0.2	80
	Core12	IP 核	110	750	0.25	50
	Core13	IP 核	180	460	0.15	30
视频数据采集模块	Core21	软件构件	95	0	0.3	100
	Core22	IP 核	150	980	0.3	60
	Core23	IP 核	180	644	0.2	50
用户界面	Core31	软件构件	30	0	0.1	20
	Core32	软件构件	50	0	0.08	10
MP3 编码模块	Core41	软件构件	180	0	0.6	400
	Core42	软件构件	220	0	0.5	350
	Core43	IP 核	350	2600	0.5	120
	Core44	IP 核	420	1368	0.4	70
MPEG4 编码模块	Core51	软件构件	150	0	0.6	400
	Core52	软件构件	180	0	0.5	380
	Core53	IP 核	320	2475	0.5	130
	Core54	IP 核	380	1400	0.4	80
音频与视频同步模块	Core61	软件构件	120	0	0.5	150
	Core62	软件构件	150	0	0.5	120
	Core63	IP 核	220	950	0.4	50

PDA 手机平台视频、音频发送系统的软硬件划分，其目的是在满足各项约束的条件下，寻找所有可能满足约束条件的构件组合。

6 个模块分别可用不同的构件类型实现：音频数据采集模块、视频数据采集模块和音频与视频同步模块可分别由 3 种方式实现，MP3 编码模块和 MPEG4 编码模块可分别由 4 种方式实现，用户界面可由两种方式实现。不同的实现方式对应的构件成本、功耗、硬件面积和执行时间也不同。我们在系统实现时，希望将这些性能参数最小化。但是这些性能参数往往是互相矛盾与制约的，比如成本的降低常常会导致执行时间的增加。因此，我们只能在各性能之间进行折衷，以期望达到较好的系统性能，或者我们只注重某一方面的性能，而其它方面只要满足约束即可。遗传算法划分以优化整体性能为目标，在满足系统约束的条件下，寻求各性能之间的平衡点。

2.1 问题的形式化描述

每个可选的 IP 核和软件构件都对应了不同的成本、硬件面积、功耗和时间特性，这样我们就可以将各 Core 定义为一个四元组，即 Core(cost, area, power, time)，其中 cost 代表 IP 核或软件构建的成本，area 代表 IP 核或软件构建的硬件面积，power 代表 IP 核或软件构建的功耗，time 代表 IP 核或软件构建的时间特性。

约束问题同样可描述为

$$SUMCOST(Core_{1i},Core_{2j},Core_{3k},Core_{4l},Core_{5m},Core_{6n})\leqslant 1100$$

$$SUMAREA(Core_{1i},Core_{2j},Core_{3k},Core_{4l},Core_{5m},Core_{6n})\leqslant 3900$$

$$SUMPOWER(Core_{1i},Core_{2j},Core_{3k},Core_{4l},Core_{5m},Core_{6n})\leqslant 1.9$$

$$SUMTIME(Core_{1i},Core_{2j},Core_{3k},Core_{4l},Core_{5m},Core_{6n})\leqslant 600$$

$$1\leqslant i\leqslant 3,1\leqslant j\leqslant 3,1\leqslant k\leqslant 2$$

$$1\leqslant l\leqslant 4,1\leqslant m\leqslant 4,1\leqslant n\leqslant 3$$

式中，函数 *SUMCOST*，*SUMAREA*，*SUMPOWER* 和 *SUM-*

TIME 分别表示系统的成本、硬件面积、功耗和时间之和。

2.2 采用的遗传算法

遗传算法的主要过程可以表述为：将问题的可能解编码表示为染色体，随机产生一个染色体群体，然后将群体中的染色体个体放在一定的环境中，按照适者生存的原则，从中选出适应环境较好的个体，进行复制(reproduction)、交叉(crossover)、变异(mutation)等操作，产生下一代更加适应环境的个体。如此一代一代进化，并保留适应函数大的子代。当适应度函数达到阈值时，进化停止，此时得到的就是问题的最优解。

在遗传算法的实现中，遗传编码采用符号编码方式，每一位代表一个功能模块，其值为对应的 IP 核或软件构件。采用符号编码对应于问题本身，简单易行、容易理解，在解决优化问题时往往比二进制编码方案更快、更稳定。比如编码(1,3,2,4,4,3)，可以确定 core1 是采用实现方式 1，core2 采用它的实现方式 3，core3 采用它的实现方式 2，core4 采用它的实现方式 4，core5 采用它的实现方式 4，core6 采用它的实现方式 3。

交叉操作采用基于概率的双点交叉策略，发生交叉的数目为种群规模 POPSIZE 的 1/2；变异操作采用基于随机概率的单点变异，发生变异的个体数目为种群规模 POPSIZE 的 0.2 倍；选择操作采用父子选择策略，子代全部进入下一代，不足的部分用父代个体中适应度大的加以补充。

2.3 适应度函数的定义

遗传算法模拟自然界的“优胜劣汰”。我们用适应度的大小来模拟个体对环境的适应能力，以此表示解点的好坏。适应度函数的友好定义表示解点的好坏，适应度之间的差距既不能太大，又要适当拉开距离，以便强化竞争，且能避免过早收敛。

根据 Pareto 最优解的定义，我们的划分问题存在多种划分方案。对设计者而言，一个好的划分算法应该能够求出 Pareto 解集，然后根据实际情况和个人偏好，选出最合适的划分方案。种群个体的适应度函数设置如式(1)所示：

$$Fit1=\frac{1}{\alpha\times SUMCOST+\beta\times SUMAREA+\gamma\times SUMPOWER+\delta\times SUMTIME} \tag{1}$$

式中，α 取值为 3，β 取值为 1，γ 取值为 2000。由于功耗的值在这里很小，为了突出功耗的作用，同时也是基于功耗对系统的发展约束的考虑，我们提高了功耗的影响因子，使 δ 取值为 6.5。这些参数的选取，均是在不断试验的基础上得到的常数，计算简单，但存在明显不足，很容易获得无效解。例如，如果 cost 变得比较大，远超过它的约束值 1100，而剩下的 3 个参数其中有一个很小，则同样可使适应度值达到要求。

固化适应度参数虽然能简化求解过程，但是 Pareto 解集应尽可能覆盖整个设计空间，太多相似的解或无效解是没有意义的。所以多目标划分的关键问题是求出 Pareto 可行解。为加快求解速度，我们对参数的变化进行自适应调整，增加有效解，并基于惩罚函数选择保留精英。定义适应度函数为：

$$Fit2=f1+f2 \tag{2}$$

$$f1=\frac{1}{\alpha1\times SUMCOST+\beta1\times SUMAREA+\gamma1\times SUMPOWER+\delta1\times SUMTIME}$$

$$f2=\frac{1}{\alpha2\times SUMCOST+\beta2\times SUMAREA+\gamma2\times SUMPOWER+\delta2\times SUMTIME}$$

控制参数 $\alpha1,\beta1,\gamma1,\delta1,\alpha2,\beta2,\gamma2,\delta2$ 的变化在下两节中阐述。

2.4 自适应的适应度函数

适应度函数的参数定义为常数，很易产生违反约束的无效解。为有效提高算法的收敛速度，我们对参数进行自适应调整，适应值之间的差距既不能太大，又要适当拉开距离，以便强化竞争且避免过早收敛。

对于个体性能，如果对应的适应度值明显大于或小于约束值，我们可通过修正控制参数 $\alpha1,\beta1,\gamma1,\delta1$ 来改善求解速度。

$$\alpha1=\begin{cases}\dfrac{SUMCOST}{MAXCOST}*\alpha1, & SUMCOST\geqslant 1.1\times MAXCSOT\\ \dfrac{MAXCOST}{SUMCOST}*\alpha1, & SUMCOST\leqslant 0.75\times MAXCSOT\end{cases} \tag{3}$$

$$\beta1=\begin{cases}\dfrac{SUMAREA}{MAXAREA}*\beta1, & SUMAREA\geqslant 1.1\times MAXAREA\\ \dfrac{MAXAREA}{SUMAREA}*\beta1, & SUMAREA\leqslant 0.9\times MAXAREA\end{cases} \tag{4}$$

$$\gamma1=\begin{cases}\dfrac{SUMPOWER}{MAXPOWER}*\gamma1, & SUMPOWER\geqslant 1.1\times MAXPOWER\\ \dfrac{MAXPOWER}{SUMPOWER}*\gamma1, & SUMPOWER\leqslant 0.85\times MAXPOWER\end{cases} \tag{5}$$

$$\delta1=\begin{cases}\dfrac{SUMTIME}{MAXTIME}*\delta1, & SUMTIME\geqslant 1.1\times MAXTIME\\ \dfrac{MAXTIME}{SUMTIME}*\delta1, & SUMTIME\leqslant 0.9\times MAXTIME\end{cases} \tag{6}$$

对 4 个控制参数赋予初值，$\alpha1$ 取值为 3，$\beta1$ 取值为 1，$\gamma1$ 取值为 2000，$\delta1$ 取值为 6.5，在后续的迭代过程中根据个体的情况不断地进行修正。如果 *SUMCOST* 大于等于 1.1 倍的 *MAXCOST*，我们就通过提高 $\alpha1$ 的值来使该个体的适应度变小，$\alpha1$ 变为 $\frac{SUMCOST}{MAXCOST}*\alpha1$。同样，当 $\alpha1$ 小于等于 0.75 倍 *MAXCOST* 时，$\alpha1$ 调整为 $\frac{MAXCOST}{SUMCOST}*\alpha1$。

同理，我们得到 *SUMAREA* 大于等于 1.1 倍 *MAXAREA* 和小于等于 0.9 倍 *MAXAREA* 的 $\beta1$ 值分别为 $\frac{SUMAREA}{MAXAREA}*\beta1$ 和 $\frac{MAXAREA}{SUMAREA}*\beta1$；*SUMPOWER* 大于等于 1.1 倍 *MAXPOWER* 和小于等于 0.85 倍 *MAXPOWER* 的 $\gamma1$ 值分别为 $\frac{SUMPOWER}{MAXPOWER}*\gamma1$ 和 $\frac{MAXPOWER}{SUMPOWER}*\gamma1$；*SUMTIME* 大于等于 1.1 倍的 *MAXTIME* 和小于等于 0.9 倍 *MAXTIME* 时的 $\delta1$ 值为 $\frac{SUMTIME}{MAXTIME}*\delta1$ 和 $\frac{MAXTIME}{SUMTIME}*\delta1$。

2.5 基于惩罚函数的适应度赋值

采用精英保留选择策略，对在解空间中无对应可行解的个体，计算其适应度时，加入一个惩罚项，从而降低该个体的适应度，使该个体被遗传到下一代种群的机会减少。

精英保留选择策略从父代种群中选择个体适应度函数值。如果个体各项性能都满足约束条件，则控制参数 $\alpha2,\beta2,\gamma2$ 和 $\delta2$ 不变，否则将控制参数施加一个惩罚项，控制参数 $\alpha2,\beta2,\gamma2$ 和 $\delta2$ 的变化分别如式(7)—式(10)所示。控制参数 $\alpha2,\beta2,\gamma2$ 和 $\delta2$ 的初始值，即 $\alpha2$ 取值为 3，$\beta2$ 取值为 1，$\gamma2$ 取值为 2000，$\delta2$ 取值为 6.5。

遗传算法经过交叉变异得到的个体只是新种群的一部

分。为了达到种群的规模，我们采用精英保留机制，将原始种群中适应度值比较大的个体保留到下一代种群中。如果这些个体的某项性能不满足系统约束，那么它们的适应度将要受到调整，主要是通过惩罚项进行调整。在初始进化过程中，为了保持种群的多样性，惩罚项所产生的影响不是很明显。但是随着进化代数的增加，惩罚项对那些不满足约束条件的个体的惩罚将越来越明显，从而个体的适应度将明显降低，这样那些不符合约束条件的个体遗传给下一代的概率就越来越小了。

控制参数 $\alpha 2$，$\beta 2$，$\gamma 2$ 和 $\delta 2$ 的变化分别如式(7)－式(10)所示。

$$\alpha 2 = e^{\frac{SUMCOST - MAXCOST}{m}} * \alpha 2 \tag{7}$$

$$\beta 2 = e^{\frac{SUMAREA - MAXAREA}{m}} * \beta 2 \tag{8}$$

$$\gamma 2 = e^{\frac{SUMPOWER - MAXPOWER}{m}} * \gamma 2 \tag{9}$$

$$\delta 2 = e^{\frac{SUMTIME - MAXTIME}{m}} * \delta 2 \tag{10}$$

式中的指数部分为惩罚项。对于性能参数大于其约束的个体，其控制参数将变大，从而使适应度函数值变小。惩罚项中的变量 m 的变化采用模拟退火的策略，即设置为 $m_i = 0.95 * m_{i-1}$，i 为进化代数。促使违反约束的个体随进化接受的惩罚越来越大，相应的适应度值也变得越来越小，这样优秀的个体就越来越明显地显示出来。

3 实验结果

遗传算法各参数值设置如下：种群规模 POPSIZE＝1000，最大进化代数 MAXGENERATION＝300，交叉概率 $P_c = 0.8$，变异概率 $P_m = 0.1$。我们分别采用适应度函数 *Fit*1 和适应度函数 *Fit*2 实现遗传算法中的个体评估，划分结果和算法各性能对比情况分别如表 2 和表 3 所列。

表 2　遗传算法划分结果

编码	成本	面积	功耗	时间	IP 核或软件构件构成
111431	1065	3843	1.9	550	$Core_{11}$ $Core_{21}$ $Core_{31}$ $Core_{44}$ $Core_{53}$ $Core_{61}$
111432	1095	3843	1.8	520	$Core_{11}$ $Core_{21}$ $Core_{31}$ $Core_{44}$ $Core_{53}$ $Core_{62}$
112431	1085	3843	1.88	540	$Core_{11}$ $Core_{21}$ $Core_{32}$ $Core_{44}$ $Core_{53}$ $Core_{61}$

表 3　不同适应度函数对遗传算法的影响

适应度函数	平均收敛代数	最大收敛代数	最小收敛代数	发散次数
Fit1	86.1	126	46	2
Fit2	70.6	103	37	0

我们可以发现共有 3 个划分结果满足约束条件，这与文献[9]中的划分结果是一致的。采用文献[9]中的基于图论的遍历算法，最后得到的结果还要进行人工操作，删掉不满足条件的划分结果 $Core_{11}$ $Core_{21}$ $Core_{32}$ $Core_{44}$ $Core_{53}$ $Core_{62}$。而采用本节中的算法，可以直接得到满足条件的结果，无须人为删除不符合约束条件的划分。采用本节提出的算法进行划分，最终可得到 3 个结果，具体在实现时采用哪一个划分方案，要根据具体的设计要求和优化目标来选择。如果我们优化目标是降低系统的成本，那么方案 $Core_{11}$ $Core_{21}$ $Core_{31}$ $Core_{44}$ $Core_{53}$ $Core_{61}$ 是最理想的选择；如果优化的目标是功耗和时间，那么方案 $Core_{11}$ $Core_{21}$ $Core_{31}$ $Core_{44}$ $Core_{53}$ $Core_{62}$ 是最理想的选择；如果优化目标是成本、功耗和时间的综合，那么方案 $Core_{11}$ $Core_{21}$ $Core_{32}$ $Core_{44}$ $Core_{53}$ $Core_{61}$ 能实现最好的性能折衷。

表 3 列出了不同适应度函数的定义对算法的影响，是算法运行 10 次获得的结果。

由表 2 和表 3 可以看出，两种适应度均能求出满足条件的划分结果，但采用 *Fit*2 适应度函数能提高收敛速度，降低结果发散的次数，因此对结果的优化效果优于 *Fit*1 适应度函数，这说明我们的改进是有效的。

结束语　本文提出了一种有效的多目标软硬件划分遗传算法，其在软硬件划分过程中同时优化系统的成本、硬件面积、功耗和时间特性。算法定义了能进行自适应参数调整的适应度函数，在增强有效解产生的同时也能避免遗传算法过早收敛。同时，算法还定义了惩罚函数，既保证了 Pareto 个体有最大的机会遗传到下一代，又维持了种群的多样性，充分发挥了遗传算法群体进化和隐含并行性的优势，并具有良好的扩展性。另外，算法还能搜索到多目标划分问题的多个分布均匀的 Pareto 最优解集，在解决实际问题中获得了良好的效果。综上所述，本文所提出的改进算法是解决多目标划分问题的一种实用有效的方法。

参 考 文 献

[1] Niemann R, Marwedel P. Hardware /software partitioning using integer programming [C]// Proceedings of European Design & Test Conference. Geneva, Switzerland, 1996: 473-479

[2] Niemann R, Marwedel P. An algorithm for HW/SW portioning using mixed integer linear programming[J]. Design Automation for Embedded Systems, special issue: Partitioning Methods for Embedded Systems, 1997, 3(2): 65-193

[3] Else P, Peng Z, Kuchinski K, et al. System Level Hardware/Software Partitioning Based on Simulated Annealing and Tabu Search. [J]. Automation for Embedded Systems, 1997, 2(1): 5-32

[4] 马天义，刘宏伟，温东新，等. 面向多处理器 SoC 设计的低功耗软硬件划分[J]. 高技术通讯，2007，17(10)：991-996

[5] Dick R P, et al. MOGAC: A multi-objective genetic algorithm for hardware-software co-synthesis of distributed embedded systems [J]. IEEE Transactions Computer-Aided Design of Integrated Circuits and Systems, 1998, 17(10): 920-935

[6] Deb K, Pratap A, Agarawal S. A fast and elitist multi-objective genetic algorithm: NSGA—II [J]. IEEE Transactions on Evolutionary Computation, 2002, 6(2): 182-197

[7] Arato P, Juhasz S. Hardware-software partitioning in embedded system design[C]// Proceedings of the IEEE International Symposium on Intelligent Signal Processing. 2003: 197-202

[8] Deb K, Pratap A, Agarwal S, et al. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II [J]. IEEE Transactions on Evolutionary Computation, 2002, 6(2): 182-197

[9] 詹瑾瑜. SoC 软/硬件协同设计方法研究[D]. 成都：电子科技大学，2005

[10] 朱学军，陈彤，薛量，等. 多个体参与交叉的 Pareto 多目标遗传算法[J]. 电子学报，2001，29(1)：106-109

[11] 陈长征，王楠. 遗传算法中交叉和变异概率选择的自适应方法及作用机理[J]. 控制理论与应用，2002，19(1)：41-43

[12] Bhattacharya A, Konar A, Das S, et al. Hardware Software Partitioning Problem in Embedded System Design Using Particle Swarm Optimization Algorithm[C]// International Conference on Complex. Intelligent and Software Intensive Systems. 2008: 171-176

[13] Harkin J, McGinnity T M, Maguire L P. Genetic algorithm driven hardware-software partitioning for dynamically reconfigurable embedded systems [J]. Microprocessors and Microsystems, 2001(25): 263-274

[14] Zheng Shijue, Zhang Yan, He Tingting. The Application of Genetic Algorithm in Embedded System Hardware-software Partitioning[C]// International Conference on Electronic Computer Technology: 219-222
[15] 高健，李涛. 三种软硬件划分算法的比较分析[J]. 计算机工程与设计，2007，29(7)：3426-3428
[16] 李涛，杨愚鲁，马平，等. 基于遗传算法的可重构系统软硬件划分[J]. 计算机工程与应用，2007，26(3)：56-58
[17] 范乐君，李斌，庄镇泉，等. 一种基于 DQCGA 算法的软硬件动态划分方法[J]. 计算机科学，2008，35(5)：201-205
[18] Wu J G, Srikanthan T, Zou G W. New model and algorithm for hardware/software partitioning[J]. Joural of Computer Science and Technology, 2008, 23(4): 644-651
[19] Yuan Mingxuan, He Xiuqiang, Gu Zonghua. Hardware/Software Partitioning and Static Task Scheduling on Runtime Reconfigurable FPGAs Using a SMT Solver[C]// IEEE Real-Time and Embedded Technology and Applications Symposium. 2008: 295-304

(上接第 251 页)

4 实验结果与分析

根据以上给出的算法，可以将一个大小为 512×512×512×2B 的体数据分割成 64 块，每一数据块的大小为 128×128×128×2B。但是，对体数据进行的分割过多，就会引起一个上下文切换的问题。为了解决这个问题，实验中将体数据分割成了 8 个大小为 256×256×256×2B 的数据块。这样，不仅可以有效地利用显存，而且可以减少上下文切换的次数。

本文采用 VC++6.0 实现了快速的体绘制算法，在配有 Pentium(R) Dual E2140 CPU、1G 内存、独立显卡(128MB)的 PC 机上对本文提出的算法进行测试。实验中的体数据模型如表 1 所列，大小分别为 84MB，176MB，253MB 和 428MB。

表 1 测试体数据集

体数据集	分辨率	切片数目	单个体素大小/B	体数据集大小/MB
心脏	512×512	168	2	84
脚部	512×512	352	2	176
头部	512×512	506	2	253
人体上半部分	512×512	856	2	428

在对大的体数据集进行体绘制时，本文所提出的算法的性能分析结果如图 2 所示。对于 428 MB 的大数据体，以 400×400 窗口进行绘制的速度为 8～11fps。在图 2 中，利用相同的 PC 机和相同的测试体数据集，将本文所提出的算法在不使用任何图像加速硬件的情况下同传统的直接体绘制算法进行了比较，DVR，OURS 分别表示传统的绘制算法和本文所提出的算法。实验结果表明，本文的算法在性能上比传统的直接体绘制算法要好。由于本算法使用了可见性测试，大量的空白数据和对最终图像不做任何贡献的数据被剔除，因此其速度约为传统直接体绘制算法的 5～7 倍。图 3 是对人体 CT 体数据集进行快速体绘制的结果图。各体数据集的大小分别是(a) 512×512×168，(b) 512×512×352，(c) 512×512×506，(d)512×512×856，绘制窗口大小为 400×400。从图 3 中可以看到，在使用了体绘制的预积分后，绘制图像没有任何质量损失。

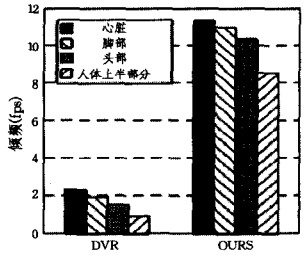

图 2 体绘制算法的性能比较

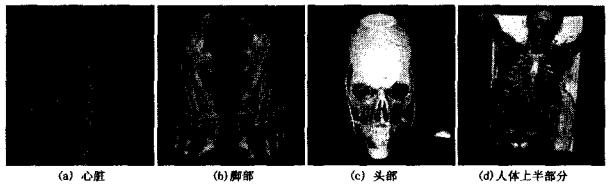

图 3 快速体绘制结果

结束语 随着现代医学影像技术的发展，所获得的医学影像体数据集的数据量也越来越大。因此，如何对大的体数据集进行快速、高质量的体绘制，是一个很大的挑战。鉴于此，本文提出了一种快速体绘制算法。通过将大的医学体数据分割成小的数据块来减少对显存的需求；对这些数据块进行一系列的可见性测试，将全部的体数据块提炼为一个近似的可见集，从而在很大程度上减少了体绘制要处理的数据量，实现了对大体数据的快速体绘制；利用体绘制预积分技术，提高了体绘制的图像质量。但是，在转换函数发生改变时，我们需要重新计算整个体数据集，可实际上，在上一转换函数中所产生的数据仍可被新的转换函数使用。因此，如何有效地利用转换函数所产生的数据，将是一个重要的研究课题。

参 考 文 献

[1] 段宝山，潘振宽. 医学断层图像三维重建的辅助轮廓线法[J]. 计算机辅助设计与图形学学报，2005，17(8)：1862-1866
[2] 鲍苏苏，林斌. 医学图像三维表面重建分叉曲面的数学形态学研究[J]. 计算机科学，2003，30(12)：136-138
[3] 何青，刘允才. 股骨模型的三维重建与变形[J]. 计算机工程，2007，33(13)：221-223
[4] Levoy M. Display of surfaces from volume data[J]. IEEE Computer Graphics and Applications, 1988, 8(3): 29-37
[5] Kye Heewon, Jeong Dong kyun. Accelerated MIP based on GPU using block clipping and occlusion query[J]. Computers and Graphics, 2008, 32(3): 283-292
[6] Choi Jae-Jeong, Shinb Byeong-Seok, Shina Byeong-Seok, et al. Efficient volumetric ray casting for isosurface rendering[J]. Computers and Graphics, 2000, 24(5): 661-670
[7] Levoy M. Efficient ray tracing of volume data [J]. ACM Transactions on Graphics, 1990, 9(3): 245-261
[8] Levoy M. Volume rendering by adaptive refinement [J]. The Visual Computer, 1990, 6(1): 2-7
[9] Lum E B, Wilson B, Ma K-L. High-quality lighting and efficient pre-integration for volume rendering[J]. Euro Graphics/IEEE Symposium on Visualization, 2004: 25-34