Website: ycyk.brit.com.cn Email: ycyk704@163.com

基于无向图的 SoC 软硬件划分方法

侯 冰
（浙江时空道宇科技有限公司 上海 200233）

摘要：软硬件划分是 SoC 软硬件协同设计中的重要步骤之一。针对软硬件划分问题，提出一种基于无向图的软硬件划分方法，将软件成本和硬件成本设定为网络图的节点，将功能模块间的通讯成本设定为无向的边，从而将芯片的软硬件划分问题归结为基于无向图理论的多目标优化求解的问题。仿真结果证明，该方法在算法效率上优于 GA 算法和 KL 算法。并且设计研制了 FPGA 测试平台，实现软硬件并行开发，提高了基带芯片开发的效率。以 BDS/GPS 双模基带芯片为例，对本方法进行了具体实施，为基带芯片的软硬件划分设计提供了理论依据。

关键词：SoC；软硬件划分；无向图；FPGA

中图分类号：TP302 **文献标识码：**A **文章编号：**CN11-1780(2021)04-0007-10

DOI：10.12347/j.ycyk.20210402001

引用格式：侯冰. 基于无向图的 SoC 软硬件划分方法[J/OL]. 遥测遥控, 2021, 42(4): 66–75[20XX-XX-XX]. http://ycyk.brit.com.cn/ycyk/article/abstract/20210402001.

Software/hardware partition method based on undirected graph theory

HOU Bing
（Zhejiang Geespace Technology Co., Ltd. Shanghai 200233, China）

Abstract: Software/hardware partition is one of the key steps in SoC software/hardware co-design. In response to the software/hardware partition problem, an undirected graph-based software/hardware partition method is proposed. The software cost and hardware cost are set as the nodes of the network graph, and the communication costs between functional modules are set as undirected edges, so that the software/hardware partition of the chip is resolved into multi-objective optimization problem based on undirected graph theory. The simulation results prove that this method is more efficient than GA algorithm and KL algorithm. FPGA test platform is designed and developed to realize parallel development of software/hardware and improve the efficiency of baseband chip development. This method is implemented in the BDS/GPS dual-mode baseband chip, which provides a theoretical basis for the software/hardware partition of the baseband chip design.

Key words: SoC; Software/hardware partition; Undirected graph theory; FPGA

DOI：10.12347/j.ycyk.20210402001

Citation: HOU Bing. Software/hardware partition method based on undirected graph theory [J/OL]. Journal of Telemetry, Tracking and Command, 2021, 42(4): 66–75[20XX-XX-XX]. http://ycyk.brit.com.cn/ycyk/article/abstract/20210402001.

引 言

随着片上系统 SoC（System on Chip）的广泛应用，其功能和规模不断增长，传统的设计方法已经无法满足设计需求。采用软硬件协同设计，可以显著缩短 SoC 系统开发的时间，并提升可靠性。软硬件划分是软硬件协同设计过程中的关键步骤之一，其主要任务是按照系统功能的需要约束，合理地将功能模块划分到可重构系统中的软件和硬件部分上，以得到性能、功耗等方面的最佳配置方案[1,2]。

软硬件划分问题可以归结为多目标优化求解的问题，软硬件划分问题的常用算法有退火算法[3,4]、爬山算法[5]、遗传算法 GA（Genetic Algorithm）[6,7]、KL 算法（Kernighan/Lin–type Algorithm）[8]及其改进算法[9,10]等。退火算法虽可找到最优解，但由于个体搜索空间大，效率不足；爬山算法的策略是

收稿日期：2021-04-02 收修改稿日期：2021-05-12

“以退为进”，但容易陷入局部最优解；遗传算法模拟生物进化过程来搜索最优解，但收敛速度较缓慢；KL 算法将一个节点图分割成两个相近大小的节点集合，使得整个系统的消耗最低，但容易陷入局部最优解。

上述算法几乎均将系统映射为有向图网络，再利用各种算法进行求解。为提高算法效率和软硬件划分质量，本文提出一种基于无向图的软硬件划分方法，将边定义为无方向，从而简化算法处理的数据，更适合网络节点量较大的系统。在仿真的同时，本文研制了测试平台对算法进行验证并实施。

1　基于无向图的软硬件划分方法

1.1　问题的提出

软硬件划分是基带芯片系统结构设计的关键。由于芯片实现的目标除功能、性能要求外，至少还包括面积、功耗等参数要求，所以芯片的软硬件划分问题通常被归结为基于图论的多目标优化求解的问题。将芯片系统映射为有向网络图，将其中的功能模块映射为网络节点，模块间的通信视为边，那么将系统目标设定为节点或边的权值，则软硬件划分问题演化为有向图的极值问题。有向图利用控制流图等将数据流的流向以及系统模型结构映射图中，更接近于实际情况。然而，在有向图网络理论中，边作为有向向量必须当作两个参数值来对待，算法需处理的数据量庞大，导致算法效率降低，难以适应节点数目庞大的系统网络的解算。

基于有向图网络节点理论的软硬件划分方法均集中在节点数量为几或几十的量级，而无向图在理解划分问题的理论基础以及软硬件划分过程的快速算法有一定优势。目前的研究表明，有向图理论更适宜解决粒度较大或是规模较小的系统，而无向图的应用更为广泛，应用于无人机集群领航顶点选取[11]、装配序列规划[12]、生物数学[13]、多窗顶点频率信号分析[14]等大规模、高复杂度的优化问题。

文献[15,16]提出了基于无向图网络的分析方法，将边的方向忽略后，算法处理的数据将被化简，更适应了网络节点量较大的情况，但由于应用范围不同，这两种方法均针对计算嵌入式系统的硬件成本，而集成电路系统的功耗计算需要计算芯片工作时的整体成本。另外，分析 BDS/GPS 双模基带芯片可知，随着可接收系统的增多，以及对定位速度及精度的要求不断提高，对通道数量的要求大幅增加。芯片规模的不断扩大，导致系统结构的网络节点爆炸式增长。因此，有必要进一步研究针对系统整体成本的软硬件划分方法。

1.2　基于无向图的软硬件划分方法

定义无向图 $G(V,E)$，顶点集 $V=\{v_1,v_2,\cdots v_n\}$。s_i、h_i 分别表示顶点 v_i 的软件成本和硬件成本，s_i、$h_i\in R+$；c_{ij} 表示当 v_i、v_j 用不同方式实现的通讯成本，$c_{ij}\in R+$，当 v_i、v_j 同属于软件实现或者硬件实现时，$c_{ij}=0$。P 是图 G 所对应软硬件划分问题的一种划分方法，则 P 为图 G 的二分图，即 $P=(V_s,V_h)$，$V_s\cup V_h=V$，$V_s\cap V_h=0$。穿越 P 的边定义为 $E_p=\{(v_i,v_j), v_i\in V_s, v_j\in V_h$ 或 $v_i\in V_h, v_j\in V_s\}$。因此，系统的硬件成本为 $H_p=\sum_{(vi\in Vh)}h_i$，软件成本 $S_p=\sum_{(vi\in Vs)}si$，通信成本 $C_p=\sum_{((vi,vj)\in Ep)}c_{ij}$。最佳的划分方法 P^*，应使得 $T_p^*=\alpha H_p^*+\beta S_p^*+\gamma C_p^*$（$\alpha,\beta,\gamma$ 为非负常数）最小。

接下来，本文将证明寻找最佳划分方法 P^* 即为寻找图 G 的最小割。

如图 1 所示，首先将图 G 转换为网络图 $G'=(V',E')$，添加源 v_s 及汇 v_h，即 $V'=V\cup\{v_h,v_s\}$，$E'=E\cup E_s\cup E_h$，其中，$E_s=\{(v,v_s),v\in E\}$，$E_h=\{(v,v_h),v\in E\}$。图 G' 中的端点不存在加权值（成本 s_i，h_i），而将加权值（成本）转换到边(v,v_s)及(v,v_h)，$v\in E$，则 E' 中的边 e 的加权值可以表示为

$$w(e)=\begin{cases}\gamma c(e) & e\in E\\ \alpha h_i & e=(v_i,v_h)\in E_h\\ \beta s_i & e=(v_i,v_s)\in E_s\end{cases}\qquad(1)$$

寻找图 G' 的最小割，则只需寻找割集 $E'*$，使得 $W(e)*=\sum w(e)$，$v\in E'*$ 最小。

因为 $W(e)=\sum_{e\in E'^*} w(e)=\sum_{e\in E^*}\gamma c_{ij}+\sum_{e\in E_s^*}\alpha si+\sum_{e\in E_h^*}\beta hi$

$=\gamma\sum_{e\in E^*} c_{ij}+\alpha\sum_{e\in E_s^*} si+\beta\sum_{e\in E_h^*} hi$

$=\gamma C_p+\alpha S_p+\beta H_p=T_p$ ，

所以 $W(e)^*=\min\{W(e)\}=T_p^*$。

故，G' 的最小割为所求最佳划分方法 P^*。

而根据 stone 定理[17]，G' 的最小割即为 G 的最小割，得证。

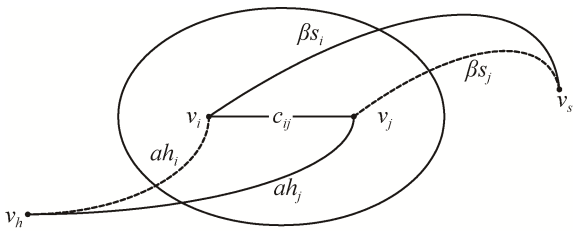

图 1　G 的扩展图 G'

Fig. 1　G' – expansion graph of G

根据上文定义描述，将最佳划分问题总结为两种问题的表述形式[15,16]。

问题 1：给定无向图 G，以及 α、β、γ>0，寻找最佳划分 P 使得 $T_p=T_p^*=\alpha H_p^*+\beta S_p^*+\gamma C_p^*$ 最小，即最佳划分问题为寻找图 G 的最小割的问题。

问题 2：给定无向图 G，以及 R>0，寻找最佳划分 P，使得在满足 S_p+C_p<R 的情况下，H_p 最小，即最佳划分问题为在受限条件下，寻找 H 的最小值问题。

在文献[15]中，针对问题 1 的不同参数 α、β、γ 进行求解，得到一组候选解，然后再根据问题 2 中给定的 R 的值，在候选解中选择问题 2 的最优解，从而最终将问题 2 转化为仅与参数 α、β 有关的求最小值的问题。

然而，问题 1 和问题 2 的表述方式均不能完全符合实际工程需要。问题 1 为在无约束的情况下求最优解，然而实际情况中，无约束条件下的求解的情况较少；问题 2 的表述虽然更符合实际情况，但问题 2 的表述仅求解了硬件成本的最小值，在芯片设计中，不仅仅要考虑单芯片的硬件成本，也要考虑到系统整体成本，包括通信成本。在约束受限条件下，求解整个系统的成本最小值，更具实际意义。

因此，得到问题 3：给定无向图 G，以及 α、β、γ>0 和 R>0，寻找最佳划分 P，使得在满足 S_p+C_p<R 的情况下，$T_p=T_p^*=\alpha H_p^*+\beta S_p^*+\gamma C_p^*$ 最小。

可见，问题 3 为问题 1 及问题 2 的综合，根据问题 3，本文提出与文献[15]相反的算法基本思路，将不同 R 参数条件下，问题 2 中的 H 最小值作为候选解，并将这些候选解带入问题 1 中，使得 $T_p^*=\alpha H_p^*+\beta S_p^*+\gamma C_p^*$ 最小的候选解作为问题 1 的解。可见最终的最佳划分问题，是仅与 R 有关的最小值问题。

定义集合 V={v_1,v_2,v_3,v_4⋯v_n}，对于无向图 G 中的任务节点 v_i，当该节点为软件实现时，v_i=1，当该节点为硬件实现时，v_i=0。于是有

$$\begin{cases} S=\sum_{i=1}^{n} s_i v_i \\ H=\sum_{i=1}^{n} h_i(1-v_i) \\ C=\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} c_{ij}\,|v_i-v_j| \end{cases} \qquad (2)$$

首先，讨论问题 2，可将问题 2 描述为

$$P=\min(\sum_{i=1}^{n} h_i(1-v_i)),\ \text{当且仅当}\sum_{i=1}^{n} s_i v_i+C\leqslant R, v_i\in\{0,1\}, i=1,2,\cdots,n \qquad (3)$$

很显然，求 H 的最小值，与求算式(4)的最大值问题相同。

$$Q=\max(\sum_{i=1}^{n} h_i v_i),\ \text{当且仅当}\sum_{i=1}^{n} s_i v_i+C\leqslant R, v_i\in\{0,1\}, i=1,2,\cdots,n \qquad (4)$$

又可知，0<C<R，则定义 $C=\mu R$，$0<\mu<1$，$S+C<R$ 转化为 $S<(1-\mu)R$，则式(4)可转化为

$$Q' = \max(\sum_{i=1}^{n} h_i v_i), \text{当且仅当} \sum_{i=1}^{n} s_i v_i \leqslant (1-\mu)R, \quad v_i \in \{0,1\}, i = 1,2,\cdots,n \tag{5}$$

式（5）表明，问题 2 求 H 最小值的问题 P，最终转化为与 μ 有关的最大值问题 Q'。且可知，Q' 是在 $\mu\in(0,1)$单调递减函数，则 P 为 $\mu\in(0,1)$单调递增函数。

已经讨论了问题 1，当 P 为其最小割时，得到最佳划分，也就是说 $v=v^*_{\text{mincut}}$ 时，得到 $T^*_{p\text{mincut}} = \alpha H^*_{p\text{mincut}} + \beta S^*_{p\text{mincut}} + \gamma C^*_{p\text{mincut}}$，此时，$\beta S^*_{p\text{mincut}} + \gamma C^*_{p\text{mincut}} < R_0$，所以得到

$$S^*_{\text{mincut}} < (1 - \frac{\gamma}{\beta R_0} C^*_{\text{mincut}})\, R_0 \tag{6}$$

若 H^*_{mincut} 为问题 2 的一个解，令 $R=R_0$，$0<\mu<1$，则得到

$$R > \frac{\gamma}{\beta} C^*_{\text{mincut}} = K \tag{7}$$

可见，当 $R>K$ 时，问题 3 与问题 1 的最优解相同，为图 G 的最小割。当 $R<K$ 时，问题 3 的解并非问题 1 的最优解，其最优解下限为

$$Tp > \alpha(\sum h_i - U_{\text{mincut}}) + \max(\beta,\gamma)R \tag{8}$$

其中，U_{mincut} 为函数 Q' 在 $\mu=C_{\text{mincut}}/R$ 情况下的上限。

利用伪代码来描述该算法的基本步骤如下

Step1　　确定 R,dr,令 $T_{p\text{-min}}=0$
Step2　　for(r=0, r<R, r=r+dr)
　　{
Step2.1　$H_{\text{min-now}}$=问题 2 中该 r 条件下的 H_{min}
Step2.2　记录 $H_{\text{min-now}}$ 所对应的 S_{now} 和 C_{now}
Step2.3　计算 $T_{p\text{-now}}=\alpha H_{\text{min-now}}+\beta S_{\text{now}}+\gamma C_{\text{now}}$
Step2.4　if($T_{p\text{-now}}<T_{p\text{-min}}$)
　　{
　　$T_{p\text{-min}}=T_{p\text{-now}}$
　　记录 *r-prev* 和 *r-next*
　　}
　　}
Step3　　令
　　d$r'=\varepsilon''$ dr, $\varepsilon''<1$, $r'\in$(r-prev,r-next)
Step4　　for(r'= r-prev, r' <r-next, r'=r'+dr')
　　{
Step4.1　$H_{\text{min-now}}$=问题 2 中该 r 条件下的 H_{min}
Step4.2　记录 $H_{\text{min-now}}$ 所对应的 S_{now} 和 C_{now}
Step4.3　计算 $T_{p\text{-now}}=\alpha H_{\text{min-now}}+\beta S_{\text{now}}+\gamma C_{\text{now}}$
Step4.4　if($T_{p\text{-now}}<T_{p\text{-min}}$)
　　{
　　$T_{p\text{-min}}=T_{p\text{-now}}$
　　}
　　}
Step5　　此时的 $T_{p\text{-min}}$ 为最终的最佳划分 T^*_p。
算法结束。

其中，Step2.1 为算法关键，下面具体描述 Step2.1 中求 H 最小值的算法。

采用与基本算法相同的思路，先寻找大范围，再在最佳值附近搜索，具体算法如下

Step2.1.1 指定 dμ，设 best-now=0
Step2.1.2 for(μ=0, μ<1, μ=μ+dμ}
　　{
Step2.1.2.1　搜索 V，找到该 μ 条件下的 best-now
Step2.1.2.2　记录 best-now 时刻的最大值以及 V 的值
Step2.1.2.3　记录此刻的 μ-prev 和 μ-next
　　}
Step2.1.3 令 d$\mu'=\varepsilon'$dμ, $\varepsilon'<1$, $\mu'\in$(μ-prev, μ-next)
Step2.1.4 for(μ'=μ-prev, $\mu'<\mu'$-next, μ'=μ+dμ)
　　{
Step2.1.4.1　搜索 V，找到该 μ 条件下的 best-now
Step2.1.4.2　记录 best-now 的最大值以及 V 的值
Step2.1.4.3　记录此刻的 μ-prev 和 μ-next
　　}
Step2.1.5 输出 best_now，以及此刻的 H、S 和 C
算法结束。

首先，讨论 Step2.1 的算法复杂度。Step2.1.2 循环语句的复杂度为 $O(1/\text{d}\mu)$，根据文献[16]说明，内

部嵌套的语句 2.1.2.1，需经过排序算法和启发式算法，其复杂度为 $O(n\log n)+O(n+m)=O(n\log n)$。Step2.1.2 的其它步骤的算法复杂度均为 $O(1)$，可忽略不计。语句 Step2.1.2 的算法复杂度为 $O(n\log n/\mathrm{d}\mu)$。同理，Step2.1.4 的算法复杂度函数为 $O(n\log n/\varepsilon')$。Step2.1 的算法复杂度为两者相加 $O(n\log n/\mathrm{d}\mu)+O(n\log n/\varepsilon')=O((1/\mathrm{d}\mu+1/\varepsilon')n\log n)$。

其次，讨论其余步骤的算法复杂度。由于 Step2 中其它语句的算法复杂度均为 $O(1)$，所以 Step2 的算法复杂度为 $O((1/\mathrm{d}\mu+1/\varepsilon')n\log n/\mathrm{d}r)$。Step4.1 的算法复杂度计算方法与 Step2.1 相同，所以 Step4 的算法复杂度为 $O((1/\mathrm{d}\mu+1/\varepsilon'')n\log n/\mathrm{d}r')$。Step1、3、5 的算法复杂度也为 $O(1)$。因此，最终该算法的算法复杂度为

$$O((1/\mathrm{d}\mu+1/\varepsilon')n\log n/\mathrm{d}r)+O(1/\mathrm{d}\mu+1/\varepsilon'')n\log n/\mathrm{d}r')= O(\max((1/\mathrm{d}\mu+1/\varepsilon'),(1/\mathrm{d}\mu+1/\varepsilon'')/\varepsilon'')n\log n/\mathrm{d}r)=O(An\log n) \quad (9)$$

1.3　仿真结果

采用 Matlab 实现本算法，并与遗传算法及 KL 算法进行了比较。由于目前尚没有针对软硬件划分方法的标准验证程序，本文选取与文献[15]中相同的实验方法。该方法的具体数据见表 1。其中，图的大小 *size* 与节点数 *n* 和边数 *m* 的关系为 $size=2n+3m$，节点软件成本 s_i 为 1 至 100 之间的随机数，硬件成本为以 *ksi* 为期望值、以 λks_i 为方差的正态分布随机数。而每条边的通信成本为区间[0，$2\xi s\max$]内的均匀分布的随机数，ξ 定义为通信计算率 *CCR*（Communication to Computation Ratio）。

表 1　实验用例说明

Table 1　Experimental use case description

节点数(*n*)	边数(*m*)	图的大小（*size*）
25	34	152
21	50	192
26	71	265
150	333	1 299
329	448	2 002
1 000	1 000	5 000
1 000	2 000	8 000
1 000	3 000	11 000
1 500	1 500	7 500
1 500	3 000	12 000
1 500	4 500	16 500
2 000	2 000	10 000
2 000	4 000	16 000
2 000	6 000	22 000

k 与 λ 对实验结果的影响较小，因此本文不做讨论。本文针对 *size* 和 *CCR* 参数以及 *R* 进行比较讨论。

首先，分析 *size* 及 *m*、*n* 对实验结果的影响。由图 2、图 3、图 4 均可得出如下结论：随着 *n* 的增大，其最佳划分所需的成本值也就越高，而与边的数量 *m* 的关系不大。分析其主要原因为：在无向图的建模过程中，如果 *n* 一定，即使 *n* 之间由多条边相连，最终选取最佳划分时，只有两点在不同的软硬件区域内时，且该两点之间的只有一条边才会被计入最终的成本，因此，最终的成本值与 *m* 值无关。

图 2 表明了 *R* 对最佳划分成本值的影响。*R* 的取值范围为[0，$\sum si$]，当 *R*=0，表示系统完全由硬件完成。而当 $R=\sum si$，则问题 2 与问题 1 相同，该算法无意义。令 $R=\sigma\sum si$，在实验中分别选取参数 σ=0.1，σ=0.6，σ=0.85，实验结果显示 σ=0.1 时，所求得成本值较大。分析原因，*R* 较小时，*R*<*K*，不能达到最小割所代表的最佳划分，而就实际情况而言，则表示对软件及通信成本的要求较高，而对硬件成本要求较低，因此，在搜索过程中，由于条件限制，容易出现局部最优解的情况。σ=0.6 与 σ=0.85 所求的成本值基本相同。显然 σ 较小，搜索范围较小，则速度更快。因此，在实际应用中，若无具体 *R* 值要求，选取 σ=0.6 作为初解的约束条件较为适宜。

图 3 和图 4 为本文采用的方法（Alg）在 *CCR*=10 和 1 的情况下与 KL 算法、GA 算法在求最佳划分成本时的比较。首先，分析 *CCR* 对于最终成本值的影响，可以看出两种情况下的成本值几乎一致，与文献[15]中当 *CCR* 较大时，所求的硬件成本较大的结果不同。这是由于文献[15]关心的是硬件成本的最小值，而本文更关心最终的系统成本的最小值，因此，在硬件成本较大的情况下，算法将转向更多选择软件成本的方向。而如前所述，边的通信成本对于总体成本影响较小，从而导致在两种情况下结果的趋同。而通过图 3 和图 4 亦可见，三种方法在求解效果上几乎一致。

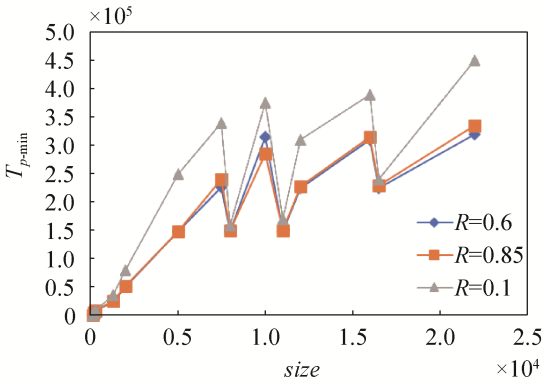

图 2　不同 R 值的运算结果的比较

Fig. 2　Comparison of calculation results with different R values

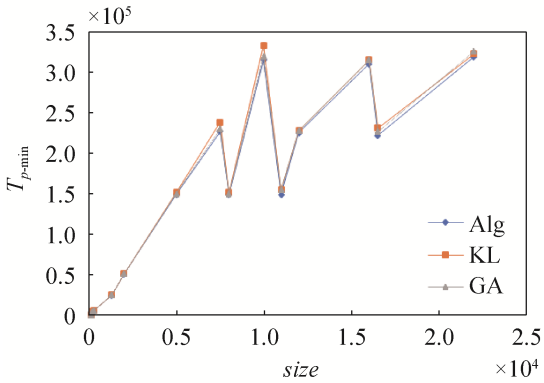

图 3　三种方法的计算结果（CCR=10）

Fig. 3　Calculation results of three methods（CCR=10）

图 5 为本文采用的方法（Alg）与 KL 算法、GA 算法在求解最佳划分时，成本为最小值时所需时间的比较。一方面，对于本文提出的算法，从总体上来看，*size* 越大，其所需时间越长，而边的数量 m 对运算时间的影响也较顶点数量 n 要小，与算法复杂度分析的结果基本一致。另一方面，从三种方法的运算时间来看，GA 算法的运算时间最长，与图的 *size* 的关系并不明显，究其原因，GA 算法属于启发式算法，其算法复杂度的悲观预期为 $O(n^3)$，且其可能达到最小值的运算时间与初始随机产生的染色体的关系较大。而 KL 算法的算法复杂度均在 $O(tn^2)$，其与初始解的选取与迭代次数相关。因此，本文算法较 GA 算法有较大提高。而与 KL 算法相比，当 n<5 000 时，本文算法的运算速度略慢，但由于均不超过 0.1 s，因此差异可基本忽略。而当 n>5 000 时，本文算法比 KL 算法的运算速度快。原因在于：当 n 较小时，A 与 n 数量相仿，本文算法的算法复杂度 $O(An\log n)\approx O(n^2\log n)>O(tn^2)$，因此，较 KL 算法慢。而 n 较大时，$A<<n$ 时，则 $O(An\log n)$必小于 $O(tn^2)$，因此较 KL 算法的速度有所提高，其提高幅度与参数 A 有关。本文选取参数，令 A 比 n 低一个数量级，并在算法精准度和复杂度之间折中，最终得到较 KL 算法速度提高 20%~30%的仿真结果。

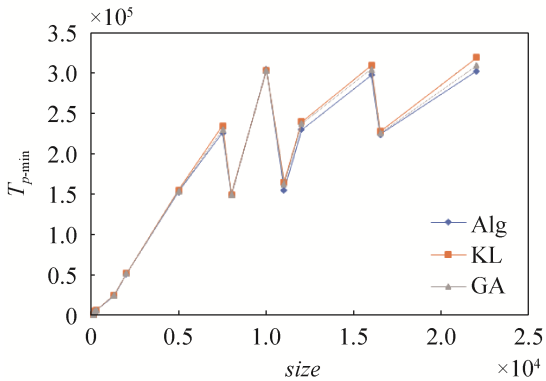

图 4　三种方法的计算结果（CCR=1）

Fig. 4　Calculation results of three methods（CCR=1）

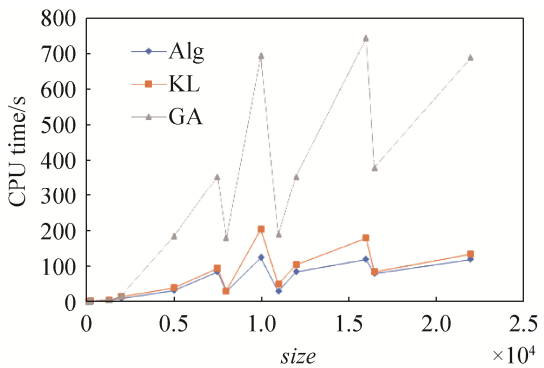

图 5　三种方法运算时间的比较

Fig. 5　Comparison of the calculation time of the three methods

2　基于 FPGA 的测试平台

随着 FPGA（Field Programmable Gate Array）性能和容量的不断提高，基于 FPGA 的 SoC 原型验证能够避免软件仿真速度慢、实时性差等缺点，加快 SoC 实现的速度，并且能够真实地反映硬件的特性，还可以检验算法的性能、验证芯片在真实环境中的工作状态。同时进行软硬件协同开发，能大大提高整个芯片的验证效率，并减小开发风险。这些优点使得基于 FPGA 的原型验证被越来越多地用于 SoC 的设计过程。

本文设计研制的 FPGA 测试平台，采用软硬件协同设计思想，使用 DSP 和 FPGA 组成可编程软硬件系统，实现软硬件并行开发，大大提高了基带芯片开发的效率。利用 FPGA 的可重构技术验证不同基带结构的性能，利用 DSP 对 FPGA 进行配置，通过实验反复验证，给芯片的集成电路设计提供技术保证。FPGA 测试平台结构框图如图 6 所示，实物图如图 7 所示。

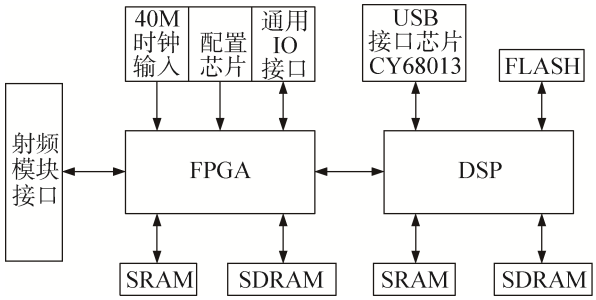

图 6　测试平台结构

Fig. 6　The structure of test platform

基带芯片的 FPGA 原型验证过程主要从硬件和软件两个方面着手。图 8 所示为基带芯片原型验证的流程图。软硬件协同设计可以在实际环境中有效地发现设计中存在的错误，进一步修改硬件和软件相关的设计，修正软件和硬件设计中的不足之处，使硬件和软件设计更加合理有效，可以大大降低流片风险，提高良品率。

软硬件的划分明确后，需要定义软硬件的接口。软硬件的通信通常通过总线来完成。软件通过访问硬件寄存器来读取或写入硬件状态，硬件亦可通过中断方式发起通信。

图 7　测试平台实物图

Fig. 7　The picture of test platform

3　实例分析

以 BDS/GPS双模基带芯片设计为例，双模基带芯片的实现形式有多种不同方案，在目前的设计实现中，数字信号处理单元在接收机芯片的软硬件划分中最具代表性，也是当前研究最为广泛的部分。因此，本文选用该单元作为软硬件划分算法的实验用例。

BDS/GPS 双模基带芯片的数字信号处理单元主要完成对下变频后的中频信号的捕获跟踪，为数据处理提供同步的卫星电文数据。利用 C 语言对该功能进行功能仿真后，则需针对芯片的系统需求进行软硬件划分。首先需要针对功能仿真代码进行初步分析，用加法和乘法来表征数据的计算量，并对各模块的计算量进行统计，得到表 2 所示的数据。然后通过利用不同的方式实现加法或乘法的成本来计算不同划分方式的芯片总成本，从而确定最优的划分方法。

在通常情况下，芯片性能指标主要包括功耗、面积以及延时，表 3 表示不同软硬件实现方式下，加法器和乘法器的性能成本。表 3 中功耗数据来自于当前设计文献中的平均水平，硬件实现面积也来自于文献中的平均值，而软件实现时，面积记为 0，在计算总面积时加入嵌入处理器面积即可。而对于延时，硬件实现的速度基本为器件延时，但在系统中，加法器或乘法器的运算结果应被寄存器锁存后可被继续使用，因此，速度记为 1 cycle(时钟周期)。而对于软件实现，由于需要寻址、读取、计算、写回的操作，平均将速度记为 3 cycle。

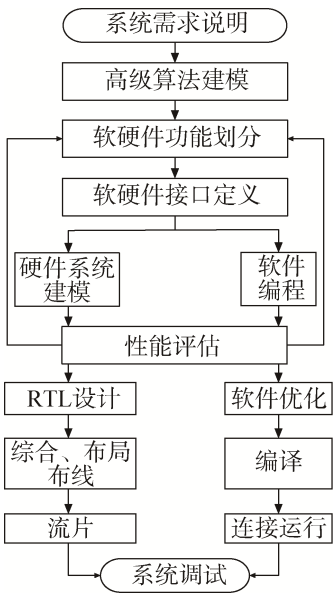

图 8　原型验证流程图

Fig. 8　Prototype verification flowchart

系统需要针对功耗、面积、延时三个性能参数均进行评估，使用遗传算法等多目标优化方式将面

临运算时间较长的问题，这在 1.3 节中已经进行了比较。为了可以使用 1.2 节中所介绍的软硬件划分方法，则需要将不同的性能参数综合计算为同一成本函数进行评估。本文采用归一化的成本公式处理不同的性能参数。

归一化的成本计算方法包含三个步骤。第一步，计算面积延时积。芯片性能指标中，面积和延时都要求尽量小。根据这一趋势，合并面积和延时两个参数，将两个参数相乘，得到面积延时积，并利用该积作为芯片面积与延时的评估函数。第二步，将功耗和面积延时积进行归一化。第三步，根据对最终性能的偏向不同，采用不同的权重参数综合两个性能参数，得到第 i 种划分的成本参数如式（10）所述，可见求最优划分方法，实为求 $C(i)$。

$$C(i)=aP'(i)+bM'(i)$$
$$0<a<1,0<b<1,a+b=1 \quad (10)$$

其中，$P'(i)$为第 i 种划分的归一化功耗值；$M'(i)$为第 i 种划分的归一化面积延时积；a、b 表示加权系数。

显而易见，当功耗越小时，总成本越低；面积延时积越小时，总成本越低；而总成本越低，则代表性能越高。因此，寻求成本最小值的划分即为该系统的最佳划分。

为了验证算法的正确性和优越性，将 BDS/GPS 双模基带芯片的捕获跟踪单元进行划分。以表 2 中的函数为网络节点，网络节点数为 16。同时，根据三种不同的关注点，如对功耗较为敏感，可选用参数 a=0.8、b=0.2；对面积较为敏感，选用参数 a=0.2，b=0.8；在一般情况下，选用 a=0.6、b=0.4。在此基础上，对算法中的参数 R 进行讨论，因为 $R=\sigma\sum si$，考虑针对不同的 σ 值进行讨论。

表 2　信号处理单元计算量统计

Table 2　Statistics of signal processing unit calculation amount

函数名称	乘法/除法次数	加法次数	特殊运算次数
carrie_nco	0	64	0
code_nco	0	64	0
cyc_cnt	0	24	0
ca_gen	0	36	0
carrier_mix	8	4	0
code_mix	24	8	0
accumulate	0	216	0
time_base	0	48	0
ch_acq	16/0	21	0
bdch_acq	16/0	21	0
ch_confirm	6/0	4	0
bdch_confirm	6/0	4	0
ch_pull_in	30/2	31	Sqrt:2;atan2:1
bdch_pull_in	40/4	53	Sqrt:3;atan2:1
ch_track	35/6	46	Sqrt:3;atan2:1;Log:1
bdch_track	76/8	76	Sqrt:5;atan2:1;Log:2

表 3　性能指标的成本说明

Table 3　Costs of performance indicators

性能指标	功耗/mW		面积/μm²		延时/cycle	
	硬件	软件	硬件	软件	硬件	软件
加法	0.64	2.21	20 303.69	0	1	3
乘法	2.23	1.67	67 529.36	0	1	5

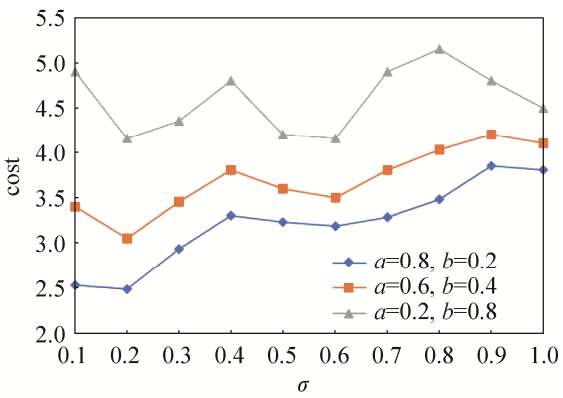

图 9　总成本比较图

Fig. 9　Comparison of total cost

图 9 为 16 个节点情况时，三种成本函数计算方法在不同 σ 值情况下的曲线，由曲线可以看出，当 σ<0.6 时，基本呈现趋势为 σ 越小，成本越小。σ=0.6 为一局部最小值，之后又重新呈缓慢上升或基本持平的形式。σ 值越小，从一方面说明软件所完成的系统功能越小；而 σ 值越小总成本越低，则说明软件成本对总成本影响较大。同时可以看出，三种不同形式的成本函数构成方式所得到的结果也不相同，对面积延时积要求越高的系统，最终得到的成本越高。通过该实验可知，所选择进行软件设计的微处理器的成本对系统成本的影响较大。

图 10 以对面积较为敏感为例，即 a=0.2、b=0.8，展示总成本、功耗成本和面积延时积成本。从图中可见，σ 为 0.6 时，此时的划分为最佳划分。本方法与 KL 算法、GA 算法的计算结果一致，在效率提升的前提下依然保持结果的准确性。

4　结束语

本文提出了一种基于无向图的软硬件划分方法，仿真结果证明，该方法在算法效率上优于 GA 算法和 KL 算法。以 BDS/GPS 双模基带芯片为例，在自行开发的 FPGA 测试平台上对算法进行了具体实施，为基带芯片的划分提供了理论指导依据。

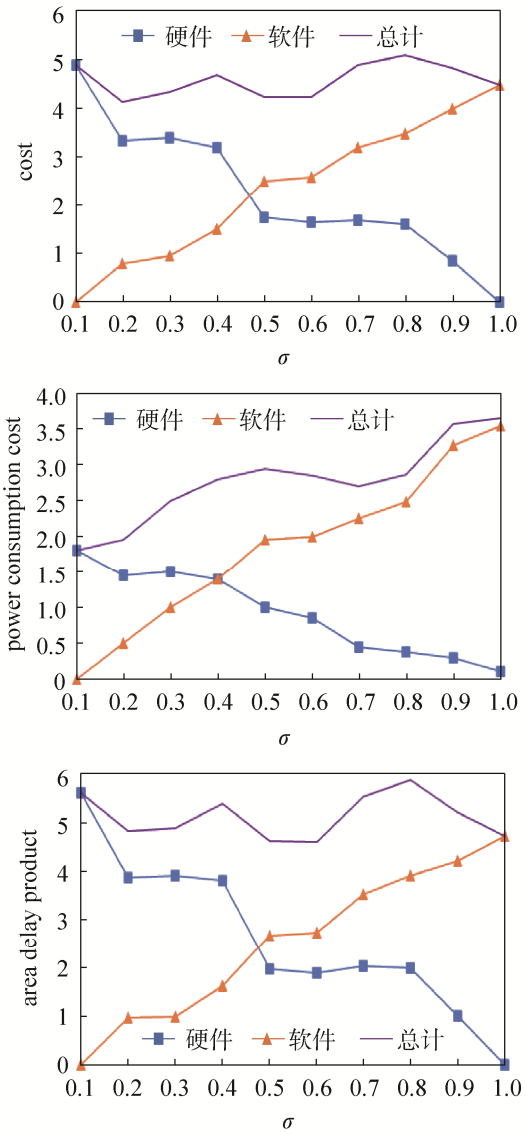

图 10　总成本、功耗成本和面积延时积（a=0.2，b=0.8）

Fig. 10　Total cost, power consumption cost and area delay product（a=0.2，b=0.8）

参考文献

[1] 黄张祥, 白瑞林, 吉峰. 图像显著性区域检测算法的软硬件协同设计及 SoC 实现[J].小型微型计算机系统, 2016, 37(9): 2114–2119.
HUANG Zhangxiang, BAI Ruilin, JI Feng. Hardware-software co-design of image salient region detection algorithm and SoC realization [J]. Journal of Chinese Computer Systems, 2016, 37(9): 2114–2119.

[2] 钟丽, 刘彦, 余思洋, 等. 嵌入式系统芯片中 SM2 算法软硬件协同设计与实现[J]. 计算机应用, 2015, 35(5): 1412–1416.
ZHONG Li, LIU Yan, YU Siyang, et al. Hardware/software co-design of SM2 encryption algorithm based on the embedded SoC[J]. Journal of Computer Applications, 2015, 35(5): 1412–1416.

[3] 王喆, 唐麒, 王玲, 等. 一种基于模拟退火的动态部分可重构系统划分–调度联合优化算法[J]. 计算机科学, 2020, 47(8): 26–31.
WANG Zhe, TANG Qi, WANG Ling, et al. Joint optimization algorithm for partition-scheduling of dynamic partial reconfigurable systems based on simulated annealing[J]. Computer Science, 2020, 47(8): 26–31.

[4] HEN S, HUANG J, XU X, et al. Integrated optimization of partitioning, scheduling, and floorplanning for partially dynamically reconfigurable systems[J]. IEEE Transactions on Computer Aided Design of Integrated Circuits&Systems, 2018: 1-1.

[5] 鄢小虎, 何发智, 陈壹林. 求解大规模软硬件划分问题的爬山淘汰粒子群算法[J]. 东南大学学报(自然科学版), 2017,47(2): 225–230.
YAN Xiaohu, HE Fazhi, CHEN Yilin. Elimination particle swarm optimization with hill climbing algorithm for solving large-scale hardware/software partitioning problem[J]. Journal of Southeast University(Natural Science Edition), 2017, 47(2): 225–230.

[6] DICK R P, JHA N K. MOGAC: a multi-objective genetic algorithm for hardware software co-synthesis of hierarchical heterogeneous distributed embedded systems[J]. IEEE Transactions on, Computer-Aided Design of Integrated Circuits and Systems, 1998, 17(10): 920-935.

[7] 侯能, 何发智. 混合并行两步调整遗传策略的软硬件划分算法[J]. 华中科技大学学报(自然科学版), 2017, 45(12): 39–45.
HOU Neng, HE Fazhi. Hybrid parallel genetic strategy with two-step adjustment for HW/SW partition[J]. Journal of Huazhong University of Science and Technology(Natural Science Edition), 2017, 45(12): 39–45.

[8] KERNIGHAN B W, LIN S. An efficient heuristic procedure for partitioning graphs[J]. The Bell System Technical Journal, 1970, 49(2): 291–307.

[9] VAHID F, LE T D. Extending the Kernighan/Lin heuristic for hardware and software functional partitioning[J]. Design

Automation for Embedded Systems, 1997, 2(2):237–261.

[10] 王东, 李娅, 吴臣, 等. 提高链式 Lin-Kernighan 算法性能的一种新策略[J]. 计算机应用, 2012(2): 425-427.
WANG Dong, LI Ya, WU Chen, et al. New strategy for improving performance of chained Lin-Kernighan algorithm[J]. Journal of Computer Applications, 2012(2): 425–427.

[11] 戴丽. 双向通信无人机集群领航顶点选取方法[J/OL]. 智能系统学报, (2020-11-09) [2021-04-02]. https://kns.cnki.net/kcms/detail/23.1538.TP.20201105.1819.002.html.
DAI Li. Leader’s selection of UAV swarm with two-way communication[J]. CAAI Transactions on Intelligent Systems, (2020-11-09) [2021-04-02]. https://kns.cnki.net/kcms/detail/23.1538.TP.20201105.1819.002.html

[12] 罗军, 盛步云, 萧筝,等. 基于无向图的铝模板装配序列规划方法[C]. 土木工程新材料、新技术及其工程应用交流会论文集（上册）, 2019, 49: 390–395.

[13] HASSAN A F, IRHAYYIM A Z. Independent (Non-Adjacent Vertices) topological spaces associated with undirected graphs, with some applications in biomathematices[J]. Journal of Physics: Conference Series, 2020, 1591(1):012096.

[14] ZHENG Xianwei, ZOU Cuiming, DONG Li, et al. Multi-windowed vertex-frequency analysis for signals on undirected graphs[J]. Computer Communications, 2021, 172,35-44.

[15] ARATO P, MANN Z A, ORBAN A. Algorithmic aspects of hardware/software partitioning[J]. Acm Transactions on Design Automation of Electronic Systems, 2005, 10(1): 136–156.

[16] WU Jigang, SRIKANTHAN T, GUANG Chen, Algorithmic aspects of hardware/software partitioning: 1d search algorithms [J]. IEEE Transactions on Computers, 2010, 59(4): 532–544.

[17] STONE H S. Multiprocessor scheduling with the aid of network flow algorithms[J]. IEEE Transactions on Software Engineering, 1977, 3(1): 85–93.

[作者简介]

侯 冰 1986 年生，博士，主要研究方向为卫星导航接收机基带结构及数据处理算法。

(本文编辑：傅 杰)