文章编号: 1001- 2486(2003) 01- 0056- 05

线型数据通道高层综合的运算单元优化分配算法*

欧 钢, 雍少为, 王飞雪
(国防科技大学电子科学与工程学院, 湖南 长沙 410073)

摘 要: 运算单元分配是高层综合的关键算法之一。采用了一种面向数字信号处理应用、规则的线型数据通道模型, 其中心思想是全局互联的最小化。以最小着色算法构造的一个初始分配为起点, 采用随机进化算法对其进行迭代改善, 以减小数据寄存器数和运算单元之间的数据交换。最后还给出了典型实例的实验结果。

关键词: 高层综合; 专用集成电路; 组合优化
中图分类号: TN911. 72 文献标识码: A

An Optimal Arithmetic Unit Allocation Algorithm in the High-level Synthesis for Linear Datapath

OU Gang, YONG Shao-wei, WANG Fei-xue
(College of Electronic Science and Engineering, National Univ. of Defense Technology, Changsha 410073, China)

Abstract: The arithmetic unit allocation is one of the key algorithm in the high-level synthesis. Mainly for digital signal process application, a linear regular datapath model is adopted whose main idea is to minimize the global interconnection. With the initial arithmetic unit allocation constructed by minimal coloring algorithm, stochastic evolution is used to improve the allocation in an iterative way, minimizing the number of the data registers and the data exchanged between the process units. Experiment results with typical design example are also presented.

Key words: high-level synthesis; application specific integrated circuit; combination optimization

在专用数字信号处理器的高层综合中, 资源分配一般在时序调度之后进行。它利用定时数据流图中各元素(数据操作、变量或数据传输) 的相容关系, 合理地分时复用硬件资源, 充分简化数据通道。资源分配算法依赖所采用的数据通道模型, 随机拓扑结构一般只适用小规模设计。本文采用的是一种分段总线、线型结构的数据通道模型, 如图 1 所示, 运算单元和其输出端的寄存器组构成一个处理胞元, 各胞元线性排列, 由分段总线连接在一起, 相邻两段之间采用双向三态缓冲器相隔, 传输一个数据将利用一个或连续的多个数据总线段, 这种总线结构支持数据广播, 而且在同一时刻一条数据总线可以传输多个不同的数据。模型中胞元的寄存器组只存储本胞元的运算结果, 而为全部胞元提供操作数据, 寄存器局部化, 以简化数据通道的互联。

针对上述数据通道模型, 资源分配主要包括三个子问题: 运算单元分配、寄存器分配以及处理胞元排列与数据总线的分配, 其中关键部分是运算单元分配, 它是把数据操作划分为数个相容子集, 每一相容子集映射至一个运算单元。本文采用随机进化算法, 减小处理器中数据寄存器的个数以及运算单元之间的数据交换, 使包含了寄存器分量与全局总线分量的代价函数最小化。

1 运算单元分配问题的定义

先给出一些符号的定义和相关的概念:

• $I=\bigcup_{k\in K} I_k$, 其中 I 为数据操作集, I_k 是由运算单元类型 k 执行的数据操作集合, K 为运算单元类

* 收稿日期: 2002- 09- 05
作者简介: 欧钢(1969—), 男, 副教授, 博士。

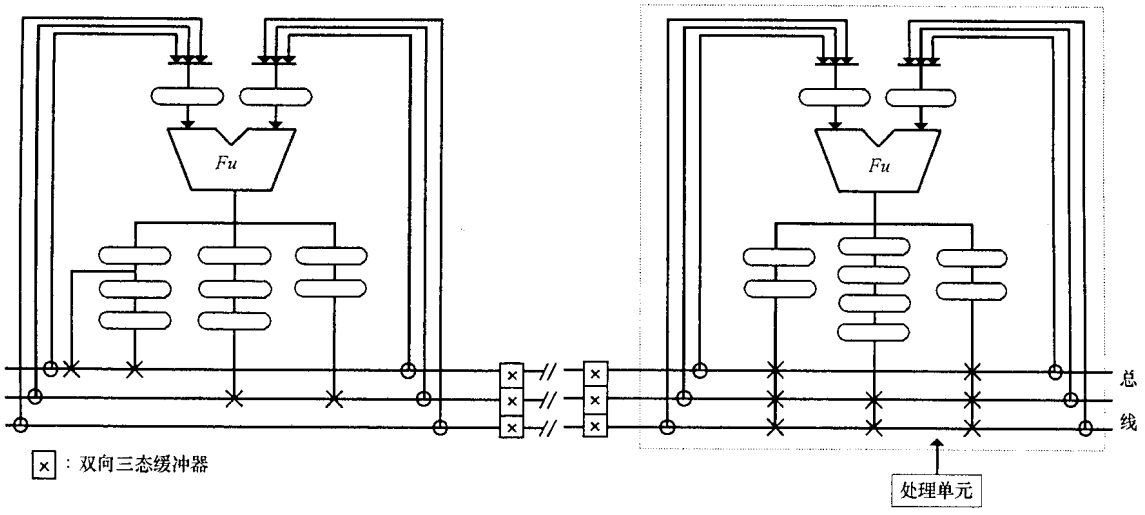

图 1　线型结构的数据通道模型

Fig. 1　The linear architecture of datapath model

型集合;

• $t \in T$ 表示时序调度的时间步, $T = [l, D_{II} + l]$, D_{II}为数据通道的初始化周期, 即数据输入的间隔;

• $INITIME_k$ 是运算单元类型 k 的数据初始化时间;

• $i \in I_k$, $OP_i = (a_i, b_i)$, 其中 a_i, b_i 为整数, 满足 $l \leqslant a_i, b_i \leqslant D_{II}$, OP_i 是数据操作 i 独占的执行时间段, 定义为 a_i, b_i 之间的开区间, 它的长度 $|OP_i| = (b_i - a_i) = INITIME_k$。在 OP_i 这一段时间内, 被数据操作 i 所占据的运算单元不能接受新的操作数。

• $\forall i, j \in I_k$, 如果称 i、j 是相容的, 当且仅当 $OP_i \cap OP_j = \phi$。

• $t \in T$, $I_{kt} = \{i \mid i \in I_k, t \in OP_i\}$, I_{kt}称之为I_k 在时刻 t 的冲突集, 集的模 $|I_{kt}|$ 为 I_k 在时刻t 的冲突指数。

运算单元分配就是把 $I_k (k \in K)$ 划分成 N_k 个子集, 使 $I_k = \bigcup_{p}^{N_k} I_k^p$, $I_k^{p1} \cap I_k^{p2} = \phi (\forall p1 \neq p2)$, 即确定映射 $F: I_k \rightarrow \{I_k^p\}_{p=1}^{N_k}$, 每一子集 I_k^p 分配至一个处理胞元(k, p)。合法的分配必须满足相容性约束, 划分后子集内所有的数据操作是相容的, $\forall p$, $\forall i, j \in I_k^p$, 有 $OP_i \cap OP_j = \phi$。

在满足相容性约束的条件下, 运算单元的优化分配使:

(1) N_k 最小, 寻求 I_k 的最小划分。

(2)使反映数据寄存器和数据通信需求的代价函数 $COST(F)$ 最小。

定义

$$COST(F) = C_B \cdot Bus + C_R \cdot R$$

其中 C_R、C_B 分别为寄存器的代价因子和数据总线的代价因子, 它们是预先已知或设定的; R、Bus 分别为数据通道中寄存器和全局总线的数量, 其中

$$Bus = \sum_{k \in K} \sum_{p=1}^{N_k} \left(\sum_{k' \in K'} \sum_{p'=1}^{N_{k'}} N_{Bus}\left(I_k^p, I_{k'}^{p'}\right) \right); \quad (1)$$

式中 $N_{Bus}(I_k^p, I_{k'}^{p'})$表示数据操作集合 I_k^p 与数据操作集合$I_{k'}^{p'}$之间的数据通信链路的个数, 它等于这两个数据操作集合同时交换的数据个数在时间轴上的最大值, 即

$$N_{Bus}(I_k^p, I_{k'}^{p'}) = \max_{t \in T} \left(DT\left(I_k^p, I_{k'}^{p'}, t\right) + DT\left(I_{k'}^{p'}, I_k^p, t\right) \right) \quad (2)$$

其中 $DT\left(I_k^p, I_{k'}^{p'}, t\right)$ 表示I_k^p 在控制步t 向$I_{k'}^{p'}$传送的数据个数。

而数据寄存器需求的计算必须在数据变量活跃周期分析的基础上进行[7], 定义符号 $Var_k^p(t)$表示

处理胞元(k, p)在控制步 t 活跃的变量集合, 那么数据通道寄存器的数量为:

$$R = \sum_{k=K}\sum_{p=1}^{N_1}(\max_{t\in T} | Var_k^p(t) |)$$

运算单元的优化分配分为两步: 第一步得到一个初始分配 F_0, 它把数据操作合法映射至最少的运算单元, 实现第一个优化目标; 第二步采用随机进化算法对初始分配 F_0 进行迭代改善, 实现第二个优化目标, 使代价函数 $COST(F)$ 最小化。

2 初始分配

如果$\exists t\in T$, 使$|I_{kt}| = 0$, 即 $t \notin OP_i$, $\forall i\in I_k$, 那么 I_k 的最小划分可以转化为区间图的最小节点着色问题。下面是转换成立的两个充分条件, 只要满足其中一个, 转化成立:

ㄋ 条件 1: 运算单元的数据初始化时间等于单位控制步, $|OP_i| = 1$, $\forall i\in I_k$;

ㄋ 条件 2: $\forall i\in I_k$, $a_i < b_i$, OP_i 不跨越处理器数据通道的初始化周期。

实际中充分条件 2 在绝大多数情况下是满足的, 许多流水调度算法不容许数据操作的执行时间跨越数据通道的数据初始化周期。这时候 I_k 的合法划分的最小子集数为

$$N_k = \max_{t\in T}|I_{kt}| \qquad (3)$$

虽然一般图的最小节点着色是 NP 难度问题, 但区间图的最小着色却存在着快速算法[6]。

假设现有 N_k 个运算单元, 记为$\{FU_p\}_{p=1}^{N_k}$, 运算单元类型 k 的初始分配可以描述如下:

第一步: 初始化

$N_k = 1$, $t = 1$, 置 FU_t 的状态为空闲;

第二步: 分配与释放

如果 b_i 等于 t, 那么释放被数据操作 i 占据的运算单元, 置其状态为空闲。

如果 a_i 等于 t, 那么把下标最小的空闲运算单元 FU_p 分配给数据操作 i, 同时置 FU_p 的状态为忙; 如果没有空闲的运算单元, 那么使 N_k 增 1, 把新增加的运算单元分配给数据操作 i。

第三步: 循环

$t = t + 1$, 如果 $t = D_{\mathrm{II}} + 1$, 算法停止, 否则转第二步。

当算法结束时, 得到的就是运算单元的初始分配。

3 合法变换

定义可分离性与可交换性: 如果 $t_1 > t_2$, t_1、$t_2\in T$, 关于 I_k^p, 有$\forall i\in I_k^p$, $t_1 \notin OP_i$ 且 $t_2 \notin OP_i$, 那么 I_k^p 在时间段$[t_1, t_2]$是可分离的。这时把 I_k^p 分成两个部分, 分别记为 $I_k^p(t_1, t_2) = \{i | OP_i \subseteq (t_1, t_2), i\in I_k^p\}$ 和 $I_k^p - I_k^p(t_1, t_2)$; 如果 I_k^{p1} 和 I_k^{p2} 都分别满足相容性约束, 而且在时间段$[t_1, t_2]$都是可分离的, 那么它们在时间段$[t_1, t_2]$是可交换的, 交换后得到的两个子集$(I_k^{p1} - I_k^{p1}(t_1, t_2)) \cup I_k^{p2}(t_1, t_2)$和$(I_k^{p2} - I_k^{p2}(t_1, t_2)) \cup I_k^{p1}(t_1, t_2)$仍满足相容性约束。这种合法的子集重组是迭代搜索的基础。图 2 是这种交换的一个示例。

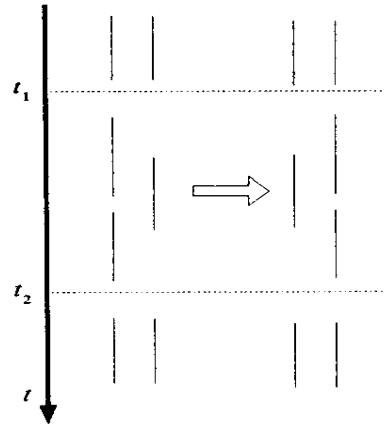

图 2 区间交换示意图

Fig. 2 Illustration for interval interchange

定义 如果 I_k^{p1} 和 I_k^{p2} 在时间段$[t_1, t_2]$上满足: (1) 是可交换的; (2) 在任何时间段$[t_3, t_4]$是不可交换的, 如果$[t_3, t_4] \subset [t_1, t_2]$, 那么定义$[t_1, t_2]$是 I_k^{p1} 和 I_k^{p2} 的最小交换区间。$I_k^{p1}(t_1, t_2)$和 $I_k^{p2}(t_3, t_4)$是最小交换对。如果 I_k^{p1} 和 I_k^{p2} 的最小交换区间是$[l, D_{\mathrm{II}} + l]$, 那么 I_k^{p1} 和 I_k^{p2} 是不可重组的。

性质 如果$[t_1, t_2]$是 I_k^{p1}和 I_k^{p2} 的最小交换区间, 那么$[t_1, t_2]$仍然是交换后两个子集的最小交换区间。

这个性质告诉我们, 子集之间的区间可交换性并不受合法的区间交换的影响。因此可以在迭代搜索之前, 先探测子集之间所有的最小交换区间, 建立数据结构$\{G_k(t_1, t_2) \mid k \in K, t_1, t_2 \in T, t_2 > t_1\}$, 其中图 $G_k(t_1, t_2) = (P, E)$ 反映子集之间的最小可交换性, 图中节点代表 I_k 的子集, 如果 $I_k^{p1}(t_1, t_2)$ 和 $I_k^{p2}(t_1, t_2)$ 是最小可交换对, 那么代表 I_k^{p1} 和 I_k^{p2} 的两个节点是相邻的, 记为 $(p_1, p_2) \in E[G_k(t_1, t_2)]$。

4　随机进化算法

随机进化算法(SE)模拟生物界变异与自然选择的进化过程, 它在标准单元布局、二分图、旅行商等组合问题取得了成功, 其解题质量和速度都胜于模拟退火算法[1]。我们期望随机进化算法同样能较好解决高层综合中的资源分配问题。随机进化算法的第一步是初始化, 构造一个初始解作为寻优搜索的起点, 而且初始化控制参数 P 与迭代计数器, 其中参数 P 控制着搜索过程中新解被接受的概率。完成初始化后, 算法进入一个迭代过程, 每一次迭代都要经历扫描变换、参数更新和改进保存这三个部分。

扫描变换: 扫描变换是由参数 P 控制的由一系列简单变换组成的复合变换, 在随机进化算法中变异是确定性的, 选择是随机的。扫描变换按一定的顺序扫描所有可移动元素, 对其依次实施简单变换, 每得到一个新解 S', 定义 $GAIN = COST(S) - COST(S')$, 如果是 $GAIN > 0$, 则接受 S'; 如果 $GAIN < 0$, 那么与随机数 r 比较, $-P < r < 0$, 如果 $GAIN > r$, 那么接受 S', 否则拒绝 S', 重复这个产生新解 – 拒绝/接受的过程, 直至所有可移动元素的扫描结束。

参数更新: 参数 P 控制着新解被接受的概率。当 P 值接近 0 时, 只有正增益或小负增益的变换才被接受, 算法处于下降模式, 搜索局部最小值; 当 P 值较大时, 高负增益的变换被接受的概率增大, 算法可以走出局部极小, 对应着算法的爬峰模式。为了提高算法的速度, P 的初始值较小, 只有在扫描变换没有改进时, 才增大 P 值, 否则 P 复位为初始值。随机进化算法根据当前搜索结果, 迅速地在下降、爬峰两种模式间切换, 如果连续迭代都没有发现改进时, 那么 P 值将不断被增大, 增加算法的爬峰能力, 使算法迅速地跳出局部极小, 搜索设计空间中的其他区域。而在模拟退火算法中, 参数 T 是单调下降的, 模拟退火算法由爬峰模式(随机游走)过渡到下降模式。

改进保存: 把变换后的解与已搜索到的最优解 S_{best} 比较, 如有所改进, 则把 S_{best} 更新, 计数器减去常数 ET, 否则计数器增 1, S_{best} 保持不变。当计数器增至停止门限时, 整个迭代算法结束。设实际的迭代次数为 IT, 那么 $[IT/ET]$ 为 S_{best} 被改进的次数。

把随机进化算法应用于运算单元分配, 这时把初始分配作为搜索的起始点, 运用随机进化算法改善数据操作的划分方案。扫描变换由四重循环组成, 从里至外这四层循环分别是图 $G_k(t_1, t_2)$ 中的连接边、时间 t_2、时间 t_1 和运算单元类型 k 的扫描。如果 $(p_1, p_2) \in E[G_k(t_1, t_2)]$, 那么把 $I_k^{p1}(t_1, t_2)$ 和 $I_k^{p2}(t_1, t_2)$ 进行交换, 计算交换的得分值, 决定交换是否被接受。

5　设计实例

运算单元分配的中心思想是尽量减少数据通道中的数据通信量, 然后采用启发式节点着色算法完成后续的寄存器分配, 把运算单元和它输出端的寄存器组成一个处理胞元, 最后利用优化算法把这些处理胞元排成一个线型阵列, 使阵列数据总线的总长度最短, 把数据传输映射至总线上。以上这些算法构成了一个完整的资源分配处理流程, 称之为单元线阵资源分配算法。

五阶椭圆滤波器是高层综合常用的测试实例, 测试中结合了定向搜索的流水时序调度算法[7]和单元线阵资源分配算法, 并把设计测试结果与 STAR[2]、SPAID[5] 和 COBRA[4] 等其他算法进行了比较, 结果见表 1。表中可以看出:

(1) STAR 算法生成的数据通道中寄存器的数量最少, 这是因为在 STAR 算法中寄存器面向全局分配, 提高了寄存器共享机会。而单元线阵算法与 COBRA 算法相近, 但比 SPAID 算法的寄存器数量要少。

(2) 考察设计中全局数据总线的数量, 单元线阵算法与 COBRA 算法的性能最好, 基本相近。但在 COBRA 的数据通道中, 每个处理胞元还存在有三条本地数据总线, 可支持处理胞元内部各种数据传

输,而本文采用的数据通道中,处理胞元内部连接更简单,局部互连支持两种数据传输:运算结果存入寄存器和处理胞元内部寄存器之间数据的转移,它并不是总线结构。

表 **1** 五阶椭圆滤波器设计实例中几种综合算法结果的比较

Tab. 1 A comparison of synthesis results for the 5th-order elliptic filter between various algorithms

	D_{II}	流水乘法器	加法器	寄存器	数据总线
单元线阵算法	16	1	3	14	4
	17	1	2	12	3
	18	1	2	14	3
	19	1	2	11	3
	20	1	2	12	3
	21	1	2	11	3
COBRA	18	1	2	13	3
	19	1	2	11	3
SPAID	16	2	4	19	7
	18	1	2	21	6
	19	1	2	19	5
	21	1	2	19	4
STAR	16	2	3	12	6
	17	1	2	11	5
	19	1	2	11	4

参考文献:

[1] Saab Y G. Combination Optimization by Stochastic Evolution[J]. IEEE TCAD, 1991, 10(4).

[2] Tsai F, Hsu Y. STAR: An Automatic Data Path Allocator[J]. IEEE TCAD, 1992, 11(9).

[3] Duncan A A, Hendry D C. DSP Datapath Eliminating Global Interconnect[J]. Proceeding of IEEE ICCAD, 1993.

[4] Duncan A A, Hendry D C. High-level Synthesis of DSP Datapaths by Global Optimization of Variable Liftimes[J]. IEE PCDT, 1995, 142(3).

[5] Haroun B S, Elmasry M I. Architecture Synthesis for DSP Silicon Compliers[J]. IEEE TCAD, 1989, 8(4).

[6] Gupta U I. An Optimal Solution for the Channel-assignment Problem[J]. IEEE Trans. on Computer, 1979, 28(11).

[7] 欧钢. 一种应用于数据通道综合的定向搜索流水调度算法[J]. 国防科技大学学报, 2002, 24(2): 36-43.