一种同时进行资源分配和布局规划的高层次综合算法

王 磊,鲁瑞兵,魏少军
(清华大学微电子所,北京 100084)

摘 要: 随着集成电路的特征尺寸不断缩小,连线延迟成为决定电路性能的主要因素之一,减小连线网络复杂度已成为高层次综合算法的一个重要内容. 本文提出了一种同时进行资源分配和布局规划的算法,使用最小割(Min-Cut)算法对已调度的数据依赖图(DFG)进行多路分割,实现了资源分配,同时又把分割的过程对应到 Slicing 结构的布局规划中. 在算法进行过程中可以不断利用前面步骤所提供的布局信息指导资源分配,从而有效的对连线进行优化. 针对设计实例进行的实验表明了这种算法的有效性.

关键词: 高层次综合; 分配; 布局规划
中图分类号: TN407 **文献标识码**: A **文章编号**: 0372-2112 (2002) 05-0766-03

Simultaneous Allocation and Floorplan Algorithm

WANG Lei, LU Rui-bing, WEI Shao-jun
(Institute of Microelectronics of Tsinghua University, Beijing 100084, China)

Abstract: With the feature size of VLSI scaling down, interconnection delays begin to dominate the circuit performance. Interconnect nets reduction become an important part of high level synthesis. We present an algorithm which cope allocation and floorplan problems simultaneously, we use Min-cut method to multi-partition scheduled DFG, implementing resource allocation as well as mapping the partition procedure to Slicing structure based floorplan. During the partition procedure, floorplan informations are used to direct allocation, thus interconnections are efficiently optimized. Design examples are presented to help concluding that our algorithm is very efficient.

Key words: high level synthesis; allocation; floorplan

1 引言

传统的高层次综合方法主要是面向电路功能单元的,对面积和性能的估计基本上忽略了连线的影响. 但随着特征尺寸的不断减小,连线在一定程度上决定了电路的性能,人们在进行高性能的数据通道设计时不得不考虑芯片物理布局因素的影响. 简单的,只用 MUX 输入端的数量、总线和资源的数目来衡量设计的性能和面积已经无法适应新的设计要求.

在近期的研究中,人们广泛采用了将高层次综合和布局规划结合起来的算法,使得在高层次设计的过程中可以得到芯片的物理信息. 3D 算法[1]是一种同时进行算子调度,资源分配和布局规划的运算方法,虽然可以在高层设计时得到一定的连线信息,然而,由于这是一种优先考虑关键路径的算法,因而无法保证非关键路径的延时,而且当电路的存在多条关键路径时,此算法也很难奏效. GB 算法[2]将资源分配和布局规划问题转化为算子在二维网格的放置问题,此网格的一维表示控制步,另一维表示布局规划. 然而,此算法所采用的一维布局规划有很大的局限,很难进行实际的使用. 另外,还有一些考虑深亚微米效应的高层次综合算法[3~5],他们的核心思想是依次进行资源调度和布局规划,把布局规划的结果作为资源调度的评价函数,对资源调度进行迭代改善. 这种算法的计算量非常大,而且由于资源调度的调整对布局规划的影响是不可预测的,因而无法保证迭代的收敛性.

本文提出了一种同时进行资源分配和布局规划的方法,同以前的算法比较,该算法有以下特点:首先,该算法在资源分配的过程中同时产生布局规划,避免了依次进行这两个步骤所带来的迭代,算法的复杂度较小,收敛性可以保证. 其次,该算法能够很方便的同时实现存储资源和运算资源的分配,可以综合考虑它们之间的相互影响,避免了一方的不确定性给另一方的优化结果带来的不确定性. 第三,这里采用的基于 Slicing-Tree 结构[7]的最小割布局规划算法是一种经过实践检验的比较完备的算法,可以满足布局规划的许多实际要求.

2 算法描述

2.1 问题定义

高层次综合处理的对象是控制数据流图(CDFG),而资源分配是将调度完成之后的数据流图中的算子映射到寄存器传输级(RTL)电路中的运算资源上,把变量映射到存储资源上. 我们用调度完成的数据流图用 $G = (V, E)$ 表示,这里 V 表示

收稿日期:2000-03-23;修回日期:2001-12-01

算子集合,E 为数据传输关系集合. $v_i, v_j \in V$,有向边 $e_{ij} \in E$ 表明算子 v_i 的运算结果传给算子 v_j. $Step(v_i)$ 表示算子 v_i 执行运算的所在的控制步. $RsType(v_i)$ 表示算子 v_i 的运算类型. 资源分配可以看成是对数据流图用 G 的分割,求出子图 $G_1(V_1, E_1), G_2(V_2, E_2), \cdots G_N(V_N, E_N)$,各子图的节点集满足:

$$V_i \cap V_j = \Phi,\ \forall 1 \le i < j \le N,\text{且有} \bigcup_{i=1}^{N} V_i = V$$

同时任一节点集 V_k 满足:

$$RsType(v_i) = RsType(v_j),\ Step(v_i) \neq Step(v_j),\ \forall v_i, v_j \in V_k \quad (1)$$

这样每一个节点集 V_k 中的所有算子可以映射到同一资源上. 衡量资源分配结果好坏的重要依据之一就是连线的复杂度,而且由于连线延迟在深亚微米 VLSI 中对电路的整体性能起到决定性的作用,所以,对数据流图 $G(V, E)$ 的多路分割的目标就是缩小各个子图之间的连线复杂度.

2.2　分割的实现方法

为了计算方便,定义超级图 $G'(S, E')$,把它作为分割的对象. 该图由数据流图 $G(V, E)$ 派生而来,每个超级节点 $s_i \in S$ 可以包含一个或多个 $v_k \in V$,但要求 $\forall v_k \in s_i, RsType(v_k) = R$,这里 R 为一固定值,同时定义 s_i 的资源类型 $RsType(s_i) = R$. E 和 E 意义基本相同,只是前者代表的是超级节点 s_i 之间的相互关系. 最初的待分割超级图 $G'_0(S, E')$ 根据下式产生:

$$s_i = \{v_i\},\ E' = E.$$

在迭代的对超级图 $G'(S, E')$ 进行二分的过程中,如果所得的某子超级图中的所有资源类型为 R 的超级节点所包含的算子节点 v 组成的集合满足式(1)要求的条件,将这些超级节点合并,即将这些超级节点所包含的节点集合并,形成一个超级节点. 这样,在每一次分割之后都对产生的子图进行次检查,如果可以就进行节点合并. 在分割结束时,最后的每个子超级图中只有一个超级节点,而这些超级节点所对应的算子节点集就是我们要求的分割结果.

我们用最小割算法[6]来实现前面提到的图的二分. 然而,由于在分割时同时要顾及连线和布局,与传统的最小割算法不同的是,分割的目标函数不再仅仅由切割线的权重和决定. 把数据流图 DFG 分成两个子图,每个子图用独立的电路模块实现,不同的分割方法可能导致分配所需的资源数目的差异,从而影响电路的面积. 因而,需要将电路的面积作为目标函数的一部分. 下式为在把一个超级图分割成 A, B 两部分时采用的目标函数.

$$f(A, B) = \sum_{\substack{i \in A \\ j \in B}} c'_{ij} + \alpha \cdot \sum_{\substack{i \in A+B \\ j \notin A+B}} k_{ij} \cdot c'_{ij} + \beta \cdot \{Area_A + Area_B + \gamma \cdot |Area_A - Area_B|\} \quad (2)$$

上式主要由 3 个部分组成,第 1 部分表示被切割线的权重和,其中 c'_{ij} 表示超级图 $G'(S, E')$ 中边 e'_{ij} 的权重值. 第 2 部分表示待分割图以外的节点对分割的影响,其中的系数 k_{ij} 和物理布局有关. 第 3 部分是分割成的子图对应的电路的面积估计值之和,$Area_A$ 和 $Area_B$ 分别对应于分割所得的 A 和 B 两个子图,面积代价项还加上一项两部分面积差,其目的是要鼓励均匀分割. α、β 和 γ 分别为各项的加权系数.

在式(2)中,面积的估计值用所需资源的面积总和来表示:

$$Area = \sum_{i \in \Phi} area(i) \cdot \max_{0 < j < StepNum} \{N(i, j)\} \quad (3)$$

在式(3)中 $N(i, j)$ 表示第 j 控制步的第 i 种资源的数目,$StepNum$ 为总控制步数,Φ 为资源类型的集合,$area(i)$ 表示 i 类型算子的面积.

2.3　连线权重值计算

因为连接资源的延时不同以及所处的位置不同,不同的连线对整体电路的性能影响是不一样的,特别是关键路径上的连线,它们将最终决定电路的性能. 所以,连线的权重应由它们对电路性能的影响程度来决定.

一条连线对电路性能的影响与此连线所在的延时路径有关,如果该延时路径上的资源数目越多,资源延时越大,将对此延时路径上连线的延时要求越高. 不妨设某控制步内的一条延时路径上有 N 个算子,算子 i 针对的资源的延时为 t_i,目标时钟周期为 T. 对于处于同一控制步内的连线(如图 1 线 a),其权重值应和连线期待的延时成反比,可由下式计算:

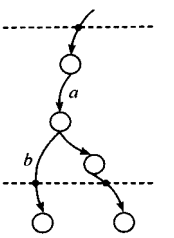


图 1　连线权重计算

$$e = \eta \cdot (N + 1) / \left(T - \sum_i t_i\right) \quad (4)$$

式中 η 为归一化因子,使权重的最大值为 1.

处于多个延时路径中连线,权重值应由其中连线延时要求最严格的那条路径来决定.

跨越两个或更多周期(如图 1 线 b)的连线,由于这种连线在最终实现时中间存在寄存器,因而,它对应的延时应包括两部分,即上下两个周期内的预期延时之和. 如果用 e_a 和 e_b 分别表示该连线在信号产生和结束周期内的权重,则它的权重值由下式计算:

$$e = \frac{1}{1/e_a + 1/e_b} = \frac{e_a \cdot e_b}{e_a + e_b} \quad (5)$$

2.4　寄存器分配问题

传统的资源分配方法将寄存器分配和运算资源分配分开,依次进行. 然而,由于跨越两个或两个周期以上的数据传输线中间必须加入存储资源(通常为寄存器),这类资源之间的连线长度仅仅用资源之间的距离来估计将变的不准确.

我们将寄存器当作一种可跨越多个周期的资源,相应增加一种寄存器算子类型,并把它插入数据流图中,对数据流图进行改造. 这样的数据流图可以更准确的反映 RTL 级电路的实际情况. 而进行了这样的改进之后,算法在资源分配和布局规划中同时考虑了寄存器的影响,可以更准确的估计连线复杂度,以达到更好的优化结果.

3　实验结果

我们实现了这个算法,并把它应用于两控制步、未经流水线操作的 FIR 滤波器.

为了进行比较,我们首先(方法 1)采用传统的设计方案,先进行资源分配,再对生成的电路产生布局规划. 资源分配采用了文献[5]中提供的结果. 布局规划采用的是基于 Slicing 结

构的最小割布局规划算法. 其次(方法 2)采用的就是在本文中我们提出的同时进行资源分配和布局规划的算法. 表 1 列出了用于实验的库参数,它们取自文献[5]. 表 2 给出了实验的结果. 比较结果可以看出,方法 2 使控制步 2 的关键路径上的连线长度减小,控制步 1 连线长度不变或略有增加. 由于电路的最终的时钟周期是由关键路径延时最长的周期决定,而在 FIR 这个实验电路中,控制步 2 的关键路径上的运算资源数目,连线数目,以及运算资源的累计延时都远大于控制步 1,因而电路的控制步 2 决定了电路的性能. 所以,从实验结果中我们可以看出,本算法可以减小电路的关键路径上的连线长度,从而提高电路性能.

表 1 库单元参数

制造工艺	16Bit 加法器		16Bit 乘法器	
	延时(ns)	面积(mm^2)	延时(ns)	面积(mm^2)
1.5μm	36	1.17	123	9.38
1.2μm	35	0.69	959	5.40
0.8μm	24	0.33	82	2.67
0.5μm	12	0.39	41	2.87

表 2 运算结果比较

制造工艺	方法 1		方法 2		减小比例
	控制步 1	控制步 2	控制步 1	控制步 2	
1.5μm	684.69	906.98	684.69	906.98	0.0 %
1.2μm	520.77	691.86	573.77	638.86	7.6 %
0.8μm	363.32	626.72	363.32	481.40	23.2 %
0.5μm	341.27	661.74	382.10	511.52	22.7 %

我们与另一种同时进行高层次综合和布局规划的 3D 算法也进行了比较,采用了表 3 所列的库参数. 表 4 中列出了这两种算法的关键路径上连线的总长度比较,可以看出本算法使关键路径上的连线长度减小 40 %左右.

表 3 3D 算法使用的库参数

制造工艺	16 位加法器		16 位乘法器	
	延时(ns)	面积(mil^2)	延时(ns)	面积(mil^2)
1.6μm	18	1195	200	13938
1.2μm	13	672	150	7840
0.35μm	10	32	20	526
0.18μm	4.13	1.73	15.2	31.5

表 4 与 3D 算法的结果比较

制造工艺	3D		Our		减小比例	
	控制步 1	控制步 2	控制步 1	控制步 2	控制步 1	控制步 2
1.6μm	273	565	256	325	6.2 %	42.5 %
1.2μm	204	423	191	244	6.4 %	42.3 %
0.35μm	54	112	48	60	11 %	46.4 %
0.18μm	13.1	27.6	11.8	14.5	1 %	47.5 %

4 总结

在这篇论文中,我们提出了一种新的同时进行资源分配和布局规划的算法,其主要目的是利用布局规划提供的芯片布局信息,指导资源分配,以产生连线复杂度更为简化的电路. 与其他的算法相比,该算法利用分割数据流图的方法,同时实现资源分配和布局规划,避免了迭代,使算法的复杂度降低. 连线权重值由该连线对电路整体性能的影响程度决定,从而可以使关键路径得到优先考虑. 另外,利用本算法可以很方便的同时实现存储资源和运算资源的分配,综合考虑它们之间的相互影响,避免了一方的不确定性给另一方的优化结果带来的不确定性.

参考文献:

[1] Weng Jen-Pin, Alice C Parker. 3D scheduling: high level synthesis with floorplanning [A]. 28th DAC [C]. 1991, 7.

[2] Jang Hyuk-Jae, Barry M Pangrle. A grid-based approach for connectivity binding with gemetric costs [A]. Proc ACM ICCAD [C]. 1993, 11.

[3] Fang Yung-Ming, D F Wong. Simultaneous functional-unit binding and floorplanning [J]. IEEE Trans, ICCAD, 1994, 11: 317 - 321.

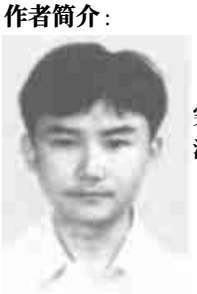
[4] A Safurm, B Haroun, K Thulasiraman. Floorplanning with datapath optimization [A]. Proc IEEE International Symposium on Circuits and Systems [C]. 1995, 4.

[5] Vasily G Moshnyaga, Hiroshi Mori, Hidetoshi Onodera, Keikichi Tamaru. Layout-driven module selection for register-transfer synthesis of sub-micron ASICs [A]. Proc IEEE ACM ICCAD [C]. 1993, 11.

[6] C M Fiduccia, R M Mattheyses. A linear-time heuristic for improving network partitions [A]. Proc IEEE ACM ICCAD [C]. 1982.

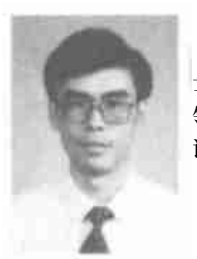
[7] D P Lapotin, S W Director. A global floorplanning approach for VLSI design [J]. IEEE Trans, on ICCAD, 1986, 11.

作者简介:



王 磊 男,1976 年 6 月生于西安,博士研究生,主要研究领域为面向互连的高层次综合方法.

鲁瑞兵 男,1974 年生,硕士研究生,主要研究领域为深亚微米集成电路设计方法学.



魏少军 男,1958 年 5 月生于北京,教授,博士生导师,电子学会高级会员,IEEE 会员,研究领域为集成电路设计方法学,通信专用集成电路设计等.