第33卷 第13期 计 算 机 工 程 2007年7月
Vol.33 No.13 Computer Engineering July 2007
·博士论文· 文章编号：1000—3428(2007)13—0040—03 文献标识码：A 中图分类号：N945

VLSI高层综合设计中的调度和互连

温东新，王 玲，杨孝宗
(哈尔滨工业大学计算机科学与技术学院，哈尔滨 150001)

摘 要：VLSI 高层次设计技术是近年来系统设计自动化研究的主要方向，高层次综合设计是高层次设计技术的关键，其主要任务是调度和互连。该文介绍了若干基本的调度和互连算法，提出将 DVS 技术应用于高层次综合设计中，实现在满足任务行为的约束条件下，动态改变时钟的速度和电源电压达到降低功耗的目的，制定了可行的研究实施方案。

关键词：VLSI；高层综合设计；调度；DVS；互连

Schedule and Interconnection at High Level Synthesis in VLSI

WEN Dongxin, WANG Ling, YANG Xiaozong
(Department of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001)

【Abstract】 High-level design technology in VLSI is a main trend in EDA, while high level synthesis is the key in it. Scheduling and interconnection is an important task at high level synthesis. This paper introduces some algorithms and provides a new idea of application of DVS technology which can dynamically change clock speed and supply voltages under satisfying performance and time constraints in order to reduce power consumption. It still makes, a feasible research plan.

【Key words】 VLSI; high level synthesis; schedule; DVS; interconnection

1 概述

在VLSI设计中可以分成不同的设计层次，从工艺处理层、版图层、电路层、逻辑层、结构层和算法层（行为层）到系统层。对不同层次的电子设计可以进行几何描述、逻辑描述或结构描述。硬件设计过程就是精炼细化概要设计或低层次概要设计模型。随着笔记本、掌上电脑等手持式便携系统的应用，低功耗成为设计的主要目标，VLSI高层次设计技术成为系统设计自动化研究的主流。VLSI高层次设计技术[1]是近年来国际在EDA技术领域研究开发应用的热门课题，它能够自动实现高层次设计到低层次抽象的转换，即从行为层描述转成结构层描述，或从结构层描述到物理层描述，如图1所示。高层综合设计中包括行为调度和功能单元的互连。

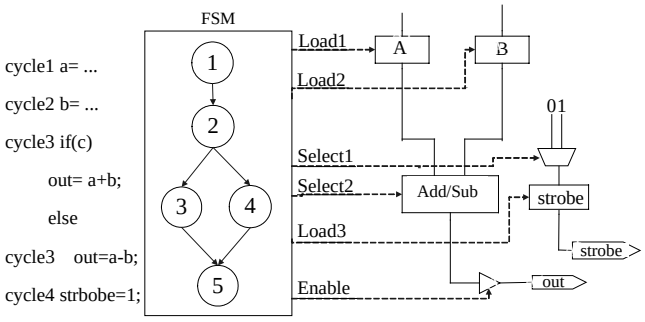


图1 高层综合设计举例

2 调度

调度是高层设计过程中最重要的任务，决定所有操作行为的执行顺序[2]。对于CDFG(control/data flow graph)图中的每个操作，ASAP算法能够实现在尽可能早的周期进行调度，ALAP算法能够实现在尽可能晚的周期进行调度，如果一个数据流图执行需4个周期，2种算法实现如图2所示，实现的前提条件是在每个周期必须提供必要的硬件资源。

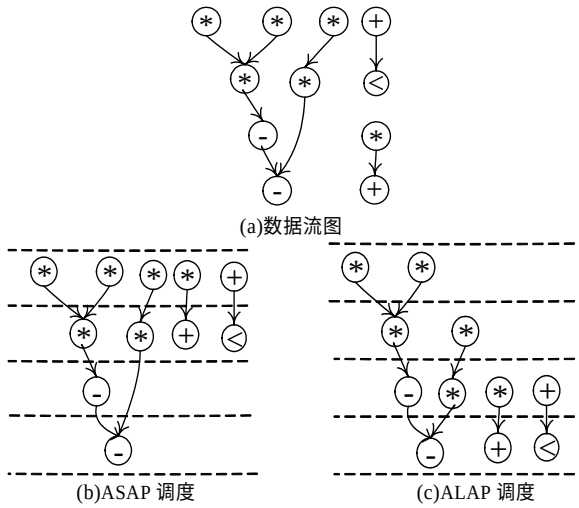


图2 算法的实现

2.1 时间约束下的调度算法

时间约束条件下的调度算法也称为固定周期数方法，对于实时系统设计非常重要，包括FD算法、ILP算法与IR算法。3种算法的比较见表1[2]。

2.1.1 FD(force directed)算法

该算法从数据流图的全局考虑，均匀控制执行各操作，以便更加高效地利用资源。首先，为每种类型的操作做出控

基金项目：黑龙江省自然科学基金资助项目(2004F17)；哈尔滨工业大学引进人才启动基金资助项目

作者简介：温东新(1971－)，女，博士研究生，主研方向：计算机体系结构，高层综合设计；王 玲，博士、副教授；杨孝宗，教授、博士生导师

收稿日期：2006-07-10 **E-mail：**wdongxin@hit.edu.cn

制分配图(图 3)，该图充分说明操作所具有的并行性。力度是并行操作的量度，通过计算力度决定操作的执行周期。依据分配图计算同类型操作的力度是因为同类型操作可以共享一个函数模块，达到高效利用资源的目的。计算出分配图中每个周期同种类型操作的力度，将确定操作在其灵活度范围内某个周期进行调度。如果一个操作的灵活度为 K 个周期，则 K 个周期中的每个周期的分配值为 $1/K$，由 CDFG 图得到的分配图中操作 O 的允许调度周期 i 的力度公式为

$$F_i = \sum_{j\in[o.earliest\ ,o.latest\ ]} DG_j \times \Delta(j,i) \tag{1}$$

其中，j 是灵活度；DGj 是该操作类型在分配图中周期 j 的分配值；$\Delta(j,i)$ 是将操作 O 由周期 j 改为 i 的分配值的变化量。

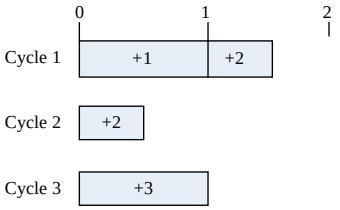


图 3　DFG 图的加法操作分配图

假定时间约束条件是 3 个周期，+1 和+3 要求分别在周期 1 和周期 3 被调度，而+2 可以在周期 1 或周期 2 被调度，所以，+2 在周期 1 和周期 2 的分配值为 1/2。+2 在周期 1 被调度的力度为

$$F_1 = 1.5*0.5+0.5*(-0.5) = 0.5 \tag{2}$$

式(1)的第 1 项中+2 在周期 1 的分配值从 0.5 变为 1，第 2 项中+2 在周期 2 的分配值从 0.5 变为 0。同理，+2 在周期 2 被调度的力度为

$$F_2 = 1.5*(-0.5)+0.5*0.5 = -0.5 \tag{3}$$

计算出所有可能情况的力度，因为它表示每个周期同种类型操作的负荷，所以选择力度最小值的情况进行调度，比较上例中的 2 个力度值，最终的+2 在周期 2 被调度。

2.1.2　ILP(integer linear programming)算法

ILP 调度形式为整型线性程序问题，表示为

$$Minimize\ \ C_{\text{step}} \tag{4}$$

$$\sum_{o_i\in FU_K} x_{ij} - N_K \le 0,\ for\ 1\le j\le s, 1\le k\le m \tag{5}$$

$$\sum_{j=o_i.earliest}^{o_i.latest} x_{ij} = 1,\ for\ \ 1\le i\le n \tag{6}$$

$$\sum_{j=o_i.earliest}^{o_i.latest} j\times x_{ij} - T_i = 0,\ \ for\ \ 1\le i\le n \tag{7}$$

$$T_i - Tj \le -1, \forall o_i \to o_j \tag{8}$$

$$T_i - C_{step} \le 0, \forall o_i\ \ without\ successors \tag{9}$$

其中，X_{ij}为 0 或 1，如果操作O_i在周期j被调度，X_{ij}=1，否则X_{ij}=0；T_i是O_i被调度的周期；C_{step}是调度所需的周期总数；m为函数模块类型数量。

式(4)为目标函数，即周期总数。式(5)是对任意周期j类型k操作的最多操作数N_k的限制。式(6)意味着操作O_i可以在它的灵活度范围内任意个周期被调度。利用式(7)可以计算出操作O_i的调度周期。式(8)是保证操作先后关系的条件。　式(9)是在C_{step}后不再调度任何操作。

2.1.3　IR 算法

IR(iterative refinement)算法是实现操作的重新调度算法。任一个初始调度只是根据数据依赖关系尽早或晚考虑操作的调度，如果某个调度有几种可进行的变化，可以选择任意一种变化，所有变化都可以作为候选方案并计算出每一种的消费，最小消费方案是选定的目标。

表 1　3 种调度算法比较

	ILP	FD	IR
计算复杂性	很高	低	高
调度效果	最佳	较好	接近于优化
空间复杂性	中	低	很高
输入问题规模	小	任意	不能过大
执行速度	低	高	中
应用技术	数学程序	结构化技术	提取技术
使用情况	少	多	中

2.2　资源约束下的调度算法

调度算法的目的是在满足资源约束的条件下尽可能进行合理化的设计，调度一经建立便不会违背资源限制条件下的数据依赖关系，每一步中的各个操作不能超过所能获得的功能单元。算法包括图表法与静态图表法，二者的比较见表 2。

表 2　2 种调度算法比较

	图表法	静态图表法
计算复杂性	高	不很高
调度效果	近似最优	近似最优
空间复杂性	很高	不高
输入问题规模	任意	任意
执行速度	较慢	较快
应用技术	若干优先表	一个优先表

2.2.1　基于图表的调度算法

图表调度算法就是在资源约束下的 ASAP 算法，该算法中的图表按优先顺序存放预备结点（即祖先结点已被调度的节点），优先顺序由优先函数决定，在每次迭代中，优先函数值高的操作先于值低的操作被调度。调度一个操作数后，其后的操作加入优先表中。图表调度算法主要依靠优先函数的设计，设计合理的优先函数是至关重要的。

2.2.2　静态图表调度算法

该方法在调度前创建一个优先权表，与基于图表调度不同，无须安排控制步骤和保存优先权表，首先利用 ASAP 算法和 ALAP 算法获得每个操作的 LCS(least control step)及 GCS(greatest contol step)的控制周期安排，将所有操作按 GCS 升序分类，将相同的 GCS 再按 LCS 降序排序。当优先权表创建后，根据优先权表中的操作开始连续调度。

2.3　其他调度算法

除了上述调度技术外，还有 2 种重要方法：模拟退火和基于路径的调度方法。

2.3.1　模拟退火

调度由(控制周期，函数单元)二维表表示。一个函数单元在给定的控制周期中仅能执行一个操作，入口不能有多个操作占有。在初始调度时算法迭代地修改二维表，每次置换一个入口，然后计算置换的消费，较差的调度会以一定概率接受。该算法的优点在于可以跳出局部优化，实现全局最优化。

2.3.2　基于路径的调度

该调度算法实现了数据流图中的关键路径控制周期最小化。首先确定所有可能执行路径独立的调度，然后不同的路径调度依据结点间的限制条件再合并生成最终的调度。

3　调度中低功耗的实现技术

在CMOS电路中的功耗定义如下[3]：

$$P_{avg} = P_{swcap} + P_{short\text{-}circuit} + P_{leakage} + P_{static}$$

其中，$P_{sw.cap.}$是充放电功耗，由电路中的电容充放电引起；

$P_{short-circuit}$ 是短路电流功耗；$P_{leakage}$ 是电流渗漏引起的功耗；P_{static} 是静态功耗。假定电路的时钟频率为 f。

$$P_{sw.cap.} = \frac{1}{2} C_L * V_{dd}^2 * N * f$$

其中，N 是每个时钟周期 CMOS 反相器输出信号的翻转率，又称为信号活性。充放电功耗在 CMOS 电路中起决定作用，所以，多数功耗优化技术以减少电路中的功耗部件为目的。

集成电路的性能和复杂度在近 20 年中得到飞速发展，功耗已经成为集成电路设计时首要考虑的因素。因为供应电压与功耗的平方关系，所以降低供应电压是最有效的降低功耗的方法。然而，减小供应电压会导致电路的延迟和流量的减小。为解决上述问题，可以采用并行结构或流水结构来保持流量，提出了利用遗传算法减小功耗。最新研究表明在系统级应用DVS技术能够协调高性能和低功耗的矛盾。DVS技术是在满足任务行为的约束条件下，动态改变时钟的速度和供应电压，达到降低功耗的目的。实时系统中有任务内部（一个任务中的子任务间）和任务之间（多个任务）的动态电压调整（DVS）方法，主要依靠调整的单位划分[4]。而本文提出将DVS技术应用于高层综合设计中动态改变时钟的速度和供应电压，以降低芯片功耗。

图 4 是一段程序代码及其相应的CFG图，图中的每个结点内部的数值表示该操作的执行周期数$C_{EC}(B_i)$，B_5到B_{wh}的逆向边表示程序的循环部分，图 4 表示循环的次数为 3 次。利用静态程序分析技术可以得到每个模块的执行时间。$C_{RWEC}(B_i)$表示从B_i开始所有路径中余下的最多执行周期数，图中“ [] ”内是每个模块的$C_{RWEC}(B_i)$，在$C_{RWEC}(B_i)$计算后，判定满足$[C_{RWEC}(B_i)\text{-}C_{EC}(B_i)] \neq C_{RWEC}(B_j)$的边$(B_i,B_j)$，这些边便是电压调整候选点，见图中带有*的边。因为余下的执行周期数为$[C_{RWEC}(B_i)\text{-}C_{EC}(B_i)\text{-}C_{RWEC}(B_j)]$，可以降低时钟的速度为$S \times \frac{C_{RWEC}(B_j)}{C_{RWEC}(B_i) - C_{EC}(B_i)}$，其中，$S$是模块$B_i$的执行速度。假定供应电压与时钟的速度成正比，便可以计算出调整后减小的供应电压[4]，达到减小功耗的目的。

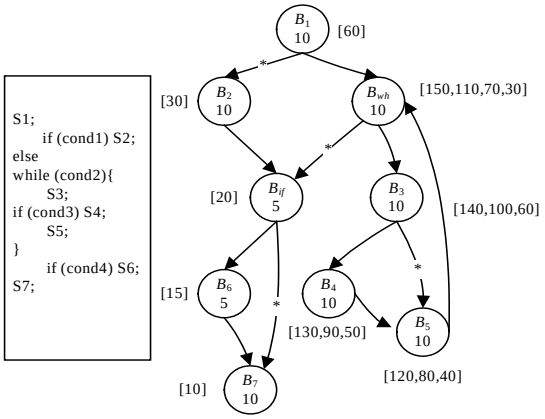


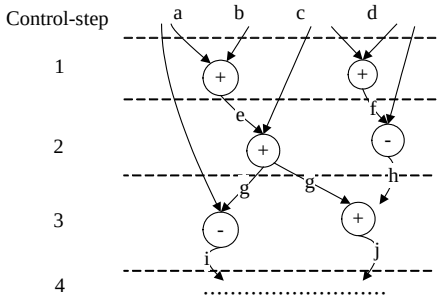
图 4　一个程序及其 CFG 图

4　互连

仅考虑调度是不够的，电路中互连部分在电路总功耗中也是主要的因素，电路中的互连包括线路连接、缓冲器、时钟、多路选择器以及总线的分配。文献[5]中通过绑定数据总线传输数据优化总线功耗。文献[6]于高层综合设计中考虑物理层设计信息，在 CDFG 中描述各边的转换力用于优化模块间的数据传输功耗，还可以通过保持输入行为的位置和均匀度来优化总线结构。文献[7]考虑了信号线的耦合电容，分析了因信号线的转换和线间的耦合电容引起的总线功耗散失，并由分布式的 RS 互连模型得出

$$P_{dyn} = (X_T * (C_s + C_l) + Y_T * C_c) * V_{dd}^2$$

其中，C_S与C_l为自身电容；C_c为耦合电容；V_{dd}为电源电压；X_T与Y_T为C_S、C_l、C_c在T时钟周期内的有效转换次数。图 5(a)是调度后的DFG图，在每个控制步骤中的数据传输情况如图 5(b)所示，在进行数据总线的绑定时要尽量减小数据传输的功耗。



(a)

控制步	数据传输
1	a, b, c, d
2	c, d, e, f
3	a, g, h
…	…

(b)

图 5　调度后的 DFG 图及数据传输

由其消费表(表 3)及互联算法选定消费最小的绑定结果：a绑定与B_2，g绑定与B_3，h绑定与B_1。

表 3　数据总线绑定消费表

	B_1	B_2	B_3	B_4
a	156	137	58	184
g	154	154	46	178
h	155	147	48	182

5　实施方案

综上所述，DVS 技术在高层综合设计中的关键是根据 CDFG 图中所有路径中各结点的执行灵活度，编写算法实现电压调整点的插入，最终实现减小供应电压，减小芯片功耗的目的。研究方案如图 6 所示。

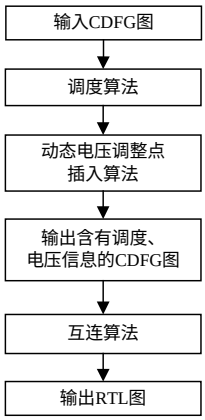


图 6　DVS 技术的实施方案

下一步的研究工作是基于基本的调度算法及系统级的动态电压调整算法，完善已有的高层次综合的调度算法、DVS 算法和互连算法，在 CDFG 工具包集成环境利用 VC++进行编程仿真，以便更加有效地实现 VLSI 中芯片的低功耗设计。

（下转第 74 页）