文章编号 :1001-2486 (2001) 01-0066-07

# 应用于流水时序调度的归一化定时数据流图理论*

欧钢

(国防科技大学电子科学与工程学院, 湖南 长沙 410073)

摘 要 :流水时序调度是专用数字信号处理器高层综合中的一个困难而急待解决的问题,文中提出了一种有着鲜明物理意义的归一化定时数据流图,基于节点移动研究了合法流水调度变换的内部机理, 从而证明从任一合法的初始流水调度出发,通过合法的节点移动可以搜索到设计空间中任何一个合法的流水调度。一个合法、完备的变换集,为寻优搜索的算法应用于流水调度解决了理论和算法实现问题。文中还给出了一些实验结果。

关键词 :高层综合 ;流水时序调度 数字信号处理器

中图分类号 :TN911.72 ;TP38 文献标识码 :A

## The Theory of Normalized Scheduled Data Flow Graph Applied in Pipeline Scheduling

OU Gang

(College of Electronie Science and Engineering, National Univ. of Defense Technology, Changsha 410073, China)

**Abstract** :In the high level synthesis of application-specific DSP, pipeline scheduling is a difficult and urgent problem. In this paper the theory of normalized scheduled data flow graph(NSDFG) which has the clear physical meaning is discussed . Based on the node movements in the NSDFG, the intrinsic mechanism of legal pipeline scheduling is studied in detail. The study proves that from any initial point in the design space any legal scheduling can be searched through the legal node movements . The self-contained set of legal transformations derived in this paper provides a key for the search-based algorithms to be applied in pipeline scheduling. Results for some examples are presented also.

**Key words** :high-level synthesis ; pipeline scheduling ; digital signal processor

专用数据通道自动化设计的核心之一是时序调度。时序调度是数据流图到时间的映射,数据流图被分割为几个子图,每一个子图对应一个控制步,同一子图中的节点代表并行的数据操作。在数字信号处理的场合,原始数据一般以规则的方式进入处理器,处理算法的每一个循环对应着一个数据的输入周期。流水时序调度把处理算法的一个周期分成几个长度相等的执行段,相邻周期的不同执行段并行执行。由于在周期内部和周期之间会同时存在数据操作的相关关系,而且这种相关关系还可能是递归的,因此流水调度要比非流水时序调度更加复杂和困难。为了挖掘周期之间数据操作的并行性,有些设计系统(文献[1,2])首先运用重定时变换和流水变换对数据流图进行优化,然后再采用一般非流水的调度算法来完成时序映射。但是,如果把流水调度的寻优搜索归结为数据操作节点在执行段内部和执行段之间移动的过程,那么流水调度中已经隐含了流水变换和重定时变换,计算结构的优化和时序调度已融合为同一个优化过程。因此在专用数字信号处理器的高层综合中应用流水调度,应能比一般时序调度得到更加优化的结果。

通过迭代改善一个合法的初始调度来搜寻最优的时序调度是很多时序调度算法的思路,其中各种寻优算法,如模拟退火算法、遗传算法等,都有所应用,但是应用一般仅限于非流水的时序调度。基于变换搜索的流水调度算法的关键之一是流水调度的变换方法。这种变换应该具有这样的性质 :(1)合法性 :合法调度的变换仍是合法的流水时序调度 ;(2)完备性 :任何两个合法的流水调度可以通过有限步的变换进行相互转换,或者说,从任何一个初始调度出发,通过变换可以搜索到设计空间中的任意一个点。本文通过提出归一化定时数据流图,即一种流水时序调度的图形表示方法,研究了流水时序调度的一些

* 收稿日期 :2000-09-05
作者简介 :欧钢(1970-),男,副教授,博士。

内在规律,并且得到一个合法完备的变换集,从而为变换搜索算法应用于流水时序调度扫清了障碍。

## 1 定义

**定义 1** (数据流图):数据流图(DFG)是一个赋权的有向图,由一个 4 元组($J$,EXE,$E$,$D$)描述,其中:

·$J$ = {INPUT,OUTPUT} ∪ I,INPUT、OUTPUT 分别是输入节点和输出节点,$I$ 是数据操作节点集合;

·$\mathrm{EXE}_i$,$i \in I$:数据操作 $i$ 的延时;

·$E \subseteq J \times J$,是代表节点之间数据相关关系的赋权有向弧的集合;

·$D(e) \in N$,是边的权重,等于延迟器的个数,$\forall e = (i \to j) \in E$,$N$ 是自然数集合,它表示数据操作 $i$ 所生成数据经过 $D(e)$ 个周期后被数据操作 $j$ 利用,成为它的操作数。

流水调度有两个重要的时间参数 数据重载周期和最大控制步,分别用符号 DII 和 $\mathrm{T_{MAX}}$ 表示,单位为控制步。数据重载周期等于分解后流水段的长度,是数据通道连续两次接收输入数据的最小时间间距,它决定了处理器所能达到的最高数据速率。最大控制步是完成算法一个完整周期所需要的总控制步数。

下面定义一些数学符号:

·$S$:流水调度;

·$S^0$:归一化流水调度;

·$S_i$,$i \in I$:节点 $i$ 的时序调度值,表示把节点 $i$ 安排在控制步 $S_i$ 开始执行,$1 \leqslant S_i \leqslant \mathrm{T_{MAX}}$,规定 $S_{\mathrm{INPUT}} = 1$;

·$\mathrm{Path}(i_1 \xrightarrow{d_{12}} i_2 \longrightarrow \cdots \xrightarrow{d_{(k-1)k}} i_k)$ 数据流图中从节点 $i$ 至节点 $i_k$ 的计算路径,关于计算路径定义两个参数:

· 长度:$\mathrm{LENGTH} = \sum_{i \in Path} \mathrm{EXE}_i$,为计算路径上所有节点的延时之和;

· 阶数:$\mathrm{ORDER} = \sum_{(i \xrightarrow{d_{ij}} j) \in \mathrm{Path}} d_{ij}$,为计算路径中所有弧的延迟器个数之和;

·$\mathrm{Loop}(i_1 \xrightarrow{d_{12}} i_2 \longrightarrow \cdots \xrightarrow{d_{(k-1)k}} i_k \xrightarrow{d_{k1}} i_1)$ 数据流图中的一条计算环路,是闭合的计算路径。

**定义 2** (节点的阶数):在数据流图中,输入节点 INPUT 至输出节点 OUTPUT 且包含节点 $i$ 的计算路径可能不止一条,节点 $i$ 的阶数定义为这些计算路径阶数的最小值。

**定义 3** (节点的刷新计算路径):在数据流图中,计算路径 Path(INPUT→⋯→i→⋯→OUTPUT)的阶数如果等于节点 $i$ 的阶数,那么这条计算路径是节点 $i$ 的刷新计算路径。节点 $i$ 的刷新计算路径的前半段 Path(INPUT→⋯→$i$)称之为节点 $i$ 的输入刷新计算路径,后半段 Path($i$→⋯→OUTPUT)为节点 $i$ 的输出刷新计算路径。

由定义可以知道:在所有由输入节点 INPUT 至节点 $i$ 的计算路径中,输入刷新计算路径的阶数最小;在所有由节点 $i$ 至输出节点 OUTPUT 的计算路径中,输出刷新计算路径的阶数最小。

**定义 4** (关键计算路径):在数据流图中最长的节点 $i$ 的输入刷新路径,称之为 $i$ 的关键输入计算路径。类似地,最长的节点 $i$ 的输出刷新路径,称之为节点 $i$ 的关键输出计算路径。

## 2 归一化定时数据流图

数据流图的时序调度结果是定时数据流图(SDFG),流水调度的定时数据流图由几个折叠体组成,每个折叠体表示一个流水段。为了研究合法的流水调度和合法的流水调度变换的内部机理,我们把定时数据流图进行改进,得到归一化定时数据流图(NSDFG)。它把数据流图嵌入在 DII 个方格中,每一个方格表示一个控制步,它取消了折叠体,代之以归一化的表达方法。因为流水段的划分仅仅是相对于某个给定的数据流图而言的,NSDFG 是在更一般的意义上表示流水调度,不再局限于数据流图所给出的具体计算结构,为了得到更好的结果,流水调度将会隐含地进行计算结构的优化变换。

NSDFG与流水调度 $S$ 有如下的对应关系：

(1) $\forall i \in I$，节点在NSDFG中的位置 $A_i$ 为：

$$A_i = \begin{cases} S_i \bmod \mathrm{DII}, & \text{如果}(S_i \bmod \mathrm{DII}) \neq 0 \\ \mathrm{DII}, & \text{否则} \end{cases} \tag{1}$$

(2)在NSDFG中节点的长度表示节点的延迟时间 $\mathrm{EXE}_i$。

(3) $\forall (i_1 \xrightarrow{d_{12}} i_2) \in E$，在NSDFG有一条有向弧连接节点 $i_1$ 与 $i_2$。

(4)段间弧是从第DII个方格中的节点至第一方格中的节点。一条这种弧线对应着信号流图中的一个延迟器，它表示流水段之间的数据通信。 $\forall (i_1 \xrightarrow{d_{12}} i_2) \in E$，NSDFG中节点 $i_1$ 至节点 $i_2$ 将包含 $d_{12} + (p_{i2} - p_{i1})$ 个段间弧，$p_{i1}$、$p_{i2}$ 是数据操作 $i_1$ 和 $i_2$ 所处流水段的标号。

(5)NSDFG中引入两种形式节点 松弛节点和紧节点，它们不代表任何实际的数据操作，仅表示数据操作之间的空闲时间或顺序冲突。规定松弛节点的延迟时间为1，紧节点的延迟时间为 $-1$，$\forall (i_1 \xrightarrow{d_{12}} i_2) \in E$，令

$$f_{12} = S_{i_2} - (S_{i_1} + \mathrm{EXE}_{i_1} - d_{12}\mathrm{DII}) \tag{2}$$

如果 $f_{12} \geqslant 1$，那么在 $i_1 \rightarrow i_2$ 的有向弧上加上 $f_{12}$ 个松弛节点。如果 $f_{12} \leqslant -1$，那么在 $i_1 \rightarrow i_2$ 有向弧上加上 $|f_{12}|$ 个紧节点。

可见，NSDFG是一种非赋权的有向图，每一条弧经过任何控制步都有节点。容易得出下面的结论：

定理1　流水调度 $S$ 是合法的，当且仅当它的归一化定时数据流图中不含紧节点。

举例说明，图1(a)和图1(b)分别是二阶IIR滤波器的信号流图和数据流图，图1(c)是的二阶IIR滤波器的一种流水调度，重画成NSDFG的形式，如图1(d)所示。图中包含三个松弛节点，分别在 $^*4 \rightarrow +6$、$+7 \rightarrow {}^*2$ 和 $+7 \rightarrow {}^*4$ 的弧上。

## 3　归一化

先给出数据流图中阶数(Order)以及NSDFG中的级数(Stage)的概念。

定义5　(计算路径的级数)：在数据流图中的计算路径一一对应着NSDFG的计算路径，计算路径包含的段间弧的个数称为计算路径在归一化定时数据流图的级数。

计算路径在数据流图中的阶数不一定等于它在NSDFG中的级数，流水调度中隐含的重定时变换与流水变换造成了两者的差异。所有刷新计算路径在NSDFG中的级数与节点的阶数之差都相等，它等于流水变换在数据流图插入或移走的延迟器的个数。

定义6　(节点的级数)：NSDFG中，节点 $i$ 的级数定义为输入刷新计算路径的级数。

NSDFG中规定输入节点INPUT、输出节点OUTPUT的延迟为0，松弛节点的延迟为1，紧节点的延迟为 $-1$，由此得出以下结论：

定理2　在NSDFG中，如果相同起点、相同终点的计算路径的级数相等，那么这些计算路径的长度也相等。

设计算路径的起点为 $i_1$，终点为 $i_2$，级数用stage表示，那么计算路径的长度：

$$\mathrm{LENGTH}(i_1, i_2, \mathrm{stage}) = A_{i_2} - A_{i_1} + \mathrm{Stage} \cdot \mathrm{DII} + 1 \tag{3}$$

NSDFG中节点的刷新计算路径、输入刷新计算路径、输出刷新计算路径往往不止一条，下面定理是定理2的特殊情况。

定理3　在NSDFG中，节点 $i$ 所有的刷新计算路径的长度都相等；节点 $i$ 所有的输入刷新计算路径的长度都相等；节点 $i$ 所有的输出刷新计算路径的长度都相等。

定义7　(归一化)：如果 $S$ 是数据流图的流水调度，那么节点 $i \in I$ 的归一化调度 $S_i^0$ 定义为归一化定时数据流图中节点 $i$ 输入刷新计算路径的长度。

$$S_i^0 = A_i + \mathrm{Stage}_i \cdot \mathrm{DII} \tag{4}$$

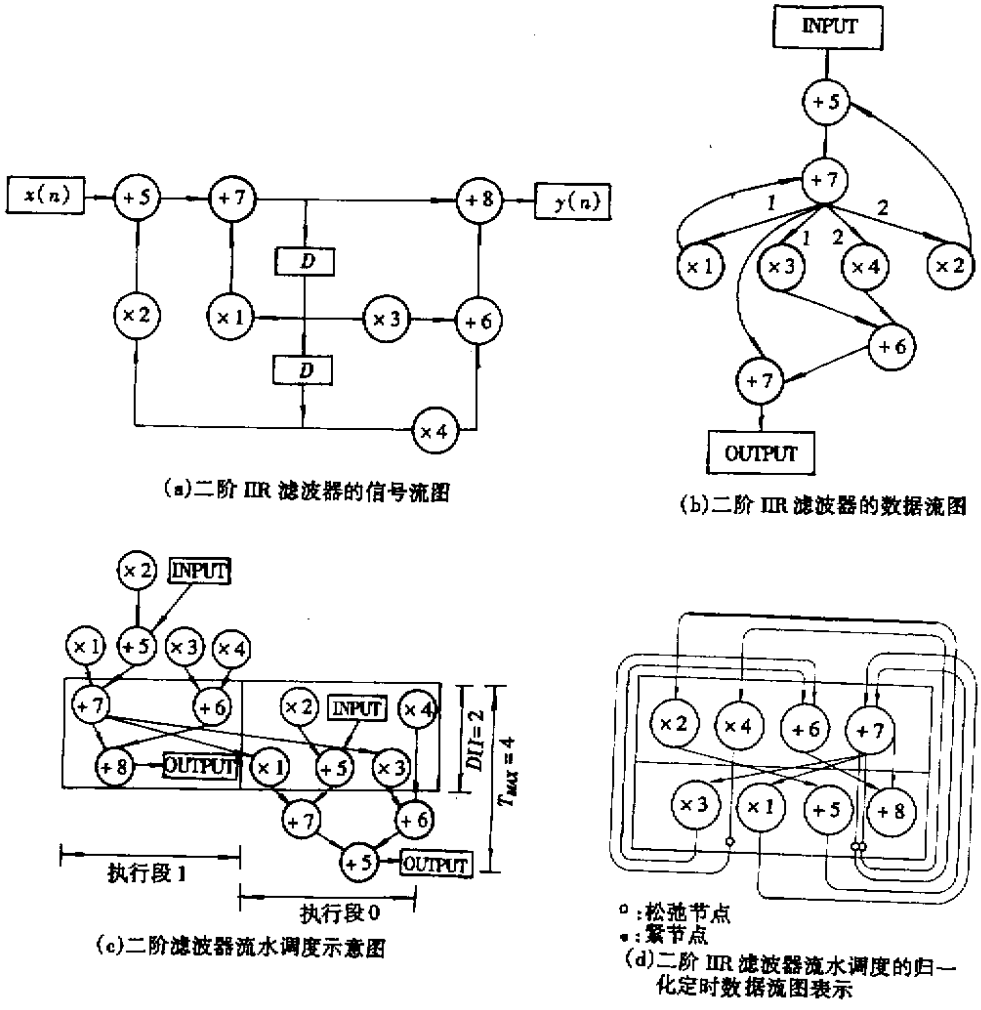


**图 1　二阶 IIR 滤波器 NSDFG 示例**

Fig.1　The NSDFG example for 2-order IIR filter

式中 $A_i$ 的含义见式(1),$\mathrm{Stage}_i$ 为节点 $i$ 在 NSDFG 中的级数。

调度的归一化有着明显的物理意义 视数据流图为一个计算网络，原始数据在输入节点 INPUT 处进入网络,沿计算路径被各节点所处理,结果在输出节点处输出。时序调度是此计算网络在时间维上的映射,节点 $i$ 的归一化调度值 $S_i^0$ 就是输入数据从输入节点 INPUT 传递至节点 $i$ 的最短时间,它等于 NSDFG 中节点 $i$ 的输入刷新计算路径的长度。

在合法的流水调度中,节点 $i$ 的归一化调度值 $S_i^0$ 有上限和下限，记上限为 $\mathrm{ALAP}_i$,下限为 $\mathrm{ASAP}_i$,它们分别决定于数据流图中节点 $i$ 的关键输出计算路径和关键输入计算路径的长度。

## 4　流水调度的变换集

在 NSDFG 上定义一组变换 $\{M_1, M_2, M_3, M_4\}$。关于归一化流水调度 $S_i^0$,记其变换后的调度为 $S_i^{0'}$ :

$M_1$ :上移变换,从节点 $i$ 的所有输入边移走 1 个松弛节点,在它所有的输出边加上 1 个松弛节点,相应地,节点 $i$ 向上移动,$S_i^{0'} = S_i^0 - 1$。

$M_2$ :下移变换,是 $M_1$ 的逆变换,从节点 $i$ 所有的输出边移走 1 个松弛节点,在它所有的输入边加上 1 个松弛节点,相应地,节点 $i$ 向下移动,$S_i^{0'} = S_i^0 + 1$。

$M_3$ :复合上移变换，围绕包含节点的割集进行,从割集的所有输入边移去 1 个松弛节点,而在割集

的所有输出边加上1个松弛节点。相应地,割集中所有的节点向上移动一个控制步,$S_i^{0'} = S_i^0 - 1, \forall i \in$ 割集。

$M_4$:复合下移变换,围绕包含节点的割集进行,从割集所有的输出边移去一个松弛节点,而在割集所有的输入边加上一个松弛节点,相应地,在归一化定时数据流图中割集中所有的节点下移一个控制步,$S_i^{0'} = S_i^0 + 1, \forall i \in$ 割集。

当割集仅包含一个节点时,$M_3$、$M_4$ 就分别退化为 $M_1$、$M_2$ 变换。变换中移去松弛节点相当于加上紧节点,加上松弛节点相当于移去紧节点,松弛节点与紧节点相遇则相互抵消。图2是 $M_1$、$M_2$ 变换的示例。

定理4 $M_1$ 是合法的,当且仅当被移动节点所有的输入边都有松弛节点;

$M_2$ 是合法的,当且仅当被移动节点所有的输出边都有松弛节点;

$M_3$ 是合法的,当且仅当割集所有的输入边都有松弛节点;

$M_4$ 是合法的,当且仅当割集所有的输出边都有松弛节点。

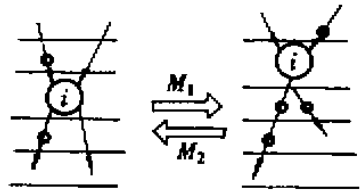


图2 $M_1$、$M_2$ 变换示意图

Fig.2 $M_1$、$M_2$ transformation

如果不特别说明,下面的流水调度(用 $U$、$V$ 表示)都已归一化,关于流水调度 $U$、$V$,节点 $i$ 在 $U$ 和 $V$ 中的差为 $d_i(U,V) = U_i - V_i$,定义流水调度 $U$、$V$ 的距离为:

$$D(U,V) = \sum_{i \in I} | d_i(U,V) | = \sum_{i \in I} | U_i - V_i | \qquad (5)$$

先不加证明地给出两个引理,然后我们再着手证明变换集的完备性。

引理1 两个合法流水调度 $U$、$V$,$D(U,V) > 0$,假设 $U_i > V_i$,节点 $i$ 的立即先继节点记为 $\mathrm{Pre}(i)$,立即后继节点记为 $\mathrm{Suc}(i)$,那么在流水调度 $U$ 中,只能出现两种情况:1)节点 $i$ 的输入边有松弛节点;或者2) $U_{\mathrm{Pre}(i)} > V_{\mathrm{Pre}(i)}$。在流水调度 $V$ 中,也只能出现两种情况:1)节点 $i$ 的输出边有松弛节点;或者2) $U_{\mathrm{Suc}(i)} > V_{\mathrm{Suc}(i)}$。

引理2:合法流水调度 $U$、$V$,$D(U,V) > 0$,假设 $U_{\mathrm{in}} > V_{\mathrm{in}}$,又且计算路径 $\mathrm{Path}(i_1 \rightarrow i_2 \rightarrow \cdots \rightarrow i_n)$ 在 $U$ 中没有松弛节点,那么一定有:$U_{i1} > V_{i1}, U_{i2} > V_{i2}, \cdots, U_{\mathrm{in}} > V_{\mathrm{in}}$。

定理5 变换集 $\{M_1, M_2, M_3, M_4\}$ 是完备的。

详细证明略。

设 $U$,$V$ 为任意两个不同的合法流水调度,$D(U,V) > 0$,假设 $U_i > V_i$,应用引理1和引理2可以证明,虽然单独的 $M_1$ 变换可能是非法的,将导致紧节点的出现,但是总能找到一系列 $M_1$ 变换使紧节点消失,而且每一次 $M_1$ 变换都使 $D(U,V)$ 减1,这些系列 $M_1$ 变换构成了一个合法的复合节点上移 $M_3$ 变换。因此,可以不断地针对 $U_i$ 实施 $M_1$ 或 $M_3$ 变换,直至 $d_i(U,V) = 0$。同理,虽然单独的 $M_2$ 变换可能是非法的,但是可以与其他 $M_2$ 变换构成一个合法的复合节点下移 $M_4$ 变换,其中每一个 $M_2$ 变换都使 $D(U,V)$ 单调减1。因此,可以不断地针对 $V_i$ 实施 $M_2$ 或 $M_4$ 变换,直至 $d_i(U,V) = 0$。

所以,任意两个合法流水调度 $U$、$V$,$D(U,V) > 0$,总能找到一系列 $M_1$、$M_2$、$M_3$、$M_4$ 变换,使 $D(U', V) = 0$。这些变换的个数不会超过 $D(U,V)$。因为每一次 $M_1$ 或 $M_2$ 变换使 $D(U,V)$ 减1,而一次 $M_3$ 或 $M_4$ 变换却使 $D(U,V)$ 减幅大于1,它等于 $M_2$、$M_4$ 变换中割集所包含的节点数。

因此变换集 $\{M_1, M_2, M_3, M_4\}$ 是完备的。

变换集的完备性说明,从任何一个合法的初始流水调度出发,通过一系列 $M_1$、$M_2$、$M_3$、$M_4$ 变换可以搜索到任何一个合法的流水调度。

参考定理5的证明同样可以证明下述定理的正确性。

定理6 任意合法流水调度 $U$,

(1)如果 $U_i > \mathrm{ASAP}_i$,那么总能找到一系列 $M_1$、$M_3$ 变换,使 $U$ 变换至 $U'$,使 $U_i' = \mathrm{ASAP}_i$。

(2)如果 $U_i < \mathrm{ALAP}_i$,那么总能找到一系列 $M_2$、$M_4$ 变换,使 $U$ 变换至 $U'$,使 $U_i' = \mathrm{ALAP}_i$。

如果在数据流图中,存在闭合计算路径 $\mathrm{Loop}(i_1 \xrightarrow{d_{12}} i_2 \longrightarrow \cdots \xrightarrow{d_{(k-1)k}} i_k \xrightarrow{d_{k1}} i_1)$,它的长度恰好等于

阶数与 DII 之积,

$$\sum_{i\in Loop} \mathrm{EXE}_i = \sum_{(i\xrightarrow{d_{ij}}j)\in Loop} d_{ij} \times \mathrm{DII}$$

我们称这种计算环路为临界计算环路。当流水调度把数据流图映射为 NSDFG 时,在临界计算环路上将不包含任何松弛节点,因此不可能对临界计算环路上的节点实施独立而合法的 $M_1$ 或 $M_2$ 变换,只可能采用 $M_3$ 和 $M_4$ 变换,而且割集中至少包含临界计算环路上所有的节点。反之,如果不存在临界计算环路,$M_3$ 和 $M_4$ 都可以分解为一个合法的 $M_1$ 或 $M_2$ 变换序列。

定理 7 如果归一化定时数据流图不含有临界计算环路,那么 $\{M_1, M_2\}$ 是完备合法的变换集。

## 5 设计实例

流水时序调度是设计专用数据通道中一个困难而急待解决的问题,利用基于调度归一化和松、紧节点等概念的归一化定时数据流图,可以证明从任一合法的初始流水调度出发,通过合法的节点移动可以搜索到设计空间中任何一个合法的流水调度。在构造了初始流水调度和得到合法、完备的变换集之后,我们可以用模拟退火算法、遗传算法等来求得流水调度的最优解或近似最优解。

应用以上理论和实现思路,我们设计了一种定向搜索算法,用以求解专用数据通道实现一些数字信号处理算法的最优流水调度,并取得了一些实验结果。这里仅给出一个设计实例,16 点 FIR 滤波器是一个不包含任何计算环路的例子,多篇文献都把它作为一个标准的测试设计,它的信号流图如图 3 所示,其中包括 15 个加法操作和 8 个乘法操作。

假设一个加法操作可以在一个控制步中完成,一个乘法操作需要两个控制步来完成 表 1 定向搜索算法的实验结果,对于不同的 DII 值,定向搜索算法的结果也无一例外地达到了运算单元的下界。关于 16 点 FIR 滤波器,许多文献采用了与我们不同的假设 :加法器完成一个加法操作的时间为半个控制步,乘法器完成一个乘法操作的时间为一个控制步。表 2 是在 DII = 3 时,Sehwa[4]、FDLS[5]、模拟进化算法[6] 和 Theda. Fold 算法[3] 的结果。相对应地,比较表 1 中 DII = 6 时的调度(带下划线的部分),这时硬件代价仅为 3 个乘法器、3 个加法器、22 个寄存器和 12 条数据总线,明显地要优于表 2 中所有算法的调度结果。

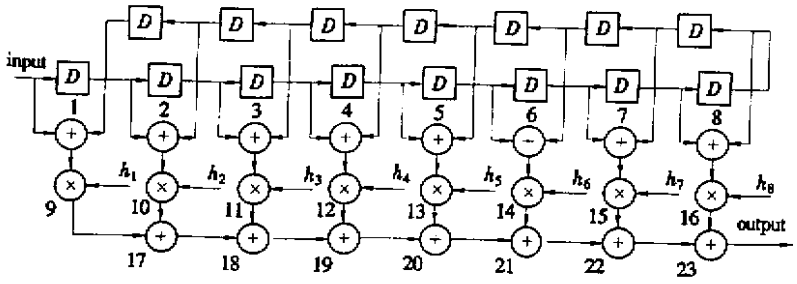


图 3 16 点 FIR 滤波器信号流图

Fig.3 the 16-tap FIR filter signal flow graph

表1 采用非流水乘法器时16点FIR滤波器流水调度的结果

Tab.1 The algorithm results for 16-top FIR filter using no-pipeline multipliers

| *DII* | 乘法器下限 | 加法器下限 | 乘法器 | 加法器 | 寄存器 | 数据总线 | 代价函数值 |
|---|---|---|---|---|---|---|---|
| 1 | 16 | 15 | 16 | 15 | 56 | 62 | 197 |
| 2 | 8 | 8 | 8 | 8 | 35 | 32 | 107 |
| 3 | 6 | 5 | 6 | 5 | 29 | 22 | 80 |
| 4 | 4 | 4 | 4 | 4 | 27 | 16 | 63 |
| 5 | 4 | 3 | 4 | 3 | 24 | 14 | 57 |
| 6 | 3 | 3 | 3 | 3 | 22 | 12 | 49 |
| 7 | 3 | 3 | 3 | 3 | 20 | 10 | 45 |
| 8 | 2 | 2 | 2 | 2 | 21 | 8 | 39 |
| 9 | 2 | 2 | 2 | 2 | 20 | 8 | 38 |
| 10 | 2 | 2 | 2 | 2 | 20 | 8 | 38 |
| 11 | 2 | 2 | 2 | 2 | 20 | 6 | 36 |
| 12 | 2 | 2 | 2 | 2 | 19 | 6 | 35 |
| 13 | 2 | 2 | 2 | 2 | 18 | 6 | 34 |
| 14 | 2 | 2 | 2 | 2 | 18 | 6 | 34 |
| 15 | 2 | 1 | 2 | 1 | 19 | 6 | 34 |
| 16 | 1 | 1 | 1 | 1 | 21 | 4 | 30 |
| 17 | 1 | 1 | 1 | 1 | 21 | 4 | 30 |
| 18 | 1 | 1 | 1 | 1 | 21 | 4 | 30 |
| 19 | 1 | 1 | 1 | 1 | 21 | 4 | 30 |

表2 关于16点FIR滤波器其他算法的调度结果

Tab.2 The other algorithms results for 16-tap FIR filter

| 算法 | DII | 乘法器 | 加法器 | 寄存器 | 数据总线 |
|---|---|---|---|---|---|
| Sehwa | 3 | 6 | 3 | 24 | 20 |
| FDLS | 3 | 6 | 3 | —— | —— |
| **模拟进化算法** | 3 | 5 | 3 | 23 | 20 |
| Theda.Fold | 3 | 5 | 3 | —— | —— |

## 参考文献：


[1] Haroun B S, Elmasry M I. Architecture synthesis for DSP silicon compliers[J]. IEEE TCAD,1989,8(4).

[2] Potkonjak M, Rabaey J. Optimizing resource utilization using transformations[J]. IEEE TCAD,1994,13(3).

[3] Lee T, Wu A C. A transformation_ based method for loop folding[J].IEEE TCAD, 1994,13(4).

[4] Park N, Parker A C. Sehwa :a software package for synthesis of pipelines from behavioral specifications[J].IEEE TCAD,1988,7(3).

[5] Paulin P G, Knight J P. Force-directed scheduling for the behavioral synthesis of ASICs[J]. IEEE TCAD,1989,8(6).

[6] Ly T A, Mowchenko J T. applying simulated evolution to high level synthesis[J].IEEE TCAD, 1993,12(3).