Delay Management for Programmable Video Signal Processors

M.L.G. Smeets E.H.L. Aarts G. Essink E.A. de Kock

Philips Research, Prof. Holstlaan 4, 5656 AA Eindhoven, The Netherlands

Abstract

We consider the problem of memory allocation for intermediate data in the mapping of video algorithms onto programmable video signal processors. The corresponding delay management problem is proved to be NP-hard. We present a solution strategy that decomposes the delay management problem into a delay minimization problem followed by a delay assignment problem. The delay minimization problem is solved with network flow techniques. The delay assignment problem is handled by a constructive approach. The performance of the combined approach is analyzed by means of a benchmark set of industrially relevant video algorithms.

Key words. *Real-time video signal processing; combinatorial optimization; retiming; life-time analysis of variables; network flow; stream processing.*

1 Introduction

At Philips Research programmable video signal processors (VSPs) have been developed for real-time processing of digital video signals [Veendrick et al., 1994]. The programming of a VSP system is done by mapping a specification of a video algorithm given by a multirate signal flow graph (SFG) onto a network of VSPs. The mapping of an SFG onto a VSP network can be viewed as a feasibility problem in which operations must be assigned to processing elements (PEs) and execution intervals such that a set of timing, processor, and communication constraints is met [Essink et al., 1991a].

We aim at fully automatic mapping with real-time execution imposed by static scheduling. The mapping problem is NP-hard and cannot be solved in its entirety [Van Dongen, 1990]. We have adopted a decomposition into the following three subproblems: delay management, partitioning, and scheduling. In this paper we concentrate on the delay management problem. Detailed discussions of the other problems are given by De Kock et al. (1995) for partitioning, and Essink et al. (1991b) for scheduling.

Delay management refers to the problem of allocating memory resources for the storage of intermediate data. The life-time of intermediate data, i.e., the time between production and consumption, is related to the time assignment of the operations. As a consequence, the time assignment determines the storage requirement for intermediate data. The available storage capacity is divided into several memories and memory types. In the delay management step, one determines the type and the amount of memory for the storage of the intermediate data by computing a preliminary time assignment.

The delay management problem resembles problems that arise in the field of life-time analysis of variables [Denk & Parhi, 1994], and pipelined IC design [Hu, Bass & Harber, 1994]. It is closely related to register allocation problems. The problem we discuss differs from those presented in the literature by the fact that we have to deal with a fixed architecture that contains multiple types of memories that can be used to store intermediate data. So, the delay management problem is in fact a feasibility problem rather than an optimization problem. Furthermore, the delay management problem contains an additional type assignment problem.

We present a solution approach to the delay management problem that relaxes certain constraints and decomposes it into two subproblems. The first subproblem called the *delay minimization problem* is solved efficiently using network flow techniques. The second subproblem called the *delay assignment problem* is handled using constructive heuristics.

The organization of this paper is as follows. In Section 2 we present the concepts and a mathematical model of the delay management problem. In Section 3 we prove that the delay management problem is NP-hard. In Section 4 we present our decomposition strategy and discuss how the delay minimization and delay assignment problem are handled. In Section 5 we present preliminary results. In Section 6 we conclude with some final remarks.

2 Delay Management

A detailed discussion of VSP chips and the mapping of video algorithms onto systems of VSPs is presented by Vissers et al. (1995), and the reader is referred to this work for

ED&TC '97 on CD-ROM

the details. Here we restrict ourselves to a brief summary of VSP concepts that are relevant for the description of delay management.

2.1 Architecture

In Figure 1 the VSP architecture is depicted graphically. A VSP contains a number of pipelined processing elements (PEs): ALEs, MEs, BEs, and OEs. ALEs execute arithmetic and logic operations, MEs contain a random access memory on which they execute read and write operations. BEs execute buffer operations, and OEs execute communication operations. All PEs are fully interconnected by means of a switch matrix. Circular buffers called silos are positioned between the outputs of the switch matrix and the inputs of the PEs.

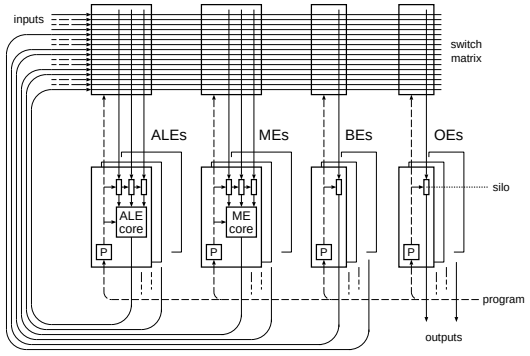

Figure 1: *VSP Architecture.*

2.2 Delay Elements.

Typical of signal processing is the production and consumption of intermediate streams of data samples. These intermediate data samples are stored temporarily in delay elements. VSPs contain two types of programmable delay elements that we call silos and compact silos.

Silos contain a random access memory of 32 words and additional address calculation logic. This logic generates the write address and cyclically increments it in each clock cycle. The read address is generated by the program of the corresponding PE. As a result, data samples can be delayed 1 up to 31 clock cycles.

Compact silos are a feature of the MEs. Each ME contains additional address generation logic in order to use part of the available random access memory in a way that is similar to the functionality of silos. One can implement multiple FIFOs, each having its own length and throughput rate, in a single compact silo. The length of a compact silo is equal to the smallest power of two that is greater or equal to the sum of the lengths of the FIFOs. The additional logic generates the addresses to implement the FIFOs. For more information about compact silos the reader is referred to Dijkstra et al. (1989).

Many existing video signal processing architectures make use of random access memory or FIFOs [Lee & Bier, 1990] to store intermediate data. Silos and compact silos are more flexible than FIFOs and have an advantage over the use of random access memories by the fact that memory addresses do not have to be specified by the programmer.

2.3 Signal Flow Graphs

SFGs consists of operations and data precedences. In this paper an SFG is denoted by a pair (O,R) where O denotes a set of operations and R a set of data precedences. Each operation has a period and an execution time. The period indicates how often the operation is executed, e.g., a period of 4 denotes that an operation needs to be executed exactly every 4 clock cycles. This corresponds with a rate of 13.5 MHz for a clock frequency of 54 MHz. The period of an operation o is given by $p(o) \in \mathbb{Z}^+$. The execution time of an operation o is given by $e(o) \in \mathbb{Z}^+$.

Data precedences are represented by 3-tuples $r = (o,o',(p,b,b'))$ with $o,o' \in O$, $p \in \mathbb{Z}^+$, and $b,b' \in \mathbb{Z}$ such that p is a multiple of $p(o)$ and of $p(o')$. A data precedence r specifies that the data sample generated in the $(kp/p(o)+b)$th execution of operation o is consumed in the $(kp/p(o')+b')$th execution of operation o', for all integers k.

In the mapping trajectory, each execution of each operation in an SFG must be assigned to a time. The time assignments are restricted by requiring that each execution of an operation o is processed on the same processing element. As a result of this assumption and the assumption of strict periodicity, the assignments only have to be found for the first execution of each operation. For reasons of convenience we use completion times to denote the time assignment of an operation. This is represented by the function $\sigma : O \rightarrow \mathbb{Z}$.

To model the use of compact silos, we associate a delay with each operation. This delay indicates how many samples the output of an operation is delayed in a compact silo. Formally, the delay assignment is represented by the function $\tau : O \rightarrow \mathbb{N}$. Consequently, if $\tau(o) = 0$ then operation o must be executed on an ALE, BE or OE. If $\tau(o) > 0$ then operation o must be executed on an ME.

2.4 Problem Statement

We consider the number of clock cycles that the data samples of data precedence r reside in a silo. Let $\rho(r) \in \mathbb{Z}$ indicate the first time a data sample of r is read from the silo, and let $\omega(r) \in \mathbb{Z}$ indicate the first time a data sample is written into the silo. The difference $\rho(r) - \omega(r)$ is the delay

$d(r)$, and is the time between consumption and production of the data samples of r.

Definition 1 (Delay). The delay $d : R \to \mathbb{Z}$ is defined as $d(r) = \rho(r) - \omega(r)$, where $\rho(r) = \sigma(o') + b'p(o') - e(o')$ and $\omega(r) = \sigma(o) + (b + \tau(o))p(o)$. □

Definition 2 (Feasible time assignment). A time assignment $\sigma : O \to \mathbb{Z}$ is called feasible if and only if it satisfies $0 < d(r) < 32$ for all $r \in R$. □

An example of an infeasible time assignment is given in Figure 2. Here, the data samples of r cannot be delayed

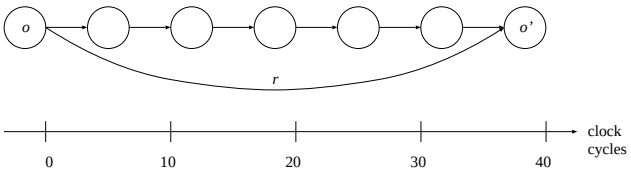

Figure 2: *Example of an SFG with a time assignment that is infeasible since the difference between the time of consumption and the time of production of the samples of data precedence r exceeds the maximum storage time of one silo.*

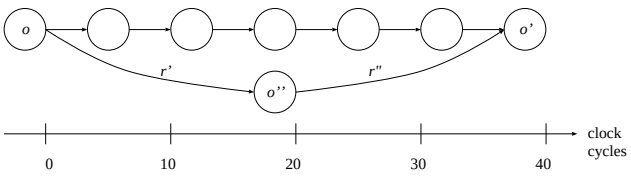

Figure 3: *The example from Figure 2 in which the SFG is extended with an extra operation o'' and data precedences r' and r'' such that the time assignment is feasible.*

sufficiently long since it requires more cycles than one silo can provide. If this is the case, either the time assignment must be changed or the SFG must be extended. Figure 3 shows an extension which is obtained from the SFG of Figure 2 by replacing r with a delay operation o'' and two data precedences r' and r''. For this extended SFG the same time assignment is feasible since the total delay of precedence r is now distributed over two precedence r' and r'' and the two resulting delays each fit into one silo. Moreover, operation o'' can be mapped onto a compact silo which delays its time of production. Several other SFG extension are permitted, but we do not present them here since they are not relevant for the discussion.

The insertion of delay operations does not alter the functionality of an SFG, but may possibly violate capacity constraints since they increase the utilization of the PEs. There are three resources under consideration:

i. The ALE, BE, and OE resources.

ii. The ME access resources.

iii. The ME storage resources.

This is modeled by three resource types p, m, and s, for i, ii, and iii, respectively. Each resource type t has a capacity C_t and requirement R_t. Then the capacity constraints are formulated as

$$R_\mathrm{p} \le C_\mathrm{p} \ \wedge\ R_\mathrm{m} \le C_\mathrm{m} \ \wedge\ R_\mathrm{s} \le C_\mathrm{s}. \tag{1}$$

For a given VSP network the capacities are fixed, i.e., C_p equals the number of PEs of type ALE, BE, and OE, C_m equals the number of MEs, and C_s equals the storage capacity of the MEs. The requirements for all resource types are given by

$$R_\mathrm{p} = \sum_{\substack{o \in O \\ \tau(o) = 0}} \frac{1}{p(o)}, \quad R_\mathrm{m} = \sum_{\substack{o \in O \\ \tau(o) > 0}} \frac{1}{p(o)}, \quad R_\mathrm{s} = \sum_{o \in O} \tau(o)$$

Definition 3 (Feasible delay assignment). A delay assignment is feasible if and only if it satisfies (1). □

The delay management problem now can be stated as follows.

Definition 4 (Delay management problem). Given are a VSP network N and an SFG A. Find an SFG extension A' of A and a feasible delay assignment such that a feasible time assignment exists for all operations in A'. □

3 Complexity

The following theorem presents a result on the complexity of the delay management problem.

Theorem 1. *The delay management problem is NP-hard in the ordinary sense.* □

Proof. (Sketch) The reduction is from bin packing with a fixed number of bins, which is NP-hard [Garey & Johnson, 1979]. We restrict ourselves to a variant of delay management in which only delay operations with period 1 are inserted into the SFG, which is, from a complexity point of view, easier than the delay management problem without this restriction.

The restricted delay management problem is formalized as follows: Given a set U of sets $V_1, \ldots V_n$ consisting of 3-tuples $(\delta_\mathrm{p}, \delta_\mathrm{m}, \delta_\mathrm{s}) \in \mathbb{Q}^3$. Each V_k corresponds to a delay that needs to be implemented. If a delay is implemented using a set of operations O, then

$$\delta_\mathrm{p} = \sum_{\substack{o \in O \\ \tau(o) = 0}} 1, \quad \delta_\mathrm{m} = \sum_{\substack{o \in O \\ \tau(o) > 0}} 1, \quad \delta_\mathrm{s} = \sum_{o \in O} \tau(o).$$

Hence, a 3-tuple $(\delta_\mathrm{p}, \delta_\mathrm{m}, \delta_\mathrm{s})$ corresponds to the additional requirement for the resource types p, m, and s, respectively,

when implementing a delay corresponding with V_k, with δ_p operations with period 1 on ALE, BE, or OE, δ_m operations with period 1 on ME, and δ_s memory requirement. As a consequence, we have

$$\delta_m \leq \delta_s \wedge (\delta_s > 0 \Rightarrow \delta_m > 0), \tag{2}$$

for all 3-tuples $(\delta_p, \delta_m, \delta_s) \in V_k$, where $1 \leq k \leq n$). Furthermore, we are given the initial requirements $I_p, I_m, I_s \in \mathbb{Q}$ of resources p, m, and s respectively, and three constants C_p, C_m, and C_s, which denote the capacities. The question is: does there exist a corresponding set U' that contains exactly one element from each V_k and for which the total requirement does not exceed the capacity of each resource, i.e.,

$$I_k + \sum_{u \in U'} \pi(u,k) \leq C_k, \quad k \in \{p,m,s\}.$$

Here, $\pi : \mathbb{Q}^3 \times \{p,m,s\} \rightarrow \mathbb{Q}$ is a projection operator defined as

$$\pi((a,b,c),p) = a,\ \pi((a,b,c),m) = b,\ \pi((a,b,c),s) = c.$$

A reduction from bin packing with three bins can be made. The bin packing problem is defined as follows: Given are a set of items $I = \{i_1, \ldots, i_n\}$ and sizes $1 \leq s(i_m) < B$, where $s(i_m) \in \mathbb{Z}^+$ and $B \in \mathbb{Z}^+$ is the bin size. Can the items be packed in three bins T_1, T_2 and T_3, such that

$$\sum_{i \in T_k} s(i) \leq B, \quad k \in \{1,2,3\}?$$

Next, we show that every instance of bin packing with three bins can be written as an instance of our restricted delay management problem. To this end we take $I_p = I_m = I_s = 0$, $C_p = B, C_m = B + n$, and $C_s = (n+1) \cdot B$. Furthermore, we take $U = \{V_1, \ldots, V_n\}$, where each V_k is defined as

$$V_k = \{(s(i_k), 1, B),\ (0, 1 + s(i_k), B),\ (0, 1, B + s(i_k))\}.$$

It can be verified that all 3-tuples in any V_k satisfy (2). The 3-tuples correspond with implementations of delays of length $s(i_k) + 1 + B$. It is straightforward to determine the bin to which they are assigned. This completes the proof. □

4 Solution strategy

We have shown that the delay management problem is NP-complete. As a result, no polynomial time algorithm is believed to exist that solves each instance of the problem. Therefore, we use a heuristic approach in which we transform the delay management problem into an optimization problem. The resulting optimization problem is subsequently decomposed into two subproblems called delay minimization and delay assignment. As a result of this decomposition, we restrict the solution space. To present our solution strategy we need the following definitions and results.

Definition 5 (Surplus delay). The surplus delay $d' : R \rightarrow \mathbb{Z}$ is defined as $d'(r) = \max(0, \frac{d(r)-31}{p})$, where $r = (o, o', (p, b, b'))$. □

Hence, the surplus delay represents the storage requirement for intermediate data that does not fit into one silo with a given time assignment.

Theorem 2. *Given is an SFG (O,R). A time assignment $\sigma : O \rightarrow \mathbb{Z}$ is called feasible if and only if it satisfies*

$$\forall_{r \in R}\ 0 < d(r) \ \wedge \sum_{r \in R} d'(r) = 0.$$

□

Proof. By the definition of d'. □

Theorem 3. *Given are an SFG $A = (O,R)$ and a time assignment σ such that $\sum_{r \in R} d'(r) > 0$. Then an SFG extension $A' = (O', R')$ of A can be constructed for which a time assignment $\sigma : O' \rightarrow \mathbb{Z}$ exists such that $\sum_{r \in R'} d'(r) < \sum_{r \in R} d'(r)$.* □

Proof. Let $k = |\{r \in R \mid d'(r) > 0\}|$, let $r' = (o, o', (p, b, b') \in \{r \in R \mid d'(r) > 0\}$, and let $n = \lceil \frac{d'(r') \cdot p}{32} \rceil$. Define $O' = O \cup \bigcup_{1 \leq i \leq n} o_i$ and

$$R' = R \setminus \{r'\} \cup \{(o, o_1, (p, b, 0)), (o_n, o', (p, 0, b'))\} \cup \bigcup_{1 \leq i < n} (o_i, o_{i+1}, (p, 0, 0)).$$

We now define a time assignment σ' for A' as follows. Let $\sigma' = \sigma \cup \bigcup_{1 \leq i \leq n} \sigma'(o_i)$, where $\sigma'(o_i) = \sigma(o) + bp(o) - e(o) + 32i$ for all $0 \leq i \leq n$. Assuming that the executions times of all operations are smaller than 32, it can easily be checked that

$$0 < \sigma'(o_1) - e(o_1) - \sigma'(o) - bp(o) < 32, \tag{3}$$

$$\forall_{1 \leq i < n}\ 0 < \sigma'(o_{i+1}) - e(o_{i+1}) - \sigma'(o_i) < 32, \text{ and} \tag{4}$$

$$0 < \sigma'(o') + b'p(o') - e(o') - \sigma'(o_n) < 32. \tag{5}$$

As a result, for all $r \in R' \setminus R$ holds $d'(r) = 0$. Since the delay of the precedences $r \in R \setminus \{r'\}$ has not changed, we have $|\{r \in R' \mid d'(r) > 0\}| = k - 1$ and that the sum of surplus delay decreases. □

Theorem 3 states that it is always possible to decrease the sum of surplus delays as long as the sum is larger than zero. We can calculate a time assignment σ yielding a minimal sum of surplus delay. Then, on account of Theorem 3, σ can be used to reduce the sum of surplus delay of the SFG. The last mentioned step can be repeated, until the sum

of surplus delays equals zero. The repetition terminates on account of Theorem 2, The combination of these theorems guarantees that only a finite number of operations need to be added for the existence of a feasible time assignment.

This strategy is efficient if the problem of finding a time assignment yielding a minimal sum of surplus delays can be solved in polynomial time. In the next subsection we show that this is indeed the case. The problem of finding such a time assignment is referred to as the *delay minimization problem*. The problem of finding an SFG extension with reduced sum of surplus delays is referred to as the *delay assignment problem*. This decomposition leads to two alternative solution strategies. In the first strategy we try to find a solution to the delay minimization problem and then repeatedly try to find solutions to the delay assignment problem, until all surplus delays are zero. In the second strategy we repeatedly try to find solutions to the delay minimization problem followed by the delay assignment problem until all surplus delays are zero. The second strategy implies more computational work, but the following example shows that the increase in resource requirements as a consequence of delay management can be lower if the second strategy is chosen.

Figure 4 shows an SFG with operations o, o' and o'', among others. The path between o and o'' forces $\sigma(o'') = \sigma(o) + 64$. A time assignment σ that yields minimal sum

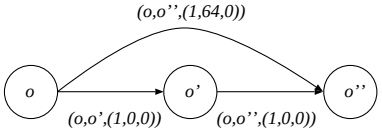

Figure 4: Example of an SFG.

of surplus delay is, for instance, $\sigma(o) = 0$, $\sigma(o') = 32$, and $\sigma(o'') = 64$, yielding a sum of surplus delay of 2. For this time assignment, two delays have to be implemented. One between o and o', and one between o' and o''. If only one delay is implemented, a new time assignment yielding a sum of surplus delay equal to zero can be computed.

On account of the example, the strategy is to repeatedly solve delay minimization and delay assignment problems in an alternating way. If one would choose the alternative strategy and aim at a minimum increase in PE utilization, one would need a piece-wise linear cost function rather than a linear cost function; see Figure 5. In the example two different time assignments can be found. One for which one data precedence has a delay 62, and another for which two precedences have a delay equal to 32. In the first case the sum of the delays larger than 31 is much larger than in the second case, 31 as opposed to 2. But only one additional operation is needed to guarantee the existence of a feasible time assignment, whereas in the second case two additional operations are needed to guarantee the existence of a feasible time assignment. With a piece-wise linear cost function the second time assignment is twice as expensive as the first time assignment. An important disadvantage of a piece-wise linear cost function is that no polynomial time algorithm is known that solves the problem.

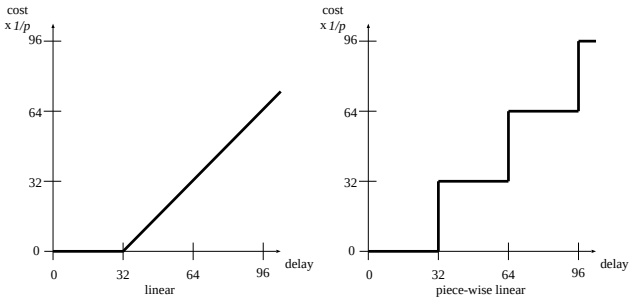

Figure 5: Linear and piece-wise linear cost function for a precedence $r = (o, o', (p, b, b'))$ with a delay $d(r)$.

4.1 Delay Minimization

The goal of the delay minimization is to find a time assignment that minimizes the sum of the surplus delay. In order to minize the storage requirement for intermediate data, we do not store copies of the data. To this end, we classify the consumers into numbered groups.

Definition 6 (Group). The set $G(o,i)$ of precedences of group i of operation o is defined as

$$G(o,i) = \{(o,o',(p,b,b')) \in R \mid b \equiv i \pmod{\frac{p}{p(o)}}\}.$$

□

For each producing operation o, the number of consuming groups is equal to

$$q(o) = \text{lcm}\{\frac{p}{p(o)} \mid (o,o',(p,b,b')) \in R\},$$

by the definition of the precedences and groups.

The delay minimization problem can be formally stated as follows

Definition 7 (Delay minimization problem). Given is an SFG (O,R). Find a time assignment $\sigma : O \to \mathbb{Z}$ such that

$$\sum_{o \in O} \sum_{0 \le i < q(o)} \max\{d'(r) \mid r \in G(o,i)\}$$

is minimal, and $0 < d(r)$ for all $r \in R$. □

Next, we show that this problem can be rewritten as the dual of the minimum cost flow problem, for which there exist efficient solution strategies [Ahuja and Orlin, 1989].

To this end, we introduce a linearization of the cost function by introducing additional operations. For each group an additional operation is introduced. This operation does not need to be mapped onto a PE, but is merely used to compute a preliminary time assignment. The time assignment of an additional operation belonging to operation o and group labeled b is denoted by $\eta_b(o)$. We require that $\eta_b(o)$ is such that $\frac{\eta_b(o)-\sigma(o)}{p(o)\cdot q(o)}$ is the largest amount of surplus delay needed for all data precedences in group b excedent from o, with period p, i.e.,

$$\frac{\eta_b(o)-\sigma(o)}{p(o)\cdot q(o)} = \max\{d'(r) \mid r \in G(o,b)\}.$$

With this definition, a new cost function and constraints can be derived, resulting in a new formulation of the delay minimization problem.

Definition 8 (Delay minimization problem reformulated)). Given is an SFG (O,R). Find a time assignment $\sigma : O \to \mathbb{Z}$ such that

$$\sum_{o\in O}\sum_{0\le b<q(o)} \frac{1}{p(o)\cdot q(o)}\cdot\ (\eta_b(o)-\sigma(o))$$

is minimal, and

$$0 < d(r), \tag{6}$$
$$\sigma(o) \le \eta_b(o), \text{ and} \tag{7}$$
$$\sigma(o) + d(r) - 32 < \eta_b(o), \tag{8}$$

for all $o \in O$, for all $0 \le b < q(o)$, and for all $r = (o,o',(p,b,b') \in R$. □

Note that the cost function can be rewritten to the following expression that must be maximized:

$$\sum_{o\in O}\frac{1}{p(o)}\cdot\sigma(o) + \sum_{o\in O}\sum_{0\le b<q(o)}\frac{-1}{p(o)\cdot q(o)}\cdot\eta_b(o). \tag{9}$$

The dual of the minimum cost flow problem is formally stated as follows.

Definition 9 (Dual of the minimum cost flow problem). Given are $c,u \in \mathbb{R}$, $b \in \mathbb{Z}$, and $\mathcal{A} \in \mathbb{Z}^2$. Find $\pi,\delta \in \mathbb{Z}$ such that

$$\begin{array}{l}\sum_{i\in N} b_i\pi_i - \sum_{(i,j)\in\mathcal{A}} u_{ij}\delta_{ij} \text{ is maximal, subject to:}\\ \delta_{ij} \ge 0\ \wedge\ \pi_i - \pi_j - \delta_{ij} \le c_{ij} \text{ for all } (i,j) \in \mathcal{A}.\end{array} \tag{10}$$

□

Mapping the variables of the dual of the minimum cost flow problem onto variables in the delay minimization problem is a straightforward task. First we construct the variables π and b associated with the nodes of the network flow graph. By looking at the first summation in (9) we would like to conclude $b_o = \frac{1}{p(o)}$. However, b_o must be an integer. To correct this, all b_os are multiplied by a suitable constant ζ. If the additional operations of group i belonging to operation o are denoted by $\bar{o_i}$, we obtain for each $o \in O$ and for each $0 \le i < q(o)$

$$\begin{array}{l} b_o = \frac{\zeta}{p(o)},\\ b_{\bar{o_i}} = \frac{-\zeta}{p(o)\cdot q(o)},\\ \pi_o = \sigma(o),\\ \pi_{\bar{o_i}} = \eta_i(o), \text{ and}\\ \zeta = \text{lcm}\{p(o)\cdot q(o) \mid o \in O\}.\end{array}$$

The variables δ_{ij} and c_{ij} are weights on edges connecting i and j in the network flow problem. To let the second summation in (10) be zero, we demand all δ_{ij} to be zero. Since we already chose δ and π, we may only vary c when mapping the constraints in the delay minimization problem to the constraints of the minimum cost flow problem. This is done by adding edges (i,j) with weights c_{ij} to the network flow graph. For the constraints of (6) we add edges (o,o') with weights $d(r)-\sigma(o')+\sigma(o)-1$ to the network flow graph. For the constraints of (7), we add edges $(o,\bar{o_b})$ with weights 0. Finally, for constraints of (8) we add edges $(o',\bar{o_b})$ with weights $31-d(r)$.

To solve the dual of the minimum cost flow problem we use the algorithm of Ford [1962].

4.2 Delay Assignment

Given the delays obtained from the solution of the delay minimization problem it is possible to identify the precedences with surplus delay. During delay assignment operations are inserted on a group of precedences with surplus delay greater than zero. Selection of the group is done by using a simple heuristic; the group with the largest surplus delay is chosen. For this group additional operations will be inserted in the SFG. This heuristic is chosen to allow the SFG to be extended with operations that have a large delay. The period of the extra operations is determined by the period of the operation o producing the data samples and equals $p(o)\cdot q(o)$. The delay assignment problem can be defined as follows.

Definition 10 (Delay assignment problem). Given are an SFG $A = (O,R)$ and a group of precedences $G(o,i)$, where $o \in O$ and $0 \le i < q(o)$. Find an extension $A' = (O',R')$ of A by inserting a set of delay operations on the data precedences in $G(o,i)$, and a feasible delay assignment $\tau : O' \to \mathbb{N}$, such that a feasible time assignment exists for the operations in O'. □

In general, non-zero surplus delays can be implemented using more than one operation. A non-zero surplus delay

of d cycles that must be implemented using operations with period p, can be implemented by a combination of δ_p operations with a delay of 0, and δ_m operations with a total delay of δ_s if and only if δ_p, δ_m, and δ_s satisfy (2) and

$$1(\delta_p + \delta_m) \leq d - p\delta_s \leq 31(\delta_p + \delta_m). \quad (11)$$

(2) states that using operations with a positive delay requires storage capacity and that each such operation uses at least one word. (11) states that the data samples are stored in a silo for a period of time that is between 1 and 31 clock cycles. Furthermore, we demand that we do not use more resources than required. This can be formally expressed by requireing that $\delta_p, \delta_m,$ and δ_s satisfy

$$\begin{aligned} 31 \cdot \max(0, \delta_p + \delta_m - 1) &< d - p\delta_s, \\ 31(\delta_p + \delta_m) &< d - p \cdot \max(0, \delta_s - 1). \end{aligned} \quad (12)$$

By implementing a non-zero surplus delay of size d by using δ_p operations with period p on ALE, BE, or OE, δ_m operations with period p on ME, and δ_s storage requirement in total, increases R_p with $\frac{\delta_p}{p}$, R_m with $\frac{\delta_m}{p}$, and R_s with δ_s.

We assume that after delay management, the probability of finding a feasible schedule for the resulting SFG is maximal if the PE utilization per type is balanced. If δ_p operations on ALE, BE, or OE, and δ_m operations on ME with δ_s words of memory are inserted, all with period p, then we choose $\delta_p, \delta_m,$ and δ_s such that the *balance index* defined as

$$\left|\frac{I_p + \frac{\delta_p}{p}}{C_p} - \frac{I_m + \frac{\delta_m}{p}}{C_m}\right| + \left|\frac{I_m + \frac{\delta_m}{p}}{C_m} - \frac{I_s + \delta_s}{C_s}\right| + \left|\frac{I_p + \frac{\delta_p}{p}}{C_p} - \frac{I_s + \delta_s}{C_s}\right| \quad (13)$$

is minimal, where $I_p, I_m,$ and I_s denote the initial requirements for type p, m, and s. This choice implies that the difference in utilization degree for all resource types is minimal. The choice for $\delta_p, \delta_m,$ and δ_s when implementing delay for data precedence r is restricted by requiring that it satisfies (2), (11), and (12), with $d = d(r) - 31$.

Within one group with multiple consumers with different delays, we handle the smallest delay first since this delay can be shared among all consumers of the group.

5 Results

The approach presented in Section 4 was implemented in C++ and was tested in the VSP mapping tools. We have used eight industrially relevant video algorithms. The results are presented in Table 1. For each of the tested video algorithms a feasible solution to the delay management problem was found. Furthermore, for all tested video algorithms except 'Vidiwall' a feasible schedule was found after delay management and partitioning.

Table 1: Delay management results for eight video algorithms. (p), (m), and (s): resources affected by delay management. (D) solution to delay management problem found. (M) solution to mapping problem found.

	Requirement before			Requirement after			D	M	Capacities		
Algorithm	p	m	s	p	m	s			p	m	s
Contrast	42.56	3.00	12768	43.50	7.25	12952	Y	Y	144	24	49152
Contour	7.72	0.78	2700	8.22	0.78	2700	Y	Y	24	4	8192
ColorConv	8.50	0.00	0	9.50	0.00	0	Y	Y	24	4	8192
HorCompr	2.50	0.50	2048	2.59	0.50	2048	Y	Y	24	4	8192
Mwtv	8.97	1.75	3424	9.31	1.75	3424	Y	Y	24	4	8192
Vidiwall	9.34	1.47	4696	9.81	1.47	4696	Y	N	24	4	8192
Gamma	6.03	1.00	1264	6.72	1.06	1331	Y	Y	24	4	8192
Panorama	6.53	2.25	6144	7.72	2.25	6144	Y	Y	24	4	8192

In order to indicate the extra resource requirements by delay management, we have compared the requirement before and after delay management. The results of this comparison are also shown in Table 1. Delay management causes only a small increase in resource requirements. As a result, the complexity of subsequent scheduling task hardly increases.

All algorithms in Table 1 could be handled within ten seconds. Although this does not cause any problems, the speed of the program could be further increased by choosing other implementations of the algorithm for solving the dual of the minimum cost flow problem, which is the performance bottleneck.

6 Conclusion

The solution strategy presented in Section 4 extends effectively and efficiently a given SFG in such a way that there exists a feasible time assignment. They also show that the resulting increase of the PE utilization is very limited. The results indicate the proposed decomposition strategy handles the delay management problem effectively and efficiently. The decomposition strategy is very flexible since it can easily handle more types of memories by changing the heuristics of the delay assignment step.

Furthermore, we conclude that we can often successfully complete the scheduling step using only the memory that was allocated in the delay management step. This suggest that the total decomposition of the mapping problem into delay management, partitioning, and scheduling is effective. In each step, necessary conditions for the next step are satisfied using lower bound estimations in order not to restrict the solution space. This is nicely illustrated by the handling of the PE utilization in the delay management problem.

Further research concentrates on improvement of the scheduling techniques and the interaction between the subproblems.

References

AHUJA, R.K., T.L. MAGNATI, AND J.B. ORLIN [1989], Network flows, in: M.J. Todd G.L. Nemhauser, A.H.G. Rinnooy Kan (ed.), *Handbooks in operations research and management science; Volume 1: Optimization*, North Holland, Amsterdam, 211–369.

DENK, T.C., AND K. PARHI [1994], Calculation of minimum number of registers in 2-d discrete wavelet transforms using lapped block processing, *Proc. 1994 International Symposium on Circuits and Systems*, 77–80.

DIJKSTRA, H., H. HOLLMANN, K. HUIZER, AND R. SLUYTER [1989], New programmable delay element, *Electronic Letters* **25 no. 16**, 1019–1021.

DONGEN, R.C.A. VAN [1990], Mapping for digital video signal processors: Models and algorithms, Master's thesis, Department of Mathematics and Computing Science, Eindhoven University of Technology.

ESSINK, G., E. AARTS, R. VAN DONGEN, P. VAN GERWEN, J. KORST, AND K. VISSERS [1991a], Scheduling in programmable video signal processors, *Proceedings of the IEEE International Conference on Computer-Aided Design*, 284–287.

ESSINK, G., E. AARTS, R. VAN DONGEN, P. VAN GERWEN, J. KORST, AND K. VISSERS [1991b], Architecture and programming of a VLIW style video signal processor, *Micro-24*, 181–188.

FORD, L.R., AND D.R. FULKERSON [1962], *Flows in Networks*, 93–130. Princeton University Press, Princeton, New Jersey.

GAREY, M.R., AND D.S. JOHNSON [1979], *Computer and Intractability: A guide to the theory of NP-Completeness*, W.H. Freeman and Company, New York.

HU, X., S.C. BASS, AND R.G. HARBER [1994], Minimizing the number of delay buffers in the synchronization of pipelined systems, *IEEE Trans. on Computer-aided design of Integrated Circuits and Systems* **13**, 1441–1449.

KOCK, E.A. DE, E.H.L. AARTS, G. ESSINK, R.E.J. JANSEN, AND J.H.M. KORST [1995], A variable-depth search algorithm for the recursive bipartitioning of signal flow graphs, *OR Spektrum* **17**, 159–172.

LEE, E.A., AND J.C. BIER [1990], Architectures for statically scheduled dataflow, *Journal of Parallel and Distributed Computing* **10**, 333–348.

VEENDRICK, H.J.M, O. POPP, G. POSTUMA, AND M. LECOUTERE [1994], A 1.5 GIPS video signal processor (VSP), *Proc. CICC 6.2*.

VISSERS, K.A., G. ESSINK, P.H.J. VAN GERWEN, P.J.M. JANSSEN, O. POPP, E. RIDDERSMA, W.J.M. SMITS, AND H.J.M VEENDRICK [1995], Architecture and programming of two generations video signal processors, *Microprocessing and Microprogramming* **41**, 373–390.