| | |
|---|---|
| Title | Assignment-Driven Loop Pipeline Scheduling and Its Application to Data-Path Synthesis |
| Author(s) | YOROZUYA, Toshiyuki; OHASHI, Koji; KANEKO, Mineo |
| Citation | IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences, E85-A(4): 819-826 |
| Issue Date | 2002-04-01 |
| Type | Journal Article |
| Text version | publisher |
| URL | http://hdl.handle.net/10119/4686 |
| Rights | Copyright (C)2002 IEICE. Toshiyuki Yorozuya, Koji Ohashi, Mineo Kaneko, IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences, E85-A(4), 2002, 819-826. http://www.ieice.org/jpn/trans_online/ |
| Description | |

PAPER *Special Section of Selected Papers from the 14th Workshop on Circuits and Systems in Karuizawa*

# Assignment-Driven Loop Pipeline Scheduling and Its Application to Data-Path Synthesis

**Toshiyuki YOROZUYA**[†a)], **Koji OHASHI**[†], ***Nonmembers***, ***and*** **Mineo KANEKO**[†b)], ***Regular Member***

**SUMMARY** In this paper, we study loop pipeline scheduling problem under given resource assignment (operation to functional unit assignments and data to register assignments), which is one of the key tasks in data-path synthesis based on the assignment solution space exploration. We show an approach using a precedence constraint graph with parametric disjunctive arcs generated from the specified assignment information, and derive a scheduling method using branch-and-bound exploration of the parameter space. As an application of the proposed scheduling method, it is incorporated with Simulated-Annealing (SA) based exploration of assignment solution space, and it is demonstrated that data-paths of the fifth-order elliptic wave filter are successfully synthesized.
***key words:*** *data-path synthesis, resource assignment, loop pipeline scheduling, dependence graph, disjunctive arc*

## 1. Introduction

Data-path synthesis is the task to transform an algorithm level description in behavioral domain to a register transfer (RT) level descriptions in structural domain and in behavioral domain [1]. RT level description in structural domain consists of functional units, registers and the other interconnection resources such as nets, buses, and multiplexers.

Most of the conventional data-path synthesis aim mainly to minimize the number of control steps and the number of functional units, and they first decide the schedule and the number of functional units by resource constraint scheduling or time constraint scheduling, which are followed by resource assignments. However, the connectivity between components is also an important metric for VLSIs for its connection with routability, signal transmission delay, power consumption, testability, etc. In the stepwise design; scheduling and resource assignment in this order, it may be hard to make decision on operation schedule with regarding connectivity which shall be fixed only after resource assignment, and some backtracking may be needed to control or to optimize connectivity related metrics. Assignment driven approach is also candidate method to control connectivity [2], [3]. In those design approaches, we often encounter scheduling problems with specified resource assignment [2]–[6].

In this paper, we propose an approach using parametric scheduling graph with disjunctive arcs generated from the specified assignment information (operation to functional unit assignments and data to register assignments) in loop pipeline scheduling problem.

The disjunctive arc approach to scheduling problem is often used in "shop scheduling problems." We can see other disjunctive arc approaches in [3], [4] and [6]. In [4], assignments are specified only for operations and data transfers, and optimum scheduling method is not discussed. In [6], the schedule analyzer transforms register binding into precedence constraints (disjunctive arcs), however disjunctive arcs are introduced only for unambiguous sequentialization of operation and data lifetimes, and the final schedule relies on "off-the-shelf (resource constraints, not binding constraints) scheduler." As the result in their approach, "the existence of a schedule is not strictly guaranteed." The method proposed in [3] does not treat loop pipeline scheduling. Also, they proposed only a simple heuristic algorithm.

In this paper, by contrast, (1) we treat both assignment of operations to functional units and assignment of data to registers, (2) we introduce disjunctive arcs with variable weights to scheduling graph for representing constraints induced by assignment specification, (3) we examine the range of available value for each unknown variable, and construct a branch-and-bound method incorporated with successive refinement of those ranges to solve our problem.

The organization of this paper is as follows. First, the problem treated in this paper and some related matters are described in Sect. 2. Section 3 presents the disjunctive arc approach to the problem and shows a branch-and-bound method to decide loop pipeline scheduling. Section 4 presents application of our proposed scheduling method to data-path synthesis and shows experimental results. Finally, Sect. 5 concludes this paper with a brief summary and suggestions for future work.

Manuscript received June 28, 2001.
Manuscript revised October 5, 2001.
Final manuscript received December 20, 2001.
[†]The authors are with the School of Information Science, Japan Advanced Institute of Science and Technology, Ishikawa-ken, 923–1292 Japan.
a) E-mail: t-yorozu@jaist.ac.jp
b) E-mail: mkaneko@jaist.ac.jp

## 2. Preliminaries

### 2.1 Loop Pipeline Scheduling Problem

The input and output of a scheduling problem under specified resource assignment (SRA) treated in this paper is represented as follows,

**Input;**

- Dependence Graph
  Target application algorithm to be implemented is specified with a directed graph $G = (V_G, A_G)$ which is called "dependence graph." $V_G$ is a union of $V_O$ a set of operations and $V_D$ a set of data. $A_G$ is a union of a set of arcs from operations to data $A_O \subset V_O \times V_D$ (source operation generates destination data) and a set of arcs from data to operations $A_I \subseteq V_D \times V_O$ (source data is used by destination operation as input). Moreover, $A_I$ has a delay function $D : A_I \to Z$. An example of the dependence graph is shown in Fig. 1(left).
  We assume that each operation in a dependence graph is executed repeatedly, where $(o_i, d_j) \in A_O$ represents that $m$th execution of $o_i$ (sometime we denote it as $o_i^{(m)}$) generates $m$th data of $d_j$ (sometime we denote it as $d_j^{(m)}$), and $(d_j, o_k) \in A_I$ represents that $m$th execution of $o_k$ (i.e. $o_k^{(m)}$) uses $(m - D(d_j, o_k))$th data of $d_j$ (i.e. $d_j^{(m-D(d_j,o_k))}$) (Fig. 2).
- Resource Assignment
  We let $\mathcal{F}$ be a set of allocated functional units, and functional unit assignment is a mapping $\rho : V_O \to \mathcal{F}$. Similarly, we let $\mathcal{R}$ be a set of allocated registers, and register assignment is represented as a mapping $\xi : V_D \to \mathcal{R}$.
- Execution time
  Execution time of operation is given by a mapping $e : V_O \to Z+$.

**Output;**

- Scheduling
  Scheduling is a mapping $\sigma : V_O \to Z$ where $\sigma$ denotes the control step of 0th execution of each operation, we assume that every operation is executed repeatedly with a common period $T_r$. That is, the execution of operation $o_i$ starts at $\sigma(o_i) + mT_r$ control steps ($m = \ldots, 0, 1, 2, \ldots$). The following constraints must be satisfied.
  1. Scheduling satisfies the precedence constraints specified by arcs and delay function in $G$.
  2. The lifetimes of operations assigned to the same functional unit do not overlap, and also the lifetimes of data assigned to the same register do not overlap.

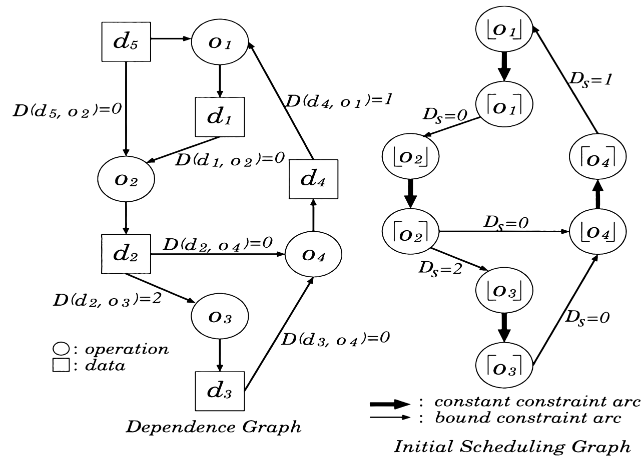


**Fig. 1** Dependence graph and its initial scheduling graph.

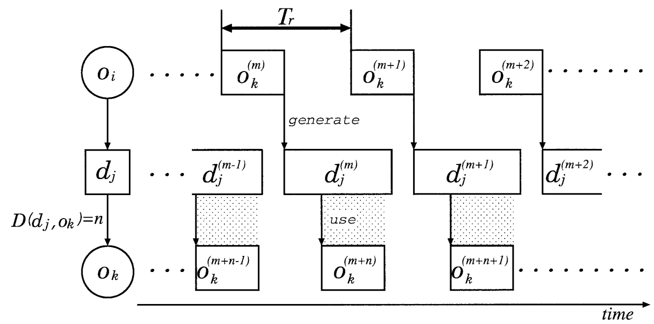


**Fig. 2** Lifetime chart for explaining delay function on an arc in $A_I$.

Now, we can consider two problems, one is the decision problem whether a scheduling exists or not under specified $T_r$, the other is the problem to find an optimum scheduling with minimum $T_r$. It is trivial that SRA (decision version) is in $\mathcal{NP}$, also the "flow shop scheduling problem" can be polynomially reduced to SRA. Hence our SRA problem is in $\mathcal{NP}$-complete.

The blocked scheduling and the chaining are not considered in this paper.

### 2.2 Introduction of Scheduling Graph

When we treat the scheduling problem, we introduce a graph called "scheduling graph" $G_S = (V_S, A_S)$ to represent explicitly the start control step and the end control step of each operation. Here, we let $\lfloor o_i \rfloor$ and $\lceil o_i \rceil$ be nodes corresponding to the start and the end of operation $o_i \in V_O$, respectively, and $V_S = V_{\lfloor\ \rfloor} \cup V_{\lceil\ \rceil}$, where $V_{\lfloor\ \rfloor} = \{\lfloor o_i \rfloor \mid o_i \in V_O\}$ and $V_{\lceil\ \rceil} = \{\lceil o_i \rceil \mid o_i \in V_O\}$. On the other hand, $A_S$ is a union of two sets of different kind of arcs, one is named "constant constraint arcs"($A_C$) and the other is named "bound constraint arcs"($A_\leq$) (the difference between constraint arc and bound constraint arc will be explained in the following paragraph).

For $A_S$, two weight functions $W$ and $D_S$ are specified as follows.

$$W : A_C \cup A_{\leq} \rightarrow N$$

$$D_S : A_{\leq} \rightarrow Z$$

When we consider the scheduling $\sigma : V_S \rightarrow Z$ on $G_S$, it is requested that $\sigma(q) = \sigma(p) + W(p,q)$ for a constant constraint arc $(p,q) \in A_C$ and $\sigma(s) \geq \sigma(r) + W(r,s) - D_S(r,s)T_r$ for a bound constraint arc $(r,s) \in A_{\leq}$.

A scheduling graph $(V_S, A_C \cup A_{\leq})$ is first constructed from the dependence graph of the input instance as Eqs. (1) through (5), which is called an "initial scheduling graph" and is denoted by $G_{S0}$, and afterward it will be modified in our scheduling procedure.

$$A_C = \{(\lfloor o_i \rfloor, \lceil o_i \rceil) \mid o_i \in V_O\} \tag{1}$$
$$W(\lfloor o_i \rfloor, \lceil o_i \rceil) = e(o_i) - 1 \tag{2}$$
$$A_{\leq} = \{(\lceil o_i \rceil, \lfloor o_j \rfloor) \mid \exists d \; s.t. \; o_i = p(d), (d, o_j) \in A_I\} \tag{3}$$
$$W(\lceil o_i \rceil, \lfloor o_j \rfloor) = 1 \tag{4}$$
$$D_S(\lceil p(d) \rceil, \lfloor o_j \rfloor) = D(d, o_j) \tag{5}$$

Note that $p(d)$ is an operation which generates data $d$ (in other words, an immediate predecessor of $d$ in $G$). Fig. 1(right) shows the initial scheduling graph for its left dependence graph.

### 2.3 ASAP and ALAP Scheduling

Sometimes we consider a scheduling with a specified reference node $v \in V_S$, in which $\sigma(v) = 0$ is retained, and we denote it as $\sigma_v : V_S \rightarrow Z$. Moreover, we define ASAP scheduling $\sigma_{ASAPv}$ and ALAP scheduling $\sigma_{ALAPv}$ for a reference node $v$ as follows,

$$\sigma_{ASAPv}(p) = \text{"the longest path length from } v \text{ to } p \text{ on } G_S\text{"}$$

$$\sigma_{ALAPv}(p) = -\text{"the longest path length from } p \text{ to } v \text{ on } G_S\text{"}$$

where we define the arc length as $W(p,q)$ for $(p,q) \in A_C$ and $W(r,s) - D_S(r,s)T_r$ for $(r,s) \in A_{\leq}$. Assuming that no positive cycle is contained in $G_S$, for any choice of reference node $v$, the following inequality can be easily verified.

$$\sigma_{ASAPv}(p) \leq \sigma_v(p) \leq \sigma_{ALAPv}(p) \tag{6}$$

## 3. Disjunctive Arc Approach

### 3.1 Resource Assignment Constraint

If two operations $o_a$ and $o_b$ in $G$ are assigned to the same functional unit, then the collision of those operations should be avoided, and it can be done if and only if there exists an integer $K_{ab}$ such that,

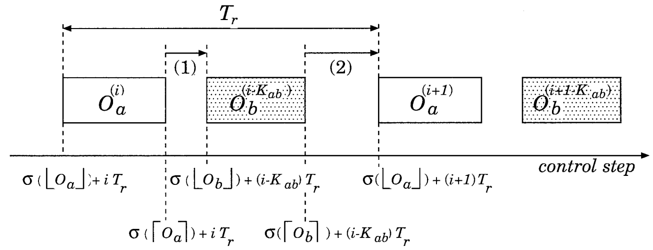


**Fig. 3** Collision-Free scheduling of $o_a$ and $o_b$ which are assigned to the same functional unit.

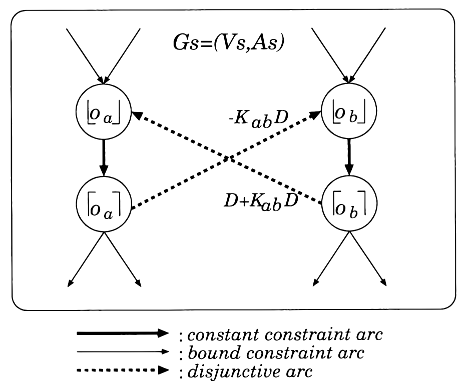


**Fig. 4** Disjunctive arcs induced by functional unit assignment.

(1) The lifetime of $i$th execution of the operation $o_a$ precedes the lifetime of $(i - K_{ab})$th execution of the operation $o_b$ (Fig. 3(1)).
(2) At the same time, the lifetime of $(i - K_{ab})$th execution of the operation $o_b$ precedes the lifetime of $(i+1)$th execution of the operation $o_a$ (Fig. 3(2)).

Since the lifetime of $i$th execution of operation $o_a$ begins at $\sigma(\lfloor o_a \rfloor) + iT_r$ and ends at $\sigma(\lceil o_a \rceil) + iT_r$, we obtain following inequalities from above two constraints.

$$\sigma(\lfloor o_b \rfloor) + (i - K_{ab})T_r > \sigma(\lceil o_a \rceil) + iT_r$$
$$\Rightarrow \sigma(\lfloor o_b \rfloor) > \sigma(\lceil o_a \rceil) + K_{ab}T_r \tag{7}$$

$$\sigma(\lfloor o_a \rfloor) + (i+1)T_r > \sigma(\lceil o_b \rceil) + (i - K_{ab})T_r$$
$$\Rightarrow \sigma(\lfloor o_a \rfloor) > \sigma(\lceil o_b \rceil) - (1 + K_{ab})T_r \tag{8}$$

On the scheduling graph $G_S$, these two constraints can be represented by a pair of bound constraint arcs $(\lceil o_a \rceil, \lfloor o_b \rfloor)$ and $(\lceil o_b \rceil, \lfloor o_a \rfloor)$ (Fig. 4) whose weights are given as follows.

$$D_S(\lceil o_a \rceil, \lfloor o_b \rfloor) = -K_{ab}$$

$$D_S(\lceil o_b \rceil, \lfloor o_a \rfloor) = 1 + K_{ab}$$

$$W(\lceil o_a \rceil, \lfloor o_b \rfloor) = W(\lceil o_b \rceil, \lfloor o_a \rfloor) = 1$$

On the other hand, two data $d_a$ and $d_f$ which are generated by operations $o_a$ and $o_f$, respectively, in $G$ are assigned to the same register, then the collision of those data lifetimes should be avoided, and it can be done if and only if there exists an integer $R_{af}$ such that,

(1) The lifetime of the data $d_a$ generated by $i$th execution of the operation $o_a$ precedes the lifetime of the data $d_f$ generated by $(i - R_{af})$th execution of operation $o_f$.
(2) At the same time, the lifetime of the data $d_f$ generated by $(i - R_{af})$th execution of operation $o_f$ precedes the lifetime of the data $o_a$ generated by $(i+1)$th execution of operation $o_a$.

When operations $o_{b_x}$ $(x = 1, 2, \dots, m)$ use data $d_a$ and $D_S(\lceil o_a \rceil, \lfloor o_{bx} \rfloor) = s_x$, then the lifetime of $i$th data of $d_a$, which is generated by $i$th execution of operation $o_a$ begins at $\sigma(\lceil o_a \rceil) + iT_r + 1$ and ends at $\mathrm{MAX}_x\{\sigma(\lceil o_{b_x} \rceil) + (i + s_x)T_r\}$. Hence, when operation $o_{b_x}$ $(x = 1, 2, \dots, m)$ use data $d_a$ and $D_S(\lceil o_a \rceil, \lfloor o_{bx} \rfloor) = s_x$, and at the same time operation $o_{g_y}$ $(y = 1, 2, \dots, n)$ use data $d_f$ and $D_S(\lceil o_f \rceil, \lfloor o_{g_y} \rfloor) = t_y$, the above two constraints are described as follows.

$$\begin{aligned} &\underset{x}{\mathrm{MAX}}\ [\sigma(\lceil o_{b_x} \rceil) + (i + s_x)T_r] \\ &\quad < \sigma(\lceil o_f \rceil) + (i - R_{af})T_r + 1 \\ &\quad \Rightarrow \sigma(\lceil o_f \rceil) \geq \underset{x}{\mathrm{MAX}}\ [\sigma(\lceil o_{b_x} \rceil) + (R_{af} + s_x)T_r] \end{aligned}$$

$$\begin{aligned} &\underset{y}{\mathrm{MAX}}\ [\sigma(\lceil o_{g_y} \rceil) + (i - R_{af} + t_y)T_r] \\ &\quad < \sigma(\lceil o_a \rceil) + (i+1)T_r + 1 \\ &\quad \Rightarrow \sigma(\lceil o_a \rceil) \geq \underset{y}{\mathrm{MAX}}\ [\sigma(\lceil o_{g_y} \rceil) \\ &\qquad\quad + (t_y - R_{af} - 1)T_r] \end{aligned}$$

Similar to the case of functional unit assignment, those constraints are represented by a set of bound constraint arcs $(\lceil o_{b_x} \rceil,\ \lceil o_f \rceil)$ $(x = 1, 2, \dots, m)$, $(\lceil o_{g_y} \rceil,\ \lceil o_a \rceil)$ $(y = 1, 2, \dots, n)$ in $G_S$ (Fig. 5), and their arc weights are given as follows,

$$D_S(\lceil o_{b_x} \rceil,\ \lceil o_f \rceil) = -D_S(\lceil o_a \rceil,\ \lfloor o_{b_x} \rfloor) - R_{af}$$

$$D_S(\lceil o_{g_y} \rceil,\ \lceil o_a \rceil) = -D_S(\lceil o_f \rceil,\ \lfloor o_{g_y} \rfloor) + 1 + R_{af}$$

$$W(\lceil o_{b_x} \rceil,\ \lceil o_f \rceil) = W(\lceil o_{g_y} \rceil,\ \lceil o_a \rceil) = 0$$

On the other hand, on a functional unit (or register) onto which only a single operation $o_h$ (or data $d_c$) is assigned, lifetime collision between different operations (data) does not occur, but lifetime collision between consecutive executions of the single operation $o_h$ (data $d_c$) may occur. To avoid the latter, we need to add the following bound constraint arc(s);

- with respect to $o_h$, arc $(\lceil o_h \rceil, \lfloor o_h \rfloor)$ with,

$$D_S(\lceil o_h \rceil, \lfloor o_h \rfloor) = 1$$

$$W(\lceil o_h \rceil, \lfloor o_h \rfloor) = 1.$$

- with respect to $d_c$, which is generated by operation $o_c$ and used by $o_{d_z}$ $(z = 1, 2, \dots, n)$ and $D_S(\lceil o_c \rceil, \lfloor o_{d_z} \rfloor) = y_z$ $(z = 1, 2, \dots, n)$, arcs $(\lceil o_{d_z} \rceil, \lceil o_c \rceil)$, $z = 1, 2, \dots, n$, with,

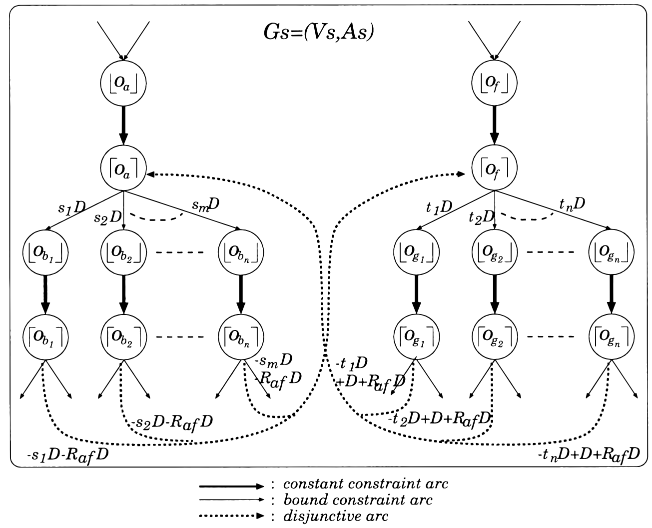


**Fig. 5** Disjunctive arcs induced by register assignment.

$$D_S(\lceil o_{d_z} \rceil,\ \lceil o_c \rceil) = -D_S(\lceil o_c \rceil,\ \lfloor o_{d_z} \rfloor) + 1$$

$$W(\lceil o_{d_z} \rceil,\ \lceil o_c \rceil) = 0.$$

Note that a variable $K_{ab}$ is introduced into $G_S$ (initially $G_S = G_{S0}$) for every pair of operations $o_a$ and $o_b$ which are assigned to the same functional unit, and also a variable $R_{af}$ is introduced into $G_S$ for every pair of data $d_a$ and $d_f$ which are assigned to the same register. In the following, we denote the set of all variables $K_{ab}$s as $\boldsymbol{K}$ and the set of all variables $R_{af}$s as $\boldsymbol{R}$.

Now, our loop pipeline scheduling problem is converted to the problem to find $\Sigma$,

$$\Sigma : \boldsymbol{K} \cup \boldsymbol{R} \rightarrow Z$$

so that the resultant scheduling graph contains no positive cycles.

### 3.2 Range of Unknown Variable

We investigate the range of unknown variables under a given $T_r$. In the following, we assume that the initial scheduling graph $G_{S0}$ is strongly connected.

We consider the unknown variable $K_{ab} \in \boldsymbol{K}$ on the disjunctive arcs derived from operation resource assignment $\rho(o_a) = \rho(o_b)$.

When we consider a scheduling $\sigma_{\lfloor o_v \rfloor}$ with a reference node $\lfloor o_v \rfloor$, inequalities,

$$\begin{aligned} &\frac{-\sigma_{\lfloor o_v \rfloor}(\lfloor o_a \rfloor) + \sigma_{\lfloor o_v \rfloor}(\lceil o_b \rceil)}{T_r} - 1 \\ &\quad < K_{ab} \\ &\quad < \frac{\sigma_{\lfloor o_v \rfloor}(\lfloor o_b \rfloor) - \sigma_{\lfloor o_v \rfloor}(\lceil o_a \rceil)}{T_r} \end{aligned} \tag{9}$$

hold from inequalities (7) and (8). Further, from inequality (6),

$$\frac{-\sigma_{ALAP\lfloor o_v\rfloor}(\lfloor o_a\rfloor)+\sigma_{ASAP\lfloor o_v\rfloor}(\lceil o_b\rceil)}{T_r}-1 \leq \frac{-\sigma_{\lfloor o_v\rfloor}(\lfloor o_a\rfloor)+\sigma_{\lfloor o_v\rfloor}(\lceil o_b\rceil)}{T_r}-1 \tag{10}$$

and,

$$\frac{\sigma_{\lfloor o_v\rfloor}(\lfloor o_b\rfloor)-\sigma_{\lfloor o_v\rfloor}(\lceil o_a\rceil)}{T_r} \leq \frac{\sigma_{ALAP\lfloor o_v\rfloor}(\lfloor o_b\rfloor)-\sigma_{ASAP\lfloor o_v\rfloor}(\lceil o_a\rceil)}{T_r} \tag{11}$$

hold.

Since inequalities (9), (10) and (11) hold for any selection of the reference node, we can obtain,

$$\underset{o_v\in V_O}{\mathrm{MAX}}\left[\frac{\sigma_{ASAP\lfloor o_v\rfloor}(\lceil o_b\rceil)-\sigma_{ALAP\lfloor o_v\rfloor}(\lfloor o_a\rfloor)}{T_r}-1\right] < K_{ab} < \underset{o_v\in V_O}{\mathrm{MIN}}\left[\frac{\sigma_{ALAP\lfloor o_v\rfloor}(\lfloor o_b\rfloor)-\sigma_{ASAP\lfloor o_v\rfloor}(\lceil o_a\rceil)}{T_r}\right] \tag{12}$$

Now we let $L(a,b)$ be the longest path length from node $a$ to node $b$ on scheduling graph $G_S$. Then for any selection of the reference node, we get,

$$\begin{aligned}&\sigma_{ASAP\lfloor o_v\rfloor}(\lceil o_b\rceil)-\sigma_{ALAP\lfloor o_v\rfloor}(\lfloor o_a\rfloor)\\&\quad = L(\lfloor o_a\rfloor,\ \lfloor o_v\rfloor)+L(\lfloor o_v\rfloor,\ \lceil o_b\rceil)\\&\quad \leq L(\lfloor o_a\rfloor,\ \lceil o_b\rceil)=\sigma_{ASAP\lfloor o_a\rfloor}(\lceil o_b\rceil)\end{aligned}$$

$$\begin{aligned}&\sigma_{ALAP\lfloor o_v\rfloor}(\lfloor o_b\rfloor)-\sigma_{ASAP\lfloor o_v\rfloor}(\lceil o_a\rceil)\\&\quad = -(L(\lfloor o_b\rfloor,\ \lfloor o_v\rfloor)+L(\lfloor o_v\rfloor,\ \lceil o_a\rceil))\\&\quad \geq -L(\lfloor o_b\rfloor,\ \lceil o_a\rceil)=-\sigma_{ASAP\lfloor o_b\rfloor}(\lceil o_a\rceil).\end{aligned}$$

Using those inequalities, inequality (12) can be simplified further, and finally the following theorem is obtained.

**Theorem 1:** When we avoid lifetime collision between $o_a$ and $o_b$, $\rho(o_a)=\rho(o_b)$, by adding bound constraint arcs with

$$D_S(\lceil o_a\rceil,\ \lfloor o_b\rfloor)=-K_{ab}$$

$$D_S(\lceil o_b\rceil,\ \lfloor o_a\rfloor)=1+K_{ab},$$

the range of $K_{ab}$ is given, without loss of optimality, as

$$\frac{\sigma_{ASAP\lfloor o_a\rfloor}(\lceil o_b\rceil)}{T_r}-1 < K_{ab} < \frac{-\sigma_{ASAP\lfloor o_b\rfloor}(\lceil o_a\rceil)}{T_r}$$

On the other hand, for the data $d_a$ which is generated by the operation $o_a$ and is used by the operation $o_{b_x}$ and the data $d_f$ which is generated by the operation $o_f$ and is used by the operation $o_{g_y}$, the unknown variable $R_{af}$ on the disjunctive arcs is introduced when $\xi(d_a)=\xi(d_f)$. From the similar discussion with the one for $K_{ab}$, we can have the following,

$$\underset{o_v\in V_O}{\mathrm{MAX}}\ \underset{y}{\mathrm{MAX}}\left[t_y-1+\frac{-\sigma_{ALAP\lfloor o_v\rfloor}(\lceil o_a\rceil)+\sigma_{ASAP\lfloor o_v\rfloor}(\lceil o_{g_y}\rceil)}{T_r}\right] \leq R_{af} \leq \underset{o_v\in V_O}{\mathrm{MIN}}\ \underset{x}{\mathrm{MIN}}\left[\frac{\sigma_{ALAP\lfloor o_v\rfloor}(\lceil o_f\rceil)-\sigma_{ASAP\lfloor o_v\rfloor}(\lceil o_{b_x}\rceil)}{T_r}-s_x\right]$$

and finally following theorem is obtained.

**Theorem 2:** When we avoid lifetime collision between $d_a$ and $d_f$, $\xi(d_a)=\xi(d_f)$, by adding bound constraint arcs with

$$D_S(\lceil o_{b_x}\rceil,\ \lceil o_f\rceil)=-D_S(\lceil o_a\rceil,\ \lfloor o_{b_x}\rfloor)-R_{af}$$
$$D_S(\lceil o_{g_y}\rceil,\ \lceil o_a\rceil)=-D_S(\lceil o_f\rceil,\ \lfloor o_{g_y}\rfloor)+1+R_{af}$$

the range of $R_{af}$ is given, without loss of optimality, as

$$\underset{y}{\mathrm{MAX}}\left[t_y-1+\frac{\sigma_{ASAP\lceil o_a\rceil}(\lceil o_{g_y}\rceil)}{T_r}\right] \leq R_{af} \leq \underset{x}{\mathrm{MIN}}\left[\frac{-\sigma_{ASAP\lceil o_f\rceil}(\lceil o_{b_x}\rceil)}{T_r}-s_x\right]$$

### 3.3 Branch-and-Bound for Exact Solution

We show the scheduling (iteration period constraint scheduling) algorithm based on branch-and-bound exploration of the solution space for $\Sigma$. The outline of the algorithm is described in Fig. 6. An initial solution space for $\Sigma$ is formed from a set of feasible integers (range), which are calculated by using Theorems 1 and 2, for every unknown variables ($\boldsymbol{K}\cup\boldsymbol{R}$). Also these ranges are updated using Theorems 1 and 2 to increase bounding opportunities, whenever a branching is proceeded. Once a feasible $\Sigma:\boldsymbol{K}\cup\boldsymbol{R}\rightarrow Z$ (i.e. the corresponding scheduling graph $G_S$ contains no positive cycles) is found, the scheduling $\sigma$ is obtained by calculating the longest path lengths from a reference node to all nodes on $G_S$.

On the other hand, with respect to the scheduling problem to find an optimum scheduling with minimum $T_r$, we are going to execute SCHEDULING $(G,\rho,\xi,T_r)$ repeatedly with increasing (or decreasing) $T_r$.

## 4. Application to Data-Path Synthesis

### 4.1 SEAS: SA Based Exploration of Assignment Space

As an application of the proposed loop pipeline schedul-

SCHEDULING($G, \rho, \xi, T_r$)
1. Construct initial scheduling graph $G_{S0}$ from $G$.
2. Construct variable list $\boldsymbol{K} \cup \boldsymbol{R}$ from $\rho$ and $\xi$.
3. **if** (BAB($G_{S0}, \boldsymbol{K} \cup \boldsymbol{R}, T_r$) == 1) "SUCCESS"
   **else** "FAIL"

BAB($G_S, \boldsymbol{K} \cup \boldsymbol{R}, T_r$)
1. Calculate the longest path length of every pair of nodes on scheduling graph $G_S$
   (if a positive cycle is detected, **return(0)**).
2. Calculate the range of remained variables in $\boldsymbol{K} \cup \boldsymbol{R}$.
3. **if** (There exists no unknown variables)
   print the longest path length from reference node to all other nodes, and **return(1)**
   **else if** (There exists an unknown variable whose range contains no integer value)
   **return(0)**
   **else if** (There exists unknown variables each of whose range contains exactly one integer value)
   **for**(each unknown variable whose range contains exactly one integer value)
   fix the value of the unknown variable to the integer value contained in its range, and add the corresponding disjunctive arcs to $G_S$.
   back to Step 1.
   **else**
   Select one unknown variable
   **for**(each integer contained in its range)
   3-1. fix the value, add the corresponding disjunctive arcs to $G_S$
   3-2. **if** (BAB($G_S, \boldsymbol{K} \cup \boldsymbol{R}, T_r$) == 1) **return(1)**
   3-3. delete disjunctive arcs added in step 3-1.
   **return(0)**

**Fig. 6** Assignment-driven scheduling.

ing under given assignment, we incorporate our scheduler into data-path synthesis based on assignment space exploration which can respect connectivity between modules explicitly throughout the synthesis process.

Now, we will treat multiplexer-type architecture which consists of adders, non-pipelined multipliers, registers, multiplexers and interconnections between modules (or terminals). We will consider data-path synthesis problem to find the data-path with minimum number of point-to-point interconnections under given set of available modules (functional unit and registers) $\mathcal{F}, \mathcal{R}$, and iteration period $T_r$. For this end, we adopt the resource assignment space exploration using Simulated-Annealing (SA). That is, each solution visited in SA is a complete resource assignment $(\rho, \xi)$, and each solution is evaluated in its scheduling feasibility ("SUCCESS" or "FAIL" by SCHEDULING $(G, \rho, \xi, T_r)$) under given $T_r$ and the number of point-to-point interconnections.

In SA, the accuracy of cost evaluation of each visited solution affects the decision of acceptance/rejection of each visited solution, the reachability from one solution to another, and a chain of accepted solutions from initial solution to a final solution. In the archetype of SA, it is assumed that each solution is evaluated its cost correctly. Also it has been proven, under the correct evaluation of each solution, that SA can produce eventually an optimum solution if the cooling schedule is appropriate. Even if such appropriate cooling schedule is unrealistic and the justification of SA method with finite steps relies only on the expectation that it will simulate SA with an appropriate cooling schedule and will produce a solution not much different from an optimum solution, the incorporation of a correct evaluation of each solution is thought as a basic configuration of SA. Those are the reason why we incorporate the branch-and-bound (exact) scheduling into SA based exploration of assignment space.

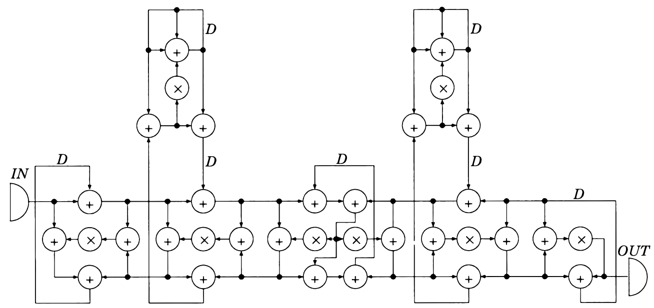


**Fig. 7** Fifth-order elliptic wave filter.

From our design experiments, it is shown that our branch-and-bound scheduling works acceptably fast enough to be executed repeatedly in SA for problem instances with the size of fifth–order elliptic wave filter. A heuristic version of assignment–driven scheduling and its incorporation into SA based exploration of assignment solution space for larger problem instances are left for a future problem.

## 4.2 Synthesis Examples

Fifth-order elliptic wave filter, which contains 26 additions, 8 multiplications, 35 variables and 8 constants (Fig. 7), is used as an input instance of data-path synthesis. We assume that an addition is performed on an adder in one control step, and a multiplication is performed on a (non-pipelined) multiplier in two control steps.

The algorithm is implemented using C on PENTIUM III (1GHz) personal computer. The total computation time is about 1 hour (for details, see Table 1), and the evaluation of each solution (mainly scheduling feasibility and data assignment to input terminals of a functional unit) takes 1.33 millisecond on average.

Table 1 shows the results of our proposed method (from SEAS1 to SEAS4), together with results of other methods for the same instance. The results of SE, HAL, EMUCS and MABAL are quoted from [7], and the result of SPLICER is quoted from [8]. In the table, FU and R show the numbers of functional units and registers, respectively. Mx and Mi shows the numbers of multiplexers and multiplexer's inputs, respec-

**Table 1** Experimental results.

| System | $T_r$ | FU | R | Mx | Mi | ME | # | CT |
|---|---|---|---|---|---|---|---|---|
| **SEAS1** | 21 | 2+, 1× | 11 | 7 | 21 | 14 | 29 | 77 |
| SE | 21 | 2+, 1× | 11 | 8 | 24 | 16 | - | - |
| SPLICER | 21 | 2+, 1× | - | 9 | 43 | 34 | - | - |
| MABAL | 21 | 2+, 2× | 11 | 13 | 43 | 30 | - | - |
| **SEAS2** | 19 | 2+, 2× | 10 | 9 | 24 | 15 | 31 | 64 |
| SE | 19 | 2+, 2× | 10 | 11 | 31 | 20 | - | - |
| HAL | 19 | 2+, 2× | 12 | - | 29 | - | - | - |
| EMUCS | 19 | 2+, 2× | 12 | 12 | 34 | 22 | - | - |
| **SEAS3** | 17 | 2+, 2× | 11 | 8 | 23 | 15 | 31 | 60 |
| **SEAS4** | 16 | 3+, 2× | 11 | 12 | 31 | 19 | 37 | 49 |

tively. ME shows the number of equivalent two-input single-output multiplexers. # denotes the number of point-to-point interconnections. Note that, in Mx, Mi, ME and #, we eliminate the interconnections between constant data (multiplier) and input terminals of functional unit. CT shows the total computation time in minutes.

In Table 1, results are grouped by iteration period. SEAS1 and SEAS2 synthesized by our method are solutions with 12.5–25% reduced numbers of multiplexers, multiplexer's inputs and equivalent two-input single-output multiplexers compared to best numbers of them from other methods in each group. SEAS3 and SEAS4 are the best solutions, with respect to not only the number of multiplexers but also the number of FUs under given $T_r$, that ever appeared in literatures.

## 5. Conclusion

In this paper, we propose an approach using parametric scheduling graph with disjunctive arcs generated from the specified assignment information (operation to functional unit assignments and data to register assignments) in loop pipeline scheduling problem, and we derive a branch-and-bound solution method with successive refinement of parameter space.

As an application of the proposed scheduling method, it is incorporated with Simulated-Annealing based exploration of assignment solution space, and it is demonstrated that data-paths of the fifth-order elliptic wave filter are successfully synthesized.

The current version of our scheduler accepts a strongly connected dependence graph as an input to ensure the finiteness of the range of unknown variables on disjunctive arcs. Treatment of input dependence graph which is not strongly connected is left for a future work. The development of an efficient heuristic method is also an important future work.

### References

[1] P. Michel, U. Lauther, and P. Duzy, The synthesis approach to digital system design, Kluwer Academic, 1992.
[2] K. Ohashi, M. Kaneko, and S. Tayu, "Assignment-space exploration approach to concurrent data-path/floorplan synthesis," Proc. IEEE Int. Conf. On Computer Design (ICCD), pp.370–375, 2000.
[3] T. Kim, K.-S. Chung, and C.L. Liu, "A stepwise refinement synthesis of digital systems for testability enhancement," IEICE Trans. Fundamentals, vol.E82-A, no.6, pp.1070–1081, June 1999.
[4] K. Ito and H. Kunieda, "VLSI system compiler for digital signal processing: Modulation and synchronization," IEEE Trans. Circuits & Syst., vol.38, no.4, pp.422–433, 1991.
[5] T. Watanabe, N. Ishiura, and M. Yamaguchi, "A code generation method for datapath oriented codesign of application specific DSPs," The 13th Workshop on Circuit and Systems in Karuizawa, pp.539–544, 2000.
[6] B. Mesman, M. Strik, A.H. Timmer, J.L. van Meerbergen, and J.A.G. Jess, "A constraint driven approach to loop pipelining and register binding," Proc. IEEE DATE, pp.377–383, 1998.
[7] Tai A. Ly and Jack T. Mowchenko, "Applying simulated evolution to high level synthesis," IEEE Trans. Comput. -Aided Des. Integrated Circuits & Syst., vol.12, no.3, pp.389–408, 1993.
[8] B.M. Pangril, "Splicer: A heuristic approach to connectivity binding," Proc. 25th ACM/IEEE Design Automation Conference, pp.536–541,1988.

**Toshiyuki Yorozuya** received the M.E. degree in Information Science, from Japan Advanced Institute of Science and Technology, Hokuriku (JAIST) in 2001. He is currently studying towards the Ph.D. degree in School of Information Science, JAIST. His research interests include scheduling theory and computer aided design for VLSI circuits.

**Koji Ohashi** received the B.E.degree in electronic engineering from Nagaoka University of Technology in 1998. He received the M.E. degree in Information Science from Japan Advanced Institute of Science and Technology, Hokuriku (JAIST) in 2000. Since 2000 he has been a doctoral candidate in School of Information Science, JAIST. His research interests include high level synthesis for VLSI.

**Mineo Kaneko** received the B.E., M.E. and Dr.E. degrees in Electrical and Electronic Engineering from Tokyo Institute of Technology in 1981, 1983 and 1986, respectively. In 1986 he joined the Department of Electrical and Electronic Engineering from Tokyo Institute of Technology as a Research Associate. From 1992 to 1996, he was Associate Professor in the same Department of Tokyo Institute of Technology. From 1996 to 2001, he was Associate Professor in School of Information Science, Japan Advanced Institute of Science and Technology, Hokuriku (JAIST). He is now a Professor in School of Information Science, JAIST. His research interests include high-speed and fault tolerant VLSI signal processing, computer aided design of integrated circuits and wafer scale integration. He received best paper awards from IEEE APCCA'92 and APCCA'94 in 1992 and 1994, respectively. He is a member of IEEE.