An Optimal Scheduling Method for Parallel Processing System of Array Architecture

Kazuhito Ito
Dept. of Elec. and Elect. Syst.
Saitama University
Urawa, Saitama 338, Japan
Tel: +81-48-858-3731
Fax: +81-48-855-0940
kazuhito@elc.ees.saitama-u.ac.jp

Tadashi Iwata
Dept. of Elec. and Elect. Eng.
Tokyo Institute of Technology
Meguro-ku, Tokyo 152, Japan
Tel: +81-3-5734-2574
Fax: +81-3-5734-2911
{tiwata,kunieda}@ss.titech.ac.jp

Hiroaki Kunieda

Abstract— In high-level synthesis for digital signal processing systems of array structured architecture, one of the most important procedures is the scheduling. By taking into account the allocation of operations to processors, it is mandatory to take into account the communication time between processors. In this paper we propose a scheduling method which derives an optimal schedule achieving the minimum iteration period and latency for a given signal processing algorithm on the specified processor array. The scheduling problem is modeled as an integer linear programming and solved by an ILP solver. Furthermore, we improve the scheduling method so that it can be applied to large scale signal processing algorithms without degrading the schedule optimality.

I. Introduction

With the development of VLSI technology, wire delay is becoming relatively larger than gate delay [1]. To implement a high speed VLSI, it is very important to estimate not only the gate delay but the wire delay even in the high-level design. The parallel processing system on an array architecture is one of the suitable architectures for high speed VLSIs of the next generation since it realizes parallel processing which is the key to fully utilize an enormous number of gates on a VLSI [2, 3]. In the array architecture, the direct data communications are limited to PEs which are physically adjacent on a VLSI chip. The data communication between not physically adjacent PEs is achieved by intermediate PEs relaying the data. In this communication model, it is easy to estimate the wire delay (data communication delay) in high-level design of an array architecture. The data communication time is proportional to the distance of the source and the destination PEs.

One of the most important procedures of high-level synthesis is scheduling. In this paper, a scheduling method for a parallel processing system of an array architecture is proposed. In general, scheduling consists of time assignment and processor allocation. The time assignment is to determine when each operation is executed. The processor allocation it to determine which PE executes the operations. It is well known that the optimal scheduling must consider the time assignment and the processor allocation simultaneously and it is a NP-hard problem [4]. To improve CPU time for the scheduling, some of the existing scheduling techniques divide the scheduling problem into the time assignment and the processor allocation at the cost of the solution optimality. However, the scheduling for an array architecture do have to consider time assignment and processor allocation simultaneously. This is because the processor allocation affects the data communication time between operations and the time assignment depends on the data communication time. In addition, the time assignment to resolve resource conflict affects the processor allocation. To precisely obtain the optimal scheduling, it is modeled as an integer linear programming (ILP) problem and solved by an ILP solver [5, 6, 7]. In this paper, an ILP model of scheduling for an array architecture is formulated. The ILP model contains a large number of variables and constraints and therefore the CPU time is very long to solve the ILP model. A technique to find an optimal schedule by using modified ILP models to improve the CPU time is proposed in this paper.

II. Scheduling Model for High-Level Synthesis

The scheduling model of array architecture is defined as follows.

1. Array topology
   The number of PEs and the topology of array structure are given as a specification.
2. Processing element
   A Processing element (PE) can execute operations and data communications with adjacent PEs simultaneously. In addition, a PE can relay data from an adjacent PE to another adjacent PE as long as no conflict on communication links occur.
3. Data communication
   Data communication links are limited between physically adjacent PEs. Data communication between physically distant PEs is achieved by intermediate PEs relaying the data. Therefore, data communication time is proportional to the distance between the sender PE and the receiver PE.

ASP-DAC '97
0-89791-851-7$5.00

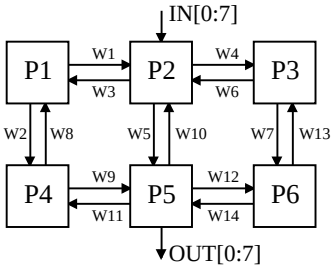

Fig. 1. Hardware model of array architecture.

4. Data Input/Output
   The locations of PEs which input and/or output the data are given as specification. Moreover, if the processing algorithm consumes and produces multiple data, then the data format of input and output is also specified.

Based on the scheduling model defined above, scheduling is done to satisfy the following scheduling constraints.

1. Satisfy precedence relations
   If there exists data dependency between operations, the precedence relation between these operations must be satisfied. Namely, if an operation depends on the data produced by another operation, the former operation cannot start until the latter completes the execution and the produced data is sent to the former operation.

2. No resource conflict
   Let *resource conflict* be defined as the situation that the resource is used at the same time by more than one executions. Hence if resource conflict occurs in a schedule, the schedule cannot be realized. No more than one operations can be executed on a PE at the same time. In addition, no more than one data can be sent/received on a data communication link at the same time.

The objective of the scheduling is to find a schedule which achieves the minimum iteration period for a given processing algorithm and a given array topology. If there exist more than one such schedule, then choose one which achieves the minimum latency. Latency is defined as the time difference between input of data and output of related data. In this paper, register minimization is not considered. Namely, a PE is assumed to be able to store any number of data.

III. Basic Scheduling Method

A. Scheduling strategy

At first, we construct an ILP model to decide whether a schedule of a processing algorithm exists or not which satisfies all the scheduling constraints for a specified iteration period and latency on a PE array of a given topology. This ILP model is called *complete model* since the model completely checks the existence of resource conflict.

The basic scheduling method by using the complete model is illustrated in Fig. 2. First, the lower bound of iteration period and the lower bound of latency are computed and set a guess iteration period and a guess latency to these lower bounds. Then the complete model is generated and run to decide whether a schedule exists. If the complete model does not terminate with a solution, i.e., no schedule satisfying scheduling constraints exists for the guess iteration period and the guess latency, then increase the latency or the iteration period, and generate and run the complete model again. By repeating the process, the complete model eventually terminates with a solution, i.e., a schedule satisfying all the scheduling constraints. At this point, a valid schedule achieving the minimum iteration period and the minimum latency has been obtained. It must be noted that the above repetition always terminates and therefore an optimal schedule is always obtained. This is because a schedule where all the operations are executed sequentially on one of the PEs is a valid schedule and it can be obtained if the iteration period and the latency are sufficiently large.

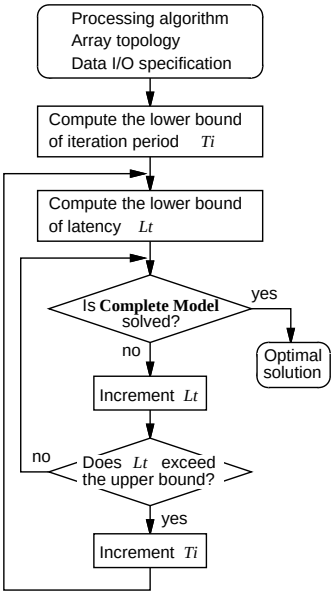

Fig. 2. Basic scheduling method.

B. Complete model

The complete ILP model is defined as Eq. (1) to (10) to decide whether a schedule of a processing algorithm exists which satisfies all the scheduling constraints for a specified iteration period and latency on a given topology of PE array.

The following terminology is used.

N	the set of operation nodes.
IN	the set of nodes to input data.
OUT	the set of nodes to output data.
P	the set of processing elements.
W	the set of data communication links between PEs.
Lt	the latency of the processing algorithm.
Ti	the iteration period of the processing algorithm.
C_i	the computation latency of operation $i \in N$.
L_i	the pipeline period of operation $i \in N$.
$D_{i,ip}$	the number of delay elements on edge (i, ip).

$FIXj_i$	the time when data $i \in IN(OUT)$ is input(output).
$FIXk_i$	the index of PE where data $i \in IN(OUT)$ is input(output).
$X_{i,j,k}$	a binary variable. $X_{i,j,k}$ = 1 implies that operation $i \in N$ starts at time j on PE k.
$Y_{i,j,l}$	a binary variable. $Y_{i,j,l}$ = 1 implies that a data produced by operation $i \in IN+N$ is sent at time j through data communication link l.
$ASAP_i$	the earliest starting time of operation $i \in N$.
$ALAP_i$	the latest starting time of operation $i \in N$.
Rx_i	the time interval in which operation i can start. $Rx_i = \{ASAP_i, ASAP_i + 1, \ldots, ALAP_i\}$
$ASAP_{y_i}$	the earliest starting time of communication of a data produced by operation i.
$ALAP_{y_i}$	the latest starting time of communication of a data produced by operation i.
Ry_i	the time interval in which communication of a data produced by operation i can start. $Ry_i = \{ASAP_{y_i}, ASAP_{y_i} + 1, \ldots, ALAP_{y_i}\}$
fr_l	the source PE of data communication link $l \in W$.
to_l	the sink PE of data communication link $l \in W$.

The *computation latency* is the time difference from an operation is started until the operation result is output. The pipeline period is the smallest time interval between successive invocation of operations on a PE. Basically there is no restriction on the data communication time between adjacent PEs in the proposed ILP model. In this paper, a data communication between adjacent PEs is assumed to take one unit of time.

Here $i \Rightarrow ip$ implies a data dependency from operation i to operation ip.

Eq. (1) ensures that an operation of each node is executed only once. Eq. (2) ensures that at most one operation is executed at the same time on each PE and hence resolves resource conflict for functional units. Eq. (3) ensures that each data is sent from one PE to another only once. Eq. (4) ensures that at most one data is sent at the same time on each data communication link and hence resolves resource conflict for data communication link. Eq. (5) and eq. (6) constrain precedence relations between data communication and data communication and between operation and data communication, respectively, in the case of data relaying. Eq. (7) and eq. (8) constrains precedence relations between data communication and operation and between operation and operation, respectively. Eq. (9) and eq. (10) decide whether data output time satisfies precedence relations with the current guess latency.

C. The upper bound of latency

A schedule where all the operations are executed sequentially on the PE specified to input data and then the results are sent to the PE specified to output data is a valid schedule since any resource conflict occurs. In this schedule, the latency is not longer than the sum of the total operation time and the distance between PEs which respectively inputs and outputs data. Let the sum of these time be denoted as Lt_M. Any schedule with a latency longer than Lt_M can be achieved by inserting idle time into the above mentioned schedule. Therefore it is not necessary to check the existence of a schedule for a latency longer than Lt_M. Consequently the upper bound of the latency Lt in the scheduling method shown in Fig. 2 is Lt_M.

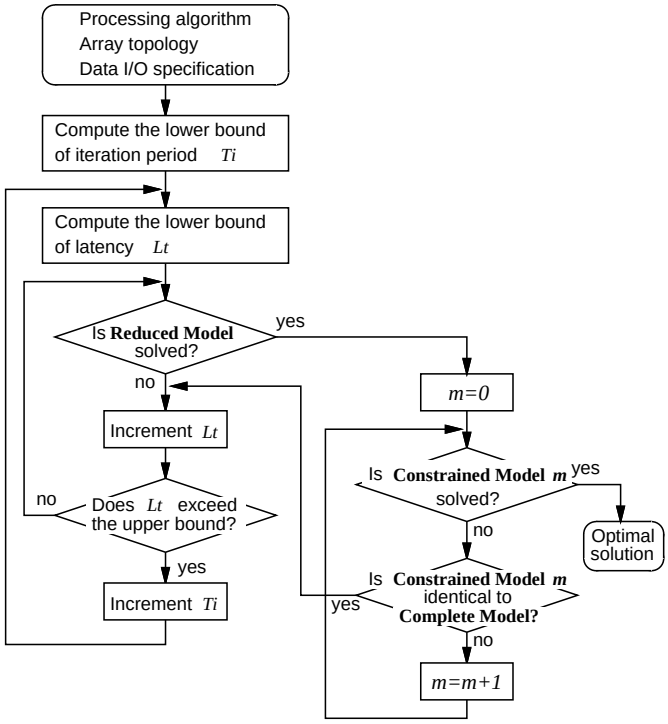

Fig. 3. Refined scheduling method.

IV. Refined Scheduling Method

The basic scheduling method described in the previous section requires execution of the complete ILP models. To strictly constrain precedence relations and check resource conflict, the complete model requires many binary variables for a large processing algorithm and therefore its solution time is very long and sometimes it cannot be solved. In this section, a refined scheduling method and ILP models are proposed to handle scheduling of large processing algorithms and achieve shorter CPU time for scheduling.

The proposed refined scheduling method is illustrated in Fig. 3. While the basic scheduling method employs only the complete model, the refined scheduling method employs two ILP models: *reduced model* and *constrained model*. The reduced model is the complete model except that the existence of resource conflict on data communication links is not checked. The purpose of the reduced model is to determine a start time of each operation so that precedence relations are satisfied. The constrained model is the complete model except that the time intervals in which operations could start are limited based on the start time determined by the reduced model. Both ILP models can be solved much faster than the complete model. By using these two ILP models, the CPU time to derive the opti-

$$\sum_{j \in Rx_i} \sum_{k \in P} X_{i,j,k} = 1 \qquad \forall i \in N \tag{1}$$

$$\sum_{i \in N} \Big\{ \sum_{q=0}^{L_i - 1} \Big\{ \sum_{p=0}^{\lfloor (ALAP_i - j - q)/Ti \rfloor} X_{i,j+p*Ti-q,k} \Big\} \Big\} \leq 1 \qquad 1 \leq j \leq Ti \ \forall k \in P \tag{2}$$

$$\sum_{j \in Ry_i} Y_{i,j,l} \leq 1 \qquad \forall i \in IN + N, \forall l \in W \tag{3}$$

$$\sum_{i \in IN+N} \sum_{p=0}^{\lfloor (ALAP_{y_i} - j)/Ti \rfloor} Y_{i,j+p*Ti,l} \leq 1 \qquad 1 \leq j \leq Ti \ \forall l \in W \tag{4}$$

$$Y_{i,j,l} \leq \Big\{ \sum_{jp<j} \sum_{\substack{lp \\ to_{lp}=fr_l}} Y_{i,jp,lp} + \sum_{jp \leq j - C_i} X_{i,jp,fr_l} \Big\} \qquad \forall i \in N, \forall j \in Ry_i, \forall l \in W \tag{5}$$

$$Y_{i,j,l} \leq \Big\{ \sum_{jp<j} \sum_{\substack{lp \\ to_{lp}=fr_l}} Y_{i,jp,lp} + \alpha \Big\} \qquad \forall i \in IN, \forall j \in Ry_i, \forall l \in W \tag{6}$$

$$\alpha = \begin{cases} 1 & \text{if } j > FIXj_i \text{ and } k = FIXk_i \\ 0 & \text{otherwise} \end{cases}$$

$$X_{ip,j,k} \leq \sum_{jp<j+D_{i,ip}*Ti} \sum_{\substack{l \\ to_l=k}} Y_{i,jp,l} + \sum_{jp \leq j+D_{i,ip}*Ti-C_i} X_{i,jp,k} \qquad i \ i \ p \in N, \forall i \Rightarrow i \ p \forall j \in Rx_i, \forall k \in P \tag{7}$$

$$X_{i,j,k} \leq \sum_{jp<j+D_{i,ip}*Ti} \sum_{\substack{l \\ to_l=k}} Y_{i,jp,l} + \alpha \qquad i \in IN, i \ p \notin N, \forall i \Rightarrow i \ p \forall j \in Rx_i, \forall k \in P \tag{8}$$

$$1 \leq \sum_{k=FIXk_{ip}} \Big\{ \sum_{j<FIXj_{jp}+D_{i,ip}*Ti} \sum_{\substack{l \\ to_l=k}} Y_{i,j,l} + \sum_{jp \leq FIXj_{ip}+D_{(i,ip)}*Ti-C_i} X_{i,j,k} \Big\} \qquad i \in IN, i \ p \in OUT, \forall i \Rightarrow i \ p \tag{9}$$

$$1 \leq \sum_{k=FIXk_{ip}} \sum_{j<FIXj_{ip}+D_{i,ip}*Ti} \sum_{\substack{l \\ to_l=k}} Y_{i,j,l} + \beta \qquad i \in IN, i \ p \in OUT, \forall i \Rightarrow i \ p \tag{10}$$

$$\beta = \begin{cases} 1 & \text{if } FIXj_i < FIXj_{ip} + D_{i,ip} * Ti \text{ and } FIXk_i = FIXk_{ip} \\ 0 & \text{otherwise} \end{cases}$$

mal schedule is greatly reduced without degrading the schedule optimality.

A. Schedule existence decision by reduced model

The reduced model is such an ILP model that the existence of a schedule is decided where all the precedence relations are satisfied and no resource conflict occur on PEs but resource conflict on data communication links is ignored. In other words, the reduced model is a complete model for an array with infinite number of data communication links between adjacent PEs. Binary variables $Y_{i,j,l}$ are not necessary in the reduced model and hence the number of binary variables is greatly redudecd. The reduced model can be solved more easily than the complete model.

Let a cutset of an array be defined as the set of data communication links such that removal of those divides the array into two connected components as illustrated in Fig. 4. Each cutset has its maximum data flow capacity. It is calculated as the number of data communication links in the cutset multiplied by the iteration period Ti. For example in the case of the cutset shown in Fig. 4, the cutset consists of 5 data communication links. If the iteration period is 6 units of time, the capacity of the cutset is 30. In each iteration period, at most 30 data can be sent from PEs of the upper component to PEs of the lower component and vice versa. For any schedule found by the reduced model, if the total number of data communications between PEs belonging to different components of a cutset is greater than the data flow capacity of the cutset, then the schedule must contain resource conflict on at least one of the data communication links. If such a schedule is allowed, all the constrained ILP models (defined in the next section) following the reduced model report that there exists no schedule without resource conflict after a long CPU time. To overcome this problem, the reduced model counts the number of data communications for all the cutsets which divide PEs into one PE

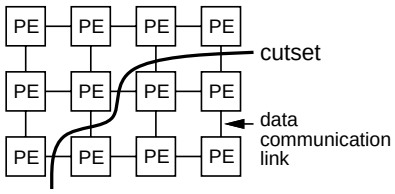

Fig. 4. Cutset of array.

and any other PEs and checks if it is no more than the data flow capacity of the cutset. Although new binary variables are introduced to count the number of data communications which cross a cutset, this augment of binary variables is much smaller than the reduction of binary variables $Y_{i,j,l}$.

In addition to the terminology defined in section B, the following terminology is defined.

$PCON_{k,kp}$	data communication time between PE k and PE kp.
$flowf_{k,i}$	the binary variable to imply data flow-out. $flowf_{k,i}$ = 1 implies that a data produced by operation i is output from PE k.
$flowt_{k,i}$	the binary variable to imply data flow-in. $flowt_{k,i}$ = 1 implies that a data produced by operation i is input to PE k.

Eq. (11) ensures that an operation of each node is executed only once. Eq. (12) ensures that at most one operation is executed at the same time on each PE and hence resolves resource conflict Eq. (13)–(16) constrain precedence relations between operation and operation, between input and operation, between operation and output, and between input and output, respectively. Eq. (17)–(20) decide if the data produced by operation i is flown out from PE k. If operation i is executed on PE k, $i \Rightarrow ip$, and operation ip is executed on a PE other than PE k, then the data must be flown out from PE k. Eq. (21) restricts the number of data communications on a cutset which divides PEs into PE k and any other PEs to be within the data flow capacity of the cutset. The right hand side is the data flow capacity of the cutset. The left hand side is the number of data flown out from PE k. Similarly, Eq. (22)–(26) restrict the number of data flown into PE k.

It must be noted that the cost function (27) is maximized in the reduced model. In the constrained ILP model (defined in the next section) which follows the reduced model, the complete schedule is found where any resource conflict on data communication links as well as PEs does not occur. For a pair of operations i and ip such that $i \Rightarrow ip$, if the time difference from the execution of operation i to the execution of operation ip is large, then it becomes easy to resolve resource conflict by modifying the execution time of these operations without violating the precedence relation between operations i and ip. Hence

$$Z = \sum_{i \Rightarrow ip} \left(\begin{array}{c} \text{start time of operation } ip - \\ \text{end time of operation } i \end{array} \right),$$

which is the sum of time difference for all the pair of operations with a data dependency between the operations, is maximized by Eq. (27).

B. Schedule existence decision by constrained model

The reduced model determines the start time for each operation so that precedence relations between operations are satisfied including data communication time and no resource conflict on PEs exists. Based on the start time determined by the reduced model, the constrained models find a schedule where all the precedence relations are satisfied and no resource conflict on data communication links as well as PEs exists.

The constrained model is parameterized by a nonnegative integer m. Let t_i denote the start time of operation i determined by the reduced model. In the complete model, the time interval in which operation i could start is $Rx_i = \{t \mid ASAP_i \leq t \leq ALAP_i\}$. The constrained model m employs the same equations (constraints) as the complete model but the time interval in which operation i could start is $Rx_i = \{t \mid \max(ASAP_i, t_i - m) \leq t \leq \min(ALAP_i, t_i + m)\}$. Namely, the constrained model m checks whether a schedule without any resource conflict exists or not by assuming that the start time of operation i can be shifted $\pm m$ units of time from t_i.

The constrained model $m = 0$ checks the existence of a schedule by fixing the start time of all the operations as determined by the reduced model. If the constrained model m terminates without a solution, i.e., any schedule without resource conflict does not exist, then m is incremented by one and the constrained model is run again. By repeating the procedure, finally $t_i - m \leq ASAP_i$ and $t_i + m \geq ALAP_i$ hold for all the operation i. At this point the constrained model m is identical to the complete model. If all the constrained models terminate without a solution, it implies that a schedule without resource conflict does not exist at current guess iteration period Ti and guess latency Lt. In this case, increase the guess latency or the guess iteration period and repeat from the reduced model.

Through the preliminary experiments, a prospect has been obtained that the start times of operations in the final schedule without any resource conflict are just the same or very close to those derived by the reduced model. This implies that the schedule without any resource conflict is likely to be found even when the time interval is small. Hence if an optimal schedule exists, it is expected to be found by the constrained model with a small m. This is the reason that the refined scheduling method would find an optimal solution within a short CPU time.

As m increases, the time interval, i.e., the search space gets larger and finally the constrained model becomes identical to the complete model. Therefore, the schedule obtained by the refined scheduling method is as optimal as the schedule obtained by the basic scheduling method.

V. Experimental Results

A. 8 point DCT

Fig. 5 shows a data-flow graph of 8 point discrete cosine transform (DCT) [8]. This processing algorithm is implemented onto an array shown in Fig. 1. As a specification, 8 input data $IN[0:7]$ are input to PE $P2$ at time steps 0 to 7 respectively and 8 output data $OUT[0:7]$ are output from PE $P5$ at time steps Lt to $Lt+7$ respectively where Lt is the specified latency. The operation execution time is assumed to be 2 units of time (u.t.) for a multiplication and 1 u.t. for an addition. In addition, operations are assumed to be not pipelined. There are 11 multiplications and 29 additions and hence the total operation execution time is 51 u.t. Since 6 PEs exist in the

$$\sum_{j \in Rx_i} \sum_{k \in P} X_{i,j,k} = 1 \quad \forall i \in N \tag{11}$$

$$\sum_{i \in N} \{ \sum_{q=0}^{L_i - 1} \{ \sum_{p=0}^{\lfloor (ALAP_i - j - q)/Ti \rfloor} X_{i,j+p*Ti-q,k} \} \} \leq 1 \quad 1 \leq j \leq Ti \ \forall k \in P \tag{12}$$

$$X_{ip,j,k} \leq \sum_{kp \in P} \sum_{jp < j - C_i - PCON_{k,kp} + D_{i,ip} * Ti} X_{i,jp,kp} \quad i \ i_p \in N, \forall i \Rightarrow i \ p \forall j \in Rx_i, \forall k \in P \tag{13}$$

$$X_{ip,j,k} \leq \begin{cases} 1 & (j > FIXj_i + PCON_{k,FIXk_i} - D_{i,ip} * Ti) \\ 0 & \text{(otherwise)} \end{cases} \quad i \in IN, i \ p \in N, \forall i \Rightarrow i \ p \forall j \in Rx_{ip}, \forall k \in P \tag{14}$$

$$X_{i,j,k} \leq \begin{cases} 1 & (j \leq FIXj_{ip} - C_i - PCON_{k,FIXk_{ip}} + D_{i,ip} * Ti) \\ 0 & \text{(otherwise)} \end{cases} \quad i \in N, i \ p \in OUT, \forall i \Rightarrow i \ p \forall j \in Rx_i, \forall k \in P \tag{15}$$

$$FIXj_{ip} - FIXj_i + D_{i,ip} * Ti - 1 \geq PCON_{FIXk_i, FIXk_{ip}} \quad i \in IN, i \ p \in OUT, \forall i \Rightarrow i \ p \tag{16}$$

$$flowf_{k,i} \geq \sum_{j \in Rx_i} X_{i,j,k} - \sum_{j \in Rx_{ip}} X_{ip,j,k} \quad i \ i_p \in N, \forall i \Rightarrow i \ p \forall k \in P \tag{17}$$

$$flo\ uf_{k,i} \geq 1 - \sum_{j \in Rxip} X_{ip,j,k} \quad i \in IN, i \ p \in N, \forall i \Rightarrow i \ pk = FIXk_i \tag{18}$$

$$flo\ uf_{k,i} \geq \sum_{j \in Rxi} X_{i,j,k} \quad i \in N, i \ p \in OUT, \forall i \Rightarrow i \ p \forall k \neq FIXk_{ip} \tag{19}$$

$$flo\ uf_{k,i} \geq 1 \quad i \in IN, i \ p \in OUT, \forall i \Rightarrow i \ pk = FIXk_i, k \neq FIXk_{ip} \tag{20}$$

$$\sum_{i \in IN+N} flo\ uf_{k,i} \leq \sum_{\substack{kp \\ PCON_{k,kp}=1}} Ti \quad \forall k \in P \tag{21}$$

$$flo\ ut_{k,i} \geq -\sum_{j \in Rx_i} X_{i,j,k} + \sum_{j \in Rx_{ip}} X_{ip,j,k} \quad i \ i_p \in N, \forall i \Rightarrow i \ p \forall k \in P \tag{22}$$

$$flo\ ut_{k,i} \geq \sum_{j \in Rxip} X_{ip,j,k} \quad i \in IN, i \ p \in N, \forall i \Rightarrow i \ p \forall k \neq FIXk_i \tag{23}$$

$$flo\ ut_{k,i} \geq -\sum_{j \in Rxi} X_{i,j,k} + 1 \quad i \in N, i \ p \in OUT, \forall i \Rightarrow i \ pk = FIXk_{ip} \tag{24}$$

$$flo\ ut_{k,i} \geq 1 \quad i \in IN, i \ p \in OUT, \forall i \Rightarrow i \ pk = FIXk_{ip}, k \neq FIXk_i) \tag{25}$$

$$\sum_{i \in IN+N} flo\ ut_{k,i} \leq \sum_{\substack{kp \\ PCON_{k,kp}=1}} Ti \quad \forall k \in P \tag{26}$$

$$\begin{aligned} \text{Maximize } Z = & \sum_{i \in IN+N} \sum_{\substack{ip \in N \\ i \Rightarrow ip}} \{ \sum_{j \in Rx_i} \sum_{k \in P} j * X_{i,j,k} \} + \sum_{i \in IN+N} \sum_{\substack{ip \in OUT \\ i \Rightarrow ip}} FIXj_{ip} \\ & - \{ \sum_{i \in N} \sum_{\substack{ip \in N+OUT \\ i \Rightarrow ip}} \{ \sum_{j \in Rx_i} \sum_{k \in P} j * X_{i,j,k} \} + C_i - D_{i,ip} * Ti \} \\ & - \sum_{i \in IN} \sum_{\substack{ip \in N+OUT \\ i \Rightarrow ip}} \{ FIXj_i + 1 - D_{i,ip} * Ti \} \end{aligned} \tag{27}$$

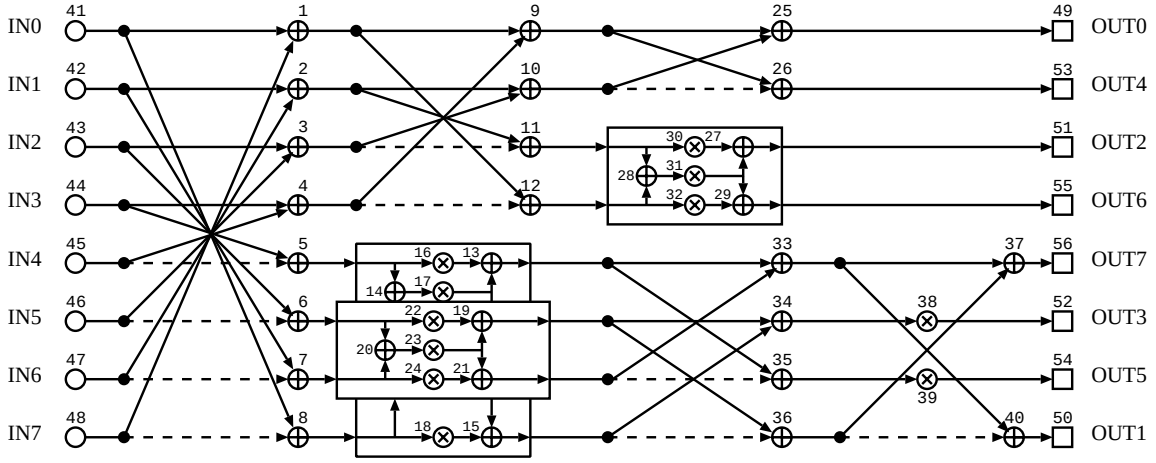

Fig. 5. Data-flow graph of 8 pint DCT.

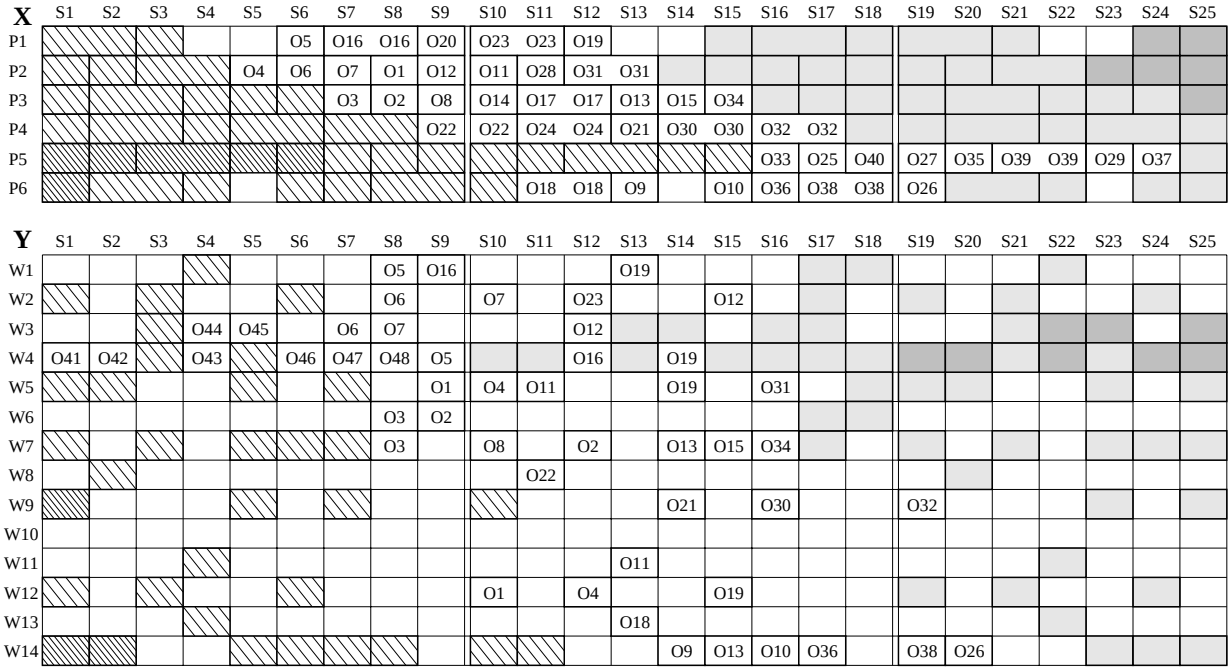

Fig. 6. An optimal schedule of 8 point DCT.

array, the lower bound of the iteration period is $\lceil 51/6 \rceil = 9$ u.t. This implies that there exists no schedule with an iteration period less than 9 u.t.

The first row of Table I shows the CPU times (the unit is second) of the scheduling done by the basic scheduling method and the refined scheduling method. All the ILP models are solved by the ILP solver GAMS/OSL[9] running on a 75MHz Sparc workstation. When the iteration period is 9 u.t. and the latency is 18 u.t., the complete model is solved, which implies the existence of a solution, and the CPU time is 15 hours 33 minutes and 22 seconds (56002 seconds). The iteration period achieves its lower bound and there exists no schedule with a latency less than 18 u.t. Hence the schedule derived is the optimal one. In the refined scheduling method, however, the CPU time for the reduced model is 10 minutes and 50 seconds (650 seconds) and the constrained model $m = 0$ terminates with a solution after 44 seconds. Thus a schedule without any resource conflict is obtained in 11 minutes and 34 seconds. Fig. 6 shows the time chart of the optimal schedule obtained by the refined scheduling method. In Fig. 6, the upper chart shows the schedule for operations. For example, operation 5, which is an addition, is executed on PE P1 at time S6. The lower chart in Fig. 6 shows the schedule for data communications. For example, the result of operation 1 is communicated on link W5 at time S9 and on link W12 at time S10, i.e., it is sent from P2 (the source of W5) to P6 (the sink of W12) through P5 (the sink of W5 and the source of W12). Dense hatching represents the operations and data communications of the 2nd previous iteration, sparse hatching the previous iteration, light gray the next iteration, and the dark gray the 2nd next iteration. Space imply that PEs and data communication links are idle.

In the schedule derived by the refined scheduling method, the

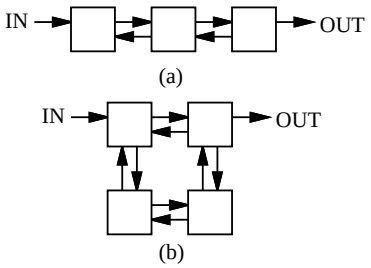

Fig. 7. Array topology and data input/output specification. (a) For 4th order Jaumann wave digital filter. (b) For 16 point FIR filter and 5th order wave elliptic digital filter.

iteration period is 9 u.t. and the latency is 18 u.t. Consequently, by using the refined scheduling method, 80 times speed up is achieved in deriving an optimal schedule.

B. Benchmarks

The proposed scheduling methods are applied to processing algorithms, such as 4th order Jaumann wave digital filter (JAU) [10], 16 point FIR filter (FIR) [11], and 5th order wave elliptic digital filter (WEF) [12]. The topology of the array used is shown in Fig. 7. The CPU times in second are summarized in Table I. Table I shows: the name of a processing algorithm; the specified iteration period Ti; CPU time for the complete model; CPU time for the reduced model (RM); CPU time for the constrained model (CM) m = 0; CPU time for the constrained model m = 1; and the CPU time ratio between the basic and refined scheduling methods. In the case of JAU an optimal schedule without resource conflict is obtained by the constrained model m = 1. In any other case, an optimal schedule without resource conflict is obtained by the constrained model m = 0. Although the lower bound of the iteration period of WEF is 16 u.t., there exists no schedule when the iteration period is less than 18 u.t. In any other case, the iteration period is the same as the lower bound and hence the schedule achieves the minimum iteration period.

Since two or more ILP models, i.e., the reduced model and the constrained models m are used in the refined scheduling method, there can be a case where the total CPU time becomes longer than the basic scheduling method, especially for a small size processing algorithm such as 4th order Jaumann wave digital filter. As shown in Table I, however, the absolute increase of the CPU time is acceptable. On the other hand for relatively larger processing algorithm such as 8 point DCT, the CPU time is greatly improved. Consequently it can be concluded that the proposed refined scheduling method is effective.

TABLE I
COMPARISON OF SCHEDULING METHODS

DFG	Ti	Basic	Refined			CPU ratio
			RM	CM m		
				0	1	
DCT	9	56002	650	44	—	80.69
JAU	10	7	26	3	10	0.18
FIR	8	248	24	6	—	8.23
WEF	16	52959	4789	—	—	11.06
	17	57305	7504	—	—	7.64
	18	305	734	42	—	0.39
	total	110569	13027	42	—	8.46

VI. CONCLUSIONS

In this paper a scheduling method was proposed to obtain an optimal scheduling method for a multiprocessor system of array architecture. The proposed scheduling method employs the reduced ILP model and the constrained ILP model to derive a schedule which achieves the minimum iteration period and the minimum latency without any resource conflict. By experimental results, the effectiveness of the proposed scheduling method was verified.

Acknowledgment

This work has been engaged as a project in CAD21 Research Body of Tokyo Institute of Technology. We wish to thank all the members of CAD21 for their suggestions and cooperations.

REFERENCES

[1] M. Yamashina, "Prospect of Sub-Quarter Micron LSI Design," in *IEICE Tech. Report*, vol. VLD95-136, pp. 53–60, 1996.

[2] S. Y. Kung, *VLSI Array Processing*. Englewood Cliffs, N.J.: Prentice Hall, 1988.

[3] K. Ito, K. Hagiwara, and H. Kunieda, "Neo-Systolic Array: A Hardware Model for VLSI System Compiler VEGA," in *The Proceeding of 1992 IEEE Asia-Pacific Conference on Circuits and Systems*, Sydney, pp. 313–318, Dec. 1992.

[4] M. R. Garey and D. S. Johnson, *Computers and Intractability: a Guide to the Theory of NP-completeness*. San Francisco: W. H. Freeman, 1979.

[5] C.-T. Hwang, J.-H. Lee, and Y.-C. Hsu, "A Formal Approach to the Scheduling Problem in High Level Synthesis," *IEEE Trans. Computer-Aided Design*, vol. CAD-10, pp. 464–475, Apr. 1991.

[6] C. H. Gebotys and M. I. Elmasry, "Global Optimization Approach for Architecture Synthesis," *IEEE Trans. Computer-Aided Design*, vol. CAD-12, pp. 1266–1278, Sept. 1993.

[7] K. Ito, L. E. Lucke, and K. K. Parhi, "Module Selection and Data Format Conversion for Cost-Optimal DSP Synthesis," in *Proc. ACM/IEEE Int. Conf. on Computer-Aided Design*, San Jose, pp. 322–329, Nov. 1994.

[8] C. Loeffler, A. Ligtenberg, and G. S. Moschytz, "Practical Fast 1-D DCT Algorithms with 11 Multiplications," in *Proc. IEEE ICASSP*, pp. 988–991, 1989.

[9] A. Brooke, D. Kendrick, and A. Meeraus, *GAMS: A User's Guide, Release 2.25*. South San Francisco, CA: The Scientific Press, 1992.

[10] M. Renfors and Y. Neuvo, "The Maximum Sampling Rate of Digital Filters under Hardware Speed Constraints," *IEEE Trans. Circuits Syst.*, vol. CAS-28, pp. 196–202, Mar. 1981.

[11] N. Park and A. C. Parker, "Sehwa: A Software Package for Synthesis of Pipelines from Behavioral Specifications," *IEEE Trans. Computer-Aided Design*, vol. 7, Mar. 1988.

[12] S. Y. Kung, H. J. Whitehouse, and T. Kailath, *VLSI and Modern Signal Processing*. Englewood Cliffs, NJ: Prentice Hall, 1985.