MULTIPAR: Behavioral Partitioning for Synthesizing Application-Specific Multiprocessor Architectures

Yunn-Yen Chen¹, Yu-Chin Hsu†, and Chung-Ta King†

¹Department of Computer Science
Hsin-chu, 30043, Taiwan, R.O.C.

Abstract

In this paper, we present methods for scheduling and partitioning behavioral descriptions in order to synthesize application-specific multiprocessor systems. Our target application domain is real-time digital signal processing (DSP). In order to meet the real-time constraints, maximizing the system throughput and minimizing the number of communications between processors are important. A model of the target processor and the communication device is defined as a basis for synthesizing the multiprocessor system. We use an integer linear programming formulation to solve the partitioning and scheduling problem simultaneously. The optimization complexity of large applications can be reduced by using a simplified formulation and an iterative partitioning heuristic. Our work also takes into account of conditional branches, loops, and critical signals.

1. Introduction

Real-time digital signal processing (DSP) applications require very high computation rates to meet hard real-time constraints. The target hardware is often dedicated to an application or a small class of applications. With the advance of integrated circuit technology, algorithms like low bit rate speech coding, speech synthesis, and medium bit rate video signal processing can now be realized on a single chip. However, for more complex algorithms and/or higher sampling rates the amount of computation required is beyond that can be supplied by single chip implementations. Multiprocessing is the only way to satisfy the hard real-time constraints. However, due to the complexity of the design space, development of such systems would require a tremendous effort. Yet, current practices still rely on designers' intuition and ad hoc approaches.

In order to shorten the design time, reduce the design errors, and explore the design space, automatic synthesis of application-specific multiprocessor architectures is indispensable. Automatic synthesis of application-specific multiprocessor systems consists of several subtasks. They are: determining the number and type of function units and communication scheme used, scheduling and partitioning the application into the processing elements to satisfy the constraints, and synthesizing data paths for processing elements and communication links for the system. The most important subtask is the scheduling and partitioning step, because it determines system throughput, processing load balance, and the amount of usable parallelism. The paper aims at approaches to solving the partitioning and scheduling problem in the automatic synthesis of application-specific multiprocessor systems. The flow of our scheduling and partitioning system is described in Fig. 1. The applications are described as control/data flow graph (CDFG). The system selects appropriate function units from resource library to satisfy the constraints and communication properties. It first uses an exact ILP formulation to optimize small- to medium-sized basic blocks and inner loops. The optimized code blocks are treated as multi-cycle operations during the following phases. A simplified ILP formulation and then iterative methods are used to optimize larger CDFGs. Since the timing of the architecture influences the scheduling and partitioning algorithm, it must be defined precisely.

2. Related Work

The APARTY system [1] is an automatic architectural partitioner in the System Architect's Workbench. It is based on a multi-stage clustering technique, which is an extension of that used in the BUD-DAA system [5]. In another research work, Schiermeier et al. [9] extended the approach of the BUD-DAA system. Evering used a simplified clustering approach and presented a partitioning method for a bus oriented design [4]. Camposano [6] uses clustering within the Yorktown Silicon Compiler to produce partitions. He shows that the partitioning reduces the time spent in logic synthesis and may improve the synthesis results.

Gupta and De Micheli [6] synthesized multi-chip systems from a behavioral description. The objectives are to satisfy both chip area and overall latency constraints. They formalized this problem as a hypergraph partitioning.

Prakash and Parker [7] proposed an approach to the problem of synthesizing application-specific multiprocessor architectures. The partitioning of the synthesized design is known a priori. CHOP [8] allows the designer to interactively create and modify partitions of behavioral specifications on multiple chips. Because CHOP is not intended to aid the synthesis of multiprocessor systems, the communication delays and the communication devices are not taken into consideration.

The CAMPUS system [8] used several stages to synthesize multiprocessor systems from behavioral descriptions. Global memory is used as inter-processor communication device. The operations must be partitioned into different processors by the user.

The rest of this paper is organized as follows. We overview previous work in Section 2. The processor and communication architectures are described in Section 3. We give an integer linear formulation of the partitioning and scheduling problem in Section 4. A simplified formulation to reduce the complexity of the problem instances is discussed in Section 5. In Section 6 we introduce an iterative algorithm to solve extreme large problem instances. We show the experimental results in Section 7. Finally, the concluding remarks and future work are given in Section 8.

Figure 1: The flow of our partitioning and scheduling system

---

¹This work is supported by the National Science Council of R.O.C. under Grant NSC81-0408-E007-10
Our work has several features. First, the processors used in our synthesis system can have more than one function unit. The designer can model different devices as processors. For example, off-chip memory can be modeled as a memory processor. Second, the timing of the partitioning is precisely considered during scheduling and partitioning process. It is not necessary to consider the timing mismatch problem afterward. Thus, the communication using devices with accessing constraints (for example, FIFO queue) can be scheduled in the step. Third, the partitioning algorithms perform in a hierarchical fashion. Different algorithms cooperate to solve problems from the basic block, subroutine, onto program level. The partition algorithms can also handle communication devices with different delays, such as cluster bus, tree bus, and segmented bus architectures.

3. The Target Architecture

In this section the model of the target architecture is introduced. An architecture model is necessary because it determines how we synthesize the system, especially the timing model that should be used. Such a model describes the minimal functionalities that the target architecture must have in order to apply our partitioning and scheduling algorithms.

Processor Architecture

The processing element used in our model consists of three major components: (1) storage; (2) function unit; and (3) communication unit. In the storage part, variables are stored in registers and constants in ROMs. Each processing element could have several function units operate simultaneously. Different function units may have different execution times. In order to hold the operands across the step boundary, a latch is needed in every input/output port of every function unit.

Internally, the communication unit is assumed. A bypass route is provided from a function unit's output latch to the input latches of others. In this way if there is a data dependency between two operations, these operations can be scheduled in consecutive control steps.

The communication unit provides the interface between the external communication devices and the processing element. Data transfer operations (read or write) and function unit operations can be performed in parallel during each control step. The communication unit could have several input/output ports operate independently, each interfaces to one communication device. The number of these input/output ports is determined either by the scheduling and partitioning algorithm or a user-given constraint. In either case, it should be no less than the maximum number of simultaneous I/O transfers in the scheduled CDFG.

Communication Architecture

The communication architecture specifies the devices used to link processing elements. A device can be connected to several processing elements. Only one write request can proceed in a communication device at any control step. Several read requests are allowed to proceed at the same control step. Thus, broadcasting is supported in our model. The communication devices are characterized by three properties in our model: the buffering ability; the accessing constraint; and the delay time.

The buffering ability of a communication device is determined by its inherent characteristics. Buses are non-buffered communication devices. Shared memory devices and FIFO queues are buffered communication devices. Concurrency is increased using buffered communication devices, because data packets will not be blocked waiting for the communication resources. However, it is expensive to provide random access communication buffers. The FIFO queues can be used to reduce the cost.

The accessing constraint of a communication device determines the relationship between input sequence and output sequence. For example, the input sequence of a FIFO queue must be the same as the output sequence. The delay time of a communication device is taken to be the worst case delay, which is a constant. The communication delays and costs associated with communication devices are given as parameters in the resource library.

4. The ILP Formulation

Partitioning and scheduling are two tightly coupled problems, which affect each other. Consider the example shown in Fig. 2. Assume the communication time between the processing elements in two control steps. There are five operations to be scheduled. Now, if partitioning is performed first, then the partitions shown in Fig. 2(a) and 2(b) have the same cost. However, after the operations are scheduled as in the figure, we can see that it takes 4 cycles to finish the data flow in Fig. 2(a), while it takes 6 cycles for that in Fig. 2(b) (due to the communication from 02 to 04).

On the other hand, if scheduling is performed first, then assigning 02 to either step 1 or step 3 will result in the same execution time. However, using the partition as shown in Fig. 2(a), it will take 4 cycles to finish the data flow if 04 were assigned to step 3, while it takes 4 cycles if 04 were assigned to step 1.

Figure 2: An example shows the concurrency of computation and communication.

In the above discussion, we can see that it is important to consider both partitioning and scheduling together. The ILP formulation presented in this section solves these two subtasks simultaneously. For the simplicity of presentation, we will only discuss the formulations for global bus and FIFO queue, and consider single cycle operations in the paper. Extensions to multi-cycle operations and other communication schemes are straightforward.

The problem formulation is integer linear programming and optimized by compound objective function. For ease of illustration, we assume each data dependency represents one data transfer. The cost calculating method may overestimate the total number of global communications. Our approach to partitioning and scheduling a given application to a multiprocessor system includes three substeps:

1. ASAP (as soon as possible) estimation: determines the earliest possible execution time for each operation assuming no communication delay nor resource conflicts.
2. ALAP (as late as possible) estimation: determines the latest possible execution time for each operation assuming maximum communication delay in all data transfers and no resource conflicts.
3. ILP formulation: minimizes the cost of the objective function.

ASAP and ALAP estimation can be solved by topological sort. All variables in the ILP formulations have nonnegative values. The definitions used in our paper are as follows.

\( C_{\text{control}} \): the total number of control steps required
\( m \): the number of partition used
\( a \): the number of operations in the CDFG
\( o_{i,j} \): the operation of the CDFG
\( s_{i,p} \): the communication queue
\( k_{t} \): the number of type \( t \) function units in partition \( p \)
\( E_{U} \): the set of operations using type \( t \) function unit
\( C_{t} \): the set of operations using type \( t \) function unit at step \( c \)
\( d_{p_{1},p_{2}} \): the delay between partition \( p_{1} \) and \( p_{2} \)
\( S_{i} \): the earliest possible scheduled time of \( o_{i} \)
\( L_{i} \): the latest possible scheduled time of \( o_{i} \)
\( g_{i,j} \): 1 if the communication for \( o_{i} \) to \( o_{j} \) starts at step \( c \), otherwise, \( g_{i,j} = 0 \)
\( g_{i,j,c} \): 1 if the communication for \( o_{i} \) to \( o_{j} \)
steps at step $c$, otherwise, $g_{i,j,c} = 0$

$B_d$ : the number of inter-partition buses with delay $d$

$M_d$ : the cost of inter-partition bus with delay $d$

$\alpha, \beta$ : the weights of given objective sub-functions

Let $\epsilon$ denote the number of steps used in the ALAP estimation.

According to our architecture, $\epsilon_{p1}$ is equal to zero if $p_1 = p_2$.

The ILP problem can be formulated as

minimizing

$$\alpha \left( \sum_{i} x_{i,p_1,p_2} + \beta C_{step} \right) \quad \forall o_1 \rightarrow o_2, \text{ and } p_1 \neq p_2$$

subject to

$$m \sum_{p=1}^{m} x_{i,p} = 1, \quad \text{for } 1 \leq i \leq n$$

$$m \sum_{p=1}^{m} x_{i,p} y_{p,c} \leq k_{p_1}, \quad \text{for } 1 \leq i \leq n, 1 \leq p \leq m, 1 \leq c \leq s$$

$$o_i \in P_{U^c}$$

$$\sum_{c \in S_i} c \cdot y_{i,c} = 1, \quad \text{for } 1 \leq i \leq n, 1 \leq c \leq s$$

$$\sum_{c \in S_i} c \cdot y_{i,c} - \sum_{c \in S_j} c \cdot y_{j,c} + \sum_{c \in S_k} \sum_{m} d_{p_1,p_2} \cdot x_{i,p_1,p_2,j,p_2} \leq -1$$

$$o_i \in o_j$$

$$\sum_{c \in S_i} c \cdot y_{i,c} - C_{step} \leq 0, \forall o_i \text{ without successors}$$

The objective function in (1) states that we are going to minimize the compound cost of the total number of inter-partition communications and the total number of control steps. The weighted objective function is stated in constraint (2), where operations are only assigned to one partition in the system. Constraint (3) states that the number of operations assigned to partition $p$ and use type $f$ function unit which is greater than $k_{p_1}$. It is clear that $o_i$ can only be scheduled in a step between $S_i$ and $L_i$, which is stated in (4). Constraint (5) ensures that the precedence relations of the CDFG and the communication delays of the relations will be preserved. No operations should be scheduled after $C_{step}$ as described in constraint (6). Note that the logical function AND, $x_1 \cdot x_2 \cdots x_n$, can be replaced by an integer linear function $x_1 + x_2 + \ldots + x_n - n + 1 \leq y$ where the value of variable $y$ is equal to the result of the AND operation.

4.1 Minimizing Number of Buses

In the above formulation, we did not take into account the communication devices explicitly. Suppose the communication devices used are buses. In order to model the bus operations and to minimize the number of buses used, we have to consider more precisely, when a data transfer starts and stops. The modified formulation is thus to

minimizing

$$\alpha \left( \sum_{d} M_d \cdot B_d \right) + \beta C_{step}$$

subject to

$$L_i \sum_{c \in S_i} c \cdot y_{i,c} - \sum_{c \in S_j} c \cdot y_{j,c} \leq 0, \forall o_i \rightarrow o_j$$

$$L_j \sum_{c \in S_j} c \cdot y_{j,c} - \sum_{c \in S_i} c \cdot y_{i,c} + \sum_{m} d_{p_1,p_2} \cdot x_{i,p_1,p_2,j,p_2} \leq -1$$

$$o_i \in o_j$$

$$L_j \sum_{c \in S_j} c \cdot y_{j,c} = 1, \quad \text{for } 1 \leq i, j \leq n, 1 \leq c \leq s$$

$$\sum_{c \in S_i} c \cdot y_{i,c} - B_d \leq 0, \quad \text{for } d \leq c \leq s, 1 \leq i, j \leq n$$

$$\text{if } d_{p_1,p_2} = d, \quad \text{and } o_i \rightarrow o_j$$

In the formulation, we assume that the communications will use the same type of buses with the same delay. The objective function in (1) states that we are going to minimize the total number of control steps and the total cost of buses. Constraint (6.1) states that all communications associated with precedence relations must be scheduled after the step that $o_i$ has been executed. It is clear that any communication must start after the step that $o_i$ has been executed and end before the step that $o_j$ begins to execute, which is reflected in (5.1) and (7.1). The communication associated with $o_i \rightarrow o_j$ can only be scheduled into a step between $S_i$ and $L_j$, as described in constraint (5.1), (7.1) and (8.1). Constraints (3), (4), (5), (6) are the same as those in the previous subsections. Constraint (9.1) states that there are at most $B_d$ communications using the buses with delay $d$ within $d$ consecutive steps. Thus, there is $B_d$ buses of type $d$ used in the system.

4.2 Minimizing Communications for FIFO queues

The constraint of using FIFO queues is that the writing sequence of the source partition to the buffer must be the same as the reading sequence of the destination partition from the buffer. Assume $o_i$ and $o_j$ are connected by $e_{i,j}$. Let $w_{i,j}$ be the time that $o_i$ writes to the buffer and $r_{i,j}$ be the time that $o_j$ reads from the buffer. The pair $(w_{i,j}, r_{i,j})$ is denoted $RWP_{i,j}$. Clearly, the value of $r_{i,j}$ must be greater than that of $w_{i,j}$ for every $RWP_{i,j}$. Two RWP's, $RWP_{i,j}$ and $RWP_{k,l}$, are said to satisfy the Enclosure Property (EP) if $w_{i,j} \leq w_{k,l} \leq r_{i,j} \leq r_{k,l}$. Given a set of RWP's, there exists a legal FIFO accessing sequence, if and only if there is no EP between any two RWP's in the set. The problem can thus be formulated as

minimizing

$$\alpha \left( \sum_{i} x_{i,p_1,p_2} + \beta C_{step} \right) \quad \forall o_1 \rightarrow o_2, \text{ and } p_1 \neq p_2$$

subject to

$$L_j \sum_{c \in S_i} c \cdot g_{i,j,c} = \sum_{c \in S_i} c \cdot g_{i,j,c} + \sum_{m} \sum_{p_1,p_2} x_{i,p_1,p_2,j,p_2} \leq 0$$

$$\forall o_i \rightarrow o_j$$

$$L_j \sum_{c \in S_i} c \cdot g_{i,j,c} \leq L_j \sum_{c \in S_i} c \cdot y_{j,c} \leq -1, \quad \forall o_i \rightarrow o_j$$

$$\sum_{c \in S_j} c \cdot g_{i,j,c} = 1, \quad \text{for } 1 \leq i, j \leq n, 1 \leq c \leq s$$

$$\sum_{c \in S_i} c \cdot g_{i,j,c} - \sum_{c \in S_i} c \cdot g_{i,j,c} + \sum_{c \in S_j} c \cdot g_{i,j,c} \leq 0$$

$$\forall o_i \rightarrow o_j$$

Constraints (2), (3), (4), (5), (6) and (8) are the same as those described in previous subsections. Constraints (7.2) states that the operation of reading from FIFO must be scheduled into a step which is after that of writing to FIFO plus the delay of FIFO. Constraint (11.2) is similar to constraint (8). Constraint (11.2) ensures that there is no FIFO access violations.

5. A Simplified ILP for Heuristics

The exact formulation presented in the previous section is too complex for CDFGs with large number of operations. It is suitable for optimizing basic block. In this section we introduce a simplified ILP to approximate the exact solution for large CDFG's.
In the approach, a list scheduling is performed based on the given resource constraints. Then, we partition the scheduled graph into two such that the resource distribution between them is balanced and the number of inter-partition communications is minimal. Each partition is recursively partitioned into two until a partition is small enough for one processing element. The number of recursions is usually very small. Re-scheduling is performed as a last step by inserting control steps and rearranging operations in order to satisfy the inter-partition communication constraints.

Let $G_s$ be the scheduled CDFG by list scheduling and $E$ be the set of edges of the CDFG. The number of control steps determined by the list scheduling is $s$. Let $O$ denote partition A and 1 denote partition B. The variable $x_i$ indicates which partition $o_i$ is assigned to. For every edge $e_{ij} \in E$, there is a communication between A and B if and only if $x_i \neq x_j$. Thus, we can use boolean algebra $x_i \oplus x_j$ to represent a communication. Our objective function is to

$$\min \sum x_i \oplus x_j$$

subject to

$$\sum x_i \leq k_{it}, \quad 1 \leq i \leq s, \forall t;$$

$$\sum (1-x_i) \leq k_{it}, \forall o_i \in E_{it};$$

The objective function is to minimize the number of communications between partition A and B. Constraints (2) and (3) restrict the number of operations using function unit of type $t$ to be smaller than the specified values. The function $x_i \oplus x_j$ can be replaced by introducing a $0-1$ variable, $y_{ij}$, and adding the two constraints, $x_i - x_j + y_{ij} > 0$, and $x_j - x_i + y_{ij} > 0$. The objective function is changed to $\sum y_{ij}$.

As mentioned, we may overestimate the number of communications because an output may be fed into several operations and requires only one communication. Suppose the operation $o_i$ has fanout operations $o_{j1}, o_{j2}, \ldots, o_{jn}$. We have the summation $\sum y_{ij}$ in the objective function and its corresponding $2k$ constraints. To replace the multiple communications by one broadcast, we can replace the summation and all $y_{ij}$’s of the $2k$ constraints by a single variable $y_{ij}$. It is easy to check that the objective variable is 1 when at least one fanout operation is in a different partition, and 0 otherwise.

Minimizing the Number of Global Buses

Let the time interval between two adjacent control steps $t$ and $t+1$ be $C_{dt}$. Using the write trigger timing scheme, we know the control step for each inter-partition communication. Let the set of candidates of inter-partition communication bound to $C_{dt}$ be $D_{dt}$. Our objective is to minimize the maximum number of inter-partition communications of all the time intervals. The objective function is thus to minimize $\max_{d} (\sum_{e_{ij} \in D_{dt}} (x_i \oplus x_j)) \forall t$. The constraints are the same as described in previous subsection.

Minimizing the Size of FIFO Buffer

Consider the case that there is an EP between the RWP’s of pairs $(o_i, o_j)$ and $(o_m, o_n)$. If $o_i$, $o_m$ are assigned to the same partition and $o_j$, $o_n$ to the other, there is an access violation. To prevent the violation, we add two constraints, $x_i - x_j + x_m - x_n > -2$ and $x_i + x_j - x_m - x_n < 2$.

Critical Signals

The communications which will cause the system to stall are more critical than those which will not. For example, edge $e_{ij}$ in Fig. 2, is more critical than edge $e_{ij}$. We increase the cost of a critical edge in the objective function. Let $N_{ct}$ be the difference of control steps between source and destination operations of edge $e_{ij}$. In the objective function, the cost of the edge $e_{ij}$ is $\frac{N_{ct}}{\text{Cost}} \times (x_i \oplus x_j)$, where $\text{Cost}$ is a constant.

6. Iterative Partitioning

In this section we introduce an iterative partitioning heuristic. The iterative heuristic is used for very large CDFGs such as a program, which are too complex to solve using ILP. A list scheduling is done before applying the algorithm. The algorithm consists of two phases: initial solution phase and iterative improvement phase.

Initial Solution Phase

We view a CDFG as a set of $n$ operations, $o_1, o_2, \ldots, o_n$, connected by a set of $s$ signals, $s_{ij}, s_{ij}, \ldots, s_{ij}$. Any two operations which share at least one signal are said to be neighbors. Given a partition of the operations, a signal is said to be in cut state if it incides to at least one operation in each partition and vacate state otherwise. A signal is said to be in stall state if it is in cut state and we have to insert at least one control step in the original schedule in order to communicate the operations in different partitions, and the stall state otherwise. Define the cutset of the partition be the set of signals which are in cut state and the stallset to be the set of signals which are in stall state. Our goal is to partition the CDFG into two, A and B, such that the resource constraints are satisfied, and the resulting cutset and stallset are minimized. The cost function is $x_{ij} || cutset || + \beta || stallset ||$.

We first find two paths with very high resource utilization and low global communications, assign them to different partitions, and assign the remaining operations to one of the two partitions according to their cost in a greedy manner. Within these two paths, the first path is a critical path. The second path is the most parallel one with respect to the first path.

Iterative Improvement Phase

We use a modified min-cut partitioning algorithm [10] to improve the initial solution. The operation with the largest gain is selected for the next move. The process is repeated until it reaches a stage which no move leads to an improvement.

We define the gain $g_i$ of $o_i$ as the change of the number of signals in the cutset or stallset if $o_i$ is moved from its current partition to its complementary partition. We define $SC(s_{ij})$ be the set of operations associated with $s_{ij}$ if its source operation and sink operations are scheduled in the adjacent steps. If it is not empty, the source operation is added to the set. For example (Fig. 3), $SC(s_{ij})=(o_2, o_3, \ldots)$ and $SC(s_{ij})=\emptyset$.

![Figure 3: An example of the SC set of signals](image-url)

Let $F$ ("From") be the current partition of $o_i$ and $T$ ("To") be its complementary partition. $FS(o_i)$ is the number of signals which incident to $o_i$ and have the same operation in the F side, and $TE(o_i)$ is the number of signals which incident to $o_i$ and have no operation in the T side. $SFS(o_i)$ and $STE(o_i)$ are defined similar to $FS(o_i)$ and $TE(o_i)$ except they are defined on the SC sets of the signals which incident to $o_i$. The gain of $o_i$ is then computed by $g_i = \alpha \left( FS(o_i) - TE(o_i) \right) + \beta \left( SFS(o_i) - STE(o_i) \right)$.

7. An Example of Test Vehicles

We have implemented the proposed algorithm in C on a VAX-8550. The ILP formulation is solved by LINDO also on the same machine. We have tested our algorithm on several benchmarks. One of them is the fifth order digital elliptical wave filter. The final partitions and scheduling are shown in Fig. 4. We assume 1 cycle delay for an addition and 2 cycle delay for a multiplication. The operation unit is inter-processor communication. The total number of inter-processor communications and the critical signals are shown in Table 1, and a graphical representation of the partition and schedule is shown in Fig. 5(a). We compare our result with that of APARTY. We use STAR [11] to allocate the data parts of these two processors and the unperturbed number of control steps used to execute the unperturbed CDFG.
The cycle time of unpartitioned processor is longer than that of partitioned processors because of the long buses. To test a larger example, we have unrolled the elliptical filter twice. The number of control steps of the unrolled filter is 35 and the number of operations is 68. Fig. 5(b) shows the result of the unrolled filter after partitioning. The run time of using ILP is less than 15 seconds while it takes only 0.14 seconds using the heuristic algorithm. Both achieve the optimal solution. Table 2 shows the final synthesis result of processor 1 and 2 of the resultant multiprocessor system. The processor synthesized by traditional method is also shown in the table. Table 3 lists the run time of the algorithms for the above examples. The results of the iterative algorithm under different parameters are shown in Table 4.

### Table 1: A summary of solution of elliptical filter

<table>
<thead>
<tr>
<th>No. of Communications</th>
<th>ASPERTY</th>
<th>MULTIPAR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 2: A summary of synthesis result

<table>
<thead>
<tr>
<th></th>
<th>ASPERTY</th>
<th>MULTIPAR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 3: The run time of the algorithms

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Method</th>
<th>CPU seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unpartitioned Filter</td>
<td>ILP</td>
<td>0.84</td>
</tr>
<tr>
<td>Partitioned Filter</td>
<td>ILP</td>
<td>0.14</td>
</tr>
</tbody>
</table>

### Table 4: The results of the mincut algorithm

<table>
<thead>
<tr>
<th>n</th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

8. Conclusion and Future Work

We have proposed new scheduling and partitioning algorithm for synthesizing application-specific multiprocessor system. The communication schemes are considered in our work so that the synthesis of a multiprocessor or multi-chip system can be done accurately. Throughput is obtained without sacrificing the cycle time of a processor. The algorithm can be easily applied to other architectures. For conditional branches, the sum of probabilities associated with each selection path of any conditional branch is equal to 1. Let the probability of selection path i of a conditional branch be denoted as \( p_i \). We can derive the probability that the data transfer of inter-partition communication is activated. The objective function is changed to \( \sum w_{ij} \cdot (x_i \oplus x_j) \) where \( w_{ij} \) is the probability that the communication represented by this edge might happen. For loops, let the number of iterations of the loop be \( \beta \). We can write the objective function of such an edge as \( \beta(x_i \oplus x_j) \) while all the constraints are not changed. In future research, more concise considerations must be developed for loops and conditional branches.

References