Optimum Resource Sharing in Pipeline Synthesis

Kazutoshi YOKOYAMA, Shin'ichi WAKABAYASHI, Jun'ichi MIYAO, *and* Noriyoshi YOSHIDA

Faculty of Engineering, Hiroshima University
Shitami, Saijo-cho, Higashi-Hiroshima, 724 JAPAN

Abstract

Conditional resource (module) sharing is one of the procedures performed during behavioral synthesis of pipeline design. Conditional resource sharing is to share resources among mutually exclusive parts of conditional branches to reduce the layout area of a circuit. In this paper, we assume that the behavioral description of a circuit to be synthesized is given by a data flow graph. First, we formulate the problem, and a set of subproblems of conditional resource sharing is given. Next, we define a module sharing graph to determine module sharing, and give algorithms to solve the subproblems optimally in polynomial time.

1 Introduction

In automatic pipeline synthesis of a circuit, a data flow graph, which gives a behavioral description of a circuit, is given as input, and a synthesis algorithm produces a pipeline data path which can realize the behavior given by the data flow graph [2]. Pipeline synthesis consists of following two tasks: allocation of hardware resources (sometimes called *modules*) for operations in the data flow graph, and determination of the pipeline stages. In designing a pipeline data path, two objective functions must be minimized, one is the total layout area, and the other is the number of pipeline stages. In order to minimize the layout area, a synthesis algorithm tries to share hardware resources. However, sharing resources may increase the number of stages, that may cause performance degradation.

In general, it is possible to share hardware resources which are not used simultaneously. There are two types of resource sharing, one is *unconditional resource sharing* that shares hardware resources among disjoint time steps, and the other is *conditional resource sharing* that shares resources among *mutually exclusive* parts of conditional branches [1]. In pipeline design, it is possible to share hardware resources only among mutually exclusive parts of conditional branches. In this paper, we discuss conditional resource sharing problems in pipeline synthesis, and propose new algorithms to solve them.

For the conditional resource sharing problem in pipeline synthesis, an algorithm is proposed in [1]. However, since the algorithm described in [1] is heuristic and its behavior is controlled by user-defined parameters, the optimum solution may not be obtained. We discuss the subproblems of conditional resource sharing which can be solved optimally, and show new algorithms to solve them.

This paper is organized as follows: In Section 2, basic terminologies are defined. We present the formulation of the pipeline data path synthesis as a *conditional resource(module) sharing problem* in Section 3. Given this formulation, we show an essential property of module sharing, and introduce an important data structure *Module Sharing Graph*, in Section 4. We also show new algorithms for conditional module sharing. In Section 5, we present conclusions.

2 Preliminaries

2.1 Data Flow Graph

We assume that the behavioral description of a circuit to be synthesized is given as a *data flow graph* with a mapping which specifies what operations are to be performed in the circuit.

[Definition 1] A *data flow graph* $DFG = (V, E, op)$ is a directed acyclic graph, where $V = \{v_1, v_2, \ldots, v_p\} \cup \{v_I, v_O\}$ is a set of nodes and $E = \{e_1, e_2, \ldots, e_q\} \subseteq V \times V$ is a set of directed edges satisfying the following three conditions:

(1) The *indegree* of v_I = the *outdegree* of $v_O = 0$. We call v_I and v_O the *input port* and the *output port*, respectively.

(2) For every node $v_i (1 \leq i \leq p)$, there exists a directed path from v_I to v_O via v_i.

(3) $op : V \rightarrow OP$ is a mapping from V to OP, where OP is a predefined set of arithmetic/logic operators such as $+$, $-$, $\wedge$ (logical-AND), $\vee$ (logical-OR), etc. , which are allowed to be used in the circuit description. We call the set OP the *operator repertory*. □

The *semantics* of a given data flow graph $DFG = (V, E, op)$ is as follows: Assume that a node v_i has s incoming edges and t outgoing edges. This means that the operation $op(v_i)$ is applied to s operands, which are given from its neighboring nodes, and the result is sent to t neighboring nodes. For the precise interpretation of the circuit description, some ordering of incoming edges of a node is assumed. Note that the data flow graph presented here specifies a *data path* of the circuit and does not specifies its control flow. We assume that the control flow of a circuit is described in some other way.

We introduce a partial order among nodes of the DFG to formulate the execution ordering of operations of DFG.

[Definition 2] Let $V = \{v_1, v_2, \ldots, v_p\} \cup \{v_I, v_O\}$ be the set of nodes of a given DFG. For every pair of nodes, u and v, in V, u *precedes* v if there is an edge from u to v, denoted by $u \succ v$. The transitive closure of $\succ$ is denoted by $\succ^+$, and the reflexive and transitive closure is denoted by $\succ^*$. □

[Definition 3] A data flow graph $DFG = (V, E, op)$ is said to be *serial* if there exists a simple directed path from v_I to v_O via all nodes except v_I and v_O in V. □

2.2 Mutually Exclusive Data Flow Graph

As mentioned in the Introduction, it is possible to reduce the layout area of *pipelined* data path by sharing hardware resources (i.e., modules) among mutually exclusive parts of conditional branches in the behavioral description of a circuit. To simplify the problem of conditional resource sharing, in this paper, we only consider the problem of resource sharing between two mutually exclusive parts of behavioral description of a circuit. To formulate this problem, we introduce an extension of the data flow graph, called a *mutually exclusive data flow graph*.

[Definition 4] A *mutually exclusive data flow graph* $MED = (TB, FB)$ is a pair of two data flow graphs $TB = (TV, TE, op)$ and $FB = (FV, FE, op)$, called a *true block* and a *false block*, respectively. $TV = \{t_1, t_2, \ldots, t_m\} \cup \{t_I, t_O\}$ is a set of nodes of TB, and $TE = \{te_1, te_2, \ldots, te_r\} \subseteq TV \times TV$ is a set of edges of TB. t_I and t_O are called the input and output ports of TB, respectively. For the false block FB, $FV = \{f_1, f_2, \ldots, f_n\} \cup \{f_I, f_O\}$, $FE = \{fe_1, fe_2, \ldots, fe_s\} \subseteq FV \times FV$, f_I and f_O are similarly defined. TB and FB adopt a common operator mapping function $op : (TV \cup FV) \rightarrow OP$, and share an operator repertory OP. □

The semantics of a given mutually exclusive DFG $MED = (TB, FB)$ seems to be obvious and natural. According to the evaluation of a predefined condition, the behavior of a circuit specified by the true block TB is realized when the condition is *true*, and the behavior specified by FB is realized when otherwise. We assume that the evaluation of a given condition is performed outside the circuit described by the mutually exclusive DFG.

[Example 1] An example of mutually exclusive DFG is shown in Figure 1. For convenience, operator symbols and variables' names are attached to the corresponding nodes and edges, respectively. In the figure, there are 5 operators and 10 variables in TB, and 4 operators and 8 variables in FB. □

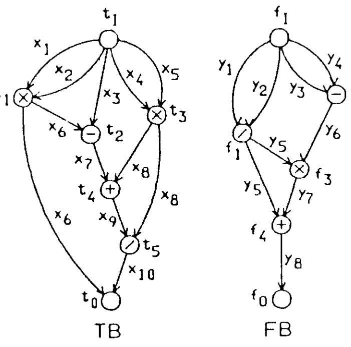

Figure 1: An example of MED.

2.3 Hardware Implementation of Operators

Given a data flow graph $DFG = (V, E, op)$, operations to be realized by hardware are specified by the mapping *op*. However, in general, there are several ways to implement an arithmetic/logic operation specified by *op*. We introduce a ***module repertory*** and two mappings to formulate this problem.

[Definition 5] Let $MO = \{mo_1, mo_2, \ldots, mo_h\}$ be a set of hardware modules, which may be used to realize operations given by a DFG. We call the set MO a *module repertory*. Let $func : OP \rightarrow 2^{MO}$ be a mapping from a given operator repertory to the power set of MO. For each $mo \in MO$, $func(mo) = \{op_{i1}, op_{i2}, \ldots, op_{i\gamma}\} \subseteq OP$ means that the module mo realizes the function of operators $op_{i1}, op_{i2}, \ldots,$ $op_{i\gamma}$. We represent the layout area and delay time of the module mo as $area(mo)$ and $delay(mo)$, which are assumed to be natural numbers. □

[Example 2] Figure 2 shows an example of module repertory. □

	func (mo)	area (mo)	delay (mo)
mo_1	+, −	3	2
mo_2	×	5	3
mo_3	/	5	3

Figure 2: An example of module repertory.

3 Pipeline Synthesis with Conditional Resource Sharing

3.1 Formulation of the Problem

We formulate the problem of conditional resource sharing between two mutually exclusive parts of a circuit described by mutually exclusive DFG.

[Definition 6] For a given mutually exclusive DFG $MED = (TB, FB)$, a *pipeline module allocation* $PMA = (select, assign, stage)$ is a set of three mappings, *select*, *assign*, and *stage*, defined as follows.

(1) $select : V \rightarrow MO$ is a mapping to specify which types of modules are used to realize operators associated with nodes. Here, V is a set of nodes in MED, and MO is a module repertory. For simplicity, we assume that MO contains two special modules, mo_I and mo_O, and the input and output ports of MED (t_I, t_O, f_I, f_O) are always mapped to these special modules, and none of other nodes in MED may be mapped to them. mo_I and mo_O are called an ***input pad*** and an ***output pad***, respectively. We also assume that $area(mo_I) = area(mo_O) = delay(mo_I) = delay(mo_O) = 0$.

(2) $assign : V \rightarrow \{0, 1, \ldots, num\_parts, num\_parts + 1\}$ is a mapping to specify which modules are assigned to nodes. We assume that $assign$ is surjective, and $assign(t_I) = assign(f_I) = 0$, and $assign(t_O) = assign(f_O) = num\_parts + 1$. For $v \in V$, $assign(v)$ is called the ***parts number*** of v, and means that the operation associated with v $(op(v))$ is realized by a module whose module type and parts number are $select(v)$ and $assign(v)$, respectively. Let represent a set of nodes which are mapped to a parts number r as $share(r)$. Furthermore, for $v \in V$, let $parts(k) = mo_\gamma$ if $select(v) = mo_\gamma$ and $assign(v) = k$.

(3) $stage : V \rightarrow \{0, 1, \ldots, num\_stage, num\_stage + 1\}$ is a mapping to specify the pipeline stage assignment of each operation. We assume that $stage$ is surjective, and $stage(t_I) = stage(f_I) = 0$, $stage(t_O) = stage(f_O) = num\_stage + 1$. □

[Definition 7] For a given mutually exclusive DFG $MED = (TB, FB)$ and a positive constant msd, a pipeline module allocation $PMA = (select, assign, stage)$ is said to be *feasible* if the following conditions hold.

(1) For all $v \in V$, $op(v) \in func(select(v))$.

(2) For all u and v in V, if $assign(u) = assign(v)$ then $select(u) = select(v)$.

(3) For all u and v in V, if $u \succ^* v$ then $stage(u) \leq stage(v)$.

(4) For all u and v in V such that $stage(u) = stage(v)$ and $u \succ^+ v$, $\sum_{i=1}^{\gamma} delay(select(u_i)) \leq msd$ holds for every simple path from u to v, i.e., $u = u_1, u_2, \ldots, u_\gamma = v$. We call msd the *maximum stage delay*.

(5) For all $i (1 \leq i \leq num\_parts)$, $|share(i)| \leq 2$.

(6) For all i such that $|share(i)| = 2$, let $share(i) =$

$\{u, v\}$. Then u is a node of TB and v is a node of FB, or vice versa, and $stage(u) = stage(v)$. □

For a given $PMA = (select, assign, stage)$ for $MED = (TB, FB)$, the ***total area*** and the ***delay*** of a pipeline data path synthesized by PMA are represented as follows.

$area(PMA) = \sum_{i=1}^{num\_parts} area(parts(i))$,
$delay(PMA) = num\_stage * msd$.

[Example 3] Figure 3 shows a feasible PMA for MED given in Figure 1, the module repertory given in Figure 2, and $msd = 5$. In the figure, $share(5) = \{t_3, f_3\}$, $share(6) = \{t_4, f_4\}$, and $area(PMA) = 27$, $delay(PMA) = 15$, respectively. □

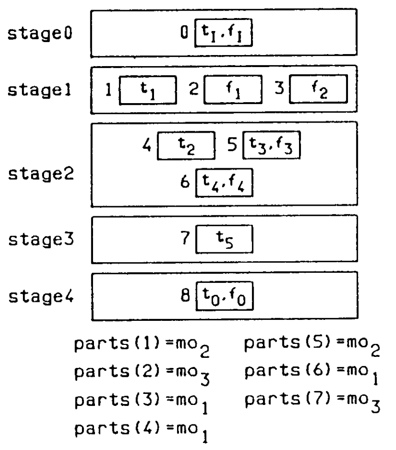

Figure 3: A feasible PMA.

[*Conditional Resource Sharing*]
Given a mutually exclusive DFG $MED = (TB, FB)$ and a set of conditions on the total area and the delay time of pipeline data path, find a feasible pipeline module allocation $PMA = (select, assign, stage)$ such that the pipeline data path induced by PMA satisfies the given set of conditions. □

3.2 Assumptions

We introduce some assumptions into the problem formulation stated in 3.1 to derive efficient algorithms.

[*Assumptions*]

(A1) Given a mutually exclusive $DFG = (TB, FB)$, both TB and FB are serial.

(A2) For every module $m_i (1 \leq i \leq h)$, $delay(m_i) = 1$.

(A3) $msd = 1$. □

3.3 Subproblems

Under the assumptions described in 3.2, we formulate three subproblems of conditional resource sharing.

[*Conditional resource sharing problem (the problem CMS)*]
Given a mutually exclusive DFG $MED = (TB, FB)$, find a pipeline module allocation $PMA = (select, assign, stage)$ such that $area(PMA)$ is minimum. □

[*Stage constrained conditional resource sharing problem (the problem SCMS)*]
Given a mutually exclusive DFG $MED = (TB, FB)$ and a positive integer k, find a pipeline module allocation $PMA = (select, assign, stage)$ such that $num\_stage \leq k$ and $area(PMA)$ is minimum □

[*Area constrained conditional module sharing problem (the problem ACMS)*]
Given a mutually exclusive DFG $MED = (TB, FB)$ and a positive integer k, find a pipeline module allocation $PMA = (select, assign, stage)$ such that $area(PMA) \leq k$ and num_stage is minimum. □

4 Allocation Algorithms

4.1 Allocation Pair

To solve the problems stated in 3., we introduce a set of some useful concepts on module sharing.

[Definition 8] For a given $MED = (TB, FB)$, a pair of nodes $u \in TV$ and $v \in FV$ is said to be an *allocation pair* if there exists a module mo such that $op(u) \in func(mo)$ and $op(v) \in func(mo)$. We denote an allocation pair of u and v by $[u, v]$. □

[Definition 9] For a given $MED = (TB, FB)$ and two allocation pairs on MED, $p_1 = [u_1, v_1]$ and $p_2 = [u_2, v_2]$, p_1 and p_2 are said to be ***twisted each other*** if either one of the following conditions holds.
(1) $u_1 \succ^+ u_2$ and $v_2 \succ^+ v_1$,
(2) $u_2 \succ^+ u_1$ and $v_1 \succ^+ v_2$. □

The following lemma can easily be derived from the above two definitions.

[Lemma 1] Given a feasible $PMA = (select, assign, stage)$ for $MED = (TB, FB)$, no two allocation pairs in MED, which are twisted each other, share modules in PMA. □

[Definition 10] Given a mutually exclusive DFG $MED = (TB, FB)$, a set of allocation pairs $S = \{[t_{s1}, f_{s1}], [t_{s2}, f_{s2}], \cdots, [t_{s\gamma}, f_{s\gamma}]\}$ is said to be a *feasible sharing* if the following conditions hold.

(1) No node in MED appears more than once in S.

(2) No two allocation pairs are twisted each other. □

It is obvious that, if a feasible sharing S is given, a feasible PMA is easily obtained. For each allocation pair in S, a module with minimum area, which realizes two functions specified by two nodes in the allocation pair, is selected and assigned. For the remaining nodes in MED, a module with minimum area is selected and assigned, which is not shared by some other node. We denote PMA constructed from a given feasible sharing in the above manner by $PMA(S)$. We also denote the number of stages given by $PMA(S)$ by $num\_stage(S)$. Note that, when $S = \phi$, the resulting $PMA(\phi)$ achieves the minimum delay.

[Definition 11] Given a mutually exclusive DFG $MED = (TB, FB)$, for every $v \in TV \cup FV$, let $lv(v)$ be the length of the longest path from t_I or f_I. □

The following lemma states an important property on feasible sharing.

[Lemma 2] Given a mutually exclusive DFG $MED = (TB, FB)$, let $S = \{[t_I, f_I], [t_{s1}, f_{s1}], [t_{s2}, f_{s2}], \cdots, [t_{s\gamma}, f_{s\gamma}]\} (\gamma \geq 0)$ be a feasible sharing for $MED = (TB, FB)$ such that $t_I \succ^+ t_{s1} \succ^+ t_{s2} \succ^+ \cdots \succ^+ t_{s\gamma}$, $f_I \succ^+ f_{s1} \succ^+ f_{s2} \succ^+ \cdots \succ^+ f_{s\gamma}$. Let $[t, f]$ be an allocation pair for MED such that $t_{s\gamma} \succ^+ t$ and $f_{s\gamma} \succ^+ f$. Let $S' = S \cup \{[t, f]\}$ and $\Delta stage = num\_stage(S') - num\_stage(S)$. Then the following holds.

(1) In case of $|TV| - lv(t_{s\gamma}) \geq |FV| - lv(f_{s\gamma})$ and $|TV| - |FV| \geq lv(t) - lv(f) \geq lv(t_{s\gamma}) - lv(f_{s\gamma})$,
$\Delta stage = 0$.

(2) In case of $|TV| - lv(t_{s\gamma}) \geq |FV| - lv(f_{s\gamma})$ and $lv(t) - lv(t_{s\gamma}) < lv(f) - lv(f_{s\gamma})$ and $|TV| - lv(t) \geq |FV| - lv(f)$,
$\Delta stage = lv(t_{s\gamma}) - lv(f_{s\gamma}) - lv(t) + lv(f)$.

(3) In case of $|TV| - lv(t_{s\gamma}) \geq |FV| - lv(f_{s\gamma})$ and $lv(t) - lv(t_{s\gamma}) \geq lv(f) - lv(f_{s\gamma})$ and $|TV| - lv(t) < |FV| - lv(f)$,
$\Delta stage = lv(t) - lv(f) - lv(t_{s\gamma}) + lv(f_{s\gamma}) + |FV| - |TV|$.

(4) In case of $|TV| - lv(t_{s\gamma}) < |FV| - lv(f_{s\gamma})$ and $lv(t) - lv(t_{s\gamma}) < lv(f) - lv(f_{s\gamma})$ and $|TV| - lv(t) \geq |FV| - lv(f)$,
$\Delta stage = lv(f) - lv(t) + |TV| - |FV|$.

(5) In case of $|TV| - lv(t_{s\gamma}) < |FV| - lv(f_{s\gamma})$ and $lv(t) - lv(t_{s\gamma}) \geq lv(f) - lv(f_{s\gamma})$ and $|TV| - lv(t) < |FV| - lv(f)$,
$\Delta stage = lv(t) - lv(f) - lv(t_{s\gamma}) + lv(f_{s\gamma})$.

(6) In case of $|TV| - lv(t_{s\gamma}) < |FV| - lv(f_{s\gamma})$ and $|TV| - |FV| < lv(t) - lv(f) < lv(t_{s\gamma}) - lv(f_{s\gamma})$,
$\Delta stage = 0$.

(*Proof*) Let $PMA(S) = (select, assign, stage)$. Since the MED is serial,

$$num\_stage(S) = stage(t_{s\gamma}) + max\{|TV| - lv(t_{s\gamma}), |FV| - lv(f_{s\gamma})\}.$$

On the other hand, let $PMA(S') = (select', assign', stage')$. Then

$$stage'(t) = stage(t_{s\gamma}) + max\{lv(t) - lv(t_{s\gamma}), lv(f) - lv(f_{s\gamma})\},$$

and

$$num\_stage(S') = stage'(t) + max\{|TV| - lv(t), |FV| - lv(f)\}$$

Therefore,

$$\begin{aligned}\Delta stage &= num\_stage(S') - num\_stage(S)\\ &= max\{lv(t) - lv(t_{s\gamma}), lv(f) - lv(f_{s\gamma})\}\\ &\quad + max\{|TV| - lv(t), |FV| - lv(f)\}\\ &\quad - max\{|TV| - lv(t_{s\gamma}), |FV| - lv(f_{s\gamma})\}.\end{aligned}$$

By the expansion of three max functions, we get the Lemma. (Q.E.D)

4.2 Module Sharing Graph

We introduce a directed graph, which is an important data structure to solve the module sharing problems.

[Definition 12] Given a mutually exclusive DFG $MED = (TB, FB)$, a *module sharing graph* $MSG = (P, Q, w, d)$ is a directed graph defined as follows.

(1) $P = \{p_1, p_2, \ldots, p_\alpha\} \cup \{p_s, p_t\}$ is a set of nodes, in which each node $p_i (1 \leq i \leq \alpha)$ is corresponding to an allocation pair of MED. p_s and p_t are corresponding to $[t_I, f_I]$ and $[t_O, f_O]$, respectively.

(2) $Q = \{q_1, q_2, \cdots, q_\beta\} \cup \{(p_s, p_i), (p_i, p_t) | 1 \leq i \leq \alpha\} \cup \{(p_s, p_t)\} \subseteq P \times P$ is a set of directed edges, in which each edge $q_i = (p_j, p_k)(1 \leq i \leq \beta)$ is defined only when two allocation pairs corresponding to p_j and p_k are ***not*** twisted each other, and p_j and p_k share no node in MED.

(3) $w : P \rightarrow N$ is a mapping to specify how large the area is reduced when an allocation pair shares a module. Precisely speaking, for a node p, which is corresponding to an allocation pair$[t, f]$, $w(p)$ is defined as $area(mo_t) + area(mo_f) - area(mo_{tf})$ where mo_t and mo_f are modules which realize $op(t)$ and $op(f)$ with minimum area, respectively, and mo_{tf} is a module which realize both $op(t)$ and $op(f)$ with minimum area.

(4) $d : Q \rightarrow N$ is a mapping to specify how much the delay time is increased when an allocation pair is adopted. Precisely speaking, for edge $q_i = (p_j, p_k)$ where p_j and p_k are corresponding to $[t_a, f_b]$ and $[t_c, f_d]$, let $d(q_i)$ be $\Delta stage = num\_stage(\{[t_a, f_b], [t_c, f_d]\}) - num\_stage(\{[t_a, f_b]\})$. For remaining edges, let $d(q) = 0$. □

[Example 4] Given a MED in Figure 4, module repertory in Figure 5, MSG for MED is shown in Figure 6. □

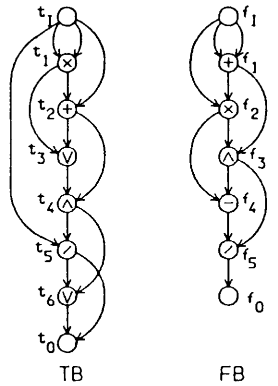

Figure 4: A serial MED.

	func(mo)	area(mo)	delay(mo)
mo_1	+, −	3	1
mo_2	×	5	1
mo_3	/	5	1
mo_4	∧	2	1
mo_5	∨	2	1

Figure 5: A module repertory.

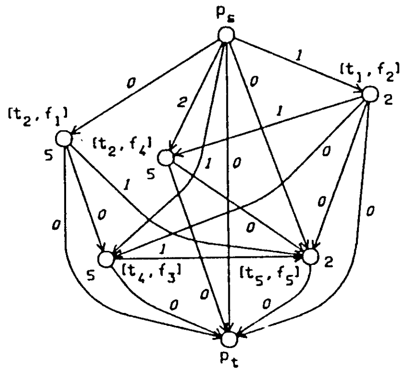

Figure 6: A MSG for MED.

From the construction of MSG, we can show the following lemma.

[Lemma 3] A module sharing graph is acyclic. □

The following is a main theorem in this paper.

[Theorem 1] Given a mutually exclusive DFG $MED = (TB, FB)$ and a module sharing graph $MSG = (P, Q, w, d)$ for MED, there exists a pipeline module allocation $PMA = (select, assign, stage)$ such that

$area(PMA) = area(PMA(\phi)) - w_0$ and

$delay(PMA) = delay(PMA(\phi)) + d_0$

if and only if there exists a simple directed path S from p_s to p_t in MSG such that

$w_0 = \sum_{i=0}^{\gamma} w(p_{si})$ and $d_0 = \sum_{i=0}^{\gamma-1} d((p_{si}, p_{si+1}))$

where $S = p_{s0}(= p_s), p_{s1}, p_{s2}, \ldots, p_\gamma(= p_t)$.

(***Proof***) IF part: From the definition of MSG, we know that a sequence of allocation pairs, which is corresponding to the path S, is a feasible sharing of MED. Consider $PMA(S)$. Then we have

$area(PMA(S)) = area(PMA(\phi)) - w_0$,

$delay(PMA(S)) = delay(PMA(\phi)) + d_0$.

ONLY IF part: Let S' be a set of node pairs, each of which shares a module in PMA. It is obvious that, in MSG, there exists a path which is corresponding to S'. (Q.E.D)

4.3 An Algorithm for the Problem CMS

We present an algorithm for the problem CMS, called *algorithm ACMS.* In order to minimize the total area of the data path, $area(PMA)$, w_0 must be maximized. From Theorem 1, we can get the solution by finding the path S on MSG such that $\sum_{i=0}^{\gamma} w(p_{si})$ is maximum, where $S = p_{s0}(= p_s), p_{s1}, p_{s2}, \ldots, p_{s\gamma}(= p_t)$. The following shows the outline of the algorithm.

```
[Algorithm CMS]
step1: Construction of $MSG = (P, Q, w, d)$.
step2: Topological sort of $P$.
    {Let $p_{t0}(= p_s), p_{t1}, p_{t2}, \ldots, p_{t\alpha}, p_{t\alpha+1}(= p_t)$ be
    the result of topological sort.}
step3: {Find the path $S$ such that $\sum_{i=0}^{\gamma} w(p_i)$ is
maximum.}
for $i \leftarrow 0$ to $\alpha + 1$ do   $area(p_{ti}) \leftarrow 0$;
for $i \leftarrow 0$ to $\alpha + 1$ do
    for all $j$ such that $(p_{tj}, p_{ti}) \in Q$ do
        $area(p_{ti}) \leftarrow max\{area(p_{ti}), area(p_{tj}) + w(p_{ti})\}$;
```

Next, we discuss the time complexity of the algorithm. Let $|TV| = m$ and $|FV| = n$. Since $|P| = O(mn)$, and $|Q| = O(m^2n^2)$, time complexity of step1 is $O(m^2n^2)$. step2 uses topological sorting and step3 uses a longest path algorithm on MSG, and hence time complexity of step2 and step3 are both $O(m^2n^2)$.

[Example 5] Consider MED in Figure 4, the module repertory in Figure 5, and MSG for MED in Figure 6. The algorithm finds the path $p_s, [t_1, f_2], [t_2, f_4], [t_5, f_5], p_t$, and a feasible sharing $S = \{[t_1, f_2], [t_2, f_4], [t_5, f_5]\}$ is obtained. Figure 7 shows the obtained $PMA(S)$. In this case, $area(PMA(S)) = 29$, $delay(PMA(S)) = 8$, respectively. □

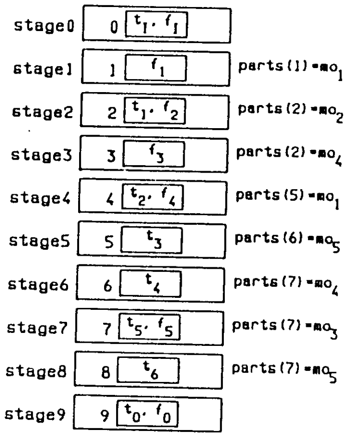

Figure 7: A feasible PMA in Example 5.

4.4 An Algorithm for the Problem SCMS

We will show an algorithm for the problem SCMS, called *algorithm ASCMS.* To solve the problem SCMS, we must find the path S on MSG such that $\sum_{i=0}^{\gamma-1} d((p_i, p_{i+1})) \leq k$ and $\sum_{i=0}^{\gamma} w(p_i)$ is maximum, where $S = p_{s0}(=p_s), p_{s1}, p_{s2}, \ldots, p_{s\gamma}(=p_t)$. The following shows the outline of *algorithm ASCMS.* In the algorithm, step3 uses the procedure ***weight-constrained longest path (WCL)*** on a directed acyclic graph. Procedure WCL determines a longest path from p_s to $p_{ti}(\in P)$ under a given stage constraint. For each $p_{ti} \in P$, $area(p_{ti}, j)$ denotes the path S' such that $\sum_{l=0}^{\beta-1} d((p_{s'l}, p_{s'l+1})) = j$ and $\sum_{l=1}^{\beta} w(p_{s'l})$ is maximum, where $S' = p_{s'0}(= p_s), p_{s'1}, p_{s'2}, \ldots, p_{s'\beta}(= p_{ti})$.

```
[Algorithm ASCMS]
step1: Construction of $MED = (P, Q, w, d)$.
step2: Topological sort of $P$.
    {Let $p_{t0} = (p_s), p_{t1}, p_{t2}, \ldots, p_{t\alpha}, p_{t\alpha+1}(= p_t)$ be
    the result of topological sort.}
step3:
[Procedure WCL]
for $i \leftarrow 0$ to $\alpha + 1$ do
    for $j \leftarrow 0$ to $k$ do   $area(p_{ti}, j) \leftarrow 0$;
for $i \leftarrow 0$ to $\alpha + 1$ do
    for all $j$ such that $(p_{tj}, p_{ti}) \in Q$ do
        for $h \leftarrow 0$ to $k - d((p_{tj}, p_{ti}))$ do
        $area(p_i, h + d(p_{tj}, p_{ti})) \leftarrow$
                        $max\{area(p_{ti}, h + d(p_{tj}, p_{ti})),$
                              $area(p_{tj}, h) + w(p_{ti})\}$;
```

Time complexity of step1 and step2 are same as algorithm ACMS. Time complexity of step3 is $O(m^2n^2)$ under the assumption that k is a constant.

[Example 6] Consider MED in Figure 4, the module repertory in Figure 5, MSG for MED in Figure 6, and $k = 7$. The algorithm finds the path $p_s, [t_1, f_2], [t_5, f_5], p_t$, and gets a feasible sharing $S = \{[t_1, f_2], [t_3, f_4]\}$. $delay(PMA(S))$=7 satisfies the condition

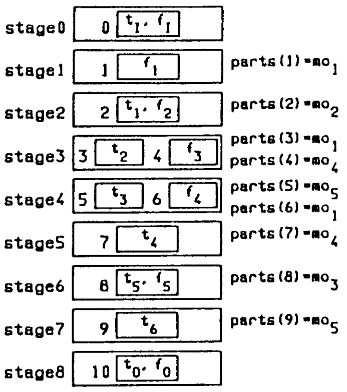

Figure 8: A feasible PMA in Example 6.

delay $(PMA(S)) \leq 7$. Note that the feasible sharing obtained in Example 5 does not satisfy the stage constraint. Figure 8 shows the obtained $PMA(S)$. □

4.5 An Algorithm for the Problem ACMS

We present an algorithm for the problem ACMS, called *algorithm AACMS*. We can solve the problem ACMS in the same manner of the problem SCMS. The difference between them is that step3 uses the procedure ***weight-constrained shortest path (WCS)***. In *WCS*, $stage(p_{ti}, j)$ denote the path S' such that $\sum_{l=0}^{\beta-1} w(p_{s'l}) = j$ and $\sum_{l=0}^{\beta} d((p_{s'l}, p_{s'+1}))$ is minimum, where $S' = p_{s'0}(= p_s), p_{s'1}, \cdots, p_{s'\beta}(= p_{ti})$. The following shows the outline of the algorithm.

```
[Algorithm AACMS]
step1: Construction of $MED = (P, Q, w, d)$.
step2: Topological sort of $P$.
  {Let $p_{t0}(= p_s), p_{t1}, p_{t2}, \cdots, p_{t\alpha}, p_{t\alpha+1}(= p_t)$ be
  the result of topological sort.}
step3:
[Procedure WCS]
for $i \leftarrow 0$ to $\alpha + 1$ do
    for $j \leftarrow 0$ to $k$ do   $stage(p_{ti}, j) \leftarrow 0$;
for $i \leftarrow 0$ to $\alpha + 1$ do
  for all $j$ such that $(p_{tj}, p_{ti}) \in Q$ do
      for $h \leftarrow 0$ to $area(PMA(\phi))$ do
      $stage(p_i, h + w(p_{ti})) \leftarrow$
                      $min\{stage(p_{ti}, h + w(p_{ti})),$
                      $stage(p_{tj}, h) + d(p_{tj}, p_{ti}))\}$;
```

Time complexity of step1 and step2 are same as algorithm ACMS. Time complexity of step3 is $O(m^2n^2)$ under assumption that k and $PMA(\phi)$ are constants.

[Example 7] Consider MED in Figure 4, the module repertory in Figure 5, MSG for MED in Figure 6, and $k = 32$. The algorithm finds the path $p_s, [t_2, f_1], [t_4, f_3], p_t$, and gets a feasible sharing $S = \{[t_2, f_1], [t_4, f_3]\}$. Figure 9 shows the obtained $PMA(S)$. $area(PMA(S)) = 32$ satisfies the condition $area(PMA(S)) \leq 32$, and $delay(PMA(S)) = 7$ acheives the minimum delay. □

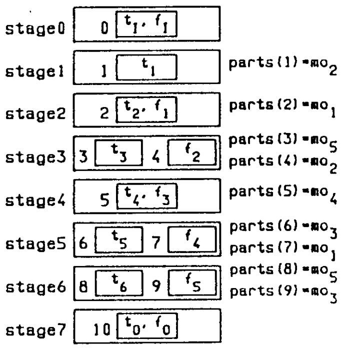

Figure 9: A feasible PMA in Example 7.

5 Conclusions

In this paper we have discussed the conditional resource sharing problems in pipeline synthesis. We formulated the subproblems, and presented new algorithms to solve them. To solve subproblems, we showed an essential property of module sharing, and developed an important data structure, *Module Sharing Graph*.

There are some topics of future research on conditional pipeline synthesis, that are currently under investigation. These are : (1) extension of algorithms so as to allow more than two conditional branches, (2) development of an algorithm when a data flow graph is not serial.

References

[1] K. S. Hwang et. al. : *"Constrained conditional resource sharing in pipeline synthesis,"* Proc. of ICCAD-88, pp.52-55 (1988).
[2] N. Park et. al. : *"Sehwa : A software package for synthesis of pipelines from behavioral specifications,"* IEEE Trans. CAD ,7, 3, pp.356-370 (1988).