3B－8

PARAdeg Computation and Scheduling of Self-Cleaning SWITCH-less Program Nets

Hidenori Yanagida and Qi-Wei Ge
Faculty of Education, Yamaguchi University, Japan

1. Introduction

A program net (denoted as PN) is a graph representation of a data-flow program and its parallelism has been studied through computing $PARAdeg = \sum(\overline{f}(z_i)\tau_i)/\underline{T}$, where $\overline{f}(z_i)$ and τ_i are maximum firing number and single firing time of node z_i respectively and $\underline{T}$ is minimum firing time of PN [1]. For a SWITCH-less PN, the numerator of $PARAdeg$ can be efficiently computed, however the denominator is usually costed with exponential computation time [1]. To improve this, we proposed an improved algorithm by a way of contracting nodes for self-cleaning SWITCH-less PNs with identical node firing time [2]. However the problem for the case of arbitrary node firing time has not been studied yet. And also it is important to schedule execution of PNs by using finite processors. In this paper, we first propose conditions that node contraction can be carried out for self-cleaning SWITCH-less PNs with arbitrary node firing time. Then we give scheduling methods by using $PARAdeg$ processors to execute SWITCH-less PNs.

2. Preliminary

PN is constituted of AND-node (○), OR-node (△) and SWITCH-node (▽°) (with node firing time τ_i) representing operations, and directed edges representing transmission channel of tokens [1]. Node firing rules and the detaileds refer to reference [1].

Minimum firing time $\underline{T}$ of a PN is the minimum time while infinite processors are allowed to fire each node z_i $\overline{f}(z_i)$ times. A SWITCH-less PN is of no SWITCH-node. A PN with 0 initial token distribution is self-cleaning if there is no token remaining on the edges after the firing of PN terminated [3]. In this paper PNs are assumed to be self-cleaning and SWITCH-less, and AND-nodes are allowed to possess multiple input and output edges.

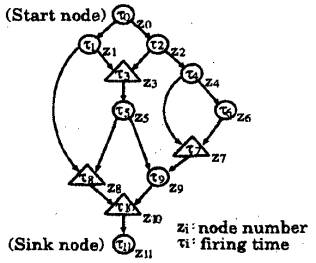

Fig.1 : An example of self-cleaning SWITCH-less PN .

The improved algorithm is to compute $\underline{T}$ for self-cleaning SWITCH-less PNs (as shown in Fig.1) with identical node firing time by applying AND-node contraction rules as shown in Fig.2 [2].

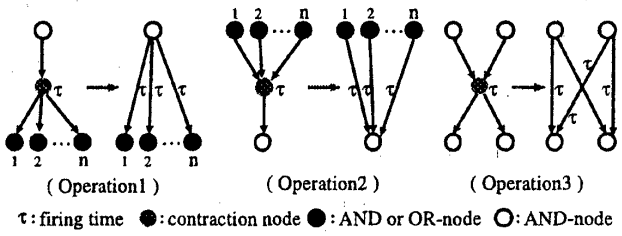

Fig.2 : AND-node contraction rules .

3. Conditions of Node Contraction

Suppose z be an AND-node with single input edge and firing time τ. If two tokens enter the input edge in succession within time interval τ, then z can not be contracted and our algorithm [2] fails to be used. The following proposition gives the condition for general AND-node contraction.

[Proposition 1] Let z be any AND-node with multiple input edges and τ be its firing time. z can be contracted if the following condition holds :

$$\max(t_n^1, \cdots, t_n^m) - \max(t_{n-1}^1, \cdots, t_{n-1}^m) \geqq \tau.$$

Here, $t_k{}^i$ shows the time when k-th token appears on the i-th input edge. □

Further, following Proposition 2-4 can be derived from Proposition 1.

[Proposition 2] All the AND-nodes of a PN are possible to be contracted if the node firing times of all the nodes are identity ($\tau_1 = \tau_2 = \ldots = \tau$). □

[Proposition 3] Let z be an AND-node with firing time τ and z_i (i = 1,2,...,n) be an input node of z with firing time τ_i. z can be contracted if the following condition holds :

$$\min(\tau_1, \tau_2, \cdots, \tau_n) \geqq \tau .$$ □

[Proposition 4] Let P_1, P_2, ..., P_n be n paths from start node to an AND-node z and τ_j^i be the firing time of j-th node of P_i as shown in Fig.3. z can be contracted if the following conditions hold :

$$P_1 : \max(\tau_1{}^1, \tau_2{}^1, \cdots, \tau_m{}^1) \geqq \tau,$$
$$P_2 : \max(\tau_1{}^2, \tau_2{}^2, \cdots, \tau_l{}^2) \geqq \tau,$$
$$\vdots$$
$$P_n : \max(\tau_1{}^n, \tau_2{}^n, \cdots, \tau_k{}^n) \geqq \tau.$$ □

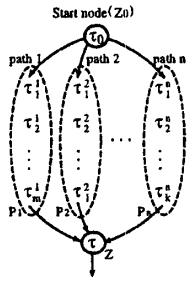

Fig.3 : An example of paths from start node to an AND-node z .

In the study [2], we treated only the PNs with identical node firing time. Hence we can extend our algorithm [2] to suiting for the PNs with arbitrary node firing time by applying Proposition 4. The way to do this is to limit node contraction only to those AND-nodes satisfying Proposition 4.

4 . Scheduling Algorithm

Scheduling problem is one of NP-complete problems even we limit nets to AND-node only ones [4]. Therefore here we propose three heuristic scheduling methods to execute SWITCH-less PNs by using $PARAdeg$ processors and then evaluate the performance of the methods.

The following algorithm is to label each node before scheduling.

〈 Labeling Node 〉

Step1$^{\cdot}$: Regard $\overline{f}(z_i)$ and τ_i as node weights of z_i respectively to calculate longest distances $D_f(z_i)$ and $D_\tau(z_i)$ from z_i to the sink node t by using BFS (Breadth First Search).

Step2$^{\cdot}$: Label each node z_i with $D(z_i) = D_f(z_i) \times D_\tau(z_i)$ and sort nodes in a list L in accordance with descending order of $D(z_i)$.

Step3$^{\cdot}$: Stop.

$D(z_i)$ indicates the maximum firing time from node z_i to the sink node.

Now we give our scheduling algorithms for PNs, which are respectively based on the following three priorities among firable nodes:

Method 1: Use L as priority list.
Method 2: Take priority of OR-node.
Method 3: Take priority of the larger τ_i.

Here in case of nodes with same priority when using Methods 2 and 3, Method 1 is to be applied. The following algorithm works for all of Methods 1, 2 and 3 using $PARAdeg$ processors.

〈 Scheduling of PN 〉

Step1$^{\cdot}$: Carry out 〈 Labeling Node 〉 and set t = 0.

Step2$^{\cdot}$: If there are unoccupied processors, then choose firing nodes from firable ones according to *Method* i (i = 1 or 2 or 3) and execute their firings.

Step3$^{\cdot}$: Update t = t+1 and release processors that have finished executing firing at current time t.

Step4$^{\cdot}$: If there is no occupied processor and no firable node, stop. Otherwise goto *Step2*$^{\cdot}$.

We have applied 〈 Scheduling of PN 〉 for executing 10 arbitrary PNs. Table 1 shows the results by Methods 1, 2 and 3 in 〈 Scheduling of PN 〉 as well as the result by using sufficient $|V|$ processors. Each data indicates processors' ratio of working measured as (total working time of processors)/($PARAdeg$ × (executing time of PN)). From Table 1, all three methods have certainly better performance than using $|V|$ processors. Among the three methods, Method 2 is best in average. That is because firing of an OR-node may probably create more firable nodes than firing other nodes.

Table 1 : Experimental results .

	method 1	method 2	method 3	|V| processors
PN-1	59.5 %	67.4 %	67.4 %	16.9 %
PN-2	68.9 %	68.9 %	70.4 %	21.1 %
PN-3	75.4 %	77.2 %	77.2 %	15.1 %
PN-4	65.9 %	65.9 %	67.5 %	18.6 %
PN-5	72.2 %	78.8 %	72.2 %	21.7 %
PN-6	75.9 %	75.9 %	73.3 %	8.4 %
PN-7	81.8 %	81.8 %	81.8 %	9.1 %
PN-8	70.9 %	72.6 %	67.8 %	20.6 %
PN-9	81.2 %	77.8 %	84.8 %	18.2 %
PN-10	77.8 %	81.7 %	77.8 %	18.7 %
Average	73.0 %	74.8 %	74.0 %	16.8 %

5 . Concluding remarks

We have firstly proposed conditions, that node contraction can be carried out for self-cleaning SWITCH-less PNs with arbitrary node firing time, in order to keep using of our previous algorithm[2] and then given scheduling methods to execute SWITCH-less PNs by $PARAdeg$ processors. The performance evaluation for scheduling has shown that Method 2 is best among the three. It still remains to be solved to evaluate performance of (1) the previous algorithm[2] for PNs with arbitrary node firing time and (2) our scheduling methods for much more PNs.

Reference

[1] Q.W. Ge, T. Watanabe, and K. Onaga: "Analysis of parallelism in autonomous execution of data-flow program nets," IEICE Trans., vol.E74, no.10, pp.3008-3017, Oct.1991.

[2] 柳田 英徳, 葛 崎偉: "SWITCH-less Program Net における最小実行時間検出アルゴリズムの改良," 電気・情報関連学会中国支部第 4 6 回連合大会講演論文集, pp448, 1995.

[3] Q.W. Ge and K.Onaga: "On verification of token self-cleanness of data-flow program nets," IEICE Trans., vol.E79, no.6, pp.812-817, Jun.1996.

[4] D.W.Gillies and J.W.-S.Liu: "Scheduling tasks with and/or precedence constraints" SIAM J. Comput., vol.24, no4, pp.797-810, Aug.1995.