PARALLEL SIMULATION OF BOUNDED PETRI NETS IN SINGLE PROCESSOR

Young Chan Kim*, In Sup Kwon**, and Tag Gon Kim†

*Department of Computer Science
Taejon National University of Technology
305-3 Samsung2-dong, Dong-ku, Taejon, 300-717, KOREA
e-mail: yckim@tnut.ac.kr

**QTS Inc, Phoenix, AZ 85016

†Department of Electrical Engineering
Korea Advanced Institute of Science and Technology
373-1 Kusung-dong, Yusung-ku, Taejon, 305-701, KOREA
e-mail: tkim@ee.kaist.ac.kr

KEYWORDS

Model analysis, SIMD-type parallel simulation, discrete simulation, Petri nets.

ABSTRACT

Petri nets are a powerful modeling tool for logical analysis of concurrent systems. Simulation may be used for such purpose. A class of problems can be logically analyzed by using bounded Petri Nets in which the number of tokens in each place does not exceed a finite number. This paper presents a new, data packing based parallel simulation algorithm of bounded Petri nets in a single processor. Experimental results showed that simulation using the data packing scheme in a single processor with 64-bits registers was faster than one without data packing by more than 10 times.

1 INTRODUCTION

As semiconductor technology has been growing rapidly, 32 bit and/or 64 bit microprocessors have been abundant in our computing environment. However, in many applications, the increased size of those microprocessors is not used fully. An example would be using a 32 or 64-bit processor for simulation of bounded Petri nets. Because systems to be being implemented with finite resources should be bounded, almost all realistic systems are modeled as bounded Petri nets.

In such a case, the number of bits to represent a marking element of a bounded Petri net will be much smaller than the number of bits of a register. To increase utilization of a processor, multiple marking elements can be packed into a register and processed simultaneously. Similarly elements of input and output matrices of the Petri net can also be packed. Then the simulation algorithm of Petri nets can be parallelized because it contains only integer addition and subtraction operations except multiplication and division operations. Based on such an idea, this paper presents a new, data packing based parallel simulation algorithm of bounded Petri nets in a single processor.

The rest of the paper is organized as follows: Section 2 introduces the definition and notation of Petri nets used in this work. Section 3 discusses the data packing scheme. Section 4 discusses the SIMD algorithm for the simulation. Section 5 reports on our empirical study.

2 DEFINITION

We have adopted a Petri net formalism that is based on (Peterson, 1981) with some modifications. Let N be the no negative integers. Formally, we define a Petri net as a tuple $PN = (P, T, D^-, D^+, M_0)$ where

- $P = \{p_1, p_2, \cdots, p_{|P|}\}$ is a finite set of places, drawn as circles.
- $T = \{t_1, t_2, \cdots, t_{|T|}\}$ is a finite set of transitions, drawn as boxes.
- For $1 \leq i \leq |P|$ and $1 \leq j \leq |T|$, $D^- = [d_{ji}^-]_{|T| \times |P|}$ and $D^+ = [d_{ji}^+]_{|T| \times |P|}$ are the input and output matrices. $d_{ji}^-$and d_{ji}^+ denote the multiplicities of the input arc from p_i to t_j, and the output arc from t_j to p_i.
- $M = [m_1, m_2, \cdots, m_{|P|}]$ is the marking vector of token counts of the places, where a token is represented by a black dot within a circle. M_0 is the initial marking. m_i denote the token count of place p_i in marking M.

The simulation rule of Petri nets is defined as follows:

(i) Find all enabled transitions in a current marking M. A transition $t_j \in T$ is enabled (firable) if and only if:

$$m_i \geq d_{ji}^-, \quad \text{for } i = 1..|P|. \tag{1}$$

(ii) Select a firing transition among enabled transitions randomly.

(iii) Fire the firing transition t_j. The next marking M' after firing the transition t_j is given by

$$m_i' = m_i - d_{ji}^- + d_{ji}^+, \quad \text{for } i = 1..|P| \tag{2}$$

and is denoted as $M \rightarrow M'$.

A Petri net is said to be *k-bounded* if the number of tokens in each place does not exceed a finite number k for any marking reachable from M_0.

Fig. 1 shows the graph and matrices of an example Petri net. The enabled transition set in the initial marking $[1, 2, 0, 0]$ is $\{t_1\}$. This Petri net is 3-bounded Petri net.

3 DATA PACKING SCHEME

Although recently a few processors such as Intel Pentium MMX (Intel Corporation, 1997) and Sun UltraSPARC have added a set of extensions to accelerate packed data processing, their availability is limited relatively and applications which use such special extensions will lose portability easily. In this paper, we present a new, data packing based parallel simulation algorithm which exploits SIMD-type parallelism in a normal processor. According to the simulation rule of Petri nets, only three binary operators ($\geq$, $-$, and $+$) of which operands are natural numbers are needed for Petri net simulation. Thus, if we pack M and each row of $D^-$and D^+properly, it is possible to efficiently imitate the corresponding packed operators ($\succeq$, $\ominus$, and $\oplus$) with only regular instructions such as CMP, SUB, ADD available to normal processors.

In order to pack M, $D^-$and D^+properly, first we should find the largest value κ among their elements. If a Petri net is k-bounded, the number of tokens in each place cannot exceed a finite number k for any marking reachable from M_0 by definition. So k is the upper bound value for all reachable markings. Let d_{max} be the largest element among those of $D^-$and D^+. Then κ is defined as follows:

$$\kappa = \max(k, d_{max})$$

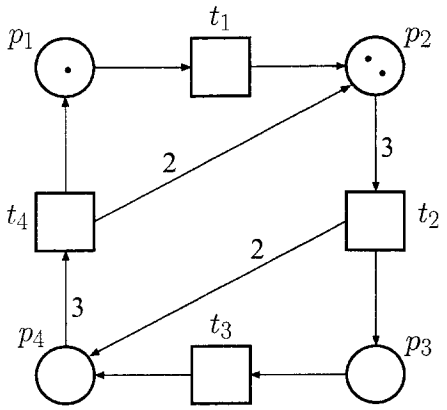
Once κ is obtained for a Petri net, we know that only $\lceil \log_2^{\kappa+1} \rceil$ bits are required to store an element of M, $D^-$and D^+, where $\lceil\ \rceil$ is a ceiling function. Let $|R|$ be the number of bits of an ALU register (for example, $|R|$ is 64 for the DEC Alpha processor). There is a possibility that $\lfloor |R| / \lceil \log_2^{\kappa+1} \rceil \rfloor$ elements are packed into a register, where $\lfloor\ \rfloor$ is a flooring function. This is actually good for the two packed arithmetic operators ($\ominus$ and $\oplus$) corresponding to Equation 2. However, we will use an extra bit (which acts as the sign bit) for each packed element, in order to implement the packed comparison operator ($\succeq$) related to Equation (1) efficiently. (The reason for this will be explained in detail in the next section.) Therefore, total $\lceil \log_2^{\kappa+1} \rceil + 1$ bits are required to store such an packed element as the number of tokens in a place. Then, the *maximum packing factor* (MPF) is defined as:

$$MPF = \left\lfloor \frac{|R|}{\lceil \log_2^{\kappa+1} \rceil + 1} \right\rfloor$$

and the *packing factor* (PF) used actually can be selected as

$$1 \leq PF \leq MPF.$$

If PF is determined, we can pack every PF neighboring elements of M and each row of $D^-$and D^+into a unit. Let $\underline{|P|}$ be $\lceil |P| / PF \rceil$. Similarly, from now on, we



(a) Graph represenation

$$D^- = \begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \\ t_4 \end{array} \begin{array}{c} \begin{array}{cccc} p_1 & p_2 & p_3 & p_4 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 3 \end{bmatrix} \end{array} \quad D^+ = \begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \\ t_4 \end{array} \begin{array}{c} \begin{array}{cccc} p_1 & p_2 & p_3 & p_4 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 1 \\ 1 & 2 & 0 & 0 \end{bmatrix} \end{array}$$

$$M_0 = \begin{array}{c} \begin{array}{cccc} p_1 & p_2 & p_3 & p_4 \end{array} \\ \begin{bmatrix} 1 & 2 & 0 & 0 \end{bmatrix} \end{array}$$

(b) Matrix representation

Figure 1: An example Petri net

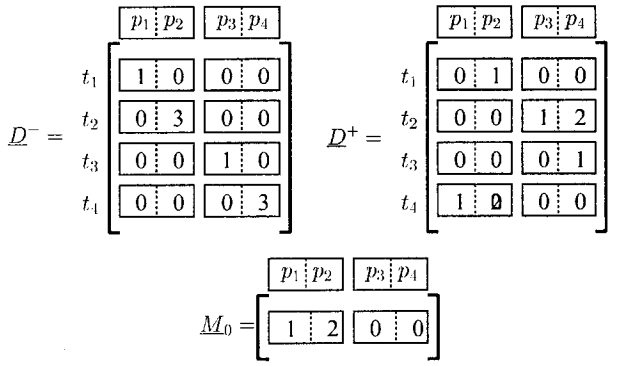


Figure 2: Packed data for Fig. 1(b) where $MPF = 10$, $PF = 2$

will use underlined symbols for packed data representation. The packed matrices corresponding to M, D^-, and D^+ are denoted $\underline{M} = [\underline{m}_i]_{|\underline{P}|}$, $\underline{D}^- = [\underline{d}^-_{ji}]_{|T|\times|\underline{P}|}$, and $\underline{D}^+ = [\underline{d}^+_{ji}]_{|T|\times|\underline{P}|}$, respectively.

For example, if we assume $|R|$ is 32, MPF for Fig. 1 is $\left\lfloor \frac{32}{\lceil \log_2^{3+1} \rceil + 1} \right\rfloor = \lfloor 32/3 \rfloor = 10$. Fig. 2 shows packed data for Fig. 1 (b) where $PF = 2$. The rectangles represent the packed data units.

4 PARALLEL SIMULATION ALGORITHM

As introduced in Section 2, the simulation rule of Petri nets consists of the following three phases: (i) finding the enabled transitions, (ii) random selection of a firing transition, and (iii) firing the selected transition. In this paper, we parallelize the (i) and (iii) phases by exploiting the packing scheme proposed in the previous section.

Let's consider a Petri net $\underline{PN} = (P, T, \underline{D}^-, \underline{D}^+, \underline{M}_0)$ packed with an arbitrary packing factor PF.

The first phase of simulation rule is redefined to be parallelized. More specifically, the enabling condition to determine whether a transition is enabled or not is parallelized for packed matrices as follows. A transition $t_j \in T$ is *enabled* in a packed marking $\underline{M}$ if

$$\underline{m}_i \succeq \underline{d}^-_{ji} \quad \text{for } i = 1..|\underline{P}|, \tag{3}$$

where $\succeq$ returns true if each packed data in the left operand is greater than or equal to the corresponding packed data in the right operand. The above enabling condition is applied to each transition $t_j \in T$ to find all possible enabled transitions.

$l \succeq r$ is defined as $((l - r)$ & $EXTRA\_BITS) == 0$ in C-like syntax where & is a *bitwise and* operator, and $EXTRA\_BITS$ is a bit mask that sets the extra bits of packed data to 1. Note that an extra bit is used for each packed data in Section 3. This guarantees the extra bits remain always unset through simulation because each element of D^-, D^+ or M use only $\lceil \log_2^{\kappa+1} \rceil$ bits. The implementation of $\succeq$ makes use of the fact. If and only if at least an element of r is greater than the

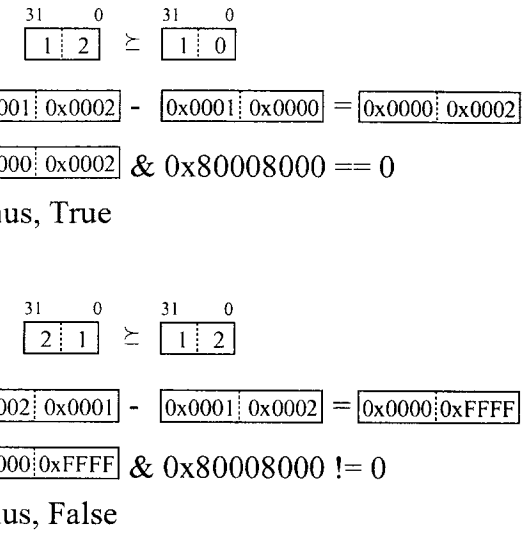


Figure 3: Comparisons of packed data where $|R| = 32$ and $PF = 2$.

corresponding one of l, the extra bit of the corresponding one of $l - r$ is set since arithmetic underflow will occur. So, by checking the extra bits after subtraction with $EXTRA\_BITS$, $l \succeq r$ can be determined easily. Fig. 3 show two examples where $|R| = 32$ and $PF = 2$.

The logical comparison $\succeq$ is executed $|\underline{P}|$ times to determine whether a transition is enabled or not. Note that execution time of $\succeq$, ET($\succeq$), takes a little more than execution time of $\geq$, ET($\geq$). Therefore, the speedup of the first phase, ratio of execution time of Equation (1) to execution time of Equation (3), is

$$\begin{aligned} \text{Speedup(phase1)} &= \frac{|\underline{P}| \cdot \text{ET}(\geq)}{|\underline{P}| \cdot \text{ET}(\succeq)} \\ &\approx PF \cdot \frac{\text{ET}(\geq)}{\text{ET}(\succeq)} \end{aligned}$$

In SUN SPARCstation 20 using gnu c++, for example, ET($\geq$) takes 4 instructions and ET($\succeq$) 8 instructions as shown in Table 1. Thus, Speedup(phase1) $\approx \frac{PF}{2}$. To expect the speedup in the first phase, PF have to be at least 3 (≥ 2).

To parallelize the third phase is as follows. In the packed marking $\underline{M} = [\underline{m_i}]_{|\underline{P}|}$, the result of firing transition t_j, $\underline{M'} = [\underline{m'_i}]_{|\underline{P}|}$ is redefined as follows:

$$\underline{m'_i} = (\underline{m_i} \ominus \underline{d^-_{ji}}) \oplus \underline{d^+_{ji}} \quad \text{for } i = 1..|\underline{P}|. \qquad (4)$$

where $\ominus(\oplus)$ subtracts (adds) the packed data of the right operand from(to) the packed data of the left operand.

$\underline{m_i} \ominus \underline{d^-_{ji}}$ can simply be $\underline{m_i} - \underline{d^-_{ji}}$ because firing t_j implies $\underline{m_i} \succeq \underline{d^-_{ji}}$ for $i = 1..|\underline{P}|$. $\oplus$ can also be $+$ because each packed data of $\underline{m'_i}$ cannot exceed $k(\leq \kappa)$ for k-bounded Petri nets. Thus, Equation (4) is redefined as follows:

$$\underline{m'_i} = (\underline{m_i} - \underline{d^-_{ji}}) + \underline{d^+_{ji}} \quad \text{for } i = 1..|\underline{P}|.$$

Therefore, the speedup of the third phase is

$$\begin{aligned} \text{Speedup(phase 3)} &= \frac{|\underline{P}| \cdot (\text{ET}(-) + \text{ET}(+))}{|\underline{P}| \cdot (\text{ET}(-) + \text{ET}(+))} \\ &\approx PF \end{aligned}$$

5 EXPERIMENT

In order to assess the effectiveness of our parallel algorithm, we analyzed two examples under two environments: (i) a DEC-3000 workstation (64 bit Alpha processor) running on the DEC OSF/1 V2.0 operation system, and (ii) a SUN SPARCstation 20 workstation (32 bit SPARC processor) running on the Solaris 2.4 operating system. In order to put focus on comparisons between the actual speedup and PF which is the theoretical upper bound, two low bounded Petri nets are used: (i) the 1-bounded dining philosophers problem (Peterson, 1981) with 32 philosophers, and (ii) the 42-bounded readers/writers problem with 42 readers and one writer, which is slightly different from the readers/writers problem in (Peterson, 1981) in that each reader is modeled with its own place.

The speedup of our algorithm, i.e., ratio of the conventional algorithm to our algorithm in execution time, is measured by varying the packing factor PF from 2 to MPF where PF is restricted to the power of two for easy experimental implementation. The results are shown in Table 2 and Table 3. In the two tables, T_{total} means the total elapsed simulation time during 40,000 transition fires. The columns labeled *old* in each table, give the data of the conventional simulation algorithm. The columns labeled *new* corresponds to different packed factor PF for our proposed parallel simulation algorithm.

As shown in the two tables, it is apparent that the speedup obtained by the new algorithm well follows PF. However the more PF is increases, the more the speedup is saturated.

Table 1: Assembly code of gnu g++ in SUN SPARCstation

logical operator	Gnu g++ code	Assembly code

l≥r	l >= r	ld [%fp-20],%o0
		ld [%fp-24],%o1
		cmp %o0,%o1
		bl .LL2

l≥r	(l - r) & EXTRA_BITS	ld [%fp-20],%o0
		ld [%fp-24],%o1
		sub %o0,%o1,%o0
		sethi %hi(-2139062144),%o2
		or %o2,%lo(-2139062144),%o1
		and %o0,%o1,%o0
		cmp %o0,0
		be .LL3

Table 2: *Experiment for the 1-bounded dining philosophers problem*

	DEC Workstation 64bit Alpha Processor $\|R\| = 64, MPF = 32$						SUN SPARCstation 20 32bit SPARC Processor $\|R\| = 32, MPF = 16$				
Algorithm	old	new					old	new			
PF	1	2	4	8	16	32	1	2	4	8	16
T_{total}(sec)	17.22	9.79	4.76	2.82	1.69	1.30	19.41	11.05	5.97	3.47	2.24
Speedup		1.76	3.47	6.11	10.19	13.25		1.76	3.25	5.59	8.09
Speedup/PF (%)		88.0	86.8	76.4	63.7	41.4		88.0	81.3	69.9	54.3

Table 3: Experiment for the 42-bounded readers/writers problem

	DEC Workstation 64bit Alpha Processor $\|R\| = 64, MPF = 32$				SUN SPARCstation 20 32bit SPARC Processor $\|R\| = 32, MPF = 16$		
Algorithm	old	new			old	new	
PF	1	2	4	8	1	2	4
T_{total} (sec)	22.12	12.87	6.60	3.68	27.16	15.58	8.14
Speedup		1.72	3.35	6.01		1.74	3.33
Speedup/PF (%)		86.0	83.8	75.1		87.0	83.3

6 CONCLUSION

Petri nets are a powerful modeling tool for logical analysis of concurrent systems. Simulation may be used for such purpose. A class of problems can be logically analyzed by using bounded Petri Nets in which the number of tokens in each place does not exceed a finite number. In such a case, the number of bits to represent a marking element of a bounded Petri net will be much smaller than the number of bits of a register. To increase utilization of a processor, multiple marking elements can be packed into a register and processed simultaneously. Similarly elements of input and output matrices of the Petri net can also be packed.

Based on such an idea, we have presented a new parallel simulation algorithm of bounded Petri nets in a single processor. It parallelizes two major phases except the trivial second phase of Petri nets simulation rule by packing multiple data into a register. To demonstrate its effectiveness, experiments were performed with two different bounded Petri nets. Experimental results showed that simulation using the data packing scheme in a single processor was faster than one without data packing by more than 13 times in case of $|R|=64$ (i.e., DEC Alpha processor) and $PF=32$. Overall, the proposed parallel algorithm well exploits the parallelism in single processor.

REFERENCES

Intel Corporation (1997). *The Complete Guide to MMX Technology.* McGraw-Hill.

Murata, T. (1989). Petri nets: Properties, analysis and applications. *Proceedings of IEEE*, 77(4):541–580.

Peterson, J. L. (1981). *Petri Net Theory and the Modeling of Systems.* Prentice-Hall, Inc., Englewood Cliffs, N.J.

SunSoft (1995). *SPARC Assembly Language Reference Manual.* SunSoft.