

# Modeling and Analysis of Simultaneous Multithreading

Wlodek Zuberek

Department of Computer Science  
Memorial University  
St.John's, Canada A1B 3X5

14<sup>th</sup> International Conference on  
Analytical and Stochastic Modeling Techniques and Applications  
Prague, Czech Republic, June 4-6, 2007

## Instruction-Level Multithreading

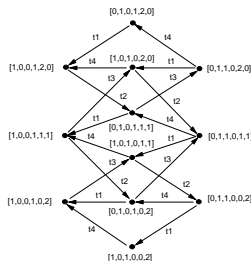
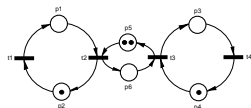
MULTITHREADING – an architectural approach to tolerating long latencies (e.g., memory accesses or synchronization delays).

In a traditional architecture, when a processor accesses memory, it waits for the result of this access, possibly after executing a few instructions that are independent of the access operation. In a modern microprocessors, this wait may involve more than a hundred processor cycles (when the required information is not in cache), so the utilization of the processor tends to be low. Alternatively, if a processor maintains multiple threads of execution, for each long-latency operation the processor can switch to another thread (which is called “context switch”) and continue to execute instructions rather than wait.

In **simultaneous multithreading**, several threads can issue instructions in each processor cycle. The threads switch context independently of each other.

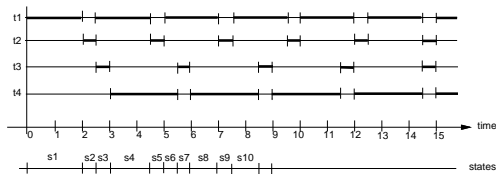
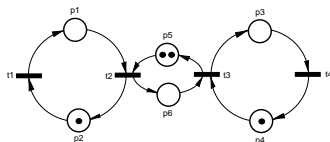
# Petri Nets

- ▶ A (marked) Petri net is a bipartite graph,  $\mathcal{M} = (P, T, A, m_0)$ , where  $P, T$  and  $A$  are a set of places, transitions and arcs, respectively, and  $m_0$  is the initial marking function,  $m_0 : P \rightarrow \{0, 1, \dots\}$ .
- ▶ A transition,  $t$ , is *enabled* by marking  $m$  iff  $\forall p \in \text{Inp}(t) : m(p) > 0$ . An enabled transition can *fire* — this removes one token from each of the transition's input places and adds one token to each of its output places.
- ▶ The *reachability graph* of a marked net can be derived by exhaustively exploring all possible markings.
- ▶ When no transition is enabled by a marking, the net is *deadlocked*.

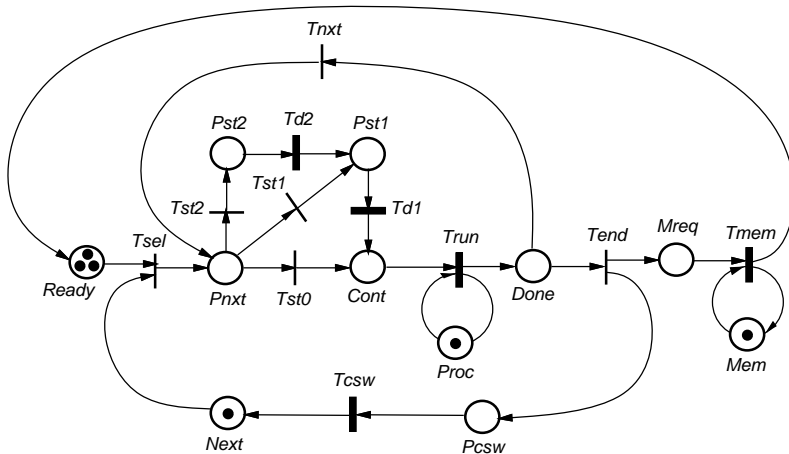


# Timed Petri Nets

- ▶ In timed Petri nets, firing times are associated with transitions and the firings occur in “real-time”, i.e., tokens are removed from all input places, then the firing continues for the duration of the firing time, and then tokens are deposited into output places.
- ▶ All firings start in the same time instants in which the transitions become enables (although some enabled transitions do not fire).
- ▶ All conflicts are resolved by random choices described by “choice probabilities” or relative frequencies of firings.
- ▶ For the firing times:  $f(t_1) = 2.0, f(t_2) = 0.5, f(t_3) = 0.5, f(t_4) = 2.5$ :

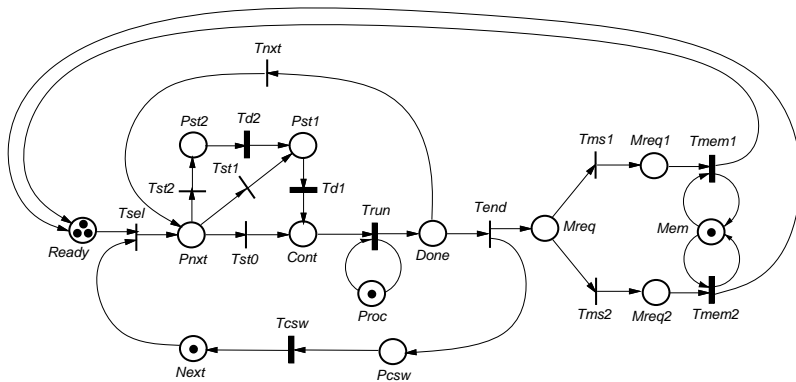


# Petri Net Model 1



Petri net model of a multithreaded processor.

## Petri Net Model 2



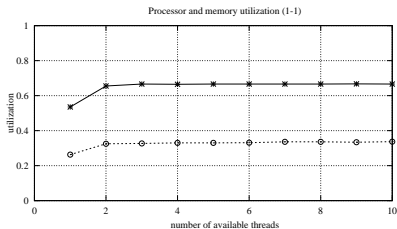
Petri net model of a multithreaded processor with two levels of memory.

# Modeling parameters

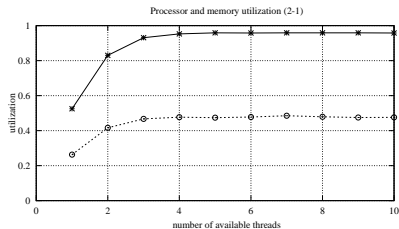
Simultaneous multithreading modeling parameters and their typical values:

<i>symbol</i>	<i>parameter</i>	<i>value</i>
$n_t$	number of available threads	1,...,10
$n_p$	number of execution pipelines	1,2,...
$n_s$	number of simultaneous threads	1,2,3,...
$\ell_t$	thread runlength	10
$t_{cs}$	context switching time	1,3
$t_m$	average memory access time	5
$p_{s1}$	prob. of one-cycle pipeline stall	0.2
$p_{s2}$	prob. of two-cycle pipeline stall	0.1

# Results 1: 1-1 and 2-1 processors



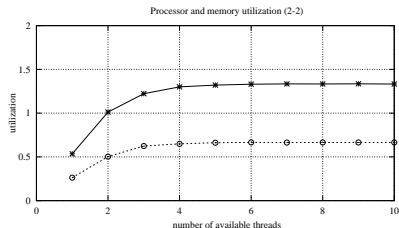
Processor (-x-) and memory (-o-) utilization for a 1-1 processor;  $l_t = 10$ ,  $t_m = 5$ ,  $t_{cs} = 1$



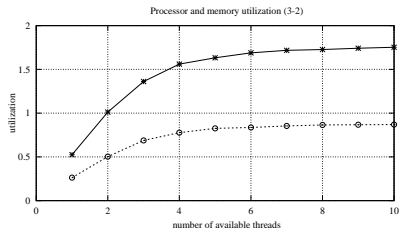
Processor (-x-) and memory (-o-) utilization for a 2-1 processor;  $l_t = 10$ ,  $t_m = 5$ ,  $t_{cs} = 1$



## Results 2: 2-2 and 3-2 processors

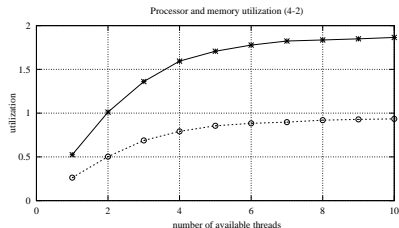


Processor (-x-) and memory (-o-) utilization for a 2-2 processor;  $l_t = 10$ ,  $t_m = 5$ ,  $t_{cs} = 1$

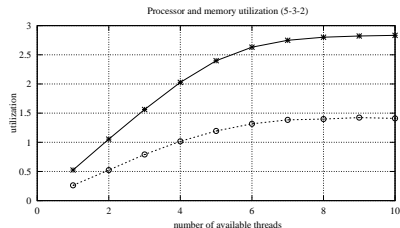


Processor (-x-) and memory (-o-) utilization for a 3-2 processor;  $l_t = 10$ ,  $t_m = 5$ ,  $t_{cs} = 1$

## Results 3: 4-2 and 5-3 processors

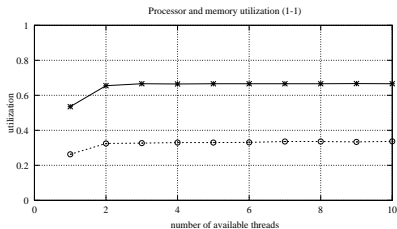


Processor (-x-) and memory (-o-) utilization for a 4-2 processor  $l_t = 10$ ,  $t_m = 5$ ,  $t_{cs} = 1$

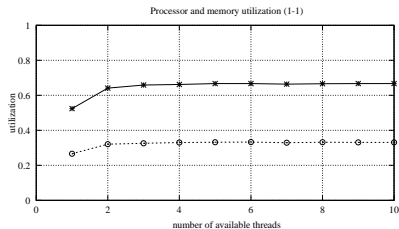


Processor (-x-) and memory (-o-) utilization for a 5-3 processor with dual-port memory;  $l_t = 10$ ,  $t_m = 5||2$ ,  $t_{cs} = 1$

## Results 4: 1-1 processor with one and two levels of memory

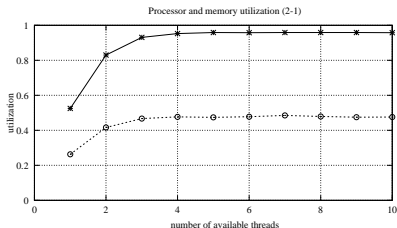


Processor (-x-) and memory (-o-) utilization for a 1-1 processor;  $l_t = 10$ ,  $t_m = 5$ ,  $t_{cs} = 1$

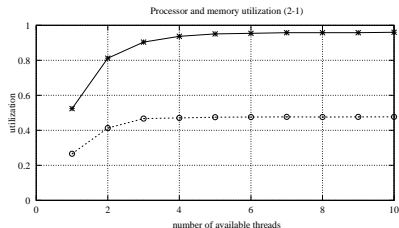


Processor (-x-) and memory (-o-) utilization for a 1-1 processor with 2-level memory;  $l_t = 10$ ,  $t_m = 4 + 20$ ,  $t_{cs} = 1$

## Results 5: 2-1 processor with one and two levels of memory

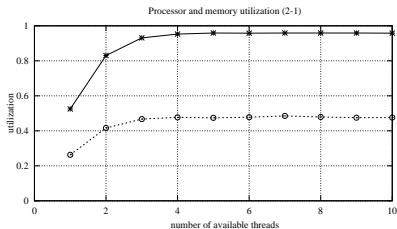


Processor (-x-) and memory (-o-) utilization for a 2-1 processor;  $l_t = 10$ ,  $t_m = 5$ ,  $t_{cs} = 1$

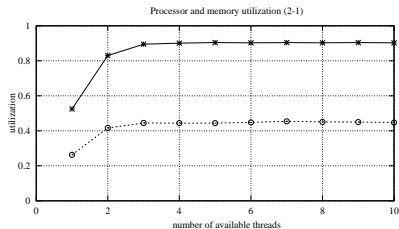


Processor (-x-) and memory (-o-) utilization for a 2-1 processor with 2-level memory;  $l_t = 10$ ,  $t_m = 4 + 20$ ,  $t_{cs} = 1$

## Results 6: 2-1 processor with short and long context switch



Processor (-x-) and memory (-o-) utilization for a 2-1 processor;  $l_t = 10$ ,  $t_m = 5$ ,  $t_{cs} = 1$



Processor (-x-) and memory (-o-) utilization for a 2-1 processor;  $l_t = 10$ ,  $t_m = 5$ ,  $t_{cs} = 3$ .

## Concluding Remarks

- ▶ Simultaneous multithreading increases the performance of processors by tolerating long-latency operations. It is not affected by the “2” barrier of the out-of-order approach.
- ▶ Since the long-latency operations play an increasingly important role in modern microprocessors (the processor-memory performance gap), the advantages of simultaneous multithreading are expected to be increasingly attractive.
- ▶ Hardware resources required for an implementation of simultaneous multithreading are much simpler than in the case of out-of-order approach. The main challenge of simultaneous multithreading is to balance the system by maintaining the right relationship between the number of simultaneous threads and the performance of the memory hierarchy.
- ▶ All presented results indicate that the number of available threads, required for improved performance of the processor, is quite small, and is typically greater by only 2 or 3 threads than the number of simultaneous threads; performance improvements due to a larger number of available threads are rather insignificant.

## Concluding Remarks

- For simple cases, the simulation results can be compared with the analytical solutions:

for the 1-1 processor:	$n_t$	<i>number of states</i>	<i>analytical utilization</i>	<i>simulated utilization</i>
	1	11	0.526	0.535
	2	57	0.656	0.655
	3	107	0.666	0.666
	4	157	0.666	0.666
	5	207	0.667	0.666

for the 3-2 processor:	$n_t$	<i>number of states</i>	<i>analytical utilization</i>	<i>simulated utilization</i>
	1	11	0.525	0.526
	2	86	1.012	1.012
	3	309	1.361	1.367
	4	660	1.560	1.553
	5	1,154	1.632	1.639