Out-of-Order Parallel Discrete Event Simulation for Transaction Level Models

Weiwei Chen

Advisor: Prof. Rainer Dömer

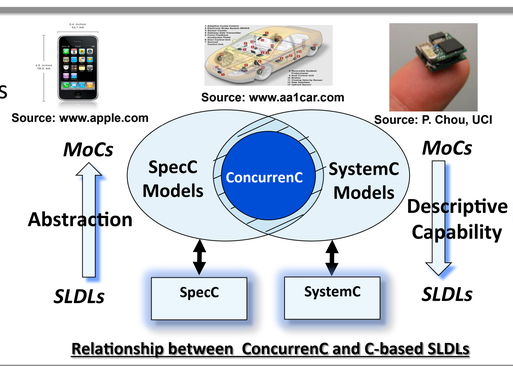
Center for Embedded Computer Systems

University of California, Irvine

Center for Embedded Computer Systems

❖ Embedded System Design

- Embedded systems are special purposed computer systems with a wide application domain, e.g. automobiles, medical devices, communication systems, etc.
- Transaction Level Modeling (TLM) models embedded systems at different abstraction levels.
- TLMs are usually described in System-Level Description Languages (SLDLs), e.g. SystemC and SpecC.
- TLMs are typically validated by Discrete Event (DE) simulation.



Relationship between ConcurrenC and C-based SLDLs

❖ Related Work: Accelerate TLM simulation

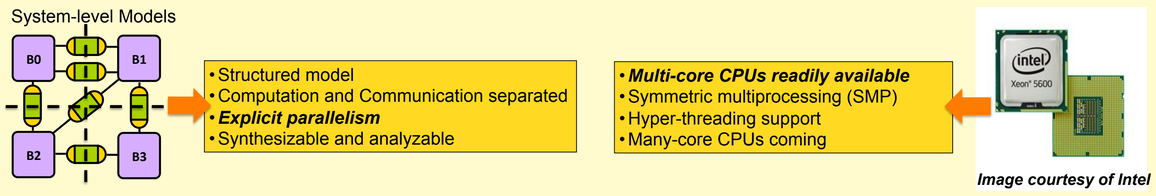
Distributed Simulation
- Chandy et al. [TSE'79]
- Huang et al. [SIES'08]
- Chen et al. [CECS'11]

Hardware-based Acceleration
- Sirowy et al. [DAC'10]
- Nanjundappa et al. [ASPDAC'10]
- Sinha et al. [ASPDAC'12]
- Vinco et al. [DAC'12]

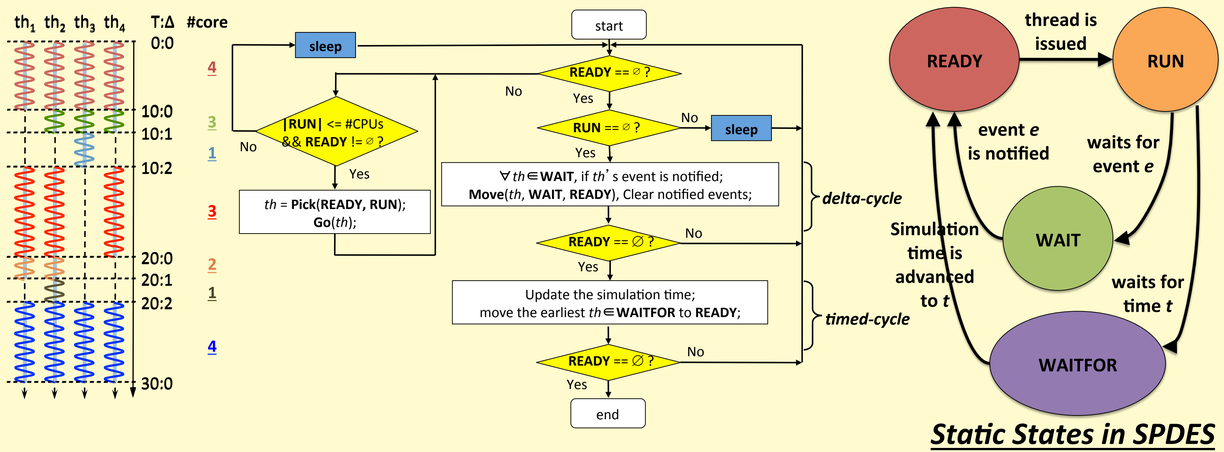
Modeling Techniques
- Transaction-level modeling (TLM)
- TLM temporal decoupling
- Source-level Simulation Stattelmann et al. [DAC'11]
- Host-Compiled Simulation Gerstlauer et al. [RSP'10]

SMP Parallel Simulation
- Fujimoto. [CACM'90]
- Chopard et al. [ICCS'06]
- Ezudheen et al. [PADS'09]
- Mello et al. [DATE'10]
- Schumacher et al. [CODES'11]
- Chen et al. [IEEED&T'11]
- Yun et al. [TCAD'12]

❖ Synchronous Parallel Discrete Event Simulation (SPDES)



- Executes threads in the same simulation cycles (*time, delta*) in parallel
- Needed protection of communication and synchronization can be automatically instrumented

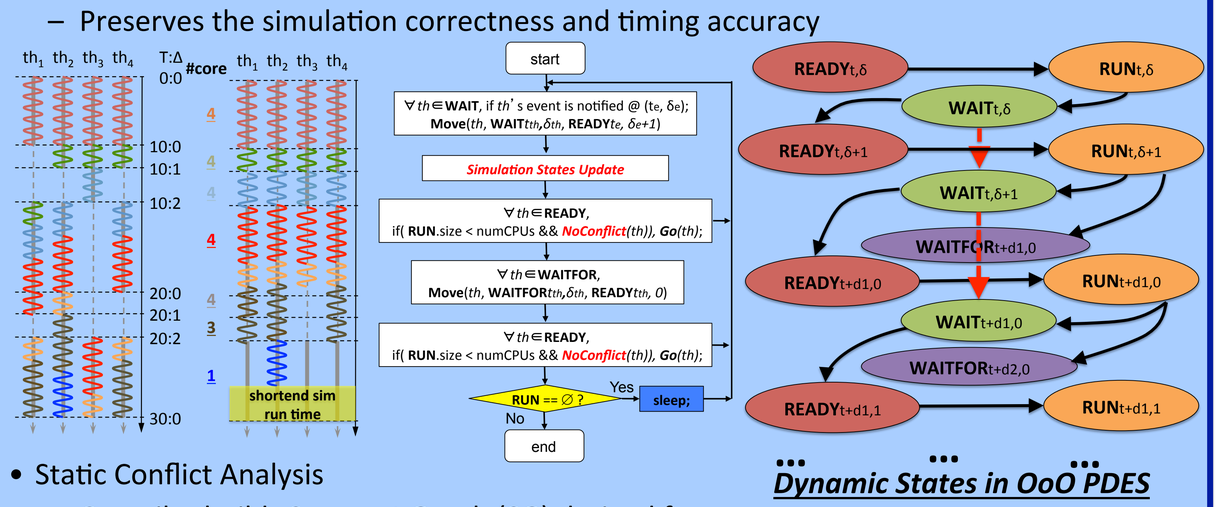


Static States in SPDES

	Traditional DE simulation	Synchronous PDES	Out-of-order PDES
Simulation Time	One global time tuple (t, δ) shared by every thread and event		Local time for each thread *th* as tuple (t_{th},δ_{th}). A total order of time is defined with the following relations: **equal:** $(t_1, \delta_1) = (t_2, \delta_2)$, *iff* $t_1 = t_2$, $\delta_1 = \delta_2$ **before:** $(t_1,\delta_1) < (t_2,\delta_2)$, *iff* $t_1 < t_2$, or $t_1 = t_2$, $\delta_1 < \delta_2$ **after:** $(t_1,\delta_1) > (t_2,\delta_2)$, *iff* $t_1 > t_2$, or $t_1 = t_2$, $\delta_1 > \delta_2$
Event Description	Events are identified by their ids, i.e, event (id).		A timestamp is added to identify every event, i.e. event (id, t, δ).
Simulation Thread Sets	**READY, RUN, WAIT, WAITFOR, JOINING, COMPLETE**		Threads are organized as subsets with the same timestamp (t_{th}, δ_{th}). Thread sets are the union of these subsets, i.e, **READY** = ∪**READY**$_{t,\delta}$, **RUN** = ∪**RUN**$_{t,\delta}$, **WAIT** = ∪**WAIT**$_{t,\delta}$, **WAITFOR** = ∪**WAITFOR**$_{t,\delta}$ (δ = 0), where the subsets are ordered in increasing order of time (t, δ)
Threading Model	User-level or OS kernel-level	***OS kernel-level***	
Run Time Scheduling	Event delivery in-order in delta-cycle loop Time advance in-order in outer loop		Event delivery out-of-order if no conflicts exist Time advance out-of-order if no conflicts exist
	Only one thread is active at one time ***No parallelism*** ***No SMP utilization***	Threads at same simulation cycle run ***in parallel*** ***Limited parallelism*** ***Inefficient SMP utilization***	Threads at same simulation cycle or with no conflicts run ***in parallel*** ***More parallelism*** ***Efficient SMP utilization***
Compile Time Analysis	No synchronization protection needed	***Need synchronization protection for shared resources,*** e.g. any user-defined and hierarchical channels, data structures in the scheduler	
	No conflict analysis needed		Static conflict analysis derives Segment Graph (SG) from the Control Flow Graph (CFG), analyzes variable and event accesses, passes conflict tables to the scheduler. **Compile time increases**

❖ Out-of-order Parallel Discrete Event Simulation (OoO PDES)

- Breaks simulation cycle barriers by localizing the time for each thread
- Aggressively issues threads to run in parallel even in different cycles
- Preserves the simulation correctness and timing accuracy

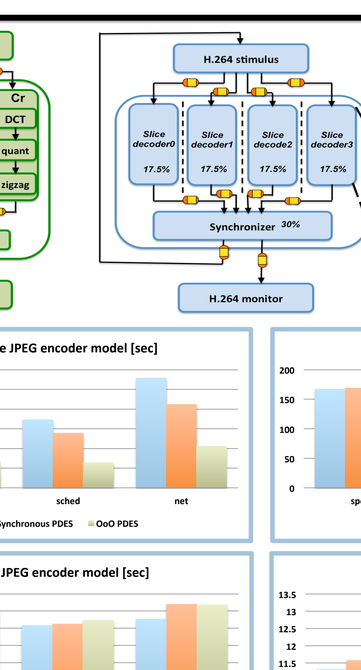


Dynamic States in OoO PDES

- Static Conflict Analysis
  - Compiler builds Segment Graph (SG) derived from Control Flow Graph (CFG) of applications
  - Compiler builds Segment Conflict Tables for quick look-up at runtime

```
1: #include <stdio.h>
2: int array[] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
3: behavior B(in int begin, in int end, out int sum)
4: { int i;
5:   void main(){
6:    int tmp; tmp = 0; i = begin;
7:    waitfor 1;  // v3, segment 3 starts
8:    while(i <= end){
9:     waitfor 2; // v4, segment 4 starts
10:     tmp += array[i]; i ++;
11:   }
12:   sum = tmp; }
13: };
14: behavior Main()
15: {int sum1, sum2;
16:  B b1(0, 4, sum1); B b2(5, 9, sum2);
17:  int main(){
18:   par{                    // v1, segment 1 starts
19:    b1.main();
20:    b2.main();
21:   } // v2, segment 2 starts
22:   printf("summation 1 is :%d \n", sum1);
23:   printf("summation 2 is :%d \n", sum2); }};
```

Example

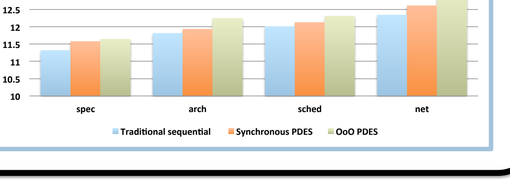


Segment Graph

seg	0	1	2	3	4	5	6
0	F	F	F	F	F	F	F
1	F	T	F	T	T	T	T
2	F	F	F	T	T	T	T
3	F	T	T	T	T	F	F
4	F	T	T	T	T	F	F
5	F	T	T	F	F	T	T
6	F	T	T	F	F	T	T

Segment Data Conflict Table

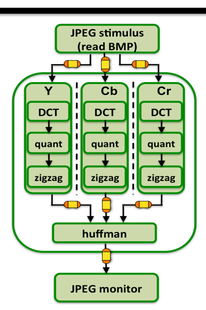
- OoO PDES Optimizations
  - ***Instance isolation*** reduces false conflicts [ASPDAC'12]
  - Scheduling ***prediction*** for more parallelism [DATE'13]
- Static code analysis applications
  - Conflicts detection for recoding
  - Debugging when parallelizing applications [HLDVT'12]

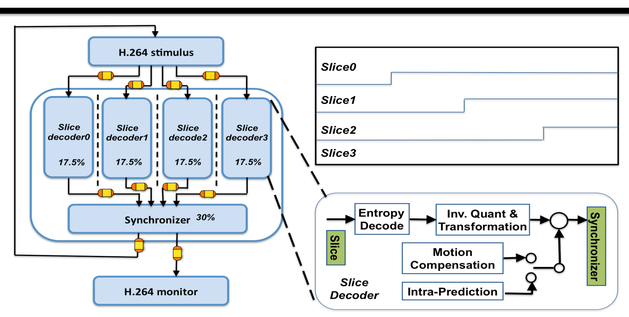


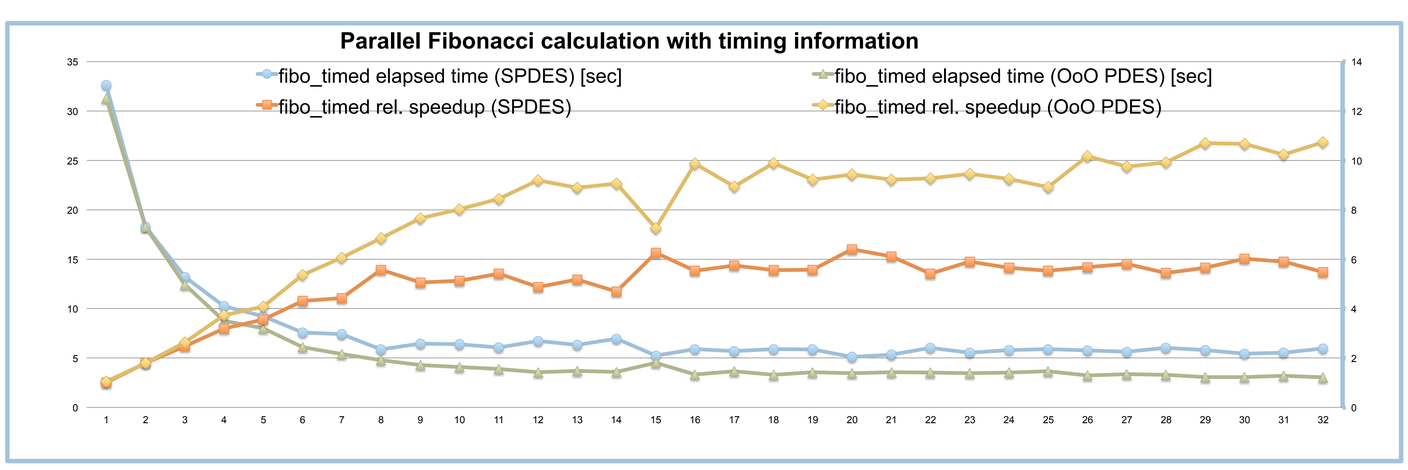
❖ Experiments and Results

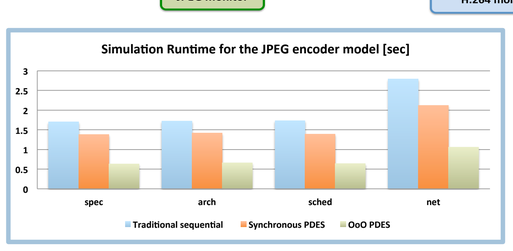
- Parallel Fibonacci calculation with timing information
- Parallel JPEG image encoder with 3 color components encoded in parallel, a sequential Huffman encoding
- Parallel H.264 video decoder with 4 slice decoders and a sequential slice reader and synchronizer

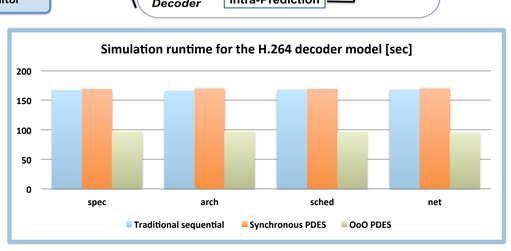
➢ Host: 64-bit Fedora 12 Linux with 2 6-core CPUs (Intel(R) Xeon(R) X5650) at 2.67 GHz with 2 hyper-threads per core (supports 24 threads in parallel)

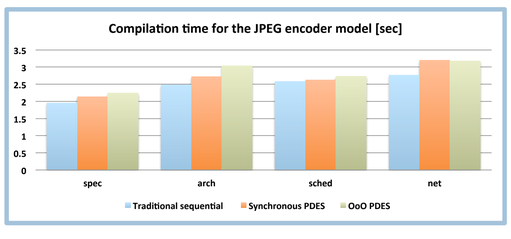












❖ Selection of Relevant Publications

- W. Chen, R. Dömer, "*ConcurrenC: A new approach towards effective abstraction of C-based SLDLs*", IESS 2009.
- W. Chen, R. Dömer, " *A Fast Heuristic Scheduling Algorithm for Periodic ConcurrenC Models*", ASP-DAC 2010.
- W. Chen, X. Han, R. Dömer, "*ESL Design and Multi-Core Validation using the System-on-Chip Environment*", HLDVT 2010
- R. Dömer, W. Chen, X. Han, Andreas Gerstlauer, "*Multi-Core Parallel Simulation of System-Level Description Languages*", ASP-DAC 2011.
- W. Chen, X. Han, R. Dömer, "*Multi-core Simulation of Transaction Level Models using the System-on-Chip Environment*", IEEE Design & Test of Computers, May-June 2011.
- R. Dömer, W. Chen, X. Han, "*Parallel Discrete Event Simulation of Transaction Level Models*", ASP-DAC 2012.
- W. Chen, R. Dömer, "*An Optimizing Compiler for Out-of-Order Parallel ESL Simulation Exploiting Instance Isolation*" ASP-DAC 2012.
- W. Chen, X. Han, R. Dömer, "*Out-of-order Parallel Simulation for ESL Design*", DATE 2012.
- W. Chen, X. Han, C. Chang, R. Dömer, "*Eliminating Race Conditions in System-Level Models by using Parallel Simulation Infrastructure*", HLDVT 2012.
- W. Chen, X. Han, C. Chang, R. Dömer, "*Advances in Parallel Discrete Event Simulation for Electronic System-Level Design*", IEEE Design & Test of Computers, January-February 2013.
- W. Chen, R. Dömer, "*Optimized Out-of-Order Parallel Discrete Event Simulation Using Predictions*" DATE 2013.