**Title:** HCCPS Line Project Final Review

**Authors:** Chaki/Dionisio de Niz SagarAndersson/BjornKlein/MarkGurfinkel/Arie

**Abstract:**
The original document contains color images.
Copyright 2014 Carnegie Mellon University

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Department of Defense.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN “AS-IS” BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

This material has been approved for public release and unlimited distribution except as restricted below.

This material may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other use. Requests for permission should be directed to the Software Engineering Institute at permission@sei.cmu.edu.


DM-0001788
Motivation

Many (DoD) systems are Cyber-Physical

- Software tightly coupled with physical world
- Increased scale, complexity, autonomy
  - Pilot Ejection ⇒ IMA ⇒ Multi-UAS Missions

Current DoD T&E regimen is expensive & inadequate to assure CPS

- Testing-based (poor coverage)
- Sufficient assurance needed for Certification

Rigorous assurance of CPS must include at least timing, functionality, and coordination

- Task 1: Timing ⇒ Schedulability analysis: multicore and memory interference
- Task 2: Functional ⇒ Model Checking: scalability, physical laws
- Task 3: Coordination ⇒ Prob. Mod. Checking: compositionality, uncertainty
Guiding Scenario: Multi-UAS Mission

- **Functional**: Tasks Free of Deadlocks and Race Conditions
- **Timing**: Collision Avoidance. Tasks Must Meet Deadlines
- **Coordination**: Optimal Coverage Within Mission Limit

Timing, functional correctness, and high-quality coordination are critical to success of modern CPSs. Each must be assured for high confidence in overall performance.
Task 1: Multicore Challenges for Real-Time Systems

Parallelization

- Computation time > Deadline
  - Must parallelized to meet deadline
  - Guarantee always finish before deadline

Shared Hardware Resources / Best Effort Schedulers

- Shared memory system creates unpredictable delays
- Memory accesses scheduled for average case hinder worst-case

Multiple elements to coordinate

- Shared cache
- Shared main memory
- Shared memory bus
Predictable Parallelization

Developed a staged execution model

Scheduled under Global Earliest-Deadline First

• Most efficient scheduling for staged execution
  – If task schedulable under optimal scheduler our scheduler need at most twice the speed to schedule task
Example: Parallel Image Processing

Edge detection

Shape classification

Shape matching

Multicore Processor

Core 1

Core 2

Core 3

Core 4

Divide image to process pieces in parallel
Shared Hardware: Multicore Memory System

- Core 1
  - L1/L2
- Core 2
  - L1/L2
- Core 3
  - L1/L2
- Core N
  - L1/L2

- Last-Level Cache (L3)

- Memory Bus (and Mem Controller)

- DRAM Bank 0
- DRAM Bank 1
- DRAM Bank 2
- DRAM Bank 2
- DRAM Bank B
DRAM Organization

DRAM Rank

- Command bus
- Address bus
- Data bus

DRAM Chip

- Command decoder
- Row decoder
- Column decoder
- Row address
- Column address
- Row buffer
- Bank 1
- Bank 8
- Columns
- Rows

DRAM access latency varies depending on which row is stored in the row buffer.
Impact of Memory Interference

- 1 attacker → Max 5.5x increase
- 2 attackers → Max 8.4x increase
- 3 attackers → Max 12x increase

We should predict, bound and reduce the memory interference delay!

![Graph showing the impact of memory interference on execution time.](image)
Timing Analysis with Bank Partitions (private/shared)

Explicitly considers the timing characteristics of major DRAM resources

- Rank/bank/bus timing constraints (JEDEC standard)
- Request re-ordering effect

Bounding memory interference delay for a task

- Combines request-driven and job-driven approaches

Software DRAM bank partitioning awareness

- Analyzes the effect of dedicated and shared DRAM banks
Page Coloring with Virtual Memory

Virtual Memory

Page table

Bank 0

Bank 1

Bank 2

...
Timing Verification: Response Time ($R_i$) < Deadline ($D_i$)
Timing Verification: Response Time ($R_i$) < Deadline ($D_i$)
Timing Verification: Response Time ($R_i$) < Deadline ($D_i$)

$$R_{i+1}^{k} = C_i + \sum_{\tau_j \in h_p(\tau_i)} \left\lceil \frac{R_i^k}{T_j} \right\rceil \cdot C_j$$

Per request

$$+ \min \left\{ H_i \cdot RD_p + \sum_{\tau_j \in h_p(\tau_i)} \left\lceil \frac{R_i^k}{T_j} \right\rceil \cdot H_j \cdot RD_p, JD_p(R_i^k) \right\}$$

Per job

Core 1
L1/L2

Core 2
L1/L2

DRAM Bank 0

time

$R_i$ $D_i$
Memory Interference with private banks

- Private DRAM Bank

Average over-estimates are 8%
(13% for a shared bank)

Cache Partitioning (Coloring)

Main Mem

Set associativity

Cache

Cache sets

One page

Address bits

16 15 14 13 12

6

Cache Index
Cache and Bank Address Bits

Address bits: 20 19 18 17 16 15 14 13 12

Cache Index

XOR

Bank Index

E.g. 2 bank bits
2 cache bits
1 shared bit

<table>
<thead>
<tr>
<th>Bank</th>
<th>00</th>
<th>01</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>01</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>11</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>
Coordinated Cache and Bank Partitioning

Avoid conflicting color assignments

Take advantage of different conflict behaviors
  • Banks can be shared within same core but not across cores
  • Cache cannot be shared within or across cores

Take advantage of sensitivity of execution time to cache
  • Task with highest sensitivity to cache is assigned more cache
  • Diminishing returns taken into account

Two algorithms explored
  • Mixed-Integer Linear Programming
  • Knapsack
Experimental Results

Partitions & Scheduling in Parallelized Tasks

Global core scheduling (gEDF)

Mixed Integer-Linear Programming:
- cache+bank partitions per page
- Interference between Parallel segments
- Interference between tasks

Round-trip parallelized tasks scheduling

Measure memory accesses per page in a task
  • Modified Valgrind profiler to count accesses to a particular virtual page in a program running on the target platform

Assign cache + bank colors to each page and test schedulability
  • Mixed-Integer Linear Programming Formulation
  • Outputs page per color

Modified Memory System (inside OS) to assign colors per page
  • Linux variant (Linux / RK)
  • Assign memory reservations (colors) to task and color regions to pages
  • Cache + Bank colors

Global Earliest-Deadline First (gEDF) implementation
  • In Linux / RK

Stage Synchronization Framework
  • For Parallel Staged Tasks

Experiments on Intel i7 quad-core 8GB RAM + 8MB Shared Cache
Task 2: Software Model Checking Using Over and Under Approximations

Periodic Program in C

Sequential Program

Sequentialization

Software Model Checker

Periods, WCETs, Initial Condition, Time bound

Result 1: Improved SMC by Combining Over and Under Approximations

Result 2: Improved Sequentialization by Using Memory Consistency Rules
Task 2: Improved Software Model Checking Using Over and Under Approximations

Program $P$

er-approx

$U$

2: SOLVE
Task 2: Improved Software Model Checking Using Over and Under Approximations

Program $P$

1. Under-approx

2. Solve

3. Feasible?

4. Abstract

5. Feasible?

6. Refine

No

PBA

$\pi_U$

Safety Proof

$C_U$

Cex

Yes

SAFE

UNSAFE

Maintain over-approx and under-approx simultaneously

Sound, Relatively Complete

Publication: Anvesh Komuravelli, Arie Gurfinkel, Sagar Chaki: SMT-Based Model Checking for Recursive Programs. CAV 2014: 17-34
Task 2: Model Checking Results

**RECMC vs. PDR Time**

- **Software Verification Competition 2014 Benchmarks**
  - Total = 855
  - RECMC better = 553
  - PDR better = 232

**TODO:**
- Bit-vector semantics
- Physical laws: additional theories

**PDR = State-of-the-art competitor for RECMC**

**NOTE:** below red line means RECMC better than PDR
Task 2: Improved Sequentialization Using Memory Consistency Rules

1. $VC$ is generated by using logical Lamport clocks that encode the priority-based preemption between threads.
2. Further optimization using variables “snapshots” that reduce redundant sub-formulas in $VC$.
3. 7 times faster than previous version of REK on benchmarks.

Publication: Sagar Chaki, Arie Gurfinkel, Nishant Sinha: Efficient Verification of Periodic Programs Using Sequential Consistency and Snapshots. FMCAD 2014
Task 3: Probabilistic Model Checking to evaluate Coordinated Multi-Robot Missions

Guiding Example

No localization, disc model of communication (i.e., within radius), probabilistic movement

Base station and Mine have disc model of communication

Each robot is Markovian
- \( \text{state} = (x, y, \text{time}, \text{direction}, \text{mine detected}) \).

No physical interaction, e.g., robots pass through

Property \( \phi_1 \) = Probability of mine detection.
\[
P = \mathcal{P}(\text{detected}_1 \lor \text{detected}_2 \lor \text{detected}_3)
\]

Property \( \phi_2 \) = Probability of detection and return to base.
\[
P = \mathcal{P}(\text{detected}_1 \land \text{dir}_1 = \text{back} \land x_1 = 0 \land y_1 = 1 \ldots)
\]

Property \( \phi_3 \) = Expected number of robots returning to base.
Overall Approach

Individual state machines are linked via communication between DTMCs.

Modal DTMC \(\{M_1, M_2, M_3\}\)

Probabilistic Model Checker for DTMCs

Validated by comparing predicted performance with measurements from actual team runs.
Technical Details

Modal DTMC
\{M_1, M_2, M_3\}

\textbf{Theorem 1} : \{M_1, M_2, M_3\} = \hat{M}

Projection of $M_1$ assuming model change at time $t_1$

Combine

PRISM
Computing $\langle M_i, t_j \rangle$

Physically run Kilobot $R_i$ and force it to turn around at $t_j$
  - Discretize time and space
  - Reprogram controller to “fake” mine detection at time $t_j$
  - Transition probability matrix of $\langle M_i, t_j \rangle$ is defined as:
    - $P(s, s') = \frac{n(s, s')}{n(s)}$
    - $n(s)$ = no. of times robot was in state $s$
    - $n(s, s')$ = no. of times robot moved from $s$ to $s'$ in one time step

Tedious to repeat these experiments using actual Kilobots
  - Use a simulator (VREP)
  - Tune parameters to reproduce behavior observed with real Kilobots

At least two sources of error
  - Finite number of observations & space and time discretization
  - Both will remain no matter how much effort we put in
  - How do we quantify and bound the error?
Error Quantification: Fuzzy Sampling

repeat this process to obtain \( \tilde{P} = \{ p_1, \ldots, p_n \} \)

Theorem 2. It has can be shown that \( \tilde{P} \) has the same distribution as the probability of a real number being the correct result given the evidence used to construct projections.

Compute the 90% credible interval of \( \tilde{P} \), i.e., the 5th and 95th percentile. Verify whether actual observations lie in this interval.

Projections Constructed \( P \)
Perturbed Projection Constructed using Dirichlet distributions with parameter \( P \)
Results: Probability that one Robot detected the mine and returned to the base = Success

<table>
<thead>
<tr>
<th>Team in Release Order</th>
<th>Observed</th>
<th>Predicted</th>
<th>Sample Mean</th>
<th>Sample 5%</th>
<th>Sample 95%</th>
</tr>
</thead>
<tbody>
<tr>
<td>3-2-1</td>
<td>1</td>
<td>0.96</td>
<td>0.96</td>
<td>0.91</td>
<td>0.99</td>
</tr>
<tr>
<td>4-6-1</td>
<td>0.97</td>
<td>0.96</td>
<td>0.96</td>
<td>0.91</td>
<td>0.99</td>
</tr>
<tr>
<td>4-6-2</td>
<td>0.47</td>
<td>0.43</td>
<td>0.43</td>
<td>0.29</td>
<td>0.58</td>
</tr>
<tr>
<td>5-6-2</td>
<td>0.5</td>
<td>0.43</td>
<td>0.43</td>
<td>0.28</td>
<td>0.61</td>
</tr>
<tr>
<td>5-6-7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6-1-7</td>
<td>0.93</td>
<td>0.96</td>
<td>0.96</td>
<td>0.91</td>
<td>0.99</td>
</tr>
<tr>
<td>6-5-7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7-3-5</td>
<td>0.7</td>
<td>0.83</td>
<td>0.83</td>
<td>0.72</td>
<td>0.92</td>
</tr>
<tr>
<td>7-3-6</td>
<td>0.83</td>
<td>0.83</td>
<td>0.84</td>
<td>0.74</td>
<td>0.92</td>
</tr>
<tr>
<td>7-6-1</td>
<td>0.9</td>
<td>0.96</td>
<td>0.96</td>
<td>0.92</td>
<td>0.99</td>
</tr>
</tbody>
</table>
## Results: Expected Number of Robots that Returned to the Base

<table>
<thead>
<tr>
<th>Team in Release Order</th>
<th>Observed</th>
<th>Predicted Oneshot</th>
<th>Sample Mean</th>
<th>Sample 5%</th>
<th>Sample 95%</th>
</tr>
</thead>
<tbody>
<tr>
<td>3-2-1</td>
<td>2.2</td>
<td>2.17</td>
<td>2.17</td>
<td>1.97</td>
<td>2.38</td>
</tr>
<tr>
<td>4-6-1</td>
<td>1.67</td>
<td>1.23</td>
<td>1.23</td>
<td>1.14</td>
<td>1.33</td>
</tr>
<tr>
<td>4-6-2</td>
<td>0.83</td>
<td>0.7</td>
<td>0.7</td>
<td>0.55</td>
<td>0.89</td>
</tr>
<tr>
<td>5-6-2</td>
<td>0.83</td>
<td>0.72</td>
<td>0.73</td>
<td>0.53</td>
<td>0.91</td>
</tr>
<tr>
<td>5-6-7</td>
<td>0.43</td>
<td>0.29</td>
<td>0.29</td>
<td>0.19</td>
<td>0.38</td>
</tr>
<tr>
<td>6-1-7</td>
<td>1.57</td>
<td>1.23</td>
<td>1.24</td>
<td>1.14</td>
<td>1.35</td>
</tr>
<tr>
<td>6-5-7</td>
<td>0.2</td>
<td>0.29</td>
<td>0.3</td>
<td>0.19</td>
<td>0.41</td>
</tr>
<tr>
<td>7-3-5</td>
<td>0.7</td>
<td>0.85</td>
<td>0.85</td>
<td>0.73</td>
<td>0.94</td>
</tr>
<tr>
<td>7-3-6</td>
<td>1.17</td>
<td>1.11</td>
<td>1.12</td>
<td>0.95</td>
<td>1.25</td>
</tr>
<tr>
<td>7-6-1</td>
<td>1.63</td>
<td>1.23</td>
<td>1.24</td>
<td>1.13</td>
<td>1.34</td>
</tr>
</tbody>
</table>

Team: HCCPS

SEI team members

- Bjorn Andersson, Ph.D.
- Sagar Chaki (co-lead), Ph.D.
- Dionisio de Niz (co-lead), Ph.D.
- Joseph Giampapa, M.S.
- Arie Gurfinkel, Ph.D.
- John Hudak, M.S.
- Mark Klein, M.S.
- Gabriel Moreno, M.S.
- Lutz Wrage, M.S.

Prior results

- FY11, FY12, FY13 HCCPS line
- FY11, FY12 LENS

Collaborators

- Prof. Marsha Chechik, Univ. of Toronto
- Prof. Ed Clarke, CMU/CS
- Prof. Lui Sha, UIUC
- Prof. John Lehoczky, CMU/Stat
- Prof. Raj Rajkumar, CMU/ECE
- Prof. Anthony Rowe, CMU/ECE
- Prof. Paul Scerri, CMU/RI
- Prof. Natasha Sharygina, Univ. of Lugano
- Prof. Ofer Strichman, Technion, Israel
- Prof. Paulo Tabuada, UCLA

Engaged Stakeholders

- LMCO Russell Kegley, Model problem
- LMCO Jonathan Preston, Model problem
Contact Information Slide Format

Dionisio de Niz
Senior MTS
SSD/CSC
Telephone:  +1 412-268-9002
Email:  dionisio@sei.cmu.edu

U.S. Mail
Software Engineering Institute
Customer Relations
4500 Fifth Avenue
Pittsburgh, PA 15213-2612
USA

Web
www.sei.cmu.edu
www.sei.cmu.edu/contact.cfm

Customer Relations
Email: info@sei.cmu.edu
Telephone:  +1 412-268-5800
SEI Phone:  +1 412-268-5800
SEI Fax:  +1 412-268-6257