Implementing and Verifying EDF Preemption-Level Resource Control

Mark Louis Fairbairn and Alan Burns
Department of Computer Science, University of York, UK

Overview

- Earliest Deadline First
- Protected Objects
- Stack Resource Protocol
- Justification for verification
- Toolchain
- Results

Earliest Deadline First

- Typical Parameters
  1. Tasks (i)
  2. Start Time (T)
  3. Period (P)
  4. Deadline (D)
  5. Worst Case Execution Time (C)

$$U = \sum_{i=1}^{n} \frac{C_i}{P_i} \leq 1$$

Shared Resources

- Want to block simultaneous access
- Access from within a Protected Object (PO)
- Want to bound Priority inversion
- For fixed priority systems we use Priority Ceiling Protocol
- For EDF we use Stack Resource Protocol

L. Sha, R. Rajkumar and J. P. Lehoczky, "Priority inheritance protocols: An approach to real-time synchronization," *IEEE Transactions on Computers,* vol. 39, no. 9, pp. 1175-1185, 1990.

T. P. Baker, "A Stack-Based Resource Allocation Policy for Realtime Processes," *IEEE Real-Time Systems Symposium,* pp. 191-200, 1990.

Bakers Stack Resource Protocol

- Priorities are replaced by preemption levels
- All tasks are assigned a preemption level. The lowest relative deadline tasks are assigned the highest preemption level
- Each shared resource has an associated ceiling level
- There is a system ceiling – highest locked resource
- A task is only allowed to preempt when its absolute deadline is less than the currently executing task and its preemption level is higher than the current system ceiling

ADA 2005 Correction

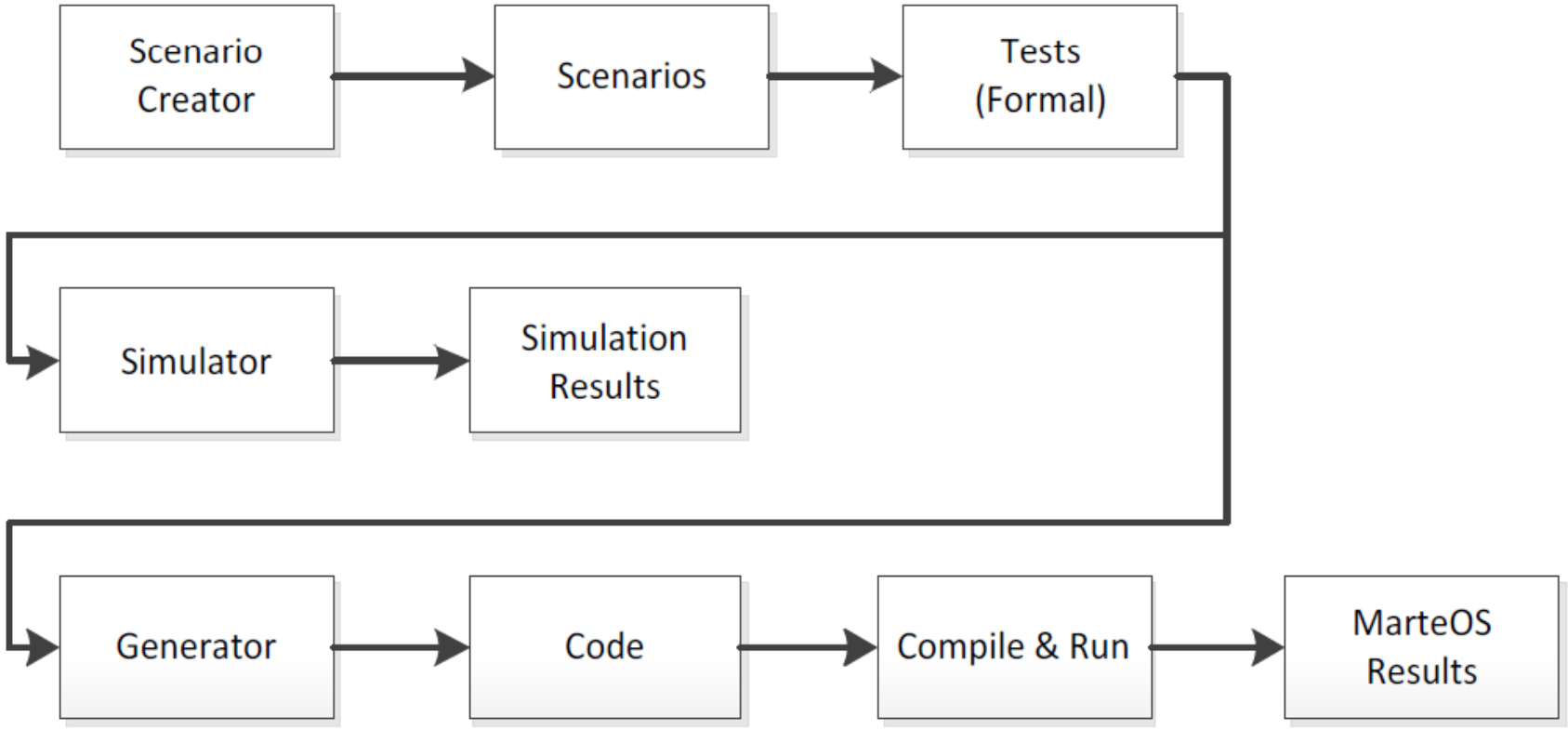
- *Base priority of the new task is a combination of rules,*
- *‘the highest priority P, if any, less than the base priority of T such that one or more tasks are executing within a protected object with ceiling priority P and task T has an earlier deadline than all such tasks’*
- *‘; and furthermore T has an earlier deadline than all other tasks on ready queues with priorities in the given EDF_Across_Priorities range that are strictly less than P.‘*

A. Zerzelidis, A. Burns and A. J. Wellings, "Correcting the EDF protocol in Ada 2005," in *13th international workshop on Real-time Ada*, Vermont, 2007.

Why Verify?

- Need to check the edge cases
- But we don't know what the edge cases are
- So we need to find them

Toolchain



M. L. Fairbairn, *An Assessment of Ada's new EDF Facilities,* MEng Report, University of York, UK, 2010 available at http://www-users.cs.york.ac.uk/~burns/FairbairnReport.pdf

RTS York

THE UNIVERSITY of York

Scenario Generator

- Generates combinations of 4 possible actions

1. Task creation
2. Task departure
3. Task promotion (when entering a protected object, PO)
4. Task demotion (leaving a PO)

Scenario Generator

- PO depth limit
- Filters repeats
- In the background it checks the numbers
- Asks user if it thinks its going the wrong way

Formal Task Representation

Task (TaskName, StartTime, Deadline, Period) { Code }

```
Task (T2, 2, 20, 30) {
        2
        R1 {
                2
                R2 {
                        2
                }
                R3 {
                        1
                }
        }
        2
}
```

RTS York

THE UNIVERSITY of York

Scenario Simulator

- Runs the original Baker's rules
- Computationally expensive
  - Rules re-evaluated at every tick
- Times are all integer based
- Generates the expected output

Generation, Compilation & Execution

- Converts formal specification to Ada code
  - Start times and priorities assigned
- Compiles and runs on i486 target machine with MarteOS
- Relatively long execution times are implemented by busy-waits
- Output is saved

Comparator

- Another tool cross-references the outputs from MarteOS and from the simulator
  - with user help

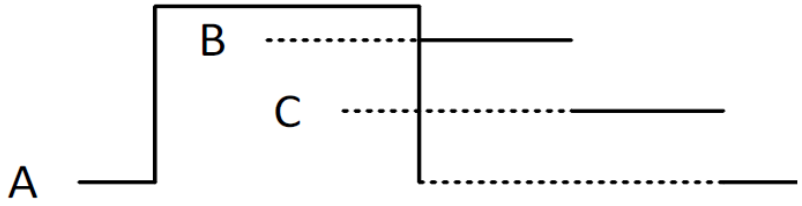
- Simulator – 4 in TA PoA
- MarteOS – 4 in TA A P4

- Flags errors

Results – Scenario 6a

```
0:A
1:A
1:A 0:B
1:A 0:BC
0:BCA
0:CA
0:A
```

```
Task(TA,0,100) {
        2
        A {
                8
        }
        2
}
Task(TB,4,50) {
        4
        A {
                2
        }
        2
}
Task(TC,6,80) {
        10
}
```



Results – Scenario 6a

Simulator Results	MarteOS Results
0 in TA 2 in TA PoA 10 out TA PoA 10 in TB 14 in TB PoA 16 out TB PoA 18 out TB 18 in TC 28 out TC 30 out TA	0 in TA 2 in TA A P3 10 out TA A P3 10 in TB 14 in TB A P10 16 out TB A P10 18 out TB 18 in TC 28 out TC 30 out TA

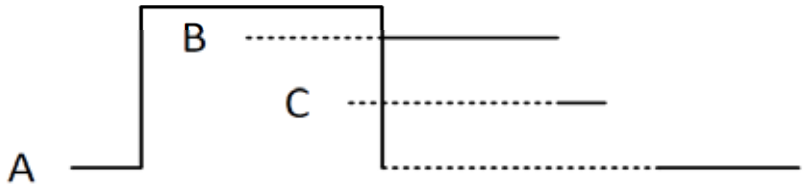
Results

- Ada05 bug is gone
- But unfortunately another was found...

Results – Scenario 6b

```
0:A
1:A
1:A 0:B
1:A 0:BC
0:BCA
0:CA
0:A
```

```
Task(TA,0,100) {
        1
        A {
                7
        }
        2
}
Task(TB,2,60) {
        1
        A {
                7
        }
        2
}
Task(TC,6,58) {
        10
}
```



Results – Scenario 6b

Simulator Results	MarteOS Results
0 in TA 1 in TA PoA 8 out TA PoA 8 in TB 9 in TB PoA 16 out TB PoA 18 out TB 18 in TC 28 out TC 30 out TA	0 in TA 1 in TA A P3 8 out TA A P3 8 in TB 9 in TC 19 out TC 19 in TB A P10 26 out TB A P10 28 out TB 30 out TA

Summary

- Specific test cases are only partially useful
- Baker's algorithm is complex
- Extensive automated testing is relatively cheap to do
  - We have demonstrated how this can be done
- But state space must be adequately explored

- Bug was found – and is now fixed