Towards A More Practical Model for Mixed Criticality Systems

Alan Burns and Sanjoy Baruah

University of York, UK

University of North Carolina, USA

Standard Model - 1

- A mixed criticality system is defined to execute in either of two modes: a HI-crit mode or a LO-crit mode
- Each task is characterised by:
  - L (equal to either LO or HI)
  - T and D, period and deadline
  - $C(HI)$ and $C(LO)$, execution times, with $C(HI) \geq C(LO)$

Standard Model - 2

- The system starts in the LO-crit mode, and remains in that mode as long as all jobs execute within their low criticality computation times ($C(LO)$)
- If any job executes for its $C(LO)$ execution time without completing then the system immediately moves to the HI-crit mode
- As the system moves to the HI-crit mode all LO-crit tasks are abandoned. No further LO-crit jobs are executed

RTS York

Standard Model - 3

- The system remains in the HI-crit mode
- Tasks are assumed to be independent of each other (they do not share any resource other than the processor)

Important Note

- LO-crit is still *critical*
- If all $C(LO)$ values are safe, then the system never leaves LO-crit mode
- However, some $C(LO)$ values may not be safe, and
- We need to address the harshness of 'abandon all LO-crit tasks'

More Realistic Assumptions

- LO-crit jobs should not be aborted
- LO-crit tasks should survive in some sense
- The LO-crit mode is eventually returned to
- Tasks are not independent of each other – see paper at ReTiMiCS
- More than two criticality levels – see next paper here
- LO-crit tasks are constrained to execution for at most $C(LO)$
  - i.e. mode change is only triggered by HI-crit jobs

AMC-rtb Test (LO)

$$R_i(LO) = C_i(LO) \; + \sum_{j \in \mathbf{hp}(i)} \left\lceil \frac{R_i(LO)}{T_j} \right\rceil C_j(LO)$$

AMC-rtb Test (HI)

$$R_i(HI) = C_i(HI) + \sum_{\tau_j \in \mathbf{hpH}(i)} \left\lceil \frac{R_i(HI)}{T_j} \right\rceil C_j(HI)$$

$$+ \sum_{\tau_k \in \mathbf{hpL}(i)} \left\lceil \frac{R_i(LO)}{T_k} \right\rceil C_k(LO)$$

Limitation/Benefits of Analysis

- All released LO-crit jobs are assumed to complete
- So no need to abort jobs during their execution
  - though these jobs may not meet their deadlines

Alternative Models – on mode change:

- Reduce priority of LO-crit tasks
- Reduce execution-time of LO-crit tasks
- Increase period/deadline of LO-crit tasks
- Allow LO-crit tasks to inherent slack from under-utilising HI-crit tasks
- Then return to LO-crit mode on idle tick

RTS York

Reduced execution time

- For HI-crit tasks: $C(HI) \geq C(LO)$

Reduced execution time

- For HI-crit tasks: $C(HI) \geq C(LO)$
- For LO-crit tasks: $C(HI) \leq C(LO)$

Reduced execution time

- For HI-crit tasks: $C(HI) \geq C(LO)$
- For LO-crit tasks: $C(HI) \leq C(LO)$
- For some tasks $C(HI) = 0$

Analysis for HI-crit

$$R_i(HI) = C_i(HI) + \sum_{\tau_j \in \mathbf{hpH}(i)} \left\lceil \frac{R_i(HI)}{T_j} \right\rceil C_j(HI) +$$

$$\sum_{\tau_k \in \mathbf{hpL}(i)} \left\lceil \frac{R_i(LO)}{T_k} \right\rceil C_k(LO) +$$

$$\sum_{\tau_k \in \mathbf{hpL}(i)} \left(\left\lceil \frac{R_i(HI)}{T_k} \right\rceil - \left\lceil \frac{R_i(LO)}{T_k} \right\rceil \right) C_k(HI)$$

Analysis for HI-crit

$$R_i(HI) = C_i(HI) + \sum_{\tau_j \in \mathbf{hpH}(i)} \left\lceil \frac{R_i(HI)}{T_j} \right\rceil C_j(HI) +$$

$$\sum_{\tau_k \in \mathbf{hpL}(i)} \left\lceil \frac{R_i(LO)}{T_k} \right\rceil (C_k(LO) - C_k(HI)) +$$

$$\sum_{\tau_k \in \mathbf{hpL}(i)} \left\lceil \frac{R_i(HI)}{T_k} \right\rceil C_k(HI)$$

Analysis for HI-crit

$$R_i(HI) = C_i(HI) + \sum_{\tau_j \in \mathbf{hp}(i)} \left\lceil \frac{R_i(HI)}{T_j} \right\rceil C_j(HI) +$$

$$\sum_{\tau_k \in \mathbf{hpL}(i)} \left\lceil \frac{R_i(LO)}{T_k} \right\rceil (C_k(LO) - C_k(HI))$$

RTS York

Analysis for LO-crit task in LO-crit mode

$$R_i^*(LO) \; = \; C_i(HI) \; + \sum_{j \in \mathbf{hp}(i)} \left\lceil \frac{R_i(LO)}{T_j} \right\rceil C_j(LO)$$

Analysis for LO-crit task in HI-crit mode

$$R_i(HI) = C_i(HI) + \sum_{\tau_j \in \mathbf{hpH}(i)} \left\lceil \frac{R_i(HI)}{T_j} \right\rceil C_j(HI) +$$

$$\sum_{\tau_k \in \mathbf{hpL}(i)} \left\lceil \frac{R_i^*(LO)}{T_k} \right\rceil C_k(LO) +$$

$$\sum_{\tau_k \in \mathbf{hpL}(i)} \left(\left\lceil \frac{R_i(HI)}{T_k} \right\rceil - \left\lceil \frac{R_i^*(LO)}{T_k} \right\rceil \right) C_k(HI)$$

RTS York

Analysis for LO-crit task in HI-crit mode

$$R_i(HI) = C_i(HI) + \sum_{\tau_j \in \mathbf{hp}(i)} \left\lceil \frac{R_i(HI)}{T_j} \right\rceil C_j(HI) +$$

$$\sum_{\tau_k \in \mathbf{hpL}(i)} \left\lceil \frac{R_i^*(LO)}{T_k} \right\rceil (C_k(LO) - C_k(HI))$$

RTS York

Alternative Models – use together:

- Reduce execution-time of LO-crit tasks, and
- Increase period/deadline of LO-crit tasks

Alternative Models – use together:

- Reduce execution-time of LO-crit tasks
- Increase period/deadline of LO-crit tasks
- Allow LO-crit tasks to inherent slack from under-utilising HI-crit tasks

RTS York

Alternative Models – use together:

- Reduce execution-time of LO-crit tasks
- Increase period/deadline of LO-crit tasks
- Allow LO-crit tasks to inherent slack from under-utilising HI-crit tasks
- Reduce priority of LO-crit tasks

Capacity Inheritance

- Capacity Sharing
- Extended Priority Exchange
- History Rewriting

Robustness

- Another practical issue is increasing system robustness
- For example, use sensitivity analysis with the schedulability tests to increase $C(LO)$s and $C(HI)$s
- And allow priorities to change as part of this process

Conclusion

- Our models must be realistic
- MCS theory must deal with the survival of LO-crit tasks following a mode change
- A number of schemes are possible
- This paper has concentrated on reducing execution time
- Paper has also addressed robust priority assignment, and capacity inheritance

Future Work

- Looking at more aggressive methods of returning system back to LO-crit mode
- Note review on mixed criticality research on my home page, and on the home page of our MCC project