Semi-Partitioned Cyclic Executives for Mixed Criticality Systems

Alan Burns, University of York, UK

Sanjoy Baruah, University of North Carolina, USA

Introduction

- Many safety-critical systems are implemented using a cyclic executive
- A series of frames (or minor cycles) are executed in series
- The set of (repeating frames) is called the major cycle
- Periods are constrained (at design time) to be multiples of the minor cycle time

Introduction

- On a multi-core platform, a collection of frames (one per code) will execute in parallel
- Each core will switch from one minor cycle to the next in a synchronised way
- Application code is partitioned if each job only executes on one core/frame, or global if jobs can migrate between cores
- In this talk I consider globally scheduling, in the next talk Tom will consider partitioned systems

Introduction

- Giannopoulou's model for mixed criticality requires all cores to be only executing code of the same criticality at the same time
- Hence changes to the 'mode' of the platform must also be synchronised
- That is: all cores first execute most critical code of the first minor cycle, then next critical etc, then move to second minor cycle and repeat

Our Contribution

- Improve on the schedulability test presented at last year's WMC
- This paper is restricted to single cycle, multi-frame applications
- That is jobs not tasks

Two Criticality Levels

- Usual HI and LO
- Usual $C(HI)$ and $C(LO)$
- Define $C(EX)$ to be $C(HI) - C(LO)$
- As discussed last year we use McNaughton's optimal allocation scheme

Simple Scheme

- Define a switch point, S
- HI-crit code executes from 0 to S
- LO-crit code executes from S to D
- But, HI-crit code can executes from 0 to D
- Where D is length of minor cycle

Analysis, LO in LO

We compute a number of makespans

$$\Delta^{\mathrm{LO}} \overset{\text{def}}{=} \max\left(\frac{\sum_{\chi_i=\mathrm{LO}} C_i(\mathrm{LO})}{m}, \max_{\chi_i=\mathrm{LO}}\{C_i(\mathrm{LO})\}\right)$$

At most $m-1$ jobs are split, hence ‘semi-partitioned’

Analysis, HI in LO

Similarly

$$S^{\min} \stackrel{\text{def}}{=} \max\left(\frac{\sum_{\chi_i=\text{HI}} C_i(\text{LO})}{m}, \max_{\chi_i=\text{HI}}\{C_i(\text{LO})\}\right)$$

Analysis, HI in HI

$$\Delta^{\mathrm{HI}} \stackrel{\text{def}}{=} \max\left(\frac{\sum_{\chi_i=\mathrm{HI}} C_i(\mathrm{EX})}{m}, \max_{\chi_i=\mathrm{HI}}\{C_i(\mathrm{EX})\}\right)$$

And so we require

$$S^{\min} + \max\left(\Delta^{\mathrm{LO}}, \Delta^{\mathrm{HI}}\right) \leq D$$

Improving Schedulability

- If we fail because $S^{\min} + \Delta^{\mathrm{HI}} > D$
- Move computation from $C(HI)$ to $C(LO)$
- Fill in holes before $S^{\min}$
- And hence reduce Δ^{HI}

$S^{\min}$

$$S^{\min} \stackrel{\text{def}}{=} \max\left(\frac{\sum_{\chi_i=\text{HI}} C_i(\text{LO})}{m}, \max_{\chi_i=\text{HI}}\{C_i(\text{LO})\}\right)$$

So if:

$$\frac{\sum_{\chi_i=\text{HI}} C_i(\text{LO})}{m} < \max_{\chi_i=\text{HI}}\{C_i(\text{LO})\}$$

Move Code

And if

$$\frac{\sum_{\chi_i=\mathrm{HI}} C_i(\mathrm{EX})}{m} < \max_{\chi_i=\mathrm{HI}}\{C_i(\mathrm{EX})\}$$

then choose largest $C_i(EX)$ and move some C to $C_i(LO)$ subject to $C_i(LO) \leq S^{\min}$

Be prepared to increase $S^{\min}$

Example, D=8, m=3

	χ_i	$C_i(\text{LO})$	$C_i(\text{HI})$	$C_i(\text{HI}) - C_i(\text{LO})$
j_1	LO	3	-	-
j_2	LO	2	-	-
j_3	LO	2	-	-
j_4	HI	2	7	5
j_5	HI	3	7	4
j_6	HI	3	3	0
j_7	HI	4	4	0

Example

- Ignoring criticality R=7+3=10
- $\Delta^{\text{LO}} = 3$
- $S^{\text{min}} = 4$
- $\Delta^{\text{HI}} = 5$
- $R = 5 + 4 = 9$

Improvement, D=8, m=3

	χ_i	$C_i(\text{LO})$	$C_i(\text{HI})$	$C_i(\text{HI}) - C_i(\text{LO})$
j_1	LO	3	-	-
j_2	LO	2	-	-
j_3	LO	2	-	-
j_4	HI	4	7	3
j_5	HI	4	7	3
j_6	HI	3	3	0
j_7	HI	4	4	0

Improvement

- $\Delta^{\mathrm{LO}} = 3$
- $S^{\mathrm{min}} = 5$
- $\Delta^{\mathrm{HI}} = 3$
- $R = 5 + 3 = 8$

A further improvement coming from a more flexible implementation is considered in the paper

Extended Model

- With, for example, 4 criticality levels, but two computation times:
  - V=4 – L_4 is lowest, L_1 is highest
  - L_1 has $C(L_1)$ and $C(L_4)$
  - L_2 has $C(L_2)$ and $C(L_4)$
  - L_3 has $C(L_3)$ and $C(L_4)$
  - L_4 has only $C(L_4)$
  - We use $C_i(\text{SF})$ and $C(\text{NL})$ in this paper

Extended Model

- Switch points S^1 to S^{V-1}
- Add S^0 and S^V
- If each job $j_i \in L_i$ executes for no more than $C_i(\text{NL})$, then all the jobs in the set L_i must fit into the interval (S^{i-1},S^i]
- If each job $j_i \in L_i$ executes for no more than $C_i(\text{SF})$, then all the jobs in the set L_i must fit into the interval (S^{i-1},S^V]

Analysis

First we compute minimum makespan for criticality level L_1:

$$S_1^{\min} \stackrel{\text{def}}{=} \max\left(\frac{\sum_{j_i \in L_1} C_i(\text{NL})}{m}, \max_{j_i \in L_1}\{C(\text{NL})\}\right)$$

Next we compute Δ^1 and check that $S_1^{\min} + \Delta^1$ is no greater than S^V $(= D)$:

$$\Delta^1 \stackrel{\text{def}}{=} \max\left(\frac{\sum_{j_i \in L_1} C_i(\text{EX})}{m}, \max_{j_i \in L_1}\{C_i(\text{EX})\}\right)$$

Analysis

If $S_1^{\min} + \Delta^1 > S^V$ then work must be brought forward so that $S_1^{\min}$ is increased but Δ^1 is decreased by a greater amount

This defines S^1

Analysis

Process is repeated for each criticality level, L_i using:

$$S_i^{\min} \stackrel{\text{def}}{=} \max\left(\frac{\sum_{j_i \in L_i} C_i(\mathrm{NL})}{m}, \max_{j_i \in L_i}\{C_i(\mathrm{NL})\}\right)$$

and

$$\Delta^i \stackrel{\text{def}}{=} \max\left(\frac{\sum_{j_i \in L_i} C_i(\mathrm{EX})}{m}, \max_{j_i \in L_i}\{C_i(\mathrm{EX})\}\right)$$

Analysis

with the conditions

$$S^{i-1} + S_i^{\min} + \Delta^i \leq S^V$$

and for all jobs of criticality L_i

$$C_i(\text{NL}) \leq S_i^{\min}$$

At all stages, modification to $C_i(\text{NL})$ (and hence $C_i(\text{EX})$) are made to ensure these two conditions are met. This fixes S^i.

Example

- A further example in the paper has 12 jobs, 2 cores and 4 criticality levels
- It shows that a schedule of length 20 is possible
- Ignoring criticality leads to a schedule length of 48

Optimality

- An allocation scheme (of jobs to frames) is *optimal* if it leads to the smallest possible switching points and a schedulable system.
- This notion of optimal is intuitive as for each criticality level the earliest switching point maximises the time available for the lower criticality levels
- The scheme produces the optimal value for each switching point, S_i (see paper)

Conclusion

- Single processor safety-critical systems are often constrained so that they can be implemented as a series of frames in a repeating cyclic executive
- In this paper we have extended this approach to incorporate multi-core platforms and mixed criticality applications
- We allow a minimum number of jobs to be split across the frames, and propose a practical means of constructing the necessary cyclic schedule