Migrating Mixed Criticality Tasks within a Cyclic Executive Framework

Alan Burns
Department of Computer Science
University Of York, UK
alan.burns@york.ac.uk

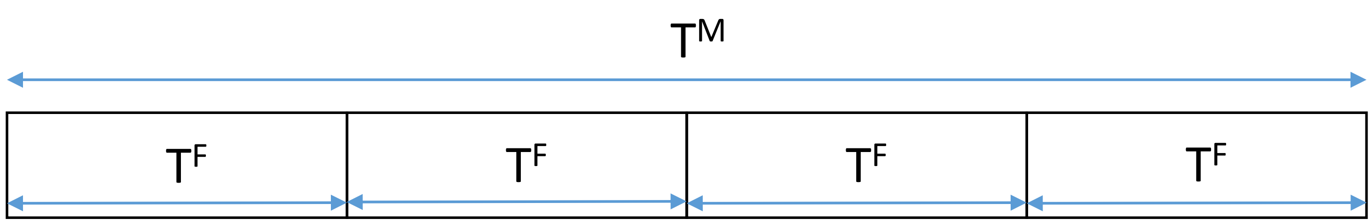
Sanjoy Baruah
Department of Computer Science
University of North Carolina, US

Introduction

- Two challenges face current developers
  - The move to multi- and many-core platforms
  - The integration of multiple applications on to the same platform
    - And those applications being of different criticality levels
  - In this talk we address these challenges within the context of the use of cyclic executives for implementation

The System Model:
Cyclic Executives

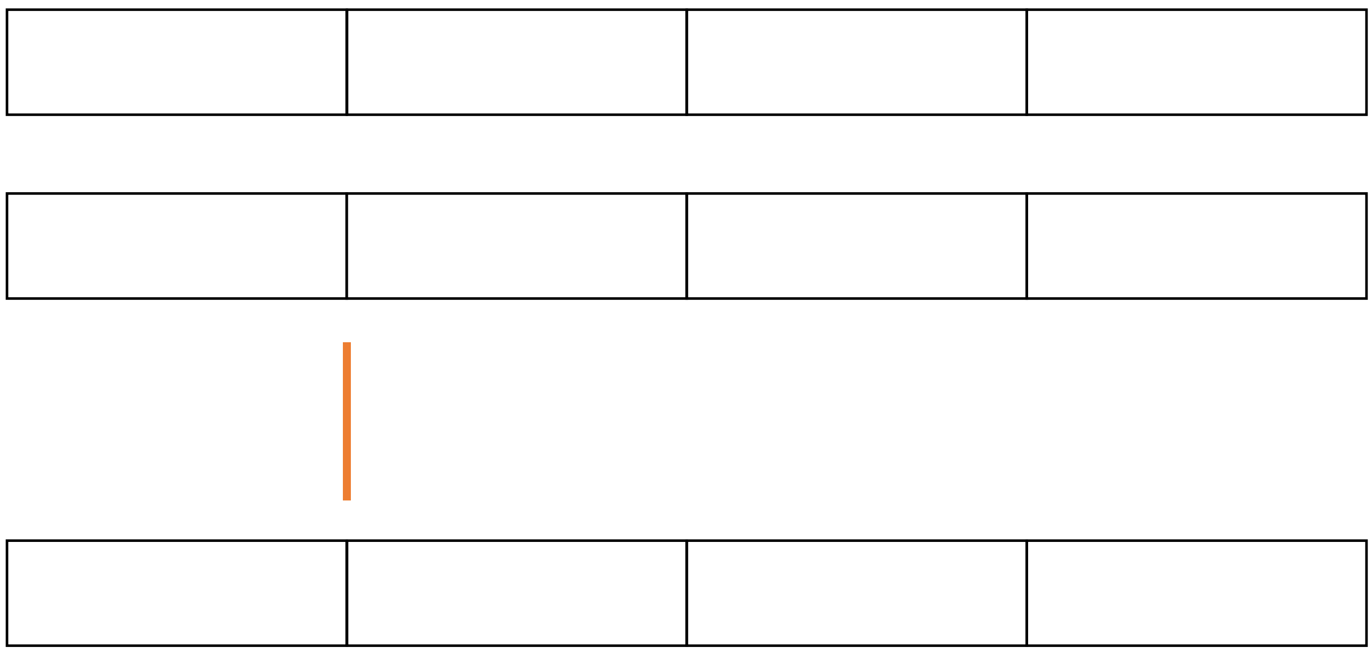
- A well known deterministic scheduling policy
- Pre-computed static schedules
- Structured around a Major Cycle T^M, composed of a number of Minor Cycles T^F:



Cyclic Executives

- CEs have a number of drawbacks:
  - Only really supports time-triggered code
  - Only supports a limited set of periods
  - Requires large computation time to be split between minor cycles
- But
  - Is very deterministic

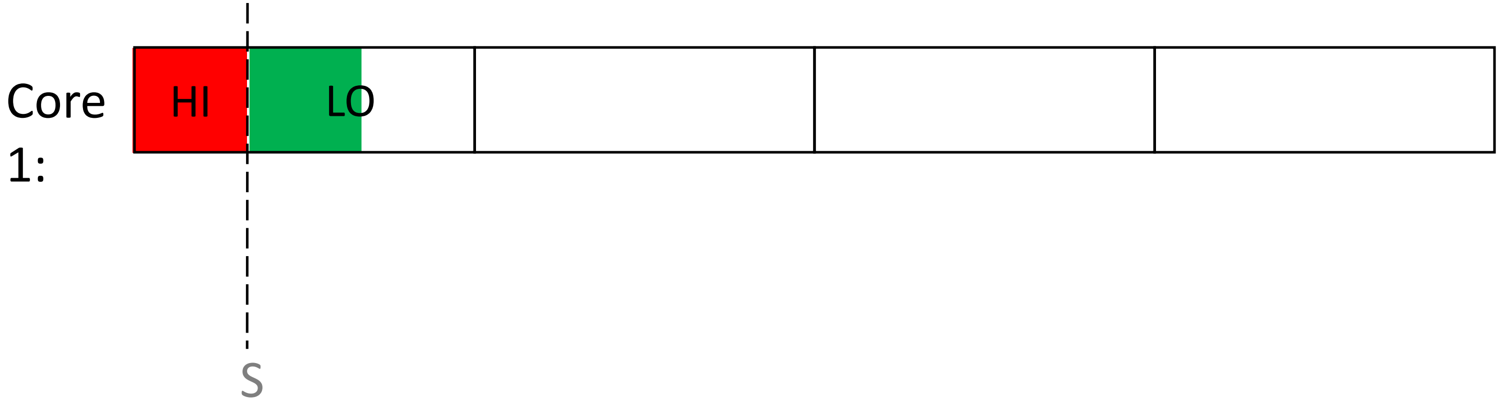
Multi-core
Cyclic Executives



Mixed Criticality

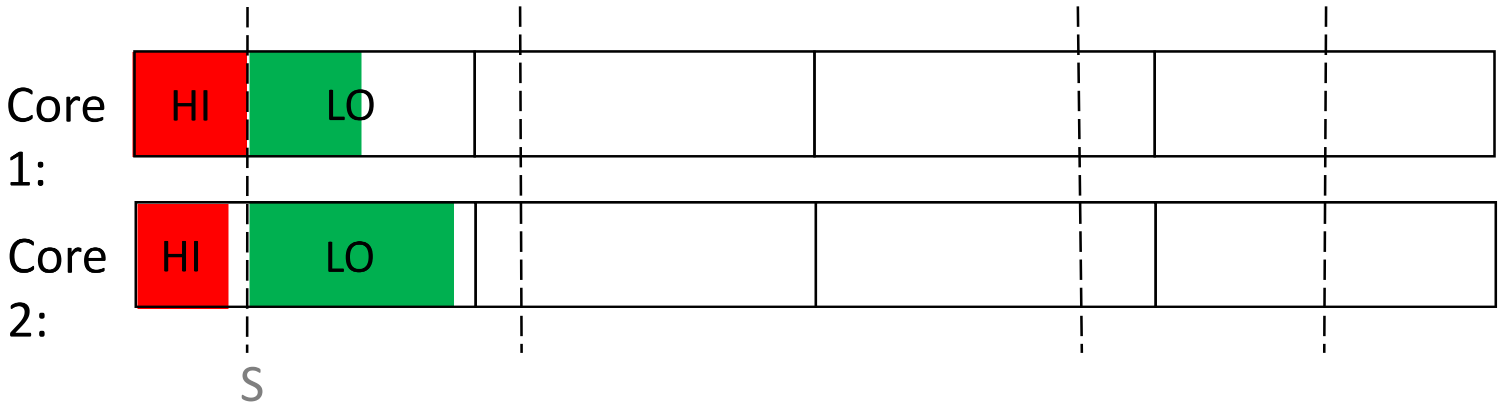
Cyclic Executive

- S is the point of mode change
- Two criticality levels, HI and LO



Mixed Criticality
Cyclic Executive

- S is the maximum point across all cores during any given minor cycle



Barriers

- The synchronised relationship between minor cycles and mode changes within frames can be implemented by either:
  - Timing events, or
  - Barriers

- When each core has completed its HI-crit work it signals the barrier
- When all cores have signalled the barrier, all cores move on to LO-crit work

Allocating code to frames

- For a partitioned system it is NP_hard to optimally allocate tasks to frames
  - We have demonstrated elsewhere the effective use of ILP
- For a non-partitioned system a task can be slit between frames
  - The scheme is optimal
  - The scheme is polynomial
  - Known as McNaugton algorithm (1959)

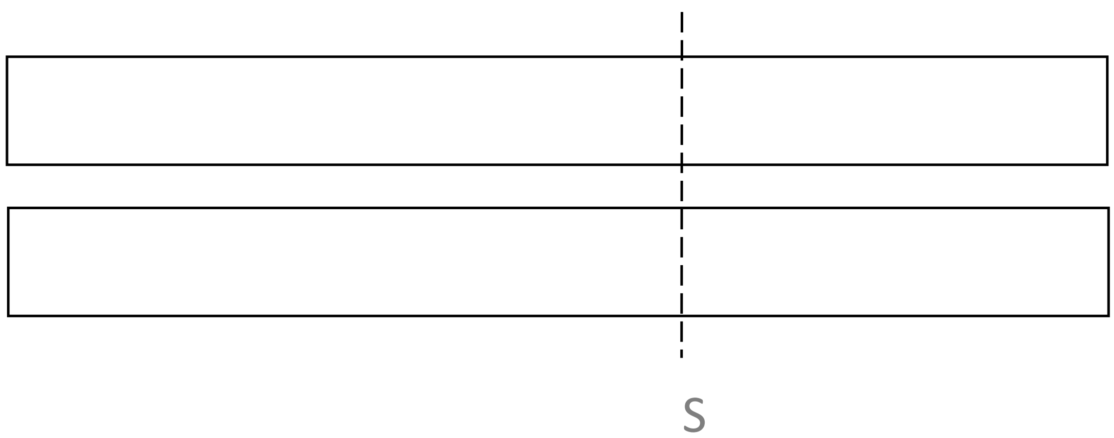
Deriving the value of S

$$S = \max\left(\frac{\sum_{i=1}^{n} C_i}{m}, \max(C_i)\right)$$

Where m is the number of nodes and C_i is the worst-case execution time of the ith task

Mixed Criticality
Cyclic Executive

- S is the point of mode change



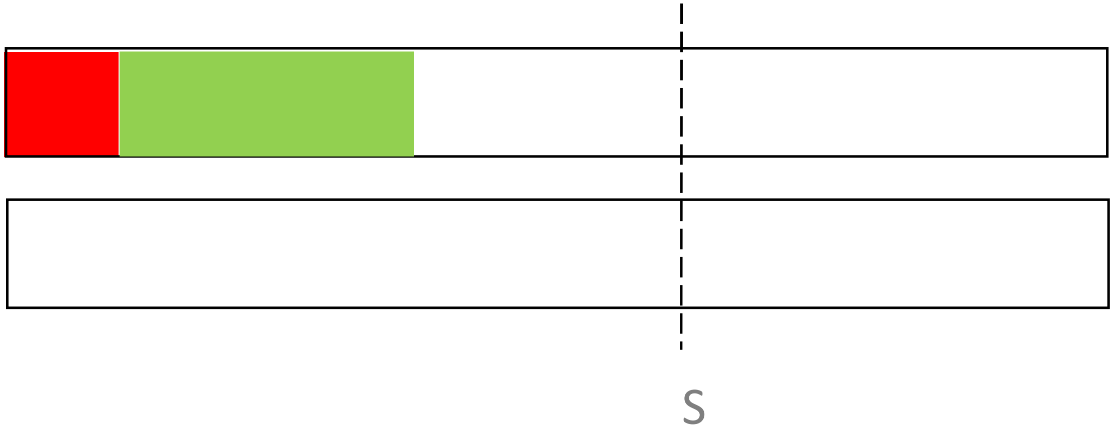
Mixed Criticality
Cyclic Executive

- S is the point of mode change

Mixed Criticality

Cyclic Executive

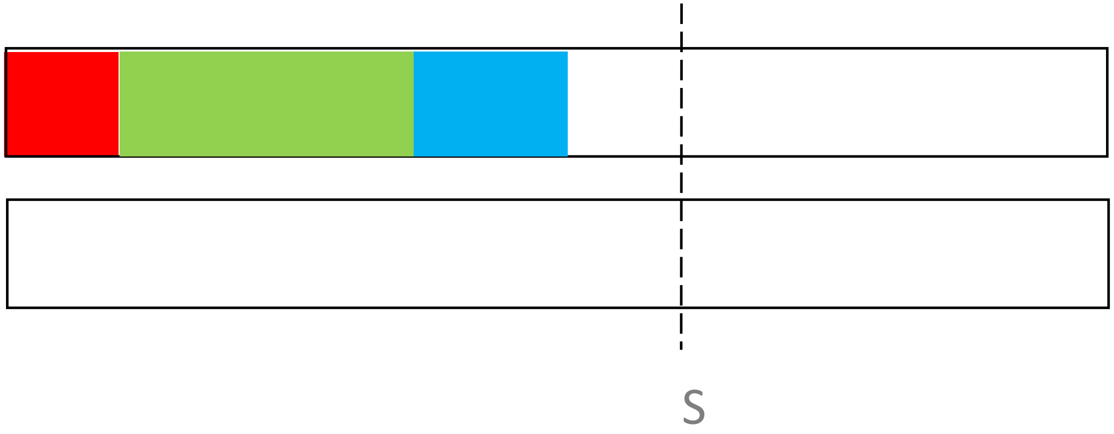
- S is the point of mode change



Mixed Criticality

Cyclic Executive

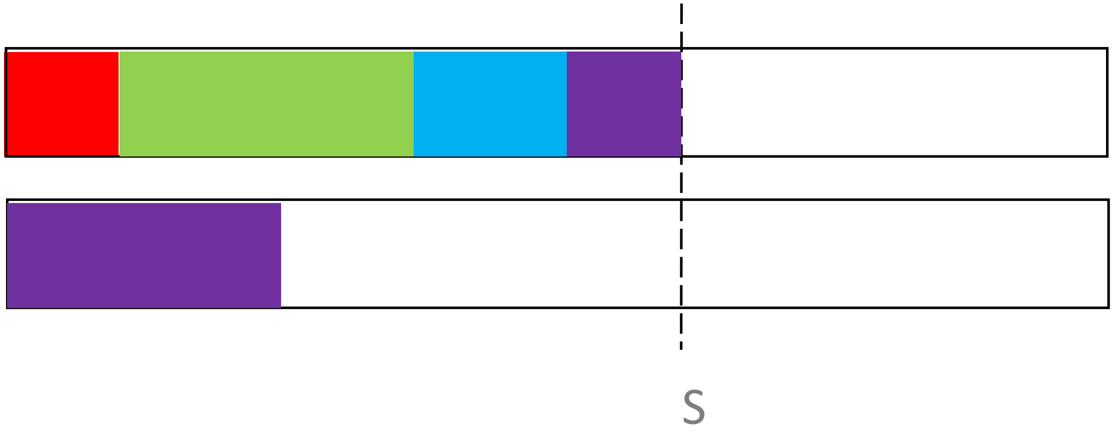
- S is the point of mode change



Mixed Criticality

Cyclic Executive

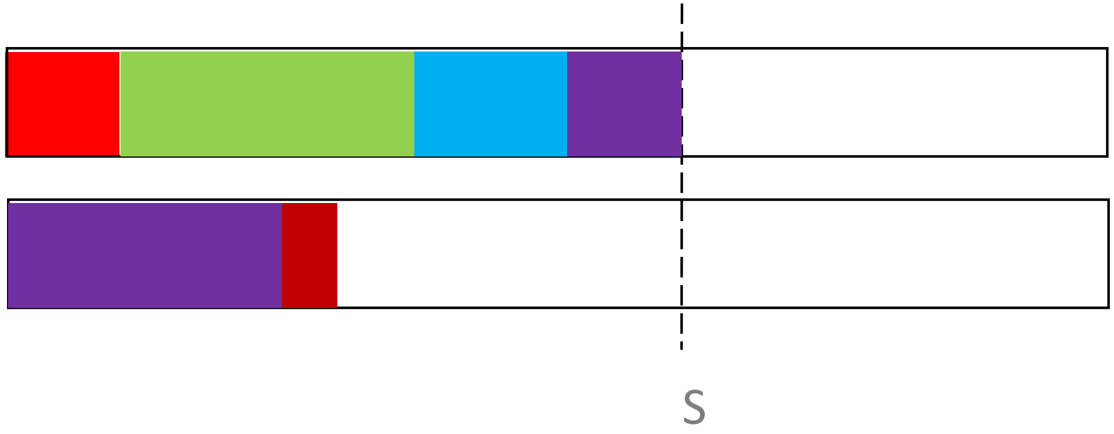
- S is the point of mode change



Mixed Criticality

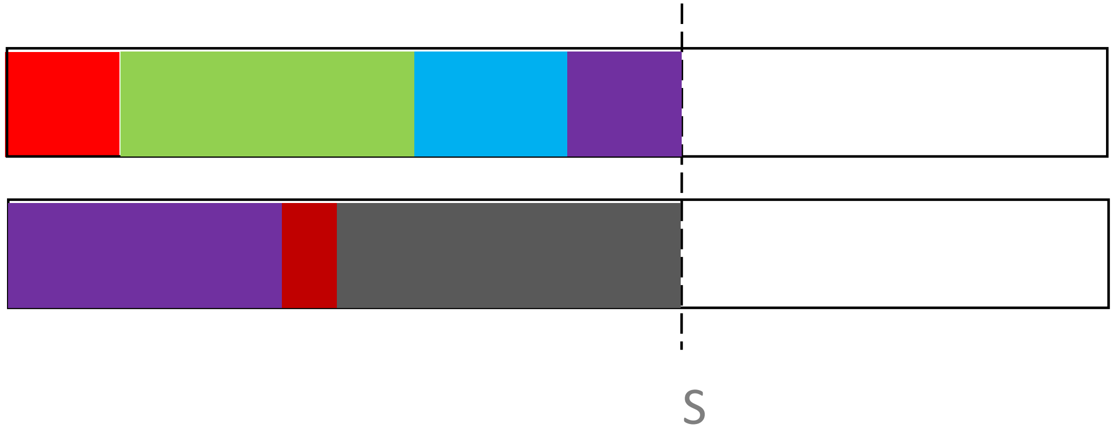
Cyclic Executive

- S is the point of mode change



Mixed Criticality
Cyclic Executive

- S is the point of mode change



Utilising Mixed Criticality

- The estimates of worst-case execution time (WCET) for High criticality code is often pessimistic (but certifiable)
- Lower estimates are safe but not certifiable
- So introduce two estimates of WCET:
  - C(LO) and C(HI), with C(LO) < C(H)
- Constraints are now
  - All HI-crit tasks to complete before S if C(LO) estimates are valid
  - All HI-crit tasks to complete before Tf if C(HI) estimates are valid

Impact for LO-crit tasks

- If a HI-crit task executes for more than $C(LO)$ then LO-crit tasks may not execute, or at least may not complete; but
- If all HI-crit task executes for no more than $C(LO)$ then LO-crit tasks must fit into the second half of the frame, i.e. within $T^F - S$
- Let $C(EX) = C(HI) - C(LO)$

Checking for Schedulability

$$S = \max\left(\frac{\sum_{i=1}^{nH} C_i(LO)}{m}, \max(C_i(\mathrm{LO}))\right)$$

$$X = \max\left(\frac{\sum_{i=1}^{nH} C_i(EX)}{m}, \max(C_i(\mathrm{EX}))\right)$$

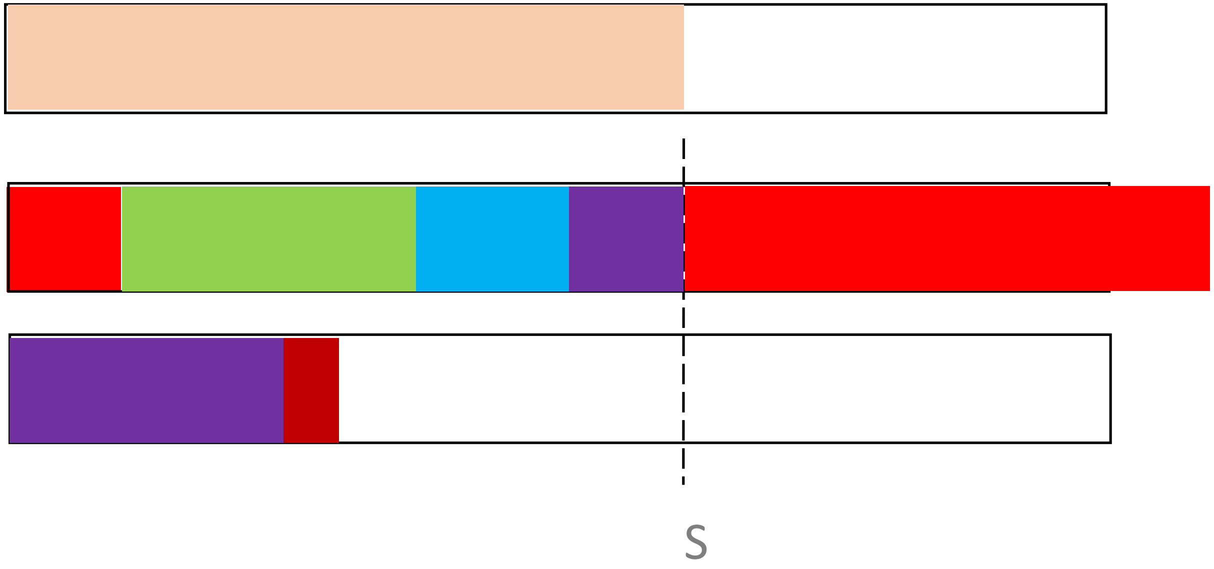
Checking for Schedulability

$$Y = \max(\frac{\sum_{i=1}^{nL} C_i(LO)}{m}, \max(C_i(\mathrm{LO})))$$

$$S + X \ \leq T^F$$
$$S + Y \ \leq T^F$$

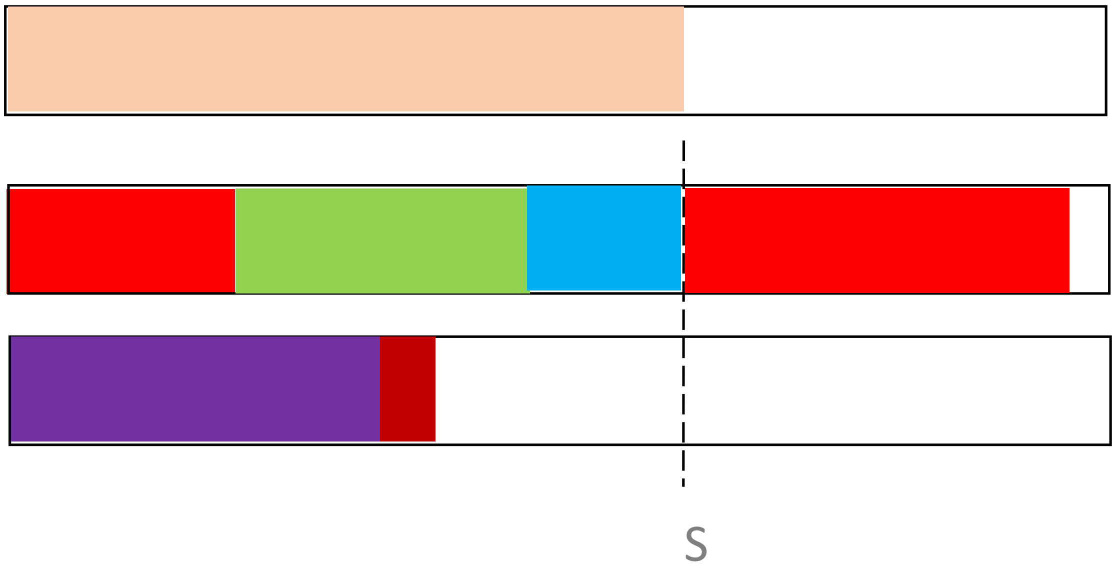
Mixed Criticality
Cyclic Executive

- Overrun in HI-crit mode



Mixed Criticality
Cyclic Executive

- Bring code forward from C(EX) to C(LO)



Optimal S values

- In the paper we show how an LP formulation can be used to find the optimal S values in polynomial time
- The paper also shows how task with larger computation times but longer periods can be split
  - The splitting of HI_crit tasks is not as straightforward as splitting LO-crit tasks

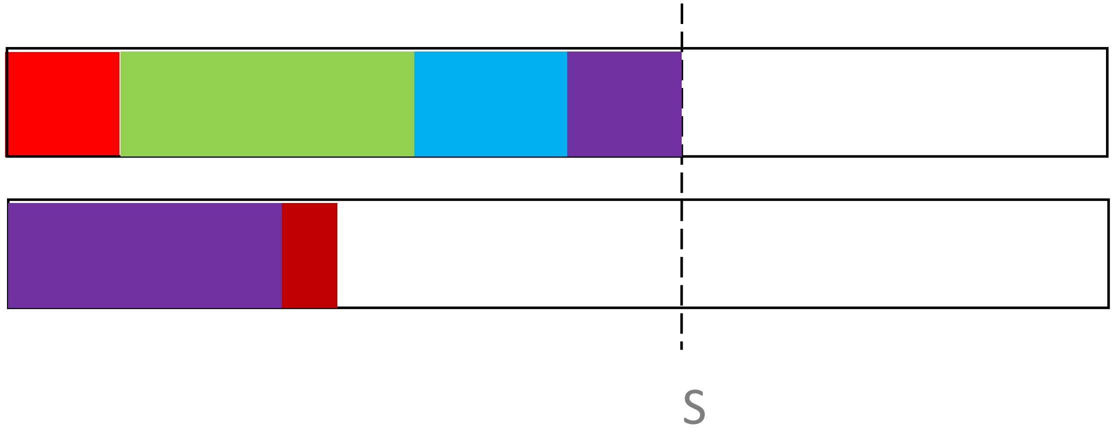
Implementing in Ada

- We will now look briefly how the proposed scheme could be supported in Ada
- First tasks can be defined to embody the code and to be allocated to specific cores

Mixed Criticality

Cyclic Executive

- S is the point of mode change



Implementing in Ada

- So the task that must migrate needs to use a timer that will signal when the movement must occur
- A barrier is used to coordinate the movement between modes
- Timing Events are used to switch between minor cycles

Conclusions

- Cyclic Executives are a common means of implementing high-integrity systems
- In this paper we show how to extend this approach to
  - Multi-core
  - Mixed-criticality
- We believe that the approach can be realised with Ada
  - But not Ravenscar, as task migration is required