# Investigating Mixed Criticality Cyclic Executive Schedule Generation

Tom Fleming
Department of Computer Science,
University of York, UK.
Email: tdf506@york.ac.uk

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

## Introduction

In this work we consider the construction of static Cyclic Executive schedules. Cyclic Executives are widely used and make for an easily certifiable Mixed Criticality platform.

This talk:

- The MC Cyclic Executive.
- Prior Work.
- Multiple Minor Cycles.
- Heuristic approaches vs Integer Linear Programming.
- Construction of our ILP model.
- Evaluating performance and scalability
- Conclusions

## The Mixed Criticality Cyclic Executive

Relies on pre-computed schedules.

A focus on multi-core architectures.

Barrier Protocol used to separate criticality levels.

- High criticality work is executed first.
- All high criticality work has completed on all CPU cores.
- Low criticality work may commence.

## Prior Work

Prior work focused on the simple single cycle case.

Where: Minor Cycle = Major Cycle.

Considered Heuristic approaches to allocation: FF, WF, FFBB.

Investigated the effect of synchronising the criticality change across all CPU cores.

## Extending the Model to Multiple Minor Cycles

We extend the system model to consider multiple minor cycles, specifically:

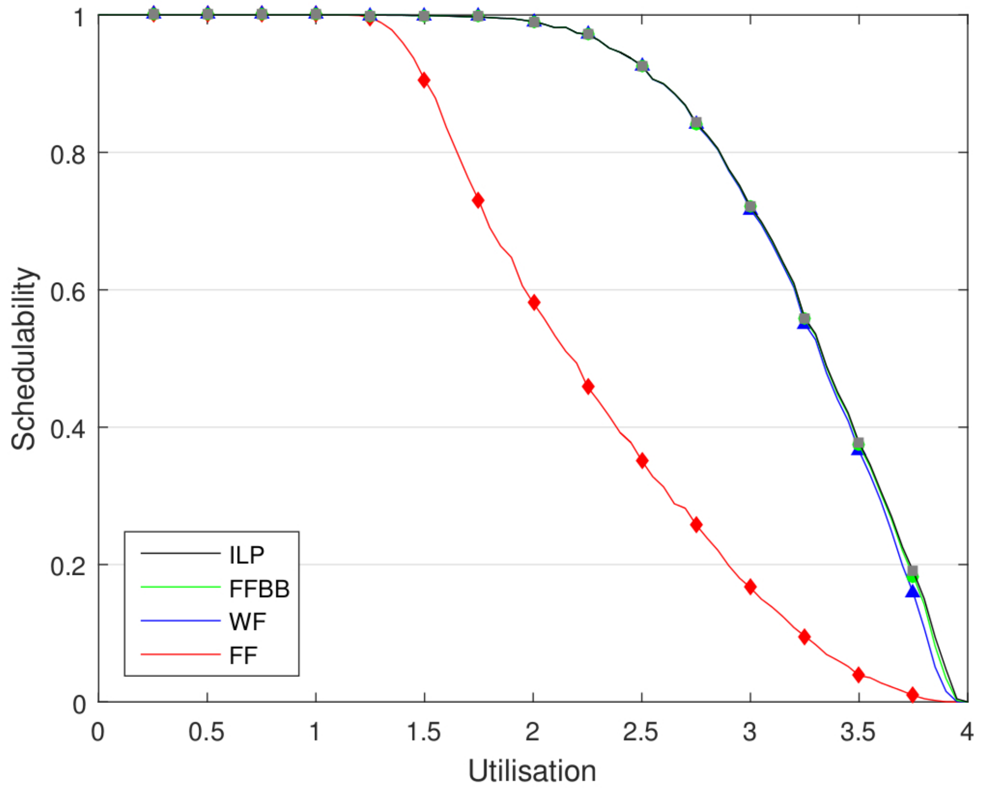
- 4 cycles
- Minor cycle = 25
- Major cycle = 100

The allocation process is now tasks to minor cycles, then within each minor cycle, tasks to CPU cores.

## Heuristics vs ILP

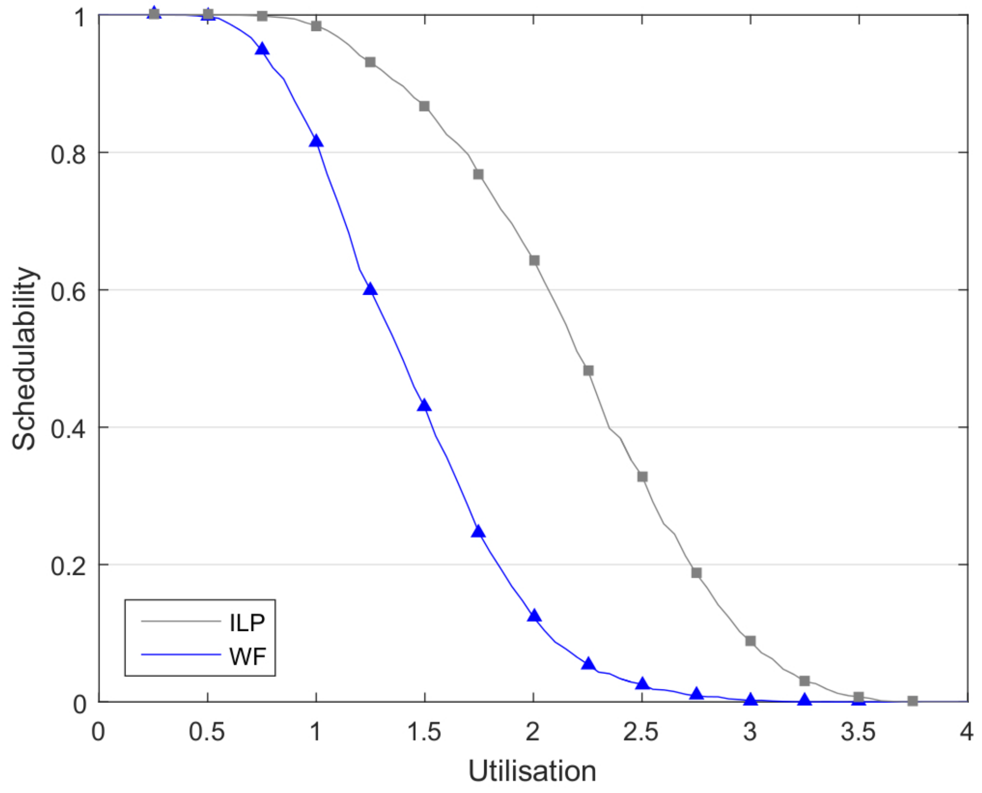
Initially we compared heuristic approaches with an ILP implementation.

For the minor cycle = major cycle case:



## Heuristics vs ILP

For the multi-cycle case:
Minor cycle = 25
Major cycle = 100



## Construction of the ILP model

No objective statement, we are investigating whether a feasible area exists, not optimising any parameter.

Set of binary variables used to model the location of each task, to indicate which minor cycle/core they are scheduled in.

Variables take the value 1 when scheduled and 0 when not.

T1_11 = T{task number}_{CPU core}{Minor Cycle}

## Construction of the ILP model: Constraints

The constraints are made up of two key groups.

Firstly constraints that ensure tasks are in the correct number of minor cycles, and on only a single core per minor cycle.

For example:
T1_11 + T1_21 = 1

As each variable is binary this statement ensure that on this 2 core system task 1 is only scheduled once in the first minor cycle.

## Construction of the ILP model: Constraints

The second set of constraints ensures that the WCET of the tasks allocated to a particular CPU core may execute before their deadlines.

In our dual criticality example we first ensure the schedulability of the HI criticality work in the HI criticality mode:

4 T1_11 + 5 T2_11 + 6 T3_11 + 15 T4_11 <= 25

The value multiplied by the variable is the WCET of the task, this ensures that in minor cycle 1, on core 1 the workload in the HI criticality mode is schedulable.

## Construction of the ILP model: Constraints

We consider the schedulability of the HI work in the LO mode and account for the remaining execution time.

$3\,T1\_11 + 4\,T2\_11 + 5\,T3\_11 + 13\,T4\_11 + X\_1 <= 25$

Same as before, but using C(LO) WCET values and the addition of X_1, this one variable is used to synchronise the point of change across all cores.

$3\,T1\_21 + 4\,T2\_21 + 5\,T3\_21 + 13\,T4\_21 + X\_1 <= 25$

Core 2 also uses this variable, this represents the time remaining for the LO criticality work to execute.

## Construction of the ILP model: Constraints

Finally we seek to ensure that the LO criticality work is able to execute within the remaining time X_{minor cycle}.

10 T5_11 + 2 T6_11 + 3 T7_11 + 5 T8_11 - X_1 <= 0

Here we use the same technique as before, this time ensuring that the resulting LO execution time fits within X_1. The same is done for core 2:

10 T5_21 + 2 T6_21 + 3 T7_21 + 5 T8_21 -X_1<= 0

## Construction of the ILP model: Bounds and variables

The final part of the model declares the bounds on any variables, in our case only the X values need bounding as they are the only non binary variables (in practice they won't exceed the 25 bound).

$X_1 <= 25$
$X_2 <= 25$
$X_3 <= 25$
$X_4 <= 25$

Finally all of the task variables are declared as binary values and the X variables as integers.

## Observed Execution Time

We observed the execution time of our ILP model over 1000s of runs.

We observed a low average execution time off 0.0125 (seconds).

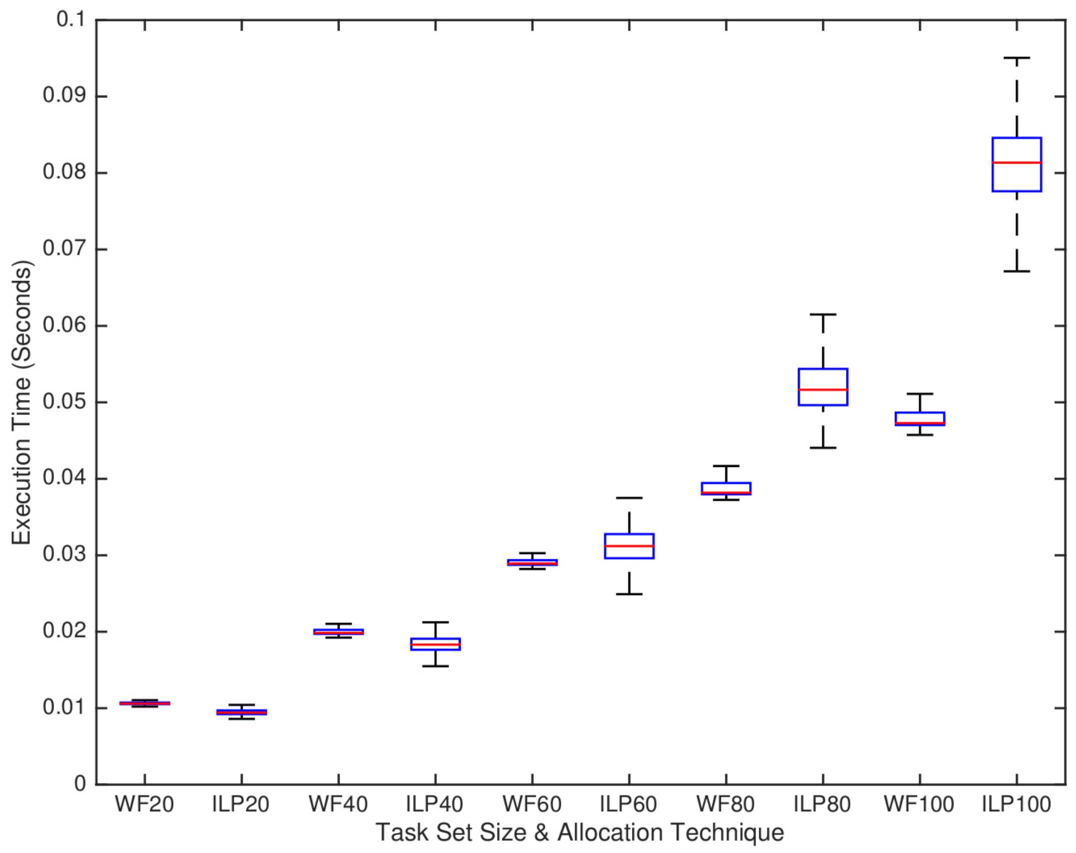
This was achieved through Matlabs timing tools, while not perfect provided a comparison to the heuristic WF.

WF had an observed average execution of 0.010 (seconds).

All experiments were based on a 32core platform.

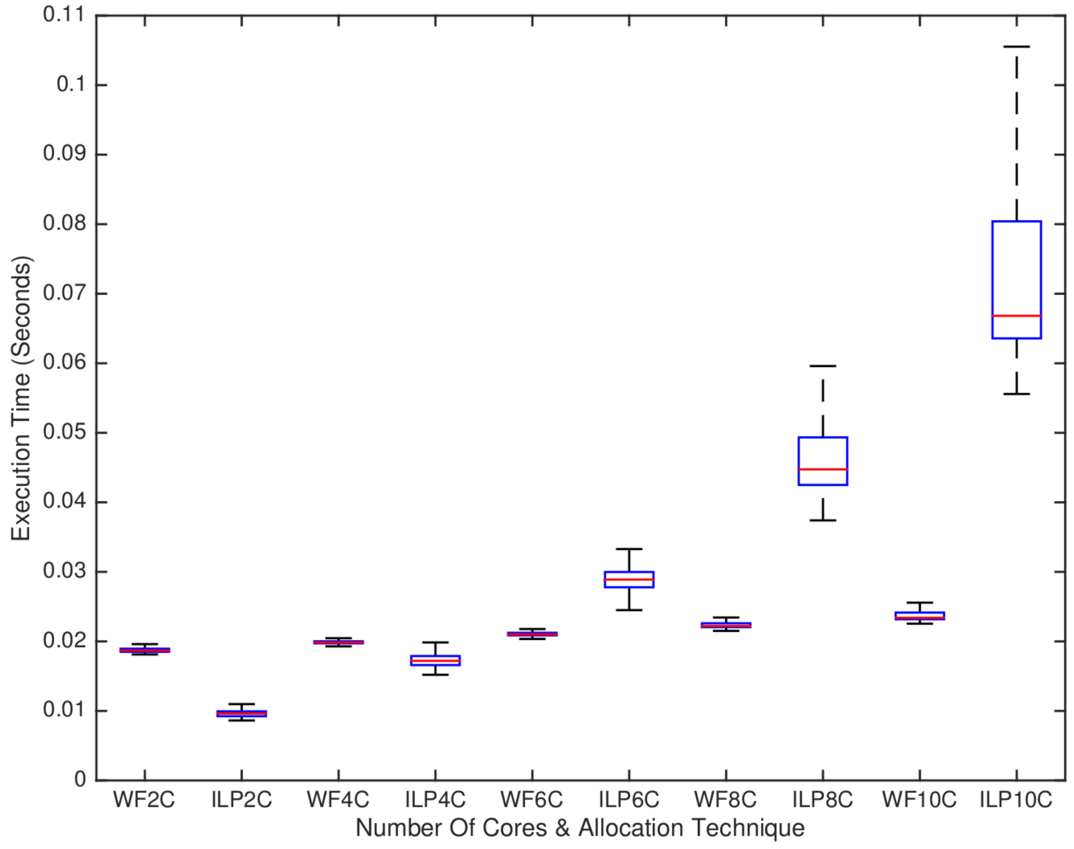
## Observed Execution Time: Scalability

We considered scalability with regard to the number of tasks:



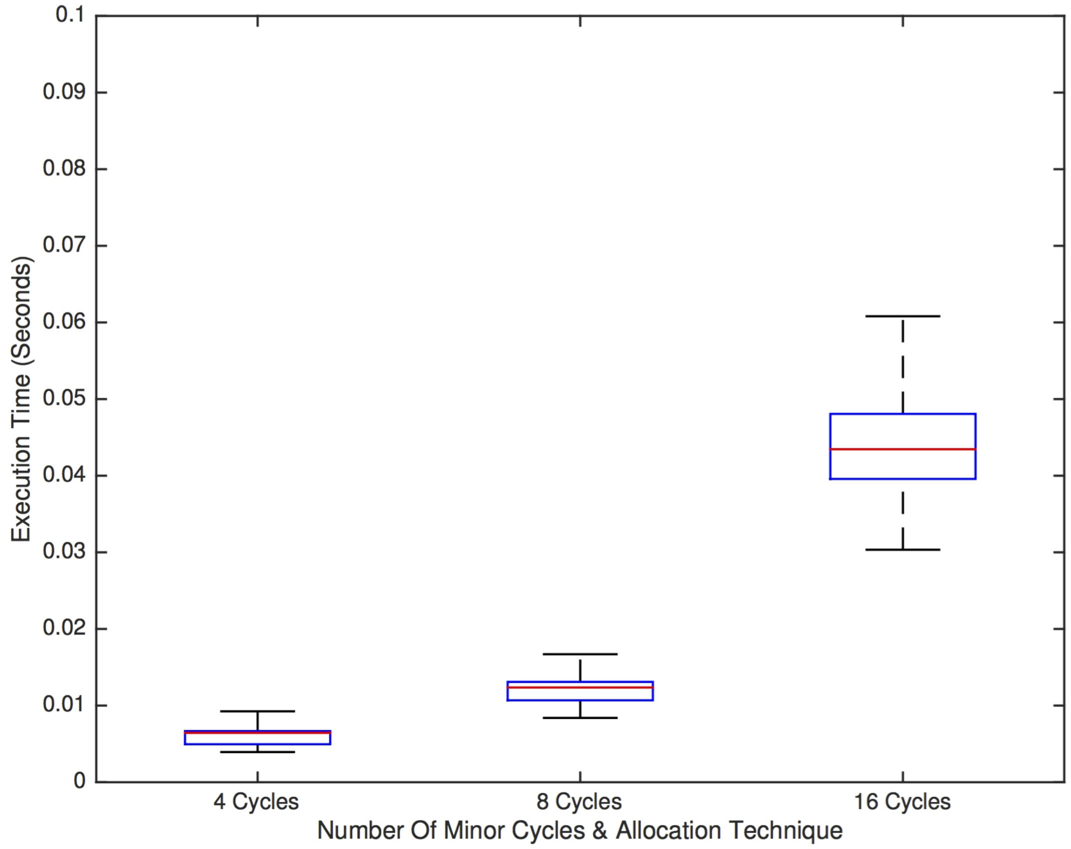
## Observed Execution Time: Scalability

Scalability via an increase in the number of CPU cores:



## Observed Execution Time: Scalability

And scalability via an increase in the number of minor cycles cores:



## Conclusions

We extended the model to include multiple minor cycles.

Created an ILP model to improve upon the poor performance of heuristic approaches using the new model.

For the simple case the heuristics provide a very close approximation of the ILP solution.

Our ILP implementation executed within very reasonable time frame.

The ILP model was shown to be scalable for the number of CPU cores and tasks in a system.

## Thank You!

Questions?