Task Scheduling on Heterogenous Processor

Introduction

ARM's innovative big.LITTLE architecture presents a solution to the challenge of optimizing power consumption in smartphones. This architecture achieves the balancing act of selecting the most appropriate core type based on the current processing load and performance demands. It seamlessly integrates high- performance Cortex-A15 cores with power-efficient A7 cores. To facilitate this intricate performance, big. LITTLE relies on interfaces such as the GIC-400 to coordinate tasks with precision. Meanwhile, the CCI-400 serves as the information highway, ensuring a swift and seamless flow of data between cores. In real-time scheduling, the Earliest Deadline First (EDF) algorithm optimally prioritizes tasks based on their absolute deadlines. It uniquely assigns priorities, with the highest priority given to the task whose deadline is closest in time, ensuring efficient and timely task execution.

Problem Statement

01 Real-Time Task Influx and Efflux

Challenge: Efficiently managing incoming and outgoing real-time tasks in EDF scheduling.
Problem: Incorporating new tasks while optimizing resource allocation and task completion monitoring.

02 Diminishing Pre-emption Expenditure

Challenge: Reducing the costs of interrupting tasks in EDF scheduling.
Problem: Balancing the need for task interruption to meet deadlines with minimizing the performance impact.

03 Perfect Timing and Deadline Management

Challenge: Ensuring precise timing and deadline management in EDF.
Problem: Creating systems to collect and maintain accurate task information for real-time scheduling.

04 Responsive Priority Delegation

Challenge: Creating a priority-based system that responds to approaching deadlines.
Problem: Developing mechanisms to prioritize tasks with imminent deadlines for timely execution.

Objectives

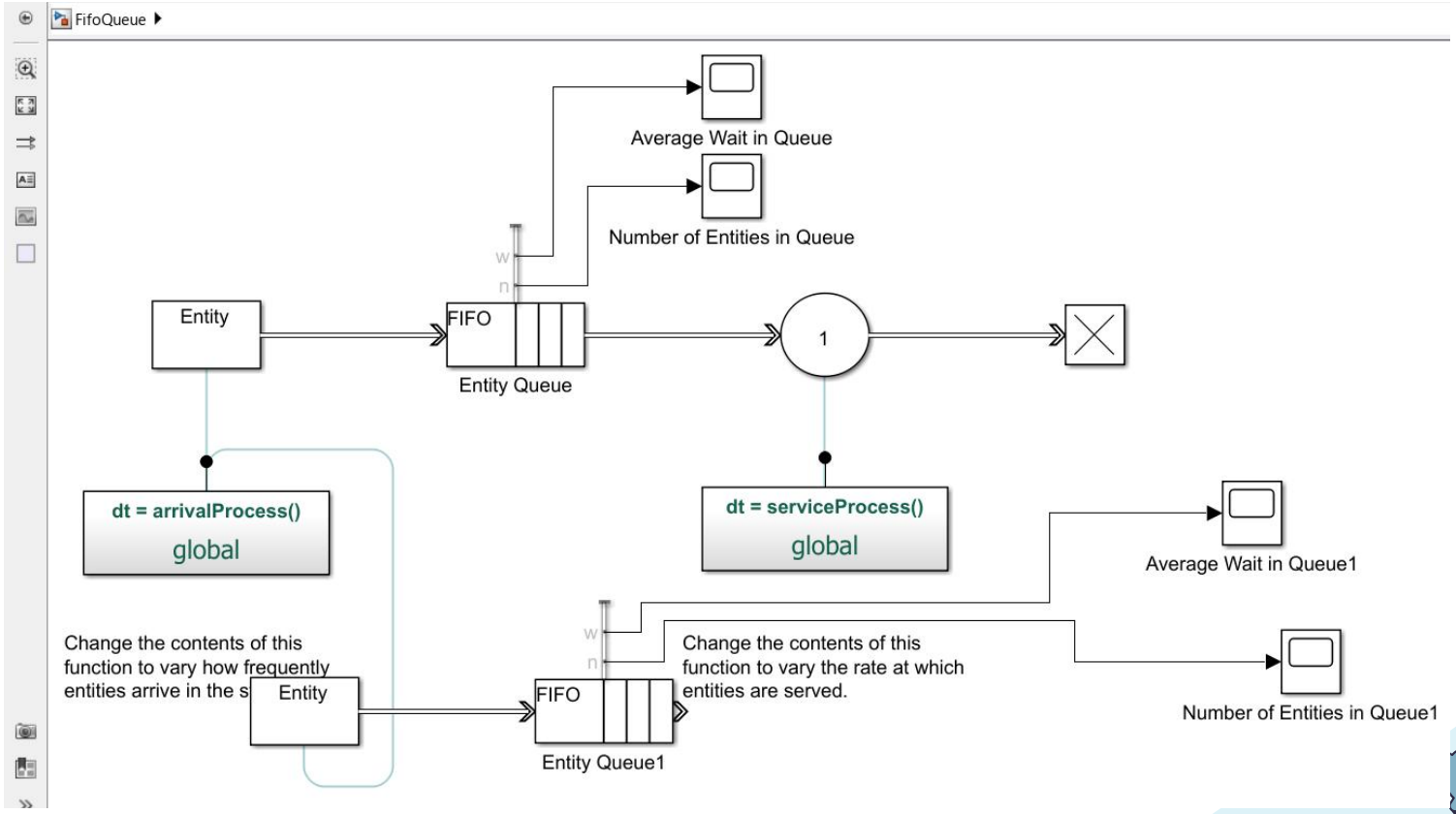
1. Performance Optimization: Improve the performance of real-time processing tasks by efficiently utilizing the capabilities of both high-performance and power-efficient cores in heterogeneous processors.

2. Power Efficiency: Focus on optimizing power consumption during real-time processing to ensure longer battery life and reduced energy usage, especially in mobile and battery-powered devices.

3. Task Scheduling: Develop or improve task scheduling algorithms to dynamically assign real-time and non-real-time tasks to appropriate cores, ensuring that real-time tasks are given higher priority.

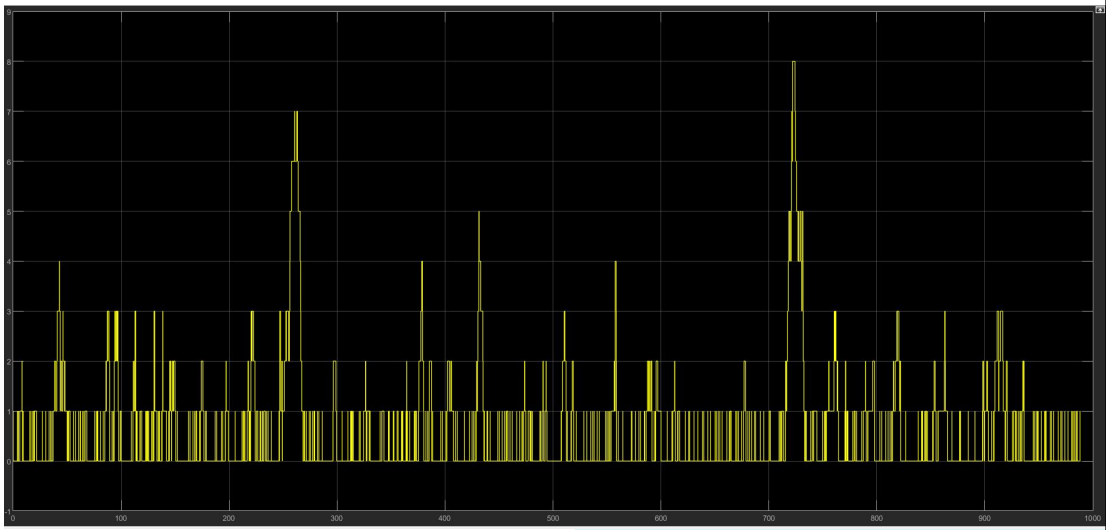
4. Latency Reduction: Minimize processing and communication latencies to meet stringent real-time requirements, such as those in robotics, autonomous vehicles, or industrial control systems.

Working & Methodology



Average Wait in Queue

Number of Entities in Queue

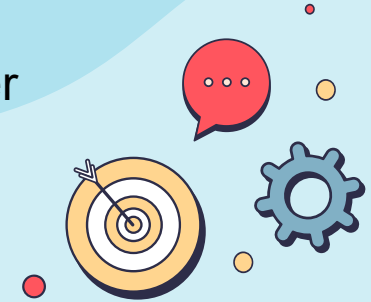


Expected Results

SCHEDULING POLICY AND THEIR PERFORMANCE

Comparision of Scheduling policies for embedded system which is to be executing the real time system. We have executed two policies like shorter service time first and longer service time first. These policies are then compared with the First In First Out policy which is very common. The camparision also gives us better understanding of how a scheduling policy can influence the execution of Real time systems performance.

From seek the above observations it is very clear that the Shorter time first policy yields the better results.



Following table1 shows the details for FIFO.

Time	10	20	30	40
Average Wait time	0.7631	2.764	3.973	3.518
Average Queue Length	0.9224	3.467	4.041	3.537

Following table2 shows the details for Shorter Service Time First.

Time	10	20	30	40
Average Wait time	0.6093	1.395	2.152	2.016
Average Queue Length	0.9768	2.306	2.299	2.027

Following table3 shows the details for Longer Service Time First.

Time	10	20	30	40
Average wait time	1.073	2.174	4.715	4.761
Average Queue Length	1.192	4.44	5.912	6.408

Conclusion

ARM's big.LITTLE technology offers significant power savings of up to 75% in low to moderate performance scenarios and can boost performance by 40% in highly threaded workloads. This energy efficiency is achieved through dynamic task scheduling, where high-performance tasks are assigned to cortex-A15 cores, while low-load tasks are handled by cortex-A7 cores. This approach optimizes power consumption and performance to meet user expectations.