Dynamic Scheduling without Real-Time Requirements

Moiz Zaheer Malik
B.Eng. Electronic Engineering
Hochschule Hamm-Lippstadt
Lippstadt, Germany
moiz-zaheer.malik@stud.hshl.de

***Abstract*—Dynamic scheduling plays an important role in improving system performance in situations where real-time requirements are absent. Without the pressure of strict deadlines, schedulers can be used more effectively to achieve secondary goals, such as reducing power consumption, resource utilization, and performance tuning. In the context of High-Level Synthesis (HLS) which comprises scheduling, allocation, and binding, dynamic scheduling enables smarter decisions about when and where operations are executed. While the initial focus of HLS was on ensuring functional correctness, the need for energy efficiency has grown significantly over time. By dynamically adjusting tasks and deciding which parts of the hardware to use based on the nature of the workload, HLS tools become valuable in saving power without slowing down performance. This highlights the importance of dynamic scheduling in systems that do not rely on real-time constraints. In this paper, we explore the role of dynamic scheduling in HLS systems without real-time constraints, focusing on how it can be leveraged to enhance energy efficiency while maintaining performance. We analyze scheduling strategies and demonstrate their impact on power consumption in non real-time scenarios.**

***Index Terms*—dynamic scheduling, high-level synthesis, energy efficiency, non-real-time systems, resource optimization**

I. Introduction

In modern computing systems, scheduling plays a critical role in determining how and when tasks are executed. In real-time systems—such as those used in automotive control or medical devices—schedulers must ensure that tasks meet strict timing deadlines. However, many systems operate without such time-critical constraints. These include general-purpose processors, cloud servers, and design automation tools like High-Level Synthesis (HLS). In these systems, there is an opportunity to use scheduling not just to meet deadlines, but to optimize other important goals such as energy efficiency, performance, and resource utilization. This flexible approach is known as dynamic scheduling without real-time requirements.

Dynamic scheduling refers to the process of making task execution decisions at runtime based on current conditions such as workload characteristics, available system resources, or performance goals. Unlike static scheduling—where decisions are made before the system begins running—dynamic scheduling allows the system to adapt during execution. In the absence of real-time constraints, the scheduler can reorder tasks, adjust resource assignments, or delay execution of lower-priority operations to improve overall efficiency.

A prominent domain where this technique is useful is High-Level Synthesis (HLS). HLS tools convert high-level programming code (e.g., C, C++) into hardware circuits by performing three main steps: scheduling, allocation, and binding. The scheduling phase decides when each operation should be executed. Traditionally, the focus was on making the circuit functionally correct and fast. However, as energy consumption has become a major concern, especially in battery-powered devices and embedded systems, dynamic scheduling has been adapted to reduce switching activity, balance resource usage, and cut power consumption—all without impacting functionality or throughput [2]. In such HLS environments, if real-time deadlines are not a concern, schedulers can make smarter decisions to save power or avoid unnecessary hardware activity.

Dynamic scheduling is also vital in modern processor architecture. One well-known technique is Tomasulo's algorithm, used in many high-performance CPUs. It enables out-of-order execution of instructions to avoid delays caused by data dependencies. This dynamic handling of instruction execution significantly boosts processor throughput and performance [5].

Furthermore, operating systems utilize dynamic priority scheduling to improve responsiveness. In this approach, the priority of tasks changes at runtime depending on system behavior, task waiting time, or resource demand. This prevents issues like starvation (where some tasks never get to run) and ensures fairer use of CPU time [4].

Beyond hardware and system software, dynamic scheduling is increasingly relevant in cloud computing, data centers, and workflow management systems. In these environments, workloads are highly variable, and real-time constraints are rare. Dynamic scheduling allows systems to monitor current usage of CPUs, memory, and power, and then adjust task allocation to maximize efficiency, reduce costs, or balance load across multiple servers [1, 3].

In summary, dynamic scheduling without real-time constraints provides a flexible and powerful mechanism to optimize modern systems. Whether in hardware design, operating system scheduling, or cloud resource management, it enables systems to adapt on the fly, improve efficiency, and meet

multiple optimization goals beyond just timing. This report explores the theory, techniques, and applications of dynamic scheduling in such environments, demonstrating its growing value in both hardware and software domains.

II. Ease of Use

III.

IV.

References

[1] H. Aydin et al. "Power-aware scheduling for periodic real-time tasks". In: *IEEE Transactions on Computers* 53.5 (2004), pp. 584–600.

[2] R. Dick and N. Jha. "MOGAC: A multiobjective genetic algorithm for hardware-software co-synthesis of distributed embedded systems". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 17.10 (1998), pp. 920–935.

[3] J. L. Hellerstein et al. *Feedback Control of Computing Systems*. Hoboken, NJ, USA: Wiley, 2004.

[4] A. Silberschatz, P. B. Galvin, and G. Gagne. *Operating System Concepts*. 10th. Hoboken, NJ, USA: Wiley, 2018.

[5] R. M. Tomasulo. "An efficient algorithm for exploiting multiple arithmetic units". In: *IBM Journal of Research and Development* 11.1 (1967), pp. 25–33.