Combining Dataflow Applications and Real-time Task Sets on Multi-core Platforms

Hazem Ismail Ali, Benny Akesson, Luís Miguel Pinho
CISTER Research Centre/INESC-TEC
Polytechnic Institute of Porto, Portugal
{haali,kbake,lmp}@isep.ipp.pt

ABSTRACT

Future real-time embedded systems will increasingly incorporate mixed application models with timing constraints running on the same multi-core platform. These application models are dataflow applications with timing constraints and traditional real-time applications modelled as independent arbitrary-deadline tasks. These systems require guarantees that all running applications execute satisfying their timing constraints. Also, to be cost-efficient in terms of design, they require efficient mapping strategies that maximize the use of system resources to reduce the overall cost.

This work proposes an approach to integrate mixed application models (dataflow and traditional real-time applications) with timing requirements on the same multi-core platform. It comprises three main algorithms: **1)** *Slack-Based Merging*, **2)** *Timing Parameter Extraction*, and **3)** *Communication-Aware Mapping*. Together, these three algorithms play a part in allowing mapping and scheduling of mixed application models in embedded real-time systems. The complete approach and the three algorithms presented have been validated through proofs and experimental evaluation.

CCS CONCEPTS

• **Theory of computation → Streaming models**; • **Computer systems organization → Embedded systems**; **Real-time system specification**; • **Software and its engineering → Data flow architectures**;

KEYWORDS

Real-time embedded systems, Synchronous dataflow model, algorithms, multi-core

ACM Reference format:
Hazem Ismail Ali, Benny Akesson, Luís Miguel Pinho. 2017. Combining Dataflow Applications and Real-time Task Sets on Multi-core Platforms. In *Proceedings of SCOPES '17, Sankt Goar, Germany, June 12-14, 2017,* 4 pages.
DOI: http://dx.doi.org/10.1145/3078659.3078671

SCOPES '17, Sankt Goar, Germany

978-1-4503-5039-6/17/06…$15.00
DOI: http://dx.doi.org/10.1145/3078659.3078671

1 INTRODUCTION

We are living the golden age of ubiquitous computing. If we look around, we find ourselves surrounded by computing devices embedded in systems that help or serve us in our daily life. These systems range from simple portable gadgets, e.g. smartphones, cameras, gaming consoles, to large complex systems, e.g. airplanes and cars. These systems are called *embedded systems*. Many embedded systems incorporate multi-core processor architectures for satisfying the increasing demands of their applications, since the need for high processing power at a low power budget is a great concern [11]. The trend of the growing functionality of embedded systems can be demonstrated by the various types of applications that run simultaneously on the system [10]. These applications may have different requirements, such as computational demands or timing constraints. The fact that embedded systems run various applications with different requirements can mean different applications may be represented using different computational models. In such systems, running mixed computational models, guarantees are required to assure stratifying requirements (computational demands or timing constraints) and the correct execution of the system, especially in case of safety-critical applications. A current example of such systems is high-end cars, which may run an advanced multimedia entertainment system (computationally demanding) along with the autonomous driving functions (safety-critical application) that allow self-driving on the highways, i.e. Tesla cars [18].

In this work, our research problem is concerned with embedded systems that incorporate mixed computational models with timing constraints running on the same multi-core platform. These computational models are dataflow with timing constraints and traditional real-time task sets, since they represent a wide range of applications running on top of embedded systems. The traditional real-time applications are modelled as independent tasks. Each task is characterised with specific parameters, e.g. Worst-Case Execution Time (WCET), deadline and period. In contrast, dataflow applications are basically graphs of communicating tasks, which are actors. These actors are defined by a different set of parameters, e.g. WCET and Production/Consumption rates (P/C) of tokens. A dataflow application has timing constraints, i.e. latency and throughput requirements, that must be satisfied. This leads to the main question of this work: *How can future real-time embedded systems safely incorporate mixed application models, dataflow and traditional real-time tasks, with timing constraints onto multi-core platforms, such that their timing constraints are satisfied?*

The rest of this paper is organized as follows. Section 2 provides an overview of related work. Afterwards, Section 3 presents an outline of the proposed solution and the main contributions of

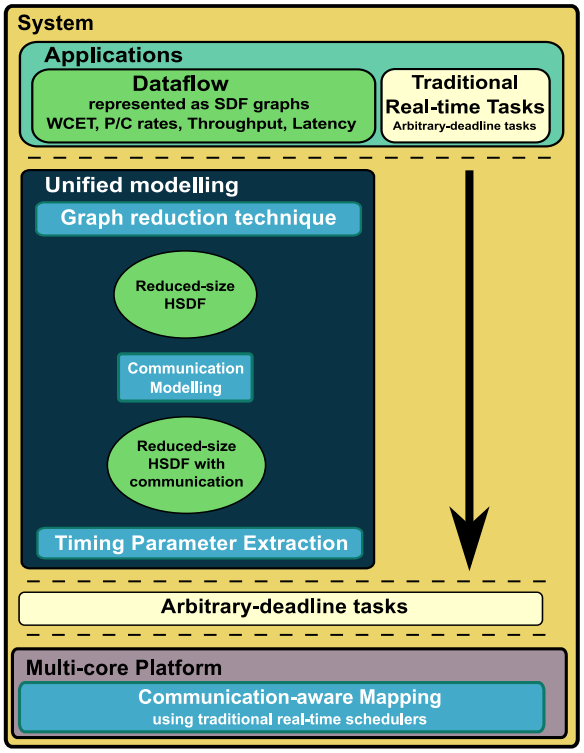


Figure 1: Overview of flow for combining mixed application models with timing constraints on multi-core platforms.

this research work. Then, Section 4 briefly provides experimental results. Finally, we provide some conclusions in Section 5.

2 RELATED WORK

To implement this kind of systems, we have to address how to map and schedule such mixed application models on multi-core platforms. Different solutions for mapping and scheduling have been proposed for each application model independently. The mapping problem has previously been tackled in several works from a high-performance point-of-view, e.g. [8, 14], where all applications are represented either as graphs or independent tasks. However, using these approaches in the mapping of real-time applications does not guarantee satisfying their timing constraints. Another mapping approach uses the First Fit (FF) bin-packing heuristic, since it has been shown to outperform other bin-packing heuristics in terms of achieved throughput [9]. However, applying approaches that satisfy timing constraints and use FF, such as [5], results in over-dimensioned systems, as our experimental evaluation shows in [3]. Moreover, such work [9] does not consider the communication cost and its effect on the schedulability of the system.

The scheduling problem has been studied extensively for traditional real-time applications through introducing several real-time scheduling algorithms either onto uniprocessors, e.g. Fixed Priority (FP) [13], Earliest Deadline First (EDF) [13], or multi-processor, e.g. Partitioned EDF (PEDF) [15] and Hierarchical scheduling [7, 20]. However, dataflow applications mostly use static scheduling, i.e. TDMA. Static scheduling works well in case of systems that only run dataflow applications. In contrast, in systems that run mixed real-time applications, a dynamic real-time scheduling algorithm may have a higher schedulability success rate than static scheduling, but it is not currently available for mixed systems. Furthermore, real-time scheduling algorithms can enable efficient real-time analysis techniques for such mixed systems. Recently, several works scheduled dataflow applications using real-time scheduling algorithms, e.g. [6, 12, 16]. However, they are either limited to dataflow applications represented as Directed Acyclic Graphs (DAG), or they are represented as implicit-deadline tasks.

3 SOLUTION OVERVIEW

In this section, we present an outline of our proposed solution addressing the stated research problem. The goal of this solution is to provide guarantees for the mixed application models executing on a multi-core platform, such that timing constraints are satisfied. The proposed solution comprises three algorithms, which are the main contributions of this work. The following sections give a quick overview on each algorithm.

3.1 Timing Parameter Extraction

The considered system runs two types of application models, traditional real-time and dataflow applications. The main idea is to enable applying real-time schedulers and analysis techniques on dataflow applications to get the same guarantees as real-time tasks. However, these techniques cannot be applied directly on dataflow applications, because they miss the appropriate task model parameters to allow using them. Therefore, a *unified model* for both types of application models is needed. The *unified modelling* is a process that transforms the dataflow applications into traditional real-time tasks using the *Timing Parameter Extraction* (TPE) heuristic shown in Figure 1 and detailed in [2]. The TPE extracts timing parameters, i.e. *offsets*, *periods* and *deadlines*, of dataflow applications with timing constraints, i.e. *throughput* and *latency*, converting them into periodic arbitrary-deadline tasks. These tasks execute in a way that preserve the dependencies of the original dataflow application using the *offset* parameter, while satisfying its timing constraints using the *period* and *deadline* parameters.

3.2 Slack-Based Merging

Before sending the dataflow graph to the timing parameter extraction heuristic, it has to go through a *graph reduction* process called *Slack-Based Merging* proposed in [4]. It is an offline dataflow graph reduction heuristic that generates a *reduced-size HSDF graph* (merged) from the original SDF graph. This is because transformation to HSDF graphs can result in an exponential explosion in the graph size, which slows down the timing parameter extraction algorithm when applied on them. Therefore, the graph reduction process speeds up the overall design process. The generated *reduced-size HSDF graph* satisfies the throughput and latency constraints of the original application graph.

3.3 Communication-Aware Mapping

Before applying the mapping algorithm, a *communication modelling* process is required for the reduced-size HSDF graph before going through the TPE algorithm. It models the communication in the reduced-size HSDF graph, generating an extended HSDF graph that

accounts for the communication cost by transforming them into actors with an initial WCET proportional to the maximum number of hops on the platform. The extended communication-aware graph is then used as input to the TPE algorithm transforming it into a set of independent *arbitrary-deadline tasks*. This enables applying real-time scheduling and analysis techniques during mapping.

The mapping algorithm, shown in Figure 1, allocates the task set on the platform guaranteeing that all applications satisfy their timing constraints. Also, it is *communication-aware*, which means that it considers the communication overhead resulting from the token exchange between different actors in the dataflow applications. The *Communication-Aware Mapping* algorithm [1] is able to do that because of the communication modelling of the HSDF graph that happened in the early stages in the solution, and the ability to update the communication cost according to the current mapping pattern on the platform. This is done by recalculating the WCET of the communication actors based on the actual number of hops they traverse on the NoC, and hence, update the timing parameters of the graphs' actors accordingly. The algorithm is based on a novel mapping heuristic called *Sensitive-Path-First* (SPF), which aims to allocate paths (sequence of actors) in the graph that have the highest impact on its execution and schedulability. It is also able to exploit parallelism in HSDF graphs during mapping. These criteria allow maximizing the usage of the available resources and potentiates parallelism. The SPF is based on a heuristic algorithm for the mapping of real-time dataflow applications called *Critical-Path-First* (CPF) [3].

4 EXPERIMENTAL EVALUATION

In this section, we evaluate our proposed solution using a set of input applications comprises SDF^3 benchmark applications [17]. The SDF^3 benchmark applications are classified into two types: high (greater than 0.5) and low (less than or equal 0.5) total utilization. From these two types, each experiment uses different weights for the random generator to create five sets, of 500 applications each, with a range of (Low % - High %): (90% - 10%), (60% - 40%), (40% - 60%), (20% - 80%), (10% - 90%). Each experiment runs the approach on these five input data sets trying to allocate as many applications as possible on the multi-core platform. To ensure the schedulability of the system, the Quick convergence Processor-demand Analysis (QPA) [19] is used to guarantee the feasibility of the mapped applications. The platform is an 8×8 2D-mesh homogeneous multi-core with a NoC of link capacity equal to 256 Gbps. Each application has a dedicated communication resource called reservation bandwidth $\mathcal{R}$, which is equivalent to a single allocated slot in a TDM frame.

The experimental evaluation shows that the *Slack-Based Merging* [4] decreases the complexity of dataflow applications, and thereby their analysis time. The algorithm reduces the run-time of the approach with 82% to 90% (using merged HSDF), compared to when it is not used (using original HSDF), as shown in Figure 2. However, in terms of number of mapped applications, the original HSDF graphs (complete approach without slack-based merging) enabled our approach to map approximately 12% more applications than the merged ones in all input data sets. This is because the original HSDF graphs contain fine-grained parallelism that the SPF heuristic exploits to efficiently use the platform resources. However, the merged HSDF graphs lose such fine-grained parallelism in the slack-based merging process, decreasing the ability of the SPF heuristic to map applications.

The evaluation of the *Communication-Aware Mapping*, shown in Figure 3, reveals that ignoring communication cost (in case of infinite $\mathcal{R}$), as frequently done in existing work, allows 76% more applications to be mapped (compared to 1% $\mathcal{R}$), although the applications in the system are no longer guaranteed to satisfy their timing constraints. Moreover, it shows that using the SPF heuristic surpasses the well-known FF bin-backing algorithm in terms of number of allocated applications and run-time. In terms of number of allocated applications, Figure 3(a) shows SPF succeeding efficiently in utilizing the computational resources through the allocation of more applications in all input data sets and for different $\mathcal{R}$ values. In case of 5% $\mathcal{R}$, the achieved gain ranges from 4% to 24% (approximately) with an average gain of 12%. In case of 1% $\mathcal{R}$, the achieved gain ranges from 3% to 28% (approximately) with an average gain of 15%. This is due to the selective nature of the SPF heuristic that enables the allocation of actors in the most sensitive paths first that have higher impact on application schedulability. Also, SPF actively enables parallelism to be exploited in each application. In addition, Figure 3(a) illustrates the effect of the communication resources on the number of allocated applications, whatever bin-packing heuristic is used (SPF or FF).

In terms of run-time, the SPF heuristic outperforms FF. As noticed in Figure 3(b), the SPF achieves a lower run-time for most data sets and $\mathcal{R}$ values. The overall achieved gain, in terms of run-time, ranges from 1% to 22% (approximately) with an average of 9%. Also, the results show that the run-time of both heuristics converge for high utilization data sets. The reason for this is that both heuristics struggle similarly to map applications, because high utilization applications consume a lot of resources leaving no space for mapping others. This struggle is illustrated in the rise of run-time with the increase of high utilization applications in the input data sets.

5 CONCLUSIONS

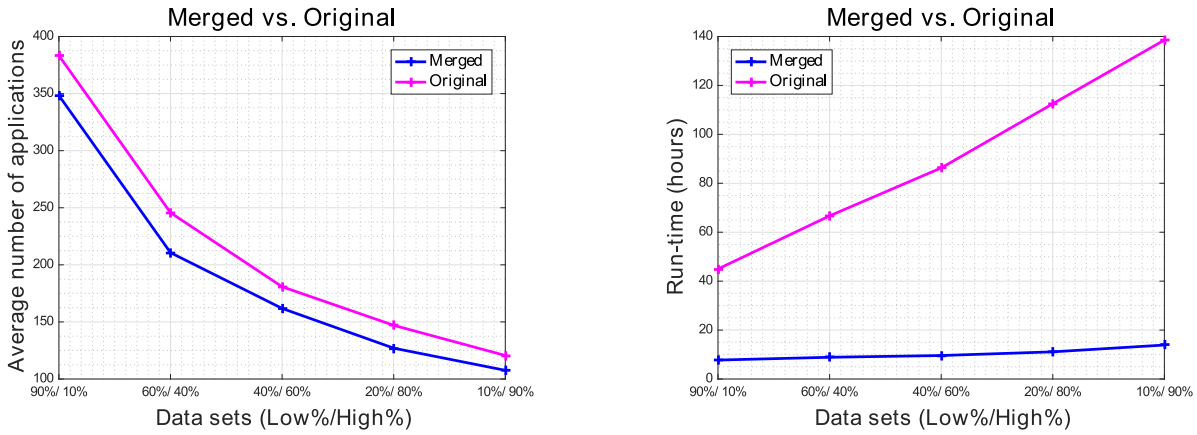
We proposed a complete approach, outlined in Section 3, which combines mixed application models, dataflow and traditional real-time tasks, with timing constraints onto multi-core platforms, such that their timing constraints are satisfied. The proposed approach comprises three main contributions, *Slack-Based Merging*, *Timing Parameter Extraction (TPE)* and *Communication-Aware Mapping*. The experimental evaluation shows that these three important contributions successfully achieve the main goal of this work and play a part in allowing embedded real-time systems to map and schedule mixed application models.

ACKNOWLEDGMENTS

This work was partially supported by National Funds through FCT/MEC (Portuguese Foundation for Science and Technology), under PhD grant SFRH/BD/79872/2011 and co-financed by ERDF (European Regional Development Fund) under the PT2020 Partnership, within the CISTER Research Unit (CEC/04234).

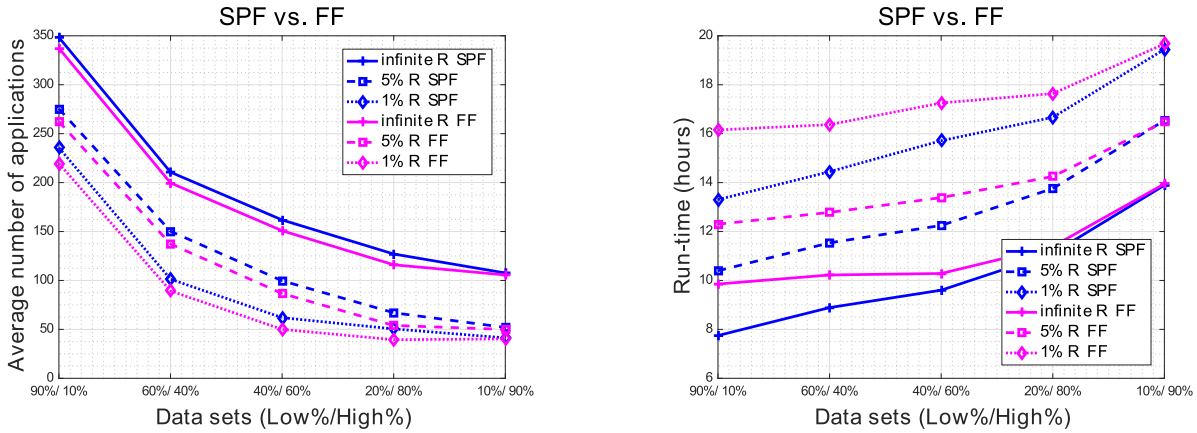
REFERENCES

[1] Hazem Ismail Ali. 2017. *Integrating Dataflow and Non-Dataflow Real-time Application Models on Multi-core Platforms.* Ph.D. Dissertation. Faculdade de Engenharia



(a) Results in terms of average number of mapped applications. **(b) Results in terms of run-time.**

Figure 2: Mapping results for merged and original HSDF graphs.



(a) Results in terms of average number of mapped applications. **(b) Results in terms of run-time.**

Figure 3: Evaluation of the mapping heuristic.

da Universidade do Porto (FEUP). (pending approval).

[2] Hazem Ismail Ali, Benny Akesson, and Luís Miguel Pinho. 2015. Generalized Extraction of Real-Time Parameters for Homogeneous Synchronous Dataflow Graphs. In *Proceedings of the 2015 23rd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*.

[3] Hazem Ismail Ali, Luís Miguel Pinho, and Benny Akesson. 2013. Critical-Path-First based allocation of real-time streaming applications on 2D mesh-type multi-cores. In *2013 IEEE 19th International Conference on Embedded and Real-Time Computing Systems and Applications*. 201–208.

[4] Hazem Ismail Ali, Sander Stuijk, Benny Akesson, and Luís Miguel Pinho. 2017. Reducing the Complexity of Dataflow Graphs Using Slack-Based Merging. *ACM Trans. Des. Autom. Electron. Syst.* 22, 2 (Jan. 2017), 24:1–24:22.

[5] Mohamed Bamakhrama and Todor Stefanov. 2011. Hard-real-time Scheduling of Data-dependent Tasks in Embedded Streaming Applications. In *Proceedings of the Ninth ACM International Conference on Embedded Software*.

[6] Mohamed Bamakhrama and Todor Stefanov. 2012. Managing Latency in Embedded Streaming Applications Under Hard-real-time Scheduling. In *Proceedings of the Eighth IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*.

[7] John M. Calandrino, James H. Anderson, and Dan P. Baumberger. 2007. A Hybrid Real-Time Scheduling Approach for Large-Scale Multicore Platforms. In *Real-Time Systems, 2007. ECRTS '07. 19th Euromicro Conference on*. 247–258.

[8] John D. Evans and Robert R. Kessler. 1992. *A Communication-Ordered Task Graph Allocation Algorithm*. Technical Report. IEEE Transactions on Parallel and Distributed Systems.

[9] Jiani Guo and Laxmi Narayan Bhuyan. 2006. Load Balancing in a Cluster-Based Web Server for Multimedia Applications. *IEEE Trans. Parallel Distrib. Syst.* 17, 11 (Nov. 2006), 1321–1334.

[10] Manuel Jiménez, Rogelio Palomera, and Isidoro Couvertier. 2014. *Introduction to Embedded Systems : Using Microcontrollers and the MSP430* (1 ed.). Springer-Verlag New York.

[11] Minsoo Kim, Joonho Song, Dohyung Kim, and Shihwa Lee. 2010. H.264 decoder on embedded dual core with dynamically load-balanced functional paritioning. In *Image Processing (ICIP), 2010 17th IEEE International Conference on*. 3749–3752.

[12] Giuseppe Lipari and Enrico Bini. 2011. On the Problem of Allocating Multicore Resources to Real-Time Task Pipelines. *4th Workshop on Compositional Theory and Technology for Real-Time Embedded Systems (CRTS '11)* (Nov. 2011).

[13] C. L. Liu and James W. Layland. 1973. Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment. *J. ACM* 20, 1 (Jan. 1973), 46–61.

[14] Yi Liu, Xin Zhang, He Li, and Depei Qian. 2007. Allocating Tasks in Multi-core Processor based Parallel System. In *Network and Parallel Computing Workshops, 2007. NPC Workshops. IFIP International Conference on*. 748–753.

[15] José María López, José Luis Díaz, and Daniel F. García. 2004. Utilization Bounds for EDF Scheduling on Real-Time Multiprocessor Systems. *Real-Time Syst.* 28, 1 (Oct. 2004), 39–68.

[16] Abusayeed Saifullah, Kunal Agrawal, Chenyang Lu, and Christopher Gill. 2011. Multi-core Real-Time Scheduling for Generalized Parallel Task Models. (Nov 2011), 217–226.

[17] Sander Stuijk, Marc Geilen, and Twan Basten. 2006. SDF^3: SDF For Free. In *Application of Concurrency to System Design, 2006. ACSD 2006. Sixth International Conference on*. 276–278.

[18] TESLA. 2016. TESLA Motors. (July 2016). Url: https://www.tesla.com/.

[19] Fengxiang Zhang and Alan Burns. 2009. Schedulability Analysis for Real-Time Systems with EDF Scheduling. *IEEE Trans. Comput.* 58, 9 (Sept 2009), 1250–1258.

[20] Haitao Zhu, Steve Goddard, and Matthew B. Dwyer. 2011. Response Time Analysis of Hierarchical Scheduling: The Synchronized Deferrable Servers Approach. In *Real-Time Systems Symposium (RTSS), 2011 IEEE 32nd*. 239–248.