Real-time priority processing on the cell platform

Citation for published version (APA):

Tempelaars, C., Heuvel, van den, M. M. H. P., Bril, R. J., Schiemenz, S., & Hentschel, C. (2011). Real-time priority processing on the cell platform. In *29th International Conference on Consumer Electronics (ICCE 2011, Las Vegas NV, USA, January 9-12, 2011)* (pp. 157-158). Institute of Electrical and Electronics Engineers. <https://doi.org/10.1109/ICCE.2011.5722514>

DOI:

[10.1109/ICCE.2011.5722514](https://doi.org/10.1109/ICCE.2011.5722514)

Document status and date:

Published: 01/01/2011

Document Version:

Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

- A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
- The final author version and the galley proof are versions of the publication after peer review.
- The final published version features the final layout of the paper including the volume, issue and page numbers.

[Link to publication](#)

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:

www.tue.nl/taverne

Take down policy

If you believe that this document breaches copyright please contact us at:

openaccess@tue.nl

providing details and we will investigate your claim.

Real-Time Priority Processing on the Cell Platform

Coen Tempelaars, Martijn M.H.P. van den Heuvel, Reinder J. Bril
Eindhoven University of Technology
Eindhoven, The Netherlands

Stefan Schiemenz, Christian Hentschel
Brandenburg University of Technology
Cottbus, Germany

Abstract—Flexible signal processing on programmable platforms are increasingly important for consumer electronic applications. Scalable video algorithms (SVAs) using the novel principle of priority processing guarantee real-time performance on programmable platforms, even with limited resources. Dynamic resource allocation is required to maximize the overall output quality of independent, competing priority processing algorithms that are executed on a shared platform. In this paper we describe the mapping of a priority processing application on the Cell/B.E. platform. We compare the performance of different implementations for dynamic-resource-allocation mechanisms, and show that priority processing achieves real-time performance.

I. INTRODUCTION

The principle of priority processing provides optimal real-time performance for scalable video algorithms (SVAs) on programmable platforms with limited system resources [1]. According to this principle, SVAs provide their output strictly periodically and processing of images follows a priority order. Hence, important image parts are processed first and less important parts are subsequently processed in a decreasing order of importance. After creation of an initial output by a basic function, processing can be terminated at an arbitrary moment in time, similar to the milestone method [2]. This principle yields the best output for given resources.

To distribute the available resources, i.e. CPU-time, among competing, independent priority processing algorithms (PPAs), a decision scheduler (DS) has been developed [3]. The DS aims at maximizing the total relative progress of the algorithms on a frame-basis. The relative progress of an algorithm is defined as the fraction of the number of already processed blocks and the total number of blocks to be processed in a frame. The DS divides the available resources within a period into fixed-sized quanta, termed *time-slots*, and dynamically allocates these time-slots to the algorithms.

Strategies and mechanisms for dynamic resource allocation have been addressed in [3] and [4], respectively. Moreover, [4] describes how priority processing applications, implemented in MatLab/Simulink, are executed under Microsoft Windows XP on a general-purpose platform. In this paper we describe the mapping of a priority processing application on an embedded platform, i.e. the Cell Broadband Engine. This platform has been chosen, because it is well supported, widely available, and suitable for consumer electronics [5]. Next, we present implementations for dynamic-resource-allocation mechanisms on the Cell. Finally, we evaluate the implementations and show that priority processing achieves real-time performance.

Fig. 1. The mapping of the DS controlling multiple PPAs on the Cell/B.E. platform using only one SPE. The layers from top to bottom are: 1) application layer; 2) operating system layer; and 3) hardware layer. Dotted arrows indicate a logical connection, solid arrows a physical connection.

II. MAPPING AN APPLICATION ON THE CELL

The Cell is a multi-processor platform that offers a general-purpose processor (PowerPC) and several dedicated streaming processors (SPEs). These SPEs are capable of processing *single instruction multiple data* (SIMD) operations. The PowerPC hosts an operating system (OS), i.e. GNU/Linux. No OS is running on the SPEs. Accordingly, the application programmer is responsible for memory management on the SPEs [5]. Fig. 1 shows the mapping of a priority processing application on the Cell, where multiple PPAs share a single SPE.

We ported a PPA implementation of a scalable de-interlacer to an SPE. To achieve real-time performance, we (i) vectorized the code using SIMD operations and (ii) implemented signal processing in parallel with data transfers by applying a double buffering scheme for both input as well as output.

III. DYNAMIC RESOURCE ALLOCATION ON THE CELL

In [4] three mechanisms for dynamic resource allocation have been identified: *preliminary termination*, *resource allocation*, and *monitoring*. The first mechanism is intrinsic for priority processing, i.e. to terminate a PPA at the end of a frame period and skip the remainder of the current frame. The latter two mechanisms are required to distribute the available resources of the SPE among competing, independent PPAs. We consider these mechanisms and their implementations below.

A. Preliminary Termination: On request of the DS a PPA terminates itself within a reasonable amount of time, i.e. in a *cooperative* way. We propose three alternative implementations: 1. per pixel polling, 2. per block polling (a block is a group of 256 pixels), and 3. using software interrupts.

B. Resource Allocation: Because the SPEs run no OS, basic means for resource management are required to make it possible to share the CPU and local memory of the SPE. We therefore implemented a lightweight mechanism for context

TABLE I
OVERHEAD AND LATENCY FOR VARIOUS IMPLEMENTATIONS OF THE
PRELIMINARY TERMINATION MECHANISM

Mechanism	Absolute Overhead	Termination Latency
1. Pixel polling	1.26 ms $\pm$ 0.004 ms	49.3 μ s $\pm$ 0.15 μ s
2. Block polling	0.06 ms $\pm$ 0.004 ms	51.5 μ s $\pm$ 0.79 μ s
3. Software interrupts	0.15 ms $\pm$ 0.004 ms	49.7 μ s $\pm$ 0.19 μ s

switching. Similar to preliminary termination, context switching is performed in a cooperative way, i.e. the DS requests the current PPA to suspend itself and another PPA to resume its execution. A PPA stores its state whenever a milestone is reached [2], i.e. after a block or row has been processed. The state consists of the current function that is running and a representation of the completed data. When a PPA resumes its execution after being switched out, it resumes from its latest stored milestone. To simplify context switching and reduce its lead-time, we partitioned the local memory, i.e. we gave each PPA a fixed memory share.

C. Monitoring: The DS accurately tracks the consumed processor time of a PPA by inspecting the *time base register* of the PowerPC. This register is updated with a frequency of 79.8 MHz. Hence, measurements have an accuracy of 12.5 ns.

IV. EXPERIMENTS AND RESULTS

We implemented a priority processing application on the Cell and tested it using standard sequences from the Video Quality Experts Group (VQEG). We used two scalable de-interlacers, each processing a different sequence, to experiment with competing, independent PPAs on a single SPE. All experiments were repeated 20 times and, where applicable, their results have a 95% confidence interval.

A. Dynamic resource allocation

We measured the overheads and latencies for the three different implementations of preliminary termination. The results for VQEG sequence 6 are shown in Table I.

For overhead, we used the mean of the 16 median values per frame. Branch hinting [5] keeps the overhead for polling variants low. This reduces the negative effects of polling on the performance during normal execution. Since processing an entire frame takes approximately 50 ms, the relative overheads for block polling and software interrupts are less than 0.5%.

For latency, we used the median of three consecutive runs per frame to filter out exceptionally high latencies. The communication between PowerPC and SPE takes on average 49.1 μ s. This results in relatively high termination latencies for all three implementations. Based on the results in Table I, we used software interrupts for both preliminary termination and context switching.

B. Real-time performance

We performed experiments with a single PPA and with two PPAs to determine whether priority processing can achieve real-time performance on an SPE. Table II shows that the non-scalable part, applied on different VQEG sequences, completes within 16 ms. This makes it possible to simultaneously run two

Fig. 2. Average progress achieved for a frame period of 40 ms by: *top*) one PPA running in isolation, which processes VQEG sequence 6, and *bottom*) two competing PPAs, which process VQEG sequence 5 (grey) and 6 (black). Completion of just the basic function of a PPA is defined as 0% progress, whereas completion of all optional parts is defined as 100% progress.

PPAs within a frame period of 40 ms. Fig. 2 illustrates the progress of a single PPA which is approximately 35%, and the progress of two PPAs, each reaching approximately 5%.

TABLE II
AVERAGE EXECUTION TIMES OF THE NON-SCALABLE PART OF THE
SCALABLE DE-INTERLACER.

VQEG 3	VQEG 5	VQEG 6
15.873 $\pm$ 0.021 ms	15.871 $\pm$ 0.018 ms	15.895 $\pm$ 0.027 ms

V. CONCLUSION

We presented real-time priority processing on a consumer platform. We described the mapping of a priority processing application on a Cell/B.E. and implementations for dynamic-resource-management mechanisms. The PowerPC executes the DS and a single SPE runs the competing, independent PPAs. Because no OS is running on the SPEs, we presented a lightweight mechanism for context switching. Based on our evaluation, we used software interrupts for cooperative preliminary termination and resource allocation. Finally, we showed that priority processing achieves real-time performance. This makes its concept attractive for consumer electronics.

REFERENCES

- [1] C. Hentschel and S. Schiemenz, "Priority-processing for optimized real-time performance with limited processing resources," in *Proc. Int. Conf. on Consumer Electronics*, Jan. 2008.
- [2] N. C. Audsley, A. Burns, R. I. Davis, and A. J. Wellings, *Integrating unbounded software components into hard real-time systems*, ser. Imprecise and Approximate Computation. Kluwer Ac., 1995, ch. 5, pp. 63 – 86.
- [3] S. Schiemenz, "Echtzeitsteuerung von skalierbaren priority-processing algorithmen," in *Tagungsband ITG Fachtagung - Elektronische Medien*, Mar. 2009, pp. 108–113.
- [4] M. M. H. P. van den Heuvel, R. J. Bril, S. Schiemenz, and C. Hentschel, "Dynamic resource allocation for real-time priority processing applications," *Trans. on Consumer Electronics*, vol. 56, no. 2, May 2010.
- [5] M. Scarpino, *Programming the Cell Processor: For Games, Graphics, and Computation*. Upper Saddle River, NJ: Prentice-Hall, 2008.