IEICE Electronics Express, Vol.7, No.24, 1766–1772

Reconfiguration for Sensitivity Technique: A QoS-aware Co-Design approach for stream-based applications

Olufemi Adeluyi and Jeong-A Lee[a)]

Department of Computer Engineering, Chosun University

375 Seosuk-Dong, Dong-Gu Gwangju 501-759, Korea

a) *jalee@chosun.ac.kr*

Abstract: Reconfiguration for Sensitivity Technique (RST) is presented as an approach for maintaining QoS in stream-based applications running on embedded systems. It is a co-design approach that uses the *reconfiguration* and *sensitivity* metrics for QoS awareness; the former for run-time reconfiguration in response to stimuli from the latter. The effectiveness of RST was tested after we used our CHARMS algorithm to screen out 95.17% of the mapping-cases for a H.263 encoder. Our tests showed a 99.25% QoS provisioning-up-time-level at any given instant, based on available battery power, using RST as compared to 80.62% with Performance Aware Reconfiguration of Software Systems(PARSY), 58.83% with Reconfigurable Service Composition(RSC) and 41.67% without RST.

Keywords: co-design, reconfigurable systems, stream-based, QoS

Classification: Integrated circuits

References

[1] M. Danelutto, M. Vanneschi, and C. Zoccolo, “A Performance Model for Stream-based Computations,” *Proc. 15th EUROMICRO International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, Napoli, Italy, pp. 91–96, Feb. 2007.

[2] A. C. J. Kienhuis, *Design Space Exploration of Stream-based Dataflow Architectures: Methods and Tools*, Ph.D Thesis, Delft University of Technology, The Netherlands, Jan. 1999.

[3] M. Marzolla and R. Mirandola, “Performance Aware Reconfiguration of Software Systems,” *Proc. 7th European Performance Engineering Workshop (EPEW 2010)*, Bertinoro, Italy, Sept. 2010.

[4] E. Park and H. Shin, “Reconfigurable Service Composition and Categorization for Power-Aware Mobile Computing,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 19, no. 11, pp. 1553–1564, Nov. 2008.

[5] S.-Y. Ahn, J. Kim, and J.-A. Lee, “Heuristic Algorithm for Reducing Mapping Sets of HW/SW Partitioning in Reconfigurable System,” *Proc. 9th Asia-Pacific Computer Systems Architecture Conference*, pp. 102–114, China, 2004.

© IEICE 2010
DOI: 10.1587/elex.7.1766
Received October 14, 2010
Accepted November 16, 2010
Published December 25, 2010

[6] A. B. Atitallah, P. Kadionik, F. Ghozzi, P. Nouel, N. Masmoudi, and H. Levi, "An FPGA implementation of HW/SW codesign architecture for H.263 video coding," *AEU – International Journal of Electronics and Communications*, pp. 605–620, Dec. 2006.

1 Introduction

Many of today's embedded computing applications are stream-based. These are applications described by task-flow graphs whose nodes represent computations and whose edges represent communications or synchronizations [1]. They describe dynamic applications with nodal data-dependencies that are resolved at runtime [2] and require processing power in the order of several dozens of RISC (Reduced Instruction Set Computer) -like operations with data rates ranging from million to billion samples per second.

In this letter we propose hardware/software (HW/SW) co-design as a means of designing high performance systems within the confines of resource constraint realities. In co-design, applications, made up of operation blocks, with their associated tasks, can be implemented either fully as hardware, fully as software or a combination of the two. A partitioning is done for a performance-cost optimal implementation that leverages on the high performance of hardware and the flexibility of software.

We present designers with a new approach, called RST, for guaranteeing the QoS in stream-based applications. RST involves the choice of two metrics (reconfiguration and sensitivity) and it involves the selection and storage of optimal mapping sets for use at runtime. This selection is made after the creation, testing and streamlining of several mapping-sets, generated by partitioning the application (H.263-encoder – Fig. 1 (a)) into the software (NIOS-II-Softcore-Processor) and hardware (FPGA: Altera-Stratix-II-EP2S60) architectures. The mapping-sets are then streamlined by our Customized Heuristic Algorithm for Reducing Mapping Sets (CHARMS) using the combo-metric cost as the streamlining parameter and the best mapping-sets are those satisfying the pre-specified combo-metric performance levels, which are also indicative of the QoS levels. They are then selected for different levels of the sensitivity metric for runtime reconfiguration. By definition the sensitivity metric is the more critical of the two, as its variations have a greater impact on system performance, while the reconfiguration metric is the parameter which has latitude for change to compensate for the fluctuations in the sensitivity metric.

Marzolla and Mirandolla [3] used a Performance Aware Reconfiguration of Software Systems (PARSY) for maintaining QoS. It involves the selective upgrading/degrading of the runtime configuration whenever the *utility* values exceed or fall below the pre-specified thresholds. Similarly, Park and Shin [4] proposed Reconfiguration Service Composition (RSC) as a power-aware approach for controlling QoS. They dynamically modified the QoS levels in response to power-profile changes by choosing a service-specific map-

DOI: 10.1587/elex.7.1766
Received October 14, 2010
Accepted November 16, 2010
Published December 25, 2010

ping option based on a predefined *Precedence-Index (PI)*. PARSY and RSC are compared with RST in section 3.

2 Y-Chart, Combo-Metrics, CHARMS and RST

The *Y-chart process* (Fig. 1 (b)) allows designers to map target application processing elements (PEs) to HW/SW Functional Elements (FEs). This is done in order to carry out an analysis and provide performance numbers used to rank the mapping-cases. These performance numbers are approximated based on available parameters from the profiling process. The task-flow graph of the H.263 encoder application is used within the Y-chart process, where mapping cases are evaluated. The Iterative Process feedback arrow allows the system to select and evaluate new mapping cases with different architecture descriptions until the designer's requirements are met or until all cases are exhausted.

The *Customized Heuristic Algorithm for Reducing Mapping Sets* (CHARMS) algorithm is a modification of the Heuristic Algorithm for Reducing Mapping Sets (HARMS) proposed by Ahn et al [5]. The HARMS streamlines the mapping cases by first excluding impossible mapping cases based on the FPGA size constraint and, as shown in Eq. (1a), it then uses the values of workload and parallelism to estimate the throughput as a basis for reducing the mapping-sets. CHARMS is a more flexible version of HARMS that takes the varying computational complexities of the task nodes into account, making them similar to one another. A formal algorithm for CHARMS is shown in Fig. 1 (c).

$$IF \quad W1 > W2 \quad \& \quad P1 < P2, \quad then \quad T1 < T2 \tag{1a}$$

$$[combo\_metrics] == \begin{bmatrix} c_1 \\ . \\ . \\ . \\ c_n \end{bmatrix} = \frac{1}{no\_of\_metrics}[metrics][weights]$$

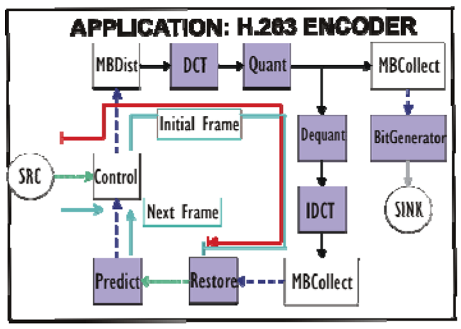
$$= \frac{1}{b}\begin{bmatrix} m_{11} \dots m_{1b} \\ \dots\dots\dots \\ \dots\dots\dots \\ \dots\dots\dots \\ m_{n1} \dots m_{nb} \end{bmatrix}\begin{bmatrix} w_1 \\ . \\ . \\ . \\ w_b \end{bmatrix} \tag{1b}$$

where W, P and T are workload, parallelism and throughput respectively. Also w_i is the weight assigned to metric i, c_j is the combo-metric for the node j in the application and m_{nb} is the bth metric value for the nth PE node in the FE under consideration.

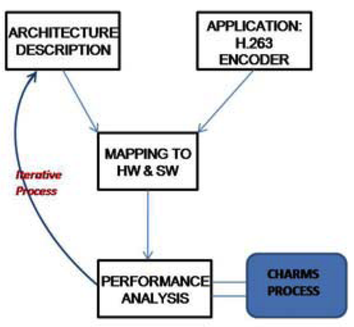
We have opted to use *Combo-Metrics*- weighted sum of contributing metrics, as this gives a more holistic view than is obtained with just one metric. For our discussions we have assigned weights of "3" and "1" to the sensitivity and reconfiguration metrics respectively. The combo-metrics are calculated

© IEICE 2010
DOI: 10.1587/elex.7.1766
Received October 14, 2010
Accepted November 16, 2010
Published December 25, 2010

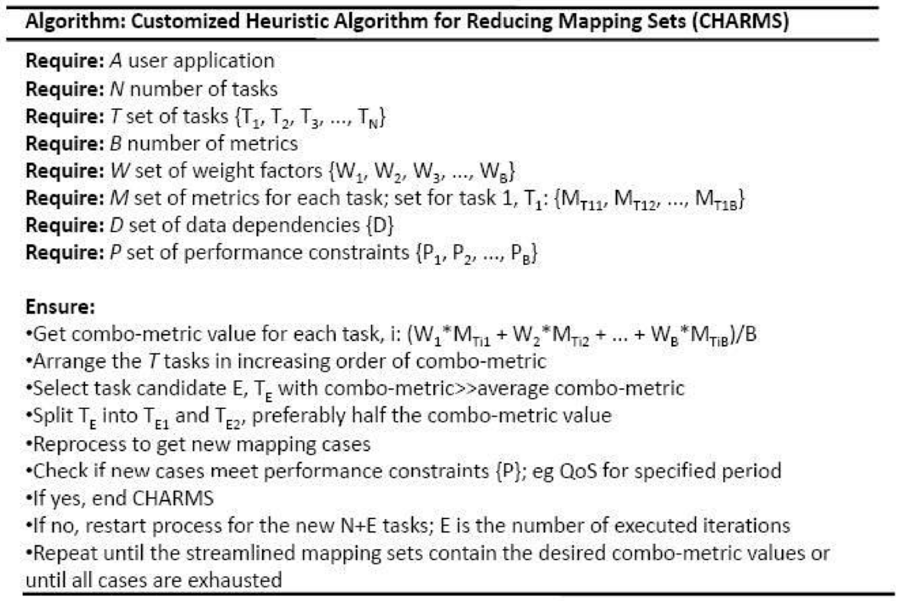
for each of the application processing elements for both the software and hardware FEs as shown in the matrix equations in Eq. (1b). The nodal combo-metrics for each given mapping case are summed in each mapping-set in order to index them based on the combo-metrics.



(1a)



(1b)



```
Algorithm: Customized Heuristic Algorithm for Reducing Mapping Sets (CHARMS)

Require: A user application
Require: N number of tasks
Require: T set of tasks {$T_1$, $T_2$, $T_3$, ..., $T_N$}
Require: B number of metrics
Require: W set of weight factors {$W_1$, $W_2$, $W_3$, ..., $W_B$}
Require: M set of metrics for each task; set for task 1, $T_1$: {$M_{T11}$, $M_{T12}$, ..., $M_{T1B}$}
Require: D set of data dependencies {D}
Require: P set of performance constraints {$P_1$, $P_2$, ..., $P_B$}

Ensure:
•Get combo-metric value for each task, i: $(W_1*M_{Ti1} + W_2*M_{Ti2} + ... + W_B*M_{TiB})/B$
•Arrange the T tasks in increasing order of combo-metric
•Select task candidate E, $T_E$ with combo-metric>>average combo-metric
•Split $T_E$ into $T_{E1}$ and $T_{E2}$, preferably half the combo-metric value
•Reprocess to get new mapping cases
•Check if new cases meet performance constraints {P}; eg QoS for specified period
•If yes, end CHARMS
•If no, restart process for the new N+E tasks; E is the number of executed iterations
•Repeat until the streamlined mapping sets contain the desired combo-metric values or
until all cases are exhausted
```

(1c)

Fig. 1. (a) Task-Flow Graph for the H.263 Encoder Application (b) Y-chart process for RST (c) The CHARMS Algorithm

The *Reconfiguration for Sensitivity Technique (RST)* is a co-design technique that works with the other methods listed in this section to execute applications in HW/SW systems so as to meet the performance level specified by the designer. Prior to live execution of the application, a number of mapping cases are generated from potential partitions to the hardware and software systems using the Y-chart process and CHARMS. These mapping cases are then arranged based on the value of the combo-metric and are screened based on the threshold combo-metric value determined by the designer. This process identifies optimal mapping-cases, which we call *fall-back mapping-cases* that can be used for QoS-aware reconfiguration during the runtime of the live system. We have used power and throughput as the sensitivity and reconfiguration metrics respectively. However, a designer can

© IEICE 2010
DOI: 10.1587/elex.7.1766
Received October 14, 2010
Accepted November 16, 2010
Published December 25, 2010

choose any sensitivity metric, provided it is the most critical to the design.

3 Performance analysis

The RST approach has been described using the same configuration used by Atitallah et al in [6]. Thus the H.263 encoder, with its processing element nodes, has been chosen as the application while the Stratix-II-EP2S60 FPGA and NIOS-II-softcore processor have been chosen as the hardware and software functional elements respectively. RST has been compared with PARSY and RSC. A features-based comparison of the three techniques is given in Table I.

The percentage of available battery power has been used as a measure for sensitivity in this letter and used to quantify QoS. To analyze the performance of the system with RST and others, we have defined two (2) inflection points; these are points in the runtime where the power profile is expected to begin to experience significant changes.

The first point chosen was the *AC2GoodBattery*, while the second was *GoodBattery2LeakyBattery*. The former refers to a transition period when the portable device has its live AC power disconnected, thus requiring it to rely on a battery that is assumed to be in good working condition. The latter refers to a transition from a battery in good working condition to one that is a faulty and leaky battery. The periods between the runtime start, end and inflection points constituted our test phases. As such, we had three (3) test phases, namely:

i. *Phase I:* From an AC power connected startup to AC2GoodBattery

ii. *Phase II:* From AC2GoodBattery to GoodBattery2LeakyBattery

iii. *Phase III:* From GoodBattery2LeakyBattery until the end of runtime

The sensitivity (power) metric reflects the amount of power consumed in the execution of a nodal task in HW/SW. The reconfiguration (throughput) metric is defined as being dependent on parallelism and workload [5]. The throughput is calculated as the product of the channel size and frequency divided by the number of execution cycles (Eq. (2)). The workload is defined as the volume of tasks to be processed, while the parallelism is calculated as the ratio of the summation of the individual execution times assuming that the tasks are all run sequentially to the execution time as a result of running some tasks in parallel.

$$Throughput \propto \frac{Parallelism}{Workload} \propto \frac{Frequency}{No\_of\_Execution\_Cycles} \tag{2}$$

We utilize the extracted mapping-sets from the previous steps (Y-chart process with the combo-metric-centric CHARMS) to perform a “hot swap” of the active mapping set at the inflection points to one of the fallback mapping-sets in order to compensate for the changes and thus stabilize the QoS parameter.

© IEICE 2010
DOI: 10.1587/elex.7.1766
Received October 14, 2010
Accepted November 16, 2010
Published December 25, 2010

Table I. Comparison between RST, RSC and PARSY

S/No	Type	Key terms	Mode of Operation	Other Comments
1.	RST	Combo-metrics, reconfigurable metric, sensitivity metric, QoS, mapping sets, weights, thresholds	Do a pre-run of application to select fallback options based on combo-metric values before live implementation. At given threshold change to appropriate fallback option	All 3 techniques support run time reconfiguration based on profile changes during run time.
2.	PARSY	Precedence Index (PI), upgrading and degrading of components, upper and lower thresholds, QoS	Assign utility values to components (tasks). Determine upper and lower thresholds and monitor these in live execution. During execution (if thresholds are surpassed) determine which components need to be upgraded or degraded to maintain QoS	RST determines fallback options before the live runtime and thus maintains QoS in a timely manner. Both PARSY and RSC determine how to adapt during the run time and thus lose some time. Both RST and PARSY adapt their solution in order to maintain Qos while RSC changes the QoS (to better or worse) depending on the runtime conditions.
3.	RSC	Utility, service composition, QoS, adaptation	Give each service (eg hardware/software) a PI that indicates expected profitability of choosing a configuration. During live implementation modify QoS based on the current situation.	

Table II. Profile table for H.263 encoder in hardware and software

	Functional Element: Software (NIOS II Softcore Processor)			**Functional Element: Hardware (Altera Stratix II EP2S60 FPGA)**		
Processing Element	*Through-put*	*Power*	*Combo-Metric*	*Through-put*	*Power*	*Combo-Metric*
DCT	936	675	2961	166568	93	166847
IDCT	936	675	2961	166568	93	166847
Quant	25266	20	25326	1874821	8	1874845
Dequant	46789	11	46822	1874821	8	1874845
ME	2339	217	2990	519354	30	519444

A comparison of the techniques and the profile table are shown in Tables I and II respectively. We have used only five(5) of the H.263 processing elements shown in Table II since they account for over 97% of the computational activity [6]. These are the Discrete Cosine Transform (DCT), Inverse DCT (IDCT), Quantization (Quant), Dequantization (Dequant) and Motion Estimation (ME) from Predict & Restore shown in Fig. 1 (a).

Using our CHARMS approach described in section 2, a total of 4096 mapping cases were generated, from which 198 of the best mapping cases were sieved, representing a streamlining of 95.17% of the cases, while retaining 4.83% of the best cases. Three (3) of these cases were selected for our test- the 1st was the mapping set with the *best throughput and worst power*, the

© IEICE 2010
DOI: 10.1587/elex.7.1766
Received October 14, 2010
Accepted November 16, 2010
Published December 25, 2010

2nd had *average throughput and average power*, such that the power never falls below 66% of battery capacity; while the 3rd had the worst *throughput and best power*, in order to lower the power consumption. These 3 cases were stored as the *fallback mapping-sets* used for the Phase I, Phase II and Phase III test phases respectively.

For power analysis, we have chosen the LG Li-Ion 3.7 v M-Commerce battery. It has a capacity of 1050 mAh. We assume that the leaky battery discharges 20% faster than normal. Unlike RST, both RSC and PARSY have an additional overhead as a result of determining the best mapping case at runtime. Similarly, unlike RST and PARSY, RSC changes (in this case it degrades) the initial QoS settings in response to worse runtime operating conditions.

Our implementation was based on a 12-minute test and we experienced an up-time (time above the agreed QoS transition point or threshold of 0.66 of the rated battery capacity) of 99.25% with RST, as opposed to the up-time of 80.62% with PARSY, 58.83% with RSC and 41.67% without RST. As shown in Fig. 2, the QoS level (as typified by the % of available battery power in our illustration) is much more stable with the RST than without it.

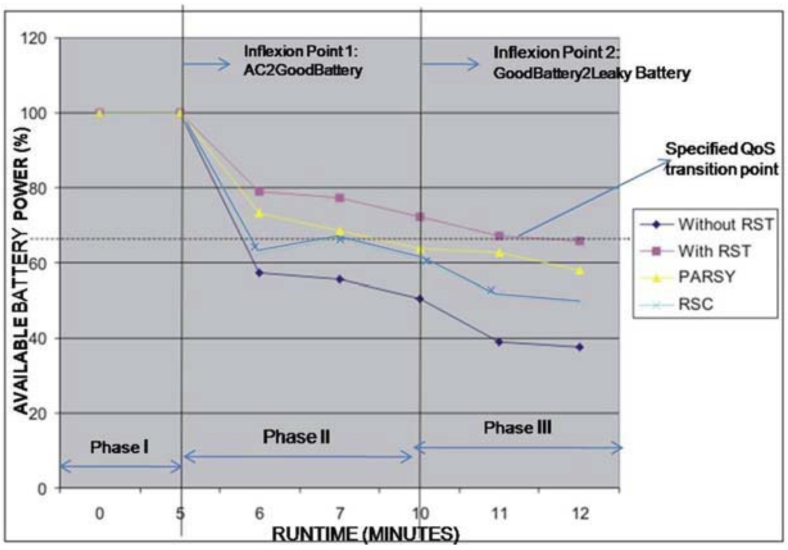


Fig. 2. Effect of RST on runtime implementation of H.263

4 Conclusion

In this paper, we have presented the Reconfiguration for Sensitivity Technique (RST) as a co-design approach that takes advantage of technologies such as FPGAs to provide flexible solutions that can significantly improve QoS guarantees in the runtime environments of stream-based applications.

© IEICE 2010
DOI: 10.1587/elex.7.1766
Received October 14, 2010
Accepted November 16, 2010
Published December 25, 2010